Dateien

open

close

```python
# Du findest die den Text hier https://www.gutenberg.org/files/1787/1787.txt

file = open("1787.txt", "rt")
# open() gibt ein File-Objekt zurück, das wir in der Variable file speichern.
# Da wir hier hamlet.txt ohne weitere Pfadangaben öffnen, wird die Datei im
# aktuellen Verzeichnis gesucht.
# Wir öffnen die Datei im Lesemodus (r) und im Textmodus (t), da wir mit
# Text arbeiten wollen. "rt" ist aber Default und könnte weggelassen werden.
count = 0

try:
    for line in file:
        if "Ham." in line:
            print(line.strip())
            # strip entfernt auch den Zeilenumbruch, den wir mit print() hinzufügen:
            # Fällt dir eine Alternative dazu ein?
            count += 1
    print("Hamlet played", count, "times")
finally:
    file.close()
    # close() wird immer ausgeführt, auch wenn eine Exception geworfen wird,
    # damit stellen wir sicher, dass die Datei nicht geöffnet und damit gesperrt bleibt.
```

with

Context Manager

```python
count = 0

# Die Verwendung eines Kontextmanagers (with) ist die bessere Variante.
with open("1787.txt") as file:
    for line in file:
        if "Ham." in line:
            print(line.strip())
            count += 1
    print("Hamlet played", count, "times")
# with schließt die Datei automatisch, auch wenn eine Exception geworfen wird.
```

```python
with open("print2file.txt", "wt") as f:
    print("Hello, World!", file=f)
    # print() schreibt hier in eine Datei, statt auf Standardausgabe.
    # Führe das Programm mehrfach aus und schau dir die Datei an.
    # Was fällt dir auf?
    # Was passiert, wenn du die Datei löscht und das Programm erneut ausführst?
    # Ändere nun den Parameter "wt" in "a" und führe das Programm erneut aus.

########## Allgemeiner Hinweis ##########
## Standardausgabe (sys.stdout) in Python ist die Konsole
## Standardfehlerausgabe (sys.stderr) in Python ist die Konsole
## Standardeingabe (sys.stdin) in Python ist das Keyboard
```

```python
# https://www.google.de/images/branding/googlelogo/1x/googlelogo_light_color_272x92dp.png

with open("googlelogo_light_color_272x92dp.png", "rb") as picture:
    print(picture.read(10))


# Du möchtest mehr mit Bildern machen? Schau dir mal PIL an.
```

CSV

```python
with open("names.csv") as file:
    for line in file:
        row = line.strip().split(",")
        print(f"{row[0]} wurde geboren im Jahr {row[1]}")

# Variante 2 mit Sortierung
team = []

with open("names.csv") as file:
    for line in file:
        name, jahr = line.strip().split(",")
        team.append(f"{name} wurde geboren im Jahr {jahr}")

for member in sorted(team):
    print(member)
```

```python
# Variante 3 mit dicts
team = []

with open("names.csv") as file:
    for line in file:
        name, jahr = line.strip().split(",")
        member = {}
        member["name"] = name
        member["jahr"] = jahr
        # oder: member = {"name": name, "jahr": jahr}
        team.append(member)
        # oder: team.append({"name": name, "jahr": jahr})

for member in team:
    print(f"{member['name']} wurde geboren im Jahr {member['jahr']}")
```

```python
# Variante 4 mit sortierten dicts
team = []

with open("names.csv") as file:
    for line in file:
        name, jahr = line.strip().split(",")
        team.append({"name": name, "jahr": jahr})


def get_name(member):
    return member["name"]


for member in sorted(team, key=get_name):
    print(f"{member['name']} wurde geboren im Jahr {member['jahr']}")
```

```python
# Variante 5 mit sortierten dicts und lambda
team = []

with open("names.csv") as file:
    for line in file:
        name, jahr = line.strip().split(",")
        team.append({"name": name, "jahr": jahr})

for member in sorted(team, key=lambda member: member["name"]):
    print(f"{member['name']} wurde geboren im Jahr {member['jahr']}")
```

```python
import csv

movies = []

with open("imdb_top_1000.csv", encoding="UTF-8") as file:
    reader = csv.reader(file)
    next(file)  # Überspringt die erste Zeile
    for row in reader:
        movies.append({"title": row[1], "rating": row[6]})

ranked = sorted(movies, key=lambda movie: movie["rating"], reverse=True)
for i in range(10):
    print(f"{ranked[i]['rating']}: {ranked[i]['title']}")
```

```python
import csv

name = input("Wie ist deine Name? ")
icecream = input("Was ist dein Lieblingseis? ")

with open("favorites.csv", "a", newline="") as file:
    writer = csv.writer(file)
    writer.writerow([name, icecream])
```

Datenbanken

SQL

Structured Query Language

Was gibt es sonst noch? NoSQL: Document, Graph, (...)

