Chapter 1.docx

Chapter 2.doc

Chapter 2.docx

Chapter 3.doc

Chapter 4.doc

Chapter 5.doc

Chapter 6.docx

Chapter 7.docx

Chapter 8.doc

Chapter 8.docx

Chapter 9.pdf

Unit 1

Data Mining Concepts

Simply stated, data mining refers to *extracting or "mining" knowledge from large amounts of data stored in databases, data warehouses, or other information repositories*. Many people treat data mining as a synonym for another popularly used term, Knowledge Discovery from Data, or KDD. Alternatively, others view data mining as simply an essential step in the process of knowledge discovery. Knowledge discovery consists of an iterative sequence of the following steps:

- **Data cleaning** - It removes noise and inconsistent data
- **Data integration** - This combines data from multiple data sources
- **Data selection** - Data relevant to the analysis task are retrieved from the database
- **Data transformation** - Data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
- **Data mining** - an essential process where intelligent methods are applied in order to extract data patterns
- **Pattern evaluation** - Identifies the truly interesting patterns representing knowledge based on some interestingness measures.
- **Knowledge presentation** - Knowledge representation techniques are used to present the mined knowledge to the user.

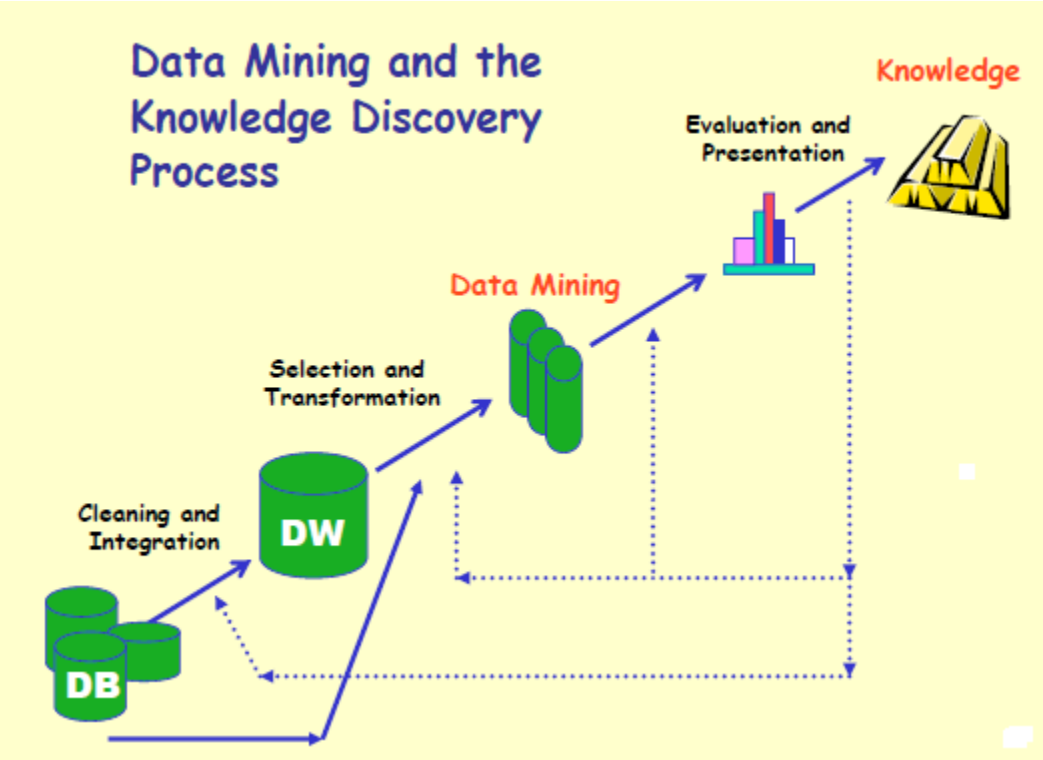


Figure: Knowledge Discovery Process (Stages of KDD)

According to this view, data mining is only one step in the knowledge discovery process. However, in industry, in media, and in the database research milieu, the term data mining is becoming more popular than the longer term of knowledge discovery from data. Therefore, in this book, we choose to use the term data mining.

Based on this view, the architecture of a typical data mining system may have the following major components.

- → **Database, Data Warehouse, World Wide Web, or Other Information Repository**: This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories. Data cleaning and data integration techniques may be performed on the data.
- → **Database or Data Warehouse Server**: The database or data warehouse server is responsible for fetching the relevant data, based on the user's data mining request.
- → **Knowledge Base**: This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. It is simply stored in the form of set of rules. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction.
- → **Data Mining Engine**: This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.
- → **Pattern Evaluation Module**: This component typically employs interestingness measures and interacts with the data mining modules so as to *focus* the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns.
- → **User interface**: This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

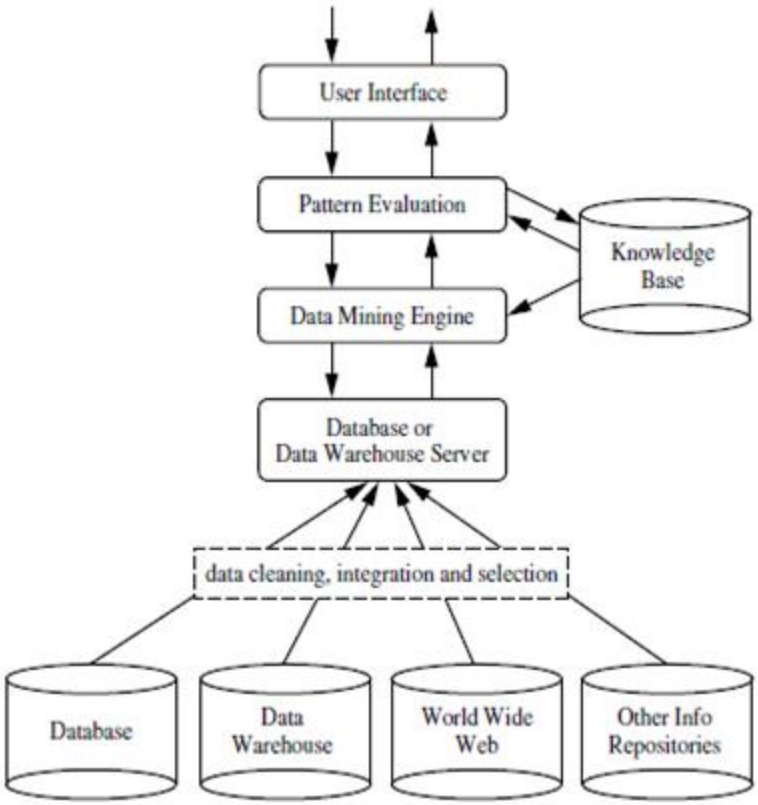


Figure: Architecture of Data Mining System

Data Warehouse Concepts

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site. Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing. To facilitate decision making, the data in a data warehouse are *organized around major subjects*, such as customer, item, supplier, and activity. A data warehouse is usually modeled by a multidimensional database structure, where each dimension corresponds to an attribute or a set of attributes in the schema, and each cell stores the value of some aggregate measure, such as *count* or *sales amount*.

Multidimensional structure is defined as a variation of the relational model that uses multidimensional structures to organize data and express the relationships between data. The structure is broken into cubes and the cubes are able to store and access data within the confines of each cube. "Each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions.

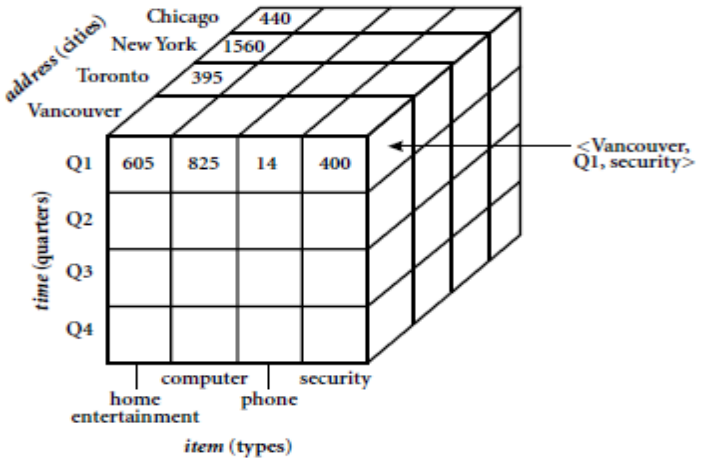
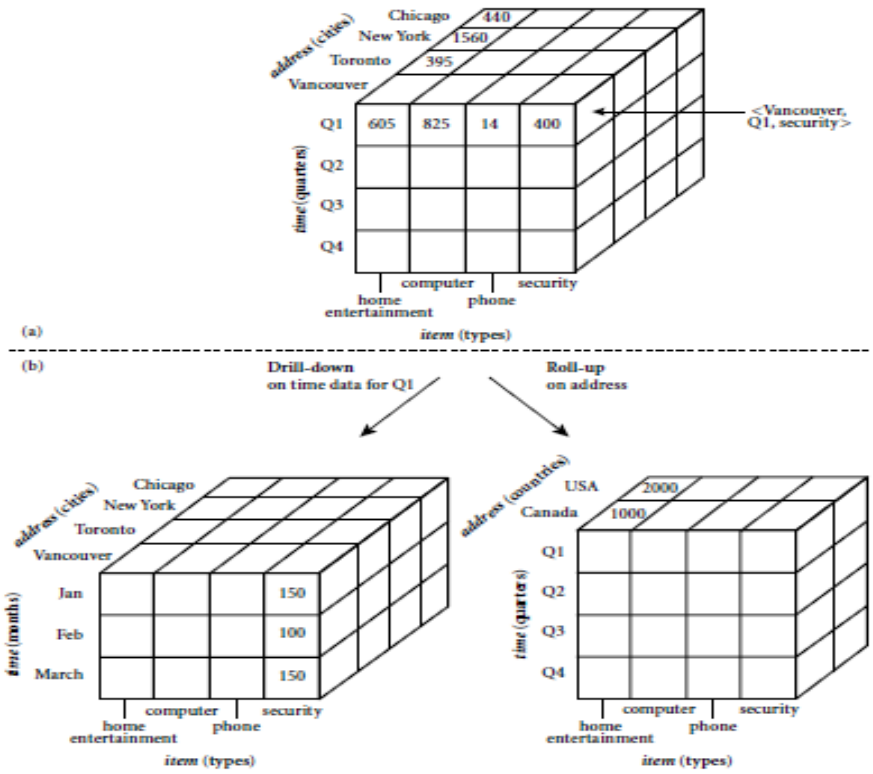


Figure: Multidimensional Database

By providing multidimensional data views and the pre-computation of summarized data, data warehouse systems are well suited for on-line analytical processing, or OLAP. Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization. We can drill down on sales data summarized by *quarter* to see the data summarized by *month*. Similarly, we can roll up on sales data summarized by *city* to view the data summarized by *country*.



Functions of Data Warehousing

As already mentioned, data Warehousing is a method for gathering and controlling data from different sources making the data easily available for querying and analysis. The following are the functions of data warehouse are discussed below:

- → **Data Extraction** - Involves gathering data from multiple heterogeneous sources.
- → **Data Cleaning** - Involves finding and correcting the errors in data.
- → **Data Transformation** - Involves converting the data from legacy format to warehouse format.
- → **Data Loading** - Involves sorting, summarizing, consolidating, checking integrity, and building indices and partitions.
- → **Refreshing** - Involves updating from data sources to warehouse.

Setting up KDD Environment

Main goal of KDD is to obtain better understanding of changing organizational environment. It includes a data mining process but using data mining as single generic tool is neither realistic nor desirable. KDD environment needs a suit of data mining tools that needs to be selected and tuned carefully for each organization according to their need. Some of the golden rules that should be followed for setting up KDD environment are discussed below:

- → **Support for Extremely Large Data Set**: Data mining deals with billions of records. Thus fast and flexible way of storing and handling such large volume of data is requires along with capacity of storing intermediate results.
- → **Support Hybrid Learning:** Leaning can be divided into three categories: Classification, Knowledge engineering, & problem solving. Complex data mining projects needs hybrid learning algorithms that possess capabilities of all three categories.
- → **Establish a Data Warehouse**: Data mining mainly depends upon availability and analysis of historic data and hence establishing data warehouse is important for this.
- → **Introduce Data Cleaning Facility**: If data stored in databases contains noise, data processing suffers from pollution. Thus data cleaning facility is necessary to avoid such pollution.
- → **Facilitate Working with Dynamic Coding**: A KDD Environment should enable the user to experiment with different coding schemes. It must keep the track of genealogy of different samples and tables as well as the semantics and transformations of the different attributes are vital.

→ **Beware of False Predictors**: If the results are too good to be true, you probably have found false predictors.

→ **Verify Result:** Examine the results carefully and repeat and refine the knowledge discovery process until you are confident.

Data Mining Functionalities

In general, data mining tasks can be classified into two categories: descriptive and predictive. Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions. Data mining functionalities, and the kinds of patterns they can discover, are described below.

Concept/Class Description: Data Mining can be used to describe individual classes and concepts in data stores. These descriptions can be derived via data characterization or data discrimination. *Data characterization* is a summarization of the general characteristics or features of a target class of data. For example, A data mining system should be able to produce a description summarizing the characteristics of customers who spend more than $1,000 in a year. Data discrimination is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes. The target and contrasting classes can be specified by the user. For example, A data mining system should be able to compare two groups of customers, such as those who shop for computer products regularly versus those who rarely shop for such products.

Mining Frequent Patterns, Associations, and Correlations: Frequent patterns are patterns that occur frequently in data. Frequent patterns may include frequent itemsets, subsequences, and substructures. A *frequent itemsets* typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread. A pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is *pattern* is a frequent subsequence. A different structural form, such as graphs, trees, or lattices, is called a (*frequent*) *structured pattern*. Mining frequent patterns leads to the discovery of interesting associations and correlations within data. For example, a marketing manager of an *Electronics store* would like to determine which items are frequently purchased together within the same transactions. For this mining rule can be

buys(*X*; "*computer*"))=>*buys*(*X*; "*software*") [*support* = 1%; *confidence* = 50%]

Where X is a variable representing a customer

Typically, association rules are discarded as uninteresting if they do not satisfy both a minimum support threshold and a minimum confidence threshold. Additional analysis can be performed to uncover interesting statistical correlations between associated attribute-value pairs.

Classification and Prediction

Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts. Mainly it is used to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data. Data object whose class label is known is considered as training data. The derived model may be represented in various forms, such as *classification (IF-THEN) rules*, *decision trees*, *mathematical formulae*, *neural networks* etc.

Whereas classification predicts categorical labels, prediction models continuous-valued functions. That is, it is used to predict missing or unavailable *numerical data values* rather than class labels. Regression analysis is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Prediction also encompasses the identification of distribution *trends* based on the available data. Classification and prediction may need to be preceded by relevance analysis, which attempts to identify attributes that do not contribute to the classification or prediction process. These attributes can then be excluded.

Cluster Analysis

Unlike classification and prediction, which analyze class-labeled data objects, clustering analyzes data objects without consulting a known class label. Clustering can be used to generate such labels. The objects are clustered or grouped based on the principle of *maximizing the intra-class similarity and minimizing the interclass similarity*. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters.

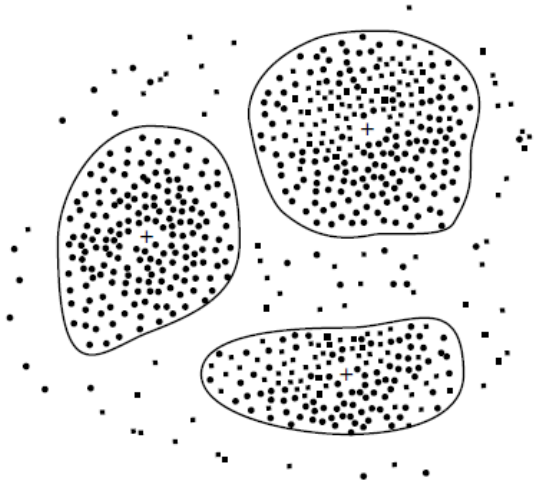


Figure: Three data clusters

Outlier Analysis

A database may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or exceptions. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are a substantial distance from any other cluster are considered outliers. For example, Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of extremely large amounts for a given account number in comparison to regular charges incurred by the same account.

Evolution Analysis

Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time. Distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis. For example, you have the major stock market (time-series) data of the last several years available from the Nepal Stock Exchange and you would like to invest in shares of high-tech industrial companies. A data mining study of stock exchange data may identify stock evolution regularities for overall stocks and for the stocks of particular companies. Such regularities may help predict future trends in stock market prices, contributing to your decision making regarding stock investments.

Major Issues in Data Mining

Major issues in data mining are about mining methodology, user interaction, performance, and diverse data types. These issues are introduced below:

Mining methodology and user interaction issues

- *Mining different kinds of knowledge in databases:* Because different users can be interested in different kinds of knowledge, data mining should cover a wide spectrum of data analysis and knowledge discovery tasks, including data characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis. These tasks may use the same database in different ways and require the development of numerous data mining techniques.
- *Interactive mining of knowledge at multiple levels of abstraction:* Because it is difficult to know exactly what can be discovered within a database, the data mining process should be interactive. Interactive mining allows users to focus the search for patterns, providing and refining data mining requests based on returned results.
- *Incorporation of background knowledge:* Domain knowledge related to databases, such as integrity constraints and deduction rules, can help focus and speed up a data mining process, or judge the interestingness of discovered patterns.
- *Data mining query languages and ad hoc data mining: H*igh-level data mining query languages need to be developed to allow users to describe ad hoc data mining tasks by facilitating the specification of the relevant sets of data for analysis, the domain knowledge, the kinds of knowledge to be mined, and the conditions and constraints to be enforced on the discovered patterns.
- *Presentation and visualization of data mining results:* Discovered knowledge should be expressed in high-level languages, visual representations, or other expressive forms so that the knowledge can be easily understood and directly usable by humans.
- *Handling noisy or incomplete data:* The data stored in a database may reflect noise, exceptional cases, or incomplete data objects. These objects may confuse the data mining process, causing the knowledge model constructed to overfit the data. Thus data cleaning methods and data analysis methods that can handle noise are required, as well as outlier mining methods for the discovery and analysis of exceptional cases.
- *Pattern evaluation – the interestingness problem:* A data mining system can uncover thousands of patterns. Many of the patterns discovered may be uninteresting to the given user, either because they represent common knowledge or lack

novelty. Thus the use of interestingness measures or user-specified constraints to guide the discovery process and reduce the search space is another active area of research.

