CHAPTER 6

OBJECT-BASED DATA MODEL: PRINCIPLES AND COMPONENTS

CHAPTER OBJECTIVES

- Understand the essentials of the object-based modeling method
- Examine and study each component of the data model
- Discuss special types of object sets represented in the data model
- Explore relationships among object sets in detail
- Study means for creating an object-based data model

Before proceeding further, let us recapitulate what we have covered so far on data modeling so that you can grasp the significance and place of the object-based data model in logical design. By now, you understand clearly that a data model is a representation or replica of the portion of real-world information that is relevant to the organization. We reviewed the concept of data views. These are user views of data to be present in the proposed database. All data views, merged and aggregated, form the basis for the data model.

We also discussed different levels of the data model. The semantic data model, capturing the true meaning of real-world information and created from the integration of user views, is said to be at the highest level of integration. Two techniques are generally employed to create a semantic data model: the object-based data modeling technique and the entity-relationship data modeling technique. These techniques are equally effective.

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4

OVERVIEW OF OBJECT-BASED MODELING

Data modeling deals with creating data models to represent real-world information. Object technology relates to the treatment of business objects, their relationships, and how they interact with one another. Object-based data modeling is a method or discipline that combines the concepts of data modeling and object technology.

You make use of both concepts and arrive at a method to represent real-world information. That is, you look for business objects in the real world, relate real-world information with the business objects, and use this association to build a data model. Obviously then, the modeling method centers on the business objects. Figure 6-1 expresses this merging of the two concepts in object-based data modeling.

A Generic Data Model

First of all, the object-based data model is a generic data model. Let us examine the significance and implications of this statement. What are the types of databases that are being implemented in organizations? You hear about an organization using a specific database product from a particular vendor. For example, the database could be an Oracle database, an Informix database, or a Microsoft SQL Server database. Or the database could be an IBM IMS database.

Oracle, Informix, and SQL Server databases are based on the relational model. The relational model is a data model in which all data are perceived as two-dimensional tables. On the other hand, the IMS database is based on the hierarchical model. The hierarchical model is also a data model but is a model in which all data are perceived as hierarchical data segments with parent-child relationships. The relational model and the hierarchical model are specific models based on defined conventions on the arrangement of the components. These two models are specific conventional models.

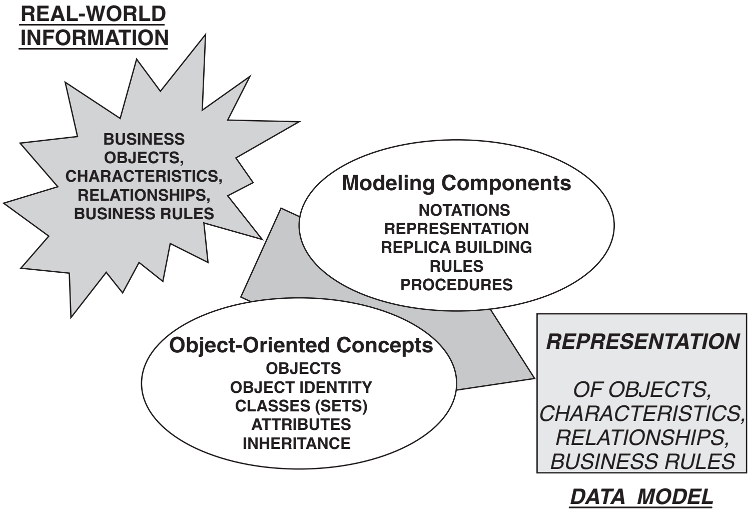


Figure 6-1 Object technology and data modeling.

In contrast, the object-based data model is a broad type of model that can be transformed into any type of conventional model dictated by nature of the commercial database management system to be implemented. In the object-based data model, no preset or predefined arrangement of data structures exists. The data model is generic, applicable to all types of information content.

Benefits of Object-Based Data Model

Being generic, the object-based data model provides a number of benefits to analysts and designers for creating a truly representative data model. Let us review the major advantages. The model is closer to the way business processes are executed.

Universal model. The model is able to reflect all types of real-world information. Also, you can represent all aspects of information—the business objects and everything relevant to each type of object.

True replica. Because it is able to represent every relevant aspect of real-world information, the data model can be a true and correct representation of real-world information.

Free from restraints. The conventional models adopt certain specific ways of representing data. Through these models, you need to perceive real-world information in predefined ways in the form of a particular arrangement of data structures. The object-based data model has no such restrictions.

Easily transformable. Once you create an object-based data model, you can easily transform it into any of the conventional data models.

Intuitive. In the object-based data model, you observe as components just what you notice as different aspects of real-world information. Therefore, learning the modeling technique becomes easy.

Introduction to Components

When you create a data model and produce a data model diagram, you need symbols or notations to denote the various components. In Chapter 5, when we introduced data modeling, we did not use any formal notations. There, the intention was to introduce the basic concept of data modeling. So let us now go over the notations.

Although you might see some variations in the notations, mostly these are standardized. Figure 6-2 presents the components and their standard notations.

Note how these components cover the various aspects of real-world information. Also, realize how, with a just a handful of components, you can represent the entire spectrum of real-world information. With a limited number of building blocks or components, you are able to represent any set of information with all its variations and complexities.

Object Set. The rectangular box denotes a set of object instances. Each box in a data model diagram represents the set of certain object instances. The name of the

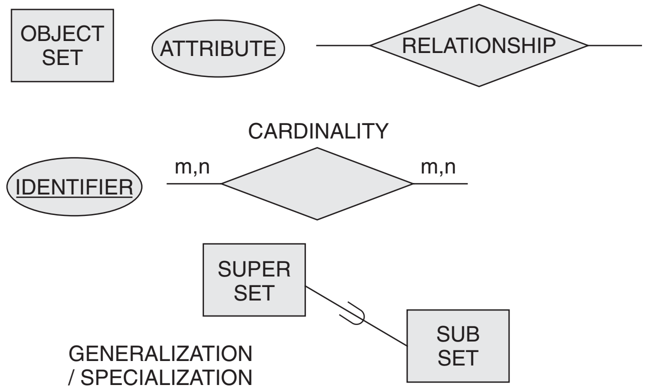


Figure 6-2 Object-based data model: components.

business object is written inside the box. For example, if a box indicates the STUDENT object, it symbolizes the set of all students in the university. You may visualize each student as being represented by a single point within the box. Then the complete set of points represents the entire set of students.

Attribute. The ellipse or oval indicates an attribute or inherent characteristic of an object. The name of the attribute goes inside the ellipse. If an ellipse denotes the attribute *First Name*, then the name of this attribute is written inside the ellipse and the ellipse is attached to the box that represents the STUDENT object.

Identifier. Inside the box representing the business object STUDENT, you assume the presence of multiple points, each denoting a single student. Each is an instance of the object in the object set. How do you uniquely identify any particular instance of the object set? How do you find the first name of a specific student? You need a method to identify a specific student instance and then find the name attribute for that instance. The values of certain attributes of an object may be used to identify instances of the object uniquely. For example, values of the attribute *Social Security Number* can uniquely identify individual instances of the object STUDENT. Social Security Number 101-38-4535 uniquely points to a particular instance of the object STUDENT. The identifier, being an attribute, is also represented by an ellipse with the name of the attribute written inside. Because it is a special type of attribute, the name written inside the ellipse is underlined or underscored.

Relationship. In the real-world information for a university, a student enrolls in a class of a particular course. Here STUDENT and CLASS are business objects. Henry Porter enrolls in the database class meeting on Tuesday nights. The instance "Henry Porter" of STUDENT object is related to the instance "Tuesday DB class" of CLASS object. A diamond, with two lines joining the two related object boxes, denotes relationships. The name of relationship itself, "enrolls" in this case, is written inside the diamond.

Cardinality. Consider the object instances "Henry Porter" and "Tuesday DB class." Henry Porter must have enrolled for other classes such as Systems Analysis and Design, Data Structures, and so on. That means that one instance of the object STUDENT is related to many instances of the object CLASS. Similarly, on the other side, one instance of "Tuesday DB class" of the object CLASS is related to many instances of the object STUDENT. In other words, many students are enrolled for the "Tuesday DB class." The indication of numbers of instances of one object relating to number of instances of another object is known as the cardinality of the relationship. Cardinality of a relationship is shown as numbers near the lines of the diamond symbol for the relationship. The variables "m, n" indicate a many-to-many relationship between two object sets.

Generalization/Specialization. Think about the business object STUDENT. The object set representing STUDENT includes all students in the university. Both graduate students and undergraduate students are part of the STUDENT object set. The set of graduate students is one part of the STUDENT object set; the set of undergraduate students is the other part. Both graduate students and undergraduate students share most of the attributes of the object called STUDENT. However, graduate students may have attributes such as undergraduate degree and undergraduate major that do not apply to undergraduate students. What you observe is that the set of graduate students is a subset of the superset STUDENT. GRADUATE and UNDERGRADUATE are subsets; STUDENT is the superset. Subsets and supersets commonly occur in real-world situations. Look at supersets and subsets from another angle. A subset is a specialization of a superset; the superset is a generalization of the subsets. As shown in Figure 6-2, a special mark like a "U" is placed on the line linking the subset to the superset to indicate that the superset contains the instances of the subset.

Mapping of Real-World Information

We have mentioned that the object-based data model is an intuitive model, a true model, and a model that represents all types and variations of real-world information. With the aid of just the defined set of components and their notations, you are able to represent any collection of information requirements.

Let us take specific examples of business processes in real-world situations of different types of organizations. Note the following details indicating how you can represent information requirements with the principles of object-based data modeling. At this introductory stage, let us describe how portions of the data model diagram would be completed without getting into any complex issues. We just want to map the elements of real-world information to the components and notations of the data model.

Airline Reservation

Business Objects Show object boxes for PASSENGER, AIRLINE, and FLIGHT.

Attributes Show attribute ellipses for characteristics of the above three objects.

Identifiers Show attribute ellipses for following identifiers, underlining the attribute names within the ellipses:

PASSENGER: Passenger Number (or Record ID assigned)
AIRLINE: Airline Code (AA, BA, AI, etc.)
FLIGHT: Flight Number (preassigned)

Relationships Indicate relationships, with diamond shapes, between FLIGHT and AIRLINE, PASSENGER and FLIGHT.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization None required.

Taking a Customer Order

Business Objects. Show object boxes for CUSTOMER, ORDER, and PRODUCT.

Attributes Show attribute ellipses for characteristics of the above three objects.

Identifiers Show attribute ellipses for the following identifiers, underlining the attribute names within the ellipses:

CUSTOMER: Customer Number (system-assigned)
ORDER: Order Number
PRODUCT: Product ID (preassigned)

Relationships Indicate relationships, with diamond shapes, between CUSTOMER and ORDER, ORDER and PRODUCT.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization None required.

