LECTURE NOTES ON

DATA WAREHOUSING & MINING(15A05602)

III B.TECH II SEMESTER

(JNTUA-R15)

NARAYANA
ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARAYANA ENGINEERING COLLEGE :: GUDUR
(AUTONOMOUS)
Approved by AICTE and Permanently Affiliated to JNTUA, Ananthapuramu, An ISO 9001:2015 Certified Institution, Recognized by UGC U/S 2(f) & 12(B)
Dhoorjati Nagar, Gudur, SPSR Nellore Dist.- 524101

NAAC

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech III-I Sem. (IT)	L	T	P	C
	3	1	0	3

15A05602 DATA WAREHOUSING & MINING

Course Objectives:

To know the basic concepts and principles of data warehousing and data mining Learn pre-processing techniques and data mining functionalities

Learn and create multidimensional models for data warehousing

Study and evaluate performance of Frequent Item sets and Association Rules

Understand and Compare different types of classification and clustering algorithms **Course Outcomes:**

Understand the basic concepts of data warehouse and data Mining Apply pre-processing techniques for data cleansing

Analyze and evaluate performance of algorithms for Association Rules Analyze Classification and Clustering algorithms

UNIT I

Introduction: Fundamentals of data mining, Data Mining Functionalities, Classification of Data Mining systems, Data Mining Task Primitives, Integration of a Data Mining System with a Database or a Data Warehouse System, Major issues in Data Mining. Data Preprocessing: Need for Preprocessing the Data, Data Cleaning, Data Integration and Transformation, Data Reduction, Discretization and Concept Hierarchy Generation.

UNIT II

Data Warehouse and OLAP Technology for Data Mining: Data Warehouse, Multidimensional Data Model, Data Warehouse Architecture, Data Warehouse Implementation, Further Development of Data Cube Technology, From Data Warehousing to Data Mining. Data Cube Computation and Data Generalization: Efficient Methods for Data Cube Computation, Further Development of Data Cube and OLAP Technology, Attribute-OrientedInduction.

UNIT III

Mining Frequent Patterns, Associations and Correlations: Basic Concepts, Efficient and Scalable Frequent Itemset Mining Methods, Mining various kinds of Association Rules, From Association Mining to Correlation Analysis, Constraint-Based Association Mining, Classification and Prediction: Issues Regarding Classification and Prediction, Classification by Decision Tree Induction, Bayesian Classification, Rule-Based Classification, Classification by Back propagation, Support Vector Machines, Associative Classification, Lazy Learners, Other Classification Methods, Prediction, Accuracy and Error measures, Evaluating the accuracy of a Classifier or a Predictor, Ensemble Methods.

UNIT IV

Cluster Analysis Introduction :Types of Data in Cluster Analysis, A Categorization of Major Clustering Methods, Partitioning Methods, Hierarchical Methods, Density-Based Methods, Grid-Based Methods, Model-Based Clustering Methods, Clustering High- Dimensional Data, Constraint-Based Cluster Analysis, OutlierAnalysis.

UNIT V

Mining Streams, Time Series and Sequence Data: Mining Data Streams, Mining Time- Series Data, Mining Sequence Patterns in Transactional Databases, Mining Sequence Patterns in Biological Data, Graph Mining, Social Network Analysis and Multi relational Data Mining, Mining Object, Spatial, Multimedia, Text and Web Data: Multidimensional Analysis and Descriptive Mining of Complex Data Objects, Spatial Data Mining, Multimedia Data Mining, Text Mining, Mining the World Wide Web.

TEXT BOOKS:

1. Data Mining: Concepts and Techniques, Jiawei Han and Micheline Kamber, Morgan Kaufmann Publishers, Elsevier, Second Edition, 2006.
2. Introduction to Data Mining – Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Pearson Education.

REFERENCES:

1. Data Mining Techniques, Arun KPujari, Second Edition, Universities Press.

2. **Data Warehousing in the Real World, Sam Aanhory& Dennis Murray Pearson EdnAsia.**
3. **Insight into Data Mining, K.P.Soman, S.Diwakar,V.Ajay, PHI,2008.**

UNIT-1

INTRODUCTION

What Is Data Mining?

Data mining refers to extracting or mining knowledge from large amountsof data. The term is actually a misnomer. Thus, d ata miningshould have been more appropriately named as knowledge mining which emphasis on mining from large amounts of data.

It is the computational process of discovering patterns in large data sets involving me thods at the intersection of artificial intellige nce, machine learning, statistics, and database systems. The overall goal of the data mini ng process is to extract information from a data set and transform it into an understandable structure for further use.

The key properties of data mining are

Automatic discovery of patterns

Prediction of likely outcomes

Creation of actionable infor mation

Focus on large datasets and databases

The Scope of Data Mining

Data mining derives its name fro m the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exact ly where the value resides. Given databases of sufficient size and quality, data mining techno logy can generate new business opportunitie s by providing these capabilities:

Automated prediction of trend s and behaviors. Data mining automates the p rocess of finding predictive information in large d atabases. Questions that traditionally required extensive hands-on analysis can now be answered di rectly from the data — quickly. A typical examp le of a predictive problem is targeted marketing. Dat a mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a p opulation likely to respond similarly to given events.

Automated discovery of previouslly unknown patterns. Data mining tools sweep thro ugh databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifyi ng anomalous data that could represent data entry keyi ng errors.

Tasks primitives of Data M ining:

Data mining involves six common c lasses of tasks:

Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be inter esting or data errors that require further investigation .

Association rule learning (Dependency modelling) – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this inform ation for marketing purposes. This is sometimes referred to as market basket analysis.

Clustering – is the task of discovering groups and structures in the data that a re in some way or another "similar", without u sing known structures in the data.

Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attem pt to classify an e-mail as "legitimate" or as "spam".

Regression – attempts to find a function which models the data with the least error.

Summarization – providing a more compact representation of the data set, including visualization and report generation.

Architecture of Data Mini ng

A typical data mining system may h ave the following major components.

1. **Knowledge Base:**
   This is the domain kno wledge that is used to guide the search or evaluate th e interestingness of resulting patterns. Su ch knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern's interestingness based on its unexpectedness, may also be included. Other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

2. **Data Mining Engine:**

   This is essential to the data mining system and ideally consists ofa set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

3. **Pattern Evaluation Module:**

   This component typically employs interestingness measures interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining processs as to confine the search to only the interesting patterns.

4. **User interface:**

   This module communicates between users and the data mining system,allowing the user to interact with the system by specifying a data mining query ortask, providing information to help focus the search, and performing exploratory datamining based on the intermediate data mining results. In addition, this componentallows the user to

browse database and data warehouse schemas or data structures,evaluate mined patterns, and visualize the patterns in different forms.

Data Mining Process:

Data Mining is a process of discovering various models, summaries, and derived values from a given collection of data.

The general experimental procedure adapted to data-mining problems involves the following steps:

5. **State the problem and formulate the hypothesis**

Most data-based modeling studies are performed in a particular application domain. Hence, domain-specific knowledge and experience are usually necessary in order to come up with a meaningful problem statement. Unfortunately, many application studies tend to focus on the data-mining technique at the expense of a clear problem statement. In this step, a modeler usually specifies a set of variables for the unknown dependency and, if possible, a general form of this dependency as an initial hypothesis. There may be several hypotheses formulated for a single problem at this stage. The first step requires the combined expertise of an application domain and a data-mining model. In practice, it usually means a close interaction between the data-mining expert and the application expert. In successful data-mining applications, this cooperation does not stop in the initial phase; it continues during the entire data-mining process.

6. **Collect the data**

This step is concerned with how the data are generated and collected. In general, there are two distinct possibilities. The first is when the data-generation process is under the control of an expert (modeler): this approach is known as a designed experiment. The second possibility is when the expert cannot influence the data- generation process: this is known as the observational approach. An observational setting, namely, random data generation, is assumed in most data-mining applications. Typically, the sampling

distribution is completely unknown after data are collected, or it is partially and implicitly given in the data-collection procedure. It is very important, however, to understand how data collection affects its theoretical distribution, since such a priori knowledge can be very useful for modeling and, later, for the final interpretation of results. Also, it is important to make sure that the data used for estimating a model and the data used later for testing and applying a model come from the same, unknown, sampling distribution. If this is not the case, the estimated model cannot be successfully used in a final application of the results.

7. Preprocessing the data

In the observational setting, data are usually "collected" from the existing databses, data warehouses, and data marts. Data preprocessing usually includes at least two common tasks:

1. Outlier detection (and removal) – Outliers are unusual data values that are not consistent with most observations. Commonly, outliers result from measurement errors, coding and recording errors, and, sometimes, are natural, abnormal values. Such nonrepresentative samples can seriously affect the model produced later. There are two strategies for dealing with outliers:

a. Detect and eventually remove outliers as a part of the preprocessing phase, or
b. Develop robust modeling methods that are insensitive to outliers.

2. Scaling, encoding, and selecting features – Data preprocessing includes several steps such as variable scaling and different types of encoding. For example, one feature with the range [0, 1] and the other with the range [−100, 1000] will not have the same weights in the applied technique; they will also influence the final data-mining results differently. Therefore, it is recommended to scale them and bring both features to the same weight for further analysis. Also, application-specific encoding methods usually achieve

dimensionality reduction by providing a smaller number of informative features for subsequent data modeling.

These two classes of preprocessing tasks are only illustrative examples of a large spectrum of preprocessing activities in a data-mining process.

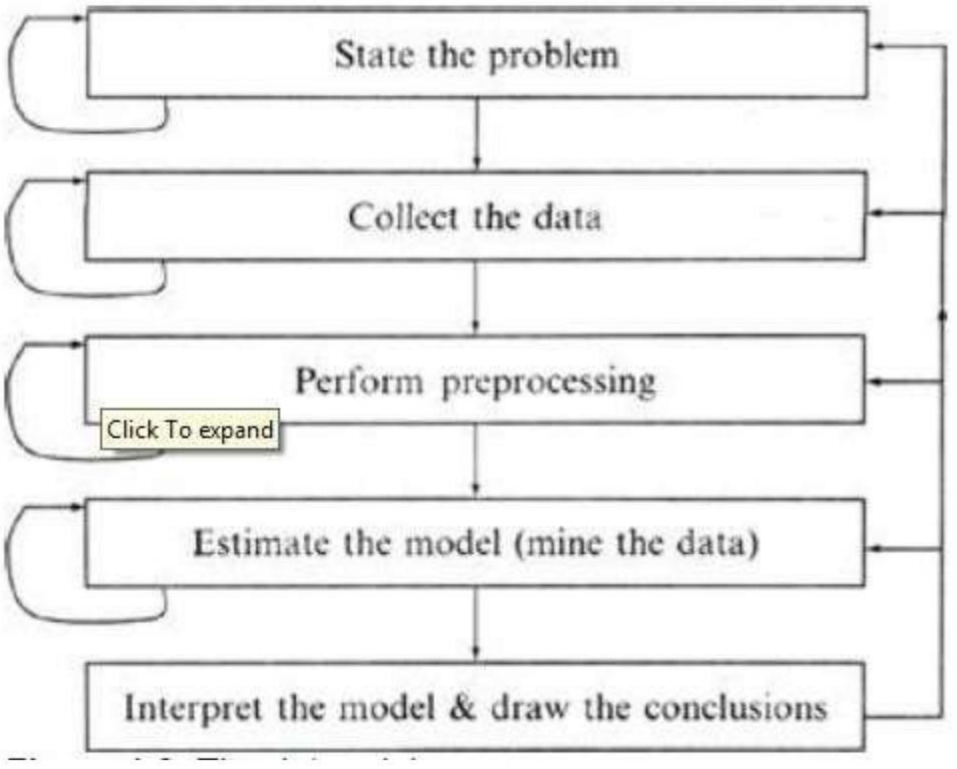
Data-preprocessing steps should not be considered completely independent from other data-mining phases. In every iteration of the data-mining process, all activities, together, could define new and improved data sets for subsequent iterations. Generally, a good preprocessing method provides an optimal representation for a data-mining technique by incorporating a priori knowledge in the form of application-specific scaling and encoding.

4. Estimate the model

The selection and implementation of the appropriate data-mining technique is the main task in this phase. This process is not straightforward; usually, in practice, the implementation is based on several models, and selecting the best one is an additional task. The basic principles of learning and discovery from data are given in Chapter 4 of this book. Later, Chapter 5 through 13 explain and analyze specific techniques that are applied to perform a successful learning process from data and to develop an appropriate model.

5. Interpret the model and draw conclusions

In most cases, data-mining models should help in decision making. Hence, such models need to be interpretable in order to be useful because humans are not likely to base their decisions on complex "black-box" models. Note that the goals of accuracy of the model and accuracy of its interpretation are somewhat contradictory. Usually, simple models are more interpretable, but they are also less accurate. Modern data-mining methods are expected to yield highly accurate results using highdimensional models. The problem of interpreting these models, also very important, is considered a separate task, with specific techniques to validate the results. A user does not want hundreds of pages of n umeric results. He does not understand them; he cannot summarize, interpret, and use them for s uccessful decision making.


State the problem
Collect the data
Perform preprocessing
Click To expand
Estimate the model (mine the data)
Interpret the model & draw the conclusions

The Data mining Process

Classification of Data mining Systems:

The data mining system can be classified according to the following criteria:

- **Database Technology**
- **Statistics**
- **Machine Learning**
- **Information Science**
- **Visualization**
- **Other Disciplines**

Some Other Classification Criteria :

Classification according to kind of data ases mined
Classification according to kind of know ledge mined
Classification according to kinds of tech niques utilized   Classification according to applications adapted

1. Classification according to kind of databases mined

We can classify the data mining system according to kind of databases mined. Databas e system can be classified according t o different criteria such as data models, types of da ta etc. And the data mining system can be classified accordingly. For example if we classif y the database according to data model then we may have a relational, transactional, object-relational, or data warehouse mining system.

2. Classification according to kind of knowledge mined

We can classify the data mining system according to kind of knowledge mined. It is means data mining system are classi ied on the basis of functionalities such as:

- **Outlier Analysis**
- **Evolution Analysis**

3. Classification according to kinds of techniques utilized

We can classify the data mining system according to kind of techniques used. We can describes these techniques according to degree of user interaction involved or the methods of analysis employed.

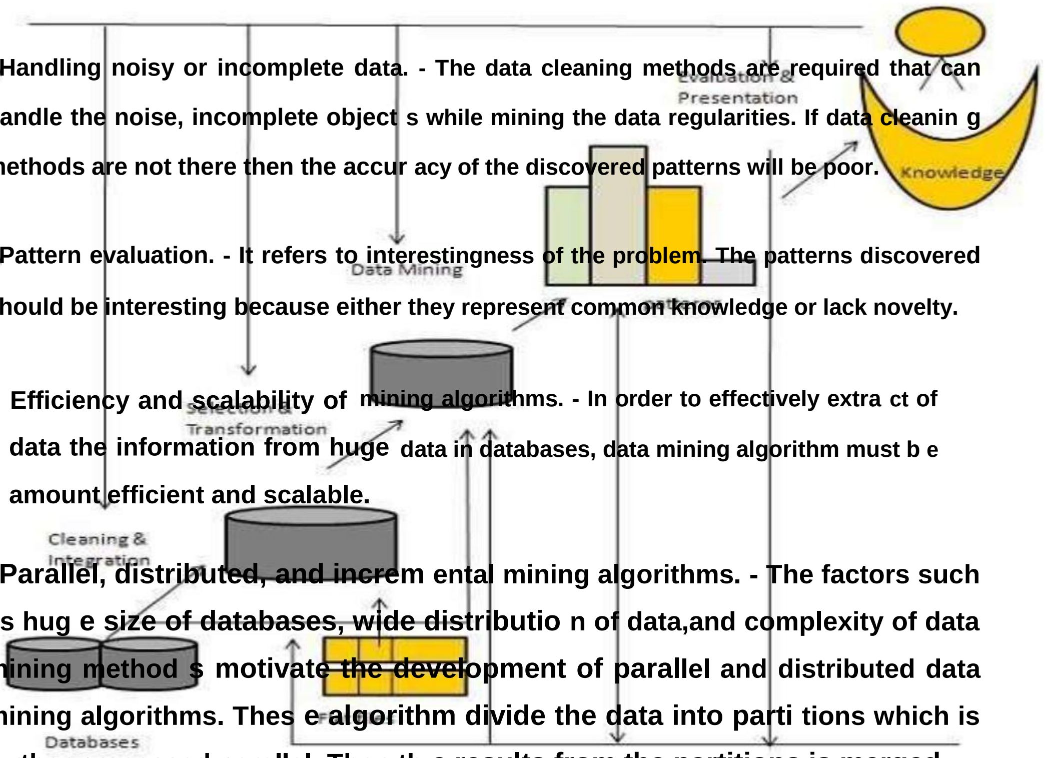
4. Classification according to appli cations adapted

We can classify the data mining system according to application adapted. These applications are as follows:

- **Finance**
- **Telecommu nications**
- **DNA**

- **Data mining query languages and ad hoc data mining. - Data Mining Query language that allows the user to describe ad hoc mining tasks, should be integrated with a** data warehouse query language and optimized for efficient and flexible data mining.

