# CS2102-2310: Database Systems
# **Midterm**

Date: 03 October 2023, Time: 12:30–13:30

## Submission Instructions

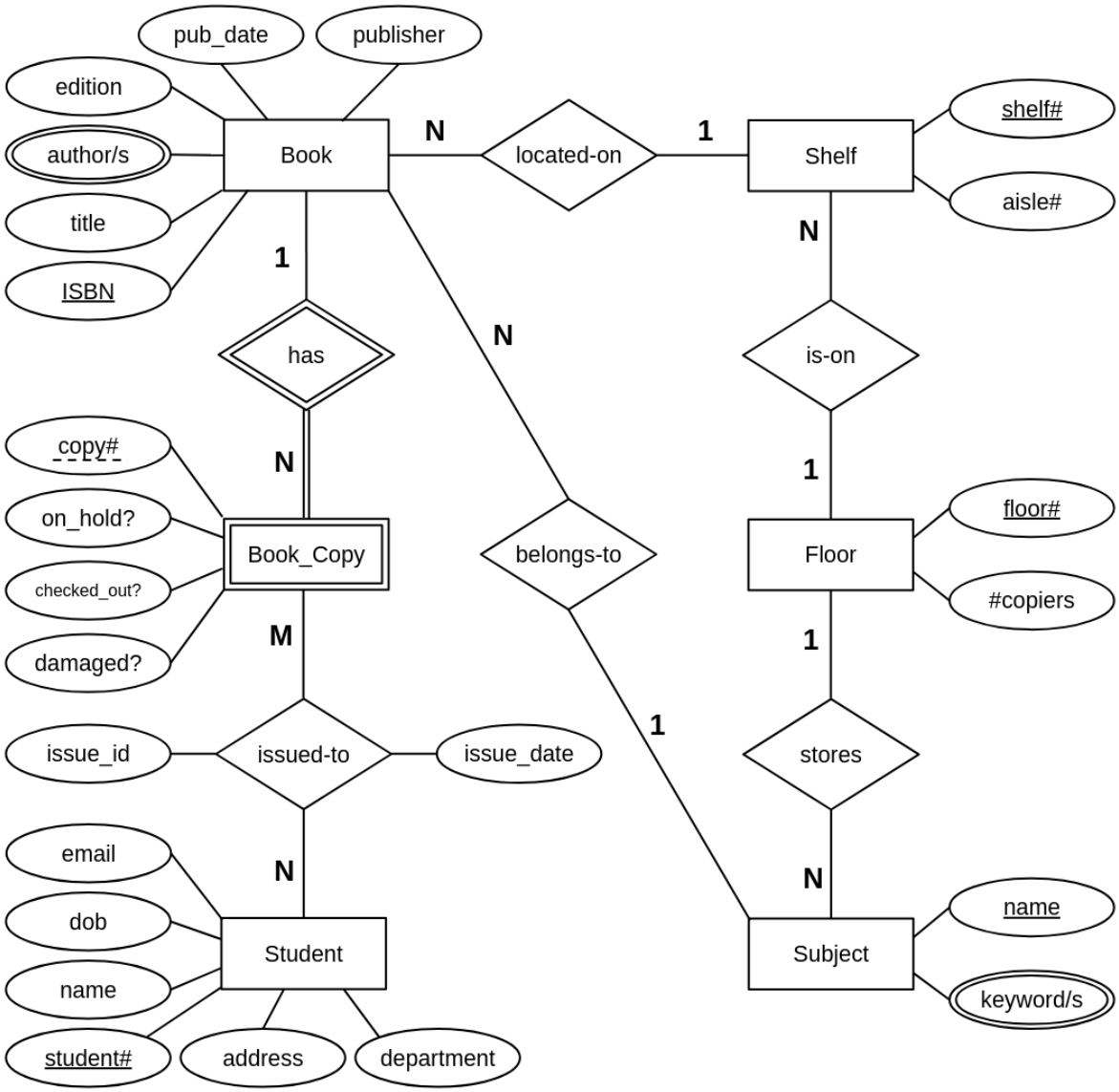
1. Please read **ALL** instructions carefully.
2. All the assessment is be done using Examplify; the assessment contains:
   (a) MCQ/MRQ: Questions 1–8
   (b) SQL Queries: Questions 9–17 (note: Q11-15 relate to the same query!)
3. This is an open-book assessment.
4. Your Internet connection will be blocked for the duration of the assessment.
5. This assessment starts at 12:30 and ends at 13:30.
   - Submit your answers by 13:30
   - No additional time will be given to submit.
6. For the MCQ/MRQ questions: Please note that the order of the answers might differ from the order in the PDF due to randomization!
7. For the SQL questions, there are additional instructions below; in a nutshell:
   - Use an IDE or text editor to write your queries and test them using `psql` or pgAdmin.
   - Prepare your final answer before submitting it to Examplify according to the instructions in the SQL Part.
   - Allocate sufficient time to prepare your final answers and to copy-&-paste them into Examplify.
8. Failure to follow each of the instructions above may result in deduction of your marks.

