Leibniz Universität IT Services

Leibniz Universität Hannover

SQLite – Nutzung in Python

Tabellen in einer Datenbank

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.

Pragma-Befehle

```
cursor.execute("PRAGMA table_info('genres')")
print(cursor.fetchall())
```

- Mit Hilfe des Befehls PRAGMA können Informationen zum Datenbank-Schema ausgelesen werden.
- Dem Befehl folgt ein Schlüsselwort. Das Schlüsselwort beschreibt welche Informationen gewünscht werden.

Informationen zur Struktur einer Tabelle

```
cursor.execute("PRAGMA table_info('genres')")
print(cursor.fetchall())
```

- Das Schlüsselwort table_info listet alle Datenfelder einer Tabelle auf.
- Informationen zu den Spalten einer Tabelle werden aufgelistet.
- Dem Schlüsselwort table_info wird als Argument ein Tabellennamen als String übergeben.
- In diesem Beispiel werden Informationen zu der Struktur der Tabelle genres ausgegeben.

Erzeugung einer neuen Tabelle

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

    statement = "CREATE TABLE IF NOT EXISTS lager ("
    statement = statement + " artikelNummer TEXT PRIMARY KEY"
    statement = statement + ", artikelname TEXT NOT NULL UNIQUE"
    statement = statement + ", mindestbestand INTEGER NOT NULL"
    statement = statement + ", einkaufspreis REAL NOT NULL"
    statement = statement + ", verkaufspreis REAL NOT NULL"
    statement = statement + ")"
    cursor.execute(statement)
```

SQL-Befehl „Create table“

```
statement = "CREATE TABLE lager("
statement = statement + ")"
```

- Der SQL-Befehl CREATE TABLE erzeugt eine neue Tabelle.
- Dem SQL-Befehl folgt der Name der Tabelle. Der Name ist eindeutig in der Datenbank.
- Für jede Tabelle in einer relationalen Datenbank muss mindestens ein Datenfeld existieren. Die Datenfelder der neuen Tabelle werden in den runden Klammern angegeben.

Erzeuge nur, wenn diese nicht existiert ...

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

- Wenn (IF) die Tabelle nicht (NOT) existiert (EXISTS) …
- Wenn die Bedingung nicht genutzt wird, wird eine Ausnahme ausgegeben, dass die neu anzulegende Tabelle schon vorhanden ist.

Tabellenname

- Der Name hat einen Bezug zu den darin abgelegten Informationen.
- Der Name spiegelt die, in der Tabelle abgelegten Objekte aus der realen Welt wieder.
- Gute Namen sind einfach zu lesen, da sie keine geheimnisvollen Abkürzungen enthalten.
- Namen sollten nur aus einem Sprachraum kommen.

Datenfelder

- Datenfelder enthalten exakt eine Information. Datenfelder enthalten exakt ein Attribut eines Objektes aus der realen Welt.
- Datenfelder in einer Tabelle haben einen eindeutigen Namen. Die Namen sind „Spaltenüberschriften“ in einer Tabelle.
- Datenfelder speichern einen bestimmten Typ von Informationen.

Definition von Datenfeldern

```
statement = "CREATE TABLE lager ("
    statement = statement + " artikelNummer TEXT PRIMARY KEY"
    statement = statement + ", artikelname TEXT NOT NULL UNIQUE"
    statement = statement + ", mindestbestand INTEGER NOT NULL"
    statement = statement + ", einkaufspreis REAL NOT NULL"
    statement = statement + ", verkaufspreis REAL NOT NULL"
    statement = statement + ")"
```

- Die Liste aller Datenfelder wird mit Hilfe der runden Klammern zusammengefasst. Die Liste folgt direkt dem Tabellennamen.
- Die Felddefinitionen in der Liste werden durch ein Komma getrennt.

Felddefinitionen

