**Лекция 1**

**Business intelligence**

The data warehousing and business intelligence (DW/BI) industry certainly has matured since Ralph Kimball published the fi rst edition of *The Data Warehouse* *Toolkit* (Wiley) in 1996. Although large corporate early adopters paved the way, DW/BI has since been embraced by organizations of all sizes. The industry has built thousands of DW/BI systems. The volume of data continues to grow as warehouses are populated with increasingly atomic data and updated with greater frequency.

Over the course of our careers, we have seen databases grow from megabytes to gigabytes to terabytes to petabytes, yet the basic challenge of DW/BI systems has remained remarkably constant. Our job is to marshal an organization’s data and bring it to business users for their decision making. Collectively, you’ve delivered on this objective; business professionals everywhere are making better decisions and generating payback on their DW/BI investments.

Since the first edition of *The Data Warehouse Toolkit* was published, dimensional modeling has been broadly accepted as the dominant technique for DW/BI presentation. Practitioners and pundits alike have recognized that the presentation of data must be grounded in simplicity if it is to stand any chance of success. Simplicity is the fundamental key that allows users to easily understand databases and software to efficiently navigate databases. In many ways, dimensional modeling amounts

to holding the fort against assaults on simplicity. By consistently returning to a business-driven perspective and by refusing to compromise on the goals of user understandability and query performance, you establish a coherent design that serves the organization’s analytic needs. This dimensionally modeled framework becomes the *platform for BI*. Based on our experience and the overwhelming feedback from numerous practitioners from companies like your own, we believe that

dimensional modeling is absolutely critical to a successful DW/BI initiative.

Dimensional modeling also has emerged as the leading architecture for building integrated DW/BI systems. When you use the conformed dimensions and conformed facts of a set of dimensional models, you have a practical and predictable framework for incrementally building complex DW/BI systems that are inherently distributed.

For all that has changed in our industry, the core dimensional modeling techniques that Ralph Kimball published 17 years ago have withstood the test of time. Concepts such as conformed dimensions, slowly changing dimensions, heterogeneous products, fact tables, and the enterprise data warehouse bus matrix continue to be discussed in design workshops around the globe. The original concepts have been embellished and enhanced by new and complementary techniques.

**Data Warehousing, Business Intelligence and Dimensional Modeling**

This first chapter lays the groundwork for the following chapters. We begin by considering *data warehousing and business intelligence* (*DW/BI)* systems from a high-level perspective. You may be disappointed to learn that we don’t start with technology and tools—first and foremost, the DW/BI system must consider the needs of the business. With the business needs firmly in hand, we work backwards through the logical and then physical designs, along with decisions about technology

and tools.

We drive stakes in the ground regarding the goals of data warehousing and business intelligence in this chapter, while observing the uncanny similarities between the responsibilities of a DW/BI manager and those of a publisher.

With this big picture perspective, we explore dimensional modeling core concepts and establish fundamental vocabulary. From there, this chapter discusses the major components of the Kimball DW/BI architecture, along with a comparison of alternative architectural approaches; fortunately, there’s a role for dimensional modeling regardless of your architectural persuasion. Finally, we review common dimensional modeling myths. By the end of this chapter, you’ll have an appreciation for the need to be one-half DBA (database administrator) and one-half MBA (business analyst)

as you tackle your DW/BI project.

Before we delve into the details of dimensional modeling, it is helpful to focus on the fundamental goals of data warehousing and business intelligence. The goals can be readily developed by walking through the halls of any organization and listening to business management.

* “We collect tons of data, but we can’t access it.”
* “We need to slice and dice the data every which way.”
* “Business people need to get at the data easily.”
* “Just show me what is important.”
* “We spend entire meetings arguing about who has the right numbers rather than making decisions.”
* “We want people to use information to support more fact-based decision making.”

**Dimensional modelling introduction**

*Dimensional modeling* is widely accepted as the preferred technique for presenting analytic data because it addresses two simultaneous requirements:

■ Deliver data that’s understandable to the business users.

■ Deliver fast query performance.

Dimensional modeling is a longstanding technique for making databases simple. In case after case, for more than five decades, IT organizations, consultants, and business users have naturally gravitated to a simple dimensional structure to match the fundamental human need for simplicity. Simplicity is critical because it ensures that users can easily understand the data, as well as allows software to navigate and deliver results quickly and efficiently.

Imagine an executive who describes her business as, “We sell products in various markets and measure our performance over time.” Dimensional designers listen carefully to the emphasis on product, market, and time. Most people find it intuitive to think of such a business as a cube of data, with the edges labeled product, market, and time. Imagine slicing and dicing along each of these dimensions. Points inside the cube are where the measurements, such as sales volume or profit, for

that combination of product, market, and time are stored. The ability to visualize something as abstract as a set of data in a concrete and tangible way is the secret of understandability. If this perspective seems too simple, good! A data model that starts simple has a chance of remaining simple at the end of the design. A model that starts complicated surely will be overly complicated at the end, resulting in slow query performance and business user rejection. Albert Einstein captured the

basic philosophy driving dimensional design when he said, “Make everything as simple as possible, but not simpler.”

Although dimensional models are often instantiated in relational database management systems, they are quite different from *third normal form (3NF) models* which seek to remove data redundancies. Normalized 3NF structures divide data into many discrete entities, each of which becomes a relational table. A database of sales orders might start with a record for each order line but turn into a complex spider web diagram as a 3NF model, perhaps consisting of hundreds of normalized tables.

The industry sometimes refers to 3NF models as entity-relationship (ER) models. *Entity-relationship diagrams* (*ER diagrams or ERDs*) are drawings that communicate the relationships between tables. Both 3NF and dimensional models can be represented in ERDs because both consist of joined relational tables; the key difference between 3NF and dimensional models is the degree of normalization. Because both model types can be presented as ERDs, we refrain from referring to 3NF models as ER models; instead, we call them normalized models to minimize confusion.

Normalized 3NF structures are immensely useful in operational processing because an update or insert transaction touches the database in only one place.

Normalized models, however, are too complicated for BI queries. Users can’t understand, navigate, or remember normalized models that resemble a map of the Los Angeles freeway system. Likewise, most relational database management systems can’t efficiently query a normalized model; the complexity of users’ unpredictable queries overwhelms the database optimizers, resulting in disastrous query performance.

The use of normalized modeling in the DW/BI presentation area defeats the intuitive and high-performance retrieval of data. Fortunately, dimensional modeling addresses the problem of overly complex schemas in the presentation area.

**Star Schemas Versus OLAP Cubes**

Dimensional models implemented in relational database management systems are referred to as *star schemas* because of their resemblance to a star-like structure. Dimensional models implemented in multidimensional database environments are referred to as *online analytical processing (OLAP) cubes*. If your DW/BI environment includes either star schemas or OLAP cubes, it leverages dimensional concepts. Both stars and cubes have a common logical design with recognizable dimensions; however, the physical implementation differs.

When data is loaded into an OLAP cube, it is stored and indexed using formats and techniques that are designed for dimensional data. Performance aggregations or precalculated summary tables are often created and managed by the OLAP cube engine. Consequently, cubes deliver superior query performance because of the Data Warehousing, Business Intelligence, and Dimensional Modeling

precalculations, indexing strategies, and other optimizations. Business users can drill down or up by adding or removing attributes from their analyses with excellent performance without issuing new queries. OLAP cubes also provide more analytically robust functions that exceed those available with SQL. The downside is that you pay a load performance price for these capabilities, especially with large data sets.



**Fact Tables for Measurements**

The *fact table* in a dimensional model stores the performance measurements resulting from an organization’s business process events. You should strive to store the low-level measurement data resulting from a business process in a single dimensional model. Because measurement data is overwhelmingly the largest set of data,

it should not be replicated in multiple places for multiple organizational functions around the enterprise. Allowing business users from multiple organizations to access a single centralized repository for each set of measurement data ensures the use of consistent data throughout the enterprise.

