Tutorial 4

Ex2, UML/ER, about Ex3

Exercise 2 Feedback

- submitted exercises look very good :)

some (maybe) useful pages:

- html color codes
- color picker
- font finder
- CSS values and units

Shared Secret Changed:

Update it here: Change Shared Secret - Zürich (**11.10.2023**)

- Gets changed regularly

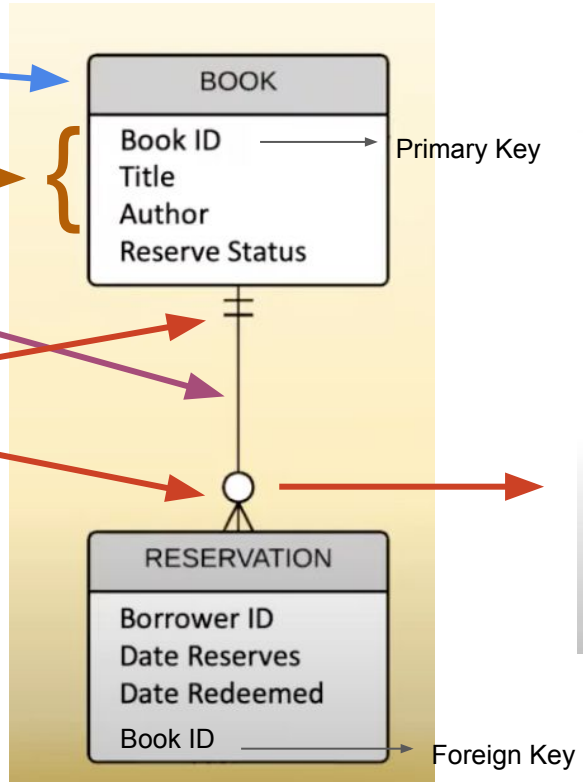
UML/ER

- In both ER and UML diagrams, the goal is to visually represent the structure and relationships of data in a clear and organized manner, making it easier to design and understand the database or system you are working on.

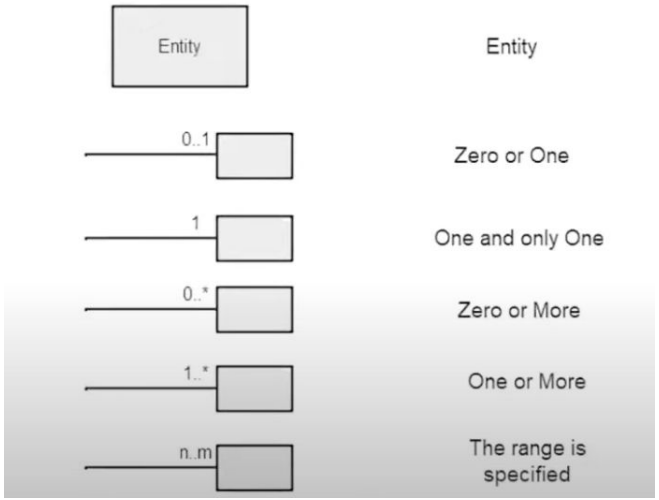
UML	- more general modeling language used for a wide range of software design aspects, including system architecture, behavior, and database design - can represent database-related information, but their main purpose is not limited to databases
ER	- specialized for representing the structure and relationships within a database - primary focus is on modeling the data in a database, including entities (tables), attributes (columns), and relationships between these entities

