Making Everything Easier!™

2nd Edition

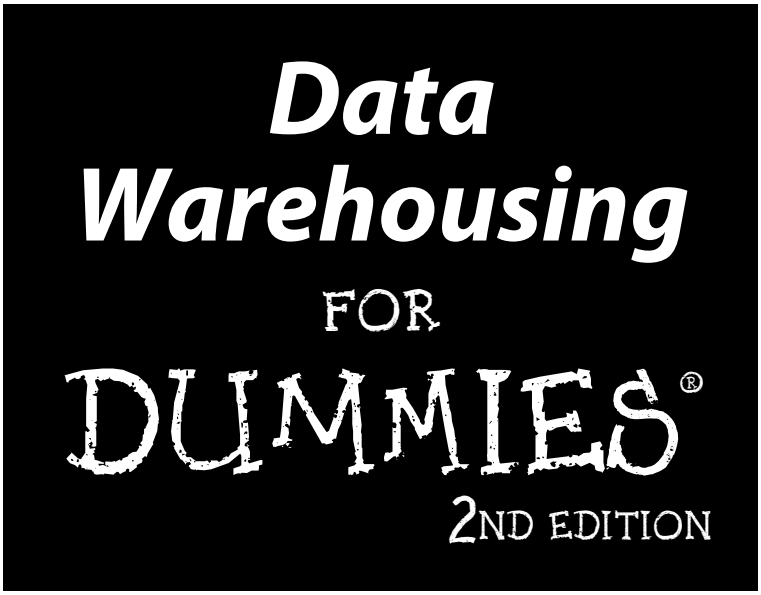
Data Warehousing

FOR DUMMIES®

Learn to:

- Analyze top-down and bottom-up data warehouse designs
- Understand the structure and technologies of data warehouses, operational data stores, and data marts
- Implement a data warehouse, step by step
- Involve end-users in the process

Thomas C. Hammergren
Alan R. Simon



**by Thomas C. Hammergren
and Alan R. Simon**

Wiley Publishing, Inc.

Data Warehousing For Dummies®, 2nd Edition
Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Control Number: 2009920908

ISBN: 978-0-470-40747-9

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

About the Author

Tom Hammergren is known worldwide as an innovator, writer, educator, speaker, and consultant in the field of information management. Tom's information management and software career spans more than 20 years and includes key roles in successful business intelligence and information management solution companies such as Cognos, Cincom, and Sybase. Tom is the founder of Balanced Insight, Inc., a leading vendor of business intelligence lifecycle management software and services that also works on innovation in semantically driven business intelligence.

While working for Sybase, Hammergren helped design and develop WarehouseStudio, a comprehensive set of tools for delivering enterprise data warehousing solutions. At Cincom, Tom helped deliver the SupraServer product line to market, one of the first fully distributed data management solutions for highly survivable network implementations. During an earlier position at Cognos, he was one of the founding members of the PowerPlay and Impromptu product teams.

Tom has published numerous articles in industry journals and is the author of two widely read books, *Data Warehousing: Building the Corporate Knowledge Base* and *Official Sybase Data Warehousing on the Internet: Accessing the Corporate Knowledge Base* (both from International Thomson Computer Press).

Dedication

This book is dedicated to my mother and father. Thank you both for the foundation and direction growing up — and, most importantly, for always supporting me in my life endeavors, no matter how crazy they have been or are. You are the best — all my love!

Author's Acknowledgments

Writing a book is much harder than it sounds and involves extended support from a multitude of people. Though my name is on the cover, many people were ultimately involved in the production of this work. As I began to think of all the people to whom I would like to express my sincere gratitude for their support and general assistance in the creation of this book, the list grew enormous.

There are those that are most responsible for making this book a reality: Kyle Looper, Acquisitions Editor; Nicole Sholly, Project Editor; and Carole Jelen McClendon of Waterside Productions, my trusted agent for more than 10 years.

The most important thank-you is to my wife, Kim, and loving children, Brent and Kristen. They created an environment in which I could successfully complete this book — an accomplishment that I share with them and one that forced all of us to sacrifice a lot.

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at http://dummies.custhelp.com. For other comments, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial

Project Editor: Nicole Sholly

Acquisitions Editor: Kyle Looper

Copy Editor: Laura K. Miller

Technical Editor: Russ Mullen

Editorial Managers: Kevin Kirschner, Jodi Jensen

Editorial Assistant: Amanda Foxworth

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Patrick Redmond

Layout and Graphics: Samantha K. Allen, Reuben W. Davis, Nikki Gately, Joyce Haughey, Melissa K. Jester, Sarah Philippart

Proofreaders: Dwight Ramsey, Nancy L. Reinhardt

Indexer: Sharon Shock

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Table of Contents

Introduction

The data warehousing revolution has been underway for over ten years within information technology (IT) departments around the world. If you're an IT professional, or you're fashionably referred to as a *knowledge worker* (someone who regularly uses computer technology in the course of your day-to-day business operations), data warehousing is for you! If you haven't heard of this phenomenon, you might be aware of the tools that access the data warehouse — business intelligence tools. *Data Warehousing For Dummies,* 2nd Edition, guides you through the overwhelming amount of hype about this subject to help you get the most from data warehousing.

If you're an IT professional (a software developer, database administrator, software development manager, or data-processing executive), this book provides you with a clear, no-hype description of data warehousing technology and methodology — what works, what doesn't work, and why.

If you regularly use computers in your job to find information and facts as a contracts analyst, researcher, district sales manager, or any one of thousands of other jobs in which data is a key asset to you and your organization, this book has in-depth information about the real business value (again, without the hype) that you can gain from data warehousing.

Why I Wrote This Book

Although data warehousing can be an incredibly powerful tool for you and others in your organization, pitfalls (a lot of them!) are scattered along your path, from thinking about data warehousing to implementing it. The path to data warehousing is similar to the yellow brick road in *The Wizard of Oz:* Even though the journey seems relatively straightforward, you have to watch out for certain obstacles along the way, such as which technology path to take when you have a choice and all kinds of things you don't expect. Although you don't have to figure out how to handle winged monkeys and apple-throwing trees, you do have to deal with products that don't work as advertised and unanticipated database performance problems.

I've been working with data warehousing since early in my career, in the late 1980s. Although the data warehousing revolution began in the early 1990s and you now can find a much broader array of technologies and tools, the principle of data warehousing isn't all that new (as mentioned in Chapter 1).

With the volume of information that companies produce internally and access externally, almost all organizations have a universal interest in data warehousing. You can't easily find an organization right now that doesn't have at least one data warehousing initiative under way, on the drawing board, or in production. Everyone wants to consume data — which leads directly to the need for a data warehouse!

This broad interest in data warehousing has, unfortunately, led to confusion about these issues:

- **Terminology:** For example, because no official definitions exist for the terms *data warehouse, data mart,* or *data mining,* product vendors declare definitions that best suit the products they sell.
- **How to successfully implement a large data warehousing system:** Should you build one large database of information and then parcel off smaller portions to different organizations, or should you build a bunch of smaller-scale databases and then integrate them later?
- **Advances in technology:** New facets of technology, such as the Internet, are having an effect on data warehousing.

This book is, in many ways, a consolidation of my down-to-earth, no-hype conversations with and presentations to clients, IT professionals, product engineers, architects, and many others in recent years about what data warehousing means to business organizations today and tomorrow.

How to Use This Book

You can read *Data Warehousing For Dummies,* 2nd Edition, in either of these ways:

- **Read each chapter in sequential order, from cover to cover.** If this book is your first real exposure to data warehousing terminology, concepts, and technology, you probably want to go with this method.
- **Read selected chapters that are of particular interest to you and in any order you want.** I wrote each chapter to stand on its own, with little dependency on any other chapter.

To give you a sense of what awaits you in *Data Warehousing For Dummies,* 2nd Edition, the following sections describe the contents of the book, which are divided into seven parts.

Part I: The Data Warehouse: Home for Your Data Assets

Part I gets down to the basics of data warehousing: concepts, terminology, roots of the discipline, and what to do with a data warehouse after you build it.

Chapter 1 gets right to the point about a data warehouse: what you can expect to find there, how and where its content is formed, and some early cautions to help you avoid pitfalls that await you during your first data warehousing project.

Chapter 2 describes, in business-oriented terms, exactly what a data warehouse can do for you.

I describe the different types of data warehouses that you can build (small, medium, or way big!) and the circumstances in which each one is appropriate in Chapter 3.

Chapter 4 describes *data marts* (small-scale data warehouses), which have become the preferred method to deliver data to end users.

Part II: Data Warehousing Technology

In Part II, you go beyond basic concepts to find out about the technology behind data warehousing, particularly database technology.

Chapter 5 talks about relational databases (if you're an IT professional, you're probably familiar with them) and how you can use these products for data warehousing. Specialized databases, such as multidimensional and column-wise (or vertical) databases, as well as other types of databases used for data warehousing, are described in Chapter 6. In this chapter, you can figure out which type of database is a viable option for your data warehousing project.

You can read about data warehousing *middleware* — software products and tools used to extract or access data from source applications and do all the necessary functions to move that data into a data warehouse — in Chapter 7, along with the issues you have to watch out for in this area.

Part III: Business Intelligence and Data Warehousing

Part III discusses the concept of *business intelligence* — the different categories of processing that you can perform on the contents of a data warehouse. From "tell me what happened" processing to "tell me what might happen," it's all here!

See Chapter 8 for an overview of business intelligence and what it means to data warehousing.

Chapters 9 through 12 each describe, in detail, one major area of business intelligence (querying and reporting, analytical processing, data mining, and dashboard and scorecards, respectively). These chapters present you with ready-to-use advice about products in each of these areas.

Part IV: Data Warehousing Projects: How to Do Them Right

Knowing about data warehousing is one thing; being able to implement a data warehouse successfully is another. Part IV discusses project methodology, management techniques, the analysis of data sources, and how to work with users.

Chapter 13 describes data warehouse development (methodology) and the similarities to and differences from the methodologies you use for other types of applications.

Find out in Chapter 14 the right way to manage a data warehouse project to maximize your chances for success.

Chapters 15 through 18 each discuss an important part of a data warehouse project (compiling requirements, analyzing data sources, delivering the end solution, and working with users, respectively) and give you a lot of tips and tricks to use in each of these critical areas.

Part V: Data Warehousing: The Big Picture

This part of the book discusses the big picture: data warehousing in the context of all the other organizations and people in your IT organization (and even outside consultants) and your other information systems.

Find out in Chapter 19 how to establish an information value chain — from acquisition to internal data to the integration with external data (information about competing companies' sales of products, for example). You can also read about how to use that information in your data warehouse.

To understand how a data warehouse fits into your overall computing environment with the rest of your applications and information systems, see Chapter 20.

For an executive boardroom view of data warehousing, check out Chapter 21. Is this discipline as high a priority to the corporate bigwigs as you might imagine, considering its popularity?

For advice about what to do if you have systems already in place that are sort of (but not really) like a data warehouse, and which you use for simple querying and reporting, read Chapter 22. To replace those systems or upgrade them to a data warehouse — that is the question.

Chapter 23 describes how to deal with data warehousing product vendors and the best ways to acquire information at the numerous data warehousing trade shows.

You probably have to deal with data warehousing consultants (or maybe you are one). Chapter 24 fills you in on the tricks of the trade.

Part VI: Data Warehousing in the Not-Too-Distant Future

Every area of technology is constantly changing, and data warehousing is no exception. Because data warehousing is on the brink of a new generation of technologies, the chapters in this part of the book detail some of the most significant trends.

Data warehouses typically include only a few different types of data: numbers, dates, and character-based information (such as names, addresses, product descriptions, and codes). Chapter 25 fills you in on the next wave of data warehousing, in which unstructured data ripe with multimedia content (pictures, images, video, audio, and documents) are included as part of a data warehouse.

Chapter 26 uncovers the concepts around semantics. Semantics have begun to appear in Internet applications to enable programs and applications to surf the Web like humans do, and it's just a matter of time before this same technology invades the data warehousing and business intelligence environment.

Chapter 27 investigates collaborative technologies and the profound effect they'll have on making information ubiquitous and easily accessible in business.

Part VII: The Part of Tens

Last, but certainly not least, this part is the *For Dummies* institution: The Part of Tens. This part of the book has seven chapters chock-full of data warehousing hints and advice.

Icons Used in This Book

This icon denotes tips and tricks of the trade that make your projects go more smoothly and otherwise ease your foray into data warehousing.

Beware! This icon points out data warehousing traps, hype, and other potentially unpleasant experiences.

Data warehousing is all about computer technology. When you see this icon, the accompanying explanation digs into the underlying technology and processes, in case you want to get behind the scenes, under the hood, or beneath the covers.

The world is on the brink of a new generation of data warehousing! This icon tells you about a major trend in technology (or a way of implementing data warehousing) that you might find important soon.

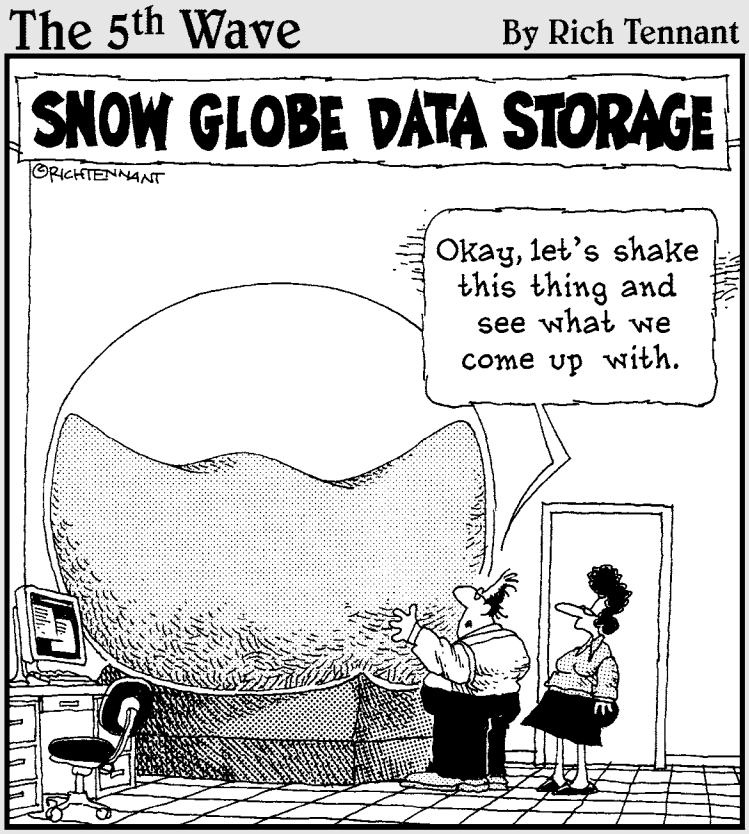
Some things about data warehousing are just so darned important that they bear repeating. This icon lets you know that I'm repeating something on purpose, not because I was experiencing déjà vu.

About the Product References in This Book

(Consider this icon a test run.) In Parts II and III, I mention a number of products and list the Web sites where you can find information about them. I paraphrase the brief product descriptions from the respective vendors' Web sites, and those descriptions were up-to-date at the time this book was written. I've mentioned the products in those chapters simply as examples of products, rather than as recommendations. (How's that for a disclaimer?)

Part I

The Data Warehouse: Home for Your Data Assets



In this part . . .

This part of the book explains, in absolutely no-hype terms, the basics of data warehousing: what a data warehouse is, where its contents come from and why, what you use it for after you build it, and options you have for choosing its level of complexity.

Chapter 1

What's in a Data Warehouse?

In This Chapter

- Understanding what a data warehouse is and what it does
- Looking at the history of data warehousing
- Differentiating between bigger and better
- Grasping the historical perspective of a data warehouse
- Ensuring that your data warehouse isn't a data dump

If you gather 100 computer consultants experienced in data warehousing in a room and give them this single-question written quiz, "Define a data warehouse in 20 words or fewer," at least 95 of the consultants will turn in their paper with a one- or two-sentence definition that includes the terms *subject-oriented, time-variant,* and *read-only.* The other five consultants' replies will likely focus more on business than on technology and use a phrase such as "improve corporate decision-making through more timely access to information."

Forget all that. The following section gives you a no-nonsense definition guaranteed to be free of both technical and business-school jargon. Throughout the rest of the chapter, I assist you in better understanding data warehousing from its history and overall value to your business.

The Data Warehouse: A Place for Your Data Assets

A *data warehouse* is a home for your high-value data, or *data assets,* that originates in other corporate applications, such as the one your company uses to fill customer orders for its products, or some data source external to your company, such as a public database that contains sales information gathered from all your competitors.

If your company's data warehouse were advertised as a product for sale, it might be described this way: "Contains high-quality, refined and purified information, all of which has undergone a 25-point quality check and is offered to you with a warranty to guarantee hassle-free ownership so that you can better monitor the performance of your business."

Classifying data: What is a data asset?

Okay, I promised a definition free of technical and business-school jargon — but in the preceding section, I introduced a term (data asset) that might be considered jargon. So, I'll clarify what the term data asset means.

You can classify data that's managed within an enterprise in three groupings:

- **Run-the-business data:** Produced by corporate applications, such as the one your company uses to fill customer orders for its products or the one your company uses to manage financial transactions. The raw materials for a data warehouse.
- **Integrate-the-business data:** Built to improve the quality of and synchronize two or more corporate applications, such as a master list of customers. Data leveraged to integrate applications that weren't designed to work with each other.
- **Monitor-the-business data:** Presented to end users for reporting and decision support, such as your financial dashboard. The data is cleansed to enable users to better understand progress and evaluate cause-and-effect relationships in the data.

A *data asset* is the result of taking the raw material from the run-the-business data and producing higher-quality-data end products to integrate the business and monitor the business. Your data warehouse team should have the mission of providing high-quality data assets for enterprise use.

Manufacturing data assets

Most organizations build a data warehouse for manufactured data assets in a relatively straightforward manner, following these steps:

1. The data warehousing team (usually computer analysts and programmers) selects a *focus area,* such as tracking and reporting the company's product sales activity against that of its competitors.

2. The team in charge of building the data warehouse assigns a group of business users and other key individuals within the company to play the role of subject-matter experts.

   Together, the data warehousing team and subject-matter experts compile a list of different types of information that can enable them to use the data warehouse to help track sales activity (or whatever the focus is for the project).

3. The group then goes through the list of information (data assets), item by item, and figures out where the data warehouse can obtain that particular piece of data (raw material).

   In most cases, the group can get the data from at least one internal (within the company) database or file, such as the one that the application uses to process orders over the Internet or the master database of all customers and their current addresses. In other cases, a piece of information isn't available from within the company's computer applications, but you could obtain it by purchasing it from some other company. Although a bank doesn't have the credit ratings and total outstanding debt for all its customers internally, for example, it can purchase that information from a third party — a credit bureau.

4. After completing the details of where the business can get each piece of information, the data warehousing team creates extraction programs.

   Extraction programs collect data from various internal databases and files, copy certain data to a *staging area* (a work area outside the data warehouse), cleanse the data to ensure that the data has no errors, and then copy the higher-quality data (data assets) into the data warehouse. Extraction programs are created either by hand (custom-coded) or by using specialized data warehousing products — ETL (extract, transform, and load) tools.

You can build a successful data warehouse by spending adequate time on the first two steps in the preceding list (analyzing the need for a data warehouse and how you should use it), which makes the next two steps (designing and implementing the data warehouse to make it ready to use) much easier to perform.

Interestingly, the analysis steps (determining the focus of the data warehouse and working closely with business users to figure out what information is important) are nearly identical to the steps for any other type of computer application. Most computer applications create data as a result of a transaction or set of transactions while a particular application is being used to run the business, such as filling a customer's order. The primary difference between run-the-business applications and a data warehouse is that a data warehouse relies exclusively on data obtained from other applications and sources. Figure 1-1 shows the difference between these two types of environments.

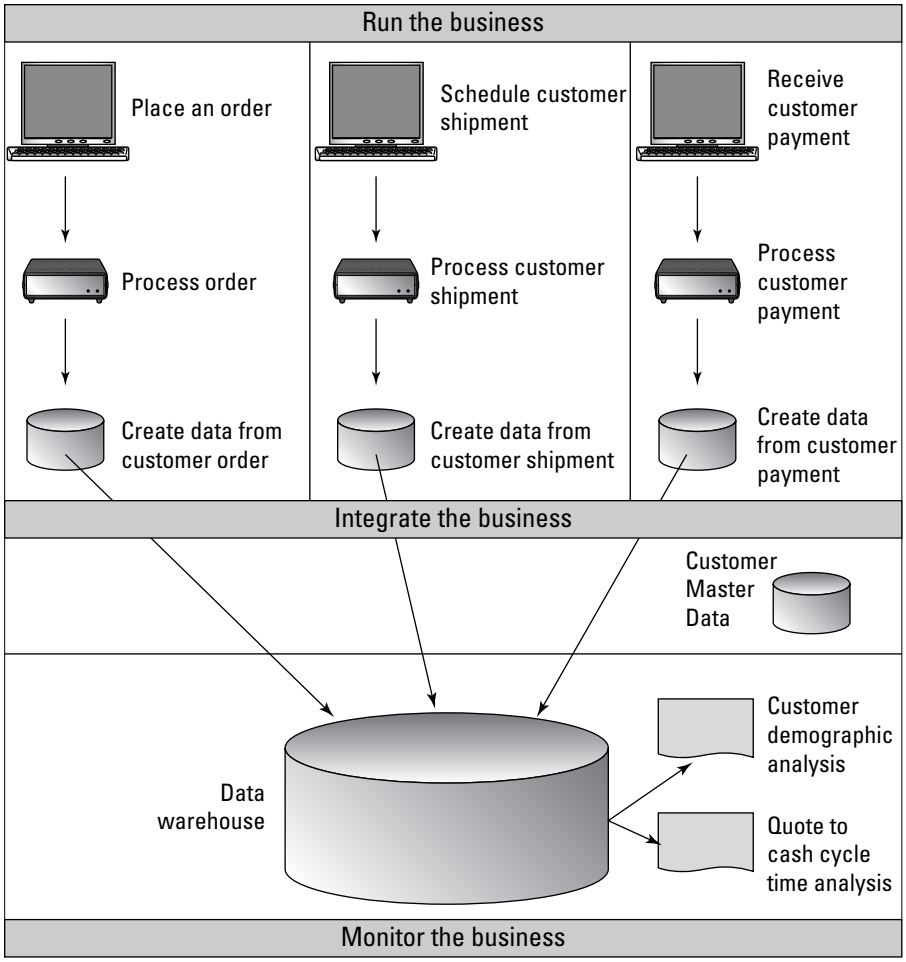


Figure 1-1: Most computer applications create data as a result of an activity or transaction; a data warehouse instead swipes data created elsewhere and converts it into information.

Data Warehousing: A Working Definition

If you cringe at the thought of defining the concept of a data warehouse and the associated project to your executive sponsors, the following sections provide a more detailed and hype-free definition and explanation that you can use to wow them.

So, what's a data warehouse? In a literal sense, it is properly described through the specific definitions of the two words that make up the term:

- **Data:** Facts and information about something
- **Warehouse:** A location or facility for storing goods and merchandise

Today's data warehousing defined

Data warehousing is the coordinated, architected, and periodic copying of data from various sources, both inside and outside the enterprise, into an environment optimized for analytical and informational processing.

The keys to this definition for computer professionals are that the data is copied *(duplicated)* in a controlled manner, and data that is copied periodically *(batch-oriented processing)*.

A broader, forward looking definition

A data warehouse system has the following characteristics:

- It provides centralization of corporate data assets.
- It's contained in a well-managed environment.
- It has consistent and repeatable processes defined for loading data from corporate applications.
- It's built on an open and scalable architecture that can handle future expansion of data.
- It provides tools that allow its users to effectively process the data into information without a high degree of technical support.

The information that you use to formulate decisions typically is based on data gathered from previous experiences — what works and what doesn't. Data warehouses capture similar data, allowing business leaders to make informed decisions based on previous business data — what's working in the business and what's doesn't work in the business. Executives are realizing that the only way to sustain and gain an advantage in today's economy is to better leverage information. The data warehouse provides the platform to implement, manage, and deliver these key data assets.

Data warehousing is therefore the process of creating an architected information-management solution to enable analytical and informational processing despite platform, application, organizational, and other barriers.

The key concept in this definition is that a data warehouse breaks down the barriers created by non-enterprise, process-focused applications and consolidates information into a single view for users to access.

A Brief History of Data Warehousing

Many people, when they first hear the basic principles of data warehousing — particularly copying data from one place to another — think (or even say), "That doesn't make any sense! Why waste time copying and moving data, and storing it in a different database? Why not just get it directly from its original location when someone needs it?"

To better understand the "why we do what we do" aspect of data warehousing, I outline its historical roots — how data warehousing became what it is today — in the following sections.

Before our time — the foundation

The evolution of data warehousing can trace its roots to work done prior to computers being widely available, including

- **The continuous marketing research conducted by Charles Coolidge Parlin (1872–1942).** Parlin is now recognized as the Father of Marketing Research. He did marketing research for the Curtis Publishing Company to gather information about customers and markets to help Curtis sell more advertising in their magazine, *The Saturday Evening Post.*
- **In 1923, Arthur C. Nielsen, Sr., established ACNielsen in the United States.** Arthur C. Nielsen was one of the founders of the modern marketing research industry. Among many innovations in consumer-focused marketing and media research, Mr. Nielsen created a unique retail-measurement technique that gave clients the first reliable, objective information about competitive performance and the impact of their marketing and sales programs on revenues and profits. Nielsen information gave practical meaning to the concept of market share and made it one of the critical measures of corporate performance.

These two events in history led to what we now know as data warehousing because each of them required high-quality data to formulate trends and enable business users to make decisions.

The 1970s — the preparation

The 1970s: Disco and leisure suits were in. And the computing world was dominated by the mainframe. Real data-processing applications, the ones run on the corporate mainframe, almost always had a complicated set of files or early-generation databases (not the table-oriented relational databases most applications use today) in which they stored data.

Although the applications did a fairly good job of performing routine data-processing functions, data created as a result of these functions (such as information about customers, the products they ordered, and how much money they spent) was locked away in the depths of the files and databases. It was almost impossible, for example, to see how retail stores in the eastern region were doing against stores in the western region, against their competitors, or even against their own performance in some earlier period. At best, you could have written up a report request and sent it to the data-processing department, where it was put on a waiting list with a couple thousand other report requests, and you might have had an answer in a few months — or not.

Some enterprising, forward-thinking people decided to take another approach to the data access problem. During the 1970s, while minicomputers were becoming popular, the thinking went like this: Rather than make requests to the data-processing department every time you need data from an application's files or databases, why not identify a few key data elements (for example, a customer's ID number, total units purchased in the most recent month, and total dollars spent) and have the data-processing folks copy this data to a tape each month during a slow period, such as over a weekend or during the midnight shift? You could then load the data from the tape into another file on the minicomputer, and the business users could use decision-support tools and *report writers* (products that allowed access to data without having to write separate programs) to get answers to their business questions and avoid continually bothering the data-processing department.

Although this approach worked (sort of) in helping to reduce the backlog of requests that the data-processing department had to deal with, the usefulness of the extracted and copied data usually didn't live up to the vision of the people who put the systems in place. Suppose that a company had three separate systems to handle customer sales: one for the eastern U.S. region, one for the western U.S. region, and one for all stores in Europe. Also, each of these three systems was independent from the others. Although data copied from the system that processed sales for the western U.S. region was helpful in analyzing western region activity for each month and maybe on a historical basis (if you retained previous batches of data), you couldn't easily answer questions about trends across the entire United States or the world without copying more data from each of the systems. People typically gave up because answering their questions just took too much time.

Additionally, commercial and hardware/software companies began to emerge with solutions to this problem. Between 1976 and 1979, the concept for a new company, Teradata, grew out of research at the California Institute of Technology (Caltech), driven from discussions with Citibank's advanced technology group. Founders worked to design a database management system for parallel processing with multiple microprocessors, specifically for decision support. Teradata was incorporated on July 13, 1979 and started in a garage in Brentwood, California. The name Teradata was chosen to symbolize the ability to manage *terabytes* (trillions of bytes) of data.

The 1980s — the birth

The 1980s: the era of yuppies. PCs, PCs, and more PCs suddenly appeared everywhere you looked — as well as more and more minicomputers (and even a few Macintoshes). Before anyone knew it, "real computer applications" were no longer only on mainframes; they were all over the place — everywhere you looked in an organization. The problem called *islands of data* was beginning to look ominous: How could an organization hope to compete if its data was scattered all over the place on different computer systems that weren't even all under the control of the centralized data-processing department? (Never mind that even when the data was all stored on mainframes, it was still isolated in different files and databases, so it was just as inaccessible.)

A group of enterprising, forward-thinking people came up with a new idea: Because data is located all over the place, why not create special software to enable people to make a request at a PC or terminal, such as "Show per-store sales in all worldwide regions, ranked in descending order by improvement over sales in the same period a year earlier"? This new type of software, called a *distributed database management system* (distributed DBMS, or DDBMS), would magically pull the requested data from databases across the organization, bring all the data back to the same place, and then consolidate it, sort it, and do whatever else was necessary to answer the user's question. (This process was supposed to happen pretty darned quickly.)

To make a long story short, although the concept of DDBMSs was a good one and early results from research were promising, the results were plain and simple: They just didn't work in the real world. Also, the islands-of-data problem still existed.

Meanwhile, Teradata began shipping commercial products to solve this problem. Wells Fargo Bank received the first Teradata test system in 1983, a parallel RDBMS (relational database management system) for decision support — the world's first. By 1984, Teradata released a production version

of their product, and in 1986, *Fortune* magazine named Teradata Product of the Year. Teradata, still in existence today, built the first data warehousing appliance — a combination of hardware and software to solve the data warehousing needs of many. Other companies began to formulate their strategies, as well.

In 1988, Barry Devlin and Paul Murphy of IBM Ireland introduced the term *business data warehouse* as a key component of the EBIS (Europe/Middle East/Africa Business Information System). *EBIS* was defined as a comprehensive architecture aimed at providing a cross-functional business information system that's easy to use and has the flexibility to change while the business environment develops, even at a rapid rate. The flexibility and cross-functional support are a result of the relational database technology on which the EBIS system is based. When describing the business data warehouse, they articulated the need to "ease access to the data and to achieve a coherent framework for such access, it is vital that all the data reside in a single logical repository."

Additionally, Ralph Kimball founded Red Brick Systems in 1986. Red Brick began to emerge as a visionary software company by discussing how to improve data access. They were promoting a specialized relational database platform which enabled large performance gains for complex ad-hoc queries. Often, they could prove performance over ten times that of other vendor databases of the time. The key to Red Brick's technology was indexes — a software answer to Teradata's hardware-based solution. These indexes where technical solutions to the key manners in which users described the data within a data warehouse — customers, products, demographics, and so on.

In short, the 1980s were the birth place of data warehousing innovation.

The 1990s — the adolescent

During the 1990s, disco made a comeback. At the beginning of the decade, some 20 years after computing went mainstream, business computer users were still no closer to being able to use the trillions of bytes of data locked away in databases all over the place to make better business decisions.

The original group of enterprising, forward-thinking people had retired (or perhaps switched to doing Web site development). Using the time-honored concept of "something old, something new" (the "something borrowed, something blue" part doesn't quite fit), a new approach to solving the islands-of-data problem surfaced. If the 1980s approach of reaching out and

accessing data directly from the files and databases didn't work, the 1990s philosophy involved going back to the 1970s method, in which data from those places was copied to another location — only doing it right this time.

And data warehousing was born.

In 1993, Bill Inmon wrote *Building the Data Warehouse* (Wiley). Many people recognize Bill as the Father of Data Warehousing. Additional publications emerged, including the 1996 book by Ralph Kimball, *The Data Warehouse Toolkit* (Wiley), which discussed general-purpose dimensional design techniques to improve the data architecture for query-centered decision support systems.

With hardware and software for data warehousing becoming common place, writings began to emerge complementing those of Inmon and Kimball. Specifically, techniques appeared that enabled those employed by Information Systems departments to better understand the trend that involved not going after data from just one place, such as a single application, but rather going after all the data you need, regardless of how many different applications and computers are used in the organization. Client/server technology can be used to put the data on servers and give users new and improved analysis tools on their PCs.

The 2000s — the adult

In the more modern era (the 2000s, the era of reality television shows and mobile communication devices), people are more connected than ever before. Information is everywhere. New languages are being created because of texting and instant messaging. Acronyms such as TTYL (talk to you later), LOL (laughing out loud), and BRB (be right back) are commonplace. And a huge number of people provide feedback to vote people off of competitions on shows such as *American Idol* — bringing new meaning to market research and understanding what will sell. For example, in 2006, viewers cast 63 million votes for the contestants in the *American Idol* finale — which exceeded the most votes obtained by a United States president (Ronald Reagan, with 54.5 million votes). So, the world is definitely now connected!

In the world of data warehousing, the amount of data continues to grow. But, while it does, the vendor community and options have begun to consolidate. The selection pool is rapidly diminishing. In 2006, Microsoft acquired ProClarity, jumping into the data warehousing market. In 2007, Oracle purchased Hyperion, SAP acquired Business Objects, and IBM merged with Cognos. The data warehousing leaders of the 1990s have been gobbled up by some of the largest providers of information system solutions in the world.

Although the vendor community has consolidated, innovation hasn't ceased. More cost-effective solutions have emerged, led by Microsoft enabling small and mid-sized businesses to implement data warehousing solutions. Additionally, less expensive alternatives are emerging from a new set of vendors, those within the open source community, including vendors such as Pentaho and Jaspersoft. Open source business intelligence tools enable corporate application vendors to embed data warehousing solutions into their software suites. And other innovations have emerged, including data warehouse appliances from vendors such as Netezza and DATAllegro (acquired by Microsoft), and performance management appliances that enable real-time performance monitoring. These innovative solutions can also provide cost savings because they're often plug-compatible to legacy data warehouse solutions.

While time ticks by, you need to have a plan in place before you begin your data warehousing process. Know the focus of what you're trying to do and the questions you're likely to be asking. Will you be asking mostly about sales activity? If so, put plans in place for regular monthly (or weekly or even daily) extractions of data about customers, the products they buy, and the amounts of money they spend. If you work at a bank and your business focus is managing the risk across loan portfolios, for example, get information from the bank's applications that handle loan payments, delinquencies, and other data you need; then, add in data from the credit bureau about your customers' respective overall financial profiles.

Is a Bigger Data Warehouse a Better Data Warehouse?

A common misconception that many data warehouse aficionados hold is that the only good data warehouse is a big data warehouse — an enormously big data warehouse. Many people even take the stance that unless they have some astronomically large number of bytes stored, it isn't truly a data warehouse. "Five hundred gigabytes? Okay, that's a *real* data warehouse; it would be a better data warehouse, however, if it had at least a terabyte (1 trillion bytes) of data. Twenty-five gigabytes? Sorry, that's a data mart, not a data warehouse." (See Chapter 4 for a discussion of the differences between data marts and data warehouses.)

The size of a data warehouse is a characteristic — almost a by-product — of a data warehouse; it's not an objective. No one should ever set out with a mission to "build a 500-gigabyte data warehouse that contains (whatever)."

To determine the size you need for your data warehouse, follow these steps:

1. **Determine the mission, or the business objectives, of the data warehouse.**

   Ask the question, "Why bother creating this warehouse?"

2. **Determine the functionality that you want the data warehouse to have.**

   Figure out what types of questions users will ask.

3. **Determine what *contents* (types of data) the data warehouse needs to support its functionality.**

   Understand what types of answers your users will seek.

4. **Determine, based on the content volume (which is based on the functionality, which in turn is based on the mission), how big you need to make your data warehouse.**

Realizing That a Data Warehouse (Usually) Has a Historical Perspective

In almost all situations, a data warehouse has a historical perspective. Some amount of time lag occurs between the time something happens in one of the data sources (a new record is added or an existing one is modified in a corporate application, for example) and the time that the event's results are available in the data warehouse.

The reason for the time lag is that you usually bulk-load data into a data warehouse in large batches. Figure 1-2 illustrates a model of bulk-loading data.

Bulk-loading is giving way to *messaging,* the process of sending a small number of updates (perhaps only one at a time) much more frequently from the data source to a target — in this case, the data warehouse. With messaging, you have a much more up-to-date picture of your data warehouse's subject areas than you do with bulk-loading because you're putting information into an operational data store (as discussed in Chapter 20), rather than into a traditional data warehouse. Additionally, the world of service-oriented architectures (SOAs) and Web 2.0 are driving the messaging and presentation of data to near real-time in some industries. The combination of the data warehouse's historic perspective with this near-real-time sourcing of information enables business leaders to monitor the situation and make decisions at the speed of the business.

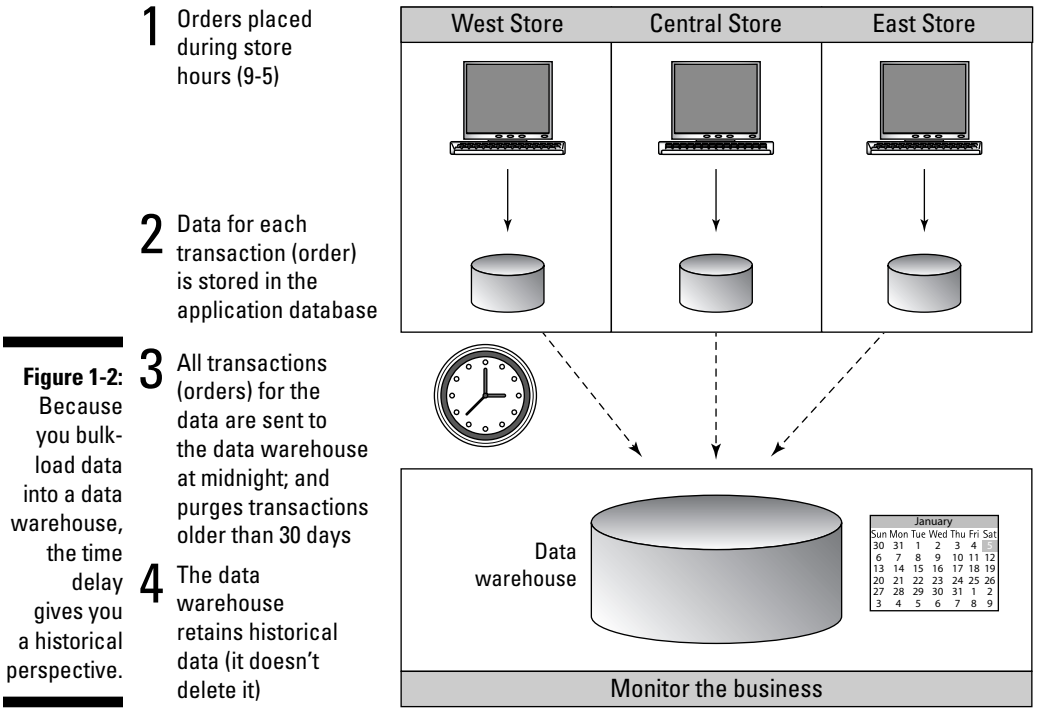


Figure 1-2: Because you bulk-load data into a data warehouse, the time delay gives you a historical perspective.

It's Data Warehouse, Not Data Dump

An often-heard argument about what should be stored in a data warehouse goes something like this: "If I have to take the trouble to pull out data from all these different applications, why not just get as much as I possibly can? If I don't get everything, or as much as possible, I won't be able to ask all the business questions I might want to."

In a commonly related story about knowledge gained from a successful data warehouse implementation, a grocery-store chain discovered an unusually high correlation of disposable baby diapers and beer sales during a two- or three-hour period early every Friday evening and found out that a significant number of people on their way home from work were buying both these items. The store then began stocking display shelves with beer and disposable diapers next to one another, and sales increased significantly.

Although I don't know whether this story is true (it certainly has been told often enough), I believe that it confuses the issue when you have to figure out what should — and should not — be in your data warehouse. The moral of

this story is usually mistaken as, "Put as much data as possible in the warehouse." In reality, the data warehouse just described was probably one that focused exclusively on sales activity. Remember that although disposable diapers and beer are dramatically different products, they're both members of the same *type* of data (retail products).

The following example emphasizes why you should be selective about what goes in your data warehouse and not just assume that you have to get every possible type of data from all the sources, just in case you want to ask your data warehouse any question.

Suppose that you're creating a data warehouse for a cruise ship company. As shown in Figure 1-3, the Tucson Desert Cruise Ship Company (its motto is "Who needs an ocean?") uses four applications that handle different tasks:

- Reservations and cancellations
- Food-and-beverage service for all cruises
- All trip itineraries and after-the-fact information about the weather, unusual events, and all onboard entertainment scheduling
- All crew assignments

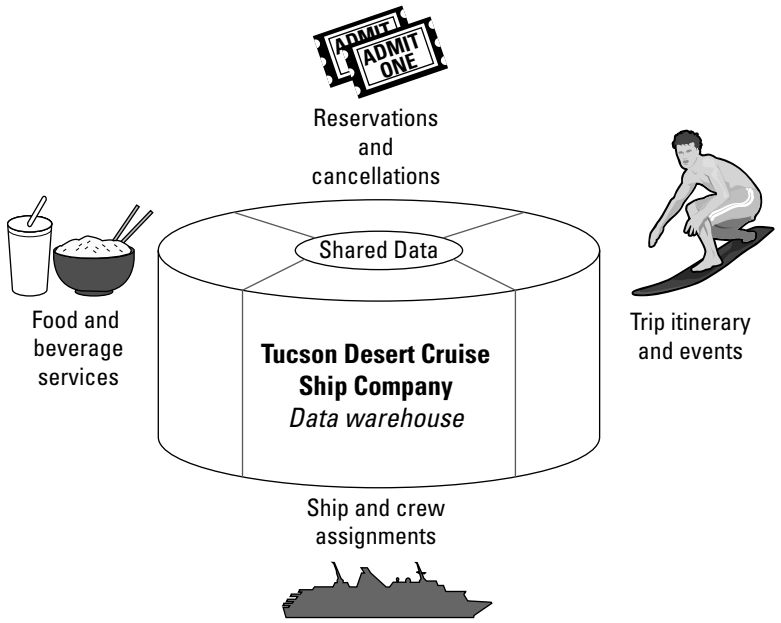


Figure 1-3: A fictional company's proposed data warehousing environment.

Figure 1-4 shows one possible environment for your data warehouse if you pursue the philosophy of "Go get everything you possibly can," or what I call the *data dump* approach.

By having the information shown in Figure 1-4 in your data warehouse, you — and every other person who uses the warehouse — can ask questions and make report requests, such as "What's the average number of room-service vegetarian meals ordered by passengers who were on their third cruise with Captain Grumby in command and in which a half-day stop was made in Grand Cayman when its temperature was between 75 and 80 degrees?"

Asking this type of question doesn't have any real business value, however. Assuming that you receive an answer to the question, what can you do with that information to have a positive business effect?

For some types of data, you can analyze, analyze, and analyze some more — and still find out little of value that could positively affect your business. Although you can put this data in your warehouse, you probably won't get much for your trouble. Other types of data, though, have significant value unavailable until placed in the data warehouse. Concentrate on the latter, and ignore the former!

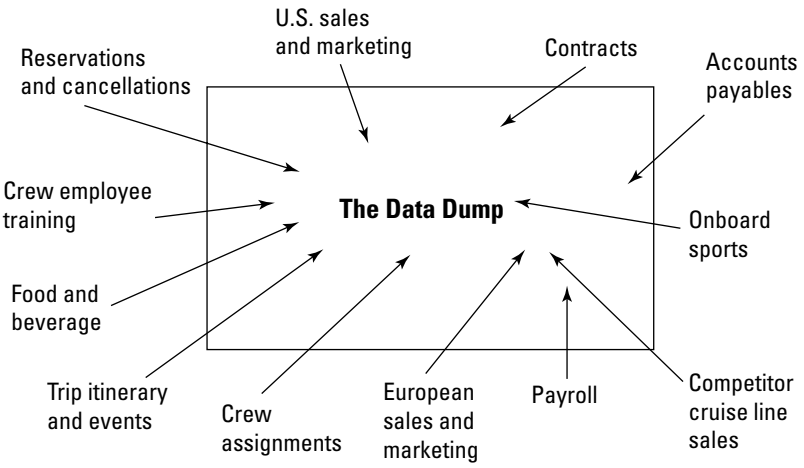


Figure 1-4: What your data dump can look like.

Chapter 2

What Should You Expect from Your Data Warehouse?

In This Chapter

- Making business decisions based on facts, not intuition
- Realizing the value of "data at your fingertips"
- Looking at cross-organization communications and your data warehouse
- Changing your business because of insights from data warehouse information

All too often, the members of a data warehouse development team proudly unveil their creation, get a few "oohs" and "aahs" from their user community, and then find out that "if you build it, they won't necessarily come." The data warehouse just sits there, quietly being restocked every week or every month, and supporting only a few random user information requests until plans eventually are drafted for its replacement.

You can avoid many, or perhaps most, of these unfortunate situations if everyone, from the executive sponsors to the technicians and developers, focuses their efforts on a single question: "What do you do with a data warehouse?"

The contents of the warehouse (the data) aren't important; rather, how the data is used in everyday business life is what makes the warehouse really useful.

Using the Data Warehouse to Make Better Business Decisions

Suppose that John is a district manager at MegaRetroMania, Ltd., a national chain of video-rental stores that specializes in movies from the 1950s and earlier. Mary is John's counterpart in another district. Both excitedly awaited

the rollout of MegaRetroMania's brand-new data warehouse, an event that finally occurred a month ago. John and Mary both attended the data warehousing orientation class in which they figured out how to run reports and make information requests by using the contents of the warehouse. Each person received a handsome, ready-for-framing certificate indicating successful completion of the course.

On July 20, the day after June's results had been finalized and loaded into the data warehouse, John sits down at his PC, clicks the icon that enables him to access the data warehouse, and makes this request:

> Show me, for all my stores, a breakdown of second-quarter sales compared to first-quarter sales, each store's second-quarter sales from a year earlier, and the sales of all competitors within two square miles of each store's location.

While John watches the little "Working, please wait" hourglass icon twirl for a few seconds on-screen, he thinks, "It's fascinating how they can make it seem like the sand is really sifting from top to bottom like that." After the results of his request are displayed on-screen, he prints the report, walks over to the printer to retrieve the five printed pages, flips through them for a few moments, and then thinks, "That's really neat!"

Then he slips the report into the middle drawer on the left side of his desk before leaving for lunch.

And he never looks at it again.

Mary runs the same identical report for her district; after printing the report and flipping through the pages, however, she (unlike John) has a pen in hand and circles one number, underlines another, and writes "Uh-oh!" next to the line containing information for a store that she knows is having problems.

Then, Mary returns to her desk, report in hand, and picks up the telephone. First, she calls her problem-child store and, after the manager comes to the phone, spends the next half hour making follow-on information requests from the data warehouse, discussing the result of each request with the store manager, and considering various options, such as sales promotions.

After Mary takes her regular two-mile lunchtime walk, she calls the manager of another store in Madison County, Iowa, whose second-quarter sales were relatively flat after experiencing at least 15-percent growth every quarter for the past three years (a fact she obtained from the data warehouse when she noted the lack of growth in the second quarter). While the store manager is on the phone, Mary gets a week-by-week sales breakdown from the warehouse, which shows that the first three weeks of the quarter had been fairly

good, with sales during each week slightly ahead of the preceding one. During the fourth week of April, however, a large dip occurred, followed by an even larger dip the following week and yet another drop the week after that (the worst week any store had ever had, it turns out, in the long, distinguished history of MegaRetroMania) before sales began picking up in mid-May.

Suddenly, the store manager has an idea — maybe the Olympic Games were causing the problem! The Olympics had been held in Madison County for the first time ever, and *everyone* had attended. Even that store's primary customer base, the legions of photographers who usually come to Madison County to take pictures of covered bridges during a four-day period and rent movies to watch in their motel rooms at night, were preoccupied for those three weeks by the Olympics.

Satisfied that no burgeoning problem is waiting in the Madison County store — just a once-in-a-lifetime anomaly, most likely — Mary turns her attention to the other annotations on her printout. By the end of the day, five stores have action plans in place for special promotional campaigns to combat competition that's opened in the past few months near those MegaRetroMania locations.

Here's a quiz. True or false: Mary and John both used the data warehouse today.

The answer: False.

You might be thinking, "What? Of course John used the data warehouse today. He ran a report and looked it over."

Although John accessed the data warehouse, he didn't use it. He looked over a printout and stuffed it away forever, and then returned to business as usual. When you compare his action to Mary's, the differences are obvious. No matter what else appears on a data warehousing project's mission statement, and no matter what the project's sponsors say to convey the project's merits to the people who control funding, the primary purpose of a data warehouse is to help people make better business decisions. Data warehousing isn't about simply accessing data and then doing nothing with it; it's about *really using* data.

Here's one way to make sure that data warehouse users act more like Mary than John: When you conduct training sessions before you turn users loose on the new data warehouse, explain to them not only how to access data (which types of queries and reports they can run or which types of data are available, for example), but also include real-world examples of how to use the results of their warehouse access.

The training session for MegaRetroMania district managers (and other warehouse users), for example, might feature end-to-end hands-on training that explains how to get sales reports and what the reports look like. To make sure that users understand the usefulness of the data warehouse, the instruction could also feature role-playing or some other type of training in what to do with those reports to improve the stores' performances.

Finding Data at Your Fingertips

Everyone has heard the phrase "information (or knowledge) is power," and the more skillfully people use information in the course of their jobs, the greater their chances for success.

The process of gathering data from many different sources (if you can even get the data at all) has traditionally been tedious, particularly before computers became commonplace — and it has remained that way during the information age. Some novels and movies would have you believe that after you press just a few keys, you can automatically access vast amounts of data from anywhere in the world, regardless of the platform you're using, the structure of the data (how it's organized), or how the data is encoded or keyed.

The real world isn't quite that orderly, and anyone who has struggled to perform what should be simple tasks (merging Pacific Rim sales data from one system with North American sales data from another, for example) is probably aware of the difficulties.

Suppose that Steve is a district sales manager at BlackAndWhiteVideos, Inc., MegaRetroMania's upstart archrival. Unlike John and Mary, though, Steve has no data warehouse from which to make requests, such as "Show me the top 20 BlackAndWhiteVideo stores across the district in which monthly total comedy video rentals are at least double those of action videos so that I can adjust the inventory accordingly." Instead, Steve has to get the answer to this question the old-fashioned way, as shown in Figure 2-1, by following these steps:

1. He writes and distributes to all store managers in the district an e-mail stating that he wants a report sent to him no later than the fifth day of each month of rental dollars by category (such as comedy, action, and romances), based on transaction records from each store's point-of-sale (POS) computer system.

2. On the fifth day of each month, he looks through the reports received and notes each store from which he hasn't received a report.
3. He spends most of the next two days calling store managers who haven't sent reports and hearing something like, "Oh, wow, I totally forgot all about that; what do you want me to send you again?"
4. While waiting for each of the missing reports, he begins entering into a spreadsheet the numbers from the reports received so far.

   Each month, it occurs to Steve that he should send a new e-mail asking managers to e-mail him a disk with each store's data on it (already in spreadsheet format) so that he doesn't have to reenter the information. He never quite gets around to that request because he assumes that it takes several months to train someone in each store to be able to extract data from the POS application into a spreadsheet.
5. By the 12th of each month, he makes another round of phone calls to the stores that still haven't submitted their reports, and this time threatens to — well, you get the idea.
6. Finally, by the 20th of each month, after the spreadsheet program has all the necessary information, he runs a few simple sorting routines to get the necessary reports.
7. And he prepares to start all over for the next month's data.

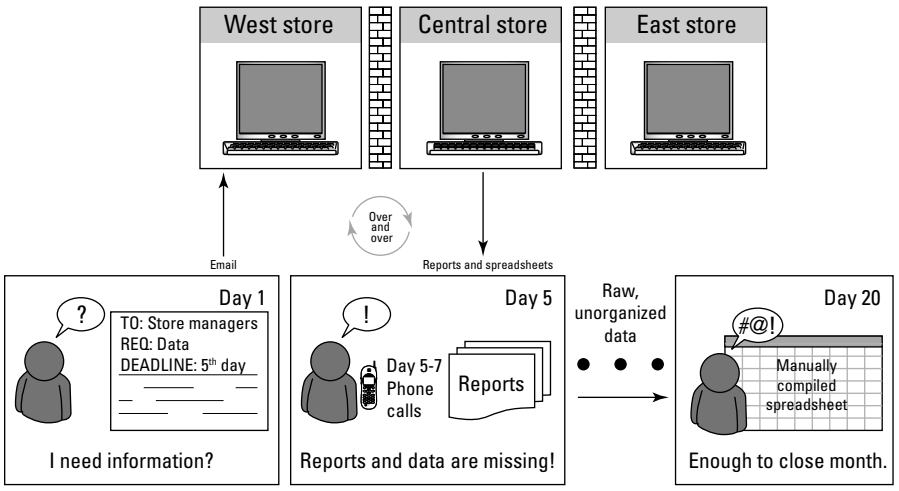


Figure 2-1: Without a data warehouse, data isn't at a user's fingertips — it's usually out of reach!

I have four questions about Mary at MegaRetroMania and Steve at BlackAndWhiteVideos (forget about John at MegaRetroMania — he's a loser):

- Who's making more effective use of technology?
- Who's likely to be more productive in a district manager's role?
- Who'll likely make more timely business decisions with (using business-school talk) a more positive impact on short-term and long-term revenues (top line) and profit (bottom line)?
- Who's more likely to be frustrated and look for another job that involves much less wasted time?

(Answers: Mary, Mary, Mary, Steve)

Both Mary and Steve *could* have access to the same information and in the same way because both MegaRetroMania and BlackAndWhiteVideos have similar computer systems with similar data. The difference is that because MegaRetroMania has invested in data warehousing technology, Mary and her counterparts (even that loser John, if he would only get with the program) can make information-based and timely business decisions.

Here are two more questions to consider:

- Does your current job more closely resemble Steve's or Mary's in how you're able to get access to information you need?
- Based on how you answered the first question, would you rather have a job more like the other's?

Facilitating Communications with Data Warehousing

A benefit of data warehousing that's much less tangible than having information for better business decisions is that data warehousing often facilitates better communications across a company than what existed before the warehouse project began:

- The information technology (IT) organization — the organization that handles infrastructure (hardware and software platforms, networking, and communications, for example) — begins working more cooperatively with its customers in the business organizations.

- Business organizations that are drawn together as part of a data warehousing project often gain more appreciation for each other's missions and challenges.

Part of the better communications picture occurs (usually) between applications development and customers in the organization's business units.

Data warehousing can promote increased levels of communication across different business units.

IT-to-business organization communications

If you're a veteran of an IT-organization-versus-business-organization war in your company, you might have won some battles and lost others. Make no mistake about it, though: The other side is the enemy!

If you're on the IT side, you might grumble about these issues:

- Business users make unrealistic requests for new applications and enhancements to existing applications.
- Users don't participate during application development and testing.
- Users are so technology-challenged that they require hand-holding for the simplest application functions.

On the other side, business users commonly make these types of complaints:

- IT application developers have no idea about how the company's business is run and usually don't even bother to ask.
- Developers just put whatever features they want into the applications, regardless of whether those features are applicable to the business processes.
- IT is too slow in responding to requests for support.

Since the early 1980s, when the then-new personal computer and local-area network permitted business organizations to take direct responsibility for part of their information management and application capabilities, many companies have experienced an era of mistrust and lack of communications between IT and business. This level of mistrust has continued to grow with the proliferation of business productivity tools such as Microsoft Excel — enabling users to formulate their own solutions without IT. And the war

within IT has also grown with the concept of Central IT, which manages the shared services within IT, and Line of Business (LOB) IT, which is closer to the business customer and manages his or her specific applications. Often, the data warehouse is managed by Central IT, which *isn't allowed to talk with* the business — people from Central IT can talk only to the LOB IT personnel. Talk about dysfunctional!

Data warehousing projects often give organizations a chance to mend the rift between IT and business users, as well as LOB IT and Central IT. The nature of data warehousing (for example, the focus on business questions and the data necessary to answer those questions) gives people from both sides an opportunity to better understand the other group's roles and concerns.

Increased IT-business organization communications often leads to more effective and efficient working relationships because everyone develops a degree of trust. A successful project or two usually can do wonders in achieving increased cooperation.

You don't necessarily get better communications between IT and business organizations just because they jointly embark on a data warehousing effort. For example, if you use the word "dysfunctional" to describe the relationship between IT and the business community in your company, you might want to look at some team-building workshops or other remedial training before proceeding with a data warehousing project.

Communications across business organizations

Data warehousing involves, in many ways, the use of technology to break down barriers that are inherent in large companies, such as different computer systems performing similar functions for different groups of users, or different databases with information that would be better kept in a single database. A sales-and-marketing organization responsible for one of its company's products naturally wants to help integrate its data (performance results) with that of another sales-and-marketing organization, responsible for a different product, that uses a different computer system (maybe because of a corporate acquisition), right?

Corporate intrigue and turf wars are the hallmark of business. Business executives and their staff members usually think of information not as a *corporate* strategic asset, but rather as a *personal* one.

The breaking-through-barriers nature of data warehousing usually takes a top-level mandate from a chief executive officer (CEO) or chief operating officer (COO): "Thou shalt all cooperate, or thou shalt look for another job." That type of mandate is probably a corporation's best chance for overcoming detrimental corporate politics.

Suppose that you're a senior company executive at a consumer products company, and that company has an environment that looks like the one shown in Figure 2-2. You decide that you want to dedicate a large part of your capital budget to enable your managers and analysts to get timely access to data so that they can make better business decisions. Implementing a data warehouse would move the data from the separate division's product sales applications into one central location for the users to access.

Using the cooperate-or-else mandate, you can ask — and receive answers to — the barrier-busting questions that present you with a consolidated picture of your sales, regardless of how many source systems the data come from. In addition, after the first successful collaborative, cross-organization data warehousing effort, people probably feel less territorial and work more closely together for the common good of the company.

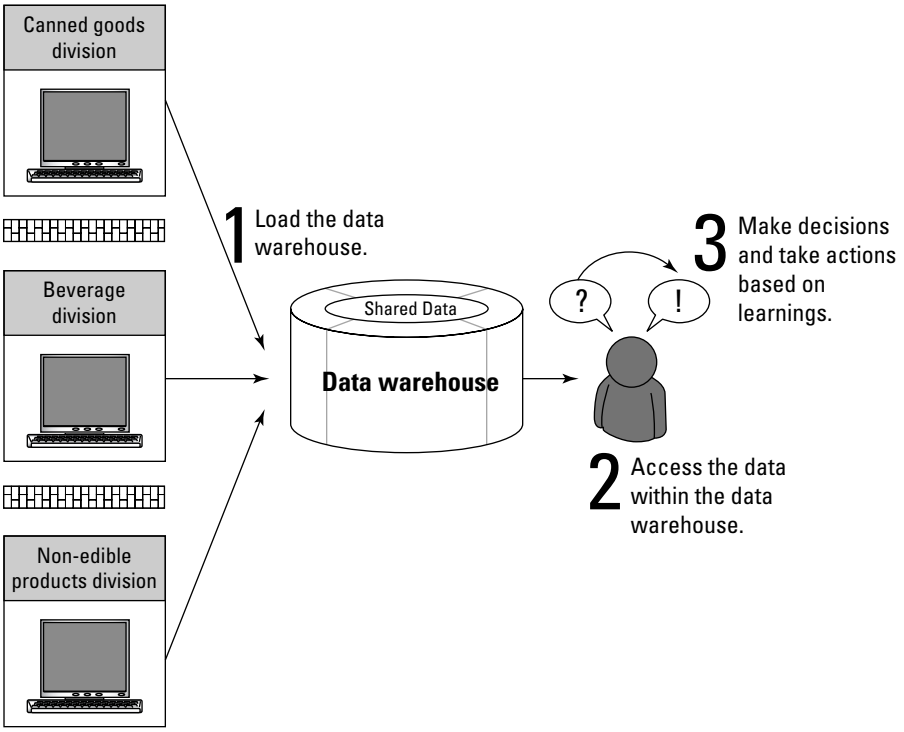


Figure 2-2: A typical data warehousing environment with different division's product sales data stored on a variety of application systems that go into the communal data warehouse.

Or they might not. Any lack of cooperation on the part of the leaders and members of a particular business organization, however, becomes extremely apparent because their data either aren't in the warehouse (often called a *data gap*) or, worse, the data warehouse project fails — and you can easily tell who's at fault, the data gap owner.

I don't mean to imply that you need to make finger-pointing and blame-casting a part of your data warehousing project; rather, you should implement, from the beginning, a clear directive for cooperating with peers that everyone has to follow.

Facilitating Business Change with Data Warehousing

Data warehousing involves facilitating change in business processes. In addition to being able to make better, information-driven operational and tactical decisions, you gain insight into key areas that can help you make strategic decisions about the fundamental aspects of your business.

Your data warehouse can act as an early-warning system to let you know that you might need to make some major business changes. Imagine that it's 2000 and you just got into the video-rental business. You're the part owner and purchasing manager for a regional chain of retail stores that rents digital video discs (DVDs) and video games, and sells compact discs (CDs). If you had a data warehouse that gathered information from the point-of-sale computers in all your stores and external data about your competitors' sales activity (purchased from an external source), you probably would have noticed a couple of interesting trends when you looked at product revenues over time:

- Sales of music in their historically popular CD format have dropped off remarkably. Because your competitors' sales for those items are also way off, you might decide that you don't need to devote the shelf space to stock these types of items anymore.
- Video games for Playstation 2 (PS2) are rented much more frequently than the same games in Dreamcast format. Because the video game business is rapidly evolving (remember, it's 2000), your rental numbers might be declaring PS2 the winner of the next-generation gaming consoles and that the Dreamcast games are another item probably not worth ordering and putting on the shelves.

Both these decisions are, for the most part, tactical because they involve product-stocking decisions. Neither decision fundamentally changes the way your company does business, but rather helps you decide which products you want to keep in stock.

But suppose that trend-line information coming back from the data warehouse shows relatively flat sales of all types of music and dramatically increasing revenues from DVD and video game rentals. The future of your chain might lie not in sales of music at all, but rather in DVD and video game rentals. Rather than have to rely on a gut feeling, you can make data-driven strategic decisions about how your business should change.

Imagine having to make the same type of decision in 2008 if you're in the automotive business. Gas guzzler sales are plummeting as fast as the price of gas is increasing. While worldwide demand for oil grows, the supply and demand curve isn't friendly to traditional car sales. The highly positive growth curve that sport utility vehicles (SUVs) had a few years ago is giving way to the hybrid and green products. Again, your data warehouse gives you the information to make data-based decisions, rather than have to rely on hunches or, at best, bits and pieces of data gathered from various computer systems.

Chapter 3

Have It Your Way: The Structure of a Data Warehouse

In This Chapter

- Constructing your data warehouse to fit your business needs
- Understanding the different data warehouse architectures
- Deciding whether to make your data warehouse centralized or distributed

Although I generally dislike trite sayings, I have to make an exception: "No two data warehouse implementations are exactly alike."

One of the worst data warehousing mistakes you can make is to try to force your business analysis and reporting needs to fit into an environment that you copied from somewhere else. Although a certain amount of analysis is standard across companies, the interpretation and actions you formulate from your data assets can give you a competitive edge in the global marketplace.

Leverage the knowledge and experience of other people by studying their experiences with data warehousing products; asking questions about the most difficult problems they encountered during product development and after they put their data warehouse into use; and determining how effective their users have been in making better, information-based business decisions.

Don't automatically assume, however, that every aspect of someone else's data warehousing environment is exactly right for you.

Ensuring That Your Implementations Are Unique

A data warehouse is composed of many different components, each of which can be implemented in several (perhaps many) ways. These components include

- The breadth; the number of different subjects and focus points, for example, or the number of different functional or regional organizations that will use it
- The number of sources that will provide raw data
- The means by which data is moved from source applications and loaded into the data warehouse
- The business rules applied to the raw source data to produce high quality data assets
- The target databases in which data assets are stored
- The data assets; the elements, the level of detail in each element, and how much history is being maintained, for example
- The business intelligence, front-end tool used to access the data assets
- The overall architectural complexity of the environment

No two data warehouse implementations (neither the implementations now in existence nor all those to be completed in the future) will be identical in all the preceding eight categories.

Two companies in the same industry, for example, each might have a sales-and-marketing data warehouse that supports 300 users across four different business organizations, allows access by using the same business intelligence tool set, and uses the same database management system in which to store approximately 50 gigabytes of data.

The two companies probably have these differences, however:

- Different data sources, unique to each company
- Different data, as a result of the different sources; for example, reference data that defines stages in the sales and qualification process
- The use of different source-to-warehouse movement techniques — for example, business rules for forecasting future revenue

Because of these differences, trying to adapt another company's data warehousing solution in its entirety would be a big mistake. (Or, like your schoolteachers used to tell you, "Do your own work — no copying!")

Classifying the Data Warehouse

Although you must ensure that your data warehouse fits your own unique needs, some guidelines can help you determine the probable complexity of its environment and structure. I use a three-tier classification for planning

a data warehouse. By determining a likely category for an implementation, I have — early in the project — some specific guidelines for the project's complexity, development schedule, and cost.

Here are my classifications:

- **Data warehouse lite:** A relatively straightforward implementation of a modest scope (often, for a small user group or team) in which you don't go out on any technological limbs; almost a low-tech implementation
- **Data warehouse deluxe:** A standard data warehouse implementation that uses advanced technologies to solve complex business information and analytical needs across a broader user population
- **Data warehouse supreme:** A data warehouse that has large-scale data distribution and advanced technologies that can integrate various "run the business" systems, improving the overall quality of the data assets across business information analytical needs and transactional needs

Each of these classifications of data warehouses implements various aspects of an overall data warehousing architecture, as shown in Figure 3-1.

This architecture assures that your data warehouse meets your user's information requirements and focuses on the following business organization and technical-architecture presentation components:

- **Subject area and data content:** The content of a data warehouse is grouped by subject areas. A *subject area* is a high-level grouping of data content that relates to a major area of business interests, such as customers, products, sales orders, and contracts. Subject area and data content will drive your user access to this data and the associated presentation through business intelligence tools.
- **Data source:** Data sources are very similar to raw materials that support the creation of finished goods in manufacturing. Your data sources are the raw materials that are refined and manufactured into the subject area's data content. Depending on the class of data warehouse you're building, you might have more comprehensive data sources — all dependent on the business user's requirements.
- **Business intelligence tools:** The user's requirements for information access dictate the type of business intelligence tool deployed for your data warehouse. Some users require only simple querying or reporting on the data content within a subject area; others might require sophisticated analytics. These data access requirements assist in classifying your data warehouse.

Data Warehouse

	Lite	*Deluxe*	*Supreme*
Scope	Personal departmental scope with minimal investment for specific purpose and smaller segment of the business	Line of Business through enterprise scope with mid-sized investment to assist in integrating the business	Enterprise scope with large multi-year investment to drive strategy and integrate various aspects of the business incorporating internal and external user populations
Technology			
Visualization	Managed reporting and business analysis	Business Intelligence suite (query, reporting, business analysis, dashboards, scorecards)	Business Intelligence suite + (query, reporting, business analysis, dashboards, scorecards) along with intelligent agents
User Access	1 to 2 subject areas	1 to 25 subject areas	Unlimited access to internal and external data
Target Data Structures	Small OLAP cubes (MDB) and/or relational database	Multi-tiered data layer including operational data store, data warehouse supreme feeding data warehouse deluxe and lite	Federated Query layer along with traditional data stores including structured and unstructured data
Data Quality & Movement	File transfer with simple business rules to extract, transformation, and load (ETL) tools	Extract, transform, and load (ETL) tool with data quality tools for complex rules like house holding	Extract, transform, and load (ETL) tools along with federated query middleware
Sources	1 to 25 sources	25 to 100 sources	Unlimited sources

Figure 3-1: You can classify a data warehouse by these features.

- **Database:** The database refers to the technology of choice leveraged to manage the data content within a set of target data structures. Depending on the class of data warehouse, a personal, departmental, or enterprise database management system may be required.
- **Data integration:** *Data integration* is a broad classification for the extraction, movement, transformation, and loading of data from the data's source into the target database. This is where the business rules are put to action to assure that the data content is of the highest possible quality for broad user adoption.

The data warehouse lite

A data warehouse lite is a no-frills, bare-bones, low-tech approach to providing data that can help with some of your business decision-making. No-frills means that you put together, wherever possible, proven capabilities and tools already within your organization to build your system.

The term *data mart* is commonly used to refer to what I call the data warehouse lite. A data mart (as described in Chapter 4) should be, in its purest sense, an environment that receives a portion of a data warehouse's content, providing easier access to this information subset for a select group of users. As often happens in information technology (IT) areas in which vendors jockey for market share, however, many people now think of a data mart as little more than a small-scale data warehousing environment.

Don't be confused by the terminology! I prefer my lite-deluxe-supreme classification because it's easy to remember and helps me visualize the complexity of the environment I'm setting out to build. When I discuss project opportunities with clients, however, and they mention that they want to build a data mart, they're usually referring to what I call a data warehouse lite. The wording doesn't really matter because all these definitions are continually being revised anyway. If you concentrate on the aspects of the environment that drive the overall complexity of the implementation — breadth, database, data content, tools, extraction and movement, and architecture — you can avoid getting confused.

