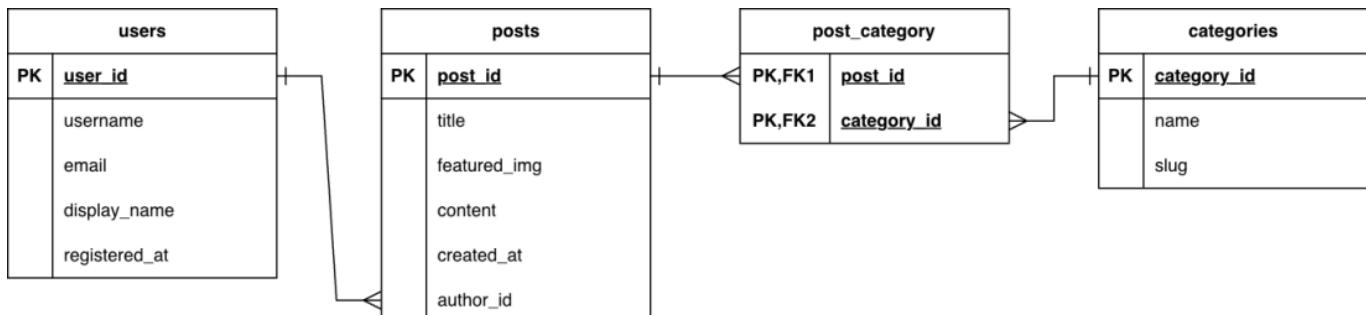


LU08c: Tabellen mit Fremdschlüssel erstellen

Ziel: Wir verteilen die Daten wie bei WordPress auf mehrere Tabellen (**users**, **posts**, **categories** und die N:M-Zuordnung **post_category**) und setzen die Fremdschlüssel direkt beim Erstellen.

ERD (Überblick)

Wir gehen vom Schema aus dem Reiseblog-Beispiel aus:



Posts können mehreren Kategorien angehören (N:M). Die saubere Lösung ist eine Zwischentabelle **post_category**. Das bauen wir später (s. LU08e: N:M-Beziehungen mit Zwischentabelle abbilden).

Fremdschlüssel: Grundsyntax



[SQL Schlüssel einfach erklärt - SQL 3^{1\)}](#) → (7:49, de) Primär- und Fremdschlüssel anschaulich erklärt - gute Ergänzung zur FK-Definition im Reiseblog-Beispiel.

```

CREATE TABLE TABLE_NAME (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    foreign_key_col  INT NOT NULL, -- Datentyp muss zum
                                    -- referenzierten Primary Key (in der anderen Tabelle) passen
    FOREIGN KEY (foreign_key_col) REFERENCES
    parent_table(parent_pk)
);
    
```

Wichtig: Die **referenzierte Tabelle** (parent_table) muss **bereits existieren**, sonst meldet MySQL/MariaDB einen Fehler.

Beispiel Reiseblog



SQL Tabellenenerzeugung mit DDL – SQL 2²⁾ → (7:46, de) Kurz und klar: CREATE TABLE, Datentypen, Constraints – ideal als DDL-Auffrischung vor dem Setzen von FKs.

1. Tabellen anlegen

Tabelle users

```
CREATE TABLE users (
    user_id      INT AUTO_INCREMENT PRIMARY KEY,
    username     VARCHAR(60) NOT NULL UNIQUE,
    email        VARCHAR(191) NOT NULL UNIQUE,
    display_name VARCHAR(100) NOT NULL,
    registered_at DATETIME      NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Tabelle posts

```
CREATE TABLE posts (
    post_id      INT AUTO_INCREMENT PRIMARY KEY,
    title        VARCHAR(200) NOT NULL,
    featured_img VARCHAR(512),
    content       TEXT,
    created_at   DATETIME      NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id    INT           NOT NULL,
    FOREIGN KEY (author_id) REFERENCES users (user_id) --
    Foreign Key wird hier angelegt!
);
```

Tabelle categories

```
CREATE TABLE categories (
    category_id  INT AUTO_INCREMENT PRIMARY KEY,
    name         VARCHAR(100) NOT NULL,
```

```

    slug      VARCHAR(120) NOT NULL UNIQUE -- z. B.
    'schweiz', 'staedtereise'
);

```

2. Beispieldaten einfügen

Für ausführliche Erklärung zum Einfügen von Daten in Tabellen via SQL schauen Sie in „LU07 - DML: Daten einfügen, ändern und löschen“ nach.