Student Registration

Business Objects Show object boxes for STUDENT, COURSE, CLASS, and TERM.

Attributes Show attribute ellipses for characteristics of the above four objects.

Identifiers Show attribute ellipses for the following identifiers, underlining the attribute names within the ellipses:

STUDENT: Student ID or Social Security Number
COURSE: Course ID
CLASS: Class Number or Section Number
TERM: Semester ID

Relationships Indicate relationships, with diamond shapes, between COURSE and CLASS, TERM and COURSE, STUDENT and CLASS.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization If necessary, show superset STUDENT and subsets GRADUATE STUDENT and UNDERGRADUATE STUDENT.

Product Inventory Verification

Business Objects Show object boxes for PRODUCT, PRODUCT LINE, DEPARTMENT, and WAREHOUSE.

Attributes Show attribute ellipses for characteristics of the above four objects.

Identifiers Show attribute ellipses for the following identifiers, underlining the attribute names within the ellipses:

PRODUCT: Product ID
PRODUCT LINE: Product Line Number
DEPARTMENT: Department Code
WAREHOUSE: Warehouse ID

Relationships Indicate relationships, with diamond shapes, between DEPARTMENT and PRODUCT LINE, PRODUCT LINE and PRODUCT, WAREHOUSE and PRODUCT.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization None required.

Patient Service at Doctor's Office

Business Objects Show object boxes for PHYSICIAN, PATIENT, DIAGNOSIS, and TREATMENT.

Attributes Show attribute ellipses for characteristics of the above four objects.

Identifiers Show attribute ellipses for following identifiers, underlining the attribute names within the ellipses:

PHYSICIAN: Social Security Number or Certification Number
PATIENT: Patient Number (assigned)
DIAGNOSIS: Diagnostic Code
TREATMENT: Procedure ID

Relationships Indicate relationships, with diamond shapes, between PHYSICIAN and PATIENT, PATIENT and DIAGNOSIS, PATIENT and TREATMENT.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization None required.

Deposit to Checking Account

Business Objects Show object boxes for ACCOUNT, CHECKING ACCOUNT, CUSTOMER, and DEPOSIT.

Attributes Show attribute ellipses for characteristics of the above four objects.

Identifiers Show attribute ellipses for following identifiers, underlining the attribute names within the ellipses:

ACCOUNT: Account Number
CHECKING ACCOUNT: Account Number

CUSTOMER: Customer Number (preassigned)
DEPOSIT: Transaction Number

Relationships Indicate relationships, with diamond shapes, between CHECKING ACCOUNT and CUSTOMER, CHECKING ACCOUNT and DEPOSIT, CUSTOMER and DEPOSIT.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization Show notations to indicate superset ACCOUNT and subset CHECKING ACCOUNT.

Medical Insurance Claims Processing

Business Objects Show object boxes for INSURED, POLICY, AGENT, and CLAIM.

Attributes Show attribute ellipses for characteristics of the above four objects.

Identifiers Show attribute ellipses for the following identifiers, underlining the attribute names within the ellipses:

INSURED: Social Security Number
POLICY: Policy Number
CLAIM: Claim Number (assigned)
AGENT: AGENT ID (assigned)

Relationships Indicate relationships, with diamond shapes, between INSURED and POLICY, AGENT and POLICY, CLAIM and POLICY.

Cardinalities Number to show cardinalities at either side of the diamonds for the above relationships.

Generalization/Specialization None required.

Example of a Model Diagram

From the above real-world information requirements and their mapping to the object-based data model, you have a glimpse of how the model is created and the model diagram drawn. Note, however, that we have just introduced the principles; we have not gone into too many details or dealt with any complexities in the real-world business situations. Nevertheless, you now have a more concrete notion about how a model represents real-world information.

The subsequent sections of this chapter elaborate on the individual model components. Before getting into a discussion of the details, let us take a specific example of real-world information about a small group practice of surgeons, review a data model diagram, and note the components. The description of the group practice operations and information requirements follows:

Three orthopedic surgeons, Dr. Samuel J. Laufer, Dr. Andrew Bowe, and Dr. Lisa Sanderson, have joined together to form Orthopedic Associates P.A. They want to install a computer system for their group practice. They also want to use a database for their information requirements. Initially, the database is required to support just the billing function.

Group Practice Operations The three surgeons are equal partners in the group practice. Irrespective of how many patients each surgeon attends to, they share the net income equally.

There are two receptionists, a business manager, and three nurses.

Patients About 20% of patients are repeat patients. The rest are new referrals. Patient visits are usually by appointment. The receptionists are responsible for scheduling the appointments for the surgeons. Patients may be allotted to the surgeons based on the availability of the surgeons and the daily schedule. Sometimes, patients may opt to see a particular surgeon, although this is not encouraged.

Billing The business manager is responsible for billing for each visit. On the basis of the diagnosis and treatment, the doctor or sometimes the nurses write down the fee for the visit. Services may include some surgical procedures performed in the premises and various types of tests. Third-party providers such as Blue Cross/Blue Shield, other insurance companies, Medicare, or Medicaid are billed. Wherever the third-party providers do not cover a the cost of a visit fully, the balance is billed to the patient. Third-party providers are invoiced for each visit. The patients receive a monthly statement showing any opening balance, the visits during the month, the amounts billed to third-party providers, payments received from the providers, amounts billed to the patient, payments received from the patient, and any balance due. Third-party providers may include one or more invoices in one payment. A patient may make a partial payment against a billing statement.

Initial Information Requirements

- Send billing invoices to third-party providers.
- Send monthly statements to the patients.
- Receive payments from third-party providers.
- Receive payments from patients.
- At the end of each month, print a list of the names of patients seen by each surgeon.

The first version of the database is intended to support only the initial information requirements. Therefore, the data model has to be created *just for the initial information requirements*.

Figure 6-3 presents a data model diagram for the initial information requirements. As desired, it is intended to represent only the information relevant to patient services and billing. It is important to note that other irrelevant details are filtered out before the model is created.

Carefully observe the notations or symbols in the model diagram and the components represented by these symbols.

Business Objects Note the rectangular boxes denoting the objects. Only the objects involved in the initial information requirements are included. You do not see an object for EMPLOYEE because this is not required for the initial information requirements. Also, note the bigger box representing the object VISIT.

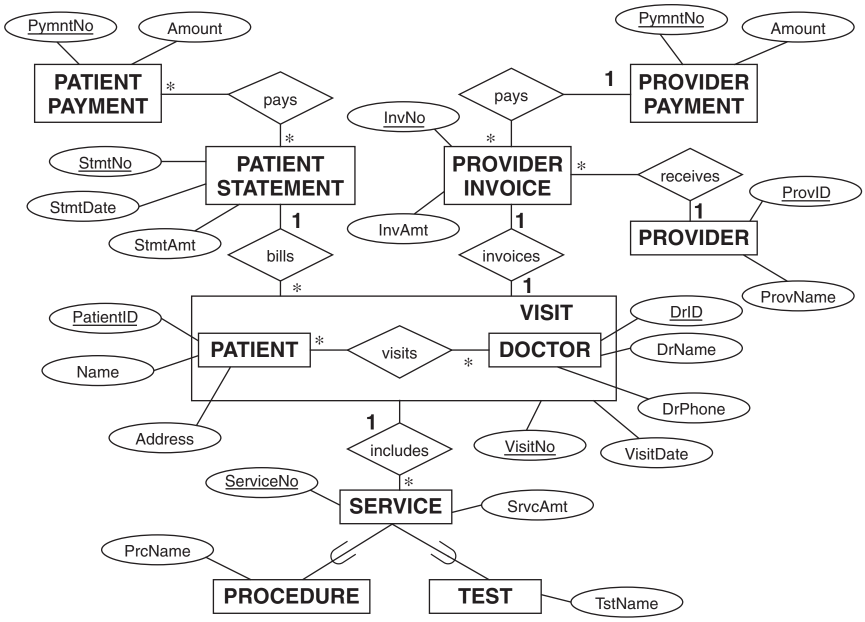


Figure 6-3 Model diagram: medical practice.

This is an aggregate object. See the explanation of aggregate objects in a later subsection.

Attributes Review the ellipses attached to each object. These are the inherent or intrinsic characteristics of the objects. Each attribute is a basic property of the object to which it is connected. Note also that only the characteristics that are relevant to the initial information requirements are included.

Identifiers Note the ellipses containing attribute names that are underlined. Each such ellipse denotes the identifier for the particular object.

Relationships Look at all diamonds and the lines connecting pairs of objects. These are the direct relationships. What are direct relationships? Consider the object PROVIDER PAYMENT. Of course, this payment pertains to a visit by a patient. Nevertheless, the payment is directly related only to invoice sent to the third-party insurance provider. The relationship of a payment from a provider to the patient is an indirect relationship. The model diagram only needs to show the direct relationships. The indirect relationships may be derived from the direct relationships displayed in the diagram.

Cardinalities Note the numbers marked along the relationship lines. Number to show cardinalities at either side of the diamonds for the above relationships. An asterisk (*) indicates that many instances of the object participate in the relationship. Note the cardinalities between the objects PROVIDER and PROVIDER INVOICE. One instance of the object PROVIDER may be related to many instances of the object PROVIDER INVOICE.

Generalization/Specialization Note the superset SERVICE and the subsets PROCEDURE and TEST. SERVICE is a generalization of PROCEDURE. On the other hand, TEST is a special type of SERVICE.

BUSINESS OBJECTS

In an object-based data model, as the name itself implies, the business object is the primary building block. The preliminary step for creating a data model rests on the identification of the business objects for a given set of information requirements. After the identification of business objects, the rest of data modeling follows.

You need to clearly understand the concept of a business object and the methods for identifying business objects and to explore how the objects relate to one another. You need to differentiate between an object set and an instance of an object set. You need to examine the nature of objects and see what different types of objects exist.

We have been discussing characteristics of objects and named these "attributes." What are these attributes for an object? Consider the object EMPLOYEE. An employee has a definite street address and is attached to a department. Are both *address* and *department* attributes of the object EMPLOYEE? If not, why not? You need to understand how characteristics qualify to become attributes of objects in a data model.

Object Sets and Instances

Sets Pursue the example of the object EMPLOYEE. As you know, a rectangular box indicates the object EMPLOYEE in a data model with the name of the object written inside the box. It is worthwhile repeating what we mentioned above about what the box actually represents. It does not just represent a single employee. It indicates the entire set of employees in the organization. The importance of the concept of an object set becomes clear when we study the concept of relationships.

Thus the object set EMPLOYEE represented by a box is the collection of all individual employees. Within the box, imagine the single points as denoting individual employees. The box contains a set of such points.

Instances Each point represents an instance or individual occurrence of the object EMPLOYEE. If you examine a single point in the box, that point would denote a particular employee. If there are 500 employees in the organization, then 500 individual points make up the box that represents the object EMPLOYEE. Each of the 500 points represents a specific employee.

