CHAPTER 12

Using CASE Tools for Database Design

Abstract

This chapter looks at the role of CASE software in the design of database systems. The chapter covers the capabilities that can be expected in a CASE tool, as well as using the CASE tool to document a wide range of aspects of a database and/or system designs.

Keywords

CASE
CASE software
computer-aided software engineering
CASE for database design

A *CASE* (computer-aided software engineering) tool is a software package that provides support for the design and implementation of information systems. By integrating many of the techniques used to document a system design—including the data dictionary, data flows, and entity relationships—CASE software can increase the consistency and accuracy of a database design. They can also ease the task of creating the diagrams that accompany a system design.

There are many CASE tools on the market. The actual "look" of the diagrams is specific to each particular package. However, the examples presented in this chapter are typical of the capabilities of most CASE tools.

Note: The specific CASE software used in Chapters 13–15 is MacA&D by Excel Software (www.excelsoftware.com). (There's also a Windows version.) Other such packages that are well suited to database design include Visio (www.microsoft.com) and Visible Analyst (www.visible.com).

A word of warning is in order about CASE tools before we proceed any further. A CASE tool is exactly that—a tool. It can document a database design and it can provide invaluable help in maintaining the consistency of a design. Although some current CASE tools can verify the integrity of a data model, they cannot design the database for you. There is no software in the world that can examine a database environment and identify the entities, attributes, and relationships that should be represented in a database. The model created with CASE software is therefore only as good as the analysis of the database environment provided by the people using the tool.

CASE Capabilities

Most CASE tools organize the documents pertaining to a single system into a "project." As you can see in Figure 12.1, by default, a typical project supports the following types of documents:

- Data dictionary: In most CASE tools, the data dictionary forms the backbone of the project, providing a single repository for all processes, entities, attributes, and domains used anywhere throughout the project.
- Requirements: CASE tool requirements documents store the text descriptions of product specifications. They also make it possible to arrange requirements in a hierarchy, typically from general to specific.
- Data flow diagrams (DFD): As you read in Chapter 4, DFDs document the way in which data travel throughout an organization, indicating who handles the data. Although it isn't necessary to create a DFD, if your only goal with the project is to document a database design DFDs can often be useful in documenting the relationships between multiple organization units and the data they handle. Data flow diagrams can, for example, help you determine whether an organization needs a single database or a combination of databases.
- Structure charts: Structure charts are used to model the structure of application programs that will be developed using structured programming techniques. The charts show the relationship between program modules.
- Data models: Data models are the ER diagrams about which you have been reading. The ER diagram on which the examples in this chapter are based can be found in Figure 12.2.
- Screen prototypes: Drawings of sample screen layouts are typically most useful for documenting the user interface of application programs. However, they can also act as a crosscheck to ensure that a database design is complete by allowing you to verify that everything needed to generate the sample screen designs is present in the database.
- State models: State models, documented in state transition diagrams, indicate the ways in which data change as they move through the information system.
- Task diagrams: Task diagrams are used to help plan application programs in which multiple operations (tasks) occur at the same time. They are therefore not particularly relevant to the database design process.
- Class diagrams: Class diagrams are used when performing object-oriented, rather than structured, analysis and design. They can also be used to document an object-oriented database design.
- Object diagrams: Object diagrams are used during object-oriented analysis to indicate how objects communicate with one another by passing messages.

