# DOKUMENTATION ONLINESHOP

Modul 153 Datenmodelle entwickeln

18.04.2019

Alexander Siegenthaler

# Inhaltsverzeichnis

# 1. Zusammenfassung des Projektauftrags

Ziel der Projektarbeit ist die Realisierung einer Datenbank, für einen Onlineshop, gemäss folgenden Anforderungen:

### Externe Geschäftsfälle

- Angebote im Onlineshop ansehen
- Erfassen eines Warenkorbs, Produkte in den Warenkorb legen
- Produkte im Warenkorb kaufen
- Bestellte Produkte verfolgen
- Benutzerkonto pflegen

### Interne Geschäftsfälle

- Anlegen und Pflegen der Produktekategorien
- Anlegen und Pflegen des Produktestamms
- Anlegen und Pflegen der Angaben zum Shop Betreiber
- Anlegen und Pflegen der Kundeninformationen
- Anlegen und Pflegen der Profildaten der Kunden
- Rechnung erstellen
- Verfügbarkeit der Produkte ermitteln

Im ersten Schritt sollen wir alle benötigten Entitäten identifizieren und beschreiben, um so ein konzeptionelles Datenmodell zu erstellen. Danach sollen wir die Tabellen definieren und das physische Datenmodell erstellen. Anschliessend wird das physische Datenmodell normalisiert, die Indizes festgelegt und das SQL-Script vom physischen Datenmodell erstellt. Zum Schluss sollen wir einen Prototyp der Datenbankanwendung erstellen (Access) und mit möglichen Datensätzen testen. Für die gesamte Projektarbeit wird eine Dokumentation inkl. Zeitplan verlangt.

# 2. Zeitplan

| Datum | Arbeitsschritte |
|---|---|
| 28.03.2019 | - Beschaffung der benötigten Informationen (Anforderungen)<br>- Zeitplan erstellen<br>- Zusammenfassung des Projektauftrags<br>- Entitäten identifizieren<br>- Konzeptionelles Datenmodell erstellen<br>- Dokumentation ergänzen |
| 04.04.2019 | - Evtl. Konzeptionelles Datenmodell ergänzen<br>- Tabellen definieren, Attribute festlegen<br>- Physisches Datenmodell grafisch darstellen<br>- Physisches Datenmodell normalisieren<br>- Dokumentation ergänzen |
| 17.04.2019 | - Indexe festlegen<br>- SQL-Script vom Physischen Datenmodell erstellen<br>- Dokumentation ergänzen |
| 18.04.2019 | - Prototyp (Access) der Datenbankanwendung erstellen<br>- Plausible Datensätze erfassen<br>- Projektarbeit überprüfen<br>- Dokumentation ergänzen<br>- Abgabe Projektarbeit |

# 3. Anforderungen Datenmodell Onlineshop

### Ort

Die Entität "Ort" enthält eine Liste von Ortschaften und wird dem Kunden zugewiesen.

*Ort-Name, PLZ*

### Kunde

Die Entität "Produkte" enthält Informationen über den Kunden und wird benötigt um den Kunden zu Identifizieren und um die Bestellung an die richtige Person zu liefern.

*Name, E-Mail, Alter, Adresse, Login-Name, Passwort*

### Bestellung

Die Entität "Bestellung" verbindet den Warenkorb mit dem Kunden. Sie gibt ausserdem noch Auskunft über die Lieferart (Liefern/Abholen), das Bestell-/Übergabedatum und, falls das Produkt bezahlt wurde, über das Bezahldatum.

*KundeID, Bestelldatum, Lieferart, Übergabedatum, Bezahldatum*

### Warenkorb

Die Entität "Warenkorb" verbindet die gewünschten Produkte, und deren Anzahl, mit der Bestellung.

*ProduktID, BestellungID, Anzahl*

### Produkte

Die Entität "Produkte" enthält Informationen bezüglich der Produkte, die verkauft werden und wird benötigt, um zu wissen, welche Produkte der Kunde will, ob das Produkt noch vorhanden ist, was der Preis ist und zu welcher Kategorie das genannte Produkt gehört.

