`Unit-1: Part-1:

Data Mining:

Data mining is a way of using special computer programs to look through a lot of information at once, and find patterns that might be useful. It can help us make predictions about what might happen in the future based on what has happened in the past. But sometimes it's hard to find useful information because there's just so much of it! And sometimes the information is different from what we're used to, so we have to use different methods to analyze it.

Data Mining and Knowledge Discovery:

Data mining is a way of finding useful information from large amounts of data. It's part of a bigger process called knowledge discovery in databases, which is about turning raw data into useful knowledge.

To do this, data goes through a bunch of steps. First, it's preprocessed - this means cleaning it up and making sure it's in a format that the computer can understand. Then, data mining algorithms are used to find patterns and useful information. Finally, the results are postprocessed - this means summarizing the findings and presenting them in a useful way.

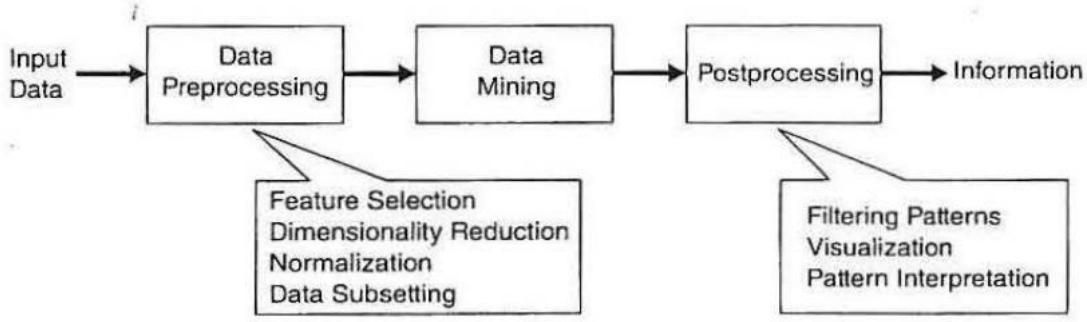


Figure 1.1. The process of knowledge discovery in databases (KDD).

Data can be stored in different formats like spreadsheets, files or tables and it can be in one place or spread out in different locations. Before analyzing the data, it needs to be prepared by a process called pre-processing.

Pre-processing helps to make sure that the data is in a suitable format for further analysis. This involves different steps like combining data from different sources, removing errors or duplicate entries, and selecting only the relevant parts of the data that are needed for the analysis.

Data preprocessing can take a lot of time because there are many ways data can be collected and stored, and it needs to be cleaned and organized before it can be analyzed properly.

Motivating Challenges:

1)Scalability:

Because of advances in data generation and collection, data sets with sizes of gigabytes, terabytes, or even petabytes are becoming common. lf data mining algorithms are to handle these massive data sets, then they must be scalable. Scalability can also be improved by using sampling or developing parallel and distributed algorithms.

2) High Dimensionality:

It is now common to encounter data sets with hundreds or thousands of attributes instead of the handful common a few

decades ago. Data sets with temporal or spatial components also tend to have high dimensionality.

For example, consider a data set that contains measurements of temperature at various locations. If the temperature measurements are taken repeatedly for an extended period, the number of dimensions (features) increases.

Traditional data analysis techniques that were developed for low-dimensional data often do not work well for such highdimensional data.

3) Heterogeneous and Complex Data:

Traditional data analysis methods often deal with data sets containing attributes of the same type, either continuous or categorical. As the role of data mining in business, science, medicine, and other fields has grown, so has the need for techniques that can handle heterogeneous attributes.

Examples of such non-traditional types of data include collections of Web pages containing semi-structured text and hyperlinks; DNA data and climate data that consists of time series measurements.

4) Data Ownership and Distribution:

Sometimes, the data needed for an analysis is not stored in one location or owned by one organization. Instead, the data is geographically distributed among resources belonging to multiple entities. This requires the development of distributed data mining techniques.

5) Non-traditional Analysis:

When people use statistics to analyze data, they usually start by making a guess about what they think might be happening - this is called a hypothesis. Then they design an experiment to gather data and see if their hypothesis is true or not.

But this process can take a long time and it's difficult when there are many hypotheses to test. Nowadays, people need to analyze a lot of data and test thousands of hypotheses.

To help with this, special computer programs called "data mining techniques" have been developed. These programs can help automate the process of coming up with hypotheses and testing them, which makes it easier and faster to analyze a lot of data.

The origins of Data Mining:

To deal with the challenges of analyzing large and diverse data sets, researchers from different fields started working together to create better tools. This led to the development of data mining, which uses ideas from different fields like statistics, artificial intelligence, and machine learning.

Data mining also borrows ideas from other fields like optimization, evolutionary computing, and information retrieval. To help with the storage and processing of large data sets, techniques from database systems and high-performance computing are used. When the data is spread out in different locations, distributed techniques are needed to bring it all together.

By combining ideas from different fields, data mining has become an effective way to analyze complex data sets and discover useful information.

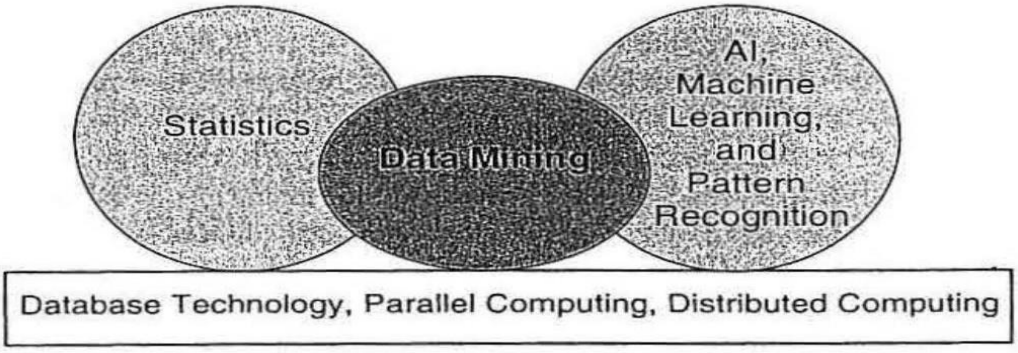


Figure 1.2. Data mining as a confluence of many disciplines.

Data Mining Tasks:

Data mining tasks are generally divided into two major categories:

1) Predictive tasks:

The objective of these tasks is to predict the value of a particular attribute based on the values of other attributes. The attribute to be predicted is commonly known as the target or dependent variable, while the attributes used for making the prediction are known as the explanatory or independent variables.

2) Descriptive tasks:

Here, the objective is to derive patterns (correlations, trends, clusters, trajectories, and anomalies) that summarize the underlying relationships in data. Descriptive data mining tasks are often exploratory in nature and frequently require postprocessing techniques to validate and explain the results.

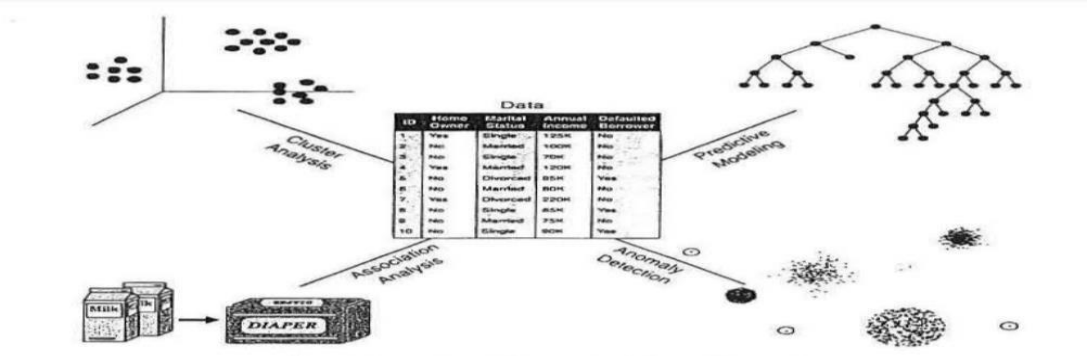


Figure 1.3. Four of the core data mining tasks.

3)Predictive modeling:

It refers to the task of building a model for the target variable as a function of the explanatory variables. There are two types of predictive modeling tasks: classification, which is used for discrete target variables, and regression, which is used for continuous target variables.Ex: Predicting the Type of a Flower

4)Association analysis:

It is used to discover patterns that describe strongly associated features in the data. The discovered patterns are typically represented in the form of implication rules or feature subsets.Ex: Market Basket Analysis

5)Cluster analysis:

It seeks to find groups of closely related observations so that observations that belong to the same cluster are more similar to each other.Ex: Document Clustering

6)Anomaly detection:

It is the task of identifying observations whose characteristicsare significantly different from the rest of the data. Such observations are known as anomalies or outliers. The goal of an anomaly detection algorithm is to discover the real anomalies and avoid falsely labeling normal objects as anomalous. Ex: Credit CardFraud Detection

Unit-1: Part-2:

What is data:

- Collection of ***data objects*** and their ***attributes***
- An ***attribute*** is a property or characteristic of an object
    - Examples: eye color of a person, temperature, etc.

- Attribute is also known as variable, field, characteristic, dimension, or feature

- A collection of attributes describe an ***object***
  - Object is also known as record, point, case, sample, entity, or instance.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Attribute values:

- **Attribute values** are numbers or symbols assigned to an attribute for a particular object
- Distinction between attributes and attribute values
  - Same attribute can be mapped to different attribute values
    - Example: height can be measured in feet or meters
  - Different attributes can be mapped to the same set of values
    - Example: Attribute values for ID and age are integers
  - But properties of attribute can be different than the properties of the values used to represent the attribute

Measurement of Length :

- The way you measure an attribute may not match the attributes properties.

Types of Attributes :

- There are different types of attributes
  - Nominal
    - Examples: ID numbers, eye color, zip codes
  - Ordinal
    - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height {tall, medium, short}
  - Interval
    - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
  - Ratio
    - Examples: temperature in Kelvin, length, counts, elapsed time (e.g., time to run a race)

Properties of Attribute Values:

- The type of an attribute depends on which of the following properties/operations it possesses:
  - Distinctness: = ≠
  - Order: < >
  - Differences are meaningful : + -
  - Ratios are meaningful * /
  - Nominal attribute: distinctness

- Ordinal attribute: distinctness & order
- Interval attribute: distinctness, order & meaningful differences
- Ratio attribute: all 4 properties/operations

	Attribute Type	Description	Examples	Operations
Categorical Qualitative	Nominal	Nominal attribute values only distinguish. (=, ≠)	zip codes, employee ID numbers, eye color, sex: {*male, female*}	mode, entropy, contingency correlation, $\chi 2$ test
	Ordinal	Ordinal attribute values also order objects. (<, >)	hardness of minerals, {*good, better, best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric Quantitative	Interval	For interval attributes, differences between values are meaningful. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
	Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation

Discrete and Continuous Attributes :

- **Discrete Attribute**
  - Has only a finite or countably infinite set of values
  - Examples: zip codes, counts, or the set of words in a collection of documents
  - Often represented as integer variables.
  - Note: binary attributes are a special case of discrete attributes
- **Continuous Attribute**
  - Has real numbers as attribute values
  - Examples: temperature, height, or weight.

- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.

Assymetric attributes:

- Only presence (a non-zero attribute value) is regarded as important
    - Words present in documents
    - Items present in customer transactions
- If we met a friend in the grocery store would we ever say the following?

  "I see our purchases are very similar since we didn't buy most of the same things."

Characteristics of data:

- Dimensionality (number of attributes)
    - High dimensional data brings a number of challenges
- Sparsity
    - Only presence counts
- Resolution
    - Patterns depend on the scale
- Size
    - Type of analysis may depend on size of data

Types of datasets:

Record

- Data Matrix
- Document Data
- Transaction Data

- Graph

- World Wide Web
- Molecular Structures

- Ordered
  - Spatial Data
  - Temporal Data
  - Sequential Data
  - Genetic Sequence Data

Record data:

- Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Data matrix:

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

- Such a data set can be represented by an *m* by *n* matrix, where there are *m* rows, one for each object, and *n* columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Document data:

- Each document becomes a 'term' vector
  - Each term is a component (attribute) of the vector
  - The value of each component is the number of times the corresponding term occurs in the document.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

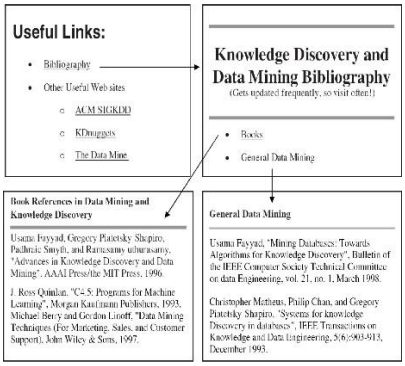
Transaction data:

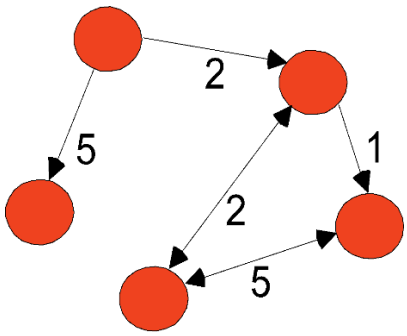
- A special type of data, where
  - Each transaction involves a set of items.
  - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

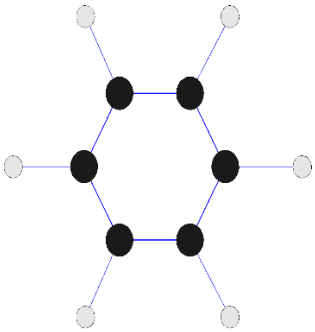
- Can represent transaction data as record data

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Graph data:







Ordered data:

(A B) (D) (C E)
(B D) (C) (E)
(C D) (B) (A E)

GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG

Data Quality:

Data quality refers to how reliable and accurate the data is. In the real world, there are many situations where the data is incomplete, inconsistent, or contains errors. This is a common problem in big databases.

Data mining is a process that helps to discover useful patterns and insights from data. However, this process can be affected by poor data quality. To overcome this problem, data mining focuses on detecting and correcting data quality issues. The first step in this process is called data cleaning.

Data cleaning involves identifying and correcting errors in the data. For example, if there are missing values in a dataset, data cleaning can involve filling in those missing values based on other information in the dataset. By cleaning the data, we can ensure that the data mining algorithms are more accurate and can provide more meaningful insights.

Measurement and Data Collection Issues:

Measurement and Data Collection Errors:

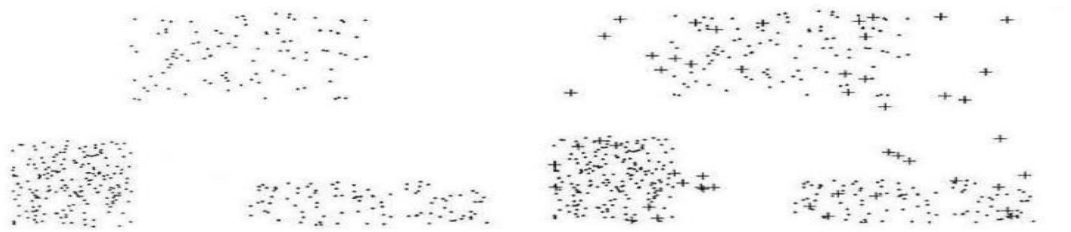
The term measureme nt error refers to any problem resulting from the measurement process. A common problem is that the value recorded differs from the true value to some extent. The term data collection error refers to errors such as omitting data objects or attri bute values, or inappropriately including a data object.

Noise and Artifacts Noise:

It is the random component of a measurement error. It may involve the distortion of a value or the addition of spurious objects.

(a) Time series. (b) Time series with noise.

Figure 2.5. Noise in a time series context.



(a) Three groups of points. (b) With noise points (+) added.

Figure 2.6. Noise in a spatial context.

