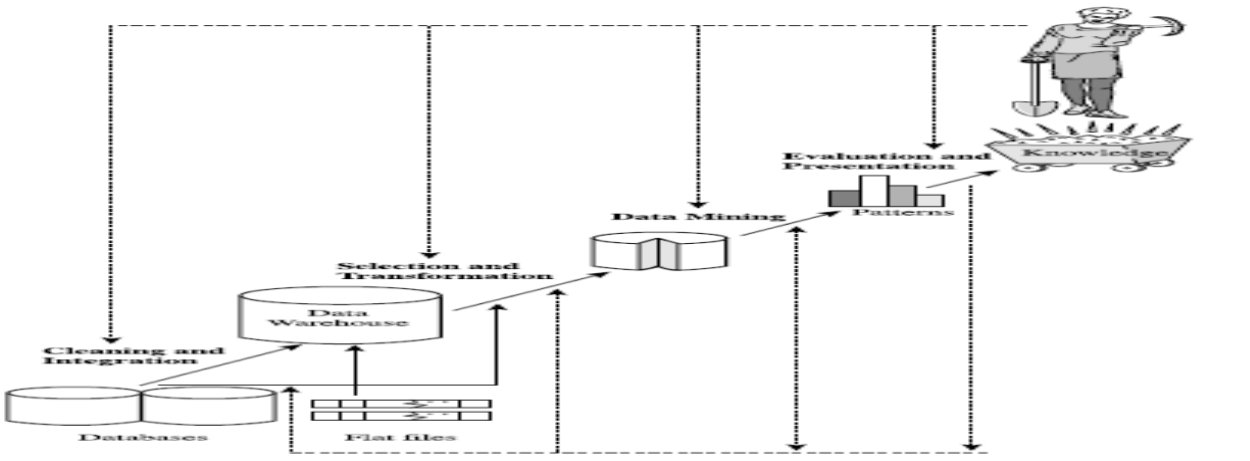
UNIT-III
INTRODUCTION

DATA MINING

Data Mining refers to **extracting** or **"mining"** knowledge from large amount of data.
It is used in terms like KDD (Knowledge Discovery from Data) which is an essential step in the process of analysis.
Data mining is an essential step in the process of knowledge discovery with the following steps

1. **Data cleaning** -- to remove noise and inconsistent data
2. **Data integration** -- where multiple data sources may be combined
3. **Data selection** -- where data relevant to the analysis task are retrieved from the database
4. **Data transformation** -- where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance
5. **Data mining**--an essential process where intelligent methods are applied in order to extract data patterns
6. **Pattern evaluation** -- to identify the truly interesting patterns representing knowledge based on some interestingness measures
7. **Knowledge presentation** --where visualization and knowledge representation techniques are used to present the mined knowledge to the user

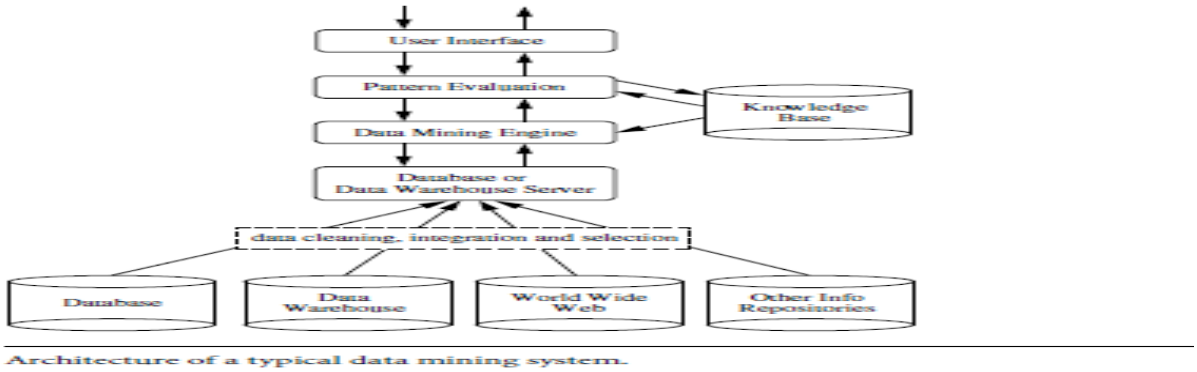


Data mining as a step in the process of knowledge discovery.

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.
- **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**: It interacts with the data mining modules so as to *focus* the search toward interesting 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, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results.



Architecture of a typical data mining system.

DATA MINING FUNCTIONALITIES

Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. Data mining tasks can be classified into two categories:

- 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 are as follows

1. **Concept/Class Description: Characterization and Discrimination:** Data can be associated with classes or concepts. Such descriptions of a class or a concept are called class/concept descriptions. These descriptions can be derived via
   (1) *data characterization*, by summarizing the data of the class under study (often called the target class) in general terms, or
   (2) *data discrimination*, by comparison of the target class with one or a set of comparative classes (often called the contrasting classes), or
   (3) both data characterization and discrimination.
   Data characterization is a summarization of the general characteristics or features of a target class of data. The output of data characterization can be presented in various forms. Examples include pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs. The resulting descriptions can also be presented as generalized relations or in rule form (called characteristic rules).