```
statement = "CREATE TABLE lager ("
    statement = statement + " artikelNummer TEXT PRIMARY KEY"
```

- Die Felddefinition beginnt mit dem Namen des Datenfeldes.
- Dem Namen folgt der Datentyp. Welche Art von Daten können in dem Feld gespeichert werden?
- Der Inhalt der Datenfelder kann mit Hilfe von Constraints eingeschränkt werden.

Feldnamen

- Feldnamen in einer Tabelle sind eindeutig. Sie identifizieren eine Spalte in einer Tabelle.
- Feldnamen spiegeln den Inhalt des Datenfeldes wieder. Der Name sollten immer auf das Attribut in der realen Welt Bezug nehmen.

Hinweise zu Bezeichnern

- Der Feldname oder der Tabellenname beginnt mit einem Buchstaben.
- Ein Bezeichner besteht aus den lateinischen Groß- und Kleinbuchstaben und den Ziffern.
- Ein Bezeichner enthält als Sonderzeichen nur den Unterstrich.
- Ein Bezeichner sollte nicht länger als 18 Zeichen sein. Die maximale Länge ist abhängig vom verwendeten System.
- SQL-Befehle können nicht als Bezeichner genutzt werden.

Datentypen eines Feldes

```
statement = "CREATE TABLE lager ("
    statement = statement + " artikelNummer TEXT PRIMARY KEY"
```

- Dem Namen des Datenfeldes folgt der Datentyp.
- Der Datentyp und der Name des Feldes werden durch ein Leerzeichen getrennt.
- Die Implementierung der verschiedenen Datentypen ist abhängig von dem genutzten Datenbanksystem.

In Abhängigkeit des Datentyps wird ...

- der Speicherbedarf berechnet. Die Größe des Containers, in der der Attribut-Wert abgelegt wird, wird festgelegt.
- der Wertebereich für ein Attribut-Wert festgelegt.
- die Nutzung des Datenfeldes definiert. Der Datentyp `Text` kann nicht in Berechnungen genutzt werden.

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.

Möglichkeiten

- `INTEGER`. Ganzzahl.
- `REAL`. Gleitkommazahlen.
- `TEXT`. String.
- `BLOB`. Binary Large Object.
- `NULL`.

Ganzzahlen

```
statement = "CREATE TABLE lager ("
        statement = statement + "mindestbestand INTEGER"
```

- Vorzeichenbehaftete Zahlen ohne Dezimalpunkt oder Exponent.
- Positive Ganzzahlen als Literale: 3, +13456 und so weiter. Negative Ganzzahlen als Literale: -5, -3456 und so weiter.
- Die Klasse `int` oder `long` in Python wird in der Storage Class `INTEGER` gespeichert.

Gleitkommazahl

```
statement = "CREATE TABLE lager ("
        statement = statement + "einkaufspreis REAL"
```

- 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.
- Die Klasse float in Python wird in der Storage Class REAL gespeichert.

Text

