Learning Objectives

Once you have mastered the material in this chapter you will be able to:

1. Understand how analysts use data dictionaries for analyzing data-oriented systems.
2. Create data dictionary entries for data processes, stores, flows, structures, and logical and physical elements of the systems being studied, based on DFDs.
3. Understand the concept of a repository for analysts’ project information and the role of CASE tools in creating them.
4. Recognize the functions of data dictionaries in helping users update and maintain information systems.

After successive levels of data flow diagrams are complete, systems analysts use them to help catalog the data processes, flows, stores, structures, and elements in a data dictionary. Of particular importance are the names used to characterize data items. When given an opportunity to name components of data-oriented systems, the systems analyst needs to work at making the name meaningful but exclusive of other existing data component names. This chapter covers the data dictionary, which is another method to aid in the analysis of data-oriented systems.

The Data Dictionary

The data dictionary is a specialized application of the kinds of dictionaries used as references in everyday life. The data dictionary is a reference work of data about data (that is, metadata), one that is compiled by systems analysts to guide them through analysis and design. As a document, the data dictionary collects and coordinates specific data terms, and it confirms what each term means to different people in the organization. The data flow diagrams covered in Chapter 7 are an excellent starting point for collecting data dictionary entries.

One important reason for maintaining a data dictionary is to keep clean data. This means that data must be consistent. If you store data about a man’s sex as “M” in one record, “Male” in a second record, and as the number “1” in a third record, the data are not clean. Keeping a data dictionary will help in this regard.

Automated data dictionaries (part of the CASE tools mentioned earlier) are valuable for their capacity to cross-reference data items, thereby allowing necessary
program changes to all programs sharing a common element. This feature supplants changing programs on a haphazard basis, or it prevents waiting until the program won’t run because a change has not been implemented across all programs sharing the updated item. Clearly, automated data dictionaries become important for large systems that produce several thousand data elements requiring cataloging and cross-referencing.

**NEED FOR UNDERSTANDING THE DATA DICTIONARY**

Many database management systems now come equipped with an automated data dictionary. These dictionaries can be either elaborate or simple. Some computerized data dictionaries automatically catalog data items when programming is done; others simply provide a template to prompt the person filling in the dictionary to do so in a uniform manner for every entry.

Despite the existence of automated data dictionaries, understanding what data compose a data dictionary, the conventions used in data dictionaries, and how a data dictionary is developed, are issues that remain pertinent for the systems analyst during the systems effort. Understanding the process of compiling a data dictionary can aid the systems analyst in conceptualizing the system and how it works. The upcoming sections allow the systems analyst to see the rationale behind what exists in automated data dictionaries.

In addition to providing documentation and eliminating redundancy, the data dictionary may be used to:

1. Validate the data flow diagram for completeness and accuracy.
2. Provide a starting point for developing screens and reports.
3. Determine the contents of data stored in files.
4. Develop the logic for data flow diagram processes.
5. Create XML (extensible markup language).

**THE DATA REPOSITORY**

Although the data dictionary contains information about data and procedures, a larger collection of project information is called a repository. The repository concept is one of the many impacts of CASE tools and may contain the following:

1. Information about the data maintained by the system, including data flows, data stores, record structures, elements, entities, and messages.
2. Procedural logic and use cases.
3. Screen and report design.
4. Data relationships, such as how one data structure is linked to another.
5. Project requirements and final system deliverables.
6. Project management information, such as delivery schedules, achievements, issues that need resolving, and project users.

The data dictionary is created by examining and describing the contents of the data flows, data stores, and processes, as illustrated by Figure 8.1. Each data store and data flow should be defined and then expanded to include the details of the elements it contains. The logic of each process should be described using the data flowing into or out of the process. Omissions and other design errors should be noted and resolved.

The four data dictionary categories—data flows, data structures, data elements, and data stores—should be developed to promote understanding of the data of the system. Procedural logic is presented in Chapter 9, entities are discussed in Chapter 13, and messages and use cases are presented in Chapters 2 and 18.
To illustrate how data dictionary entries are created, we use an example for World’s Trend Catalog Division. This company sells clothing and other items by mail order using a toll-free phone order system (or faxing the mail order form), and via the Internet using customized Web forms. Regardless of the origin of the order, the underlying data captured by the system are the same for all three methods.

The World’s Trend order form shown in Figure 8.2 gives some clues about what to enter into a data dictionary. First, you need to capture and store the name, address, and telephone number of the person placing the order. Then you need to address the details of the order: the item description, size, color, price, quantity, and so on. The customer’s method of payment must also be determined. Once you have done this, these data may be stored for future use. This example is used throughout this chapter to illustrate each part of the data dictionary.

DEFINING THE DATA FLOWS

Data flows are usually the first components to be defined. System inputs and outputs are determined from interviewing, observing users, and analyzing documents and other existing systems. The information captured for each data flow may be summarized using a form containing the following information:

1. ID, an optional identification number. Sometimes the ID is coded using a scheme to identify the system and the application in the system.
2. A unique descriptive name for this data flow. This name is the text that should appear on the diagram and be referenced in all descriptions using the data flow.
3. A general description of the data flow.
4. The source of the data flow. The source could be an external entity, a process, or a data flow coming from a data store.
5. The destination of the data flow (same items listed under the source).
6. An indication of whether the data flow is a record entering or leaving a file or a record containing a report, form, or screen. If the data flow contains data that are used between processes, it is designated as *internal*.
7. The name of the data structure describing the elements found in this data flow. For a simple data flow, it could be one or several elements.
8. The volume per unit of time. The data could be records per day or any other unit of time.

9. An area for further comments and notations about the data flow.

Once again we can use our World’s Trend Catalog Division example from Chapter 7 to illustrate a completed form. Figure 8.3 is an example of the data flow description representing the screen used to add a new CUSTOMER ORDER and to update the customer and item files. Notice that the external entity CUSTOMER is the source and that PROCESS 1 is the destination, providing linkage back to the data flow diagram. The checked box for “Screen” indicates that the flow represents an input screen. It could be any screen, such as a Web page, graphical user interface (GUI), mobile phone, or perhaps a mainframe screen. The detailed description of the data flow could appear on this form, or it could be represented as a data structure.

Data flows for all inputs and outputs should be described first, because they usually represent the human interface, followed by the intermediate data flows.
and the data flows to and from data stores. The detail of each data flow is described using elements, sometimes called fields, a data structure, or a group of elements.

A simple data flow may be described using a single element, such as a customer number used by an inquiry program to find the matching customer record.

