Introduction to Database Systems BSc and MSc Retake Exams

Björn Thór Jónsson

March 12, 2021

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.

Database Description for Questions 1–3

In this exam you will work with a fictional (and poorly designed!) database of vaccinations. To start working with the database, run the commands in `idb-march-2021-DB.sql` found in LearnIT 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:

```
Categories(ID, name)
Diseases(ID, catID, code, name, curable)

Companies(ID, name, country)
Vaccines(ID, name, disID, comID, effectyears)

People(ID, name, birthyear)
Injections(peoID, vacID, injectionyear)
```

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. Following are some additional notes that you need for the queries:

- The attribute `curable` is a boolean value (can be either `true` or `false`).
- The `effectyears` attribute of a vaccine encodes its *effect* as the number of years that immunity to the disease lasts following an injection. An injection is called *active* in 2021 if `injectionyear` + `effectyears` $\geq$ 2021.

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.

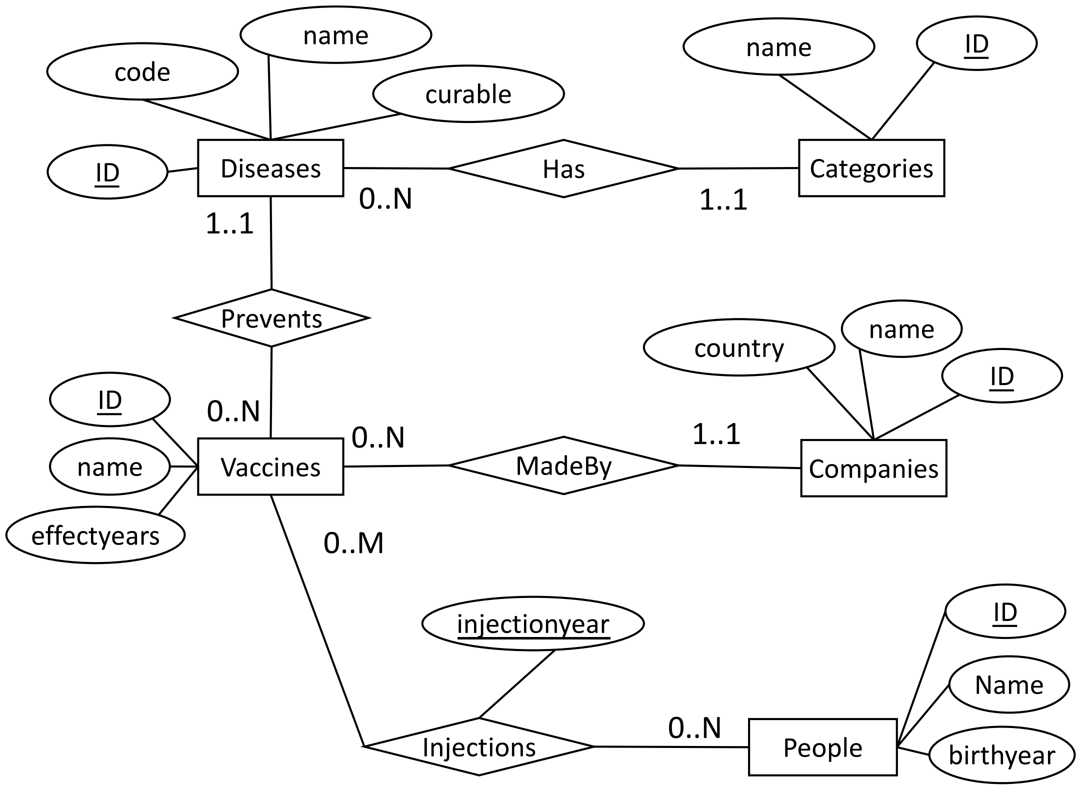


Figure 1: ER Diagram for the vaccine database.

1 SQL (40 points)

Answer each of the following questions using a single SQL query on the examination database. Enter each query, along with its numerical answer, in LearnIT. Queries must adhere to the detailed guidelines given on Page 2.

(a) The database lists 23 vaccines produced by Amgen. How many vaccines have been produced for the disease Coronavirus?

(b) There are 282 people that have received some injection of a vaccine for an incurable disease from the category Immune diseases. How many people have received some injection of a vaccine for a *curable* disease from the category Bone diseases.

(c) The Coronavirus vaccine with the shortest effect has ID = 535. What is the ID of the Coronavirus vaccine with the longest effect?

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

(d) How many diseases have fewer than 5 vaccines?

(e) According to the database, there are 113 people who have an active vaccination in 2021 for Sleep Apnea. How many people have an active vaccination for Coronavirus in 2021?

Note: In this query, you are allowed to compare to the number 2021. Alternatively, `date_part('year', CURRENT_DATE)` *would return 2021 at the time of the exam.*

(f) Person with ID = 23 has had 4 different vaccinations. What is the average number of vaccinations for each person in the database?

Note: This question will accept a range that includes the correct floating point number.