Examples:

Object Set	EMPLOYEE
Instances	John Doe, Mary Smith, Jane Williams, Maryanne Rogers
Object Set	CUSTOMER
Instances	Harold Epstein, Bill Richardson, Walker Electricals, Inc., Jerry Philips and Sons, Inc.
Object Set	PRODUCT
Instances	Coat, slacks, shirt, dress, socks, hat, gloves
Object Set.	INVOICE
Instances.	No.1234 dated 10/15/2002, No.2345 dated 12/23/2002, No.3456 dated 1/10/2003
Object Set	BUILDING
Instances	121 Main Street, 33 Park Avenue, 101 Center Boulevard, 25 Ray Road

Types of Objects

We have gone through several examples of business objects and their instances. You now have a fairly good understanding of objects represented in a data model. You have seen that EMPLOYEE is a business object. You have also noted that PRODUCT, INVOICE, and BUILDING are objects as well. What then are objects? What are the various types of objects? How do you differentiate between one type of object and another?

A person, as in the case of EMPLOYEE, is an object. A BUILDING is a material object. In the case of INVOICE, it is neither a person nor a material. When you refer to the object INVOICE, you do not indicate the piece of paper on which the invoice details are printed and sent to the customer. The piece of paper is a manifestation of a single instance of the object INVOICE. What then is the object INVOICE? It is a concept that covers the billing to a customer for services or products.

Listed below are the common types of objects:

Person	EMPLOYEE, STUDENT, PATIENT
Place	STATE, COUNTRY, REGION
Material	MACHINE, PART, PRODUCT, BUILDING
Event	SALE, VISIT, APPOINTMENT
Concept	ACCOUNT, COURSE, INVOICE

Recognizing Object Sets

Let us say that you have the responsibility for examining the information requirements of a department store and creating an object-based data model. As you very well know, business objects are the fundamental building blocks for creating the model. So, by inspecting the requirements definition, you must first recognize the business objects that are relevant. How do you accomplish this initial task?

By carefully going through the requirements definition, you will discern the relevant real-world information tied to people, places, materials, events, and concepts. As mentioned above, information requirements for each process tend to be grouped in terms of the relevant things. Initially, note all such things with which the business processes are concerned.

Let us quickly do this exercise for a department store. From the information requirements, you will recognize the things or objects for the business. The store buys and sells products; PRODUCT therefore, is a relevant business object. The store sells products to customers; CUSTOMER becomes a relevant object. The store invoices customers for the products sold; therefore, INVOICE is a relevant object. As you continue the inspection of the information requirements for the department store, one by one, you will recognize all the objects for the business and come up with a complete list.

Occasionally, you may run into a situation in which it may not be clear whether a data element must be recognized as a business object to be shown distinctly in the model or must be indicated as just an attribute of another object. For example, in the case of the data model for the department store, consider the data element *product line.* Each product is part of a product line. The question arises: Should product line be considered as an attribute of the object PRODUCT or be represented as a separate object PRODUCT LINE? If the information requirements demand that the characteristics of product line such as line description, date of introduction of product line, and so on be shown in the model, then product line must be represented as a distinct object. If the information content dictates that only the name of the product line be known, then product line becomes an attribute of the object PRODUCT.

Attributes

Every business object possesses certain properties or characteristics that are relevant to the information requirements. These characteristics describe the various instances of an object set. For example, *last name* McKeown describes an instance of the object EMPLOYEE. Similarly, if that employee was hired on 10/1/1985, *hire date* is another attribute describing that instance of the object.

Figure 6-4 shows the attributes of an object STUDENT. Note the names of the attributes written inside the ellipses or ovals. Observe how these are characteristics or descriptors of the individual instances of the object set.

So, what are attributes? Attributes of an object are characteristics, properties, or descriptors. Let us explore the meaning and significance of attributes further.

Inherent Characteristics Consider the data elements StudentID, StudentName, SocSecNo, StudentPhone, StudentMajor. These data elements are associated with the object STUDENT. They are innate, natural, or inherent properties of STUDENT. Next, think of a particular course for which a student has enrolled. CourseNo is also a data element associated with the object STUDENT. If so, is CourseNo also an attribute of the object STUDENT? Compare the two data elements StudentName and CourseNo. StudentName is a natural characteristic of the object STUDENT, whereas CourseNo does not indicate a basic property of the object. CourseNo does not describe some intrinsic property of STUDENT but only

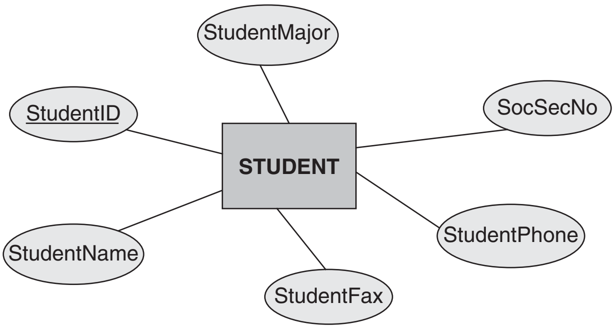


Figure 6-4 STUDENT object: attributes.

StudentID	StudentName	SocSecNo	StudentPhone	StudentMajor	StudentFax
111245	Kristin Rogers	214-56-7835	516-777-9584	Networking	516-777-9587
121365	Rob Smith	123-44-5546	718-312-4488	Web Design	
123456	Mary Williams	101-54-3838	212-313-1267	Networking	212-313-1267
234754	Shaina Gonzales	213-36-7854	212-126-3428	Social Science	
388910	Andrew McAllister	311-33-4520	718-567-4321	Engineering	718-567-4322
400500	Kassy Goodman	512-22-8542	732-346-5533	Programming	732-346-5538
511675	Rob Smith	111-22-3344	908-212-5629	Liberal Arts	908-212-5630

Figure 6-5 STUDENT object: sample attribute values.

its relationship with another object called COURSE. Therefore, CourseNo does not qualify to be an attribute of STUDENT.

Unique Values for an Instance Take a few samples of the attributes for the object STUDENT as shown in Figure 6-5.

Look at the set of values shown in one row of the figure. These values for the attributes StudentID, StudentName, SocSecNo, and StudentPhone relate to one student, a specific instance of the object STUDENT. These attributes do not relate to any random instance of the object; they relate to one particular student.

Let us say that this student is Rob Smith. Then these attributes are characteristics of Rob Smith. If you observe a specific instance of the object STUDENT, namely Rob Smith, then you will note a unique set of values for the attributes. This unique set of values describes Rob Smith. In the data model, each instance of an object possesses a unique set of values for its attributes.

Changeable Values We have noted that each instance of an object is described by a unique set of values for its attributes. Review the unique set of values for the attributes describing the instance represented by Mary Williams in Figure 6-5. Let us say that Mary marries John Pearson and changes her name to Mary Pearson. Also, she changes her phone number. What are the implications?

What you notice is that although each instance is described by a unique set of values for its attributes, these values may change over time. The values for StudentName and StudentPhone would change, but the instance would still refer to the same student. Again, after the changes, a unique set of values for the attributes describes the student Mary Pearson. It is important to note that values of the attributes for an object instance may change, but the instance itself remains the same.

Null Values Suppose the object STUDENT has another attribute, StudentFax. What about the values of this attribute for the various object instances? If Mary Pearson has a fax machine, then that fax number will be the value for StudentFax for her.

On the other hand, if Rob Smith does not have a fax machine, then what about the value for StudentFax for Rob Smith? We then say that the attribute StudentFax for Rob Smith has a *null* value. If an object instance has no value for one of its attributes, then this attribute has a null value for that instance. Null values are not blanks or spaces. Null value for an attribute in an object instance indicates the absence of a value for that attribute in that instance.

Null values for attributes play a significant role in databases. The value of an attribute for a particular instance of an object may be set to null if a value is not available, is missing, or is genuinely absent. In a database, as you will see later, null values may not be permitted for certain attributes. Using appropriate language commands, you can check for null values in attributes for object instances.

Attribute Domains Let us now consider the example of a different object and its possible attributes.

Object:	EMPLOYEE
Attributes:	EmployeeID, SocSecNo, EmpName, EmpZip, Salary, HireDate

Examine the values of each of these attributes. You will note that the values for a particular attribute are from a definite set of values. For example, the values of EmployeeID may be any number with six digits. That means that the set of values from which the value of EmployeeID for a particular instance of the object EMPLOYEE is derived is the set of numbers from 000000 to 999999. This is the domain of values for the attribute EmployeeID.

The domain of an attribute is, therefore, the set of legal or allowable values for that attribute. In the case of EmployeeID, the attribute domain is the set of numbers from 000000 to 999999. Attribute domains are sets based on natural limitations as for numbers or characters of the alphabet. Mostly, business rules determine attribute domains.

Each attribute has its own domain for its values. Here are some examples of domains for the above attributes:

EmployeeID	Set of numbers from 000000 to 999999
SocSecNo	Set of legal 9-digit Social Security Numbers
EmpName	Any text up to 45 characters
EmpZip	Set of legal zip codes

Salary	Currency value from 000000.00 to 999999.99
HireDate	Legal values for date greater than January 1, 1900

Identifiers for Instances

Go back to Figure 6-5 displaying sample attributes for object STUDENT. You have noted that the set of values in each row describes a single instance. But do you know which set describes a particular instance? If asked "what is the address of Rob Smith?", how can you find the address from the database? You can go through all instances of the object STUDENT and check for the name value Rob Smith and then find the value for the address. What if there are three students in the university with the same name Rob Smith? Which Rob Smith are we referring to?

You need a method to identify a particular instance of an object. Obviously, values of an attribute may be used to identify particular instances. You tried to do that by looking for STUDENT instances with value "Rob Smith" for the attribute StudentName. However, you could not say for sure that you found the correct student because of the possibility of duplicate values for this attribute. On the other hand, if you had used values of the attribute StudentID, you could uniquely identify the student. What is the address of the student with StudentID 123456? It is Mary Williams; nobody else has that StudentID. Therefore, values of the attribute StudentID can be used to uniquely identify instances of STUDENT.

Attributes whose values can be used to uniquely identify instances of an object are known as identifiers. They are also known as keys for the object. Now, review the attributes for the object STUDENT. StudentName, StudentPhone, or Student-Major cannot be used to identify instances uniquely. This is because these may have duplicate values in the set of instances for the object. Therefore, any attribute that is likely to have duplicate values does not qualify to be a key or identifier for an object.

How about the attributes StudentID and SocSecNo? Either of these attributes qualifies to be a key for STUDENT. Such attributes are called *candidate keys*. One of the candidate keys is chosen to be the identifier for the object. The *primary key* of an object is a candidate key selected as the identifier for the object. For some objects, one attribute alone will not be enough to form the identifier. You may have to use a combination of two or more attributes as the identifier. In such cases, the primary key is a *composite key*.

