Leibniz Universität IT Services

Leibniz Universität Hannover

SQLite – Nutzung in Python

Tabellen in einer Datenbank

Constraint

- Definition von Regeln für die Eingabe von Daten
- Einschränkungen bei der Eingabe von Daten.
- Falls die Einschränkungen verletzt werden, kann der Datensatz nicht in der Tabelle gespeichert werden.

… in Bezug auf ein Datenfeld

```
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 + ")"
```

Erläuterung

- Felddefinition: feldname datentyp constraint constraint ...
- Die Dateneingabe eines Datenfeldes wird eingeschränkt.
- Die Nichterfüllung einer Einschränkung verhindert die Speicherung des Datensatzes.

Möglichkeiten

- not null. Die Eingabe ist erforderlich.
- unique. Der Wert ist einmalig in der Tabelle.
- primary key. Definition des Primärschlüssels.
- foreign key. Definition des Fremdschlüssels.
- check.
- collate.

… Bezug auf die Tabelle

```
statement = "CREATE TABLE IF NOT EXISTS  lieferant ("
    statement = statement + " lieferantID INTEGER PRIMARY KEY
                                                AUTOINCREMENT"
    statement = statement + ", firmenname TEXT NOT NULL UNIQUE"
    statement = statement + ", email TEXT NOT NULL UNIQUE"
    statement = statement + ", UNIQUE(firmenname, email)"
    statement = statement + ")"
```

Erläuterung

- constraint(feldname, feldname, ...)
- Die Einschränkungen werden am Ende der Tabelle definiert.
- Mit Hilfe eines Schlüsselwortes wird die Einschränkung beschrieben. In diesem Beispiel sind die angegebenen Felder einmalig.
- Dem Schlüsselwort folgt eine Liste von Feldnamen. Die Feldnamen werden durch ein Komma getrennt. Die Einschränkung bezieht sich auf die Felder in der angegebenen Kombination.

Möglichkeiten

- unique. Der Wert ist einmalig in der Tabelle.
- primary key. Definition des Primärschlüssels.
- check.

Zwingend erforderliche Eingabe

```
statement = "CREATE TABLE lager ("
    statement = statement + "artikelname TEXT NOT NULL"
    statement = statement + ", mindestbestand INTEGER NOT NULL"
    statement = statement + ", einkaufspreis REAL NOT NULL"
```

- Dem Feldnamen folgt die die Angabe NOT NULL.
- Die Daten werden durch die Angabe eingeschränkt. Das Feld darf nicht (NOT) leer (NULL) sein.
- Die Eingabe in dieses Datenfeld ist zwingend erforderlich.

Einzigartiger Wert

```
statement = statement + ", artikelname TEXT NOT NULL UNIQUE"
statement = statement + ", UNIQUE(firmenname, email)"
```

- Jeder Wert in dem gekennzeichneten Datenfeldern kommt einmal in der Tabelle vor.
- Das Datenfeld enthält für jeden Datensatz einen anderen Wert. Der Wert ist einmalig.
- Primärschlüssel müssen eindeutig sein. Der Wert kommt nur einmal in einer Tabelle vor.

Hinweise

- Das Constraint UNIQUE folgt immer dem Constraint DEFAULT.
- Null-Werte können beliebig oft vorkommen. Null-Werte sind nie zu einem anderen Wert gleichartig.

Standardwerte

```
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 DEFAULT 1"
    statement = statement + ", einkaufspreis REAL
                                                  NOT NULL DEFAULT 1.0"
    statement = statement + ", verkaufspreis REAL
                                                NOT NULL DEFAULT 1.0"
    statement = statement + ")"
    cursor.execute(statement)
```

Erläuterung

- Falls beim Einfügen des Datensatzes kein Wert für das Feld angegeben ist, wird der Standardwert genutzt.
- Der Standardwert wird entsprechend des Datentyps des Feldes angegeben.

Beispiel für den Datentyp

- REAL: DEFAULT 1.0
- INTEGER: DEFAULT 1
- TEXT: DEFAULT 'UNKOWN'

… für Datums- und Zeitwerte

```
statement = "CREATE TABLE IF NOT EXISTS  bestellung ("
    statement = statement + " bestellnummer  TEXT PRIMARY KEY  "
    statement = statement + ", lieferantID INTEGER NOT NULL"
    statement = statement + ", bestelldatum TEXT
                                                DEFAULT current_timestamp"
    statement = statement + ")"
    cursor.execute(statement)
```

Möglichkeiten

- Das Schlüsselwort current_date gibt einen Datumswert in der Form YYYY-MM-DD zurück.
- Das Schlüsselwort current_time gibt einen Zeitwert in der Form HH-MM-SS zurück.
- Das Schlüsselwort current_timestamp gibt eine Kombination aus den Schlüsselwörtern current_date und current_time zurück.

