Data Flows and Processes

11/12/2015

Contents

Context Diagram

The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. To draw a context diagram, you start by placing a single process symbol in the center of the page.

The symbol represents the entire information system, and you identify it as **process 0** (the numeral zero, and not the letter O). Then you place the system entities around the perimeter of the page and use data flows to connect the entities to the central process. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

How do you know which entities and data flows to place in the context diagram? You begin by reviewing the system requirements to identify all external data sources and destinations. During that process, you identify the entities, the name and content of the data flows, and the direction of the data flows. If you do that carefully, and you did a good job of fact-finding in the previous stage, you should have no difficulty drawing the context diagram. Now review the following context diagram examples.

EXAMPLE: CONTEXT DIAGRAM FOR A GRADING SYSTEM

The context diagram for a grading system is shown in Figure 5-12 on the previous page. The GRADING SYSTEM process is at the center of the diagram. The three entities (STUDENT RECORDS SYSTEM, STUDENT, and INSTRUCTOR) are placed around the central process. Interaction among the central process and the entities involves six different data flows. The STUDENT RECORDS SYSTEM entity supplies data through the CLASS ROSTER data flow and receives data through the FINAL GRADE data flow. The STUDENT entity supplies data through the SUBMITTED WORK data flow and receives data through the GRADED WORK data flow. Finally, the INSTRUCTOR entity supplies data through the GRADING PARAMETERS data flow and receives data through the GRADE REPORT data flow.

EXAMPLE: CONTEXT DIAGRAM FOR AN ORDER SYSTEM

The context diagram for an order system is shown in Figure 5-13. Notice that the ORDER SYSTEM process is at the center of the diagram and five entities surround the process. Three of the entities, SALES REP, BANK, and ACCOUNTING, have single incoming data flows for COMMISSION, BANK DEPOSIT, and CASH RECEIPTS ENTRY, respectively. The WAREHOUSE entity has one incoming data flow – PICKING LIST – that is, a report that shows the items ordered and their quantity, location, and sequence to pick from the warehouse. The WAREHOUSE entity has one outgoing data flow, COMPLETED ORDER. Finally, the CUSTOMER entity has two outgoing data flows, ORDER and PAYMENT, and two incoming data flows, ORDER REJECT NOTICE and INVOICE

The context diagram for the order system appears more complex than the grading system because it has two more entities and three more data flows. What makes one system more complex than another is the number of components, the number of levels, and the degree of interaction among its processes, entities, data stores, and data flows.

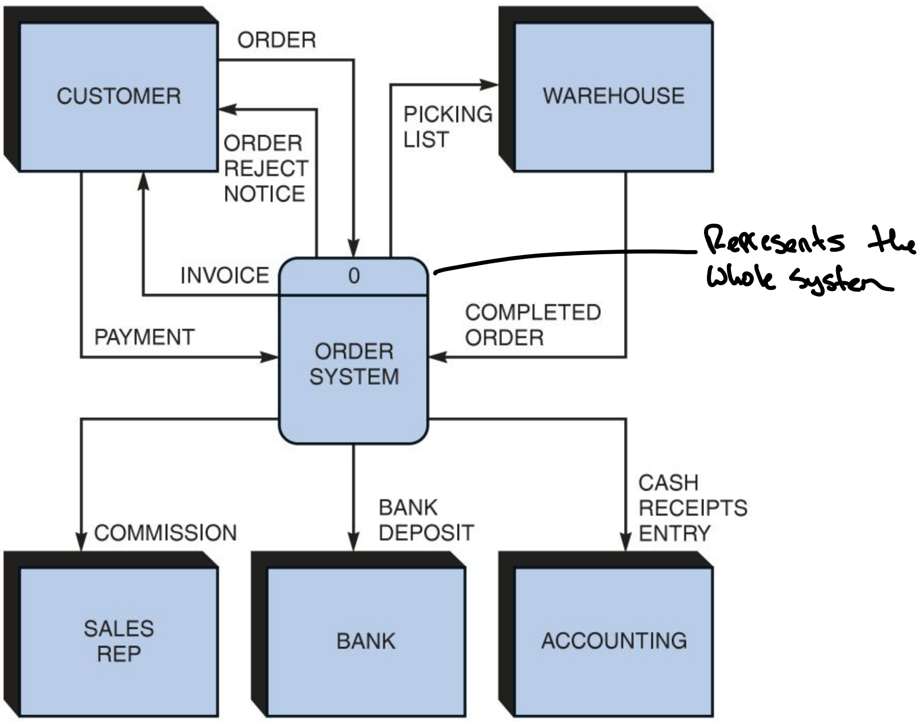


FIGURE 5-13 Context diagram DFD for an order system.

EXAMPLE: DIAGRAM 0 DFD FOR AN ORDER SYSTEM

Figure 5-16 on the next page shows the diagram 0 for an order system. Process 0 on the order system's context diagram is exploded to reveal three processes (FILL ORDER, CREATE INVOICE, and APPLY PAYMENT), one data store (ACCOUNTS RECEIVABLE), two additional data flows (INVOICE DETAIL and PAYMENT DETAIL), and one diverging data flow (INVOICE). The following walkthrough explains the DFD shown in Figure 5-16:

1. A CUSTOMER submits an ORDER. Depending on the processing logic, the FILL ORDER process either sends an ORDER REJECT NOTICE back to the customer or sends a PICKING LIST to the WAREHOUSE.
2. A COMPLETED ORDER from the WAREHOUSE is input to the CREATE INVOICE process, which outputs an INVOICE to both the CUSTOMER process and the ACCOUNTS RECEIVABLE data store.
3. A CUSTOMER makes a PAYMENT that is processed by APPLY PAYMENT. APPLY PAYMENT requires INVOICE DETAIL input from the ACCOUNTS RECEIVABLE data store along with the PAYMENT. APPLY PAYMENT also outputs PAYMENT DETAIL back to the ACCOUNTS RECEIVABLE data store and outputs COMMISSION to the SALES DEPT, BANK DEPOSIT to the BANK, and CASH RECEIPTS ENTRY to ACCOUNTING.

The walkthrough of diagram 0 illustrates the basic requirements of the order system. To learn more, you would examine the detailed description of each separate process.

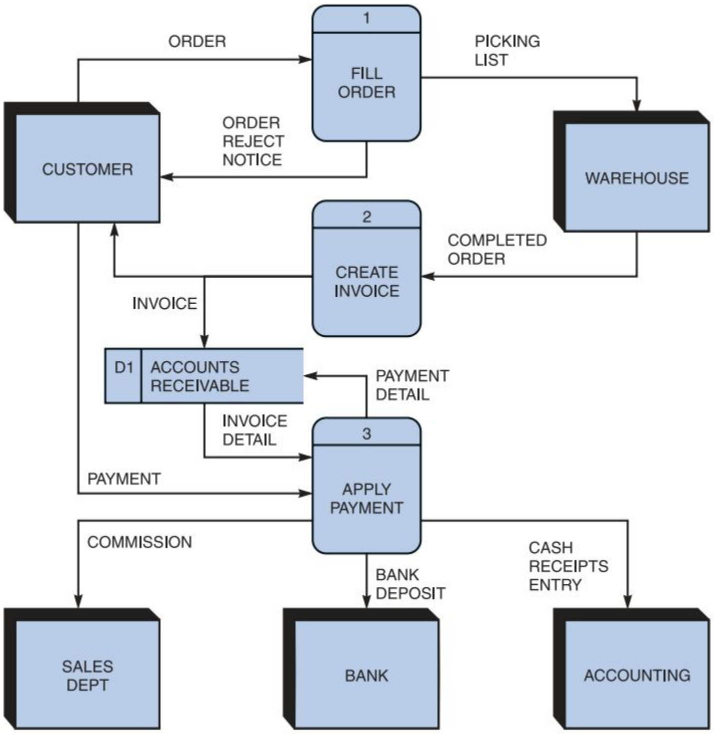


FIGURE 5-16 Diagram 0 DFD for the order system.