Refer back to our discussion of null values for attributes. Can an attribute with null values qualify to be a key? If null values are permitted for an attribute, then many instances of the object may have null values for this attribute. That means this attribute is useless for identifying those instances for which the attribute has null values. Attributes for which null values are allowed cannot be used as keys.

Here are a few tips on selecting a primary key:

- Obviously, exclude attributes for which null values are allowed.
- If possible, prefer a single attribute to a combination of multiple attributes.
- Select an attribute whose values will not change for the life of the database system.

- Choose an attribute whose values do not have built-in meanings. For example, if part of the key for an object PRODUCT represents the warehouse code, the key value will have to change if the product is no longer stored in the original warehouse.

RELATIONSHIPS BETWEEN OBJECTS

Consider an order entry process. This is a typical process in most organizations. Examine the information requirements. Apart from others, three business objects feature prominently in the information requirements. The information requirements are about CUSTOMER, ORDER, and PRODUCT. Each object has its own set of attributes. If your data model represents just the three objects and their attributes, then the model will not be a true representation of the information requirements. Of course, in the real-world situation, you have these three objects. But that is not the whole story. These three objects do not just remain in seclusion from one another. The business process of order entry takes place on the basis of associations between the three objects. Accordingly, the data model must reflect these associations.

Customers place orders; orders contains products. At a basic level, the object CUSTOMER and the object ORDER are associated with each other. Similarly, the object ORDER and the object PRODUCT are linked to each other. As you know, such links and associations are represented as relationships in data model diagrams.

Role of Relationships

Let us inspect the associations among the three objects CUSTOMER, ORDER, and PRODUCT. Figure 6-6 shows a data model diagram with these objects.

First, observe the relationship between CUSTOMER and ORDER. The relationship symbol indicates that the relationship exists because in the real-world situation a customer *places* an order. The action of placing an order forms the basis for the relationship. A relationship name is an action word—usually a single verb. Relationships indicate interaction between objects. They represent associations between objects and the types of action that govern the associations.

Next, review the relationship between ORDER and PRODUCT. Here the action is not as apparent; nevertheless, it is the action of an order containing products. The association of ORDER with PRODUCT rests on the fact that orders are for products, that is, orders contain products. The verb or action word in this case is *contains*.

Look at the relationship symbol between CUSTOMER and ORDER. Does the symbol indicate the linkage between the two boxes representing the two objects or individual object instances within each of the two boxes? Figure 6-7 illustrates this important distinction.

The relationship indicated between two object boxes is actually the association between specific instances of one object and particular occurrences of another object. Take a little time to grasp the significance of the meaning and role of relationships symbolized in data model diagrams. The connection shown between two

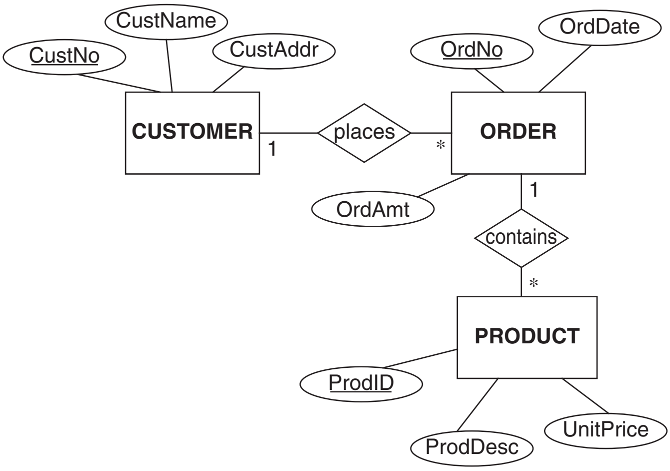


Figure 6-6 Relationships: CUSTOMER, ORDER, PRODUCT.

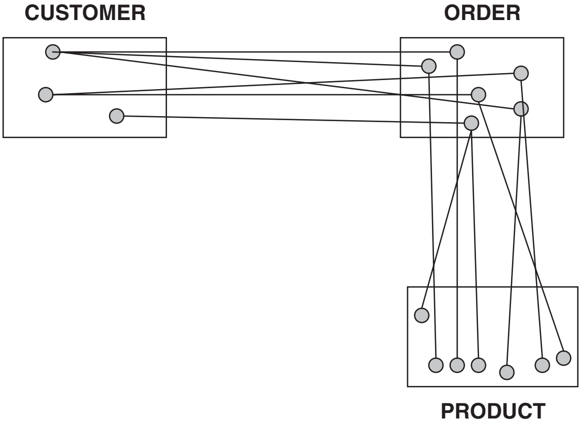


Figure 6-7 Relationships: links between instances.

object boxes simply indicates that the instances are related with one another. The name of the relationship as written within the diamond applies to each link between instances.

Cardinality in Relationships

You now clearly understand that relationships between two objects indicate the associations of instances of one object with instances of a second object. Let us take the example of the relationship between the objects CUSTOMER and ORDER. The relationship exists because a customer places one or more orders. Examine this relationship. Let us say that customer Techchoice, Inc. has placed four orders and that the order numbers for these orders are 1234, 3456, 4567, and 5678. What does this relationship between CUSTOMER and ORDER mean for this example? The explicit association is between customer Techchoice, Inc., an instance of the object CUSTOMER, and orders 1234, 3456, 4567, and 5678, instances of the object ORDER. So far, this is clear enough.

Now, the question arises: If instances of the objects CUSTOMER and ORDER are related, how many instances of the object ORDER can be associated with one instance of the object CUSTOMER? The answer to this question depends on the business rules of the real-world situation. In our example, one customer may place just one order or may place many orders. There is no restriction. That means one instance of the object CUSTOMER is related to many instances of the object ORDER. These numbers, referring to how many instances of one object may be related to how many of the other object, express the cardinality of the relationship.

Cardinality of relationships between two objects as shown in a data model indicates how many instances of the first object may be related to how many instances of the second—whether the number of instances is *one* or *many*. So three possibilities exist for the expression of the cardinality of a relationship: one-to-one, one-to-many, and many-to-many. We will explore these three types further.

One-to-One Relationship Take an example of the objects INVOICE and PAYMENT. Although not practical, suppose a business rule in the organization states that each invoice must be paid in full by a single payment because of accounting restrictions. In this case, the relationship between the objects INVOICE and PAYMENT is a one-to-one relationship. The cardinality is 1 : 1. Figure 6-8 illustrates this one-to-one relationship.

Note the associations indicated between instances of one object with instances of the other object. Observe how only one instance of INVOICE may be connected to a particular instance of PAYMENT. Similarly, only one instance of PAYMENT may be linked to a specific instance of INVOICE. This is a one-to-one relationship—one instance to one instance from either side of the relationship in the model diagram.

One-to-Many Relationship Refer back to the example of the relationship between CUSTOMER and ORDER. Here the relationship is a one-to-many relationship. The cardinality is 1:* (asterisk indicating *many*). Figure 6-9 shows this one-to-many relationship.

Let us go over the associations between the objects instances, one by one. From the side where the object CUSTOMER is shown, select customer instance A. This customer A placed orders numbers 2, 5, and 7. So the relationship observed from CUSTOMER side is: one customer instance related to many order instances. Note the * (asterisk) placed closer to the ORDER object on the relationship line.

Now look at the relationship associations from the ORDER side. Note that order number 1 from this side is linked to only one customer, namely B, on the CUSTOMER side. So the relationship observed from ORDER side is: one order instance is related to one customer instance. Notice the number 1 placed closer to the CUSTOMER object on the relationship line.

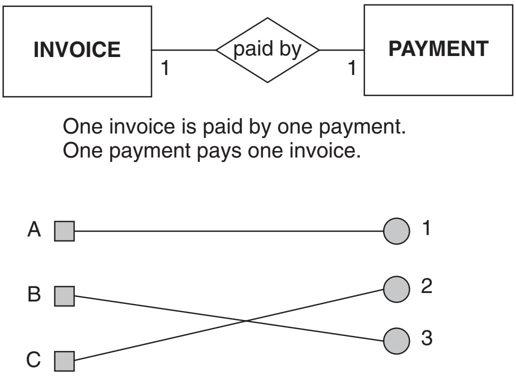

INVOICE
paid by
PAYMENT
1
1
One invoice is paid by one payment.
One payment pays one invoice.
A
B
C
1
2
3

Figure 6-8 Relationship cardinality: one-to-one.

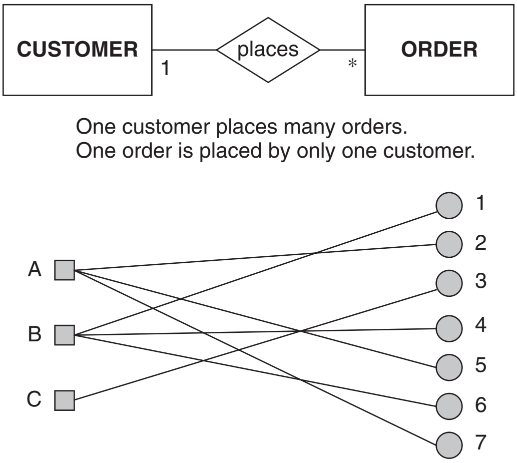

CUSTOMER
places
ORDER
1
*
One customer places many orders.
One order is placed by only one customer.
A
B
C
1
2
3
4
5
6
7

Figure 6-9 Relationship cardinality: one-to-many.

Observing from the CUSTOMER side, one instance of the object CUSTOMER is related to many instances of the other object ORDER. Again, observing from the ORDER side, one instance of object ORDER is related to only one instance of the other object CUSTOMER. So is the relationship one-to-many, or many-to-one? It appears to be both. Conventionally, this type of relationship is only referred to as a one-to-many relationship, and not the other way round.

Many-to-Many Relationship Consider an example of the relationship between the objects EMPLOYEE and PROJECT. Let us say that an employee may be assigned to several projects. Also, a project may have many employees. What is the nature of the association between instances of EMPLOYEE and PROJECT? Figure 6-10 presents this many-to-many relationship where the cardinality is *:*.

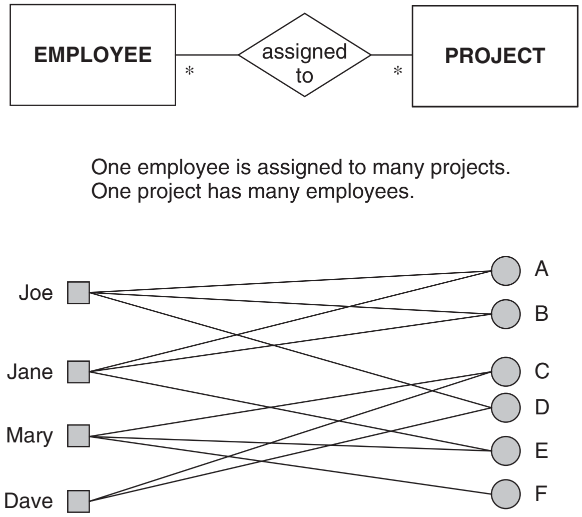

EMPLOYEE
*
assigned to
*
PROJECT
One employee is assigned to many projects.
One project has many employees.
Joe
Jane
Mary
Dave
A
B
C
D
E
F

Figure 6-10 Relationship cardinality: many-to-many.