DESCRIPTING DATA STRUCTURES

Data structures are usually described using algebraic notation. This method allows the analyst to produce a view of the elements that make up the data structure along with information about those elements. For instance, the analyst will denote whether there are many of the same element in the data structure (a repeating group), or whether two elements may exist mutually exclusive of each other. The algebraic notation uses the following symbols:

1. An equal sign (=) means “is composed of.”
2. A plus sign (+) means “and.”
3. Braces { } indicate repetitive elements, also called repeating groups or tables. There may be one repeating element or several in the group. The repeating group may have conditions, such as a fixed number of repetitions, or upper and lower limits for the number of repetitions.
4. Brackets [ ] represent an either/or situation. Either one element may be present or another, but not both. The elements listed between the brackets are mutually exclusive.
5. Parentheses ( ) represent an optional element. Optional elements may be left blank on entry screens and may contain spaces or zeros for numeric fields in file structures.
Figure 8.4 is an example of the data structure for adding a customer order at World’s Trend Catalog Division. Each NEW CUSTOMER screen consists of the entries found on the right side of the equal signs. Some of the entries are elements, but others, such as CUSTOMER NAME, ADDRESS, and TELEPHONE, are groups of elements or structural records. For example, CUSTOMER NAME is made up of FIRST NAME, MIDDLE INITIAL, and LAST NAME. Each structural record must be further defined until the entire set is broken down into its component elements. Notice that following the definition for the CUSTOMER ORDER screen are definitions for each structural record. Even a field as simple as the TELEPHONE NUMBER is defined as a structure so that the area code may be processed individually.
Structural records and elements that are used in many different systems are given a nonsystem-specific name, such as street, city, and zip, that does not reflect the functional area in which they are used. This method allows the analyst to define these records once and use them in many different applications. For example, a city may be a customer city, supplier city, or employee city. Notice the use of parentheses to indicate that (MIDDLE INITIAL), (APARTMENT), and (ZIP EXPANSION) are optional ORDER information (but not more than one). Indicate the OR condition by enclosing the options in square brackets and separating them with the symbol ‘\’.

**LOGICAL AND PHYSICAL DATA STRUCTURES**

When data structures are first defined, only the data elements that the user would see, such as a name, address, and balance due, are included. This stage is the logical design, showing what data the business needs for its day-to-day operations. As we learned from HCI, it is important that the logical design accurately reflect the mental model of how the user views the system. Using the logical design as a basis, the analyst then designs the physical data structures, which include additional elements necessary for implementing the system. Examples of physical design elements are the following:

1. Key fields used to locate records in a database table. An example is an item number, which is not required for a business to function but is necessary for identifying and locating computer records.
2. Codes to identify the status of master records, such as whether an employee is active (currently employed) or inactive. Such codes can be maintained on files that produce tax information.
3. Transaction codes are used to identify types of records when a file contains different record types. An example is a credit file containing records for returned items as well as records of payments.
4. Repeating group entries containing a count of how many items are in the group.
5. Limits on the number of items in a repeated group.
6. A password used by a customer accessing a secure Web site.

Figure 8.5 is an example of the data structure for a CUSTOMER BILLING STATEMENT, one showing that the ORDER LINE is both a repeating item and a
structural record. The ORDER LINE limits are from 1 to 5, indicating that the customer may order from one to five items on this screen. Additional items would appear on subsequent orders.

The repeating group notation may have several other formats. If the group repeats a fixed number of times, that number is placed next to the opening brace, as in 12 {Monthly Sales}, where there are always 12 months in the year. If no number is indicated, the group repeats indefinitely. An example is a table containing an indefinite number of records, such as Customer Master Table = {Customer Records}.

The number of entries in repeating groups may also depend on a condition, such as an entry on the Customer Master Record for each item ordered. This condition could be stored in the data dictionary as {Items Purchased} 5, where 5 is the number of items.

DATA ELEMENTS

Each data element should be defined once in the data dictionary and may also be entered previously on an element description form, such as the one illustrated
in Figure 8.6. Characteristics commonly included on the element description
form are the following:

1. Element ID. This optional entry allows the analyst to build automated data
dictionary entries.
2. The name of the element. The name should be descriptive, unique, and based
on what the element is commonly called in most programs or by the major
user of the element.
3. Aliases, which are synonyms or other names for the element. Aliases are names
used by different users in different systems. For example, a CUSTOMER
NUMBER may also be called a RECEIVABLE ACCOUNT NUMBER or a
CLIENT NUMBER.
5. Whether the element is base or derived. A base element is one that is initially
keyed into the system, such as a customer name, address, or city. Base elements
must be stored in files. Derived elements are created by processes as the result
of a calculation or a series of decision-making statements.
6. The length of an element. Some elements have standard lengths. In the United
States, for example, lengths for state name abbreviations, zip codes, and tele-
phone numbers are all standard. For other elements, the lengths may vary, and
the analyst and user community must jointly decide the final length based on
the following considerations:
   a. Numeric amount lengths should be determined by figuring the largest
      number the amount will probably contain and then allowing reasonable
      room for expansion. Lengths designated for totals should be large enough
      to accommodate the sum of the numbers accumulated in them.
   b. Name and address fields may be given lengths based on the following
      table. For example, a last name field of 11 characters will accommodate
      98 percent of the last names in the United States.
   c. For other fields, it is often useful to examine or sample historical data
      found in the organization to determine a suitable field length.

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Percentage of Data That Will Fit (U.S.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Name</td>
<td>11</td>
<td>98</td>
</tr>
<tr>
<td>First Name</td>
<td>18</td>
<td>95</td>
</tr>
<tr>
<td>Company Name</td>
<td>20</td>
<td>95</td>
</tr>
<tr>
<td>Street</td>
<td>18</td>
<td>90</td>
</tr>
<tr>
<td>City</td>
<td>17</td>
<td>99</td>
</tr>
</tbody>
</table>

If the element is too small, the data that need to be entered will be truncated.
The analyst must decide how that will affect the system outputs. For example,
if a customer’s last name is truncated, mail would usually still be delivered; if
an email address is truncated, however, it will be returned as not found.
7. The type of data—numeric, date, alphabetic, varchar, or character, which is
sometimes called alphanumeric or text data. Varchar data may contain any
number of characters, up to a limit set by the database software. When using
varchar, specifying the length is optional. Several of these formats are shown
in Figure 8.7. Character fields may contain a mixture of letters, numbers,
and special characters. If the element is a date, its format—for example,
MMDDYYYY—must be determined. If the element is numeric, its storage
type should be determined.

Personal computer formats, such as currency, number, or scientific, depend
on how the data will be used. Number formats are further defined as integer,
## Data Type and Meaning

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>A value of 1 or 0, a true/false value</td>
</tr>
<tr>
<td>Char, varchar, text</td>
<td>Any alphanumeric character</td>
</tr>
<tr>
<td>Datetime, smalldatetime</td>
<td>Alphanumeric data, several formats</td>
</tr>
<tr>
<td>Decimal, numeric</td>
<td>Numeric data that are accurate to the least significant digit; can contain a whole and decimal portion</td>
</tr>
<tr>
<td>Float, real</td>
<td>Floating-point values that contain an approximate decimal value</td>
</tr>
<tr>
<td>Int, smallint, tinyint</td>
<td>Only integer (whole digit) data</td>
</tr>
<tr>
<td>Currency, money, smallmoney</td>
<td>Monetary numbers accurate to four decimal places</td>
</tr>
<tr>
<td>Binary, varbinary, image</td>
<td>Binary strings (sound, pictures, video)</td>
</tr>
<tr>
<td>Cursor, timestamp, uniqueidentifier</td>
<td>A value that is always unique within a database</td>
</tr>
<tr>
<td>Autonumber</td>
<td>A number that is always incremented by one when a record is added to a database table</td>
</tr>
</tbody>
</table>

## Formatting Character and Meaning

<table>
<thead>
<tr>
<th>Formatting Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>May enter or display/print any character</td>
</tr>
<tr>
<td>9</td>
<td>Enter or display only numbers</td>
</tr>
<tr>
<td>Z</td>
<td>Display leading zeros as spaces</td>
</tr>
<tr>
<td>,</td>
<td>Insert commas into a numeric display</td>
</tr>
<tr>
<td>.</td>
<td>Insert a period into a numeric display</td>
</tr>
<tr>
<td>/</td>
<td>Insert slashes into a numeric display</td>
</tr>
<tr>
<td>-</td>
<td>Insert a hyphen into a numeric display</td>
</tr>
<tr>
<td>V</td>
<td>Indicate a decimal position (when the decimal point is not included)</td>
</tr>
</tbody>
</table>

Long integer, single precision, double precision, and so on. There are many other types of formats used with PC systems. Unicode is a standardized coding system for defining graphic symbols, such as Chinese or Japanese characters. Unicode is described in greater detail in a later chapter. There are three standard formats for mainframe computers: zoned decimal, packed decimal, and binary. The zoned decimal format is used for printing and displaying data. The packed decimal format is commonly used to save space on file layouts and for elements that require a high level of arithmetic to be performed on them. The binary format is suitable for the same purposes as the packed decimal format but is less commonly used.