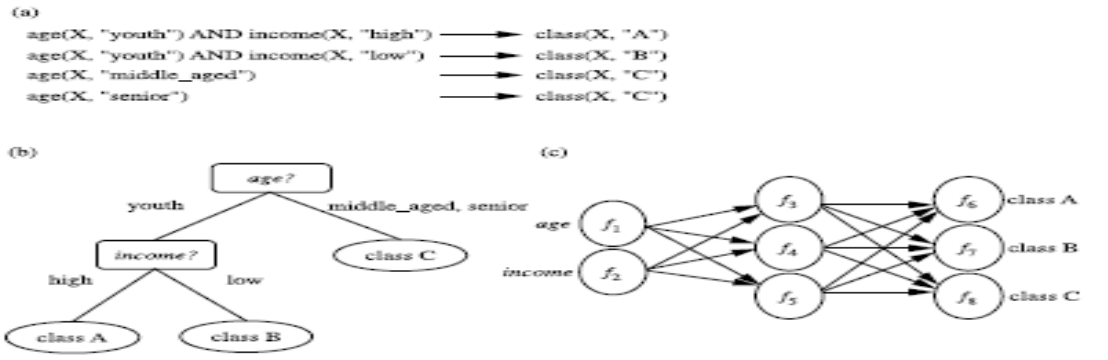
2. **Mining Frequent Patterns, Associations, and Correlations:** Frequent patterns are patterns that occur frequently in data. There are many kinds of frequent patterns, including itemsets, subsequences, and substructures.
   - A *frequent* ***itemset*** typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread.
   - A frequently occurring **subsequence**, such as the pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is a (*frequent*) *sequential pattern*.

- A **substructure** can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) ***structured pattern***.

Association is the two products are associated if they are dependent.
Correlation is a measure which calculates the dependency of the product truly dependent, positive or negative and independent.

3. **Classification and Prediction: Classification** is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model 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 (i.e., data objects whose class label is known).
   The derived model may be represented in various forms, such as *classification (IF-THEN) rules*, *decision trees*, *mathematical formulae*, or *neural networks.*

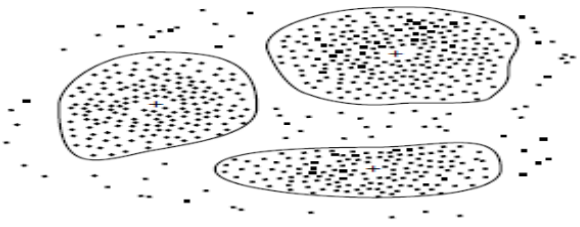
   A **decision tree** is a flow-chart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easily be converted to classification rules.
   A **neural network**, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units.



A classification model can be represented in various forms, such as (a) IF-THEN rules, (b) a decision tree, or a (c) neural network.

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.

4. **Cluster Analysis:** Clustering analyzes data objects without consulting a known class label. The objects are clustered or grouped based on the principle of *maximizing the intraclass similarity and minimizing the interclass similarity*. Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together.



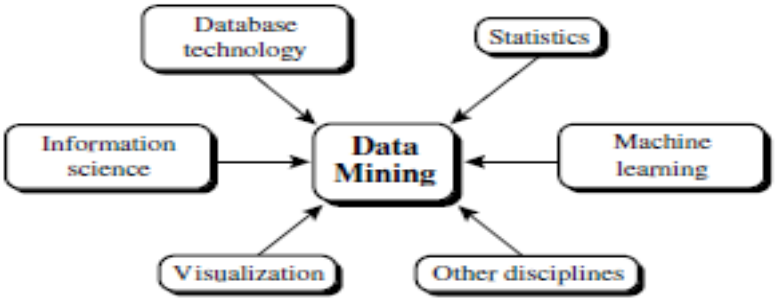
A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster "center" is marked with a "+".

5. **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. The analysis of outlier data is referred to as **outlier mining**.

6. **Evolution Analysis:** Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time.

CLASSIFICATION OF DATA MINING SYSTEMS

Data mining is an interdisciplinary field, the confluence of a set of disciplines, including database systems, statistics, machine learning, visualization, and information science. Depending on the kinds of data to be mined or on the given data mining application, the data mining system may also integrate techniques from spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, computer graphics, Web technology, economics, business, bioinformatics, or psychology. Data mining systems can be categorized according to various criteria, as follows:



- **Classification according to the *kinds of databases* mined**: A data mining system can be classified according to the kinds of databases mined. Database systems can be classified according to different criteria (such as data models, or the types of data or applications involved), each of which may require its own data mining technique. Data mining systems can therefore be classified accordingly.
- **Classification according to the *kinds of knowledge* mined**: Data mining systems can be categorized according to the kinds of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis.
- **Classification according to the *kinds of techniques* utilized**: Data mining systems can be categorized according to the underlying data mining techniques employed. These techniques can be described according to the degree of user interaction involved or the methods of data analysis employed.
- **Classification according to the *applications adapted***: Data mining systems can also be categorized according to the applications they adapt. For example, data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on.

MAJOR ISSUES IN DATA MINING

The major issues of the data mining are as follows

1. Mining methodology and user interaction issues

2. Performance issues
3. Issues relating to the diversity of database types

Mining methodology and user interaction issues: These reflect the kinds of knowledge mined the ability to mine knowledge at multiple granularities, the use of domain knowledge, ad hoc mining, and knowledge visualization.

- *Mining different kinds of knowledge in databases:* Different users can be interested in different kinds of knowledge, data mining should cover the functionalities of data characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis
- *Interactive mining of knowledge at multiple levels of abstraction:* It is difficult to know exactly what can be discovered within a database, the data mining process should be *interactive*
- *Incorporation of background knowledge:* Background knowledge, or information regarding the domain under study, may be used to guide the discovery process and allow discovered patterns to be expressed in concise terms and at different levels of abstraction. 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:* Relational query languages (such as SQL) allow users to pose ad hoc queries for data retrieval or a data ware house query language called DMQL.
- *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. This requires the system to adopt expressive knowledge representation techniques, such as trees, tables, rules, graphs, charts, crosstabs, matrices, or curves.
- *Handling noisy or incomplete data:* The data stored in a database may reflect noise, exceptional cases, or incomplete data objects. 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:* To guide the discovery process and reduce the search space in another active area of research.