```
statement = "CREATE TABLE lager ("
    statement = statement + " artikelNummer TEXT"
```

- 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.
- Die Klasse str oder unicode in Python wird in der Storage Class TEXT gespeichert.

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.

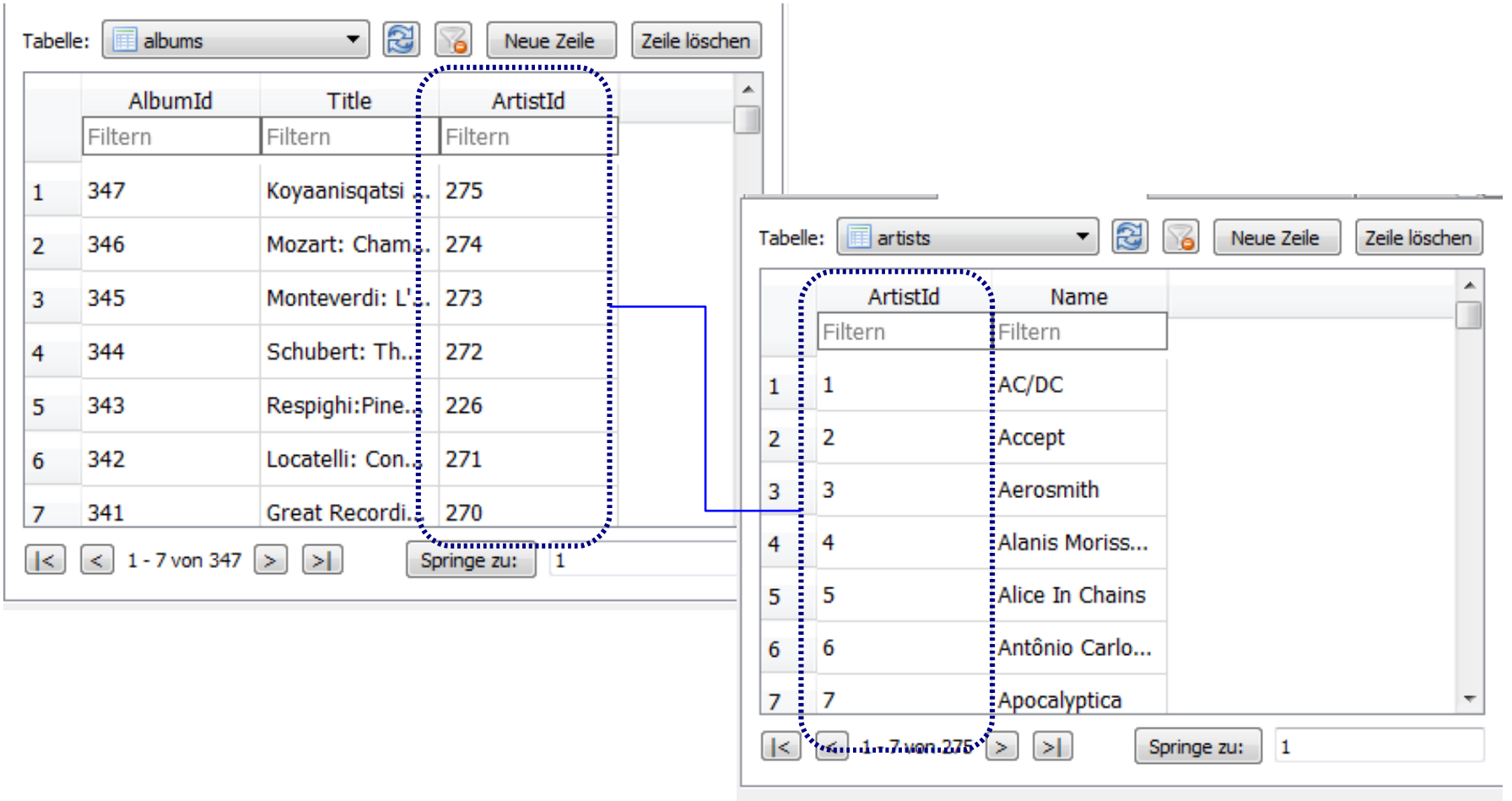
Null

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

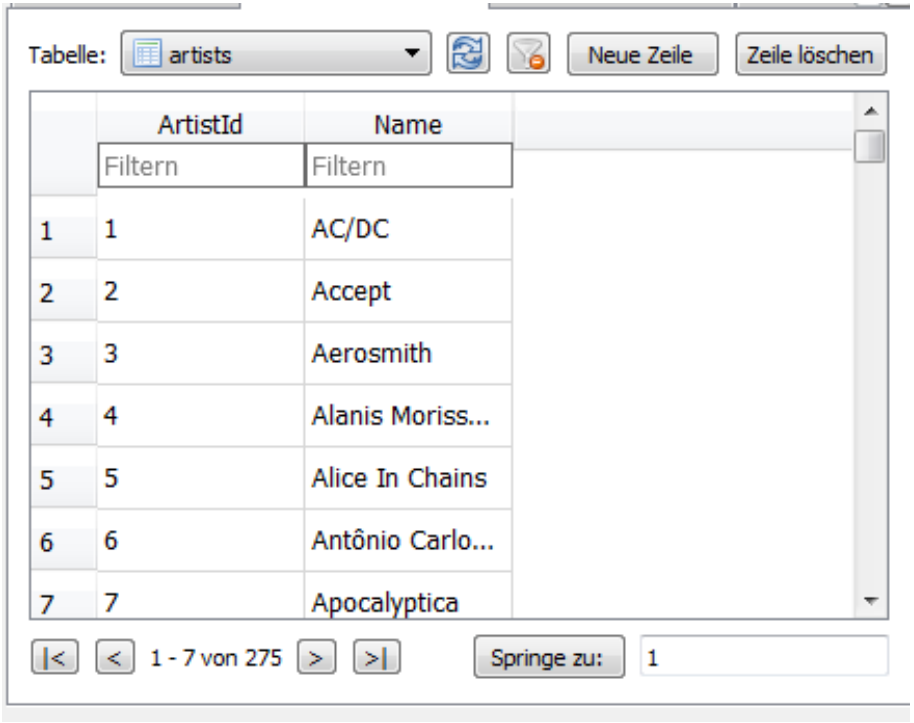
Schlüsselwerte in Tabellen

- Jeder Datensatz in einer Tabelle kann mit Hilfe eines Schlüssels eindeutig identifiziert werden.
- Jede Zeile unterscheidet sich von allen anderen in der Tabelle durch den Schlüsselwert. Der Schlüssel ist eindeutig.
