10 Anhang: Die Beispieldatenbank Bike

Die Beispieldatenbank *Bike* wird in diesem Buch zu Übungszwecken durchgängig verwendet. Diese Datenbank ist der Praxis entnommen: Der hier betrachtete Fahrradhändler kann seine Aufgaben mit dieser Datenbank verwalten, auch wenn noch Wünsche offen bleiben. Dafür ist diese Datenbank mit nur zehn Relationen sehr übersichtlich und daher optimal für den Anfänger geeignet.

Es heißt, wir lernen aus Fehlern, und so enthält die Datenbank *Bike* einige Designschwächen. Nicht jede Relation befindet sich in der 3. Normalform, auch inhaltlich ist die Datenbank *Bike* nicht immer optimal aufgebaut. Im Buch wird gezielt auf diese Problemfälle aufmerksam gemacht. Davon unabhängig sei als Übung empfohlen, die noch enthaltenen Designfehler zu identifizieren und zu beseitigen.

Diese Beispieldatenbank muss zunächst installiert werden. Im ersten Abschnitt finden sich Hinweise zu Oracle, SQL Server und MySQL. Die Installationsskripte sind aber auch leicht auf andere Datenbanken portierbar.

Im zweiten Abschnitt stellen wir die Beispieldatenbank vor. Ein Entity-Relationship-Diagramm gibt einen ersten Überblick. Danach werden alle zehn Relationen im Detail behandelt. Zu ausgewählten Relationen folgen schließlich die *Create-Table*-Befehle zum Erzeugen dieser Basisrelationen, und im letzten Abschnitt zeigen wir schließlich noch einige komplexere PHP-Programmbeispiele mit Zugriffen auf diese Datenbank.

10.1 Hinweise zum Download von Materialien

Die Installationsskripte für die Datenbank *Bike* werden kostenlos für Oracle, MS SQL Server und MySQL im Internet unter der folgenden Adresse zur Verfügung gestellt:

http://bike.hs-regensburg.de

Auf dieser Seite sind ferner verfügbar:

- Alle PHP-Programme und die umfangreichen Lösungen aus Kapitel 6
- Alle Beispielprogramme zur Objektorientierung aus Kapitel 9
- Die Foliensätze zu allen Kapiteln im PDF-Format
- Installationsskript zum Bierdepot aus Kapitel 1

Auf dieser Downloadseite wird zusätzlich eine installierte Bike-Datenbank angeboten. Auf diese Datenbank kann mit Select-Befehlen direkt zugriffen werden.

Ein Hinweis auf diesen Download-Bereich findet sich auch auf der Homepage des Springer-Vieweg-Verlags unter:

http://www.springer-vieweg.de

10.2 Installation der Datenbank Bike

Wir setzen voraus, dass eine Datenbank zur Verfügung steht. Sowohl Oracle als auch Microsoft bieten kostenlose abgespeckte Versionen ihrer Datenbanken an. Diese Versionen sind für Übungszwecke voll ausreichend. In Tab. 10.1 sind diese zusammen mit der derzeit aktuellen Internetadresse aufgelistet. Die Installation dieser Systeme dauert ca. 15 bis 30 Minuten. Anschließend steht ein voll funktionsfähiges Datenbanksystem mit entsprechenden grafischen Benutzerschnittstellen zur Verfügung. Unter Oracle sollte zusätzlich die Oberfläche SQL Developer installiert werden. Auch diese Software ist frei verfügbar.

Tab. 10.1 Voll funktionsfähige Datenbanksysteme

Oracle	Oracle Database Express Edition Oracle SQL Developer	www.oracle.de (Downloads)
Microsoft	Microsoft SQL Server Express Edition	www.microsoft.com/de-de/download (Tools für Entwickler, Serverprodukte)
Oracle	MySQL Community Server	www.mysql.de (Downloads)

Diese Systeme legen bei der Installation mindestens eine Systemkennung an. Es ist wichtig, sich das Passwort dieser Systemkennung zu merken. Nur so können beispielsweise weitere Kennungen und Schemata eingerichtet werden. Ist diese erste Hürde genommen, oder existiert bereits ein Zugriff auf eine Daten-

bank, so kann die Beispieldatenbank *Bike* installiert werden. Die entsprechenden Installationsskripte können im Internet heruntergeladen werden, siehe hierzu den vorherigen Abschnitt.

In Tab. 10.2 sind die angebotenen Installationsskripte aufgelistet. Alle diese Skripte sind Textdateien, die die erforderlichen SQL-Befehle zur Installation enthalten. Es wird empfohlen, diese Skripte vor dem Ausführen in einem Texteditor zu öffnen. Diese Skripte enthalten in den ersten Zeilen einige zusätzliche Hinweise zur Installation auf dem entsprechenden System.

