Das Berechtigungssystem von MySQL-Datenbanken

Berechtigungen

Bei allen Anwendungen von MySQL muss gewährleistet sein, dass nur jene Personen Zugriff auf die Daten erhalten, die auch die Berechtigung dazu haben. **Von Stefan Pröll, Eva Zangerle und Wolfgang Gassler**

AUF EINEN BLICK

- MySQL steuert die Sicherheit mit sogenannten Berechtigungen, die an Datenbankbenutzer vergeben werden können.
- Der Workshop erläutert, welche Berechtigungen MySQL zur Verfügung stellt und wie diese vergeben werden.

Der MySQL-Server kommt in zahlreichen Anwendungsbereichen zum Einsatz (Bild 1). Es kann sich um eine einfache Installation auf Ihrem lokalen Rechner, aber auch um eine Großinstallation über mehrere Server hinweg in einem Hochsicherheits-Rechenzentrum handeln. In beiden Fällen muss die Sicherheit Ihrer Daten in den Datenbanken gewährleistet sein, damit nur jene Personen Zugriff auf Ihre gespeicherten und unter Umständen sehr kritischen Informationen erhalten, die auch wirklich die Berechtigung dazu haben. MySQL bietet hierfür zahlreiche Funktionen und Sicherheitsmerkmale.

MySQL steuert die Sicherheit mit sogenannten Berechtigungen (oder auch Rechte genannt), die an Datenbankbenutzer vergeben werden können. Berechtigungen erlauben oder beschränken dabei das Ausführen von Aktionen in der Datenbank. So kann zum Beispiel eine Berechtigung *SELECT*-Abfragen auf einer bestimmten Tabelle erlauben. Ist keine Berechtigung vorhanden, wird die Durchführung des Befehls von MySQL aus Sicherheitsgründen verweigert.

Szenarien für das Rechtemanagement

Beginnen wir zunächst mit dem einfachsten Einsatzszenario, einer MySQL-Installation mit Ihnen als einzigem Benutzer auf Ihrem lokalen Rechner, auf den nur Sie Zugriff haben. In diesem Fall können Sie für erste Testzwecke die Anforderungen bezüglich Sicherheit und Rechtemanagement in den meisten Fällen auf ein Minimum reduzieren. Es ist ausreichend, wenn Sie ein Passwort für den bereits vorinstallierten Administratorbenutzer *root* setzen und sich aus Sicherheitsgründen ein zusätzliches Benutzerkonto mit weniger Berechtigungen für die eigentliche Arbeit mit MySQL anlegen.

MySQL ist eine der populärsten Datenbanken im Open-Source-Bereich (Bild 1)

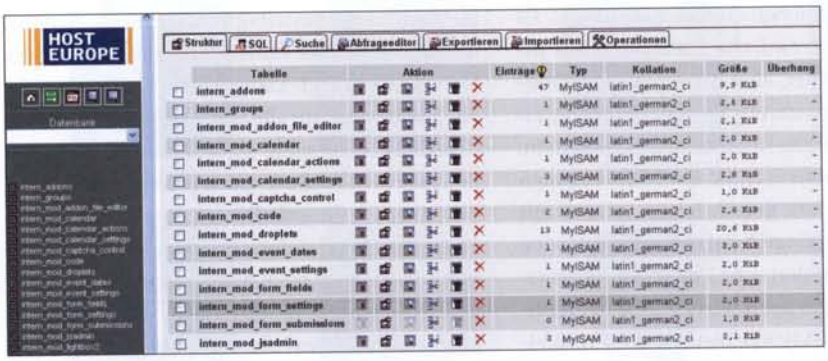
Sobald mehrere Personen oder Applikationen auf die Datenbank zugreifen, sollten Sie sich auf jeden Fall mehr Gedanken über die Sicherheit machen und das sehr ausgereifte Sicherheitskonzept von MySQL nutzen. Denken Sie etwa an die Betreiberfirma eines Flughafens, die alle firmenrelevanten Informationen in der Datenbank ablegt. Dabei sollen natürlich nicht alle Mitarbeiter auf alle Daten zugreifen können. Gerade sehr sensible Informationen, wie zum Beispiel die Gehälter eines jeden Mitarbeiters, dürfen nicht für jede Person einsehbar sein. Es muss aber auch unterschieden werden, ob Daten nur gelesen oder auch geschrieben – also verändert – werden dürfen. In unserem Flughafenbeispiel etwa müssen zahlreiche Benutzer der Datenbank die Flugtabelle lesen können. Es soll jedoch nicht jeder beliebige Flughafenmitarbeiter Änderungen an der Flugtabelle vornehmen dürfen.

