Databases Exam

TDA357 (Chalmers), DIT621 (University of Gothenburg)

2019-08-29 14:00-18:00

Department of Computer Science and Engineering

Examiner: Aarne Ranta

Responsible teacher: 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. Also, a standard reference is handed out in a separate document.

One English language dictionary is also allowed. 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 sometimes 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.

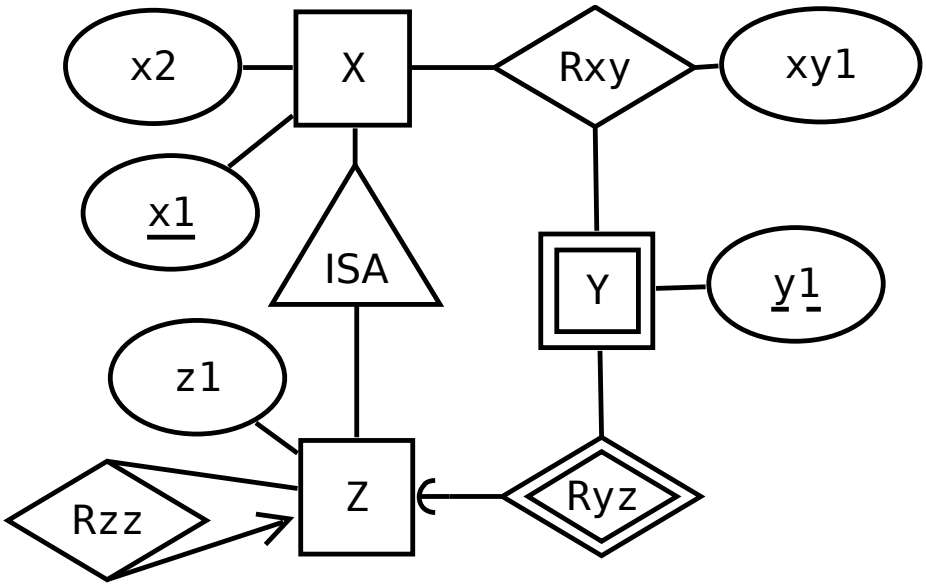
Question 1: ER-design (10 points, 5+5)

a) Make an ER-diagram for a web application where users can register, join interest groups, and post messages to these groups.

Users have unique user names, and groups have unique names. Users can be members of any number of groups. The time each user joined a group should be recorded. Each group has an owner, which is a user. Each post is posted in a group by a user. Posts contain text. Posts are identified by their timestamp and the username of the user posting it. There is also a special kind of post called a group link, these contain a link to a group in addition to the normal parts of a post.

You do <u>not</u> need to translate your diagram into a schema!

b) Translate this (symbolic) ER-diagram into a relational schema, including keys and constraints.



(yes - I was too lazy to come up with another domain for this...)

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

Consider a relation R(A, B, C, D, E) with the following Functional Dependencies

A -> B
B -> A
B, D -> E
B, C -> E
E -> A

a) Calculate the following transitive closures:
$\{E\}^+$
$\{B, C\}^+$
$\{C, D\}^+$

b) List 3 different minimal keys of this relation (each key is a set of attributes, write each set in alphabetical order on its own line)

c) Decompose the relation into BCNF. You only have to provide the final schema, not all the steps taken to compute it. If you do include all the steps, be sure to clearly indicate which relations are actually in the final schema (as opposed to intermediate relations).

Question 3: SQL Queries (10 p, 5+5)

Below is the schema for a database of dots in a plane that can have (directed) connections between them. Each dot has an x- and a y-coordinate, and an identifying number. Each connection has a numeric weight value.

```
Dots(x, y, idnr)
  (x, y) UNIQUE

Connections(from, to, weight)
  from -> Dots.idnr
  to -> Dots.idnr
  from ≠ to
```

Example contents:

x	y	idnr
0	0	0
3	8	1
-5	7	2

from	to	weight
0	1	4
1	0	7
2	1	9

In this example, dots 0 and 2 are both connected to 1 (weights 4 and 9 respectively) and 1 is connected back to 0 with a weight of 7.

