CHAPTER 1

# The Origins of Business Intelligence



The origins of business intelligence may be traced to the first data-processing applications such as accounts payable and receivable. These applications ran on sequential technology (such as magnetic and paper tapes). Using sequential media for storage meant the entire file had to be accessed, even if only a fraction of the file was needed. Oxide often stripped off of magnetic tapes, and entire files were lost. These issues led to the need for a new way to analyze information.

This chapter discusses how enterprises tackled the challenges of extracting data from online transaction-processing systems, dealing with poor data quality, and structuring data. This discussion describes what a data warehouse is from its data model, table structure, granularity, and support of historical data and how it fits into a broader intellectual concept called the *corporate information factory* (CIF). While the CIF was evolving, so was SAP, until finally the two converged.

## Evolution of Information Processing

The computer profession is historically immature. Other professions have been around for millennia. In caves in Chile, bones have been found showing that humankind practiced medicine (in at least a crude form) as long as 10,000 years

ago. In Rome, walls and streets are used today that engineers built before the time of Christ. In Egypt, hieroglyphs on the walls of the tombs indicate that an accountant declared that wheat was owed the Pharaoh 3,000 years ago.

On the contrary, the computer profession has been around a measly half a century or so, depending on when you start counting. So, comparing the maturity of the IT profession is a mismatch; IT professionals are infants in comparison to their other professional brethren.

## Data-Storage Advancements

The early beginnings of the computer profession featured wired boards and paper tape. These crude forms of programming allowed the programmer to route and reroute data as it passed from paper tape (or punched cards) to reports. To even the most ardent technologists, early boards and paper tape were crude. Soon came computer languages such as Assembler, Fortran, and Cobol. These languages allowed people to perform complex logic, something not possible with early forms of computing.

And with early forms of computing came magnetic tape. Magnetic tape could store massive amounts of data. Also, magnetic tape was reusable. Once someone wrote on a tape, the tape could be rewritten. Magnetic tape was a leap ahead from punched cards and paper tapes. And with this leap ahead came programs and systems, unleashing a flood of computing.

But magnetic tape was not without its limitations. It had to be read sequentially. With magnetic tape, you had to access 100 percent of the data to find the five percent you really wanted. Magnetic tape had the nasty habit of shredding. When shredding occurred, all the oxide on the tape came off, and with the stripping of the oxide, data was permanently lost. And, as a last issue, magnetic tape required a significant amount of manual manipulation.

Nevertheless, with magnetic tape, the gates of commercial computing opened.

But around the corner was disk storage. *Disk storage* was data stored on a spinning drum read by read/write arms. With the disk eternally spinning, data was, for all purposes, directly accessible. No longer was it necessary to read all the data to get five percent of what was needed. With disk storage, data could be accessed directly.

With disk storage came the notion of a database. A *database* was data stored on disk that could serve multiple purposes. With older magnetic files, data was stored in a *master file*, which essentially had a single purpose — to satisfy the needs of the application for which the master file was built. But with a database and disk storage, it became possible to build databases used in multiple applications.

## Transaction Processing Dominates

Soon, applications that centered on the database sprung up everywhere. But a new usage of disk storage soon became a possibility: the advent of *online transaction processing* (OLTP). With online transaction processing came possibilities never before possible. Prior to OLTP, computation was done in terms of reports, batched transactions, and overnight processing. The length of time it took to conclude a transaction limited the usage of the computer in the business. In those days, accounts payable, accounts receivable, and human resources were the primary applications that a shop had automated. But with online transaction processing, it became possible to use the computer in ways intimately tied to the business.

With OLTP (where there is consistent two- to three-second response time), applications such as bank-teller processing, airline-reservation processing, insurance-claims processing, telecommunications operator assistance, and so forth became the norm. For the first time, the computer became one of the cornerstones of business. When the computer was interrupted, of course, business immediately felt the stoppage.

With online systems, the computer became an essential part of business. And businesses everywhere began to use the computer in ways never before imagined.

In fact, applications — batch and online — became so pervasive that soon they were everywhere. And with this explosive growth in applications came some unforeseen and unanticipated problems. Soon the problem of *unintegrated data* reared its ugly head. Unintegrated data occurred because with so many applications, the same unit of data was found in many places. The problem was that, in each place, the unit of data had a different value. In some ways, this was worse than having no data at all. The organization was confused, the customers were confused, and decisions were based on factors other than data and information — a dangerous way to make decisions.

The applications were known as "spider web systems." The applications and the data inside them were tied together in a structure reminiscent of a spider web.

In many ways, the unintegrated data found in the spider-web environment was more insidious than other challenges that the IT community had met. Prior to the spider-web environment, when a problem occurred, there was always a new technological advance to bail everyone out. But no such easy technological fix existed when it came to solving the spider-web problem. For the first time, an architectural solution was needed.

## Extract Files Appear

The first reaction to the challenge of not having corporate data was to create an *extract file*. An extract file would be created by a database from one application and shipped to another application, so it seemed that data could be shared and corporate data could be created. Extracts became very popular, and soon there were a lot of them. Every new extraction exacerbated the problems of the spider web. Adding extractions made matters worse, not better. The problems of the spider web included the following:

- **Data integrity** — The same element of data appeared in many places. In one place, the element had a value of 25. In another place, the element had a value of 67. In still another place, the element had a value of 135. No one really knew what the right value was.
- **Data redundancy** — The sheer redundancy of data was enormous. The same data was shuffled from one place to the next; the burden of massive amounts of data being repeated over and over began to add up to significant amounts of storage and processing power.
- **Timeliness of data** — While shuffled around the system, data was aging. In one day, the value of a unit of data may change five or six times. The extract processing simply was not capable of keeping up with the speed with which data changed.
- **Multiple *silo* of data were created** — A *silo system* is not easily integrated with other systems. The use of the term comes from the cylindrical towers found in Middle America, where corn or feed is stored in an air-tight manner. Many versions of the same data exist in various silos, which violates the idea of a single "truth." Silos are sometimes also referred to as *data islands* or *stovepipe applications*. Each silo was its own operating domain with no coordination or integration with outside silos. One part of the organization found itself making decisions contrary to the interest of other parts of the organization.
- **The extract processing froze an already moribund system** — Online transaction applications were difficult to change in any case. But wrapping lines of extraction around the online applications glued those applications into a permanent position.
- **Data became much more inaccessible** — The extract processing placed coordination requirements on the environment, which ensured that accurate data was impossible to obtain and so forth.

Of particular interest is the lack of historical data. Online applications value current data. How much is a bank account balance right now? Where

is a shipment right now? What is the status of an insurance claim right now? Online applications optimize the "right now" aspect of information processing. As soon as data became dated, it was discarded. Lots of historical data clogged the arteries of efficient online processing. Therefore, online data and processing required that older data be jettisoned as soon as possible.

But there is real value in historical data. With historical data, organizations can start to see the forest *and* the trees. With historical data, organizations can start to understand their customer base, because customers are creatures of habit.

Because there was no corporate integrated data or historical data, data was difficult to access. Even if accessed, data was not trustworthy, so it is no wonder organizations began to grow frustrated with an inability to find and process information. Department after department would say, "I know the data is somewhere in my corporation; if I could only get at it."

The frustrations of the end user with data locked in the spider-web environment resulted in the realization that there were different kinds of data. There was an essential difference between *operational data* and *informational data*. Table 1-1 outlines those differences.

**Table 1-1** Characteristics of Operational versus Data Warehouse Systems

| OPERATIONAL | DATA WAREHOUSE/DSS |
|---|---|
| Modest amount of data | Immodest amount of data |
| Can be updated | Snapshot records; no updates allowed |
| Accurate up to the second | Timestamp on each record |
| Used for clerical purposes | Used by management and analysts |
| Built based on requirements | Built from data model |
| Supports small uniform transactions | Supports mixed workload |
| Yields two- to three-second response time | Yields 30- to 24-hour response time |
| Data designed for optimal storage | Data designed for optimal access |
| Very current data | Mainly historical data |
| Data is application oriented | Data is integrated |
| Data designed around functional usage | Data designed around subject areas |
| Referential integrity is useful | Referential integrity is not useful |
| High availability is normal | High availability is nice to have |

A fundamental split exists between *operational information* and *informational information*. Operational information is used to support the daily operations of a business. Informational information is commonly called *decision support system* (DSS) information. The foundation for DSS processing became an architectural structure known as the data warehouse. A *data warehouse* is a place physically distinct from the online operational application. The following sections describe the data warehouse and how it enables analytic information.