Überprüfung von Eingaben

```
statement = "CREATE TABLE IF NOT EXISTS  lieferant ("
    statement = statement + " lieferantID INTEGER
                                    PRIMARY KEY  AUTOINCREMENT"
    statement = statement + ", firmenname TEXT NOT NULL""
    statement = statement + ", Postleitzahl TEXT"
    statement = statement + ", Ort TEXT"
    statement = statement + ", email TEXT NOT NULL UNIQUE"
    statement = statement + ", UNIQUE(firmenname, email)"
    statement = statement + ", CHECK (length(Postleitzahl) = 5)"
    statement = statement + ")"
    cursor.execute(statement)
```

Beispiel

```
statement = statement + ", CHECK (length(Postleitzahl) = 5)"
```

- Dem Befehl `CHECK` folgt in runden Klammern eine Bedingung.
- Eine Bedingung bildet Ja / Nein-Fragen ab. Wenn die Bedingung erfüllt (wahr, true) ist, wird der Datensatz gespeichert. Andernfalls wird ein Fehler gemeldet.

Bedingung

```
length(Postleitzahl) = 5
```

- [ausdruck] [vergleichsoperator] [wert].
- Datenfeld wird mit einem Literal mit Hilfe von Operatoren verglichen.
- Einer Funktion wird der Wert eines Datenfeldes übergeben. Der Rückgabewert der Funktion wird mit einem Literal verglichen.
- Mehrere Bedingungen können mit Hilfe der Operatoren and, or und not verknüpft werden..

Vergleichsoperatoren

ist ...	Operator
gleich	=
ungleich	<>
	!=
kleiner	<
kleiner gleich	<=
größer	>
größer gleich	>=

Verknüpfungsoperatoren

- `[Bedingung] OR [Bedingung]`. Eine der Bedingungen muss zu treffen.
- `[Bedingung] AND [Bedingung]`. Beide Bedingungen müssen zu treffen.
- `NOT([Bedingung])`. Die Bedingungen wird negiert.

Nutzung von Funktionen

```
length(Postleitzahl) = 5
```

- Jede Funktion hat einen Namen. Dieser ist eindeutig. Mit Hilfe des Namens wird eine Funktion aufgerufen.
- Jede Funktion hat eine Parameterliste. Die Parameterliste kann leer sein. Die Liste kann beliebig viele Parameter enthalten. In diesem Beispiel wird der Funktion ein Parameter (der Name eines Datenfeldes) übergeben.
- Eine Funktion kann einen Rückgabewert besitzen. Dieser Rückgabewert kann in Bedingungen genutzt werden. In diesem Beispiel wird die Textlänge des Datenfeldes Postleitzahl zurückgegeben.

Hinweise

- Funktionen in SQL werden immer in Abhängigkeit des Datenbanksystems implementiert.
- Implementierte Funktionen in SQLite: https://www.sqlite.org/lang_corefunc.html aufgelistet.

Vergleich von Text

```
statement = "CREATE TABLE IF NOT EXISTS  lieferant ("
   statement = statement + " lieferantID INTEGER PRIMARY KEY"
   statement = statement + ", firmenname TEXT NOT NULL"
   statement = statement + ", strasse TEXT"
   statement = statement + ", hausnummer TEXT"
   statement = statement + ", Postleitzahl TEXT"
   statement = statement + ", Ort TEXT"
   statement = statement + ", email TEXT NOT NULL UNIQUE collate nocase"
   statement = statement + ", UNIQUE(firmenname, email)"
   statement = statement + ", CHECK (length(Postleitzahl) = 5)"
   statement = statement + ")"
   cursor.execute(statement)
```

Vergleichsmethoden

```
statement = statement + ", email TEXT NOT NULL UNIQUE collate nocase"
```

- binary. Text wird Byte für Byte verglichen. Standardeinstelllung.
- nocase. Unterscheidung von Groß- und Kleinschreibung bei den 26 ASCII-codierten lateinischen Buchstaben.
- reverse.