Lower-Level Diagrams

This set of lower-level DFDs is based on the order system. To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified. **Balancing** maintains consistency among a set of DFDs by ensuring that input and output data flows align properly. Leveling and balancing are described in more detail in the following sections.

LEVELING EXAMPLES

Leveling uses a series of increasingly detailed DFDs to describe an information system. For example, a system might consist of dozens, or even hundreds, of separate processes. Using leveling, an analyst starts with an overall view, which is a context diagram with a single process symbol. Next, the analyst creates diagram 0, which shows more detail. The analyst continues to create lower-level DFDs until all processes are identified as functional primitives, which represent single processing functions. More complex systems have more processes, and analysts must work through many levels to identify the functional primitives. Leveling also is called **exploding**, **partitioning**, or **decomposing.**

Figure 5-16 and Figure 5-17 provide an example of leveling. Figure 5-16 shows diagram 0 for an order system, with the FILL ORDER process labeled as process 1. Now consider Figure 5-17, which provides an exploded view of the FILL ORDER process. Notice that FILL ORDER (process 1) actually consists of three processes: VERIFY ORDER (process 1.1), PREPARE REJECT NOTICE (process 1.2), and ASSEMBLE ORDER (process 1.3)

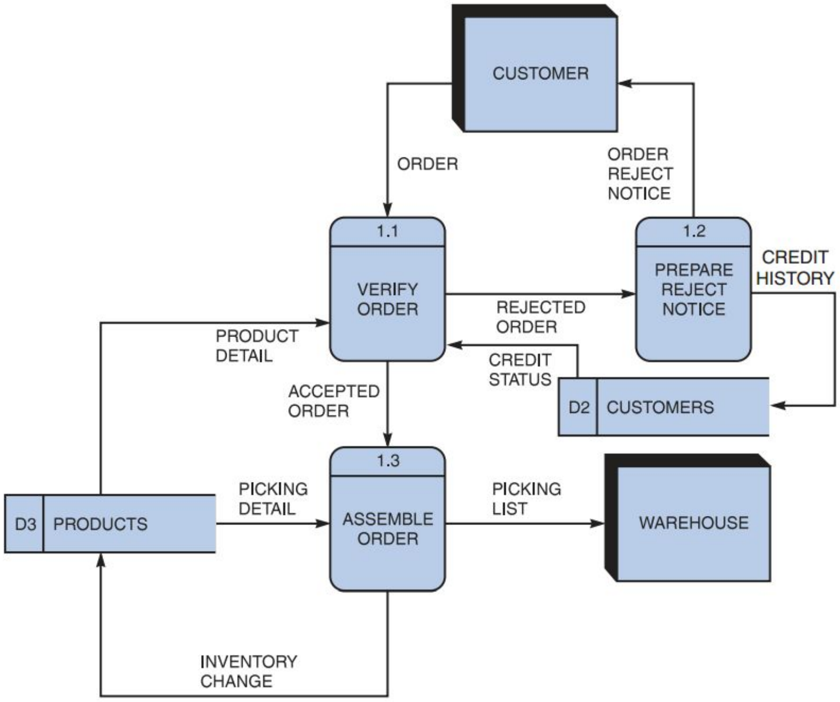
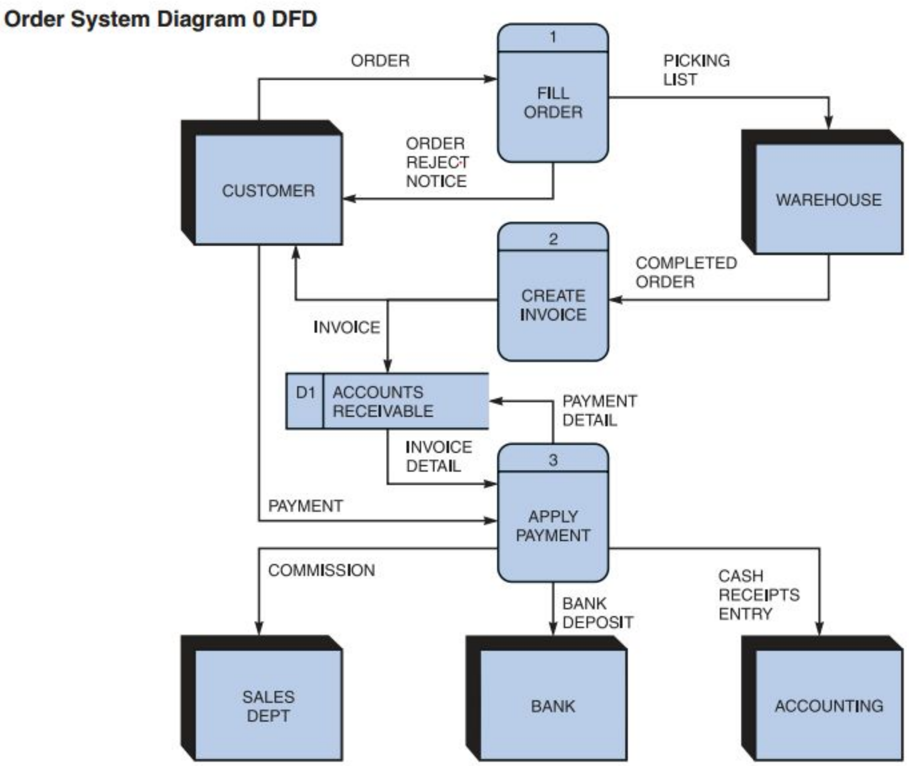


FIGURE 5-17 Diagram I DFD shows details of the FILL ORDER process in the order system.

As Figure 5-17 shows, all processes are numbered using a decimal notation consisting of the parent's reference number, a decimal point, and a sequence number within the new diagram. In Figure 5-17, the parent process of diagram 1 is process 1, so the processes in diagram 1 have reference numbers of 1.1, 1.2, and 1.3. If process 1.3, ASSEMBLE ORDER, is decomposed further, then it would appear in diagram 1.3 and the processes in diagram 1.3 would be numbered as 1.3.1, 1.3.2, 1.3.3, and so on. This numbering technique makes it easy to integrate and identify all DFDs.

