9 Beispiel: In 12 Schritten zur MySQL-Anwendung

In der Regel werden Datenbanken in eine vollständige Anwendung eingebunden. In diesem Kapitel werden beispielhaft an der Einrichtung eines Online-Shops die einzelnen Schritte gezeigt, in denen eine Anwendung mit MySQL-Unterstützung realisiert werden kann.

9.1 Ziel

In diesem Buch sind eine Reihe von Beispielen zu den jeweiligen Themen gegeben worden. Für eine Anwendung zählt aber in letztendlich nur die Gesamtfunktionalität, um aus der Technik Vorteile, sei es zur einfacheren Verwaltung von Daten oder der effektiveren Erledigung von Arbeitsaufgaben, zu ziehen.

Beispiel: Online-Shop

In diesem Kapitel soll deshalb an einem zusammenhängenden Beispiel gezeigt werden, wie eine MySQL-Anwendung realisiert wird. Das Beispiel nimmt ein Thema auf, das für das E-Business unabdingbar ist: die Koordinierung und Abwicklung von Bestellvorgängen über das Internet durch einen Online-Shop. Internetgestützte Vorgänge bieten für den Benutzer die Möglichkeit der weltweiten Verfügbarkeit rund um die Uhr. Für den Betreiber von Online-Shop-Systemen bietet sich die Möglichkeit, einen höheren Servicelevel für den Kunden zu erreichen (Verfügbarkeitsinformation in Echtzeit, schnelle E-Mail-Kommunikation) sowie Geschäftsvorgänge (Controlling, Prozessketten) effektiver zu gestalten. An diesem Beispiel kann deshalb auch sehr gut nachvollzogen werden, warum Datenbanken kombiniert mit Softwareintelligenz einen wichtigen Platz in digital gesteuerten Prozessketten haben. Die einzelnen Schritte von der Definition der Anwendung über die Erstellung des Datenbankentwurfs bis zur Realisierung der Benutzerschnittstelle einschließlich hierfür notwendiger Funktionen werden im Folgenden dargestellt.

Ratsam ist auf jeden Fall, den gesamten Anwendungsfall vor Beginn der Definition von Datenbank und Tabellen ausreichend genau zu definieren. Auch wenn vielleicht der ein oder andere den Aufwand hierfür scheut, eine gute Anwendungsplanung wird sich immer positiv auswirken. Folgende Gründe können hierfür angeführt werden:

Anwendungsplanung ist unerlässlich

- **Besseres Datenbankdesign**
  Auf der Basis vollständiger Informationen zu Umfang und Inhalt der zu speichernden Daten kann ein optimiertes Datenmodell besser erstellt werden (z. B. Vermeidung von Redundanzen).
- **Benutzerschnittstellen**
  Die Benutzerschnittstellen können kompakt realisiert werden. Ansonsten müssen bei Erweiterung der Anwendung auch die Benutzerschnittstellen erweitert werden. Im schlimmsten Fall müssen ganze Dialogelemente aufgrund neuer Gewichtung von Inhalten oder Platzbedarf neu erzeugt werden.
- **Abfragen (SQL-Befehle)**
  Wenn Sie neue Felder in die Datenbank integrieren, müssen Sie in der Regel auch eine Reihe von SQL-Befehlen überarbeiten. Der Aufwand zur Überarbeitung der Abfragen ist dann nicht zuletzt wegen des notwendigen Bedarfs an Überprüfung der Funktion höher.

9.2 Planung und Definition der Anwendung

9.2.1 Schritt 1: Anwendungsübersicht

Bei der Anwendung handelt es sich um eine typische Shop-Lösung für das Internet. Dem Kunden wird ein Warenangebot in Form eines elektronischen Katalogs angeboten. Er hat die Möglichkeit, in diesem Katalog zu suchen und Waren daraus online zu bestellen.

Merkmale des Online-Shops

Folgende Merkmale soll dieses Beispiel besitzen:

Kunden

- Katalog mit Suchmöglichkeit
- Bestellmöglichkeit/Warenkorbsystem

- Kundenkonto mit der Möglichkeit, Bestellungen und deren Status einzusehen

Interne Verwaltung

- Anlegen neuer Artikel in verschiedenen Kategorien
- Artikelinformationen verwalten (Preis, Beschreibung)
- Bearbeitung der Bestellung: z. B. Weiterleitung an Auslieferung, Rechnungserstellung und Überprüfung des Rechnungseingangs
- Berechnung der Versandkosten nach Gewicht des Artikels
- Bestellstatus (offen, in Bearbeitung, versendet)
- Zahlungseingang überwachen
- Statistikfunktionen (gekaufte Artikel, Kundenstatistik)

Wie Sie sehen, erfolgt die erste Planung am zweckmäßigsten in kurzer Form als Übersicht. Diese Ausarbeitung wird häufig auch als Lastenheft bezeichnet, weil sie den Zweck und den Umfang einer Anwendung allgemein verständlich definiert. Die Anwendung wird in dieser Phase mehr aus der Sicht des »use case« und weniger aus der Sicht der technischen Funktionen formuliert.

9.2.2 Schritt 2: Anwendungsfunktionen

Nachdem die Anwendungsziele definiert sind, werden die Funktionen detailliert spezifiziert. Dies erfolgt zum einen, um anschließend das Datenmodell aufstellen zu können, zum anderen können auf Basis dieser genauen Beschreibung der Funktionalität die Benutzerschnittstellen sowie die notwendigen Softwarefunktionen geplant werden. In der Praxis wird dieser Teil häufig als Pflichtenheft bezeichnet. In dieser Phase wird die Anwendung auch verstärkt bezüglich ihrer technischen Realisierbarkeit geplant. Jetzt werden auch die Funktionen konkret benannt, die relalisert werden sollen. Da unser Beispiel überschaubar bleiben soll, werden hier nur die wichtigsten Punkte aufgelistet.

