Introduction to Data Warehousing

Information assets are immensely valuable to any enterprise, and because of this, these assets must be properly stored and readily accessible when they are needed. However, the availability of too much data makes the extraction of the most important information difficult, if not impossible. View results from any Google search, and you'll see that the data = information equation is not always correct—that is, too much data is simply too much.

Data warehousing is a phenomenon that grew from the huge amount of electronic data stored in recent years and from the urgent need to use that data to accomplish goals that go beyond the routine tasks linked to daily processing. In a typical scenario, a large corporation has many branches, and senior managers need to quantify and evaluate how each branch contributes to the global business performance. The corporate database stores detailed data on the tasks performed by branches. To meet the managers' needs, tailor-made queries can be issued to retrieve the required data. For this process to work, database administrators must first formulate the desired query (typically an aggregate SQL query) after closely studying database catalogs. Then the query is processed. This can take a few hours because of the huge amount of data, the query complexity, and the concurrent effects of other regular workload queries on data. Finally, a report is generated and passed to senior managers in the form of a spreadsheet.

Many years ago, database designers realized that such an approach is hardly feasible, because it is very demanding in terms of time and resources, and it does not always achieve the desired results. Moreover, a mix of analytical queries with transactional routine queries inevitably slows down the system, and this does not meet the needs of users of either type of query. Today's advanced data warehousing processes separate online analytical processing (OLAP) from online transactional processing (OLTP) by creating a new information repository that integrates basic data from various sources, properly arranges data formats, and then makes data available for analysis and evaluation aimed at planning and decision-making processes.

Let's review some fields of application for which data warehouse technologies are successfully used:

- **Trade** Sales and claims analyses, shipment and inventory control, customer care, and public relations
- **Craftsmanship** Production cost control, supplier and order support
- **Financial services** Risk analysis and credit cards, fraud detection
- **Transport industry** Vehicle management
- **Telecommunication services** Call flow analysis and customer profile analysis
- **Health care service** Patient admission and discharge analysis and bookkeeping in accounts departments

The field of application of data warehouse systems is not only restricted to enterprises, but it also ranges from epidemiology to demography, from natural science to education. A property that is common to all fields is the need for storage and query tools to retrieve information summaries easily and quickly from the huge amount of data stored in databases or made available by the Internet. This kind of information allows us to study business phenomena, learn about meaningful correlations, and gain useful knowledge to support decision-making processes.

1.1 Decision Support Systems

Until the mid-1980s, enterprise databases stored only *operational data*—data created by business operations involved in daily management processes, such as purchase management, sales management, and invoicing.

1.2 Data Warehousing

Data warehouse systems are probably the systems to which academic communities and industrial bodies have been paying the greatest attention among all the DSSs. Data warehousing can be informally defined as follows:

Data Warehousing
Data warehousing is a collection of methods, techniques, and tools used to support *knowledge workers*—senior managers, directors, managers, and analysts—to conduct data analyses that help with performing decision-making processes and improving information resources.

The definition of data warehousing presented here is intentionally generic; it gives you an idea of the process but does not include specific features of the process. To understand the role and the useful properties of data warehousing completely, you must first understand the needs that brought it into being. In 1996, R. Kimball efficiently summed up a few claims frequently submitted by end-users of classic information systems:

- "*We have heaps of data, but we cannot access it!*" This shows the frustration of those who are responsible for the future of their enterprises but have no technical tools to help them extract the required information in a proper format.
- "*How can people playing the same role achieve substantially different results?*" In midsize to large enterprises, many databases are usually available, each devoted to a specific business area. They are often stored on different logical and physical media that are not conceptually integrated. For this reason, the results achieved in every business area are likely to be inconsistent.
- "*We want to select, group, and manipulate data in every possible way!*" Decision-making processes cannot always be planned before decisions are made. End-users need a tool that is user-friendly and flexible enough to conduct ad hoc analyses. They want to choose which new correlations they need to search for in real-time as they analyze the information retrieved.
- "*Show me just what matters!*" Examining data at the maximum level of detail is not only useless for decision-making processes but is also self-defeating because it does not allow users to focus their attention on meaningful information.
- "*Everyone knows that some data is wrong!*" This is another sore point. An appreciable percentage of transactional data is not correct—or it is unavailable. You cannot achieve good results if you base your analyses on incorrect or incomplete data.