Tab. 10.2 Installationsskripte für Datenbank Bike

Oracle	bikeOracle.sql
Microsoft	bikeSQLServer.sql
Oracle	bikeMySQL.sql

In allen drei Datenbanksystemen ist letztlich die entsprechende Datei zu öffnen und auszuführen. Nach wenigen Sekunden ist die Datenbank *Bike* inklusive aller Daten installiert. Die Datenbank ist nun bereit für Abfragen. Auch Änderungen können problemlos ausgeführt werden. Die Originaldatenbank ist jederzeit wiederherstellbar. Dazu muss das Skript nur nochmals ausgeführt werden.

10.3 Die Datenbank Bike

Die Datenbank *Bike* besitzt zwei wichtige Bereiche, die Verwaltung der Artikel und das Auftragswesen. Darüber hinaus gibt es Lieferanten-, Kunden- und Personal-Entitäten. Einen sehr guten Überblick gibt das Entity-Relationship-Diagramm der Datenbank in Abb. 10.1.

Betrachten wir zunächst die Verwaltung der Artikel: Wir erkennen eine Entität *Artikel*. Diese enthält alle Artikel, beginnend von Einzelteilen bis zum komplett montierten Fahrrad. Weiter finden wir eine Entität *Lager*, aus der der momentane Lagerbestand ersichtlich ist. Den Aufbau komplexer Artikel aus Einzelteilen erhalten wir schließlich mittels einer Beziehungsentität *Teilestruktur*. Eine vereinfachte Lieferverwaltung ist mittels der beiden Relationen *Lieferant* und *Lieferung* gegeben.

Etwas komplexer ist das Auftragswesen aufgebaut: Zunächst liegt eine Entität *Auftrag* vor, in der alle eingehenden Aufträge verwaltet werden. Sie besitzt Beziehungen zur *Kunden*- und *Personal*-Entität. Einzelheiten zu den einzelnen

Aufträgen finden wir in der Entität *Auftragsposten*. Hier sind alle Einzelpositionen zu den einzelnen Aufträgen aufgeführt. Diese Beziehungsentität besitzt je eine Beziehung zu den Entitäten *Artikel* und *Auftrag*. Damit können wir in dieser Entität ablegen, welche Artikel zu welchem Auftrag gehören. Der Händler baut gelegentlich Fahrräder individuell zusammen. Bei solchen Aufträgen reserviert er die entsprechenden Zwischenartikel und vermerkt dies in der Entität *Reservierung*.

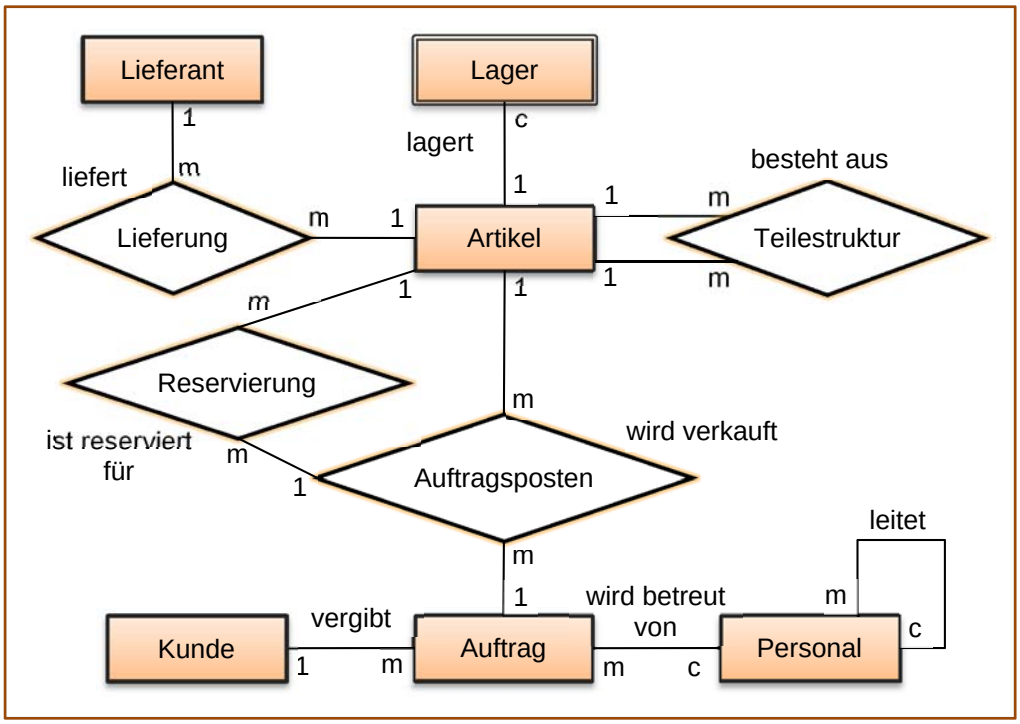

Abb. 10.1 Das Entity-Relationship-Modell der Bike-Datenbank

Im Entity-Relationship-Diagramm erkennen wir eine schwache Entität und vier Beziehungsentitäten. Jede Verbindungslinie entspricht genau einem Fremdschlüssel. Verwendet wird die in diesem Buch vorgestellte Notation.