- **Presentation and visualization of data mining results. - Once the patterns ar e discovered it needs to be expressed in high level languages, visual representations. Th is representations should be easily und erstandable by the users.**



- **Handling noisy or incomplete data. - The data cleaning methods are required that can handle the noise, incomplete object s while mining the data regularities. If data cleanin g methods are not there then the accur acy of the discovered patterns will be poor.**

- **Pattern evaluation. - It refers to interestingness of the problem. The patterns discovered should be interesting because either they represent common knowledge or lack novelty.**

- **Efficiency and scalability of mining algorithms. - In order to effectively extra ct of data the information from huge data in databases, data mining algorithm must b e amount efficient and scalable.**

- **Parallel, distributed, and increm ental mining algorithms. - The factors such as hug e size of databases, wide distributio n of data,and complexity of data mining method s motivate the development of parallel and distributed data mining algorithms. Thes e algorithm divide the data into parti tions which is further processed parallel. Then th e results from the partitions is merged.**

Data Warehouse:

A data warehouse is a subject-orient ed, integrated, time-variant and non-volatile collection of data in support of management's decision making process. Subject-Oriented: A data warehouse can be used to analyze a particular subject area. Fo r example, "sales" can be a particular ubject.

Integrated: A data warehouse integ rates data from multiple data sources. For exampl e, source A and source B may have different ways of identifying a product, but in a datta warehouse, there will be only a single way of identifying a product.

Time-Variant: Historical data is ke pt in a data warehouse. For example, one can retriev e data from 3 months, 6 months, 12 m onths, or even older data from a data warehouse. Th is contrasts with a transactions system, where often only the most recent data is kept. Foor example, a transaction system may hold the most recent address of a customer, where a data warehouse can hold all addresses associated with a customer.

Non-volatile: Once data is in the data warehouse, it will not change. So, historical data in a data warehouse should never be altered.

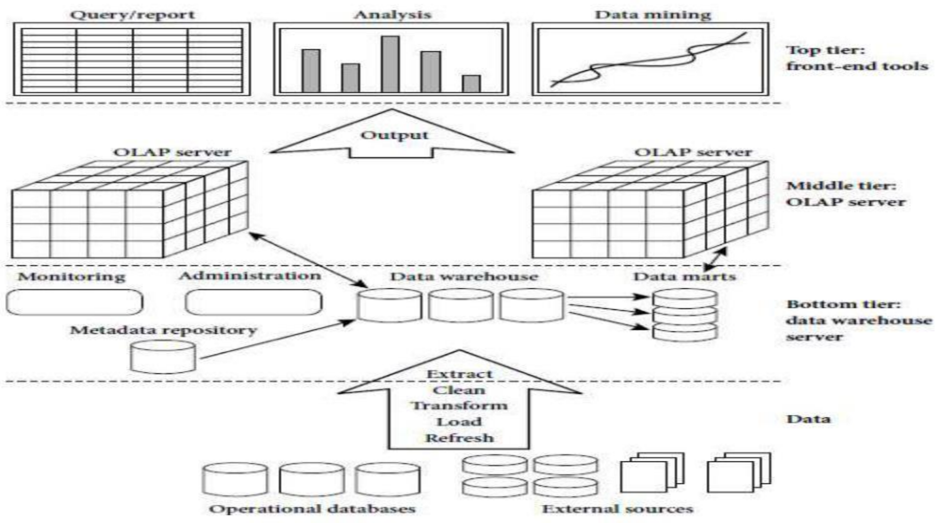
Data Warehouse Design Pr ocess:

A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both.

- The top-down approach starts with the overall design and planning. It is useful in cases where the technology is mature and well known, and where the businesss problems that must be solved are clear and well understood.

- The bottom-up approach starts with experiments and prototypes. This is useful in th e early stage of business modeling and technology development. It allows an organization to move forward at considerably less expense and to evaluate th e benefits of the technology befor e making significant commitment

A Three Tier Data Wareho use Architecture:



Tier-1:

The bottom tier is a war ehouse database server that is almost always a relationaldatabase system. B ack-end tools and utilities are used to feed data into the bottomtier from operational databases or other external sources (such a s customer profileinformation provided by external consultants). These tools and utilities performdataextraction, cleaning, and transformation (e.g., to merg e similar data from differents ources into a unified format), as well as load and refresh functions to update thedata warehouse . The data are extracted using application programinterfaces known as gateways. A gateway is Examplesoof gateways include ODBC (Open Database Connection) and OLEDB (Open Linkingand Embedding for Databases) by Microsoft and JDBC (Java Databas e Connection).

This tier also contains a metadata repository, which stores information aboutth e data warehouse and its contents.

Tier-2:

The middle tier is an OLA server that is typically implemented using either a relational OLAP (ROLAP) model or a multidimensional OLAP.

- **OLAP model is an extend ed relational DBMS thatmaps operations on** multidimensional data to standard relational operations.

- A multidimensional OLA P (MOLAP) model, that is, a special-purpose server that directly implements m ultidimensional data and operations.

Tier-3:

The top tier is a front-end cl ient layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on). supported by the underlying DBMS andallows client programs to generatte SQL code to be executed at a server.

Data Warehouse Models:

There are three data warehouse mod els.

1. Enterprise warehouse:

- An enterprise warehouse co llects all of the information about subjects spanning the entire organization.
- It provides corporate-wide data integration, usually from one or more operation al systems or external information providers, and is cross-functional in scope.
- It typically contains detailed data aswell as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.
- An enterprise data wareho use may be implemented on traditional mainframes, computer superservers, or parallel architecture platforms. It requires extensiv e business modeling and may take years to design and build.

2. Data mart:

- A data mart contains a subset of corporate-wide data that is of value to aspecifiic group of users. The scope is confined to specific selected subjects. For example,,a marketing data mart may co nfine its subjects to customer, item, and sales.

- Depending on the source of data, data marts can be categorized as independe nt ordependent. Independent data marts are sourced fromdata captured fromone or moreoperational systems or external information providers, or fromda ta generated locallywithin a particular department or geographic area. Depende nt data marts are sourceddirectly from enterprise data warehouses.

3. **Virtual warehouse:**

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

Meta Data Repository:

Metadata are data about data.Whe n used in a data warehouse, metadata are the datta thatdefine warehouse objects. Metad ata are created for the data names anddefinitions of the given warehouse. Additional metadata are created and captured fortimestamping an y extracted data, the source of the extracted data, and missing fieldsthat have been adde d by data cleaning or integration processes.

A metadata repository should contai n the following:

- **A description of the str ucture of the data warehouse, which includes th e warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.**

- **Operational metadata, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archive d, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).**

- **The algorithms used for sum marization, which include measure and dimension.**

Data Preprocessing:

Data Integration:

It combines datafrom multiple sources into a coherent data store, as in data warehousing.

These sourcesmay include multiple databases, data cubes, or flat files.

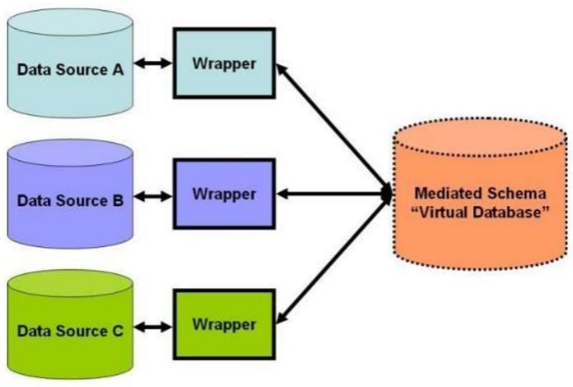
The data integration systems are formally defined as

triple<G,S,M> Where G: The global schema

S:Heterogeneous source of schemas

M: Mapping between the queries of source

and global schema



Issues in Data integration:

1. **Schema integration and object matching:**

   How can the data analyst or the computer be sure that customer id in one database and customer number in another reference to the same attribute.

2. **Redundancy:**

   An attribute (such as annual revenue, forinstance) may be redundant if it can be derived from another attribute or set ofattributes. Inconsistencies in attribute or dimension naming can also cause redundanciesin the resulting data set.

3. **detection and resolution of datavalue conflicts:**

   For the same real-world entity, attribute values fromdifferent sources may differ.

Data Transformation:

In data transformation, the data are tr ansformed or consolidated into forms appropriatefor mining.

Data transformation can involve the following:

- Smoothing, which works to remove noise from the data. Such technique s includebinning, regression, a nd clustering.
- Aggregation, where summ ary or aggregation operations are applied to the dat a. For example, the daily sales data may be aggregated so as to compute monthly and annualtotal amounts. T his step is typically used in constructing a data cub e for analysis of the data at mu ltiple granularities.
- Generalization of the data, where low-level or —primitive‖ (raw) data ar e replaced byhigher-level con cepts through the use of concept hierarchies. Foor example, categoricalattributes, like street, can be generalized to higher-lev el concepts, like city or country .
- Normalization, where the attribute data are scaled so as to fall within a small specifiedrange, such as 1:0 t o 1:0, or 0:0 to 1:0.
- Attribute construction (or feature construction),wherenewattributes ar e constructedand added from the given set of attributes to help the mining process.

Data Reduction:

Data reduction techniques can be ap plied to obtain a reduced representation of thedata s et that ismuch smaller in volume, yet closely maintains the integrity of the originaldat a. That is, mining on the reduced data set should be more efficient yet produce thesame (o r almost the same) analytical results.
Strategies for data reduction include the following:

- Data cube aggregation, w ere aggregation operations are applied to the data in theconstruction of a data cub e.
- Attribute subset selection, where irrelevant, weakly relevant, or redunda nt attributesor dimensions may be detected and removed.
- Dimensionality reduction, where encoding mechanisms are used to reduce th e dataset size.
- Numerosityreduction,where the data are replaced or estimated by alternativ e,

smallerdata representations such as parametric models (which need store only the modelparameters instead of the actual data) or nonparametric methods such as clustering,sampling, and the use of histograms.

- Discretization and concept hierarchy generation,where rawdata values for attributesare replaced by ranges or higher conceptual levels. Data discretization is a form ofnumerosity reduction that is very useful for the automatic generation of concept hierarchies.Discretization and concept hierarchy generation are powerful tools for datamining, in that they allow the mining of data at multiple levels of abstraction.

Part A: (2 Marks)

1.Define Data mining.

It refers to extracting or “mining” knowledge from large amount of data. Data mining is a process of discovering interesting knowledge from large amounts of data stored either, in database, data warehouse, or other information repositories.

2. Give some alternative terms for data mining.

□□Knowledge mining
□□Knowledge extraction
□□Data/pattern analysis.
□□Data Archaeology
□□Data dredging

3.What is KDD.

KDD-Knowledge Discovery in Databases.

4.What are the steps involved in KDD process.

□□Data cleaning
□□Data Mining
□□Pattern Evaluation
□□
□□Data Integration
□□Data Selection
□□Data Transformation

5. What is the use of the knowledge base ?

Knowledge base is domain knowledge that is used to guide search or evaluate the interestingness of resulting pattern. Such knowledge can include concept hierarchies used t o organize attribute /attribute values in to different levels of abstraction.

6. Mention some of the data mining techniques.

Statistics

Machine learning

Decision Tree

Hidden markov models Artifici al Intelligence Genetic Algorithm Meta learning

7. Give few statistical techniques.

Point Estimation

Data Summarization Bayesian Techniques Testing Hypothesis Correlation Regression

8. What is meta learning.

Concept of combining the pred ictions made from multiple models of data mining a nd analyzing those predictions to formulate a new and previously unknownprediction.

GUI Pattern Evaluation

Database or Data warehouse
server DB DW

9. Define Genetic algorithm.

Search algorithm.enables us to locate optimal binary string by processing an initial random population of binary strings by performing operations such as artificial mutation , crossover and selection.

10. What is the purpose of Data mining Technique?

It provides a way to use various data mining tasks.

11.Define Predictive model.

It is used to predict the values of data by making use of known results from a different set of sample data.

12.Data mining tasks that are belongs to predictive model

Classification
Regression
Time series analysis

13.Define descriptive model

It is used to determine the patterns and relationships in a sample data. Data mining tasks that belongs to descriptive model:
Clustering
Summarization Association rules Sequence discovery

14. Define the term summarization?

The summarization of a large chunk of data contained in a web page or a document.
Summarization = caharcterization=generalization

15. List out the advanced database systems.

Extended-relational databases

Object-oriented databases Deductive databases Spatial databases Temporal databases
Multimedia databases Active databases Scientific databases Knowledge databases

16 .Classifications of Data mining systems. Based on the kinds of databases mined: According to model

_Relational mining system
_ Transactional mining system
_ Object-oriented mining system
_ Object-Relational mining system
_ Data warehouse mining system o Types
of Data _ Spatial data mining system

_ Time series data mining system
_ Text data mining system
_ Multimedia data mining system

17. what are the functionalities of data mining?

_ Characterization
_ Discrimination
_ Association
_ Classification
_ Clustering
_ Outlier analysis
_ Evolution analysis

18. According to levels of abstraction of the knowledge minedhow many tuypes are there?

_ Generalized knowledge (High level of abstraction)
_ Primitive-level knowledge (Raw data level)

19. Describe challenges to data mining regarding data mining methodology and user interaction issues.

Mining different kinds of knowledge in databases
Interactive mining of knowledge at multiple levels of abstraction Incorporation of background knowledge
Data mining query languages and ad hoc data mining Presentation and visualization of data mining results Handling noisy or incomplete data
Pattern evaluation

20. Describe challenges to data mining regarding performance issues.

Efficiency and scalability of data mining algorithms Parallel, distributed, and incremental mining algorithms

Part B: (10 Marks)

1. **Define the datamining and their functionalities?**
2. **Define data mining. On what kind of data, data mining can be performed?**
3. **Compare and contrast operational database systems with data warehouse.**
4. **What is the importance of data marts in data warehouse?**
5. **Discuss about a three-tier data warehouse architecture?**
6. **What are Major issues in Data Mining ?**
7. **What is Data Preprocessing and explain the Need for Preprocessing in the Data minig?**
8. **Explain briefly about Data Cleaning?**
9. **What is Data Integration and Transformation explain in detail?**
10. **Explain briefly about Concept Hierarchy Generation with neat diagram?**

UNIT-2

OLAP(Online analytical Processing)

Introduction and Need for Data Warehouse

The construction of data warehouses, which involves data cleaning and data integration, can be viewed as an important preprocessing step for data mining. Moreover, data warehouses provide on-line analytical processing (OLAP) tools for the interactive analysis of multidimensional data of varied granularities, which facilitates effective data mining. Furthermore, many other data mining functions such as classification, prediction, association, and clustering, can be integrated with OLAP operations to enhance interactive mining of knowledge at multiple levels of abstraction.

Hence, data warehouse has become an increasingly important platform for data analysis and online analytical processing and will provide an effective platform for data mining. Such an overview is essential for understanding data mining technology.

What is Data Warehouse?

Defined in many different ways:

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. According to W. H. Inmon, a leading architect in the construction of data warehouse systems, “a data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process."

Subject Oriented: Data that gives information about a particular subject instead of about a company's ongoing operations.

Integrated: Data that is gathered into the data warehouse from a variety of sources and merged into a coherent whole.

Time-variant: All data in the data warehouse is identified with a particular time period.

Non-volatile: Data is stable in a data warehouse. More data is added but data is never removed. This enables management to gain a consistent picture of the business.

Data warehousing is defined as the process of constructing and using data warehouses. The construction of a data warehouse requires data integration, data cleaning, and data consolidation. The utilization of a data warehouse often necessitates a collection of decision support technologies. This allows \knowledge workers" (e.g., managers, analysts)

(3) Analyzing operations and looking for sources of profit, and

(4) managing the customer relationships, making environmental corrections, and managing the cost of corporate assets.

Data warehousing is also very useful from the point of view of heterogeneous database integration. Many organizations typically collect diverse kinds of data and maintain large databases from multiple, heterogeneous, autonomous, and distributed information sources. To integrate such data, and provide easy and efficient access to it is highly desirable, yet challenging. Much effort has been spent in the database industry and research community towards achieving this goal.

Data Warehousing Architecture

The design of a data warehouse: A business analysis framework

First, having a data warehouse may provide a competitive advantage by presenting relevant information from which to measure performance and make critical adjustments in order to help win over competitors.

Second, a data warehouse can enhance business productivity since it is able to quickly and efficiently gather information which accurately describes the organization. Third, a data warehouse facilitates customer relationship marketing since it provides a consistent view of customers and items across all lines of business, all departments, and all markets.