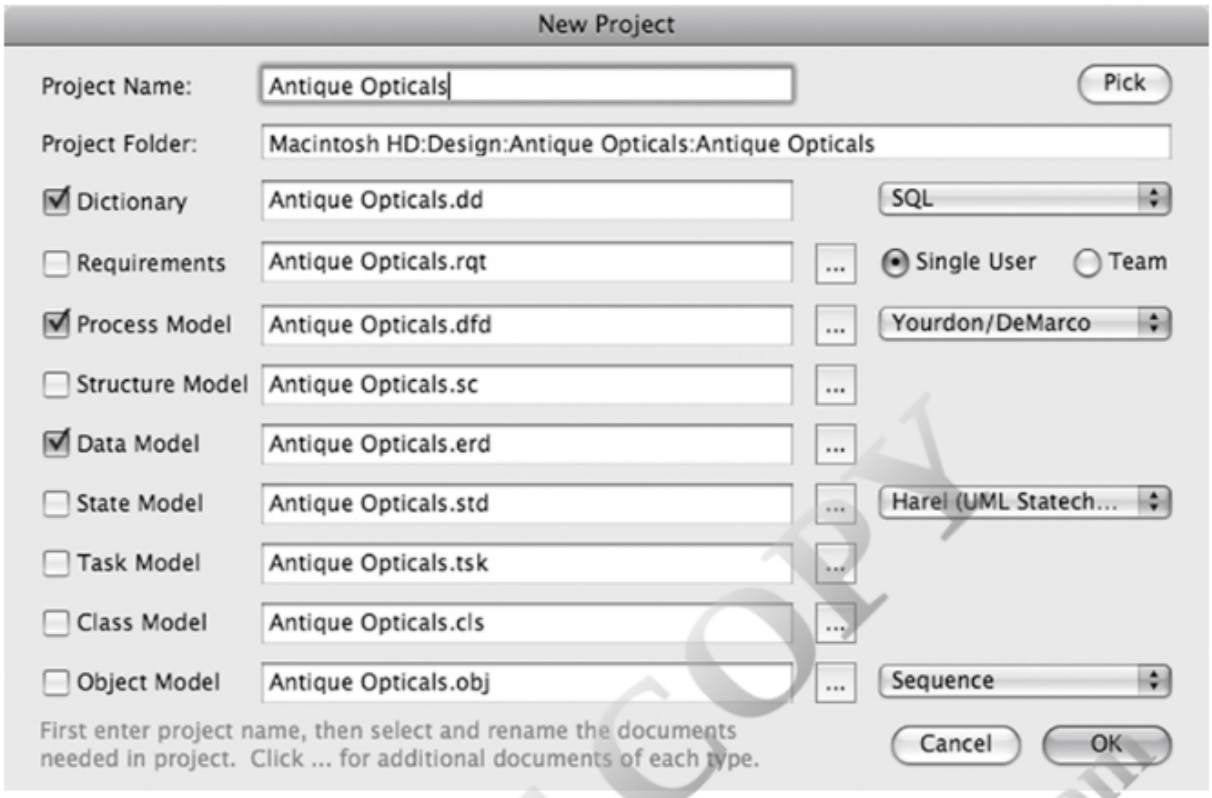
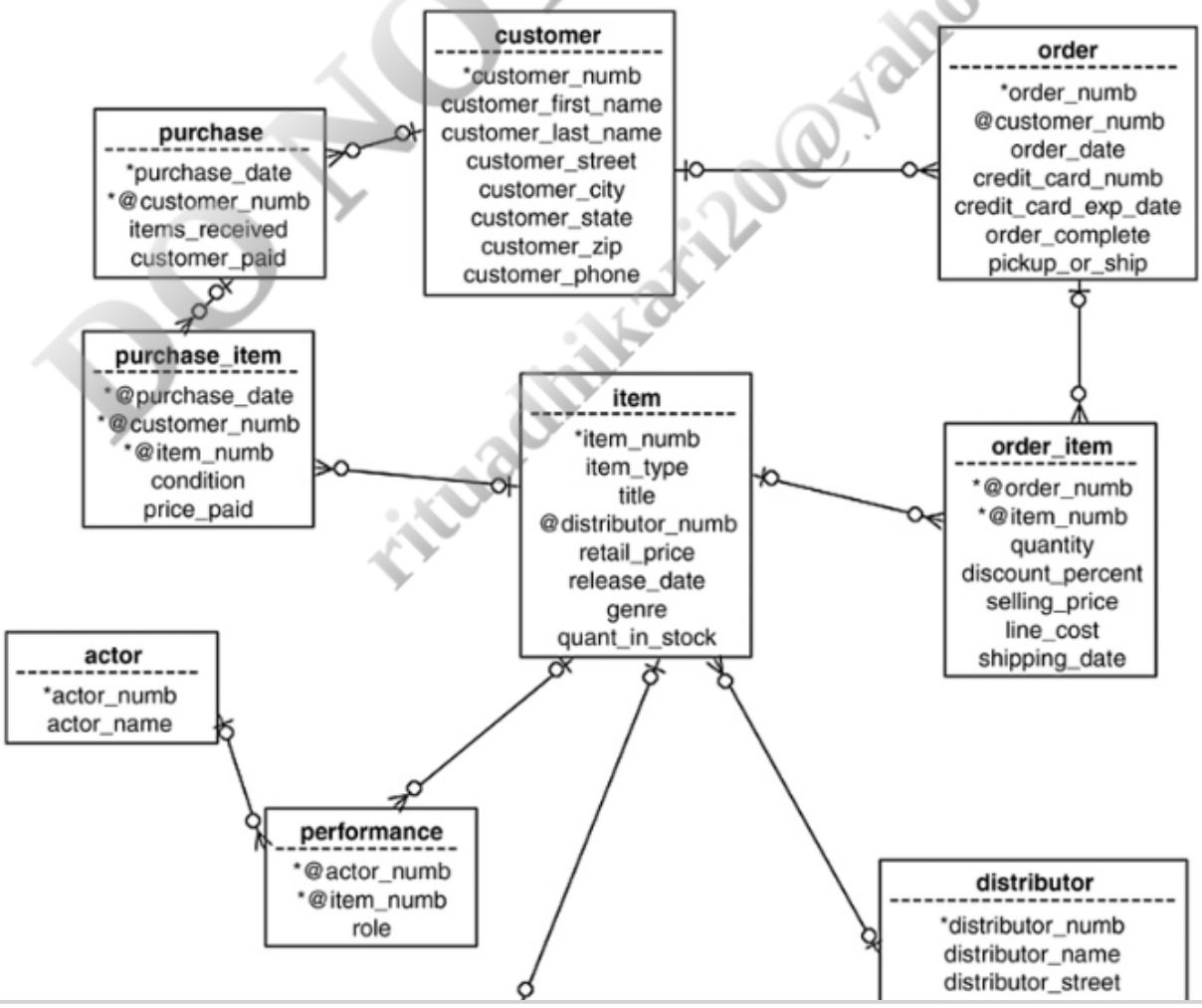


FIGURE 12.1 **CASE software project documents.**



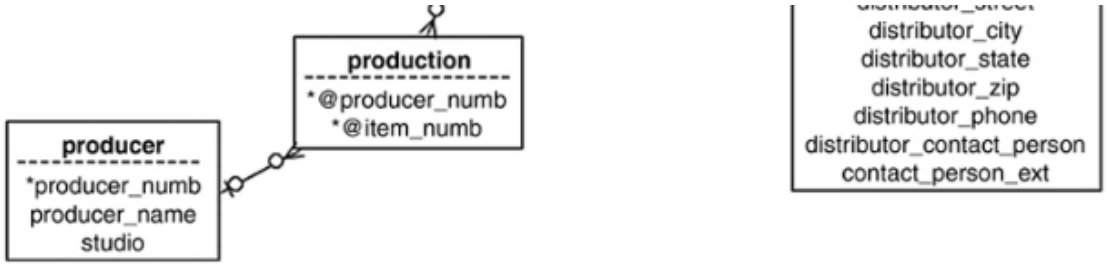


FIGURE 12.2 **ER diagram created with the sample CASE tool for *Antique Opticals*.**

Many of the diagrams and reports that a CASE tool can provide are designed to follow a single theoretical model. For example, the ER diagrams that you have seen earlier in this book might be based on the Chen model, or the Information Engineering model. Any given CASE tool will support some selection of diagramming models. You must therefore examine what a particular product supports before you purchase it to ensure that it provides exactly what you need.