When you compare Figures 5-16 and 5-17, you will notice that Figure 5-17 (the exploded FILL ORDER process) shows two data stores (CUSTOMERS and PRODUCTS) that do not appear on Figure 5-16, which is the parent DFD. Why not? The answer is based on a simple rule: When drawing DFDs, you show a data store only when two or more processes use that data store. The CUSTOMERS and PRODUCTS data stores were internal to the FILL ORDER process, so the analyst did not show them on diagram 0, which is the parent. When you explode the FILL ORDER process into diagram 1 DFD, however, you see that three processes (1.1, 1.2, and 1.3) interact with the two data stores, which now are shown.



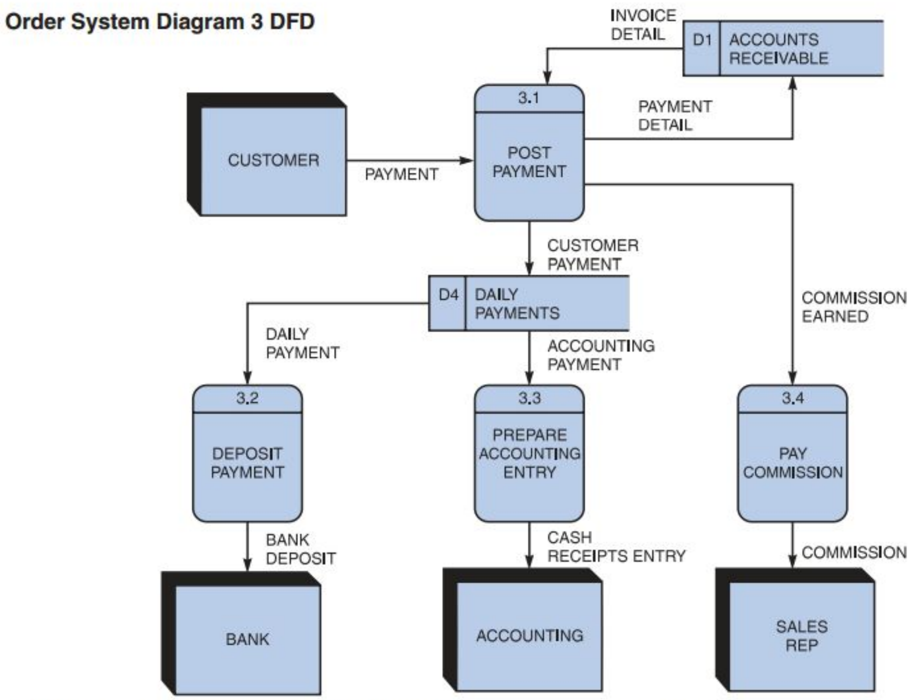
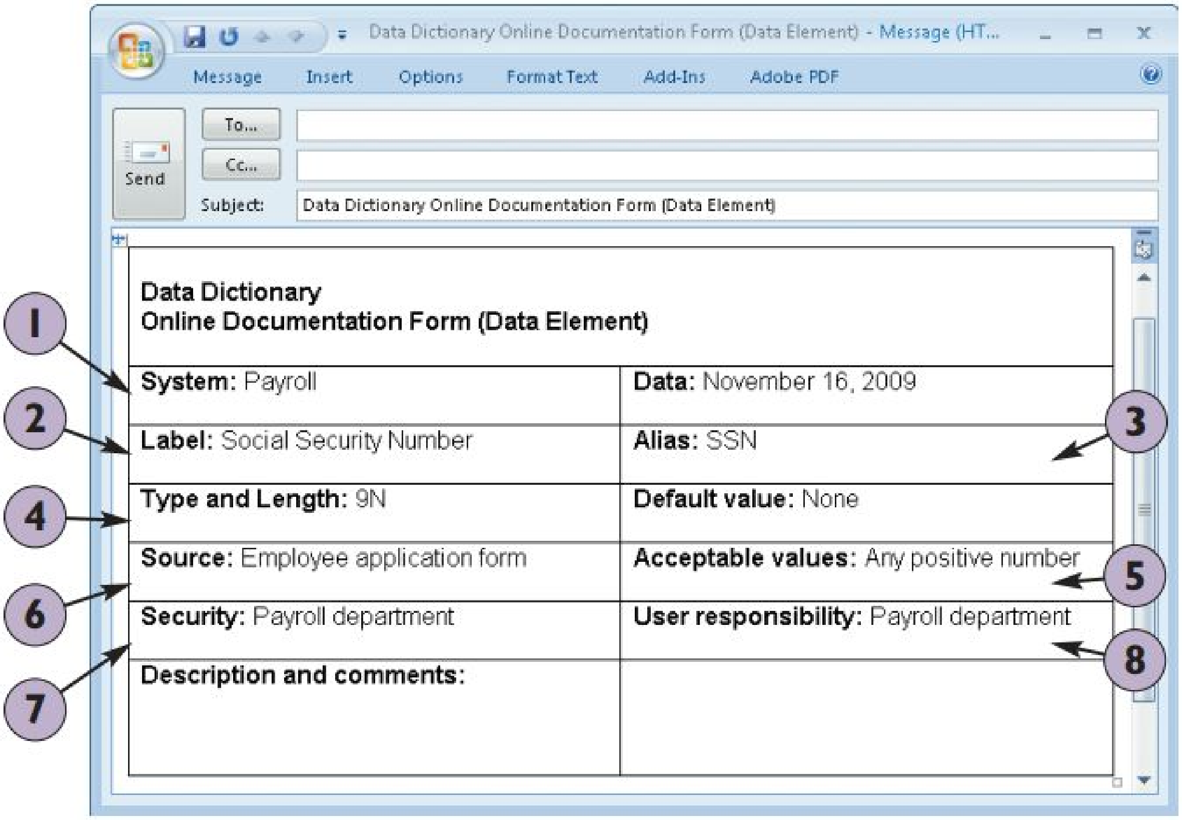


FIGURE 5-19 The order system diagram 0 is shown at the top of the figure, and exploded diagram 3 DFD (for the APPLY PAYMENT process) is shown at the bottom. The two DFDs are balanced, because the child diagram at the bottom has the same input and output flows as the parent process 3 shown at the top.

Data Element

1. Online or manual documentation entries often indicate which system is involved. This is not necessary with a CASE tool because all information is stored in one file that is named for the system.
2. The data element has a standard label that provides consistency throughout the data dictionary.
3. The data element can have an alternative name, or alias.
4. This entry indicates that the data element consists of nine numeric characters.
5. Depending on the data element, strict limits might be placed on acceptable values.
6. The data comes from the employee's job application.
7. This entry indicates that only the payroll department has authority to update or change this data.
8. This entry indicates the individual or department responsible for entering and changing data


Data Dictionary Online Documentation Form (Data Element) - Message (HT...
Message
Insert
Options
Format Text
Add-Ins
Adobe PDF
Send
To...
Cc...
1
2
3
4
5
6
7
8

Subject: Data Dictionary Online Documentation Form (Data Element)

Data Dictionary
Online Documentation Form (Data Element)

System: Payroll	**Data:** November 16, 2009
Label: Social Security Number	**Alias:** SSN
Type and Length: 9N	**Default value:** None
Source: Employee application form	**Acceptable values:** Any positive number
Security: Payroll department	**User responsibility:** Payroll department
Description and comments:	

Figure 5-24 shows a sample screen that illustrates how the SOCIAL SECURITY NUMBER data element might be recorded in the Visible Analyst data dictionary.

