**Building Relational Tables**

Since MySQL is a *relational* database, it has the capability of relating two or more tables together using links called *constraints*.

A constraint is a rule placed on a data field that restricts the data you're allowed to place in it. You can create a constraint that only allows you to enter positive values in a numeric data field or only lowercase letters in a character data field. A relational database utilizes constraints to allow only data contained in one table to be entered into a data field in another table, thus creating the relationship.

For example, the *catid* data field in the products table is directly related to the *catid* data field in the categories table. You shouldn't be able to enter a product record with a catid value that doesn't appear in the categories table.

The relational database server handles all of the data relationships for you automatically. This allows you to know your data will always be in sync. The problem is, you need to tell MySQL about these relationships.

If you create just individual tables in MySQL, no relationships exist between the tables. Then MySQL will let you enter any data you want into the products table even if there isn't a corresponding catid value.

We need to tell MySQL that the catid data field in the products table is related to the catid data field in the categories table. You do this by using a *foreign key* constraint.

When you define a data field as a foreign key, MySQL establishes an internal constraint between that data field and the data field referenced in another (foreign) table. The data field in the other table must be the primary key for that table; so only one record can be related to the foreign key value.

|  |
| --- |
| **Note:**Foreign keys are only available if you use the InnoDB database engine for your tables. The MyISAM table type doesn't support foreign keys. |

Let's take a look at our database design layout and see how the table data fields are related.

The Food Store database layout

I've designed the layout for each of the tables to easily accommodate foreign keys. Notice that the products table relates to the categories table by incorporating the catid primary key value from the categories table as a data field. The order\_items table uses the prodid primary key from the products table to uniquely identify products purchased.

Similarly, the orders table relates to the customers table using the custid primary key data field to uniquely identify the customer who placed the order. The order\_items table relates to both the orders and products tables by using both the orderid and prodid primary key data fields.

Now that we've established the design of our database, you'll need to create the tables and relate the foreign key data fields. Unfortunately, this can be problematic.

**Creating Tables With Foreign Keys**

Sadly, our trusty friend the phpMyAdmin graphical database tool can help us out only so far. It lacks the ability to implement the more exotic database features. Unfortunately, phpMyAdmin doesn't allow you to define foreign keys when you create a table using the graphical interface. For that, you'll need to perform some manual interventions.

To create the tables for our project, you'll need to use the MySQL Console feature in WampServer. The MySQL Console gives you a command prompt interface, which allows you to enter SQL commands directly to the MySQL server. To get to the MySQL Console, follow these steps:

1. Start the WampServer services by clicking **Start** > **Programs** > **WampServer** > **start WampServer**.
2. Click the **WampServer** icon in the system tray.
3. Select **MySQL** > **MySQL Console**.

The MySQL Console window appears, asking you for the root password. The default root password in WampServer is empty; so just press the ENTER key. Next you're greeted by a short message and then the MySQL prompt.

The MySQL Console window

At the prompt, you must first connect to a database using the *connect* statement. Once connected to a database, you can enter any SQL statement for the database. The MySQL server processes the SQL and returns the results in the Console window.

For this application, you'll need to create a new database, connect to it, and then create the tables. Follow these steps:

1. Enter the following SQL statement to create a database called *store*:

CREATE DATABASE store DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8\_unicode\_ci;

|  |
| --- |
| **Note:** The semicolon at the end of the SQL statement is important. It tells MySQL that your statement is ready for processing. You can use that to your advantage by spreading out a long SQL statement over several lines. Just don't use the semicolon at the end of a line. If you do, MySQL will provide a secondary prompt for you to enter more text for the SQL statement. |

This creates the store database and sets the default collation using the Unicode case-insensitive sorting method (utf8\_unicode\_ci), which uses the multi-lingual UTF8 character set with case-insensitive sorting. You should get a message back indicating that the SQL statement completed successfully.

1. Enter the connect command to connect to the new database:

connect store;

1. Enter the SQL CREATE TABLE statement to create the categories table:
2. CREATE TABLE categories(catid int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name varchar(30) NOT NULL) ENGINE=INNODB;

This creates the categories table using the InnoDB database engine.

1. Enter the SQL CREATE TABLE statement to create the products table:
2. CREATE TABLE products (prodid int NOT NULL AUTO\_INCREMENT PRIMARY KEY,
3. catid int NOT NULL,
4. description varchar(100) NOT NULL,
5. picture mediumblob,
6. price decimal(6,2) NOT NULL,
7. quantity int NOT NULL,
8. onsale BOOL,
9. FOREIGN KEY (catid) REFERENCES categories(catid))

ENGINE=INNODB;

This creates the products table using the InnoDB database engine. We specified the foreign key constraint in the CREATE TABLE SQL statement by using the FOREIGN KEY clause. This matches the catid data field in this table to the catid data field in the categories table.

1. To exit the MySQL Console, type *exit* at the command prompt.

Congratulations, you have now manually created a database and two tables from the MySQL Console! Now, let's walk through some SQL testing to see if the foreign key constraint really works.

1. Start the MySQL Console by clicking the WampServer icon in the system tray, then selecting **MySQL** > **MySQL Console**.
2. Connect to the store database:

connect store;

1. Use the INSERT SQL statement to insert a new record into the products table without creating a category in the categories table:

INSERT INTO products (catid, description, price, quantity) VALUES (1, "test", 6.25, 10);

1. Examine the message that MySQL returns:

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint  
fails (`store2/products`, CONSTRAINT `products\_ibfk\_1` FOREIGN KEY (`catid`) REFERENCES `categories` (`catid`))

The MySQL database automatically rejected our attempt to insert a new products record without a corresponding catid in the categories table!

You'll be creating the rest of the tables in this lesson's assignment. For now, let's move on to Chapter 4 and look at some fancy queries.