

LU08c - Fremdschlüssel nachträglich hinzufügen und Referenzaktionen

Lernziele

- Ich kann mit ALTER TABLE eine Spalte und einen Fremdschlüssel zu einer bestehenden Tabelle hinzufügen.
- Ich kann erklären, was RESTRICT, CASCADE und SET NULL beim Löschen bewirken.
- Ich kann für eine Beziehung begründen, welche Referenzaktion passt.

Wozu nachträglich?

Eine Datenbank entsteht selten in einem Zug. Oft gibt es die Tabellen schon, samt Daten, und dann kommt eine neue Spalte oder eine neue Verknüpfung dazu.

Bei Ihren Cases aus LU05 ist das genau so. Dort haben Sie die Spalten für die Fremdschlüssel angelegt, aber noch nicht als Fremdschlüssel gesetzt. Das holen Sie mit **ALTER TABLE** nach.

Wir bleiben beim Reiseblog der letzten Seite. Zur Erinnerung: Caro hat die *user_id* 1, Martin die 2, Shaolin die 3.

ALTER TABLE

Zuerst kommt die neue Spalte in die bestehende Tabelle:

```
ALTER TABLE kind_tabelle  
ADD COLUMN neue_spalte DATENTYP;
```

Danach wird genau diese Spalte zum Fremdschlüssel:

```
ALTER TABLE kind_tabelle  
ADD CONSTRAINT fk_name  
FOREIGN KEY (neue_spalte)  
REFERENCES eltern_tabelle (primaerschluessel)  
ON DELETE referenzaktion;
```

Den *fk_name* wählen Sie selbst, zum Beispiel *fk_posts_editor*. So erkennt man bei Fehlermeldungen, welcher Constraint sich hier bemerkbar macht.

Die drei Referenzaktionen

Mit `ON DELETE` legen Sie fest, was mit den Kindzeilen passiert, wenn ihre Elternzeile gelöscht wird.

Referenzaktion	Was passiert mit den Kindzeilen?	Beispiel
RESTRICT	Nichts. Das Löschen der Elternzeile wird verhindert.	Eine Autorin mit Posts kann nicht gelöscht werden
CASCADE	Sie werden mitgelöscht.	Wird ein Post gelöscht, verschwinden seine Kommentare
SET NULL	Sie bleiben, der Verweis wird leer (NULL).	Eine Redaktorin wird gelöscht, ihre Posts bleiben

Ohne Angabe gilt **RESTRICT**. Das haben wir auf der letzten Seite gesehen, als sich Martin nicht löschen liess.

In MySQL heisst der Standard genau genommen *NO ACTION*. Er verhält sich gleich wie **RESTRICT**.

Neben `ON DELETE` gibt es auch `ON UPDATE`, das beim Ändern eines Primärschlüssels greift. Primärschlüssel werden in der Regel nie geändert, deshalb lassen wir es weg. Es gilt dann ebenfalls **RESTRICT**.

Für die Wahl hilft eine Frage: **Ergibt die Kindzeile ohne ihre Elternzeile noch Sinn?**

- Nein, sie wäre nutzlos. Dann passt **CASCADE**.
- Ja, der Verweis ist nur eine Zusatzinformation. Dann passt **SET NULL**.
- Die Elternzeile soll gar nicht gelöscht werden können, solange Kindzeilen existieren. Dann bleibt es bei **RESTRICT**.

Beispiel 1: SET NULL

Jeder Post bekommt eine Redaktorin, die den Text gegenliest (*editor_id*). Das ist freiwillig. Wird diese Person gelöscht, soll der Post bleiben. Nur der Verweis fällt weg.

Spalte ergänzen und befüllen:

```
ALTER TABLE posts
ADD COLUMN editor_id INT NULL;

UPDATE posts SET editor_id = 2 WHERE post_id = 1; --
Hasselt: Martin liest gegen
UPDATE posts SET editor_id = 1 WHERE post_id = 2; --
Utrecht: Caro liest gegen
UPDATE posts SET editor_id = 3 WHERE post_id = 3; --
```

Lissabon: Shaolin liest gegen

Wichtig ist das NULL bei der Spalte. Ohne dieses dürfte die Spalte nie leer sein, und SET NULL könnte nicht funktionieren.

