PART
I

Requirements, Realities, and Architecture

CHAPTER
1

Surrounding the Requirements

Ideally, you must start the design of your ETL system with one of the toughest challenges: surrounding the requirements. By this we mean gathering in one place all the known requirements, realities, and constraints affecting the ETL system. We'll refer to this list as the *requirements*, for brevity.

The requirements are mostly things you must live with and adapt your system to. Within the framework of your requirements, you will have many places where you can make your own decisions, exercise your judgment, and leverage your creativity, but the requirements are just what they are named. They are required. The first section of this chapter is intended to remind you of the relevant categories of requirements and give you a sense of how important the requirements will be as you develop your ETL system.

Following the requirements, we identify a number of architectural decisions you need to make at the beginning of your ETL project. These decisions are major commitments because they drive everything you do as you move forward with your implementation. The architecture affects your hardware, software, coding practices, personnel, and operations.

The last section describes the mission of the data warehouse. We also carefully define the main architectural components of the data warehouse, including the back room, the staging area, the operational data store (ODS), and the presentation area. We give a careful and precise definition of data marts and the enterprise data warehouse (EDW). Please read this chapter very carefully. The definitions and boundaries we describe here drive the whole logic of this book. If you understand our assumptions, you will see why our approach is more disciplined and more structured than any other data warehouse design methodology. We conclude the chapter with a succinct statement of the mission of the ETL team.

PROCESS CHECK

Planning & Design: *Requirements/Realities* → Architecture → Implementation → Test/Release

Data Flow: Haven't started tracing the data flow yet.

Requirements

In this book's introduction, we list the major categories of requirements we think important. Although every one of the requirements can be a show-stopper, business needs have to be more fundamental and important.

Business Needs

Business needs are the information requirements of the end users of the data warehouse. We use the term *business needs* somewhat narrowly here to mean the information content that end users need to make informed business decisions. Other requirements listed in a moment broaden the definition of business needs, but this requirement is meant to identify the extended set of information sources that the ETL team must introduce into the data warehouse.

Taking, for the moment, the view that business needs directly drive the choice of data sources, it is obvious that understanding and constantly examining business needs is a core activity of the ETL team.

In the *Data Warehouse Lifecycle Toolkit*, we describe the process for interviewing end users and gathering business requirements. The result of this process is a set of expectations that users have about what data will do for them. In many cases, the original interviews with end users and the original investigations of possible sources do not fully reveal the complexities and limitations of data. The ETL team often makes significant discoveries that affect whether the end user's business needs can be addressed as originally hoped for. And, of course, the ETL team often discovers additional capabilities in the data sources that expand end users' decision-making capabilities. The lesson here is that even during the most technical back-room development steps of building the ETL system, a dialog amongst the ETL team, the data warehouse architects, and the end users should be maintained. In a larger sense, business needs and the content of data sources are both moving targets that constantly need to be re-examined and discussed.

Compliance Requirements

In recent years, especially with the passage of the Sarbanes-Oxley Act of 2002, organizations have been forced to seriously tighten up what they

report and provide proof that the reported numbers are accurate, complete, and have not been tampered with. Of course, data warehouses in regulated businesses like telecommunications have complied with regulatory reporting requirements for many years. But certainly the whole tenor of financial reporting has become much more serious for everyone.

Several of the financial-reporting issues will be outside the scope of the data warehouse, but many others will land squarely on the data warehouse. Typical due diligence requirements for the data warehouse include:

- Archived copies of data sources and subsequent stagings of data
- Proof of the complete transaction flow that changed any data
- Fully documented algorithms for allocations and adjustments
- Proof of security of the data copies over time, both on-line and off-line

Data Profiling

As Jack Olson explains so clearly in his book *Data Quality: The Accuracy Dimension*, data profiling is a necessary precursor to designing any kind of system to use that data. As he puts it: "[Data profiling] employs analytic methods for looking at data for the purpose of developing a thorough understanding of the content, structure, and quality of the data. A good data profiling [system] can process very large amounts of data, and with the skills of the analyst, uncover all sorts of issues that need to be addressed."

This perspective is especially relevant to the ETL team who may be handed a data source whose content has not really been vetted. For example, Jack points out that a data source that perfectly suits the needs of the production system, such as an order-taking system, may be a disaster for the data warehouse, because the ancillary fields the data warehouse hoped to use were not central to the success of the order-taking process and were revealed to be unreliable and too incomplete for data warehouse analysis.

Data profiling is a systematic examination of the quality, scope, and context of a data source to allow an ETL system to be built. At one extreme, a very clean data source that has been well maintained before it arrives at the data warehouse requires minimal transformation and human intervention to load directly into final dimension tables and fact tables. But a dirty data source may require:

- Elimination of some input fields completely
- Flagging of missing data and generation of special surrogate keys
- Best-guess automatic replacement of corrupted values
- Human intervention at the record level
- Development of a full-blown normalized representation of the data

And at the furthest extreme, if data profiling reveals that the source data is deeply flawed and cannot support the business' objectives, the data-warehouse effort should be cancelled! The profiling step not only gives the ETL team guidance as to how much data cleaning machinery to invoke but protects the ETL team from missing major milestones in the project because of the unexpected diversion to build a system to deal with dirty data. Do the data profiling up front! Use the data-profiling results to prepare the business sponsors for the realistic development schedules, the limitations in the source data, and the need to invest in better data-capture practices in the source systems. We dig into specific data- profiling and data-quality algorithms in Chapter 4.

Security Requirements

The general level of security awareness has improved significantly in the last few years across all IT areas, but security remains an afterthought and an unwelcome additional burden to most data warehouse teams. The basic rhythms of the data warehouse are at odds with the security mentality. The data warehouse seeks to publish data widely to decision makers, whereas the security interests assume that data should be restricted to those with a need to know.

