Part

I

Fundamentals

COPYRIGHTED MATERIAL

In This Part

Chapter 1: Analytic Databases and Dimensional Design
Chapter 2: Data Warehouse Architectures
Chapter 3: Stars and Cubes

CHAPTER 1

Analytic Databases and Dimensional Design

This book describes a set of powerful and effective techniques for the design of analytic databases. These techniques unleash business analytics in a very simple way: *they model the measurement of business processes*.

The dimensional model of a business process is made up of two components: measurements and their context. Known as facts and dimensions, these components are organized into a database design that facilitates a wide variety of analytic usage. Implemented in a relational database, the dimensional model is called a star schema. Implemented in a multidimensional database, it is known as a cube. If any part of your data warehouse includes a star schema or a cube, it leverages dimensional design.

This chapter introduces the basic concepts of dimensional design, which arose in response to the unique requirements of analytic systems. The concept of measurement is defined in terms of facts and dimensions, and translated into a database design, or star schema. This chapter shows how basic interaction with a star schema can support a wide variety of measurement scenarios, and defines two simple guiding principles for the development of dimensional solutions.

Dimensional Design

Information systems fall into two major categories: those that support the execution of business processes and those that support the analysis of business

processes. The principles of dimensional design have evolved as a direct response to the unique requirements of analytic systems. The core of every dimensional model is a set of business metrics that captures how a process is evaluated, and a description of the context of every measurement.

Purpose

Analytic systems and operational systems serve fundamentally different purposes. An operational system supports the *execution* of a business process, while an analytic system supports the *evaluation* of the process. Their distinct purposes are reflected in contrasting usage profiles, which in turn suggest that different principles will guide their design.

Operational Systems

An operational system directly supports the execution of a business process. By capturing details about significant events or transactions, it constructs a record of activities. A sales system, for example, captures information about orders, shipments, and returns; a human resources system captures information about the hiring and promotion of employees; an accounting system captures information about the management of the financial assets and liabilities of the business. The activities recorded by these systems are sometimes known as *transactions*. The systems themselves are sometimes called *online transaction processing (OLTP) systems*, or *transaction systems* for short.

To facilitate the execution of a business process, operational systems must enable several types of database interaction, including inserts, updates, and deletes. The focus of these interactions is almost always atomic: a specific order, a shipment, a refund. These interactions will be highly predictable in nature. For example, an order entry system must provide for the management of lists of products, customers, and salespeople; the entering of orders; the printing of order summaries, invoices, and packing lists; and the tracking order status.

Because it is focused on process execution, the operational system is likely to update data as things change, and purge or archive data once its operational usefulness has ended. When a customer moves, for example, his or her old address is no longer useful for shipping products or sending invoices, so it is simply overwritten.

Implemented in a relational database, the optimal schema design for an operational system is widely accepted to be one that is in *third normal form*. The design may be depicted as an *entity/relationship model*, or E/R model. Coupled with appropriate database technology, this design supports high-performance inserting, updating, and deleting of atomic transactions in a consistent and predictable manner. Developers refer to the characteristics of transaction processing as the ACID properties—atomic, consistent, isolated, and durable.

Analytic Systems

While the focus of the operational system is the execution of a business process, the analytic system supports the *evaluation* of the process. How are orders trending this month versus last? Where does this put us in comparison to our sales goals for the quarter? Is a particular marketing promotion having an impact on sales? Who are our best customers? These questions deal with the measurement of the overall orders process, rather than asking about individual orders.

Interaction with an analytic system takes place exclusively through queries that *retrieve* data about business processes; information is not created or modified. These queries can involve large numbers of transactions, in contrast to the operational system's typical focus on individual transactions. Specific questions asked are less predictable, and more likely to change over time. Historic data will remain important to the analytic system long after its operational use has passed. The differences between operational systems and analytic systems are highlighted in Figure 1-1.

	Operational System	**Analytic System**
Purpose	Execution of a business process	Measurement of a business process
Primary Interaction Style	Insert, Update, Query, Delete	Query
Scope of Interaction	Individual transaction	Aggregated transactions
Query patterns	Predictable and stable	Unpredictable and changing
Temporal Focus	Current	Current and historic
Design Optimization	Update concurrency	High performance query
Design principle	Entity relationship (E/R) design in third normal form (3NF)	Dimensional design (Star Schema or Cube)
Also known as	Transaction System On Line Transaction Processing (OLTP) System Source system	Data Warehouse System Data Mart

Figure 1-1: Operational systems vs. analytic systems

The principles of dimensional modeling address the unique requirements of analytic systems. A dimensional design is optimized for queries that may

access large volumes of transactions, not just individual transactions. It is not burdened with supporting concurrent, high-performance updates. It supports the maintenance of historic data, even as the operational systems change or delete information.

