Für einen Hersteller können kein, ein oder mehrere Artikel gespeichert werden. Die Beziehung ist optional mehrdeutig (`0..*`), ein Artikel muss über einen Hersteller verfügen und ist damit obligatorisch eindeutig. Die Notation im Klassenmodell der UML sieht wie folgt aus:

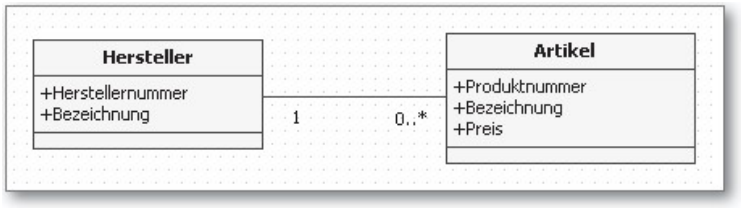

Abbildung 17.1 UML-Darstellung für Übung 2.2

Übung 2.3

In der Beispieldatenbank (siehe Abbildung 1.1) existiert die Tabelle `Abteilung`. Die Ausstattung an Kopiergeräten der einzelnen Abteilungen soll nun in dieser Tabelle gespeichert werden. Dazu wird sie um das Feld Kopierer erweitert. Ein Datenbankauszug sieht dann wie folgt aus:

Abteilungsnr.

Bezeichnung

Kopierer

4

Einkauf

Canon ES1000, Toshiba TH555

5

Vertrieb

Canon ES1000

Der Tabellenaufbau entspricht nicht der 1. und 2. Normalform. Die Speicherung von mehreren Werten, in diesem Fall der Kopierer, in einer Tabellenspalte verletzt die Bedingungen der 1. Normalform. Da die Kopierer zudem nicht vom Primärschlüssel abhängig sind, ist auch die 2. Normalform nicht erfüllt. Dazu müsste man die Kopierer in einer eigenen Tabelle (z. B. Ausstattung) speichern. Durch die 1:n-Beziehung wird in der Tabelle die Abteilungsnummer als Fremdschlüssel definiert.

17.2 Lösungen zu Kapitel 3

Übung 3.1

Definieren Sie eine Tabelle `geburtstag` mit Name, Vorname und Geburtstag. Die Datensätze sollen eine laufende Nummer (`gebnr`) erhalten.

Der Primärschlüssel kann in der Spaltendefinition oder direkt nach den Spaltendefinitionen anlegt werden. Als Spaltengröße für `name` und `vorname` wurde hier Text mit der Länge von 50 Zeichen gewählt. Das Geburtsdatum ist ein Datumswert (`DATE`):

```
CREATE TABLE geburtstag
(
   gebnr INTEGER NOT NULL,
   name VARCHAR(50),
   vorname VARCHAR(50),
   geburtstag DATE,
   PRIMARY KEY (gebnr)
);
```

Die Definition des Primärschlüssels kann auch bei der Spaltendefinition erfolgen. Also ist auch diese Lösung gültig:

```
CREATE TABLE geburtstag
(
   gebnr INTEGER NOT NULL PRIMARY KEY,
   name VARCHAR(50),
   vorname VARCHAR(50)
   geburtag DATE
);
```

Übung 3.2

Definieren Sie eine Tabelle `urlaub`, in der Sie Name, Vorname, Urlaubsantritt (`beginn`) und Urlaubsende (`ende`) speichern. Auch hier sollen die Datensätze eine laufende Nummer (`urlnr`) als Primärschlüssel erhalten.

Da die Felder *Urlaubsantritt* (`beginn`) und *Urlaubsende* (`ende`) Datumswerte enthalten, ist hier der Datentyp `DATE` sinnvoll:

```
CREATE TABLE urlaub
(
   urlnr INTEGER NOT NULL,
   name VARCHAR(50),
   vorname VARCHAR(50),
```

```
   beginn DATE,
   ende DATE,
   PRIMARY KEY (urlnr)
);
```

Übung 3.3

Definieren Sie eine Tabelle `telefonliste`, in der Sie `name`, `vorname`, `telefonnummer` und einen Buchstaben als `bemerkung` wie H für Handy oder G als geschäftliche Verbindung speichern wollen. Der Name und die Telefonnummer sollen auf jeden Fall eingegeben werden. Die Datensätze sollen eine laufende Nummer (`telnr`) als Primärschlüssel erhalten. Der Primärschlüssel soll als `CONSTRAINT` mit dem Namen `primaerschluessel` angelegt werden.