Throughout the Toolkit series of books we have recommended a role-based approach to security where the ability to access the results from a data warehouse is controlled at the final applications delivery point. This means that security for end users is not controlled with grants and revokes to individual users at the physical table level but is controlled through roles defined and enforced on an LDAP-based network resource called a directory server. It is then incumbent on the end users' applications to sort out what the authenticated role of a requesting end user is and whether that role permits the end user to view the particular screen being requested. This view of security is spelled out in detail in *Data Warehouse Lifecycle Toolkit*.

The good news about the role-based enforcement of security is that the ETL team should not be directly concerned with designing or managing end user security. However, the ETL team needs to work in a special environment, since they have full read/write access to the physical tables of the data warehouse. The ETL team's workstations should be on a separate subnet behind a packet-filtering gateway. If the ETL team's workstations are on the regular company intranet, any malicious individual on that intranet can quietly install a packet sniffer that will reveal the administrative passwords to all the databases. A large percentage, if not the majority, of malicious attacks on IT infrastructure comes from individuals who have legitimate physical access to company facilities.

Additionally, security must be extended to physical backups. If a tape or disk pack can easily be removed from the backup vault, security has been compromised as effectively as if the on-line passwords were compromised.

Data Integration

Data integration is a huge topic for IT because ultimately IT aims to make all systems work together seamlessly. The *360 degree view of the business* is the business name for data integration. In many cases, serious data integration must take place among the primary transaction systems of the organization before any of that data arrives at the data warehouse. But rarely is that data integration complete, unless the organization has settled on a single enterprise resource planning (ERP) system, and even then it is likely that other important transaction-processing systems exist outside the main ERP system.

In this book, data integration takes the form of conforming dimensions and conforming facts. Conforming dimensions means establishing common dimensional attributes (often textual labels and standard units of measurement) across separate databases so that *drill across* reports can be generated using these attributes. This process is described in detail in Chapters 5 and 6.

Conforming facts means agreeing on common business metrics such as key performance indicators (KPIs) across separate databases so that these numbers can be compared mathematically by calculating differences and ratios.

In the ETL system, data integration is a separate step identified in our data flow thread as the *conform* step. Physically, this step involves enforcing common names of conformed dimension attributes and facts, as well as enforcing common domain contents and common units of measurement.

Data Latency

The data latency requirement describes how quickly the data must be delivered to end users. Data latency obviously has a huge effect on the architecture and the system implementation. Up to a point, most of the traditional batch-oriented data flows described in this book can be sped up by more clever processing algorithms, parallel processing, and more potent hardware. But at some point, if the data latency requirement is sufficiently urgent, the architecture of the ETL system must convert from batch oriented to streaming oriented. This switch is not a gradual or evolutionary change; it is a major paradigm shift in which almost every step of the data-delivery pipeline must be reimplemented. We describe such streaming-oriented real time systems in Chapter 11.

Archiving and Lineage

We hint at these requirements in the preceding compliance and security sections. But even without the legal requirements for saving data, every data warehouse needs various copies of old data, either for comparisons with new data to generate change capture records or for reprocessing.

In this book, we recommend staging the data at each point where a major transformation has occurred. In our basic data flow thread, these staging points occur after all four steps: extract, clean, conform, and deliver. So, when does staging (writing data to disk) turn into archiving (keeping data indefinitely on permanent media)?

Our simple answer is conservative. All staged data should be archived unless a conscious decision is made that specific data sets will never be recovered. It is almost always less of a headache to read data back in from permanent media than it is to reprocess data through the ETL system at a later time. And, of course, it may be impossible to reprocess data according to the old processing algorithms if enough time has passed.

And, while you are at it, each staged/archived data set should have accompanying metadata describing the origins and processing steps that produced the data. Again, the tracking of this lineage is explicitly required by certain compliance requirements but should be part of every archiving situation.

End User Delivery Interfaces

The final step for the ETL system is the handoff to end user applications. We take a strong and disciplined position on this handoff. We believe the ETL team, working closely with the modeling team, must take responsibility for the content and the structure of data, making the end user applications simple and fast. This attitude is much more than a vague motherhood statement. We believe it is irresponsible to hand data off to the end user application in such a way as to increase the complexity of the application, slow down the final query or report creation, or make data seem unnecessarily complex to end users. The most elementary and serious error is to hand across a full-blown normalized physical model and to walk away from the job. This is why Chapters 5 and 6 go to such length to build dimensional physical structures that comprise the actual final handoff.

In general, the ETL team and the data modelers need to work closely with the end user application developers to determine the exact requirements for the final data handoff. Each end user tool has certain sensitivities that should be avoided, and certain features that can be exploited, if the physical data is in the right format. The same considerations apply to data prepared for OLAP cubes, which we describe in Chapter 6.

Available Skills

Some of the big design decisions when building an ETL system must be made on the basis of who builds and manages the system. You shouldn't build a system that depends on critical C++ processing modules if those programming skills are not in house, and you cannot reasonably acquire and keep those skills. You may be much more confident in building your ETL system around a major vendor's ETL tool if you already have those skills in house and you know how to manage such a project.

In the next section, we look in depth at the big decision of whether to hand code your ETL system or use a vendor's package. Our point here is that technical issues and license costs aside, you should not go off in a direction that your employees and managers find unfamiliar without seriously considering the implications of doing so.

Legacy Licenses

Finally, in many cases, major design decisions will be made for you implicitly by senior management's insistence that you use existing legacy licenses. In many cases, this requirement is one you can live with and for which the advantages in your environment are pretty clear to everyone. But in a few cases, the use of a legacy system for your ETL development is a mistake. This is a difficult position to be in, and if you feel strongly enough about it, you may need to bet your job. If you must approach senior management and challenge the use of an existing legacy system, be well prepared in making your case, and be man enough (or woman enough) to accept the final decision or possibly seek employment elsewhere.

Architecture

The choice of architecture is a fundamental and early decision in the design of the ETL system. The choice of architecture affects everything, and a change in architecture almost always means implementing the entire system over again from the very start. The key to applying an architectural decision effectively is to apply it consistently. You should read each of the following subsections with the aim of first making a specific architectural choice and then applying it everywhere in your ETL system. Again, while each one of the categories in this section can be a showstopper, the most important early architectural choice is whether to build the ETL system around a vendor's ETL tool or to hand code the system yourself. Almost every detail of the design of your ETL system will depend on this choice.

