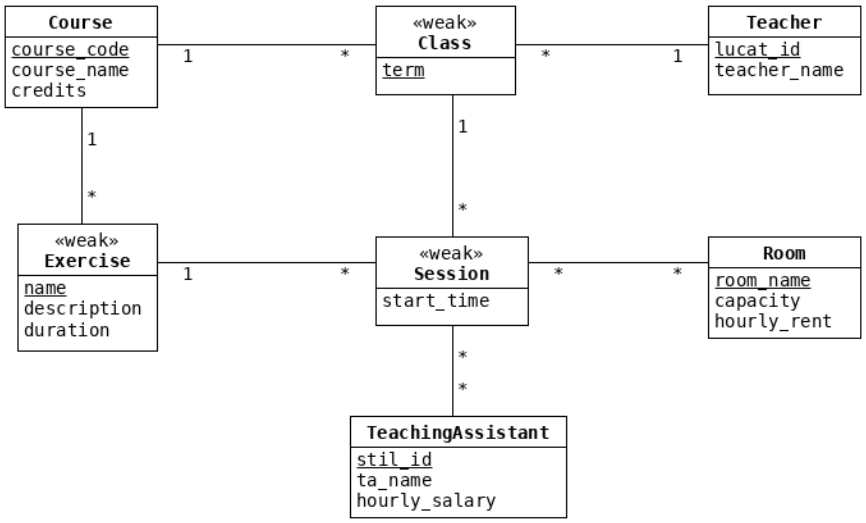
Lösningar till tentamen i EDAF75

19 mars 2019

Lösning 1

(a) Förslag till ER-modell (det finns alternativa lösningar):



(b) Det finns några 'svaga' entity sets i modellen ovan, och vi kan välja att införa 'invented keys' för några av dem, eller kanske rent av alla (`classes`, `exercises`, `sessions`). En invented key är egentligen en implementationsdetalj, så man kan argumentera för att den inte skall synas i ER-modellen, men eftersom den ibland returneras från REST-apier, så är det inte orimligt att skriva ut den.

courses(course_code, course_name, credits)
classes(class_id, ***course_code***, term, ***teacher_lucat_id***)
teachers(lucat_id, teacher_name)
exercises(exercise_id, ***course_code***, exercise_name, description, duration)
sessions(session_id, ***class_id***, ***exercise_id***, start_time)
rooms(room_name, capacity, hourly_rent)
booked_rooms(***session_id***, ***room_name***)
teaching_assistants(stil_id, ta_name, hourly_salary)
ta_hours(***session_id***, ***stil_id***)

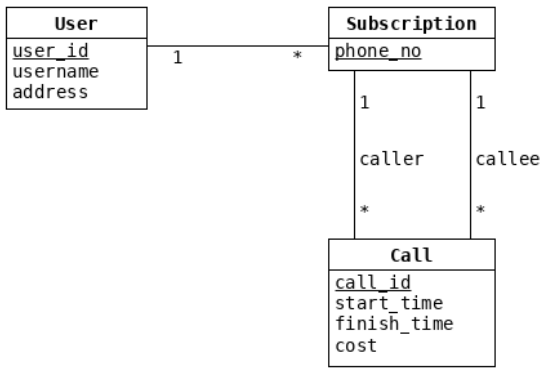
(c) Med de relationer vi beskrev ovan får vi något i stil med:

```
SELECT   start_time, room_name
FROM     booked_rooms
JOIN     sessions
USING    (session_id)
JOIN     classes
USING    (class_id)
JOIN     courses
```

```
USING     (course_code)
WHERE     course_name = "Databasteknik"
          AND term = "2019-vt1"
ORDER BY  start_time, room_name;
```

Lösning 2

(a)



(b) Vi kan välja att representera `call_id` på olika sätt – idealiskt vore att använda en `uuid`, men även `INTEGER` eller `TEXT` fungerar bra här (och `uuid` finns inte i SQLite3). Även kostnaden för samtalen kan representeras på olika sätt – om vi räknar i ören/cent så kan vi använda ett heltal, man kan även tänka sig att använda `DECIMAL(6,2)`, eller `REAL` (även om man ibland försöker att undvika reella tal när man räknar med pengar).

```
DROP TABLE IF EXISTS calls;
CREATE TABLE calls (
  call_id      TEXT,
  caller_no    TEXT NOT NULL,
  callee_no    TEXT NOT NULL,
  start_time   DATETIME NOT NULL,
  finish_time  DATETIME NOT NULL,
  cost         INT,
  PRIMARY KEY  (call_id),
  FOREIGN KEY  (caller_no) REFERENCES subscriptions(phone_no),
  FOREIGN KEY  (callee_no) REFERENCES subscriptions(phone_no)
);
```

