Solutions included

Databases Exam

TDA357 (Chalmers), DIT621 (University of Gothenburg)

2022-08-25 14:00-18:00

Department of Computer Science and Engineering

Examiner: Jonas Duregård.

Will visit the exam hall at 14:40 and 17:00.

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 from exam date

Maximum points: 60

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

Question 1: SQL Data Definition Language (9 p)

The company *StairwayToHeaven* has been requested to help with the booking system of the tenant's association *ParadiseOnEarth*. For this purpose, they have created a database with the following relational schema. Notice that the schema is incomplete, specifically it's missing primary and secondary keys as well as value constraints.

```
Tenants (personal_nr, name, tel)
Building (tenant, apt_nr)
  tenant → Tenants.personal_nr
BookingPrices (facility, price)
Bookings (facility, day, time, apt_nr)
  facility → BookingPrices.facility
```

Basic information about tenants is kept in Tenants.

The building has 100 apartments with numbers 1 up to and including 100. Who rents which apartment is stored in Building. Observe that more than one person can be registered as tentants of a particular apartment, and a person can be the tenant of more than one apartment.

BookingPrices lists the cost of booking various facilities. The available facilities are 2 different washing rooms (with numbers 1 and 2) that are booked separately, a room for parties and a sauna. Adding additional facilities should not be possible (without modifying the table code).

Bookings states which apartment number has booked a particular facility on a particular day and time (day is a complete date with year, month and day). The party room can only be booked for the whole day and then the time is set to 8. The other facilities can be booked for 3 hours at times 8, 11, 14, 17 or 20.

Naturally, a facility cannot be double booked on a particular day/time. Additionally, no apartment can book the same facility more than once per day, but they can book different facilities (including both washing rooms) on the same day.

Hint: Use DATE for the datatype of day.

a) (4 pts) Define SQL tables for the relational schema above. Add primary keys, unique constraints and value constraints (CHECK constraints) needed to implement all of the requirements above.

b) (2 pts) A problem with this design is that bookings can be made for apartments that currently have no tenants. Write the reference constraint that would be required to prevent this (in schema form, no SQL needed), and explain why such a reference can not be added to the design (three to-the-point sentences or so should be enough).

c) (3 pts) Create an SQL query that lists the top ten most frequent bookings this year (that is, from 2022-01-01), for (apartment, facility)-pairs. For instance, the top of the list could say

that apartment 3 has booked the sauna 17 times, followed by apartment 2 booking the sauna 14 times followed by apartment 3 booking washing room nr. one 10 times etc.

The list should be extended to include the name of all the tenants in each apartment on the list, so the result can be more than 10 rows if any of the involved apartments has more than one tenant, basically duplicating booking-frequencies for each tenant of the apartments on the top ten list.

The result should have four columns for apartment number, tenant name, facility and total number of bookings.

Hint: Use LIMIT 10 at the end of a SELECT query to limit it to 10 entries. Be careful to apply this for the right SELECT in case you have subqueries. LIMIT should always be placed directly after an ORDER BY.