Precision, Bias, and Accuracy:

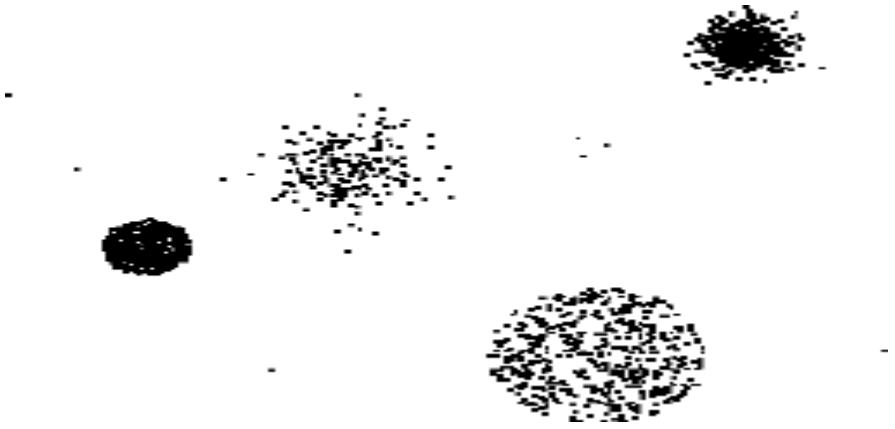
Precision: The closeness of repeated measurements of the same quantity to one another.

Bias: A systematic variation of measurements from the quantity being measured.

Accuracy: The closeness of measurements to t.he true value of the quantity being measured.

Outliers:

Outliers are either data objects that, in some sense, have characteristics that are different from most of the other data objects in the data set, or values of an attribute that are unusual with re spect to the typical values for that attribute.



Missing Values:

It is not unusual for an object to be missing one or more attribute values. In some cases, the information was not collected. The techniques used for dealing missing data:

Eliminate Data Objects or Attributes:

A simple and effective strategy is to eliminate objects with missing values.

Estimate Missing Values:

Sometimes missing data can be reliably estimatedthe missing values can be estimated by using the remai ing values.

Ignore the Missing Value during Analysis:

Many data mining approaches can be modified to ignoremissing values. lf one or both objects of a pair have missing values for some attributes, then the similari ty can be calculated by using only the attributes that do not have missing values.

Inconsistent Values:

Data can contain inconsistent values. Some types of inconsistences are easy to detect and some are not. Once an inconsistency has been detected, it is sometimes possible to correct the data.

Duplicate Data:

A data set may include data objects that are duplicates, or almost duplicates, of one another.

Data Preprocessing:

Data preprocessing is like getting the ingredients ready before cooking a meal. It involves preparing the data in a way that makes it easier for a computer program to understand and analyze. This can involve different techniques like cleaning up the data, removing errors, formatting it in a certain way, and more. The end goal is to make the data as clear and useful as possible for further analysis..

- Aggregation
- Sampling

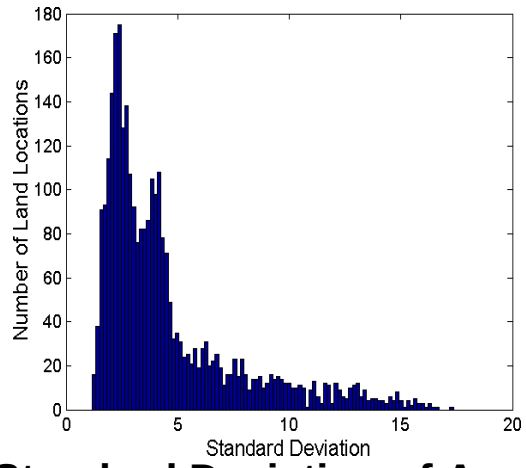
- Dimensionality reduction
- Feature subset selection
- Feature creation
- Discretization and binarization
- Attribute transformation

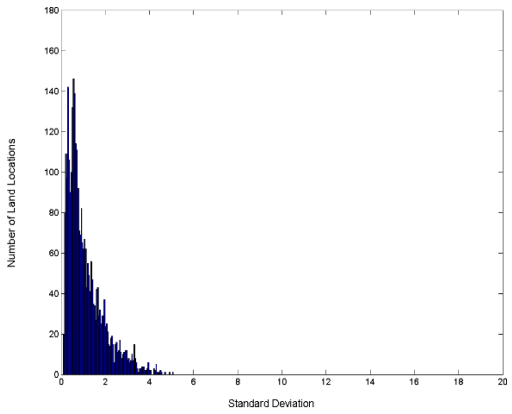
Aggregation:

Combining two or more attributes (or objects) into a single attribute (or object)

Purpose:

- Data reduction - reduce the number of attributes or objects
- Change of scale
  - Cities aggregated into regions, states, countries, etc.
  - Days aggregated into weeks, months, or years
- More “stable” data - aggregated data tends to have less variability.





Standard Deviation of Average Monthly Precipitation
Standard Deviation of Average Yearly Precipitation

Sampling:

Sampling is the main technique employed for data reduction.It is often used for both the preliminary investigation of the data and the final data analysis.

Statisticians often sample because obtaining the entire set of data of interest is too expensive or time consuming.

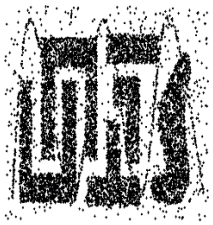
Sampling is typically used in data mining because processing the entire set of data of interest is too expensive or time consuming.

- The key principle for effective sampling is the following:

  Using a sample will work almost as well as using the entire data set,if the sample is representative

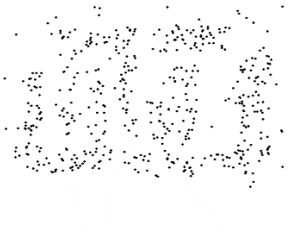
  A sample is representative if it has approximately the same properties (of interest) as the original set of data

Sample size:



8000 points

5000 points



500 points

Types of sampling:

- **Simple Random Sampling**
  - There is an equal probability of selecting any particular item
  - **Sampling without replacement**
    - As each item is selected, it is removed from the population
  - **Sampling with replacement**

- Objects are not removed from the population as they are selected for the sample.
- In sampling with replacement, the same object can be picked up more than once

- **Stratified sampling:**
  - Split the data into several partitions; then draw random samples from each partition.

Dimentionality reduction:

- Avoid curse of dimensionality
- Reduce amount of time and memory required by data mining algorithms
- Allow data to be more easily visualized
- May help to eliminate irrelevant features or reduce noise.

- Techniques
  - Principal Components Analysis (PCA)
  - Singular Value Decomposition
  - Others: supervised and non-linear techniques

Cursue of dimentionality:

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies.
- Definitions of density and distance between points, which are critical for clustering and outlier detection, become less meaningful.
- Another way to reduce dimensionality of data.

Feature Subset Selection:

- Redundant features
  - Duplicate much or all of the information contained in one or more other attributes
  - Example: purchase price of a product and the amount of sales tax paid

- Irrelevant features
    - Contain no information that is useful for the data mining task at hand
    - Example: students' ID is often irrelevant to the task of predicting students' GPA
- Many techniques developed, especially for classification.

Feature Creation:

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes
- Three general methodologies:
    - Feature extraction
        - Example: extracting edges from images
    - Feature construction
        - Example: dividing mass by volume to get density
    - Mapping data to new space

Example: Fourier and wavelet analysis

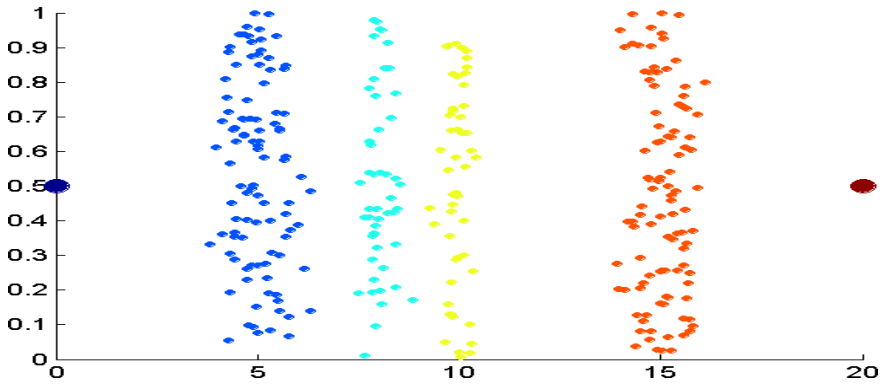
Discretization and binarization:

Discretization:

Discretization is the process of converting a continuous attribute into an ordinal attribute.

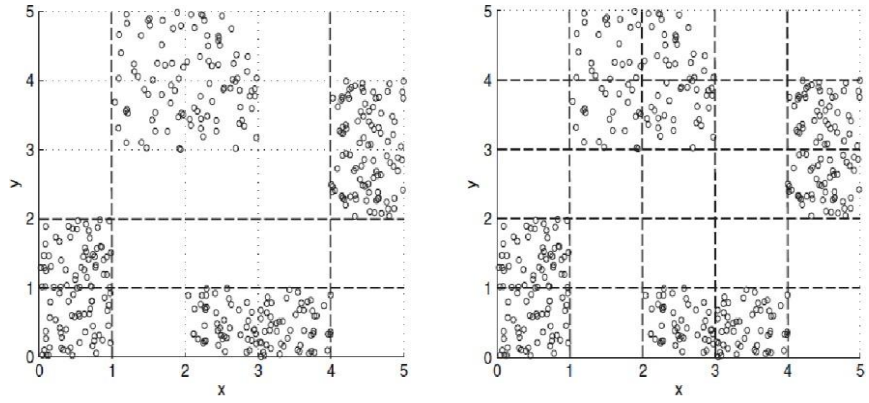
- A potentially infinite number of values are mapped into a small number of categories
- Discretization is used in both unsupervised and supervised settings

- **Unsupervised Discretization**



Data consists of four groups of points and two outliers. Data is one-dimensional, but a random y component is added to reduce overlap.

Supervised Discretization:



(a) Three intervals (b) Five intervals

Figure 2.14. Discretizing x and y attributes for four groups (classes) of points.

Binarization:

It maps a continuous or categorical attribute into one or more binary variables.

Table 2.6. Conversion of a categorical attribute to five asymmetric binary attributes.

Categorical Value	Integer Value	x_1	x_2	x_3	x_4	x_5
awful	0	1	0	0	0	0
poor	1	0	1	0	0	0
OK	2	0	0	1	0	0
good	3	0	0	0	1	0
great	4	0	0	0	0	1

Attribute transform:

- An attribute transform is a function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
    - Simple functions: x^k, log(x), e^x, |x|
    - Normalization
        - Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
        - Take out unwanted, common signal, e.g., seasonality
    - In statistics, standardization refers to subtracting off the means and dividing by the standard deviation

Unit-1:Part-3:

MEASURES OF SIMILARITY AND DISSIMILARITY:

The **similarity** between two objects is a numerical measure of the degree to which the two objects are alike.Similarities are higher for pairs of objects that are more alike. Similarities are usually non-negative and are often between 0 (no similarity) and 1 (complete similarity).

The **dissimilarity** between two objects is a numerical measure of the degree to which the two objects are different. Dissimilarities are lower for more similar pairs of objects. Dissimilarities sometimes fall in the interval [0, I], but it is also common for them to range from 0 to oo.

Similarity and Dissimilarity between Simple Attributes:

The proximity of objects with a number of attributes is typically defined by combining the proximities of individual attributes, and thus, we first discuss proximity between objects having a single attribute.

Table 2.7. Similarity and dissimilarity for simple attributes

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = \lvert x - y \rvert / (n-1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = \lvert x - y \rvert$	$s = -d$, $s = \frac{1}{1+d}$, $s = e^{-d}$, $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Dissimilarities between Data Objects:

Distances:

Equation is generalized by the Minkowski distance metric:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r},$$

where r is a parameter. The following are the three most common examples of Minkowski distances.

r = 1 City block (Manhattan) distance

r = 2. Euclidean distance

r = oo. Supremum distance. This is the maximum difference between any attribute of the objects.

Distances, such as the Euclidean distance, have some well-known properties. If d(x, y) is the distance between two points, x and y, then the following properties hold.

1. Positivity

   (a) d(x,x) >=0 for all x and y,

   (b) d(x,y) = 0 only if x = y.

2. Symmetry

   d(x,y) = d(y,x) for all x andy.

3. Triangle Inequality

   d(x, z) <=d{x, y) + d(y , z) for all points x, y, and z

Ex:

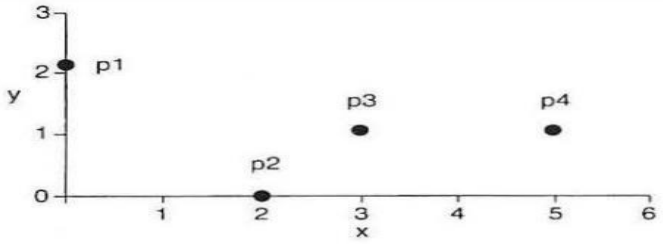


Figure 2.15. Four two-dimensional points.

Table 2.8. x and y coordinates of four points.

point	x coordinate	y coordinate
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Table 2.9. Euclidean distance matrix for Table 2.8.

	p1	p2	p3	p4
p1	0.0	2.8	3.2	5.1
p2	2.8	0.0	1.4	3.2
p3	3.2	1.4	0.0	2.0
p4	5.1	3.2	2.0	0.0

Table 2.10. L_1 distance matrix for Table 2.8.

L_1	p1	p2	p3	p4
p1	0.0	4.0	4.0	6.0
p2	4.0	0.0	2.0	4.0
p3	4.0	2.0	0.0	2.0
p4	6.0	4.0	2.0	0.0

Table 2.11. L_∞ distance matrix for Table 2.8.

L_∞	p1	p2	p3	p4
p1	0.0	2.0	3.0	5.0
p2	2.0	0.0	1.0	3.0
p3	3.0	1.0	0.0	2.0
p4	5.0	3.0	2.0	0.0

Similarities between Data Objects:

For similarities, the triangle inequality (or the analogous property) typically does not hold, but symmetry and positivity typically do.

To be explicit, if s(x, y) is the similarity between points x and y , then the typical properties of similarities are the following:

1. s(x,y) = 1 only if x = y. (0 <=S s <=S 1)
2. s(x,y) = s(y,x) for all x and y. (Symmetry)

Examples of Proximity Measures:

provides specific examples of some similarity and dissimilarity measures.

Similarity Measures for Binary Data:

Similarity measures between objects that contain only binary attributes are called similarity coefficients, and typically have values between 0 and 1. A value of 1 indicates that the two objects are completely similar, while a value of 0 indicates that the objects are not at all similar.

Let x and y be two objects that consist of n binary attributes. The comparison of two such objects, i.e., two binary vectors, leads to the following four quantities (frequencies):

f oo = the number of attributes where x is 0 and y is 0

f01 = the number of attributes where x is 0 and y is 1

f1O = the number of attributes where x is 1 and y is 0

f11 = the number of attributes where x is 1 and y is 1

Simple Matching Coefficient One commonly used similarity coefficient is the simple matching coefficient (SMC) , which is defined as :

$$SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}.$$

Jaccard Coefficient Suppose that x and y are data objects that represent two rows (two transactions) of a transaction matrix.

$\mathbf{x} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$
$\mathbf{y} = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$

$f_{01} = 2$ the number of attributes where x was 0 and y was 1
$f_{10} = 1$ the number of attributes where x was 1 and y was 0
$f_{00} = 7$ the number of attributes where x was 0 and y was 0
$f_{11} = 0$ the number of attributes where x was 1 and y was 1

$$SMC = \frac{f_{11}+f_{00}}{f_{01}+f_{10}+f_{11}+f_{00}} = \frac{0+7}{2+1+0+7} = 0.7$$

$$J = \frac{f_{11}}{f_{01}+f_{10}+f_{11}} = \frac{0}{2+1+0} = 0$$

Cosine Similarity:

Documents are often represented as vectors, where each attribute represents the frequency with which a particular term (word) occurs in the document. Even though documents have thousands or tens of thousands of attributes (terms), each document is sparse since it has relatively few non-zero attributes.

The cosine similarity, defined next, is one of the most common measure of document similarity. If x and y are two document vectors, then

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|}, \tag{2.7}$$

where $\cdot$ indicates the vector dot product, $\mathbf{x} \cdot \mathbf{y} = \sum_{k=1}^{n} x_k y_k$, and $\|\mathbf{x}\|$ is the length of vector $\mathbf{x}$, $\|\mathbf{x}\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$.

