Tentamen för DD1370
Databasteknik och informationssystem

Exempeltenta för kursen ht2013

Hjälpmedel:
Inga hjälpmedel utom papper och penna

Tänk på:
Skriv *högst* en uppgift på varje blad.
Använd *endast framsidan* på varje blad.
Uppgifterna kommer inte i svårighetsordning.
Skriv tydligt, motivera svaren – endast begriplig och läsbar
lösning ger poäng.
Maximal poäng finns angiven inom parentes vid varje uppgift.
Totalt ger tentamen en poäng (max 65), som sedan läggs ihop med era bonuspoäng.
En summa på 40 ger säkert godkänt.

Lycka till,

Petter

1. (*Totalt: 24p*) Ett företag använder en databas för att administrera uthyrning av campingstugor till privatpersoner. Databasen har följande struktur:

   Stuga(StugId, StugAdress, PostNr)

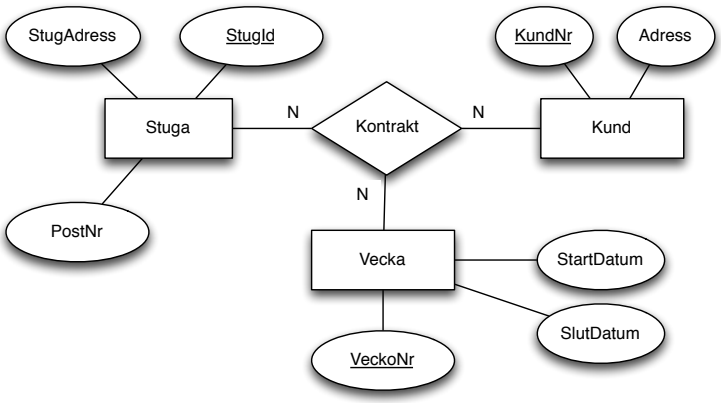
   Kontrakt(StugId,VeckoNr,KundNr)

   Kund(KundNr, Adress)

   Vecka(VeckoNr, StartDatum, SlutDatum)

   a) (*6p*) Rita upp en ER-modell som skulle resultera i ovanstående Databasstruktur. Motivera varje steg genom angivande av vilka regler i ”kokboken” som använts.

   Lösning: Se figur 1. Steg 1 i kokboken (Varje vanlig entitetstyp blir en tabell, attribut blir kolumner) verkar ha tillämpats på *Stuga*, *Kund* och *Vecka*. Gör vi dem till egna entiteter med attribut så får vi rätt databasstruktur för dessa tre. *Kontrakt* ser ut som resultatet av steg 5 (Varje flervägssamband blir en egen tabell, med sammansatt primärnyckel bestående av primärnycklarna från de ingående tabellerna). Tillämpar vi reglerna i kokboken på ER-modellen i figur 1 kommer nr 1 och 5 ge den givna databasstrukturen. De övriga reglerna resulterar inte i några förändringar av databasstrukturen. (Detta är en minimal förklaring. Det viktiga är att ni anger vilka regler som tillämpas och vad de leder till, samt vilka regler som inte leder till någon åtgärd. Det går även utmärkt att gå igenom reglerna lite mer strukturerat, som i lösningarna nedan).



Figur 1: En ER-modell som skulle ge upphov till den givna databasstrukturen.

   b) (*4p*) Ange för varje attribut i Databasstrukturen vilken datatyp som passar. Motivera dina svar.

   Lösning: StugId, KundNr, VeckoNr, PostNr skulle passa som Integer (heltal). StugAdress och Adress skall innehålla text, så de blir Varchar. StartDatum och SlutDatum är datum, och passar som Date.

   c) Skriv SQL-frågor som löser följande uppgifter.

i) (*1p*) Lista alla upptagna stugor (StugId) i Vecka 22.
Lösning:

```
SELECT "StugId" FROM "Kontrakt"
WHERE "VeckoNr" = 22
```

ii) (*2p*) Lista alla lediga stugor i vecka 22.
Lösning:

```
SELECT "StugId" FROM "Stuga" WHERE
"StugId" NOT IN
( SELECT "StugId" FROM "Kontrakt"
WHERE "VeckoNr" = 22 )
```

iii) (*1p*) Lista alla lediga stugor i vecka 22, som ligger i Postnummer 12345.
Lösning:

```
SELECT "StugId" FROM "Stuga" WHERE
"StugId" NOT IN
( SELECT "StugId" FROM "Kontrakt"
WHERE "VeckoNr" = 22 )
AND "PostNr" = '12345'
```