Die detaillierte Beschreibung der Funktionen sieht wie folgt aus:

Funktionen für den Kunden

- Konto einrichten
  Der Kunde kann ein persönliches Benutzerkonto beim Online-Shop einrichten. Dies dient der Hinterlegung der Versanddaten (Zustelladresse), der Kontaktinformationen (Telefon, E-Mail) sowie der Login-Daten (Anmeldename, Passwort). Nach dem Einrichten erhält der neue Kunde eine E-Mail-Benachrichtigung. Mit den Login-Daten kann der Kunde sein Konto einsehen.
- Artikel suchen
  Der Kunde erhält die Möglichkeit, Artikel des Online-Shops zu suchen.
- Artikel in Warenkorb speichern
  Der Kunde kann Produkte des Online-Shops aussuchen und in den Warenkorb legen. Der Warenkorb ist der virtuelle Einkaufswagen.
- Artikel bestellen
  Der Kunde kann beliebige Artikel bestellen. Der Bestellvorgang wird dabei über den Warenkorb ausgeführt, d.h., der Inhalt des Warenkorbs wird vom Kunden bestätigt.
- Konto prüfen
  Der Kunde kann seine Stammdaten sowie seine Bestellungen überprüfen.

Verwaltungsfunktionen

Für die Verwaltung des Online-Shops sind folgende Funktionen vorgesehen:

- Produkte anlegen
  Über ein Verwaltungsmenü können Produkte und deren Eigenschaften (Produktbeschreibung, Preis etc.) angelegt werden.
- Bestellungen prüfen
  Aktuell eingegangene Bestellungen können überprüft werden.
- Bestellungen bearbeiten
  Der Status einer Bestellung kann die Zustände »offen«, »in Bearbeitung« und »versendet« erhalten. Diese Information bekommt der Kunde angezeigt.
- Produkte löschen
  Löschen von Artikeln aus dem Online-Katalog.

- Kunden verwalten
  Verwaltung der Kundeninformationen einschließlich Kundenstatistiken und gekauften Artikel.

9.3 Datenbankentwurf

9.3.1 Schritt 3: Entitätstypen und Beziehungen ermitteln

Nachdem alle Anforderungen definiert sind, erfolgt im nächsten Schritt die Strukturierung der Daten. Hierzu werden alle Daten, die gespeichert werden, so strukturiert, dass sie zusammenhängende Einheiten (so genannte Entitätstypen) bilden. Die Bildung der Entitätstypen erfolgt im ersten Schritt aus der Überlegung, welche Informationen zusammengehören. Folgende Entitätstypen können für das Beispiel ermittelt werden:

- `Kunden`
  Beinhaltet alle Kundeninformationen, z.B. Namens- und Adressbestandteile
- `Hersteller`
  Enthält alle notwendigen Informationen zu den Herstellern der Produkte
- `Produkte`
  Alle Informationen zu den einzelnen Produkten (z.B. Bezeichnung, Preis etc.)
- `Warenkorb`
  Alle Produkte, die der Kunde während eines Einkaufs in den virtuellen Einkaufswagen legt. Einträge auf dem Warenkorb können während der Einkaufstour wieder gelöscht werden.
- `Bestellungen`
  Enthält die Informationen zu Bestellvorgängen (z.B. Datum der Bestellung)
- `Bestellte Produkte`
  Enthält alle Produkte, die bestellt wurden. Der Unterschied zum Warenkorb besteht darin, dass dieser Entitätstyp die rechtlich verbindlichen Bestellungen enthält und somit eine andere Behandlung erhält (Löschen von Datensätzen nicht zulässig).

Bei der Einteilung von Entitätstypen gilt natürlich auch das Prinzip, Redundanzen zu vermeiden. Die Entitätstypen `Warenkorb` und `Bestellte Produkte` haben mit der Speicherung einer Produktauswahl eigentlich die gleichen Inhalte. An dieser Stelle würde man grundsätzlich überlegen, ob diese nicht zusammengefasst werden können. Die Trennung dieser beiden Entitätstypen erfolgt hier auf Basis der unterschiedlichen Behandlung. Der Warenkorb ist eine temporäre Tabelle, mit vielen Löschvorgängen, der Entitätstyp `Bestellte Produkte` enthält dagegen die rechtlich verbindlichen Produktbestellungen.

Abbildung 9.1 Modellierung der Entitätstypen für das Shop-System

Sie haben also an dieser Stelle schon die erste wichtige Grundlage geschaffen, die Inhalte der Datenbank von ihrer Struktur und Funktion her zu ermitteln.

Jede Entität weist bestimmte Eigenschaften auf, die für die Anwendung relevant sind. Während der Modellierung werden die benötigten Eigenschaften als Attribute des Entitättyps erfasst. Die Eigenschaften werden dabei möglichst fein zerlegt, weil später mit den einzelnen Attributen, z. B. bei Auswertungen, gearbeitet wird. Je genauer diese Attribute unterteilt sind, um so mehr Kombinationen sind später möglich. Die erste Näherung der Beschreibung der Attribute kann verbal erfolgen.

Im relationalen Modell und bei der Umsetzung in die Datenbank werden aus den Entitätstypen dann die Relationen (Tabellen) und aus den Attributen die Tabellenspalten (Felder) mit ihren Datentypen.

Attribute

Im nächsten Schritt müssen also die Attribute der einzelnen Entitätstypen genauer beschrieben werden. Dieser Schritt begrenzt die Informationen zu einem Entitätstyp auf die wesentlichen und bringt Sie in der Modellierung weiter, weil Sie in der Folge mit diesen Merkmalen weiterarbeiten können.

Für die Kunden sollen alle Informationen gespeichert werden, die für die Abwicklung einer Bestellung notwendig sind. Um das Beispiel überschaubar zu halten, werden Attribute, die für die Funktionsfähigkeit nicht gebraucht werden, allerdings in einem umfangreichen Online-Shop vorhanden sind, nicht berücksichtigt. Solche Attribute sind z. B. eine separate Rechnungsadresse, die Verwaltung mehrerer Währungen oder mengenabhängige Preise.