Wenn Sie nun die Rechteebene noch um eine Stufe verfeinern, befinden wir uns nicht mehr auf der Tabellenebene, sondern auf Spalten- oder sogar Zeilenebene. Betrachten Sie zum Beispiel die Mitarbeitertabelle, die neben Kontaktinformationen auch eine Spalte *gehalt* beinhaltet. Wie bereits erwähnt, darf die Höhe des Gehalts nur von einem sehr beschränkten Benutzerkreis – etwa der Managementebene – eingesehen werden. Die Kontaktdaten (Spalte der E-Mail-Adresse) hingegen sollen den meisten Mitarbeitern zugänglich gemacht werden. Wir benötigen daher eine genaue Beschränkung von Berechtigungen auf Spaltenebene.

Dieselbe Einschränkung wäre auf Zeilenebene denkbar. Eventuell dürfen gewisse Daten der Mitarbeiter nur von Mitarbeitern aus der gleichen Abteilung eingesehen werden. Zum Beispiel könnte eine Abteilungsleiterin die Gehälter der Mitarbeiter ihrer Abteilung einsehen dürfen. Die anderen Zeilen (= Mitarbeiter anderer Abteilungen) müssten jedoch weiterhin geschützt bleiben und der Zugriff müsste verweigert werden. MySQL bietet auf Spalten- und Zeilenebene nur beschränkte Möglichkeiten, Berechtigungen und damit verbundene Beschränkungen zu definieren, jedoch gibt es auch dafür Tricks und Lösungswege.

In sehr umfangreichen Installationen mit vielen Datenbankbenutzern ist es üblich, dass gewisse Bereiche auch von geschulten Datenbankbenutzern selbstständig administriert werden. Sie kennen eine derartige Konfiguration vielleicht aus sogenannten Shared-Hosting-An- ▶

geboten im Internet, die Webspace inklusive Datenbank für den Internetauftritt vermieten (Bild 2). Dabei werden viele Kunden beziehungsweise ihre Webseiten und Datenbanken auf einem Server betrieben. Zum Einsatz kommt jedoch nur eine einzelne MySQL-Installation. Trotzdem können die Administratoren der jeweiligen Webseiten ihre Tabellen oder sogar ihre Datenbank selbst verwalten, anlegen, löschen und manipulieren. MySQL bietet hier sehr flexible Einstellungsvarianten im Rechtemanagement, die es erlauben, Berechtigungen auf bestimmte Datenbanknamenmuster zu setzen. So kann etwa realisiert werden, dass ein Administrator der Webseite www.mein-kleiner-flughafen.at alle Datenbanken, die mit *flughafen_* beginnen, selbstständig administrieren kann. Er kann somit zum Beispiel die Datenbanken *flughafen_cms* und *flughafen_management* anlegen und darin auch alle Tabellen selbst verwalten.



Bei Shared-Hosting-Angeboten werden Webspace inklusive Datenbank für den Internetauftritt vermietet (Bild 2)

Erhält jeder Administrator einen eigenen Bereich, der sich in diesem Fall durch den Beginn des Datenbanknamens definiert, können so auch Hunderte Webseiten, deren Datenbanken und die Datenbankbenutzer flexibel mit einer Installation verwaltet werden. Die beschriebenen Szenarien sind natürlich nur ein Bruchteil der möglichen Realisierungen. Der MySQL-Server gibt Ihnen mit seinem Rechtemanagement ein sehr mächtiges, aber dennoch einfaches Werkzeug an die Hand, mit dem Sie zahlreiche Szenarien – von der kleinen Installation bis zur großen Installation mit Tausenden Benutzern – sicher und verlässlich bewältigen können.

Das Zugriffsberechtigungssystem stellt das wichtigste Sicherheitsmerkmal von MySQL dar und ist dem eigentlichen Datenbanksystem vorgeschaltet. Es überwacht die gesamte Kommunikation zwischen Server und Client, überprüft dabei ständig die Berechtigungen und gewährleistet dadurch die notwendige Sicherheit.

Dabei wird zwischen zwei Zugriffsphasen unterschieden. Die erste sicherheitsrelevante Phase, in der bereits Zugriffsrechte überprüft werden müssen, ist die Phase der Verbindung. In der Verbindungsüberprüfung wird kontrolliert, ob sich der Benutzer von seinem Computer (Host) aus zur Datenbank verbinden darf. Ist diese Überprüfung erfolgreich, wird vom Client das Passwort in Kombination mit dem Benutzernamen übertragen und vom Server überprüft. Sind alle drei Daten (Host, Benutzer, Passwort) korrekt, kann die Verbindung zur Datenbank ordnungsgemäß hergestellt werden.