ER Diagram Reports

In addition to providing tools for simplifying the creation of ER diagrams, many CASE tools can generate reports that document the contents of an ERD. For example, in Figure 12.3, you can see a portion of a report that provides a description of each entity and its attributes, including the attribute's data type. The "physical" line contains the name that the database element will have in SQL CREATE TABLE statements; it can be different than the element's data dictionary entry name.[1] For many designers, this type of report actually constitutes a paper-based data dictionary.

```
************************************************

Entity: actor

Language: SQL
Physical: actor##

Attributes:

...*actor_numb
Language: SQL
DataType: INTEGER
Physical: actor_numb##

...actor_name
Language: SQL
DataType: long_name
Physical: actor_name##


************************************************
```

FIGURE 12.3 **Part of an entity specification report.**

A CASE tool can also translate the relationships in an ER diagram into a report, such as that in Figure 12.4. The text in the report describes the *cardinality* of each relationship in the ERD (whether the relationship is one-to-one, one-to-many, or many-to-many) and can therefore be very useful for pinpointing errors that may have crept into the graphic version of the diagram.

```
actor is associated with zero or more instances of performance.
performance is associated with zero or one instance of actor.

customer is associated with zero or more instances of order.
```

order is associated with zero or one instance of customer.

customer is associated with zero or more instances of purchase.
purchase is associated with zero or one instance of customer.

distributor is associated with zero or more instances of item.
item is associated with zero or one instance of distributor.

item is associated with zero or more instances of order_item.
order_item is associated with zero or one instance of item.

item is associated with zero or more instances of performance.
performance is associated with zero or one instance of item.

item is associated with zero or more instances of production.
production is associated with zero or one instance of item.

item is associated with zero or more instances of purchase_item.
purchase_item is associated with zero or one instance of item.

order is associated with zero or more instances of order_item.
order_item is associated with zero or one instance of order.

producer is associated with zero or more instances of production.
production is associated with zero or one instance of producer.

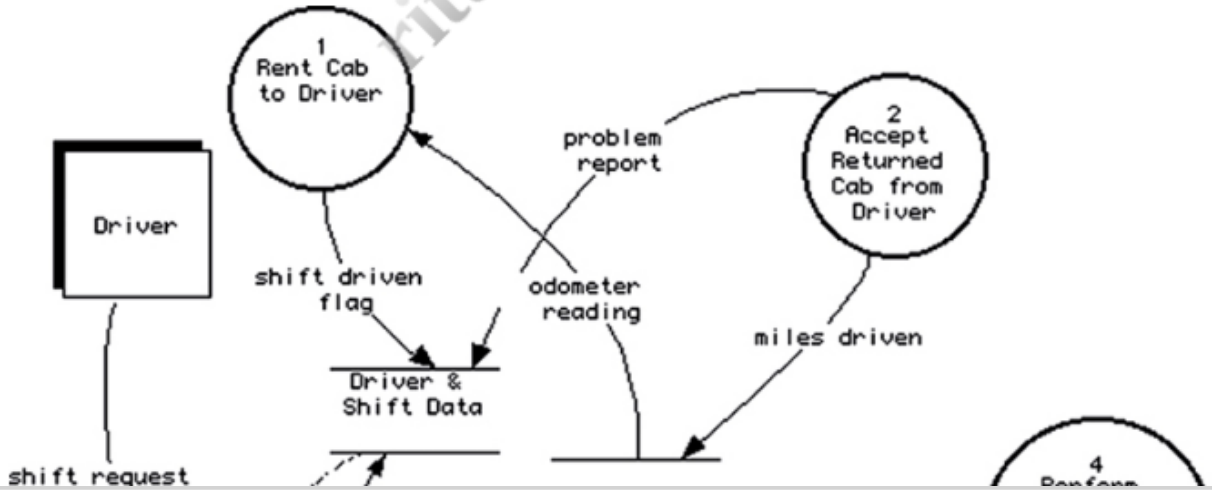
purchase is associated with zero or more instances of purchase_item.
purchase_item is associated with zero or one instance of purchase.

FIGURE 12.4 A relation specification report.

Data Flow Diagrams

There are two widely used styles for DFDs: Yourdon/DeMarco (which has been used throughout this book), and Gene & Sarson.

The Yourdon/DeMarco style, which you can see in Figure 12.5, uses circles for processes. (This particular example is for a small taxi company that rents its cabs to drivers.) Data stores are represented by parallel lines. Data flows are curved or straight lines, with labels that indicate the data that are moving along that pathway. External sources of data are represented by rectangles.



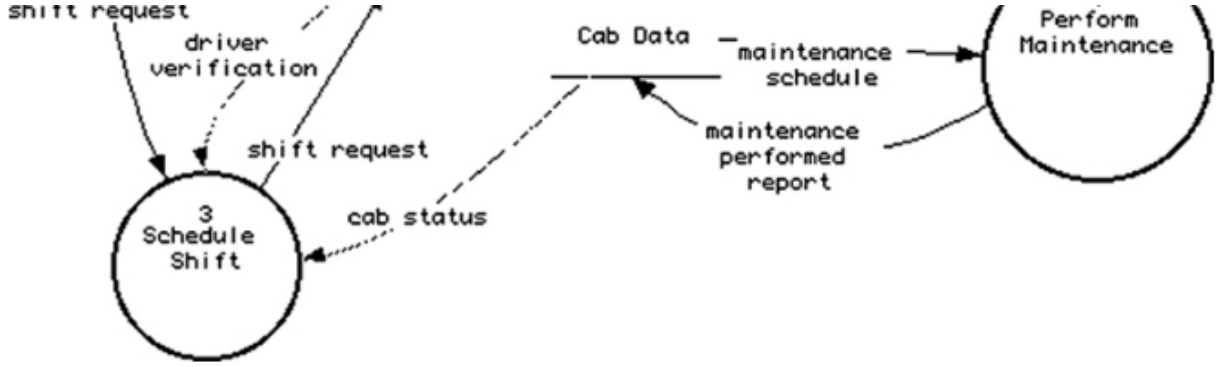


FIGURE 12.5 Yourdon/Marco style DFD.

