SQLite – Nutzung in Python
Auswahlabfragen

Alle Kunden

Die Namen der Mitarbeiter und deren E-Mail-Adresse

Bestellungen, nach dem Datum sortiert

Auswahlabfragen

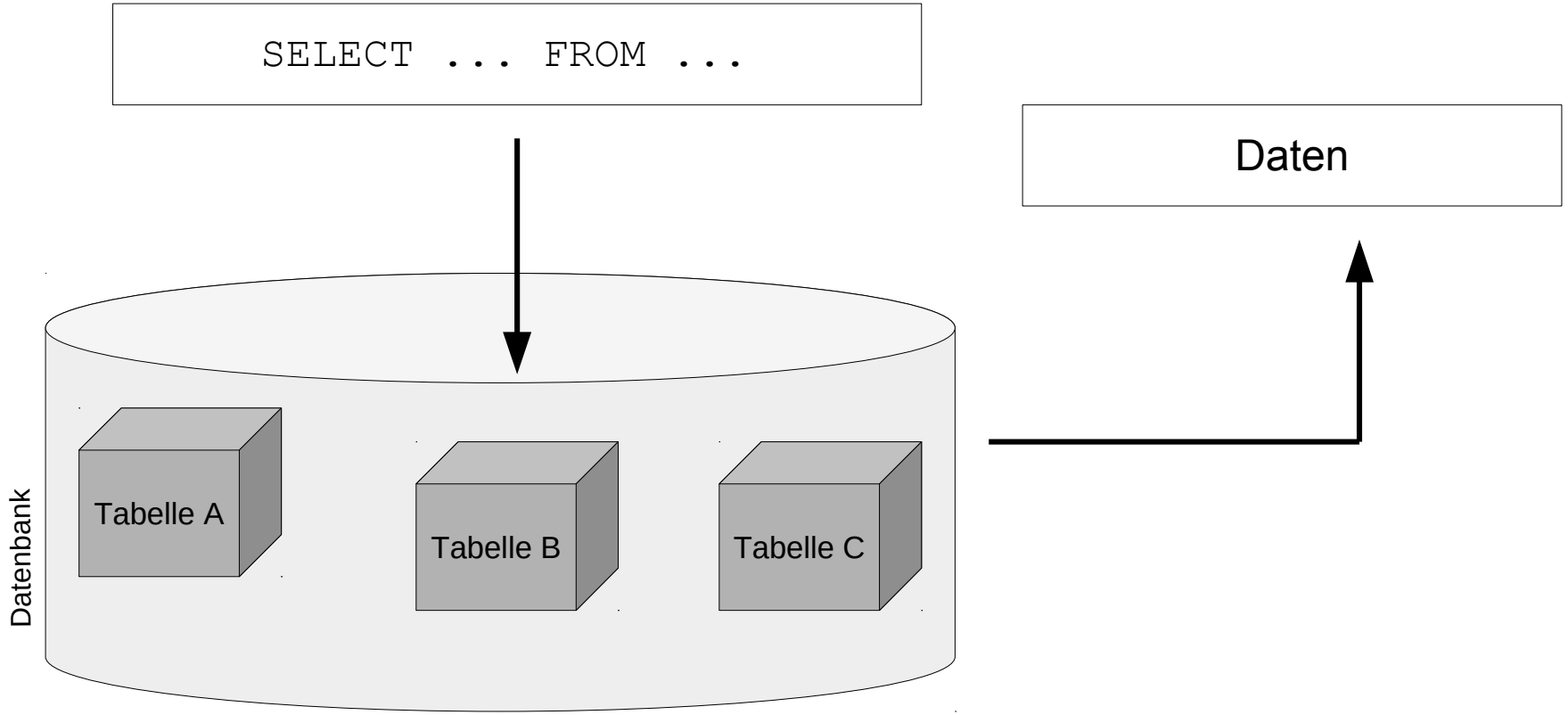
- Beginn mit `SELECT`.
- Auswahl von Daten, die in einer Tabelle gespeichert sind.
- Anzeige von allen oder ausgewählten Attributen eines Objekts.
- Filterung und Sortierung von Informationen zu einem Element.
- Gruppierung von Daten.

