Tentamen för DD1370
Databasteknik och informationssystem

16 Januari 2015

Hjälpmedel:
Inga hjälpmedel utom papper och penna

Tänk på:
Skriv *högst* en uppgift på varje blad.
Använd *endast framsidan* på varje blad.
Skriv *namn och personnr* 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 (tenta+bonus) på 40 ger säkert godkänt.
(lösningsförslag kommer på kurswebben)

Lycka till,

Petter

1. (*Totalt: 23p*)

   a) (*2p*) Vad är en Vy, hur skapar man en Vy i Base, och vad använder man den till?

   Lösning: En vy är en SQL-fråga som fått ett eget namn, och som sedan kan användas i andra SQL-frågor på samma sätt som tabellerna i databasen. Man skapar den genom att klicka på Tabeller/Skapa Vy..., och sedan skriva in SQL-frågan. Den används till att samla bearbetad information från resten av databasen i ett format som ser ut som de andra tabellerna. (I standard-SQL skriver man Create View X As)

   b) (*3p*) Förklara skillnaden mellan 1:1, 1:N och N:M-samband. Hur avgör man vilken typ ett samband är? Ge exempel på alla 3 typerna.

   Lösning: Som exempel noterar vi att en person kan *köra* en bil, *äga* en bil och *ha kört* en bil.

   Förutom att dessa samband innebär olika saker i praktiken är de dessutom olika vad gäller antalet deltagande Entiteter på varje sida av sambandet.

   Kör är ett så kallat 1:1-samband. 1 person kan bara styra 1 bil åt gången, och 1 bil kan bara styras av 1 person åt gången.

   Äger är ett 1:N samband. 1 person kan äga en eller flera bilar, men en bil kan bara ägas av en person. Det senare kan tyckas lite märkligt, men enligt svensk lag kan man inte samäga bilar.

   Har kört är slutligen ett N:M samband. Flera personer kan ha kört samma bil, och en person kan ha kört flera olika bilar.

   Se kompendiet för en formell beskrivning av hur man avgör vilken typ ett samband är.

   c) (*2p*) Förklara skillnaden mellan group by, och order by. Ge ett exempel på användning av group by och ett exempel på användning av order by.

   Lösning: group by är ett SQL-uttryck som används för att gruppera data för aggregerade funktioner så som sum eller avg. order by är ett SQL-uttryck för att sortera en tabell. Exempel finns i kompendierna.

   d) (*1p*) Givet följande databasstruktur:

   Bil(RegNr,Modell, Tillverkningsår)

   Skriv en SQL-fråga för att lista alla bilar med Q som tredje bokstav i registreringsnumret.

   Lösning:

```
SELECT * FROM Bil WHERE RegNr LIKE '__Q___'
```

   e) (*4p*) Förklara förkortningarna, och beskriv skillnaden mellan OLAP och OLTP och ge exempel på användning.

   Lösning: OLAP - On Line Analytical Processing. OLTP - On Line Transaction Processing. OLAP används för beslutsstödssystem, mycket data som ändras sällan, komplexa frågor som skapas efter hand. Ex: aktiekurser eller fastighetspriser. OLTP används för

driftdatabaser, data ändras ofta, återkommande frågor. Ex: varulager, kund- och order-hantering.

f) (*3p*) Vad är skillnaden på en partiell nyckel och en primärnyckel?

Lösning: En primärnyckel är unik och identifierar varje rad i tabellen. En partiell nyckel tillhör en svag entitet i en ER-modell. Den svaga entiteten får sedan en primärnyckel som består av kombinationen av den partiella nyckeln och primärnyckeln från den identifierande entiteten, se kompendiet.

g) (*3p*) Antag att tabellerna A och B är givna enligt nedan.

A=

Namn	Nummer
Kalle	2
Lisa	5
Kim	5

B=

Namn	Nummer
Adam	2
Anna	4

Rita tabellen man får som resultat av:

```
SELECT * FROM A,B WHERE A.Nummer=5.
```

Lösning: Resultatet blir följande

Namn	Nummer	Namn	Nummer
Lisa	5	Adam	2
Lisa	5	Anna	4
Kim	5	Adam	2
Kim	5	Anna	4

h) (*2p*) Antag att tabellerna A och B är givna enligt ovan. Rita tabellen man får som resultat av

```
SELECT * FROM A JOIN B ON A.Nummer=B.Nummer.
```

Lösning: Resultatet blir följande

Namn	Nummer	Namn	Nummer
Kalle	2	Adam	2

i) (*2p*) Antag att tabellerna A och B är givna enligt ovan. Rita tabellen man får som resultat av

```
SELECT * FROM A JOIN B ON A.Nummer=5.
```

