What Is a Data Warehouse?

Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions. Data warehouses have been defined in many ways, making it difficult to formulate a rigorous definition. Loosely speaking, a data warehouse refers to a data repository that is maintained separately from an organization's operational databases. Data warehouse systems allow for integration of a variety of application systems. They support information processing by providing a solid platform of consolidated historic data for analysis. According to William H. Inmon, a leading architect in the construction of data warehouse systems, "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process" [Inm96]. This short but comprehensive definition presents the major features of a data warehouse. The four keywords—*subject-oriented*, *integrated*, *time-variant*, and *nonvolatile* —distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

■ Subject-oriented: A data warehouse is organized around major subjects such as customer, supplier, product, and sales. Rather than concentrating on the day-to-day operations and transaction processing of an organization, a data warehouse focuses on the modeling and analysis of data for decision makers. Hence, data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process.

■ Integrated: A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records. Data cleaning and data integration techniques are applied to ensure consistency in naming conventions, encoding structures, attribute measures, and so on.

■ Time-variant: Data are stored to provide information from an historic perspective (e.g., a sequence over the past 5–10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, a time element.

■ Nonvolatile: A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. Due to this separation, a data warehouse does not require transaction processing, recovery, and concurrency control mechanisms. It usually requires only two operations in data accessing: initial loading of data and access of data.

So, what is the purpose? It is said the data warehouse acts as a semantically consistent data store that serves as a physical implementation of a decision support data model. It stores the information an enterprise needs to make strategic decisions. A data warehouse is also often viewed as an architecture, constructed by integrating data from multiple heterogeneous sources to support structured and/or ad hoc queries, analytical reporting, and decision making.

Based on this information, we view data warehousing as the process of constructing and using data warehouses. The construction of a data warehouse requires data cleaning, data integration, and data consolidation. The utilization of a data warehouse often necessitates a collection of decision support technologies. This allows “knowledge workers” (e.g., managers, analysts, and executives) to use the warehouse to quickly and conveniently obtain an overview of the data, and to make sound decisions based on information in the warehouse. Some authors use the term data warehousing to refer only to the process of data warehouse construction, while the term warehouse DBMS is used to refer to the management and utilization of data warehouses. We will not make this distinction here.

How are organizations using a Data Warehouse?

Many organizations use this information to support business decision-making activities, including (1) increasing customer focus, which includes the analysis of customer buying patterns (such as buying preference, buying time, budget cycles, and appetites for spending); (2) repositioning products and managing product portfolios by comparing the performance of sales by quarter, by year, and by geographic regions in order to fine-tune production strategies; (3) analyzing operations and looking for sources of profit; and (4) managing customer relationships, making environmental corrections, and managing the cost of corporate assets.

Data warehousing is also very useful from the point of view of heterogeneous database integration. Organizations typically collect diverse kinds of data and maintain large databases from multiple, heterogeneous, autonomous, and distributed information sources. It is highly desirable, yet challenging, to integrate such data and provide easy and efficient access to it. Much effort has been spent in the database industry and research community toward achieving this goal.

The traditional database approach to heterogeneous database integration is to build wrappers and integrators (or mediators) on top of multiple, heterogeneous databases. When a query is posed to a client site, a metadata dictionary is used to translate the query into queries appropriate for the individual heterogeneous sites involved. These queries are then mapped and sent to local query processors. The results returned from the different sites are integrated into a global answer set. This query-driven approach requires complex information filtering and integration processes, and competes with local sites for processing resources. It is inefficient and potentially expensive for frequent queries, especially queries requiring aggregations.

Data warehousing provides an interesting alternative to this traditional approach. Rather than using a query-driven approach, data warehousing employs an update-driven approach in which information from multiple, heterogeneous sources is integrated in advance and stored in a warehouse for direct querying and analysis. Unlike online transaction processing databases, data warehouses do not contain the most current information. However, a data warehouse brings high performance to the integrated heterogeneous database system because data are copied, preprocessed, integrated, annotated, summarized, and restructured into one semantic data store. Furthermore, query processing in data warehouses does not interfere with the processing at local sources. Moreover, data warehouses can store and integrate historic information and support complex multidimensional queries.