In concept, the Gene & Sarson style is very similar: It varies primarily in style. As you can see in Figure 12.6, the processes are round-cornered rectangles as opposed to circles. Data stores are open-ended rectangles rather than parallel lines. External sources of data remain as rectangles and data flows use only straight lines. However, the concepts of numbering the processing and exploding each process with a child diagram that shows further detail is the same, regardless of which diagramming style you use.

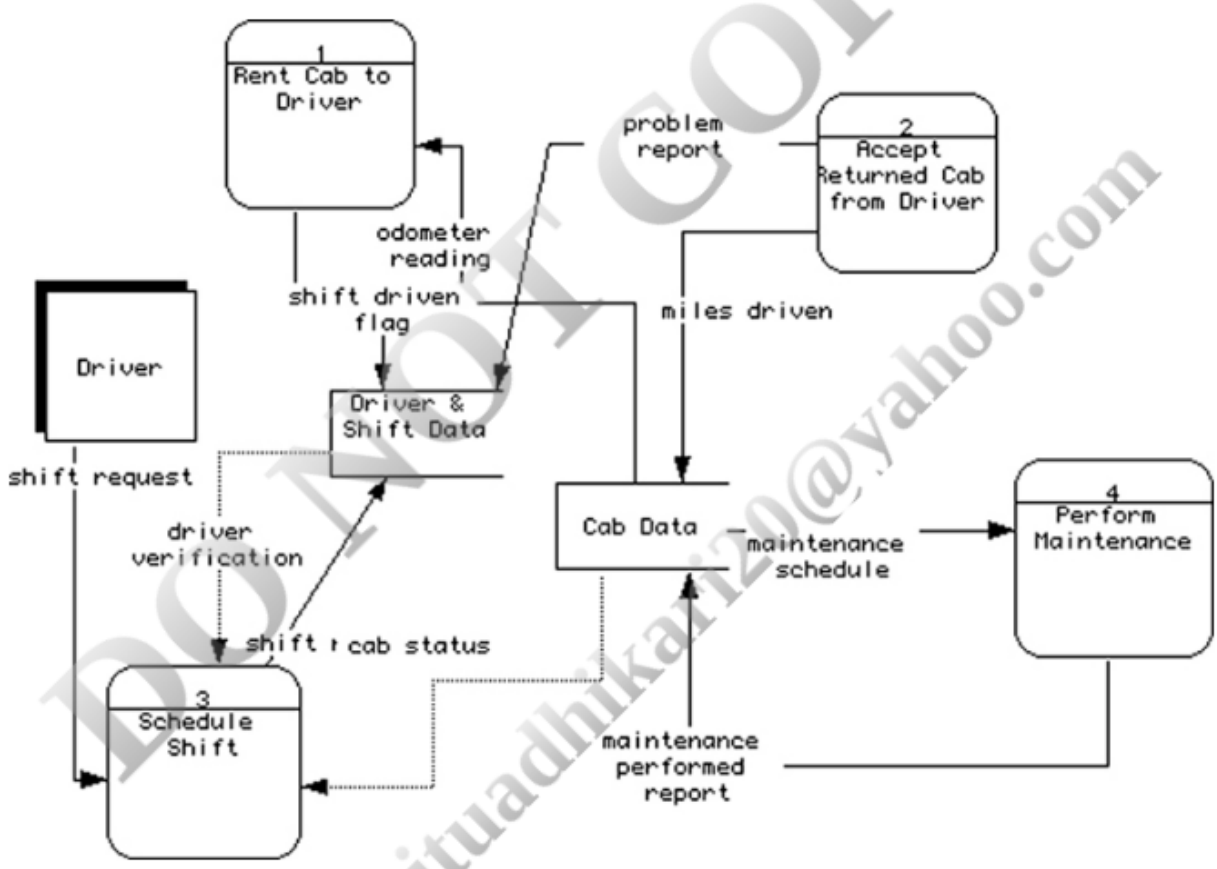


FIGURE 12.6 Gene & Sarson style DFD.

As mentioned earlier, DFDs are very useful in the database design process for helping a designer to determine whether an organization needs a single, integrated database or a collection of independent databases. For example, it is clear from the taxi company's DFDs that an integrated database is required. Of the four processes shown in the diagram, three use data from both the cab data store and the drive and shift data store. (Only the maintenance process uses just one data store.) You will see examples of using DFDs in this way in one of the case studies in the following three chapters.

The Data Dictionary

From a database designer's point of view, the ER diagram and its associated data dictionary are the two most important parts of CASE software. Since you were introduced to several types of ER diagrams in Chapter 4, we will not repeat them here, but instead focus on the interaction of the diagrams and the data dictionary.

A data dictionary provides a central repository for documenting entities, attributes, and domains. In addition, by linking entries in the ER diagram to the data dictionary you can provide enough information for the CASE tool to generate the SQL CREATE statements needed to define the structure of the database.

