

LU08b: Warum Fremdschlüssel bzw. mehrere Tabellen?

Ausgangslage

Naheliegender ist: **Alles in eine Tabelle** (Post + Autor + Kategorie). In echten Blogs (z. B. WordPress) führt das aber zu **Wiederholungen, Fehlern** und **hohem Wartungsaufwand**.

Wir bleiben beim Reiseblog-Beispiel von der letzten Seite [We Travel The World Blog](#).

Beispiel: Alles in einer Tabelle (schlechte Idee)

```
-- Eine Tabelle für alles: Post + Autor + Kategorie
(redundant!)
CREATE TABLE blog_posts (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  post_title  VARCHAR(200) NOT NULL,
  post_content TEXT,
  author_name VARCHAR(100) NOT NULL,
  author_email VARCHAR(200) NOT NULL,
  category_name VARCHAR(100) NOT NULL,
  created_at  DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Daten einfügen (nur eine Kategorie pro Zeile möglich):

```
INSERT INTO blog_posts
(post_title, post_content, author_name, author_email,
category_name, created_at)
VALUES
('Hasselt – 10 Highlights',
'Kurzguide: Die schönsten Ecken von Hasselt ...',
'Martin Merten', 'martin@wetraveltheworld.de',
'Städtereise', '2025-05-07 10:15:00'),

('Utrecht – 10 Sehenswürdigkeiten',
'Cafés, Grachten und Restaurant-Tipps ...',
'Martin Merten', 'martin@wetraveltheworld.de',
'Städtereise', '2025-06-05 09:30:00'),

('Lissabon – 8 Tipps zu den wichtigsten Sehenswürdigkeiten',
'Aussichtspunkte, Viertel und Highlights ...',
'Caro Steig', 'caro@wetraveltheworld.de', 'Portugal',
```

'2025-03-21 08:40:00');

Warum nur eine Kategorie pro Zeile? Komma-Listen wirken bequem, sind aber ungünstig:

- **Keine Prüfung / kein FK¹⁾**. Die Datenbank (DB) kann bei «Belgien, Städtereise» **nicht** prüfen, ob diese Kategorien wirklich existieren. ²⁾ → Tippfehler bleiben unbemerkt ³⁾.
- **Unzuverlässiges Filtern**. Suchen mit LIKE liefern leicht Teiltreffer/Varianten. Beispiel: *WHERE category_name LIKE '%Guinea%'* trifft auch «Equatorial Guinea», «Guinea», «Guinea-Bissau» und «Papua New Guinea» – vier verschiedene Länder.
- **Schwierig auszuwerten & langsam**. Zählen/Gruppieren erfordert Strings⁴⁾ zu zerlegen; darauf kann die DB nicht sinnvoll indexieren⁵⁾.

Besser: Pro Zeile **eine** Kategorie. Für mehrere Kategorien pro Post (N:M-Beziehung) verwenden wir eine **Zwischentabelle** *post_category*.

Auszug:

id	post_title	author_name	author_email	category_name	created_at
1	Hasselt - 10 Highlights	Martin Merten	martin@wetraveltheworld.de	Städtereise	2025-05-07 10:15:00
2	Utrecht - 10 Sehenswürdigkeiten	Martin Merten	martin@wetraveltheworld.de	Städtereise	2025-06-05 09:30:00
3	Lissabon - 8 Tipps zu den wichtigsten Sehenswürdigkeiten	Caro Steig	caro@wetraveltheworld.de	Portugal	2025-03-21 08:40:00

Probleme auf einen Blick:

- **Redundanz:** Autorname/E-Mail wiederholen sich bei mehreren Posts.
- **Fehleranfällig:** Kategorienamen können unterschiedlich geschrieben werden.
- **Aufwendig:** E-Mail-Wechsel eines Autors → alle Zeilen suchen und ändern.
- **Nur eine Kategorie möglich:** Idealerweise möchten wir aber mehrere Kategorien pro Blog-Post vergeben – z.B. beim Blog-Post «Utrecht - 10 Sehenswürdigkeiten»: Niederlande, Städtereise.

Tippfehler in Kategorie: sichtbare Folgen

Ein fehlender Buchstabe reicht: **Städtereise** vs. **Stätdereise**.

