a) Figure A-33 shows an initial data model for a small library.Explain to the librarian what the initial data model means.

b) Design tables for a relational database which would capture the information represented by the model. Include primary and foreign keys and other appropriate constraints.

EXCERCISE 7-1

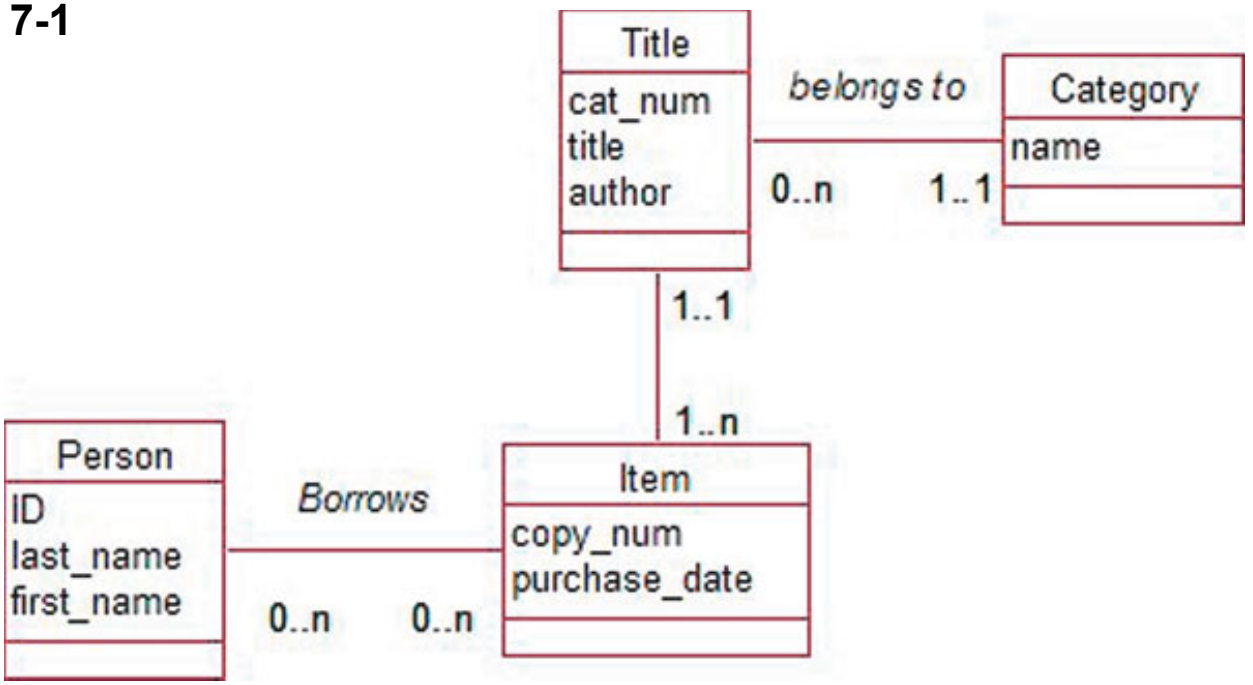


Figure A-33. *Draft data model for a small library*

a) Here is a possible way to explain to the librarian what the data model means.

"The model shows that you are intending to keep information about people (who each have a number and names). You are also keeping information about the book titles you stock: author, title, and category. For each of these titles there is at least one copy and possibly more. These copies have a copy number and the date it was purchased. You will be keeping information about which people borrow a particular copy."

There are clearly important things missing from this model—we will get to some of them!

b) Now let's look at how to represent the model in a relational database.

For each class we need a table with fields to represent the attributes.

A direct translation gives us:

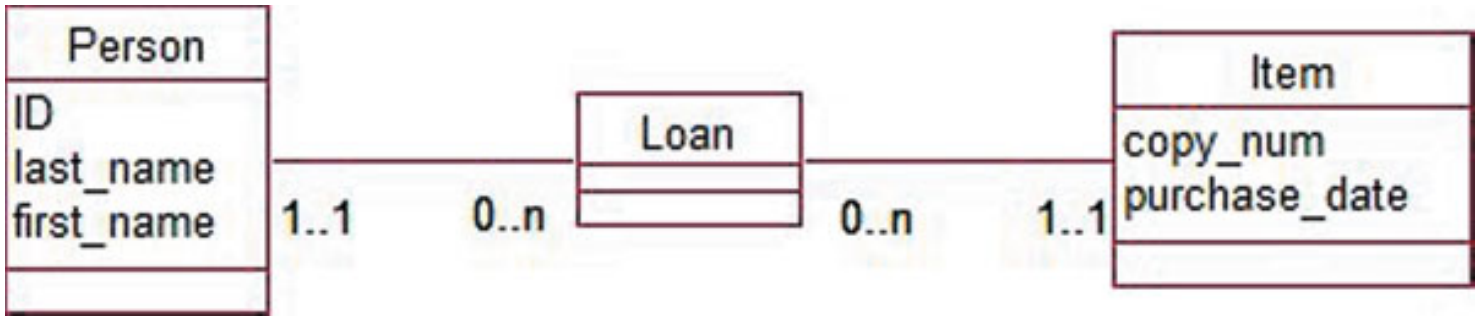
```
Person   (ID, last_name, first_name)
Title    (cat_num, title, author)
Item     (copy_num, purchase_date)
Category (name)
```