Performance Issues

- *Efficiency and scalability of data mining algorithms: R*unning time of a data mining algorithm must be predictable and acceptable in large databases. From a database perspective on knowledge discovery, efficiency and scalability are key issues in the implementation of data mining systems.
- *Parallel, distributed, and incremental mining algorithms:* The huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods are factors motivating the development of parallel and distributed data mining algorithms. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged. Moreover, the high cost of some data mining processes promotes the need for incremental data mining algorithms that incorporate database updates without having to mine the entire data again "from scratch." Such algorithms perform knowledge modification incrementally to amend and strengthen what was previously discovered.

Issues Relating to the Diversity of Database Types

- *Handling of relational and complex types of data:* Because relational databases and data warehouses are widely used, the development of efficient and effective data mining systems for such data is important. However, other databases may contain complex data objects, hypertext and multimedia data, spatial data, temporal data, or transaction data. It is unrealistic to expect one system to mine all kinds of data, given the diversity of data types and different goals of data mining. Specific data mining systems should be constructed for mining specific kinds of data. Therefore, one may expect to have different data mining systems for different kinds of data.
- *Mining information from heterogeneous databases and global information systems:* Local- and wide-area computer networks connect many sources of data, forming huge, distributed, and heterogeneous databases. Data mining may help disclose high-level data regularities in multiple heterogeneous databases that are unlikely to be discovered by simple query systems and may improve information exchange and interoperability in heterogeneous databases. Web mining, which uncovers interesting knowledge about Web contents, Web structures, Web

usage, and Web dynamics, becomes a very challenging and fast-evolving field in data mining.

Major Issues in Data Warehousing

Building a data Warehouse is very difficult and a pain. It is challenging, but it is a fabulous project to be involved in, because when data warehouses work properly, they are magnificently useful, huge fun and unbelievably rewarding. Some of the major issues involved in building data warehouse are discussed below:

General Issues: It includes but is not limited to following issues:

- → What kind of analysis do the business users want to perform?
- → Do you currently collect the data required to support that analysis?
- → How clean is data?
- → Are there multiple sources for similar data?
- → What structure is best for the core data warehouse (i.e., dimensional or relational)?

Technical Issues: It includes but is not limited to following issues

- → How much data are you going to ship around your network, and will it be able to cope?
- → How much disk space will be needed?
- → How fast does the disk storage need to be?
- → Are you going to use SSDs to store "hot" data (i.e., frequently accessed information)?
- → What database and data management technology expertise already exists within the company?

Cultural Issues: It includes but is not limited to following issues

- → How do data definitions differ between your operational systems? Different departments and business units often use their own definitions of terms like "customer," "sale" and "order" within systems. So you'll need to standardize the definitions and add prefixes such as "all sales," "recent sales," "commercial sales" and so on.
- → What's the process for gathering business requirements? Some people will not want to spend time for you. Instead, they will expect you to use your telepathic powers to divine their warehousing and data analysis needs.

Applications of Data Warehousing

Information processing, analytical processing, and data mining are the three types of data warehouse applications that are discussed below:

- **Information Processing** - A data warehouse allows to process the data stored in it. The data can be processed by means of querying, basic statistical analysis, reporting using crosstabs, tables, charts, or graphs.
- **Analytical Processing** - A data warehouse supports analytical processing of the information stored in it. The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.
- **Data Mining** - Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using the visualization tools.

Application of Data Mining

→ **Market Analysis and Management**: Target marketing, customer relation management, market basket analysis, cross selling, market segmentation, Find clusters of customers who share the same characteristics: interest, income level, spending habits, etc. Determine customer purchasing patterns over time

→ **Risk Analysis and Management**: Forecasting, customer retention, improved underwriting, quality control, competitive analysis, credit scoring.

→ **Fraud Detection and Management**: Use historical data to build models of fraudulent behavior and use data mining to help identify similar instances. For example, detect suspicious money transactions.

→ **Sports:** Data mining can be used to analyze shots & fouls of different athletes, their weaknesses and helps athletes to assist in improving their games.

→ **Space Science:** Data mining can be used to automate the analysis image data collected from sky survey with better accuracy.

→ **Internet Web Surf-Aid:** Surf-Aid applies data mining algorithms to Web access logs for market-related pages to discover customer preference and behavior pages, analyzing effectiveness of Web marketing, improving Web site organization, etc.

→ **Social Web and Networks**: There are a growing number of highly-popular user-centric applications such as blogs, wikis and Web communities that generate a lot of structured and semi-structured information. In these applications data mining can be used to explain and predict the evolution of social networks, personalized search for social interaction, user behavior prediction etc.

Unit 2

Key Features of Data Warehouse

The key features of a data warehouse are discussed below:

- **Subject Oriented** - A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations; rather it focuses on modeling and analysis of data for decision making.
- **Integrated** - A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.
- **Time Variant** - The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view.
- **Non-volatile** - Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database are not reflected in the data warehouse.

DBMS vs. Data Warehouse

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

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

→ **Data Contents:** An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. An OLAP system manages large

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

→ **Database Design**: An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design. An OLAP system typically adopts either a *star* or *snowflake* model (to be discussed in Section 3.2.2) and a subject oriented database design.

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

→ **Access Patterns:** The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations (because most data warehouses store historical rather than up-to-date information)

Why Separate Data Warehouse?

Databases store huge amounts of data. Now the major question is *"why not perform on-line analytical processing directly on such databases instead of spending additional time and resources to construct a separate data warehouse?"* A major reason for such a separation is to help promote the *high performance of both systems*.

→ An operational database is designed and tuned from known tasks and workloads, such as indexing and hashing using primary keys, searching for particular records, and optimizing canned queries. On the other hand, data warehouse queries are often complex. They involve the computation of large groups of data at summarized levels, and may require the use of special data organization, access, and implementation methods based on multidimensional views. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.

→ Concurrency control and recovery mechanisms, such as locking and logging, are required to ensure the consistency and robustness of transactions in database systems. An OLAP query often needs read-only access of data records for summarization and aggregation. Concurrency control and recovery mechanisms,

if applied for such OLAP operations may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

→ Finally, the separation of operational databases from data warehouses is based on the different structures, contents, and uses of the data in these two systems. Decision support requires historical data, whereas operational databases do not typically maintain historical data. In this context, the data in operational databases, though abundant, is usually far from complete for decision making.

Multidimensional Data Model

Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a *data cube*.

From Tables and Spreadsheets to Data Cubes

A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by *dimensions* and *facts*. Dimensions are the entities with respect to which an organization wants to keep records. For example, an organization may create a *sales* data warehouse in order to keep records of the store's sales with respect to the dimensions *time*, *item*, *branch*, and *location*. Each dimension may have a table associated with it, called a *dimension table*. This table further describes the dimensions. For example, a dimension table for *item* may contain the attributes *item name, brand*, and *type*.

A multidimensional data model is typically organized around a central theme, like *sales*. This theme is represented by a fact table. Facts are numerical measures. Themes are the quantities by which we want to analyze relationships between dimensions. Examples of facts for a sales data warehouse include *dollars_sold* (sales amount in dollars), *units_sold* (number of units sold), and *amount budgeted*. The fact table contains the names of the *facts*, or measures, as well as keys to each of the related dimension tables.

	location = *"Chicago"*				***location*** = *"New York"*				***location*** = *"Toronto"*				***location*** = *"Vancouver"*			
	item				***item***				***item***				***item***			
	home				*home*				*home*				*home*			
time	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Figure: *Sales data for an organization according to the dimensions time, item, and location. The measure displayed is dollars_sold.*

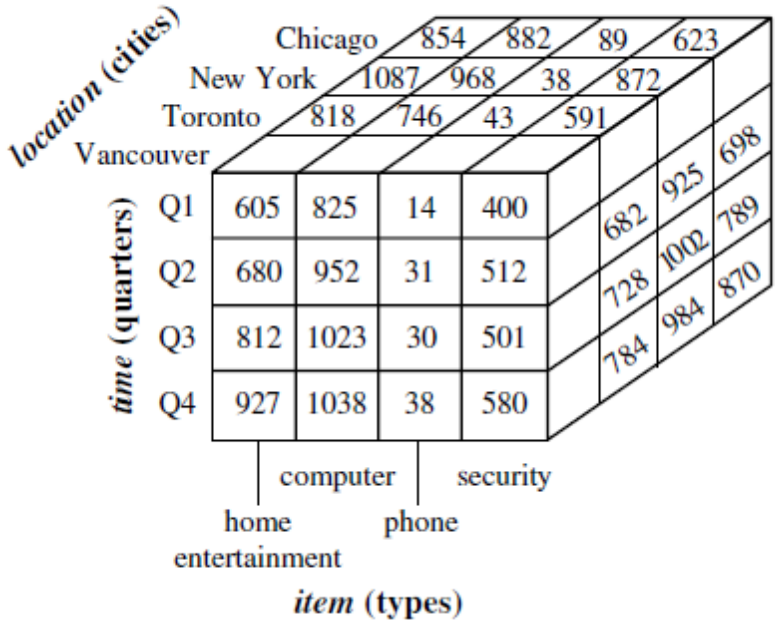


Figure A 3-D data cube representation of the data in above table, according to the dimensions time, item, and location. The measure displayed is dollars_sold (in thousands).

Suppose that we would now like to view our sales data with an additional fourth dimension, such as *supplier*. Viewing things in 4-D becomes tricky. However, we can think of a 4-D cube as being a series of 3-D cubes, as shown in Figure below.

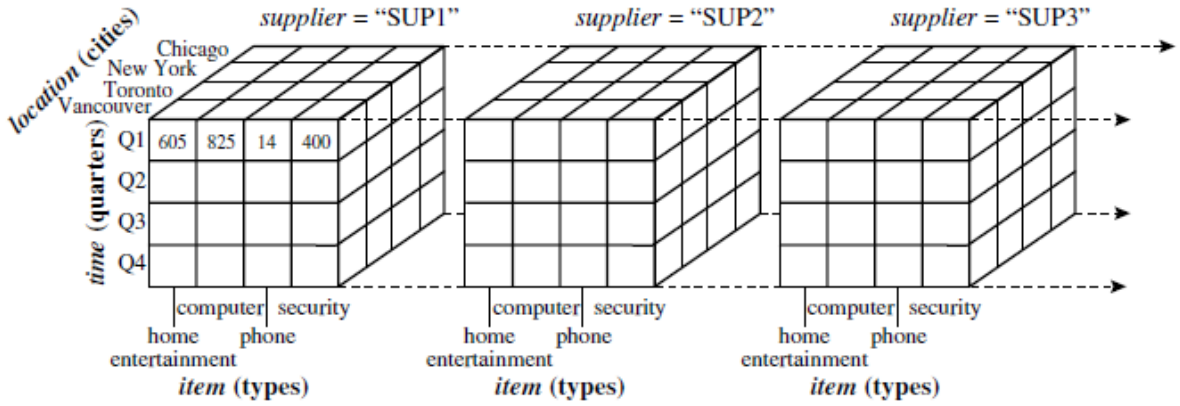


Figure 4-D data cube representation of sales data, according to the dimensions time, item, location, and supplier. The measure displayed is dollars_sold (in thousands)

If we continue in this way, we may display any *n*-dimensional data as a series of (*n*-1) dimensional cubes. The data cube is a metaphor for multidimensional data storage. The actual physical storage of such data may differ from its logical representation. The

important thing to remember is that data cubes are *n*-dimensional and do not confine data to 3-D.

Schemas for Multidimensional Database

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. A data warehouse uses *Star, Snowflake, and Fact Constellation schema*.

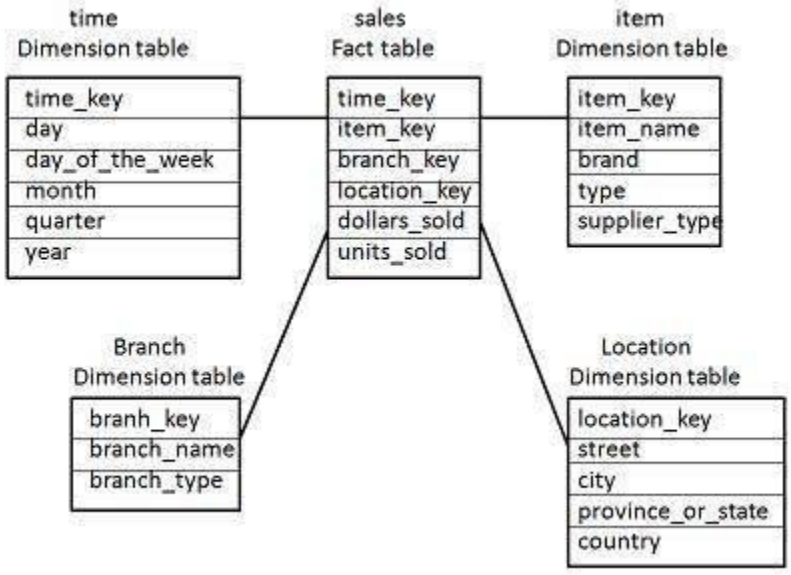
Start Schema

It is the data warehouse schema that contains two types of tables: *Fact Table and Dimension Tables.* Fact Table lies at the center point and dimension tables are connected with fact table such that star share is formed.

→ **Fact Tables:** A fact table typically has two types of columns: foreign keys to dimension tables and measures those that contain numeric facts. A fact table can contain fact's data on detail or aggregated level.

→ **Dimension Tables:** Dimension tables usually have a relatively small number of records compared to fact tables, but each record may have a very large number of attributes to describe the fact data.

Each dimension in the star schema has only one dimension table and each table holds a set of attributes. This constraint may cause data redundancy. The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location.

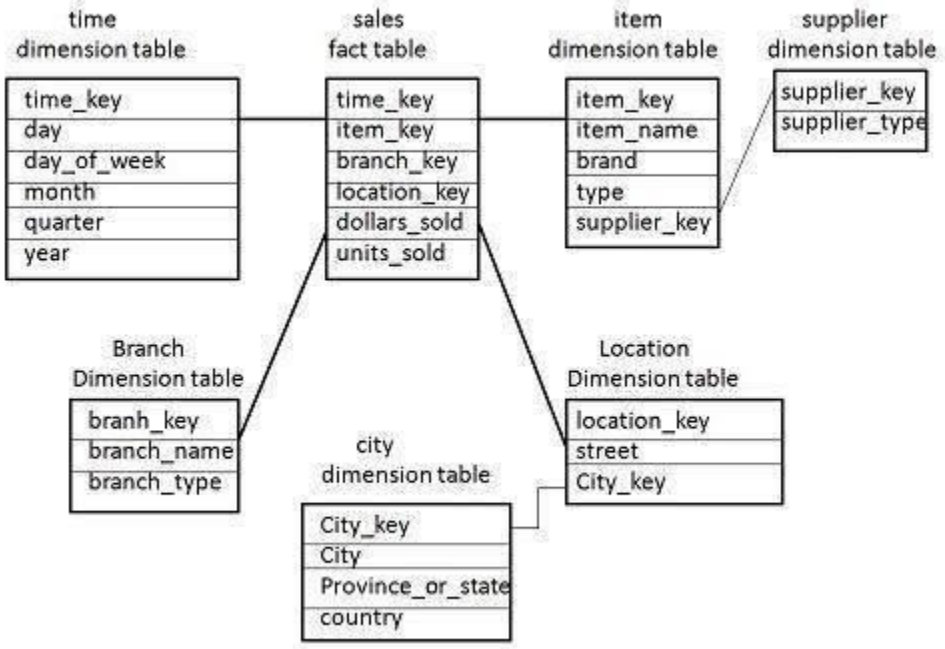


There is a fact table at the center. It contains the keys to each of four dimensions. The fact table also contains the attributes, namely dollars sold and units sold.

Since star schema contains de-normalized dimension tables, it leads to simpler queries due to lesser number of join operations and it also leads to better system performance. On the other hand it is difficult to maintain integrity of data in star schema due to de-normalized tables. It is the wifely used data warehouse schema and is also recommended by oracle

Snowflake Schema

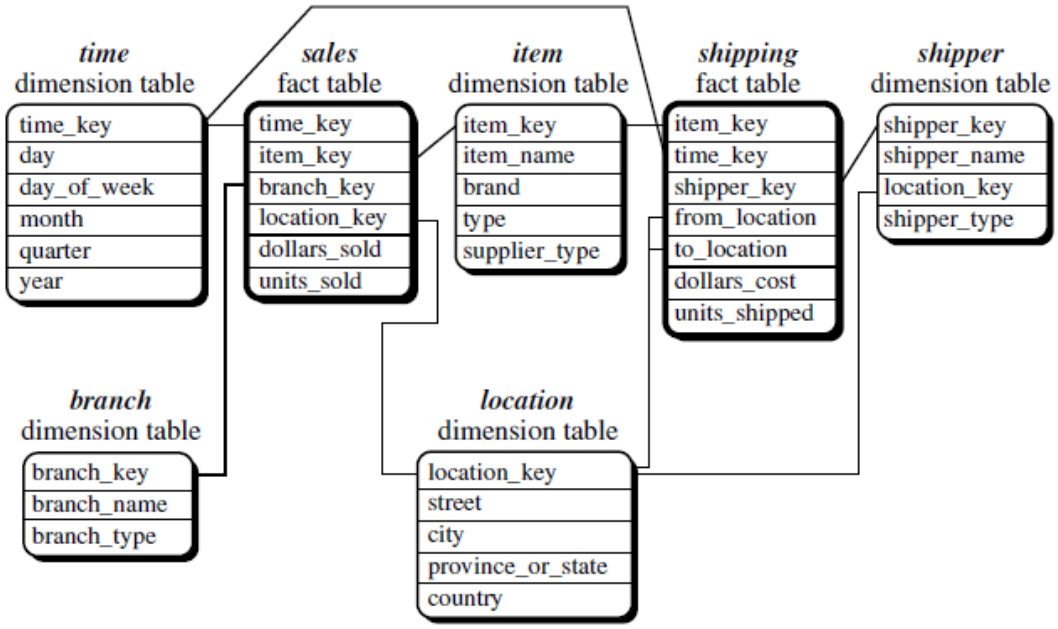
The snowflake schema is a variant of the star schema model, where some dimension tables are *normalized*, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake. For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.