(g) There are 150 people who have been vaccinated for all diseases in category Immune diseases. How many people have been vaccinated for at least one disease from each category?

Note: This is a division query; points will only be awarded if division is attempted.

(h) What is the ID of the oldest person which has had injections of the most different vaccines for any one curable disease from category Immune diseases?

Note: This query returns an identifier, not a count of result rows. Having the most different vaccines for the same disease is most important; being the oldest is second most important.

2 (BSc ONLY) SQL programming (5 points)

Consider the SQL trigger code in Figure 2, which aims at ensuring the correctness of the `injectionyear` attribute.

```
CREATE FUNCTION CheckYear() RETURNS TRIGGER
AS $$ BEGIN
  -- Check 1: In the injection now?
  IF (NEW.injectionyear <> date_part('year', CURRENT_DATE)) THEN
    RAISE EXCEPTION 'Injection must be now'
    USING ERRCODE = '45000';
  END IF;
  -- Check 2: Is the person born?
  IF (NEW.injectionyear < (SELECT P.birthyear
                           FROM people P
                           WHERE P.ID = NEW.peoID)) THEN
    RAISE EXCEPTION 'Person must be born'
    USING ERRCODE = '45000';
  END IF;
  RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER CheckYear
BEFORE INSERT ON injections
FOR EACH ROW EXECUTE PROCEDURE CheckYear();

INSERT INTO injections(peoID, vacID, injectionyear) VALUES (20, 4, 2022);
INSERT INTO injections(peoID, vacID, injectionyear) VALUES (20, 4, 2021);
```

Figure 2: Insertion trigger `CheckYear` for the `injections` table.

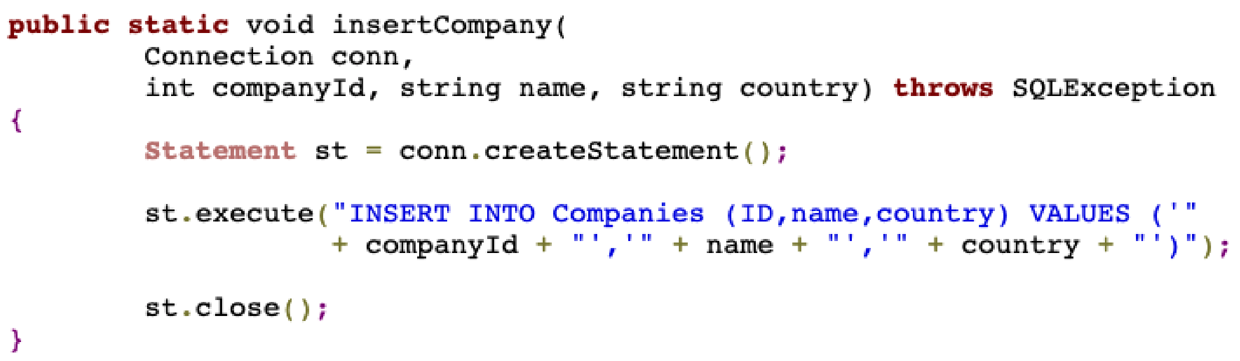
Select the true statements below.

(a) Check 1 can be replaced by a CHECK constraint on the `injections` table.
(b) Check 2 can be replaced by a CHECK constraint on the `injections` table.
(c) This trigger is more efficient as a BEFORE trigger than an AFTER trigger.
(d) The first INSERT statement will succeed (at the time of the exam).
(e) The second INSERT statement will succeed (at the time of the exam).

If in doubt, you can study the DDL for the table and query the database to see any relevant entries.

3 (MSc ONLY) Database programming (5 points)

Consider the Java code in Figure 3.



```
public static void insertCompany(
        Connection conn,
        int companyId, string name, string country) throws SQLException
{
        Statement st = conn.createStatement();

        st.execute("INSERT INTO Companies (ID,name,country) VALUES ('"
                   + companyId + "','" + name + "','" + country + "')");

        st.close();
}
```

Figure 3: Code for inserting information into the `Companies` relation.

Select the true statements:

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

(b) The code should be using transactions to preserve the integrity of the database.

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

(d) The code frees the resources associated to the statement when no longer needed.

(e) By default the company will be inserted into the database, even though conn.commit() is not called.