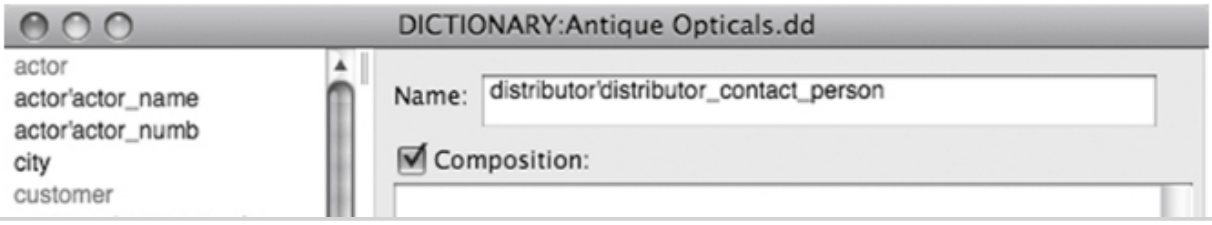
The layout of a data dictionary varies with the specific CASE tool, as does the way in which entries are configured. In the CASE tool used for

The layout of a data dictionary varies with the specific CASE tool, as does the way in which entries are configured. In the CASE tool used for examples in this chapter, entities are organized alphabetically, with the attributes following the entity name. Entity names are red; attributes are blue. (Of course, you can't see the colors in this black-and-white book, so you'll have to take my word for it.) Domain names appear alphabetically among the entities. Each relationship in the related ERD also has an entry. Because each item name begins with "Relation," all relationship entries sort together in the data dictionary.

When you select an entity name, the display shows the entity's name, composition (the attributes in the entity), definition (details needed to generate SQL, and so on), and type of database element (in the References section). Figure 12.7, for example, shows the information stored in the data dictionary for *Antique Opticals' customer* relation. All of the information about the entity (and all other entries, for that matter) is editable, but because the format is specific to the CASE tool, be careful when making changes unless you know exactly how entries should appear.

FIGURE 12.7 **Definition of an entity in a data dictionary window.**

Attribute entries (Figure 12.8) are similar to entity entries, but have no data in the composition section. Attribute definitions can include the attribute's data type, a default value, and any constraints that have been placed on that attribute. In most cases, these details are entered through a dialog box, relieving the designer of worrying about specific SQL syntax.



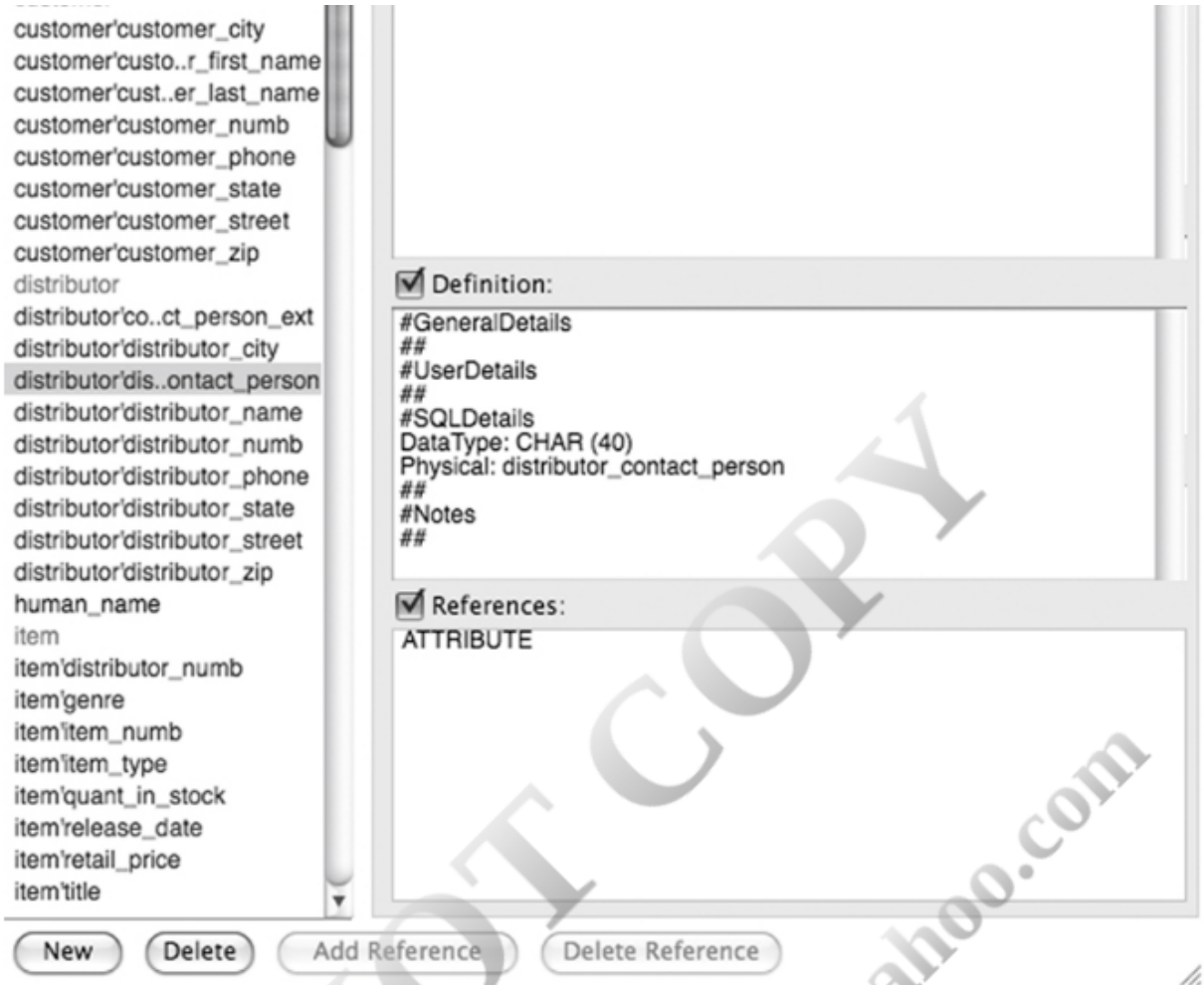
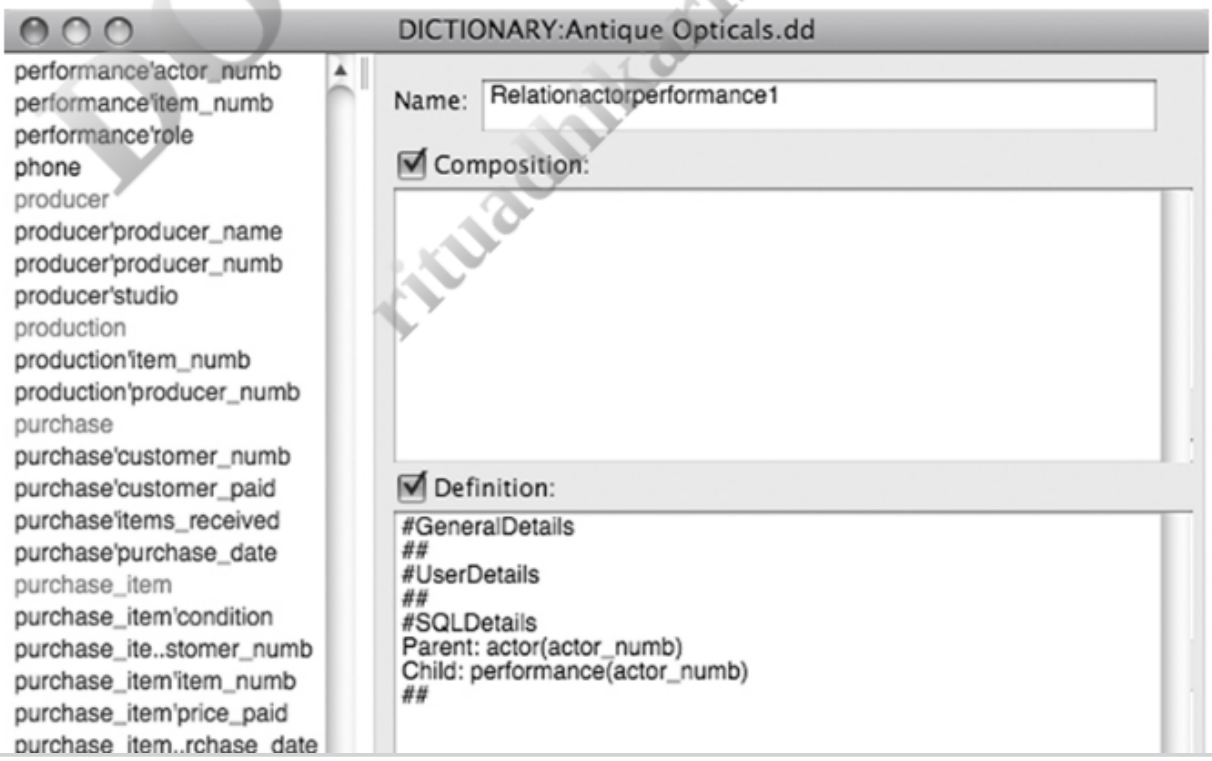