8. Input and output formats should be included, using special coding symbols to indicate how the data should be presented. These symbols and their uses are illustrated in Figure 8.8. Each symbol represents one character or digit. If the same character repeats several times, the character followed by a number in parentheses indicating how many times the character repeats is substituted for the group. For example, XXXXXXXX would be represented as X(8).

9. Validation criteria for ensuring that accurate data are captured by the system. Elements are either discrete, meaning they have certain fixed values, or continuous, with a smooth range of values. Here are common editing criteria:

   a. A range of values is suitable for elements that contain continuous data. For example, in the United States a student grade point average may be from...
0.00 through 4.00. If there is only an upper or lower bound to the data, a limit is used instead of a range.
b. A list of values is indicated if the data are discrete. Examples are codes representing the colors of items for sale in World’s Trend’s catalog.
c. A table of codes is suitable if the list of values is extensive (for example, state abbreviations, telephone country codes, or U.S. telephone area codes.)
d. For key or index elements, a check digit is often included.

10. Any default value the element may have. The default value is displayed on entry screens and is used to reduce the amount of keying that the operator may have to do. Usually, several fields in each system have default values. When using GUI lists or drop-down lists, the default value is the one currently selected and highlighted. When using radio buttons, the option for the default value is selected, and when using check boxes, the default value (either “yes” or “no”) determines whether or not the check box will have an initial check in it.

11. An additional comment or remarks area. This might be used to indicate the format of the date, special validation that is required, the check digit method used (explained in Chapter 15), and so on.

Data element descriptions such as CUSTOMER NUMBER may be called CLIENT NUMBER elsewhere in the system (perhaps old code written with this alias needs to be updated).

Another kind of data element is an alphabetic element. At World’s Trend Catalog Division, codes are used to describe colors: for example, BL for blue, WH for white, and GR for green. When this element is implemented, a table will be needed for users to look up the meanings of these codes. (Coding is discussed further in Chapter 15.)

**DATA STORES**

All base elements must be stored in the system. Derived elements, such as the employee year-to-date gross pay, may also be stored in the system. Data stores are created for each different data entity being stored. That is, when data flow base elements are grouped together to form a structural record, a data store is created for each unique structural record.

Because a given data flow may only show part of the collective data that a structural record contains, you may have to examine many different data flow structures to arrive at a complete data store description.

Figure 8.9 is a typical form used to describe a data store. The information included on the form is as follows:

1. The data store ID. The ID is often a mandatory entry to prevent the analyst from storing redundant information. An example would be D1 for the CUSTOMER MASTER.
2. The data store name, which is descriptive and unique.
3. An alias for the table, such as CLIENT MASTER for the CUSTOMER MASTER.
4. A short description of the data store.
5. The file type, either computer or manual.
6. The format designates whether the file is a database table or if it has the format of a simple flat file. (File formats are detailed in Chapter 13.)
7. The maximum and average number of records on the file as well as the growth per year. This information helps the analyst to predict the amount of disk space required for the application and is necessary for hardware acquisition planning.
8. The file or data set name specifies the file name, if known. In the initial design stages, this item may be left blank. An electronic form produced using Visible Analyst is shown in Figure 8.10. This example shows that the CUSTOMER MASTER is stored on a computer in the form of a database with a maximum number of 45,000 records. (Records and the keys used to sort the database are explained in Chapter 13.)

9. The data structure should use a name found in the data dictionary, providing a link to the elements for this data store. Alternatively, the data elements could be described on the data store description form or on the CASE tool screen for the data store. Primary and secondary keys must be elements (or a combination of elements) found in the data structure. In the example, the CUSTOMER NUMBER is the primary key and should be unique. The CUSTOMER NAME, ZIP, and YEAR-TO-DATE AMOUNT PURCHASED are secondary keys used to control record sequencing on reports and to locate records directly. (Keys are discussed in Chapter 13.) Comments are used for information that does not fit into any of the above categories. They may include update or backup timing, security, or other considerations.
CREATING THE DATA DICTIONARY

Data dictionary entries may be created after the data flow diagram has been completed, or they may be constructed as the data flow diagram is being developed. The use of algebraic notation and structural records allows the analyst to develop the data dictionary and the data flow diagrams using a top-down approach. For instance, the analyst may create a Diagram 0 data flow after the first few interviews and, at the same time, make the preliminary data dictionary entries. Typically, these entries consist of the data flow names found on the data flow diagram and their corresponding data structures.

After conducting several additional interviews with users to learn the details of the system and the ways they interact with it, the analyst will expand the data flow diagram and create the child diagrams. The data dictionary is then modified to include the new structural records and elements gleaned from further interviews, observation, and document analysis.

Each level of a data flow diagram should use data appropriate for the level. Diagram 0 should include only forms, screens, reports, and records. As child diagrams are created, the data flow into and out of the processes becomes more and more detailed, including structural records and elements.

Figure 8.11 illustrates a portion of two data flow diagram levels and corresponding data dictionary entries for producing an employee paycheck. Process 5, found on Diagram 0, is an overview of the production of an EMPLOYEE PAYCHECK. The corresponding data dictionary entry for EMPLOYEE RECORD shows the EMPLOYEE NUMBER and four structural records, the view of the data...
### FIGURE 8.11

Two data flow diagrams and corresponding data dictionary entries for producing an employee paycheck.

---

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Employee Record = Employee Number + Personal Information + Wage Information + Current Pay Information + Year-to-Date Information</td>
</tr>
<tr>
<td>Timefile</td>
<td>Timefile Record = Employee Number + Employee Name + Hours Worked</td>
</tr>
<tr>
<td>Produce</td>
<td>Employee Paycheck = Employee Number + Employee Name + Address + Current Pay Amounts + Year-to-Date Figures</td>
</tr>
<tr>
<td>Hours Worked</td>
<td>Wage Information = Rate of Pay + Number of Dependents</td>
</tr>
</tbody>
</table>

obtained early in the analysis. Similarly, TIMEFILE RECORD and the EMPLOYEE PAYCHECK are also defined as series of structures.

It is important that the data flow names on the child data flow diagram are contained as elements or structural records in the data flow on the parent process. Returning to the example, WAGE INFORMATION (input into process 5.3, COMPUTE CURRENT PAY AMOUNTS) is a structural record contained in the EMPLOYEE RECORD (input to process 5). Similarly, GROSS PAY (output from process 5.3.4, a lower-level process not shown in the figure) is contained in the structural record CURRENT PAY AMOUNTS (output from the parent process 5.3, COMPUTE CURRENT PAY AMOUNTS).