Neben der üblichen Prüfung von Benutzername und Passwort bringt die Prüfung des Hosts eine zusätzliche Steigerung der Flexibilität und Sicherheit. So können Sie zum Beispiel gewissen Benutzern erlauben, sich nur von ihrem eigenen Rechner mit einer speziellen IP-Adresse zu verbinden. Üblich ist hier auch die Angabe eines ganzen IP-Bereichs, etwa dem des internen Firmennetzwerks. So sind zum Beispiel auch Zugriffe von außen, bei denen eventuell das Passwort durch ein automatisches Hacker-Programm erraten werden könnte, nicht möglich und werden abgelehnt. Verwenden Sie dennoch »starke« Passwörter (mit Sonderzeichen und Zahlen) für Ihre Datenbankbenutzer, da auch Host-Angaben unter Umständen gefälscht werden können. Die Host-Einschränkung ist daher eine gute Zusatzabsicherung, sie kann jedoch den Passwortschutz durch ein »starkes« Passwort nicht ersetzen.

Wenn Sie nun die Verbindungsüberprüfung erfolgreich absolviert haben, werden Sie mit dem MySQL-Server verbunden und können Anfragen an den Server senden. Jede dieser Anfragen, wie zum Beispiel die Wahl einer Datenbank oder eine SQL-Abfrage, wird ebenfalls überprüft. Diese sogenannte Anfrageüberprüfung stellt die zweite Zugriffsphase dar. Dabei wird genau kontrolliert, ob Sie die ausreichenden Berechtigungen in der jeweiligen Datenbank und Tabelle besitzen, damit die Anfrage erfolgreich durchgeführt werden kann. Wie bereits oben erklärt, bietet MySQL für diese Überprüfung ein sehr umfangreiches Rechtesystem, das vom einfachen Lesen bis zur Administration einer Tabelle sehr genau für jeden Benutzer eingestellt werden kann.

Privilegiert – die Benutzerrechte im Detail

Um einem Benutzer Berechtigungen zuweisen zu können, müssen Sie zunächst einen Benutzer anlegen (Bild 3). Verbinden Sie sich dazu als Benutzer *root* mit dem MySQL-Server und erstellen Sie einen neuen Benutzer:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO
'test'@'localhost' IDENTIFIED BY
'testpasswort' WITH GRANT OPTION;
```

Mit dieser Anweisung wird ein neuer Benutzer mit dem Namen *test* erstellt und das Passwort *testpasswort* gesetzt. Dieser Datenbankbenutzer

verfügt durch die Anweisung *ALL PRIVILEGES* über alle verfügbaren Rechte in der MySQL-Installation. Anschließend legen Sie fest, auf welchen Datenbanken beziehungsweise Tabellen die angegebenen Rechte gültig sind.

Grundsätzlich wird vor dem Punkt der Datenbankbezeichner und nach dem Punkt ein Tabellenbezeichner einer Tabelle in der angegebenen Datenbank angezeigt, auf welche die Berechtigungen gewährt werden sollen. Das Zeichen * steht für einen beliebigen Namen beziehungsweise Bezeichner.

Durch die Bezeichnung eines Hosts nach dem @, in diesem Fall *localhost*, kann sich der Benutzer *test* nur vom lokalen Computer aus zum MySQL-Server verbinden. Der Benutzer muss sich daher an demselben Computer wie der MySQL-Serverdienst befinden. Selbstverständlich können Sie auch durch % als Host-Angabe spezifizieren, dass sich der Benutzer von einem beliebigen Host aus anmelden kann. Wird kein @ beziehungsweise kein Host angegeben, wird ebenfalls die Anmeldung von jedem Host erlaubt. Die letzte Anweisung *WITH GRANT OPTION* erlaubt dem Benutzer, die *GRANT*-Anweisung auszuführen und damit seine Rechte auch an andere Benutzer weiterzugeben.

Bei der *GRANT*-Anweisung handelt es sich um den Befehl zum Setzen von gewissen Benutzerrechten. Existiert der angegebene Benutzer wie in unserem Beispiel jedoch noch nicht, legt die *GRANT*-Anweisung auch diesen automatisch an und setzt anschließend die Rechte.

Neben der *GRANT*-Anweisung ist die Anweisung *REVOKE* sehr wichtig und wird dazu verwendet, bereits vergebene Rechte zu widerrufen beziehungsweise aufzuheben. Im folgenden Listing sehen Sie eine Anweisung, die alle Rechte auf allen Datenbanken und das *GRANT*-Recht unseres Testbenutzers wieder aufhebt:

```
mysql> REVOKE ALL PRIVILEGES, GRANT
OPTION -> FROM 'test'@'localhost';
```

Beachten Sie, dass der Benutzer zwar alle Rechte auf allen Datenbanken verliert, dass das Benutzerkonto jedoch weiterhin bestehen bleibt und der Benutzer sich daher auch weiterhin mit dem MySQL-Server verbinden kann. Die Anweisung *REVOKE* löscht also in keinem Fall einen Benutzer aus dem MySQL-System. Wollen Sie den Benutzer vollständig und unwiderruflich löschen, ist dies mit der Anweisung *DROP USER* möglich:

```
mysql> DROP USER 'test'@'localhost';
```

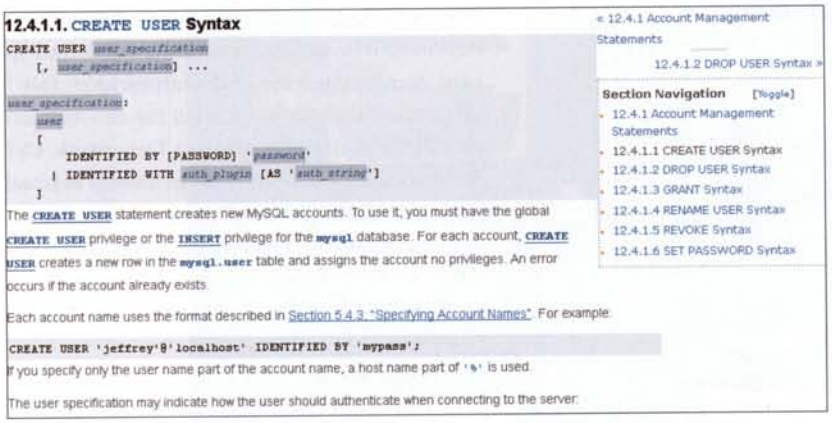
Nun haben Sie die wichtigsten drei Grundanweisungen kennengelernt, mit denen Sie bereits neue Benutzer anlegen, Berechtigungen vergeben und widerrufen und selbstverständlich auch Benutzer wieder löschen können. Im Folgenden gehen wir nun näher auf die verschiedenen Ebenen der Rechteverwaltung ein. Diese ermöglichen es Ihnen, Berechtigungen zum Beispiel nur auf einer gewissen Spalte oder aber auch datenbankweit zu setzen.

Die Ebenen der Benutzerberechtigungen

Je nach Berechtigung können die von MySQL zur Verfügung gestellten Rechte in unterschiedlichen Granularitäten vergeben werden. Dies hilft Ihnen dabei, mit dem Berechtigungssystem auf die Anforderungen in diversen Anwendungsszenarien möglichst flexibel zu reagieren. Dabei sind die folgenden Granularitäten oder Berechtigungsebenen in MySQL möglich:

- globale Ebene,
- Datenbankebene,
- Tabellenebene,
- Spaltenebene,
- Zeilenebene (nur begrenzt möglich),
- Objektebene

In Bild 4 sehen Sie die Berechtigungsebenen und wie diese zusammenwirken. Zum Beispiel schließt die Datenbankebene die Tabellenebene völlig mit ein. Das bedeutet, dass Berechtigungen auf Datenbankebene automatisch auch auf allen Tabellen in der jeweiligen Datenbank gültig sind. Andere Bereiche, wie zum Beispiel die Spalten- und Zeilenebene, schließen sich nicht vollständig ein, können sich aber bei gewissen Daten (Zellen) natürlich überschneiden. Auch Objekte sind unabhängig und können sich auf mehreren Ebenen befinden. Im Folgenden stellen wir Ihnen die verschiedenen Granularitäten beziehungsweise Ebenen und deren Anwendung genauer vor.

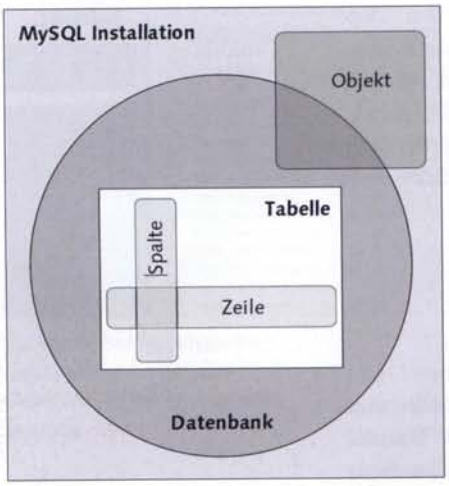


MySQL-Datenbankbenutzer können mit unterschiedlichen Rechten ausgestattet werden (Bild 3)

Berechtigungen auf der globalen Ebene sind über die gesamte MySQL-Installation hinweg gültig. Eine dieser Berechtigungen, die *GRANT*-Berechtigung zur Vergabe von Berechtigungen, haben Sie bereits kennengelernt. Dieses Recht kann nur auf globaler Ebene vergeben wer- ▶

den, da es sich nicht auf eine spezielle Datenbank oder Tabelle bezieht. In diesem Kontext spricht man auch von Administratorberechtigungen, da globale Berechtigungen meist mit der Verwaltung der MySQL-Installation in Verbindung stehen. Im folgenden Listing sehen Sie eine Vergabe der Berechtigung *SELECT* auf der globalen Ebene an den Benutzer *admin*:

```
mysql> GRANT SELECT ON *.* TO
'admin'@'%';
```



Berechtigungsebenen in MySQL (Bild 4)

Berechtigungen auf Datenbankebene können Sie jeweils auf eine oder mehrere Datenbanken setzen. Gerade in großen MySQL-Installationen, in denen es oft zahlreiche Datenbankadministratoren gibt, ist eine Rechte-Granularität auf Datenbankebene vonnöten. So kann man zum Beispiel einem Webadministrator des Flughafens erlauben, dass er die Tabellen in der Datenbank *cms* selbst verwaltet. Er nimmt so dem Hauptadministrator viel Arbeit ab und kann in seiner Datenbank selbst Tabellen erzeugen *(CREATE)*, editieren *(ALTER)* oder auch löschen *(DROP)*. Die Datenbankebene wird mit dem Namen der Datenbank und einem darauffolgenden .* gekennzeichnet. Der * (Wildcard-Symbol) steht dabei für eine beliebige Tabelle in der angegebenen Datenbank. Der Benutzer *admin* besitzt damit in diesem Beispiel die *DROP*-Berechtigung auf allen Tabellen (.*) in der angegebenen Datenbank *FlughafenDB*:

```
mysql> GRANT DROP ON FlughafenDB.* TO
'admin'@'%';
```

Selbstverständlich können Sie Berechtigungen auch auf Tabellenebene vergeben, um zum Beispiel gewissen Benutzern nur Leserechte *(SELECT)* auf einer Tabelle zu gewähren. Das Einfügen *(INSERT)* oder Aktualisieren *(UPDATE)* neuer Datensätze ist damit nicht möglich und wird vom MySQL-System verwehrt. Im folgenden Listing sehen Sie die Zuweisung der *SELECT*-Berechtigung auf die Tabelle *mitarbeiter*. Der Benutzer *admin* besitzt somit nur Leserechte auf der Tabelle *mitarbeiter*. Die Tabellenebene wird durch Angabe des Tabellennamens gekennzeichnet:

```
mysql> GRANT SELECT ON
FlughafenDB.mitarbeiter TO
'admin'@'%';
```

Die Lese- und Schreibrechte können Sie ebenfalls auf Spaltenebene setzen. So können Sie zum Beispiel einem Benutzer erlauben, die Spalte mit der E-Mail-Adresse der Mitarbeiter zu lesen. Sensible Daten in derselben Tabelle, wie zum Beispiel die Spalte mit der Höhe des Gehalts, können Sie für die Allgemeinheit sperren und zum Beispiel nur dem Personalmanager zugänglich machen. Im folgenden Listing wird die *UPDATE*-Berechtigung auf der Spalte *gehalt* dem Benutzer *admin* zugewiesen. Die Spalten werden dabei in einer Liste nach dem Bezeichner der Berechtigung in runden Klammern angegeben:

```
mysql> GRANT UPDATE(gehalt) ON
FlughafenDB.mitarbeiter -> TO
'admin'@'%';
```

MySQL bietet derzeit keine direkte Unterstützung für die Einschränkung von Zugriffen auf Zeilenebene. Mit ein wenig Fingerfertigkeit kann aber auch dieses Problem gelöst werden.

Objektebene

Die bisher beschriebenen Ebenen betrachten Berechtigungen in Bezug auf den Zugriff auf die in MySQL gespeicherten Daten – also Tabellen gruppiert in Datenbanken. Neben den gespeicherten Daten existieren aber auch noch andere Konstrukte in MySQL, die unter dem Sammelbegriff »Objekte« zusammengefasst werden. Diese Objekte können Sichten (Views), gespeicherte Prozeduren (Stored Procedures), Ereignisse (Events) oder Indizes sein, die selbstverständlich auch durch ein Sicherheitskonzept beziehungsweise durch Berechtigungen geschützt werden müssen.

Rechte auf Objektebene werden durch eine zusätzliche Angabe des Objekttyps angegeben:

```
mysql> GRANT EXECUTE ON PROCEDURE
FlughafenDB.berechne_distanz -> TO
'admin'@'%';
```

In diesem Beispiel wird die *EXECUTE*-Berechtigung auf eine Prozedur (Schlüsselwort *PROCEDURE)* mit dem Namen *berechne_distanz* vergeben.

Die Berechtigungen in MySQL können meistens auf einer speziellen Ebene gesetzt werden. Gewisse Berechtigungen lassen sich auch auf mehreren Ebenen setzen. So können Sie je nach Bedarf eine einfache Leseberechtigung *(SELECT)* global, auf Datenbanken oder nur auf einzelne Tabellen setzen. Das Recht zur Erzeugung von neuen Benutzern *(CREATE USER)* hingegen können Sie nur auf globaler Ebene setzen (Bild 5).

Im Folgenden gehen wir nun näher auf die einzelnen Rechte ein, die Ihnen MySQL zur Ver-

fügung stellt. Beginnen wir zunächst mit dem wichtigsten Gut einer Datenbank – Ihren Daten. In Tabelle 1 finden Sie eine Auflistung der wichtigsten Berechtigungen zur Verwaltung der Datenbanken, Tabellen und des Zugriffs auf die gespeicherten Daten. Die Spalte *Ebene/Kontext* beschreibt dabei, in welchem Kontext die Berechtigung verwendet wird beziehungsweise auf welcher Granularitätsebene diese gesetzt werden kann.

Die Berechtigung *SELECT* können Sie auf Tabellen- und Spaltenebene setzen. Sie berechtigt einen Benutzer, lesend auf eine Tabelle beziehungsweise Spalte zuzugreifen. Beachten Sie, dass bei *UPDATE*- und *DELETE*-Anweisungen stets eine Leseberechtigung vonnöten ist. So benötigt zum Beispiel die folgende SQL-Anweisung zur korrekten Durchführung auch Leserechte für die Spalte *flug_id*:

```
mysql> DELETE FROM flug WHERE flug_id
< 5;
```

Betrachten Sie das Beispiel eines Webadministrators, der für den Webauftritt des Flughafens verantwortlich ist. Dieser soll auf allen Tabellen

TABELLE 1: BERECHTIGUNGEN

Berechtigung	Ebene/Kontext
SELECT	Tabellen, Spalten
INSERT	Tabellen, Spalten
UPDATE	Tabellen, Spalten
DELETE	Tabellen
CREATE	Datenbanken, Tabellen, Indizes
CREATE TEMPORARY TABLES	Datenbanken
ALTER	Tabellen
DROP	Datenbanken, Tabellen, Sichten
INDEX	Tabellen
LOCK TABLES	Datenbanken
CREATE VIEW	Sichten
SHOW VIEW	Sichten

des Content-Management-Systems in der Datenbank *cms* uneingeschränkten Lesezugriff erhalten. Im folgenden Listing wird diese Berechtigung für den Benutzer mit dem Namen *webadmin* gesetzt:

```
mysql> GRANT SELECT ON `cms`.* TO
'webadmin'@'%';
```

Gewisse *SELECT*-Anweisungen benötigen jedoch auch keine besonderen Berechtigungen. Wird zum Beispiel bei einer *SELECT*-Anweisung nicht auf Daten zugegriffen, so können diese Anweisungen auch ohne *SELECT*-Berechtigung durchgeführt werden. Sogar der Aufruf von Funktionen ist ohne Berechtigung mög- ▶

lich, wenn in der Funktion nicht auf Daten zugegriffen wird.

Die Berechtigungen *INSERT* und *UPDATE* können Sie in derselben Weise wie die Berechtigung *SELECT* auf Tabellen- und Spaltenebene vergeben. *INSERT* erlaubt dabei das Einfügen von neuen Datensätzen in der jeweiligen Tabelle. Beachten Sie, dass die Berechtigung *INSERT* auch für die korrekte Durchführung der Operationen *ANALYZE_TABLE*, *OPTIMIZE_TABLE* und *REPAIR_TABLE* benötigt wird.

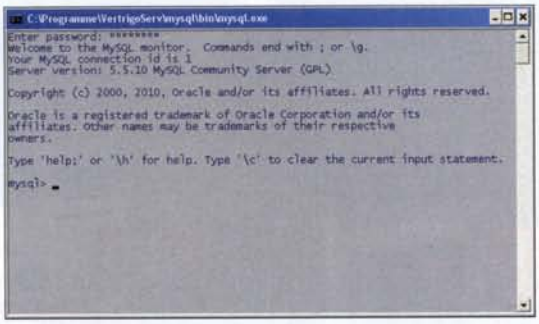
Das Recht *UPDATE* erlaubt einem Benutzer, den Inhalt einer Zeile zu aktualisieren, also zu verändern.

Die Berechtigung *DELETE* erlaubt das Löschen von Zeilen in einer angegebenen Tabelle. Da mit der *DELETE*-Anweisung immer ganze Zeilen gelöscht werden, können Sie diese Berechtigung nicht auf Spaltenebene, sondern nur auf Tabellenebene setzen.

Die Berechtigung *CREATE* erlaubt das Erstellen von Tabellen und Datenbanken. Für die Erstellung von Tabellen können Sie genau spezifizieren, in welchen Datenbanken es erlaubt ist, Tabellen anzulegen. Zusätzlich können Sie die Namensgebung von Datenbanken und Tabellen beeinflussen.

Erzeugung von temporären Tabellen

Die Berechtigung *CREATE TEMPORARY TABLE* erlaubt die Erzeugung von temporären Tabellen. Diese Tabellen können Sie mit dem erweiterten Befehl *CREATE TEMPORARY TABLE* zusammen mit den gewünschten Spalten wie gewohnt anlegen.



Berechtigungen lassen sich über die MySQL-Konsole oder mit Hilfe von Admin-Tools setzen (Bild 5)

Temporäre Tabellen sind dabei nur für die aktuelle Verbindung gültig und werden danach automatisch wieder gelöscht. Die jeweilige temporäre Tabelle ist auch nur für den erstellenden Benutzer beziehungsweise innerhalb der aktuellen Verbindung sichtbar, wobei temporäre Tabellen bei *SHOW TABLES* nicht mit angezeigt werden. Andere Benutzer sehen die temporäre Tabelle nicht und können diese auch nicht ansprechen.

Die Berechtigung *ALTER* erlaubt das Ausführen der Anweisung *ALTER TABLE* zur Modifizierung von Tabellen. Das Recht vergeben Sie daher auf der Granularitätsebene von Tabellen. Beachten Sie, dass Sie für die Veränderung von Tabellen auch die Berechtigungen *INSERT* und *CREATE* zusätzlich vergeben müssen.

Das Recht *DROP* erlaubt dem Benutzer das Löschen von Datenbanken und Tabellen. Im folgenden Listing wird dem Benutzer *webadmin* erlaubt, auch alle Datenbanken mit dem Präfix *cms_* beziehungsweise die darin befindlichen Tabellen und Sichten (Views) zu löschen:

```
mysql> GRANT DROP ON `cms\_%`.* TO
'webadmin'@'%';
```

Beachten Sie, dass die Berechtigung *DROP* auch für die Anweisung *TRUNCATE TABLE* benötigt wird, da diese Anweisung eine angegebene Tabelle löscht *(DROP)* und anschließend wieder anlegt *(CREATE)* und damit ein sehr schnelles Leeren von sehr speicherintensiven Tabellen realisiert. Auch Sichten werden mit dieser tabellenbezogenen Berechtigung verwaltet, da sie als »virtuelle« Tabellen angesehen werden.

Indizes anlegen

Mit der Berechtigung *INDEX*, die auf Tabellenebene spezifiziert wird, kann der jeweilige Benutzer Indizes auf den angegebenen Tabellen anlegen und auch wieder löschen. Beachten Sie, dass auch ein Benutzer mit der Berechtigung *CREATE* Indizes direkt bei der Erzeugung von Tabellen anlegen kann. Soll ein Index nachträglich angelegt oder gelöscht werden, wird jedoch das *INDEX*-Recht benötigt.

Die Berechtigung *LOCK TABLES* ist auf Datenbankebene gültig und erlaubt das Setzen von Sperren auf den Tabellen in der jeweiligen Datenbank. Die zugehörigen Befehle *LOCK TABLES* beziehungsweise *UNLOCK TABLES* ermöglichen mit der Berechtigung *LOCK TABLES* das Setzen und Aufheben von Schreib- oder Lesesperren.

Beachten Sie, dass dieses Recht keinen Einfluss auf das automatische Setzen von Sperren durch den MySQL-Server hat. MySQL setzt zum Beispiel auch unabhängig von Berechtigungen selbstständig Sperren zur Gewährleistung der Konsistenz im laufenden Betrieb.

Die Berechtigung *CREATE VIEW* erlaubt das Erstellen von neuen Sichten (Views) mit der Anweisung *CREATE VIEW* und kann in derselben Weise wie die Berechtigung *CREATE* verwendet werden. Es ist ebenfalls möglich, die zu erstellenden Sichten auf ein Namensmuster einzuschränken. Zusätzlich zu diesem Recht setzen Sichten ein weiteres Sicherheitskonzept ein, das während ihrer eigentlichen Ausführung ebenfalls zusätzliche Berechtigungen überprüft.

Die Berechtigung *SHOW VIEW* erlaubt die Verwendung der Anweisung *SHOW CREATE VIEW*, die den Quellcode der Sicht (View) anzeigt. Im ersten Moment erscheint die Erstellung einer eigenen Berechtigung für die Anzeige von Quellcodes etwas überspitzt. Sichten werden jedoch oft als Sicherungsschicht eingesetzt, die ge-

TABELLE 2: PROGRAMMIERUNG

Berechtigung	Ebene/Kontext
EVENT	Datenbanken
CREATE ROUTINE	gespeicherte Prozeduren
ALTER ROUTINE	gespeicherte Prozeduren
EXECUTE	gespeicherte Prozeduren
TRIGGER	Tabellen

wisse Details und Daten von Tabellen verstecken. Dadurch kann allein schon die Ansicht des Quellcodes einer Sicht bereits ein Sicherheitsrisiko darstellen.

Berechtigungen zur Programmierung

Neben den Berechtigungen zur Verwaltung und zum Zugriff auf Daten werden auch die Erstellung und Verwaltung von Funktionen, Prozeduren, Triggern und Events ebenfalls über eigene Berechtigungen kontrolliert. In Tabelle 2 sehen Sie eine Auflistung der Berechtigungen in Bezug auf die Programmierung mit MySQL.

Die Berechtigung *EVENT* setzen Sie auf Datenbankebene. Sie erlaubt dem Benutzer das Anlegen *(CREATE EVENT)* und Verwalten *(ALTER EVENT, DROP EVENT)* von Events, die auf den angegebenen Datenbanken operieren. Vielleicht fragen Sie sich, warum für dieses mächtige Konstrukt der Events nur eine Berechtigung existiert und diese nicht weiter unterteilt wird. Bei Events gibt es jedoch noch weitere Sicherheitsmaßnahmen, die mit dem Benutzer, der das Event erstellt beziehungsweise startet, gekoppelt sind.