Die Spalten `name` und `telefonnummer` sollen mit `NOT NULL` definiert werden. Die Telefonnummer wird als Zeichenkette definiert. Die Bemerkung hat mit einem Zeichen immer die gleiche Länge und kann daher als `CHAR` angelegt werden. Der besseren Übersichtlichkeit wegen ist der Primärschlüssel am Ende angelegt:

```
CREATE TABLE telefonliste
(
   telnr INTEGER NOT NULL,
   name VARCHAR(50) NOT NULL,
   vorname VARCHAR(50),
   telefonnummer VARCHAR(50) NOT NULL,
   bemerkung CHAR(1),
   CONSTRAINT primaerschluessel PRIMARY KEY (telnr)
);
```

Übung 3.4

Sehen Sie sich die folgenden Datensätze A bis J mit `abteilungsnr` und `bezeichnung` an. Wenn Sie sie in die Tabelle `abteilung` eingeben, werden sie dann angenommen oder nicht? Warum?

Die Datensätze A, B, D, E, F, I und J werden angenommen, weil die `abteilungsnr` noch nicht existiert. Der Name der Abteilung ist nicht von Bedeutung, da diese Spalte nicht mit `NOT NULL` definiert wurde. Der Datensatz C wird abgelehnt, weil die `abteilungsnr`, die als Primärschlüssel angegeben werden muss, fehlt. Bei G und H sind die Abteilungsnummern schon vorhanden und werden deshalb als Primärschlüssel abgelehnt.

Übung 3.5

Sehen Sie sich die folgenden Datensätze A bis E an. Sie sollen in die Tabelle `mwstsatz` eingefügt werden und bestehen aus der `mwstnr` und `prozent`. Wenn Sie diese Datensätze eingeben, werden sie dann angenommen oder nicht? Warum?

A
1
7
B
2
16
C
3
120
D
29,6
E
2
7

Die Datensätze A und B werden angenommen, weil die `mwstnr` noch nicht vergeben wurde. C wird abgelehnt, weil die Prozentzahl drei Stellen vor dem Komma hat, aber nur zwei erlaubt sind. Bei D fehlt die `mwstnr`. Bei E ist die `mwstnr` schon als Primärschlüssel vorhanden und wird deshalb abgelehnt.

Übung 3.6

Definieren Sie für die Beispielfirma eine Tabelle `ferien`, in der der Urlaubsbeginn (`beginn`) und das Urlaubsende (`ende`) für die einzelnen Mitarbeiter gespeichert werden sollen. Anstatt der Namen der Mitarbeiter sollen Sie dort die `mitarbeiternr` aus der Tabelle `mitarbeiter` speichern. Die Tabelle muss also einen Fremdschlüssel haben. Auf einen Primärschlüssel können Sie verzichten.

Die Tabellendefinition lautet:

```
CREATE TABLE ferien
(
   mitarbeiternr INTEGER,
   beginn DATE,
```

```
   ende DATE,
   FOREIGN KEY (mitarbeiternr)
      REFERENCES mitarbeiter (mitarbeiternr));
```

Übung 3.7

Definieren Sie für die Beispielfirma eine Tabelle `notfall`, in der für jeden Mitarbeiter Telefonnummern zur Benachrichtigung von Angehörigen gespeichert werden sollen. Auch hier beziehen Sie sich auf die Tabelle `mitarbeiter`. Sie benötigen keine Primärschlüsselspalte.

Die Beziehung erfolgt über die Spalte `mitarbeiternr`. Die Tabelle `notfall` ist also eine abhängige Tabelle, die Spalte `mitarbeiternr` hier ein Fremdschlüssel:

```
CREATE TABLE notfall
(
   mitarbeiternr INTEGER,
   telefonnummer VARCHAR(50),
   FOREIGN KEY (mitarbeiternr)
      REFERENCES mitarbeiter (mitarbeiternr)
);
```

Übung 3.8

Die Tabelle `artikel` ist selbst auch die Vatertabelle für die Tabelle `posten`. Diese Tabelle ist auch von der Tabelle `bestellung` abhängig, deren Primärschlüsselspalte die Spalte `bestellnr` ist. Außerdem werden `bestellmenge` und `liefermenge` in `posten` gespeichert. Wie sieht der `CREATE TABLE`-Befehl für die Tabelle `posten` aus?