iv) (*1p*) Lista alla kontrakt för vecka 20-30.
Lösning:

```
SELECT * FROM "Kontrakt" WHERE
"VeckoNr" < 31 AND "VeckoNr" > 19
```

v) (*1p*) Lista alla kunder som har en tom adressrad.
Lösning:

```
SELECT * FROM "Kund"
WHERE "Adress" IS NULL
```

vi) (*1p*) Lista kundnummer på alla kunder som har ett Z i adressen, byt namn på kundnummerkolumnen till *Znubbar*.
Lösning:

```
SELECT "KundNr" AS "Znubbar" FROM "Kund"
WHERE "Adress" LIKE '%Z%'
```

vii) (*2p*) Skapa en lista på alla kunder (KundNr), som är sorterad efter antalet kontrakt.
Lösning:

```
SELECT "KundNr", COUNT( * ) AS "Antal Kontrakt"
FROM "Kontrakt"
GROUP BY "KundNr"
ORDER BY "Antal Kontrakt"
```

viii) (*2p*) Antag att man sparat resultatet av ovanstående fråga som en vy med namnet “KunderOchAntKontrakt”, med kolumnerna “KundNr” och “Antal Kontrakt”. Skapa med hjälp av denna vy en lista på alla kunder (KundNr och Adress), som är sorterad efter antalet kontrakt.
Lösning:

```
SELECT "Kund"."KundNr", "Antal Kontrakt", "Adress"
```

```
FROM "Kund" JOIN "KunderOchAntKontrakt"
ON "Kund"."KundNr" = "KunderOchAntKontrakt"."KundNr"
ORDER BY "Antal Kontrakt"
```

ix) (*3p*) Lista alla dubbelbokningar i systemet? (bokningar där samma stuga är bokad (genom ett kontrakt) av olika kunder samma vecka).

Lösning:

```
SELECT * FROM
"Kontrakt" JOIN "Kontrakt" AS "K2"
ON "Kontrakt"."VeckoNr" = "K2"."VeckoNr"
AND "Kontrakt"."StugId" = "K2"."StugId"
AND "Kontrakt"."KundNr" <> "K2"."KundNr"
```

2. (*Totalt: 16p*)

a) (*3p*) Beskriv skillnaden mellan en Driftsdatabas och ett Data Warehouse

Lösning: (se bok och föreläsningsanteckningar)

b) (*5p*) Beskriv för och nackdelar med Databaser respektive Kalkylbladsprogram (t.ex. Microsoft Excel, Openoffice Calc, Apple Numbers)

Lösning: (se bok och föreläsningsanteckningar)

c) (*2p*) Beskriv skillnaden mellan en Databas och ett DBMS

Lösning: (se bok och föreläsningsanteckningar)

d) (*2p*) Beskriv skillnaden mellan SQL-kommandona *Select* och *Create view*

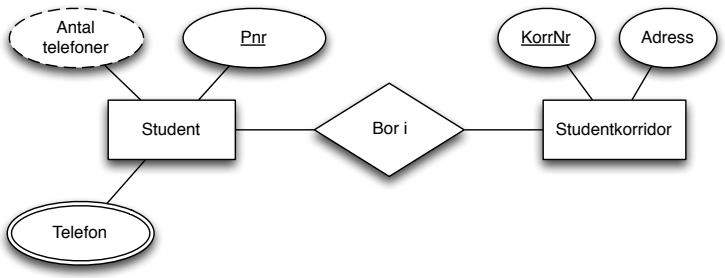
Lösning: (se bok och föreläsningsanteckningar)

e) (*2p*) Beskriv skillnaden mellan en *Svag* entitetstyp och en vanlig.

Lösning: (se bok och föreläsningsanteckningar)

f) (*2p*) Beskriv skillnaden mellan ett N:M-samband och ett flervägssamband.

Lösning: (se bok och föreläsningsanteckningar)



Figur 2: En ER-modell.

3. (*Totalt: 6p*) Betrakta ER-modellen i Figur 2.

   a) (*1p*) Avgör vilken typ sambandet är (1:1, 1:N, N:M) och motivera ditt svar.

