Unit – I

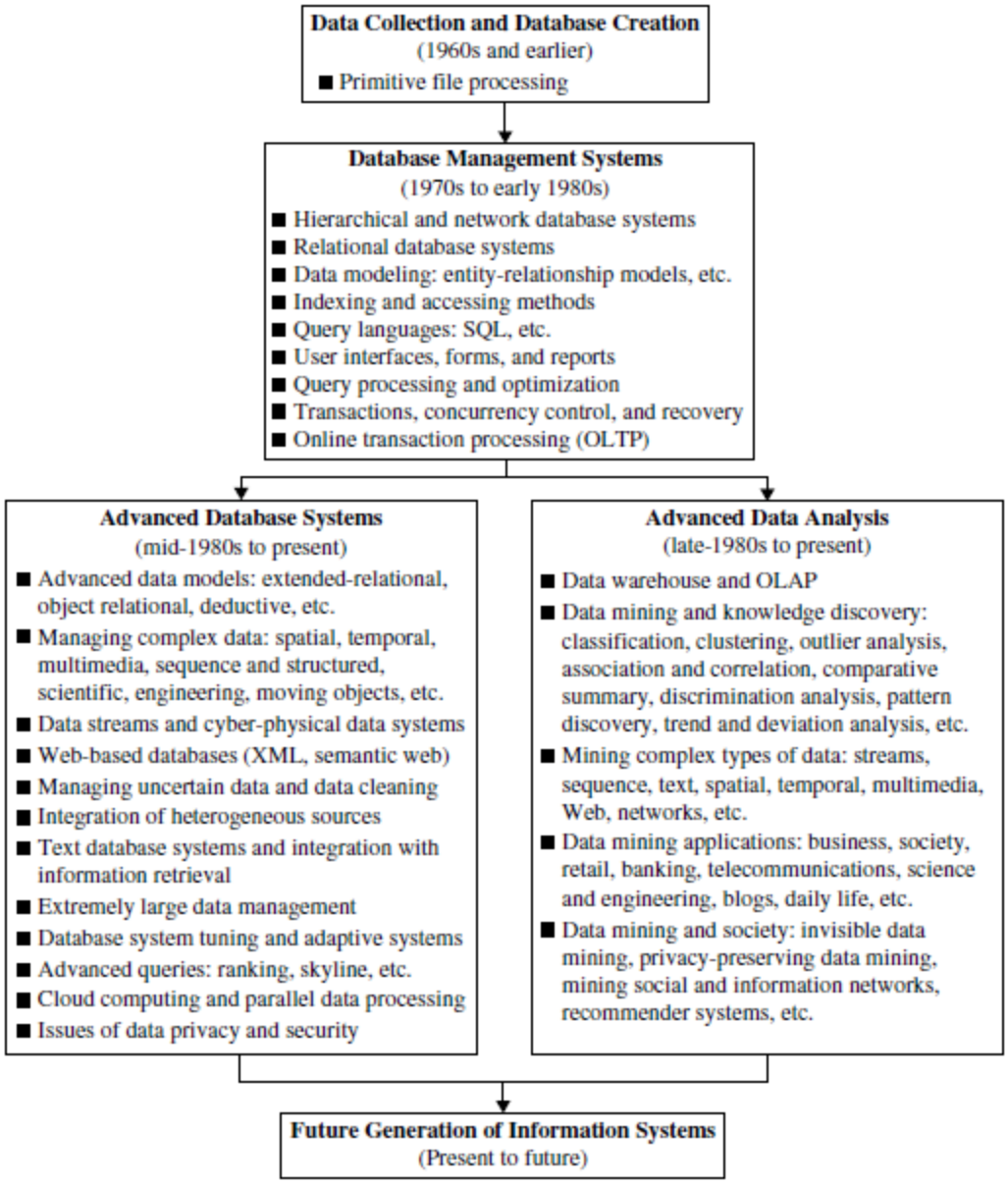
Introduction

- Why Data Mining?
- What is Data Mining?
- What Kinds of Data Can Be Mined?
- What Kinds of Patterns Can Be Mined?
- Which Technologies Are Used?
- Which Kinds of Applications Are Targeted?
- Major Issues in Data Mining
- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- Measuring Data Similarity and Dissimilarity

WHY DATA MINING?

Necessity, who is the mother of invention. – Plato

- ✓ Data mining turns a large collection of data into knowledge
- ✓ The major reason that Data Mining has attracted a great deal of attention in the information industry in recent years is *due to thee wide availability of huge amount of data and the imminent need for turning such data into useful information and knowledge.*
- ✓ The information and knowledge gained can be used for applications ranging from business management, production control and market analysis.
- ✓ The abundance of data, coupled with the need for powerful data analysis tools, has been described as a *data rich but information poor* situation. The fast-growing, tremendous amount of data, collected and stored in large and numerous data repositories, has far exceeded our human ability for comprehension without powerful tools. As a result, data collected in large data repositories become "data tombs"—data archives that are seldom visited. Consequently, important decisions are often made based not on the information-rich data stored in data repositories but rather on a decision maker's intuition, simply because the decision maker does not have the tools to extract the valuable knowledge embedded in the vast amounts of data. Efforts have been made to develop expert system and knowledge-based technologies, which typically rely on users or domain experts to *manually* input knowledge into knowledge bases. Unfortunately, however, the manual knowledge input procedure is prone to biases and errors and is extremely costly and time consuming. The widening gap between data and information calls for the systematic development of *data mining tools* that can turn data tombs into "golden nuggets" of knowledge.



Evolution of Database Technology

WHAT IS DATA MINING?

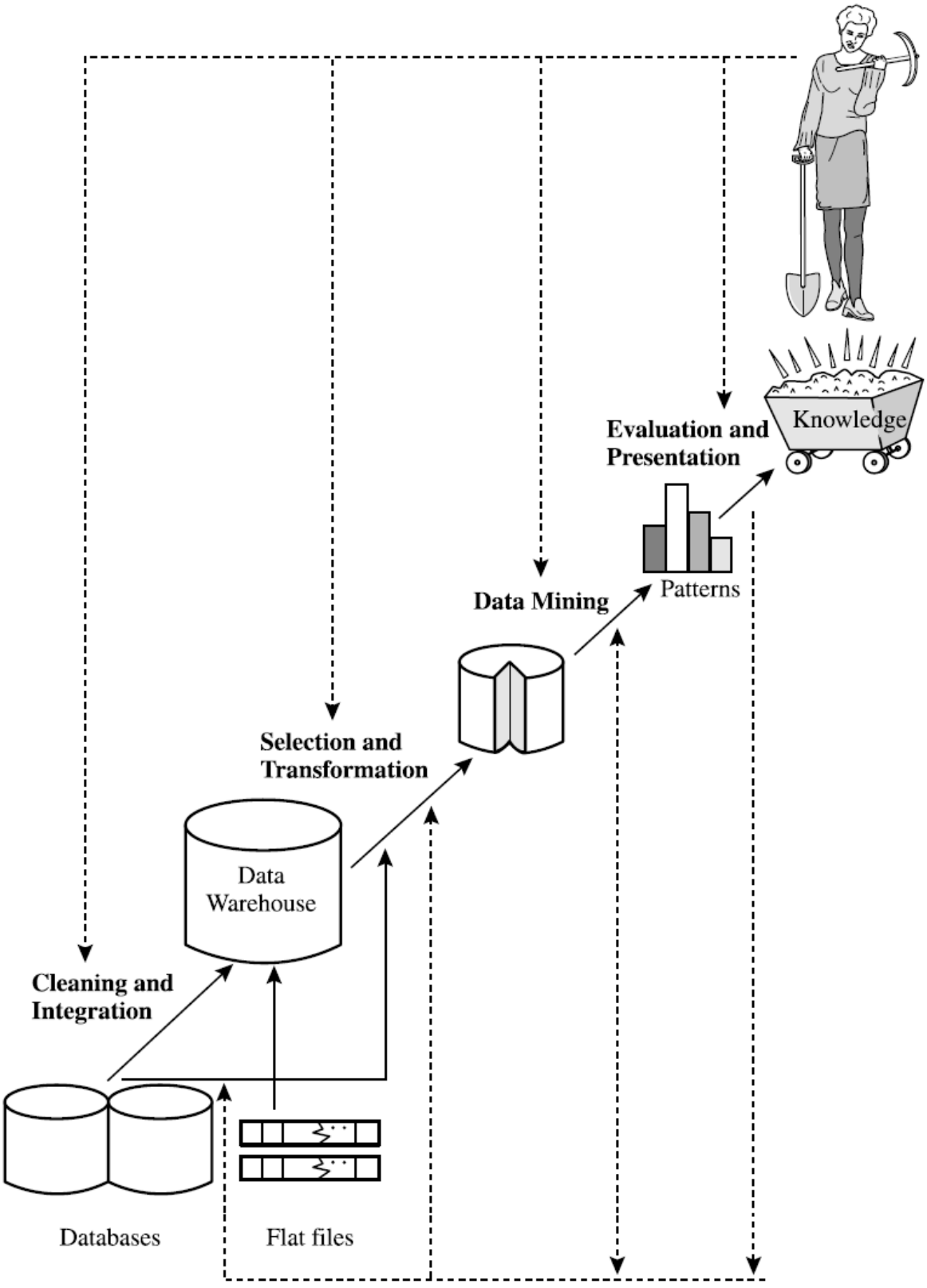
Data Mining refers to extracting or mining knowledge from large amounts of data.

Synonyms to Data Mining:

Knowledge mining from databases, knowledge extraction, data/ pattern analysis, data archeology and data dredging

Popular synonym is *"Knowledge Discovery in Databases (KDD)".*

Knowledge Discovery as a Process



1. **Data Cleaning:** remove noise and inconsistent data
2. **Data Integration:** multiple data sources to be combined
3. **Data Selection:** data relevant to analysis task are retrieved from the database
4. **Data Transformation:** data are transformed or consolidated into apt forms for mining. This is done by doing summary or aggregation operations

5. **Data Mining:** mining methods are applied for extracting patterns
6. **Pattern Evaluation:** identifies truly interesting patterns based on some interesting measures.
7. **Knowledge Representation:** visualization and knowledge representation techniques are used to present the discovered knowledge

WHAT KINDS OF DATA CAN BE MINED?

1. Relational Databases
2. Data Warehouses – Data Cube
3. Transactional Databases – Transactional Data Set
4. Advanced Database Systems and Advanced Database Applications
    a. Object Oriented Databases
    b. Object Relational Databases
    c. Spatial Databases
    d. Temporal and Time Series Databases
    e. Text Databases & Multimedia Databases
    f. Heterogeneous Databases and Legacy Databases
    g. World Wide Web

WHAT KINDS OF PATTERNS CAN BE MINED? (*Data Mining Functionalities*)

1. **Concept/ Class Description: Characterization and Discrimination:**

Data can be associated with classes or concepts. It is useful to describe individual classes or concepts. Such descriptions are called class/ concept descriptions. These are:

(a) *Data Characterization:* by summarizing the data of the class (target class).

(b) *Data Discrimination:* by comparison of target class with one or set of comparative classes.

The output of data characterization can be presented in various forms like pie-charts, bar charts, curves, multidimensional data cubes and tables.

Discrimination descriptions are expressed in rule forms are referred as discriminant rules.

2. **Association Analysis:**

It is the discovery of association rules showing attribute value conditions that occur frequently together in a given set. Association analysis is widely used for market basket or transaction analysis.

Rules are of the form, X=>Y

Uses *support* and *confidence* values

Ex: contains(T, “Computer”) => contains(T, “Software”)

[support=1%, confidence=50%]

3. **Classification and Prediction:**

Classification is the process of finding a set of models that describe and distinguish classes or concepts. Classification predicts a class of objects whose class label is unknown which is based on the analysis of training dataset (objects whose class label is known). This is represented by simple “IF-THEN” rules.

A decision tree is a flow chart like tree structure where each node denotes a test on the attribute value, each branch represents an outcome of the test and tree leaves represents classes. A neural network when used for classification is typically a collection of neuron like processing units with weighted connections between the units.

Classification can be used for predicting the class label of data objects. In some applications users wish to predict some missing or unavailable data values rather than class labels. This is usually the case when the predicted values are numerical data and is often specifically referred to as *Prediction.*

4. **Cluster Analysis:**

Clustering analyzes data objects without consulting a known class label. These objects are clustered based on the principle:

"maximizing the intra-class similarity and minimizing the inter-class similarity"

Each cluster can be viewed as a set of objects from which rules for that cluster can be derived.

This also facilitates grouping formation i.e. hierarchy of classes.

5. **Outlier Analysis:**

A database may contain some objects which do not fit into model of data. Such objects are called *outliers.* Most of the mining methods exclude outliers as noise or exceptions during mining. Outliers in some applications like fraud detection proved to be interesting. Analysis of outlier data is referred as outlier mining.

Outliers are identified by using statistical test like distribution or probability model or distance measures. Rather than these deviations based methods identify outliers by examining differences in the main characteristics of objects in a group.

Are All Patterns Interesting?

No, only small fractions of the patterns are interesting for analysis.

A pattern is interesting if,

1. It is easily understood
2. Valid on new or test data with some degree of certainty
3. Useful potentially
4. Novel

An interesting pattern represents knowledge.

