CHAPTER 4

The Implementation Model

In this chapter we take the step from the theoretical to the actual. In the preceding chapter we built the analysis and design models. These models provided a conceptual view of the objects within the scope of the data warehouse. In their creation we developed an understanding of how things work. We now take these conceptual models and build the implementation model.

The structure we are about to design supports the way in which strategists view their environment. In Chapter 1 we discussed how the strategist is looking for behaviors in the marketplace, searching for ways to understand the current environment and predict its future. As we shall see in this chapter, the architect must create a data structure that is natural to the business strategist. It must allow strategists to think in terms of the objects they are studying. To achieve this end the warehouse architect structures the data dimensionally. Dimensions draw borders around the area of study, and they also provide a way to define specific relationships between objects within the business environment. The construction of the implementation model builds this multidimensional structure, known as a star schema, within a relational database.

To fully appreciate the way in which we use dimensional analysis and construct the implementation model, let's begin at the very beginning. When we count, we begin with one, two, three. When we build, we begin

with dimensionality, dimensionality. The first section of this chapter discusses dimensionality in its most basic form. In this section we explore coordinate systems and how they are used to define space. In the second section we relate dimensionality to information systems. We first explore traditional applications and how they fall short in providing a multidimensional environment for the strategist. Then we discuss multidimensional databases (MDDB) as an alternative. Although multidimensional systems provide the desired environment for the business strategist, certain limitations make them unsuitable for many environments. Another solution to multidimensional analysis is the star schema. The last two sections of this chapter present the star schema as a means to implement multidimensionality in a relational environment.

The structure of this chapter is important. It is simple enough to present the basic star schema. It would not take an entire chapter to explain the tables and how they are related. What is most important, however, is an understanding of what the star schema creates. The beauty of the star schema is its ability to provide a very complex multidimensional view of data in a very simple structure. The goal of this chapter is to develop an appreciation for the multidimensional nature of the star schema. This chapter develops the concept that the star is in fact a multidimensional model implemented within a relational environment. It shows how the dimensions create a space within which the strategist can perform analyses.

4.1 Dimensionality

In this section we will discuss multidimensional data and what we mean by multidimensional analysis. A *dimension* is an axis of a coordinate system that binds a space. The prefix *multi-* means multiple or more than one. *Multidimensionality*, then, refers to more than one dimension or to binding a space in multiple ways. This is natural to our way of thinking. The universe in which we live exists in multiple dimensions—height, depth, width, and time. In our daily lives we deal with these four dimensions, although we may not be consciously aware of them as we go about our business. What about data, though? How does this all relate to data and our understanding of the objects with an organization's environment?

Let's step back in time for a moment to review what we all learned in junior high. We studied the cartesian coordinate system, in which we started out working with two dimensions. We called those dimensions the *X* and *Y* axes and divided each into equal units, as shown in Figure 4–1. The space *dimensioned by* these axes is a plane. That is, the plane is bound, or contained within, these axes.

Envision a space whose edges are the dimensions. Every point within that space has some relationship to the binding dimensions. This relationship is defined by the point's coordinates. To locate a point using its coordinates we move along each axis the specified number of units and draw a line perpendicular to the axis at that coordinate. The intersection of these lines is our point. In Figure 4–1 we have the point (3, 4). We have moved 3 units along the *X* axis and 4 units along the *Y*. As you may recall, in the cartesian coordinate system the first number represents the *X*-axis and the second the *Y*.

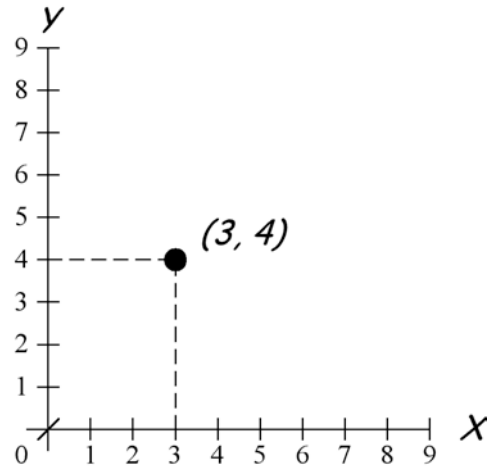

Figure 4–1 Cartesian coordinate system.

The point at which the two axes meet, point (0, 0), is referred to as the point of origin. The set of axes emanating from a single point of origin is a coordinate system. When we plot a point or a graph on some coordinate system, we are expressing that the point's *X*-coordinate is some function of

its *Y*-coordinate. This is true of all dimensions within the coordinate system; all dimensions are related to one another.

Perhaps we are representing a sine wave or the fluctuations of the stock market over a period of time. The points along the *X*-axis are related to the points along the *Y*-axis by some function. Look at any stock market report and you will see this relation. Typically what you will see is the *X*-axis representing time and the *Y*-axis representing the value of the stock market. The relation between the two axes is the set of all possible values of the market for all points within a given time period.

In looking at these dimensions, note how they represent objects of analysis. The objects in the stock market report are time and the value of the market. When we organize related objects into dimensions, creating a coordinate system, the space formed by this coordinate system is the *analysis space*. In our stock market report the points in the analysis space are the values of the market at a particular time.

In a two-dimensional world we are limited to existence on a plane. This world exists in only two dimensions: height and width. There is no depth. This world of two dimensions differs drastically not only from our physical existence but also from the way in which we view our business environment. Organizations work in a universe of complex issues composed of not just two or even three objects, but many objects. We must work with all of them in our analysis to understand their impact on the organization.

To add another object we must add another axis—that is, another dimension. We have done this in Figure 4–2. We can go beyond three dimensions and extend our analysis to as many dimensions as we like. The general rule of thumb, however, is that you should not have more than six or seven dimensions in any one analysis. As you will see later in this chapter, working in many dimensions at one time can become quite confusing and result in clouding an issue that you actually hope to clarify.

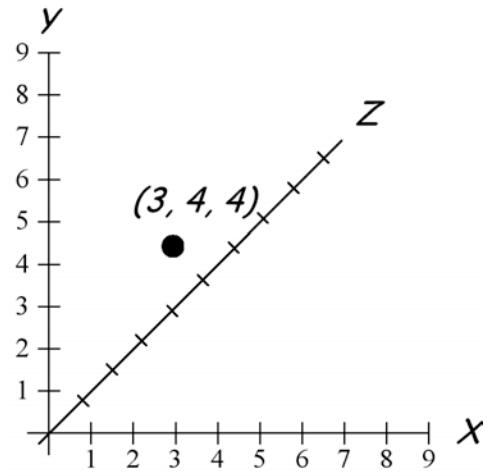

Figure 4–2 Three-dimensional coordinate system.

4.2 Dimensionality and Information Systems

When we look at a relational table we can almost see a two-dimensional space. We have columns that serve as the *X*-axis and rows that serve as the *Y*. In actuality a relational table has a single dimension. In Figure 4–3 we see a typical relational database table of employee records. In a normalized database each row in the table represents a different employee. The columns represent different characteristics of that employee—name, address, and phone number, for example. The primary key of this table is the employee ID. Every employee characteristic is referenced by this primary key. We locate the record by its single coordinate, employee ID. There is no second dimension. Compare the relational table with a two-dimensional space. The second axis in a two dimensional space, the *X*-axis, contains all the possible values for the second object. The columns of the relational table, however, contain different characteristics of the same *Y*-axis object, employee.

