Data Mining:
Concepts and Techniques

Chapter I: Introduction to Data Mining

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed. The proliferation of database management systems has also contributed to recent massive gathering of all sorts of information. Today, we have far more information than we can handle: from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Information retrieval is simply not enough anymore for decision-making. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the "essence" of information stored, and the discovery of patterns in raw data.

Data mining is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. It has been defined as:

The automated analysis of large or complex data sets in order to discover significant patterns or trends that would otherwise go unrecognised.

The key elements that make data mining tools a distinct form of software are:

Automated analysis

Data mining automates the process of sifting through historical data in order to discover new information. This is one of the main differences between data mining and statistics, where a model is usually devised by a statistician to deal with a specific analysis problem. It also distinguishes data mining from expert systems, where the model is built by a knowledge engineer from rules extracted from the experience of an expert.

The emphasis on automated discovery also separates data mining from OLAP and simpler query and reporting tools, which are used to verify hypotheses formulated by the user. Data mining does not rely on a user to define a specific query, merely to formulate a goal - such as the identification of fraudulent claims.

Large or complex data sets

One of the attractions of data mining is that it makes it possible to analyse very large data sets in a reasonable time scale. Data mining is also suitable for complex problems involving relatively small amounts of data but where there are many fields or variables to analyse. However, for small, relatively simple data analysis problems there may be simpler, cheaper and more effective solutions.

Discovering significant patterns or trends that would otherwise go unrecognised

The goal of data mining is to unearth relationships in data that may provide useful insights.

Data mining tools can sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions, performance bottlenecks in a network system and identifying anomalous data that could represent data entry keying errors. The ultimate significance of these patterns will be assessed by a domain expert - a marketing manager or network supervisor - so the results must be presented in a way that human experts can understand.

Data mining tools can also automate the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms to enhance the value of existing information resources, and can be implemented on new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing systems, they can analyse massive databases to deliver answers to questions such as:

"Which clients are most likely to respond to my next promotional mailing, and why?"

Data mining is ready for application because it is supported by three technologies that are now sufficiently mature:

- Massive data collection

- Powerful multiprocessor computers

- Data mining algorithms

Commercial databases are growing at unprecedented rates, especially in the retail sector. The accompanying need for improved computational engines can now be met in a cost-effective manner with parallel multiprocessor computer technology. Data mining algorithms embody techniques that have existed for at least 10 years, but have only recently been implemented as mature, reliable, understandable tools that consistently outperform older statistical methods.

The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. Today, the maturity of these techniques, coupled with high-performance relational database engines and broad data integration efforts, make these technologies practical for current data warehouse environments.

The key to understanding the different facets of data mining is to distinguish between data mining applications, operations, techniques and algorithms.

Applications	Database marketing customer segmentation customer retention fraud detection credit checking web site analysis
Operations	Classification and prediction clustering association analysis forecasting
Techniques	Neural networks decision trees K-nearest neighbour algorithms naive Bayesian cluster analysis

What kind of information are we collecting?

We have been collecting a myriad of data, from simple numerical measurements and text documents, to more complex information such as spatial data, multimedia channels, and hypertext documents. Here is a non-exclusive list of a variety of information collected in digital form in databases and in flat files.

- **Business transactions**: Every transaction in the business industry is (often) "memorized" for perpetuity.Such transactions are usually time related and can be inter-business deals such as purchases, exchanges, banking, stock, etc., or intra-business operations such as management of in-house wares and assets. Large department stores, for example, thanks to the widespread use of bar codes, store millions of transactions daily representing often terabytes of data. Storage space is not the major problem, as the price of hard disks is continuously dropping, but the effective use of the data in a reasonable time frame for competitive decision-making is definitely the most important problem to solve for businesses that struggle to survive in a highly competitive world.
- **Scientific data**: Whether in a Swiss nuclear accelerator laboratory counting particles, in the Canadian forest studying readings from a grizzly bear radio collar, on a South Pole iceberg gathering data about oceanic activity, or in an American university investigating human psychology, our society is amassing colossal amounts of scientific data that need to be analyzed. Unfortunately, we can capture and store more new data faster than we can analyze the old data already accumulated.
- **Medical and personal data**: From government census to personnel and customer files, very large collections of information are continuously gathered about individuals and groups. Governments, companies and organizations such as hospitals, are stockpiling very important quantities of personal data to help them manage human resources, better understand a market, or simply assist clientele. Regardless of the privacy issues this type of data often reveals, this information is collected, used and even shared. When correlated with other data this information can shed light on customer behaviour and the like.
- **Surveillance video and pictures**: With the amazing collapse of video camera prices, video cameras are becoming ubiquitous. Video tapes from surveillance cameras are usually recycled and thus the content is lost. However, there is a tendency today to store the tapes and even digitize them for future use and analysis.
- **Satellite sensing**:There is a countless number of satellites around the globe: some are geo-stationary above a region, and some are orbiting around the Earth, but all are sending a non-stop stream of data to the surface. NASA, which controls a large number of satellites, receives more data every second than what all NASA researchers and engineers can cope with. Many satellite pictures and data are made public as soon as they are received in the hopes that other researchers can analyze them.
- **Games**: Our society is collecting a tremendous amount of data and statistics about games, players and athletes. From hockey scores, basketball passes and car-racing lapses, to swimming times, boxers pushes and chess positions, all the data are stored. Commentators and journalists are using

this information for reporting, but trainers and athletes would want to exploit this data to improve performance and better understand opponents.

- **Digital media**: The proliferation of cheap scanners, desktop video cameras and digital cameras is one of the causes of the explosion in digital media repositories. In addition, many radio stations, television channels and film studios are digitizing their audio and video collections to improve the management of their multimedia assets. Associations such as the NHL and the NBA have already started converting their huge game collection into digital forms.
- **CAD and Software engineering data**: There are a multitude of Computer Assisted Design (CAD) systems for architects to design buildings or engineers to conceive system components or circuits. These systems are generating a tremendous amount of data. Moreover, software engineering is a source of considerable similar data with code, function libraries, objects, etc., which need powerful tools for management and maintenance.
- **Virtual Worlds**: There are many applications making use of three-dimensional virtual spaces. These spaces and the objects they contain are described with special languages such as VRML. Ideally, these virtual spaces are described in such a way that they can share objects and places. There is a remarkable amount of virtual reality object and space repositories available. Management of these repositories as well as content-based search and retrieval from these repositories are still research issues, while the size of the collections continues to grow.
- **Text reports and memos (e-mail messages)**: Most of the communications within and between companies or research organizations or even private people, are based on reports and memos in textual forms often exchanged by e-mail. These messages are regularly stored in digital form for future use and reference creating formidable digital libraries.
- **The World Wide Web repositories**: Since the inception of the World Wide Web in 1993, documents of all sorts of formats, content and description have been collected and inter-connected with hyperlinks making it the largest repository of data ever built. Despite its dynamic and unstructured nature, its heterogeneous characteristic, and its very often redundancy and inconsistency, the World Wide Web is the most important data collection regularly used for reference because of the broad variety of topics covered and the infinite contributions of resources and publishers. Many believe that the World Wide Web will become the compilation of human knowledge.

What are Data Mining and Knowledge Discovery?

With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important, if not necessary, to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision-making.

Data Mining, also popularly known as *Knowledge Discovery in Databases* (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data

mining is actually part of the knowledge discovery process. The following figure (Figure 1.1) shows data mining as a step in an iterative knowledge discovery process.

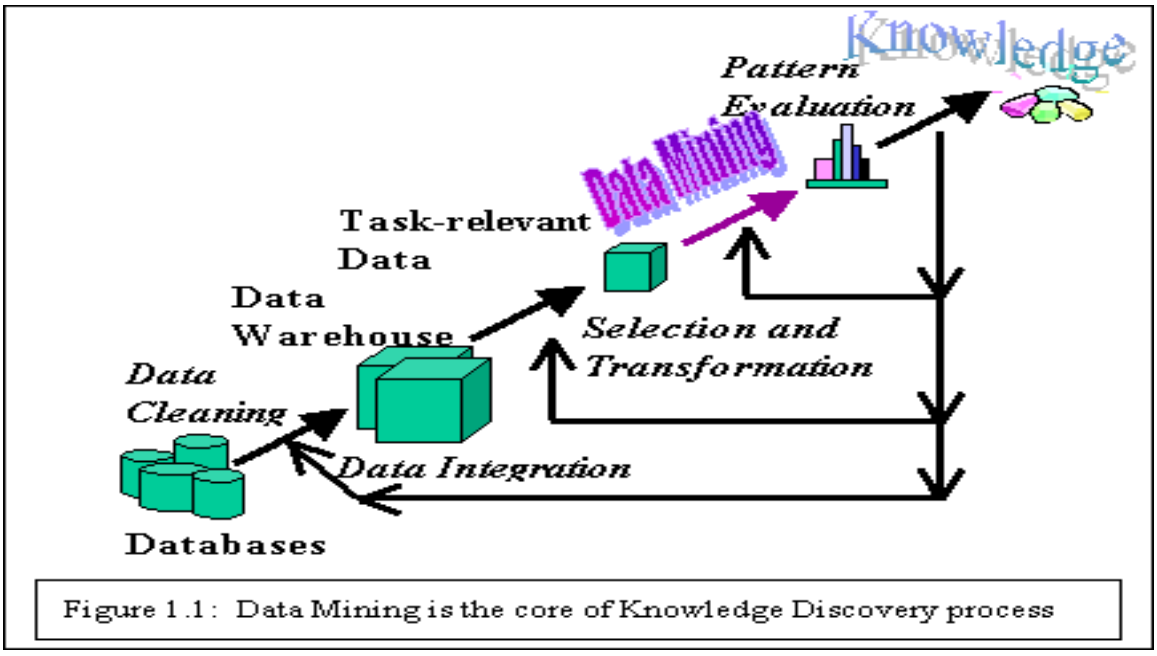


Figure 1.1: Data Mining is the core of Knowledge Discovery process

The Knowledge Discovery in Databases process comprises of a few steps leading from raw data collections to some form of new knowledge. The iterative process consists of the following steps:

- **Data cleaning**: also known as data cleansing, it is a phase in which noise data and irrelevant data are removed from the collection.
- **Data integration**: at this stage, multiple data sources, often heterogeneous, may be combined in a common source.
- **Data selection**: at this step, the data relevant to the analysis is decided on and retrieved from the data collection.
- **Data transformation**: also known as data consolidation, it is a phase in which the selected data is transformed into forms appropriate for the mining procedure.
- **Data mining**: it is the crucial step in which clever techniques are applied to extract patterns potentially useful.
- **Pattern evaluation**: in this step, strictly interesting patterns representing knowledge are identified based on given measures.
- **Knowledge representation**: is the final phase in which the discovered knowledge is visually represented to the user. This essential step uses visualization techniques to help users understand and interpret the data mining results.

It is common to combine some of these steps together. For instance, *data cleaning* and *data integration* can be performed together as a pre-processing phase to generate a data warehouse. *Data selection* and *data transformation* can also be combined where the consolidation of the data is the result of the selection, or, as for the case of data warehouses, the selection is done on transformed data.

The KDD is an iterative process. Once the discovered knowledge is presented to the user, the evaluation measures can be enhanced, the mining can be further refined, new data can be selected or further transformed, or new data sources can be integrated, in order to get different, more appropriate results.

Data mining derives its name from the similarities between searching for valuable information in a large database and mining rocks for a vein of valuable ore. Both imply either sifting through a large amount of material or ingeniously probing the material to exactly pinpoint where the values reside. It is, however, a misnomer, since mining for gold in rocks is usually called "gold mining" and not "rock mining", thus by analogy, data mining should have been called "knowledge mining" instead. Nevertheless, data mining became the accepted customary term, and very rapidly a trend that even overshadowed more general terms such as knowledge discovery in databases (KDD) that describe a more complete process. Other similar terms referring to data mining are: data dredging, knowledge extraction and pattern discovery.

What kind of Data can be mined?

In principle, data mining is not specific to one type of media or data. Data mining should be applicable to any kind of information repository. However, algorithms and approaches may differ when applied to different types of data. Indeed, the challenges presented by different types of data vary significantly. Data mining is being put into use and studied for databases, including relational databases, object-relational databases and object-oriented databases, data warehouses, transactional databases, unstructured and semi-structured repositories such as the World Wide Web, advanced databases such as spatial databases, multimedia databases, time-series databases and textual databases, and even flat files. Here are some examples in more detail:

- **Flat files**: Flat files are actually the most common data source for data mining algorithms, especially at the research level. Flat files are simple data files in text or binary format with a structure known by the data mining algorithm to be applied. The data in these files can be

transactions, time-series data, scientific measurements, etc.