Solution 1:

```
-- 1 a)
CREATE TABLE Tenants (
  -- A text type like CHAR(10) works for this as well, and is arguably
better
  personal_nr NUMERIC PRIMARY KEY,
  name TEXT NOT NULL,
  -- Same thing as for personal_nr
  tel NUMERIC NOT NULL CHECK (tel > 0) );

CREATE TABLE Building (
  tenant NUMERIC NOT NULL REFERENCES Tenants,
  apt_nr INT CHECK (apt_nr > 0 AND apt_nr <= 100),
  -- Important to get the compound key right here, to allow co-tentants.
  PRIMARY KEY (tenant, apt_nr) );

CREATE TABLE BookingPrices (
  facility TEXT PRIMARY KEY CHECK (facility in ('sauna', 'wash1', 'wash2',
'party')),
  price NUMERIC NOT NULL CHECK (price >= 0) );

CREATE TABLE Bookings (
  facility TEXT NOT NULL REFERENCES BookingPrices,
  day DATE NOT NULL,
  time INT NOT NULL CHECK (time in (8, 11, 14, 17, 20)),
  apt_nr INT NOT NULL CHECK (apt_nr > 0 AND apt_nr <= 100),
  CHECK (facility != 'party' OR time = 8), -- Parties always at 8
  PRIMARY KEY (facility, day, time), -- Prevents double booking
  UNIQUE (facility, day, apt_nr) ); -- Prevents multiple bookings in a day

-- 1b
-- The reference would be apt_nr -> Building.apt_nr in Bookings. This does
not work in SQL though, since apt_nr is not a key of Builing. Making apt_nr
a key would prevent mutliple tentants in appartments.

-- 1c
-- top bookings
WITH MostBookings AS
(SELECT apt_nr, facility, COUNT(*) AS total
FROM Bookings
WHERE day >= '2022-01-01'
GROUP BY facility, apt_nr
ORDER BY total DESC
LIMIT 10)
SELECT apt_nr, name, facility, total
FROM (MostBookings JOIN Building USING (apt_nr))
     JOIN Tenants ON tenant = personal_nr
ORDER BY total DESC, apt_nr, facility;
```

Question 2: SQL queries and Relational Algebra (11 p)

We continue with the same domain as in question 1:

```
Tenants (personal_nr, name, tel)
Building (tenant, apt_nr)
  tenant → Tenants.personal_nr
BookingPrices (facility, price)
Bookings (facility, day, time, apt_nr)
  facility → BookingPrices.facility
```

a) (3+3 pts) Write an SQL query and a relational algebra that lists the name of all the tenants with their apartment numbers who have a particular booking today (CURRENT_DATE). The output should also contain which facility has been booked and at what time. Order the output first by the facility and then by the (ascending) time.

b) (2.5+2.5 pts) Bills for an apartment (for example the rent or the cost of all the monthly bookings) are equality distributed among all the tenants of that particular apartment.

So, if apartment number 1 has two tenants, then each of them pays half of the bills associated to the apartment.

Write an SQL query and a relational algebra that computes the percentage that needs to be charged to each of the tenants of each apartment. In the example of apartment number 1 above, the computed percentage should be 50.

Solution 2:

```
-- 2a
-- tenants with bookings today
SELECT apt_nr, name, facility, time
FROM Tenants, (Building NATURAL JOIN Bookings)
WHERE day = CURRENT_DATE AND tenant = personal_nr
ORDER BY facility, time;
```

RA is something like this:

$$\tau_{\text{facility, time}}(\pi_{\text{apt\_nr,name,facility,time}}(\sigma_{\text{day=CURRENT\_DATE}}(\text{Building X Bookings } X_{\text{tentant=personal\_nr}} \text{ Tenants})))$$

```
-- % per tenant in apartment
CREATE OR REPLACE VIEW Percentages AS
--2b
-- Using round is not strictly required.
SELECT apt_nr, ROUND(100/COUNT(*)) AS perc
FROM Building
GROUP BY apt_nr
ORDER BY perc DESC, apt_nr;
```

RA is something like this:

$$\tau_{\text{-perc,apt\_nr}}(\gamma_{\text{apt\_nr,100/count(*)->perc}}(\text{Building}))$$

Question 3: Views and Triggers (10 p)

We continue with the same domain as in question 1:

```
Tenants (personal_nr, name, tel)
Building (tenant, apt_nr)
  tenant → Tenants.personal_nr
BookingPrices (facility, price)
Bookings (facility, day, time, apt_nr)
  facility → BookingPrices.facility
```

a) (4 pts) Assume there is a view called Percentages(apt_nr, percent) with the solution to the question 2b). You should now help StairwayToHeaven to keep track of the monthly fee to be charged to each tenant for the use (Bookings) of the facilities this month. If a person is a tenant in more than one apartment, then the person will only get one fee which is the sum of all the fees he/she has to pay in each of the apartment he/she is a tenant of. The output should contain the personal number of the tenant and the amount to pay.

