Leibniz Universität IT Services

Leibniz Universität Hannover

SQLite – Nutzung in Python
Einführung in SQL

SQLite

- Datenbank, basierend auf Dateien mit der Endung „.sql“ oder „.db“.
- Programmbibliothek, die im Standard der Programmiersprache Python enthalten ist (Python\Python...\DLLs).
- Nutzung in eingebetteten Systemen wie zum Beispiel Android oder zur Speicherung von Lesezeichen im Firefox.
- Webseite: https://sqlite.org/

Tutorials im Web

- https://docs.python.org/3.6/library/sqlite3.html
- https://www.python-kurs.eu/sql_python.php
- https://www.pythoncentral.io/introduction-to-sqlite-in-python/
- https://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html

Import der Programmbibliothek

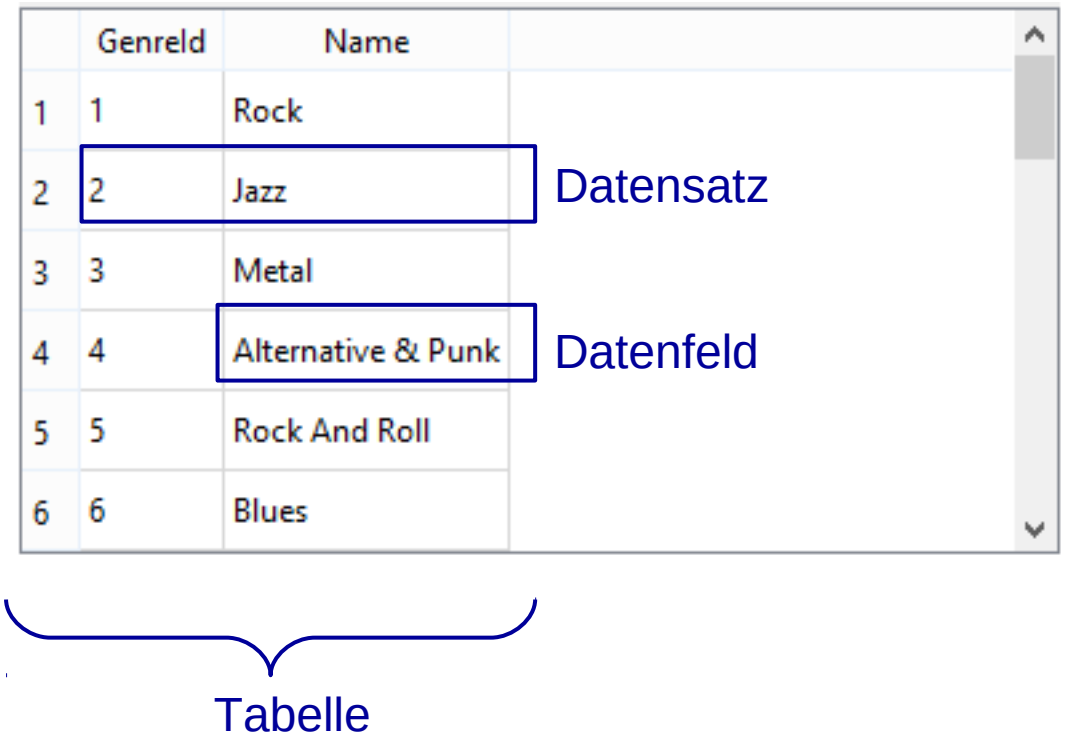
```
import sqlite3
```

- Bereitstellung der Funktionen, Methoden und Datentypen der Programmbibliothek sqlite3.
- Mit Hilfe des Schlüsselwortes import wird ein Modul in eine Code-Datei eingebunden.
- Nutzung mit Python 3.x.

Relationales Datenbankmodell

- Ablage von Daten in Tabellen. Für jede abzubildende Objektgruppe wird eine Tabelle angelegt.
- Erstellungen von Beziehungen (Relationen) zwischen Tabellen. Objektgruppen interagieren miteinander. Zu einem Vorgang werden Details angezeigt.
- Manipulation der Daten mit Hilfe der Sprache SQL.
- Zum Beispiel nutzen das Datenbanksystem SQLite ein relationales Datenbankmodell.

