Introduction to Database Design / Data Management
MSc and BSc Exams

Björn Thór Jónsson

August 13, 2020

Instructions

You have 4 hours to answer 6 problems described in the following. There are 7 problems in the exam, but problem 2 is only for BSc students and problem 3 is only for MSc students. The exam consists of 11 numbered pages.

Instructions for SQL Queries in Question 1

Queries must work for any database instance and should avoid system-specific language features, including the LIMIT keyword. Queries should not return anything except the answer; a query that returns more information will not receive full points, even if the answer is part of the returned result. A sequence of several queries that answer the question will not receive full points, but subqueries and views can be used. Queries should be as simple as possible; queries that are unnecessarily complex may not get full marks, despite returning the correct answer. If you are unable to complete the query you can still submit your attempt, along with a brief description of the problem with the query, and it may be given partial points.

Database Description for Questions 1–3

In this exam you will work with a fictional database for a social media site. To start working with the database, run the commands in `idb-august-2020-DB.sql` using the PostgreSQL DBMS on your laptop. It is recommended to use `psql` for this purpose.

The database contains a variety of information in the following schema:

```
Zips(zip, municipalityID, city)
Users(ID, name, zip)

Relationships(fromID, toID)
Roles(fromID, toID, role)

Posts(ID, posterID, time, text, url)
Likes(userID, postID)
Comments(ID, postID, posterID, userID, text)
```

Primary keys and foreign keys are defined and attributes are largely self-explanatory. You may study the DDL commands to understand the details of the tables (the CREATE TABLE statements are at the top of the script), consider the ER-diagram in Figure 1 on the next page, or inspect the tables using SQL queries. Some additional notes are in order:

- The Zips table is the result of normalisation. The 'municipalityID' in Zips is an identifier for a township or such. It would normally refer to a Municipalities table, but this table is not needed and therefore omitted here.
- Relationships are directed from one user to another, so A ('fromID') may have a relationship to B ('toID') although B has no relationship to A. Each relationship can have different roles, which are represented in the Roles table.
- Any user can "like" any post, but to allow a comment on a post there must be some relationship from the poster to the commenter; the table Posts has a secondary key (UNIQUE) to allow the enforcement of this condition.
- The actual texts of posts and comments are tragically boring; my apologies for this!