Ein besonderes Augenmerk sei auf die Beziehungsentität *Teilestruktur* gerichtet. Diese Beziehungsrelation heißt Stückliste, da sie angibt, wie Artikel aus einfacheren Komponenten zusammengesetzt sind. Diese Entität *Teilestruktur* besitzt zwei Fremdschlüssel auf die Relation *Artikel*.

10.4 Die Relationen der Bike-Datenbank

Das Entity-Relationship-Modell aus Abb. 10.1 enthält zehn Entitäten, davon vier Beziehungsentitäten. Diese Entitäten sind die Basis unserer relationalen Datenbank und werden eins zu eins in Relationen umgesetzt. Die vier Beziehungsentitäten lösen *m* zu n Beziehungen auf, die in relationalen Datenbanken immer als eigenständige Relationen realisiert werden. Jede der zehn Relationen besitzt einen Primärschlüssel. Jede der zwölf Verbindungslinien repräsentiert einen Fremdschlüssel. Dieser Fremdschlüssel befindet sich immer in der Relation, dem die *m* Beziehung zugeordnet ist. Bei der einzigen *c* zu 1 Beziehung zwischen den Relationen *Lager* und *Artikel* enthält die schwache Entität *Lager* den Fremdschlüssel. Das Entity-Relationship-Modell (ERM) führt also direkt zu den Relationen unserer relationalen Datenbank. Wir stellen im Folgenden diese Relationen zusammen mit Beispieldaten vor. Null-Werte werden durch den Eintrag *NULL* gekennzeichnet.

- Lieferant: Diese Relation beinhaltet als Attribute (Eigenschaften) den Firmennamen, den Wohnort, die Postleitzahl, die Straße und die Hausnummer. Zusätzlich markiert das Attribut *Sperre* (in Tab. 10.3 mit *Sp* abgekürzt), ob ein Sperrvermerk für diesem Lieferanten vorliegt, etwa wegen Zahlungsverzögerungen in der Vergangenheit. In der Praxis werden häufig noch weitere Daten zu Statistik- oder Informationszwecken aufgenommen.

Tab. 10.3 Relation Lieferant der Beispieldatenbank

Nr	Name	Strasse	PLZ	Ort	Sp.
1	Firma Gerti Schmidtn.	Dr.Gesslerstr.59	93051	Regensburg	0
2	Rauch GmbH	Burgallee 23	90403	Nürnberg	0
3	Shimano GmbH	Rosengasse 122	51143	Köln	0
4	Suntour LTD	Meltonstreet 65	NULL	London	0
5	MSM GmbH	St-Rotteneckstr.13	93047	Regensburg	0

- Kunde: Diese Relation entspricht vom Aufbau der Relation *Lieferant*. Das *Sperre*-Attribut (in Tab. 10.4 mit *Sp* abgekürzt) kennzeichnet hier, ob etwa mangels Liquidität an den Kunden noch etwas verkauft werden darf.

Tab. 10.4 Relation Kunde der Beispieldatenbank

Nr	Name	Strasse	PLZ	Ort	Sp.
1	Fahrrad Shop	Obere Regenstr. 4	93059	Regensburg	0
2	Zweirad-Center Stal.	Kirschweg 20	44267	Dortmund	0
3	Maier Ingrid	Universitätsstr. 33	93055	Regensburg	1
4	Rafa - Seger KG	Liebigstr. 10	10247	Berlin	0
5	Biker Ecke	Lessingstr. 37	22087	Hamburg	0
6	Fahrräder Hammerl	Schindlerplatz 7	81739	München	0

- Personal: Diese Relation ist ähnlich wie die Relationen *Lieferant* und *Kunde* aufgebaut. Zusätzlich sind noch das Geburtsdatum, der Familienstand, der Vorgesetzte, das Gehalt, eine persönliche Beurteilung und die Aufgabe in der Firma (Arbeiter, Vertreter, ...) aufgeführt.

