Örebro universitet
Institutionen för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i

Databasteknik

för D1 m fl

tisdag 10 januari 2017

Gäller som tentamen för:
DT105G Databasteknik, provkod 0100
DT1026 Datateknik A, Databasteknik, provkod 0100
DT1030 Datateknik A, Tillämpad datavetenskap, provkod 0310

Hjälpmedel:	Ordbok för översättning.
Poängkrav:	Maximal poäng är 47. För godkänt betyg krävs 31 poäng, varav minst fyra poäng på uppgift 1 och minst fem poäng på uppgift 2. För den som följt kursen hösten 2016 ger varje i tid inlämnad inlämningsuppgift en extra poäng. Den som *inte* gått kursen hösten 2016 får dessa (fem) extrapoäng ändå.
Resultat:	Meddelas på kursens hemsida eller via e-post senast tisdag 31 januari 2017.
Återlämning av tentor:	Elektroniskt via Studentforum.
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.
- 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

Örebro universitetssjukhus, USÖ, har blivit för litet. Därför bygger man ett nytt sjukhus, Örebros Stora Översjukhus, förkortat ÖSÖ. ÖSÖ består av flera avdelningar, med patienter som vårdas på avdelningarna. Patienterna har diagnosticerats med olika sjukdomar. Nu behöver ÖSÖ förstås en databas för att lagra data om verksamheten.

Det vi ska lagra i databasen är följande:

- **Avdelningar.** Varje avdelning har både ett unikt namn, till exempel **Kirurgi**, och ett unikt nummer, till exempel **17**.
- **Patienter.** Varje patient har ett unikt personnummer, till exempel **19631211-1658**, ett (inte nödvändigtvis unikt) namn, till exempel **Anna A. Andersson**, och kan också (men måste inte) vårdas på en av sjukhusets avdelningar.
- **Sjukdomar.** Varje sjukdom har ett unikt namn, till exempel **pest**.
- **Symtom.** Varje sjukdom har noll eller flera symtom, till exempel **bölder**, **hosta** och **feber**.
- **Diagnoser.** Patienterna har diagnosticerats med olika sjukdomar. Varje patient kan ha flera olika sjukdomar, och flera olika patienter kan förstås ha samma sjukdom.

Uppgift 1 (5 p)

Rita ett ER- eller EER-diagram för den beskrivna databasen. Använd informationen i scenariot ovan, men tänk också på att det ska gå att svara på frågorna i uppgift 4 nedan.

ER- och EER-diagram kan ritas på flera olika sätt. Om du använder en annan notation än kursboken, måste du förklara den notation som du använder.

Uppgift 2 (6 p)

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

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 alla kandidatnycklar, vilken av dessa som är primärnyckel, samt vilka referensattribut som finns och vad de refererar till.

Implementationen ska vara bra. (Tips: Se även uppgift 3 nedan.)

Uppgift 3 (3 p)

I uppgiften ovan står det att implementationen ska vara bra. Ett sätt att göra en dålig implementation är att ha en tabell som heter **Sjukdomar**, där det finns en kolumn som heter **Symtom**, och där man skriver in sjukdomens alla symtom separerade med kommatecken. Pest skulle till exempel kunna ha texten "**bölder, huvudvärk, feber**" i symtom-rutan.

a) (1p) Vilken normalform bryter detta mot?

b) (2p) Vilka praktiska problem innebär det?

Uppgift 4 (10 p)

Formulera följande frågor i SQL. Definiera gärna vyer om det underlättar, men skapa inte nya tabeller.

a) (1p) Vad heter de avdelningar som har namn som börjar på bokstaven **K**?

b) (2p) Vilka symtom drabbas man av om man har pest?

c) (2p) Det kommer in en patient som har feber. Vilka sjukdomar kan hon ha? Vi vill alltså veta namnen på de sjukdomar som har **feber** bland sina symtom.

d) (2p) Vad heter den avdelning där det just nu vårdas flest patienter?

e) (3p) Vad heter de stackars patienter som samtidigt har både pest och kolera?

Uppgift 5 (3 p)

I frågan ovan står det att man gärna får definiera vyer, men inte skapa nya tabeller. Vilka är de viktigaste likheterna och skillnaderna mellan en vy och en tabell?

Uppgift 6 (3 p)

Databasen innehåller realistiska mängder data, vilket innebär 30-40 avdelningar, många tusen sjukdomar och bortåt en miljon patienter och diagnoser. SQL-frågorna a, b och c i uppgift 4 körs väldigt ofta (men kanske med andra konstanter, till exempel att man söker efter avdelningar som har namn som börjar på bokstaven **A**). De tar för lång tid att köra, och behöver snabbas upp. Vi märker att det inte finns några index alls i databasen, inte ens på nycklar.

a) Vilka index bör man skapa för att just SQL-frågorna a-c i uppgift 4 ska gå snabbt att köra?

b) Ge ett exempel på ett index som *inte* skulle snabba upp någon av frågorna a-c i uppgift 4, och förklara varför det inte förbättrar tiderna!

Uppgift 7 (3 p)

ÖSÖ har tusentals anställda, som kommer att behöva jobba med databasen i scenariot. Nu behöver vi välja vilken databashanterare som ska användas.

a) Föreslå en databashanterare som kan vara lämplig att använda. Motivera varför den är lämplig!

b) Föreslå en databashanterare som *inte* är lämplig. Motivera varför den inte är lämplig!

Uppgift 8 (14 p)

Ange för varje påstående om det är sant eller falskt! Lämna in denna sida tillsammans med resten av svaren. Fel svar ger inte minuspoäng.

Påstående	Sant	Falskt
Uttrycket null = NULL i ett WHERE-villkor blir falskt.		
Många databashanterare skapar automatiskt ett index på primärnyckeln i varje tabell.		
Vi söker ofta efter patienter med ett visst namn. Då bör vi skapa ett sammansatt index på (ID, Namn), eftersom ID är primärnyckel i tabellen.		
En vy kan användas för att dela upp en komplicerad fråga i flera steg.		
Flera SQL-satser som hör ihop, så att antingen ska alla utföras eller också ingen, kan grupperas till en enhet i form av en transaktion.		
Isolering mellan transaktioner betyder att varje transaktion har sina egna tabeller, som andra transaktioner inte kan se.		
I en trigger kan vi göra mer komplexa kontroller än vad vi kan göra med integritetsvillkor i SQL.		
SQL injection innebär att en hacker kan komma åt och förändra en databas genom att skriva in särskilt utformade texter till exempel i ett textfält på en webbsida.		
Man kan använda kommandona LOCK och UNLOCK för att gruppera flera SQL-kommandon till en transaktion.		
INSERT-kommandot kontrollerar nyckelvillkor, så man inte får dubbletter i primärnyckeln, men det gör inte UPDATE-kommandot, så med UPDATE kan man skapa dubbletter i primärnyckeln.		
För att ta bort rättigheter till ett objekt i databasen från en användare använder man kommandot REVOKE.		
"Relationerna" i en relationsdatabas är kopplingarna mellan tabeller med främmande nycklar ("foreign keys").		
Ett funktionellt beroende är samma sak som en främmande nyckel.		
Boyce-Codds normalform (BCNF) ställer fler krav på hur en tabell får se ut än andra normalformen, så det finns tabeller som uppfyller andra normalformen men inte BCNF.		