SQLite – Nutzung in Python
Aktionsabfragen

Lösche alle Kunden, die im letzten Jahr kein Ware bestellt haben.

Neue Waren werden in den Bestand aufgenommen.

Der Preis für Waren der Kategorie „Süßwaren“ erhöhen sich um 10 %.

Aktionsabfragen

- Einfügen, aktualisieren und löschen von Datensätzen in einer Tabelle.
- Die Änderungen werden automatisiert für alle passenden Datensätze durchgeführt und gespeichert.
- Hinweis: Vor großen Änderungen an den Bestandsdaten sollte immer eine Kopie angelegt werden.

Möglichkeiten

- Aktualisierungsabfrage. Gespeicherte Attributwerte werden angepasst.
- Einfügeabfrage. Neue Datensätze werden in eine bestehende Tabelle eingefügt.
- Löschabfrage. Datensätze werden in Abhängigkeit von Kriterien aus einer Tabelle entfernt.

1. Schritt: Umbenennung der Tabelle

```
strSQL = "ALTER TABLE employees RENAME TO employee_copy"
cursor.execute(strSQL)
```

- Der Befehl ALTER TABLE kann die Tabellenstruktur einer Tabelle verändern.
- RENAME TO nennt die angegebene Tabelle in der Datenbank um.

2. Schritt: Tabelle erzeugen

```
strSQL = "CREATE TABLE employees AS"
strSQL = strSQL + " SELECT *"
strSQL = strSQL + " FROM employee_copy"
strSQL = strSQL + " WHERE HireDate IS NOT NULL;"
cursor.execute(strSQL)
```

- CREATE TABLE erzeugt eine neue Tabelle mit dem Namen Employee.
- Die Tabellenstruktur kann als Liste im Anschluss an den Tabellennamen oder mit Hilfe einer Auswahlabfrage angegeben werden.

2. Schritt: Nutzung einer Auswahlabfrage

```
strSQL = "CREATE TABLE employees AS"
strSQL = strSQL + " SELECT *"
strSQL = strSQL + " FROM employee_copy"
strSQL = strSQL + " WHERE HireDate IS NOT NULL;"
cursor.execute(strSQL)
```

- Dem Schlüsselwort AS (Wie) folgt eine Auswahlabfrage.
- Die Auswahlabfrage beginnt immer mit dem Verb SELECT.
- Die Daten und die Struktur einer Tabelle werden in eine neu zu erstellende Tabelle kopiert.

Erläuterung der Auswahlabfrage

```
strSQL = "CREATE TABLE employees AS"
strSQL = strSQL + " SELECT *"
strSQL = strSQL + " FROM employee_copy"
strSQL = strSQL + " WHERE HireDate IS NOT NULL;"
cursor.execute(strSQL)
```

- In diesem Beispiel werden alle Datenfelder (Sternchen) aus der Tabelle Employee_copy genutzt, um die Tabellenstruktur der Tabelle Employee zu erstellen.
- Anschließend werden in die neu erstellte Tabelle alle Datensätze aus der Tabelle Employee_copy kopiert, deren Einstellungsdatum (HireDate) nicht leer ist.

Aktualisierungsabfrage

```
strSQL = "UPDATE employees"
        strSQL = strSQL + " SET"
        strSQL = strSQL + " Title = 'Sales Agent'"
        strSQL = strSQL + ", Salery = 2000.00"
        strSQL = strSQL + " WHERE (Title = 'Sales Support Agent')"

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

Aufbau

UPDATE employees	Ändere diese Tabelle
SET Title = 'Sales Agent', Salery = 2000.00	Setze DatenfeldA = wert, DatenfeldB = wert
WHERE (Title = 'Sales Support Agent')	Filtere die Datensätze

Aktualisierung der Tabelle

```
strSQL = "UPDATE employees"
```

- Aktualisierungsabfragen beginnen immer mit dem Verb UPDATE.
- Dem Verb folgt die zu aktualisierende Tabelle.
- Die zu ändernden Datensätze können mit Hilfe einer WHERE-Klausel eingeschränkt werden.

Änderung der Attribut-Werte