PROCESS CHECK

Planning & Design: Requirements/Realities → *Architecture* → Implementation → Test/Release

Data Flow: Haven't started tracing the data flow yet.

ETL Tool versus Hand Coding (Buy a Tool Suite or Roll Your Own?)

The answer is, "It depends." In an excellent Intelligent Enterprise magazine article (May 31, 2003, edited by Ralph Kimball), Gary Nissen sums up the tradeoffs. We have augmented and extended some of Gary's points.

Tool-Based ETL Advantages

- A quote from an ETL tool vendor: "The goal of a valuable tool is not to make trivial problems mundane, but to make impossible problems possible."
- Simpler, faster, cheaper development. The tool cost will make up for itself in projects large enough or sophisticated enough.
- Technical people with broad business skills who are otherwise not professional programmers can use ETL tools effectively.
- Many ETL tools have integrated metadata repositories that can synchronize metadata from source systems, target databases, and other BI tools.
- Most ETL tools automatically generate metadata at every step of the process and enforce a consistent metadata-driven methodology that all developers must follow.
- Most ETL tools have a comprehensive built-in scheduler aiding in documentation, ease of creation, and management change. The ETL tool should handle all of the complex dependency and error handling that might be required if things go wrong.
- The metadata repository of most ETL tools can automatically produce data lineage (looking backward) and data dependency analysis (looking forward).
- ETL tools have connectors prebuilt for most source and target systems. At a more technical level, ETL tools should be able to handle all sorts of complex data type conversions.
- ETL tools typically offer in-line encryption and compression capabilities.
- Most ETL tools deliver good performance even for very large data sets. Consider a tool if your ETL data volume is very large or if it will be in a couple of years.

- An ETL tool can often manage complex load-balancing scenarios across servers, avoiding server deadlock.
- Most ETL tools will perform an automatic change-impact analysis for downstream processes and applications that are affected by a proposed schema change.
- An ETL-tool approach can be augmented with selected processing modules hand coded in an underlying programming language. For example, a custom CRC (cyclic redundancy checksum) algorithm could be introduced into an ETL vendor's data flow if the vendor-supplied module did not have the right statistical performance. Or a custom seasonalization algorithm could be programmed as part of a data-quality step to determine if an observed value is reasonable.

Hand-Coded ETL Advantages

- Automated unit testing tools are available in a hand-coded system but not with a tool-based approach. For example, the JUnit library (www.junit.org) is a highly regarded and well-supported tool for unit testing Java programs. There are similar packages for other languages. You can also use a scripting language, such as Tcl or Python, to set up test data, run an ETL process, and verify the results. Automating the testing process through one of these methods will significantly improve the productivity of your QA staff and the quality of your deliverables.
- Object-oriented programming techniques help you make all your transformations consistent for error reporting, validation, and metadata updates.
- You can more directly manage metadata in hand-coded systems, although at the same time you must create all your own metadata interfaces.
- A brief requirements analysis of an ETL system quickly points you toward file-based processing, not database-stored procedures. File-based processes are more direct. They're simply coded, easily tested, and well understood.
- Existing legacy routines should probably be left as-is.
- In-house programmers may be available.
- A tool-based approach will limit you to the tool vendor's abilities and their unique scripting language. But you can develop a hand-coded system in a common and well-known language. (In fairness, all the ETL tools allow *escapes* to standard programming languages in isolated modules.)

- Hand-coded ETL provides unlimited flexibility, if that is indeed what you need. You can literally do anything you want. In many instances, a unique approach or a different language can provide a big advantage.

We would add one more advantage to the ETL Tool suite list: It is likely that the ETL tool suite will be more self-documenting and maintainable over a period of years, especially if you have a typical IT staff *churn*. The counter argument to this is that if your ETL development staff has a strong software-development tradition and good management, documentation and maintenance will not be as big a problem.

Using Proven Technology

When it comes to building a data warehouse, many initial costs are involved. You have to buy dedicated servers: at least one database server, a business intelligence server, and typically a dedicated ETL server. You need database licenses, and you have to pay for the ability of your users to access your business intelligence tool. You have to pay consultants and various other costs of starting up a new project. All of these costs are mandatory if you want to build a data warehouse. However, one cost is often not recognized as mandatory and is often avoided in an effort to reduce costs of the project—the cost of acquiring a dedicated ETL tool. It is possible to implement a data warehouse without a dedicated tool, and this book does not assume you will or won't buy one. However, it is advised that you do realize in the long run that purchasing an ETL tool actually reduces the cost of building and maintaining your data warehouse. Some additional benefits of using proven ETL technology are as follows:

- **Define once, apply many.** Share and reuse business rules and structured routines, keeping your data consistent throughout the data warehouse.
- **Impact analysis.** Determine which tables, columns, and processes are affected by proposed changes.
- **Metadata repository.** Easily create, maintain, and publish data lineage; inherit business definitions from a data-modeling tool, and present capture metadata in your BI tool.
- **Incremental aggregation.** Dynamically update summary tables by applying only new and changed data without the need to rebuild aggregates with each load process.
- **Managed batch loading.** Reduce shell scripts and enable conditional loading, load statistics, automated e-mail notification, and so on.

- **Simpler connectivity** to a wide variety of complex sources such as SAP and mainframes.
- **Parallel pipe-lined multithreaded operation**.
- **Vendor experience,** including success with dimensional models and a proven track record of supporting data warehouses.

More important than taking advantage of advanced functionality is that investing in a proven ETL tool can help you avoid reinventing the wheel. These tools are designed for one purpose: to do exactly what you are trying to do—load a data warehouse. Most have evolved into stable, robust ETL engines that have embedded capabilities to extract data from various heterogeneous sources, handle complex data transformations, and load a dimensional data warehouse.

Don't add new and untested products to your ETL configuration. The dashboard-of-the-month approach, which has a certain charm in the end user environment, is too reckless in the back room. Be conservative and wait for ETL technologies to mature. Work with vendors who have significant track record and who are likely to support your products five years down the road.

Batch versus Streaming Data Flow

The standard architecture for an ETL system is based on periodic batch extracts from the source data, which then flows through the system, resulting in a batch update of the final end user tables. This book is mostly organized around this architecture. But as we describe in Chapter 11, when the real-time nature of the data-warehouse load becomes sufficiently urgent, the batch approach breaks down. The alternative is a streaming data flow in which the data at a record level continuously flows from the source system to users' databases and screens.

Changing from a batch to a streaming data flow changes everything. Although we must still support the fundamental data flow steps of extract, clean, conform, and deliver, each of these steps must be modified for record-at-a-time processing. And especially with the fastest streaming flows, many of the usual assumptions about the arrival of data and even referential integrity have to be revisited. For instance, the basic numeric measures of a sales transaction with a new customer can arrive before the description of the customer arrives. Even after the customer is identified, an enhanced/cleaned/deduplicated version of the customer record may be introduced hours or even days after the original event. All of this requires logic and database updating that is probably avoided with batch-oriented data flow.

At the beginning of this section, we advise applying each architectural decision uniformly across the entire data warehouse. Obviously, in the case of choosing a batch or streaming approach, the choice should be made on an application-by-application basis. In Chapter 11, we discuss the points of commonality between the two approaches and show where the results of the batch approach can be used in the streaming context.

Horizontal versus Vertical Task Dependency

A horizontally organized task flow allows each final database load to run to completion independently. Thus, if you have both orders and shipments, these two database loads run independently, and either or both can be released on time or be late. This usually means that the steps of extract, clean, conform, and deliver are not synchronized between these two job flows.

A vertically oriented task flow synchronizes two or more separate job flows so that, above all, the final database loads occur simultaneously. Usually, the earlier steps are synchronized as well, especially if conformed dimensions like customer or vendor are used by more than one system. Either all the job streams reach the conform step and the delivery step or none of them do.

Scheduler Automation

A related architectural decision is how deeply to control your overall ETL system with automated scheduler technology. At one extreme, all jobs are kicked off by a human typing at a command line or starting an icon. At the other extreme, a master scheduler tool manages all the jobs, understands whether jobs have run successfully, waits for various system statuses to be satisfied, and handles communication with human supervisors such as emergency alerts and job flow status reporting.

Exception Handling

Exception handling should not be a random series of little ad-hoc alerts and comments placed in files but rather should be a system-wide, uniform mechanism for reporting all instances of exceptions thrown by ETL processes into a single database, with the name of the process, the time of the exception, its initially diagnosed severity, the action subsequently taken, and the ultimate resolution status of the exception. Thus, every job needs to be architected to write these exception-reporting records into the database.

Quality Handling

Similarly, you should decide on a common response to quality issues that arise while processing the data. In addition to triggering an exception-reporting record, all quality problems need to generate an *audit record* attached to the final dimension or fact data. Corrupted or suspected data needs to be handled with a small number of uniform responses, such as filling in missing text data with a question mark or supplying least biased estimators of numeric values that exist but were corrupted before delivery to the data warehouse. These topics are further developed in Chapter 4.

Recovery and Restart

From the start, you need to build your ETL system around the ability to recover from abnormal ending of a job and restart. ETL jobs need to be re-entrant, otherwise impervious to incorrect multiple updating. For instance, a job that subtracts a particular brand sales result from an overall product category should not be allowed to run twice. This kind of thinking needs to underlie every ETL job because sooner or later these jobs will either terminate abnormally or be mistakenly run more than once. Somewhere, somehow, you must keep this from happening.

Metadata

Metadata from DBMS system tables and from schema design tools is easy to capture but probably composes 25 percent of the metadata you need to understand and control your system. Another 25 percent of the metadata is generated by the cleaning step. But the biggest metadata challenge for the ETL team is where and how to store process-flow information. An important but unglamorous advantage of ETL tool suites is that they maintain this process-flow metadata automatically. If you are hand coding your ETL system, you need to implement your own central repository of process flow metadata. See Chapter 9.

Security

Earlier in this chapter, we describe our recommended architecture for role-based security for end users. Security in the ETL environment is less granular than in the end user environment; nevertheless, a systematic approach to security demands that physical and administrative safeguards surround every on-line table and every backup tape in the ETL environment. The most sensitive and important data sets need to be instrumented with operating system printed reports listing every access and every command performed by all administrators against these data sets. The print log should

be produced on a dedicated impact printer locked in a room that cannot be opened by any of the normal IT staff. Archived data sets should be stored with checksums to demonstrate that they have not been altered in any way.

The Back Room – Preparing the Data

PROCESS CHECK

Planning & Design: Requirements → *Architecture* → Implementation → Release
Data Flow: Extract → Clean → Conform → Deliver.

The back room and the front room of the data warehouse are physically, logically, and administratively separate. In other words, in most cases the back room and front room are on different machines, depend on different data structures, and are managed by different IT personnel.

Figure 1.1 shows the two distinct components of a typical data warehouse.

Preparing the data, often called *data management*, involves acquiring data and transforming it into information, ultimately delivering that information to the query-friendly front room. *No query services are provided in the back room*. Read that sentence again! Our approach to data warehousing assumes that data access is prohibited in the back room, and therefore the front room is dedicated to just this one purpose.