Measurement and Context

The founding principle of dimensional design is disarmingly simple. Dimensional design supports analysis of a business process *by modeling how it is measured*.

Measurement is easy to discern, whether by listening to people talk or reading a report or chart. Consider the following business questions:

- ''What are gross margins by product category for January?''
- ''What is the average account balance by education level?''
- ''How many sick days were taken by marketing employees last year?''
- ''What are the outstanding payables by vendor?''
- ''What is the return rate by supplier?''

Each of these questions centers on a business process: sales, account management, attendance, payables, return processing. These process-centric questions do not focus on individual activities or transactions. To answer them, it is necessary to look at a group of transactions.

Most importantly, each of these questions reveals something about how its respective business process is *measured*. The study of sales involves the measurement of *gross margin*. Financial institutions measure *account balance*. In human resources, they measure *number of absences*. The finance department measures *payables*. Purchasing managers watch the *return quantities*.

Without some kind of context, a measurement is meaningless. If you are told ''sales were $10,000,'' there is not much you can do with this information. Is that sales of a single product, or many products? Does it represent a single transaction, or the company's total sales from conception to date? Without some context, the measurement is useless.

Like the measurements themselves, context is revealed in business questions or reports. In the preceding questions, for example, gross margin is viewed in the context of *product categories* and *time* (the month of January). Sick days are viewed in the context of a *department* (marketing) and *time* (last year). Payables are viewed in the context of their *status* (outstanding) and *vendor*.

These two simple concepts, measurement and context, are the foundation of dimensional design. Every dimensional solution describes a process by capturing what is measured and the context in which the measurements are evaluated.

Facts and Dimensions

In a dimensional design, measurements are called *facts*, and context descriptors are called *dimensions*. Every dimensional design sorts information requirements into these categories. They may be identified within statements or questions, or found within report specifications. Sorting them into categories for facts and dimensions is easy, once you know what to look for.

Facts and Dimensions in Speech

In a spoken or written statement, the word ''by'' is almost always followed by a dimension. Consider the question ''What are order dollars *by* product category for January?'' Clearly, the person asking this question wants a separate measurement for each product category, as indicated by the words ''by product category.'' Product category is a dimension.

Similarly, the word ''for'' is also a good indicator of the presence of a dimension. In this case, the next word is likely to be an instance value of a dimension. For example, ''What are order dollars by product category *for January*?'' Here, ''for'' is followed by ''January.'' January is a value or instance of a dimension. The dimension is not named, but we can infer that it is month.

Facts tend to be numeric in value, and people want to see them at various levels of detail. You can identify facts by looking for things that people want to ''roll up'' or ''break out.'' Again, look at the question ''What are order dollars by product category for January?'' Here, order dollars will presumably be numeric, and they can be rolled up by categories.

Not everything that is numeric is a fact. Sometimes, a numeric data element is really a dimension. The key is to determine how it is being used. Is it something that can be specified at varying levels of detail? If so, it is a fact. Is it providing context? If so, it is a dimension. ''Show me margin dollars by order number,'' contains two numeric elements. Margin dollars can be specified at various levels of detail, and the speaker is asking that it be rolled up by order number. Clearly, ''margin dollars'' is a fact. What about order number? It too is numeric, but the speaker is not asking that order numbers be added together. Instead, order number is being used to specify context for margin dollars. Order number is a dimension.

Other examples of numeric data elements behaving as dimensions include size, ages, phone numbers, document numbers, and unit amounts such as unit cost or unit price. The clue that these elements are dimensions lies in their use. The numbers themselves are not rolled up or down according to the question. Instead, they specify the context by which something else may be rolled up or down.

Facts and Dimensions in Reports

Facts and dimensions can also be recognized by considering how they would be used if they were stored in a database and shown on a report. Dimensions serve as ''filters,'' or ''query predicates.'' The report page shown in Figure 1-2, for example, is filtering detailed data for the western region and January 2009. These filters imply several dimensions: region, month, and year.