4 ER Diagrams and Normalization (25 points)

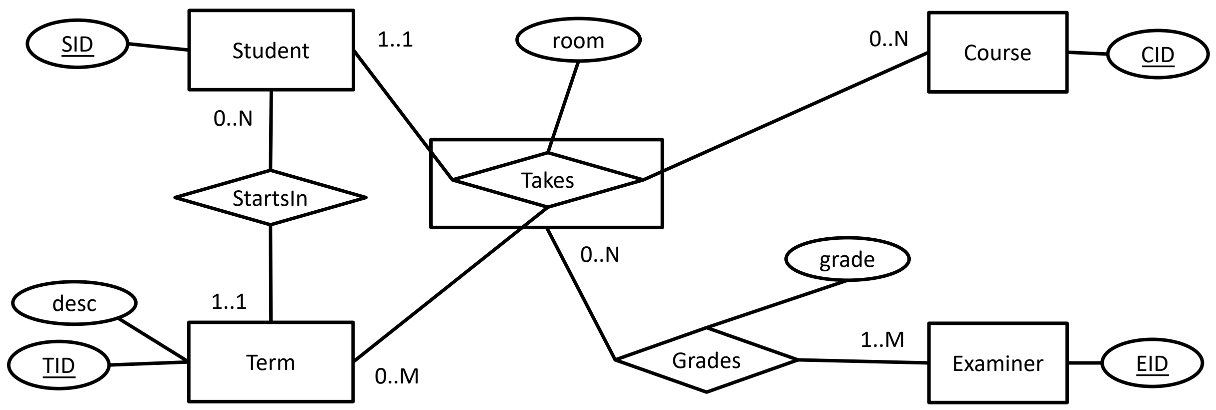


Figure 4: ER Diagram for a university database.

a) The ER diagram in Figure 4 shows a database for a (poorly designed) university application. Select the statements that are definitely true for any database instance, based **only** on the ER diagram:

(a) Multiple students can take the same course in the same term.

(b) Students must take at least one course in the term that they start in.

(c) Students must have at least one grade.

(d) Students can get multiple grades in the same course in the same term.

(e) For a course given in a term, at least one student must take the course.

(f) Each examiner must grade at least one student.

b) Write SQL DDL commands to create a university 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 as a whole. 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. Attributes should be of types INTEGER or VARCHAR, as appropriate.

c) Write an ER diagram for a database for a sports league. 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.

- A person may be either a coach or a player, or even both. Each person has ID and salary. Each coach has a CV.
- A player may have a contract with multiple clubs. If a player has multiple contracts with the same club, they must start on different days.
- Each club is controlled by one person; each person can at most control one club.
- Both clubs and people can be beneficiaries (here, a beneficiary is an entity that can accept sponsorships). Each beneficiary has a unique BN (short for beneficiary number).
- Each sponsor supports one beneficiary.

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

$$\begin{aligned} A &\rightarrow BC \\ B &\rightarrow D \\ C &\rightarrow E \end{aligned} \tag{1}$$

Select the true statements:

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

(b) $AC \rightarrow B$ is a trivial functional dependency.

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

(d) The relation can 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:

$$\begin{aligned} DE &\rightarrow ABC \\ A &\rightarrow C \\ B &\rightarrow D \end{aligned} \tag{2}$$

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 following large relation with phone call data:

Calls(fromnum, tonum, timestamp, duration, <detailed call info>)

Assume that all indexes are *unclustered* B+-trees. Also, assume that the relation has call data for the last 10 years with 10 billion requests, that calls are uniformly distributed over each day of those 10 years, that there are 100 million telephone numbers (tonum and fromnum) that are used uniformly. For each query below, select the index (or indexes) that a good query optimiser is most likely to use for the Calls table to process the query, or select "no index" if a full table scan would yield better performance than any of the possible indexes.

(a) Calls(fromnum, tonum, timestamp)

(b) Calls(duration)

(c) Calls(fromnum, duration)

(d) Calls(duration, fromnum)

(e) Calls(timestamp)

(f) No index

The queries are:

Query 1

```
select fromnum
from Calls
where duration = (select max(duration) from Calls);
```

Query 2

```
select sum(duration)
from Calls
where fromnum = 12345678;
```

Query 3

```
select max(timestamp)
from Calls
where tonum = 87654321;
```

Query 4

```
select *
from Calls
where tonum = 87654321 and fromnum = 12345678;
```

6 Hardware and DBMS Design (10 points)

a) Select the correct statements below:

(a) A lock manager is necessary to implement ACID-style isolation.

(b) When a transaction is committed, all the pages it has changed must be written to disk.

(c) A distributed system with multiple replicas of data items can provide both strict consistency and availability at the same time.

(d) A nested-loops join implementation is necessary for relational systems, to evaluate joins with complicated conditions.

b) Consider a web content distributor, which serves many static web-pages based on unique URLs. Discuss, with arguments, which of the classes of systems discussed during the course would be most suitable for this workload.

7 Data Systems for Analytics (10 points)

a) Select the correct statements below:

(a) Without the ability to join relations, there would be no big data analytics applications.

(b) Value in big data analytics refers to more than monetary value.

(c) In big data applications, there is no need to worry about inserting new data.

(d) A major reason why Spark is more efficient than Hadoop is that it uses memory more effectively.

b) Consider a clustering algorithm—such as k-means, which was briefly described in class—which repeatedly reads the whole collection with the goal of writing a new re-organised copy of it, where each cluster is located together on the disk. Assume that memory is limited, such that the clusters must be created incrementally. Argue which part of this process benefited most from the transition from HDDs to SSDs.