### Figure 3.6 Typical three-dimensional display.

In contrast, the page dimension doesn’t correspond to anything that is actually on the screen. No matter how big the screen, all that is seen is an indicator saying which page is currently visible: shoes, socks, shirts, and so on. Still, it is easy to visualize the relationship between the data shown on the screen and the whole of the data set stored in the computer. All you have to do is imagine a three-dimensional data cube and a screen display showing one slice of that cube, as illustrated in Figure 3.7.

### Beyond Three Dimensions

Suppose that you are tracking different measures of different products by month for a chain of stores. You’ve now got a four-dimensional data set. What kind of visual metaphor should you use to form a picture of the whole? And how should the relevant information be organized on the screen? Trying to use a cube as the basis for a four- or higher-dimensional visualization can get very messy, very quickly. Figure 3.8 shows a picture of a tesseract, the technical name for a four-dimensional cube. Something seems wrong. It looks too complicated. The cube metaphor shown in Figure 3.3 seemed easy to understand. And adding the notion of stores to the data set in Figure 3.2 didn’t...
Figure 3.7 Slice of cube.

seem to add that much more additional complexity. So why, when we add just one more factor to the data set, does the complexity of its visualization skyrocket? This brings us to the second major point: the difference between logical and physical dimensions.

Figure 3.8 A view of a tesseract.
The cubes that most of us studied in a high school geometry class were implicitly physical because Euclidean or standard textbook geometry is based on the physical notions of length, width, and height. (Incidentally, the Greeks inherited their geometry from the Egyptians, who used it for surveying.) In textbook geometry, the $x$ axis is perpendicular to the $y$ axis is perpendicular to the $z$ axis. The three perpendicular angle-related axes translate perfectly into the physical dimensions of length, width, and height. Physical geometry is very practical, and it has served humankind well for over two millennia.

Now what does angle have to do with the relationship between variables, stores, and products? Does it make any sense to say that stores are perpendicular to products? Think about it for a minute. It makes no sense. What does make sense is to say that the display of stores on the computer screen is perpendicular to the display of products (but that is totally different). Figure 3.9 is a simple sketch of an event that could have generated the data shown in Figure 3.7. It is a picture of a retailer selling a product to a customer in exchange for money and buying supplies from a supplier. It certainly doesn't look like a cube. Figure 3.10 displays a series of these pictures: one for each month and their correspondence with a cube arrangement of data. Yet there is something about the relationship between the cube and the event pictured by the cube that makes the cube an intuitive representation of the event. We need to explore this further.

Every point in the cube represents a particular measurement taken from the event. For example, one point in the cube might represent the fact that $1,000$ worth of shoes were sold in February. Another point might represent the fact that $500$ worth of pants were bought in March. Notice that each fact in the cube is identified by one value from each dimension.

Going back to the event image in Figure 3.9, every dimension of the event is a coexistent factor. For each sales transaction, one can always identify a product sold, a dollar amount, and a time. (Where there is smoke there is fire.) Furthermore, each of the coexistent factors is independent of all the others. In other words, any product could be sold at any time. And at any

Figure 3.9 A retailer-centric view of the world.
time, any product could be sold. (Actually, the condition of independence does not hold for the variables, but it is safe to ignore that now. For a full treatment of this see Chapter 15.)

The cube works so well as an intuitive representation of the event because all of the dimensions coexist for every point in the cube and they are all independent of one another. Figure 3.11 shows how any point \((x_n,y_n,z_n)\) in three-dimensional space is identified by its \(x\) value, its \(y\) value, and its \(z\) value (or its \textit{product}, \textit{time}, and \textit{variable} values). From any point in the cube, one can move in any value in any dimension independent of any other change in any other dimension. This is shown in Figure 3.12. The problem with the cube is that, physically speaking, there are only three independent dimensions. So the cube breaks down as a metaphor for visualizing more than three dimensions.

