LS4-2: SQL-Data Definition Language (DDL)

# Situation

Nach den ersten Funktionstests haben Sie festgestellt, dass die Datenbank nicht vollständig implementiert ist. Das relationale Datenbankmodell weist neben den bereits im System hinterlegten Tabellen noch zwei weitere Tabellen auf.

Herzlichen Dank für den ausführlichen Funktionstest. Wie Sie festgestellt haben, wurden bei der Umsetzung der Datenbank die Tabellen ‚mindestbestellmenge‘ und ‚lieferer‘ nicht berücksichtigt.

Ich bitte Sie daher, die implementierte SQL-Datenbank zu erweitern und die beiden Tabellen hinzuzufügen. Berücksichtigen Sie bei der Umsetzung das relationale Datenbankmodell, um für beide Tabelle alle Attribute und alle Beziehungen zu hinterlegen.

Welches Problem/Aufgabe stellt sich dar?

Welche Lösungsmöglichkeit(n) besteht(en)?

# Handlungsaufträge

1. Informieren Sie sich über die SQL-Befehle der ‚Data Definition Language‘ (DDL).

2. Lösen Sie die vorbereitenden Übungsaufgaben zur DDL.

3. Erstellen Sie die Tabellen ‚mindestbestellmenge‘ und ‚lieferer‘ mit deren Attributen und Beziehungen

5. Erstellen Sie die Beziehungen zu den Nachbartabellen.

Datenbank erweitern

Die aktuelle Datenbank des Online-Shops ist noch unvollständig. Damit sie vollständig wird, müssen die Tabellen ‚mindestbestellmenge‘ und ‚lieferer‘ hinzugefügt werden.

1. **Erstellen Sie die Tabellen ‚mindestbestellmenge‘ und ‚lieferer‘. Beachten Sie dabei folgende Konfiguration:**

* Datentypen ‚mindestbestellmenge‘:  
  *artikel\_id = integer, lieferer\_id = integer, mindestbestellmenge = double;*
* Datentypen ‚lieferer‘:  
  *lieferer\_id = integer, firmenname = varchar (255), strasse = varchar (50), plz\_id = varchar (11), liefervertragsbeginn = date, laufzeitjahre = integer;*
* NOT NULL-Werte ‚mindestbestellmenge‘: *artikel\_id, lieferer\_id*;
* NOT NULL-Werte ‚lieferer‘: *lieferer\_id, firmenname, strasse*;
* Belegen Sie außerdem das Attribut *‚lieferer\_id‘* als Primärschlüssel

1. **Stellen Sie die Beziehungen zu den Nachbartabellen her. Legen Sie dazu die Fremdschlüssel fest.**

artikel

**artikel\_id (PK)**

bezeichnung

**warengruppe\_id (FK)**

geschlecht

einkaufspreis

nettoverkaufspreis

mindestbestellmenge

**artikel\_id (FK)**

**lieferer\_id (FK)**

mindestbestellmenge

lieferer

**lieferer\_id (PK)**

firmenname

strasse

**plz\_id (FK)**

liefervertragsbeginn

laufzeitjahre

plz

**plz\_id (PK)**

ort

1

n

n

1

n

1

Tabelle vorhanden

Tabelle fehlt

Tabelle fehlt

Tabelle vorhanden

**Datenbanktyp InnoDB vs. MyISAM**

**ACHTUNG!** Wenn Sie mit phpmyadmin die Tabellen ‚lieferer‘ bzw. ‚mindestbestellmenge‘ erzeugen, dann werden diese vermutlich mit dem Datenbanktyp „InnoDB“ deklariert. Ist dies der Fall, so können keine Fremdschlüsselbeziehungen zu den bestehenden Tabellen (‚artikel‘ bzw. ‚plz‘) hergestellt werden, weil diese mit dem Datenbanktyp „MyISAM“ deklariert sind (detaillierte Erläuterung: <https://bit.ly/3OCDo67>).

