1 Basic Data Warehousing Distinctions

In this chapter…

They said it...

*"Once a distinction is drawn,
the spaces, states, or contents
on each side of the boundary,
being distinct, can be indicated."*
G. Spencer Brown[1]

AN ARCHITECTURE, NOT A PRODUCT

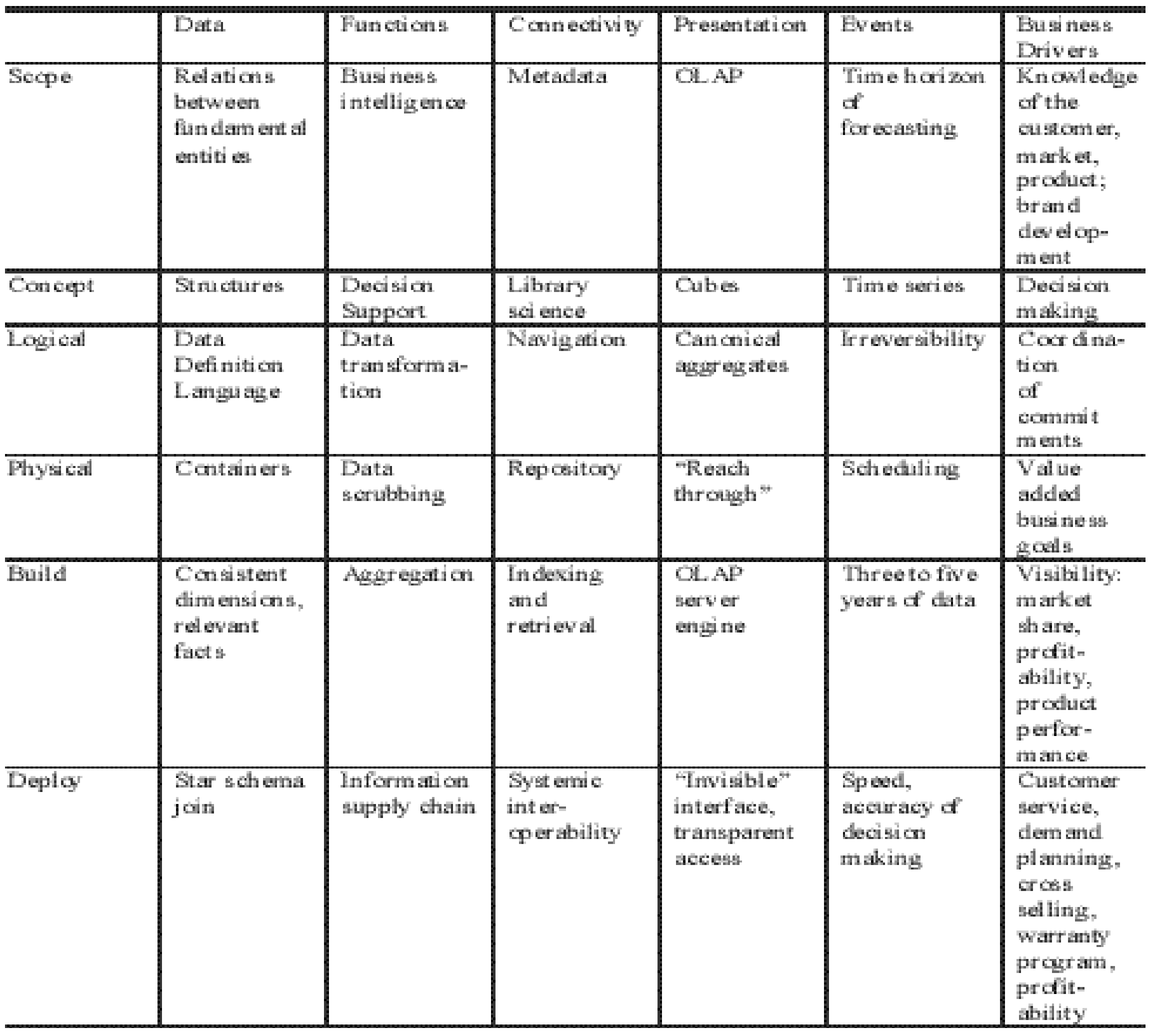
The data warehouse is not a software product or application. It is a system architecture. An *architecture* is an arrangement of methods according to first principles. These principles make possible business processes through client, network, and database software. These processes, in turn, provide knowledge, addressing fundamental business imperatives. This is an important point. Given that the data warehouse market place is filled with useful and innovative software products claiming to be "solutions," taking a moment to gain perspective is a necessary step. To be sure, these products have merit and deserve attention. Check lists will be provided later in the discussion for evaluating products, though the approach of this book is not product-oriented. Likewise, many of these products imply an architecture, because "no product is an island." However, the point remains—no product in itself is an architecture.

As an architecture, the data warehouse combines a number of products, each of which has operational uses besides data warehousing. Simply stated, a data warehouse architecture is a blue print, an arrangement, or a map.[2] When implemented, the architecture provides an infrastructure that makes possible business applications that deliver certain special forms of knowledge of the customer. These will be considered in detail shortly. The data warehouse system architecture provides patterns within which applications are consistently connected with one another and integrated with hardware, operating system, database, network, and interface software, and cross-referenced with business processes.

In elementary terms, a working set of data warehousing software includes a database engine, including stored procedures containing data access and manipulation logic; an analytic application server to perform complex business processes, such as forecasting, customer profiling, or promotion evaluation; an end-user interface, typically, a graphical user interface (GUI) on the client work station or desktop; and connectivity to bind the client and server(s), possibly across a wide area network (WAN) using an intranet-based web browser. Software to provide a navigation layer and all-important metadata (the subject of an entire chapter below) is required for the long-term viability of the warehouse. Provision must also be made for "on the fly," user-defined ad hoc inquiries against the data. This list of minimal essential components of a data warehouse will be helpful when planning or shopping for products. However, the main reason for making this list is to throw it away—at least temporarily. This is done to make a point. What makes the data warehouse a warehouse is not primarily all this software, but rather the way the software represents dimensions—and so provides knowledge—of the business enterprise: the customer, product, the channel, and their interaction in time and location (and other related structures). The way that these dimensions line up, or are aligned, with the business is the origin of the promise and the power of the data warehouse as a source of knowledge to be used in driving the business forward.

It is the essential nature of architecture to stand fast. It is the nature of requirements to be in flux. Although it is beneficial in constructing systems to freeze requirements prior to implementation and during migrations of system components, this is a temporary measure. Likewise, any architecture, no matter how robust and flexible, can be extended only so far. The point is that, in comparison with a data warehouse system architecture, end-user requirements are dynamic and fluctuating. Because these requirements are constantly evolving, dynamic, and undergoing refinement, building a system based on requirements alone risks surprises. With all-too-common inevitability, unanticipated events in the business environment cause external shocks to the data warehouse planning or construction effort. If the architecture is brittle, it will be overtaken by dynamic events, be cut off from business meaning, and be obsolete, even before implementation. On the other hand, if the architecture provides a flexible platform that encompasses a myriad of possibilities in a highly coherent, loosely coupled framework, it will be able to adapt to the inevitable slings and arrows of outrageous fortune, bend without breaking, and continue to be of service in a variety of evolving contexts.

Data warehouse architecture does its work by cross-referencing the essential features of any information system with the system construction methods. In this way, the conceptual and logical vision of the architecture gets physically implemented. For example, basic architectural categories refer to data, application functionality, connectivity, presentation (user interface), events (time series data or schedule), and business drivers. Any robust architecture implies methods for its own implementation. The progression is like that of a project implementation in time. It proceeds from scope, through conceptual, logical, and physical models. At this point, the target technology implementation—the representation of the data, functions, and rules in system hardware and software—becomes a constraining and overriding factor in the construction and deployment of the system. As an illustration of a robust flexible architecture, John Zachman's legendary framework for information systems architecture is an example of itself—it is a flexible and survivable construct encompassing many possible requirements scenarios (see Figure1.1 for an adaptation to data warehousing that it inspired).[3] Although originally published prior to the impact of the client-server revolution, the addition of human interface and presentation have brought that dimension up to date. The communication network—including the WAN dimension—was always a feature of the framework. That means that it is still friendly toward network-centric, Internet computing platforms. In fact, entire books have been written on this architecture (e.g., see Spewak and Hill, 1993). It is a superset of client-server architecture, arguably the most prevalent form in which



	Data	Functions	Connectivity	Presentation	Events	Business Drivers
Scope	Relations between fundamental entities	Business intelligence	Metadata	OLAP	Time horizon of forecasting	Knowledge of the customer, market, product; brand develop-ment
Concept	Structures	Decision Support	Library science	Cubes	Time series	Decision making
Logical	Data Definition Language	Data transforma-tion	Navigation	Canonical aggregates	Irreversibility	Coordina-tion of commit ments
Physical	Containers	Data scrubbing	Repository	"Reach through"	Scheduling	Value added business goals
Build	Consistent dimensions, relevant facts	Aggregation	Indexing and retrieval	OLAP server engine	Three to five years of data	Visibility: market share, profit-ability, product perfor-mance
Deploy	Star schema join	Information supply chain	Systemic inter-operability	"Invisible" interface, transparent access	Speed, accuracy of decision making	Customer service, demand planning, cross selling, warranty program, profit-ability

Figure 1.1
Data Warehouse Architecture

systems are currently being implemented. In the following discussion, architecture will provide a thread guiding us through layers of system components and functions—business drivers, data applications, connectivity, time series, work flow, and presentations.

Reading Figure 1.1 from left to right, a few special features are worth comment. These will be the target of detailed discussion in the remainder of the book. However, a preliminary pass will provide an overview and orientation.

