Databases Exam

TDA357 (Chalmers), DIT622 and DIT621 (University of Gothenburg)

2024-01-10 14:00-18:00

Department of Computer Science and Engineering

Examiner: Jonas Duregård, Will visit the exam hall at least twice, Phone: 031 772 1028

Allowed aids: One double sided A4 page of hand-written notes, the notes should be handed in along with your solution. Write your anonymous code on the notes if you wish, but do not write your name on it.

Results will be published within three weeks of the exam date. Maximum points: 60

Grade limits: 27 for 3, 38 for 4, 49 for 5. Grade limits for DIT621: 27 for G, 45 for VG.

Question 1: SQL queries (10 p)

Consider this database of a manufacturing company:

Stores(<u>city</u>, address, manager)

Retailers(<u>id</u>, city, address)

Tags(<u>retailer</u>, <u>tag</u>)
retailer → Retailer.id

Stores are locations owned by the company (at most one store in each city), retailers are independent locations that sell the company's products. Retailers can be assigned tags (a tag is just any piece of text).

a) (3p) Write an SQL query that finds all (city,citytag)-pairs such that citytag is either a tag of at least one retailer in the city, or citytag is the word 'store' for a city with a store in it (so basically treat the presence of a store as an extra tag).

b) (3p) Write an SQL query that finds the names of all cities with at least three locations (including both stores and retailers).

Hint: Remember that the SQL set operations have variants for bag (multiset) semantics.

c) (4 p) Write an SQL query that gives all the information of all retailers that have the 'outlet' tag, along with the manager of the store in the city of the retailer - or null for retailers in cities where there is no store. The result should have four columns, three from Retailer and a fourth for the manager (or null for no manager).

Solution 1:

```sql
-- a)
SELECT city, tag FROM Retailers JOIN TAGS ON retailer=id
UNION
SELECT city, 'store' FROM Stores;

-- b)
SELECT city
FROM (SELECT city FROM Stores UNION ALL SELECT city FROM Retailers) AS
combined
GROUP BY city
HAVING COUNT(*) >= 3;

-- c)
SELECT id, city, Retailers.address, manager
FROM Retailers
     JOIN Tags ON retailer=id
     LEFT JOIN Stores USING (city)
WHERE tag = 'outlet';
```

Question 2: SQL Data Definition Language (10 p)

a) Write SQL code for creating a database with the following public interface (tables and/or views). You may have additional tables that are not part of the public interface.

Stations(stationId, area, elevation)
SMeasure(time, stationId, temperature)
AMeasure(sequence, stationId, area, temperature)

Explanation of tables/views and columns:

- ***Stations*** contains the set of all weather stations. The stationID column is a unique identifier (text), and each station is in a specified area and at a specified elevation. If a station is deleted, all measurements performed by the station should also be deleted.
- ***SMeasure*** (for StationMeasure) contains a row for each measurement from each weather station. The time column is an integer timestamp used to distinguish different measurements from the same station (higher number is newer). The temperature column contains the measured temperature in degrees Celsius.
- ***AMeasure*** (for AreaMeasure) also contains a row for each measurement from each weather station. Columns stationID and area are the ID and area of the measuring weather station, temperature is the measured temperature. The column seq (sequence) should be consecutive numbers 0,1,2... for each stationID, such that higher numbers are given to newer measurements (see example).

Here is an example of valid content of (the public interface of) the database, for five measurements performed by three stations:

stationId	area	elevation
S1	Sweden	200
S2	Sweden	300
N1	Norway	400

time	stationId	temperature
20	S1	5
50	S1	-1
40	S2	1
30	N1	1
80	N1	8

seq	stationID	area	temperature
0	S1	Sweden	5
1	S1	Sweden	-1
0	S2	Sweden	1
0	N1	Norway	1
1	N1	Norway	8

b) Insertions: Show all the SQL inserts required to get the data above.

