# LU08A1: Fremdschlüssel per ALTER TABLE + DML üben

**Ziel der Übung:** Ihr greift euren **LU05-Case** wieder auf (z. B. *Tierheim, Postzustellung, Online-Kleidershop, Eishockeyverein, ÖV, Schweizer Regionen*). Damals habt ihr Tabellen **ohne** Fremdschlüssel erstellt und Daten via WebStorm-Interface eingefügt. Jetzt

- 1. ergänzt ihr Fremdschlüssel mit ALTER TABLE,
- 2. testet die Referenzaktionen (Standard: RESTRICT),
- 3. übt DML (INSERT, UPDATE, DELETE) aus LU07.

Arbeitsform: **2er-Gruppen** · Zeit: 1–2 Lektionen · Hilfsmittel: Kursunterlagen LU07/LU08 · Abgabe: siehe unten

# 1) Ausgangsbasis

- Öffnet eure LU05-Datenbank in WebStorm/MySQL.
- Prüft, dass **Primärschlüssel vorhanden** sind (wie in LU05), aber **keine** Fremdschlüssel.
- Falls ihr neu starten wollt: erstellt die Tabellen nochmals wie in LU05 (PK ja, FK noch nicht) und füllt ein paar **Beispieldaten** ein (mind. 3 Zeilen pro Tabelle).
- PDF mit Schema zu den einzelnen Cases:

aufgaben\_crowsfoot\_sql-ddl.pdf

# 2) Fremdschlüssel mit ALTER TABLE hinzufügen

Analysiert euer ERD (Crow's Foot aus LU05). Legt für jede 1:N-Beziehung den Fremdschlüssel fest:

## Beispiele:

- **Tierheim:** tier.art id → tierart.art id, tier.gehege id → gehege.gehege id
- **Postzustellung:** briefkasten.gebiet\_id → postgebiet.gebiet\_id, poestler.gebiet\_id → postgebiet.gebiet id
- Online-Kleidershop: kleidungsstueck.kategorie\_id → kategorie.kategorie\_id, kleidungsstueck.marke\_id → marke.marke\_id
- Eishockeyverein: block.team\_id → team.team\_id, spieler.block\_id → block.block\_id
- ÖV: haltestelle.linie id → linie.linie id, fahrt.linie id → linie.linie id
- Schweizer Regionen: gemeinde.kanton\_code → kanton.kanton\_code, gemeinde.kva\_id → kva.anlage\_id

#### **SQL-Muster (anpassen):**

1. Erstellen Sie eine neue Spalte für den Fremdschlüssel:

ALTER TABLE TABLE\_NAME
ADD COLUMN neue\_spalte DATENTYP [AFTER bestehende\_spalte];

2. Setzen Sie die soeben erstellte Spalte als Fremdschlüssel (Foreign Key):

```
ALTER TABLE kind tabelle
ADD FOREIGN KEY (neue spalte)
REFERENCES eltern tabelle(primaerschluessel spalte)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

#### Hinweise:

- Die **Eltern-Tabelle** (referenzierte Tabelle) muss **vorher existieren**.
- Der **Datentyp** der FK-Spalte muss zum PK der Eltern passen (z. B. beide INT).
- RESTRICT ist ein sicherer Standard: Löschen/Ändern der Elternzeile ist verboten, solange Kindzeilen darauf zeigen.

## 3) DML testen - RESTRICT erfahrbar machen

DML = Data Manipulation Language → Daten einfügen, ändern, löschen

## So gehen Sie vor (für euren gewählten Case):

1. Daten anlegen 2. Tests A-D ausführen und das Verhalten beobachten (Kommentar sagt, was passieren soll).

Hinweis: Falls eure Tabellennamen und Spaltennamen abweichen, dann müsst ihr das entsprechend ändern. Also entweder die hier erwähnten Codesnippets anpassen oder die Spaltennamen bei euch ändern.

**Case: Tierheim (Tierart → viele Tiere)** 

**FK:** tier.art\_id → tierart.art\_id

## **Daten**

```
INSERT INTO tierart (art id, bezeichnung) VALUES (1, 'Hund'),
(2, 'Katze');
INSERT INTO tier (tier id, name, geburtsdatum, geschlecht,
VALUES (101, 'Luna', '2022-05-10', 'w', 1); -- Kind zeigt auf
Hund (1)
```

https://wiki.bzz.ch/ Printed on 2025/11/24 08:56

#### **Tests A-D**

