Databases

TDA357/DIT621/DIT620 (4.5 hec)

Responsible: Ana Bove

Friday 19th of March 2021, 8:30–12:30

Total: 60 points	
CTH: $\geqslant$ 24: 3, $\geqslant$ 36: 4, $\geqslant$ 48: 5	GU: $\geqslant$ 24: G, $\geqslant$ 42: VG

General notes:

- You can use any help material but you CANNOT communicate with other people other than the teacher of the course in any way.
- There will be one assignment for each of the questions in the exam.
- You need to upload the solution to each question in the corresponding assignment in canvas. You have 30 min for this so it should be more than enough to get it right!
- We would prefer if you use a computer to type all your answers instead of writing them by hand since handwritten solutions are not always easy to read. In addition, pictures are sometimes not too sharp and clear. Think that what we cannot read we cannot correct!
- If you take pictures of your handwritten solutions please make sure your handwriting is clearly visible and readable in the picture.
- The file extensions allowed are pdf, jpg, jpeg, png, and and txt.
- We advice you to really start working on the uploading of your solution at 12:30 since at 13:00 sharp the system will close for submissions (unless you have extra time) and we will NOT accept submissions by mail.

Good luck!

1 SQL and Constraints (9 pts)

Upload your solution to this question into assignment 1 in canvas.
If possible use a computer to type your answer instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

A new agency for renting cars is about to open in Gothenburgh. They want to keep things as simple as possible and this is the relational schema they have decided to use to keep track of the bookings:

Persons (id, name, email, year)
Cars (regnr, model, doors, fee)
Rentals (person, car, fromdate, length)
  person → Persons.id
  car → Cars.regnr

The company records the year a person was born. Only people that are at least 25 years are allowed to rent cars and hence be stored in the database.

Cars have 2, 4 or 5 doors and a fee which is at least 500 kr per day.

The default renting period is 3 days, but it can be changed as long as it is at least 2 day.

a) (3 pts) Define SQL tables for this relational schema.

b) (3 pts) Write an SQL query that lists all cars which are available for renting for a period of one week starting today.

c) (3 pts) Write an SQL query that computes the average age of all people who have rented cars with at least 4 doors and for more than a week.

   Observe that if a person satisfies these conditions more than once, his/her age should only be used once when computing the average.

Note: In order to obtain the current year one can use the following PostgreSQL expression: date_part('year', CURRENT_DATE).

Also, CURRENT_DATE + 1 gives tomorrow's date and CURRENT_DATE - 1 yesterday's date.

Solution:

a)
```
CREATE TABLE Persons (
  id CHAR(10) PRIMARY KEY,
  name TEXT NOT NULL,
  email TEXT NOT NULL,
  year INT NOT NULL CHECK (date_part('year', CURRENT_DATE) - year >=25));

CREATE TABLE Cars (
  regnr CHAR(6) PRIMARY KEY,
  model TEXT NOT NULL,
  doors INT NOT NULL CHECK (doors IN (2,4,5)),
  fee INT NOT NULL DEFAULT 500 CHECK (fee >= 500));

CREATE TABLE Rentals (
  person CHAR(10) REFERENCES Persons,
  car CHAR(6) NOT NULL REFERENCES Cars,
  fromdate DATE,
  length INT NOT NULL DEFAULT 3 CHECK (length > 1),
  PRIMARY KEY (person, fromdate));
```

b)
```
(SELECT regnr FROM Cars)
EXCEPT
(SELECT regnr
FROM Cars JOIN Rentals ON regnr = car
WHERE (fromdate < current_date AND fromdate + length > current_date) OR
      (fromdate > current_date AND current_date + 7 > fromdate));
```

c)
```
SELECT AVG(age)
FROM  (SELECT DISTINCT person, date_part('year', CURRENT_DATE) - year AS age
       FROM (Cars JOIN Rentals ON regnr = car) JOIN Persons ON person = id
       WHERE doors > 2 AND length > 7) AS Ages;
```

2 Views and Triggers (10 pts)

Upload your solution to this question into assignment 2 in canvas.
If possible use a computer to type your answer instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

Recall the relation schema from question 1 on SQL:

Persons (id, name, email, year)
Cars (regnr, model, doors, fee)
Rentals (person, car, fromdate, length)
  person → Persons.id
  car → Cars.regnr

After running the company for a few years now, the owners would like to make some improvements in their database. Propose a solution (meaning, the corresponding full SQL code) to each of the changes the company want to make.