*Name, Preis, Anzahl, Kategorie*

### Kategorie

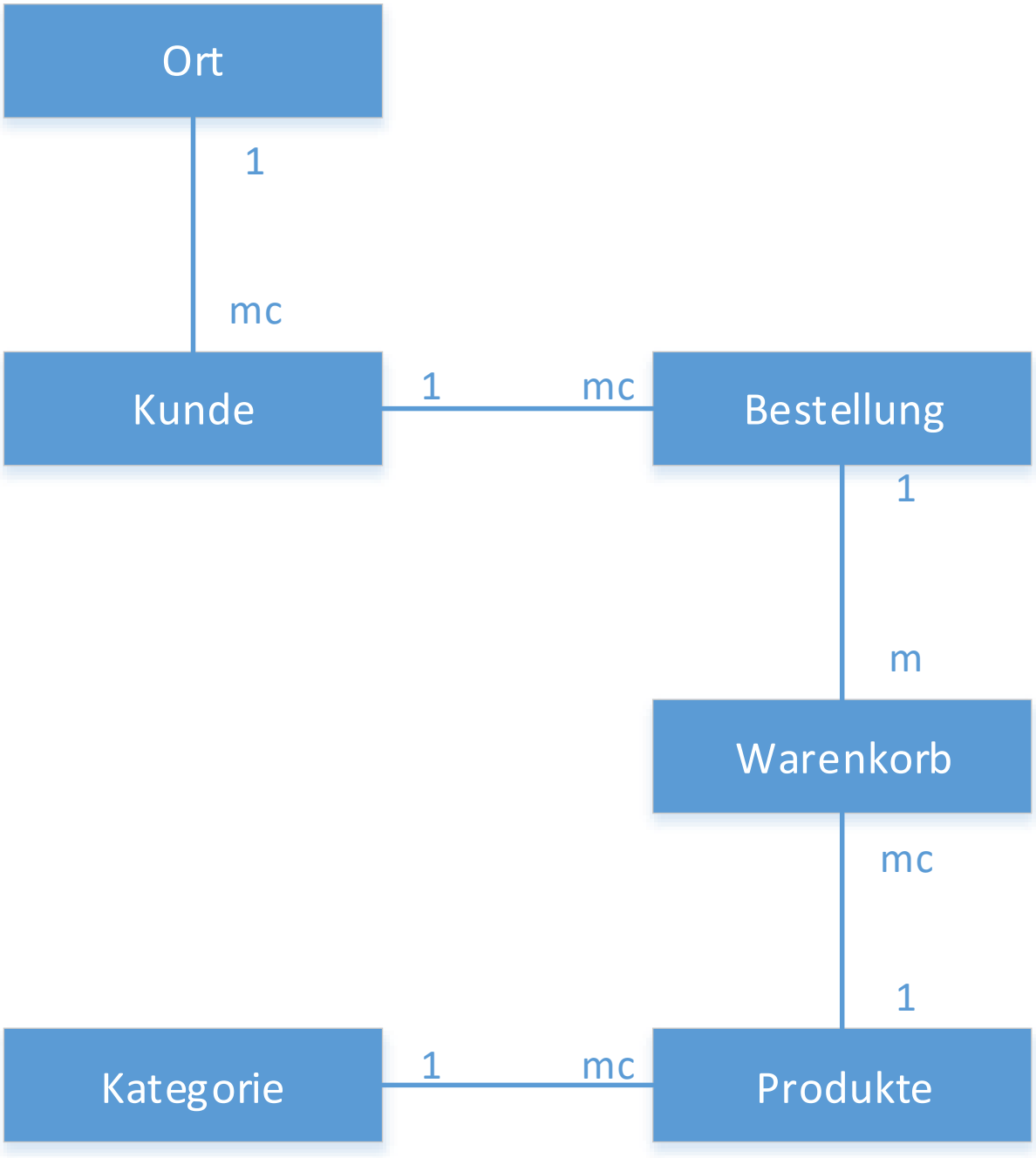
Die Entität "Kategorie" enthält den Kategorienamen (z.B. Monitor, Maus) und eine Beschreibung der Kategorie.

*Name, Beschreibung*

### Sonstige Überlegungen

Die Login-Daten könnte man noch in einer eigenen Entität festhalten, damit nicht alle Datenbank-Benutzer darauf Zugriff haben und um gleiche Login-Namen zu verhindern. Da ich aber von einem einzelnen Datenbank-Benutzer ausgehe, gebe ich einfach dem Attribut für den Login-Namen noch die Eigenschaft Unique mit.