Finally, a data warehouse may bring about cost reduction by tracking trends, patterns, and exceptions over long periods of time in a consistent and reliable manner.

To design an effective data warehouse one needs to understand and analyze business needs, and construct a business analysis framework. The construction of a large and complex information system can be viewed as the construction of a large and complex building, for which the owner, architect, and builder have different views.

Four different views regarding the design of a data warehouse must be considered: the top-down view, the data source view, the data warehouse view, and the business query view.

The top-down view allows the selection of the relevant information necessary for the data warehouse. This information matches the current and coming business needs. The data source view exposes the information being captured, stored, and managed by operational systems. This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables. Data sources are often modeled by traditional data modeling techniques, such as the entity-relationship model or CASE (Computer Aided Software Engineering) tools.

The data warehouse view includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.

Finally, the business query view is the perspective of data in the data warehouse from the view point of the end-user.

The process of data warehouse design

A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both.

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

Various kinds of data warehouse design tools are available. Data warehouse development tools provide functions to define and edit metadata repository contents such as schemas, scripts or rules, answer queries, output reports, and ship metadata to and from relational database system catalogues. Planning and analysis tools study the impact of schema changes and of refresh performance when changing refresh rates or time windows.

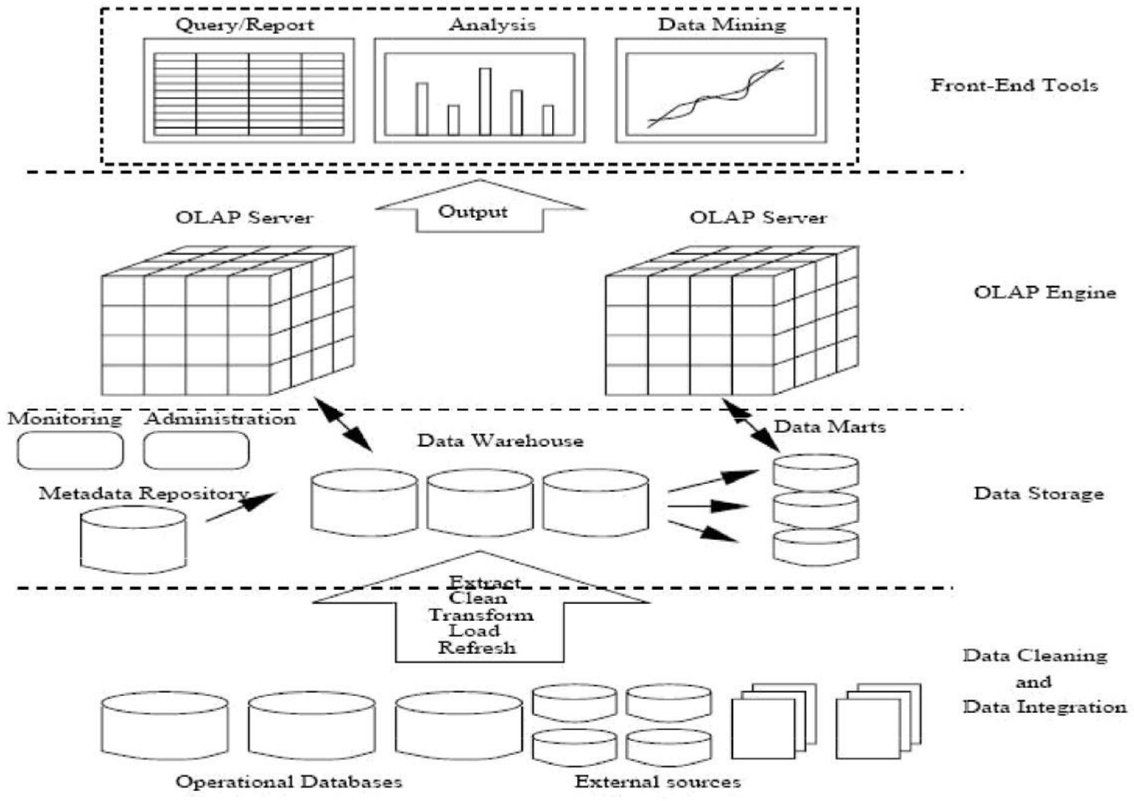
Three-tier data warehouse architecture

Data warehouses often adopt three-tier architecture, as presented in Fig The bottom tier is a ware-house database server which is almost always a relational database system. The middle tier is an OLAP server which is typically implemented using either

1. A Relational OLAP (ROLAP) model, i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations; or

2. A Multidimensional OLAP (MOLAP) model, i.e., a special purpose server that directly implements multidimensional data and operations. The top tier is a client, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

Figure : three-tier data warehouse architecture



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

Enterprise warehouse:

An enterprise warehouse collects all of the information about subjects spanning the entire organization. It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cr oss-functional in scope. It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabbytes, or beyond.

An enterprise data warehouse may be implemented on traditional mainframes, UNI X super servers, or parallel architecture platforms. It req uires extensive business modeling and may take yea rs to design and build

Data mart:

A data mart contains a subset of corporate-wide data that is of value to a specific gr oup of users. The scope is confined to specific, selected su bjects. For example, a marketing data mart may con fine its subjects to customer, item, and sales. The data c ontained in data marts tend to be summarized. Data marts are usually implemented on low cost departmental servers that are UNIX-, Windows/NT-, or OS/2-base d. The implementation cycle of a data mart is ore likely to be measured in weeks rather than months or years.

However, it may involve compl ex integration in the long run if its design and planni ng were not enterprise-wide. Depending on the sour ce of data, data marts can be categorized into the fol lowing two classes:

- Independent data marts are sourced from data captured from one or more operationa l systems or external information pro viders, or from data generated locally within a particular department or geographi c area.
- Dependent data marts are sourc ed directly from enterprise data warehouses.

Virtual warehouse:

A virtual warehouse is a set of v iews over operational databases. For efficient query processing, only some of the possible summary views m y be materialized. A virtual warehouse is easy to bu ild but requires excess capacity on operational database servers.

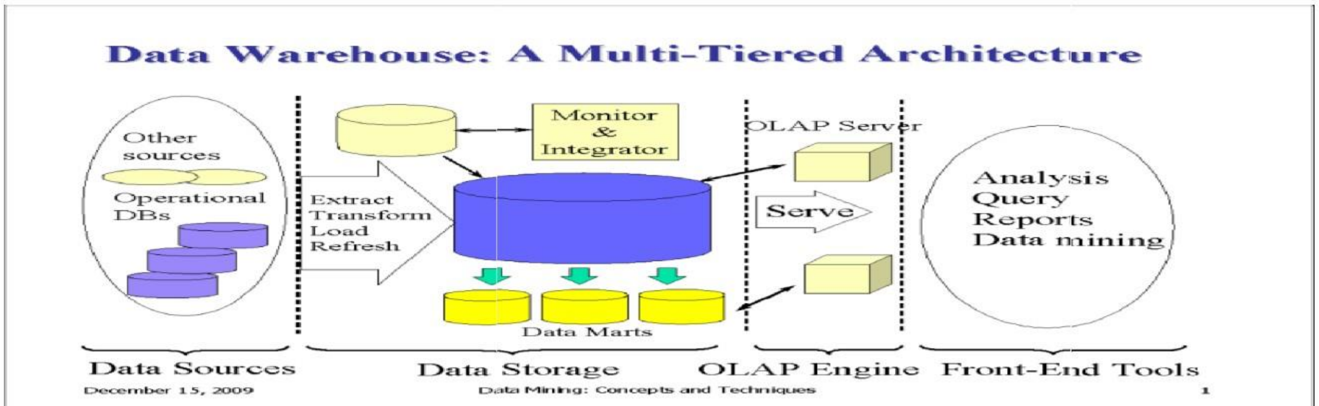
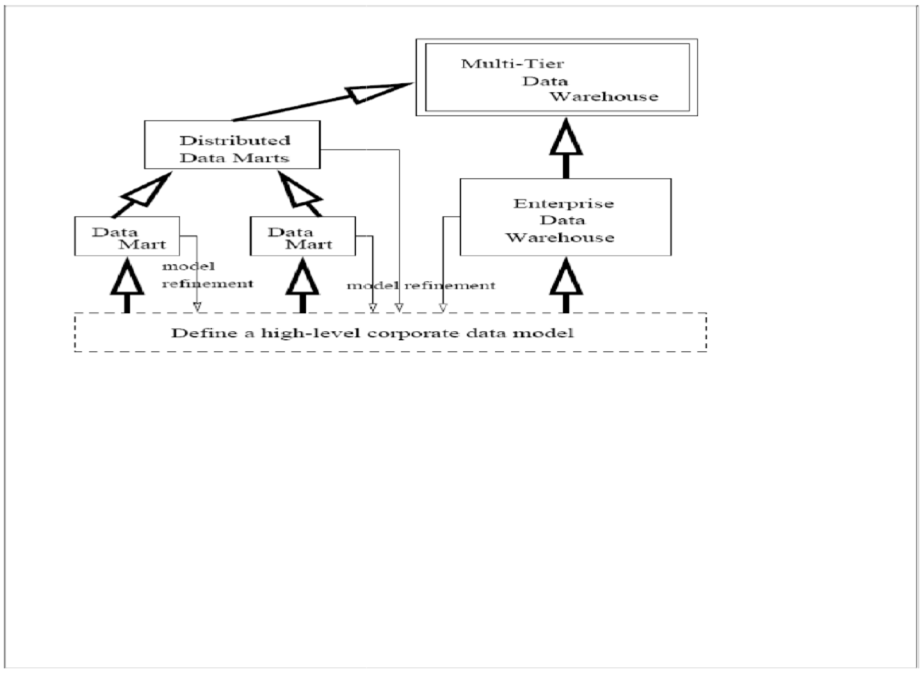

Data Warehouse: A Multi-Tiered Architecture
Other sources
Operational DBs
Extract Transform Load Refresh
Monitor & Integrator
Data Marts
OLAP Server
Serve
Analysis Query Reports Data mining
Data Sources
Data Storage
OLAP Engine
Front-End Tools
December 15, 2009
Data Mining: Concepts and Techniques
1

Figure Multi-tiered Architecture

A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner, as shown in fig. First, a high-level c orporate data model is defined within a reasonably short period of time (such as one or two months) that p rovides a corporate-wide, consistent, integrated view of data among different subjects and potential ussages. This high-level model, although it will need to be refined in the further development of enterprise data warehouses and departmental data marts, will greatly re uce future integration problems

Figure: A recommended approach fo r data warehouse development.


Multi-Tier Data Warehouse
Distributed Data Marts
Data Mart
Data Mart
Enterprise Data Warehouse
model refinement
model refinement
Define a high-level corporate data model

Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set as above. Third, distributed data marts can be constructed to integrate different data marts via hub servers. Finally, a multi-tier data warehouse is constructed where the enterprise warehouse is the sole custodian of all warehouse data; this is then distributed to the various dependent data marts.

OLAP server architectures: ROLAP vs. MOLAP vs. HOLAP

Logically, OLAP engines present business users with multidimensional data from data warehouses or data marts, without concerns regarding how or where the data are stored.

However, the physical architecture and implementation of OLAP engines must consider data storage issues.

Implementations of a warehouse server engine for OLAP processing include:

Relational OLAP (ROLAP) servers:

These are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces. ROLAP servers include optimization for each DBMS backend, implementation of aggregation navigation logic, and additional tools and services. ROLAP technology tends to have greater scalability than MOLAP technology. The DSS server of Microstrategy and Metacube of Informix, for example, adopt the ROLAP approach2.

Multidimensional OLAP (MOLAP) servers:

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

Hybrid OLAP (HOLAP) servers:

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

store. The Microsoft SQL Server 7.0 O AP Services supports a hybrid OLAP server.

Specialized SQL servers:

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

The OLAP functional architectu re consists of three components: the data store, the OLAP server, and the user presentation module. The data s tore can be further classified as a relational data sto re or a multidimensional data store, depending on whether ROLAP or MOLAP architecture is adop ted.

Data warehouse implementation

Data warehouses contain huge v olumes of data. OLAP engines demand that decisio n support queries be answered in the order of seconds. Therefore, it is crucial for data warehouse systems to s upport highly efficient cube computation techniques, a ccess methods, and query processing techniques. \H ow can this be done?", you may wonder. In this section , we examine methods for the efficient implementation of data warehouse systems.

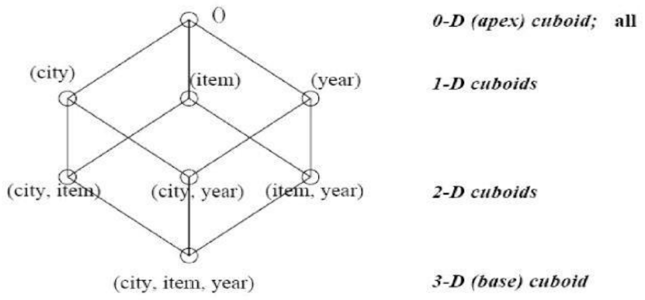


Figure: Lattice of cuboids, making up a 3-dimensional data cube.

Efficient computation of data cubes

At the core of multidimensional data analysis is the efficient computation of aggreg ations across many sets of dimensions. In SQL terms, these aggregations are referred to as group-by's. The compute cube operator and its implementation One approach to cube computation extends SQL so as to include a compute cube operator. The compute cube

operator computes aggregates over all subsets of the dimensions specified in the operation. Example 10 Suppose that you would like to create a data cube for All-Electronics sales which contains the following: item, city, year, and sales in dollars. You would like to be able to analyze the data, with queries such as the following:

1. **“Compute the sum of sales, grouping by item and city."**
2. **“Compute the sum of sales, grouping by item."**
3. **“Compute the sum of sales, grouping by city".**

What is the total number of cuboids, or group-by’s that can be computed for this data cube? Taking the three attributes, city, item, and year, as three dimensions and sales in dollars as the measure, the total number of cuboids, or group-by’s that can be computed for this data cube is 23 = 8. The possible group-by's are the following: f(city; item; year), (city; item), (city; year), (item; year), (city), (item), (year), ()g, where () means that the group-by is empty (i.e., the dimensions are not grouped).

This group-by's form a lattice of cuboids for the data cube, as shown in fig. The base cuboid contains all three dimensions, city, item, and year. It can return the total sales for any combination of the three dimensions. The apex cuboid, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales. Consequently, it is represented by the special value all.

An SQL query containing no group-by, such as \compute the sum of total sales" is a zero-dimensional operation. An SQL query containing one group-by, such as \compute the sum of sales, group by city" is a one-dimensional operation. A cube operator on n dimensions is equivalent to a collection of group by statements, one for each subset of the n dimensions. Therefore, the cube operator is the n-dimensional generalization of the group by operator. The data cube in Example 11, can be defined as

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

For a cube with n dimensions, there are a total of 2n cuboids, including the base cuboid. The statement compute cube sales explicitly instructs the system to compute the sales aggregate cuboids for all of the eight subsets of the set of {item, city, year}, including the empty subset. A cube computation operator was first proposed and studied by Gray, et al. (1996).

On-line analytical processing may need to access different cuboids for different queries. Therefore, it does seem like a good idea to compute all or at least some of the cuboids in a data cube in advance. A pre-computation lead to fast response time and avoids some redundant computation. Actually, most, if not all, OLAP products resort to some degree of pre-computation of multidimensional aggregates.

A major challenge related to thi s pre-computation, however, is that the required storage space may explode if all of the cuboids in a data cube are pre-computed, especially when the cube has s everal dimensions associated with multiple level hierarchies. “How many cuboids are there in an n-dimensional data cube?" If there were no hierarchies associated with each dimension, then the total numb er of cuboids for an n-dimensional data cube, as we have seen above, is 2n.

However, in practice, many dimensions do have hierarchies. For example, the dime nsion time is usually not just one level, such as year, but rather a hierarchy or a lattice, such as day < week < month < quarter < year. For an n-dimensional dat a cube, the total number of cuboids that can be gene rated (including the cuboids generated by climbing up the hierarchies along each dimension) is:

$$T = \prod_{i=1}^{n} (L_i + 1),$$

where Li is the number of levels associated with dimension i (excluding the virtual top level all since generalizing to all is equivalent to the removal of a dimension). This formula is based on the fact that at most one abstraction level in each dimension will appear in a cuboid. For example, if the cube ha s 10 dimensions and each dimension has 4 levels, the tot al number of cuboids that can be generated.

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

Partial materialization: Selected computation of cuboids There are three choices for data cube materialization:

(1) pre-compute only the base cuboid and none of the remaining \non-base" cuboids (no materialization),

(2) Pre-compute all of the cuboids (full materialization), and

(3) Selectively compute a proper subset of the whole set of possible cuboids (partial materialization).