Die Tabelle `posten` hat zwei Fremdschlüssel. Sie wurde in der Beispieldatenbank so angelegt:

```
CREATE TABLE posten
(
   bestellnr INTEGER NOT NULL,
   artikelnr INTEGER,
   bestellmenge INTEGER,
   liefermenge INTEGER,
   FOREIGN KEY (bestellnr)
      REFERENCES bestellung,
   FOREIGN KEY (artikelnr)
      REFERENCES artikel
);
```

Wenn Sie dieses Beispiel nachvollziehen wollen, müssen Sie die Tabelle `posten` zuerst löschen.

Übung 3.9

In der Tabelle `abteilung` gibt es die Datensätze 1 bis 6, sie ist die Vatertabelle für die Tabelle `mitarbeiter`. In die Tabelle `mitarbeiter` sollen die Datensätze A bis J eingegeben werden. Werden sie angenommen oder nicht? Warum?

Die Datensätze A, B, C, D, E, F, G und J werden angenommen, da nicht unbedingt alle Angaben gemacht werden müssen und die Abteilungsnummern in der Tabelle `abteilung` existieren. Bei den Datensätzen H und I ist das nicht der Fall.

Übung 3.10

In der Tabelle `mwstsatz` gibt es zwei Datensätze 1 und 2. In der Tabelle `hersteller` gibt es die Datensätze 1 bis 10. In der Tabelle `kategorie` gibt es ebenfalls die Datensätze 1 bis 10. Alle diese Tabellen sind, wie Sie wissen, Vatertabellen für die Tabelle `artikel`. Sehen Sie sich nun die Datensätze A bis J an, die in die Tabelle `artikel` eingegeben werden sollen. Werden sie angenommen oder nicht? Warum? Denken Sie auch an den Primärschlüssel der Tabelle `artikel`.

Die Datensätze A, B, C, H und I werden angenommen, da die angegebenen `artikelnr` noch nicht vorhanden sind und die anderen angegebenen Nummern in der jeweiligen Tabelle existieren. Bei D fehlt die `artikelnr`. Bei E ist die `mwst` nicht in der Tabelle `mwstsatz` vorhanden. Bei F ist die `kategorie` in der Tabelle `kategorie` nicht vorhanden. Bei G ist `hersteller` in der Tabelle `hersteller` nicht vorhanden. Bei J ist die `artikelnr` bereits vorhanden.

Übung 3.11

Finden Sie drei Anwendungsfälle, bei denen die Verwendung von `UNIQUE` sinnvoll ist.

Im Folgenden sind einige Beispiele für die sinnvolle Verwendung von `UNIQUE` aufgelistet:

- Eingabe von Kontonummern
- die Postleitzahl in einer Postleitzahlentabelle

- Matchcodes (Suchwort) in der Kundentabelle
- Bezeichnung von Firmennamen in einer Firmentabelle, um Verwechslungen zu vermeiden

Übung 3.12

Definieren Sie eine Tabelle `rabatt`, in der zu der Kundennummer eine Rabattstufe gespeichert werden soll. Es gibt die Rabattstufen »Bronze (`B`)«, »Silber (`S`)« und »Gold (`G`)«. Die eingegebene Rabattstufe soll immer gültig sein. Die Tabelle braucht keinen Primärschlüssel.

Die Rabattstufen werden über eine `CHECK`-Bedingung wie folgt definiert:

```
CREATE TABLE rabatt
(
   kunde INTEGER NOT NULL,
   rabatt char(1) CHECK (rabatt IN ('B', 'G', 'S'))
);
```

Sie können überprüfen, ob der `CHECK` korrekt ausgeführt wird, indem Sie versuchen, einen Datensatz zu speichern, der gegen die `CHECK`-Bestimmung verstößt, z. B.:

```
INSERT INTO rabatt (kunde, rabatt) VALUES (1, 'Z');
```

Übung 3.13

In der Tabelle `kunde` gibt es eine Spalte `zahlungsart`. Wie sieht die Spalte aus, wenn Sie eine `CHECK`-Klausel hinzufügen? Gültige Eingaben sollen `R`, `B`, `N`, `V` und `K` sein.

Die Spalte `zahlungsart` könnte mit einer `CHECK`-Klausel so definiert werden:

```
zahlungsart CHAR(1)
   CHECK (zahlungsart IN ('R', 'B', 'N', 'V', 'K'));
```

Übung 3.14

Nehmen wir an, die Tabelle `posten` wäre tatsächlich wie in Übung 3.13 mit der `CHECK`-Klausel definiert worden. Sehen Sie sich die Datensätze A bis J an. Werden sie in die veränderte Tabelle `posten` aufgenommen oder nicht? Warum? Denken Sie auch an die Fremdschlüssel der Tabelle! Gegenwärtig gibt es Bestellungen von 1 bis 150 und Artikel von 1 bis 50.