- In Datenbanken werden häufig künstliche Schlüssel genutzt.
- Schlüssel sind zum Beispiel: Artikelnummer, Kundennummer.

Nutzung von Schlüsselwerten

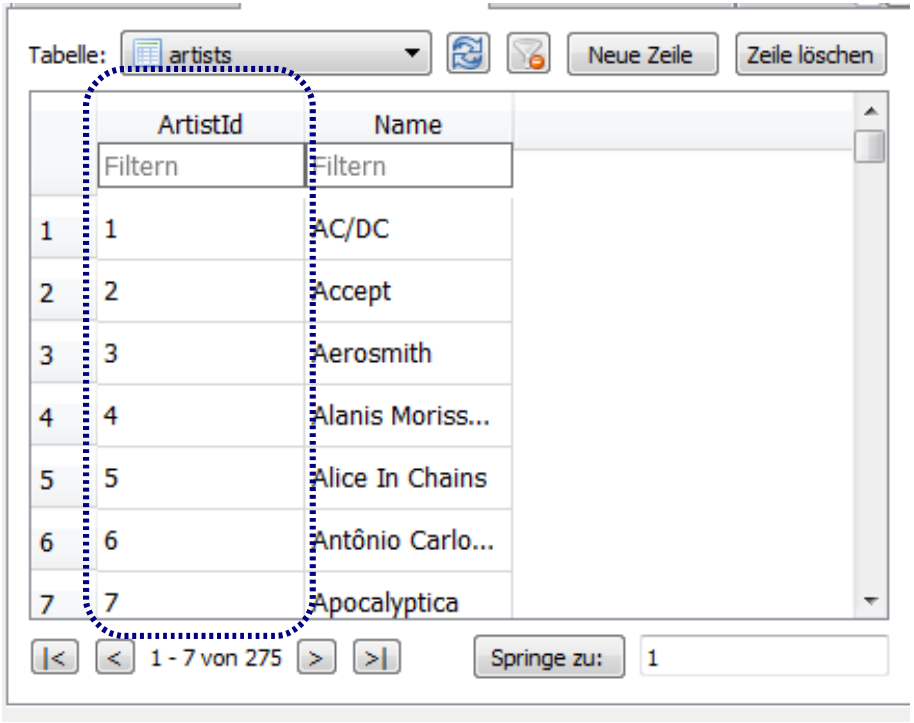


Master-Tabelle (Primärtabelle)



- Abbildung der Hauptdatensätze.
- Jeder Datensatz wird durch einen Primärschlüssel identifiziert.
- Die oberste Tabelle in der Hierarchie enthält keine Schlüsselwerte, die auf andere Tabellen verweisen.

Primärschlüssel in einer Tabelle



- Eindeutige Identifizierung eines Datensatzes in einer Tabelle.
- Der Schlüsselwert wird sofort bei der Anlage des Datensatzes vergeben.
- Während der Existenz des Datensatzes wird der Schlüssel niemals geändert.

… in einer SQL-Anweisung definieren

