Tentamen i EDAF75

17 mars 2026

Skrivtid: 14:00-19:00

- Skriv tydligt: om texten inte går att läsa kan du inte få några poäng.
- Skriv bara på ena sidan av pappret: tentorna kommer att scannas in, och endast framsidorna rättas.
- Skriv ingen personlig kod på tentan: den 'personliga kod' som används på en del tentor gör tentorna mindre anonyma, så vi vill inte ha den!
- Sätt identitet och sidnummer på varje inlämnat blad: om din anonymkod skulle vara `EDAF75-0123-ABC` så skall du bara skriva `123-ABC` längst upp i högerkanten på varje inlämnat blad (hoppa över kurskod och inledande nollor) – skriv dessutom sidnumret och antal inlämnade blad under identiteten, som nedan (exemplet visar sidan 3 av 5 på tentan med anonymkoden `EDAF75-0123-ABC`):

  ```
  123-ABC
  3/5
  ```

- Kontrollera noga att du får med alla blad, och lägg dem i rätt ordning i omslaget. Om du markerar sidorna enligt ovan, *och lägger dem i ordning*, så är det mycket enklare att kontrollera att alla sidor finns med.
- *Preliminär* maxpoäng anges vid varje uppgift.
- *Preliminära* betygsgränser:

  3: Minst ca 60% på *var och en* av uppgift 1-3 (alla tre måste vara godkända).

  4: Minst ca 70% på *var och en* av uppgift 1-3, och ca 70% totalt på uppgift 4-6.

  5: Minst ca 85% på *var och en* av uppgift 1-3, och ca 85% totalt på uppgift 4-6.

Uppgift 1

I denna uppgift förekommer ett antal begrepp som kan vara svåra att översätta till engelska under tentan, om du vill så får du gärna använda svenska namn i din modell (å, ä och ö går bra!).

En andrahandsbutik för kursböcker vill ha en databas. Studenter som inte längre behöver en kursbok kan lämna in den till butiken, och om butiken kan hitta en köpare till boken så får säljaren betalt för den (minus en liten avgift till butiken).

Kursböckerna har en ISBN-kod[1], och på varje exemplar som butiken får in sätter man en unik QR-kod (om en bok säljs vidare vid ett senare tillfälle så får den en ny unik QR-kod).

Kvaliteten på böckerna som butiken får in bedöms, och butiken sätter en procentsats på varje exemplar – ett exemplar som är nästan som nytt kan få 80%, och då blir priset på boken 80 procent av nypriset, en sliten bok får en lägre procentsats.

I systemet vill vi hålla reda på förlagens rekommenderade nypriser för alla böcker som skall säljas, eftersom försäljningspriset beror av nypriset – om nypriset på en bok går upp så skall även försäljningspriset på motsvarande begagnade bok gå upp.

Böckerna skall användas av studenter som läser olika kurser, och för varje kurs håller vi reda på kurskod, kursnamn och vilka böcker som ingår i kurslitteraturen. En del kurslitteratur är 'obligatorisk', och annan är valfri ('bredvidläsning'), vi vill för varje bok som ingår i kurslitteraturen för en kurs hålla reda om den är obligatorisk eller inte (observera att en bok kan vara obligatorisk i en kurs, och valfri i en annan).

Butiken kan användas bara av studenter som har ett stil-id, både säljare och köpare skall vara inlagda i vår databas, och vi vill hålla reda på namn och adress på dem alla.

För varje bokexemplar som säljs i butiken vill vi hålla reda på vem som köpte, när köpet genomfördes, och vilket priset blev.

Du behöver inte ta med mer information i din databas än vad som beskrivs ovan.

(a) Gör en E/R-modell som beskriver databasen, redovisa med ett UML-diagram, markera nycklar, och skriv ut multipliciteter på samtliga associationer. *Skriv inte ut främmande nycklar som redan finns markerade i diagrammet.* Du behöver inte markera dina entity-sets med "`«weak»`".

Visa sedan hur din modell skall implementeras, antingen på det format som beskrivs på sidan 6 i denna tentamen, eller med SQL-satser. Visa samtliga tabeller som behövs för din E/R-modell – var noga med att deklarera primärnycklar och främmande nycklar. Om du skriver SQL-kod behöver du inte skriva med stora bokstäver, och du behöver inte göra `DROP TABLE` innan du skapar en tabell.