## The Data Warehouse Is Conceived

The needed solution was a split in database types. Prior to this point, it had been widely held that a database was a place for all processing against the data. But the spider-web problem was so difficult that a new approach to understanding what a database should be was needed. Thus, the data warehouse was born.

And indeed there are many more significant differences between the data warehouse and an operational database.

The advent of a new database type sent the database theoreticians of day into a tailspin. It was once heresy to suggest that there should be anything but one database type for all purposes. But the database theoreticians of the day got over it.

## What Is Data Warehousing?

Since the beginning of the movement toward data warehousing, data warehouses have been defined as:

- **Subject oriented** — Data is organized around a major object or process of an organization. Classic examples include subject-area databases for customer, material, vendor, and transaction.
- **Integrated** — Data from various subject areas should be rationalized with one another.
- **Nonvolatile** — Data in a data warehouse is not updated. Once a record is properly placed in the warehouse, it is not subject to change. This contrasts with a record of data in an online environment, which is indeed very much subject to change.
- **Time variant** — A record is accurate only as of some moment in time. In some cases, the moment in time is a single moment. In other cases, it is a span of time. But in any case, the values of data found in a data warehouse are accurate and relevant only to some moment in time.

- **Decision making** — Data warehouses were created for the purpose of management decisions.

In addition, the data warehouse provides the following:

- Detailed or granular data
- Integrated data
- Historical data
- Easy-access data

The data warehouse is at the center of the business-intelligence environment. The data warehouse represents the single version of truth for the corporation and holds data at a granular level. In addition, the data warehouse contains a robust amount of historical data. The need for a data warehouse is as true within the confines of SAP as it is outside SAP. And the elements of a data warehouse are as valid for SAP as for the non-SAP environment.

The data warehouse emerges from these requirements and supports the process of moving data from source systems and transforming and cleansing data so that it may be stored in an integrated data model at an atomic level of granularity. Many factors influence the design of a data warehouse and the structure in which data records are stored. The following sections discuss some of these factors.

## The Data Model

The design of the data warehouse begins with a *data model*. At the highest level, the data model is known as an *entity relationship diagram* (ERD). The ERD represents the abstraction of the granular data found in the data warehouse. *Granular data* refers to very low-level detailed records. Note that for the purposes of data warehouse design, the ERD represents only granular data, not derived data. *Derived data* is created from other data elements. This distinction is important because it greatly limits the size and complexity of the data model. There are, of course, other data models outside the data warehouse environment that do attempt to take into account derived data and atomic data. Granular data is discussed more a bit later in this chapter.

The ERD consists of entities and relationships. Each entity represents a major subject area of the corporation. Typical subject areas are customer, product, transaction, and vendor. Each entity is further defined at a lower level of data modeling called the *data item set* (DIS). The DIS specifies a lower level of detail than the entity does, encompassing such things as keys and attributes, as well as the structure of those things. The DIS is further broken down into a low level of design called the *physical design*. At the physical level of design, the physical characteristics of the data are created.

The data warehouse is now specified and defined to the database management system (DBMS) that will house it. Other physical aspects of database design (such as partitioning, loading, indexing, storage media, and time-stamping) are determined here as well. Figure 1-1 shows the design of the data warehouse from the different components of the data model.

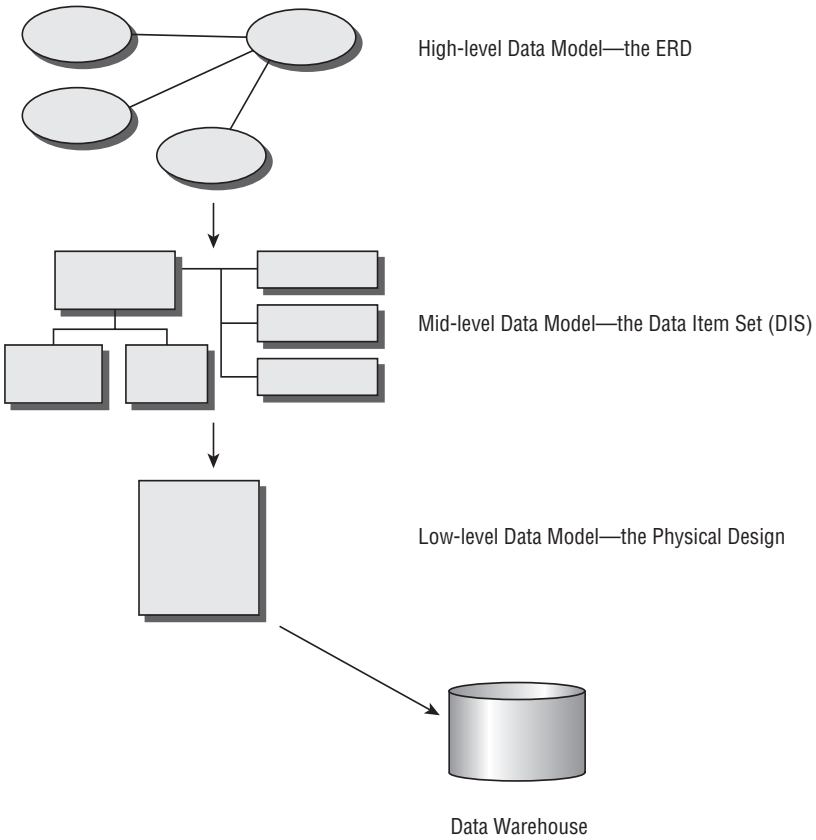


**Figure 1-1** The data warehouse is designed from the data model

## Different Physical Tables

The data warehouse is made up of interrelated tables or physical databases. Within the data warehouse, different physical tables represent different subject areas or even subsets of subject areas. One table relates to another by means of a shared-key or foreign-key relationship. The data warehouse typically has several areas, including the following:

- Customer
- Product

- Shipment
- Vendor
- Order

Each subject area resides on a separate physical table or database. Collectively, the different tables along with their relationships form a data warehouse.

## Integration and Transformation Processing

One of the most important and most difficult aspects of data warehouse development and population is the movement and conversion of data from the operational/legacy source environment. It is estimated that for the first iteration of development, at least 75 percent of the resources required for development will be expended here. During extraction, data is pulled from the legacy environment and moved into the data warehouse environment. This data is pulled from a variety of sources, such as mainframe order-entry systems, proprietary shop flow-control systems, and custom-built payroll systems.

But data is not merely moved from the legacy environment to the data warehouse environment. Instead, data undergoes a thorough transformation as it is moved, including:

- Converting data into a common format
- Reformatting data
- Realigning encoded values
- Restructuring data
- Assigning default values
- Summarizing
- Resequencing
- Converting keys
- Converting from one DBMS to another
- Converting from one operating system to another
- Converting from one hardware architecture to another
- Merging different record types
- Creating metadata that describes the activities of conversion
- Editing data
- Adding a timestamp

## Metadata

One of the essential aspects of the data warehouse is metadata. *Metadata* is information about the contents of what has come to be termed the corporate information factory (CIF). Every application has its own metadata, which is distributed across the entire landscape of architectural components. Metadata has two functions:

- To describe data found in the architectural component
- To exchange metadata with other components

Metadata in the data warehouse plays several roles. One role is describing what data resides where for normal usage. It also acts as a coordinator between different services from extract/transfer/load (ETL) software to information access. (ETL is discussed in more detail later in this chapter in the section titled "Extraction Transformation Loading.") The services of the architecture have very different foundations and functions. Some serve under one DBMS, others under another DBMS. Some services operate under one type of multidimensional technology, and other services operate under other multidimensional technologies. And each service has a very different function. For the services to operate in unison, there must be coordination from one service to the next. Coordination is achieved through metadata being passed from one architectural layer to another.

There are distinctly different kinds of metadata, including the following:

- **Technical metadata** — This describes the structure and content of the different types of data. This type of data has been housed in data dictionaries and repositories for a long time.
- **Operating metadata** — This represents the metrics generated by the day-to-day operation of the data warehouse. Metrics such as records passed from one software component to another, length of operation of a program, number of records in a database, and so forth make up operating metadata.
- **Business metadata** — This is couched in terms that the businessperson understands. Business definitions, business formulae, and business conditions all make up business metadata.

All three types of metadata are needed for controlling the operation of a data warehouse.

One concern is the metadata's integrity. To maintain control and believability of metadata when it is distributed across many different components, a certain protocol is necessary. To maintain integrity of metadata across a distributed environment, each unit of metadata must be unique and have one owner. The owner of the metadata is the only person or organization that has the right to

update, create, and delete a unit of metadata. Everyone else becomes a sharer of the metadata. As metadata is passed from one node to the next, a careful track of ownership must be kept.