WHICH TECHNOLOGIES ARE USED?

Data mining has incorporated many techniques from other domains such as statistics, machine learning, pattern recognition, database and data warehouse systems, information retrieval, visualization, algorithms, high performance computing, and many application domains. In this section, examples of several disciplines that strongly influence the development of data mining methods are included.

a. **Statistics**

Statistics studies the collection, analysis, interpretation or explanation, and presentation of data. Data mining has an inherent connection with statistics. A **statistical model** is a set of mathematical functions that describe the behavior of the objects in a target class in terms of random variables and their associated probability distributions. Statistical models are widely used to model data and data classes. For

example, in data mining tasks like data characterization and classification, statistical models of target classes can be built.

b. **Machine Learning**

Machine learning investigates how computers can learn (or improve their performance) based on data. A main research area is for computer programs to *automatically* learn to recognize complex patterns and make intelligent decisions based on data. For example, typical machine learning problem is to program a computer so that it can automatically recognize handwritten postal codes on mail after learning from a set of examples.

Supervised learning is basically a synonym for classification. The supervision in the learning comes from the labeled examples in the training data set.

Unsupervised learning is essentially a synonym for clustering. The learning process is unsupervised since the input examples are not class labeled. Typically, we may use clustering to discover classes within the data.

Semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled examples when learning a model. In one approach, labeled examples are used to learn class models and unlabeled examples are used to refine the boundaries between classes.

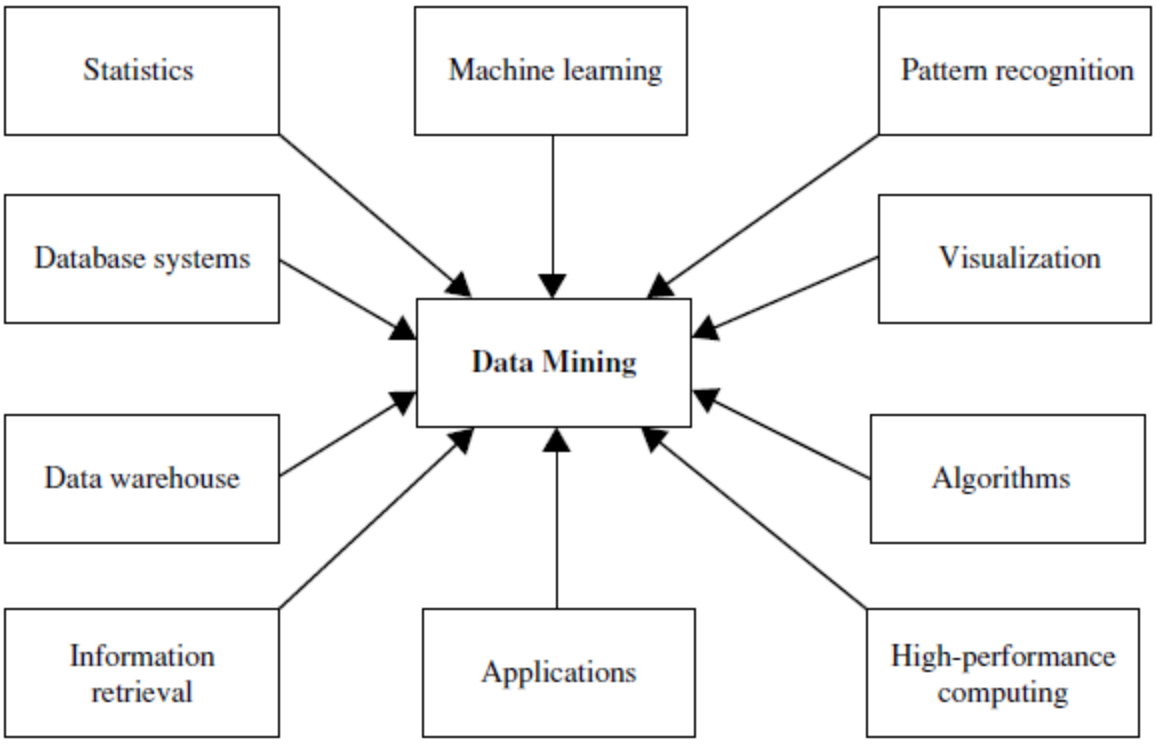
Active learning is a machine learning approach that lets users play an active role in the learning process. An active learning approach can ask a user (e.g., a domain expert) to label an example, which may be from a set of unlabeled examples or synthesized by the learning program.

c. **Database Systems and Data Warehouses**

A **data warehouse** integrates data originating from multiple sources and various time frames. It consolidates data in multidimensional space to form partially materialized data cubes.

d. **Information Retrieval**

Information retrieval (**IR**) is the science of searching for documents or information in documents. Documents can be text or multimedia, and may reside on the Web. The differences between traditional information retrieval and database systems are twofold: Information retrieval assumes that (1) the data under search are unstructured; and (2) the queries are formed mainly by keywords, which do not have complex structures



Data mining adopts techniques from many domains.

WHICH KINDS OF APPLICATIONS ARE TARGETED?

- Business Intelligence
- Web Search Engines

ISSUES IN DATA MINING

Major issues in data mining research, partitioning them into five groups: *mining methodology, user interaction, efficiency and scalability, diversity of data types*, and *data mining and society*.

a. **Mining Methodology**
   - Mining various and new kinds of knowledge
   - Mining knowledge in multidimensional space
   - Data mining—an interdisciplinary effort
   - Boosting the power of discovery in a networked environment
   - Handling uncertainty, noise, or incompleteness of data
   - Pattern evaluation and pattern- or constraint-guided mining

b. **User Interaction**
   - Interactive mining
   - Incorporation of background knowledge
   - Ad hoc data mining and data mining query languages
   - Presentation and visualization of data mining results

c. **Efficiency and Scalability**
   - Efficiency and scalability of data mining algorithms
   - Parallel, distributed, and incremental mining algorithms

d. **Diversity of Database Types**
   - Handling complex types of data
   - Mining dynamic, networked, and global data repositories

e. **Data Mining and Society**
   - Social impacts of data mining
   - Privacy-preserving data mining
   - Invisible data mining

DATA OBJECTS AND ATTRIBUTE TYPES

Data sets are made up of data objects. A **data object** represents an entity. Data objects can also be referred to as *samples, examples, instances, data points*, or *objects*.

What Is an Attribute?

An **attribute** is a data field, representing a characteristic or feature of a data object. The nouns *attribute*, *dimension*, *feature*, and *variable* are often used interchangeably in the literature. The term *dimension* is commonly used in data warehousing. Machine learning literature tends to use the term *feature*, while statisticians prefer the term *variable*. Data mining and database professionals commonly use the term *attribute.* The distribution of data involving one attribute (or variable) is called *univariate*. A *bivariate* distribution involves two attributes, and so on.

Nominal Attributes

The values of a **nominal attribute** are symbols or *names of things*. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as **categorical**. The values do not have any meaningful order; Also known as *enumerations*.

Ex: Suppose that *hair_color* and *marital_status* are two attributes describing *person* objects. In our application, possible values for *hair_color* are *black*, *brown*, *blond*, *red*, *auburn*, *gray*, and *white*. The attribute *marital_status* can take on the values *single, married, divorced*, and *widowed*. Both *hair color* and *marital status* are nominal attributes.

Binary Attributes

A **binary attribute** is a nominal attribute with only two categories or states: 0 or 1, where 0 typically means that the attribute is absent, and 1 means that it is present. Binary attributes are referred to as **Boolean** if the two states correspond to *true* and *false*.

Ex: the attribute *smoker* describing a *patient* object, 1 indicates that the patient smokes, while 0 indicates that the patient does not. The attribute *medical test* is binary, where a value of 1 means the result of the test for the patient is positive, while 0 means the result is negative.

A binary attribute is **symmetric** if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute *gender* having the states *male* and *female*.

A binary attribute is **asymmetric** if the outcomes of the states are not equally important, such as the *positive* and *negative* outcomes of a medical test.

Ordinal Attributes

An **ordinal attribute** is an attribute with possible values that have a meaningful order or *ranking* among them, but the magnitude between successive values is not known.

Ex: Attribute *drink_size* corresponds to the size of drinks available at a fast-food restaurant. This nominal attribute has three possible values: *small, medium*, and *large*. The values have a meaningful sequence.

Ordinal attributes may also be obtained from the discretization of numeric quantities by splitting the value range into a finite number of ordered categories.

The central tendency of an ordinal attribute can be represented by its mode and its median (the middle value in an ordered sequence), but the mean cannot be defined.

Note: nominal, binary, and ordinal attributes are *qualitative*. That is, they *describe* a feature of an object.

Numeric Attributes

A **numeric attribute** is *quantitative*; that is, it is a measurable quantity, represented in integer or real values. Numeric attributes can be *interval-scaled* or *ratio-scaled*.

Interval-Scaled Attributes

Interval-scaled attributes are measured on a scale of equal-size units. The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition to providing a ranking of values, such attributes allow us to compare and quantify the *difference* between values.

Ex: Interval-scaled attributes. A *temperature* attribute is interval-scaled. Suppose that we have the outdoor *temperature* value for a number of different days, where each day is an object. By ordering the values, we obtain a ranking of the objects with respect to *temperature*. In addition, we can quantify the difference between values. For example, a temperature of 20^{o}C is five degrees higher than a temperature of 15^{o}C. Calendar dates are another example. For instance, the years 2002 and 2010 are eight years apart.

Ratio-Scaled Attributes

A **ratio-scaled attribute** is a numeric attribute with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value. In addition, the values are ordered, and we can also compute the difference between values, as well as the mean, median, and mode.

Ex: *count* attributes such as *years of experience* (e.g., the objects are employees) and *number of words* (e.g., the objects are documents). Additional examples include attributes to measure weight, height, latitude and longitude coordinates.

BASIC STATISTICAL DESCRIPTIONS OF DATA

Statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers.

This section discusses three areas of basic statistical descriptions. We start with *measures of central tendency,* which measure the location of the middle or center of a data distribution

(mean, median, mode, and midrange). Also *dispersion of the data*. That is, how are the data spread out? The most common data dispersion measures are the *range*, *quartiles*, and *interquartile range*; the *five-number summary* and *boxplots*; and the *variance* and *standard deviation* of the data. graphic displays of basic statistical descriptions to visually inspect our data (bar charts, pie charts, line graphs, *quantile plots*, *quantile–quantile plots*, *histograms*, and *scatter plots*).

Measuring the Central Tendency: Mean, Median, and Mode

Center of set of data is arithmetic mean i.e; the **MEAN** (average) of set of values is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

mean() i.e; sum()/count().

WEIGHTED MEAN:

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_N x_N}{w_1 + w_2 + \cdots + w_N}$$

called as weighted arithmetic mean.

Use - *trimmed mean* because of the disadvantage of using mean is its avoid 2% of high and low sensitivity to extreme values.

For skewed (asymmetric) data, center of data is median.

(for calculating median) N values are in sorted order,

if N is even-> avg of the middle two numbers

N is odd-> center value

MEDIAN:

$$median = L_1 + \left(\frac{N/2 - (\sum freq)_l}{freq_{median}} \right) width$$

L1->lower boundary of the median interval

N-> No. of values

(∑freq)l->sum of all the intervals that are lower than the median intervals that are lower than the median interval.

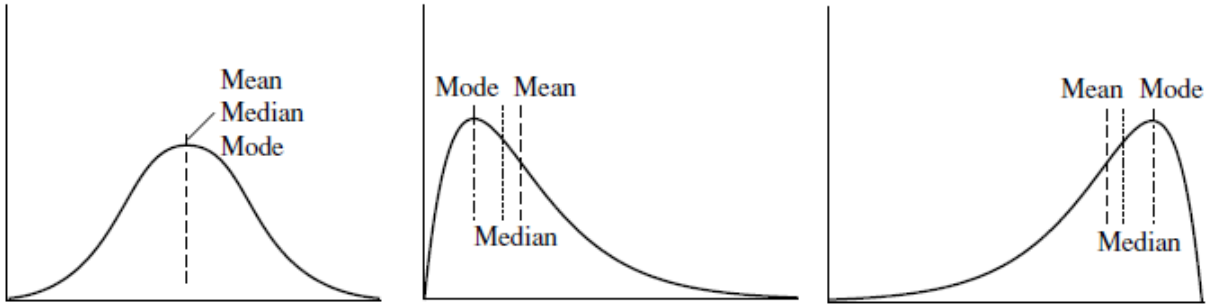
MODE:

i.e; that occurs most frequently in the set.

Data sets with one, two (or) three modes are called unimodal, bimodal and trimodal & multimodal

mean-mod = 3*(mean-median)

The **MIDRANGE** can also be used to assess the central tendency of a numeric data set. It is the average of the largest and smallest values in the set.