Table

name	level	class	race
Conan	20	Fighter	Human
Gandalf	20	Wizard	Human
Frodo	4	Thief	Halfling
Gimli	8	Fighter	Dwarf
Legolas	8	Archer	Elves

SQL Databank Management Systeme

- MySQL
- Microsoft SQL
- PostgreSQL
- SQLite

(…)

SQL Datentypen

- SQLite
  - `TEXT`
  - `NUMERIC`
  - `INTEGER`
  - `REAL`
  - `BLOB`
- MySQL
  - `CHAR(size)`
  - `VARCHAR(size)`
  - `SMALLINT`
  - `INT`
  - `BIGINT`
  - `FLOAT`
  - `DOUBLE`
  - …

CREATE TABLE

```
CREATE TABLE characters (
     id INTEGER PRIMARY KEY AUTOINCREMENT,
     name TEXT NOT NULL,
     level INTEGER NOT NULL,
     class TEXT NOT NULL,
     race TEXT NOT NULL
);
```

Constraints

- CHECK
- DEFAULT
- NOT NULL
- PRIMARY KEY
- UNIQUE
- …

INSERT

```
INSERT INTO characters

     (name, level, class, race)

     VALUES (‘Conan’, 20, ‘Fighter’, ‘Human’);
```

```
SELECT * FROM characters;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT name, level FROM characters;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE id = 4;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE race = ‚Human‘;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```sql
SELECT * FROM characters WHERE level < 15;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE level < 15
        AND class = ‚Fighter‘;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE level < 15
        OR class = ‚Fighter‘;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE class IN
    (‚Wizard‘, ‚Thief‘);
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

```
SELECT * FROM characters WHERE race LIKE
    ‚%l%‘;
```

id	name	level	class	race
1	Conan	20	Fighter	Human
2	Gandalf	20	Wizard	Human
3	Frodo	4	Thief	Halfling
4	Gimli	8	Fighter	Dwarf
5	Legolas	8	Archer	Elves

Funktionen

- `AVERAGE`
- `COUNT`
- `MAX`
- `MIN`
- `SUM`
- …

UPDATE

```
UPDATE characters

     SET level = 21

    WHERE name = ‘Conan’