Hint: The expressions date_part('year', d) and date_part('month', d) gives the year and number of the month respectively for a given DATE value d. Using this with CURRENT_DATE gives the current year/month.

b) (6 pts) When a person misbehave in one of the facilities (for example, makes too much noise when having a party or does not clean after using the sauna), the person is added to a list of banned people, which means that *none of the apartments* the person is tenant of can place a booking for that particular facility during 90 days after the ban has been issued, even if the actual day the person would like to use the facility is beyond 90 days in the future.

Create an SQL table that keeps the information of who has been banned for booking which facility. Expired bans should be kept so that the steering group of the association has a record of how many bans were issued for a particular tenant.

Make sure to add all reasonable constraints, including primary and foreign keys, and unique constraints (if any).

After the banned-list has been introduced in the association, bookings are not as straightforward as before and StairwayToHeaven needs help to make sure only apartments with non-banned tenants on a facility can place a booking on it (you do not need to remove existing bookings, just prevent new ones from being added.

Hint: Standard arithmetic operators work on DATE values, meaning expressions like d+1 gives a date one day later than d.

Solution 3:

```
-- monthly bill
-- 3a
CREATE OR REPLACE VIEW MonthlyBill AS
WITH
BillPerApt AS
(SELECT apt_nr, SUM(PRICE)
FROM Bookings JOIN BookingPrices USING (facility)
WHERE date_part('year', CURRENT_DATE) = date_part('year', day) AND
      date_part('month', CURRENT_DATE) = date_part('month', day)
GROUP BY apt_nr),
BillPerTenant AS
(SELECT tenant, ROUND(sum * perc/100) AS apt_fee
FROM (BillPerApt JOIN Building USING (apt_nr)) JOIN Percentages USING
(apt_nr))
SELECT tenant, SUM(apt_fee) AS fee
FROM BillPerTenant
GROUP BY tenant;

-- 3b
CREATE TABLE Banned (
  personal_nr INT REFERENCES Tenants,
  facility TEXT REFERENCES BookingPrices,
  day DATE NOT NULL,
  PRIMARY KEY (personal_nr, facility, day) );

-- Test data not part of solution
INSERT INTO Banned VALUES (2020, 'sauna', '2022-07-20');
INSERT INTO Banned VALUES (2020, 'party', '2022-07-20');
INSERT INTO Banned VALUES (2017, 'wash1', CURRENT_DATE-10);
INSERT INTO Banned VALUES (2017, 'wash2', CURRENT_DATE-1);



-- insert a booking
CREATE OR REPLACE FUNCTION bookPremise() RETURNS TRIGGER AS $$
DECLARE
  nr INT;
BEGIN
 IF (EXISTS ((SELECT personal_nr
              FROM Banned
                 WHERE facility = NEW.facility AND day + 90 >= CURRENT_DATE)
                 INTERSECT
                 (SELECT tenant
                  FROM Building
                  WHERE apt_nr = NEW.apt_nr)))
 THEN RAISE EXCEPTION 'cannot insert';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER makeBookings
  BEFORE INSERT OR UPDATE ON Bookings
  FOR EACH ROW
  EXECUTE FUNCTION bookPremise();
```

Question 4: ER-modelling (11p)

a) (7 pts) You will make an ER-diagram for the employee structure of a university.

The university is divided into departments. Departments in turn are divided into multiple divisions, which in turn are divided into multiple units.

Departments, divisions, and units are identified by name with (only) the following limitations:

- No two departments can have the same name.
- There cannot be two divisions with the same name within a department.
- There cannot be two units with the same name within a division.