Recall that as a general rule, contraints, views and triggers should be applied in the appropriate order: if a constraint is enough just add it, and if a constraint or a view can adequately do the job, do not use a trigger!

a) (2.5 pts) Keep track of all non-active customers so that now and then the company can send them an email with a discount offer. A person is considered a non-active customer if the person has not rented a car in the last 3 years.

b) (4 pts) The company wants to introduce differential prices depending on the age of the driver, and the possibility to give discounts when a car is rented for a longer period.

   Customers that are younger than 30 years will pay 10% extra. On the other hand, those that rent a car for at least one week but less than two weeks will get a discout of 5%, and if the renting period exceeds 2 weeks then they will get a discount of 10%.

   With these new price regulations, keep track of the billing of all the customers that start renting a car today.

c) (3.5 pts) For each **new** rental request, make sure that a person can only rent one car at a time (that is, cannot start a new renting period before the other has finished) and that each car is only rented by one person at a time (that is, a car cannot be rented in two overlapping periods).

Solution:

a)
```
CREATE VIEW NonActiveCustomers AS (
SELECT name, email FROM Persons
EXCEPT
(SELECT name, email
 FROM Persons JOIN Rentals ON id = person
 WHERE fromdate >= CURRENT_DATE - 365*3));
```

b) OBS: the text actual indicates that one gets a 10% discount when the period exceeds 2 weeks. With that text a period of exactly 2 weeks will not have any discount. If you have answer in that way your answer is not exactly the one down here, but it will be considered corrected anyway of course!

```
CREATE VIEW Billing AS (
WITH PeriodFees AS
     (SELECT regnr, person, fee*length AS periodfee
      FROM Cars JOIN Rentals ON regnr = car
      WHERE length < 7 AND fromdate = CURRENT_DATE
      UNION
      SELECT regnr, person, fee*length*0.95 AS TotalFee
      FROM Cars JOIN Rentals ON regnr = car
      WHERE length >= 7 AND length <= 14 AND fromdate = CURRENT_DATE
      UNION
      SELECT regnr, person, fee*length*0.9 AS TotalFee
      FROM Cars JOIN Rentals ON regnr = car
      WHERE length > 14 AND fromdate = CURRENT_DATE)
SELECT id, name, regnr, periodfee*1.1 AS TotalFee
FROM PeriodFees JOIN Persons ON id = person
WHERE date_part('year', CURRENT_DATE) - year < 30
UNION
SELECT id, name, regnr, periodfee AS TotalFee
FROM PeriodFees JOIN Persons ON id = person
WHERE date_part('year', CURRENT_DATE) - year >= 30);


-- Alternative solution using CASE
CREATE VIEW BillingCase AS (
WITH PeriodFees AS
     (SELECT regnr, person,
             CASE WHEN length < 7  THEN fee*length
                  WHEN length < 14 THEN fee*length*0.95
                                   ELSE fee*length*0.9
             END AS periodfee
      FROM Cars JOIN Rentals ON regnr = car
      WHERE fromdate = CURRENT_DATE)
SELECT id, name, regnr,
       CASE WHEN date_part('year', CURRENT_DATE) - year < 30 THEN periodfee*1.1
                                                             ELSE periodfee
       END AS totalfee
FROM PeriodFees JOIN Persons ON id = person);
```

c)
```
CREATE OR REPLACE FUNCTION new_rental() RETURNS TRIGGER AS $$
BEGIN
 IF EXISTS (SELECT person, car FROM Rentals
            WHERE (person = New.person OR car = NEW.car) AND
          ((fromdate < NEW.fromdate AND fromdate + length > NEW.fromdate) OR
     (fromdate > NEW.fromdate AND NEW.fromdate + NEW.length > fromdate)))
    THEN RAISE EXCEPTION 'Rent already taking place in the period';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER rent_a_car
  BEFORE INSERT ON Rentals
  FOR EACH ROW
  EXECUTE FUNCTION new_rental();
```

3 Relational Algebra (8 pts)

Upload your solution to this question into assignment 3 in canvas.
If possible use a computer to type your answer instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

Recall the relation schema from question 1 on SQL:

Persons (id, name, email, year)
Cars (regnr, model, doors, fee)
Rentals (person, car, fromdate, length)
  person $\to$ Persons.id
  car $\to$ Cars.regnr

a) (2.5+3 pts) Write an SQL query and a relational algebra expression that gives the names and ages of all the people who have rented a car so far, that is, only rental periods that have already started should be taken into account here.