```

INSERT INTO blog_posts
(post_title, post_content, author_name, author_email,
category_name, created_at)
VALUES
('Maastricht an einem Tag',
'Spaziergang, Restaurants, Altstadt ...',
'Caro Steig', 'caro@wetraveltheworld.de', 'Städtereise',
'2025-06-12 11:05:00'); -- Tippfehler!

```

Direkte Folgen in Abfragen:

Abfrage	Zweck	Effekt bei Tippfehler
SELECT DISTINCT category_name FROM blog_posts ORDER BY category_name;	Kategorienliste (Navigation/Filter)	Liste zeigt zwei Einträge: <i>Städtereise</i> und <i>Städtereise</i> .
SELECT id, post_title FROM blog_posts WHERE category_name = 'Städtereise';	Beiträge in „Städtereise“	Der Datensatz mit <i>Städtereise</i> (dt vertauscht) fehlt im Resultat.

Teil-Update: uneinheitliche E-Mail

Nur **eine** von mehreren Zeilen eines Autors wird geändert → inkonsistente Daten.

```

UPDATE blog_posts
SET author_email = 'martin.new@wetraveltheworld.de'
WHERE id = 1;

SELECT id, post_title, author_name, author_email
FROM blog_posts
WHERE author_name = 'Martin Merten';

```

Ergebnis:

id	post_title	author_name	author_email
1	Hasselt - 10 Highlights	Martin Merten	martin.new@wetraveltheworld.de
2	Utrecht - 10 Sehenswürdigkeiten	Martin Merten	martin@wetraveltheworld.de

Gleicher Autor, unterschiedliche E-Mail → Daten sind inkonsistent.

Warum eine N:M-Beziehung (Posts ↔ Kategorien) auflösen?

Ein einzelner Post kann zu mehreren Kategorien gehören (z. B. *Hasselt* → *Belgien* und *Städtereise*). Umgekehrt enthält eine Kategorie viele verschiedene Posts. Das ist eine klassische **N:M-Beziehung**. Versucht man, das ohne Zwischentabelle abzubilden, entstehen typische Probleme:

Variante 1: Duplikate statt Beziehung

id	post_title	author_name	category_name
4	Hasselt - 10 Highlights	Martin Merten	Belgien
5	Hasselt - 10 Highlights	Martin Merten	Städtereise

Hier wird derselbe Post mehrfach angelegt, nur um verschiedene Kategorien zu speichern. Ändert sich der Titel oder der Autor, müssen mehrere Zeilen angepasst werden. Beim Löschen bleibt schnell ein „Zwilling“ übrig – das macht Datenpflege fehleranfällig.

Variante 2: Komma-Liste in einer Spalte

id	post_title	category_name (Text)
4	Hasselt - 10 Highlights	Belgien, Städtereise

In dieser Form steckt mehr als ein Wert in einer Zelle. Die Datenbank kann nicht prüfen, ob „Belgien“ und „Städtereise“ tatsächlich gültige Kategorien sind (kein Fremdschlüssel). Abfragen werden umständlich („LIKE“-Suchen, Trennen von Texten), zählen pro Kategorie ist mühsam und bei grossen Datenmengen langsam.



Bessere Lösung: Wir trennen **posts** und **categories** und führen eine **Zwischentabelle** *post_category* ein. Jede Zeile dieser Tabelle speichert genau **ein Paar** (*post_id*, *category_id*). So bleiben Primärschlüssel eindeutig, Fremdschlüssel prüfbar und Abfragen klar und schnell.

Ausblick

In **LU08c** bauen wir genau dieses Mehrtabellen-Schema **mit Fremdschlüsseln** auf und füllen es mit den obigen Reiseblog-Beispieldaten. In **LU08d** steuern wir dann mit **Referenzaktionen** (*RESTRICT*, *CASCADE*, *SET NULL*), was beim Löschen/Ändern verknüpfter Daten passiert.

1)

Foreign Key = Fremdschlüssel

2)

Wenn Kategorien und Posts in **separate Tabellen** liegen und per Fremdschlüssel verbunden sind, kontrolliert die DB beim Speichern, ob «Belgien» in der Kategorien-Tabelle vorhanden ist.

3)

z. B. «Belgien, Städtereisen» statt «Belgien, Städtereise»

4)

Zeichenketten

5)

Index: Ein Index ist wie ein Inhaltsverzeichnis der Datenbank. Er beschleunigt Suchen/Sortieren auf **einzelnen** Spaltenwerten. Bei Komma-Listen stecken **mehrere** Werte in **einem** Feld – darauf lässt sich kein brauchbarer Index aufbauen; zudem können Suchmuster wie LIKE '%Wort%' einen vorhandenen Index oft nicht nutzen.

From:

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

Permanent link:

https://wiki.bzz.ch/modul/m290_guko/learningunits/lu08/theorie/b_fk-grundlagen?rev=1760818986

Last update: 2025/10/18 22:23

