Introduction to Database Systems
BSc and MSc Exams

Björn Thór Jónsson

January 23, 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.

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 (and poorly designed!) database of car sales. To start working with the database, run the commands in `idb-january-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:

```
Makers(ID, name)
Models(ID, name, firstyear, lastyear, makerID)

Cars(ID, licence, prodyear, modelID)
People(ID, name, gender, birthyear)

Sales(ID, carID, personID, saleyear)
Sellers(saleID, personID)
```

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 *firstyear* and *lastyear* attributes of *Models* represent the time period during which a model was made; if *lastyear* is NULL, it is still being made. The *prodyear* attribute of *Cars* is the year the actual car was built; it should fit within the range of actual production, but that is not always the case in the database instance.
- The attribute *Licence* is a 3-5 character string, which contains two upper-case letter, followed by a number.
- In *Sales*, the (*carID*, *personID*) attribute pair is unique.
- The *Sellers* table contains records indicating which people took part in selling the car. If the car was bought without assistance, there is no row for the sale.

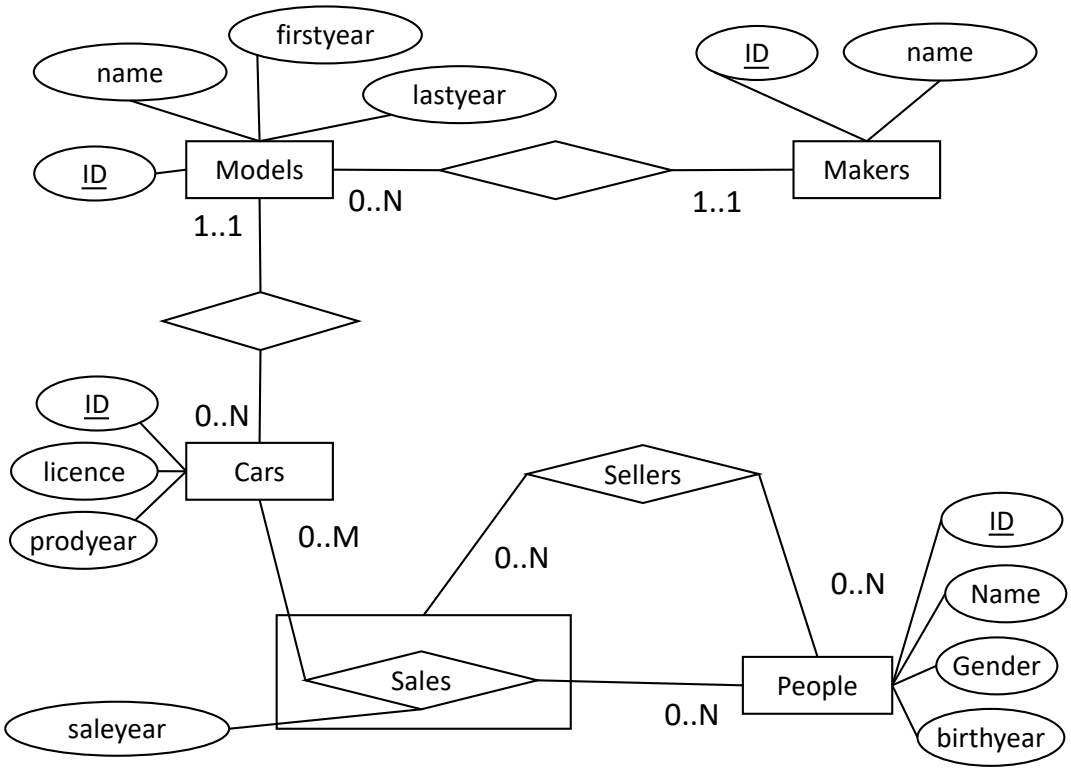

Figure 1: ER Diagram for the car sales database.

1 SQL (40 points)

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

(a) The database has 13 models made by the maker VOLVO. How many actual cars have been made by VOLVO?

(b) Person with ID = 34 has bought cars from 7 different makers. How many different makers has the person with ID = 45 bought from?

(c) Six different makers have produced the fewest models, or only one each. What is the ID of the maker which has produced the most models?

