Tentamen i DATABASER

Svar:

Obs! Lärarversion, med lösningar

DAG: 17 December, 2002 **TID:** kl. 14.15 – 18.15 **PLATS:** M-huset

Förfrågningar/ *Questions*:	Niklas Röjemo, ankn. 5423
Resultat/ *Result*:	anslås den/*will be announced* 15 Januari, 2002
Poängantal/ *Points*:	60 poäng.
Betygsgränser/ *Grades*:	CTH: 3:a 24 p., 4:a 36 p., 5:a 48 p.
	GU: G 28 p., VG 48 p.
	Doktorander/*Ph.D.*: G 28 p.
Hjälpmedel/ *Extra*:	Inga/*None*

Observera/*Note:*

- Skriv tydligt och disponera pappret på ett lämpligt sätt.
  Write clearly and avoid writing too much on each page.
- Börja varje uppgift på nytt blad. Skriv endast på en sida av pappret.
  Start every new exercise on a new sheet.
- Alla svar skall **motiveras väl** och ej vara onödigt komplicerade!
  *All answers should be **well motivated** and not unnecessarily complicated!*
- Ange på tentan om du går på GU eller Chalmers och vilken linje/program du går.
  Mark on the exam if you are study at GU or Chalmers.
- Svar kan ges antingen på engelska eller svenska.
  Your answers may be in Swedish or English.

Lycka till! *Good luck!*

Uppgift 1. 3*4p

a) Beskriv fragmentering med uttryck i relationsalgebra, samt ge orsaker till varför man vill använda det.
Describe fragmentation with relation algebra, and explain why it is used.

Svar:

- **Vertikal fragmentering**
  Dela upp en relation i olika projektioner. $\pi_M(r) \bowtie \pi_N(r) = r$
  Vertikal fragmentering används för lokalitet (hemligheter/prestanda) samt för att skapa normalformer.
- **Horisontell fragmentering**
  Dela upp en relation i olika selektioner. $\sigma_v(r) \cup \sigma_{\texttt{not } v}(r) = r$
  Horisontell fragmentering används för utrymme, lokalitet, effektivitet.
- ***Vertical fragmentation***
  Split the relation with projections. $\pi_M(r) \bowtie \pi_N(r) = r$
  Vertical fragmentation is used to localize data (security/performance) and to create normal forms.
- ***Horisontal fragmentation***
  Split the relation with selections. $\sigma_v(r) \cup \sigma_{\texttt{not } v}(r) = r$
  Horisontal fragmentation is used for space, location, efficiency.

b) Vad används NULL för i databaser? Vilka problem kan NULL medföra?
Why are NULL-values used in databases? Give examples of problems that NULL values may cause.

Svar:

NULL-värden användas när det riktiga värdet saknas eller är okänt.

NULL-värden kan ställa till med problem vid join där rader faller bort. Lämplig outer join löser problemet.

Lätt att missa NULL-värden vid val pga 3-värd logik.

NULL-values are used when a real value is either missing or is unkown.

NULL-values can create problems in join operations, some lines are lost. A suitable outer join can solve this problem.

It is easy to miss NULL-values due to the 3-valued logic they introduce.

c) Förklara kvorumkonsensusprotokollet.
Decribe the quorum consensus protocol.

Svar:

Kvorumkonsensusprotokollet är en generalisering av majoritetsprotokollet. Det finns ett läs kvorum Q_r och ett skriv kvorum Q_w så att $Q_r + Q_w > 1$ och $Q_w + Q_w > 1$.

The quorum consensus protocol is a generalized form of the majority protocoll. There is a read quorum Q_r *and a write quorum* Q_w *such as* $Q_r + Q_w > 1$ *and* $Q_w + Q_w > 1$.