a) Write an SQL query that lists the from/to coordinates of each connection along with the weights of the connections. The result should have five columns: x_from, y_from, x_to, y_to, and weight. Below is the expected result for the example above:

x_from	y_from	x_to	y_to	weight
0	0	3	8	4
3	8	0	0	7
-5	7	3	8	9

b) Write an SQL query that lists the Radix (total number of incoming and outgoing connections) and the total weight of all incoming and outgoing connections for all dots. Below is the expected result for the example above:

idnr	radix	total
0	2	11
1	3	20
2	1	9

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

Consider this relational schema for part of the database of an online role-playing game.

```
Players(name, level, money)

Items(id, itemname, value)

Equippable(id, equipslot)
  id -> Items.id

PlayerInventory(item, player)
  item -> Items.id
  player -> Players.name

Equipped(item, player, equipslot)
  (item, equipslot) -> Equippable(id, equipslot)
  player -> Players.name
```

Players contains the name, levels and in-game money of all players.

Items describe all the items players can find in the game, each has an ID-number, a name and an in-game value.

Equippable further describes items that can be worn as equipment by players, each such item has an equipslot attribute e.g. 'armor', 'weapon' or such.

PlayerInventory contains the items all players are carrying (not including equipped items).

Equipped contains the equipment all players are wearing, note that a player can equip at most one item in each equipslot.

a) Write a relational algebra expression for finding the names of all items with a value of at least 1000 and equipped by at least one player with a level above 75. The result should have a single column containing item names.

b) Write a relational algebra expression for finding for each player, the total combined value of all items in its equipment and inventory. The result should have two columns, one for player names and the other for total value. You may exclude players that have no items. You may assume bag semantics for duplicate rows.

c) Write a relational algebra expression for finding all equippable items in player 'Jonas' inventory, whose equipslot is currently not occupied (i.e. every items Jonas is carrying that Jonas can equip without removing any current equipment). The result should have two columns, for item id and one for equipslot.

Hints: Do not use relational algebra expressions in conditions!
Hint: Don't forget about set operations, they are often very useful.

Question 5: Views, constraints and triggers (12 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 or constraint can adequately do the job, do not use a trigger.

The task in this question is to implement a small database. You may 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.

The database contains warehouses and shipments to/from those warehouses. The database should have this public interface (may include tables and views):

```
Warehouses(address)
Inventory(warehouse, item, quantity)
Shipment(warehouse, item, quantity_change, time)
```

Here **item** specifies what is stored (as an item ID-number), **quantity_change** is a number that is positive for incoming and negative for outgoing shipments, e.g. -3 means three items were sent from the warehouse, 11 means eleven items were received. 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) Any warehose/item pair can occur at most once in Inventory.
b) The quantities in Inventory should be consistent with Shipment, meaning no items can magically appear or disappear without a registered shipment.
c) If an update query is executed on Inventory to change a quantity, a row should be added to shipment that reflects the change. Use CURRENT_TIMESTAMP as time.
d) All warehouse values must refer to actual warehouses (present in Warehouses)
e) If two rows in Shipment have the same warehouse and item, they must have different times.
f) A shipment must either send or receive at least one item.

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

a) Give an example of a SQL Injection vulnerability in a Java (JDBC) application. You may use the student portal domain from the lab without explaining it, or any other with a short explanation. The example should include:

1. How is the vulnerability created? (Include pseudo-code)
2. How can an attacker exploit the vulnerability and what are the negative consequences?
3. How can the vulnerability be avoided?

b) Study this JSON Schema for a file system containing files and folders:

```
{
  "oneOf": [{"$ref": "#/definitions/file"},
            {"$ref": "#/definitions/folder"
  }],
  "definitions": {
    "file": {
      "type": "object",
      "properties": {
        "filename": {"type": "string"},
        "size": {"type": "integer"}
      },
      "required": ["filename", "size"]
    },
    "folder": {
      "type": "object",
      "properties": {
        "foldername": {"type": "string"},
        "contents": {
          "type": "array",
          "items": {"$ref": "#"}
        }
      },
      "required": ["foldername", "contents"]
    }
  }
}
```