Even though there is nothing wrong with using an angle-based cube for representing up to three dimensions of an event, the angle-based definition of a dimension is not necessary for a useful representation of the event. A useful representation requires coexistent and independent dimensions regardless of how that coexistence and independence are defined. These two properties are logical properties, not physical properties. Any metaphor that provides a consistent definition of independent and coexistent dimensions will work.
Multidimensional Domain Structures

Let's introduce a new metaphor for representing events that is not based on angle-defined dimensions and that is capable of representing any number of event dimensions. If you want to give it a name you may call it a multidimensional domain structure (MDS). The new metaphor is shown in Figure 3.13 along the path between data-generating events and data cubes shown in Figure 3.10. Each dimension is represented by a vertical line segment. Every...
CHAPTER 3

**Data generating event**

**MDS**

**Datacube**

- Time
- Product
- Variable

**Figure 3.13** Multidimensional domain structures are a way of representing events.

A member within a dimension is represented by a unit interval within the segment. As we are starting with a three-dimensional example, there are three line segments: one for time, one for products, and one for variables. Any union of one interval from each of the three line segments is connected to an element in the event and in the cube. For example, in Figure 3.13, the MDS highlights March shoe sales as does the cube. In the same way that one can move independently in each cube dimension, one can move independently in each MDS dimension, as shown in Figure 3.14.

In Figure 3.14, where there are 12 time periods, 10 products, and 5 variables, there are 12*10*5 = 600 hypercube intersections or potential data points. In this sense, an MDS is more descriptive than a physical cube. However, an MDS doesn’t show actual data points, just possible combinations of dimension members. So here it is less descriptive than a cube that can at least allude to (though it cannot actually show) all data points. Then again, the purpose of a visual metaphor is to give a useful picture of the whole structure of a model. It is the job of the display to show the data.

**Adding a Fourth Dimension**

Using an MDS, it is easy to add a fourth dimension to the model. Remember when we tried to add a store dimension to the cube? That’s when things started to break down. But not with an MDS. It’s a cinch. Just add a fourth line segment called stores, as shown in Figure 3.15. The MDS is not a pictorial representation of the data-generating event, but then neither was the cube. The MDS shows the number of data points extracted from the event and their logical organization. It shows all the dimensions one can browse in and how far one can go in any dimension. It shows more structural informa-
Figure 3.14 You can move independently in each MDS dimension.

Figure 3.15 The fourth dimension may be represented by a fourth line segment.
Figure 3.16 A two-dimensional arrangement.

Representing Hypercubes on a Computer Screen

We've figured out why the physical cube metaphor breaks down, and we've introduced a logical visualization principle for representing the structure of N-dimensional data sets. We still need to see them on the computer screen. This brings us to our final hurdle: mapping multiple logical dimensions onto a single physical (screen) dimension. Look again at the image of a three-dimensional grid-style interface, as shown in Figure 3.6. How are we going to represent four or more logical dimensions given our three display dimensions of row, column, and page? The answer is we combine multiple logical dimensions within the same display dimension. Let's examine this more closely.

Look at Figure 3.16. It shows a two-dimensional arrangement of products by variables with the name of each intersection explicitly written. Notice how each point or intersection in the two-dimensional grid is formed from one member of each dimension. And notice how each member of each dimension combines with each member of the other dimension. In this simple example there are two products and three variables for a total of six product variable combinations.

Mapping the two dimensions into one dimension means creating a one-dimensional version of the two-dimensionally arranged intersections. Although any one-dimensional arrangement will work, the typical method is to nest one dimension within the other. For example, Figure 3.17 shows how

Figure 3.17 Nesting variables within products.
Figure 3.18 Nesting products within variables.

to create a one-dimensional list of variables nested within products from the two-dimensional grid originally shown in Figure 3.16. Notice how the list scrolls through every member of the variables dimension for each product as it is scrolling through products. You can think of it as a generic code loop:

For variables = 1 to N

    For products = 1 to N
    End products

End variables

This is what it means for variables to be nested within products. In contrast, Figure 3.18 shows how to create a one-dimensional list of products nested within variables from the same two-dimensional grid. Notice how in both figures, the number of elements is the same for the one- as for the two-dimensional arrangement. No data is lost by combining dimensions. (Actually, some information is lost as you will see later.) Any number of dimensions can be combined.