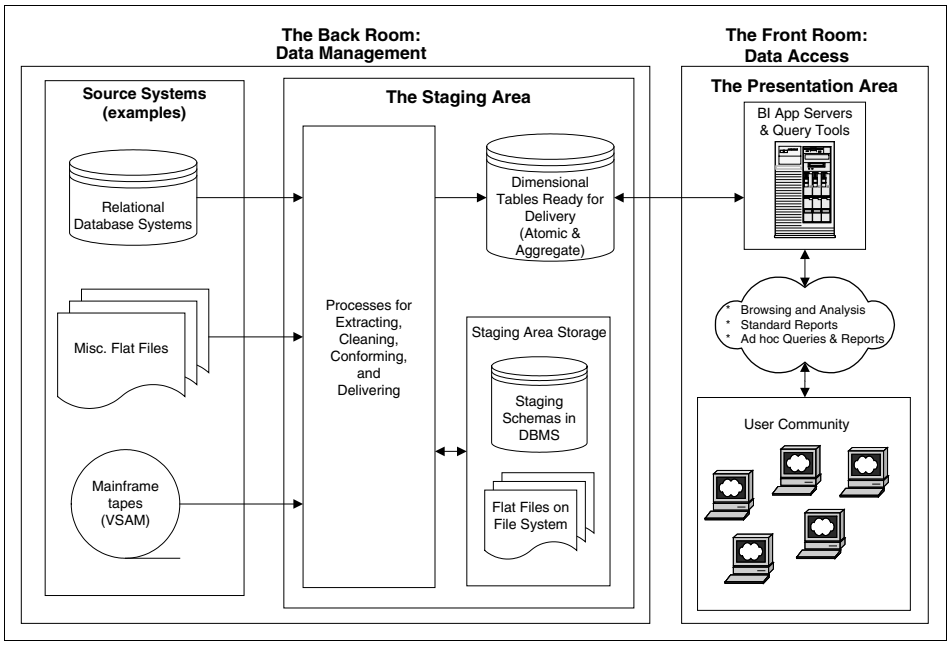

Figure 1.1 The back room and front room of a data warehouse.

Think of a restaurant. Imagine that patrons of the restaurant are end users and the food is data. When food is offered to patrons in the dining room, it is served and situated exactly as they expect: clean, organized, and presented in a way that each piece can be easily identified and consumed.

Meanwhile, before the food enters the dining room, it is prepared in the kitchen under the supervision of an experienced chef. In the kitchen the food is selected, cleaned, sliced, cooked, and prepared for presentation. The kitchen is a working area, off limits to the patrons of the restaurant. In the best restaurants, the kitchen is completely hidden from its customers—exposure to the kitchen, where their food is still a work-in-progress, spoils the customer's ultimate dining experience. If a customer requests information about the preparation of food, the chef must come out from the kitchen to meet the customer in the dining room—a safe, clean environment where the customer is comfortable—to explain the food preparation process.

The staging area is the kitchen of the data warehouse. It is a place accessible only to experienced data integration professionals. It is a back-room facility, completely off limits to end users, where the data is placed after it is extracted from the source systems, cleansed, manipulated, and prepared to be loaded to the presentation layer of the data warehouse. Any metadata generated by the ETL process that is useful to end users must come out of the back room and be offered in the presentation area of the data warehouse.

Prohibiting data access in the back room kitchen relieves the ETL team from:

- Providing detailed security at a row, column, or applications level
- Building query performance-enhancing indexes and aggregations
- Providing continuous up-time under service-level agreements
- Guaranteeing that all data sets are consistent with each other

We need to do all these things, but in the front room, not the back room. In fact, the issue of data access is really the crucial distinction between the back room and the front room. If you make a few exceptions and allow end user clients to access the back room structures directly, you have, in our opinion, fatally compromised the data warehouse.

Returning to the kitchen, we often use the word *staging* to describe discrete steps in the back room. Staging almost always implies a temporary or permanent physical snapshot of data. There are four staging steps found in almost every data warehouse, as shown in Figure 1.2, which is the same four-step data flow thread we introduce in the this book's introduction, but with the staging step explicitly shown. Throughout this book, we assume that every ETL system supporting the data warehouse is structured with these four steps and that data is staged (written to the disk) in parallel with

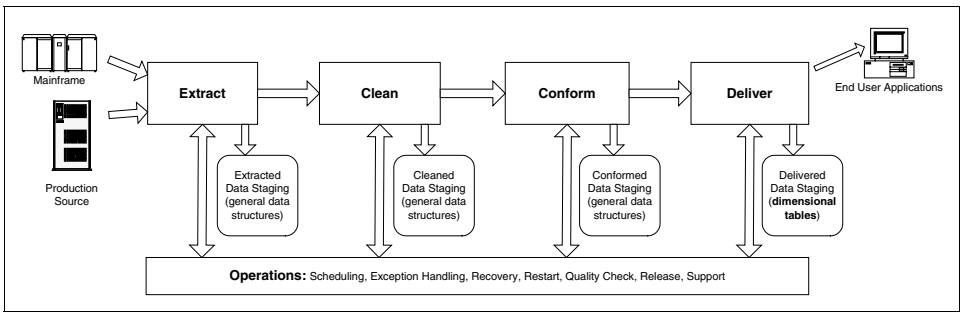

Figure 1.2 The Four Staging Steps of a Data Warehouse.

the data being transferred to the next stage. The central chapters of this book are organized around these steps. The four steps are:

1. **Extracting**. The raw data coming from the source systems is usually written directly to disk with some minimal restructuring but before significant content transformation takes place. Data from structured source systems (such as IMS databases, or XML data sets) often is written to flat files or relational tables in this step. This allows the original extract to be as simple and as fast as possible and allows greater flexibility to restart the extract if there is an interruption. Initially captured data can then be read multiple times as necessary to support the succeeding steps. In some cases, initially captured data is discarded after the cleaning step is completed, and in other cases data is kept as a long-term archival backup. The initially captured data may also be saved for at least one capture cycle so that the differences between successive extracts can be computed.

   
   We save the serious content transformations for the cleaning and conforming steps, but the best place to resolve certain legacy data format issues is in the extract step. These format issues include resolving repeating groups, REDEFINEs, and overloaded columns and performing low-level data conversions, including converting bit encoding to character, EBCDIC to ASCII, and packed decimal to integer. We discuss these steps in detail in Chapter 3.