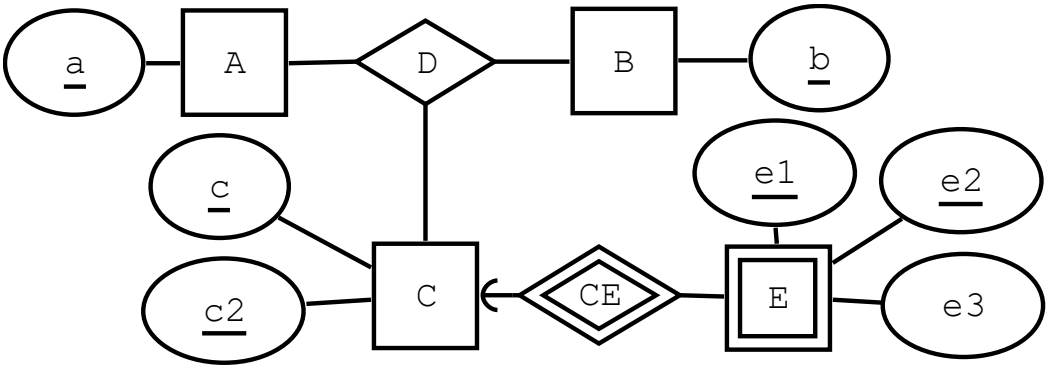
An example of a unit would be the "Functional Programming" unit of the "Computing Science" division of the "Department of Computer Science and Engineering".

Each employee is a member of a unit (and thus indirectly of a division and department). Each employee has a name and an identifying number.

Employees can be designated as heads of departments, heads of divisions and heads of units, with the following rules:

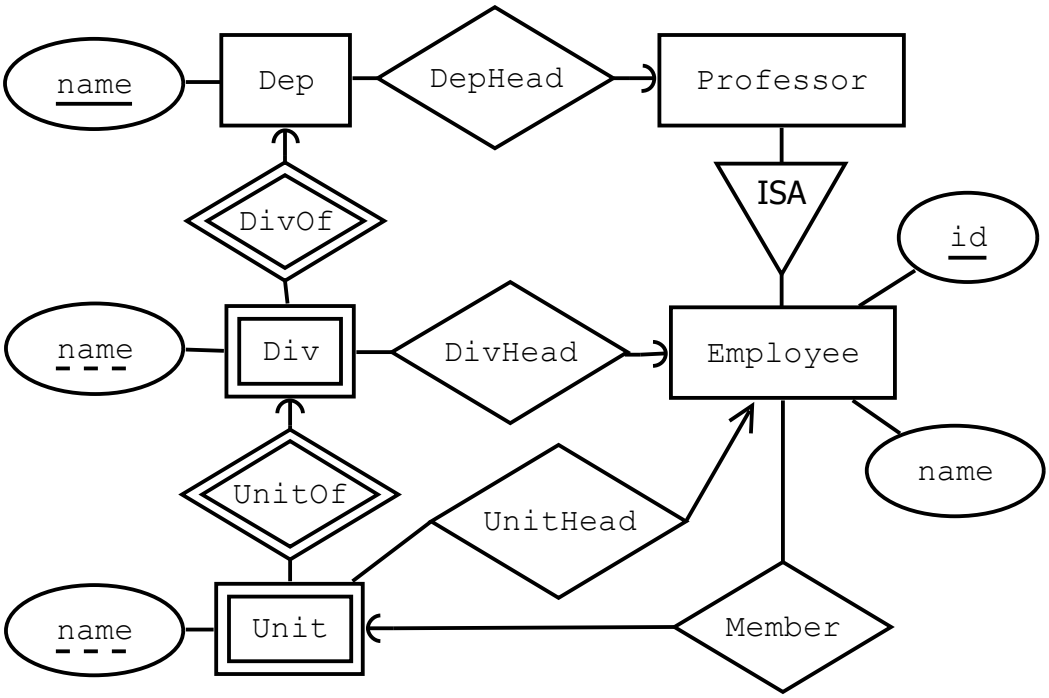
- Each department and division must have a single head.
- Each head of department must be a professor (and only some of the employees are professors).
- Units may or may not have a head (units that have no head are handled by the head of division).

b) (4 pts) Translate this (symbolic) ER-diagram into a relational schema, using the standard translation techniques:



Solution 4:

a)



b)

A(a)
B(b)
C(c, c2)
D(a, b, c, c2)
  a -> A.a
  b -> B.b
  (c,c2) -> C.(c,c2)
E(c, c2, e1, e2, e3)
  (c, c2) -> C.(c, c2)

Your solution should be an ER-diagram that translates into this schema using the standard translation rules for ER-diagrams