### ANALYZING INPUT AND OUTPUT

An important step in creating the data dictionary is to identify and categorize system input and output data flow. Input and output analysis forms contain the following commonly included fields:

1. A descriptive name for the input or output. If the data flow is on a logical diagram, the name should identify what the data are (for example, CUSTOMER INFORMATION). If the analyst is working on the physical design or if the user has explicitly stated the nature of the input or output, however, the name should include that information regarding the format. Examples are CUSTOMER BILLING STATEMENT and CUSTOMER DETAILS INQUIRY.
2. The user contact responsible for further details clarification, design feedback, and final approval.
3. Whether the data is input or output.
4. The format of the data flow. In the logical design stage, the format may be undetermined.
5. Elements indicating the sequence of the data on a report or screen (perhaps in columns).
6. A list of elements, including their names, lengths, and whether they are base or derived, and their editing criteria.

Once the form has been completed, each element should be analyzed to determine whether the element repeats, whether it is optional, or whether it is mutually exclusive of another element. Elements that fall into a group or that regularly combine with several other elements in many structures should be placed together in a structural record.

These considerations can be seen in the completed Input and Output Analysis Form for World’s Trend Catalog Division (see Figure 8.12). In this example of a CUSTOMER BILLING STATEMENT, the CUSTOMER FIRST NAME, CUSTOMER LAST NAME, and CUSTOMER MIDDLE INITIAL should be grouped together in a structural record.

**DEVELOPING DATA STORES**

Another activity in creating the data dictionary is developing data stores. Up to now, we have determined what data needs to flow from one process to another. This information is described in data structures. The information, however, may be stored in numerous places, and in each place the data store may be different. Whereas data flows represent data in motion, data stores represent data at rest.

For example, when an order arrives at World’s Trend (see Figure 8.13), it contains mostly temporary information, that is, the information needed to fill that particular order, but some information might be stored permanently. Examples of the latter include information about customers (so catalogs can be sent to them) and information about items (because these items will appear on many other customers’ orders).

Data stores contain information of a permanent or semipermanent (temporary) nature. An ITEM NUMBER, DESCRIPTION, and ITEM COST are examples of information that is relatively permanent. So is the TAX RATE. When the ITEM COST is multiplied by the TAX RATE, however, the TAX CHARGED is calculated (or derived). Derived values do not have to be stored in a data store.
When data stores are created for only one report or screen, we refer to them as “user views,” because they represent the way that the user wants to see the information.

**USING THE DATA DICTIONARY**

The ideal data dictionary is automated, interactive, online, and evolutionary. As the systems analyst learns about the organization’s systems, data items are added to the data dictionary. On the other hand, the data dictionary is not an end in itself and must never become so. To avoid becoming sidetracked with the building of a complete data dictionary, the systems analyst should view it as an activity that parallels systems analysis and design.

To have maximum power, the data dictionary should be tied into a number of systems programs so that when an item is updated or deleted from the data dictionary, it is automatically updated or deleted from the database. The data dictionary becomes simply a historical curiosity if it is not kept current.

The data dictionary may be used to create screens, reports, and forms. For example, examine the data structure for the World’s Trend ORDER PICKING

---

**FIGURE 8.12**

An example of an input/output analysis form for World’s Trend Catalog Division.
ANALYZING SYSTEMS USING DATA DICTIONARIES

CHAPTER 8

Customer Master = Customer Number + Customer Name + Address + Telephone + Corporate Credit Card Number + Expiration Date

Item Master = Item Number + Price + Quantity on Hand

Order Record = Customer Number + Catalog Number + Order Date + (Available Order Items) + Merchandise Total + (Tax) + Shipping and Handling + Order Total + Method of Payment + (Credit Card Type) + (Credit Card Number) + (Expiration Date)

Available Order Items = Item Number + Quantity Ordered + Quantity Shipped + Current Price

Method of Payment = [Check, Charge, Money Order]

Credit Card Type = [World’s Trend, American Express, MasterCard, Visa]

SLIP in Figure 8.14. Because the necessary elements and their lengths have been defined, the process of creating physical documents consists of arranging the elements in a pleasing and functional way using design guidelines and common sense. Repeating groups become columns, and structural records are grouped together on the screen, report, or form. The report layout for the World’s Trend ORDER PICKING SLIP is shown in Figure 8.15. Notice that FIRST NAME and LAST NAME are grouped together in NAME, and that QUANTITY (PICKED and ORDERED), SECTION, SHELF NUMBER, ITEM NUMBER, ITEM DESCRIPTION, SIZE, and COLOR form a series of columns, because they are the repeating elements.

The data structure and elements for a data store are commonly used to generate corresponding computer language source code, which is then incorporated into computer programs. The data dictionary may be used in conjunction with a data flow diagram to analyze the system design, detecting flaws and areas that need clarification. Some considerations are:

1. All base elements on an output data flow must be present on an input data flow to the process producing the output. Base elements are keyed and should never be created by a process.
2. A derived element must be created by a process and should be output from at least one process into which it is not input.

3. The elements that are present in a data flow coming into or going out of a data store must be contained in the data store.

If begun early, a data dictionary can save many hours of time in the analysis and design phases. The data dictionary is the one common source in the organization for answering questions and settling disputes about any aspect of data definition. An up-to-date data dictionary can serve as an excellent reference for maintenance efforts on unfamiliar systems. Automated data dictionaries can serve as references for both people and programs.

**USING DATA DICTIONARIES TO CREATE XML**

Extensible markup language (XML) is a language that can be used to exchange data between businesses or between systems within a business. It is similar to HTML,
the markup language used to create Web pages, but is more powerful. HTML is concerned primarily with formatting a document; XML addresses the problem of sharing data when users have different computer systems and software or different database management systems (for example, one company using Oracle and another using IBM’s DB2). If everyone used the same software or database management system, there would be little need for XML.

Once an XML document has been created, the data may be transformed into a number of different output formats and displayed in many different ways, including printed output, Web pages, output for a handheld device, and portable document format (PDF) files. Thus, the document’s data content is separated from the output format. The XML content is defined once as data and then transformed as many times as necessary.

The advantage of using an XML document is that the analyst may select only the data that an internal department or external partner needs to have in order to function. This helps to ensure the confidentiality of data. For example, a shipping company may receive only the customer name, the address, the item number, and the quantity to ship, but not credit card information or other financial data. This efficient approach also cuts down on information overload.

XML therefore is a way to define, sort, filter, and translate data into a universal data language that can be used by anyone. XML may be created from databases, a form, or software programs, or it may be keyed directly into a document, text editor, or XML entry program.
The data dictionary is an ideal starting point for developing XML content. The key to using XML is creating a standard definition of the data. This is accomplished by using a set of tags or data names that are included before and after each data element or structure. The tags become the metadata, or data about the data. Data may be further subdivided into smaller elements and structures until all elements are defined. XML elements may also include attributes, an additional piece of data included within the tag that describes something about the XML element.

Figure 8.16 illustrates a data dictionary containing customer, order, and payment information. The overall collection of customers is included in what is called the root element, customers. An XML document may contain only one root element, so it is often the plural of the data contained in the XML document.
Each customer may place many orders. The structure is defined in the two left columns, and the XML code appears on the right. CUSTOMER, as you can see, consists of a NAME, ADDRESS, CURRENT BALANCE, multiple ORDER INFORMATION entries, and a PAYMENT. Some of these structures are further subdivided.

The XML document tends to mirror the data dictionary structure. The first entry (other than an XML line identifying the document) is <customer>, which defines the entire collection of customer information. The less than (<) and greater than (>) symbols are used to identify tag names (similar to HTML). The last line of the XML document is a closing tag, </customer>, signifying the end of the customer information.