Tab. 10.5 Relation Personal der Beispieldatenbank (Teil1)

Persnr	Name	Strasse	PLZ	Ort
1	Maria Forster	Ebertstr. 28	93051	Regensburg
2	Anna Kraus	Kramgasse 5	93047	Regensburg
3	Ursula Rank	Dreieichstr. 12	60594	Frankfurt
4	Heinz Rolle	In der Au 5	90455	Nürnberg
5	Johanna Köster	Wachtelstr. 7	90427	Nürnberg
6	Marianne Lambert	Fraunhofer Str. 3	92224	Landshut
7	Thomas Noster	Mahlergasse 10	93047	Regensburg
8	Renate Wolters	Lessingstr. 9	86159	Augsburg
9	Ernst Pach	Olgastr. 99	70180	Stuttgart

Tab. 10.6 Relation Personal (Teil2)

Persnr	GebDatum	Stand	Vorg.	Gehalt	Beurt.	Aufgabe
1	05.07.79	verh	NULL	4800.00	2	Manager
2	09.07.75	led	1	2300.00	3	Vertreter
3	04.09.67	verh	6	2700.00	1	Facharbeiterin
4	12.10.57	led	1	3300.00	3	Sekretär
5	07.02.84	gesch	1	2100.00	5	Vertreter
6	22.05.74	verh	NULL	4100.00	1	Meister
7	17.09.72	verh	6	2500.00	5	Arbeiter
8	14.07.79	led	1	3300.00	4	Sachbearbeiter
9	29.03.92	led	6	800.00	NULL	Azubi

In der Praxis werden meist noch weitere Daten aufgenommen, etwa das Einstelldatum, die Vorbildung und verbale Beurteilungen. Aus Platzgründen wurde die Relation *Personal* in zwei Teiltabellen (Tab. 10.5 und Tab. 10.6) aufgeteilt. Der Primärschlüssel *Persnr* ist jeweils mit angegeben, und die Attribute *Vorgesetzt* und *Beurteilung* wurden mit *Vorg* bzw. *Beurt* abgekürzt.

- Auftrag: Diese Relation ist gegenüber den Anforderungen in der Praxis erheblich vereinfacht. Zu jedem Auftrag mit Primärschlüssel *AuftrNr* existieren nur folgende Daten: Datum, Kundennummer und Mitarbeiternummer (*Persnr*).

Tab. 10.7 Relation Auftrag der Beispieldatenbank

AuftrNr	Datum	Kundnr	Persnr
1	04.01.2013	1	2
2	06.01.2013	3	5
3	07.01.2013	4	2
4	18.01.2013	6	5
5	03.02.2013	1	2

- Auftragsposten: Diese Beziehungsrelation enthält alle wichtigen Auftragsdaten. Sie nimmt in der Datenbank eine zentrale Rolle ein, da sie mit den Relationen *Auftrag* und *Artikel* in Beziehung steht. Mit der Relation *Artikel* bestehen sogar zwei Beziehungen, zum einen direkt und zum anderen über die Beziehungsrelation *Reservierung*. Im ersten Fall wird der in Auftrag gegebene Artikel gemerkt, im zweiten Fall die zu reservierenden Einzelteile.

Tab. 10.8 Relation Auftragsposten der Beispieldatenbank

PosNr	AuftrNr	Artnr	Anzahl	Gesamtpreis
101	1	200002	2	800,00
201	2	100002	3	1.950,00
202	2	200001	1	400,00
301	3	100001	1	700,00
302	3	500002	2	100,00
401	4	100001	1	700,00
402	4	500001	4	30,00
403	4	500008	1	94,00
501	5	500010	1	40,00
502	5	500013	1	30,00

- Artikel: In dieser Relation wird jedes Teil durch eine Bezeichnung, einen Netto- und Bruttopreis inklusive Steuer und gegebenenfalls durch eine Maßangabe (Abmessung), die Maßeinheit (z.B. *ST* für Stück oder *CM* für Zentimeter) beschrieben. Das Attribut *Typ* spezifiziert, ob ein Endprodukt (*E*), ein zusammengesetztes Teil (*Z*) oder ein Fremdartikel (*F*) vorliegt. Fremdartikel sind Artikel, die immer von Lieferanten geliefert werden. Aus Platzgründen sind in Tab. 10.9 die Attribute *Einheit* und *Typ* durch *Ei* und *T* abgekürzt.

