

Einschränkungen (Constraints)

Die meisten DBMS bieten eine Möglichkeit, den zulässigen Wertebereich für eine Spalte durch Constraints (Einschränkung) einzuschränken. Die Regel kann mit CHECK wahlweise direkt bei der Deklaration der Spalte oder als eigener Codeblock nach den Spalten angegeben werden.

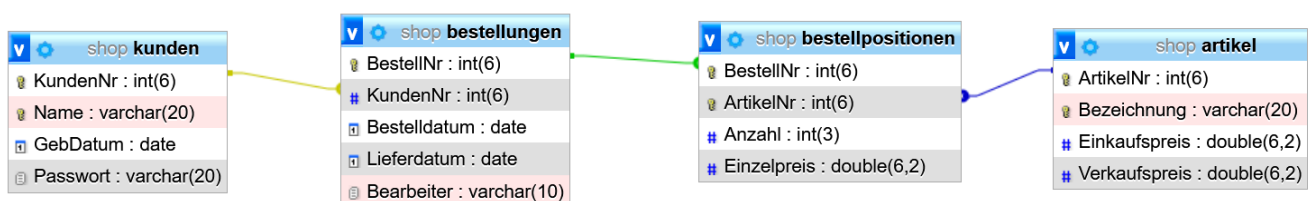
In MySQL werden CHECK-Regeln erst ab Version 8.0.16 berücksichtigt, in MariaDB ab Version 10.2.1.

NOT NULL	Das Attribut muss bei jedem Datensatz einen Wert enthalten.
CHECK	Einschränkung des Wertebereichs.
DEFAULT	Standard-Werte, falls beim Speichern nichts angegeben wird.
UNIQUE	Die angegebenen Attribute sind eindeutig, es dürfen keine doppelten Werte vorkommen.

Beispiel:

```
CREATE TABLE Artikel
(
  ArtikelNr          INT(6)  AUTO_INCREMENT,
  Bezeichnung        VARCHAR(20) NOT NULL UNIQUE DEFAULT 'Elektro',
  Einkaufspreis      DOUBLE(6,2) CHECK (Einkaufspreis >= 0),
  Verkaufspreis      DOUBLE(6,2) CHECK (Verkaufspreis >= 0),
  PRIMARY KEY (ArtikelNr),
  CHECK (Verkaufspreis >= Einkaufspreis)
);
```

- Legen Sie die Datenbank *shop* in phpmyadmin mit der Datei *Shop.sql* an.
- Fügen Sie mit Hilfe des Python-Programms *mysql_artikel_einfügen.py* Datensätze in die Tabelle *artikel* ein.



Aufgabe:

Fügen Sie mit Hilfe des Python-Programms *mysql_artikel_einfügen.py* einen Datensatz in die Tabelle *kunde* ein, indem Sie das Script *mysql_artikel_einfügen.py* umschreiben. Die Datensätze können mit dem Script *mysql_anzeigen.py* angezeigt werden.

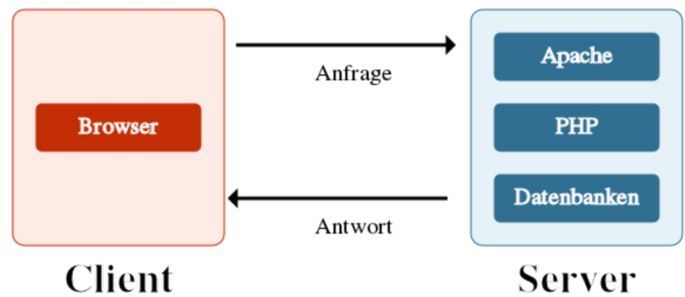
Der neu hinzugefügte Kunde hat einen der Artikel bestellt. Speichern Sie die Verknüpfungsinformationen (Fremdschlüssel) in die Zwischentabellen. Arbeiten Sie dazu mit SQL-Variablen und dem insert – Befehl.

Flexibler ist es, wenn neue Daten über Eingabefelder in die Tabelle *kunden* eingegeben werden können (*name = input("Name: ")*): → *kunden_einfügen_input.py*

SQL Injections

Demonstration: <https://www.hacksplaining.com/prevention/sql-injection>

Ein Angreifer kann über die Parameter-eingabe SQL-Abfragen manipulieren und weiteren SQL-Code einschleusen.