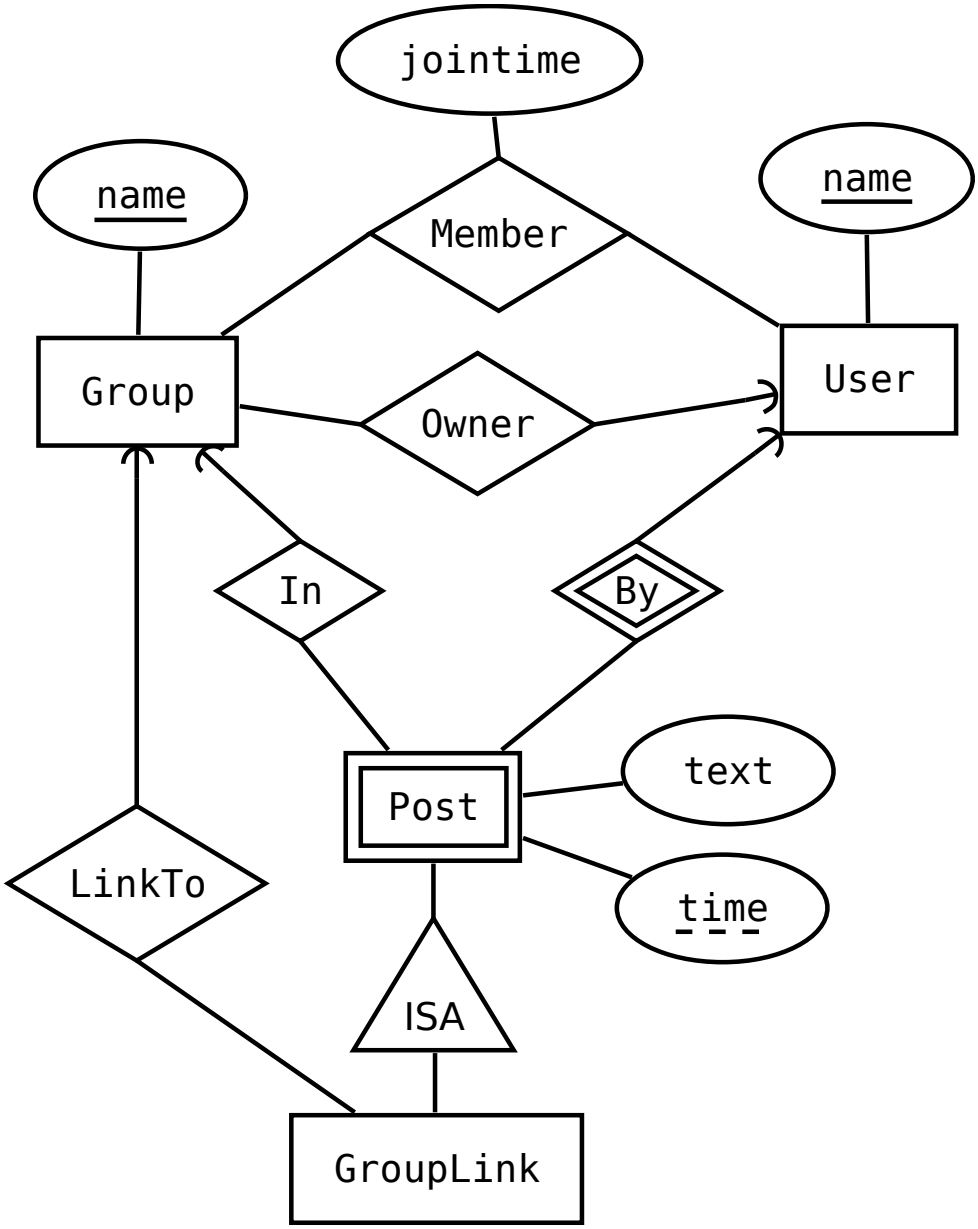
Write a JSON document encoding a file system that is valid w.r.t. the schema above, and contains the following:

1. A folder called “root”
2. An empty folder in root called “temp”
3. A folder in “root” called “img”
4. A file in “img” called “pic.jpg” of size 1234
5. A file in “root” called “notes” of size 100.

```
root
  temp
  img
    pic.jpg (1234)
  notes (100)
```

c) Using the same JSON Schema, write a JSON Path expression that finds the sizes of all files located (directly) in all folders named “temp” (throughout the queried JSON document). The result should be an array of sizes (integers).