(c)

```
SELECT    caller_no, cost
FROM      calls
ORDER BY  start_time DESC
LIMIT     10;
```

(d)

```
SELECT  username, address
FROM    subscriptions
JOIN    users
USING   (user_id)
WHERE   phone_no = "0707-123456";
```

(e)

```
SELECT    username, address
FROM      subscriptions
JOIN      users
USING     (user_id)
GROUP BY  user_id
HAVING    count(user_id) > 1
ORDER BY  username;
```

(f)

```
SELECT    username, address, sum(cost) AS total_cost
FROM      calls
JOIN      subscriptions
ON        calls.caller_no = subscriptions.phone_no
JOIN      users
USING     (user_id)
GROUP BY  user_id
HAVING    total_cost > 1000;
```

(g) Vi kan lösa uppgiften antingen med hjälp av subqueries:

```
SELECT phone_no
FROM subscriptions
WHERE
  phone_no NOT IN (SELECT caller_no FROM calls) AND
  phone_no NOT IN (SELECT callee_no FROM calls);
```

eller med outer joins:

```
WITH
  no_calls_to(phone_no) AS (
    SELECT     phone_no
    FROM       subscriptions
    LEFT JOIN  calls
    ON         subscriptions.phone_no = calls.callee_no
    WHERE      calls.callee_no IS NULL
  ),
  no_calls_from(phone_no) AS (
    SELECT     phone_no
    FROM       subscriptions
    LEFT JOIN  calls
    ON         subscriptions.phone_no = calls.caller_no
    WHERE      calls.caller_no IS NULL
  ),
  no_activity(phone_no) AS (
    SELECT     phone_no
    FROM       no_calls_to
    INTERSECT
    SELECT     phone_no
    FROM       no_calls_from
  )
SELECT     phone_no
FROM       no_activity;
```

Lösning 3

(a) Vi har alltså relationen $R(A, B, C, D, E, F)$, och de funktionella beroendena

FD_1: $AB \longrightarrow C$

FD_2: $A \longrightarrow D$

FD_3: $D \longrightarrow AE$

FD_4: $E \longrightarrow F$

Vi ser att B måste ingå i alla nycklar, eftersom den inte förekommer i högerledet för något beroende. Så vi börjar med att testa $\{B\}^+$, men får då bara $\{B\}$, så B är ingen nyckel.

Vi testar därefter alla möjliga tvåattributsnycklar som innehåller B, och får då:

$\{AB\}^+ = \{ABCDEF\}$

$\{BC\}^+ = \{BC\}$

$\{BD\}^+ = \{ABCDEF\}$

$\{BE\}^+ = \{BEF\}$

$\{BF\}^+ = \{BF\}$

Detta visar att $\{AB\}$ och $\{BD\}$ är nycklar. Möjliga tre-attributsnycklar är $\{BCE\}$, $\{BCF\}$ och $\{BEF\}$, men:

$\{BCE\}^+ = \{BCEF\}$

$\{BCF\}^+ = \{BCF\}$

$\{BEF\}^+ = \{BCEF\}$

och den enda möjliga fyr-attribut nyckeln är $\{BCEF\}$, men:

$\{BCEF\}^+ = \{BCEF\}$

Detta visar att $\{AB\}$ och $\{BD\}$ är de enda nycklarna.

(b) Vi är inte i BCNF eftersom FD_2, FD_3 och FD_4 alla har vänsterled som inte är supernycklar.

(c) Vi är inte i 3NF eftersom högerleden i FD_3 och FD_4 inte är delar av nycklar.

(d) Vi kan börja med att utveckla samtliga funktionella beroenden (vi beräknar deras transitiva höljen), och får då:

FD_1: $AB \longrightarrow CDEF$

FD_2: $A \longrightarrow DEF$

FD_3: $D \longrightarrow AEF$

FD_4: $E \longrightarrow F$

Sammanfattningsvis har vi:

Relation:	$R(A, B, C, D, E, F)$
Beroenden:	$AB \longrightarrow CDEF$, $A \longrightarrow DEF$, $D \longrightarrow AEF$, $E \longrightarrow F$
Nycklar:	$\{AB\}$, $\{BD\}$

Vi kan välja att bryta ner R med hjälp av $A \longrightarrow DEF$, som ju bryter mot BCNF, och får då relationerna $R_1(A, D, E, F)$ och $R_2(A, B, C)$. För R_1 får vi:

Relation:	$R_1(A, D, E, F)$
Beroenden:	$A \longrightarrow DEF$, $D \longrightarrow AEF$, $E \longrightarrow F$
Nycklar:	$\{A\}$, $\{D\}$

Här bryter $E \longrightarrow F$ mot BCNF, så vi bryter upp R_1 ett steg till, till $R_{1a}(E, F)$ och $R_{1b}(A, D, E)$. R_{1a} har bara två attribut, och är därför i BCNF, för R_{1b} får vi:

Relation:	$R_{1b}(A, D, E)$
Beroenden:	$A \longrightarrow DE$, $D \longrightarrow AE$
Nycklar:	$\{A\}$, $\{D\}$

Denna är i BCNF, eftersom båda våra funktionella beroenden har vänsterled som är nycklar.