**The Effects of Combining Dimensions**

Two main things change as a result of combining dimensions: axis lengths and neighborhoods. One main thing does not change: truth criteria.

The first thing that changes is the shape of the viewable data. As you can see from either Figure 3.17 or Figure 3.18, the length of the one-dimensional list is equal to the product of the lengths of each of the two dimensions in the two-dimensional arrangement. When the lengths are as small as they are in these examples, the impact is negligible, but when the axis lengths are in the hundreds or thousands it can make a real difference. For example, browsing through a two-
dimensional grid of 100 by 100, which may fit all at once on your screen, is considerably easier than browsing through a list of 10,000 rows, which most certainly will require substantial scrolling. This doesn’t mean you shouldn’t combine dimensions (as you will learn in Chapter 10, doing so is frequently desirable), but simply you should be aware of how you are changing the shape of the data by combining them.

The second thing that changes as a result of combining dimensions is the set of neighbors surrounding any point. In two dimensions, each two-dimensional point has four immediate neighbors, when the two dimensions are combined; each point in a one-dimensional list has just two immediate neighbors. This is shown in Figure 3.19. Notice how in the top panel, where the data is arranged in two dimensions, the sales value for Newbury in February has four neighboring cells. Compare this with the lower panel, which shows the exact same data, only in one dimension. Here, the sales value for Newbury in February has only two neighbors. The changing of neighbors can affect analyses and graphical visualizations because they make use of information about neighbors. (For a discussion of multidimensional visualization, see Chapter 6. For a discussion of multidimensional analysis, see Chapter 15.)

(Note the difference in adjacency)

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridgewood</td>
<td>555</td>
<td>611</td>
<td>677</td>
</tr>
<tr>
<td>Newbury</td>
<td>490</td>
<td>539</td>
<td>590</td>
</tr>
<tr>
<td>Avon</td>
<td>220</td>
<td>243</td>
<td>271</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridgewood</td>
<td>555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newbury</td>
<td>490</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avon</td>
<td>220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ridgewood</td>
<td>611</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newbury</td>
<td>539</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avon</td>
<td>242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ridgewood</td>
<td>677</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newbury</td>
<td>590</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avon</td>
<td>271</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: A highlights the adjacent cells

Figure 3.19 Two different dimensional structurings of the same data.
One important thing does not change during the process. No matter how dimensions are combined, and no matter how the data grids are arranged, they make the same statements, claims, or propositions about the world whose truth or falsehood are a function of the same criteria. (For a discussion of dimensions, variables, and truth conditions, see Chapter 15.)

Now that we have learned how to combine dimensions, let us more fully demonstrate the process by adding two dimensions to our previous four-dimensional example. Figure 3.20 shows a six-dimensional data set consisting of products, times, stores, customers, variables, and scenarios. Figure 3.21 shows each dimension of Figure 3.20 connected to either a row, column, or page role of a three-dimensional grid display. Notice how multiple dimensions are combined in the row, column, and page dimensions of the grid display. The same visual display that works for three dimensions can easily be extended to work with N-dimensions. From here on, we will call this type of display a multidimensional grid display. Figures 3.22 and 3.23 show two different ways that the same six model dimensions can be mapped onto row, column, and page axes.

If you look at Figures 3.21 through 3.23 you will notice that there is always exactly one member represented from each model dimension shown on pages of the screen. There is no way around this fact. Try it. You will see that if you attempt to show more than one member from a dimension represented as a page, you will need to choose how to arrange the two or more