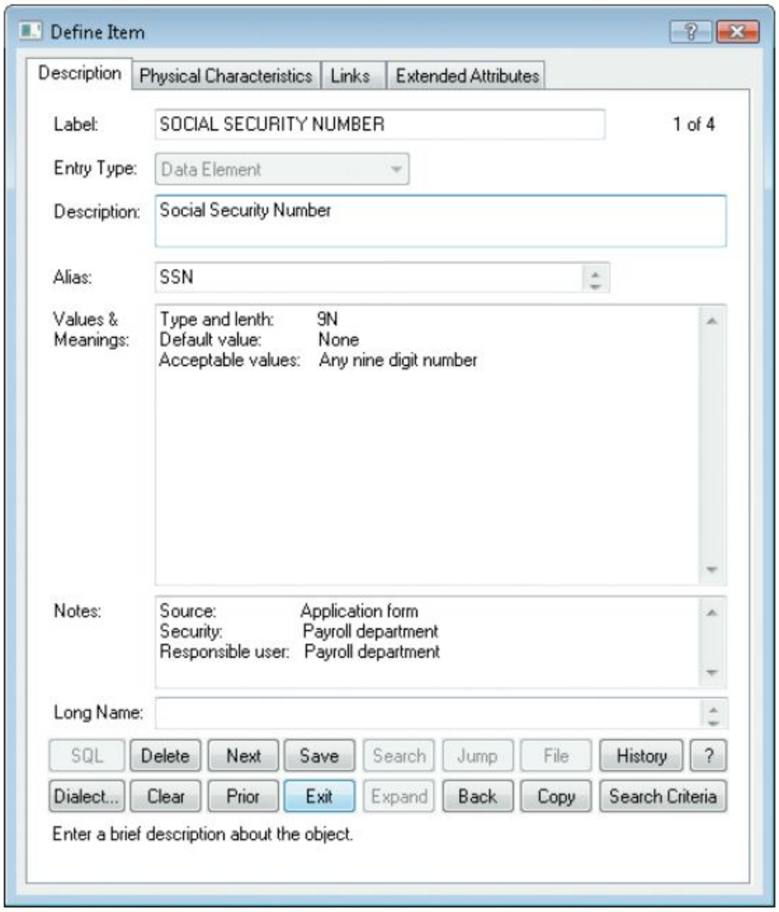
Regardless of the terminology or method, the following attributes usually are recorded and described in the data dictionary:

Data element name or label. The data element's standard name, which should be meaningful to users.

Alias. Any name(s) other than the standard data element name; this alternate name is called an alias. For example, if you have a data element named CURRENT BALANCE, various users might refer to it by alternate names such as OUTSTANDING BALANCE, CUSTOMER BALANCE, RECEIVABLE BALANCE, or AMOUNT OWED.

Type and length. Type refers to whether the data element contains numeric, alphabetic, or character values. Length is the maximum number of characters for an alphabetic or character data element or the maximum number of digits and number of decimal positions for a numeric data element. In addition to text and numeric data, sounds and images also can be stored in digital form. In some systems, these binary data objects are managed and processed just as traditional data elements are. For example, an employee record might include a digitized photo image of the person.

Default value. The value for the data element if a value otherwise is not entered for it. For example, all new customers might have a default value of $500 for the CREDIT LIMIT data element.



Label: SOCIAL SECURITY NUMBER 1 of 4

Entry Type: Data Element

Description: Social Security Number

Alias: SSN

Values & Meanings:
Type and lenth: 9N
Default value: None
Acceptable values: Any nine digit number

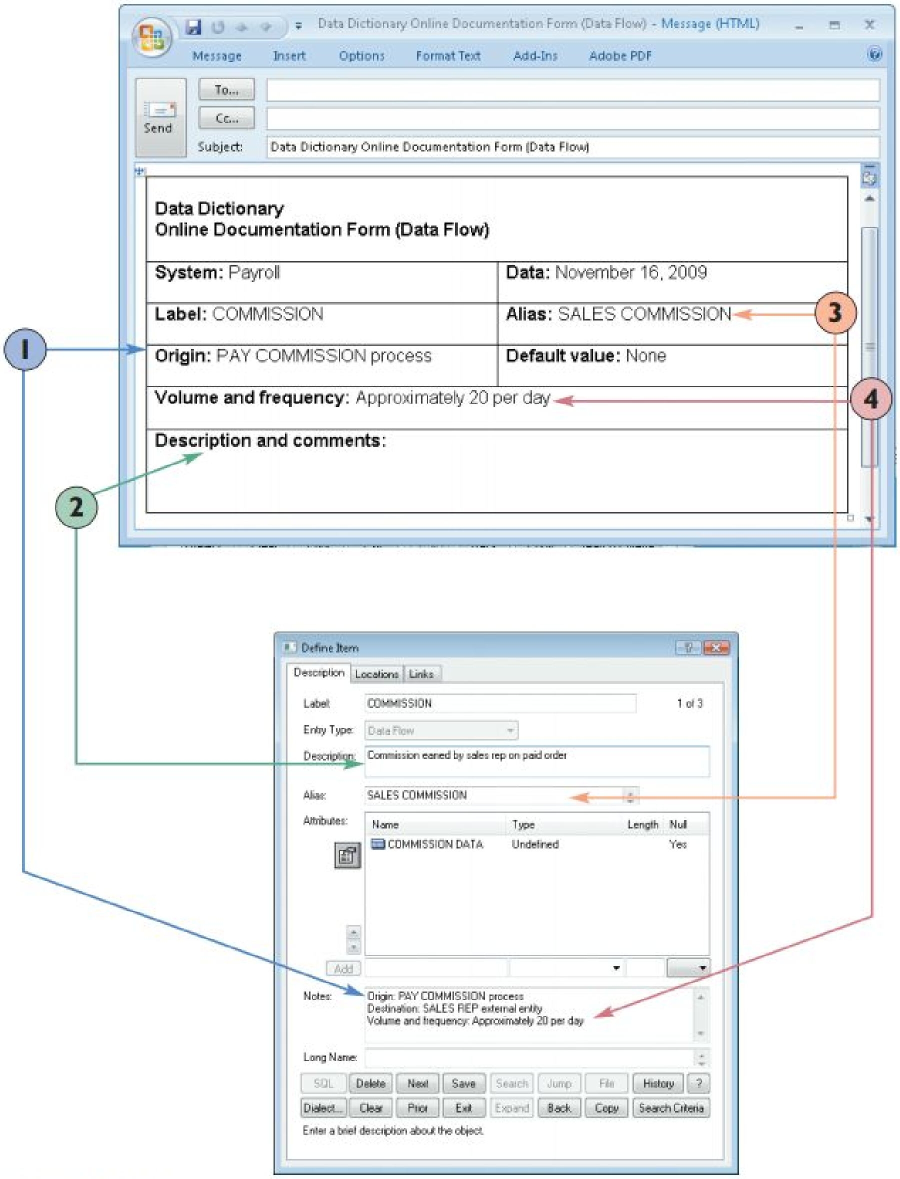
Notes:
Source: Application form
Security: Payroll department
Responsible user: Payroll department

Long Name:

Enter a brief description about the object.