Performance issues: These include efficiency, scalability, and parallelization of data mining algorithms.

- *Efficiency and scalability of data mining algorithms:* To effectively extract information from a huge amount of data in databases, data mining algorithms must be efficient and scalable. In other words, the running time of a data mining algorithm must be predictable and acceptable in large databases.
- *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.

Issues relating to the diversity of database types:

- *andling of relational and complex types of data:* Specific data mining systems should be constructed for mining specific kinds of data like text, spatial, transactional, relational etc.

- *Mining information from heterogeneous databases and global information systems:* Local- and wide-area computer networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases like Web mining, multimedia, sequence database, temporal database, WWW etc.

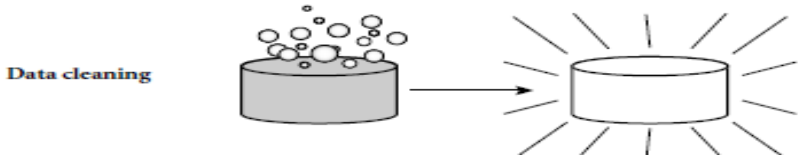
DATA PREPROCESSING

Data Preprocessing is an important issue for both data warehousing and data mining, as real-world data tend to be incomplete, noisy and inconsistent. Data preprocessing includes

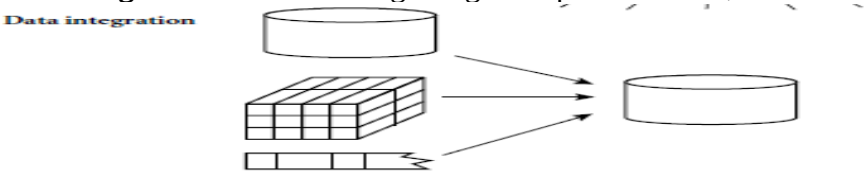
- Data cleaning
- Data integration
- Data transformation
- Data reduction

Need for preprocessing the data are as follows

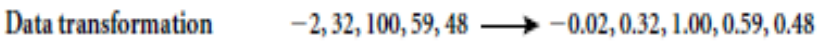
- **Data cleaning** routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.



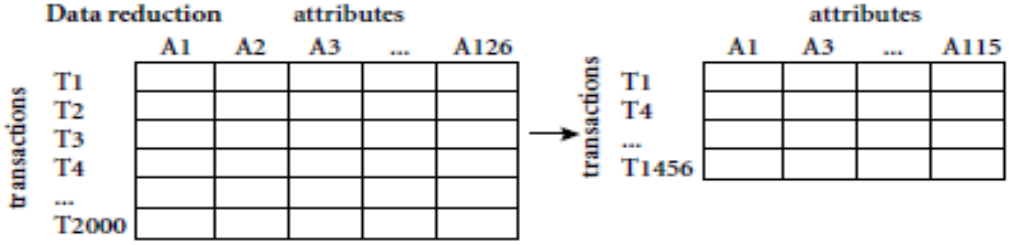
- **Data integration** involves integrating multiple databases, data cubes, or files.



- **Data transformation** involves normalization and aggregation, are additional data preprocessing procedures that would contribute toward the success of the mining process.



- **Data reduction** obtains a reduced representation of the data set that is much smaller in volume, yet produces the same (or almost the same) analytical results.



DATA CLEANING

Data cleaning or ***data cleansing*** routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

1. **Missing Values:** Many tuples have no recorded values for several attributes, for filling the missing values for attribute consider the following methods

- **Ignore the tuple**: Used when class label is missing. Not very effective. Especially poor when percentage of missing values per attribute varies considerably.
- **Fill in the missing values manually:** It is time consuming and may not be feasible for large data set.
- **Use a global constant to fill in the missing value:** Replace all missing attribute values by the same constant, such as a label like *"Unknown"* or -∞.
- **Use the attribute mean to fill in the missing value:** For example, suppose that the average income of *AllElectronics* customers is $56,000. Use this value to replace the missing value for *income*.
- **Use the attribute mean for all samples belonging to the same class as the given tuple**: For example, if classifying customers according to *credit risk*, replace the missing value with the average *income* value for customers in the same credit risk category as that of the given tuple.
- **Use the most probable value to fill in the missing value**: This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

2. **Noisy Data**: **Noise** is a random error or variance in a measured variable. Data smoothing techniques are as follows
   - **Binning**: Binning methods smooth a sorted data value by consulting its "neighborhood," i.e, the values around it. The sorted values are distributed into a number of "buckets," or *bins*. Because binning methods consult the neighborhood of values, they perform *local* smoothing.
     Example:

     Sorted data for *price* (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34

     The data for *price* are first sorted and then partitioned into ***equal-frequency*** bins of size 3 (i.e., each bin contains three values).

     Partition into (equal-frequency) bins:

     Bin 1: 4, 8, 15
     Bin 2: 21, 21, 24
     Bin 3: 25, 28, 34

     Smoothing by bin means, each value in a bin is replaced by the mean value of the bin.

     Smoothing by bin means:

     Bin 1: 9, 9, 9
     Bin 2: 22, 22, 22
     Bin 3: 29, 29, 29

     Smoothing by bin medians can be employed, in which each bin value is replaced by the bin median.
     Smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value.

