Introduction to Database Systems
MSc and BSc Final Exams

Björn Thór Jónsson / Philippe Bonnet / Andrea Vezzosi

June 1, 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 *Final Exam June 2021*.

Database Description for Questions 1–3

In this exam you will work with a (poorly designed!) database of football results, for a fictional future "European Super League" yet for some reason with past football stars. To start working with the database, run the commands in `idb-june-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:

```
cities(ID, name)
clubs(ID, name, cityID)
seasons(ID, startdate, enddate)
matches(homeID, awayID, seasonID, homegoals, awaygoals, awaywin)
players(ID, name)
signedwith(playerID, clubID, seasonID)
```

Primary keys and foreign keys are (almost always) 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:

- A club had a home match, if its ID is in *matches.homeID*, and an away match, if its ID is in *matches.awayID*. The home team scored *matches.homegoals* and the away team *matches.awaygoals*. The (redundant) boolean attribute *matches.awaywin* is true if *matches.awaygoals* > *matches.homegoals*.
- Each club receives 3 points for each victory (they score more goals than the opponent) and 1 point for a tie (both teams score equally many goals).
- The foreign key for *signedwith.clubID* is missing, which means that there are some entries in the *signedwith* table that point to non-existing clubs.
- Since the players in the *players* table are all very famous, we consider that if they are signed with a team during a season, then (a) they played all the matches of the club, and (b) they were *involved with* all the goals of the club.

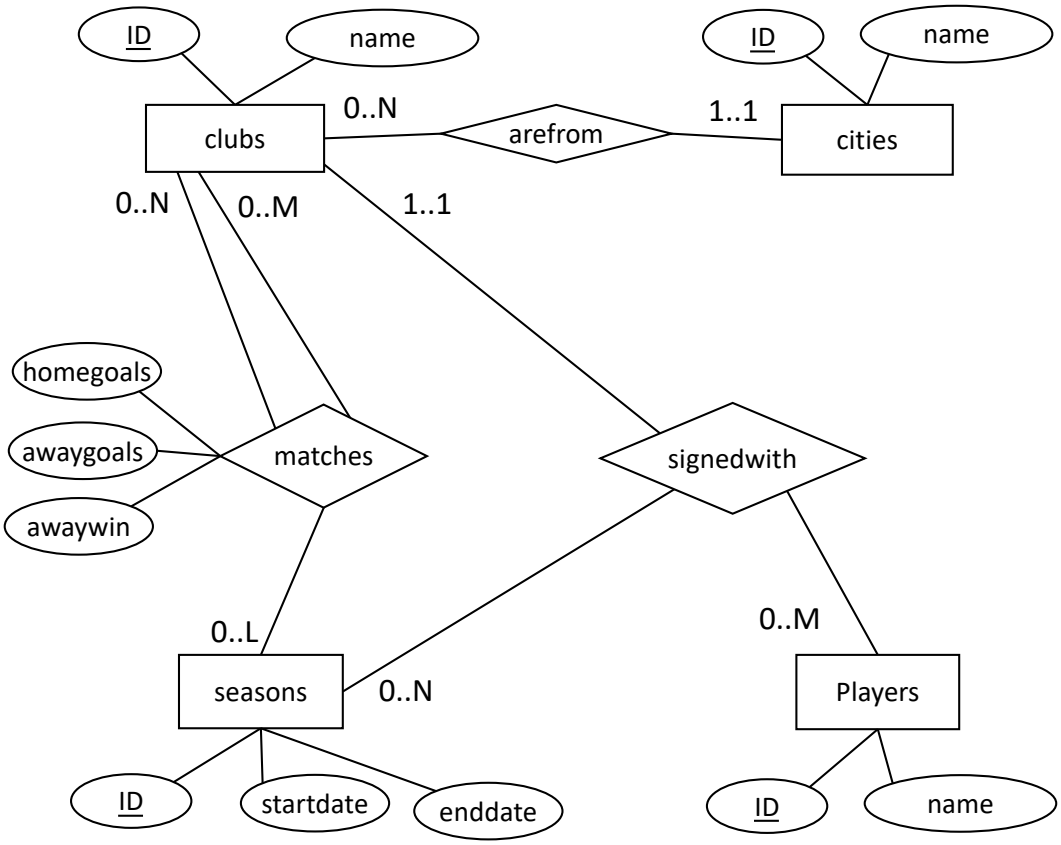

Figure 1: ER Diagram for the football database.