Du får gärna använda samma svenska namn på tabeller och attribut som du använt i din E/R-modell (se ovan). *11 p*

(b) Utgående från de tabeller du skapar i uppgift (a), skriv en SQL-sats som skriver information om varje köp under 2026 som gäller böcker som används som 'bredvidläsning' i kursen EDAF99. Utskriften skall innehålla köparnas namn, titlarna på böckerna, och de priser de betalade (du kan använda funktionen `year()` för att få året för ett datum). *2 p*

[1] I verkliga livet omarbetas kursböcker ofta, och kommer då ut i uppdaterade editioner – varje ny edition får ett nytt ISBN-nummer. För att göra denna uppgift enklare antar vi att våra kursböcker bara har en edition, och därmed bara ett ISBN-nummer.

Uppgift 2

Infektionskliniken på ett sjukhus skall undersöka effekten av antibiotikabehandling av olika sjukdomar och har utvecklat en databas med följande relationer:

```
patients(patient_id, patient_name, address)
bacteria(bacterium_name, description)
antibiotics(antibiotic_name, description, maker)
test_leaders(employee_id, employee_name)
treatments(patient_id, bacterium_name, antibiotic_name, initial_date, employee_id, effect)
```

I tabellen `test_leaders` håller vi reda på de läkare som kan vara ansvariga för olika behandlingar, det är denna person som pekas ut av `employee_id` i `treatments`. Attributet `effect` i `treatments` anger på en skala 0-10 hur effektiv behandlingen mot en viss bakterie med ett visst antibiotikum var för en viss patient. Attributet `maker` i `antibiotics` är namnet på tillverkaren av antibiotikan.

(a) Rita ett ER-diagram i UML-format för att beskriva databasen, rita diagrammet enligt anvisningarna i problem 1a. *2 p*

(b) Skriv ut första behandlingsdatum, namn och behandlande läkare (alltså namnet på den som är test-ansvarig) för alla de patienter som behandlas med 'antibiotikum-5' mot 'bakterie-C' – ordna efter i första hand första behandlingsdatum, och i andra hand efter patientens namn. *2.5 p*

(c) Skriv ut namn och tillverkare för de antibiotika som inte har använts i någon behandling. *2.5 p*

(d) Skriv ut id-nummer, namn och adress för de patienter som fått mer än en behandling. *3 p*

(e) För varje bakterie och varje antibiotikum som använts mot bakterien: skriv ut antalet behandlingar som gjorts, och den sämsta och bästa effekt man har uppmätt.

För de bakterier som inte har behandlats alls vill vi ha texten 'ej behandlad', och värdet 0 på både bästa och sämsta effekt i utskriften. Ordna i första hand på bakteriens namn, och i andra hand på den bästa sämsta-effekt man uppnått med det givna antibiotikat.

Du behöver inte få precis utskriften nedan för att få poäng, men gör så gott du kan.

Bakterie	Antibiotika	Antal behandlingar	Sämsta effekt	Bästa effekt
bakterie-A	antibiotikum-2	120	7	8
bakterie-A	antibiotikum-1	52	4	9
bakterie-B	ej behandlad	0	0	0
bakterie-C	antibiotikum-2	42	8	9
bakterie-C	antibiotikum-5	20	8	9
bakterie-C	antibiotikum-8	40	7	8
bakterie-C	antibiotikum-6	50	5	8
bakterie-D	antibiotikum-1	20	8	9
...	...	...	...	...

Tabell 1: Exempel på utskrift.

3 p

Uppgift 3

För att ingen skall tappa onödiga poäng på ett eventuellt misstag i (a), så har vi helt separerat uppgift (a) från uppgift (b)..(e).

(a) I många lagsporter, som handboll, innebandy eller fotboll, deltar varje *spelår*[2] ett antal *lag* i ett antal *serier* – varje lag spelar hela spelåret i en given serie. Varje lag har ett antal *spelare*, och spelarna knyts till laget för ett helt spelår – i detta lag får de ett tröjnummer som de använder under hela spelåret.

Definiera de funktionella beroenden som gäller mellan nedanstående attribut (använd attributnamnen som de står nedan, inför inga nya beteckningar):

- `spelår`
- `serie`
- `lag`
- `spelare`
- `tröjnummer`

Vi vill bara ha beroenden som tillför något nytt – du behöver inte skriva ut transitiva beroenden som bara är konsekvenser av de andra beroenden du angivit. *4 p*

(b) I relationen $R(A, B, C, D, E, F)$ gäller följande funktionella beroenden:

FD_1: $AB \longrightarrow C$
FD_2: $BD \longrightarrow E$
FD_3: $AD \longrightarrow BF$
FD_4: $CD \longrightarrow A$

Förklara kortfattat begreppen *nyckel* och *supernyckel*, och bestäm nycklarna för relationen R (motivera ditt svar). *2 p*

(c) Visa att relationen i (b) inte är i BCNF. *1 p*

(d) Bryt ner relationen R i (b) i mindre relationer som är i BCNF, och som kan rekonstrueras till den ursprungliga relationen utan att vi förlorar eller lägger till rader. Motivera nedbrytningen genom att *i varje steg* skriva dina relationer på nedanstående form, och *förklara de uppdelningar du gör*:

$R(A, B, C, D, E, ...)$
Beroenden: $A \longrightarrow B$, $BC \longrightarrow D$, ...
Nycklar: AC, BD, ...