Die Datensätze A, B, C, E und H werden angenommen. Bei D ist eine falsche Bestellmenge eingetragen. Bei F existiert die Artikelnummer nicht. Bei G fehlt die Artikelnummer. Bei I fehlt die Bestellnummer. Bei J existiert die Bestellnummer nicht.

Übung 3.15

Werden die folgenden Urlaubsanträge A bis J in die oben definierte Tabelle `urlaub` aufgenommen oder nicht? Warum?

Die Datensätze A, B, C, E, G, H und I werden angenommen. Die Datensätze D, F und J werden abgelehnt, weil die Urlaubszeit länger als die vorgegebene maximale Dauer der Abwesenheit wäre.

Übung 3.16

Werden die Datensätze A bis J in die Tabelle `rabatt`, die Sie in Übung 3.12 definiert haben, aufgenommen oder nicht? Warum? Achten Sie auf den Fremdschlüssel – es gibt zurzeit die Kundennummern 1 bis 100.

Die Datensätze A, B und C werden angenommen. Datensatz D wird abgelehnt, weil hier ein Kleinbuchstabe eingegeben werden soll. Datensatz E wird abgelehnt, weil hier ein Leerzeichen eingegeben werden soll. Bei Datensatz G soll eine Rabattstufe eingegeben werden, die nicht vorhanden ist. Bei Datensatz H ist keine Rabattstufe angegeben. Bei Datensatz I wird eine nicht existierende Kundennummer angegeben, er wird also auch abgelehnt. In Datensatz J wird versucht, eine Zahl anstatt eines Buchstabens einzugeben.

Übung 3.17

Nehmen wir an, die Tabelle kunde hätte die eben verlangte `CHECK`-Klausel für die Spalte `zahlungsart` (vergleichen Sie dazu mit Übung 3.13). Welche der folgenden Datensätze A bis J werden dann angenommen oder abgelehnt? Warum? Denken Sie an den Primärschlüssel (vergleichen Sie dazu mit Übung 3.16).

Die Datensätze A, B, C und I werden angenommen. Bei D und E existiert die Zahlungsart nicht. Bei F wird ein Kleinbuchstabe eingegeben. Bei G wird ein falscher Buchstabe eingegeben. Bei J wird eine schon bestehende Kundennummer angegeben.

Übung 3.18

In Übung 3.15 wurde ein unsinniger Datensatz angenommen: Ein Urlaub sollte enden, bevor er überhaupt begonnen hat. Wie können Sie weitere Fehler dieser Art mit einer `CHECK`-Klausel verhindern?

Das Urlaubsende darf nicht vor dem Urlaubsantritt liegen. Sie können solche Fehleingaben vermeiden, wenn Sie eine `CHECK`-Bedingung definieren, die verhindert, dass das Urlaubsende vor dem Urlaubsanfang liegt. Die Tabelle kann wie folgt definiert werden:

```
CREATE TABLE urlaub
(
   urlnr INTEGER NOT NULL,
   name VARCHAR(50),
   vorname VARCHAR(50),
   beginn DATE,
   ende DATE CHECK (ende > beginn),
   PRIMARY KEY (urlnr)
);
```

Übung 3.19

Erstellen Sie eine Domäne mit dem Namen `d_plz` für den Gültigkeitsbereich von Postleitzahlen. Der Gültigkeitsbereich soll nur deutsche Postleitzahlen umfassen, also eine fünfstellige Zahl sein. Beachten Sie, dass dabei auch eine führende Null gespeichert werden muss. Eine Definition eines Zahlendatentyps scheidet deshalb aus, weil bei Zahlen keine führende Null gespeichert wird.

Die Domänendefinition lautet:

```
CREATE DOMAIN d_plz
  AS CHAR(5);
```

Da für die Postleitzahl die führende Null unbedingt erforderlich ist, muss ein String-Datentyp verwendet werden. Bei einem Zahlenwert werden führende Nullen nicht gespeichert.

Übung 3.20

Definieren Sie eine Domäne `d_groesser_null`, die die Eingabe von negativen Werten verhindert. Diese Domänendefinition kann z. B. für die Spalte `mindestbestand` der Tabelle `artikel` angewendet werden.