Order Report
Western Region
January 2009
(cont'd)

Category	Product	SKU	Quantity Ordered	Cost	Order Dollars
Packing Matter	5x7 bubble mailer	011-4822	2,000	$ 950.53	$ 980.00
	8x10 bubble mailer	011-4899	3,450	$ 1,321.84	$ 1,380.00
	9x12 bubble mailer	011-5744	3,800	$ 1,254.72	$ 1,330.00
	Packing tape	011-1729	400	$ 352.82	$ 356.00
	Box Type A	021-0011	2,000	$ 2,017.46	$ 2,080.00
	Box Type B	021-0012	1,200	$ 1,166.51	$ 1,212.00
	Box Type C	021-0017	2,200	$ 2,524.93	$ 2,684.00
	All Packing Matter			$ 9,588.81	$ 10,022.00
Pens	Gel Pen Blue	017-1999	500	$ 116.39	$ 120.00
	Elegant Pen	017-2444	299	$ 600.88	$ 624.91
	Gel Pen Black	017-3001	500	$ 128.46	$ 130.00
	All Pens			*$ 845.73*	*$ 874.91*
Grand Total				***$ 207,229.42***	***$ 214,896.91***

Page 10 of 10

Figure 1-2: Sample page from an orders report

In a report, dimensions also serve to specify ''groupings,'' ''break levels'' or to identify levels of ''subtotals.'' This can be seen in the orders report, where *category* is used to break up the report into sections; individual categories are also used to label rows containing subtotals. These features are clues that *category* is a dimension.

Dimensions can also be identified by their use in controlling the aggregation of measurements. In the orders report, several measurements are specified for each product name and SKU. (SKU is a retail term that identifies an individual product, or ''stock keeping unit.'') Product and SKU are dimensions. Notice, too, that dimension values may be used to sort data.

Facts can also be spotted based on their use in queries or reports. Elements that are aggregated, summarized, or subtotaled are facts. Clearly, the report in Figure 1-2 contains the facts cost and order dollars. These numbers are being specified at various levels, including products, categories, and a grand total. Look again, and you may notice another fact: quantity ordered. Although it is not being subtotaled, this number is presumably being aggregated. In this case, it looks like quantity ordered by product is being aggregated from individual orders.

The style of the report in Figure 1-2 is sometimes referred to as tabular. Other reports may cross reference facts and dimensions in matrix format, or even display facts graphically. The values being plotted are facts; axis values, bar labels, and qualifications are dimensions.

In some cases, it can be hard to tell whether a data element is a fact or a dimension. Sometimes, a single data element can be used in both ways. Chapter 6, ''More On Dimension Tables,'' discusses behavioral attributes, which are dimensions that have been derived from facts. Later, Chapter 11, ''Transactions, Snapshots, and Accumulating Snapshots,'' looks at techniques used to construct facts out of dimension values. It is also possible to find a process where there does not appear to be a clear measurement. Chapter 12, ''Factless Fact Tables,'' describes how the factless fact table can be used to deal with this kind of situation. You also may find that some important business metrics are not so easily ''rolled up.'' Chapter 3, ''Stars and Cubes,'' considers nonadditive facts, while Chapter 11 discusses the curious case of semi-additive facts.

Grouping Dimensions and Facts

A dimensional design organizes facts and dimensions for storage in a database. It is common for a set of dimensions to share relationships to one another, independent of facts. These are grouped together in a single table to reflect their natural clustering. Similarly, facts that are available at the same level of detail are grouped together.

For example, suppose that a study of the orders process reveals the facts and dimensions shown in Figure 1-3. In this example, products are being sold by a distributor to retail stores, so the customers are companies. Each of the facts listed in the column on the left may be rolled up according to the dimension values in the right hand column.

Notice that there are natural affinities among the dimension values. Some of them clearly belong together. For example product and product description both seem to be associated with SKU. Salesperson name is determined by salesperson ID. Other dimensions are more distantly associated. Products, for example, have brands. Salespeople work in regions. Figure 1-4 shows how the facts and dimensions for the orders process might be grouped together.