För R_2 har vi:

Relation:	$R_2(A, B, C)$
Beroenden:	$AB \longrightarrow C$
Nycklar:	$\{AB\}$

som är i BCNF, eftersom dess enda funktionella beroende har ett vänsterled som är en nyckel.

Så, vi delar ner $R(A, B, C, D, E, F)$ till $R_{1a}(E, F)$, $R_{1b}(A, D, E)$ och $R_2(A, B, C)$, och är därmed i BCNF.

Lösning 4

ACID är en förkortning för Atomicity-Consistency-Isolation-Durability.

(a) De två huvudproblem som transaktioner hjälper oss med är:

- Hantering av fel – detta kopplar till:
  - *Atomicity*, dvs våra transaktioner är atomära – om något går fel under en transaktion så vill vi att vi antingen kan återställa databasen till det tillstånd den hade innan transaktionen (*rollback*), eller att samtliga ändringar genomförs (*commit*).
  - *Durability* – vi vill att data som commit-ats i databasen skall kunna återställas efter en krasch.
- Vi vill kunna låta flera användare använda databasen 'samtidigt' utan att förstöra för varandra – detta kopplar till begreppet *Isolation*.

(b) I avtagande säkerhetsordning:

- `SERIALIZABLE`: Detta är den högsta nivån av säkerhet – vi är garanterade att två transaktioner beter sig som om de körts efter varandra.
- `REPEATABLE READ`: Garanterar att vi i en transaktion kommer att få samma resulat varje gång vi läser en given rad i en tabell (vi kan dock få nya rader).
- `READ COMMITTED`: Garanterar att vi i vår transaktion aldrig kan läsa data som inte har 'commit'-ats (men vi kan få värden som ändrats av andra transaktioner efter att vi påbörjade vår transaktion).
- `READ UNCOMMITTED`: Detta är den lägsta nivån av säkerhet – vi kan läsa data som ändrats i en parallell transaktion som inte ens commit-ats.

Lösning 5

För en relation med två attribut, $R(A, B)$, kan våra funktionella beroenden bara se ut på endera av följande sätt:

1. vi har inga beroenden alls,
2. vi har ett beroende, $A \longrightarrow B$ eller $B \longrightarrow A$, eller
3. vi två har två beroenden: $A \longrightarrow B$ och $B \longrightarrow A$.

Vi behöver bara visa att vi i vart och ett av dessa fall är i BCNF, dvs att *samtliga beroenden i vart och ett av dessa fall har vänsterled som är en supernyckel.*

Så vi tar våra fall i tur och ordning:

1. Inga beroenden alls: vår enda nyckel i detta fall är $\{AB\}$, men vi har inga beroenden, så vi är i BCNF.
2. Ett beroende, $A \longrightarrow B$ eller $B \longrightarrow A$: Av symmetriskäl räcker det att vi behandlar det ena fallet, så antag att vi har $A \longrightarrow B$ som enda beroende. Vår nyckel är i så fall $\{A\}$, och vi är i BCNF eftersom $A \longrightarrow B$ har ett vänsterled som är en (super)nyckel.
3. Två beroenden: $A \longrightarrow B$ och $B \longrightarrow A$. Vi får här två nycklar, $\{A\}$ och $\{B\}$, och det gör att båda våra beroenden har vänsterled som är (super)nycklar.

Detta visar att alla relationer med bara två attribut är i BCNF (och därmed även i 3NF).

Lösning 6

(a) Man kan lösa det på flera sätt, exempelvis med en inner join, eller med en subquery – med en inner join kan vi skriva:

```
SELECT DISTINCT a
FROM   r1
JOIN   r2
USING  (a);
```

(b) Även denna uppgift kan lösas på flera sätt, ett sätt är att skapa en tabell med samtliga värden som finns i någon av tabellerna, och en tabell med de värden som finns i båda tabellerna, och bara ta differensen:

```
WITH
  in_any(a) AS (
    SELECT a
    FROM   r1
    UNION
    SELECT a
    FROM   r2
  ),
  in_both(a) AS (
    SELECT a
    FROM   r1
    INTERSECT
    SELECT a
    FROM   r2
  )
SELECT a
FROM   in_any
EXCEPT
SELECT a
FROM   in_both;
```