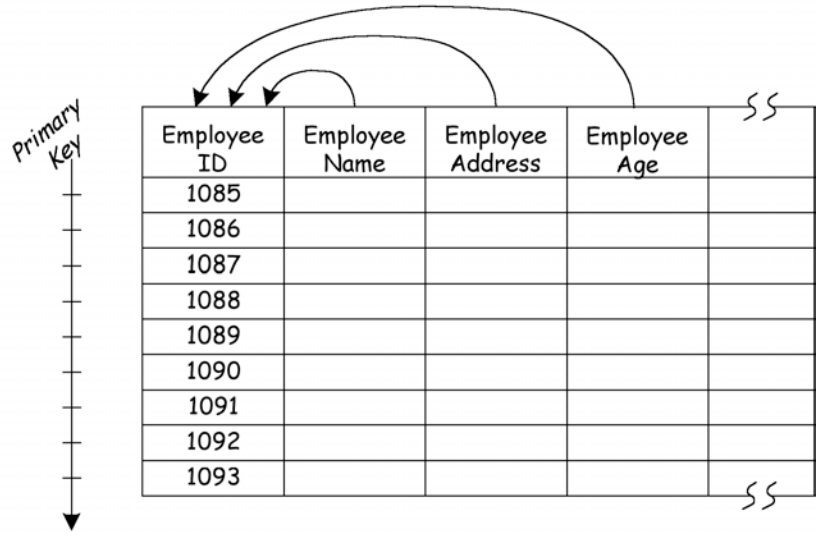

Figure 4–3 Relational database table.

A possible common multidimensional tool is the spreadsheet. Note that it is only a possible multidimensional tool. A spreadsheet that simply lays out data and performs a calculation has no dimensionality. If, for example, a CFO uses a single spreadsheet to lay out an income statement, there is no real axis to determine upon which row or in which column a data item will appear. We could also lay out a table of a relational database on a spreadsheet, in which case we would be working in a single dimension. Spreadsheets can also be used in a multidimensional fashion. Figure 4–4 demonstrates one such use. In this diagram we see that the *Y*-axis represents the different products for a chain of car dealerships. Each column, the *X*-axis, in the spreadsheet represents time. The cells define the sales for a particular product for a particular month. The individual worksheets each represent a particular dealership. These worksheets could be viewed as the *Z*-axis.

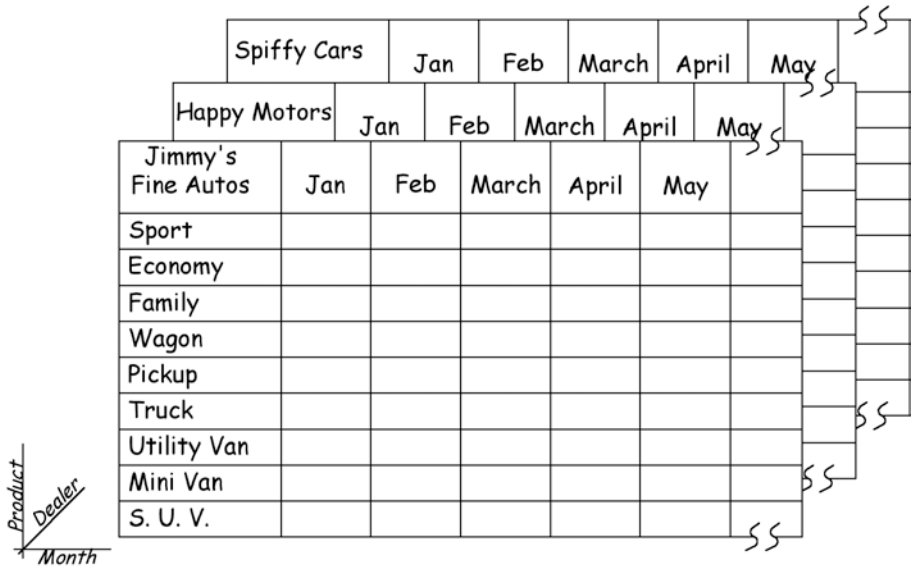

Figure 4–4 Multidimensional spreadsheet for auto sales.

Although spreadsheets do lend themselves to working in multiple dimensions, they are limited to only three in one file. Working with multiple dimensions in a spreadsheet is cumbersome even for the most technically astute business strategist. Consider what it would take in our example to show a user the sales by product—say, the sales for sports cars—for all dealerships in the month of March.

In the first chapter we discussed how the decision support system must be a tool that facilitates the business strategist's interactive dialog with the data. Imagine your CEO or CFO attempting to gather answers using the spreadsheet presented in Figure 4–4.

4.2.1 Multidimensional Databases

As we have stressed throughout this book, the strategist is interested in the behavior of objects. In discussing object-oriented analysis in Chapter 2 we showed how the strategist will look at a set of objects and group them into categories. The example shown in Figure 4–4 organizes objects into three

dimensions. The products are ordered along the *Y*-dimension, time along the *X* and dealerships along the *Z*. Each dimension represents a different object. The relationship between the objects is described within the cells of the worksheet. A dealership sold a car. Information about the sale is stored in the cell. Perhaps the cell contains the price for which the car sold or the dealer's profit from that sale. In either case we see the world from the eyes of the business strategist, objects interacting with objects.

The question is how to create an environment in which the strategist can work with these multiple dimensions. As we have seen, traditional databases and spreadsheets fall short of this goal. Typical normalized relational databases work in one dimension. A spreadsheet can work in only three dimensions, and even then it is cumbersome. Just our simple auto-sales example requires more than three dimensions. Let's refer back to the object model in Figure 3–4. We have four objects: people, cars, dealership, and payment. We also need to add the all-important dimension of time. This gives our analysis a total of five dimensions. A strategist might ask *what types of cars* are being sold by *which dealerships* in *which months* to *what customer type*.

Figure 4–5(c) presents the resultant analysis space. Since things get complicated once we move beyond three dimensions, we will take a moment to build this space. We begin with Figure 4–5(a). In it we present a simple data cube. The data cube is dimensioned by the product, time, and dealership dimensions. Adding a fourth dimension is similar to creating a data cube of data cubes; this is referred to as a hypercube. We see this in Figure 4–5(b). One could imagine the lower front left corner of one cube touching the upper back right corner of another. The new fourth dimension moves up and at an angle to the other three dimensions. To build the fifth dimension we create a hypercube of hypercubes. This is presented in Figure 4–5(c).

It is rather difficult in two dimensions to present four- and five-dimensional objects. Although I have presented hypercubes as best I could here, future representations will be limited to three dimensions, data cubes.