FIGURE 5-24 A Visible Analyst screen describes the data element named SOCIAL SECURITY NUMBER. Notice that many of the items were entered from the online form shown in Figure 5-23 on page 217.

Data Flow



Subject: Data Dictionary Online Documentation Form (Data Flow)

Data Dictionary
Online Documentation Form (Data Flow)

System: Payroll	**Data:** November 16, 2009
Label: COMMISSION	**Alias:** SALES COMMISSION
Origin: PAY COMMISSION process	**Default value:** None
Volume and frequency: Approximately 20 per day	
Description and comments:	

Label: COMMISSION 1 of 3

Entry Type: Data Flow

Description: Commission eared by sales rep on paid order

Alias: SALES COMMISSION

Attributes:

Name	Type	Length	Null
COMMISSION DATA	Undefined		Yes

Notes: Origin: PAY COMMISSION process
Destination: SALES REP external entity
Volume and frequency: Approximately 20 per day

Long Name:

Enter a brief description about the object.

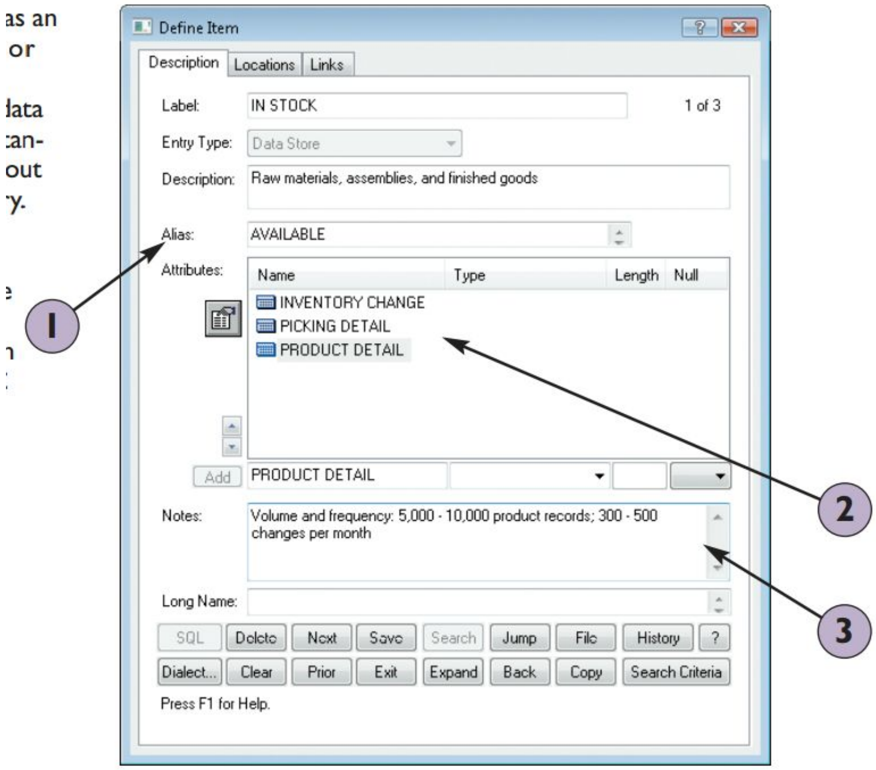
FIGURE 5-25 In the upper screen, an analyst has entered four items of information in an online documentation form. The lower screen shows the same four items entered into a Visible Analyst data dictionary form.

Data Store

Typical characteristics of a data store are as follows:

- **Data store name or label.** The data store name as it appears on the DFDs.
- **Description.** Describes the data store and its purpose.

- **Alternate name(s).** Aliases for the DFD data store name.
- **Attributes.** Standard DFD names that enter or leave the data store.
- **Volume and frequency.** Describes the estimated number of records in the data store and how frequently they are updated.



Label: IN STOCK 1 of 3

Entry Type: Data Store

Description: Raw materials, assemblies, and finished goods

Alias: AVAILABLE

Attributes:

Name	Type	Length	Null
INVENTORY CHANGE			
PICKING DETAIL			
PRODUCT DETAIL			

Add PRODUCT DETAIL

Notes: Volume and frequency: 5,000 - 10,000 product records; 300 - 500 changes per month

Long Name:

Press F1 for Help.

FIGURE 5-26 Visible Analyst screen that documents a data store named IN STOCK.

1. This data store has an alternative name, or alias.
2. For consistency, data flow names are standardized throughout the data dictionary.
3. It is important to document these estimates, because they will affect design decisions in subsequent SDLC phases.

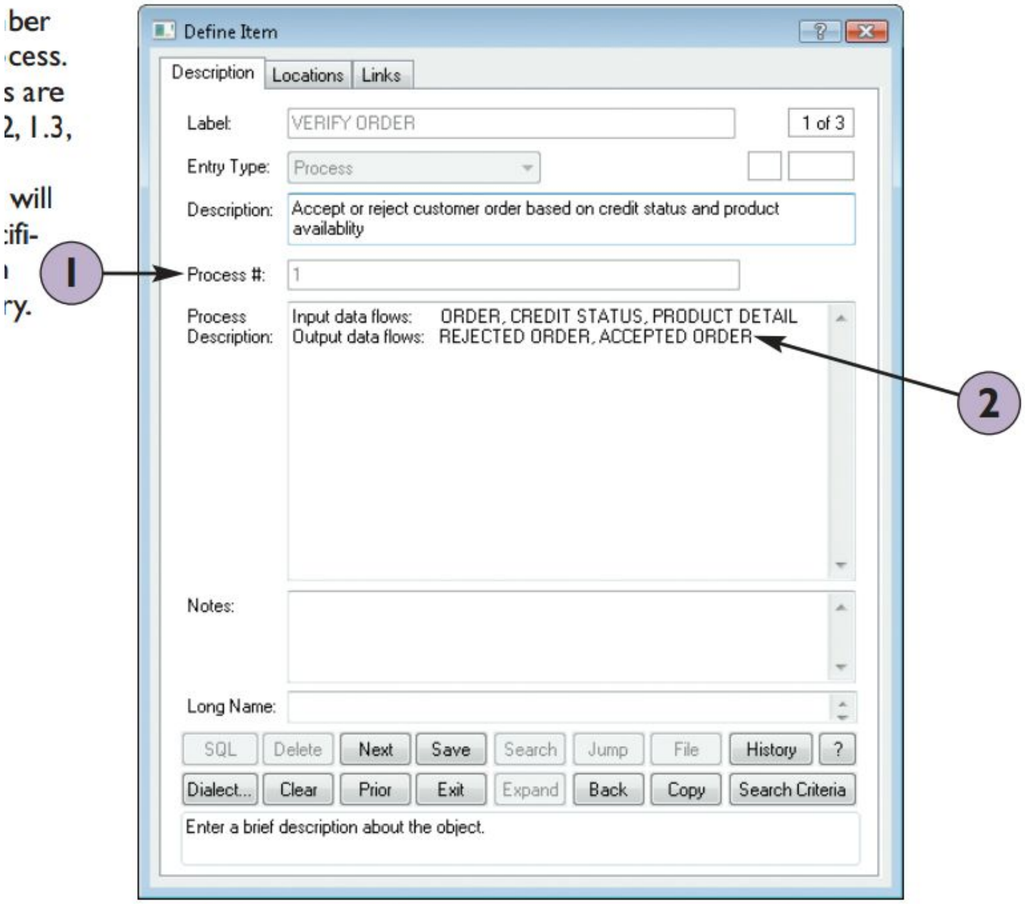
Process

You must document every process, as shown in Figure 5-27. Your documentation includes a description of the process's characteristics and, for functional primitives, a process description, which is a model that documents the processing steps and business logic.