2. **Cleaning.** In most cases, the level of data quality acceptable for the source systems is different from the quality required by the data warehouse. Data quality processing may involve many discrete steps, including checking for valid values (is the zip code present and is it in the range of valid values?), ensuring consistency across values (are the zip code and the city consistent?), removing duplicates (does the same customer appear twice with slightly different attributes?), and checking whether complex business rules and procedures have been enforced (does the Platinum customer have the associated extended

credit status?). Data-cleaning transformations may even involve human intervention and the exercise of judgment. The results of the data-cleaning step are often saved semipermanently because the transformations required are difficult and irreversible. It is an interesting question in any environment whether the cleaned data can be fed back to the sources systems to improve their data and reduce the need to process the same data problems over and over with each extract. Even if the cleaned data cannot be physically fed back to the source systems, the data exceptions should be reported to build a case for improvements in the source system. These data issues are also important for the final business intelligence (BI) user community.

3. **Conforming.** Data conformation is required whenever two or more data sources are merged in the data warehouse. Separate data sources cannot be queried together unless some or all of the textual labels in these sources have been made identical and unless similar numeric measures have been mathematically rationalized so that differences and ratios between these measures make sense. Data conformation is a significant step that is more than simple data cleaning. Data conformation requires an enterprise-wide agreement to use standardized domains and measures. We discuss this step extensively in the book when we talk about conformed dimensions and conformed facts in Chapters 5 and 6.
4. **Delivering.** The whole point of the back room is to make the data ready for querying. The final and crucial back-room step is physically structuring the data into a set of simple, symmetric schemas known as dimensional models, or equivalently, star schemas. These schemas significantly reduce query times and simplify application development. Dimensional schemas are required by many query tools, and these schemas are a necessary basis for constructing OLAP cubes. We take the strong view in this book that dimensionally modeled tables should be the target of every data warehouse back room. In Chapter 5 we carefully describe the structures of dimensional tables, and we give a fairly complete justification for building the data warehouse around these structures. For a more comprehensive treatment of dimensional modeling, please refer to the other Toolkit books, especially the *Data Warehouse Toolkit*, Second Edition (Wiley, 2002).

Figure 1.2 makes it look like you must do all the extracting, cleaning, conforming and delivering serially with well-defined boundaries between each pair of steps. In practice, there will multiple simultaneous flows of data in the ETL system, and frequently some of the cleaning steps are embedded in the logic that performs extraction.

The ODS has been absorbed by the data warehouse.

Ten years ago, the operational data store (ODS) was a separate system that sat between the source transactional systems and the data warehouse. It was a *hot extract* that was made available to end users to answer a narrow range of urgent operational questions, such as "was the order shipped?" or "was the payment made?" The ODS was particularly valuable when the ETL processes of the main data warehouse delayed the availability of the data or aggregated the data so that these narrow questions could not be asked.

In most cases, no attempt was made to transform a particular ODS's content to work with other systems. The ODS was a hot query extract from a single source.

The ODS also served as a source of data for the data warehouse itself because the ODS was an extraction from the transactional systems. In some cases, the ODS served only this function and was not used for querying. This is why the ODS has always had two personalities: one for querying and one for being a source for the data warehouse.

The ODS as a separate system outside the data warehouse is no longer necessary. Modern data warehouses now routinely extract data on a daily basis, and some of the new real-time techniques allow the data warehouse to always be completely current. Data warehouses in general have become far more operationally oriented than in the past. The footprints of the conventional data warehouse and the ODS now overlap so completely that it is not fruitful to make a distinction between the two kinds of systems.

Finally, both the early ODSs and modern data warehouses frequently include an interface that allows end users to modify production data directly.

The Front Room – Data Access

Accessing data in the presentation area of the data warehouse is a client, or follow-on, project that must be closely coordinated with the building and managing of the ETL system. The whole purpose of the ETL system is to feed the presentation layer of dimensionally modeled tables that are directly accessed by query tools, report writers, dashboards, and OLAP cubes. The data in the front room is what end users actually see.

Data marts are an important component of the front room. A data mart is a set of dimensional tables supporting a business process. Some authors refer to business processes as subject areas. *Subject area* is a fuzzy phrase with multiple meanings. For example, we've heard people refer to subject areas as *products, customers, and orders*. But we believe there is a big difference between product and customer entities and true measurement-intensive processes such as orders. In our view, data marts are always

measurement-intensive subject areas (like orders), and they are surrounded by descriptive entities like products and customers.

Although this book is not about using data marts, we need to make some strong statements about them.

1. **Data marts are based on the source of data, not on a department's view of data.** In other words, there is only one orders data mart in a product-oriented company. All the end user query tools and applications in various departments access this data mart to have a single, consistently labeled version of orders.
2. **Data marts contain all atomic detail needed to support drilling down to the lowest level.** The view that data marts consist only of aggregated data is one of the most fundamental mistakes a data warehouse designer can make. Aggregated data in the absence of the lowest-level atomic data *presupposes the business question* and makes drilling down impossible. We will see that a data mart should consist of a continuous pyramid of identically structured dimensional tables, always beginning with the atomic data as the foundation.
3. **Data marts can be centrally controlled or decentralized.** In other words, an enterprise data warehouse can be physically centralized on a single machine and the deployment of data marts can wait until a certain level of integration takes place in the ETL staging areas, or the data marts can be developed separately and asynchronously while at the same time participating in the enterprise's conformed dimensions and facts. We believe that the extreme of a fully centralized and fully prebuilt data warehouse is an ideal that is interesting to talk about but is not realistic. A much more realistic scenario is the incrementally developed and partially decentralized data warehouse environment. After all, organizations are constantly changing, acquiring new data sources, and needing new perspectives. So in a real environment, we must focus on incremental and adaptable strategies for building data warehouses, rather than on idealistic visions of controlling all information before a data warehouse is implemented.

There are many tasks and responsibilities in the front room that are outside the scope of this book. Just so there is no confusion, we do *not* talk in this book about:

- Indexing dimensional tables in the presentation area for query performance
- Choosing front-end tools, including query tools, report writers, and dashboards
- Writing SQL to solve end user queries

- Data-mining techniques
- Forecasting, behavior scoring, and calculating allocations
- Security on the tables and applications accessible by end users
- Metadata supporting end user tools
- End user training and documentation

This book is about the ETL systems for getting data out of its original source system and delivering it to the front room.

