Zugriff auf eine SQL-Datenbank per Access-Frontend

Eine Datenbank verwendet heute so gut wie jeder. Dabei ist eine elegante Verwaltung und Benutzung der Daten schon selbstverständlich. Kein Anwender möchte per ssh-Zugriff im Terminal oder mit phpMyAdmin die Datenbank verwalten. Somit gehört die Entwicklung eines *Frontends* zu den Aufgaben eines Datenbank-Frontendentwicklers.
Der vorliegende Beitrag zeigt, wie du von einem Access-Frontend aus auf die MySQL-Datenbank auf dem Internetserver zugreifen und mit den Daten so arbeiten kannst, als würden diese auf dem heimischen Rechner liegen. Die benutzte Sprache ist *Python*. Die Schritte sind jedoch mit jeder anderen Hochsprache möglich. Wir starten zuerst mit dem Zugriff auf eine Datenbank, die auf demselben Rechner liegt, wie der an dem der Client arbeitet. (localhost)

Python und SQLite

SQLite (SQL - Structured Query Language) ist ein dateibasiertes Datenbanksystem, welches in vielen Endgeräten und in Standardbrowsern (Safari, Chrome, Firefox) verwendet wird. Es ist ein relationales Datenbanksystem und basiert auf dem SQL-92 Standard (Quelle: Wikipedia.) Der große Vorteil ist in der einfachen Verwendung von SQLite. Es braucht kein Datenbakserver installiert werden, die Speicherung der Daten erfolgt in einfachen Textdateien und die API ist standardiserit. So besitzt Python eine API, welche per `import sqlite3` verfügbar gemacht wird.
Der folgende Quelltext gibt einen Einblick in die Benutzung von SQLite.

```
1  import os, sys, sqlite3
2
3  # Existenz feststellen
4  if os.path.exists("universitaet.db"):
5      print("Datei␣bereits␣vorhanden")
6      sys.exit(0)
7
8  # Verbindung zur Datenbank erzeugen
9  connection = sqlite3.connect("universitaet.db")
10
11 # Datensatz-Cursor erzeugen
12 cursor = connection.cursor()
13
14 # Datenbanktabelle erzeugen
```

```
15 sql = "CREATE␣TABLE␣professoren(" \
16       "Name␣TEXT," \
17       "PersNr␣INTEGER␣PRIMARY␣KEY," \
18       "gehalt␣REAL," \
19       "Rang␣TEXT," \
20       "Raum␣INTEGER)"
21 cursor.execute(sql)
22
23 # Datensatz erzeugen
24 sql = "INSERT␣INTO␣professoren␣VALUES('Socrates'," \
25       "2125,3500,'C4',226)"
26 cursor.execute(sql)
27 connection.commit()
28
29 # Datensatz erzeugen
30 sql = "INSERT␣INTO␣professoren␣VALUES('Russel'," \
31       "2126,3750,'C3',320)"
32 cursor.execute(sql)
33 connection.commit()
34
35 # SQL-Abfrage
36 sql = "SELECT␣*␣FROM␣professoren"
37
38 # Kontrollausgabe der SQL-Abfrage
39 # print(sql)
40
41 # Absenden der SQL-Abfrage und Empfang des Ergebnisses
42 cursor.execute(sql)
43
44 # Ausgabe des Ergebnisses
45 for dsatz in cursor:
46     print(dsatz[0],dsatz[1],dsatz[2],dsatz[3],dsatz[4])
47
48 # Datensatz entfernen
49 sql = "DELETE␣FROM␣professoren␣WHERE␣PersNr␣=␣2125"
50 cursor.execute(sql)
51 connection.commit()
52
53 # Ausgabe der Tabelle
```

```
54 sql = "SELECT␣*␣FROM␣professoren"
55 cursor.execute(sql)
56
57 for dsatz in cursor:
58     print(dsatz[0],dsatz[1],dsatz[2],dsatz[3],dsatz[4])
59
60 # Verbindung beenden
61 connection.close()
```

Python und lokaler MySQL-Server

Dieser Fall ist sehr gut in den Dokumentation von Python aufgeführt. Benötigt wird lediglich das Modul *mysql.connector*, welche per `pip install mysql-connector-python` installiert werden kann.

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4    host="localhost",
5    user="dbuser",
6    password="informatik"
7  )
8
9  mydb.database = 'unidatenbank'
10 print(mydb)  # Kontrolle erfolgreiche Kontaktaufnahme
11
12 # SQL-Befehl ausfuehren
13 cursor = mydb.cursor()
14 SQLBefehl = 'SELECT␣*␣FROM␣Professoren'
15 cursor.execute(SQLBefehl)
16
17 # Durchlaufen der Ergebnisse
18 row=cursor.fetchone()
19 while (row!=None):
20   print(row[0], row[1])
21   row = cursor.fetchone()
22
23 # Ende der Verarbeitung
24 cursor.close()
```

Alternativ kann man auch mit dem Modul `MySQLdb` arbeiten.

Python und entfernter MySQL-Server

Diese Variante kommt immer dann zum Tragen, wenn die Daten und somit die Datenbank auf einem anderen Rechner arbeitet als der Client. Das kann z. B. ein DB-Server im Internet sein, oder auch ein Server im lokalen Netzwerk, z. B. einer Firma. Auch hier zeigt wieder das Beispiel (Quelltext unten, wie man den Zugriff realisiert. Eines ist jetzt zu beachten. Der Zugriff auf die Datenbank muss für den DB-User und auch der Datenbank selbst authorisiert werden. Ein DB-User wird in der Datenbank standardmäßig nur für den Zugriff von *localhost* eingerichtet. Weiterhin ist der Datenbank mitzuteilen, dass sie nicht nur Anfragen vom *Looback-Interface (127.0.0.1)* entgegennimmt. Im einzelnen ist folgendes durchzuführen:

1. Deaktiviere in der Datei `/etc/my.cnf` die Zeile `bind-address = 127.0.0.1.` durch `#`
2. Gib dem DB-User das Recht von jedem Rechner auf den DB-Server zuzugreifen: `GRANT ALL ON <dbname>.* to '<dbuser>'@'%' IDENTIFIED BY '<password>' WITH GRANT OPTION;` (Die eckigen Klammern sind nicht miteinzugeben und die jeweiligen korrekten Bezeichner einzusetzen. Das '%' bedeutet Zugriff von jedem anderen Rechner ist erlaubt.)
3. `FLUSH PRIVILEGES;` (Der Inhalt des Speichers wird zur Datenbank übermittelt.)

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4    host="85.13.143.90",  #Achtung! Ohne http://
5    user="d035c1e0",
6    password="informatik"
7  )
8
9  mydb.database = 'd035c1e0'
10 print(mydb)  # Dient nur zur Kontrolle der erfolgreichen
       Kontaktaufnahme
11
12 # SQL-Befehl ausfuehren
13 cursor = mydb.cursor()
14 SQLBefehl = 'SELECT␣*␣FROM␣Professoren'
15 cursor.execute(SQLBefehl)
16
17 # Durchlaufen der Ergebnisse
18 row=cursor.fetchone()
```

```
19 while (row!=None):
20   print(row[0], row[1])
21   row = cursor.fetchone()
22 
23 # Ende der Verarbeitung
24 cursor.close()
```