Data warehousing emphasizes data structures distinguishing between dimensions such as customer product, time, place, and facts, such as quantity of product sold, delivered, or used by a customer at a particular occasion. This joining of different dimensions into a meaningful, unique fact structure provides the famous "star schema," which will be discussed in detail. Many of the functions (processes) of data warehousing require transforming data from the raw form in which it is produced in transactional systems into a form that provides decision support. This includes summarizing the data into meaningful aggregations that provide decision support perspective. It also includes scrubbing the data—purging it of inconsistencies or inaccuracies that may have occurred in traversing operational systems. The way in which data is transformed between transactional and decision support systems requires tracking and synchronizing the semantics from source to target systems. The aggregation of data requires tracing and synchronizing summary to details. When performed for dozens or hundreds of programs and systems, this requires the ability to index, store, catalog, and retrieve vast amounts of information about how these systems interoperate. The central role of a repository is akin to the coordination delivered by library science. Called *metadata*, this is one of the most challenging features of data warehousing and can crucially advance or limit the scalability, flexibility, and maintainability of the overall data warehousing architecture. In general, the way human factors show up in a data warehousing context is by "slicing and dicing" information on the desktop of the "power user." That is the realm of OLAP (on-line analytic processing). The time horizon of decision support events is significantly different than that of transactional systems. The latter is often focused on thirty-day "open inventory," whereas decision support applications such as forecasting require a time horizon of three to five years of available data. The acceleration of decision making and the making of better-informed decisions is an important deployment objective of data warehousing architecture. The business goals from which data warehousing aims to furnish architectural support are different than those basic to the day-to-day operation of the enterprise. These include strategic processes such as brand development, cross-selling based on knowledge of the customer, and a variety of supply chain and value chain (logistic and marketing) initiatives. In short, Figure 1.1 is a comprehensive overview, and much of the rest of this book may be read as a "drill down" on it.

Data warehouse architecture maps closely to the form of client-server computing. At the back-end is a data store, usually a relational database; at the front-end are desktop presentation tools to slice and dice the data cubes and aggregates returned from the data store; and in the middle is a variety of auxiliary applications—navigation, aggregation, analytic, and metadata layers—quite complex in themselves but designed to hide complexity and deliver a uniform and coherent interface to the end-user (see Figure 1.2: Client-Server Definition). In fact, the three layers described here—front-end, middle, and back-end—are not guaranteed to map neatly to the desktop work station (usually a PC), midsized NT or UNIX server in the middle, and mainframe or enterprise server at the back-end. As astute system architects have pointed out (see Loosley and Douglas, 1998), this is really a three-by-three matrix (see Figure 1.3: Alternate Client-Server Partitioning). Consider: The presentation, application, and database layers could all be implemented by computing cycles executing on a main-

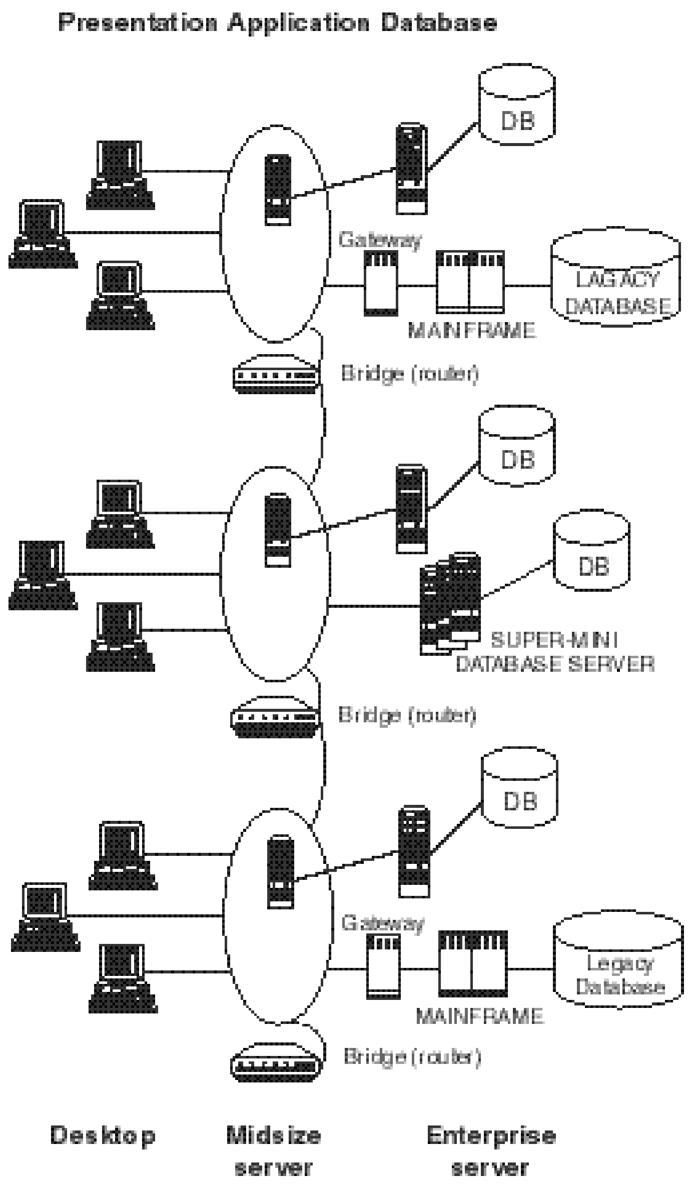


Figure 1.2
Client-Server Definition

frame, as in the days of 3270 dumb terminals. That is the bottom row of Figure 1.3, where presentation, application, and database are at the back-end, enterprise server layer. Alternatively, the presentation layer could be a process on a middle layer server, as occurs with the so-called X-Windows system (corresponding to the middle row in Figure 1.3, as indicated) or all three could be implemented on a high-powered workstation at the desktop by OLAP tools (will be treated in detail in a latter chapter). (Figure 1.3 actually shows connectivity to a database on the middle, server layer, but it is heavily front-end loaded.) Also different partitioning of the application functionality can be distributed between the fat client on the desktop, containing lots of validation and application logic and processes such as stored procedures, and triggers on the database at the back. In this scenario, client-server collapses into the two-tiered implementations, common in the early 1990s, when such tools as PowerBuilder and SQL Server made visible alternatives to mainframe data center computing.

How, then, does this apply to data warehousing? Data warehousing applications encompass the basic issues of marketing, brand (product) development, inventory or asset management, and generation of satisfaction for customers and stakeholders up and down the value chain. They provide the business analyst, demand planner, marketing manager, product development specialist, or knowledge worker in general with visibility to the behavior of fundamental quantitative features of the business processes vital to the firm. The business motivations are as multivariate as the goals of the business itself. Often, these applications are described as OLAPs to distinguish them from traditional day-to-day operations as on-line transaction processing (OLTP).[4]

To be sure, data warehousing is bound to be a data-centric architecture. That is, data-centric—having to do with databases—as opposed to computation-centric, or real-time- or operational-control-focused. Yet, if ad hoc, end-user inquiries against the data warehouse are being considered as an application, a connection with user-centric computing, previously restricted to the information center, is in view. In no way is this an add-on to or any kind of legacy system "solution," because many enterprise systems feed the data warehouse, including electronic commerce and enterprise resource planning systems. Still, the data warehouse is likely to require data extraction, scrubbing, or transformation from one or more legacy systems, along with other enterprise systems. These systems may be local or remote, that is, distributed. Thus, to accommodate data-intensive processing using even the most state-of-the art database management systems tools and techniques, the data warehouse will have to be a simple model that exploits common patterns among various business functions.

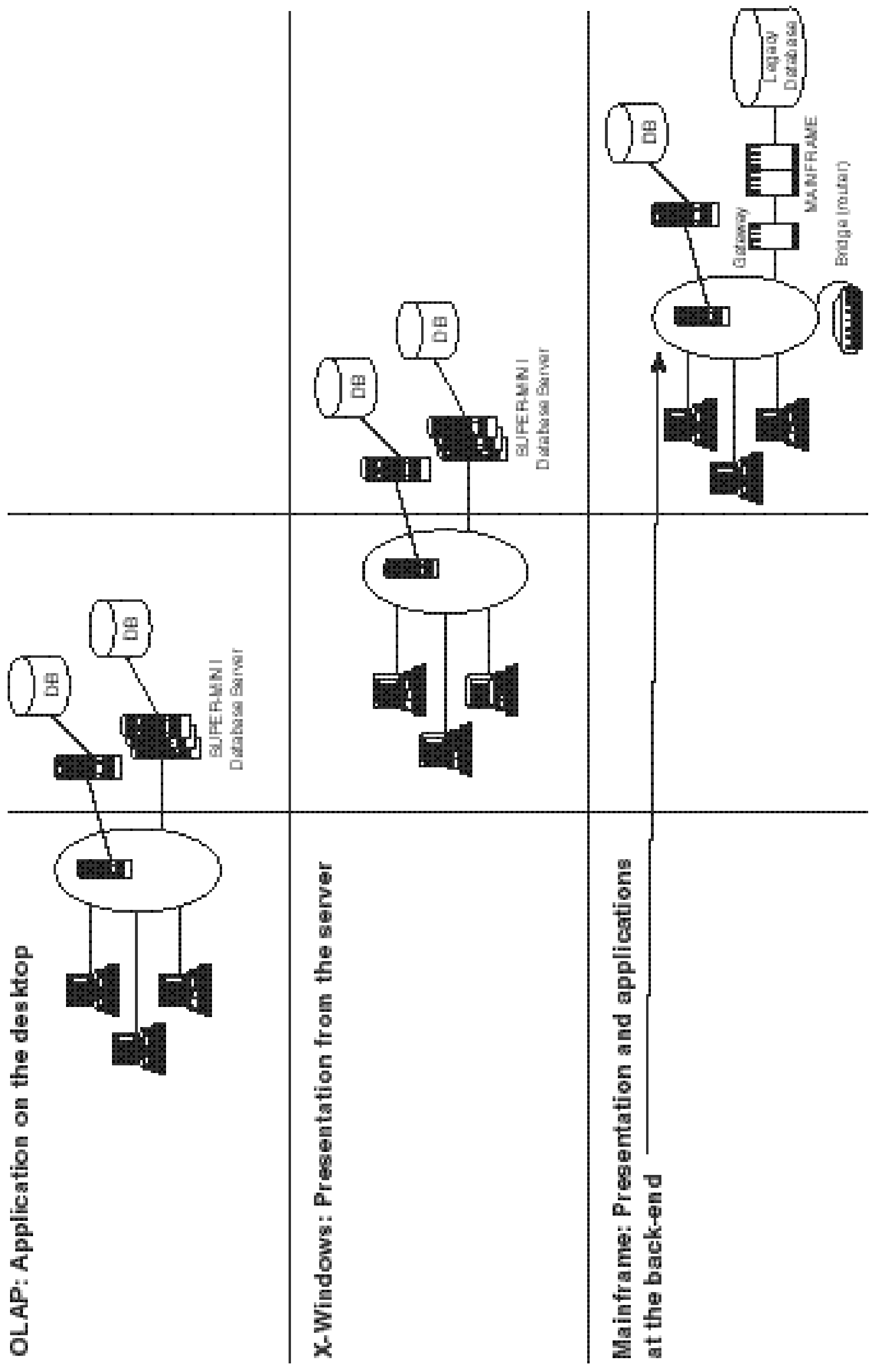

OLAP: Application on the desktop
DB
DB
SUPER-MINI Database Server
X-Windows: Presentation from the server
DB
DB
SUPER-MINI Database Server
Mainframe: Presentation and applications at the back-end
DB
Legacy Database
MAINFRAME
Gateway
Bridge (router)

Figure 1.3
Alternate Client-Server Partitioning

Like the rest of the planet, data warehouse systems are migrating in the direction of open Internet standards and protocols. Sometimes described as "the cavalry to the rescue of the IT department" (Keen, 1997), such standards promote the cooperation of customers and suppliers, along the material and informational supply chain out across WANs. Thus, although data warehouse content tends not to be the sort of sensitive data that firms routinely share with customers or suppliers, exceptions are to be found in situations like that at Wal-Mart, where suppliers are invited (i.e., required) to access the data warehouse in order to manage their own inventory control and stock replenishment.