Begriffe


Genreld
Name
1
1
Rock
2
2
Jazz
3
3
Metal
4
4
Alternative & Punk
5
5
Rock And Roll
6
6
Blues
Datensatz
Datenfeld
Tabelle

Tabelle

- Container für die Sammlung von Daten in einer Zeilen / Spalten - Struktur
- Sammlung von Elementen einer bestimmten Gruppe.
- Strukturierte Ablage von Attribut-Werten eines Objekts von einer bestimmten Kategorie.
- Jede Zeile in einer Tabelle kann durch ein Schlüssel eindeutig identifiziert werden.

Datensätze

- Sammlung von Attributen, die eine Gruppe von Elementen charakterisieren.
- Der Satz von Daten beschreibt ein Element aus einer Gruppe.
- Jede Zeile in einer Tabelle stellt einen Datensatz dar.

Datenfelder

- Datenfelder enthalten exakt eine Information.
- Die Attribut-Werte eines Elements werden gesetzt.
- In den Kreuzungspunkt Zeile – Spalte wird der Wert eines Attributs gesetzt. Jede Spalte in einer Tabelle definiert ein bestimmtes Attribut. Jedes Element unterscheidet sich von allen Elementen in mindestens einem Attribut-Wert.

Was ist eine „Datenbank“?

- Zusammenfassung von Tabellen zu einem Thema
- Verwaltung von großen Datenmengen.
- Strukturierte Ablage von Daten.
- SQLite: Ablage von großen Datenmengen in einer Datei an einem beliebigen Speicherort.

Beziehung (Relation) zwischen Tabellen

	AlbumId	Title	ArtistId
1	1	For Those About To Rock We Salute You	1
2	2	Balls to the Wall	2
3	3	Restless and Wild	2
4	4	Let There Be Rock	1
5	5	Big Ones	3
6	6	Jagged Little Pill	4
7	7	Facelift	5
8	8	Warner 25 Anos	6
9	9	Plays Metallica By Four Cellos	7
10	10	Audioslave	8
11	11	Out Of Exile	8
12	12	BackBeat Soundtrack	9

	ArtistId	Name
1	1	AC/DC
2	2	Accept
3	3	Aerosmith
4	4	Alanis Morissette
5	5	Alice In Chains
6	6	Antônio Carlos Jobim
7	7	Apocalyptica
8	8	Audioslave
9	9	BackBeat
10	10	Billy Cobham
11	11	Black Label Society
12	12	Black Sabbath

Erläuterung

- Abbildung einer Hierarchie von Tabellen.
- Mit Hilfe von Schlüsselwerten werden Datensätze verknüpft.
- Verweis mit Hilfe eines eindeutigen Schlüssels auf eine andere Tabelle.

Beispiel-Datenbanken im Web

- https://chinookdatabase.codeplex.com/
- https://www.microsoft.com/en-us/download/details.aspx?id=23654

Öffnen einer Datenbank

```
import sqlite3

datenbank = "chinook.db"
connection = sqlite3.connect(datenbank)
```

- Mit Hilfe der Methode .connect() wird eine Verbindung zu einer Datenbank hergestellt.
- Die angegebene Datenbank wird geöffnet. Die Referenz auf die geöffnete Datenbank wird in der Variablen connection gespeichert.
- Hinweis: Die Referenz darf nicht verloren gehen. Andernfalls kann die Datenbank nicht geschlossen werden.

Hinweise zum Dateinamen

- Der Dateiname plus die Dateiendung werden als String an die Methode .connect() übergeben.
- Falls nur der Dateiname angegeben wird, wird die Datenbank im Verzeichnis der Code-Datei gesucht.
- Der Speicherort der Datenbank kann relativ zum Programmcode oder absolut angegeben werden.

Schließen einer Datenbank

```
import sqlite3

datenbank = "chinook.db"
connection = sqlite3.connect(datenbank)

connection.close()
```

- Mit Hilfe der Methode .close() wird die Verbindung zur Datenbank getrennt.
- Die Datenbank, auf die die Instanz verweist, wird geschlossen.

S(tructured)Q(uery)L(anguage)

- Strukturierte Abfragesprache.
- Standardsprache für relationale Datenbanken.
- Daten in Tabellen manipulieren, aktualisieren, eintragen und löschen.
- Nutzung in allen gängigen relationalen Datenbanksystemen.