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, 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) The database has one club from the city named Copenhagen. How many clubs are from the city named London?

(b) In the *signedwith* table, there are three different *clubID* values that no longer exist in the *clubs* table. How many players signed with those clubs?

(c) The club named Liverpool has a total of 243 away wins, while the highest number of away wins by any club happens to be 260. How many different clubs jointly have the most away wins?

(d) During the playing career of Andrea Pirlo, he was *involved with* 319 home goals. As outlined above, this means that while he was signed with different clubs, they scored a total of 319 home goals. How many away goals was Steven Gerrard *involved with*?

(e) During his illustrious playing career, Bjorn signed with 7 different clubs. Write a query to output the name(s) of the player(s) who signed with the largest number of different clubs.

Note: The output of this query is a set of one or more player names.

(f) How many players never signed with a club from the city named London?

(g) London clubs are defined here as clubs from the city named London. All 14 non-London clubs have beaten all London clubs away (meaning that the London club was the home team) during some season registered in the database. How many non-London clubs have beaten all London clubs away *in a single season*?

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

(h) Given the points rule in the database description, write a query that correctly outputs the final standings (team names and total points, in descending order of total points) for the season with ID = 2035. The figure below shows the first five lines of the final standings for the season with ID = 2044.

Note: The output of this query contains 20 rows of text. Do not worry about formatting.

	name character varying	points bigint
1	West Ham	72
2	Bayern	71
3	Everton	64
4	Juventus	61
5	Arsenal	60

2 (BSc ONLY) SQL programming (5 points)

Consider the SQL trigger code in Figure 2, which aims at ensuring the correctness of attributes for the `matches` table.

```
CREATE FUNCTION CheckGoals() RETURNS TRIGGER
AS $$ BEGIN
  -- Check 1: Are the goals negative?
  IF ((NEW.homegoals < 0) OR (NEW.awaygoals < 0)) THEN
    RAISE EXCEPTION 'Goals must be 0 or higher'
    USING ERRCODE = '45000';
  END IF;
  -- Action 2: Set the awaywin attribute correctly
  IF ((NEW.awaygoals > NEW.homegoals) AND NOT (NEW.awaywin)) THEN
    NEW.awaywin = true;
  END IF;
  RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER CheckGoals
AFTER INSERT ON matches
FOR EACH ROW EXECUTE PROCEDURE CheckGoals();

DELETE FROM matches WHERE seasonID = 2070;
INSERT INTO matches (homeID, awayID, seasonID, homegoals, awaygoals, awaywin)
VALUES (1, 2, 2070, 1, -1, false);
INSERT INTO matches (homeID, awayID, seasonID, homegoals, awaygoals, awaywin)
VALUES (1, 3, 2070, 1, 0, false);
INSERT INTO matches (homeID, awayID, seasonID, homegoals, awaygoals, awaywin)
VALUES (1, 4, 2070, 0, 1, false);
```

Figure 2: Insertion trigger `CheckGoals` for the `matches` table.

Consider that the DELETE and INSERT statements in Figure 2 are executed in order, on the existing database instance. Select the true statements below.

(a) Check 1 can be replaced by CHECK constraints on the `matches` table.

(b) This trigger is correctly written as an AFTER trigger.

(c) The first INSERT statement will fail.

(d) The second INSERT statement will succeed.

(e) The third INSERT statement will succeed, and the inserted row will correctly have the *awaywin* attribute changed to *true*.

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<Match> getMatchesByClubID(
                Connection conn, int clubID, int season) throws SQLException {
    PreparedStatement st = conn.prepareStatement(
            "SELECT * FROM matches WHERE (homeID = ? OR awayID = ?) AND season = ?;");
    st.setInt(1, clubID);
    st.setInt(2, clubID);
    st.setInt(3, season);
    ResultSet rs = st.executeQuery();

    List<Match> matches = new ArrayList<>();
    while (rs.next()) {
            matches.add(new Match(
                    rs.getInt("homeID"),
                    rs.getInt("awayID"),
                    rs.getInt("season"),
                    rs.getBoolean("awaywin"),
                    rs.getInt("homegoals"),
                    rs.getInt("awaygoals")
            ));
    }

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

Figure 3: Code for retrieving information from the `matches` relation.