Beispiele

```
-- Wähle alle Künstler aus.
SELECT *
FROM artists;
```

```
-- Zeige den Name des Kunden an.
-- Aus welchen Land kommt der Kunde?
SELECT FirstName, LastName, Country
FROM customers;
```

Arbeitsweise



Ergebnis einer Auswahlabfrage

- Anzeige von ausgewählte Datensätzen in einer temporären Ergebnistabelle.
- Das Resultat der Auswahlabfrage ist abhängig von den gespeicherten Informationen in den genutzten Tabellen.
- Die Ergebnistabelle zeigt nur die gewählten Datenfelder an. In diesen Datenfeldern sind die benötigten Informationen abgelegt.

SQL-Befehle

- SQL-Befehle beginnen immer mit einem Buchstaben.
- Verben wie zum Beispiel SELECT beschreiben die auszuführende Anweisung.
- Präpositionen wie zum Beispiel FROM zeigen an woher die Daten kommen.
- Substantive wie zum Beispiel VALUES legen die Art der Daten fest.
- Um die Lesbarkeit zu erhöhen, werden die Befehle häufig groß geschrieben.

SQL-Anweisung

```
SELECT [Feld], [Feld]
FROM [Tabelle]
ORDER BY [Feld] ASC|DESC, [Feld] ASC|DESC;
```

- Präzises Anweisung zur Verarbeitung von Daten durch einen Satz.
- Beginn mit einem SQL-Befehl (hier: SELECT, Wähle aus). Der Befehl beschreibt die gewählte Aktivität.
- Beendigung mit einem Semikolon.

Aufbau von Auswahlabfragen

SELECT	Wähle aus
[Feld] , [Feld]	Liste von Datenfeldern
FROM [Tabelle] INNER | LEFT OUTER JOIN [Tabelle] ON [Primärschlüssel] = [Fremdschlüssel]	Speicherort der Daten Abbildung von Relationen
WHERE ([Bedingung])	Filterung der Daten
GROUP BY [Gruppenfeld] HAVING ([Bedingung])	Gruppierung der Daten
ORDER BY [Feld] ASC|DESC, [Feld] ASC|DESC;	Sortierung der Daten

Auswahlabfragen mit Python starten

```
datenbank = "chinook.db"
strSQL = "SELECT * FROM customers"
strSQL = strSQL + " WHERE (Company IS NOT NULL);"

with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)
    data = cursor.fetchall()
```

Verbindung zu einer Datenbank

```
import sqlite3

with sqlite3.connect(datenbank) as connection:
```

- Mit Hilfe der Methode .connect() wird eine Verbindung zu einer Datenbank hergestellt. Der Methode wird als Argument der Name der gewünschten Datenbank übergeben.
- Für die Verbindung wird ein Alias connection vergeben.

with-Anweisung

```
import sqlite3

with sqlite3.connect(datenbank) as connection:
```

- Es wird versucht eine Verbindung zu der angegebenen Datenbank herzuerstellen.
- Falls die Verbindung möglich ist, wird für die Verbindung ein Alias vergeben und der dazugehörige Block ausgeführt.
- Falls eine Verbindung nicht hergestellt werden kann, wird eine entsprechende Ausnahme angezeigt.
- Beim Verlassen des Blocks oder eines Auftreten eines Fehlers wird automatisch die Verbindung geschlossen.

Cursor-Objekt

```
with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
```

- Der Alias connection symbolisiert ein Objekt vom Typ Connection.
- In dieser Klasse ist die Methode .cursor() definiert. Objekt und Methode werden über den Punkt-Operator miteinander verbunden.
- Die Methode gibt ein Cursor-Objekt auf die angegebene Datenbank zurück.

Cursor

- Bereitstellung von Methoden zur Verarbeitung von SQL-Anweisung in einer relationalen Datenbank.
- Verweis auf einen Datensatz oder einen kleinen Block von Datensätzen, der von einer Auswahlabfrage zurückgegeben wird.
- Datensatz-Zeiger. Verweis auf einen bestimmten Datensatz in einer Tabelle.