Customer is defined first and contains an attribute, the customer number. There is often a discussion about whether data should be stored as an element or an attribute. In this case, they are stored as an attribute.

The name tag, <name>, is defined next because it is the first entry in the data dictionary. NAME is a structure consisting of LAST NAME, FIRST NAME, and an optional MIDDLE INITIAL. In the XML document, this structure starts with <name> and is followed by <lastname>, <firstname>, and <middle_initial>. Because spaces are not allowed in XML tag names, an underscore is typically used to separate words. The closing </name> tag signifies the end of the group of elements. Using a structure such as name saves time and coding if the transformation displays the full name. Each of the child elements will be on one line separated by a space. Name also contains an attribute, either I for individual or C for corporation.

Indentation is used to show which structures contain elements. Note that <address> is similar to <customer>, but when we get to <order_information> there is a big difference.

There are multiple entries for <order_information>, each containing an <order_number>, <order_date>, <shipping_date>, and <total>. Because the payment is made either by check or credit card, only one of these may be present. In our example, payment is by check. The dates have an attribute called format that indicates whether the date appears as month, day, year; year, month, day; or day, month, year. If a credit card is used to make a payment, a TYPE attribute contains either an M, V, A, D, or an O indicating the type of credit card (MasterCard, Visa, and so on).

**XML DOCUMENT TYPE DEFINITIONS**

Often the element structure of XML content is defined using a document type definition (DTD). A DTD is used to determine whether the XML document content is valid, that is, whether it conforms to the order and type of data that must be present in the document. The DTD is easy to create and well supported by standard software. Once the DTD has been completed, it may be used to validate the XML document using standard XML tools. The DTD is easier to create if a data dictionary has been completed, since the analyst has worked with users and made decisions on the structure of the data.

Figure 8.17 illustrates the document type definition for the Customer XML document. Keywords, such as !DOCTYPE, indicating the start of the DTD, must be in capital letters. !ELEMENT describes an element, and !ATTLIST describes an attribute, listing the element name followed by the attribute name. An element that has the keyword #PCDATA, for parsed character data, is a primitive element, not further defined. An element that has a series of other elements
within parenthesis means that they are child elements and must be in the order listed. The statement `<!ELEMENT name (lastname, firstname, middle_initial?)>` means that the name must have the last name followed by the first name followed by the middle initial. The question mark after “middle_initial” means that the element is optional and may be left out of the document for a particular customer. A plus sign means that there are one or more repeatable elements. Customers must contain at least one customer tag but could contain many customer tags. An asterisk means that there is zero or more of the elements. Each customer may have zero to many orders. A vertical bar separates two or more child elements that are mutually exclusive. Payment contains by either check or credit card.

The attribute list definition for a customer number contains a keyword ID (in uppercase letters). This means that the attribute number must appear only once in the XML document as an attribute for an element with an ID and that it is somewhat similar to a primary key. The difference is that, if the document had several different elements, each with an ID attribute, the given ID (C15008 in this example) could appear only once. An ID must start with a letter or an underscore and cannot be solely a number. The reason behind putting the customer number as an ID is to ensure that it is not repeated in a longer document. The keyword
#REQUIRED means that the attribute must be present. A keyword of #IMPLIED means that the attribute is optional. A document may also have an IDREF attribute, which links one element with another that is an ID. The ORDER tag has a customer_number attribute defined as an IDREF, and the value C15008 must be present in an ID somewhere in the document. An attribute list containing values in parentheses means that the attribute must contain one of the values. A DTD definition `<!ATTLIST credit_card type (M|V|A|D|O) #REQUIRED>`, means that the credit card type must be either an M, V, A, D, or O.

**XML SCHEMAS**

A schema is another, more precise way to define the content of an XML document. Schemas may include the exact number of times an element may occur as well as the type of data within elements, such as character or numeric values, including the length of the element, limits on the data, and the number of places to the left and right of a decimal number.

A data dictionary is an excellent starting point for developing an XML document and a document type of definition or schema. The advantage of using XML to define data is that, in the XML format, data are stored in a pure text format and not dependent on any proprietary software. The document may be easily validated and transformed into many different output formats.

Industry groups or organizations may be involved in defining an industry-specific XML structure so that all involved parties understand what the data mean. This is very important when an element name may have several meanings. An example is “state,” which may mean a postal state abbreviation or the state of an order or account. Examples of industry-specific XML document type definitions and schemas may be found at www.xml.org.

**SUMMARY**

Using a top-down approach, the systems analyst uses data flow diagrams to begin compiling a data dictionary, which is a reference work containing data about data, or metadata, on all data processes, stores, flows, structures, and logical and physical elements in the system being studied. One way to begin is by including all data items from data flow diagrams.

A larger collection of project information is called a repository. CASE tools permit the analyst to create a repository that may include information about data flows, stores, record structures, and elements; about procedural logic screen and report design; and about data relationships. A repository can also contain information about project requirements and final system deliverables; and about project management information.

Each entry in the data dictionary contains the item name, an English description, aliases, related data elements, the range, the length, encoding, and necessary editing information. The data dictionary is useful in all phases of analysis, design, and ultimately documentation, because it is the authoritative source on how a data element is used and defined by users in the system. Many large systems feature computerized data dictionaries that cross-reference all programs in the database using a particular data element. The data dictionary can also be used to create XML that enables businesses with different systems, software, or database management systems to exchange data.
“You’re really doing very well. Snowden says you’ve given him all sorts of new ideas for running the new department. That’s saying quite a lot, when you consider that he has a lot of his own ideas. By now I hope you’ve had a chance to speak with everyone you would like to: certainly Snowden himself, Tom Ketcham, Daniel Hill, and Mr. Hyatt.

“Mr. Hyatt is an elusive soul, isn’t he? I guess I didn’t meet him until well into my third year. I hope you get to find out about him much sooner. Oh, but when you do get to see him, he cuts quite a figure, doesn’t he? And those crazy airplanes. I’ve almost been conked on the head by one in the parking lot. But how can you get angry, when it’s The Boss who’s flying it? He’s also got a secret—or should I say private—oriental garden off his office suite. No, you’ll never see it on the building plans. You have to get to know him very well before he’ll show you that, but I would wager it’s the only one like it in Tennessee and maybe in the whole U.S. He fell in love with the wonderful gardens he saw in Southeast Asia as a young man. It goes deeper than that, however. Mr. Hyatt knows the value of contemplation and meditation. If he has an opinion, you can be sure it has been well thought through.”

HYPERCASE QUESTIONS

1. Briefly list the data elements that you have found on three different reports produced at MRE.
2. Based on your interviews with Snowden Evans and others, list the data elements that you believe you should add to the Management Unit’s project reporting systems to better capture important data on project status, project deadlines, and budget estimates.
3. Create a data dictionary entry for a new data store, a new data flow, and a new data process that you are suggesting based on your response to Question 2.
4. Suggest a list of new data elements that might be helpful to Jimmie Hyatt but are clearly not being made available to him currently.