Vorgehensweise: Wandeln Sie die Engine der entsprechenden Tabellen (‚lieferer‘ bzw. ‚mindestbestellmenge‘) mit folgendem SQL-Statement um:

**alter table mindestbestellmenge ENGINE=myisam;**

Lösung zu 1 (Tabelle ‚mindestbestellmenge‘)

CREATE TABLE mindestbestellmenge

( artikel\_id INTEGER NOT NULL,

lieferer\_id INTEGER NOT NULL,

mindestbestellmenge DOUBLE);

Lösung zu 1 (Tabelle ‚lieferer‘)

CREATE TABLE lieferer

( lieferer\_id INTEGER NOT NULL,

firmenname VARCHAR (255) NOT NULL,

strasse VARCHAR (50) NOT NULL,

plz\_id INTEGER (11),

liefervertragsbeginn DATETIME,

laufzeitjahre INTEGER,

CONSTRAINT PK\_lieferer PRIMARY KEY (lieferer\_id));

Lösung zu 2 (Fremdschlüssel ‚artikel\_id‘)

ALTER TABLE mindestbestellmenge ADD

CONSTRAINT FK\_mbmart FOREIGN KEY (artikel\_id)

REFERENCES artikel (artikel\_id);

Lösung zu 2 (Fremdschlüssel ‚lieferer\_id‘)

ALTER TABLE mindestbestellmenge ADD

CONSTRAINT FK\_mbmlief FOREIGN KEY (lieferer\_id)

REFERENCES lieferer (lieferer\_id);

Lösung zu 2 (Fremdschlüssel ‚plz\_id‘)

ALTER TABLE lieferer ADD

CONSTRAINT FK\_artplz FOREIGN KEY (plz\_id)

REFERENCES plz (plz\_id);

SQL – Data Definition Language (DDL)

# Structured Query Language (SQL)

SQL (= Structured Query Language) ist eine standardisierte Datenmanipulations- und abfragesprache. Sie enthält alle erforderlichen Sprachelemente, um sämtliche Aktionen im Umgang mit relationalen Datenbanken ausführen zu können.

SQL-Befehle lassen sich in vier Kategorien einteilen:



# DDL-Befehle zur Definition der Datenstruktur

## Anlegen: CREATE

### Anlegen von Datenbanken: CREATE DATABASE

Mittels SQL lassen sich Datenbanken anlegen, die dann als Container für alle Tabellen dienen.

Syntax Datenbank anlegen:

CREATE DATABASE [IF NOT EXISTS] datenbank;

Beispiel:

CREATE DATABASE dbUnternehmen;

Optional kann nach dem Befehl CREATE DATABASE der Zusatz IF NOT EXISTS eingefügt werden. Damit werden Fehlermeldungen vermieden, wenn die Datenbank bereits existiert.

### Anlegen von Tabellen: CREATE TABLE

Mit der SQL-Anweisung CREATE TABLE wird eine Tabelle mit allen dazugehörigen Spalten und deren Datentypen erzeugt.

Syntax Tabelle anlegen *(leichte Abweichungen je nach DBMS sind möglich)*:

CREATE TABLE [IF NOT EXISTS] tabellenname

( spaltenname\_1 datentyp\_1 [NOT NULL],

[ spaltenname\_2 datentyp\_2 [NOT NULL],... );

*Mit dem Schlüsselwort NOT NULL bzw. NULL sorgen Sie dafür, dass eine Eingabe verpflichtend bzw. optional wird.*

Beispiel:

CREATE TABLE tblArtikel

( A\_Nr LONG NOT NULL,

A\_Bez TEXT(30) NOT NULL,

A\_Preis CURRENCY);

Mögliche Datentypen in SQL (Auswahl):

|  |  |
| --- | --- |
| Schlüsselwort | Beschreibung |
| COUNTER | Autowert |
| CURRENCY (nicht in MySQL - hier für Währung 'DECIMAL(10,2)' nutzen) | Währung |
| DATE | Datum. Format ‘YYYY-MM-DD’ |
| TIME | Zeitangabe. Format ‘HH:MM:SS’ |
| DOUBLE | Große Fließkommazahl |
| INTEGER (INT) | Integer normaler Größe |
| LONG | Long Integer |
| TEXT(Zeichenzahl) | Normaler Text |
| BOOLEAN | True/False |
| VARCHAR | Zeichenkette/Bytefolge |

### Einfügen von Primärschlüsseln

1. Einfügen eines Primärschlüssels: CREATE TABLE CONSTRAINT PRIMARY KEY

Die PRIMARY KEY-Klausel kann sowohl beim Erstellen von Tabellen als auch nachträglich durch Hinzufügen eines Feldes geschehen.

