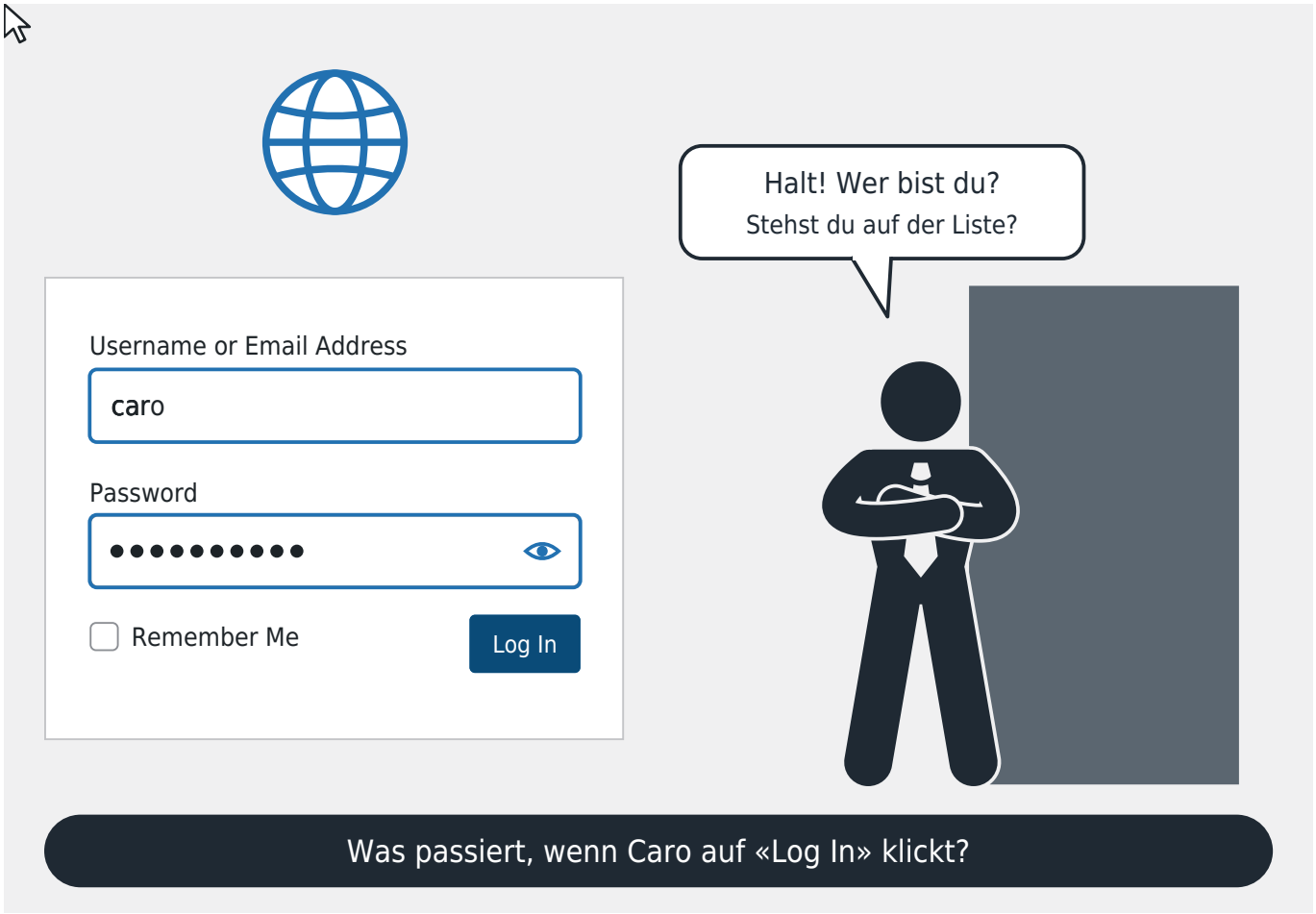
LU12a - Was passiert, wenn wir uns bei WordPress einloggen?

Lernziele

- Ich kann erklären, welche zwei Tabellen beim Login in WordPress im Spiel sind: `wp_users` und `mysql.user`.
- Ich kann erklären, wo MySQL seine Benutzer speichert (Tabelle `mysql.user`) und warum der Benutzer `root` alle Rechte hat, aber für den Alltag nicht verwendet wird.
- Ich kann erklären, wie man in MySQL Rechte für Datenbanken, Tabellen und Befehle vergibt, direkt oder über eine Rolle.

Der Einstieg



Icon «Bouncer» von Gan Khoon Lay, Noun Project, CC BY 3.0

Caro schreibt für den Reiseblog. Sie gibt ihren Benutzernamen und ihr Passwort ein und klickt auf **Log In**. Was passiert jetzt im Hintergrund?

