Tekniska Högskolan i Linköping
Institutionen för datavetenskap
Jalal Maleki

T E N TA M E N
732G16 Databaser
Design och programmering

Datum: **20170608 kl 8-12**

Jourhavande lärare: **Eva Ragnemalm**
Tel: **070-1907391**
Besöker tentamenslokalen c:a kl 10

Hjälpmedel: **Inga**

Följ instruktionerna på tentaomslaget.

Poängfördelning

Uppgift	Poäng
1	10
2	6
3	3
4	5
5	5
6	5
7	5
8	10
9	6

Betyg: G: 30p VG: 43p Max: 55p

1. **Begrepp**: För varje nedanstående påstående, är det sant eller falskt? Rätt svar ger en poäng. Fel svar ger minus en poäng. Uppgiften kan dock ej ge negativt resultat. (10p)

   a) Alla data i en *databas* har något samband med varandra, annars är det inte en databas.

   b) En databas' *schema* beskriver precis när värdena i olika tabeller får ändras.

   c) Ett *relationsschema* är en *konceptuell datamodell*.

   d) Ett *referensattribut* är ett villkor som gäller för ett attribut, t.ex. att ett lagersaldo inte får ha ett negativt värde.

   e) Ett *fullt funktionellt beroende* är ett funktionellt beroende där man inte kan ta bort något attribut ur determinanten och fortfarande ha ett funktionellt beroende.

   f) I en Select-sats kan en *vy* (skapad med Create View...) användas på samma sätt som en tabell, men vyn tar ingen plats i databasen.

   g) Om man ska lagra en lång lista av personer i en *hashtabell* med 1 000 buckets är födelsedagen (mmdd) en bra *hashfunktion*.

   h) *Läsning av smutsiga data* (*dirty read*) innebär att transaktion 1 läser data som transaktion 2 ändrat innan transaktion 2 har gjort commit.

   i) Om transaktion 1 väntar på att få låsa databasobjekt A, som är låst av transaktion 2, som i sin tur inte kan köra klart (kan inte släppa A) innan den får tillgång till databasobjekt B, som i sin tur är låst av transaktion 1, kallas det *deadlock* (dödlig låsning).

   j) *Konservativ tvåfaslåsning* är ett låsningsprotokoll som kräver att dataobjekten låses i en viss förbestämd turordning.

Kombinationsuppgift, bakgrund för upg 2-6: En konsultfirma har ett antal konsulter som utför uppdrag åt olika kunder. I samband med uppdragen har konsulterna utgifter, som sedan hänförs till olika konton (logi, resor, representation, övrigt). Man skapar följande relationer för att hålla reda på detta:

Uppdrag

Uppdragsnr	Kund	Start	Slut	Konsult

Utgift

Uppdragsnr	Konto	Belopp

2. **Begrepp:** Antag att Uppdragsnr i Utgift är en främmande nyckel som refererar till Uppdrag enligt definitionen: 6p

```
alter table Utgift add constraint utgift_uppdrag foreign key
(Uppdragsnr) references Uppdrag (Uppdragsnr);
```

a) Ange ett innehåll i Uppdrag sådant att nedanstående kommando går att genomföra:

```
insert into Utgift values (324, Logi, 1234);
```

b) Antag att det finns en rad i Uppdrag med värdet 2 på Uppdragsnr och en rad i Utgift med värdet 2 på Uppdragsnr. Vad händer när man ger nedanstående kommando? Varför?

```
delete from Uppdrag where Uppdragsnr=2;
```

c) Antag samma värden som i b. Vad händer när man ger nedanstående kommando? Varför?

```
delete from Utgift where Uppdragsnr=2;
```

3. **Relationsalgebra:** Utifrån ovanstående relationer (Uppdrag och Utgift), skriv följande frågor i relationsalgebra: 3p
   a) Vilka olika kunder har man utfört uppdrag åt? Om samma kund förekommer flera gånger ska namnet bara listas en gång ändå.
   b) Vilka olika kunder har konsult 10 arbetat åt?
   c) Lista alla utgiftsposter (uppdragsnr, konto och belopp) som konsult 10 genererat under sina uppdrag.

4. **SQL:** Utifrån ovanstående tabeller (uppdrag och utgift), skriv följande frågor i SQL: 5p
   a) Vilka olika kunder har konsult 10 arbetat åt? Om samma kund förekommer flera gånger ska namnet bara listas en gång ändå.
   b) Summera alla utgifter som konsult 10 genererat under sina uppdrag.
   c) Summera alla utgifter, uppdelat per uppdrag, som konsult 10 genererat. Lista alltså uppdragsnummer och summa utgifter för det uppdraget.

5. **Normalisering:** I tabellen Uppdrag finns ett funktionellt beroende från Uppdragsnr till alla andra attribut. Antag att konsultfirman har den situationen att alla kunder alltid vill ha tillbaks samma konsult när de väl haft den en gång. Det medför att det finns ett fullt funktionellt beroende från kund till konsult i tabellen. Antag att inga andra fulla funktionella beroenden finns. Vilken normalform uppfyller tabellen Uppdrag, och varför (dvs motivera)? Du kan anta att attributen är atomära. 5p

6. **Fysiska databasen:** Med tiden blir tabellerna ganska stora och man vill skapa index för att snabba upp sökningarna. Idag är Uppdrag sorterad på Uppdragsnr, men det finns inget index på Uppdragsnr. Man söker ofta efter en viss konsults uppdrag och tycker det tar tid. Men om man vill skapa ett klusterindex till konsultfältet måste filen sorteras på det fältet. Filen Uppdrag har 4800 poster, varje post tar 200byte, konsult representeras med heltal som tar 20

byte. Adressen till ett block tar 20 byte. Hårddiskens blockstorlek är 4096 byte och accesstiden 10ms. 5p

a) Hur lång tid tar det att söka ut alla en konsults uppdrag i dagsläget? Redovisa dina beräkningar och ev. antaganden tydligt så kan du få poäng för delar även om det inte blir helt rätt.

b) Vad behöver du veta, mer än det som anges ovan, för att kunna beräkna hur stort klusterindexet på konsult blir?