```
statement = "CREATE TABLE lager ("
    statement = statement + " artikelNummer TEXT PRIMARY KEY"
```

- Die Angabe PRIMARY KEY kennzeichnet den Primärschlüssel einer Tabelle.
- Schlüsselwerte sind vom Datentyp TEXT oder INTEGER.

Nutzung eines Autowerts

```
statement = "CREATE TABLE IF NOT EXISTS  lieferant ("
   statement = statement + " lieferantID INTEGER
                                    PRIMARY KEY  AUTOINCREMENT"
```

- Die Angabe PRIMARY KEY kennzeichnet den Primärschlüssel einer Tabelle.
- Der Schlüsselwert ist vom Datentyp INTEGER. Das Schlüsselwort AUTOINCREMENT kennzeichnet einen Zähler als Schlüsselwert.
- Bei Eingabe eines neuen Datensatzes wird automatisch ein entsprechender eindeutiger Schlüsselwert gesetzt.

Hinzufügung eines Datensatzes

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

Erläuterung

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

- INSERT INTO tabelle VALUES.
- Dem Schlüsselwort VALUES folgt eine Liste von Werten. Die Werte werden durch ein Komma in der Liste getrennt.
- Die angegebenen Werte werden von links nach rechts den entsprechenden Datenfeldern zugeordnet.

Setzen des Primärschlüssels

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

- Das Schlüsselwort AUTOINCREMENT nutzt eine Ganzzahl. Als Schlüsselwert kann als Wert eine Ganzzahl in der Liste angegeben werden. Die Ganzzahl ist eindeutig.
- Durch Angabe von NULL als Schlüsselwert wird automatisiert eine freie Zeilennummer als Primärschlüssel genutzt.

Zusammengesetzte Schlüsselwerte

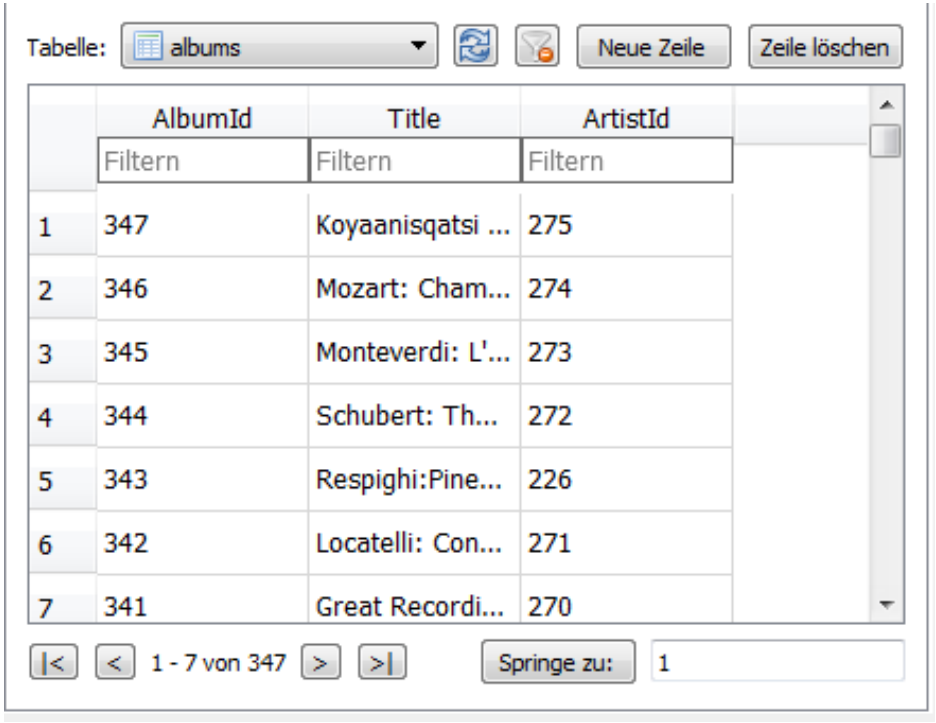
```
statement = "CREATE TABLE IF NOT EXISTS bestellposten ("
    statement = statement + " bestellnummer  TEXT NOT NULL"
    statement = statement + ", postenNr INTEGER NOT NULL"
    statement = statement + ", lagerID TEXT NOT NULL"
    statement = statement + ", bestellmenge INTEGER NOT NULL"
    statement = statement + ", einzelpreis REAL NOT NULL"
    statement = statement + ", PRIMARY KEY(bestellnummer, postenNr)"
    statement = statement + ")"
 cursor.execute(statement)
```

Zusammengesetzte Schlüsselwerte