Facts	Dimensions
Order Dollars	Product
Cost Dollars	Product Description
Quantity Ordered	SKU
	Brand Code
	Brand
	Brand Manager
	Category Code
	Category
	Order Date
	Month of Order
	Quarter of Order
	Fiscal Period of Order
	Year of Order
	Salesperson
	Salesperson ID
	Territory
	Territory Code
	Territory Manager
	Region
	Region Code
	Region VP
	Customer
	Customer ID
	Headquarters State of Customer
	Billing Address
	Billing City
	Billing State
	Billing Zip
	SIC Code of Customer
	Industry Name of Customer

Figure 1-3: Facts and dimensions for the orders process

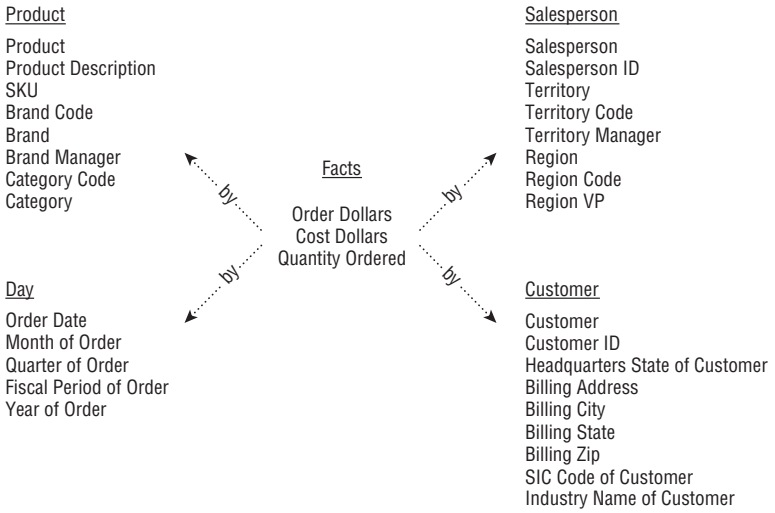


Figure 1-4: Sorting out facts and dimensions for the orders process

The diagram in Figure 1-4 depicts measurement of the orders process. It identifies the important measurements for the process and shows the dimensional context in which those measurements are evaluated. The facts and dimensions in this diagram can be combined in a number of ways, answering a wide array of business questions. Take any fact (or facts), add the word ''by,'' and then any combination of dimensions. The diagram can be easily understood by a businessperson. ''Things I measure'' go in the middle; ''the ways I look at them'' go in the surrounding boxes, loosely grouped or categorized.

The dimensional framework for the orders process in Figure 1-4 can easily be translated into a database design, or more specifically a *dimensional design*. A dimensional design organizes facts and dimensions for storage in a database management system. In a relational database management system (RDBMS), the design is referred to as a *star schema*. In a multidimensional database (MDB), the design is referred to as a *cube*. This chapter introduces the star schema; we will get back to cubes in Chapter 3.

The Star Schema

A dimensional design for a relational database is called a star schema. Related dimensions are grouped as columns in dimension tables, and the facts are stored as columns in a fact table. The star schema gets its name from its appearance; when drawn with the fact table in the center, it looks like a star or asterisk. Figure 1-5 shows a simple star schema, based on the facts and dimensions for the orders process.

Dimension Tables

In a star schema, a dimension table contains columns representing dimensions. As discussed, these columns will provide context for facts. Sometimes, ''dimension table'' is simply shortened to ''dimension.'' This may give rise to confusion, since that word can also describe the individual columns. For example, when describing the schema shown in Figure 1-5, one might refer to the columns product, product_description, and SKU as dimensions, but it is also common practice to refer to the table itself as the product dimension.

Generally, one can tell from the context whether the word ''dimension'' refers to a table or column. When it would not otherwise be clear, this book will use the terms *dimension attribute* and *dimension table* to make the distinction.

The dimension tables serve to provide the rich context needed for the study of facts. In queries and reports, the dimensions will be used to specify

how facts will be rolled up—their level of aggregation. Dimension values may be used to filter reports. They will be used to provide context for each measurement, usually in the form of textual labels that precede facts on each row of a report. They may also be used to drive master-detail relationships, subtotaling, cross-tabulation, or sorts.

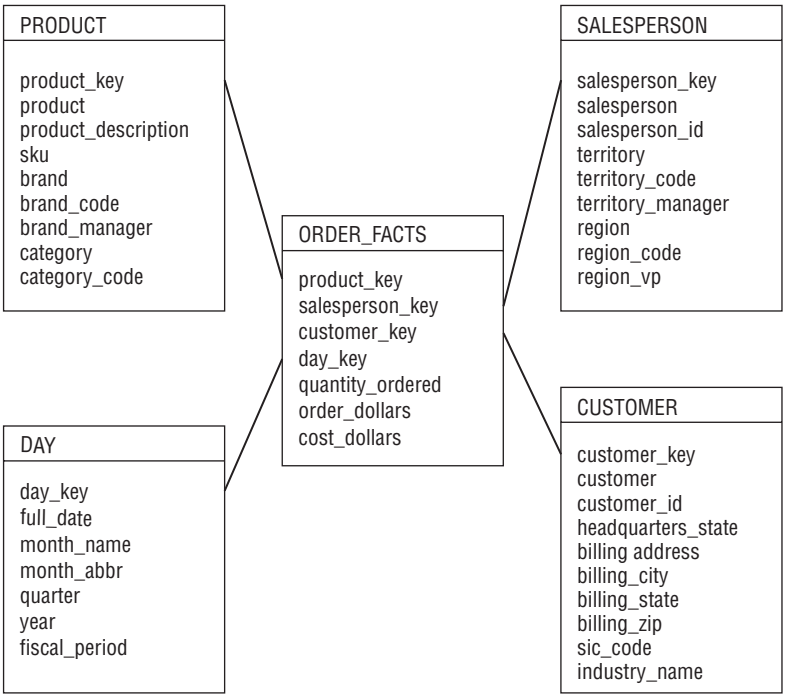


Figure 1-5: A simple star schema for the orders process

Readers with experience designing operational systems will notice that the dimension tables are not in third normal form. At first, many developers find this disconcerting. For example, there may be a strong desire to move brand, brand_code, and brand_manager into a separate table called brand, Remember, though, that a dimensional model serves a different purpose from an E/R model. It is not necessary to isolate repeating values in an environment that does not support transaction processing. While usually avoided, designers do occasionally perform additional normalization within dimensions. In such cases, the schema is referred to as a *snowflake*. The additional tables that result are sometimes called *outriggers*. Situations in which snowflaking may be useful are discussed in Chapter 7, ''Hierarchies and Snowflakes.''

Because the dimensions in a dimension table are often loosely related, it can sometimes be difficult to decide how to group things. For example, should

sales region be part of the salesperson dimension, or does it belong as a separate geographical dimension? Issues like this are tackled in Chapter 6.

Keys and History

In a star schema, each dimension table is given a surrogate key. This column is a unique identifier, created exclusively for the data warehouse. Surrogate keys are assigned and maintained as part of the process that loads the star schema. The surrogate key has no intrinsic meaning; it is typically an integer. Surrogate keys are sometimes referred to as *warehouse keys*. The surrogate key is the primary key of the dimension table.

In this book, surrogate keys will be easily identifiable by the suffix ''_key'' in the column name. In Figure 1-5, for example, the surrogate key for the customer dimension is called *customer_key*, the surrogate key for the salesperson dimension is called *salesperson_key*, and so forth. Illustrations in this book will always list the surrogate key for a dimension table as its first attribute.

Dimension tables also contain key columns that uniquely identify something in an operational system. Examples in Figure 1-5 include customer_id, sku, and salesperson_id. In the operational systems, these columns identify specific customers, products, and salespeople, respectively. These key columns are referred to as *natural keys*.

The separation of surrogate keys and natural keys allows the data warehouse to track changes, even if the originating operational system does not. For example, suppose that customer ABC Wholesalers is identified by customer_id 10711 in an operational system. If the customer changes its headquarters location, the operational system may simply overwrite the address for customer_id 10711, For analytic purposes, however, it may be useful to track the history of ABC Wholesalers. Since the star schema does not rely on customer_id to identify a unique row in the customer dimension, it is possible to store multiple versions of ABC Wholesalers, even though both have the same customer_id—10711. The two versions can be distinguished by different surrogate key values. While it would also be possible to support change tracking by supplementing a natural key with a sequence number, the surrogate key allows fact and dimension tables to be joined based on a single column.

The term *slowly changing dimension* refers to the manner in which a dimensional schema responds to changes in a source system. Detailed examples will be explored in Chapter 3, ''Stars and Cubes.'' Advanced techniques are explored in Chapter 8, ''More Slow Change Techniques.''

Fact Tables

At the core of a star schema is the fact table. In addition to the facts, the fact table includes surrogate keys that refer to each of the associated dimension

tables. The simple orders star in Figure 1-5, for example, includes the facts quantity_ordered, cost_dollars, and order_dollars. It also includes surrogate keys that reference products, salesperson, customers, and order dates.

