

MySQL / MariaDB

- Alle Benutzer in MySQL / MariaDB anzeigen
- Datenbanken in MySQL / MariaDB anlegen
- Benutzerverwaltung
 - Admin Benutzer in MySQL / MariaDB anlegen
 - Anmeldung mit root auf lokalem Datenbank-Server
- MySQL / MariaDB Verbindung von Außen erlauben
- Daten Verwalten
 - MySQL String ersetzen
- Backup einer MySQL Datenbank erstellen
- Backup einer MySQL Datenbank einspielen
- Auto_Increment in MySQL / MariaDB zurücksetzen

Alle Benutzer in MySQL / MariaDB anzeigen

Einleitung

Beim Arbeiten mit Datenbanken möchtest du vielleicht einmal nachschauen, welche Benutzer noch angelegt sind, und welchen Zugriff auf die Datenbanken haben. Alte Benutzer können ein sehr hohes Sicherheitsrisiko darstellen, daher sollten Benutzer, die nicht mehr verwendet werden, umgehend beseitigt werden.

Alle Benutzer anzeigen

Um alle Benutzer anzeigen zu lassen, setzt du folgenden Befehl in der SQL Shell ab

```
SELECT * FROM mysql.user;
```

Du bekommst dann eine Tabelle zurückgegeben, mit allen Benutzern, die noch angelegt sind, mit Informationen darüber, von welchem Host diese sich anmelden dürfen, wie der Benutzername ist und eine Angabe über das Passwort, natürlich in verschlüsselter Form. Des Weiteren sieht man die Berechtigungen auf den MySQL Server vom jeweiligen Benutzer.

Datenbanken in MySQL / MariaDB anlegen

Einleitung

Um Daten langfristig zu speichern, bieten sich Datenbanken an. Datenbanken haben den entscheidenden Vorteil, dass sich mehrere Benutzer gleichzeitig dort anmelden können. Dadurch können Daten schnell und effizient abgefragt werden.

Datenbank anlegen

Wenn du eine Datenbank anlegen möchtest, musst du dir zuerst einen Namen für diese Ausdenken. Der Name darf Buchstaben von **A-Z** ohne Umlaute enthalten.

Ein Datenbank Namen darf auch keine führende Zahl enthalten. Des weiteren können Sonderzeichen verwendet werden, aber es können nicht alle Sonderzeichen für Datenbanknamen verwendet werden. Beispielsweise kann ein Bindestrich (-) nicht verwendet werden, ein Unterstrich(_) ist aber in Ordnung.

Wenn du dich nun für einen Namen entschieden hast, kannst du nun deine Datenbank erstellen.

```
CREATE DATABASE <datenbank-name>;
```

Benutzerverwaltung

Admin Benutzer in MySQL / MariaDB anlegen

Einleitung

In diesem Beitrag geht es kurz darum, wie wir in MySQL und MariaDB einen Administrator-Account anlegen können. Diesen nutzen wir z.B. um administrative Tätigkeiten auf unserem Server durchführen zu können. Um diesen zu erstellen. Benötigen wir einen Datenbank-Benutzer, der schon administrative Privilegien besitzt.

Benutzer erstellen

Im ersten Schritt stellen wir eine Verbindung mit unserem Datenbank-Server her und öffnen die Shell unseres MySQL / MariaDB Servers. Wenn wir uns direkt auf dem Datenbank-Server befinden, können wir den nachstehenden Befehl eingeben, um die Shell zu öffnen.

```
# Öffnen der Shell ohne Passwort  
mysql -u root
```

```
# Öffnen der Shell mit Passwort  
mysql -u root -p
```

Um jetzt den Benutzer zu erstellen, überlegen wir uns einen Benutzernamen, ein Kennwort und von welchem Server / Client auf den Server zugegriffen wird.

Mit dem nachfolgenden Befehl kann der Benutzer erstellt werden:

```
CREATE USER 'benutzer'@'%' IDENTIFIED BY 'SicheresKennwort123!';
```

Wir können bei diesem Befehl mit angeben, von welchem Client / Server sich mit dem Benutzer angemeldet werden kann. Diese Information geben wir beim `CREATE USER` Befehl in die Apostroph-Zeichen hinter dem `@` Zeichen ein.

| Zugriff von: | Angabe |
|---|-----------|
| Vom Host selbst (Direkt vom Datenbank Server) | localhost |

| | |
|-------------------------------------|--|
| Von jeder IP-Adresse aus | % |
| Von einer bestimmten IP-Adresse aus | IP-Adresse/Subnetzmaske z.B. <input type="text" value="192.168.10.5/255.255.255.0"/> |

Als letzten Schritt müssen wir jetzt einmal die Berechtigungen geben. Wir können hier administrative Privilegien für eine Datenbank oder für alle aktuellen und zukünftigen Datenbanken geben.

```
# Admin für alle aktuellen und zukünftigen Tabellen
```

```
GRANT ALL PRIVILEGES ON *.* TO 'benutzername'@'%';
```

```
# Admin nur für eine Datenbank
```

```
GRANT ALL PRIVILEGES ON db.* TO 'benutzername'@'%';
```

```
# Admin nur für eine Tabelle
```

```
GRANT ALL PRIVILEGES ON db.tabelle to 'benutzername'@'%';
```

Wir können uns jetzt mit dem Benutzer an unserer Datenbank anmelden.