Det vi är intresserade av är alltså framförallt stegen som leder fram till de slutliga relationerna, *att svara genom att bara ange relationer utan att motivera nedbrytningarna ger inga poäng.* *5 p*

(e) Visa med en SQL-sats hur vi kan återskapa den ursprungliga relationen i (b) med hjälp av relationerna som du bröt upp den i ovan. Denna uppgift ger bara poäng om lösningen i (d) motiverar att vi får tillbaka exakt samma rader. *1 p*

[2] Ett spelår sträcker sig i de flesta sporter från höst till vår (ett märkligt undantag är fotboll, där det svenska seriesystemet är nästan ensamt i Europa om att spela från vår till höst).

Uppgift 4

Det sammanlagda antalet olika antibiotika i alla de behandlingar som en läkare ansvarar för i databasen i uppgift 2 får inte vara större än fem (ingen läkare kan hålla sig tillräckligt informerad om fler läkemedel än så). Skriv en trigger som garanterar att detta krav uppfylls. *4 p*

Uppgift 5

Skriv ett Pythonprogram som med samma databas som i uppgift 2 löser nedanstående uppgift. Inläsning och utskrift skall göras i Python, men delegera inte till Pythonkoden sådant som naturligt kan hanteras i SQL.

> Ett antibiotikum anses vara effektivt mot en bakterie om det har testats i minst 32 behandlingar, och effekten av samtliga behandlingar är minst 7.
>
> Läs in namnet på en bakterie, och skriv ut namn och tillverkare för de antibiotika som visat sig vara effektiva i behandling mot den aktuella bakterien. Skriv för varje effektivt antibiotikum även ut hur många behandlingar vi genomfört med det mot den aktuella bakterien, och en lista med namnen på alla de läkare som har varit ansvariga för behandlingarna. Vi vill ha en rad per effektivt antibiotikum, med all information:
>
> ```
> Ange bakterie: bakterie-C
> Användbara antibiotika mot bakterie-C:
> ------------------------------------------
> antibiotikum-2  ACME    42  Alice, Bob, Carol
> antibiotikum-8  CRM114  40  Carol, David, Erin
> ```
>
> Om det inte finns något effektivt antibiotikum mot den aktuella bakterien så räcker det att utskriften ovan blir tom. För full poäng vill vi att varje läkares namn bara skrivs ut en gång på varje rad, och dessutom i bokstavsordning, men man kan få poäng även om man inte lyckas med det (det ger inget stort avdrag att misslyckas med det).

Databasen ligger i SQLite-filen `"antibiotics.sqlite"`.

Du skall skriva ett program som går att köra, men du behöver inte skriva några import-satser (det är underförstått att du använder `sqlite3`-biblioteket).

För att läsa in en sträng i Python kan du använda följande anrop:

```python
s = input("Ange s: ")
```

4 p

Uppgift 6

(a) Ange de två bästa anledningar du kan komma på för att vi skulle låta tabeller i vår databas vara i 3NF istället för i BCNF. *1.5 p*

(b) Visa att en relation med två attribut alltid är i BCNF. *1.5 p*

Alternativt format för tabelldefinitioner i uppgift 1a

Ni kan välja att skriva vanlig SQL-kod för att definiera tabellerna, men det är även OK att använda följande format, om ni vill ha lite mindre att skriva (exemplet visar de tabeller för studenter som vi haft under flera föreläsningar).

I texten nedan betyder `PK` primary key, och `FK` betyder foreign key – var noga med att ange vad era foreign keys refererar till (se pilarna nedan):

```
tabell: students
  s_id     INT
  s_name   TEXT
  gpa      REAL
  PK:      s_id

tabell: colleges
  c_name      TEXT
  state       TEXT
  enrollment  INT
  PK:         c_name

tabell: majors
  major        TEXT
  credits      INT
  description  TEXT
  PK:          major

tabell: programs
  c_name      TEXT
  major       TEXT
  enrollment  INT
  PK:         c_name, major
  FK:         c_name -> colleges(c_name)
  FK:         major -> majors(major)

tabell: applications
  s_id      INT
  c_name    TEXT
  major     TEXT
  decision  BOOLEAN
  PK:       s_id, c_name, major
  FK:       s_id -> students(s_id)
  FK:       c_name, major -> programs(c_name, major)
```