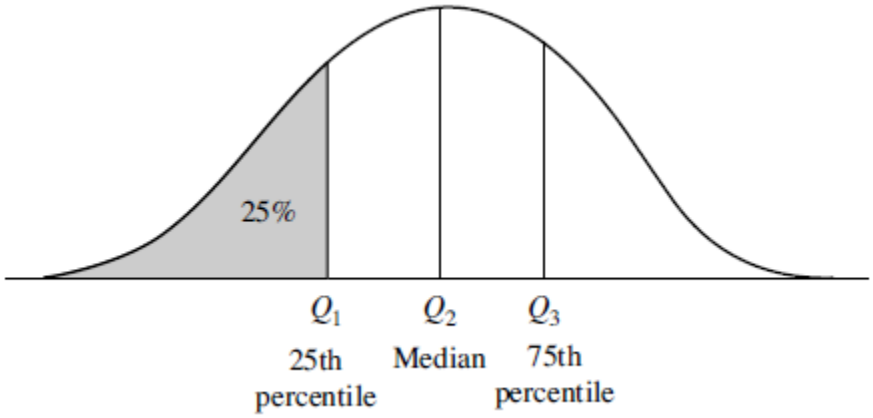
Mean, median, and mode of symmetric versus positively and negatively skewed data.

Measuring the Dispersion of Data: Range, Quartiles, Variance, Standard Deviation, and Interquartile Range

RANGE, QUARTILES, AND INTERQUARTILE RANGE

Let $x_1, x_2, \ldots, x_n$ be a set of observations for some numeric attribute, X. The **RANGE** of the set is the difference between the largest [max()] and smallest [min()] values.

Suppose that the data for attribute X are sorted in increasing numeric order. Imagine that we can pick certain data points so as to split the data distribution into equal-size consecutive sets, as shown:



Plot of the data distribution for some attribute X. The quantiles plotted are quartiles. The three quartiles divide the distribution into four equal-size consecutive subsets. The second quartile corresponds to the median.

These data points are called *quantiles*. **QUANTILES** are points taken at regular intervals of a data distribution, dividing it into essentially equal size consecutive sets.

The k^{th} *q-quantile* for a given data distribution is the value *x* such that at most k/q of the data values are less than *x* and at most $(q-k)/q$ of the data values are more than *x*, where *k* is an integer such that $0 < k < q$. There are q-1 q-quantiles.

Ex: The 2-quantile is the data point dividing the lower and upper halves of the data distribution. It corresponds to the median. The 4-quantiles are the three data points that split the data distribution into four equal parts; each part represents one-fourth of the data distribution. They are more commonly referred to as **quartiles**.

The 100-quantiles are more commonly referred to as **percentiles**; they divide the data distribution into 100 equal-sized consecutive sets. The median, quartiles, and percentiles are the most widely used forms of quantiles.

The quartiles give an indication of a distribution's center, spread, and shape. The **first quartile**, denoted by Q_1, is the 25^{th} percentile. It cuts off the lowest 25% of the data. The **third quartile**, denoted by Q_3, is the 75^{th} percentile—it cuts off the lowest 75% (or highest 25%) of the data. The second quartile is the 50^{th} percentile. As the median, it gives the center of the data distribution.

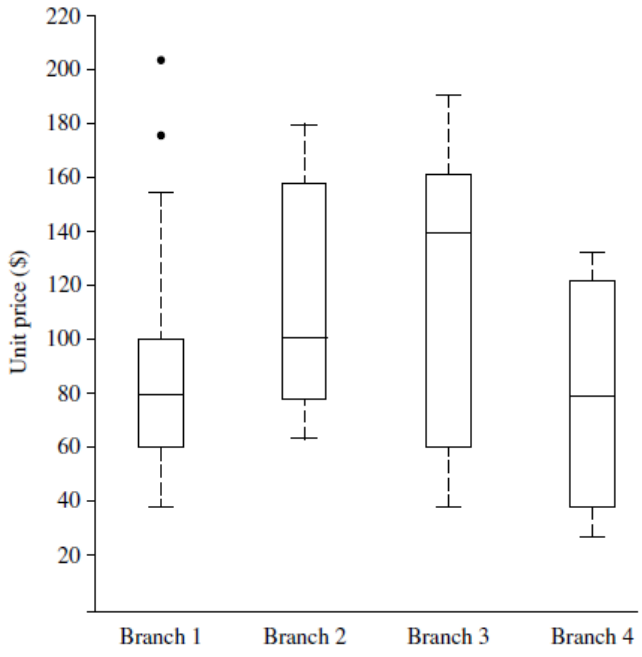
The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the **INTERQUARTILE RANGE** (**IQR**) and is defined as

$$IQR = Q_3 - Q_1$$

FIVE-NUMBER SUMMARY, BOXPLOTS, AND OUTLIERS

The **FIVE-NUMBER SUMMARY** of a distribution consists of the median (Q_2), the quartiles Q_1 and Q_3, and the smallest and largest individual observations, written in the order of *Minimum*, Q_1, *Median*, Q_3, *Maximum*.

BOXPLOTS are a popular way of visualizing a distribution. A boxplot incorporates the five-number summary as follows:



- Typically, the ends of the box are at the quartiles so that the box length is the interquartile range.
- The median is marked by a line within the box.
- Two lines (called *whiskers*) outside the box extend to the smallest (*Minimum*) and largest (*Maximum*) observations.

VARIANCE AND STANDARD DEVIATION

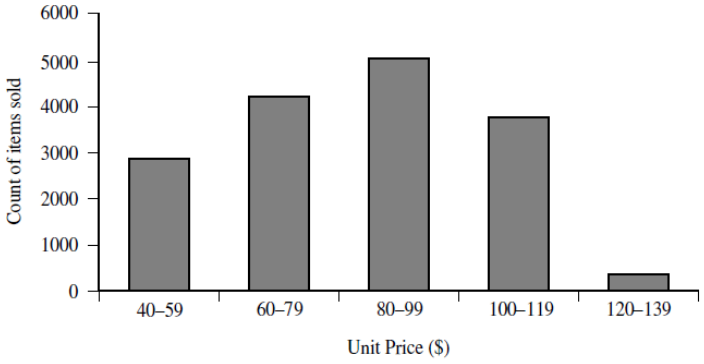
Variance of N observations, $x_1, x_2, \ldots\ldots, x_N$ is

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N}\left[\sum x_i^2 - \frac{1}{N}(\sum x_i)^2\right]$$

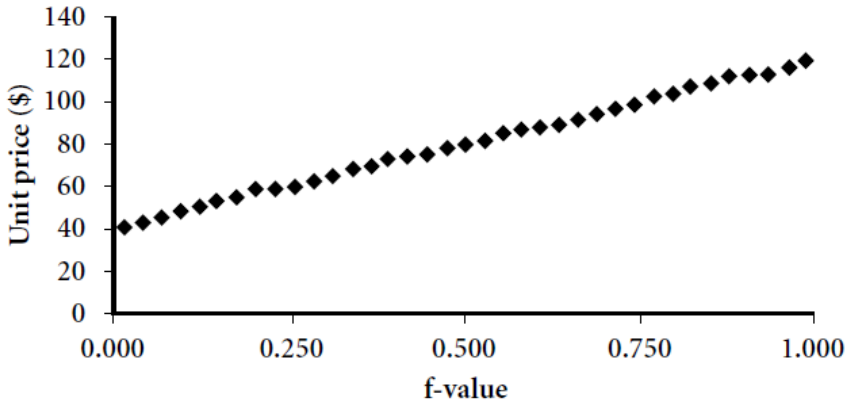
Where σ is Standard deviation.

GRAPHIC DISPLAYS OF BASIC STATISTICAL DESCRIPTIONS OF DATA

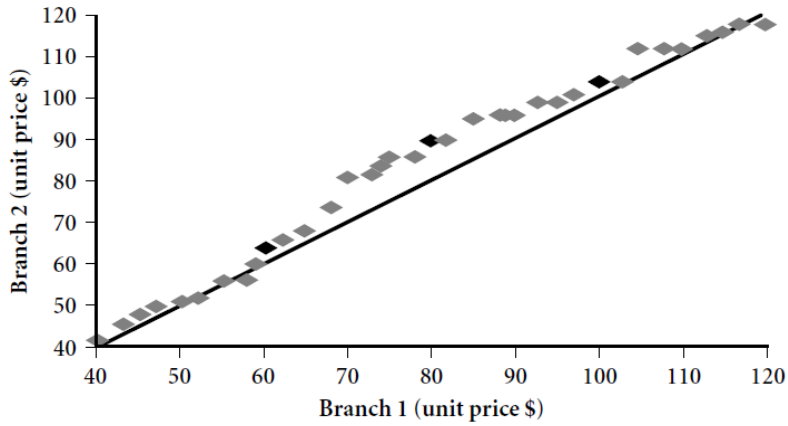
- Histograms, quantile plots, q-q plots, scatter plots and less curves
- Frequency histograms->Data buckets & Distribute data information
- Quantile plot-> q-q plot.
- Scatter plot



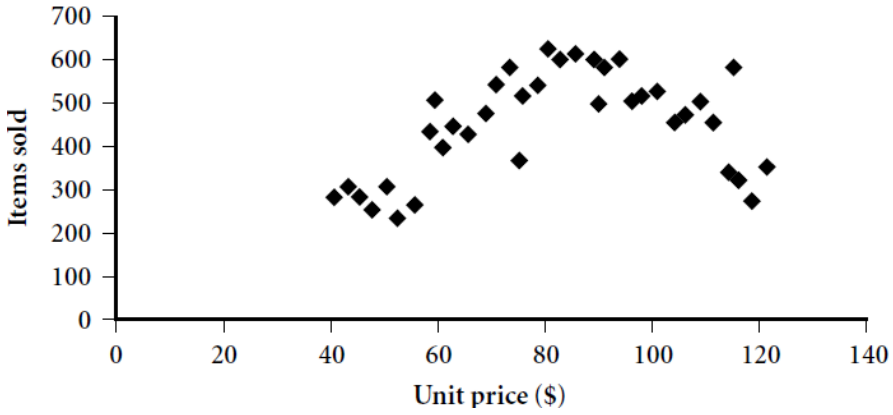
Histogram



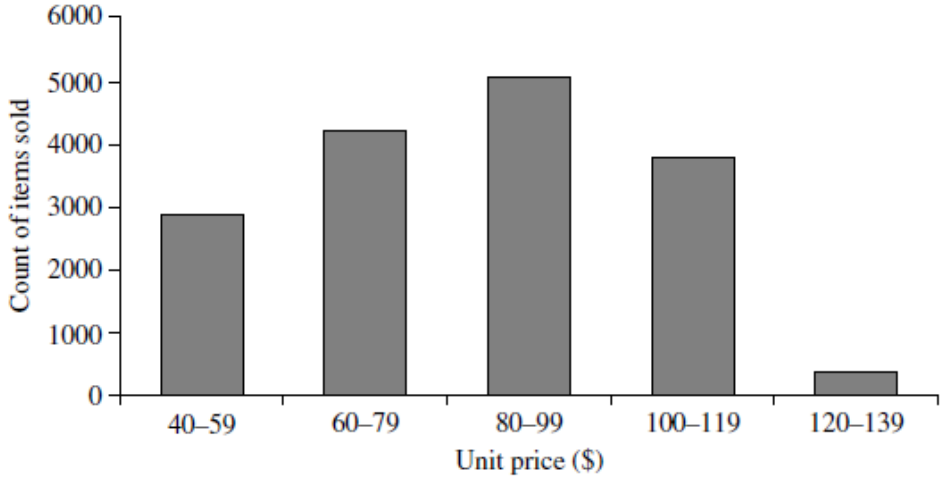
A quantile plot, A **quantile plot** is a simple and effective way to have a first look at a univariate data distribution



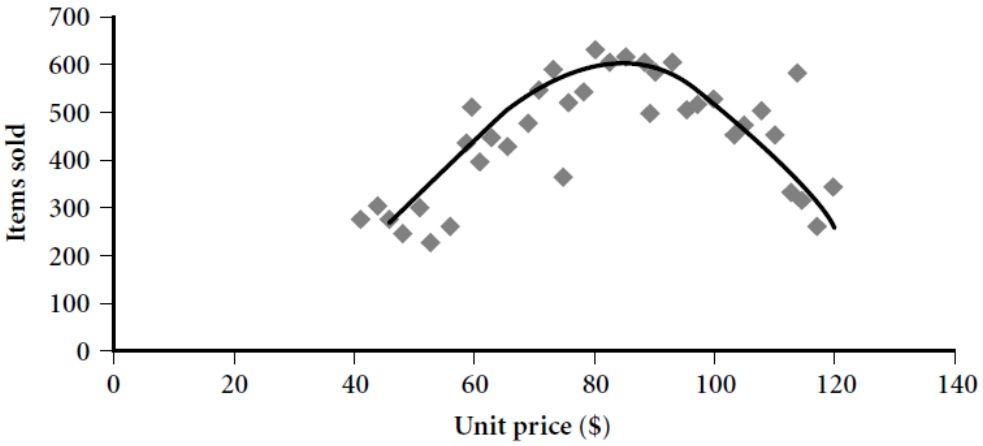
A q-q plot; A **quantile–quantile plot**, or **q-q plot**, graphs the quantiles of one univariate distribution against the corresponding quantiles of another.