Tab. 10.9 Relation Artikel der Beispieldatenbank

ANr	Bezeichnung	Netto	Steuer	Preis	Farbe	Mass	Ei.	T.
100001	Herren-City-Rad	588.24	111.76	700.00	blau	26 Zoll	ST	E
100002	Damen-City-Rad	546.22	103.78	650.00	rot	26 Zoll	ST	E
200001	He-Rahmen lack.	336.13	63.87	400.00	blau	NULL	ST	Z
200002	Da-Rahmen lack.	336.13	63.87	400.00	rot	NULL	ST	Z
300001	He-Rahmen gesch.	310.92	59.08	370.00	NULL	NULL	ST	Z
300002	Da-Rahmen gesch.	310.92	59.08	370.00	NULL	NULL	ST	Z
400001	Rad	58.82	11.18	70.00	NULL	26 Zoll	ST	Z
500001	Rohr 25CrMo4 9mm	6.30	1.20	7.50	NULL	9 mm	CM	F
500002	Sattel	42.02	7.98	50.00	NULL	NULL	ST	F
500003	Gruppe Deore LX	5.88	1.12	7.00	NULL	LX	ST	F
500004	Gruppe Deore XT	5.04	0.96	6.00	NULL	XT	ST	F
500005	Gruppe XC-LTD	6.72	1.28	8.00	NULL	Xc-Ltd	ST	F
500006	Felgensatz	33.61	6.39	40.00	NULL	26 Zoll	ST	F
500007	Bereifung Schwalbe	16.81	3.19	20.00	NULL	26 Zoll	ST	F
500008	Lenker + Vorbau	78.99	15.01	94.00	NULL	NULL	ST	F
500009	Sattelstütze	4.62	0.88	5.50	NULL	NULL	ST	F
500010	Pedalsatz	33.61	6.39	40.00	NULL	NULL	ST	F
500011	Rohr 34CrMo4 2.1	3.36	0.64	4.00	NULL	2,1 mm	CM	F
500012	Rohr 34CrMo3 2.4	3.61	0.69	4.00	NULL	2,4 mm	CM	F
500013	Tretlager	25.21	4.79	30.00	NULL	NULL	ST	F
500014	Gabelsatz	10.08	1.92	12.00	NULL	NULL	ST	F
500015	Schlauch	6.72	1.28	8.00	NULL	26 Zoll	ST	F

- Teilestruktur: Diese Relation enthält zu allen Artikeln (*Artnr*) aus der Relation *Artikel*, die aus einfacheren Teilen zusammengesetzt sind, die entsprechenden Einzelteile (*Einzelteilnr*). Das Attribut *Anzahl* gibt die Anzahl der benötigten Einzelteile an, und das Attribut *Einheit* bezieht sich

auf die Maßeinheit des Einzelteils. Besteht ein Artikel aus mehreren verschiedenen einfachen Teilen, so werden entsprechend viele Tupel in dieser Relation eingetragen. Diese Relation *Teilestruktur* entspricht einer einfachen Stückliste, die in der Praxis häufig vorkommt.

Tab. 10.10 Relation Teilestruktur der Beispieldatenbank

Artnr	Einzelteilnr	Anzahl	Einheit
100001	200001	1	ST
100001	500002	1	ST
100001	500003	1	ST
100001	400001	1	ST
100001	500008	1	ST
100001	500009	1	ST
100001	500010	1	ST
100002	200002	1	ST
100002	500002	1	ST
100002	500004	1	ST
100002	400001	1	ST
100002	500008	1	ST
100002	500009	1	ST
100002	500010	1	ST
200001	300001	1	ST
200002	300002	1	ST
300001	500001	180	CM
300001	500011	161	CM
300001	500012	20	CM
300001	500013	1	ST
300001	500014	1	ST
300002	500001	360	CM
300002	500011	106	CM
300002	500012	20	CM
300002	500013	1	ST
300002	500014	1	ST
400001	500007	2	ST
400001	500006	1	ST
400001	500015	2	ST

- Lager: Diese Relation enthält neben dem Artikel den Lagerort und den Bestand. Weiter wird ein Mindestbestand, die Anzahl der reservierten sowie der bereits bestellten Stücke gespeichert. Nicht jedes in der Relation *Artikel* angegebene Teil muss in dieser Relation enthalten sein. Teile, die weder auf Lager, noch reserviert oder bestellt sind, brauchen hier nicht aufgeführt zu werden.

Tab. 10.11 Relation Lager der Beispieldatenbank

Artnr	Lagerort	Bestand	Mindbest	Reserviert	Bestellt
100001	001002	3	0	2	0
100002	001001	6	0	3	0
200001	NULL	0	0	0	0
200002	004004	2	0	0	0
300001	NULL	0	0	0	0
300002	002001	7	0	2	0
400001	005001	1	0	0	0
500001	003005	8050	6000	184	0
500002	002002	19	20	2	10
500003	001003	15	10	0	0
500004	004001	18	10	0	0
500005	003002	2	0	0	0
500006	003004	21	20	0	0
500007	002003	62	40	0	0
500008	003003	39	20	1	0
500009	002007	23	20	0	0
500010	001006	27	20	1	0
500011	001007	3250	3000	161	0
500012	004002	720	600	20	0
500013	005002	20	20	2	0
500014	005003	27	20	1	0
500015	002004	55	40	0	0

- Reservierung: Diese Relation gibt darüber Auskunft, welche Artikel für welchen Auftragsposten reserviert wurden. Diese Beziehungsrelation besitzt je einen Fremdschlüssel auf die Relationen *Artikel* und *Auftragsposten*, die zusammen den Primärschlüssel bilden. Ein weiteres Attribut speichert die Anzahl der einzelnen reservierten Artikel.

Tab. 10.12 Relation Reservierung der Beispieldatenbank

Posnr	Artnr	Anzahl
101	300002	2
201	100002	3
202	500001	180
202	500011	161
202	500012	20
202	500013	1
202	500014	1
301	100001	1
302	500002	2
401	100001	1
402	500001	4
403	500008	1
501	500010	1
502	500013	1

- Lieferung: Diese Relation gibt an, welche Artikel von welchem Lieferanten in welcher Lieferzeit geliefert werden. Zusätzlich werden der Nettoeinkaufspreis und die laufenden Bestellungen gemerkt.