- **Relational Databases**: Briefly, a relational database consists of a set of tables containing either values of entity attributes, or values of attributes from entity relationships. Tables have columns and rows, where columns represent attributes and rows represent tuples. A tuple in a relational table corresponds to either an object or a relationship between objects and is identified by a set of attribute values representing a unique key. In Figure 1.2 we present some relations *Customer*, *Items*, and *Borrow* representing business activity in a fictitious video store OurVideoStore. These relations are just a subset of what could be a database for the video store and is given as an example.

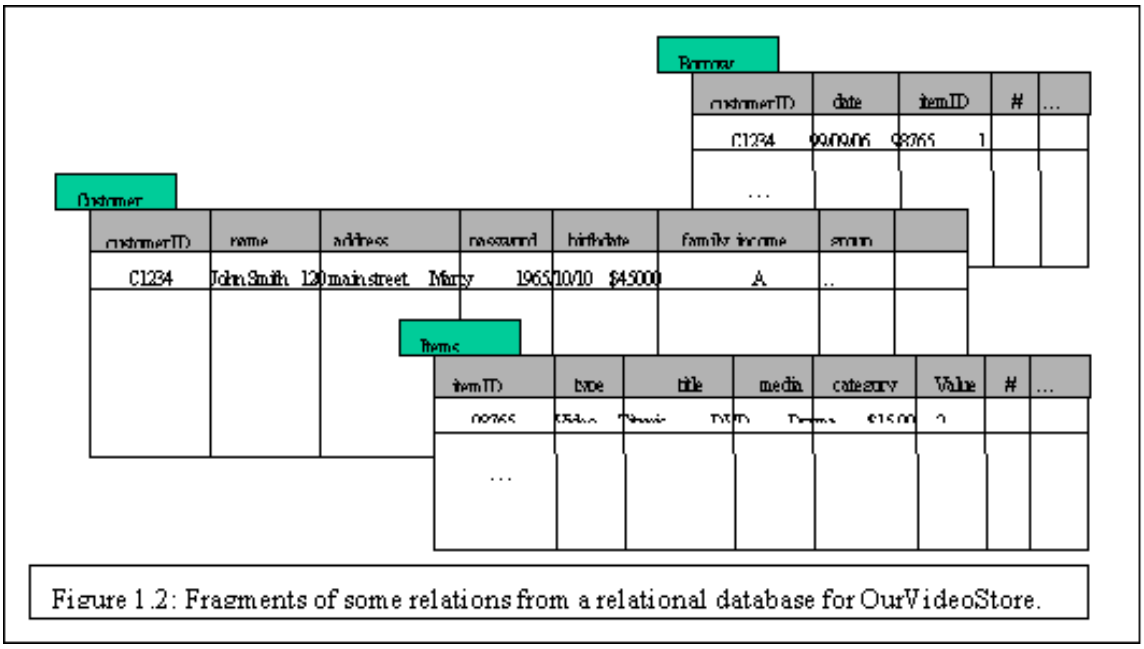


Figure 1.2: Fragments of some relations from a relational database for OurVideoStore.

The most commonly used query language for relational database is SQL, which allows retrieval and manipulation of the data stored in the tables, as well as the calculation of aggregate functions such as average, sum, min, max and count. For instance, an SQL query to select the videos grouped by category would be:

SELECT count(*) FROM Items WHERE type=video GROUP BY category.

Data mining algorithms using relational databases can be more versatile than data mining

algorithms specifically written for flat files, since they can take advantage of the structure inherent to relational databases. While data mining can benefit from SQL for data selection, transformation and consolidation, it goes beyond what SQL could provide, such as predicting, comparing, detecting deviations, etc.

- **Data Warehouses**: A data warehouse as a storehouse, is a repository of data collected from multiple data sources (often heterogeneous) and is intended to be used as a whole under the same unified schema. A data warehouse gives the option to analyze data from different sources under the same roof. Let us suppose that OurVideoStore becomes a franchise in North America. Many video stores belonging to OurVideoStore company may have different databases and different structures. If the executive of the company wants to access the data from all stores for strategic decision-making, future direction, marketing, etc., it would be more appropriate to store all the data in one site with a homogeneous structure that allows interactive analysis. In other words, data from the different stores would be loaded, cleaned, transformed and integrated together. To facilitate decision-making and multi-dimensional views, data warehouses are usually modeled by a multi-dimensional data structure. Figure 1.3 shows an example of a three dimensional subset of a data cube structure used for OurVideoStore data warehouse.

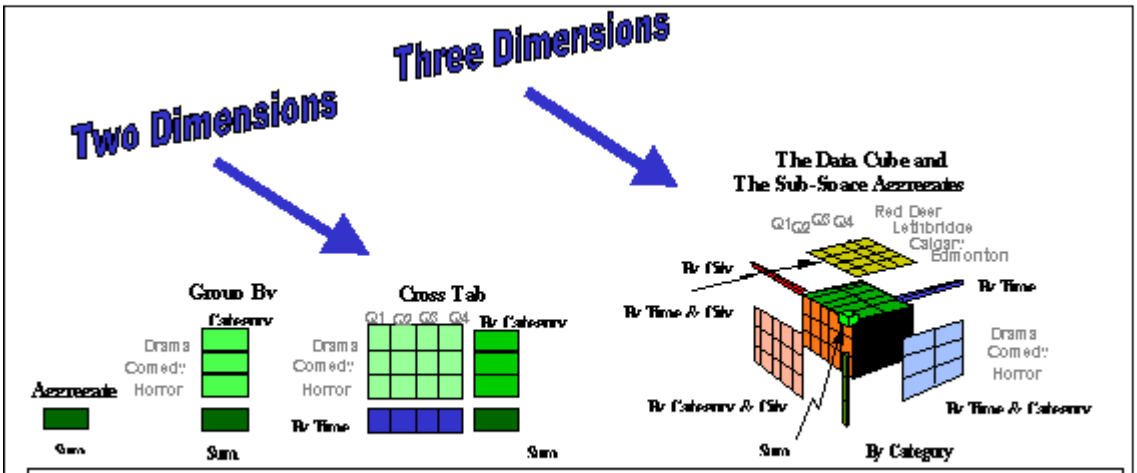


Figure 1.3: A multi-dimensional data cube structure commonly used in data for data warehousing.

The figure shows summarized rentals grouped by film categories, then a cross table of summarized rentals by film categories and time (in quarters). The data cube gives the summarized rentals along three dimensions: category, time, and city. A cube contains cells that store values of some aggregate measures (in this case rental counts), and special cells that store summations along dimensions. Each dimension of the data cube contains a hierarchy of values for one attribute.

Because of their structure, the pre-computed summarized data they contain and the hierarchical

attribute values of their dimensions, data cubes are well suited for fast interactive querying and analysis of data at different conceptual levels, known as On-Line Analytical Processing (OLAP). OLAP operations allow the navigation of data at different levels of abstraction, such as drill-down, roll-up, slice, dice, etc. Figure 1.4 illustrates the drill-down (on the time dimension) and roll-up (on the location dimension) operations.

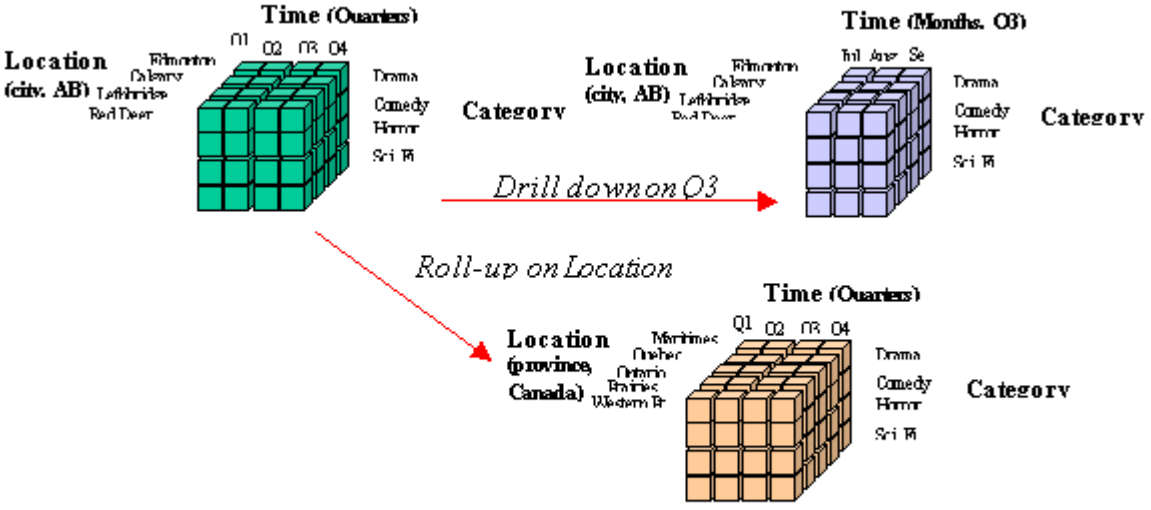


Figure 1.4: Summarized data from *OurVideoStore* before and after *drill-down* and *roll-up* operations.

- **Transaction Databases**: A transaction database is a set of records representing transactions, each with a time stamp, an identifier and a set of items. Associated with the transaction files could also be descriptive data for the items. For example, in the case of the video store, the rentals table such as shown in Figure 1.5, represents the transaction database. Each record is a rental contract with a customer identifier, a date, and the list of items rented (i.e. video tapes, games, VCR, etc.). Since relational databases do not allow nested tables (i.e. a set as attribute value), transactions are usually stored in flat files or stored in two normalized transaction tables, one for the transactions and one for the transaction items. One typical data mining analysis on such data is the so-called market basket analysis or association rules in which associations between items occurring together or in sequence are studied.

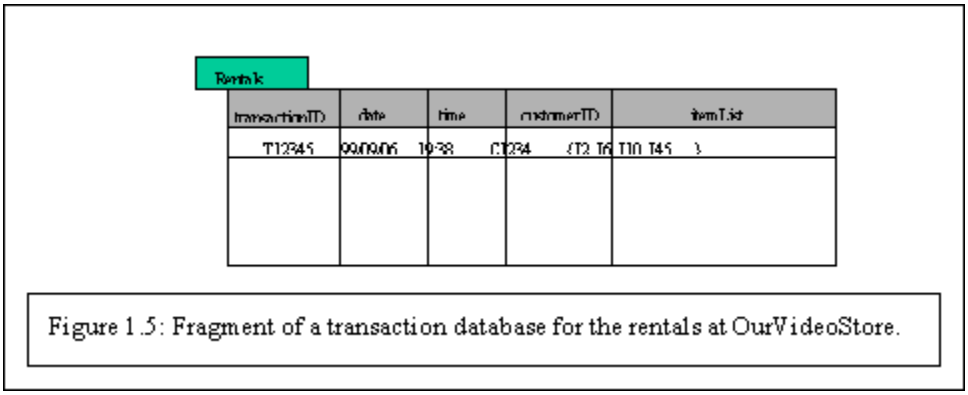


Figure 1.5: Fragment of a transaction database for the rentals at OurVideoStore.

- **Multimedia Databases**: Multimedia databases include video, images, audio and text media. They can be stored on extended object-relational or object-oriented databases, or simply on a file system. Multimedia is characterized by its high dimensionality, which makes data mining even more challenging. Data mining from multimedia repositories may require computer vision, computer graphics, image interpretation, and natural language processing methodologies.

- **Spatial Databases**: Spatial databases are databases that, in addition to usual data, store geographical information like maps, and global or regional positioning. Such spatial databases present new challenges to data mining algorithms.