The following are typical characteristics of a process:

- **Process name or label.** The process name as it appears on the DFDs.
- **Description.** A brief statement of the process's purpose.

- **Process number**. A reference number that identifies the process and indicates relationships among various levels in the system.
- **Process description**. This section includes the input and output data flows. For functional primitives, the process description also documents the processing steps and business logic. You will learn how to write process descriptions in the next section.



Label: VERIFY ORDER 1 of 3

Entry Type: Process

Description: Accept or reject customer order based on credit status and product availablity

Process #: 1

Process Description: Input data flows: ORDER, CREDIT STATUS, PRODUCT DETAIL
Output data flows: REJECTED ORDER, ACCEPTED ORDER

Notes:

Long Name:

Enter a brief description about the object.

FIGURE 5-27 Visible Analyst screen that describes a process named VERIFY ORDER.

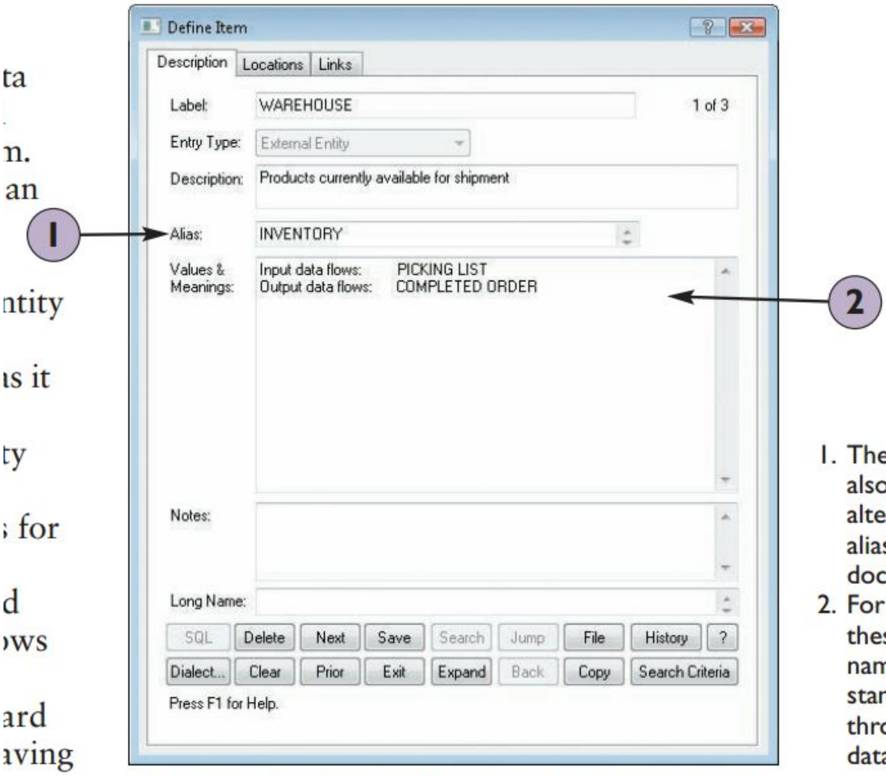
1. The process number identifies this process. Any subprocesses are numbered 1.1, 1.2, 1.3, and so on.
2. These data flows will be described specifically elsewhere in the data dictionary.

External Entity

Documenting the Entities

By documenting all entities, the data dictionary can describe all external entities that interact with the system. Figure 5-28 shows a definition for an external entity named WAREHOUSE. Typical characteristics of an entity include the following:

- **Entity name.** The entity name as it appears on the DFDs.
- **Description**. Describe the entity and its purpose.
- **Alternate name(s)**. Any aliases for the entity name.
- **Input data flows.** The standard DFD names for the input data flows to the entity.
- **Output data flows.** The standard DFD names for the data flows leaving the entity.



Label: WAREHOUSE 1 of 3

Entry Type: External Entity

Description: Products currently available for shipment

Alias: INVENTORY

Values & Meanings: Input data flows: PICKING LIST
Output data flows: COMPLETED ORDER

Notes:

Long Name:

Press F1 for Help.

FIGURE 5-28 Visible Analyst screen that documents an external entity named WAREHOUSE.

1. The external entity also can have an alternative name, or alias, if properly documented.
2. For consistency, these data flow names are standardized throughout the data dictionary.

Record (Data Structure)

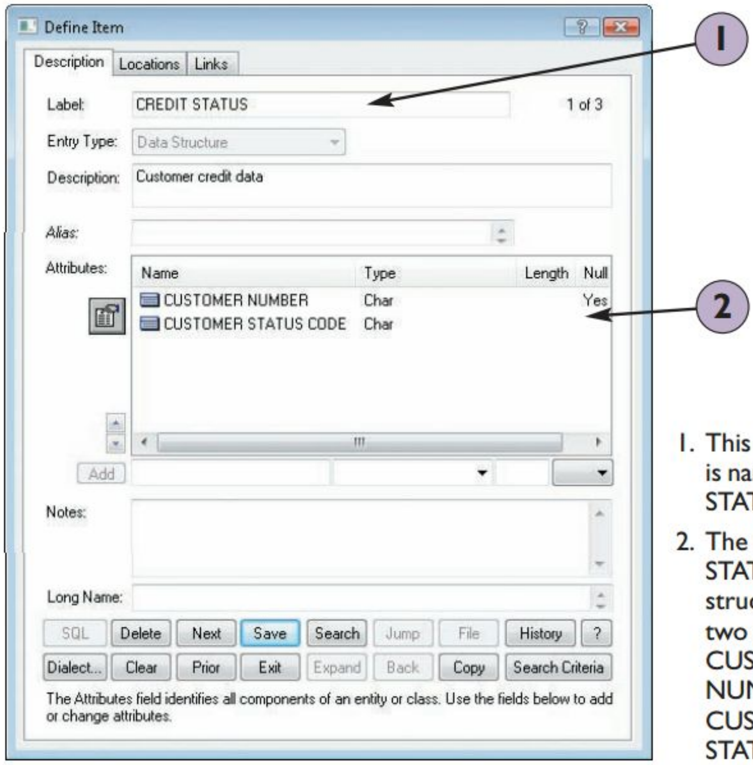
Documenting the Records

A record is a data structure that contains a set of related data elements that are stored and processed together. Data flows and data stores consist of records that you must document in the data dictionary. You define characteristics of each record, as shown in Figure 5-29.