Lösning: Resultatet blir följande

Namn	Nummer	Namn	Nummer
Lisa	5	Adam	2
Lisa	5	Anna	4
Kim	5	Adam	2
Kim	5	Anna	4

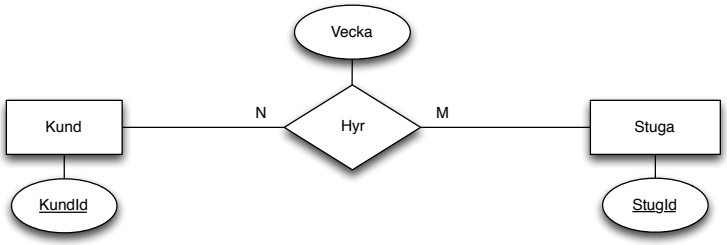
(alltså samma som ovan).

2. (*Totalt: 7p*) Betrakta ER-modellen i Figur 1.

a) (*2p*) Överför ER-modellen i figur 1 till en Databasstruktur. I just denna (lilla) uppgift behöver ni inte motivera och räkna upp vilka regler ni använder, Det räcker med svaret.

Lösning:

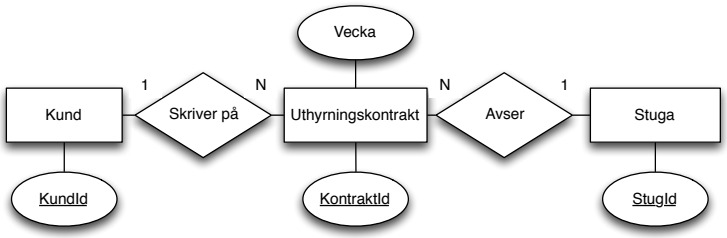
Kund(<u>KundId</u>)



Figur 1: En enkel ER-modell.

Stuga(StugId)
Hyr(KundId,StugId, Vecka)

b) (*3p*) Vad blir resultatet av att objektifiera sambandet Hyr? Rita en ny ER-modell som visar resultatet av objektifieringen. Överför den nya ER-modellen till en databasstruktur.
Lösning:



Figur 2: Resultatet av objektifieringen av sambandet Hyr i figur 1. Notera att man även får poäng för andra rimliga namn på de två nya sambanden och det nya nyckelattributet.

Kund(KundId)
Stuga(StugId)
Hyr(KontraktId,KundId,StugId, Vecka)

c) (*2p*) Varför vill man ibland objektifiera samband? Använd exemplet ovan i din förklaring. Vad skiljer i ER-modellerna? Vad skiljer i databasstrukturerna?
Lösning: Syftet med att objektifiera ett samband är att möjliggöra flera samband mellan samma par av entiteter. T.ex. flera uthyrningar av samma stuga till samma kund. Skillnaden är att den sammansatta primärnyckeln som uppstod då N:M-sambandet översattes har ersatts av en enda primärnyckel. Kolumnerna KundId och StugId finns fortfarande kvar, men nu som resultatet av två 1:N samband.

3. (*Totalt: 13p*) Mäklarfirman Elit använder en databas för att administrera sin verksamhet. Databasen har följande struktur:

Mäklare(Pnr)

Email(Pnr,Email)

Anställd(PNr, Förnamn, Efternamn, Anställningsdatum, Kön, Lön)

HarKunderIOmråde(Pnr,PostNr)

Objekt(ObjektId, Gatuadress, PostNr, Yta, Utropspris, Balkong, Slutpris, Pnr, Kontraktsdatum)

Område(PostNr, Namn)

Visning(VisningsNr, ObjektId, Datum)

a) (*7p*) Rita upp en ER-modell som skulle resultera i ovanstående Databasstruktur. Modellen skall innehålla minst ett sammansatt attribut. Motivera varje steg genom att mkt kort beskriva den regel i ”kokboken” som använts. Om du ser flera möjliga alternativa ER-modeller, så välj det alternativ som gör att man kan skriva vettiga SQL-frågor i uppgift 4 nedan.

Lösning:

Se figur 3. Steg 1 i kokboken (Varje vanlig entitetstyp blir en tabell, attribut blir kolumner) verkar ha tillämpats på *Mäklare*, *Anställd*, *Objekt* och *Område*. Gör vi dem till egna entiteter med attribut så får vi ungefär rätt databasstruktur för dessa fyra. Dock väntar vi lite med Visning, Email och HarKunderIOmråde, vilka kanske är resultat av någon annan regel.

Steg 2 i kokboken (Varje 1:N-samband blir referensattribut i “många”-sidans tabell) verkar ha tillämpats på ett samband mellan Objekt och Område, vi kallar det sambandet Ligger i. Samma sak med Mäklare och Område, vi kallar det sambandet Säljs av.