What are the differences between a traditional database system and a data warehouse?

Most of us are already familiar with commercial relational database systems, such as SQL or Oracle so I am fairly certain we are capable of understanding what a data warehouse is by comparing two kinds of systems, and doing so starting with their primary tasks.

The major task of online operational database systems is to perform online transaction and query processing. These systems are called **online transaction processing (OLTP)** systems. They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting. Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as **online analytical processing (OLAP)** systems.

The major distinguishing features of OLTP and OLAP are summarized as follows:

■ Users and system orientation: An OLTP system is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.

■ Data contents: An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. An OLAP system manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity. These features make the data easier to use for informed decision making.

■ Database design: An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design. An OLAP system typically adopts either a star or a snowflake model (see Section 4.2.2) and a subject-oriented database design.

■ View: An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations. In contrast, an OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization. OLAP systems also deal with information that originates from different organizations, integrating information from many data stores. Because of their huge volume, OLAP data are stored on multiple storage media.

■ Access patterns: The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations (because most data warehouses store historic rather than up-to-date information), although many could be complex queries.

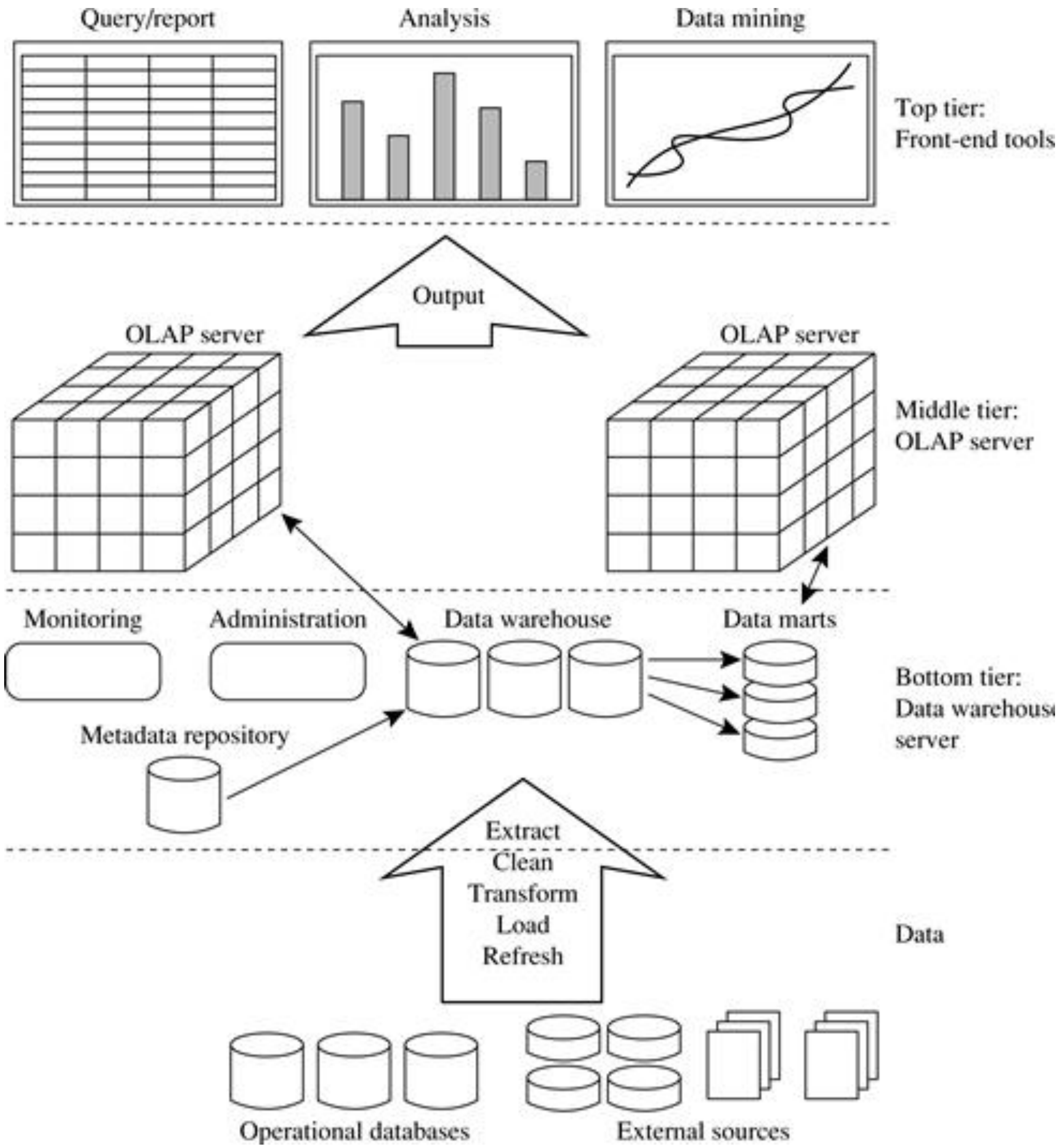
So, why separate from the database?