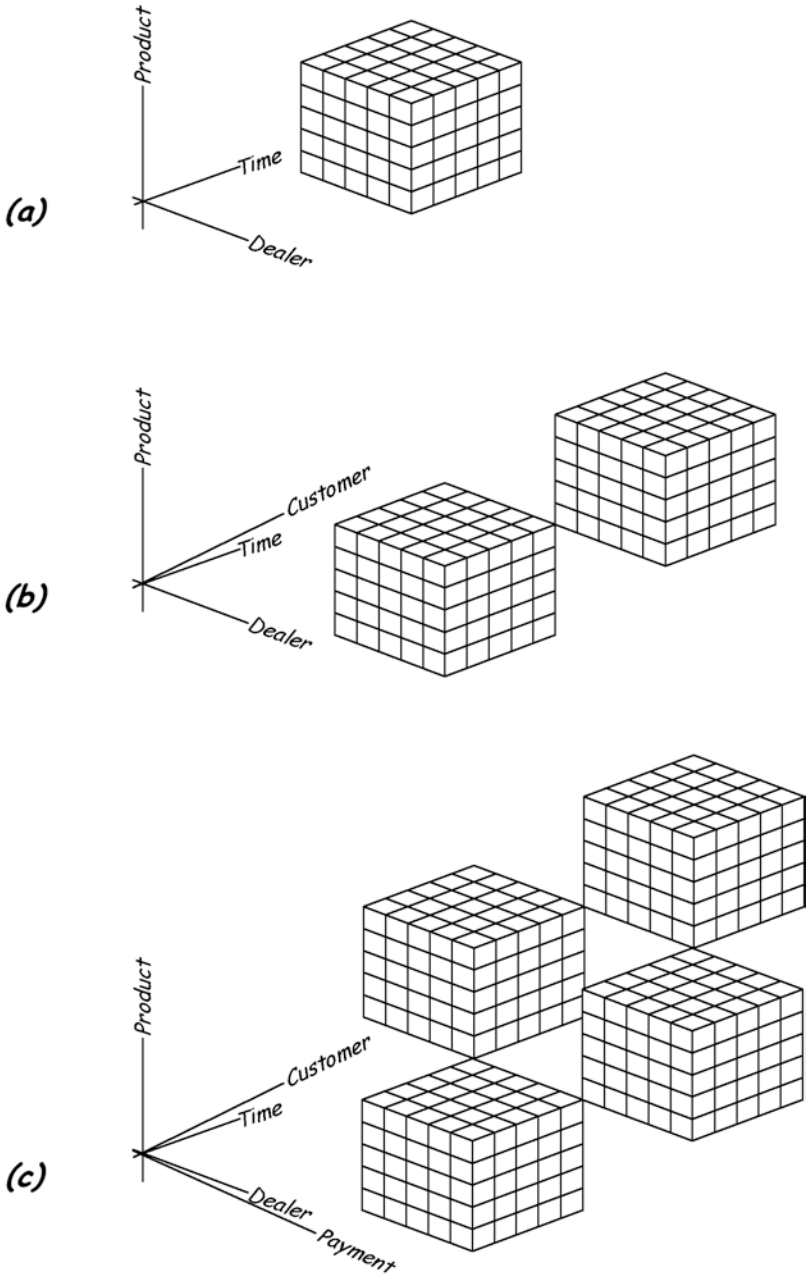

Figure 4–5 Auto sales hypercube construction.

Technically, the term data *cube* is a misnomer. A cube is a three-dimensional object whose sides are of equal length. It is a three-dimensional extension of a square, which is a quadrilateral having four right angles and four sides of equal length. Squares are actually a subset of rectangles, which are quadrilaterals having four right angles. A rectangle does not have the additional constraint of having equal sides. The three-dimensional extension of a rectangle is a parallelepiped. The length of a data *cube*'s side is determined by the number of values for that dimension within that analysis space. The sides of a data *cube*, therefore, would most probably not be equal. The more technically correct terms would be data parallelepipeds and hyperparallelepipeds. As it is, data warehouse architects speak in a strange enough language; I believe that for simplicity's sake we can all live with the terms data cubes and hypercubes.

The points within the analysis space are the individual sales. The business strategist thinks of these sales in terms of a specific car sold by a specific dealership in a specific month. The units along an axis are referred to as *values*. Just as in a coordinate system, the multidimensional database defines a cell in the data cube by the values of its dimensions. To access a specific sale the strategist defines the coordinates of that sale. The coordinates are the values for each of the respective dimensions: car, dealership, and time. For example, we specify the values (Testosterone 400, Don's Spiffy Cars, July '99) to access the cell containing the sales of the Testosterone 500 for Spiffy Cars in July 1999. Figure 4–6 demonstrates how we reference the sale.

The business strategist now has a tool to analyze the relationships between the objects in the business space. The cube becomes an object that can be manipulated. Rather than being lost in the details of storing and retrieving data, the strategist is able to focus on the thing that is most important, the analysis of the data. In looking at Figure 4–6 one can almost imagine the business strategist holding this cube. Simply by turning it, the strategist can view the business space from different perspectives. Perhaps the strategist could even break off sections of the cube and perform analysis on subsets of the data. Well, that is exactly the point of multidimensional

data. The strategist can now easily gain different perspectives on the data as well as focus his or her analysis on subsets of the data.

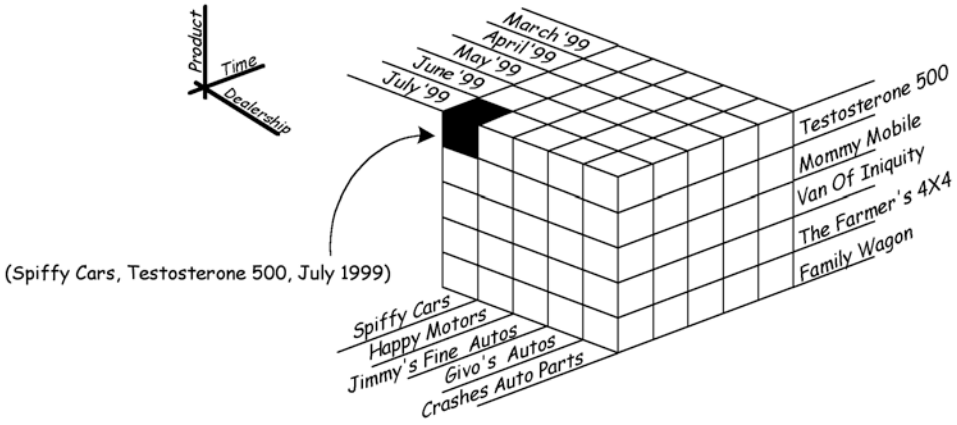

Figure 4–6 Data cube for auto sales model.

4.2.2 Working in Multiple Dimensions

Think of some of the possible questions a business strategist might attempt to answer with a data cube. Perhaps the district manager is interested in how well Spiffy Cars performed over the past few months. In such a case the spreadsheets presented in Figure 4–4 would perform well. What if the district manager would like to see the performance of all dealerships in the district for the month of May for all products? The spreadsheet distributes the sales for the month across the different worksheets; creating a view that presented all sales for an individual month would be difficult. A multidimensional tool, however, makes such a change in perspective easy.

By simply rotating, or pivoting, the cube you could change the perspective of the data. As we can see in Figure 4–7, rotation about one axis switches the other axes. In Figure 4–7(a), for example, rotation about the *Z*-axis, dealership, moves product to the *X*-axis and month to the *Y*. Similarly, we could rotate the cube about the *Y*- and *X*-axes. We could easily combine these rotations to provide any perspective that might be convenient to our analysis. The cube could first be rotated about the *Z*-axis and then about the *Y*-axis.