```
statement = statement + ", PRIMARY KEY(bestellnummer, postenNr)"
```

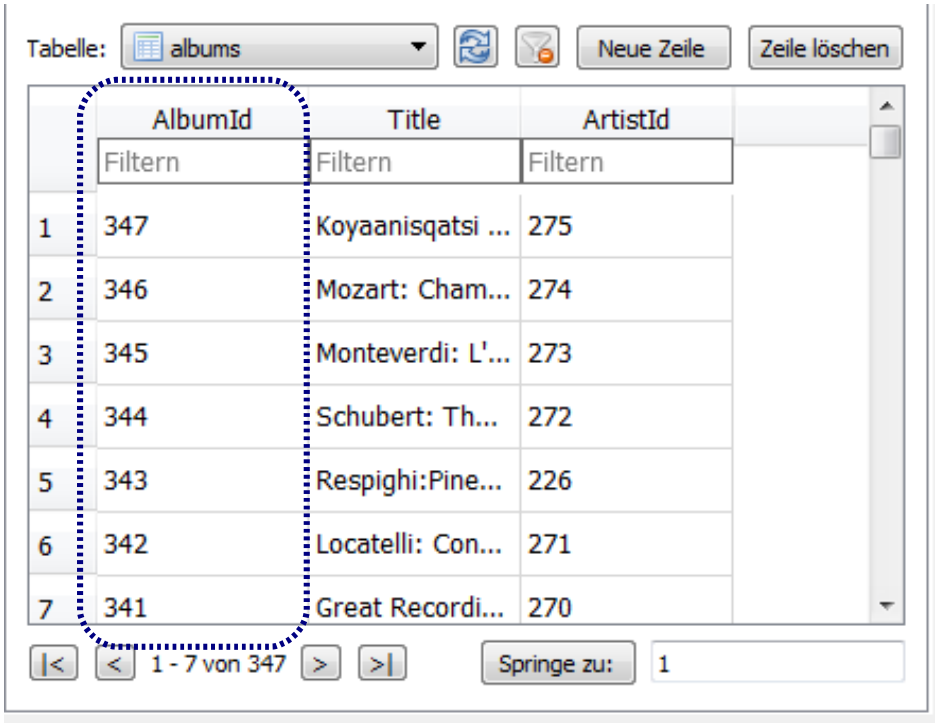
- Der Primärschlüssel PRIMARY KEY wird am Ende der Felddefinition definiert.
- Dem Schlüsselwort folgt direkt eine Liste von Feldnamen. Die Liste wird mit Hilfe der runden Klammern begrenzt. Die einzelnen Elemente in der Liste werden durch ein Komma getrennt.
- Der, aus den angegebenen Feldern zusammengesetzte Wert muss eindeutig sein.

Detail-Tabelle



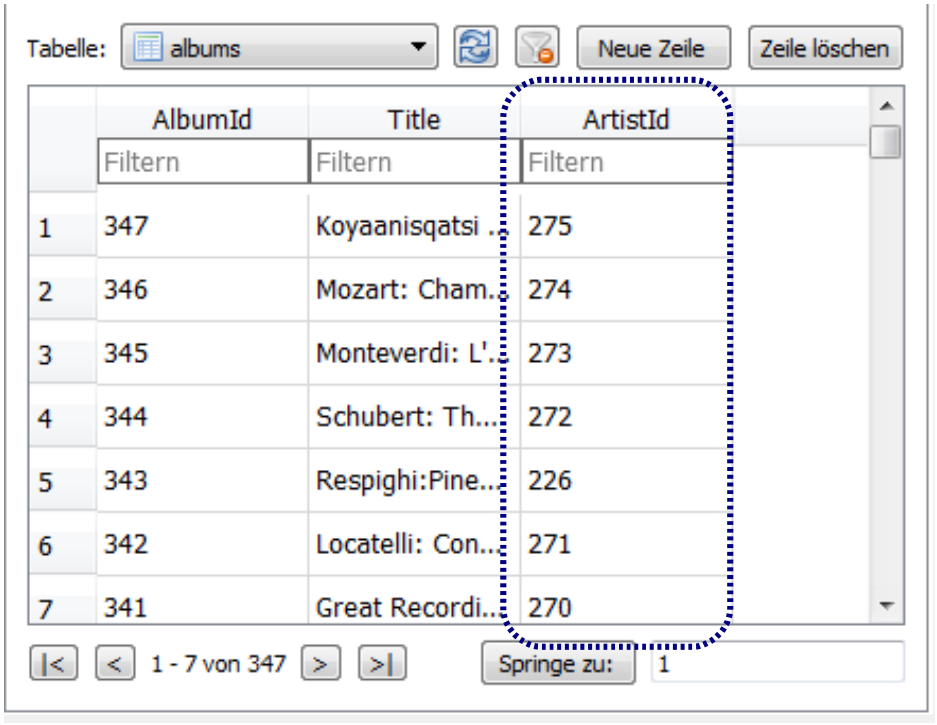
- Abbildung der Detail-Datensätze.
- Nutzung eines Fremdschlüssel, um auf eine andere Tabelle zu verweisen.
- Eine Detail-Tabelle ist das Kind einer Master-Tabelle.

Primärschlüssel



- Jede Detail-Tabelle hat einen Primärschlüssel, der die Datensatz in der Tabelle eindeutig identifiziert.
- Eine Detail-Tabelle kann auch wieder Master-Tabelle sein. Der Primärschlüssel dieser Tabelle wird in einer anderen untergeordneten Tabelle genutzt.

Fremdschlüssel



- Verweis auf einen Datensatz in einer Master-Tabelle.
- Der Primärschlüssel der Master-Tabelle kommt beliebig oft als Fremdschlüssel in einer Detail-Tabelle vor.
- Primär- und Fremdschlüssel sollten vom gleichen Datentyp sein.

… in einer SQL-Anweisung