```

-- 1) Autor:innen
INSERT INTO users (username, email, display_name) VALUES
    ('caro', 'caro@wetraveltheworld.de', 'Caro Steig'),
    ('martin', 'martin@wetraveltheworld.de', 'Martin
Merten'),
    ('shaolin', 'shaolin@wetraveltheworld.de', 'Shaolin
Tran');

-- 2) Kategorien
INSERT INTO categories (name, slug) VALUES
    ('Städtereise', 'staedtereise'),
    ('Abenteuer', 'abenteuer'),
    ('Roadtrip', 'roadtrip'),
    ('Belgien', 'belgien'),
    ('Niederlande', 'niederlande'),
    ('Portugal', 'portugal'),
    ('Deutschland', 'deutschland'),
    ('Montenegro', 'montenegro'),
    ('Oman', 'oman'),
    ('USA', 'usa');

-- 3) Posts
INSERT INTO posts (title, featured_img, content, created_at,
author_id) VALUES
    ('Hasselt – 10 Highlights',
    'https://wetraveltheworld.de/wp-content/uploads/2025/05/hass
elt.jpg',
    'Kurzguide: Die schönsten Ecken von Hasselt ...',
    '2025-05-07 10:15:00', 2), -- Martin

    ('Utrecht – 10 Sehenswürdigkeiten',
    'https://wetraveltheworld.de/wp-content/uploads/2025/06/utre
cht.jpg',
    'Cafés, Grachten und Restaurant-Tipps ...',
    '2025-06-05 09:30:00', 2), -- Martin

    ('Lissabon – 8 Tipps zu den wichtigsten Sehenswürdigkeiten',
    'https://wetraveltheworld.de/wp-content/uploads/2025/03/liss
abon.jpg');

```

```

abon.jpg',
'Aussichtspunkte, Viertel und Highlights ...',
'2025-03-21 08:40:00', 1), -- Caro

('Maastricht an einem Tag',
'https://wetraveltheworld.de/wp-content/uploads/2025/06/maastricht.jpg',
'Spaziergang, Restaurants, Altstadt ...',
'2025-06-12 11:05:00', 1), -- Caro

('Montenegro Roadtrip – 10 Highlights',
'https://wetraveltheworld.de/wp-content/uploads/2025/06/montenegro.jpg',
'Bucht von Kotor, Durmitor, Tara-Schlucht ...',
'2025-06-11 14:22:00', 1), -- Caro

('Oman – Top 22 Highlights',
'https://wetraveltheworld.de/wp-content/uploads/2025/06/oman.jpg',
'Nizwa, Wadis, Wüste und Roadtrip-Tipps ...',
'2025-06-11 09:00:00', 1), -- Caro

('Chicago in 3 Tagen – 17 Highlights',
'https://wetraveltheworld.de/wp-content/uploads/2024/10/chicago.jpg',
'Riverwalk, The Bean, Museen und Skyline ...',
'2024-10-07 07:50:00', 2); -- Martin

```

Resultate (nach dem Einfügen):

Verknüpfung Tabelle users & posts (one-to-many)



Tabelle categories (wird später mit 'posts' verknüpft)

category_id	name	slug
1	Städtereise	staedtereise
2	Abenteuer	abenteuer
3	Roadtrip	roadtrip
4	Belgien	belgien
5	Niederlande	niederlande
6	Portugal	portugal
7	Deutschland	deutschland
8	Montenegro	montenegro
9	Oman	oman
10	USA	usa

3. Fremdschlüssel in Aktion (Standard: RESTRICT)

Beim Setzen von Fremdschlüsseln überwacht MySQL/MariaDB Änderungen an den verknüpften Spalten und verhindert Operationen, die zu inkonsistenten Verweisen führen. Diese Reaktion heisst *Referenzaktion*. Standardmäßig gilt RESTRICT: Eine Zeile in der Elterntabelle darf nicht gelöscht werden und ihr Primärschlüssel darf nicht geändert werden, solange Kindzeilen auf sie zeigen.

Bezogen auf unser Reiseblog-Beispiel: `posts.author_id` verweist auf `users.user_id`. Damit ist `users` die Elterntabelle und `posts` die Kindtabelle. Die Folge von RESTRICT:

- Löschen eines Users ist blockiert, solange Posts auf diesen User verweisen.
- Ändern von `users.user_id` ist blockiert, solange es verweisende `posts.author_id` gibt.
- Änderungen an nicht referenzierten Spalten (z. B. `users.username`) sind weiterhin erlaubt.

Probieren Sie folgende Codesnippets in Webstorm/MySQL, damit Sie gleich das entsprechende Gefühl dafür bekommen, was RESTRICT passiert.

-- Sicherheit: Welche FKs sind gesetzt?