Example 2.18 (Cosine Similarity of Two Document Vectors). This example calculates the cosine similarity for the following two data objects, which might represent document vectors:

$\mathbf{x} = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$
$\mathbf{y} = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$

$\mathbf{x} \cdot \mathbf{y} = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$
$\|\mathbf{x}\| = \sqrt{3*3 + 2*2 + 0*0 + 5*5 + 0*0 + 0*0 + 0*0 + 2*2 + 0*0 + 0*0} = 6.48$
$\|\mathbf{y}\| = \sqrt{1*1 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 1*1 + 0*0 + 2*2} = 2.24$
$\cos(\mathbf{x}, \mathbf{y}) = 0.31$

■

Issues in Proximity Calculation:

(1) how to handle the case in which attributes have different scales and/or are correlated,

(2) how to calculate proximity between objects that are composed of different types of attributes, e.g., quantitative and qualitative,

(3) and how to handle proximity calculation when attributes have different weights; i.e., when not all attributes contribute equally to the proximity of objects.

Standardization and Correlation for Distance Measures:

A related issue is how to compute distance when there is correlation between some of the attributes, perhaps in addition to difrerences in the ranges of values. A generalization of Euclidean distance, the Mahalanobis distance , is useful when attributes are correlated, have different ranges of values.

$$\text{mahalanobis}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})^T,$$

Combining Similarities for Heterogeneous Attributes:

The previous definitions of similarity were based on approaches that assumed all the attributes were of the same type. A general approach is needed when the attributes are of different types. One straightforward approach is to compute the similarity between each attribute separately and then combine these similarities using a method that results in a similarity between 0 and l.

Algorithm 2.1 Similarities of heterogeneous objects.

1: For the k^{th} attribute, compute a similarity, $s_k(\mathbf{x}, \mathbf{y})$, in the range $[0, 1]$.
2: Define an indicator variable, δ_k, for the k^{th} attribute as follows:

$$\delta_k = \begin{cases} 0 & \text{if the } k^{th} \text{ attribute is an asymmetric attribute and} \\ & \text{both objects have a value of 0, or if one of the objects} \\ & \text{has a missing value for the } k^{th} \text{ attribute} \\ 1 & \text{otherwise} \end{cases}$$

3: Compute the overall similarity between the two objects using the following formula:

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{n} \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^{n} \delta_k} \tag{2.15}$$

the formulas for proximity can be modified by weighting the contribution of each attribute.

If the weights w_k sum to 1, then (2.15) becomes

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{n} w_k \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^{n} \delta_k}. \tag{2.16}$$

The definition of the Minkowski distance can also be modified as follows:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} w_k |x_k - y_k|^r \right)^{1/r}. \tag{2.17}$$

Selecting the Right Proximity Measure:

The following are a few general observations that may be helpful. First, the type of proximity measure should fit the type of data. For many types of dense, continuous data, metric distance measures such as Euclidean distance are often used.

Proximity between continuous attributes is most often expressed in terms of differences, and distance measures provide a well-defined way of combining these differences into an overall proximity measure. Although attributes can have different scales and be of differing importance.

For sparse data, which often consists of asymmetric attributes, we typically employ similarity measures that ignore 0- 0 matches.

In some cases, transformation or normalization of t he data is important for obtaining a proper similarity measure since such transformations are not always present in proximity measures.

DATA WAREHOUSE AND OLAP TECHNOLOGY:

Data Warehouse:

A data warehouse is like a big library where an organization stores important information from different sources over time. It's separate from the database the organization uses to run its daily operations. This helps the organization to analyze and make decisions based on the stored data.

Data warehousing provides a way for business leaders to organize and use their data to make important decisions. It helps them to understand the past trends and patterns in their data which can be used to make future predictions and develop strategic plans.

Data warehouses are very useful for companies today as they provide a competitive advantage in a fast-changing world. Many organizations invest a lot of money in building enterprise-wide data warehouses to ensure they have easy access to their historical data for decision making.

Data Warehouse subject-oriented:

Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing.

Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

Data Warehouse integrated:

Constructed by integrating multiple, heterogeneous data sources

- relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
  - When data is moved to the warehouse, it is converted.

Data Warehouse time-variant:

The time horizon for the data warehouse is significantly longer than that of operational systems

- Operational database: current value data
- Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)

Every key structure in the data warehouse

- Contains an element of time, explicitly or implicitly
- But the key of operational data may or may not contain "time element".

Data Warehouse non-volatile:

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - initial loading of data and access of data.

Why a Separate Data Warehouse?

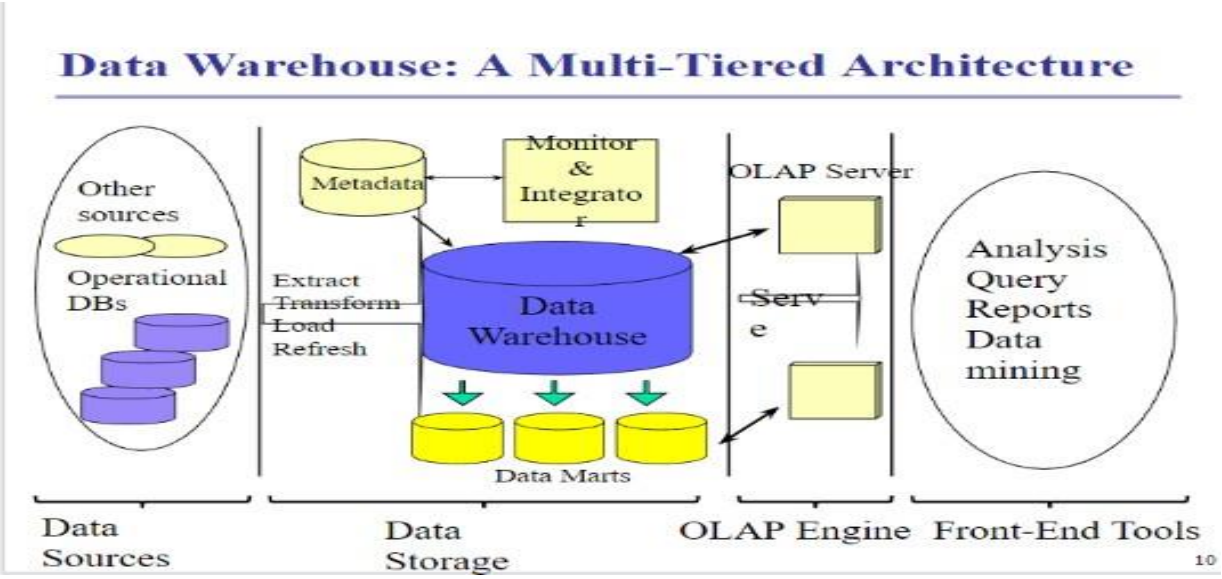
- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
  - missing data: Decision support requires historical data which operational DBs do not typically maintain
  - data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources

- data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

Table 3.1 Comparison between OLTP and OLAP systems.

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

NOTE: Table is partially based on [CD97].



Three Data Warehouse Models:

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

Extraction, Transformation, and Loading (ETL):

- **Data extraction**
  - get data from multiple, heterogeneous, and external sources
- **Data cleaning**
  - detect errors in the data and rectify them when possible
- **Data transformation**
  - convert data from legacy or host format to warehouse format
- **Load**
  - sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

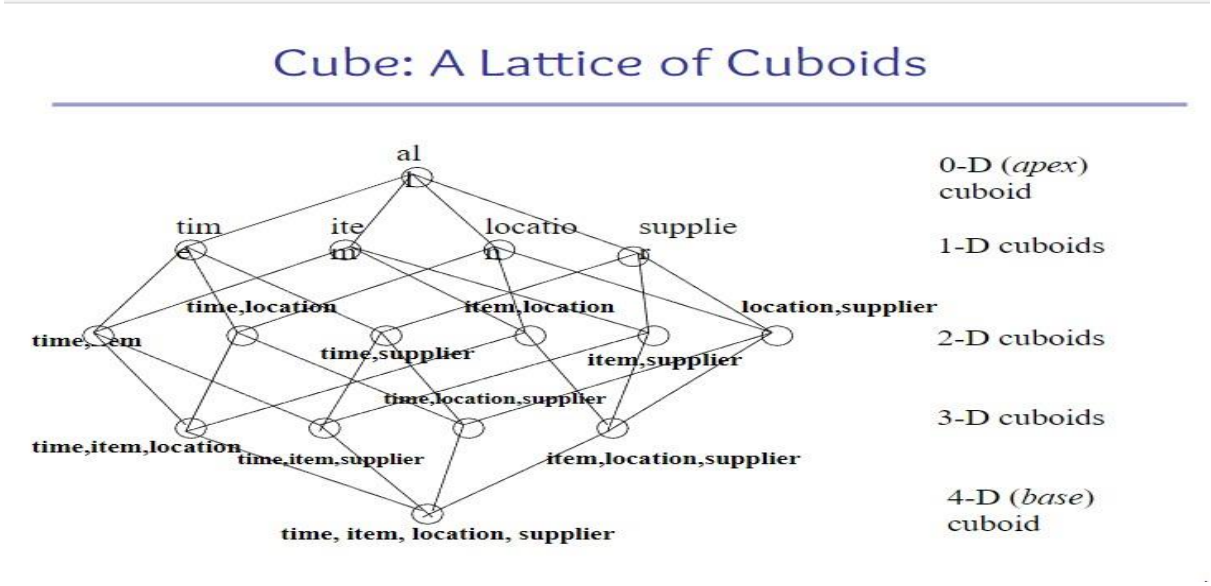
Metadata Repository:

- **Meta data** is the data defining warehouse objects. It stores:
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
- Operational meta-data
  - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarization
- The mapping from operational environment to the data warehouse
- Data related to system performance
  - warehouse schema, view and derived data definitions
- Business data
  - business terms and definitions, ownership of data, charging policies

From Tables and Spreadsheets to Data Cubes:

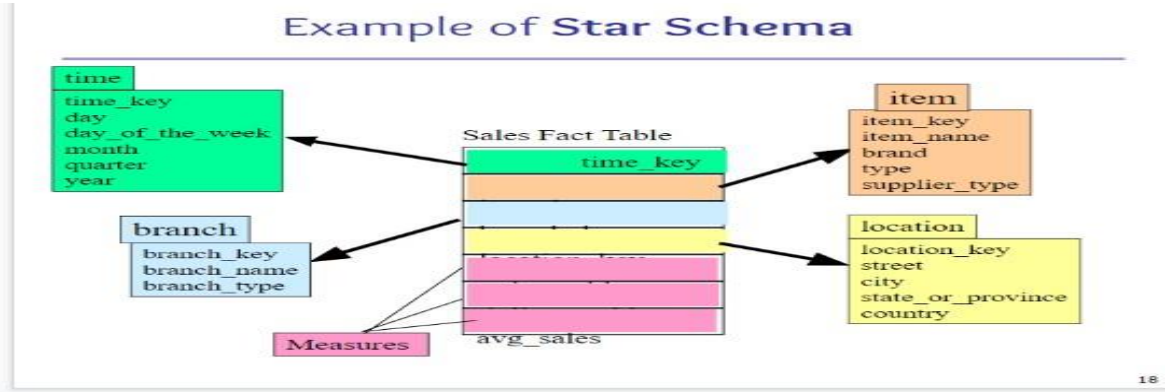
- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - **Dimension tables**, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - **Fact table** contains **measures** (such as dollars_sold) and keys to each of the related dimension tables
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of

summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

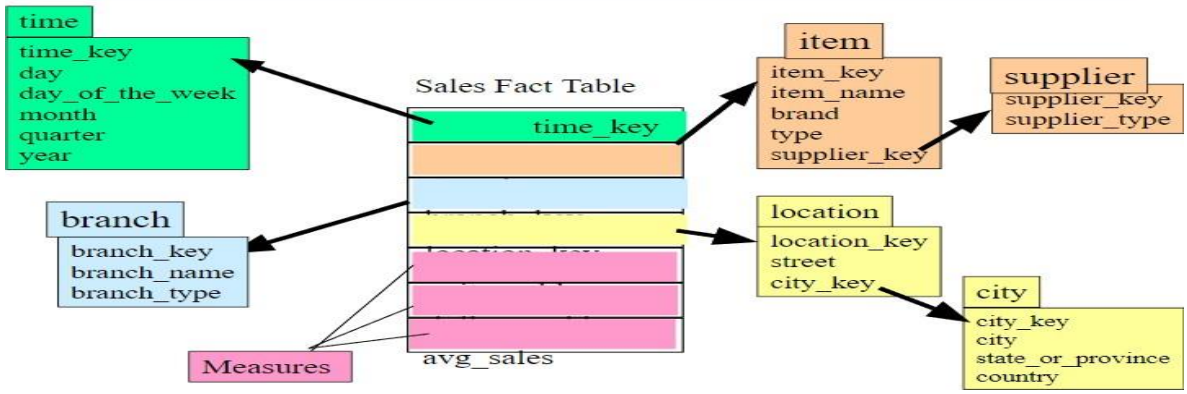


Conceptual Modeling of Data Warehouses:

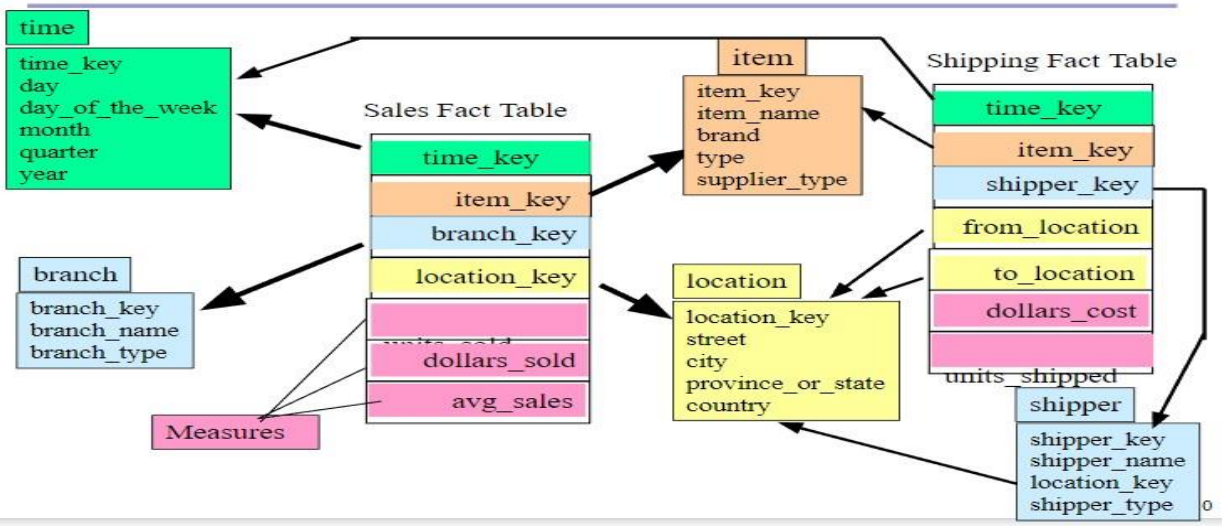
- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation



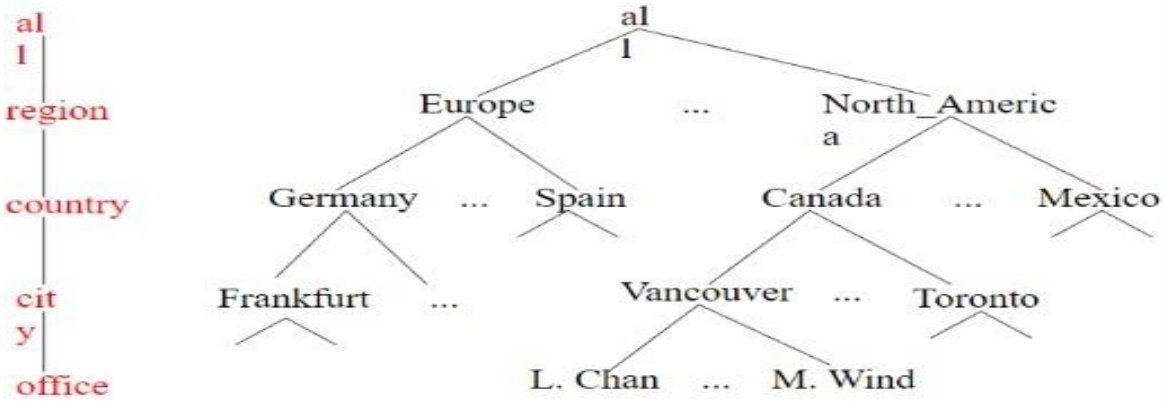
Example of **Snowflake Schema**



Example of **Fact Constellation**



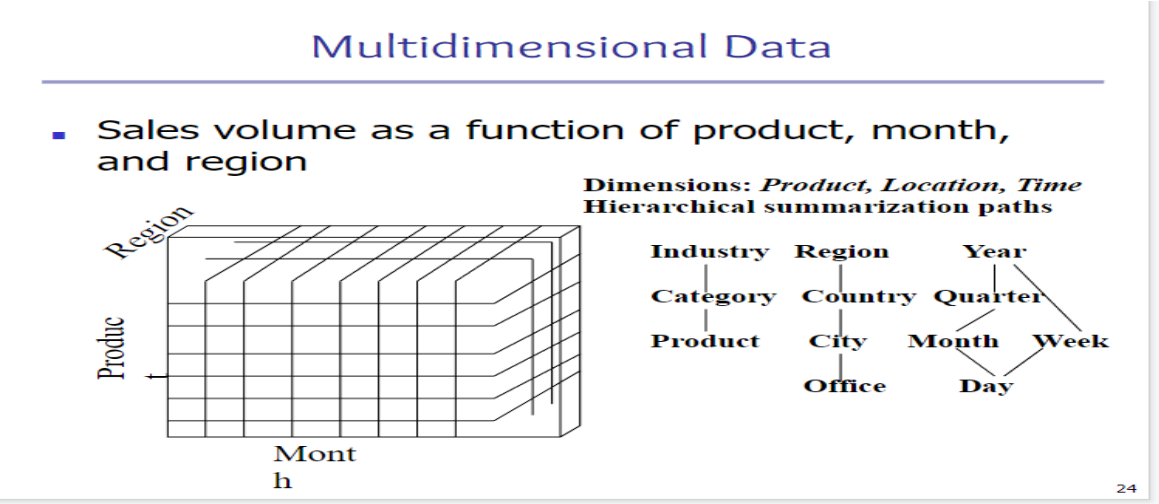
A Concept Hierarchy: **Dimension** (location)

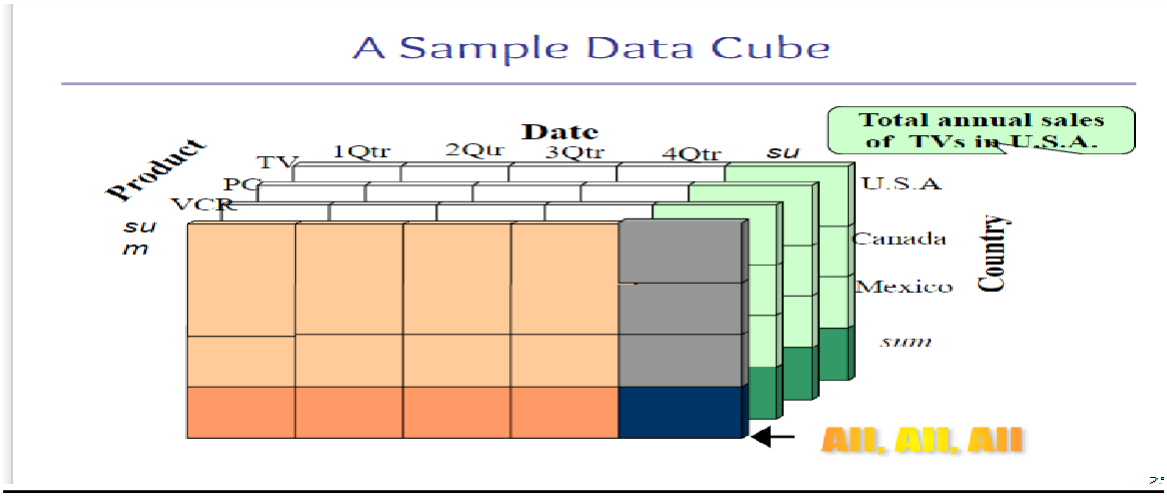


Data Cube Measures: Three Categories:

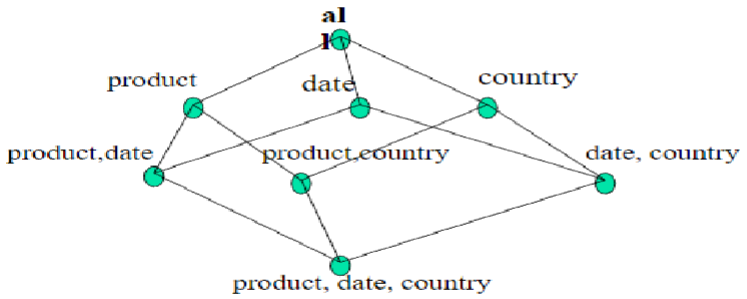
- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
    - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
    - E.g., avg(), min_N(), standard_deviation()
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
    - E.g., median(), mode(), rank()

View of Warehouses and Hierarchies:



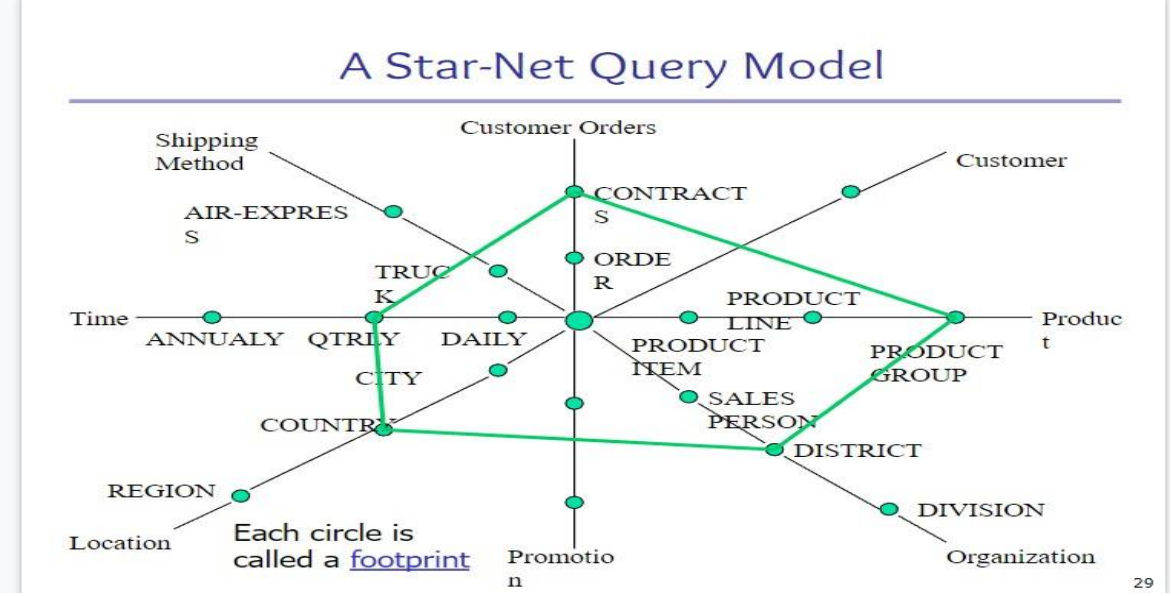


Cuboids Corresponding to the Cube



Typical OLAP Operations:

- Roll up (drill-up): summarize data
  - by climbing up hierarchy or by dimension reduction
- Drill down (roll down): reverse of roll-up
  - from higher level summary to lower level summary or detailed data, or introducing new dimensions
- Slice and dice: project and select
- Pivot (rotate):
  - reorient the cube, visualization, 3D to series of 2D planes
- Other operations
  - drill across: involving (across) more than one fact table
  - drill through: through the bottom level of the cube to its back-end relational tables (using SQL)

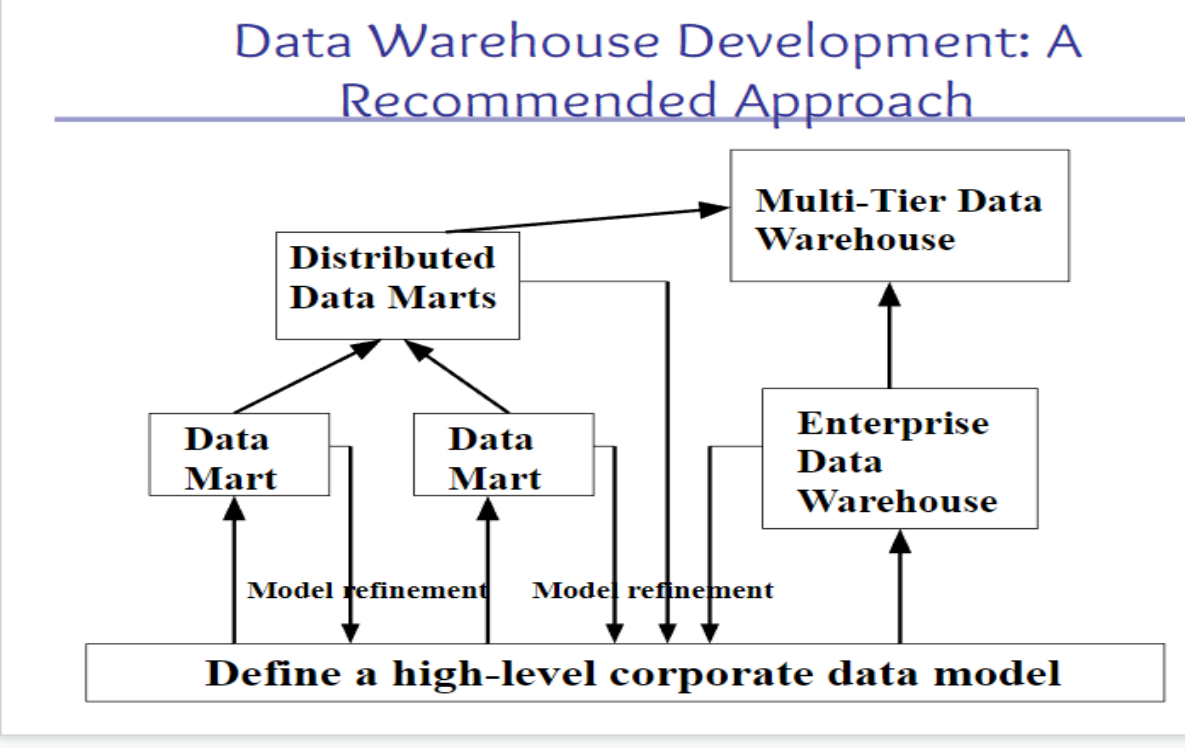


Design of Data Warehouse: A Business Analysis Framework:

- Four views regarding the design of a data warehouse
  - Top-down view
    - allows selection of the relevant information necessary for the data warehouse
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user
- **Top-down, bottom-up approaches or a combination** of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- **From software engineering point of view**
  - Waterfall: structured and systematic analysis at each step before proceeding to the next

- Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around

- **Typical data warehouse design process**
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the *grain* (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record



Data Warehouse Usage:

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing

        - multidimensional analysis of data warehouse data
        - supports basic OLAP operations, slice-dice, drilling, pivoting
    - Data mining
        - knowledge discovery from hidden patterns
        - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

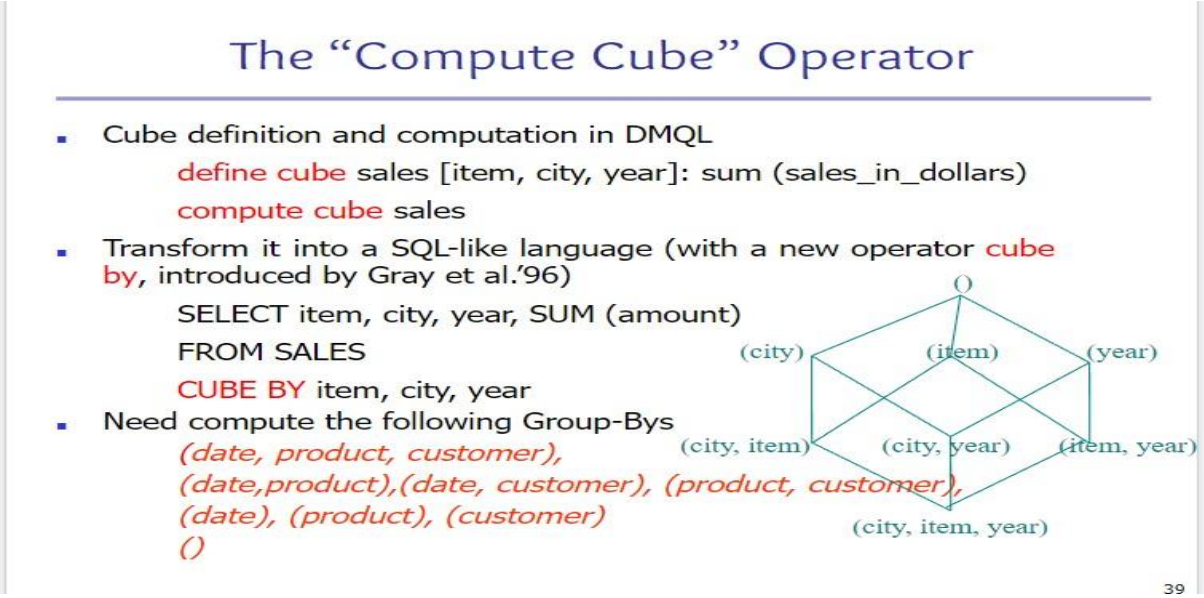
From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM):

- Why online analytical mining?
    - High quality of data in data warehouses
        - DW contains integrated, consistent, cleaned data
    - Available information processing structure surrounding data warehouses
        - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
    - OLAP-based exploratory data analysis
        - Mining with drilling, dicing, pivoting, etc.
    - On-line selection of data mining functions
        - Integration and swapping of multiple mining functions, algorithms, and tasks

Efficient Data Cube Computation:

- Data cube can be viewed as a lattice of cuboids
    - The bottom-most cuboid is the base cuboid
    - The top-most cuboid (apex) contains only one cell
    - How many cuboids in an n-dimensional cube with L levels?
- Materialization of data cube
    - Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)

- Selection of which cuboids to materialize
  - Based on size, sharing, access frequency, etc.
- The "Compute Cube" Operator



Indexing OLAP Data: Bitmap Index:

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Indexing OLAP Data: Join Indices:

- Join index: JI(R-id, S-id) where R (R-id, …) >< S (S-id, …)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join

- In data warehouses, join index relates the values of the dimensions of a start schema to rows in the fact table.
    - E.g. fact table: *Sales* and two dimensions *city* and *product*
        - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
    - Join indices can span multiple dimensions

Efficient Processing OLAP Queries:

- Determine which operations should be performed on the available cuboids
    - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection
- Determine which materialized cuboid(s) should be selected for OLAP op.
    - Let the query to be processed be on {*brand, province_or_state*} with the condition "*year = 2004*", and there are 4 materialized cuboids available:

      1) {*year, item_name, city*}

      2) {*year, brand, country*}

      3) {*year, brand, province_or_state*}

      4) {*item_name, province_or_state*} where *year = 2004*

      Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

OLAP Server Architectures:

- Relational OLAP (ROLAP)
    - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware

        - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
        - Greater scalability
- Multidimensional OLAP (MOLAP)
    - Sparse array-based multidimensional storage engine
    - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
    - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
    - Specialized support for SQL queries over star/snowflake schemas

Attribute-Oriented Induction:

- Proposed in 1989 (KDD '89 workshop)
- Not confined to categorical data nor particular measures
- How it is done?
    - Collect the task-relevant data (*initial relation*) using a relational database query
    - Perform generalization by attribute removal or attribute generalization
    - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
    - Interaction with users for knowledge presentation

Attribute-Oriented Induction: An Example:

Example: Describe general characteristics of graduate students in the University database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,

Select * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa)

from student

where student_status in {“Msc”, “MBA”, “PhD” }

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

Basic Principles of Attribute-Oriented Induction:

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*
- Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*’s higher level concepts are expressed in terms of other attributes
- Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize *A*
- Attribute-threshold control: typical 2-8, specified/default
- Generalized relation threshold control: control the final relation/rule size

Attribute-Oriented Induction: Basic Algorithm:

- InitialRel: Query processing of task-relevant data, deriving the *initial relation*.
- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- PrimeGen: Based on the PreGen plan, perform generalization to the right level to derive a “prime generalized relation”, accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Concept Description vs. Cube-Based OLAP:

- Similarity:
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing
- Differences:
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms

UNIT-2

1.) Explain various concepts of Data Warehouse.

Ans:

Data Warehouse:

A decision support database that is maintained separately from the organization's operational database

Support information processing by providing a solid platform of consolidated, historical data for analysis.

Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions.

Data warehouse systems are valuable tools in today's competitive, fast-evolving world. In the last several years, many firms have spent millions of dollars in building enterprise-wide data warehouses.

Data Warehouse subject-oriented:

Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing.

Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

Data Warehouse integrated:

Constructed by integrating multiple, heterogeneous data sources relational databases, flat files, on-line transaction records Data cleaning and data integration techniques are applied.

- Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
- When data is moved to the warehouse, it is converted.

Data Warehouse time-variant:

The time horizon for the data warehouse is significantly longer than that of operational systems

- Operational database: current value data
- Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years) Every key structure in the data warehouse
- Contains an element of time, explicitly or implicitly

- But the key of operational data may or may not contain “time element”.

Data Warehouse non-volatile:

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
- Does not require transaction processing, recovery, and concurrency control mechanisms
- Requires only two operations in data accessing:
- initial loading of data and access of data.

2.) What are the needs and goals of data warehouse.

Ans :

Goals of Data Warehousing :

- **To help reporting as well as analysis**
- **Maintain the organization's historical information**
- **Be the foundation for decision making.**

Need for Data Warehouse

Data Warehouse is needed for the following reasons:

1. **Business User: Business users require a data warehouse to view summarized data from the past. Since these people are non-technical, the data may be presented to them in an elementary form.**
2. **Store historical data: Data Warehouse is required to store the time variable data from the past. This input is made to be used for various purposes.**
3. **Make strategic decisions: Some strategies may be depending upon the data in the data warehouse. So, data warehouse contributes to making strategic decisions.**
4. **For data consistency and quality: Bringing the data from different sources at a commonplace, the user can effectively undertake to bring the uniformity and consistency in data.**
5. **High response time: Data warehouse has to be ready for somewhat unexpected loads and types of queries, which demands a significant degree of flexibility and quick response time.**

Benefits of Data Warehouse

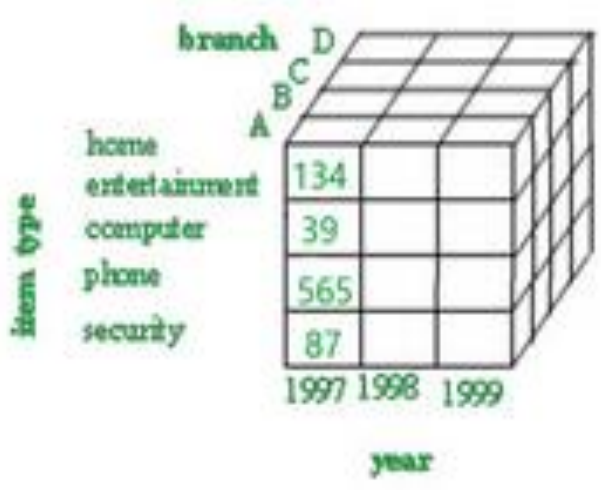
1. **Understand business trends and make better forecasting decisions.**
2. **Data Warehouses are designed to perform well enormous amounts of data.**
3. **The structure of data warehouses is more accessible for end-users to navigate, understand, and query.**
4. **Queries that would be complex in many normalized databases could be easier to build and maintain in data warehouses.**
5. **Data warehousing is an efficient method to manage demand for lots of information from lots of users.**
6. **Data warehousing provide the capabilities to analyze a large amount of historical data.**

3.) What is Data Cube and how to give the representation.

Ans :

Grouping of data in a multidimensional matrix is called data cubes. In Dataware housing, we generally deal with various multidimensional data models as the data will be represented by multiple dimensions and multiple attributes. This multidimensional data is represented in the data cube as the cube represents a high-dimensional space. The Data cube pictorially shows how different

attributes of data are arranged in the data model. Below is the diagram of a general data cube.



The example above is a 3D cube having attributes like branch(A,B,C,D),item type(home,entertainment,computer,phone,security), year(1997,1998,1999) .

Data cube classification:
The data cube can be classified into two categories:

- **Multidimensional data cube: It basically helps in storing large amounts of data by making use of a multi-dimensional array. It increases its efficiency by keeping an index of each dimension. Thus, dimensional is able to retrieve data fast.**
- **Relational data cube: It basically helps in storing large amounts of data by making use of relational tables. Each relational table displays the dimensions of the data cube. It is slower compared to a Multidimensional Data Cube.**

Advantages of data cubes:

- **Multi-dimensional analysis: Data cubes enable multi-dimensional analysis of business data, allowing users to view data from different perspectives and levels of detail.**
- **Interactivity: Data cubes provide interactive access to large amounts of data, allowing users to easily navigate and manipulate the data to support their analysis.**
- **Speed and efficiency: Data cubes are optimized for OLAP analysis, enabling fast and efficient querying and aggregation of data.**
- **Data aggregation: Data cubes support complex calculations and data aggregation, enabling users to quickly and easily summarize large amounts of data.**
- **Improved decision-making: Data cubes provide a clear and comprehensive view of business data, enabling improved decision-making and business intelligence.**

- **Accessibility: Data cubes can be accessed from a variety of devices and platforms, making it easy for users to access and analyze business data from anywhere.**
- **Helps in giving a summarised view of data.**
- **Data cubes store large data in a simple way.**
- **Data cube operation provides quick and better analysis,**
- **Improve performance of data.**

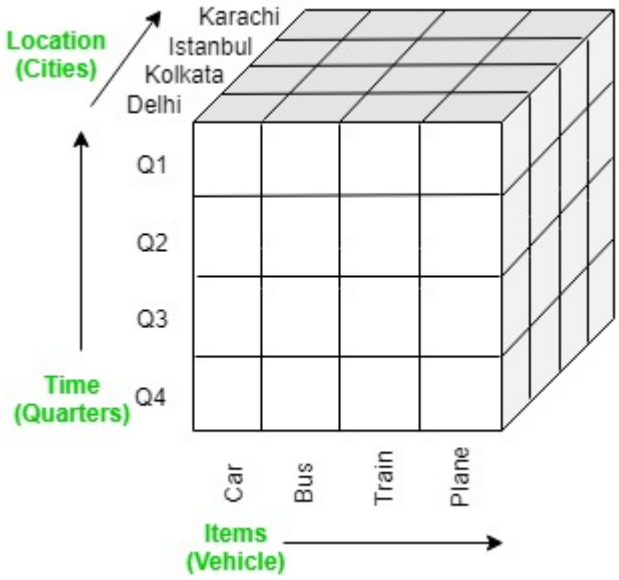
Disadvantages of data cube:

- **Complexity: OLAP systems can be complex to set up and maintain, requiring specialized technical expertise.**
- **Data size limitations: OLAP systems can struggle with very large data sets and may require extensive data aggregation or summarization.**
- **Performance issues: OLAP systems can be slow when dealing with large amounts of data, especially when running complex queries or calculations.**
- **Data integrity: Inconsistent data definitions and data quality issues can affect the accuracy of OLAP analysis.**
- **Cost: OLAP technology can be expensive, especially for enterprise-level solutions, due to the need for specialized hardware and software.**
- **Inflexibility: OLAP systems may not easily accommodate changing business needs and may require significant effort to modify or extend.**

4.) Define OLAP and explain its operations.

Ans :

OLAP stands for *Online Analytical Processing* Server. It is a software technology that allows users to analyze information from multiple database systems at the same time. It is based on multidimensional data model and allows the user to query on multi-dimensional data (eg. Delhi -> 2018 -> Sales data). OLAP databases are divided into one or more cubes and these cubes are known as *Hyper-cubes*.



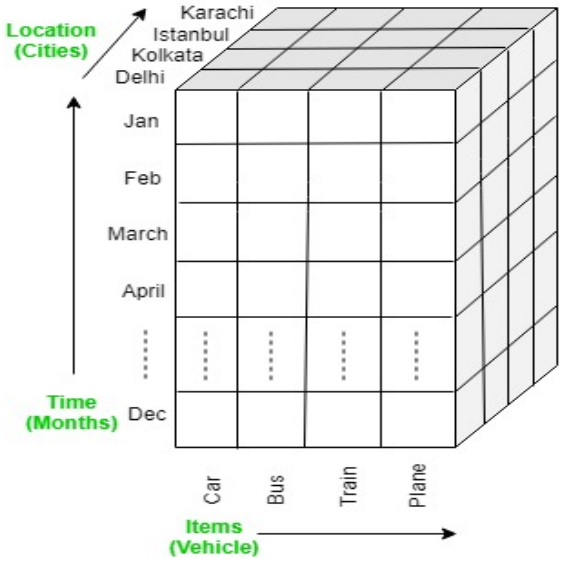
OLAP operations:

There are five basic analytical operations that can be performed on an OLAP cube:

1.)Drill down: In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:

- **Moving down in the concept hierarchy**
- **Adding a new dimension**

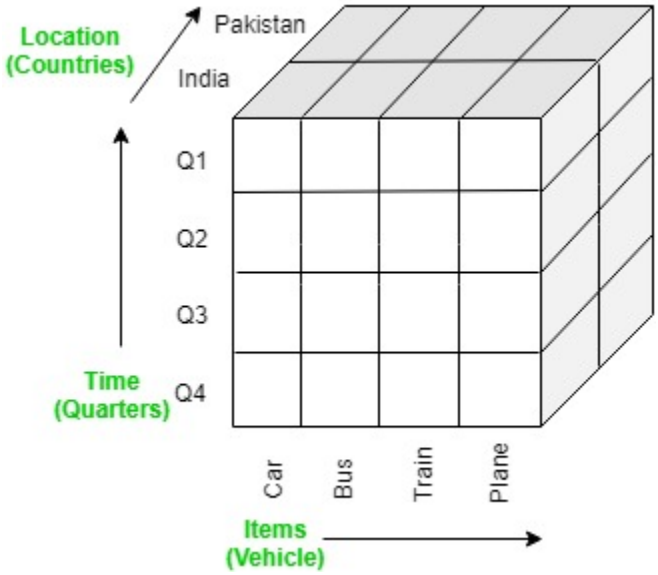
In the cube given in overview section, the drill down operation is performed by moving down in the concept hierarchy of *Time* dimension (Quarter -> Month)



2.) Roll up: It is just opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:

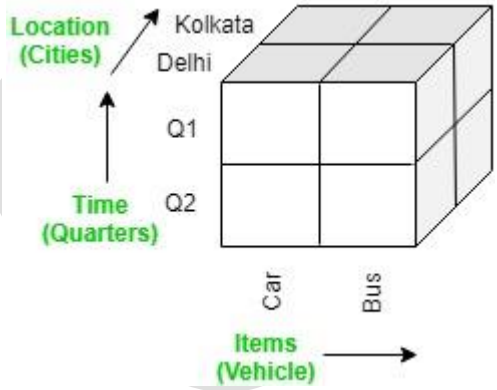
- **Climbing up in the concept hierarchy**
- **Reducing the dimensions**

In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of *Location* dimension (City -> Country).



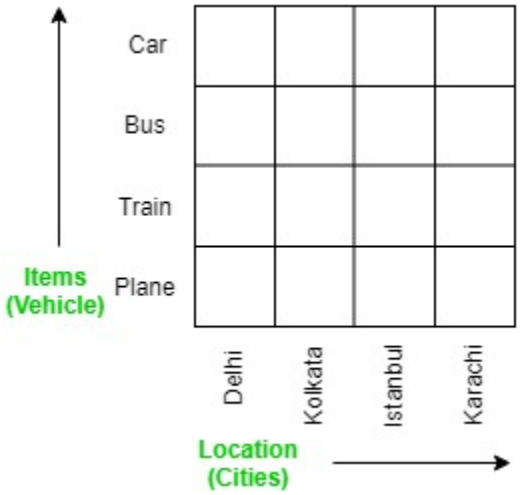
3.) Dice: It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting following dimensions with criteria:

- **Location = "Delhi" or "Kolkata"**
- **Time = "Q1" or "Q2"**
- **Item = "Car" or "Bus"**



4.) Slice: It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = "Q1".

5.) Pivot: It is also known as *rotation* operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing pivot operation gives a new view of it.



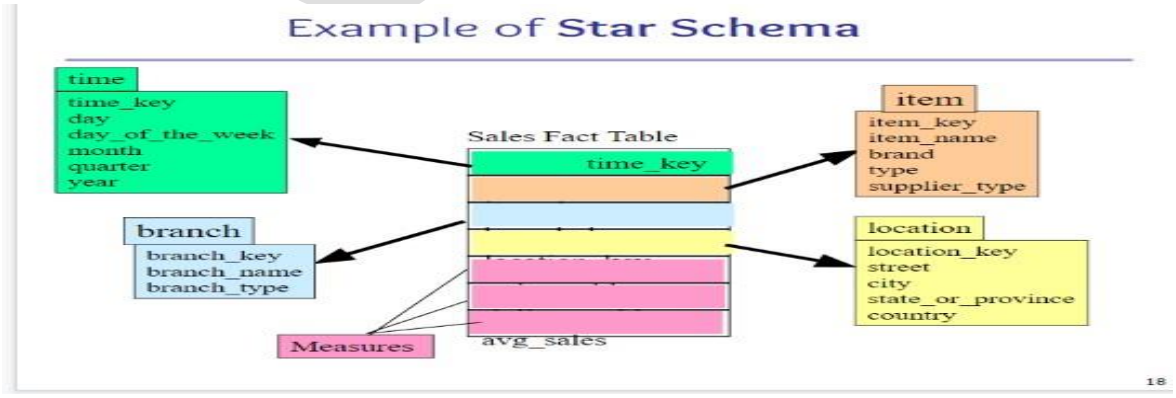
5.) Explain the following.