Solution 2

```sql
-- a)
CREATE TABLE Stations(
    stationID TEXT PRIMARY KEY,
    area TEXT NOT NULL,
    elevation INT NOT NULL
);

-- Points for making this a view.
-- Defining the seq column is a bit tricky, here it's done using a
--  correlated query in the select, that counts the number of earlier
--  measures from the same station.
CREATE TABLE SMeasure(
  time INT,
  stationID TEXT REFERENCES Stations ON DELETE CASCADE,
  temperature INT  NOT NULL,
  PRIMARY KEY (time, stationID)
);

CREATE VIEW AMeasure AS
SELECT (SELECT COUNT(*)
        FROM SMeasure S2
        WHERE S2.stationID=S1.stationID AND S2.time < S1.time
        ) AS seq, stationID, area, temperature
FROM SMeasure AS S1 NATURAL JOIN Stations;


-- b)
INSERT INTO Stations VALUES ('S1','Sweden',200);
INSERT INTO Stations VALUES ('S2','Sweden',300);
INSERT INTO Stations VALUES ('N1','Norway',400);

INSERT INTO SMeasure VALUES (20,'S1',5);
INSERT INTO SMeasure VALUES (50,'S1',-1);
INSERT INTO SMeasure VALUES (40,'S2',1);
INSERT INTO SMeasure VALUES (30,'N1',1);
INSERT INTO SMeasure VALUES (80,'N1',20);
```

Question 3: Relational Algebra (10 p)

Consider this (possibly somewhat familiar?) database schema:

Programs(<u>name</u>, abbr)
UNIQUE abbr

Students(<u>studentId</u>, name, program)
program→Programs.name

ProgramMandatory(<u>program</u>, <u>course</u>)
program→Programs.name

PassedCourses(studentId, course)
studentId→Students.studentId

Students attend programs that have names and abbreviations. Courses can be mandatory for programs, and students can pass courses.

a) (3 p) Write a relational algebra expression equivalent to this SQL query:

```
SELECT abbr FROM Programs JOIN Students ON program=Programs.name
GROUP BY abbr
HAVING COUNT(*)>20;
```

Also describe in one sentence what the query/expression does (its purpose in terms of the domain, not in technical terms!).

b) (3 p) Write a relational algebra expression for finding all information of each student (id, name, program) along with the number of courses they have passed. You only need to include students who have passed at least one course. The result should have four columns.

c) (4 p) Write a relational algebra expression for finding the id of all students who have passed all their mandatory courses.

Hint: Set operations are nice.

Hint: A more complicated way of saying "the students who have passed all courses" is "the students who are not among the ones that have at least one missing course".

Reminder: You are <u>not</u> allowed to put relational algebra expressions in subscripts (e.g. in sigma operator conditions).

Solution 3:

a)

$\pi_{abbr}(\sigma_{num>20}\ (\gamma_{abbr,\ COUNT(*)\rightarrow num}(Programs \bowtie_{program=Programs.name} Students)$

Finds the abbreviation of all programs with more than 20 attending students.

b)

$\gamma_{studentId,\ name,\ program,\ COUNT(*)\rightarrow numPassed}(Students \bowtie PassedCourses)$

c)Using nested set subtraction (all students – (students with at least one missing course)):

$\pi_{studentId}(Students)$- $\pi_{studentId}(\pi_{studentId,course}(Students \bowtie ProgramMandatory)$-PassedCourses)

Question 4: ER-modelling (10p)

a) (7 p) You are designing a database for keeping inventory of computers used by employees at a company. Draw an ER-diagram that satisfies these requirements:

- Each individual computer has an identifying serial number and a computer model.
- Stationary computers additionally have a MAC address and a port number required for wired network connections.
- Each computer model has a year when their service life expires.
- Computers can be used by employees.
- Each computer can have a single current user, but some computers may be unused.
- The date when the current user started using a computer should be recorded in the database.
- All previous users of each computer should also be recorded in the database, along with the dates when they started and stopped using it.
- Employees are identified by their login names.
- Some employees do not use a computer.

Some examples of things that can be stored in the database (doesn't cover everything, but may be good to check that you have understood things correctly):

- "Employee JonasDuregard is using a Dell Latitude 5540 computer with serial number 1111-111-11, since 2023-12-15".
- "Dell Latitude 5540 computers have a service life until 2029".
- "The computer with serial number 1111-111-11 was previously used by Pelle between 2023-06-10 and 2023-12-10".

b) (3 p) Draw an ER-diagram that translates into exactly this relational schema (from Q1):