1) Zwei Türsteher, zwei Gästelisten

Video: Master the database checks that secure your blog · 1:05 min · with voice-over

Stellen Sie sich einen Club mit zwei Eingängen hintereinander vor. An jedem steht ein Türsteher mit einer Gästeliste. Er fragt: «Stehst du auf der Liste? Zeig mal deinen Ausweis.» Erst dann darf man hinein.

Beim Login sind genau **zwei** solche Türsteher am Werk:

	Türsteher 1	**Türsteher 2**
Wer prüft?	WordPress	MySQL
Wer will hinein?	Caro, eine Person	WordPress, ein Programm
Gästeliste	Tabelle `wp_users`	Tabelle `mysql.user`
Ausweis	Caros Passwort	das Passwort von WordPress für MySQL
Danach darf man	im Blog schreiben, je nach WordPress-Rolle	Daten lesen und ändern, je nach Rechten

Der Ablauf, wenn Caro auf **Log In** klickt:

1. Caro gibt bei WordPress Benutzername und Passwort ein.
2. Um Caro zu prüfen, braucht WordPress die Liste `wp_users`. Die liegt in der Datenbank. Also meldet sich WordPress selbst bei MySQL an. **Türsteher 2** prüft in `mysql.user`: Darf WordPress hinein?
3. WordPress liest Caros Eintrag aus `wp_users`.
4. **Türsteher 1** vergleicht das Passwort. Es passt: Caro ist angemeldet.

Das Wichtigste dieser Lektion: Caro steht nur auf der Liste von WordPress.
Bei MySQL meldet sich nie Caro an, sondern immer WordPress.

2) Türsteher 1: WordPress prüft wp_users



In `wp_users` stehen alle Personen, die sich bei WordPress anmelden können:

```
SELECT ID, user_login, user_pass, display_name, ROLE
FROM wp_users;
```

ID	user_login	user_pass	display_name	role
1	caro	wp2y10Q46g5lal5WT…	Caro Steig	administrator
2	martin	wp2y10ONzzyboQg0y…	Martin Merten	editor
3	shaolin	wp2y10ZayEi3sAeyz…	Shaolin Tran	editor

- In der Tabelle steht kein Passwort im Klartext, nur sein **Hash**. WordPress berechnet beim Login den Hash des eingegebenen Passworts und vergleicht ihn mit diesem Wert.
- Die Spalte `role` bestimmt, was die Person in WordPress darf. Mehr dazu in Abschnitt 6.

In unserem Reiseblog steht die Rolle direkt in `wp_users`. WordPress selbst speichert sie in einer zweiten Tabelle (`wp_usermeta`). Für das Prinzip spielt das keine Rolle.

Und wenn Caro versucht, sich mit ihrem Benutzernamen direkt bei MySQL anzumelden?

ERROR 1045 (28000): Access denied for user 'caro'@'localhost'

Türsteher 2 kennt Caro nicht. Sie steht auf der falschen Liste.

3) Die Verbindung: Wo steht der Ausweis von WordPress?



Damit WordPress bei MySQL hineinkommt, braucht es einen eigenen MySQL-Benutzer mit Passwort. Bei der Installation trägt man ihn ein, und WordPress speichert ihn in der Datei `wp-config.php`:

```
define( 'DB_NAME', 'wp_reiseblog' );
define( 'DB_USER', 'reiseblog_web' );
define( 'DB_PASSWORD', 'ein-langes-zufaelliges-passwort' );
define( 'DB_HOST', 'localhost' );
```

Es gibt genau **einen** solchen Eintrag, egal ob der Blog drei oder dreitausend Autorinnen und Autoren hat. Alle arbeiten über WordPress, und WordPress arbeitet mit `reiseblog_web`.

4) Türsteher 2: MySQL prüft mysql.user



Die Gästeliste von MySQL ist eine Tabelle in der Datenbank *mysql*. Direkt nach der Installation sieht sie so aus:

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

user	host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

- Die drei Konten mit `mysql.` braucht MySQL intern.
- `root` steht seit der Installation auf der Liste. Mit diesem Konto haben Sie bisher immer gearbeitet.
- `host` gehört zum Namen dazu. `localhost` heisst: Dieser Benutzer darf sich nur vom selben Computer aus anmelden.

Bei jeder Anmeldung fragt Türsteher 2 dreierlei:

Frage des Türstehers	in MySQL
Stehst du auf der Liste?	Gibt es den Benutzer in `mysql.user`?
Zeig deinen Ausweis!	Stimmt das Passwort?
Was darfst du hier drin?	Welche Rechte hat der Benutzer (GRANT)?

Wer auf der Liste steht, darf also noch lange nicht alles. Die Rechte bestimmen, was er drinnen tun darf.

5) Was darf ein Gast? Rechte fein einstellen



In MySQL legen Sie für jeden Benutzer fest, **wo** er etwas darf und **was**.