Issue: Is Architecture Made Obsolete by Web Development

At least one industry analyst, Richard Fichera of the Giga Information Group, makes a strong case that considerable overlap exists between allegedly competing architectures and the important benefits derived from having any architecture as opposed to the particular advantages of a certain one. How does this claim hold up in the face of the claim that the web makes obsolete (or irrelevant) the requirement for enterprise architecture planning in any case. In an article on the subject accessible on the publicly available portion of www.gigaweb.com, Mr. Fichera engages this issue. Web development often is initiated by groups outside of the IT mainstream using bottom up and opportunistic methods to get the job done. When operating on Internet time there is a severe penalty for slow results, and marketing, production, and customer care departments are strongly incented to fast track system development around the defined process of centralized IT planning and justification. Use is made of staffing and outside consulting resources that cut their teeth on Internet development independent of the mainstream of the professional IT organization. The response is threefold. The development of the digital economy *via* the web has taken everyone by surprise. So everyone is scrambling to catch up. *Gonzo* development works well for web publishing and is valid as a prototype. However, by the time mission critical 7x24 electronic commerce applications are in question, where real customer relations are at stake, the requirement returns with full force for rigorous, rapid planning, execution, backup management, version control, change control, and scheduling. The trade off remains operative between short term gratification *via* tactical solutions and long term business advantages sustainable thanks to infrastructure not purchasable by anyone (e.g., the competition) as a vendor package. There is a fire hose of data coming at the enterprise off of business activities on the web and nothing less than an enterprise approach will be sustainable. Gaining business advantage from the output of such activities requires a disciplined approach as exemplified by data warehousing. This is a natural conclusion from the experiences and insights marshaled here at the executive level.

Data warehouse information frequently deals with time series data—how sales or deliveries or prices vary from one time period to another. This data can be complex in a variety of ways. It's abstract, dealing with multiple dimensions, including time, place (geography), product, and a host of other dimensions to be considered. It is voluminous. There is a lot of it. Also it tends to involve comparisons of quantities. To manage this complexity, visual presentation is useful. For selected people, visual presentation may be an operational necessity. Complexity becomes accessible and manageable through a clear graphical presentation. The presentation of complex time series data by means of a simple linear design makes for an intuitive and pleasing elegance. These are important pointers for any designer of GUIs on the desktop. The chief idea of human interface design is to get the interface out of the way of the presentation. In other words, the best interface is invisible (easier said than done, to be sure). When widgets, buttons, and boxes are in the foreground, the meaning is not accessible, and the point is lost. The interface should be like a good waiter—there when you need him but otherwise unobtrusive.

Data warehouse architecture enables a particular variation on work flow. In terms of three-tiered client-server, this is situated in the middle layer, where an analytic application engine implements business processes relevant to decision support, such as customer profiling, forecasting, product promotion, and the like. There is a difference between work flow as coordination and as assembly line. The groups of people brought together by a data warehouse system are likely to be cooperating in a different kind of work flow than that characteristic of operational transactional systems. Rather than an information assembly line, where repetitive processes are automated and passed around in input-output stepwise progress, the data warehouse enables work flow as a coordination of interconnecting commitments—requests and satisfactions of requests. Commitments? For example, estimating product demand based on the knowledge of prior experience and promotions implies a commitment to source and deliver the product. Without the interconnection of a commitment between availability and source, the planning operation is an idle wheel. The work and the product literally do not flow. With the interconnection between source and delivery, however, real improvements are possible in supply chain hand-offs, reductions in inventory show up, and just-in-time processing works for all the stakeholders, employees, and customers. For the mechanism represented by the data warehouse system to be engaged, commitment is of the essence. That commitment shows up as a shared architecture, providing work flow as a coordination of commitments. Naturally, those commitments extend beyond the boundary of an individual enterprise. Commitment is usually not thought of as a feature of architecture, but it is. So architecture includes technology but is not limited to it. The data warehouse includes a representation of features of the market itself—customer buying behavior, deliveries, and relevant service behavior. All of these and more are made visible and captured by the system architecture.

The data warehouse architecture supports processes whose time horizon is different than that of day-to-day system operations. Here is a system whose objective is to rise above the perspective of day-to-day operations. The objective is to reduce risk of surprises and to increase control of performance and operations of the business by envisioning options and opportunities coming at us out of the future. The idea that the future will be like the past is a useful working assumption. Many data warehouses use three to five years of historical data, subjected to statistical corrections, to make inferences about the behavior of customers, products, services, and sales. The time horizon is long, transactions are long, and the business events are of interest from a decision support perspective. This doesn't mean that the future is an unknown quantity coming at us out of the void, though that is undoubtedly all too often the case. What this means is that one way of knowing the future is by bringing it forth from our actions, based on a plan of our own formulation. Data warehousing provides business with a powerful tool for doing so. Note, however, that the better thought out and flexible the plan, the more powerful and useful the data warehouse architecture is likely to be.

To encourage the use of common patterns among system technology components, a data warehouse architecture has to provide for more than a dozen essential features. These features are not unique to data warehouse systems but the data warehouse presents certain priorities in understanding and using them. The more valuable the information contained in the warehouse, the more essential is security. It must provide for defense, preferably using group authorizations, against intentional or accidental attack or damage, based on authorization control on the desktop, in the network layer, and at the back-end data stores. The whole point of an architecture is to escape from the birth and death cycle of ever-changing business requirements. Therefore, the architecture must be flexible—that is, it must be able to adapt to changing business requirements and contexts, allowing for effective modification, administration, and management. Because corporate mergers and divestitures are the order of the day, the data warehouse must be transportable. Components of the system are able to be installed in a variety of hardware/software implementations without heroic rewriting. This, in turn, implies the use of open, standards-based components with reusability and important characteristics. The design guideline of high coherence and loose coupling remains valid. The parts of the system should be tight and coherent. They should able to stand on their own behind well-defined interfaces that represent the contract by which they are able to be accessed. At the same time, they should be loosely coupled. A change in one module must not cause a bug to show up as a side effect “over there” in another module. The data should be hidden behind consistent, well-defined interfaces. However, if you know the API (application programming interface), then it must be available. Information and functionality are available by defined interfaces, paths, and connections from elsewhere in the system, including across distributed nodes. Data and transactions must work together across environments from different vendors and implementations. Complexity of implementation is hidden; simplicity of design is presented. Scalability remains one of the nontrivial and highly desirable features of any system architecture. It is particularly important in a data warehousing context where volumes of data are large. A scalable system is one where performance improves linearly or nearly linearly as system components are added to handle additional data volume, users, and processing requirements. In this context, flexibility is the inverse side of scalability. Performance in the face of evolving business context and volume deteriorates gradually without sudden loss of viability, allowing time to respond with corrective action. Those general features of a data warehouse architecture that focus on the technology aspects are summarized in Table 1.1.

Table 1.1: *Features of a Data Warehouse Technical Architecture*

Secure	Defensible against intentional or accident attack or damage, based on authorization control on the desktop, in the network layer, and at the back-end data stores
Robust	Able to adapt to changing business requirements and contexts, allowing for effective modification, administration, and management; gradual (not sudden) degradation under stress
Transportable	Components of the system can be installed in a variety of hardware/software implementations without heroic rewriting
Open	Implemented either in publicly available standards, independent of the power of single firm, or in the dominant technology design, representing a de facto standard; application programming interfaces are published and modified by change control
Coherent	Made of individual components with defined interfaces that are unaffected by changes in the implementation of other components
Maintainable	Useful life of the system can be prolonged, perhaps indefinitely, by routine attention to features that change or (in effect) wear out due to changes in the environment
Extensible	The system components can be extended to new, unanticipated contexts and situations
Instrumented	The system is provided with built-in sensors or data gathering devices so that if things go wrong, diagnosis is possible without heroic efforts
Reusable	Components are well defined, subjected to configuration management, and documented, so that they can be used again in different contexts
Connected	Information and functionality are available by defined interfaces, paths, and connections from elsewhere in the system, including across distributed nodes
Collaborative	Data and transactions work across environments from different vendors and implementations
Hidden [Data Hiding]	Interfaces are well defined, complexity of implementation is hidden, simplicity of design is presented
Scalable	Performance improves linearly or nearly linearly as system components are added to handle additional data volume, users, and processing requirements
Flexible	Performance in the face of evolving business context and environment deteriorates gradually without sudden loss of viability, allowing time to respond with corrective action

THE ONE FUNDAMENTAL QUESTION

This points directly to *the* one fundamental business question to be answered in the architecture and building of an enterprise data warehouse, and it does so in such a way as to define and address the constraints on constructing a data warehouse aligned with business imperatives. This is the reason why building and operating the data warehouse provides such a powerful lever—a mechanism for management in moving the enterprise in the direction of addressing critical business issues relating to brand development, customer intimacy, and the information supply chain in time. Successful data warehouses are those that accurately represent these issues and the possibility of engaging and resolving them productively in the software, capturing the necessary data, transforming it in the information supply chain, and making it accessible to business analysts and decision makers.