# 4. Konzeptionelles Datenmodell



# 5. Physisches Datenmodell

## 5.1. Tabellen definieren, Attribute festlegen

### Tabelle Ort

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| ort_id | Integer | X | X |
| ort | Varchar(30) | X | |
| plz | Integer | X | |

### Tabelle Kunde

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| kunde_id | Integer | X | X |
| name | Varchar(20) | X | |
| vorname | Varchar(20) | X | |
| email | Varchar(40) | X | |
| geburtstag | Date | X | |
| strasse | Varchar(40) | X | |
| ort_id | Integer | X | |
| login | Varchar(20) | X | |
| passwort | Varchar(20) | X | |

### Tabelle Bestellung

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| bestellung_id | Integer | X | X |
| kunde_id | Integer | X | |
| bestelldatum | Datetime | X | |
| lieferart | Enum(Lieferung,Abholen) | X | |
| übergabedatum | Datetime | | |
| bezahldatum | Datetime | | |

### Tabelle Warenkorb

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| warenkorb_id | Integer | X | X |
| produkt_id | Integer | X | |
| anzahl | Integer | X | |
| bestellung_id | Integer | X | |

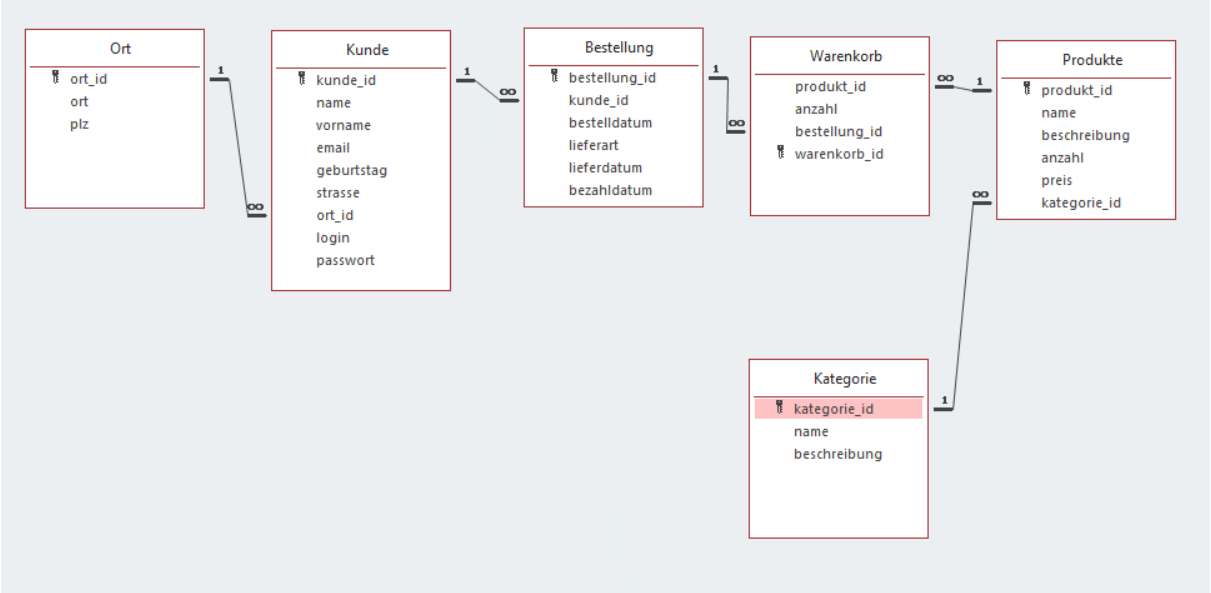
### Tabelle Produkte

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| produkt_id | Integer | X | X |
| name | Varchar(30) | X | |
| beschreibung | Text | X | |
| anzahl | Integer | X | |
| preis | Decimal(8,2) | X | |
| kategorie_id | Integer | X | |

### Tabelle Kategorie

| Attribut | Datentyp | erforderlich | Primary Key |
|---|---|---|---|
| kategorie_id | Integer | X | X |
| name | Varchar(30) | X | |
| beschreibung | Text | X | |

## 5.2. Grafische Darstellung




## 5.3. Normalisierung

### Ort

#### Erste Normalform

Die Tabelle Ort enthält den Ortsnamen, die Postleitzahl und die ID. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

#### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

#### Dritte Normalform

Der Name(ort) ist transitiv von ort_id abhängig. Dies könnte ich umgehen, wenn ich die Postleitzahl(plz) als Primärschlüssel definieren würde. Würden jedoch die Postleitzahlen geändert werden, müsste ich meine Datenbank dementsprechend anpassen. Daher habe ich mich gegen eine Änderung entschieden und deswegen erfüllt diese Tabelle nicht die dritte Normalform.