The list should be ordered by the total number of days of the renting periods put together, where those people who have rented more days in total should appear first in the list.

(You can use the same expression in relational algebra as in SQL to compute the age.)

b) (2.5 pts) Write a relational algebra query that gives the model of all cars with at least 4 doors, with a booking of at least 8 days by a driver who is at least 50 years.

Solution:

a)
```
SELECT name, date_part('year', CURRENT_DATE) - year AS age
FROM Persons JOIN Rentals ON id = person
WHERE fromdate <= current_date
GROUP BY id, name, year
ORDER BY SUM(length) DESC;
```

Relational algebra:

$$\pi_{\texttt{name, (date\_part('year',CURRENT\_DATE)-year)}\to\texttt{age}} \\ (\tau_{-\texttt{sum}}(\gamma_{\texttt{id, name, year, SUM(length)}\to\texttt{sum}}(\sigma_{\texttt{fromdate<=current\_date}}(\texttt{Persons} \bowtie_{\texttt{id=person}} \texttt{Rentals}))))$$

b)
$$\delta(\pi_{\texttt{model}}(\sigma_{\texttt{doors}\geqslant 4\,\wedge\,\texttt{length}\geqslant 8\,\wedge\,\texttt{year}\leqslant(\texttt{date\_part('year',CURRENT\_DATE)}-50)} \\ ((\texttt{Cars} \bowtie_{\texttt{regnr=car}} \texttt{Rental}) \bowtie_{\texttt{person=id}} \texttt{Persons})))$$

4 ER Modelling (12 pts)

Upload your solution to this question into assignment 4 in canvas.
If you write your diagram by hand, please make sure the picture you upload of it can be seen properly.
If possible use a computer to type your relational schema instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

Askim riding school in Gothenburg wants help to digitalise their data, which we describe below.

The school has many pupils that ride one or more times a week. Each pupil has a name and a unique client number (billing information is collected around this number), an email address that is used for communication between the school and the pupil, and a telephone number for ICE purposes (riding is a dangerous sport!)

Instruction is done in groups, and there is only one group per day and time. Each group has a level and in some cases, also an extra characteristic like ”no jump”, ”jump group”, etc. Also, groups can have 45 or 60 min lessons.

A pupil can only ride in groups that are at the level of their riding experience or lower, and groups cannot have more than 8 pupils in order to comply with the Public health authority restrictions in Covid-19 times.

There are a handful of instructors working in the riding school and each group is assigned an instructor that will held the lessons. Their telephone numbers is known to the school but not to the pupils.

Askim riding school is a relatively small school (though it plans to expand) and has at the moment about 20 horses with different names. Quite a lot of information is needed for horses, for example their age, when they arrived to the school, where they come from (Poland, Ireland, Holland, etc), how much food they should eat. Sometimes horses get injured or sick and need to rest until they get better.

Every day one assigns a horse for each pupil in each of the groups of the day. Of course, only active horses (those not resting) will be assigned.

Some of the advance pupils are “contract riders”, which means that they can come at extra times (when there is no instruction going on at the school) and ride the horse they have a “contract” with. The starting date of the contract needs to be recorded.

All pupils in the school need to be a member of Askim riding club in order to be insured in case of accidents. The club has several working groups that take care of different activities (competitions, café, premises, etc). Each pupil can be active in one of those working groups if they so want.

a) (6 pts) Draw an ER-diagram for the domain described above.

