24.3 A books Database

We introduce relational databases in the context of this chapter's `books` database, which you'll use in several examples. Before we discuss SQL, we discuss the *tables* of the `books` database. We use this database to introduce various database concepts, including how to use SQL to obtain information from the database and to manipulate the data. We provide a script to create the database. You can find the script in the examples directory for this chapter. Section 24.5 explains how to use this script.

`Authors` Table

The database consists of three tables: `Authors`, `AuthorISBN` and `Titles`. The `Authors` table (described in Fig. 24.3) consists of three columns that maintain each author's unique ID number, first name and last name. Figure 24.4 contains sample data from the `Authors` table.

Column	Description
`AuthorID`	Author's ID number in the database. In the `books` database, this integer column is de as **autoincremented**—for each row inserted in this table, the `AuthorID` value is inc by 1 automatically to ensure that each row has a unique `AuthorID`. This column represents the table's primary key. Autoincremented columns are so-called identity columns. The SQL script we provide for this database uses the SQL `IDENTITY` keyv to mark the `AuthorID` column as an identity column. For more information on usin `IDENTITY` keyword and creating databases, see the Java DB Developer's Guide at

	`http://docs.oracle.com/javadb/10.10.1.2/devguide/derbydev`
`FirstName`	Author’s first name (a string).
`LastName`	Author’s last name (a string).

Fig. 24.3

`Authors` table from the `books` database.

`AuthorID`	`FirstName`	`LastName`
`1`	`Paul`	`Deitel`
`2`	`Harvey`	`Deitel`
`3`	`Abbey`	`Deitel`
`4`	`Dan`	`Quirk`
`5`	`Michael`	`Morgano`

Fig. 24.4

Sample data from the `Authors` table.

`Titles` Table

The `Titles` table described in Fig. 24.5 consists of four columns that maintain information about each book in the database, including its ISBN, title, edition number and copyright year. Figure 24.6 contains the data from the `Titles` table.

Column	Description
`ISBN`	ISBN of the book (a string). The table's primary key. ISBN is an abbreviation for "International Standard Book Number"—a numbering scheme that publishers use to give every book a unique identification number.
`Title`	Title of the book (a string).
`EditionNumber`	Edition number of the book (an integer).
`Copyright`	Copyright year of the book (a string).

Fig. 24.5

`Titles` table from the `books` database.

ISBN	Title	EditionNumber	Copyright
0132151006	Internet & World Wide Web How to Program	5	2012
0133807800	Java How to Program	10	2015
0132575655	Java How to Program, Late Objects Version	10	2015
013299044X	C How to Program	7	2013
0132990601	Simply Visual	4	2013

	Basic 2010		
0133406954	Visual Basic 2012 How to Program	6	2014
0133379337	Visual C# 2012 How to Program	5	2014
0136151574	Visual C++ 2008 How to Program	2	2008
0133378713	C++ How to Program	9	2014
0133570924	Android How to Program	2	2015
0133570924	Android for Programmers: An App-Driven Approach, Volume 1	2	2014
0132121360	Android for Programmers: An App-Driven Approach	1	2012

Fig. 24.6

Sample data from the `Titles` table of the `books` database .

`AuthorISBN` Table

The `AuthorISBN` table (described in Fig. 24.7) consists of two columns that maintain ISBNs for each book and their corresponding authors' ID numbers. This table associates authors with their books. The `AuthorID` column is a **foreign key**—a column in this table that matches the primary-key column in another table (that is, `AuthorID` in the `Authors` table). The `ISBN` column is also a foreign key—it matches the primary-key column (that is, `ISBN`) in the `Titles` table. A database might consist of many tables. A goal when designing a database is to *minimize* the amount of *duplicated* data among the database's tables. Foreign keys, which are specified when a database table is created in the database, link the data in *multiple* tables. Together the `AuthorID` and `ISBN` columns in this table form a *composite primary key*. Every row in this table *uniquely* matches *one* author to *one* book's ISBN. Figure 24.8 contains the data from the `AuthorISBN` table of the `books` database. [*Note:* To save space, we split the table into two columns, each containing the `AuthorID` and `ISBN` columns.]

Column	**Description**
`AuthorID`	The author's ID number, a foreign key to the `Authors` table.
`ISBN`	The ISBN for a book, a foreign key to the `Titles` table.

Fig. 24.7

`AuthorISBN` table from the `books` database.

