Databases Exam

TDA357 (Chalmers), DIT621 (University of Gothenburg)

2019-01-16 14:00-18:00

Department of Computer Science and Engineering

Examiner: Jonas Duregård tel. 031 772 1028.

Results: Will be published within three weeks from exam date

Maximum points: 60

Grade limits Chalmers: 24 for 3, 36 for 4, 48 for 5.
Grade limits GU: 24 for G, 42 for VG.

Allowed material: One double sided A4 sheet with hand-written notes. If you bring a sheet, it must be handed in with your answers to the exam questions, add "+1" in the box for number of pages on the exam cover.

One English language dictionary is also allowed. Specific instructions: You can answer in English or Swedish. Begin the answer to each question (numbers 1 to 6) on a new page. The a,b,c,... parts with the same number can be on the same page.

Write the question number on every page. Write clearly, unreadable answers give no points! Fewer points are given for solutions that are clearly unnecessarily complicated. Indicate clearly if you make any assumptions that are not given in the question. In particular: in SQL questions, use standard SQL or PostgreSQL. If you use any other variant (such as Oracle or MySQL), say this; but full points are not guaranteed since this may change the nature of the question.

Question 1: ER-design (10 points, 5+5)

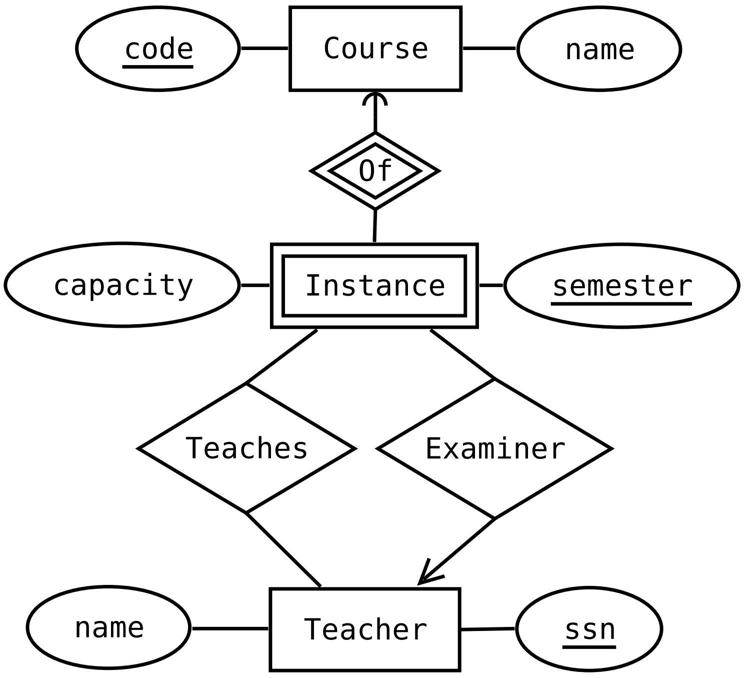
a) Your task is to make an ER-diagram for the database of a gym company, managing their facilities and their customer records.

The database should contain a set of current and past customers. Each customer has a name and an email-address. Not every customer is a current member, and the database should keep track of which customers are currently members, and when their membership expires.

Each gym facility has a city, an address and a name. Two facilities can have the same name, but only if they are in different cities. Gym facilities can be established in any city across the world (but you can assume cities have unique names).

The database should also keep a record of times when each customer has accessed any gym facility. This may include multiple accesses from the same customer to the same facility at different times.

b) This ER-diagram describes courses (divided into instances) that can have multiple teachers and at most one examiner.



Translate the diagram into relations, but modify the resulting schema to ensure the following additional constraint: The examiner of a course must be one of the teachers of that course.

Question 2: Functional Dependencies, Normal Forms (9 p, 3+3+3)

Consider this (symbolic) relational schema:

`R(a,b,c,d,e)`

The following functional dependencies apply to the relation:

$a \rightarrow b$
$b, c \rightarrow d$
$d, e \rightarrow a$

a) List 3 different minimal keys of the relation. Your solution should be three sets of attributes (the keys).

b) Calculate the following transitive closures. Your solution should be three sets of attributes (the transitive closures), each in alphabetical order.

$\{b, c\}^+$
$\{a, c\}^+$
$\{b, d, e\}^+$

c) Assume all the keys you identified have appropriate unique constraints for the keys you identified, show a table (the contents of R) that does not respect the functional dependency $b, c \rightarrow d$. Use integer values for all the columns.
In other words: Your solution should be a five column table (for columns a,b,c,d and e) with at least two rows, that does not satisfy the functional dependency, but does have unique values for the three keys you found.