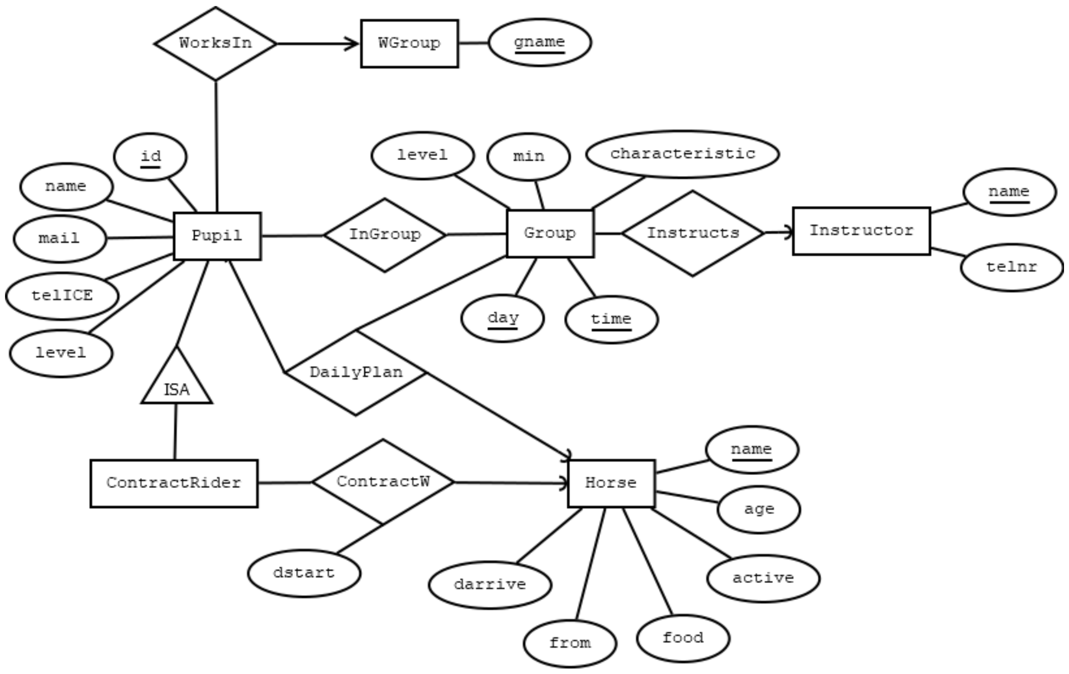
   When it has not been explicitly said how elements in certain entities are to be identified you must come up with a reasonable solution.

   You will also notice that some information in the description of the domain is not really relevant when doing an ER-diagram and you might ignore it.

b) (6 pts) Translate your diagram into a relational schema.

Solution:

a)



b) Pupils (<u>id</u>, name, mail, telICE, level)
Instructors (<u>name</u>, telnr)
Groups (<u>day</u>, <u>time</u>, instructor, level, min, extra)
  instructor → Instructors.name
Horses (<u>name</u>, age, active, food, from, darrive)
InGroups (<u>gday</u>, <u>gtime</u>, <u>rider</u>)
  gday → Groups.day
  gtime → Groups.time
  rider → Pupils.id
DailyPlans (<u>gday</u>, <u>gtime</u>, <u>rider</u>, horse)
  gday → Groups.day
  gtime → Groups.time
  rider → Pupils.id
  horse → Horses.name
  UNIQUE (gday, gtime, horse)
ContractRiders (<u>rider</u>, horse, dstart)
  rider → Pupils.id
  horse → Horses.name
WGourps (<u>gname</u>)
WorksIn (<u>rider</u>, wgroup)
  rider → Pupils.id
  wgroup → WGroups.gname

To ensure only pupils in the group get a horse assigned one could add the reference (gday, gtime, rider) → InGroups (gday, gtime, rider) in DailyPlans.

An alternative solution for DailyPlans is a many-to-many-to-exactly-one from groups and horses to pupils (the exact one) and have UNIQUE (gday, gtime, rider) instead.

5 Functional and Multivalued Dependencies (11pts)

Upload your solution to this question into assignment 5 in canvas.
If possible use a computer to type your answer instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

a) Consider the relational schema `R(a,b,c,d,e,f)` and the following functional dependencies:

$\texttt{a} \rightarrow \texttt{b}$
$\texttt{e} \rightarrow \texttt{f}$
$\texttt{b c} \rightarrow \texttt{d f}$

i) (3 pts) Construct a table (as simple as possible) containing these and only these dependencies.
Note: 6 rows seem to be enough so try not to have many more than 6 rows.
(1 pt) Explain your solution as best as you can!

ii) (3pts) Convert the schema into a relational schema in BCNF. Show all the steps in the process. Do not forget to state the primary keys in the final schema!
(1 pt) Do you see any problem with the final relational schema that was obtained?

b) Consider the relational schema `R(g,h,i,j,k)` and the multivalued dependency $\texttt{g h} \twoheadrightarrow \texttt{i}$.

i) (2 pts) Complete the following table so that the dependency is valid

g	h	i	j	k
0	0	1	2	2
0	0	1	3	3
0	0	4	2	2
5	5	1	2	2
5	5	1	3	3

ii) (1 pt) Convert the schema into a relational schema in 4NF.