The number of possible different views of an analysis space is a function of the number of dimensions used to construct the space. For example, we could create two views out of a two-dimensional analysis space. We could view product along the *X*-axis and month along the *Y*-axis, or we could view month along the *X* and product along the *Y*. Data cubes—three dimensions—have six possible different views. Hypercubes of four dimensions have twenty-four possible views.

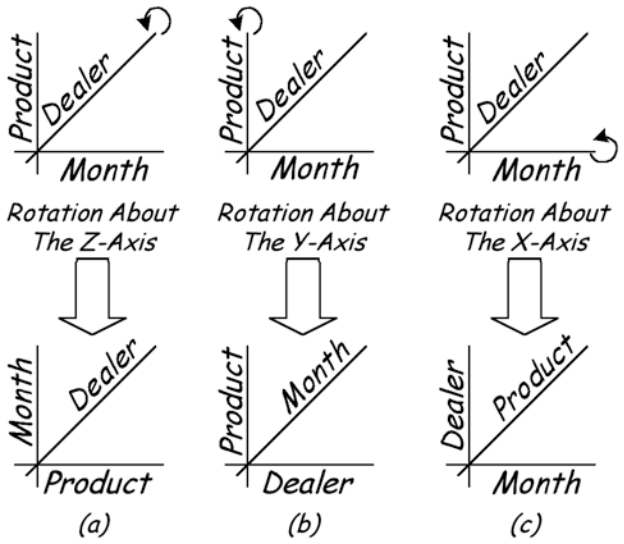

Figure 4–7 Rotation about an axis.

Let's return to the business strategist to see how we can use this to our advantage. To change the display to show the sale of all products in all dealerships for a particular month, we would perform the rotation in Figure 4–7(b). This would display all the products along the *Y*-axis and the dealerships along the *X*-axis. Units of time, or months, are displayed along the *Z*-axis. To move from one month to another we would simply change the value of the time axis, the *Z*-coordinate.

Another benefit of multidimensional databases is the ability to segment the data. As shown in Figure 4–8, the business strategist can easily extract a subset of the data from the entire cube. This is referred to as *slicing*. In

essence we are contracting the analysis space. The analysis space initially contains all possible values for each of the dimensions. We create a slice of this space by defining a subset of values for one or more dimensions. In our example diagram we see where the number of rows has been limited to just two products, while all the values from the time and dealership dimensions have been maintained. We could do this for the other dimensions of the analysis space as well. In Figure 4–8 we have restricted the number of dealerships to three values while accepting all values for the time and product dimension. Although our examples slice along one dimension at a time, there is nothing to prohibit a strategist from slicing along multiple dimensions. Slicing along multiple dimensions at one time is sometimes referred to as dicing.

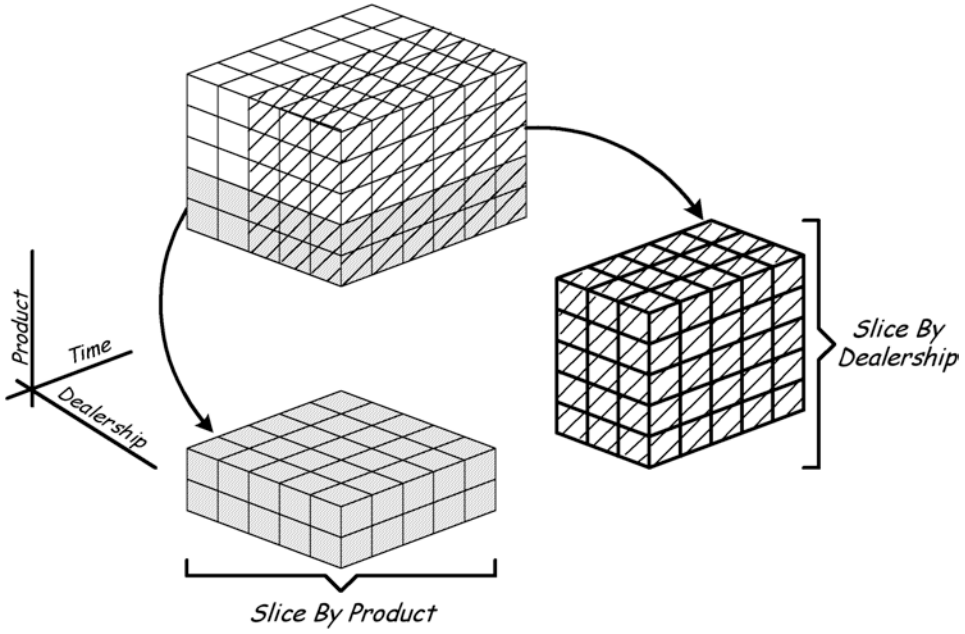

Figure 4–8 Slicing data.

The business strategist would not think in terms of cutting dimensions along rows and columns. The strategist would think in terms of objects. He or she might be interested in an analysis space of one fiscal quarter for all

sport utility vehicles in the Northwest Region. While architects realize that these objects are in fact values of a dimension, strategists are simply working with the objects in their day-to-day world.

There are actually some subtle benefits to slicing data. First, it provides better performance. When working with a slice, we are defining a subset of the data. We can rotate this smaller set of data faster than the entire cube. Slicing also provides better control of the data. It extracts a subset of the data from the multidimensional database and passes it to the user. Users only have access to the information they are given. They can't see what they haven't received.

In these examples the strategist is examining very detailed data—specific dealerships, products, and months. Business strategists, however, are interested in differing levels of detail. We saw this earlier in our case study. The CEO and the CFO were interested in a *dashboard* of their organization's leading indicators. The regional managers were interested in a regional level of details. Lower-level managers and individual contributors were interested in very detailed information, often down to the level of individual transactions. Typically the level of detail is commensurate with the business strategist's position within the organization. High-level strategists, such as CEOs, CFOs, and CIOs are interested in high-level summaries of the organization's performance. Individual contributors and line managers are interested in the most detailed data.

Data can be summed along the different axes of the data cube to provide this higher-level view. This is presented in Figure 4–9. We could determine the total sales for each dealer for each month for all products by simply summing the cells in the columns. To determine the total sales for each car for each dealer for all time, we perform our summation on the cells along the *Z*-axis. There are times, however, when we like to group the sums according to some characteristic of the objects, such as the total sales of sport cars of dealerships in the Western Region. This is where aggregation comes into play.

