

LU05a - Datentypen in MySQL

Lernziele

- Ich kann erklären, warum jede Spalte einen Datentyp braucht.
- Ich kenne die wichtigsten MySQL-Datentypen und weiss, wofür sie eingesetzt werden.
- Ich kann für eine gegebene Information den passenden Datentyp auswählen und meine Wahl begründen.

Warum Datentypen?

In LU04 haben wir geplant, **welche** Tabellen und Spalten es geben soll. Bevor wir die Tabellen anlegen, müssen wir für jede Spalte festlegen, **welche Art von Werten** dort hineindarf.

Erinnern Sie sich an LU02: Beim Import von *imdb_top_1000* hat WebStorm **alle** Spalten als Text angelegt. Das hat Folgen:

```
SELECT Series_Title, Meta_score
FROM imdb_top_1000
ORDER BY Meta_score DESC;
```

Diese Abfrage zeigt ganz oben Filme mit dem Wert **99** – obwohl es Filme mit **100** gibt. Der Grund: Text wird **alphabetisch** sortiert. Wie *Anna* vor *Berta* kommt, kommt *100* vor *99*, weil die 1 vor der 9 steht. Wäre die Spalte als Zahl angelegt, würde korrekt sortiert.

Der richtige Datentyp sorgt also dafür, dass

- **richtig sortiert und gerechnet** wird,
- **falsche Eingaben abgewiesen** werden (ein Geburtsdatum kann kein Text sein),
- **Speicherplatz sinnvoll** genutzt wird und Abfragen schnell bleiben.

Die wichtigsten Datentypen

Wir gehen nicht die ganze MySQL-Dokumentation durch, sondern die Typen, die Sie in diesem Modul tatsächlich brauchen. Als durchgehendes Beispiel dient eine **Bibliothek** mit einer Tabelle *buch*.

Zahlen

2.3

Typ	Wofür	Beispiel in der Tabelle buch
INT	Ganze Zahlen ohne Nachkommastellen	<i>seiten</i> (350), <i>buch_id</i> (1, 2, 3 ...)
DECIMAL(M,D)	Kommazahlen mit exakter Genauigkeit. M = Stellen gesamt, D = Nachkommastellen	<i>preis</i> als DECIMAL(6,2) → bis 9999.99
FLOAT / DOUBLE	Kommazahlen mit ungefährer Genauigkeit	Messwerte, Sensordaten – bei uns selten

Geldbeträge immer DECIMAL, nie FLOAT.

FLOAT speichert Zahlen nur näherungsweise. Bei Preisen führt das zu Rundungsfehlern, die sich über viele Datensätze aufsummieren – in einer Rechnung fehlen dann plötzlich ein paar Rappen.

Text

Tt

Typ	Wofür	Beispiel in der Tabelle buch
VARCHAR(n)	Text mit variabler Länge, bis n Zeichen. Der Normalfall für Text.	<i>titel</i> als VARCHAR(100)
CHAR(n)	Text mit immer gleicher Länge	<i>isbn</i> als CHAR(13), Ländercodes wie CH oder FR
TEXT	Sehr lange Texte ohne feste Obergrenze	Inhaltsangabe, Rezension

Faustregel: **VARCHAR ist die Standardwahl.** CHAR nur, wenn die Länge wirklich immer gleich ist. TEXT nur für lange Fliesstexte.

Datum und Zeit

12.8.25
20:00

Typ	Wofür	Format
DATE	Nur ein Datum	2026-09-07 (YYYY-MM-DD)
TIME	Nur eine Zeit	14:30:00 (hh:mm:ss)
DATETIME	Datum und Uhrzeit	2026-09-07 14:30:00

Typ	Wofür	Format
TIMESTAMP	Wie DATETIME, kann sich beim Ändern eines Datensatzes automatisch aktualisieren	2026-09-07 14:30:00
YEAR	Ein Jahr	1965

Die Frage lautet immer: **Brauche ich die Uhrzeit?** Bei einem Erscheinungsdatum nicht (DATE), bei einer Ausleihe schon (DATETIME).