Solution:

a) i)

a	b	c	d	e	f
2	0	0	0	2	0
1	0	0	0	3	0
3	1	1	0	3	0
3	1	2	2	4	1
4	2	1	2	5	3
5	3	1	3	5	3

The table satisfies the given FD, in particular row 3 and 4 clearly guarantee $\texttt{a} \rightarrow \texttt{b}$, rows 1 and 2 clearly guarantee $\texttt{b c} \rightarrow \texttt{d f}$, and the last 2 rows clearly guarantee $\texttt{e} \rightarrow \texttt{f}$.

No other dependency is valid either here although this is more difficult to exhaustively explain.

It is easy to see however that non singel attribute determine other attributes (except for the to FD already present) since there are always 2 rows with the same value in a column but different values in the rest of the columns. Rows 2 and 3 show that `d f` do not determine `b c`.

ii) We start with `R(a,b,c,d,e,f)` and the dependency `a` $\rightarrow$ `b` which is a BCNF violation.

After the split we get `R1(a,b)` which is in BCNF and `R2(a,c,d,e,f)`.

We now consider `e` $\rightarrow$ `f` which is a BCNF violation in `R2(a,c,d,e,f)`.

After the split we get `R21(e,f)` which is in BCNF and `R22(a,c,d,e)`.

From `a` $\rightarrow$ `b` we can get `a c` $\rightarrow$ `b c` by augmentation and reflexivity, and by transitivity we get `a c` $\rightarrow$ `d f`. So `a c` $\rightarrow$ `d` is a violation of `R22(a,c,d,e)` and we then need to split into `R221(a,c,d)` and `R222(a,c,e)`.

Final relational schema with the keys: R1($\underline{\texttt{a}}$,b), R21($\underline{\texttt{e}}$,f), R221($\underline{\texttt{a}}$,$\underline{\texttt{c}}$,d) and R222($\underline{\texttt{a}}$,$\underline{\texttt{c}}$,$\underline{\texttt{e}}$).

This final schema doesn't guarantee the FD `b c` $\rightarrow$ `d f`.

b) i)

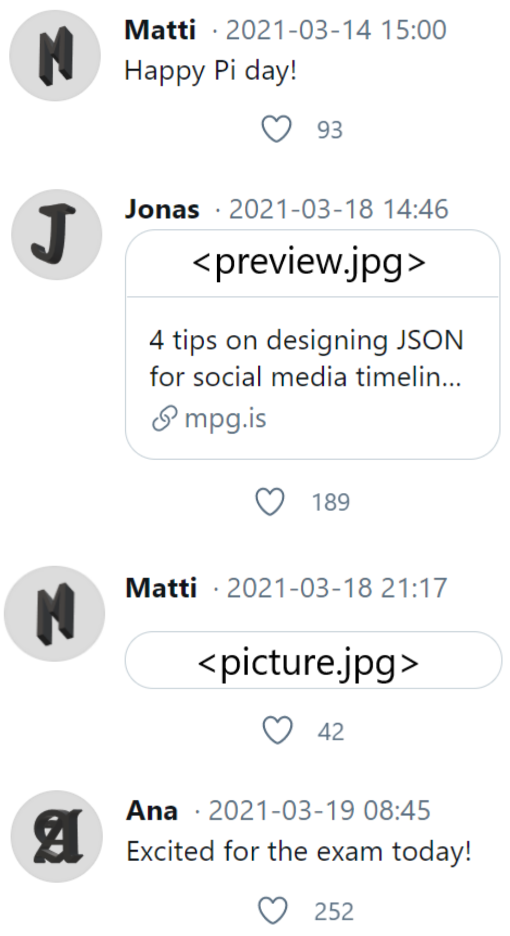
g	h	i	j	k
0	0	1	2	2
0	0	1	3	3
0	0	4	2	2
0	0	4	3	3
5	5	1	2	2
5	5	1	3	3

ii) Using this MVD we obtain the schemas R1($\underline{\texttt{g}}$,$\underline{\texttt{h}}$,$\underline{\texttt{i}}$) and R2($\underline{\texttt{g}}$,$\underline{\texttt{h}}$,$\underline{\texttt{j}}$,$\underline{\texttt{k}}$).

6 JSON (10 pts)

Upload your solution to this question into assignment 6 in canvas.
If possible use a computer to type your answer instead of writing it by hand.
File extensions allowed are pdf, jpg, jpeg, png, and and txt.

