

LU09a: SELECT über mehrere Tabellen

Ziel: Sie können Daten aus **mehreren Tabellen** abfragen – mit **INNER JOIN ... ON** (empfohlen) und der **älteren WHERE-Variante**. Sie verstehen, **welche Tabelle in FROM** steht, **welche in JOIN** folgt, und ob die **Reihenfolge** bei mehr als zwei Tabellen eine Rolle spielt.

Voraussetzung: Datenbank aus **LU08** (*users, posts, categories, post_category*) ist erstellt und mit Beispieldaten gefüllt

1. Laden Sie dazu dieses SQL-Skript (gezippt) herunter
Travel Blog DB
2. Entpacken Sie das .zip
3. In **Webstorm** im Datenbank-Plugin **rechts-klicken** auf die Verbindung zu MySQL (z.B. mysql@localhost)
4. Im Dropdown-Menü **SQL-Skripts > Run SQL-Script...** auswählen
5. heruntergeladene .sql-Datei auswählen
6. Nachdem das Skript durchgeführt wurde > **rechts-klicken** auf die Verbindung zu MySQL (z.B. mysql@localhost)
7. **Tools > Manage Shown Schemas...** auswählen und *travel_blog* Datenbank ankreuzen

1) Warum JOINS? Kurze Einordnung

In relationalen DBs verteilen wir Daten auf **mehrere Tabellen**.

In unserem Reiseblog liegen die Infos verteilt:

- Autor:innen → *users*
- Beiträge → *posts* (mit *author_id*)
- Kategorien → *categories*
- Zuordnung Post↔Kategorie → *post_category*

Um zum Beispiel Posttitel und Autor:innenname zusammen zu sehen, müssen wir *posts* und *users* verbinden.

Kindtabelle: posts

post_id	title	featured_img	content	created_at	author_id
1	Hasselt – 10 Highlights	https://wetraveltheworld.de/wp-content/uploads/2025/05/hasselt.jpg	Kurzguide: Die schönsten Ecken von Hasselt ...	2025-05-07 10:15:00	2
2	Utrecht – 10 Sehenswürdigkeiten	https://wetraveltheworld.de/wp-content/uploads/2025/06/utrecht.jpg	Cafés, Grachten und Restaurant-Tipps ...	2025-06-05 09:30:00	2
3	Lissabon – 8 Tipps zu den wichtigsten Sehenswürdigkeiten	https://wetraveltheworld.de/wp-content/uploads/2025/03/lissabon.jpg	Aussichtspunkte, Viertel und Highlights ...	2025-03-21 08:40:00	1
4	Maastricht an einem Tag	https://wetraveltheworld.de/wp-content/uploads/2025/06/maastricht.jpg	Spaziergang, Restaurants, Altstadt ...	2025-06-12 11:05:00	1
5	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
6	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
7	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

Foreign Key (FK)

1:N (one-to-many)
1 user → many posts | 1 post → 1 user

Primary Key (PK)

Elterntabelle: users

user_id	username	email	display_name	registered_at
1	caro	caro@wetraveltheworld.de	Caro Steig	2017-06-13 08:15:00
2	martin	martin@wetraveltheworld.de	Martin Merzen	2018-11-01 08:15:00
3	shaolin	shaolin@wetraveltheworld.de	Shaolin Tran	2018-10-18 14:38:00

SELECT-Abfragen kennen wir bereits für **einzelne Tabellen**:

```
SELECT post_id, title
FROM posts
WHERE post_id = 1;
```

Mögliches Resultat:

post_id	title
1	Hasselt - 10 Highlights

Um nun aber Daten aus mehreren Tabellen gleichzeitig abzufragen, brauchen wir eine erweiterte Syntax.

