Inst. för Data- och Systemvetenskap
SU/KTH

Tentamen

EIT:DB, SP:DB, DVK:DB, ATD:DB, FK:DB, DSK:DB

Ska även användas av äldre kurstillfällen, t ex för IV1018, IV1003, ITK3:DB, ITK1:DB, DSV1:DB mfl.

Databasmetodik
Lördag 27 september 2014 kl. 10 - 14

Skriv bara på en sida av pappret
Skriv läsligt, annars kan det vara omöjligt att rätta tentamen

Lycka till!

Uppgift 1 (Databasteknik och informationsadministration)

Del 1 till 10 är flervalsfrågor där rätt alternativ ska väljas (bara *ett* är rätt). Välj ***ett*** alternativ för var och en av flervalsfrågorna nedan.

1) Vilket av följande gäller för 2PL (Two-Phase Locking)?
a) En transaktion kan begära lås och släppa lås om vartannat och när som helst under sin livstid.
b) Under vissa omständigheter kan en transaktion få ett nytt lås trots add en redan släppt ett lås.
c) En transaktion kan bara begära lås så länge den inte har släppt ett lås.

C.

2) Normalisering syftar till att eliminera redundans i databasen. Vad kan redundans leda till?
a) Att det inte går att lägga till nya tabeller i efterhand om det skulle visa sig behövas.
b) Att man inte kan garantera att databasens data är konsistent och komplett efter INSERT, DELETE och UPDATE.
c) Att alla tabeller i databasen måste ges en surrogatnyckel för att garanterat undvika dubblett-rader.

B.

3) Vilket av följande påståenden gäller för en kandidatnyckel?
a) Den kan inte ha dubletter, men den behöver inte vara minimal.
b) Ingen del av den kan vara NULL, men den behöver inte vara minimal.
c) Den kan inte ha dubletter, och måste vara minimal.

C.

4) Vilket påstående är korrekt?
a) Kardinaliteten anger antalet *främmande nycklar* i en tabell.
b) Graden anger antalet *kolumner* i en tabell.
c) Kardinaliteten anger antalet *kolumner* i en tabell.

B.

5) Vilken av grupperna har med begränsning av vilka värden som kan matas in i databasen att göra?
a) Selektion, 3NF, GROUP BY.
b) Intension, extension, deadlock.
c) Entitetsintegritet, referensintegritet, assertion.

C.

6) Vilket av följande är ett krav på en relation/tabell för att den ska följa relationsmodellen?
a) Varje kolumn i relationen/tabellen måste ha atomära värden från en bestämd domän (eller eventuellt vara NULL).
b) Det måste finnas minst en främmande nyckel i relationen/tabellen.
c) Den måste vara i minst 3NF.

A.

7) Vad behövs för att göra en återställning (recovery) av en databas efter en systemkrasch?

a) Transaktionsloggar enbart.

b) Transaktionsloggar och senaste databasbackupen för att kunna köra om alla loggade transaktioner.

c) Transaktionsloggar och databasen efter kraschen för att kunna jämföra vilka transaktioner som måste köras om.

8) Vad är en trigger?

a) En tabell som tillkommit på grund av en M:M-association i konceptuella modellen.

b) Ett databasobjekt som kan användas för att implementera komplexa verksamhetsregler.

c) Ett databasobjekt som schemalägger databasbackuper.

B.

9) Vilket av följande påståenden är sant om entitetsintegritet?
a) Ett primärnyckelvärde får aldrig ändras efter att det lagts in.
b) En primärnyckel får bara bestå av en kolumn, och den får inte vara NULL.
c) En primärnyckel kan vara sammansatt av flera kolumner, och ingen av de kolumnerna får vara NULL.

C.

10) För vilka operationer nedan är det krav på unionskompatibilitet?
a) Natural join och theta-join.
b) Union och kartesisk produkt.
c) Snitt och differens.

C.

11) Förklara vad var och en av bokstäverna i ***ACID*** står för och innebär. För svar räcker det alltså inte med att bara skriva ut respektive term, utan man måste också förklara vad begreppen som termerna pekar på innebär. (Max 1 A4-sida totalt!)