Die Domänendefinition lautet:

```
CREATE DOMAIN d_groesser_null
   AS integer
   CHECK (VALUE >= 0);
```

Übung 3.21

Setzen Sie die Mehrwertsteuer als Domäne `d_mehrwertsteuer` um. Der Datentyp soll eine Nachkommastelle haben, der Vorgabewert soll 19 (Prozent) sein.

Die Domänendefinition lautet:

```
CREATE DOMAIN d_mehrwertsteuer
   AS DECIMAL(3,1)
  DEFAULT '19';
```

Übung 3.22

Erstellen Sie eine Tabelle mit dem Namen `anschrift` und den Spalten `name`, `strasse`, `postleitzahl`, `ort`. Weisen Sie der Spalte `postleitzahl` die erstellte Domäne `d_plz` zu.

Die Domänendefinition wird dem Spaltennamen zugeordnet. Die Tabellendefinition lautet wie folgt:

```
CREATE TABLE anschrift
(
   name VARCHAR(50),
   strasse VARCHAR(50),
   plz d_plz,
   ort VARCHAR(50)
);
```

Übung 3.23

Fügen Sie der Domäne `d_zahlungsart` (aus Abschnitt 3.5.1, »Domänen erstellen [CREATE DOMAIN]«) eine weitere Kategorie hinzu. Als zusätzliche Zahlungsart soll `L` (Lastschrift) akzeptiert werden.

Die Änderung ist in zwei Schritten durchzuführen. Im ersten Schritt wird die bestehende `CONSTRAINT`-Definition gelöscht:

```
ALTER DOMAIN d_zahlungsart DROP CONSTRAINT;
```

Danach kann die neue Einschränkung definiert werden:

```
ALTER DOMAIN d_zahlungsart
   ADD CHECK
   (
      VALUE IN ('R', 'B', 'N', 'V', 'K', 'L')
   );
```

Übung 3.24

In Übung 3.19 wurde die Domäne `d_plz` mit dem Datentyp `CHAR(5)` definiert. Für internationale Postleitzahlen reichen diese fünf Stellen nicht aus. Das Postleitzahlenfeld soll deshalb 14 Zeichen speichern können. Wie gehen Sie vor?

Eine Änderung des Datentyps einer Domäne ist nicht möglich. Sie müssen die Domäne löschen und neu anlegen.

Übung 3.25

Geben Sie Ihrer Domäne `d_zahlungsart` den Vorgabewert `L` (Lastschrift).

Hier setzen Sie einfach den neuen Vorgabewert:

```
ALTER DOMAIN d_zahlungsart
SET DEFAULT 'L';
```

Übung 3.26

Schaffen Sie den eben definierten Vorgabewert für die Domäne `d_zahlungsart` wieder ab.

Der Vorgabewert wird mit `DROP` gelöscht:

```
ALTER DOMAIN d_zahlungsart
DROP DEFAULT;
```

Übung 3.27

Ändern Sie die Domäne `d_email`. Der aufzunehmende Wert soll auch einen Punkt enthalten. Nehmen Sie die Endung `ch` in die Liste auf.

Auch hier ist die `CONSTRAINT`-Definition im ersten Schritt zu löschen:

```
ALTER DOMAIN d_email
DROP CONSTRAINT;
```

Anschließend wird die geänderte Definition neu definiert:

```
ALTER DOMAIN d_email
ADD CHECK
(
(VALUE CONTAINING '@' AND VALUE CONTAINING '.')

AND
(VALUE LIKE '%.de' OR VALUE LIKE '%.com'
   OR VALUE LIKE '%.at' OR VALUE LIKE '%.ch')
AND
(VALUE NOT IN ('gmx.de', 'web.de')));
```

Übung 3.28

Löschen Sie die oben angelegten Domänen `d_gehalt` und `d_email`.

Für jede `Domain`-Definition ist der Befehl `DROP DOMAIN` auszuführen:

```
DROP DOMAIN d_gehalt;

DROP DOMAIN d_email;
```

Übung 3.29

Ändern Sie die Definition der Tabelle `hersteller` aus der Beispieldatenbank. Ergänzen Sie die Felder `plz` und `ort`. Verwenden Sie sinnvolle Datentypen.

Die Änderungen werden mit dem `ALTER TABLE`-Befehl vorgenommen. Sie können beide Änderungen nacheinander durchführen:

```
ALTER TABLE hersteller
ADD plz CHAR(5);

ALTER TABLE hersteller
ADD ort VARCHAR(50);
```

Beide Änderungen können auch in einem Befehl ausgeführt werden:

```
ALTER TABLE hersteller
ADD plz CHAR(5),
ADD ort VARCHAR(50);
```

Übung 3.30

Ändern Sie die Definition der Spalte `name` aus der Tabelle `kunde` der Beispieldatenbank. Vergrößern Sie die Feldgröße von `VARCHAR(50)` auf `VARCHAR(60)`.