A scatter plot



A Histogram



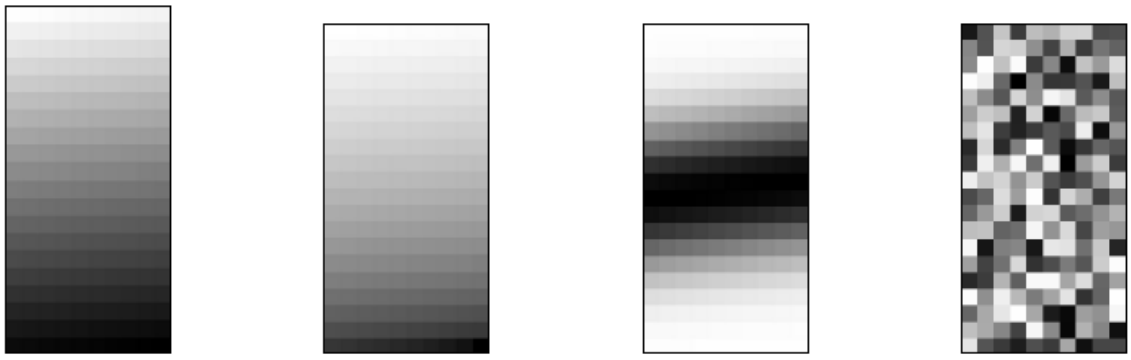
A loose curve

DATA VISUALIZATION

Data visualization aims to communicate data clearly and effectively through graphical representation.

Pixel-Oriented Visualization Techniques

A simple way to visualize the value of a dimension is to use a pixel where the color of the pixel reflects the dimension's value. For a data set of *m* dimensions, **pixel-oriented techniques** create *m* windows on the screen, one for each dimension. The *m* dimension values of a record are mapped to *m* pixels at the corresponding positions in the windows. The colors of the pixels reflect the corresponding values.



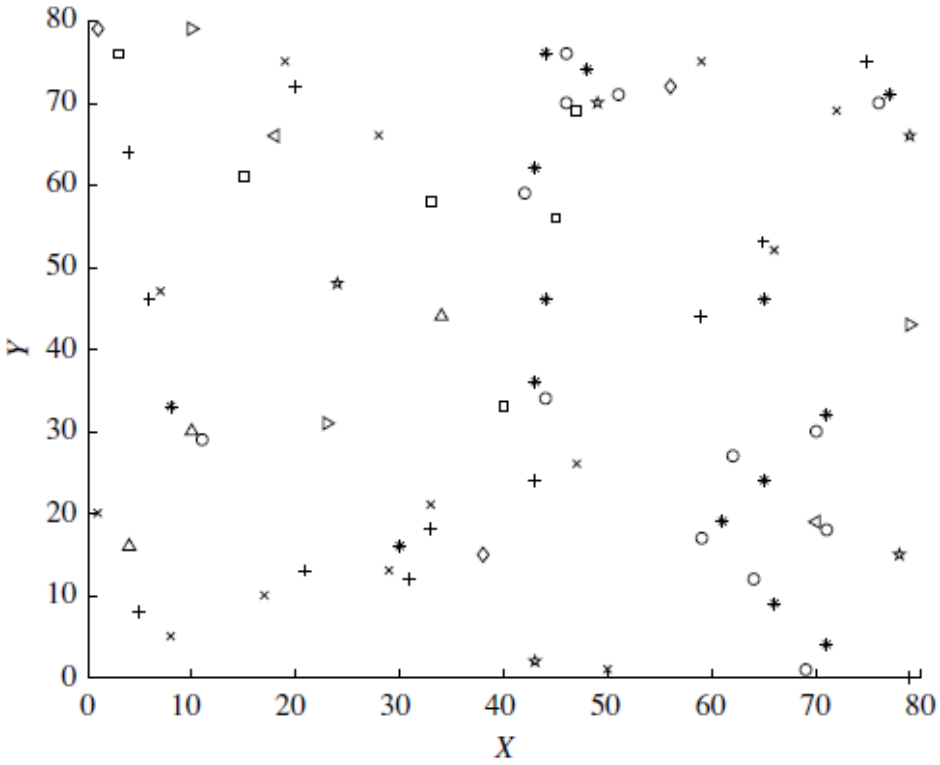
(a) *income* (b) *credit_limit* (c) *transaction_volume* (d) *age*

Customers are sorted in income-ascending order, and use this order to lay out the customer data in the four visualization windows, as shown in above. The pixel colors are chosen so that the smaller the value, the lighter the shading. Using pixel based visualization; we can easily observe the following: *credit limit* increases as *income* increases; customers whose income is in the middle range are more likely to purchase more from *AllElectronics*; there is no clear correlation between *income* and *age*.

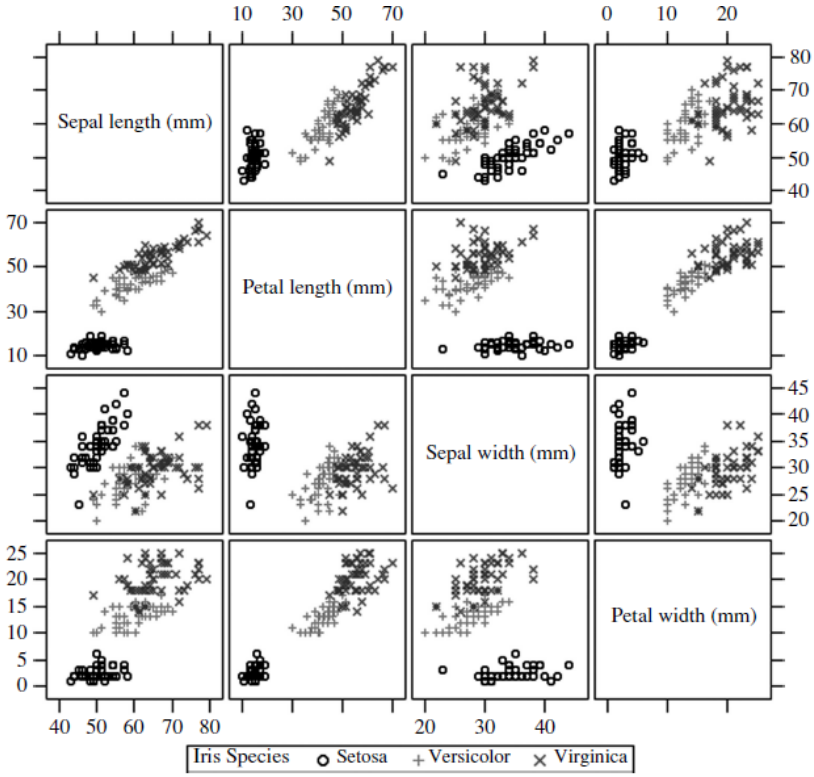
Geometric Projection Visualization Techniques

- ✓ Visualization of a 2-D data set using a scatter plot
- ✓ A **scatter plot** displays 2-D data points using Cartesian coordinates
- ✓ A third dimension can be added using different colors or shapes to represent different data points. "+" and "" tend to be co-located

- ✓ A 3-D scatter plot uses three axes in a Cartesian coordinate system. If it also uses color, it can display up to 4-D data points
- ✓ Data sets with more than four dimensions, scatter plots are usually ineffective



- ✓ **Scatter-plot matrix** technique is a useful extension to the scatter plot. For an *n* dimensional data set, a scatter-plot matrix is an *n X n* grid of 2-D scatter plots that provides a visualization of each dimension with every other dimension.

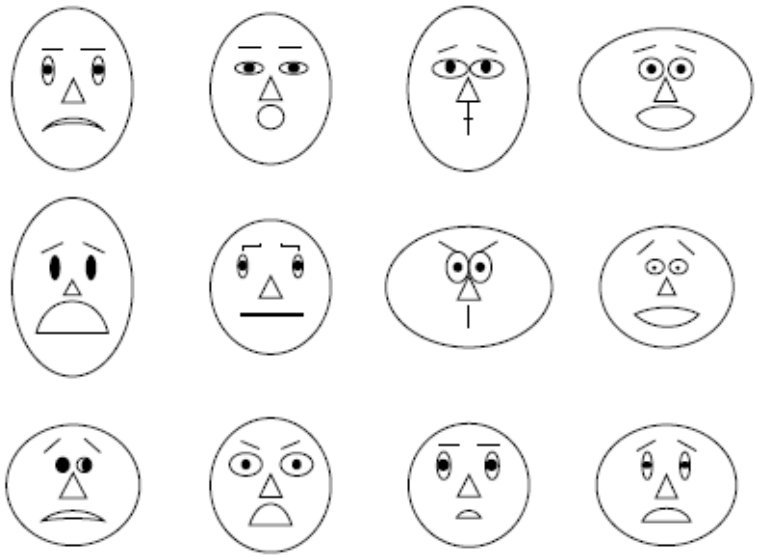


Scatter-plot matrix

- ✓ Scatter-plot matrix becomes less effective as the dimensionality increases
- ✓ **Parallel coordinates** technique draws *n* equally spaced axes, one for each dimension, parallel to one of the display axes; it cannot effectively show a data set of many records

Icon-Based Visualization Techniques

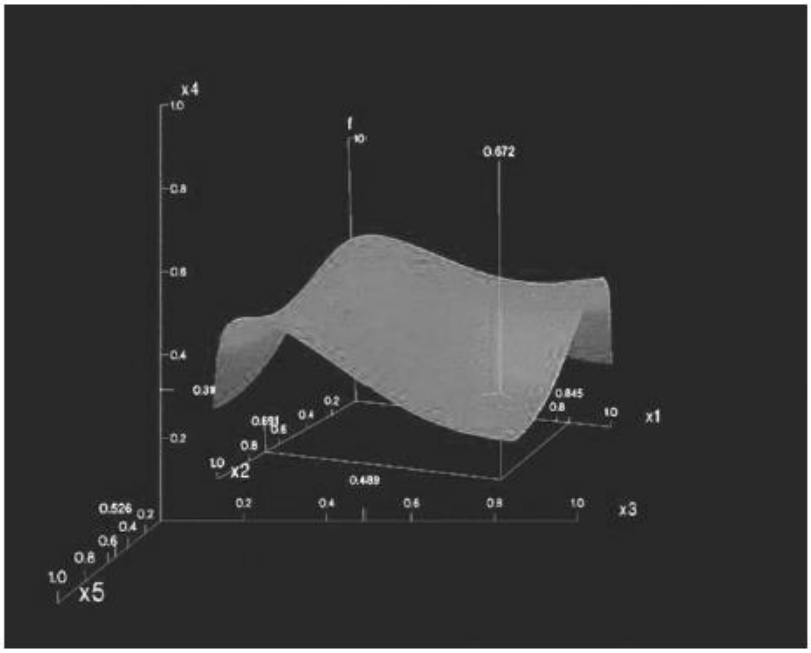
- ✓ Uses small icons to represent multidimensional data values.
- ✓ Two popular icon-based techniques: *Chernoff faces* and *stick figures*
- ✓ **Chernoff faces** were introduced in 1973 by statistician Herman Chernoff. They display multidimensional data of up to 18 variables (or dimensions) as a cartoon human face



- ✓ ***Asymmetrical Chernoff*** faces were proposed as an extension to the original technique. Since a face has vertical symmetry (along the *y*-axis), the left and right side of a face are identical, which wastes space. Asymmetrical Chernoff faces double the number of facial characteristics, thus allowing up to 36 dimensions to be displayed.
- ✓ The **stick figure** visualization technique maps multidimensional data to five-piece stick figures, where each figure has four limbs and a body. Two dimensions are mapped to the display (*x* and *y*) axes and the remaining dimensions are mapped to the angle and/or length of the limbs. Data is mapped to the display axes and stick figures.

Hierarchical Visualization Techniques

- ✓ For a large data set of high dimensionality
- ✓ Partition all dimensions into subsets
- ✓ Visualized in a hierarchical manner
- ✓ **"Worlds-within-Worlds,"** also known as *n*-Vision



- ✓ **Tree-maps** display hierarchical data as a set of nested rectangles.

Visualizing Complex Data and Relations

- ✓ Visualization techniques were mainly for numeric data. Recently, more and more non-numeric data, such as text and social networks, have become available
- ✓ A **tag cloud** is a visualization of statistics of user-generated tags. People on the Web tag various objects such as pictures, blog entries, and product reviews.
- ✓ Tag clouds are often used in two ways.
- ✓ First, in a tag cloud for a single item, we can use the size of a tag to represent the number of times that the tag is applied to this item by different users.

- ✓ Second, when visualizing the tag statistics on multiple items, we can use the size of a tag to represent the number of items that the tag has been applied to, i.e., the popularity of the tag.

MEASURING DATA SIMILARITY AND DISSIMILARITY

A **cluster** is a collection of data objects such that the objects within a cluster are *similar* to one another and *dissimilar* to the objects in other clusters.

Data Matrix versus Dissimilarity Matrix

Data matrix (or *object-by-attribute structure*): This structure stores the *n* data objects in the form of a relational table, or *n*-by-*p* matrix (*n* objects X *p* attributes):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}.$$

n-by-*p* matrix

Each row corresponds to an object. As part of our notation, we may use *f* to index through the *p* attributes.

Dissimilarity matrix (or *object-by-object structure*): This structure stores a collection of proximities that are available for all pairs of *n* objects. It is often represented by an *n*-by-*n* table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

n-by-*n*

where d(i, j) is the measured **dissimilarity** or "difference" between objects i and j. If d(i, j) is a non-negative number that is close to 0 when objects *i* and *j* are highly similar or "near" each other. d(i, j) = 0; that is, the difference between an object and itself is 0. Furthermore, d(i, j) = d(j, i).