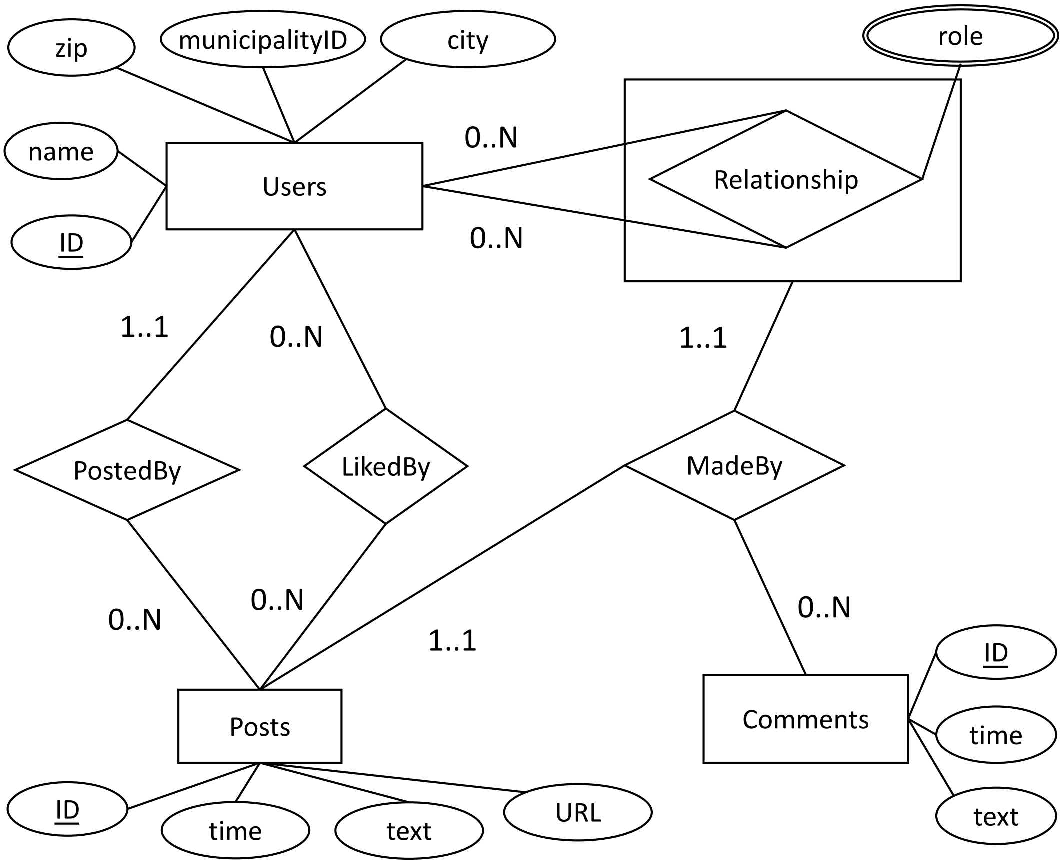


Figure 1: ER Diagram for the social media database.

1 SQL (40 points)

Answer each of the following questions using a single SQL query on the examination database:

(a) There are 286 users which are related to some other users (= have a 'fromID'). How many users have someone related to them (= have a 'toID')?

(b) How many bidirectional relationships exist in the database? Here, if A has a relationship to B and B has a relationship to A that is one bidirectional relationship.

(c) In total, there are 1939 entries in the Relationship table. How many of those relationship entries have no associated entry in the Roles table?

(d) Needless to say, posts have a varying number of comments (some have none). For example, post 24 has 3 comments. What is the ID of the post that has the most comments?

Note: This query returns an identifier, not a count of result rows.

(e) There are 6 posts which have more than 5 comments. How many posts have 2 or fewer comments?

(f) In the database, there are 90 different users who have commented on the post of their spouse. How many different municipalities have at least one user who has commented on the post of their spouse?

(g) All in all, you should find that there are 3 different roles in the database. There are 83 users who have a relationship ('fromID') to some other users with all the roles in the database. How many users have some other user related to them ('toID') with all the roles in the database?

Note: This is a division query; points will only be awarded if division is attempted. In particular, you must not use the constant 3 in the query.

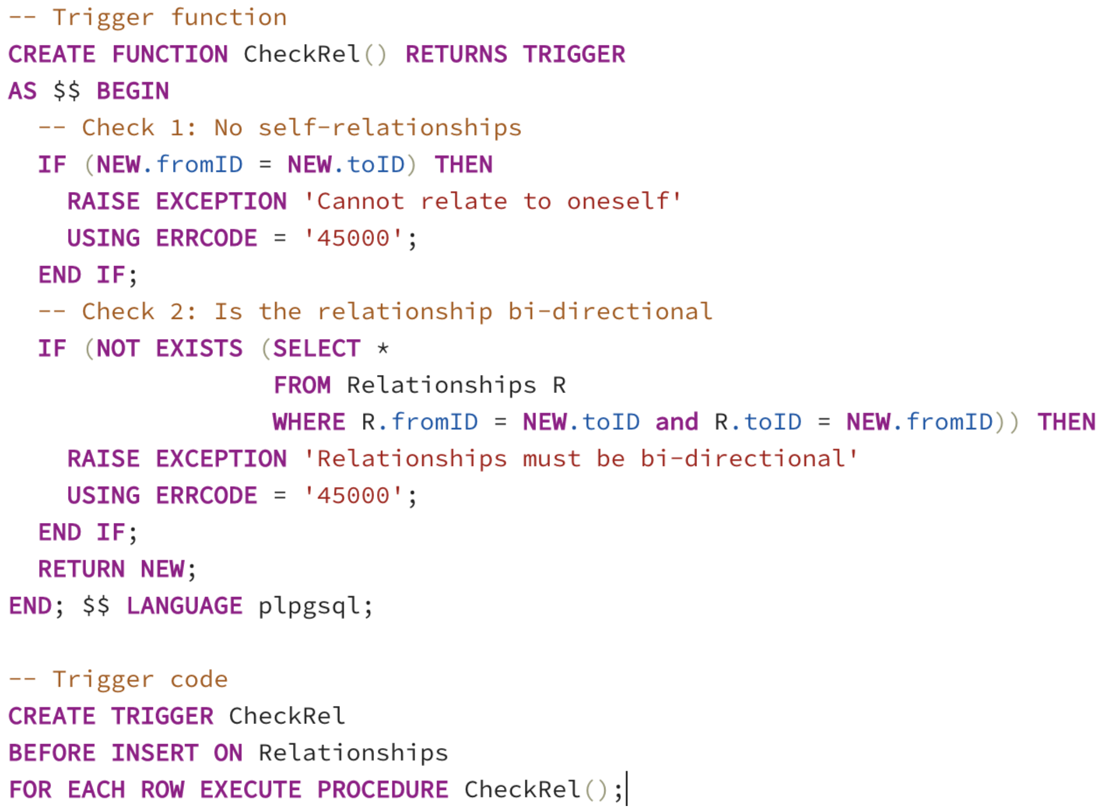
(h) How many users have posted a post which has at least one like, but no comments?