```
-- A) INSERT: gültig vs. ungültig
INSERT INTO tier (tier id, name, geburtsdatum, geschlecht,
art id)
VALUES (102, 'Milo', '2023-01-03', 'm', 2); -- ok (Katze
existiert)
INSERT INTO tier (tier id, name, geburtsdatum, geschlecht,
art id)
VALUES (103, 'Nala', '2023-03-01', 'w', 9999); -- sollte
scheitern (FK!)
-- B) DELETE Eltern mit Kind
DELETE FROM tierart WHERE art_id = 1; -- sollte scheitern
(RESTRICT), weil Tier 101 darauf zeigt
-- C) UPDATE FK im Kind (umhängen)
UPDATE tier SET art_id = 2 WHERE tier_id = 101; -- ok (Katze)
existiert jetzt)
-- D) UPDATE PK in Eltern
UPDATE tierart SET art_id = 5 WHERE art_id = 2; -- sollte
scheitern, solange Tiere auf 2 verweisen
```

## Case: Postzustellung (Postgebiet → viele Briefkästen)

**FK:** briefkasten.gebiet\_id → postgebiet.gebiet\_id

### **Daten**

```
INSERT INTO postgebiet (gebiet_id, name, plz_bereich) VALUES
(10,'Stadt Nord','8000-8099'),
(20,'Stadt Süd','8100-8199');
INSERT INTO briefkasten (briefkasten_id, standort,
leerungszeit, gebiet_id)
VALUES (501,'Bahnhofplatz','18:00',10); -- Kind zeigt auf
Gebiet 10
```

## **Tests A-D**

```
-- A) INSERT
INSERT INTO briefkasten (briefkasten_id, standort, leerungszeit, gebiet_id)
```

```
VALUES (502, 'Zentrum', '17:30', 20); -- ok
INSERT INTO briefkasten (briefkasten id, standort,
leerungszeit, gebiet id)
VALUES (503, 'Park', '17:00', 9999); -- FK-Fehler erwartet
-- B) DELETE Eltern
DELETE FROM postgebiet WHERE gebiet id = 10; -- scheitert
(RESTRICT), weil BK 501 darauf zeigt
-- C) UPDATE FK im Kind
UPDATE briefkasten SET gebiet_id = 20 WHERE briefkasten_id =
501; -- ok (Gebiet 20 existiert)
-- D) UPDATE PK in Eltern
UPDATE postgebiet SET gebiet id = 11 WHERE gebiet id = 20; -
- scheitert, solange Kinder auf 20 zeigen
```

## Case: Online-Kleidershop (Kategorie → viele Kleidungsstücke)

**FK:** kleidungsstueck.kategorie id → kategorie.kategorie id

#### **Daten**

```
INSERT INTO kategorie (kategorie id, name) VALUES
(1, 'Schuhe'), (2, 'T-Shirts');
INSERT INTO kleidungsstueck (artikel id, name, preis,
groesse, zielgruppe, kategorie_id)
VALUES (9001, 'City Sneaker', 89.90, '42', 'Herren', 1);
```

#### **Tests A-D**

```
-- A) INSERT
INSERT INTO kleidungsstueck (artikel id, name, preis,
groesse, zielgruppe, kategorie_id)
VALUES (9002, 'Basic Tee', 19.90, 'M', 'Damen', 2); -- ok
INSERT INTO kleidungsstueck (artikel id, name, preis,
groesse, zielgruppe, kategorie id)
VALUES (9003, 'Ghost Item', 9.90, 'S', 'Damen', 9999); -- FK-
Fehler
-- B) DELETE Eltern
DELETE FROM kategorie WHERE kategorie id = 1; -- scheitert
```

https://wiki.bzz.ch/ Printed on 2025/11/24 08:56

```
(RESTRICT), Schuh 9001 existiert

-- C) UPDATE FK im Kind
UPDATE kleidungsstueck SET kategorie_id = 2 WHERE artikel_id
= 9001; -- ok

-- D) UPDATE PK in Eltern
UPDATE kategorie SET kategorie_id = 5 WHERE kategorie_id = 2; -- scheitert bei referenzierten Zeilen
```

## **Case: Eishockeyverein (Team → viele Blöcke)**

**FK:** block.team\_id → team.team\_id

#### **Daten**

```
INSERT INTO team (team_id, name, altersklasse) VALUES
(100,'U18','U18'), (200,'Herren','Aktiv');
INSERT INTO block (block_id, bezeichnung, team_id) VALUES
(301,'Erste Linie',100);
```

## **Tests A-D**

```
INSERT INTO block (block_id, bezeichnung, team_id) VALUES
(302,'Powerplay',200); -- ok
INSERT INTO block (block_id, bezeichnung, team_id) VALUES
(303,'Penaltykill',999); -- FK-Fehler

-- B) DELETE Eltern
DELETE FROM team WHERE team_id = 100; -- scheitert
(RESTRICT), Block 301 hängt dran

-- C) UPDATE FK im Kind
UPDATE block SET team_id = 200 WHERE block_id = 301; -- ok