### Kunde

#### Erste Normalform

Die Tabelle Kunde enthält Informationen über den Kunden und seine Login-Daten. Der Name besteht aus Vornamen und Nachnamen, Adresse besteht aus Strasse und Ortschaft und der Login aus Login-Namen und Passwort. Bei der Adresse enthält die Strasse den Namen und die Nummer, da sich durch die Trennung in unterschiedliche Attribute jedoch keine Vor-/Nachteile ergeben benütze ich dafür ein einzelnes Attribut. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

#### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

#### Dritte Normalform

Alle Attribute sind funktional vom Primärschlüssel abhängig. Die Tabelle befindet sich in der zweiten Normalform. Die dritte Normalform ist somit gegeben.

## Bestellung

### Erste Normalform

Die Tabelle Bestellung enthält die Verbindung zum Kunden und zum Warenkorb und Informationen zum Bestell-/Bezahlvorgang. Das Bestell-/Bezahl- und Lieferdatum sind in unterschiedlichen Attributen festgehalten. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

### Dritte Normalform

Alle Attribute sind funktional vom Primärschlüssel abhängig. Die Tabelle befindet sich in der zweiten Normalform. Die dritte Normalform ist somit gegeben.

## Warenkorb

### Erste Normalform

Die Tabelle Warenkorb enthält die Verbindung zur Bestellung und zu den Produkten und die Anzahl der Produkte. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

### Dritte Normalform

Alle Attribute sind funktional vom Primärschlüssel abhängig. Die Tabelle befindet sich in der zweiten Normalform. Die dritte Normalform ist somit gegeben.

## Produkte

### Erste Normalform

Die Tabelle Produkte enthält Namen, Preis und Anzahl vom Produkt und die Verbindung zu der Kategorie. Der Name besteht aus dem Produktnamen und der Beschreibung. Die Anzahl und der Preis sind in unterschiedlichen Attributen. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

### Dritte Normalform

Alle Attribute sind funktional vom Primärschlüssel abhängig. Die Tabelle befindet sich in der zweiten Normalform. Die dritte Normalform ist somit gegeben.

## Kategorie

### Erste Normalform

Die Tabelle Kategorie enthält den Namen der Kategorie. Der Name besteht aus dem Kategorienamen und der Beschreibung. Alle Attribute weisen einfache Attributwerte auf. Die erste Normalform ist somit gegeben.

### Zweite Normalform

Die Zweite Normalform ist erfüllt, da kein zusammengesetzter Primärschlüssel existiert.

### Dritte Normalform

Alle Attribute sind funktional vom Primärschlüssel abhängig. Die Tabelle befindet sich in der zweiten Normalform. Die dritte Normalform ist somit gegeben.

## 5.4. Prognose über Mengen und Häufigkeiten, Indexe festlegen

**Kundenanforderungen und Zugriffe auf Tabellen**

| Kundenanforderungen | Ort | Kunde | Bestellung | Warenkorb | Produkte | Kategorie |
|---|---|---|---|---|---|---|
| Kunde: Angebote im Onlineshop ansehen | - | - | - | - | X | X |
| Kunde: Erfassen eines Warenkorbs | X | X | X | X | X | X |
| Kunde: Bestellung | X | X | X | X | X | X |
| Kunde: Bestellungsvorgang verfolgen | - | X | X | X | X | X |
| Kunde: Benutzerkonto pflegen | X | X | - | - | - | - |
| Intern: Anlegen/Pflegen Kategorien | - | - | - | - | - | X |
| Intern: Anlegen/Pflegen Produkte | - | - | - | - | X | X |
| Intern: Anlegen/Pflegen Kundeninformationen | X | X | - | - | - | - |
| Intern: Anlegen/Pflegen Profildaten(Login) | - | X | - | - | - | - |
| Rechnung erstellen | - | X | X | X | X | - |
| Verfügbarkeit der Produkte | - | - | - | - | X | - |