The first choice leads to computing expensive multidimensional aggregates on the y, which could be slow. The second choice may require huge amounts of memory space in order to store all of the pre-computed cuboids. The third choice presents an interesting trade-o_ between storage space and response time.

The partial materialization of cuboids should consider three factors:

(1) identify the subset of cuboids to materialize,

(2) exploit the materialized cuboids during query processing, and

(3) efficiently update the materialized cuboids during load and refresh.

The selection of the subset of cuboids to materialize should take into account the queries in the workload, their frequencies, and their accessing costs. In addition, it should consider workload characteristics, the cost for incremental updates, and the total storage requirements. The selection must also consider the broad context of physical database design, such as the generation and selection of indices. Several OLAP products have adopted heuristic approaches for cuboid selection.

A popular approach is to materialize the set of cuboids having relatively simple structure. Even with this restriction, there are often still a large number of possible choices. Under a simplified assumption, a greedy algorithm has been proposed and has shown good performance.

A multidimensional data model

Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a data cube. In this section, you will learn how data cubes model n-dimensional data. You will also learn about concept hierarchies and how they can be used in basic OLAP operations to allow interactive mining at multiple levels of abstraction.

From tables and spreadsheets to data cubes

\What is a data cube?"

A data cube allows data to be modeled and viewed in multiple dimensions. It is de ned by dimensions and facts.

In general terms, dimensions are the perspectives or entities with respect to which an organization wants to keep reco rds. For example, AllElectronics may creat e a sales data warehouse in order to keep re cords of the store's sales with respect to the dimensions time, item, branch, and location. These dimensions allow the store to keep track of things like monthly sales of items, and the branches and locations at which the items were sold. Each dimension may have a table associated with it, called a dimension table, which further describes the dimension. For example, a dimension table f or item may contain the attributes item na me, brand, and type. Dimension tables can b e speci ed .

by users or experts, or automatically generated and adjusted based on data distributions.

A multidimensional data mo del is typically organized around a central t heme, like sales, for instance. This theme is represented by a fact table. Facts are numerical measures. Think of them as the q uantities by which we want to analyze rel¯ ationships between ¯dimensions. Examples of facts¯for a sales data warehouse include d ollars sold (sales amount in dollars), units sold (number of units sold), and amount budg eted. The fact table contains the names of the facts, or measures, as well as keys to ea ch of the related dimension tables. You will soon get a clearer picture of how this wo rks when we later look at multidimensional schemas.

Although we usually think of cubes as 3-D geometric structures, in data warehousing the data cube is n- dimensional. To gain a better understandin g of data cubes and the multidimensional data model, let's start by looking at a simple 2-D data cube which is, in fact, a table orr spreadsheet for sales data from AllElectronics. In particular, we will look at the AllE lectronics sales data for items sold per quarter in the city of Vancouver. These data are shown in Table 2.2. In this 2-D represent ation, the sales for Vancouver are shown with respect to the time dimension (organized in quarters) and the item dimension (organized according to the types of items s old). The fact, or measure displayed is dollars sold.

time (quarter)	Sales for all locations in Vancouver			
	item (type)			
	home entertainment	computer	phone	security
Q1	605K	825K	14K	400K
Q2	680K	952K	31K	512K
Q3	812K	1023K	30K	501K
Q4	927K	1038K	38K	580K

Table: A 2-D view of sales data for AllElectronics according to the dimensions time and item, where the sales are from branches located in the city of Vanco uver. The measure displayed is dollars sold.

	location = \Vancouver"				locat ion = \Montreal"				location = \New York"				location = \Chicago"			
t	item nt.				item nt.				item nt.				item nt.			
i	home	comp.	phone	sec.	home	comp.	phone	sec.	home	comp.	phone	sec.	home	comp.	phone	sec.
1	05K	25K	4K	00K	18K	46K	3K	91K	087K	68K	8K	72K	54K	82K	9K	23K
2	80K	52K	1K	12K	94K	69K	2K	82K	130K	024K	1K	25K	43K	90K	4K	98K
3	12K	023K	0K	01K	40K	95K	8K	28K	034K	048K	5K	002K	032K	24K	9K	89K
Q4	927K	1038K	38K	580K	978K	864K	59K	784K	1142K	1091K	54K	984K	1129K	992K	63K	870K

Table 2.3: A 3-D view of sale s data for AllElectronics, according to the dimensions time, item, and location. ~~T~~he measure displayed is dollars sold.

Now, suppose that we woul d like to view the sales data with a third dimension. For instance, suppose we would l ike to view the data according to time, ite m, as well as location. These 3-D data are shown in Table 2.3. The 3-D data of Table 2.3 are represented as a series of 2-D table s. Conceptually, we may also represent the same data in the form of a 3-D data cube, as in Figure 2.1.

Suppose that we would now like to view our sales data with an additional fourth dimension, such as supplier. Vie wing things in 4-D becomes tricky. Howeve r, we can think of a 4-D cube as being a s eries of 3-D cubes, as shown in Figure 2 .2. If we continue in this way, we may disp lay any n-D data as a series of (n 1)-D \cubes". The data cube is a metaphor for multid imensional data storage. The actual physical storage of such data may di er from its logical representation. The important thing to remember is that data cubes are n-dimensional, and do not con ne data to 3-D.

The above tables show the d ata at di erent degrees of summarization. In the data warehousing research literature, a data cube such as each of the above is referred to as a cuboid. Given a set of dimensions, we can construct a lattice of cuboids, each showing the data at a di erent level of summ arization, or group by (i.e., summarized b y a

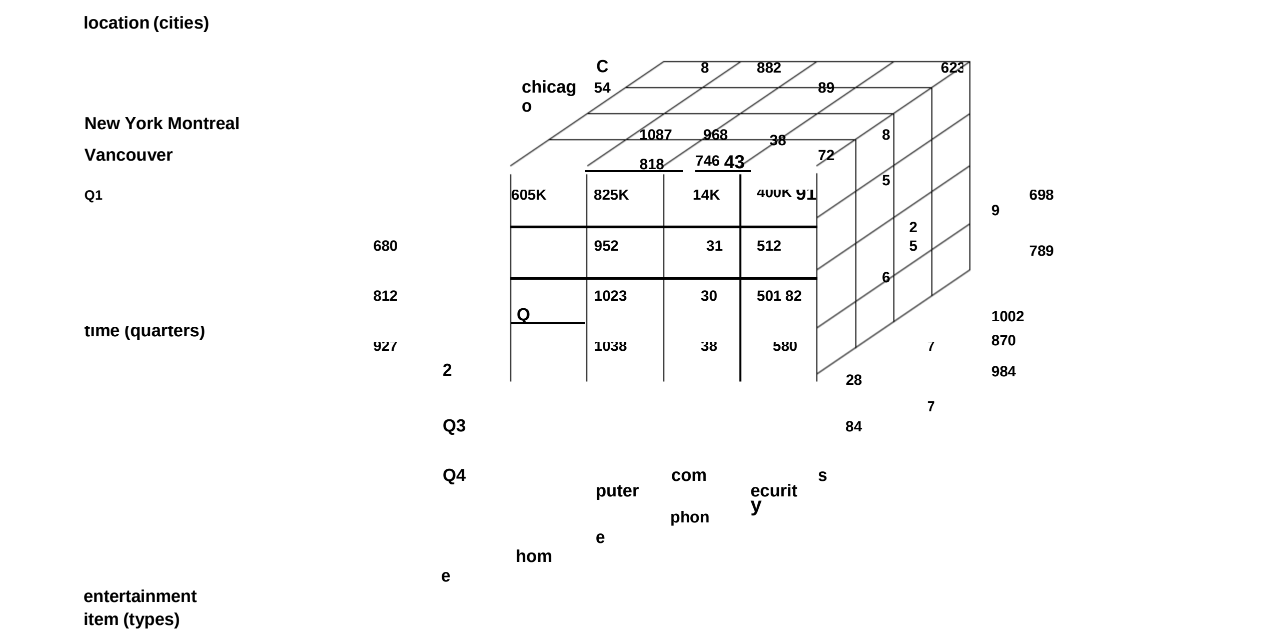


Figure 2.1: A 3-D data cube r epresentation of the data in Table 2.3, according to the dimensions time, item, and location. The measure displayed is dollars sold.

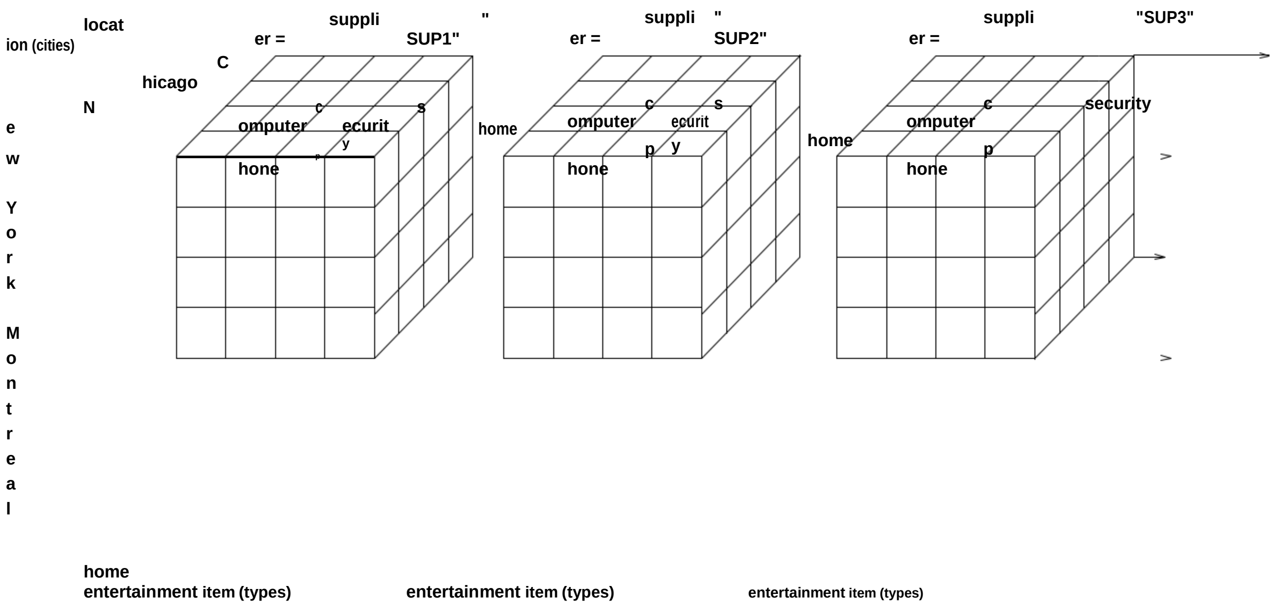


Figure 2.2: A 4-D data cube r epresentation of sales data, according to the dimensions time, item, location, and supplier. T he measure displayed is dollars sold.

di erent subset of the dimen sions). The lattice of cuboids is then referred to as a data cube. Figure 2.3 shows a lattice of cuboids forming a data cube for the d mensions time, item, location, and supplier.

The cuboid which holds the l owest level of summarization is called the base cuboid. For example, the 4-D cuboid in Figure 2.2 is the base cuboid for the given ti me, item, location, and supplier dimensions. Figure 2.1 is a 3-D (non-base) cuboid for time, item, and location, summarized for all suppliers. The 0-D cuboid which holds the highest level of summarization is called the apex cuboid. In our example, this is the total sales, or dollars sold, summarized for all four dimensions. The apex cuboid is typically denoted by all.

Stars, snow akes, and fact constellations: schemas for multidimensional databases

The entity-relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities or objects, and the relationships between them. Such a data model is appropriate for on- line transaction processing. Data warehouses, however, require a concise, subject-oriented schema which facilitates on-line data analysis.

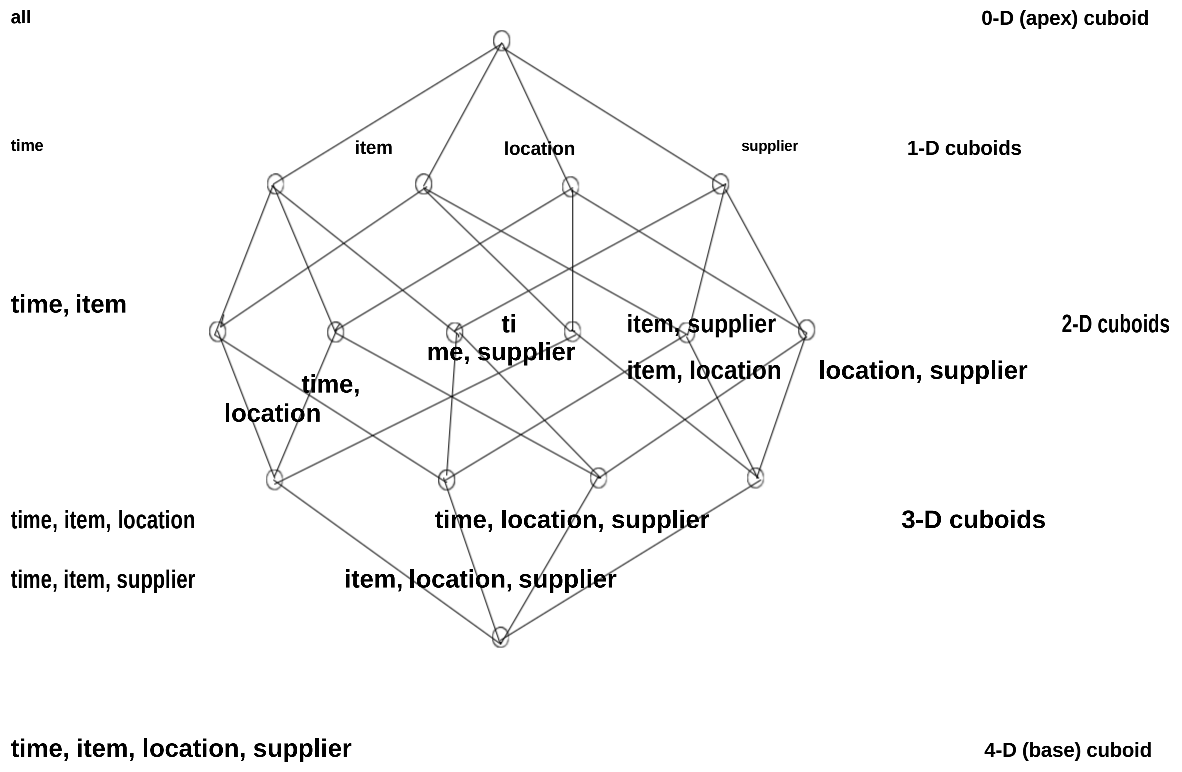
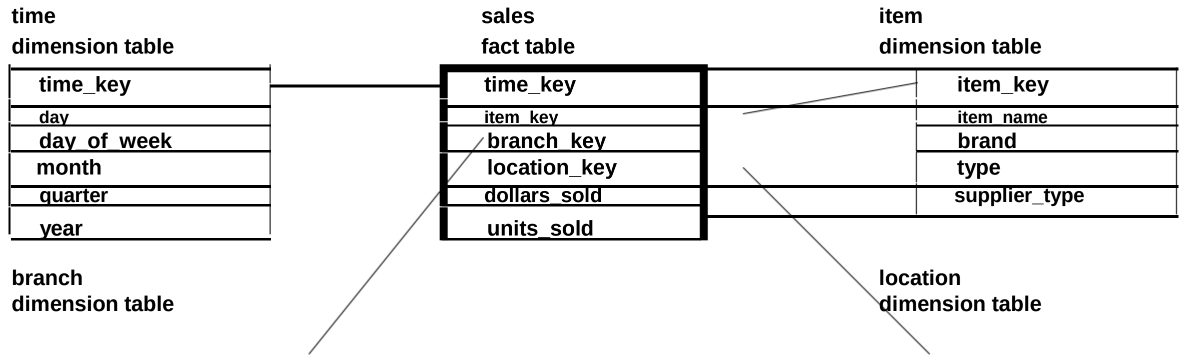


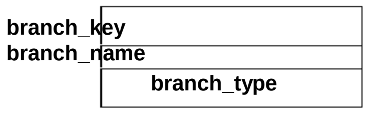
Figure: Lattice of cuboids, making up a 4-D data cube for the dimensions time, item, location, and supplier. Each cuboid represents a di erent degree of summarization.

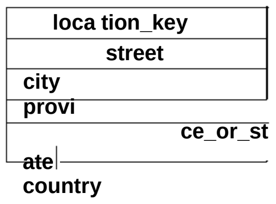
form of a star schema, a snow ake schema, or a fact constellation schema. Let's have a look at each of these schema types.

Star schema: The most common modeling paradigm is the star schema in which the data warehouse contains

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







An example of a star schema for AllElectronics sales is shown in Figure 2.4. Sales are considered along four dimensions, namely time, item, branch, and location. The schema contains a central fact table for sales which contains keys to each of the four dimensions, along with two measures: dollars sold and units sold.

To minimize the size of the fact table, dimension identi ers (such as time key and item key) are system-generated identfiers.

Notice that in the star schema, each dimension is represented by only one tab le, and each table contains a set of attributes. For example, the location dimension table contains the attribute set flocation key, street, city, province or state, countryg. This constraint may introduce some redundancy. For exa mple, \Vancou- ver" and \Victoria" are both cities in the Canadian province of British Columbia. Entries for such cities in the location dim ension table will create redundancy among the attributes province or state and country, i.e., (.., Vancouver, British Columbia, Canada) and (.., Victoria, British Co lumbia, Canada). More- over, the attributes within a dimension table may form either a hie rarchy (total order) or a lattice (partial order).

Snow flake schema: The sn ow ake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snow flake.

The major di erence between the snow ake and star schema models is that the dim ension tables of the snow ake model may be kept in normalized form. Such a table is easy to ma intain and saves storage space because a large dimension table can become enormous when the d imensional structure is included as columns. However, this saving of space is negligible in comparison to the typical magnitude of the fact table. Fu rthermore, the snow ake structure can reduce the e ectiveness of browsing since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, the snow ake schema is not as popular as the star schema in data warehouse desi gn.

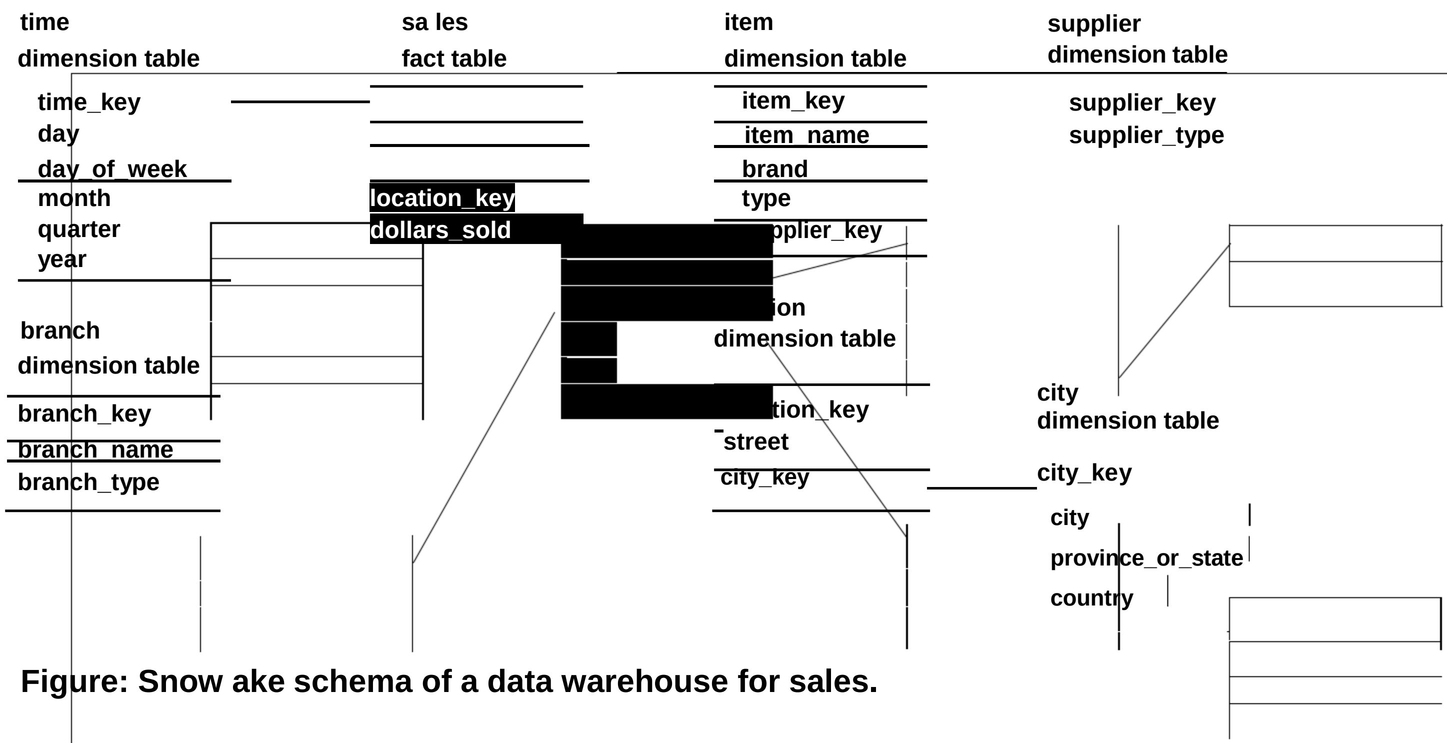


Figure: Snow ake schema of a data warehouse for sales.

An example of a snow ake schema for AllElectronics sales is given in Figure Here, the sales fact table is ident ical to that of the star schema. The main di erence between the two schemas is i n the de nition of dimension tables. The single dimension table for item in the star schema is normalized in the snow ak e schema, resulting in new item and supplier tables. For example, the item dimension table now contains the attributes supplier key, type, brand, item name, and item k ey, where supplier key is linked to the su pplier dimension table, containing supplier type and supplier key information. Similarly, the single dimension table for location in the star schema can be normalized into t wo new tables: location and city. The city key in the new location table links to the city dimension. Notice that further normaliz ation can be performed on province or state and country in the snow flake schema.

A compromise between the st ar schema and the snow ake schema is to ado pt a mixed schema where only the very large di mension tables are normalized. Normalizing large dimension tables saves storage space, while keeping small dimension tables unnormal ized may reduce the cost and performance degra dation due to joins on multiple dimension tables. Doing both may lead to an overall performan ce gain. However, careful performance tuning c ould be required to determine which dimension tables should be normalized and split into multiple tables.

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

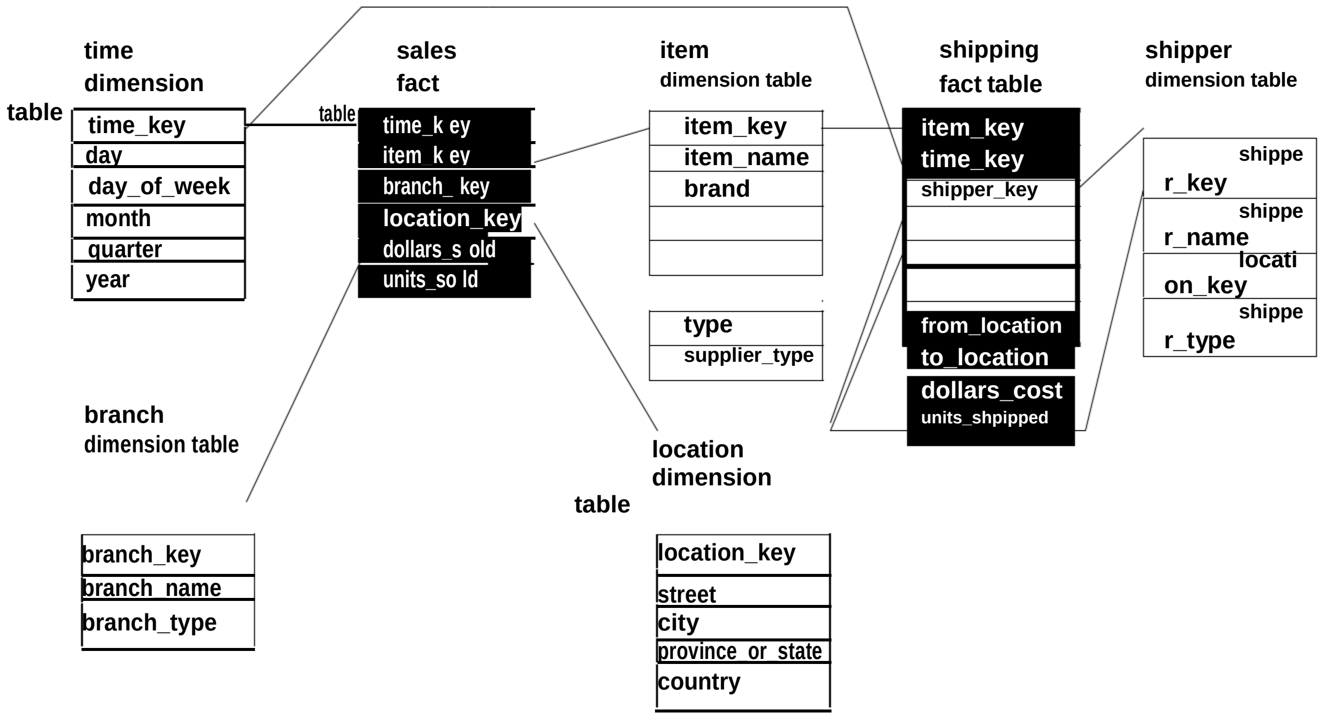


Figure: Fact constellation sche ma of a data warehouse for sales and shipping.

An example of a fact cons tellation schema is shown This schema speci es two fact tables, sales and shipping. The sales table de nition is identical to that of the star schema The shipping table has ve dimensions, or keys: time key, item key, shipper key, from location, and too location, and two measures: dollars cost and units shipped. A fact constellation schema allows dimension tables to be share d between fact tables. For example, the dime nsions tables for time, item, and location, are shared between both the sales and shipp ing fact tables. 2

In data warehousing, there is a distinction between a data warehouse and a data mart. A data warehouse collects information about subjects that span the entire organizatio n, such as customers, items, sales, assets, and personnel, and thus its scope is enterprise-wide. For data warehouses, the fact constellation schema is commonly used since it can model multiple, interrelated subjects. A data mart , on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is department-wide. For data marts, the star or snow ake schema are popular since each are geared towards modeling single subjects.

Examples for d e ning star, snow ake, and fact c onstellation schemas

\How can I de ne a multidimensional schema for my data?"

Just as relational query languages like SQL can be used to specify relational

queries, a data mining query language can be used to specify data mining tasks. In particular, we examine an SQL-based data mining query language called DMQL which contains,

language primitives for de ning data warehouses and data marts. Language primitives for specifying other data mining tasks, such as the mining o f concept/class descriptions, associations, classi cations, and so on.

defi ne dimension #dimension name as (attribute or subdimension list)

Let's look at examples of how to de ne the star, snow ake and constellations schemas of to
using DMQL. DMQL keywords are displayed in sans serif font.

The star schema of is defined in DMQL as follows. de ne

cube sales star [time, item, branch, location]:
dollars sold = sum(sales in dollars), units sold = count(*)
de ne dimension time as (time key, day, day of week, month, quarter, year) de ne dimension item as (item key, item name, brand, ty pe, supplier type) de ne dimension branch as (branch key, branch name, branch type)
define dimension location as (location key, street, city, province or state, country)

The de ne cube statement de nes a data cube called sales star , which corresponds to the central sales fact table. This command speci es the key s to the dimension tables, and the two m easures, dollars sold and units sold. The d ata cube has four dimensions, namely ti me, item, branch, and location. A de ne di mension
statement is used to de ne each of the dimensions. 2

The snow flake schema is defined in DMQL as follows.

define cube sales snow ake [time, item, branch, location]:
dollars sold = sum(sales in dollars), units sold = count(*)
de fine dimension time as (time key , day, day of week, month, quarter, year)
de fine dimension item as (item key, item name, brand, type, supplier (supplier key, supplier type)) defi ne dimension branch as (branch key, branch name, branch type)
defi ne dimension locatio n as (location key, street, city (city key, ci ty, province or state, country))

This de finition is similar to that of sales star , except that, here, the item and location dimensions tables are n ormalized. For instance, the item dimension of the sales star data cube has been normalized in the sales snow ake cube into two dimension tables, item and supp lier. Note that the dimension de nition for supplier is speci ed within the de nition for item. De ning supplier in this way im plicitly creates a supplier key in the ite m dimension table de nition. Similarly, the location dimension of the sales star dat a cube has been normalized in the sales sn ow ake cube into two dimension tables, location and city. The dimension de nition for city is speci ed within the de nition for location. In this way, a city key is im plicitly created in the location dimension table defi nition.

Finally, a fact constella tion schema can be defined as a set of

interconnected cubes. The fact constellation schema of is defined in DMQQL

as follows.

de ne cube sales [time, item, branch, location]:
dollars sold = sum(sales in dollars), units sold = count(*)
de ne dimension time as (time key, day, day of week,
month, quarter, year)
de ne dimension item as (item key, item name, brand, type, supplier ty pe)
de ne dimension branch as (branch key, branch name, branch type)
de ne dimension location as (location key, street, city, province or state, c ountry)

de ne cube shipping [time, item, shipper, from location, to location]:
dollars cost = sum(cost in dollars), units shipped = count(*) de
ne dimension time as time in cube sales

Measures: their categorization and comput ation

Measures can be organize d into three categories, based on the kind of aggregate functions used.

distributive: An aggreg ate function is distributive if it can be computed in a distributed manner as follows: Suppose the data is partitioned into n sets. The computation of the functi on on each partition derives one aggregate value. If the result derived by apply ing the function to the n aggregate values is the same as that derived by ap plying the function on all the data without partitioning, the function can be computed in a distributed manner. For example, count() can be compu ted for a data cube by rst partitioning t he cube into a set of subcubes, computi ng count() for each subcube, and then sum ming up the counts obtained for each s ubcube. Hence count() is a distributive ag gregate function. For the same reason , sum(), min(), and max() are distributive ag gregate functions. A measure is distri butive if it is obtained by applying a distributive aggregate function.

algebraic: An aggregate function is algebraic if it can be compute d by an algebraic function with M argu- ments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function. For e xample, avg() (average) can be comput ed by sum()/count() where both sum() and co unt() are

min N(), max

is algebraic

dis- tributive aggregate functions. Similarly, it can be shown that N(), and standard deviation() are al gebraic aggregate functions. A measure if it is obtained by applying an al gebraic aggregate function.

holistic: An aggregate f unction is holistic if there is no constant b ound on the storage size needed to describe a subaggregate. That is, there does not exist an algebraic function with M argu ments (where M is a constant) that characterizes the computation. Common exa mples of holistic functions include median (), mode() (i.e., the most frequently occur ring item(s)), and rank(). A measure is holistic if it is obtained by applying a holistic aggregate function.

Once the selected cuboids have been materialized, it is important to take advantage of them during query processing. This involves determining the relevant cuboid(s) from among the candidate materialized cuboids, how to use available index structures on the materialized cuboids, and how to transform the OLAP operations on to the selected cuboid(s).

Finally, during load and refresh, the materialized cuboids should be updated efficiently. Parallelism and incremental update techniques for this should be explored. Multiway array aggregation in the computation of data cubes In order to ensure fast on-line analytical processing, however, we may need to pre-compute all of the cuboids for a given data cube. Cuboids may be stored on secondary storage, and accessed when necessary. Hence, it is important to explore efficient methods for computing all of the cuboids making up a data cube, that is, for full materialization.

These methods must take into consideration the limited amount of main memory available for cuboid computation, as well as the time required for such computation. To simplify matters, we may exclude the cuboids generated by climbing up existing hierarchies along each dimension. Since Relational OLAP (ROLAP) uses tuples and relational tables as its basic data structures, while the basic data structure used in multidimensional OLAP (MOLAP) is the multidimensional array, one would expect that ROLAP and MOLAP each explore very different cube computation techniques. ROLAP cube computation uses the following major optimization techniques.

Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples. Grouping is performed on some sub-aggregates as a \partial grouping step". These partial groupings" may be used to speed up the computation of other sub-aggregates.

Aggregates may be computed from previously computed aggregates, rather than from the base fact tables. “How do these optimization te chniques apply to MOLAP?" ROLAP uses value-based addressing, where dimension values are accessed by key-based addressing search strategies. In contrast, MOLAP uses direct array addressing, where dimension values are accessed via the position or index of their corresponding array locations. Hence, MOLAP cannot perform the value-based reordering of the first optimization technique listed above for ROLAP. Therefore, a different approach should be developed for the array-based cube construction of MOLAP, such as the following.

1. Partition the array into chunks. A chunk is a sub cube that is small enough to _t into the memory available for cube computation. Chunking is a method for dividing an n dimensional array into small n-dimensional chunks, where each chunk is stored as an object on disk. The chunks are compressed so as to remove wasted space resulting from empty array cells (i.e., cells that do not contain any valid data). For instance, “chunkID + offset" can be used as a cell addressing mechanism to compress a sparse array structure and when searching for cells within a chunk. Such a compression technique is powerful enough to handle sparse cubes, both on disk and in memory.

2. Compute aggregates by visiting (i.e., accessing the values at) cube cells. The order in which cells are visited can be optimized so as to minimize the number of times that each cell must be revisited, thereby reducing memory access and storage costs. The trick is to exploit this ordering so that partial aggregates can be computed

Simultaneously, and any unnecessary revisiting of cells is avoided. Since this chunking technique involves “overlapping" some of the aggreg ation computations, it is referred to as multi way array aggregation in data cube computation.

Summary

· A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data organized in support of management decision making.

· A multidimensional data model is typically used for the design of corporate data warehouses and departmental data marts.

· Concept hierarchies organize the values of attributes or dimensions into gradual levels of abstraction. They are useful in mining at multiple levels of abstraction.

On-line analytical processing (OLAP) can be performed in data warehouses/marts using the multidimensional data model. Typical OLAP operations include roll-up, drill-(down, cross, through), slice-and-dice, pivot (rotate), as well as statistical

operations such as ranking, computing moving averages and growth rates, etc. OLAP operations can be implemented efficiently using the data cube structure.

· Data warehouses often adopt three-tier architecture. The bottom tier is a warehouse database server, which is typically a relational database system.

· The middle tier is an OLAP server, and the top tier is a client, containing query and reporting tools. OLAP servers may use Relational OLAP (ROLAP), or Multidimensional OLAP (MOLAP), or Hybrid OLAP (HOLAP).

PART A(2MARKS):

1. When we can say the association rules are interesting?

Association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Users or domain experts can set such thresholds.

2. Explain Association rule in mathem atical notations.

Let I-{i1,i2,…..,im} be a set of items
Let D, the task relevant data be a set of database transaction T is a set of items
An association rule is an implication o f the form A=>B where A C I, B C I,
and An B=□. The rule A=>B contains in the transaction set D with support s,

where s is the percentage of transaction s in D that contain AUB. The Rule A=> B has confidence c in the transaction set D if c is the percentage of transacti ons in D containing A that also contain B.

3. Define support and confidence in As sociation rule mining.

Support S is the percentage of transacti ons in D that contain AUB.
Confidence c is the percentage of transactions in D containing A that also contain B. Supp ort (A=>B)= P(AUB)
Confidence (A=>B)=P(B/A)

4. How are association rules mined from large

databases? I step: Find all frequent item sets:

II step: Generate strong association rules from frequent item sets

5. Describe the different classifications of Association rule
mining. Based on types of values hand led in the Rule
i. Boolean association rule
ii. Quantitative association rul e
Based on the dimensions of data involved
i. Single dimensional association rule
ii. Multidimensional associatio n rule
Based on the levels of abstraction involved
i. Multilevel association rule
ii. Single level association rule
Based on various extensions
i. Correlation analysis
ii. Mining max pattern

6. What is the purpose of Apriori Algorithm?

Apriori algorithm is an influential algorithm for mining frequent item sets for Boolean association rules. The name of the algo rithm is based on the fact that the algorithm uses prior knowledge of frequent item set properties.

7. Define anti-monotone property.

If a set cannot pass a test, all of its sup ersets will fail the same test as well.

8. How to generate association rules fr om frequent item sets?

Association rules can be generated as follows
For each frequent item set1, generate all non empty subsets of 1. For every non empty subsets s of 1, output the rule "S=>(1-s)"if Support count(1)=min_conf, Support_count(s) Where min_conf is the minimum confidence threshold.

9. Give few techniques to improve the efficiency of Apriori algorithm.

Hash based technique
Transaction Reduction
Portioning
Sampling
Dynamic item counting

10. What are the things suffering the performance of Apriori candidate generation technique. Need to generate a huge number of candidate sets
Need to repeatedly scan the scan the database and check a large set of candidates by pattern matching

11. Describe the method of generating frequent item sets without candidate generation.

Frequent-pattern growth(or FP Growth) adopts divide-and-conquer strategy.
Steps:
Compress the database representing frequent items into a frequent pattern tree or FP tree Divide the compressed database into a set of conditional database
Mine each conditional database separately

12. What are multidimensional association rules?

Association rules that involve two or more dimensions or predicates Interdimension association rule: Multidimensional association rule with no repeated predicate or dimension. Hybrid-dimension association rule: Multidimensional association rule with multiple occurrences of some predicates or dimensions.

13. Define constraint-Based Association Mining.

Mining is performed under the guidance of various kinds of constraints provided by the user. The constraints include the following
Knowledge type constraints Data constraints Dimension/level constraints Interestingness constraints Rule constraints.

14. Define the concept of classification.

Two step process
A model is built describing a predefined set of data classes or concepts.
The model is constructed by analyzing database tuples described by attributes. The model is used for classification.

15. What is Decision tree?

A decision tree is a flow chart like tree structures, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distributions. The top most in a tree is the root node.

16. What is Attribute Selection Measure?

The information Gain measure is used to select the test attribute at each node in the decision tree. Such a measure is referred to as an attribute selection me asure or a measure of the goodness of split.

17. Describe Tree pruning methods.

□□Pre pruning

□□Post pruning

When a decision tree is built, many of the branches will reflect anomalies in the training da ta due to noise or outlier. Tree pruning methods address this pro blem of over fitting the data Approaches.

18. Define Pre Pruning

A tree is pruned by halting its construction early. Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset sampl es.

19. Define Post Pruning.

Post pruning removes branches from a "Fully grown" tree. A tree node is pruned by removing its branches. Eg: Cost Complexity Algorithm.

PART B(10 MARKS):

1. Explain the various primitives for specifying Data mining Task.
2. Describe the various descrip tive statistical measures for data mining.
3. Discuss about different typ es of data and functionalities.
4. Describe in detail about Int erestingness of patterns.
5. Explain in detail about data mining task primitives.
6. Discuss about different Issu es of data mining.
7. Explain in detail about data preprocessing. How data mining system are classified? Discuss each classification with an examp le.
8. How data mining system ca n be integrated with a data warehouse? Discuss with an example.
9. Explain data mining applic ations for Telecommunication industry.

.

UNIT-III

MININ G FREQUENT PATTERNS

Frequent patterns:

patterns (such as itemsets, su bsequences, or substructures) that appear in a data se t frequently. For example, a set of items, such as milk and bread, that appear . A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs fr equently in a shopping history database, is a (freque nt) sequential pattern. A substructure can refer to different struc tural forms, such as subgraphs, subtrees, or sublatti ces, which may be combined with itemsets or subsequences

Association Rule Min ing:

- Association rule mining is a po pular and well researched method for discovering interesting relations between variables in l arge databases.
- It is intended to identify strong rules discovered in databases using different measures of interestingness.
- Based on the concept of strong rules. introduced association rules.

Problem Definition:

The problem of association rule mi ning is defined as:

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a s et of n binary attributes called items.

Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the database.

Each transaction in D has a unique transaction ID and contains a subset of the items in I.

A rule is defined as an implication of the form

$X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

The sets of items (for short itemsets) X and Y are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively.

Example:

To illustrate the concepts, we use a small example from the supermarket domain. The s et of items is $I = \{\mathrm{milk}, \mathrm{bread}, \mathrm{butter}, \mathrm{beer}\}$ and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table.

An example rule for the supermarket could be $\{\mathrm{butter}, \mathrm{bread}\} \Rightarrow \{\mathrm{milk}\}$ meaning that if butter and bread are bought, custo mers also buy milk.

Example database with 4 items and 5 transactions

	Transaction ID	milk	bre ad	butter	beer
1		1	1	0	0
2		0	0	1	0
3		0	0	0	1

4	1	1	1	0
5	0	1	0	0

Important concepts of As sociation Rule Mining:

- The support $\mathrm{supp}(X)$ of an itemset X is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset $\{\text{milk}, \text{bread}, \text{butter}\}$ has a support of $1/5 = 0.2$ since it occurs in 20% of all transactions (1 out of 5 transactions).
- The confidenceof a rule is defined

$$\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X).$$

For example, the rule $\{\text{butter}, \text{bread}\} \Rightarrow \{\text{milk}\}$ has a confidence of $0.2/0.2 = 1.0$ in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter and bread, milk is bought as well). Confidence can be interpreted as an estimate of the probability $P(Y|X)$, th e probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.

- The liftof a rule is defined as

$$\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X) \times \mathrm{supp}(Y)}$$

or the ratio of the observed support to that expected if X and Y were independe nt. The rule $\{\text{milk}, \text{bread}\} \Rightarrow \{\text{butter}\}$ has a lift of $\frac{0.2}{0.4 \times 0.4} = 1.25$.

- The conviction of a rule is defined as

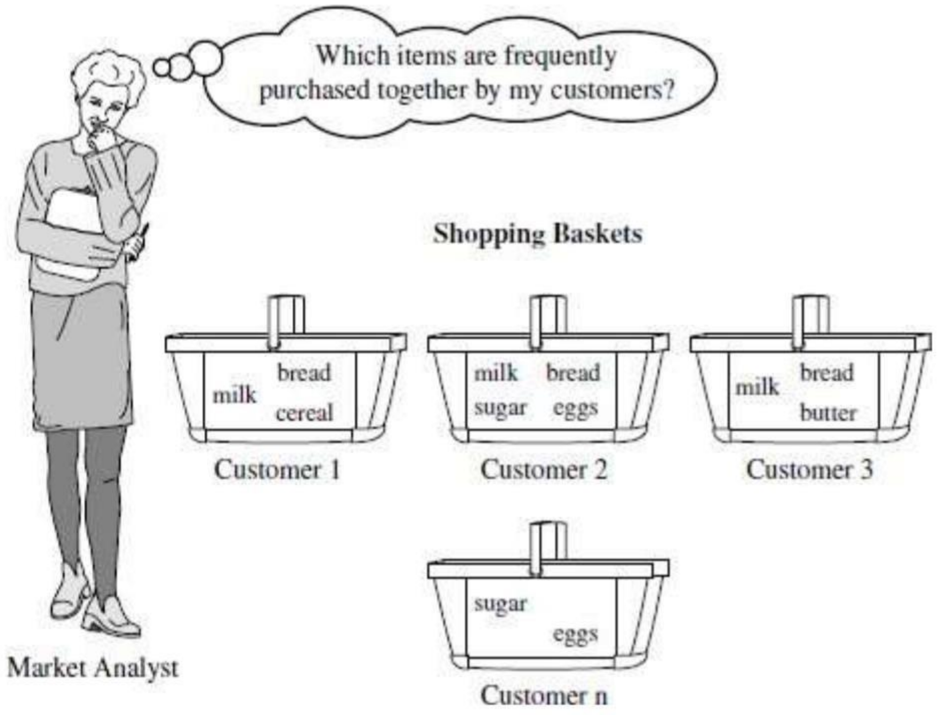
$$\mathrm{conv}(X \Rightarrow Y) = \frac{1 - \mathrm{supp}(Y)}{1 - \mathrm{conf}(X \Rightarrow Y)}$$

The rule $\{\text{milk}, \text{bread}\} \Rightarrow \{\text{butter}\}$ has a conviction of $\frac{1 - 0.4}{1 - .5} = 1.2$, and can be interpreted as t he ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X a nd Y were independent divided by the observed frequency of incorrect predictions.

Market basket analysis:

This process analyzes customer buy ing habits by finding associations between the different items thatcustomers place in their shop ping baskets. The discovery of such association scan help retailers develop marketing strate gies by gaining insight into which items are frequently

purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket. Such in formation can lead to increased sales by helpi ng retailers do selective marketing and plan their shelf space.



Example:

If customers who purchase comput ers also tend to buy anti virus software at the same time, then placing the hardware display close to the software display may help increase the sal es of both items. In an alternative strategy, pla cing hardware and software at opposite ends of the store may entice customers who purchase su ch items to pick up other items along the way. Fo instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading toward the software displa y to purchase antivirus software and may decide to purchase a home security system as well. Mar ket basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers tog ether, then having a sale on printers may encou rage the sale of printers as well as computers.

Frequent Pattern Mining:

Frequent pattern mining can be classified in various ways, based on the following criteria:

1. **Based on the completeness of patterns to be mined:**
   - **We can mine the complete set of frequent item sets, the closed frequent item sets, and the maximal frequent item sets, given a minimum support threshold.**
   - **We can also mine constrain ed frequent item sets, approximate frequent item set s ,near-match frequent item sets, to p-k frequent item sets and so on.**
2. **Based on the levels of abstr action involved in the rule set:**

Some methods for association rule mining can find rules at differing levels of abstraction.

For example, suppose that a set of association rules mined includes the followin g rules where X is a variable repre senting a customer:

buys(X, ―computer‖))=>bu ys(X, ―HP printer‖) (1)

buys(X, ―laptop computer‖)) =>buys(X, ―HP printer‖) (2)

In rule (1) and (2), the items bought are referenced at different levels of abstraction (e.g., ―computer‖ is a higher-level abstraction of ―laptop computer‖).

3. Based on the number of data dimensions involved in the rule:

- If the items or attributes in an association rule reference only one dimension, then it is a single-dimensional association rule.
  buys(X, ―computer‖))=>buys(X, ―antivirus software‖)
- If a rule references two or mo re dimensions, such as the dimensions age, income, and buys, then it is a multidimensional association rule. The following rule is an exam ple of a multidimensional rule:
  age(X, ―30,31…39‖) ^ income(X, ―42K,…48K‖))=>buys(X, ―high resolution TV ‖)

4. Based on the types of values handled in the rule:

- If a rule involves associations between the presence or absence of items, it is a Boolean association rule.
- If a rule describes associations between quantitative items or attributes, then it is a quantitative association rul e.

5. Based on the kinds of rules to be mined:

- Frequent pattern analysis c an generate various kinds of rules and other interesting relationships.
- Association rule mining ca n generate a large number of rules, many of which a re redundant or do not indicat e a correlation relationship among item sets.
- The discovered associations can be further analyzed to uncover statistical correlations, leading to correlation rules.

6. Based on the kinds of patter ns to be mined:

- Many kinds of frequent patterns can be mined from different kinds of data
- sets. Sequential pattern mining searches for frequent subsequences in a sequence data set, where a sequence records an ordering of events.
- For example, with sequential pattern mining, we can study the order in which items are frequently purchased. For instance, customers may tend to first buy a PC, foll owed by a digital camera,and then a m emory card.
-
-

data set. Single items are the simplest form of structure.

Each element of an item se t may contain a subsequence, a subtree, and so on.

- Therefore, structured pattern mining can be considered as the most general formof frequent pattern mining.

Efficient Frequent Itemset Mining Methods:

Finding Frequent Itemsets Using Candidate Generation:The Apriori Algor ithm

- Apriori is a seminal algo rithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties.
- Apriori employs an iterative approach known as a level-wise search, where k-it emsets are used to explore (k+1)-itemsets.
- First, the set of frequent 1- itemsets is found by scanning the database to accu mulate the count for each item, and collecting those items that satisfy minimum sup port. The resulting set is denoted L1.Next, L1 is used to find L2, the set of frequent 2 -itemsets, which is used to find L3, an d so on, until no more frequent k-itemsets can be fo und.
- The finding of each $L_k$requires one full scan of the database.
- A two-step process is follo wed in Apriori consisting of joinand prune action.

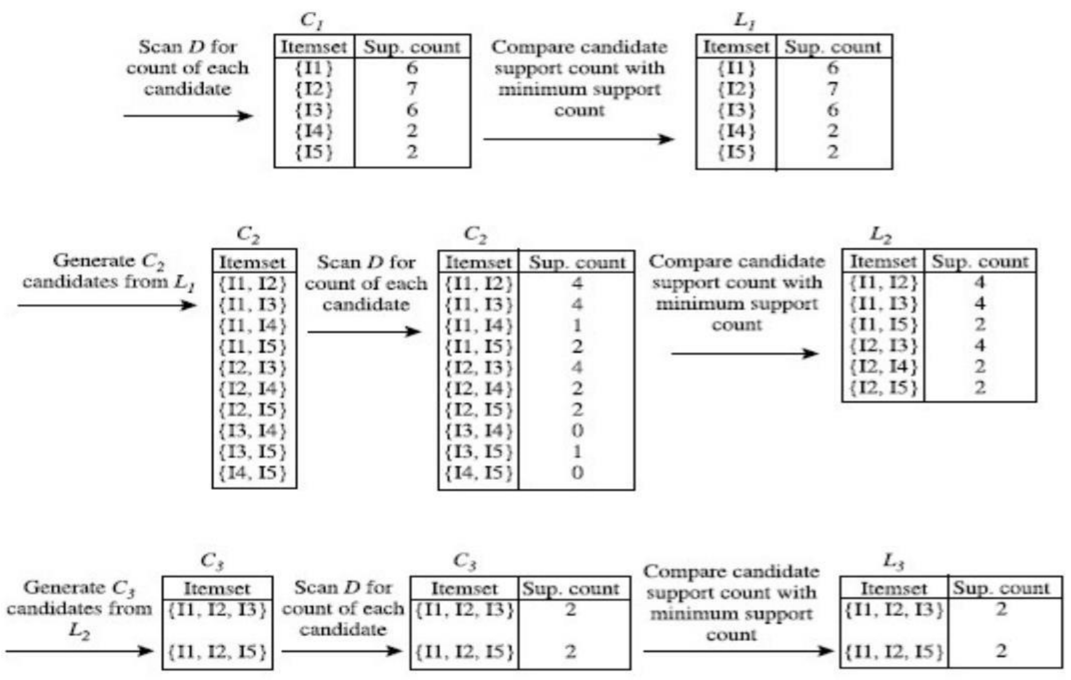
Example:

TID	List of item IDs
T10	I1, I2, I5
T20	I2, I4
T30	I2, I3
T40	I1, I2, I4
T50	I1, I3
T60	I2, I3
T70	I1, I3
T80	I1, I2, I3, I5
T90	I1, I2, I3

There are nine transactions in this database, that is, |D| = 9.