Every foreign-key value must appear as another table's primary-key value so the DBMS can ensure that the foreign key value is valid—this is known as the **Rule of Referential Integrity**. For example, the DBMS ensures that the `AuthorID` value for a particular row of the `AuthorISBN` table is valid by checking that there is a row in the `Authors` table with that `AuthorID` as the primary key.

AuthorID	ISBN	AuthorID	ISBN
1	0132151006	2	0133379337
2	0132151006	1	0136151574
3	0132151006	2	0136151574
1	0133807800	4	0136151574
2	0133807800	1	0133378713
1	0132575655	2	0133378713
2	0132575655	1	0133764036
1	013299044X	2	0133764036
2	013299044X	3	0133764036
1	0132990601	1	0133570924
2	0132990601	2	0133570924
3	0132990601	3	0133570924
1	0133406954	1	0132121360
2	0133406954	2	0132121360
3	0133406954	3	0132121360
1	0133379337	5	0132121360

Fig. 24.8

Sample data from the `AuthorISBN` table of `books`.

Foreign keys also allow *related* data in *multiple* tables to be *selected* from those tables—this is known as **joining** the data. There is a **one-to-many relationship** between a primary key and a corresponding foreign key (for example, one author can write many books and one book can be written by many authors). This means that a foreign key can appear *many* times in its own table but only *once* (as the primary key) in another table. For example, the `ISBN 0132151006` can appear in several rows of `AuthorISBN` (because this book has several authors) but only once in `Titles`, where `ISBN` is the primary key.

Entity-Relationship (ER) Diagram

There's a one-to-many relationship between a primary key and a corresponding foreign key (e.g., one author can write many books). A foreign key can appear many times in its own table, but only once (as the primary key) in another table. Figure 24.9 is an **entity-relationship** (**ER**) **diagram** for the `books` database. This diagram shows the *database tables* and the *relationships* among them. The first compartment in each box contains the table's name, and the remaining compartments contain the table's columns. The names in italic are primary keys. *A table's primary key uniquely identifies each row in the table.* Every row must have a primary-key value, and that value must be unique in the table. This is known as the **Rule of Entity Integrity**. Again, for the `AuthorISBN` table, the primary key is the combination of both columns—this is known as a composite primary key.

The lines connecting the tables represent the *relationships* among the tables. Consider the line between the `Authors` and `AuthorISBN` tables. On the `Authors` end, there's a `1`, and on the `AuthorISBN` end, an infinity symbol (∞). This indicates a *one-to-many relationship*—for *each* author in the `Authors` table, there can be an *arbitrary number* of ISBNs for books written by that author in the `AuthorISBN` table (that is, an author can write *any* number of books). The relationship line links the `AuthorID` column in the `Authors` table (where `AuthorID` is the primary key) to the `AuthorID` column in the `AuthorISBN` table (where `AuthorID` is a foreign key)—the line between the tables links the primary key to the matching foreign key.

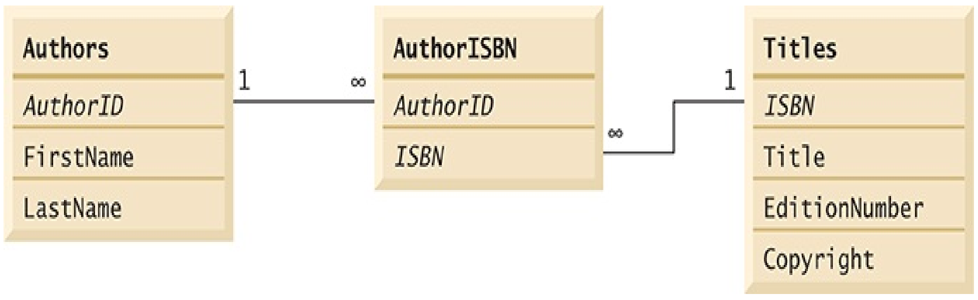

Fig. 24.9

Table relationships in the `books` database.

Description

The line between the `Titles` and `AuthorISBN` tables illustrates a *one-to-many relationship*—one book can be written by many authors. Note that the line between the tables links the primary key `ISBN` in table `Titles` to the corresponding foreign key in table `AuthorISBN`. The relationships in Fig. 24.9 illustrate that the sole purpose of the `Author-ISBN` table is to provide a **many-to-many relationship** between the `Authors` and `Titles` tables—an author can write *many* books, and a book can have *many* authors.