FIGURE 12.8 **Definition of an attribute in a data dictionary window.**

Relationships (Figure 12.9) are named by the CASE tool. Notice that the definition indicates which entities the relationship relates, as well as which is at the "many" end of the relationship (the child) and which is at the "one" end (the parent).



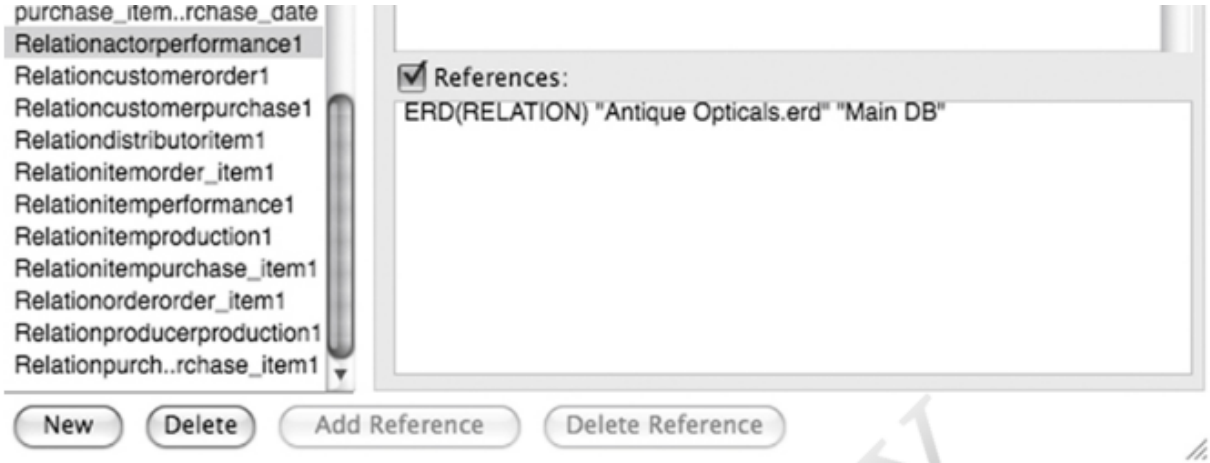


FIGURE 12.9 **Data dictionary entry for a relationship between two entities in an ERD.**

Many relational DBMSs now support the definition of custom domains. Such domains are stored in the data dictionary (Figure 12.10), along with their definitions. Once a domain has been created and is part of the data dictionary, it can be assigned to attributes. If a database administrator needs to change a domain, it can be changed once in the data dictionary and propagated automatically to all attributes entries that use it.