Go over the association indicated between instances of the two objects. Look at the associations from the EMPLOYEE side. Employee Joe is assigned to projects A, B, and D. That is, one instance of EMPLOYEE is related to many instances of PROJECT. Now, observe the associations from the PROJECT side. Project B is associated with Joe and Jane. This means one instance of PROJECT is related to many instances of EMPLOYEE. Note the asterisks placed at both ends of the relationship line to indicate this many-to-many relationship.

Minimum and Maximum Cardinalities Go back and look at the cardinality indicators such as 1 or * placed on the relationship lines in the model diagram. Analyze the meaning of these indicators. What do they indicate? These indicators tell us about the number of instances of an object that can be related. Related to how many instances of the other object? Let us review the meanings from the above figures.

Figure 6-8: "1" placed closer to object INVOICE Indicates that a maximum of one instance of the object INVOICE can be related to one instance of the other object PAYMENT.

Figure 6-9: "" placed closer to object ORDER* Indicates that a maximum of many instances of the object ORDER can be related to one instance of the other object CUSTOMER.

Figure 6-10: "" placed closer to object EMPLOYEE* Indicates that a maximum of many instances of the object EMPLOYEE can be related to one instance of the other object PROJECT.

At this point, you realize that the cardinality indicators we have used so far really indicate the maximum cardinalities. Is there something known as a minimum cardinality in a relationship between objects? If so, what is the significance of minimum cardinality? First, let us formally define maximum and minimum cardinalities.

Maximum cardinality. Maximum number of instances of one object that are related to a single instance of another object.

Minimum cardinality. Minimum number of instances of one object that are related to a single instance of another object.

Figure 6-11 shows two relationships, one between SUPERVISOR and EMPLOYEE and the other between EMPLOYEE and PROJECT.

Review the cardinality indicators marked on the relationship line between SUPERVISOR and EMPLOYEE. Each set of indicators has two parts. The first part indicates the minimum cardinality; the second part indicates the maximum cardinality. Whenever cardinality indicators are shown as a set of two indicators, they denote the minimum and maximum cardinalities, respectively. If only one indicator is shown in a relationship, then assume it to be the maximum cardinality indicator, the minimum cardinality indicator being omitted. Interpret the cardinality indicators between SUPERVISOR and EMPLOYEE.

Indicators next to SUPERVISOR

Minimum 1: A minimum of one instance of SUPERVISOR associated with one instance of EMPLOYEE.

Maximum 1: A maximum of one instance of SUPERVISOR associated with one instance of EMPLOYEE.

Indicators next to EMPLOYEE

Minimum 1: A minimum of one instance of EMPLOYEE associated with one instance of SUPERVISOR.

Maximum *: A maximum of many instances of EMPLOYEE associated with one instance of SUPERVISOR.

What do these cardinality indicators really mean? Which aspects of real-world information do these indicators represent? Note carefully that the indicators next

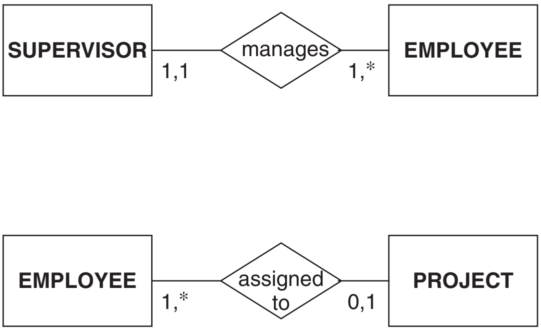


Figure 6-11 Relationships: maximum and minimum cardinalities.

to one object indicate the number of the instances of that object associated with one instance of the other object.

Looking at the above cardinality indicators, what can you say about the real-world information portrayed by the data model? How many employees does one supervisor can have under his or her supervision? For the answer, look at the cardinality indicators next to the other object, EMPLOYEE. What do you see? A minimum cardinality of 1 and a maximum cardinality of *. This means a supervisor must supervise at least one employee but may supervise many employees. Now ask the other question. How many supervisors does one employee report to? To answer this question, look at the cardinality indicators "1,1" next to the other object, SUPERVISOR. These indicators represent the real-world condition that each employee must report to one supervisor and an employee cannot report to more than one supervisor.

Continue with the other relationship in the figure between EMPLOYEE and PROJECT. A project must have at least one employee; furthermore, a project may have many employees. From the other point of view, an employee may be assigned to only one project and not more. Also, some employees may be associated with no project (note minimum cardinality of "0") at all.

Let us summarize maximum and minimum cardinalities with one more example involving three objects. In particular, note carefully how the minimum cardinality indicator expresses the optional or mandatory participation of object instances in a relationship. Figure 6-12 displays three objects, CUSTOMER, ORDER, and PRODUCT, and indicates their relationships.

CUSTOMER and ORDER are in a one-to-many relationship; ORDER and PRODUCT are in a many-to-many relationship. Note the minimum cardinality of "1" or "0." Read the explanation given in the figure and understand how "1" as the minimum cardinality indicates the mandatory nature of the relationship and how "0" as the minimum cardinality denotes its optional nature.

Aggregate Objects

Let us suppose you are modeling the information requirements for a dental office. Two objects in your data model will be PATIENT and SERVICE. Now consider

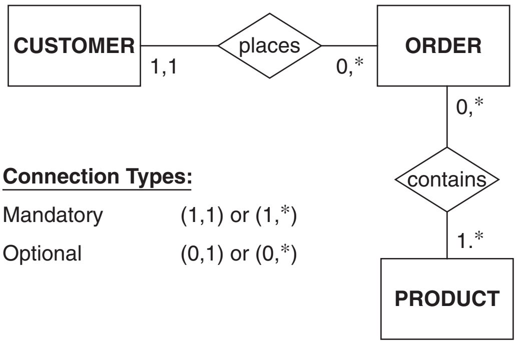


Figure 6-12 Optional and mandatory participation in relationship.

the instances of the object PATIENT. This object set contains all patients who come for dental service. The instances of the object SERVICE comprise all services received by patients at the dental office. These include services such as oral exams, X rays, amalgams, tooth extractions, prosthodontics, crowns, endodontics, and periodontics.

Typically, the cost for a certain service received by a particular patient depends on the type of insurance coverage the patient has. If patient Robert Bridges gets complete upper dentures, cost for him is $595, whereas patient Diane Silverstein is charged $475 for the same service. Different types of insurance policies cover the patients, hence the variation in the service costs.

Is *Cost* an attribute of the object SERVICE? You find that values of *Cost* are not dependent just on particular instances of this object alone. What are the values of *Cost*? The $595 and $475 indicated above are values of the attribute *Cost*. For complete upper dentures to Robert, the value of the *Cost* attribute is $595; for the same service to Diane, the value is $475. What does that mean? The values of the attribute *Cost* depend on pairs of instances from PATIENT and SERVICE put together as shown in the following examples:

(Robert Bridges, complete upper dentures)	$595
(Diane Silverstein, complete upper dentures)	$475
(John Rawlins, plastic crown)	$498
(Ruth Goldberg, plastic crown)	$325

Review the above examples. Robert, Diane, John, and Ruth are instances of the object PATIENT. Complete upper dentures and plastic crown are instances of the object SERVICE. What can you say about the pairs shown in parenthesis? How can you represent this kind of real-world information in your data model? You may consider these as instances of the two objects PATIENT and SERVICE put together—instances of an aggregate object comprising the two individual objects.

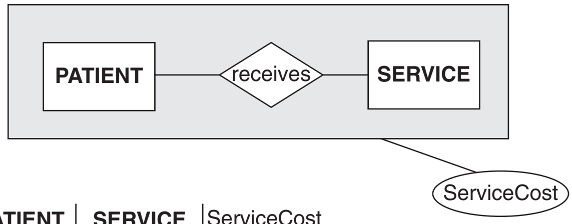
Figure 6-13 illustrates this concept of an aggregate or composite object. A relationship viewed as an object gives rise to the notion of an aggregate object.

An aggregate object behaves like any other object. A rectangular box represents an aggregate object also. The aggregate object PATIENT-SERVICE is shown as a large box surrounding the two objects whose relationship produces the aggregate.

Figure 6-14 presents the aggregate object called SHIPMENT. Here the aggregate is made up of three objects that participate in a three-way relationship. Note the larger box representing the aggregate object encompassing all the three participating objects.

From these two figures, note the following features of aggregate objects that are common to all types of objects:

Attributes. An aggregate object has specific attributes representing real-world information.



PATIENT	SERVICE	ServiceCost
Joe	Chest X-ray	75.00
Jane	Blood Test	95.00
Mary	Blood Test	65.00
Rob	MRI	325.00
Lisa	MRI	345.00

Figure 6-13 Example of an aggregate object.

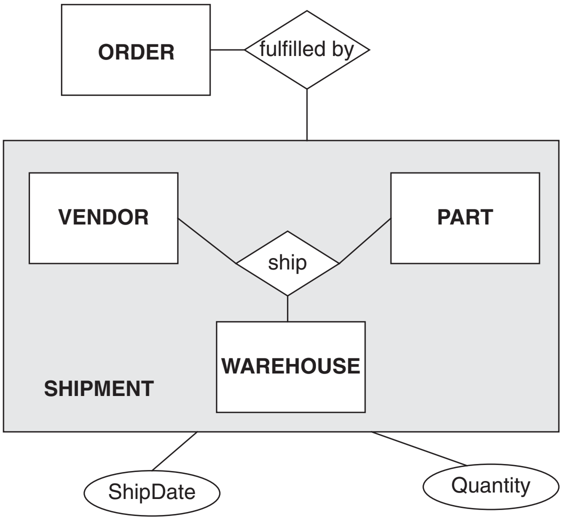


Figure 6-14 Aggregate object: three-way relationship.

Relationships. An aggregate object may have relationships with other single objects or other aggregate objects depending on the real-world information portrayed by the data model.

Under what circumstances do you have to show aggregate objects in your data model diagram? If you find that a certain characteristic such as cost or date depends on combinations of object instances, then an aggregate must be shown with that characteristic as an attribute of the aggregate object. Again, if you note that instances of some object are related to combinations of instances of other objects and not to instances of single objects, then you need an aggregate object to show the combination. Look at Figure 6-14 again for such a relationship.

Degrees of Relationships

Before we leave the topic of relationships, let us explore one final aspect of relationships. From the examples seen so far, you note that objects are related in twos or threes. That is, instances of two objects are associated with each other, or instances of three objects are associated to form a combination. Figure 6-15 indicates three types of relationships. Note the associations between instances of the same object, between instances between two objects, and among instances of three objects.