Standard für SQL

- Aktueller Standard: SQL:2016 ISO/IEC 9075:2016.
- SQL3 oder SQL:1999
- SQL2 oder SQL-92
- 1986: SQL1
- SQL-Sprachumfang in SQLite: https://www.sqlite.org/lang.html.

Bestandteile

- DDL (Data Definition Language).
- DML (Data Manipulation Language).
- DCL (Data Control Language).
- TCL (Transaction Control Language).

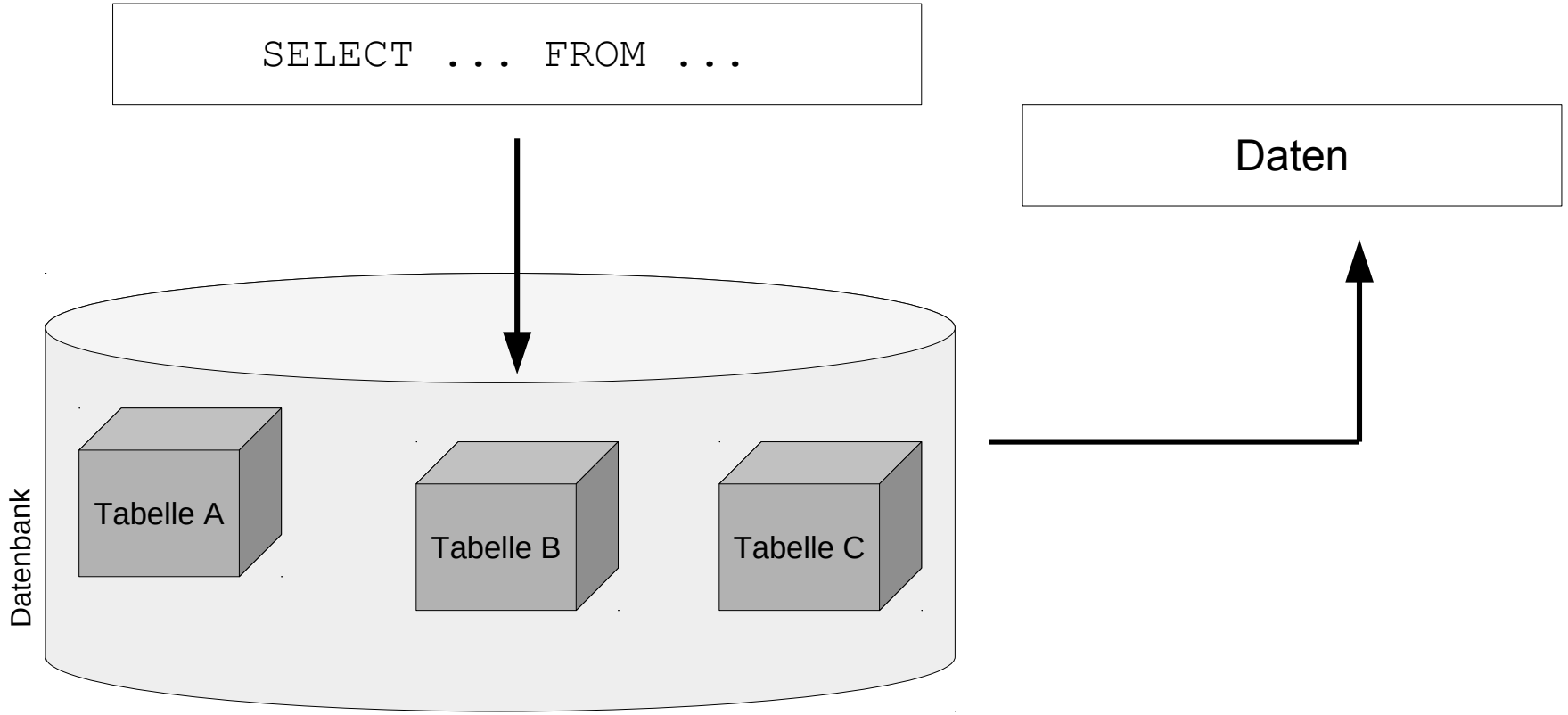
Data Control Language

- Rechteverwaltung.
- Zugriffsrechte auf eine Tabelle.
- Nutzung durch den Administrator.
- Befehle: GRANT, REVOKE.
- Hinweis: Wird in SQLite aufgrund der Ablage der Daten in eine Datei nicht genutzt.