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

(d) How many cars have been sold fewer than 2 times?

(e) The car with licence LX363 has been sold eleven times, according to the database. What is the licence of the car sold most often?

Note: This query returns a short string.

(f) According to the sellers table, how many people have made at least one sale alone?

(g) There are 23 people who have bought cars of all models made by the maker LAMBORGHINI. How many people have bought cars of all models made by VOLVO?

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

(h) The database is randomly generated, and contains very little quality control relating to the different *-year* attributes in the various tables. One problem that may arise, for example, is that some cars are produced before the model production started ($Cars.prodyear < Models.firstyear$), but there are multiple other problems that might arise. How many different cars have *some* problem with one of the *-year* attributes?

Note: In this query, you must figure out the potential problems and embed them all into a single query. The more (real) problems you find, the higher your score.

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 ensuring some correctness of the `prodyear` attribute.

```
CREATE FUNCTION CheckYear() RETURNS TRIGGER
AS $$ BEGIN
  -- Check 1: Is the person born?
  IF (NEW.prodyear < (SELECT M.firstyear
                      FROM models M
                      WHERE M.ID = NEW.modelID)) THEN
    RAISE EXCEPTION 'Production must have started'
    USING ERRCODE = '45000';
  END IF;
  -- Check 2: In the production now?
  IF (NEW.prodyear <> (SELECT extract(year from CURRENT_DATE))) THEN
    RAISE EXCEPTION 'Production must be now'
    USING ERRCODE = '45000';
  END IF;
  RETURN NEW;
END; $$ LANGUAGE plpgsql;

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

INSERT INTO cars(ID, licence, prodyear, modelID) VALUES (5678, 'AB100', 2021, 1);
INSERT INTO cars(ID, licence, prodyear, modelID) VALUES (5679, 'AB101', 2020, 1);
INSERT INTO cars(ID, licence, prodyear, modelID) VALUES (5680, 'AB102', 1821, 1);
```

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

Select the true statements below.

(a) Check 1 can be replaced by a CHECK constraint on the *cars* table.

(b) Creating the trigger for the given database instance will fail because there are already cars that were made before the model started.

(c) The first INSERT statement will fail Check 2 (at the time of the exam).

(d) The second INSERT statement will succeed, because the given database instance does not have a car with licence AB101.

(e) The third INSERT statement will fail Check 1.

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 List<Model> getModelsByMakerId(Connection conn, int makerId)
                throws SQLException {
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(
                "SELECT * FROM models WHERE makerID = " + makerId);

        List<Model> models = new ArrayList<>();
        while (rs.next()) {
                Model model = new Model(
                        rs.getInt("ID"),
                        rs.getString("name"),
                        rs.getInt("firstyear"),
                        rs.getInt("lastyear"),
                        rs.getInt("makerID"));
                models.add(model);
        }