Question 3: SQL Queries (9 p, 3+3+3)

Below is the schema for a database of users. Users can follow other users (and choose if they should receive notifications of that users activity or not).

```
Users(username, email)

Follows(follower, follows, notifications)
  follower -> Users.username
  follows -> Users.username
```

a) Write an SQL query that lists the number of followers each user has. The result should have three columns: username, email and total_followers. For full score, every user must be in the result exactly once.

Hint: The aggregate function COUNT(x) will not count rows where column x is null.

b) Write an SQL query for finding all pairs of usernames A and B, where A follows B, but B does not follow A.

c) Write an SQL query for finding the usernames of all followers of the user with username 'jonas', and all followers of those followers (no duplicates).

Question 4: Relational Algebra (9, 3+3+3)

Consider this relational schema, containing football players and goals they have scored.

```
Players(idnr, name, birthyear, birthmonth, birthday)

Goals(player, game, goalnr, minute)
  player -> Players.idnr
```

a) Write a relational algebra expression for the names and id-numbers (no duplicates) of all players that have scored at least one goal in overtime (minute 90 or later).

b) Write an expression for computing the total number of goals scored by players born in each month of the year. The result should have two attributes: birthMonth and total_goals. It is OK to only list months with at least one player that has made at least one goal.

c) Write an expression for computing the idnr of all players that have scored two goals in the same minute of any game. Duplicates are allowed (but not required).

Question 5: Views, constraints and triggers (13 p)

Database integrity can be improved by several techniques:

- Views: virtual tables that show useful information that would create redundancy if stored in the actual tables
- SQL Constraints: conditions on attribute values and tuples
- Triggers (and assertions): automated checks and actions performed on entire tables

As a general rule, these methods should be applied in the above order: if a view can do the job, constraints are not needed, and if constraints can do the job, triggers are not needed.

The task in this question is to implement a database for a cell phone company.

You are allowed to use any SQL features we have covered in the course. While the description below gives requirements for what should be in the database, you are allowed to divide it across as many tables and views as you need to. Points will be deducted if your solution uses a trigger where a constraint or view would suffice, or if your solution is drastically over-complicated.

For triggers, it is enough to specify which actions and tables it applies to, and PL/(pg)SQL pseudo-code of the function it executes.

Your task: The database contains Customers and Subscriptions. Each customer can have any number of subscriptions. Below are values that should be in the database:

- A Customer has a unique id number and a name, a monthly billing and a Boolean indicating if it is a private customer (true meaning it is a private customer).
- A Subscription belongs to a customer and has a unique phone number, a plan, a monthly fee and a balance.

Implement the following additional constraints in your design. Put letters in the margin of your code indicating where each constraint is implemented (possibly the same letter in several places):

a) Each plan must be one of 'prepaid', 'flatrate' or 'corporate'.
b) The balance value must be 0 if the plan is not 'prepaid'.
c) Private customers cannot have 'corporate' plans (but non-private customers may still have any plans including but not limited to 'corporate').
d) The monthly billing of a customer must be the sum of the fees of all that customers numbers, and all fees must be non-negative.
e) If a customer is deleted, its connected subscriptions should be deleted automatically.
f) If the last subscription belonging to a customer is deleted, the customer should be deleted automatically.

Question 6: Semi-structured data and other topics (10 p, 3+4+3)

Read all parts of the question before you start, the answers are not independent.

In this task you will store restaurant menus in a semi-structured data format.

Menus have dishes with prices, divided into categories that may further be divided into subcategories.

Here is an example menu with 3 categories (Starters, Salads and Burgers) and Burgers contains an additional category (Vegetarian burgers).

```
Starters
  Calamari          $8.50
Salads
  Caesar            $8.50
  Chicken           $9.25
Burgers
  Standard          $9
  Bacon             $10
  Vegetarian burgers
    Haloumi         $12
    Mushroom        $10
```

a) Encode the menu above in a JSON document (OR in XML if you prefer). Note that the order of items in the menu is important. Prices are all numbers (the $ does not need to be included).

Hint: Perhaps the root of the JSON document should not be an object?

b) Write a JSON Schema for menus (OR a XML DTD if you use XML). It is OK if your schema allows extra data (like descriptions of categories or ingredients of dishes), but it should not allow things like a category having a price or a dish containing other dishes. You should allow arbitrarily deep nesting of categories (i.e. categories in categories in categories...).

c) Write a JSON Path expression (OR an XML Path expression if you use XML) for finding the prices of all burgers (in a menu valid in your schema, and structured similarly to the example above with burgers as a top level category). In this example the result would be 9, 10, 12, 10 (order is not important).