(i)Star Schema
(ii)Snowflake Schema
(iii)fact constellation Schema

Ans :

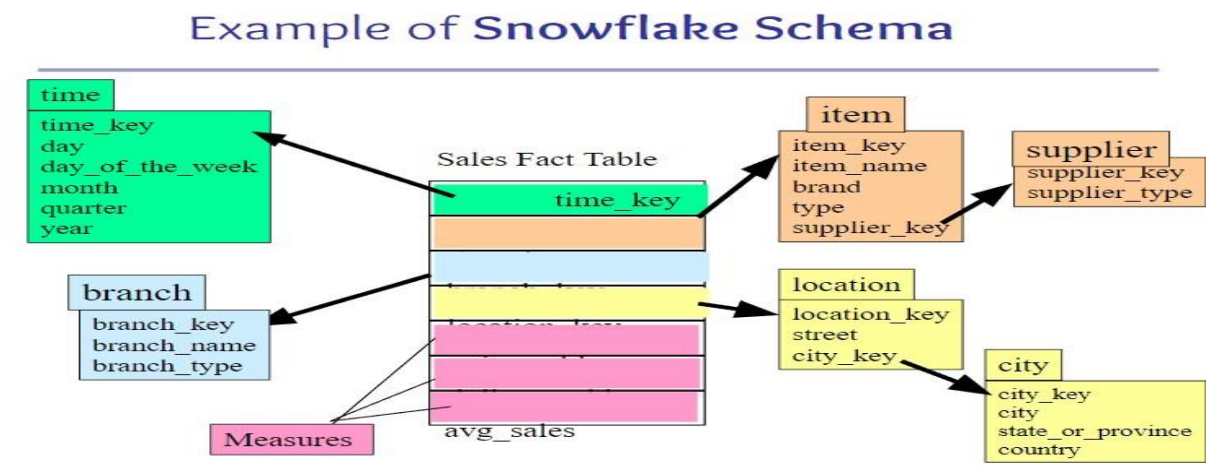
Star Schema :

A fact table in the middle connected to a set of dimension tables



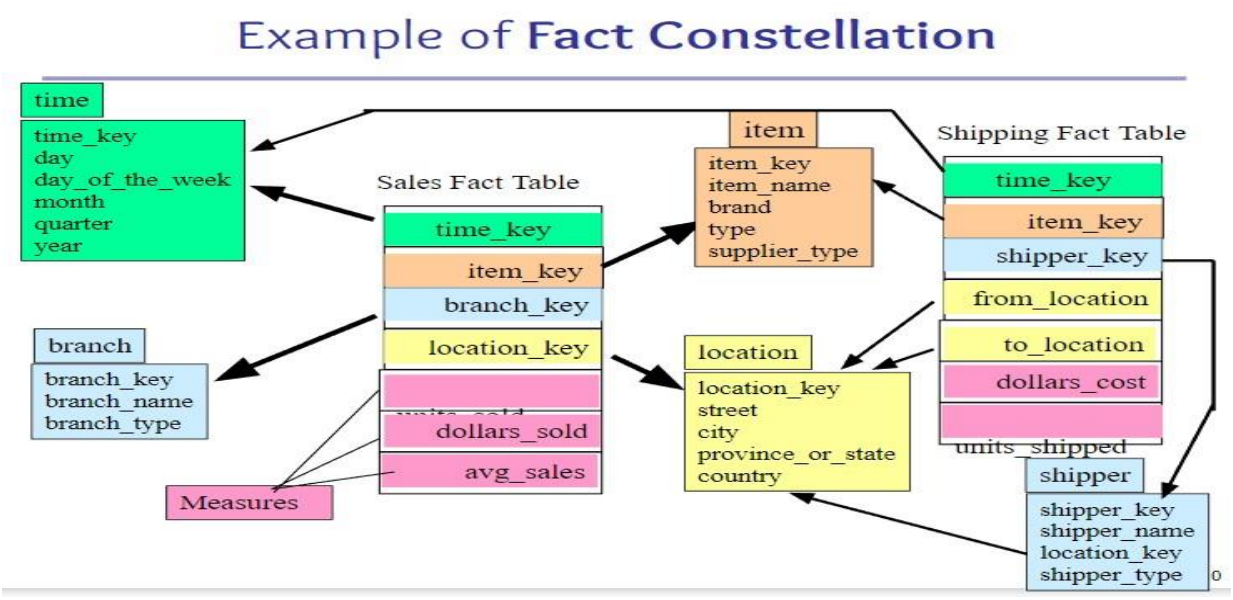
Snowflake schema:

A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake



Fact constellations:

Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation



Q) Write about Apriori algorithm with example

A) The Apriori algorithm is a popular algorithm used in data mining and association rule learning. It is designed to discover frequent itemsets from a given dataset and generate association rules based on the frequent itemsets. The algorithm is widely used in market basket analysis, where the goal is to find relationships between items frequently purchased together.

The Apriori algorithm uses an iterative approach to find frequent itemsets by exploiting the downward closure property. This property states that if an itemset is infrequent, then all of its supersets will also be infrequent. The algorithm starts by finding frequent individual items, then extends the search to larger itemsets until no more frequent itemsets can be found.

Here's an overview of the Apriori algorithm:

Support: Define a minimum support threshold (a value between 0 and 1) to determine the minimum frequency required for an itemset to be considered frequent. For example, if the minimum support threshold is set to 0.5, itemsets occurring in at least 50% of the transactions will be considered frequent.

Generate frequent 1-itemsets: Scan the dataset to find the support of each individual item. Discard any items that do not meet the minimum support threshold.

Generate frequent k-itemsets: Use the frequent (k-1)-itemsets obtained in the previous step to generate candidate k-itemsets. To do this, join pairs of frequent (k-1)-itemsets and check if their (k-1) subsets are all frequent. Prune any candidate k-itemsets that contain subsets that are not frequent.

Count support for candidate itemsets: Scan the dataset again and count the support of each candidate itemset by checking how many transactions contain the itemset.

Prune infrequent itemsets: Discard any candidate itemsets that do not meet the minimum support threshold.

Repeat steps 3-5: Repeat steps 3 to 5 until no more frequent itemsets can be generated.

Once the frequent itemsets are discovered, association rules can be generated based on these itemsets. Association rules express relationships between different items and are typically represented in the form of "If X, then Y." The strength of an association rule is measured by two metrics: support and confidence.

Here's a simplified example to illustrate the Apriori algorithm:

Suppose we have a transaction dataset representing purchases made by customers:

Transaction 1: {bread, milk, eggs}

Transaction 2: {bread, diapers}

Transaction 3: {milk, diapers}

Transaction 4: {bread, milk, diapers}

Transaction 5: {bread, diapers}

Let's set the minimum support threshold to 0.4 (40% of transactions):

Generate frequent 1-itemsets:

{bread}: 4 (Transaction 1, 2, 4, 5)

{milk}: 3 (Transaction 1, 3, 4)

{eggs}: 1 (Transaction 1)

{diapers}: 4 (Transaction 2, 3, 4, 5)

Generate frequent 2-itemsets:

{bread, milk}: 2 (Transaction 1, 4)

{bread, diapers}: 3 (Transaction 2, 4, 5)

{milk, diapers}: 2 (Transaction 3, 4)

Generate frequent 3-itemsets:

{bread, milk, diapers}: 2 (Transaction 4)

No more frequent itemsets can be generated, so we stop here.

Based on the frequent itemsets, we can generate association rules. Let's consider an example rule.

Q) Explain how association rules are generated from frequent item sets.

A) Association rules are generated from frequent itemsets to express relationships between different items in a dataset. These rules provide valuable insights into the co-occurrence and dependencies among items. The generation of association rules involves two main components: support and confidence.

Support measures the frequency of an itemset in the dataset, indicating how often the itemset appears in transactions. It is calculated as the ratio of the number of transactions containing the itemset to the total number of transactions. A support value of 1 means the itemset appears in all transactions, while a value of 0 means it does not appear in any transaction.

Confidence measures the strength of the relationship between items in an association rule. It is calculated as the ratio of the number of transactions containing both the antecedent and consequent of the rule to the number of transactions containing only the antecedent. Confidence reflects the conditional probability that the consequent will occur given the antecedent.

To generate association rules from frequent itemsets, the following steps are typically followed:

Start with frequent itemsets: Begin with the frequent itemsets obtained from the Apriori algorithm or any other method for discovering frequent itemsets.

Generate rules from frequent itemsets: For each frequent itemset, generate all possible non-empty subsets of items as potential antecedents. These subsets represent different combinations of items that could potentially imply other items. The remaining items in the frequent itemset become the consequents.

Calculate support and confidence: For each association rule generated, calculate its support and confidence. Support is determined by the frequency of the entire rule (both antecedent and consequent) in the dataset, while confidence is calculated based on the support of the antecedent and consequent.

Apply thresholding: Set minimum thresholds for support and confidence to filter out weak rules. Only association rules that meet or exceed these thresholds are considered significant and retained.

Evaluate and interpret rules: Analyze the generated association rules based on their support, confidence, and other evaluation measures. Interpret the rules to gain insights into the relationships between items and make informed decisions or recommendations.

For example, consider the frequent itemset {bread, milk} with support value 0.4 and confidence value 0.5. From this itemset, we can generate the following association rule:

If a customer buys bread, then they are likely to buy milk (confidence: 0.5).

This rule suggests a positive association between bread and milk, indicating that these two items are often purchased together. The confidence value of 0.5 indicates that in 50% of transactions where bread is purchased, milk is also present.

By generating and analyzing association rules from frequent itemsets, we can uncover interesting patterns, correlations, and dependencies within a dataset, which can be valuable for market basket analysis, recommendation systems, and decision-making in various domains.

Q) Write about FP- growth algorithm with example

A) The FP-growth algorithm is a popular algorithm used in data mining for frequent itemset mining and association rule learning. It efficiently discovers frequent itemsets by building a compact data structure called the FP-tree. The algorithm is known for its ability to handle large datasets and its relatively fast performance compared to other frequent itemset mining algorithms like Apriori.

Here's an overview of the FP-growth algorithm:

Build the FP-tree: Scan the dataset to construct the FP-tree. The FP-tree structure consists of a root node and a set of conditional subtrees. Each node represents an item, and the edges between nodes indicate the frequency and order of item occurrences. The items are usually sorted in descending order of their support count in the dataset.

Generate conditional pattern bases: For each frequent item in the dataset, create a conditional pattern base. A conditional pattern base is a set of all transactions that contain the frequent item, with the frequent item removed from

each transaction. These conditional pattern bases will be used to construct conditional FP-trees.

Build conditional FP-trees: For each frequent item, construct a conditional FP-tree using its corresponding conditional pattern base. The conditional FP-tree is built in a similar manner as the main FP-tree, recursively constructing the tree structure by scanning the conditional pattern bases.

Mine frequent itemsets: Starting with the least frequent item in the FP-tree, recursively mine frequent itemsets by performing a depth-first search. For each item in the tree, create a conditional pattern base and construct a conditional FP-tree. By traversing the conditional FP-tree, generate frequent itemsets based on the item prefix paths.

Repeat steps 3-4: Repeat steps 3 and 4 until no more frequent itemsets can be found.

FP Growth Method:

- It is used to find frequent patterns & generates the strong association rules.
- It eliminates the drawbacks of Apriori algorithm i.e Candidate itemset generation.
- we need to construct a FP-Tree.

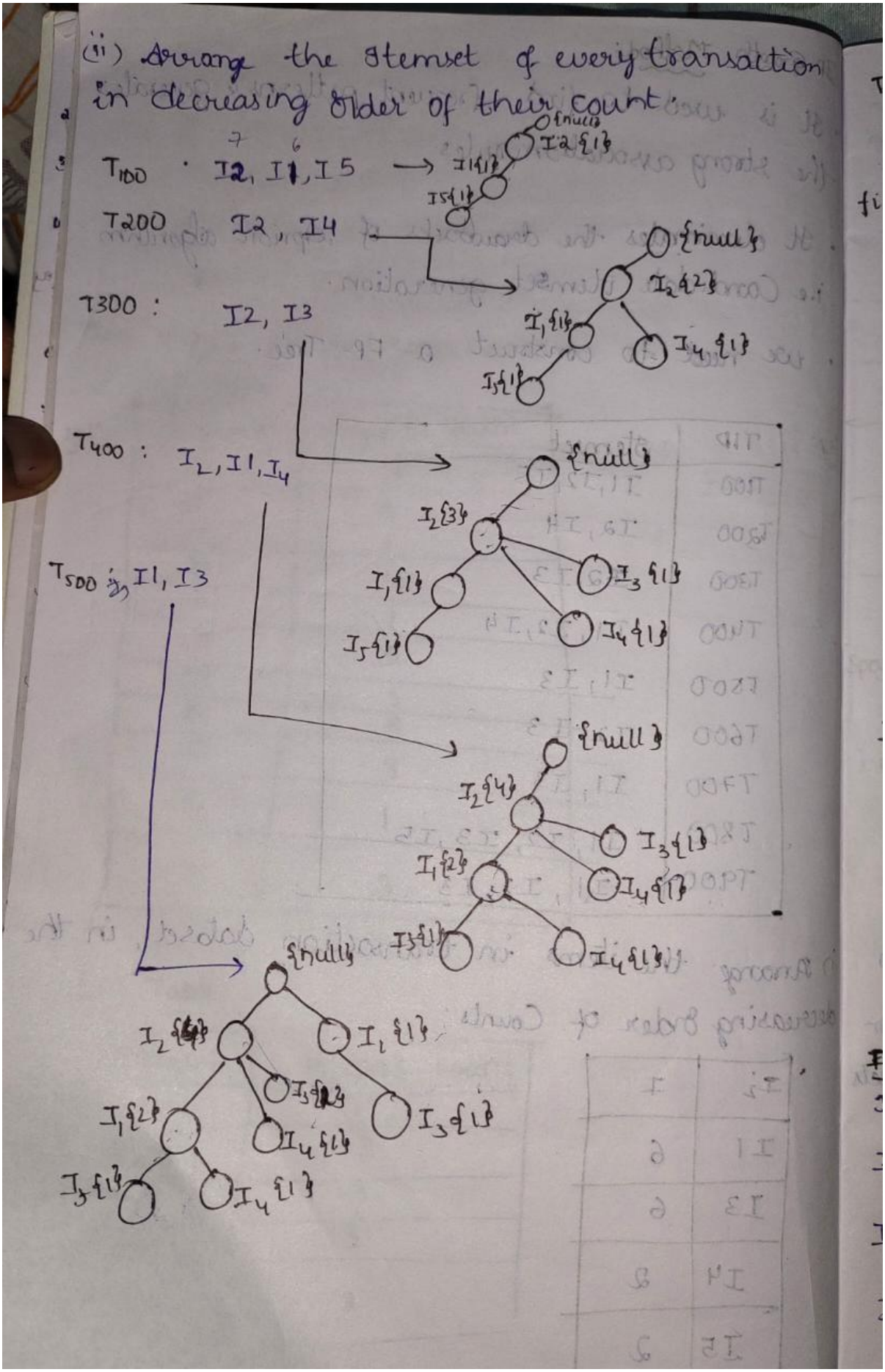
ex:

TID	Itemset
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

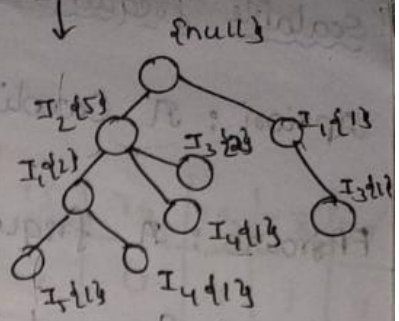
i) Arrange the items in transaction dataset, in the decreasing order of Counts.

I_i	7
I1	6
I3	6
I4	2
I5	2

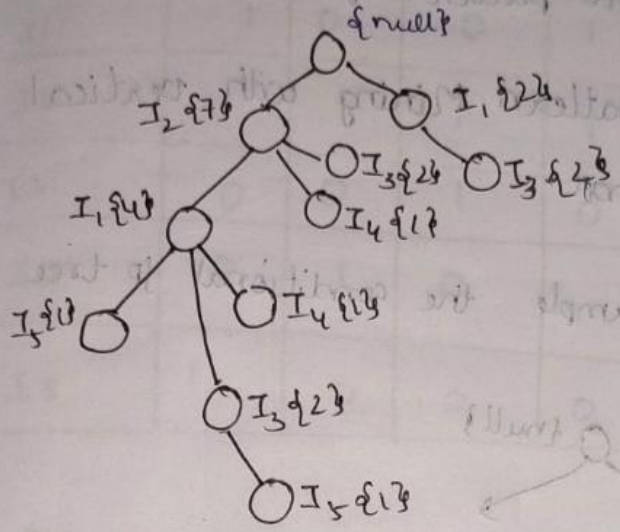
(ii) Arrange the itemset of every transaction in decreasing order of their count.



T_{600}: I2, I3



finally T_{600} : I2, I1, I3



item	Conditional Pattern base	Conditional FP tree
I5	{{I2, I1 : 1}, {I2, I1, I3 : 1}}	<I2 : 2, I1 : 2>
I4	{{I2, I1 : 1}, {I2 : 1}}	<I2 : 2>
I3	{{I2, I1 : 2}, {I2 : 2}, {I1 : 2}}	<I2 : 4, I1 : 2>, <I1 : 2>
I1	{{I2 : 4}}	<I2 : 4>

Item	Frequent Patterns generated:
I5	{I2, I5 : 2}, {I1, I5 : 2}, {I2, I1, I5 : 2}
I4	{I2, I4 : 2}
I3	{I2, I3 : 4}, {I1, I3 : 4}, {I2, I1, I3 : 2}
I1	{I2, I1 : 4}

Q) Explain about Naïve Bayes algorithm with example

A) The Naïve Bayes algorithm is a popular classification algorithm based on Bayes' theorem and the assumption of feature independence. It is widely used for text classification, spam filtering, sentiment analysis, and various other tasks involving categorical data. Despite its simplicity, Naïve Bayes often performs well and is computationally efficient.