Subject areas and data content

A data warehouse lite is focused on the reporting or analysis of only one or possibly two subject areas. Suppose that in your job at a wireless division of a telephone company, you analyze the sales of services such as in-network minutes, out-of-network minutes, text messaging, Internet access, and other mobile usage to consumer households. If you build a data warehouse lite exclusively for this purpose, you have all the necessary information to support your analysis and reporting for the consumer market. You don't have any information about business users' and payment history, however, because that information is part of a different subject area, as shown in Figure 3-2.

Based on the subject area limitation, a data warehouse lite has just enough data content to satisfy the primary purpose of the environment, but not enough for many unstructured what-if scenarios its users might create. You must choose carefully, therefore, from among the set of all possible data elements and select a manageable subset — elements that, without a doubt, are important to have. This process is the same for any data warehouse implementation, except that you must be extremely disciplined when you're making decisions about what content to include.

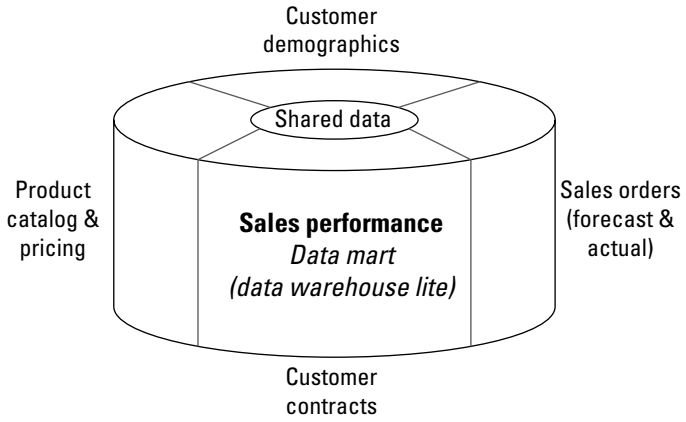


Figure 3-2: A data warehouse lite has a narrow subject-area focus.

Use standard reports, particularly those that currently require a great deal of manual preparation, as one of your primary guides to determine data content in a data warehouse lite.

Data sources

A data warehouse lite has a limited set of data sources — typically, one to a handful. As part of an overall single-application environment, for example, the data warehouse lite acts as the restructuring agent for the application's data to make it more query- and report-friendly.

The most common means of restructuring a single application's data is to denormalize the contents of the application's relational database tables to eliminate as many *relational join operations* (the process of bringing together data from more than one database table) as possible when users run reports or do simple querying. Denormalization is the opposite of the relational database concept of normalization, a somewhat complex set of guidelines that tells you which data elements should be in which tables in a database (see Chapter 5). For purposes of data warehousing, denormalization is the important concept to understand, and, fortunately, it's a simple one. When you *denormalize* a database, you don't worry about duplicated data; you try to create rows of data in a single table that most likely mirrors the reports and queries that users run. Figure 3-3 shows an example of a single-source data warehouse lite built on denormalization.

Although you can use externally provided data (see Chapter 19) in a data warehouse lite implementation, the data you use is rarely newly acquired. You're more likely to incorporate data that you already use for analysis (perhaps in a stand-alone manner).

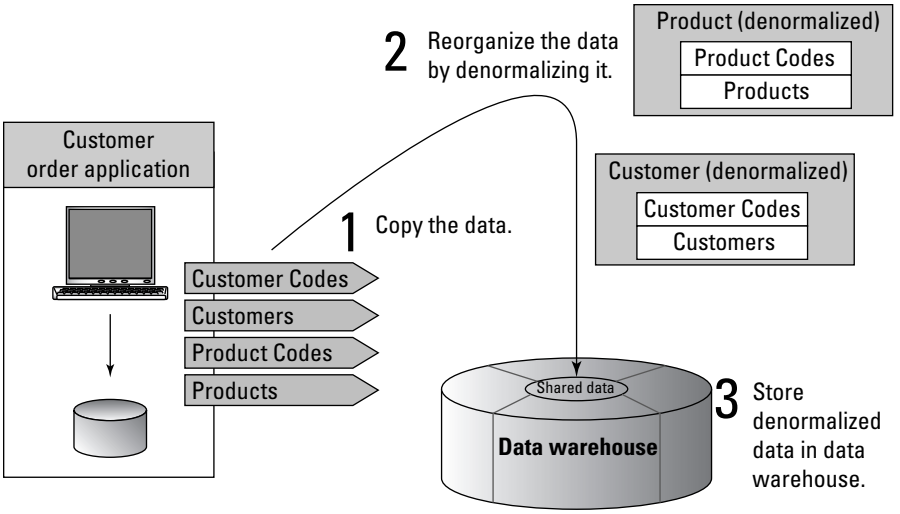


Figure 3-3: Denormalizing data from a single application restructures that data to make it more conducive to reporting needs.

Business intelligence tools

The users of a data warehouse lite usually ask questions and create reports that reflect a "Tell me what happened" perspective, as described in Chapter 9. Because those users don't do much heavy-duty analytical processing, the products they use to access the data warehouse should be easy for them to use. Although some power users might use more advanced tools to access the data warehouse for more complicated processing, they're a minority of the user community.

Database

Data warehouse lite solutions are limited by users, data content, and the type of business intelligence tools utilized. These limitations are the primary reason that a data warehouse lite is usually built on a standard, general-purpose relational database management system (you can find out about RDBMSs in Chapter 5). Your organization's general familiarity with RDBMSs, particularly the easy-to-use structure built around tables and columns, can help users easily access the managed data content. In some situations, though, a multidimensional database (MDB) is used, as described in Chapter 6, because of your users' knowledge of the data content and their need or desire to analyze more than to report.

For a data warehouse lite implementation, you can use either a general-purpose RDBMS or MDB product. Because of relatively modest data volumes and the narrow subject-area focus, you probably don't have to worry about technical barriers that might drive your database platform decision one way or the other.

Data extraction, movement, and loading

Simplicity is the name of the game in a data warehouse lite. Therefore, make the process of extracting data from sources and performing all the functions necessary to prepare that data for loading as straightforward as possible by using these two elements:

- Simple file extracts from the run-the-business systems and file transfers that allow you to move data from its sources to the data warehouse lite
- Straightforward custom code (or perhaps an easy-to-use tool) that can extract and move the data

If the data source for your data warehouse lite is built on a relational database and you're planning to use the same database product for your data warehouse, use SQL to easily handle data extraction and movement. These steps — as shown in Figure 3-4 (they're easier to understand than SQL code!) — provide a standard procedure for this process (you'll want to tailor these steps to your particular environment, of course):

1. **On the system that houses your warehouse, use the SQL CREATE TABLE statement to create the definition for each table in your data warehouse lite.**
2. **Create a database backup that contains copies of all tables from the source that provide data to the warehouse, and then reload those tables into a staging area on the system where you plan to locate your data warehouse.**

   You should ensure that the network bandwidth and time window are adequate to copy all the source tables to the system by using a file-transfer program.
3. **Use the SQL INSERT statement, with a nested SELECT statement specifying the source tables and their respective columns that will populate the data warehouse table (and how the tables will be joined), to load the data into your data warehouse lite.**
4. **Run a series of quality assurance (QA) routines to verify that all data has been loaded properly.**

   Check row counts, numeric totals, and whatever else you can.

Architecture

The architecture of a data warehouse lite is composed of the database used to store the data, the front-end business intelligence tools used to access the data, the way the data is moved, and the number of subject areas. The watchword of this environment is minimalist: no bells, no whistles, nothing fancy — just enough technology applied to the environment to give users access to data they need.

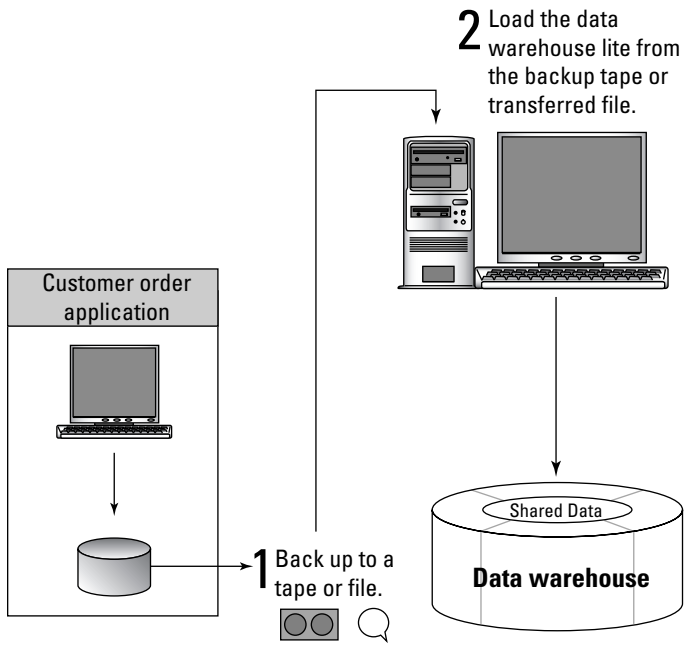


Figure 3-4: The low-tech approach to moving data into a data warehouse lite: database backup tapes.

The architecture of a data warehouse lite, as shown in Figure 3-5, contains these major component types:

- A single database contains the warehouse's data.
- That database is fed directly from each of the sources providing data to the warehouse.
- Users access data directly from the warehouse.

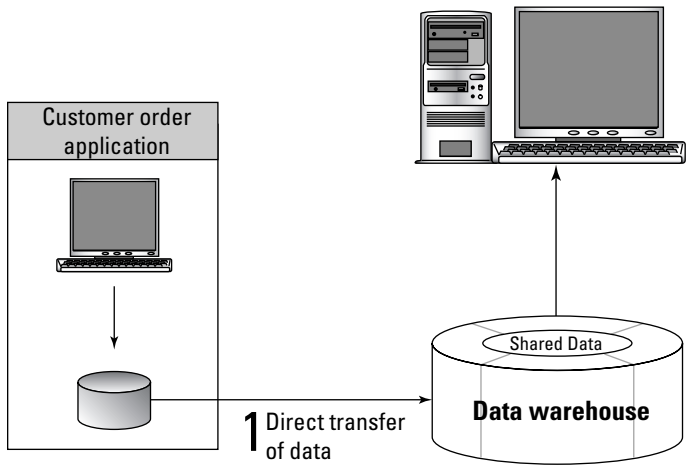


Figure 3-5: The architecture of a data warehouse lite is built around straight-line movement of data.

The data warehouse deluxe

You'll most likely focus most of your data warehousing-related activities on the data warehouse deluxe environment, as shown in Figure 3-6. Data from many different sources converge in these "real" data warehouses, which make available a wealth of architectural options that you can tailor to meet your specific needs.

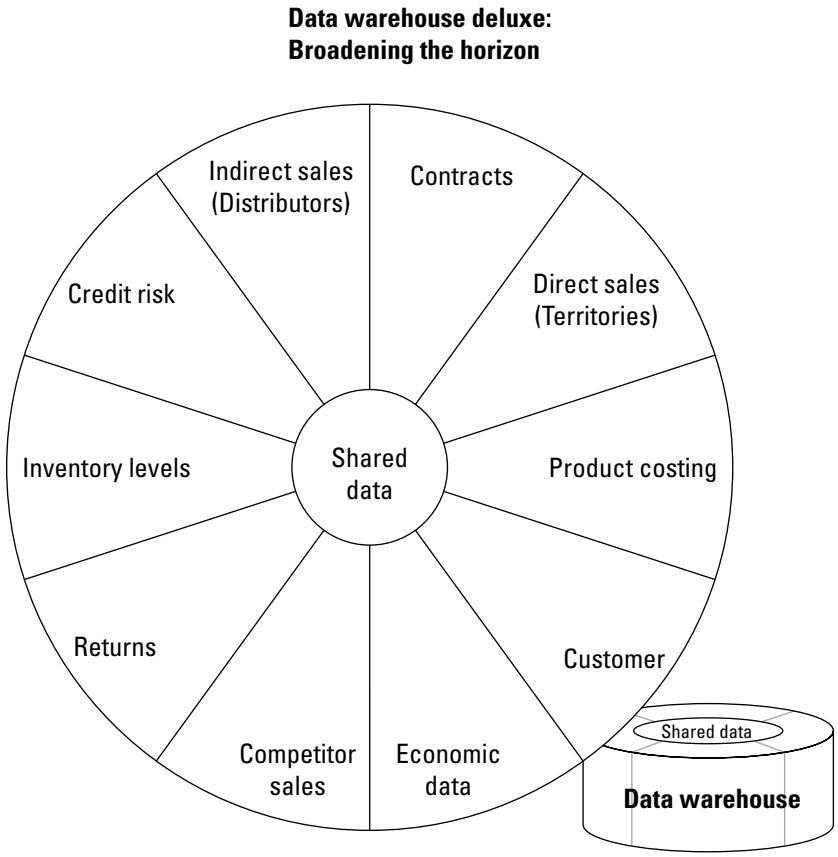


Figure 3-6: A data warehouse deluxe has a broader subject-area focus than a data warehouse lite.

Subject areas and data content

A data warehouse deluxe contains a broad range of related subject areas — everything (or most things) that would follow a natural way of thinking about and analyzing information.

In a data-warehouse-deluxe version of the telephone-company example I give in the "Subject areas and data content" section of the data warehouse lite description, earlier in this chapter, you likely find not only the subject area of consumer wireless services (among other items), but also these elements:

- Consumer basic calling revenues and volumes
- Consumer long-distance calling revenues and volumes
- Consumer wireless calling revenues and volumes
- Business wireless services
- Business basic calling revenues and volumes
- Business long-distance calling revenues and volumes
- Business wireless calling revenues and volumes
- Internet access (DSL) services
- Internet revenues and volumes

The subject range is broader than a data warehouse lite for a data warehouse deluxe because

- The user base is broader (more organizations have their people use the data warehouse).
- The scope of any given user's queries and reports is broader than just one or two subject areas. For example, a user might run reports comparing trends in add-on services for businesses and consumers to see where to concentrate future sales-and-marketing efforts.

When you implement a data warehouse deluxe, you almost always need access capabilities (unlike with a data warehouse lite), in addition to simple results reporting. Therefore, although you might be able to use standard reports as a starting point when you're deciding what should be in your warehouse, that's rarely enough. Follow these steps to thoroughly understand your source systems:

1. **Take a complete inventory of available information.**

   This inventory is called a *source systems analysis,* as discussed in Chapter 16.

2. **Review each candidate source element and answer these questions:**

   - What data do you need to include in the data warehouse and what should you leave out?

- What information should be summarized and what should be left at the detailed level?
- What data should remain in the data warehouse forever, and what data should you purge from the data warehouse after it has aged?
- What else do you need to know about the data before you put it in your data warehouse?

This step is one of the most severe tests of how well the IT people and business users get along throughout the data warehousing project.

Data sources

You won't be lucky enough to find any single-source environments when you're building a data warehouse deluxe.

Now, you have a whole new set of — I have to use the word — problems that you must deal with, including the ones in this list:

- **Different encodings for similar information:** Different sets of customer numbers come from different sources, for example.
- **Data integrity problems across multiple sources:** The information in one source is different from the information in another when they should be the same.
- **Different source platforms:** As an example, an IBM mainframe that has DB2/MVS databases might contain the data in one of the sources, another IBM mainframe that has VSAM files might have another set of source data, a set of servers might contain data within Oracle databases, and the rest of the source data might all be stored in SQL Server databases on Windows servers.

Although the exact number of data sources depends on the specifics of your implementation, data warehouse deluxes tend to have an average of eight to ten applications and external databases that provide data to the warehouse.

Business intelligence tools

The broad range of subject areas and the wealth of data in a data warehouse deluxe means that you usually have several different ways of looking at that warehouse's contents. This list shows the different ways that you can use a data warehouse (Part III discusses them in-depth):

- **Simple reporting and querying:** Like with data warehouse lite, the purpose of the warehouse deluxe is to "Tell me what happened."
- **Business analysis:** You use the warehouse to "Tell me what happened — and why."
- **Dashboards and scorecards:** In this model, a variety of information is gathered from the data warehouse and that information is made available to users who don't want to mess around with the data warehouse — they want to see snapshots of many different things. Its purpose is to "Tell me a lot of things, but don't make me work too hard to get the answers I want."
- **Data mining or statistical analysis:** In this area, statistical, artificial intelligence, and related techniques are used to mine through large volumes of data and provide knowledge without users even having to ask specific questions. Its purpose is to "Tell me something interesting, even though I don't know what questions to ask, and also tell me what might happen."

You're likely to employ at least three — and perhaps all four — of these types of data warehouse user-access techniques when you use a data warehouse deluxe. Although tool vendors increasingly try to provide suites of products to handle as many of these different functions as possible, you do have to deal with different products — and so does your user community.

Don't assume that you can simply select a single vendor whose products satisfy all the business intelligence capabilities your users need. Make sure that you carefully check out the vendors' products — all of them — because you have no guarantee that a top-notch OLAP vendor's data mining tool is equally as good, for example. Don't be afraid to mix and match; you have no reason to shortchange your data warehouse's users simply to avoid having to deal with one more vendor.

When evaluating your user access needs, ask yourself the following questions:

- Do my users want the best-of-breed tools, which might not necessarily be integrated, requiring professional developers to build visualization solutions?
- Do my users want a well-integrated platform that enables integration between user-access strategies so that they can develop all visualization solutions themselves?

Answers to these questions (and if you answer "Yes" to one of them, you answer "No" to the other) can help you evaluate the business intelligence tools.

Database

Data warehouse deluxe implementations are big — and getting bigger all the time. Implementations that use hundreds of gigabytes (a gigabyte equals 1 billion bytes) and even terabytes (1 trillion bytes) are increasingly more common. To manage this volume of data and user access, you need a very robust server and database.

Data extraction, movement, and loading

Prepare for the challenge! With a data warehouse lite, you can usually handle source-to-warehouse movement of data in a straightforward, low-tech manner — but with the data warehouse deluxe, you're now entering the Difficulty Zone, where many data warehousing projects meet their Waterloo.

You're likely to experience difficulty in this domain for several reasons:

- **You're dealing with many different data sources, some of which might contain overlapping data.**

  For example, suppliers' information might come from two different purchasing systems, and some of your suppliers have entries in both systems. You'll probably run into different sets of identifiers that you have to converge (for example, six alphanumeric characters that are identified as the SUPPLIER_ID in one of the systems and a unique integer known as SUP_NUM in the other).

- **If your data warehouse is large (measuring more than about 250 gigabytes), you're likely to experience difficulties in extracting, moving, and loading your batch windows.**

  Batch windows, the time frames in which updates are made to the warehouse, are complicated by the number of data sources you have to handle.

- **The chances of having a messed-up extraction, movement, transformation, and loading process is exponentially related to the number of data elements to be loaded into the data warehouse.**

  If you could assign some difficulty factor (an integer, for example) to the process of getting data into the warehouse, the following measures would hold true: You have *n* data elements that you want to include in the data warehouse with a difficulty factor of *x.* If you now

have $2n$ data elements, your difficulty factor isn't $2x$; rather, it's x squared.

To make this difficulty factor easier to understand, assign some numbers to n and x. Say that your data warehouse has 100 elements (n) and the difficulty factor (x) is 5. If you double the number of elements (n = 200), your difficulty factor is 25 (5 squared), not 10 (5×2).

- **The process of dealing with so many data sources, all headed toward one place (your data warehouse deluxe), has all the elements of too many cooks in the kitchen, or whatever that saying is.**

  To make the extraction, movement, transformation, and loading process go smoothly, you probably have to deal with many different application owners, official keepers of the database, and other people from a variety of different organizations, all of whom have to cooperate like they're part of a professional symphony orchestra. The reality, though, is that they perform more like a group of kindergarten students who each pick a musical instrument from the toy bin and are told, "Now play something!" Although the process isn't necessarily doomed to failure, expect a number of iterations until you can get the data warehouse deluxe loaded just right.

Architecture

A data warehouse deluxe can have three tiers (like a data warehouse lite), and it has architecture similar to what's shown in Figure 3-5, except with more data sources and perhaps more than one type of user tool accessing the warehouse. But the architecture for a data warehouse deluxe probably looks more like what's shown in Figure 3-7. In addition to other necessary "way stations" for your particular environment, your environment might have these elements:

- **Data mart:** Receives subsets of information from the data warehouse deluxe and serves as the primary access point for users. (Data marts are discussed in depth in Chapter 4.)
- **Interim transformation station:** An area in which sets of data extracted from some of the sources undergo some type of transformation process before moving down the pipeline toward the warehouse's database.
- **Quality assurance station:** An area in which groups of data undergo intensive quality assurance checks before you let them move into the data warehouse.

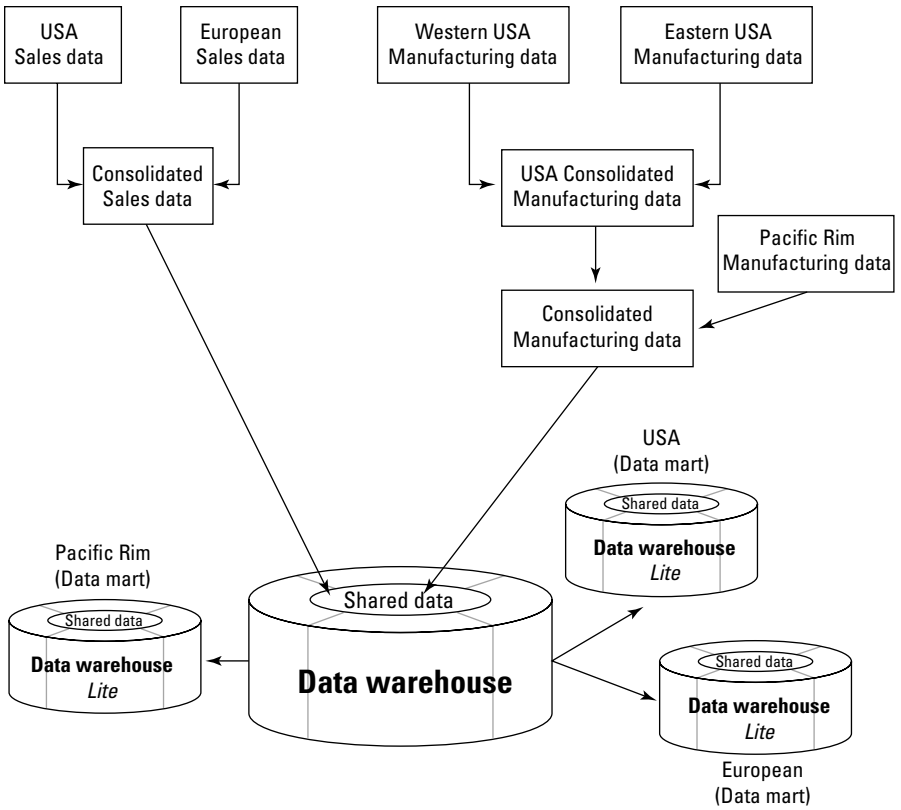


Figure 3-7: A data warehouse deluxe often has a complicated architecture with many different collection points for data.

The data warehouse supreme

Although today's state-of-the-art data warehouse typically looks like a complicated data warehouse deluxe, if you read the following sections, you can know what the data warehouse of tomorrow will look like. There are few enterprises that have ventured in this direction, though due to overall cost and capabilities, it is still rare to find many data warehouse supremes.

Subject areas and data content

The number of subject areas in a data warehouse supreme is unlimited because the data warehouse is virtual; it isn't all contained in a single database or even within multiple databases that you personally load and maintain. Instead, only part of your warehouse (probably a small part) is

physically located on some data warehouse server; the rest is out there in cyberspace somewhere, accessible through networking capabilities as though it were all part of some physically centralized data warehouse. With a data warehouse supreme, your warehouse users have an infinite number of subject-area possibilities — anything that could possibly be of interest to them.

Think of how you use the Internet today to access Web sites all over the world — sites that someone else creates and maintains. Now, imagine that each of those sites contains information about some specific area of interest to you — rather than advertising, job ads, electronic storefronts, and whatever else you spend your time surfing the Internet trying to find. Also imagine that you can query and run reports by using the contents of one or more of these sites as your input. That's the model of the data warehouse supreme: opening up an unlimited number of possibilities to users.

The leading-edge corporations are beginning to pursue and deliver seamless convergence of different types of data: narrative documents, video, image, and ordinary data (such as numbers and character information). A data warehouse supreme has all this — all the different types of data that you need to support better business decision-making.

In terms of total capacity, a data warehouse supreme is huge; it surpasses today's limits. The distribution of the information across many different platforms, much faster and higher-performance networking infrastructure, and increasingly "smarter" database management systems — in addition to, of course, steadily increasing disk storage capacities — create this capacity expansion.

Data sources

Because of the wide breadth of subject areas in a data warehouse supreme, it has numerous data sources. The good news: Because many of the sources are external to your own warehousing environment, you aren't personally responsible for all the extraction, transformation, and loading to get them into your warehouse. The bad news: Someone has to perform those tasks, and you have little or no control over elements such as quality assurance processes or how frequently the data is refreshed.

I have more good news, though: Because the most critical part of a data warehouse supreme is still *internally acquired data* (the data coming from your internal applications), from that aspect, the things you do today to make the data warehouse-ready will still be done in the future.

Because you populate your data warehouse supreme with multimedia information — in addition to traditional data, such as numeric, alphabetic, and dates — the types of data sources broaden from traditional applications to video servers, Web sites, and databases that store documents and text.

Business intelligence tools

As far as I can tell, the Big Four types of business intelligence discussed in the section "Business intelligence tools" in the discussion of the data warehouse deluxe, earlier in this chapter — basic reporting and querying, business analysis, dashboards and scorecards, and data mining — are all part of the data warehouse supreme environment. Of the four, the most significant advances and improvements during the next few years probably will occur with data mining while vendors push enhancements into their products. However, these user-access methods will be relegated to providing information that will be visualized in other forms. The business intelligence tools will enable users to pull information from the data warehouse supreme and integrate it with a better visualization — for instance, Google Earth or Microsoft Virtual Earth. Such combinations, known as *mash-ups,* are becoming more prevalent and enable users to see the data from the data warehouse supreme in more realistic forms — not columns on a report, but dots or shadings on a map.

The biggest difference between today's state-of-the-art data warehouses and the data warehouse supreme, however, is the dramatically increased use of push technology. By using intelligent agents ("assistants" you program to perform certain functions for you), you can have information fed back to you from the far ends of the Internet-based universe, not to mention your own large data warehouse servers within your own company. Figure 3-8 illustrates some of the ways in which intelligent agents can help you make very efficient use of data warehousing.

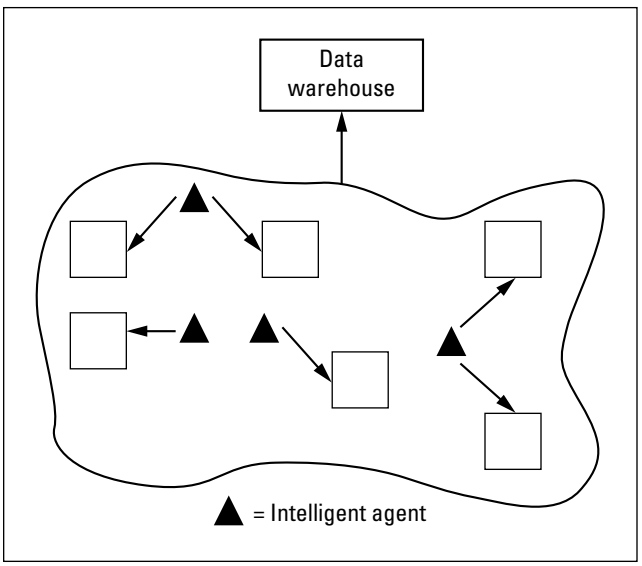


Figure 3-8: Intelligent agents are an important part of the push-technology architecture of a data warehouse supreme.

Database

A data warehouse supreme most likely consists of a database environment that meets these requirements:

- It's distributed across many different platforms.
- It operates in a location-transparent manner: Users make queries that access data from the appropriate platform without the users having to know the physical location (in much the same way that you access Internet Web sites by name, rather than by network address).
- It has object-oriented capabilities to store images, videos, and text in addition to the traditional data, such as numeric and date information.
- Because of dramatically faster performance than current data warehouses, it increasingly permits you to access data directly from transactional databases without having to copy the information to a separate data warehouse database.

Data extraction, movement, and loading

Here's how the extraction, movement, and loading of data occurs in a data warehouse supreme:

- Data that's moved (copied) from a source application's database or file system into a separate database in the data warehouse is handled almost identically to how you perform those tasks in a data warehouse deluxe.

- The increasing use of *operational data stores,* or ODSs (real-time availability of analytical data so that you don't have to deal with delayed access) means that more messaging occurs between your data sources and your warehouse database. The data source determines when data should be moved into the warehouse environment, so the warehouse doesn't have the responsibility to request updates and additions. When new data is inserted into the source database (or existing data is modified or deleted), the appropriate instructions and accompanying data are sent to the warehouse.

Architecture

Figure 3-9 shows an example of what the architecture of a data warehouse supreme might look like. But with all the upcoming technology trends and improvements discussed in the preceding sections, your data warehouse supreme can look like (almost) anything you want.

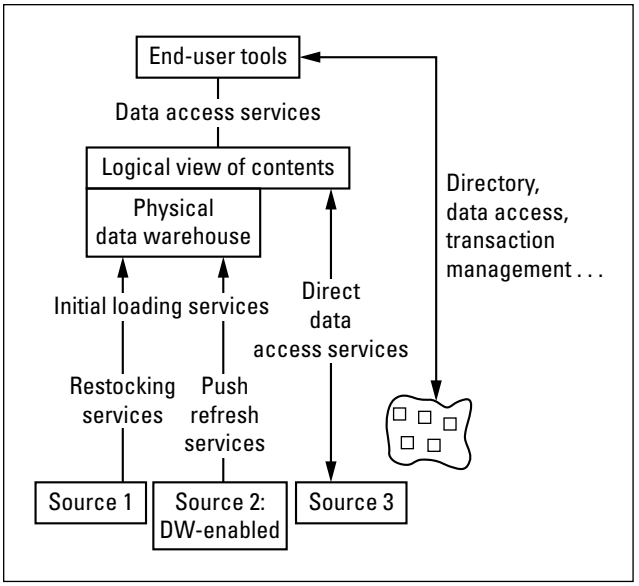


Figure 3-9: Sample architecture from a data warehouse supreme (although it can look like just about anything).

To Centralize or Distribute, That Is the Question

Traditional data warehousing — creating a data warehouse lite or data warehouse deluxe — usually involves copying data from one or more different source databases and files into a single warehouse-owned database. Whether that database is relational or multidimensional is irrelevant. The main point is that it's centralized: Only one database is on one platform that all users access.

Even when you take part of your data warehouse's content and copy that information into one or more data marts for users, as shown in Figure 3-7, you still have a single, centralized collection point into which all the source data converges.

While data warehouse environments become larger and more complex, though, you should consider not funneling all your data into a single database; rather, you could make your data warehouse a collection of databases that make up your overall information delivery and analytical environment. Figure 3-10 shows what a brokerage firm's distributed, non-centralized data warehousing environment might look like.

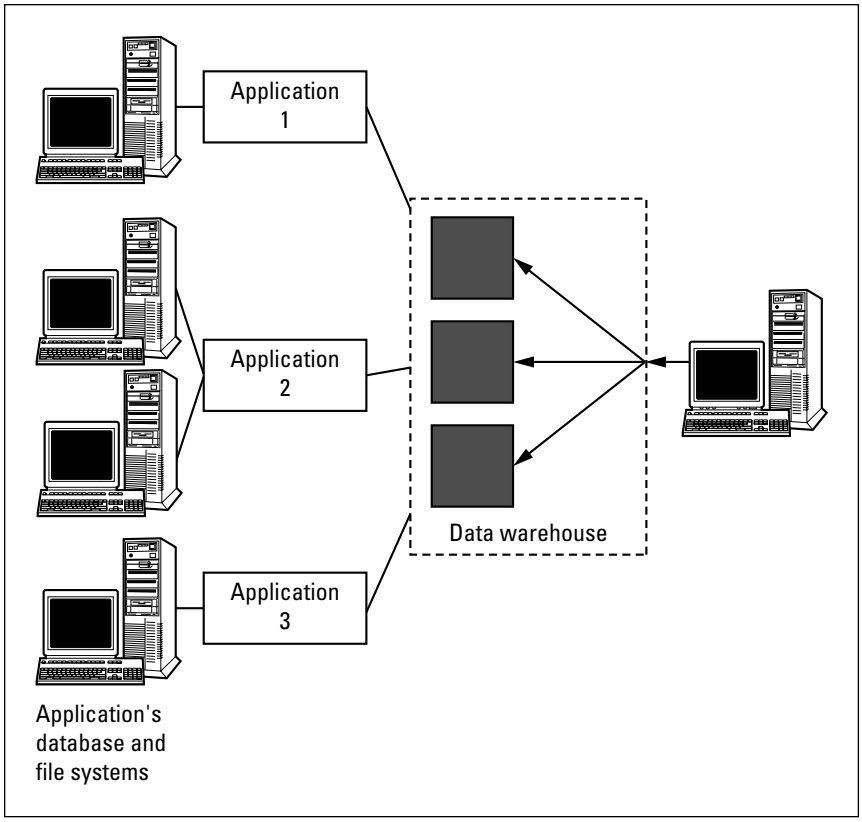


Figure 3-10: A data warehouse might consist of more than one database, under the control of the overall warehousing environment.

Chapter 4

Data Marts: Your Retail Data Outlet

In This Chapter

- Looking at different architectures for data marts
- Determining what should be in your data mart
- Developing a data mart

Several years ago, a regional (I think) hardware chain in southern California ran a radio ad based on this premise: "Shop at our stores because we have fewer products than the big warehouse-like competition. You can much more easily get in and out here quickly with what you need." Interestingly, another hardware chain (on the east coast) had run ads a few years earlier with almost the identical theme. This chain used to make fun of the warehouse-size competition by featuring radio ads that had helicopter search parties looking for shoppers lost in a distant department and references to shuttle buses having to take shoppers between departments in the warehouse-size stores. The premise was the same: "We have less merchandise than the other guy, so shop with us because it's easier."

And that's the idea of the data mart.

Don't get caught up in the hype. The idea of a data mart is hardly revolutionary, despite what you might read on blogs and in the computer trade press, and what you might hear at conferences or seminars. A data mart is simply a scaled-down data warehouse — that's all.

Vendors do their best to define data marts in the context of their products; consultants and analysts usually define data marts in a way that's advantageous to their particular offerings and specialties. That's the way this business goes; be prepared to ask the tough questions.

Architectural Approaches to Data Marts

You can take one of three main approaches to creating a data mart:

- Sourced by a data warehouse (most or all of the data mart's contents come from a data warehouse)
- Quickly developed and created from scratch
- Developed from scratch with an eye toward eventual integration

In the following sections, I examine each approach separately.

Data marts sourced by a data warehouse

Many data warehousing experts would argue (and I'm one of them, in this case) that a true data mart is a "retail outlet," and a data warehouse provides its contents, as shown in Figure 4-1.

In an environment like the one shown in Figure 4-1, the data sources, data warehouse, data mart, and user interact in this way:

1. The data sources, acting as suppliers of raw materials, send data into the data warehouse.

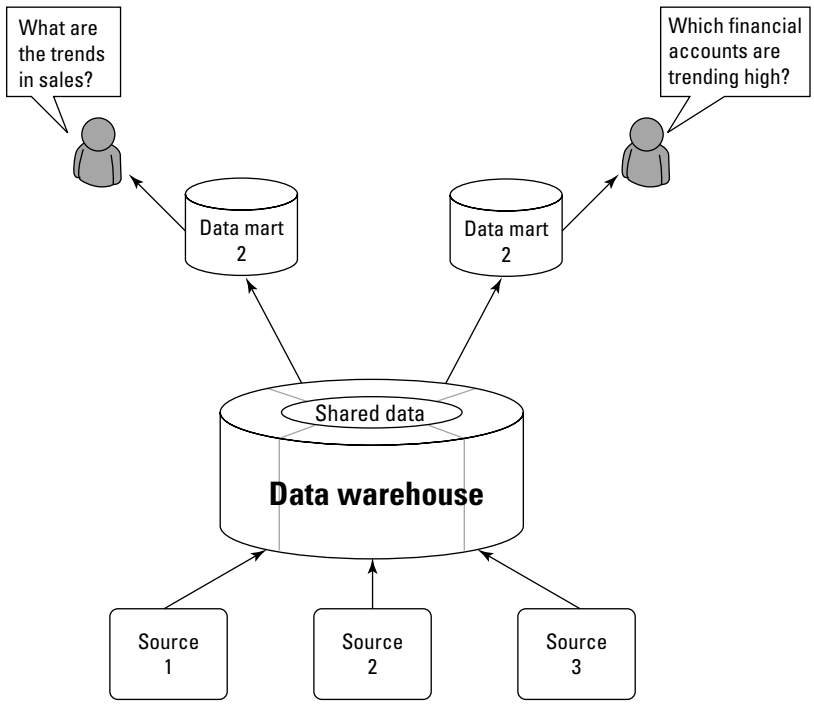


Figure 4-1: The retail-outlet approach to data marts: All the data comes from a data warehouse.

2. The data warehouse serves as a consolidation and distribution center, collecting the raw materials in much the same way that any data warehouse does.
3. Instead of the user (the consumer) going straight to the data warehouse, though, the data warehouse serves as a wholesaler with the premise of "we sell only to retailers, not directly to the public." In this case, the retailers are the data marts.
4. The data marts order data from the warehouse and, after stocking the newly acquired information, make it available to consumers (users).

In a variation of the sourced-from-the-warehouse model, the data warehouse that serves as the source for the data mart doesn't have all the information the data mart's users need. You can solve this problem in one of two ways:

- Supplement the missing information directly into the data warehouse before sending the selected contents to the data mart, as shown in Figure 4-2.
- Don't touch the data warehouse; instead, add the supplemental information to the data mart in addition to what it receives from the data warehouse, as shown in Figure 4-3.

If your data mart is the only one within your company that needs additional data (be sure to ask around), leave the warehouse alone and bring the supplemental data directly into your data mart. If other data marts or other projects served by the data warehouse can use the additional information, add that information to the data warehouse first and then send it, along with the other contents you need, to the appropriate data marts.

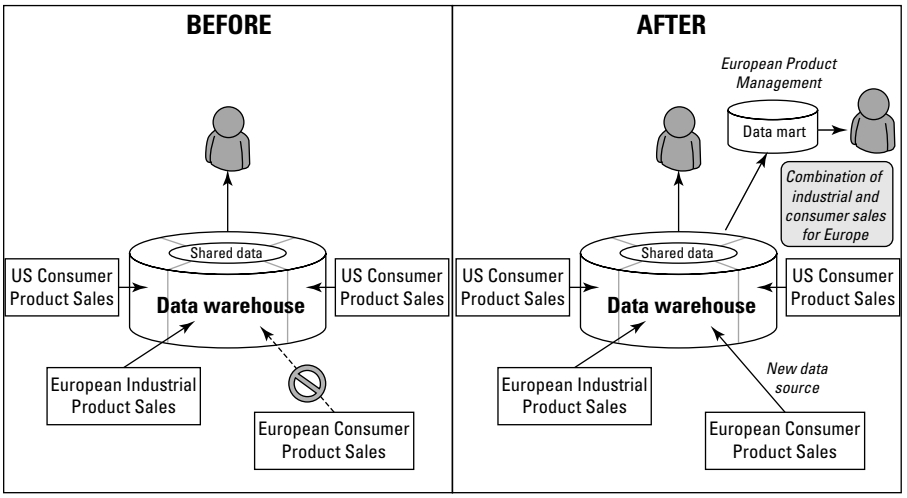


Figure 4-2: Sprucing up the data warehouse's contents as part of a data mart project.

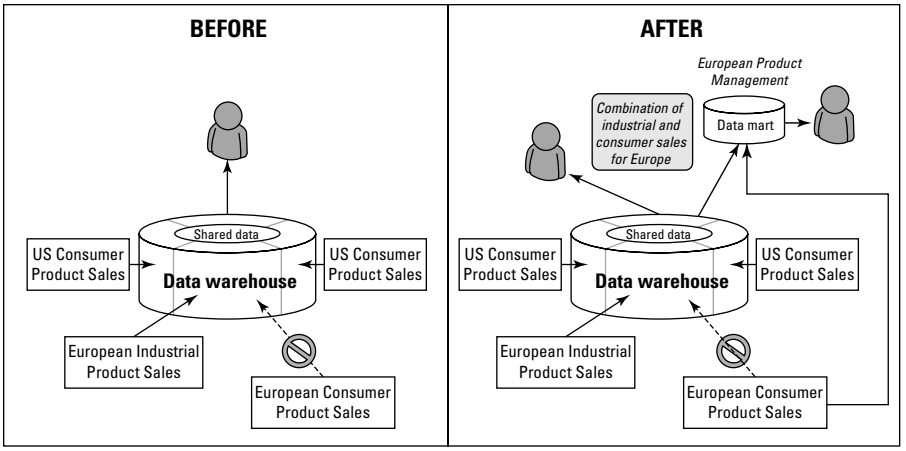


Figure 4-3: Supplement the data mart's contents with additional information acquired directly from the missing sources.

Top-down, quick-strike data marts

Sometimes, you just don't have a data warehouse from which to get data for your data mart, so you have to source the data from applications yourself. In many (probably most) of these situations, you create a *quick-strike data mart* — in effect, a miniature data warehouse but built to meet the demands of a set of users who need the data content now. You follow the same methodology and complete the same processes of data extraction, transformation, quality assurance, and loading as described in Chapter 3. The difference is that you use this methodology on a smaller scale than you do with a full-blown data warehouse.

As shown in Figure 4-4, you often need to bring data into a top-down, quick-strike data mart to answer a specific set of business questions within relatively narrow confines. For example, you can add data about a specific region or territory within a company, a subset of a company's overall product line, or some other subsetting model.

So, if you need to start from scratch and don't have a data warehouse to provide data to your data mart, why not build a full-scale data warehouse instead? Here are three reasons to go the data-mart route:

- **Speed:** A quick-strike data mart is typically completed in 90 to 120 days, rather than the much longer time required for a full-scale data warehouse.
- **Cost:** Doing the job faster means that you spend less money; it's that simple.
- **Complexity and risk:** When you work with less data and fewer sources over a shorter period, you're likely to create a significantly less complex environment — and have fewer associated risks.

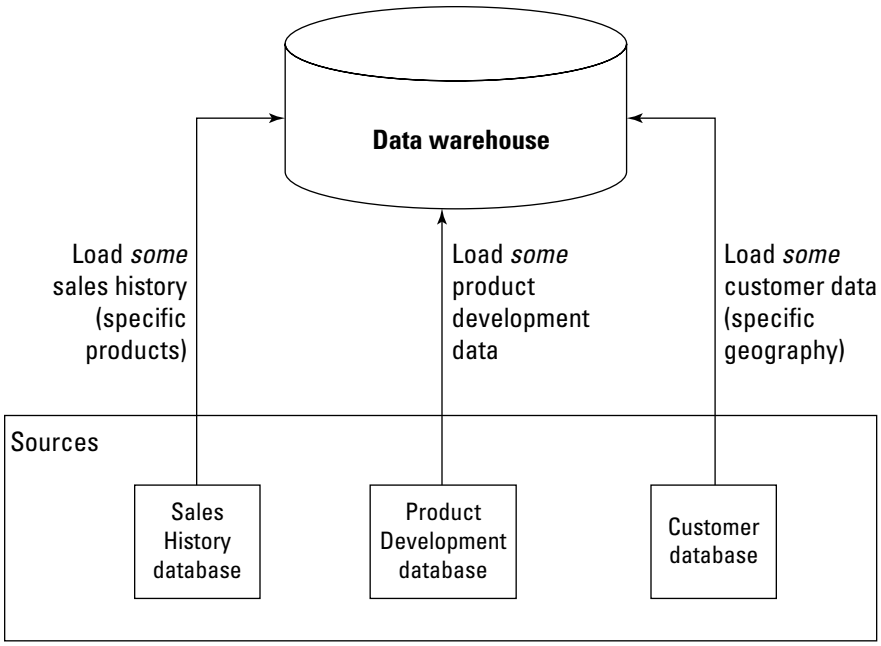


Figure 4-4: A top-down, quick-strike data mart is a subset of what can be built if you pursue full-scale data warehousing instead.

Bottom-up, integration-oriented data marts

If pressing business needs steer you toward a quick-strike data mart but you have a longer-term vision of integrating its contents with other data, what can you do? Have you created an architectural dead end in your data mart? Will you have to throw away your data mart at some point and start over with a "real" data warehousing effort? Will Harry Potter save his mates from Lord Voldemort? (Sorry about that — the numerous releases are getting to me.)

Theoretically, you can design data marts so that they're eventually integrated in a bottom-up manner by building a data warehousing environment (in contrast to a single, monolithic data warehouse).

Bottom-up integration of data marts isn't for the faint-hearted. You can do it, but it's more difficult than creating a top-down, quick-strike data mart that will always remain stand-alone. You might be able to successfully use this approach . . . but you might not.

What to Put in a Data Mart

If a data mart is a smaller-scale version of a data warehouse, this question comes up: What does "smaller scale" mean in reference to the contents of a data mart? The answer to this question is typically that the data will be a subset of the overall *enterprise data*.

The following sections describe some ways that you can select subsets of information for a data mart and the circumstances under which you might want to try each approach.

Geography-bounded data

A data mart might contain only the information relevant to a certain geographical area, such as a region or territory within your company. Figure 4-5 illustrates an example of geography-bounded data.

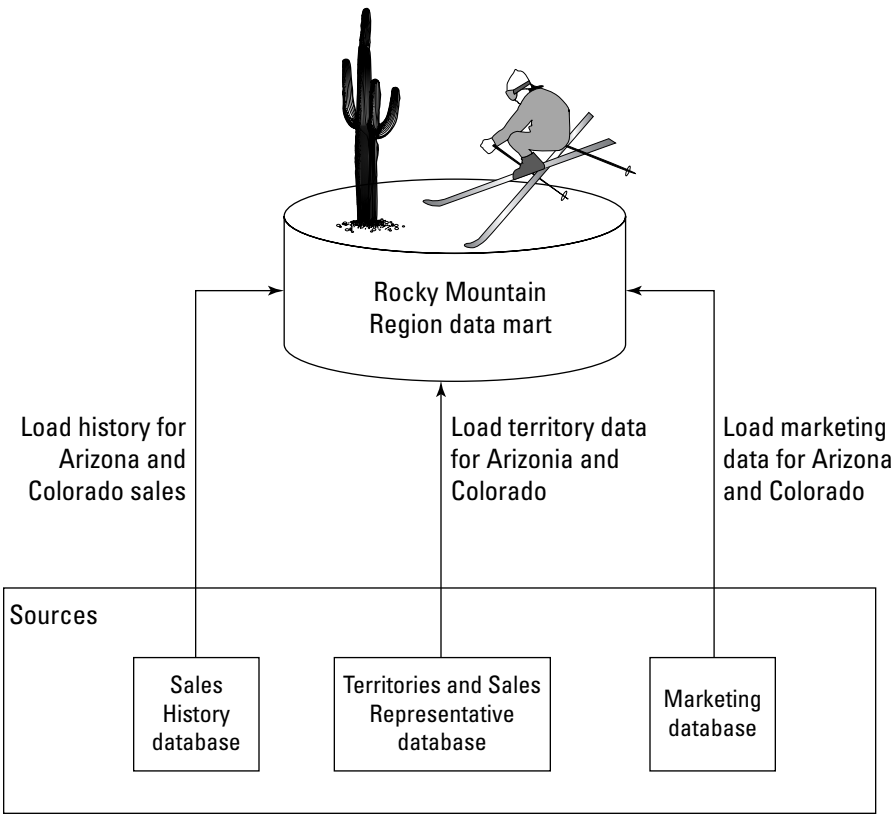


Figure 4-5: A geography-bounded data mart contains only the sales information relevant to a given territory.

Although you technically can use a geography-bounded data mart in a relatively straightforward way, you probably don't want to subset your data in this manner. Users often want to see a cross-geography comparison (for example, "How are our Arizona stores doing versus our Pennsylvania stores?") in their data warehouse environment. When you create separate data marts for various geographical reasons, these types of comparisons become much more difficult to make.

Organization-bounded data

When deciding what you want to put in your data mart, you can base decisions on what information a specific organization needs when it's the sole (or, at least, primary) user of the data mart. As shown in Figure 4-6, a bank might create one data mart for consumer checking-account analysis and another data mart for commercial checking accounts.

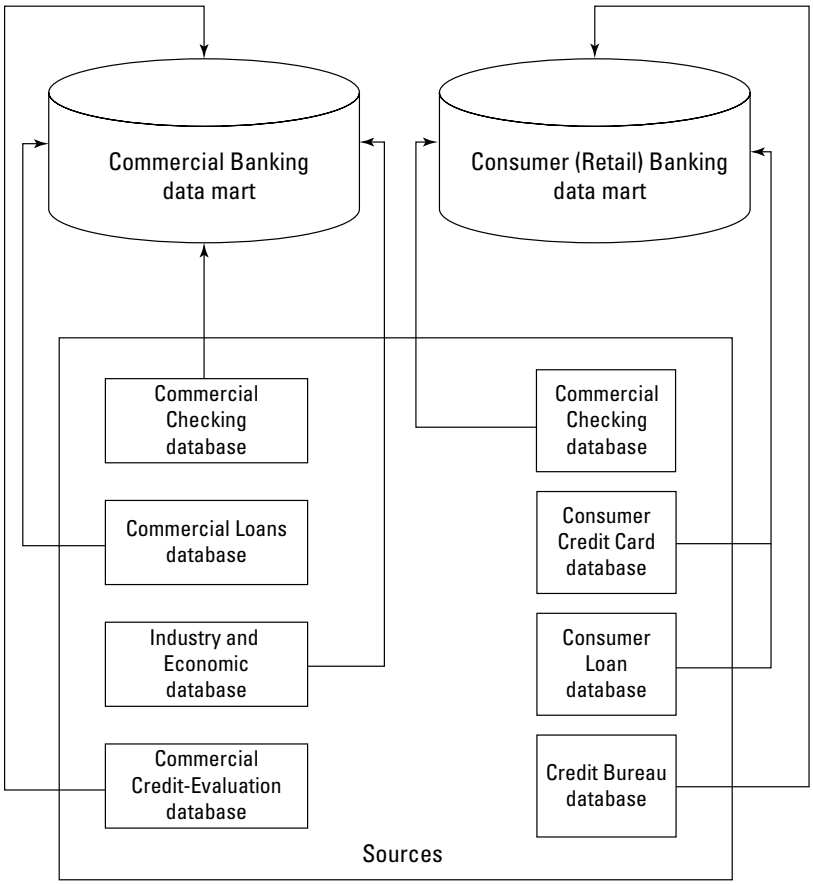


Figure 4-6: Each organization in this bank gets its own data mart, tuned to its specific needs.

This approach works well when the overwhelming majority of inquiries and reports are organization-oriented. For example, the commercial checking group has no need whatsoever to analyze consumer checking accounts and vice versa. It pays to dig into the business needs during the scope phase of a data warehousing or data mart project. Outsiders, for example, might think, "Okay, put all checking-account information, both consumer and commercial, into the same environment so that Marketing or Risk Management Analysts can run reports comparing average balances and other information for the entire checking-account portfolio at the bank." After additional analysis, though, you might notice that the bank doesn't do this type of comparison, so why not keep the two areas separate and avoid unnecessary complexity?

Function-bounded data

Using an approach that crosses organizational boundaries, you can establish a data mart's contents based on a specific function (or set of related functions) within the company. A multinational chemical company, for example, might create a data mart exclusively for the sales and marketing functions across all organizations and across all product lines, as shown in Figure 4-7.

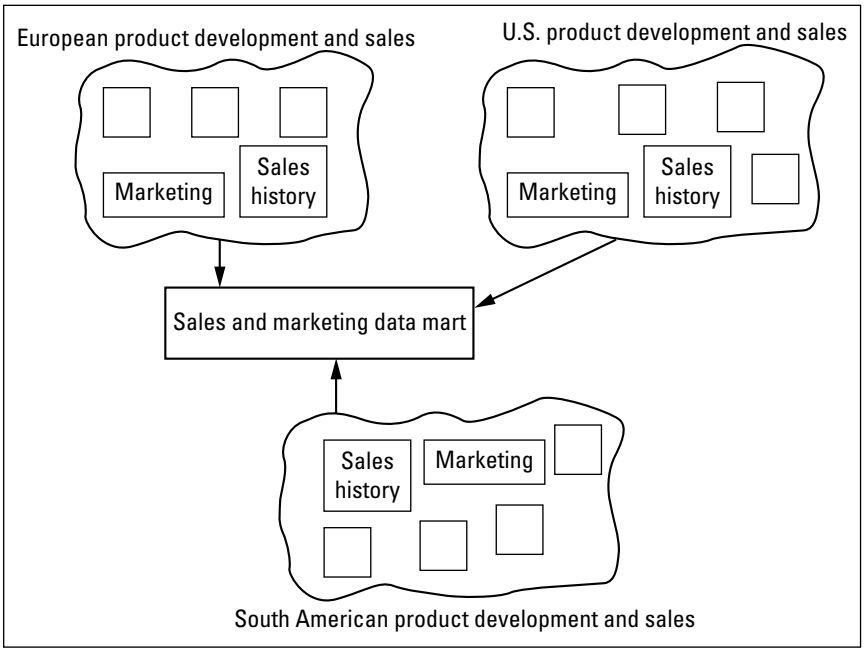


Figure 4-7: If it relates to sales and marketing, it goes into the data mart.

Market-bounded data

A company might occasionally be so focused on a specific market and the associated competitors that it makes sense to create a data mart oriented with that particular focus. As shown in Figure 4-8, this type of environment might include competitive sales, all available public information about the market and competitors (particularly if you can find this information on the Internet), and industry analysts' reports, for example.

To truly provide the business intelligence that a company needs in a competitor-driven situation, construct the data mart to include multimedia information, in addition to the traditional data types typically found in a data warehouse. (Chapter 25 describes multimedia data and data warehousing.)

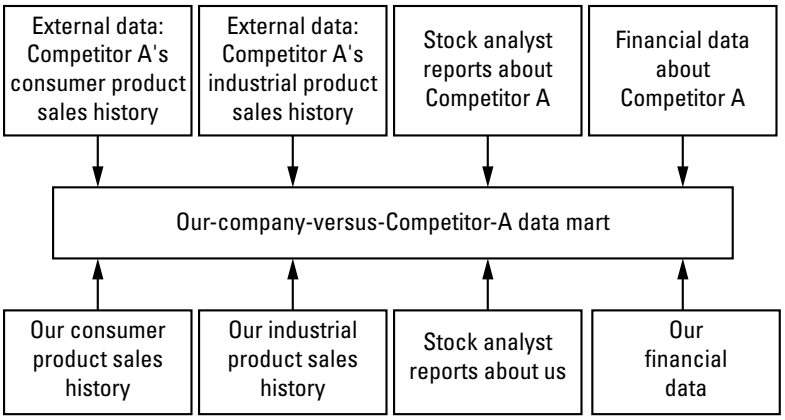


Figure 4-8: The data mart as a weapon of war: Collect information about your market and competitors, and see how you stack up.

Answers to specific business questions

The answers to a select number (often a handful) of business questions occasionally drive an organization's operations. Based on the answers, a company might speed up or slow down production lines, start up extra shifts to increase production or initiate layoffs, or decide whether to acquire other companies.

Business questions that have this degree of weighty importance traditionally cause nightmares for the in-house employees chartered with digging out data and reports, consolidating and checking the information, and reporting the results to executive management. Sounds like a job for a data warehouse, you say? Unfortunately business analysts have often used spreadsheets, such as

Microsoft Excel. These types of "spread marts" often lack the repeatability and data quality required to leverage the data for more than one moment in time. So, I'll say it again — sounds like a job for a data warehouse!

Before constructing a full-scale data warehouse that can answer these (and many other) business questions, however, you probably want to consider whether a small-scale data mart designed specifically to answer those high-impact, high-value "How are we doing?" type of questions can get the job done.

Later, this type of environment might grow into a larger-scale data warehouse. It often makes more sense, however, to concentrate your efforts on supporting a data mart that has known business value, instead of on supplementing it with volumes of additional data that might provide business value (but can also slow response time or significantly complicate the end-to-end architecture). Again, the job you do in the early phases of your project makes a big difference in the direction you take and your level of success.

Anything!

Any set of criteria that you can dream up can determine a data mart's contents. Some make sense; others don't. Some take you into an architectural dead end because you get only limited value and have to start all over to expand your capabilities.

Data mart or data warehouse?

If you start a project from the outset with either of the following premises, you already have two strikes against you:

- "We're building a real data warehouse, not a puny little data mart."
- "We're building a data mart, not a data warehouse."

By labeling your project as one or the other of these terms, you already have some preconceived notions about the work you'll do, before you even begin to dig into the business problem. Until you understand the following three issues, you have no foundation on which to classify your impending project as either a data mart or a data warehouse:

- The volumes and characteristics of data you need
- The business problems you're trying to solve and the questions you're trying to answer
- The business value you expect to gain when your system is successfully built

If you're extracting and re-hosting a subset of data from an existing application into another environment, you can accurately call what you're building a data mart.

But if you're starting from scratch, extracting data from one or more source systems, handling the quality assurance and transformation, and copying that data into a separate environment, what determines whether you're building a data warehouse or a data mart? Although some guidelines exist, such as number of subject areas and volumes of data, it all comes down to this statement: As soon as you start labeling your environment as one or the other, you're adding preconceived notions and beliefs about its characteristics that might not fit your business needs.

Here's the answer: Forget about the terms data warehouse and data mart. Concentrate instead on your business problem and its possible solution. What data do you need in order to perform certain informational and analytical functions; where is that data now and in what form; and what do you have to do to make it available to your users?

Leave the terminology wars to the vendors and analysts. Don't get caught up in the hype.

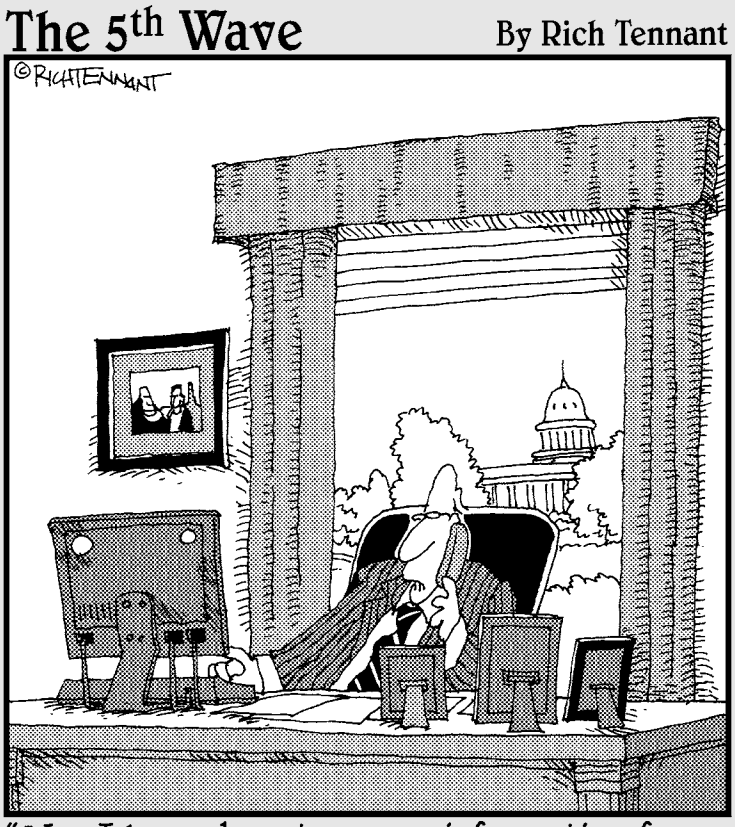
Implementing a Data Mart — Quickly

No matter how you decide to divide the universe of possible contents into some subset for your data mart, remember that in order to obtain maximum business value from your data mart, you must implement it quickly.

Here are the three keys to speedy implementation:

- **Follow an iterative, phased methodology.** As described in Chapter 13, you spend the majority of your upfront time on the project focusing on the specific business value the end user wants and over several iterations build the solution into their vision.
- **Hold to a fixed time for each phase.** If you set aside two weeks for your scope, for example, stick to that window. Don't extend any phase (especially the early ones) unless the project is doomed to failure if you don't.
- **Avoid scope creep at all costs.** Although costly and dangerous in any project (data warehousing or otherwise), *scope creep* (when additional feature requests keep creeping in long past the cutoff point) can devastate a data mart effort. By adding these last-minute features, you probably add complexity to your data mart with only marginal incremental business value (if any), so you do little other than put your project at risk.

Part II
Data Warehousing Technology



"Yes, I know how to query information from the program, but what if I just want to leak it instead?"

In this part . . .

A data warehouse without a database is like a day without sunshine. But what kind of database do you need for your data warehouse?

Whether you're inclined toward making a traditional relational database, a multidimensional database, or a somewhat newfangled column-wise (vertical) database as the home for your warehouse's data, you have to understand the basics of all the options and when each one is appropriate. It's all here!

Chapter 5

Relational Databases and Data Warehousing

In This Chapter

- Using relational databases, past and present
- Making relational databases work well for data warehousing
- Checking out relational database products for your data warehousing project

You don't, strictly speaking, have to store the contents of a data warehouse in a relational database. However, in nearly all situations, your project benefits significantly from the use of a relational database management system (RDBMS).

This chapter explains the use of relational database technology (today's overwhelmingly dominant database technology) for your data warehouse, including its benefits and challenges. Chapter 6 discusses alternative technologies that you can use to store and manage your data warehouse.

The 1990s generation of data warehousing implementations grew up on multidimensional databases, but the current marketplace is experiencing a clear-cut trend toward using relational databases, particularly for large-volume data warehouses (more than about 250 gigabytes of data). Multidimensional databases aren't dead — they can still provide value to smaller-scale environments (data warehouse lite systems or data marts, as discussed in Chapters 3 and 4, respectively).

The Old Way of Thinking

Jump back to 1995. The data warehousing revolution was picking up steam, and companies all over the U.S. (yours was probably one of them) and around the world were captivated by not only the concept of data warehousing, but also the principles of online analytical processing (OLAP — discussed in more detail in Chapters 6 and 10).

In 1995, the world of data warehousing and OLAP widely held the belief that you absolutely couldn't successfully build a data warehouse by storing data in a relational database; you could use only a multidimensional database.

This relational-versus-multidimensional war ranked up there with the all-time champion of polarized technology proponents: the Ethernet-versus-token-ring-LAN (local area network) wars of the mid-1980s. (Proponents of either LAN technology fervently — and sometimes savagely — argued that their approach was far superior to the other and that only fools would attempt to implement a LAN by using the other technology. If you were involved with client/server computing or networking back then, you probably remember the argument well. Relational-versus-multidimensional was a 1990s version of this memorable classic!)

The following section describes how this can't-do-it-with-relational belief got started.

A technology-based discussion: The roots of relational database technology

Relational database management systems (RDBMSs) have their roots in the relational data model developed by Dr. E.F. Codd, then with IBM, back in 1970. (He first described it in his landmark paper "A Relational Model of Data for Large Shared Data Banks," published in *Communications of the ACM, Vol. 13, No. 6,* June 1970, pp. 377–387. You can check out the paper on the Web, at `www.sigmod.org/codd-tribute.html`). Throughout the 1970s, in an attempt to commercialize relational technology, IBM and a few other organizations worked on prototypes, proof-of-concept systems, and early starts at product development.

When you join tables in a relational database, problems start occurring. Back in the early, pre-relational days of databases, data usually was linked together by using pointers, which told the DBMS software the location of the next logical piece of data or the previous piece of data, or some other path that made sense. Because it's somewhat complicated, don't worry about the details — you just need to know that old-style databases usually had pointers intermixed among the data.

Relational databases *don't* have pointers in them because one of the fundamental principles of this relational approach is that you can join together any pieces of information in the database, so pointers don't make sense. (Again, although the principle is somewhat more complicated than that and has to do with set-based operations on the data, just remember that relational databases, unlike their predecessors, don't use pointers.)

What's a relational database management system?

Forget all about the mathematical foundations of the relational model, the principles of normalization, and other highly technical aspects of RDBMSs. If you're interested, consult any one of the many available textbooks that discuss RDBMS principles and technology in detail.

For purposes of this book, an *RDBMS* is a software system that manages relational databases. So, what's a relational database?

In a typical spreadsheet program, columns and rows form a series of cells. If each column is headed by the name of a data attribute (CUSTOMER_NUMBER, PRODUCT, and QUANTITY_PURCHASED, for example) and each row has a single value for each attribute, you have the basics of a relational database table, as shown in this example:

```
CUSTOMER_NUMBER   PRODUCT           QUANTITY_PURCHASED
12345             Vegetable soup    5
45678             Cooking oil       3
42973             Lawn fertilizer   2
81115             Blankets          88
81115             Vegetable soup    33
```

A relational database typically has many different tables — a CUSTOMER_MASTER table and a PRODUCT_MASTER table, for example, in addition to the preceding table, which you could call ORDER_DETAIL or WHO_ORDERED_WHAT. You can combine information from across the various tables by *joining* those tables (making a match between tables, usually by looking for columns in two or more tables that are the same). For example, CUSTOMER_MASTER might contain the following rows of data:

```
CUSTOMER_NUMBER   CUSTOMER_NAME
12345             Mark Jones
45678             Daniel Michaels
42973             Karen Warner
81115             Susan Robinson
```

Joining these two tables, using CUSTOMER_NUMBER as the common attribute, might tell you that customer number 81115, named Susan Robinson, ordered 88 blankets and 33 cans of vegetable soup. (She must be going on a camp-out!) From either of the individual tables, you have only bits and pieces of this information: You don't know the customer's name from looking only at the first table, for example, and you don't know what products Susan Robinson ordered from looking only at the second table. The power of the relational database becomes evident when you join tables together.

Back when databases used pointers, accessing logical sequences of data was fairly quick, even on older hardware. Data access went something like this:

1. Go directly to the first record you want (usually from some type of index).
2. After reading the record, read the pointer attached to it that indicates where to go next.

3. Go directly to the record indicated by that pointer.
4. Read that second record, and then read its pointer.
5. Repeat this process until you reach the end of the list or until some other criteria tells your program to stop.

In a relational database, without pointers, the RDBMS (remember, the RDBMS is a software program that manages the relational database) has to figure out how to *most efficiently* access data and provide the answer you're looking for. Using the example in the sidebar "What's a relational database management system?" in this chapter, you might want to know how many products Susan Robinson ordered, as well as how many of each.

The RDBMS could find this information by scanning the CUSTOMER_MASTER table from top to bottom until it finds the customer named Susan Robinson. (To keep this example simple, assume that no two customers have the same name.) After the RDBMS finds that row of data, it gets the customer number and goes to the WHO_ORDERED_WHAT table. The RDBMS then scans that table, from top to bottom, looking for any row of data where CUSTOMER_NUMBER is equal to 81115 (Susan Robinson's customer number). Unlike in the first table, though, the RDBMS doesn't stop when it finds the first row of data that meets these criteria because the table might have 1, 100, 1,000, or any number of rows indicating what Susan Robinson ordered.

A relational database doesn't store data in any kind of sorted order, though you could use an index to help locate data. In an official sense, the exact physical order in which data is stored is *implementation-defined:* The DBMS product decides how the product works (and it's none of your business), as long as the product does work.

Back to the history of relational databases (this discussion is leading somewhere, I swear!). Vendors spent most of the 1970s (the research era of relational databases) and the 1990s trying to solve a nasty problem: If a relational database doesn't have pointers to efficiently tell the software how to return data to users, could they build dynamic logic to speed the retrieval of data? An RDBMS product has a subsystem, usually known as a *query optimizer* (or a similar term), that determines the best method to retrieve stored data by evaluating many aspects of how the data is organized, such as the ones in this list:

- The number of tables from which you're asking for information as part of the join process
- The size (if it's known) of each of those tables
- Any indices that are available to prevent having to scan each table from end to end
- System characteristics (for example, what type of CPU processors are configured, how much memory is available, and how the data is stored and physically on the disks)

Here's the punch line: RDBMS vendors spent about two decades working on their query optimizers to make relational database technology suitable for online transaction processing (OLTP) applications, such as customer-order processing, checking-account systems, and zillions of other types of applications that companies use to run their businesses. Finally, near the end of the 1980s, the corporate IT world (most of it, anyway) became convinced that you could use RDBMS products for more than just play. Real-world OLTP applications began to use RDBMSs, replacing the old-style databases (the ones with the pointers) or file systems.

And along comes data warehousing. So what's the significance? Here it is, summed up in a few sentences: OLTP applications typically access a small number of tables (preferably one, but usually only two or three) during a given *transaction,* a series of steps that either access data from a database or put data into it. That 20 years' worth of optimization, therefore, was oriented toward making these OLTP-style transactions, which included a small number of tables, as efficient as possible.

Data warehousing applications — the business analytics, or online analytical processing (OLAP) ones, as described in the following section — rarely access one, two, or three tables at a time. Rather, the nature of data warehousing (bringing together a lot of information that, when the pieces of data are related to one another, provides business intelligence capabilities) usually means that a single query must access a large number of tables.

The way you design your relational database plays a large part in the number-of-tables situation you have to deal with.

When RDBMS products that had evolved over two decades to support OLTP-type database access with reasonable performance were suddenly put to work in OLAP environments, they began performing in a somewhat less-than-desirable manner. It wasn't anyone's fault: The products just couldn't do what they hadn't been developed to do — analyze and aggregate large volumes of data distributed across various database tables. So, the philosophy of not using RDBMSs for data warehousing was developed.

To use an analogy, a four-wheel-drive, off-road vehicle and a sports car both have similar characteristics: an engine, four wheels with tires, and a steering wheel, for example. A sports car, however, has been developed to use the basic common framework of an automobile to go fast; in off-road situations out in the wilderness, the driver of a sports car is probably in trouble.

The OLAP-only fallacy

Because the preceding section says that performance in the 1990s was subpar when RDBMSs were used for OLAP purposes, you might think that RDBMSs were therefore deemed unsuitable for data warehousing.

Wrong!

Without going off on a tangent of blaming vendors and their marketing pitches (I do that in Chapter 23), you absolutely can't make an indelible link between OLAP functionality and a data warehouse. Consider the following points:

- OLAP and data warehousing hit the big leagues at the same time.
- You almost always perform OLAP business analysis functionality by using data that you've loaded into a data warehouse.
- You can do much more than business analysis functionality, or OLAP, with a data warehouse.

Part III of this book discusses the area of business intelligence, which is the reason you build a data warehouse. Go ahead — take a peek at the Table of Contents or jump ahead to those chapters. OLAP business analysis capabilities are only one of four different classes of business intelligence.

To make a blanket statement that RDBMS technology is unsuitable for data warehousing means unequivocally that the only thing you want to do with a data warehouse is OLAP processing. Again, that is incorrect. You will find the data warehouse valuable for data quality purpose necessary to integrate your run-the-business applications as well as other business intelligence capabilities, such as data mining and historical consolidated reporting.

The New Way of Thinking

So, now it's the 21st century; relational database vendors have updated their products so they are now optimized for data warehousing.

Fine-tuning databases for data warehousing

Smart people run RDBMS vendors — at least, the vendors still in business after surviving the Great Database Wars of the 1980s and 1990s. These smart people saw what was going on — the growth of data warehousing and the problems the vendors' respective products were having in supporting the type of processing most commonly done after the data warehouse was built — and they figured, "Hey! We have to do something about this situation."

The major problem vendors faced was that, unlike OLTP applications, many data warehousing queries against a database involved database join operations with four to ten tables, and sometimes even more tables. The typical relational query optimizer, when faced with a join operation that involved a large number of tables, would do the programming equivalent of throwing its

hands in the air, shaking its head in confusion, and saying, "I dunno — you tell me!" (I'm oversimplifying a little, but it gets the point across.) The query optimizer would give up and do one large "join the tables in all possible ways and then figure it all out afterward" response (known as a *Cartesian product*). This inefficient means of joining tables caused these types of queries to run slowly.

So, for the first couple decades in the history of RDMSs, the vendors focused on optimizing their products for run-the-business capabilities. Then, the vendors focused for a couple of decades on how to enhance their RDBMS' query optimizers for monitor-the-business capabilities.

Optimizing data access

As described in the preceding section, the challenges that the RDMS vendors had to address involved the method in which users accessed data. Business analysis queries tend to pull two forms of data — descriptive data, often called dimensions; and measurement data, often called facts. The optimization techniques that many RDBMS vendors provide recognized this design technique — which became known as a *star schema* (discussed in the section "Exploring new ways to design a relational-based data warehouse," later in this chapter).

When a database was designed according to the principles of a star schema, the problems that occurred because of the large number of tables involved in a single query were dramatically reduced by using new data access optimizations, such as a *star join* — a different, more efficient way of doing joins when you have a large number of tables involved in a query.

Avoiding scanning unnecessary data

When users performed their business analysis, they commonly ran into problems when they needed to scan large volumes of data. This type of activity occurs when someone asks a question of the data, such as, "How many customers have orders with more than 50 items?" In the run-the-business database design, this information appears in a column on the WHO_ORDERED_WHAT table. However, this type of question forces every row in the table to be read.

Yikes! Just think of a user asking this question of Wal-Mart, whose staff would have to read all the cash register receipts from all their stores worldwide. In general terms, the old way of thinking forced the database to

1. Read a row in the WHO_ORDERED_WHAT table.
2. Ask itself whether the value in QUANTITY_PURCHASED is greater than 50.

3. If the QUANTITY_PURCHASED is greater than 50, return it to the user and move on; if the QUANTITY_PURCHASED isn't greater than 50, skip that row and move on to the next.
4. Repeat this process until it reaches the end of the WHO_ORDERED_WHAT table.

Once again, envision this happening at Wal-Mart! You don't have enough time in the year or processing power to make this data come back to you. So, RDBMS vendors came up with creative indexing schemes to assist in directly finding data that have high *cardinality* — basically, the number of unique values in a database column. Traditional indexing technology supported only low-cardinality indexing. The new way of thinking introduced the concept of high-cardinality indexing.

Handling large data volume

In the early days of data warehousing, most data warehouses were rather small when compared to the current data warehouses, measuring about 50 gigabytes of data or smaller. While technologies mature and IT professionals become more comfortable with data warehousing (and when they become more daring after getting a few successful projects behind them), those IT folks commonly need or want increasingly larger amounts of data in a warehouse. They want not only larger amounts of data, but also increasing levels of detail (not just summarized information).

For years, the DBMS world has had a term that applies to very large databases. The term is *very large database,* or VLDB. How's that for descriptive?

VLDB has become increasingly synonymous with data warehousing. And, almost without fail, VLDB data warehouses have been implemented by using RDBMS technologies. The combination of warehouse-sensitive query optimization (the star joins mentioned in the section "Optimizing data access," earlier in this chapter), new types of indices, and parallel processing capabilities have permitted RDBMS products to deal with VLDB situations much more effectively than they could in the 1990s.

Parallel processing in the RDBMS world is a relatively complex proposition and a vendor-versus-vendor battlefield because of different ways of doing it. The mechanics of parallel processing (not to mention the arguments used by both sides) don't matter much here. It's important, though, to understand that in parallel database processing, a single database table is divided into multiple *partitions* (different segments of the same table, each containing different rows of data). Furthermore, queries against the database table run in parallel against each of the partitions, effectively reducing the time it takes to get an answer to a query because each parallel query is operating against a smaller amount of data than in the non-partitioned entire table.

Designing Your Relational Database for Data Warehouse Usage

The traditional usage of relational databases, to support the run-the-business transaction-processing applications, has meant that you had to follow certain design principles. If you deliberately violated one of those principles, you had to handle your own work-arounds.

The following sections give you the story behind relational database design and data warehousing.

Looking at why traditional relational design techniques don't work well

In their purest sense, relational databases are designed according to the principles of normalization. Without getting into all the mathematical formalities, the following sections explain what normalization means.

Explaining normalization in plain language (or trying to)

Because a relational database is laid out like a table, it can't have any repeating groups of data (more than one telephone number for a customer, for example) within that row. Although you could have columns called PHONE_NUMBER_1 and PHONE_NUMBER_2, those columns are technically different pieces of data, even though they relate to the same concept (a telephone number). The official way to handle repeating groups, known in the world of conceptual data modeling as *multivalued attributes,* is to create a separate table that has a primary-key join column. (In this example, you put the CUSTOMER_ID column in both the master customer table and in the table that has the phone numbers so that you can reconstruct a customer's complete record.)

You go through this seemingly ridiculous, overly complex set of steps to ensure that a relational database is in *first normal form* (meaning it has no repeating groups of data).

You can also design a relational database to include other normal forms. *Second normal form* (which has no partial-key dependencies) and *third normal form* (which has no non-key dependencies) are the mostly commonly used forms in relational database design, though the seldom-used fourth normal form, fifth normal form, and Boyce-Codd normal form are also options. (A database modeling book can give you more information about these terms.)

To get to the point: A relational database is *normalized* (when the rules of normalization are followed or mostly followed) because of the way it is designed for and accessed during typical OLTP functions.

One of the primary purposes of normalization is to prevent *update anomalies,* which occur when you update a column in a given row but don't update the other rows in that column that have the same value. Update anomalies are a little complicated, and you don't really need to understand them unless your focus is on OLTP database modeling. Just remember that normalization is important for update operations — more specifically, for optimizing the run-the-business database design.

A data warehouse almost never permits update operations because data is modified by bulk reloading operations, rather than single in-place updates as part of a transaction. One of the main benefits of normalization-based relational design, therefore, doesn't really apply to a data warehouse.

The side effect of normalization

What if update operations don't really apply to a relational database? One of the side effects (results) of a highly normalized database is that it has many tables in it; to create data warehousing facts of information (sales by quarter by territory, for example), many multi-table join operations must occur.

If a highly normalized database has a number of tables because facts are broken down into multiple tables (primarily to prevent update anomalies), and if a data warehouse doesn't support update operations (in the OLTP sense), why normalize?

Exploring new ways to design a relational-based data warehouse

Over the last ten years or so, the design techniques for data warehousing by using RDBMSs have largely boiled down to three options:

- **Corporate Information Factory (CIF):** A comprehensive architectural framework that houses various data architectures leveraged to support the three styles of applications within most enterprises (run-the-business, integrate-the-business, and monitor-the-business). These data architectures include the operational data store (ODS), data warehouse, and data marts, along with various interfaces for applications and the operational environment. Bill Inmon, Claudia Imhoff, and Ryan Sousa developed the CIF.
- **Star schema:** This method mimics the multidimensional structures of facts and dimensions, discussed in Chapter 6, but it uses RDBMS tables — specifically, fact tables and dimension tables. The star schema design techniques leverage highly denormalized structures. You gleefully throw away the rules of normalization and put data where it makes the most sense, not based on update-oriented restrictions but based on query patterns. Ralph Kimball developed the star schema design technique.

- **Data vault (Common Foundational Integration Modeling Architecture):** This is a data integration architecture that contains a detail-oriented database containing a uniquely linked set of normalized tables that support one or more functional areas of business tables with satellite tables to track historical changes. This hybrid approach encompasses the best of breed between third normal form (3NF) and star schema. The design is flexible, scalable, consistent, and adaptable to the needs of the enterprise. This data model is architected specifically to meet the needs of enterprise data warehouses.

  Inside the data vault model, you can find familiar structures that match traditional definitions of star schema and 3NF that include dimensions, many-to-many linkages, and standard table structures. The differences lie in relationship representations, field structuring, and granular time-based data storage. The modeling techniques built into the data vault have undergone years of design and testing across many different scenarios, giving them a solid foundational approach to data warehousing. Dan Linstedt is the data vault method author, creator, and inventor.

Choose your relational design approach carefully. Although people have a tendency to choose one method over another, the technology and tools that you leverage impact your design approach. In other words, you can't find one "silver bullet" method. Based on your needs and tool selection, one technique might work better than the others. Check out all these approaches: Run benchmarks, but don't just assume that one or the other design approach can efficiently support your enterprise's data warehousing requirements.

Relational Products and Data Warehousing

The following sections discuss some leading relational database products that you might want to use for your data warehouse. Almost all these vendors have, during the past few years, acquired additional products, including OLAP or multidimensional-oriented technology and other RDBMSs to support very diversified platforms and integrate into their product lines and architectures. You might want to keep an eye on the whole picture because a data warehousing environment might well have both relational and multidimensional servers. In such an environment, you need integration that's as seamless as possible.

IBM Data Management family

```
www.ibm.com/software/data/management
```

The IBM Data Management family is an outgrowth of the IBM flagship DB2 relational DBMS product for MVS/ESA mainframes, as well as a number of acquisitions, including Informix. Many organizations that have corporate standards and mandates, such as "Thou shalt do all large database processing on the mainframe," deploy data warehousing by using a version of DB2 for this platform.

Microsoft SQL Server

```
www.microsoft.com/sqlserver
```

Microsoft initially targeted SQL Server technology for many departmental applications in which Microsoft products are dominant. Over the years, Microsoft has added features to enable organizations to expand capacity for data warehousing, changes which have made SQL Server the preferred platform for data marts.

Oracle

```
www.oracle.com
```

Oracle, a leading RDBMS vendor, has been a mainstay in the areas of data warehousing and data marts since mid-1990s. In the early years, Oracle was the alternative to IBM, and it was a dominate factor in the non-mainframe data warehousing and data mart environments. Oracle has continued to innovate its database platform to address needs for data warehousing.

Chapter 6

Specialty Databases and Data Warehousing

In This Chapter

- Looking into multidimensional databases
- Weighing the benefits of specialty data stores
- Using appliances designed for your data warehousing needs
- Checking out some specialty database products

Relational databases have become the steady, general-purpose champion of the data-management world. The straightforward table-row-column structure, not all that different (at least, conceptually) from a basic spreadsheet, is a flexible method by which you can organize data for many different purposes.

That's the good news.

The not-so-good news is that the flexibility comes with a price. Specifically, in some areas of data management (not many, but some), the table-row-column structure, also known as a *horizontal storage manager,* is inefficient and performs poorly — at least, until you enhance (or, in some cases, overhaul) the relational database management system (RDBMS) to handle these out-of-the-ordinary missions.

One of these areas is *multidimensional analysis,* a way of looking at data as facts organized by dimensions. (I cover all this stuff in the section "The idea behind multidimensional databases," later in this chapter.) As Chapter 5 points out, RDBMS products have been augmented with specialized, enhanced multi-table query optimization so that they can handle data more efficiently in a multidimensional manner.

Multidimensional Databases

This isn't the first time in recent history that new types of database products have emerged and overcome RDBMS inefficiencies. Back in the 1980s, a class of applications was identified in which RDBMS products ill-handled the data-management needs (especially the generation of RDBMSs available at that time). These applications all needed user-specified data types that varied among different implementations. For example, computer-aided design/computer-aided manufacturing (CAD/CAM) applications had to be capable of specifying data types that related to product drawings, blueprints, and other related factors. Computer-aided software engineering (CASE) needed data types to represent applications and systems, databases, graphical representations of entities and attributes, process and data flows, and other parts of the application-development process.

What resulted was *object-oriented database management systems (ODBMSs),* which eliminated the table-row-column structures of relational databases and instead introduced the concepts of classes and subclasses (or types and subtypes), objects, properties, methods, and the other parts of object-oriented technology directly into the database engine.

Because RDBMS technology wasn't well-suited to multidimensional analysis, particularly in terms of performance, vendors set out to develop their own structures tuned and optimized for improved performance.

If you track happenings in the database management world, you're probably familiar with the convergence of relational and object-oriented database technology, as mentioned in Chapter 5. RDBMS products are being equipped with object-oriented extensions. Arguably, this approach to handling complex data types (objects) has won out over non-relational products ("pure" ODBMSs), primarily because of the large installed base of relational products and applications running on top of them. Will the same thing happen in the data warehousing world — relational technology overtaking and then overwhelming specialized multidimensional products? Only time will tell.

The idea behind multidimensional databases

Multidimensional databases (MDDBs) throw out the conventions of their relational ancestors and organize data in a manner that's highly conducive to multidimensional analysis. To understand multidimensional databases, therefore, you must first understand the basics of the analytical functions performed with the data stored in them.

Multidimensional analysis is built around a few simple data organization concepts — specifically, facts and dimensions:

- **Facts:** A *fact* is an instance of some particular occurrence or event and the properties of the event all stored in a database. Did you sell a watch to a customer last Friday afternoon? That's a fact. Did your store receive a shipment of 76 class rings yesterday from a particular supplier? That's another fact.
- **Dimensions:** A *dimension* is a key descriptor, an index, by which you can access facts according to the value (or values) you want. For example, you might organize your sales data according to these dimensions: time, customer, and product.

The basics

In these simple examples, you can organize and view your sales data as a three-dimensional array, indexed by the time, customer, and product dimensions:

- In October 2008 (the time dimension), Customer A (the customer dimension) bought class rings (the product dimension) — 79 of them for $8,833.
- In 2007 (the time dimension), Customer A (the customer dimension) bought many different products (the product dimension) — a total of 3,333 units for $55,905 (the facts).

Notice the subtle different between the way the dimensions are used in these two examples. In the first one, the time dimension relates to a month; the customer dimension relates to a specific customer; and the product dimension is for a specific product.

In the second example, however, time is for a year, not a month; customer is still the same (an individual customer); and product is for the entire product line.

Multidimensional analysis supports the notion of *hierarchies* in dimensions. For example, you can organize time in a hierarchy of year➪quarter➪month. You can view facts (or the consolidation of facts) in the database at any one of these levels: by year, quarter, or month.

Similarly, you can organize products in a hierarchy of product family➪product type➪specific products. Class rings might be a product type; "class ring, modern style, onyx stone" might be a specific product. Furthermore, class rings, watches, other rings, and other items all would roll up into the jewelry product family.

Is there a limit to the number of dimensions?

Theoretically (and I do mean theoretically), you can have as many dimensions in your multidimensional model as necessary. The question always exists, however, of whether your multidimensional database product can support them. But here's a more important question — even if a product allows for a certain number of dimensions (15, for example), does it make sense to create a model of that size? You should work closely with your users to determine whether the number of dimensions makes your solution too complex — and therefore limiting the population of users — or improves the ease of use — and therefore expanding the user population.

You can, for example, add geography to the dimension list that contains time, customer, and product so that you can see and organize facts according to sales territories, states, cities, and specific stores.

How should I choose the levels in a hierarchy?

The levels in a hierarchy enable you to perform *drill-down* functionality, as discussed in Chapter 10. And by having multiple levels within a hierarchy, you can quickly get answers to your questions because of the information that has been set up at each of those specified levels, so that information is just waiting for your queries.

Unfortunately, unless you explicitly determine that you want to include a level in a hierarchy, you can't see your facts at that particular level by using a multidimensional database. (You might see them at a higher, more summarized level of detail or at a lower level of detail, but not at *that* level).

Before you create a bunch of levels in each of your dimension hierarchies (just in case you need them), first consider that every one of these predefined levels affects storage requirements for your multidimensional model.

Because multidimensional databases have fairly rigid structures built around the *pre-calculation* of facts (creating and storing aggregates in the database, rather than performing report-time aggregation and calculation), the more dimensions you have and the more levels in each dimension you have, the greater your storage requirements and the longer your build or load times.

Physical database structures in an MDDB

Although nearly all MDDB products are built around the concept of facts, dimensions, and hierarchies, no one has come up with an MDDB standard definition. In the relational world, non-standardization has also been somewhat of a problem, particularly in relation to value-added features, such as constraints and stored procedures. The basic relational table-row-column structure, however, has been fairly easy to export or unload into a flat file of some type and then reload it into another RDBMS product.

Don't forget your spreadsheet!

Though MDDB products typically provide data to an OLAP tool, many users happily use a plain old spreadsheet program as their primary analytical tool. For example, Essbase (an Oracle-Hyperion product), SQL Server Analysis Server (a Microsoft product), and PowerCubes (a Cognos product) all have an add-in interface to Microsoft Excel. You use the interface to make your data requests and then, after receiving data back into your spreadsheet, manipulate it like you manipulate any other data in a spreadsheet.

When you put together a multidimensional-analysis component for your business development environment, don't forget about your spreadsheet users: They might happily receive dimensionally stored data through their spreadsheet product, rather than through the interface that the underlying MDDB product provides.

In the MDDB world, vendors have taken a variety of different approaches to their respective products' physical representations of data. They're all seeking ways to overcome storage and complexity problems caused by large numbers of dimensions (for example, more than 15) and deep levels of hierarchies (for example, 20 levels deep).

MDDB vendors have tried to use, wherever possible, "sparse array-management techniques" to reduce the amount of wasted space in any data model. (If you're a computer science or MIS major, you might recall this subject from a class in data structures. A *sparse array* is an array in which most of the elements have the same value known as the default value — usually 0 or null. A naive implementation of an array might allocate space for the entire array, but in the case where there are few non-default values, this implementation is inefficient. MDDB vendors have leveraged algorithms that determine the sparcity of the array in advance and arrange the data according in blocks.) The vendors have all taken different paths, and portability from one MDDB product to another hasn't exactly been a high priority for these companies.

When you're evaluating products, don't get caught up in worrying about physical storage techniques: Just make sure the logical representations that come with the products (such as the hierarchies, levels, and facts) can meet your business needs. Eliminate products that seem clunky or that have, for example, a hierarchy model that doesn't seem quite right for your data. Then, after you find products that seem to fit your business, kick the tires a little (so to speak) to see how they work inside.

Are multidimensional databases still worth looking at?

If relational technology is absorbing the multidimensional world and quickly becoming capable of handling this area of data management, are MDDBs now a dead-end technology? Not necessarily. This question is much the same as the continuing ROLAP-versus-MOLAP debate. (Check out Chapter 10 for guidelines on the style of OLAPs you should consider under various circumstances.) You might well have specific needs that make MDDBs the sensible answer for your environment.

Horizontal versus Vertical Data Storage Management

Most relational database managers have been built on a *horizontal storage manager,* which places all data in a database by row (or record) when a transaction occurs. A database table is represented as a chain of database pages that contain one or more data rows. A horizontal storage manager provides fast online transaction processing (OLTP) support because most transactions occur in a record format — for example, inserting a general ledger entry or writing a check. However, when a user requests a record, the database page that contains the data is often moved into memory, which for business intelligence applications is highly inefficient.

Several specialty database products have emerged over the years designed to assist and optimize query-centered applications, such as business intelligence. Such products enable you to more readily develop interactive data warehouses. The goal of these column-wise databases is to increase the speed of decision support queries performed against large amounts of data.

When speaking to audiences around the globe, I often asked database administrators in the audience if they would ever place an index on a column that contains a person's area code, a student's grade point average, or a customer's total transactions. The response is a resounding, "No!" They usually give this adamant response for reasons based on technology, such as:

- We index based only on standard, well-known paths (such as name) because of the overhead of indexes.
- The *cardinality,* or unique occurrence of data, would force the database to perform a table scan anyway.

Yet, when you ask users what information they need to fulfill their job responsibilities, they respond with these kinds of requirements:

- See the number of people by area code in my territory so that I can more effectively manage my promotions.
- Identify the top-ranked students in the graduating class so I can arrange the proper interviews.
- Figure out which customers do business with my company and spend between $100,000 and $500,000 annually.

Each of these three requirements characterizes a different user request, yet they all perform similar functions: They're decision support-oriented queries. A need to access data drives user information requirements, but the users' access patterns aren't compatible with most RDBMS indexing strategies. In short, the RDBMS technology gets in the way of the applications' success.

A database table is represented as a chain of database pages that contain one or more data rows, as shown in Figure 6-1. A horizontal storage manager provides fast online transaction processing (OLTP) support because most transactions occur in a record format.

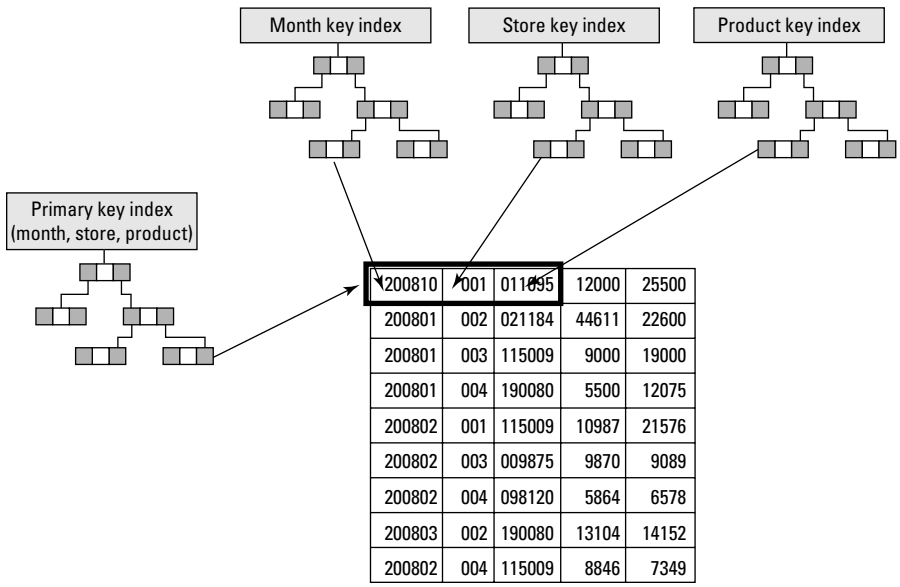


Figure 6-1: Traditional relational databases store data as transactional rows and overlay these rows with indexes to simplify data retrieval.

These relational databases assist query activity by using indexes. Indexes are built on top of the rows to simplify and accelerate data retrieval on common paths, as show in Figure 6-1. Data warehousing solutions, such as business intelligence, don't use many of these indexing techniques because they have been crafted to assist OLTP applications in the frequent need to find and update individual rows within database tables.

To properly support typical user queries found in business intelligence, other storage and indexing techniques are required. Vendors such as Sybase and Vertica have built vertical storage managers. Instead of storing data by row, these products store the data by columns — hence the name *vertical storage manager* or column-wise storage. This method of storage effectively solves the problem of user queries against large sets of data because a user often seeks only a few columns, versus the large number of columns managed in a row by a horizontal storage manager. With the data stored as a series of page changes, with each page containing column data, query processing time is reduced by a factor of 500 or more to 1. Additionally, these products support other optimizations that assist in the speed of query processing, including data compression, parallel query operations, and multiple indexing techniques per column. The challenge in implementing such technologies involves standardization. Therefore, many data management departments refuse to implement such technologies because those technologies require additional support labor.

Data Warehouse Appliances

Like with bell-bottom jeans, hardware-assisted databases are on the comeback trail. (Actually, my daughter tells me that bell-bottoms are out of style again, but hardware-assisted databases aren't.) Microsoft, Oracle, and Netezza are all the rage at database seminars around the globe. In the mid- to late 1980s, vendors Britton Lee and Teradata (which eventually merged) were all the rage. They provided dedicated machines that optimized database processing — the first machines used by heavy data consumers, including many of the consumer-packaged goods companies. The objective of these boxes was to dedicate all aspects of a computer to getting data to the users faster. This dedicated machine included a query-centered database, memory, CPU, and disk operations. Eventually, such products moved out of vogue, and the database management systems were migrated to a more open, run-on-any-box architecture.

Now, they're back!

A *data warehouse appliance* is an integrated set of servers, storage, operating system, DBMS, and software specifically pre-installed and pre-optimized for data warehousing. Data warehouse appliances provide solutions for the mid- to large-volume data warehouse market, offering low-cost performance on data volumes in the terabyte to petabyte range (that's a lot of data!).

Most data warehouse appliance vendors use massively parallel processing (MPP) architectures to provide high query performance and platform scalability. MPP architectures consist of independent processors or servers executing in parallel. Most MPP architectures implement a *shared nothing architecture,* in which each server is self-sufficient and controls its own

memory and disk. Shared nothing architectures have a proven record for high scalability and little contention. Most data warehouse appliances distribute data onto dedicated disk storage units connected to each server in the appliance. This distribution allows the appliances to resolve a relational query by scanning data on each server in parallel. The divide-and-conquer approach delivers high performance and scales linearly when you add new servers into the architecture.

And, from a price perspective, most of the vendors in this arena are attempting a strategy of plug-and-play. For instance, Netezza typically sells a new user their product as plug-compatible with Teradata for less than the Teradata maintenance cost. This price point makes the products very attractive, giving them a growing adoption rate.

Data Warehousing Specialty Database Products

The following sections discuss vendors that have data warehousing specialty database products that you might want to check out.

Cognos (An IBM company)

```
www.cognos.com
```

Cognos was one of the early vendors to offer a multidimensional database that had PowerPlay. The product has a modeling environment (Transformer) that produces physical MDDBs known as PowerCubes. Additionally, Cognos acquired TM1 in the mid-2000s. TM1 is one of the innovation leaders in the area of in-memory MDDBs.

Microsoft

```
www.microsoft.com
```

Microsoft introduced its first MDDB, SQL Server Analysis Services, with SQL2000. The product has gone through a number of innovations, and its scale and capabilities have expanded in the latest release, SQL2008. Additionally, Microsoft recently acquired DATallegro, a database appliance, and will begin providing customers and partners early access to the combined solution through community technology previews in 2009. Full product availability is scheduled for the first half of calendar year 2010.

Oracle

```
www.oracle.com
```

Oracle offers a plethora of specialty database stores, including a new database appliance introduced in 2008 — multidimensional databases Express and Essbase. Oracle also has other index-based innovations within their traditional relational database platform.

Sybase IQ

```
www.sybase.com
```

Sybase introduced IQ in the mid-1990s, a product built on bit-mapped indexing technology. Sybase IQ is a column-wise, vertical storage manager that supports bit-mapped indexing for both low-cardinality and high-cardinality data. For example, a particular attribute that has a small range of values (the colors of an automobile, for example) is low-cardinality data; high-cardinality data consists of range-oriented values (for example, sales volumes and unit sales).

Vertica

```
www.vertica.com
```

Vertica is the commercial release of a Michael Stonebraker database product which grew in the university research environment known as C-Store. (Stonebreaker was the father of Ingres, PostgreSQL, and other successful database products.) Vertica is relatively new to the market, founded in 2005, and consists of a scalable, vertical storage manager focused on analytical applications such as data warehouses.

Chapter 7

Stuck in the Middle with You: Data Warehousing Middleware

In This Chapter

- Defining middleware
- Discovering middleware services and functionality
- Checking out some middleware products

In the world of *distributed computing* — basically, any environment in which data has to move from one system to another — middleware is a key component in making it all work. This chapter discusses data warehousing middleware in detail, from the basics to the specific services that apply to your data warehousing environment. At the end of the chapter, you can find a list of vendors that have middleware products you might want to use in your data warehousing project.

What Is Middleware?

Loosely defined, *middleware* is a set of services that perform various functions in a distributed computing environment, across a wide set of server and client systems. In essence, middleware is computer software that connects software components. Here are some types of middleware services:

- **Security:** Authenticates a particular client program to some system component to verify, for example, that the client program and its user are really who they say they are.
- **Transaction management:** Ensures *transactional integrity* — that a system or database doesn't get corrupted if problems occur.
- **Message queues:** Enables loosely coupled systems to pass messages back and forth to each other, and those messages trigger actions and/or transactions to occur. Messages sent from one application to another are collected and stored until they're acted on, while the application continues with other processing.

- **Application server:** A server that hosts an application programming interface (API), which exposes business logic and business processes so that other applications, either on the same or different servers, can use the shared logic and processes.
- **Web server:** A computer program that's responsible for accepting requests from Web browsers, as well as sending responses and content to those browsers — usually Web pages, such as HTML documents, and linked objects, such as images.
- **Directory:** Enables a client program to find other services or servers located in a distributed enterprise.

These types of services are typically part of a distributed transaction-processing environment. I don't mean that a data warehousing environment can't also include these services; it's just that other middleware services are more important to a data warehousing environment, as described in the following section.

Middleware for Data Warehousing

In a data warehousing environment, the middleware services are the set of programs and routines that do the following:

- Pull data from the source (or sources).
- Make sure that the data's correct.
- Move the data around the environment from platform to platform, as necessary.
- Handle any necessary data transformations.
- Load the data into the data warehouse's database (or databases).

The services

In a more formal sense, the items in the preceding list are handled by these middleware services (described in more depth later in this chapter):

- Data selection and extraction
- Data quality assurance, part I (at the component level)
- Data movement, part I (also at the component level)
- Data mapping and transformation
- Data quality assurance, part II (after transformation has occurred)

- Data movement, part II (into the data warehouse's platform environment)
- Data loading (into the data warehouse)

Figure 7-1 illustrates how these middleware services flow together in a moderately sized data warehousing environment.

But your data warehousing environment might differ from Figure 7-1, particularly in the area of the data-movement services. A data-movement service is necessary every time data crosses system boundaries. Your conceptual picture differs, depending on the details of your particular end-to-end environment.

You absolutely, positively need to plan, design, and otherwise think about data warehouse middleware in terms of the individual services in the preceding list (and described in the section "What Each Middleware Service Does for You," later in this chapter), rather than in generic terms, such as "extraction tools." Many different vendors provide some, many, or all these services as part of a single product or a suite of products. But a tool that has strong mapping and transformation services, for example, might be weak in data-loading services, or a tool that provides a rich set of extraction services might be less effective in the mapping and transformation space, in addition to data quality assurance.

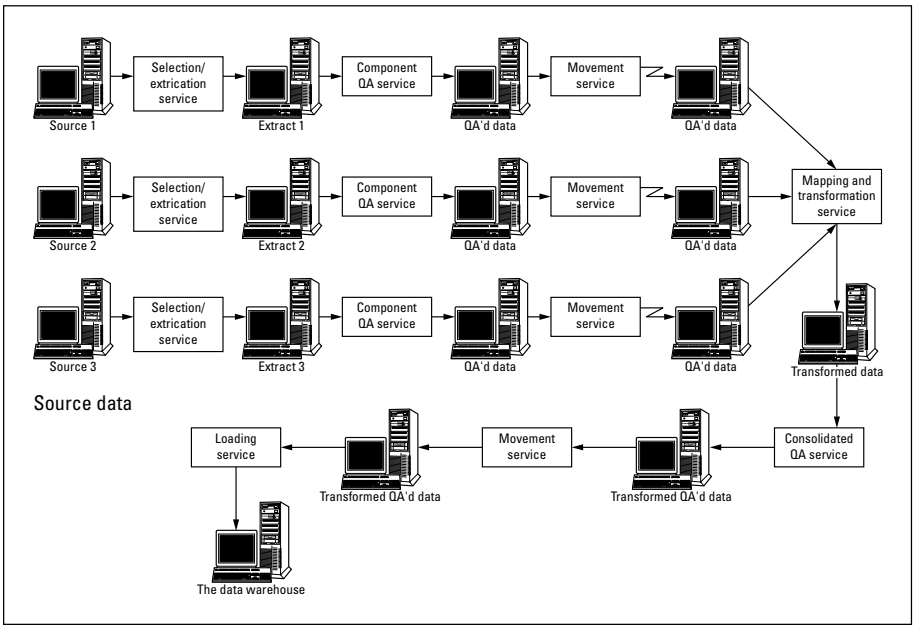


Figure 7-1: The data warehousing middleware services flow together, from end to end, from data sources to the data warehouse.

Before selecting a tool for your data warehousing project (if that's the route you take, rather than custom coding, which I discuss in the following section), make sure that you have a good idea about the particular challenges in your environment. If you have relatively straightforward data-extraction needs, for example, but challenging data-quality problems, concentrate on finding the best quality-assurance tool available, even if it has only so-so extraction capabilities. (This advice applies even if the tool has no extraction capabilities, in which case, you have to combine it with another tool.)

Should you use tools or custom code?

In the early days of data warehousing, most organizations handled middleware services through custom coding, rather than with the few tools available at the time, as shown in this example:

1. An organization writes a program in a programming language such as COBOL, or perhaps in an environment such as SAS, to handle the data extracts from a mainframe data source and then do the quality assurance checking and the transformation.
2. A file-transfer service, such as standard FTP (File Transfer Protocol), is used to copy the transformed and "cleansed" data to the machine on which the data warehouse will reside.
3. Plain old SQL, or a bulk loading utility, is used to load a relational database with the new (or updated) contents of the data warehouse.

Nothing's wrong with this programmatic approach. Always determine for your specific environment whether custom coding or tools are the "right" way to go. Don't automatically assume that you should implement your data warehouse by using middleware tools. But your team might find replicating the reusable logic built into most middleware tools very expensive. And, furthermore, you can find open-source (free) middleware tools available, making the argument of "we'll save money by using internal resources" a difficult one to justify. Therefore, most implementations today are done using *Extract, Transform,* and *Load* (ETL) tools.

What Each Middleware Service Does for You

The following sections describe each of the data warehousing middleware services and what they mean to your data warehousing environment.

Data selection and extractions

The primary purpose of the data-selection and -extraction service is to *select* from (find in) a data source the data that you want to move into the data warehouse and then *extract* (pull out) that data into a form that can be readied for quality assurance services.

You can use one of two different types of selection and extraction services for your data warehousing environment:

- **Get 'em all and sort 'em out later:** Find and extract all the data elements in a source that you want to load into your data warehouse, regardless of whether a specific element has been previously extracted.
- **Change-oriented:** Find and extract only the data elements that have been either newly added to the data source or updated since the last extraction.

The first type of service requires less complex logic in order to perform the extraction. But you have to deal with larger volumes (sometimes, much larger volumes) of data than with the second type, the change-oriented service.

The change-oriented method of selection and extraction is fairly straightforward when your source is a relational database that has a time stamp you can use to detect when a row of data was added or last updated. You can compare a row of data against the date and time of the last extraction process to determine whether data needs to be selected and extracted. But when the data is stored in a file that doesn't have a time stamp (a VSAM file, for example), this process can be significantly more difficult.

You might also face a challenge when source data has been deleted from either a file or a database. If the business rules for your data warehousing environment call for the deletion of corresponding data from the warehouse, you must have a way to detect deletions that were made since the last extraction process to ensure that appropriate deletions are made in your warehouse.

The result of the selection and extraction is, well, an extract of data that's ready to undergo additional processing: checking out the data quality (covered in the following section).

Data quality assurance, part 1

You should establish two different quality assurance (QA) services in the flow of middleware services. You have to perform the first QA tasks against the extract from the data source before you perform any more middleware

services. Try to catch (and correct) errors and problems as early in the process as possible. Moving data down the pipeline toward the data warehouse is pointless if problems are so significant that they either require significantly more effort to correct later in the process or simply can't be corrected.

So, what types of problems should you look for? Here are a few:

- **Values in data elements that exceed a reasonable range:** A customer has submitted 150 million purchase orders in the past month, for example, or an employee has worked with the company for 4,297 years, according to the employee database and the stored hiring date.
- **Values in data elements that don't fit the official and complete list of permissible values:** A value might have an A code, for example, when the only permissible values for that field are M and F. (If that field were labeled GENDER, A might stand for androgynous!)
- **Cross-table inconsistencies:** For entries in the CUSTOMER_ORDER table, no corresponding entries (as identified by CUSTOMER_ID) exist in the CUSTOMER_MASTER_TABLE.
- **Cross-field inconsistencies:** Records that have an incorrect state or zip code for the city indicated.
- **Missing values:** Records that have missing values in certain fields where they should have contents.
- **Data gaps:** For example, a source table should contain one row of data that includes total units sold and sales dollars for each month over the past two years. For a large number of customers, however, no rows exist for at least one of those months.
- **Incomplete data:** If information about every product the company sells is supposed to be available, for example, are all products included in the extract?
- **Violations of business rules:** If a business rule states that only one wholesaler can sell products to any one of the company's customers, you should check to see whether any customer records indicate sales made through more than one wholesaler, which could indicate incorrect data in the source.
- **Data corruption since the last extract:** If extraction occurs monthly, for example, you should keep track of data values or sums that should be constant, such as SALES PER CUSTOMER PER MONTH. If, in a subsequent month, the value of SALES PER CUSTOMER PER MONTH changes for a given customer for a previous month, the underlying data might have been corrupted.
- **Spelling inconsistencies:** A customer's name is spelled several different ways, for example.

What do you do when you find problems? You can try one of the following techniques:

- **Apply an automatic-correction rule.** When you find an inconsistent spelling, for example, do a lookup in a master table of previous spelling corrections and automatically make the change in the data.
- **Set aside the record for a team member to analyze and correct later.** In this case, you might do the human part of the QA in conjunction with automatic correction. For example, automatic corrections are made, if possible, and a report about other problems are put into a separate file and sent to the QA person. When the QA person makes all the manual corrections, you merge the corrections back into the data that has gone through the automatic QA process.
- **Cool your jets.** If you discover enough problems that are serious or require an indeterminate amount of research, consider halting the entire process until after you find and fix the problem.

You can make the QA process much more efficient, and much less problematic, if you perform a thorough source systems analysis, as described in Chapter 16. If you have a fairly good idea about what types of data problems you might find in each data source, you can reprogram your QA process to detect and (hopefully) correct those problems before continuing.

Historically, organizations treated the data warehouse QA process as a one-directional flow. Problems are corrected before the data is moved further into the flow of middleware processes but is never corrected in the data sources. Most new data warehouses have a built-in feedback loop from the QA process that corrects data quality issues in the source data.

Data movement, part 1

In most situations, the two services I describe in the preceding sections (selection and extraction, and quality assurance) take place on the same *platform* (system) on which the data source resides. If your data warehouse will be hosted on a different platform than the data source, though, you have to use a data-movement service to effect the system-to-system transfer of the data.

You can likely use a relatively simple service (handled by a simple file-transfer program, for example). The movement service, if you need it at this point, simply moves the QA'd data into the environment in which you plan to make additional transformations.

Data mapping and transformation

Figure 7-1 shows an environment in which data is being extracted from three different data sources for inclusion in a data warehouse, and each of the three sources is on a different platform. At some point in the middleware process, these QA'd extracts must be brought together for a combined mapping and transformation process.

The mapping and transformation service handles classical data warehousing problems. Suppose that one data source stores customers by using a five-character customer ID, and another source uses a six-digit numeric customer identifier. To enable comparisons and other data warehouse processing, you need a common method of customer identification: One of the identification schemes must be converted to the other, or perhaps a third, neutral identification system, depending on the environment's characteristics.

In addition to handling cross-system incompatibilities, additional transformations might include

- **Data summary:** A summary can be performed earlier in the process, before cross-system movement, depending on the peculiarities of your specific data warehousing environment.
- **Selective inclusion of data:** You might include records from only one data source, for example, if you get a comparable record from another extract. You don't know, until you converge all the data source's contributions, how selective inclusion rules are applied.
- **Data convergence:** Certain elements from one data source are combined with elements from another source to create one unified record for each customer, product, contract, or whatever type of data you're dealing with.

The main point to remember about the mapping and transformation service is that you should have, at its conclusion, a unified set of data that's ready to load into the data warehouse — as soon as you complete a few more steps.

In complex data warehousing environments, you might want to consider multiple transformation processes. As shown in Figure 7-2, for example, data extracts converge at several different levels of transformation before moving farther down the middleware pipeline, enabling you to apply more horsepower to the transformation process by using multiple servers early in the flow.

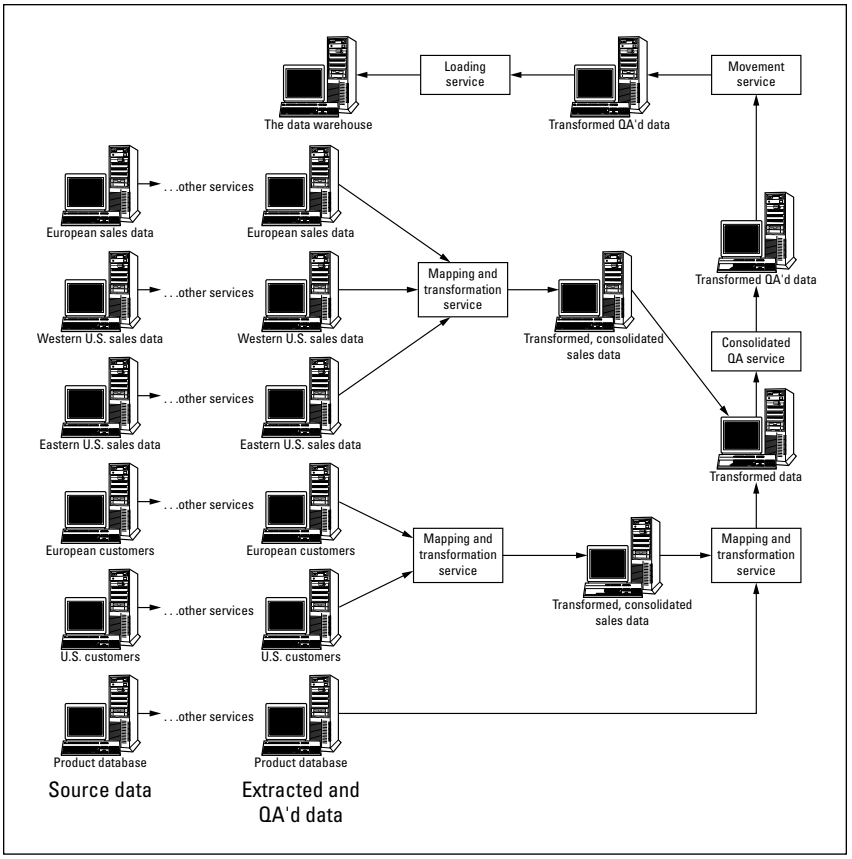


Figure 7-2: Complex data warehousing environments might have several different mapping and transformation points.

Data quality assurance, part II

Following completion of the transformation processes, data must be QA'd — again. You never know what type of errors or discrepancies the transformation process might have introduced into the data. After changes have occurred, any previous QA processes are no longer valid.

Run the consolidated, transformed data through the same type of QA steps discussed in the section "Data quality assurance, part I," earlier in this chapter. Although you probably don't find as many rudimentary errors (such as spelling mistakes or values that are out of range) if you did a thorough job on your first-level QA, you still want to make sure. Furthermore, ensure that the code or scripts used for data transformation didn't accidentally cause new errors to creep in.

The goal of this second-level QA is to make sure that your consolidated and transformed data is ready to load into the data warehouse — as soon as one more step occurs, if necessary.

Data movement, part II

If you're doing your transformation and QA processing on a platform that's different from the platform on which you run your production data warehouse (on a development server, for example, rather than on the operational server), you must execute one more data-movement service to get the data to the place where you want it to eventually reside. This process usually involves only a relatively simple file transfer.

Data loading

The data-loading service loads the extracted, QA'd, transformed, and re-QA'd data into your warehouse. You might load data via a customized program, SQL (an INSERT statement, for example), or a utility.

If you need to load a large volume of data, try to use a fast-loading utility, which usually involves much less time than a programmatic or SQL-based approach.

If you use SQL to load your data into a relational database, try to make the loading as efficient as possible by turning off logging (if your DBMS product permits it). If the loading job is abnormally terminated, you just have to use the DROP or TRUNCATE statement to get rid of your partially loaded table, fix the problem that caused the termination, and restart the job. This process usually is much faster than if you turn on the facilities needed for OLTP-style data and transaction integrity (with accompanying overhead).

Specialty Middleware Services

For more sophisticated and/or simplified data warehousing needs, you can deploy a set of specialty middleware services. These services include

- Replication and change data capture
- Virtual, or federated, data access (also known as Enterprise Information Integration, or EII)

The following sections describe each of these specialty middleware services and what they mean to your data warehousing environment.

Replication services for data warehousing

Replication middleware services combine selection and extraction, movement, and loading from one database to one or more others, usually managed by a single DBMS product. (The source database and all the targets are all Oracle, all Sybase, or all Microsoft SQL Server, for example.)

Although replication service capabilities vary among DBMS products, traditionally, they've been snapshot-oriented: A snapshot of either an entire database or the changes since the last replication occurred are extracted, at a predetermined time, from the source and copied over a networked environment to the intended targets. The data is then transmitted and loaded as-is (no transformation occurs). Many database vendors have implemented their replication by reading the log files for changes. This style of replication is very efficient because it doesn't increase the overhead of your transactional systems to replicate the data.

But replication doesn't replace the long list of data warehousing middleware services discussed in this chapter. You want to use replication in a data warehousing environment primarily when capturing changes in the source database, often called *change data capture* (CDC), or after you load the data into your data warehouse and then extract data and send it to data marts, as shown in Figure 7-3.

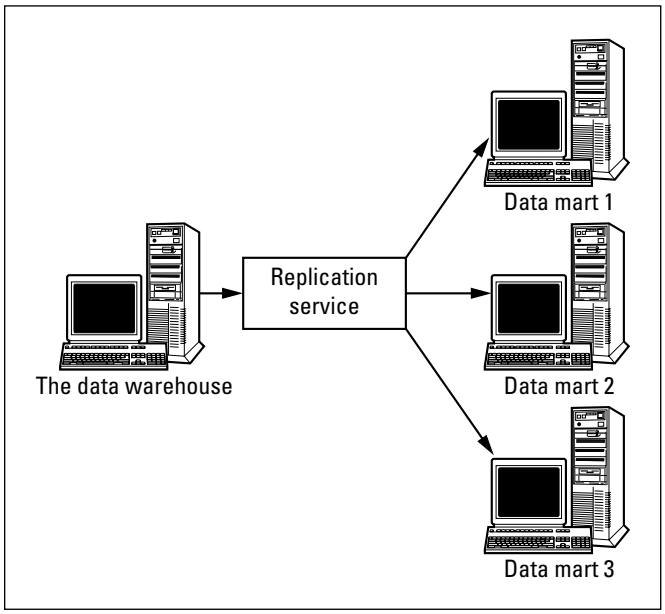


Figure 7-3: RDBMS-based replication services capture changes in the source system to optimize the data selection and extraction process.

Enterprise Information Integration services

Around 1995, vendors began positioning their software as virtual data warehousing tools. The fundamental premise was that sometimes it just doesn't make sense to copy and manipulate a bunch of data, just in case someone needs it. Why not access data directly from the source on an as-needed basis?

Alas, accessing data over a network at its source has proved to be the least challenging of the problems in trying to provide a kind of in-place data warehousing. The same challenges faced in any data warehousing environment (such as dealing with data quality, deciding what types of transformations must occur, and choosing how to handle those transformations when different sources are inconsistent) are still present. Just because you can get to data at its source (in almost any database or file structure) doesn't mean that data provides the necessary business intelligence when it's in your hands.

To solve these data quality issues, many data architects have begun to perform bottom-up data mart construction to develop a component-based data warehouse. Rather than have a single database into which you feed all data (creating your data warehouse), a series of components each handles a particular set of functions (such as answering specific business questions) or certain subjects. Together, these data marts (or components) comprise a data warehousing environment.

This component-based, dynamic access data architecture is the basis for virtual data warehousing and, more specifically, what *Enterprise Information Integration* (EII) servers are offering to the market.

EII architecture

Figure 7-4 shows an environment in which individual components are created within the data warehousing environment in a bottom-up manner. Instead of combining the components into one large database (and copying all the data again), EII creates a data warehousing environment in which users can access each component's contents from a business intelligence tool like they were all stored together, even though they're not.

Think about how you use a Web browser on your desktop. You either click a link or type a specific URL, and the environment, working behind the scenes, takes you to the right place for the content you asked for. Now, imagine the Internet running much faster. When you go to various sites, you're not accessing ads for the latest four-wheel drive you've been coveting, sports scores, Dilbert cartoons, or whatever else it is you do on the Internet. You're bringing back pieces of data that are then combined and sent back to your browser. That's virtual data warehousing — it's just like the Internet!

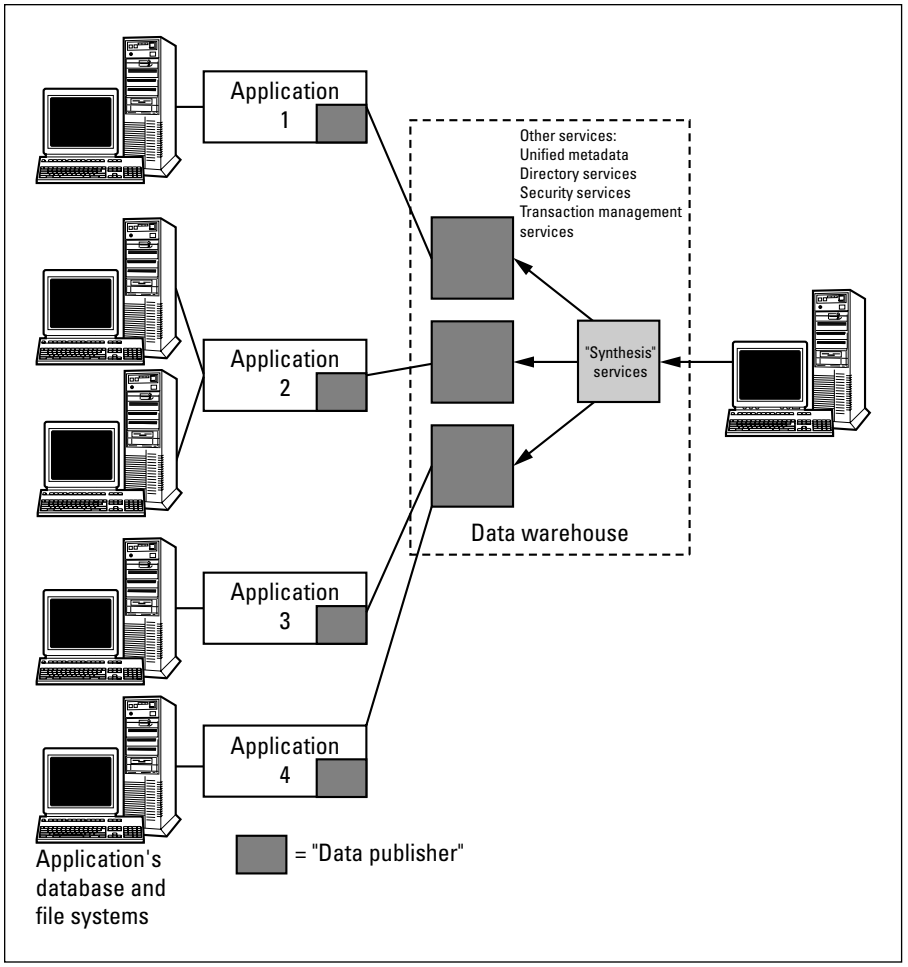


Figure 7-4: Enterprise Information Integration (EII) can provide a virtual data warehousing service.

I highly recommended that you don't build a virtual data warehousing environment to access source data directly, in its native format. As mentioned earlier in this chapter, your challenge isn't figuring out how to join cross-platform databases (combining IMS data with DB2 data, for example) and handling those types of system-level transformation, it's ensuring that the quality of the data is high and doesn't require the user to manually cleanse the data.

Each application should therefore be warehouse-enabled (refer to Chapter 25) and contain a data publisher that's responsible for all the middleware services (such as extraction and quality assurance), as specified in the environment's business rules. The data publisher could conceivably operate almost in real-time mode, like it would have to do in an operational data store

(again, refer to Chapter 25), or it could function in a periodic (batch-oriented) mode if instantaneous updates aren't required. In this situation, the data publisher is a mini-middleware product embedded in the application (or a service accessed by the application).

When you think of virtual data warehousing, replace the question "Can I get to the data?" with the question "Can I get to usable data?" The data publisher plays an important role, and should not be neglected.

You also can't neglect data architecture. Just because you're developing components in a bottom-up manner and they're being accessed in place, rather than being copied into a larger data warehouse database, doesn't mean that you can neglect this function. Say that one component stores customer IDs as five-digit numbers after transformation occurs and contains only customers who made purchases within the past six months. And another component, which contains all customers who have ever bought your company's products, uses seven-character alphanumeric identifiers. In this situation, you might have the same type of data mismatch problems you would if you were accessing data directly from the sources. Although EII allows for differences between component contents, you must understand and manage the differences so that you don't impede the business intelligence mission.

Additional EII services

Your virtual data warehousing environment includes services that a single database would handle in a centralized environment. These true middleware services complement the traditional data warehousing middleware, such as extraction and transformation:

- **A unified metadata service:** Users see a single logical view of the environment's contents without having to know the location and particulars of each component.
- **A directory service:** Individual components within the environment, even if they're relocated or otherwise modified.
- **Security services:** These services handle permissions, authentication, and other security needs in a distributed environment.
- **Synthesis services:** Unlike traditional Internet usage, for example, where each hit returns information that probably won't have to be correlated with information from other hits, a virtual data warehouse might have to combine facts from one component with facts from another. (Figure 7-4 shows a synthesis service component within the virtual data warehouse.)

  If the data warehousing environment, which "knows" that it's *virtually* (remotely) accessing data from multiple sources, has a component that handles the synthesis of results from the other components, the responsibility for consolidating these independent results sets can be offloaded from the requesting business intelligence tool.

- **Transaction-management services:** In addition to the synthesis service, which is a type of transaction-management function, these services provide routing, load balancing, conflict resolution, and other functions necessary to ensure data integrity. (Yes, you need these services, even in a supposedly read-only environment.)

Facing the EII infrastructure challenge

Wow! Virtual data warehousing sounds like a fairly neat, state-of-the-art idea. Why isn't it more widespread?

The answer, in a word, is infrastructure. Although you can talk all you want about emerging networking and communications technologies, and the tremendous throughput we'll all have someday, most corporations are still several generations behind the state of the art in their networking infrastructure. They're struggling to deal with the major investment required for what's essentially the Internet and Distributed Computing Age.

Delivering virtual data warehousing that includes EII services requires significant throughput of data in what's often an unpredictable manner. The irony is that a virtual data warehousing environment almost assuredly requires, over time, significantly less data movement across the enterprise than traditional data warehousing, which has a philosophy of copying millions (perhaps billions) of bytes, just in case someone wants to ask a question possibly requiring that data.

Traditional data warehousing has the advantage, however, of working the clock (as they say in football) by scheduling large-scale batch loads for either system downtimes or relatively light operational loads, using offline data-transfer means (tape, for example), rather than network file transfers and other tricks of the trade.

Until an organization's communications infrastructure can pump the data through, EII services and concepts (such as virtual data warehousing) probably will remain on the fringes, just out of reach.

If you're interested in virtual data warehousing as an architecture that uses EII services, here's a list of what you have to do:

- **Supercharge your networking infrastructure.** Yes, it's expensive, but you gotta do what you gotta do.
- **Stop thinking about data warehousing in centralized terms.** Think distributed but interconnected subject areas of data. Free your mind!
- **Insist that any new application developed for your organization, no matter who does it, be warehouse-enabled.** (See Chapter 20.)

- **Supercharge your EII servers with memory and processors.** Reading data from memory is far faster than across the network or off a disk. Most EII servers enable you to cache the results of a query in memory and share them across users. By providing the proper memory and processor configuration, the EII server gets faster with each user request through reuse.
- **Think enterprise!** Just because you're dealing with bottom-up component development doesn't mean that you're developing a stand-alone piece of data. The last thing you want to wind up with is an islands-of-data-mart problem to replace the islands-of-data situation you've been dealing with your entire career. (I know that you have because we all have.) In a traditional data warehousing environment, you're forced to deal with these enterprise-scale data architecture issues. Don't let your foray into virtual data warehousing undermine all the good that you've done in your organization when it comes to dealing with enterprise data architecture.

Vendors with Middleware Products for Data Warehousing

The following sections list the names of vendors that offer data warehousing middleware products you might want to take a look at.

Composite Software

```
www.compositesw.com
```

Composite provides Enterprise Information Integration (EII) products and technology. By using Composite, you can access and combine data from disparate data sources, including packaged applications such as SAP, custom applications, data warehouses, and XML data stores, as well as other data sources.

IBM

```
www.ibm.com
```

IBM Information Integration provides a data integration platform that incorporates EII; Extract, Transform, and Load (ETL); change data capture; replication; and other data warehousing middleware functionality.

Informatica

```
www.informatica.com
```

Informatica provides a platform for data integration. The specific offerings from Informatica include the PowerCenter, On Demand data loader, B2B Data Exchange, and Data Quality.

Ipedo

```
www.ipedo.com
```

Ipedo XIP is a data virtualization platform that allows organizations to aggregate and present information from disparate sources into their business intelligence, enterprise reporting, corporate portal, or service-oriented architecture (SOA) applications. Ipedo XIP's Virtual Views allow applications to provide users with the information they want on-demand in real-time.

Microsoft

```
www.microsoft.com
```

Within the Microsoft SQL Server product offering, you can find an ETL component known as SQL Server Integration Services (SSIS). This ETL technology leverages Microsoft's .NET and SQL Server platforms for execution and deployment.

Oracle

```
www.oracle.com
```

Oracle Warehouse Builder (OWB) enables you to design ETL processes and implement data integration solutions for your data movement requirements. Using a common metadata repository, OWB combines data integration capabilities with enterprise data quality tools to deliver end-to-end data integration optimized for Oracle databases. Additionally, the Oracle Business Intelligence (BI) foundation component provides a unified business model, along with an enterprise information model that unifies metadata across the Oracle BI tools and analytical applications. This BI infrastructure integrates with a majority of the popular data sources, major business applications, and databases, including IBM DB/2, Teradata, Microsoft SQL Server; SAP Business Information Warehouse (BW), Microsoft Analysis Services; flat files; XML data; and unstructured data.

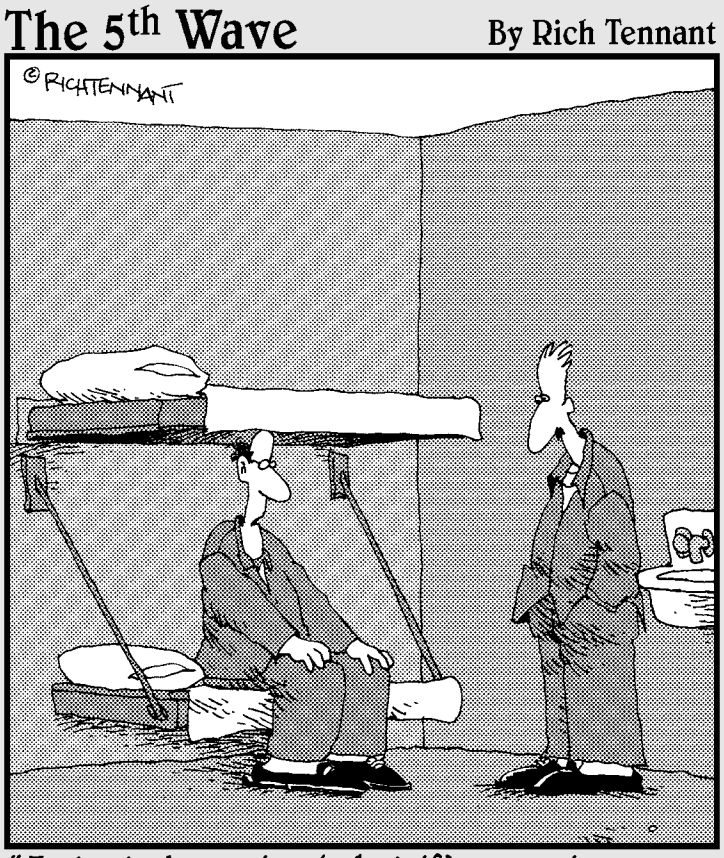
Sybase (Avaki)

```
www.sybase.com
```

Sybase Avaki is an enterprise application integration software product designed for EII systems. Avaki uses a federated approach, delivering data from original sources.

Part III

Business Intelligence and Data Warehousing



"I started running 'what if' scenarios on my spreadsheet, like, 'What if I were sick of this dirtwad job and funneled some of the company's money into an off-shore account?"

In this part . . .

No, the phrase business intelligence isn't an oxymoron. You have no reason to build a data warehouse, in fact, if you don't have the objective to provide business intelligence to members of your organization.

But what is business intelligence? The chapters in this part of the book explain different aspects of business intelligence — from the simplest to the most complex — in down-to-earth, hype-free language, complete with examples and a description of products that might interest you.

Chapter 8

An Intelligent Look at Business Intelligence

In This Chapter

- Understanding the different kinds of business intelligence
- Using business intelligence in data warehousing

In Chapters 5 and 6, I discuss the various types of database management systems (DBMSs) that are suitable for data warehousing. In the early days of data warehousing, controversy swirled around what type of DBMS to use — the focus was very technical, not solution-oriented. At times, the controversy stemmed from the lack of understanding of a new technology form and the immaturity of data warehousing architectures.

It is now understood that you can make the generalization that you create a data warehouse for the purposes of business intelligence. One variety of business intelligence includes query, reporting, and OLAP functionality.

If you build a data warehouse without clearly understanding what types of business intelligence your organization needs, you're almost certain to build and put into use something that doesn't even come close to providing the business value you're seeking, regardless of how error-free your data, how sophisticated your user tools, and how wonderful your environment's performance.

This chapter helps you make sense of the business intelligence quandary. (***Note:*** Much of the information in this chapter is explored in-depth in the other chapters in this part of the book. This chapter, however, provides a place where you can skim through the descriptions of each of these categories to see how they relate to each other.)

The Main Categories of Business Intelligence

At the outset of your data warehousing project, don't focus on the type of tools you need — yet. Instead, concentrate on figuring out the types of questions users will ask against the data warehouse's contents, the types of reports that will be run and for what purposes, and the general models of processing that will occur.

To help you get past hype, buzzwords, and techno-jargon, use the model shown in Table 8-1, which describes the four categories of business intelligence functionality.

Table 8-1 Business Intelligence Categories

Type	*Information You Want*	*See This Chapter*
Basic querying and reporting	"Tell me what happened."	9
Business analysis (OLAP)	"Tell me what happened and why."	10
Data mining	"Tell me what might happen" or "Tell me something interesting."	11
Dashboards and scorecards	"Tell me a lot of things, but don't make me work too hard."	12

Each of the four categories in Table 8-1 describes a way of accessing data, doing something with the information that's retrieved, and providing information to whomever requested it. Each category, however, has different attributes.

Querying and reporting

Basic querying and reporting largely represents the traditional uses of data for analytical purposes. The data is retrieved in accordance with either regular standard reports or in response to a particular question (an ad hoc query, for example); and then it's formatted and presented to the user in a specific format, either on-screen or in a printout. The interaction model is usually a set of regular, predictable steps:

1. **Make a data request.**
2. **Retrieve the data.**
3. **Manipulate the data slightly.**

   Summarize or reorganize, for example, if necessary.
4. **Format the data.**
5. **Present the data.**

Business analysis (OLAP)

Business analysis is the term used to describe visualizing data in a multidimensional manner. Query and report data typically is presented in row after row of two-dimensional data. The first dimension is the headings for the data columns; the second dimension is the actual data listed below those column headings.

Business analysis allows the user to plot data in row and column coordinates to further understand the intersecting points. In essence, this visualization of OLAP is most often utilized with an OLAP specialty database, although the business analysis can be done without such a database. The most common form of this business analysis involves using Microsoft Excel PivotTables, in which a user can plot data as rows, columns, or intersecting cells.

Business analysis introduces analytical processes and a degree of variability into the user interaction model. Conceptually, the first steps are pretty much the same as the steps in the preceding section for querying and reporting, but then the user takes over:

6. **Manipulate the data.**

   Look at it in a different way. Place some data on the rows, place other data on the columns, place the interested measurement at the intersecting cell.
7. **View the new results.**

   Change the presentation style of the data into a bar graph or line chart. Add in another column of data to refine the points on the graph. In addition, the user can request the data beneath the data, such as the details underneath a summary.

From that point, the variability of the process might iteratively cycle through the preceding steps (continually manipulating and reviewing the new results, for example) or even add new data for more analysis.

Data mining

At times, data mining isn't comingled with the other forms of business intelligence. This lack of integration occurs for two reasons:

- Business users don't have the required knowledge in data mining's statistical foundations.
- The mainstream business intelligence vendors don't provide the robust data mining tools, and data mining vendors don't provide robust business intelligence tools.

Data mining tools provide a degree of technical analysis that requires a base understanding in statistical algorithms to be successful in their use.

Data mining is often presented as a magical technique that you can use to uncover the secrets of the universe from your organization's data. In reality, data mining is an umbrella term for a series of advanced statistical techniques and models born in the 1980s as part of artificial intelligence research (neural networks, for example). I don't focus on individual technologies in this chapter (you can read about those in Chapter 11), but data mining as a technique has one or both of these aspects:

- **Predictive:** Data mining tools and capabilities search through large volumes of data, look for patterns and other aspects of the data in accordance with the techniques being used, and try to tell you what *might* happen based on the information that the data analysis found. Notice the emphasis on the word *might:* Data mining is a technique of probability, not a fortune-telling service.
- **Discovery-oriented:** Both the basic querying-and-reporting and business analysis/OLAP categories of business intelligence tools provide business intelligence based on either questions users explicitly ask (sort of the question of the moment) or "institutionalized" questions that members of the organization regularly ask in the form of regular reports (or both). The key word is *question*: If no questions are asked, no answers are forthcoming.

Data mining's discovery-oriented nature is intended to provide answers, even if you don't ask any questions. (I always refer to this model as "tell me something interesting, even if I don't know what questions to ask.") The data mining system typically provides these answers by building complex models that are used to analyze data, looking for some trend or tendency within the data that might be appropriate, and then telling you what it found.

Dashboards and scorecards

The early and mid-1980s experienced a frenzy in executive information systems (EIS) technology, a sort of predecessor to the 1990s data warehousing boom. Early EIS technology received a mixed welcome and sort of faded away near the end of the decade. Some people therefore consider EIS to be a predecessor to, but not a relative of, data warehousing. When you consider the aspect of entire systems, this view seems accurate because many EISs were built on top of relatively simple data extracts (see Chapter 22) with a narrow range of content.

In this new era of data warehousing, though, EIS systems are characterized by portals that contain dashboards and scorecards. Therefore, EIS is alive and well — just more mature and informative. Dashboards and scorecard environments best serve the broad category of users who want to receive key business information and indicators that can help them understand how the users are performing.

The analogy of your car's dashboard is very useful here. You can look at your tachometer to find out that your idle speed is somewhere near 1200 RPM. You also realize that the red line on the tachometer signifies danger — and therefore, you avoid revving the engine to that level. This is your dashboard. If, for some reason, you notice that the engine is idling faster — say, 3000 RPM — you might take your car to the mechanic for further analysis. The mechanic wants to investigate the details, making him or her similar to users of query and reporting technology. These individuals are very interested in the detailed information that can assist in fixing a "red line" problem. There are subtle differences between dashboards and scorecards that are covered in Chapter 12. In general, a dashboard presents current information on your operational performance while a scorecard shows your performance against a plan or set of objectives.

Dashboard and scorecard users certainly don't want to sift through reams of data from dozens or hundreds of reports. The philosophy of the dashboard and scorecard user is, "Tell me what I need to know — just a little information so that I know I'm on the right course — and please don't make me work too hard to get to it!"

Results either from querying and reporting tools, or from business analysis tools, typically feed dashboards and scorecards. A tool in one of those categories does the work, and a dashboard or scorecard makes the result available to the user.

Dashboards and scorecards aren't only for executives! And executives don't use only dashboards and scorecards to the exclusion of other categories of tools, either. Every person driving a car has a dashboard, and in business, all employees can have their own personal dashboards or scorecards to align their activities across the enterprise.

Dashboard and scorecard technology involves two major types of environments (both discussed in Chapter 12):

- **Briefing books:** A *briefing book* is an electronic (though it can be printed) sequence of key information and indicators that someone regularly uses for decision-making or performance monitoring. Users typically don't "wander" through information; rather, they take a relatively particular path.
- **Command centers:** The *command center* might be a console of on-screen push-buttons (perhaps a lot of them), each of which shows the user a different kind of information — a report, document, image, or indicator.

Command center interfaces are tailor-made for multimedia integration environments, such as Web pages presented on your company's intranet. From such an environment, you can integrate key components, such as the enterprise strategy, high-level plan focal areas, and performance across the organization. Incorporating the dashboard and scorecard information into such a Web page, along with nontraditional data, makes for a very robust business intelligence environment.

Other Types of Business Intelligence

Alas, the neat, organized model that has four different types of business intelligence categories can be expanded for more complex applications. For example, an OLAP or dashboard tool might have geographical information system (GIS) capabilities — or it might not. As shown in Figure 8-1, several other horizontal categories of business intelligence can apply to some or all of the categories discussed in the preceding sections.

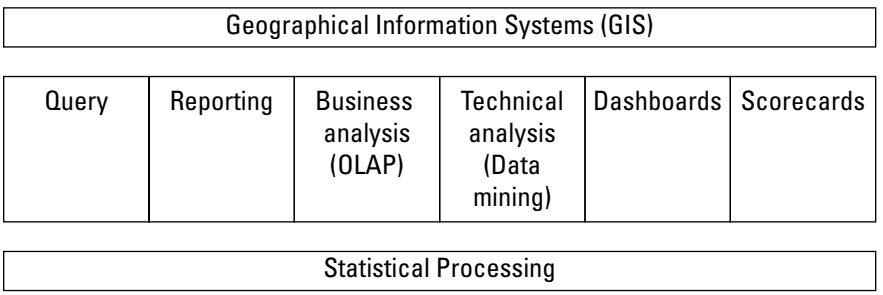


Figure 8-1: Business intelligence includes horizontal categories that span some or all of the vertical types.

Statistical processing

Most business intelligence tools do rudimentary statistical processing, such as averages, summaries, maximum values, minimum values, and standard deviations.

Think back, though, to that statistics class you endured in college. (If statistics class was easy for you, more power to you.) Remember z-scores? Chi-square tests? Poisson distributions? Some folks in the real world really use those concepts (and many other related concepts that most of us left way behind) as part of their jobs.

These people always want statistical functionality integrated into a business intelligence tool so that they don't have to save results from one tool into an intermediate storage facility (such as a Microsoft Excel spreadsheet) and import that data into a statistical tool for processing. Tool integration is a good thing!

Additionally, a large part of some data mining processing is based on *heavy stats* (advanced statistics, such as probabilities). Some environments include simulation and gaming capabilities that can identify and test various outcomes, and try to handle sophisticated what-if processing based on real data, rather than on assumptions and hypotheses.

Geographical information systems

Geographical information systems (GISs) make up a category of business intelligence functionality that spans multiple categories. The simplest way to understand GIS technology is to consider the concept of maps. You could, for example, do querying in a tabular manner and show product sales by country. Those sales are divided into product sales by territory and then broken down into product sales by increasingly smaller groupings, down to product sales by department in each store, for example.

I have a customer who made the profound statement, "Our business is geographically based — why can't I see our key information on a map?" GISs do just that: They provide a way of viewing information based on presenting that information by using maps. Suppose that all countries colored in red indicate that sales revenue is lower in the most recent quarter than it was in the preceding quarter. If you click your mouse on any of those countries, another map appears, which is divided into sales territories and again color-coded. Clicking on a red-colored territory displays another map (a U.S. map by state, for example) with additional levels of detail.

When you use GIS technology, data isn't just managed in a hierarchy such as this:

Departments➪stores➪states➪territories➪countries

The data is also managed *spatially* — in a manner that's sensitive to on-screen layout. For example, when you click a map of Pennsylvania, the GIS would "know" that you want to see sales in Pittsburgh, without you having to type or otherwise select that city from a drop-down list or other on-screen control.

Mash-ups

The capability to mix and match information and presentation services, such as those found on the Web, is a simple example of a mash-up. Companies such as Google, Yahoo!, and Microsoft are driving much of the technology platforms for future applications and provide key technologies in the area of mash-ups. Therefore, the example for GIS (in the preceding section) could actually take your standard database query and present that data by using Microsoft's Virtual Earth technology. Furthermore, you might want to combine that information with external benchmark information to give you greater insight. For example, if your marketing team felt that a new product would sell well to those individuals who have a median income of $50,000, you could plot census data with your sales performance data on a map provided by Google, Yahoo!, or Microsoft. How cool is that?

Although few companies are using mash-ups today, mash-ups will likely emerge as the future presentation vehicle. Current mash-ups combine external data, such as information that you can find on Craigslist or Amazon, with external visualization tools, such as Google documents or maps.

Business intelligence applications

While the business intelligence and data warehousing markets mature, large vendors of commercial off-the-shelf software, as well as specialty boutique firms, have started offering out-of-the-box business intelligence applications. These solutions typically are targeted at a functional area of a business, such as finance, supply chain, or customer relationships. The solutions combine all aspects of what I discuss in Chapter 3: subject area data content that has an associated database and set of business intelligence tools. These business intelligence applications can help solve 60 to 70 percent of issues you might commonly find in a functional area of your business. The application comes with a set of predefined content, views, or reports that enable you to simply hook up the application to your current systems and go. Vendors such as SAP and Oracle dominate this area because they also control much of the "run the business" components required as data sources for the "monitor the business" applications that they provide.

Business Intelligence Architecture and Data Warehousing

The early days of business intelligence processing (any variety except data mining) had a strong, two-tier, first-generation client/server flavor. (Some business intelligence environments that were hosted on a mainframe and did querying and reporting were built with a centralized architecture.)

Conceptually, early business intelligence architectures made sense, considering the state of the art for distributed computing technology (what really worked, rather than today's Internet, share-everything-on-a-Web-page generation).

Many of these early environments had a number of deficiencies, however, because tools worked only on a client desktop, such as Microsoft Windows, and therefore didn't allow for easy deployment of solutions across a broad range of users. Additionally, long-running reports and complex queries often bottlenecked regular work processes because they gobbled up your personal computer's memory or disk space.

Most, if not all, tools were designed and built as *fat clients* — meaning most of their functionality was stored in and processed on the PC. In addition to the bottleneck problem, all users' PCs had to be updated because software changes and upgrades were often complex and problematic, especially in large user bases.

The beginning of a new era of business intelligence architecture has arrived, regardless of whether your tool of choice is a basic querying and reporting product, a business analysis/OLAP product, a dashboard or scorecard system, or a data mining capability. Although product architecture varies between products, keep an eye on some major trends when you evaluate products that might provide business intelligence functionality for your data warehouse:

- **Server-based functionality:** Rather than have most or all of the data manipulation performed on users' desktops, server-based software (known as a *report server*) handles most of these tasks after receiving a request from a user's desktop tool. After the task is completed, the result is made available to the user, either directly (a report is passed back to the client, for example) or by posting the result on the company intranet.
- **Web-enabled functionality:** Almost every leading tool manufacturer has delivered Web-enabled functionality in its products. Although product capabilities vary, most products post widely used reports on a company intranet, rather than send e-mail copies to everyone on a distribution list.

- **Support for mobile users:** Many users who are relatively mobile (users who spend most of their time out of the office and use laptops or mobile devices, such as a Blackberry, to access office-based computing resources) have to perform business intelligence functions when they're out of the office. In one model, mobile users can dial in or otherwise connect to a report server or an OLAP server, receive a download of the most recent data, and then (after detaching and working elsewhere) work with and manipulate that data in a standalone, disconnected manner. In another model, mobile users can leverage Wi-Fi network connectivity or data networks, such as the Blackberry network, to run business intelligence reports and analytics that they have on the company intranet on their mobile device.
- **Agent technology:** In a growing trend, intelligent agents are used as part of a business intelligence environment. An intelligent agent might detect a major change in a key indicator, for example, or detect the presence of new data and then alert the user that he or she should check out the new information.
- **Real-time intelligence:** Accessing real-time, or almost real-time, information for business intelligence (rather than having to wait for traditional batch processes) is becoming more commonplace. In these situations, an application must be capable of "pushing" information, as opposed to the traditional method of "pulling" the data through a report or query. Like with traditional data-extraction services (described in Chapter 7), business intelligence tools must detect when new data is pushed into its environment and, if necessary, update measures and indicators that are already on a user's screen. (In most of today's business intelligence tools, on-screen results are "frozen" until the user requests new data by issuing a new query or otherwise explicitly changing what appears on the screen.)

Chapter 9

Simple Database Querying and Reporting

In This Chapter

- Looking at basic querying and reporting functionality
- Determining whether you need only basic querying from your data warehouse
- Designing your relational database to support querying and reporting
- Checking out some tools for your data warehouse

Querying and reporting is the low end of the spectrum of business intelligence functionality that applies to your data warehouse. (Chapter 8 provides an overview of business intelligence functionality.) Querying and reporting handles "tell me what happened" processing that's relatively static and predictable, for the most part. The data is retrieved in accordance with either regular standard reports or in response to a particular question (an ad hoc query, for example). Then, that data is formatted and presented to the user either on-screen or on a printout.

The interaction model for querying and reporting business intelligence typically follows a pattern of regular, predictable steps (these steps are also in Chapter 8, so you can skip ahead if you've already read that chapter):

1. **Make a data request.**
2. **Retrieve the data.**
3. **Manipulate the data slightly.**

   Summarize or reorganize, for example, if necessary.
4. **Format the data.**
5. **Present the data.**

These steps don't vary much between tools, scenarios, or users. If a user decides that the presented result looks odd or otherwise needs to be augmented, the process usually just begins again with a new request.

What Functionality Does a Querying and Reporting Tool Provide?

To help you understand the functionality that a querying and reporting tool offers, this list describes some of the tasks they can help you perform:

- **Run regular reports.** Your organization might regularly produce standard reports that come from an operational system or from data extracted from one or more of those systems.
- **Create organized listings.** You might produce a list of all the salespeople in your company or those who meet a specific criteria (they cover more than two territories, for example), and their sales in the most recent month. Organized means that you can list your report or query alphabetically by salesperson's last name; alphabetically by customer name and the salesperson who covers that customer; by rank, from highest sales revenue generated to the least; or any other way you want to look at the data. Figure 9-1 shows an example, using fictional company names.
- **Perform cross-tabular reporting and querying.** Cross-tabular reports, sometimes called cross-tabs, are slightly more complex than a basic organized listing of data. In addition to the sequential, ordered vertical listing (the company's salespeople), you see across the top (the other axis) of the report a decomposition of various categories and values associated with each category. In the example shown in Figure 9-2, sales revenues are broken down by product.

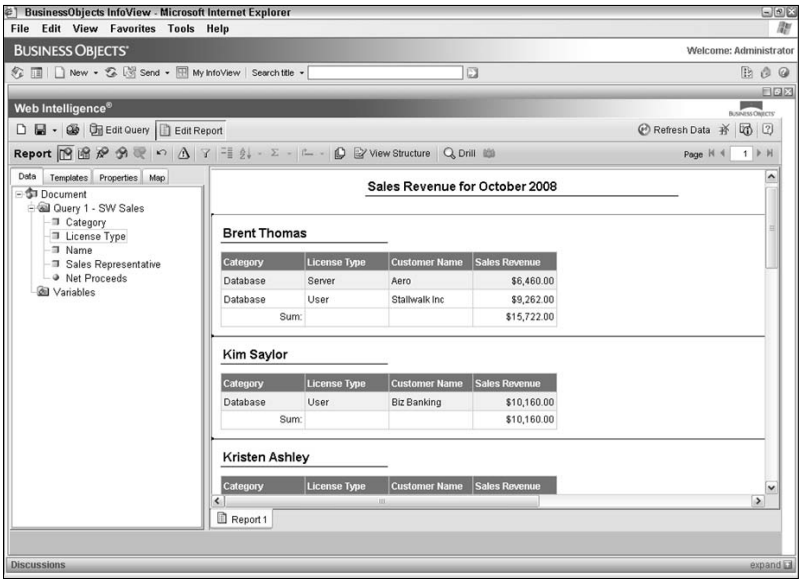
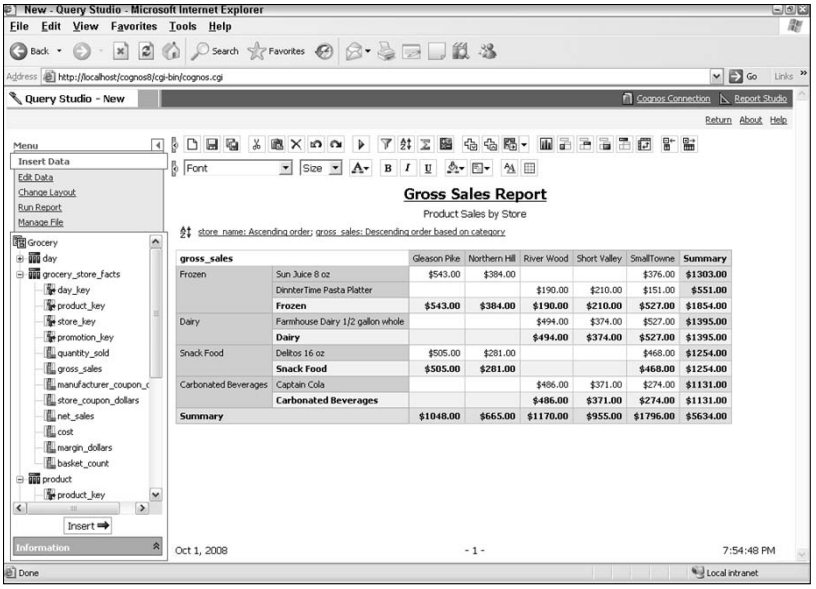


Figure 9-1: You can use a querying and reporting tool to produce a comprehensive, organized listing of monthly sales revenue by company salesperson.



gross_sales		Gleason Pike	Northern Hill	River Wood	Short Valley	SmallTowne	Summary
Frozen	Sun Juice 8 oz	$543.00	$384.00			$376.00	$1303.00
	DinnterTime Pasta Platter			$190.00	$210.00	$151.00	$551.00
	Frozen	**$543.00**	**$384.00**	**$190.00**	**$210.00**	**$527.00**	**$1854.00**
Dairy	Farmhouse Dairy 1/2 gallon whole			$494.00	$374.00	$527.00	$1395.00
	Dairy			**$494.00**	**$374.00**	**$527.00**	**$1395.00**
Snack Food	Delitos 16 oz	$505.00	$281.00			$468.00	$1254.00
	Snack Food	**$505.00**	**$281.00**			**$468.00**	**$1254.00**
Carbonated Beverages	Captain Cola			$486.00	$371.00	$274.00	$1131.00
	Carbonated Beverages			**$486.00**	**$371.00**	**$274.00**	**$1131.00**
Summary		**$1048.00**	**$665.00**	**$1170.00**	**$955.00**	**$1796.00**	**$5634.00**

Figure 9-2: This cross-tab breaks down sales revenue by product.

The barriers between business analysis/OLAP tools and products that have historically been oriented toward querying and reporting are getting somewhat grayer. Query tools, for example, commonly also permit some level of *drill-down,* an OLAP function that enables you to see underlying, more detailed data, as explained in Chapter 10.

The role of SQL

SQL is the official database language of the National Football League; any use of an SQL tool without the express written consent of the National Football League is strictly prohibited. (Sorry about that — a little too much football fever.)

SQL is the official database query language used to access and update the data contained within a relational database management system, or RDBMS.

The roots of SQL go back to IBM and its research labs during the early days of relational database technology. IBM and Oracle were among the first to adopt SQL as the language used to access their relational products (other RDBMSs used different languages that their respective vendors invented). In the mid-1980s, SQL was submitted for approval to both the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO), and during the next few years, other database access languages faded away. Later versions of the SQL standard were published every few years.

Doesn't OLAP do querying and reporting also?

Without jumping ahead to Chapter 10, let me answer a question you might be wondering about if you have business analysis, OLAP experience: Wouldn't an OLAP tool have the capabilities listed in the section "What Functionality Does a Querying and Reporting Tool Provide?" in this chapter, plus many more?

Absolutely! Querying and reporting is, at least conceptually, a subset of OLAP functionality. OLAP architectures and capabilities are more complex than those oriented toward reporting and querying, though. I split the functionality of OLAP between the visualization (business analysis) and storage (OLAP specialty database) for this reason. More business intelligence tool vendors are enabling the use of their query and reporting tools against relational and OLAP databases. However, business analysis tools don't always support visualizing relational databases.

If all you need is basic querying and reporting capabilities — and you're absolutely sure that's all you need! — you might not want to deal with the overhead of setting up and maintaining an OLAP environment if you don't anticipate gaining much business benefit from those capabilities (if you just don't plan to use them, for example).

The significance of SQL for querying and reporting (and for data warehousing) is that the language has represented a *mostly* standard way to access multiple RDBMS products.

Each RDBMS product has a slightly different SQL dialect. Although the basic syntax is the same, especially for the most commonly used commands, all SQL dialects are slightly different. In the early 1990s, despite these syntactical differences, several different efforts provided a common gateway to SQL RDBMS products. The most successful was Microsoft Open Database Connectivity (ODBC). The phrase *ODBC-compliant* became important to RDBMS applications in the early and mid-1990s. A similar standard for the Java community also emerged in the late-1990s — Java Database Connectivity (JDBC). Virtually all major database manufacturers today have accessibility through both of these standard connectivity interfaces.

Technical query tools

The use of SQL as the basis for most querying and reporting tools was both good and bad for data warehousing. On the positive side, many more product-to-product matchups are possible in data warehousing environments, enabling tools to be provided both by RDBMS vendors and other third-party vendors.

On the negative side, though, SQL is a relatively complex language after you get past the basics. A series of query tools primarily allow users to type in and edit SQL queries. I don't see these tools as designed for end users and therefore refer to them as technical query tools, though I'm amazed how often I find these tools deployed in end-user organizations.

User query tools

Most end-user querying and reporting tools provide visually oriented, painting, environments that enable users to design screens for report layouts, the data columns desired for the report, or the rows of data that they want to select (only salespeople who have met their quota, for example).

Using all this "painted" information, most tools have increasingly taken a smart query generation approach. Instead of generating a single, overly complex SQL statement that could get you an A in database class but draw a disgusted shake of the head from someone who has done this stuff in the real world, a sequence of SQL statements (usually taking advantage of temporary tables for intermediate results) is generated. This sequence, in effect, decomposes the query into a more efficient series of steps.

Reporting tools

When end users want more complex user interaction or sophisticated formats, a tool with more reporting features is leveraged. You can find a separation between pure query tools and pure reporting tools. The query tool provides data access, filtering, and simple formatting. If you're distributing reports across your enterprise or need to generate form-safe presentation, you use a reporting tool.

Like with query tools, reporting tools provide an environment that enables you to create sophisticated layouts that focus on formatting the data retrieved by the database query.

The idea of managed queries and reports

Turning the average group of users loose with a querying and reporting tool and directing them to "go forth and ask all kinds of questions against the data warehouse" is a dangerous idea. Even with visual tools to aid in generating queries and reports — and even with wizards to help — at least some of these people will undoubtedly issue all kinds of overly complex queries that do little more than bring your data warehouse to its knees.

Although you certainly want to enable *power users* (technically sophisticated users) to do whatever they need to do, you should handle average users differently.

In a *managed querying and reporting environment,* users get templates of queries or reports that they can use to ask questions and perform business intelligence functions in the data warehousing environment. Each template is set up so that it's not too rigid and not too flexible — it's just right.

Suppose that a user must run a report periodically (daily, weekly, monthly, or quarterly) and needs different data, depending on which period is indicated. For example, the user might need to run a weekly report for all product sales to wholesalers, a monthly report for all product sales directly to retailers, and a quarterly report for wholesaler, retailer, and cross-company product sales. Also, one variation of the report must show sales for the entire United States, and the user needs to distribute another report for each sales territory to the appropriate sales manager.

To complicate matters, sales managers sometimes request a weekly preview run of what would ordinarily be a quarterly or monthly report. The reports can't be totally *canned* (set up once and never modified), and each run requires a little (although not much) human input.

The managed environment might have a preset format and links to the tables that are needed. Before running the report, however, the user is prompted for which type of customer he or she wants to include in the report; whether the report covers the entire United States or a specific territory (or perhaps more than one); and any other parameters (for a previous month's sales if someone requests a rerun of a previous report, for example).

Other than these types of guided inputs, the queries and reports are managed and controlled to reduce the chance of runaway queries that use an enormous amount of system resources and probably never run to completion.

Is This All You Need?

Suppose that querying and reporting sounds like a good idea, but you really want to check out that OLAP stuff, too.

You can give some users basic tools and still let others use OLAP tools (and, for that matter, let your statistician use a data mining product). One of the benefits of designing and developing your data warehouse in this flexible, component-oriented manner is that you can equip users with the tools that are most applicable to their respective missions and the business value they're expected to provide from using the data warehouse.

Designing a Relational Database for Querying and Reporting Support

Your data warehousing environment or a specific data mart that your main data warehouse will feed might have the mission of generating a finite and predictable set of reports. This section gives you one approach to designing a relational database to support that mission, built around the principle of *database denormalization,* or deliberately violating good relational database design principles in the interest of performance efficiency. (Chapter 5 discusses normalization, if you're interested in the background to this approach.)

Denormalization is best suited for *quick-hit solutions,* in which you must get a small-scale relational data warehouse or a data mart up and running quickly. For example, you might create a denormalized relational database for a *specific* charter to produce a certain set of reports that will no longer be available as a result of a legacy system migration effort. Although denormalization isn't quite a dead end, it does create a great deal of duplicate data, and the database structures you create don't have much flexibility. Additionally, you probably have limited querying capabilities (in addition to your standard reports) because those capabilities are closely tied to the reporting structures formalized in the table design. Still, you might want to check out this approach.

A simple example of denormalization in Figure 9-3 shows what the source database tables look like in an application that tracks sales performance, with those tables structured primarily according to standard relational database design principles (they're normalized). To support the report format shown at the bottom of the figure, the source structures are mapped into a denormalized table from which the report can be generated without having to join any tables. (To put it more simply, your report runs very quickly.)

Note: A real-world example would involve many more tables (from 10 to 50 or more) and many more reports than Figure 9-3. This figure should get the idea across, however.

Alternatively, you might want to follow the principles and techniques of dimensional design. Because RDBMSs now have much less trouble dealing with dimensionally oriented structures than in the past, you're likely to get adequate performance for your reporting needs and still have the flexibility to support a large variety of ad hoc, multidimensional queries.

For rapid deployment that's reporting-oriented, though, at least consider denormalization-based design for relational data.

CUSTOMERS

CUST_ID	CUST_LAST_NAME	CUST_FIRST_NAME	CUST_TYPE	GROUP_ID
12345	Ashley	Kristen	W	1
23456	Thomas	Brent	W	2
34567	Saylor	Kimberly	R	2

GROUP_TYPES

GROUP_ID	GROUP_DESCRIPTION
1	Exporter only
2	Importer only
3	Importer/exporter

PRODUCT_MAIN

PRODUCT_ID	PRODUCT_DESCRIPTION	CATEGORY_CODE
1000	Glasses	1
2000	Coffee Cups	1
3000	Ski Poles	2

CUST_TYPES

CUST_TYPE	TYPE_DESCRIPTION
W	Wholesaler
R	Retailer
O	Other

SALES

DATE	CUST_ID	PRODUCT_ID	QUANTITY	DOLLARS
3/10/2007	12345	1000	3	$ 4.50
7/6/2007	23456	2000	15	$ 22.50
3/19/2008	34567	1000	55	$ 82.50
3/20/2008	12345	3000	1	$ 25.00
7/27/2008	23456	1000	1	$ 1.50
9/5/2008	34567	2000	30	$ 35.00

PRODUCT_CATEGORIES

CATEGORY_CODE	CATEGORY_DESCRIPTION
1	Drinking containers, household
2	Athletic gear

Required sales report format

PRODUCT_ CATEGORY	SALES_ DATE	GROUP_ TYPE	TOTAL_ QTY	TOTAL_ $	WH_ QTY	RET_ QTY	OTHER_ QTY	WH_ $	RET_ $	OTHER_ $
Drinking containers, households	yyyy/mm/dd	Exporters only	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##
	yyyy/mm/dd	Importers only	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##
	yyyy/mm/dd	Importer/ exporter	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##
Athletic gear	yyyy/mm/dd	Exporters only	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##
	yyyy/mm/dd	Importers only	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##
	yyyy/mm/dd	Importer/ Exporter	#,###.##	$##,###.##	#,###.##	#,###.##	#,###.##	$##,###.##	$##,###.##	$##,###.##

Figure 9-3: Normalized database tables in a source application.

Vendors with Querying and Reporting Products for Data Warehousing

The following sections list some vendors that provide querying and reporting tools you might want to consider using with your data warehouse.

Business Objects (SAP)

```
www.businessobjects.com
```

Business Objects, now an SAP company, provides a suite of products for query and reporting. Specifically, they provide

- **WebIntelligence:** Also known as WebI; a Web-based solution for ad-hoc querying, reporting, and analysis
- **Crystal Reports:** Formerly from Seagate Software, a tool for report creation and distribution

Both of these tools operate against the Business Objects semantic layer, known as a Universe, which maps business terms to their technical match within the data warehouse. Allowing for broader user adoption by minimizing the need for technical knowledge of how things are stored and organized within the data warehouse.

Cognos (IBM)

```
www.cognos.com
```

Cognos, now an IBM company, provides a suite of products for querying and reporting:

- **Query Studio:** A Web-based solution for ad-hoc querying
- **Report Studio:** A Web-based tool for report creation, query aggregation, and distribution

Both of these tools operate against the Cognos semantic layer defined within the design tool, known as Framework Manager, that maps business terms to their technical match within the data warehouse. Cognos's semantic layer allows for broader user adoption by minimizing the need for technical knowledge of how things are stored and organized within the data warehouse. The older Cognos technology also includes Impromptu, which was a Microsoft Windows–based tool that Query Studio and Report Studio have replaced.

Information Builders

```
www.informationbuilders.com
```

Information Builders produces the Focus family of reporting and querying products, which has its roots in the mainframe Focus system. WebFOCUS enables you to produce an ad-hoc query or managed reporting environment on a broad range of database management systems.

Microsoft

```
www.microsoft.com/bi
```

Microsoft provides a suite of data warehousing products, including products for reporting, although they don't currently provide a competitive query tool. The SQL Server product line incorporates SQL Server Reporting Services (SSRS), which provides a sophisticated server-side reporting tool. Because SSRS is a server-side technology, companies can deploy these reports for visualization purposes within Microsoft Excel and Microsoft SharePoint, as well as other integrated productivity tools.

Oracle

```
www.oracle.com/solutions/business_intelligence/index.html
```

Oracle has not only built its own tools over the years, but also acquired Hyperion (which had acquired Brio). Therefore, Oracle has a number of query and reporting tools to assist people in the area of data warehousing:

- **Oracle Business Intelligence Publisher:** A reporting solution for authoring, managing, and delivering highly formatted documents, such as operational reports, electronic funds transfer documents, government PDF forms, shipping labels, checks, and sales and marketing letters
- **Oracle Business Intelligence Suite, Enterprise Edition Plus:** A comprehensive business intelligence platform, formerly Siebel Analytics, which provides querying and reporting functionality across distributed data sources, including data warehouses

Chapter 10

Business Analysis (OLAP)

In This Chapter

- Getting a no-hype picture of OLAP
- Looking into OLAP features
- Checking out some OLAP vendors

A man walks down the street in Manhattan carrying a data warehouse under his arm. From the other direction, a woman approaches, carrying an OLAP. (Bear with me — this stuff will make sense in a second.)

The man sees a billboard advertising a watch he's had his eye on, and now it's on sale. To catch the ending date for the sale, he keeps his eyes on the billboard while he walks.

The woman notices a crowd of people gathered outside a theater and looks in that direction to see what the commotion is. She too continues walking.

Suddenly, the man and woman collide. Stunned for a moment, the man looks down and then says to the woman, "Hey, you got OLAP in my data warehouse!" The woman recovers from her surprise, looks down, and says to the man, "Hey, you got a data warehouse on my OLAP!"

Together, they both say, "Mmmmm . . ."

If you're at least as old as I am, this scenario should bring back memories of a mid-1980s series of commercials for a certain chocolate-covered peanut-butter-cup candy. If it doesn't sound familiar, check out the retromercials (old, original television commercials) on YouTube. You might see this one.

Step over from TV Land to Data Warehousing Land. Without OLAP, data warehousing would hardly be what it is today. At the same time, the roots of OLAP — or, more precisely, multidimensional business analysis — go back to the 1960s.

The Origins of OLAP

You can find an interesting history of OLAP, written by Nigel Pendse, on the Internet. It traces multidimensional analysis back to the APL programming language, developed at IBM in the late 1960s based on work done a few years earlier. Products were introduced throughout successive decades, including a convergence with spreadsheet technology in the 1980s. But not until the early 1990s, when data warehousing became popular, did the two disciplines click and make history.

OLAP-based business analysis and data warehousing made each other what they are today: an intensely popular mode of data analysis that almost every organization has either implemented or is considering.

Despite the natural synergy between data warehousing and OLAP, you can, and likely will, do more with a data warehouse than with OLAP. For that matter, OLAP isn't performed exclusively against a data warehousing environment. (If you want, you can use an OLAP tool against multidimensionally oriented tables in a low-volume production database.)

What Is Business Analysis?

Business analysis is the visualization capabilities provided by OLAP platforms. The business analysis style of accessing data relates to "tell me what happened and why" processing that you do after you build your data warehouse. Unlike querying and reporting tools, business analysis leverages OLAP functionality, enabling you to dig deeper, poke around a little, and (hopefully) come up with the "why" aspect of what's happening in your business.

The distinguishing characteristic of business analysis is that it enables you to perform multidimensional analysis. As discussed in Chapter 6, users have a natural tendency to view business results in different ways, organized by various dimensions. By using the dimension of time, for example, you can perform trend analysis. By using other dimensions particular to a given fact (products, business units, and geography, for example), you can get down to the nitty-gritty of finding problem areas, pinpointing your company's strengths, and generally getting a clear picture of what's going on and why it's happening.

The OLAP Acronym Parade

In the data warehousing world, business analysis hit the big time by operating against multidimensional databases specially structured to support this type of processing. These environments have come to be known as multidimensional OLAP (MOLAP) systems.

A corresponding approach (the archenemy of MOLAP) is relational OLAP, or ROLAP. ROLAP uses plain old relational databases, rather than specialized multidimensional structures. As discussed in Chapter 6, however, ROLAP requires that the DBMS products perform cross-table joining of data somewhat more efficiently than they traditionally have done.

Ah, but I'm not done yet. Hybrid OLAP, or HOLAP, is an attempt by vendors to call a truce in the ROLAP-versus-MOLAP war and bridge the gap. Different vendors are taking different approaches. Some vendors link a MOLAP front end to a ROLAP back end and route user requests to the engine that can serve the answer most efficiently. Other vendors cache multidimensional aggregated results for subsequent use, instead of re-creating those aggregates. Others cache the dimension data, such as product and customer, to optimize navigation while dynamically retrieving the fact data, such as sales amounts and costs.

Want more? How about desktop OLAP (DOLAP) environments, in which a client system (rather than an OLAP server) is the primary storage repository for a specialized multidimensional structure. DOLAP is particularly useful for laptop users who want to perform business analysis when they're not hooked up to the company network — while on the road or at a client site, for example.

To help explain the OLAP architecture, I've broken the OLAP world up into three architectural layers: business analysis (visualization), OLAP middleware, and OLAP databases. I discuss all three in the following sections.

Business analysis (Visualization)

Business analysis, or the visualization capabilities of OLAP, is the focal point of most end users. Probably the most familiar visualization tool that enables business analysis is Microsoft Excel's PivotTable. Within the visualization tool, OLAP features (discussed in the section "Business Analysis (OLAP) Features: An Overview," later in this chapter) are available. These features allow users to slice and dice, nest, pivot, trend, and change displays — all with the click of a mouse. (It sounds like I'm selling the ginsu knife collection, doesn't it?)

All these capabilities enable users to discover trends within the information presented without needing support from technical personnel. These trends might be good or bad — either way, the user has the ability to get to the root cause within the analysis.

OLAP middleware

OLAP middleware emerged in the late 1990s and early 2000s. In essence, standards have been driven to allow the world of OLAP to split between vendors that are good at business analysis (visualization) and those that are good at physical storage and access optimization. The OLAP middleware market is between standards, such as multidimensional expressions (MDX) and XML for Analysis (XMLA), and the ROLAP vendor community.

MDX was initially introduced by a consortium of vendors led by Microsoft as part of the OLE DB for OLAP specification. MDX provided a query language for OLAP data stores in a fashion very similar to SQL for relational databases.

XMLA is the most recent attempt at a standardized application programming interface (API) for OLAP products. What differentiates XMLA from previous attempts at a standard is that it offers broader solution provider support with companies such as Oracle (Hyperion), Microsoft, SAP, and SAS. XMLA allows client applications to talk to multidimensional or OLAP data sources through the exchange of messages back and forth by using Web standards — HTTP, SOAP, and XML. The query language used is MDX. Oracle (Hyperion) Essbase, Microsoft Analysis Services, and SAP Business Warehouse all support the MDX language and the XMLA specification.

ROLAP providers include vendor offerings from MicroStrategy, IBM/Cognos TM1, and Pentaho Analysis (Mondrian). You need a strong understanding of data access and modeling to properly implement these offerings because they most often work against a relational data store — simulating the OLAP database in memory on a server, called *building a cube on the fly*.

OLAP databases

OLAP databases are the physical storage locations for multidimensional databases. While middleware standards and processing capabilities of middleware solutions have evolved, this community of products has shrunk to a handful. The most popular include Oracle (Hyperion) Essbase, Microsoft Analysis Services, SAP Business Warehouse, and Cognos PowerPlay (PowerCubes).

First, an Editorial

This section provides you with some valuable context to the OLAP hype you might face, if you haven't already. Just browse the Internet and check out vendor's white papers about OLAP. Vendors make a serious amount of definitive statements about why whatever approach they *don't* sell won't work for you.

Here's my opinion: No OLAP silver bullet exists — at least, not of this writing. Here are some guidelines and trade-offs for OLAP architectures:

- Strongly consider a MOLAP solution (and check out ROLAP alternatives) if response time is a key factor; if summarized data, rather than detail-level data, can address your business intelligence mission; and if your data volumes won't grow to more than 50 gigabytes.
- Some financial-analysis and budgeting packages are tied to a particular multidimensional database product. In these cases, you might want to set up a financial-analysis data mart that uses MOLAP technology, regardless of the other data marts and data warehouse databases in your environment.
- If detail-level data is important to your business intelligence processing and you're likely to have a very large database (VLDB) environment, consider ROLAP solutions. You might also want to consider a HOLAP environment, however, if your descriptive, dimensional data is relatively static as described in the section "The OLAP Acronym Parade," earlier in this chapter.
- Regardless of the approach you take, check out products carefully. Look for hidden performance "gotchas," recalculation times for multidimensional structures, inefficient relational processing in ROLAP environments, scalability problems (after you reach a certain number of users or a certain database size, problems occur), and other events that can cause your data warehousing environment to be less than satisfactory.

Business Analysis (OLAP) Features: An Overview

The following sections present a brief list of some business analysis (OLAP) features so that you can get an idea of what to expect from products. Because it's impossible to discuss this subject in detail in a few pages, I hit just the strategic highlights to show you what this area means to you and your data warehouse.

For more details about these and other features, check out vendors' Web sites (see the section "Data Warehousing Business Analysis Products," later in this chapter), product brochures, and demos; textbooks and tutorials; magazine articles; and other sources.

Drill-down

Drill-down analysis is probably the feature that average users use most frequently. The concept is fairly simple: A user can see selective increasing levels of detail as required.

Figure 10-1 shows a report of Zip's Video and Games third-quarter sales, broken down by region on one axis and by product on the other.

Suppose that you want to see the next level of detail for sales in the northeast region. You can double-click that row to display an additional level of detail for sales by product in Pennsylvania and New Jersey (as shown in Figure 10-2).

Say that you're curious about the additional level of detail for sales in cities in Pennsylvania. Double-click Pennsylvania, and — presto! — you have the results shown in Figure 10-3: sales for each city.

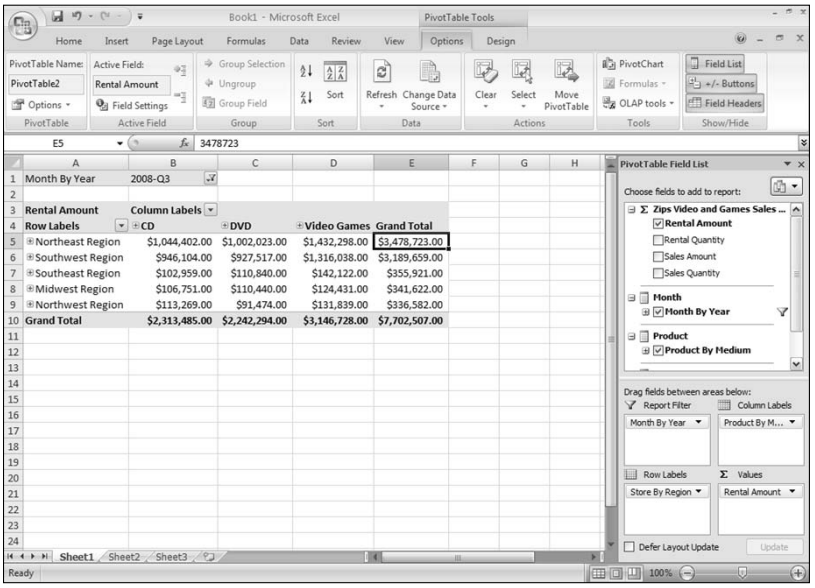


Figure 10-1: A basic level of detail on a quarterly sales report.

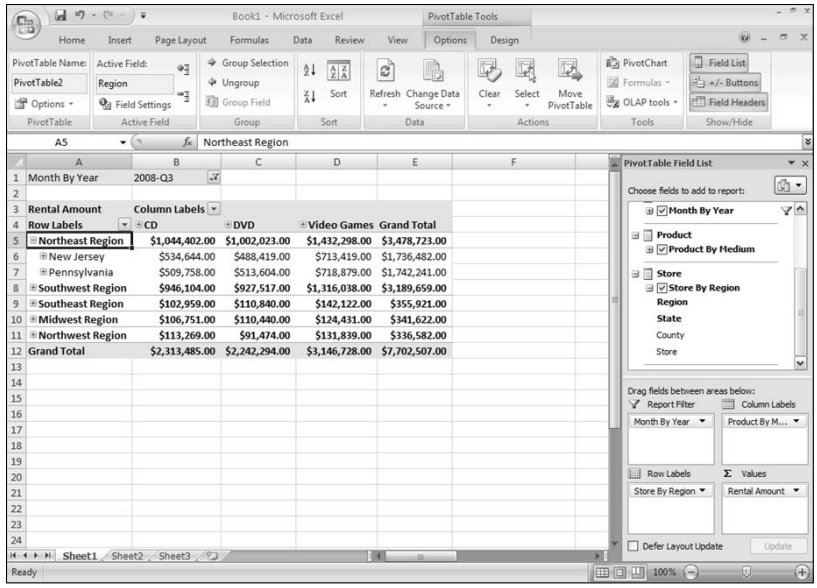


Figure 10-2: Drilling down to the next level of detail for the northeast region doesn't affect other summary levels.

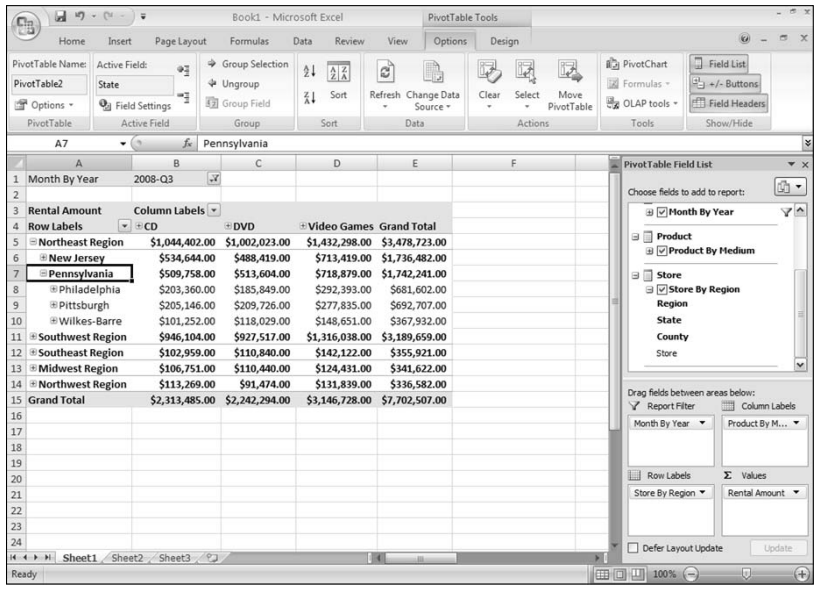


Figure 10-3: Drilling down even further, you can see sales by city in Pennsylvania.

Now, you want to see the sales amount for each store. After you double-click each city's name, you see the lowest level of detail — quarterly sales by product and by store — as shown in Figure 10-4.

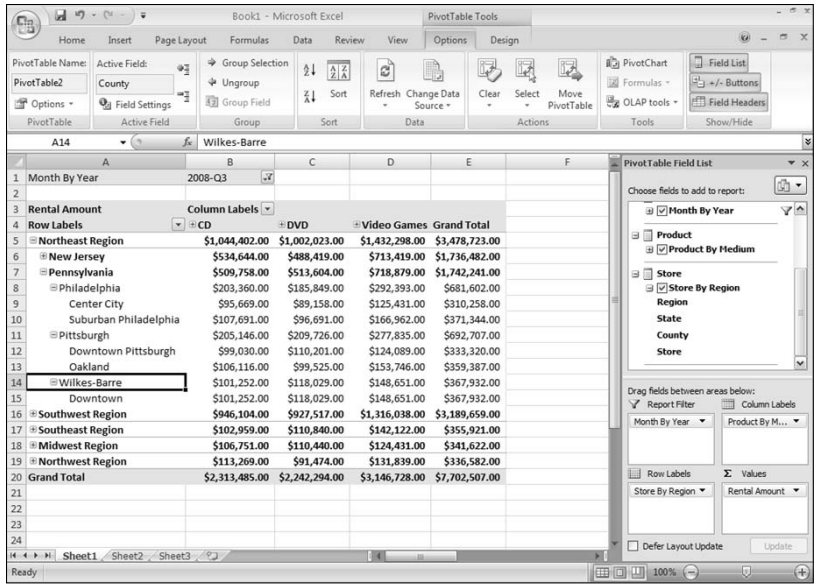


Figure 10-4: By drilling down to the lowest level of detail, you can see quarterly sales by product and by store.

Drill-down analysis can be selective. Suppose that your particular area of interest is the northeast region, but you're filling in for the analyst who usually handles Arizona because that person's on vacation. You can use drill-down on only the portions of the total result set, as shown in Figure 10-5.

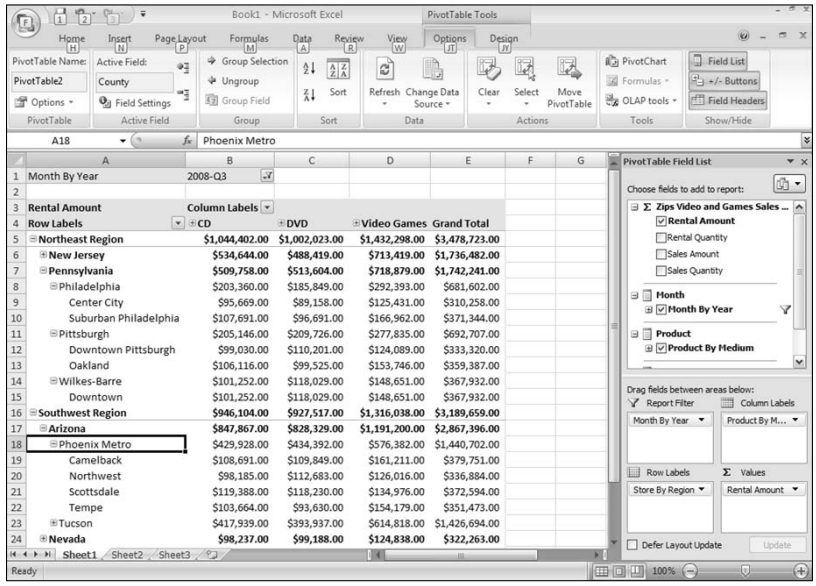


Figure 10-5: You can pick and choose the level of detail you want to see in mix-and-match mode.

What's so great about this drilling stuff?

"Big deal," you might be thinking. "I can run reports with varying levels of detail in the query tool I've been using for years. What's so wonderful about this drill-down and drill-across business?"

The major advantage of business analysis (OLAP) drilling capability, as compared to traditional methods of getting this information, is that basic querying and reporting tools usually have had to run separate database access queries for each level of detail (often by using the SQL GROUP BY clause and along with an associated SQL WHERE clause). Each run is a separate SQL statement issued to the database, a separate pass through the database, a separate return of all the requested data, and a separate formatting of the results.

Multidimensional analysis and its drilling capability, on the other hand, are instantaneous because the information you need is staged for you. By clicking the mouse or selecting a command, you see less detail, more detail, or whatever you want. The tool and the database don't have to collaborate for successive data access requests — it's all there for you.

Hint: If you haven't used a drill-down feature and want to get a feel for it, try using the HIDE/UNHIDE features for rows and columns in your spreadsheet program. Set up a set of detailed rows of data, total them into another row, and then do the same thing again. When you HIDE the detail rows, you're performing a drill-up function; when you UNHIDE them, you're drilling down.

As mentioned in Chapter 9, some reporting tools now have business analysis (OLAP) drill-down capabilities, which blurs the distinction between members of these two classes of business intelligence tools.

Drill-up

Drill-up analysis is, not surprisingly, the exact opposite of drill-down analysis. Using a detailed report, such as the one shown earlier in Figure 10-5, you can drill up to group the results from all the stores in the northeast region to the city level, and then all the cities to the state level, and eventually all the states to the regional level.

Drill-across

Drill-across analysis takes some value on a horizontal axis (in the example in Figures 10-1 through 10-5, the type of product) and receives an additional level of detail. Suppose that you want to see why Nintendo Wii sales are so high in your New Jersey stores (not that you're complaining). Figure 10-6 shows how to get this increased level of detail by doing a drill-across function.

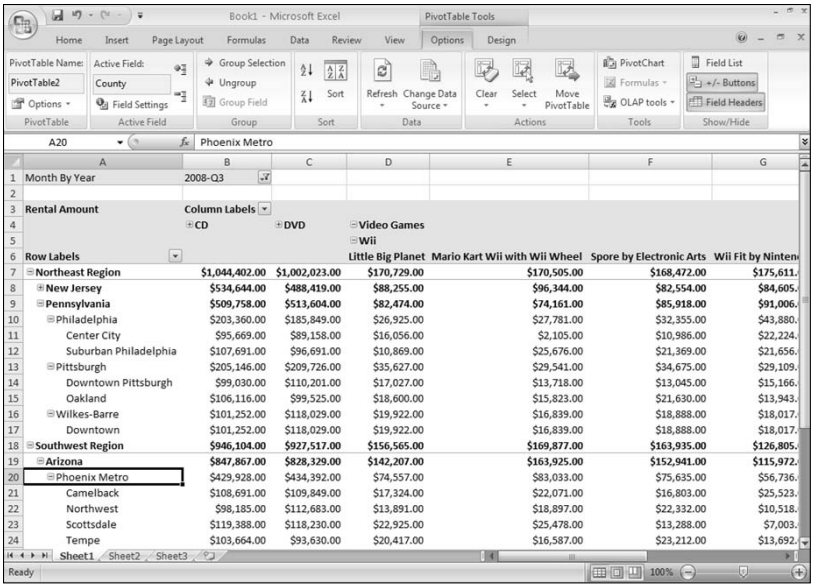


Figure 10-6: Drill-across analysis, combined with drill-down, can give you a fix on problem areas.

Drill-through

When you use drill-through analysis (sometimes referred to as reach-through analysis), you get as much detail as possible from your data. You set up the data warehousing environment (ideally, in a behind-the-scenes manner) so that users can access additional levels of detail from a supplemental database. In a typical scenario, you get the types of reports shown in the figures in this chapter from a data mart and then request to see daily sales by store, perhaps further divided by hour. For example, your environment drills through to a main relationally based data warehouse that has a large store of detailed data, pulls out the information that applies to your request, and sends it back to you.

It's still early in the game for drill-through technology. Check out capabilities such as drill-through carefully, especially if you're looking at multivendor, multiproduct solutions. Because users frequently request this feature, expect to see increased capabilities in this area leveraging Web services that support multivendor, multiproduct drill-through.

Pivoting

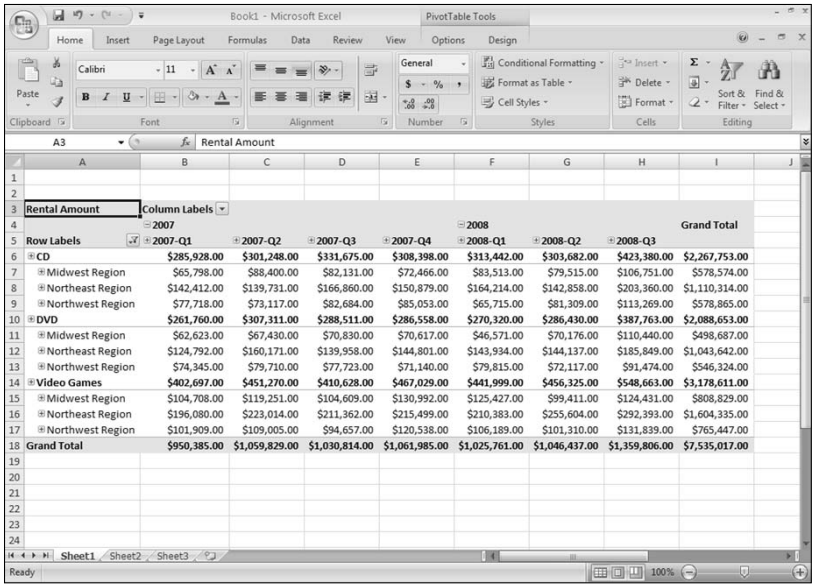
Another business analysis (OLAP) capability is pivoting. You can rearrange the look of an on-screen report (drag spreadsheet columns over to become rows, for example) by using a few drag-and-drop mouse operations.

Trending

One of the strengths of multidimensional business analysis is that you can perform trending of key information relatively easily when time is one of the dimensions (which it usually is). You can see, for example, how sales data, market share, units sold, or any other measurement changes over time without having to go through a number of gyrations in your database model to support this capability.

Nesting

The ability to present more than one dimension on either a row or column is often described as *nesting*. If you want to see the trend in product sales across regions, you can place product on the rows, nesting the region within the product rows, as shown in Figure 10-7.



Rental Amount	Column Labels							
	2007				2008			Grand Total
Row Labels	2007-Q1	2007-Q2	2007-Q3	2007-Q4	2008-Q1	2008-Q2	2008-Q3	
CD	$285,928.00	$301,248.00	$331,675.00	$308,398.00	$313,442.00	$303,682.00	$423,380.00	$2,267,753.00
Midwest Region	$65,798.00	$88,400.00	$82,131.00	$72,466.00	$83,513.00	$79,515.00	$106,751.00	$578,574.00
Northeast Region	$142,412.00	$139,731.00	$166,860.00	$150,879.00	$164,214.00	$142,858.00	$203,360.00	$1,110,314.00
Northwest Region	$77,718.00	$73,117.00	$82,684.00	$85,053.00	$65,715.00	$81,309.00	$113,269.00	$578,865.00
DVD	$261,760.00	$307,311.00	$288,511.00	$286,558.00	$270,320.00	$286,430.00	$387,763.00	$2,088,653.00
Midwest Region	$62,623.00	$67,430.00	$70,830.00	$70,617.00	$46,571.00	$70,176.00	$110,440.00	$498,687.00
Northeast Region	$124,792.00	$160,171.00	$139,958.00	$144,801.00	$143,934.00	$144,137.00	$185,849.00	$1,043,642.00
Northwest Region	$74,345.00	$79,710.00	$77,723.00	$71,140.00	$79,815.00	$72,117.00	$91,474.00	$546,324.00
Video Games	$402,697.00	$451,270.00	$410,628.00	$467,029.00	$441,999.00	$456,325.00	$548,663.00	$3,178,611.00
Midwest Region	$104,708.00	$119,251.00	$104,609.00	$130,992.00	$125,427.00	$99,411.00	$124,431.00	$808,829.00
Northeast Region	$196,080.00	$223,014.00	$211,362.00	$215,499.00	$210,383.00	$255,604.00	$292,393.00	$1,604,335.00
Northwest Region	$101,909.00	$109,005.00	$94,657.00	$120,538.00	$106,189.00	$101,310.00	$131,839.00	$765,447.00
Grand Total	$950,385.00	$1,059,829.00	$1,030,814.00	$1,061,985.00	$1,025,761.00	$1,046,437.00	$1,359,806.00	$7,535,017.00

Figure 10-7: Nesting enables you to present more than one dimensional descriptor on a row or column of a report.

Visualizing

Today's business analysis (OLAP) tools offer a robust set of visualization capabilities. These capabilities include cross-tabs, as shown in Figures 10-1 through 10-7, as well a graphics, such as the line graph shown in Figure 10-8. Data and trends often jump out when you change the style of visualization. And you can perform most of the features described in the preceding sections, regardless of the visualization style you choose.

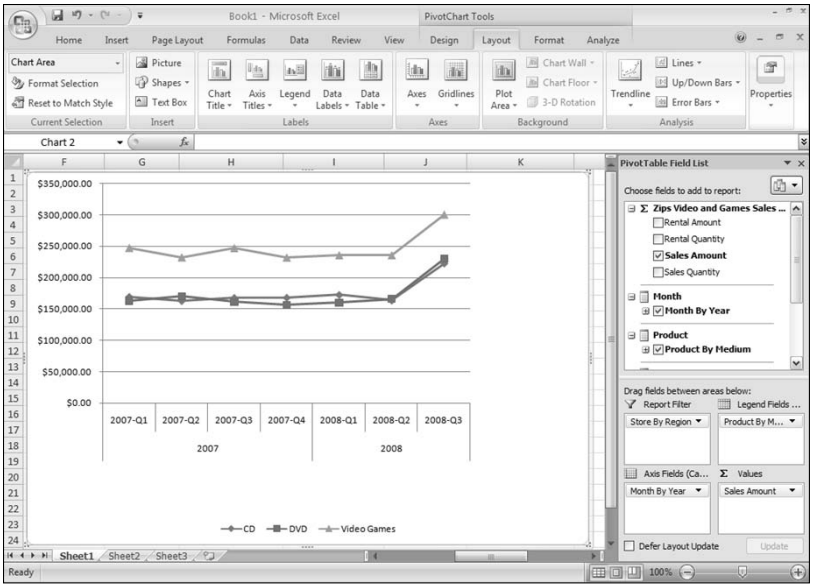


Figure 10-8: Leverage in-line chart visualization to better spot trends.

Data Warehousing Business Analysis Vendors

The following sections describe some vendors that provide business analysis (OLAP) products and where you can find them on the Web. Check out these vendors' products (and those from other companies).

IBM

```
www.ibm.com
www.cognos.com
```

In January 2008, IBM acquired Cognos Corporation, increasing IBM's capabilities in the area of business analysis (OLAP). Cognos brought several business analysis (OLAP) products to IBM, including PowerPlay, Cognos8 Analysis Studio, and TM1. The PowerPlay engine is predominately leveraged for its DOLAP and MOLAP storage. The TM1 engine is predominately leveraged for high-volume solutions or dynamic ROLAP capabilities. And Cognos8 Analysis Studio is emerging as their business analysis visualization tool.

MicroStrategy

```
www.strategy.com
```

MicroStrategy produces a suite of ROLAP products that are part of its business intelligence platform. MicroStrategy OLAP Services is an extension of the MicroStrategy Intelligence Server that allows MicroStrategy Desktop, Office, and Web users to manipulate Intelligent Cubes™, a finite set of report objects that are automatically or dynamically populated by MicroStrategy's multi-pass SQL generation engine. Users can analyze the full depth and breadth of their data warehouse by using OLAP Services.

Oracle

```
www.oracle.com
```

Oracle, like many other enterprise vendors, has acquired a number of products in the area of business analysis. These products include Express from IRI, which has been embedded into the Oracle database platform, and (most recently) Hyperion Essbase, a MOLAP/MDDB environment which evolved from Arbor in the 1990s. Oracle is integrating these products into Oracle Business Intelligence (OBI), a middleware offering rebranded from the acquired Seibel Analytics Platform. Additionally, Oracle inherited Hyperion Intelligence, the Brio tools of which were integrated into the Hyperion suite.

Pentaho

```
www.pentaho.org
```

Pentaho is the commercial open source business intelligence (BI) solution. Pentaho Analysis combines the Mondrian OLAP server with JPivot visualization to enable users to perform business analysis (OLAP). Users who want powerful capabilities that they can embed into applications seamlessly, or who aren't focused on all the bells and whistles of the more mature business intelligence providers, might find Pentaho (or other open source BI solutions) a viable alternative.

SAP

```
www.sap.com
www.businessobjects.com
```

SAP, though initially focused on "run the business" applications, has grown over the years to offer more capabilities in the area of business intelligence. In 2008, SAP acquired Business Objects, which had previously acquired Crystal Decisions. With these acquisitions, SAP now possesses several business analysis products, including Business Objects XI, Crystal Analysis, SAP/BW, and Business Explorer (BEx). The strength of SAP's legacy offerings include the ability to activate the solution orientation, or content, with the "flip of a switch." With Business Objects in the fold, SAP also now has a general-purpose tool that can enable business analysis.

SAS

```
www.sas.com
```

SAS provides a multidimensional data store, known as the SAS OLAP Server, that's designed to provide quick access to pre-summarized data. Although SAS is often known for its roots in more technical, statistical analysis, it has grown its BI offerings so that it can move more into the business analysis arena with such products.

Chapter 11

Data Mining: Hi-Ho, Hi-Ho, It's Off to Mine We Go

In This Chapter

- Figuring out how you use data mining
- Discovering data mining techniques
- Keeping data mining stats by using algorithms
- Finding data mining products

The distinguishing characteristic about data mining, as compared with querying, reporting, or even OLAP, is that you can get information without having to ask specific questions.

Data mining serves two primary roles in your business intelligence mission:

- **The "Tell me what might happen" role:** The first role of data mining is predictive, in which you basically say, "Tell me what might happen." Using hidden knowledge locked away in your data warehouse, probabilities and the likelihood of future trends and occurrences are ferreted out and presented to you.
- **The "Tell me something interesting" role:** In addition to possible future events and occurrences, data mining also tries to pull out interesting information that you probably should know about, such as a particularly unusual relationship between sales of two different products and how that relationship varies according to placement in your retail stores. Although many of these interesting tidbits are likely to exist, what questions would you ask if you were using a querying or OLAP tool, and how would you interpret the results? Data mining assists you in this arduous task of figuring out what questions to ask by doing much of the grunt work for you.

Data Mining in Specific Business Missions

Data mining is particularly suited for these specific types of business missions:

- Detecting fraud
- Determining marketing program effectiveness
- Selecting whom, from a large customer base or the general population, you should target as part of a marketing program
- Managing customer life cycle, including the customer retention mission
- Performing advanced business process modeling and what-if scenarios

Think about what's behind each of the business missions in the preceding list:

- A large amount of data
- An even larger number of combinations of various pieces of data
- Intensive results set analysis, usually involving complex algorithms and advanced statistical techniques

Now, think about what you would have to do if you were using a reporting or OLAP tool to accomplish these missions. You'd find it virtually impossible to thoroughly perform any of the preceding missions if you had to ask a question and get a result, ask another question and get another result, and then keep repeating those steps.

Data Mining and Artificial Intelligence

If you've been in the information technology (IT) field for at least a decade, some of the preceding terms might sound vaguely familiar. Unlocking hidden knowledge? Predictive functionality? Wait a minute — that's artificial intelligence!

From the earliest days of commercial computing, there has been a tremendous interest in developing "thinking machines" that can process large amounts of data and make decisions based on that analysis. Interest in artificial intelligence (AI) hit its zenith in the mid-1980s. At that time, database vendors worked on producing knowledge base management systems (KBMSs); other vendors came out with *expert system shells,* or AI-based application development frameworks that used techniques such as forward-chaining

and backward-chaining to advise users about decisions; and neural networks were positioned as the next big AI development. Interest in AI waned in the early 1990s, when expectations exceeded available capabilities and other frenzies, such as client/server migration and (of course) data warehousing, took center stage.

Now, AI is back!

The highest-profile AI technique used in data mining is neural networks. Neural nets were originally envisioned as a processing model that would mimic the way the human brain solves problems, using neurons and highly parallel processing to do pattern solving.

Applying neural network algorithms to the areas of business intelligence that data mining handles (again, predictive and "tell me something interesting" missions) seems to be a natural match.

Although the data mining/neural network game is definitely worth checking into, you should do it carefully. You can find a lot of interesting and exciting technologies that, in the hands of those who don't understand the algorithms, will likely fail. However, with proper knowledge and education, you can make a full-scale commitment to bringing this type of processing into your business intelligence framework as the technical-analysis pairing for the OLAP-focused business analysis.

Data Mining and Statistics

The more mature area of data mining is the application of advanced statistical techniques against the large volumes of data in your data warehouse. Different tools use different types of statistical techniques, tailored to the particular areas they're trying to address.

Without a statistical background, you might find much of data mining confusing. You need to do a lot of work to train the algorithms and build the rules to assure proper results with larger datasets. However, assuming that you're comfortable with this concept, or have a colleague who can assist, here are some of the more widely leveraged algorithms:

- **Classification algorithms:** Predict one or more discrete variables, based on the other attributes in the dataset. By using classification algorithms, the data mining tool can look at large amounts of data and then inform you that, for example, "Customers who are retained through at least two generations of product purchases tend to have these characteristics: They have an income of at least $75,000, and they own their own homes."

- **Regression algorithms:** Predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset. Regression algorithms are driven through historical information presented to the data mining tool "over time," better known as *time series* information.
- **Segmentation algorithms:** Divide data into groups, or clusters, of items that have similar properties.
- **Association algorithms:** Find correlations between different attributes in a dataset. The most common application of this kind of algorithm creates association rules, which you can use in a market basket analysis. Note that, for example, if a customer purchases a particular software package, he or she has a 65-percent chance of purchasing at least two product-specific add-on packs within two weeks.
- **Sequence analysis algorithms:** Summarize frequent sequences or episodes in data, such as a Web-path flow.

Many more methods exist. Dust off that old statistics book and start reading.

Some Vendors with Data Mining Products

The following sections detail vendors that sell data mining products.

Microsoft

```
www.microsoft.com
```

Microsoft introduced server-side data mining with Microsoft SQL Server 2005. Although it's not as mature and sophisticated as SAS and SPSS, Microsoft has proven over time their ability to simplify and generalize technology offerings. Their integration of common data mining routines into SQL Server Analysis services provides an interesting, cost-effective solution. In addition, you can use several algorithms within Microsoft Excel when you access your data warehouse or data mart.

SAS

```
www.sas.com
```

SAS, one of the two industry leaders in analytics, provides an integrated environment for predictive analytics and descriptive modeling, data mining, text mining, forecasting, optimization, simulation, experimental design, and more. Their analytic solutions provide a range of techniques and processes for the collection, classification, analysis and interpretation of data to reveal patterns, anomalies, key variables, and relationships — leading ultimately to new insights for guided decision making. SAS is one of the more mature vendors and a leader in this market segment.

SPSS

```
www.spss.com
```

SPSS is one of the two industry leaders that have focused heavily in the area of statistics and data mining. Through acquisitions and internal development, SPSS has built out a comprehensive set of tools to help you with predictive modeling through Clementine and various statistical algorithms through SPSS Statistics.

Chapter 12

Dashboards and Scorecards

In This Chapter

- Getting a quick look at important data with dashboards and scorecards
- Understanding the relationship between dashboards, scorecards, and business intelligence
- Keeping tabs on key indicators for executives
- Using briefing books and portal command centers to summarize reports and data
- Finding quality dashboard and scorecard products

Think about how a child learns to read. You begin by reading a picture-oriented book to the child, one with a short sentence at the bottom of each page ("See the bunny eat lettuce!"). Pretty soon, the child figures out how to read the book without your help. The child turns the pages, occasionally squealing in delight because of that favorite picture coming up on the next page.

Now, think about how behavioral researchers work with dolphins or chimpanzees to better understand how they think. When the chimp pushes a button, a banana drops down; when the dolphin presses a lever with a flipper, it's rewarded with a dolphin treat (whatever that is).

These are the principles behind dashboards and scorecards. Wait a minute! Stop that booing and hissing, and put down those rotten vegetables! I'm not trying to insult anyone! I'm very serious. Dashboards and scorecards are intended to be a ridiculously easy way to provide online business intelligence to people who are, uh, "too busy" to figure out how to use the full complement of business intelligence tools.

Dashboard and Scorecard Principles

The fundamental principle behind dashboard and scorecards is, "Tell me a lot of things, but don't make me work too hard." Despite the best efforts of vendors, despite all the human factors and usability research that's gone

into business intelligence products, and no matter how much training you provide to your data warehousing users, you always have someone who doesn't grasp the concept of painting a report screen, doing a drill-down analysis, or taking full advantage of the power available from today's tools.

Or these folks might think they're too busy to figure it all out. Believe it or not, the mentality that computers are for "clerical types" still pervades many of corporate America's executive suites. Most of these folks grudgingly accept delivery of that brand-new, supercharged laptop (although they still refuse to type their own letters) and want to do only "a couple of things" with the computer.

Should those people be shut out of the world of business intelligence in your data warehousing environment? No!

In most cases, you need to create an environment that has a set of dashboards and scorecards. Dashboards and scorecards, like other areas of business intelligence, predate the data warehousing era — evolving from the executive information systems (EIS) of the 1980s and 1990s. Alas, like early multidimensional analysis (pre-OLAP OLAP), no one realized the full power of an EIS at the time, and the EIS faded to the background while full-scale business intelligence solutions took hold along with data warehousing.

Dashboards

A *dashboard* is a collection of graphs, reports, and KPIs that can help monitor such business activities as progress on a specific initiative. Everyone who's driven a car has seen a dashboard. A dashboard (whether in your car or on your computer) provides a lot of information in a summary form to show you how you're currently performing in an operational manner. Your car dashboard shows you how fast you're currently going, and you have to evaluate whether this value falls within the target speed limit. Alongside this information, you can find out what your fuel level is, how many miles you've traveled, what the temperature of the engine is, and . . . well, whatever that tachometer thing is supposed to tell you.

A business intelligence dashboard is the equivalent of your car's dashboard. In your data warehousing environment, this business intelligence dashboard shows users the effectiveness of operations. You can build a dashboard that supports a wide variety of users, from individuals to the company as a whole. As shown in Figure 12-1, when built correctly, dashboards can give you quick insight into company performance.

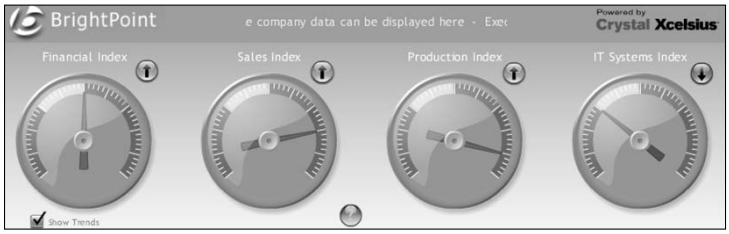


Figure 12-1: Dashboards monitor the performance of operational processes.

Scorecards

A *scorecard* is a visual representation of your company's strategy. A scorecard helps make it easy to take critical metrics and map them to your strategic goals throughout the organization. Scorecards offer a rich, visual gauge that everyone in your organization can reference in order to see

- The performance of specific initiatives, business units, or the company as a whole
- Individual goals in the context of larger corporate strategy

Scorecards, as shown in Figure 12-2, distill information into a small number of metrics and targets. With a scorecard, your users get an at-a-glance perspective of information. Scorecards are designed to increase productivity by allowing users to stop sifting through stacks of reports to find what's right or wrong with the business in relationship to the corporate strategy. Scorecards show users immediately how the company, a division, their team, or they themselves are performing against targets that were set within the overall strategy or plan.

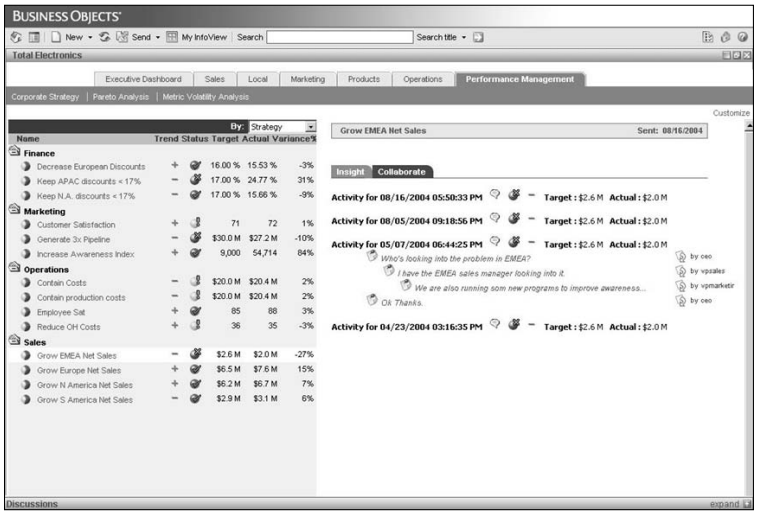


Figure 12-2: Scorecards show performance against goals and strategies.

The Relationship between Dashboards, Scorecards, and the Other Parts of Business Intelligence

Dashboards and scorecards are often linked to results from other business intelligence tools, representing a presentation mechanism, rather than an analytical mechanism. For example, you might create the "pages" of a briefing book (discussed in the section "The Briefing Book," later in this chapter) from the results of various standardized reports (run from the reporting tool) in addition to a rudimentary multidimensional analysis from the organization's OLAP tool that's ready for inclusion in the briefing book.

Often, geographical information system (GIS) capabilities, as described in Chapter 8, are included with dashboards and scorecards. A dashboard that has GIS integration, for example, might present a series of standardized views of important information to a user by using maps or other geographical tools as the primary interface. The user then can take a cursory look through each screen of information; if everything looks okay, the user can continue to the next page, or if something looks askew, he or she can perform GIS operations, such as double-clicking a map to access the underlying data.

Dashboards and scorecards aren't just for executives. Almost any user in an organization can have a dashboard or scorecard. You might find presenting everyone in your organization dashboards and scorecards preferable to giving everyone the "official" business intelligence tool and having three-quarters of the group never use it.

Don't assume that you automatically need to give anyone classified as an executive (above a certain level) in your organization a dashboard or scorecard to the exclusion of other tools. Many a computer-savvy executive can wind up as the primary user of OLAP or a reporting tool. In addition, don't assume that users access their dashboards and scorecards only from a Web browser. Blackberries, iPhones, and other smart phone devices are quickly becoming the preferred delivery platform for the busy executive.

EIS and Key Indicators

In keeping with the philosophy of "Don't make me work too hard to get my information," many dashboards and scorecards are built around the concept of key indicators. An executive might have, for example, a handful of (seven

or eight) items that are monitored on a weekly basis and that represent the pulse of the organization. If all these items fall into the expected range (whatever that means for each measure), he or she doesn't have to worry about the business and can go off for a quick nine holes of golf during a long lunch break.

On the other hand, an indicator that's out of whack likely means that the user should do a little digging — or maybe a great deal of digging — in that particular area.

A quick snapshot of key indicators (ideally, all on a dashboard or scorecard) can therefore be a valuable part of a well-organized enterprise portal environment.

Customize your dashboard and scorecard implementations! One executive's key indicators aren't likely to be the same as another's. Make dashboards and scorecards personal, an environment for communication — not an unvarying, rigid set of screens installed on several users' respective browsers.

During the scope phase of your project, spend enough time working with likely users to determine each person's key indicators. Explore other possibilities and show examples; in the end, however, let users decide which indicators they want to see and how they want those indicators to look.

The Briefing Book

If you've ever used a presentation program (such as Microsoft PowerPoint) that contains a slide show feature, you have a basic understanding of the briefing book concept. A briefing book is usually constructed to flow from one screen to another, covering key information, indicators, and other pertinent data in a relatively predictable manner.

You can design a briefing book to have an up-front screen or indicator that provides an overall assessment: "All is well" or "Update your résumé," for example. Then, each successive screen displays the items important to that user.

One interesting aspect of a well-architected briefing book is that you must strike a balance between not enough detail and too much detail in the information presented to a user. For example, include a browser link or button labeled More Information that the reader can click to access additional detail. A large part of the simplicity of the briefing book, however, is that the flow from scorecard to dashboard to analytical view is predictable and not too complex, as described in the section "The Relationship between Dashboards,

Scorecards, and the Other Parts of Business Intelligence," earlier in this chapter. Therefore, make users' More Information navigation paths access only one more visualization, perhaps two. If the user needs additional information, he or she might need assistance getting it. Don't lose the flow of the briefing book by trying to make the environment overly elegant and flexible.

The Portal Command Center

The portal command center is a type of dashboard and scorecard that, in one way, is less predictable than a briefing book environment but, in another way, is just as predictable.

Huh?

The less predictable aspect is that, unlike the sequentially oriented nature of a briefing book, a command center user can choose whether to check out a particular set of information. If the user's in-browser scan of visual indicators shows that all is well, no more information access occurs.

In the just as predictable part, though, each portal command center user has an environment tailored to a set of reports or sets of data that are important to that particular user. A sales executive, for example, might have on his or her portal command center one link for each region the person is responsible for, including a visual indicator that provides a quick look at the state of that region. The visual indicator might be a stoplight, for example: green for "Everything's okay," yellow for "Better check this out before things get worse," and red for "Uh-oh."

Collaborative portal products that are leveraged to build a portal command center interface are tailor-made for multimedia business intelligence environments (see Chapter 25). For example, certain links on a portal command center can lead to posted results from standard OLAP-generated reports on the company's intranet. Other links can lead to the company's latest ads, product diagrams, training videos, or other elements. You can make your presentation of key business intelligence perspectives by using dashboards and scorecards just like a spiffy Web site home page with both structured and unstructured data underneath.

You can also use command centers operationally, rather than analytically. Sometimes, the distinctions between the two are subtle. In an operational command center, an executive usually monitors the environment regularly (daily or perhaps more frequently) as an active part of operational decision-making; analytical command centers usually are usually accessed less frequently (weekly or monthly, for example), with the purpose of determining "How did we do?" rather than "What do I have to do?"

Who Produces EIS Products

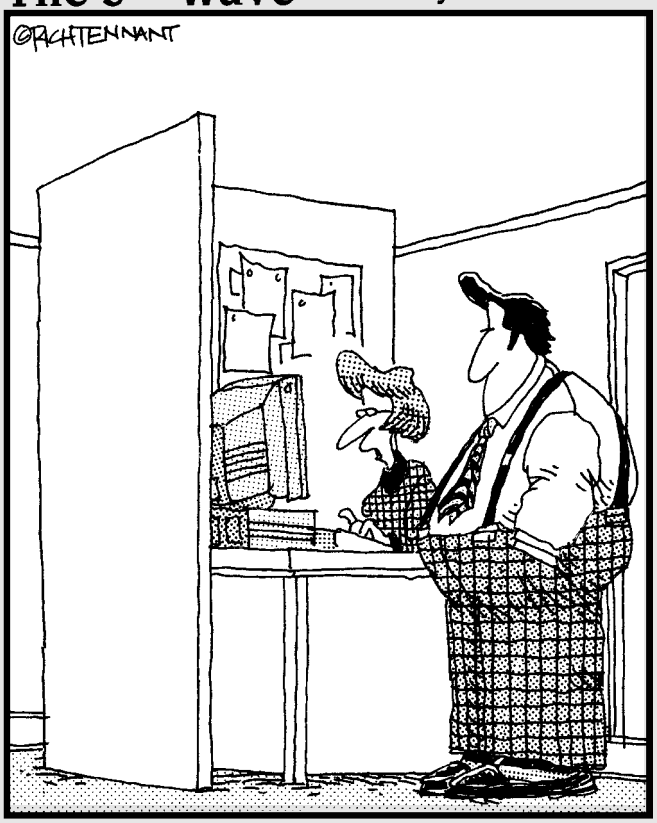
Because most business intelligence suites now have dashboard, scorecard, and portal components to them, I recommend checking out these components when you research business intelligence tools in these other areas.

Alternatively, you can use Internet technology to develop your own presentation and visualization environment as part of a Web-enabled business intelligence framework for your users. You can develop a home page, for example, that uses hyperlinks to guide users through briefing books or to enable them to navigate among command-center-driven capabilities.

Part IV

Data Warehousing Projects: How to Do Them Right

The 5th Wave By Rich Tennant



"Look you've got Project Manager, Acct. Manager, and Opportunity Manager, but Sucking Up to the Manager just isn't a field the program comes with."

In this part . . .

A data warehousing project is just like any other application development project. Or, to be more blunt, you can mess up a perfectly good data warehouse effort in a lot of ways: poor project management, putting the wrong people on the development team — you get the idea.

If you want to find out how to do a data warehouse project the right way, this part of the book is for you! I uncover what aspects of developing data warehouses are the same as traditional applications and what few subtleties can alter your approach to building a data warehouse from traditional applications.

Chapter 13

Data Warehousing and Other IT Projects: The Same but Different

In This Chapter

- Comparing data warehousing projects to other application development efforts
- Looking into some issues that come with secondhand data
- Getting someone at the top to support your data warehousing project
- Keeping a large data warehousing project on track

Psst! Yes, you. Do you want to know a secret? No, this isn't the Beatles song; this secret is about data warehousing projects and how you can almost guarantee success.

I thought that would get your attention! Listen closely because I sum up this secret in three sentences:

- Data warehousing projects are remarkably (about 95 percent) similar to any other application development project.
- The 5 percent that's different is because of two key items: an unclear method for identifying requirements and scope, and the reliance on data from other applications' databases and files (as discussed in Chapter 1).
- By applying your organization's application development "best practices" to the 95-percent similar portion (as though this project is just like any other) and by following a few guidelines to handle the other 5 percent, you can almost certainly develop your data warehouse successfully.

Why a Data Warehousing Project Is (Almost) Like Any Other Development Project

Way back in the mid- and late 1970s, when disco reigned supreme, IT professionals realized that certain things made sense and other things didn't in developing computer applications and systems. One of the things that made sense was that before choosing hardware, off-the-shelf products (such as DBMSs), and even programming languages, you first had to determine the business requirements you were trying to satisfy and then take a number of steps to specify and design the programs you plan to develop. Then, after gaining a good understanding of these details, you could make an informed decision about your hardware and development software.

Then, in the early 1990s, a tremendous, growing interest in data warehousing emerged. I can't tell you how many times I sat in a room with clients, discussing their data warehousing projects, when someone (not me!) said, "I think that we want to use Brand X business intelligence tool and the Brand Z extraction, transformation, and load (ETL) product." They made this choice before they'd even begun to analyze their business requirements, let alone specify their architecture or do any design work.

The lessons from the 1970s seemed to have disappeared. You don't even have to look at IT projects to see these lessons. I recently asked a customer of mine in the energy and utilities business whether his company would build a nuclear power plant by using a technique that involved picking the technology first and then determining the requirements — of course, he almost shouted, "No!" When I asked him why, he explained that you must understand the requirements, properly design the architecture, and then — only then — begin selecting the technologies and building the finished product, or solution, to the specifications you established.

Some of the application development revelations made in the '70s (and remade in the '90s) about choosing hardware and software that are appropriate for the business problem at hand still make a great deal of sense for any type of project — even a data warehousing project.

Maybe premature software purchases were made because of the data warehousing vendors and their product hype: The phrase, "It slices! It dices!" is appropriate for business analysis software found in OLAP (refer to Chapter 10). Rather than try to find someone to blame, however, how about if everyone

agrees to make a fresh start? No premature selection of hardware, database software, business intelligence tools, middleware products, or any other products without first concentrating on business requirements and then doing appropriate analysis and design work.

How to Apply Your Company's Best Development Practices to Your Project

I hope that your company has jumped on the application development methodology bandwagon and has published a set of guidelines that you're supposed to use in developing applications. The Sarbanes-Oxley (SOX) Act of 2002 essentially mandated that large companies publish guidelines. If you happen to be with an enterprise that hasn't clearly defined a development method, read this section anyway because it describes how your company is supposed to develop applications.

The current conventional wisdom says that no matter how large or complex the business problem for which you're developing an application, you should, if at all possible, divide that problem into chunks, each of which you can deliver in a manageable, relatively short time (typically, three to nine months). By using *agile* methods of delivery, you often can deliver solutions to these problems in closer to three months.

A number of methodologies exist based on this philosophy of manageable chunks. Methods that are well documented by industry thought leaders such as Ralph Kimball, Larissa Moss, Dan Linstedt, and others fall in line with such a philosophy.

My company, Balanced Insight Incorporated, uses an approach that's oriented toward extremely fast delivery of results (the Information Packaging Method), with each project (and each project chunk) divided into several iterations that cross very traditional phases:

- **Requirements:** Capture the business needs for information including how the user community desires to navigate the information in the final solution. This is done by clearly defining the scope through a series of rapid solution workshops:
  - *Scope:* All relevant people reach consensus about what you're going to build, why you should build it, and other factors.
  - *Rapid solutions workshop (RSW):* Visible results are delivered in a few days to a few weeks, leveraging tools such as Balanced Insight Consensus.

- **Design:** All the technical details are decided to make sure that development (the next phase) goes smoothly and delivers an application that meets the business requirements.
- **Development and testing:** Databases are created, logic is developed to move data from the various sources to the new destination, the user access semantic layer is built out, and maybe pre-built views of the data are developed in the business intelligence platform if the user population is inexperienced.
- **Deployment:** The customer delivers the application to the users, and they begin using it.

For my company's data warehousing projects, we follow the same methodology that we use for building our product set — Balanced Insight Consensus, which is an Agile development method.

Agile methodologies generally promote

- A project management process that encourages frequent inspection and adaptation
- A leadership philosophy that encourages team work, self-organization, and accountability
- A set of engineering best practices that allow for rapid delivery of high-quality software
- A business approach that aligns development with customer needs and company goals

Although the data warehousing version of the methodology is tailored to handle the transfer of data from the source systems into the data warehouse, Balanced Insight's employees engage the end user in frequent inspection of the target solution. They begin, for example, by working with clients on the project scope that is leveraged to build the business case for the data warehouse. They collect requirements and help the client determine what business value to expect from the data warehouse by simulating what the final target application will deliver. We don't sit down with a client in the first meeting and, after hearing that they're interested in developing a data warehouse, tell them, "Okay, you should use Brand X OLAP tool and the Brand Z extraction-and-transformation product. Now, what do you want this data warehouse to do?" As shown in Figure 13-1, we clearly gain an understanding of what data the client needs so that they can understand their business better, and we also figure out how the user wants to navigate, traverse, and integrate the data.

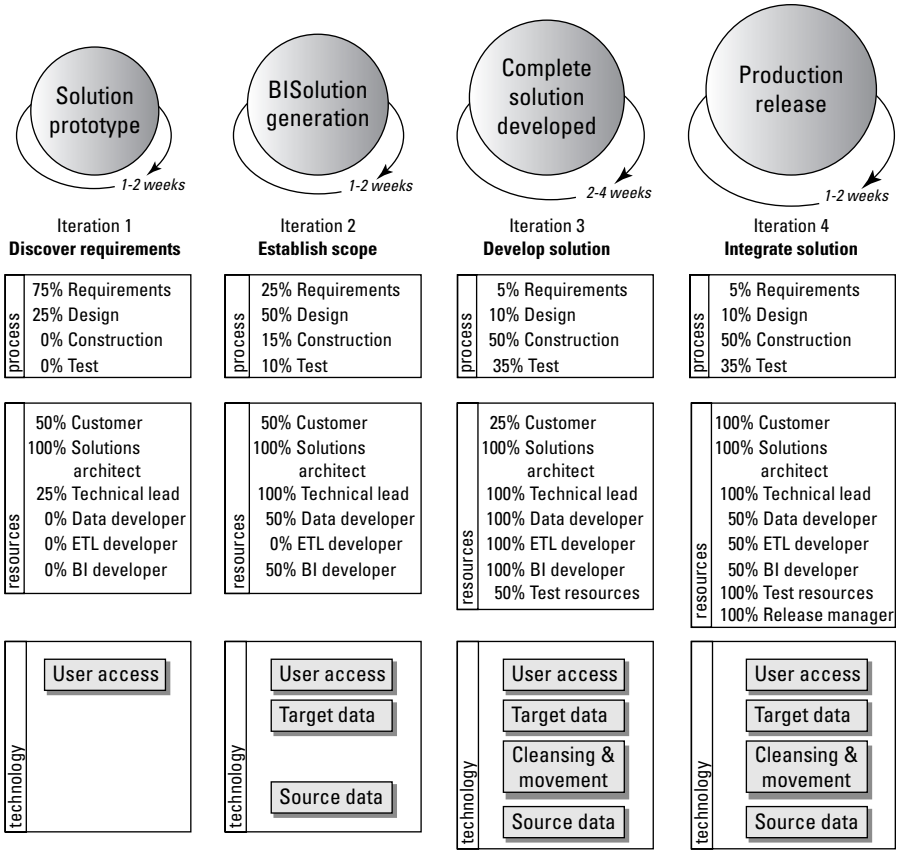


Figure 13-1: Agile delivery of data warehousing enables you to deliver data assets in the most cost-effective and highest quality manner.

Properly defining scope within the boundaries of the business's need is the ultimate key to success. Data warehousing teams often overlook this step, leading to numerous failures. Performing a *Field of Dreams*–oriented project — the "If you build it, they will come" method — fails more often than it succeeds.

If your company's application development methodology works (if you usually deliver projects on time and on budget), use that methodology as the foundation for your data warehousing project and make adjustments (such as the ones described in the following section) to handle the unique properties of data warehousing.

How to Handle the Uniqueness of Data Warehousing

Your company's standard application development methodology probably doesn't cover how to handle all the issues that come with integrating data from a plethora of applications that were never intended to integrate. Most run-the-business applications create data in their databases as a result of transactions (a customer making a savings account deposit, for example, or someone placing an order for season tickets to Pittsburgh Steelers games). As long as the application's internal processes that are responsible for creating, storing, modifying, and retrieving the data are correct, you most likely don't have to worry about data problems or data integration problems within your application.

The following steps, which are for your warehouse's data environment, differ from the steps in a traditional application development project. The project phase is listed in bold, followed by a brief explanation of how the task is different from a traditional application.

Requirements

1. **Identify the key questions.**

   Working with the business users, identify key business questions and declarative statements that require information to assist the business in making decisions.

2. **Define the focused answers.**

   Within a workshop, analyze the questions and statements that have been defined to determine the measurement answer sought by the business.

3. **Define the key business objects.**

   Within a workshop, pull the key words out of the business questions and declarative statements that appear to be the descriptors of the business — called *business objects* (not to be confused with the vendor product!).

4. **Determine usability requirements.**

   Working with the business users, determine how they want to navigate the data — including how they want to drill into details, what they need from an aggregation perspective, and other key reporting requirements.

5. **Prototype the solution.**

   Using a tool such as Balanced Insight Consensus, prototype the requirements so that the users can see what they asked for early and clarify their requirements before you get the expensive team working on the effort.

Design

6. **Identify all candidate data sources.**

   Have the users identify the system that they typically use to get this information or where they enter that information. It's remarkable how good they are at this type of thing — and how challenged IT people are in this area.

7. **Prioritize data sources.**

   Most applications don't use external data. Because some applications do need the data for processing, no one does prioritization: The application won't work without the data.

8. **Obtain several data extract samples.**

   This step is more important than in a traditional application, even if you use external data.

9. **Quickly create a database design.**

   You must orient the design toward the type of business intelligence the warehouse will support (reporting and querying, and OLAP, for example) rather than make it a general purpose design.

10. **Populate the database design with extracted data.**

    You usually don't do this step in a traditional application.

11. **Get feedback from users about whether they like the views of the data you created.**

    This step usually isn't important in a traditional application. In addition to giving you feedback on the user access strategy, users will also explain data quality issues that might not be obvious by merely looking at the raw data, including items such as data that has embedded meaning or missing values.

Development and testing

12. **Create a complete database design.**

    Depending on the complexity of your environment, you might orient the design of the data warehouse toward business intelligence, rather than toward transactions. Make your decision based on whether you're implementing a monitor-the-business solution only or also including integrate-the-business solutions (transactional integration). If you're merely doing a monitor-the-business solution, look into methods such as Ralph Kimball's. If you're performing a more robust and comprehensive solution to both integrate and monitor the business, investigate Dan Linstedt's Data Vaulting Method.

13. **Do a complete inventory of target data elements and match them to the proper source data elements.**

This step is much more extensive than when you're developing and documenting a traditional application that has external data needs, and you need business assistance to define the key business rules necessary to transform the data from the source systems to the target data warehouse consistently across the enterprise to improve the quality and confidence in the data for future use.

14. **Spend time meeting with owners of source data.**

    Although you perform this step in a traditional application that has external data needs, you have to understand the details of the data quality, availability, and dependencies much more extensively when building a data warehouse.

15. **Dig into source data.**

    Analyzing data integrity and other issues is an important part of the process (see Chapter 16).

16. **Create transformation algorithms.**

    This step is much more extensive than for a traditional application that has external data needs. Translating the business rules that have been identified within your specifications into properly coded transformation logic is complex, often involving data from more than one application.

17. **Dig into old versions of the data that might be needed in the data warehouse.**

    Look for different versions of the data and different codes, for example. This step is rarely done in a traditional application because old data isn't applicable.

18. **Select tools or design code to handle middleware tasks.**

    You usually don't have to do this step for a traditional application; for a data warehouse, do it once and reevaluate periodically to assure a consistent architecture across the enterprise.

19. **Make sure that middleware functions exist for initial warehouse loading and updates.**

    You don't usually have to worry about this step for a traditional application.

20. **Define quality assurance (QA) procedures for all source data.**

    This step is much more extensive than even a traditional application with external data needs. Data warehousing leverages much more reference data than traditional applications, where the reference data might be optional or user entered. You will need to make sure you understand how data will be investigated and tried out by the user community.

21. **Spend extensive time coding, or using tools, to extract, transform, move, and load data.**

This step is much more extensive than even a traditional application with external data needs. The need to sequence data loading is key to properly deliver high quality data. Building "one version of the truth" from numerous systems can be tricky. The logic that you build should be standard and reusable across applications that are sources for the data warehouse.

22. **Conduct preliminary tests for all the source-to-warehouse data extraction, transformation, movement, and loading.**

    This step is much more extensive than even a traditional application with external data needs. The interdependencies between applications and their associated data, as well as volume and sequencing of your data loads, can make the extraction, transformation, movement, and loading processes tricky.

23. **Conduct end-to-end performance tests.**

    You conduct a step similar to this one when performance testing in a traditional application, but this step is more complex because of the unpredictability of how users will access data.

24. **Conduct performance tests for user access to data.**

    This step involves different measurement techniques than you need for a transaction-processing application. You want to assure that the data warehouse can be loaded within a defined time period, often during off-business hours. Additionally, you want to assure that users' queries will return in the proper amount of time, defined as *response time*.

25. **Do preliminary determinations of data summary levels.**

    You usually don't have to do this step for a traditional application; only rarely do you do any kind of intra-database summarization in a traditional application. However in a data warehouse, based on performance testing, you might want to create pre-stored aggregations of the data to optimize user access and scalability.

Deployment

26. **Perform initial population of the warehouse's database.**

    This step is much more extensive than the initial population of a traditional application's database.

27. **Monitor the data warehouse reloading process.**

    This step is much more extensive than the equivalent in a traditional application. Your testing of performance during the development and testing often does not clearly reflect your production environment. Therefore, once you deploy your data warehouse, you might need to further tune the loading process to fit within defined time boundaries, such as a late-night batch window.

28. **Determine purge and archive strategies.**

    Because data warehouses have historic content, you need different strategies from those you use for traditional applications. A larger volume of data will be kept online, and users might need access to archives in quicker recovery periods than typical applications.

29. **Put plans in place to keep track of source data changes.**

    Again, this step is much more extensive than that of a traditional application. Change management routines should be altered to assure that application changes are not implemented without determining their impact to the data warehouse.

Why Your Data Warehousing Project Must Have Top-Level Buy-In

Although your data warehousing project might be the most important part of your job, you must keep in mind that others within your company probably don't share your perspective. For example

- Most of the people who operate the applications from which you acquire data probably see your project as one large nuisance that does little more than cause them to work a number of overtime hours.
- Some users who already perform rudimentary analytical tasks by using extracts from an application's database (quasi-warehouses, as discussed in Chapter 22) likely don't want to change the way they operate, even though the data warehouse can provide them with a much richer set of data.
- When you try to divide a large data warehouse project into the chunks mentioned in the section "How to Apply Your Company's Best Development Practices to Your Project," earlier in this chapter, each business organization sees its piece as the most important, and you have a battle over whose chunk is most important on your hands.

Many other issues might arise that are a result of organizational politics and have nothing to do with data warehousing technology.

This situation calls for a hero — a courageous person willing to take a stand and look at the big picture. This person has to make and enforce statements such as, "You will support this data warehousing project. Put some people on

it; if you don't have any, go hire some consultants." Or the person might say, "Hold on, this data warehouse will certainly support your group, but you'll be in the second phase of development. In the meantime, I need you to participate in evaluating business intelligence tools and put a couple of people on source data analysis." (We data warehousing consultants love people who say things like that.)

In case you haven't guessed, this job calls for an executive, someone as high up as possible in the organizational chart, such as the company's chief financial officer (CFO) or the chief marketing officer (CMO) or whatever title the director of operations has. Better yet, try to get a joint directive from the chief executive officer (CEO), the chief operating officer (COO), and the chief information officer (CIO) that will get everyone on the same page of the playbook, indicating that both the business and technology organizations are behind the data warehousing project.

So what's the big deal? This prospect seems straightforward, right? Unfortunately, the complications usually occur in large (Fortune 500–size) companies that have a number of different business divisions, each with its own organizational structure, from a president on down. Each division also often has its own CIO and COO, as well as its own data center. To put it bluntly, you can have problems determining who in the overall picture reports to whom. As soon as a data warehousing project crosses these divisional boundaries, the turf wars spring to life. Unless you're one of these high-powered individuals, the best you can hope for is that whoever sponsors your data warehouse project has enough clout and interpersonal skills to get everyone to support it. If not, be prepared.

How Do I Conduct a Large, Enterprise-Scale Data Warehousing Initiative?

For large, complex data warehousing projects that cross a large number of organizational boundaries (the upper bounds of the data warehouse deluxe group, discussed in Chapter 3), choose one of following approaches:

- Top-down
- Bottom-up
- Mixed-mode (combining the best aspects of the other two methods)

Top-down

To develop a large data warehouse in a top-down manner, follow these steps:

1. **Proceed with scope and design phase activities from the perspective of the entire data warehousing environment, no matter how large.**

   I discuss scope and design phase activities in the section " How to Apply Your Company's Best Development Practices to Your Project," earlier in this chapter.

2. **Create an all-inclusive data model of your data warehouse (called an *enterprise data warehouse model*).**

   For your blueprint, use all the data elements that you plan to store anywhere in the data warehouse and the source for each element.

3. **Decompose the enterprise data warehouse model into as many *component models* (smaller models) as appropriate for your environment.**

   Group data elements and subject areas according to the primary function of each sub-model area, with little or no data overlap between models. (Your goal is to have one official storage place for each data element.)

4. **Develop each part of your data warehouse, with each part containing one of the component models.**

This approach is similar to the way most people have been taught to handle any type of large problems, not just those in the information technology business: Get an idea of the "big picture," decompose the problem into manageable chunks, and then work on each chunk. The problem is that although this approach makes sense conceptually, carrying it out successfully in the real world is difficult. The major stumbling block is the creation of the enterprise data warehouse model because of the large number of source applications you need for an enterprise-scale data warehousing project. Even if you're successful, you can have even more trouble keeping a model that large up-to-date, especially with all the data sources you have to consider. More often than not, the enterprise view of data turns out to be of little value.

Use the business users as the sounding board — if they need the data, they should say so in the requirements; if they don't, it will come in a secondary iteration. Get the business objects clearly defined — not that all data elements that exist in your enterprise for the business object are defined in the first iteration. If you want to follow a top-down method, use the business requirements as your guide rather than what you know exists in the various databases.

Bottom-up

You can develop a large data warehouse environment from the bottom up. After you identify a number of different subject areas, or groups of subject areas, that are within the scope of your project, you treat each one as a separate project with little overlap between them. When you finish these different projects, you have your data warehouse — sort of.

The major risk of this approach is that even if each of the smaller projects is successful, the components of the data warehouse probably won't fit together neatly forever, and the environment will eventually fall into disuse. Additionally, without the level-headed influence of the business user, you might spend an enormous amount of time reconciling, transforming, and moving data that's germane to only the source application and therefore has no use in the data warehouse.

Mixed-mode

In mixed mode (my preferred approach), you combine the best parts of the top-down and bottom-up methods by following these steps:

1. **Start with a project scope phase.**

   As discussed earlier in this chapter, explore the business mission, vision, and other defining constraints for the overall data warehouse environment that you want to build.

2. **Initiate a separate data warehouse architecture phase.**

   Concentrate on the overall architecture for the enterprise. For example, catalog all the data sources and the platforms on which they run, identify the entire user community and who will perform which functions, and identify which external data sources are likely to be necessary. The goal of this phase is to create a complete conceptual picture of your corporate data environment (from the data sources to the data warehouse, and all points in between) so that you can identify the pieces that overlap and the pieces that are stand-alone.

3. **Create a data warehouse architecture pilot program.**

   All the overlapping pieces of your environment (the pieces that must communicate with one another) undergo an evaluation, both in concept (checking out vendor literature, for example) and in implementation (seeing whether vendor products do what they're supposed to do and whether they're suitable for your environment, for example).

4. **Revise the data warehousing architecture, based on the results of your pilot program.**
5. **Create two separate design and development paths: one for infrastructure capabilities (the shared pieces used across much or all of your data warehouse) and another path for the functionality of each component of the data warehouse.**

Assign "systems people" (IT staff members) to the design and development tasks for the infrastructure capabilities. Assign a mix of business and IT people to the tasks associated with developing the component data warehousing features.

6. **Decompose the component data warehousing functionality into a series of projects.**

   Each series lasts from three to nine months and has few dependencies on any other project piece.

7. **Continue with the other project phases.**

   I describe these phases in the section "How to Apply Your Company's Best Development Practices to Your Project," earlier in this chapter: requirements, design, development and testing, and deployment.

   Each project piece, as well as your infrastructure, incorporates all these phases.

Create a data warehousing project office that includes representatives from each thread of development activity, including the infrastructure development. Even though each thread of development activity should proceed as independently as possible, to make your overall effort proceed smoothly, ensure that information such as business intelligence product evaluation, issues, and risks can be shared. Data warehousing project offices have recently begun to fall under titles of Business Intelligence Competency Centers or Business Intelligence Centers of Excellence, and key resources from the business and IT departments staff them. Creating such an organization can provide great value to your enterprise because the program will continually improve the manufacturing and maintenance of your key data assets.

Chapter 14

Building a Winning Data Warehousing Project Team

In This Chapter

- Avoiding assumptions when creating your data warehousing team
- Understanding the roles that you need in your team
- Putting the right people in the right roles
- Organizing how your team members will work together

Your attention, please. Here's the starting lineup for your data warehousing project.

Batting first, and playing data architect, with four years of data warehousing experience, Vicki. (Applause from the crowd.)

Batting second, your project manager, with seven years of project-management experience and three of those years spent in the data warehousing area, Paul. (More applause.)

Batting third, last year's most valuable player for her key role on a 90-day data mart development project that was delivered two weeks ahead of schedule, and serving as the senior developer, Amanda. (Still more applause.)

Okay, being part of a data warehousing team isn't exactly like being a member of a major-league baseball team. In reality, it's nothing like being a member of a major-league baseball team, except for one thing: teamwork.

Your best chances of data warehousing project success lie with building a winning team. Like in baseball, your team must have a balance of skills across a variety of roles. If your baseball team has ten excellent pitchers bound for the Hall of Fame, but all the infielders play baseball like I did in Little League, you probably won't win many games.

Don't Make This Mistake!

A development team that does a top-notch, bang-up job of developing or deploying an application (an SAP deployment or your online commerce site designed to handle large volumes of product orders over the Web) might also do a great job if it's assigned to a data warehousing project.

Then again, it might not. Don't make these assumptions:

- A person who can code in Visual Basic, Java, or C#, for example, can handle data warehousing and/or business intelligence tools.
- A project manager, comfortable with and successful in transactional, production-oriented application development, can deal with the twists and turns of informational, analytical data warehousing projects.
- A database administrator who can tune a database for transaction-processing performance has the knowledge and skills to do the same for data warehousing performance, which has different access patterns.

In short, don't make decisions about building your data warehousing project team without performing some research, including the role of each team member and the qualifications of the people who will fill those roles.

The Roles You Have to Fill on Your Project

The following statement might seem obvious: The size of your data warehousing project dictates the number of team members you need.

Wow! And you're reading this chapter for that piece of wisdom? Wait! Don't skip ahead, thinking that the rest of this chapter is a waste of your time.

The reason I made that statement is that although the number of people on your project might vary, the roles they fill almost always remain constant from one data warehousing project to another. Depending on the size and complexity of your project, it might have

- One person filling each role
- One person filling more than one role
- More than one person filling a single role

In the section "And Now, the People," later in this chapter, I discuss how to determine the matchup of people and roles. The following sections focus on the roles you must assign, before you begin to choose people to fill them.

The following roles almost always have to be filled in a data warehousing project:

- Project manager
- Technical leader
- Chief data-warehousing architect
- Business requirements analyst
- Data modeler and conceptual/logical database designer
- Database administrator and physical database designer
- Data movement and middleware specialist
- Front-end tools specialist and developer
- Quality assurance specialist
- Source data analyst
- User-community interaction manager
- Technical executive sponsor
- User-community executive sponsor

If you don't assign one or more of these roles, either explicitly or because of an oversight, you put your data warehousing project at risk. In all except the rarest circumstances, each of these roles is critical to the success of your project.

Project manager

Here's Rule Number One about your data warehousing project manager: The manager must be a full-time, dedicated resource. Dedicated means, in this case, 100-percent assigned to the data warehousing project, not dedicated as in "very interested in and passionate about" (although also fulfilling the latter definition doesn't hurt).

I mention this rule because I've seen a lot of data warehousing projects get totally messed up because someone wanted to scrimp on the resources or budget assigned to the project, and the temptation is often to say, "Well, maybe we need a project manager assigned to this job on only a half-time

basis." Before you know it, someone has been assigned as your project's manager on an additional-duty basis, schedule conflicts pull this person all over the place, and the project goes to pieces.

This situation seems to happen more often on data warehousing projects (particularly smaller-scale projects that can be classified as data warehouse lite) because of the tendency to think, "We're not developing a real application; it's only copying data from a bunch of different places and putting that data in one place. How difficult can that be?" This type of tremendously shortsighted thinking often occurs in one of these situations:

- An organization, already short on resources, has so many projects taking place that an internal full-time project manager "just doesn't make sense."
- Because of budget pressures, the people responsible for a data warehousing project that's contracted out to one or more consulting firms have this attitude: "We can save thousands of dollars over the life of the project by paying for only a half-time project manager."

Forget the shortsighted thinking. Assign, pay for, or find a full-time project manager for your data warehousing project.

Your data warehousing project manager should be able to do the following:

- Create, manage, and adjust to project plans.
- Communicate effectively, both verbally and in writing, to people in both the technical and user communities.
- Weather project storms without falling to pieces.
- Stick to the project plan and, at the same time, be flexible. That's not a contradiction: When a project is going well, the project manager must ensure that it continues to go well; when a problem surfaces, however, the project manager must be able to steer team members around the obstacle without losing sight of the objective.
- Be both responsive to team members' needs and dedicated to the successful completion of all project tasks.
- Be organized.
- Be diplomatic without being wishy-washy.
- Know enough about data warehousing to be effective in the project manager role.
- Make timely decisions.

Does the same person have to manage all phases of the data warehousing project?

Although some project managers are skilled in managing all phases of a project (scope, rapid development workshop, design, development, and deployment, as described in Chapter 13), others might be more adept at one or two of the phases than at others. An individual might be a top-notch project manager for the construction activities of development (visualization development in OLAP tools and database creation, population, and testing, for example) and struggle during the early phases of a project (such as scoping the project and working closely with the end users to determine their requirements). Other people might have a knack for early-phase activities and have problems with all the nitty-gritty details of development and deployment.

An organization that has a number of data warehousing initiatives might reasonably have a pool of project managers, some of whom concentrate on the earlier phases (scope and design) of a data warehousing project and others who concentrate on development and deployment. Early-phase project managers can hand off projects to managers skilled in the latter phases, in much the same way that a starting pitcher in baseball might regularly leave the game after the seventh or eighth inning (no matter how well he had been pitching) and yield the pitching mound to the ace reliever to close out the game. (Sorry, I couldn't resist a baseball analogy.)

Even if this type of setup makes sense for your organization, however, the key is to make sure that the project manager (whoever is filling that role) is still, at any phase, 100-percent dedicated to the project.

Technical leader

From the first stages of design activity through the successful deployment of the data warehousing environment, the technical leader is the person whom other team members look to for, well, technical leadership. All the details, all the issues, all the product problems, and all the interface issues eventually fall under the realm of the person in this role. The project manager might make assignments and ensure that they're successfully completed, but while team members accomplish these assignments, the technical leader has to make sure that all the assignments fit together and lead toward a successful data warehousing implementation. The pairing of a technical leader with one or more project managers is often good because the chemistry of the leadership team and the relationship between them can help you achieve success in your implementation. Much like Robin had Batman covered, Tonto had the Lone Ranger covered, Patrick had SpongeBob covered — oops, sorry about that one!

Chief architect

The role of chief architect isn't the same as the role of technical leader. (But, as I discuss in the section "And Now, the People," later in this chapter, one individual sometimes fills both roles.) Although the technical leader has to make sure that all aspects of the data warehousing technology (front-end tools, networking, databases, and middleware tools, for example) are successfully implemented and deployed, these activities occur according to the initial architecture created primarily by the project's chief architect.

The architect performs architectural functions, based on the convergence of business needs, the current data warehousing and computing technology, and an organization's internal standards and guidelines. When the chief architect completes that work, though, the implementation becomes the responsibility of the technical leader and the team members involved in the development process.

When only one person serves as both chief data warehouse architect (in the early phases of the project) and, later, as the technical leader, he or she must "change hats" at the appropriate time. Specifically, the role of data warehousing architect has an element of creativity to it: The architect has to take a fresh look at how your company can use technology and products to meet business objectives or to solve business problems. An architect should look at a number of different approaches and architectures, and then choose (or recommend) one that's most sensible for that specific environment. When the architect changes roles and becomes the technical leader, though, he or she must restrain creativity. For example, a technical leader can reasonably look at different ways that a selected OLAP tool might perform some task and then implement a better method. It's risky, however, for that person to begin adjusting the data warehousing architecture several weeks into development.

Business requirements analyst

During the data warehousing project's scope phase, the business requirements analyst collects, consolidates, organizes, and prioritizes business needs and problems that the user community presents. He or she eventually wants to create a set of requirements that ensure the data warehouse accomplishes its original intent when it's deployed.

The business requirements analyst is therefore important to defining the business scope of the data warehousing project. Unless the analyst correctly notes and subsequently validates business needs and problems, and all their characteristics, you run a substantial risk of creating a data warehouse that's successful in a technological sense (users can make requests and receive data back) but a failure in a business sense. If the data warehouse is not tied

to business requirements, users won't obtain any business value from making requests and getting responses, no matter how quickly or how elegantly the results are formatted.

The business requirements analyst must

- Be a good listener.
- Ask insightful questions.
- Be able to create a consolidated set of requirements from bits and pieces of information that surface throughout the early days of a data warehousing project.
- Understand the basics of data warehousing (that most of or all the data exists somewhere and that it's important to find out what pieces are really necessary and determine how to prioritize those needs).
- Be a diplomat, especially in dealing with user groups created from different organizations that probably have conflicting objectives and priorities.

Data modeler and conceptual/logical database designer

After the business requirements analyst collects and validates business needs, someone needs to organize the resulting bucketful of data elements in a manner that can be implemented in whatever database management system the warehousing environment features. The person who does this organizing is the data modeler (sometimes known as the conceptual/logical database designer).

The person in this role concentrates on the conceptual side of the data requirements, a business- and application-specific focus, rather than physical and implementation-specific issues (such as tuning the database for performance and the various nuances of a particular database management system). (The fine-tuning tasks are performed by the person filling the role of the database administrator and physical database designer, as described in the following section. Although one person can fill both these roles, the roles themselves are distinct.)

The data modeler creates data structures that are in tune with the way users will access data and the types of reports and queries they'll run, as determined during the scope phase and the early design stages. If the implementation database has a dimensional nature (refer to Chapter 5), the data modeler identifies facts and dimensions; if the database features relationally oriented structures that have some degree of denormalization (refer to Chapter 5), data modeler structures the data model in that way.

Does a business requirements analyst have to be an industry specialist?

A school of thought in not only the data warehousing realm, but also general applications development, says that unless the people filling the business requirements analyst role are industry specialists (sometimes referred to as *vertical-market specialists*), accurately specifying business requirements that the analyst hands off to people in the design phase becomes difficult (if not impossible).

In my opinion, not only is this school of thought wrong, the exact opposite is true.

During the first days (the scope phase) of a data warehousing project, the business requirements analyst must ask users many questions. One of the dangers of vertical-market specialists serving as data warehousing requirements analysts is that they tend to do two things:

- Enter into a project with solutions already in mind, based on previous successes.
- Close their minds to new, innovative possibilities for how they can transform data into business intelligence (the "Aha!" factor) because they feel that they already have the answers (and the data required to support those answers).

In the worst extremes, an industry specialist can seem arrogant when interacting with a group of users, particularly when an outside vertical-market consultant fills the business requirements analyst role (see Chapter 24).

Whoever leads the scope phase must ask a lot of questions; users, in turn, must provide a lot of explanation. A business requirements analyst should continually ask users questions such as, "Can you explain why you use (some group of data elements) in this way?" and "So, what would the business impact be if it weren't available until the second phase of the project, rather than at the initial delivery?" A data warehousing project is much more likely to deliver a high degree of business value when the business requirements analyst provokes users into thinking about and justifying their data needs and the expected usefulness of the data warehouse, instead of some expert leading users into a solution that might or might not be suitable for their business requirements.

In a data warehousing environment, unlike a traditional application environment, the conceptual data modeling process is complicated by source-to-target data mappings. The conceptual data modeling function concentrates on the target side (the data warehouse), rather than on the source side and various transformations. The data warehousing middleware specialist (discussed in the section "Middleware specialist," later in this chapter) is responsible for mapping and transforming source data into the target environment that the data modeler specifies. One person can fill both these roles, or a team of individuals (each with his or her own assignments and responsibilities) can fill them on a larger project. Regardless of how you establish the team and their assignments, the data modeler must concentrate on data delivery needs and avoid getting bogged down in the difficulties and problems of data transformation and quality assurance.

Database administrator and physical database designer

A conceptual data model makes an environment understandable by grouping data elements into structures such as dimensions or facts. The relationships among different data objects present a fairly clear picture of which data relate to certain other data.

Until you implement a conceptual data model, however (which is the physical database designer's job), the conceptual data model is useful only in a descriptive manner.

The physical database design role is extremely important. The person in this role takes a set of concepts created by the conceptual data modeler and adjusts them for the constraints of the real world. Whoever fills this role typically also serves as an ongoing database administrator during development by performing these tasks:

- Create the initial database *schema* (the physical structure).
- Modify the database schema, as required, throughout development.
- Run load scripts to handle initial population of the database with either test data or real data, and run scripts to reload the database with new data (the data warehouse restocking processes).
- Tune the database for performance by analyzing where response-time problems occur and how you can tweak the database structure to make it run faster.
- Perform backup and restore operations, as necessary.

Front-end tools specialist and developer

The conceptual data modeler and database administrator deal with the database environment. The developers evaluate, select, and build programs in the front-end visualizations that users have access to through the enterprise portal or business intelligence environment: Tools used for simple reporting and querying, online analytical processing, data mining, dashboards, or scorecards. (Part III of this book describes these various types of business intelligence tools.)

The role of front-end tools specialist is much the same as any developer's role: That person creates specifications and designs based on user requirements. Many of a traditional (C or Visual Basic) programmer's characteristics and skills apply to a data warehousing tools specialist. In addition to being creative, a tools specialist must be able to

- Debug logic.
- Determine which of several different implementation strategies makes the most sense in a specific environment — and why.
- Define strategies for verifying the data, building a testing environment to certify the results through the front-end tool.
- Follow design and specification guidelines to ensure that whatever is implemented is correct — such as rules that transform the data and presentation logic that evaluates the data. This requirement is probably the most important.

Middleware specialist

The unique nature of data warehousing (specifically, the reliance on extracting, transforming, and moving data across environments) requires that the middleware functions of extraction, selection, transformation, and other tasks (as described in Chapter 7) be performed to change source data into warehouse-ready data.

The middleware specialist makes sure that data is moved efficiently and accurately into the data warehousing environment, whether by tools or custom code.

Quality assurance (QA) specialist

When I was starting out in the world of computers, people fashionably referred to the "garbage in, garbage out" metaphor in discussing the process of edit-checking input data before writing it to a file or database. That metaphor is just as valid today, especially in data warehousing environments.

The source data isn't the only thing that that has to undergo rigorous quality checking. Someone (the QA specialist) must determine whether variables and parameters in the tools are used correctly; whether all the transformation algorithms and code are done correctly; how exception handling works when data errors occur; and all the other QA tasks necessary for any application development.

You might be tempted to let quality assurance be done "after the fact" — after you develop a system and prepare to deploy it. Don't make this mistake. You need to perform quality assurance from day one in any data warehousing project. Don't think about saving money in your budget by ensuring quality only near the end of a project, and don't think about how much faster you can meet

scheduled deadlines if QA doesn't "interrupt" design and development. Pay attention to quality from the first day of a project and assign someone full-time to this role (on a large project, assign more than one person). The data warehouse is only as good as the data it stores. Therefore, you should architect quality into the flow of the system.

Source data analyst

Source data analysis plays a key role in a data warehousing project because all the subsequent extraction and transformation processes depend on what information the data sources contain. Chapter 16 describes source data analysis — how to do it and why. After reading that chapter, you can't possibly question why you need a source data analyst on a data warehousing project. The success or failure of a data warehousing project often depends on whether a source data analyst has done a thorough job.

User community interaction manager

Chapter 18 discusses the importance of actively involving users in all aspects of a data warehousing project. User involvement is so important that you need to establish a definitive, formal role so that someone can manage interaction with the user community: their requests, issues, and concerns — everything.

Technical executive sponsor

Even though the project manager serves as day-to-day Project King or Queen, you need to officially recognize someone (usually a "high-placed muckity-muck") in the information technology (IT) organization as the project's executive sponsor. When issues and conflicts surface during the project (and they will), resolution and getting back on track often depends on the executive sponsor taking a stand, such as saying in a more formal way, "This is my project. Stop acting like children, solve your problems, and get this thing going again."

The technical executive sponsor does more than just step in when a situation starts getting bad. This person also usually deals with the project's budget, sticking up for the data warehousing project during budgeting time (especially during potential cutbacks), and convincing even higher levels of management that the data warehouse is important.

The technical executive sponsor, though not officially a day-to-day team member, should be more than just a figurehead or a name that shows up on a project's organizational chart. The more involved this person is (by attending regular status meetings and briefings, for example), the more likely the hands-on team members will take the technical executive sponsor role seriously and make use of it for guidance or clarifications, as necessary.

User community executive sponsor

The counterpart to the technical executive sponsor from the user community side is, of course, the user community executive sponsor. Everything about the technical executive sponsor (as mentioned in the preceding section) applies also to this person: Don't be a figurehead, be involved on a regular basis, sell the project's importance — the whole bit.

And Now, the People

You fill the roles in your data warehousing project differently, depending on the size of the project:

- **Large data warehousing project:** A bunch of hands-on people should fill all the different database, middleware, and front-end development roles.
- **Small data warehousing project:** Some individuals might be able to fill more than one role (one person can be both the business requirements analyst and the data modeler, for example, or both the data modeler and the physical database administrator).

How's that for stating the obvious? You were expecting something of substance in this section, weren't you?

Try this suggestion: Don't make assumptions about or definite assignments for the composition of your team until you know what you're dealing with. Because you know that Brandon is both a top-notch data modeler and a pretty darned good database administrator, you probably can have him handle both tasks if the workload and the project schedule allow for it. Or you might realize that Amanda is a first-rate data warehousing project manager and that her diplomatic manner makes her the perfect person to be the primary interface with the user community. Because the project management will mostly involve managing by exception after things get going, she should be able to fit both roles into her schedule.

You might want me to make more definitive statements in this section, such as "You need one source data analyst if you have three or fewer data sources, two analysts for four to six sources, and three analysts if you have seven to nine sources." Because every data warehousing project (or every project, in general) is unique, however, generalizations about exact staffing levels and who should fill which role don't provide anything of value and can even cause harm to your project.

After you've been creating data warehouses for a while, you begin to get a sense that "this one feels like a five-person project," for example. You might think, "This one's so big that we need a project control person to do nothing other than manage the project plan so that the project manager can be free to handle all the issues that are likely to surface." You begin to recognize the people who can handle a variety of tasks, such as analyzing source data, modeling the data warehouse's database, and serving as the project's database administrator. You also get to know the specialists who might be good in one particular role but who really don't have the background and experience to perform other roles. This insight helps you figure out your team's composition a little faster than if you have no knowledge of people and their capabilities and no sense of a project's complexity.

Don't, however, let this data warehousing "sense" strictly govern the way you assign people to project roles. Plan your data warehousing project and its roles, and decide carefully who will fill those roles.

Organizational Operating Model

You need to consider how the project team members will organize and operate with each other. In recent years, the concept of a Business Intelligence Competency Center (BICC) or a Business Intelligence Center of Excellence (BICOE) has begun to emerge. This structure enables the roles and people described in this chapter to operate with one focus — building one thing; the enterprise data warehouse that manages all the information assets of your company.

I like to frame things this way — you're architecting and building one thing, a data warehouse. Users think they should go one place to securely get access to the information assets they need to do their job, and that's what you need to provide. By establishing this vision of building one data warehouse, you can begin to create an operating model. While your efforts mature, you can shift the roles and people defined in this chapter into three main areas:

- **Solution management:** These roles interact mostly with the end-user business customers. The people filling these roles can define the features that end users want in the data warehouse.
- **Information factory:** Your flex staff, including your key technical resources, such as architects and developers. Their jobs are to build, in an integrated manner, the new features that the solution management team defines. Most importantly, this staff packages the features into well-integrated releases of the one product — without breaking previous releases.
- **Operations:** Manage key functions, such as quality assurance, configuration management, release management, infrastructure liaison work, and production support. In essence, this team acts as the door to production and those who keep the lights on for the one system that everyone uses to monitor the business.

You can see this organizational operating model in Figure 14-1.

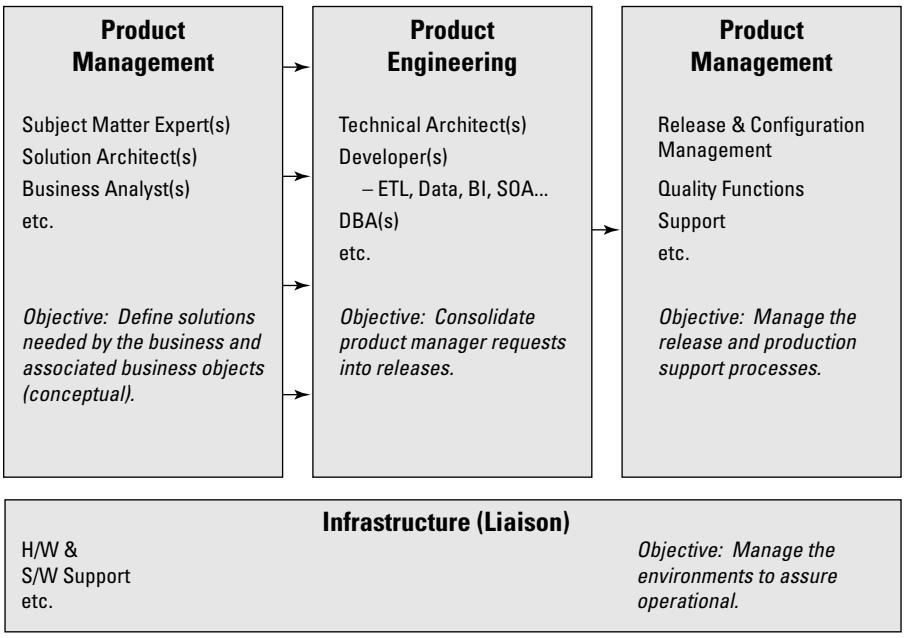


Figure 14-1: Building one enterprise data warehouse through a well-structured organizational operating model.

Chapter 15

You Need What? When? — Capturing Requirements

In This Chapter

- Deciding whether to make your requirements business or technically driven
- Using technically-driven requirement strategies
- Strategizing for business-driven requirements

Every time I think of those who don't listen to user's requirements, I think about that famous *Saturday Night Live* skit depicting a busy diner owned and run by the blustering, mustachioed Pete Dionasopolis. Pete doesn't let a customer order what she really wants to eat (a club sandwich and tea), instead giving her a cheeseburger and soda. In the end, he relents and gives her tea, but not without first frustrating her and probably making her wish she had gone somewhere else.

When it comes to finding out users' needs, you don't want to be like Pete. Instead, listen to and hear what users say they need. Then, you can build an effective data warehouse that users love to use.

Choosing between Being Business or Technically Driven

Hopefully, you decide that if the user wants a tea, you give him or her a tea! Many IT people find focusing on what innovations and implementations will drive the greatest value for the investment a challenge. So, should the business drive IT, or should IT drive the business? A long time ago, when I was consulting with key IT personnel at Procter and Gamble, a leader by the name of Bob Herbold declared, "We are not a software company, we are a consumer goods company." This quote really set my perspective. Such declarations by business leaders should lead us all to focus on understanding what the business will benefit from the most in our deliverables.

Which brings us to the question of being technically driven or business driven with your data warehouse initiatives. This is a trick question — because the answer should always be business driven! However, at times, you need a technical solution. For example, two companies merge, and you need to integrate the data to improve the quality of both systems. This integration creates more of an operational data store than a data warehouse — but it could lead to a business intelligence implementation.

Seriously question an initiative that doesn't define business involvement or inherent business value. Although the ball players finally arrived in Kevin Costner's baseball field in the film *Field of Dreams,* don't risk taking such an approach with your data warehouse — if you build it, they might not come, especially if the data warehouse underserves the users' information needs.

Technically-Driven Data Warehousing

If you proceed down the path of a technically-driven data warehousing initiative, base that decision on the key business principle of finding data that's widely dispersed across a differing set of systems that the business will benefit from. An example of data that is widely dispersed across multiple systems is customer data!

I once worked on a project for the State of Indiana. The premise behind the data warehouse initiative was defined as a business need for understanding the number of customers — a business need that all businesses seem to have a problems with. Specifically, the governor wanted to know how many citizens of Indiana, their customers, were receiving government services. IT came back with a number of people larger than the population of Indiana. Guess it was all those Ohioans or Illinois citizens sneaking across the border.

A technically-driven data warehousing solution doesn't have a true business requirement defined, and the business itself isn't involved in creating the data warehouse. Often, these kinds of projects fail because the business doesn't get involved with and commit to it. For this reason alone, I often recommend that technically driven data warehousing projects not be initiated.

Subject area

When doing a technically-driven project, pick off subject areas. Master data management (MDM) initiatives have driven many of the technically driven data warehousing initiatives of late. Although these technically driven projects do have business value, they're often low-level system integration initiatives to

improve data quality. The benefit to the business is clear, so having the business involved can be a good thing — but, at times, IT just wants to go it alone. Key subject areas that are ripe for the picking in most companies include

- **Customer:** Businesses today need to know who their customers are — and identifying customers can be challenging (see Chapter 26). After you establish a good working definition of your customer, you can begin integrating the customer data. The customer data integration MDM initiative that you undertake works on integrating all systems that own, create, update, delete, or read customer data. The key here is to unify around one customer identifier and then uniquely mark each customer with this *gold key* — one unique identifier of the customer across all applications.
- **Product:** The product master-data initiative cleans up all the data associated with products and the raw materials that make up products. All too often, the manufacturing facilities — which are out of central control — implement the systems that manage products. Therefore, your company manages unnecessarily high volumes of inventory or produces too many finished goods. If you can create a baseline definition for what constitutes a product, you have the basis for integrating the disparate systems and coming up with one uniform product (or item) master.

Enterprise data modeling

To technically build an enterprise data model, begin working on one uniform model that has a centralized meaning for the enterprise data entities. This model assists the technical team in harnessing rogue data throughout the enterprise. Although enterprise data modeling has gotten a bad name over the years, you still need to do it to successfully answer your information management question — and, more specifically, to build out an enterprise data warehouse. The key is to build the enterprise data model one project at a time; *don't* model the whole enterprise and then begin building what you modeled! Attempting to build an enterprise model, and then build it, will take too long before business value is seen — and your project will likely be canceled.

Business-Driven Business Intelligence

The best method for gaining the most momentum and impact on the business is to drive your data warehousing needs from business requirements. You can find those business requirements for business intelligence everywhere, and you don't even have to be a detective — you just need to listen!

Business-driven business intelligence always gets you kudos. It provides users with answers to the business questions they're constantly asking:

- How many profitable customers do we have?
- Which profitable products are being cannibalized by new products?
- How does the employee attrition impact the business most?
- What market scenarios pose the greatest risk to our success?
- Which customers pose the greatest risk to us because of credit concerns?

I'm sure you've heard questions like the ones in the preceding list in the halls of your enterprise. Answering questions such as these provide the basis for your requirements.

When looking at your requirements, consider several key principles for successfully capturing the essence of the business needs:

- You need a consistent methodology to deliver a uniform architecture for strategic applications such as a data warehouse.
- Any business that has a planning process has articulated key performance metrics, so you can often start your initiative by focusing on the business strategy and delivering information to demonstrate progress (or lack of progress) against the corporate goals.
- The human view of data is, by its very nature, multidimensional, so model and deliver information to your users in a dimensional manner so that they can more easily consume and understand the data.
- No one person needs to see it all. Users ask for all the data so that they can do what the data warehouse should do. In other words, they think, "If you can't do it, either I will or I'll find someone who will." Information overload happens often when users are given *all* the data — most people live their lives on six to eight key metrics.
- All data is interconnected and therefore needs an infrastructure to support these connections. It sounds like six degrees of Kevin Bacon! In the end, a set of interconnected processes within your enterprise produce the data. So, build your data warehousing efforts understanding this concept — working to share and reuse data as frequently as possible, as well as enable the user to surf the data from subject area to subject area over connections between that data.

Starting with business questions

When you begin your quest to build a business-impacting data warehousing solution, search for high-impact, highly feasible solutions. To make this a successful search, you first need to understand the business. Gain a strong understanding of the business value chain. Recently, I worked with an energy and utilities company — I had no experience in this industry, but I did with manufacturing. So, I first focused on the value chain. As it turns out, the energy and utilities industry is very similar to manufacturing, with one exception — the product of this industry, energy, has no shelf life. So, here's the energy and utilities value chain that I created:

- **Fuels:** The raw material — in this instance, coal — purchased to produce the end product.
- **Generation:** The manufacturing process in which coal is burned to generate power, the end product.
- **Trading:** Because energy must be consumed as soon as it's produced, enterprises must determine whether they need to purchase generated power from another enterprise in the situation that they underproduce or sell off excess production.
- **Transmission:** Distributing the product in a non-consumable form, over those big towers that span the highways and countryside.
- **Delivery:** The final mile to your house. From the high wires of transmission, down to various stations, to the transformer outside your house, and finally to the outlet in your wall.

You must understand how the business executes its value chain (expense to revenue/profit) before you can determine where key data assets are produced and consumed. From this perspective, you can begin to understand the consumption of data across the value chain. Ask the target business user several key questions:

- What information do you need to do your job?
- From whom do you receive this information?
- In what form do you receive the information?
- How often do you receive the information?

Of course, information also formulates its own value chain. Some data serve as raw materials to drive decisions or actions (the final product). Therefore, when asking the questions in the preceding list, also finish the flow of information by asking the following questions of the target business user:

- What information do you owe others?
- To whom do you send this information?
- In what form do you send this information?
- How often do you send the information?

If you perform enough of these interviews, you can connect the information flow throughout the enterprise and determine where you need to fill missing information gaps to optimize decision making and drive fact-based decision making — a quantum leap from most intuition-based decision making that occurs in the old-boy network!

Accessing the value of the information

After you understand what information the business requires, perform a bit of an assessment to understand that information's overall value and cost. To gain insight to these two important factors, assess

- Which questions, answers, and decisions do you need to manage and which are optional and therefore nothing needs to be done about them? What information do these questions demand?
- How do the users receive this information? Do they get it with
    - **No pain:** They get the information in a report or spreadsheet, so they can consume it without having to modify it.
    - **Moderate pain:** The end user merges various sources into spreadsheets, manually integrating the data in a non-automated fashion, probably riddled with quality problems.
    - **High pain:** The user doesn't receive any information and therefore has to make assumptions.
- What's the value (in dollars) of the decisions that users make by using the required information?
- What's the feasibility of delivering the required information? Is the information readily accessible, or do users find it very difficult or impossible to obtain?

Using a scale from 1 (low) to 7 (high), formulate your assessment of the business questions and plot them in a two-by-two matrix, as shown in Figure 15-1. Documenting and classifying key business questions allows you to assess which business intelligence and data warehousing solutions can have the greatest return on investment.

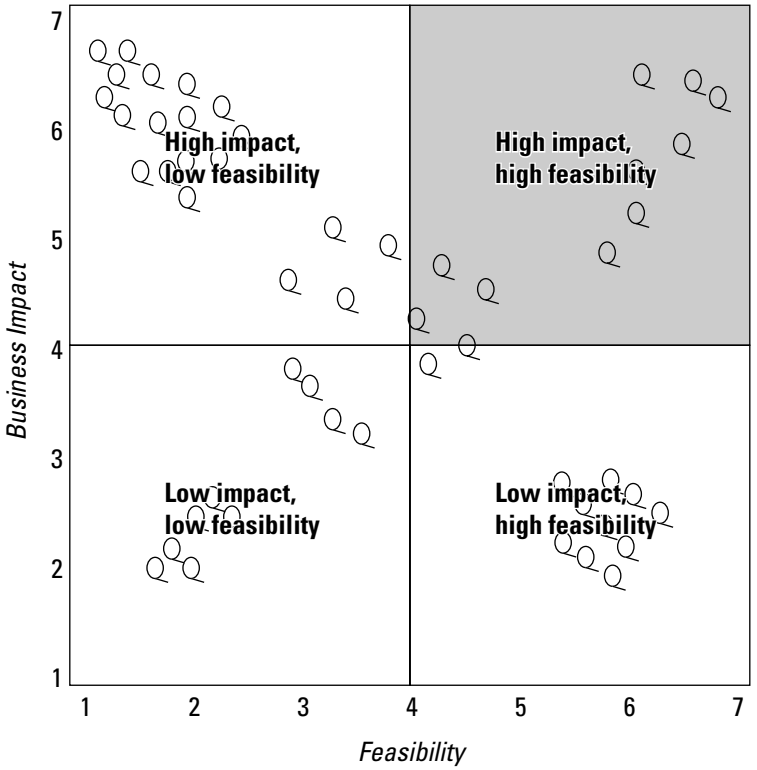


Figure 15-1: Plotting your assessment of the business questions.

Defining key business objects

After you determine the questions that will be included in your project scope, you need to extract key terminology and assure that these words are properly defined. For example, say that you assess the following business requirement:

We have a receivables problem approaching $50M. Do we know whether it's isolated to an operating company or a service area? Is it getting worse over time? Why aren't customers paying?

Key terms for the preceding requirement include answers or focal points of the questions, as well as key descriptors. Performing this analysis would lead you to pull out the following terms:

- **Receivables:** The users clarify that this term might be best described as outstanding balances.
- **Operating company:** The users state that this is a legal entity performing business and recording independent financial statements.
- **Service location:** The users define as a geographical location where an operating company's services are offered to a customer.
- **Time:** The users explain that this term really refers to time periods of months, quarters, and years, enabling trending.
- **Reasons:** The users say this term might be best defined as non-payment reasons (or the non-political word — excuses!).
- **Customers:** The users explain that this term refers to a person, or company — also known as a legal entity — who pays for services from an operating company.

You really need to capture the business objects and their associated definition in business terms. You need a technology solution that enables access to such information for users and other project members if you want to drive consistency and reuse, as shown in Figure 15-2.

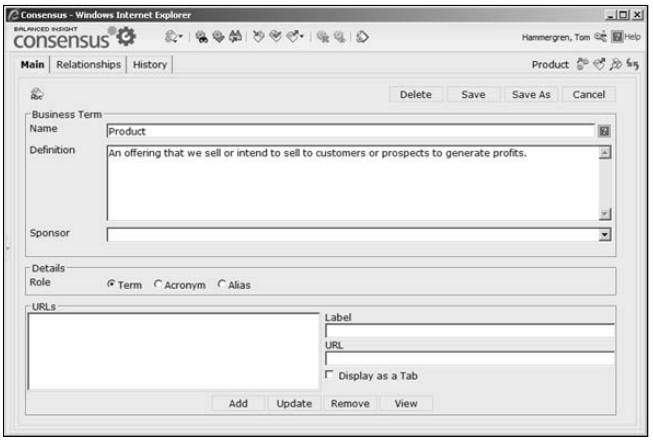


Figure 15-2: Capture vocabulary to enable better understanding and reuse to optimize your data warehousing efforts.

Building a business model

After defining key business objects (as I talk about in the preceding section), work with key users so that you can understand the usability requirements that they have for surfing the data, including key navigational hierarchies and reporting attributes required by your solution. Through your work with the users, you begin to develop a business data model.

The *business data model* provides a visual representation of the desired solution required by the users, consolidating their business questions. The model you produce enables analytical reporting and includes data hierarchies and measures for use by every stakeholder in the data warehouse team. Ultimately, this business data model formulates focused documentation of reference, defining the way the business wants to use the information and capturing the analytical business requirements for reporting and data. You must leverage a tool to formulate linkage between the business questions and their use of key terminology, business objects, and finally formulated into solution-oriented business models, as shown in Figure 15-3.

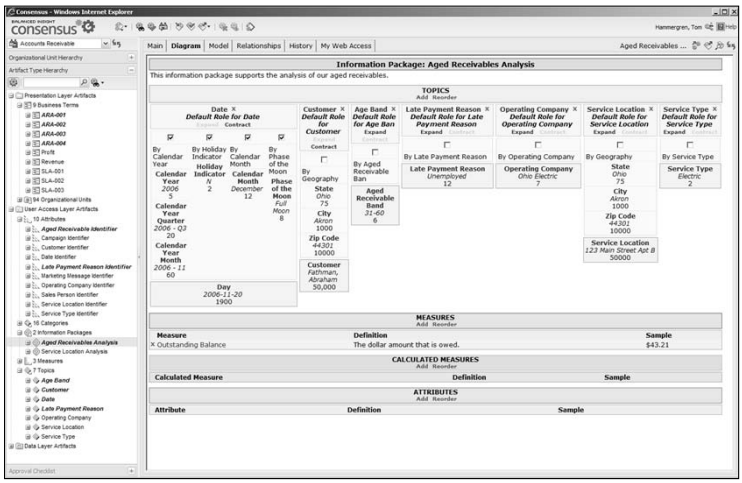


Figure 15-3: Present the key relationships between business objects and measures to document and solidify your project's scope.

Prototyping and iterating with the users

Documenting the requirements helps to a certain degree, but the business users don't necessarily believe that you've captured their needs and will ultimately deliver. Therefore, you need to build out, using the business terminology you've captured, a simulated version of the target application, including the target database environment, possibly OLAP cubes, and the business intelligence semantic layer.

Power users can best understand the ad-hoc environment by using this simulation, and they usually get verbal diarrhea (as a friend of mine charmingly calls it). When a user sees a real, working version of the solution, he or she begins blurting out impressions, such as

- That's not right. Can you add another level to group that information?
- I don't want that at all. Can you change the presentation to include these additional measures?
- Oh, that's what you were trying to tell me — now I see! This is great!

Signing off on scope

After you go through a set of iterations with the users and have their confidence that you've captured the business metadata (semantics) and usability requirements, it's time to go in for the kill: time to get them to sign off on the business requirements and associated business models and then move to the next phase of your development efforts. The technical aspects of the project (design, construction, and testing) typically require less user time than gathering the requirements does. Proceed with users' blessing — not with their concern. Be certain that they're confident you captured their needs. After that happens, get sign-off in some recordable form, whether that's an e-mail or through an electronic approval system.

Obtaining this official sign-off on the business requirements allows you to successfully manage scope, changes, and delivery for the remaining phases of your data warehousing project.

Chapter 16

Analyzing Data Sources

In This Chapter

- Digging into source data
- Putting together an action plan for analyzing source data
- Ensuring that you assign the right people to the job
- Employing different techniques to analyze source data
- Analyzing what's not there
- Introducing mapping and transformation logic

Although the process of extracting, transforming, and moving data from its sources to the data warehouse is complicated, some people would have you believe that it's still a relatively straightforward mapping exercise that you do at the *structural* (database definition) level.

I would (and do) argue that the structural transformation is the least complicated part of the process of determining what you want to include in the data warehouse and then populating that warehouse. The most complicated part of the process involves digging through the source data (the files, databases, and various archives formats) and finding whatever quirks, oddities, omissions, and outright errors are waiting to bite you directly in the — you get the idea.

Source data analysis plays a key role in a data warehousing project because all the subsequent extraction and transformation processes depend on what data the data sources really contain.

A couple of years ago, I was working on a data warehouse lite project (see Chapter 3) that was being done in conjunction with an application migration project. One team (another consulting company) was working on the application migration, and my team was developing a reporting and querying environment to replace the current one.

The other team was responsible for converting the source data files into the new application's database environment (SQL 2005 running on a Windows Server). While the other team completed phases of the database migration,

they made the converted data available to my team so that we could check out our scripts and screens against real data. Until that point, we had been working to build our reporting environment from a long list of database table definitions, as well as their accompanying data element descriptions and database-enforced business rules (lists of permissible values and rules for cross-table data relationships, for example).

After we began receiving the post-conversion data, though, I decided to dig into the data to see what was there. Using plain old SQL SELECT statements through Microsoft SQL Server Management Studio, I poked around, looking for nothing in particular — just checking out the interesting little tidbits of information.

While browsing through a list of the company's customers, I saw the name of an Ohio supermarket chain where I had held a summer job in college. Because I couldn't remember, 20-plus years later, the specific store number and address, I issued an SQL SELECT statement to return all the store numbers so that I could see whether an address would jog my memory.

(Because the client organized and grouped its customers according to how the client issued their contracts, a master customer record was linked to all the members — in many cases, individual store locations belonging to that customer. The use of SQL to get my "show me all their stores" query answered was, therefore, relatively easy.)

Much to my surprise, I found the address of the store where I had worked and others where I had occasionally shopped. (I even found the location of a store in another part of Cincinnati where my college girlfriend had worked.) I also found records, however, for individual stores from several different national drugstore chains, linked back to the master record for this regional grocery chain. Puzzled at first and then suspicious, I dug a little more into the database and found a number of similar discrepancies (or so I assumed) in which the associations between customer master records and individual stores seemed to make no sense.

The strange records had shown up because problems with the data-conversion routines had messed up the associations between a large number of customer data records. After fixing the problem, the application-migration team had to redo most of its data conversion.

You should remember two things from this story:

- Unless someone (perhaps more than one person) is filling the role of source data analyst, by digging into files and databases to see what's there and trying all sorts of different hands-on inquiries to find problems, you might be populating your data warehouse with all kinds of erroneous information, which puts the success of your entire project at risk.

- Because I had once worked at this supermarket chain — computer jobs were hard to come by in the early 1980s, before personal computers became popular — I was able to use my old summer job in my data warehousing career, even if it was only the store's address and location. (It probably wouldn't have worked the other way around: putting data warehouse knowledge and experience to use in a career at a supermarket chain.)

Begin with Source Data Structures, but Don't Stop There

Before you begin digging into the data, spend some time looking at the structural definitions of your warehouse's various data sources:

- Database table and column definitions, constraints, and other Data Definition Language (DDL) statements
- The structures of source data files (VSAM or ISAM files on an IBM mainframe, or Relational Tables on a DB2 or Oracle Database, for example), as described in source program listings
- COBOL "copy book" definitions (if they're used — and yes, for all you young ones out there, COBOL still exists in the modern world)
- Definitions that might be stored in a centralized data dictionary or repository

Begin with program listings and other such material and then gain access through various tools to look at the content of the files and databases as soon as possible. You must ensure that your analysis is based on the most recent information available.

Your goal is simple: Begin building up your knowledge about the data that's likely to find its way into your data warehousing environment.

You might have to study more than just the current versions of database and file structures. If your data warehouse contains historical information — which it probably does, especially if you perform any type of trend analysis — you're likely to use archived data from one, two, three, or more years ago that has been dumped from the active production systems onto various storage formats (online and offline), including tape, optical disk, or some other medium (even regular files just sitting around on your disk drive). Over time, application changes almost certainly required data structure modifications. For each iteration of archived data, you must determine the structure that was in use at that time in order to understand what data elements and their respective characteristics (data types and size, for example) you have to bring into the data warehouse.

Identify What Data You Need to Analyze

Suppose that you face the following situation:

- The data warehouse obtains data from five different source systems.
- Two of the source systems have more than 200 database tables apiece, and each source system has more than 3,000 data elements (the two systems have more than 6,000 data elements between them).
- Two other source systems have approximately 100 database tables apiece and collectively another 3,000 data elements (a total of 9,000 data elements, if you're counting).
- A fifth data source has 50 database tables and 1,000 data elements, for a grand total of 10,000 data elements across the five data sources.

To do the source system analysis in this scenario, therefore, you must know the structural definitions for approximately 650 database tables and 10,000 data elements, right?

Wrong! (Fortunately.)

Under the guiding principles of the good data warehousing seal of approval, you want to bring into your warehouse only the data that's part of your business intelligence picture. Therefore, you have to study only the data that's aligned with the set of business needs for which you're building the data warehouse.

As I mention elsewhere in this book (and specifically in Chapter 15), you don't do data warehousing for the sake of doing data warehousing. After you have a specific business mission in mind, align everything you do in designing and building the data warehouse to meet that mission. This point might seem repetitive, but I can't stress it enough.

To put an action plan in place that can help you sift through the numerous volumes of data elements in a reasonable, timely manner, follow these steps (Chapter 13 has more details about the recommended data warehousing project phases):

1. **During the scope phase, identify a number of facts as they relate to possible data warehousing functionality.**

   During this phase, you're trying to figure out what you need to do with a general business objective that best works in a data warehouse format. To analyze each store by department sales against how the same store

did in the preceding month, for example, your data warehouse must have sales data. Digging a little more into the project's specific needs during the scope, you (or users taking part in the scope work) might determine that the sales data must have dollar and unit amounts by product and department, and be cross-referenced with the staff member responsible for the sale.

2. **Based on the consolidated set of facts (and the data it represents), identify the data sources necessary to provide the elements to build those facts within the data warehouse.**

   For example, two different applications might have sales information recorded as part of the production environment: one that handles the eastern half of the United States and another that handles the western half. These systems are different from one another: The western stores are part of the overall environment from a corporate merger, and the systems conversion (to get everyone on the same sales-tracking application) hasn't occurred yet. You also need to incorporate order completion and bill collection information housed in the enterprise's financial system.

3. **Begin the data warehouse design phase.**

   While your data modeler begins designing the warehouse's database environment (refer to Chapter 5), the source systems analyst begins figuring out what data you need from the various sources and what you need to do to the data to bring it into the data warehouse.

   The source systems analyst studies each data source and its tables or files to match them against the facts and data sources identified during the scope phase. If he or she doesn't identify a need for data in a source, you don't have to analyze it any further (for now, anyway). It's that simple.

In addition to continually checking the guidelines developed in the scope phase, you can use a couple of tricks to make the source systems analysis go even faster by eliminating data you almost certainly don't need for your data warehouse. Here are the guidelines you should follow:

- You can usually eliminate any source database table that's used for systems management purposes, such as storing the physical addresses of nodes on a network, from additional analysis because that data is of no value for the business intelligence mission of the data warehouse in most (if not all) situations.
- Any source database table that contains only a single column that has a nondescript name, VARCHAR2(255) data type (or some other large number of characters), and a description that indicates it's an "interface

table" for electronic data interchange (EDI) or some other type of inter-environment data exchange is usually not useful to your data warehouse, so you don't have to look at it.

- Any source database table that includes the comment "reserved for future use," despite having what seems to be a complete list of database columns, is likely to be empty. No data, no analysis.

  When you find a table with this kind of comment, you must ensure that the table is empty and that you're not just looking at an old comment. The best way to do that quickly is to issue the SQL statement SELECT COUNT(*) FROM *table-name.* If a value of zero is returned, the table is empty, so just move on.

Line Up the Help You'll Need

Never, never, never call up a person responsible for maintaining an application and its database (or files) and say something like this: "Hey, Ellen, can you do me a favor? I need to analyze that inventory application you maintain to see what the data looks like. If you have some time later in the week, could you let me know whether the data has any problems I should know about? Thanks. I'll talk with you Friday."

Don't make the person (or people, if you're working on a larger project that has multiple data sources) assigned to perform source systems analysis for your data warehousing project the same staff member responsible for maintaining the applications and their data. Get a fresh set of eyes and a curious mind looking at every data source — someone who wants to hunt out little oddities and tidbits of information. If you assign the source systems analyst role to someone who struggles to keep the application running day after day, it doesn't matter how conscientious the person is: You're unlikely to get the same level of data analysis that you would with someone from the "outside."

That outside person can be an external consultant or even a person working on the data warehousing project who works mainly with another organization. If you rationalize that the source systems analysis can go faster if someone familiar with the application and the data it manages, you're probably right. I can almost guarantee, however, that the quality of the analysis work won't be as good as when someone else has to dig around a little to get the answers — and find all kinds of interesting items in the process.

Techniques for Analyzing Data Sources and Their Content

You have to consider both the structures of the data sources and each source's contents (the data). Here are some techniques you can use to accomplish these tasks:

- If your source data is stored in a relational database management system (Oracle, SQL Server, DB/2, or Sybase Adaptive Server, for example), SQL is your best friend. The more skilled you are with variations of the SQL SELECT statement, particularly nested sub-queries for cross-table data relationships, the more productive and efficient your source data analysis. You probably won't use many INSERT, UPDATE, or DELETE statements as part of your source data analysis, unless you're using temporary tables for interim results storage and subsequent comparison.
- Use the SQL SELECT COUNT(*) FROM table-name statement frequently to obtain quick counts of the rows in a table so that you can quickly determine whether one table has missing data based on relationships with the content of another.
- Use the DISTINCT phrase in SQL to quickly identify duplicate key values.
- Use a data profiling tool or statistical tool, such as SAS, to execute functions so that you can determine statistical frequencies of source data. Additionally, these tools can assist in finding values that shouldn't be there, either through patterns or rules (for example, dates that have months greater than 12 or less than 1).
- Look at all coded data fields (A = ACTIVE or I = INACTIVE, for example) that are candidates for inclusion in your data warehouse so that you can see whether any data rows or records have invalid values, as shown in this example:

```
SELECT * FROM CUSTOMERS
WHERE STATUS <> A AND STATUS <> I
```

  Be careful to use AND, not OR; otherwise, every row is returned to you.
- Check out the summary tables that some application databases contain. A point-of-sale application, for example, might contain, in addition to the details of every time-stamped sales transaction, a by-day and by-product summary (how many of each product were sold that day) and a monthly summary, also organized by product.

See whether the sales dollars and units recorded for a specific product on a specific day match the sum of all the detailed transactions for that day. See whether the daily dollars and units roll up correctly into the monthly table. By performing such verifications, when you choose the level of data warehouse detail, you know that you can extract the appropriate information from your data source (at the right level of detail) and that it's correct.

Analyze What's Not There: Data Gap Analysis

As challenging as it is to analyze what's in a source system's data, it's even more of a challenge to determine what should be there — and isn't. This data gap analysis is an important piece of your source systems analysis. Here's what you should look for:

- **"Holes" in otherwise complete data, such as no sales data for February 2007:** These holes might exist because, for example, a systems migration occurred that month and data was lost during the cut-over, a monthly archive is missing, or for a similar reason.
- **An inadequate level of detail to support the business needs of the data warehouse being built:** Here's a real-life example: A client purchased external data about its competitors' product sales activity. (Chapter 19 describes external data in a warehousing environment.) In concert with a systems migration, the client was changing its data-purchasing policies and would still be acquiring competitive sales data from the same source, except that its data would be summarized at a higher level of detail. The problem: To do the business analysis functions that were a key part of the reason it was building the data warehouse, it needed the lower level of detail it had previously been receiving. The result is a (subtle) data gap. Competitive sales data was still there; it just wasn't adequately detailed for the client's needs.
- **Changes over time in the structure and contents of a data source:** For example, you might have made programming changes a year earlier that resulted in the database tables no longer storing some not-so-important data values after you used those values during transaction processing (a purchase order, for example). For trend analysis, however, you find those data values important — or you would, if you had them.

When you're using SQL or some other language to analyze data contents, make sure that you issue a number of queries to find and list spaces, zeroes, or null values in place of data. For example, suppose that a mail-order processing application has a MAIL_ORDERS table and a NUM_PRODUCTS_NOT_IN_STOCK column originally used to record how many products on the order

weren't available and had to be back-ordered. Because no one used this column, the code was changed during a maintenance update so that it no longer calculates and writes a value into this column when a new database row is written. From a business analysis perspective, however, this information is extremely valuable. You have identified a data gap. At a certain point, you can no longer get information on an order-by-order basis about the number of products that were ordered but are no longer available.

You might find handling data gaps either easy or difficult, depending on the ability to recover or re-create the missing data. In some cases, you can re-create history by doing the following:

- Run old programs to re-create missing files (if you know the starting data values or saved them in some type of transaction log).
- Repurchase missing externally provided data from the original source.
- Dig through old transaction logs and, by writing specialized programs, create the missing data elements.

In other situations, you just can't fill data gaps, and you must make decisions about how you want the data warehousing environment to handle these issues.

Determine Mapping and Transformation Logic

You perform source systems analysis as part of a data warehousing project for one important reason: When you write code or use a tool to extract and transform source data, you must ensure that the data warehouse can handle any types of conditions or oddities in the source data that you find. Newcomers to data warehousing often think of the transformation process as being fairly straightforward: "If Source A has a five-character alphabetic customer identifier and Source B uses a four-digit numeric customer identifier, just select one. For the other source, have a conversion table on hand to unify the customer data from the two sources."

But what about inactive customers? If the data warehousing functionality is intended for only active customers, why should data about inactive customers (perhaps a large volume of data) be brought into the data warehouse?

Or how about the SUPPLIER_TYPE column in the SUPPLIERS table in Source C, the one that's supposed to have a value of either W for Wholesaler or F for Factory, but actually has a large number of rows in which SUPPLIER_TYPE is equal to B? (Originally, the source analyst thought that the B stood for Bob, a guy on the street corner who used to occasionally provide parts that were

in high demand, but no one ever asked any questions about "the Bob connection.") Should you bring those rows into the data warehouse? If so, should you leave the code as the unknown B or change it to U for Unknown?

Without doing the source data analysis and knowing the real contents of the data, the mapping and transformation portion of your data warehousing project is, at best, a hit-or-miss proposition. You might have to slow development, therefore, to handle these problems that you didn't identify earlier. Or someone (not you, of course) might say, "Oh, to heck with it. Just load the data anyway, and we'll deal with it in the warehousing environment." In either of these options, the overall quality of the data warehouse and ability to support the loading process in production might be compromised.

Doing a thorough source systems analysis goes a long way toward preventing unpleasant surprises during a data warehousing project. When the source data analyst completes this source systems analysis, a source-to-target specification (as shown in Figure 16-1) can aid developers so that they can create high-quality extract, transform, and load logic in their preferred tool.

Source to Target Data Map

Target table	Target column	Data type	Len	Target column description	Source system	Source table / file	Source col / field	Data txform notes
Inventory facts	PERIOD_KEY	Date	--	See primary key table.	POS	TRANS	TXN_DTE	Direct
Inventory facts	PROD_KEY	Num	8	See primary key table.	POS	TRANS	X_CAT_ID, X_P_ID	Concatenate X_CAT_ID + X_P_ID
Inventory facts	units_sold	Num	9	The number of units sold.	POS	TRANS	REG_UNITS	Direct
Inventory facts	dollars_sold	Num	9.2	The dollar amount for the purchase.	POS	TRANS	REG_AMT	Direct
Inventory facts	Inventory	Num	9	The number of consumer units in stock.	POS	TRANS	UNITS_RMN	Direct

Figure 16-1: This source-to-target specification uses the Kimball Method template.

And, with regard to source system analysis naysayers, I go with the theme of a certain shoe company's motto — "Just do it."

Chapter 17

Delivering the Goods

In This Chapter

- Understanding data warehousing architectural fundamentals
- Getting the keys to your architecture
- Breaking down your architecture design
- Simplifying your architecture for the user

Remember the old Abbott and Costello routine in which they went back and forth about who was on first? Confusion ran rampant throughout that conversation — just like confusion reigns (unfortunately) in many data warehousing delivery efforts. In this chapter, I attempt to provide you with a common-sense framework to bring order to the chaos found in many enterprise data-management organizations attempting to deliver a data warehouse.

Exploring Architecture Principles

Have you ever been envious of other companies and their ability to deliver information to their users in a rapid, well-organized manner? You've purchased all the best-of-breed products. You have your first data mart up and running — well, you originally called it a data warehouse, but it serves only a small section of the enterprise, so you reclassified it as a data mart. You've found a receptive and excited user base just waiting for you to pick their business area for your next project. But while you begin to expand your data mart, you quickly begin to realize the products, platforms, and tools you previously implemented can't handle the voluminous data and users that you need to support in the new business area.

So, what do you do? Do you reevaluate the products? Purchase similar products from different vendors? No! These actions often just make the problem of nonintegrated data worse, project by project.

The answer to the question of how you can best deliver a data warehouse is easily stated — you need to build an adaptable architecture.

What's an architecture?

Architecture has many meanings among people who develop and use computer systems. The literal definition of architecture is "a style and method of design and construction with an orderly arrangement of parts." But when you ask someone about their data warehouse architecture, they usually say, "Oracle" or "DB2" — the name of a product.

When looking at your data warehouse architecture, make sure that it can facilitate and create a resource that's accurate, shareable, and easily accessible throughout the enterprise. An architecture provides a blueprint that explains how you plan to deliver the enterprise's vision, goals, and objectives for the data warehouse. The components of your architecture include shared data, technical infrastructure, and reusable program logic.

To build a data warehouse, you must establish the proper architecture before you start development, not after development has begun. The enterprise needs to accept, in principle, your established data warehouse architecture before you proceed with development. Without acceptance and support, development efforts that target an integrated enterprise data warehouse will inevitably fail. Widely accepted architecture, along with its associated components, can help ensure that everyone is aware of factors such as what business functions your data warehouse will support and which decision-making functions your data warehouse will enable.

What's an adaptable architecture?

If companies want to build a sustainable competitive advantage, they must deliver an architecture that changes faster than the business does. The goal is *faster cycle time* — the ability to complete a task faster and faster over time. Just like in a manufacturing company, your data warehouse development function needs faster cycle time. You can deliver on this requirement by using an adaptable architecture.

Adaptable means capable of being made suitable to a specific use or situation. Historically, information systems departments have utilized methodologies that produce rigid and inflexible architectures. These methodologies used the same system development life cycle: Analysts and designers defined a black box; got users of the system to sign off on a specification; and, in 12 to 18 months, delivered a system. These development cycles never took into account that in their time period, the business might change. Therefore, more times than not, the final system was out of synchronization with the business.

One of the most important aspects of a data warehouse is its ability to assist the enterprise in managing shrinking business cycles. Top management doesn't tell current information systems departments to show payback in 12- to 18-month intervals if those departments want funding for their projects.

Users want the business to drive the implementation of a key system such as a data warehouse — not for the data warehouse to drive the business process. Because of this desire — or, probably better stated, this goal — you must realize that your project isn't data driven, it's process driven.

Data is the artifact of a process. Any change to the business is in the process, but the data will reflect such a change. An adaptive architecture allows you to facilitate changes in your business processes — almost at the speed of the business. Taking an adaptive approach to architecture fundamentally changes how a system behaves. Specifically for a data warehouse, an adaptive architecture provides users with the ability to take the data that they've always used and dynamically view and navigate through it. This process provides a new, dynamic behavior that's characteristic of interactive data warehousing. Everyone on the development team must understand this concept. Those data warehouse designers who understand adaptive architectures will succeed, those who don't understand these architectures will build systems that are beautiful to look at but don't fundamentally allow the type of leverage that they could.

Understanding Data Warehousing Architectural Keys

Based on the concepts that I describe in the preceding sections, focus on several areas when defining a consistent method of delivery within your data warehouse:

- People and their roles
- A repeatable process
- A standard delivery platform

In the following sections, I describe in detail what each area entails.

People and their roles

As discussed in Chapter 14, one of the most difficult concepts for management to recognize is that the course of developing software in new areas is a human-intensive process. While the development process evolves, people acquire new skills and find different ways to solve problems.

When you begin the process of developing data warehouse solutions, you have an immature development process — and so too do the people associated with the effort. So, start your first project with a foundation team who will become the experience base for future projects. You'll split and clone

this team over time to expand the knowledge of how to build, in rapid progression, your data warehouse — and how to groom the development process to your enterprise.

Although the need for dedicated resources to improve the development process might seem obvious, organizations often avoid allocating such resources. A matrix-style organization in which managers rely on borrowing or reassigning their people as needed has become commonplace. But you really need to build and clone a strong knowledge base throughout an organization when you enter into a data warehousing effort to improve the delivery process in areas such as planning, automating, testing, training, and quality.

Consistent delivery process

Like with consistency in people, you need to establish a consistent process (as discussed in Chapter 13). A data warehouse provides a single source for key performance measurement and historically significant information assets. All the entities contained in the data warehouse are interconnected — therefore, the processes that comprise the delivery of a data warehouse should also be interconnected. Decisions that are required within the project management space of development rely on these interconnections. These stated items lead to an iterative development methodology — similar to the agile methods leveraged within the application development space. Such a process can quickly deliver, in an iterative fashion, new subject areas within the data warehouse that target audiences require.

Have a consistent delivery process in place to overcome potential weaknesses in the development staff. These potential weaknesses don't disappear when you use a consistent delivery process, but the methodology compensates and allows the implementation team to counter the inevitable problems that arise in development situations. Thinking about your own development projects, does anyone really understand the complete system requirements until you have some part of the system implemented, prototyped, or clearly modeled? An iterative approach assists you in more rapid discovery and delivery cycles — allowing your development staff to avoid lengthy development cycles that miss the requirements.

Standard delivery platform

Make certain that the people involved all realize that you're building one data warehouse. Even if your data warehouse architecture has a number of servers and databases — making it appear, technically, as multiple data warehouses — you're building one. You don't want your users thinking something like this:

```
CASE what system am I using?
  If SAP then information is accessed by Bex
  If not SAP then
     Do I want to perform a query?
        Then I use Business Objects WebIntelligence
     Do I want to write a report?
        Then I use Cognos ReportStudio
          Etc.
```

If your users have to think this way, you've failed as a data warehouse designer. If you do fail, a user organization will evolve that builds a series of queries to extract data from the data warehouse and do what you should have done in the first place — except the users don't really know how to build such a system, and it will be a labor-intensive system (meaning users assemble data rather than analyze it). Consequently, your business suffers and you get a key indicator that you need to make your data warehouse adaptable.

Think about all projects for data warehousing as merely releases for one product — which means you're thinking like a software engineer, but this perspective enables you to build a highly integrated solution. Therefore, the delivery platform should provide key components in the various software areas that I discuss in Parts II and III of this book.

Assessing Your Data Warehouse Architecture

So how do you know if you are in good shape with all that I discuss in the preceding sections? I guide people down a very simple path to perform an assessment. And in the section "Is the delivery automated?" later in this chapter, I ask you to assess yourself, which can help you correct any and all evil ways that you might currently possess in your data warehousing efforts. Start with these key questions:

- What are you building?
- How are you building it?
- Have you made the delivery process as automated as it can be?

The answers to these key questions can provide you with a method of assessing your current data warehousing status.

What are you building?

Knowing the answer to this question enables you to understand a lot about your enterprise's maturity as it relates to integrated and adaptable architectures. The key to understanding this question lies in a four-layer reference architecture, as shown in Figure 17-1.

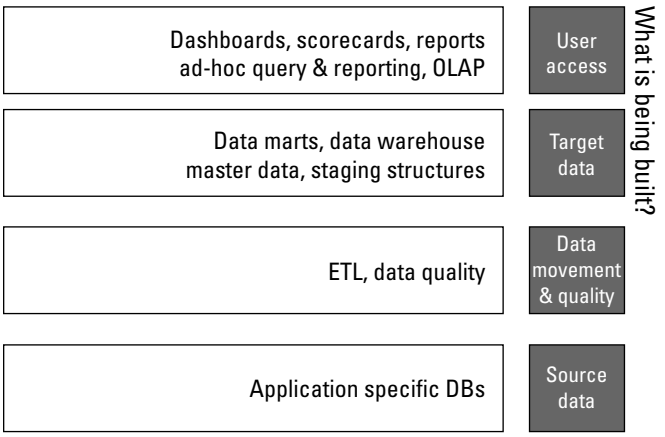


Figure 17-1: The four-layer reference architecture helps you define what you're building.

The answer to this question doesn't contribute to what the users understand — it helps inform you, the designer.

What one platform is being used to delivery user access? User access comprises a set of tools or a platform, as described in Part III. This platform constitutes your business intelligence suite of products. Each time you extend your data warehouse, more queries, reports, dashboards, scorecards, and OLAP views will be added to your data warehouse. Do you want to incorporate these changes in a uniform technology, or do you want to require the user to determine what system and data he or she needs to choose a technology? I hope by now you understand that, to access the data, the user should have to go only to one place to request the data!

The end user really cares about only this layer — but for your own optimization of costs and delivery cycle time, you need to focus on two other key architectural layers, including the target data structures and the data movement and quality layers.

You can make your target data structures very sophisticated and robust — supporting all layers of the monitor-your-business and integrate-your-business components that I discuss in Chapter 1. Additionally, you might incorporate various specialty database technologies, as discussed in Chapter 6. However when you set up your data structures and layers, every delivery person needs to understand why he or she uses the specific technology and how the data warehousing architecture plans to extend that technology for project deliverables.

Lastly, you need to assure that the data movement and quality layer is consistent. Give the platform that you use a consistent set of standards and peer code reviews — whether it be an extract, transform, and load (ETL) tool or a scripting engine. Similarly, data quality modules or products should be integrated and leveraged for their specific purpose across project boundaries — not specifically for one project.

The source data layer is comprised of the various *run-the-business* applications and external data sources that provide the raw materials to build the data assets in the data warehouse. For more information on sources and analyzing source data, see Chapter 16.

Build one technical architecture stack to deliver your data warehousing solutions and stick with that stack through thick and thin!

How are you building it?

After you determine the technical architecture being leveraged as the delivery platform for your data warehouse, you need to assess the process you're using to deliver your data warehouse. The key here is to maintain consistency and establish enterprise understanding of the process. Although I profess that iterative, agile techniques are the best for delivering a data warehouse, I believe that a consistent and repeatable method is the true requirement (even if you use a waterfall method).

Because I'm attempting to make your assessment an easy one, draw the columns to intersect with the delivery platform rows. In this effort, you want to answer questions like those in Figure 17-2.

Around the country and the world, people often have great difficulty clearly articulating their data warehousing delivery process. But they can usually describe their delivery process for many traditional applications. I'm not certain how the business world got to this point, but you need to fix it, and quickly!

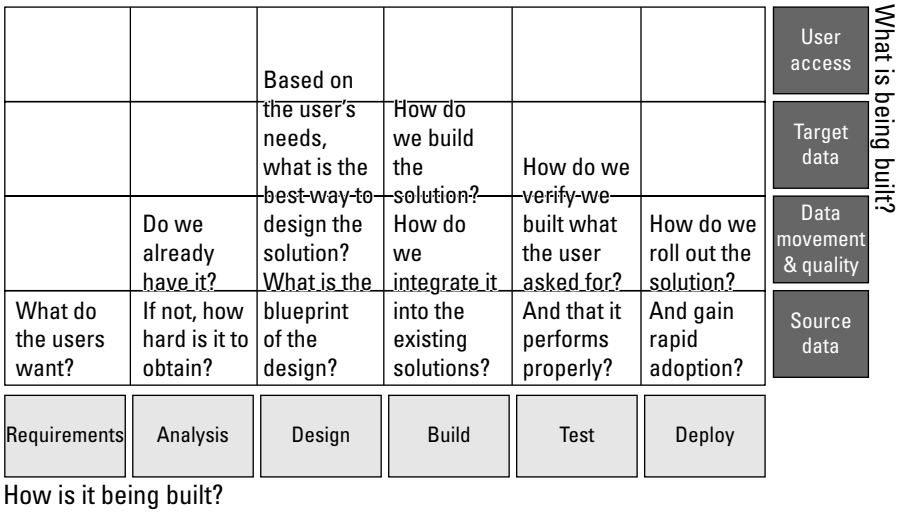


Figure 17-2: Defining a consistent and repeatable delivery method for all data warehousing projects to follow.

The key parts of your data warehousing delivery process should include

- **Requirements:** A standard method of defining what information users need.
- **Analysis:** A technique that your data warehousing team leverages to see whether the required data already exists in the data warehouse. If it doesn't exist, how can you properly source it and determine what you need to do to transform it into useful information?
- **Design:** Based on the user needs and sourcing. What's the best way to design the solution, and how do you present that to the developers in a manner similar to a blueprint being presented to a construction crew building a house?
- **Build:** How can your team build the solution and integrate it with other solutions that other delivery teams have built or are building?
- **Test:** How can the delivery organization and user organization verify the correctness and quality of the information prior to releasing the solution? Also, how can you assure that user access times are reasonable for the user and technical load times can occur within the allotted schedule?
- **Deploy:** How can you successfully train users to accept and utilize the solutions and therefore drive the adoption rate for consuming the information in the user community?

Is the delivery automated?

After you answer the questions "What are we building?" and "How are we building it?" you're ready to do an assessment of your environment. Simply answer these questions in each cell of the matrix in Figure 17-3:

- Who does this function?
- What tool does he or she use to capture and automate the delivery of the key deliverables?

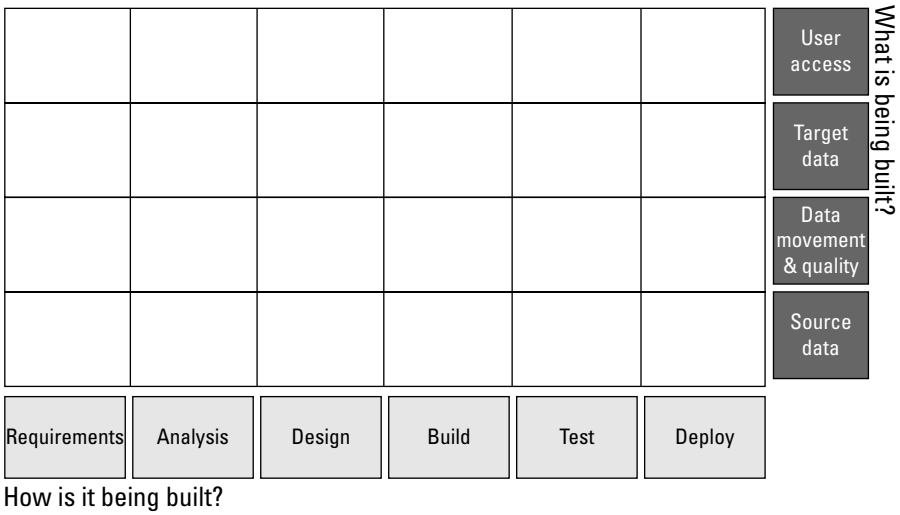


Figure 17-3: A self-assessment tool that you can use to test your data warehousing delivery automation maturity.

Over my years of assisting companies in strategically delivering data warehouses, I've discovered that completing this simple picture is very difficult and implementing it can get convoluted. I implore you — make your environment simple to deliver by putting productivity tools in place that enable faster delivery cycle time. Meaning the process of moving from one cell to the next is automated, which drives reuse and consistency throughout your delivery. The last thing you want is manual rekeying, as shown in Figure 17-4.

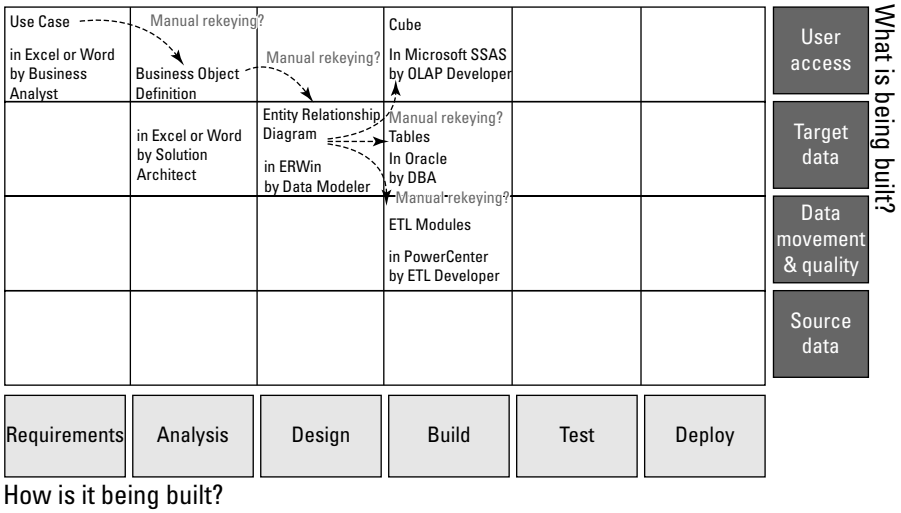


Figure 17-4: Avoid manual, error-prone delivery processes at all cost!

Architecting through Abstraction

One thing's certain in the technology field — things change rapidly. I've seen some very significant shifts during my career. Those shift might not be obvious to people who have only just entered the information technology field in the last couple of years — but you can now store more on a thumb drive than you could on a mid-range computer drive in the early 1980s!

Along the way, various technologies have evolved, assisting the information technology department in building things better but not offering similar value to the end user. For instance, is Java better than COBOL? The technical audience gives a resounding, "Yes!" But if you simply implement the same accounting system in Java that you had in COBOL without adding distributed computing, Web interfaces, and other nice new features that you can more easily do in Java, the users tell you the rewrite was a waste of money.

Technical people can very easily fall in love with technology and attempt to use every bell and whistle provided, regardless of the ultimate value to the business. If you want to be successful in your efforts to migrate through the continuing progress of platform technologies, work to abstract the keys to your solution away from proprietary technologies, including items such as

- **Business intelligence tool modeling and semantics:** Where possible, generalize the platform but standardize business metadata and access standards. If you work for a company that currently uses Cognos and you acquire a company that uses Business Objects, an abstracted semantic layer that has simplified presentation can enable you to bring the new employees from the other company on board more rapidly without swapping out the business intelligence tool.
- **ETL tool logic:** Where possible, build externalized routines for classes of data, such as dates, that can enable you to go to one place for the specific logic. Most people code such logic in each transformation, which is not advisable. Modular programming principles enable you to swap out ETL platforms or upgrade in a much more cost-effective manner.

After you perform your self-assessment by leveraging the matrix in Figure 17-3, verify that there is a standard guide for each cell of the matrix that tells people how to perform their functions rapidly in a standard method across projects. Having such a repeatable process allows you and your team to decrease cycle time for delivering new information to your users.

Chapter 18

User Testing, Feedback, and Acceptance

In This Chapter

- Making users an important part of your data warehousing project
- Letting real business situations determine how you design your data warehouse
- Defining user acceptance

And now, presenting The Data Warehousing User's Anthem, as sung by a misinformed, snooty data warehousing specialist (with apologies to James Taylor):

Don't know much about the OLAP tool

I think data mining is for fools

Don't know what a database is for

Just give me a bunch of data to the core.

Now, I don't claim to be a technologist

But I'll pretend to be

Cause then I can sit in meetings all day long

Rather than become just another worker bee.

This song summarizes the most contemptible attitude I've seen (occasionally) on the part of those who deem themselves data warehousing professionals.

To be perfectly clear, this song doesn't reflect my view of users in a data warehousing project (or any project, for that matter). Increasingly, however, I've noticed a changing perception among some data warehousing specialists. They seem to be saying, "We know what's best for your data warehouse — not just the tools, but also how you should use the data, what levels of detail you should have — all that. Step aside, step aside: professionals at work."

Without user involvement during all stages of a data warehousing project (from the first moments of the project's scope, not just during after-the-fact testing of what has been developed "for them"), you can't have a successful project.

Getting Users Involved Early in Data Warehousing

Don't build a data warehouse unless you can directly relate the project to a specific set of business needs. Furthermore, the purpose of the data warehouse is to make selected data available to business users (maybe just a handful and maybe a number of them) to help them perform informational and analytical functions, such as "Tell me what happened and why" or "Tell me what might happen."

I hope that you consider it difficult (or impossible) to argue with this premise of data warehousing tied to business value. When you consider the business-value premise, no one but the organization members who do the business work should establish and prioritize the data warehouse's functionality, right?

You need to ask questions of users, such as, "Why do you do a particular task this way?" or "What would the effect be if you couldn't perform this task the same way you do it now?" You must handle a likely overflow of requests for features in some sort of order. Users need some help with prioritization and figuring out which features they want delivered first and which ones can wait.

A skilled data warehousing technologist must work with users while he or she evaluates tools in order to point out various products' advantages and disadvantages (by using real-world examples) and to help users choose products to fit their needs.

When a data warehousing initiative crosses organizational boundaries (and corporate politics inevitably become part of the picture), someone who has no vested interest in those politics must be able to steer or cajole people from various organizations toward a potential win-win solution.

Every data warehousing professional, every executive sponsor (on either the IT or business side), and everyone else involved in a data warehousing initiative must keep the following statement in mind: Unless a data warehouse is used regularly (over a period of at least six months) and real business decisions and actions are based on information from the data warehouse, the project is a failure. Lack of use indicates that your data warehouse either doesn't do

what users need it to do or that it contains poor quality data. You might want to review Chapter 2 where I cover successful data warehouse implementations. Even if you build a database, deploy tools, reengage outside consultants for a follow-on phase or for another project, and throw a congratulatory party (with cake and ice cream), your project might still have failed.

So, unless you want your data warehousing project to be one that "grasps defeat from the jaws of victory" (I've always loved that little saying), make sure that your business users will use what you're building after you implement and deploy it.

If you complete the scope phase of your data warehousing project without establishing user consensus about the warehouse's functionality and purpose, you're in trouble. Unless the collective frame of mind of the user community is in general consensus, you should seriously think about delaying the start of the design phase until such consensus is achieved.

Using Real Business Situations

One frustration that both users and data warehousing professionals experience during the early stages of a project is that discussions are often too conceptual and too imprecise for many people to relate to. From the outset of the project, you need to tie together all discussions about warehouse functionality, the data necessary to support that functionality, and the effect of not having these capabilities available to real business situations.

Regarding real-world business situations, make sure that you discuss and analyze both *business problems* (situations that could benefit from more accurate, timely information) and *business opportunities* (situations that aren't now part of the business environment but which, with appropriate information available, could soon become important).

Like in golf, be sure to follow through. Don't just touch on real business situations and then say, "Okay, put that one on the functionality list. What's next?" Dig into each item. Explore the end-to-end impact, all the way through the system (the customer life cycle) to figure out what might happen if a data warehouse that has the features and capabilities you're discussing is successfully implemented. Then, discuss what might happen if the project fails.

At some point, you have to brief the executive sponsors or possibly even people higher up in the organization. And you shouldn't get people interested in the data warehousing project only based on how cool the idea of data warehousing is. Frankly, no one on the business side of your organization whom you have to ask for project funding or continued support should

care about issues such as ROLAP versus MOLAP or Cognos versus Microsoft. These folks want to know what this expensive funding request can buy them in business value — either a hard return on investment (ROI) or at least "softer" returns, such as improved customer service or better supply chain management. Using real-life examples from the proposed data warehousing functionality, you should be able to walk decision-makers through the end-to-end processes of how that data can help people in your organization do whatever must be done.

Ensuring That Users Provide Necessary Feedback

Behold, the reluctant user — drafted into the data warehousing scope phase, ticked off because he or she has "real work" to do, and resistant to any project that takes away those comfy little extract files that he or she has used for the past five years (and work just fine, thank you).

You must do whatever it takes to break through this reluctance, determine why a user is skeptical about or resistant to the project, and ensure that each individual contributes to the data warehousing project. If you don't, that person might become a one-person threat to your project's success.

I'm a firm believer in facilitated work sessions as the primary vehicle for involving users as efficiently as possible when you're trying to determine and prioritize data warehousing functionality. With reluctant users, however, you can often get the most valuable insights from one-on-one discussions (perhaps in an informal setting, such as over lunch) to try to draw out the true reasons for their reticence or resistance.

Whoever's running the data warehousing work sessions during the scope phase has the responsibility of keeping a checklist of each user's level of interaction and involvement. Like in most group settings, a small number of (perhaps only one to three) strong personalities emerge who dominate conversation, perhaps intimidating others into silence. The *facilitator* (the person standing up in front of the room and leading the discussion about functionality and data needs) must call on others as necessary to achieve a more balanced view. That person can use techniques such as voting or requesting that everyone in the room state an opinion or make a contribution.

For larger user groups, don't overlook the value of *breakout sessions:* dividing the overall group into a number of smaller groups and sending them to separate rooms for discussion. Cluster stronger personalities together where they can counteract each other so that you can draw out the opinions and specific business needs of people in other groups.

After the Scope: Involving Users during Design and Development

This chapter talks mostly about the importance of user involvement in the scope phase — the point at which you make critical discoveries and decisions about the business mission of the data warehouse, its specific functionality, and the data necessary to support that functionality.

User involvement doesn't end when the scope phase moves into the design and development phases. For more on the data warehousing delivery process and phases, see Chapter 13. Rather, involve users throughout all phases, performing roles such as the ones in this list:

- Provide feedback about different user interaction options, screen layouts, and other features of front-end tools.
- Help with the source-data analysis process (refer to Chapter 16) to decipher often cryptic codes and assist in analyzing potential source-data problems and data gaps.
- Work with the quality assurance (QA) team members as early as possible to ensure that requirements are being implemented correctly.
- Participate in regular status meetings to discuss issues such as training, deployment, desktop integration issues, and the next phase of development.

Understanding What Determines User Acceptance

Some folks might define user acceptance as the point at which users stop making last-minute requests for changes in functionality or database contents, and actually begin using the data warehouse.

I don't like this definition. It's too haphazard and depends too much on imprecise criteria. Set the exact criteria to determine user acceptance early in the project — ideally, during the scope phase. Although the exact nature of the acceptance criteria varies from one project to another, all parties should agree to these types of statements:

- **What constitutes preliminary user acceptance:** Usually, you've completed development, the development team has run the final QA processes, a user QA team has verified that the system is performing correctly, and a user QA team has verified all numbers tie out with the source systems.

- **What constitutes final user acceptance:** You've completed training and deployed tools to all appropriate desktops, and all users have performed a series of "check it out" steps to ensure that they can correctly access data with acceptable performance and data quality.

Remember, user acceptance doesn't occur when everyone gathers in the company cafeteria for the congratulatory party. Users truly need to use the data warehouse.

Part V

Data Warehousing: The Big Picture

"Okay, well, I think we all get the gist of where Jerry was going with the site map."

In this part . . .

No data warehouse is an island. And it definitely isn't the island on *Lost*!

This part of the book describes a data warehouse in the context of many other parts of your professional life: how the folks in the executive boardroom look at data warehousing, how to acquire external data about your customers or your competitors (and what to do with it after you get it), and how to avoid letting changes in other applications affect your data warehouse in, I'll just say, unpleasant and unfortunate ways.

Chapter 19

The Information Value Chain: Connecting Internal and External Data

In This Chapter

- Figuring out what data your data warehouse needs from outside your company
- Looking into the quality of your external data
- Fitting external data into your data warehouse
- Keeping your external data up-to-date and in working order

Your data warehouse is most likely incomplete until it includes data that comes from sources outside your company. You might want to build a picture of sales activity across all your divisions for the past three years; but wouldn't you also like to include competitors' sales results so that your data warehouse users can see how your company is doing in comparison?

If you call a competitor's chief information officer to ask whether someone could send you a regular stream of data pulled from the company's sales applications or reports from the consolidated sales analysis data warehouse that the company has already built, you probably get a response similar to, "Are you crazy?" Fortunately, you can obtain much of this type of information from publicly available sources.

Identifying Data You Need from Other People

Simply ask yourself this question: Looking at the complete list from within your company of all data sources that will provide data to the warehouse, what else do you need that you don't already have on the list — and what can you not get somewhere else within the company to help make business decisions?

Here's a list you can use as a starting point:

- Your competitors' unit and revenue sales results from the regions in which you both compete (or will compete)
- Historical demographic data, such as population trends, per-household and per-capita income, and local and regional unemployment data
- Economic forecasts
- Information about your customers' activities and behavior with companies other than your own

In addition to comparative data, be certain to look into partnerships. You might want to determine whether an *information value chain* has emerged because of key relationships with suppliers and vendors within your companies supply chain. I describe an information value chain as a set of interconnected data that relates to your business's value chain, hopefully including the extended value chain data from your suppliers and buyers. For example, Wal-Mart provides stocking and sales information to Procter and Gamble — enabling proper store inventory levels and shipments to avoid overstocking, as well as making certain that stores always have stock available. Similarly, a raw-materials provider needs to gain insight into a finished-goods manufacturer's plans for production. And to connect all the components of this information value chain, the raw materials provider sees the finished goods manufacturer's plans, which the manufacturer formulates by using the distribution partner's sales figures. If your company currently hasn't opened up or gained access to your key partners' data, try to pursue this style of information value chain.

Recognizing Why External Data Is Important

External data is important for one simple reason: To ensure that you make the right business decisions, you need to see the big picture, which usually means you can't find all the answers stored in your company's various computer applications and databases. Here are some examples:

- Your data warehouse might be able to tell you that a customer's bill-paying record throughout all her accounts with you has been satisfactory, with only an occasional brief blip. Before you offer a dramatically increased credit limit, however, you might want to know that this same customer has been continually late with payments almost everywhere else and has a poor credit rating.

- According to the consolidated results in your data warehouse, pulled in from 20 different sales applications across the world, the trend is "up, up, up!" no matter how you slice and dice the data. Your archenemy competitor, however, is doing much better than you and, worse, is getting 75 percent of all new business in key geographic regions, leaving you only the crumbs.
- Your unit sales across your entire product line have been increasing steadily in all stores nationwide except for every store in Colorado. Is the problem an economic slowdown there? New competitors moving into the area? A steady, dramatic population decrease during the past 12 months?

Viewing External Data from a User's Perspective

This section outlines how a data warehouse user sees external data and its importance. Consider these two tables. Table 19-1 shows the sales performance for Good Guys, Inc., and Table 19-2 shows the sales performance Bad Guys, Inc.

Table 19-1 Good Guys, Inc. Sales Performance

Region	*2007 Q1 Results*	*2008 Q1 Results*	*Change*
Northeast	$2,000,000	$2,500,000	+25 percent
Southeast	$1,500,000	$2,000,000	+33 percent
Midwest	$2,000,000	$2,200,000	+10 percent
Southwest	$1,000,000	$1,200,000	+20 percent
Pacific	$3,000,000	$3,300,000	+10 percent

Table 19-2 Bad Guys, Ltd. Sales Performance

Region	*2007 Q1 Results*	*2008 Q1 Results*	*Change*
Northeast	$1,500,000	$2,000,000	+33 percent
Southeast	$0	$0	N/A
Midwest	$0	$0	N/A
Southwest	$1,000,000	$2,000,000	+100 percent
Pacific	$0	$2,000,000	+100 percent

The first table doesn't lie: It shows analysts and executives at Good Guys, Inc., that its sales in every region have increased in the first quarter of 2008, as compared with the first quarter of 2007. Based solely on year-to-year sales increases, they might easily assess the sales performance and growth in each region as good.

The big picture (as shown in Table 19-3), though, should tell even the most out-of-touch executive that if someone doesn't do something soon, all those stock options will be about as valuable as a flowery Hawaiian shirt, baggy shorts, and sandals at the North Pole. The Northeast, Southwest, and Pacific regions of Good Guys, Inc are all being outperformed by their competitors — Bad Guys, Ltd.

Table 19-3 Year-to-Year Competitive Assessment

Region	*Good Guys, Inc (Us) Change*	*Bad Guys Ltd (Them) Change*	*Competitive Assessment*
Northeast	+25 percent	+33 percent	Problem
Southeast	+33 percent	N/A	Good
Midwest	+10 percent	N/A	Good
Southwest	+20 percent	+100 percent	Big Problem
Pacific	+10 percent	+100 percent	Uh-Oh!

Determining What External Data You Really Need

Don't overdo it. The same rule that applies to internal data in your warehouse is just as applicable to externally sourced data: Make sure that your analysis and decision making will have *true* business value before you go through the trouble of analyzing, transforming, storing, and making available all this data. If your competitors' sales data helps you get a clear picture of how you're doing, go get it. If the knowledge that certain city populations are dramatically increasing or decreasing has no bearing on your company's decision making, why bother acquiring and storing that data?

Suppose that the database service bureau from which you decide to purchase sales data has an extensive catalog of companies, time periods, and types of data elements, with a variety of package prices available. When you're considering raw data for your data warehouse, you might be tempted

to think more is obviously better. Just like with internally provided data, however, you have to apply your business needs analysis before you begin to consider what data to acquire.

Some, perhaps many, of the data warehouse users will probably apply the simple "tell me what happened" style of querying and reporting, not OLAP "help me understand why something happened" or data mining "tell me what might happen" styles. (Part III describes the different types of business intelligence.) Because simple querying and reporting almost always has an internal focus, you don't need to consult those types of users about external data needs.

By process of elimination, therefore, you must make the remainder of the user community part of the external data business-needs analysis. Figure out who falls into this group as soon as possible so that you can focus your analysis and design efforts toward externally focused users. Follow these steps:

1. **Revalidate your list of total users.**

   This list includes everyone in the company who's a potential data warehouse user, as described in Chapter 14. Is everyone on the list still a candidate to use the data warehouse? (Or, if you've already deployed the data warehouse, does everyone on the list actually use it?) Do you need to add anyone to the list? If you're satisfied with the accuracy of your data warehouse user list, continue to Step 2. If not, make sure that you adjust the list until it's correct.

2. **For each person on the list, answer this question: To perform most effectively his or her assigned business functions, does this person need any data that's not available from the company's internal computer systems?**

3. **Using the results from your interviewing, create a consolidated list of external data needs, the sources from which you can obtain the data, prices and fees, restrictions, and contact information.**

4. **Talk to your project sponsor about budget approval.**

   Make the request and do whatever else your company requires.

Often, in dealing with large user populations (100 or more people), data warehousing developers have a tendency to take a shortcut and apply the preceding question to groups of users, not to individual people, in the interest of meeting deliverable schedules. If a bank's credit-analysis organization, for example, has five people (Martha, Robin, Karen, Robert, and Sidney), all who have the same title of credit risk analyst, report to Suellen as peers on the organizational chart, and use the data warehouse, the same data needs apply across the entire group, right?

Don't make this mistake. In more cases than not, a group of this size has at least two distinct business roles, each of which requires different external data (not to mention internal data). Robin and Robert might focus, for example, on credit card risk, so they need credit scores and market data only for bank cards; others in the group might concentrate on installment loan risk and therefore need external credit-risk data and other market data for different types of installment loans, such as auto, small business, and signature. If you work with Robin and find out that she needs credit-card-oriented external data but wouldn't use externally provided installment-loan data even if she had access to it, you absolutely don't want to assume that no one else in Robin's organization needs installment-loan data and that you don't need to pursue that information.

Talk to everyone, even if it takes a little extra time.

Ensuring the Quality of Incoming External Data

After you determine what external data you need, place an order (similar to ordering clothes or a fruit basket from an online site). After you begin receiving data via a stream, file transfer, or some other means, it's smooth sailing — or is it?

What about the quality of the incoming data? You absolutely must apply the same set of quality assurance (QA) procedures to externally provided data that you do to data coming from your own internal systems. Just because you purchase the information on the open market doesn't guarantee that the data is flawless.

Apply QA procedures to every incoming batch of data by following these steps:

1. **Find out whether the incoming data has check values appended to the files.**

   Some examples of check values are the number of records in each file, the total value of each numeric column (total sales dollars for all records and total units sold for all records, for example), and subsets of the total column values (total amounts of sales and units by state, for example). If check values are provided, they must be stored and used as part of the end-to-end loading procedures. No one should officially update the warehouse's contents until the check totals agree with the calculations that you made when you prepared the data for loading.

2. **If no check values are provided, request them.**

   Although the request might take a few cycles (a few weeks or months, for example) to fill, any data provider interested in providing a high level of customer service takes this type of request seriously and strives to make the requested control information available.

3. **During your loading procedures, filter each row.**

   Make sure that the following conditions are true:

   - *Keys* (unique identifiers for each record) are correct across all the information. For example, if each record in the SalesMasterRecord group of data must have exactly 12 related records in SalesDetail Record (one for each month), make sure that all the detail records are present by comparing record key values.
   - Ranges of values are correct. Product sales per month, for example, must be within reasonable bounds for that type of product (airplanes are different from bolts, for example).
   - Missing fields of information (a likely — almost inevitable — occurrence with externally provided data) don't distort the meaning of the incoming data. For example, although the absence of supplemental pieces of data (defined according to the business rules for your specific industry or organization) might not be too serious a problem, if half the incoming records have an empty space where UnitsSold, TotalSalesPrice, or some other critical type of information should be, the value of the data is questionable at best.
   - Especially in the early stages of acquiring external data (the first three or four months, for example), use your analytical tools, as described in Chapter 10, to perform data quality analysis before your users use the same tools to perform business analysis.

     Search for oddities, anomalies, puzzling results, inconsistencies, apparent paradoxes, and anything else that just looks weird. Then, drill down to the roots of the data to check for the source of the weirdness. Remember that you're probably dealing with many millions of rows of incoming data: In addition to not being able to personally check out every single row, you might have difficulty setting up your filtering and QA checking criteria for every possible condition. Anyone who has ever done anything with externally provided source data has come across all kinds of strange inconsistencies and missing data in the incoming information. By putting yourself in the place of users and using the same tools they use, you can probably discover a thing or two that you can correct, making your data warehouse a much better store of valuable business information.

Filtering and Reorganizing Data after It Arrives

The same rules regarding levels of detail and the organization of data-warehouse-resident information (which I talk about in Chapter 1) apply to externally provided data. If your provider sends you a tape that has detailed transaction-level information on it, you have no reason not to summarize and regroup that data into a more manageable format that takes up less disk space.

If you choose to summarize incoming information before it's loaded into your warehouse, do you lose the detailed data? Not necessarily. Even though your main data warehouse might have sales for all your competitors summarized by state and by week, you can keep the raw data that comprises those summarizations in a *data warehouse auxiliary* — some storage mechanism that you don't routinely use but that you can access on a just-in-case basis. Your data warehouse auxiliary might include parts of your data warehouse for staging information, your operational data store, or a stand-alone database that looks like a source.

To help train and educate all your data warehouse users about these supplemental places to seek business intelligence data, give them a guidebook that has a section titled something like "All the places other than the main data warehouse where you can possibly find additional detailed information in case you need it."

Restocking Your External Data

You have to determine, based on your data warehouse business requirements and source analysis (refer to Chapter 16), which of the following four models apply to each externally provided source when you receive a new batch of data and update (restock) your data warehouse.

Model	*What Happens*	*Why*
Complete replacement	Incoming data overwrites older, now obsolete data in the warehouse.	Because the information provider is continually updating all data, even historical information might be different from the last time you received data.
Example: Although the information provider collects sales data from many different chemical companies, the chemical companies provide not only new sales (those in the past month, for example), but also old sales that they hadn't previously provided. Historical sales are then constantly changing; to make sure that your data warehouse has the most accurate information, you have to do a complete replacement.		

Model	*What Happens*	*Why*
Append	You append incoming data to existing data in the warehouse.	History never changes; the information provider gives you only new information.
Example: Each monthly incoming file from the credit bureau contains activity and credit-score changes since the bureau sent you the last file, in addition to the most recent balances. To perform trend analysis over multiple time periods, your data warehouse should always retain the old credit information and add the new information when it acquires that information.		

Model	*What Happens*	*Why*
Rolling append	Although you append incoming data to existing data in the warehouse, you delete the oldest data to make room for new data.	Users perform analysis on a 24-month rolling basis, for example; in this situation, any data older than 24 months is useless, so you no longer have to keep it in the data warehouse (although you can, and should, retain it in the archives.
Example: You receive a monthly file that contains econometric and demographic data, and your statistical models are all built for at most 48 months of data. The oldest month's data in the data warehouse can therefore be deleted and replaced by the incoming data.		

Model	*What Happens*	*Why*
In-place update	You apply each record of incoming data to some record (or records) in the data warehouse, doing the same SQL UPDATE statement.	Although you probably don't use this scenario for externally provided data (it's more common with internal sources), in some circumstances, such a small percentage of the group of data needs to change that you can more efficiently send along only change information.
Example: You receive not only full information provided monthly by a credit bureau, but also daily file-transfer critical-information reports about any of your customers whose credit rating has changed dramatically. You don't get a complete list each day — only a list of customers who meet some "I want to know about this!" criteria (typically about 2 percent of your total customer list). You can apply this update information directly to those records in your data warehouse.		

Acquiring External Data

Up to this point in the chapter, I do my best to convince you that you usually need externally provided data to get maximum value from your data warehouse. Now, what can you do about it? In the following sections, I show you ways to go about acquiring that external data.

Finding external information

If you're looking for industry-specific data, your industry probably has some type of clearinghouse or online exchange from which you can purchase data consolidated from many different sources — a kind of mega-warehouse that provides data to your own data warehouse.

These industry clearinghouse companies get the data they sell directly from you — and your competitors. If you ask around in your information systems organization, you probably can find a person or small department responsible for sending a regular feed of results data to one or more of these clearinghouses. A weekly, monthly, or quarterly transmission is typically done in the form of a direct file transfer that contains information such as sales by product or by geographic region. Standards often dictate the format and content of these transmissions; if you're curious about the details, find out who in your company is responsible for sending data to the clearinghouse company and ask him or her for the scoop.

Why in the world would someone in your company make this information available so that your competitors can get their hands on it? The answer is simple: so that you can have access to your competitors' detailed results. One of the first rules of most private providers of industry sales data is that if you don't participate and send your information, you're not allowed to buy information from them.

In some cases, the process of acquiring external data isn't as simple as the old song: "If you want it, here it is; come and get it." Some companies that collect industry-specific sales results from you and your competitors attach restrictions to what information you can access. You might be able to obtain only certain types of credit information and ratings for your customers and anyone who wants to be your customer. Before performing any detailed analysis and design on the external data that you want to put in your data warehouse and how the information will be used, make sure that you understand all the rules and restrictions that apply.

Gathering general information

You can get a lot of information that cuts across most, if not all, industries. *Econometric data* (a lot of statistical stuff), as well as demographic and population data, are some examples. You can obtain this type of information from a variety of private and public governmental sources.

Cruising the Internet

The Internet is changing the relationship between data warehousing and external data. With increasing frequency, you can find the data you need somewhere on the Internet. Look for financial information at Yahoo! Finance (`http://finance.yahoo.com`) or Google Finance (`http://finance.google.com`).

Data that you can find at these kinds of sites used to be available only if you purchased contracts with clearinghouses. You can readily access some of this data by using tools to pull information from Web pages — but read the fine print to determine whether you're *legally* allowed to tap into such a source.

Maintaining Control over External Data

Imagine ordering online a couple of expensive, white, cotton, button-down dress shirts from one of those catalog places in New England. After you receive the package, you eagerly open it and say, "Hey — this isn't what I ordered!" while you stare at two chocolate brown, polyester, no-top-button shirts that would have gone great with your lime green leisure suit and platform shoes in the mid-1970s. The same sort of thing (but less startling) happens occasionally when you acquire externally supplied data for use in your data warehouse.

Last month, perhaps PRODUCT-FAMILY was a three-character field that, within each record structure, came right after MARKET-CODE. On this month's data feed, PRODUCT-FAMILY has four characters, and a new field called MARKET-SUBTYPE-CODE appears before it. Keeping track of changes in your external data isn't an easy job, but someone has to do it. That someone is likely you. Lucky for you, in the following sections, I detail how you can wrangle unforeseen changes and prevent new ones from catching you unaware.

Staying on top of changes

Sometimes, the changes aren't as obvious as new or missing data fields, or easy-to-spot modifications to the data types or formats. Have regions been shuffled and rearranged? Has the provider's database dropped certain products for some reason? Has one of your competitors stopped providing information to a certain clearinghouse because that competitor now uses a different clearinghouse? You have a right to know whether situations change from one month (or quarter or whatever your acquisition timetable is) to the next because you're paying for this data.

At any company that provides you with external data for your warehouse, demand that your customer liaison tell you of upcoming changes before those changes show up on your doorstep. You don't want surprises when they affect your data warehouse. Play the customer-service card as much as necessary. You're paying for information that you can easily integrate with your internal data, not a surprise package that you have to continually adjust every time a new version appears.

Depending on the volume of data you're receiving, you might need to build an XML-style parser. XML stands for *extensible markup language,* which leverages tagging mechanisms to define not only the data (its type and value) but what the data is and represents (its true business definition). This standard enables applications to communicate more effectively without the need for heavy change management. However, even with descriptive tags embedded in your files via XML, you want to know whether new information is available or old information has been deleted.

Knowing what to do with historical external data

If your company has been acquiring external data for a while for specialized analysis outside a data warehouse (building statistical datasets, for example), when you begin building a data warehouse, you probably have to use as a data source a couple years' worth of backups that are sitting in some archived, offline form such as tapes, CDs, or DVDs, collecting dust.

Be prepared to analyze each archived media for its structure and content before loading the data into the warehouse. You can't stop your data analysis after you study only the current format and content guidelines because at least a few things probably have changed over the period of time that those tapes, CDs, or DVDs represent.

Determining when new external data sources are available

Suppose that you finally get everything situated in the external-source corner of your data warehouse. What about six months from now? Or next year? One phenomenon of the Internet era has been the proliferation of homegrown service providers, with small companies encroaching on larger ones by being able to quickly and cost-effectively deliver information to you. How do you know when new, potentially valuable sources of data appear?

Pay close attention to your industry's trade periodicals. These periodicals almost always announce new providers, mergers of existing information bureaus, and similar events, and they analyze the impact of those occurrences.

Switching from one external data provider to another

Knowing what you need to do to switch to another data provider depends on these considerations:

- **The format and structure in which data is provided:** If your current data source complies with some type of industry standard and your potential new provider supports that standard, *in theory,* you should have no changes. That's the theory, of course. Although most standards have options associated with them (ever wonder, then, why they're called standards?), you hope that changes are minimal.
- **The means of transportation:** No, you don't make a choice about whether Federal Express or the U.S. Postal Service ships the data to you. In this context, transportation refers to whether the data is shipped by CD/DVD (which format? which density?), by file transfer (which protocol? between which operating system platforms?), or over the Internet.
- **Data gaps:** Does your new source provide all the data you get — every single element? Are range constraints (which time periods are included, for example) the same?
- **New data:** Perhaps the reason you're considering switching is that the new external source can provide more data — perhaps much more — than your current source. Would this new data be of value to users of your warehouse, and if so, how would they use it? How will this data affect the size and performance of your data warehouse?

Don't underestimate the complexity of switching from an existing provider of external data to another. Like with anything else in computing (your applications, your hardware — you name it), you can benefit from change, but surprises are almost always waiting for you. Carefully plan any changes to external data like they're development projects in their own right. Overplan so that you don't have to be stuck in a situation in which you've canceled a contract with an existing provider, no more data is forthcoming, and your switch to the new provider is running months behind schedule.

Chapter 20

Data Warehousing Driving Quality and Integration

In This Chapter

- Working a data warehouse into your information systems infrastructure
- Understanding data warehouse data stores
- Dealing with inter-organizational conflict

The nature of a data warehouse (that it's composed primarily, or exclusively, of data that comes from elsewhere, other application databases, and is converted into a data asset) means that it can't stand alone as an independent entity within your organization. Here are some examples of the relationships between your data sources and data warehouse:

- Users have requested new data for evolving business needs, which require adding functionality to the data warehouse. In turn, you must review and modify all transformation, movement, quality assurance, and extraction procedures back to the data sources that are impacted by this new functionality and data.
- Changes to an application that provides data to a warehouse likely affect the contents of the data warehouse and how data moves along the path to the warehouse.
- Acquiring another firm drives you to integrate their systems, which are different from yours, with the data warehouse.
- Application developers might undertake an initiative to validate data from your run-the-business operational systems against the data warehouse master data to improve the overall quality of customer data — which can help reduce customer complaints.

In short, a data warehouse is part of a system (probably fairly complex, rather than relatively simple) of interacting components that you must review from end to end every time change occurs anywhere in the system.

The Infrastructure Challenge

The phenomenal growth of distributed computing (Internet and intranet, as well as data warehousing internal and external data) has resulted in a fundamental shift in the way applications are constructed. In the old days of mainframes and minicomputers, a single physical system largely contained the infrastructure (operating systems, databases and file systems, and communications and transaction managers). With distributed computing now the dominant model (even mainframes and minicomputers are usually part of a larger distributed environment), the infrastructure is spread over many different platforms across your enterprise and possibly outside of your enterprise.

I sum up the significance of distributed infrastructures in this way: When you develop any application or system, either data warehousing or a more traditional transaction-processing application, you have significant dependencies on pieces of the overall environment over which you have no direct control. Here are some examples specific to data warehousing:

- You design a data warehouse that, based on business requirements and applications' data availability policies, must have approximately 25 gigabytes of new and updated data extracted from various sources each evening and sent over the network to the hardware platform on which the data warehouse is running.

  Your corporate networking infrastructure is still undersized. After additional analysis, the network can't come close to supporting the throughput necessary to move the data into your warehouse in the available time window.

- During the data warehousing project's scope phase, you determine that a push strategy to update the data warehouse is the most appropriate model to follow. To implement a push strategy, though, you must modify each source application to include code that detects when that application must *push* (send) data to the data warehouse.

  The legacy applications that provide data to the warehouse are, unfortunately, so difficult to understand that a policy of making no changes unless absolutely necessary is in effect for each application.

- You decide to pursue a relational OLAP (or ROLAP, as described in Chapter 10) solution and run a series of benchmarks against three relational DBMS (RDBMS) products to see which one best supports informational and decision-support processing (rather than transaction processing).

  The product that performed most poorly in your benchmarks is, unfortunately, also your corporate standard, and any relational database installed anywhere in your company must be of this variety, no matter how you plan to use it.

Think conceptually (not worrying about implementation details) in the early stages of a data warehousing project, or any other application development effort — it's not only acceptable, it's also good systems development practice. At some point, however, you must consider hardware, software, costs, budget, and other types of real-world constraints. Before you begin construction, be sure to consider everything that can affect your designs and plans for your data warehouse. This project is very similar to building a house. You follow a process whereby you determine your needs, and then the architect draws up blueprints. The blueprints highlight the materials that you need to support your requirements — assuring that the finished product fulfills the vision established in the beginning.

Data Warehouse Data Stores

A data warehouse is, by its very nature, a distributed physical data store. Distribution of your information assets assists in the performance and usability across systems and across the enterprise. Make this level of usability the cornerstone of your data warehousing mission and objective.

Figure 20-1 shows how the important data stores of a data warehousing architecture incorporate sources of data, the data warehouse, an operational data store, data marts, and master data.

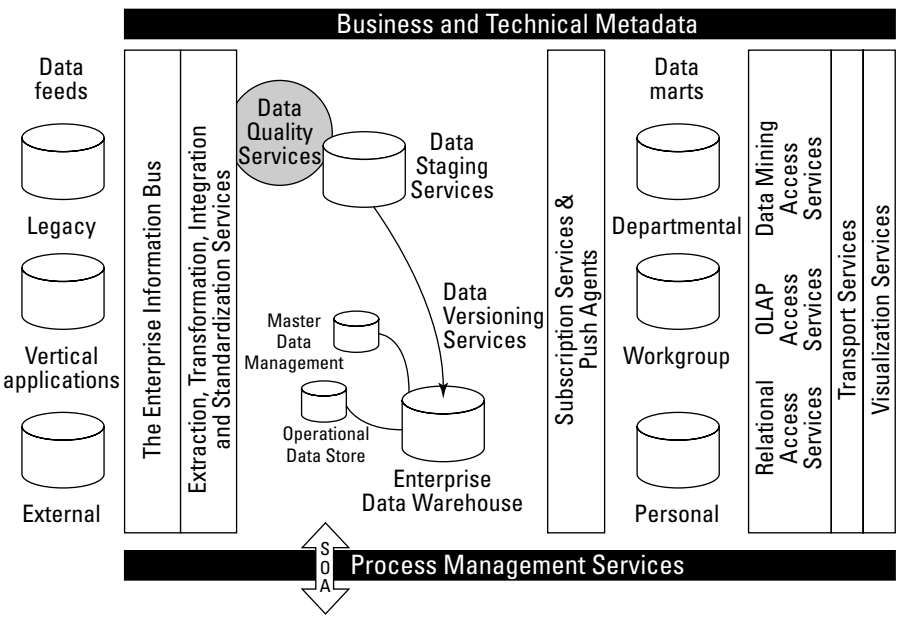


Figure 20-1: A data warehouse incorporates distinct and layered data stores to enable all systems to properly access key data assets.

Source data feeds

Source data feeds are the inputs that feed the data warehouse — typically, your run-the-business application databases, as well as external data sources, such as credit rating data or market segment information. Although the data warehousing team doesn't manage the data and architecture associated with these data stores, the team needs to understand the data feeds. Just like a horse without hooves can't function properly, a data warehouse without sources can't get the job done. The most difficult task you face in data warehousing is choosing the right source, or system of record, for data that moves into the data warehouse. If the data is of low quality or isn't readily available, you have a hard time supporting a high-quality data warehouse.

Operational data store (ODS)

One of the more confusing concepts in the world of data warehousing is the operational data store. No one really agrees on what an ODS actually is.

Some definitions of an ODS make it sound like a classical data warehouse, with periodic *(batch)* inputs from various operational sources into the ODS, except that the new inputs overwrite existing data. In a bank, for example, an ODS (by this definition) has, at any given time, one account balance for each checking account, courtesy of the checking account system, and one balance for each savings account, as provided by the savings account system. The various systems send the account balances periodically (such as at the end of each day), and an ODS user can then look in one place to see each bank customer's complete profile (such as the customer's basic information and balance information for each type of account).

If you want to call an environment such as this one an ODS, by all means, go right ahead. (Oooh — sarcasm.) Terminology aside, this example is just a batch-oriented data warehousing environment doing an update-and-replace operation on each piece of data that resides there (and, of course, adding new data as applicable), rather than keeping a running history of whatever measures are stored there. You can implement this so-called ODS pretty easily, and you can apply almost everything discussed in the earlier chapters of this book to this ODS (for example, you can use the batch-oriented middleware tools and services described in Chapter 7, and the reporting and OLAP tools described in Chapter 10).

My version of an ODS is a little more architecturally challenging. It uses an end-to-end approach that requires warehouse-enabled applications (because you know that they'll provide data to a data warehouse). Warehouse-enabled applications support a push or pull architecture and enable an informational database to be refreshed in real-time (or near to real-time).

Although the premise of breaking down application and system barriers is very much in concert with what you do with a data warehouse, you have one major problem: The pace of updates into your informational and analytical environment is much too slow if you use classical data warehousing and its batch-oriented processes for extracting and moving data.

Forget about terminology and buzzwords. Focus instead on the architectural and time-oriented differences between the ODS that I describe in this chapter and the information I provide about data warehousing earlier in this book.

The ODS defined

Here's my definition of an ODS (it's a long one): an informational and analytical environment that reflects at any point the current operational state of its subject matter, even though data that makes up that operational state is managed in different applications elsewhere in the enterprise. This list explains each part of the preceding definition:

- **Informational and analytical environment:** The user interface and behavior of an ODS look and feel like a data warehouse. So, an ODS user has a querying and reporting tool, an OLAP tool, or possibly other business intelligence tools through which he or she can request and receive information and analysis.
- **Reflects at any time the current operational state:** Okay, sit down at your query tool and ask the ODS a question. The answer you get back must reflect data as it's currently stored in whatever operational system it came from. If an update occurs in an operational system to a customer's checking account balance, the ODS must make that same change in real-time or almost real-time (meaning very quickly). In almost all situations, therefore, extracting batch-oriented data for inclusion in an ODS doesn't work.
- **Subject matter:** Like with a data warehouse, create an ODS with a specific business mission in mind for a manageable set of subject areas.
- **Data managed in different applications elsewhere in the enterprise:** An ODS isn't a single unified database that a number of applications use. Rather, it's a separate database that receives information from various sources through the appropriate transformations, quality assurance, and other processes.

An ODS example

Suppose that you work in a large financial company that provides a variety of services to elite companies and individuals across the world. Your company has grown to its current form as a result of a series of mergers and acquisitions throughout the last 25 years. The trend in recent years toward a convergence of banking and securities services has given your company an opportunity to become a full-service provider to your customers.

Your company's average customer is likely to participate in many (perhaps all) of these types of activities:

- Traditional stock brokering (buying and selling shares of stocks, including margin-account activity)
- Fixed-income investments (corporate and governmental bonds)
- Options trading accounts, including risk arbitrage
- Cash asset management
- Short-term loans and other debt instruments
- Intermediate- and long-term loans and other debt instruments
- Venture capital investments

You want your customers to use your company as one-stop shopping for anything involving large sums of money. Your company's situation is a little complicated, however — particularly in these two areas:

- The mergers and acquisitions have left your IT infrastructure with a large number of solo applications (applications that aren't integrated with one another, even though they probably should be). One system handles stock trading for U.S. stocks, for example, and another system handles stock market activity from non-U.S. global exchanges. In addition, separate systems handle fixed-income activity, all debt activity in the U.S., all short-term debt in Europe, and all intermediate-, and long-term European debt. And the list goes on.
- The definition of a customer is somewhat hazy. Individuals set up corporations and partnerships through which they make investments or secure loans for business deals. Your corporate customers might be subsidiaries of other corporations, which might also be your customers.

Your business practices call for all credit activity with every customer to pass through a series of quality assurance checks before being approved:

- Every customer of yours, whether an individual or a company, has several ceilings on debt activity. One ceiling is an amount of total outstanding debt at any given time. Until a customer reaches this first-level ceiling, he or she can, without human intervention, automatically take out a new loan or act against a credit line, buy stocks on margin, or perform any other type of activity that increases debt.
- Every customer can exceed the first-level ceiling to a second ceiling amount after receiving approval from one of your company's executives.

- For an executive to approve credit activity past the first ceiling to the second, he or she must check a series of measures. For example, the customer must have a certain asset balance in place; the customer might not have reduced total assets on hand in all accounts of all types (such as cash, stocks, and bonds) by more than 15 percent in the preceding 30 days; and the bank has maximum amounts for total debt in each country, adjusted by assets held in each country.
- To help control risk, your company tracks the relationships between all your customers to get a real picture of a customer's financial state. For example, an individual might control a series of corporations, each of which you treat as an individual customer with its own asset and debt activity, in addition to that of the individual's own accounts. When your company's executives approve any additional beyond-the-first-ceiling debt, however (for a real estate partnership that involves that individual, for example), the executives must assess an overall picture of what's going on with that individual's activity to avoid too much risk exposure in case of financial problems.

Although the quality assurance checks described in the preceding list are conceptually straightforward, they're extremely complex to implement, for one simple reason: Those checks need data from systems all over your enterprise, from many different systems. This data includes information such as all the asset activity, all the debt activity and current loans outstanding, and information about which loans were just paid down earlier in the day.

Can an ODS contain historical data?

Some folks declare that an ODS can't contain historical data — only the current values for all its data elements. They say that if the environment has historical data, it's a data warehouse, not an ODS.

I disagree. I worked not long ago on a banking client's ODS that fit all the real-time update requirements from various data sources, pushed out via messaging, as described in the sections "Master Data Management and Service-oriented architecture," in this chapter. The client requested that the ODS have some summarization tables because its customers occasionally ask for historical information, such as, "What was the total interest paid on all loans of all types in the preceding three quarters?" The client's objective was to have a single place for its users to access not only all the various operational data points, such as the current outstanding balance on various credit instruments, but also the answers to these historically oriented questions.

So, what's wrong with a little history stored in some summarization tables in the ODS? I say, "Nothing."

One approach you can try is to provide your company's executives (the ones who have to make the loan-approval decisions) with interfaces into every system in which they might find necessary data. These executives then can run a long series of queries (if they can even be supported), pull out the appropriate values, paste them into a spreadsheet program, and make the decision.

This approach has two problems, however: The chance for human error is high, and the pace at which this type of activity must occur is okay only during "ordinary" times. During a time of financial crisis, when many or most of your firm's customers are buying and selling stocks, covering margins, buying and selling options, trying to handle their hedge accounts, executing against credit lines, and doing all kinds of other activities very quickly, your company's employees just can't keep up.

In this situation, the ODS comes to the rescue. Figure 20-2 illustrates a conceptual architecture that you can use to implement an ODS that meets your business missions. First, the ODS provides a consolidated picture of a client's balances for automatic loan processing under the first ceiling. Next, the ODS enables executives to make yes-or-no decisions about loan requests up to the second ceiling.

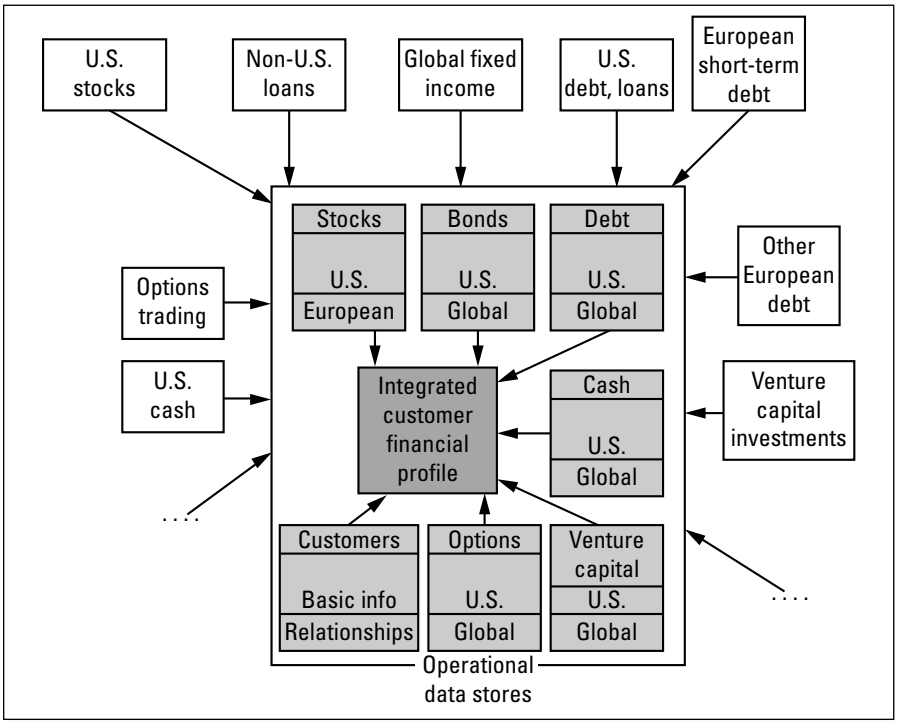


Figure 20-2: The ODS provides users with a consolidated, almost instantaneous picture of different data in support of a specific business mission.

To get a better look at the data flows within the ODS environment, see Figure 20-3, in which updates to one of the data sources (the system that handles U.S. debt) propagate into the ODS environment.

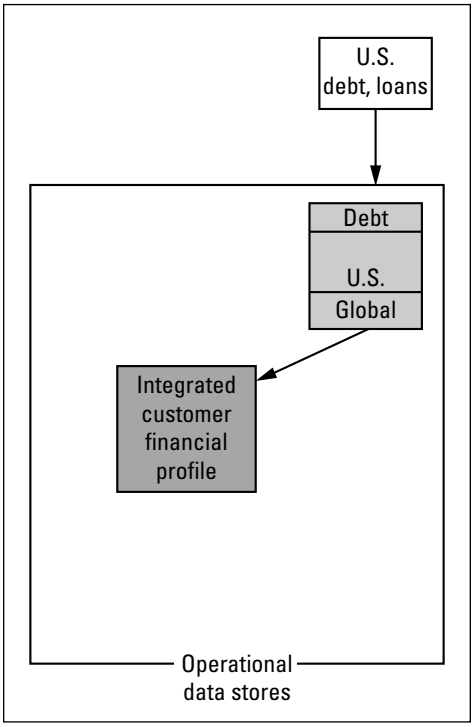


Figure 20-3: The ODS has to reflect the state of data throughout the enterprise as quickly as possible.

The following steps indicate what occurs in the ODS environment:

1. A customer makes a regularly scheduled loan payment, and the system that handles payments on U.S. loans and lines of credit processes the payment.
2. The loan payment application updates its database to reflect the payment.
3. The loan payment application then immediately pushes the updated data to the ODS.
4. The ODS receives the update and processes it, updating its database contents (in this example, reducing the customer's total outstanding debt amount).
5. The ODS performs any internal processing, consolidation, alerts, or other necessary functions.

An environment like the one in the preceding list can — if everything is architected properly — provide a picture of all relevant data from all over the place — now — in support of the firm's risk-management mission.

You must validate the need for real-time updates into your ODS because these updates are complex to create, as described in the following section. Constantly challenge assumptions and ask questions: "What happens if you have to wait until the end of the day? What if updates were twice a day? Every hour?" Be absolutely certain that the mission dictates real-time updates because creating an ODS takes longer (and is more expensive) than a data warehouse.

ODS feedback loops

One of the main challenges you face in an ODS environment isn't in the technical arena (which is challenging enough!) — it's in business processes and how you might have to change them.

Consider the example from Figures 20-2 and 20-3. Suppose that you've traditionally used the two different source applications that handle customer credit-line payments to also approve new loans or permit a customer to act on an established line of credit.

According to the guidelines and business rules (the reason for the ODS in the first place), loan approval can't happen in those applications. The loan system doesn't have access to all the necessary information (such as cash and other asset balances, loan activity handled by other systems, and the relationship among various customer partnerships and corporations) to make the decision according to the business rules.

You can handle this type of problem in one of two ways. The first approach is to remove that type of functionality from the source application — or, at least, block it from being able to be used. After you put the ODS in place, users of the ODS make all the yes-or-no loan decisions according to the business rules.

I generally recommend this approach: Make business decisions by using environment support wherever it exists (in this case, in the ODS).

If organizational politics and other considerations come into play, however, you might have to take the second approach. Assume that your firm's chief credit officer insists that all U.S. and global loan decisions remain with credit officers who continue to use the respective systems they've been using for the past five years.

In this situation, you can enhance the enterprise with feedback loops from the ODS to one or more of the data sources, as shown in Figure 20-4. This scenario operates as follows:

1. As described earlier in this chapter, all data sources propagate all relevant updates into the ODS environment from their respective databases and files as soon as those updates occur.
2. The ODS still provides a consolidated picture of the state of each customer, and other nonloan decisions and information are handled through the ODS.
3. A request to act on a line of credit or take out a new loan, however, is entered into one of the three systems that handle loan processing.
4. Each of those systems makes a request to the ODS for advice for that customer, and the ODS provides a yes-or-no answer about the loan in a message (as indicated by the heavier arrows in the figure), based on its internal business rules.
5. The credit officer, using the regular system, either books a loan or declines it.

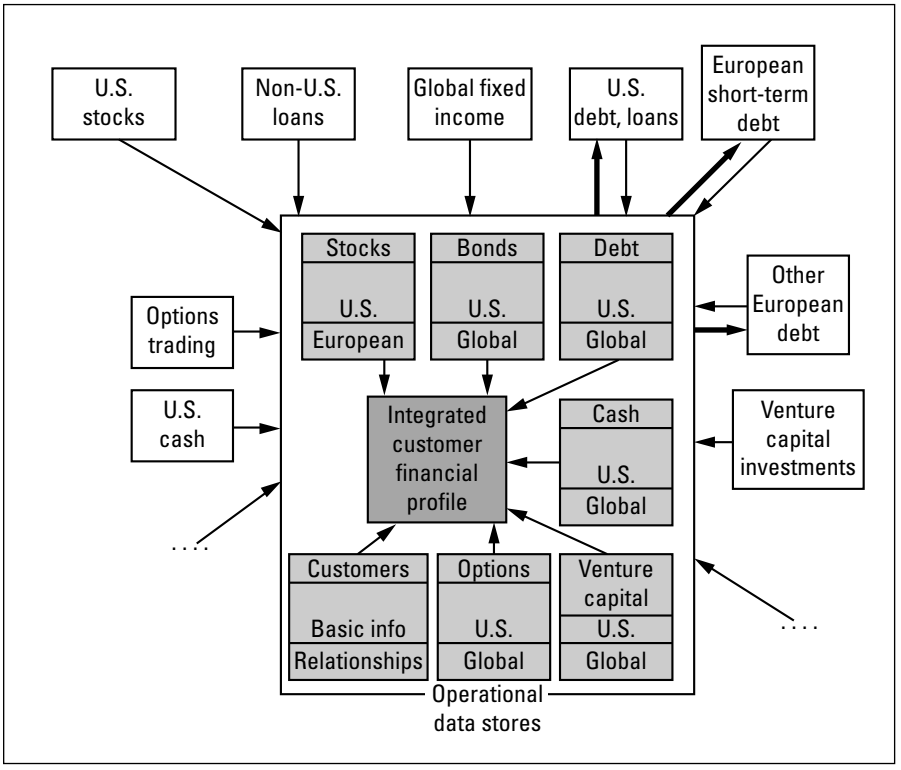


Figure 20-4: A feedback loop sends consolidated data from the ODS, or an ODS-produced decision, back to a source application.

Feedback loops are complicated! They cause increased network traffic, and they introduce another set of cross-application, cross-system dependencies.

Master data management (MDM)

In recent years, ODS-style feedback systems defined for a specific purpose — reference data — have emerged. All systems are packed with reference data. This data can include the set of data you use to describe the stage of a sale opportunity (for example, a lead, a qualified lead, an opportunity, a forecasted opportunity, and so on). Additionally, this reference data can be far more complex. In the financial services example from the preceding section, the integrated customer profile can also be considered reference data.

At a basic level, MDM seeks to ensure that an organization doesn't use multiple (potentially inconsistent) versions of the same reference (or *master*) data in different application systems or parts of its operations. Here's a common example of poor MDM: A bank at which a customer has taken out a mortgage begins to send credit-card solicitations to that customer, ignoring the fact that the person already has an account relationship with the bank. The customer information used by the mortgage-lending section within the bank lacks integration with the customer information used by the credit card organization of the bank. Often, this problem exists because the applications were written to optimize internal processes — not to unify customer information. Similar situations exist for products, locations, employees, and vendors — all considered reference data.

Processes commonly seen in MDM solutions include source identification, data collection, data transformation, normalization, rule administration, error detection and correction, data consolidation, data storage, data distribution, and data governance. These processes are also used in the logical evolution of the ODS feedback loop. A set of tools that have been put together to enable the efficient interchange of reference data between various systems — including run-the-business and monitor-the-business solutions.

What entities you consider for MDM depends somewhat on the nature of your organization. In the common case of commercial enterprises, MDM might apply to such entities as customer (Customer Data Integration), product (Product Information Management), employee, and vendor. MDM processes identify the sources from which to collect descriptions of these entities. In the course of transformation and normalization, administrators adapt descriptions to conform to standard formats and data domains, making it possible to remove duplicate instances of any entity. Such processes generally result in

an organizational MDM repository, from which all requests for a certain entity instance produce the same description, irrespective of the originating sources and the requesting destinations.

With the emergence of MDM, data warehousing teams can resolve problems such as issues with the quality of data, consistent classification and identification of data, and data reconciliation.

Service-oriented architecture (SOA)

If you establish more data integration by using ODS and MDM data stores, you also need a messaging, or communication, architecture to enable systems that weren't built to communicate with each other to do so. Enter the concept of service-oriented architectures, or SOAs.

SOA is a method for systems development and integration in which functionality is grouped around business processes and packaged as interoperable services. SOA also describes IT infrastructure that allows different applications to exchange data with one another while they participate in business processes. An SOA aims to loosely couple services with operating systems, programming languages, and other technologies that underlie applications. This process is very similar to what happened with audio visual equipment while it evolved. You can buy the best speakers for your surround-sound system, hook them up to your audio-visual receiver, hook the receiver up to a high-definition projector, and operate it all with a universal remote. The interfaces between these components has been standardized so that different manufacturers can interoperate with each other's "best of breed" components.

SOA separates functions into distinct units, or services, which are made accessible over a network so that the run-the-business and monitor-the-business applications can combine and reuse those functions. Ultimately, these services reside in the integrate-the-business layer that I discuss in Chapter 1. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are built upon older concepts of distributed computing and modular programming that advancement in various technical infrastructure components and general software engineering have made possible.

SOA provides messaging as a mechanism for moving data (in this case, master data) from one environment to another. Regardless of the products and technologies you use, take a look at cross-system messaging architectures.

Messaging is typically an asynchronous means of communications from one environment to another. The source of the message (in this case, the application in which someone makes an update) can continue with its own work without having to hook up with the recipient of the message (in this case, the MDM system). The messaging system and its associated protocols handle verification and validation services. Messaging and asynchronous communications give you a great deal of flexibility in architecting distributed environments in which you must send data back and forth across systems quickly and can't afford to tie up any one system while it waits for another to do whatever it needs to with the message.

MDM, along with SOA, provides you the technology platform to deliver a number of feedback loops between several different operational data stores and your run-the-business application portfolio. MDM helps resolve the problem of point-to-point data integration between systems. Before MDM implementations, point-to-point solutions typically resulted in a spider's web of communication lines that were complex to manage and maintain. MDM and SOA provide a robust alternative approach that implements a data message hub architecture which serves as a collection and distribution point for messages across your enterprise. Each application then *publishes* (makes available) a certain set of messages and also *subscribes to* (accesses) other messages that might come from other applications. Each hub keeps a list of which applications are subscribing to which messages and, after receiving any message, distributes that message to the appropriate destinations.

Dealing with Conflict: Special Challenges to Your Data Warehousing Environment

One thing's for certain — data warehousing isn't like most IT projects that have a beginning and an end. The data warehouse, if done properly, becomes the heart of your systems — managing the data assets and assuring their quality and accessibility. You absolutely need a knowledgeable team to properly invest in and manage these data assets if you want your data warehouse to succeed.

Not everyone you come in contact with, or depend on, will agree with you about the importance of your data warehousing project. Many of them, in fact, might think that it's little more than a downright nuisance. Some might take it a step further and think that you came up with this little data warehousing scheme solely to make their professional lives miserable. Others will minimize the importance of the data warehousing — defining it merely as a "dumb reporting solution."

Why would someone define it this way? Consider these factors:

- Because your data warehousing project affects every application from which you plan to extract data, no matter how un-intrusively, the project also affects the staff members who support those applications and their databases (or files).
- The network administration staff, already struggling with major company-wide networking initiatives, now must help you figure out whether the network has enough bandwidth available to support your regular data extraction, movement, and loading procedures — and, furthermore, whether you can secure the data from outside intrusion.
- The *database administration group,* the people who handle all database structure creation and modification, are appalled that you want to frequently make changes to your data warehouse's database definitions — by yourself, without their involvement.
- The program management office (PMO) is wondering why you haven't filled out their mandatory set of templates to be Sarbanes-Oxley (SOX) compliant because the PMO doesn't realize that you're not using their long, drawn-out waterfall technique for delivery (instead going with a more apt agile-delivery technique).
- The business organization whose members today use data extracts along the lines of "they aren't perfect, but they give us mostly what we need" (as discussed in Chapter 22) suddenly find out that they're losing their report-writing tools. To get their regular reports, they must figure out how to use the new business intelligence suite of tools that you're deploying with the data warehouse.

To say it in plain language, more than a few people might be less than enthusiastic about your data warehousing project (and they won't be disappointed if your project is a miserable failure).

You need these people — make no mistake about it. Here's how to handle these situations:

- **Take the "I feel your pain" approach.** Never treat these people like they're members of your personal support staff for your project. And don't treat users like you're doing them a great favor by developing a data warehouse for their use. Talk to these folks, take the time to understand their concerns, and work with someone to find an amicable solution when the concerns are valid (for example, when a database administrator's workday gets extended by 50 percent for the next three months because of the effect of your new data mart project).
- **Address problems and conflicts as soon as you're aware of them.** Don't let situations fester — that's just asking for trouble. Talk to someone, call a group meeting, elevate an issue that you just can't resolve — just do something!

- **Don't make unreasonable requests.** If the normal turnaround time for putting in a database structure change request is two hours in a development environment, don't pout or cause problems because you can't get changes done in a half-hour. Although the situation might not be perfect, it could be worse.
- **Explain yourself.** Don't assume that everyone understands data warehousing concepts, let alone the myriad of technology and implementation options.
- **Never be arrogant.** Don't have the attitude that "data warehousing is where it's at today," especially with the people responsible for supporting the production applications. They hate that attitude, and you'll feel the effects of their wrath. Instead, communicate to others your belief that data warehousing is one part of an effective, state-of-the-art information management solution.

Chapter 21

The View from the Executive Boardroom

In This Chapter

- Telling top management what they need to hear
- Analyzing data warehousing business trends
- Deciding whether data warehousing in a cross-company setting can work
- Getting your organization connected from executive to front line worker

I'll try to say this gently and not upset you too much (it has to be said): Executive management doesn't care one iota about data warehousing.

You might think, "This guy doesn't know what he's talking about. Why, just last week, both our chief operating officer and chief financial officer said that the two data warehousing projects that are under way are two of the most important IT initiatives in the corporation. How could anyone have the audacity to say that executive management doesn't care about data warehousing?"

They don't. It's that simple. They do care about the business value a successful data warehousing project delivers. To be blunt, if your company had done a better job in the past of implementing and managing its information systems and applications, it wouldn't need data warehousing projects, at least the way you're probably doing them or are about to do them (such as copying massive amounts of — mostly — poor-quality data into another database, trying your best to clean up that data, or handling ridiculous situations such as product codes sorted alphabetically in one application and numerically in another).

Just try to pontificate to executive management about the wonders of data warehousing. As soon as you explain what you need to do, you already have two strikes against you.

Although this description is somewhat harsh, the plain truth is that the folks at the top don't care about the mechanics of data warehousing, the methodology you follow, or even the principles of OLAP and data mining.

What *do* they care about? The principle of return on investment, or ROI. *ROI* includes both tangible benefits (expenses avoided or additional revenue generated as a result of the investment made in data warehousing, for example) and intangible benefits, which are more difficult to quantify because they might not have a near-term monetary result.

What Does Top Management Need to Know?

Somewhere in your organizational hierarchy, somebody has control over budgeted funds that he or she can allocate to your data warehousing project or to another project elsewhere in the organization, to purchase capital equipment (more computers, for example), or to pay for some other purpose.

You have to sell that person on the business value of a successful data warehousing implementation and deployment.

And you also have to convince that person's boss. And that next person's boss, and the next person's boss, and all the way up the hierarchy until you present your argument on the business value of data warehousing to someone whose job title has the words *chief* and *officer* in it. Whether that person is the chief operating officer, the chief financial officer, or even the chief executive officer, someone at or near the top must believe that a substantial return on investment will come from successful data warehousing in your company.

All the job titles in the preceding paragraph are from the business side of your organization. Although you do need the chief information officer or the chief technology officer from the IT side of the business to be 100-percent behind both the principles of data warehousing and the investment in your organization's data warehousing projects, IT support isn't enough.

Executives up the organizational hierarchy on the business side don't have to know about the mechanics behind data warehousing. Some of them might have attended conferences where data warehousing was discussed, or they might have read about the subject. (Subliminal suggestion: Every executive should read *Data Warehousing For Dummies.*) They might even be grounded in its fundamentals.

Tell them this

When you're pitching a data warehousing project, work the following two statements into your presentation:

- "We have data all over the place on a bunch of different machines, and frankly, we can't get at any of it. Oh, yeah, much of it's inconsistent, too."
- "You know those reports you and your staff are always complaining about — the ones that always show up too late every quarter to be of any value? And the ones that have incorrect data, and no one uses them anyway? Do you want to fix that problem once and for all, using technology that really works?"

In these two statements, you have addressed both the technology side and the business-value side of data warehousing. The first statement, the data warehousing side, addresses the reality that the company doesn't have one big data bank in which every piece of information is carefully catalogued, organized, and available to everyone in your company who needs it.

Don't laugh. You might be surprised at how many people without an IT background think that corporate systems are much better than they really are. These folks probably get this idea from movies and novels in which you can use a few keystrokes to access, in just seconds, every piece of information about an individual — for evil purposes, of course — or change it, or even delete it, all in a flawlessly seamless way. Hah!

A good dose of reality about the state of corporate data assets is usually a good topic to mention to justify why a project will take more than a weekend to complete.

The second statement addresses the question, "Why bother to do anything with that dispersed, inconsistent data? Why not just ignore it?"

You can talk all you want about scorecards, dashboards, and data mining in your pitch. You can show slides and demos that depict drill-down and other OLAP principles. You can go on and on about being predictive, not reactive. I've never seen anything work better, however, to explain what a data warehouse can do than to put the business-value discussion in the context of what's almost always a point of pain in an organization: dealing with those convoluted reports that are always late, inaccurate, and worthless.

I strongly believe in the hypothesis that every business manager has between four and seven attention points that he or she uses to help run an organization's operations and make decisions. In business life, the traditional way

many managers address these points has involved reports (usually one or two for each attention point) that they use to analyze information and make decisions.

In many settings, any information-driven analysis and decision making (rather than seat-of-the-pants decision making) boils down to the use of reports. That's right — reports. Reports are a concept that most managers and executives can understand, so talk about the benefits in terms of reports when you're trying to sell data warehousing.

So, what's a recommended way to sell the concept of data warehousing and your project, in particular, to executive management? As a much better, much more timely way to generate reports. (I figured the point in the preceding paragraph came across — I wanted to make sure.)

Keep selling the data warehousing project

Anyone who has been involved in any type of application development effort, whether from the corporate, consulting, or vendor side, knows that continued funding is almost always in question. No matter how enthusiastic the support is at the beginning of the project, no matter how many statements they make that "this project is the most important one in the corporation today," and no matter what they budget for your project, budget cuts or shifting priorities can always disrupt a project.

Therefore, from the earliest possible opportunity in the project (90 days at most — 60 days is even better), you need to establish a tangible business benefit for real-life use by the highest-ranking business executive who holds the funding strings for your project. Be sure to

- Provide a set of interconnected reports, dashboards, and OLAP views that you know can help this person perform analysis and make decisions. (***Hint:*** Just ask.)
- Use real, not fake, data (even a small amount).
- Solicit feedback about usability, later steps, and everything else you can think of to ensure that the executive fully believes in your project and (unless something out-of-the-ordinary happens) will continue the funding until you complete the project.

Data Warehousing and the Business-Trends Bandwagon

Business process engineering, six sigma, balanced scorecard, the loyalty effect, and managing in one-minute increments.

I entered the business world in the mid 1980s; in the 20-plus years since then, a number of trends and fads have come and gone. First, the book arrives, then the seminars and consulting engagements that have extremely high daily fees, and then more books by others trying to jump on the bandwagon. In some cases, these events are even followed by the "Uh-oh, I guess we were wrong" books, seminars, and damage-control consulting engagements that have extremely high daily fees.

Okay, maybe that description is a little cynical. Or maybe I'm just envious and bitter because none of my books has ever started a major business or management fad.

Now that the data warehousing era is here, the next generation of business and management trends (you had better believe that a next generation will come along) might have a little more substance — a little more information — that you can use to determine whether a trend is a step in the positive direction or just another fad that will eventually be as useful as a snowmobile in Phoenix.

If you've done your data warehousing correctly and have access to a large amount of corporate business intelligence, you can get the numbers to see whether your company has a problem in whatever area a trendy business approach might address. You can also perform a what-if analysis *before* you spend a gazillion dollars on consulting fees (and disrupt your business processes and organization) to implement whatever's being sold as the answer.

Suppose that data warehousing were as mature in the 1990s as it is today and your organization had successfully implemented a data warehousing environment. Your profits are down, sales are flat, and the boom days your company experienced in the 1990s are gone. Along comes a consulting firm pushing this new idea of lean six sigma and business-process management as the answer to all your corporate woes. You engage a team of consultants to study your business processes and to make recommendations for improvement. If your team of consultants is anything like many others, it spends a

large amount of your money and then makes a standard recommendation: Fire many of your employees and let the consulting company employees outsource the work they leave behind.

(Oops, cynicism creeps in again.)

Before taking such a drastic step, like many organizations did, you could use information from your data warehouse to perform extensive analysis on the effect of such a drastic measure. Using what-if analysis with data pulled from the data warehouse, you might determine that the purported benefits and cost savings from the move just don't exist.

Before following any business trends, try to figure out whether they have any substance. In most situations, it's nearly impossible to determine that information without looking into real corporate data, such as current sales and how they're likely to be affected by a recommended set of actions, current expenses across departments, how proposed budget cuts and staff recommendations would affect those expenses, and how those cuts might affect sales and marketing.

Data Warehousing in a Cross-Company Setting

Data warehousing is usually a private affair. Even when external data about your competitors is part of your environment, as described in Chapter 19, it's still your company's data warehouse, built for your company's benefit and use.

An interesting trend — one that's surely noticeable at the executive boardroom level, primarily because those folks are steering corporations in this direction — is to have multi-company cooperation. Two or three pharmaceutical companies might share the research-and-development expenses on a new generation of drug products, or two manufacturing companies might work together in a partnership to develop a product. A commercial bank and a brokerage institution might work together to offer jointly a series of financial products to the mass market, with the bank administering some products and the brokerage administering others.

Whatever the specifics of the industry and the situation, your company has a good chance of being involved in a multi-company partnership in which cross-company cooperation and sharing of information is a key part of success.

To that end, an interesting spin on the theme of data warehousing as a breaker of barriers is to have a multi-company data warehouse dedicated to more efficient analytical and information-delivery capabilities in support of a joint effort between your two companies.

As you might guess, a multi-company data warehouse is a slightly more complex creature than one dedicated to the support of a single company. Although you experience all the wonderful challenges of data extraction and transformation, tool selection, performance support, and other aspects of data warehousing, you also have to consider these issues:

- **Corporate standards:** Two (or more) sets of corporate standards can affect how you deploy your data warehouse and its tools. For example, one company might be a Business Objects environment, and another might be a Cognos environment. Whose standards will you use, and what's the effect on the other company's users?
- **Proprietary and sensitive data:** The business case for the multi-company data warehouse, and the accompanying functionality and data necessary to support the data warehouse's business mission, might require that sales history information is made available for predictive sales forecasting. Sales history information involves a breakdown by region, territory, and possibly even customer. What are the effects of revealing customer lists, and strengths and weaknesses in various regions, to a partner that's also a competitor?
- **Security concerns:** In addition to data security issues, any linkage between two environments can open up one environment to any security weaknesses in the other, such as unauthorized outside access and hacking.
- **Support costs:** Which organization has primary support responsibility for the data warehouse? Does it bear the full burden of support costs, or will the two organizations share those costs? If the organizations share the costs, how do they calculate support costs, and how do they bill and pay those costs?
- **Development methodology:** One organization might develop its data warehousing applications and environments by using Kimball techniques, but the other organization uses Inmon techniques. Who controls the development processes? Do individuals from one organization have to figure out and use the other's methods and techniques?
- **Dispute resolution:** Resolving any type of dispute in which parties are from different companies always proves an interesting challenge.
- **Ongoing enhancements:** What cross-company management structure must you put in place to prioritize and approve enhancements to the environment?

Connecting the Enterprise

Regardless of whether you create a single- or multiple-company integration, one thing is certain — if you can get the executives the timely information they need in an accurate form, the herds of analysts they employ can now focus on analyzing the data, rather than assembling it. And, furthermore, the executives can begin connecting the strategy of the enterprise with key metrics by using the organizational layer. Each person up and down the food chain can have the four to six metrics that he or she focuses on — and possibly are compensated on. With this level of connectivity and focus, the company can much more easily deliver upon a strategy.

But you must take that first step — convince your executive sponsor or team that the enterprise from an intuition-based decision-making model to a fact-based one. In that simple statement, you go from the decisions being made "because my gut tells me that is the right decision" to "based on the trends we should be investing in . . . or change our direction."

Which type of organization would you rather oversee? I'm sure you're no different than many executives or shareholders: The proof is in the execution, and to execute a data warehouse, you need executive leadership.

Chapter 22

Existing Sort-of Data Warehouses: Upgrade or Replace?

In This Chapter

- Taking a complete inventory of your organization's extract files
- Deciding what to do with each extract file
- Mastering the complexities of migrating extract files to a real data warehouse
- Recognizing that you (almost) never want to take away user functionality

Your organization has overwhelmingly favorable odds of having at least one *sort-of data warehouse* — a reporting system that provides informational capabilities and, sometimes, analytical capabilities to one or more groups of users. Your users probably use the term *extract file* to describe this type of environment because it's populated by extracts of data from production systems, rather than by users being forced to execute their queries or receive their reports from the operational production databases or files. Still interested in playing the odds? Here are a few more examples of types of data environments that might be described as sort-of data warehouses:

- Although the extracted data is almost always housed in a single file or database, a merge process probably combines extracted data from more than one application source.
- Only selected elements, not all elements from all tables or files, from each data source are usually extracted and copied to the extract file.
- Some sort of data quality assurance process is usually going on each step of the way, from the initial extract to loading the data into the extract file.
- Some power users probably can execute queries or create statistical programs (in SAS or SPSS, for example) against the data, but many users aren't likely to touch the data directly. Instead, they probably regularly receive reports generated either automatically or in response to their requests.

Sure sounds like a data warehouse, doesn't it? The reality is that these sort-of data warehouses are typically serving a very small population and are not done in a standard manner to support the broader needs of the enterprise. You might also call them *wanna-be data warehouses.*

The Data Haves and Have-Nots

About two years ago, I began working on a data warehousing project with a fairly large bank. I was working with the retail part of the bank, in which individual consumer (credit card, lease, and mortgage) loans were handled. (The wholesale side handled large commercial loans and other financial transactions with companies.)

I set up a kickoff meeting with the project's sponsor, who outlined all the (numerous) initiatives the project encompassed. We put together a time line of which areas to tackle first and which ones would follow, and we then set up a series of meetings with different managers and technologists in the various business groups in the bank's retail organization.

The first meeting was held with a group responsible for analyzing how the bank's customers handled their payments for consumer loans: how many paid on time, paid only the minimum amount due each month, and defaulted on or missed payments, for example. This group's goal was to manage the bank's risk against its entire portfolio of loans and to perform statistical analysis in an attempt to predict where problems might occur. To accomplish this goal, the group would look at historical payment patterns and perform trend analysis and other statistical functions.

The meeting began with the project sponsor explaining what the group was planning for the bank: Build a data warehouse to support risk management for consumer loans in addition to other functions handled by other departments. We spent the rest of the meeting with the loan payment analysts describing how they did their analysis, what data they used and where it came from, and other pertinent information.

A few days later, I was speaking with the manager of the group who had been in the meeting. He said, "You know, I don't understand this data warehousing thing. We already pull key data elements from our production systems into an Excel spreadsheet — I personally hand-picked which elements we need — and we've been doing this type of analysis for about two years. Tell me again, what additional benefit will I get from a data warehouse?"

To complicate matters, the second group meeting was with another organization responsible for analyzing credit bureau scores for the bank's customers. This group would periodically receive, from a provider of external data (refer to Chapter 19), credit bureau data about anyone who had any type of line of credit (including a bank card) or loan.

This meeting began much like the first one, except that the project's sponsor wasn't there and I was facing the group alone. At their request, I explained the principles of data warehousing and how data would be pulled from various sources and made available for analysis. Next, they took turns explaining to me how they performed statistical analysis with their own data extracts that merged data from different sources and analyzed bankruptcies, delinquencies, and a lot of other measures in concert with the credit bureau data.

Again, the request was made: "Explain again what new functionality a data warehouse will give me."

Why aren't extract files considered to be data warehouses?

They are, sort of. Chapter 1 describes the historical roots of data warehousing in the 1970s. Extract files, whether in the 1970s, 1980s, 1990s, or still in use today, exist for the same basic reasons that a full-fledged data warehouse or a data mart does: to provide information delivery despite a variety of barriers, such as hard-to-understand data structures, "don't touch the production system" rules, and the lack of multi-file or multi-database cross-reference.

Some data warehousing proponents argue that combining and reconfiguring data simply for the purpose of generating reports or to perform statistical analysis is hardly a data warehouse in the modern sense of the term. Extract files aren't equipped with multidimensional or business-analysis capabilities, such as drill-down and data pivoting.

I argue that if you separate the data-warehouse side (what it takes to gather, move, and reconfigure data from one or more sources) from the business-intelligence side (what you do with the data after you have it available), the picture becomes much clearer. Extract files, or whatever you want to call them, are very much part of the barrier-breaking philosophy of a data warehouse. Many of what users refer to as "extract files" are file-based systems (rather than built on databases), and they probably aren't flexible enough to support ad hoc querying and dimensional analysis. In a real sense, however, these environments serve the purpose of warehousing data for subsequent use.

To many users, business-analysis capabilities, such as drill-down and data pivoting, have little or no use — at least, not in the context of their current job definitions. The users' jobs call for functionality that these extract files can deliver, as well as the static reports and statistical analysis accomplished with that data.

I'm not saying that everyone should scrap their data warehousing initiatives and continue to run on top of existing extract files. But, for as much interest as data warehousing has generated since the early 1990s, the premise of "getting the job done" is the same old story for a large segment of the business community.

The moral of the story: Don't go into an organization that effectively uses data through extract files and expound on the wonders of data warehousing. Instead, as discussed in the section "Choice 3: Retain It," in this chapter, be cautious about proposing any data warehousing solution that can be viewed as a step backward. If you do make this kind of proposition, you're in for a long, bumpy ride.

The real answer to this request, made by both groups, was, "In your particular situation, not much." Doing the tap dance of a good data warehousing consultant, I explained the big picture: It wasn't just their own respective groups, but also many other groups on the retail side of the bank, who needed that same information. The data in its current form just wasn't available or wasn't in a flexible enough form to be used in other ways (business analysis, for example) than the statistical processes that were being used.

Honestly, each individual had the right to ask, "What do I gain from this data warehousing project?" Neither organization had its own data warehouse or was doing its processing against a data mart extracted from some larger organizational data warehouse. What they were doing, however small-time it was, was well suited to their respective needs. Their business needs didn't call for business analysis, OLAP-style functionality, and other advanced business intelligence capabilities. Summarized data? "Sorry, no can do." These folks needed detail-level data for their statistical analysis. Drill-down? "Not interested, thanks for asking." They needed only statistical processing to do their jobs, and mainframe SAS did that just fine, thank you.

Note: Other groups on the retail side of the bank did need to do drill-down analysis by region against summarized data. To their managers and analysts, the idea of a data warehouse supported by business analysis tools was an enticing, high-value proposition because they didn't have anything similar at the time.

Here's the dichotomy of most organizations access to data:

- **Data analysis "have-nots":** Organizations and individuals who have few (and more likely no) capabilities to do the type of analysis that can bring about information-driven decision-making
- **Data analysis "haves":** Organizations and individuals who might not have a data warehouse up and running, but are doing something with data that they're getting from somewhere. In many cases, it's suiting their business needs just fine.

The first step: Cataloguing the extract files, who uses them, and why

Before you even begin to consider what to do about the extract files and other types of sort-of data warehouse environments that exist, you must find them — which you might find difficult, considering the homegrown nature of these environments.

Here's a hint: Follow the reports. Through group work sessions and individual meetings, determine and catalog the reports that employees use throughout the organization that you're working with to build a data warehouse. Some of those reports probably come directly from the production applications, and

their respective databases and files. For now, don't worry about these production application reports. (Keep track of them, though, because you can use them as an excellent starting point for the "what data do we need?" analysis, which determines what you want to put in the data warehouse.) Other reports come from data extracted from one or more applications and stored somewhere. Those reports are the ones to concentrate on now.

Using the set of reports as your starting point, first determine who's using them and who's responsible for generating them. You need to know who is using the generated reports because you might find that nobody uses them anymore. Just by assessing the inventory of reports and their current utilization, you're halfway finished with eliminating this don't-really-use-it functionality from your data warehousing environment (and managing its complexity).

Once you understand the report usage patterns, get to know the people responsible for generating the reports. They're the ones who probably can tell you where they get the data, what processes they use to prepare and load that data before running the reports, and what issues and problems they have with data availability and integrity.

Sometimes, no single individual knows the entire end-to-end sequence of steps used to extract data, prepare and organize that data, and run the reports — especially when these processes cross organizational boundaries. (For example, the IT organization handles the initial extraction of the data and some rudimentary quality assurance, and the business organization handles the merge processes and runs the reports.) In these situations, get all these people in the same room to discuss and agree on how things work. You can avoid spending a great deal of time playing "he said, she said" with people who, frankly, you're probably aggravating with your constant questions and requests for meetings.

Eventually, through diligence, you get a complete picture of who's using which data, who's responsible for making that data available, and what's going on behind the scenes to make it all happen.

Don't forget the why part of the picture — for what business purposes extract files are being used. You have to find out this information anyway, as part of your requirements analysis. While you're checking out what's going on today, ask questions while you have the users' attention, such as:

- Why do you need the information in the report?
- What decisions does this report assist you in making?
- When you analyze the data on the report, who do you communicate your findings to?

Your line of questioning should follow a traditional understanding of what the report's requirements are as discussed in Chapter 15. Getting a firm understanding of these questions at this analysis point will save you time in the long run.

And then, the review

Don't skip this section. If you do, serious problems lie ahead.

After you create your catalog of information, you must embark on a candid, no-holds-barred review process with each organization that uses extract files. Although you'd ideally make this process part of the data warehousing project's scope (Chapter 13 discusses how to choose a methodology), schedule conflicts and timing constraints might require additional meetings to complete the review process. To help you establish the proper project scope, conduct these meetings — to gain a firm understanding of the extract files' use and purpose — as early in the process as possible.

You want to determine the true business value that the organization obtains from using these extract files and what additional business value that organization might obtain. You have to take this approach with the extract file users: "Forget this data warehousing stuff — tell me about your extracts." They're already likely to feel that the evil data warehousing empire is about to take away their capabilities (as discussed in the section "Beware: Don't Take Away Valued Functionality," later in this chapter).

While the data warehousing project moves forward, figure out what part of the existing environment is worth salvaging (not only functionality, but also steps and processes).

Decisions, Decisions

If you have been diligent, you've now ferreted out a complete inventory of extract files that fill the role of prehistoric data warehouse, each one most likely serving a single organization's needs. Next, you and the business users have reached consensus about what's good about each file, what needs to be improved, and other aspects of the data use throughout the organization.

Now, it's decision time.

Although it might come as a surprise to you, you have no reason not to build a data warehousing environment to contain one or more existing extract files. Although a single, monolithic data warehouse probably would have difficulty interacting with these extract files, an environment constructed in a mixed-mode, component-oriented manner (refer to Chapter 13) can encompass a long-in-the-tooth component or two.

You must test each extract file (as I describe in the following sections), and then you will be left with one of three answers regarding what to do with that extract file:

- Discard it.
- Replace it.
- Retain it, possibly with some upgrades or enhancements.

Choice 1: Get rid of it

If, and only if, you have universal agreement from every corner of the organization that an extract file has absolutely no use (for example, no one looks anymore at the reports it generates, and no one has updated the data-extraction and data-input processes, so the system is generally doing little, other than wasting disk space and processing time), you can probably just dump the file without any type of replacement or upgrade.

Although some individual data elements might eventually find their way into the data warehouse, they have no business value in how the data is currently organized (data organized in a specific way).

Be brave: Throw it out.

Choice 2: Replace it

Suppose that an organization does actually use an extract file, but, honestly, that file is somewhat cumbersome and difficult to use. It doesn't include all the necessary information, users struggle to change the extraction processes to extend the list of attributes, and it meets only about 50 percent (or less) of the users' needs.

In this situation, replace the file by following these steps:

1. **Retain the extract file's functionality that users want as part of your data warehousing environment.**
2. **Create designs and plans for the functionality that users need but don't have in the extract files.**
3. **Replace the extract file by folding the existing functionality into the data warehouse, along with the newly designed features.**

   Just get rid of the old, antiquated environment.

Choice 3: Retain it

If these conditions exist with the current extract-file setup, strongly consider retaining this type of environment:

- The extract file is a relatively recent addition to the organization's capabilities (within the past one to three years, for example).
- The data is stored not in a flat file, but rather in a relational database.
- The data quality is excellent.
- Users have access to basic reporting tools and are doing some degree of ad hoc querying on their own.
- The environment generally doesn't look too bad. (A grandmotherly saying is appropriate here: "Your data warehouse should look and feel so good when it's that age.")

Although you might want to consider upgrading the extract file a little (you can read more about upgrading in the following section), definitely don't throw it out and try to replace it.

These kinds of upgrades might be appropriate:

- Add data elements closely related to ones that already exist (their point of origin is the same application file or database, for example), which people in other organizations might use.
- Do a little performance tuning to increase response time if a larger group of users will access the database.
- Increase the frequency of updates, and improve the freshness of the data if business needs dictate doing so.
- Equip users with new tools, in addition to their existing ones, to expand their horizons in business analysis and use.
- Add — only if you're daring — a new subject area or two to this environment to provide an even richer set of data for business analysis purposes. Be careful to avoid disrupting existing functionality.

If you're considering adding new subject areas to your extract, look at an extract file that's worth salvaging as a '67 Corvette that needs a little work and has been garaged for a while. Although you can always sell the car and buy a new one, when you step back and consider all your options, it's probably less expensive to invest in the required maintenance. Besides, you're already familiar with the car. It might not be new, and it doesn't have all kinds of advanced

computer controls, GPS, and antilock braking; it might not have a back seat or a rear window; but it still gets you where you want to go. ("Yeah, we're going to Surf City, gonna have some fun. . . ."

Caution: Migration Isn't Development — It's Much More Difficult

When you decide to replace an extract file (discussed in the section "Choice 2: Replace it," earlier in this chapter), you face a migration situation: moving functionality and data from an existing environment to a new environment.

Because I can't possibly describe in just a few short paragraphs all the complexities of migration and how to deal with them, I hit just the highlights in this section.

Unlike a development project, in which no functionality exists, a migration project requires that you not only develop and deploy capabilities in a new environment, but also retire that same functionality from the system you're replacing. You have to consider these issues:

- **Cut-over requirements:** For example, what functionality moves, in how many phases, and when?
- **Your fall-back plans:** For example, what happens if the new system doesn't work?
- **Additional staff requirements:** Potentially, you need additional staff to support two environments simultaneously, if you have a multiphase migration plan. And you might need special staff to train employees in new technologies and products, for example.
- **User training in the new tools and contents:** What new data is available and how can (and should) it be used?

The process of migrating an informational and analytical environment, such as an extract file and the processes used to populate it, is often significantly less complex than migrating a transaction-processing system that's mission-critical to your organization, such as the customer order-entry system or the payroll system.

Don't overlook the complexities of migration, however, even when a data warehouse is a target.

Beware: Don't Take Away Valued Functionality

If you decide to pursue a replacement-migration strategy for your existing extracts, please pay careful attention to this section. (Don't say that I didn't warn you.)

One of the worst things you can do (something that causes users to snarl every time your name is mentioned) is implement a newfangled, much-improved, all-kinds-of-gadgets data warehouse and, at the same time, take away functionality and capabilities from the users of existing extract files.

Oh, sure, you might have a certain feature on your Phase II list and another one on the maybe-next-year-if-we-get-funding list. But average business users don't care whether answers are provided by a data warehouse that has all the bells and whistles or a plain-vanilla extract file. They care only about reports, business analysis, statistical analysis, or whatever else they need to do.

Keep in mind that functionality in this context might equate to certain data, such as the facts used to support decision-making or provide analytical capabilities. Your new data warehousing environment might have several reporting and OLAP tools that support querying, generate reports, or perform statistical analysis, just like the old environment did. But if users can no longer access data that was in the old extract files because that data isn't in the data warehouse, you've done it — you've taken away functionality.

You don't need to propagate every data element into the data warehouse or automatically generate every report by using the new tools. Getting down and dirty with users about what business value truly exists (like I talk about in the section "The first step: Cataloguing the extract files, who uses them, and why," earlier in this chapter) can help you deliver a high value data warehousing solution. Don't spend time and effort moving data and functionality that no one uses; at the same time, don't take away necessary capabilities. If you do remove functionality, the users will hunt you down in a manner similar to the villagers going after Frankenstein's monster — with pitchforks and torches — to burn you in the public square.

Chapter 23

Surviving in the Computer Industry (and Handling Vendors)

In This Chapter

- Shopping the smart way at conferences and trade shows
- Dealing effectively with data warehousing vendors
- Looking at the future of data warehousing and mainstream information technologies

The name of the game in data warehousing since the early 1990s has been hype. A lot of hype. Massive amounts of hype. Hype. Hype! HYPE! Whew — I had to get that out of my system. I'm serious, though. Whether you're talking about business intelligence suites (the best-of-breed versus highly integrated), OLAP tools (ROLAP versus MOLAP), middleware tools (data-movement-ETL versus federated-EII), or any other aspect of data warehousing, you may have a lot of trouble separating merit from empty promise, fact from fiction, in the wonderful world of data warehousing.

Although data warehousing has more substance and provides more value than many of its predecessors on the hype hit parade, the discipline has, in my opinion, been somewhat compromised by the usual bag of tricks that the shiftier side of some product and service providers (vendors and consultants) like to pull out.

Although I'm not one to point fingers, anyone who has been to a data warehousing trade show or sat through a couple of product demonstrations (and then tried to implement a real-world data warehousing environment with products that just didn't work as promised) knows what I'm talking about. A world of difference exists between looking at a few glossy brochures before watching an oversimplified drill-down demonstration and facing real-world problems while you try to implement a data warehouse (for example, struggling with performance and response-time problems, or integrating fact data that has mixed levels of detail).

Hype: Is it all part of the game?

Back in the early 1980s, the new Ada programming language was the answer to everyone's software development woes, especially for military and government systems. Ada was supposed to solve everything. It was the language of the future, built especially to overcome all the shortcomings of every other programming language in existence at that time.

In the mid-1980s, computer-aided software engineering (CASE) tools were touted as the answer to software-development woes. You drew your data models and process flows by using the tools, and — presto! — out came a complete application, ready to go. This was productivity, brought to you by the wonderful world of CASE.

Back in the late 1980s and early 1990s, enterprise computing architectures, such as the IBM SAA, the Digital Equipment NAS, and nearly a dozen others from different hardware vendors, were the answer to the problems and challenges of distributed heterogeneous (multivendor) computing. You just wrote all your applications to use the application programming tap-dance interfaces (APIs) for this set of standards, and before you knew it, you had a seamlessly integrated enterprise across multiple platforms.

The mid- to late 1980s held the promise of artificial intelligence (AI) and expert systems shells. A little forward chaining here, a little backward chaining there, throw in a neural network, and — poof! — you created a "thinking" application that could aid your decision making.

The list goes on, including first-generation client/server computing (remember the promise that you could save millions of dollars in maintenance costs as compared to your mainframe?) and specific change-the-industry standards efforts that no longer exist, such as Unix International and the Distributed Management Environment (DME) standard from the Open Software Foundation.

More recently, the claims of service-oriented architecture (SOA), software as a service (SAAS), and Web 2.0 bring the same vendor promises of the past.

The common theme was highly touted silver-bullet solutions from vendors (sometimes one and sometimes a pack of them operating in a consortium) that provided significantly less benefit to you (and often no benefit), despite the thousands — sometimes tens or hundreds of thousands — of dollars you spent on their products and services.

Here are some data warehousing facts of life:

- Vendors are in the business of making money by selling products.
- Vendors' sales representatives make money if they sell products to you and others. If they don't sell those products, they don't make money and can even lose their jobs.

- Data warehousing consultants would love to tell you everything you have to know to successfully develop a data warehouse in your organization and maybe would even want to help you develop it. But they have to regale you with the wonders of data warehousing and the fabulous benefits you can gain and — oh, yeah — give you the reasons why you should listen to and employ them on your project.
- Within every IT discipline (data warehousing is no exception), stories get old, and you're bombarded with products with this theme: "Okay, maybe the old version didn't work quite the way you wanted it to, or maybe it wasn't as scalable as you needed it to be, but guess what? Our new version has all these new features."

The challenge you face involves dealing with these issues in a productive, non-confrontational manner without being steered down a path that doesn't make sense for you. In this chapter, I tell you how.

How to Be a Smart Shopper at Data Warehousing Conferences and Trade Shows

Make no mistake about it: You can gain tremendous value from attending any type of conference or trade show, including those oriented toward data warehousing. Vendors tout and demonstrate their latest products, you get to hear real-life case studies and stories about successful data warehousing implementations, and you can gain unique insight into up-and-coming problems you're likely to face from consultants and others.

When you attend a data warehousing event, behave like you're shopping for a car or other expensive personal item:

- Do your homework before you attend.
- Ask a lot of questions.
- Be skeptical.
- Don't get rushed into a purchase.

The short version: Be a smart shopper.

Do your homework first

When you register for a data warehousing trade show or conference, you typically do so from a schedule on the Internet that gives you a complete list of vendors who'll attend, consultants and other speakers who'll give presentations, their respective topics, special sessions and seminars, and hospitality suites and other services you can use to spend more time researching your specific data warehousing needs.

Plan your entire agenda well in advance of the event. Most conferences have a number of parallel session tracks, with anywhere from two to five simultaneous lectures and presentations. After reviewing the entire agenda for each day, mark for each time block the topic that's most interesting or pertinent to you. In addition, mark a secondary topic, just in case the presentation you had as your primary is cancelled or you lose interest in the first five minutes — always have a back-up. Don't be distracted by the headings given to tracks, such as systems track or OLAP track. Plan your schedule by topic.

If you're attending an event with other people from your organization, split up whenever possible and cover as many sessions as you can.

Ask a lot of questions

Asking questions at presentations and during demonstrations at vendor booths is not only permissible, it's also encouraged. After all, you're there to find out as much as possible about specific techniques, experiences, and products. Don't use your question-asking time, however, to do any of the following:

- Show off how much you know (or think you know) about a particular subject.
- Pointedly contradict or embarrass a speaker, especially on matters of philosophy that have no right or wrong answers and are just different ways of doing things.
- Do or say anything that reflects negatively on your company (which probably appears prominently on your name badge).

Although these statements may seem somewhat silly, most of us have had the experience of attending a session that's continually disrupted by an audience member who argues with the presenter and who seems to be doing little other than trying to draw attention to him- or herself. Don't be one of those people.

Be skeptical

Wait a minute. In the previous section, I say that contradicting a speaker or presenter is considered to be in poor taste. How, then, could I now advise you to be skeptical? Simple. First and foremost, be skeptical. When you hear about "revolutionary new features," "order-of-magnitude increases in performance over our previous product version," or anything else that sounds a little too hype-tinged, say to yourself: "What has changed so dramatically in the past few months that suddenly the product offers all these wonderful new capabilities?"

Next, ask questions privately or in small groups, not in a large forum in which a speaker may feel defensive. If you see something that seems too good to be true during a demonstration, ask the presenter after the session is over. If the waiting line is too long, go back later; that person, or someone else who works for that vendor, will still be there. An even better idea is to ask several people at a vendor's booth the same question on different days and see whether you get consistent responses.

Although your questions may be somewhat general at first, try to present a specific, real-life example from your environment as the context for digging into whether a feature or capability would truly benefit your data warehousing project. You may say something like this, for example: "We looked at Version 3.1 of your product six months ago and ran some tests against a demo copy to check on performance. Although we were okay with 250 gigabytes of data, as soon as we went above that number, volume performance was terrible, even though we had only two fact tables and four dimensions. You mentioned that response time with 500 gigabytes in the new version is as good as we used to get with 250 gigabytes. Please tell me what has changed in the new version to make performance so much better."

Don't get rushed into a purchase

You may be faced occasionally with a limited-time-only offer of a steep discount on a product, but only if you order before the end of the trade show.

Never buy a data warehousing product at a trade show. Okay, maybe a book (such as this one, if you're thumbing through it while standing in a trade show booth) or another low-priced item, but never buy a business intelligence tool, middleware product, data quality assurance tool, or an entire database management system at a trade show.

Use a conference or trade show as a fact-finding mission. Collect the glossy brochures and white papers. Take home the conference proceedings with the presentation slides, but don't buy on impulse.

Dealing with Data Warehousing Product Vendors

The same basic smart-shopper guidelines that you use at trade shows apply when you deal with data warehousing product vendors:

- Do your homework.
- Ask a lot of questions.
- Be skeptical.
- Take your time before committing to purchasing products.

The one-on-one nature of the vendor-customer relationship is somewhat different from the contact interaction that occurs at a conference or trade show. On the positive side, you (and your data warehousing needs) can get much more attention from a vendor when you're meeting in your office to discuss your data warehousing project and that vendor's sales representative is trying to make a sale to you, compared to the trade show cast-the-net approach, when a vendor tries to reach as many new prospects as possible.

On the negative side, though, the vendor's sales rep can focus tactics specifically on you and others in your organization, and you must be particularly cautious regarding the sales techniques that the rep uses.

A product sales representative isn't in the business of solving your business problem through a cost-effective, timely data warehousing solution. Although that person (and, in a larger sense, the product company as a whole) wants very much for you to be successful in your data warehousing endeavors (particularly so that your success story becomes a reference for them), never forget that their product sales and revenue take precedence over your budget and schedule.

Check out the product and the company before you begin discussions

Any category of data warehousing product (such as OLAP tools, data mining tools, basic reporting and querying tools, database engines, extraction products, data quality tools, and data warehouse administration and management tools) has a lot of different products. Each vendor that makes one or more of those data warehousing products wants to involve you in a one-on-one discussion with a sales representative.

But your time is a valuable commodity. Even if you weren't dealing with project schedule pressures, you and others in your organization should give only a finite amount of time to vendor meetings.

Do your research at trade shows and conferences (and, as noted in the section "Do your homework first," earlier in this chapter, even before you go to the trade shows). See which user interfaces look most appealing, and study performance and response-time statistics (as many as are available) for database volumes similar to what your environment will have. Don't just lug around those glossy brochures you pick up at the trade show booths — read them.

Next, get on the Internet and check out vendors' Web sites. Use an Internet search engine to retrieve product reviews, analysts' comments, news releases, and anything else you can find about the vendor's company (history, financial strength, and what others say about them, for example) and the products in which you're interested.

Request or download a demonstration copy of the product, if one is available.

Then, you're ready to talk to a sales representative in person or over the Web, assuming that the vendor and their wares seem to fit your needs.

Take the lead during the meeting

Before a vendor sales representative sets foot inside your office or initiates a Web conference with you, make absolutely clear what you expect to cover during that one- or two-hour initial meeting. You should do at least the following:

- Hear a presentation of no more than ten minutes about the company's history and background, as well as the background of the products you're discussing.
- See an initial, end-to-end (or as close as possible) demonstration of a product's capabilities. The demonstration should last no more than a half-hour.
- Have a list of specific questions that cover features and capabilities, product installation base (how many copies have been sold and used and at how many companies), new version enhancements and features, and product architecture (interfaces, different platforms supported and differences across platforms, and scalability). Again, make these questions specific to your environment and data warehousing project.

During discussions before the vendor meeting, give some, but not all, of the questions to the vendor representative. State that those items are important to you and that you want to discuss them during your meeting. You can gauge vendor responsiveness by how well the representative answers your prepared questions and also compare those answers to the responses that he or she gives to your impromptu questions. With the latter, you can gauge how well vendors know their own products and how well they react to unanticipated questions and challenges (rather than just hear scripted responses).

Be skeptical — again

You need to have down-to-earth, open discussions with your product vendors (not chats held in a crowded booth at a trade show). If you've heard certain things about a product that concern you (product scalability above a certain number of users, for example), ask! Demand proof (reference sites, discussions with a development manager, and hands-on testing in your organization, for example) of anything and everything that concerns you.

Be a cautious buyer

No matter how attractive a product looks, take your time in committing to a purchase. It's software, not a one-of-a-kind work of art. You have no reason to hurry, even if your data warehousing project has an aggressive schedule.

Always test-drive software under your environment's conditions:

- Use your data.
- See how many attempts it takes to install the software correctly.
- Determine the responsiveness of the vendor's support staff.
- See what works as advertised — and what doesn't.
- Ask what your users think about the product's usability.
- Find out how stable the software is. Does it cause your client and/or server systems to crash or lock up? If so, how frequently?
- Find out what performance is like in your environment.

Sometimes, a desktop product (a business-rule design tool, for example) is suitable for only a certain portion of your user base. Perhaps only a small number of power users would use that tool, and the rest of the user community would use a basic reporting tool or, for the first iteration of the data warehouse, no tool. Perhaps casual users would use a standard browser to access standard

reports posted on the company intranet. In these situations, never let a vendor pressure you into purchasing more copies of a product than you need. You can always buy more later, if necessary. The last thing you want is a bunch of shelfware sitting around and taking valuable funds out of your budget.

A Look Ahead: Data Warehousing, Mainstream Technologies, and Vendors

While data warehousing and traditional computing technologies converge, you're looking at a whole new ball game when you try to sift through vendors' claims and promises. Traditional data warehousing vendors are already trying to make their products' respective capabilities *go enterprise* (be able to work in large, enterprise-wide global settings), and others vendors see the lucrative data warehousing market as an area into which they can expand.

Beware! I've already seen more than a few 1980s-era marketing messages from the distributed database world make a comeback:

- "This product provides transparent access to any data in any database anywhere."
- "Put one subject area into this data mart and another subject into a second data mart, and — presto! — you can join them whenever you need to and treat the two data marts as one logical data warehouse."

Although I'm not one to scoff automatically at everything I hear about new and improved product capabilities, much of what's showing up to address shortcomings in first-generation data warehousing has a feel (to me, anyway) of "been there, done that." I don't mean to imply that they don't work; it's just that many of the capabilities from "best of breed" products are now appearing in the "mainstream" products.

The proof is in the pudding. (I have no idea what that saying means, and I don't think that it applies to data warehousing. It just seemed like the right thing to say.) Dealing with vendor promises and claims (and the consequences of product characteristics that you wish you knew about before you bought those products) will become even more of a burden when the extract-and-copy-and-copy-again first-generation data warehousing morphs into a more advanced data warehousing solution including features discussed in Part VI of this book, which describes topics such as near-real-time updates to the data warehouse, the access of data at its point of origin (rather than from the database into which you copy it), and the inclusion of multimedia in your data warehouse.

Chapter 24

Working with Data Warehousing Consultants

In This Chapter

- Deciding whether you need data warehousing consultants for your project
- Being careful when choosing a consultant
- Considering all your consultant options

Let me begin this chapter with a disclaimer: I'm a data warehousing consultant and a data warehousing product vendor — a double whammy!

I try to treat the subject of whether you need data warehousing consultants fairly in this chapter. You won't see any subliminal advertising (call Tom . . . blink, blink . . . call Tom) anywhere in this (use Tom for your data warehouse needs) chapter.

The reason I discuss data warehousing consultants is simple: You'll most likely need consultants for your data warehousing project, and I give you as unbiased a perspective on this subject as I can, like I did in Chapter 23, which deals with the product vendors.

Do You Really Need Consultants to Help Build a Data Warehouse?

You probably need data warehousing consultants, not because people from within your organization aren't capable of working with data warehousing technology or completing a project without outside help. A simple fact of current corporate IT life overwhelms factors such as capabilities and knowledge: We're in a consulting-driven era, plain and simple.

I've been in the software and consulting industry since the early 1980s, and the demand year after year for outside consulting expertise in nearly every medium- to large-size organization has skyrocketed remarkably. The increased demand for consulting services partly stems from the convergence of two major factors:

- **The increasing pace of technological change:** Client/server computing, the Internet, and improvements in relational database capabilities, for example.
- **The aftereffects of the U.S. economic recessions and downturns in the early 1990s, 2000s, and the current market:** Although some areas were hit harder than others, companies across the board were hit hard by downsizing, reengineering, and the rest of the slow-economy stuff that we all know so well. Corporate IT organizations continue operating under the "stay lean and mean" philosophy.

The combination of these two factors means that outside consulting organizations are doing (at least, in part) an overwhelming majority of new systems development, including data warehousing.

You can always fight the trend and try to do in-house data warehouse development, and you can easily be successful. Your internal IT staff is probably spread fairly thinly across a multitude of initiatives, however, and applying for personnel requisitions for a major hiring plan is just plain out of style in most corporations. (Can you say outsourcing? Off shoring?)

I wholeheartedly encourage you to try to accomplish your data warehousing initiatives through the use of in-house staff, either exclusively or with the majority of your team members, at least, being employees of your organization. If your organization is like most others, though, your data warehousing fate rests solely on how well you identify and use outside data warehousing consulting expertise.

Watch Out, Though!