## Granular Data

Data found in a data warehouse is very *granular*. This means that data is placed in the data warehouse at a very low level of detail. Data may then be reshaped by an application so that it can be viewed in a distinct manner.

Sometimes called the *atomic data* of the corporation, granular data makes up the "single version of truth" that is at the basis of reconciliation for informational processing. Having granular data at the core of the data warehouse provides many benefits. A primary advantage is that the same data can be viewed in different ways. Figure 1-2 shows that marketing looks at data one way, sales looks at it another way, and finance yet another way. But all three departments have a single source of reconcilability.

Usually, each grain of information in the data warehouse represents some finite unit of measure or business activity for the corporation. For example, a grain of information might represent details of the following:

- **A sale** — The amount, the date, the item sold, the location of the sale, or the customer
- **An order** — The date of the order, the product ordered, or the amount of the order
- **A telephone call** — The time of the call, the length of the call, the calling party, or the person called
- **A delivery of a product** — The date of the delivery, the location of the delivery, or the person making the delivery

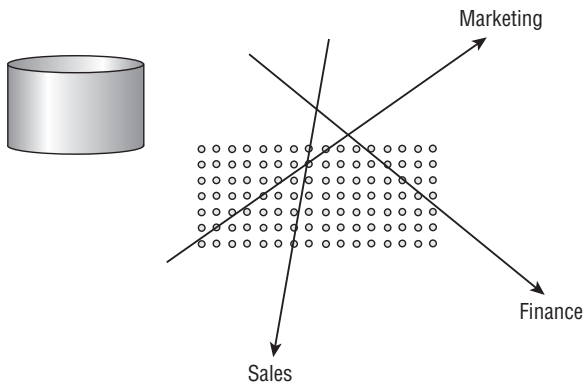


**Figure 1-2** Granular data allows the same data to be examined in different ways

Each grain of information can be combined with other grains to provide a different perspective of data.

In addition to allowing data to be viewed differently by different parties, another benefit is that granular data may lie in wait in the data warehouse for unknown and future requirements. Then, when a requirement becomes known, granular data can be shaped immediately to suit the new requirements. There is no need to go to the operational/legacy environment and pull data out. This means that the data warehouse puts the corporation in a proactive rather than reactive position for new needs for information.

## Historical Data

One of the most important characteristics of a data warehouse is that it contains a robust amount of *historical data*. Figure 1-3 shows a data warehouse that contains five years of history. Such an amount of history is typical. However, some data warehouses may contain even more historical data, and other data warehouses may contain less data, depending on the business needs of the corporation.

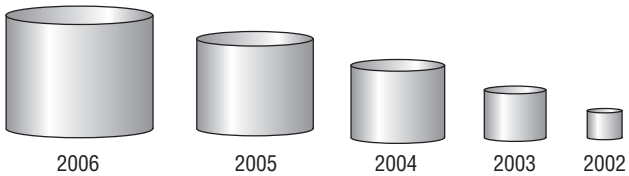


**Figure 1-3** The data warehouse contains robust amounts of historical data

Although historical data has many applications, perhaps the most potent is the ability to step backward in time and perform what-if analysis. Doing so allows you to gain insights that cannot be achieved any other way.

## Timestamping

The units of data stored inside the data warehouse are *timestamped* so that each unit of data has some element of time associated with the record. The timestamping of data-warehouse data signifies that the unit of data is accurate as of the timestamp.

In general, there are two ways that a record is stored in the data warehouse: discretely or continuously (see Figure 1-4). In a *discrete record*, there is one instant in time for which the record is accurate. In a *continuous record*, there is a span of time for which the record is accurate. These records form a larger definition of information over time.

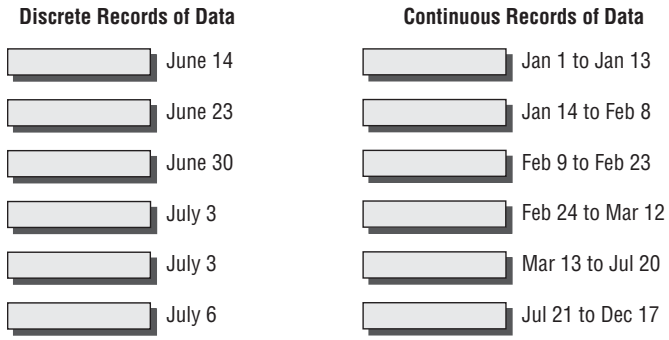


**Figure 1-4** Timestamped records are either continuous or discrete

Usually, discrete records are used for a large number of fast-changing variables. Continuous timestamps are used for a small number of variables that change slowly and for which there is value in knowing information over time.

## Data Relationships

The different types of data found in the data warehouse relate to each other by means of *foreign keys* pointing to *actual keys*. For example, suppose customer ABC places an order. There would be a customer record for customer ABC, as well as a separate order record for the order. The order record, in its body, would have a foreign-key reference to customer ABC.

Data relationships found in the data warehouse are special in that they are delimited by time. When a relationship is indicated in the data warehouse, the relationship is intended to be valid only for the moment in time indicated by the timestamps found on participating records. This interpretation of a relationship is quite different from that found in the online environment. Online environments enforce *referential integrity* (where the values of one data set are dependent on the existence of a value or values in another data set).

## Generic Data versus Specific Data

One design issue that arises in every data warehouse is how to account for generic data and specific data at the same time. *Generic data* applies to all instances of a subject area. *Specific data* applies only to certain occurrences of a subject area.

The generic database stores customer information along with related tables, including a wholesale customer table, a European customer table, a long-term customer table, and a preferred customer table. Each of the outlying tables contains information specific to the class of tables that meet the criteria. For example, a preferred wholesale customer would have data in the generic customer table, in the preferred customer table, and in the wholesale customer table.

In such a manner, data of different types can be represented efficiently in a data warehouse.

## Data Quality

*Data quality* is an important issue for the data warehouse environment. As shown in Figure 1-5, data quality is addressed in three places:

1. At the point of data entry to the legacy/operational environment
2. At the point of ETL processing
3. Once the data resides inside the data warehouse itself

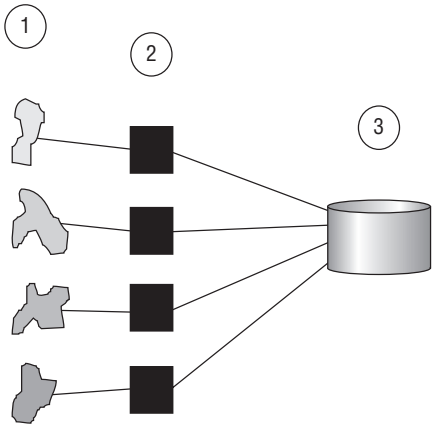


**Figure 1-5** Base data for all customers is kept in one table; specific data for different types of customers is kept in separate, unique tables

For example, raw data entry is addressed inside the legacy/operational environment. The problem is that the budget for doing tasks such as maintenance here has long gone away. In addition, no shop is eager to go poking around old, fragile, undocumented legacy applications, lest something untoward and unexpected happens. Therefore, not much data-quality activity occurs in the legacy/operational environment. The adoption of ERP systems has greatly improved data quality at the time of data entry.

Most data-quality activity occurs at the moment of ETL, which does not require that older applications be manipulated or touched in any way. Data that comes out of the legacy application can be isolated. And data coming from different applications can be integrated. Data is in transit in any case, so this becomes an ideal place in which to examine and audit it and to make changes if needed.

The third place where data quality can be applied is once the data has arrived in the data warehouse. Over time, data changes, and what was accurate and proper one year is not accurate and proper the next. So even if the data is loaded perfectly into the data warehouse, there still is a need for periodic adjustment of data based on changes in business conditions that have occurred over time.

## Volumes of Data

The volume of data grows beyond any expectations in a data warehouse. Once terabyte (about 1,000 gigabytes or GB) data warehouses were a dream; today they are a reality. In fact, it is not unheard of to build petabyte (about 1 million GB) data warehouses.

As data volumes grow large, approaches to their management change. One of the most important characteristics of a data warehouse's growth is the appearance of *dormant data* that just sits there taking up space and costing money. When the data warehouse was small, all or nearly all of the data that resided in it was used. But as the data warehouse grows large, increasingly large amounts of data reside in the warehouse in an unused state.

When a data warehouse is around the 100 GB range, there may be only 10 to 20 percent dormant data. But as a data warehouse approaches a terabyte, it is not unusual for the dormancy ratio to increase to 50 to 75 percent. And, as a data warehouse goes beyond several terabytes, the amount of dormant data frequently approaches 90 to 99 percent.