Generellt gäller att ACID har med transaktionshantering att göra och beskriver vilka egenskaper transaktioner måste ha:

A – Atomicity: Innebär att endast "allt eller inget" accepteras vid transaktioner. Allt som ingår i transaktionen ska genomföras, eller inget alls av det. Transaktionen ses som en odelbar enhet. Om något fel uppstår under transaktionen ska ROLLBACK utföras, dvs allt som tidigare gjorts under transaktionen ska göras ogjort och transaktionen avslutas.

C – Consistency: Innebär att DBHS:t måste se till att alla begränsningar, t ex integritetsregler, som gäller för en databas också efterföljs vid varje transaktion så att databasen inte befinner sig i ett icke--konsistent läge (dvs ett läge som är otillåtet enligt begränsningarna) vare sig före eller efter transaktionen. Man kan också hävda att ett visst ansvar för consistency även faller på den som matar in data i databasen, så att rätt data hamnar rätt, för i vissa fall är det omöjligt för ett DBHS att kunna verifiera viss inmatning.

I – Isolation: Innebär att DBHS ska se till att alla transaktioner körs oberoende av varandra. En transaktion ska inte kunna se, och påverkas av, delresultat från andra icke-slutförda transaktioner.

D – Durability: Innebär att effekterna av en fullt och korrekt slutförd transaktion, ska skrivas fullt bestående, permanent, till databasen. Dvs även om någon komplikation senare uppstår efter transaktionen (t ex en diskkrasch) ska inget förloras. Det ska gå att återställa allt data som det var före komplikationen uppstod.

Uppgift 2 (konceptuell modellering)

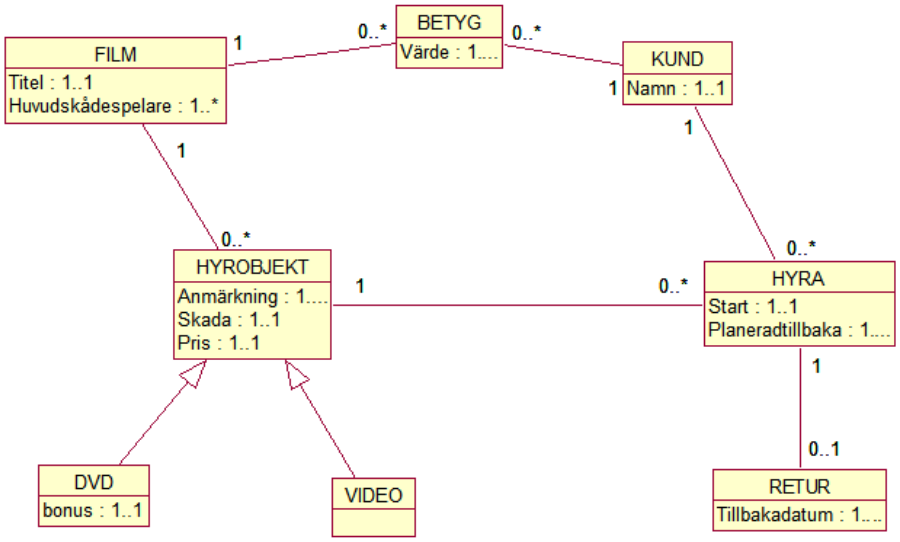
En musikaffär hyr ut filmer på video och DVD till sina kunder. Konstruera ett konceptuellt schema, t.ex. i form av ett UML klassdiagram, som klarar av att hantera informationskraven nedan:

- Vilka kunder har lämnat in någon film för sent?
- Vilka kunder har hyrt filmer vid mer än tre tillfällen under 2006?
- Vilken är den mest populära filmen?
- Vilken film har fått högst betyg av kunder?
- Vilka filmer finns på både video och DVD?
- Vilka filmer har fått en anmärkning för att de inte gått att spela upp?
- När skall Pelle Persson lämna tillbaka den film han hyrde 061012?
- Vilka skivor har blivit skadade?
- Vilka videoband har blivit skadade?
- Vilka kunder har hyrt filmer enbart på DVD?
- Vad kostar det att hyra ”Borta med vinden” på DVD?

- Vad kostar det att hyra ”Borta med vinden” på video?
- Vilka är huvudskådespelarna i ”Borta med vinden”?
- Hur lång är filmen ”Borta med vinden”?
- Vilka DVD-filmer har bonusmaterial?
- Har Pelle Persson och Nisse Nilsson hyrt samma DVD-skiva?