Because operational databases store huge amounts of data, you may wonder, “Why not perform online analytical processing directly on such databases instead of spending additional time and resources to construct a separate data warehouse?” A major reason for such a separation is to help promote the high performance of both systems. An operational database is designed and tuned from known tasks and workloads like indexing and hashing using primary keys, searching for particular records, and optimizing “canned” queries. On the other hand, data warehouse queries are often complex. They involve the computation of large data groups at summarized levels, and may require the use of special data organization, access, and implementation methods based on multidimensional views. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.

Moreover, an operational database supports the concurrent processing of multiple transactions. Concurrency control and recovery mechanisms (e.g., locking and logging) are required to ensure the consistency and robustness of transactions. An OLAP query often needs read-only access of data records for summarization and aggregation. Concurrency control and recovery mechanisms, if applied for such OLAP operations, may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

Finally, the separation of operational databases from data warehouses is based on the different structures, contents, and uses of the data in these two systems. Decision support requires historic data, whereas operational databases do not typically maintain historic data. In this context, the data in operational databases, though abundant, are usually far from complete for decision making. Decision support requires consolidation (e.g., aggregation and summarization) of data from heterogeneous sources, resulting in high-quality, clean, integrated data. In contrast, operational databases contain only detailed raw data, such as transactions, which need to be consolidated before analysis. Because the two systems provide quite different functionalities and require different kinds of data, it is presently necessary to maintain separate databases. However, many vendors of operational relational database management systems are beginning to optimize such systems to support OLAP queries. As this trend continues, the separation between OLTP and OLAP systems is expected to decrease.

Data warehouses often adopt a three-tier architecture, as presented below:



1. The bottom tier is a warehouse database server that is almost always a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse. The data are extracted using application program interfaces known as gateways. A gateway

is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

2. The middle tier is an OLAP server that is typically implemented using either (1) a relational OLAP (ROLAP) model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations); or (2) a multidimensional OLAP (MOLAP) model (i.e., a special-purpose server that directly implements multidimensional data and operations). The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

3. The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

What are the various types of DW models?

From the architecture point of view, there are three data warehouse models: the enterprise warehouse, the data mart, and the virtual warehouse.

Enterprise warehouse: An enterprise warehouse collects all of the information about subjects spanning the entire organization. It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope. It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond. An enterprise data warehouse may be implemented on traditional mainframes, computer superservers, or parallel architecture platforms. It requires extensive business modeling and may take years to design and build.

