Introduction to Database Systems
MSc and BSc Retake Exams

Björn Thór Jónsson / Philippe Bonnet / Andrea Vezzosi

August 20, 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. Unless instructed otherwise your answers must be provided in the LearnIT quiz *Retake Exam August 2021*.

Database Description for Questions 1–3

In this exam you will work with a fictional (and poorly designed!) database of flowerbeds. To start working with the database, run the commands in `idb-august-2021.sql` found in LearnIT using the PostgreSQL DBMS on your laptop. It is recommended to use `psql` for this purpose. The database has the following schema:

```
types(ID, name)
families(ID, typeID, name)
plants(ID, familyID, name)
people(ID, name, position)
parks(ID, name)
flowerbeds(ID, parkID, size, description)
plantedin(plantID, flowerbedID, percentage, planterID)
```

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, or inspect the tables using SQL queries. Following are some additional notes that are important for your queries:

- The terms used are not official scientific terminology; my apologies to horticulturists and other experts!
- In this exam, the term "plants" refers to the entity type, not individual plants.
- Plants belong to families, and families belong to types. For some plants, the family is unknown (note the 0..1 constraint in the ER diagram).
- The `flowerbeds.size` attribute is in square meters. The `plantedin.percentage` attribute is the percentage of the flowerbed devoted to the given plant, where a value of 1 represents 1%. To compute the *area* devoted to a particular plant in a particular bed, you can use the following formula (with the appropriate tables and join conditions, of course):

  1.0 * `flowerbeds.size` * `plantedin.percentage` / 100

  This area can then be summed for individual plants, families, flowerbeds, etc.
- Flowerbeds may have plants of any type. Not all flowerbeds are planted to 100% capacity. And, likely due to data errors, some flowerbeds are even planted to more than 100% capacity; we say that these flowerbeds have overfilled capacity.

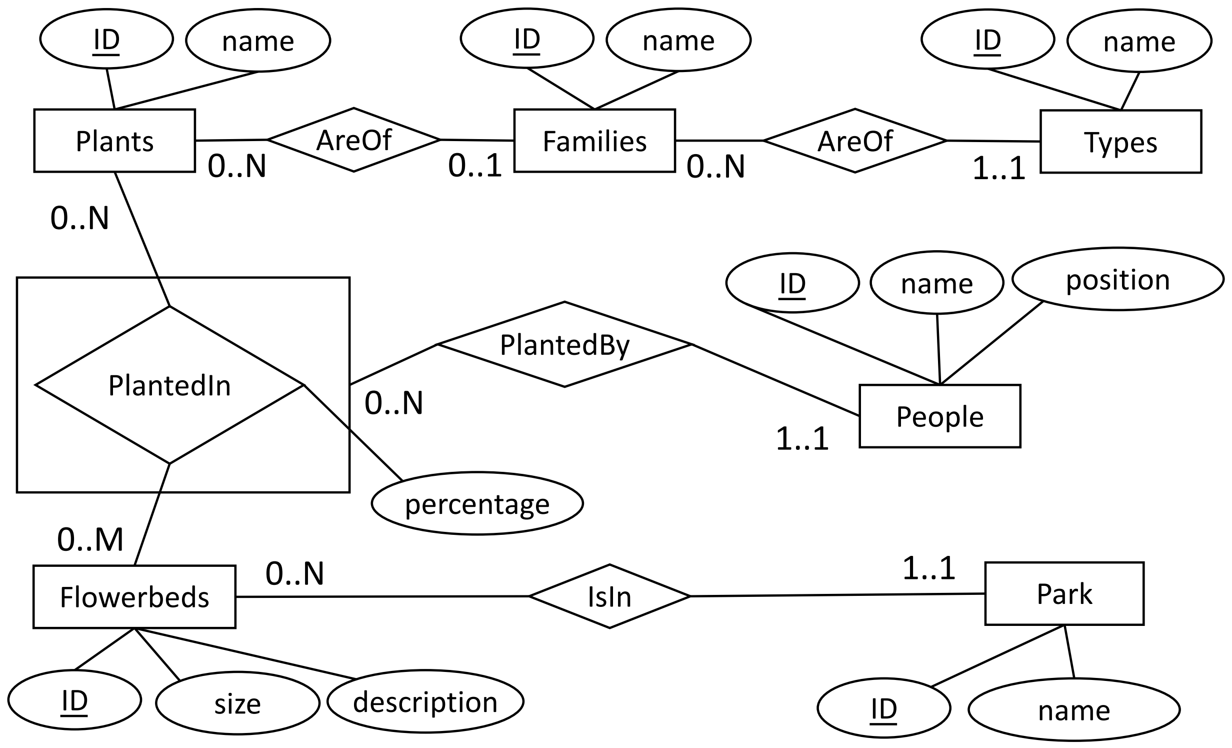

Figure 1: ER Diagram for the flowerbeds database.

Instructions for SQL Queries in Question 1

Queries must return correct results for any database instance. They should avoid system-specific 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, and it may be given partial points.