Sträva efter att modellera på ett sådant sätt att partiella attribut (attribut med minimum-värde=0) undviks! Ange multiplicitet (avbildnigsregler) för alla associationer och attribut!

Lösningsförslag:



Uppgift 3 (analytisk databasdesign)

Betrakta följande relationsschema:

ANSTÄLLD(Anställningsnummer, Rumsnummer, Lönegrad, Avdelning, Antal_kvadratmeter)

Följande funktionella beroenden råder:

Anställningsnummer → Rumsnummer, Lönegrad, Avdelning

Rumsnummer → Antal_kvadratmeter

a) Bestäm primärnyckel för ANSTÄLLD. Motivera ditt val.

Lösningsförslag: PN = Anställningsnummer pga att denna bestämmer alla övriga attribut funktionellt (= om vi anger ett anställningsnummer så ger detta nummer max ett värde på alla övriga kolumner).

b) Vilken normalform är tillämplig för ANSTÄLLD? Motivera ditt val.

Här kan det, beroende på hur man svarade i a) bli olika svar. Jag har relaterat alla svar till valet i a). Har man alltså svarat fel på a) men sen konsekvent använt sig av sitt svar på ett korrekt sätt i b) så beaktas detta. Mitt svar här baserar sig på lösningen till a). Eftersom vi har en icke-sammansatt PN så är vi i minst 2NF med avseende på den icke sammansatta primärnyckeln (givet att alla attribut är atomära, annars bara i 0 NF). Vi är dock inte i 3NF pga att vi har ett transitivt beroende mellan icke nyckel-kolumnen 'Antal_kvadratmeter' och primärnyckeln.

För att nå 3NF bryter vi ut kolumnen 'Antal_kvadratmeter' tillsammans med den kolumn som (förutom primärnyckeln) bestämmer 'Antal_kvadratmeter' funtionellt, dvs kolumnen 'Rumsnummer':

RUM(**Rumsnummer,** Antal_kvadratmeter)

Kvar i den gamla tabellen blir:

ANSTÄLLD(**Anställningsnummer,** Rumsnummer, Lönegrad, Avelning) med en ny främmande nyckel: ANSTÄLLD.Rumsnummer är FN mot RUM.Rumsnummer.

c) Visa via exemplet ovan och motivera i text varför en högre normalform är att föredra framför en lägre.

En högre normalform är bättre främst av konsistens och redundansskäl. Med en högre normalform slipper man repetera information (annat än på ett kontrollerat sätt via främmande nycklar, vilket databashanteringssystemet dessutom har stöd för att hålla konsistenta). I vårt fall här måste man (eftersom vi inte uppfyller 3NF) upprepa antal kvadratmeter för ett visst rum på varenda rad där detta rum förekommer.

Gammal tabell:

ANSTÄLLD

Anställningsnummer	Rumsnummer	Lönegrad	Avelning	Antal_kvm
11111	1	A	Admin	25
22222	13	B	Städ	10
33333	1	A	Städ	25
44444	1	A	Städ	25

Nya tabeller:

ANSTÄLLD

Anställningsnummer	Rumsnummer	Lönegrad	Avelning
11111	1	A	Admin
22222	13	B	Städ
33333	1	A	Städ
44444	1	A	Städ

RUM

Rumsnummer	Antal_kvm
1	25
13	10

Här har vi alltså mindre redundans, i detta fall = storleken på ett rum (Antal kvadratmeter) förekommer nu bara på ETT ställe, inte på varje rad där någon bor i ett rum.

Uppgift 4 (syntetisk databasdesign)

Betrakta följande relationsdatabasschema:

FLOD(**Namn**, Längd)

LAND(**Namn,** Yta)

FLYTER_GENOM(**Flod, Land**)

(Primärnycklar är angivna med fetstil.)