```
strSQL = "UPDATE employees"
        strSQL = strSQL + " SET"
        strSQL = strSQL + " Title = 'Sales Agent'"
        strSQL = strSQL + ", Salery = 2000.00"
```

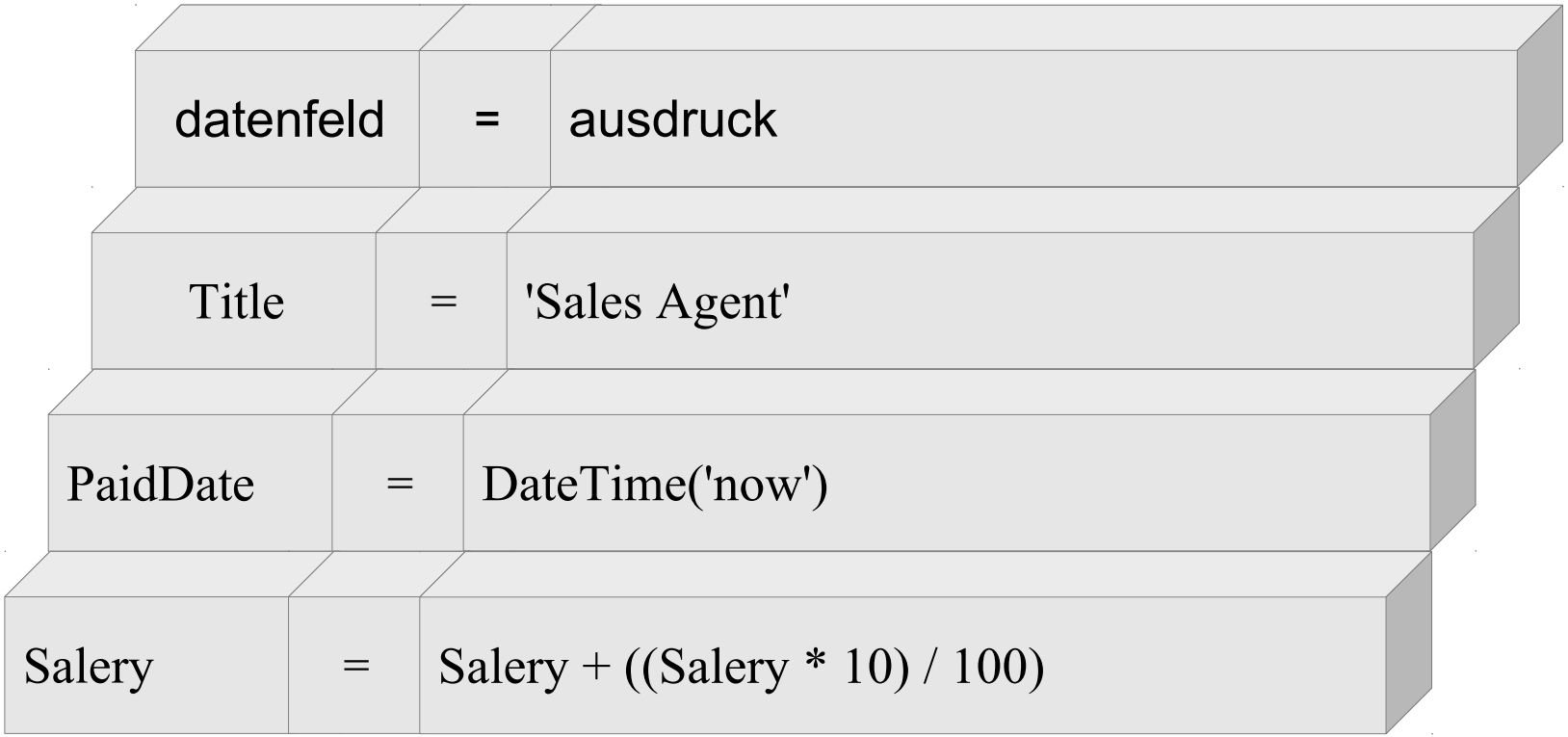
- Dem Schlüsselwort SET folgt eine Liste von Wertzuweisungen.
- Jede Wertzuweisung verändert den Wert eines Datenfeldes.
- Die Wertzuweisungen werden durch ein Komma getrennt.

Wertzuweisungen

```
strSQL = "UPDATE employees"
        strSQL = strSQL + " SET"
        strSQL = strSQL + " Title = 'Sales Agent'"
        strSQL = strSQL + ", Salery = 2000.00"
```

- Mit Hilfe des Gleichheitszeichens wird einem Datenfeld ein Wert zugewiesen.
- Das zu ändernde Datenfeld steht links vom Gleichheitszeichen. Der Name entspricht exakt dem Namen eines Datenfeldes in der angegebenen Tabelle.
- Der neue Attribut-Wert wird rechts vom Gleichheitszeichen mit Hilfe eines Ausdrucks berechnet.

Beispiele


datenfeld = ausdruck
Title = 'Sales Agent'
PaidDate = DateTime('now')
Salery = Salery + ((Salery * 10) / 100)

Nutzung von Literalen

