LU05c - Constraints (Einschränkungen)

Lernziele

- Ich kann erklären, was Constraints sind und wozu sie dienen.
- Ich kann PRIMARY KEY, AUTO_INCREMENT, NOT NULL und UNIQUE beim Erstellen einer Tabelle setzen.
- Ich kann begründen, welcher Constraint für welche Spalte sinnvoll ist.

Was sind Constraints?

Am Ende der letzten Seite stand ein Problem: Unsere Tabelle *buch* erlaubt zwei Bücher mit derselben *buch_id* und ein Buch ganz ohne Titel. Die Datenbank prüfte so unsere Eigaben nicht – das wollen wir nun ändern.

Constraints sind Regeln für Spalten. Sie sorgen dafür, dass fehlerhafte Daten gar nicht erst gespeichert werden können. Die Datenbank weist einen Eintrag ab, der gegen eine Regel verstösst.

Constraints kurz erklärt - 0:56min [1)]

PRIMARY KEY

Der Primärschlüssel identifiziert jeden Datensatz **eindeutig**. Jede Tabelle kann genau **einen** Primärschlüssel haben.

```
CREATE TABLE buch (
  buch_id INT PRIMARY KEY,
  titel VARCHAR(100)
);
```

Wirkung: Keine zwei Bücher dürfen dieselbe *buch_id* haben. Ausserdem darf die *buch_id* nie leer (NULL) sein – ein Primärschlüssel schliesst NOT NULL automatisch mit ein.

AUTO_INCREMENT

Bisher müssten wir jede *buch_id* selbst vergeben – und dabei aufpassen, keine Nummer doppelt zu verwenden. Das übernimmt **AUTO_INCREMENT**: MySQL vergibt bei jedem neuen Datensatz automatisch die nächste freie Nummer.

```
CREATE TABLE buch (
  buch_id INT AUTO_INCREMENT PRIMARY KEY,
  titel VARCHAR(100)
);
```

Wirkung: Das erste Buch bekommt die 1, das zweite die 2, und so weiter. Sie müssen die *buch_id* beim Einfügen gar nicht mehr angeben.

AUTO_INCREMENT wird fast immer zusammen mit dem Primärschlüssel verwendet.

NOT NULL

NOT NULL legt fest, dass eine Spalte nicht leer bleiben darf – ein Pflichtfeld.

```
CREATE TABLE buch (
  buch_id INT AUTO_INCREMENT PRIMARY KEY,
  titel VARCHAR(100) NOT NULL,
  seiten INT
);
```

Wirkung: Ein Buch ohne Titel wird abgewiesen. Ein Buch ohne Seitenzahl ist dagegen erlaubt, weil bei *seiten* kein NOT NULL steht.

Fragen Sie sich bei jeder Spalte: **Ergibt ein Datensatz ohne diesen Wert überhaupt Sinn?** Wenn nein, gehört NOT NULL dorthin.

UNIQUE

UNIQUE verhindert, dass ein Wert in einer Spalte doppelt vorkommt.

```
CREATE TABLE buch (
  buch_id INT AUTO_INCREMENT PRIMARY KEY,
  titel VARCHAR(100) NOT NULL,
  isbn CHAR(13) UNIQUE
);
```

Wirkung: Dieselbe ISBN kann nur einmal eingetragen werden.

Unterschied zum PRIMARY KEY

	PRIMARY KEY	**UNIQUE**
Wie oft pro Tabelle?	genau auf einer Spalte	bei beiligen vielen Spalten

	PRIMARY KEY	**UNIQUE**
Darf leer sein?	nein	ja
Zweck	identifiziert den Datensatz	verhindert Doppeleinträge

Eine ISBN wäre theoretisch auch als Primärschlüssel geeignet. In der Praxis nimmt man trotzdem meist eine eigene ID-Spalte: Sie ist kürzer, ändert sich nie und existiert auch dann, wenn ein Buch ausnahmsweise keine ISBN hat.

Alles zusammen

```
CREATE TABLE buch (
  buch_id INT AUTO_INCREMENT PRIMARY KEY,
  titel VARCHAR(100) NOT NULL,
  isbn CHAR(13) UNIQUE,
  seiten INT,
  preis DECIMAL(6,2),
  erschienen DATE,
  ausgeliehen BOOLEAN NOT NULL
);
```

Vergleichen Sie das mit der Fassung von der letzten Seite: Dieselbe Tabelle, aber die Datenbank passt jetzt auf, welche Daten wir eingeben.

Übersicht

Constraint	**Bedeutung**
PRIMARY KEY	Eindeutige Identifikation, nie leer, einmal pro Tabelle
AUTO_INCREMENT	Automatisch fortlaufende Nummer
NOT NULL	Spalte darf nicht leer bleiben
UNIQUE	Wert darf in dieser Spalte nur einmal vorkommen

Es gibt einen weiteren wichtigen Constraint: den **FOREIGN KEY**, mit dem zwei Tabellen verknüpft werden. Sie haben ihn in LU04 bereits in Ihre Diagramme eingezeichnet. In SQL setzen wir ihn erst später im Modul – bis dahin bleiben unsere Tabellen einzeln.

M290-LU05

Guido Koch

1)

Quelle: KnowledgeBits Education/YouTube