Beispiel: Parametereingabe über ein HTML-Formular und Übergabe an PHP

Artikel:

Anzeigen

Neue Artikel eingeben:

Bezeichnung:

Einkaufspreis:

Verkaufspreis:

Eintragen

Username:

Passwort:

Login

1.

Nur ausgewählte User sollen, nach Eingabe von Username und Passwort, neue Artikel eingeben können. Die Eingaben gelangen in die Variablen *user* bzw. *pw*.

SQL-Abfrage:

*"Select * from kunden where Name='user' and Passwort='pw'"*

SQL-Injection:

Username:

Passwort:

Der username wird erraten, der Rest der Abfrage wird als Kommentar markiert.

*"Select * from kunden where Name='Müller'#' and Passwort='pw'"*

2. Einschleusen von gefährlichem SQL-Code bei einer Datenbankeingabe.

SQL-Abfrage:

"insert artikel values (NULL,'bez', einkauf, verkauf)"

SQL-Injection:

Bezeichnung:

Bürostuhl

Einkaufspreis:

152.90

Verkaufspreis:

400);Delete from artikel;

Das SQL-Statement wird beendet und ein zweites angefügt.

"insert artikel values (NULL,'Bürostuhl', 152.90, 400);Delete from artikel;)"

3. Eine Eingabe wird um eine UNION-Vereinigung erweitert.

Das SQL-Statement liefert zusätzlich alle Daten der Tabelle *kunden*. Dabei ist lediglich zu beachten, dass die Ausgangstabelle und die Tabelle, die mit UNION vereinigt wird, dieselbe Anzahl an Feldern haben.

Artikel:

Monitor' union Select Name, Passwort from kunden;

Anzeigen

UNION: Bei der Vereinigungsmenge enthält die Ergebnismenge alle Datensätze, die in Tabelle 1 oder in Tabelle 2 oder in beiden Tabellen enthalten sind.

SQL-Abfrage:

"Select Bezeichnung, Verkaufspreis from artikel where Bezeichnung= 'artikel'"

SQL-Injection:

"Select Bezeichnung, Verkaufspreis from artikel where Bezeichnung= 'Monitor' union Select Name, Passwort from kunden;"

Abhilfe:

- in den SQL-Statements (in Python, PHP etc.) stehen nicht die übergebenen Parameter, sondern Platzhalter.
- Die SQL-Abfrage wird mit den Platzhaltern ausgeführt, die Parameter werden dann nachgereicht → SQL-Schadcode wird nicht ausgeführt, sondern als String gespeichert.

KundenNr	Name	GebDatum	Passwort
8	Hacker	1978-12-11	alles');Delete from kunden;

In Python werden standardmäßig prepared statements bei Datenbankeinträgen benutzt:
kunden_einfügen_input.py

Passwörter in der Datenbank hashen

KundenNr	Name	GebDatum	Passwort
1	Roscher	1999-11-11	oma
2	Meier	1998-11-11	1234
4	Werner	1969-11-11	\$2b\$12\$bHgbVcUnBym5YhG5J9ep..0/IJWGRZZ5z4CO2WS6.At...
5	Siemens	1999-11-11	\$2b\$12\$guAPgEMRSpH3ji328XkR3u4DBT8QVcUbvdKhjtJ36KE...

Persönliche Daten, insbesondere Passwörter sollten verschlüsselt, besser gehasht werden, falls die Datenbank gehackt wird.

- Verschlüsseln:** Ein Passwort mit einem Verschlüsselungsalgorithmus verschlüsselt. Der Entschlüsselungsalgorithmus (mit passendem Schlüssel) entschlüsselt das Passwort wieder in den Originaltext.
- Hashen:** Die Verschlüsselung ist nicht umkehrbar, durch hinzufügen von Salt (einer Zufallszahl) werden gleiche Passwörter unterschiedlich dargestellt. Salt wird am Ende des Passworts vor dem Hashing hinzugefügt und gibt eine pseudozufällige Zeichenfolge zurück.

Rechte auf dem Datenbankserver einschränken

Legen Sie in phpmyadmin über den Reiter *Rechte* zwei Benutzerkonten an, zwei für die Datenbank *Shop* und ein Konto für alle Datenbanken.