We can use the previous list of problems and difficulties to extract a list of keywords that become distinguishing marks and essential requirements for a *data warehouse process*, a set of tasks that allow us to turn operational data into decision-making support information:

- *accessibility* to users not very familiar with IT and data structures;
- *integration* of data based on a standard enterprise model;
- *query flexibility* to maximize the advantages obtained from the existing information;
- *information conciseness* allowing for target-oriented and effective analyses;
- *multidimensional representation* giving users an intuitive and manageable view of information;
- *correctness and completeness* of integrated data.

Data warehouses are placed right in the middle of this process and act as repositories for data. They make sure that the requirements set can be fulfilled.

Data Warehouse
A *data warehouse* is a collection of data that supports decision-making processes. It provides the following features (Inmon, 2005):

- It is subject-oriented.
- It is integrated and consistent.
- It shows its evolution over time and it is not volatile.

Data warehouses are subject-oriented because they hinge on enterprise-specific concepts, such as customers, products, sales, and orders. On the contrary, operational databases hinge on many different enterprise-specific applications.

We put emphasis on integration and consistency because data warehouses take advantage of multiple data sources, such as data extracted from production and then stored to enterprise databases, or even data from a third party's information systems. A data warehouse should provide a unified view of all the data. Generally speaking, we can state that creating a data warehouse system does not require that new information be added; rather, existing information needs rearranging. This implicitly means that an information system should be previously available.

Operational data usually covers a short period of time, because most transactions involve the latest data. A data warehouse should enable analyses that instead cover a few years. For this reason, data warehouses are regularly updated from operational data and keep on growing. If data were visually represented, it might progress like so: A photograph of operational data would be made at regular intervals. The sequence of photographs would be stored to a data warehouse, and results would be shown in a movie that reveals the status of an enterprise from its foundation until present.

Fundamentally, data is never deleted from data warehouses and updates are normally carried out when data warehouses are offline. This means that data warehouses can be essentially viewed as read-only databases. This satisfies the users' need for a short analysis query response time and has other important effects. First, it affects data warehouse–specific database management system (DBMS) technologies, because there is no need for advanced transaction management techniques required by operational applications. Second, data warehouses operate in read-only mode, so data warehouse–specific logical design solutions are completely different from those used for operational databases. For instance, the most obvious feature of data warehouse relational implementations is that table normalization can be given up to partially denormalize tables and improve performance.

Other differences between operational databases and data warehouses are connected with query types. Operational queries execute transactions that generally read/write a small number of tuples from/to many tables connected by simple relations. For example, this applies if you search for the data of a customer in order to insert a new customer order. This kind of query is an OLTP query. On the contrary, the type of query required in data warehouses is OLAP. It features dynamic, multidimensional analyses that need to scan a huge amount of records to process a set of numeric data summing up the performance of an enterprise. It is important to note that OLTP systems have an essential workload core "frozen" in application programs, and ad hoc data queries are occasionally run for data maintenance. Conversely, data warehouse interactivity is an essential property for analysis sessions, so the actual workload constantly changes as time goes by.

The distinctive features of OLAP queries suggest adoption of a *multidimensional* representation for data warehouse data. Basically, data is viewed as points in space, whose dimensions correspond to many possible analysis dimensions. Each point represents an event that occurs in an enterprise and is described by a set of measures relevant to decision-making processes. Section 1.5 gives a detailed description of the multidimensional model you absolutely need to be familiar with to understand how to model conceptual and logical levels of a data warehouse and how to query data warehouses.

Table 1-2 summarizes the main differences between operational databases and data warehouses.

Feature	Operational Databases	Data Warehouses
Users	Thousands	Hundreds
Workload	Preset transactions	Specific analysis queries
Access	To hundreds of records, write and read mode	To millions of records, mainly read-only mode
Goal	Depends on applications	Decision-making support
Data	Detailed, both numeric and alphanumeric	Summed up, mainly numeric
Data integration	Application-based	Subject-based
Quality	In terms of integrity	In terms of consistency
Time coverage	Current data only	Current and historical data
Updates	Continuous	Periodical
Model	Normalized	Denormalized, multidimensional
Optimization	For OLTP access to a database part	For OLAP access to most of the database

TABLE 1-2 Differences between Operational Databases and Data Warehouses (Kelley, 1997)

1.3 Data Warehouse Architectures

The following architecture properties are essential for a data warehouse system (Kelly, 1997):

- **Separation** Analytical and transactional processing should be kept apart as much as possible.
- **Scalability** Hardware and software architectures should be easy to upgrade as the data volume, which has to be managed and processed, and the number of users' requirements, which have to be met, progressively increase.
- **Extensibility** The architecture should be able to host new applications and technologies without redesigning the whole system.
- **Security** Monitoring accesses is essential because of the strategic data stored in data warehouses.
- **Administerability** Data warehouse management should not be overly difficult.