```
strSQL = "UPDATE employees"
        strSQL = strSQL + " SET"
        strSQL = strSQL + " Title = 'Sales Agent'"
        strSQL = strSQL + ", Salery = 2000.00"
```

- Entsprechend des Datentyps des Datenfeldes werden Literale angegeben.
- Der neue Wert wird direkt rechts vom Zuweisungsoperator = angegeben.

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.

Datums- und Zeitangaben

```
FeldTimeStamp = '2009-01-01 00:00:00',
FeldDate = '2009-01-01',
FeldTime = '00:00:00',
```

- Ablage als Text.
- Datumsangaben werden durch ein Apostroph begrenzt.
- Datumsangaben werden häufig in dem Format yyyy-mm-dd angegeben.
- Zeitangaben werden in dem Format hh:mm:ss angegeben.

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.

Nutzung eines Ausdrucks

```
strSQL = "UPDATE tracks"
        strSQL = strSQL + " SET"
        strSQL = strSQL + " UnitPrice = UnitPrice + ((UnitPrice * 0.1)"
```

- Ausdrücke geben einen Wert zurück. Dieser Wert kann mit Hilfe des Gleichheitszeichens einem Datenfeld zugewiesen werden.
- Bei Datenfeldern vom Datentyp REAL oder INTEGER kann ein mathematischer Ausdruck zur Berechnung des neuen Wertes genutzt werden.

Mathematische Operatoren

Berechnung	Operator	Beispiel
Addition	[Wert] + [Wert]	9 = 7 + 2
Subtraktion	[Wert] - [Wert]	5 = 7 - 2
Multiplikation	[Wert] * [Wert]	14 = 7 * 2
Division	[Wert] / [Wert]	3.5 = 7 / 2
Modula	[Wert] % [Wert]	1 = 7 % 2

- Siehe: https://www.w3resource.com/sqlite/arithmetic-operators.php

Hinweise

- Eine Division durch Null ist nicht erlaubt.
- Es gilt die Punkt- vor Strichrechnung.
- Die Operatoren werden entsprechend ihrer Rangfolge ausgewertet. Die Rangfolge der Operatoren ist nicht standardisiert.
- Komplizierte Ausdrücke können geklammert werden.

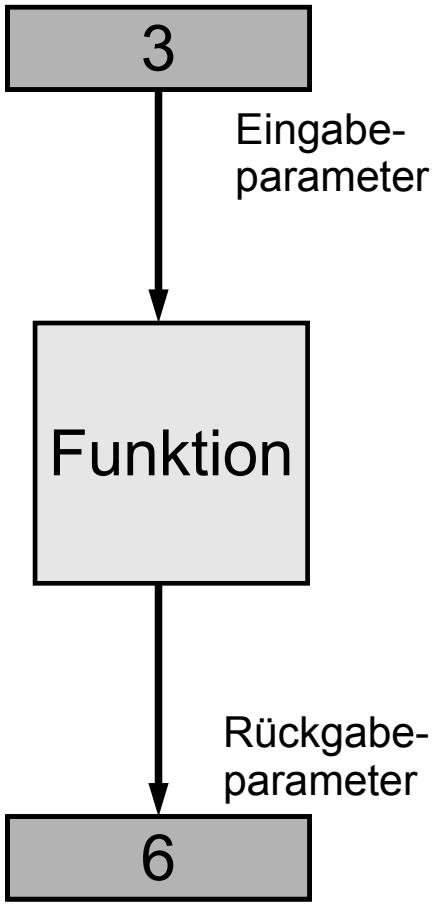
Funktionen in SQL-Anweisungen nutzen