Tab. 10.13 Relation Lieferung der Beispieldatenbank

ANr	Liefnr	Lieferzeit	Nettopreis	Bestellt
500001	5	1	6.50	0
500002	2	4	71.30	10
500002	1	5	73.10	0
500003	3	6	5.60	0
500003	4	5	6.00	0
500003	2	4	5.70	0
500004	3	2	5.20	0
500004	4	3	5.40	0
500005	4	5	6.70	0
500006	1	1	31.00	0
500007	1	2	16.50	0
500008	1	4	83.00	0
500009	1	2	4.10	0
500009	2	1	4.60	0
500010	1	3	35.20	0

500011	5	1	3.10	0
500012	5	1	3.40	0
500013	1	4	21.00	0
500014	1	5	9.20	0
500015	1	1	6.20	0

10.5 Deklaration der Bike-Datenbank

Die *Bike*-Datenbank basiert auf dem Entity-Relationship-Diagramm aus Abb. 10.1. Im letzten Abschnitt haben wir bereits die Eigenschaften und damit die Attribute der zehn erforderlichen Basisrelationen angegeben und die einzelnen Relationen mit Inhalt gefüllt. Wir geben jetzt zu einigen Relationen die Create-Table-Befehle an. Die Attribute der einzelnen Relationen sind bereits aus dem letzten Abschnitt bekannt. Wir konzentrieren uns daher auf die Primär- und Fremdschlüssel dieser Relationen, um die beiden wichtigen Integritätsregeln (Entität und Referenz) zu erfüllen. Wir empfehlen dringend, diese Schlüssel zunächst selbst zu erarbeiten. Als Hinweis zu den Fremdschlüsseln sei angemerkt, dass es genauso viele Fremdschlüssel wie Verbindungen im Entity-Relationship-Modell geben muss.

Im Folgenden sind Create-Table-Befehle zu den Relationen *Artikel*, *Auftrag*, *Auftragsposten*, *Teilestruktur* und *Lieferung* aufgeführt. Die weiteren seien als Übung empfohlen, siehe hierzu auch die Übungsaufgaben in Abschnitt 5.15. Aus Übersichtsgründen wurde teilweise auf die Überprüfung der semantischen Integrität mittels zusätzlicher Tabellen- oder Spaltenbedingungen verzichtet. Diese Erweiterungen der *Create-Table*-Befehle seien wieder als Übung empfohlen. Beginnen wir mit der Relation *Artikel*:

```
CREATE TABLE  Artikel
(   ANr           INT             PRIMARY KEY,
    Bezeichnung CHAR (35)       NOT NULL,
     Nettopreis   NUMERIC(7,2)  CHECK ( Nettopreis > 0),
     Steuer       NUMERIC(7,2)  CHECK ( Steuer > 0),
     Preis        NUMERIC(7,2)  CHECK ( Preis > 0),
     Farbe        CHAR (10),
     Mass         CHAR (15),
     Einheit      CHAR (2)       NOT NULL,
     Typ          CHAR (1)     NOT NULL CHECK ( Typ IN ('E', 'Z', 'F') ),
) ;
```

In dieser Relation ist das Attribut *ANr* der Primärschlüssel, die Beträge müssen positiv sein, und einige Angaben zu den Teilen müssen zwingend angegeben werden. Als Artikeltyp wird einer von drei Werten vorgegeben.

```
CREATE TABLE  Auftrag
(  AuftrNr     INT      PRIMARY KEY,
   Datum       DATE,
   Kundnr      INT      NOT NULL  REFERENCES Kunde
                        ON DELETE NO ACTION ON UPDATE CASCADE,
   Persnr      INT      REFERENCES Personal
                        ON DELETE SET NULL  ON UPDATE CASCADE
);
```

Die Relation *Auftrag* ist einfach aufgebaut. In den beiden Fremdschlüsseln wird allerdings beim Löschen des entsprechenden Kunden- bzw. Personaleintrags unterschiedlich reagiert. Ein Kunde ist so eng mit einem Auftrag verbunden, dass dieser Kunde nicht aus der Kundentabelle entfernt werden darf, wenn einer seiner Aufträge noch existiert. Scheidet hingegen ein Mitarbeiter aus, so wird der entsprechende Eintrag im Auftrag einfach auf NULL gesetzt.

```
CREATE TABLE Auftragsposten
(  PosNr       INT      PRIMARY KEY,
   AuftrNr     INT      NOT NULL REFERENCES Auftrag
                        ON DELETE CASCADE ON UPDATE CASCADE,
   Artnr       INT      NOT NULL REFERENCES Artikel
                        ON DELETE NO ACTION ON UPDATE CASCADE,
   Anzahl      SMALLINT,
   Gesamtpreis  NUMERIC(10,2) CHECK (Gesamtpreis > 0),
   UNIQUE (AuftrNr, Artnr)
);
```