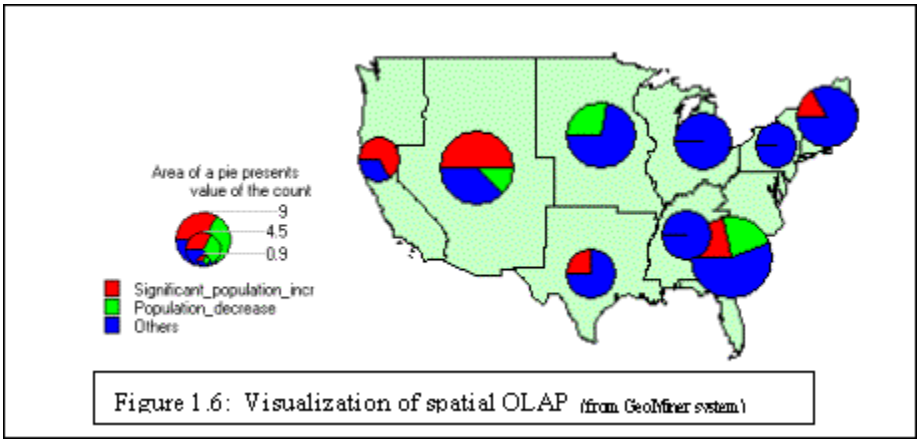


Figure 1.6: Visualization of spatial OLAP (from GeoMiner system)

- **Time-Series Databases**: Time-series databases contain time related data such stock market data or logged activities. These databases usually have a continuous flow of new data coming in, which

sometimes causes the need for a challenging real time analysis. Data mining in such databases commonly includes the study of trends and correlations between evolutions of different variables, as well as the prediction of trends and movements of the variables in time. Figure 1.7 shows some examples of time-series data.

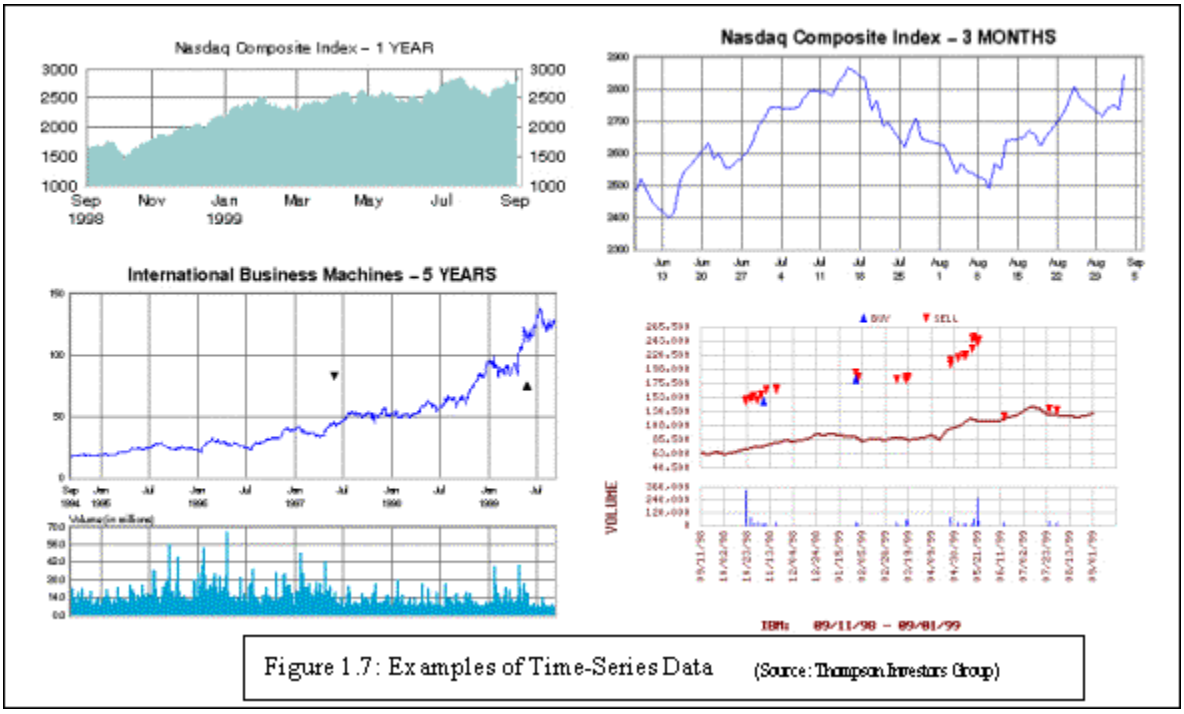


Figure 1.7: Examples of Time-Series Data (Source: Thompson Investors Group)

- **World Wide Web**: The World Wide Web is the most heterogeneous and dynamic repository available. A very large number of authors and publishers are continuously contributing to its growth and metamorphosis, and a massive number of users are accessing its resources daily. Data in the World Wide Web is organized in inter-connected documents. These documents can be text, audio, video, raw data, and even applications. Conceptually, the World Wide Web is comprised of three major components: The content of the Web, which encompasses documents available; the structure of the Web, which covers the hyperlinks and the relationships between documents; and the usage of the web, describing how and when the resources are accessed. A fourth dimension can be added relating the dynamic nature or evolution of the documents. Data mining in the World Wide Web, or web mining, tries to address all these issues and is often divided into web content mining, web structure mining and web usage mining.

What can be discovered?

The kinds of patterns that can be discovered depend upon the data mining tasks employed. By and large, there are two types of data mining tasks: *descriptive data mining* tasks that describe the general properties of the existing data, and *predictive data mining* tasks that attempt to do predictions based on inference on available data.

The data mining functionalities and the variety of knowledge they discover are briefly presented in the following list:

- **Characterization**: Data characterization is a summarization of general features of objects in a target class, and produces what is called *characteristic rules*. The data relevant to a user-specified class are normally retrieved by a database query and run through a summarization module to extract the essence of the data at different levels of abstractions. For example, one may want to characterize the OurVideoStore customers who regularly rent more than 30 movies a year. With concept hierarchies on the attributes describing the target class, the *attribute-oriented induction* method can be used, for example, to carry out data summarization. Note that with a data cube containing summarization of data, simple OLAP operations fit the purpose of data characterization.
- **Discrimination**: Data discrimination produces what are called *discriminant rules* and is basically the comparison of the general features of objects between two classes referred to as the *target class* and the *contrasting class*. For example, one may want to compare the general characteristics of the customers who rented more than 30 movies in the last year with those whose rental account is lower than 5. The techniques used for data discrimination are very similar to the techniques used for data characterization with the exception that data discrimination results include comparative measures.
- **Association analysis**: Association analysis is the discovery of what are commonly called *association rules*. It studies the frequency of items occurring together in transactional databases, and based on a threshold called *support*, identifies the frequent item sets. Another threshold, *confidence*, which is the conditional probability than an item appears in a transaction when another item appears, is used to pinpoint association rules. Association analysis is commonly used for market basket analysis. For example, it could be useful for the OurVideoStore manager to know what movies are often rented together or if there is a relationship between renting a certain type of movies and buying popcorn or pop. The discovered association rules are of the form: P -> Q [s,c], where P and Q are conjunctions of attribute value-pairs, and s (for support) is the probability that P and Q appear together in a transaction and c (for confidence) is the conditional probability that Q appears in a transaction when P is present. For example, the hypothetic association rule:
  RentType(X, "game") AND Age(X, "13-19") -> Buys(X, "pop") [s=2% ,c=55%]
  would indicate that 2% of the transactions considered are of customers aged between 13 and 19 who are renting a game and buying a pop, and that there is a certainty of 55% that teenage customers who rent a game also buy pop.
- **Classification**: Classification analysis is the organization of data in given classes. Also known as

supervised classification, the classification uses given class labels to order the objects in the data collection. Classification approaches normally use a *training set* where all objects are already associated with known class labels. The classification algorithm learns from the training set and builds a model. The model is used to classify new objects. For example, after starting a credit policy, the OurVideoStore managers could analyze the customers behaviours vis-à-vis their credit, and label accordingly the customers who received credits with three possible labels "safe", "risky" and "very risky". The classification analysis would generate a model that could be used to either accept or reject credit requests in the future.

- **Prediction**: Prediction has attracted considerable attention given the potential implications of successful forecasting in a business context. There are two major types of predictions: one can either try to predict some unavailable data values or pending trends, or predict a class label for some data. The latter is tied to classification. Once a classification model is built based on a training set, the class label of an object can be foreseen based on the attribute values of the object and the attribute values of the classes. Prediction is however more often referred to the forecast of missing numerical values, or increase/ decrease trends in time related data. The major idea is to use a large number of past values to consider probable future values.
- **Clustering**: Similar to classification, clustering is the organization of data in classes. However, unlike classification, in clustering, class labels are unknown and it is up to the clustering algorithm to discover acceptable classes. Clustering is also called *unsupervised classification*, because the classification is not dictated by given class labels. There are many clustering approaches all based on the principle of maximizing the similarity between objects in a same class (*intra-class similarity*) and minimizing the similarity between objects of different classes (*inter-class similarity*).
- **Outlier analysis**: Outliers are data elements that cannot be grouped in a given class or cluster. Also known as *exceptions* or *surprises*, they are often very important to identify. While outliers can be considered noise and discarded in some applications, they can reveal important knowledge in other domains, and thus can be very significant and their analysis valuable.
- **Evolution and deviation analysis**: Evolution and deviation analysis pertain to the study of time related data that changes in time. Evolution analysis models evolutionary trends in data, which consent to characterizing, comparing, classifying or clustering of time related data. Deviation analysis, on the other hand, considers differences between measured values and expected values, and attempts to find the cause of the deviations from the anticipated values.

It is common that users do not have a clear idea of the kind of patterns they can discover or need to discover from the data at hand. It is therefore important to have a versatile and inclusive data mining system that allows the discovery of different kinds of knowledge and at different levels of abstraction. This also makes interactivity an important attribute of a data mining system.

Is all that is discovered interesting and useful?

Data mining allows the discovery of knowledge potentially useful and unknown. Whether the knowledge discovered is new, useful or interesting, is very subjective and depends upon the application and the user. It is certain that data mining can generate, or discover, a very large number of patterns or rules. In some cases the number of rules can reach the millions. One can even think of a meta-mining phase to mine the oversized data mining results. To reduce the number of patterns or rules discovered that have a high probability to be non-interesting, one has to put a measurement on the patterns. However, this raises the problem of completeness. The user would want to discover ***all*** rules or patterns, ***but only*** those that are ***interesting***. The measurement of how interesting a discovery is, often called ***interestingness***, can be based on quantifiable objective elements such as *validity* of the patterns when tested on new data with some degree of *certainty*, or on some subjective depictions such as *understandability* of the patterns, *novelty* of the patterns, or *usefulness*.

Discovered patterns can also be found interesting if they confirm or validate a hypothesis sought to be confirmed or unexpectedly contradict a common belief. This brings the issue of describing what is interesting to discover, such as meta-rule guided discovery that describes forms of rules before the discovery process, and interestingness refinement languages that interactively query the results for interesting patterns after the discovery phase. Typically, measurements for interestingness are based on thresholds set by the user. These thresholds define the completeness of the patterns discovered.

Identifying and measuring the interestingness of patterns and rules discovered, or to be discovered, is essential for the evaluation of the mined knowledge and the KDD process as a whole. While some concrete measurements exist, assessing the interestingness of discovered knowledge is still an important research issue.

How do we categorize data mining systems?

There are many data mining systems available or being developed. Some are specialized systems dedicated to a given data source or are confined to limited data mining functionalities, other are more versatile and comprehensive. Data mining systems can be categorized according to various criteria among other classification are the following:

- **Classification according to the type of data source mined**: this classification categorizes data mining systems according to the type of data handled such as spatial data, multimedia data, time-series data, text data, World Wide Web, etc.

- **Classification according to the data model drawn on**: this classification categorizes data mining systems based on the data model involved such as relational database, object-oriented database, data warehouse, transactional, etc.
- **Classification according to the king of knowledge discovered**: this classification categorizes data mining systems based on the kind of knowledge discovered or data mining functionalities, such as characterization, discrimination, association, classification, clustering, etc. Some systems tend to be comprehensive systems offering several data mining functionalities together.
- **Classification according to mining techniques used**: Data mining systems employ and provide different techniques. This classification categorizes data mining systems according to the data analysis approach used such as machine learning, neural networks, genetic algorithms, statistics, visualization, database-oriented or data warehouse-oriented, etc. The classification can also take into account the degree of user interaction involved in the data mining process such as query-driven systems, interactive exploratory systems, or autonomous systems. A comprehensive system would provide a wide variety of data mining techniques to fit different situations and options, and offer different degrees of user interaction.

What are the issues in Data Mining?

Data mining algorithms embody techniques that have sometimes existed for many years, but have only lately been applied as reliable and scalable tools that time and again outperform older classical statistical methods. While data mining is still in its infancy, it is becoming a trend and ubiquitous. Before data mining develops into a conventional, mature and trusted discipline, many still pending issues have to be addressed. Some of these issues are addressed below. Note that these issues are not exclusive and are not ordered in any way.

