

LU12a: Data Security & User Management (DCL) - Rollen & Berechtigungen

Lernziele (gemäss Bildungsplan)

- **Rollen/Berechtigungen** vergeben (**GRANT/REVOKE**, Benutzer & Rollen verwalten)
- **Datensicherheit/Datenschutz**: Prinzipien & schützenswerte Daten erkennen
- MySQL-spezifische **DCL-Befehle** (MySQL **9.4**) sicher anwenden

Voraussetzungen



- MySQL **9.4**, WebStorm mit DB-Plugin, Root-Verbindung vorhanden.
- Achtung: Für Produktivsysteme nie mit root arbeiten – im Unterricht nutzen wir root **nur** zum Einrichten.

0) Kontext & Prinzipien

- **Least Privilege**: Jeder Account erhält nur die **minimal** benötigten Rechte.
- **Trennung nach Aufgaben**: Lese-User (**RO**)¹⁾, Schreib-User (**RW**)²⁾, Admin-User (DDL/Grants).
- **Datenschutz**: Personenbezug erkennen (**PII**)³⁾, Zugriff beschränken, Backups schützen.

0.1) Was ist der ****root**-User** - wofür (nicht)?

root ist der **Superuser** von MySQL: hat **alle** Rechte auf dem Server (Benutzer/Grant-Verwaltung, alle Schemas, alle Objekte). **Verwenden für**: Installation/Initialkonfiguration, **Benutzer/Rollen anlegen, Rechte vergeben, DDL/Administration**, Backups testen. **Nicht verwenden für**: Applikationsbetrieb, tägliche Abfragen, Unterrichtsübungen mit normalem CRUD – hier **dedizierte Rollen/Benutzer** (RO/RW) einsetzen. **Warum?** Minimierung von Risiken (Fehlbedienung, Datenabfluss), bessere Nachvollziehbarkeit (Auditing), saubere Trennung von Verantwortlichkeiten.

1) Begriffe - einfach erklärt

Datenbank vs. Schema In MySQL sind „**Datenbank**“ und „**Schema**“ **synonym** (CREATE DATABASE ...). Eine **Tabelle** ist eine Struktur innerhalb der Datenbank (Zeilen/Spalten).

Benutzer (User) Login-Identität in der Form user'@'host (z. B. app_ro'@'localhost). Steuert **wer** sich **woher** verbinden darf.

Rolle (Role) Bündel aus **Rechten** (Privilegien). Erleichtert Verwaltung: Rechte **einmal** an die Rolle vergeben, Rolle an **viele Benutzer** zuweisen.

Grant (Privileg) Konkrete **Berechtigung**, z. B. SELECT auf eine Tabelle oder ALL PRIVILEGES auf ein Schema. Mit **GRANT/REVOKE** vergeben/entziehen.

1.1) Wo wirken Rechte? (Ebenen)

Ebene	Beispiel	Typische Grants
Global (Serverweit)	alles auf dem Server	CREATE USER, SUPER, RELOAD
Datenbank/Schema	app_demo.*	SELECT, INSERT, ALL PRIVILEGES
Tabelle	app_demo.notes	SELECT, UPDATE
Spalte	notes(owner)	SELECT(owner)
Routine	PROCEDURE p_sync	EXECUTE

Hinweis: Hier folgt in der Stunde eine **Grafik** (Levels-Diagramm).

1.2) Benutzer vs. Rollen vs. Grants - auf einen Blick

Begriff	Kurzdefinition	Beispiel-Befehl
Benutzer	Wer darf sich von wo verbinden?	CREATE USER 'app_ro'@'localhost' IDENTIFIED BY '...';
Rolle	Bündel von Rechten, mehrfach nutzbar	CREATE ROLE 'app_read'; GRANT SELECT ON app_demo.* TO 'app_read';
Grant	Konkrete Berechtigung	GRANT SELECT ON app_demo.* TO 'app_ro'@'localhost';
Zuweisung	Rolle → Benutzer	GRANT 'app_read' TO 'app_ro'@'localhost';
Aktivierung	Standardrolle setzen	SET DEFAULT ROLE 'app_read' TO 'app_ro'@'localhost';

1.3) Hosts - "localhost" vs. "%"

user'@'host bestimmt, **von wo** der Login erlaubt ist. • **localhost**: Zugriffe **vom gleichen Rechner** (oft via Unix-Socket). • %: **beliebiger Host** (Vorsicht, nur wenn nötig!). • **Spezifische Hosts/IPs**: z. B. 'app_ro'@'192.168.1.%' (nur internes Netz). **Best Practice**: so **eng wie möglich** einschränken (z. B. nur App-Server IP), nie unnötig %.