Die Berechtigung *CREATE ROUTINE* operiert auf Datenbankebene und erlaubt dem jeweiligen Benutzer, gespeicherte Prozeduren und Funktionen in der angegebenen Datenbank zu erstellen. Beachten Sie, dass es nicht möglich ist, diese Berechtigung auf spezielle Namen oder Namensmuster für Prozeduren einzuschränken. Die Berechtigung gilt immer uneingeschränkt in der angegebenen Datenbank oder auf eine angegebene Prozedur.

Die Berechtigung *ALTER ROUTINE* erlaubt einem Benutzer die Modifizierung von Prozeduren. Beachten Sie, dass Sie bei Prozeduren nur die Metadaten modifizieren können. Für die Veränderung des Quellcodes einer Prozedur müssen Sie die Prozedur löschen und wieder erneut erstellen. Daher erlaubt diese Berechtigung neben der Modifizierung auch die Löschung (über den Befehl *DROP PROCEDURE)* der jeweiligen Prozedur beziehungsweise Funktion. Beachten Sie auch, dass Benutzer beim Erzeugen einer neuen Routine das zugehörige Recht *ALTER ROUTINE* für die erstellte Routine automatisch erhalten. Dieses Verhalten können Sie mit Hilfe der Variablen *automatic_sp_privileges* verändern und durch den Wert *0* deaktivieren.

Die Berechtigung *EXECUTE* benötigen Sie, um eine gespeicherte Prozedur auszuführen. Diese Berechtigung wenden Sie in derselben Weise wie die oben angeführte Berechtigung *ALTER ROUTINE* an, und zwar auf einzelne oder mehrere Prozeduren. Wie auch das Recht *ALTER ROUTINE* wird die *EXECUTE*-Erlaubnis dem Ersteller einer Prozedur automatisch zugewiesen. Dieses automatische Verhalten können Sie ebenfalls über die Variable *automatic_sp_privileges* deaktivieren (Wert *0*).

Beachten Sie aber, dass das Recht *EXECUTE* allein nicht die ordnungsgemäße Ausführung einer Prozedur garantiert. Während der Ausführung können noch weitere Rechte benötigt werden, die in Zusammenhang mit den in der Prozedur verwendeten Befehlen stehen.

Trigger beziehen sich immer auf Tabellen. Die zugehörige Berechtigung *TRIGGER* vergeben Sie daher auf Tabellenebene. Sie erlaubt das Anlegen, Löschen und Ausführen der Trigger auf der jeweiligen Tabelle. Die eigentliche Rechteüberprüfung des Triggers wird über das *DEFINER*- beziehungsweise *INVOKER*-Konzept realisiert. **[mb]**

Wenn Sie mehr über MySQL erfahren wollen, lesen Sie das Buch mit dem gleichnamigen Titel von den Autoren dieses Artikels, Stefan Pröll, Eva Zangerle und Wolfgang Gassler, erschienen im Verlag Galileo Computing.

BUCHTIPP: MYSQL – DAS HANDBUCH FÜR ADMINISTRATOREN

Installation, Konfiguration, Administration, Programmierung, Skalierung und Performance-Tuning sind die Themen dieses MySQL-Handbuchs.

Aktuell zu MySQL 5.5 und 5.6 finden Administratoren oder erfahrene Anwender in diesem Handbuch alles zur Installation, Konfiguration und der laufenden Verwaltung von MySQL. Das Buch vermittelt dabei nicht nur die Grundlagen, sondern geht darüber hinaus intensiv auf wichtige Themen wie die Performance- und Abfrageoptimierung ein. Einen weiteren Schwerpunkt legt es auf die Themen Sicherheit und Programmierung mit MySQL.

Die Autoren stellen aus ihrer langjährigen Erfahrung im Umgang mit dem Datenbanksystem zahlreiche leicht nachvollziehbare Beispiele zur Verfügung, damit der Leser alle Inhalte sofort in die Praxis umsetzen kann. Eine umfassende Beispieldatenbank auf DVD ermöglicht es dem Leser, sein Wissen sicher anzuwenden und hilfreiche Anwendungen wie zum Beispiel Analyse- und Monitoring-Tools direkt zu testen. Eine aussagefähige Befehlsreferenz zum Nachschlagen sowie ein umfangreicher Index runden dieses Handbuch ab. **[mb]**

Autoren	Stefan Pröll, Eva Zangerle, Wolfgang Gassler
Verlag	Galileo Computing
ISBN	978-3-8362-1715-6, 750 Seiten mit DVD
Preis	49,90 Euro