Syntax Primärschlüssel anlegen:

CREATE TABLE tabellenname

( spaltenname\_1 datentyp\_1, [spaltenname\_2 datentyp\_2],

CONSTRAINT regelname PRIMARY KEY (spaltenname\_x));

Beispiel *(Kundentabelle mit Kundennummer und -namen; Kundennummer als Primärschlüssel)*:

CREATE TABLE tblKunde

(K\_Nr LONG, K\_Name TEXT(20),

CONSTRAINT PK\_Kunde PRIMARY KEY (K\_Nr));

1. Nachträgliches Einfügen eines Primärschlüssels: ALTER TABLE CONSTRAINT PRIMARY KEY

Syntax Primärschlüssel nachträglich anlegen  
*(Primärschlüssel wird nachträglich als neue Spalte eingefügt)*:

ALTER TABLE tabellenname ADD COLUMN spaltenname\_1 datentyp\_1

CONSTRAINT regelname PRIMARY KEY;

Beispiel:

ALTER TABLE tblProjekt ADD COLUMN P\_Nr LONG

CONSTRAINT PK\_Projekt PRIMARY KEY;

Syntax Primärschlüssel nachträglich anlegen  
*(Primärschlüssel-Befehl auf ein bestehendes Attribut (Spalte) anwenden)*:

ALTER TABLE tabellenname ADD

CONSTRAINT regelname PRIMARY KEY (spaltenname\_x);

Beispiel:

ALTER TABLE tblProjekt ADD

CONSTRAINT PK\_Projekt PRIMARY KEY(P\_Nr);

1. Erstellen von Beziehungen: CREATE TABLE CONSTRAINT FOREIGN KEY REFERENCES

Mit Hilfe der FOREIGN KEY-Klausel und der REFERENCES-Klausel kann ein Fremdschlüssel in einer Tabelle erzeugt werden.

Syntax Fremdschlüssel anlegen und Beziehung herstellen:

CREATE TABLE tabellenname

( spaltenname\_1 datentyp\_1, [spaltenname\_2 datentyp\_2],

CONSTRAINT regelname FOREIGN KEY (spaltenname\_y)

REFERENCES tabellenname (spaltenname\_y));

Beispiel *(zwei Tabellen erstellen, Primärschlüssel vergeben und verknüpfen)*:

CREATE TABLE tblKunde (K\_Nr LONG, K\_Name TEXT(10),

CONSTRAINT PK\_Kunde PRIMARY KEY (K\_Nr));

CREATE TABLE tblAuftrag (A\_Nr LONG, K\_Nr LONG,

CONSTRAINT PK\_Auftrag PRIMARY KEY (A\_Nr),

CONSTRAINT FK\_AufKun FOREIGN KEY (K\_Nr)

REFERENCES tblKunde (K\_Nr));

## Änderungen an den Tabellenstrukturen: ALTER TABLE

Von Zeit zu Zeit wird es nötig sein, eine bereits bestehende Datentabelle um eine Spalte zu erweitern.

Syntax Tabelle um Spalte erweitern:

ALTER TABLE tabellenname ADD COLUMN spaltenname\_1 datentyp\_1;

Beispiel:

ALTER TABLE tblArtikel ADD COLUMN A\_Katalogseite INTEGER;

Diese Anweisung fügt der Tabelle tblArtikel die Spalte A-Katalogseite hinzu. Bei einer Änderung der Tabellenstruktur ist die Option NOT NULL nicht verfügbar, da die Felder der neuen Spalte für alle existierenden Datensätze nur mit NULL-Werten belegt werden kann.

