Örebro universitet
Institutionen för teknik
Thomas Padron-McCarthy (Thomas.Padron-McCarthy@oru.se)

Tentamen i

Databasteknik

för D1, SDU1 m fl

tisdag 13 januari 2009 kl 08:00 - 12:00

Gäller som tentamen för:
DT1012 Datateknik A, Databasteknik, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0310

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 40. För betyget 3 respektive G krävs 25 poäng. För den som följt kursen hösten 2008 ger varje i tid inlämnad inlämningsuppgift en extra poäng. Den som *inte* gått kursen hösten 2008 får dessa (fem) extrapoäng ändå.
Resultat och lösningar:	Meddelas på kursens hemsida eller via e-post senast tisdag 3 februari 2009.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på institutionen. Man kan också få sin rättade tenta hemskickad.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070 - 73 47 013.

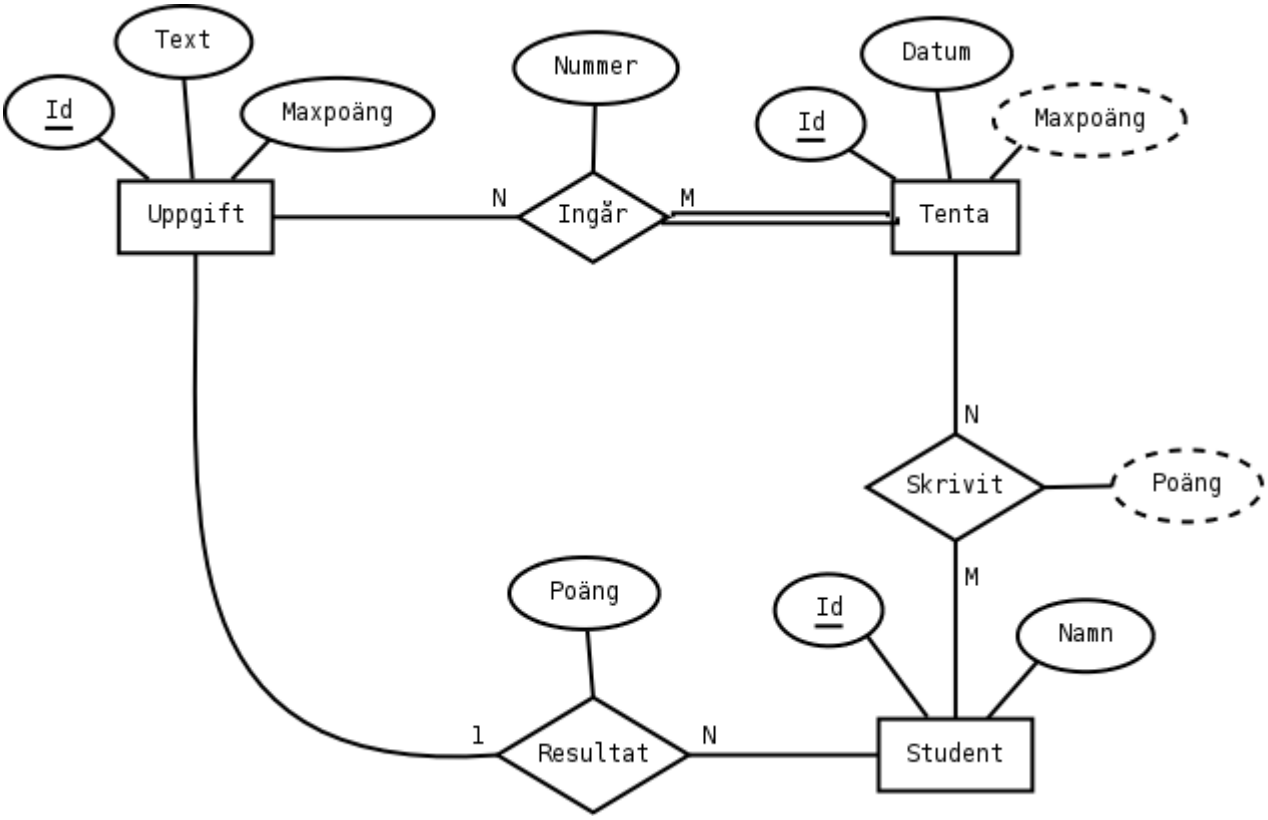
- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Lös bara en uppgift per blad. Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges. Gjorda antaganden får inte förändra den givna uppgiften.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

Scenario till uppgifterna

Läraren är trött på att hitta på uppgifter till sina databastentor, så i fortsättningen kommer han bara att återanvända uppgifter från gamla tentor. Han samlar förstås alla uppgifterna i en databas, tillsammans med studenternas resultat.