Here's how the Naïve Bayes algorithm works:

Bayes' Theorem:

Bayes' theorem provides a way to calculate the probability of an event given prior knowledge or evidence. It states that the posterior probability of an event A, given evidence B, is proportional to the product of the prior probability of A and the conditional probability of B given A, divided by the marginal probability of B. Mathematically, it can be represented as:

$$P(A|B) = (P(B|A) * P(A)) / P(B)$$

Feature Independence Assumption:

Naïve Bayes assumes that the features (attributes) used for classification are independent of each other, given the class label. This is a simplifying assumption that helps in making the algorithm computationally efficient, although it may not hold true in all cases.

Training Phase:

During the training phase, Naïve Bayes calculates the prior probabilities and conditional probabilities based on the training dataset.

Prior probabilities (P(C)):

Calculate the probability of each class label in the training dataset.

Conditional probabilities (P(X|C)):

For each feature, calculate the conditional probability of its values given each class label. This involves estimating the likelihood of each feature value occurring, given each class label.

Classification Phase:

During the classification phase, Naïve Bayes calculates the posterior probabilities of each class label for a given instance, and assigns the instance to the class label with the highest posterior probability.

Posterior probabilities (P(C|X)):

Calculate the posterior probability of each class label given the instance's feature values using Bayes' theorem. Since the denominator P(X) is the same for all class labels, it can be ignored during comparison.

Assigning class label:

Assign the instance to the class label with the highest posterior probability.

Let's illustrate the Naïve Bayes algorithm with a simple example of email spam classification:

Suppose we have a training dataset of emails labeled as "spam" or "not spam" and the following features: "contains the word 'money'" (M), "contains the word 'lottery'" (L), and "contains the word 'free'" (F).

Training dataset:

Email 1: "money lottery" (spam)

Email 2: "free money" (spam)

Email 3: "free lottery" (spam)

Email 4: "no money" (not spam)

Email 5: "no lottery" (not spam)

Calculate Prior Probabilities:

P(spam) = 3/5

P(not spam) = 2/5

Calculate Conditional Probabilities:

For the feature M:

P(M|spam) = 2/3

P(M|not spam) = 1/2

For the feature L:

P(L|spam) = 2/3

P(L|not spam) = 1/2

For the feature F:

P(F|spam) = 3/3

P(F|not spam) = 0/2

Classification:

Let's classify a new email: "money free lottery"

Calculate the posterior probabilities:

P(spam|X) = P(M|spam) * P(F|spam) * P(L|spam) * P(spam) = (2/3) * (3/3) * (2/3) * (3/5) = 4/15

P(not spam|X) = P(M|not spam) * P(F|not spam) * P(L|not spam) * P(not spam) = (1/2) * (0/2) * (1/2) * (2/5) = 0

Since P(spam|X) > P(not spam|X), classify the email as "spam".

In this example, the Naïve Bayes algorithm predicts the email as "spam" based on the highest posterior probability.

Naïve Bayes is a simple yet effective algorithm for classification tasks, especially when dealing with categorical data and large datasets. It provides probabilistic predictions and can handle high-dimensional feature spaces. However, it may not perform well when the independence assumption is violated or when features are highly correlated.

Q) Discuss in detail about Decision tree algorithm with example.

A) The Decision Tree algorithm is a popular supervised machine learning algorithm used for classification and regression tasks. It creates a tree-like model of decisions and their possible consequences based on the features of the input data. Decision trees are easy to interpret and can handle both categorical and numerical data. They are widely used in various domains, including finance, healthcare, and marketing.

Here's an in-depth explanation of the Decision Tree algorithm:

1. ****Tree Structure:****

Decision trees consist of nodes and edges. Each node represents a feature or attribute, and each edge represents a decision rule or condition. The tree structure begins with a root node and branches out to internal nodes, which further branch out to leaf nodes representing the final decisions or predictions.

2. ****Node Types:****

- **Root Node:** The topmost node in the tree, representing the entire dataset.

- **Internal Node:** Represents a feature or attribute along with a decision rule.

- **Leaf Node:** Represents a class label or a numerical value (for regression tasks) that provides the final decision or prediction.

3. **<u>Building the Decision Tree:**</u>

The process of building a decision tree involves recursively splitting the dataset based on the features that provide the most significant information gain or decrease in impurity. The key steps are as follows:

- **Selecting the best attribute:** Calculate a measure of impurity or information gain for each attribute. Common measures include Gini Index and Information Gain (using entropy). The attribute with the highest information gain or the lowest impurity is selected as the splitting attribute for the current node.

- **Splitting the dataset:** Divide the dataset into subsets based on the chosen attribute. Each subset represents a branch or child node of the current node.

- **Repeat recursively:** For each child node, repeat the above steps until a stopping criterion is met, such as reaching a maximum depth, a minimum number of instances per leaf, or a minimum information gain threshold.

4. **<u>**Handling Categorical and Numerical Features:**</u>

- **Categorical Features:** For categorical features, each branch represents a unique value of the attribute, and instances are assigned to the corresponding branch based on their attribute value.

- **Numerical Features:** For numerical features, various strategies can be employed to determine the split point. These include binary splitting, multiway splitting, and threshold-based splitting. The optimal splitting point is chosen based on criteria like information gain or mean squared error reduction.

5. **Handling Missing Values:**

Decision trees can handle missing values by employing surrogate splits or assigning a default path for instances with missing attribute values. Surrogate splits use alternative features to make decisions when the primary splitting attribute has missing values.

6. **Pruning the Tree:**

After constructing the tree, pruning techniques can be applied to reduce overfitting. Pruning involves removing unnecessary branches or merging nodes to simplify the tree while maintaining its predictive accuracy. Common pruning methods include cost complexity pruning and reduced error pruning.

Let's illustrate the Decision Tree algorithm with a simple example of classifying fruits based on color and diameter:

Training Dataset:

Fruit	Color	Diameter (cm)
Apple	Red	5
Apple	Red	6
Orange	Orange	7
Orange	Orange	8

1. **Selecting the best attribute:**

Calculate the information gain for each attribute:

- Information Gain(Color) = Entropy(Fruits) - [P(Red) * Entropy(Red) + P(Orange) * Entropy(Orange)]

- Information Gain(Diameter) = Entropy(Fruits) - [P(D<=5) * Entropy(D<=5

) + P(D>5) * Entropy(D>5)]

Choose the attribute with the highest information gain. In this case, let's assume Color has the highest information gain.

2. **Splitting the dataset:**

Create child nodes for each unique value of the selected attribute (Color). We have two unique values: Red and Orange.

3. **Repeat recursively:**

- For the Red branch, all instances belong to the Apple class, so it becomes a leaf node with the class label Apple.

- For the Orange branch, all instances belong to the Orange class, so it becomes a leaf node with the class label Orange.

The resulting decision tree would look like this:

```
```

```
        Color
        /   \
       Red  Orange
      (Apple) (Orange)
```

Now, if we want to classify a new fruit with a color of Red and a diameter of 7 cm, we traverse the decision tree by following the Color attribute. Since the color is Red, we reach the leaf node labeled Apple, and the prediction is Apple.

The Decision Tree algorithm provides interpretable models that can be visualized and understood easily. However, it may suffer from overfitting if not pruned properly. It is also sensitive to small changes in the training data, which can result in different tree structures. Nonetheless, Decision Trees are widely used due to their simplicity, scalability, and effectiveness in various machine learning tasks.

Q) What is Meant by Density Based methods explain with example

A) Density-based methods are a type of clustering algorithm that group data points based on their proximity and density. These methods identify areas of high-density regions and separate them from low-density regions. Density-based methods are particularly useful for discovering clusters with arbitrary shapes, handling noise and outliers, and scaling well to large datasets.

One popular density-based clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN works by defining a dense region as a cluster and a sparser

region as noise. The key parameters for DBSCAN are the radius (eps) and the minimum number of points (minPts) required to form a dense region.

Here's how DBSCAN works:

Finding Core Points:

DBSCAN starts by randomly selecting an unvisited data point and checks if it has at least minPts neighboring points within the radius eps. If so, the point is considered a core point, and all its neighbors are added to its cluster.

Finding Border Points:

If a data point has fewer than minPts neighbors but belongs to the radius eps of a core point, it is considered a border point. Border points are assigned to the same cluster as their corresponding core point.

Finding Noise Points:

If a data point does not belong to any core point's radius, it is considered a noise point and is not assigned to any cluster.

Merging Clusters:

DBSCAN repeats the above steps until all points are visited. After the clusters are formed, they may contain outliers or noise. DBSCAN

provides an additional step to merge clusters that are close enough, based on their minimum distance.

Let's illustrate the DBSCAN algorithm with a simple example of clustering points in a 2D space:

Dataset:

Point	x	y
A	1	2
B	1	4
C	2	3
D	5	7
E	6	6
F	7	5
G	8	4

Finding Core Points:

Let's assume the radius eps is 2 and the minimum points minPts is 2. Point A has two neighboring points B and C within the radius eps and is considered a core point. Similarly, point C has two neighbors A and B, making it a core point. Points D, E, F, and G do not have enough neighbors and are not core points.

Finding Border Points:

Points B and C are border points since they are within the radius eps of core point A and C, respectively.

Finding Noise Points:

Points D, E, F, and G are noise points since they do not belong to any core point's radius.

Merging Clusters:

The resulting clusters are {A, B, C} and {D, E, F, G}.

DBSCAN is a powerful clustering algorithm that can handle complex and irregularly shaped clusters. It is suitable for a wide range of applications, including image segmentation, customer segmentation, and anomaly detection. However, it may be sensitive to the parameter settings and the density distribution of the data.

Q) **Explain the following algorithms with solved example**

(i) K-Means and K-Medoids

A) Both K-Means and K-Medoids are clustering algorithms used to partition a dataset into K distinct clusters. They aim to minimize the distance between data points within the same cluster and maximize the distance between different clusters. However, they differ in how they define the cluster centers or representatives. Let's understand each algorithm with a solved example.

K-Means Algorithm:

K-Means is an iterative algorithm that assigns data points to clusters based on the proximity to the cluster centers. The steps involved in the K-Means algorithm are as follows:

Initialization:

Choose the number of clusters K and randomly initialize K cluster centers.

Assignment Step:

Assign each data point to the nearest cluster center based on the Euclidean distance or any other distance metric. This step forms K clusters.

Update Step:

Recalculate the cluster centers by taking the mean of the data points within each cluster. The new cluster centers become the updated representatives.

Repeat:

Repeat the Assignment and Update steps until convergence. Convergence occurs when the cluster centers no longer change significantly or when a maximum number of iterations is reached.

Let's consider the following dataset with six data points (A to F) and K = 2.

Dataset:

Point	x	y
A	2	10
B	2	5
C	8	4
D	5	8
E	7	5
F	6	4

Initialization:

Randomly choose two cluster centers: C1(2, 10) and C2(5, 8).

Assignment Step:

Calculate the Euclidean distance between each data point and the cluster centers and assign the points to the nearest cluster.

Point	Distance to C1	Distance to C2	Assigned Cluster
A	0	3.16	C1
B	5	3.16	C2
C	9.22	3.61	C2
D	4.24	0	C1
E	7.07	1	C1
F	7.28	1.41	C1

Update Step:

Recalculate the cluster centers by taking the mean of the data points within each cluster.

C1 = (A + D + E + F) / 4 = (2 + 5 + 7 + 6) / 4 = (20 / 4, 20 / 4) = (5, 5)

C2 = (B + C) / 2 = (2 + 8) / 2 = (5, 6)

Repeat:

Repeat the Assignment and Update steps until convergence.

After another iteration, the new assignments and cluster centers are:

Point	Distance to C1	Distance to C2	Assigned Cluster
A	0	3.61	C1
B	3.16	0	C2
C	7.28	0	C2
D	3.16	1.41	C1
E	5	1	C1
F	4.24	1.41	C1

Updated cluster centers:

C1 = (A + D + E + F) / 4 = (2 + 5 + 7 + 6) / 4 = (20 / 4, 20 / 4) = (5, 5)

C2 = (B + C) / 2 = (2 + 8) / 2 = (5, 6)

The algorithm converges as there is no change in the assignments and cluster centers. The final clusters are:

Cluster 1: {A, D, E, F}

Cluster 2: {B, C}

K-Medoids Algorithm:

K-Medoids is a variation of K-Means that uses actual data points as cluster representatives or medoids. Instead of calculating the mean of the data points, K-Medoids selects K representative points from the dataset. The steps involved in the K-Medoids algorithm are as follows:

Initialization:

Choose the number of clusters K and randomly select K data points as the initial medoids.

Assignment Step:

Assign each data point to the nearest medoid based on the distance metric (e.g., Euclidean distance).

Update Step:

For each cluster, evaluate the total distance between each data point and the medoids. Select the data point with the lowest total distance as the new medoid for that cluster.

Repeat:

Repeat the Assignment and Update steps until convergence, where there is no change in the assignments and medoids.

The K-Medoids algorithm follows a similar process as K-Means but uses actual data points as medoids instead of calculating the mean.

Both K-Means and K-Medoids are iterative algorithms that aim to minimize the within-cluster variance. They differ in terms of the measure used to define the cluster representatives. K-Means uses the mean, while K-Medoids uses actual data points. The choice between the two algorithms depends on the specific

characteristics of the dataset and the desired interpretability of the cluster representatives.

Attribute Selection Measures:

Attribute selection measures, also known as feature selection measures, are techniques used to evaluate and select the most relevant and informative features or attributes in a dataset. These measures help in identifying the subset of features that contribute the most to the prediction or classification task at hand. In simple terms, they help us determine which features are the most important in making accurate predictions or classifications.

They are classified into several types they are:

- **Information gain**: This measure calculates the amount of information that is gained by knowing the value of an attribute. The higher the information gain, the more important the attribute is for predicting the target variable.

  The expected information needed to classify a tuple in D is given by:

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

Where

Info(D) = entropy of D.

"Pi" is the nonzero probability that an arbitrary tuple in D belongs to class Ci

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

The term $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

- **Gini index**: This measure calculates the probability that a randomly chosen data point will be misclassified if it is classified according to the attribute's values. The lower the Gini index, the more important the attribute is for predicting the target variable.

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

- **Gain ratio:** This measure is a combination of information gain and Gini index. It is calculated by dividing information gain by the Gini index. The higher the gain ratio, the more important the attribute is for predicting the target variable.

$$Gain(A) = Info(D) - Info_A(D).$$

Types of Clustering:

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

Hierarichal clustering:

Hierarchical clustering is a popular algorithm used in data analysis and machine learning to group similar data points together based on their similarities or dissimilarities. It creates a hierarchy of clusters, organizing the data into a tree-like structure known as a dendrogram.