Due to normalization table is easy to maintain and saves storage space. However, this saving of space is negligible in comparison to the typical magnitude of the fact table. Furthermore, the snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Fact Constellation Schema

This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation. A fact constellation schema allows dimension tables to be shared between fact tables. For example, following schema specifies two fact tables, *sales* and *shipping*. The *sales* table definition is identical to that of the star schema. The *shipping* table has five dimensions, or keys: *item key, time key, shipper key, from location*, and *to location*, and two measures: *dollars cost* and *units shipped*.



Schema Definition

Multidimensional schema is defined using Data Mining Query Language (DMQL). The two primitives, cube definition and dimension definition, can be used for defining the data warehouses and data marts.

Syntax for Cube Definition

define cube < *cube_name* > *[* < *dimension-list* > *}:* < *measure_list*>

Syntax for Dimension Definition

define dimension < *dimension_name* > *as (* < *attribute_or_dimension_list* > *)*

Star Schema Definition

The star schema that we have discussed can be defined using Data Mining Query Language (DMQL) as follows.

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(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)

Snowflake Schema Definition

Snowflake schema can be defined using DMQL as follows:

define cube sales snowflake [time, item, branch, location]:
dollars_sold = sum(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, city, province_or_state, country)

Fact Constellation Schema Definition

Fact constellation schema can be defined using DMQL as follows:

define cube sales [time, item, branch, location]:
dollars sold = sum(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]:
dollars_cost = sum(cost_in_dollars), units_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

Meta Data

Metadata is simply defined as data about data. The data that is used to represent other data is known as metadata. For example, the index of a book serves as a metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data. In terms of data warehouse, we can define metadata as follows.

- → Metadata is the road-map to a data warehouse.
- → Metadata in a data warehouse defines the warehouse objects.
- → Metadata acts as a directory. This directory helps the decision support system to locate the contents of a data warehouse.

Metadata can be broadly categorized into three categories:

- → **Business Metadata** - It has the data ownership information, business definition, and changing policies.
- → **Technical Metadata** - It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural information such as primary and foreign key attributes and indices.
- → **Operational Metadata** - It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.

Data Marts

A data mart is a subject-oriented archive that stores data and uses the retrieved set of information to assist and support the requirements involved within a particular business function or department. Data marts exist within a single organizational data warehouse repository. Data marts improve end-user response time by allowing users to have access to the specific type of data they need to view most often.

A data mart is basically a condensed and more focused version of a data warehouse that reflects the regulations and process specifications of each business unit within an organization. Each data mart is dedicated to a specific business function or region. This subset of data may span across many or all of an enterprise's functional subject areas. It is common for multiple data marts to be used in order to serve the needs of each

individual business unit (different data marts can be used to obtain specific information for various enterprise departments, such as accounting, marketing, sales, etc.).

Unit 2

Key Features of Data Warehouse

The key features of a data warehouse are discussed below:

- **Subject Oriented** - A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations; rather it focuses on modeling and analysis of data for decision making.
- **Integrated** - A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.
- **Time Variant** - The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view.
- **Non-volatile** - Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database are not reflected in the data warehouse.

DBMS vs. Data Warehouse

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

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

→ **Data Contents:** An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. An OLAP system manages large

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

→ **Database Design**: An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design. An OLAP system typically adopts either a *star* or *snowflake* model (to be discussed in Section 3.2.2) and a subject oriented database design.

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

→ **Access Patterns:** The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations (because most data warehouses store historical rather than up-to-date information)

Why Separate Data Warehouse?

Databases store huge amounts of data. Now the major question is *"why not perform on-line analytical processing directly on such databases instead of spending additional time and resources to construct a separate data warehouse?"* A major reason for such a separation is to help promote the *high performance of both systems*.

→ An operational database is designed and tuned from known tasks and workloads, such as indexing and hashing using primary keys, searching for particular records, and optimizing canned queries. On the other hand, data warehouse queries are often complex. They involve the computation of large groups of data at summarized levels, and may require the use of special data organization, access, and implementation methods based on multidimensional views. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.

→ Concurrency control and recovery mechanisms, such as locking and logging, are required to ensure the consistency and robustness of transactions in database systems. An OLAP query often needs read-only access of data records for summarization and aggregation. Concurrency control and recovery mechanisms,

if applied for such OLAP operations may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

→ Finally, the separation of operational databases from data warehouses is based on the different structures, contents, and uses of the data in these two systems. Decision support requires historical data, whereas operational databases do not typically maintain historical data. In this context, the data in operational databases, though abundant, is usually far from complete for decision making.

Multidimensional Data Model

Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a *data cube*.

From Tables and Spreadsheets to Data Cubes

A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by *dimensions* and *facts*. Dimensions are the entities with respect to which an organization wants to keep records. For example, an organization may create a *sales* data warehouse in order to keep records of the store's sales with respect to the dimensions *time*, *item*, *branch*, and *location*. Each dimension may have a table associated with it, called a *dimension table*. This table further describes the dimensions. For example, a dimension table for *item* may contain the attributes *item name, brand*, and *type*.

A multidimensional data model is typically organized around a central theme, like *sales*. This theme is represented by a fact table. Facts are numerical measures. Themes are the quantities by which we want to analyze relationships between dimensions. Examples of facts for a sales data warehouse include *dollars_sold* (sales amount in dollars), *units_sold* (number of units sold), and *amount budgeted*. The fact table contains the names of the *facts*, or measures, as well as keys to each of the related dimension tables.

	location = *"Chicago"*				***location*** = *"New York"*				***location*** = *"Toronto"*				***location*** = *"Vancouver"*			
	item				***item***				***item***				***item***			
	home				*home*				*home*				*home*			
time	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*	*ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Figure: *Sales data for an organization according to the dimensions time, item, and location. The measure displayed is dollars_sold.*

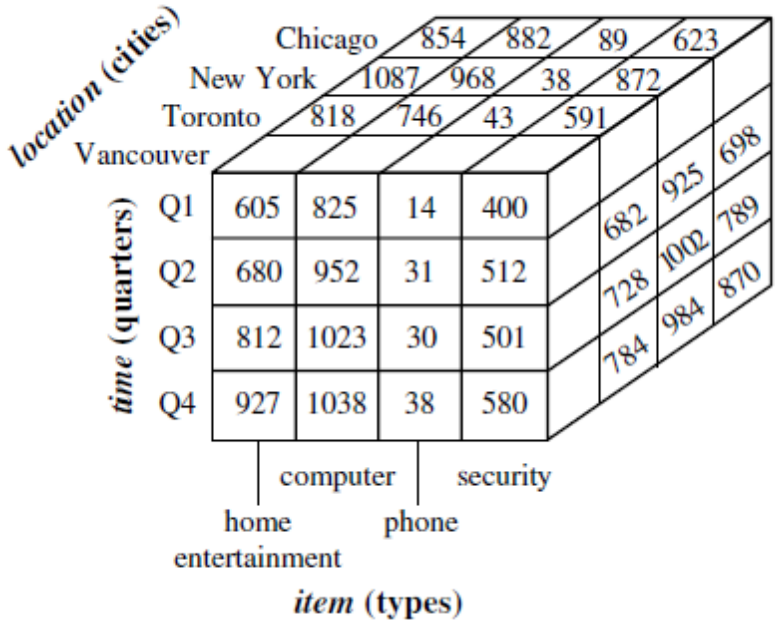


Figure A 3-D data cube representation of the data in above table, according to the dimensions time, item, and location. The measure displayed is dollars_sold (in thousands).

Suppose that we would now like to view our sales data with an additional fourth dimension, such as *supplier*. Viewing things in 4-D becomes tricky. However, we can think of a 4-D cube as being a series of 3-D cubes, as shown in Figure below.

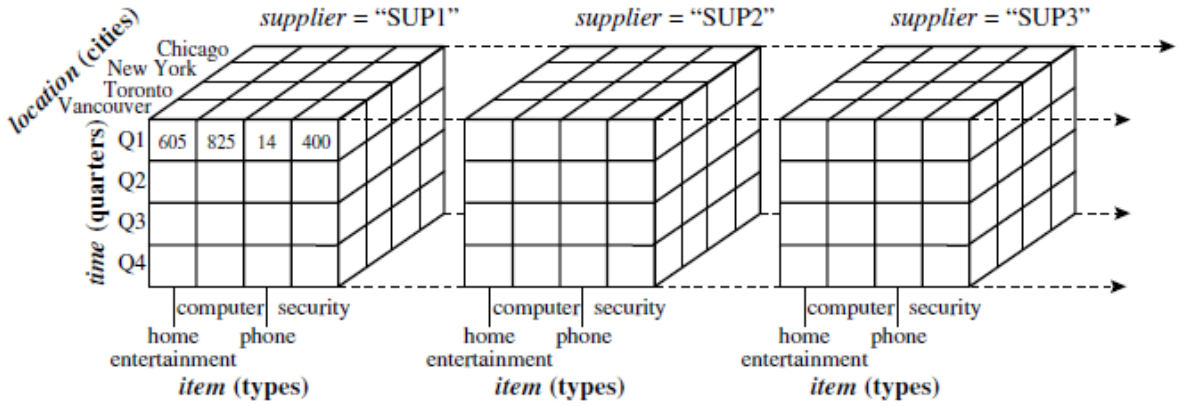


Figure 4-D data cube representation of sales data, according to the dimensions time, item, location, and supplier. The measure displayed is dollars_sold (in thousands)

If we continue in this way, we may display any *n*-dimensional data as a series of (*n*-1) dimensional cubes. The data cube is a metaphor for multidimensional data storage. The actual physical storage of such data may differ from its logical representation. The

important thing to remember is that data cubes are *n*-dimensional and do not confine data to 3-D.

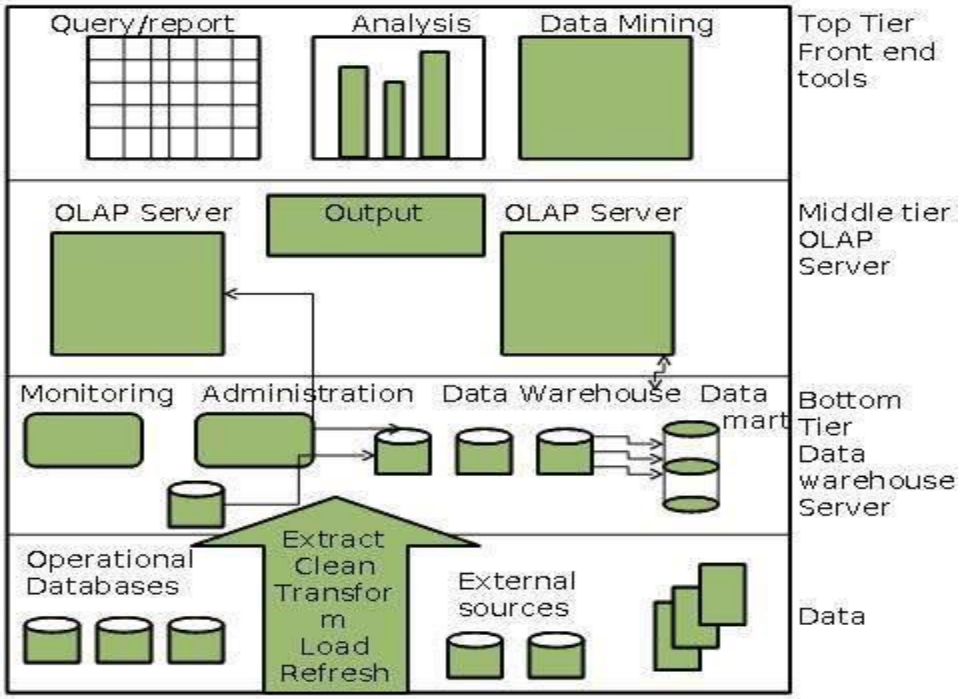
Schemas for Multidimensional Database

Unit 3

Data Warehouse Architecture

Generally a data warehouses adopts three-tier architecture. Following are the three tiers of the data warehouse architecture.

- **Bottom Tier** - The bottom tier of the architecture is the data warehouse database server. It is the relational database system. We use the back end tools and utilities to feed data into the bottom tier. These back end tools and utilities perform the Extract, Clean, Load, and refresh functions.
- **Middle Tier** - In the middle tier, we have the OLAP Server that can be implemented in either of the following ways.
    - By Relational OLAP (ROLAP), which is an extended relational database management system. The ROLAP maps the operations on multidimensional data to standard relational operations.
    - By Multidimensional OLAP (MOLAP) model, which directly implements the multidimensional data and operations.
- **Top-Tier** - This tier is the front-end client layer. This layer holds the query tools and reporting tools, analysis tools and data mining tools.



Data Warehouse Models

From the perspective of data warehouse architecture, we have the following data warehouse models:

- Virtual Warehouse
- Data mart
- Enterprise Warehouse

Virtual Warehouse

The view over an operational data warehouse is known as a virtual warehouse. It is easy to build a virtual warehouse. Building a virtual warehouse requires excess capacity on operational database servers.

Data Mart

Data mart contains a subset of organization-wide data. This subset of data is valuable to specific groups of an organization. In other words, we can claim that data marts contain data specific to a particular group. For example, the marketing data mart may contain data related to items, customers, and sales. Data marts are confined to subjects.

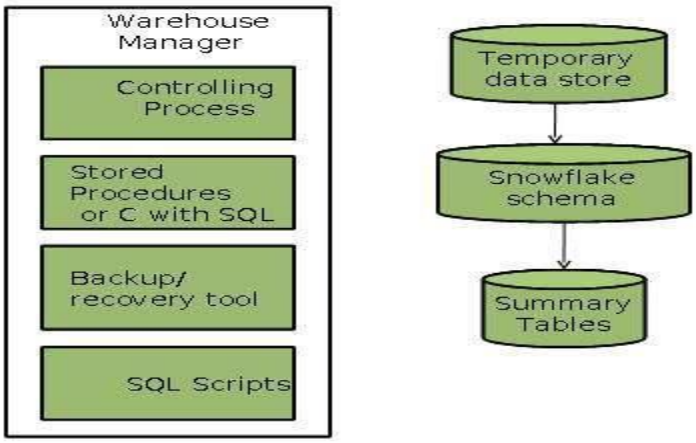
Enterprise Warehouse

An enterprise warehouse collects all the information and the subjects spanning an entire organization. It provides us enterprise-wide data integration. The data is integrated from operational systems and external information providers. This information can vary from a few gigabytes to hundreds of gigabytes, terabytes or beyond.

Warehouse Manager

A warehouse manager is responsible for the warehouse management process. It consists of third-party system software, C programs, and shell scripts. The size and complexity of warehouse managers varies between specific solutions. A warehouse manager includes the following:

- The controlling process
- Stored procedures or C with SQL
- Backup/Recovery tool
- SQL Scripts



Operations Performed by Warehouse Manager

- A warehouse manager analyzes the data to perform consistency and referential integrity checks.
- Creates indexes, business views, partition views against the base data.
- Generates new aggregations and updates existing aggregations. Generates normalizations.
- Transforms and merges the source data into the published data warehouse.
- Backup the data in the data warehouse.

Online Analytical Processing (OLAP)

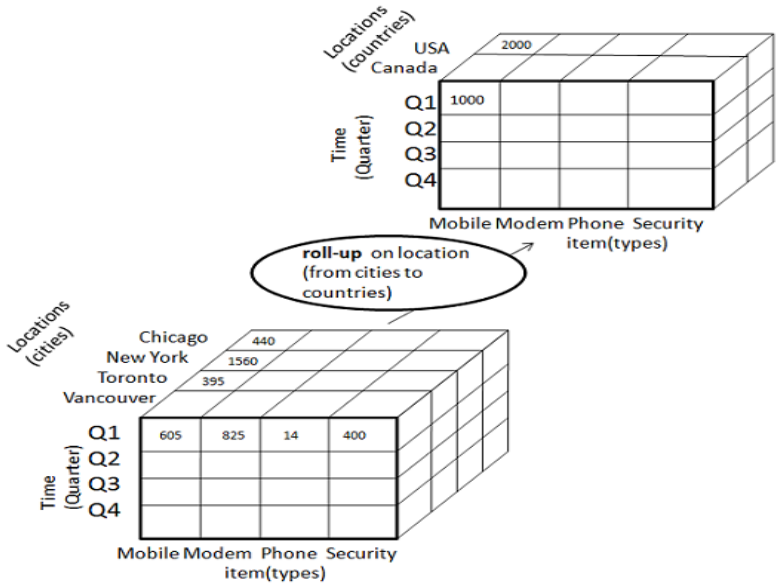
Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information.

OLAP Operations

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data. Here is the list of OLAP operations:

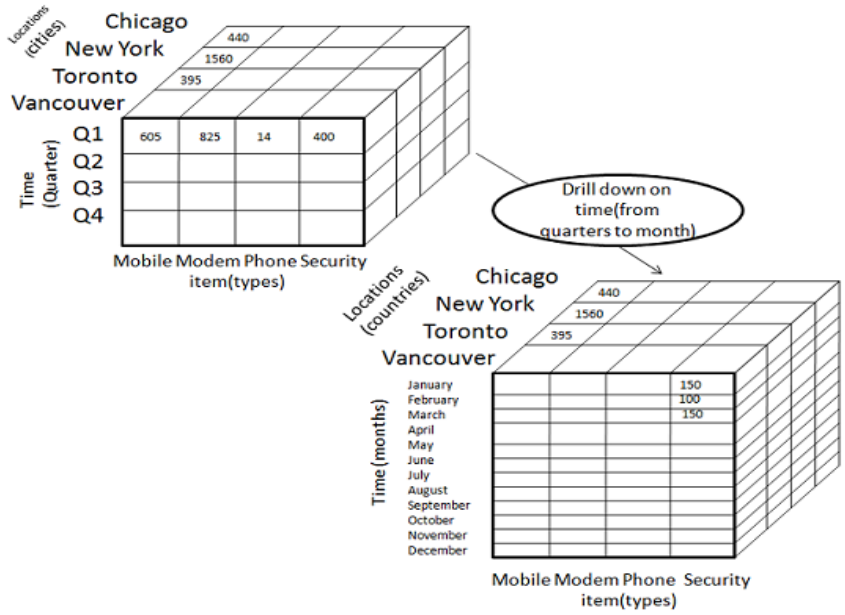
→ Roll-up
→ Drill-down
→ Slice and dice
→ Pivot (rotate)

Roll-up: Roll-up performs aggregation on a data cube in any of the following ways: By climbing up a concept hierarchy for a dimension or by dimension reduction. The following diagram illustrates how roll-up works.