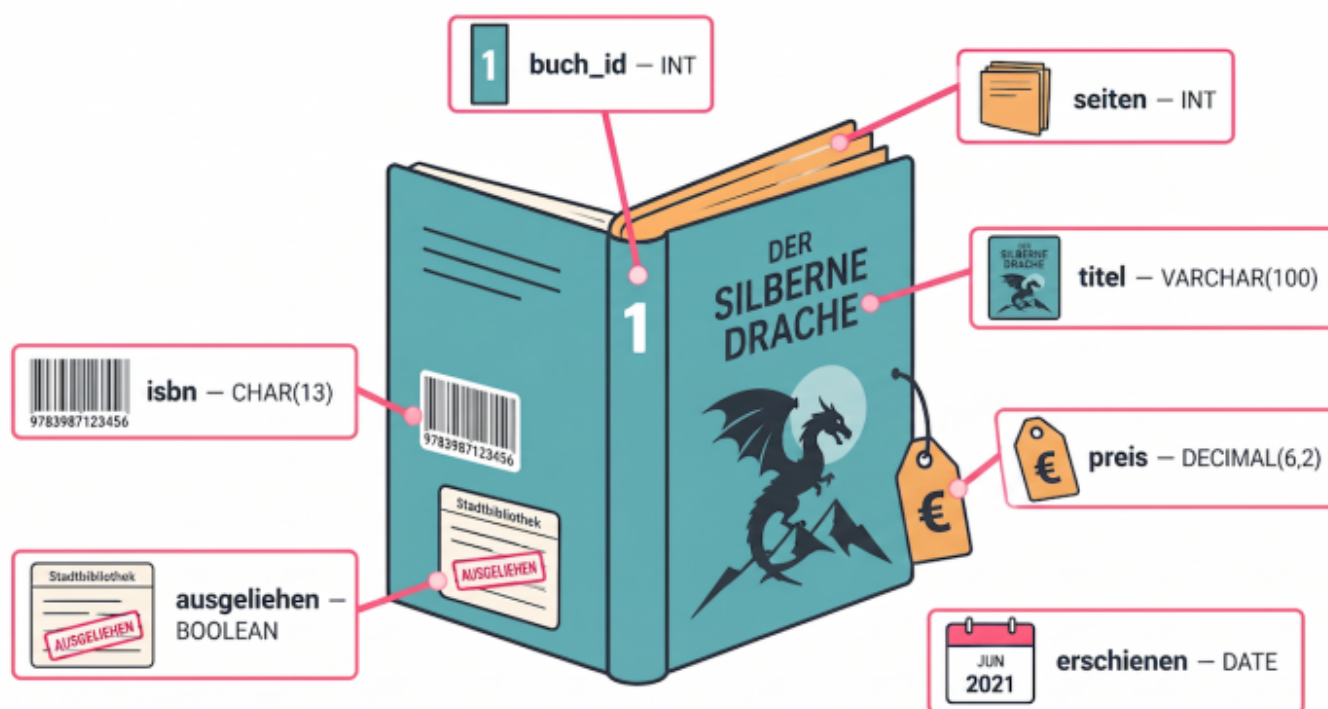
Ja oder Nein

true
false

Typ	Wofür	Werte
BOOLEAN	Genau zwei Zustände	true/false bzw. 1/0

Beispiel: *ausgeliehen* – ein Buch ist ausgeliehen oder nicht. MySQL speichert BOOLEAN intern als TINYINT, deshalb sehen Sie in der Tabelle 0 und 1.

Beispiel Buch



Spalte	Datentyp	Warum?
buch_id	INT	Fortlaufende ganze Zahl
titel	VARCHAR(100)	Titel sind unterschiedlich lang

Spalte	Datentyp	Warum?
isbn	CHAR(13)	Eine ISBN hat immer genau 13 Stellen
seiten	INT	Ganze Zahl, keine halben Seiten
preis	DECIMAL(6,2)	Geldbetrag, exakt auf zwei Nachkommastellen
erschienen	DATE	Datum ohne Uhrzeit
ausgeliehen	BOOLEAN	Ja oder nein

Häufige Fehler

- **Preis als FLOAT** → Rundungsfehler. Nehmen Sie DECIMAL.
- **Telefonnummer als INT** → führende Nullen verschwinden (aus 0791234567 wird 791234567). Telefonnummern sind Text, nicht Zahlen – man rechnet nie damit.
- **Datum als VARCHAR** → man kann weder sortieren noch Zeiträume berechnen.

Merken Sie sich die Faustregel: **Rechnet man damit, ist es eine Zahl. Sortiert man danach zeitlich, ist es ein Datum. Alles andere ist Text.**

Hinweis

Jedes DBMS (MySQL, PostgreSQL, Oracle ...) bietet leicht unterschiedliche Datentypen. In diesem Modul arbeiten wir mit **MySQL** und beschränken uns auf die hier vorgestellten Typen. Eine detaillierte Übersicht über alle MySQL-Datentypen finden Sie auf der offiziellen Dokumentation: [MySQL Docs Datentypen](#)

M290-LU05

 Guido Koch

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

Permanent link:
https://wiki.bzz.ch/de/modul/m290_guko/learningunits/lu05/theorie/a_ddl

Last update: **2026/09/14 09:31**