## Good Luck!

# ER Model (*8 Marks*)

The ERD represents a physical library where multiple floors (storeys) of the library store books on multiple subjects. The floor contains shelves; each shelf bears a unique shelf# on which the books are shelved. Each book with all its copies is assigned to a specific physical shelf on a specific floor. Students are issued (loaned) copies of books.

With the above context please answer the following questions. Note that in the mapping, we may have used attribute names that do not exactly match the attribute names from ERD. That is simply done for better readability.



## MCQ/MRQ: ER Diagram (4 Marks)

**Q1:** How many multi-valued attributes are present in the schema? (1 Mark)

- ◯ Zero
- ◯ One
- √ **Two**
- ◯ Cannot determine

**Q2:** The complete identifier for `Book_Copy` according to the ERD is... (1 Mark)

- ◯ copy#
- √ **(copy#, ISBN)**
- ◯ ISBN
- ◯ None of the above

**Q3:** Which of the following statement(s) is/are **TRUE** about the relationship types in this ERD? Select all that apply! (1 Mark)

- √ **There is only one relationship type that has its own relationship attributes.**
- √ **There is only one "identifying relationship" in this schema.**
- ◯ The `FLOOR` entity type participates in the maximum number of relationship types among all entity types.
- ◯ All of the above.

**Q4:** Ignoring the identifying relationship, which of the following statement(s) is/are **TRUE**? Select all that apply! (1 Mark)

- ◯ There are three one-to-many relationship types.
- √ **There are four one-to-many relationship types.**
- √ **There is only one many-to-many relationship type.**
- ◯ There is one ternary relationship type.

## MCQ/MRQ: Mapping ERD to Schema (4 Marks)

**Q5:** When the `Issued-to` relationship type is mapped to the relational model, the primary key of the resulting relation (table) will be: (1 Mark)

- ◯ `(student#, copy#)`
- √ `(student#, ISBN, copy#)`
- ◯ `(student#, ISBN)`
- ◯ `(student#, ISBN, copy#, author)`

**Q6:** When the `Book` entity type is mapped to the relational model, incorporation of the following attributes as foreign keys into the `Book` table is appropriate: (1 Mark)

- √ `shelf#, subject_name`
- ◯ `shelf#`, `copy#`, `subject_name`
- ◯ `shelf#`, `floor#`, `subject_name`
- ◯ `shelf#`, `subject_name`, `author`

**Q7:** When the `Subject` entity type is mapped to a table, which of the following statement(s) about this table is/are **TRUE**? Select all that apply! (1 Mark)

- ◯ It does not need any foreign keys in it.
- √ **It must have a foreign key `floor#`.**
- ◯ It must have a foreign key `book_ISBN`.
- ◯ It must have both foreign keys `floor#` and `book_ISBN`.

**Q8:** What can we say about handling the multi-valued attribute `keywords` of the `Subject` entity type during mapping? Mark all correct statements! (1 Mark)

- ◯ It can be included in the `Subject` table as an attribute.
- ◯ It becomes a part of the primary key in the `Subject` table.
- √ **It must be dealt with by setting up a new table besides the `Subject` table.**
- √ **The new table `Subject_Keyword` contains (`subject_name`, `keyword`) and that combination is a composite primary key of this table.**

# SQL Queries (*12 Marks*)

In this part of the exam your task is to formulate queries in SQL. We use the same database that we introduced in Assignment 1. This means you can directly run and test your queries using `psql`. If needed, we provide the ER diagram and the `CREATE TABLE` statements for the database schema in the Appendix.

**Instructions.** To ensure successful submission and grading of your SQL queries adhere to the following instructions:

- For each question, you are to write a **single SELECT SQL statement** to answer the question.
- You are only allowed to use SQL constructs that have been covered in the lectures (e.g., CTEs are not allowed).
- Each answer must be a syntactically valid SQL query and executable on PostgreSQL. Test with `psql` or pgAdmin!
- Each question must be answered independently of any other questions.
- You must not enter any extraneous text for your answer (e.g., "My answer is: ...), or enter multiple answers (i.e., submit multiple SQL statements). Your answer should not contain any comments.
- Note: The exact names of the columns in the result do not matter!

**Recommendations.** We recommend the following steps to answer each question:

- Write the answer query using your IDE or text editor of choice, and test it on your machine with `psql` or pgAdmin.
- Once you are happy with your query, copy the `SELECT` statement for the question into the corresponding answer field in Examplify.