Security and social issues: Security is an important issue with any data collection that is shared and/or is intended to be used for strategic decision-making. In addition, when data is collected for customer profiling, user behaviour understanding, correlating personal data with other information, etc., large amounts of sensitive and private information about individuals or companies is gathered and stored. This becomes controversial given the confidential nature of some of this data and the potential illegal access to the information. Moreover, data mining could disclose new implicit knowledge about individuals or groups that could be against privacy policies, especially if there is potential dissemination of discovered information. Another issue that arises from this concern is the appropriate use of data mining. Due to the value of data, databases of all sorts of content are regularly sold, and because of the competitive advantage that can be attained from implicit knowledge discovered, some important information could be withheld, while other information could be widely distributed and used without control.

User interface issues: The knowledge discovered by data mining tools is useful as long as it is interesting, and above all understandable by the user. Good data visualization eases the interpretation of data mining results, as well as helps users better understand their needs. Many data exploratory analysis tasks are significantly facilitated by the ability to see data in an appropriate visual presentation. There are many visualization ideas and proposals for effective data graphical presentation. However, there is still much research to accomplish in order to obtain good visualization tools for large datasets that could be used to display and manipulate mined knowledge.The major issues related to user interfaces and visualization are "screen real-estate", information rendering, and interaction. Interactivity with the data and data mining results is crucial since it provides means for the user to focus and refine the mining tasks, as well as to picture the discovered knowledge from different angles and at different conceptual levels.

Mining methodology issues: These issues pertain to the data mining approaches applied and their limitations. Topics such as versatility of the mining approaches, the diversity of data available, the dimensionality of the domain, the broad analysis needs (when known), the assessment of the knowledge discovered, the exploitation of background knowledge and metadata, the control and handling of noise in data, etc. are all examples that can dictate mining methodology choices. For instance, it is often desirable to have different data mining methods available since different approaches may perform differently depending upon the data at hand. Moreover, different approaches may suit and solve user's needs differently.

Most algorithms assume the data to be noise-free. This is of course a strong assumption. Most datasets contain exceptions, invalid or incomplete information, etc., which may complicate, if not obscure, the analysis process and in many cases compromise the accuracy of the results. As a consequence, data preprocessing (data cleaning and transformation) becomes vital. It is often seen as lost time, but data cleaning, as time-consuming and frustrating as it may be, is one of the most important phases in the knowledge discovery process. Data mining techniques should be able to handle noise in data or incomplete information.

More than the size of data, the size of the search space is even more decisive for data mining techniques. The size of the search space is often depending upon the number of dimensions in the domain space. The search space usually grows exponentially when the number of dimensions increases. This is known as the *curse of dimensionality*. This "curse" affects so badly the performance of some data mining approaches that it is becoming one of the most urgent issues to solve.

Performance issues: Many artificial intelligence and statistical methods exist for data analysis and interpretation. However, these methods were often not designed for the very large data sets data mining is dealing with today. Terabyte sizes are common. This raises the issues of scalability and efficiency of the data mining methods when processing considerably large data. Algorithms with exponential and even medium-order polynomial complexity cannot be of practical use for data mining. Linear algorithms are usually the norm. In same theme, sampling can be used for mining instead of the whole dataset. However,

concerns such as completeness and choice of samples may arise. Other topics in the issue of performance are *incremental updating*, and parallel programming. There is no doubt that parallelism can help solve the size problem if the dataset can be subdivided and the results can be merged later. Incremental updating is important for merging results from parallel mining, or updating data mining results when new data becomes available without having to re-analyze the complete dataset.

Data source issues: There are many issues related to the data sources, some are practical such as the diversity of data types, while others are philosophical like the data glut problem. We certainly have an excess of data since we already have more data than we can handle and we are still collecting data at an even higher rate. If the spread of database management systems has helped increase the gathering of information, the advent of data mining is certainly encouraging more data harvesting. The current practice is to collect as much data as possible now and process it, or try to process it, later. The concern is whether we are collecting the right data at the appropriate amount, whether we know what we want to do with it, and whether we distinguish between what data is important and what data is insignificant. Regarding the practical issues related to data sources, there is the subject of heterogeneous databases and the focus on diverse complex data types. We are storing different types of data in a variety of repositories. It is difficult to expect a data mining system to effectively and efficiently achieve good mining results on all kinds of data and sources. Different kinds of data and sources may require distinct algorithms and methodologies. Currently, there is a focus on relational databases and data warehouses, but other approaches need to be pioneered for other specific complex data types. A versatile data mining tool, for all sorts of data, may not be realistic. Moreover, the proliferation of heterogeneous data sources, at structural and semantic levels, poses important challenges not only to the database community but also to the data mining community.

DATA WAREHOUSE COMPONENTS & ARCHITECTURE

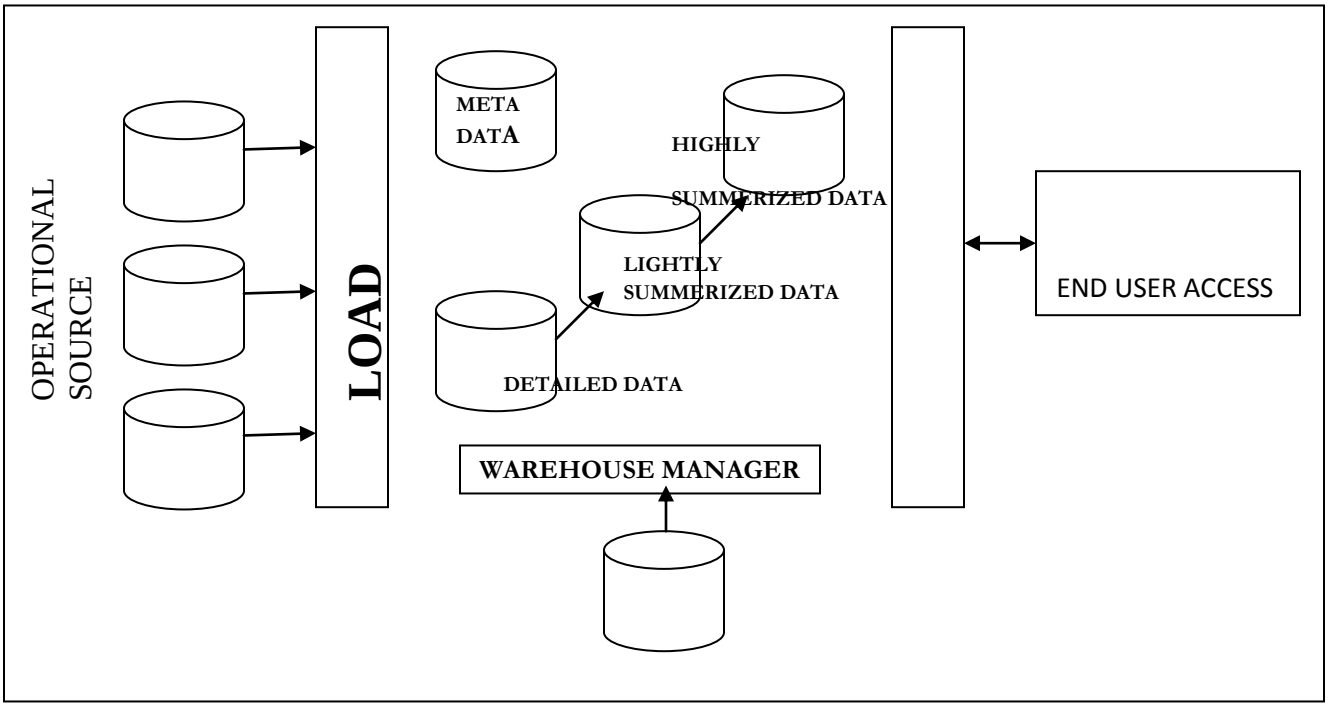
The data in a data warehouse comes from operational systems of the organization as well as from other external sources. These are collectively referred to as ***source systems***. The data ***extracted*** from source systems is stored in a area called ***data staging area***, where the data is cleaned, ***transformed***, combined, deduplicated to prepare the data for us in the data warehouse. The data staging area is generally a collection of machines where simple activities like sorting and sequential processing takes place. The data staging area does not provide any query or presentation services. As soon as a system provides query or presentation services, it is categorized as a ***presentation server***. A presentation server is the target machine on which the data is ***loaded*** from the data staging area organized and stored for direct querying by end users, report writers and other applications. The three different kinds of systems that are required for a data warehouse are:

1. Source Systems
2. Data Staging Area
3. Presentation servers

The data travels from source systems to presentation servers via the data staging area. The entire process is popularly known as ETL (extract, transform, and load) or ETT (extract, transform, and transfer). Oracle's

ETL tool is called Oracle Warehouse Builder (OWB) and MS SQL Server's ETL tool is called Data Transformation Services (DTS).

A typical architecture of a data warehouse is shown below:



Each component and the tasks performed by them are explained below:

1. OPERATIONAL DATA

The sources of data for the data warehouse is supplied from:

(i) The data from the mainframe systems in the traditional network and hierarchical format.
(ii) Data can also come from the relational DBMS like Oracle, Informix.
(iii) In addition to these internal data, operational data also includes external data obtained from commercial databases and databases associated with supplier and customers.

2. LOAD MANAGER

The load manager performs all the operations associated with extraction and loading data into the data warehouse. These operations include simple transformations of the data to prepare the data for entry into the warehouse. The size and complexity of this component will vary between data warehouses and may be constructed using a combination of vendor data loading tools and custom built programs.

3. WAREHOUSE MANAGER

The warehouse manager performs all the operations associated with the management of data in the warehouse. This component is built using vendor data management tools and custom built programs. The operations performed by warehouse manager include:

(i) Analysis of data to ensure consistency
(ii) Transformation and merging the source data from temporary storage into data warehouse tables
(iii) Create indexes and views on the base table.
(iv) Denormalization
(v) Generation of aggregation
(vi) Backing up and archiving of data

In certain situations, the warehouse manager also generates query profiles to determine which indexes ands aggregations are appropriate.

4. QUERY MANAGER

The query manager performs all operations associated with management of user queries. This component is usually constructed using vendor end-user access tools, data warehousing monitoring tools, database facilities and custom built programs. The complexity of a query manager is determined by facilities provided by the end-user access tools and database.

5. DETAILED DATA

This area of the warehouse stores all the detailed data in the database schema. In most cases detailed data is not stored online but aggregated to the next level of details. However the detailed data is added regularly to the warehouse to supplement the aggregated data.

6. LIGHTLY AND HIGHLY SUMMERIZED DATA

The area of the data warehouse stores all the predefined lightly and highly summarized (aggregated) data generated by the warehouse manager. This area of the warehouse is transient as it will be subject to change on an ongoing basis in order to respond to the changing query profiles. The purpose of the summarized information is to speed up the query performance. The summarized data is updated continuously as new data is loaded into the warehouse.

7. ARCHIVE AND BACK UP DATA

This area of the warehouse stores detailed and summarized data for the purpose of archiving and back up. The data is transferred to storage archives such as magnetic tapes or optical disks.

8. META DATA

The data warehouse also stores all the Meta data (data about data) definitions used by all processes in the warehouse. It is used for variety of purposed including:

(i) The extraction and loading process – Meta data is used to map data sources to a common view of information within the warehouse.
(ii) The warehouse management process – Meta data is used to automate the production of summary tables.
(iii) As part of Query Management process Meta data is used to direct a query to the most appropriate data source.

The structure of Meta data will differ in each process, because the purpose is different. More about Meta

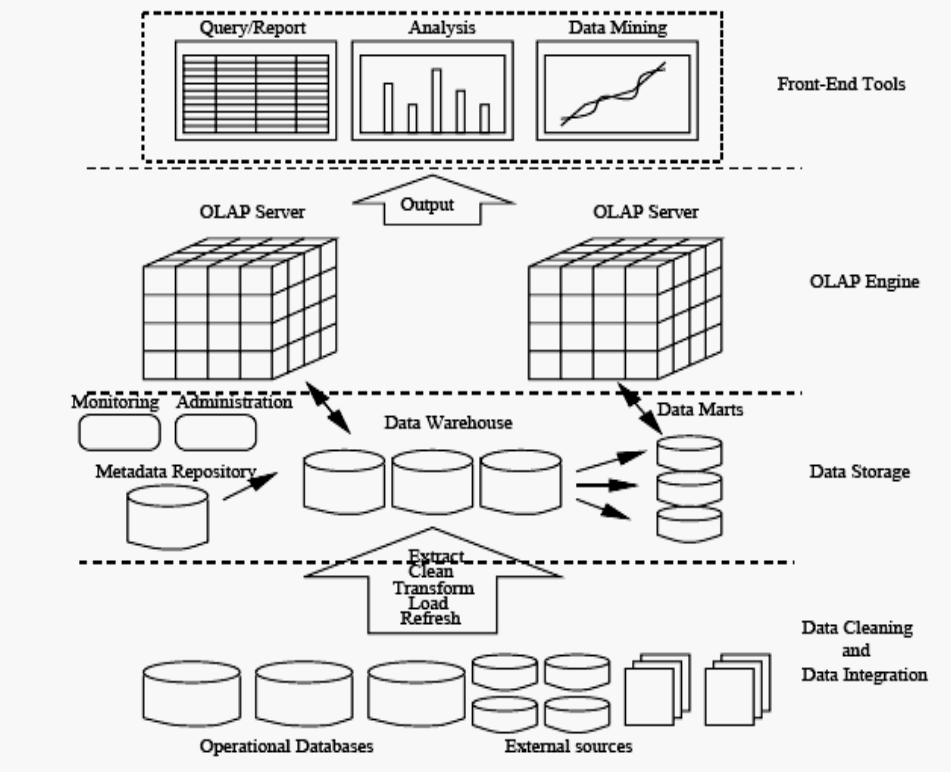
data will be discussed in the later Lecture Notes.

9. END-USER ACCESS TOOLS

The principal purpose of data warehouse is to provide information to the business managers for strategic decision-making. These users interact with the warehouse using end user access tools. The examples of some of the end user access tools can be:

(i) Reporting and Query Tools
(ii) Application Development Tools
(iii) Executive Information Systems Tools
(iv) Online Analytical Processing Tools
(v) Data Mining Tools

Three-tier Data warehouse architecture



The bottom tier is a ware-house database server which is almost always a relational database system. The middle tier is an OLAP server which is typically implemented using either (1) a Relational OLAP (ROLAP) model, (2) a Multidimensional OLAP (MOLAP) model. The top tier is a client, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

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.

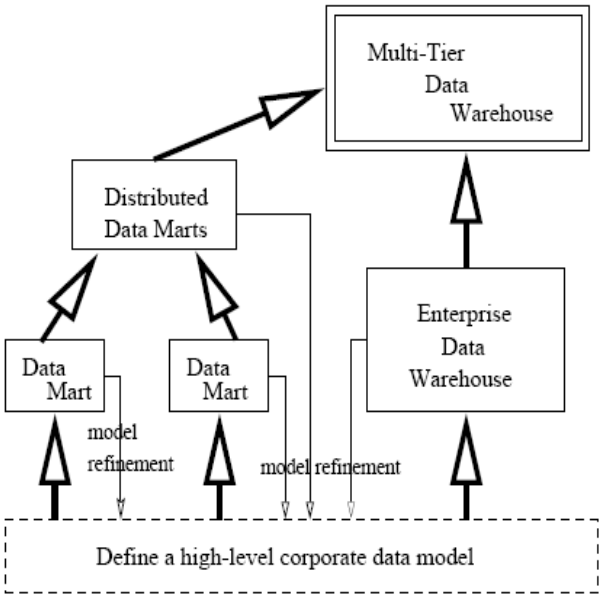
- **Data mart:** A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is connected to specific, selected subjects. For example, a marketing data mart may connect its subjects to customer, item, and sales. The data contained in data marts tend to be summarized. Depending on the source of data, data marts can be categorized into the following two classes:

  (i).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.

  (ii).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.

Figure: A recommended approach for data warehouse development.



Building a Data warehouse

The ETL (Extract Transformation Load) process

In this section we will discussed about the 4 major process of the data warehouse. They are extract (data from the operational systems and bring it to the data warehouse), transform (the data into internal format and structure of the data warehouse), cleanse (to make sure it is of sufficient quality to be used for decision making) and load (cleanse data is put into the data warehouse).

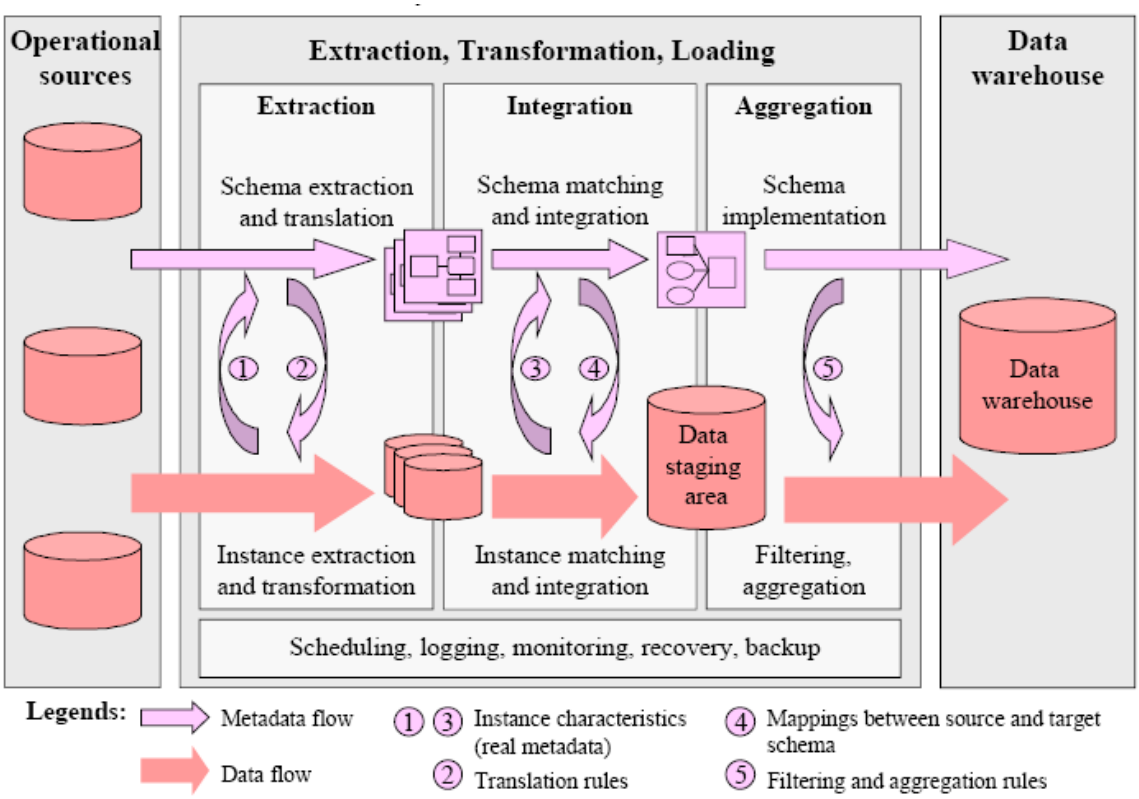


Figure 1. Steps of building a data warehouse: the ETL process

The four processes from extraction through loading often referred collectively as **Data Staging**.

EXTRACT

Some of the data elements in the operational database can be reasonably be expected to be useful in the decision making, but others are of less value for that purpose. For this reason, it is necessary to extract the relevant data from the operational database before bringing into the data warehouse. Many commercial tools are available to help with the extraction process. **Data Junction** is one of the commercial products. The user of one of these tools typically has an easy-to-use windowed interface by which to specify the following:

(i) Which files and tables are to be accessed in the source database?

(ii) Which fields are to be extracted from them? This is often done internally by SQL Select statement.

(iii) What are those to be called in the resulting database?
(iv) What is the target machine and database format of the output?
(v) On what schedule should the extraction process be repeated?

TRANSFORM

The operational databases developed can be based on any set of priorities, which keeps changing with the requirements. Therefore those who develop data warehouse based on these databases are typically faced with inconsistency among their data sources. Transformation process deals with rectifying any inconsistency (if any).

One of the most common transformation issues is 'Attribute Naming Inconsistency'. It is common for the given data element to be referred to by different data names in different databases. Employee Name may be EMP_NAME in one database, ENAME in the other. Thus one set of Data Names are picked and used consistently in the data warehouse. Once all the data elements have right names, they must be converted to common formats. The conversion may encompass the following:

(i) Characters must be converted ASCII to EBCDIC or vise versa.
(ii) Mixed Text may be converted to all uppercase for consistency.
(iii) Numerical data must be converted in to a common format.
(iv) Data Format has to be standardized.
(v) Measurement may have to convert. (Rs/ $)
(vi) Coded data (Male/ Female, M/F) must be converted into a common format.

All these transformation activities are automated and many commercial products are available to perform the tasks. **DataMAPPER** from Applied Database Technologies is one such comprehensive tool.

CLEANSING

Information quality is the key consideration in determining the value of the information. The developer of the data warehouse is not usually in a position to change the quality of its underlying historic data, though a data warehousing project can put spotlight on the data quality issues and lead to improvements for the future. It is, therefore, usually necessary to go through the data entered into the data warehouse and make it as error free as possible. This process is known as **Data Cleansing**.

Data Cleansing must deal with many types of possible errors. These include missing data and incorrect data at one source; inconsistent data and conflicting data when two or more source are involved. There are several algorithms followed to clean the data, which will be discussed in the coming lecture notes.

LOADING

Loading often implies physical movement of the data from the computer(s) storing the source database(s) to that which will store the data warehouse database, assuming it is different. This takes place immediately after the extraction phase. The most common channel for data movement is a high-speed communication link. Ex: Oracle Warehouse Builder is the API from Oracle, which provides the features to perform the

ETL task on Oracle Data Warehouse.

Data cleaning problems

This section classifies the major data quality problems to be solved by data cleaning and data transformation. As we will see, these problems are closely related and should thus be treated in a uniform way. Data transformations [26] are needed to support any changes in the structure, representation or content of data. These transformations become necessary in many situations, e.g., to deal with schema evolution, migrating a legacy system to a new information system, or when multiple data sources are to be integrated. As shown in Fig. 2 we roughly distinguish between single-source and multi-source problems and between schema- and instance-related problems. Schema-level problems of course are also reflected in the instances; they can be addressed at the schema level by an improved schema design (schema evolution), schema translation and schema integration. Instance-level problems, on the other hand, refer to errors and inconsistencies in the actual data contents which are not visible at the schema level. They are the primary focus of data cleaning. Fig. 2 also indicates some typical problems for the various cases. While not shown in Fig. 2, the single-source problems occur (with increased likelihood) in the multi-source case, too, besides specific multi-source problems.

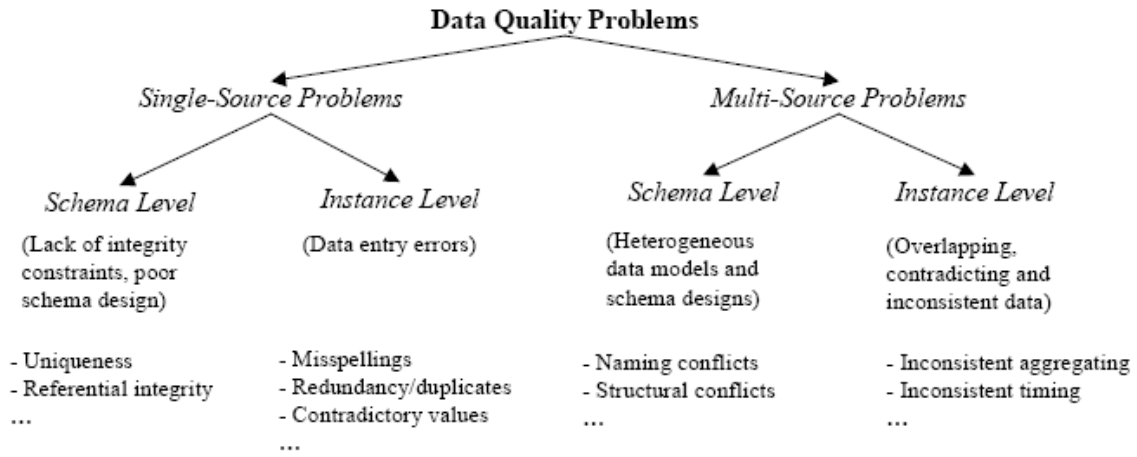


Figure 2. Classification of data quality problems in data sources

Single-source problems

The data quality of a source largely depends on the degree to which it is governed by schema and integrity constraints controlling permissable data values. For sources without schema, such as files, there are few restrictions on what data can be entered and stored, giving rise to a high probability of errors and inconsistencies. Database systems, on the other hand, enforce restrictions of a specific data model (e.g., the relational approach requires simple attribute values, referential integrity, etc.) as well as application-specific integrity constraints. Schema-related data quality problems thus occur because of the lack of appropriate model-specific or application-specific integrity constraints, e.g., due to data model limitations or poor schema design, or because only a few integrity constraints were defined to limit the overhead for integrity control. Instance-specific problems relate to errors and inconsistencies that cannot be prevented at the

schema level (e.g., misspellings).