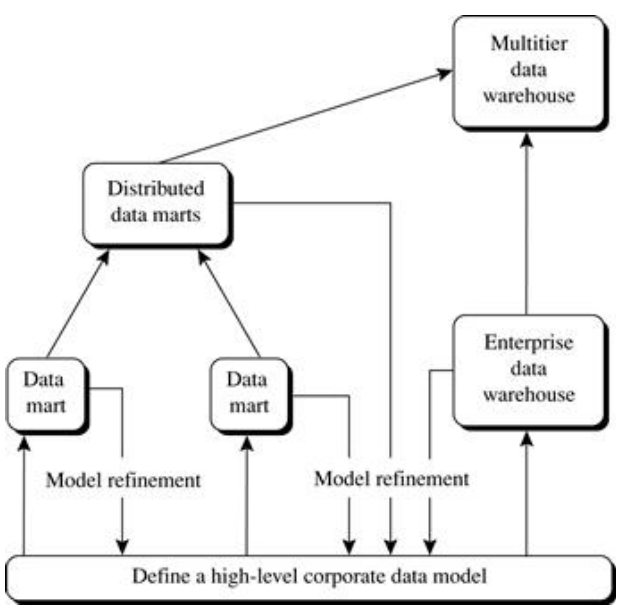
Data mart: A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized.

Data marts are usually implemented on low-cost departmental servers that are Unix/Linux or Windows based. The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years. However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.

Depending on the source of data, data marts can be categorized as independent or dependent. Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area. Dependent data marts are sourced directly from enterprise data warehouses.

Virtual warehouse: A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers. *"What are the pros and cons of the top-down and bottom-up approaches to data warehouse development?"* The top-down development of an enterprise warehouse serves as a systematic solution and minimizes integration problems. However, it is expensive, takes a long time to develop, and lacks flexibility due to the difficulty in achieving consistency and consensus for a common data model for the entire organization. The bottom-up approach to the design, development, and deployment of independent data marts provides flexibility, low cost, and rapid return of investment. It, however, can lead to problems when integrating various disparate data marts into a consistent enterprise data warehouse.

A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner, as shown below. First, a high-level corporate data model is defined within a reasonably short period (such as one or two months) that provides a corporate-wide, consistent, integrated view of data among different subjects and potential usages. This high-level model, although it will need to be refined in the further development of enterprise data warehouses and departmental data marts, will greatly reduce future integration problems. Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set noted before. Third, distributed data marts can be constructed to integrate different data marts via hub servers. Finally, a multitier data warehouse is constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.



Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and utilities include the following functions:

■ Data extraction, which typically gathers data from multiple, heterogeneous, and external sources.

■ Data cleaning, which detects errors in the data and rectifies them when possible.

■ Data transformation, which converts data from legacy or host format to warehouse format.

■ Load, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.

■ Refresh, which propagates the updates from the data sources to the warehouse.

Besides cleaning, loading, refreshing, and metadata definition tools, data warehouse systems usually provide a good set of data warehouse management tools. Data cleaning and data transformation are important steps in improving the data quality and, subsequently, the data mining results. Because we are mostly interested in the aspects of data warehousing technology related to data mining, we will not get into the details of the remaining tools, and recommend interested readers to consult books dedicated to data warehousing technology.

What is Meta-data and why is it important for DW?

Metadata are *data about data*. When used in a data warehouse, metadata are the data that define warehouse objects. The first figure in the document showed a metadata repository within the bottom tier of the data warehousing architecture. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for timestamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.

A metadata repository should contain the following:

■ A description of the data warehouse structure, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.

■ Operational metadata, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).

■ The algorithms used for summarization, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.

■ Mapping from the operational environment to the data warehouse, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).

■ Data related to system performance, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.

■ Business metadata, which include business terms and definitions, data ownership information, and charging policies.

A data warehouse contains different levels of summarization, of which metadata is one. Other types include current detailed data (which are almost always on disk), older detailed data (which are usually on tertiary storage), lightly summarized data, and highly summarized data (which may or may not be physically housed).

Metadata also play a very different role than other data warehouse data and are important for many reasons. For example, metadata are used as a directory to help the decision support system analyst locate the contents of the data warehouse, and as a guide to the data mapping when data are transformed from the operational environment to the data warehouse environment. Metadata also serve as a guide to the algorithms used for summarization between the current detailed data and the lightly summarized data, and between the lightly summarized data and the highly summarized data. Metadata should be stored and managed persistently (i.e., on disk).

What can business analysts gain from having a data warehouse?

