Databases Exam

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

2025-01-15 14:00-18:00

Department of Computer Science and Engineering

Examiner: Jonas Duregård. Will visit at least twice. Phone: **031 772 1028**

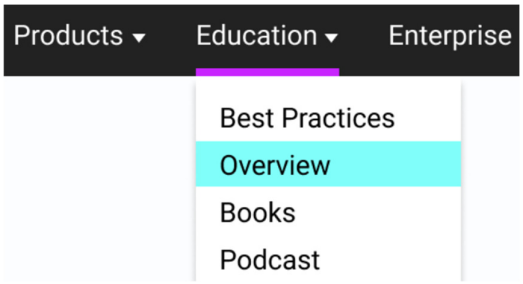
Allowed aids: One double sided A4 page of hand-written notes, the notes should be handed in along with your solution. Write your anonymous code on the notes if you wish, but do not write your name on it.

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. (DIT621: 27 for G, 45 for VG)

Question 1: Constraints, triggers, views and tables (12 p)

You are working on an (incomplete) database for a drop-down menu like this one:



The categories on the top (Products, Education,...) are called tabs. The menu items under a tab can either refer to pages stored in the database (TabPages) or be links to external pages (TabLinks). Tabs and items have position values to indicate which order they appear in (higher values appear further right for tabs/down for items). Here is the (incomplete) code for the database including inserts to create the menu above, with three external links and one page (Overview) in the Education tab):

```
CREATE TABLE Pages(
  name TEXT NOT NULL PRIMARY KEY,
  content TEXT NOT NULL DEFAULT 'Under Construction');

CREATE TABLE Tabs(
  label TEXT NOT NULL PRIMARY KEY,
  position INTEGER NOT NULL);
```

```
CREATE TABLE TabPages(
  page TEXT NOT NULL REFERENCES Pages,
  tab TEXT NOT NULL,
  position INTEGER NOT NULL
  );

CREATE TABLE TabLinks(
  linktext TEXT NOT NULL,
  URL TEXT NOT NULL,
  tab TEXT NOT NULL,
  position INTEGER NOT NULL,
  clickCount INTEGER NOT NULL DEFAULT 0,
  );

INSERT INTO Tabs VALUES ('Products', 0);
INSERT INTO Tabs VALUES ('Education', 10);
INSERT INTO Tabs VALUES ('Enterprise', 20);
INSERT INTO Pages VALUES ('Overview', 'Overview page...');
INSERT INTO TabPages VALUES ('Overview', 'Education', 0);
INSERT INTO TabLinks VALUES
       ('Best Practices', 'example.com/bp', 'Education', 0);
INSERT INTO TabLinks VALUES
       ('Books', 'example.com/books', 'Education', 1);
INSERT INTO TabLinks VALUES
       ('Podcast', 'example.com/pc', 'Education', 2);
```

Describe in detail (with SQL code) what modifications/additions you make to add each the features below. Use any parts of SQL we have taught in the course including table elements (columns and constraints), views, triggers, ...). Trigger functions can be written in pseudo-code (the body of the function and the CREATE TRIGGER statement are the important parts).

Note: Describe clearly what modifications you make, and where you make them, e.g. "add this constraint to TabLinks: <SQL code for constraint definition>".

a) Keep a count of how many links (not counting pages) each Tab has.

b) Keep a count of the number of page visits (an integer value) for each Page.

c) When a Tab is deleted, all associated menu items (TabLinks and TabPages) should be deleted automatically.

d) If a TabPage is inserted for a page name that doesn't exist, that page should be created with the default content value (you can ignore modifications done in other parts of the question).

e) Prevent two tabs from having the same position value.

f) Prevent two TabPages in the same Tab from having the same position value.

Hint: As always, avoid overusing triggers.