The Mission of the Data Warehouse

The mission of the data warehouse is to publish the organization's data assets to most effectively support decision making. The key word in this mission statement is *publish*. Just as the success of a conventional publication like a magazine begins and ends with its readers, the success of a data warehouse begins and ends with its end users. Since the data warehouse is a decision support system, our main criterion of success is whether the data warehouse effectively contributes to the most important decision-making processes in the organization. Although the costs of hardware, software, labor, consulting services, and maintenance have to be managed carefully, the hidden costs of failing to support the important decisions of an organization are potentially much larger. The tangible costs of a data warehouse, managed by IT, are tactical, but the more important costs and benefits of decision support are strategic.

Transaction database applications have been penetrating the corporate world for over 30 years. Although we have entered data into dedicated transaction applications for decades, it has become apparent that getting the data out of these systems for analytic purposes is too difficult. Billions of dollars have been spent on database applications, and their data is kept prisoner within them. An immeasurable amount of time is spent trying to get data from transaction systems, but like navigating through a labyrinth, most of that time is spent hitting dead ends. The ETL system must play a major role in handing the data to the final end user applications in a usable form.

Building a comprehensive, reliable data warehouse is a significant task that revolves around a set of standard components. The most important and fundamental components of the data warehouse are the back room and the front room. This book is about the back room.

What the Data Warehouse Is

Data warehousing is the process of taking data from legacy and transaction database systems and transforming it into organized information in a

user-friendly format to encourage data analysis and support fact-based business decision making. The process that involves transforming data from its original format to a dimensional data store accounts for at least 70 percent of the time, effort, and expense of most data warehouse projects. After implementing many data warehouses, we've determined that a data warehouse should have the following definition:

> *A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making.*

We've come up with this definition to alleviate confusion about data-warehouse implementation costs. Historically, the most visible part of a data warehouse project is the data access portion—usually in the form of products—and some attention is brought to the dimensional model. But by spotlighting only those portions, a gaping hole is left out of the data warehouse lifecycle. When it comes time to make the data warehouse a reality, the data access tool can be in place, and the dimensional model can be created, but then it takes many months from that point until the data warehouse is actually usable because the ETL process still needs to be completed.

By bringing attention to building the back room data management component, data warehouse sponsors are better positioned to envision the real value of the data warehouse—to support decision making by the end users—and allot realistic budgets to building data warehouses.

Unanticipated delays can make the data warehouse project appear to be a failure, but building the ETL process should not be an unanticipated delay. The data warehouse team usually knows that the ETL process consumes the majority of the time to build the data warehouse. The perception of delays can be avoided if the data warehouse sponsors are aware that the deployment of the data warehouse is dependent on the completion of the ETL process. The biggest risk to the timely completion of the ETL system comes from encountering unexpected data-quality problems. This risk can be mitigated with the data-profiling techniques discussed in Chapter 4.

What the Data Warehouse Is Not

What constitutes a data warehouse is often misunderstood. To this day, you can ask ten experts to define a data warehouse, and you are likely to get ten different responses. The biggest disparity usually falls in describing exactly what components are considered to be part of the data warehouse project. To clear up any misconceptions, anyone who is going to be part of a data warehouse team, especially on the ETL team, must know his or her boundaries.

The environment of a data warehouse includes several components, each with its own suite of designs, techniques, tools, and products. The most important thing to remember is that none of these things alone constitutes a data warehouse. The ETL system is a major component of the data warehouse, but many other components are required for a complete implementation. Throughout our experiences of implementing data warehouses, we've seen team members struggling with the same misconceptions over and over again. The top five things the data warehouse is mistaken to be are as follows:

1. **A product.** Contrary to many vendor claims, you cannot buy a data warehouse. A data warehouse includes system analysis, data manipulation and cleansing, data movement, and finally dimensional modeling and data access. No single product can achieve all of the tasks involved in building a data warehouse.
2. **A language.** One cannot learn to *code* a data warehouse in the way you learn to implement XML, SQL, VB, or any other programming language. The data warehouse is composed of several components, each likely to require one or more programming or data-specification languages.
3. **A project.** A properly deployed data warehouse consists of many projects (and phases of projects). Any attempt to deploy a data warehouse as a single project will almost certainly fail. Successful data warehouses plan at the enterprise level yet deploy manageable dimensional data marts. Each data mart is typically considered a separate project with its own timeline and budget. A crucial factor is that each data mart contains conformed dimensions and standardized facts so that each integrates into a single cohesive unit—the enterprise data warehouse. The enterprise data warehouse evolves and grows as each data mart project is completed. A better way to think of a data warehouse is as a **process**, not as a project.
4. **A data model.** A data model alone does not make a data warehouse. Recall that the data warehouse is a comprehensive process that, by definition, must include the ETL process. After all, without data, even the best-designed data model is useless.
5. **A copy of your transaction system.** A common mistake is to believe copying your operational system into a separate reporting system creates a data warehouse. Just as the data model alone does not create a data warehouse, neither does executing the data movement process without restructuring the data store.

Industry Terms Not Used Consistently

In this section, we call out industry terms that are given different meanings by different writers. There is probably no realistic hope of getting the industry to settle on uniform definitions of these terms, but at least we can take a clear stand on how we use the terms in this book.

Data Mart

Other authors frequently define a data mart as an *aggregated set of data prebuilt to answer specific business questions for a given department*. Of course, this definition contains its own criticism! In this book and in our writings for the last decade, we have consistently defined a data mart as a process-oriented subset of the overall organization's data based on a foundation of atomic data, and that depends only on the physics of the data-measurement events, not on the anticipated user's questions. Note the differences among data mart definitions:

CORRECT DEFINITION	MISGUIDED DEFINITION
Process Based	Department Based
Atomic Data Foundation	Aggregated Data Only
Data Measurement Based	User Question Based

Our data marts (call them *dimensional data marts*) look the same to all observers and would be implemented identically by anyone with access to the underlying measurement events. Furthermore, since dimensional data marts are always based on the most atomic data, these data marts are impervious to changes in application focus; by definition, they contain all the detail that is possible from the original sources. Data marts constructed according to the misguided definitions will be unable to handle changing business requirements because the details have been presummarized.