Typical characteristics of a record include the following:

- **Record or data structure name.** The record name as it appears in the related data flow and data store entries in the data dictionary.

- **Definition or description.** A brief definition of the record.
- **Alternate name(s).** Any aliases for the record name.
- **Attributes.** A list of all the data elements included in the record. The data element names must match exactly what you entered in the data dictionary.



Label: CREDIT STATUS 1 of 3

Entry Type: Data Structure

Description: Customer credit data

Alias:

Attributes:

Name	Type	Length	Null
CUSTOMER NUMBER	Char		Yes
CUSTOMER STATUS CODE	Char		

Notes:

Long Name:

The Attributes field identifies all components of an entity or class. Use the fields below to add or change attributes.

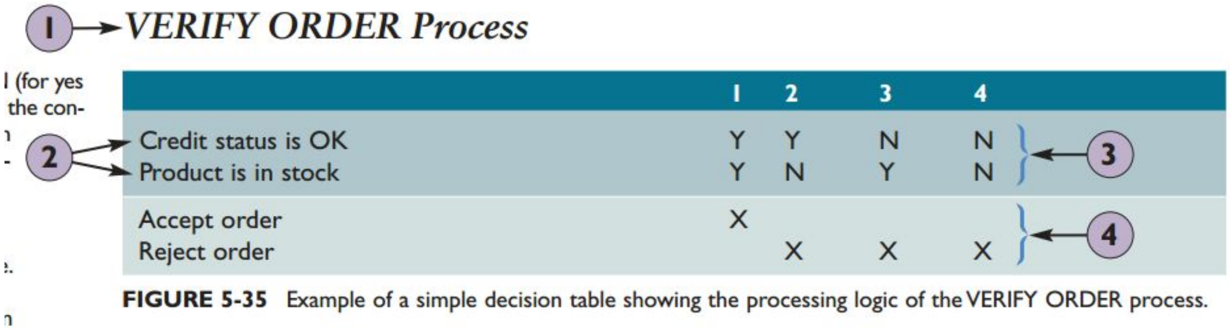
1. This data structure is named CREDIT STATUS.
2. The CREDIT STATUS data structure consists of two data elements: CUSTOMER NUMBER and CUSTOMER STATUS CODE.

FIGURE 5-29 Visible Analyst screen that documents a record, or data structure named CREDIT STATUS.

Decision Table

A **decision table** shows a logical structure, with all possible combinations of conditions and resulting actions. Analysts often use decision tables, in addition to structured English, to describe a logical process and ensure that they have not overlooked any logical possibility.

A simple example of a decision table based on the VERIFY ORDER process is shown in Figure 5-35. When documenting a process, it is important to consider every possible outcome to ensure that you have overlooked nothing. From the structured English description shown in Figure 5-33, we know that an accepted order requires that credit status is OK and the product is in stock. Otherwise, the order is rejected. The decision table shown in Figure 5-35 shows all the possibilities. To create the decision table, follow the steps indicated in the figure.


1
2
3
4
I (for yes
the con-

VERIFY ORDER Process

	1	2	3	4
Credit status is OK	Y	Y	N	N
Product is in stock	Y	N	Y	N
Accept order	X			
Reject order		X	X	X

FIGURE 5-35 Example of a simple decision table showing the processing logic of the VERIFY ORDER process.

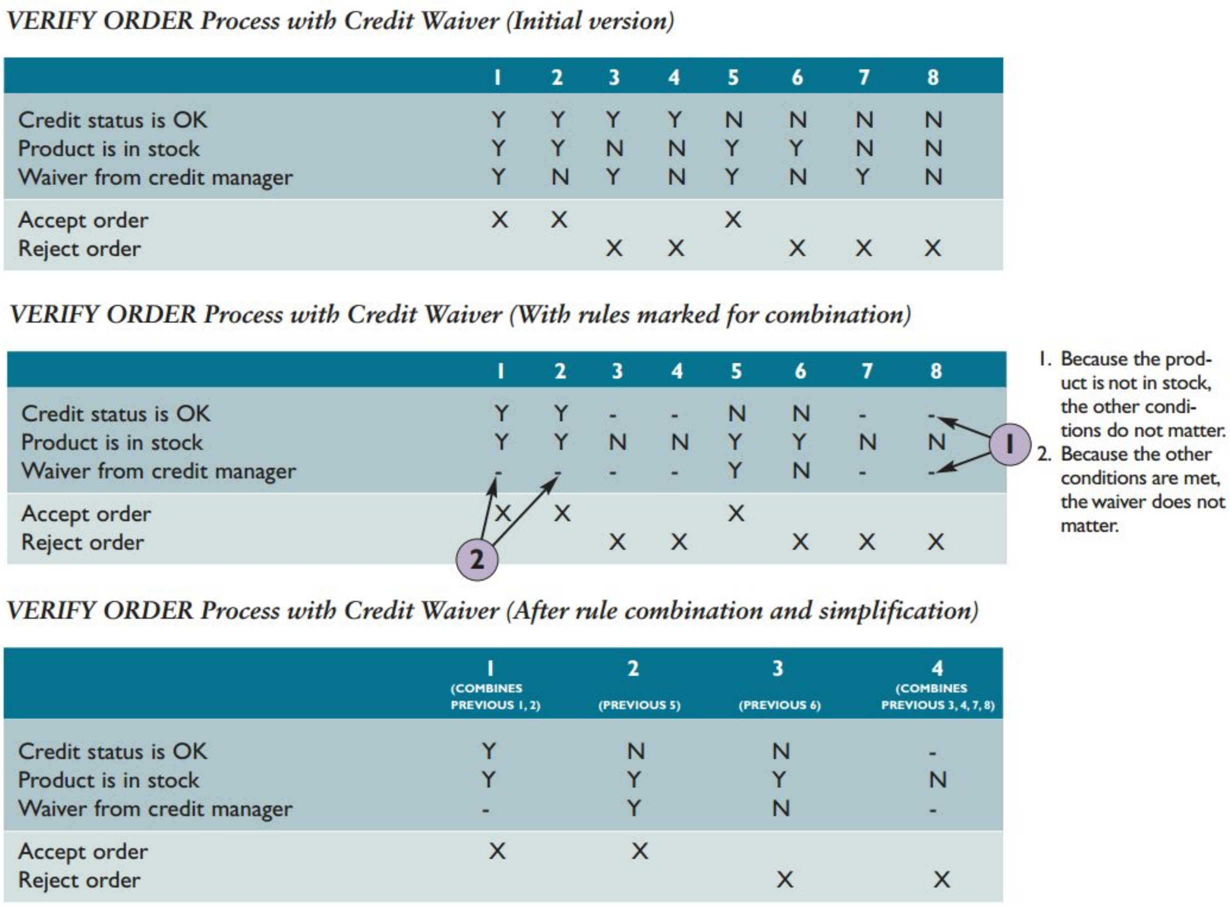
1. Place the name of the process in a heading at the top left.
2. Enter the conditions under the heading, with one condition per line, to represent the customer status and availability of products
3. Enter all potential combinations of Y/N (for yes and no) for the conditions. Each column represents a numbered possibility called a rule.
4. Place an X in the action entries area for each rule to indicate whether to accept or reject the order

The first table in Figure 5-36 shows eight rules. Because some rules are duplicates, however, the table can be simplified. To reduce the number of rules, you must look closely at each combination of conditions and actions. If you have rules with three conditions, only one or two of them may control the outcome, and the other conditions do not matter. You can indicate that with dashes (-) as shown in the second table in Figure 5-36. Then you can combine and renumber the rules, as shown in the final table.