It is wasteful for a corporation to continue to increase the size of a data warehouse when the proportion of dormant data increases as well. In addition, increasing the size of a data warehouse when there is much dormant data grossly hurts performance.

## Removing Dormant Data

Dormant data must be periodically removed from disk storage and placed on another media. Active data is placed on disk storage and is managed and accessed in a normal manner, while inactive data is placed in a physically separate facility, sometimes called *alternate storage* or *near-line storage*. (Near-line storage is discussed in detail later in this chapter.) This causes the cost of the data warehouse to drop dramatically and the speed with which data can be accessed to increase.

To make the marriage between dormant data and actively used data residing on disk work well, a technology called a *cross-media storage manager* (CMSM) may be utilized. The CMSM sits between disk storage and alternate storage and manages the traffic between the two environments so that the end user is presented with a seamless view of the data residing in the data warehouse.

## Architected Solutions

The data warehouse concept was just the beginning of the split between operational and informational databases. Soon there were all sorts of other database types. There were personal computer databases. There were Online Analytic Processing (OLAP) multidimensional databases. There were customer databases gleaned from national marketing and sales efforts. There were statistical databases. And with these new database types came all sorts of other applications and uses of data. There were *data marts*, which were statistical databases such as *exploration* and *data mining warehouses*. There was a hybrid structure known as an *operational data store* (ODS). There were adaptive data marts. There were change data capture files. There were Web databases full of click-stream data.

**NOTE Data marts, ODS, and exploration warehouses are all discussed in detail later in this chapter.**

In a few short years, there was a bonanza of data, architecture types, and new and innovative applications. And with this bonanza came new users — users who in an earlier day and age had been neglected and whose information needs had not been served. There became a real need for tying together all of these different forms of data. Into the arena came the corporate information factory (CIF).

### Corporate Information Factory

The CIF was an architecture that weaved together the many different architectural components. The CIF accounted for the many information needs of the organization. The CIF is widely used by vendors and corporate IT shops alike to describe how the different components of architecture fit together. Figure 1-6 shows the progression from the earliest technologies to the CIF.

The CIF is a long-term architectural blueprint for business intelligence. Like any good blueprint for the future, there will be features not be needed now but perhaps needed at some point in the future. Whether or not you use all of the features, there is a certain comfort in knowing that those features are integrated should you need them. The CIF is not static. Every three or four years, different components are added to the CIF, as technology and common business use warrant.

But there are other advantages to having a full architecture laid out, such as that specified by the CIF. One advantage of an architecture is to see how other people who have gone before you have solved the same problems that you are

facing. Nothing is worse than feeling as though you are facing problems that no one has ever encountered. (Rarely is that the case.) There is a certain comfort in knowing that someone, somewhere, has faced the same problem and has found a solution. Even though the issues are still in front of you, the fact that you are not facing a unique and insurmountable problem is of some comfort.

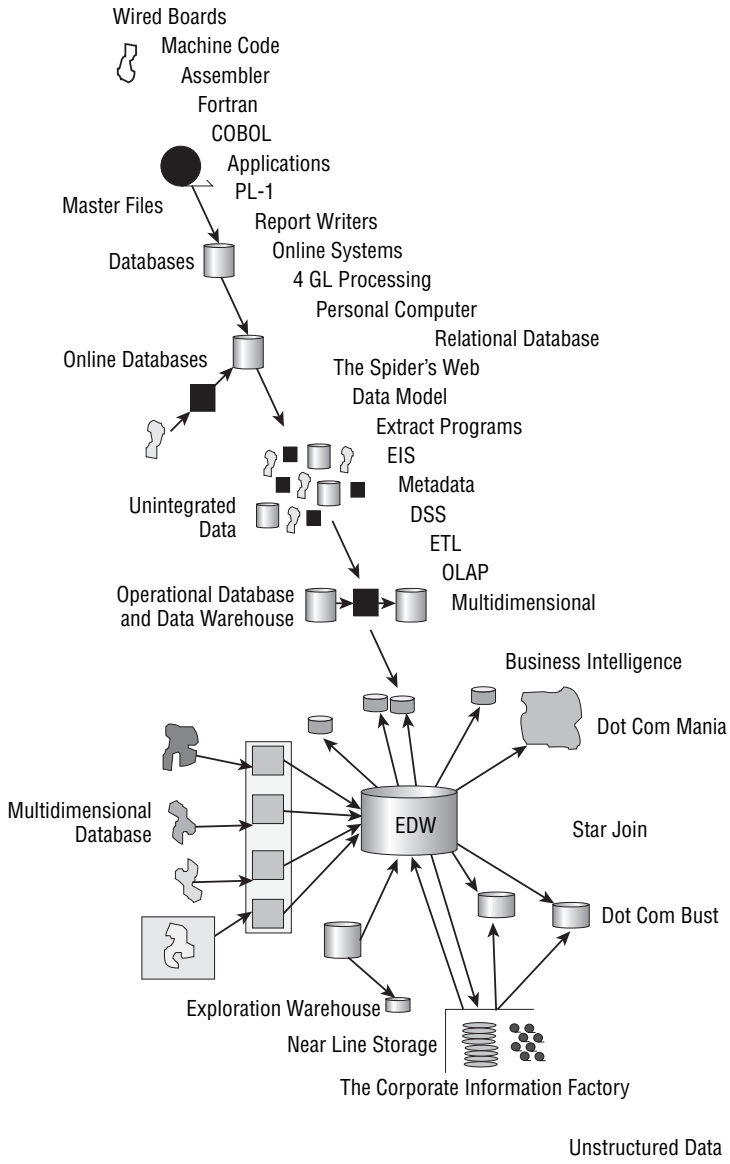


**Figure 1-6** From paper tape to the CIF

And a blueprint can do just that. You look at the blueprint, you see where other people have addressed the same problem, and you don't feel quite so alone. Another use of a blueprint is that it helps you plan. The blueprint allows you to see all that is in front of you and to make a rational decision as to the order that architectural components should be addressed. Blueprints, then, are a very useful planning tool. Figure 1-7 shows a blueprint and the CIF architecture.

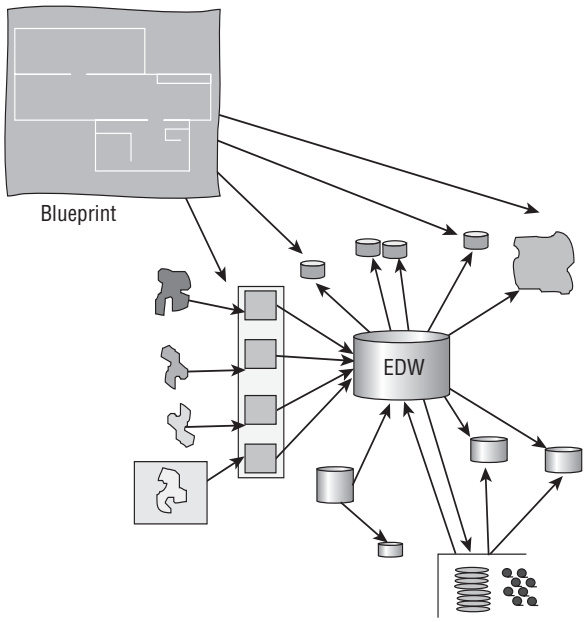


**Figure 1-7** The long-term value of a blueprint

## *Enterprise Data Warehouse*

At the heart of the CIF architecture that has been described is the data warehouse (sometimes called the *enterprise data warehouse* or the EDW). The data warehouse acts — in many ways — like a terminal (such as Grand Central Station). In a terminal, data arrives, is stationed there for a while, and eventually moves on to its ultimate destination. Unlike a terminal, in which a person has a single destination, data in a data warehouse has multiple destinations.

For example, a woman sitting in Grand Central Station wants to go to New Haven, CT. If she is going to New Haven, she is not also going to Baltimore, MD, at the same time. But a unit of data in the data warehouse does not have the same restrictions. A unit of data sitting in the data warehouse may have as its ultimate destination two data marts, an exploration warehouse, and near-line storage all at the same time.

As previously described, data in the data warehouse is at the lowest level of granularity. This low level of granularity is the most important factor to the success of the data warehouse. With data at a low level of granularity, many different forms of data can be created. One person can summarize data by the day, another by the week, and another by the month. One person can look at data by sales region, another by product, and another by package. When data is at a low level of granularity, it can be shaped and reshaped many ways. The fact that the information entering the EDW is transformed into an integrated data set means that lots of people may look at the same data in lots of ways.