Together, the foreign keys in a fact table are sometimes considered to identify a unique row in the fact table. This is certainly true in Figure 1-5, where each fact table row represents orders of a product sold by a salesperson to a customer on a given day. In other cases, however, the foreign keys in a fact table are not sufficient to identify a unique row. As we will see in Chapter 3, sometimes a fact table row can be uniquely identified by a *subset* of its foreign keys, or even by using some nonkey attributes.

Each row in the fact tables stores facts at a specific level of detail. This level of detail is known as the fact table's *grain*, a term that will be explored in Chapter 3. The information held in fact tables may be consumed at a variety of *different* levels, however, by aggregating the facts. In some data warehouse architectures, it is critical that the star schema capture information at the lowest level of detail possible. In other architectures, this is less important because a separate part of the data warehouse architecture is reserved for atomic data. These variations in data warehouse architecture are explored in Chapter 2, ''Data Warehouse Architectures.''

Using a Star Schema

There is still much to be said about the fundamentals of dimensional design, a discussion that continues in Chapter 3, but enough background has now been provided to look at an important feature of the star schema: *how it is actually used*. Understanding the basic usage pattern of the star schema allows the dimensional designer to make intelligent choices.

Querying Facts

Most queries against a star schema follow a consistent pattern. One or more facts are requested, along with the dimensional attributes that provide the desired context. The facts will be summarized in accordance with the dimensions present in the query. Dimension values are also used to limit the scope of the query, serving as the basis for filters or constraints on the data to be fetched and aggregated.

A properly configured relational database is well equipped to respond to such a query, which is issued using Structured Query Language (SQL). Suppose that someone has asked to see a report showing order dollars by product category and product name during the month of January 2009. The orders star schema from Figure 1-5 can provide this information, even though order dollars is stored at a lower level of detail. The SQL query in

Figure 1-6 produces the required results, summarizing tens of thousands of fact table rows.