Durchaus sinnvoll ist es, bei der Ermittlung der Attribute bereits den Gültigkeitsbereich der Informationen zu benennen. Für das im nächsten Schritt zu erstellende ER-Modell ist das zwar nicht notwendig, Sie bereiten damit aber schon die spätere Umsetzung in das relationale Datenmodell vor.

Für unser Beispiel ergeben sich folgende Attribute mit entsprechenden Gültigkeitsbereichen:

- Kunden

Attribut	Gültigkeitsbereich
Anrede	Herr, Frau, Firma
Titel	Bezeichnung mit bis zu 20 Zeichen
Name1	Bezeichnung mit bis zu 60 Zeichen
Name2	Bezeichnung mit bis zu 60 Zeichen
Plz	Bezeichnung mit bis zu 14 Zeichen (intern.)
Ort	Bezeichnung mit bis zu 46 Zeichen
Strasse	Bezeichnung mit bis zu 46 Zeichen
Hausnummer	Bezeichnung mit bis zu 6 Zeichen

Attribut	Gültigkeitsbereich
Land	Bezeichnung mit bis zu 30 Zeichen
E-Mail	Bezeichnung mit bis zu 50 Zeichen
Telefonnummer	Bezeichnung mit bis zu 60 Zeichen
Faxnummer	Bezeichnung mit bis zu 60 Zeichen
Anmeldename	Bezeichnung mit 4 bis zu 10 Zeichen
Passwort	Bezeichnung mit bis zu 60 Zeichen

- Hersteller

Attribut	Gültigkeitsbereich
Name	Bezeichnung mit bis zu 80 Zeichen
www	Bezeichnung mit bis zu 80 Zeichen

Die Adresse des Herstellers wird aus Vereinfachungsgründen nicht gespeichert.

- Produkte

Attribut	Gültigkeitsbereich
Artikelnummer	Zahl zwischen 1 und 999999
Bezeichnung	Bezeichnung mit bis zu 60 Zeichen
Beschreibung	Beschreibungstext variabler Länge
Kategorie	Bezeichnung mit bis zu 60 Zeichen
Preis	Zahl zwischen 0,00 und 99999,99
Umsatzsteuer	Prozentzahl
Bilddatei	Bezeichnung mit bis zu 60 Zeichen
Lagermenge	Zahl zwischen 0 und 1000000
Hersteller	Bezeichnung mit bis zu 60 Zeichen
Hinzugefügt	Datum
Zuletzt geändert	Datum
Gewicht (kg)	Zahl zwischen 0,00 und 9999,99

▶ Warenkorb

Attribut	Gültigkeitsbereich
Kundennummer	Zahl zwischen 1 und 99999
Artikelnummer	Zahl zwischen 1 und 999999
Anzahl	Zahl zwischen 0 und 99999
Gesamtpreis	Zahl zwischen 0,00 und 999999,99
Hinzugefügt am	Datum

▶ Bestellungen

Attribut	Gültigkeitsbereich
Bestelldatum	Datum
Status	»offen«, »in Bearbeitung«, »versendet«
Gesamtpreis	Zahl zwischen 0,00 und 99999,99
Bemerkung	Bezeichnung mit bis zu 255 Zeichen

▶ Bestellte Produkte

Attribut	Gültigkeitsbereich
Kundennummer	Zahl zwischen 1 und 99999
Artikelnummer	Zahl zwischen 1 und 999999
Anzahl	Zahl zwischen 0 und 99999
Preis	Zahl zwischen 0,00 und 999999,99
Umsatzsteuer	Prozentzahl
Bestellt am	Datum

Tabelle 9.1 Aufstellung der Attribute einschließlich ihrer Gültigkeitsbereiche

9.3.2 Schritt 4: ER-Modell erstellen

Beziehungen ermitteln

Im nächsten Schritt werden die Beziehungen zwischen den Entitäten ermittelt, und anschließend wird mit dem ER-Modell (Entity-

Relationship-Model) die grundlegende Beziehungsstruktur der Datenbank entworfen. Diese Beziehungen sind Abhängigkeiten zwischen den Entitäten und erlauben es, die gegliederten Daten wieder passend zusammenzufügen.

Die Abhängigkeiten können dabei nicht automatisch erzeugt werden, sondern müssen manuell festgestellt werden. Ein gutes Datenmodell stellen Sie dann auf, wenn Sie alle Arbeitsschritte und Inhalte begründen können.

Für unser Beispiel sehen die Beziehungen einschließlich ihrer Beziehungstypen (1:n, 1:1, n:m) wie folgt aus:

- Kunde erstellt Warenauswahl (Warenkorb, 1:1)
- Kunde nimmt Bestellungen vor (1:n)
- Eine Bestellung besteht aus (bestellten) Produkten (1:n)
- Bestellte Produkte werden aus Warenkorb übernommen (1:1)
- Warenkorb besteht aus Produkten (1:n)
- Hersteller stellt Produkt her (1:n)

Die Beziehungen sehen in der grafischen Darstellung wie folgt aus:

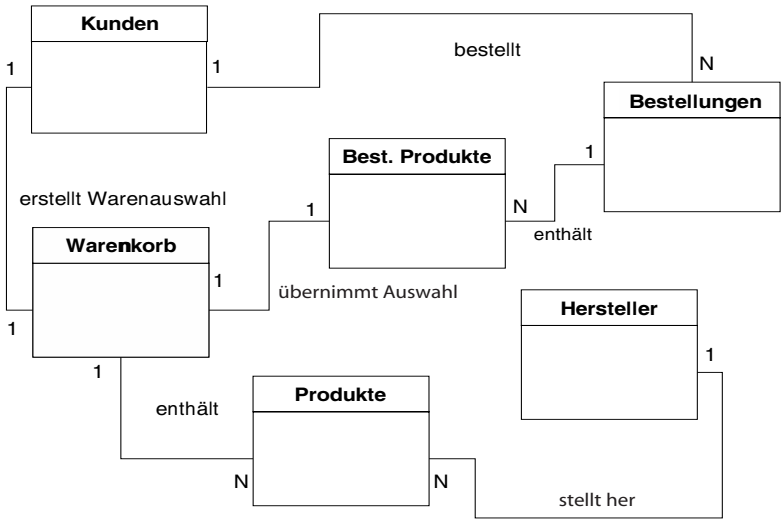

Abbildung 9.2 Darstellung der Beziehungen (ohne Attribute)

9.3.3 Schritt 5: Relationales Datenmodell erstellen

Nachdem die Basisarbeit der Modellierung durchgeführt ist, kann im nächsten Schritt die Überführung in das relationale Datenbankmodell erfolgen. Damit erfolgt die Umsetzung des konzeptionellen Schemas in ein internes Schema zur Datenspeicherung.