Not only can integrated, granular data serve today's need for data, but integrated, granular data can serve future, unknown needs. With a body of integrated, granular data available, the business analyst is prepared for tomorrow's needs for information, even though tomorrow's needs have not been articulated.

An EDW contains a lot of historical data. Unlike almost anywhere else in the IT environment, historical data is welcomed in the data warehouse. Consider historical data and the OLTP environment. The OLTP environment centers on performance during the inserting and updating of records. And one of the most basic ways that OLTP developers achieve performance is to remove historical data from the transaction-processing environment. Historical data acts like cholesterol in slowing high performance. For these reasons, the OLTP systems programmer typically jettisons historical data as fast as possible. But in the data warehouse, historical data is welcomed. Two years, three years, or even ten years of historical data can be found in a data warehouse.

### *Extraction Transformation Loading*

It is important to remember that in the beginning there was just the data warehouse. The data warehouse by itself was a significant accomplishment. But it was very quickly recognized that the data warehouse was difficult to build because there was no way to create the interface programs between the legacy environment and the data warehouse, other than to write them by hand. But with the innovation of extract/transfer/load (ETL) software, which automatically creates the interfaces needed to bring data into the data warehouse, the floodgates for the building of data warehouses were opened.

One of the real benefits of ETL processing is that data enters the ETL process in a specific application mode and exits in an integrated enterprise mode. The processing that occurs inside ETL software allows data to be integrated and metadata about the transformations to be cataloged.

### Operational Data Store

After the data warehouse was built, another type of architectural structure was needed — the ODS.

**NOTE** **The operational data store (ODS) is a hybrid structure that has characteristics of both the data warehouse and operational systems. Because the ODS is a hybrid structure, it is difficult to build and operate. The ODS allows the user to have OLTP response time (two to three seconds), update capabilities, and DSS capabilities.**

In truth, not all organizations needed an ODS. If an organization had a need for integrated operational processing, or if the organization needed online transaction response time for access to integrated data, there was a need for an ODS.

Other architectural entities began to appear. One of those entities was the *virtual operational data store* (VODS). The VODS was a temporary structure where data from a variety of sources was pulled together to form an analysis. The VODS is very inexpensive to construct, and with the right technology, the VODS can be constructed very quickly. However, the VODS provides only a fleeting glimpse of data. If a VODS report is constructed at 11:43 A.M., the analysis is valid only for that moment. In other words, if the same analysis is constructed at 12:03 P.M., the organization could not expect to get the same results, since the underlying data may have changed between 11:42 A.M. and 12:03 P.M. Chapters 4 and 5 discuss the various classes of ODS in greater detail.

### Data Marts

Shortly after the ODS was discovered, data marts came into being. A *data mart* is a departmental manifestation of the data warehouse. While the data warehouse was built for the entire enterprise, the data mart is built for one group of like-minded users (such as the finance department, the sales department, the marketing department, and so forth). The data mart collected data from the data warehouse and reshaped the data to create a form and structure of data applicable to the department that uses it. Data marts made widespread usage of multidimensional technology, which happened to fit the needs of data-mart processing.

In addition, *adaptive project marts* (sometimes called *adaptive data marts*) began to appear. An adaptive project mart was a highly flexible temporary structure. The difference between a data mart and an adaptive project mart is that once built, a data mart becomes a permanent structure, while an adaptive project mart is never a permanent structure. The adaptive project mart can evolve into a full-fledged data mart over time, but it is not the nature of an adaptive project mart to become a permanent structure.

### Hub and Spoke

The architecture that resulted from the adoption of data marts was called *hub-and-spoke architecture.* Similar to the flight plans and strategies used by the commercial airline industry where a city acts as the hub and enables connections to various destinations through routes, the data warehouse sits at the hub, and various analytic applications and data marts act as destinations. The process of delivering information to the destinations is analogous to the routes or spokes.

### Exploration Warehouse

The next structure added to the CIF was the *exploration warehouse* or *data mining warehouse*. The exploration warehouse is a place where data miners or exploration analysts go to do statistical analysis of data or to create hypotheses and assertions. By isolating the statisticians from the data warehouse, several benefits accrued.

First, there was no performance conflict between the statistician and the normal user of the warehouse. Second, on occasion, the exploration analyst performed heuristic processing. There may have been a need to "shut off" the flow of fresh data to verify the results of changing an algorithm. By having a separate warehouse, the analyst could "shut off" new data from the statistical-analysis environment.

### Near-Line Storage

As data warehouses grew, they started to exhibit some characteristics not previously seen. With the growth in data volume, the data inside the data warehouse divided itself into one of two classes: *frequently used data* and *infrequently used data*. In fact, in most large data warehouses, far less data was being used than was not being used. It made no sense to keep unused data in expensive, high-performance storage. A new architectural component started to grow out of the desire to optimize storage investments. The component is referred to as *near-line storage*.

Near-line storage is not the same physical storage type as disk storage. Typically, near-line storage is sequential storage controlled by robots. Near-line storage is suitable for storing large, bulky data that does not need to be updated, because it is only accessed infrequently. This description of near-line data fits perfectly with the need to store infrequently accessed data from a data warehouse. In many ways, the near-line storage component becomes a form of "overflow" storage for the data warehouse.

## Government Information Factory

On September 11, 2001, the world changed. One of the many changes that occurred that day was the awareness that government information systems needed to be altered. Prior to September 11, government information systems were typified by what can be called *stovepipe systems*. Stovepipe systems are individual and do not share data. One agency gets one piece of data, another agency gets another piece of data, and yet another agency gets another piece of data. And there is no way to piece all this data together to form a useful and meaningful picture.

The events of September 11 proved to the government and the public in general the weakness of stovepipe information systems.

Unfortunately, correcting the difficulties of stovepipe systems is a hard task. Correcting stovepipe systems requires not a new technology, not a new methodology, and not a rebuilding of systems. To solve the problems of stovepipe systems, there must be a change in architecture and a change in basic attitudes of the agencies collecting and using data. Without both of these changes, there can be no victory over the tyranny of stovepipe systems.

From an architectural standpoint, the CIF evolved into the *government information factory* (GIF) as a result of the events of September 11. Figure 1-8 shows the evolution from the CIF to the GIF.

The GIF in many ways is similar to the CIF. Indeed, about 60 percent of the architecture is the same. But there are some interesting and significant differences between the GIF and the CIF, as explained in the following sections.

### *Integration Among Agencies*

The GIF requires integration among agencies; the CIF does not require the same level of integration.

To see how the CIF is different, consider a commercial data warehouse. Chevron builds a data warehouse for its own purposes. Chevron does not build a data warehouse to share data with Bank of America. Chrysler builds a data warehouse to suit its own information needs. Chrysler does not build a data warehouse to share information with Burlington Northern. Citicorp builds a data warehouse for its information needs. Citicorp does not build a data warehouse to share information with Pepsi Cola and so forth.

In the commercial world, data warehouses are built to suit the needs of the entity that owns the data. This is not so in government. If the government is ever to break out of the mold of stovepipe systems, there *must* be true sharing of data among agencies. This means that the FBI must share data with the CIA; the Immigration and Naturalization Service (INS) must share data with the Internal Revenue Service (IRS); the Health Care Finance Administration (HCFA) must share data with the Social Security Administration (SSA) and so forth.