Die Änderung erfolgt über einen `ALTER TABLE`-Befehl:

```
ALTER TABLE kunde
ALTER name TYPE VARCHAR(60);
```

Der `ALTER`-Befehl liegt bei verschiedenen Datenbanken in unterschiedlicher Syntax vor. So könnte auch dieser Befehl gültig sein:

```
ALTER TABLE kunde
MODIFY name VARCHAR(60);
```

Übung 3.31

Legen Sie eine Tabelle mit dem Namen `telefonliste` und den Spalten `name`, `telefonnummer` an. Wählen Sie sinnvolle Datentypen für die Spalten. Löschen Sie anschließend die Tabelle `telefonliste`.

Die Lösung lautet:

```
CREATE TABLE telefon
(
name VARCHAR(50),
telefonnummer VARCHAR(20)
);

DROP TABLE telefon;
```

Übung 3.32

Definieren Sie einen Index für das Feld `ort` der Tabelle `kunde`. Geben Sie dem Index den Namen `idx_ort`.

Der Index wird mit einem `CREATE INDEX`-Befehl definiert:

```
CREATE INDEX idx_ort
   ON kunde (ort);
```

Übung 3.33

Definieren Sie einen Multi-Column-Index mit dem Namen `idx_name_vorname` für die Tabelle `kunde`. Dieser Index soll die Felder `name` und `vorname` beinhalten.

Bei Multi-Column-Indizes werden alle Spalten durch Kommata getrennt aufgelistet:

```
CREATE INDEX idx_name_vorname
ON kunde (name, vorname);
```

Übung 3.34

Löschen Sie den in Übung 3.33 angelegten Index `idx_ort`.

Indizes werden mit dem Befehl `DROP INDEX` gelöscht:

```
DROP INDEX idx_ort;
```

17.3 Lösungen zu Kapitel 4

Übung 4.1

Fügen Sie in die Tabelle `mitarbeiter` einen neuen Datensatz ein, der folgende Werte enthält:

Vorname: Hans
Nachname: Ulm
PLZ: 53113
Ort: Bonn
Straße: Talweg 7
Eintrittsdatum: 01.01.2011
Mitarbeiternummer: 304

Der `INSERT`-Befehl lautet wie folgt:

```
INSERT INTO mitarbeiter
   (name, vorname, strasse, plz, ort, eintrittsdatum,
   mitarbeiternr)
VALUES
   ('Ulm', 'Hans', 'Talweg 7', '53113', 'Bonn',
   '01.01.2011', 304);
```

Übung 4.2

Fügen Sie einen neuen Hersteller mit dem Namen »betamax« in die Herstellertabelle ein.

Der `INSERT`-Befehl lautet wie folgt:

```
INSERT INTO hersteller (herstellernr, name)
   VALUES (11, 'betamax');
```

Beachten Sie, dass der Wert in der Spalte `herstellernr` nicht schon vergeben sein darf, weil es sich um einen Primärschlüssel handelt.

Übung 4.3

Fügen Sie einen neuen Artikel in die Tabelle `artikel` ein. Beachten Sie dabei den Fremdschlüssel `mwst`, `hersteller` und `kategorie`. Wählen Sie frei: Artikelname, Hersteller, Nettopreis, Kategorie und Mehrwertsteuersatz. Hersteller, Kategorie und Mehrwertsteuersatz sind dabei Fremdschlüssel.

Wenn Sie einen `INSERT`-Befehl formulieren, müssen Sie die entsprechenden Abhängigkeiten zu den Fremdschlüsseln beachten. In der folgenden Lösung muss deshalb der Primärschlüssel für den Hersteller (»HP«) bzw. für die Kategorie (»Drucker«) ermittelt werden. Der `INSERT`-Befehl lautet dann wie folgt:

```
INSERT INTO artikel
   (artikelnr, bezeichnung, hersteller, nettopreis,
   kategorie, mwst)
   VALUES (75, 'DeskJet 9300', 4, 350, 7, 2);
```