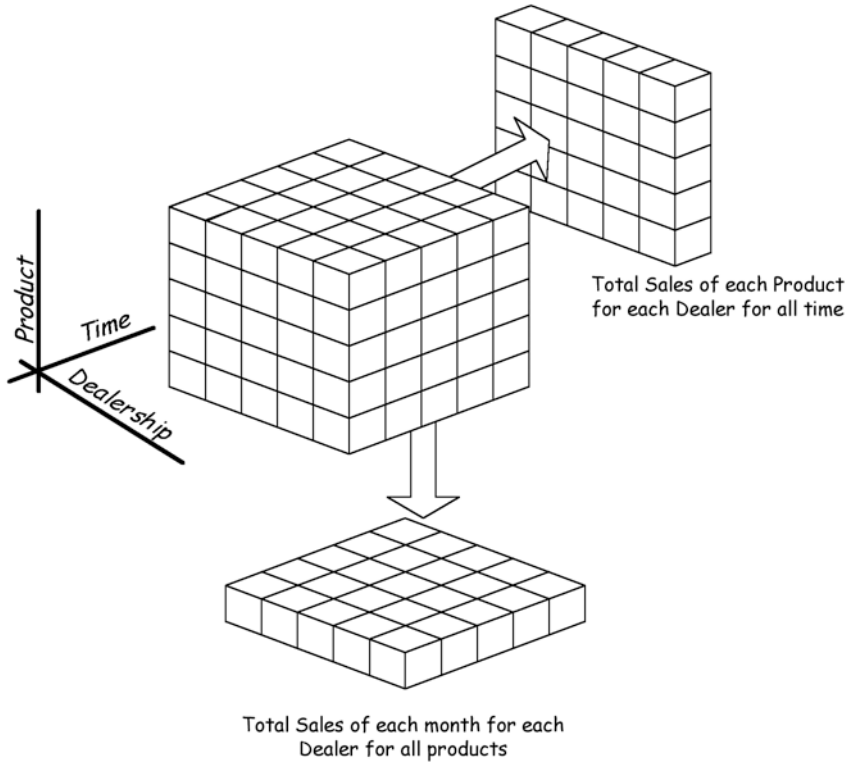

Figure 4–9 Data summation.

In Chapter 2 our discussion of objects presented the concept of aggregation. We defined aggregation as a unique relationship between objects that form a new object. For example, a disk drive, memory, terminal, motherboard, CPU, and keyboard are combined to make a computer. The computer is an aggregation. In the object model presented in Figure 3–4 we see two sets of objects that are aggregations. These are the objects that were created by joining other objects together. Refer back to this diagram to see if you could identify the two aggregations.

The organization itself is the first aggregation. It is composed of regions, which are composed of districts, which are composed of dealerships. These levels of aggregation parallel management's level of interest. The CEO is interested in the performance of the company and at times may

look down into the region. The regional manager is interested in the region and at times may look down into the district. This flows down the corporate hierarchy to the individual sales rep, who is interested in the most detailed level of information. Another term for aggregation is *roll-up*. A roll-up is a summation of cells to provide a higher-level view of the data. We roll-up the dealership sales numbers into district numbers. The district numbers are rolled-up into regional numbers.

Figure 4–10 demonstrates how an aggregation is performed in a multidimensional database. The lowest level of detail is the dealership data. This cube contains the sales data for the individual dealerships. This data is then divided by district and summed. The sales data for all the dealers for each district is added together into another cube. This process is then repeated for each level of aggregation up the hierarchy. Note that the other dimensions of the cube have not changed. If the detailed data cube's other dimensions were months for time and individual makes of cars for products, the aggregate cubes would have the same time and product dimensions. The aggregations of these other dimensions are built separately.

Aggregations can be created for each of the hierarchies within the data warehouse. Time, customer, and product dimensions each have a hierarchy for which aggregations can be created. They can also be created across different levels of hierarchies. A sum can be created for each district or region by product or product category. These sums can be compiled for any desired time period, weekly, monthly, or even annually.

The reverse of a roll-up is a *drill-down*, which is the examination of the data at a deeper level of detail. Perhaps a CEO notices on the corporate dashboard that there is a problem in a particular region. The drill-down will allow the CEO to select that indicator and examine the detailed data underneath. In a multidimensional environment the drill-down moves from one cube to next, slicing the detailed data to only those records that formed the aggregation.

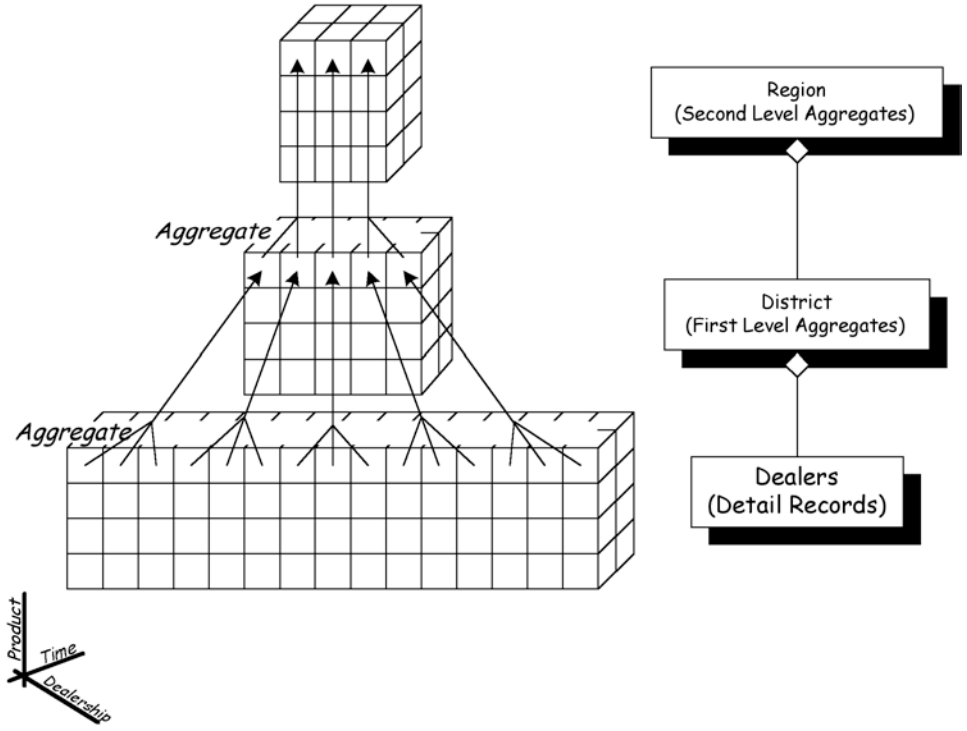

Figure 4–10 Data aggregation.

4.2.3 Multidimensional Database Issues

As we can see, multidimensional databases offer the business strategist significant advantages over traditional information systems. Considering all these benefits, one might ask why all data warehouses aren't implemented with multidimensional databases. As with most things in life, there are trade-offs. The most significant drawback to multidimensional databases is the size requirements. Adding dimensions to an analysis space and values to existing dimensions increases the size of a multidimensional database much more dramatically than adding attributes to a relational table.

The number of cells in a hypercube is the product of the number of values in each of the dimensions defining the analysis space. The cells increase geometrically with each additional value. Figure 4–11 demonstrates

this relationship. Let us assume that we have 100 different cars sold by 200 different dealerships. We collect data from each of these dealerships on a daily basis. The number of cells in the data cube would therefore equal the number of dealerships times the number of cars times the number of days. If we retained just one year's worth of information the data cube would contain 7,300,000 cells (100 × 200 × 365). If the data warehouse architect were to add payment type as a fourth dimension we would see a dramatic increase in the number of cells. Just three different values for the payment-type dimension increases the number of cells to 21,900,000 (100 × 200 × 365 × 3). Obviously the number of cells could increase quite rapidly for even our modest data cube. These calculations do not include any aggregations associated with the cube, either. As we add these supporting data structures, the size requirements continue to increase.