```
strSQL = "UPDATE employees"
    strSQL = strSQL + " SET"
    strSQL = strSQL + " Salery = Round(Salery + ((Salery * 3.4) / 100))"
```

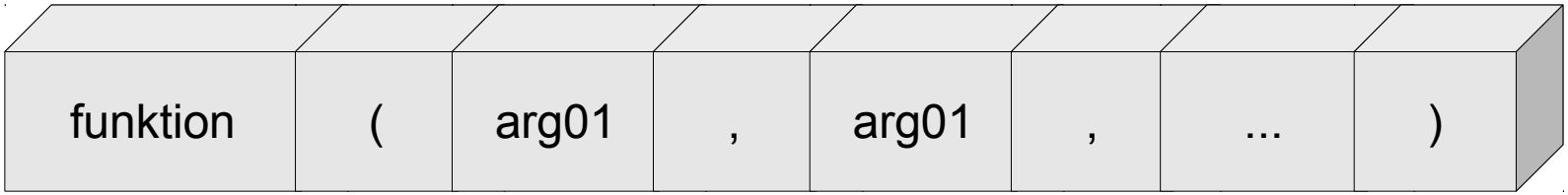
- Zur Berechnung von einem Wert können Funktionen genutzt werden.
- Die implementieren Funktionen sind abhängig vom verwendeten Datenbanksystem.
- Funktionen in SQLite: https://www.sqlite.org/lang_corefunc.html.

Funktionen in Ausdrücken

- Ein Stück Code, das eine Aufgabe beschreibt.
- Der Funktionsrumpf ist eine Blackbox für den Nutzer. Der Nutzer weiß nur wie die Funktion aufgerufen wird und welchen Wert sie zurückgibt.
- Funktionen können Argumente als Startparameter übergeben werden. Die Argumente werden entsprechend der auszuführenden Aktion verarbeitet.
- Eine Funktion gibt einen Wert zurück.



Aufbau



- Der Name der Funktion beschreibt die zu erledigende Aufgabe.
- Dem Namen der Funktion folgen die runden Klammern.
- In den runden Klammern folgt die Argumentliste. Die einzelnen Argumente werden durch ein Komma getrennt.
- Die Anzahl der Argumente variiert von Funktion zu Funktion.

Einfügeabfrage

- Beginn mit dem Verb INSERT INTO.
- Datensätze werden in eine Tabelle eingefügt.
- Attribut-Werte für alle Datenfelder oder einige Datenfelder werden in der Tabelle gespeichert.

Einfügen in die Tabelle ...

```
statement = "INSERT INTO lager
```

- Dem Verb INSERT INTO folgt der Name einer Tabelle.
- An diese Tabelle werden neue Datensätze angefügt.

Einfügung eines Datensatzes

```
statement = "INSERT INTO lager
statement = statement + " VALUES("
statement = statement + "'P-100-10'"
statement = statement + ", 'Produkt X'"
statement = statement + ", 10"
statement = statement + ", 1.5"
statement = statement + ", 2.54"
statement = statement + ");"

cursor.execute(statement)
```

Wertliste

- Die Wertliste beginnt mit dem Schlüsselwort VALUES.
- Dem Schlüsselwort folgt eine Liste von Attribut-Werten in runden Klammern.
- Die Attribut-Werte werden durch ein Komma getrennt und den Datenfeldern von links nach rechts zugeordnet. Der erste Attribut-Wert wird in der ersten Spalte und so weiter gespeichert.
- Der zu speichernde Attribut-Wert entspricht dem Datentyp des Datenfeldes.

Einfügung von Attribut-Werten

```
statement = "INSERT INTO lager(artikelNummer, artikelname)"
statement = statement + " VALUES("
statement = statement + "'P-200-10'"
statement = statement + ", 'Produkt Y'"
statement = statement + ");"

cursor.execute(statement)
```

Liste von Datenfeldern

```
statement = "INSERT INTO lager(artikelNummer, artikelname)"
```

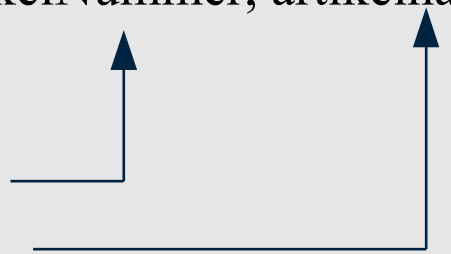
- Dem Verb INSERT INTO folgt der Name einer Tabelle.
- Dem Namen der Tabelle kann eine Liste von Datenfeldern folgen. Die Namen der Datenfelder sind in der angegebenen Tabelle definiert und werden durch ein Komma getrennt.
- Wenn alle Datenfelder befüllt werden, muss die Liste nicht angegeben werden.

Zuordnung der Attribut-Werte