First, having a data warehouse may provide a competitive advantage by presenting relevant information from which to measure performance and make critical adjustments to help win over competitors. Second, a data warehouse can enhance business productivity because it is able to quickly and efficiently gather information that accurately describes the organization. Third, a data warehouse facilitates customer relationship management because it provides a consistent view of customers and items across all lines of business, all departments, and all markets. Finally, a data warehouse may bring about cost reduction by tracking trends, patterns, and exceptions over long periods in a consistent and reliable manner.

To design an effective data warehouse we need to understand and analyze business needs and construct a business analysis framework. The construction of a large and complex information system can be viewed as the construction of a large and complex building, for which the owner, architect, and builder have different views. These views are combined to form a complex framework that represents the top-down, business-driven, or owner's perspective, as well as the bottom-up, builder-driven, or implementor's view of the information system.

Four different views regarding a data warehouse design must be considered: the top-down view, the data source view, the data warehouse view, and the business query view.

■ The top-down view allows the selection of the relevant information necessary for the data warehouse. This information matches current and future business needs.

■ The data source view exposes the information being captured, stored, and managed by operational systems. This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables. Data sources are often modeled by traditional data modeling techniques, such as the entity-relationship model or CASE (computer-aided software engineering) tools.

■ The data warehouse view includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including precalculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.

■ Finally, the business query view is the data perspective in the data warehouse from the end-user's viewpoint.

Building and using a data warehouse is a complex task because it requires business skills, technology skills, and program management skills. Regarding business skills, building a data warehouse involves understanding how systems store and manage their data, how to build extractors that transfer data from the operational system to the data warehouse, and how to build warehouse refresh software that keeps the data warehouse reasonably up-to-date with the operational system's data. Using a data warehouse involves understanding the significance of the data it contains, as well as understanding and translating the business requirements into queries that can be satisfied by the data warehouse.

Regarding technology skills, data analysts are required to understand how to make assessments from quantitative information and derive facts based on conclusions from historic information in the data warehouse. These skills include the ability to discover patterns and trends, to extrapolate trends based on history and look for anomalies or paradigm shifts, and to present coherent managerial recommendations based on such analysis. Finally, program management skills involve the need to interface with many technologies, vendors, and end-users in order to deliver results in a timely and cost-effective manner.

What is the data warehouse design process?

A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both. The top-down approach starts with overall design and planning. It is useful in cases where the technology is mature and well known, and where the business problems that must be solved are clear and well understood. The bottom-up approach starts with experiments and prototypes. This is useful in the early stage of business modeling and technology development. It allows an organization to move forward at considerably less expense and to evaluate the technological benefits before making significant commitments. In the combined approach, an organization can exploit the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach.

From the software engineering point of view, the design and construction of a data warehouse may consist of the following steps: planning, requirements study, problem analysis, warehouse design, data integration and testing, and finally deployment of the data warehouse. Large software systems can be developed using one of two methodologies: the waterfall method or the spiral method. The waterfall method performs a structured and systematic analysis at each step before proceeding to the next, which is like a waterfall, falling from one step to the next. The spiral method involves the rapid generation of increasingly functional systems, with short intervals between successive releases. This is considered a good choice for data warehouse development, especially for data marts, because the turnaround time is short, modifications can be done quickly, and new designs and technologies can be adapted in a timely manner.

In general, the warehouse design process consists of the following steps:

1. **Choose a business process to model** (e.g., orders, invoices, shipments, inventory, account administration, sales, or the general ledger). If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed. However, if the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.

2. **Choose the business process grain**, which is the fundamental, atomic level of data to be represented in the fact table for this process (e.g., individual transactions, individual daily snapshots, and so on).

3. **Choose the dimensions that will apply to each fact table record.** Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.

4. **Choose the measures that will populate each fact table record.** Typical measures are numeric additive quantities like dollars_sold and units_sold.

Because data warehouse construction is a difficult and long-term task, its implementation scope should be clearly defined. The goals of an initial data warehouse implementation should be specific, achievable, and measurable. This involves determining the time and budget allocations, the subset of the organization that is to be modeled, the number of data sources selected, and the number and types of departments to be served.