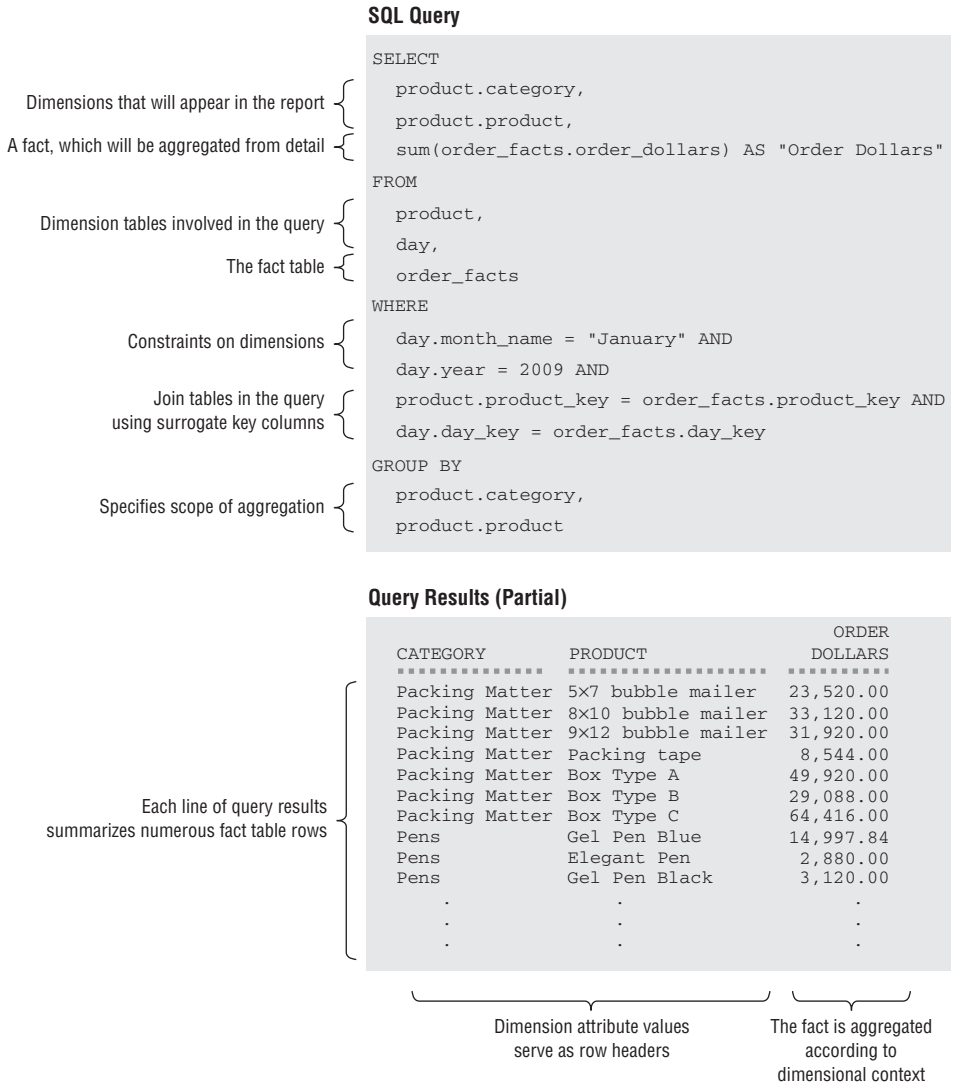


Figure 1-6: Querying the star schema

The `SELECT` clause of the query indicates the dimensions that should appear in the query results (category and product), the fact that is requested (order dollars), and the manner in which it will be aggregated (through the SQL `sum()` operation). The `FROM` clause specifies the star schema tables that are involved in the query.

The WHERE clause constrains the query based on the values of specific dimension columns (month and year), and specifies the join relationships between tables in the query. In terms of processing time, joins are among the most expensive operations the database must perform; notice that in the case of a star schema, dimension attributes are always a maximum of one join away from facts. Finally, the GROUP BY clause specifies the context to which the fact will be aggregated by the relational database.

For readers new to dimensional design, there are two key insights to take away. First, the star schema can be used in this manner with any combination of facts and dimensions. This permits the star to answer questions that may not have been posed during the design process. Although facts are stored at a specific level of detail, they can be rolled up or summarized at various levels of detail. The reporting possibilities increase dramatically as the richness of the dimension tables is increased.

Second, note that the ability to report facts is primarily limited by the level of detail at which they are stored. While it is possible to aggregate the detailed fact table rows in accordance with any set of dimensions, it is not possible to produce a *lower* level of detail. If a fact table stores daily totals, for example, it cannot be used to look at an individual order. The importance of this limitation depends in part on your data warehouse architecture, as you will see in the next chapter.

Of course, star schema queries can get much more complex than this example. Queries may build on this template in a number of ways. A very important type of report requires that we merge query results sets from more than one star. These *drill-across reports* will be discussed in Chapter 4, ''A Fact Table for Each Process.'' Some reports add subqueries to the mix, enhancing the selection of relevant facts, as discussed in Chapter 16, ''Design and Business Intelligence.'' Another type of report necessitates the performance of set operations with the results of more than one query, as discussed in Chapter 14. It is also possible that facts may be aggregated in other ways, perhaps by averaging them or simply counting them.

Browsing Dimensions

An often overlooked, but equally important, form of interaction with a dimensional schema is the *browse* query. Browsing is the act of exploring the data within a dimension. The results of browse queries appear as reference data, and may make useful reports. A browse activity may also be an exploratory precursor to a larger query against the fact table.

Like a query against a fact table, a browse query is not limited to studying information at the level of detail stored in the database. Instead, queries may browse for distinct combinations of attributes. Figure 1-7 shows some queries that browse the product dimension.