Anmeldung mit root auf lokalem Datenbank-Server

Einleitung

In diesem kurzen Artikel geht es kurz darum, wie wir auf unserem **MySQL** oder **MariaDB-Server** einstellen können, dass wir uns dort mit dem Benutzer **root** auf unserem Datenbank-Server anmelden können.

Info: Beachte das kein externer Zugriff möglich ist. Dies stellt ein Sicherheitsrisiko dar!

Problem beheben

Um das Problem zu beheben, müssen wir lediglich die lokale **MySQL-Shell** aufrufen und dort den folgenden Befehl eingeben:

```
ALTER USER 'root'@'localhost' IDENTIFIED VIA mysql_native_password USING PASSWORD('password');
```

Wichtig: Verändere das Wort "*password*" durch dein eigenes Passwort!

Ein Login sollte jetzt möglich sein!

MySQL / MariaDB

Verbindung von Außen erlauben

Einleitung

In diesem Beitrag gehe ich kurz darauf ein, wie wir auf unserem Linux Server, auf dem **MySQL** oder **MariaDB** läuft, Verbindungen von Außen zulassen. Dies benötigen wir z.B. wenn wir eine Applikation nutzen wollen, die auf einem anderen Server läuft als unsere Datenbank. Dafür müssen wir eine Konfigurationsdatei editieren.

Bedenke: Eine Eröffnung von neuen Wegen zum Zugriff auf Server stellen immer neue Sicherheitsrisiken dar! Stelle sicher das dass du dir dem Risiko bewusst bist!

MySQL

Um in MySQL Verbindungen von Außen zu erlauben, müssen wir die Konfigurationsdatei mit einem Editor unserer Wahl öffnen. Ich verwende dazu **nano**. Dieser ist auf fast allen Linux Distributionen vorinstalliert und lässt sich sehr leicht verwenden.

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Dort müssen wir den Punkt **bind-address** finden. Dort müsste aktuell die **localhost Adresse (127.0.0.1)** stehen. Diesen Wert ändern wir auf **0.0.0.0**, um von allen Servern den Zugriff zu erlauben. Möchten wir aber, dass der Zugriff nur von einem bestimmten Server möglich sein soll, setzen wir dort die **IP-Adresse des Servers** ein.

```
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
bind-address            = 0.0.0.0
```

Zum Schluss müssen wir nur noch den MySQL Service neu starten. Danach sollte eine Verbindung möglich sein.


```
sudo systemctl restart mysql
```

MariaDB

In MariaDB funktioniert das fast genauso wie in MySQL. Es verändert sich lediglich nur die Konfigurationsdatei, die wir öffnen und editieren müssen. Ich verwende hier wieder **nano**, um die Konfigurationsdatei zu öffnen.

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Bei dieser Datei müssen wir auch den Punkt **bind-address** finden und diese dann auf die **IP-Adresse 0.0.0.0** setzen. Natürlich können wir hier auch wieder die **IP-Adresse** des **Servers** eintragen, welcher Zugriff auf den SQL Server haben soll.

Im Anschluss starten wir auch den MySQL Dienst wieder neu.

```
sudo systemctl restart mysql
```

Daten Verwalten

MySQL String ersetzen

Einleitung

In dieser Anleitung beschreibe ich kurz, wie wir einen spezifischen **MySQL-String** in einem **MySQL-Datensatz** ersetzen können. So können wir schnell größere Datensätze bearbeiten.

String ersetzen

Um den String in einer Datenbank-Abfrage zu ändern, müssen wir den folgenden Code verwenden:

```
UPDATE tblName  
SET columnName = REPLACE(columnName, 'Value1', 'Value2');
```

Anhand des oben angegebenen Codes können wir die Datensätze in der Datenbank verändern.

Backup einer MySQL Datenbank erstellen

Einleitung

Es ist immer ratsam, Backups von einem System zu erstellen. Wir werden hier den Befehl `mysqldump` verwenden. Dieser kann dazu verwendet werden, um Backups einer bestimmten Datenbank oder mehreren Datenbanken zu erstellen.