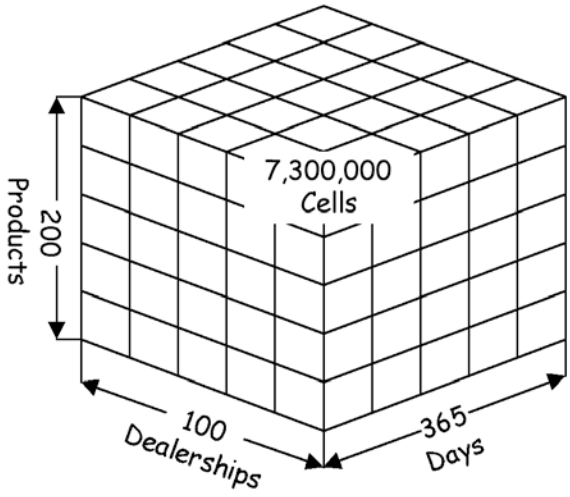

Figure 4–11 Data cube size.

Unlike records in a relational database, a cell is created whether it contains a value or not. As a result, many of the cells in a multidimensional database are empty. Consider our simple auto-sales example. Few dealerships will actually sell at least one car of each type in a particular day. The probability that more than one dealership will do the same is almost non-

existent. Consider the example of a national tire-store chain. Stores in New York and Minnesota will probably be selling a good number of snow tires for at least half the year. On the other hand, stores in Southern California or Nevada probably will not be selling a lot of snow tires, especially in June and July. The cells for these stores for these products will be empty. We would say that the data in the data cube is sparse; there is a lot of space between the data points in the cube. We can see that having many multidimensional databases can give rise to creating very large storage areas that have significant percentages of empty space.

Multidimensional databases are beginning to deal with the problem of sparse data, otherwise known as *sparcity*. Many of the approaches used, however, require the data warehouse architect to anticipate which dimensions will have sparse data.

4.2.4 Big Brother and the Building Company's Analysis Space

In this section we will build the BBBC analysis space. We construct this space by identifying the objects in the object model that are to be part of our analysis. These objects will dimension, or define, the analysis space. We can then go on to develop the implementation model that provides the structure to our database.

In reviewing the BBBC object model we see that there are three main objects: broker, property, and client. Although it is not specifically called out in the analysis model, we know simply by the fact that we are implementing a data warehouse that we will want to examine the data in terms of the time dimension. With rare exceptions, all analysis spaces will contain the dimension of time. We will also see that the structure of the time dimension can be fairly standardized from one implementation to the next. The resulting analysis space is presented in Figure 4–12.

It may seem at first glimpse that we have not included all the objects in our object model. So let's look at how we selected these dimensions. First, we discussed in the preceding chapter that we are trying to achieve the greatest possible impact with our first iteration of the data warehouse. In

Section 3.2.2.1, "Big Brother & The Building Company Analysis" on page 94, we agreed with the user community that to achieve this end we will limit the initial scope to the brokerage business. We chose brokerage because it provides a significantly larger percentage of revenue to BBBC than any of the other business units. A slight increase in the profitability of brokerage will have greater effect on the overall organization than increases in the other business units.

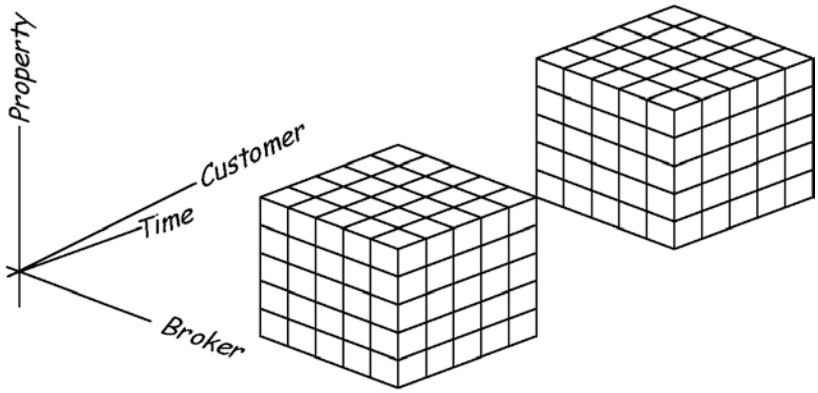

Figure 4–12 BBBC analysis space.

We also see that we have several objects that are a superclass. These are the broker, client, and property objects. Attributes that are common to all subclasses of a superclass are contained within the superclass. Each subclass, however, has its own private set of attributes. In constructing a dimension we reverse what we have done in creating the superclass. We collapse all the subclasses into one object that contains all the attributes of all the subclasses. Although it may seem as if we are simply creating something to subsequently destroy it, we are actually making progress. In the process of expanding the different subclasses out in the object model we gain a better understanding of the variety of attributes pertaining to the object. When we collapse the subclasses into one object, we can contain the attributes of all subclasses in one dimension. This simplifies the overall design of the structure. An example of a superclass and a subclass within the BBBC analysis space is property. All properties have an address and an owner. These would be attributes of the superclass property. An industrial property, how-

ever, may have the attribute of a crane. The crane would be an attribute of the industrial subclass property.

Management will want to view information based on the different levels within the organization. The organization, therefore, will be implemented as an aggregate. In examining the object model we see that this aggregation extends from the broker up to the company. The company is composed of business units that are composed of profit centers. The profit centers are a superclass. The subclass brokerage profit center is also an aggregate, which is composed of brokers. Time is also part of an aggregation. It is only logical that the business strategist will want the ability to see information on each of the different time aggregates: weekly, monthly, quarterly and annually. We should plan on it. We will see in the next chapter how to structure these aggregates.

Our final aggregation is that of client. A client may be a completely independent company or a subsidiary of a large corporation. BBBC Management stressed that it was important to look at activities with an individual client as well as any parent company or companies. Management would like to capture this information and roll-up the performance for the entire corporation as well as its subsidiaries. The customer dimension, therefore, will require an aggregation as well.

In the next section we will review how to create an implementation model. In that model we will design the database structures that will store the information for the BBBC analysis space.

4.3 Star Schema

As we shall see, the star schema overcomes the shortcomings of a multidimensional database by providing a multidimensional analysis space within a relational database. The structure of the star schema is actually quite simple. Figure 4–13 presents a star for our auto-sales example. In this diagram we see two basic types of tables, dimension and fact. The primary keys of the dimension tables are foreign keys in the fact table. Each of these tables correlates nicely to the basic elements of the hypercubes discussed earlier in this chapter. The dimension tables are the dimensions of our cube, our axes. The fact-table records are the cells.