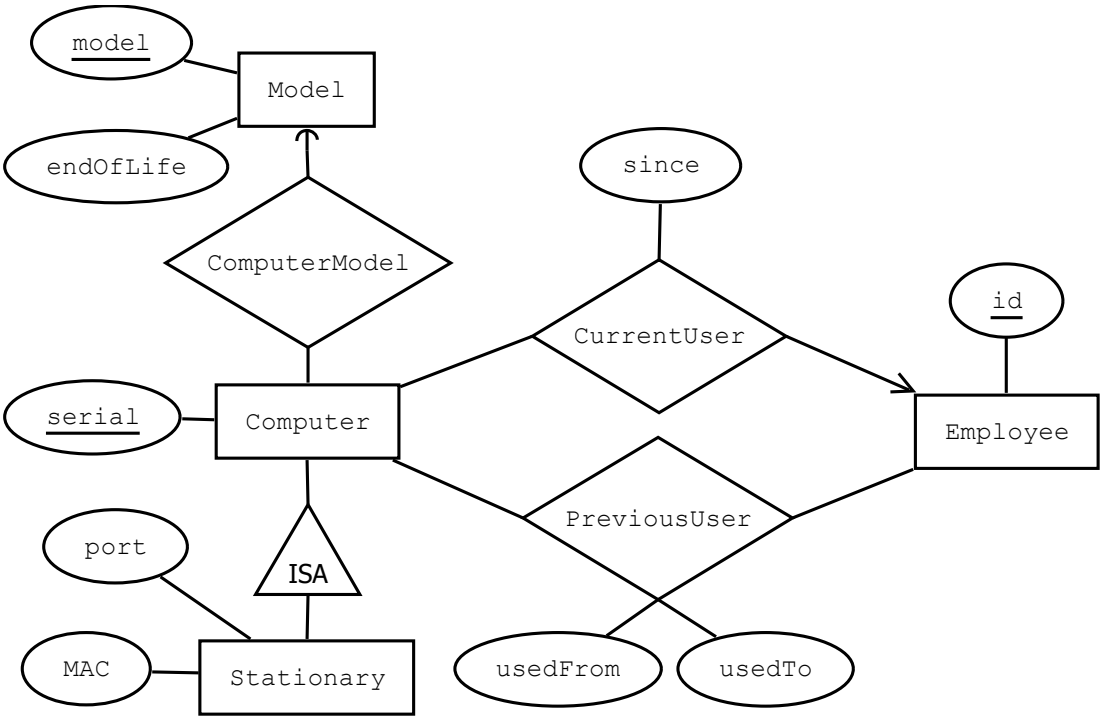
Store(<u>city</u>, address, manager)

Retailer(<u>id</u>, city, address)

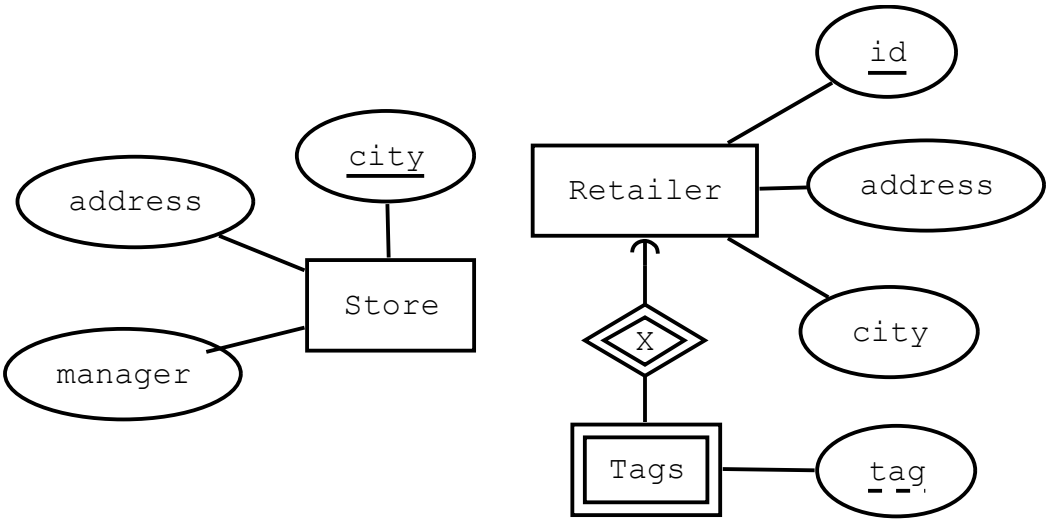
Tags(<u>retailer</u>, <u>tag</u>)
retailer → Retailer.id

Solution 4:

a)



b) The relationship X can have any name.