<table>
<thead>
<tr>
<th>Store</th>
<th>Cust.Type</th>
<th>Variable</th>
<th>Time</th>
<th>Scenario</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tables</td>
</tr>
<tr>
<td>Store 2</td>
<td>18 males</td>
<td>Margin</td>
<td>Jan</td>
<td>Actual</td>
<td>Desk</td>
</tr>
<tr>
<td>Store 3</td>
<td>18 females</td>
<td>Total sales</td>
<td>Feb</td>
<td>Planned</td>
<td>Chair</td>
</tr>
<tr>
<td>Store 4</td>
<td>(3)</td>
<td>Direct sales</td>
<td>Mar</td>
<td></td>
<td>Lamp</td>
</tr>
<tr>
<td>Store 5</td>
<td>Adult</td>
<td>Indirect sales</td>
<td>Apr</td>
<td></td>
<td>Shirt</td>
</tr>
<tr>
<td>Store 6</td>
<td>65 males</td>
<td>Cost</td>
<td>May</td>
<td></td>
<td>Shoe</td>
</tr>
<tr>
<td>Store 7</td>
<td>(4)</td>
<td></td>
<td>Jun</td>
<td></td>
<td>Sock</td>
</tr>
<tr>
<td>Store 8</td>
<td></td>
<td></td>
<td>Jul</td>
<td></td>
<td>Caviar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Aug</td>
<td></td>
<td>Coffee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sep</td>
<td></td>
<td>Wine</td>
</tr>
</tbody>
</table>

Figure 3.20 A six-dimensional MDS.
Figure 3.21 A six-dimensional data display.

Figure 3.22 A different six-dimensional data display of the same MDS.
members. And the only choices you have (in a flat static screen display) are across the rows or across the columns.

Multidimensional grids are so flexible that they can mimic any kind of (regular) table. After all, a table is just a special case of a grid where all or most of the dimensions are represented as columns. The table shown in Figure 3.24 can be thought of as a spreadsheet-like grid where five dimensions are represented as column headings whose members are the column values. And one dimension, in this case the variables dimension, has its members represented as column headings. Tables, such as the one shown in Figure 3.24, where one of the dimensions has its members represented as column headings and where the rest of the dimensions have their members represented as row values, are frequently (and in this book will be) called type one tables.2 (Using this same nomenclature, Figure 3.25, which is discussed below, would be called a type zero table because it has zero dimensions whose members are represented as column headings.)

Notice how every instance of a sales variable in the type one table is associated with an instance from every dimensional attribute. The identifier dimensions of the situation are represented as primary keys in the table. The variables of the situation are represented as nonkey attributes. Relationally

Figure 3.23 Yet a third six-dimensional data display of the same MDS.
Figure 3.24 Electronic capture of point-of-sale data in a type one table.

Speaking, this grid is a table in third normal form. This means that all of the nonkey attributes, in this case sales and costs, apply to or are uniquely identified by all of the primary or dimension keys. There are no sales variables that are not associated with a product and store and time. For every unique dimensional combination there exists one and only one instance of a variable.

Figure 3.25 A type zero table.
Finally, look at Figure 3.25, which shows a different table-like representation of the same data found in Figure 3.24. Although both table forms allocate one column per dimension, notice how in Figure 3.25 the individual variable names, "sales" and "units," are treated as the values of a column called variables. Also notice the addition of a "value" column whose row entries are the data values that, in the preceding table, were situated in each of the variables columns. (For an in-depth discussion of the value dimension and how it relates to the dimensionality of a multidimensional model, see Chapter 15.)

As a practical matter, the table in Figure 3.24 will be shorter in rows and wider in columns than the table in Figure 3.25. The long and narrow table in Figure 3.25 will use more space for storing keys because it has an extra key column.

The ability to easily change views of the same data by reconfiguring how dimensions are displayed is one of the great benefits of multidimensional systems. It is due to the separation of data structure, as represented in the MDS, from data display, as represented in the multidimensional grid. The actual method will be different from tool to tool, but the essence is the same. For example, the commands or actions to create Figure 3.21 are as follows:

1. Show variables nested within months along the rows of the screen.
2. Show scenarios nested within products along the columns of the screen.
3. Show stores and customer type along the pages of the screen.

**Analytical Screens**

While there is no such thing as a right or wrong grid display, there are some rules of thumb that you should keep in mind when analyzing multidimensional data in grid form. First, nesting dimensions across rows and columns consumes lots of screen resources relative to putting dimensions into screen pages. And because we still live in an age of limited screen reality, the more screen space is consumed displaying dimension members, the less space is left for displaying data. The less space left for displaying data, the more scrolling you need to do between screens to see the same data. And the more scrolling you need to perform, the harder it is to understand what you are looking at.