Nach dieser Vorarbeit ist die Umsetzung dann auch relativ einfach. Aus den Entitätstypen werden Tabellen und aus den Attributen die Spalten der Tabellen. Da wir oben bereits einen Gültigkeitsbereich für die Attribute festgelegt haben, kann daraus der am besten geeignete Datentyp abgeleitet werden.

Was für die Aufstellung des relationalen Datenmodells noch fehlt, ist die Darstellung der Beziehungen. Diese werden im relationalen Datenbankmodell über die Schlüssel hergestellt. Auch hier ist die Ableitung einfach. Um Beziehungen über Werte herzustellen, benötigen wir einen Wert, der jeden Datensatz eindeutig bestimmt, den Primärschlüssel. Ein Primärschlüssel kann aus einem oder mehreren Feldern bestehen. Die Handhabung von Primärschlüsseln, die nur aus einem Feld bestehen, ist in der Praxis wesentlich einfacher. Aus diesem Grund werden im Allgemeinen Tabellen mit einer fortlaufenden Nummer, der als Primärschlüssel dient, versehen. Mit `AUTO_INCREMENT` wird diese fortlaufende Nummer automatisch erzeugt.

Anschließend werden die Beziehungen zwischen Tabellen über Fremdschlüssel definiert. Prinzip ist, dass in einer Detailtabelle (dem n bei 1:n-Beziehungstypen) ein zusätzliches Feld eingeführt wird, welches auf den Primärschlüssel einer Mastertabelle verweist.

Auch an dieser Stelle kann die Vorarbeit des ER-Modells also nahtlos in die Umsetzung des relationalen Datenmodells erfolgen. Dort haben wir die Beziehungen benannt, im Datenmodell wird bei einer solchen Beziehung dann ein zusätzliches Feld in der Detailtabelle eingefügt.

9.3.4 Schritt 6: Datenmodell optimieren (Normalisierung)

In Abschnitt 4.2.6, »Optimierung des Datenmodells (Normalisierung)«, wurde dargestellt, wie Tabellen über die so genannte Normalisierung optimiert werden können. Hier nur kurz zur Erinnerung die verschiedenen Normalformen. In der 1. Normalform liegen alle Attribute einer Tabelle in atomarer Form vor. Bei der zweiten Normalform werden die Werte aller Nicht-Schlüsselattribute ausschließlich vom Primärschlüssel bestimmt (und die Relation befindet sich in der ersten Normalform). In der dritten Normalform bestehen keine Abhängigkeiten von Nicht-Schlüssel-Attributen (keine inneren Abhängigkeiten). Die Relation muss sich in der zweiten Normalform befinden.

Bei der Betrachtung unseres obigen ER-Modells kann festgehalten werden, dass die erste und zweite Normalform erfüllt sind. Durch die Einführung einer eindeutigen, fortlaufenden ID in den Tabellen sind diese automatisch in der zweiten Normalform.

9.4 Benutzerschnittstellen

9.4.1 Schritt 7: Softwarekomponenten definieren

Die definierten Anforderungen sowie das relationale Datenmodell sind zum jetzigen Zeitpunkt noch universell und nicht an eine bestimmte Software gebunden.

An diesem Punkt fällt dann die Entscheidung, mit welcher Software die Anwendung realisiert wird. Ein solcher Entscheidungsprozess kann durchaus aufwendig sein. In die Entscheidung sind Kostenkalkulationen, Personalfragen (stehen ausreichend Entwickler mit dem notwendigen Know-how zur Verfügung) und technologische Gesichtspunkte (verfügt die Entwicklungssoftware über alle benötigten Merkmale) einzubeziehen. Diese Entscheidung hat dann fundamentalen Charakter, weil sie bis zur endgültigen Realisierung der Anwendung gelten sollte. Alle nachträglichen Änderungen führen in der Regel zu Zeitverzögerungen und erhöhtem Aufwand in der Gesamtrealisierung.

Da dieses Kapitel beispielhaft den Ablauf bis zu einer fertigen Anwendung zeigen soll, kürzen wir den Entscheidungsprozess hier ab. Da dies ein MySQL-Buch ist, steht die Wahl des Datenbankservers fest. Aus den Anforderungen ergibt sich weiterhin, dass als Internetapplikation ein Webserver die Basistechnologie sein sollte. Für unsere Zwecke nehmen wir hier Apache, den zurzeit am weitesten verbreiteten Webserver. Über den Webserver können dann alle HTML-Seiten an den Benutzer ausgeliefert werden.

Bleibt noch die Wahl der Programmiersprache, um die Funktionen sowie die Benutzerschnittstellen zu realisieren. Als Internetapplikation für Massenanwendungen sollte die Programmiersprache Internetoperationen (z. B. E-Mail-Unterstützung), MySQL-Unterstützung sowie eine enge Anbindung an den Webserver aufweisen. Eine gute Wahl ist hier PHP.

In der Zusammenfassung sieht die Softwaretechnologie für unsere Beispielanwendung wie folgt aus:

Software	Aufgabe
MySQL	Speichern aller relevanten Anwendungsdaten
Apache	Webserver. Bedienung von Anfragen über das Internet
PHP	Programmiersprache für anwendungsspezifische Funktionen
HTML	Seitenbeschreibungsprache für die Benutzerschnittstellen

Tabelle 9.2 Softwaretechnologie für das Beispiel

Unsere Beispielanwendung läuft also auf einem typischen LAMP-System.