Note: Query complexity will impact the grade strongly for this query.

Enter each query, along with its numerical answer, in LearnIT. Queries must adhere to the detailed guidelines given on Page 1.

2 (BSc ONLY) SQL programming (5 points)

Consider the SQL trigger code in Figure 2, which aims at preventing self-relationships and ensuring bidirectional relationship roles. For example, if A is a ‘Friend’ of B, then B should be a ‘Friend’ of A.



```
-- Trigger function
CREATE FUNCTION CheckRel() RETURNS TRIGGER
AS $$ BEGIN
  -- Check 1: No self-relationships
  IF (NEW.fromID = NEW.toID) THEN
    RAISE EXCEPTION 'Cannot relate to oneself'
    USING ERRCODE = '45000';
  END IF;
  -- Check 2: Is the relationship bi-directional
  IF (NOT EXISTS (SELECT *
                  FROM Relationships R
                  WHERE R.fromID = NEW.toID and R.toID = NEW.fromID)) THEN
    RAISE EXCEPTION 'Relationships must be bi-directional'
    USING ERRCODE = '45000';
  END IF;
  RETURN NEW;
END; $$ LANGUAGE plpgsql;

-- Trigger code
CREATE TRIGGER CheckRel
BEFORE INSERT ON Relationships
FOR EACH ROW EXECUTE PROCEDURE CheckRel();
```

Figure 2: Insertion trigger `CheckRel` for the `Relationships` relation.

Select the true statements:

(a) Check 1 can be replaced by a CHECK constraint on the `Relationships` relation.

(b) Check 2 can be replaced by a CHECK constraint on the `Relationships` relation.

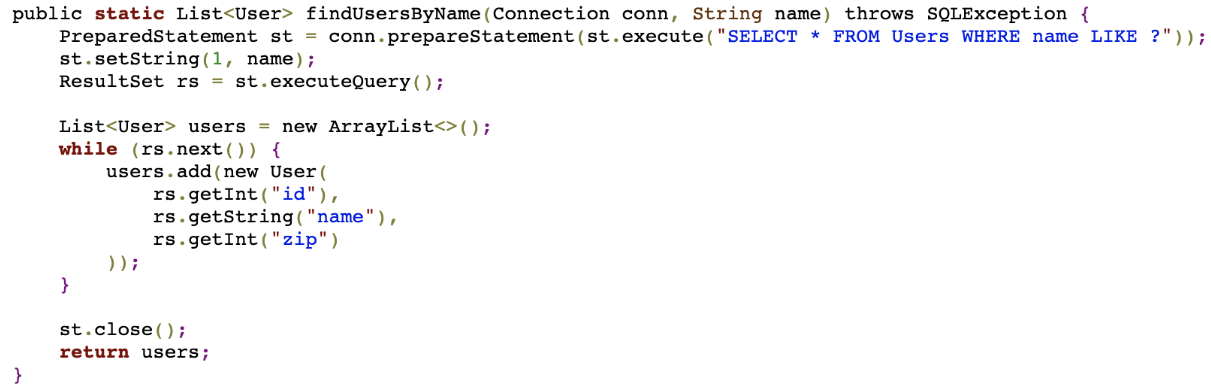
(c) Creating the trigger for the given database instance will fail because there are already relationships that are not bi-directional.