Once a data warehouse is designed and constructed, the initial deployment of the warehouse includes initial installation, roll-out planning, training, and orientation. Platform upgrades and maintenance must also be considered. Data warehouse administration includes data refreshment, data source synchronization, planning for disaster recovery, managing access control and security, managing data growth, managing database performance, and data warehouse enhancement and extension. Scope management includes controlling the number and range of queries, dimensions, and reports; limiting the data warehouse's size; or limiting the schedule, budget, or resources.

Various kinds of data warehouse design tools are available. Data warehouse development tools provide functions to define and edit metadata repository contents (e.g., schemas, scripts, or rules), answer queries, output reports, and ship metadata to and from relational database system catalogs. Planning and analysis tools study the impact of schema changes and of refresh performance when changing refresh rates or time windows.

As we have already described by example, data warehouses and data marts are used in a wide range of applications. Business executives use the data in data warehouses and data marts to perform data analysis and make strategic decisions. In many firms, data warehouses are used as an integral part of a plan-execute-assess "closed-loop" feedback system for enterprise management. Data warehouses are used extensively in banking and financial services, consumer goods and retail distribution sectors, and controlled manufacturing such as demand-based production.

Typically, the longer a data warehouse has been in use, the more it will have evolved. This evolution takes place throughout a number of phases. Initially, the data warehouse is mainly used for generating reports and answering predefined queries. Progressively, it is used to analyze summarized and detailed data, where the results are presented in the form of reports and charts. Later, the data warehouse is used for strategic purposes, performing multidimensional analysis and sophisticated slice-and-dice operations.

Finally, the data warehouse may be employed for knowledge discovery and strategic decision making using data mining tools. In this context, the tools for data warehousing can be categorized into access and retrieval tools, database reporting tools, data analysis tools, and data mining tools.

Business users need to have the means to know what exists in the data warehouse (through metadata), how to access the contents of the data warehouse, how to examine the contents using analysis tools, and how to present the results of such analysis.

There are three kinds of data warehouse applications: **information processing, analytical processing, and data mining.**

■ Information processing supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts, or graphs. A current trend in data warehouse information processing is to construct low-cost web-based accessing tools that are then integrated with web browsers.

■ Analytical processing supports basic OLAP operations, including slice-and-dice, drill-down, roll-up, and pivoting. It generally operates on historic data in both summarized and detailed forms. The major strength of online analytical processing over information processing is the multidimensional data analysis of data warehouse data.

■ Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

"How does data mining relate to information processing and online analytical processing?" Information processing, based on queries, can find useful information. However, answers to such queries reflect the information directly stored in databases or computable by aggregate functions. They do not reflect sophisticated patterns or regularities buried in the database. Therefore, information processing is not data mining.

Online analytical processing comes a step closer to data mining because it can derive information summarized at multiple granularities from user-specified subsets of a data warehouse. Such descriptions are equivalent to the class/concept descriptions discussed in Week 1. Because data mining systems can also mine generalized class/concept descriptions, this raises some interesting questions: "Do OLAP systems perform data mining? Are OLAP systems actually data mining systems?"

The functionalities of OLAP and data mining can be viewed as disjoint: OLAP is a data summarization/aggregation tool that helps simplify data analysis, while data mining allows the automated discovery of implicit patterns and interesting knowledge hidden in large amounts of data. OLAP tools are targeted toward simplifying and supporting interactive data analysis, whereas the goal of data mining tools is to automate as much of the process as possible, while still allowing users to guide the process. In this sense, data mining goes one step beyond traditional online analytical processing.

An alternative and broader view of data mining may be adopted in which data mining covers both data description and data modeling. Because OLAP systems can present general descriptions of data from data warehouses, OLAP functions are essentially for user-directed data summarization and comparison (by drilling, pivoting, slicing, dicing, and other operations). These are, though limited, data mining functionalities. Yet according to this view, data mining covers a much broader spectrum than simple OLAP operations, because it performs not only data summarization and comparison but also association, classification, prediction, clustering, time-series analysis, and other data analysis tasks.