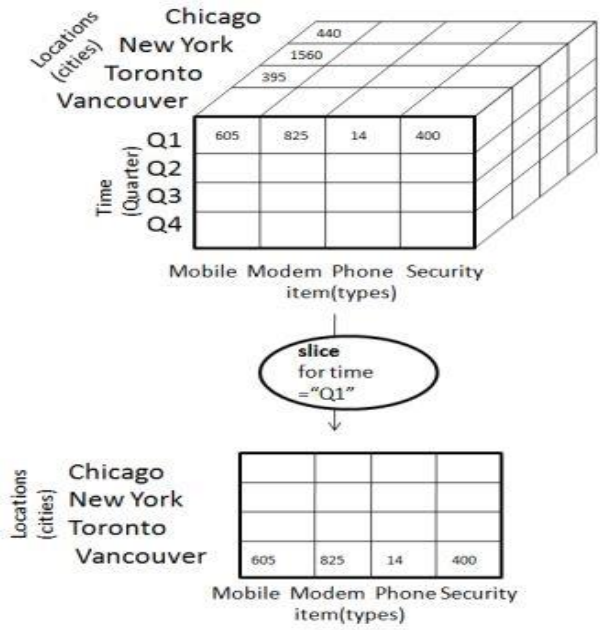
Roll-up is performed by climbing up a concept hierarchy for the dimension location. Initially the concept hierarchy was "street < city < province < country". On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country. The data is grouped into cities rather than countries. When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down: Drill-down is the reverse operation of roll-up. It is performed by either of the following ways: By stepping down a concept hierarchy for a dimension or by introducing a new dimension. The following diagram illustrates how drill-down works:



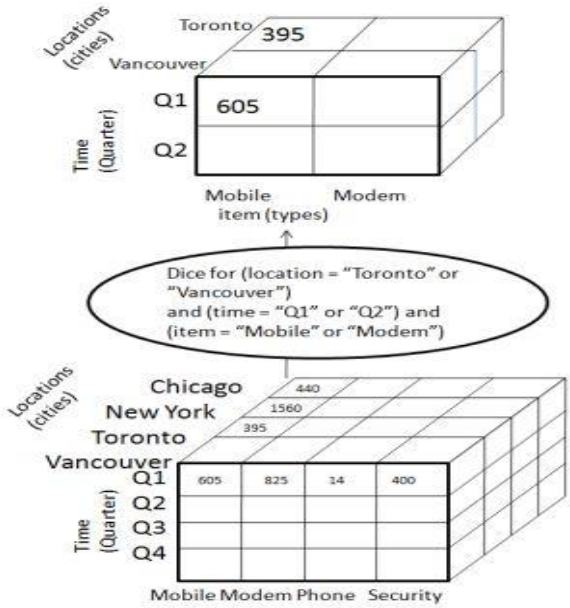
Drill-down is performed by stepping down a concept hierarchy for the dimension time. Initially the concept hierarchy was "day < month < quarter < year." On drilling down, the time dimension is descended from the level of quarter to the level of month. When drill-down is performed, one or more dimensions from the data cube are added. It navigates the data from less detailed data to highly detailed data.

Slice: The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.



Here Slice is performed for the dimension "time" using the criterion time = "Q1". It will form a new sub-cube by selecting one or more dimensions.

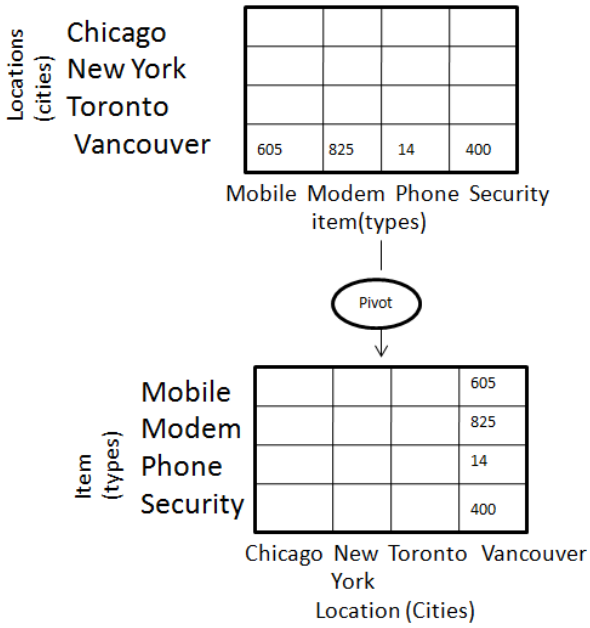
Dice: Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.



The dice operation on the cube based on the following selection criteria involves three dimensions.

- (location = "Toronto" or "Vancouver")
- (time = "Q1" or "Q2")
- (item =" Mobile" or "Modem")

Pivot: The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data. Consider the following diagram that shows the pivot operation.



OLAP vs OLTP

OLAP	OLTP
Involves historical processing of information.	Involves day-to-day processing.
OLAP systems are used by knowledge workers such as executives, managers and analysts.	OLTP systems are used by clerks, DBAs, or database professionals.
Useful in analyzing the business.	Useful in running the business.
Based on Star Schema, Snowflake, Schema and Fact Constellation Schema.	Based on Entity Relationship Model.
Provides summarized and consolidated data.	Provides primitive and highly detailed data.
Highly flexible.	Provides high performance.

Types of OLAP Servers

We have four types of OLAP servers:

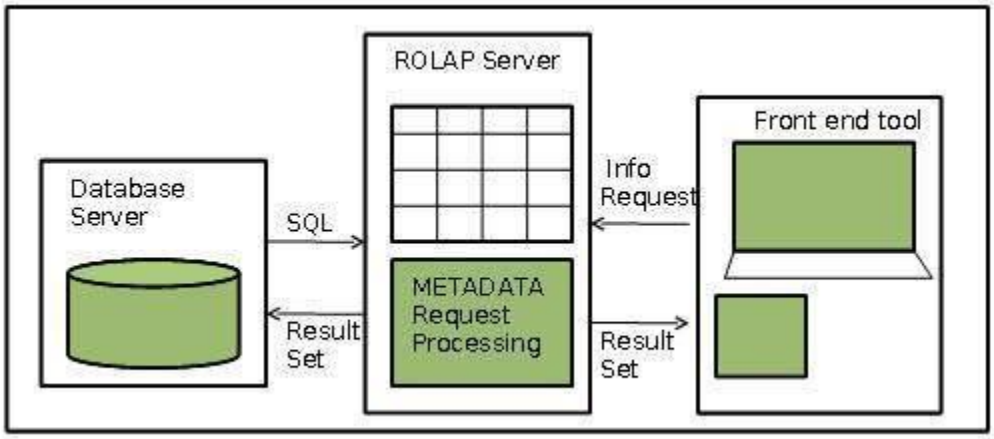
- Relational OLAP (ROLAP)
- Multidimensional OLAP (MOLAP)
- Hybrid OLAP (HOLAP)

Relational OLAP:- Relational OLAP servers are placed between relational back-end server and client front-end tools. To store and manage the warehouse data, the relational OLAP uses relational or extended-relational DBMS. ROLAP includes the following:

- Implementation of aggregation navigation logic
- Optimization for each DBMS back-end
- Additional tools and services

ROLAP includes the following components:

- Database server
- ROLAP server
- Front-end tool.



Advantages

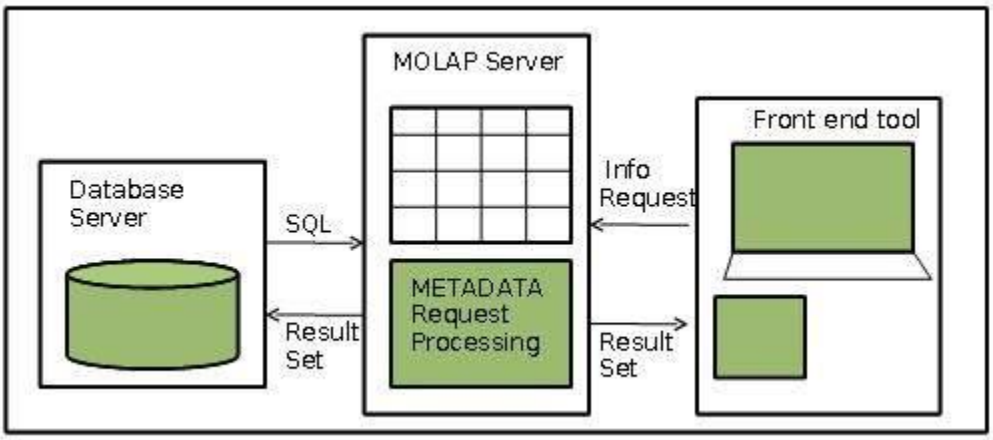
- ROLAP servers can be easily used with existing RDBMS.
- ROLAP tools do not use pre-calculated data cubes.

Disadvantages

- Poor query performance.
- Some limitations of scalability depending on the technology architecture that is utilized.

Multidimensional OLAP: Multidimensional OLAP (MOLAP) uses array-based multidimensional storage engines for multidimensional views of data. With multidimensional data stores, the storage utilization may be low if the data set is sparse. Therefore, many MOLAP servers use two levels of data storage representation to handle dense and sparse data-sets. MOLAP includes the following components:

- Database server.
- MOLAP server.
- Front-end tool.



Advantages

- MOLAP allows fastest indexing to the pre-computed summarized data.
- Easier to use, therefore MOLAP is suitable for inexperienced users.

Disadvantages

- MOLAP are not capable of containing detailed data.
- The storage utilization may be low if the data set is sparse.

Hybrid OLAP (HOLAP)

Hybrid OLAP is a combination of both ROLAP and MOLAP. It offers higher scalability of ROLAP and faster computation of MOLAP. HOLAP servers allows to store the large data volumes of detailed information. The aggregations are stored separately in MOLAP store.

MOLAP vs. ROLAP

MOLAP	ROLAP
Information retrieval is fast.	Information retrieval is comparatively slow.
Uses sparse array to store data-sets.	Uses relational table.
MOLAP is best suited for inexperienced users, since it is very easy to use.	ROLAP is best suited for experienced users.
Maintains a separate database for data cubes.	It may not require space other than available in the Data warehouse.
DBMS facility is weak.	DBMS facility is strong.

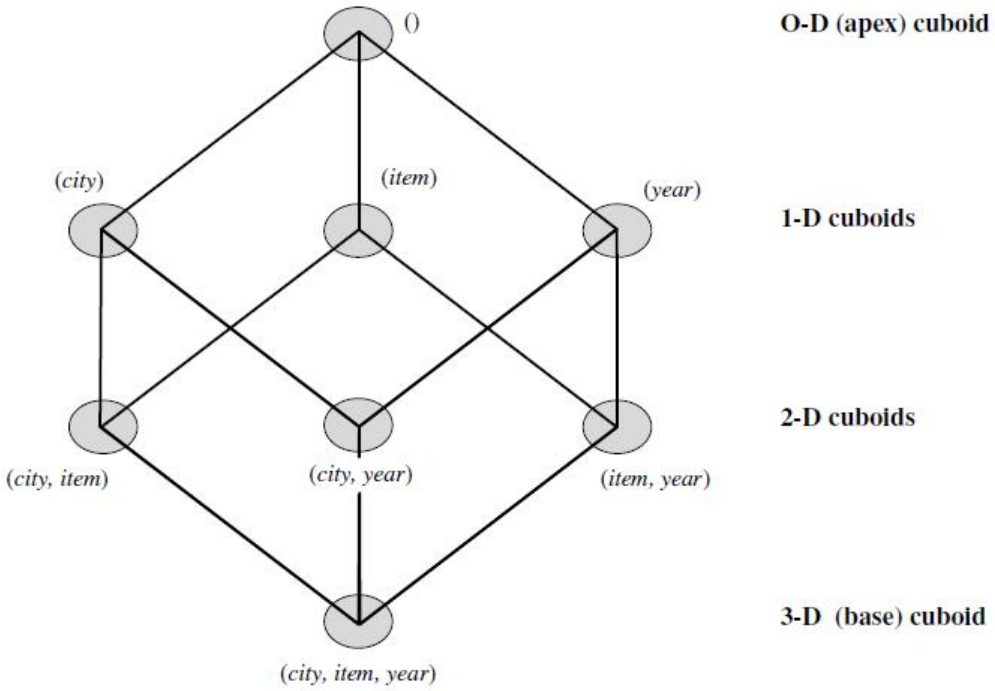
Unit 4

Computation of Data Cubes and OLAP Queries

Data warehouses contain huge volumes of data. OLAP servers demand that decision support queries be answered in the order of seconds. Therefore, it is crucial for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques. At the core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions. In SQL terms, these aggregations are referred to as group-by's. Each group-by can be represented by *cuboids*, where the set of group-by's forms a lattice of cuboids defining a data cube.

One approach to cube computation extends SQL so as to include a compute cube operator. The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation. This can require excessive storage space, especially for large numbers of dimensions. Suppose that you would like to create a data cube for *Electronics* sales that contains the following: *city, item, year*, and *sales in dollars*.

Taking the three attributes, *city, item*, and *year*, as the dimensions for the data cube, and *sales in dollars* as the measure, the total number of cuboids, or groupby's, that can be computed for this data cube is $2^3 = 8$. The possible group-by's are the following: f(*city, item, year*), (*city, item*), (*city, year*), (*item, year*), (*city*), (*item*), (*year*), ()g, where () means that the group-by is empty (i.e., the dimensions are not grouped). These group-by's form a lattice of cuboids for the data cube, as shown in Figure below. The base cuboid contains all three dimensions, *city, item*, and *year*. It can return the total sales for any combination of the three dimensions. The apex cuboid, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales. The base cuboid is the least generalized (most specific) of the cuboids. The apex cuboid is the most generalized (least specific) of the cuboids, and is often denoted as all. If we start at the apex cuboid and explore downward in the lattice, this is equivalent to drilling down within the data cube. If we start at the base cuboid and explore upward, this is akin to rolling up.



An SQL query containing no group-by, such as "compute the sum of total sales," is a *zero-dimensional operation*. An SQL query containing one group-by, such as "compute the sum of sales, group by city," is a *one-dimensional operation*. A cube operator on *n* dimensions is equivalent to a collection of group by statements, one for each subset of the *n* dimensions. Therefore, the cube operator is the *n*-dimensional generalization of the group by operator. Based on the syntax of DMQL, the data cube in this example could be defined as

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

For a cube with *n* dimensions, there are a total of 2*n* cuboids, including the base cuboid. A statement such as

compute cube sales cube

would explicitly instruct the system to compute the sales aggregate cuboids for all of the eight subsets of the set {*city, item, year*}, including the empty subset.

On-line analytical processing may need to access different cuboids for different queries. Therefore, it may seem like a good idea to compute all or at least some of the cuboids in a data cube in advance. Pre-computation leads to fast response time and avoid some redundant computation. A major challenge related to this pre-computation, however, is

that the required storage space may explode if all of the cuboids in a data cube are pre-computed, especially when the cube has many dimensions. The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels. This problem is referred to as the ***curse of dimensionality***.

If there were no hierarchies associated with each dimension, then the total number of cuboids for an *n*-dimensional data cube, as we have seen above, is 2^n. However, in practice, many dimensions do have hierarchies. For example, the dimension *time* is usually not explored at only one conceptual level, such as *year*, but rather at multiple conceptual levels, such as in the hierarchy "*day* < *month* < *quarter* < *year*". For an *n*-dimensional data cube, the total number of cuboids that can be generated is:

$$Total\ number\ of\ cuboids = \prod_{i=1}^{n}(L_i + 1),$$

Where *Li* is the number of levels associated with dimension *i*. One is added to *Li* in Equation (3.1) to include the *virtual* top level, all.

Example

If the cube has 10 dimensions and each dimension has 4 levels, what will be the number of cuboids generated?

Solution

Here n=10 Li=4 for i=1,2…….10

Thus

Total number of cuboids= $5\times5\times5\times5\times5\times5\times5\times5\times5\times5=5^{10}\approx9.8\times10^6$

From above example, it is unrealistic to pre-compute and materialize all of the cuboids that can possibly be generated for a data cube (or from a base cuboid). If there are many cuboids, and these cuboids are large in size, a more reasonable option is *partial materialization*, that is, to materialize only *some* of the possible cuboids that can be generated.

Computation of Selected Cuboids

There are three choices for data cube materialization given a base cuboid:

1. **No materialization:** Do not pre-compute any of the cuboids. This leads to computing expensive multidimensional aggregates on the fly, which can be extremely slow.

2. **Full materialization:** Pre-compute all of the cuboids. The resulting lattice of computed cuboids is referred to as the *full cube*. This choice typically requires huge amounts of memory space in order to store all of the pre-computed cuboids.
3. **Partial materialization:** Selectively compute a proper subset of the whole set of possible cuboids. Alternatively, we may compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion, such as where the tuple count of each cell is above some threshold. We will use the term *sub-cube* to refer to the latter case, where only some of the cells may be pre-computed for various cuboids. Partial materialization represents an interesting trade-off between storage space and response time.

The partial materialization of cuboids or sub-cubes should consider three factors: (1) identify the subset of cuboids or sub-cubes to materialize; (2) exploit the materialized cuboids or sub-cubes during query processing; and (3) efficiently update the materialized cuboids or sub-cubes during load and refresh.

The selection of the subset of cuboids or sub-cubes to materialize should take into account the queries in the workload, their frequencies, and their accessing costs. In addition, it should consider workload characteristics, the cost for incremental updates, and the total storage requirements. .A popular approach is to materialize the set of cuboids on which other frequently referenced cuboids are based. Alternatively, we can compute an ***iceberg cube***, which is a data cube that stores only those cube cells whose aggregate value (e.g., count) is above some minimum support threshold. Another common strategy is to materialize a ***shell cube***. This involves pre-computing the cuboids for only a small number of dimensions (such as 3 to 5) of a data cube. Queries on additional combinations of the dimensions can be computed on-the-fly. An iceberg cube can be specified with an SQL query, as shown in the following example.

```
compute cube sales iceberg as
select month, city, customer group, count(*)
from salesInfo
cube by month, city, customer group
having count(*) >= min_sup
```

The compute cube statement specifies the pre-computation of the iceberg cube, *sales iceberg*, with the dimensions *month*, *city*, and *customer group*, and the aggregate measure count(). The input tuples are in the *salesInfo* relation. The cube by clause specifies that

aggregates (group-by's) are to be formed for each of the possible subsets of the given dimensions.