Fremdschlüssel setzen:

```
ALTER TABLE posts
ADD CONSTRAINT fk_posts_editor
FOREIGN KEY (editor_id)
REFERENCES users (user_id)
ON DELETE SET NULL;
```

Testen:

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

SELECT post_id, title, editor_id FROM posts ORDER BY
post_id;
```

Shaolin ist gelöscht. Den Lissabon-Post gibt es weiterhin, seine *editor_id* ist jetzt NULL.

Shaolin liess sich löschen, weil er bei keinem Post **Autor** ist. Der Fremdschlüssel *author_id* blockiert also nichts. Bei Martin sieht das anders aus: Er ist Autor von drei Posts, und *author_id* steht weiterhin auf RESTRICT. Martin lässt sich deshalb immer noch nicht löschen.

Beispiel 2: CASCADE

Jetzt kommen Kommentare dazu. Jeder Kommentar gehört zu genau einem Post. Ohne seinen Post ergibt ein Kommentar keinen Sinn. Wird der Post gelöscht, sollen seine Kommentare deshalb automatisch mitgehen.

Tabelle anlegen:

```
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
```

```
);
```

Hier steht der Fremdschlüssel direkt im CREATE TABLE, wie auf der letzten Seite. Eine Referenzaktion lässt sich auf beiden Wegen angeben.

Befüllen:

```
INSERT INTO comments (post_id, author, body) VALUES
(2, 'Leser A', 'Toller Utrecht-Tipp!'),
(2, 'Leserin B', 'Gute Café-Empfehlungen.'),
(5, 'Reisefreund', 'Kotor war mein Highlight!');
```

Testen:

```
DELETE FROM posts WHERE post_id = 2;

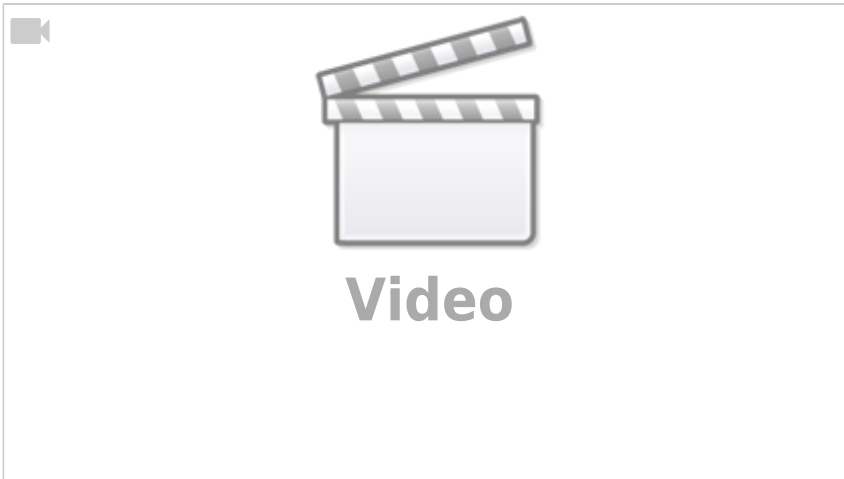
SELECT * FROM comments;
```

Der Utrecht-Post ist weg, und mit ihm seine beiden Kommentare. Übrig bleibt nur der Kommentar zum Montenegro-Post.

CASCADE ist bequem, aber mit Vorsicht zu geniessen. Ein einziger DELETE kann so viele Zeilen in anderen Tabellen mitlöschen. Verwenden Sie es nur dort, wo die Kindzeilen ohne Elternzeile wirklich nutzlos sind.

Zusammenfassung

Referenzaktion	Wann einsetzen?
RESTRICT	Standard. Die Elternzeile soll nicht gelöscht werden, solange Kindzeilen darauf verweisen.
CASCADE	Die Kindzeilen sind ohne Elternzeile nutzlos, zum Beispiel Kommentare.
SET NULL	Der Verweis ist freiwillig und die Kindzeile soll bleiben.



ON DELETE: NO ACTION, SET NULL, CASCADE, SET DEFAULT (9:22 min, deutsch).

Quelle: Prof. Dr. Jens Dittrich / YouTube

Das Video zeigt zusätzlich SET DEFAULT. Diese Option brauchen wir im Modul nicht.

M290-LU08

 Guido Koch

From:

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

Permanent link:

https://wiki.bzz.ch/de/modul/m290_guko/learningunits/lu08/theorie/c_fk-create-table

Last update: **2026/10/17 16:40**