Backup einer Datenbank

Um ein Backup einer Datenbank zu erstellen, brauchen wir den Namen der entsprechenden Datenbank. Wir können uns alle Datenbanken anzeigen lassen, mit dem folgenden Befehl in der **MySQL Shell**.

```
SHOW DATABASES;
```

Im Beispiel erstellen wir ein Backup der Datenbank **_erp_prod**. Um dieses Backup zu erstellen, verwenden wir dann den folgenden Befehl. Bei dem Befehl müssen wir nur den **Datenbank-Namen** und den Namen der **Backup-Datei** ändern.

```
mysqldump -u root -p _erp_prod > backup.sql
```

Der Befehl wird aus der **Linux-Shell** abgesetzt, nicht aus der **MySQL-Shell**!

Wir erhalten dann die Backup-Datei in unserem aktuellen Verzeichnis und können diese dann sichern, oder auf einem anderen Server wieder einspielen.

Backup mehrerer Datenbanken in einer Datei

Um mehrere Datenbanken in einer Datei zu sichern, verwenden wir einen ähnlichen Befehl. Dabei verändern wir wieder die Namen der Datenbanken und der Backup-Datei.

```
mysqldump -u root -p --databases datenbank_eins datenbank_zwei > backup.sql
```

Die beiden Datenbanken befinden sich dann jetzt in der einen **.sql** Datei und können diese auch wieder woanders sichern oder einspielen.

Alle Datenbanken in einer Datei sichern

Wenn wir jetzt alle verfügbaren Datenbanken in einer Datei sichern möchten, müssen wir lediglich den folgenden Befehl verwenden. Dann wird wieder eine Backup-Datei erstellt und alle Daten der Datenbanken werden in diese Datei geschrieben.

```
mysqldump -u root -p --all-databases > alle_datenbanken.sql
```

Datenbanken in verschiedenen Dateien sichern

Jetzt zum Schluss können wir die einzelnen Datenbanken in jeweils einer eigenen Datei sichern. Dazu verwenden wir ein **Bash-Skript** welches wir auf unserem Server ausführen. Es werden dann einzelne Dateien angelegt, die den Inhalt der jeweiligen Datenbank haben.

```
for DB in $(mysql -e 'show databases' -s --skip-column-names); do
    mysqldump $DB > "$DB.sql";
done
```

Backup einer MySQL Datenbank einspielen

Einleitung

Sobald wir ein Backup einer **MySQL Datenbank** erstellt haben, möchten wir diese vielleicht auch auf einen anderen Server spielen. Dies machen wir, wenn wir z.B. den **Datenbankserver** umziehen möchten. Dazu brauchen wir die **mysqldump** Datei. Diese haben wir erstellt, als wir das Backup angelegt haben.

Backup einspielen

Wir müssen zuerst, in den meisten Fällen, erstmal die Datenbank anlegen, bevor wir ein Backup für die Datenbank einspielen können. Dazu verwenden wir den `CREATE DATABASE` Befehl.

```
CREATE DATABASE <datenbank>;
```

Wenn wir die Datenbank angelegt haben, können wir mit dem unten stehenden Befehl die Datenbank mit den Daten des Backups befüllen. Dazu geben wir noch den Namen der Datenbank ein, und den Pfad zur Backupdatei. Dadurch wird das Backup dann in die angegebene Datenbank eingespielt.

```
mysql datenbank_name < backup.sql
```

Datenbank aus einer großen Backupdatei wiederherstellen

Wir können auch, wenn wir eine Datenbankdatei haben, welche mehrere Datenbanken beinhaltet, nur eine Datenbank wiederherstellen. Der Befehl wird dann durch einen weiteren Parameter erweitert.

```
mysql --one-database datenbank_name < backup.sql
```

Dadurch wird nur die Datenbank mit dem entsprechenden Namen wiederhergestellt. Dabei müssen wir beachten, dass die Datenbank vorher auf dem Server angelegt sein muss.

Auto_Increment in MySQL / MariaDB zurücksetzen

Einleitung

Mit dem SQL-Befehl `AUTO_INCREMENT` können wir eindeutig identifizierbare Datensätze in der Datenbank erstellen. Die ID zählt sich von alleine immer um einen hoch. Wenn wir diese **ID** zurücksetzen wollen, müssen wir entweder die Tabelle neu anlegen, oder wir können einen Befehl verwenden, um die **ID** zurückzusetzen.

Auto_Increment zurücksetzen

Wenn wir die **ID** zurücksetzen möchten, müssen wir lediglich nur den nachstehenden Befehl auf unserem Server abschicken. Wir müssen dabei nur den Tabellen-Namen anpassen.

```
ALTER TABLE tabellen_name AUTO_INCREMENT = 1;
```