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 3.

(a) There are 8 different plants that are missing the family information. How many plants belong to the family "Thespesia"?

(b) Of the people in the database, 11 have not planted anything. How many of those, who have not planted anything, have the position "Planter"?

(c) The total area of the family "Thespesia" is 66.62 (the unit is square meters; on my machine, the exact number is 66.62000000000003). What is the total area of the family "Vicia"?

Note: The result needs only be accurate to two digits after the decimal point.

(d) The most overfilled flowerbed is planted to 105% capacity. What are the ID(s) of the flowerbed(s) with the most overfilled capacity?

Note: The output of this query could contain more than one identifier.

(e) There are 9 flowerbeds that are planted to more than 100% capacity. How many flowerbeds are planted to less than 100% capacity.

(f) How many flowerbeds are planted to less than 100% capacity, and have a plant of the type "shrub" planted in them?

(g) There are 354 families that are planted in at least one flowerbed in all the parks from the database. How many flowerbeds have at least one plant of all types from the database.

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

(h) Write a query that returns the ID and name of people, and the total area that they have planted. The list should be restricted to people who have the position "Planter" and who have planted some plant of type "flower" in the park "Kongens Have". The total area returned, however, should not have those restrictions and should represent all the area planted by these people. The list should have the largest area first.

Note: The readability of the solution is important for this query.

2 (BSc ONLY) SQL programming (5 points)

Consider the SQL trigger code in Figure 2, which aims at ensuring that flowerbeds are not planted over capacity. Then consider that the DELETE and INSERT statements in Figure 2 are executed in order on the existing database instance (where the relevant plants, flowerbeds and people actually exist).

```
CREATE FUNCTION CheckPercentage() RETURNS TRIGGER
AS $$ BEGIN
  -- Check 1: Is the percentage OK for the row?
  IF ((NEW.percentage < 0) OR (NEW.percentage > 100)) THEN
    RAISE EXCEPTION 'Percentage must be 0..100'
    USING ERRCODE = '45000';
  END IF;
  -- Check 2: Is the total percentage OK for the flowerbed?
  IF (100 < (SELECT SUM(I.percentage)
             FROM plantedin I
             WHERE I.flowerbedID = NEW.flowerbedID)) THEN
    RAISE EXCEPTION 'Total percentage must be 0..100'
    USING ERRCODE = '45000';
  END IF;
  RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER CheckPercentage
AFTER INSERT ON plantedin
FOR EACH ROW EXECUTE PROCEDURE CheckPercentage();

DELETE FROM plantedin;
INSERT INTO plantedin (plantID, flowerbedID, percentage, planterID) VALUES (1, 1,  -1, 1);
INSERT INTO plantedin (plantID, flowerbedID, percentage, planterID) VALUES (2, 1, 100, 1);
INSERT INTO plantedin (plantID, flowerbedID, percentage, planterID) VALUES (3, 1,   1, 1);
```

Figure 2: Insertion trigger `CheckPercentage` for the `plantedin` table.

Select the true statements:

(a) Check 1 can be replaced by a CHECK constraint on the `plantedin` table.

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

(c) The first INSERT statement will fail.

(d) The second INSERT statement will fail.

(e) The third INSERT statement will fail.

3 (MSc ONLY) Database programming (5 points)

Consider the Java code in Figure 3.

```java
public static void deletePlant(Connection conn, int plantId) throws SQLException {
        try (Statement st = conn.createStatement()) {
                conn.setAutoCommit(false);
                st.execute("DELETE FROM PlantedIn WHERE plantID=" + plantId);
                st.execute("DELETE FROM Plants WHERE ID=" + plantId);
                conn.commit();
        }
        catch (Exception e) {
                throw e;
        }
        finally {
                conn.setAutoCommit(true);
        }
}
```

Figure 3: Code for removing information from the flowerbeds database.

Select the true statements:

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

(b) The code is using transactions correctly.

(c) The code will only work correctly if the connection is to a PostgreSQL server.

(d) The try-with block will automatically close the statement, so calling st.close() is not necessary.