1 a



1 b) valda namn är i stort sett oväsentliga

X(x1,x2)
Z(x1,z1)
x1 -> X.x1
Y(y1,z)
z -> Z.x1
Rzz(za,zb)
za -> Z.x1
zb -> Z.x1
Rxy(x,y,yz,xy1)
x -> X.x1
(y,yz) -> Y.(y1,z)

2 a)
$\{E\}^+ = \{E,A,B\}$
$\{B, C\}^+ = \{B,C,E,A\}$
$\{C, D\}^+ = \{C,D\}$

b)
A,C,D
B,C,D
C,D,E

c) One of several possible solutions:
R1(B, A)
R2(E, B)
R3(C, D, E)

3 Untested solutions:
a)

```
SELECT F.x AS x_from, F.y AS y_from, T.x AS x_to, T.y AS y_to, weight
FROM Dots F, Connections, Dots T
WHERE from = F.idnr AND to = T.idnr
```

b) The first opeand of the UNION is only to include nodes with RADIX=0, you can still get full points even if you did not include it (providing the rest of your solution is spotless)

```
SELECT idnr, COUNT(weight) AS radix, SUM(weight) AS total
FROM  (SELECT idnr, NULL AS weight FROM Connections)
      UNION
      (SELECT from, weight FROM Connections) UNION
      UNION
      (SELECT to, weight FROM Connections)) AS Combined
GROUP BY idnr;
```

4)

a)

$\pi_{itemname}$ (
  $\sigma_{value >= 1000 \text{ AND } level>75 \text{ AND } id=item \text{ AND } player=name}$ (
    Items X Equipped X Players
))

b)

$\gamma_{name, SUM(value)->total}$(
  (PlayerInventory U $\pi_{item, player}$(Equipped))
  $X_{item=id}$
  Items
)

c) *First I find unused slots using subtraction (-), same as SQLs EXCEPT/MINUS operator*

let Unused = $\sigma_{player='jonas'}$(
  $\pi_{player, equipslot}$(PlayerInventory $X_{id=item}$ Equippable)
  -
  $\pi_{player, equipslot}$(Equipped)
))

Then I join the other attributes back in and project the correct ones, here using natural join:

R = $\pi_{id, equipslot}$(PlayerInventory $X_{id=item}$ Equippable ⋈ Unused))

5) Untested code

```
CREATE TABLE Warehouses (address TEXT PRIMARY KEY);

CREATE TABLE Shipment (
  warehouse TEXT REFERENCES Warehouses,  -- d
  item INT,
  quantity_change INT,
  time TIMESTAMP,
  PRIMARY KEY (warehouse,item,time),  -- e
  CHECK quantity_change != 0  -- f
);

CREATE VIEW Inventory AS  -- b
SELECT warehouse, item, SUM(quantity_change) AS quantity
FROM Shipment
GROUP BY warehosue, item;  -- a

-- For c, create a trigger INSTEAD OF UPDATE ON Inventory, that executes this statement:
INSERT INTO Shipment
 VALUES (OLD.warehouse,
         OLD.item,
         NEW.quantity-OLD.quantity
         CURRENT_TIMESTAMP)
```

6)

a)

Any example including user input, concatenating that user input to a query, and an example of malicious input and its consequences. Avoided using prepared statements.

b) There was a small mistake in the schema, saying "file" and "folder" were required instead of filename and foldername.

```
{
 "foldername":"root",
 "contents": [
    {
     "foldername":"temp",
     "contents": [{"filename":"me", "size":100} ]
    },
    {
     "foldername":"img",
     "contents": [
        {"filename":"pic.jpg", "size":1234}
     ]
    },
    {"filename":"notes", "size":100}
 ]
}
```

c)

```
$..[?(@.foldername=="temp")].contents.[*].size
```