Data Manipulation Language

- Lesen und filtern von Informationen.
- Auswählen, einfügen, aktualisieren oder löschen von Daten in Tabellen.
- Nutzung durch den Anwender.
- Befehle: SELECT, INSERT, UPDATE, DELETE.

Lesen von Informationen



Programmcode

```
import sqlite3

datenbank = "chinook.db"
connection = sqlite3.connect(datenbank)
cursor = connection.cursor()

for row in cursor.execute("SELECT * FROM genres"):
    print(row)

connection.close()
```

Datenbank-Zeiger

```
cursor = connection.cursor()
```

- Mit Hilfe der Methode .cursor() wird ein neues leeres Cursor-Objekt für die geöffnete Datenbank erzeugt.
- Zeiger auf ein Element in der Datenbank.
- Ausführung von Auswahlabfragen. Auswahlabfragen liefern einen Satz von Daten aus einer Tabelle zurück.

Ausführung einer SQL-Anweisung

```
cursor.execute("SELECT * FROM genres")
```

- Mit Hilfe der Methode .execute() wird eine SQL-Anweisung ausgeführt.
- Die Methode liefert entsprechend der Anweisung x Datensätze zurück.
- Die Datensätze können mit Hilfe einer for-Schleife durchlaufen werden.

SQL-Anweisung

```
cursor.execute("SELECT * FROM genres")
```

- Beginn mit einem englischsprachigen Verb (hier: SELECT, Wähle aus).
- Das Verb beschreibt wie die Daten verarbeitet werden sollen.

Hinweise zu der SQL-Anweisung

```
statement = "SELECT FirstName, LastName, Email FROM employees
WHERE (Title = 'Sales Support Agent')"
```

- Die Groß- und Kleinschreibung bei Befehlen, Feldnamen oder Tabellennamen wird nicht beachtet.
- Zwischen Feldnamen und SQL-Befehlen muss ein Leerzeichen stehen.
- Vor und nach Operatoren können Leerzeichen stehen. Nach dem Komma in der Feldliste kann ein Leerzeichen stehen.
- Runde Klammern werden genutzt, um die Lesbarkeit von Bedingungen zu erhöhen.

Weitere Hinweise

```
statement = "SELECT FirstName, LastName, Email FROM employees
WHERE (Title = 'Sales Support Agent')"
```

- SQL-Anweisungen sind vom Datentyp „String“.
- Die SQL-Anweisung beginnt und endet mit Anführungszeichen. Strings innerhalb der SQL-Anweisung werden durch ein Apostroph geklammert.

SQL-Befehle

- Beschreibung einer Aktivität. Zum Beispiel symbolisiert der Befehl SELECT die Tätigkeit „Wähle aus“.
- SQL-Befehle beginnen immer mit einem Buchstaben.
- Um die Lesbarkeit zu erhöhen, werden die Befehle häufig groß geschrieben.

Syntax einer SELECT-Anweisung

SELECT customers.LastName, customers.FirstName, invoices.Total

Welche Felder werden angezeigt?

FROM customers INNER JOIN invoices

In welchen Tabellen sind die Felder definiert?

WHERE invoices.BillingState = 'FL'

Nach welchen Kriterien werden die Daten gefiltert?

GROUP BY invoices.InvoiceDate Gruppierung der Daten

HAVING (strftime('%Y',invoices.InvoiceDate) LIKE '2010') und Filterung

ORDER BY LastName, FirstName

Sortierung der Daten

Data Definition Language

- Definition des Datenbankschemas.
- Erstellung, Änderung und Löschung von Datenbankstrukturen.
- Erstellung und Löschung von Tabellen, in denen die Informationen gespeichert werden.
- Nutzung durch den Administrator.
- Befehle: CREATE, ALTER, DROP.