In HyperCase, you can look at the data dictionary kept at MRE.
KEYWORDS AND PHRASES

- base element
- binary format
- data dictionary
- data element
- data structure
- derived element
- document type definition (DTD)
- extensible markup language (XML)
- ID
- IDREF
- packed decimal
- physical data structure
- repeating group
- repeating item
- repository
- schema
- structural record
- system deliverables
- varchar
- zoned decimal

REVIEW QUESTIONS

1. Define the term *data dictionary*. Define *metadata*.
2. What are four reasons for compiling a complete data dictionary?
3. What information is contained in the data repository?
4. What is a structural record?
5. List the eight specific categories that each entry in the data dictionary should contain. Briefly give the definition of each category.
6. What are the basic differences among data dictionary entries prepared for data stores, data structures, and data elements?
7. Why are structural records used?
8. What is the difference between logical and physical data structures?
9. Describe the difference between base and derived elements.
10. How do the data dictionary entries relate to levels in a set of data flow diagrams?
11. List the four steps to take in compiling a data dictionary.
12. Why shouldn’t compiling the data dictionary be viewed as an end in itself?
13. What are the main benefits of using a data dictionary?
14. What does extensible markup language (XML) describe?
15. What is a document type definition?
16. How does a document type definition help to ensure that an XML document contains all necessary elements?
17. When should attributes be used in an XML document?
18. What does an ID attribute ensure?
19. What does an IDREF attribute validate?

PROBLEMS

1. Based on Figure 7.EX1 in Chapter 7, Joe, one of your systems analysis team members, made the following entry for the data dictionary used by Marilyn’s Tours:

   DATA ELEMENT = TOURIST* * * PAYMENT
   ALIAS = TOURIST PAY
   CHARACTERS = 12–24
   RANGE = $5.00–$1,000
   VARIABLES = $5.00, $10.00, $15.00 up to $1,000, and anything in between in dollars and cents.
TO CALCULATE = TOTAL COST OF ALL TOURS, ANY APPLICABLE N.Y. STATE TAX, minus any RESERVATION DEPOSITS made.

a. Is this truly a data element? Why or why not?
b. Rewrite the data dictionary entry for TOURIST PAYMENT, reclassifying it if necessary. Use the proper form for the classification you choose.

2. Sue Kong, the systems analyst, has made significant progress in understanding the data movement at Shanghai Megabank. To share what she has done with other members of her team as well as the head of regional operations, she is composing a data dictionary.
   a. Write an entry in Sue’s data dictionary for three of the data flows in regional banking. Be as complete as possible.
   b. Write an entry in Sue’s data dictionary for three of the data stores in regional banking. Be as complete as possible.

3. Jorge Alvarez, the manager of the bookstore that your systems analysis team has been working with to build a computerized inventory system, thinks that one of your team members is making a nuisance of himself by asking him extremely detailed questions about data items used in the system. For example, he asks, “Jorge, how much space, in characters, does the listing of an ISBN take?”
   a. What are the problems created by going directly to the manager with questions concerning data dictionary entries? Use a paragraph to list the problems you can see with your team member’s approach.
   b. In a paragraph, explain to your team member how he can better gather information for the data dictionary.

4. Michael Bush owns a store specializing in travel gear and clothes. Manufacturers have their own coding, but there are many manufacturers. Set up data elements for six different travel hats from three different suppliers.

5. Michael (from Problem 4) also assembles packages of camping kits. Each kit is a group of separate products that are sold as a package. Each package (called a PRODUCT) is built using many parts, which vary from product to product. Interviews with the head parts clerk have resulted in a list of elements for the PRODUCT PART Web page, showing which parts are used in...
the manufacture of each product. A prototype of the PRODUCT-PART Web page is illustrated in Figure 8.EX1. Create a data structure dictionary entry for the PRODUCT-PART.

6. Analyze the elements found on the PRODUCT-PART Web page and create the data structure for the PRODUCT MASTER and the PART MASTER data stores.

7. Which of the elements on the PRODUCT-PART Web page are derived elements?

8. The Pacific Holiday Company arranges cruise vacations of varying lengths at several locations. When customers call to check on the availability of a cruise, a CRUISE AVAILABILITY INQUIRY, illustrated in Figure 8.EX2, is used to supply them with information. Create the data dictionary structure for the CRUISE AVAILABILITY INQUIRY.

9. List the master files that would be necessary to implement the CRUISE AVAILABILITY INQUIRY.

10. The following ports of call are available for the Pacific Holiday Company:

    - Apia
    - Pago Pago
    - Bora Bora
    - Moorea
    - Nuku Hiva
    - Papeete
    - Raiatea
    - Napier
    - Auckland
    - Wellington
    - Christ Church
    - Dunedin

Create the PORT OF CALL element. Examine the data to determine the length and format of the element.

11. Raul Esparza, the ecommerce manager for Moonlight Mugs, a company that sells customized coffee mugs, would like to send information to another company that maintains the warehouse and provides shipping services. Order information is obtained from a secure Web site, including customer number, name and address, telephone number, email address, product number and quantity, as well as credit card information. There may be several different products shipped on one order. The shipping company handles items for other small businesses as well. Define an XML document that will include only the information that the shipping company needs to ship goods to the customer.
12. Once the order in Problem 11 has been shipped, the shipping company sends information back to Moonlight Mugs, including the customer name and address, shipper tracking number, data shipped, quantity ordered, quantity shipped, and quantity backordered. Define an XML document that will include the information sent to Moonlight Mugs.

13. Create a document type definition for Problem 11.

14. Western Animal Rescue is a nonprofit organization that supports the fostering and adoption of animals, such as cats, dogs, and birds. People can register to adopt animals. Others register and add animals for adoption. Create the data dictionary structure representing a person registering to adopt an animal. Include name, address (street, city, state or province, zip or mailing code), telephone number, email address, date of birth, current pets (type, breed, age of pet), and references. Each person may have multiple pets and must have at least three references. References must include name, address, telephone number, email address, and how they know the person registering to adopt an animal. Be sure to include notation for repeating elements and optional elements.

15. Define the length, the type of data, and the validation criteria for each of the elements in Problem 14.

16. List the data stores that would be required to implement the person registering in Problem 14.

17. Create an XML document with sample data for one person registering to adopt an animal.

**GROUP PROJECTS**

1. Meet with your group and use a CASE tool or a manual procedure to develop data dictionary entries for a process, data flow, data store, and data structure based on the data flow diagrams you completed for Maverick Transport in the Chapter 7 group exercises. As a group, agree on any assumptions necessary to make complete entries for each data element.

2. Your group should develop a list of methods to help you make complete data dictionary entries for this exercise as well as for future projects. For example, study existing reports, base them on new or existing data flow diagrams, and so on.

**SELECTED BIBLIOGRAPHY**


We can use the data flow diagrams we completed to create data dictionary entries for all data flow and data stores,” Chip says to Anna at their next meeting. Each of these components has a composition entry in the repository. The records created for the computer system are thus linked directly to the data flow diagram components that describe data.

Anna and Chip meet to divide the work of creating records and elements. “I’ll develop the data dictionary for the software portion of the system,” Anna says. “Good thing I enjoy doing the hardware,” Chip kids her good-naturedly.

Records, or data structures, are created first. They may contain elements, the basic building blocks of the data structure, and they may also contain other records in them called structural records. Visible Analyst also maintains relationships among graph components, records, and elements that may be used for analysis and reporting.