Question 5: Dependencies and normal forms (10p)

The questions below all relate to this this table with four columns (a,b,c,d) and five rows:

a	b	c	d
a0	b0	c0	d0
a0	b0	c0	d1
a1	b1	c1	d1
a2	b1	c1	d0
a0	b1	c0	d1

The values (a0, b0, ...) are symbolic, the only important thing is that a0 differs from a1 etc.

a) (2 pts) Which of the following FDs hold on the table data?

a b → c
a b → d
c b → a
c b → c

Your solution should be one **or more** of the FDs above, no motivation is required.

b) (3 pts) The FD c d → a holds on this table. Explain why it is also a BCNF violation.

Your answer should be a concise argument for why c d -> a is a BCNF violation (a single sentence is enough if it contains the crucial piece of information that makes it a violation). Points will be deducted for incorrect and/or irrelevant information. General definitions that do not mention anything specific to the given data are **not** acceptable.

c) (3 pts) Perform one BCNF decomposition using c d -> a, and then decompose the table data into the data of both resulting tables.

Your solution should be two tables with column names and table contents (rows). You do not need to mark primary keys.

Hint 1: The natural join of the two tables should be the original tables.
Hint 2: In a correct solution, the tables have different number of rows.

d) (2 pts) Find one more BCNF violation that still holds in the tables resulting from c). You do not need to perform normalization or motivate your answer, just find a violation.

Your solution should be a single functional dependency.

Solution 5:

a)

a b → c, and c b → c.

b)

Because c d → b does not hold (because (c0, d1) maps to both b0 and b1), meaning {c,d} is not a key.

c)

a	c	d
a0	c0	d0
a0	c0	d1
a1	c1	d1
a2	c1	d0

b	c	d
b0	c0	d0
b0	c0	d1
b1	c1	d1
b1	c1	d0
b1	c0	d1

d)

a -> c

This is the only violation as far as I can see. Keep in mind that a violation needs to have a single attribute on its left hand side /e.g. b c -> d cannot be a violation, since even if it held that would make {b, c} a key), so there are relatively few combinations to look for.

Question 6: Semi-structured data and other topics (9p)

a) (3 pts) A web shop is sending lists of products in a user-selected category, by running SQL queries constructed from this pattern:

```
SELECT product, price FROM Inventory WHERE category='<user input goes here>'
```

Explain using an example how an SQL Injection vulnerability in the web shop application could be used by an attacker to retrieve sensitive data from the web shop. Assume there is a table called Secrets, with columns 'user' and 'cardNumber' that the attacker wishes to access.

b) (3 pts) You are processing a list of servers where each server has a number and a location, and some servers additionally have a name. Here is a tiny example, in the form of a table:

Server Number	Location	Name
23	Chicago	CH1
15	Chicago	
18	New York	

A fellow engineer has written a JSON Schema for representing lists like this (see below). It has a few clever tricks like merging all servers in the same location in a single list, and just using numbers instead of objects to refer to servers that have no names.

Encode the example data above into a JSON document that validates against the schema.

c) (3 pts) Assuming the same schema, write a JSON Path expression for listing all server names. **NOTE**: "name" could occur in other parts of the document, so do **not** assume all "name"-attributes in the document are server names (basically, don't use **).

```
{
  "description":"Keys are names of cities, values are server lists",
  "type": "object",
  "additionalProperties": {
    "description":"A list of servers for a city",
    "type": "array",
    "items": {
      "oneOf": [
        {
          "description":"Servers with a number and a name",
          "type": "object",
          "properties": {
            "number": {"type": "integer"},
            "name": {"type": "string"}
          },
          "required": ["number","name"]
        }, {
          "description":"Servers with just a number",
          "type": "integer"
        }
      ]
    }}}
```

Solution 6:

a)

Injecting the UNION operator could append the secret to the end of the result. The attack user input would look something like:

```
' UNION SELECT user, cardNumber FROM Secrets --
```

b)

```
{ "Chicago" : [{"number":23, "name":CH1}, 15],
  "New York" : [18]
}
```

c)

```
$.*[*].name
```