   Lösning: Det är ett N:1 samband, Många studenter kan bo i samma studentkorridor, men en student kan inte bo i flera studentkorridorer (svaret kan förstås bero på hur man definierar begreppet "bo i").

   b) (*5p*) Överför ER-modellen till en Databasstruktur. Motivera varje steg genom angivande av vilka regler i "kokboken" som använts.

   Lösning: (se även boken sidan 85-86)

   Steg 1: Varje vanlig entitetstyp blir en tabell. Vanliga attribut blir kolumner i tabellerna.

   Vi ser att Student och Studentkorridor är vanliga entitetstyper, och Pnr, KorrNr, Adress är vanliga attribut. Antal telefoner och Telefon är däremot inte vanliga attribut (se nedan). Vi får alltså till en början följande:

   Student(Pnr)

   Studentkorridor(KorrNr,Adress)

   Steg 2: Varje 1:N-samband blir ett referensattibut i "många"-entitetstypens tabell.

   Vi har ett sådant samband (både 1:N och N:1 faller förstås inom denna regel). Många-sidan är Student, alltså får vi en ny kolumn i Student-tabellen.

   Student(Pnr, KorrNr)

   Studentkorridor(KorrNr,Adress)

   Steg 3: (Vi har inga 1:1-samband, alltså ingen åtgärd)

   Steg 4: (Vi har inga N:M-samband, alltså ingen åtgärd)

   Steg 5: (Vi har inga flervägssamband, alltså ingen åtgärd)

   Steg 6: (Vi har inga attribut på samband, alltså ingen åtgärd)

   Steg 7: (Vi har inga svaga etitetstyper, alltså ingen åtgärd)

   Steg 8: (Vi har inga sammansatta attribut, alltså ingen åtgärd)

   Steg 9: Varje flervärt attribut blir en egen tabell, primärnyckeln består av entitetstypens primärnyckel, kombinerad med det flervärda attributet.

   Telefon är ett flervärt attribut (se dubbelovalen). Därför får vi.

   Student(Pnr, KorrNr)

   Studentkorridor(KorrNr,Adress)

   Telefon(Pnr,TelefonNr)

   (här kallade vi attributet "TelefonNr" men det kunde även ha kallats "Telefon", attribut kan ha samma namn som tabeller)

   Steg 10: Härledda attribut tas inte med i databasen.

   Antal telefoner är ett härlett attribut (streckad oval). Detta hanteras med hjälp av en vy, men är inte en del av databasstrukturen.

   Slutresultatet är alltså:

Student(Pnr, KorrNr)
Studentkorridor(KorrNr,Adress)
Telefon(Pnr,TelefonNr)

4. (*Totalt: 6p*) Funktionaliteten i en databas kompletterar på många sätt funktionaliteten i ett kalkylbladsprogram (t.ex. Microsoft Excel eller Open Office Calc).

   a) (*3p*) Beskriv hur man exporterar en tabell från Base till Calc
   Lösning: Se din egen lösning i lab 2.

   b) (*3p*) Beskriv hur man importerar en tabell från Calc till Base
   Lösning: Se din egen lösning i lab 2.

5. (*Totalt: 13p*) I ett fastighetsregister har databasen följande struktur:

   Fastighet(Fastighetsbeteckning, Adress)
   Äger(Fastighetsbeteckning,Pnr, Kontraktsdatum)
   Person(Pnr, Namn, BostadsAdress)

   a) Skriv en SQL-fråga som skapar en lista med namn på personer som äger fastigheter som inte bytt ägare på 2000-talet.

      i) (*2p*) Skriv ovanstående fråga med hjälp av *join*.
      Lösning:
      ```
      SELECT "Namn"
      FROM "Person" JOIN "Äger"
      ON "Person"."Pnr" = "Äger"."Pnr"
      WHERE "Kontraktsdatum" < '2000-01-01'
      ```

      ii) (*2p*) Skriv ovanstående fråga med hjälp av två ihopkopplade *Select*-frågor.
      Lösning:
      ```
      SELECT "Namn" FROM "Person"
      WHERE "Pnr" IN
      ( SELECT "Pnr" FROM "Äger"
      WHERE "Kontraktsdatum" < '2000-01-01' )
      ```

      iii) (*2p*) Skriv ovanstående fråga utan *join* och med bara en *Select*-fråga.
      Lösning:
      ```
      SELECT "Namn"
      FROM "Person", "Äger"
      WHERE "Person"."Pnr" = "Äger"."Pnr"
      AND "Kontraktsdatum" < '2000-01-01'
      ```