Wo? Datenbank und Tabelle

Schreibweise	gilt für
`*.*`	alle Tabellen aller Datenbanken
`wp_reiseblog.*`	alle Tabellen der Datenbank *wp_reiseblog*
`wp_reiseblog.wp_posts`	nur die Tabelle *wp_posts*

Was? Die Rechte

	darf	Rechte	Teilsprache
🔍	Daten abfragen	`SELECT`	DML
✏	Daten einfügen, ändern, löschen	`INSERT`, `UPDATE`, `DELETE`	DML
🏗	Tabellen anlegen, ändern, löschen	`CREATE`, `ALTER`, `DROP`	DDL
🔑	neue Gäste auf die Liste setzen und ihnen Rechte geben	`CREATE USER`, `WITH GRANT OPTION`	DCL

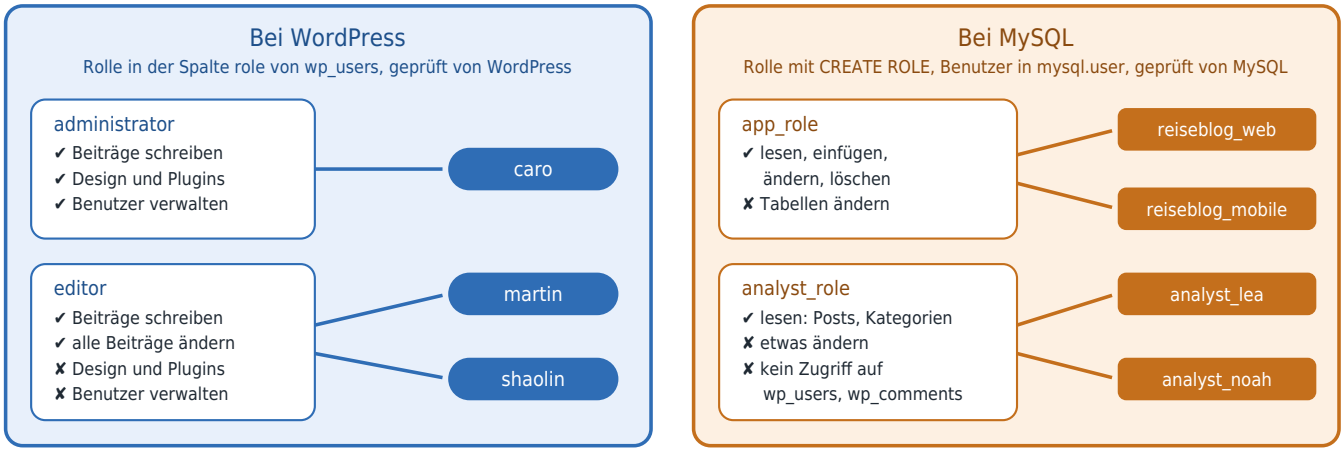
Die Befehle, mit denen man Rechte vergibt und wieder wegnimmt, heissen GRANT und REVOKE. Sie

gehören zur dritten Teilsprache von SQL, der **DCL** (*Data Control Language*). DDL und DML kennen Sie schon.

```
GRANT SELECT ON wp_reiseblog.wp_posts TO
'analyst_lea'@'localhost';
```

Gelesen: Gib das Recht SELECT auf die Tabelle *wp_posts* der Datenbank *wp_reiseblog* an den Benutzer `analyst_lea`.

6) Rollen: mehrere Gäste, dieselben Rechte



Bei WordPress

Martin und Shaolin sind beide **editor**. Sie dürfen Beiträge schreiben und alle Beiträge ändern, aber das Design oder die Plugins nicht anfassen. Caro ist **administrator** und darf alles. Kommt eine dritte Redakteurin dazu, bekommt sie einfach auch die Rolle editor.

Bei MySQL ist es dasselbe

Der Reiseblog hat neben der Website auch eine App fürs Handy. Beide brauchen genau dieselben Rechte: Daten lesen, einfügen, ändern und löschen. Dazu kommen zwei Datenanalysten, Lea und Noah, die nur Posts und Kategorien auswerten.

