about the Student database. What data should it contain? Possibilities are: *Students, Classes, Grades, Emails, Office_Visits, Majors, Advisers, Student_Organizations*—the list could go on and on. Further, how much detail should be included in each? Should the database include campus addresses? Home addresses? Billing addresses?

In fact, there are dozens of possibilities, and the database developers do not and cannot know what to include. They do know, however, that a database must include all the data necessary for the users to perform their jobs. Ideally, it contains that amount of data and no more. So, during database development the developers must rely on the users to tell them what to include in the database.

Database structures can be complex, in some cases very complex. So, before building the database the developers construct a logical representation of database data called a **data model**. It describes the data and relationships that will be stored in the database. It is akin to a blueprint. Just as building architects create a blueprint before they start building, so, too, database developers create a data model before they start designing the database.

For a philosophical perspective on data models, see the Guide on pages 168–169.

Figure 5-16 summarizes the database development process. Interviews with users lead to database requirements, which are summarized in a data model. Once the users have approved (validated) the data model, it is transformed into a database design. That design is then implemented into database structures. We will consider data modeling and database design briefly in the next two sections. Again, your goal should be to learn the process so that you can be an effective user representative for a development effort.

What Is the Entity-Relationship Data Model?

The **entity-relationship (E-R) data model** is a tool for constructing data models. Developers use it to describe the content of a data model by defining the things (*entities*) that will be stored in the database and the *relationships* among those entities. A second, less popular, tool for data modeling is the **Unified Modeling Language (UML)**. We will not describe that tool here. However, if you learn how to interpret E-R models, with a bit of study you will be able to understand UML models as well.

Entities

An **entity** is some thing that the users want to track. Examples of entities are *Order, Customer, Salesperson,* and *Item*. Some entities represent a physical object, such as *Item* or *Salesperson*; others represent a logical construct or transaction, such as *Order* or *Contract*. For reasons beyond this discussion, entity names are always singular. We use *Order*, not *Orders*; *Salesperson*, not *Salespersons*.

Entities have **attributes** that describe characteristics of the entity. Example attributes of *Order* are *OrderNumber, OrderDate, SubTotal, Tax, Total,* and so forth. Example attributes of *Salesperson* are *SalespersonName, Email, Phone,* and so forth.

Entities have an **identifier**, which is an attribute (or group of attributes) whose value is associated with one and only one entity instance. For example, *OrderNumber* is an identifier of *Order*, because only one *Order* instance has a given value of *OrderNumber*. For the same reason, *CustomerNumber* is an identifier of *Customer*. If each member of the sales staff has a unique name, then *SalespersonName* is an identifier of *Salesperson*.

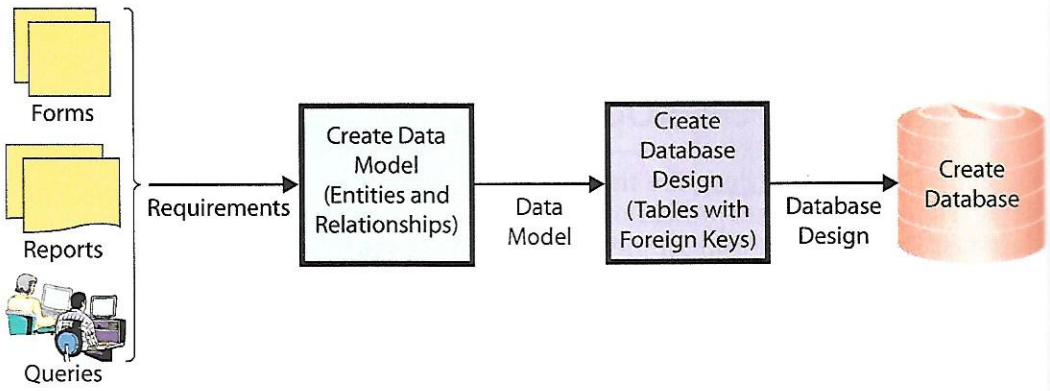

Figure 5-16
Database Development Process