FIGURE 12.10 **Data dictionary entry for custom domain.**

foreign keys. This is yet another way in which the consistency of attribute definitions enforced by a CASE tool's data dictionary can support the database design process.

Note: Mac A&D is good enough at identifying foreign keys to pick up concatenated foreign keys.

Keep in mind that a CASE tool is not linked dynamically with a DBMS. Although data definitions in the data dictionary are linked to diagrams, changes made to the CASE tool's project will not affect the DBMS. It is up to the database administrator to make the actual changes to the database.

Code Generation

The end product of most database design efforts is a set of SQL CREATE TABLE commands. If you are using CASE software, and the software contains a complete data dictionary, then the software can generate the SQL for you. You will typically find that a given CASE tool can tailor the SQL syntax to a range of specific DBMSs. In most cases, the code will be saved in a text file, which you can then use as input to a DBMS.

Note: Most of today's CASE tools will also generate XML for you. XML provides a template for interpreting the contents of files containing data and therefore isparticularly useful when you need to transfer schemas and data between DBMSs with different SQL implementations, or between DBMSs that do not use SQL at all. XML has become so important for data exchange that it is covered in Chapter 26.

The effectiveness of the SQL that a CASE tool can produce, as you might expect, depends on the completeness of the data dictionary entries. To get truly usable SQL, the data dictionary must contain:

- Domains for every attribute.
- Primary key definitions (created as attributes, are added to entities in the ER diagram).
- Foreign key definitions (created as attributes, are added to entities in ER diagram or by the CASE tool after the ER diagram is complete).
- Any additional constraints that are to be placed on individual attributes or on the entity as a whole.

Sample Input and Output Designs

Sample input and output designs form part of the system documentation, especially in that they help document requirements. They can also support the database designer by providing a way to double-check that the database can provide all the data needed by application programs. Many CASE tools therefore provide a way to draw and label sample screen and report layouts.

Most of today's CASE tools allow multiple users to interact with the same project. This means that interface designers can work with the same data dictionary that the systems analysts and database designers are building, ensuring that all the necessary data elements have been handled.

For example, one of the most important things that the person scheduling cab reservations for the taxi company needs to know is which cabs are not reserved for a given date and shift. A sample screen, such as that in Figure 12.11, will do the trick.[2] The diagram shows what data the user needs to enter (the shift date and the shift name). It also shows the output (cab numbers). The names of the fields on the sample screen design can be linked to the data dictionary.

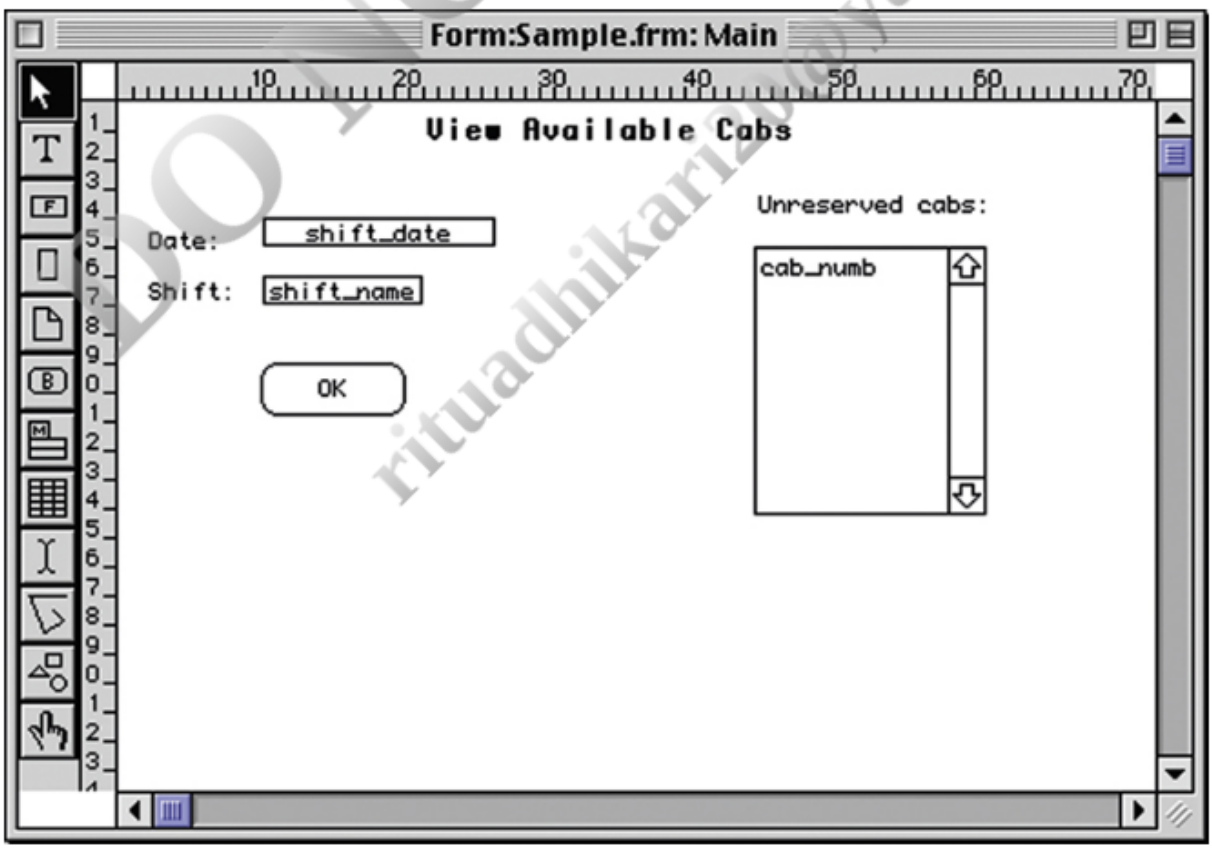


FIGURE 12.11 **Sample screen design.**

A CASE tool can be used to model an entire application program. The “browse” tool at the very bottom of the tool bar in Figure 12.11 switches into browse mode, in which buttons and menus become active. Users can make choices from pull-down menus that can be linked to other forms. Buttons can also trigger the opening of other forms. Users can click into data entry fields and tab between fields. Users can therefore not only see the layout and output screen and documents, but also navigate through an application.