        rs.close();
        st.close();
        return models;
}
```

Figure 3: Code for retrieving information from the *Models* relations.

Select the true statements:

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

(b) If the SQL result is empty, rs.next() will throw an exception.

(c) The code should use a transaction for better consistency.

(d) The code is using Object Relational Mapping.

(e) The connection is closed automatically when a statement is closed.

4 ER Diagrams and Normalization (25 points)

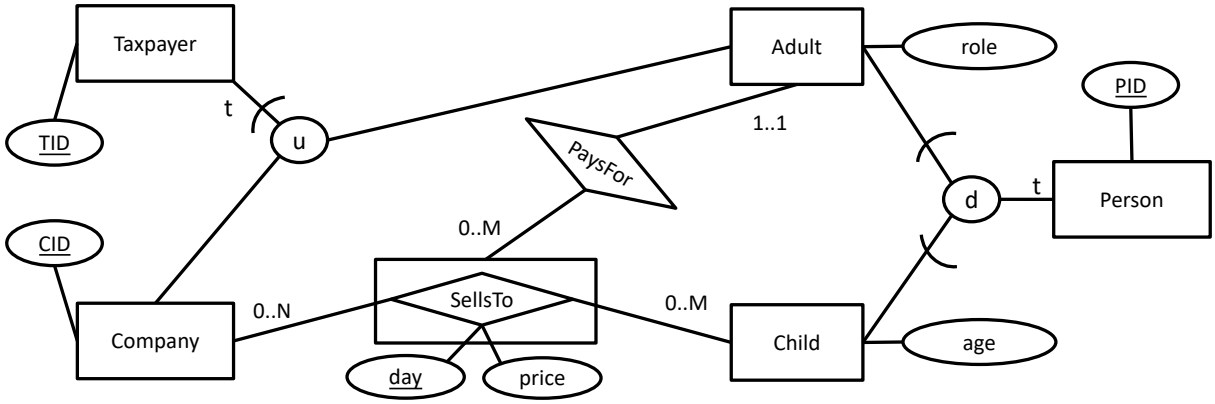

Figure 4: ER Diagram for an e-commerce database.

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

(a) Every Person is a Taxpayer.

(b) Every Adult is a Taxpayer.

(c) Each Company must get payments from the same Adult for all its sales.

(d) Each Company must get payments from different Adults for all its sales.

(e) One Taxpayer cannot be both a Company and an Adult.

(f) One Person cannot be both an Adult and a Child.

b) Write SQL DDL commands to create an e-commerce 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, VARCHAR or DATE, as appropriate.

c) Write an ER diagram for a medical equipment database for a hospital chain. 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.

- Each piece of medical equipment is of one (or more) of three different types: electrical; mechanical; or small.
- Each hospital in the chain has multiple wards; each ward has a unique number only within its hospital.
- One doctor runs each hospital ward.
- Doctors may have expertise in some pieces of medical equipment.
- Each piece of medical equipment is owned by one doctor.

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

$$\begin{aligned} DE &\rightarrow ABC \\ A &\rightarrow DE \\ D &\rightarrow B \end{aligned} \tag{1}$$

Select the true statements:

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

(b) $AC \rightarrow B$ is an unavoidable 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:

$$\begin{aligned} ABC &\rightarrow DE \\ C &\rightarrow C \\ C &\rightarrow D \\ E &\rightarrow AB \end{aligned}$$

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 following large relation with web logging data:

Requests(ID, userID, baseURL, options, timestamp, <detailed logging info>)

For each of the queries below, select the index that a good query optimiser is most likely to use for the Requests table to process the query. Assume that all indexes are *unclustered* B+-trees. Also, assume that the relation has request data for the last 5 years with 10 billion requests, that requests are uniformly distributed over each day of those 5 years, that there are 100 million users, but only 10 possible base URLs (the variability is in the options). For each case, select the best index for the query, or select "no index" if a full table scan would yield better performance than any of the possible indexes.

(a) Requests(ID)

(b) Requests(userID)

(c) Requests(baseURL)

(d) Requests(timestamp)

(e) Requests(userID, timestamp, ID)

(f) No index

The queries are:

Query 1

```
select *
from Requests
where baseURL = 'http://www.ThisURLisOneOfTheTen.com';
```

Query 2

```
select max(ID)
from Requests;
```

Query 3

```
select max(timestamp)
from Requests
where userID = 543423;
```

Query 4

```
select count(distinct userID)
from Requests
where options like '%include%';
```

6 Hardware and DBMS Design (10 points)

a) Select the correct statements below:

(a) Database backups are needed in case secondary storage (HDDs or SSDs) fails.

(b) Many NoSQL systems offer distributed storage.

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

(d) An RDBMS which implements clustered indexes can have multiple such clustered indexes per table.

b) Consider a world-wide distributed key-value store which enforces strict consistency: any read of a key will return the last data value inserted anywhere in the world. Explain why such a system could suffer considerable latency during regular operation (no hardware failures).

7 Data Systems for Analytics (10 points)

a) Select the correct statements below:

(a) Hadoop is significantly more efficient than Spark for complex processing pipelines.

(b) The main data access pattern of analytics applications consists of full sequential scans of large data collection.

(c) Using machine learning on existing data is likely to produce models that have built-in bias.

(d) In big data, *volume* refers to the massive quantities of data that must be stored and processed.

b) Discuss whether database systems could have a role to play in eliminating bias in machine learning.