Data Warehouse Back end Tools

As already mentioned, data warehousing systems use various data extraction and cleaning tools, and load and refresh utilities for populating data warehouses. Below we describe the back-end tools and utilities.

Data Cleaning

The data warehouse involves large volumes of data from multiple sources, which can lead to a high probability of errors and anomalies in the data. Inconsistent field lengths, inconsistent descriptions, inconsistent value assignments, missing entries and violation of integrity constraints are some of the examples. The three classes of data cleaning tools are popularly used to help detect data anomalies and correct them:

- **Data migration tools** allow simple transformation rules to be specified.
- **Data scrubbing tools** use domain-specific knowledge to do the scrubbing of data. Tools such as Integrity and Trillum fall in this category.
- **Data auditing tools** make it possible to discover rules and relationships by scanning data. Thus, such tools may be considered variants of data mining tools.

Load

After extracting, cleaning, and transforming, data will be loaded into the data warehouse. A load utility has to allow the system administrator to monitor status, to cancel, suspend and resume a load, and to restart after failure with no loss of data integrity. Sequential loads can take a very long time to complete especially when it deals with terabytes of data. Therefore, pipelined and partitioned parallelism are typically used. Also incremental loading over full load is more popularly used with most commercial utilities since it reduces the volume of data that has to be incorporated into the data warehouse.

Refresh

Refreshing a warehouse consists in propagating updates on source data to correspondingly update the base data and derived data stored in the warehouse. There are two sets of issues to consider: when to refresh, and how to refresh. Usually, the warehouse is refreshed periodically (e.g., daily or weekly). Only if some OLAP queries need current data, it is necessary to propagate every update. The refresh policy is set by the warehouse administrator, depending on user needs and may be different for different sources.

Refresh techniques may also depend on the characteristics of the source and the capabilities of the database servers. Extracting an entire source file or database is usually too expensive, but may be the only choice for legacy data sources. Most contemporary database systems provide replication servers that support incremental techniques for propagating updates from a primary database to one or more replicas. Such replication servers can be used to incrementally refresh a warehouse when the sources change.

Data Warehouse Tuning

The process of applying different strategies in performing different operations of data warehouse such that performance measures will enhance is called data warehousing tuning. For this, it is very important to have a complete knowledge of data warehouse. We can tune the different aspects of a data warehouse such as performance, data load, queries, etc. A data warehouse keeps evolving and it is unpredictable what query the user is going to post in the future. Therefore it becomes more difficult to tune a data warehouse system. Tuning a data warehouse is a difficult procedure due to following reasons:

→ Data warehouse is dynamic; it never remains constant.
→ It is very difficult to predict what query the user is going to post in the future.
→ Business requirements change with time.
→ Users and their profiles keep changing.
→ The user can switch from one group to another.
→ The data load on the warehouse also changes with time.

Data Warehouse - Testing

Testing is very important for data warehouse systems to make them work correctly and efficiently. There are three basic levels of testing performed on a data warehouse:

- **Unit testing**
- **Integration testing**
- **System testing**

Unit Testing

- In unit testing, each component is separately tested.
- Each module, i.e., procedure, program, SQL Script, Unix shell is tested.
- This test is performed by the developer.

Integration Testing

- In integration testing, the various modules of the application are brought together and then tested against the number of inputs.

- It is performed to test whether the various components do well after integration.

System Testing

- In system testing, the whole data warehouse application is tested together.
- The purpose of system testing is to check whether the entire system works correctly together or not.
- System testing is performed by the testing team.
- Since the size of the whole data warehouse is very large, it is usually possible to perform minimal system testing before the test plan can be enacted.

Unit 5

Data Mining Definition and Task

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it. *Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data*.

On the basis of the kind of data to be mined, there are two types of tasks that are performed by Data Mining:

- Descriptive
- Classification and Prediction

Descriptive Function

The descriptive function deals with the general properties of data in the database. Here is the list of descriptive functions −

- Class/Concept Description
- Mining of Frequent Patterns
- Mining of Associations
- Mining of Correlations
- Mining of Clusters

Classification and Prediction

Classification is the process of finding a model that describes the data classes or concepts. The purpose is to be able to use this model to predict the class of objects whose class label is unknown. This derived model is based on the analysis of sets of training data. The derived model can be presented in the following forms −

- Classification (IF-THEN) Rules
- Decision Trees
- Mathematical Formulae
- Neural Networks

Prediction is used to predict missing or unavailable numerical data values rather than class labels.

KDD versus Data Mining

KDD (Knowledge Discovery in Databases) is a field of computer science, which includes the tools and theories to help humans in extracting useful and previously

unknown information (i.e. knowledge) from large collections of digitized data. KDD consists of several steps, and Data Mining is one of them. Data Mining is application of a specific algorithm in order to extract patterns from data. Nonetheless, KDD and Data Mining are used interchangeably. In summary, Data Mining is only the application of a specific algorithm based on the overall goal of the KDD process.

Data Mining Techniques

There are several major data mining ***techniques*** have been developing and using in data mining projects recently including *association*, *classification*, *clustering*, *prediction*, *sequential patterns* and *decision tree*. We will briefly examine those data mining techniques in the following sections.

Association

Association is one of the best known data mining technique. In association, a pattern is discovered based on a relationship between items in the same transaction. That's is the reason why association technique is also known as *relation technique*. The association technique is used in *market basket analysis* to identify a set of products that customers frequently purchase together. Retailers are using association technique to research customer's buying habits. Based on historical sale data, retailers might find out that customers always buy crisps when they buy beers, and therefore they can put beers and crisps next to each other to save time for customer and increase sales.

Classification

Classification is a classic data mining technique based on machine learning. Basically classification is used to classify each item in a set of data into one of predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. In classification, we develop the software that can learn how to classify the data items into groups. For example, we can apply classification in application that "given all records of employees who left the company, predict who will probably leave the company in a future period." In this case, we divide the records of employees into two groups that named "leave" and "stay". And then we can ask our data mining software to classify the employees into separate groups.

Clustering

Clustering is a data mining technique that makes meaningful or useful cluster of objects which have similar characteristics using automatic technique. The clustering technique defines the classes and puts objects in each class, while in the classification techniques, objects are assigned into predefined classes. To make the concept clearer, we can take book management in library as an example. In a library, there is a wide range of books

in various topics available. The challenge is how to keep those books in a way that readers can take several books in a particular topic without hassle. By using clustering technique, we can keep books that have some kinds of similarities in one cluster or one shelf and label it with a meaningful name. If readers want to grab books in that topic, they would only have to go to that shelf instead of looking for entire library.

Prediction

The prediction, as it name implied, is one of a data mining techniques that discovers relationship between independent variables and relationship between dependent and independent variables. For instance, the prediction analysis technique can be used in sale to predict profit for the future if we consider sale is an independent variable, profit could be a dependent variable. Then based on the historical sale and profit data, we can draw a fitted regression curve that is used for profit prediction.

Sequential Patterns

Often used over longer-term data, sequential patterns are a useful method for identifying trends, or regular occurrences of similar events. For example, with customer data you can identify that customers buy a particular collection of products together at different times of the year. In a shopping basket application, you can use this information to automatically suggest that certain items be added to a basket based on their frequency and past purchasing history.

Decision trees

Decision tree is one of the most used data mining techniques because its model is easy to understand for users. In decision tree technique, the root of the decision tree is a simple question or condition that has multiple answers. Each answer then leads to a set of questions or conditions that help us determine the data so that we can make the final decision based on it.

Data Mining Tools

Different types of data mining tools are available in the marketplace, each with their own strengths and weaknesses. These tools use artificial intelligence, machine learning and other techniques to extract data. Most data mining tools can be classified into one of three categories: traditional data mining tools, dashboards, and text-mining tools. Below is a description of each.

→ ***Traditional Data Mining Tools:*** Traditional data mining programs help companies establish data patterns and trends by using a number of complex algorithms and techniques. Some of these tools are installed on the desktop to monitor the data and highlight trends and others capture information residing outside a database. The majority are available in both Windows and UNIX

versions, although some specialize in one operating system only. In addition, while some may concentrate on one database type, most will be able to handle any data using online analytical processing or a similar technology.

→ ***Dashboards:*** Installed in computers to monitor information in a database, dashboards reflect data changes and updates onscreen — often in the form of a chart or table — enabling the user to see how the business is performing. Historical data also can be referenced, enabling the user to see where things have changed (e.g., increase in sales from the same period last year). This functionality makes dashboards easy to use and particularly appealing to managers who wish to have an overview of the company's performance.

→ ***Text-mining Tools:*** The third type of data mining tool sometimes is called a text-mining tool because of its ability to mine data from different kinds of text — from Microsoft Word and Acrobat PDF documents to simple text files, for example. These tools scan content and convert the selected data into a format that is compatible with the tool's database, thus providing users with an easy and convenient way of accessing data without the need to open different applications. Scanned content can be unstructured (i.e., information is scattered almost randomly across the document, including e-mails, Internet pages, audio and video data) or structured (i.e., the data's form and purpose is known, such as content found in a database). Capturing these inputs can provide organizations with a wealth of information that can be mined to discover trends, concepts, and attitudes.

Trends in Data Mining

Data mining concepts are still evolving and here are the latest trends that we get to see in this field −

- Application Exploration.
- Scalable and interactive data mining methods.
- Integration of data mining with database systems, data warehouse systems and web database systems.
- Standardization of data mining query language.
- Visual data mining.
- New methods for mining complex types of data.
- Biological data mining.
- Data mining and software engineering.
- Web mining.

- Distributed data mining.
- Real time data mining.
- Multi database data mining.
- Privacy protection and information security in data mining.

Unit 6

Data Mining Query Languages

How can we integrate data mining more closely with traditional database systems, particularly querying?" Three possible ways are there.

- DMQL: A Data Mining Query Language for Relational Databases
- Integrating Data Mining with SQL Databases: OLE DB for Data Mining
- MSQL: A Query Language for Database Mining

The Data Mining Query Language (DMQL) was proposed by Han, Fu, Wang, et al. for the DBMiner data mining system. The Data Mining Query Language is actually based on the Structured Query Language (SQL).Data Mining Query Languages can be designed to support ad hoc and interactive data mining. The DMQL can work with databases and data warehouses as well. DMQL can be used to define data mining tasks.

The language adopts an SQL-like syntax, so that it can easily be integrated with the relational query language SQL. The syntax of DMQL is defined in an extended BNF grammar, where "[ ]" represents 0 or one occurrence, "{ }" represents 0 or more occurrences, and words in sans serif font represent keywords.

Syntax for Task-Relevant Data Specification

Here is the syntax of DMQL for specifying task-relevant data −

use database database_name
or
use data warehouse data_warehouse_name
in relevance to att_or_dim_list
from relation(s)/cube(s) [where condition]
order by order_list
group by grouping_list

Example

Use database ABCompany_db
In relevance to I .name,I.price, C.income, C.age
From customer C, item I, purchases P, items_sold S

Where I.item_ID=S. item.JD and S.trans_ID =P.trans_ID and P .custJD=C.cust_ID and C. country –"Sri Lanka" Group by p.data

Syntax for Specifying the Kind of Knowledge

Here we will discuss the syntax for Characterization, Discrimination, Association, Classification, and Prediction.

Characterization

The syntax for characterization is −

mine characteristics [as pattern_name]
analyze {measure(s) }

The analyze clause, specifies aggregate measures, such as count, sum, or count%. For example − Description describing customer purchasing habits.

mine characteristics as customerPurchasing
analyze count%

Discrimination

The syntax for Discrimination is −

mine comparison [as {pattern_name]}
For {target_class } where {target_condition }
{versus {contrast_class_i }
where {contrast_condition_i}}
analyze {measure(s) }

For example, a user may define big spenders as customers who purchase items that cost $100 or more on an average; and budget spenders as customers who purchase items at less than $100 on an average. The mining of discriminant descriptions for customers from each of these categories can be specified in the DMQL as −

mine comparison as purchaseGroups
for bigSpenders where avg(I.price) ≥$100
versus budgetSpenders where avg(I.price)< $100
analyze count

Association

The syntax for Association is−

mine associations [as {pattern_name}]
{matching {metapattern} }

For Example

mine associations as buyingHabits

matching P(X:customer,W) ^ Q(X,Y) ≥ buys(X,Z)

Where X is key of customer relation; P and Q are predicate variables; and W, Y, and Z are object variables, Such as

age(X,"20-30") ^ *inclome(X,"40-50K") ≥ buys (X, "Computer")*

This rule states that customers in their thirties, with an annual income of between 40K and 50K, are likely to purchase a Computer.

Classification

The syntax for Classification is −

mine classification [as pattern_name]
analyze classifying_attribute_or_dimension

Example:

mine classifications as classifyCustomerCreditRating
analyze credit_rating

For categorical attributes or dimensions, each value represents a class (such as low-risk, medium risk, high risk). For numeric attributes, each class defined by a range (such as 20-39, 40-59, 60-89 for age)

Prediction

The syntax for prediction is

mine prediction [as pattern_name]
analyze prediction_attribute_or_dimension
{set {attribute_or_dimension_i= value_i}}

Syntax for Concept Hierarchy Specification

To specify concept hierarchies, use the following syntax

use hierarchy <hierarchy> for <attribute_or_dimension>

We use different syntaxes to define different types of hierarchies such as−

- schema hierarchies

  define hierarchy time_hierarchy on date as [date,month quarter,year]

- set-grouping hierarchies

  define hierarchy age_hierarchy for age on customer as
  level1: {young, middle_aged, senior} < level0: all
  level2: {20, ..., 39} < level1: young
  level3: {40, ..., 59} < level1: middle_aged

level4: {60, ..., 89} < level1: senior

- Operation-derived hierarchies

define hierarchy age_hierarchy for age on customer as
{age_category(1), ..., age_category(5)}
:= cluster(default, age, 5) < all(age)

This statement says that hierarchy is generated by default clustering algorithm with 5 as *fan-out* value. Fan-out value defined levels in tree while generating concept hierarchy.

- Rule-based hierarchies

define hierarchy profit_margin_hierarchy on item as
level_1: low_profit_margin < level_0: all

if (price - cost)< $50
level_1: medium-profit_margin < level_0: all

if ((price - cost) > $50) and ((price - cost) ≤ $250))
level_1: high_profit_margin < level_0: all

Syntax for Interestingness Measures Specification

- Interestingness measures and thresholds can be specified by the user with the statement

with <interest_measure_name> threshold = threshold_value

For Example −

with support threshold = 0.05
with confidence threshold = 0.7

Syntax for Pattern Presentation and Visualization Specification

We have a syntax, which allows users to specify the display of discovered patterns in one or more forms.

display as <result_form>

For Example −

display as table

Full Specification of DMQL

As a market manager of a company, you would like to characterize the buying habits of customers who can purchase items priced at no less than $100; with respect to the customer's age, type of item purchased, and the place where the item was purchased.

You would like to know the percentage of customers having that characteristic. In particular, you are only interested in purchases made in Canada, and paid with an American Express credit card. You would like to view the resulting descriptions in the form of a table.

```
use database AllElectronics_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age,I.type,I.place_made
from customer C, item I, purchase P, items_sold S,  branch B
where I.item_ID = S.item_ID and P.cust_ID = C.cust_ID and
P.method_paid = "AmEx" and B.address = "Canada" and I.price ≥ 100
with noise threshold = 5%
display as table
```

Data Mining Languages Standardization

Standardizing the Data Mining Languages will serve the following purposes −

- Helps systematic development of data mining solutions.
- Improves interoperability among multiple data mining systems and functions.
- Promotes education and rapid learning.
- Promotes the use of data mining systems in industry and society.

Unit 7

Association Rule Mining

Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. An example of an association rule would be "*If a customer buys a dozen eggs, he is 80% likely to also purchase milk.*" An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Association rule mining is a method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. For example, the rule found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing. In addition to the above example from market basket analysis association rules are employed today in many application areas including web usage mining, intrusion detection, bioinformatics etc.

The problem of association rule mining is defined as: Let $I = \{I1, I2 \ldots\ldots\ldots In\}$ be a set of binary attributes called *items*. Let $D = \{T1, T2 \ldots\ldots\ldots Tm\}$ be a set of transactions called the *database*. Each *transaction* in has a unique transaction ID and contains a subset of the items in A *rule* is defined as an implication of the form:

$$X => Y$$

Where $X, Y \subseteq I$ and $X \cap Y = \Phi$

Every rule is composed by two different set of items, also known as X and Y *itemsets* and, where X is called *antecedent* or left-hand-side (LHS) and Y is *consequent* or right-hand-side (RHS).

Transaction ID	Milk	Bread	Butter	Beer	Diaper
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

To illustrate the concepts, we use a small example from the supermarket domain. The set of items in above table shows a small database containing the items, where, in each entry, the value 1 means the presence of the item in the corresponding transaction, and the value 0 represent the absence of an item in a that transaction. An example rule for the supermarket could be $\{butter, bread\} => \{milk\}$ meaning that if butter and bread are bought, customers also buy milk.

Support and Confidence

In order to select interesting rules from the set of all possible rules, constraints on various measures of significance and interest are used. The best-known constraints are minimum thresholds on *support and confidence*.

Support of association rule $A => B$ is the percentage of transactions in *dataset* that contain both items. In formula

$$Support(A => B) = P(A \cup B)$$

For example, in above data-set, the association rule $bread => milk$ has a support of *2/5* since both items occurs in *40%* of all transactions (2 out of 5 transactions).

Confidence of association rule $A => B$ with respect to set of transactions T in *dataset D is* the percentage of transactions in D containing A that also contain B. In formula

$$Confidence(A => B) = P(B \mid A) = P(A \cup B) / P(A) = Support(A => B) / P(A)$$

For example, in above data-set, the association rule $bread => milk$ has a confidence of *2/3,* since *66.66%* of all transactions containing *bread* also contains *milk*.

Rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong. By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

Why Association Mining

In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in shopping basket data analysis, product clustering, and catalog design and store layout.
Programmers use association rules to build programs capable of machine learning

Apriori Algorithm

It is a classic algorithm used in data mining for learning association rules. It is very simple. Learning association rules basically means finding the items that are purchased together more frequently than others. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent item set properties.

Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are used to explore (*k*+1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted *L*1.Next, *L*1 is used to find *L*2, the set of frequent 2-itemsets, which is used to find *L*3, and so on, until no more frequent *k*-itemsets can be found. The finding of each *Lk* requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property, presented below, is used to reduce the search space. *Apriori Property states that any subset of frequent item set must be frequent*.