Question 5: Dependencies and normal forms (10p)

a) (3 p) Draw a table for the contents of a relation r(a, b, c, d) such that the functional dependency a b → c holds, but neither a → b nor a → c holds, and d determines no other attribute (none of these FDs hold: d → a b c). Use the numbers 0,1,2... for values in table cells.

b) (3 p) Identify three separate FDs that hold on the data below. Each FD should have a single attribute on the left hand side (so not FDs like x y z -> w).

x	y	z	w	q
X1	Y1	Z1	W1	Q1
X2	Y1	Z1	W1	Q2
X1	Y1	Z1	W1	Q1
X1	Y2	Z2	W2	Q1
X1	Y3	Z1	W2	Q3

c) (2 p) Consider this set of functional dependencies (seven in total!):

F={
A→B,
B→A,
C D → F,
C E → F C,
D E → F,
E → F
}

Find a minimal basis F^- (also known as a minimal cover) for this set. As a reminder, this means a subset of F where no FD in F^- can be deduced from other FDs in F^-.

d) (2 p) Show the results of normalizing R(a,b,c,d,e) to fourth normal form using the multivalued dependency a c ↠ d e.

Solution 5:

a) Here is one solution, but there are many others. Three rows should be a minimum since a needs to have two identical rows (to falsify a->b) and two non-identical (to falsify d->a).

a	**b**	**c**	**d**
0	0	0	0
0	1	1	0
1	0	0	0

b) the only correct answer (disregarding order): y→z, y→w, q→x

c) F^{-}={
A→B,
B→A,
C D → F,
E → F
}

d)

R1(a,c,d,e)

R2(a,c,b)

Names of relations and order of attributes/relations may differ.

Question 6: Semi-structured data and other topics (10p)

A database containing placements of objects in a room for a 3D simulation is being migrated to a JSON document database. A JSON Schema for the new format is available (see below).

The database has two tables for rooms with sizes in three dimensions, and for the positions of items contained in rooms. Here is an example of table contents (three items in three rooms):

Table Rooms:

room	**x_size**	**y_size**	**z_size**
Room1	20	20	3
Room2	15	15	4
Room3	10	10	5

Table Room_content:

room	**item**	**x**	**y**	**z**
Room1	Lamp	10	15	5
Room2	Chair	5	8	2
Room2	Table	5	10	null

a) (5p) Write a JSON document that validates against the schema and correctly encodes the example data given above.

b) (5p) Write two JSON Path expression (should work for any valid document) for finding:

- Item names (e.g. lamp/chair/table) of items placed on y-position 8 in Room2.
- The z-size of rooms where the first item (based on order in array) is out of bounds in its z-position (meaning the item has a z-position that exceeds the z-size of the room). In the example, the result would be 3, since the Lamp in Room1 is out of bounds.

```
{
  "description":"keys are room names, values are room sizes and contents",
  "type":"object",
  "additionalProperties":{
    "type":"object",
    "properties":{
      "size":{
        "description":"room size on the form [x,y,z]",
        "type":"array",
        "items":{"type":"integer"},
        "minItems":3, "maxItems":3
      },
      "content":{
        "type":"array",
        "items":{
          "type":"object",
          "properties":{
            "x":{"type":"integer"},
            "y":{"type":"integer"},
            "z":{"type":"integer"},
            "item":{"type":"string"}
          },
          "additionalProperties":false,
          "required":["x","y","item"]
        }
      }
    },
    "required":["size", "content"],
    "additionalProperties":false,
  }
}
```

Solution 6:

a)


```
{
  "room1":{
    "size":[20,20,3],
    "content":[{"x":10,"y":15,"z":5,"item":"lamp"}]
  },
  "room2":{
    "size":[15,15,4],
    "content":[{"x":5,"y":8,"z":2,"item":"chair"},
               {"x":5,"y":10,"item":"table"}]},
  "room3":{
    "size":[10,10,5],
    "content":[]
  }
}
```


b)

$.room2.content[*]?(@.y==8).item

$.*?(@.size[2]<@.content[0].z).size[2]