Neue Datenbank

```
import sqlite3

datenbank = "myDatabase.db"
connection = None

try:
    connection = sqlite3.connect(datenbank)
except sqlite3.Error as e:
    print(e)
finally:
    if not(connection is None):
        connection.close()
```

Öffnen einer Datenbank

```
import sqlite3


datenbank = "myDatabase.db"
connection = sqlite3.connect(datenbank)
```

- Mit Hilfe der Methode .connect() wird eine Verbindung zu einer Datenbank hergestellt.
- Falls die Datenbank an dem angegebenen Speicherort nicht vorhanden ist, wird automatisiert eine neue anlegt.
- Die Referenz auf die neu angelegte Datenbank wird in der Variablen connection gespeichert.

Abfangen von Fehlern

```
try:
    connection = sqlite3.connect(datenbank)

except sqlite3.Error as e:
    print(e)

finally:
    if not(connection is None):
        connection.close()
```

Versuche ...

```
try:
    connection = sqlite3.connect(datenbank)
```

- Das Schlüsselwort try: leitet einen Code-Block ein.
- Die Anweisungen in diesem Block versucht der Python-Interpreter auszuführen.

Wenn ein Fehler auftritt, ...

```
try:
except sqlite3.Error as e:
    print(e)
except:
  print("Laufzeitfehler im Code")
```

- Das Schlüsselwort except leitet die gewünschte Fehlerbehandlung ein.
- Aufgetretene Ausnahmen werden behandelt.
- Die Anweisung except sqlite3.Error as e fängt Fehler ab, die von SQLite erzeugt wurden.
- Die Anweisung except fängt alle nicht explizit behandelnden Fehler ab.

Führe die Anweisungen immer aus

```
finally:
    if not(connection is None):
        connection.close()
```

- Das Schlüsselwort finally fasst Anweisungen zusammen, die immer ausgeführt werden. Egal ob ein Fehler aufgetreten ist oder nicht.
- Aufräumarbeiten ausführen.

Neue Tabelle

```
statement = "CREATE TABLE IF NOT EXISTS kontakt ("
statement = statement + " idKontakt integer PRIMARY KEY"
statement = statement + ", nachname text NOT NULL"
statement = statement + ", vorname text"
statement = statement + ", anrede text NOT NULL"
statement = statement + ", email text NOT NULL UNIQUE"
statement = statement + ")"

cursor = connection.cursor()
cursor.execute(statement)
```

SQL-Anweisung

```
statement = "CREATE TABLE IF NOT EXISTS kontakt ("
```

- Der Befehl CREATE TABLE erzeugt eine neue Tabelle.
- Falls die Tabelle nicht existiert (IF NOT EXISTS), wird die Tabelle neu angelegt.
- Der Name der Tabelle (kontakt) ist in der geöffneten Datenbank eindeutig.

Datenfelder in der Tabelle

```
statement = "CREATE TABLE IF NOT EXISTS kontakt ("
statement = statement + " idKontakt integer PRIMARY KEY"
```

- In den runden Klammern werden die Datenfelder in der Tabelle neu angelegt.
- Jedes Datenfeld hat einen eindeutigen Namen und einen bestimmten Datentyp.
- Die verschiedenen Datenfelder werden durch ein Komma in der Liste getrennt.

Transaction Control Language

- Ablaufsteuerung.
- Jede SQL-Anweisung ist eine Transaktion. Oder: Mehrere Befehle werden zu einer Transaktion zusammengefasst.
- Befehle: COMMIT, ROLLBACK, SAVEPOINT.

Neue Datensätze speichern

```
statement = "INSERT INTO kontakt (nachname, anrede, email)"
statement = statement +
            " VALUES('Muster', 'Herr / Frau', 'mail@muster.de')"

cursor.execute(statement)
connection.commit()
```

Definition des neuen Datensatzes

```
statement = "INSERT INTO kontakt (nachname, anrede, email)"
statement = statement +
            " VALUES('Muster', 'Herr / Frau', 'mail@muster.de')"
```

- Der Befehl INSERT INTO fügt einen neuen Datensatz in die angegebene Tabelle ein.
- Dem Namen der Tabelle folgt eine Liste von Feldern, die befüllt werden.
- Dem Befehl VALUES folgt eine Liste der Werte, die in die angegebenen Felder eingefügt werden.

Speicherung des Datensatzes

```
connection.commit()
```

- Die Methode .commit() speichert alle Änderungen an der Datenbank.