THE ONE QUESTION— THE THOUSAND AND ONE ANSWERS…

This is *the* fundamental question that the data warehouse is designed and implemented to answer. The number of dimensions that can be "hung" off of a statement of this form tends to grow rapidly. However, the simple and basic idea is to identify, discover, and track who is buying what and at what specific time and place are they doing so. Thus, in its 1001 forms, the data warehouse is designed to answer the question, Who is buying what—and when and where are they doing so? Who [which customer] is buying / using / delivering / shipping / ordering / returning what [products/services] from what outlet / store / clinic / branch [location] on what occasion [when] and why [causation]? With this fundamental question in view, let us now take a step back and consider the basic distinctions on which the data warehouse will be built.

THE FIRST DISTINCTION: TRANSACTION AND DECISION SUPPORT SYSTEMS

The first distinction is between transactional and decision support systems. This fundamental distinction is between information technology systems that drive business operations on a day-to-day basis and those that determine the outcome of decisions about strategic moves in the market relating to customers, products, suppliers, etc., and the timing of exchanges between them and the firm. This is the difference between transactional (operational) and data warehousing systems. Transactional systems address everyday operations about business events significant to individual customers, products, suppliers, and those mandated by government regulations and agencies. Transactional systems may indeed be strategic, as well as tactical, but strategic the way that a hotel chain that ties its reservation system into that of an airline reservation system is strategic. The competitive advantage of locking down one reservation at the same time that the other one is being made is a good move under any interpretation. Decision support systems, on the other hand, take a broader and more global perspective, especially in terms of a longer and more continuous time horizon over which sales, usage, and product trends are compared and contrasted. Decision support systems, likewise, can be tactical, as well as strategic, but tactical the way that having the right amount of product at the right store at the right time is tactical, even within an overall strategy of reducing inventory through better forecasting.

Much of the transactional work of operational systems concerns answering day-to-day questions about customer service inquiries, product function, delivery schedules, payment schedules, exceptions to expected outcomes, and relations with other business entities (suppliers and customers, up and down stream roles and functions).

As an illustration, transactional questions from an operational system might include:

- When did Mr. Ralph's package with tracking number 423 leave the dock?
- How much is owed on Ms. Arendt's order as of 01/15/1999?
- Has Mr. Wittgenstein's insurance coverage taken effect as of the last week in September?
- How much does Dr. Ramon charge for procedure RKO?
- How many times did Mr. Douglas see Dr. Ramon and on what dates?

These matters are mission critical to the day-to-day business transaction. *They are not data warehouse questions*. To be sure, these are important matters and significant transactions, especially to the customer. But there is a "but." From a data warehouse point of view, these transactions are logically prior to—one might say "subatomic" with respect to—a business relationship with a customer, developing a brand, or driving customer- and product-sustaining activities.

What does *subatomic* mean here? This addresses the level of granularity at which data warehouse facts are defined. For example, if an atomic fact is defined as how much of product *Y* customer *X* bought on day *Z,* then a transaction for product *Y* in the morning, at noon, and at night are subatomic. To get to the basic, atomic level, the three purchases must be added (aggregated) up to the atomic level. That is not to be taken for granted.

Instead of addressing internal operations and customer service issues, data warehouse questions tend to address marketing and sales analysis, including sales trend and sales analysis, including competitive analysis, product comparisons in a regional (space) or seasonal (time) framework, causal connections between events (advanced applications), usage (including sales) trends, and the behavior of customers as part of a group or profile. Examples of the kinds of questions that get posed:

- What is the weekly shipment for each product, along with the weekly year-to-date amounts, for the eastern region?
- What is the profit attributable to department *XYZ*?
- What are sales by responsible product line manager for September?
- What is the productivity of factory ABC by materials input, shift, and week ending date?
- What is the six-month moving average of product shipments for the last quarter of 1998?
- What are the top ten products, based on first quarter sales and sorted by product name?
- What is the cost of health care services used by employer group *X* for the third quarter?
- Which dinner menu of the seven optional menus generates the most sales?

These questions invite answers that are different than those required by day-to-day operations. They are both more general and more abstract—more general because they often deal with accumulations of data by brand, region, marketing manager, or extended time period; more focused on understanding and diagnosis than mere descriptions of what is so; more abstract, because quantities are often compared across time and place.

DATA WAREHOUSE SOURCES OF DATA

Data warehouses are sourced from three kinds of transactional systems. These include legacy, ERP (enterprise resource planning), and electronic commerce systems (see Figure 1.4: Data Warehouse Sources of Data.)

Legacy systems used to mean systems hosted on mainframes or midrange computers, usually developed prior to client-server computing initiatives that emerged in the early 1990s. Recently, the term has come to refer only half jokingly to any system already in production. The surprising thing about legacy systems has been their durability. People building systems in the 1970s never expected them to still be around twenty-five years later, but many of them are. It turns out that, if designed properly—a big “if”—software is basically immortal. Once the curve of discovered bugs levels off, it (the legacy system) is really more stable and robust than recent alternatives,which have not proven themselves in a variety of different situations. That is also the case with many legacy systems, which are the bread-and-butter systems for day-to-day transactions and business operations.

However, everyone agrees that the year 2000 bug does present a new situation and one that many older systems will be challenged to accommodate because of near-sighted design decisions of years gone by. This bug is really a highly dramatic special case of lack of attention to testing and accountability for design in a profession that was still struggling to find rigorous standards. It is driving the replacement of many legacy systems with ERP packages. Rather than try to disentangle the years of patches, maintenance, and enhancements under which the millennium bug lurks, replacing the entire configuration actually seems easier. At least it is easier to justify from a management perspective, because additional business requirements can be accommodated. The implementation, however, is in no way easy.

Make no mistake about it, ERP systems are transactional systems. They are run-your-business databases and applications whose data structures are highly snowflaked, normalized, and optimized for high-volume update activity. This is a performance profile at variance with the “read mostly” star schema and dimensions. The “sweet spot” for the implementation of ERP systems are those firms that are fragmented (“diversified”) in form but which would benefit from centralization. Benefits include better integration and related processing efficiencies with trading partners who are also implementing ERP systems and improved vertical communications within the firm up and down the internal supply chain. Notice that business processes will have to be changed to accommodate the ERP system, not vice versa. See the discussion in the text box entitled Marriage Made in Heaven or Shot Gun Wedding?

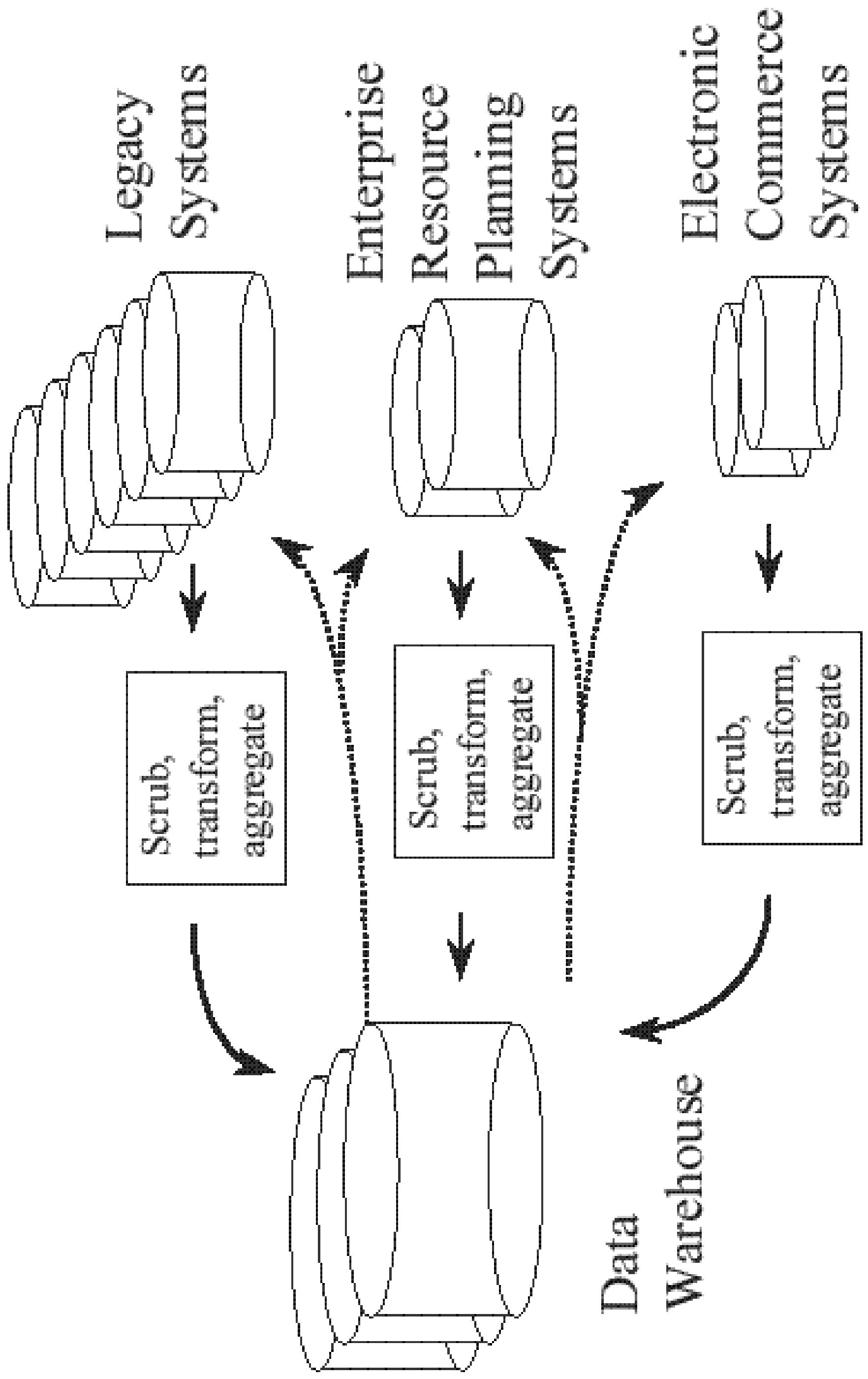

Legacy Systems
Enterprise Resource Planning Systems
Electronic Commerce Systems
Scrub, transform, aggregate
Scrub, transform, aggregate
Scrub, transform, aggregate
Data Warehouse

Figure 1.4
Data Warehouse Sources of Data

Marriage Made in Heaven or Shot Gun Wedding?