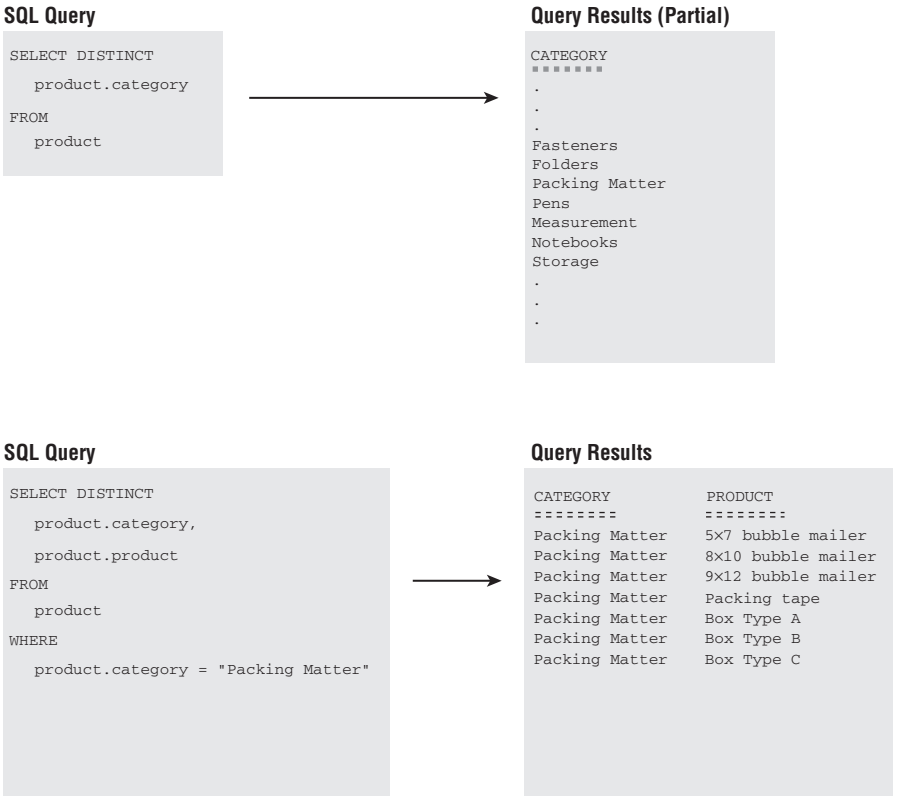


Figure 1-7: Browse queries and their results

The first browse in Figure 1-7 simply fetches a list of product categories. The second browse seeks the list of products within a specific category. Browse queries may return many attributes from within a dimension; some tools support browsing in a grid-like interface.

The browse query is important in several respects. It may serve as the basis for the selection of query predicates, or filters, for a query that involves a fact table. A browse query may also allow users to explore the relationship between dimension values. This kind of browsing may be considered when making decisions about how to group attributes into dimensions, as discussed in Chapter 6.

Guiding Principles

The remainder of this book covers a wealth of dimensional design techniques which you can use to describe any business process. Sometimes it will be useful to understand the reason some of these techniques have been developed. Other

times, it may be necessary for you to choose from some design options. Two simple guiding principles drive these decisions: accuracy and performance.

It may seem obvious, but it is important to consider the *accuracy* of any given design. The questions that will be asked of an operational system can be determined in advance, and remain consistent over time, but analytic questions always lead to new questions. They will change over time, sometimes dramatically so. Designers must pay close attention to how a dimensional schema represents facts. Is it possible that they will be aggregated in ways that do not make sense? Is there a design alternative that can prevent such a situation?

Of equal importance is the *performance* of the schema. An analytic design may offer little value over an operational design if it cannot produce timely results. Dimensional designs are very good at providing a rapid response to a wide range of unanticipated questions, There will be times, however, when a basic design may not be able to serve important business needs efficiently. The performance profile of a solution may drive the decision to provide information in more than one format, as will be seen throughout this book.

Summary

Dimensional modeling is a design approach optimized for analytic systems. A dimensional model captures how a process is measured. Data elements that represent measurements are called facts. Data elements that provide context for measurements are called dimensions. These elements are grouped into dimension tables and fact tables. Implemented in a relational database, the design is called a star schema.

The dimension tables in a star schema employ surrogate keys, enabling the analytic system to respond to changes in operational data in its own way. The granular facts in a star schema can be queried at various levels of detail, aggregated according to desired dimensional context. Exploring the details within a dimension is referred to as browsing.

This chapter has only begun to introduce the fundamentals of dimensional design. After a discussion of architectures in Chapter 2, Chapter 3 will return to the basics of dimensional design.

Further Reading

For more information on the design of operational systems, there is no finer reference than Chris Date's *An Introduction to Database Systems, Eighth Edition* (Addison Wesley, 2003). This book fully explains the principles of

normalization that are used to support transaction in a relational database management system.

A wealth of information is available on the differences between operational and analytic systems. Two good places to start are Chapter 1 of Ralph Kimball and Margy Ross's *The Data Warehouse Toolkit, Second Edition* (Wiley, 2002) and Inmon, Imhoff and Sousa's discussion in *The Corporate Information Factory, Second Edition* (Wiley, 2000).

For more information on separating facts from dimensions, you can consult any book on dimensional design. *Data Warehouse Design Solutions* (Adamson and Venerable; Wiley, 1995), and *Mastering Data Warehouse Aggregates* (Adamson; Wiley, 2006) both cover the topic in their opening chapters, as does Kimball and Ross's *The Data Warehouse Toolkit*. These books also cover the prototypical query pattern for a star schema; the browse query is discussed in *The Data Warehouse Toolkit.*