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Illegal values	bdate=30.13.70	values outside of domain range
Record	Violated attribute dependencies	age=22, bdate=12.02.70	age = (current date – birth date) should hold
Record type	Uniqueness violation	emp_1=(name="John Smith", SSN="123456") emp_2=(name="Peter Miller", SSN="123456")	uniqueness for SSN (social security number) violated
Source	Referential integrity violation	emp=(name="John Smith", deptno=127)	referenced department (127) not defined

Table 1. Examples for single-source problems at schema level (violated integrity constraints)

For both schema- and instance-level problems we can differentiate different problem scopes: attribute (field), record, record type and source; examples for the various cases are shown in Tables 1 and 2. Note that uniqueness constraints specified at the schema level do not prevent duplicated instances, e.g., if information on the same real world entity is entered twice with different attribute values (see example in Table 2).

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city="Liipzig"	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience="B"; occupation="DB Prog."	
	Embedded values	name="J. Smith 12.02.70 New York"	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city="Germany"	
Record	Violated attribute dependencies	city="Redmond", zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= "J. Smith", $name_2$="Miller P."	usually in a free-form field
	Duplicated records	emp_1=(name="John Smith",...); emp_2=(name="J. Smith",...)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name="John Smith", bdate=12.02.70); emp_2=(name="John Smith", bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name="John Smith", deptno=17)	referenced department (17) is defined but wrong

Table 2. Examples for single-source problems at instance level

Multi-source problems

The problems present in single sources are aggravated when multiple sources need to be integrated. Each source may contain dirty data and the data in the sources may be represented differently, overlap or contradict. This is because the sources are typically developed, deployed and maintained independently to serve specific needs. This results in a large degree of heterogeneity w.r.t. data management systems, data models, schema designs and the actual data.

At the schema level, data model and schema design differences are to be addressed by the steps of schema translation and schema integration, respectively. The main problems w.r.t. schema design are naming and structural conflicts. Naming conflicts arise when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms). Structural conflicts occur in many variations and refer to different representations of the same object in different sources, e.g., attribute vs. table representation, different component structure, different data types, different integrity constraints, etc. In addition to schema-level conflicts, many conflicts appear only at the instance level (data conflicts). All problems from the single-source case can occur with different representations in different sources (e.g., duplicated records, contradicting records,…). Furthermore, even when there are the same attribute names and data types, there may be different value representations (e.g., for marital status) or different interpretation of the values (e.g., measurement units Dollar vs. Euro) across sources. Moreover, information in the sources may be provided at different aggregation levels (e.g., sales per product vs. sales per product group) or refer to different points in time (e.g. current sales as of yesterday for source 1 vs. as of last week for source 2).

A main problem for cleaning data from multiple sources is to identify overlapping data, in particular matching records referring to the same real-world entity (e.g., customer). This problem is also referred to as the object identity problem, duplicate elimination or the merge/purge problem. Frequently, the

information is only partially redundant and the sources may complement each other by providing additional information about an entity. Thus duplicate information should be purged out and complementing information should be consolidated and merged in order to achieve a consistent view of real world entities.

Customer (source 1)

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Client (source 2)

Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Customers (integrated target with cleaned data)

No	*LName*	*FName*	*Gender*	*Street*	*City*	*State*	*ZIP*	*Phone*	*Fax*	*CID*	*Cno*
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

Figure 3. Examples of multi-source problems at schema and instance level

The two sources in the example of Fig. 3 are both in relational format but exhibit schema and data conflicts. At the schema level, there are name conflicts (synonyms *Customer*/*Client*, *Cid*/*Cno*, *Sex*/*Gender*) and structural conflicts (different representations for names and addresses). At the instance level, we note that there are different gender representations ("0"/"1" vs. "F"/"M") and presumably a duplicate record (Kristen Smith). The latter observation also reveals that while *Cid*/*Cno* are both source-specific identifiers, their contents are not comparable between the sources; different numbers (11/493) may refer to the same person while different persons can have the same number (24). Solving these problems requires both schema integration and data cleaning; the third table shows a possible solution. Note that the schema conflicts should be resolved first to allow data cleaning, in particular detection of duplicates based on a uniform representation of names and addresses, and matching of the *Gender*/*Sex* values.

Data cleaning approaches

In general, data cleaning involves several phases

 Data analysis: In order to detect which kinds of errors and inconsistencies are to be removed, a detailed data analysis is required. In addition to a manual inspection of the data or data samples, analysisprograms should be used to gain metadata about the data properties and detect data quality problems.

 Definition of transformation workflow and mapping rules: Depending on the number of data sources, their degree of heterogeneity and the "dirtyness" of the data, a large number of data transformation and cleaning steps may have to be executed. Sometime, a schema translation is used to map sources to a common data model; for data warehouses, typically a relational representation is used. Early datacleaning steps can correct single-source instance problems and prepare the data for integration. Latersteps deal with schema/data integration and cleaning multi-source instance problems, e.g., duplicates.

For data warehousing, the control and data flow for these transformation and cleaning steps should be specified within a workflow that defines the ETL process (Fig. 1).

The schema-related data transformations as well as the cleaning steps should be specified by a declarative query and mapping language as far as possible, to enable automatic generation of the transformation code. In addition, it should be possible to invoke user-written cleaning code and specialpurpose tools during a data transformation workflow. The transformation steps may request user feedback on data instances for which they have no built-in cleaning logic.

 Verification: The correctness and effectiveness of a transformation workflow and the transformation definitions should be tested and evaluated, e.g., on a sample or copy of the source data, to improve the

definitions if necessary. Multiple iterations of the analysis, design and verification steps may be needed, e.g., since some errors only become apparent after applying some transformations.

 Transformation: Execution of the transformation steps either by running the ETL workflow for loading and refreshing a data warehouse or during answering queries on multiple sources.

 Backflow of cleaned data: After (single-source) errors are removed, the cleaned data should also replace the dirty data in the original sources in order to give legacy applications the improved data too and to avoid redoing the cleaning work for future data extractions. For data warehousing, the cleaned data is available from the data staging area (Fig. 1).

Data analysis

Metadata reflected in schemas is typically insufficient to assess the data quality of a source, especially if only a few integrity constraints are enforced. It is thus important to analyse the actual instances to obtain real (reengineered) metadata on data characteristics or unusual value patterns. This metadata helps finding data quality problems. Moreover, it can effectively contribute to identify attribute correspondences between source schemas (schema matching), based on which automatic data transformations can be derived.

There are two related approaches for data analysis, data profiling and data mining. *Data profiling* focuses on the instance analysis of individual attributes. It derives information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, typical string pattern (e.g., for phone numbers), etc., providing an exact view of various quality aspects of the attribute. Table 3 shows examples of how this metadata can help detecting data quality problems.

Problems	Metadata	Examples/Heuristics
Illegal values	cardinality	e.g., cardinality (gender) > 2 indicates problem
	max, min	max, min should not be outside of permissible range
	variance, deviation	variance, deviation of statistical values should not be higher than threshold
Misspellings	attribute values	sorting on values often brings misspelled values next to correct values
Missing values	null values	percentage/number of null values
	attribute values + default values	presence of default value may indicate real value is missing
Varying value representations	attribute values	comparing attribute value set of a column of one table against that of a column of another table
Duplicates	cardinality + uniqueness	attribute cardinality = # rows should hold
	attribute values	sorting values by number of occurrences; more than 1 occurrence indicates duplicates

Table 3. Examples for the use of reengineered metadata to address data quality problems

Data mining helps discover specific data patterns in large data sets, e.g., relationships holding between several attributes. This is the focus of so-called descriptive data mining models including clustering, summarization, association discovery and sequence discovery [10]. As shown in [28], integrity constraints among attributes such as functional dependencies or application-specific "business rules" can be derived, which can be used to complete missing values, correct illegal values and identify duplicate records across data sources. For example, an association rule with high confidence can hint to data quality problems in instances violating this rule. So a confidence of 99% for rule "*total=quantity*unit price*" indicates that 1% of the records do not comply and may require closer examination.

Defining data transformations

The data transformation process typically consists of multiple steps where each step may perform schema and instance-related transformations (mappings). To allow a data transformation and cleaning system to generate transformation code and thus to reduce the amount of self-programming it is necessary to specify the required transformations in an appropriate language, e.g., supported by a graphical user interface.

Various ETL tools (see Section 4) offer this functionality by supporting proprietary rule languages. A more general and flexible approach is the use of the standard query language SQL to perform the data transformations and utilize the possibility of application-specific language extensions, in particular userdefined

functions (UDFs) .UDFs can be implemented in SQL or a general purpose programming language with embedded SQL statements. They allow implementing a wide range of data transformations and support easy reuse for different transformation and query processing tasks. Furthermore, their execution by the DBMS can reduce data access cost and thus improve performance.

```
CREATE VIEW Customer2 (LName, FName, Gender, Street, City, State, ZIP, CID) AS
SELECT  LastNameExtract (Name), FirstNameExtract (Name), Sex, Street, CityExtract (City),
        StateExtract (City), ZIPExtract (City), CID
FROM Customer
```

Figure 4. Example of transformation step definition

necessary data transformations to be applied to the first source. The transformation defines a view on which further mappings can be performed. The transformation performs a schema restructuring with additional attributes in the view obtained by splitting the name and address attributes of the source. The required data extractions are achieved by UDFs (shown in boldface). The UDF implementations can contain cleaning logic, e.g., to remove misspellings in city names or provide missing zip codes.

UDFs may still imply a substantial implementation effort and do not support all necessary schema transformations. In particular, simple and frequently needed functions such as attribute splitting or merging are not generically supported but need often to be re-implemented in application-specific variations (see specific extract functions in Fig. 4).

Conflict resolution

A set of transformation steps has to be specified and executed to resolve the various schema- and instancelevel data quality problems that are reflected in the data sources at hand. Several types of transformations are to be performed on the individual data sources in order to deal with single-source problems and to prepare for integration with other sources. In addition to a possible schema translation, these preparatory steps typically include:

 Extracting values from free-form attributes (attribute split): Free-form attributes often capture multiple individual values that should be extracted to achieve a more precise representation and support further cleaning steps such as instance matching and duplicate elimination. Typical examples are name and address fields (Table 2, Fig. 3, Fig. 4). Required transformations in this step are reordering of values within a field to deal with word transpositions, and value extraction for attribute splitting.

 Validation and correction: This step examines each source instance for data entry errors and tries to correct them automatically as far as possible. Spell checking based on dictionary lookup is useful for identifying and correcting misspellings. Furthermore, dictionaries on geographic names and zip codes help to correct address data. Attribute dependencies (birthdate – age, total price – unit price / quantity, city – phone area code,…) can be utilized to detect problems and substitute missing values or correct wrong values.

 Standardization*:* To facilitate instance matching and integration, attribute values should be converted to a consistent and uniform format. For example, date and time entries should be brought into a specific format; names and other string data should be converted to either upper or lower case, etc. Text data may be condensed and unified by performing stemming, removing prefixes, suffixes, and stop words.

Furthermore, abbreviations and encoding schemes should consistently be resolved by consulting special synonym dictionaries or applying predefined conversion rules. Dealing with multi-source problems requires restructuring of schemas to achieve a schema integration, including steps such as splitting, merging, folding and unfolding of attributes and tables. At the instance level, conflicting representations need to be resolved and overlapping data must to be dealt with. The *duplicate elimination* task is typically performed after most other transformation and cleaning steps, especially after having cleaned single-source errors and conflicting representations. It is performed either on two cleaned sources at a time or on a single already integrated data set. Duplicate elimination requires to first identify (i.e. match) similar records concerning the same real world entity. In a

second step, similar records are merged into one record containing all relevant attributes without redundancy. Furthermore, redundant records are purged.