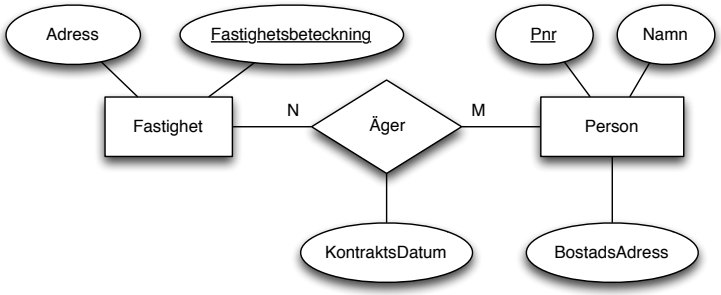
      iv) (*2p*) Antag att man nu vill inkludera fastighetsbeteckningar i listorna ovan. Dvs, man vill skapa en lista med två kolumner, ägare (Namn) till fastigheter och fastighetsbeteckning för fastigheten, och ta med alla fastigheter i databasen som inte bytt ägare på 2000-talet.

För vart och ett av de 3 alternativen ovan, (i),(ii) och (iii), beskriv om det går att ordna med en liten modifikation av SQL-frågan, och i sådana fall hur man gör.
Lösning: I fall (i) och (iii) ovan räcker det att lägga till `,Fastighetsbeteckning` efter `Namn` på första raden. I (ii) går det inte att fixa med en liten modifikation.

b) (*3p*) Skapa en lista på alla personer (Namn) som bor i en fastighet de själva äger.
Lösning:

```
SELECT "Namn"
FROM "Person", "Äger", "Fastighet"
WHERE "Person"."Pnr" = "Äger"."Pnr"
AND "Äger"."Fastighetsbeteckning"
= "Fastighet"."Fastighetsbeteckning"
AND "Bostadsadress" = "Adress"
```

c) (*2p*) Rita upp en ER-modell som skulle resultera i ovanstående Databasstruktur. Motivera varje steg genom angivande av vilka regler i ”kokboken” som använts.
Lösning: Se figur 3. Person och Fastighet verkar vara vanliga entitetstyper, med några vanliga attribut. Äger har en kombinerad primärnyckel som består av primärnycklarna hos Person och Fastighet. Vi gissar att det är ett N:M samband. Denna gissning verifieras sedan när vi kollar att ER-modellen verkligen ger upphov till den givna databasstrukturen. Kontraktsdatum blir då ett attribut på sambandet.



Figur 3: En ER-modell som ger databasstrukturen i uppgiften.

För att motivera vår lösning (så att vi får full poäng) och vara säkra på att vi gjort rätt går vi igenom reglerna för överföring mellan ER-modell och databasstruktur.
Steg 1: Varje vanlig entitetstyp blir en tabell. Vanliga attribut blir kolumner i tabellerna.
Vi får alltså följande:

Fastighet(Fastighetsbeteckning, Adress)

Person(Pnr, Namn, BostadsAdress)

Steg 2: (Vi har inga 1:N-samband, alltså ingen åtgärd)

Steg 3: (Vi har inga 1:1-samband, alltså ingen åtgärd)

Steg 4: Varje N:M-samband blir en egen tabell. Vi får:

Fastighet(Fastighetsbeteckning, Adress)

Äger(Fastighetsbeteckning,Pnr)

Person(Pnr, Namn, BostadsAdress)

Steg 5: (Vi har inga flervägssamband, alltså ingen åtgärd)

Steg 6: Attribut på samband blir kolumner. Vi får:

Fastighet(Fastighetsbeteckning, Adress)

Äger(Fastighetsbeteckning,Pnr, Kontraktsdatum)

Person(Pnr, Namn, BostadsAdress)

Steg 7: (Vi har inga svaga etitetstyper, alltså ingen åtgärd)

Steg 8: (Vi har inga sammansatta attribut, alltså ingen åtgärd)

Steg 9: (Vi har inga flervärda attribut)

Steg 10: Härledda attribut tas inte med i databasen.

Vi får alltså följande:

Fastighet(Fastighetsbeteckning, Adress)

Äger(Fastighetsbeteckning,Pnr, Kontraktsdatum)

Person(Pnr, Namn, BostadsAdress)

Vilket stämmer med den givna databasstrukturen. Därför är vår ER-modell rätt.