Die Beziehungsrelation *Auftragsposten* besitzt zwei Fremdschlüssel, einen zur Relation *Auftrag*, um jede Auftragsposition einem Auftrag eindeutig zuordnen zu können. Weiter wird der zum Auftrag gehörende Artikel mit der Relation *Artikel* verknüpft. Auch hier sind die Merkmale der Fremdschlüssel unterschiedlich: Wird ein Auftrag gelöscht, so sind auch die Einzelpositionen nicht mehr nötig. Artikel dürfen hingegen nicht entfernt werden, wenn noch dazugehörige Aufträge existieren. Zu beachten ist ferner der in einer Beziehungsrelation vorkommende zusammengesetzte Schlüsselkandidat, der mit der Tabellenbedingung *Unique* gekennzeichnet wurde. Wir haben uns hier für einen eigenen

Primärschlüssel (*Posnr*) entschieden, da noch von anderer Seite auf diese Relation verwiesen wird.

```
CREATE TABLE Lieferung
(  ANr           INT  REFERENCES Artikel
                      ON DELETE CASCADE   ON UPDATE CASCADE,
   Liefnr        INT  REFERENCES Lieferant
                      ON DELETE CASCADE   ON UPDATE CASCADE,
   Lieferzeit    SMALLINT,
   Nettopreis    NUMERIC(7,2),
   Bestellt      SMALLINT,
   PRIMARY KEY (ANr, Liefnr)
);
```

Die Relation *Lieferung* ist eine weitere Beziehungsrelation. Hier bilden die beiden Fremdschlüssel auch den gemeinsamen Primärschlüssel. Diese Relation beschreibt, welche Teile welcher Lieferant zu welchem Preis liefert. Existiert ein Lieferant nicht mehr, so sind auch Einträge in dieser Relation nicht mehr nötig, daher *On Delete Cascade*. Existiert ein Artikel nicht mehr, so ist dieser Eintrag ebenfalls nicht mehr nötig. Je nach interner Organisation ist hier statt *On Delete Cascade* auch *On Delete No Action* denkbar.

```
CREATE TABLE  Teilestruktur
(  Artnr         INT  REFERENCES Artikel
                      ON DELETE CASCADE ON UPDATE CASCADE,
   Einzelteilnr  INT  REFERENCES Artikel
                      ON DELETE NO ACTION ON UPDATE CASCADE,
   Anzahl        INT,
   Einheit       CHAR (2),
   PRIMARY KEY (ANr, Einzelteilnr)
) ;
```

Die Relation *Teilestruktur* ist ebenfalls eine Beziehungsrelation: Es existieren zwei Fremdschlüssel, die in unserem Fall auf die gleiche Relation verweisen. Weiter besteht der Primärschlüssel aus eben diesen beiden Fremdschlüsselattributen. Beachten wir deshalb, dass hier beim Löschen oder Ändern von Fremdschlüsselwerten die Option *Set Null* nicht erlaubt ist! Wieder reagieren wir beim Löschen unterschiedlich. Wird ein Artikel gelöscht, so interessiert die Zusammensetzung dieses Artikels nicht mehr, wir können diese Informationen kaskadierend löschen. Ein Einzelteil darf hingegen erst dann gelöscht werden, wenn keine Verweise mehr darauf existieren.

Zuletzt sei noch erwähnt, dass nicht alle Datenbanken diesen hier verwendeten SQL-Standard vollständig unterstützen. Beispielsweise müssen in Oracle die *On-Update*-Optionen weggelassen werden, ebenso *On Delete No Action*. In SQL Server müssen einige Angaben geändert werden, da dieser sonst theoretisch mögliche rekursive Konstrukte verbietet. In MySQL sind bei Fremdschlüsseln Spaltenbedingungen nicht erlaubt. Es müssen Tabellenbedingungen verwendet werden. Die einzelnen Installationsskripte berücksichtigen diese Voraussetzungen.

Zu beachten ist beim Erzeugen der Datenbank die Reihenfolge der DDL-Befehle, da bei Bezügen auf andere Relationen (Referenzangaben) diese anderen Relationen bereits existieren müssen. In ganz wenigen seltenen Fällen kann es in der Praxis vorkommen, dass Relationen direkt oder indirekt gegenseitig aufeinander verweisen. Dann müssen in den *Create-Table*-Befehlen einzelne Referenzen zunächst weggelassen und mit Hilfe des *Alter-Table*-Befehls erst später hinzugefügt werden. In unserem Beispiel sind zunächst die Relationen *Lieferant, Kunde, Artikel* und *Personal* zu erzeugen; denn diese Relationen haben gemeinsam, dass sie keine Fremdschlüssel enthalten.