FLYTER_GENOM.Flod utgör främmande nyckel mot FLOD.Namn

FLYTER_GENOM.Land utgör främmande nyckel mot LAND.Namn

CREATE TABLE-satser för FLOD och LAND följer nedan:

CREATE TABLE FLOD AS (Namn varchar(25) NOT NULL, Längd integer NOT NULL, primary key(Namn));

CREATE TABLE LAND AS (Namn varchar(25) NOT NULL, Yta integer NOT NULL, primary key(Namn));

Realisera nu tabellen FLYTER_GENOM (se ovan) som en CREATE TABLE-sats. Definitionerna ska inkludera sk. key business rules, dvs regler för vad som händer om man tar bort eller förändrar det som de främmande nycklarna i FLYTER_GENOM refererar till. Motivera key-business reglerna.

Syntax för CREATE TABLE (se även exemplen ovan men de saknar främmande nycklar):

CREATE Tabellnamn AS(Kolumnnamn$_1$ datatypnamn$_1$eventuellt NOT NULL , Kolumnnamn$_2$ datatypnamn$_2$ eventuellt NOT NULL, …,Kolumnnamn$_n$ datatypnamn$_n$ eventuellt NOT NULL, primary key(välj ett eller flera kolumnnamn), foreign key$_1$ (välj ett eller flera kolumnnamn) references (välj annat tabellnamn) ON DELETE välj en effekt ON UPDATE välj en effekt, …, foreign key$_n$ (välj ett eller flera kolumnnamn) references (välj annat tabellnamn) ON DELETE välj en effekt ON UPDATE välj en effekt);

Lösningsförslag:

CREATE TABLE FLYTER_GENOM(Flod varchar(25) NOT NULL, Land varchar(25) NOT NULL, primary key(Flod, Land), FOREIGN KEY(Land) REFERENCES LAND(Namn) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY(Flod) REFERENCES FLOD(Namn) ON DELETE CASCADE ON UPDATE CASCADE)

Motiv till FN-reglerna: Om man tar bort ett LAND bör även motsvarande referens till det landet = en rad i tabellen FLYTER_GENOM som gäller det landet tas bort. Pss om man tar bort en FLOD. Samma sak om man ändrar namn på en flod eller ett land: detta bör slå igenom i de tabeller där namnen på floder och länder förekommer som främmande nyckel-värfden. Man kan också tänka sig att inte tillåta vare sig borttag eller ändring av floder och länder – i så fall ändrar vi de två FOREIGN KEY-reglerna ovan till RESTRICT istället för CASCADE.

Uppgift 5 (Frågespråk: SQL och relationsalgebra)

Betrakta följande tabeller (relationsscheman):

DANSARE(**Namn**, Skostorlek, Längd)
BALETT(**Namn**, Upphovsman)
ROLLINNEHAV(**Dansare, Balett, Från, Till**)

(Primärnycklar är angivna med fetstil.)

ROLLINNEHAV.Dansare utgör främmande nyckel mot DANSARE.Namn
ROLLINNEHAV.Balett utgör främmande nyckel mot BALETT.Namn

Formulera följande frågor i relationsalgebra:
a) Vilka dansare är längre än 1.60 m?
b) Vilka dansare har dansat i alla baletter?

a) $\pi_{Namn}\sigma_{Längd\ >160}$ (DANSARE)
b)
NÄMNARE(Balett) ← π_{Namn} (BALETT)
Täljare ← $\pi_{Dansare,Balett}$ (ROLLINNEHAV)

Svar ← TÄLJARE KVOT NÄMNARE

Formulera följande frågor i SQL:

a) Vilka dansare är längre än 1.60 m?

SELECT Namn FROM
DANSARE WHERE
Längd > 1.60

b) Vilken dansare är längst?

SELECT Namn
FROM DANSARE
WHERE Längd = (SELECT MAX(Längd) FROM DANSARE)

I exemplen ovan ska koden vara generell och inte basera sig på att exempeldatabasen nedan ser ut på ett visst sätt (just nu).

Relationsalgebraiska operatorer:

σ = SELECT , Π = PROJECT, $\bowtie$ = JOIN,

$\cap$ = INTERSECTION (SNITT) , $\div$ = KVOT (DIVISION) -- = DIFFERENCE, U = UNION

SQL-syntax:

```
SELECT [DISTINCT] <attributlista>
FROM <tabellista>
[WHERE <villkorsuttryck>]
[GROUP BY <kolumnlista>
    [HAVING <villkorsuttryck>]]
[ORDER BY <KOLUMNLISTA>];
```