Die Rechte könnte man jedem Benutzer einzeln geben:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON wp_reiseblog.* TO
'reiseblog_web'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON wp_reiseblog.* TO
'reiseblog_mobile'@'localhost';
```

Das geht, wird aber schnell unübersichtlich. Ändert sich etwas, muss man es bei jedem Benutzer

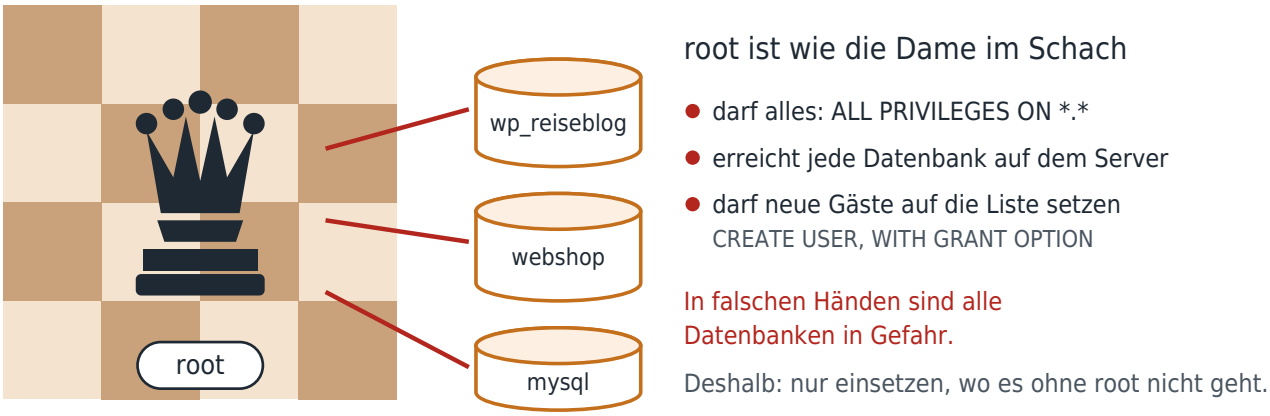
einzeln nachtragen und darf keinen vergessen. Deshalb bündelt man die Rechte in einer **Rolle**:

```
CREATE ROLE app_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON wp_reiseblog.* TO
app_role;
GRANT app_role TO 'reiseblog_web'@'localhost',
'reiseblog_mobile'@'localhost';
```

Eine Rolle ist ein Bündel von Rechten. Wer die Rolle hat, bekommt die Rechte. Das Prinzip ist bei WordPress und MySQL gleich. Der Unterschied: WordPress-Rollen prüft WordPress, MySQL-Rollen prüft MySQL.

Auch Rollen stehen in `mysql.user`. Sie sind dort aber gesperrt und können sich nicht selbst anmelden. Eine Rolle ist kein Gast, sondern eine Farbe für das Eintrittsbändel.

7) root: die Dame im Schach



`root` steht seit der Installation auf der Gästeliste, und er darf alles:

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

*GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, … CREATE USER, … ON *.* TO `root`@`localhost` WITH GRANT OPTION*

Wie die Dame im Schach ist `root` die stärkste Figur. `ON *.*` heisst: auf jeder Datenbank des Servers. `CREATE USER` und `WITH GRANT OPTION` heissen: `root` darf neue Gäste auf die Liste setzen und ihnen Rechte geben. Normalerweise ist er der Einzige, der das darf.

Und wie im Schach gilt: Wer die Dame verliert, ist in grosser Gefahr.

- ⚠ Wer das Passwort von `root` kennt, kann jede Datenbank lesen, ändern oder löschen und sich selbst neue Zugänge anlegen.
- ⚠ Steht `root` in der `wp-config.php` und findet jemand eine Lücke in WordPress, hat er Zugriff auf **alle** Datenbanken des Servers, auch auf solche, die mit dem Blog nichts zu tun haben.
- ⚠ Ein Fehler wie ein DELETE ohne WHERE trifft mit `root` alles.

`root` nur dort einsetzen, wo es ohne nicht geht: beim Einrichten, beim Anlegen von Benutzern und Rollen. Jede Applikation und jede Person bekommt einen eigenen Benutzer mit genau den Rechten, die sie braucht.

8) Das Prinzip der minimalen Rechte

Jeder Benutzer bekommt **nur die Rechte, die er für seine Aufgabe braucht**. Man nennt das auch *Least Privilege*.

- Wer wenig darf, kann bei einem Fehler oder Angriff wenig kaputtmachen.
- Wer bestimmte Daten nicht braucht, sieht sie nicht. Die Analysten zählen Posts und brauchen keine E-Mail-Adressen aus `wp_users`. Also bekommen sie keinen Zugriff darauf.

Auf einen Blick

	Gästeliste von WordPress	**Gästeliste von MySQL**
Tabelle	`wp_users`	`mysql.user`
Wer steht drauf?	Personen: Caro, Martin, Shaolin	Programme und Fachleute: `reiseblog_web`, `analyst_lea`, `root`
Wer prüft?	WordPress	MySQL
Rollen	administrator, editor …	`app_role`, `analyst_role` …
Wer setzt neue Gäste auf die Liste?	ein WordPress-Administrator	`root`

Auf der nächsten Seite **LU12b** richten Sie die Gästeliste von MySQL für den Reiseblog selbst ein.

M290-LU12

Guido Koch

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

Permanent link:
https://wiki.bzz.ch/en/modul/m290_guko/learningunits/lu12/theorie/a_intro

Last update: **2026/11/25 22:55**