**Ort**
Auf den Ortsnamen und die Plz gibt es vergleichsweise wenig Zugriffe, daher keine Indizes.

**Kunde**
Auf den Kunden gibt es relativ viele Zugriffe, daher indexier ich das Attribut name und vorname.

**Bestellung**
Auf die Bestellung gibt es vergleichsweise wenig Zugriffe, daher keine Indizes.

**Warenkorb**
Auf den Warenkorb gibt es vergleichsweise wenig Zugriffe, daher keine Indizes.

**Produkte**
Auf die Produkte gibt es relativ viele Zugriffe, daher indexier ich das Attribut name, anzahl und preis.

**Kategorie**
Auf die Kategorie gibt es relativ viele Zugriffe, daher indexier ich das Attribut name.

**Sonstige Überlegungen**
Da die meisten Datenbanktools, mit denen ich gearbeitet habe, prinzipiell alle Fremdschlüssel indexieren, sind diese in meiner Datenbank ebenfalls indexiert. Ausserdem macht es Sinn diese zu indexieren da auf diese vergleichsweise oft zugegriffen wird.

## 5.5. SQL-Script

```
drop database if exists onlineshop;
create database onlineshop;
use onlineshop;
create table    ort(
ort_id          integer not null auto_increment,
ort             varchar(30) not null,
plz        integer not null,
primary key(ort_id)
);
create table kunde(
kunde_id   integer not null auto_increment,
name  varchar(20) not null,
vorname varchar(20) not null,
email varchar(40) not null,
geburtstag date not null,
strasse    varchar(40) not null,
ort_id     integer not null,
login varchar(20) not null unique,
passwort   varchar(20) not null,
primary key(kunde_id),
foreign key(ort_id) references ort(ort_id) on update cascade on
delete cascade
);
create table kategorie(
kategorie_id    integer not null auto_increment,
name  varchar(30) not null,
beschreibung    text not null,
primary key(kategorie_id)
);
create table produkte(
produkt_id integer not null auto_increment,
name  varchar(30) not null,
beschreibung    text not null,
anzahl     integer not null,
preis decimal(8,2) not null,
kategorie_id    integer not null,
primary key(produkt_id),
foreign key(kategorie_id) references kategorie(kategorie_id) on
update cascade on delete cascade
);
create table bestellung(
bestellung_id   integer not null auto_increment,
kunde_id   integer not null,
bestelldatum    datetime not null default current_timestamp,
lieferart  enum('lieferung','abholen') not null default 'abholen',
uebergabedatum  datetime,
bezahldatum     datetime,
primary key(bestellung_id),
foreign key(kunde_id) references kunde(kunde_id) on update cascade
on delete cascade
);
create table warenkorb(
warenkorb_id    integer not null auto_increment,
produkt_id integer not null,
bestellung_id   integer not null,
```

```
anzahl integer not null,
primary key(warenkorb_id),
foreign key(produkt_id) references produkte(produkt_id) on update
cascade on delete cascade,
foreign key(bestellung_id) references bestellung(bestellung_id) on
update cascade on delete cascade
);
#Indexe definieren
CREATE INDEX ort_id_index
ON kunde (ort_id);
CREATE INDEX kategorie_id_index
ON produkte (kategorie_id);
CREATE INDEX kunde_id_index
ON bestellung (kunde_id);
CREATE INDEX produkt_id_index
ON warenkorb (produkt_id);
CREATE INDEX bestellung_id_index
ON warenkorb (bestellung_id);
CREATE INDEX name_kunde_index
ON kunde (name);
CREATE INDEX vorname_kunde_index
ON kunde (vorname);
CREATE INDEX name_produkte_index
ON produkte (name);
CREATE INDEX anzahl_produkte_index
ON produkte (anzahl);
CREATE INDEX preis_produkte_index
ON produkte (preis);
CREATE INDEX name_kategorie_index
ON kategorie (name);
```

# 6. Anforderungen überprüfen, Prototyp

Für die Überprüfung ist eine Access Anwendung mit entsprechenden Formularen erstellt worden. Diverse Datenfeldtypen wurden geändert, da das Sql-Script für MariaDB erstellt wurde und diese teilweise in Access nicht vorhanden sind. Gewisse Attributnamen wurden ebenfalls leicht abgeändert, da Access reservierte Namen hat und es sonst zu Fehlern kommen kann z.B. name -> namen. Anschliessend wurde für jede Tabelle plausible Datensätze erfasst.