Steg 3 handlar om 1:1-samband, något sådant verkar vi inte ha.

Steg 4 (Varje N:M-samband bildar egen tabell) verkar ha gett upphov till tabellen *HarKunderIOmråde*.

Steg 5 handlar om flervägssamband, något sådant verkar inte finnas.

Steg 6 handlar om attribut på samband, något sådant verkar inte finnas.

Steg 7 (svaga entiteter bildar egen tabell, primärnyckeln utgörs av kombinationen av den svaga entitatens partiella nyckel, och den identifierande entitettypens primärnyckel). Denna regel verkar ha gett upphov till *Visning*. Det verkar rimligt att VisningsNr är partiell nyckel (unikt för varje Objekt/Lägenhet, men inte unikt för hela databasen). Vi ritar den med streckad understykning.

Steg 8 (sammansatta attribut blir som delarna). Det var givet att vår modell skall innehålla minst ett sammansatt attribut, vi låter Namn vara sammansatt av Förnamn och Efternamn.

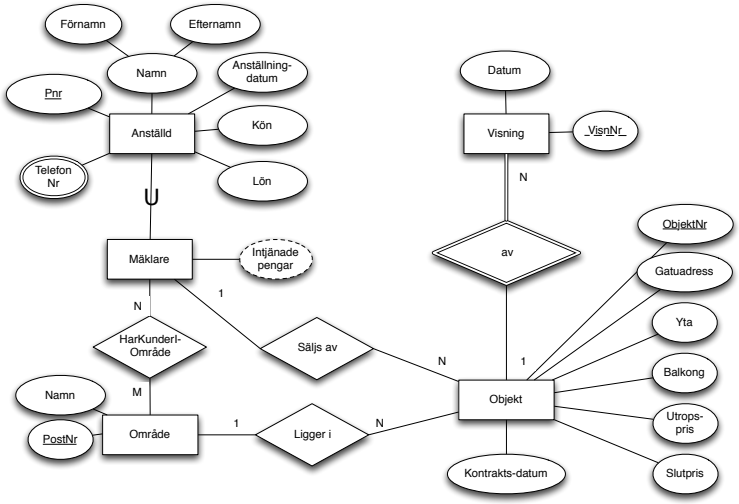
Steg 9 (Varje flervärt attribut blir en egen tabell. Primärnyckeln består av entitetstypens primärnyckel, kombinerad med det flervärda attributet.)

Email ser ut som resultatet av denna regel.

Steg 10 handlar om härledda attribut. Dessa syns inte i databasstrukturen.

Steg 11 (Varje subklass blir en egen tabell) Det verkar rimligt att tro att Mäklare är en subklass av anställda, annars är vi inte säkra på att alla mäklare finns med i anställda-tabellen, och kan inte kolla upp löner för mäklarna i uppgift 4d.

Tillämpar vi reglerna i kokboken på ER-modellen i figur 3 kommer vi således få den givna databasstrukturen.



Figur 3: En ER-modell som skulle ge upphov till den givna databasstrukturen. Notera att *Intjänade pengar* tillkommer i deluppgift 3c

b) (*4p*) Ange för varje attribut i Databasstrukturen vilken datatyp som passar. Motivera dina svar. (Notera att BIGINT är en datatyp som skapats för att hantera extra stora heltal, den nämndes kort på sista föreläsningen)

Lösning:

Varchar (text): Förnamn, Efternamn, Kön, Gatuadress, Namn

Integer (heltal): Lön, PostNr, ObjektId, Utropspris, Slutpris, VisningsNr

BigInt (stora heltal):Pnr

Float (flyttal): Yta

Date: Anställningsdatum, Kontraktsdatum, Datum

Boolean (sant/falskt): Balkong

Notera att vissa attribut kan väljas antingen som BigInt, Integer eller Float, detta gäller t.ex. Lön, Yta och Priser.

c) (*2p*) Hur skulle ER-modell och databasstruktur ändras om man lade till ett härlett attribut ‘Intjänade pengar’ till Mäklarna?

Lösning: I figuren skulle vi få en ny sträckad oval som det står “Intjänade pengar” i, med enkelstreck till Mäklar-entiteten, se figur 3.

Databasstrukturen ändras inte eftersom de härledda attributen inte innehåller någon egen data. Istället kan man lägga till en vy för att beräkna det härledda attributet.