Tool support

ETL tools

A large number of commercial tools support the ETL process for data warehouses in a comprehensive way, e.g., COPYMANAGER (InformationBuilders), DATASTAGE (Informix/Ardent), EXTRACT (ETI), POWERMART (Informatica), DECISIONBASE (CA/Platinum), DATATRANSFORMATIONSERVICE (Microsoft), METASUITE (Minerva/Carleton), SAGENTSOLUTIONPLATFORM (Sagent), and WAREHOUSEADMINISTRATOR (SAS). They use a repository built on a DBMS to manage all metadata about the data sources, target schemas, mappings, script programs, etc., in a uniform way. Schemas and data are extracted from operational data sources via both native file and DBMS gateways as well as standard interfaces such as ODBC and EDA. Data transformations are defined with an easy-to-use graphical interface. To specify individual mapping steps, a proprietary rule language and a comprehensive library of predefined conversion functions are typically provided. The tools also support reusing existing transformation solutions, such as external C/C++ routines, by providing an interface to integrate them into the internal transformation library. Transformation processing is carried out either by an engine that interprets the specified transformations at runtime, or by compiled code. All engine-based tools (e.g., COPYMANAGER, DECISIONBASE, POWERMART, DATASTAGE, WAREHOUSEADMINISTRATOR), possess a scheduler and support workflows with complex execution dependencies among mapping jobs. A workflow may also invoke external tools, e.g., for specialized cleaning tasks such as name/address cleaning or duplicate elimination. ETL tools typically have little built-in data cleaning capabilities but allow the user to specify cleaning functionality via a proprietary API. There is usually no data analysis support to automatically detect data errors and inconsistencies. However, users can implement such logic with the metadata maintained and by determining content characteristics with the help of aggregation functions (sum, count, min, max, median, variance, deviation,…). The provided transformation library covers many data transformation and cleaning needs, such as data type conversions (e.g., date reformatting), string functions (e.g., split, merge, replace, sub-string search), arithmetic, scientific and statistical functions, etc. Extraction of values from free-form attributes is not completely automatic but the user has to specify the delimiters separating sub-values. The rule languages typically cover *if-then* and *case* constructs that help handling exceptions in data values, such as misspellings, abbreviations, missing or cryptic values, and values outside of range. These problems can also be addressed by using a table lookup construct and join functionality. Support for instance matching is typically restricted to the use of the join construct and some simple string matching functions, e.g., exact or wildcard matching and soundex. However, user-defined field matching functions as well as functions for correlating field similarities can be programmed and added to the internal transformation library.

Metadata repository

Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouseobjects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for time stamping 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:

- A description of the structure of the data warehouse. This 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;

- The 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; and

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

Features of OLTP and OLAP

The major distinguishing features between OLTP and OLAP are summarized as follows.

1. 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.

2. 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 historical data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity. These features make the data easier for use in informed decision making.

3. 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 snowflake model and a subject-oriented database design.

4. View: An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historical data or data in different organizations. In contrast, an OLAP system often spans multiple versions of a database schema. 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.

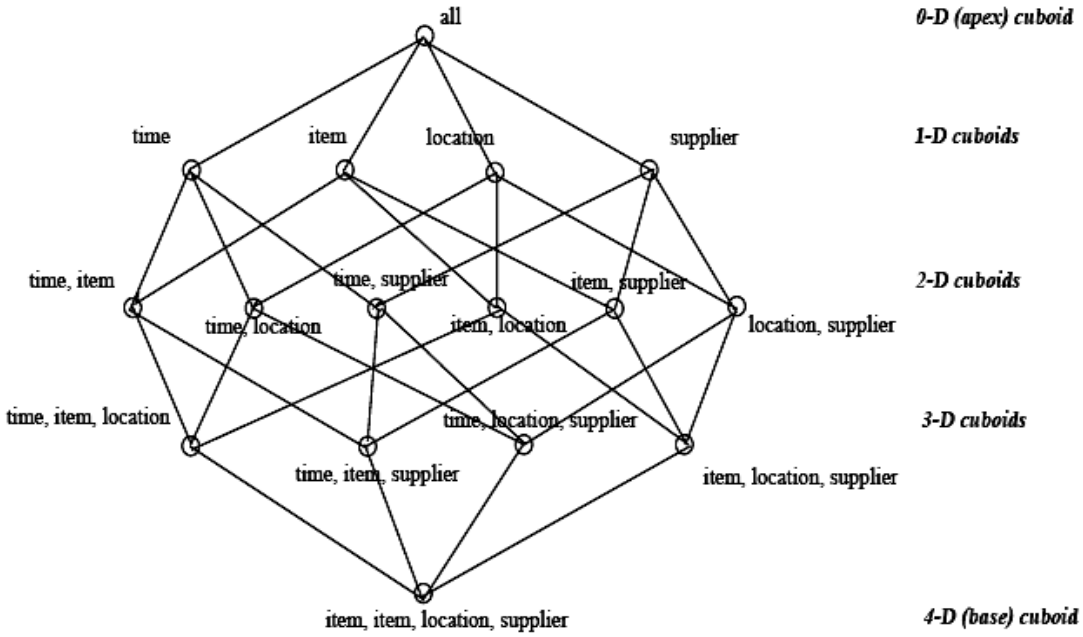
5. Access patterns: The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations although many could be complex queries.

Comparison between OLTP and OLAP systems.

Feature	OLTP	OLAP
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long term informational requirements, decision support
DB design	E-R based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
# of records accessed	tens	millions
# of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

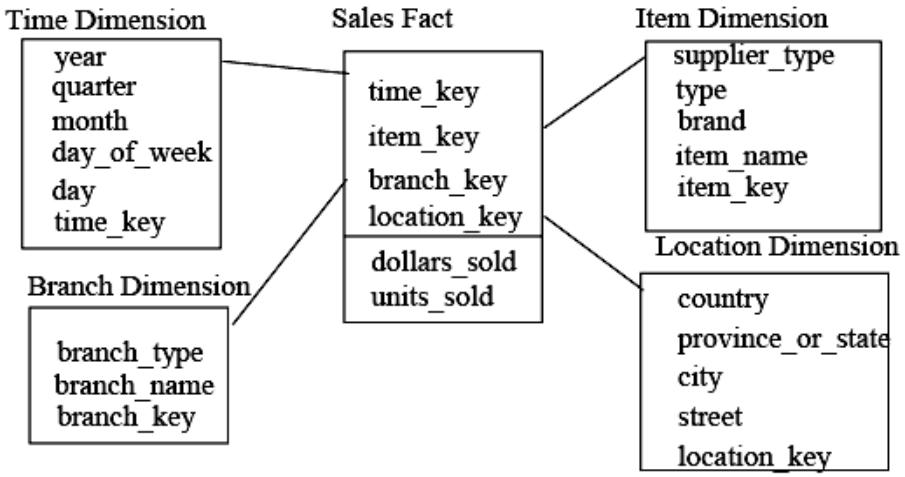
Multidimensional DataModel.

The most popular data model for data warehouses is a multidimensional model. This model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema. Let's have a look at each of these schema types.



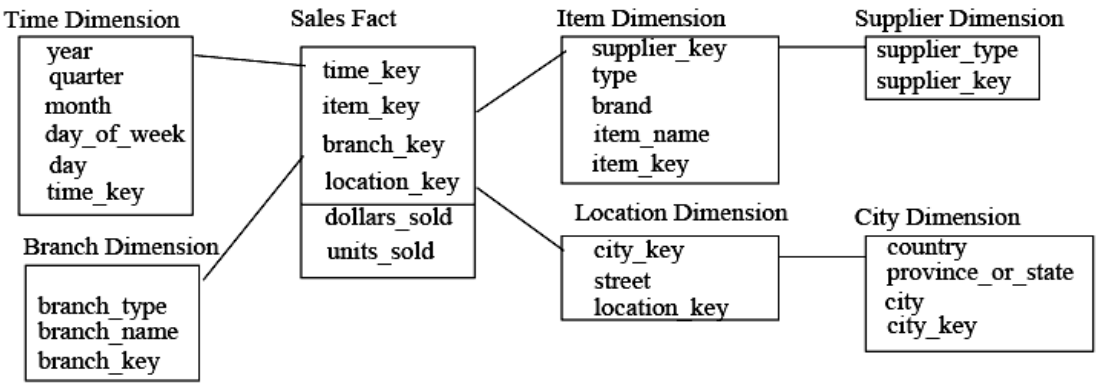
- **Star schema**: The star schema is a modeling paradigm in which the data warehouse contains (1) a large central table (fact table), and (2) a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.

Figure Star schema of a data warehouse for sales.



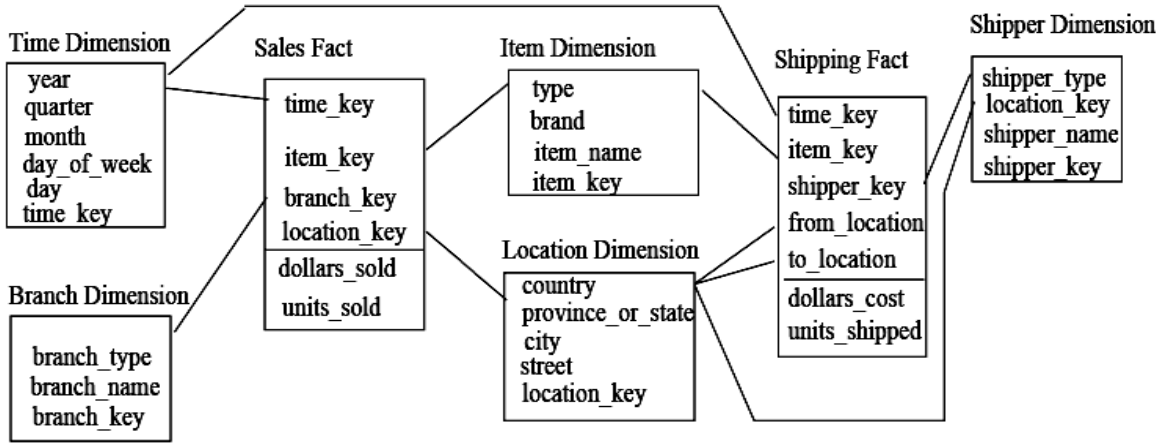
- **Snowflake schema**: The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake. The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form. Such a table is easy to maintain and also saves storage space because a large dimension table can be extremely large when the dimensional structure is included as columns.

Figure Snowflake schema of a data warehouse for sales.



- **Fact constellation**: Sophisticated applications may require multiple fact tables to share dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

Figure Fact constellation schema of a data warehouse for sales and shipping.



A Data Mining Query Language, DMQL: Language Primitives

- Cube Definition (Fact Table)

define cube <cube_name> [<dimension_list>]: <measure_list>

- Dimension Definition (Dimension Table)

define dimension <dimension_name> as (<attribute_or_subdimension_list>)

- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining a Star Schema in DMQL

define cube sales_star [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

Defining a Snowflake Schema in DMQL

define cube sales_snowflake [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city(city_key, province_or_state, country))

Defining a Fact Constellation in DMQL

define cube sales [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

define cube shipping [time, item, shipper, from_location, to_location]:

dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)

define dimension time as time in cube sales

define dimension item as item in cube sales

define dimension shipper as (shipper_key, shipper_name, location as location in cube sales, shipper_type)

define dimension from_location as location in cube sales

define dimension to_location as location in cube sales

Measures: Three Categories

Measure: a function evaluated on aggregated data corresponding to given dimension-value pairs.

Measures can be:

- distributive: if the measure can be calculated in a distributive manner.
    - E.g., count(), sum(), min(), max().
- algebraic: if it can be computed from arguments obtained by applying distributive aggregate functions.
    - E.g., avg()=sum()/count(), min_N(), standard_deviation().
- holistic: if it is not algebraic.
    - E.g., median(), mode(), rank().

A Concept Hierarchy

Concept hierarchies allow data to be handled at varying levels of abstraction

OLAP operations on multidimensional data.