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

4 ER Diagrams and Normalization (25 points)

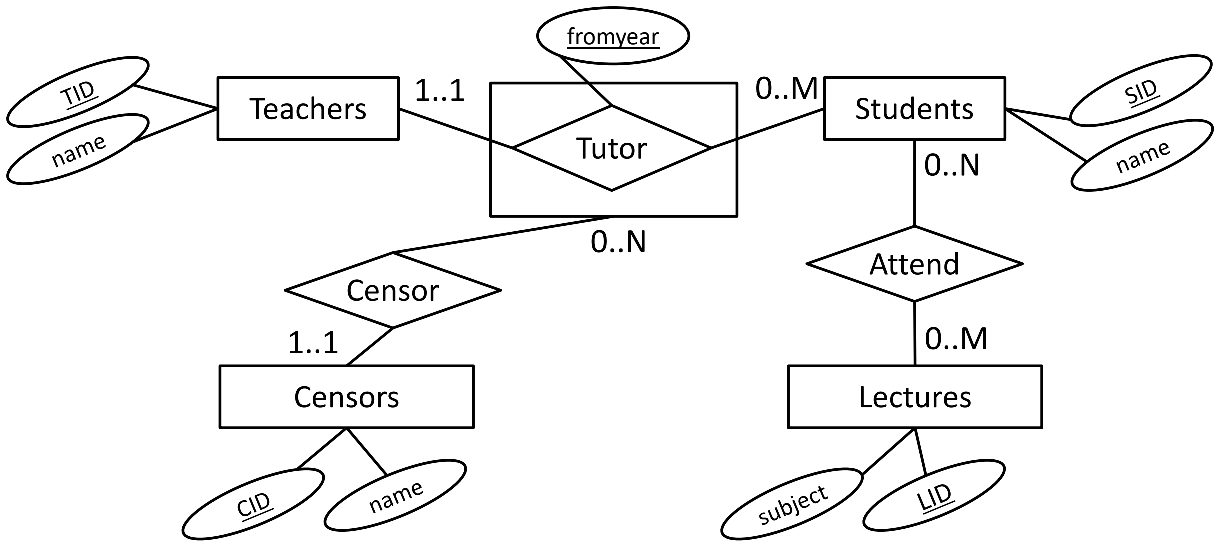

Figure 4: Abstract ER Diagram.

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

(a) Each teacher is connected through relationships to at least one lecture.

(b) Each teacher is connected through relationships to at least one censor.

(c) Each student is connected through relationships to at most one censor.

(d) Each student is connected through relationships to at least one censor.

(e) Each student may be connected multiple times to the same teacher.

(f) Each censor may be connected multiple times to the same teacher.

b) Write SQL DDL commands to create a 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 type INTEGER (including fromyear) or VARCHAR.

c) Write an ER diagram for a guru database, based on the following requirements. The diagram should clearly show the entities, relationships, participation constraints and keys described in the text. 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.

- Gurus have a unique identifier, and a name.
- Gurus are of one of the following types: mental, spiritual or practical. (For each type, there are many attributes, but you can ignore these.)
- Each guru may be mentored by one other guru. (Type is irrelevant here.)
- Monks are guided by one spiritual guru at each time. The time interval (to, from) of guidance must be recorded, and monks may be guided by the same guru more than once.
- Each guidance interval may be monitored by one practical guru.

d) Consider a table $R(L, M, N, O, P)$ with the following dependencies:

$$\begin{aligned} LM &\rightarrow NOP \\ N &\rightarrow L \\ O &\rightarrow P \end{aligned}$$

Select the true statements:

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

(b) $LM \rightarrow P$ 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 redundant dependencies).

e) Consider a table $R(L, M, N, O, P)$ with the following dependencies:

$$\begin{aligned} LM &\rightarrow NOP \\ N &\rightarrow OP \\ L &\rightarrow N \\ M &\rightarrow M \end{aligned}$$

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 very large relation with information on map data:

MapObjects(ID, objecttypeID, objectID, lat, long, radius, <many long attributes>)

The attributes have the expected types and none of them are nullable. Consider that about half the objects have radius > 1 and that there are about 50 object types. Now consider the following three SQL queries:

Query 1

```
select *
from MapObjects
where objecttypeID = 22;
```

Query 2

```
select lat, long
from MapObjects
where radius > 1;
```

Query 3

```
select objectID, count(*)
from MapObjects
group by objectID;
```

Answer each of the following questions:

(a) Indicate for each query (in isolation) whether a clustered index could be profitably defined (i.e., would be preferable to a non-clustered index or no index at all). Explain your answer and define the indexes you consider.

(b) Indicate for each query (in isolation) whether a covering index could be profitably defined (i.e., would be preferable to a clustered index). Explain your answer and define the indexes you consider.

(c) Recall that a relation can only have one clustered index. Considering all three queries, which clustered index would you define on the relation? Explain your answer.

6 DBMS Architecture (10 points)

a) Consider a database server with persistent main memory and no secondary storage. Define the STEAL buffer management policy and discuss its usefulness for such a server.

b) Select the true statements:

(a) Writing the log is usually slower than writing the actual updates to relations.

(b) SSD storage is volatile.

(c) The FORCE policy is difficult to implement in reality, since committing large transactions can cause many disk writes, and the server might crash in the middle of the writing process.

(d) Query optimisation are often improved by gathering statistics over relations.

7 Transactions (10 points)

a) Briefly discuss (a) what consistency means in the context of ACID transactions, and (b) how relational systems support such consistency.

b) Select the true statements:

(a) Locking can be used to implement isolation.

(b) A buffer manager with a NO STEAL policy complicates the implementation of atomicity.

(c) Isolation and durability are mutually exclusive.

(d) Transactions are a general concept that could potentially be applied to, for example, file systems.