9.4.2 Schritt 8: Benutzerschnittstellen (Dialoge) entwerfen

Wie bereits erläutert, lassen sich PHP und HTML sehr leicht kombinieren. Für unsere Anwendung ist das auch in Bezug auf die Entwicklungsarbeit ideal. Die grafische Benutzerfläche wird per HTML definiert, und alle notwendigen Funktionen können über PHP eingebaut werden.

Dies kommt auch einem Projektablauf entgegen. Sinnvoll ist es in der Regel, einen Prototypen der Anwendung zu erzeugen, um mit diesem die Benutzerführung und die Funktionalitäten überprüfen zu können. Durch einen Prototypen können auch noch Schwachstellen im Konzept aufgedeckt werden.

Die Kombination HTML und PHP erlaubt sehr einfaches Prototyping. Die Programmoberfläche kann mit HTML-Elementen relativ schnell erstellt werden. Hierfür ist es sinnvoll, ein Programm zur Webseitengestaltung wie Dreamweaver, HomeSite oder GoLive zu verwenden.

Bevor man die Dialoge gestaltet, sollte man sich zur besseren Planung am besten eine Liste aller benötigten Dialoge erstellen. Für den Besucher müssen folgende Dialoge erstellt werden:

Dialoge für Online-Shop-Benutzer

- **Anmeldung**. Name und Login-Daten bzw. Neuanmeldung mit Konto einrichten
- **Online-Shop**. Der Hauptdialog für den Benutzer, um Produkte zu suchen und auszuwählen
- **Produktdetails**. Detailinformationen zu einem Produkt anzeigen
- **Warenkorb**. Anzeige aller ausgewählten Produkte. Benutzer hat die Möglichkeit, die Mengen zu ändern bzw. Produkte aus dem Warenkorb zu löschen (siehe Abbildung 9.3)
- **Kasse**. Endgültige Bestätigung der Bestellung und nochmalige Überprüfung der Kontaktdaten
- **Kundenkonto**. Anzeige aller Bestellungen und ihres Status (offen, in Bearbeitung, versendet).

Für die Verwaltung benötigen wir diese Dialoge:

- **Produktverwaltung**. Verwaltung des Produktkatalogs mit Anlegen, Aktualisieren und Löschen der Produkte
- **Bestellverwaltung**. Anzeige und Bearbeitung der Bestellungen
- **Kundenverwaltung**. Anzeige der Kunden einschließlich der Kontakt- und Bestellinformationen (siehe Abbildung 9.4)

Abbildung 9.3 Prototyping der Benutzerdialoge mit HTML

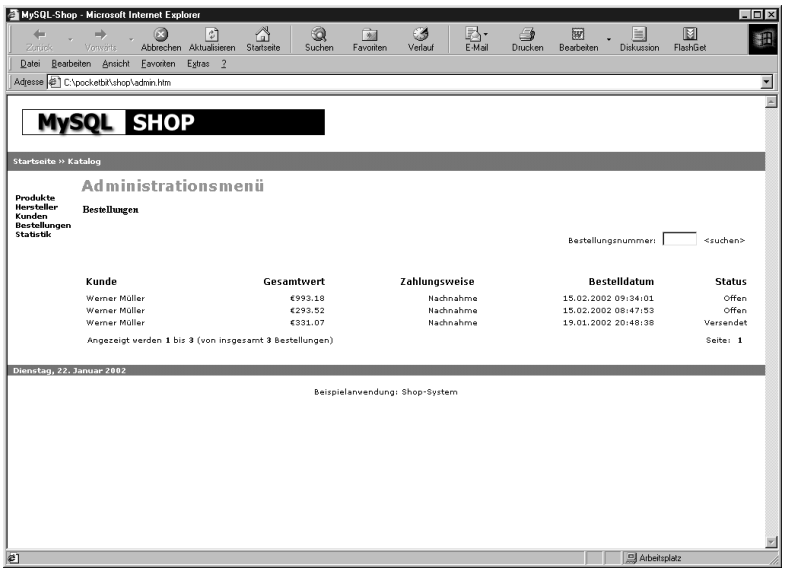

Abbildung 9.4 Das Administrationsmenü der Beispielanwendung

Für die Erstellung der Benutzerschnittstellen sind wie oben erwähnt HTML-Kenntnisse notwendig, die hier als gegeben vorausgesetzt werden. Eine umfassende Einleitung zur Erstellung von HTML findet sich beispielsweise im Internet mit SELFHTML (*selfhtml.teamone.de*).

9.5 Implementierung

9.5.1 Schritt 9: Datenbank und Tabellen anlegen

Nachdem die Anforderungen definiert sind, das Datenbankmodell erarbeitet und die Oberfläche gestaltet ist, kann im nächsten Schritt die vollständige Implementierung der Anwendung beginnen.

Hierfür müssen Sie jetzt auch endgültig die Datenbank und Tabellen in MySQL anlegen.

Zuerst muss eine Datenbank angelegt werden:

```
mysql>CREATE DATABASE shop;
```

Anschließend werden die einzelnen Tabellen angelegt. Sie können die folgenden Tabellendefinitionen per Hand eingeben.

CD: shop.sql

Auf der beiliegenden Buch-CD sind alle Tabellendefinitionen in der Datei **shop.sql** gespeichert. Sie können damit die Tabellendefinitionen auch über

```
$>mysql -ubenutzername -ppasswort < shop.sql
```

in Ihre Datenbank einlesen.

Die Tabellendefinitionen sind im Folgenden aufgelistet. In den Tabellendefinitionen sind auch die Fremdschlüssel (`FOREIGN KEY`) definiert, die allerdings nicht von den MyISAM-Tabellentypen nicht unterstützt werden. Daher wurde hier der Tabellentyp InnoDB gewählt. Falls Sie eine MySQL-Installation ohne InnoDB besitzen, lassen Sie einfach in den unten folgenden Tabellendefinitionen den Eintrag `[TYPE=INNODB]` weg. In den Tabellen wurden Indices für Felder angelegt, in denen später häufig gesucht wird. Dies sind z. B. die Produktbezeichnung oder der Anmeldename.

