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

Tentamen i
Databaskonstruktion
för D1, TDVA, SDU1 m fl

tisdag 21 augusti 2007 kl xx:00 - xx:00 i xxxx

Gäller som tentamen för:
DAT024 Databaskonstruktion, provkod 0300 (och 0100)
TDD121 Tillämpad datavetenskap A, provkod 0500 (och 0300)

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 48. För betyget 3 respektive G krävs 24 poäng.
Resultat och lösningar:	Meddelas på kursens hemsida senast måndag 10 september 2007.
Visning:	Tisdag 11 september 2007 kl 12:00-12:30 i mitt rum (T2220). Efter visningen kan tentorna hämtas på expeditionen.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070-7347013.

- 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

Sveriges främste samlare av jordglobar vill lagra data om sina jordglobar i en databas. Det man behöver lagra i databasen är, mer detaljerat, följande:

1. **Jordglobar**. En jordglob har de här egenskaperna:
   - Jordglobens inköpsdatum, som lagras som en textsträng, exempelvis **2007-08-21**.
   - Staden där jordgloben köptes. Namn på städer kan vara trettio bokstäver långa.
   - Jordglobens diameter i centimeter. Den anges med ett flyttal.
   - En bedömning av hur snygg jordgloben är, på en skala från 1 till 5. Den lagras som ett heltal.

   Vi räknar med att samlaren bara köper en jordglob per dag i varje stad.
2. **Säljare** som han köpt jordglobarna från. Varje säljare har ett namn, en adress och ett telefonnummer. I databasen vill vi lagra från vilken säljare som varje (köpt) jordglob är köpt från.
3. **Jordglobsutställningar** som han deltagit i och visat upp jordglobarna. Varje jordglobsutställning har ett namn, den skedde ett visst datum i en viss stad och ett visst land. Det ska också gå att lagra vilka jordglobar som vår samlare tog med till den utställningen.
4. De **priser** som jordglobarna fått på utställningarna. Priset har ett kort namn, till exempel **bäst polerade träfot**, och varje pris gavs till en viss jordglob vid en viss utställning.

Uppgift 1 (7 p)

Rita ett ER-diagram för den beskrivna databasen. Använd informationen i scenariot ovan, men tänk också på att det ska gå att svara på frågorna i uppgift 3 nedan.

ER-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 vad som är primärnyckel, och vilka referensattribut som finns och vad de refererar till.

Implementationen ska vara bra.

Uppgift 3 (10 p)

Formulera följande frågor i SQL.

a) (1p) Vilka jordglobsutställningar har det varit i **Kumla**?

b) (2p) Vilka jordglobar har samlaren haft med sig till **Gnesta International Earthglobe Fair**? Vi vill veta dessa jordglobars inköpsdatum och inköpsstad.

c) (2p) Vilka jordglobar har fått pris i samma stad som de är inköpta? Vi vill veta dessa jordglobars inköpsdatum och inköpsstad.

e) (2p) Har någon jordglob som inte fick femma i snygghetsbetyg fått pris? Vi vill veta vilka priser det i så fall var, dvs prisernas namn.

e) (3p) I vilken stad har samlaren fått flest priser?

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

Uppgift 4 (4 p)

SQL-frågorna a, b och c i uppgiften ovan körs väldigt ofta, och behöver snabbas upp. Vi märker att det inte finns några index alls i databasen, inte ens på primärnycklar. Vilka index bör man alltså skapa för att frågorna a-c ska gå snabbt att köra? Motivera valet!

(Jordglobssamlingen är stor, med många tusen jordglobar, och samlaren har genom årens lopp besökt tusentals internationella jordglobsutställningar.)

Uppgift 5 (6 p)

Om man frågar en sotare, så består världen av **hus** och **skorstenar**. Varje hus har ett nummer och en adress. Varje skorsten har ett nummer, en höjd och ett antal skorstenspipor, och den sitter på ett hus.

Ett hus är antingen ett **bostadshus** eller något som kallas **verksamhetshus**. Bostadshus delas i sin tur in i **enfamiljshus** och **flerfamiljshus**. Flerfamiljshus har ett **antal lägenheter**. Ett verksamhetshus kan vara antingen en **kontorsbyggnad**, ett **varuhus** eller en **industrifastighet**. Ett verksamhetshus har, till skillnad från bostadshus, en **brandfarlighetsklassning**.

a) (3p) Rita upp ett EER-diagram över husen och skorstenarna.

b) (3p) Översätt EER-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 vad som är primärnyckel, och vilka referensattribut som finns och vad de refererar till.

Uppgift 6 (6 p)

Vad är det för skillnad på:

a) SQL-kommandona **delete** och **drop**
b) en entitetstyp och en tabell
c) ett index och en främmande nyckel

Uppgift 7 (5 p)

I ett operativsystem (som Windows eller Unix) kan man för det mesta ange rättigheter på filer, för att styra vilka användare som får läsa, skriva och exekvera filerna.

Det fungerar på liknande sätt i Windows, men om vi tittar mer i detalj på Unix så finns det **användare** och **grupper**, där varje användare kan tillhöra en eller flera grupper. Varje fil ägs av en viss användare och en viss grupp. Sen kan man ange om filen får läsas, skrivas och/eller exekveras av användaren som äger den, gruppen som äger den, och alla andra. Totalt är det alltså nio rättigheter som kan sättas:

1. Den användare som äger filen kan ha rätt att läsa filen.
2. Den användare som äger filen kan ha rätt att skriva (dvs ändra i) filen.
3. Den användare som äger filen kan ha rätt att exekvera (dvs köra) filen.
4. De användare som ingår i gruppen som äger filen kan ha rätt att läsa filen.
5. De användare som ingår i gruppen som äger filen kan ha rätt att skriva filen.
6. De användare som ingår i gruppen som äger filen kan ha rätt att exekvera filen.
7. Alla andra användare kan ha rätt att läsa filen.
8. Alla andra användare kan ha rätt att skriva filen.
9. Alla andra användare kan ha rätt att exekvera filen.

I en databashanterare kan man för det mesta också ange vad de olika användarna ska få göra med innehållet i databasen. SQL använder kommandona **grant** och **revoke** för detta.

a) (2p) Visa med exempel hur **grant** och **revoke** fungerar.

b) (3p) Med **grant** och **revoke** kan man styra vad användarna får göra på ett betydligt mer avancerat sätt än i ett operativsystem som Unix. Förklara vad man kan göra, och visa med exempel.

Uppgift 8 (4 p)

Vad är CGI, och hur fungerar det? Vad har det med databaser att göra?