For example, the Coca-Cola Company and eleven of its major downstream supply chain partners—those are called *anchor bottlers* in the soft drink business—signed a contract with SAP[5] for a single license that covers a dozen or so separate companies. Traditionally, bottlers have a strong heritage of independent operations. As far as the details of business operations are concerned, that is changing. The mandating of a single set of business processes, dictated by the software, on all of the partners is expected to have a unifying result. Much of the value of this undertaking will be realized as Coke's North American data warehousing systems are expanded to accommodate the additional inputs. This will enable Coke and its bottlers to quickly assess the success of promotions or ads for particular brands in certain regions, down to the individual store. For example, the critical path for increasing sales lies through answering such questions as how well 2-liter bottles of Sprite or Diet Coke are moving in one market versus another, whether different sizes of containers would perform better, or whether select products on store shelves would benefit from different positioning. It is important to note that the data warehousing system is a central component, but not the only one, in a process of coordinating the sharing of information (about what's selling, where, and why) between a dozen separate firms unified by a brand, a transactional ERP product, and a data warehouse.

Upon first implementation, the direction in which the data flows is from transaction system to warehouse. This is the typical scenario as operational data is scrubbed and massaged to assist in decision support processes. However, a surprising result occurs as the unified representations of the dimensions built to support the warehouse "feedback" into the transactional system. It turns out that the representations of the customer, product, and other crucial enterprise dimensions in the data warehouse are consistent, coherent, and of a superior data quality than are those in the transactional system. This is most often the case with legacy systems that have data quality and integrity issues, but it is also the case with new ERP systems that may have different quality and integrity issues, precisely because they are getting the "kinks" worked out in the "college of hard knocks." This is depicted as the dotted line "feedback" in Figure 1.4: Data Warehouse Sources of Data. Such feedback does *not* make the data warehouse into a data hub, which is and remains a separate architecture and product space. The point of data warehousing remains decision support, but a decision is an idle wheel unless, at times, it moves relevant parts of the transactional systems, and that is what is depicted here.

For example, feedback from the data warehouse to transaction systems is a typical advanced application in data warehousing. The data warehouse actually becomes the "front-end" at one phase in the information supply chain. When the data warehouse is used for a forecasting application, orders can be generated to the transactional system. The idea of a forecast is precisely to prepare transactional systems to accommodate expected demand. This demand is invisible, as long as one remains at the level of elementary detail transactions. But when the warehouse forecasting application surfaces, a trend based on aggregations is visible, as long as one is viewing only elementary transactions detail. Orders can be generated from the warehouse to the transactional system to adjust inventory, to provide replenishment, or to take actions to allocate resources to accommodate the forecasted demand. This is sometimes called *sourcing* the forecast because that is what it does—provide a source to satisfy orders.

On the other hand, the "nightmare scenario" for those contemplating ERP implementation is the complementary situation. The firm is decentralized *and* has competitive advantages from such diversity of processes and approaches. Construction companies, investment banking firms, and technology firms whose work varies significantly from project to project fit such a profile. They benefit from local customizations and accommodations that do not necessarily scale to uniform processes at the total enterprise level. Such firms are poor candidates for standardization through ERP systems because they eliminate the differences that are the source of advantage.

A development occurring as this book goes to press—too recent for much experience to have been accumulated with the option—is the offering of data warehouse products from ERP vendors. Such an option might make sense if the only source of warehouse data is the ERP system itself. After all, no one knows an ERP system better than its own vendor. On the other hand, different competencies and methods are involved in building transactional and warehouse systems. The first are optimized for update, direct manipulation of single records, and high numbers of short transactions, the latter for "read mostly" access, browsing, and relatively smaller numbers of longer transactions. It makes sense to apply to the ERP vendor "warehouse solution" many of the distinctions and methods that will be developed in the following chapters on metadata, the information supply chain, OLAP, and aggregation. Does the product construct a cube on a special purpose server, does it have "reach through" back to underlying relational data, is it a star schema or a snowflake approach?

Finally, a third source of data for the warehouse is electronic commerce (EC) systems. Business-to-business electronic commerce systems include intra- and Internet-based systems. These build on the business imperative of electronic data interchange (EDI), cost reductions, and increased efficiencies in supply and value chain management. The business case for eliminating manual processing has always been a strong one. The coordination of commitments entailed in optimizing the distribution of products in manufacturing and retailing invites "action at a distance," the real power of commerce "over the wire." This case is augmented by pressures for just-in-time deliveries to manage and reduce inventories, for vendor-managed inventories (where the supplier has visibility to the customer's on-shelf inventory), and for disintermediation (eliminating the middleman). What the Internet does is insert a new group of "open" technologies on which to base action at a distance. It provides a sort of entry level "value-added network" (VAN) for the conduct of EC, thus lowering the bar to small and midsized companies. Business-to-business EC is what has experienced solid, low double-digit growth for years on the order of 20%.[6] However, it is consumer sales over the Internet that has generated the buzz in the trade and popular press outlets. To be sure, as the trust of the consumer grows—especially where branded products are concerned (FedEx, Amazon, Toys-R-Us, United Airlines)[7]—the bet is that the Internet as a channel will be perceived as being as reliable as the telephone. With good industrial strength security—once again, a big "if"—the Internet holds the promise of being even more reliable. Thus, TCP/IP is the new dial tone for deliveries to the busy professional. Capturing this data in useable form and forwarding it for analysis to the data warehouse, either directly or through intermediate transactional systems, is essential to have a complete picture of enterprise business systems at work. The real knowledge occurs when business managers are able to answer questions about what percentage of business is occurring in what channel—call center business, retail outlet, Internet sales—and what factors influence the mix and dynamics of each. This points in the direction of yet another advanced application—data cubes, aggregations, and dimensions based on who is clicking on what web pages and who is ordering from them.

Each of these sources of data—legacy, ERP, and EC systems—represents a different path or different channel for capturing inputs to the data warehouse. Relative to the data warehouse, they are systems functioning as information producers; the warehouse itself includes the information manager function; and the business analysts who use the warehouse, in turn, are the information consumers. The movements of data between these systems implies a work flow as a coordination of commitments. The commitments are business imperatives, such as developing customer relations, developing product brand, and optimizing the supply and value chains. The knowledge to understand and further these initiatives emerges from the interaction of enterprise systems in the crucible of invention of information technology application to the optimization of business processes.

DIMENSIONS

The one fundamental question names basic business drivers—customers, products, services, suppliers, locations, channels, periods of time within which events occur, and additional other entities significant to the business. When these business drivers are abstracted and represented in a relational database, they are called *dimensions*. Dimensions are what give meaning to facts and make them unique. As you can see, dimensions are really very close to what data modelers and database administrators call *entities*. Normal people would call them whatever we designate by means of the nouns in our language. They are the basic states of affairs—persons, places, things—of which the real world is composed, that is, the real world of business being represented in the database. Database professionals sometimes have strong feelings—bordering on religious intensity—about how many structures (tables and relations) are required to represent accurately the real world of the business without redundancy or risk of inconsistencies during update processing. The "right answer" is generally described as "normalized data," and when compromises are intentionally made to improve performance, that is "denormalization." However, it should be clear from the business drivers that, if one has a reasonable approximation to the structure, relations, and components of the business, one can practically and pragmatically use them to

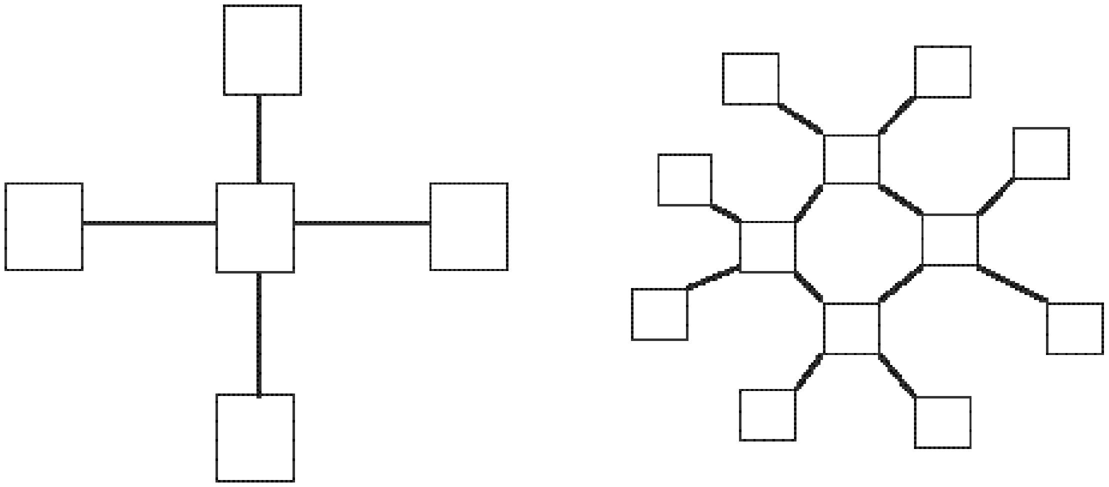


Figure 1.5
The Star Schema and the Snowflake

guide the decision-making activities of that business. Keep in mind throughout this discussion that agreement in detail on these fundamental business drivers—the dimensions—will go a long way toward avoiding misunderstanding. Such agreement is not to be taken for granted and often must be attained as a result of a struggle between professionals of good will willing to compromise with reality and one another.

Some information technology (IT) professionals—database administrators, in particular—are concerned about building a consistent and extendible model of their data. They do this (among many reasons) to validate that the system is logically coherent and workable, prior to incurring the effort and expense of implementing it in hardware, software, and applications. At a high level, this model consists of things—products, customers, locations (stores), and time periods)—and relations between them—customers "buy" or "take delivery of" or "use" products, etc. Naturally, based on these two basic components, the model is called an *entity-relationship* diagram (ER diagram). Of course, there is a lot more to an ER diagram than this, but as soon as one introduces the term *dimension*, a controversy occurs.

It is now a matter of controversy in the data warehousing industry whether a different kind of model, method of analysis, or even principles of thinking are required to deal with dimensional data, as opposed to classical ER diagrams. It is true that, as a visual presentation, the resulting diagram of a data warehouse data model has a significantly different visual appearance than does that presented by the ER diagram of a transactional system. In short, the resulting data models certainly appear to be different. The one looks a bit like a snowflake, elegant and complex. The other looks more like a star, a simple central body surrounded by multiple structures (see Figure 1.5: The Star Schema and the Snowflake).

The essential claim of this guide is that the methods and principles of relational model are valid and sound when applied to data warehouses. This may not be completely clear to the reader until the chapter on data warehouse technical design, but the executive summary must be attempted. Whether the data modeling exercise is called *dimensional* or *ER*, the rules are the same: Atomic data satisfies first normal form. Each attribute is dependent on a primary key. Each attribute depends on the key, the entire key, and nothing but the key. Deciding between multiple candidate keys requires appreciation of data semantics and functional dependencies. Independent attributes, whether constrained by particular business rules, should be broken out into separate tables to eliminate redundancy and to update anomalies. The same method yields different results because the purposes of transactional and decision support systems are different. So, the degree of abstraction and perspective on the analysis are different. This results in different boundaries to the entities that have been identified. Both entities and dimensions are abstractions of business objects that we encounter in the world of commerce and trade. The one is abstracted and refined to speed concise update processing. The other is abstracted and refined to speed access and inquiries. Nevertheless, considerable overlap exists. The fact table, as an entity, satisfies third normal form, but it is rather like an abstract data type (ADT), in that it represents a point in time where a customer interacts with (buys, takes delivery of, uses, etc.) a product. The supporting dimension structures, such as product, customer, calendar, promotion, or channel, are consciously denormalized into long, attribute-filled structures to improve performance. In particular, browsing performance is optimized in this way. Notice, however, that this denormalization is a defined and proven method of solving certain problems within the ER diagram and relational model under any interpretation. It is quite consistent with the ER method.

Issue: Dimensional Versus Entity-Relational Diagramming

Reasonable persons might disagree about whether it (dimensional data modeling) is useful in one context or another, but that disagreement occurs within the framework of the classic ER approach to data analysis and definition. In fact, if one looked at other kinds of systems, one might easily come up with other figures besides a snowflake and a star. Systems that store images of documents ("imaging" systems) tend to have a bulky back-end, due to the collection of binary large objects—the BLOBs. They resemble a hippopotamus. Others are shaped more like a giant squid, with tentacles in every direction. The shape is interesting but accidental. It is a useful mnemonic device but otherwise without significance. Mark down the imagined requirement for a different method of data analysis from relational modeling for data warehouses as one of the great nonissues of data warehouse development. The rigors of data modeling cannot be escaped that easily. If you already understand ER diagramming, don't sign up for yet another course in dimensional data modeling. The course will tell you when to break the rules of ER diagramming. But it will not really provide any fundamental new rules. Rather, study the questions, parameters, and concepts of your business and how they appear in the business of decision support.

Finally, dimensions provide the paths along which basic aggregation, roll up, or drill down of the data occur. This path lies along a hierarchy. Thus, customers are often grouped into geographic hierarchies—individual customer, section, district, region, or area (for example). Customer purchases or uses of relevant products at selected places and times are summarized into sections, then into districts, then into regions, and, finally, into areas. A high-level customer aggregate—say, at the region or area level in the hierarchy—summarizes a lot of information about the behavior of customers. Products often fall into the hierarchies with which we are familiar as consumers—universal product code (UPC), item, product, brand, or category. Alternative hierarchies can coexist and result in different "segmentations" of the market or ways of using the product. Products in the insurance, finance, or subscription services industries have their own characteristic groupings, highly customized by industry. The basic point being made here is to note that the paths along which "drilling" or "aggregation" occurs is laid out in advance by the contents of the dimensions.

THE DATA WAREHOUSE FACT

A customer buys a product at a certain location at a certain time. When the intersection of these four dimensions occurs, a sale is made. The point at which these dimensions intersect, providing an answer to the fundamental question, is a basic business event—a transaction. That sale is describable as amount of dollars received, number of items sold, weight of goods to be shipped, etc.—a quantity that is a continuous value that can be added to other sales similar in definition. A meaningful and *measurable* event of significance to the business occurs at the intersection point of these dimensions. The intersection of these dimensions—it is hard to visualize more than three at a time—provides us with a fundamental feature. We have now defined a *fact*.

A *fact* is a measurement captured from an event in the marketplace. It is the moment of value when the customer intersects with the product at a particular space and time. It is the raw material for knowledge—observations. This is a significant result. (See Figure 1.6: What Is a Data Warehouse Fact?)

A data warehouse *fact* is defined as an intersection of the dimensions constituting the basic entities of the business transaction. Naturally, a *transaction* is not yet a business relationship. Brand development, customer intimacy, sales trend analysis, and product trends are of the essence. These are what provide a breakthrough for the business. However, we at least have a start in understanding how the structures are designed and built that will provide the subsequent breakthrough.

At this juncture, the number of dimensions that can be drawn through the point of intersection is easily and usefully multiplied—store (location), vendor, promotion, ship-to location, sales personnel, and department. The dimensions tend to define the boundaries of the vertical industry segment. The dimensions are a catalog of entities and attributes of significance to the vertical market and industry in question. If you are

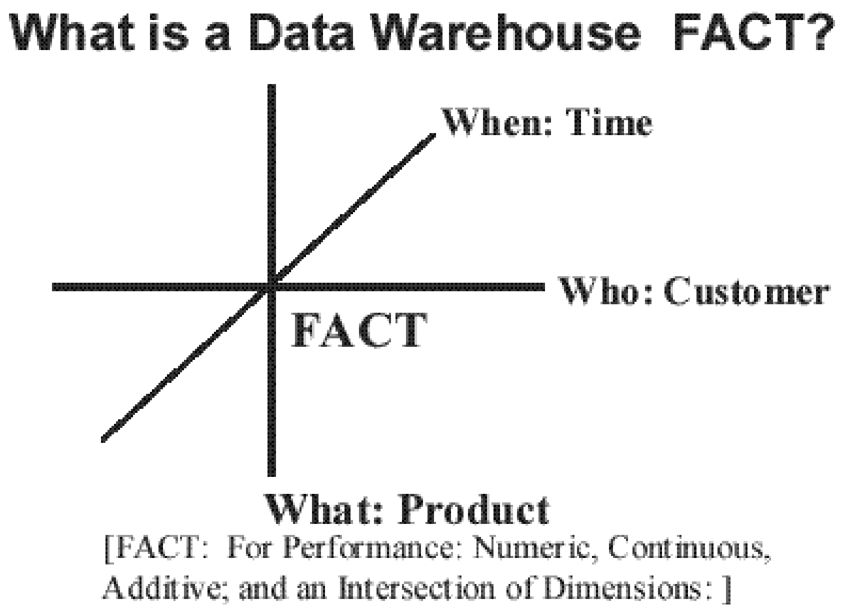

What is a Data Warehouse FACT?
When: Time
Who: Customer
FACT
What: Product
[FACT: For Performance: Numeric, Continuous, Additive; and an Intersection of Dimensions:]

Figure 1.6
What Is a Data Warehouse Fact?

selling seats in an airplane, frequent flier number is an important dimension. If you are selling rooms in a hotel, room type, service, and classification are important. If you are a managed care health maintenance organization (HMO), the diagnosis, procedure code, physician, lab tests, etc. are significant. The thing to keep in mind is that the logic and principles are fundamentally the same as those that apply to product and customer.

Describing a data warehouse fact occurs in terms of the key term, *granularity*. Granularity, as the label suggests, expresses how much detail is captured in the elementary fact. Are we dealing with a line item on an invoice? Or are we dealing with all customer purchases of a given product for a given day? Or week? How much detail is required to make decisions about business practices, strategies, and initiatives? Operational systems contain the most detailed data available, because that is where it is originally generated—think of the lines on the invoice. However, they usually do not attempt to store and manipulate three to five years of this data. The work-in-progress data stores of transactional systems usually encompass two to three months—a fiscal quarter. The rest gets archived to inexpensive media, such as sequential tape. Thus, the determination of the level of granularity—how elementary or composite are the facts—is an important consideration throughout the data warehouse life cycle. It is a critical performance factor, but it is a factor driven by the business questions that we want to answer, not by the technology. To be more precise, it is a factor driven by the business and constrained by the technology. Too large a granularity, and specific details are lost and much processing time is wasted trying to decompose aggregates;

too small a granularity, and the trees obscure the forest. Much valuable processing time is wasted building aggregates.

Along these lines, an issue of the first importance arises when a dimension gets restated due to corporate reorganization, change in business process required to support marketing or product design, or merger and acquisition. For example, *restating geography* means that the customers belonging to region *A* and *B* are now defined as belonging to regions *X* and *Y*. Indeed, there is some volatility in the geographic dimension, due the benefit of periodically realigning customers with marketing talent and distribution of service resources. In these days of the "virtual corporation," it is easier to rearrange hierarchies conceptually to accommodate business processes to which geography may be irrelevant. Because the dimensions are what make the aggregations of facts meaningful and unique, the redefinition of a geographic dimension implies that the aggregate has to be rebuilt. This is possible only if the data available is sufficiently granular. If the details have been "thrown away," once the aggregate has been built, the meaning of the aggregate has been irretrievably lost because the dimension is redefined. Customer "Lou" used to belong to region *A*, but now belongs to region *X*. It is no longer possible to compare an aggregate containing (using) region *A* at a point in time *prior* to the redefinition with one containing region *A* at a point in time *after* the redefinition. This is because they mean something different—region *A* used to contain data relating to customer Lou, but now does not. One must be able to break down all of the related and relevant aggregates into their respective details and add them up again to restate the meaning of the aggregate in *consistent* dimensional terms. It is sufficient to note the trade-off here between the work of carrying the granular, detailed data and the flexibility of being able to rebuild aggregates and facts if the definitions of dimensions are restated. This is indeed an advanced case—discussion of it is resumed in the section on reinterpreting the past in the chapter on data warehouse data quality.

The basic distinctions between dimensions and facts is now available. Therefore, it is appropriate to turn to design, admittedly at a high level, as befits a chapter on basic distinctions.