Solution 1:

a) Add a view like this (getting the SQL exactly correct is not required):

```
CREATE VIEW TabsCount AS
  SELECT MB.*, (SELECT COUNT(*) FROM TabLinks WHERE tab=MB.label) AS
linkcount
  FROM Tabs AS MB;
SELECT * FROM TabsCount;
```

b) Add a column to Pages like visits INTEGER NOT NULL or such.

c) Add a constraint to both TabPages and TabLinks:
tab REFERENCES Tabs ON DELETE CASCADE;

d) This requires a trigger as such (pseudocode for the function is ok):

```
CREATE FUNCTION addLink() RETURNS trigger AS
$$ BEGIN
    IF NOT EXISTS(SELECT * FROM Pages WHERE name=NEW.page) THEN
        INSERT INTO Pages VALUES (NEW.page, DEFAULT);
    END IF;
    RETURN NEW;
END$$
LANGUAGE plpgsql;

CREATE TRIGGER addLink
    BEFORE INSERT ON TabPages
    FOR EACH ROW
    EXECUTE PROCEDURE addLink();
```

e) Add UNIQUE (position) to Tabs.

f) Add PRIMARY KEY (tab, position) to TabPages.

Question 2: SQL Queries(8p)

Using the same tables as in Question 1:

a) (4p) Write an SQL query that lists all menu items (links and pages) for all menu tabs. The query should have four columns for bar, position, menutext, and url. For pages, url should be null and menutext should be the name of the page. To make sure all positions are unique, links will use even numbers and pages odd numbers (so if a link and page both have position 3 in the tables, the link gets position 3*2=6 and the page 3*2+1=7). The result for the data above would be:

```
Education, 0, Best Practices,  example.com/bp
Education, 1, Overview,        (null)
Education, 2, Books,           example.com/books
Education, 4, Podcast,         example.com/pc
```

b) (4p) Write an SQL query for finding the combined click count of links in each tab (0 for tabs that have no links). The result should have two columns for label (of the tab) and totalClicks (the sum of all clickCounts).
Hint: The SUM of null is null (not 0).

Question 3: More queries and RA (10p)

a) (6 p) Write both an SQL query and a relational algebra expression for finding the tab label of all tabs that contain more than three pages (TabPages). The result should have a single column that contains tab labels (like "Education", if the education tab had more pages in it).

b) (4 p) Write a relational algebra expression (no SQL needed) for finding all links that have the same URL as at least one link in another tab. The result should have two columns, for the tab label and menu text of the links. The result should not have any duplicate rows.

Solution 2:

a)
Note: The question says “bar” where it should say “tab”.

```
SELECT tab, position*2 AS position, linktext, url FROM
TabLinks
UNION
SELECT tab, position*2+1, page, null FROM TabPages ;
SELECT * FROM MenuItems;
```

b)

```
SELECT label, COALESCE (SUM(clickCount), 0) AS totalClicks
FROM Tabs LEFT JOIN TabLinks ON label=tab
GROUP BY label;
```

Solution 3:

a)

```
SELECT tab
FROM TabPages
GROUP BY tab
HAVING COUNT(*) > 3;
```

$\pi_{\text{tab}}(\sigma_{\text{num}>3}(\text{gamma}_{\text{tab, count(*)->num}}(\text{TabPages})))$

b) It’s possible to do this with gamma, but it’s much more complicated.

$$\delta(\pi_{\text{A.tab, A.menuText}}(\sigma_{\text{A.url=B.url AND A.tab}\neq\text{B.tab}}(\rho_{\text{A}}(\text{TabLinks}) \times \rho_{\text{B}}(\text{TabLinks}))))$$

Question 4: ER-modelling (12p)

a) (7p) A company has given you the task of designing a database for a vague business venture of theirs, involving users and facilities.

