Prof. Dr. Sandra Geisler
Arnab Chakrabarti, Soo-Yon Kim

October 18, 2021

Implementation of Databases
(Winter Term 2021/2022)

Exercise 1

Due until Friday, October 29 at 23:59.
Please submit your solutions in two files to Moodle: As a Jupyter notebook *and* an exported PDF version of your notebook.
Please do *not* submit handwritten solutions!
Find instructions on how to export notebooks as PDF in the Index (only accessible in the Jupyter Hub - see instructions below).
Please submit your solutions in groups of three.
Solutions to this exercise will be presented on Friday, November 5.

Group members: [Name, matriculation number], [name, matriculation number], [name, matriculation number]

Insert all group members by double-clicking on this cell.

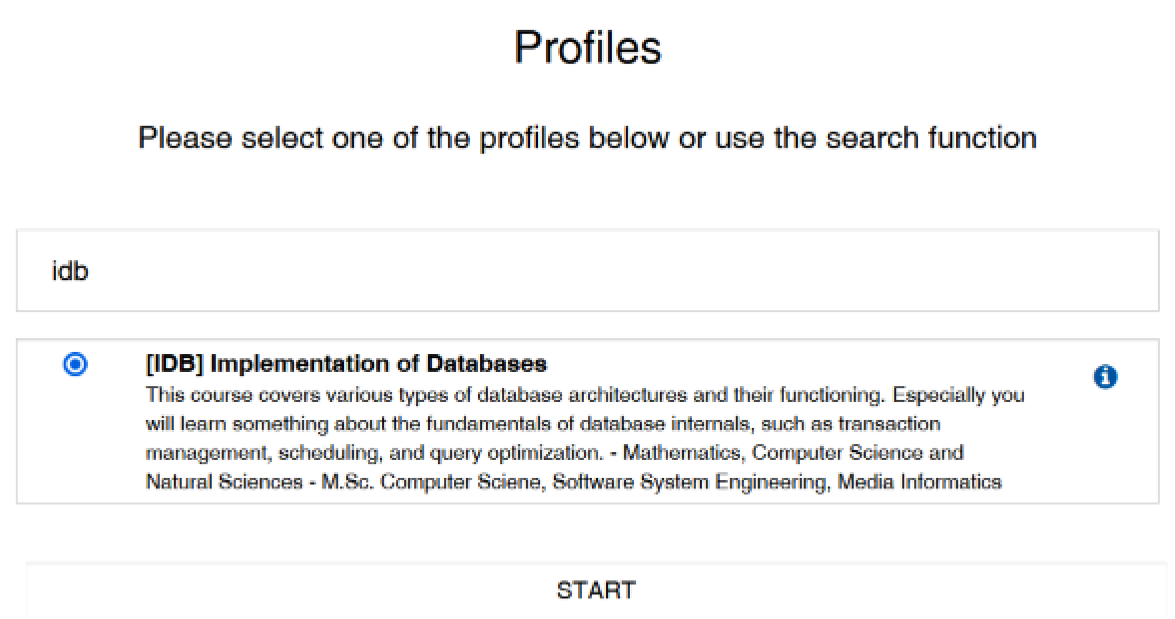
Instructions - **please read carefully!**

Resources linked in this sheet are only accessible via Jupyter Hub!

Jupyter Hub

Go to the website https://jupyter.rwth-aachen.de/ and sign in with Shibboleth:

Select the profile `[IDB] Implementation of Databases`.



Now you can work on exercises and access resources linked in this sheet, such as the Index and Troubleshooting files, where you can find further help on Jupyter.

How to fill in the assignment

At various points in this task sheet, you will find the following code structure:

```
# YOUR CODE HERE
raise NotImplementedError
```

Insert your solution code **instead of these two lines**. Example:

```
# Select all rows from the table "Artist".
# YOUR CODE HERE
raise NotImplementedError
```

The above code should be changed to:

```
# Select all rows from the table "Artist".
%sql SELECT * FROM "Artist"
```

For written tasks, you will find Markdown cells marked with: `[ Please replace this text with your answer. ]`
Here, you can double-click to edit the cell and use Markdown to format the answer.

Relational algebra

Expressions of relational algebra can be written down in Markdown using LaTeX. You can create the expressions with the corresponding LaTeX commands described below. Double-click this cell to see the commands.