17.4 Lösungen zu Kapitel 5

Übung 5.1

Fragen Sie aus der Tabelle `kunde` die Kundennummer (`kundennr`) und die Zahlungsart (`zahlungsart`) ab.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT kundennr, zahlungsart
   FROM kunde;
```

Übung 5.2

Listen Sie aus der Tabelle `artikel` die Bezeichnungen und den Preis aus.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT bezeichnung, nettopreis
   FROM artikel;
```

Übung 5.3

Suchen Sie aus der Tabelle `mitarbeiter` die Namen und die jeweilige Abteilungsnummer heraus.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT name, vorname, abteilung
   FROM mitarbeiter;
```

Übung 5.4

Lassen Sie sich die Namen der Hersteller aus der gleichnamigen Tabelle ausgeben.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT name
   FROM hersteller;
```

Übung 5.5

Listen Sie alle Artikel der Tabelle `artikel` auf, deren Nettopreis höher als 100 Euro liegt.

Über `SELECT *` werden alle Spalten ausgegeben:

```
SELECT *
   FROM artikel
   WHERE nettopreis > 100;
```

Übung 5.6

Listen Sie alle Mitarbeiter auf, die in der Abteilung 2 beschäftigt sind.

Der SELECT-Befehl lautet wie folgt:

```
SELECT *
   FROM mitarbeiter
   WHERE abteilung = 2;
```

Übung 5.7

Listen Sie alle Artikel auf, die zur Kategorie »Grafikkarten« (Kategorienummer 3) gehören.

Der SELECT-Befehl lautet:

```
SELECT *
   FROM artikel
   WHERE kategorie = 3;
```

Übung 5.8

Verbinden Sie beide Werbeaktionen der Beispielfirma. Beachten Sie, dass Sie die jeweilige Auswahl in Klammern setzen müssen.

Die beiden Auswahlbedingungen werden in diesem Fall mit OR verknüpft:

```
SELECT name, vorname, strasse, plz, ort
   FROM kunde
   WHERE (ort = 'Hamburg' OR ort = 'Bonn')
   OR (name = 'Kaufmann' AND vorname = 'Andreas');
```

Übung 5.9

Geben Sie alle Kunden aus, deren Kundennummer größer als 50 ist und die nicht in Köln wohnen.

Die beiden Auswahlbedingungen werden in diesem Fall mit AND verknüpft:

```
SELECT name, vorname, strasse, plz, ort, kundennr
   FROM kunde
   WHERE kundennr > 50 AND NOT ort = 'Köln';
```

Übung 5.10

Listen Sie alle Artikel in der Reihenfolge der Kategorie und dann alphabetisch auf.

Die Ausgabe soll sortiert erfolgen. Aus diesem Grund ist ein `ORDER BY` anzuwenden:

```
SELECT *
   FROM artikel
   ORDER BY kategorie, bezeichnung;
```

Übung 5.11

Listen Sie alle Mitarbeiter nach ihrem Gehalt und dann nach der Abteilung auf. Das Gehalt soll absteigend sortiert werden.

Mit dem Schlüsselwort `DESC` wird die Sortierreihenfolge umgekehrt:

```
SELECT *
   FROM mitarbeiter
   ORDER BY gehalt DESC, abteilung;
```

Übung 5.12

Listen Sie alle Artikel der Kategorie `4` (Festplatten) absteigend nach dem Preis auf.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT bezeichnung, nettopreis, kategorie
FROM artikel
WHERE kategorie = 4
ORDER BY nettopreis DESC;
```

Übung 5.13

Listen Sie alle Kunden, die per Nachnahme (`N`) bezahlen, nach Postleitzahlenbezirken auf.

Der `SELECT`-Befehl lautet wie folgt:

```
SELECT name, vorname, plz, zahlungsart
FROM kunde
```

```
WHERE zahlungsart = 'N'
ORDER BY plz;
```

Übung 5.14

Sorgen Sie bei der letzten Abfrage der Beispielfirma für eine sortierte Ausgabe der Städte nach der Anzahl der dort lebenden Kunden. Die Stadt mit den meisten Kunden soll dabei zuerst ausgegeben werden.

In diesem Fall müssen Sie noch ein `ORDER BY` ergänzen. Da Sie nach Anzahl der Kunden sortieren wollen, müssen Sie ein `ORDER BY COUNT(*)` verwenden. Um absteigend zu sortieren, ist ein `DESC` zu verwenden:

```
SELECT ort, COUNT(*)
   FROM kunde
   GROUP BY ort
   HAVING COUNT(*) >= 10
   ORDER BY COUNT(*) DESC;
```

Übung 5.15

Lassen Sie die Städte nach der Anzahl der dort lebenden Kunden ausgeben (wie in Übung 5.14). Bei gleicher Anzahl der Kunden soll die Ausgabe der Städte alphabetisch erfolgen.

Die `ORDER BY`-Klausel ist im Vergleich mit Übung 5.14 um den Ort zu ergänzen. Bei gleicher Anzahl an Kunden in einer Stadt wird dann nach dem Ortsnamen in aufsteigender alphabetischer Reihenfolge sortiert:

```
SELECT ort, COUNT(*)
   FROM kunde
   GROUP BY ort
   HAVING COUNT(*) >= 10
   ORDER BY COUNT(*) DESC, ort;
```

Übung 5.16

Lassen Sie sich die Anzahl der Artikel pro Kategorie ausgeben, die teurer als 50 Euro sind.

Um diese Aufgabe zu lösen, muss nach Kategorien gruppiert werden. Dies erfolgt über `GROUP BY kategorie`. Die Selektion aller Artikel, die teurer als 50 Euro sind, erfolgt über `WHERE nettopreis > 50`. Die Aggregat-

funktion `COUNT(*)` schließlich sorgt für die Zählung der jeweiligen Datensätze in den Kategorien:

```
SELECT kategorie, COUNT(*)
   FROM artikel
   WHERE nettopreis > 50
   GROUP BY kategorie;
```

Übung 5.17

Lassen Sie sich die Bestellnummern von allen Bestellungen aus der Tabelle posten ausgeben, bei denen fünf Artikel oder mehr bestellt wurden. Sorgen Sie bitte auch für die Ausgabe der Größe nach.

Über die `COUNT`-Funktion und ein `GROUP BY bestellnr` kann die Anzahl der Positionen für eine Bestellung ermittelt werden. Über eine anschließende Selektionsbedingung kann dann die Anzahl »größer« oder »gleich 5« ermittelt werden. Die Selektionsbedingung muss über `HAVING` definiert werden, weil sie auf die erst während der Abfrage erzeugte Anzahl Bezug nimmt:

```
SELECT bestellnr, COUNT(*)
   FROM posten
   GROUP BY bestellnr
   HAVING COUNT(*) >= 5
   ORDER BY COUNT(*) DESC;
```

Übung 5.18

Bilden Sie eine Vereinigungsmenge aus der Tabelle `kunden_meier` mit der Tabelle `kunden_schmidt`. Aus der Tabelle `kunden_meier` sollen nur alle weiblichen Kunden (`anrede='Frau'`), aus der Tabelle `kunden_schmidt` nur alle Kunden aus Berlin selektiert werden.

Die beiden Tabellen werden über `UNION` vereinigt. Die Selektionsbedingungen sind für den jeweiligen `SELECT`-Befehl zu definieren:

```
SELECT * FROM kunden_meier
   WHERE anrede = 'Frau'
UNION
SELECT * FROM kunden_schmidt
   WHERE ort = 'Berlin';
```

Übung 5.19

Bilden Sie eine Vereinigungsmenge aus der Tabelle `kunden_meier` mit der Tabelle `kunden_schmidt`. Die Ausgabe soll auch Datensätze, die mehrfach in den Tabellen vorkommen, mehrfach enthalten.

Damit die in den Tabellen mehrfach geführten Datensätze auch mehrfach aufgelistet werden, muss der Befehl `UNION ALL` verwendet werden:

```
SELECT * FROM kunden_meier
UNION ALL
SELECT * FROM kunden_schmidt;
```

Übung 5.20

Wie hoch ist der Durchschnittsverdienst der Angestellten der Beispielfirma insgesamt und nach Abteilungen gruppiert?

Zuerst suchen Sie das durchschnittliche Gehalt der Firma insgesamt:

```
SELECT AVG(gehalt)
   FROM mitarbeiter;
```

Nun müssen Sie auch die Abteilung abfragen und die Ausgabe nach ihr gruppieren:

```
SELECT abteilung, AVG(gehalt)
   FROM mitarbeiter
   GROUP BY abteilung;
```

Übung 5.21

Ermitteln Sie das Eintrittsdatum des Mitarbeiters, der zuletzt in die Firma eintrat.

Die Funktion `MAX()`kann auch auf Datumswerte angewendet werden. Wenn Sie das höchste Eintrittsdatum suchen, finden Sie den entsprechenden Datensatz:

```
SELECT MAX(eintrittsdatum)
FROM mitarbeiter;
```

Übung 5.22

Wie groß ist die höchste Bestellmenge in der Tabelle `posten`?