- Benutzer 1: bekommt nur das Recht SELECT,
 Benutzer 2: bekommt die Rechte SELECT und INSERT,
 Benutzer 3: bekommt alle Rechte auf alle Datenbanken

Öffnen Sie zwei Terminalfenster und rufen Sie den MySQL-Monitor mit *mysql* auf. Melden Sie sich jeweils mit den neuen Benutzerdaten an (s. unten) und führen Sie je einen SELECT- und INSERT-Befehl aus (auf die Tabelle *artikel*).

Unter Windows:

Gehen Sie über die Eingabeaufforderung in den 'bin-Ordner' von MySQL und melden Sie sich auf den MySQL-Server als Administrator mit folgenden Verbindungsparametern an: ***mysql -h localhost -u root -p*** → Enter; bei der Passwortabfrage Enter.

cd.. → in den übergeordneten Ordner wechseln

```
C:\>cd xampp
```

```
C:\xampp>cd mysql
```

cd xampp → in den untergeordneten Ordner *xampp* wechseln

```
C:\xampp\mysql>cd bin
```

```
C:\xampp\mysql\bin>mysql -h localhost -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with
```

Hinweis: Wenn eine Datenbank mit SQL-Befehlen bearbeitet werden soll, muss zuvor mit dem ***use***-Befehl die entsprechende Datenbank ausgewählt werden.

Benutzerverwaltung MySQL

Jeder Benutzer benötigt ein Konto für den Datenbankserver → Konto: `root@localhost` mit Hostnamen (oder IP-Adresse) – der Computer, von dem aus das Konto den MySQL-Server erreichen kann und ein Passwort.

Es können keine Rechte auf eine Datenbank bzw. Tabelle vergeben werden, wenn der Benutzer kein Konto auf dem Datenbankserver besitzt.

Grundsätzlich ist jeder Benutzer, der ein Datenbankobjekt wie z. B. eine Tabelle erstellt, automatisch auch Besitzer dieses Objektes. Alle anderen Benutzer des Systems besitzen für ein Datenbankobjekt eines anderen keine Zugriffsrechte, Ausnahme ist der Administrator.

SQL-Code:

Benutzer anlegen

CREATE USER 'jupp'@'%';	Ein Benutzer mit Benutzernamen 'jupp' wird angelegt. Es ist egal, von welchem PC aus er sich anmeldet.
CREATE USER 'jupp'@'localhost';	Zugriff nur für den Rechner, auf dem die MySQL Datenbankserver läuft
CREATE USER 'willi'@'192.168.2.100';	Bei Zugriff von einem bestimmten Client.

Benutzern Rechte zuordnen

<i>grant all on *.* to 'willi'@'localhost' ;</i>	Der Benutzer 'willi' hat alle Rechte auf den MySQL-Datenbankserver
<i>grant all on hdd.* to 'willi'@'localhost';</i>	Der Benutzer 'willi' hat nur Rechte auf die Datenbank hdd
<i>grant select, insert on *.* to 'willi'@'localhost' with grant option;</i>	Der Benutzer 'willi' darf auf dem Datenbankserver die Funktionen select und insert ausführen und diese Zugriffsrechte anderen Benutzern gewähren.
<i>grant Select, Update on hdd.Tabelle1 to willi'@'localhost';</i>	Der Benutzer 'willi' darf auf der Tabelle1 der Datenbank hdd nur die Funktionen select und update ausführen
<i>with grant option</i> – Erlaubt einem Benutzer, anderen Benutzern Zugriffsrechte zu gewähren. Diese Rechte können dem 2. Benutzer vom Administrator nicht entzogen werden.	

Passwörter für Benutzer anlegen

SET PASSWORD FOR 'jupp'@'localhost' = PASSWORD('siemens');

Widerrufen von Benutzerrechten

REVOKE ALL ON hdd.* FROM 'willi'@'localhost';	Dem Benutzer 'willi' werden alle Rechte auf die Datenbank hdd entzogen
REVOKE SELECT, UPDATE ON hdd.* FROM 'willi'@'localhost';	Der Benutzer 'willi' darf die Funktionen Select und Update bei der Datenbank hdd nicht einsetzen