Similarity measure, sim(i, j) = 1 – d(i, j)

Proximity Measures for Nominal Attributes

A nominal attribute can take on two or more states. For example, *map color* is a nominal attribute that may have, say, five states: *red, yellow, green, pink*, and *blue*. Let the number of states of a nominal attribute be *M*. The states can be denoted by letters, symbols, or a set of integers, such as 1, 2, . . . , *M*. Notice that such integers are used just for data handling and do not represent any specific ordering.

The dissimilarity between two objects *i* and *j* can be computed based on the ratio of mismatches:

$$d(i, j) = \frac{p - m}{p}$$

where *m* is the number of *matches* (i.e., the number of attributes for which *i* and *j* are in the same state), and *p* is the total number of attributes describing the objects.

Weights can be assigned to increase the effect of *m* or to assign greater weight to the matches in attributes having a larger number of states.

Similarity can be computed as,

$$sim(i, j) = 1 - d(i, j) = \frac{m}{p}$$

Ex: Consider the following data set

Object Identifier	test-1 (nominal)	test-2 (ordinal)	test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

test-1 is nominal, therefore p=1. Dissimilarity is calculated between objects using test-1 attribute alone.

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Proximity Measures for Binary Attributes

Binary attribute has only one of two states: 0 and 1, where 0 means that the attribute is absent and 1 means that it is present.

How can we compute the dissimilarity between two binary attributes?

One approach involves computing a dissimilarity matrix from the given binary data. If all binary attributes are thought of as having the same weight, we have the 2 X 2 contingency table,

		Object j		
		1	0	sum
Object i	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

where
q is the number of attributes that equal 1 for both objects *i* and *j*,
r is the number of attributes that equal 1 for object *i* but equal 0 for object *j*,
s is the number of attributes that equal 0 for object *i* but equal 1 for object *j*, and
t is the number of attributes that equal 0 for both objects *i* and *j*.
The total number of attributes is *p*, where $p = q + r + s + t$

Symmetric Binary Dissimilarity

$$d(i, j) = \frac{r+s}{q+r+s+t}$$

Asymmetric Binary Dissimilarity

$$d(i, j) = \frac{r+s}{q+r+s}$$

Asymmetric Binary Similarity (Jaccard Coefficient)

$$sim(i, j) = \frac{q}{q+r+s} = 1 - d(i, j)$$

Ex: Consider the following binary valued data set

name	gender	fever	cough	test-1	test-2	test-3	test-4
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Dissimilarity between few objects,

$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67,$$

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33,$$

$$d(Jim, Mary) = \frac{1+2}{1+1+2} = 0.75.$$

Dissimilarity of Numeric Data

Euclidean distance

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

Manhattan (or city block) distance

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|$$

p is the number of attributes

Euclidean and the Manhattan distance satisfy the following mathematical properties:

Non-negativity: $d(i, j) \geq 0$: Distance is a non-negative number.

Identity of indiscernibles: $d(i, i) = 0$: The distance of an object to itself is 0.

Symmetry: $d(i, j) = d(j, i)$: Distance is a symmetric function.

Triangle inequality: $d(i, j) \leq d(i, k) + d(k, j)$: Going directly from object i to object j in space is no more than making a detour over any other object k.

Euclidean distance and Manhattan distance. Let $x_1 = (1, 2)$ and $x_2 = (3, 5)$ represent two objects as shown in Figure 2.23. The Euclidean distance between the two is $\sqrt{2^2 + 3^2} = 3.61$. The Manhattan distance between the two is $2 + 3 = 5$.

Minkowski Distance is a generalization of the Euclidean and Manhattan distances. It is defined as

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where *h* is a real number such that *h >=1*
p is the number of attributes
It represents the Manhattan distance when *h* = 1 (i.e., *L*1 norm) and Euclidean distance when *h* = 2 (i.e., *L*2 norm).

The **Supremum Distance** (also referred to as L_{max}, L_∞ **norm** and as the **Chebyshev Distance**) is a generalization of the Minkowski distance for $h \rightarrow \infty$

$$d(i,j) = \lim_{h\to\infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

Weighted Euclidean Distance

$$d(i,j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \cdots + w_m|x_{ip} - x_{jp}|^2}$$

Weighting can also be applied to other distance measures.

Proximity Measures for Ordinal Attributes

Let M represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking 1, … , Mf

The dissimilarity computation with respect to f involves the following steps:

1. The value of f for the i^{th} object is x_{if} , and f has M_f ordered states, representing the ranking 1, . . . , M_f . Replace each x_{if} by its corresponding rank, $r_{if} \in \{1, \ldots, M_f\}$
2. Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto [0.0, 1.0] so that each attribute has equal weight. We perform such data normalization by replacing the rank r^{if} of the i^{th} object in the f^{th} attribute by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

3. Dissimilarity can then be computed using any of the distance measures described before for numeric attributes, using z_{if} to represent the f value for the i^{th} object.

Ex: Consider the following data set, consider only test-2 attribute

Object Identifier	test-1 (nominal)	test-2 (ordinal)	test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

1. There are three states for *test-2*: *fair*, *good*, and *excellent*, that is, $Mf = 3$.
2. Replace each value for *test-2* by its rank, the four objects are assigned the ranks 3, 1, 2, and 3, respectively
3. Normalizes the ranking by mapping rank 1 to 0.0, rank 2 to 0.5, and rank 3 to 1.0
4. Use Euclidean distance , for constructing dissimilarity matrix

5. Resultant dissimilarity matrix is,

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

Similarity values for ordinal attributes can be interpreted from dissimilarity as,

$$sim(i,j) = 1 - d(i,j)$$

Dissimilarity for Attributes of Mixed Types

- ✓ Combines the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0]
- ✓ Let the data set contains ***p* attributes** of mixed type. The dissimilarity d(i, j) between objects *i* and *j* is defined as

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

where

the indicator $\delta_{ij}^{(f)} = 0$ if either (1) x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute f for object i or object j), or (2) $x_{if} = x_{jf} = 0$ and attribute f is asymmetric binary; otherwise, $\delta_{ij}^{(f)} = 1$

The contribution of attribute f to the dissimilarity between i and j (i.e., $d_{ij}^{(f)}$) is computed dependent on its type:

- If f is numeric: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$, where h runs over all nonmissing objects for attribute f.
- If f is nominal or binary: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; otherwise, $d_{ij}^{(f)} = 1$.
- If f is ordinal: compute the ranks r_{if} and $z_{if} = \frac{r_{if} - 1}{M_f - 1}$, and treat z_{if} as numeric.

*** *A more preferable approach is to process all attribute types together, performing a single analysis. One such technique combines the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0].*

Ex: Consider the following data set, consider only test-3 attribute

Object Identifier	test-1 (nominal)	test-2 (ordinal)	test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

Compute $d_{(3,1)}^{(3)}$ using,

If f is numeric: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$,

where h runs over all nonmissing objects for attribute f.

Then

$max_h x_{h3}$ = 64 and $min_h x_{h3}$ = 22.

Next, $d_{(3,1)}^{(3)} = \frac{64-45}{64-22}$, results with the value 0.45

Similarly other d(i, j) values are computed which results in the following dissimilarity matrix,

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & & & \\ 0.55 & 0 & & \\ 0.45 & 1.00 & 0 & \\ 0.40 & 0.14 & 0.86 & 0 \end{bmatrix}$$

Dissimilarity matrices for the three attributes for the following data set,

Object Identifier	test-1 (nominal)	test-2 (ordinal)	test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

Compute d(3,1),

$$\text{The indicator } \delta_{ij}^{(f)} = 1 \text{ for each of the three attributes, } f$$

$$d(3, 1) = \frac{1(1)+1(0.50)+1(0.45)}{3} = 0.65.$$

Similarly compute other dissimilarity values of [d(i, j)] ,

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & & & \\ 0.85 & 0 & & \\ 0.65 & 0.83 & 0 & \\ 0.13 & 0.71 & 0.79 & 0 \end{bmatrix}$$

Cosine Similarity

- ✓ Used for finding similarity for Document based Records
- ✓ Document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document.
- ✓ Thus, each document is an object represented by what is called a ***term-frequency vector***
- ✓ Term-frequency vectors are typically very long and **sparse**
- ✓ **Cosine similarity** is a measure of similarity that can be used to compare documents

Ex: Let ***x*** and ***y*** be two vectors for comparison. Consider the following data set,

Document Vector or Term-Frequency Vector

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Cosine Similarity is calculated using,

$$sim(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||}$$

where $||x||$ is the Euclidean norm of vector $x = (x_1, x_2, \ldots, x_p)$, defined as $\sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$

- ✓ **When attributes are binary-valued**, the cosine similarity function can be interpreted n terms of shared features or attributes.

- Then $\boldsymbol{x}^t \cdot \boldsymbol{y}$ is the number of attributes possessed (i.e., shared) by both ***x*** and ***y***, and |x||y| is the *geometric mean* of the number of attributes possessed by ***x*** and the number possessed by ***y***.

- Thus, *sim (x, y)* is a measure of relative possession of common attributes.

$$sim(x, y) = \frac{x \cdot y}{x \cdot x + y \cdot y - x \cdot y}$$

which is the ratio of the number of attributes shared by ***x*** and ***y*** to the number of attributes possessed by ***x*** or ***y***. This function, known as the **Tanimoto coefficient** or **Tanimoto distance**

Note: Dissimilarity, d(x,y) = 1 – sim(x,y)

Data Warehouse and OLAP Technology

- What is a Data Warehouse?
- A Multidimensional Data Model
- Data Warehouse Architecture
- Data Warehouse Implementation
- From Data Warehouse to Data Mining

What is Data Warehouse?

A Data Warehouse is a subject–oriented, integrated, time variant and non-volatile collection of data in support to management's decision making system.

Subject-oriented: DW is organized around major subjects. Relies on subject issues & excludes data which is not useful in decision support system.

Integrated: DW is constructed by integration of multiple sources of data (Heterogeneous). Data Preprocessing is involved (Data Cleaning & Data Integration) to maintain consistency.

Time Variant: Data are stored to provide information from past historic perspective.

Non-Volatile: It doesn't require transaction processing, recovery and concurrency control. DW requires only two operations in data accessing:

- Initial loading of data
- Access of data

Differences Between OLTP (On-Line Transaction Processing) & OLAP (On-Line Analytical Processing)

Feature	OLTP	OLAP
Characteristic	Operational Processing	Information Processing
Orientation	Transaction	Analysis
User	Clerk, DBA, DB Professional	Knowledge Worker (Ex: analyst)
Function	Day-by-day operations	Long-term informational requirements, decision support
DB Design	ER-Based; Application Oriented	Star/ Snow flake; Subject Oriented
Data	Current; guaranteed up-to-date	Historical; accuracy maintained over time
Summarization	Primitive and highly detailed	Summarized and consolidated
View	Detailed flat relations	Summarized multidimensional
Unit of Work	Short simple transaction	Complex query
Access	Read/ Write	Mostly Read
Focus	Data in	Information out
Operations	Index/ hash on primary key	Lots of scans
Number of records accessed	Tens	Millions
No. of users	Thousands	Hundreds

DB size	100MB to GB	100GB to TB
Priority	High Performance High Availability	High Flexibility
Metric	Transaction Throughput	Query Throughput, response time

A Multidimensional Data Model

Data warehouses and OLAP tools are based on *multi-dimensional data model.* This model views data in the form of a ***Data Cube.*** A ***Data Cube*** allows data to be modeled and viewed in multiple dimensions. It is defined by *facts* and *dimensions.*

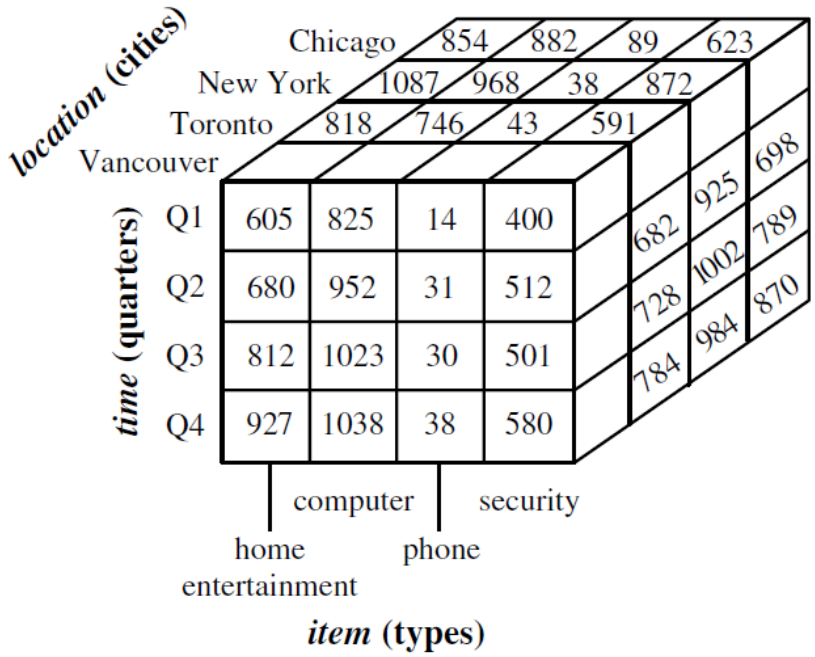
In general terms, dimensions are the perspectives or entities with respect to which an organization wants to keep records. For example, *AllElectronics* may create a *sales* data warehouse in order to keep records of the store's sales with respect to the dimensions *time*, *item*, *branch*, and *location*. These dimensions allow the store to keep track of things like monthly sales of items and the branches and locations at which the items were sold. Each dimension may have a table associated with it, called a dimension table, which further describes the dimension. For example, a dimension table for *item* may contain the attributes *item name, brand*, and *type*. Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions. A multidimensional data model is typically organized around a central theme, like *sales*, for instance. This theme is represented by a fact table. Facts are numerical measures. Think of them as 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 = "Vancouver"			
	item *(type)*			
time *(quarter)*	*home entertainment*	*computer*	*phone*	*security*
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

A 2-D view of sales data for *AllElectronics* according to the dimensions *time* and *item*, where the sales are from branches located in the city of Vancouver. The measure displayed is *dollars sold* (in thousands).