     Smoothing by bin boundaries:

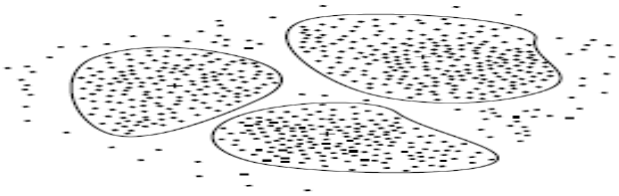
     Bin 1: 4, 4, 15
     Bin 2: 21, 21, 24
     Bin 3: 25, 25, 34

   - **Regression**: Data can be smoothed by fitting the data to a function.
     - *Linear regression* involves finding the "best" line to fit two attributes (or variables), so that one attribute can be used to predict the other.

- *Multiple linear regression* is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

- **Clustering**: Errors and redundancies may be detected by clustering. Here similar values are organized into groups.

A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster centroid is marked with a "+", representing the average point in space for that cluster. Outliers may be detected as values that fall outside of the sets of clusters.



3. **Data Cleaning as a Process:** The first step in data cleaning as a process is ***discrepancy detection***. The poorly designed data entry forms like human error in the data entry, deliberate errors, and data decay etc. This discrepancy can be proceeding by using **metadata** such as knowledge or data about data. **Field overloading** is another source of errors that typically results when developers squeeze new attribute definitions into unused (bit) portions of already defined attributes.

   The data should be examined by rules as follows

   - A unique rule says that each value of the given attribute must be different from all other values for that attribute.
   - A consecutive rule says that there can be no missing values between the lowest and highest values for the attribute, and that all values must also be unique (e.g., as in check numbers).
   - A null rule specifies the use of blanks, question marks, special characters, or other strings that may indicate the null condition (e.g., where a value for a given attribute is not available), and how such values should be handled.

   There are a number of different commercial tools that can aid in the step of discrepancy detection.

   - **Data scrubbing tools** use simple domain knowledge (e.g., knowledge of postal addresses, and spell-checking) to detect errors and make corrections in the data.
   - **Data auditing tools** find discrepancies by analyzing the data to discover rules and relationships, and detecting data that violate such conditions.
   - **Data migration tools** allow simple transformations to be specified, such as to replace the string *"gender"* by *"sex"*.
   - **ETL (extraction/transformation/loading) tools** allow users to specify transforms through a graphical user interface (GUI).

DATA INTEGRATION AND TRANSFORMATION

DATA INTEGRATION:

D*ata integration* combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. Numbers of issues are as follows

- Schema integration
- Object matching
- Redundancy

Equivalent real-world entities from multiple data sources be matched up is refereed as **entity identification problem.** For example, how can the data analyst or the computer be sure that *customer id* in one database and *cust number* in another refer to the same attribute.

- Metadata can be used to help avoid errors is referred as **schema integration.** Examples of metadata for each attribute include the name, meaning, data type, and range of values permitted for the attribute, and null rules for handling blank, zero, or null values.
- **Object Matching** is a crucial task in data integration systems.
- **Redundancy** is another important issue, where an attribute (such as *annual revenue*, for instance) may be redundant if it can be "derived" from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

Some redundancies can be detected by **correlation analysis.** Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For numerical attributes, we can evaluate the correlation between two attributes, *A* and *B*, by computing the **correlation coefficient.** This is

$$r_{A,B} = \frac{\sum_{i=1}^{N}(a_i - \bar{A})(b_i - \bar{B})}{N\sigma_A\sigma_B} = \frac{\sum_{i=1}^{N}(a_i b_i) - N\bar{A}\bar{B}}{N\sigma_A\sigma_B},$$

_____ Equation 1

Where N ---number of tuples

$\bar{A}\bar{B}$ ----mean values of A and B

$a_i b_i$ ---- Values of A and B

$\sigma_A\sigma_B$ ---- Standard deviation of A and B

r ---- co-relation co-efficient

$\Sigma(a_i b_i)$ ---- sum of the *AB* cross-product

Note that $-1 \le r_{A,B} \le +1$. If $r_{A,B}$ is greater than 0, then *A* and *B* are positively correlated, meaning that the values of *A* increase as the values of *B* increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other).

Note that correlation does not imply causality. That is, if *A* and *B* are correlated, this does not necessarily imply that *A* causes *B* or that *B* causes *A*.

For categorical (discrete) data, a correlation relationship between two attributes, *A* and *B*, can be discovered by a χ^2 (chi-square) test. Suppose *A* has c distinct values, namely *a*1,*a*2,…*ac*. *B* has *r* distinct values, namely *b*1,*b*2,…,*br*. The data tuples described by *A* and *B* can be shown as a **contingency table**, with the *c* values of *A* making up the columns and the *r* values of *B* making up the rows. Let (*Ai*,*Bj*) denote the event that attribute *A* takes on value *ai* and attribute *B* takes on value *bj*, that is, where (*A* = *ai*,*B* = *bj*). Each and every possible (*Ai*,*Bj*) joint event has its own cell (or slot) in the table. The χ^2 value is computed as:

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o_{ij} - e_{ij})^2}{e_{ij}},$$

_____ Equation 2

where *oij* is the *observed frequency* (i.e., actual count) of the joint event (*Ai*,*Bj*) and *eij* is the *expected frequency* of (*Ai*,*Bj*), which can be computed as

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{N},$$ _____ Equation 3

where N is the number of data tuples, $count(A=ai)$ is the number of tuples having value ai for A, and $count(B = bj)$ is the number of tuples having value bj for B.