</WRAP> ===== 2) Schritt-für-Schritt: Rollen & Benutzer anlegen =====



Ziel: Demo-Datenbank app_demo + Rollen app_read, app_write, app_admin + passende Benutzer (RO/RW/Admin).
Werkzeug: WebStorm → Datenbank → SQL-Konsole (als root).

2.1 Demo-Schema & Tabelle

```
CREATE DATABASE IF NOT EXISTS app_demo;
USE app_demo;
```

```
CREATE TABLE IF NOT EXISTS notes(
  id INT PRIMARY KEY AUTO_INCREMENT,
  owner VARCHAR(64) NOT NULL,
  body TEXT
);
```

2.2 Rollen definieren (MySQL 8+ / 9.4)

```
CREATE ROLE IF NOT EXISTS 'app_read',
'app_write', 'app_admin';
```

```
GRANT SELECT ON
app_demo.* TO 'app_read';
GRANT SELECT, INSERT, UPDATE, DELETE ON
app_demo.* TO 'app_write';
GRANT ALL PRIVILEGES ON
app_demo.* TO 'app_admin';
```

2.3 Benutzer anlegen (lokal)

```
CREATE USER IF NOT EXISTS 'app_ro'@'localhost'
IDENTIFIED BY 'Str0ng!Ro';
CREATE USER IF NOT EXISTS 'app_rw'@'localhost'
IDENTIFIED BY 'Str0ng!Rw';
CREATE USER IF NOT EXISTS
'app_admin'@'localhost' IDENTIFIED BY
'Str0ng!Admin';
```

2.4 Rollen zuweisen & Standardrolle setzen

```
GRANT 'app_read' TO 'app_ro'@'localhost';
GRANT 'app_write' TO 'app_rw'@'localhost';
GRANT 'app_admin' TO 'app_admin'@'localhost';

SET DEFAULT ROLE 'app_read' TO
'app_ro'@'localhost';
SET DEFAULT ROLE 'app_write' TO
'app_rw'@'localhost';
SET DEFAULT ROLE 'app_admin' TO
'app_admin'@'localhost';
```

2.5 Kontrolle

```
SHOW GRANTS FOR 'app_rw'@'localhost';
```

===== 3) Test: Rollen wirken lassen =====

Als app_ro verbinden (neue Verbindung in WebStorm).
Erwartung: SELECT ☐, INSERT/UPDATE/DELETE ☐.

3.1 Read-Only testen

```
-- als app_ro
USE app_demo;
SELECT * FROM notes; -- sollte funktionieren
INSERT INTO notes(owner, body) VALUES
('lisa', 'hello'); -- sollte fehlschlagen
(permission)
```

3.2 Rechte ändern (root)

```
-- als root
REVOKE 'app_write' FROM 'app_rw'@'localhost';
GRANT 'app_read' TO 'app_rw'@'localhost';
SHOW GRANTS FOR 'app_rw'@'localhost';
```

===== 4) Feingranular & Sicherheit =====

Spaltenrechte

```
GRANT SELECT(id, owner) ON app_demo.notes TO  
'app_ro'@'localhost';
```

Passwort-Policy & Ablauf

```
ALTER USER 'app_rw'@'localhost' PASSWORD EXPIRE  
INTERVAL 90 DAY;  
ALTER USER 'app_rw'@'localhost' ACCOUNT LOCK;  
-- bei Bedarf  
-- ALTER USER 'app_rw'@'localhost' ACCOUNT  
UNLOCK;
```

Benutzer & Hosts auflisten

```
SELECT USER, host FROM mysql.user ORDER BY USER,  
host;
```

===== 5) Mini-Quiz (Kontrollfragen) ===== * Worin unterscheiden sich **Benutzer**, **Rollen** und **Grants** - und wie spielen sie zusammen? * Warum ist GRANT ALL ON *.* für App-User problematisch? * Was bedeutet user'@'localhost vs. user'@'%' - und welche Risiken bestehen?

- Bonus: Warum gehört **root** nicht in den Applikationsbetrieb?

Best Practice



- Keine App unter root betreiben.
- Host einschränken (localhost/IP statt %).
- Rollen verwenden (RO/RW/Admin).
- Passwörter rotieren; Accounts sperren/ablaufen lassen.
- Backups schützen (siehe **LU12B**).
- Im Code **Prepared Statements** verwenden (Thema in der Backend-LU).

1)

RO = Read-Only: darf nur lesen (SELECT).

2)

RW = Read-Write: darf lesen + schreiben (SELECT/INSERT/UPDATE/DELETE).

3)

PII = Personally Identifiable Information: personenbezogene Daten wie Name, E-Mail, Adresse, Geburtsdatum.

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m290_guko/learningunits/lu12/theorie/a_einfuehrung?rev=1762892894

Last update: **2025/11/11 21:28**