Example

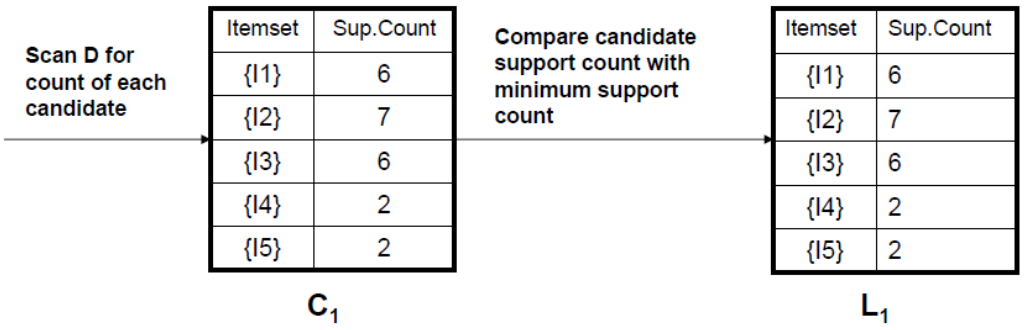
TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Consider a database, D, consisting of 9 transactions. Suppose min. support count required is 2 (i.e. min-sup = 2/9 = 22 %). Let minimum confidence required is 70%. We have to first find out the frequent item set using Apriori algorithm. Then, Association rules will be generated using min. support & min. confidence.

Solution

Step 1: Generating 1-itemset Frequent Pattern

The set of frequent 1-itemsets, L1, consists of the candidate 1-itemsets satisfying minimum support. In the first iteration of the algorithm, each item is a member of the set of candidate.



Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

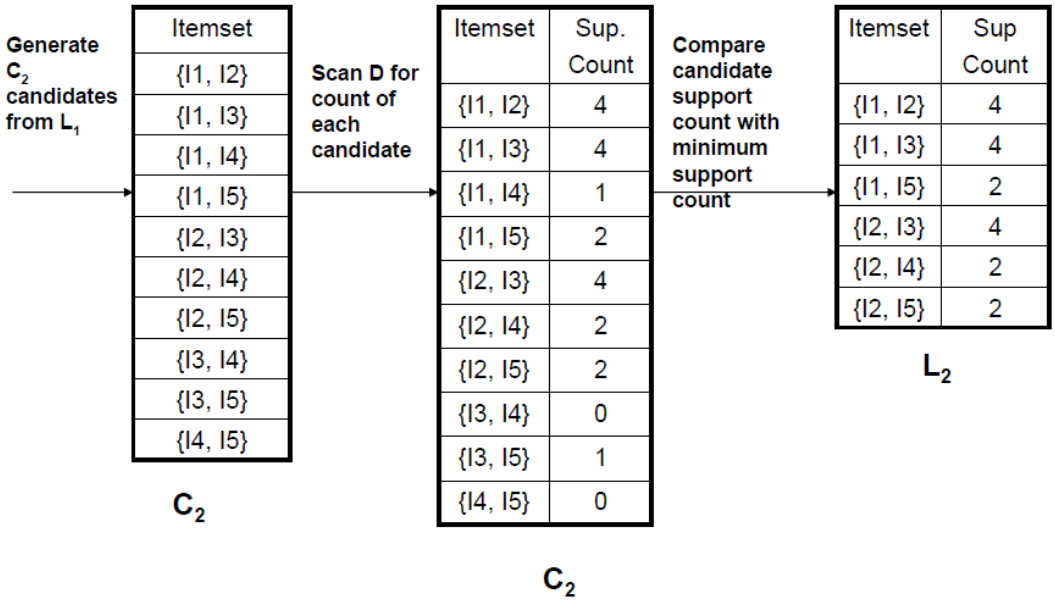
Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2: Generating 2-itemset Frequent Pattern

To discover the set of frequent 2-itemsets, L2, the algorithm uses L1 *Join* L1to generate a candidate set of 2-itemsets, C2. Next, the transactions in D are scanned and the support count for each candidate item set in C2 is accumulated (as shown in the middle table). The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.



Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

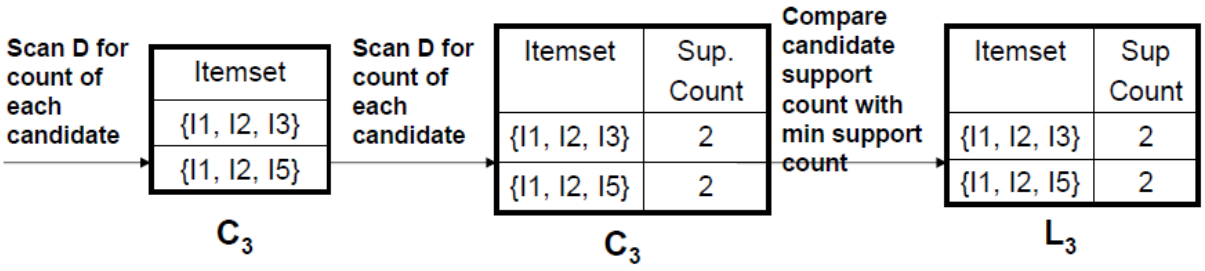
Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 3: Generating 3-itemset Frequent Pattern

The generation of the set of candidate 3-itemsets, C3, involves use of the Apriori Property. In order to find C3, we compute L2*Join*L2. C3= L2 *Join*L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}. Now, Join step is complete and Prune step will be used to reduce the size of C3. Prune step helps to avoid heavy computation due to large Ck.

Based on the Apriori property that all subsets of a frequent item set must also be frequent, we can determine that four latter candidates cannot possibly be frequent. For example, let's take {I1, I2, I3}.The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, We will keep {I1, I2, I3} in C3. Lets take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}. BUT, {I3, I5} is not a member of L2and hence it is not frequent violating Apriori Property. Thus we will have to remove {I2, I3, I5} from C3. Therefore, C3= {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning. Now, the transactions in D are scanned in order to determine L3, consisting of those candidates 3-itemsets in C3 having minimum support.



Step 4: Generating 4-itemset Frequent Pattern

The algorithm uses L3 *Join*L3to generate a candidate set of 4-itemsets, C4. Although the join results in {{I1, I2, I3, I5}}, this item set is pruned since its subset {{I2, I3, I5}}is not frequent. Thus, C4= φ, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.

These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets
Procedure:
For each frequent item set ***"l"***, generate all nonempty subsets of ***l.*** For every nonempty subset ***s*** of ***l***, output the rule "$s => l - s$"if support_count(l) / support_count(s) >= **min_conf** where min_conf is minimum confidence threshold.

Back To Example
We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
Let's take ***l*** = {I1, I2, I5}. It's all nonempty subsets are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, {I5}.

Let minimum confidence threshold is, say 70%. The resulting association rules are shown below, each listed with its confidence.
R1: I1 ^ I2 =>I5
Confidence = support_count {I1, I2,

I5}/support_count{I1,I2} = 2/4 = 50%, R1 is Rejected.
R2: I1 ^ I5 =>I2
Confidence = support_count {I1,I2,I5}/ support_count {I1,I5} = 2/2 = 100%, R2 is Selected.
R3: I2 ^ I5 =>I1
Confidence = support_count {I1,I2,I5}/ support_count {I2,I5} = 2/2 = 100%, R3 is Selected.
R4: I1 =>I2 ^ I5
Confidence = support_count {I1,I2,I5}/ support_count {I1} = 2/6 = 33%, R4 is Rejected.
R5: I2 =>I1 ^ I5
Confidence = support_count {I1,I2,I5}/ support_count {I2} = 2/7 = 29%, R5 is Rejected.
R6: I5 =>I1 ^ I2
Confidence = support_count {I1,I2,I5}/ support_count {I5} = 2/2 = 100%, R6 is Selected.

In this way, we have found three strong association rules.

Improving Efficiency of Apriori Algorithm

Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the original algorithm. Several of these variations are summarized as follows:

Hash-based technique

A hash-based technique can be used to reduce the size of the candidate *k*-itemsets, *Ck*, for $k > 1$. For example, when scanning each transaction in the database to generate the frequent 1-itemsets, *L*1, from the candidate 1-itemsets in *C*1, we can generate all of the 2-itemsets for each transaction, hash them into the different *buckets* of a *hash table* structure, and increase the corresponding bucket counts. A 2-itemset whose corresponding bucket count in the hash table is below the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash-based technique may substantially reduce the number of the candidate *k*-itemsets examined.

Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y)) \bmod 7$

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

Transaction Reduction

It reduces the number of transactions scanned in future iterations. A transaction that does not contain any frequent *k*-itemsets cannot contain any frequent (*k*+1)-itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent scans of the database for *j*-itemsets, where $j > k$, will not require it.

Partitioning

The set of transactions may be divided into a number of disjoint subsets. Then each partition is searched for frequent itemsets. These frequent itemsets are called local frequent itemsets. *Any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions*. Therefore, all local frequent itemsets are candidate itemsets with respect to *D*. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to *D*.

Sampling

A random sample (usually large enough to fit in the main memory) may be obtained from the overall set of transactions and the sample is searched for frequent itemsets. These frequent itemsets are called *sample frequent itemsets*. Because we are searching for frequent itemsets in *S* rather than in *D*, it is possible that we will miss some of the global

frequent itemsets. To lessen this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to *S*

Unit 8

Introduction to Clustering

The process of grouping a set of physical or abstract objects into classes of *similar* objects is called clustering. A cluster is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters. A cluster of data objects can be treated collectively as one group. Although classification is an effective means for distinguishing groups or classes of objects, it requires the often costly collection and labeling of a large set of training tuples or patterns, which the classifier uses to model each group.

Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their *similarity*. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and frequent purchases, may be of interest as possible fraudulent activity.

Measure of Distance between Data Points

During clustering of data objects, we need to find distance between data objects. The most popular distance measure is *Euclidean distance*, which is defined as:

$$d(x,y) = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2}$$

Where, $x = (x_1, y_1)$ *and* $y = (x_2, y_2)$

Another well-known metric is *Manhattan (or city block) distance*, defined as

$$d(x,y) = |x_2 - x_1| + |y_2 - y_1|$$

Minkowski distance is a generalization of both Euclidean distance and Manhattan distance. It is defined as

$$d(x,y) = \left(|x_2 - x_1|^p + |y_2 - y_1|^p\right)^{1/p}$$

Where p is a positive integer, such a distance is also called L_p norm, in some literature. It represents the Manhattan distance when $p = 1$ (i.e., L_1 norm) and Euclidean distance when $p = 2$ (i.e., L_2 norm).

Categorization of Clustering Algorithms

Many clustering algorithms exist in the literature. In general, the major clustering methods can be classified into the following categories.

→ **Partitioning methods**: Given a database of n objects or data tuples, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k < n$. Given k, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another.

→ **Hierarchical methods:** A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*. The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds. The *divisive approach*, also called the *top-down* approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

→ **Density-based methods:** Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. Other clustering methods have been developed based on the notion of *density*. Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold.

→ **Model-based methods:** Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model. EM is an algorithm that performs expectation-maximization analysis based on statistical modeling.

K-means Clustering Algorithm

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed

in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.

Algorithmic steps for k-means clustering

Let X = {x_1,x_2,x_3,……..,x_n} be the set of data points and V = {v_1,v_2,…….,v_c} be the set of centers.

1) Randomly select '*c*' cluster centers.
2) Calculate the distance between each data point and cluster centers.
3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..
4) Recalculate the new cluster center using: $1/c_i \sum_{i=1}^{c_i} x_i$, where, '$c_i$' represents the number of data points in i^{th} cluster.
5) Recalculate the distance between each data point and new obtained cluster centers.
6) If no data point was reassigned then stop, otherwise repeat from step 3.

Advantages

→ Fast, robust and easier to understand.
→ Gives best result when data set are distinct or well separated from each other.

Disadvantages

→ The learning algorithm requires apriori specification of the number of cluster centers.
→ Randomly choosing of the cluster center cannot lead us to the fruitful result.
→ Applicable only when mean is defined i.e. fails for categorical data.
→ Algorithm fails for non-linear data set.

Example

Divide the data points {(1,1), ((2,1), (4,3), (5,4)} into two clusters.

Solution

Let p1=(1,1) p2=(2,1) p3=(4,3) p4=(5,4)

Initial step

Let c1=(1,1) and c2=(2,1) are two initial cluster centers.

Iteration 1

Calculate distance between clusters centers and each data points

$d(c1, p1) = 0$ $\quad$ $d(c2, p1) = \sqrt{(2-1)^2 + (1-1)^2} = 1$

$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1$ $\quad$ $d(c2, p2) = \sqrt{(2-2)^2 + (1-1)^2} = 0$

$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6$ $\quad$ $d(c2, p3) = \sqrt{(4-2)^2 + (3-1)^2} = 2.83$

$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5$ $\quad$ $d(c2, p4) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$

Thus after first iteration

Cluster 1= {p1}

Cluster 2={p2,p3,p4}

Now, new cluster centers are:

c1=(1,1) and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

Iteration 2

Calculate distance between new cluster centers and each data points

$d(c1, p1) = 0$ $\quad$ $d(c2, p1) = \sqrt{(11/3-1)^2 + (8/3-1)^2} = 3.14$

$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1$ $\quad$ $d(c2, p2) = \sqrt{(11/3-2)^2 + (8/3-1)^2} = 2.35$

$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6$ $\quad$ $d(c2, p3) = \sqrt{(11/3-4)^2 + (8/3-3)^2} = 0.46$

$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5$ $\quad$ $d(c2, p4) = \sqrt{(11/3-5)^2 + (8/3-4)^2} = 1.88$

Thus after second iteration

Cluster 1= {p1,p2}

Cluster 2={p3,p4}

Now, new cluster centers are:

c1={(1+2)/2,(1+1)/2}={3/2,1} and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

Repeat this process till no re-assignment of points to groups.

K-medoid Clustering Algorithm

A *medoid* can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. i.e. it is a most centrally located point in the cluster. In contrast of K-means algorithm, K-medoid algorithm chooses data point as centers and works with arbitrary matrix of distances instead of l_2.*K-medoid* is a classical partitioning

technique of clustering that clusters the data set of n objects into k clusters known *a priori*. It is more robust to noise and outliers as compared to because it may minimize a sum of pair-wise dissimilarities instead of a sum of squared Euclidean distances.

Algorithms

The most common realization of k-medoid clustering is the **Partitioning around Medoid (PAM)** algorithm and is as follows:

1. Initialize: randomly select (without replacement) k of the n data points as the medoid
2. Associate each data point to the closest medoid.
3. While the cost of the configuration decreases:
   - For each medoid m, for each non-medoid data point o:
     - Swap m and o, re-compute the cost (sum of distances of points to their medoid)
     - If the total cost of the configuration increased in the previous step, undo the swap

Example

Cluster the following data set of ten objects into two clusters i.e. $k = 2$. Data Points are {(1,3), (4,2), (6,2), (3,5), (4,1)}

Solution

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1)

Initial step

Let m1=p1=(1,3) and m2=p4=(3,4) are two initial medoid

Iteration 1

Calculate distance between clusters centers and each data point

$d(m1, p2) = 3 + 2 = 5$ $d(m2, p2) = 1 + 0 = 1$

$d(m1, p3) = 5 + 0 = 5$ $d(m2, p3) = 3 + 1 = 4$

$d(m1, p5) = 1 + 2 = 3$ $d(m2, p5) = 1 + 3 = 4$

Thus after first iteration

Cluster 1= {p1, p5}

Cluster 2={p2,p3,p4}

Total Cost= $\{d(m1, p5) + d(m2, p2) + d(m2, p3)\}$=10

Iteration 2

Lets swap m1 and p2

Now m1=p2= (4,5) and m2=p4=(3,4) are two medoid

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1)

Calculate distance between new cluster centers and each data points

$d(m1, p1) = 3 + 2 = 5$ $d(m2, p1) = 2 + 1 = 3$

$d(m1, p3) = 2 + 1 = 3$ $d(m2, p3) = 3 + 1 = 4$

$d(m1, p5) = 2 + 4 = 6$ $d(m2, p5) = 1 + 3 = 4$

Thus after second iteration

Cluster 1= {p2,p3}

Cluster 2={p1,p4,p5}

Total Cost= $\{d(m1, p3) + d(m2, p1) + d(m2, p5)\}$=3+3+4=10

Same as previous, thus no undo swap

Repeat this process till cost of configuration do not decrease

CLARA (Clustering Large Applications)

It draws multiple samples of data set, applies PAM on each sample, and gives best clustering as output. Thus, it can handle larger data sets than PAM. Efficiency and effectiveness depends on the sampling technique used.

Algorithm

- Set mincost to MAXIMUM;
- Repeat *q* times // draws *q* samples
- Create *S* by drawing *s* objects randomly from *D*;
- Generate the set of medoids *M* from *S* by applying the PAM algorithm;
- Compute cost(M,D)
- If cost(*M*, *D*)<mincost
    - Mincost = cost(*M*, *D*);
    - Bestset = *M*;
- Endif;
- Endrepeat;
- Return Bestset;

CLARANS ("Randomized" CLARA)

CLARANS is a Clustering Algorithm based on Randomized Search. CLARANS draws sample in solution space dynamically. A solution is a set of *k* medoids. The solution space can be represented by a graph where every node is a potential solution, i.e., a set of *k* medoids

Hierarchical Clustering

In data mining and statistics, **hierarchical clustering** (also called **hierarchical cluster analysis** or **HCA**) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Distance matrix is used for deciding which clusters to merge/split. There are losts of alternatives to define distance between two sets of points.

Single-Link Distance

Single-Link Distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***minimum distance*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$**.** The distance is **defined by the two most similar objects**

$$D_{sl}(C_i, C_j) = \min_{x,y}\left\{d(x,y) \middle| x \in C_i, y \in C_j\right\}$$

Complete-Link Distance

Complete-Link Distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***maximum distance*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$**.** The distance is **defined by the two most dissimilar objects**.

$$D_{cl}(C_i, C_j) = \max_{x,y}\left\{d(x,y) \middle| x \in C_i, y \in C_j\right\}$$

Group Average Distance

Group Average Distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the ***average distance*** between any object in $\mathbf{C_i}$ and any object in $\mathbf{C_j}$**.**

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x,y)$$

Centroid Distance

Centroid Distance between clusters $\mathbf{C_i}$ and $\mathbf{C_j}$ is the distance between the centroid $\mathbf{r_i}$ of $\mathbf{C_i}$ and the centroid $\mathbf{r_j}$ of $\mathbf{C_j}$.