The degree of a relationship is the number of object sets that participate in the relationship. If three objects participate in a relationship, its degree is 3. Occasionally, the real-world information being modeled may require relationships of degrees 4, 5, or higher. Generally, the binary relationship is the most common type.

GENERALIZATION AND SPECIALIZATION

A true data model must reflect every aspect of real-world information. Do you agree? If there are peculiarities about certain business objects, then the model must represent those special conditions. If some relationships are different from regular relationships, then the data model must portray those special relationships. A realistic data model should display everything about the set of real-world information requirements. Frequently, you will find that some of the attributes and relationships are the same for more than one object. The object-based data modeling technique handles this special condition very well. Generalization and specialization of object sets is a powerful feature of this technique.

Take the case of modeling the real-world information for a medical center. One of the main business objects to be modeled is the PATIENT object. Think about

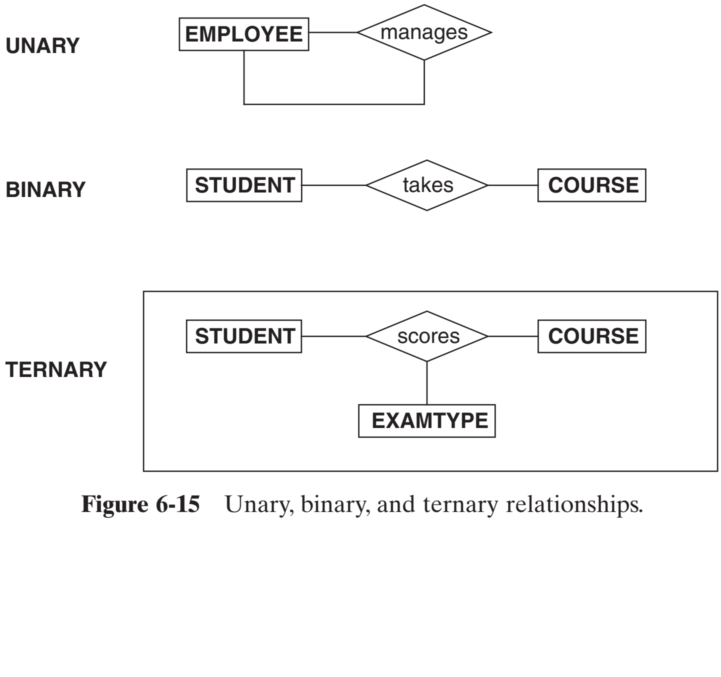


Figure 6-15 Unary, binary, and ternary relationships.

coming up with a model for this object. Modeling any object requires considerations of the possible and relevant attributes for the object. Also, you must consider the relationships that instances of this object have with instances of other objects. As you proceed to model the object PATIENT, you realize that there are inpatients, outpatients, and emergency room patients. Your model must include all these patient types. Now examine these patient types for attributes. You note that all three of these types of patients have common attributes such as PatientID, PatientName, Address, Diagnosis, and so on. But you also realize that each set of patients has attributes that are not common to the other two sets of patients. For example, inpatients have attributes such as AdmissionDate, DischargeDate, LengthOfStay, TypeOf Meals, and so on that are not shared by outpatients and ER patients. Furthermore, inpatients are related to another object in the model, namely, ROOM. They may be transferred from one room to another.

You see that there is something special about the three sets of patients in the way they share their attributes and in the manner in which some attributes are specific to each set. Clearly, all patients in the medical center cannot be modeled with one object PATIENT. Then what are the options? You can opt to model the patients with three separate objects INPATIENT, OUTPATIENT, and ERPATIENT. If you make this choice, then your model will repeat several attributes and perhaps relationships for each of the three objects.

Step back and look at the four objects PATIENT, INPATIENT, OUTPATIENT, and ERPATIENT. It appears that an object PATIENT is a supertype of object and that the other three are subtypes whose attributes and relations may be derived from the supertype object. You will find it best to use these four objects in your data model to truly represent the real-world information in the case of the medical center. Figure 6-16 explains the need for this method of representation in the model.

What you have noted is the concept of generalization and specialization in a data model. This concept enables special types of objects to be represented in a data model. As you examine the information requirements of any business, you will

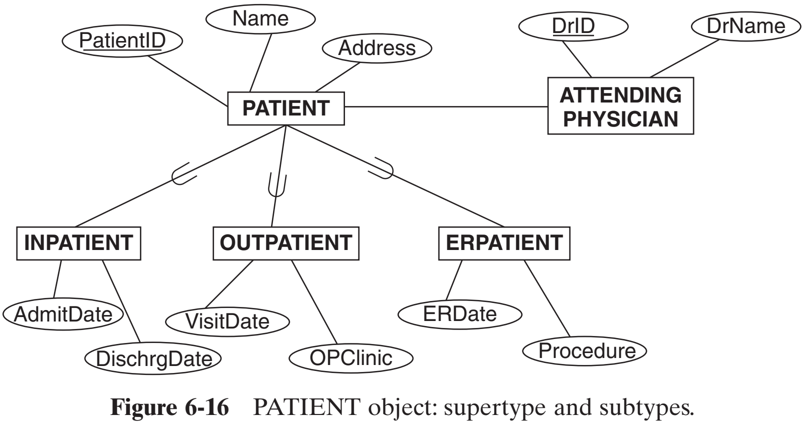


Figure 6-16 PATIENT object: supertype and subtypes.

observe these types of objects. Two main reasons warrant the indication of generalization and specialization in a data model:

- Although each subtype object may share most of its attributes with all the other subtypes, certain attributes may apply only to that subtype.
- Only the instances of certain subtype objects may participate in some relationships.

In practice, how do you note these special object types while modeling the information requirements? You may adopt the top-down or the bottom-up approach. You may look at several objects and realize that these may be subtypes of some supertype object. Conversely, you may examine an object and find that it would break down into subtype objects. Let us summarize these two routes:

Generalization. Create object boxes for every possible type of business object. Examine objects and see whether some of these may be subtypes of some other object. Suppress the differences between the subtypes, identify the common attributes, and generalize to define the supertype object.

Specialization. Create object boxes for only the high-level business objects. That is, ignore any possible variations in the set of attributes for instances within each high-level object. In the case of the medical center, first you would come up with the PATIENT object. Then examine the instances within the object and note the differences in the sets of attributes for the instances. Separate out the instances that possess the same set of instances as a special subtype object for PATIENT.

Supersets and Subsets

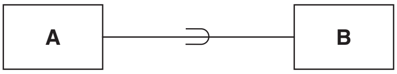
Object sets that are supertypes are also known as supersets; similarly, subtype object sets are also called subsets. Figure 6-17 presents a few examples of supersets and subsets.

Note each superset and the corresponding subsets. Taking any particular example, while modeling the information requirements you could model by the generalization method from the subsets or by the specialization method from the superset.

Generalization Hierarchy

Refer back to Figure 6-16 showing the subsets and the superset representing patients in a medical center. As certain attributes for a subset are derived from the superset, the superset and its subsets form a hierarchy in the arrangement of these objects in a data model. The hierarchical, top-down arrangement with the superset object box above the subset boxes gives an indication of how the superset provides the attributes common to the subsets in real-world situations.

Figure 6-16 shows two levels of the hierarchy, PATIENT at the higher level and the other three objects one level down. Sometimes, you will come across more



A is a generalization of B.
B is a specialization of A.

Superset	Subsets
EMPLOYEE	HOURLY EMPLOYEE SALARIED EMPLOYEE
PASSENGER	BUSINESS CLASS ECONOMY CLASS
PATIENT	INPATIENT OUTPATIENT
SHAREHOLDER	INDIVIDUAL INSTITUTIONAL
INSURANCE	AUTO LIFE

Figure 6-17 Supersets and subsets: examples.

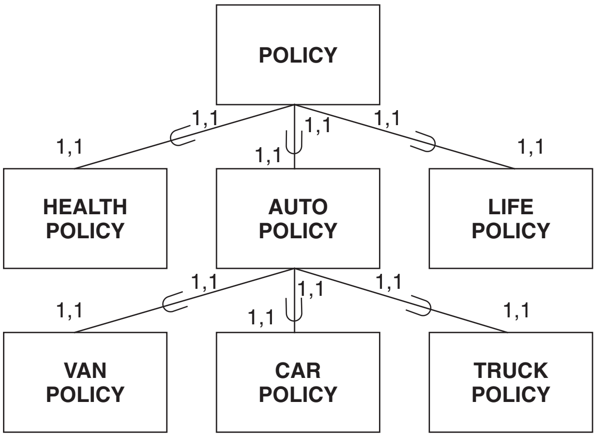


Figure 6-18 POLICY: generalization/specialization hierarchy levels.

than two levels in the generalization/specialization hierarchy. Figure 6-18 shows three levels in the hierarchy for POLICY, showing levels of subsets for insurance policies.

What about the instances in the superset and each of the subsets? The set of instances within the supertype object is a collection of all the instances in the lower-level subtype objects. If a particular instance is present in subtype AUTOPOLICY, then that instance also exists in the supertype POLICY. Note the cardinality indicator "1,1" between the supertype and each subtype object.

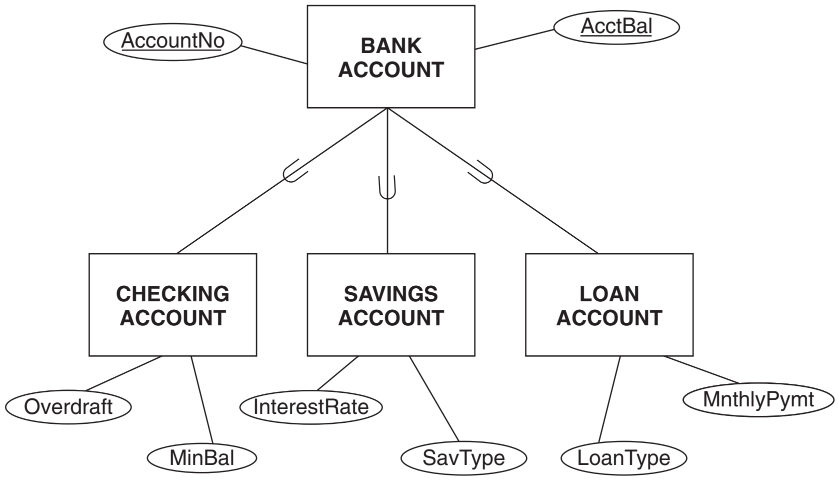


Figure 6-19 Subsets: inheritance of attributes.