Steps:

1. **In the first iteration of the algorithm, each item is a member of the set of candidate1- itemsets, C1. The algorithm simply scans all of the transactions in order to countthe number of occurrences of each item.**
2. **Suppose that the minimum support count required is 2, that is, min sup = 2. The set of frequent 1- itemsets, L1, can thenbe determined. It consists of the candidate 1-itemsets satisfying minimum support.In our example, all of the candidates in C1 satisfy minimum support.**
3. **To discover the set of frequent 2-itemsets, L2, the algorithm uses the join L1 on L1 togenerate a candidate set of 2-itemsets, C2.No candidates are removed fromC2 during the prune step because each subset of thecandidates is also frequent.**
4. **Next, the transactions inDare scanned and the support count of each candidate itemsetInC2 is accumulated.**
5. **The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate2- itemsets in C2 having minimum support.**
6. **The generation of the set of candidate 3-itemsets,C3, Fromthejoin step, we first getC3 =L2x L2 = ({I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4},{I2, I3, I5}, {I2, I4, I5}. Based on the Apriori property that all subsets of a frequentitemsetmust also be frequent, we can determine that the four latter candidates cannotpossibly be frequent.**
7. **The transactions in D are scanned in order to determine L3, consisting of those candidate 3- itemsets in C3 having minimum support.**
8. **The algorithm uses L3x L3 to generate a candidate set of 4-itemsets, C4.**


Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C2 candidates from L1
Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C3 candidates from L2
Scan D for count of each candidate
Compare candidate support count with minimum support count

C_1

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

L_1

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

C_2

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

C_2

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

L_2

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

C_3

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

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie$
$\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

Generating Association Rules from Frequent Itemsets:

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

The conditional probability is expressed in terms of itemset support count, where $support\_count(A \cup B)$ is the number of transactions containing the itemsets $A \cup B$, and $support\_count(A)$ is the number of transactions containing the itemset A. Based on this equation, association rules can be generated as follows:

- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Example:

Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown in Table 5.1. Suppose the data contain the frequent itemset $l = \{I1, I2, I5\}$. What are the association rules that can be generated from l? The nonempty subsets of l are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$. The resulting association rules are as shown below, each listed with its confidence:

$I1 \wedge I2 \Rightarrow I5,$	$confidence = 2/4 = 50\%$
$I1 \wedge I5 \Rightarrow I2,$	$confidence = 2/2 = 100\%$
$I2 \wedge I5 \Rightarrow I1,$	$confidence = 2/2 = 100\%$
$I1 \Rightarrow I2 \wedge I5,$	$confidence = 2/6 = 33\%$
$I2 \Rightarrow I1 \wedge I5,$	$confidence = 2/7 = 29\%$
$I5 \Rightarrow I1 \wedge I2,$	$confidence = 2/2 = 100\%$

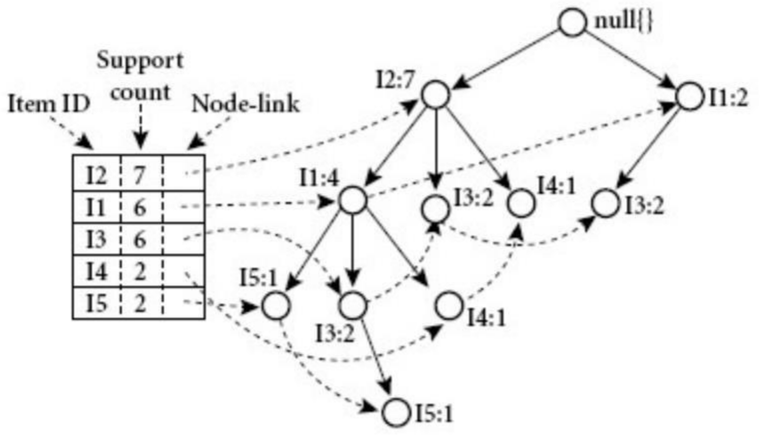
FP-Growth Method: Mining Frequent Itemsets without Candidate Generation

As we have seen, in many cases the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain.
An interesting method in this attempt is called frequent-pattern growth, or simply FP-growth, which adopts a divide-and-conquer strategy as follows. First, it compresses the database representing frequent items into a frequent-pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or ―pattern fragment,‖ and mines each such database separately. You'll see how it works with the following example.

FP-growth (finding frequent itemsets without candidate generation). We re-examine the mining of transaction database, D, of Table 5.1 in Example 5.3 using the frequent pattern growth approach.

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or *list* is denoted *L*. Thus, we have *L* = {{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}.

An FP-tree is then constructed as follows. First, create the root of the tree, labeled with "null." Scan database *D* a second time. The items in each transaction are processed in *L* order (i.e., sorted according to descending support count), and a branch is created for each transaction. For example, the scan of the first transaction, "T100: I1, I2, I5," which contains three items (I2, I1, I5 in *L* order), leads to the construction of the first branch of the tree with three nodes, ⟨I2: 1⟩, ⟨I1:1⟩, and ⟨I5: 1⟩, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T200, contains the items I2 and I4 in *L* order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common **prefix**, I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, ⟨I4: 1⟩, which is linked as a child of ⟨I2: 2⟩. In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

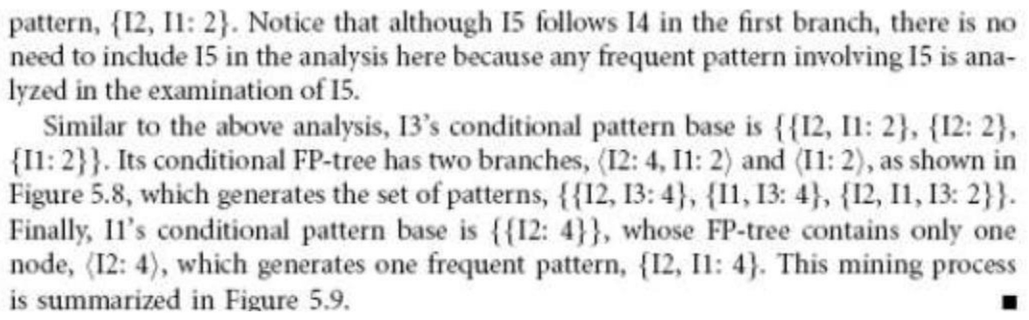


Mining the FP-tree by creating conditional (sub-)pattern bases.

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

Mining of the FP-tree is summarized in Table 5.2 and detailed as follows. We first consider I5, which is the last item in *L*, rather than the first. The reason for starting at the end of the list will become apparent as we explain the FP-tree mining process. I5 occurs in two branches of the FP-tree of Figure 5.7. (The occurrences of I5 can easily be found by following its chain of node-links.) The paths formed by these branches are ⟨I2, I1, I5: 1⟩ and ⟨I2, I1, I3, I5: 1⟩. Therefore, considering I5 as a suffix, its corresponding two prefix paths are ⟨I2, I1: 1⟩ and ⟨I2, I1, I3: 1⟩, which form its conditional pattern base. Its conditional FP-tree contains only a single path, ⟨I2: 2, I1: 2⟩; I3 is not included because its support count of 1 is less than the minimum support count. The single path generates all the combinations of frequent patterns: {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}.

For I4, its two prefix paths form the conditional pattern base, {{I2 I1: 1}, {I2: 1}}, which generates a single-node conditional FP-tree, ⟨I2: 2⟩, and derives one frequent

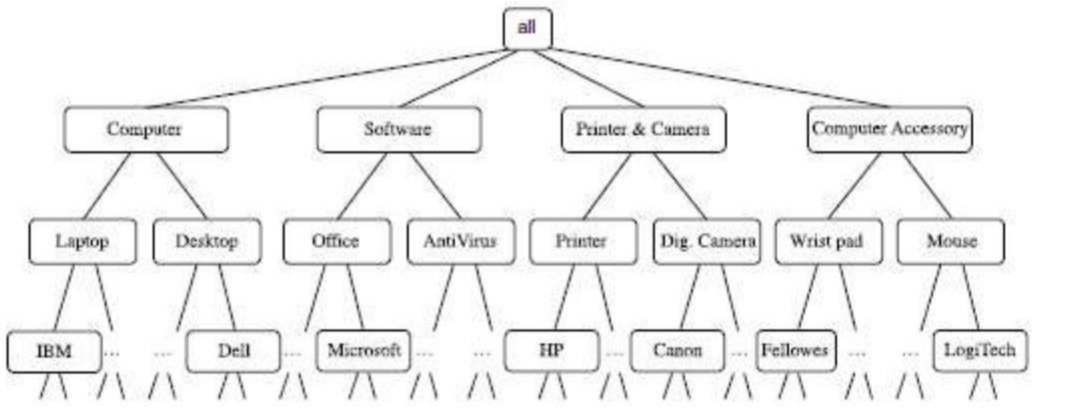

pattern, {I2, I1: 2}. Notice that although I5 follows I4 in the first branch, there is no need to include I5 in the analysis here because any frequent pattern involving I5 is analyzed in the examination of I5.

Similar to the above analysis, I3's conditional pattern base is {{I2, I1: 2}, {I2: 2}, {I1: 2}}. Its conditional FP-tree has two branches, ⟨I2: 4, I1: 2⟩ and ⟨I1: 2⟩, as shown in Figure 5.8, which generates the set of patterns, {{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}}. Finally, I1's conditional pattern base is {{I2: 4}}, whose FP-tree contains only one node, ⟨I2: 4⟩, which generates one frequent pattern, {I2, I1: 4}. This mining process is summarized in Figure 5.9. ■

Mining Multilevel Association Rules:

- **For many applications, it is difficult to find strong associations among data item s at low or primitive levels of abstraction due to the sparsity of data at those levels.**
- **Strong associations discove red at high levels of abstraction may represent com monsense knowledge.**
- **Therefore, data mining systems should provide capabilities for mining association rules at multiple levels of abstractio n, with sufficient flexibility for easy traversal among different abstraction spaces.**
- **Association rules generated from mining data at multiple levels of abstraction a re called multiple-level or multilevel association rules.**
- **Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework.**
- **In general, a top-down strategy is employed, where counts are accumulated fo r the calculation of frequent itemsets at each c oncept level, starting at the concept level 1 and wor king downward in the hierarchy toward the more specific concept levels,until no more freque nt itemsets can be found.A concepthierarchy defines a sequence of mappings froma set of low -level concepts to higherlevel,more general concepts. Data can be generalized by re placing low-level conceptswithin the data by their higher-level concepts, or ancestors, from a concept hierarchy**

TID	Items Purchased
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...



A concept hierarchy for *AllElectronics* computer items.

The concept hierarchy has five levels, respectively referred to as levels 0to 4, starting with level 0 at the root node for all.

- **Here, Level 1 includes compute r, software, printer&camera, and computer accessory. Level 2 includes laptop computer, desktop c omputer, office software, antivirus software Le el 3 includes IBM desktop computer . , Microsoft office software, and so on. Level 4 is the most specific a bstraction level of this hierarchy.**

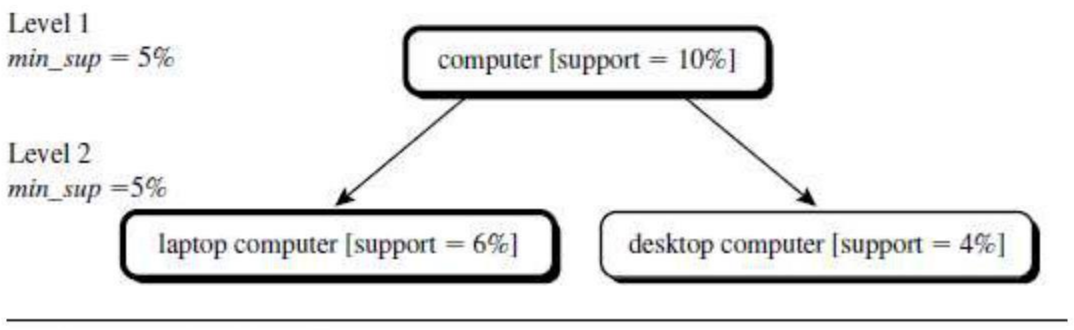
Approaches ForMining M ultilevel Association Rules:

1. UniformMinimum Support:

- **The same minimum support threshold is used when mining at each level of abstraction. When a uniform minimum support thres hold is used, the search procedure is simplified. The method is also simple in that users are required to specify only one minimum support threshold.**

The uniform support approach, however, has some difficulties. It is unlikely thatite ms at lower levels of abstraction will occur as frequently as those at higher levelsof abstraction.

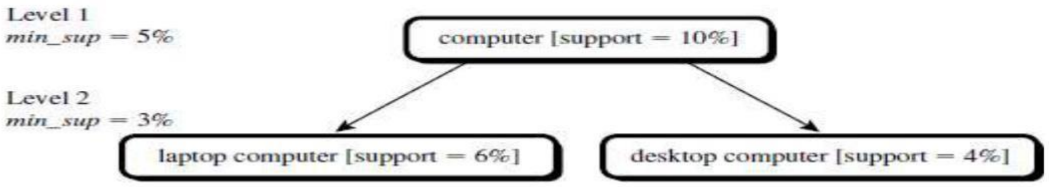
- **If the minimum support thresho ld is set too high, it could miss somemeaningful associations occurring at low abstraction levels. If the threshold is set too low, it may generate m any uninteresting associations occur ring at high abstractionlevels.**


Level 1
min_sup = 5%
computer [support = 10%]
Level 2
min_sup =5%
laptop computer [support = 6%]
desktop computer [support = 4%]

Multilevel mining with uniform support.

2. Reduced Minimum Support:

- **Each level of abstraction has its own minimum support threshold.**
- **The deeper the level of abstraction, the smaller the corresponding threshold is.**
- **For example,the minimum supp ort thresholds for levels 1 and 2 are 5% and 3%,res ectively. In this way, —computer,‖ —laptop comp uter,‖ and —desktop computer‖ areall considered frequent.**



3. Group-Based Minimum Support:

- **Because users or experts often have insight as to which groups are more importa nt than others, it is sometimes more desirable to set up user-specific, item, or group based minimal support thresholds when mining multilevel rules.**
- **For example, a user could set up the minimum support thresholds based on produc t price, or on items of interest, such as by setting p articularly low support thresholds for laptop computersand flash drives in order to pay particular attention to the association patterns containing items in th ese categories.**

Mining Multidimensional Association Rules from Relational Databases and Data Warehouses:

- **Single dimensional or intr adimensional association rule contains a single distinct predicate (e.g., buys)with multiple occurre nces i.e., the predicate occurs more than once within the rule.**

  buys(X, —digital camera‖)= >buys(X, —HP printer‖)

- **Association rules that involve two or more dimensions or predicates can be refe rred to as multidimensional associatio n rules.**

  age(X, “20…29”)^occupati on(X, “student”)=>buys(X, “laptop”)

- Above Rule contains three predicates (age, occupation,and buys), each of which occurs only once in the rule. Hence, we say t hat it has norepeated predicates.
- Multidimensional association rules with no repeated predicates arecalled interdimensional association rules.
- We can also mine multid imensional associationrules with repeated predicates, which contain multiple occurrences of some predicates.These rules are called hybrid- dime nsional association rules. An example of sucha rule is the following, where the predicate buys is repeated:

  age(X, ―20…29‖)^buys(X, ―laptop‖)=>buys(X, ―HP printer‖)

Mining Quantitative Association Rules:

- Quantitative association rules a re multidimensional association rules in which the numeric attributes are dynamically discretized durring the mining process so as to satisfy some minin g criteria, such as maximizing the confidence or compactness of the rules mined.
- In this section, we focus spe cifically on how to mine quantitative association rules having two quantitative attributes on the left-hand side of the rule and one categorical attribute on the right-hand side of the rule. That is

Aquan1 ^Aquan2 =>Acat

whereAquan1 and Aquan2 are tests on quantitative attribute interval

Acattests a categorical attribute fromthe task-relevantdata.

- Such rules have been referred to as two-dimensional quantitative association rules, because they contain two quantita tive dimensions.
- For instance, suppose you ar e curious about the association relationship betw een pairs of quantitative attributes, like cus tomer age and income, and the type of television (such as high-definition TV, i.e., HDTV) that customers like to buy.