Uppgift 2. 4*2p

En transaktion körs på en distribuerad databas så att 4 datorer (dator A,B,C och D) är inblandade. På dator A kör en koordinantor. Beskriv vad som händer när transaktionen skall avslutas under följande scenarier:
A transaction is run on a distributed database in such a way that 4 computers (computer A,B,C, and D) are involved. The coordinator is on computer A. Describe what will happen at the end of the transaction in the following cases:

a) Ingen av datorna upptäcker något lokalt hinder att göra commit.
None of the computers detects any local reasons not to commit.

b) A,B och C kan göra commit, men D vill göra rollback.
A,B, and C can do commit, but D wants to rollback.

c) Samma som i b) men D går sönder innan den hinner besluta om den vill göra rollback eller commit.
Same as b), but D breaks before it can decide if it wants to rollback or commit.

d) Samma som i a) men A går sönder innan den hinner besluta om den vill göra rollback eller commit.
Same as in a) but A breaks before it can decide if it wants to rollback or commit.

Svar: a) B,C och D skickar prepare till A som skickar ut commit till alla.
B,C, and C send prepare to A, A sends commit to all the others.

b) B,C skickar prepare till A, men D skickar abort. A beslutar om abort och meddelar B,C och D.
B, and C send prepare to A, D sends abort to A. A sends abort to all the others.

c) B,C skickar prepare till A. Efter ett tag gör A time-out och skickar abort till alla.
B, and C send prepare to A. After a while A does a time-out and sends abort to all the others.

d) B,C och D skickar prepare till A och väntar sedan på ett svar som aldrig kommer. Protokollet kommer vänta tills koordinatorn startar om igen. När koordinatorn startar om ser den vad som står i dess logfil. Finns det ingen commit där skickas abort till C,D och B som äntligen kan göra rollback. Om det finns en commit i logfilen så skickas commit till de av C,D och B som frågar vad som hänt.
B,C, and C send prepare to A and wait for an answer that never arrives. The protocol waits until the coordinator is restarted. The coordinator uses its log file to decide what to do. If there is no commit in the log file then it sends abort to C,D, and B. If there is a commit in the log file then the coordinator answers with commit if any of the other asks what the outcome was.

Uppgift 3. 8p
Låt $r(A, B, C, D)$ och $s(A, C, E)$ vara tabeller med följande innehåll:
The tables $r(A, B, C, D)$ and $s(A, C, E)$ are given as:

r:

A	B	C	D
Hej	17	10%	2.72
Hej	38	20%	3.14
Ö	2	10%	1.41

s:

A	C	E
Hej	20%	Do
Hej	10%	Re
Ö	30%	Mi
Hej	20%	Fa

Beräkna resultatet av följande relationsalgebra-uttryck!
Evaluate the following expressions!

(a) $\pi_A(\sigma_{C \neq 10\%}(s))$

(b) $(\pi_A(s) \times \pi_C(s)) - \pi_{A,C}(r)$

(c) $r \bowtie s$ (naturlig samkörning/*natural join*)

Svar:

(a, b) $\pi_A(\sigma_{C \neq 10\%}(s))$:

A
Hej
Ö

: **2p.**

$(\pi_A(s) \times \pi_C(s)) - \pi_{A,C}(r)$:

A	C
Hej	30%
Ö	20%
Ö	30%

: **3p.**

(c) $r \bowtie s$:

A	B	C	D	E
Hej	17	10%	2.72	Re
Hej	38	20%	3.14	Do
Hej	38	20%	3.14	Fa

: **3p.**

Kolumnerna kan även vara i annan ordning, t.ex. (A, C, B, D, E).
The columns can be in another order, e.g., (A, C, B, D, E).

Uppgift 4. Givet transaktionen T1:
12p *The transaction T1 is given as:*
B R(a) R(b) W(a) R(c) W(b) C

a) Ge en operationsföljd för operationerna i transaktionenen T1 och en annan transaktion T2, som du själv konstruerat, så att operationsföljden inte är serialiserbar.
Write a sequence of the operations in T1 and another transaction T2, which you define yourself, so that the sequence of operations can not be serialized.

Svar: En möjlig transaktion är:
One possible transaction is:

T2: B W(a) C

Schedule

```
T1: B R(a)               R(b) W(a) R(c) W(b) C
T2:       B W(a) C
```

R(a) i T1 är före W(a) i T2 som i sin tur är före W(a) i T1.
R(a) in T1 is before W(a) in T2 which is before W(a) in T1.

b) Lägg till 2-faslåsning i transaktionerna T1 och T2. Lås så lite som möjligt.
Add 2-phase locking to the transactions T1 and T2. Lock as little as possible.