Using information from interviews and the prototype screens, Anna starts to create the Software records. Because the output of a system will determine what data need to be both stored and obtained via data entry screens, the starting point is the output data flow SOFTWARE INSTALLATION LIST. This prototype identifies some of the elements that should be stored in the SOFTWARE MASTER:

SOFTWARE INVENTORY NUMBER
VERSION NUMBER
NUMBER OF CDs
CAMPUS LOCATION
TITLE
HARDWARE INVENTORY NUMBER
ROOM LOCATION

Other output prototype reports and screens are also examined. Additional elements are obtained from the ADD SOFTWARE prototype screen. These elements are arranged into a logical sequence for the SOFTWARE MASTER file. The following standards for arranging elements within a record are used:

1. The major key element that uniquely identifies the record. An example is the SOFTWARE INVENTORY NUMBER.
2. Descriptive information, such as TITLE, VERSION NUMBER, and PUBLISHER.
3. Information that is periodically updated, such as NUMBER OF COPIES.
4. Any repeating elements, such as HARDWARE INVENTORY NUMBER, denoting the machines on which the software has been installed.

Next, the SOFTWARE MASTER record is created using the Visible Analyst repository. The description screen for creating a record is shown in Figure E8.1. (Note: This screen may differ from the data structure screen in your copy of Visible Analyst. To view the screen that is in the same format, click the Options menu and then click so there is a check in front of Classical User Interface.) Notice the entry area for an alias, or a different name for the record, used by a different user group. Because each user may refer to the same record by a different name, all such names should be documented, resulting in enriched communication among users.
Each element or structural record needs to be defined as part of the whole record, and it is entered in the Composition area. If the element or structural record is a repeating group, the name is enclosed in curly brackets ({}), and the number of times it repeats is placed in front of the name. If the data are keys, a code is put in brackets ([ ]) in front of the name. The symbol [pk] represents a primary key. The symbol [akn] represents an alternate key, where n is 1, 2, 3 and so on and defines each different key or group of fields that, when combined, make a secondary key. When a group of fields makes up a secondary key, that key is called a concatenated key.

Examine the SOFTWARE MASTER. It contains a primary key of SOFTWARE INVENTORY NUMBER and a concatenated secondary key of TITLE, OPERATING SYSTEM, and VERSION NUMBER.

Visible Analyst allows you to easily describe each structural record or element composing the larger record. Anna places the cursor in each name in the Composition area and clicks the Jump button. Further record and element screens are displayed and detailed information is entered.

“This is great!” Anna thinks to herself. “It’s so easy to enter the details, and by using this method, I won’t accidentally forget to describe an element.”
Chip is also impressed with the simplicity of creating the data dictionary. Following a process similar to Anna’s, he creates a record description for the COMPUTER MASTER. It contains a table of five internal boards and two structural records, PERIPHERAL EQUIPMENT and MAINTENANCE INFORMATION. The Composition area for entering element or record names is a scroll region, meaning that more lines may be keyed than will fit in the display area. As entries are added to the bottom of the region, top entries scroll out of the area.

As elements are added to the record, Chip decides to describe each in detail. The element description screen for the HARDWARE INVENTORY NUMBER is shown in Figure E8.2. Observe the areas for entering element attributes. Several aliases may be included along with a definition. A Notes area contains any other useful information about the element. Chip and Anna employ this area to enter further edit criteria and other useful notation. The description for the HARDWARE INVENTORY NUMBER details how this number is used to keep physical track of the machines.

Clicking on the Physical Characteristics tab displays a second screen for the HARDWARE INVENTORY NUMBER, illustrated in Figure E8.3. It contains an area showing within which structures the element is contained, as well as an area for the

![Figure E8.2](image-url)  
**Figure E8.2**  
Element description screen, HARDWARE INVENTORY NUMBER.
type of data, the length, and the picture used to describe how the data are formatted. Each such picture is a coded entry, similar to those used in programming languages. Examples of some of the codes are as follows:

- **9** Represents numeric data: Only numbers may be entered when prototyping.
- **A** Alphabetic: Only alphabetic characters may be entered.
- **X** Alphanumeric: Any characters may be entered.
- **Z** Zero suppression: Replace leading zeros with spaces.
- **$** Dollar sign: Replace leading zeros with a dollar sign.

Chip is careful to include complete entries for these areas, including any default values and whether the entry may be null or not.

Anna and Chip repeat this process for all elements found on each record. This effort is time consuming but worthwhile. After the first few records are created, it becomes easier to create the remaining record structures. Visible Analyst has a search feature that provides lists of the elements contained in the design.

“I think that we’ve designed a complete set of elements,” Chip says at a checkpoint meeting.
“Yes,” replies Anna. “There are reports that will show us the details of the data structures and help us to spot duplications and omissions. Let’s put Visible Analyst to work producing record layouts for us.”

The Reports feature was used to print record layouts for all master data stores.

**RECORD AND ELEMENT ANALYSIS**

“No let’s really put the power of Visible Analyst to use,” Anna says. “Let’s see how well we’ve really designed our data.”

“What do you mean?” Chip asks.

“I’ve been studying the analysis features contained in Visible Analyst, and there’s a wealth of options for checking our design for consistency and correctness,” Anna replies. “The first step is to use the Reports feature to produce a summary report of the elements we’ve added. Then we can examine the list for duplications and redundancy.”

Figure E8.4 is an example of a portion of the element summary report displayed using Microsoft Internet Explorer. Analysts would examine the contents carefully and look for redundancy, or elements defined more than once. These redundancies are usually easy to spot because the list is sorted by element name. The elements HARDWARE INVENTORY NUMBER and HARDWARE NUMBER and the elements SOFTWARE INVENTORY NUMBER and SOFTWARE NUM appear to be duplicate elements. Other duplicates, such as ROOM LOCATION and LOCATION, are harder to spot.

“Next we should use the No Location References option, which shows all the elements that are not included on any record,” says Anna.
“This is terrific!” exclaims Chip. “This No Location References shows design work that needs to be completed. We should produce this report for all the design components.”

The elements were either added to other structures or deleted as duplicates. Producing the No Location References report a second time revealed no further isolated elements.

“Well, I guess that wraps up the data portion of the system design,” Chip says.

“Guess again,” replies Anna. “We’ve only begun to analyze. The Report Query feature will provide us with a lot of design information, both for analysis and documentation.”

The analysts select a report called Def Entities without Composition as their first choice. The report shows entries that are data store or data structure and do have a composition entry. The output shows that there are no records in error. The next report query is Elements without Pictures, and it shows all elements that do not have pictures defined for them. A last report that Chip and Anna create is called Undefined Elements, indicating all elements that have not been defined; that is, they exist in the repository as a name only, but with no physical characteristics.

“I’m really impressed with this analysis,” Chip says. “Since correcting the errors in our design, I’ve come to realize how easy it is to feel confident that the design has been completed when there are discrepancies and omissions still needing our attention.”

“We’re not finished yet. There are some useful matrices that will provide documentation for any changes that may be made in the future. Let’s produce the Data Elements versus Data Structures matrix, which shows records and their elements,” Anna suggests.

The Report feature has the ability to produce reports as well as matrices in a grid representation. It shows all elements and the data structures in which they are contained. This matrix is used to access the effect of changing an element by showing which corresponding data structures must be changed.

The next matrix created is the Diagram Location Matrix, showing all data stores and the diagrams in which they are located. This information is useful if a change needs to be made to the data store, because it will indicate where programs and documentation need to be changed.

A final matrix is the Composition Matrix, showing all data elements and the data stores in which they are contained. This matrix gives Chip and Anna a picture of which elements may be stored redundantly, that is, in several data stores rather than one.

“There are many other reports and matrices that would be useful for us to produce,” Anna says. “Some of these should be used later for documentation and tracking any proposed changes. I’m really pleased with what we’ve accomplished.”

