University of Basel

Prof. Dr. Heiko Schuldt
David Lengweiler, M. Sc.
Martin Vahlensieck, M. Sc.

Introduction to Databases Autumn 2023

Exercise 2

(Practical) Hand-in: 20.10.2023 (during Exercise)

(Theory) Hand-in: 22.10.2023 (ADAM, 23:59)

Solving the Exercises: The exercises can be solved in small groups of a maximum of two people. Use the notations introduced in the lecture. The DMI plagiarism guidelines apply for this lecture.

Submission Information: Please upload all deliverables BEFORE the deadline to ADAM as **a single PDF** using the team hand-in feature. Solutions that are handed in too late cannot be considered. For practical exercises upload the deliverables to ADAM and present them to one of the assistants/tutors during the exercises, both is required to recieve the points!

Task (Practical) 1: Connecting to a Database (3 points)

This week you will do the same SQL query as last week — but this time with a programming language. We will show you an example in Python, R and Java. You are of course free to use any other language as well, but you will be mostly on your own if you do that.

For each of the languages we show first the required dependencies, then how to connect to the database and how to execute and retrieve the result of `SELECT version()`. You will then do the same with the analysis statement from last week and print the result. In addition you will have to compute and print the sum of all the wins by white.

Python

Create a virtual environment and install the `psycopg2` package.

```
python3 -m venv venv
. venv/bin/activate // Linux, macOS
.\venv\Scripts\activate // Windows
pip install psycopg2-binary
```

Now to connect to the database do:

```
import psycopg2

con = psycopg2.connect(host="localhost", user="demo",
    password="demo", port=54321)
```

To execute a statement, you need a cursor first:

```
cur = con.cursor()
```

Then use it to execute a statement:

```
cur.execute("SELECT version()")
print(cur.fetchone())
```

You can iterate over all results in a query using '`for row in cur`'.

R

Install the `RPostgres` package and load the required libraries.

```
install.packages("RPostgres")
library(DBI)
library(RPostgres)
```

Now you can connect to the database and execute a query:

```
con <- dbConnect(RPostgres::Postgres(), "demo", "localhost", 54321,
                     "demo", "demo")
res <- dbGetQuery(con, "SELECT version()")
print(res)
```

Java

Use your favorite build tool to create a new project and add the JDBC driver as dependency:

```
// Maven
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.6.0</version>
</dependency>
// Gradle
implementation "org.postgresql:postgresql:42.6.0"
```

You can use the following code to connect and execute a statement:

```
Connection con = DriverManager.getConnection(
    "jdbc:postgresql://localhost:54321/demo", "demo", "demo");
Statement statement = con.createStatement();
ResultSet res = statement.executeQuery("SELECT version()");
res.next();
System.out.println(res.getString(1));
```

Hand-In: Show your code, the printed results and the computed sum to an assisstant/-tutor.

Task (Theory) 2: ER University (4 points)

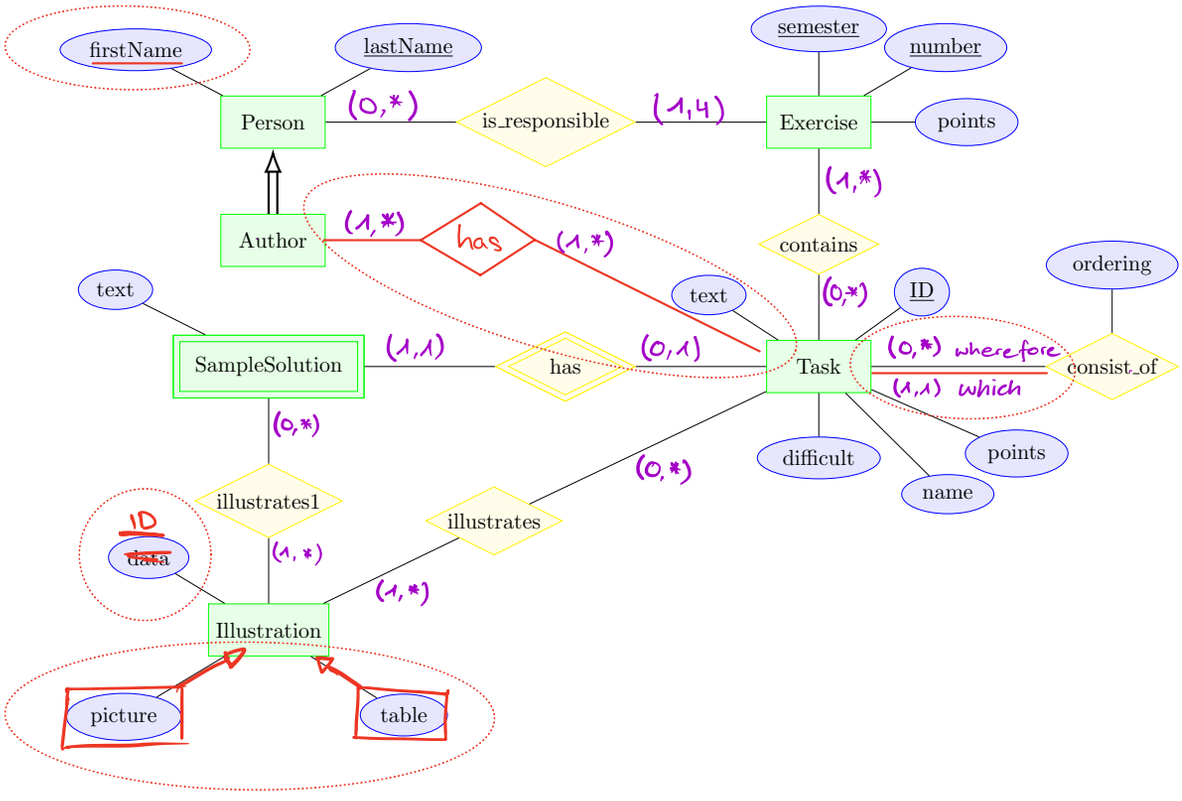
For this exercise you will find errors in an entity-relationship (ER) model for a provided scenario, which is as follows:

You are tutor of the lecture *Introduction to Databases* and you have a large pool of theoretical and practical tasks. You want to administer the tasks in a database to be able to create the exercises for the current term more simply and faster.