Inheritance of Attributes

A significant feature of a superset and its subsets is the inheritance of the attributes by each subset from the superset. These are the attributes that are common to all the subsets. In the case of the objects for a medical center as discussed above, all the subsets share common attributes such as PatientID, PatientName, Address, and Diagnosis. Because the subsets share these attributes, there is no point in repeating these as attributes of each of the subsets. In a data model diagram, you may, therefore, show these as attributes of the superset. The principle of inheritance of attributes by the subsets from the superset implies that each of the subsets has these attributes. In addition, each subset may have other attributes specific only to that subset.

Figure 6-19 illustrates the principle of inheritance of attributes by the subsets. Note the common attributes shown at the superset level. Also, observe the attributes specific to individual subsets.

Inheritance of Relationships

Let us take an example of a company leasing vehicles to customers. A lease agreement covers a particular leasing arrangement with a customer. Examining the information requirements, you can come up with two initial objects, namely, VEHICLE and AGREEMENT. Of course, there would be other objects. But let us consider these two objects for now. Over a period of time, the same vehicle would relate to different lease agreements. That is, when a vehicle comes out of one lease agreement, it would be leased to another customer or to the same customer with a new lease agreement. You note that a direct relationship exists the objects AGREEMENT and VEHICLE.

Examine the instances of the object VEHICLE. Do all the instances have the same attributes? You quickly notice that cars, trucks, and vans that are leased have common attributes. More importantly, each of these three types has specific attrib-

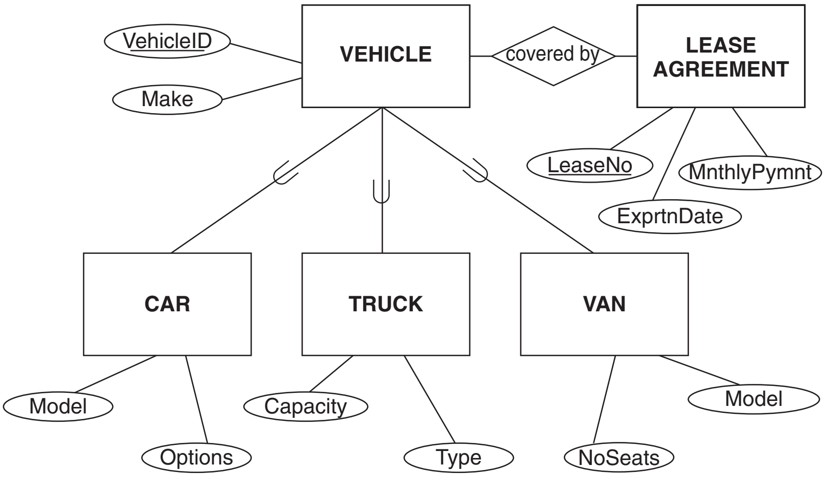


Figure 6-20 Subsets: inheritance of relationships.

utes not shared with the other two types. You have now come across the situation of superset and subsets in the information requirements. VEHICLE is the superset and CAR, TRUCK, and VAN are the subsets.

What about the association of the instances of VEHICLE with instances of AGREEMENT? In the same way, do instances of CAR have associations with instances of AGREEMENT? They do, because cars are covered by lease agreements. You note that if the superset VEHICLE has a relationship with another object, AGREEMENT, then its subset CAR also has the same relationship with the same object AGREEMENT.

Figure 6-20 illustrates this principle of inheritance of relationships by the subsets from the superset. Note that each of the subsets inherits the relationship with the object AGREEMENT.

Special Cases

In real-world situations, you will run into a few special cases of specialization. Usually, these special cases deal with the presence of particular instances in just one subset or in more than one subset. Let us examine these special cases. We will describe each special case with an example. Each example shows the links between instances in the superset and instances in the subsets. The notations to represent the special cases in a data model diagram vary. You need not be unnecessarily confused with variation in the notations. Please pay attention to the concepts; concepts are more important than notations.

Exclusive Subsets Figure 6-21 describes this special case. In this case, an instance of the superset can be an instance of only one of the subsets. The subsets are exclusive.

Nonexclusive Subsets Figure 6-22 shows nonexclusive subsets. In this case, an instance of the superset can be an instance of more than one subset. The subsets are nonexclusive.

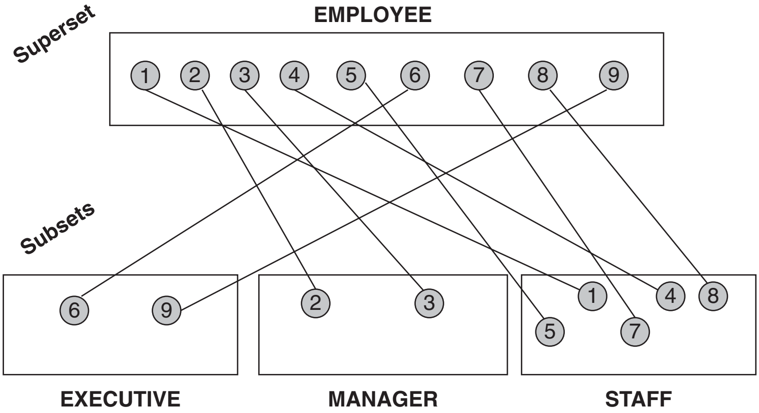


Figure 6-21 Exclusive subsets.

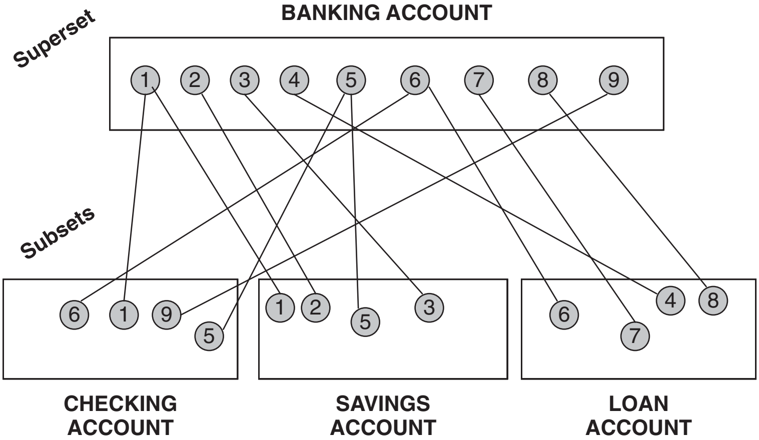


Figure 6-22 Nonexclusive subsets.

Total Specialization Figure 6-23 illustrates the special case in which every instance of the superset must be an instance of some subset. That means the list of subsets is total or complete. The model includes all possible subsets known at the time of design. Total specialization implies exhaustive specialization; all possible subsets are considered.

Partial Specialization Figure 6-24 shows an example of partial specialization. Here, some instances of the superset are not found in any of the subsets. That means that the list of subsets is not exhaustive; all possibilities are not accounted for in the subsets. The list of subsets is only partial.

In a particular data model, you are likely to see combinations of these special cases. The following are the possible combinations:

- Exclusive, total specialization
- Nonexclusive, total specialization

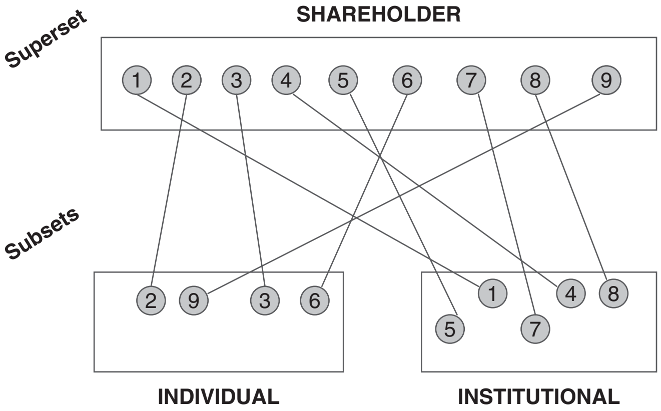


Figure 6-23 Total specialization.

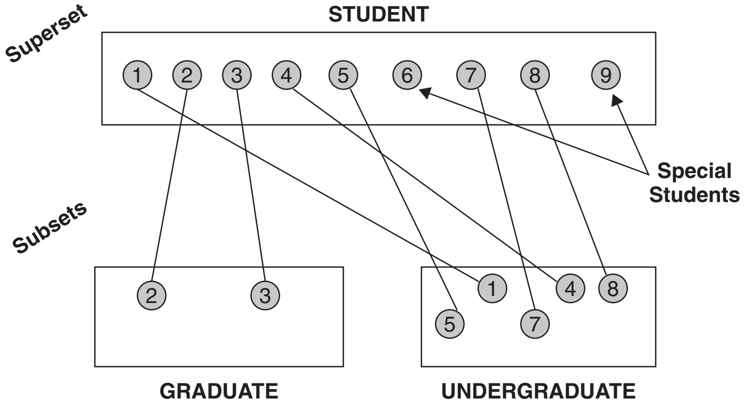


Figure 6-24 Partial specialization.

- Exclusive, partial specialization
- Nonexclusive, partial specialization

SPECIAL OBJECT TYPES AND RELATIONSHIPS

We have discussed various examples of objects and their relationships. The object sets you represent by a rectangular box in a data model diagram are all business objects relevant to the information requirements of the real world portrayed by the model. You have also noted, through many examples, that instances of one object may have associations with instances of another object. If so, you draw lines connecting the two associated objects, indicating the relationship by a diamond shape in between with the name of the relationship written inside the diamond. By now, you know all of this clearly.

We also went through special cases of objects that are related in a particular way. One object appeared to be contained within another or one object seemed to be inheriting attributes and relationships from another object. This is the concept of generalization and specialization. You have also noted examples of supersets and subsets.

We now want to consider a few remaining concepts about special object types and relationships. This would complete the types of objects and relationships you commonly come across in real-world information requirements. As you know, a data model must be a true replica of everything relevant in the real world.

Conceptual and Physical Objects

Assume that you are asked to model the information requirements for an airline company. Very quickly, you realize that AIRCRAFT must be one of the objects in your data model. Now examine the requirements of the company with regard to the object AIRCRAFT. The company's fleet consists of many aircraft. Each of these aircraft has a serial number, a given number of seats, the date it was placed in service, the chief mechanic responsible to service it, and so on. The company needs to keep track of all the planes in its fleet. Notice also that the company uses different types of aircraft like Boeing 747, MD 11, Airbus 321, and so on. The company needs to keep track of the aircraft types used in its fleet as well. What are Boeing 747, MD11, Airbus 321, and so on? Are these aircraft? No, these are types of aircraft, not the physical aircraft themselves. In this case of information requirements for the airline company, you find two types of related objects. One is the object AIRCRAFT and the other is the object AIRCRAFT TYPE.

Now consider the information requirements for a library. The library has to keep track of the individual copies of the books so that it will know which particular copy is out with a member. Each copy is marked with a call number. Furthermore, the library must also have a catalog of books available for members. These are not the actual copies of the books but the titles. So, in this case, you see the need for two related objects in your data model: BOOK and BOOK COPY. The two are not the same. The instances within each object are different. In the case of the object BOOK, the instances are the individual titles; for the object BOOK COPY the instances are individual copies of the books.