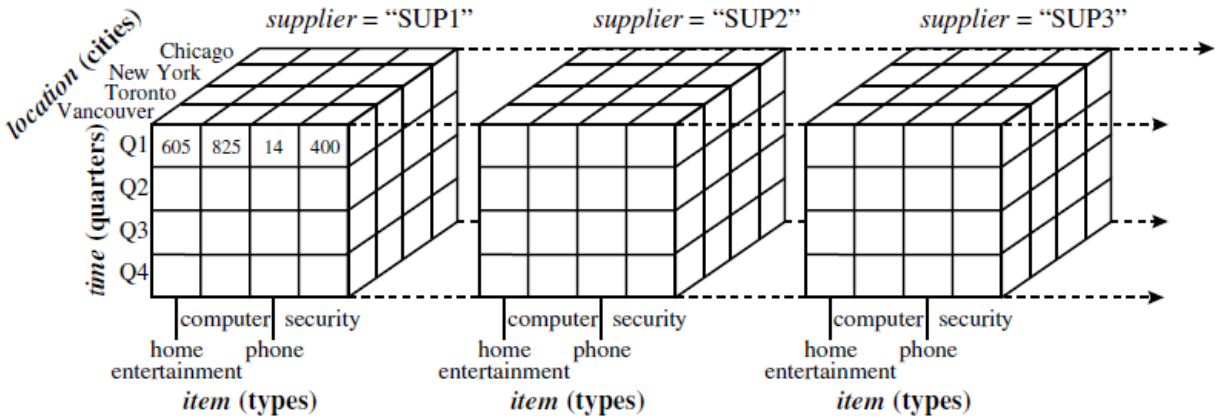
	location = "Chicago"				location = "New York"				location = "Toronto"				location = "Vancouver"			
	item				item				item				item			
time	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	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

A 3-D view of sales data for *AllElectronics*, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars sold* (in thousands).



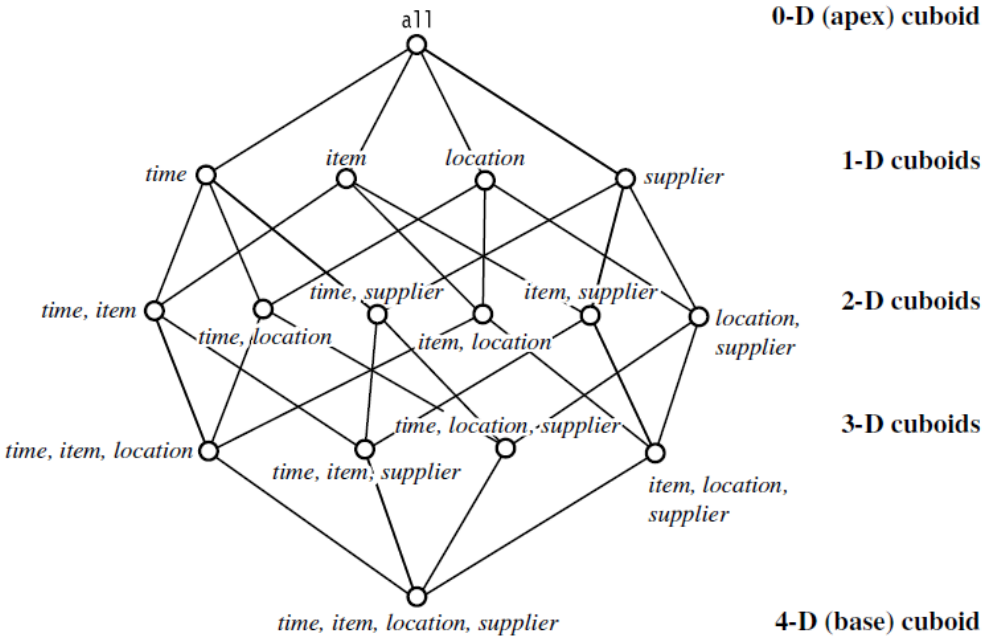
A 3-D data cube representation of the data in Table 3.3, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars sold* (in thousands).

View of sales can be increased to fourth dimension by adding an additional supplier.



A 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).

The above tables show the data at different degrees of summarization. In the data warehousing research literature, a data cube such as each of the above is often referred to as a cuboid. Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions. The result would form a *lattice* of cuboids, each showing the data at a different level of summarization, or group by. The lattice of cuboids is then referred to as a data cube. The following figure shows a lattice of cuboids forming a data cube for the dimensions *time*, *item*, *location*, and *supplier*.



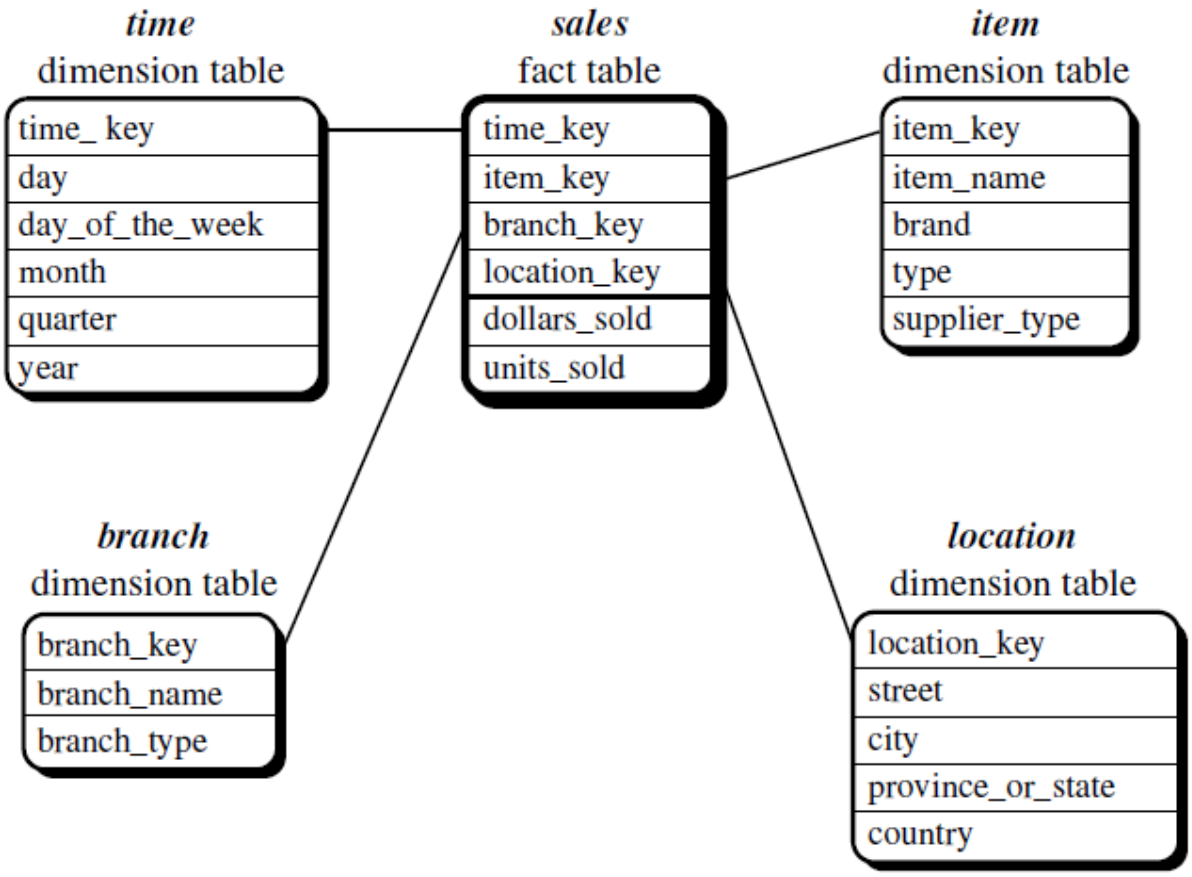
Lattice of cuboids, making up a 4-D data cube for the dimensions *time*, *item*, *location*, and *supplier*. Each cuboid represents a different degree of summarization.

- Apex cuboid holds highest level of summarization. Generally denoted by all.
- Base cuboid holds lowest level of summarization.

Stars, Snowflakes and Fact Constellations Schemas (Schemas for Multidimensional Databases)

Star Schema:

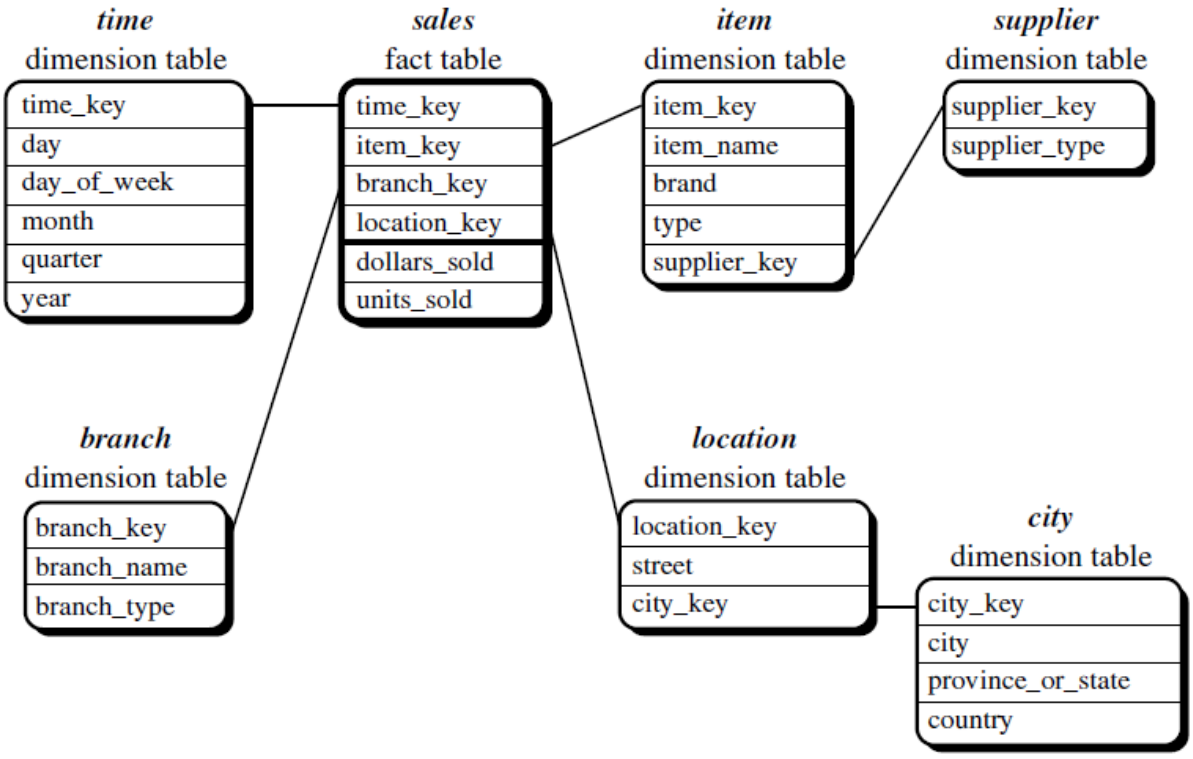
The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (fact table) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.



Star schema of a data warehouse for sales.