(d) Adding relationships for new users will always fail.

(e) The trigger is not defined for the correct table to achieve its goal of ensuring bidirectional relationship roles.

3 (MSc ONLY) Database programming (5 points)

Consider the Java code in Figure 3.



```
public static List<User> findUsersByName(Connection conn, String name) throws SQLException {
    PreparedStatement st = conn.prepareStatement(st.execute("SELECT * FROM Users WHERE name LIKE ?"));
    st.setString(1, name);
    ResultSet rs = st.executeQuery();

    List<User> users = new ArrayList<>();
    while (rs.next()) {
        users.add(new User(
            rs.getInt("id"),
            rs.getString("name"),
            rs.getInt("zip")
        ));
    }

    st.close();
    return users;
}
```

Figure 3: Code for retrieving information from the `Users` relations.

Select the true statements:

(a) The code is safe against SQL injection attacks.

(b) The code is not using Object Relational Mapping.

(c) The query should use a transaction for better security.

(d) The query will be compiled on the server, and only the parameters are sent when the query is executed.

(e) The method allows the name parameter to contain wildcards, e.g., “Peter %”.

4 ER Diagrams and Normalization (25 points)

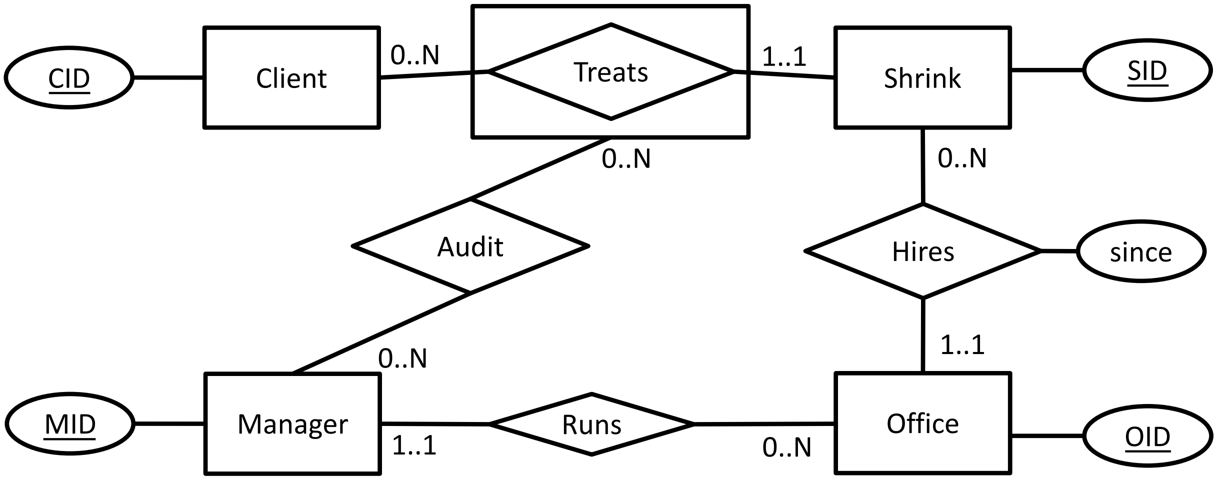


Figure 4: ER Diagram for a psychiatric facility database.

a) The ER diagram in Figure 4 shows a database for a psychiatric facility. Select the statements that are definitely true, based **only** on the ER diagram:

(a) Every client is related to exactly one manager via the audits relationship.

(b) Every client is related to exactly one manager when the audits relationship is excluded.

(c) Every manager is related to exactly one client when the audits relationship is excluded.

(d) Managers might also be clients.