10.6 Zugriffe auf die Bike-Datenbank

In den letzten Abschnitten haben wir die *Bike*-Datenbank spezifiziert und mittels mehrerer Create-Table-Befehle erstellt. Mit dem folgenden Beispiel wollen wir demonstrieren, dass wir auf diese Datenbank auch zugreifen können. Wir wollen über das Internet mittels der Sprache PHP einen bestehenden Auftrag komplett löschen. Dies erfordert Lösch- und Änderungsaktionen in bis zu drei Relationen. Wir setzen hier HTML- und PHP-Kenntnisse voraus. Das Beispielprogramm *loeschauftrag.php* lautet:

```
<html>
<head>
<title>Löschen eines Auftrags aus der Datenbank BIKE</title>
<meta name="description" content="Löschen in der BIKE-DB">
<meta name="author" content="Edwin Schicker">
</head>
<body>
<center><h1>Datenbanken und SQL</h1></center>
<center><h3>Edwin Schicker</h3></center>
<!-- Einloggen in die Datenbank: -->
<p>Programm zum Löschen eines Auftrags<br></p>
```

```
<p>Bitte Kennung, Passwort und zu löschenden Auftrag eingeben:</p>
<!-- Eingabeformular definieren: -->
<form action="loeschauftrag.php" method="post">
<table cellpadding=10>
<tr>
 <td align=right>Datenbank-Kennung: </td>
 <td><input type="Text" name="Kennung" size="20" > </td>
</tr><tr>
 <td align=right>Datenbank-Passwort: </td>
 <td><input type="Password" name="Passwort" size="20" > </td>
</tr><tr>
 <td align=right >Datenbankname(z.B. ora11g oder xe): </td>
 <td><input type="Text" name="Connect" size="40"> </td>
</tr><tr>
 <td align=right>Nummer des zu löschenden Auftrags: </td>
 <td><input type="Text" name="Nr" size="10"> </td>
</tr><tr>
<td></td>
 <td align=right> <input type="Submit" value="Weiter"> </td>
</tr>
</table>
</form>
<hr noshade size="1">
<?php
 // Der folgende Code wird nur ausgefuehrt, wenn die Post-Variable
 // Kennung bereits einmal uebergeben wurde!
if (isset($_POST['Kennung'])) {
  try {
    // neues PDO-Objekt anlegen und mit Oracle-Datenbank verbinden:
    $conn = new PDO("oci:dbname=$_POST[Connect]",
                                  $_POST['Kennung'], $_POST['Passwort']);
    $conn->beginTransaction();                        // Transaktionsmodus

    // Ueberpruefen, ob Auftrag existiert:
    $sql =    "Select Auftrnr
               From Auftrag
               Where Auftrnr = $_POST[Nr]";
    $stmt = $conn->query($sql);    // Select Befehl ausfuehren

    if (!$stmt->fetch())
      die ("Die angegebene Auftragsnummer existiert nicht!");    // Abbruch
```

```
    // Loeschen der Reservierungen zu dem Auftrag:
    $sql =   "Delete From Reservierung
              Where Posnr In (  Select Posnr
                                From Auftragsposten
                                Where Auftrnr = $_POST[Nr])";
    $stmt = $conn->query($sql);

    // Loeschen des Auftrags:
    $sql =   "Delete From Auftrag
               Where Auftrnr = $_POST[Nr]";
    $stmt = $conn->query($sql);
    echo "Der Auftrag $_POST[Nr] wurde geloescht.";
    // Auftragspositionen werden wegen Delete Cascade gelöscht !

     // Die Datenbank wird jetzt geschlossen:
    $conn->commit();
  }  // end try
  catch (Exception $e)  {
    echo "Das Programm endete mit Fehler: ".$e->getMessage();
  }
} // endif isset
?>
</body>
</html>
```

In einer einfachen Eingabemaske werden Kennung, Passwort, Datenbank und Auftragsnummer eingegeben. Anschließend loggen wir uns in eine Oracle-Datenbank ein. Ist die eingegebene Auftragsnummer korrekt, werden die Einträge zu diesem Auftrag in den Relationen *Reservierung*, *Auftrag* und *Auftragsposten* gelöscht. Tritt ein Fehler auf, so wird der Try-Block verlassen und in den Catch-Block gesprungen.

Wir haben die datenbankunabhängige Schnittstelle PDO gewählt. Wir müssen nur das Erzeugen des PDO-Objekts *$conn* anpassen, um auch auf SQL Server oder MySQL zugreifen zu können. Weiter setzt PDO im Catch-Teil automatisch den Befehl *Rollback* ab. Somit wird der Transaktionsmodus im Fehlerfall immer sichergestellt.

Zu ausführlichen Informationen zum Programmieren mit PHP und PDO sei auf Kapitel 6 verwiesen.