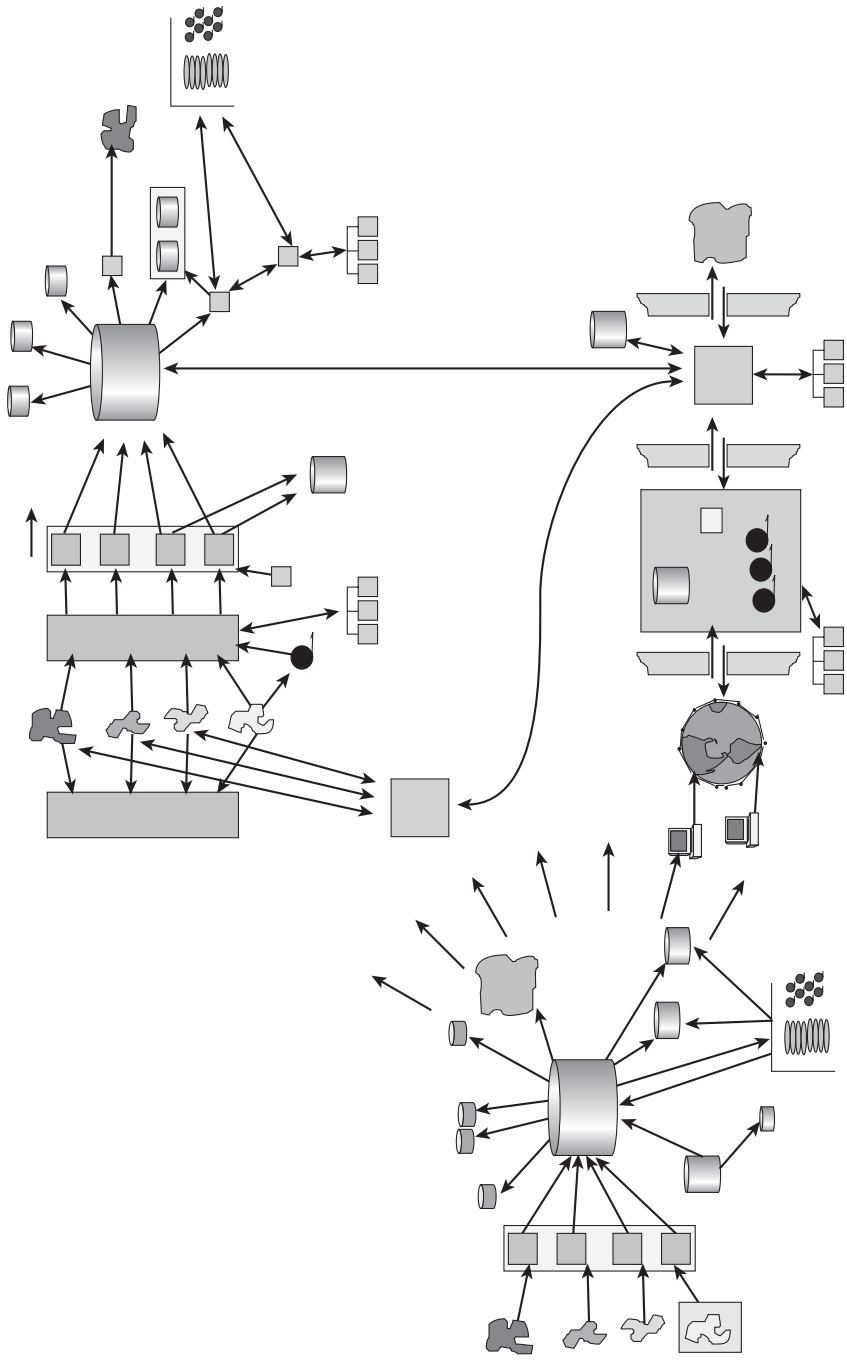

**Figure 1-8** From the CIF to the GIF

The mentality of sharing data across agencies is required to remedy the stovepipe attitude. And the architecture that supports that sharing is a big part of the picture. For these reasons, then, the GIF is significantly different from the CIF, because the CIF assumes no agency-wide sharing of data.

### *Security*

The CIF calls for security. But in truth security in the CIF environment is almost an afterthought. In the GIF, however, security is paramount.

In the world of the CIF, if security is breached, someone loses money. But in the world of the GIF, if security is breached, someone may die. So there is a significant difference between emphasis on security in the two environments.

For example, both active and passive security are found in the GIF. *Active security* is designed to keep someone from having unauthorized access to certain data. *Passive security* is not designed to stop anyone from doing anything. Instead, passive security keeps track of what has been done, should there be an unauthorized access of data. And security is found at many places in many forms in the GIF environment,

### *Longevity of Data*

Data lives longer in the government environment than in the commercial environment. For this reason, bulk-data management, near-line storage, and archival processing all have a very important role in the GIF.

## Evolution of SAP

The CIF represents the progression of thought and development that occurred with informational processing. This progression occurred in the same timeframe that SAP and Enterprise Resource Planning (ERP) were developing and maturing. It was inevitable that the worlds of SAP/ERP and data warehouse/corporate information factory would merge.

From the standpoint of timing, the CIF was intellectually articulated before SAP Business Information Warehouse (SAP BW) became available. However, this fact hardly means that the business world went out and built the CIF immediately. Yet, that SAP BW followed the intellectual establishment of the CIF in no way diminishes or tarnishes the value of the SAP BW product. Indeed, SAP BW provides an easy path to the actualization of the CIF for many organizations.

In many ways, the CIF is like a city plan. A city plan takes years and decades to develop. In the case of grand cities built from a plan (such as Washington,

D.C.), it may take decades for the city plan to be realized. Thus it is with the CIF. The CIF is a blueprint and, as such, may require many years for implementation. The CIF in a robust state is illustrated in Figure 1-9.

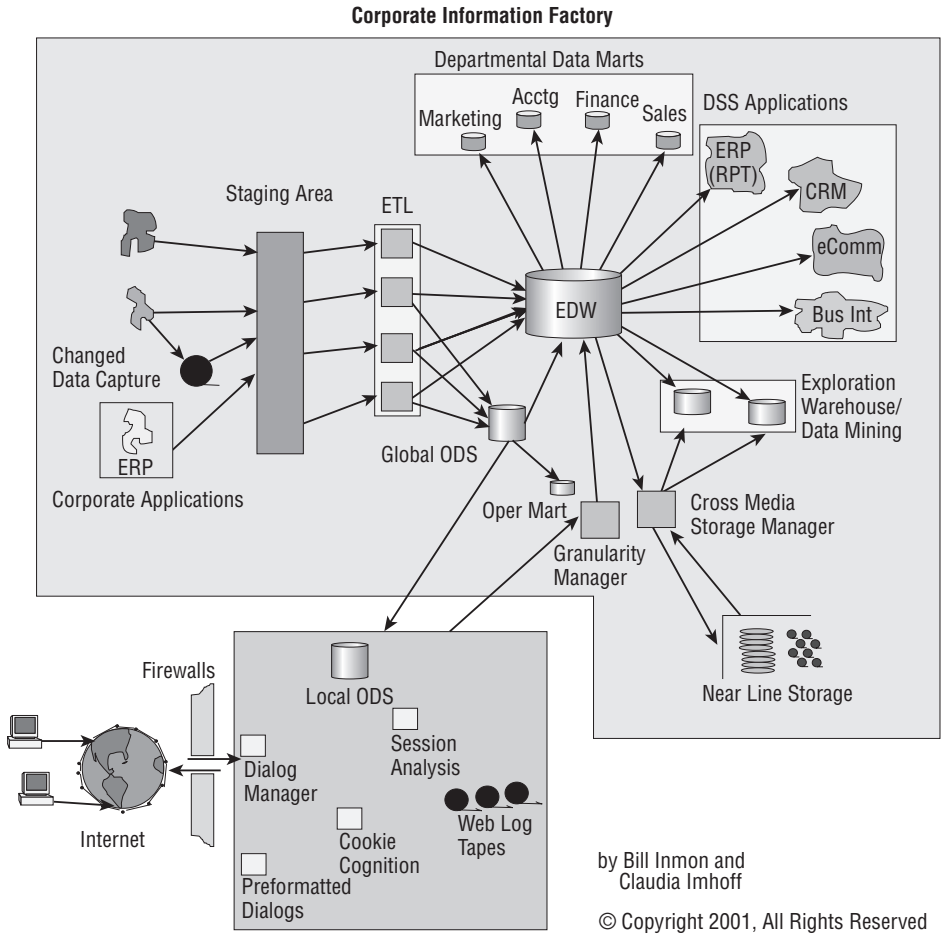


**Figure 1-9** The CIF and the Web-based e-business environment

The progression of SAP started with the early vestiges of R/2, the mainframe predecessor to its client/server brother SAP R/3. What made SAP so enormously successful and set it on the road to dominating the ERP market? Was it the support of complex business processes without writing specific application code for each variation of a business process? Was it the technological advancements of so-called lock objects? Was it SAP's heavy investment in its proprietary Advanced Business Application Programming (ABAP) language? Was it the partnering with the big consulting firms? Was it business-process reengineering or Y2K? Perhaps it was the elimination of the cap on the

commissions that SAP sales people could earn. Whatever the reason, today the letters SAP are synonymous with Enterprise Resource Planning (ERP), and there are many indications that SAP will also become synonymous with business intelligence (BI).

SAP and ERP have developed from early applications that ran financial transactions to a complete solutions set serving the needs of entire vertical markets. More recently, SAP extended the product line to e-business, customer relationship management, supply chain management, and enterprise portals.