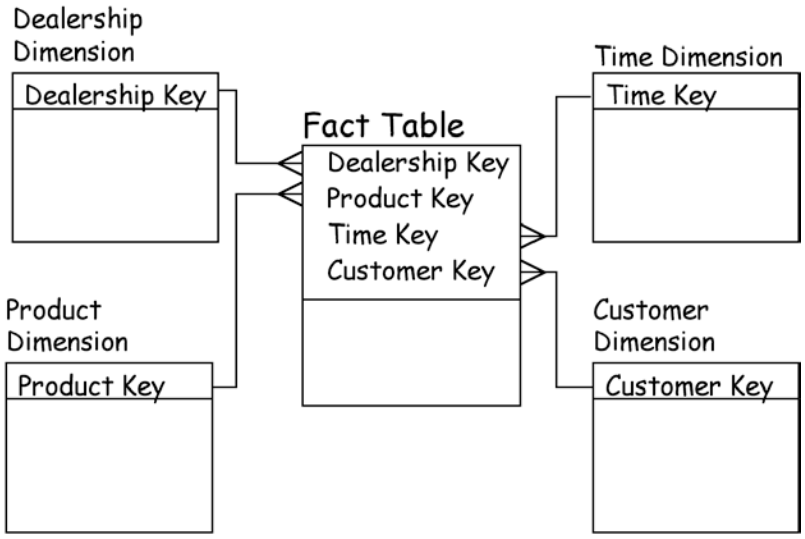

Figure 4–13 Star schema.

At this point we should briefly discuss the keys that link the tables in the star. The temptation is to use the primary keys from the operational environment as the data warehouse primary keys. The consistency between the two environments makes the option attractive. Don't do it! While it is useful to maintain the operational primary keys as attributes of the objects in the data warehouse, they should not be used as data warehouse keys. Data warehouse keys should be anonymous, system-generated keys. These are often referred to as surrogate or synthetic keys. We will discuss the use of keys in more detail in Chapters 5 and 6.

The star schema creates a multidimensional space out of relational tables. This is presented in Figure 4–14. As we noted in the beginning of this chapter, to form a coordinate system the dimensions of the coordinate system must be related to one another. We also noted how relational tables are one-dimensional structures. The relationship between the dimensions in the star schema is described within the fact table. Just as a coordinate system joins related dimensions, the star schema joins the single-dimensional dimension tables to one another. This forms a multidimensional analysis

space within the relational database. Examining Figure 4–14, one could almost imagine these tables as planes *glued* together to form a cube. The glue is the fact table. It does this with keys. The primary key of the fact table is a composite of the dimension-table keys. How a record or set of records in one dimension is related to a record or set of records in another is defined by the keys contained in the primary key of the fact-table record.

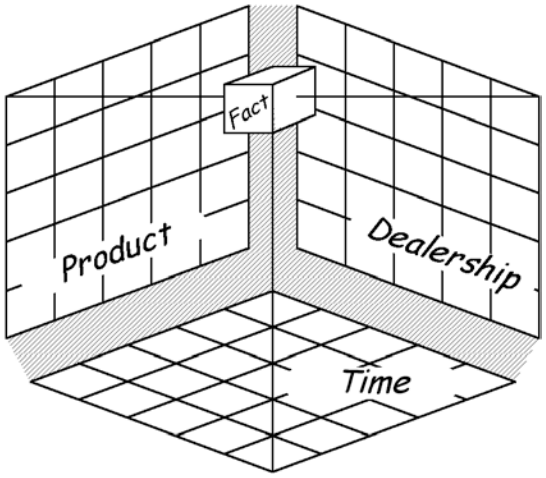

Figure 4–14 Star schema dimensionality.

The fact table does more than just bind the dimensions together to form a multidimensional space. The fact table contains the facts—numbers that describe the characteristics of the relationship between the dimensions. It contains the points in the analysis space. In our auto-sales example, the product, dealership and time dimension are related at a specific point. If we are tracking sales, the point at which Spiffy Cars is related to the Testosterone 500 and July 1999 is defined by the fact in the fact-table record. In this record we might see number of cars sold or perhaps the total sales revenue for those cars by that dealership. These are the characteristics of the relationship between those three objects. Fact-table records will tend to be much shorter than dimension-table records, as we shall see in the discussion to follow. The length of the fact table, however, will be much longer than the length of its surrounding dimensions.

As we can see, the interesting part of the star schema to the business strategist is the fact table. This is where the data is located. To get there, though, we need the dimensions. The dimensions in both the hypercube and star schema are entry points. When working with hypercubes we access the cells by specifying the values (coordinates) of the dimensions. In a similar fashion we define the records of interest within the fact table via the dimension specifications. Although the fact table is where the action is, everything begins with the dimension. One could imagine the dimension tables as doorways through which the business strategist enters a room containing the facts, the fact table.

As we have said earlier, the values of the dimension are the different objects of that dimension. In the star schema each record or row of the dimension table is the equivalent of the dimension values, the objects of that dimension. The rows in the product table of our auto-sales example are the different makes cars sold by the dealerships. To gather facts on any of these products we select the product(s) of interest from the dimension table. The primary keys of the product records are the foreign keys in the fact table.

At times the business strategist may select specific products or groups of products. At other times, however, the strategist may select products based on a set of common characteristics or attributes. The strategist may be interested in the sales data for cars with four-cylinder engines or cars with certain performance characteristics. These attributes are included as columns of the dimension table. The attributes included in the dimension tables are only those that may be the basis of some future analysis. In the customer dimension we may want to include demographic information, but we would probably not include descriptive information such as a client name or social security number. We will discuss attribute selection in more detail in the next chapter. What we need to understand at this point is that the selection of records from the dimension table is based on the attributes of the objects contained in that table's record. These records in turn will provide the keys for entry into the fact table.

A typical analysis may ask how well certain cars with a specific engine size, seating capacity, and fuel efficiency sold to a specific market segment.

These are all object attributes. All the attributes of an objects should be located within that object's dimension-table record. Normalizing these attributes across several tables will greatly complicate the data-retrieval process. Architects coming from the OLTP environment may balk at this level of denormalization. After all, won't this create long tables of wide records that will ultimately lead to poor performance? Well, yes and no. Yes, this will lead to tables of wide records, but it will not lead to poor performance. Denormalizing the dimension tables will actually improve performance. The queries of the database will be kept much simpler by keeping all the attributes in one place, thus reducing the number of joins. That's the short answer.

In looking at this question in more detail, keep in mind that normalization is a tool of the OLTP world. Distributing the data across a normalized structure improves the performance of searching for a specific record. It takes much longer to search through a long table of very wide records than a table of the same length with significantly shorter records. It also eliminates the data redundancy that occurs in environments where data is frequently updated. The difference within the star schema is that the number of records in the dimension table is *relatively* static and small. In our simple auto-sales example the product, dealership and time dimensions will not grow nearly as fast as the fact table. The relationship between the dimension and fact tables is one-to-many. We expect to have for every product, dealership, and time record many fact records. This translates to many sales for every product for every dealership within a particular period of time. True, new records will be added to each of the dimension tables, but these tables will grow slowly in comparison to the fact table.

The denormalization of the dimension tables also greatly simplifies queries against the database. In the star schema there is a join between each of the dimension tables and the fact table. A query searching for the sales of large sport utility vehicles for dealerships in the southwest would simply join three tables. Now imagine if the tables were normalized. We could easily double the number of dimension tables to create a query against five tables or more. In a fully normalized database, it would not be surprising to experience an exponential growth in the number of tables as the number of

dimensions increases. Denormalization, however, simplifies the queries and ultimately enhances overall performance.

The star schema takes significantly less space than a multidimensional database with the same dimensions. Remember, the multidimensional database will create a cell regardless of use. The star schema will create a fact record only when there are facts to record. If a dealership does not sell a particular car in a particular month, no fact record is created. Even those multidimensional databases that provide for sparcity require some knowledge on the part of the architect to predict where sparcity will occur. None of this is required with the star schema.