## Löschen: DROP

### Löschen von Spalten: DROP COLUMN

Syntax Spalte löschen:

ALTER TABLE tabellenname DROP COLUMN spaltenname\_1;

Beispiel:

ALTER TABLE tblArtikel DROP COLUMN A\_Preis;

### Löschen von Tabellen: DROP TABLE

Syntax Tabellen löschen:

DROP TABLE tabellenname;

Beispiel:

DROP TABLE tblArtikel;

### Löschen von Datenbanken: DROP DATABASE

Syntax Datenbanken löschen:

DROP DATABASE datenbank;

Beispiel:

DROP DATABASE dbVertrieb;

Auch hier gilt wieder, um Fehlermeldungen zu vermeiden, kann nach DROP DATABASE der Zusatz IF EXISTS eingesetzt werden.

SQL – Data Definition Language

1. **Befehl CREATE DATABASE:**

CREATE DATABASE dbLEIT-Systems;

Ergebnis des Befehls:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Erläuterung des Befehls:

1. **Befehl CREATE TABLE:**

Ergebnis des Befehls:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  | **tblMitarbeiter** |  |  |  |  |
|  | M\_Nr | M\_Name |  |  |  |

Erläuterung des Befehls:

Erstellung Tabelle tblMitarbeiter mit den Spalten

M\_Nr (Ganzzahl) und M\_Name (Zeichenkette mit Begrenzung auf 20 Zeichen) > Beide dürfen nicht frei gelassen werden

1. **Befehl CONSTRAINT <Regelname> PRIMARY KEY:**

CREATE TABLE tblAbteilung

( A\_Nr INTEGER NOT NULL,

A\_Bezeichnung VARCHAR (20) NOT NULL,

A\_Leitung VARCHAR (20) NOT NULL,

CONSTRAINT PK\_Abteilung PRIMARY KEY (A\_Nr));

Ergebnis des Befehls:

Erläuterung des Befehls:

Erstellung Tabelle mit Spalten

Festlegen von A\_Nr als Primärschlüssel unter dem Namen der Regel „Prim\_S\_tblAbteilung“

1. **Befehl ALTER TABLE <Tabellenname> ADD COLUMN:**

ALTER TABLE tblAbteilung ADD COLUMN A\_Raum VARCHAR (20);

Ergebnis des Befehls:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  | **tblAbteilung** |  |  |  |  |
|  | A\_Nr | A\_Bezeichnung | A\_Leitung |  |  |

Erläuterung des Befehls:

1. **Befehl ALTER TABLE <Tabellenname> DROP COLUMN:**

Ergebnis des Befehls:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  | **tblAbteilung** |  |  |  |  |
|  | A\_Nr | A\_Bezeichnung | A\_Raum |  |  |

Erläuterung des Befehls:

Abänderung Tabelle:

Löschen der Spalte „A\_Leitung“

1. **Befehl ALTER TABLE <Tabellenname> CONSTRAINT <Regelname> PRIMARY KEY:**

ALTER TABLE tblMitarbeiter ADD

CONSTRAINT PK\_Mitarbeiter PRIMARY KEY (A\_Nr);

Ergebnis des Befehls:

Erläuterung des Befehls:

Abänderung Tabelle:

Hinzufügen eines Primärschlüssels

1. **Ausgangssituation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  | **tblMitarbeiter** |  |  |  |  |
|  | # M\_Nr | M\_Name |  |  |  |
|  |  |  |  |  |  |
|  | tblAbteilung |  |  |  |  |
|  | # A\_Nr | A\_Bezeichnung | A\_Raum |  |  |

Befehl DROP TABLE:

Ergebnis des Befehls:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dbLEIT-Systems** | |  |  |  |  |
|  | **tblAbteilung** |  |  |  |  |
|  | # A\_Nr | A\_Bezeichnung | A\_Raum |  |  |

Erläuterung des Befehls:

Löschen der Tabelle „tblMitarbeiter“

1. **Befehl DROP DATABASE:**

DROP DATABASE dbLEIT-Systems;

Ergebnis des Befehls:

Erläuterung des Befehls: