Tentamen i EDA216

14 mars 2017

Skrivtid: 8-13

- Skriv bara på ena sidan av pappret – tentorna kommer att scannas in, och endast framsidorna rättas.
- Skriv **inte** med färgpenna – enda tillåtna färg är svart/blyerts.
- Skriv tydligt – om texten inte går att läsa kan du inte få några poäng.
- Sätt identitet och sidnummer på varje inlämnat blad, kontrollera att sidnumret på din sista sida är samma som det antal blad du markerar på omslagspappret.
- Preliminär maxpoäng på uppgifterna: 13 + 13 + 13 + 4 + 4 + 3.

Uppgift 1

Vi vill utveckla en databas för att hantera rättning av tentor. Varje tenta som ges är kopplad till en kurskod, och den ges vid ett givet datum. Tentorna består av ett antal uppgifter med ett uppgiftsnummer (1a, 1b, 2a, ...), och till varje uppgift hör en uppgiftstext. En uppgift hör alltid bara till en tentamen.

För varje uppgift finns det saker ('items') som man skall göra för att få poäng, och saker man inte skall göra, och som ger avdrag. Varje 'item' är kopplad till *en* uppgift, den har en textbeskrivning, en unik kod (för alla tentor), och ger ett givet antal poäng (plus eller minus).

En student har ett stil-id och ett namn, och måste anmäla sig till varje tentamen den vill skriva – studenten registreras då som tenterande, och får en anonym kod som används vid den aktuella tentan. Vi antar i uppgiften att denna anonyma kod är unik för samtliga tentor som någonsin skrivs.

Efter rättningen kommer varje student som tenterat (en student som tenterar kallas en *tentand*[1]) och lämnat in en lösning att få ett protokoll som visar precis vad hen fått poäng för, och vad hen har fått avdrag för – här är ett verkligt exempel från tentamen i EDA061 i höstas (varje rad motsvarar ett 'item', och på varje rad anges dess textbeskrivning och hur många poäng den ger):

Problem 4b (1.10 av 1.50 poäng)

- ☐ Tar bort `yieldLimit` från `Portfolio` (~~0.15 p~~)
- ☑ `Portfolio` notifierar varje gång (0.15 p)
- ☑ `Alarm`: håller reda på `yieldLimit` (0.05 p)
- ☑ `Alarm`: gör `addObserver` på portfölj (0.60 p)
- ☑ `Alarm`: implementerar `update(Observable,Object)` (0.45 p)
- ☑ `Alarm`: meddelar observers (0.10 p)
- ☑ Exponerar attribut (bryter mot lokalitetsprincipen) (−0.25 p)

Tentanden i detta exempel hade i sin lösning inte tagit bort attributet `yieldLimit` från klassen `Portfolio` (och missade på så vis 0.15 poäng), och hade dessutom felaktigt exponerat ett attribut (vilket gav 0.25 poängs avdrag) – i övrigt var lösningen korrekt, och gav därför 1.10 poäng av 1.50 möjliga.

[1] *Examinee* på engelska.

En eller flera granskare kommer att gå igenom de lösningar som lämnas in – för varje lösning noteras vilka 'items' som påträffas (både sådant som ger pluspoäng och sådant som ger avdrag), och vilken granskare som gjort noteringen. Noteringarna skall läggas in i databasen[2], och protokollen genereras med hjälp av dessa noteringar.

I vårt rättningssystem lägger vi redan vid tentamenskonstruktionen in de 'items' som vi vill se i en korrekt lösning. Vi markerar dem som 'standard'-items, och de skall finnas med i alla utskrivna protokoll, men utan 'tick' till vänster om de inte förekommer i lösningen (som för den första punkten i figuren ovan), så att tentanderna skall kunna se var de eventuellt tappar poäng. Maxpoängen för en uppgift beräknas som summan av poängen för dess 'standard-items'.

När vi börjar granska de inlämnade lösningarna och hittar olika slags fel så tittar vi först om det är ett 'item' som redan har lagts in i databasen – om det redan finns så noterar vi att det förekommer i den aktuella lösningen, annars lägger vi först in en ny 'item' med felet ifråga, innan vi markerar att det förekommer i den aktuella lösningen.

En granskare har en unik signatur, ett lösenord, och en email-adress, vad gäller granskarna vill vi kunna:

- se vilken granskare som lagt in en ny 'item' i databasen, och
- se vilken granskare som har noterat att en given 'item' förekommer i en given lösning.

Vi vill även kunna bestämma i vilken ordning olika 'items' för en uppgift skall skrivas ut i protokollen.

(a) Gör en E/R-modell för det rättningssystem som beskrivs i texten ovan (använd UML-notation).

(b) Definiera relationer för din E/R-modell.

(c) Skriv SQL-satser som räknar ut maximal poäng för denna uppgift (alltså uppgift '1c' på tentamen i denna kurs idag).

Uppgift 2

En student som just har läst kursen i databasteknik, vill sätta upp en hemsida där registrerade användare kan skriva och publicera artiklar. En artikel är alltså skriven av en användare och består av en unik titel, en text och tidpunkten för publicering. Användare kan också kommentera sina egna och andras artiklar. Varje användare har ett unikt nummer, (`user_id`), som genereras automatiskt, och ett unikt användarnamn, (`username`).

Databasen består av följande relationer (där primärnycklar är `understrukna`, och främmande nycklar är ***kursiva och fetstilta***).

```
articles(article_id, title, user_id, message, posted_at)
comments(article_id, user_id, comment, posted_at)
users(user_id, username, password, email)
```

Skriv SQL-satser i frågorna b)-g)

(a) Rita ett ER-diagram som beskriver databasen (använd UML-notation).

(b) Skapa tabellen `comments`.

(c) Hämta titlarna för de tio senaste artiklarna och sortera efter datum (nyast först).

(d) Hämta titlarna på alla artiklar skrivna av användaren med användarnamnet 'niklas'.

(e) Räkna hur många kommentarer artikeln med titeln 'Det finns inget roligare än databaser!' har.

(f) Lista användarnamnen för de användare som varken har skrivit artiklar eller kommentarer.

(g) Lista alla användarnamn och antalet kommentarer användaren har skrivit, men bara för användare som har skrivit fler än 5 kommentarer.

[2] Observera att det bara är noteringarna för en lösning som skall läggas in, och inte lösningstexten.

Uppgift 3
Relationen R(A, B, C, D, E) har följande funktionella beroenden:

FD1. AD $\rightarrow$ BC
FD2. AC $\rightarrow$ D
FD3. C $\rightarrow$ E

(a) Vilka nycklar finns i relationen?

(b) Visa att relationen inte är i BCNF och inte i 3NF.

(c) Dela upp relationen i mindre relationer som är i BCNF.

Uppgift 4
Vilket är målet med normalisering av databaser? I undantagsfall kan man välja att använda en relation som inte är normaliserad. Förklara kortfattat varför?

Uppgift 5
(a) Ge ett exempel på ett problem som man lätt kan lösa i vanliga programmeringsspråk, men som traditionell SQL (utan rekursiva queries) inte kan hantera.

(b) Många databashanterare kan använda sig av lagrade procedurer (*stored procedures*), vad är en lagrad procedur? Vad kan man vinna på att använda en lagrad procedur jämfört med att använda vanliga SQL-satser?

Uppgift 6
Förklara kortfattat vad *Isolation Level* och *Repeatable Read* innebär.