In the example, rules 1 and 2 can be combined because credit status is OK, and the waiver is not needed. Rules 3, 4, 7, and 8 also can be combined because the product is not in stock, and credit status does not matter. The result is that instead of eight possibilities, only four logical rules are created that control the VERIFY ORDER process.

In addition to multiple conditions, decision tables can have more than two possible outcomes. An example is presented in the SALES PROMOTION POLICY decision table shown in Figure 5-37 on the next page. The decision table shown here is based on the sales promotion policy described in Figure 5-34. Here three conditions exist: Was the customer a preferred customer, did the customer order more than $1,000, and did the customer use our charge card? Based on these three conditions, four possible actions can occur, as shown in the table.

Decision tables often are the best way to describe a complex set of conditions. Many analysts use decision tables because they are easy to construct and understand, and programmers find it easy to work from a decision table when developing code.



VERIFY ORDER Process with Credit Waiver (Initial version)

	1	2	3	4	5	6	7	8
Credit status is OK	Y	Y	Y	Y	N	N	N	N
Product is in stock	Y	Y	N	N	Y	Y	N	N
Waiver from credit manager	Y	N	Y	N	Y	N	Y	N
Accept order	X	X			X			
Reject order			X	X		X	X	X

VERIFY ORDER Process with Credit Waiver (With rules marked for combination)

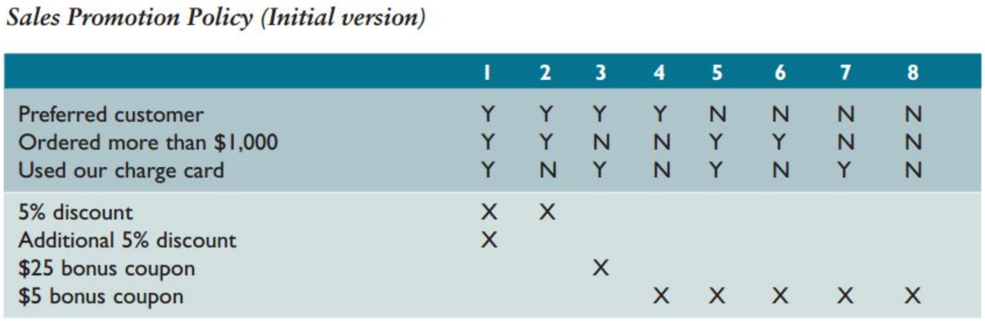
	1	2	3	4	5	6	7	8
Credit status is OK	Y	Y	-	-	N	N	-	-
Product is in stock	Y	Y	N	N	Y	Y	N	N
Waiver from credit manager	-	-	-	-	Y	N	-	-
Accept order	X	X			X			
Reject order			X	X		X	X	X

1. Because the product is not in stock, the other conditions do not matter.
2. Because the other conditions are met, the waiver does not matter.

VERIFY ORDER Process with Credit Waiver (After rule combination and simplification)

	1 (COMBINES PREVIOUS 1, 2)	2 (PREVIOUS 5)	3 (PREVIOUS 6)	4 (COMBINES PREVIOUS 3, 4, 7, 8)
Credit status is OK	Y	N	N	-
Product is in stock	Y	Y	Y	N
Waiver from credit manager	-	Y	N	-
Accept order	X	X		
Reject order			X	X

FIGURE 5-36 This example is more complex, because the credit manager can waive the credit status requirement in certain cases. To ensure that all possibilities are covered, notice that the first condition provides an equal number of Ys and Ns, the second condition alternates Y and N pairs, and the third condition alternates single Ys and Ns.



Sales Promotion Policy (Initial version)

	1	2	3	4	5	6	7	8
Preferred customer	Y	Y	Y	Y	N	N	N	N
Ordered more than $1,000	Y	Y	N	N	Y	Y	N	N
Used our charge card	Y	N	Y	N	Y	N	Y	N
5% discount	X	X						
Additional 5% discount	X							
$25 bonus coupon			X					
$5 bonus coupon				X	X	X	X	X

FIGURE 5-37 Sample decision table based on the sales promotion policy described in Figure 5-34 on page 224. This is the initial version of the table, before simplification.

Decision Trees

A **decision tree** is a graphical representation of the conditions, actions, and rules found in a decision table. Decision trees show the logic structure in a horizontal form that resembles a tree with the roots at the left and the branches to the right. Like flowcharts, decision trees are useful ways to present the system to management. Decision trees and decision tables provide the same results, but in different forms. In many situations, a graphic is the most effective means of communication, as shown in Figure 5-38.

Figure 5-39 shows the same SALES PROMOTION POLICY conditions and actions shown in Figure 5-37. A decision tree is read from left to right, with the conditions along the various branches and the actions at the far right. Because the example has two conditions with four resulting sets of actions, the example has four terminating branches at the right side of the tree.

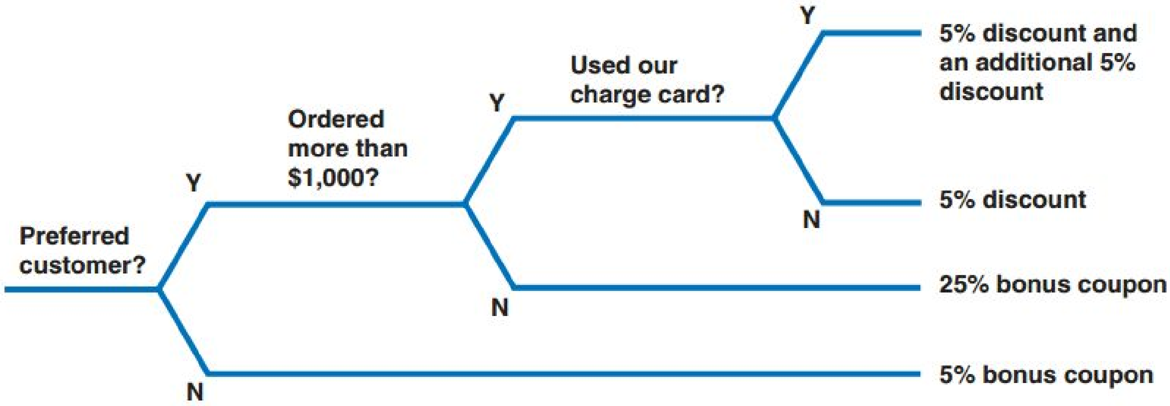


FIGURE 5-39 Sample decision tree. Like a decision table, a decision tree illustrates the action to be taken based on certain conditions, but presents it graphically. This decision tree is based on the sales promotion policy described in Figures 5-34 and 5-37 on pages 224 and 226.

Whether to use a decision table or a decision tree often is a matter of personal preference. A decision table might be a better way to handle complex combinations of conditions. On the other hand, a decision tree is an effective way to describe a relatively simple process.