Example: Correlation analysis of categorical attributes using χ^2

Suppose that a group of 1,500people was surveyed. The gender of each person was noted. Each person was polled as to whether their preferred type of reading material was fiction or nonfiction. Thus, we have two attributes, *gender* and *preferred reading*. The observed frequency (or count) of each possible joint event is summarized in the contingency table as shown below

A 2 X 2 contingency table for the data

	male	*female*	Total
fiction	250 (90)	200 (360)	450
non_fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

the numbers in parentheses are the expected frequencies (calculated based on the data distribution. By equation3 verify the expected frequencies for each cell. For example, the expected frequency for the cell (male, fiction) is.

$$e_{11} = \frac{count(male) \times count(fiction)}{N} = \frac{300 \times 450}{1500} = 90,$$

Using Equation 2 for χ^2 computation, get

$$\begin{aligned} \chi^2 &= \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} \\ &= 284.44 + 121.90 + 71.11 + 30.48 = 507.93. \end{aligned}$$

The test is based on a significance level, with (r-1)*(c-1) degrees of freedom. For this 2 X 2 table, the degrees of freedom are (2-1)(2-1) = 1. For 1 degree of freedom, the χ^2 values needed to reject the hypothesis at the 0.001 significance level is 10.828

DATA TRANSFORMATION:

*In **data transformation***, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

- **Smoothing**, which works to remove noise from the data. Such techniques include binning, regression, and clustering.
- **Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.
- **Generalization** of the data, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like *street*, can be generalized to higher-level concepts, like *city* or *country*.
- **Normalization**, where the attribute data are scaled so as to fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0.
- **Attribute construction** (or ***feature construction***), where new attributes are constructed and added from the given set of attributes to help the mining process.

There are many methods for data normalization. There are as follows

1. *min-max normalization*

2. *z-score normalization*
3. *normalization by decimal scaling*

1**. Min-max normalization:** It performs a linear transformation on the original data. Suppose that *minA* and *maxA* are the minimum and maximum values of an attribute, *A*. Min-max normalization maps a value, *v*, of *A* to v^1 in the range [*new_ minA, new_ maxA*] by computing

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original data range for *A*.

Example: Suppose that the minimum and maximum values for the attribute *income* are $12,000 and $98,000, respectively. To map *income* to the range [0.0,1.0]. By min-max normalization, a value of $73,600 for *income* is transformed to

$$\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716.$$

Where 0.716 < 1.

2**. z-score normalization (or *zero-mean normalization*):** In this the values for an attribute,*A*, are normalized based on the mean and standard deviation of *A*. A value, *v*, of *A* is normalized to v^1 by computing

$$v' = \frac{v - \bar{A}}{\sigma_A},$$

where *A* and σ_A are the mean and standard deviation, respectively, of attribute *A*. This method of normalization is useful when the actual minimum and maximum of attribute *A* are unknown, or when there are outliers that dominate the min-max normalization.

Example: Suppose that the mean and standard deviation of the values for the attribute *income* are $54,000 and $16,000, respectively. With z-score normalization, a value of $73,600 for *income* is transformed to

$$\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225.$$

3. **Normalization by decimal scaling:** This normalizes by moving the decimal point of values of attribute *A*. The number of decimal points moved depends on the maximum absolute value of *A*. A value, *v*, of *A* is normalized to v^1 by computing

$$v' = \frac{v}{10^j},$$

where *j* is the smallest integer such that $Max(|v^1|) < 1$.

Example: Suppose that the recorded values of *A* range from -986 to 917. The maximum absolute value of *A* is 986. To normalize by decimal scaling, therefore divide each value by 1,000 (i.e., *j* = 3) so that -986 normalize to -0.986 and 917 normalizes to 0.917.

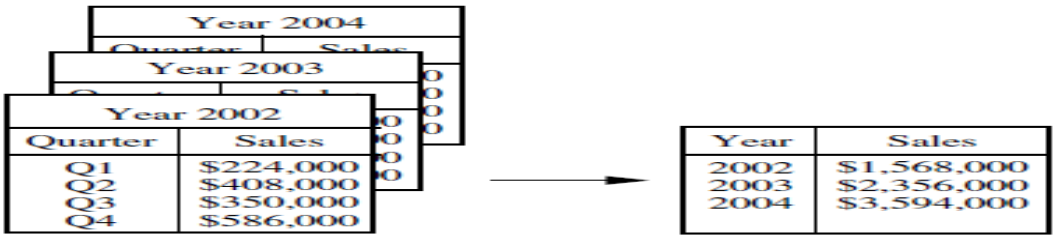
DATA REDUCTION

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. Strategies for data reduction include the following:

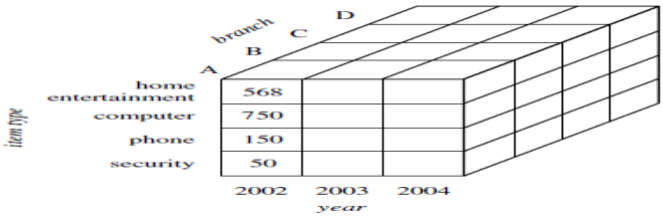
1. Data cube aggregation, where aggregation operations are applied to the data in the construction of a data cube.

2. Attribute subset selection, where irrelevant, weakly relevant or redundant attributes or dimensions may be detected and removed.
3. Dimensionality reduction, where encoding mechanisms are used to reduce the data set size.
4. Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models or nonparametric methods such as clustering, sampling, and the use of histograms.
5. Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Data discretization is a form of numerosity reduction that is very useful for the automatic generation of concept hierarchies. Discretization and concept hierarchy generation are powerful tools for data mining, in that they allow the mining of data at multiple levels of abstraction.

1. **Data Cube Aggregation:**Aggregation operations are applied to the data in the construction of a data cube. Consider the data for analysis which consists of sales per quarter year, but the customer is interested to see annual sales then the data can be aggregated so that the resulting data summarize the total sales per year rather per quarter.
Example: Sales data 2002 to 2004, on the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.