Svar:

Kravet på minimal låsning är för att få maximal parallellism. Det åstadkoms genom att använda exklusiva lås (X) endast vid skrivningar i övriga fall används delade lås (S). Det gäller även att låsa så sent som möjligt och låsa upp så tidigt som möjligt. Kravet på 2-faslåsning gör dock att vi inte kan börja låsa upp (U) förrän vi låst allt som behövs.
The requirement to use minimal locking is to maximize concurrency. Use of shared locks (S) increases the concurrency, hence we will only use exclusive locks (X) when they are needed (i.e., writes). We will also lock as late as possible, and unlock (U) as early as possible. However, the requirement to use 2-phase locking prevent us from doing any unlocks antil all locks are aquired.

```
T1: B S(a) R(a) S(b) R(b) X(a) W(a) S(c) R(c) X(b) U(a) U(c) W(b) U(b) C
T2: B X(a) W(a) U(a) C
```

c) Går det att få deadlock med den låsning som du införde i b) ifall flera instanser av T1 körs samtidigt?
Is deadlock possible with the locking used in b) if several instanse of T1 run at the same time?

Svar:

Ja det går att få deadlock om vi använt delade lås. Ifall alla lås är exklusiva kan inte T1 får deadlock med sig själv.
Yes it is possible if we use shared locks. If all locks are exclusive then T1 can not dead-lock with itself.

```
T1:  B S(a) R(a) S(b) R(b)                              X(a)....
T1':                        B S(a) R(a) S(b) R(b) X(a) ...
```

Både T1 och T1' väntar på att få låsa a exklusivt.
Both T1 and T1' waits for exclusive lock on a.

d) Går det garantera serialiserbarhet även utan lås? Svara antingen med varför det inte går eller hur det kan göras.
Is it possible to guarantee that all schedules are serializable without using any locks? Describe why not, or how to do it.

Svar:

Ja, t ex tidsstämpling (16.2) och validering (16.3).
Yes, e.g., time stamps (16.2) or validation (16.3).

Uppgift 5. Relationen Person innehåller ett unikt ID-nummer, ett namn och ett födelsedatum.
12p *The relation Person contains a unique id-number, a name and when the person was born.*

Relationen Född kopplar samman tre personer: ett barn, dess biologiska mor och far.
The relation Född connects three persons: a child, and its biologic mother and father.

(a) Skriv en SQL-fråga som ger namnen på Sven Svenssons syskon.
Write an SQL-query which returns the name of all Sven Svenssons' brothers and sisters.

Svar:

Jag döper attributen som följer:
I will use the following relations:

person(id,namn,född)
född(barn,mor,far)

I relationen person gäller att id är det unika ID-numret, namn är en sträng och född en datum. I relationen född är alla attribut ID-nummer och referenser till person.

Under antagandet att Sven Svensson är unik fungerar nedanstående fråga för alla helsyskon till Sven. Ifall Sven Svensson inte är unik går det inte svara på frågan för vi vet inte vilken Sven Svensson frågan gäller.

The relation person contains the unique id-number (id), the name as a string (namn), and the date the person was born (född). The relation född (eng born) contains one attribute each for the child (barn), mother (mor), and father(far). All of the attributes are id-numbers.
If we asssume that Sven Svensson is unique then the question below gives all of Sven Svenssons' brothers and sisters that have both the same father and mother. It is not possible to answer the question if Sven Svensson is not unique, since we do not know which Sven Svensson in that case.

```
select pp.namn
from person as ss, född as sb, född as pb, person as pp
where ss.namn = 'Sven Svensson'
  and ss.id = sb.barn
  and sb.mor = pb.mor
  and sb.far = pb.far
  and pb.barn = pp.id
  and pp.id <> ss.id
```

(b) Skapa en vy med SQL som innehåller alla mödrar och hur många barn de har.
Create a view, in SQL, whith all mothers and how many children they have.