2) Allgemeine Syntax

Ältere Schreibweise: FROM + WHERE

```
SELECT tabelle1.spalten, tabelle2.spalten
FROM   tabelle1, tabelle2, tabelle3
WHERE  tabelle1.fk = tabelle2.pk
AND    tabelle2.fk = tabelle3.pk
AND    ...           -- weitere Filter
ORDER BY ...;
```

Empfohlen (modern & klar): INNER JOIN ... ON

```
SELECT tabelle1.spalten, tabelle2.spalten
FROM   tabelle1
INNER JOIN tabelle2 ON tabelle1.fk = tabelle2.pk
-- optional weitere Verknüpfungen:
INNER JOIN tabelle3 ON tabelle2.fk = t3.pk
WHERE  ...           -- filtern
ORDER BY ...;       -- sortieren
```

3) Beispiel: Posts mit Autor:in (2 Tabellen)

WHERE-Schreibweise:

```
SELECT posts.post_id, posts.title, users.display_name
FROM posts, users
WHERE posts.author_id = users.user_id;
```

INNER JOIN ... ON (empfohlen):

```
SELECT p.post_id, p.title, u.display_name AS author
FROM posts AS p
INNER JOIN users AS u ON p.author_id = u.user_id;
```



Aliase (p → posts, u → users) verkürzen Schreibarbeit und erhöhen Lesbarkeit.

Mögliches Resultat:

post_id	title	author
1	Hasselt - 10 Highlights	Martin Merten
2	Utrecht - 10 Sehenswürdigkeiten	Martin Merten
3	Lissabon - 8 Tipps zu den wichtigsten Sehenswürdigkeiten	Caro Steig
4	Maastricht an einem Tag	Caro Steig
5	Montenegro Roadtrip - 10 Highlights	Caro Steig
6	Oman - Top 22 Highlights	Caro Steig
7	Chicago in 3 Tagen - 17 Highlights	Martin Merten

4) Drei Tabellen: Posts mit Kategorien (N:M via Zwischentabelle)

Umsetzung einer n:m-Beziehung (Post-Kategorien) mit einer Zwischentabelle



Schritt-für-Schritt mit INNER JOIN (empfohlen):

```
SELECT p.title, c.name AS category
FROM posts p
INNER JOIN post_category pc ON p.post_id = pc.post_id
INNER JOIN categories c ON pc.category_id = c.category_id
ORDER BY p.post_id, c.name;
```

Ausschnitt (mögliche Ausgabe):

title	category
Hasselt – 10 Highlights	Belgien
Hasselt – 10 Highlights	Städtereise
Utrecht – 10 Sehenswürdigkeiten	Niederlande
Utrecht – 10 Sehenswürdigkeiten	Städtereise
Lissabon – 8 Tipps zu den wichtigsten Sehenswürdigkeiten	Portugal
Lissabon – 8 Tipps zu den wichtigsten Sehenswürdigkeiten	Städtereise



Pro Kategorie entsteht **eine Ergebniszeile**. Ein Post mit 3 Kategorien erscheint **dreimal** – das ist korrekt.

5) Welche Tabelle in FROM - und welche in JOIN? Spielt die Reihenfolge eine Rolle?

Grundregel (für INNER JOIN):

- In **FROM** steht die „**führende**“ Tabelle – jene, **deren Zeilen Sie primär auflisten möchten** (z. B. *posts*, wenn Sie Posts auflisten).
- Alles, was Sie **zusätzlich brauchen**, kommt in **JOIN** (z. B. *users* für den Autorname, *categories* via *post_category*).

Reihenfolge bei mehreren INNER JOINs: Bei **INNER JOIN** ändert die Reihenfolge **das Ergebnis nicht**, solange alle Join-Bedingungen korrekt sind. Vom Bedarf her denken (z. B. *Posts anzeigen*), entlang der Schlüsselbeziehungen „weiterjoinen“:

- 1:n: *posts* → *users*
- n:m (via Junction): *posts* → *post_category* → *categories*

6) Filtern & Sortieren - wo kommt die WHERE-Klausel hin?

- **JOIN-Bedingungen** gehören bei der JOIN-Schreibweise in **ON**.
- **Inhaltliche Filter** (z. B. nur bestimmte Autor:innen/Kategorien) kommen in **WHERE**.
- **ORDER BY** bestimmt die Ausgabe-Reihenfolge.

From:

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

Permanent link:

https://wiki.bzz.ch/modul/m290_guko/learningunits/lu09/theorie/a_select_multiple_tables

Last update: 2025/10/26 23:05