THE DATA WAREHOUSE MODEL OF THE BUSINESS: ALIGNMENT

The data model resulting from building a central fact structure with the smaller, supporting dimensional structures placed around the periphery resembles a star. The intersection of different dimensions to form a structure off of which, in effect, a fact table is "hung" actually looks like a star. The fact table in the center and the various dimensions are the many points of the star. Hence, the name given to this form of joining together the various dimensions is the "star schema." When a customer id, a product id, and a time period are used to determine which rows are selected from the fact table, this way of collecting the data is called the *star schema join* (see Figure 1.7: The Star Schema). In many ways, this is another name for the entire dimensional model. The really important and interesting thing to note is that the model includes and represents aspects of the market and the product supply chain, as well as the transactions driving the business. This is not an entirely new feature because, to be useful, any minimal data model requires a working representation of the customer, but it is a new emphasis. Rather than modeling only those features internal to the integrated firm, other relevant and useful aspects of the business environment are brought onto the radar, so to speak—up- and downstream suppliers, vendors, partners, channels, and customer- and product-sustaining activities of all kinds. Thus, the alignment of the structure of the data warehouse and fundamental aspects of the business are so logical, precise, and exact that one might suspect a preestablished harmony (see Figure 1.8: The Data Warehouse Model of the Business).

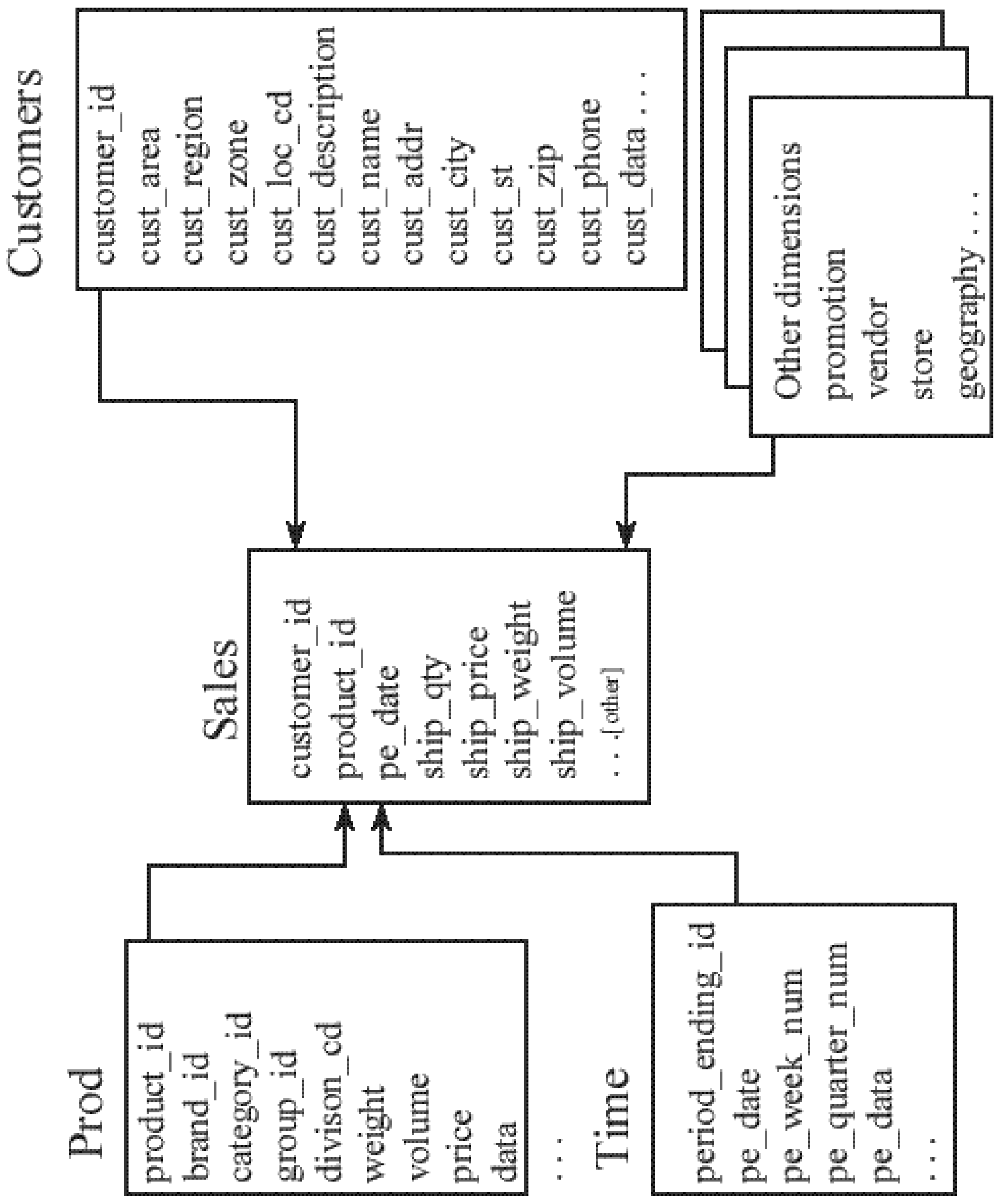


Figure 1.7
The Star Schema

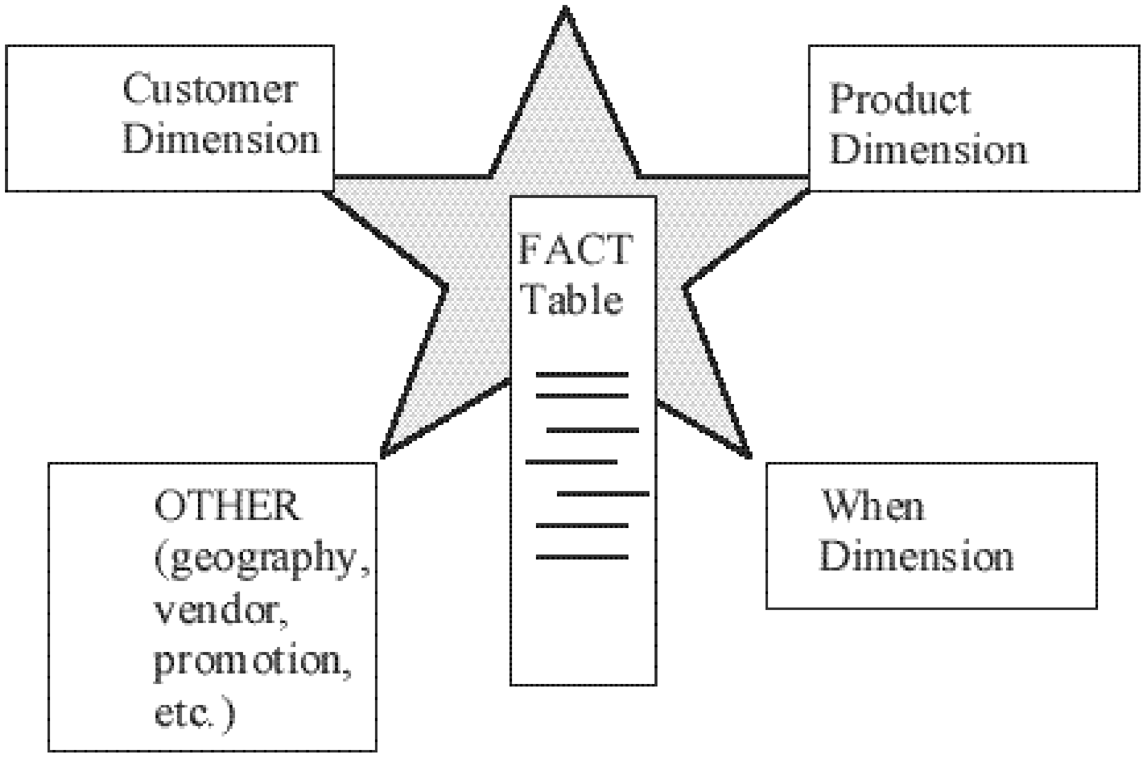


Figure 1.8
The Data Warehouse Model of the Business

THE DATA CUBE

First among these is the data cube. The c*ube* is defined as the available aggregation by basic business drivers, depicted as completed data structures suitable for inquiry by SQL or an other interface. The cube is the intersection of dimensions to provide a structure of facts of interest to the business. The classic cube is customer by product by time or place (see Figure 1.9: The Data Cube). If a customer, Lou, buys a Tasty brand granola bar on April 2, 1999 for one dollar, that transaction is given meaning as the intersection of customer, product (item), and date. The most important quantities associated with the transaction are the sale price (one dollar) and the item sold. These quantities form the granularity of the cube. The granularity of the cube is a function of the intersecting of the dimensions. Granularity is the specific level of detail that results when dimensions intersect—usually, a quantitative, continuous, value, able to be added. If an individual customer intersects with an individual product, the result is an elementary granularity or atomic transaction. Cubes are listed toward the top of the list of available *metadata.* Although not limited to cubes, metadata are defined as the available aggregations by basic business drivers, usually defined and maintained in a repository built on a relational catalog. Thus, cubes end up being a subset of metadata,

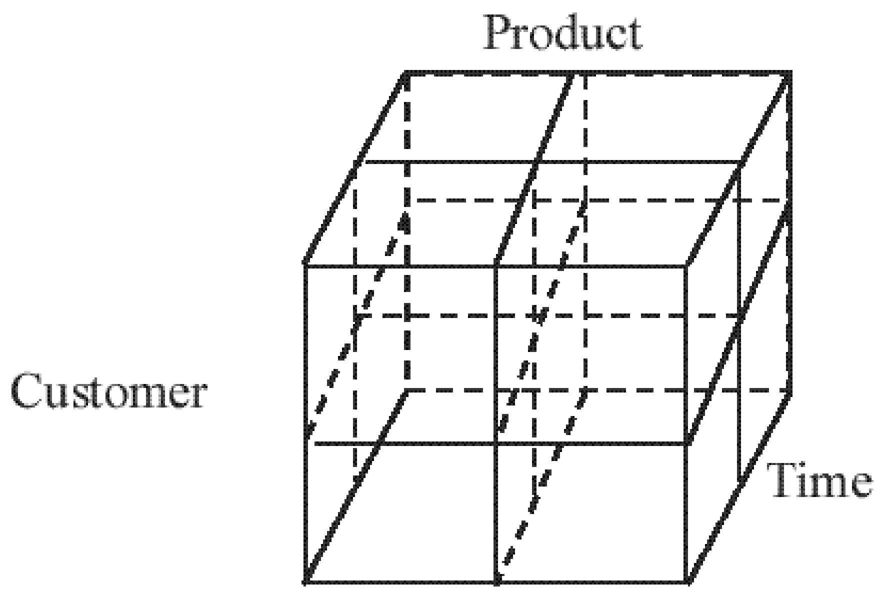


Figure 1.9
The Data Cube

which, however, is a more encompassing category to which a chapter in this book is devoted (see Chapter Thirteen on metadata and metaphor).