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

Agglomerative Clustering Algorithm

Algorithm

1. Compute the distance matrix between the input data points
2. Let each data point be a cluster
3. **Repeat**
   - Merge the two closest clusters
   - Update the distance matrix
4. **Until** only K clusters remains

Example

Cluster the data points (1,1), (1.5,1.5), (5,5), (3,4), (4,4), (3, 3.5) into two clusters.

Solution

Assume A=(1,1), B= (1.5,1.5), C=(5,5), D=(3,4), E=(4,4), F=(3,3.5)

Distance Matrix

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

In this case, the closest cluster is between cluster F and D with shortest distance of 0.5. Thus, we group cluster D and F into cluster (D, F). Then we update the distance matrix (see distance matrix below). Distance between ungrouped clusters will not change from the original distance matrix. Now the problem is how to calculate distance between newly grouped clusters (D, F) and other clusters?

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

Looking at the lower triangular updated distance matrix, we found out that the closest distance between cluster B and cluster A is now 0.71. Thus, we group cluster A and cluster B into a single cluster name (A, B). Now we update the distance matrix. Aside from the first row and first column, all the other elements of the new distance matrix are not changed.

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

Observing the lower triangular of the updated distance matrix, we can see that the closest distance between clusters happens between cluster E and (D, F) at distance 1.00. Thus, we cluster them together into cluster ((D, F), E). The updated distance matrix is given below

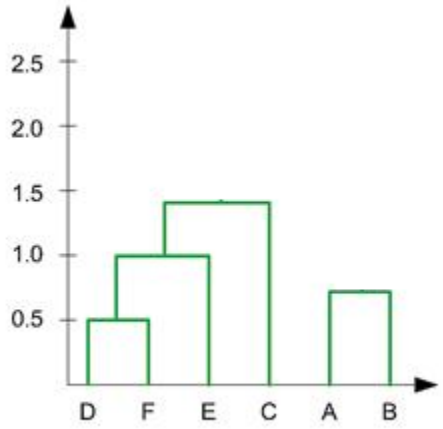
Min Distance (Single Linkage)

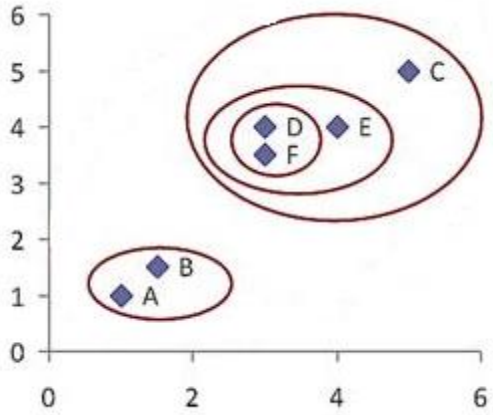
Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	2.50	1.41	0.00

After that, we merge cluster ((D, F), E) and cluster C into a new cluster name (((D, F), E), C). The updated distance matrix is shown in the figure below

Min Distance (Single Linkage)

Dist	(A,B)	(D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	2.50	0.00



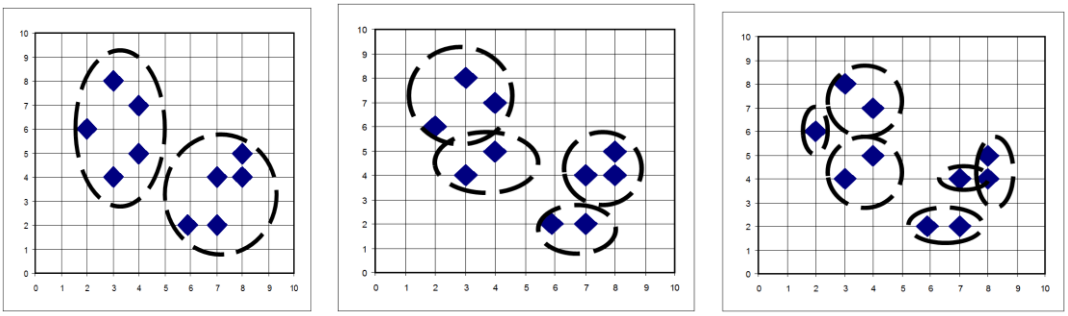


Divisive clustering

A cluster hierarchy can also be generated top-down. This variant of hierarchical clustering is called *top-down clustering* or *divisive clustering*. We start at the top with all data in one cluster. The cluster is split two clusters such that the objects in one subgroup are far from the objects in the other. . This procedure is applied recursively until required numbers of clusters are formed. This method is not considered attractive because there exist $O(2^n)$ways of splitting each cluster.

Algorithm

1. Start with all data point in single cluster
2. **Repeat**
   - Choice of the cluster to be split
   - Split of this cluster
3. **Until** only K clusters are created



Dissimilarity Measures of Binary Attributes

q- number of attributes having value 1 in both object

r- number of attributes having value 1 in object I and 0 in object J

s- number of attributes having value 0 in object I and 1 in object I

d(i,j)=r+s/q+r+s for asymmetric attributes

d(i,j)=r+s/q+r+s+t for symmetric attributes

sim(i,j)=1-d(i,j) , which is also called jacquard coefficient

Data Transformation

Data transformation is the process of converting data or information from one format to another, usually from the format of a source system into the required format of a new destination system. The usual process involves converting documents, but data conversions sometimes involve the conversion of a program from one computer language to another to enable the program to run on a different platform. The usual reason for this data migration is the adoption of a new system that's totally different from the previous one.

Techopedia explains *Data Transformation*

In real practice, data transformation involves the use of a special program that's able to read the data's original base language, determine the language into which the data that must be translated for it to be usable by the new program or system, and then proceeds to transform that data.

Data Transformation involves two key phases:

1. **Data Mapping**: The assignment of elements from the source base or system toward the destination to capture all transformations that occur. This is made more complicated when there are complex transformations like many-to-one or one-to-many rules for transformation.
2. **Code Generation**: The creation of the actual transformation program. The resulting data map specification is used to create an executable program to run on computer systems.

Data Normalization

Normalization is normally done, when there is a distance computation involved in our algorithm. Some of the **techniques** of normalization are:

Min-Max Normalization –

Min Max Normalization transforms a value A to B which fits in the range[C,D]. It is given my the below formula

$$B = \left(\frac{(A - minimum\,value\,of\ A)}{(maximum\,value\,of\ A - minimum\,value\,of\ A)}\right) * (D - C) + C$$

Consider the below example, the salary value is 50000, we want to transform this in to the range [0.0, 1.0], the maximum value of salary is 55000 and the minimum value of salary is 25000 so the new scaled value for 50000 will be.

$$= \left(\frac{(50000 - 25000)}{(55000 - 25000)}\right) * (1 - 0) + 0$$

$$= 0.83333333333333337$$

Unit 8

Classification and Prediction

There are two forms of data analysis that can be used for extracting models describing important classes or to predict future data trends. These two forms are as follows:

→ Classification

→ Prediction

Classification models predict categorical class labels; and prediction models predict continuous valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.

What is classification?

Following are the examples of cases where the data analysis task is Classification:

- A bank loan officer wants to analyze the data in order to know which customer (loan applicant) is risky or which are safe.
- A marketing manager at a company needs to analyze a customer with a given profile, who will buy a new computer.

In both of the above examples, a model or classifier is constructed to predict the categorical labels. These labels are risky or safe for loan application data and yes or no for marketing data.

What is prediction?

Following are the examples of cases where the data analysis task is Prediction: Suppose the marketing manager needs to predict how much a given customer will spend during a sale at his company. In this example we are bothered to predict a numeric value. Therefore the data analysis task is an example of numeric prediction. In this case, a model or a predictor will be constructed that predicts a continuous-valued-function or ordered value. Regression analysis is a statistical methodology that is most often used for numeric prediction.

How Does Classification Works?

With the help of the bank loan application that we have discussed above, let us understand the working of classification. The Data Classification process includes two steps:

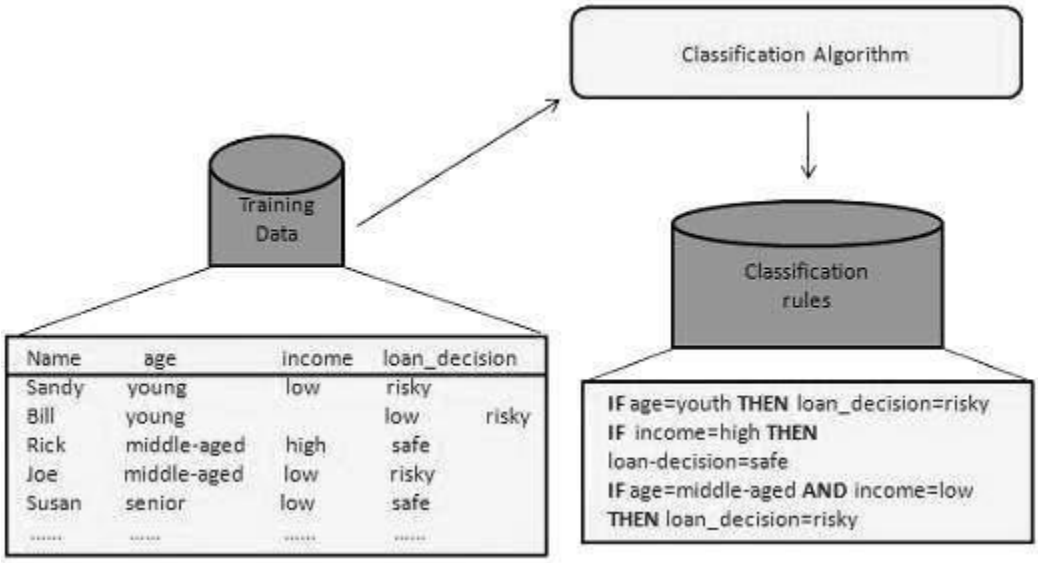
→ Building the Classifier or Model

→ Using Classifier for Classification

Building the Classifier or Model

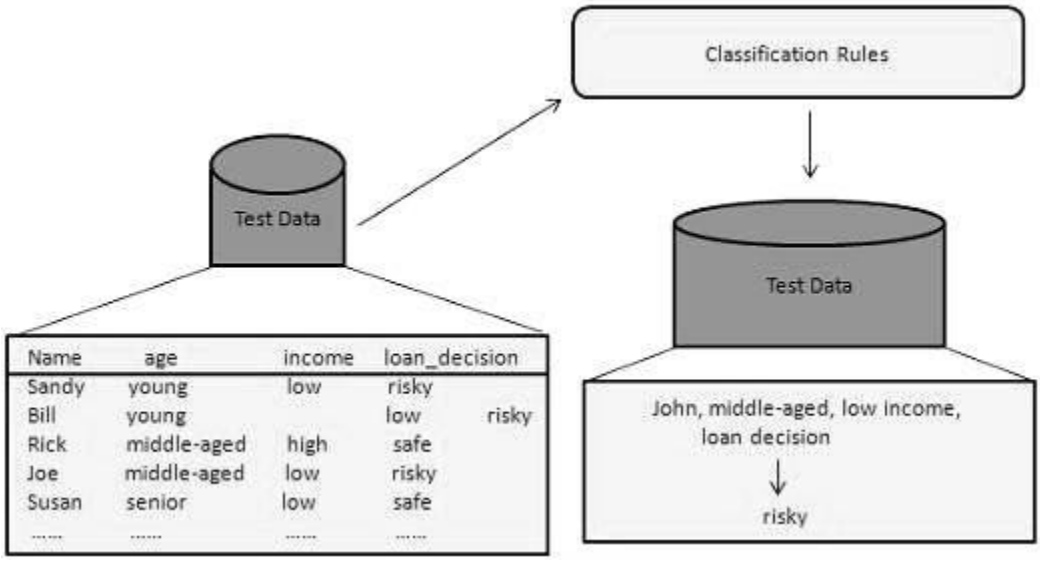
- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.

- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.



Using Classifier for Classification

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.



Classification and Prediction Issues

The major issue is preparing the data for Classification and Prediction. Preparing the data involves the following activities:

- **Data Cleaning** − Data cleaning involves removing the *noise and treatment of missing values*. The noise is removed by applying smoothing techniques and the problem of missing values is solved by replacing a missing value with most commonly occurring value for that attribute.
- **Relevance Analysis** − Database may also have the irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
- **Data Transformation and reduction** − The data can be transformed by any of the following methods:
    - **Normalization** − Data is transformed using normalization. Normalization involves scaling all values for given attribute in order to make them fall within a small specified range. Normalization is used when in the learning step, the neural networks or the methods involving measurements are used.
    - **Generalization** − Data can also be transformed by generalizing it to the higher concept. For this purpose we can use the concept hierarchies.

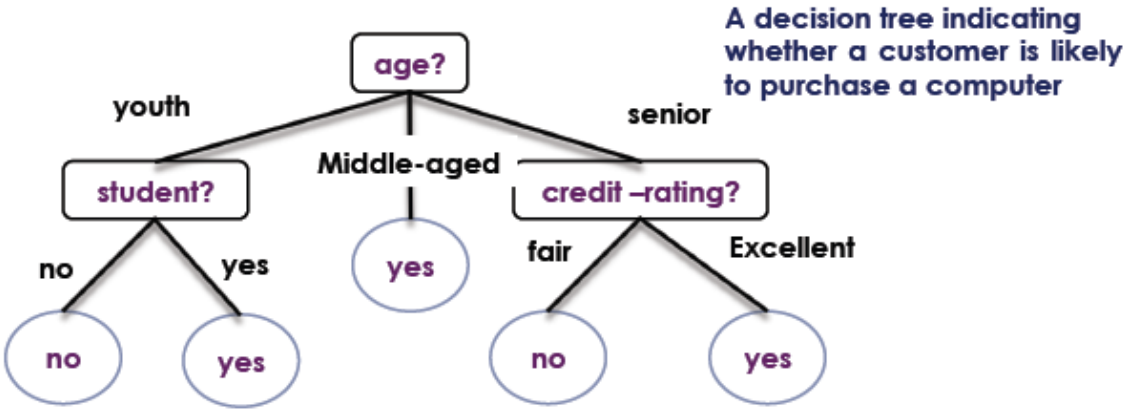
Comparison of Classification and Prediction Methods

Here are the criteria for comparing the methods of Classification and Prediction:

- **Accuracy** − Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.
- **Speed** − This refers to the computational cost in generating and using the classifier or predictor.
- **Robustness** − It refers to the ability of classifier or predictor to make correct predictions from given noisy data.
- **Scalability** − Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.
- **Interpretability** − It refers to what extent the classifier or predictor understands.

Classification by Decision Tree Induction

Decision tree induction is the learning of decision trees from class labeled training tuples. It is a decision tree is a flowchart-like tree structure where **internal nodes** (non leaf node) denotes a test on an attribute b**ranches** represent outcomes of tests **Leaf nodes** (terminal nodes) hold class labels and **Root node** is the topmost node.



Class-label Yes: The customer is likely to buy a computer
Class-label no: The customer is unlikely to buy a computer

The attributes of a tuple are tested against the decision tree. A path is traced from the root to a leaf node which holds the prediction for that tuple.

Example

RID	age	income	student	credit-rating	Class
1	*youth*	*high*	*no*	*fair*	*?*

Test on age: youth
Test of student: no
Reach leaf node
Class NO: *the customer Is Unlikely to buy a computer*

Why decision trees classifiers are so popular?

- Why decision trees classifiers are so popular?
- The construction of a decision tree does not require any domain

- knowledge or parameter setting
- They can handle high dimensional data
- Intuitive representation that is easily understood by humans
- Learning and classification are simple and fast
- They have a good accuracy

Algorithm for constructing Decision Tress

Constructing a Decision tree uses **greedy algorithm.** Tree is constructed in a top-down recursive divide-and-conquer manner.