```
SHOW CREATE TABLE posts;
```

Demo 1 - User ohne Posts löschen (erlaubt)

```
DELETE FROM users
WHERE username = 'shaolin'; -- hat keine Posts zugewiesen
SELECT user_id, username FROM users;
```

Erwartung: Die Zeile wird gelöscht (keine Posts verweisen auf den User).

Demo 2 - User mit Posts löschen (blockiert)

```
DELETE FROM users
WHERE username = 'martin';
```

Erwartete Fehlermeldung (sinngemäss):



[23000][1451] Cannot delete or update a parent row: a foreign key constraint fails (travel_blog.posts, CONSTRAINT posts_ibfk_1 FOREIGN KEY (author_id) REFERENCES users(user_id))

Grund: In posts.author_id gibt es Kindzeilen (z.B. „Hasselt - 10 Highlights“), die auf users.user_id von martin verweisen → RESTRICT verhindert das Löschen.

Demo 3 - Unkritisches Attribut ändern (erlaubt)

```
UPDATE users
SET username = 'martin_travels'
WHERE user_id = 2; -- OK: FKs verweisen auf user_id, nicht auf username
```

Demo 4 - Primärschlüssel ändern (blockiert)

```
UPDATE users
SET user_id = 5
WHERE user_id = 2; -- erwartet: Fehler (RESTRICT), da
posts.author_id -> users.user_id
```

Erklärung: `posts.author_id → users.user_id` ist ein FK mit Standard ON UPDATE RESTRICT. Solange Posts existieren, darf der `user_id`-Wert nicht verändert werden.

Demo 5 - Primärschlüssel ändern: Geht das? (ja)

```
UPDATE categories
SET category_id = 11
WHERE category_id = 10; -- USA → 11: funktioniert - Primary
Keys dürfen geändert werden.
```

Warum ist das so?

`RESTRICT` (Standard) schützt die Elternzeile (z. B. `users`): Solange Kindzeilen (z. B. `posts`) auf sie zeigen, sind `DELETE/UPDATE` am referenzierten Primärschlüssel blockiert.

Änderungen an nicht referenzierten Spalten (z. B. `users.username`) sind frei möglich – FKs verweisen hier nicht darauf.

Ob eine Änderung blockiert oder mitgezogen (CASCADE) wird, hängt von der `ON DELETE/ON UPDATE`-Einstellung im FK ab.

Merke: Fremdschlüssel geben dir Datensicherheit:



- Verhindern verwaiste Daten (z. B. Posts ohne gültigen Autor),
- definieren klares Verhalten bei Löschen/Ändern (RESTRICT, CASCADE, SET NULL),
- halten die Datenbank konsistent.

Ausblick: In **LU08d** fügen wir FKs per **ALTER TABLE** nachträglich hinzu und testen die Referenzaktionen **RESTRICT**, **CASCADE** und **SET NULL** gezielt.

[1\)](#) , [2\)](#)

Informatik – simpleclub / YouTube

Last
update:
2025/10/20 modul:m290_guko:learningunits:lu08:theorie:c_fk-create-table https://wiki.bzz.ch/modul/m290_guko/learningunits/lu08/theorie/c_fk-create-table?rev=1760962901
14:21

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

Permanent link:
https://wiki.bzz.ch/modul/m290_guko/learningunits/lu08/theorie/c_fk-create-table?rev=1760962901

Last update: **2025/10/20 14:21**