```
statement = "CREATE TABLE IF NOT EXISTS bestellposten ("
    statement = statement + " bestellnummer  TEXT"
    statement = statement + " NOT NULL"
    statement = statement + " references bestellung(bestellung)"
    statement = statement + " on delete restrict on update cascade"
    statement = statement + " deferrable initially deferred"

    statement = statement + ", PRIMARY KEY(bestellnummer, postenNr)"
    statement = statement + ")"
    cursor.execute(statement)
```

Verweis auf den Primärschlüssel

```
statement = "CREATE TABLE IF NOT EXISTS bestellposten ("
    statement = statement + " bestellnummer  TEXT"
    statement = statement + " NOT NULL"
    statement = statement + " references bestellung(bestellung)"
```

- references mastertabelle(Feldname).
- Der Fremdschlüssel verweist auf ein Feld (bestellung) in der Master-Tabelle (bestellung).
- Beide Felder sollten den gleichen Datentyp besitzen.
- Der Fremdschlüssel ist zwingend erforderlich (NOT NULL).

Referentielle Integrität

- Regeln beim Löschen und Aktualisieren von Datensätzen.
- Sind die Beziehungen zwischen den Tabellen korrekt?
- Existiert zu einem Fremdschlüssel in einer Detail-Tabelle auch ein Primärschlüssel in der Master-Tabelle?

… in der SQL-Anweisung

