specific information that they want to see on a daily basis, such as news reports and stock prices.

**Preprogrammed versus Ad Hoc** To provide preprogrammed access to data, programmers speak with users and identify a small set of queries that can be programmed once and then used repetitively. To permit ad hoc access, the users must have data access tools that allow them to specify individual queries that will generate the information they want at any particular time. Accordingly, push systems are preprogrammed because the user identifies the specific information to be provided on a schedule or whenever a triggering event occurs, such as a large swing in a company's stock price. Information requests in pull systems are ad hoc because they are specified for an information need that may never recur. Providing easy ways to specify ad hoc queries was one of the major breakthroughs in information systems. Before ad hoc query capability was available, users often had to ask programmers to produce special programs whenever they needed information not included in existing preprogrammed reports.

**DATA MODELING: DEFINING AND ORGANIZING DATA**

The previous section introduced the idea that a database might include data files devoted to different kinds of things, and that the data in these files might be related. This section provides the basic concepts needed to describe data and relationships within a database. Consistent with current system development ideas, the general discussion of these questions is introduced through **data modeling**, the process of defining what data is used or produced in an information system and how that data is organized. Data modeling goes hand in hand with the process modeling introduced in Chapter 3. The basic tool for data modeling is called an entity-relationship diagram.

**Entity-Relationship Diagrams**

Assume you were designing a registration system for a university. What data should the database contain? This question can be broken down into three parts:

- **What are the kinds of things this information system collects data about?** The specific things it collects data about are **entities**. The kinds of things it collects data about are called **entity types**. In a registration system, the entity types usually include courses, professors, students, course sections, classrooms, and perhaps many others. Entity instances of each type might include Economics 101, Professor Jones, Dana Watts, the Monday night section of Economics 101, and classroom E324.

- **What is the relationship between the entity types?** Relationships between entity types govern the entities of each type. The **relationship** between two entity types is the way specific entities of one type might be related to specific entities of the other type. For example, "student" and "section" are entity types. A student can be enrolled in no sections, one section, or several sections.

- **What specific data does the database contain for each entity type?** A database contains the same data items for each entity within a particular entity type. The specific data items stored for each entity type are called its **attributes**. For example, attributes of "student" may include address, telephone number, and whether or not fees have been paid. The attributes of "course" may include a description and a list of prerequisites.

These questions are the basis of **entity-relationship diagrams (ERDs)**, a technique for identifying the entity types in a situation and diagramming the relationships between those entity types. ERDs help in identifying the data in a system and making sure it is represented properly. They help create a shared understanding of the basic ideas underlying the specific data in the system. This technique forces people involved in the analysis to focus on the business situation instead of just listing every relevant item they can think of.

Figure 4.4 contains an entity-relationship diagram for part of a registration system. It uses one of several common notations for ERDs. This diagram identifies six entity
4.4 Entity-relationship diagram for part of a university registration system

This entity-relationship diagram (ERD) identifies six entity types and shows relationships among them. The different types of relationships in ERDs are explained in Figure 4.5.

types and the relationships between those entity types. For example, it says that a course may have no sections or may have one or more, and that each section has a single professor and one or more students.

The relationships in Figure 4.4 apply at some universities, but they aren't true in others. Looking at the ERD raises questions such as:

- Does each professor really belong to exactly one department? Is it possible for a professor to belong to several departments or none?
- Is it possible to have several professors assigned to the same section? This would be the case for a team-taught course.
- Does each section really have a professor, or is the more appropriate term "instructor" since people teaching some courses may not be professors?
- Is it permissible for a course to have no sections? This would be permissible if a course in the catalogue is not offered during a particular semester, but the rules of the school would determine whether that is allowed.
- Is it permissible for a section to have no students? This would certainly be true until the first student signed up for it, but a section that had no students would make no sense after the semester started.

Asking questions such as these is essential in building information systems. They help determine what data will be included and excluded, how the database will be structured, and some of the ways the system will eventually detect errors. In addition, they provide an excellent communication medium for system participants who often have trouble explaining the current and desired situation to the technical staff building the system. The term entity-relationship diagram sounds very technical, but these diagrams are actually used for the nontechnical purpose of identifying the types of things within the system's scope and the relationships among these types of things (see Figure 4.4).