Data mining is not confined to the analysis of data stored in data warehouses. It may analyze data existing at more detailed granularities than the summarized data provided in a data warehouse. It may also analyze transactional, spatial, textual, and multimedia data that are difficult to model with current multidimensional database technology. In this context, data mining covers a broader spectrum than OLAP with respect to data mining functionality and the complexity of the data handled.

Because data mining involves more automated and deeper analysis than OLAP, it is expected to have broader applications. Data mining can help business managers find and reach more suitable customers, as well as gain critical business insights that may help drive market share and raise profits. In addition, data mining can help managers understand customer group characteristics and develop optimal pricing strategies accordingly. It can correct item bundling based not on intuition but on actual item groups derived from customer purchase patterns, reduce promotional spending, and at the same time increase the overall net effectiveness of promotions.

The data mining field has conducted substantial research regarding mining on various data types, including relational data, data from data warehouses, transaction data, time-series data, spatial data, text data, and flat files. Multidimensional data mining (also known as exploratory multidimensional data mining, online analytical mining, or OLAM) integrates OLAP with data mining to uncover knowledge in multidimensional databases. Among the many different paradigms and architectures of data mining systems, multidimensional data mining is particularly important for the following reasons:

■ High quality of data in data warehouses: Most data mining tools need to work on integrated, consistent, and cleaned data, which requires costly data cleaning, data integration, and data transformation as preprocessing steps. A data warehouse constructed by such preprocessing serves as a valuable source of high-quality data for OLAP as well as for data mining. Notice that data mining may serve as a valuable tool for data cleaning and data integration as well.

■ Available information processing infrastructure surrounding data warehouses: Comprehensive information processing and data analysis infrastructures have been or will be systematically constructed surrounding data warehouses, which include accessing, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web accessing and service facilities, and reporting and OLAP analysis tools. It is prudent to make the best use of the available infrastructures rather than constructing everything from scratch.

■ OLAP-based exploration of multidimensional data: Effective data mining needs exploratory data analysis. A user will often want to traverse through a database, select portions of relevant data, analyze them at different granularities, and present knowledge/results in different forms. Multidimensional data mining provides facilities for mining on different subsets of data and at varying levels of abstraction—by drilling, pivoting, filtering, dicing, and slicing on a data cube and/or intermediate data mining results. This, together with data/knowledge visualization tools, greatly enhances the power and flexibility of data mining.

■ Online selection of data mining functions: Users may not always know the specific kinds of knowledge they want to mine. By integrating OLAP with various data mining functions, multidimensional data mining provides users with the flexibility to select desired data mining functions and swap data mining tasks dynamically.

Rather than asking a data mining system to generate patterns and knowledge automatically, a user will often need to interact with the system to perform exploratory data analysis. OLAP sets a good example for interactive data analysis and provides the necessary preparations for exploratory data mining. Consider the discovery of association patterns, for example. Instead of mining associations at a primitive (i.e., low) data level among transactions, users should be allowed to specify roll-up operations along any dimension.

For example, a user may want to roll up on the item dimension to go from viewing the data for particular TV sets that were purchased to viewing the brands of these TVs (e.g., SONY or Toshiba). Users may also navigate from the transaction level to the customer or customer-type level in the search for interesting associations. Such an OLAP data mining style is characteristic of. In our study of the principles of data mining in this book, we place particular emphasis on, that is, on the integration of data mining and OLAP technology.

Key Terms

■ A **data warehouse** is a subject-oriented, integrated, time-variant, and nonvolatile data collection organized in support of management decision making. Several factors distinguish data warehouses from operational databases. Because the two systems provide quite different functionalities and require different kinds of data, it is necessary to maintain data warehouses separately from operational databases.

■ Data warehouses often adopt a **three-tier architecture**. The bottom tier is a warehouse database server, which is typically a relational database system. The middle tier is an OLAP server, and the top tier is a client that contains query and reporting tools.