**NOTE** **SAP markets solutions that contain components. For example, the mySAP Business Suite consists of the MySAP ERP, MySAP CRM, and NetWeaver, which includes the business-intelligence capabilities discussed throughout this book.**

The advent of SAP occurred over a three-decade period. Movement was afoot for organizations to build a solid applications base well before the year 2000, but there is no question that the challenges to the world of information technology posed by the turn of the century gave impetus to the need for revamping the enterprise software applications. Many organizations decided to completely replace their legacy applications, rather than go back into older applications and refurbish them to handle the year-2000 problem. But there were problems with the older applications other than those posed by the year-2000 dilemma. Some of the problems included the following:

- **Older applications had long ago ceased to be documented** — No one knew what the application really did or how it worked.
- **The applications were brittle and fragile** — Corporations were afraid to go into an application and make changes unless something completely unexpected and unrelated happened.
- **The applications were written in older technology** — Applications were designed to store data efficiently, rather than in an easily accessible way. Consequently, data was hard to get to.
- **The applications were not integrated** — Often acquired by merger, purchase, or other means, older legacy applications were never designed to run in an integrated manner.
- **The staff that built the older legacy applications left** — This means that no one has the knowledge or skills to update older legacy applications.

ERP applications promised not only to be the panacea for Y2K and business-process reengineering, but they also promised to provide an integrated view of a corporation's information. Once the ERP solution was implemented, however, organizations discovered that solving the problems of transaction processing

was different from solving the problems of informational processing. In truth, this was the same discovery that had been made previously by the non-ERP community, the result of which was the advent of data warehousing.

There still was the need for information in the face of a successful ERP implementation. Once the ERP environment was created, the corporation asked, "Where's my information?" It simply wasn't there or could not be accessed.

Why is there a need for information in the ERP environment even when there is a solid application foundation? There are several reasons:

- ERP applications are not designed to store history. Applications store current information, and many powerful forms of analysis need history.
- ERP applications create an integrated environment when *all* the applications are under the ERP umbrella. But often only some of the applications are ERP based. In this case, there is still a need for integration.
- The technology optimal for running ERP processing is optimized on the efficient running of transactions. Informational processing does not run well on this kind of technology (that is, predictable processes are found in the transaction-processing world, while unpredictable requests are made in the analysis world).
- ERP applications often have thousands of tables collecting data in a highly normalized schema that may be difficult to access and to make available to the informational analyst.
- ERP applications require their own processing windows. These processing requirements are often for real-time information, while informational processing may tolerate latency.

There are a host of reasons why a need still exists for informational processing even after a successful implementation of ERP. One of the more common questions we are asked by data-warehousing savvy professionals is this: "Why did SAP create a data warehouse and business intelligence toolset when there are several vendors with mature tools in the marketplace with whom they could have partnered?" The answer is really quite simple when you look at SAP's history of development in the areas of reporting and analysis and data management. It already had several years of experience developing such tools as part of SAP R/3. A brief look back to the origins of SAP reporting and analysis may shed some light on the decision-making process.

## Evolution of SAP Reporting and Analysis

The first approach taken by SAP was to make reports easier to obtain. SAP accomplished this by creating an information systems layer within the SAP R/3

product. The information systems layer was built directly in the SAP R/3 in releases as early as 2.0. The information systems layer differed throughout the different application areas in the SAP R/3 product. For example, the Logistics Information System (LIS) had its own set of information access tools called standard and flexible analysis, whereas the Human Resources Information System (HRIS) had its own set of query tools. Each of these tools was developed by its respective development team. While this leveraged the deep domain expertise of application developers, it created challenges for implementing organizations.

The first challenge was that the different information systems needed to be configured in totally different ways. Where the LIS utilized InfoStructures, Communication Structures, and Copy Methods, the HRIS utilized Infotypes, Logical Databases, and ABAP Queries. These configuration and administration differences created a need for specialists to configure the information systems, because the data structures and update mechanisms varied widely.

The second challenge was for information consumers or end users. End users were forced to learn different tools for accessing information depending on the application area the information was originally processed in. In many cases, seemingly simple cross-application reporting requirements were satisfied only by creating multiple reports or custom-written ABAP programs. These differences caused a tremendous amount of frustration and waste because end users were requesting information that crossed modules. A seemingly simple request to view purchase orders' line items with accounts payable to a vendor would more than likely be satisfied with two different reporting tools.

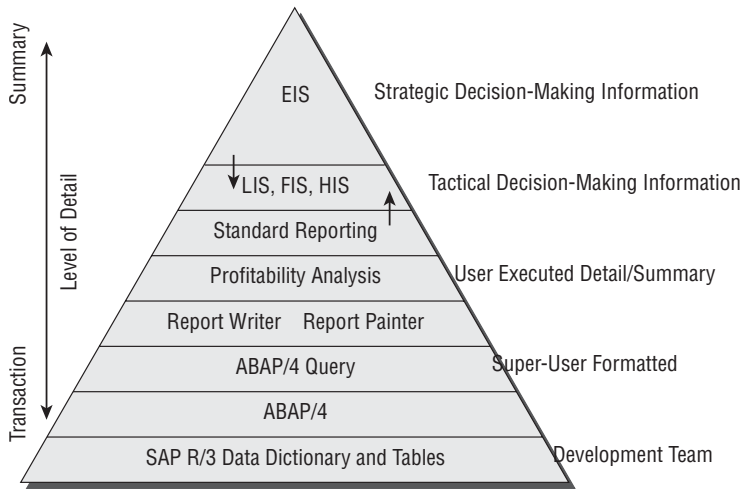


**Figure 1-10** Legacy-reporting environment

Source: Hashmi, Naeem, *Business Information Warehouse for SAP* (Prima Tech 2000, ISBN 0-7615-2335-9)

The original development work that had been done in SAP R/3 in the online analytic processing (OLAP) area was originally called a *research processor*. In the early 1990s, the tool was used initially by the *Controlling Profitability Analysis* (CO-PA) development team as a means of reporting profitability across numerous dimensions by allowing the end user to interactively navigate through a virtual cube of aggregated data. This tool is found in SAP R/3 and is referred to as *Drill-Down Reporting*. The customer demand for multidimensional analysis caused the different development departments to start to adopt the Drill-Down Reporting tool in other aspects of SAP R/3.

SAP found itself with a handful of very powerful analytic tools developed in the SAP R/3 application, a dissatisfied customer base hungry for information processing, a thriving partner ecosystem consisting of vendors of maturing business-intelligence tools, and a fair amount of knowledge on how to architect a separate data-warehousing solution. This combination led to SAP cofounder Hasso Platner's mandate that SAP create a reporting server. Thus, the SAP Business Information Warehouse (SAP BW) was conceived.

## SAP BW and the New Dimension Applications

When SAP BW was designed, it may never have been a consideration to use a third-party OLAP engine. More than likely, it was a very quick decision to port the concepts from SAP R/3 to SAP BW and leverage its development experience. Two examples of tools developed years ago as a core part of SAP R/3 that have found their way into SAP BW (not the code but the concepts) are the *Early Warning System* and the *Report-to-Report Interface* (RRI). In highlighting the reuse of R/3 tools, we are not saying that specific code was ported from one product to the next but only that SAP has had many years of experience in business intelligence and OLAP.

The Early Warning System, developed as part of the LIS in SAP R/3, has also found its functionality (not its code) in SAP BW as the reporting agent. Setting thresholds for conditions and exceptions, and announcing the findings to a user or group of users, is by no means a development revelation, but an evolution from R/3. A common use of the reporting agent is monitoring vendor-managed inventory. The quantity of inventory at a customer's location may be the vendor's responsibility to restock. This process may be managed with the assistance of the reporting agent. The vendor in this case would set an alert in SAP BW that is to be evaluated at regular intervals to make certain that when inventory levels of the product reach a reorder amount, the category manager is automatically notified so a replenishment order is shipped.

The RRI and the reporting agent are just two examples of many functions originally developed in R/3. SAP BW is built on the same base technology as SAP R/3: the SAP Web Application Server (WAS). (This technology is examined

further in Chapter 2.) SAP has significant experience gained over many years solving business problems with OLAP and data-management technologies that predate SAP BW.

SAP has been developing software according to its city plan, or, as they refer to it, a *solution map*, for the past ten years. Like a city plan, SAP's solution maps do provide the general direction in which SAP is heading with the software releases they are providing. More or less according to plan, SAP has evolved from its enterprise resource planning (ERP) origins into a complete business software solutions company covering areas such as customer-relationship management, supply-chain optimization, and business intelligence. Now, SAP is once again evolving. This time it is leading the change process.

