

LU08d: FKs per ALTER TABLE + Referenzaktionen

Ziel: Bereits bestehende Tabellen mit **ALTER TABLE** um Fremdschlüssel ergänzen und verstehen, was **RESTRICT**, **CASCADE** und **SET NULL** bewirken. **Voraussetzung:** Die Codebeispiele verwenden die Tabellen **users**, **posts**, **categories**, **post_category** inkl. Beispieldaten aus der vorherigen Seite.

0) ALTER TABLE - Kurzüberblick

Mit **ALTER TABLE** können bestehende Tabellen geändert werden – also auch Fremdschlüssel hinzugefügt werden.

Spalte hinzufügen

```
ALTER TABLE TABLE_NAME
ADD COLUMN neue_spalte DATENTYP [NULL|NOT NULL] [AFTER
bestehende_spalte];
```

Fremdschlüssel hinzufügen

```
ALTER TABLE TABLE_NAME
ADD CONSTRAINT fk_name
FOREIGN KEY (fk_spalte)
REFERENCES parent_table(parent_pk)
[ON DELETE {RESTRICT|CASCADE|SET NULL}]
[ON UPDATE {RESTRICT|CASCADE|SET NULL}];
```

1) Beispiel: Redaktor:in (editor_id) in posts SET NULL

Wir ergänzen in **posts** eine **optionale** verantwortliche Redaktor:in (**editor_id**) und verknüpfen sie mit **users**.

1.1 Spalte ergänzen und Beispielwerte setzen

```
-- Spalte hinzufügen
ALTER TABLE posts
ADD COLUMN editor_id INT NULL AFTER author_id;

-- Shaolin wieder hinzufügen, falls gelöscht
INSERT INTO users (username, email,
display_name)
```

```
VALUES('shaolin','shaolin@wetraveltheworld.de','  
Shaolin Tran');
```

```
-- Beispielwerte passend zu user_id: 1=caro,  
2=martin, 3=shaolin  
UPDATE posts SET editor_id = 2 WHERE post_id =  
1; -- Post #1: Editor = martin  
UPDATE posts SET editor_id = 1 WHERE post_id =  
2; -- Post #2: Editor = caro  
UPDATE posts SET editor_id = 3 WHERE post_id =  
3; -- Post #3: Editor = shaolin
```

1.2 Fremdschlüssel setzen - SET NULL beim Löschen

```
ALTER TABLE posts  
ADD CONSTRAINT fk_posts_editor  
FOREIGN KEY (editor_id)  
REFERENCES users (user_id)  
ON DELETE SET NULL -- User gelöscht → Post  
bleibt, Verweis wird NULL  
ON UPDATE RESTRICT; -- Primärschlüssel von  
users bleibt stabil
```



Warum SET NULL? Die Redaktor:in ist **optional**. Wird der zugehörige User gelöscht, soll der Post **nicht** verschwinden – der optionale Verweis fällt auf **NULL**.

Test:

```
-- Lösche User 'shaolin' (ist nur Editor, nicht  
Autor)  
DELETE FROM users WHERE username = 'shaolin';  
  
-- Kontrolle: editor_id des betroffenen Posts  
ist jetzt NULL  
SELECT post_id, title, editor_id FROM posts  
ORDER BY post_id;
```

Absolut – nimm für **CASCADE** in *LU08d* statt der (erst später kommenden) ``post_category`` einfach eine kleine **Kommentare-Tabelle** als Kind von ``posts``.

Wenn ein Post gelöscht wird, sollen seine Kommentare automatisch mitgelöscht werden – genau dafür ist **ON DELETE CASCADE** gemacht.

Ersetze in *LU08d* den bisherigen CASCADE-Abschnitt durch den folgenden:

2) Beispiel CASCADE mit Kommentaren (comments → posts)

Wir fügen eine Kindtabelle **comments** hinzu. Jeder Kommentar gehört zu **genau einem** Post. Wird ein Post gelöscht, sollen die zugehörigen Kommentare **automatisch verschwinden**.

2.1 Tabelle anlegen (mit CASCADE)

```
CREATE TABLE comments (  
  comment_id INT AUTO_INCREMENT PRIMARY KEY,  
  post_id    INT NOT NULL,  
  author     VARCHAR(100) NOT NULL,  
  body       TEXT NOT NULL,  
  created_at DATETIME NOT NULL DEFAULT  
    CURRENT_TIMESTAMP,  
  FOREIGN KEY (post_id)  
    REFERENCES posts (post_id)  
  ON DELETE CASCADE      -- Post gelöscht →  
    zugehörige Kommentare automatisch löschen  
  ON UPDATE CASCADE      -- (optional) ändert sich  
    die post_id, wird sie hier mitgeändert  
);
```

2.2 Kurz befüllen

```
INSERT INTO comments (post_id, author, body,  
  created_at) VALUES  
  (2, 'Leser A', 'Toller Utrecht-Tipp!',  
  '2025-06-05 11:00:00'),  
  (2, 'Leserin B', 'Gute Café-Empfehlungen.',  
  '2025-06-05 12:10:00'),  
  (5, 'Reisefreund', 'Kotor war mein Highlight!',  
  '2025-06-11 16:00:00');
```

****2.3 Test (CASCADE IN Aktion)****

```
<WRAP box round center 80%> <code sql>
```

```
-- Post #2 löschen ...
```

```
DELETE FROM posts WHERE post_id = 2;
```

```
-- ... die zugehörigen Kommentare sind automatisch
```

```
weg:  
SELECT * FROM comments WHERE post_id = 2; -- →  
keine Zeilen  
  
-- Rest bleibt unverändert:  
SELECT post_id,  
FROM comments  
ORDER BY post_id; -- zeigt jetzt nur noch  
post_id = 5 mit 1 Kommentar
```



Warum hier CASCADE? Kommentare ohne zugehörigen Post sind nutzlos. Mit **ON DELETE CASCADE** bleibt die Datenbank **konsistent** und **aufräumen** passiert automatisch.

3) Zusammenfassung

* **RESTRICT**: Löschen/Ändern der Elternzeile **nur** möglich, wenn **keine** Kindzeilen verweisen (Standardverhalten; schützt Datenkonsistenz). * **CASCADE**: Kindzeilen werden bei Änderungen/Löschungen der Eltern **automatisch** mitgeändert oder gelöscht (optimal für Zwischentabellen). * **SET NULL**: Kindzeile bleibt, der **optionale** Verweis wird **NULL**.

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

Permanent link:
https://wiki.bzz.ch/modul/m290_guko/learningunits/lu08/theorie/d_fk-alter-table?rev=1760905456

Last update: **2025/10/19 22:24**