SQL-Anweisung ausführen

```
datenbank = "chinook.db"
strSQL = "SELECT * FROM customers"
strSQL = strSQL + " WHERE (Company IS NOT NULL);"

with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)
```

Erläuterung

```
cursor.execute(strSQL)
```

- Die Methode .execute() ist in der Klasse Cursor definiert. Ein Objekt von der Klasse wird mit der Methode durch einen Punkt verbunden.
- Der Methode wird als Parameter eine SQL-Anweisung vom Typ String übergeben.

Hinweise

```
strSQL = "SELECT * FROM customers"
strSQL = strSQL + " WHERE (Company IS NOT NULL);"
```

- Ein String wird durch die Anführungszeichen oder den Apostroph in Python begrenzt.
- Ein String als Literal in der SQL-Anweisung muss durch den Apostroph begrenzt werden, falls der Python-String am Anfang und Ende Anführungszeichen nutzt.

Abholung des Resultats

```
with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)
    data = cursor.fetchall()
```

- Die Methode .fetchall() holt alle Datensätze in der Ergebnistabelle.
- Die Methode liefert eine Liste von Datensätzen zurück.
- Falls keine passenden Datensätze vorhanden sind, wird eine leere Liste zurück gegeben.

Nutzung einer for-Schleife

```
data = cursor.fetchall()

for zeile in data:
    print(noneToString(zeile[1]) + ' ' + noneToString(zeile[2]))
    print(noneToString(zeile[3]))
    print(noneToString(zeile[4]))
    print('\n')
```

Datensätze

- Mit Hilfe einer for-Schleife wird das Resultat Datensatz für Datensatz durchlaufen.
- Jedes Listenelement entspricht einer Zeile in der Ergebnistabelle.
- Die Datensätze können vollständig ausgegeben werden.

Datenfelder

- Datenfelder sind Listen in der Liste „Datensatz“.
- In jedem Listenelement „Datensatz“ kann ein Datenfeld über ein Index angesprochen werden. Ein Index ist immer eine Ganzzahl. Das erste Datenfeld hat den Wert 0 und so weiter.
- Pro Datensatz können die Datenfelder vollständig mit einer for-Schleife durchlaufen werden.
- Die Daten in den Feldern können entsprechend des Datentyps mit Hilfe der Programmiersprache verarbeitet oder ausgegeben werden.

Datentypen der Datenfelder

- Die Nutzung des Datenfeldes wird definiert. Der Datentyp `Text` kann nicht in Berechnungen genutzt werden.
- Die Größe der Datenfelder wird festgelegt. Der benötigte Speicherbedarf wird festgelegt.
- Der Wertebereich für ein Attribut-Wert wird festgelegt.

Storage Classes in SQLite

- SQLite hat keine Datentypen, sondern nur Storage Classes.
- Die Klassen beschreiben einen Datentyp allgemein.
- Informationen im Web: https://www.sqlite.org/datatype3.html.

Datentypen in Python

- Everything is a object.
- Jede Variable in Python kann jeden Datentyp annehmen.
- Entsprechend der Initialisierung der Variablen wird der Datentyp bestimmt.

Datentypen in SQLite und Python

INTEGER	<class 'int'>
REAL	<class 'float'>
TEXT	<class 'str'>
BLOB	Binäre Objekte
NULL	NONE

Ganzzahlen

INTEGER	<class 'int'>

- Vorzeichenbehaftete Zahlen ohne Dezimalpunkt oder Exponent.
- Positive Ganzzahlen als Literale: 3, +13456 und so weiter.
  Negative Ganzzahlen als Literale: -5, -3456 und so weiter.

Gleitkommazahl

REAL	<class 'float'>

- Zahlen, die sich einen bestimmten Wert nähren.
- Zahlen mit einem Dezimalpunkt oder Exponenten.
- 8-Byte große IEEE Fließkommazahlen (siehe http://www.iti.fh-flensburg.de/lang/informatik/ieee-format.htm)
- Literale wie zum Beispiel 3.5, 0.345667, 2.0e+24 und so weiter.

Text

TEXT	<class 'str'>

- Mit Hilfe von `Text` können alphanumerische und numerische Zeichen abgelegt werden.
- Alphanumerische und numerische Zeichen in einer bestimmten Zeichenkodierung. Zum Beispiel UTF-8, UTF-16.
- Literale werden durch ein Apostroph begrenzt.