■ A data warehouse contains **back-end tools and utilities** for populating and refreshing the warehouse. These cover data extraction, data cleaning, data transformation, loading, refreshing, and warehouse management.

■ Data warehouse **metadata** are data defining the warehouse objects. A metadata repository provides details regarding the warehouse structure, data history, the algorithms used for summarization, mappings from the source data to the warehouse form, system performance, and business terms and issues.

■ A **multidimensional data model** is typically used for the design of corporate data warehouses and departmental data marts. Such a model can adopt a star schema, snowflake schema, or fact constellation schema. The core of the multidimensional model is the **data cube**, which consists of a large set of facts (or measures) and a number of dimensions. Dimensions are the entities or perspectives with respect to which an organization wants to keep records and are hierarchical in nature.

■ A **data cube** consists of a **lattice of cuboids**, each corresponding to a different degree of summarization of the given multidimensional data.

■ **Concept hierarchies** organize the values of attributes or dimensions into gradual abstraction levels. They are useful in mining at multiple abstraction levels.

■ **Online analytical processing** can be performed in data warehouses/marts using the multidimensional data model. Typical OLAP operations include roll-up, and drill-(down, across, through), slice-and-dice, and pivot (rotate), as well as statistical operations such as ranking and computing moving averages and growth rates. OLAP operations can be implemented efficiently using the data cube structure.

■ Data warehouses are used for information processing (querying and reporting), analytical processing (which allows users to navigate through summarized and detailed data by OLAP operations), and data mining (which supports knowledge discovery). OLAP-based data mining is referred to as **multidimensional data mining** (also known as exploratory multidimensional data mining, online analytical mining, or OLAM). It emphasizes the interactive and exploratory nature of data mining.

■ OLAP servers may adopt a **relational OLAP (ROLAP), a multidimensional OLAP (MOLAP), or a hybrid OLAP (HOLAP) implementation**. A ROLAP server uses an extended relational DBMS that maps OLAP operations on multidimensional data to standard relational operations. A MOLAP server maps multidimensional data views directly to array structures. A HOLAP server combines ROLAP and MOLAP. For example, it may use ROLAP for historic data while maintaining frequently accessed data in a separate MOLAP store.

■ **Full materialization** refers to the computation of all of the cuboids in the lattice defining a data cube. It typically requires an excessive amount of storage space, particularly as the number of dimensions and size of associated concept hierarchies grow. This problem is known as the curse of dimensionality. Alternatively, partial materialization is the selective computation of a subset of the cuboids or subcubes in the lattice. For example, an iceberg cube is a data cube that stores only those cube cells that have an aggregate value (e.g., count) above some minimum support threshold.

■ OLAP query processing can be made more efficient with the use of indexing techniques. In **bitmap indexing**, each attribute has its own bitmap index table. Bitmap indexing reduces join, aggregation, and comparison operations to bit arithmetic. Join indexing registers the joinable rows of two or more relations from a relational database, reducing the overall cost of OLAP join operations. Bitmapped join indexing, which combines the bitmap and join index methods, can be used to further speed up OLAP query processing.

■ **Data generalization** is a process that abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels. Data generalization approaches include data cube-based data aggregation and attribute-oriented induction. Concept description is the most basic form of descriptive data mining. It describes a given set of task-relevant data in a concise and summative manner, presenting interesting general properties of the data. Concept (or class) description consists of characterization and comparison (or discrimination). The former summarizes and describes a data collection, called the target class, whereas the latter summarizes and distinguishes one data collection, called the target class, from other data collection(s), collectively called the contrasting class(es).

■ **Concept characterization** can be implemented using data cube (OLAP-based) approaches and the attribute-oriented induction approach. These are attribute- or dimension-based generalization approaches. The attribute-oriented induction approach consists of the following techniques: data focusing, data generalization by attribute removal or attribute generalization, count and aggregate value accumulation, attribute generalization control, and generalization data visualization.

■ **Concept comparison** can be performed using the attribute-oriented induction or data cube approaches in a manner similar to concept characterization. Generalized tuples from the target and contrasting classes can be quantitatively compared and contrasted.