Select the true statements:

(a) The use of PreparedStatement is necessary to make the code safe from SQL injection attacks.

(b) Both inputs and outputs of the getMatchesByClubID method make use of Object Relational Mapping.

(c) The call to rs.close() is redundant, since the ResultSet will be closed by st.close().

(d) The resources associated with the statement are not always released by this code.

(e) The query string should include a "COMMIT;" at the end to close the transaction.

4 ER Diagrams and Normalization (25 points)

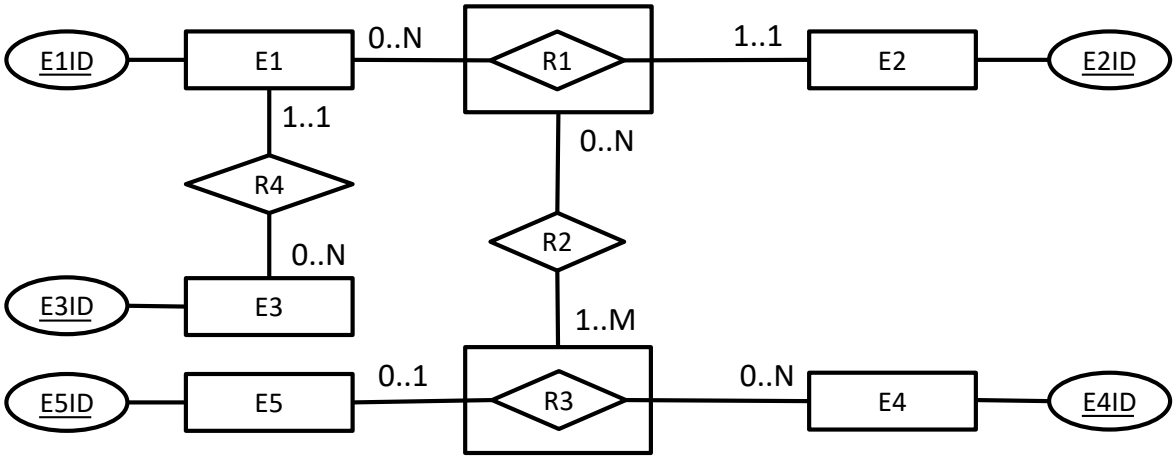

Figure 4: Abstract ER Diagram.

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

(a) Each E1 entity is connected through relationships to at least one E5 entity.

(b) Each E1 entity is connected through relationships to at least one E2 entity.

(c) Each E3 entity is connected through relationships to at most one E5 entity.

(d) Each E3 entity is connected through relationships to at most one E2 entity.

(e) E3 is a weak entity.

(f) The number of columns in the table for R2 will depend on whether option 1 or option 2 is chosen for implementing aggregation.

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.

c) Write an ER diagram for a farm 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.

- Each farm animal is one of pig, cow or chicken. All animals have weight, cows have yield (i.e., amount of milk), and pigs have a rating.
- Each plot of land has a (unique) identifer and GPS location (for simplicity, model this as a single attribute).
- Different grains are identified by their name.
- A plot is dedicated to exactly one grain every year; this grain varies across years.
- Each pig must eat some of the grains.
- Workers are hired to work on one specific plot of land.

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

$$\begin{aligned} L &\rightarrow N \\ N &\rightarrow P \\ M &\rightarrow O \end{aligned} \tag{1}$$

Select the true statements:

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

(b) $L \rightarrow P$ is a redundant 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 redundant dependencies).

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

$$\begin{aligned} NO &\rightarrow PLM \\ M &\rightarrow M \\ N &\rightarrow M \\ O &\rightarrow PL \end{aligned} \tag{2}$$

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 relation with information on webstore orders:

OrderLines(ID, itemID, customerID, date, time, price, <many long attributes>)

The attributes have the expected types and none of them are nullable. Now consider the following three SQL queries:

Query 1

```
select customerID, sum(price)
from OrderLines
group by customerID;
```

Query 2

```
select max(date)
from OrderLines
where customerID = 2200;
```

Query 3

```
select *
from OrderLines
where itemID = 1234;
```

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) Argue why a buffer manager is needed in an HDD-based DBMS.

b) What is the query optimizer component responsible for? Select all statements that apply:

(a) Generating a query execution plan.

(b) Parsing queries.

(c) Comparing access methods.

(d) Maintaining statistics over existing relations.