**Q9:** What was the total length in kilometers of the Tour de France 2023 up to and including Stage 10? All lengths and distances in the database are recorded in kilometers. (1 Mark)

**Solution:**

```
SELECT SUM(s.length) AS total_length
FROM stages s
WHERE s.nr <= 10;
```

**Q10:** Which locations were both a start and a finish of a stage? Return the name and the country for each location! (2 Marks)

**Solution:**

```
SELECT l.name, l.country
FROM (SELECT start AS location
      FROM stages INTERSECT
      SELECT finish AS location
      FROM stages) s, locations l
WHERE s.location = l.name;
```

or

```
SELECT s1.start, l1.country
FROM stages s1, locations l1, stages s2
WHERE s1.start = s2.finish
AND s1.start = l1.name;
```

This query is for questions **11**–**15**: Return the bib and the name of the different riders who abandoned the 2023 Tour de France at Stage 1, i.e., the riders who did not even finish Stage 1.

**Q11:** Use one of the five different SQL constructs that we have discussed in the lectures for this type of query! For the following questions **12**–**15**, use another of the five different SQL constructs that we have discussed in the lectures for this type of query. (1 Mark)

**Solution:**

```
SELECT r.bib, r.name
FROM riders r LEFT OUTER JOIN results_individual i ON  r.bib = i.
   rider
WHERE i.rider IS NULL;
```

or

```
SELECT r.bib, r.name
FROM riders r LEFT OUTER JOIN results_individual i ON  r.bib = i.
   rider
GROUP BY r.bib, r.name
HAVING COUNT(i.rider)=0;
```

**Q12:** Use any of the five different SQL constructs which you have not used so far to answer the original query! (0.5 Mark)

**Solution:**

```
SELECT r.bib, r.name
FROM riders r
EXCEPT
SELECT i.rider, r1.name
FROM riders r1, results_individual i
WHERE r1.bib = i.rider;
```

**Q13:** Use any of the five different SQL constructs which you have not used so far to answer the original query! (0.5 Mark)

**Solution:**

```
SELECT r.bib, r.name
FROM riders r
WHERE r.bib NOT IN (SELECT rider i
FROM results_individual i);
```

**Q14:** Use any of the five different SQL constructs which you have not used so far to answer the original query! (0.5 Mark)

**Solution:**

```
SELECT r.bib, r.name
FROM riders r
WHERE r.bib <> ALL (SELECT i.rider
FROM results_individual i);
```

**Q15:** Use any of the five different SQL constructs which you have not used so far to answer the original query! (0.5 Mark)

**Solution:**

```
SELECT r.bib, r.name
FROM riders r
WHERE NOT EXISTS (SELECT 1
FROM results_individual i
WHERE i.rider = r.bib);
```

**Q16:** Which riders had to abandon the Tour before it finished? Return the name of the different riders who abandoned the Tour and the stage number at which they abandoned it (i.e. the first stage for which there is no individual result for a rider). Sort the result from the earlier to the later stages, and in alphabetic order within a stage! (3 Marks)

**Solution:**

```
SELECT r.name, coalesce(max(i.stage), 0)+1 AS stage_nr
FROM riders r LEFT OUTER JOIN  results_individual i
ON r.bib = i.rider
GROUP BY r.bib, r.name
HAVING COUNT(*) <> ALL (SELECT COUNT(*) FROM stages)
ORDER BY stage_nr ASC, r.name ASC ;
```

`GROUP BY r.bib` would also work in PostgreSQL.

**Q17:** For each stage, list the number of mountains of Category 3! If a stage does not have such a mountain, the result should list 0 for this stage. Return the result ordered by increasing stage number and decreasing number of mountains of Category 3! (3 Marks)

**Solution:**

```
SELECT stage_nr, num_cat3
FROM
(SELECT s.nr AS stage_nr, coalesce(cat3.count, 0) AS num_cat3
 FROM stages s LEFT OUTER JOIN
 (SELECT stage, COUNT(*)
  FROM mountains
```

```
  WHERE CATEGORY = '3'
  GROUP BY stage) cat3
 ON (s.nr = cat3.stage)
) tab
ORDER BY stage_nr ASC, num_cat3 DESC ;
```

or

```
SELECT s.nr, COUNT(m.stage) AS num_cat3
FROM stages s LEFT OUTER JOIN mountains m ON m.stage = s.nr AND
   CATEGORY = '3'
GROUP BY s.nr
ORDER BY s.nr ASC, num_cat3 DESC ;
```

# Appendix

## Database Schema for SQL Queries

For this test, we use the same database as for Assignment 1. You should therefore be already familiar with it. For convenience, we include the schema of the database.

- regions(name)
- subregions(name, region)
- countries(code, name, subregion)
- teams(name, country)
- riders(bib, name, dob, country, team)
- locations(name, country)
- stages(nr, day, start, finish, length, type)
- sprints(stage, location, distance)
- mountains(stage, location, distance, height, length, percent, category)
- results_individual(stage, rank, rider, time, bonus, penalty)
- results_sprints(stage, location, rank, rider, points)
- results_mountains(stage, location, rank, rider, points)
- results_combative(stage, rider)