Let R and S be matching relations, $A_1, \dots, A_n$ be a choice of n pairwise distinct attributes of R, and let F be a logical formula:

- Union: $R \cup S$
- Intersection: $R \cap S$
- Difference: $R - S$
- Cartesian product: $R \times S$
- Projection: $\Pi_{A_1,\dots,A_n}(R)$
- Selection: $\sigma_F(R)$
- Renaming: $\rho_S(R)$ or $\rho_{A \leftarrow A_1}(R)$
- Natural Join: $R \bowtie S$
- Theta Join: $R \bowtie_\theta S$
- Left Outer Join: $⟕$
- Right Outer Join: $⟖$
- Full Outer Join: $⟗$
- Semi-join: $\rtimes$ or $\ltimes$
- Logical And: $\wedge$
- Logical Or: $\vee$
- Comparison operators: $<, >, \leq, \geq, =, \neq$

Note: '$' is used to enclose mathematical expressions in LaTeX in Markdown. For linebreaks within mathematical expressions, insert '\\'.

Querying with SQL within the Jupyter notebook

Some tasks will require you to write SQL queries. We will work with the Chinook database (source) which stores data of an online shop selling music tracks and albums.
You can take a look at the database schema here or open the chinook.png file.

To start the database and load the SQL extension in this notebook, run `Strg + Enter` in the following code cells. You will then be able to run SQL queries within this notebook.

In [ ]:
```
# Start the database
from IPython.display import Markdown, display
path = "assets/data2"
try:
    running_tests
except NameError:
    import os.path
    if not os.path.exists(path):
        display(Markdown("# Initializing database."))
        display(Markdown("### Extracting database."))
        !tar -zx --touch --checkpoint=.50 -f assets/chinook.tar.gz -C assets/
        display(Markdown("### Initializing database"))
    !chmod 700 $path
    display(Markdown("# (Re)starting server."))
    if os.path.exists(path + "/postmaster.pid"):
        !pg_ctl -D $path restart
        display(Markdown("### Database restart OK"))
    else:
        !pg_ctl -D $path start
        display(Markdown("### Database start OK"))
```

In [ ]:
```
# Load SQL extension and set environment variable
%load_ext sql
%set_env DATABASE_URL=postgresql://jovyan:jovyan@localhost:5432/chinook
```

In [ ]:
```
# Now you can run SQL queries.
# Start the code with '%sql'. For linebreaks, insert \ at the end of line.
# Run an example query:
%sql SELECT *\
    FROM "Artist"
# The query should return 275 rows.
```

Exercise 1.1 (Database Architecture) (14 pts)

1. Name each of the five layers in the database architecture specified in the lecture. Explain the concepts handled in each layer, and the interfaces between layers.

[Please replace this text with your answer.]

2. The following tasks belong to different layers. Sort them so that they match the architecture top-down.
   (a) buffering
   (b) logical relation and cursor management
   (c) media access
   (d) access path management
   (e) view formulation and management

[Please replace this text with your answer.]

3. (a) What does data independence mean? Explain the different types of data independence.
   (b) Why is it an important feature of database systems?
   (c) How is data independence achieved in the five-layered architecture?

[Please replace this text with your answer.]

Exercise 1.2 (Query Languages) (16 pts)

For the Chinook database (schema), formulate the following queries as expressions in both relational algebra and SQL:

1. Find the ID and the name of all tracks where the artist is also the composer.
2. Find the ID, last name and first name of all employees who report to employees which have been hired after themselves.
3. Find the ID, invoice date, and total amount (attribute Total) of the invoice with the highest total amount in Germany.
4. Find the managers of employees supporting Brazilian customers.

Relational algebra:

Query 1

[Please replace this text with your answer.]

Query 2

[Please replace this text with your answer.]

Query 3

[Please replace this text with your answer.]

Query 4

[Please replace this text with your answer.]

SQL:

In [ ]:
```
# Start the code with '%sql'. For linebreaks, insert \ at the end of line.
# Query 1
# YOUR CODE HERE
raise NotImplementedError
```

In [ ]:
```
# Query 2
# YOUR CODE HERE
raise NotImplementedError
```

In [ ]:
```
# Query 3
# YOUR CODE HERE
raise NotImplementedError
```

In [ ]:
```
# Query 4
# YOUR CODE HERE
raise NotImplementedError
```