Draw an ER-diagram for the database, these requirements are given by the company:

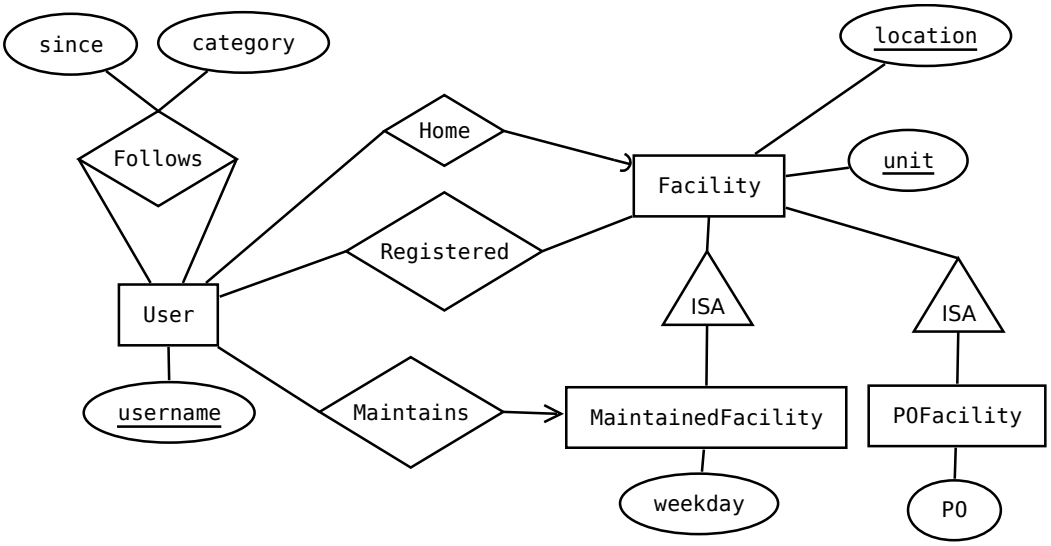
- Every user has their own username.
- Users can "follow" other users online. The database should keep track of the time when a user started following the other user, and the following user can assign the followed user a category (an arbitrary text label like "colleagues", "family" or anything else, not a fixed set of categories).
- The company has facilities in various locations. Sometimes there are multiple facilities in one location, so each facility is designated a unit ID to distinguish them from other facilities (so three facilities could be "unit1 on Johanneberg", "unit2 on Johanneberg" and "unit1 on Lindholmen").
- Every user has a designated home facility, but they can also be registered to any number of additional facilities.
- Some facilities have a PO box number (you don't need to know what that is).
- Some facilities are maintained by users. Each of these facilities have decided a weekday when the maintaining users meet up. Each of these facilities can have any number of maintaining users, but a user cannot be maintainer of more than one facility.

b) (5p) Translate your diagram into a relational schema. The company has requested that you use the null approach whenever possible (a weirdly specific demand, but the customer is always right). Make sure to write "(or null)" after any attribute in your schema that can be null.

Hint: In general the null approach can be used when a subentity has a single attribute and no relationships, and when a many-to-at-most-one relationship doesn't require any compound references.

Solution 4:

Note: The word "can" is an unfortunate choice of words for "user can assign the followed user a category". The category isn't optional here:



b)

Facility(<u>location</u>, <u>unit</u>, po (or null))

MaintainedFacility(<u>location</u>, <u>unit</u>, weekday)
(location, unit)->Facility(location, unit)

User(<u>username</u>, homeLocation, homeUnit)
(homeLocation, homeUnit)->Facility(location, unit)

Registered(<u>user</u>, <u>location</u>, <u>unit</u>)
(location, unit)-> Facility(location, unit)

Follows(<u>user</u>, <u>follows</u>, since, category)
user -> User.username
follows -> User.username

Maintains(<u>user</u>, location, unit)
(location, unit)-> MaintainedFacility(location, unit)
user -> User.username

Question 5: Dependencies and normal forms (10p)

FDs: Consider this (symbolic) domain with seven attributes and seven functional dependencies:
R(a, b, c, d, e, f, g)

e → d	d → e	e → f	a c → b
a e → e	c → a	d → f	

a) (2p) Write a minimal cover for this set of functional dependencies (your solution should be a reduced set of functional dependencies equivalent to the ones above, where no dependency can be derived from the others and with as few left hand side attributes as possible).

b) (3p) Normalize R into BCNF. You should show the intermediate steps, but make sure your final normalized schema is stated in its entirety at the end of your solution. You do <u>not</u> need to mark keys.
Hint: Double check that you included all of R in your normalization.