Här är ett ER-diagram för databasen:



Här är några kommentarer om ER-diagrammet:

- Som vi kan se finns det ett antal **uppgifter**, och varje uppgift har ett unikt id-nummer, en text (som är själva frågan), och en maxpoäng.
- Man sätter ihop uppgifter till ett antal **tentor**. Varje tenta har ett unikt id-nummer, ett datum (när tentan gavs), och utifrån vilka uppgifter som ingick kan man räkna ut en maxpoäng för den tentan.
- Varje tenta innehåller en eller flera uppgifter, och varje uppgift kan ingå i flera olika tentor. Dessutom har uppgifterna ett ordningsnummer i just den tentan. (Exempelvis kan uppgiften med id-nummer 17 vara uppgift 1 på en tenta, och uppgift 3 på en annan.)
- Studenterna skriver tentorna, och löser uppgifterna. Varje student får en viss poäng i resultat på varje uppgift som han eller hon löser, och utifrån den poängen kan man räkna ut en poäng för studenten på hela den tentan.

Uppgift 1 (3 p)

Det finns en eller flera saker som verkar konstiga med ER-diagrammet. (Jämför med din egen erfarenhet av hur tentor fungerar.) Förklara vad, och hur det nog borde vara i stället!

Uppgift 2 (7 p)

Implementera den beskrivna databasen i relationsmodellen, dvs översätt diagrammet till tabeller.

Utgå från det givna ER-diagrammet, och inte eventuella förändringar som du föreslagit i uppgift 1.

Du behöver inte skriva **create table**-kommandon i SQL, men du ska ange vilka relationer som finns och vilka attribut varje relation innehåller. Ange också vad som är primärnyckel, och vilka referensattribut som finns och vad de refererar till.

Implementationen ska vara bra.

Observera att du alltså ska utgå från det givna ER-diagrammet, och inte ha med eventuella förändringar som du föreslagit i uppgift 1.

Uppgift 3 (13 p)

Formulera följande frågor i SQL.

a) (1p) Hur lyder uppgiften med id-nummer 199?

b) (2p) Det är bara en tenta som ges 2009-01-13. Hur lyder uppgift nummer 3 på den tentan?

c) (2p) Vad är maxpoängen på den tentan?

d) (2p) Vilka uppgifter har aldrig varit med på någon tenta? (Vi vill veta de uppgifternas id-nummer.)

e) (3p) Vilken uppgift har varit med på flest tentor? (Vi vill veta den uppgiftens id-nummer.)

f) (3p) Ordna studenterna i ordning efter hur många tentor de skrivit. Studenter med flest tentor ska komma först, och studenter med minst tentor ska komma sist. Även studenter som inte skrivit några tentor alls ska vara med i resultatet, och förstås komma sist. Vi vill ha med studenternas id-nummer och namn.

Definiera gärna vyer om det underlättar, men skapa inte nya tabeller.

Uppgift 4 (4 p)

Databasen är stor, med många rader i varje tabell. De fyra SQL-frågorna a, b, c och d i uppgiften ovan körs väldigt ofta (men med andra konstanter än **199** och **2009-01-13**), och behöver snabbas upp. Vi märker att det inte finns några index alls i databasen, inte ens på nycklar. Vilka index bör man skapa för att just de frågorna ska gå snabbt att köra? Förklara varför!

Uppgift 5 (4 p)

I kursen har vi tagit upp tre databashanterare: Mimer, MySQL och Microsoft Access. Diskutera för var och en av dessa om det hade varit lämpligt att använda just den databashanteraren till den här databasen.

Uppgift 6 (9 p)

För att lägga in en ny student i databasen behövs två SQL-kommandon: först ett kommando för att ta reda på vilket studentnummer (kolumnen **Id** i tabellen **Student**) som är det högsta hittills, och sen ett kommando för att lägga in den nya studenten i tabellen.

a) (2p)

Skriv de två SQL-satserna. Vi kan kalla den nya studenten för **Olle**, och vi kan anta att det högsta studentnumret var **4711**.

b) (1p)

Hur gör man för att samla dessa satser i en transaktion?

c) (1p)

Vad är en transaktion?

d) (4p)

Transaktioner ska ha fyra egenskaper, som man brukar ange med förkortningen **ACID**. Förklara kort vad var och en av dessa egenskaper innebär.

e) (1p)

Visa med ett exempel något fel som skulle kunna uppstå om man inte samlade SQL-satserna från delfråga a i en transaktion.