EXERCISES

E-1. Use Visible Analyst to view the COMPUTER MASTER data store. Jump to the data structure and browse the elements and structural records.

E-2. Print the SOFTWARE MASTER record using the Report feature.

The exercises preceded by a web icon indicate value-added material is available from the Web site at www.prenhall.com/kendall. Students can download a sample Visible Analyst Project and a Microsoft Access database that can be used to complete the exercises. Visible Analyst software can be packaged with this text for an additional fee.
E-3. Use the **Jump** button to move to the **Software Record Structure**. Delete the following elements:
- ACTIVE SOFTWARE CODE
- INSTALLATION COMPUTER
- SOFTWARE EXPERT

E-4. Modify the SOFTWARE CHANGES record, supplying changes to the SOFTWARE MASTER record. The modifications are as follows:
   a. Add a [pk], for primary key, in front of the SOFTWARE INVENTORY NUMBER.
   b. Add the following elements: COMPUTER BRAND, COMPUTER MODEL, MEMORY REQUIRED, MONITOR REQUIRED, PRINTER REQUIRED, SITE LICENSE, and NUMBER OF COPIES.

E-5. Modify the COMPUTER ADD TRANSACTION record, which contains new computer records to be placed on the COMPUTER MASTER data store.
   a. Insert the BRAND NAME and MODEL above the SERIAL NUMBER.
   b. Place the CAMPUS LOCATION and ROOM LOCATION after the SERIAL NUMBER.
   c. Add the following elements at the bottom of the list: HARD DRIVE 1, HARD DRIVE 2, and CD-RW.
   d. Delete the INTERNAL BOARDS element, which will be determined after the computer installation.

E-6. Modify the INSTALLED SOFTWARE TRANSACTION, which is used to update the SOFTWARE MASTER and to produce the SOFTWARE INSTALLATION LISTING. Delete the TITLE and VERSION NUMBER, because they may be obtained from the SOFTWARE MASTER and are redundant keying. Add the HARDWARE INVENTORY NUMBER, specifying the installation computer. Delete the CAMPUS LOCATION and ROOM LOCATION, because they are elements of the installation computer.

E-7. View the alias entry for the SOFTWARE MASTER TABLE.

E-8. Modify the INSTALLED SOFTWARE data store. Add the composition record INSTALLED SOFTWARE TRANSACTION. The index elements are SOFTWARE INVENTORY NUMBER and HARDWARE INVENTORY NUMBER.

E-9. Define the data store SOFTWARE LOG FILE. This file is used to store information on the new software records, plus the date, time, and user ID of the person entering the record. Index elements are SOFTWARE INVENTORY NUMBER, TITLE, VERSION (a concatenated key), and SOFTWARE CATEGORY.

E-10. Define the data store PENDING COMPUTER ORDERS. This file is created when a purchase order is made for ordering new computers, and it is updated by the computer system. Place a comment in the **Notes** area stating that the average number of records is 100. Index elements are PURCHASE ORDER NUMBER and a concatenated key consisting of BRAND NAME and MODEL.

E-11. View the entry for the SOFTWARE RECORD data flow. Click **Jump** with the cursor in the **Composition** area and examine the SOFTWARE MASTER record. Click **Back** to return to the data flow description screen.

E-12. Modify the SOFTWARE UPGRADE INFORMATION data flow. The composition record is SOFTWARE UPGRADE INFORMATION.

E-13. Modify the SOFTWARE CROSS-REFERENCE REPORT data flow. The composition record is SOFTWARE CROSS-REFERENCE REPORT.
E-14. Modify the data flow entity for INSTALL UPDATE. This flow updates the COMPUTER MASTER record with installation information. Its data structure is INSTALL UPDATE RECORD. Include a comment that it processes about 50 records per month in updating the COMPUTER MASTER.

E-15. Use the INSTALL UPDATE data flow to jump to (and create) the INSTALL UPDATE RECORD. Provide a definition based on information supplied in the previous problem. Enter the following elements:

HARDWARE INVENTORY NUMBER (primary key)
CAMPUS LOCATION
ROOM
INTERNAL BOARDS (occurs 5 times)
HARD DRIVE 2
PRINTER
MAINTENANCE INTERVAL
DATE INSTALLED

E-16. Create the data flow description for the SOFTWARE INSTALLATION LIST. This flow contains information on specific software packages and the machines on which the software should be installed. The composition should include the SOFTWARE INSTALLATION LISTING, a data structure.

E-17. Use the SOFTWARE INSTALL LIST to jump to (and therefore create) the SOFTWARE INSTALLATION LISTING. The elements on the listing are as follows:

SOFTWARE INVENTORY NUMBER
TITLE
VERSION NUMBER
HARDWARE INVENTORY NUMBER
CAMPUS LOCATION
ROOM LOCATION

E-18. Modify and print the element HARDWARE SUBTOTAL. Change the type to Numeric, the length to 6,2, and the picture to Z, ZZZ, ZZ9.99.

E-19. Modify and print the MONITOR NAME element, the result of a table lookup using a monitor code. The type should be Character, the length 30, and the Picture X(30).

E-20. Modify and print the DEPARTMENT NAME element. Create an alias of STAFF DEPARTMENT NAME. In the Notes area, enter the following comment: Table of codes: Department Table. The type should be Character, the length 25, and the picture X(25).

E-21. Create the following element descriptions. Use the values supplied in the table. Create any alternate names and definitions based on your understanding of the element.

<table>
<thead>
<tr>
<th>Name</th>
<th>PURCHASE ORDER NUMBER</th>
<th>PROBLEM DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Character</td>
<td>Character</td>
</tr>
<tr>
<td>Length</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>Picture</td>
<td>99999999</td>
<td>X(70)</td>
</tr>
<tr>
<td>Name</td>
<td>TOTAL</td>
<td>NEXT PREVENTIVE</td>
</tr>
<tr>
<td></td>
<td>COMPUTER</td>
<td>MAINTENANCE</td>
</tr>
<tr>
<td></td>
<td>COST</td>
<td>DATE</td>
</tr>
</tbody>
</table>
E-22. Use the Repository Reports feature to produce the following reports and matrices, either by printing the reports or by previewing them using your Web browser. The selection criteria from the Repository Reports dialogue box are listed, separated with a slash (/). Explain in a paragraph where the information produced may be effectively used.

a. Data Flow/Cross-Reference Listing/Data Element/Entire Project
b. Data Flow/Cross-Reference Listing/Data Structure/Entire Project
c. Record Contains Element (One Level) Matrix
d. Data Flow/Single-Entry Listing/Software Master—Normalized
e. Data Flow/Diagram Location Matrix/Data Stores versus Diagrams
f. Data Flow/Composition Matrix/Data Elements versus Data Flows
g. Data Flow/Composition Matrix/Data Elements versus Data Structures
h. Data Flow/Composition Matrix/Data Element versus Data Stores

E-23. Use the Report Query feature to produce the following reports. Explain in a sentence what information the report is providing you with.

a. The Undefined Elements report
b. The Elements without Pictures report
c. The Coded Elements report
d. The Any Item with Components report

E-24. Print a summary report for all data flow components that do not have a description. (Hint: Click the No Descriptive Info. radio button.)

E-25. Print a summary report for all data flow components that are not on a diagram. (Hint: Click the No Location References radio button.)

E-26. Print a detailed report for all elements. Include only the physical information and the values and meanings. (Hint: Click the Fields button and then the Invert button and select the fields that you want printed.) Why would this report be useful to the analyst?