The requirements for the database are the following:

> Tasks have an identifier (ID) and one or several authors who are identified by their first and last name. In addition, the tasks contain a text and possibly also pictures and / or tables. Furthermore, a task possesses a name, a certain degree of difficulty and an amount of points assigned to it. A task can consist of several subtasks whereby also the sequence of the subtasks has to be specified. A subtask is also a task which can contain by itself further subtasks. The points of a task correspond to the sum of the points of the subtasks. To each task a sample solution may exist which again consists of text and possibly pictures and / or tables.
>
> Exercises should be arranged from the task pool. An exercise has a clear identification by semester and sequence number; one to four people are responsible ~~for it~~. Furthermore, the exercise has a total score which is calculated by the sum of the points of its individual tasks.

You came up with the following ER, which is missing cardinalities and includes 5 mistakes.



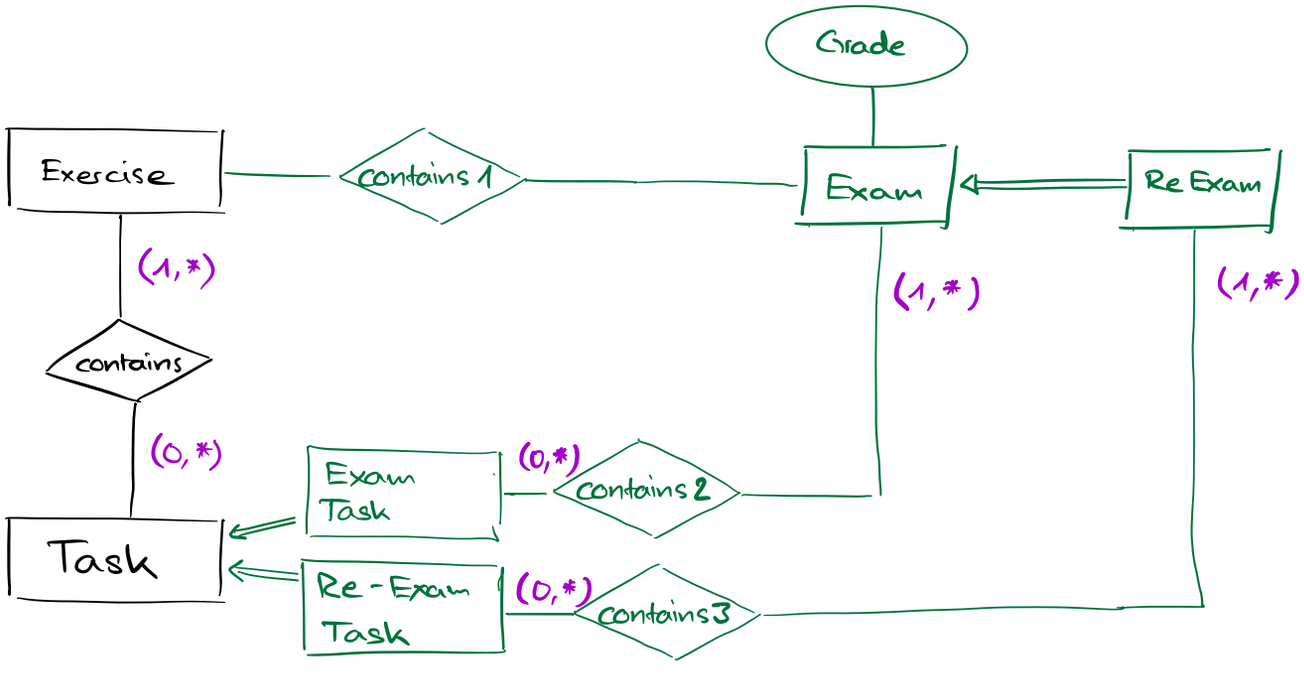
Hand-In: Add the missing cardinalities and correct the 5 mistakes. For each mistake give a short explanation (one sentence) why it is wrong. Hand in your solutions via ADAM.

Task (Theory) 3: ER University Adjusted (3 points)

For this exercise you will adjust the entity-relationship (ER) model from the previous scenario:

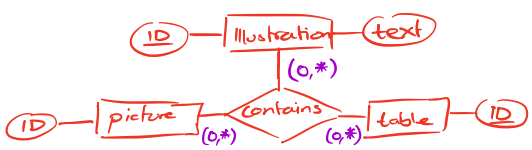
> Since you have successfully modeled your exercise pool, you have the idea that you could use this pool not only to create the exercises but also to arrange examinations. What changes do you have to make to your model to be able to do that? Note that there are examinations and re-examinations; their contents must not overlap and tasks used in exercises must not be used in examinations. In general, an examination task should be used only once while exercise tasks can be used several times. Which of the conditions stated above cannot be represented in an ER diagram?

Hand-In: Do not enter your additions into the model of the task 2. Draw only the parts which change by the new circumstances. Hand in your solutions via ADAM.



- Exam contains an Exercise which contains a Task

↳ Task is now in Exercise and in Exam.


ID
Illustration
text
(0,*)
ID
picture
(0,*)
contains
(0,*)
table
ID