The term *fact* represents a business measure. Imagine standing in the marketplace watching products being sold and writing down the unit quantity and dollar sales amount for each product in each sales transaction. These measurements are captured as products are scanned at the register, as illustrated in Figure 1-2.

Each row in a fact table corresponds to a measurement event. The data on each row is at a specific level of detail, referred to as the *grain*, such as one row per product Data Warehousing, Business Intelligence, and Dimensional Modeling sold on a sales transaction. One of the core tenets of dimensional modeling is that all the measurement rows in a fact table must be at the same grain. Having the discipline to create fact tables with a single level of detail ensures that measurements aren’t inappropriately double-counted.



The most useful facts are numeric and additive, such as dollar sales amount. Throughout this book we will use dollars as the standard currency to make the case study examples more tangible—you can substitute your own local currency if it isn’t dollars.

Additivity is crucial because BI applications rarely retrieve a single fact table row. Rather, they bring back hundreds, thousands, or even millions of fact rows at a time, and the most useful thing to do with so many rows is to add them up. No matter how the user slices the data in Figure 1-2, the sales units and dollars sum to a valid total. You will see that facts are sometimes semi-additive or even nonadditive.

Semi-additive facts, such as account balances, cannot be summed across the time dimension. Non-additive facts, such as unit prices, can never be added. You are forced to use counts and averages or are reduced to printing out the fact rows one at a time—an impractical exercise with a billion-row fact table.

*Dimension tables* are integral companions to a fact table. The dimension tables contain the textual context associated with a business process measurement event. They describe the “who, what, where, when, how, and why” associated with the event.

As illustrated in Figure 1-3, dimension tables often have many columns or attributes. It is not uncommon for a dimension table to have 50 to 100 attributes; although, some dimension tables naturally have only a handful of attributes.

Dimension tables tend to have fewer rows than fact tables, but can be wide with many large text columns. Each dimension is defined by a single primary key (refer to the PK notation in Figure 1-3), which serves as the basis for referential integrity with any given fact table to which it is joined.



Dimension attributes serve as the primary source of query constraints, groupings, and report labels. In a query or report request, attributes are identified as the *by* words. For example, when a user wants to see dollar sales by brand, brand must be available as a dimension attribute.

Dimension table attributes play a vital role in the DW/BI system. Because they are the source of virtually all constraints and report labels, dimension attributes are critical to making the DW/BI system usable and understandable. Attributes should consist of real words rather than cryptic abbreviations. You should strive to minimize the use of codes in dimension tables by replacing them with more verbose textual attributes. You may have already trained the business users to memorize operational codes, but going forward, minimize their reliance on miniature notes attached to their monitor for code translations. You should make standard decodes for the operational codes available as dimension attributes to provide consistent labeling on queries, reports, and BI applications. The decode values should never be buried in the reporting applications where inconsistency is inevitable.

Sometimes operational codes or identifiers have legitimate business significance to users or are required to communicate back to the operational world. In these cases, the codes should appear as explicit dimension attributes, in addition to the corresponding user-friendly textual descriptors. Operational codes sometimes have intelligence embedded in them. For example, the first two digits may identify the line of business, whereas the next two digits may identify the global region. Rather

than forcing users to interrogate or filter on substrings within the operational codes, pull out the embedded meanings and present them to users as separate dimension attributes that can easily be filtered, grouped, or reported.

In many ways, the data warehouse is only as good as the dimension attributes; the analytic power of the DW/BI environment is directly proportional to the quality and depth of the dimension attributes. The more time spent providing attributes with verbose business terminology, the better. The more time spent populating the domain values in an attribute column, the better. The more time spent ensuring the quality of the values in an attribute column, the better. Robust dimension attributes deliver robust analytic slicing-and-dicing capabilities.