Enterprise Data Warehouse (EDW)

EDW is sometimes used as the name the name for a specific design approach (as contrasted with the uncapitalized *enterprise data warehouse*, which refers generically to the data warehouse assets of a large organization). Many people also refer to the EDW as the *CIF*, or *Corporate Information Factory*. The EDW approach differs materially from the Data Warehouse Bus Architecture approach described in our Toolkit books. EDW embodies a number of related themes that need to be contrasted individually with the DW Bus approach. It may be helpful to separate logical issues from physical issues for a moment.

Logically, both approaches advocate a consistent set of definitions that rationalize the different data sources scattered around the organization. In the case of the DW Bus, the consistent set of definitions takes the form of conformed dimensions and conformed facts. With the EDW approach, the consistency seems much more amorphous. You must take it on faith that if you have a single, highly normalized ER model of all the enterprise's information, you then know how to administer hundreds or thousands of tables consistently. But, overlooking this lack of precision, one might argue that the two approaches are in agreement up to this point. Both approaches strive to apply a unifying coherence to all the distributed data sources.

Even if we have a tenuous agreement that both approaches have the same goal of creating a consistent representation of an organization's data, as soon as you move into physical design and deployment issues, the differences between the EDW and the DW Bus become really glaring.

Conformed dimensions and conformed facts take on specific forms in the DW Bus architecture. Conformed dimensions have common fields, and the respective domains of the values in these fields are the same. That guarantees that you can perform separate queries on remote fact tables connected to these dimensions and you will be able to merge the columns into a final result. This is, of course, drill across. We have written extensively on the steps required to administer conformed dimensions and conformed facts in a distributed data warehouse environment. We have never seen a comparable set of specific guidelines for the EDW approach. We find that interesting because even in a physically centralized EDW, you have to store data in physically distinct table spaces, and that necessitates going through the same logic as the replication of conformed dimensions. But we have never seen systematic procedures described by EDW advocates for doing this. Which tables do you synchronously replicate between table spaces and when? The DW Bus procedures describe this in great detail.

The denormalized nature of the dimensions in the DW Bus design allows us to administer the natural time variance of a dimension in a predictable way (SCD types 1, 2, and 3). Again, in the highly normalized EDW world, we have not seen a comparable description of how to build and administer the equivalent of slowly changing dimensions. But it would seem to require copious use of time stamps on all the entities, together with a lot more key administration than the dimensional approach requires. By the way, the surrogate key approach we have described for administering SCDs actually has nothing to do with dimensional modeling. In an EDW, the root table of a normalized, snowflaked *dimension* would have to undergo exactly the same key administration (using either a surrogate key or a natural key plus a date) with the same number of repeated records if it tracked the same slowly changing time variance as the DW Bus version.

The denormalized nature of dimensions in the DW Bus design allows a systematic approach to defining aggregates, the single most powerful and cost effective way to increase the performance of a large data warehouse. The science of dimensional aggregation techniques is intimately linked to the use of conformed dimensions. The *shrunken* dimensions of an aggregate fact table are perfectly conformed subsets of the base dimensions in the DW Bus architecture. The EDW approach, again, has no systematic and documented approach for handling aggregates in the normalized environment or giving guidance to query tools and report writers for how to use aggregates. This issue interacts with *drilling down*, described in a moment.

Most important, a key assumption built into most EDW architectures is that the centralized data warehouse *releases* data marts. These data marts are often described as *built to answer a business question*, as described in the previous subsection on data-mart definitions. A final, unworkable assumption of the EDW is that if the user wants to ask a precise question involving atomic data, he or she must leave the aggregated dimensional data mart and descend into the 3NF atomic data located in the back room. EVERYTHING is wrong with this view in our opinion. All of the leverage we developed in the DW Bus is defeated by this two level architecture: drilling down through conformed dimensions to atomic data; uniform encoding of slowly changing dimensions; the use of performance-enhancing aggregates; and the sanctity of keeping the back room data-staging area off limits to query services.

Resolving Architectural Conflict: The Hybrid Bus Approach

Is it possible to reconcile the two architectural approaches? We think so. Throughout this book, we support the judicious use of normalized data structures for data cleaning. A really dirty data source benefits from the discipline of enforcing the many-to-1 relationships brought to the surface by the process of normalization. THEN we urge the ETL team to convert any such normalized structures into simple dimensional structures for the conforming and final handoff steps. This includes the atomic base layer of data. At this point, an IT organization that has already invested in normalized physical structures can leverage that investment. We can call this the *Hybrid Bus Approach*.

How the Data Warehouse Is Changing

As we write this book, the data warehouse is undergoing a significant change, perhaps the most significant change since the beginning of data

warehousing. Everything we have said in this chapter about the data warehouse supporting decision making remains true, but the focus of new development in the data warehouse is in many cases drastically more *operational* and *real time*. Although the basic front room and back room components of the data warehouse are still very necessary for these real-time applications, the traditional batch-file-oriented ETL processing is giving way to streaming ETL processing, and the traditional user-driven query and reporting tools are giving way to data-driven and event-driven dashboards. We describe these new developments and how they extend the central concepts of the data warehouse in Chapter 11.

The Mission of the ETL Team

We are finally in a position to succinctly describe the mission of the ETL team, using the vocabulary of this chapter. *The mission of the ETL team at the highest level is to build the back room of the data warehouse.* More specifically, the ETL system must:

- Deliver data most effectively to end user tools
- Add value to data in the cleaning and conforming steps
- Protect and document the lineage of data

We will see that in almost every data warehouse the back room must support four keys steps:

- Extracting data from the original sources
- Quality assuring and cleaning data
- Conforming the labels and measures in the data to achieve consistency across the original sources
- Delivering data in a physical format that can be used by query tools, report writers, and dashboards.

This book deals with each of these steps in great detail.