What is not easy to depict visually in Figures 1.7 and 1.9 by means of the star schema and the data cube is the trade-off between the two approaches. In general, the star schema join is a method and a result of using a relational database to represent the data by means of associations between multiple dimensions and facts. In general, cubes are contrasted with the star schema in comparing the methods and results of OLAP engines with relational databases. However, this is an inaccurate oversimplification, because cubes can be translated into star schemas, and vice versa. The real issue is trade-offs involving performance and efficiency. High-level aggregations are more efficiently stored as cubes, having been precalculated; alternative roll-ups across changing dimensions are more efficiently and flexibly performed by star schemas, based on available details. (Further details on the strengths and limitations of OLAP will be addressed in Chapter Fifteen.)

AGGREGATION

Having defined data cubes, we can define aggregates in terms of them. Aggregates are cubes that accumulate (summarize) basic transaction data, based on a meaningful intersection of nonelementary levels in the dimensional hierarchy. Customers are grouped into regions. Brands contain products. A single sale occurs as a customer buys a product. A region does not buy a brand. There (obviously) is no such elementary transaction, but a group of customers in a region buy a set of products contained in a brand. This abstraction—sales of brand *X* to customers in the southwest region—generates an entirely new level of emerging insight, a knowledge of the behavior of a brand and customers. An aggregate reports on the accumulation of brand by region. Here, the granularity is not an atomic (or basic or elementary) transaction, but rather an aggregation. As indicated, an aggregation is an accumulation or summing of additive values, based on a meaningful intersection of dimensions. The dimensions in questions are typically nonindividual and themselves describe a group, as in the cited example. The variety, number, and uses of aggregations are functions of the groupings of customers, products, and other dimensions into hierarchies. One can arbitrarily start combining customer by product, customer by subbrand, customer by brand, customer by product category, and region by product, region by subbrand, region by brand, region by product category, etc., up and down the available hierarchies. Next, cross-reference these and similar aggregates by time period—customer by product by week, customer by product by month, customer by product by quarter, region by brand by week, region by brand by month, region by brand by quarter, etc. This is useful only as an intellectual exercise to appreciate the multiplying diversity of aggregates. Other methods—to be examined in Chapter 14 on aggregation—will be required to proceed efficiently into this area.

DATA WAREHOUSE PROFESSIONAL ROLES

Roles are divided into three kinds. Roles include information producers, information consumers, and information managers. The administrator who collects information from a customer who is opening an account at a bank is a producer. A bank automatic teller machine (ATM) is an information producer. The thousands of detailed transactions are captured in data stores, logs, and electronic media of all kinds. Likewise, a point of sale terminal—a cash register—at a retail discount store is a data collector and information source. As shoppers and buyers of merchandise and services, we are all information producers. We leave literally hundreds of electronic traces per day on operational systems every time a credit or debit card is swiped at a gas pump or lunch counter. These transactions, in turn, trigger others—interbank transfers of money or names and addresses added to mailing and direct marketing phone lists. Automated data processing systems are an entire class of information producers. Once the data is captured, processes up and down the information supply chain are triggered to manipulate, transform, and report on the relevance and results of the processing. Every time we pick up the phone or other communication device, we leave traces for telephone billing systems to charge back to us or to telemarketers to try their latest promotion. This yields a vibrant image of the modern digital economy. In truth, insofar as these transactions find their way into warehouse data structures, we are all caught up in the data warehousing initiative, whether we know it or not. Thus, in the data warehouse world, other systems—legacy, ERP, and EC systems, the detailed transactions of the day-to-day operations—are the information producers. They are the source of data to be extracted, scrubbed, transformed and aggregated, and loaded into the data warehouse structures.

Information consumers include marketing specialists working on positioning or selling products and services. The users of automated systems are in the role of consumers of the information that the system is designed to deliver. Customer-focused firms contain legions of staff devoted to consuming all kinds of information about the customers to whom they are dedicated. Product performance staffs are consumers of data about the behavior of their products. Because of handoffs from one phase in the information supply chain to another, one person's information producer is another's information consumer. However, in general, in the data warehouse world business analysts, marketing and product specialists, and knowledge workers of all kinds are the main consumers of the data warehouse content. The thousand and one questions they pose about who is buying what and where and when are what the data warehouse is designed and constructed to answer. Answers to our questions, especially when they enable timely and useful business initiatives, are good examples of the benefits of knowledge.

The information managers—caretakers or custodians—are charged with lining up the questions posed by the knowledge workers and the information consumers, with the data captured in the data warehouse from the information producers. It is this lining up—this "alignment"—that presents the challenge faced by the traditional information technology roles. This alignment is what makes answering the questions and what makes the data warehouse an information product—a form of knowledge—in itself.

To be sure, the traditional IT roles—designer, developer, network administrator, security administrator, storage administrator, and data and database administrator—are still valid. For example, the distinction between data administration and database administration is captured by assigning tasks emphasizing logic, design, and modeling to the data administrator; whereas, the database administrator is responsible for physical implementation, backup, and maintenance of the database proper. However, both roles are now informed by the requirement not only to do software engineering and integration but to applying these skills to knowledge management and integration.

From the designer's point of view, this means that requirements are not complete until the questions are answered. What would we like to know about the behavior of the customer (or product, service, etc.) and how can we use the available data to address the answer? Here, *knowledge engineering* means constructing an answer that traces the data forming the answer back to the question and doing so in context.

From the development perspective, *knowledge engineering* means using desktop tools, database-stored procedures, and application code to represent the answer to the question in such a way that it is understandable, meaningful, and actionable in a timely way.

The more accurate and useful the content stored in the data warehouse, the more important that it be secured against unauthorized access, whether accidental or malicious. The security administrator is charged with leading the effort to secure the enterprise's resources. It is important that the knowledge be made accessible to those who need it, regardless of time or place. Here is the case for distributed and WANs. The network administrator provides the leadership.

From the DBA's (database administrator) perspective, data quality is critical path. If the data possesses the features that mark it as a quality product—if it is accurate, unambiguous, credible, and has other attributes, to be discussed in the Chapter Six on data warehouse data quality—it deserves the respect and dignity that we accord to information when we call it *knowledge*. Another way of saying the same thing, with a significant difference in emphasis, is that the DBA is the "lightning rod" for data integrity, the first mark of data quality, even if the end-user is the ultimate owner of the data and ultimate arbiter of its validity. In any real world situation, as soon as the DBA team provides "access to data" (makes it available within the framework of a basically sound warehouse design), the phone is likely to ring with an opportunity to improve data quality. Something is incomplete, and a "bootstrap" operation is initiated, within which data quality is pursued as an iterative goal.

THE DATA WAREHOUSE PROCESS MODEL

The data warehouse process model can now be defined. It is defined as that which transforms operational into decision support data. This includes the processes of data scrubbing, denormalization, aggregation, partitioning, parallel access, and asking business questions grounded in the data to generate the business intelligence (knowledge) that makes a difference in reaching a firm's business objectives and delivering competitive advantage. This process will be the subject of Chapter Twelve: Data Warehouse Operations: The Information Supply Chain.

The information supply chain makes data warehousing seem like a miracle in many ways. The input is dirty data and, after a serious amount of data scrubbing, out comes what we call *knowledge*. Although an oversimplification, that is what happens in the information supply chain. The data warehouse itself is an information product of the highest degree, which, when used to answer questions about the business, generates knowledge in the full sense of the term.

SUMMARY . . . .

Data warehousing is a system architecture, not a software product or application. Architecture refers to first principles of representation, design, and information technology processing results. The essential minimal working set of data warehousing components includes the database engine, application (analytic) server, connectivity, and presentation layer. When made the target of a fundamental inquiry, data warehousing architecture turns out to be a variation on work flow. Here “work flow” is understood as a coordination of commitments, not an assembly line. Commitment is not usually regarded as a feature of architecture. But it is. Commitment to the business imperative to know the customer, build (develop) the brand, know who is buying what.

Data warehousing addresses *the* one fundamental question: Who is buying (using) what product (service) and when and where are they doing so? This immediately implies the first distinction between transactional and decision support systems. Data warehousing sources of data are itemized as transactional (legacy) systems, ERP systems, and e-commerce systems. Basic business drivers get represented in the data warehousing database as structures such as customers, products, services, suppliers, locations, calendars, etc. The point at which dimensions intersect – a customer buys a product at a certain store on a certain day (say) – is a quantitative, measurable event (transaction) that can be captured as a data warehouse fact. The definition of a fact and its granularity is a significant result. Other basic distinctions are defined including, the data cube, aggregations, the star schema (the data warehousing model of the business), data warehousing professional roles, and the data warehousing process model.

[1] Brown, G. Spencer. ***The Laws of Form***, New York, NY: The Julian Press, Inc. 1972, p. 1.

[2] The word *architecture* contains the ancient Greek roots *arche,* meaning “ruling principle” or “first principle,” and *techne*, meaning “method,” from which we get *technology*. Thus, *archeology* is the study of first (or old) things, and *archetype* is another word for first patterns or basic designs.

[3] A graphic of the full-blown Zachman framework document is available for noncommercial use as a free download in TIF format off of Zachman’s web page at www.ZIFA.com, courtesy of the Zachman Institute for Framework Analysis.

[4] The reader will note the challenge to the author (and so to the reader) of defining many interrelated terms at once. For those instances where a subsequent chapter is devoted to a term or subject or when the reader finds that the terms come too fast, an extensive glossary is provided. Please make good use of it.

[5] As reported by Bob Violino in Extended Enterprise, ***Information Week***, March 22, 1999. SAP stands for Systems Applications & Products in Data Processing. See www.sap.com.

[6] The relevant technologies will be detailed below. Here, the emphasis is on understanding interacting enterprise business systems. In my opinion, the best concise introduction to the subject of EC is Peter G. W. Keen and Craigg Ballance***, On-Line Profits: A Manager’s Guide to Electronic Commerce***, 1997, which cuts through the hype of growth doubling every year and documents good, solid year-to-year gains. (See also www.PeterKeen.com.)

[7] Another work by P. Keen, T. Terragrossa, and W. Mougayar makes an important distinction between brands born on the web—Amazon, Virtual Vineyards, Auto-by-Tel—and brands successfully extended to the web—FedEx, UAL, Cisco Systems—see ***The Business Internet and Intranets*** (Harvard Business School Press, 1998).