Ob diese Indices vollständig ausreichend sind, kann unter Umständen erst im Betrieb mit vielen Datensätzen abschließend bewertet werden. Ergänzend sei an dieser Stelle darauf hingewiesen, dass bei InnoDB-Tabellen für `FOREIGN KEY` Indices auf die Primärschlüssel benötigt werden.

Tabellendefinitionen

```
CREATE TABLE produkte
(
ID INT AUTO_INCREMENT PRIMARY KEY,
bezeichnung               VARCHAR(60),
beschreibung              TEXT
kategorie                 VARCHAR(60),
preis                     DEC(7,2),
umsatzsteuer              INT,
bilddatei                 VARCHAR(60),
lagermenge                INT,
hinzugefügt               DATETIME,
zuletzt_geaendert         DATETIME,
gewicht_kg                DEC(6,2),
hersteller_id INT,
INDEX bezeichnung_ind(bezeichnung)
) TYPE=INNODB;

CREATE TABLE hersteller
(
ID INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(60)
) TYPE=INNODB;

CREATE TABLE kunden
(
ID INT AUTO_INCREMENT PRIMARY KEY,
anrede                    VARCHAR(6),
name1                     VARCHAR(60),
name2                     VARCHAR(60),
plz                       VARCHAR(14),
ort                       VARCHAR(46),
strasse                   VARCHAR(46),
hausnummer                VARCHAR(6),
land                      VARCHAR(30),
```

```
email                       VARCHAR(50),
telefonnummer               VARCHAR(60),
faxnummer                   VARCHAR(60),
anmeldename                 VARCHAR(10),UNIQUE,
passwort                    VARCHAR(60),
INDEX name1_ind (name1),
INDEX name2_ind (name2),
INDEX ort_ind (ort),
INDEX anmeldename_ind (anmeldename)
) TYPE=INNODB;

CREATE TABLE warenkorb
(
ID INT AUTO_INCREMENT PRIMARY KEY,
kundennr INT,
artikelnr INT,
anzahl INT,
gesamtpreis                 DEC(10,2),
hinzugefuegt_am             DATETIME,
produkte_id INT,
INDEX produkte_ind (produkte_id),
FOREIGN KEY (produkte_id) REFERENCES produkte(ID)
) TYPE=INNODB;

CREATE TABLE bestellungen
(
ID INT AUTO_INCREMENT PRIMARY KEY,
kunden_id INT,
bestelldatum DATETIME,
status VARCHAR(50),
gesamtpreis DEC(10,2),
bemerkung BLOB
INDEX kundenid_ind (kunden_id),
FOREIGN KEY (kunden_id) REFERENCES kunden(ID)
) TYPE=INNODB;

CREATE TABLE bestellte_produkte
(
ID INT AUTO_INCREMENT PRIMARY KEY,
```

```
kunden_id INT,
produkte_id INT,
bestellungen_id INT,
anzahl INT,
preis DEC(10,2),
ust INT,
bestellam DATETIME,
INDEX produktid_ind (produkte_id),
INDEX kundenid_ind (kunden_id),
INDEX bestellungenid_ind (bestellungen_id),
FOREIGN KEY (produkte_id) REFERENCES produkte(ID),
FOREIGN KEY (kunden_id) REFERENCES kunden(ID),
FOREIGN KEY (bestellungen_id)
REFERENCES bestellungen(ID)
) TYPE=INNODB;
```

9.5.2 Schritt 10: PHP-Funktionen definieren

Im ersten Schritt haben wir bereits die grundsätzlichen Anwendungsanforderungen definiert, die der Shop aufweisen soll. Diese Anwendungsanforderungen sind jetzt in PHP zu realisieren.

Bei umfangreichen Projekten stellt sich für Sie natürlich auch die Frage einer effektiven Anwendungsentwicklung. Zu den wesentlichen Prinzipien der Anwendungsentwicklung gehören:

- die einfache Wartbarkeit von Programmcode
- die Wiederverwendbarkeit von Programmcode

Diese Prinzipien werden für das Beispiel wie folgt berücksichtigt:

- Oberfläche und Programmfunktionen bzw. -methoden werden getrennt. Die PHP-Funktionen bzw. -Methoden werden in der eigenen Datei **functions.php** gespeichert. Die Oberfläche wird jeweils über HTML erzeugt. Programmfunktionen und Benutzerschnittstelle werden so übersichtlich und damit besser wartbar.
- Unser Beispiel nutzt die objektorientierten Möglichkeiten von PHP.

Die vollständige Erläuterung aller Funktionen und Methoden unseres Beispiels würde den Rahmen dieses Buches sprengen. Ich möchte deshalb hier die wichtigsten Funktionen, insbesondere im Hinblick auf die Anwendung von MySQL, besprechen.

CD: shop.zip

Der vollständige Quellcode der Anwendung liegt der Buch-CD bei.

Artikel suchen

Um einen Artikel zu suchen, wird eine Suchanfrage an die MySQL-Datenbank gesendet und das Suchergebnis auf der Hauptseite ausgegeben. Im Bereich von Internetanwendungen haben sich insbesondere »Ein Feld«-Suchdialoge durchgesetzt. Für den Benutzer ist dies wesentlich einfacher, als die Auswahl aus verschiedenen Optionen. Die Suchanfrage an die MySQL-Datenbank muss in diesem Fall also so aufgebaut werden, dass möglichst alle Benutzereingaben gefunden werden. Ein Benutzer wird Produkte in der Regel nach Produktbezeichnung, nach Eigenschaften des Produkts oder nach dem Hersteller suchen.

Unscharfe Suche mit LIKE

Wir müssen also unsere Suchanfrage so formulieren, dass diese Fälle berücksichtigt werden. Die Suche wird deshalb mit `LIKE` über die Felder `bezeichnung` und `beschreibung` der Produkttabelle sowie den Namen des Herstellers formuliert.