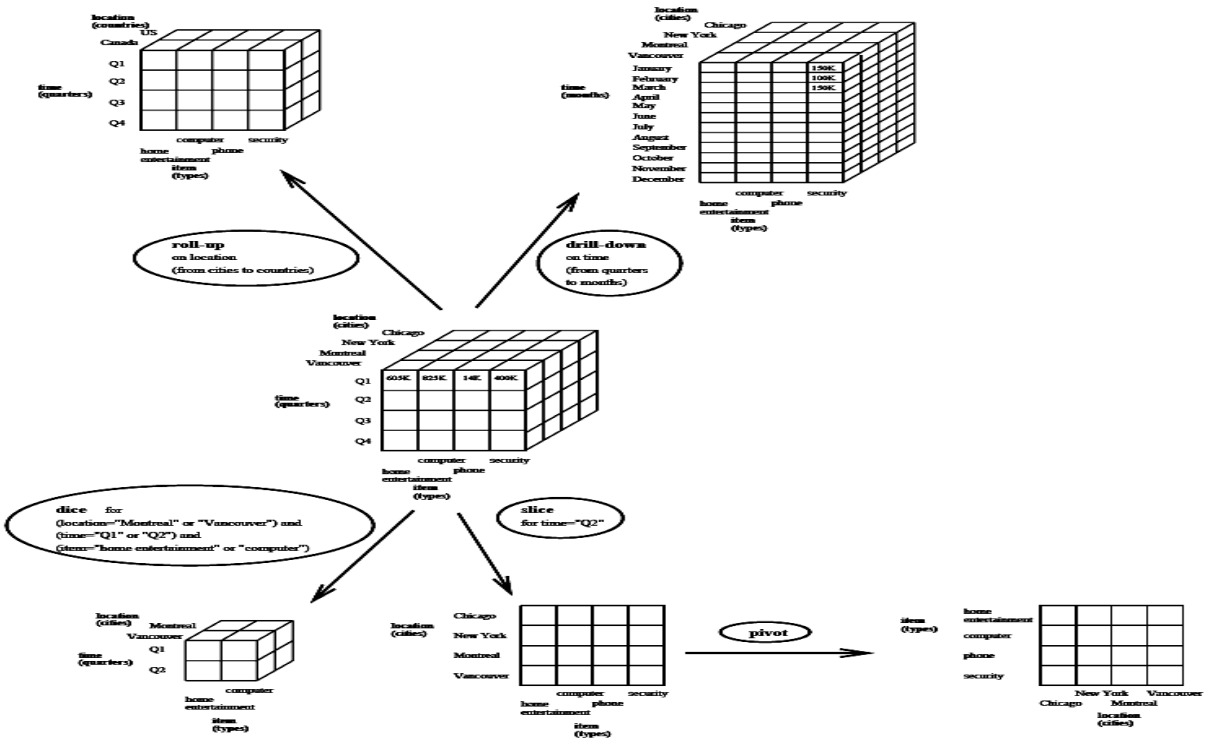
1. **Roll-up**: The roll-up operation performs aggregation on a data cube, either by climbing-up a concept hierarchy for a dimension or by dimension reduction. Figure shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for location. This hierarchy was defined as the total order street < city < province or state <country.

2. **Drill-down**: Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping-down a concept hierarchy for a dimension or introducing additional dimensions. Figure shows the result of a drill-down operation performed on the central cube by stepping down a concept hierarchy for time defined as day < month < quarter < year. Drill-down occurs by descending the time hierarchy from the level of quarter to the more detailed level of month.

3. **Slice and dice:** The slice operation performs a selection on one dimension of the given cube, resulting in a subcube. Figure shows a slice operation where the sales data are selected from the central cube for the dimension time using the criteria time=”Q2". The dice operation defines a subcube by performing a selection on two or more dimensions.

4. **Pivot (rotate):** Pivot is a visualization operation which rotates the data axes in view in order to provide an alternative presentation of the data. Figure shows a pivot operation where the item and location axes in a 2-D slice are rotated.

Figure : Examples of typical OLAP operations on multidimensional data.



From on-line analytical processing to on-line analytical mining.

On-Line Analytical Mining (OLAM) (also called OLAP mining), which integrates on-line analytical processing (OLAP) with data mining and mining knowledge in multidimensional databases, is particularly important for the following reasons.

1. 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 transformation and data integration 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.

2. 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, reporting and OLAP analysis tools.

3. OLAP-based exploratory data analysis.

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. On-line analytical mining provides facilities for data mining on different subsets of data and at different levels of abstraction, by drilling, pivoting, filtering, dicing and slicing on a data cube and on some intermediate data mining results.

4. On-line selection of data mining functions.

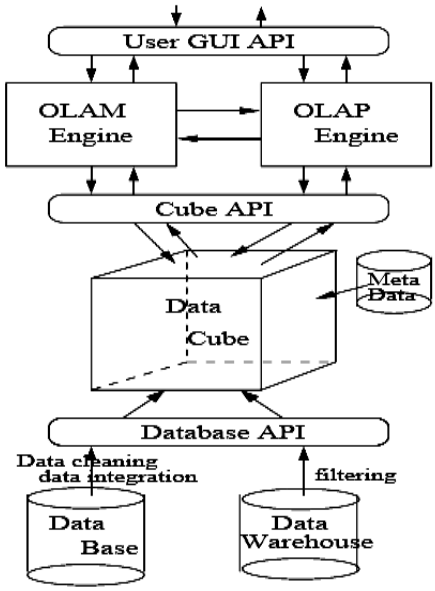
By integrating OLAP with multiple data mining functions, on-line analytical mining provides users with the exibility to select desired data mining functions and swap data mining tasks dynamically.

Architecture for on-line analytical mining

An OLAM engine performs analytical mining in data cubes in a similar manner as an OLAP engine performs on-line analytical processing. An integrated OLAM and OLAP architecture is shown in Figure, where the OLAM and OLAP engines both accept users' on-line queries via a User GUI API and work with the data cube in the data analysis via a Cube API.

A metadata directory is used to guide the access of the data cube. The data cube can be constructed by accessing and/or integrating multiple databases and/or by filtering a data warehouse via a Database API which may support OLEDB or ODBC connections. Since an OLAM engine may perform multiple data mining tasks, such as concept description, association, classification, prediction, clustering, time-series analysis,etc., it usually consists of multiple, integrated data mining modules and is more sophisticated than an OLAP engine.

Figure: An integrated OLAM and OLAP architecture.



Data Cube Computation.

Data cube can be viewed as a lattice of cuboids

- The bottom-most cuboid is the base cuboid
- The top-most cuboid (apex) contains only one cell
- How many cuboids in an n-dimensional cube with L levels?

Materialization of data cube

- Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
- Selection of which cuboids to materialize
- Based on size, sharing, access frequency, etc.

Cube Operation

- Cube definition and computation in DMQL

  define cube sales[item, city, year]: sum(sales_in_dollars)

  compute cube sales

- Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

  SELECT item, city, year, SUM (amount)

FROM SALES

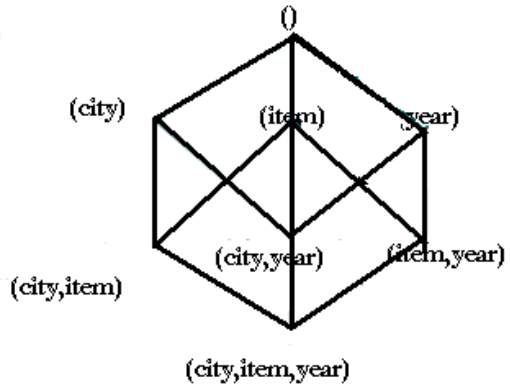
CUBE BY item, city, year

- Need compute the following Group-Bys
  (date, product, customer),

  (date,product),(date, customer), (product, customer),

  (date), (product), (customer)

  ()

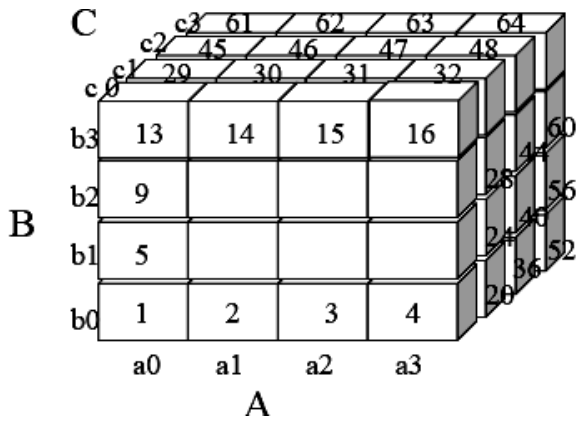


Cube Computation: ROLAP-Based Method

- Efficient cube computation methods
  - ROLAP-based cubing algorithms (Agarwal et al'96)
  - Array-based cubing algorithm (Zhao et al'97)
  - Bottom-up computation method (Bayer & Ramarkrishnan'99)
- ROLAP-based cubing algorithms
  - Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
  - Grouping is performed on some subaggregates as a "partial grouping step"
  - Aggregates may be computed from previously computed aggregates, rather than from the base fact table

Multi-way Array Aggregation for Cube

- Computation
- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



Indexing OLAP data

The bitmap indexing method is popular in OLAP products because it allows quick searching in data cubes.

The **bitmap index** is an alternative representation of the record ID (RID) list. In the bitmap index for a given attribute, there is a distinct bit vector, By, for each value v in the domain of the attribute. If the domain of a given attribute consists of n values, then n bits are needed for each entry in the bitmap index

The **join indexing** method gained popularity from its use in relational database query processing. Traditional indexing maps the value in a given column to a list of rows having that value. In contrast, join indexing registers the joinable rows of two relations from a relational database. For example, if two relations R(RID;A) and S(B; SID) join on the attributes A and B, then the join index record contains the pair (RID; SID), where RID and SID are record identifiers from the R and S relations, respectively.

Efficient processing of OLAP queries

1. Determine which operations should be performed on the available cuboids. This involves transforming any selection, projection, roll-up (group-by) and drill-down operations specified in the query into corresponding SQL and/or OLAP operations. For example, slicing and dicing of a data cube may correspond to selection and/or projection operations on a materialized cuboid.

2. Determine to which materialized cuboid(s) the relevant operations should be applied. This involves identifying all of the materialized cuboids that may potentially be used to answer the query, pruning the

Metadata repository

Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouseobjects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for time stamping 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:

- A description of the structure of the data warehouse. This 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;

- The 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; and

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

Data warehouse back-end tools and utilities

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

1. Data extraction, which typically gathers data from multiple, heterogeneous, and external sources;
2. Data cleaning, which detects errors in the data and recti_es them when possible;
3. Data transformation, which converts data from legacy or host format to warehouse format;
4. Load, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions;
5. Refresh, which propagates the updates from the data sources to the warehouse.
6. Besides cleaning, loading, refreshing, and metadata defnition tools, data warehouse systems usually provide a good set of data warehouse management tools.

OLAP Server Architectures.

Relational OLAP (ROLAP) servers: These are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces. ROLAP servers include optimization for each DBMS back-end, implementation of aggregation navigation logic, and additional tools and services. ROLAP technology tends to have greater scalability than MOLAP technology.

Multidimensional OLAP (MOLAP) servers: These servers support multidimensional views of data through array-based multidimensional storage engines. They map multidimensional views directly to data cube array structures.Many OLAP servers adopt a two-level storage representation to handle sparse and dense data sets: the dense subcubes are identified and stored as array structures, while the sparse subcubes employ compression technology for efficient storage utilization.

Hybrid OLAP (HOLAP) servers: The hybrid OLAP approach combines ROLAP and MOLAP technology, benefitting from the greater scalability of ROLAP and the faster computation of MOLAP. For example, a HOLAP server may allow large volumes of detail data to be stored in a relational database, while aggregations are kept in a separate MOLAP store.

Specialized SQL servers: To meet the growing demand of OLAP processing in relational databases, some relational and data warehousing forms (e.g., Redbrick) implement specialized SQL servers which provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

Comparison between MDDBs and RDBMSs

MDDB	RDBMS
Data is stored in multidimensional arrays	Data is stored in relations
Direct inspection of an array gives a great deal of information	Not so
Can handle limited size databases (< 100GB)	Proven track record for handling VLDBs
Takes long to load and update	Highly volatile data are better handled
Support aggregations better	RDBMSs are catching up-Aggregate Navigators
New investments need to be made and new skill sets need to be developed	Most enterprises already made significant investments in RDBMS technology and skill sets
Adds complexity to the overall system architecture	No additional complexity
Limited no. of facts an dimensional tables	No such restriction
Examples	Examples

• Arbor-Essbase • Brio Query-Enterprise • Dimensional Insight-DI Diver • Oracle-Express Server	• IBM-DB2 • Microsoft-SQL Server • Oracle-Oracle RDBMS • Red Brick Systems-Red Brick Warehouse

References

- M. S. Chen, J. Han, and P. S. Yu. Data mining: An overview from a database perspective. IEEE Trans. Knowledge and Data Engineering, 8:866-883, 1996.

- U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996.

- W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, Knowledge Discovery in Databases: An Overview. In G. Piatetsky-Shapiro et al. (eds.), Knowledge Discovery in Databases. AAAI/MIT Press, 1991.

- J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

- T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM, 39:58-64, 1996.

- G. Piatetsky-Shapiro, U. M. Fayyad, and P. Smyth. From data mining to knowledge discovery: An overview. In U.M. Fayyad, et al. (eds.), Advances in Knowledge Discovery and Data Mining, 1-35. AAAI/MIT Press, 1996.

- G. Piatetsky-Shapiro and W. J. Frawley. Knowledge Discovery in Databases. AAAI/MIT Press, 1991.