Data cube is as follows



Data cubes store multidimensional aggregated information. Data cubes provide fast access to precomputed, summarized data, thereby benefiting on-line analytical processing as well as data mining. The cube created at the lowest level of abstraction is referred to as the ***base cuboid***. A cube at the highest level of abstraction is the ***apex cuboid***. Data cubes created for varying levels of abstraction are often referred to as ***cuboids***, so that a data cube may instead refer to a ***lattice of cuboids*.**

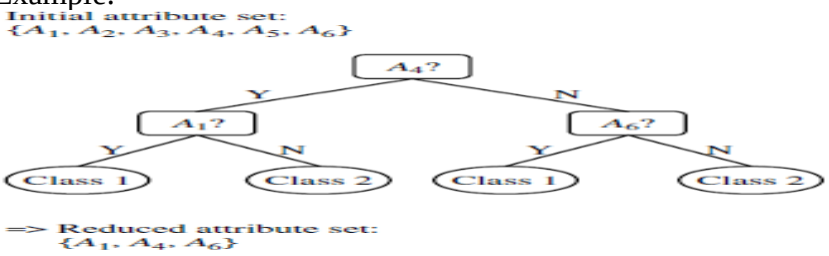
2. Attribute Subset Selection:
Attribute subset selection reduces the data set size by removing irrelevant or redundant attributes (or dimensions). To find a 'good' subset of the original attributes, consider the heuristic (Greedy) methods as follows

1. **Stepwise forward selection**: The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.
   Example: Initial attribute set : {*A1*, *A*2, *A*3, *A*4, *A*5, *A*6}
   Initial reduced set: {}
   => {A1}

=> {*A*1, *A*4}
=> Reduced attribute set: {*A*1, *A*4, *A*6}

2. **Stepwise backward elimination**: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
   Example: Initial attribute set: {*A*1, *A*2, *A*3, *A*4, *A*5, *A*6}
   => {*A*1, *A*3, *A*4, *A*5, *A*6}
   => {*A*1, *A*4, *A*5, *A*6}
   => Reduced attribute set: {*A*1, *A*4, *A*6}
3. **Combination of forward selection and backward elimination**: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
4. Decision tree induction: Decision tree induction constructs a flowchart like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction.

Example:



3. Dimensionality Reduction:

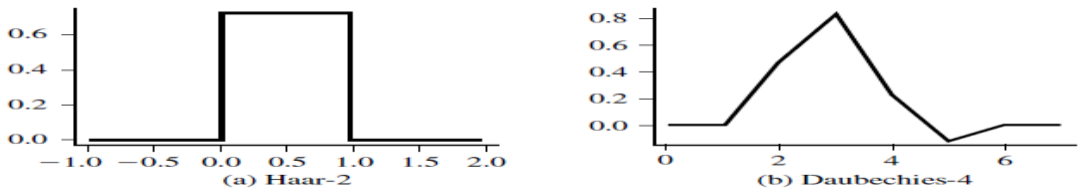
Data encoding or transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be *reconstructed* from the compressed data without any loss of information, the data reduction is called **lossless.** Reconstruct only an approximation of the original data, then the data reduction is called **lossy.** The two popular and effective methods of lossy dimensionality reduction:

- *wavelet transforms*
- *principal components analysis*.

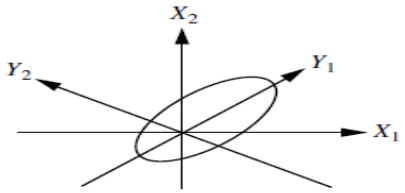
Wavelet Transforms: The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector ***X***, transforms it to a numerically different vector, ***X*^{1}**, of **wavelet coefficients**.

The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines. There is only one DFT, yet there are several families of DWTs. Popular wavelet transforms include the Haar-2, Daubechies-4, and Daubechies-6 transforms.

Below figure shows the examples of wavelet families is as follows



Principal Components Analysis: Suppose that the data to be reduced consist of tuples or data vectors described by *n* attributes or dimensions. Principal components analysis, or PCA (also called the Karhunen-Loeve, or K-L, method), searches for *k* *n*-dimensional orthogonal vectors that can best be used to represent the data, where $k \leq n$. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction.



Principal components analysis. Y_1 and Y_2 are the first two principal components for the given data.

.

4. Numerosity Reduction:

Reduce the data volume by choosing alternative, 'smaller' forms of data representation. These techniques may be parametric or nonparametric.

- For *parametric methods*, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data.
  Examples: Log-linear models, which estimate discrete multidimensional probability distributions.
- *Nonparametric methods* for storing reduced representations of the data include histograms, clustering, and sampling.

The numerosity reduction techniques are as follows

1. Regression and log-linear models
2. Histograms
3. Clustering
4. Sampling

1. Regression and log-linear models:

The data are modeled to fit a straight line is **linear regression**. For example, a random variable, *y* (called a *response variable*), can be modeled as a linear function of another random variable, *x* (called a *predictor variable*), with the equation

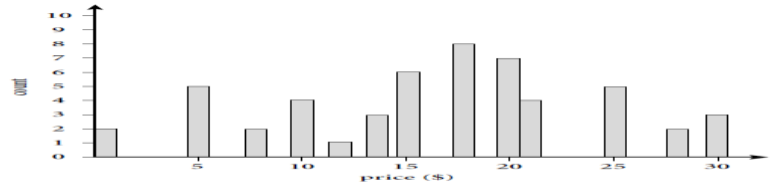
$$y = wx + b,$$

where the variance of *y* is assumed to be constant. In the context of data mining, *x* and *y* are numerical database attributes. The coefficients w and b are called regression coefficients specify the slope of the line and the *Y*-intercept, respectively.