Snowflake schema:

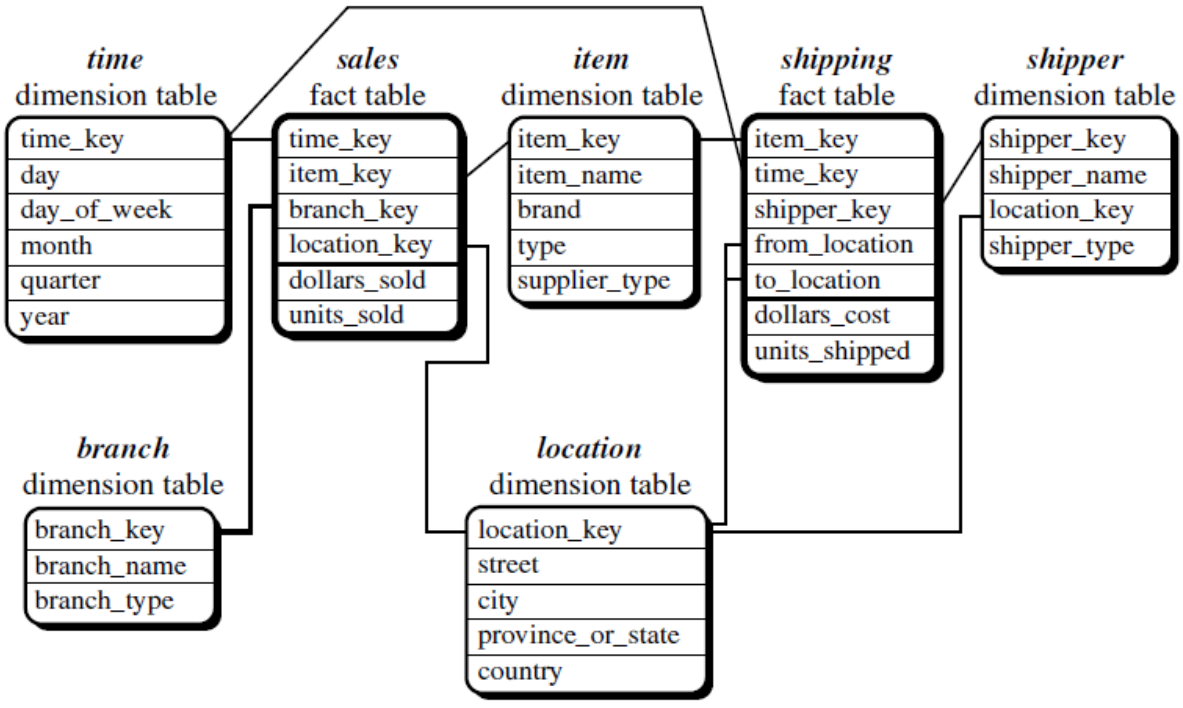
The snowflake schema is a variant of the star schema model, where some dimension tables are *normalized*, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.



Snowflake schema of a data warehouse for sales.

Fact constellation:

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



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

Data Cube Measures:

A data cube measure is a numerical function that can be evaluated at each point in the data cube space. A measure value is computed for a given point by aggregating the data corresponding to the respective dimension-value pairs defining the given point. Measures can be organized into three categories:

Distributive: An aggregate function is *distributive* if it can be computed in a distributed manner as follows. Suppose the data are partitioned into *n* sets. We apply the function to each partition, resulting in *n* aggregate values. If the result derived by applying the function to the *n* aggregate values is the same as that derived by applying the function to the entire data set (without partitioning), the function can be computed in a distributed manner. For example, count() can be computed for a data cube by first partitioning the cube into a set of subcubes, computing count() for each subcube, and then summing up the counts obtained for each subcube. Hence, count() is a distributive aggregate function. For the same reason, sum(), min(), and max() are distributive aggregate functions. A measure is *distributive* if it is obtained by applying a distributive aggregate function. Distributive measures can be computed efficiently because they can be computed in a distributive manner.

Algebraic: An aggregate function is *algebraic* if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded positive integer), each of which is obtained by applying a distributive aggregate function. For example, avg() (average) can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions. Similarly, it can be shown that min N() and max N() (which find the *N* minimum and *N* maximum values, respectively, in a given set) and standard deviation() are algebraic aggregate functions. A measure is *algebraic* if it is obtained by applying an algebraic aggregate function.

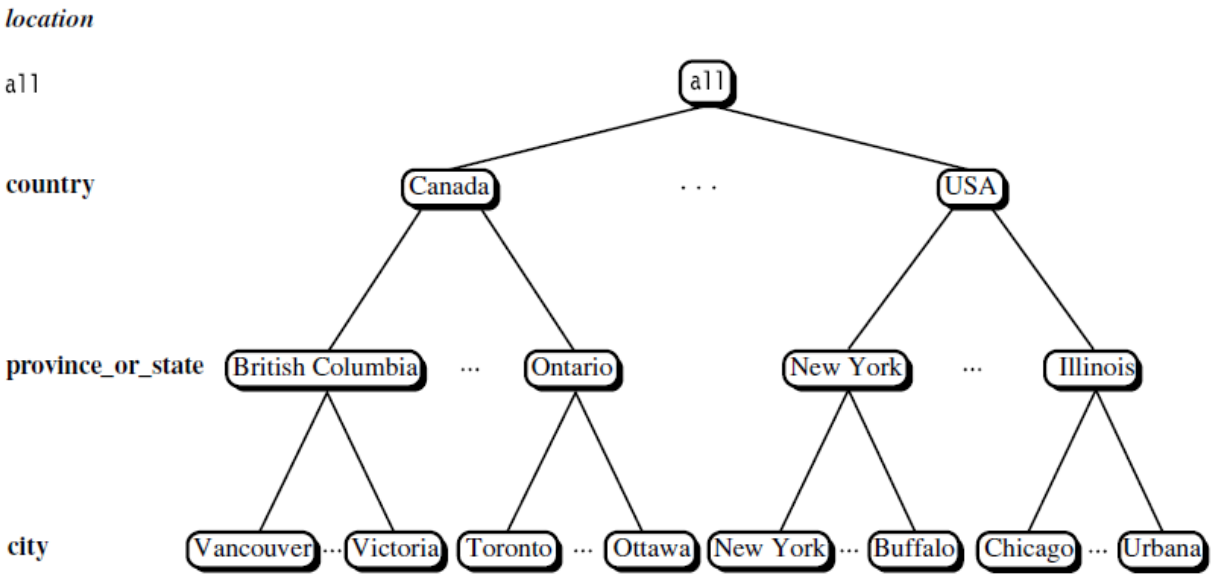
Holistic: An aggregate function is *holistic* if there is no constant bound on the storage size needed to describe a subaggregate. That is, there does not exist an algebraic function with *M* arguments (where *M* is a constant) that characterizes the computation. Common examples of holistic functions include median(), mode(), and rank(). A measure is *holistic* if it is obtained by applying a holistic aggregate function.

Concept Hierarchy:

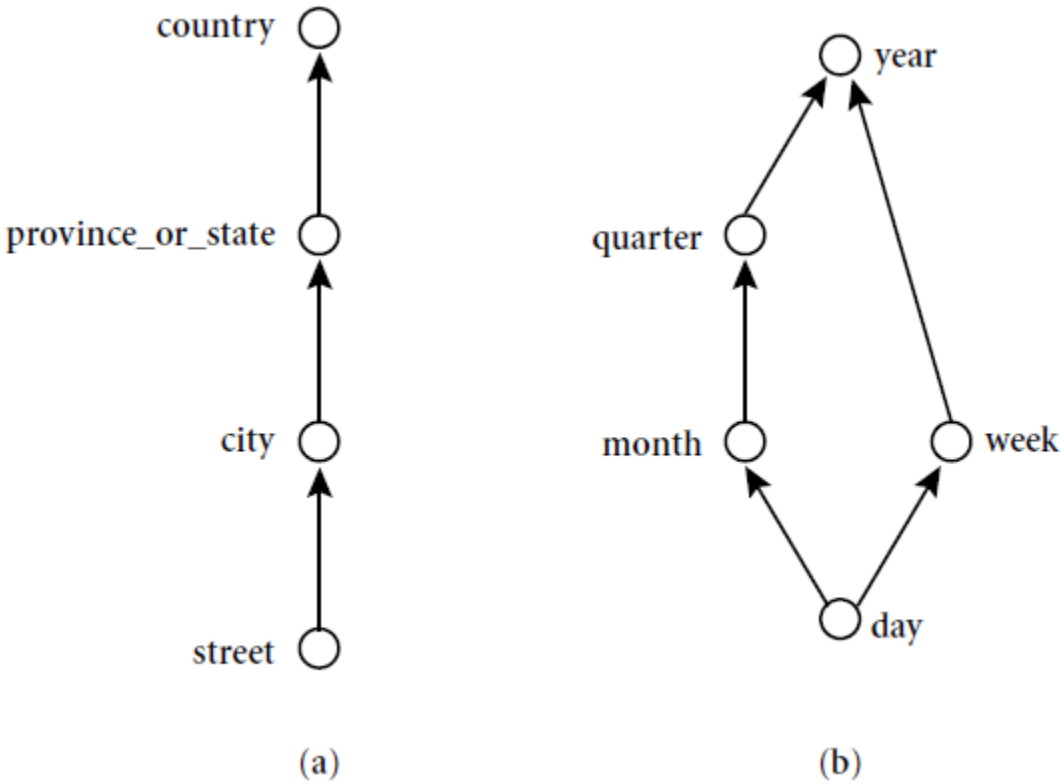
A **concept hierarchy** defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. Consider a concept hierarchy for the dimension *location*. City values for *location* include Vancouver, Toronto, New York, and Chicago. Each city, however, can be mapped to the province or state to which it belongs. For example, Vancouver can be mapped to British Columbia, and Chicago to Illinois. The provinces and states can in turn be mapped to the country to which they belong, such as Canada or the USA. These mappings form a concept hierarchy for the dimension *location*, mapping a set of low-level concepts (i.e., cities) to higher-level, more general concepts (i.e., countries).

For example, suppose that the dimension *location* is described by the attributes *number, street, city, province or state, zipcode*, and *country*. These attributes are related by a total order, forming a concept hierarchy such as "*street* < *city* < *province or state* < *country*". Alternatively, the attributes of a dimension may be organized in a partial order, forming a

lattice. An example of a partial order for the *time* dimension based on the attributes *day, week, month, quarter*, and *year* is "*day* < {*month* <*quarter; week*} < *year*".



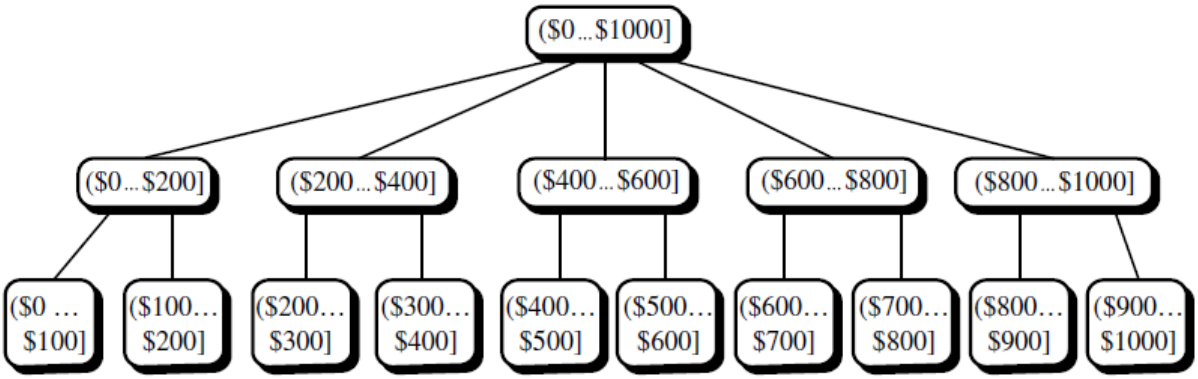
A concept hierarchy for the dimension *location*



Hierarchical and lattice structures of attributes in warehouse dimensions:
(a) a hierarchy for *location*; (b) a lattice for *time*

A concept hierarchy that is a total or partial order among attributes in a database schema is called a **schema hierarchy**. Data mining systems should provide users with the flexibility to tailor predefined hierarchies according to their particular needs.

Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a **set-grouping hierarchy**. A total or partial order can be defined among groups of values. An example of a set-grouping hierarchy is shown in below figure for the dimension *price*, where an interval ($*X*…$*Y*] denotes the range from $*X* (exclusive) to $*Y* (inclusive). There may be more than one concept hierarchy for a given attribute or dimension, based on different user viewpoints. For instance, a user may prefer to organize *price* by defining ranges for *inexpensive, moderately priced*, and *expensive*.