Die Abfrage sieht als PHP-Funktion wie folgt aus:

```
function search_shop($searchstr, $p, $limit=6)
{
[...]
$sql="SELECT p.bezeichnung, p.preis+(preis*
(umsatzsteuer/100)),p.artikelnummer";
$sql.=" FROM produkte AS p LEFT JOIN hersteller ON
(p.hersteller = hersteller.ID)";
$sql.=" WHERE p.bezeichnung LIKE '%".$searchstr."%'";
$sql.=" OR p.beschreibung LIKE '%".$searchstr."%'";
break;
[...]
$this->res = mysql_query($sql,$this->hDB);
return $this->res;
}
```

Die Funktion `search_shop()` wurde hierbei so angelegt, dass verschiedene Abfragen damit ausgeführt werden können. Die Variable `$p` dient hier zur Unterscheidung der Abfragen.

Mit der Funktion `mysql_query($sql,$this->hDB)` wird der SQL-Befehl an die Datenbank gesendet und als Resultset in der Variablen `$res` gespeichert (hier als `$this->res` aufgrund der objektorientierten Programmierung).

Der Inhalt des Resultset wird dann in der Funktion `show_list()` als Tabellenausgabe mit HTML aufbereitet. Die Variable `$res`, die das Resultset der Abfrage beinhaltet, steht als Klassenvariable jetzt auch dieser Funktion zur Verfügung.

```
function show_list()
{
echo "<table border=0 cellpadding=2 cellspacing=0
width=\"100\"><tbody><tr><td height=\"90\">
<table border=0 cellpadding=2 cellspacing=0 width=\
"100%\">
<tbody><tr ><td> Bezeichnung </td>
<td align=right> Einzelpreis </td></tr>";
while ($row = mysql_fetch_array($this->res))
{
echo " <tr >
<td> <BR><a href=p_info.php?productsid=".
$row[2].">".$row[0]."</a> </td><td
align=right> &#8364;".$row[1]." <BR></td></
tr>";
}
echo "</tbody></table></table><BR><BR><BR>";
}
```

In der HTML-Seite können beide Funktionen dann über folgenden Befehl eingebunden werden.

```
$shop = new CShop($conn);
$shop->search_shop($searchstr,1);
$shop->show_list();
```

Registrieren

Um Produkte in unserem Online-Shop kaufen zu können, muss sich der Benutzer zunächst registrieren und seine persönlichen Daten mitteilen. Diese Informationen werden in der Tabelle `kunden` gespeichert. Die hier gespeicherten Informationen dienen zum einen bei Bestellungen als Versandinformationen, zum anderen werden hier auch die Login-Informationen wie Anmeldename und Passwort gespeichert.

Die Funktion nimmt dabei die Variablen aus dem HTML-Formular auf und erzeugt daraus einen gültigen SQL-String, der anschließend benutzt wird, um die Information in der Datenbank zu speichern. Überprüft wird in dieser Funktion auch, ob die Angaben vollständig sind. Bei unvollständiger Angabe wird die Variable `$ro=0` zurückgegeben. Über die Variable kann dann gesteuert werden, ob das Formular noch einmal angezeigt werden soll.

```
function reg()
{
if ($GLOBALS["password"] <> $GLOBALS["conf"]
|| strlen($GLOBALS["password"]) < 4
|| strlen($GLOBALS["lastname"]) < 2
|| strlen($GLOBALS["plz"]) < 5
[...]
{
echo "Angaben unvollständig. Bitte ergänzen<BR><BR>";
$ro = 0;
}
else
{
$sql = "INSERT INTO kunden ";
$sql .= "(anrede,name1,name2,email,strasse,plz,";
$sql .= "ort,land,telefonnummer,faxnummer,passwort)";
$sql .= "VALUES ('";
$sql .= $GLOBALS["gender"]."','";
$sql .= $GLOBALS["firstname"]."','";
$sql .= $GLOBALS["lastname"]."','";
$sql .= $GLOBALS["email"]."','";
$sql .= $GLOBALS["street"]."','";
$sql .= $GLOBALS["postcode"]."','";
$sql .= $GLOBALS["city"]."','";
```

```
$sql .= $GLOBALS["country"]."','";
$sql .= $GLOBALS["tel"]."','";
$sql .= $GLOBALS["fax"]."','";
$sql .= $GLOBALS["password"]."'";
$sql .= ")";
mysql_query($sql,$this->hDB);
$ro=1;
}
return $ro;
}
```

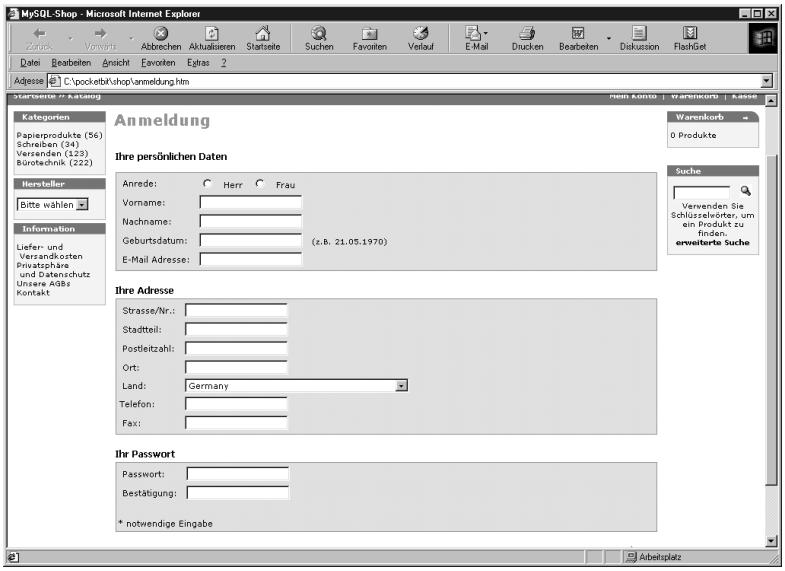

Abbildung 9.5 Der Anmeldedialog unseres Beispielshops

Kundenkonto anzeigen

Um das Kundenkonto einzusehen, muss der Benutzer angemeldet sein. Auf Basis seiner dann in den Sessionvariablen gespeicherten Kundennummer können alle Einträge in der Tabelle `bestellungen` selektiert werden. Aufgelistet werden dann alle Bestellungen, deren Detailtabelle über einen Link auf die Bestellinformation abgerufen werden können.