Multiple linear regression is an extension of (simple) linear regression, which allows a response variable, *y*, to be modeled as a linear function of two or more predictor variables.

Log-linear models approximate discrete multidimensional probability distributions.

2. **Histograms:** A histogram for an attribute, *A*, partitions the data distribution of *A* into disjoint subsets, or ***buckets*.** If each bucket represents only a single attribute-value/frequency pair, the buckets are called ***singleton buckets***. **Multidimensional histograms** can capture dependencies between attributes.
   Example: The following data are a list of prices of commonly sold items at *AllElectronics*(rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5,8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20,20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.



A histogram for *price* using singleton buckets—each bucket represents one price-value/ frequency pair.

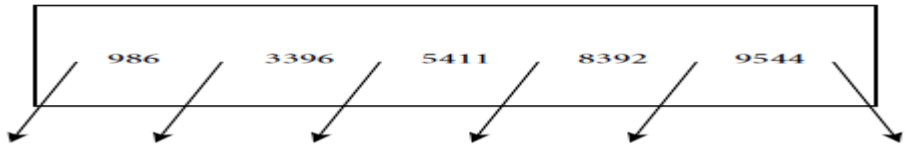
An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

The buckets determined and the attribute values partitioned using partitioning rules as follows

- **Equal-width**: The size of the bucket is same (Such as $10 for the buckets in Figure2).
- **Equal-frequency (or equidepth):** Each bucket contains roughly the same number of contiguous data samples.
- **V-Optimal**: The bucket weight is equal to the number of values in the bucket.
- **MaxDiff:** It considers the difference between each pair of adjacent values.

3. **Clustering**:

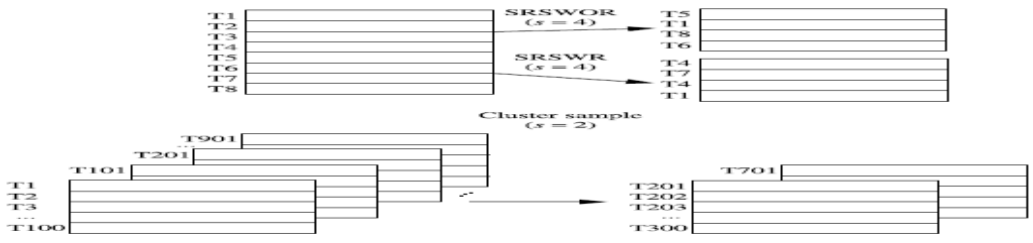
Clustering techniques consider data tuples as objects. They partition the objects into groups or *clusters*, so that objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based on a distance function. The "quality" of a cluster may be represented by its *diameter*, the maximum distance between any two objects in the cluster. ***Centroid distance*** is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "average object," or average point in space for the cluster).



The root of a B+-tree for a given set of data.

4. **Sampling:**

Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random sample (or subset) of the data.
Suppose that a large data set, *D*, contains *N* tuples. Let's look at the most common ways that we could sample *D* for data reduction, as illustrated in Figure
Sampling can be used for data reduction.



Stratified sample (according to *age*)

T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior

- **Simple random sample without replacement (SRSWOR) of size *s***: This is created by drawing *s* of the *N* tuples from*D* (*s* < *N*), where the probability of drawing any tuple in *D* is 1=*N*, that is, all tuples are equally likely to be sampled.
- **Simple random sample with replacement (SRSWR) of size *s***: This is similar to SRSWOR, except that each time a tuple is drawn from *D*, it is recorded and then *replaced*.
- **Cluster sample:** If the tuples in *D* are grouped into *M* mutually disjoint "clusters," then an SRS of *s* clusters can be obtained, where *s* < *M*.
- **Stratified sample**: If *D* is divided into mutually disjoint parts called *strata*, a stratified sample of *D* is generated by obtaining an SRS at each stratum.

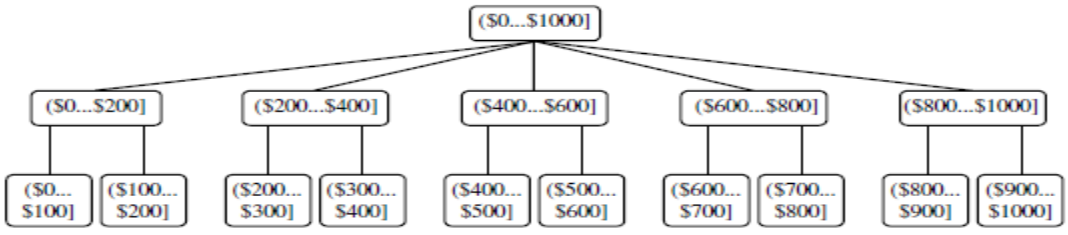
DATA DISCRETIZATION AND CONCEPT HIERARCHY GENERATION

Data discretization techniques can be used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values. Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up).

- If the discretization process uses class information, then it is ***supervised discretization*.** Otherwise, it is ***unsupervised*.**
- If the process starts by first finding one or a few points (called *split points* or *cut points*) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called ***top-down discretization* or *splitting***.
- ***Bottom-up discretization* or *merging***, which starts by considering all of the continuous values as potential split-points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals.

Here it uses three types of attributes as nominal (values from an unordered set), ordinal(values from an ordered set) and continuous (real numbers).

An example of a concept hierarchy for the attribute *price* is given in Figure. More than one concept hierarchy can be defined for the same attribute in order to accommodate the needs of various users.