-- D) UPDATE PK in Eltern
UPDATE team SET team_id = 250 WHERE team_id = 200; -- scheitert, wenn Blocks auf 200 zeigen
```

## **Case:** Öffentlicher Verkehr (Linie → viele Fahrten)

**FK:** fahrt.linien id → linie.linien id

#### **Daten**

```
INSERT INTO linie (linien_id, name, betreiber) VALUES
(7,'Tram 7','VBZ'), (9,'Bus 9','VBZ');
INSERT INTO fahrt (fahrt_id, datum, abfahrtszeit, preis, linien_id)
VALUES (7001,'2025-11-05','07:45',3.80,7);
```

#### **Tests A-D**

```
INSERT INTO fahrt (fahrt_id, datum, abfahrtszeit, preis,
linien_id)
VALUES (7002,'2025-11-05','08:15',3.80,9); -- ok
INSERT INTO fahrt (fahrt_id, datum, abfahrtszeit, preis,
linien_id)
VALUES (7003,'2025-11-05','09:00',3.80,99); -- FK-Fehler

-- B) DELETE Eltern
DELETE FROM linie WHERE linien_id = 7; -- scheitert
(RESTRICT)

-- C) UPDATE FK im Kind
UPDATE fahrt SET linien_id = 9 WHERE fahrt_id = 7001; -- ok

-- D) UPDATE PK in Eltern
UPDATE linie SET linien_id = 10 WHERE linien_id = 9; -- scheitert bei referenzierten Zeilen
```

## **Case: Schweizer Regionen (Kanton → viele Gemeinden)**

**FK:** gemeinde.kanton id → kanton.kanton id

## **Daten**

```
INSERT INTO kanton (kanton_id, name, einwohnerzahl, flaeche)
```

https://wiki.bzz.ch/ Printed on 2025/11/24 08:56

```
VALUES (1,'ZH',1550000,1729),
(2,'GR',200000,7105);
INSERT INTO gemeinde (gemeinde_id, name, plz, kanton_id)
VALUES (10001,'Zürich','8001',1);
```

#### **Tests A-D**

```
INSERT INTO gemeinde (gemeinde_id, name, plz, kanton_id)
VALUES (10002, 'Chur', '7000', 2); -- ok
INSERT INTO gemeinde (gemeinde_id, name, plz, kanton_id)
VALUES (10003, 'Nowhere', '9999', 999); -- FK-Fehler

-- B) DELETE Eltern
DELETE FROM kanton WHERE kanton_id = 1; -- scheitert
(RESTRICT), Zürich hängt dran

-- C) UPDATE FK im Kind
UPDATE gemeinde SET kanton_id = 2 WHERE gemeinde_id = 10001;
-- ok

-- D) UPDATE PK in Eltern
UPDATE kanton SET kanton_id = 3 WHERE kanton_id = 2; -- scheitert bei referenzierten Zeilen
```

# 4) (Optional) SET NULL oder CASCADE bewusst einsetzen

Falls in **eurem Modell** eine Beziehung **optional** ist (FK darf NULL sein), könnt ihr ON DELETE SET NULL wählen:

```
ALTER TABLE <kind_tabelle>
ADD CONSTRAINT fk_<kind>_<eltern>
FOREIGN KEY (<fk_spalte>)
REFERENCES <eltern_tabelle>(<pk_spalte>)
ON DELETE SET NULL
ON UPDATE RESTRICT;
```

Einsatzbeispiel: team.trainer\_id darf leer sein → beim Löschen eines Trainers bleibt das Team bestehen, trainer id wird **NULL**.

**CASCADE** verwenden wir bevorzugt bei **Zwischentabellen** (N:M). Falls euer Case eine N:M-Erweiterung hat (z. B. im Kleidershop: *Artikel* ↔ *Grösse*), legt ihr die Zwischen-Tabelle wie in **LU08e** an und setzt dort **CASCADE**. Für reine 1:N-Beziehungen bleibt **RESTRICT** meist die beste Wahl.

# 5) Abgabe (pro Gruppe)

## Kurz-Beschrieb (Word-Dokument oder PDF, max. 1 Seite) mit:

- ERD (Crow's Foot) mit PK/FK markiert.
- Liste eurer ALTER TABLE-Befehle (FK-Namen, ON DELETE/UPDATE-Regeln).
- 3-5 **DML-Beispiele** (INSERT/UPDATE/DELETE) inkl. kurzer Interpretation des Verhaltens (z. B. Fehlermeldung bei RESTRICT).

From:

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

Permanent link:

https://wiki.bzz.ch/modul/m290\_guko/learningunits/lu08/aufgaben/a\_fk\_in\_lu05

Last update: 2025/10/27 11:02



https://wiki.bzz.ch/ Printed on 2025/11/24 08:56