    AND class = ‘Fighter’;
```

DELETE

```
DELETE FROM characters WHERE race = ‘Halfling’;
```

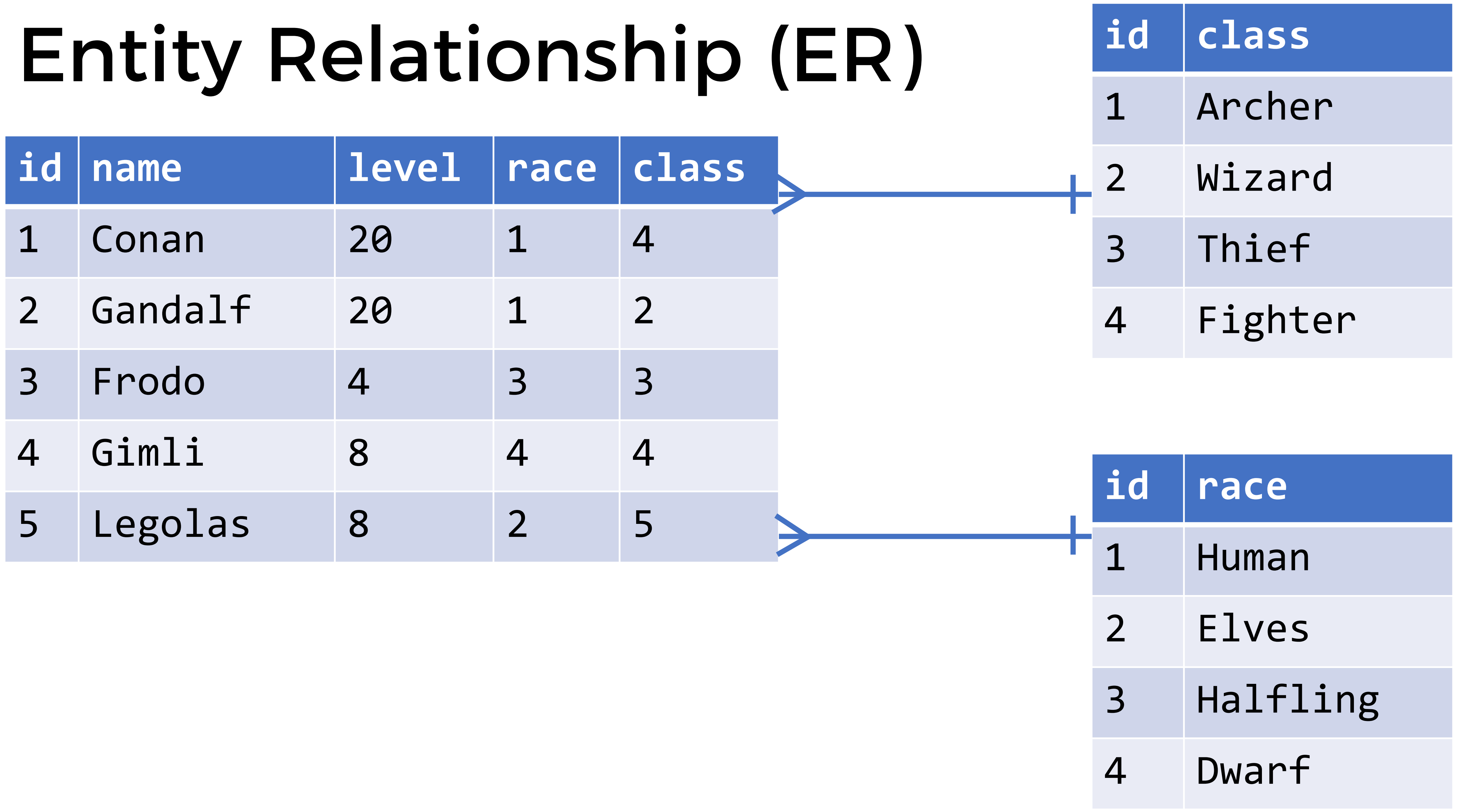
CRUD

Sonstiges

- `LIMIT`
- `ORDER BY`
- `GROUP BY`
- `HAVING`
- …

FOREIGN KEYS

Entity Relationship (ER)



id	name	level	race	class
1	Conan	20	1	4
2	Gandalf	20	1	2
3	Frodo	4	3	3
4	Gimli	8	4	4
5	Legolas	8	2	5

id	class
1	Archer
2	Wizard
3	Thief
4	Fighter

id	race
1	Human
2	Elves
3	Halfling
4	Dwarf

SQLite3

SQLite CLI

- `.mode TABLE`
- `.table`
- `.schema`
- `.quit`
- …

```
SQLite version 3.41.2 2023-03-22 11:56:21
Enter ".help" for usage hints.
sqlite> SELECT COUNT(title) FROM movies WHERE id IN (SELECT movie_id FROM stars WHERE person_id =
(SELECT id FROM people WHERE name = 'John Belushi’));
7
```

```
SQLite version 3.41.2 2023-03-22 11:56:21
Enter ".help" for usage hints.
sqlite> SELECT title,year FROM movies WHERE id IN (SELECT movie_id FROM stars WHERE person_id = (SELECT
id FROM people WHERE name = 'John Belushi’));

+----------------------------------+------+
|              title               | year |
+----------------------------------+------+
| Goin' South                      | 1978 |
| National Lampoon's Animal House | 1978 |
| 1941                             | 1979 |
| Old Boyfriends                   | 1979 |
| The Blues Brothers               | 1980 |
| Continental Divide               | 1981 |
| Neighbors                        | 1981 |
+----------------------------------+------+
```