Below is a sketch of the timeline of the latest social network YASN (Yet Another Social Network). It is still in development, and at the moment it only has three types of posts: text, images, and links. Each post has the timestamp of when it was posted, a user with a name and a display picture, and its content, type, and number of likes on the post.



Matti · 2021-03-14 15:00
Happy Pi day!
93

Jonas · 2021-03-18 14:46
<preview.jpg>
4 tips on designing JSON
for social media timelin...
mpg.is
189

Matti · 2021-03-18 21:17
<picture.jpg>
42

Ana · 2021-03-19 08:45
Excited for the exam today!
252

a) (3 pts) Encode the timeline above as a JSON document. The information in your JSON document should represent exactly the data needed to draw the image. All pictures are represented by file names: “m.jpg”, “j.jpg”, and “a.jpg” for the profile pictures, and “picture.jpg” and “preview.jpg” for the picture/preview. Timestamps are represented as a string, e.g. “2021-03-14 15:00” represents 15:00 at March 14th.

b) (5 pts) Write a JSON schema for the timeline that validates against your encoding of it, and enforces that the content of the post matches its type (e.g. a text post should only have the content of a text post, and not the content of an image post etc.).

c) (2 pts) Write a JSON Path expression that returns all the timestamps of posts by Matti (should return two results for the timeline above).

Solution:

a)
```
[
     { "user": { "name": "Matti" , "picture": "m.jpg"},
       "time": "2021-03-14 15:00",
       "likes": 93,
       "type" : "text",
       "content": {"text": "Happy Pi day!"}
     },
     { "user": { "name": "Jonas" , "picture": "j.jpg"},
       "time": "2021-03-18 14:46",
       "likes": 189,
       "type": "link",
       "content": { "text": "4 tips on designing JSON for social media timelin..."
                  , "location": "mpg.is"
                  , "preview": "preview.jpg"}
     },
     { "user": { "name": "Matti" , "picture": "m.jpg"},
       "time": "2021-03-18 21:17",
       "likes": 42,
       "type": "image",
       "content": {"image": "picture.jpg"}
     },
     { "user": { "name": "Ana" , "picture": "a.jpg"},
       "time": "2021-03-19 08:45",
       "likes": 252,
       "type": "text",
       "content": {"text": "Excited for the exam today!"}
     }
]
```

b)
```
{
   "type": "array",
   "definitions": {
     "user": { "type": "object",
               "properties": { "picture": { "type": "string", "minLength": 1 },
                               "name": { "type": "string", "minLength": 1 } },
                "additionalProperties": false,
                "required": ["name", "picture"] },
     "textContent": {"type": "object"
                    , "properties": {"text": {"type": "string", "minLength": 1}}
                     , "additionalProperties": false
                    ,  "required": ["text"]},
     "imageContent": {"type": "object",
                       "properties": {"image":{"type": "string", "minLength": 1}},
                        "additionalProperties": false,
                        "required": ["image"]},
    "linkContent": {"type": "object",
                     "properties": {"text": {"type": "string", "minLength": 1},
                                    "location": {"type": "string", "minLength": 1},
                                    "preview": {"type": "string", "minLength": 1}},
                     "additionalProperties": false,
                     "required": ["text", "location", "preview"]},
    "post": {"allOf": [
        {"type": "object",
         "properties": {"user": {"$ref": "#/definitions/user"},
                        "time": {"type": "string"},
                        "likes": {"type": "number", "minimum": 0},
                        "content": true,
                        "type": {"enum": ["text", "link", "image"]}
                 },
        "additionalProperties": false,
        "required": ["user", "time", "likes", "content", "type"] },
        {"oneOf": [{"properties": {
                       "content": {"$ref":"#/definitions/textContent"},
                        "type": {"const": "text"}}},
                   {"properties": {
                      "content": {"$ref":"#/definitions/imageContent"},
                      "type": {"const": "image"}}},
                   {"properties": {
                      "content": {"$ref":"#/definitions/linkContent"},
                      "type": {"const": "link"}}}] }]}},
"items": {"$ref": "#/definitions/post"}
}
```

c)
```
WITH Timeline AS
   (SELECT '
    <object as in part a) of this exercise>
    '::JSONB AS val)
SELECT jsonb_path_query_array (val,
             'strict $.**?(@.user.name == "Matti").time')
FROM Timeline as res;
```