Just one more example. An appliance store needs to keep track of individual units of each appliance. Also, the store has to maintain a list of the types of appliances available in the store. Here you note the need for two related objects in the data model: APPLIANCE UNIT and APPLIANCE TYPE.

Physical Objects The objects like AIRCRAFT, BOOK COPY, and APPLIANCE UNIT are examples of physical objects. A physical object refers to tangible objects that you can see, touch, or feel. These are physical things. You need to have physical objects in your data model, whenever the information requirements call for keeping track of individual, physical things.

What about the instances within a physical object? Each instance represents a particular physical thing. The instances within the object AIRCRAFT are the physical aircraft in the company's fleet. If the company owns 100 aircraft, 100 instances exist within the object set. In the same way, if the library has a total inventory of

10,000 physical copies of various books, the object set BOOK COPY contains these 10,000 instances.

Conceptual Objects AIRCRAFT TYPE, BOOK, and APPLIANCE TYPE do not represent any tangible things. You cannot touch or see an AIRCRAFT TYPE. What you can see is a physical aircraft that is of the Boeing 747 type. So these are not examples of physical objects. The objects represented by these are conceptual; these are conceptual objects.

What kinds of instances does a conceptual object set contain? These instances are not physical things. Each instance within the object set BOOK refers to a particular title in the library. The library may have the book, *A Brief History of Time* by Stephen Hawking. This title will be an instance of the conceptual object BOOK. If the library holds four copies of this book, then you will have four corresponding instances in the physical object BOOK COPY.

Data Model Diagrams Figure 6-25 presents the data model diagrams for the three examples discussed.

Note the relationships between the conceptual object and the corresponding physical object in each example. Also, observe the cardinality indicators and understand the association of the instances.

Recursive Relationships

EMPLOYEE is a fairly common object in many real-world information requirements. Think about the instances for this object set. Every employee in the organization will be represented by an instance. In the set of instances, you will find workers who supervise other workers. An instance representing a supervisor associates with those instances of employees under his or her supervision. What you find here is instances of an object associating with instances of the same object. Associations recur within the same object. These are recursive associations, and this type

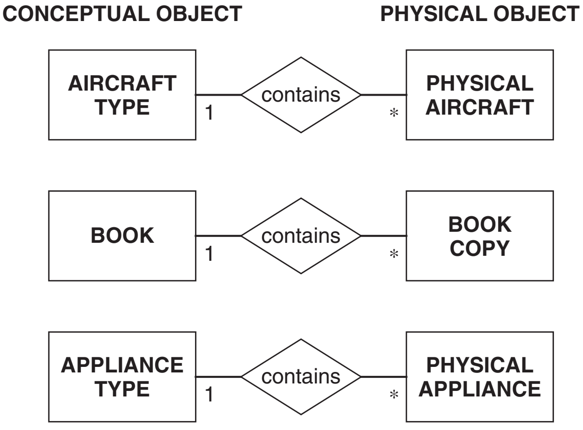


Figure 6-25 Data model diagrams: conceptual and physical objects.

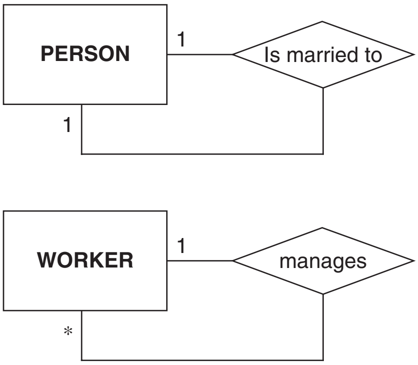


Figure 6-26 Recursive relationships.

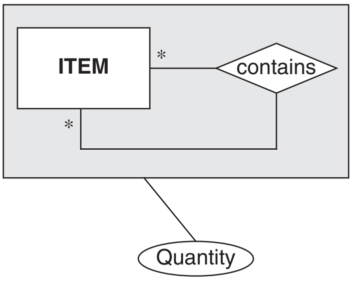


Figure 6-27 Assembly structure.

of association is a recursive relationship. Your data model must be able to indicate recursive relationships.

Figure 6-26 shows examples of recursive relationships.

Note how the relationship line from an object is linked back to itself. Cardinalities may vary with recursive relationships just like they do with regular relationships. Observe the different sets of cardinality indicators represented in the figure.

Assembly Structures

What is a bill of materials structure? You come across bill of materials processing (BOMP) in manufacturing. In manufacturing of automobiles, an assembly of a major part consists of several subassemblies; each subassembly in turn may be broken down further. Assembly structures are cases of recursive relationships in which the associations recur at many levels.

Figure 6-27 displays an example of an assembly structure.

Notice the attribute Quantity shown in the diagram for the object ITEM. This attribute Quantity does not represent quantities of individual instances of the object. Quantity indicates units for the various recursive associations. How many units are there for the combinations of item number 1 with item numbers 4,5, and 6? It is an attribute of an aggregate object consisting of just one object ITEM.

REVIEW OF OBJECT-BASED DATA MODEL

As we conclude our discussion, let us recapitulate what we covered on this important topic. Object-based data modeling is a natural, intuitive method of representing real-world information requirements. In the real world, most commonly you notice business objects. This modeling technique provides components to represent business objects, their characteristics, and their interactions.

Summary of Components

You can create a true data model for any set of real-world information requirements by using the following basic components, showing these with standard notations or symbols:

Object Set. To represent a business object.

Attribute. To represent an inherent or intrinsic characteristic of an object.

Identifier. An attribute or set of attributes whose values will uniquely identify individual instances of an object.

Relationship. Direct association between instances of objects.

Cardinality Indicators. Set of values to denote the minimum and maximum number of instances that may participate in relationships between objects.

Generalization/Specialization. To represent a superset and its subsets.

Comprehensive Data Model Diagram

Figure 6-28 presents a data model diagram for manufacturing operations.

Carefully note each component in the diagram. Study the various object sets, relationships, and cardinality indicators and also observe the special cases. Even though the data model represents many aspects of real-world information requirements, note that the whole model is constructed with only the basic components listed above.

CHAPTER SUMMARY

- Object-based data modeling based on both object technology and data modeling is an effective method focusing on business objects.
- The object-based data model is a generic model that has several advantages: universal model, true replica, free from restraints, easily transformable, and intuitive.
- The components or building blocks of the data model are object set, attribute, identifier, relationship, cardinality indicators, generalization/specialization.

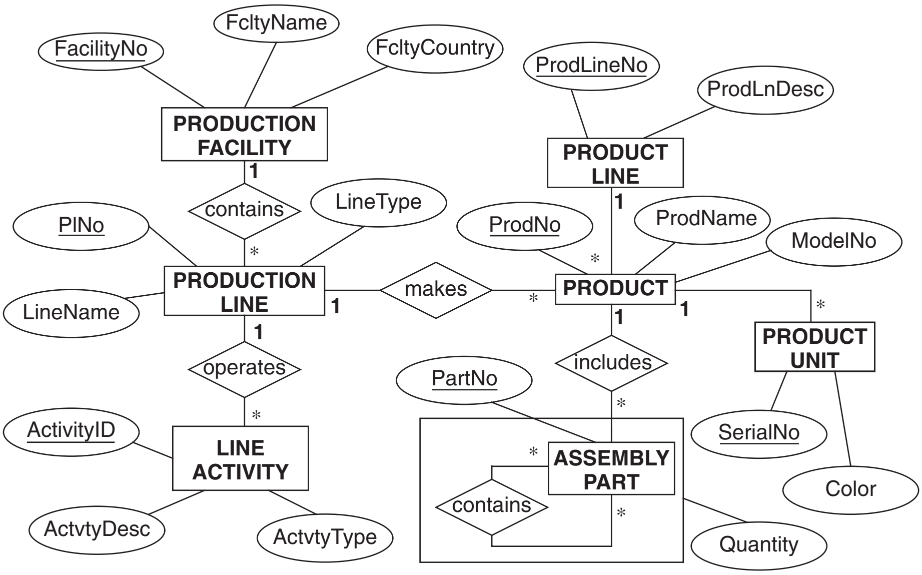


Figure 6-28 Data model diagram: manufacturing operations.

- With just these components, you can represent any collection of real-world information in the data model.
- Some object sets are subsets of other object sets. Generalization/specialization deals with supersets and subsets. Subsets inherit attributes and relationships from a superset.
- Physical objects represent physical things that can be seen or touched, for example, a copy of a book; conceptual objects indicate nontangible things, for example, "book."
- In a recursive relationship, instances of an object are associated with other instances of the same object.

REVIEW QUESTIONS

1. Explain how an object-based data model is a generic data model.
2. Name the components of the object-based data model and list the standard symbols or notations used to construct the model.
3. Distinguish between object sets and object instances. Give an example.
4. Describe the different types of objects, with an example for each type.
5. What is your understanding of null values for attributes? When are null values used?
6. "Relationships are really associations between instances of objects." Explain this with an example.
7. What do maximum and minimum cardinality indicators represent? What is an optional relationship?

8. What is an aggregate object? Give an example.
9. What is your understanding of total specialization? Explain with an example. Do you think total specialization is common in data models?
10. Describe the notion of conceptual objects and physical objects with an example.

EXERCISES

1. Indicate whether true or false:
   A. The object-based data modeling technique produces generic models.
   B. An identifier for an object set must not include more than three attributes.
   C. Cardinality indicators denote the number of object sets that are in a relationship.
   D. The maximum cardinality indicator denotes whether a relationship is optional or mandatory.
   E. The attributes of an object include only the intrinsic or inherent characteristics of that object.
   F. Two or more attributes of an object may share an attribute domain.
   G. The many-to-many relationship is actually a one-to-many relationship from either side of the relationship line.
   H. Aggregate objects may have attributes but not relationships.
   I. If some instances of the superset are not found in any of the subsets, then this is partial specialization.
   J. All recursive relationships are one-to-many relationships.
2. As a data modeler for a company manufacturing automobile parts, review any one of the business processes in your company. Identify the objects, relationships, and so on supporting this process. Create a partial data model diagram showing these components.
3. You are a member of the database project team responsible for analyzing information requirements for a grocery store and identifying the business objects to be included in the data model. Describe how you will proceed with your task. List the business objects you would identify.
4. Consider data modeling for an auction business. Customers in an auction business fall into different categories. Consignors bring items for sale; buyers purchase items at auction; property owners bring items for appraisal of values; subscribers subscribe to auction catalogs. Can you apply the principles of generalization/specialization to customers in this business? If so, describe the superset and the subsets. If not, explain the reasons.
5. You are a Senior Data Modeler for a local bank. Create a data model to cover the major banking operations. State your assumptions on the types of processes included in your data model.