ER

- Entities
- Attributes
- Relationships
- Cardinality

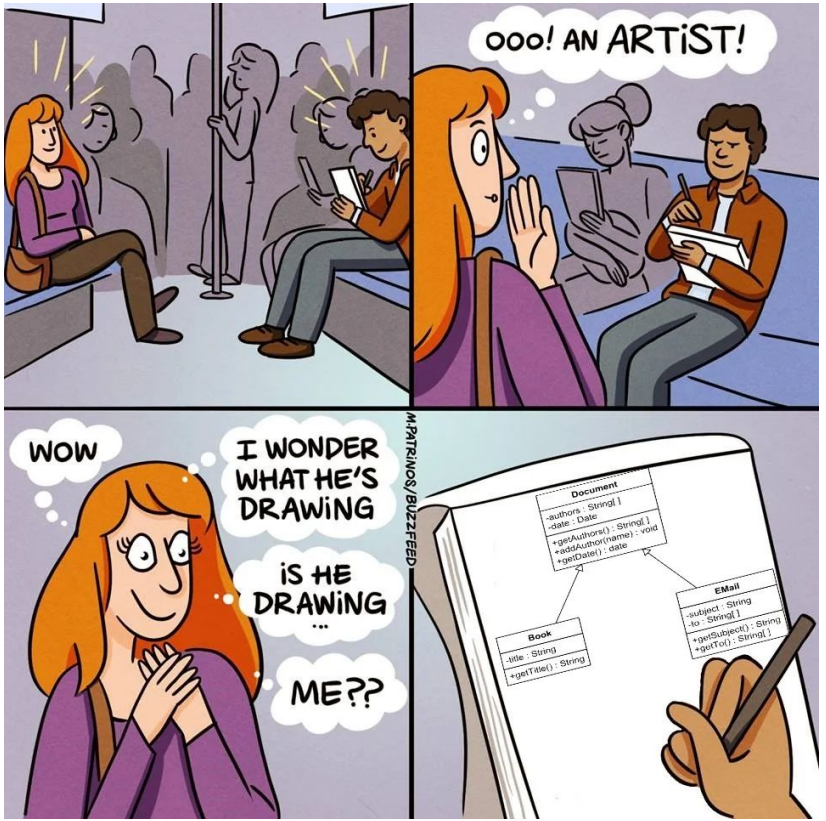


Cardinalities



UML & ER

- ER Tutorial + Example
- ER Tutorial Part 1
- ER Tutorial Part 2
- UML Tutorial



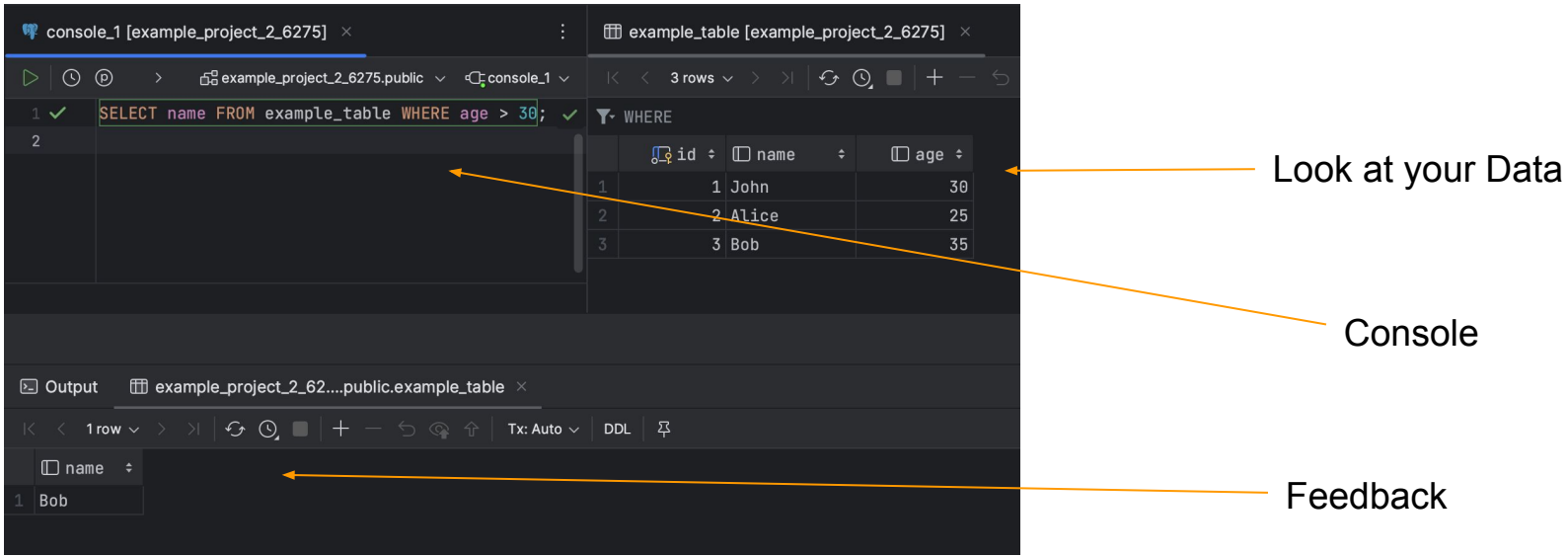
Exercise 3

3 Tasks:

- Getting familiar with PostgreSQL Commands
- Creating a ER Diagram based on provided Data
- Answering questions about the data with queries using psycopg connection

For task 3: Manipulating your Database with SQL

- Through a Console: (Example Image from DataGrip Tool)
  - connect to your database as owner and run your commands in a console directly
  - no python needed
  - instant feedback run multiple commands at once or select specific queries



For task 3: Manipulating your Database with SQL

- Through python using psycopg:
  - Connect via webapp_user
  - You can read a .sql file with a script and execute it:
  - Or you can write single SQL commands like:
    - the output of your query can then be processed further in your python script

```
#get your connection
cur = connection.cursor()
# open a sql file thats in your app folder:
with app.open_resource('schema.sql') as f:
    # execute the SQL commands written inside it
    #(careful to always separate multiple commands with a ; )
    cur.execute(f.read().decode('utf8'))
# commit the changes
db.commit()
# close connection again
cur.close()
```

```
select_query = """
    SELECT *
    FROM messages
    ORDER BY added_on DESC
    LIMIT 5;
"""
cursor.execute(select_query)
messages = cursor.fetchall()
```

For task 3: Manipulating your Database with SQL

You can have two separate connection open in parallel

For example: one owner connection and one webapp_user

- Usually we want to give the webapp_user just enough access to our database
- The owner as an administrator on the other hand can access everything and should in no way be accessible for anyone (security risks etc.)

Uniform Templates for your Flask App (second try)

→ (Demo)