Two different classifications are commonly adopted for data warehouse architectures. The first classification, described in sections 1.3.1, 1.3.2, and 1.3.3, is a structure-oriented one that depends on the number of layers used by the architecture. The second classification, described in section 1.3.4, depends on how the different layers are employed to create enterprise-oriented or department-oriented views of data warehouses.

1.3.1 Single-Layer Architecture

A single-layer architecture is not frequently used in practice.

1.3.2 Two-Layer Architecture

Data Marts

A *data mart* is a subset or an aggregation of the data stored to a primary data warehouse. It includes a set of information pieces relevant to a specific business area, corporate department, or category of users.

The data marts populated from a primary data warehouse are often called *dependent*. Although data marts are not strictly necessary, they are very useful for data warehouse systems in midsize to large enterprises because

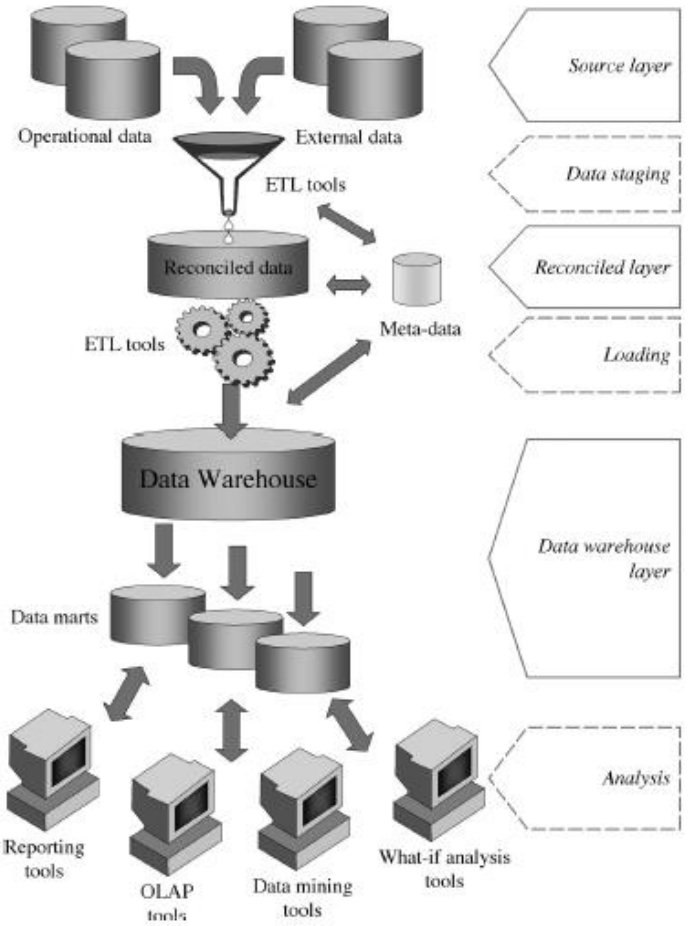
- they are used as building blocks while incrementally developing data warehouses;
- they mark out the information required by a specific group of users to solve queries;
- they can deliver better performance because they are smaller than primary data warehouses.

1.3.2 Three-Layer Architecture

In this architecture, the third layer is the *reconciled data layer* or *operational data store*. This layer materializes operational data obtained after integrating and cleansing source data. As a result, those data are integrated, consistent, correct, current, and detailed. Figure 1-4 shows a data warehouse that is not populated from its sources directly, but from reconciled data.

The main advantage of the reconciled data layer is that it creates a common reference data model for a whole enterprise. At the same time, it sharply separates the problems of source data extraction and integration from those of data warehouse population. Remarkably, in some cases, the reconciled layer is also directly used to better accomplish some operational tasks, such as producing daily reports that cannot be satisfactorily prepared using the corporate applications, or generating data flows to feed external processes periodically so as to benefit from cleaning and integration. However, reconciled data leads to more redundancy of operational source data. Note that we may assume that even two-layer architectures can have a reconciled layer that is not specifically materialized, but only virtual, because it is defined as a consistent integrated view of operational source data.