The process of hierarchical clustering can be summarized as follows:

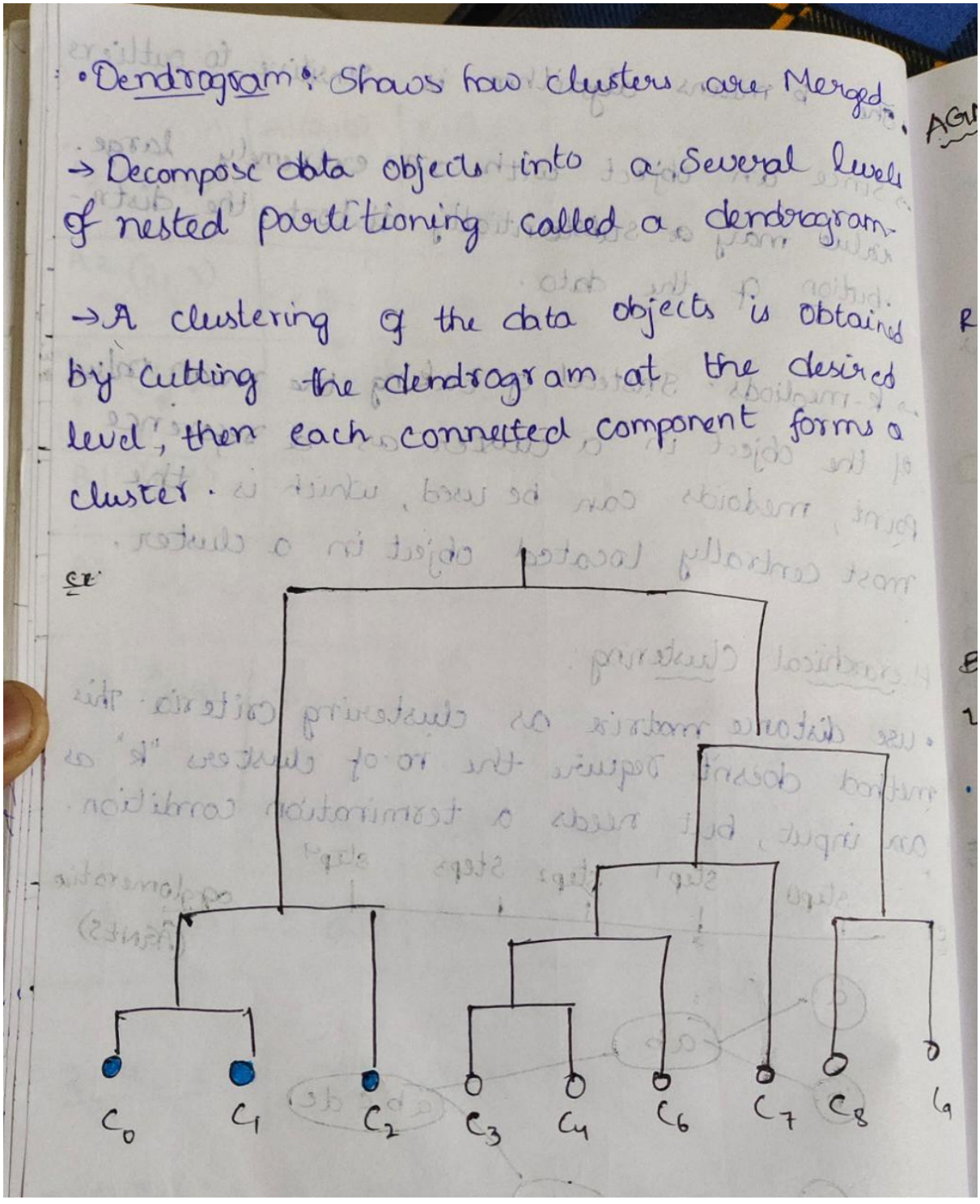
1. Initial Step: Each data point is considered as an individual cluster.

2. Distance Calculation: The algorithm calculates the distance or dissimilarity between each pair of data points. The choice of distance metric depends on the nature of the data (e.g., Euclidean distance for numerical data, Hamming distance for categorical data).

3. Agglomeration: The two closest data points or clusters are merged together to form a new cluster. This step is repeated iteratively until all data points or clusters are combined into a single cluster.

4. Dendrogram Construction: As the merging process continues, a dendrogram is built. The dendrogram represents the hierarchy of clusters and shows the order in which clusters are merged. The vertical axis of the dendrogram represents the dissimilarity between clusters.

5**. Cutting the Dendrogram:** To determine the number of clusters, a cut is made at a specific height on the dendrogram. The height represents a threshold or similarity level. The number of resulting branches below the cut represents the number of clusters.

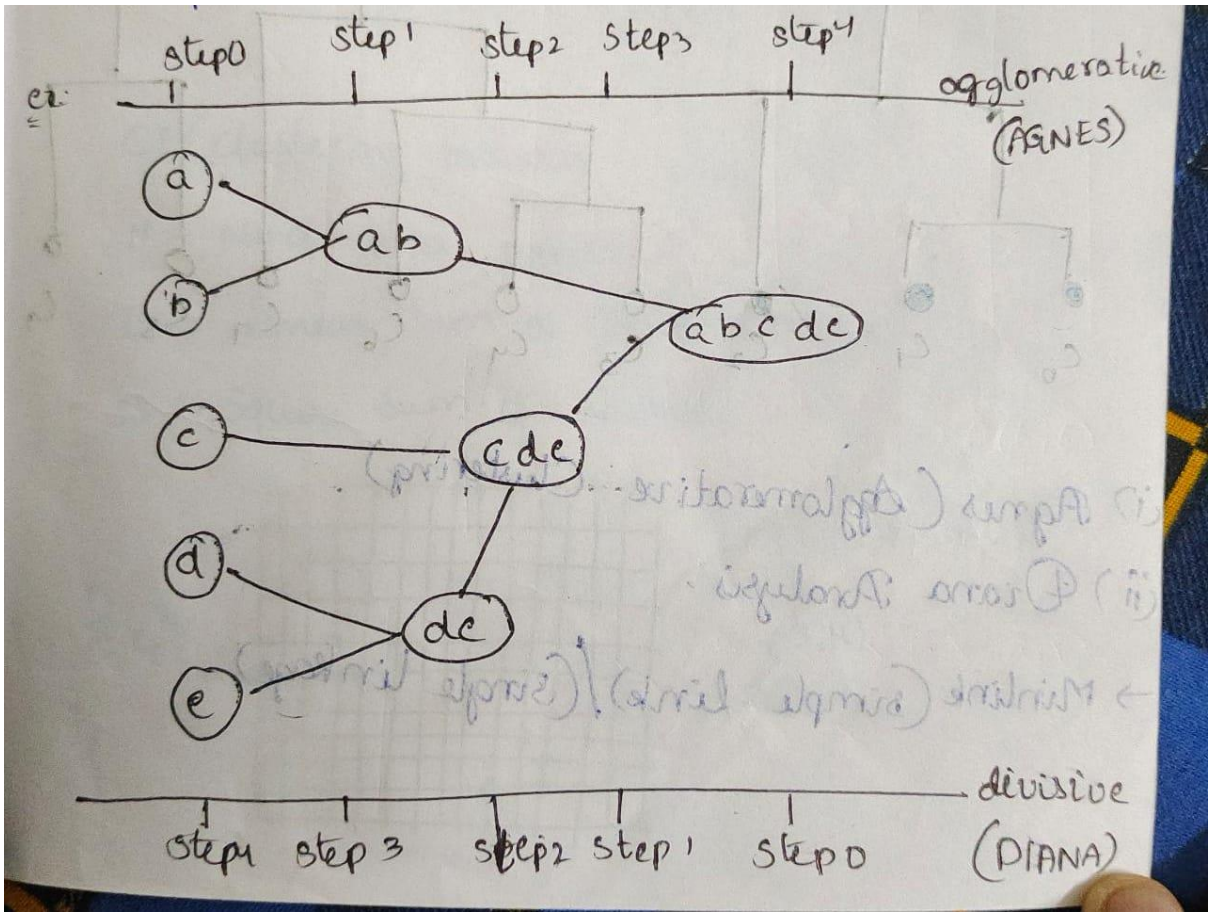


There are two types of hierarchical clustering approaches: agglomerative and divisive.

- **Agglomerative Hierarchical Clustering**: It starts with each data point as a separate cluster and gradually merges the closest clusters until

all data points belong to a single cluster. This bottom-up approach is computationally efficient and widely used.

- **Divisive Hierarchical Clustering**: It begins with all data points in a single cluster and splits them recursively into smaller clusters based on their dissimilarities. This top-down approach can be more computationally intensive and less commonly used.



Hierarchical clustering has several advantages. It does not require specifying the number of clusters in advance, as it generates a dendrogram that allows for visual interpretation and determination of the optimal number of clusters. It can also handle different types of data and is robust to noise and outliers.

However, hierarchical clustering can be computationally expensive for large datasets, and the choice of distance metric and linkage method (criteria for merging clusters) can affect the results. Additionally, it is a deterministic algorithm, meaning it will always produce the same clustering solution given the same input data.

Overall, hierarchical clustering provides a flexible and intuitive approach to discovering structure and patterns within data by organizing them into a hierarchy of clusters.

DBSCAN:

Density-based methods are a type of clustering algorithm that group data points based on their proximity and density. These methods identify areas of high-density regions and separate them from low-density regions. Density-based methods are particularly useful for discovering clusters with arbitrary shapes, handling noise and outliers, and scaling well to large datasets.

One popular density-based clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN works by defining a dense region as a cluster and a sparser region as noise. The key parameters for DBSCAN are the radius (eps) and the minimum number of points (minPts) required to form a dense

region.

Here's how DBSCAN works:

Finding Core Points:

DBSCAN starts by randomly selecting an unvisited data point and checks if it has at least minPts neighboring points within the radius eps. If so, the point is considered a core point, and all its neighbors are

added to its cluster.

Finding Border Points:

If a data point has fewer than minPts neighbors but belongs to the radius eps of a core point, it is considered a border point. Border points are assigned to the same cluster as their corresponding core point.

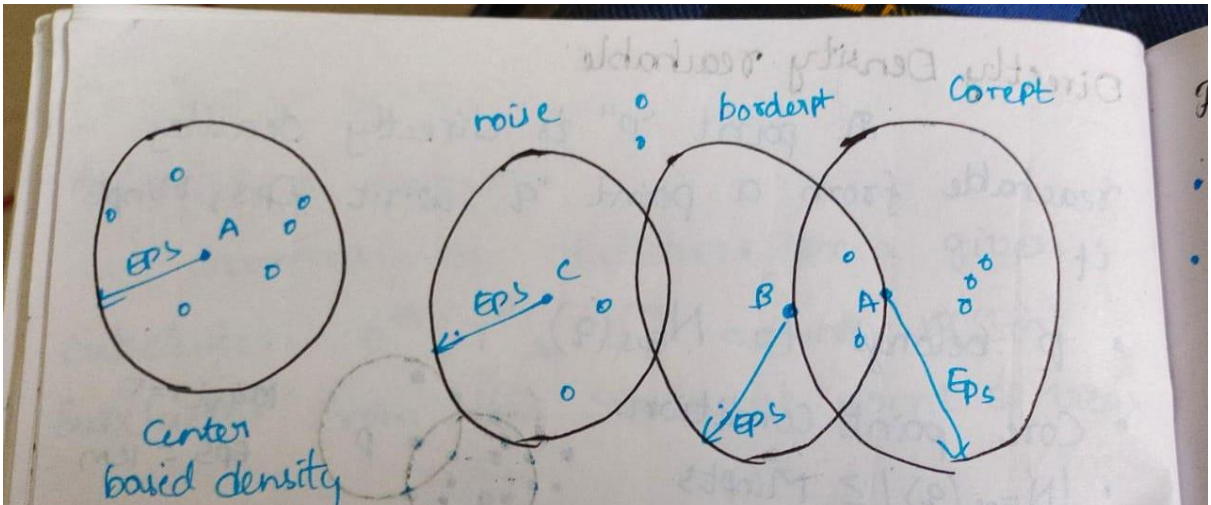
Finding Noise Points:

If a data point does not belong to any core point's radius, it is considered a noise point and is not assigned to any cluster.

Merging Clusters:

DBSCAN repeats the above steps until all points are visited. After the clusters are formed, they may contain outliers or noise. DBSCAN provides an additional step to merge clusters that are close enough, based on their minimum distance.

Example:



Rule-Based Classification

Using IF-THEN Rules for Classification

A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

“IF condition THEN conclusion.”

An example is rule R1,

R1: IF age =youth AND student =yes THEN buys computer = yes.

The “IF” part (or left side) of a rule is known as the rule antecedent or precondition. The “THEN” part (or right side) is the rule consequent. In the rule antecedent, the condition consists of one or more attribute tests (e.g., age = youth and student = yes) that are logically ANDed. The rule’s consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). R1 can also be written as

R1: (age = youth) ^ (student = yes)

(buys_computer = yes).

If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is satisfied (or simply, that the rule is satisfied) and that the rule covers the tuple.

A rule R can be assessed by its coverage and accuracy. Given a tuple, X, from a classlabeled data set,D, let ncovers be the number of tuples covered by R; n correct be the number of tuples correctly classified by R; and |D|1 be the number of tuples in D. We can define the coverage and accuracy of R as

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

Rule Extraction from a Decision Tree:

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent ("IF" part). The leaf node holds the class prediction, forming the rule consequent ("THEN" part).

Example 8.7 **Extracting classification rules from a decision tree.** The decision tree of Figure 8.2 can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from Figure 8.2 are as follows:

*R*1: IF *age* = *youth*	AND *student* = *no*	THEN *buys_computer* = *no*
*R*2: IF *age* = *youth*	AND *student* = *yes*	THEN *buys_computer* = *yes*
*R*3: IF *age* = *middle_aged*		THEN *buys_computer* = *yes*
*R*4: IF *age* = *senior*	AND *credit_rating* = *excellent*	THEN *buys_computer* = *yes*
*R*5: IF *age* = *senior*	AND *credit_rating* = *fair*	THEN *buys_computer* = *no*

■

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they are mutually exclusive and exhaustive. Mutually exclusive means that we cannot have rule conflicts here because no two rules will be triggered for the same tuple. (We have one rule per leaf, and any tuple can map to only one leaf.) Exhaustive means there is one rule for each possible attribute–value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter—they are unordered.

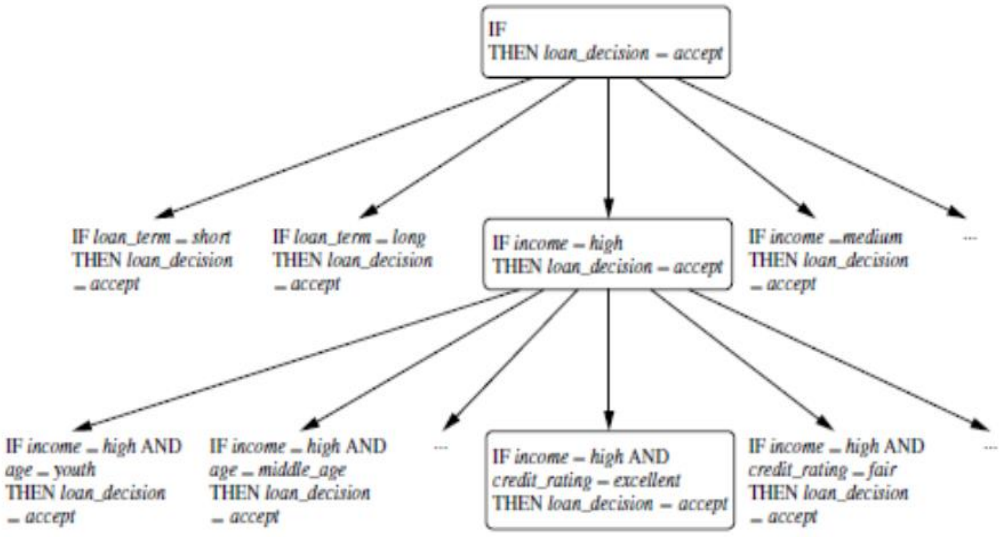


Figure 8.11 A general-to-specific search through rule space.

OLAP VS OLTP:

Feature	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Purpose	Supports complex analysis and decision-making based on historical data.	Supports day-to-day transactional operations and real-time data processing.
Data Type	Handles large volumes of historical data (typically read-intensive).	Handles smaller, current data sets (both read and write operations).
Database Design	Uses a multidimensional model (star or snowflake schema) for analytical queries.	Uses a relational model (normalized schema) for efficient transaction processing.
Query Complexity	Executes complex, ad-hoc queries involving aggregations, drill-downs, and calculations.	Performs simple, predefined queries (inserts, updates, deletes) efficiently.
Performance	Optimized for query performance and response time.	Optimized for transaction throughput and response time.
Data Latency	Data can have latency, as it involves processing historical information.	Data is up-to-date in real-time or near-real-time.
User Role	Primarily used by analysts, managers, and decision-makers.	Primarily used by operational staff, including employees, customers, and suppliers.

Feature	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Data Modification	Rarely modifies data, mostly focuses on analysis and reporting.	Frequently modifies data due to frequent transactions and updates.
Data Integrity	Emphasizes data accuracy and consistency, with less focus on strict concurrency controls.	Requires strong concurrency controls and ensures data integrity in a transactional environment.
Data Storage	Optimized for read-intensive operations, may involve denormalization for faster query performance.	Optimized for read and write operations, adhering to normalization principles.
Example Applications	Business intelligence, data mining, forecasting, and trend analysis.	E-commerce, banking transactions, inventory management, and order processing.