4.3.1 Working with the Star Schema

Although the data warehouse architect is well aware that the structure of the database is a star schema, this should be invisible to the business strategist. In a well-designed system, the strategist will have a multidimensional view of data contained within a relational database. All of the advantages of working in a multidimensional environment are available to the strategist using the star schema.

Let's begin by looking at rotation. In the multidimensional world the *Y*-axis becomes the rows of our reports and the *X*-axis becomes the columns. The *Z*-axis is what we refer to as a *page item*, in which each page of the report, or display, is a separate value of the *Z*-axis. The strategist specifies the orientation of the data, and the presentation layer displays the data accordingly. How the data is retrieved from the database is independent of the orientation of the display. Each point in the analysis space is the equivalent of a cell in a data cube, which is the equivalent of a record in the fact table. Each record contains the coordinates of its respective position within the analysis space. It is a simple matter for the presentation layer to change the orientation of the data to suit the specifications of the strategist.

The second feature of multidimensional data we discussed was the ability to slice data. In this discussion we noted how slicing data is merely the constriction of the analysis space by the specification of a subset of dimensional values. In our auto-sales example we might specify one prod-

uct or one dealership. In the star schema we would do almost the same thing. We would simply select the fact-table records based on the keys of a subset of the dimensions. We would select the record from the fact table whose product key matched the key to the product table, regardless of the values of the other keys. We can apply the same technique to the other dimensions or multiple dimensions at one time, just as we did with in the multidimensional environment.

In the next chapter we will examine dimension tables in more detail. As part of this discussion we will look at slicing data within the star schema. The next chapter will also discuss how hierarchies are implemented within the star schema.

4.3.2 Big Brother and the Building Company's Implementation Model

Now that we understand the star schema, we are ready to construct the implementation model for BBBC. This model is presented in Figure 4–15. When we last visited our case study we defined the analysis space as being dimensioned by the broker, property, customer, and time dimensions. The implementation model creates a separate dimension table for each of these objects.

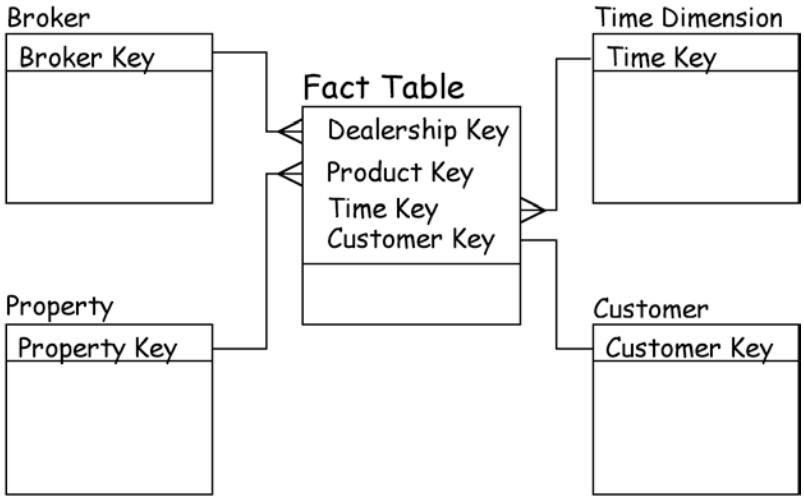

Figure 4–15 BBBC star schema.

The fact table will initially contain revenue information for the brokerage business unit only. The facts in the fact table represent either the sale or lease of property. We could have chosen to create two fact tables, one for the sale of property and another for the lease. In a sense they represent two distinct ways in which the organization has interacted with the client. This would give us problems in future iterations of the data warehouse. As we added other business units such as Appraisal and Realty Advisors, we would have to create additional fact tables to reflect the different ways in which these units interacted with the user. By combining them into one table we haven't lost anything. It still provides the business strategist with the ability to view lease and sale information individually. Creating multiple fact table would only slow down our queries through multiple joins when examining the overall organization.

In the case study we have not yet discussed the detailed attributes of the dimension tables or the facts in the fact table. We will discuss each of the topics in more detail in subsequent chapters. In Chapter 5 we will present the dimension table as well as discuss the inclusion of object attributes. In Chapter 6 we will present the issues that relate to facts within the fact table.

4.4 Summary

When developing a strategy, the business strategist seeks to understand the behaviors of objects that relate to his or her organization. We have defined behaviors as predictable responses to a set of stimuli. Multidimensional analysis allows the strategist to study the relationship between several objects at one time. He or she is able to stand back and look at the entire picture to see the ebb and flow of activity relating to the organization.

Multidimensional analysis begins with dimensions. These are the objects that relate to our organization. We create a dimension for every object that is to be part of our study. We can group related objects, dimensions, into a single coordinate system. While there is no limit on the number of dimensions that can be in one coordinate system, working beyond six or seven begins to get rather cumbersome. The coordinate system defines a

space; we describe this as binding the space. We call the space defined by the objects relating to our organization our analysis space. All points in a space exist within the dimensions that comprise the coordinate-system binding space. We identify points relative to each of these dimensions.

While multidimensional analysis is important to the business strategist, many traditional information systems are not well adapted to such uses. Relational databases work in only one dimension, and spreadsheets are inflexible in the manipulation of data. Multidimensional databases allow the strategist to create structures that fit his or her view of the analysis space. Objects are translated into dimensions forming a data cube or hypercube. The cells of these *cubes* contain the characteristics of the behavior between the objects that form the dimensions. Multidimensional databases, however, have a size limitation that makes them inappropriate for very large data warehouses.

A star schema allows the data warehouse architect to create a multidimensional space within a relational database. Dimensions are implemented as tables with wide records containing complete descriptions of the objects that make the dimensions. In the middle of these dimension tables is the fact table. Records of the fact table contain the facts of the relationship. This is the data one would expect to find in the cells of the data cube. The fact table's primary key is a composite of the keys of the surrounding dimension tables. In this way the fact table *glues* the single-dimensioned dimension tables together into an analysis space.

4.5 Glossary

Analysis Space Space defined by the dimensions of an analysis.

Coordinate System The set of related dimensions emanating from a point of origin.

Data Cube Three-dimensional object composed of columns and rows of cells. The edges of the cube are the dimensions of the analysis space.

Dimension The axes of a coordinate system that bind the space defined by that coordinate system. In an analysis space the dimensions are the objects of the analysis.

Dicing A constriction of the analysis space across multiple dimensions. Dicing selects a subset of the data to form another cube.

Drill-Down Presentation of the data at a deeper level of detail.

Hypercube An object of four dimensions or more composed of columns and rows of cells. The edges of the cube are the dimensions of the analysis space.

Multidimensional A space or database that exists in more than one dimension.

Point of Origin The point at which all axes of a coordinate system meet. The value of all dimensions in the coordinate system at this point is 0.

Roll-Up The presentation of data at a higher level of detail.

Slicing A constriction of the analysis space to include only a subset of the data. Slicing can occur along any of the dimensions.

Surrogate Keys Keys generated by the data warehouse, used in place of keys from the operational environment. Also known as synthetic keys.

Synthetic Keys Keys generated by the data warehouse. Also known as surrogate keys.