FIGURE 1-4
Three-layer architecture for a data warehouse system



Finally, let's consider a supplementary architectural approach, which provides a comprehensive picture. This approach can be described as a hybrid solution between the single-layer architecture and the two/three-layer architecture. This approach assumes that although a data warehouse is available, it is unable to solve all the queries formulated. This means that users may be interested in directly accessing source data from aggregate data (*drill-through*). To reach this goal, some queries have to be rewritten on the basis of source data (or reconciled data if it is available). This type of architecture is implemented in a prototype by Cui and Widom, 2000, and it needs to be able to go dynamically back to the source data required for queries to be solved (*lineage*).

1.4 Data Staging and ETL

Now let's closely study some basic features of the different architecture layers. We will start with the data staging layer.

The data staging layer hosts the ETL processes that extract, integrate, and clean data from operational sources to feed the data warehouse layer. In a three-layer architecture, ETL processes actually feed the reconciled data layer—a single, detailed, comprehensive, top-quality data source—that in its turn feeds the data warehouse. For this reason, the ETL process operations as a whole are often defined as *reconciliation*. These are also the most complex and technically challenging among all the data warehouse process phases.

ETL takes place once when a data warehouse is populated for the first time, then it occurs every time the data warehouse is regularly updated. Figure 1-8 shows that ETL consists of four separate phases: *extraction* (or *capture*), *cleansing* (or *cleaning* or *scrubbing*), *transformation,* and *loading*. In the following sections, we offer brief descriptions of these phases.

The scientific literature shows that the boundaries between cleansing and transforming are often blurred from the terminological viewpoint. For this reason, a specific operation is not always clearly assigned to one of these phases. This is obviously a formal problem, but not a substantial one. We will adopt the approach used by Hoffer and others (2005) to make our explanations as clear as possible. Their approach states that cleansing is essentially aimed at rectifying data *values*, and transformation more specifically manages
data *formats*.

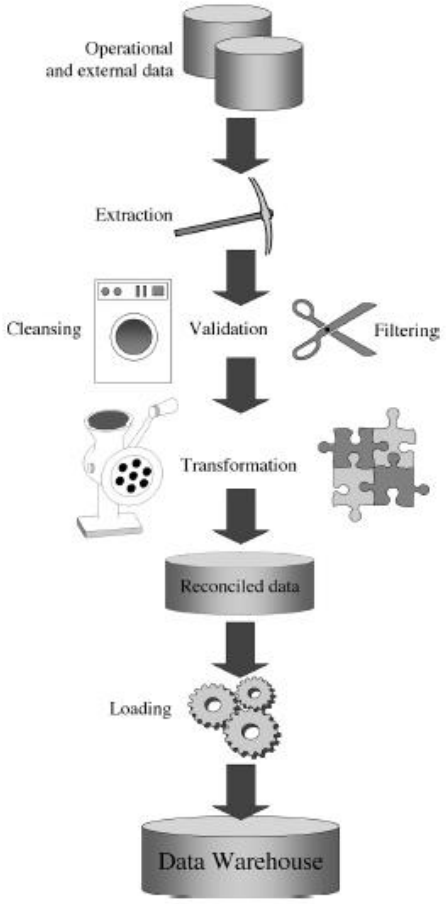
Chapter 10 discusses all the details of the data-staging design phase. Chapter 3 deals with an early data warehouse design phase: *integration*. This phase is necessary if there are heterogeneous sources to define a schema for the reconciled data layer, and to specifically transform operational data in the data-staging phase.

1.4.1 Extraction

Relevant data is obtained from sources in the extraction phase. You can use *static extraction* when a data warehouse needs populating for the first time. Conceptually speaking, this looks like a snapshot of operational data. *Incremental extraction*, used to update data warehouses regularly, seizes the changes applied to source data since the latest extraction. Incremental extraction is often based on the log maintained by the operational DBMS. If a timestamp is associated with operational data to record exactly when the data is changed or added, it can be used to streamline the extraction process. Extraction can also be source-driven if you can rewrite operational applications to asynchronously notify of the changes being applied, or if your operational database can implement triggers associated with change transactions for relevant data.

The data to be extracted is mainly selected based on its quality (English, 1999). In particular, this depends on how comprehensive and accurate the constraints implemented in sources are, how suitable the data formats are, and how clear the schemata are.