A concept hierarchy for the attribute *price*, where an interval ($X . . . $Y] denotes the range from $X (exclusive) to $Y (inclusive).

The generation of concept hierarchies for numerical and categorical data is as follows

1. **Discretization and Concept Hierarchy Generation for Numerical Data:**
Concept hierarchies for numerical attributes can be constructed automatically based on data discretization and the methods are as follows:

- *Binning*
- H*istogram analysis*
- *Entropy-based discretization*
- χ^2*-merging*
- C*luster analysis* and
- *Discretization by intuitive partitioning.*

Binning: Binning is a top-down splitting technique based on a specified number of bins.

Histogram Analysis: It is an unsupervised discretization technique because it does not use class information. Histograms partition the values for an attribute, *A*, into disjoint ranges called ***buckets*.**
The rules for defining histograms are an ***equal-width*** **histogram**, for example, the values are partitioned into equal-sized partitions or ranges and an ***equalfrequency*** histogram, the values are partitioned so that, ideally, each partition contains
the same number of data tuples. The ***minimum interval size*** can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level.

Entropy-Based Discretization: ***Entropy*** is one of the most commonly used discretization measures. Entropy-based discretization is a supervised, top-down splitting technique. It explores class distribution information in its calculation and determination of split-points (data values for partitioning an attribute range).
Let *D* consist of data tuples defined by a set of attributes and a class-label attribute. The class-label attribute provides the class information per tuple. The basic method for entropy-based discretization of an attribute *A* within the set is as follows:

$$Info_A(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} Entropy(D_2),$$

$$Entropy(D_1) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Interval Merging by χ^2Analysis:
ChiMerge is a χ^2-based discretization method. The discretization methods have to employ a top-down, splitting strategy. This contrasts with ChiMerge, which employs a bottom-up approach by finding the best neighboring intervals and then merging these to form larger intervals, recursively. The method is supervised in that it uses class information. The basic notion is that for accurate discretization, the relative class frequencies should be fairly consistent within an interval. Therefore, if two adjacent intervals have a very similar distribution of classes, then the intervals can be merged. Otherwise, they should remain separate.
ChiMerge proceeds as follows.

- Initially, each distinct value of a numerical attribute *A* is considered to be one interval.
- χ^2 tests are performed for every pair of adjacent intervals.

- Adjacent intervals with the least χ^2values are merged together, because low χ^2values for a pair indicate similar class distributions. This merging process proceeds recursively until a predefined stopping criterion is met.

Cluster Analysis: Cluster analysis is a popular data discretization method. A clustering algorithm can be applied to discretize a numerical attribute, *A,* by partitioning the values of *A* into clusters or groups. Clustering takes the distribution of *A* into consideration, as well as the closeness of data points, and therefore is able to produce high-quality discretization results.

Discretization by Intuitive Partitioning: Above discretization methods are useful in the generation of numerical hierarchies; many users would like to see numerical ranges partitioned into relatively uniform, easy-to-read intervals that appear intuitive or "natural." For example, annual salaries broken into ranges like ($50,000, $60,000] are often more desirable than ranges like ($51,263.98, $60,872.34], obtained by, say, some sophisticated clustering analysis.

$$\frac{\text{High - Low}}{\text{-Low}}$$

$$= \frac{2000000 - (-1000000)}{-(-1000000)} = 3$$

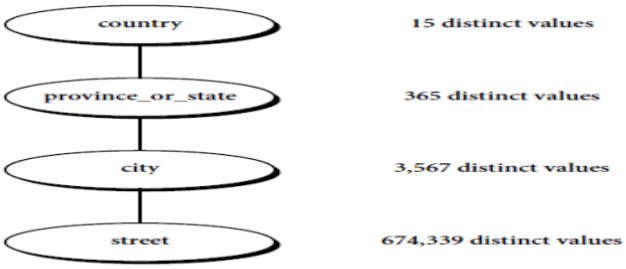
Partition the range into three equal intervals as
($1000000….$0)($0….$1000000)($10000000……$2000000)

2. Concept Hierarchy Generation for Categorical Data:
Categorical data are discrete data. Categorical attributes have a finite (but possibly large) number of distinct values, with no ordering among the values.
Examples include *geographic location, job category,* and *item type.*
There are several methods for the generation of concept hierarchies for categorical data as follows

- **Specification of a partial ordering of attributes explicitly at the schema level by users or experts**: Concept hierarchies for categorical attributes or dimensions typically involve a group of attributes. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level. For example, a relational database or a dimension *location* of a data warehouse may contain the following group of attributes: *street, city, province or state,* and *country*. A hierarchy can be defined by specifying the total ordering among these attributes at the schema level, such as *street < city < province or state < country*.
- **Specification of a portion of a hierarchy by explicit data grouping:** This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it is unrealistic to define an entire concept hierarchy by explicit value enumeration.
  For example, after specifying that *province* and *country* form a hierarchy at the schema level, a user could define some intermediate levels manually, such as "{*Alberta, Saskatchewan, Manitoba*} *C prairies Canada*" and "{*British Columbia, prairies Canada}* *C Western Canada*".
- **Specification of a *set of attributes*, but not of their partial ordering:** A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly state their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy.

Without knowledge of data semantics, a hierarchical ordering for an arbitrary set of categorical attributes can be found as follows

Consider the following observation that since higher-level concepts generally cover several subordinate lower-level concepts, an attribute defining a high concept level (e.g., *country*) will usually contain a smaller number of distinct values than an attribute defining a lower concept level (e.g., *street*).



Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

Based on this observation, a concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set.

- **Specification of only a partial set of attributes:** Sometimes a user can be sloppy when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in the hierarchy specification.