```
statement = "CREATE TABLE IF NOT EXISTS bestellposten ("
    statement = statement + " bestellnummer  TEXT"
    statement = statement + " NOT NULL"
    statement = statement + " references bestellung(bestellung)"
    statement = statement + " on delete restrict on update cascade"
```

- Beim Löschen (on delete) oder bei Änderung (on update) des Primärschlüssels in der Master-Tabelle.
- In diesem Beispiel wird die Löschung des Primärschlüssels abgebrochen, falls ein Fremdschlüssel vorhanden ist.
- In diesem Beispiel werden automatisiert alle Fremdschlüssel zu einem Primärschlüssel verändert, wenn dieser sich verändert.

Möglichkeiten

- on delete | update set null. Bei Löschung oder Änderung des Primärschlüssel werden alle dazugehörigen Fremdschlüssel auf den Wert NULL gesetzt.
- on delete | update set default. Bei Löschung oder Änderung des Primärschlüssel werden alle dazugehörigen Fremdschlüssel auf den Standardwert gesetzt.

Möglichkeiten

- on delete | update cascade. Bei Löschung werden alle Datensätze mit einem dazugehörigen Fremdschlüssel entfernt. Bei einer Änderung werden die dazugehörigen Fremdschlüssel angepasst.
- on delete | update restrict. Eine Löschung oder Änderung von Primärschlüsseln, die als Fremdschlüssel verwendet werden, wird abgebrochen.
- on delete | update no action. Es wird nur ein Fehler gemeldet.

Integritätsregeln einschalten

```
statement = "CREATE TABLE IF NOT EXISTS bestellposten ("
    statement = statement + " bestellnummer  TEXT"
    statement = statement + " NOT NULL"
    statement = statement + " references bestellung(bestellung)"
    statement = statement + " on delete restrict on update cascade"
    statement = statement + " deferrable initially deferred"
```

- Standardmäßig ist die referentielle Integrität in den Pragmas nicht eingeschaltet.
- Die Anweisung `deferrable initially deferred` schaltet für den angegebenen Fremdschlüssel die referentielle Integritätsregel ein.

Umbenennung einer Tabelle

```
statement = "ALTER TABLE land RENAME TO laender"
cursor.execute(statement)
```

- ALTER TABLE table. Verändere die angegebene Tabelle.
- table RENAME TO neuTable. Umbenenne die Tabelle.
- Hinweis: Der alte Tabellenname muss auch dort geändert werden, wo dieser im Programm verwendet wird.

Existiert die Tabelle?

```
blnFound = False

if (not((cursor is None) or (cursor  == ''))):

    if (not((tblName is None) or (tblName == ''))):
        strSQL = "SELECT name FROM sqlite_master WHERE (type='table')"
        strSQL = strSQL + " and (tbl_name = '" + tblName + "')"
        strSQL = strSQL + ";"

        cursor.execute(strSQL)
        result = cursor.fetchall()

        if len(result) > 0:
            blnFound = True
```

Hinzufügung von Datenfeldern

```
statement = "ALTER TABLE kontakt"
  statement = statement + " ADD COLUMN"
  statement = statement + " strasse TEXT"
  cursor.execute(statement)
```

- ALTER TABLE table. Verändere die angegebene Tabellenstruktur.
- ADD COLUMN feld datentyp. Füge das Datenfeld hinzu.
- Hinweis: Die Definition des Datenfeldes entspricht der Definition in der Anweisung CREATE TABLE.

Existiert das Datenfeld?

```
def existsColumn(cursor, tblName, fieldname):

    cursor.execute("PRAGMA table_info('" + tblName + "')")
    result = cursor.fetchall()
    blnFound = False

    for rows in result:
        if (rows[1] == fieldname):
                blnFound = True

    return blnFound
```

Löschen von Tabellen

```
statement = "DROP TABLE IF EXISTS laender"
    cursor.execute(statement)
```

- DROP TABLE IF EXISTS table.
- Wenn die Tabelle in der Datenbank existiert, lösche diese.