FIGURE 1-8
Extraction, transformation, and loading



1.4.2 Cleansing

The cleansing phase is crucial in a data warehouse system because it is supposed to improve data quality—normally quite poor in sources (Galhardas et al., 2001). The following list includes the most frequent mistakes and inconsistencies that make data "dirty":

- **Duplicate data** For example, a patient is recorded many times in a hospital patient management system
- **Inconsistent values that are logically associated** Such as addresses and ZIP codes
- **Missing data** Such as a customer's job
- **Unexpected use of fields** For example, a socialSecurityNumber field could be used improperly to store office phone numbers
- **Impossible or wrong values** Such as 2/30/2009
- **Inconsistent values for a single entity because different practices were used** For example, to specify a country, you can use an international country abbreviation (I) or a full country name (Italy); similar problems arise with addresses (Hamlet Rd.and Hamlet Road)
- **Inconsistent values for one individual entity because of typing mistakes** Such as Hamet Road instead of Hamlet Road

In particular, note that the last two types of mistakes are very frequent when you are managing multiple sources and are entering data manually.

The main data cleansing features found in ETL tools are *rectification* and *homogenization*. They use specific dictionaries to rectify typing mistakes and to recognize synonyms, as well as *rule-based cleansing* to enforce domain-specific rules and define appropriate associations between values. See section 10.2 for more details on these points.

1.4.3 Transformation

Transformation is the core of the reconciliation phase. It converts data from its operational source format into a specific data warehouse format. If you implement a three-layer architecture, this phase outputs your reconciled data layer. Independently of the presence of a reconciled data layer, establishing a mapping between the source data layer and the data warehouse layer is generally made difficult by the presence of many different, heterogeneous sources. If this is the case, a complex integration phase is required when designing your data warehouse. See Chapter 3 for more details.

The following points must be rectified in this phase:

- *Loose texts may hide valuable information.* For example, BigDeal LtD does not explicitly show that this is a Limited Partnership company.
- *Different formats can be used for individual data.* For example, a date can be saved as a string or as three integers.

Following are the main transformation processes aimed at populating the reconciled data layer:

- *Conversion* and *normalization* that operate on both storage formats and units of measure to make data uniform
- *Matching* that associates equivalent fields in different sources
- *Selection* that reduces the number of source fields and records

When populating a data warehouse, normalization is replaced by *denormalization* because data warehouse data are typically denormalized, and you need *aggregation* to sum up data properly.

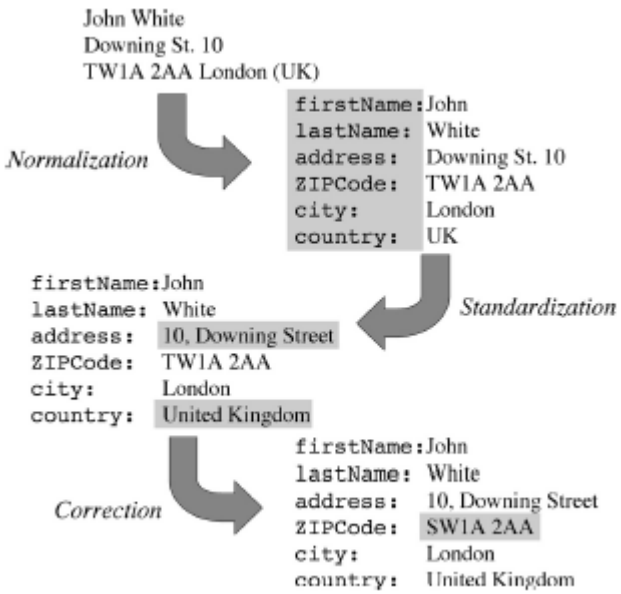


FIGURE 1-9 Example of cleansing and transforming customer data

Cleansing and transformation processes are often closely connected in ETL tools. Figure 1-9 shows an example of cleansing and transformation of customer data: a field-based structure is extracted from a loose text, then a few values are standardized so as to remove abbreviations, and eventually those values that are logically associated can be rectified.

1.4.4 Loading

Loading into a data warehouse is the last step to take. Loading can be carried out in two ways:

- **Refresh** Data warehouse data is completely rewritten. This means that older data is replaced. Refresh is normally used in combination with static extraction to initially populate a data warehouse.
- **Update** Only those changes applied to source data are added to the data warehouse. Update is typically carried out without deleting or modifying preexisting data. This technique is used in combination with incremental extraction to update data warehouses regularly.