A concept hierarchy for the attribute *price*

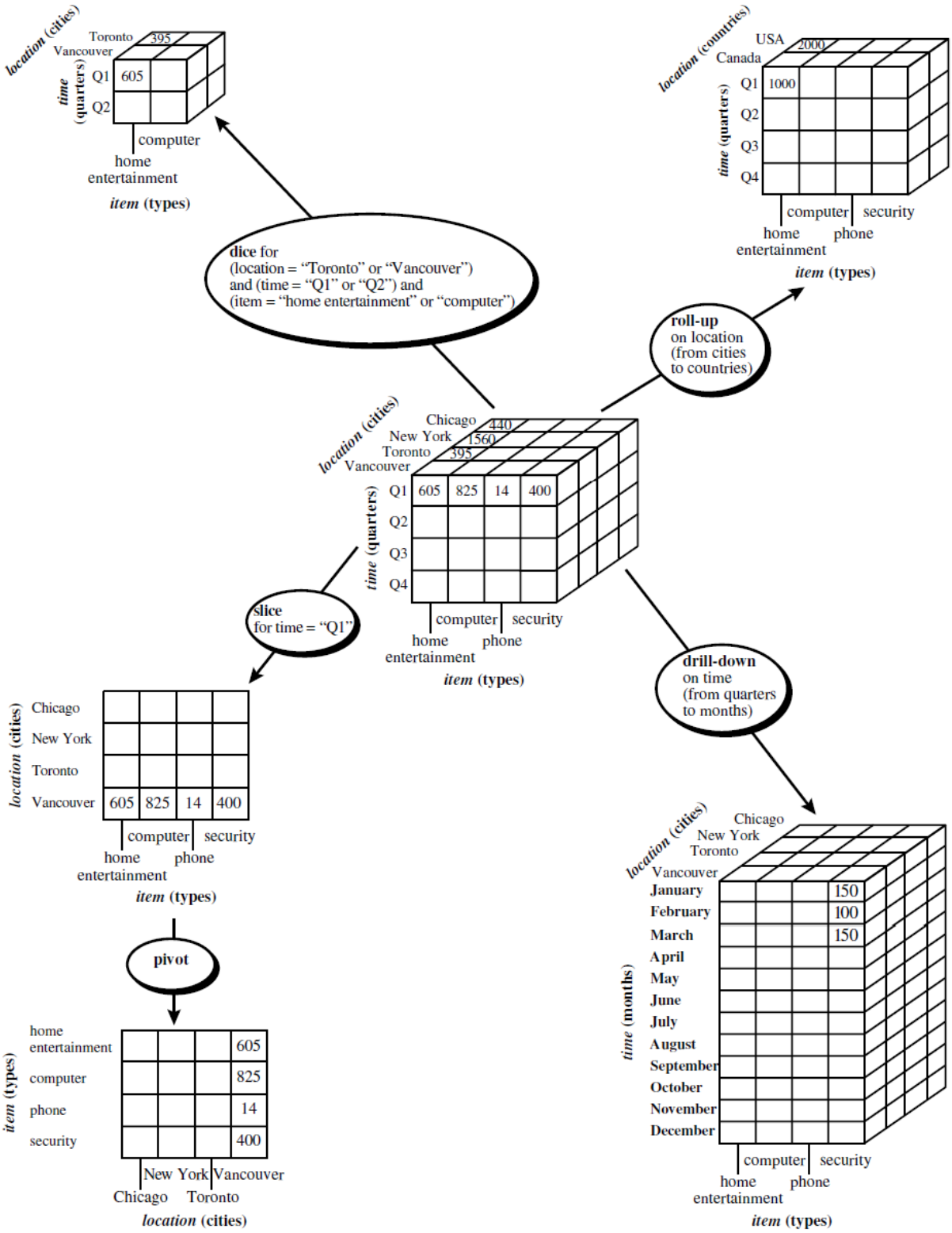
OLAP Operations in the Multidimensional Data Model
(Data Cube Operations)

Concept Hierarchies are very much useful in OLAP Data Cube operations. In the multidimensional model, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. This organization provides users with the flexibility to view data from different perspectives. A number of OLAP data cube operations exist to materialize these different views, allowing interactive querying and analysis of the data at hand.

Roll-up: The roll-up operation (also called the *drill-up* operation by some vendors) performs aggregation on a data cube, either by *climbing up a concept hierarchy* for a dimension or by *dimension reduction*. The figure given below shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for *location* given in figure. This hierarchy was defined as the total order "*street* < *city* < *province or state* < *country*." The roll-up operation shown aggregates the data by ascending the *location* hierarchy from the level of *city* to the level of *country*. In other words, rather than grouping the data by city, the resulting cube groups the data by country. When roll-up is performed by dimension reduction, one or more dimensions are removed from the given cube. For example, consider a sales data cube containing only the two dimensions *location* and *time*. Roll-up may be performed by removing, say, the *time* dimension, resulting in an aggregation of the total sales by location, rather than by location and by time.

Drill-down: Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either *stepping down a concept hierarchy* for a dimension or *introducing additional dimensions*. The figure shown the result of a drill-down operation performed on the central cube by stepping down a

concept hierarchy for *time d*efined as "*day* < *month* < *quarter* < *year*." Drill-down occurs by descending the *time* hierarchy from the level of *quarter* to the more detailed level of *month*. The resulting data cube details the total sales per month rather than summarizing them by quarter. Because a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube.



OLAP operations on multidimensional data

Slice and dice: The *slice* operation performs a selection on one dimension of the given cube, resulting in a subcube. Figure 3.10 shows a slice operation where the sales data are selected from the central cube for the dimension *time* using the criterion *time = "Q1"*. The *dice* operation defines a subcube by performing a selection on two or more dimensions. The above figure shows a dice operation on the central cube based on the following selection criteria that involve three dimensions: (*location = "Toronto"* or *"Vancouver"*) and (*time = "Q1"* or *"Q2"*) and (*item = "home entertainment"* or *"computer"*).

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

Other OLAP operations: Some OLAP systems offer additional drilling operations. For example, drill-across executes queries involving (i.e., across) more than one fact table. The drill-through operation uses relational SQL facilities to drill through the bottom level of a data cube down to its back-end relational tables.

Data Warehouse Architecture

Steps for the Design and Construction of Data Warehouse

a. The Design of a Data Warehouse: A Business Analysis Framework

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

- The **top-down view** allows the selection of the relevant information necessary for the data warehouse. This information matches the current and future business needs.
- The **data source view** exposes the information being captured, stored, and managed by operational systems. This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables. Data sources are often modeled by traditional data modeling techniques, such as the entity-relationship model or CASE (computer-aided software engineering) tools.
- The **data warehouse view** includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- The **business query view** is the perspective of data in the data warehouse from the viewpoint of the end user.

b. The Process of Data Warehouse Design

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

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

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

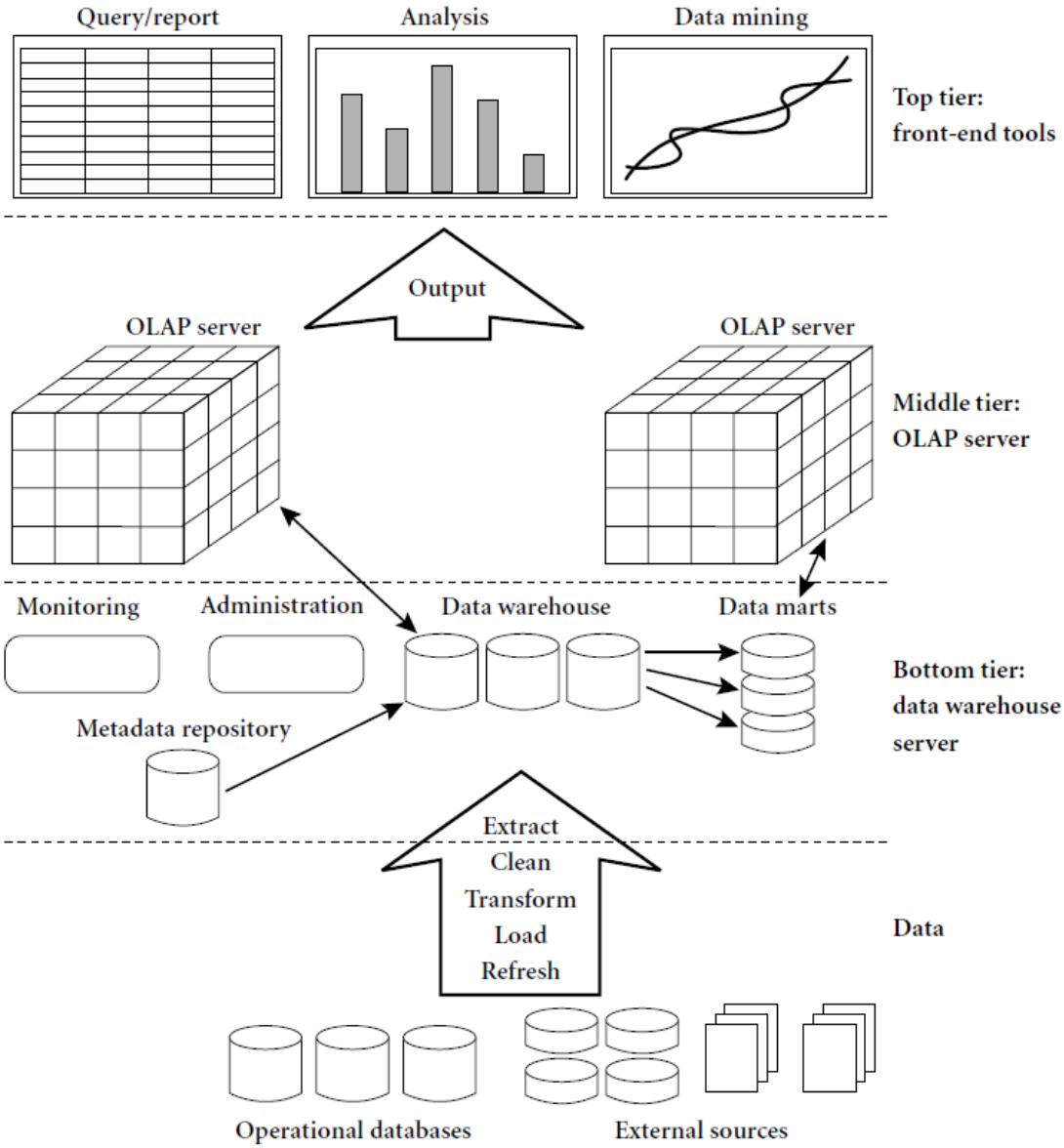
1. Choose a *business process* to model, for example, orders, invoices, shipments, inventory, account administration, sales, or the general ledger. If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed. However, if the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.
2. Choose the *grain* of the business process. The grain is the fundamental, atomic level of data to be represented in the fact table for this process, for example, individual transactions, individual daily snapshots, and so on.
3. Choose the *dimensions* that will apply to each fact table record. Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
4. Choose the *measures* that will populate each fact table record. Typical measures are numeric additive quantities like *dollars sold* and *units sold.*

A Three-Tier Data Warehouse Architecture

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

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

warehouse. The data are extracted using application program interfaces known as gateways. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Open Linking and Embedding for Databases) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents.



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

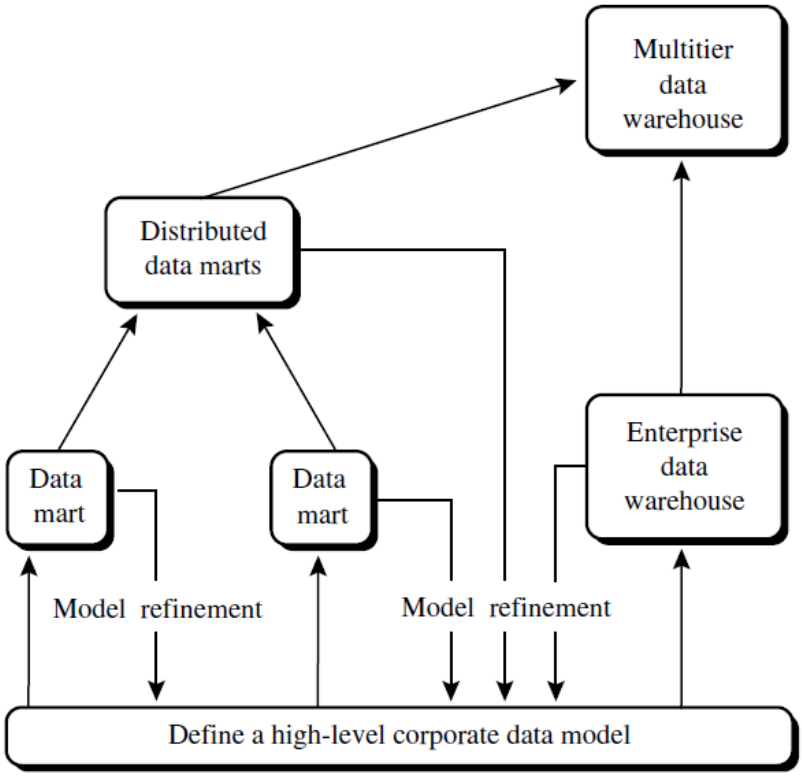
From the architecture point of view, there are three data warehouse models: the *enterprise warehouse*, the *data mart*, and the *virtual warehouse*.

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

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

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

Virtual warehouse: A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers.



A recommended approach for data warehouse development

Types of OLAP Servers

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

Multidimensional OLAP (MOLAP) servers: These servers support multidimensional views of data through *array-based multidimensional storage engines*. They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to pre-computed summarized data. Notice that with multidimensional data stores, the storage utilization may be low if the data set is sparse. In such cases, sparse matrix compression techniques should be explored. Many MOLAP servers adopt a two-level storage representation to handle dense and sparse data sets: denser subcubes are identified and stored as array structures, whereas sparse subcubes employ compression technology for efficient storage utilization.

Hybrid OLAP (HOLAP) servers: The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP. For example, a HOLAP server may allow large volumes of detail data to be stored in a relational database, while aggregations are kept in a separate MOLAP store. The Microsoft SQL Server 2000 supports a hybrid OLAP server.

Specialized SQL servers: To meet the growing demand of OLAP processing in relational databases, some database system vendors implement specialized SQL servers that provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

An integrated OLAM and OLAP architecture