4. (*Totalt: 23p*) Givet samma databasstruktur som i frågan ovan, dvs

   Mäklare(Pnr)

   Email(Pnr,Email)

   Anställd(PNr, Förnamn, Efternamn, Anställningsdatum, Kön, Lön)

   HarKunderIOmråde(Pnr,PostNr)

   Objekt(ObjektId, Gatuadress, PostNr, Yta, Utropspris, Balkong, Slutpris, Pnr, Kontraktsdatum)

   Område(PostNr, Namn)

   Visning(VisningsNr, ObjektId, Datum)

   Skriv SQL-frågor som löser följande uppgifter.

   a) (*1p*) Lista alla Gatuadresser för de objekt som sålts under Juni 2014.
   Lösning:

   ```
   SELECT "Gatuadress" FROM "Objekt"
   WHERE "Kontraktsdatum" <= '2014-06-30'
   AND "Kontraktsdatum" >=  '2014-06-01'
   ```

   b) (*1p*) Lista all information om alla anställda som inte är mäklare.
   Lösning:

   ```
   SELECT * FROM "Anställd"
   WHERE "Pnr" NOT IN (SELECT * FROM "Mäklare")
   ```

   c) (*2p*) Lista snittlöner för män/kvinnor i företaget, beräknat på alla anställda.
   Lösning:

   ```
   SELECT "Kön", AVG( "Lön" )
   FROM "Anställd"
   GROUP BY "Kön"
   ```

   d) (*2p*) Använd en nästlad SQL-fråga för att lista snittlöner för män/kvinnor som jobbar som mäklare i företaget.
   Lösning:

   ```
   SELECT "Kön", AVG( "Lön" )
   FROM "Anställd"
   WHERE "Pnr" IN (SELECT * FROM "Mäklare")
   GROUP BY "Kön"
   ```

   e) (*1p*) Lista Efternamn på de anställda som anställts under 2014 och vars efternamn börjar på A, byt namn på namnkolumnen till *Nyanställda-A*.
   Lösning:

```
SELECT "Efternamn" AS "Nyanställda-A"
FROM "Anställd"
WHERE "Efternamn" LIKE 'A%'
AND "Anställningsdatum"<= '2014-12-31'
AND "Anställningsdatum">= '2014-01-01'
```

f) (*1p*) Lista för och efternamn på alla som har kunder i postnummerområde 17286.

Lösning:

```
SELECT "Förnamn", "Efternamn"
FROM "Anställd"
WHERE "Pnr" IN
(SELECT Pnr FROM "HarKunderIOmråde"
WHERE "PostNr"=17286)
```

g) (*3p*) Lista för och efternamn på alla som har kunder i Danderyd eller Lidingö.

Lösning:

```
SELECT "Förnamn", "Efternamn"
FROM "Anställd"
WHERE "Pnr" IN
(SELECT "Pnr" FROM "HarKunderIOmråde"
WHERE "PostNr" IN
(SELECT "PostNr" FROM "Område"
WHERE "Namn"='Danderyd'
OR "Namn"='Lidingö'))
```

h) (*2p*) Beräkna företagets totala intäkter, givet att man tar ut en avgift på 15% av försäljningspriset. Kalla kolumnen Intäkt.

Lösning:

```
SELECT SUM( "Slutpris" * 0.15 )
AS "Intäkt" FROM "Objekt"
```

i) (*2p*) Beräkna företagets totala intäkter, givet att man tar ut en avgift på 5% av slutpriset och 50% av skillnaden mellan slutpris och utropspris. Kalla kolumnen Intäkt.

Lösning:

```
SELECT SUM( "Slutpris" * 0.05+
("Slutpris"-"Utropspris")*0.5)
AS "Intäkt" FROM "Objekt"
```

j) (*2p*) Lista personnummer på alla mäklare som sålt något objekt, sorterade efter hur mycket de sålt för, med den största summan först.

Lösning:

```
SELECT "Pnr", SUM( "Slutpris" )
FROM "Objekt" GROUP BY "Pnr"
ORDER BY SUM( "Slutpris" ) DESC
```

k) (*2p*) Lista efternamn på alla mäklare som sålt något objekt, sorterade efter hur många objekt de sålt. Notera att vi räknar ett objekt som sålt om kolumnen Slutpris är ifylld.

Lösning:

```
SELECT "Efternamn", COUNT( "Slutpris" )
FROM "Objekt","Anställda"
WHERE "Objekt"."Pnr"="Anställda"."Pnr"
GROUP BY "Pnr"
ORDER BY COUNT( "Slutpris" ) DESC
```

l) (*4p*) Hitta de misslyckade uppdragen, dvs de objekt som har mer än 2 visningar men ännu inget slutpris!

Lösning:

```
SELECT "Objekt"."ObjektId", COUNT( * )
FROM "Objekt", "Visning"
WHERE "Objekt"."ObjektId" = "Visning"."ObjektId"
AND "Objekt"."Slutpris" IS NULL
GROUP BY "Objekt"."ObjektId"
HAVING COUNT( * ) > 2
```