Binäre Objekte

- Der Datentyp BLOB kann jeden beliebige Typ von Daten speichern.
- Speicherung von binären Daten.
- Speicherung von großen Datenmengen, die nicht in einem Textfeld gespeichert werden können.

Null

NULL	NONE

- Fehlende Informationen.
- Der Wert None in Python wird in der Datenbank als Null gespeichert.
- Null kann wie None nicht mit anderen Werten verglichen werden.

Attribut-Werte

- Beispiel-Aussage: Jedes Fell einer Katze hat eine Farbe.
- Katze A ist grau.Das Attribut „Fellfarbe“ ist für das Objekt „Katze A“ definiert und nicht leer. Das Attribut hat den Wert „grau“.
- Katze B hat irgendeine Fellfarbe. Das Attribut ist definiert. Aber der Wert ist nicht bekannt. Das Attribut hat eine leere Zeichenfolge ".
- Es sind keine Aussagen zum Fell von Katze C vorhanden. Zu dem Attribut „Fellfarbe“ kann keine Aussage getroffen werden. Das Attribut ist undefiniert. Das Attribut hat den Wert None.

Strings: None → Leere Zeichenkette

```
noneToString = lambda text: text or ''
```

```
def noneToString(text):
    if (text is None):
        return ''
    else:
        return str(text)
```

- Wenn der Inhalt des Datenfeldes nicht als Text interpretiert werden kann, wird ein leerer String '' zurückgegeben.tzt.

lambda-Funktion

lambda	text	:	text or ''
lambda	Argumentliste	:	Ausdruck

- Die Funktion beginnt mit dem Schlüsselwort lambda.
- Die Funktion besitzt keinen Funktionsnamen. Die Funktion ist anonym.
- Ein Verweis auf die Funktion kann in einer Variablen gespeichert werden. Über diese Variable kann die Funktion aufgerufen werden.

Argumentliste

lambda	text	:	text or ''
lambda	Argumentliste	:	Ausdruck

- Einer lambda-Funktion können beliebig viele Argumente übergeben werden.
- Die einzelnen Argumente werden durch ein Komma getrennt.

Anweisung

lambda	text	:	text or ''
lambda	Argumentliste	:	Ausdruck

- Jede anonyme Funktion hat nur einen Ausdruck.
- Entsprechend des Ausdrucks gib die Funktion einen Wert zurück. Dieser Wert kann weiterverarbeitet werden.
- In diesem Beispiel wird das übergebene Argument übergebenen, wenn es nicht den Wert None hat.

Abholung eines Datensatzes

```
try:
    strSQL = "SELECT"
    strSQL = strSQL + " AVG(tracks.Milliseconds) AS Durchschnitt"
    strSQL = strSQL + " FROM tracks;"

    with sqlite3.connect(datenbank) as connection:
        cursor = connection.cursor()
        cursor.execute(strSQL)
        data = cursor.fetchone()

        print("Der Durchschnitt aller Tracks beträgt:
                {0:.2f}".format(data[0]))
```

Erläuterung

```
with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)
   data = cursor.fetchone()
```

- Die Methode .fetchone() holt exakt einen Datensatz aus dem Resultat der Ergebnistabelle ab.
- Falls keine Datensätze vorhanden sind, wird NONE zurück geliefert.

Abholung von x Datensätzen

```
with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)

    while True:
        data = cursor.fetchmany(3)

        if data is None:
            break

        for zeile in data:
            print("Album '{0}'  -  Laufzeit von {1}
                    Millisekunden".format(zeile[0], zeile[1]))
```

Erläuterung

```
with sqlite3.connect(datenbank) as connection:
    cursor = connection.cursor()
    cursor.execute(strSQL)
    data = cursor.fetchmany(3)
```

- Die Methode .fetchmany() holt x Datensätze aus dem Resultat der Ergebnistabelle ab.
- Die Anzahl der Datensätze wird der Methode als Ganzzahl übergeben. In diesem Beispiel werden die ersten drei Datensätze geholt.
- Falls keine Datensätze vorhanden sind, wird NONE zurück geliefert.