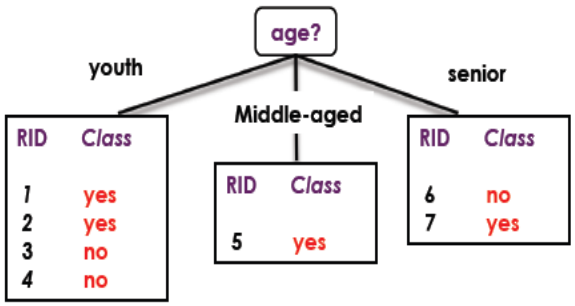
1. At start, all the training tuples are at the root
2. Tuples are partitioned recursively based on selected attributes
3. If all samples for a given node belong to the same class
   - Label the class
4. If There are no remaining attributes for further partitioning
   - **Majority voting** is employed for classifying the leaf
5. There are no samples left
   - Label the class and terminate
6. Else
   - Got to step 2

Example

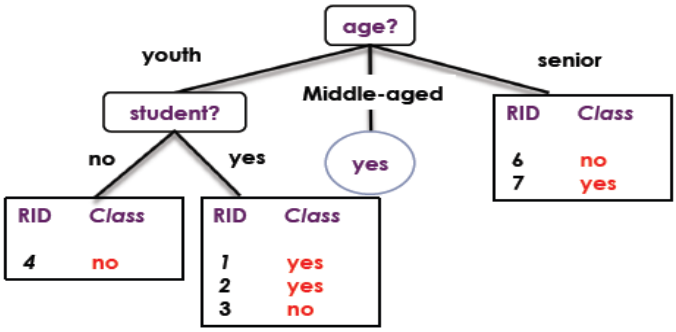
RID	age	student	credit-rating	Class: Buys-computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Solution

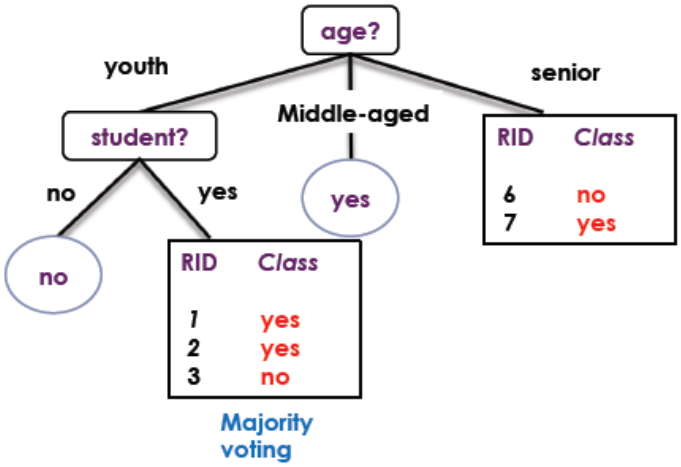
Step 1



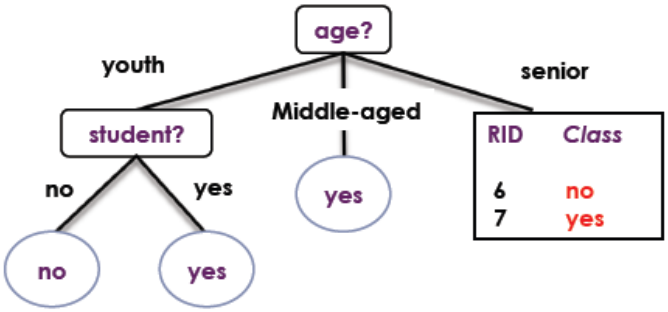
Step 2



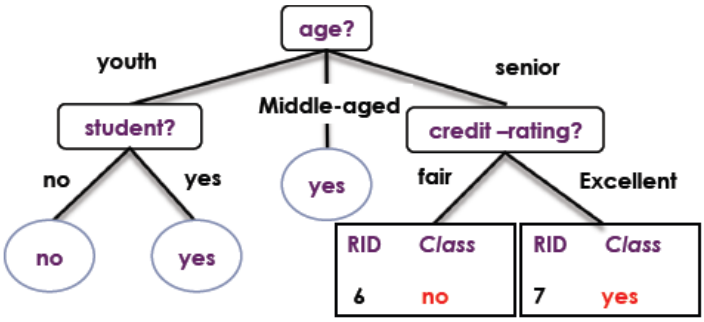
Step 3



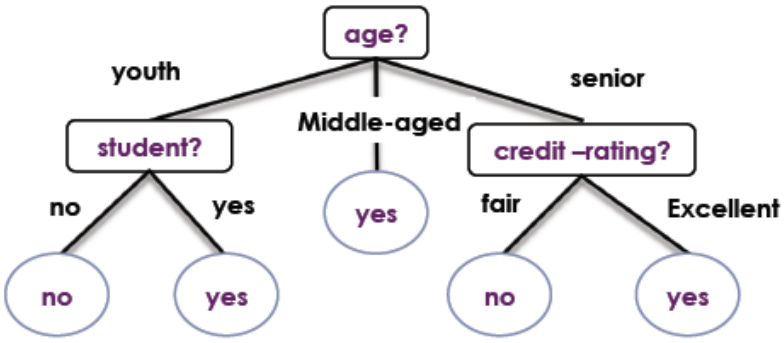
Step 4



Step 5



Step 6



Prediction

As already mentioned, numeric prediction is the task of predicting continuous values for given input. For example, we may wish to predict the salary of college graduates with 10 years of work experience, or the potential sales of a new product given its price. The most widely used approach for numeric prediction is regression. Regression analysis can be used to model the relationship between one or more *independent* or predictor variables and a *dependent* or response variable. In general, the values of the predictor variables are known. The response variable is what we want to predict.

Linear Regression

Straight-line regression analysis involves a response variable, y, and a single predictor variable, x. It is the simplest form of regression, and models y as a linear function of x. That is,

$$y = b+wx$$

Where, the b and w are regression coefficients specifying the y-intercept and slope of the line respectively. These coefficients, w and b, can also be thought of as weights, so that we can equivalently write,

$$y = w0+w1x$$

Let D be a training set consisting of values of predictor variable, x, for some population and their associated values for response variable, y. The training set contains $|D|$ data points of the form$(x1, y1)$, $(x2, y2)$, (xj, yj). The regression coefficients can be estimated using this method with the following equations:

$$w1 = \frac{\sum_{i=1}^{|D|}(xi - \bar{x})(yi - \bar{y})}{\sum (xi - \bar{x})^2} \qquad w0 = \bar{y} - w1\bar{x}$$

Example

Table given below shows a set of paired data where x is the number of years of work experience of a college graduate and y is the corresponding salary of the graduate. Predict the value of salary after 10 years of experience.

Years of Experience (x)	Salary (y)
1	20000
3	40000
6	58000
8	66000
9	70000
12	82000
13	85000
15	92000
17	100000
20	108000

Solution

$$\bar{x} = 10.4 \qquad \bar{y} = 72100$$

$$w1 = \frac{\left\{\begin{array}{l}(1-10.4)*(20000-72100)+(3-10.4)*(40000-72100)+\\(6-10.4)*(580000-72100)+(8-10.4)*(70000-72100)+\\(9-10.4)*(66000-72100)+(12-10.4)*(82000-72100)+\\(13-10.4)*(85000-72100)+(15-10.4)*(92000-72100)+\\(17-10.4)*(100000-72100)+(20-10.4)*(108000-72100)\end{array}\right\}}{\begin{array}{l}(1-10.4)^2+(3-10.4)^2+(6-10.4)^2+(8-10.4)^2+(9-10.4)^2+\\(12-10.4)^2+(13-10.4)^2+(15-10.4)^2+(17-10.4)^2+(20-10.4)^2\end{array}} = ??????(calculate \quad value)$$

$$w0 = \bar{y} - w1\bar{x} = ?????(calculate \quad value)$$

Now, predict the value by using the equation

$y = w0 + w1x = ????????$ *(Calculate Value) use* $x=10$

Introduction to Clustering

The process of grouping a set of physical or abstract objects into classes of *similar* objects is called clustering. A cluster is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters. A cluster of data objects can be treated collectively as one group. Although classification is an effective means for distinguishing groups or classes of objects, it requires the often costly collection and labeling of a large set of training tuples or patterns, which the classifier uses to model each group.

Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their *similarity*. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and frequent purchases, may be of interest as possible fraudulent activity.

Measure of Distance between Data Points

During clustering of data objects, we need to find distance between data objects. The most popular distance measure is *Euclidean distance*, which is defined as:

$$d(x,y) = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2}$$

Where, $x = (x_1, y_1)$ *and* $y = (x_2, y_2)$

Another well-known metric is *Manhattan (or city block) distance*, defined as

$$d(x,y) = |x_2 - x_1| + |y_2 - y_1|$$

Minkowski distance is a generalization of both Euclidean distance and Manhattan distance. It is defined as

$$d(x,y) = \left(|x_2 - x_1|^p + |y_2 - y_1|^p\right)^{1/p}$$

Where p is a positive integer, such a distance is also called L_p norm, in some literature. It represents the Manhattan distance when $p = 1$ (i.e., L_1 norm) and Euclidean distance when $p = 2$ (i.e., L_2 norm).

Categorization of Clustering Algorithms

Many clustering algorithms exist in the literature. In general, the major clustering methods can be classified into the following categories.

→ **Partitioning methods**: Given a database of *n* objects or data tuples, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster and $k < n$. Given *k*, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another.

→ **Hierarchical methods:** A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*. The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds. The *divisive approach*, also called the *top-down* approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

→ **Density-based methods:** Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. Other clustering methods have been developed based on the notion of *density*. Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold.

→ **Model-based methods:** Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model. EM is an algorithm that performs expectation-maximization analysis based on statistical modeling.

K-means Clustering Algorithm

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed

in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.

Algorithmic steps for k-means clustering

Let X = {x_1,x_2,x_3,……..,x_n} be the set of data points and V = {v_1,v_2,…….,v_c} be the set of centers.

1) Randomly select '*c*' cluster centers.
2) Calculate the distance between each data point and cluster centers.
3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..
4) Recalculate the new cluster center using: $1/c_i \sum_{i=1}^{c_i} x_i$, where, '$c_i$' represents the number of data points in i^{th} cluster.
5) Recalculate the distance between each data point and new obtained cluster centers.
6) If no data point was reassigned then stop, otherwise repeat from step 3.

Advantages

→ Fast, robust and easier to understand.
→ Gives best result when data set are distinct or well separated from each other.

Disadvantages

→ The learning algorithm requires apriori specification of the number of cluster centers.
→ Randomly choosing of the cluster center cannot lead us to the fruitful result.
→ Applicable only when mean is defined i.e. fails for categorical data.
→ Algorithm fails for non-linear data set.

Example

Divide the data points {(1,1), ((2,1), (4,3), (5,4)} into two clusters.

Solution

Let p1=(1,1) p2=(2,1) p3=(4,3) p4=(5,4)

Initial step

Let c1=(1,1) and c2=(2,1) are two initial cluster centers.

Iteration 1

Calculate distance between clusters centers and each data points

$$d(c1, p1) = 0 \qquad d(c2, p1) = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1 \qquad d(c2, p2) = \sqrt{(2-2)^2 + (1-1)^2} = 0$$

$$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6 \qquad d(c2, p3) = \sqrt{(4-2)^2 + (3-1)^2} = 2.83$$

$$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5 \qquad d(c2, p4) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Thus after first iteration

Cluster 1= {p1}

Cluster 2={p2,p3,p4}

Now, new cluster centers are:

c1=(1,1) and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

Iteration 2

Calculate distance between new cluster centers and each data points

$$d(c1, p1) = 0 \qquad d(c2, p1) = \sqrt{(11/3-1)^2 + (8/3-1)^2} = 3.14$$

$$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1 \qquad d(c2, p2) = \sqrt{(11/3-2)^2 + (8/3-1)^2} = 2.35$$

$$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6 \qquad d(c2, p3) = \sqrt{(11/3-4)^2 + (8/3-3)^2} = 0.46$$

$$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5 \qquad d(c2, p4) = \sqrt{(11/3-5)^2 + (8/3-4)^2} = 1.88$$

Thus after second iteration

Cluster 1= {p1,p2}

Cluster 2={p3,p4}

Now, new cluster centers are:

c1={(1+2)/2,(1+1)/2}={3/2,1} and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

Repeat this process till no re-assignment of points to groups.

K-medoid Clustering Algorithm

A *medoid* can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. i.e. it is a most centrally located point in the cluster. In contrast of K-means algorithm, K-medoid algorithm chooses data point as centers and works with arbitrary matrix of distances instead of l_2.*K-medoid* is a classical partitioning

technique of clustering that clusters the data set of n objects into k clusters known *a priori*. It is more robust to noise and outliers as compared to because it may minimize a sum of pair-wise dissimilarities instead of a sum of squared Euclidean distances.

Algorithms

The most common realization of k-medoid clustering is the **Partitioning around Medoid (PAM)** algorithm and is as follows:

1. Initialize: randomly select (without replacement) k of the n data points as the medoid
2. Associate each data point to the closest medoid.
3. While the cost of the configuration decreases:
   - For each medoid m, for each non-medoid data point o:
     - Swap m and o, re-compute the cost (sum of distances of points to their medoid)
     - If the total cost of the configuration increased in the previous step, undo the swap

Example

Cluster the following data set of ten objects into two clusters i.e. $k = 2$. Data Points are {(1,3), (4,2), (6,2), (3,5), (4,1)}

Solution

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1)

Initial step

Let m1=p1=(1,3) and m2=p4=(3,4) are two initial medoid

Iteration 1

Calculate distance between clusters centers and each data point

$d(m1, p2) = 3 + 2 = 5$ $d(m2, p2) = 1 + 0 = 1$

$d(m1, p3) = 5 + 0 = 5$ $d(m2, p3) = 3 + 1 = 4$

$d(m1, p5) = 1 + 2 = 3$ $d(m2, p5) = 1 + 3 = 4$

Thus after first iteration

Cluster 1= {p1, p5}

Cluster 2={p2,p3,p4}

Total Cost= $\{d(m1, p5) + d(m2, p2) + d(m2, p3)\}$=10

Iteration 2

Lets swap m1 and p2

Now m1=p2= (4,5) and m2=p4=(3,4) are two medoid

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1)

Calculate distance between new cluster centers and each data points

$d(m1, p1) = 3 + 2 = 5$ $d(m2, p1) = 2 + 1 = 3$

$d(m1, p3) = 2 + 1 = 3$ $d(m2, p3) = 3 + 1 = 4$

$d(m1, p5) = 2 + 4 = 6$ $d(m2, p5) = 1 + 3 = 4$

Thus after second iteration

Cluster 1= {p2,p3}

Cluster 2={p1,p4,p5}

Total Cost= $\{d(m1, p3) + d(m2, p1) + d(m2, p5)\}$=3+3+4=10

Same as previous, thus no undo swap

Repeat this process till cost of configuration do not decrease

Unit 9

Mining Text Databases

Text mining is an interdisciplinary field that draws on information retrieval, data-mining, machine learning, statistics and computational linguistics. Text mining has been very active. An important goal is to derive high-quality information from the text. Text databases consist of huge collection of documents. They collect these information from several sources such as news articles, books, digital libraries, e-mail messages, web pages, etc. Due to increase in the amount of information, the text databases are growing rapidly. In many of the text databases, the data is semi-structured. For example, a document may contain a few structured fields, such as title, author, publishing-date, etc. But along with the structure data, the document also contains unstructured text components, such as abstract and contents. Without knowing what could be in the documents, it is difficult to formulate effective queries for analyzing and extracting useful information from the data. Users require tools to compare the documents and rank their importance and relevance. Therefore, text mining has become popular and an essential theme in data mining.

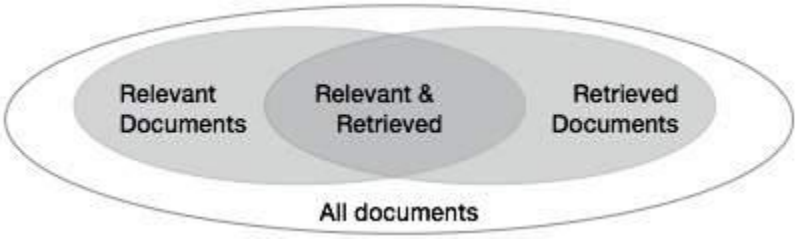
Information Retrieval

Information retrieval deals with the retrieval of information from a large number of text-based documents. Examples of information retrieval system include − Online Library catalogue system, Online Document Management Systems, Web Search Systems etc.

The main problem in an information retrieval system is to locate relevant documents in a document collection based on a user's query. This kind of user's query consists of some keywords describing an information need. In such search problems, the user takes an initiative to pull relevant information out from a collection. This is appropriate when the user has ad-hoc information need, i.e., a short-term need. But if the user has a long-term information need, then the retrieval system can also take an initiative to push any newly arrived information item to the user. This kind of access to information is called Information Filtering. And the corresponding systems are known as Filtering Systems or Recommender Systems.

Basic Measures for Text Retrieval

We need to check the accuracy of a system when it retrieves a number of documents on the basis of user's input. Let the set of documents relevant to a query be denoted as {Relevant} and the set of retrieved document as {Retrieved}. The set of documents that are relevant and retrieved can be denoted as {Relevant} ∩ {Retrieved}. This can be shown in the form of a Venn diagram as follows



There are three fundamental measures for assessing the quality of text retrieval

- Precision
- Recall
- F-score

Precision

Precision is the percentage of retrieved documents that are in fact relevant to the query. Precision can be defined as

Precision= |{Relevant} ∩ {Retrieved}| / |{Retrieved}|

Recall

Recall is the percentage of documents that are relevant to the query and were in fact retrieved. Recall is defined as

Recall = |{Relevant} ∩ {Retrieved}| / |{Relevant}|

F-score

F-score is the commonly used trade-off. The information retrieval system often needs to trade-off for precision or vice versa. F-score is defined as harmonic mean of recall or precision as follows

F-score = recall x precision / (recall + precision) / 2

Mining World Wide Web

The World Wide Web contains huge amounts of information that provides a rich source for data mining. The web poses great challenges for resource and knowledge discovery based on the following observations

- **The web is too huge** − The size of the web is very huge and rapidly increasing. This seems that the web is too huge for data warehousing and data mining.

- **Complexity of Web pages** − The web pages do not have unifying structure. They are very complex as compared to traditional text document. There are huge amount of documents in digital library of web. These libraries are not arranged according to any particular sorted order.
- **Web is dynamic information source** − The information on the web is rapidly updated. The data such as news, stock markets, weather, sports, shopping, etc., are regularly updated.
- **Diversity of user communities** − The user community on the web is rapidly expanding. These users have different backgrounds, interests, and usage purposes. There are more than 100 million workstations that are connected to the Internet and still rapidly increasing.
- **Relevancy of Information** − It is considered that a particular person is generally interested in only small portion of the web, while the rest of the portion of the web contains the information that is not relevant to the user and may swamp desired results.

Mining Web page layout structure

The basic structure of the web page is based on the Document Object Model (DOM). The DOM structure refers to a tree like structure where the HTML tag in the page corresponds to a node in the DOM tree. We can segment the web page by using predefined tags in HTML. Thus the DOM structure can be used to facilitate information extraction. The HTML syntax is flexible therefore, the web pages does not follow the W3C specifications. Not following the specifications of W3C may cause error in DOM tree structure.

The DOM structure was initially introduced for presentation in the browser and not for description of semantic structure of the web page. The DOM structure cannot correctly identify the semantic relationship between the different parts of a web page.

Multimedia Data Mining

A multimedia database system stores and manages a large collection of *multimedia data*, such as audio, video, image, graphics, speech, text, document, and hypertext data, which contain text, text markups, and linkages. Typical multimedia database systems include NASA's EOS (Earth Observation System), various kinds of image and audio-video databases, and Internet databases.

Normally study of multimedia data mining focuses on image data mining. For similarity searching in multimedia data, we consider two main families of multimedia indexing and retrieval systems:

- **Description-based retrieval systems**: which build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation.
- **Content-based retrieval systems:** It support retrieval based on the image content, such as color histogram, texture, pattern, image topology, and the shape of objects and their layouts and locations within the image.

Spatial Data Mining

A spatial database stores a large amount of space-related data, such as maps, preprocessed remote sensing or medical imaging data, and VLSI chip layout data. Spatial databases have many features distinguishing them from relational databases. They carry topological and/or distance information. Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial databases. Such mining demands an integration of data mining with spatial database technologies. It can be used for understanding spatial data, discovering spatial relationships and relationships between spatial and non-spatial data. It is expected to have wide applications in geographic information systems, geomarketing, remote sensing, image database exploration, medical imaging, navigation, traffic control, environmental studies, and many other areas where spatial data are used. A crucial challenge to spatial data mining is the exploration of *efficient* spatial data mining techniques due to the huge amount of spatial data and the complexity of spatial data types and spatial access methods.