7. **Transaktioner**: Din databas-server hängde sig och du blev tvungen att starta om den. Lyckligtvis behövde du inte läsa tillbaka backup-en, för inget blev skadat. Loggfilen, som byggts upp sedan senaste backup-en, ser ut som nedan. För respektive markerat kommando (a-d) i loggfilen, tala om ifall det ska **repeteras** (dvs nya värdet skrivs i databasen) eller **rivas upp** (dvs gamla värdet skrivs i databasen), eller kan **ignoreras**. All information du behöver finns i loggfilen. Inom parenteserna har värdena följande betydelse: ”tnnn” är ett transaktions-id, ”konto nnn” ett databasfält, det första numeriska värdet är det gamla värdet på databasfältet, det andra är det nya värdet. 5p

```
--filen börjar här--
   start (t123)
   läs (t123, konto 567)
(a) skriv (t123, konto 567, 150, 350)
   läs (t123, konto 568)
   skriv (t123, konto 568, 12 300, 12 100)
   commit (t123)
   checkpoint
   start (t124)
   läs (t124, konto 345)
   start (t125)
(b) läs (t125, konto 789)
(c) skriv (t124, konto 345, 3450, 2450)
   läs (t124, konto 346)
   skriv (t124, konto 346, 8 330, 9 330)
   commit (t124)
(d) skriv (t125, konto 789, 2000, 1900)
--filen slutar här--
```

Design: bakgrundsinformation för upg 8-9: Det stora spelkonventet HinCon behöver hjälp med att utveckla ett nytt registreringssystem för sina deltagare. Det är viktigt att hålla rätt på alla olika event som arrangeras och som folk ska anmäla sig till. Det ska samtidigt innehålla alla data man behöver för att planera aktiviteter och lokaler av lagom storlek.

Konventet har ett antal event, aktiviteter som arrangeras av en eller flera deltagare och deltas i av andra deltagare. Varje event har en titel (som är unik och max får vara 30 tecken), en textbeskrivning, en eller flera arrangörer, en lokal och ett eller flera pass. För att förenkla

schemaläggningen delas konventet upp i tre pass per dag under konventshelgen, där ett pass representeras av nummer 1-11 där pass nr 1 motsvarar onsdag kväll, pass två torsdag förmiddag, pass tre torsdag eftermiddag osv till pass 11 söndag förmiddag. Alla pass för ett visst event är i samma lokal så att arrangörerna kan ha visst material på plats. Flera event kan dock dela lokal, fast inte samtidigt. Deltagare kan anmäla sig till event (då till något specifikt pass) men alla event kräver inte föranmälan. Man behöver kunna kolla vilka event en deltagare är anmäld till och vilka som är anmälda till ett visst event. Event behöver inte ha anmälda deltagare men det måste finnas minst en arrangör när eventet läggs in i databasen.

För varje deltagare behöver man veta personnummer (man vill använda det som nyckel), namn, adress, epostadress samt telefon. Man vill också hålla rätt på ifall de betalat inträde. Om man är arrangör för något event behöver man inte betala inträde.

Man disponerar ett 60-tal lokaler som var och en har ett namn (unikt, inget mer än 15 tecken), ett maxantal platser och en städansvarig deltagare. Lokalerna fördelas på eventen då anmälningstiden gått ut för alla event med föranmälan. Event som inte har föranmälan antas behöva 30 platser.

Följande frågor behöver man kunna ställa till databasen.

a) Dags att boka lokaler. Lista alla event (namn, pass nr samt antal deltagare det passet) som har pass med mer än 50 anmälda deltagare, för att fördela de största salarna.
b) Vilken lokal är ledig så att vi kan lägga in ett event till, som har 12 anmälda och ligger torsdag em (dvs lista lokaler med kapacitet >= 12 som är lediga då).
c) Skapa en lista över de event en viss deltagare är anmäld till (en lista innehållande dag, tid, event-namn och lokal, för deltagare 890608-0123).

8. Rita ett ER-diagram för den tänkta databasen. Glöm inte markera nyckelattribut för entitetstyper, kardinalitet och deltagande för sambandstyper. Om du tycker att beskrivningen ovan underspecificerar databasens innehåll, kan du göra egna antaganden för att komplettera beskrivningen så länge de inte motsäger något givet. Sådana antaganden måste skrivas ner. För varje entitetstyp, ge ett exempel på attributvärde för alla attribut. Kom ihåg att skriva ut eventuella semantiska integritetsvillkor som inte representeras på annat sätt i text. 10p

9. Konvertera ER-diagrammet till relationsschema. Markera primärnycklar och främmande nycklar som vi gjort i kursen. 6p

Lycka till!