If you've ever worked with IT consultants (data warehousing specialists), you realize that not all consultants are equally skilled, equally dedicated, or equipped with the same, shall we say, degree of ethics.

Before getting into the aspects of individual consultants and the roles they can play on your data warehousing team, let me distinguish between different types of consultants and their relevance to your project.

A large part of the growth in consulting services in the 1990s and into the 2000s has been in the area of consulting formally known as *staff augmentation,* and less formally known as body shopping. A number of consulting companies have been successful in creating a collection of individuals (sometimes employees of the company, sometimes independent consultants, and sometimes both) and placing these individuals, one or two at a time, in organizations looking to fill a personnel gap here and there.

To be fair, many consulting companies that perform primarily staff augmentation work put together small-scale project teams of four or five people (rather than provide a person here and there) when they have an opportunity to do an entire client project. These firms typically steer away from project-oriented work, however, because placing individuals in staffing augmentation positions is, to be blunt, a relatively low-risk way to build a company.

If your organization has an ongoing data warehousing initiative staffed primarily by internal IT members, but your team has a few open slots, you might best be served by working with a staff-augmentation-oriented consulting company to find the one or two missing links on your project team.

If you're a data warehousing consultant

This chapter is oriented primarily toward the user community and how the people in it might find and retain you (or your consulting brethren) for their data warehousing projects. If you're a data warehousing consultant, though, here are some tidbits of advice:

- **Stay on top of all the changes taking place in the field of data warehousing.** Have you heard of hybrid OLAP? Vertical databases? What are your thoughts about data mining? Is it for real? What are the new products in the middleware game for federating data? Although you never know everything that goes on in this or any other technology area, you have to avoid getting caught by surprise during an interview or business development session if you're asked your opinion about a new product or architectural trend.
- **Remember that it's not only technology skills that make a successful data warehousing consultant, but also business skills.** Although I don't believe in the industry-expert approach to data warehousing implementation, you must have a core knowledge about business to be able to ask the right questions during the scope process or to create, for example, a definition for a sales-and-marketing fact table.
- **Be flexible.** Be able to work as a hands-on technologist (performing source systems analysis or data modeling, for example), a data warehousing architect or advice-oriented consultant, a project manager, or other types of roles. This flexibility helps you stay billable (whether you're independent or working for a consulting company) and generally helps make you a better consultant by giving you a broader perspective of both data warehousing and the consulting profession than if you continually play the same role in one engagement after another.

Be careful, though: If you're paying good money (and you will, believe me) for the services of these individuals, be prepared to complete a thorough interview process to ensure that they're not only technically qualified but also a good cultural fit for working on-site, or increasingly remote, within your organization. When a consulting company uses subcontractors, they're likely to have little or no history. They might have been chosen for your project based solely on a few keywords that showed up on a résumé database search. Make sure that you carefully determine whether these people are the right fit for your data warehousing project.

The other side of consulting, the part dominated by large systems integrators, is oriented primarily toward project work, rather than staffing services. Many of these firms eschew staffing work and engage a client only (or primarily) if the client gives the firm control over a project's methodology, the resources assigned to the project (even those from your own company), and the format of deliverables.

Although this rigid stance might sound harsh at first ("Do it my way or else, even though you're paying me"), a sound theory is behind it. Some firms are so experienced at putting together successful project teams and performing in a certain, methodology-driven manner that to do otherwise can lead a client's project to fail — so, they continually adapt their processes and techniques for each client.

In the area of data warehousing consulting, it's almost comforting to work with a firm that has these qualities:

- A successful track record of data warehouse implementations, using a variety of technologies (for example, not just data warehouses or data marts, but both)
- Insight into the direction of data warehousing technology and architecture as they apply to your business problem, rather than a canned solution ("We've always done it this way") that might not be a good fit for you
- A commitment to the success of your project by taking on full responsibility for all aspects of implementation, rather than offering a supply of technologists who don't assume responsibility for project management and direction

In many situations, the lines between consulting companies — the different types of firms, and even firms within the same category (staffing-oriented or project-oriented, for example) — are somewhat blurry. In this section, I don't want to steer you in one direction, but rather point out that a data warehousing consultant from one type of environment might be a good fit for your particular needs — but he or she might not. Similarly, a particular consulting

firm might best serve your needs, depending on how you want to proceed with your data warehousing project (for example, how many — if any — people internal to your organization will work on the project). Or perhaps that consulting organization, regardless of its data warehousing expertise, isn't a good fit for the particular implementation model you're pursuing.

Here's the key to avoiding mistakes: Before you talk with any consulting firms or individuals about your data warehousing needs, have a good idea first about what type of model you're most likely to pursue (internal management of the project, rather than management by the consulting organization). With this information, you can better determine not only the technical capabilities, but also the cultural fit, that best meets your needs, deliverable dates, and budget.

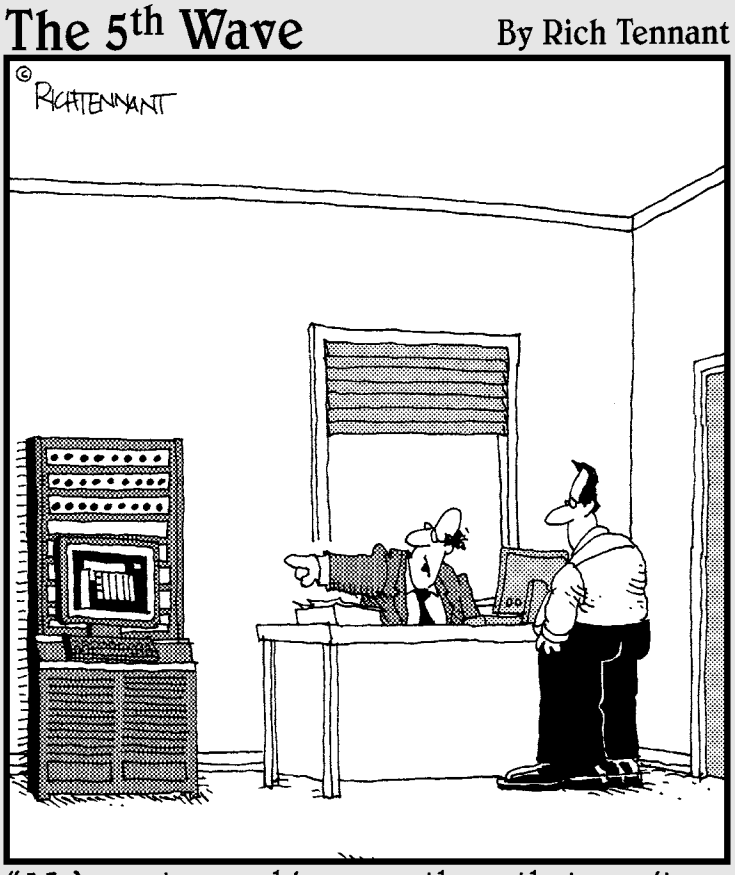
A Final Word about Data Warehousing Consultants

Although it's important for both individual consultants and consulting companies to check references and technical and business qualifications (such as product and industry experience), follow your instincts. Information is growing and data warehousing technology is changing so rapidly that you might do more harm than good if you insist on using a consultant (or a company) who has experience with specific products, who has previously implemented a data warehouse in your industry, or who otherwise fits a checklist of qualifications.

Certainly consider someone who lacks experience in a particular industry or in the use of a certain product, but who otherwise impresses you as an insightful, hard worker who can get the job done. For example, you might want to see the person in action for perhaps a week or two, as part of a team doing the data warehousing scope. Don't get caught up in the laundry-list mode of finding and retaining consulting assistance.

Part VI

Data Warehousing in the Not-Too-Distant Future



"We've got a machine over there that monitors our quality control. If it's not working, just give it a couple of kicks."

In this part . . .

You may think of data warehousing as old technology. However, data warehousing is changing — right before your eyes.

Five to ten years from now, you're likely to implement a data warehouse in a way that's much different from how you do it today. You'll have to make the leap from today's generation of data warehousing to a new generation of technologies and architectures.

Be prepared: Read this part of the book to see what's coming soon and what these new data warehousing models mean to you.

Oh, yeah — you might also want to visit some of the Internet sites mentioned in this part to get a very real sense of what's coming.

Chapter 25

Expanding Your Data Warehouse with Unstructured Data

In This Chapter

- Recognizing the limits of today's data warehousing
- Getting data through multimedia
- Looking at business intelligence and unstructured data
- Going from unstructured information to structured data

Today's data landscape now encompasses a dizzying array of new information channels, new sources of data, and new analysis and reporting imperatives. According to analyst groups, nearly 80 to 85 percent of today's data is unstructured, and new information channels such as Web, e-mail, voice over IP, instant messaging (IM), text messaging, and podcasts are rapidly creating huge stores of nontraditional data. Data from any of these sources will be requested from your users to be integrated into your data warehouse.

Traditional Data Warehousing Means Analyzing Traditional Data Types

Unless you've used an extraordinary, state-of-the-art data warehouse, your business intelligence functionality has probably been limited to these types of data:

- **Numbers:** Numeric data in the technical form of integers and decimal numbers
- **Text:** Character data, typically fixed-length alphanumeric information that's rarely more than about 255 characters per occurrence, although (very rarely) it might go up to 4,000 characters

- **Dates and times:** Either actual dates and times or, more likely, ranges of dates (such as a month and year for which product sales are grouped and stored)

That's about it.

To be fair, data warehousing in its original incarnation, as a storage place for information drawn from legacy applications to support reporting and analysis, hasn't needed anything other than these traditional data types. These traditional data types form the basis of structured data managed by databases, the most popular being relational databases or multidimensional databases.

You know what Bob Dylan said about the times and how they change.

It's a Multimedia World, After All. . . .

Fire up your Web browser. Spend a few hours poking around the Internet, checking out all kinds of cool sites. (If you're at work and your boss walks in, point to this chapter and say that you're doing data warehousing research — honest!) You can find images, video and audio clips, entry forms for filling out information to submit to a site's database, tabular results based on requests you might make — almost anything.

Today, an organization typically stores a large proportion of its data in documents created by using productivity tools such as Microsoft Excel and Word. Additionally, digitization advances in photography, document scanning, video production, and audio formats have further extended the range of unstructured data formats that you can use for business data.

The lines between structured data (traditional data types that computer applications have been using for years) and unstructured data (such as multimedia documents) have blurred. Not that long ago, if you wanted to create a multimedia environment that included both structured and unstructured data, you loosely followed these steps:

1. Build a relational database for your structured data.
2. Use a document-management system or an image-management system for your unstructured data.
3. To handle logical links across environments, set aside in each relational database row one or more columns that point to related documents or images, as appropriate.

These environments were relatively awkward and prone to problems. For example, software upgrades to one system had an adverse effect on the other (links that break, for example).

The emergence of a new generation of business applications that merges traditional relational data structures with unstructured digital content has already begun. This profusion of digital content means that organizations are now seeking to manage both relational (structured) data and unstructured data at the enterprise level.

For example, consider a medical records application. Fifteen years ago, the application would most likely have maintained a list of medical records that were stored as simple rows and columns. Today, and in the near future, a medical records application is more likely to manage a set of visit records that have reference images, x-rays, CAT scans, prescriptions, and other reference documents — and those records might also include higher-level capabilities such as spatial visualization, reporting, and analysis.

Many businesses will be (or currently are) eager to turn this unstructured data into useful information, but they'll find (or they found) that their current data warehousing and business intelligence technology can't deliver thorough analysis of this data. Traditional data warehousing and business intelligence technologies and infrastructure have inherent technological constraints that limit their ability to address this data.

How Does Business Intelligence Work with Unstructured Data?

Suppose that you're using an unstructured, multimedia-enabled data warehousing environment to do comparative analysis between services offered by your company (a bank) and your competitors' corresponding offerings.

You run some basic reports and a few queries (as I talk about in Chapter 9) to check out market share, portfolio performance, and other measures. Or, for more advanced analysis, you use a business analysis OLAP tool (refer to Chapter 10) to perform all kinds of drill-down analysis on the data in an attempt to fully understand the intricacies of your company's performance with respect to your competitors.

Sometimes, though, you can't find the answers in the numbers. Suppose that you notice a sudden increase in account closures at your bank during the past two months. What's going on?

You can understand the premise of business intelligence in the term itself: Get as much intelligence as possible — as fast as possible — from as many sources as possible, to help you understand what's going on and take informed action. Under this broad definition, intelligence can easily include the following types of information that you can't find in (or access through) a traditional data warehouse:

- A competitor's local newspaper advertisement offering no-fee checking for one year and an extra 1.5 percent earned on money-market deposits if a potential customer shows a bank statement indicating that he or she has closed an account at your bank
- An advertising banner on Google that features your competitor's same offer
- A link to each of your competitors' Web sites, where you can analyze the types of electronic banking services they offer
- A transcript of an interview with a regional economic expert stating that your bank is a prime takeover target and probably won't be in business under its current name at the same time next year

In this simple example, because the items occur locally or regionally, you might believe that you can access all this information from a multimedia-enabled data warehouse. ("A good banking analyst probably knows all this stuff anyway," right?)

Think about this example on a global scale, however. Are you wondering why your company's sales are slipping in Sweden? You might need to have these types of real-time, intelligence-gathering capabilities for a globally competitive situation.

I once created for a client in the chemical industry the architecture for a quasi-data warehouse environment (*quasi* because it had only a single source of data but a huge amount of historical information that I had to bring into the new system).

About 80 percent of the historical information was on paper, and the client was considering eventually entering that information into a document-management system. For budgetary reasons, I dealt only with the conversion of traditional historical data (character, numeric, and date information), and mapping and transforming the new incoming data. The documents would be handled later.

Imagine an environment in which you can treat all this data, which deals with the same subject matter, equally. If the data is on paper, you can scan it in as an image, index it by keyword, and make it accessible through the same environment as the traditional data. You tremendously increase the client's business intelligence by giving them access to this information.

An Alternative Path: From Unstructured Information to Structured Data

The example in the preceding section demonstrates an approach of putting structured data first, in which a business analyst uses data warehousing as a gateway into appropriate unstructured supporting information.

You can just as easily take the opposite path toward a unified approach to business intelligence. Suppose that you're browsing the Internet or the company intranet, and a product diagram, blueprint, or some other type of image or document catches your attention. Each piece of unstructured information can just as easily provide a path for you to access an OLAP-generated report posted on the company intranet (which can, in turn, have links that point you toward other structured or unstructured information) or can involve a mash-up with your information projected on a Google Map (as discussed in Chapter 26).

The structured and unstructured data barriers are breaking down quickly, just like the pathways across those softened barriers. Increasingly, businesses are amassing large volumes of non-relational, unstructured data in the form of digital images, documents, videos, and other multimedia formats — and these new data formats are quickly becoming a key component in formal and informal business processes that integrate with existing business applications, comply with regulatory requirements, or simply provide a richer user experience. Consider the following business scenarios:

- A pharmaceutical company needs to access lab documentation compiled over years and generations of clinical trials to gain FDA approval for a new medicine.
- An insurance company needs to store policy documents and retrieve them for claims processing.
- A call center company needs to store agent-recorded sessions as audio streams so that they can be retrieved remotely for quality assurance and contract compliance.
- An industry analyst firm needs to make a searchable library of podcasts available for download from its Web site.
- A legal practice needs to store electronic copies of documents as images and easily retrieve the documents relating to an individual client or case.
- An architect partnership needs to store and retrieve digital plans with the associated client data.
- A library needs to convert and archive large volumes of existing paper and analog content for indexing and use in a digital research tool (remember microfiche?).

The preceding list gives you a few examples of the ways in which businesses throughout the world can and do use unstructured digital data. You can analyze such information as easily as you can create digital content. Organizations are finding new, innovative ways to use this digital content to improve or extend their business capabilities, and many of those organizations need data warehousing and business intelligence solutions to leverage this information.

If you use traditionally collaborative work processes, performing tasks such as workflow or image management, you can easily augment those processes to point you toward data warehousing capabilities that provide you with additional value. In addition, the reports and query results you get and use as part of traditional analytical processing can serve as a pathway into a world of multimedia information that can supplement the data you typically handle.

Chapter 26

Agreeing to Disagree about Semantics

In This Chapter

- Defining semantics in computer science
- Anticipating the Semantic Web
- Using semantics in data warehousing
- Preparing for the semantic wave

In researching how best to present the topic of semantics, I searched many areas on the Web, including sites such as Wikipedia (`www.wikipedia.org`) and the World Wide Web Consortium (`www.w3.org`). The concept of the *Semantic Web* is one with deep technical roots, but in its simplest sense, it's making the Web as easy to navigate for applications as it is for you and me (humans). You know to go to Google and search for *the Finnish word for "monkey"* — a computer cannot accomplish the same task without direction from you or me.

Defining Semantics

Semantics is the study of meaning in communication, including the meaning (or an interpretation of the meaning) of a word, sign, or sentence. How many times in the middle of an argument have you heard the phrase, "Let's not argue about semantics"? Linguists and semanticists have been dealing with semantics for a long time. However, in the world of computer science, semantics are relatively new.

In May 2001, Tim Berners-Lee, James Hendler, and Ora Lassila authored an article in the magazine *Scientific American* titled "The Semantic Web." You can find the article here:

```
www.sciam.com/article.cfm?id=the-semantic-web
```

In the final line of this article, the authors state that the Semantic Web will open up the knowledge and workings of mankind to software agents — productivity applications that will perform analysis on our behalf.

With this article, The Semantic Web era began. The Semantic Web is about two specific technical standards:

- **Common integration formats:** The Semantic Web uses common formats to integrate and combine data drawn from diverse sources. The original Web mainly concentrated on the interchange of documents.
- **Language for relationship mapping:** The Semantic Web uses language to record how the data relates to real-world objects. These records allow a person, or a machine, to start off in one database, and then move through an unending set of databases connected by a common subject.

These two concepts can be applied to data warehousing and business intelligence efforts. In the reference architecture I present in Chapter 17, a user access layer maps to a target data layer; additionally, a target data layer maps through data movement rules to the source data layer. In this context, business intelligence and data warehousing architectures are no different than Web architectures — they all need common integration formats and a language for relationship mapping.

Emergence of the Semantic Web?

Between the Semantic Web article being published in 2001 (which I describe in the preceding section) and now, what technical innovation has emerged? The answer is Web 3.0, although I'm not too certain many of you have heard of this version of the Web. The Semantic Web is really no different than the current Web, aside from *agents,* or programs that do work on your behalf (and which can now surf the Web, too). Just what you wanted, your programs wasting time trying to find something out there!

Web 3.0 will most likely not be up and running until sometime after 2010. This third generation of the World Wide Web will be filled with Internet-based services that collectively comprise what you might call The Intelligent Web. Technologies and techniques, such as the Semantic Web, natural language search, data-mining, machine learning, recommendation agents, and artificial intelligence technologies will formulate the foundation of Web 3.0 and emphasize machine-facilitated understanding of information in order to provide a more productive and intuitive user experience.

You can already see small samples of what will come with Web 3.0 in the form of mash-ups. In Web development, a *mash-up* is an application that combines data from more than one source into a single integrated tool; for

example, Google Maps uses geospatial map data to add location information to real-estate data, thereby creating a new and distinct Web service that neither source originally provided. Many people are experimenting with mash-ups by using Amazon, eBay, Facebook, Flickr, Google, Microsoft, Yahoo!, and YouTube APIs. Examples of mash-ups include

- **The Chicago Police Department CLEARMAP Crime Summary:** This mash-up application summarizes crime data by using mapping software to help warn citizens of high crime areas within the city:

  ```
  http://gis.chicagopolice.org/CLEARMap_crime_sums/
          startPage.htm
  ```

- **Flickrvision:** This mash-up application combines the power of mapping software with a popular photo sharing site. The applications are separate and distinct, but by using APIs and semantics, they integrate to provide you with a very unique experience that neither of the application originators intended:

  ```
  http://flickrvision.com/maps/show_3d
  ```

This level of innovation demonstrates what's possible with technology, and both data warehousing and business intelligence platforms can really benefit from enhancement through semantics. The biggest challenge involves the need for openness among the community of vendors.

Think of a world in which you can use Cognos query and reporting tools against a Business Objects Universe running on a Composite Software federated data access layer, which accesses data from your data warehouse and various streams of information from the World Wide Web — presenting the final results on a Google Map. The world is headed in this widely distributed, yet easily integrated direction in the not too distant future!

Preparing for Semantic Data Warehousing

If you evaluate how the technologies associated with data warehousing work, a lot of metadata is managed in a proprietary manner within the tool. Some have attempted in the past to solve the integration problem by creating yet another technology, known as *technical metadata repositories,* to integrate all this metadata from the various tools. Such technologies generally have failed to provide overall business value because they're point-to-point solutions, and a data warehouse must have support and funds to stay alive.

You must begin separating the language for relationship mapping and integration formats from a proprietary base into one that's abstracted away from the tools. Several technologies are emerging to assist in this area:

- **Balanced Insight Consensus (`www.balancedinsight.com`):** Defines key business terminology and, through a set of relationships, the inherent business models that exist within data. From there the business terms and models are mapped to data structures, thus producing business intelligence and data warehousing solutions in specific technologies. With a press of a button, you can see the exact same model generated in Microsoft, Cognos, Business Objects, Pentaho, and other business intelligence platforms. Balanced Insight Consensus is the first open business modeling tool on the market that has significant promise to drive the semantic layer, regardless of which tool or database you use.
- **Composite Software (`www.compositesw.com`):** Provides a layer of software that you can use for data services such as virtualization, integration, and federation. Composite Software's platform has key technology in the areas of optimization, caching, and integration that enables you to take one query and execute it against data sources that are structured and/or unstructured; internal and/or external — proving that there can be very separated sources of information integrated that originally were not intended to be in an optimized fashion.
- **Expressor Software (`www.expressor-software.com`):** Redefines data integration by employing a fundamentally new design concept around semantics, allowing organizations to rationalize physical metadata constructs around common business terms and write target-specific data transformation and business rules that are highly reusable. Expressor Software tackles the complexity and cost of enterprise IT projects, and it delivers much higher productivity and data processing performance based on its semantic metadata foundation.

These products are just three of the growing list of companies you'll see emerging in the area of semantics.

All these tools don't try to reinvent technologies such as relational databases, business intelligence tools, or data movement platforms — however, they do all optimize the business-to-technology interfaces so that the existing technologies you are using for your data warehousing and business intelligence products can work more efficiently together to solve business problems.

Starting Out on Your Semantic Journey

Ask yourself and your data warehousing project teams this question: How will you prepare for these technologies? Focus on three areas that you're probably already working in — your business intelligence semantic layer, business rules management, and possibly federated query definition.

Business intelligence semantic layer management

If you dig deep enough into your process and tools (as discussed in Chapter 17), you can find someone in the business or delivery team who maps key business terminology to technology requirements or implementations. People who use Business Objects Designer or Cognos Framework Manager should map key business terminology (or business metadata) to technical database tables and columns. If your process is mature, you might even find that business analysts capture the business metadata in Microsoft Word templates, which your business intelligence tool administrators leverage. If this is the case, raise the bar on these efforts. Begin to define the *semantics* (another word for the business metadata) and initiate an alignment process across the silos of your business. By performing such a process, you can begin to determine key terminology conflicts across lines of business or functional areas of your business. The sales department might refer to the dollar amount on a contract as Revenue, but the finance department refers to it as Bookings. If finance and sales worked together to resolve this term, they might agree to use the term Booked Revenue, defined as *money committed by a customer legally for the delivery of products and/or services within a given specification laid out in a contract.* Additionally, the finance and sales team might conclude that the business needs to monitor a revenue life cycle that spans sales (Booked Revenue), field operations (Deferred Revenue), and accounting (Recognized Revenue). If you want to bring the semantic world to your data warehousing environment, you start by understanding key terms, as well as their associations and rules.

Business rules management

Additionally, investigate the process around business rules — both how you define those rules and how you generate them. The process of defining business rules is very similar to the business intelligence semantic definition process. Users define key concepts; for example, Customer means *a legal entity which has conducted business by purchasing goods or services from one of our companies in the current 12-month period.* Such a definition evolves into a database query or data movement routine that flags a legal entity as a customer or not. You can move your enterprise toward the world of semantics by creating an alignment process that gathers a cross section of the business community together to agree on such terminology and rules.

Chapter 27

Collaborative Business Intelligence

In This Chapter

- Looking into a collaborative business intelligence support model
- Translating current collaboration technologies for use with business intelligence solutions
- Using collaborative business intelligence in the future

According to Wikipedia, *collaboration* is a recursive process where two or more people or organizations work together toward an intersection of common goals — for example, an intellectual endeavor that's creative in nature — by sharing knowledge and building consensus. Collaboration doesn't require leadership and can sometimes bring better results through decentralization and egalitarianism. In particular, teams that work collaboratively can obtain greater resources, recognition, and reward when facing competition for finite resources.

Collaboration technology has an enormous role in driving user interactions in the areas of entertainment and problem-solving. Collaboration in the technology sector refers to a wide variety of tools that enable groups of people to work together through asynchronous and synchronous methods of communication. Examples of collaboration technology include synchronous Web conferencing, using tools such as Microsoft Live Meeting or Cisco Webex, and asynchronous collaboration by using software such as Microsoft SharePoint. Additionally, Instant Messaging platforms such as those from Yahoo!, AOL, ICQ, and Microsoft enable peer-to-peer and group forum collaboration.

If properly implemented, a collaborative work environment can assist companies in making work dramatically easier. Not only can a group cheaply communicate and test, but the wide reach of the Internet allows such groups to easily form in the first place, even among niche interests, creating virtual

communities online. The Internet has enabled low-cost and nearly instantaneous sharing of ideas, knowledge, and skills by leveraging collaborative technologies such as wikis and blogs.

Future Business Intelligence Support Model

Currently, business intelligence support is often relegated to is-it-working support — if the system is down, the support personnel get it running again. This support model (along with vendor pricing schemes) has prevented broad adoption of business intelligence. This support model is like a sit-down restaurant that has a limited menu.

For broad adoption of business intelligence to occur, you need a support model that enables collaboration. The necessary technology does exist — but no one is really combining data warehousing, business intelligence, and collaboration software. Some early adopters are venturing down this path, and some vendors are beginning to introduce limited collaboration functionality into their products.

In the future, you'll see concepts that are currently emerging from leading Internet sites such as Amazon, Facebook, YouTube, Google, and Wikipedia become commonplace for business intelligence solutions. It has to happen, it's such a natural support model. And collaboration is all about sharing knowledge and building consensus: Business intelligence is about gaining knowledge — the missing link is sharing and building consensus.

As we move to the next phase of business intelligence, in which everyone can have access to the data that's important to them, the need for collaboration is growing. Users will need to support users, both internal and external to an enterprise.

Internal support could come from regional managers trying to isolate shared costs and quickly collaborating with a functional analyst in finance to determine the correct slice or drill to perform on their data (see Chapter 10 for business analysis functionality). External support could come from a customer whom you've enabled to surf through his or her orders, who needs some quick assistance on how to combine your enterprise's data with other data that he or she has — say, from his or her bank — and merge that data within a GoogleDoc. Such support isn't way out in the future.

Knowledge retention

Every day, no one captures the millions of items of intellectual property (conversations, e-mails, and telephone calls) in organizations all over the world. Using collaborative software to institutionalize *quantitative* (structured) and *qualitative* (unstructured) information — which would otherwise be lost — can enable the sharing of information, thoughts, insights, and best practices.

How we have progressed this far without leveraging collaboration within business intelligence solutions is beyond me. The use of collaboration software can enable your users to capture intelligence from outside the data warehouse and operational systems and use this captured intelligence as part of the business intelligence solution.

Knowledge discovery

Collaborative business intelligence is an environment within which users can easily collaborate and communicate with each other, sharing ideas, information, and decision making within their community. Through such collaboration, users can begin reusing existing ideas and/or capabilities, removing the need to reinvent the wheel — which happens quite often when information technology departments isolate a user's experience to reports. Using collaboration software side by side with business intelligence software can enhance information presented to users, enabling everyone to share insights they gain with other users in their community.

Knowledge proliferation

Collaboration allows participants to share knowledge, observations, and analytics in a community of interest with the goal of producing an action response to a situation. This sharing can become infectious. Remember your first experience with eBay or YouTube? You probably arrived at these very popular sites after one of your friends or colleagues directed you there. This guidance is viral in nature. If users find software sites and solutions easy to use, discuss, and gain support for, they pass the word on to others.

Such activities resulting from user-to-user collaboration will drive the sharing of knowledge across organizational boundaries — including observations, insights, and data from customers; economic and industry information; and psychographic information (interests, attitudes, and opinions). In other words, your data warehouse will grow and become more virtual in nature because the conversations that occur will require much information from both inside and outside the boundaries of a single organization's data warehouse.

Leveraging Examples from Highly Successful Collaboration Solutions

Collaborative business intelligence might look like today's Internet-based collaborative experience in several ways. The following sections discuss features that are most likely to become commonplace within a collaborative business intelligence solution.

Rate a report

If you look up a book on Amazon, you find a rating and customer feedback. This rating and associated feedback might be the information that drives you to purchase that book — or go for a different book. In the world of business intelligence, users could leverage report rating features to locate the most valuable information for their needs and provide descriptive feedback, which allows users to understand how the report serves the user population best.

Report relationships

When you access a particular book's page on Amazon, you can scroll down a little on the page to see a section that highlights books that other shoppers purchased at the same time they bought the one you're evaluating. This feature in your data warehouse could assist users in understanding interrelated reports and views provided within the business intelligence environment. For example, you might be looking at the top ten sales report and see that users who viewed this report also reviewed the active promotions report, the commission schedule report, and the products by territory analytical view. The relationship of data across reports (which you might not see within individual reports) can become obvious if you see the patterns of users' report access.

Find a report

You're probably comfortable with searching technologies, such as the ones that Google offers, and taxonomies, such as those you can find through Yahoo!. Search-engine technology has evolved over the years to incorporate relationships, ranking, popularity, and user weighting, along with various linguistic capabilities needed to search.

Most business-intelligence vendors have begun offering search capabilities, often from vendors such as Google, within their platforms. In the future, you'll use the search engine inside your firewall, out on the Internet, in your data warehouse, and in public data sources — in essence, securely searching all over the place for the information you need.

Find the meaning

Wikis, such as Wikipedia, enable users to capture content (edit and publish) and shape it to become key intellectual property that you can think of as corporate knowledge. How many times have users come to you with that question mark look on their face, trying to decipher the meaning of a report or the data presented in a column or graph? Leveraging and integrating this technology enables a growing knowledge base to form around each view or perspective presented out of the data warehouse and business intelligence solution, which can establish the corporate knowledge on the user of your data assets.

Shared interests — shared information

Social networking brings people together through shared interests — for example, your sales teams with marketing promotions personnel or inventory control personnel. YouTube, Facebook, MySpace, and Flickr are all examples of social network environments that consumers are now swarming. Are bored people at home the only people going to these sites? Definitely not! Large advertisers, such as Procter and Gamble, have spotted this trend and are spending large sums of their advertising budgets on these sites because they see a huge transfer of consumers from the old technology (television) to the new technology (social networking on the Internet).

Imagine being able to post a view of information that you don't quite understand so that your peer group can try to figure out a reasonable explanation for the changes that have you confused — especially before the boss comes asking for an answer.

Visualization

Gaming software of today provides stimulating entertainment and intriguing lessons in interface design. These interfaces provide a strong attraction for users, which is very different from the anxiety and resistance that users display when they utilize the tools you provide them for accessing their data warehouse. Advances in visual technology have been in place for some time,

and with Nintendo's Wii, even the hardware devices are simplified! And now that you can access gaming online, you can play some death-and-destruction game with your long-lost college roommate who now lives in Thailand.

Well, why not get together with your sales counterpart in the Far East to determine how to better sell your products to a specific ethnic community? You can make this connection by collaborating on utilizing all the technologies described within this chapter and both of you surfing the data with more natural techniques. If you don't think this kind of collaboration will happen, just watch some movies. The creative minds are working on these kinds of technological collaborations, and they'll expand on the concepts that I talk about in Chapter 26, giving the world mash-ups of data access technologies with visualization and manipulation technologies soon enough.

The Vision of Collaborative Business Intelligence

Collaborative business intelligence is real in many consumer situations. eBay, Amazon, and Google all leverage key data to present information to you, the consumer. Companies such as eBay are even contemplating products in the area of data warehousing and business intelligence to assist vendors in better moving products through their platform.

But these technologies need a more open vendor community than currently exists. At this point, Microsoft has all the tools — you just have to find a really good consultant to put it all together because Microsoft is so subdivided that one person employed by Microsoft could never help (you'd need three to five people). Business Objects, now an SAP company, has introduced collaboration for their data integrator product, enabling project teams to communicate. Because Business Objects is now part of SAP, various collaboration features of NetWeaver will eventually sneak into the solution — of course, you'll have to sole source it initially. Cognos, now part of IBM, has the same story as Business Objects. And Oracle — yes, they have all the tools.

I'm guessing that each vendor will sneak in and try to provide you with the "complete solution," but the open vendors of the Internet will win out. Current trends, with companies such as Google, which has GoogleDocs working with Panorama Software, against Microsoft's Analysis Services or SAP's BW InfoCubes, suggest that multi-vendor solutions will begin to emerge. This will be far more appealing to users than a single proprietary vendor solution that requires heavy migration of solutions that are already in place.

Part VII
The Part of Tens

In this part . . .

If you want easy-to-access, succinct advice about many different data warehousing topics, this part of the book is for you.

Chapter 28

Ten Questions to Consider When You're Selecting User Tools

In This Chapter

- Asking questions about using tools
- Getting the scoop on how a tool might work in your organization
- Finding out about the quality of online help
- Figuring out how a tool deals with the computer-crash test

Few things are more frustrating than successfully building a data warehouse and then having it rendered unusable by less-than-satisfactory user tools. This chapter presents some questions to consider when you're evaluating tools that you might want to purchase.

Do I Want a Smorgasbord or a Sit-Down Restaurant?

Bet you weren't expecting a question like this to pop up, were you? I use this analogy often because it helps frame the discussion of what you're looking at in a tool. When business intelligence was initially created, there was a broad vision that users could serve themselves. The user would go to one environment, insulated by all the underpinnings of the data, and merely ask a question and get an answer, ask the next question, and so on. But IT has often restricted the tool so that end user can't access it, so they can't self-serve — like in a sit-down restaurant. Let me explain:

- The diner must wait to be seated at the restaurant, which is equivalent to the user trying to find someone in IT to listen to him or her.

- When a waiter is assigned to a section of the restaurant that the diner is seated in, that waiter must serve the other customers while the diner sits and waits, much like someone in IT supporting multiple departments in your enterprise (and we all know some people seem to find all kinds of ways to pull attention away from your needs).
- When the waiter does finally come and take your order, he delivers this order to a cook — who, by the way, is supporting all the waiters in the restaurant. This relationship is equivalent to IT's way of prioritizing who's request gets a project assigned — yet another period of waiting.
- And, of course, the cook can't serve you until she has plates from the dishwasher, who can't provide the plates until the busboy finishes clearing tables — much like the user waiting for the current projects that are over budget and off schedule to complete.

You have to determine whether you want the users to take on the burden of writing a majority of their own reports — or whether you want to make IT a report-manufacturing shop. If you want the sit-down restaurant, you need to find the best query tool, best reporting tool, best OLAP tool, and so on. If you want the smorgasbord, you want to get the most integrated product that allows the user to move easily from tool module to tool module.

In most data warehousing environments, a significant portion of the user community has a substantial number of report and query screens and templates, and the development team builds these capabilities for the users. At the same time, to avoid creating a backlog of requests that the support staff can't easily handle, users must be able to create and use their own queries and reports.

While you're evaluating a tool, find two users who have no experience in user reporting and querying products, and a third user who has used another product but not the one you're considering. A few days before you run your usability test, give the tutorial documentation (the written version) to only one of the users who has no experience. Then, have all three people try to solve two or three business problems of varying difficulty by creating a query or report with the tool. See how they do! If the users can perform query, reporting, and analysis tasks, you might avoid requiring the technical support staff to build all reports for the users.

Can a User Stop a Runaway Query or Report?

Almost every tool user occasionally submits a query (or performs some other type of operation, such as running a report) that keeps going and going and going. . . .

A user tool must give users a way to stop this type of query or report gracefully, without doing any of the following:

- Locking up the user's desktop PC and forcing him or her to turn it off or reboot
- Interfering with other users' work (by requiring that you halt the database server and restart it, for example)
- Otherwise causing a disruption in business as usual

Check out each product, and make sure that any user can stop a runaway query or report from his or her browser or desktop PC.

How Does Performance Differ with Varying Amounts of Data?

You may have determined during the project scope that your data warehouse will start with 500 gigabytes of data, for example, and grow to 1 terabyte during the next two years. It pays to know, however, how each tool will perform with not only the initial 500 gigabytes and the eventual target, but also with 2 or even 3 terabytes, just in case.

In case of what? Here are just a few possibilities:

- New data sources that no one could foresee during the project scope.
- A decision to add an increased level of detail to the data.
- A decision not to delete old data, but rather to keep it in the data warehouse.
- An unforeseen merger occurs, and you must incorporate the new entities' information into your data warehouse.

If your data warehouse will never approach terabyte-size (a trillion bytes of data), don't worry about how a tool performs with that much data — it's irrelevant. What's more significant is whether the tool can perform as well (or nearly as well) with 100 gigabytes of data as it does with 500 gigabytes. This information will allow you to sleep at night knowing that the tool will scale to perform beyond the defined requirements.

Performance isn't a tool-only situation; it also depends on the DBMS you use, how you design your database, and many other factors. Ask your questions in an environment as close as possible to what will be available during production.

Can Users Access Different Databases?

I'm talking about different databases of information from the same tool, not necessarily different DBMS products. For example, a user may access the regular data mart stored on a local Windows NT server for most queries and reports and, by using the same tool, have access to this information:

- Another department's data mart, for occasional queries
- The organization's main data warehouse
- An external data provider over the Internet

Users shouldn't have to switch tools to perform similar functions (basic querying and reporting, or OLAP, for example) against different data sources.

And, furthermore, the extra credit question would be: Can the user access the different databases within one query or report? Although you may not require this ability currently, it may become something you want to incorporate in the future. For example, you may want to tie your historic data warehouse data with the transactional data in real time or with external data in near real time.

Can Data Definitions Be Easily Changed?

Although the process of getting the first set of data definitions up and running is fairly easy in most user tools, you need to ask yourself these questions about when you need to make modifications:

- How easily can you update your entire user community's data definitions, and how long does it take? (Do you have 100 users? 1,000? 5,000?)
- How do you modify queries and reports that use data that no longer exists or data with a modified structure (its data type and size, for example)?
- What happens to scripts and programs that are part of the tool?

How Does the Tool Deploy?

Does the tool require desktop deployment, or can you deploy it within your organization's standard browser configuration? If you're looking at a tool that's not available for deployment in a browser, you may want to reevaluate your criteria. Server-side deployment lowers a lot of costs that exist when you have to deploy tools on the PC.

However, most business intelligence suites haven't advanced in one area — their modeling and administration tools. There still is a heavy leaning towards desktop tools in this area. Considering that most organizations have internal standards for their client PCs, the process of figuring out how well a tool fits into your standard configuration should be straightforward. Assuming that the product runs on your desktop operating system (always a showstopper if it doesn't), you should consider these issues:

- How well the connectivity and interoperability software you need works, such as drivers for database connectivity and repository connectivity.
- Whether you should load the software in any special order. Yes, vendors still provide software that conflicts if you don't load it in the proper sequence.
- When you have choices about where certain components of a tool can reside (on each client, for example, or located once on each server), determine the recommended configuration and whether any problems exist in your environment.

As far as the end-user interface, these components should run in the browser without the need for any special downloads — do yourself a favor and get an Internet-architected tool. You have plenty to choose from!

How Does Performance Change If You Have a Large Number of Users?

You have to know how performance changes when the number of users increases. You should see little or no performance effect if a tool's environment is designed correctly and efficiently — make sure the tool you're considering falls in this category.

What Online Help and Assistance Is Available, and How Good Is It?

Any tool worth its salt should include pretty extensive online help. Features to look for include wizards, tutorials, context-sensitive help, and templates for queries and reports.

Does the Tool Support Interfaces to Other Products?

The desktop analysis tool of choice is the handy-dandy, trusted spreadsheet program (and this preference probably isn't going to change for many years). Although some OLAP products feature direct interfaces from their database into a spreadsheet (typically Microsoft Excel), even if you use a product's reporting and querying capabilities, users should always be able to bring data back from the business intelligence tool and pop it into Excel for more analysis, manipulation, or whatever they want to do.

You generally should be able to include reports in word-processing documents (such as a Microsoft Word file), graphics and presentation programs (for example, Microsoft PowerPoint), a personal database (such as Microsoft Access), and (of course) a spreadsheet (such as Excel).

What Happens When You Pull the Plug?

Go ahead — try it. In the middle of a query or report, turn off your PC and see what happens. You want to ensure not only that users can restart a desktop PC without any leftover configuration problems (for example, temporary files and workspace errors that prevent the user from doing additional work until the errors are fixed), but also that the interruption doesn't affect your warehouse's database (and any intermediate servers). Or better yet, find out whether the product is smart enough to welcome the user back through his or her browser, reconnecting that user to his or her query.

Chapter 29

Ten Secrets to Managing Your Project Successfully

In This Chapter

- Dealing effectively with everyone involved in your project
- Keeping an eye on the project plan
- Connecting your project team members and supporters
- Taking a break from project work every now and then

Being a successful data warehouse project manager means that you have to do more — much more — than simply create project plans and ask team members to turn in weekly status reports. This chapter presents the secrets to success.

Tell It Like It Is

It doesn't matter whether you're working with users, executive sponsors, consultants, vendors, team members, or anyone else. The most important thing you can do to set the groundwork for successfully managing a data warehousing project is to speak your mind in a completely honest manner.

You don't have to be abrupt or rude, or have the attitude "It's my way or no way." Follow these guidelines to keep communication open and to solve problems sooner rather than later:

- When problems occur, don't bury them or pretend that they don't exist. Other people know about the problems, so deal with them aggressively.
- Don't be afraid to tell an executive sponsor that those out-of-the-blue budget cuts or the absence of those three team members who have been reassigned "for just a little while" will adversely affect your project.
- Don't hesitate to tell a vendor when a product isn't performing as promised and demand that they do something about it.

The key: Communicate quickly, openly, and with integrity. You won't be sorry.

Put the Right People in the Right Roles

The right person in the right role is an important key to project success.

You have to recognize that the best database designer might be somewhat challenged when it comes to working with front-end OLAP or data mining tools. The person who can do whiz-bang tasks with a particular OLAP tool might be a lousy facilitator and should, during the scope phase of a project (refer to Chapter 13), either sit silently in the back of the room or just not even be there.

Be a Tough but Fair Negotiator

Budget cuts, pressures to compress the development schedule, vendor support, working with the corporate infrastructure group to line up installation and rollout support — the project manager usually has responsibility for all these issues, and many more. The manager must ensure that these tasks, which all involve negotiation, take place. After you speak your mind, as discussed in the section "Tell It Like It Is," earlier in this chapter, you establish the groundwork for tough, fair negotiations that are grounded in reality, not in emotion or speculation. Don't be afraid to negotiate from this basis: "If X happens, Y will be the result."

Deal Carefully with Product Vendors

Be careful when you gather information from vendors and other sources (as described in Chapter 23) and when you question a vendor about a product (as explained in Chapter 34). Recognize that vendors want to sell you products, not solve your business problems. Although it's great when they can do both, you don't share the same priorities as the vendors.

It's *your* project. Don't be pushed into making product decisions that aren't in your best interest.

Watch the Project Plan

Although I firmly believe that being a good project manager means more than just tracking how the project schedule is going, you can't ignore the project plan.

If you're not interested in gathering team members' regular submissions to help keep your project plan up-to-date, add to your team a project-control staff member who has the specific task of managing the project plan. Work with a local college or university to get a work-study or cooperative education (co-op) student; it's a cost-effective way to handle this important task.

Don't Micromanage

Everyone has a particular management style. Some people focus on delegating tasks, and others are more hands-on. If you're the type who likes to handle most things yourself, the advice in this section is for you.

Don't *micromanage,* or insist on knowing every little detail about every task that everyone is doing. (That panicky, out-of-control feeling will go away.) Even on smaller projects, trust your developers and analysts to know their jobs. Check in on them to see how they're doing, and make sure that they're progressing on schedule. Let them do their jobs, though — especially on larger projects. You have enough to worry about as a project manager; don't take on additional worries that team members usually can resolve for you.

Use a Project Wiki

Start off every project with a comprehensive project wiki to which you provide access to every member of your extended team (not only the developers and key users, but also the executive sponsors).

For those of you not familiar with wikis, a *wiki* is a page or collection of Web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified markup language. Wikis are often used to create collaborative Web sites and to power community Web sites. Wikis are used in business to provide intranets and knowledge-management systems.

The wiki should have these items:

- Copies of key memos
- A description of earlier work done on the data warehousing project (prototypes, for example)
- A summary of discussions you've already had with vendors
- High-level statements of requirements
- Whatever else you need to ensure that the wiki represents the complete picture of what you're trying to accomplish

Even if your organization has a fantastic *intranet* (an Internet environment inside the company) or a widely used Lotus Notes (or other type of groupware) environment, build out a wiki — which you can find out how to do in *Wikis For Dummies,* by Dan Woods and Peter Thoeny (Wiley). Because you can't assume that everything you need is available in electronic form (it usually isn't), be sure to have access to a scanner to pull in those items that aren't currently electronic. Additionally, you might want to provide secure access for consultants and others who might be working remotely. Be sure to have your wiki available and operational at your kickoff meeting so that no one gets left out.

Don't Overlook the Effect of Organizational Culture

Suppose that you work for a consulting firm whose employees typically work 50 or 60 hours each week on projects. You're assigned to be the project manager for a client's data warehousing effort and will manage a team composed of four members from your company and four from the client's. You develop an aggressive (but realistic) project plan, based on the client's budget and time constraints, that will likely involve some late-night and weekend work. (That concept is nothing new to your company's employees.)

Suppose, however, that your client's employees won't even consider working more than 40 hours a week. Now what?

Although you can factor this attitude into the workload of the client's team members as best you can, you have to consider other factors, too:

- If your client insists on having responsibility for database builds and rebuilds, who will handle those tasks if they have to be done over the weekend or late at night during a critical stage of development?
- When the unavoidable project hiccups occur (and you had better believe that they will), will your client's team members be as likely as your firm's team members to give a little more effort?

Don't overlook the impact of organizational culture on the project you're managing, especially if you're an outsider (a consultant, for example, or someone from another company who's working on a joint cross-company data warehousing project).

Don't Forget about Deployment and Operations

Design and development are difficult enough for a data warehouse (or any environment, for that matter). Don't overlook how the results of your work will function in the real world, with real users. Make sure that your project plan allows time for lining up support after the data warehouse goes live.

Take a Breather Occasionally

Insist that everyone leave early on Friday after a particularly hard week. Don't sneer and scowl when team members tell you that they want to go to the company picnic when you're a day or two behind schedule with three weeks to go.

It's only work. By taking an occasional breather, you (and your team members) become reinvigorated, and productivity increases. It's well worth your while to take off a weekend here and there, or to spend slightly fewer hours working overtime.

Chapter 30

Ten Sources of Up-to-Date Information about Data Warehousing

In This Chapter

- Checking out specialized data warehousing Web sites
- Turning to industry analysts' Web sites for information
- Getting product details by going to vendors' Web sites

You probably won't be surprised to hear that the term *up-to-date* means "go look on the Internet." It's worth your time to check out the Web sites in this chapter.

The Data Warehousing Institute

```
www.tdwi.org
```

At the Data Warehousing Institute's comprehensive site, you can find these items:

- Education information
- Upcoming events
- White papers, including links to white papers at other sites
- Case studies
- A directory of vendors
- Best practices (so that you can find out how others have succeeded in various data warehousing categories)

The Data Warehousing Information Center

```
www.dwinfocenter.org
```

At the Data Warehousing Information Center, you can find out about data warehousing technology, find links to a variety of other sites, and provide the site's operator (Larry Greenfield, of LGI Systems, Incorporated) a place for "rants and raves" (his words) about data warehousing.

The site has tables of various tool categories and products in those categories, along with links to the vendors' respective sites. It also has links to white papers, articles, periodicals, conferences, and many other services.

The OLAP Report

```
www.olapreport.com
```

The OLAP Report is published by Business Application Research Center, a subscription service. This Web site has information available for nonsubscribers, including

- Documentation on the origins of OLAP
- Product reviews
- Benchmarks
- Market analysis, including market share information
- Case studies

Intelligent Enterprise

```
www.intelligententerprise.com
```

Intelligent Enterprise has articles, interviews, white papers, and resources about the various technologies leveraged for data warehousing. It's focused on how technologies can work together to form the strategic applications vital to businesses. This Web site offers both the context and the technical detail needed to go from concept to purchase to deployment.

b-eye Business Intelligence Network

```
www.b-eye-network.com
```

The Business Intelligence Network focuses on business intelligence, performance management, data warehousing, data integration, and data quality, serving these communities with industry coverage and resources. The site has a plethora of industry expert blogs, product spotlights, white papers, and podcasts to help you through your data warehousing journey.

Wikipedia

```
www.wikipedia.org
```

This Web site might seem misplaced, but Wikipedia is a growing source for key information — it's nearly the new Encyclopedia Britannica! Though, at times, the articles and information might be self-serving for vendors or consultants, in general, this site is a wealth of information.

DMReview.com

```
www.dmreview.com
```

DMReview.com delivers market insight through interviews with, as well as articles and columns written by industry consultants, hands-on practitioners, and technology solution leaders. Editorial focus is on business intelligence, performance management, analytics, integration, and enterprise data warehousing, as well as emerging areas that include business process management and technology architectures.

BusinessIntelligence.com

```
www.businessintelligence.com
```

BusinessIntelligence.com provides articles, white papers, research, and news for the business intelligence marketplace.

Industry Analysts' Web Sites

You can get information about data warehousing, and the IT industry in general, from sites run by some of the leading industry analysts:

- Gartner: `www.gartner.com`
- International Data Corporation (IDC): `www.idcresearch.com`
- Forrester Research: `www.forrester.com`
- Ventana Research: `www.ventanaresearch.com`

Product Vendors' Web Sites

Hundreds, if not thousands, of sites exist for product vendors. Although the vendors are dedicated to selling their respective products, a lot of their sites have white papers, late-breaking news about their products and features, and other worthwhile material.

In addition to using a search engine to find these sites, you can find links to vendor sites from many of the Web sites listed in this chapter.

Chapter 31

Ten Mandatory Skills for a Data Warehousing Consultant

In This Chapter

- Knowing a little about a lot of data warehousing topics
- Knowing a lot about a few data warehousing topics
- Working well and effectively with the people in your organization
- Having access to up-to-date information
- Using a hype-free vocabulary

A good data warehousing consultant has certain abilities in dealing with people and a knowledge of various aspects of data warehousing. This chapter lets you in on a few required skills that all data warehousing consultants should possess.

Broad Vision

Even a data warehousing consultant who's an expert in a particular area (star schema design in a relational database in support of OLAP functionality, for example) should have a broad vision in at least these areas:

- Overall end-to-end data warehousing architecture, from tools to middleware to data quality to orchestration software
- An understanding of client/server, Web-based, and server-side computing architectures
- A firm understanding of database optimization concepts for tuning data access queries
- Skills in digging through data sources to see what's really there

Because the components of a data warehousing environment are interrelated, a consultant must be able to not only provide technical expertise in one or two areas of a project (as discussed in the following section), but also see the big picture.

Deep Technical Expertise in One or Two Areas

If you're going to pay the big bucks for a consultant who claims to be a data warehousing expert, that person must be a true expert. More specifically, a consultant should be able to claim, proudly and *accurately,* to be the best in one or two areas (database design and front-end tools, for example).

Communications Skills

"Um, well, you know, I think that, uh, that requirement the guy in back mentioned, like, last week, right? You know, like, what were we talking about?"

Although a consultant's written and verbal grammar doesn't have to be perfect (an occasional dangling modifier is okay), even the most technically astute consultant must be able to convey ideas and understand what others are communicating. It's critical!

The Ability to Analyze Data Sources

A consultant should never design the necessary transformations for a data warehouse (refer to Chapter 7) solely by using listings of data structures and definitions provided by the keepers of an application or the IT department. A consultant must be able to dig into source databases, even if this source analysis is only a secondary role for the consultant. For example, even a consultant who isn't the primary source-data analyst might have to figure out why the business intelligence tool returns strange results.

The Ability to Distinguish between Requirements and Wishes

A consultant's ability to distinguish between user requirements and wishes is important primarily in working on the scope (the first phase) of a data warehousing project. A disparate group of users probably bombard you with cries of, "I need this!" and "I want that!" During crunch time, good facilitation and negotiation skills are essential when functionality has to be cut from the list — or, at least, deferred until the next version of the data warehouse.

Conflict-Resolution Skills

No matter what role a consultant plays, from project manager to data analyst to quality assurance (QA) specialist, that person is an outsider to the members of an organization — and someone from the client company is almost always resentful of the outsider's "intrusion." A consultant on a data warehousing project (or any other project, for that matter) must identify these situations early and do the best possible job of diffusing any conflict that threatens to destroy a project.

An Early-Warning System

A consultant should act as an early-warning system to identify and report problems to you, the client, so that you can deal with them. The consultant shouldn't be a snitch, but he or she should be more than just a nose-to-the-grindstone technician. Because this person is an outsider and not involved (you hope!) in your internal organizational politics, he or she should have some freedom to notify you of problems.

A consultant whose organization has problems (another consultant who isn't performing up to par, for example) might not feel free to let you know about those problems. That's where your company's people should also act as an early-warning sign for the consulting organization's staff members. (That's why conflict-resolution skills, discussed in the preceding section, are so important!)

General Systems and Application Development Knowledge

While data warehousing and mainstream computing continue to converge, an increasing number of warehouses will be built using distributed objects; the use of messaging and other data-movement technologies for near-real-time business intelligence will increase; and a lot of other capabilities that weren't part of a typical first-generation data warehousing environment will develop. A consultant who has strong skills should have at least a working knowledge of these areas, in addition to basic programming skills and other abilities.

The Know-How to Find Up-to-Date Information

From data warehousing product bug fixes to information about the latest architectural trends, a good consultant knows how to find up-to-date information quickly — in time to be put to good use on your data warehousing project.

A Hype-Free Vocabulary

Because it's almost impossible to avoid catchy buzzwords (can you say "data mart"?) in the data warehousing world, don't hold it against a consultant (or anyone else) who uses these phrases. But I'm generally wary of consultants who sound like they went to a trade show and met up with the data warehousing pod people: "Don't be afraid. Join us for some neural network data mining that uses subject-oriented data to give you predictive pattern recognition built by using data vaulting techniques in SSAS (Microsoft SQL Server Analysis Services) and SSIS (Microsoft SQL Server Integration Services) — we are your friends!"

Chapter 32

Ten Signs of a Data Warehousing Project in Trouble

In This Chapter

- Failing to agree on your project during various phases
- Making bad project decisions
- Losing team members
- Dealing with poor communication
- Discovering problems with your data warehousing products

You can most easily tell that your data warehousing project is in trouble when you don't have anything to show for your efforts when you thought you would. Try to get some indication that trouble's brewing, however, *before* you reach that point. This chapter presents ten early warning signs.

The Project's Scope Phase Ends with No General Consensus

The allotted time for the scope phase of your data warehousing project ends (usually two or three weeks — a little longer for large projects), and the members of your constituency are unhappy. They're still grumbling and disagreeing about the project's direction and its potential business value (or lack thereof), the relative priorities of capabilities and how they map to various project phases, and other points of contention.

You're in trouble.

The Mission Statement Gets Questioned after the Scope Phase Ends

You're three weeks into the design phase, following a four-week scope. You're in an all-morning status meeting with the IT and business organization executive sponsors, as well as four key managers from the business groups who plan to use the data warehouse the most.

Just before a coffee break, one of the managers says, "You know that mission statement we talked about on the second day of the project? I want to talk about that some more because I have some problems with it."

You're in trouble.

Tools Are Selected without Adequate Research

A project decision-maker looks around the room in disgust, sighs deeply, and says, "Look, we just don't have time to check out these tools because the schedule is too tight. That vendor who was in here yesterday — what was that company's name again? You know, the ones with the product that — I can't remember all the details. Anyway, I liked their demo. We'll buy that tool."

You're *probably* in trouble.

People Get Pulled from Your Team for "Just a Few Days"

"I'm going to borrow Mary, John, and Sue Ellen for a couple of days because we have something important over in the shoelace plastic-tips division that must get done as soon as possible. Anyway, I think that it'll be for only a few days. Try to stay on schedule."

You're in trouble.

You're Overruled When You Attempt to Handle Scope Creep

You're in the last week of the design phase, and a business unit manager sits down across from you in the cafeteria. Between mouthfuls of Chef's Daily Surprise, he tells you about "these one or two things I just thought of that would make this data warehouse thing work much better." You politely explain the concept of scope creep in the context of "Wait until the %$#^@ next phase of the data warehouse!" (except that you're more diplomatic about it). Then, two days later, the manager's boss (who also happens to be your boss) sends an e-mail message directing you to "add those one or two things to the features list, but don't let them affect the schedule."

You're in trouble.

Your Executive Sponsor Leaves the Company

You've done a fantastic job of selling the business value of your data warehousing project to executive management, and everything is rolling along nicely. Suddenly, two days after a stunning announcement of disappointing quarterly sales and earnings, the executive sponsor from the business side of the organization resigns. Now, your project doesn't have an executive sponsor.

You *might* be in trouble: Work fast and don't look back.

You Overhear, "This Will Never Work, but I'm Not Saying Anything"

Everyone in the company is supportive of your data warehousing project. You're pushing the cutting edge of technology, and everyone on your team is enthused. Their weekly status reports even reflect the progress they're making. The project's chief architect assures you that the more you get into the project, the more everyone is convinced that you've made sound technical decisions.

Then, in the cafeteria, you overhear two of the more senior developers discussing the project. One says, "There's no way that this thing can work. Performance is terrible, and half the time the same query against the same data returns different results! But I'm not going to be the one to bring it up!"

You're in trouble.

You Find a Major "Uh-Oh" in One of the Products You're Using

Despite your best efforts at product evaluation, something has slipped through the cracks, and a major feature simply doesn't work. Although you can use one work-around, that work-around really negatively affects performance. The vendor's representatives slyly say, "Well, we had heard that it might be a problem. Our development organization is looking into it and will probably make a patch available in the next month or so."

You're in trouble.

The IT Organization Responsible for Supporting the Project Pulls Its Support

Your development group is in charge of most of the data warehousing development, including the business intelligence tools and the database definitions. The IT organization, though, is responsible for creating the databases and performing the loading routines, performing the backup and restore procedures, and taking care of many of the project's other infrastructure elements.

Because of higher priorities, the IT organization pulls the people responsible for supporting your project, and their manager promises to "look into another answer." She says, "Maybe we'll hire a couple of contractors, but I won't be able to look into that until next week."

You're in trouble.

Resignations Begin

Resignations are a sure sign that major problems lie ahead. Even people who are unhappy with a company often give in to loyalty or a sense of duty, and they stick around until the completion of a project. (Or maybe they just want the résumé fodder.)

When a number of people resign in the middle of a project, however, you're in trouble.

Chapter 33

Ten Signs of a Successful Data Warehousing Project

In This Chapter

- Getting praise from executives and coworkers
- Seeing people actually use and discuss the data warehouse
- Gaining your CEO's confidence
- Climbing the corporate ladder

As mentioned elsewhere in this book, just because everyone gathers in the company cafeteria for cake and plasters the walls with congratulatory banners doesn't mean that your data warehousing project was a success. This chapter gives you some ways to tell that you were really successful.

The Executive Sponsor Says, "This Thing Works — It Really Works!"

Suppose that a senior executive at your company makes it a point to find you so that he or she can tell you that you did a great job *and* that you're a nice person *and* that the data warehouse you built and delivered really works — and everyone is using it. The executive even points out that the warehouse is delivering information that is being factored into boardroom-level decisions.

You succeeded!

You Receive a Flood of Suggested Enhancements and Additional Capabilities

Sometimes, after the celebratory party in the cafeteria, a data warehouse slowly fades away like an old soldier. (Quiz: Who used a similar phrase, to what audience, and in what year?)

Users and their managers might bang on your office door (or, more likely, invade your cubicle) to show you memo pads that contain sketches of additional reports and queries that they want, asking questions such as, "How hard would it be to add this feature?"

You succeeded!

User Group Meetings Are Almost Full

Your company should always organize a data warehouse user group in which you can discuss and handle issues such as training, enhancement requests, and tips and tricks for how to use query tools in a coordinated manner.

User group meetings that often get canceled because no one has issues to discuss gives you a good indication that few people are using the data warehouse. In contrast, regular user group meetings that are packed give you a strong indication that . . . you succeeded!

The User Base Keeps Growing and Growing and Growing

You start off with an initial user community of 50 business-area analysts. Two months later, you add another 50. During the next six months, an additional 150 users, including several in executive management, join the "family."

You succeeded!

The Executive Sponsor Cheerfully Volunteers Your Company as a Reference Site

Your executive sponsor is so enamored with the data warehouse that management wants your company to serve as a reference for product vendors and for the outside consulting company that worked with you to build it.

You succeeded!

The Company CEO Asks, "How Can I Get One of Those Things?"

The big cheese wants his or her own executive dashboard system (and all his or her direct reports) to have access and monitor it daily. To that end, you are summoned to the CEO's office for a weekly hour-long private session to answer specific questions on further uses for data warehousing within the company.

You've succeeded!

The Response to Your Next Funding Request Is, "Whatever You Need — It's Yours."

Corporations are notorious for this type of funding policy: "Don't tell us what you did yesterday. Tell us what you'll do tomorrow *if* we deem you worthy to send funding your way, and don't you dare fail to deliver because you and your résumé will be on the street faster than. . . ."

But if your data warehouse is popular and held in high esteem, your organization might give you a blank check for your next project.

You've succeeded!

You Get Promoted — and So Do Some of Your Team Members

Nothing says lovin' in the corporate world like a promotion. Suppose that the job you did on your data warehouse leads directly to a promotion for you *and* for other members of your team.

You've succeeded!

And although it might not feel as good, if your team is suddenly raided and all your direct reports get promotions, you've also succeeded!

You Achieve Celebrity Status in the Company

Company employees stop you in the hallway or invite you to lunch to ask your opinion about technology, development methods, and all sorts of other subjects. Although you're not quite at the level of rock star or professional athlete, diners in the cafeteria point toward your table and whisper, "That's the one!"

You've succeeded!

You Get Your Picture on the Cover of the Rolling Stone

Okay, your data warehousing project won't get your picture on *Rolling Stone* magazine — but if it did, you'd know that you truly succeeded!

Chapter 34

Ten Subject Areas to Cover with Product Vendors

In This Chapter

- Finding out who works for a vendor (and whether they stay)
- Asking about the vendor's customers
- Figuring out the vendor's role in the marketplace
- Looking at the vendor's future plans
- Determining the vendor's integrity

This chapter presents ten subject areas that you should discuss with any data warehousing product vendor, no matter which product category (business intelligence tool, middleware tool, or RDBMS, for example) you're considering buying. I've also listed the specific question you might want to ask. All these questions are somewhat odd because they have little to do with product features and how they work; if you do a good job of hands-on evaluation of the products, you can figure out how the features work by yourself. The questions in this chapter are more along the lines of "I want to know your company's character." In my opinion, these questions are *extremely* important because it will provide an understanding of the vendor's openness with its customers as well as its position in the market at large.

Product's Chief Architect

Who's your product's chief architect, and what background does this person have?

Most companies have a single individual who's primarily responsible for crafting and setting a product's technical direction. This person isn't often the same as the "company visionary," who might be the CEO who has seen a market need and is now trying to fill it.

You're asking about the person whose imprint is all over the product. Find out as much as you can about this person's background and experience.

If the current chief architect isn't the original chief architect, ask what happened to that person. It's not a good sign when a person who probably had a decent compensation package and attractive stock options is no longer with a company, especially if it's a relatively new company.

Development Team

How big is your development team, and how does it compare with last year's?

Here are some guidelines to consider when you ask this question:

- A relatively small development staff (for example, five or six people for a commercial product) might indicate skimpy quality assurance (QA) and product testing. It might also indicate that the vendor (or their financial backers) is taking a wait-and-see attitude toward the product before pumping in more funds. You might also see other signs that the company is making a less-than-full-scale commitment to the product.
- A development team that's the same size as or — worse — smaller than it was a year earlier most likely indicates internal concern about the product's market viability.
- A development team that's significantly larger than it was a year earlier indicates the vendor's enthusiasm for the product's chances and viability in the marketplace.

Customer Feedback

How have you addressed the top three customer complaints about the preceding version of this product?

Vendors have customer-support service organizations and call centers, so they should know the answer to this question. (If they don't, consider it a bad sign.)

You should determine whether customer complaints occur in clusters according to specific features, performance, connectivity, or whatever — and then determine the company's reaction and responsiveness.

Employee Retention

How high was employee turnover in the past year?

Pay attention to this indicator of a company's internal mood, especially if it's a relatively young company that hasn't gone public yet. Few people leave a growing company in which a big payoff waits around the corner.

Marketplace

Which company do you see as your chief competitor?

A vendor usually targets one company as their chief competitor and strives to overtake it. Or, if a vendor is the industry leader, they might just try to stay ahead of the latest up-and-comer.

You can find out a great deal about the product direction and use from the way the vendor answers this question.

If a vendor responds, "We really don't have any competitors," they're either blissfully ignorant or arrogant. Either way, watch out!

Product Uniqueness

What are the three most significant innovations in your product?

In what's often described as a *commodity product marketplace,* vendors' products are pretty much the same in terms of features and capabilities. Vendors usually base competition on price and other basic attributes, or they can try to position their product as truly innovative and worthy of market leadership — as well as a hefty price tag and significant support costs.

If the vendor claims their product is innovative, ask this question to see what makes the vendor tick.

Clients

How many clients have bought this product, and how many can serve as references?

What kind of market share does this product have?

Of the companies who bought it, how many are happy enough to tell others about their wonderful decision?

Watch out for vendors who claim that their customers are happy but don't want the vendor to identify them because they're doing top-secret, strategic work. How could the world knowing that a company purchased and is using a product be damaging to that company if it doesn't have to reveal application and business use? If a product vendor avoids providing references, start asking questions!

The Future

What significant advances do you expect in this market segment this year, and what are you doing in each area?

Here's what you want to find out (but without asking the question this way): "Are you guys just standing still, milking money off this product, or are you continuing to improve it?"

Internet and Internet Integration Approach

What's your strategy for Internet technologies and how are you opening your product up to enable integration with other Internet technologies?

Almost everyone has a Web-enablement story. Ask the vendor for their version. With the Internet moving into semantic connectivity, often called *mash-ups,* how is the vendor assuring that their technology can be leveraged in ways they never expected?

Integrity

Do you guarantee that your product will work as advertised?

Ah, a truly inspirational question. Gauging a vendor's response to this question tells you a great deal about the character of both the company and its sales representatives. It also generally indicates whether you're in for a bumpy ride if you do business with these folks.

Index

• C •

• D •

• E •

• F •

• G •

• H •

• I •

• J •

• K •

• L •

• M •

• N •

• O •

• P •

• Q •

• R •

• S •

• T •

• U •

• V •

• W •

• X •

There's more to data warehousing than you think, so start right here!

You don't need a forklift to work with a data warehouse, but you do need a hefty load of know-how to make wise decisions when setting one up. Data is probably your company's most important asset, so your data warehouse should serve your needs. Here's how to understand, develop, implement, and use data warehouses, plus a sneak peek into their future.

- ***Know your stuff* — understand what a data warehouse is, what should be housed there, and what data assets are**
- ***Get a handle on technology* — learn about column-wise databases, hardware assisted databases, middleware, and master data management**
- ***The intelligent view* — see how business intelligence and data warehousing work together**
- ***Ask the right questions* — explore data mining and learn to find what you need**
- ***Do the groundwork* — choose your project team and apply best development practices to your data warehousing projects**
- ***Keep the user in mind* — involve your users in defining business needs through testing, and learn how to get valuable feedback**
- ***Fix or replace?* — learn how to review and upgrade existing data storage to make it serve your needs**
- ***Buyer beware* — be prepared when dealing with data warehousing product vendors**

Open the book and find:

- **What to expect from your data warehouse**
- **The difference between data warehouses and data marts**
- **All about specialty database technologies**
- **What to look for in a consultant**
- **How your data warehouse feeds dashboards and scorecards**
- **Secrets for managing a successful data warehouse project**
- **How to effectively capture business needs and requirements**
- **Ten signs your project is in trouble**

Go to dummies.com® for more!

Thomas C. Hammergren has been involved with business intelligence and data warehousing since the 1980s. He has helped such companies as Procter & Gamble, Nike, FirstEnergy, Duke Energy, AT&T, and Equifax build business intelligence and performance management strategies, competencies, and solutions. **Alan R. Simon** is a data warehousing expert and author of many books on data warehousing.

For Dummies®
A Branded Imprint of
WILEY

$34.99 US / $41.99 CN / £27.99 UK

ISBN 978-0-470-40747-9