To maximize the degree to which everything on the screen is relevant, try keeping dimensions along pages unless you know you need to see more than one member at a time. And when you do need to nest multiple dimensions across rows and columns, since there is generally more usable vertical screen space than horizontal screen space, it is generally better to nest more dimensions across columns than across rows. For example, Figure 3.26 shows total sales by month with products, stores, and scenarios in the page. Imagine how
much more sales data you can see this way as compared with showing all the dimensions across rows and columns, as shown in Figure 3.22.

Second, ask yourself "What do I want to look at?" or "What am I trying to compare?" before deciding how to display information on the screen. For example, you may want to look at and compare actual costs across stores and time, for some product and customer type. If this is the case, you should set your page dimensions to that product and customer type you are analyzing and organize your display to show stores and times for actual costs, as shown in Figure 3.27.

Now, what would you do if you were trying to compare the ratio between sales and advertising costs for low- and high-priced products across stores and times? You could try to see whether the returns on advertising for low- and high-priced products varied across stores or times. Perhaps it costs less to sell expensive products in high-income area stores, or perhaps there were better returns on advertising cheaper products during holiday time periods. How would you want to set up the screen to look at this? The complicating factor here is that what you are looking at—variables by product category—is itself a two-dimensional structure.
Figure 3.27 Arranging data to compare costs across stores and time.

The way to show this type of information on the screen is to put one of the looked-at dimensions in the most nested position along one visible row or column axis and to put the other looked-at dimension across the other visible column or row axis, as shown in Figure 3.28. The more you think in terms of complicated multidimensional structures, such as the one shown above in Figure 3.28, the more you may want to use graphical visualization techniques for displaying the information. (These techniques are discussed in Chapter 6).

Although any combination of dimensions may be mapped to any combination of rows, columns, and pages, you will be most productive when you think about what you are trying to learn before defining screen representations for your data.

Summary of Dimensions
We have seen that dimensions are logical factors or identifying attributes of measurable events or things that we track. We call these dimensions identi-
fier dimensions. We have also seen dimensions that identify what we track in a situation. We call these variable dimensions. The things tracked are called variables or members of the variables dimension. As distinguished from physical dimensions, which are based on angles and limited to three, logical dimensions have no such limit.

Frequently, the number of dimensions in a data set exceeds the three dimensions found in a typical row, column, and page screen display. Fortunately, multidimensional software enables multiple dimensions of information to be combined onto each row, column, and page axis of a display device, thus making it possible to visualize and understand a multidimensional data set in terms of information presented on a flat computer screen.

The ability of multidimensional software to model multiple dimensions of information and to handle the user representation of the information makes it better suited for working with complex data sets than either SQL databases or traditional spreadsheets.

Hierarchical Dimensions

Hierarchies Are the Backbone of Aggregating

The second major feature common to most multidimensional software tools, after that of support for multiple dimensions, is that of support for hierarchies. A hierarchy is an attribute of a dimension. Most dimensions have a hierarchical or multilevel structure. Similar concepts include abstraction, grouping, aggregation, and consolidation.

In an informal way, everybody is familiar with some hierarchical dimensions. Time, which we think of, for example, in hours, days, weeks, months, quarters, and years, forms a multilevel hierarchical dimension. Geography, which we may think of in terms of neighborhoods, cities, states, and countries, forms a multilevel hierarchical dimension. And corporate reporting structures, which frequently include a task, project, department, business unit, and company level, form a hierarchical dimension as well.

In contrast, a scenario dimension, which is common to most business models, typically has a small number of members such as an actual member, a planned or perhaps several planned members, and a variance member for each combination of planned and actual members. It would almost never be portrayed hierarchically.

In business, as in most types of activity, hierarchies are a necessity of life. It would be impossible to run a company effectively if all the company's data were confined to the transaction level. Whether it is done in the computer or