Auch hier wird das gleiche Prinzip verwendet. Die Funktion wird in der **functions.php** definiert und kann dann relativ einfach in die Webseite eingebunden werden.

```
function show_orders($customerid)
{
$sql = "SELECT * FROM bestellungen
    WHERE kunden_id = ".$customerid;
$this->res = mysql_query($sql,$this->hDB);
echo "<table border=0 cellpadding=2 cellspacing=0
width="100%">
  <tbody>
  <tr>
    <td>Auftragsnummer</td>
    <td>Bestelldatum</td>
    <td>Preis</td>
    <td>Bestellstatus</td>
  </tr>";
while ($row = mysql_fetch_array($this->res))
       {
       echo "<tr >
    <td align=middle>".$row[0]."</td>
    <td ><a href=account_history_info.php?order_id=
         ".$row[0].">".$row[1]."</a></td>
    <td align=right>&#8364;".$row[3]."</td>
    <td align=right >".$row[2]."</td></tr>";
   }
echo  "</tbody></table>";
}
```

Die Einbindung der Funktion in die HTML-Seite sieht dabei wie folgt aus:

```
$shop = new CShop($conn);
$shop->show_orders(1);
```

9.5.3 Schritt 11: Benutzerschnittstelle und Funktionen integrieren

Im nächsten Schritt werden jetzt die in PHP definierten Funktionen mit der in HTML erstellen Benutzeroberfläche (siehe Schritt 8) verknüpft. Zu diesem Zweck fügen wir in die HTML-Datei die

jeweils benötigten PHP-Funktionen ein. In Abschnitt 7.4.2., »PHP«, wurde die grundsätzliche Verknüpfung von HTML mit PHP bereits erläutert.

Wie bereits oben gezeigt können die Funktionen direkt in den HTML-Code eingebaut werden. Um z. B. eine Funktion mit Zugriff auf die MySQL-Datenbank zu realisieren, werden jetzt nur noch wenige Zeilen Code benötigt.

```
<?php
include 'functions.php';
$conn = mysql_connect('localhost','usser','passwort');
mysql_select_db('mysqlshop',$conn);
$shop = new CShop($conn);
$shop->search_shop("Suchwort",2,6);
?>
```

Variablen aus dem HTML-Formular (z. B. die Eingabe im Suchen-Dialog) werden direkt an PHP übergeben und können dann für die definierten Funktionen benutzt werden. In diesem Beispiel soll dieses anhand des Suchen-Dialogs noch einmal gezeigt werden.

Der Codeteil für den Suchen-Dialog hat folgendes Aussehen:

```
<FORM action=search_result.php4 method=get
      name=quick_find>
<INPUT maxLength=30 name=searchstr size=10> 
<INPUT border=0 src="images/button_quick_find.gif"
type=image><BR> Verwenden Sie Schlüsselwörter, um ein
    Produkt zu finden.<BR></FORM>
```

Der Benutzer gibt also in ein Textfeld mit der Variablen `$searchstr` einen Suchbegriff ein, und beim Klicken auf die Suchlupe wird das Suchformular über `action=search_result.php` aufgerufen. Dieses Formular verfügt beim Laden über die Variable `$searchstr`, die über `GET` übergeben wird. Die anschließende Aufgabe besteht darin, aus dieser Variablen die vollständige Suchanfrage zu generieren und das Ergebnis in formatierter Form an das Formular zurückzugeben. Um diese Aufgabe zu bewerkstelligen, muss jetzt die Suchfunktion `search_shop()` mit der Variablen `searchstr` aufgerufen werden.

```
$shop->search_shop($searchstr,1);
```

Das Ergebnis wird dann in einer Schleife so aufbereitet, dass die Suchergebnisse angezeigt werden.

Analog zu diesen Beispielen werden alle Benutzerfunktionen implementiert.

9.5.4 Schritt 12: Anwendung testen und weitergehende Fragen

Im letzten Schritt sollte dann die Anwendung intensiv getestet werden. In der Regel werden sich in der Anwendung noch kleinere oder größere Fehler befinden. Die abschließende Aufgabe besteht also darin, die gesamte Anwendung auf Fehler zu testen. Außer der Gesamtfunktionalität interessiert im Rahmen dieses Buches insbesondere die Frage, ob die MySQL-Funktionen richtig funktionieren. Deshalb sollte nicht nur überprüft werden, ob das Programm richtig arbeitet, sondern auch, ob die Datenbankeingaben während der Programmbenutzung korrekt erfolgen und vollständig sind. Am besten macht man dies, indem man die Speicherungsvorgänge in der Datenbank betrachtet, also direkt in Tabellen die Datenänderung mitverfolgt.

Weitere Fragestellungen, die während der Testphase beantwortet werden sollten, können sein:

- Sind die richtigen Datenbankfelder indiziert? Die korrekte Indizierung der Datenbank kann unter Umständen erst mit umfangreichen Datenbeständen endgültig geprüft werden. Wer die Datenbank mit vielen Datensätzen testen möchte, kann z. B. skriptgesteuert (siehe Abschnitt 8.7, »Datenbanktests durchführen«) Daten erzeugen.
- Ist die referenzielle Integrität der Datenbank in Ordnung? Wird also beispielsweise beim Löschen eines Produktes überprüft, ob hierzu keine Bestellungen vorliegen
- Ist die Datenbank für die zu erwartenden Zugriffszahlen konfiguriert (`max_connections`)?
- Sind Backup-Verfahren für die Datenbank notwendig und funktionieren diese?
- Welche Größe wird die Datenbank erreichen? Ist der Festplattenplatz ausreichend dimensioniert?