Svar:

Det står inte om svaret skall innehålla mödrarnas namn, id eller båda. Här väljer jag endast id.
The question does not specify if the answer should contain the name and/or the id of the mothers. I have choosen to answer with id only.

```
create view mödrar as
  select född.mor,count(*) as barn
  from född
  group by född.mor
group by född.mor
```

Vill vi göra en lista där även namnen ingår blir det lite mer komplicerat. För att kunna skriva ut person.namn måste attributet ingå i group by.
If we want to create a list with both name and id then it gets slightly more complicated. Note that person.namn must be in group, otherwise it can not be used in the output.

```
create view mödrar as
  select person.id, person.namn, count(*) as barn
  from person, född
  where person.id = född.mor
  group by person.id, person.namn
group by född.mor
```

(c) Skapa en vy som innehåller hur många barn en person har i genomsnitt. Det är rekommenderat att skapa vyer med delresultat.
Create a view with the average number of children a Person has. It is recomended to create other views as step-stones.

Svar:

Det finns en trivial lösning där man räknar antalet barn och antalet personer. Svaret blir sedan *2*antal barn/antal personer*. (2 behövs då varje barn räknas en gång för sin mor och en gång för sin far.)
*There is a trivial solution, count the number of children and the number of persons. The answer is then 2*number of children/number of persons. (The 2 is needed since every child is counted twice.)*

Ifall man inte gör ovanstående, utan först räknar ut en tabell med hur många barn var och en har och sedan använder avg(), gäller det att inte glömma bort de barnlösa.
A more compicated method is to create a table with number of children for each person and then use avg(). It is easy to miss people without children.

```
create view harbarn as
  select person.id,count(*) as barn
  from person join född on (person.id = född.mor) or (person.id = född.far)
  group by person.id;

create view barnlösa as
  (select id from person)
  minus
  (select id from harbarn);
```

```
create view antalbarn as
  select id, barn
  from harbarn union (select id, 0 as barn from barnlösa);

create view avgbarn as
  select avg(barn) avgbarn
  from antalbarn;
```

Uppgift 6. Förklara, kortfattat, hur man översätter följande konstruktioner i ER-diagram till relationer.
8p *Explain, briefly, the rules for translating the following ER-diagram concepts into relational tables*:

Svenska	*Engelska*
stark entitet	*strong entity set*
svag entitet	*weak entity set*
M-N relation	*M-to-N relation*
flervärda attribut	*multi-valued attributes*

Svar:

stark entitet En stark entitet med attributen A översätts till en relation med en kolumn för varje attribut i A.
A strong entity is translated into a table with one column for each attribute.

svag entitet En svag entitet beror alltid på en annan entitet, föräldrarentiten. Anta att den svaga entiteten har attributen A och föräldrarentitetens nyckel består av attrinbuten B. Den svaga entiteten översätts då till en tabell med en kolumn för varje attribute i $A \cup B$.
A weak entity does always depend on another entity, the parent entity. Assume that the weak entity has the attributes A, and the parent entity has a key with the attributes B. The weak entity is then translated into a table with one column for each of the attributes in $A \cup B$.

Det finns en extra svårighet i att en svag entitet kan bero på en kedja av svaga entiteter. I slutet på en sådan kedja måste det finnas en stark entitet. Relationerna för sådana svaga entiteter kommer att få kolumner för alla attribut som hör till svaga nycklar "högre" upp i kedja plus attributen i starka entitetens nyckel.
There can be a chain of weak entities, but such a chain always ends in a strong entity.

M-N relation En M-N relationer med attributen A i ER-diagram översätts till en relationer som är unionen av nycklarna från de två, eller flera, entiteter som relationen binder samma plus attributen i A.
A M-N relation with attributs A in the ER-diagram is translated into a relation where the key is the concatenation of the keys from the two, or more, entities that the relation connects. The attributes in A also creates columns in the relation.

flervärda attribut För det flervärda attributet M skapas en tabell T med rader som består av en kolumn med värdet av M och en kolumn med entitetens nyckel.
A multi-valued attribute M is translated into a table T where each row contains a value of M and the key of the entity.

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*.
- `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.