The Drawing Environment

To this point, you’ve been reading about the way in which the functions provided by CASE software can support the database design effort. In this last section we will briefly examine the tools you can expect to find as part of CASE software, tools with which you can create the types of documents you need.

Because many of the documents you create with CASE software are diagrams, the working environment of a CASE tool includes a specialized drawing environment. For example, in Figure 12.12, you can see the drawing tools provided by the sample CASE tool for creating ER diagrams. (Keep in mind that each CASE tool will differ somewhat in the precise layout of its drawing tool bars, but the basic capabilities will be similar.)

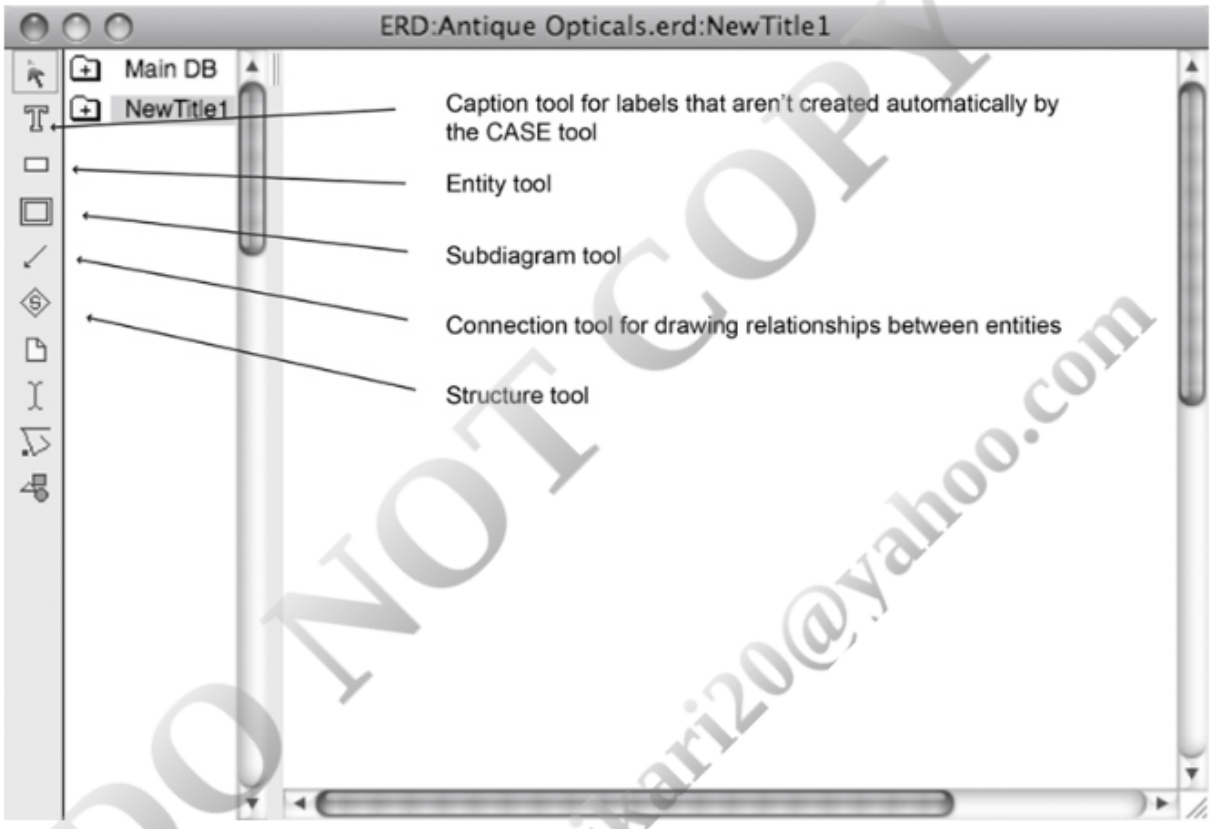


FIGURE 12.12 **Example CASE tool drawing environment for ER diagrams.**

The important thing to note is that the major shapes needed for the diagrams—for ER diagrams, typically just the entity and relationship line—are provided as individual tools. You therefore simply click the tool you want to use in the tool bar and draw the shape in the diagram, much like you would if you were working with a general-purpose object graphics program.

For Further Reading

To learn more about the Yourdon/DeMarco method of structure analysis using DFDs, see either of the following:

DeMarco T, Flauger PJ. *Structured analysis and system specification*. Prentice-Hall; 1985.
Yourdon E. *Modern structured analysis*. Prentice Hall PTR; 2000.

[1] The ## at the end of each Physical line is a terminator used by the CASE tool; it isn’t part of the name of the entity or attribute.

[2] In the interest of complete disclosure, you should know that when Mac A&D was ported from Mac OS 9 to Mac OS X, the screen and report design module wasn’t included. (It is now available as a stand-alone product.) Therefore, the sample screen designs that you will see in this chapter and in Chapter 13 are from an older version of the product.