MVDs: R describes lab sessions much like the ones in this course. R is in BCNF.
R(<u>course</u>, <u>ta</u>, <u>time</u>, <u>room</u>)

- Every course has a set of times (day and hour) when lab sessions start.
- During each lab session, a number of rooms are booked for the course.
- Teaching assistants (ta) can be booked to attend some of the sessions of one or more courses (not for any particular room, they roam around).
- Sometimes multiple courses share a room (e.g. when two similar courses are given in parallel).

c) (2p) Your colleague argues that course ↠ ta, since every course has a set of teaching assistants working in it. Show that is not a valid MVD with a counterexample (a possible content of R where the MVD does not hold). Use as few rows as you can. Use these values for columns (you will only need a few of them):

course: {TDA357, TDA417}	ta: {Lorenzo, Niklas}
time: {Tue 8:00, Wed 10:00}	room: {NC1, NC2}

d) (3p) Give another suggestion for an MVD that does hold on R, and is a 4NF violation. Also give the content of the two tables resulting from the 4NF normalization, using the data from your counterexample in (c).

Solution 5:

a) You end up with these five

e → d
d → e
c → b
c → a
d → f

a e→e is trivial, e→f follows from e→d and d→f, a c → b is reduced to just c→b since c → a.

b) You should end up with these three after two steps of decomposition:

R1(e,d,f)
R2(c,a,b)
R3(c,e,g)

The intermediate step would have either (e,c,a,b,g) or (c,e,d,f,g) depending on the order of decomposition.

c) One example. Here Lorenzo and Niklas are booked for different sessions (so TA is not independent from time).

(TDA357, Tue 8:00, Lorenzo, NC1)
(TDA357, Wed: 10:00, Niklas, NC1)

d) course time ↠ ta should hold (and equivalently course time ↠ room)

For the data above, we get these tables:

R1 (course, time, room):
(TDA357, Tue 8:00, NC1)
(TDA357, Wed 10:00, NC1)

R2 (course, time, ta):
(TDA357, Tue 8:00, Lorenzo)
(TDA357, Wed 10:00, Niklas)

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

Consider this JSON document for storing information about courses (number of students, when they have lab sessions and which TAs are on the sessions):

```
{
    "TDA357":{
        "students":240,
        "labSessions":[
          {"time":"2024-12-20", "tas":["Lorenzo"]}
        ]
    },
    "TDA417":{
      "students":190
    }
}
```

a) (4p) Sketch a JSON Schema for documents like this. These are the requirements:

- The course codes ("TDA357", "TDA417" in example) can be any strings, all other object keys ("students", "labSessions", "time", "tas") are fixed.
- Types must be respected (e.g. the "tas" key should be an array of strings etc.
- All properties should be required unless they are absent in some part of the example.

Note: Do not add any other constraints, e.g. for date formatting or limits on array sizes etc.

b) (3p) Write a JSON Path for finding all TA lists of lab sessions on 2024-12-20 (in all courses). In the example, this would give a single array ["Lorenzo"] (but it can give multiple arrays in general).

c) (3p) Write a JSON Path for listing all individual TAs in TDA357 lab sessions after 2024-09-01 (you may assume comparison operators on strings work for dates). Every result should be a string (the name of a TA). In the example, it would only give "Lorenzo" as result (but of course it should work for any valid document).

Solution 6:

a)

Arguably can be omitted since the question was phrased “unless they are absent”.

```
{
    "type":"object",
    "additionalProperties":{
      "type":"object",
      "properties":{
        "students":{"type":"integer"},
        "labSessions":{
          "type":"array",
          "items":{
            "type":"object",
            "properties":{
                "time":{"type":"string"},
                "tas":{"type":"array", "items":{"type":"string"}}
            },
            "required":["time","tas"]
          }
        }
      },
      "required":["students"]
    }
}
```

b)

$.*.labSessions[*]?(@.time=”2024-12-20”).tas

c)

$.tda357.labSessions[*]?(time > “2024-09-01”).tas[*]