## The Road to Business Process Integration

Throughout the past decade, the IT industry has evolved from proprietary technologies and interfaces to open standards that are independent of any one vendor's technical architecture. One of the outcomes of this evolution is known as *Web services*. Web services are programs that perform a function using open standards that enable the application to be self-contained and described. Once deployed, a Web service may be discovered and invoked by other Web services and applications.

For this type of integration to occur, quite a few standards needed to be agreed upon by the software vendors and industry leaders. The following contains a list of common Web-service standards:

- **Hyper Text Transfer Protocol (HTTP)** — Enables the exchange of various file types over the World Wide Web.
- **Extensible Markup Language (XML)** — Provides the structure and semantics for definition, interpretation, validation, and transmission of data.
- **Simple Object Access Protocol (SOAP)** — XML-based framework that describes what is in a message and how to process a given instance of application-defined data types.
- **Universal Description Discovery and Integration (UDDI)** — A directory that provides a standard way of describing Web services and finding already published services.
- **Web Services Description Language (WSDL)** — The language the UDDI directories use to describe how to access a service, as well as the operations the services will perform.

The general acceptance of these standards is only the first step toward dynamically assembled business processes that span trading partners. So-called *Service*

*Oriented Architectures* (SOAs) have grown out of the ability to provide and consume Web services.

An analogy commonly used to describe the challenge of integrating applications is that of verbal communication. To communicate verbally, people must first have an agreed-upon alphabet. The Web-service standards act as an alphabet. It is clear that more is needed to communicate verbally; for example, words and the definitions of those words must be agreed upon as well.

The software industry has been defining its vocabulary. This vocabulary is commonly referred to as *business objects*. Business objects have a set of characteristics and functions that describe them and their behavior. For example, the Customer object may have characteristics (or attributes) such as Company Name, Address, Credit Limit, and so on. As well, the Customer object supports a set of functions (or methods) such as Create Customer, Display Customer, and so on. The standards for these so-called business objects are primarily driven by vertical industry leaders and their supply-chain business partners. Table 1-2 lists the organizations driving standards for describing business objects by industry.

**Table 1-2** Driving Industry Standards

| ORGANIZATION | INDUSTRY |
|---|---|
| 1SYNC | Consumer products manufactures and retail |
| RosettaNet | High-tech manufacturing |
| CIDX | Chemical |
| Extensible Business Reporting Language (XBRL) | Accounting (cross industry) |
| Automotive Industry Action Group (AIAG) | Automotive OEM |
| Standards for Technology in Automotive Repair (STAR) | Automotive retail |
| papiNet | Paper |
| Petroleum Industry Data Exchange (PIDX) | Petroleum |
| Association for Cooperative Operations Research and Development (ACORD) | Insurance |
| Health Level Seven (HL7) | Healthcare |
| SWIFT | Banking |

These organizations are not only driving standards for business-object definition but also for business-process standardization. The standardization of business processes and the individual tasks within a business process will provide the quantum leap in productivity for organizations able to effectively

manage, monitor, and modify them. Alongside the process of defining standards for processes is the process for defining standards for how to define and invoke tasks within a business process.

As organizations define and publish services that adhere to a common set of technical and semantic standards, the next challenge becomes effectively utilizing these Web services. It is not a trivial problem. Keeping in mind that the goal of exposing these Web services is to be able to flexibly adapt business processes (especially when a business process crosses an enterprise's boundaries and becomes an integral part of a business partner's business process) integrated with business partners, difficult problems emerge.

If we look to a common business process such as canceling a customer order, an ugly set of problems emerge when the individual tasks within the business process "cancel order" are performed by business partners. The canceling of an order would trigger events such as stopping production, purchasing of component parts, crediting the customer's accounts, adjusting the commissions because of the referring channel partner, and so on.

This is where having an Enterprise Services Architecture (ESA) enters the IT landscape. Many times, business processes are not able to be undone by calling a single Web service. Much like drilling a hole into a piece of wood, there is no un-drill function. Granted there are some tasks in business processes that may be one-way in nature, but others are bidirectional and require either synchronous or asynchronous responses, If/Then/Else-style routing, and so on. Organizations may no longer hardwire their business processes and remain competitive. ESA raises the level of intelligence from a SOA in that it enables complete business functions to be managed, monitored, secured, and modified in real-time.

Let's look at the cancel order example a bit further. Some of the tasks in canceling the customer order are fairly straightforward and conform to conventional ERP thinking. Stopping the sales order is integrated into the production planning and purchasing systems within the organization canceling the order.

Now, imagine that the business partners are as forward-looking as we are and they are able to provide sub-assembled components for our production orders. In fact, we automatically send two vendors a component-availability request and await a reply from both vendors. If only one vendor replies to our request, we use that vendor. If both vendors reply, we automatically decide which supplier to use for a customer's order based on a set of business rules that include shipping date and proximity to our customer's desired shipping location. Canceling such an order may require a message to be sent to the supplying vendor. The vendor may (manually or in an automated fashion) check a set of contractual conditions before confirming the cancellation of our purchase order and calculate the appropriate cancellation penalty.

As we describe further in Chapter 2, SAP has evolved into a provider of a business process integration platform that enables an ESA to be actualized on

its SAP NetWeaver software. You may be asking yourself what this has to do with business intelligence. Everything. The adoption of ESA in organizations allows for analytics to truly be embedded into business processes and the results of analytic processes to alter the tasks in an instance of a business process. Henry Morris's vision back in the mid-1990s when he coined the term *analytic application* is now bound for ubiquity.

**ANALYTIC APPLICATIONS**

**Dr. Henry Morris of IDC coined the term *analytic application* in 1997 as packaged software that meets three criteria:**

1. **Process support: Packaged application software that structures and automates a group of tasks pertaining to the review and optimization of business operations or the discovery and development of new business.**
2. **Separation of function: The application functions independently of an organization's core transactional applications yet can be dependent on such applications for data and may send results back to these applications.**
3. **Time-oriented, integrated data: The application extracts, transforms, and integrates data from multiple sources, supporting a time-based dimension for analysis of past and future trends.**

**(Source: IDC Bulletin 1997)**

During the past five years, information-based services have become a critical part of business processes in the financial services industry (think fraud protection and anti-money laundering). You could foresee a whole new set of information-based business services being made available as the adoption of ESAs continues.

One not-so-far-fetched example could involve eBay. Imagine you are purchasing a digital camera on eBay. With the entire historical marketplace data that eBay collects on sales of digital cameras, they could train a predictive model to evaluate the probability of a specific digital camera selling for less than the amount you have just entered as your maximum bid. Based on very low probability of the camera auction closing at a lower price, eBay could offer price-protection insurance to bidders. The insurance coverage and premium would scale up and down based on the results of the historically trained model with all its variables (day, time of day, merchandising options, auction closing rates, and so on) and the real-time marketplace data for other cameras currently being offered. Of course, not every camera being auctioned would qualify for price-protection insurance.

This type of real-time adaptability (which auctions qualify for the insurance offer and at what bid prices) could enable the auction site an opportunity to

increase revenue, while not upsetting their sellers with ever-increasing posting fees, and increase buyer satisfaction by providing peace of mind that they are not irrationally overbidding.

## Summary

In the beginning were simple applications, followed by online applications. Soon a spider web of systems and data was created. With the spider web came many difficulties, such as redundancy of data, lack of integrity, and the inability to make changes.

In parallel it was recognized that there was a fundamental difference between operational data and informational (or DSS) data, requiring a change in architecture. The spider-web environment needed to be split into two kinds of processing: operational processing and informational processing. Operational processing was done out of an ERP system like SAP, and informational processing was done out of a data warehouse, which became the "single version of truth" for the corporation at the most granular level.

Data warehouses — whether from SAP environments or non-SAP environments — share some common elements:

- **Data model** — The ERD determines what the major subject areas are. The data item set determines what the attribution and keys will be.
- **Different physical tables or databases linked by a common key structure** — The different tables are built in an iterative manner. It is patently a mistake to build the data warehouse in an all-at-once manner or a "big bang" approach.
- **ETL processing** — ETL processing accesses legacy/operational data and prepares the data for entry into the data warehouse. ETL processing represents the point of integration for data found in the data warehouse.
- **Granular data** — Granular data can be examined in many ways, and it sits in wait for unknown requirements.
- **Robust history** — Typically, five years' worth of data is found in the data warehouse environment.

The data warehouse sets the stage for the many different forms of business intelligence and is the central component in the CIF. The adoption rate of industry standards is enabling adaptive business processes to be managed at a task level, with analytics and business intelligence being embedded into business processes.

Chapter 2 discusses the components of SAP NetWeaver and examines how it may be used to drive business analytics.