Identifying the Data in a Database

After identifying the entity types and their relationships, it is much easier to identify the data that should be in the system. For each entity type and relationship, this data consists of the attributes that are significant in the situation. Table 4.1 lists some of the possible attributes that might be included in the registration system for each entity type in Figure 4.4. As the analysis of the system continues, these attributes might be renamed or modified, and many other attributes would surely be added.
Common relationships between entity types

- **One-to-one relationship**
  (each professor has one office and each office is assigned to one professor)

- **One-to-many relationship**
  (each section is taught by one professor, but each professor may teach many sections)

- **Optional one-to-many relationship**
  (each course may have many sections, but could have none)

- **Many-to-many relationship**
  (each student may be registered in many sections, and each section may have many students registered in it)

- **Entity type with a relationship to itself**
  (a course may have many prerequisites and may be prerequisite for other courses)

The ERD in Figure 4.4 includes four common relationships: one-to-one, one-to-many, optional one-to-many, and many-to-many. Shown here are examples from Figure 4.4 plus one additional relationship not included in that figure. Other relationships not shown include either-or relationships and relationships between entity types and subentity types.

Believe it or not, the innocuous looking list in Table 4.1 could create a lot of debate among the users and designers analyzing the system. Here are some possible issues:

- **Is any data missing about each entity type?** For example, should the system include course prerequisites or the average grades given by this professor in this course in previous years? Including the prerequisites would be necessary if the system is supposed to check automatically that the student has taken all prerequisites. Including the average grades given by the professor might help students decide which section to attend but would also raise many contentious issues.

- **Are some attributes unnecessary or inappropriate?** For example, do we want to use the professor’s or student’s social security number, birth date, gender, or ethnicity? Attributes such as these might be needed, might be extraneous, or might be improper or illegal to use or divulge.

- **Is there any ambiguity in what the various attributes mean?** For example, does the professor’s address refer to home or office? To avoid mistakes, separate attributes might be named “office address” and “home address.” Even seemingly obvious terms often have different meanings to different people. For example, the Wall Street Journal reported that EuroDollar, the European arm of Chrysler’s Dollar Rent-A-Car, gave one-week specials but considered a week to be five days. A traveler who kept a car for seven days was surprised to receive a bill for one week plus two days.7

- **Do the same attributes appear in two places?** Notice how office telephone is an attribute of professor and telephone extension is an attribute of office. This kind of thing causes confusion for two reasons. First, there are two different terms for the same thing, and second, the information system needs to have each item in one place to make sure it is updated correctly. Telephone extension should be either an attribute of the professor or an attribute of the office, but not both.

It would be easy to generate many more questions about the details of how things are named and what attributes of which entity types should be included in the system.
### Possible Attributes for the Entity Types in Figure 4.4

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Possible attributes of this entity type</th>
</tr>
</thead>
</table>
| Department  | • Department identifier  
• College  
• Department head  
• Scheduling coordinator |
| Course      | • Course number  
• Department  
• Required of department major (y/n)  
• Course description |
| Section     | • Section identification number  
• Semester  
• Year  
• Classroom  
• Start time  
• End time  
• Days of week for class meetings |
| Professor   | • Employee identification number  
• Name  
• Address  
• Birth date  
• Office telephone  
• Social security number |
| Student     | • Student identification number  
• Name  
• Address  
• Birth date  
• Telephone  
• Gender  
• Ethnic group  
• Social security number |
| Office      | • Office number  
• Building  
• Telephone extension |

This analysis can be tedious and requires great attention to detail. Notice, however, that the main questions are about the situation and the logic of the registration process, not about the details of computer technology. Answering these questions incorrectly could result in work wasted developing an information system ill-suited to the situation.

Data modeling is a comparatively new idea in building information systems. The first paper on the entity-relationship diagram was published in 1976, but the need for this step is now widely accepted and has been incorporated into system development methods because it summarizes the business view of the data stored in the database.

### Reality Check

Data Modeling

We have introduced the idea of data modeling and have explained how entity relationship diagrams work.

1. Study Figure 4.4 and modify it to make it more consistent with your understanding of how your university’s registration system operates. Add or remove entity types and relationships as necessary.

2. Study Table 4.1 and modify it to include other attributes you think might be important in the registration process. Include attributes of any entity types you added in question #1.