(e) If a shrink is hired by a new office, information about old hirings disappears.

(f) Managers cannot audit treatment of clients in their own offices.

b) Write SQL DDL commands to create a psychiatric facility database based on the ER diagram in Figure 4 using the methodology given in the textbook and lectures. The DDL script *must run* in PostgreSQL. The relations must include all relevant primary key, candidate key, foreign key and NOT NULL constraints. Constraints that cannot be enforced with these standard constraints should be omitted. All attributes should be of type INTEGER.

c) Write an ER diagram for a car sales database. The diagram should clearly show the entities, relationships and participation constrains described below. Attributes are only important if they are mentioned in this description; you should not add other attributes. Follow precisely the notation presented in the textbook and lectures.

- Dealerships are identified by a unique ID.
- Each dealership has multiple showrooms, which are identified with a name. Car dealers are not very creative, so many of them simply name their showrooms using single letters, e.g., ‘A’ and ‘B’.
- Each dealerships has a contract with at least one manufacturer.
- Each car is made by exactly one manufacturer.
- Each car is either new or used. For used cars, the mileage is registered.
- Dealerships can promote any car, regardless of manufacturer, but they can only promote each car in one showroom.

d) Consider a table $R(A, B, C, D, E)$ with the following dependencies:

$$\begin{aligned} AB &\rightarrow CDE \\ DE &\rightarrow ABC \\ C &\rightarrow AE \end{aligned} \tag{1}$$

Select the true statements:

(a) AB is the only (candidate) key of R.

(b) $ABC \rightarrow A$ is a redundant functional dependency.

(c) Normalizing to 3NF/BCNF results in exactly two relations.

(d) The relation cannot be normalized to BCNF without losing functional dependencies (excluding trivial, unavoidable, and derivable dependencies) .

e) Consider a table $R(A, B, C, D, E)$ with the following dependencies:

$$\begin{aligned} AB &\rightarrow CDE \\ C &\rightarrow A \\ C &\rightarrow D \\ D &\rightarrow E \end{aligned}$$

Normalize R to the highest possible normal form (3NF or BCNF), based on functional dependencies, while allowing all functional dependencies (excluding trivial, unavoidable, and derivable dependencies) to be checked within a single relation. For each resulting relation, write its columns and clearly indicate whether it is in BCNF.

5 Index Selection (10 points)

Consider the Likes relation from the social media database:

Likes(userID, postID)

For each of the queries below, select one or more of the available indexes that a good query optimiser is most likely to use for the Likes table to process the query. Assume that all indexes are *unclustered* B+-trees. Also, assume here that the Likes relation has billions of entries. For each case, select the best index (or indexes) for the query, or select "no index" if a full table scan would yield better or equal performance than any of the available indexes.

(a) No index

(b) Likes(userID)

(c) Likes(postID)

(d) Likes(userID, postID)

The queries are:

Query 1

```
select count(*)
from Likes
where userID = 25;
```

Query 2

```
select max(postID)
from Likes
where userID = 25;
```

Query 3

```
select distinct userID
from Likes
order by userID;
```

Query 4

```
select count(distinct userID)
from Likes
where postID = (select max(postID) from Likes);
```

6 Hardware and DBMS Design (10 points)

a) Select the correct statements below:

(a) Main memory cannot break or fail, so database backups are not needed for main-memory database systems.

(b) CAP-style consistency is actually very similar to ACID-style isolation.

(c) Defragmentation (reorganizing files together on disk) does not make much sense on SSDs.

(d) By definition, a relational database can only use B+-tree indexes.

b) Discuss whether a mobile app, used by millions of users, that connects to a database server with strong consistency should be considered strongly consistent.

7 Data Systems for Analytics (10 points)

a) Select the correct statements below:

(a) Data in big data collections is never correct.

(b) Clustering algorithms can be used to identify groups of related records in, for example, banking applications.

(c) Key-value stores are better for web-caching than analytics applications.

(d) Spark is significantly more flexible than Hadoop for complex processing pipelines.

b) Explain why using a Spark pipeline running on large HDFS-based text files is a poor choice for running an application based on simple point queries.