7 Transactions (10 points)

a) Argue why the FORCE buffer manager policy is not appropriate for an HDD-based DBMS.

b) In the WAL protocol, when are log buffers flushed to disk? Select all statements that apply:

(a) Before a transaction commits.

(b) Before dirty data is stolen (i.e., written to disk at any time before the transaction that modified it commits).

(c) After dirty data is stolen.

(d) After a transaction commits.

Final Exam June 2021

1. **0) Declaration**

   0) Upload your declaration of independent work.

   Notes: (not included in XML)

   - `Not included`

2. **1a) SQL**

   1a) Write your SQL query here:

   Notes: (not included in XML)

   - `See idb-june-2021-SQL.sql`

3. **1a) Query answer**

   1a) Run the query of the previous question and paste the result here (an integer):

   - 6 ✓

4. **1b) SQL**

   1b) Write your SQL query here:

   Notes: (not included in XML)

   - `See idb-june-2021-SQL.sql`

5. **1b) Query answer**

   1b) Run the query of the previous question and paste the result here (an integer):

   - 7 ✓

6. **1c) SQL**

   1c) Write your SQL query here:

   Notes: (not included in XML)

   - `See idb-june-2021-SQL.sql`

7. **1c) Query answer**

   1c) Run the query of the previous question and paste the result here (an identifier):

   - 2 ✓

8. **1d) SQL**

   1d) Write your SQL query here:

   Notes: (not included in XML)

- `See idb-june-2021-SQL.sql`

9. **1d) Query answer**

   1d) Run the query of the previous question and paste the result here (an integer):

   - 62 ✓

10. **1e) SQL**

    1e) Write your SQL query here:

    Notes: (not included in XML)

    - `See idb-june-2021-SQL.sql`

11. **1e) Query answer**

    1e) Run the query of the previous question and paste the result here (a string):

    - Ruud van Nistelrooy ✓

12. **1f) SQL**

    1f) Write your SQL query here:

    Notes: (not included in XML)

    - `See idb-june-2021-SQL.sql`

13. **1f) Query answer**

    1f) Run the query of the previous question and paste the result here (an integer):

    - 22 ✓

14. **1g) SQL**

    1g) Write your SQL query here:

    Notes: (not included in XML)

    - `See idb-june-2021-SQL.sql`

15. **1g) Query answer**

    1g) Run the query of the previous question and paste the result here (an integer):

    - 2 ✓

16. **1h) SQL**

    1h) Write your SQL query here:

    Notes: (not included in XML)

- See `idb-june-2021-SQL.sql`

17. **1h) Query answer**

    1h) Run the query of the previous question and paste the result here (text):

    Notes: (not included in XML)

    - `Chelsea 63 West Ham 62 Fulham 62 FCK 61 AC 60 City 57 Juventus 56 Liverpool 56 Roma 55 PSG 54 Inter 54 Rangers 53 Barcelona 53 Crystal Palace 48 Everton 48 Tottenham 45 Arsenal 44 United 43 Bayern 41 Real 38`

18. **2) [BSc only] SQL programming**

    2) [BSc only] Select the true statements:

    (a) Check 1 can be replaced by CHECK constraints on the `matches` table. (33.33333%)
    (b) This trigger is correctly written as an AFTER trigger. (0%)
    (c) The first INSERT statement will fail. (33.33333%)
    (d) The second INSERT statement will succeed. (33.33333%)
    (e) The third INSERT statement will succeed, and the inserted row will correctly have the *awaywin* attribute changed to *true*. (0%)

19. **3) [MSc only] Java programming**

    3) [MSc only] Select the true statements:

    (a) The use of PreparedStatement is necessary to make the code safe from SQL injection attacks. (0%)
    (b) Both inputs and outputs of the getMatchesByClubID method make use of Object Relational Mapping. (0%)
    (c) The call to rs.close() is redundant, since the ResultSet will be closed by st.close(). (50%)
    (d) The resources associated with the statement are not always released by this code. (50%)
    (e) The query string should include a "COMMIT;" at the end to close the transaction. (0%)

    Note: The ER diagram was somewhat more difficult to interpret than in previous semesters. The grading of 4a and 4b was adjusted to compensate.