We need some types for the fields. `ID` should be an integer and you might consider some sort of auto-incrementing type. Depending on the cataloging system, `cat_num` will probably be a text field. `copy_num` will be an integer (1, 2, 3, etc.), `purchase_date` is obviously a date, and the other fields are all text. We should probably look more closely at the `author` field. There are a few possibilities here; at the least it should be split into first and last names, but as people are likely to want to search by author it probably should be a class on its own. We won't worry about that for now.

What about constraints? All the fields seem very important to me and I see no reason not to make them all required (not null). The purchase date could perhaps be a date in the past, but then that might cause problems if planned future purchases need to be stored (and that increases the scope so we'll just leave that for now too.)

Primary keys are easy for the `Person (ID)`, `Title (cat_num)`, and `Category (name)` tables, but the `Item` table does not have an obvious candidate for a primary key. Every title will have a copy with `copy_num = 1` and they may have been purchased on the same date. So no field or combination of fields is suitable. We could add a unique identifier but we'll wait a bit to see what eventuates.

Now to represent the relationships. The 1–Many between `Title` and `Category` can be represented with a foreign key in the `Title` table and similarly we need a foreign key in the `Item` table referencing the `Title` table. These must have the same datatypes as the tables they are referencing. To represent the Many–Many relationship `borrows`, we need to first add a new class (`Loan`) as in Figure A-34.



ure A-34. *A new class to help represent the Many–Many relationship borrows*

At this stage we should realize (you probably did before now) that there is information about pairings of `Person` and `Item` that need to be stored (`date_out`, `due_date`, etc). These will need to be discussed with the client. We'll just make a note of it for now. To represent the two 1–Many relationships in Figure A-34 we will need two foreign keys in the `Loan` table referencing `Person` and `Item`.

Below, I have recapped our progress so far. (I have used emboldened underlines to denote primary keys and italics to denote foreign keys.)

`Person`	(**`ID`**, `last_name, first_name`)
`Title`	(**`cat num`**, `title, author,` ***`category`***)
`Item`	(`copy_num, purchase_date,` ***`title_cat_num`***)
`Category`	(**`name`**)
`Loan`	(`<dates>,` ***`item, person`***)

We are nearly there; we just have the primary keys for the `Item` and `Loan` tables to decide. For the `Item` table it is now possible to use the combination of `title_cat_num` and `copy_num` as a primary key. Each title has a unique catalog number and each title will have only one copy1, one copy2, and so on. For the `Loan` table the combination `item` and `person` is not suitable; someone may check out the same item a second time. However, if we include the date it was taken out, then the three fields would be unique (assuming you don't check an item out twice in one day). It would also be possible to add a unique loan number to act as a primary key in the `Loan` table (we look at this choice in Chapter 8).

Listing A-1 shows the SQL to create these tables in SQL Server (we haven't covered all the required syntax for dealing with concatenated keys and foreign keys in this design book, but it is fairly self explanatory).

Listing A-1. SQL to Create Library Database

```
CREATE TABLE Category (name VARCHAR (20) PRIMARY KEY);

CREATE TABLE Person  (ID INT PRIMARY KEY,
                      last_name VARCHAR(20) NOT NULL,
                      first_name VARCHAR(20) NOT NULL);

CREATE TABLE Title   (cat_num VARCHAR(40) PRIMARY KEY,
                      title VARCHAR(80) NOT NULL,
                      author VARCHAR(80) NOT NULL,
                      category VARCHAR(20) NOT NULL FOREIGN KEY REFERENCES Category);

CREATE TABLE Item    (title_cat_num VARCHAR (40) FOREIGN KEY REFERENCES Title,
                      copy_num INT,
                      purchase_date DATE NOT NULL,
                      PRIMARY KEY (title_cat_num, copy_num));

CREATE TABLE Loan    (date_out DATE,
                      cat_num VARCHAR(40),
                      copy_num INT,
                      borrower INT FOREIGN KEY REFERENCES Person,
                      FOREIGN KEY (cat_num, copy_num) REFERENCES Item,
                      PRIMARY KEY (date_out, cat_num, copy_num, borrower) )
```