```
statement = "INSERT INTO lager(artikelNummer, artikelname)"

statement = statement + " VALUES("
statement = statement + "'P-200-10'"
statement = statement + ", 'Produkt Y'"
statement = statement + ");"

cursor.execute(statement)
```



Erläuterung

- Die neuen Werte werden von links nach rechts den Felder in der Liste zugeordnet.
- Die Feldliste und die Wertliste haben die gleiche Anzahl von Elementen.
- Falls keine Feldliste angegeben ist, werden die Werte den Datenfelder in der Tabelle von links nach rechts zugeordnet.

Hinweise

- Die Werte in der Wertliste müssen entsprechend des Datentyps des Datenfeldes interpretierbar sein.
- Für Datenfelder, die als Primärschlüssel gekennzeichnet sind, muss ein eindeutiger Wert angegeben werden.

Automatisch erzeugter Primärschlüssel

```
statement = "INSERT INTO lieferant VALUES("
    statement = statement + "NULL"
    statement = statement + ", 'Firma A' , 'mail@mail.de'"
    statement = statement + ")"
cursor.execute(statement)
```

- Voraussetzung. Der Primärschlüssel ist als PRIMARY KEY AUTOINCREMENT gekennzeichnet
- Durch Angabe von NULL für den Primärschlüssel wird ein Wert automatisch erzeugt.

Weitere Möglichkeit

```
statement = "INSERT INTO lieferant(lieferantID, firmenname)"
statement = statement + " VALUES("
statement = statement +
        "(SELECT Max(lieferantID) + 1 FROM lieferant)"
statement = statement + ", 'Firma B'"
statement = statement + ")"

cursor.execute(statement)
```

Nutzung einer SQL-Anweisung

```
SELECT Max(lieferantID) + 1 FROM lieferant
```

- Mit Hilfe der Funktion Max() wird der größte Wert in dem angegebenen Datenfeld ermittelt. Dieser Wert wird mit eins addiert.
- SQL-Anweisungen zur Berechnung eines Attribut-Wertes geben einen Wert zurück.
- Die SQL-Anweisung selber muss in der INSERT INTO – Anweisung geklammert werden.

Kopieren von Struktur und Datensätzen

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

    statement = "CREATE TABLE invoices_2009 AS "
    statement = statement + strSQL
    cursor.execute(statement)
    connection.commit()
```

Kopieren von Datensätzen

```
statement = "INSERT INTO invoicesTotalPrice (invoiceDate, priceTotal)"
        statement = statement + strSQL
        cursor.execute(statement)
```

- Mit Hilfe einer Auswahlabfrage werden die zu kopierenden Datensätze gewählt.
- Die Datenfelder in der Auswahlabfrage entsprechen der Anzahl der Elemente in der Feldliste der Einfügeabfrage.
- Falls für den Primärschlüssel kein Wert angegeben wird, kann dieser in SQLite Null sein.

Löschabfrage

```
statement = "DELETE FROM artists"
        statement = statement + " WHERE( artists.ArtistId IN"
        statement = statement + "("
        statement = statement + "SELECT artists.Name"
        statement = statement + " FROM artists"
        statement = statement + " WHERE (artists.ArtistId“
        statement = statement + " NOT IN (SELECT albums.ArtistId“
        statement = statement + " FROM albums))"
        statement = statement + ")"
        statement = statement + ")"
```

Löschen von allen Datensätzen

```
statement = "DELETE FROM artists"
```

- Die Aktionsabfrage beginnt mit dem Schlüsselwort `DELETE`.
- Dem Schlüsselwort `FROM` folgt der Tabellenname aus dem die Datensätze gelöscht werden.
- Die Tabelle wird geleert. Alle Datensätze aus der Tabelle werden entfernt.

Löschen von einigen Datensätzen

```
statement = "DELETE FROM artists"
        statement = statement + " WHERE( artists.ArtistId IN"
        statement = statement + "("
        statement = statement + "SELECT artists.Name"
        statement = statement + " FROM artists"
        statement = statement + " WHERE (artists.ArtistId“
        statement = statement + " NOT IN (SELECT albums.ArtistId“
        statement = statement + " FROM albums))"
        statement = statement + ")"
        statement = statement + ")"
```

Erläuterung

- In Abhängigkeit einer WHERE-Klausel können die zu löschenden Datensätze ausgewählt werden.
- Alle Datensätze auf die, die angegebene Bedingung zu trifft, werden gelöscht.
- In diesem Beispiel werden alle Künstler gelöscht, zu denen kein Album in der Datenbank gespeichert ist.

Speicherung der Änderung

```
connection.commit()
```

- Änderungen etc. werden in der Datenbank gespeichert.
- Falls die Methode nicht aufgerufen wird, werden Änderungen in anderen Datenbank-Verbindungen nicht sichtbar.

… und zurückrollen

```
connection.rollback()
```

- Alle Änderungen bis zum letzten .commit() werden zurückgerollt.