20. **4a) ER Diagram Interpretation**

    4a) Select the true statements:

    (a) Each E1 entity is connected through relationships to at least one E5 entity. (33.33333%)
    (b) Each E1 entity is connected through relationships to at least one E2 entity. (33.33333%)

(c) Each E3 entity is connected through relationships to at most one E5 entity. (0%)
(d) Each E3 entity is connected through relationships to at most one E2 entity. (33.33333%)
(e) E3 is a weak entity. (0%)
(f) The number of columns in the table for R2 will depend on whether option 1 or option 2 is chosen for implementing aggregation. (0%)

21. **4b) SQL DDL**

    4b) Write your DDL for creating the database. You can also write any extra assumptions, attributes or explanations you feel are necessary.

    Notes: (not included in XML)

    - `See idb-march-2021-DDL.sql.`

22. **4c) ER Diagram Creation**

    4c) Upload the ER diagram or deliver a hand drawing at the exam.

    Notes: (not included in XML)

    - `See idb-march-2021-ER.png.`

23. **4d) Normalisation**

    4d) Select the true statements:

    (a) LN is the only (candidate) key of R. (0%)
    (b) $L \rightarrow P$ is a redundant functional dependency. (50%)
    (c) Normalizing to 3NF/BCNF results in exactly two relations. (0%)
    (d) The relation can be normalized to BCNF without losing functional dependencies (excluding trivial, unavoidable, and redundant dependencies). (50%)

24. **4e) Normalisation**

    4e) Write down the normalized relations. For each resulting relation, write its columns and clearly indicate whether it is in BCNF.

    Notes: (not included in XML)

    - `NO in BCNF (key = NO), NM in BCNF (key = N), OPL in BCNF (key = O).`

25. **5a) Clustering Indexes**

    5a) Argue for a clustered index, compared to unclustered or no index.

    Notes: (not included in XML)

- (Q1) To run the GROUP BY, a system would typically sort the relation. An index can be used to read the relation in a sorted order. A clustered index on (customerID) would perform better than an unclustered index or no index. Since an unclustered index on (customerID, price) would be covering, then a clustered index on (customerID, price) would not be better.
- (Q2) A clustered index on (customerID) would perform better than an unclustered index (unless each customer has one or very few orders) and no index (this is always worse). Since an unclustered index (customerID, date) would be covering, then a clustered index on (customerID, date) would not be better.
- (Q3) A clustered index on (itemID) would be optimal for this query, since (a) it outperforms no index, (b) it outperforms an unclustered index, and (c) covering indexes don't really apply.

26. **5b) Covering Indexes**

    5b) Argue for a covering index for each query, compared to clustered index.

    Notes: (not included in XML)

    - (Q1) A covering index on (customerID, price) would perform optimally, by reading the minimal number of columns, while avoiding a sort of the relation.
    - (Q2) Likewise, a covering index on (customerID, date) would be optimal, as it reads fewer columns than a clustered index.
    - (Q3) A covering index does not apply here, since all attributes are returned.

27. **5c) Best Clustered Index**

    5c) Considering all three queries, explain which clustered index would you define.

    Notes: (not included in XML)

    - Since covering indexes can be used effectively for queries Q1 and Q2, but not for Q3, the best clustered index for these three queries would be on (itemID).

28. **6a) DBMS Architecture**

    6a) Write your reflections here:

    Notes: (not included in XML)

    - The crux of the answer should be: Data must be in memory before it is processed. Data is transferred in fixed size units between disk and RAM. Buffering avoids repeating expensive I/Os. Partial points

were awarded for discussing high cost of IOs, strategies for reducing IO cost, managing updates and maintaining transactional properties.

29. **6b) DBMS Architecture**

    6b) Select the true statements:

    (a) Generating a query execution plan. (50%)
    (b) Parsing queries. (0%)
    (c) Comparing access methods. (50%)
    (d) Maintaining statistics over existing relations. (0%)

30. **7a) Transactions**

    7a) Write your definition here:

    Notes: (not included in XML)

    - The crux of your answer should be: Writing all the updated pages would be very expensive (and cannot be done atomically). But with WAL we don't have to, as sequential writes of log buffer are orders of magnitude cheaper than random writes of data. Some side points were given for good related points.

31. **7b) Transactions**

    7b) Select the true statements:

    (a) Before a transaction commits. (50%)
    (b) Before dirty data is stolen (i.e., written to disk at any time before the transaction that modified it commits). (50%)
    (c) After dirty data is stolen. (0%)
    (d) After a transaction commits. (0%)