An example of such a 2-D quantitative association rule is

age(X, ―30…39‖)^income(X, ―42K …48K‖)=>buys(X, ―HDTV‖)

From Association M ining to Correlation Analysis:

- A correlation measure can be used to augment the support-confidence framework for association rules. This lead s to correlation rules of the form A=>B [support, confidence, correlation]
- That is, a correlation rule is measured not only by its support and confide nce but alsoby the correlation between itemsetsA and B. There are many different correlation measures from which to choose. In this section, we study various correlation measures to determine which would be good for mining large data sets.
- Lift is a simple correlation measure that is given as follows. The occurrence of itemset A is independent of the occurren ce of itemsetB if $P(A \cup B)$ = P(A)P(B); otherwise, itemsetsA and B are dependent and correlated as events. This definition can easily be extended to more than two item sets.

The lift between the occurrence of A and B can be measured by computing

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- If the lift(A,B) is less than 1, then the occurrence of A is negatively correlated with the occurrence of B.
- If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other.
- If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.

From Association Mining to Correlation Analysis:

Most association rule mining algorithm s employ a support-confidence framework. Often, many interesting rules can be found using low support thresho lds. Although minimum support and confidence thresholds help weed out or exclude the exploration of a good number of uninteresting rules, many rules so generated are still not interesting to the users. Unfortunately, this is especia lly true when mining at low support thresholds or m ining for long patterns. This has been one of the major bottlen ecks for successful application of association rule mining.

1)Strong Rules Are Not Necessarily Interesting: An Example

Whether or not a rule is interesting can be assessed either subjectively or objectively. Ultim ately, only the user can judge if a given rule is interesting, and this judgment, being subjective, may differ from one user to another. However, objective interestingness measures, based on the statistics ―behind‖ the data, can be used as one step toward the goal of weeding out uninteresting rules from presentation to the user. The support and confidence measures are insufficient at filtering out uninteresting associat ion rules. To tackle this weakness, a correlation measure can b e used to augment the support-confidence framewor k for association rules. This leads to correlation rules of the fo rm

$$A \Rightarrow B\ [support, confidence. correlation].$$

Constraint-Based Association Mining

A data mining process may uncover thousands of rules from a given set of data, most of which end up being unrelated or uninteresting to the users. Often, users have a good sense of which —direction‖ of mining may lead to interestingpatterns and the —form‖ of the patterns or rules they would like to find. Thus, a good heuristic is to have the users specifysuch intuition or expectations as constraints to confine the search space. This strategy is known as constraint-based mining. The constraints can include the following

- **Knowledge type constraints:** These specify the type of knowledge to be mined, such as association or correlation.
- **Data constraints:** These specify the set of task-relevant data.
- **Dimension/level constraints:** These specify the desired dimensions (or attributes) of the data, or levels of the concept hierarchies, to be used in mining.
- **Interestingness constraints:** These specify thresholds on statistical measures of rule interestingness, such as support, confidence, and correlation.
- **Rule constraints:** These specify the form of rules to be mined. Such constraints may be expressed as metarules (rule templates), as the maximum or minimum number of predicates that can occur in the rule antecedent or consequent, or as relationships among attributes, attribute values, and/or aggregates.

1) Metarule-Guided Mining of Association Rules

"How are metarules useful?" Meta rules allow users to specify the syntactic form of rules that they are interested in mining. The rule forms can be used as constraints to help improve the efficiency of the mining process. Meta rules may be based on the analyst's experience, expectations, or intuition regarding the data or may be automatically generated based on the database schema.

Metarule-guided mining:- Suppose that as a market analyst for AllElectronics, you have access to the data describing customers (such as customer age, address, and credit rating) as well as the list of customer transactions. You are interested in finding associations between customer traits and the items that customers buy. However, rather than finding all of the association rules reflecting these relationships, you are particularly interested only in determining which pairs of customer traits promote the sale of office software.A metarule can be used to specify this information describing the form of rules you are interested in finding. An example of such a metarule is

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{"office software"}),$$

where P1 and P2 are predicate variables that are instantiated to attributes from the given database during the miningprocess, X is a variable representing a customer, and Y and W take on values of the attributes assigned to P1 and P2,respectively. Typically, a user will specify a list of attributes to be considered for instantiation with P1 and P2. Otherwise, adefault set may be used.

2) Constraint Pushing: Mining Guided by Rule Constraints

Rule constraints specify expected set/subset relationships of the variables in the mined rules, constant initiation of variables, and aggregate functions. Users typically employ their knowledge of the application or data to specify rule constraints for the mining task. These rule constraints may be used together with, or as an alternative to, metarule-guided mining. In this section, we examine rule constraints as to how they can be used to make the mining process more efficient. Let's study an example where rule constraints are used to mine hybrid-dimensional association rules.

Our association mining query is to "Find the sales of which cheap items (where the sum of the prices is less than $100)may promote the sales of which expensive items (where the minimum price is $500) of the same group for Chicagocustomers in 2004." This can be expressed in the DMQL data mining query language as follows,

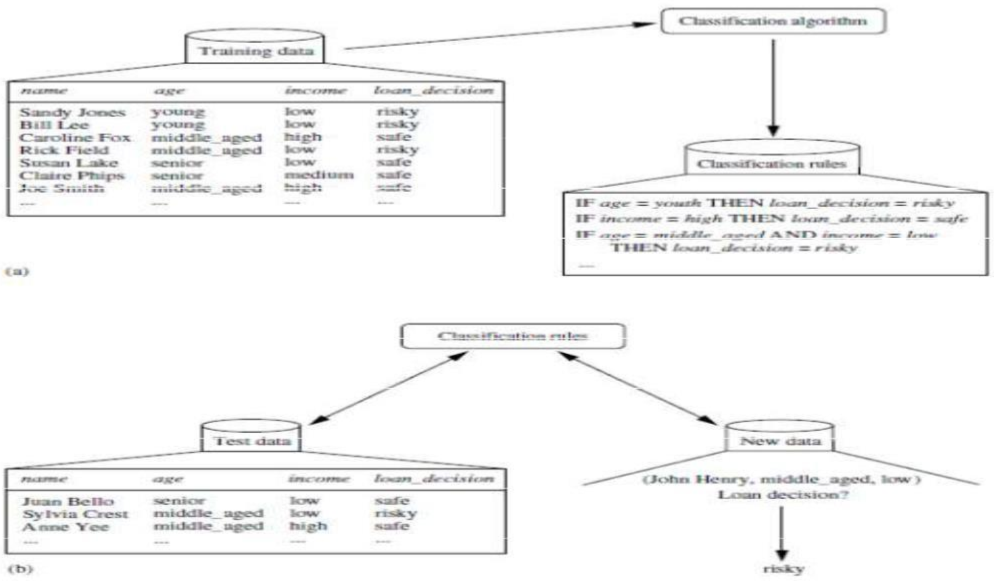
```
(1) mine associations as
(2) lives_in(C, _, "Chicago") ∧ sales+ (C, ?{I}, {S}) ⇒ sales+ (C, ?{J}, {T})
(3) from sales
(4) where S.year = 2004 and T.year = 2004 and I.group = J.group
(5) group by C, I.group
(6) having sum(I.price) < 100 and min(J.price) ≥ 500
(7) with support threshold = 1%
(8) with confidence threshold = 50%
```

Classification and Prediction:

Data analysis task is classification, where a model or classifier is constructed to predict categorical labels, such as "safe" or "risky" for the loan application data; "yes" or "no" for the marketing data; or "treatment A," "treatment B," or "treatment C" for the medical data. Suppose that the marketing manager would like to predict how much a given customer will spend during a sale at AllElectronics. This data analysis task is an example of numeric prediction, where the model constructed predicts a continuous-valued function, or ordered value, as opposed to a categorical label. This model is a predictor.

Data classification is a two-step process, as shown forthe loan application data of Figure 6.1. (The data are simplified for illustrative purposes.

In reality, we may expect many more attributes to be considered.) In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels. A tuple, X, is represented by an n-dimensional attribute vector, X = (x1, x2, : : : , xn), depicting n measurements made on the tuple from n database attributes, respectively, A1, A2, : : : , An.1 Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. class label of each training tuple is provided, this step is also known as supervised learning (i.e., the learning of the classifier is "supervised" in that it is told to which class each training tuple belongs). It contrasts with unsupervised learning (or clustering),



name	*age*	*income*	*loan_decision*
Sandy Jones	young	low	risky
Bill Lee	young	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

IF *age* = *youth* THEN *loan_decision* = *risky*
IF *income* = *high* THEN *loan_decision* = *safe*
IF *age* = *middle_aged* AND *income* = *low*
THEN *loan_decision* = *risky*
...

name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

Isues Regarding Classification and Prediction:

1. Preparing the Data for Classification and Prediction:

The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

(i) Data cleaning:

- **This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques) and the treatment of missingvalues (e.g., by replacing a missing value.Although most classification algorithms hav e some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.**

(ii) Relevance analysis:

- **Many of the attributes in the data may be redundant.**
- **Correlation analysis can be used to identify whether any two given attributes are statisticallyrelated.**
- **For example, a strong corre lation between attributes A1 and A2 would suggest that one of the two could be removed from further analysis.**
- **A database may also conta in irrelevant attributes. Attribute subset selection ca n be used in these cases to find a reduced set of attributes such that the resulting probability distr ibution of the data classes is as close as possible to the original distribution obtained using all attri butes.**

  Hence, relevance analysis, in the form of correlation analysis and attribute subset selection, can be used to detect attributes that do not contribute to the classification or predicti on task.
- **Such analysis can help improve classification efficiency and scalability.**

(iii) Data Transformation And Reduction

- **The data may be transfor med by normalization**

  Normalization involves s caling all values for a given attribute so that they fall within a small specified range, such as -1 to +1 or 0 to 1.
- **The data can also be trans formed by generalizing it to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous valuedattributes.**

.

Comparing Classification and Prediction Methods:

Accuracy:

- **The accuracy of a clas sifier refers to the ability of a given classifier to correctly predict the class label o f new or previously unseen data (i.e., tuples with out class label information).**
- **The accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.**

Speed:

This refers to the computati onal costs involved in generating and using the given classifier or predictor.

Robustness:

This is the ability of the classifier or predictor to make correct predictions given noisy dat a or data with missing values.

Scalability:

This refers to the ability to construct the classifier or predictor efficiently given large amo unts of data.

Interpretability:

- **This refers to the level of understanding and insight that is providedby the lassifier or predictor.**
- **Interpretability is subjective and therefore more difficultto assess.**

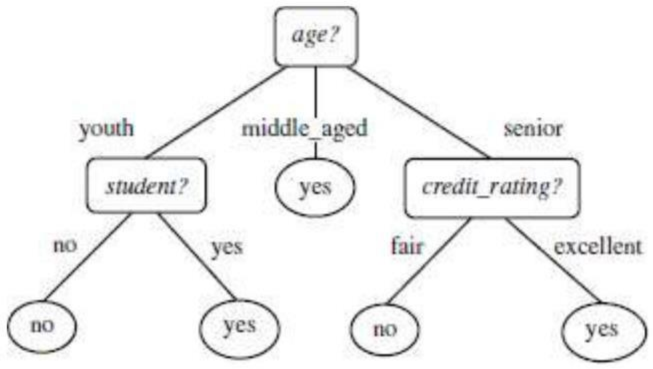
Classification by Decision Tree Induction:

•

tuples. • A decision tree is a flowchart-like tree structure,where

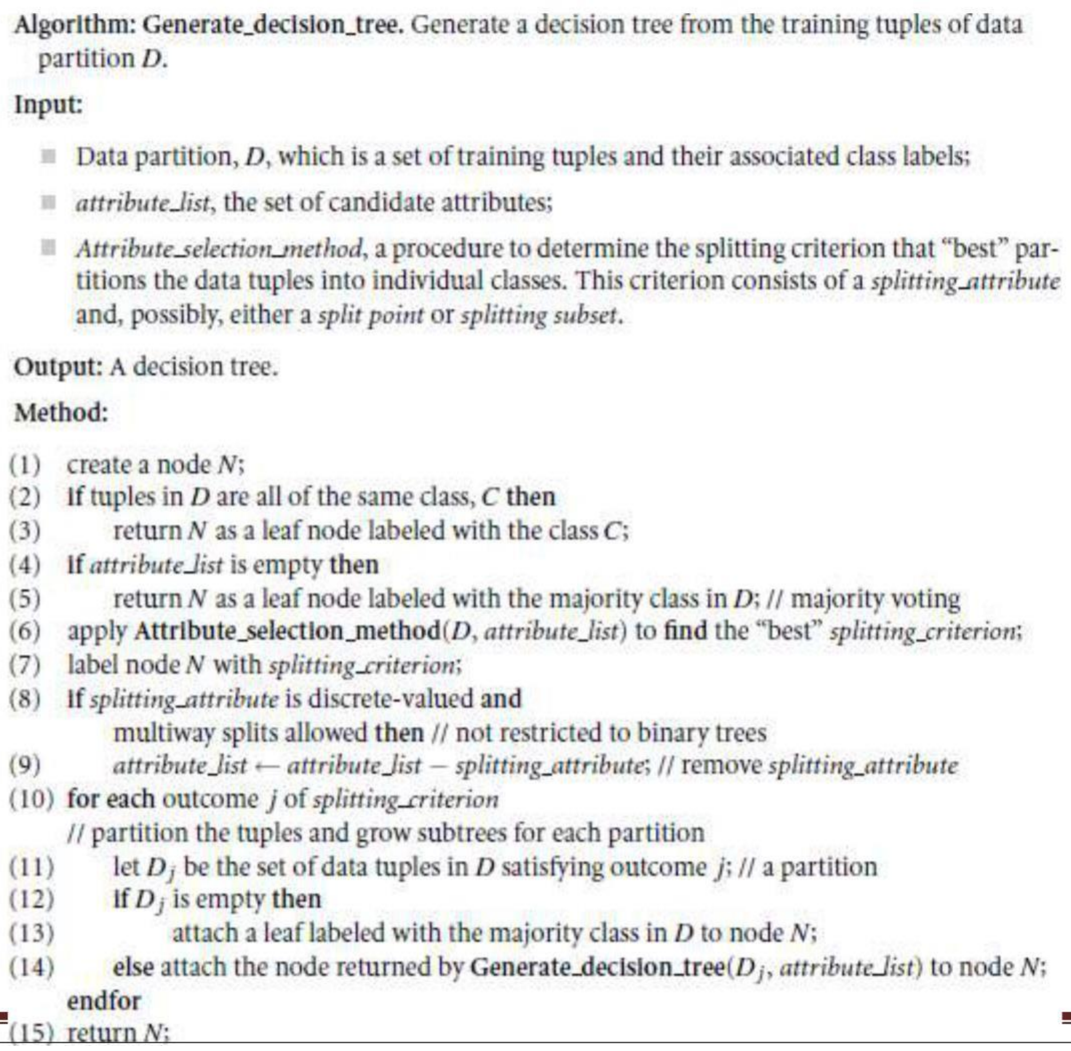
Each internal nodedenotes a test on an attribute. Each branch represents a n outcome of the test. Each leaf node holds a cl ass label.

The topmost node in a tre e is the root node.



- The construction of decisio n treeclassifiers does not require any domain knowledge or parameter setting, and there fore I appropriate for exploratory knowledge discovery.
- Decision trees can handle high dimensional data.
- In general, decision tree classifiers have good accuracy.
- Decision tree induction algorithms have been used for classification in many application areas, such as medicine,manufacturing and production, financial analysis, astronomy, and molecular b iology.

Algorithm For Decision Tree Induction:



Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C then
(3)        return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)        return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
           multiway splits allowed then // not restricted to binary trees
(9)        attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)       let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)       if D_j is empty then
(13)             attach a leaf labeled with the majority class in D to node N;
(14)       else attach the node returned by Generate_decision_tree(D_j, attribute_list) to node N;
     endfor
(15) return N;
```

The algorithm is called with three parameters:

Data partition

Attribute list

Attribute selection method

- The parameter attribute list is a list of attributes describing the tuples.
- Attribute selection method specifies a heuristic procedure for selecting the attri bute that —bestǁ discriminates the given tuples according to class.
- The tree starts as a single node, N, representing the training tuples in D.
- If the tuples in D are all of the same class, then node N becomes a leaf and is labeledwith that class .
- Allof the terminating conditions are explained at the end of the algorithm.
- Otherwise, the algorithm ca lls Attribute selection method to determine the splitting criterion.
- The splitting criterion tells us which attribute to test at node N by determining the —best way to separate or partition the tuples in D into individual classes.

There are three possible scenarios.L et A be the splitting attribute. A has v distinct v alues, {a1, a2, … ,av}, based on the tra ining data.

1 A is discrete-valued:

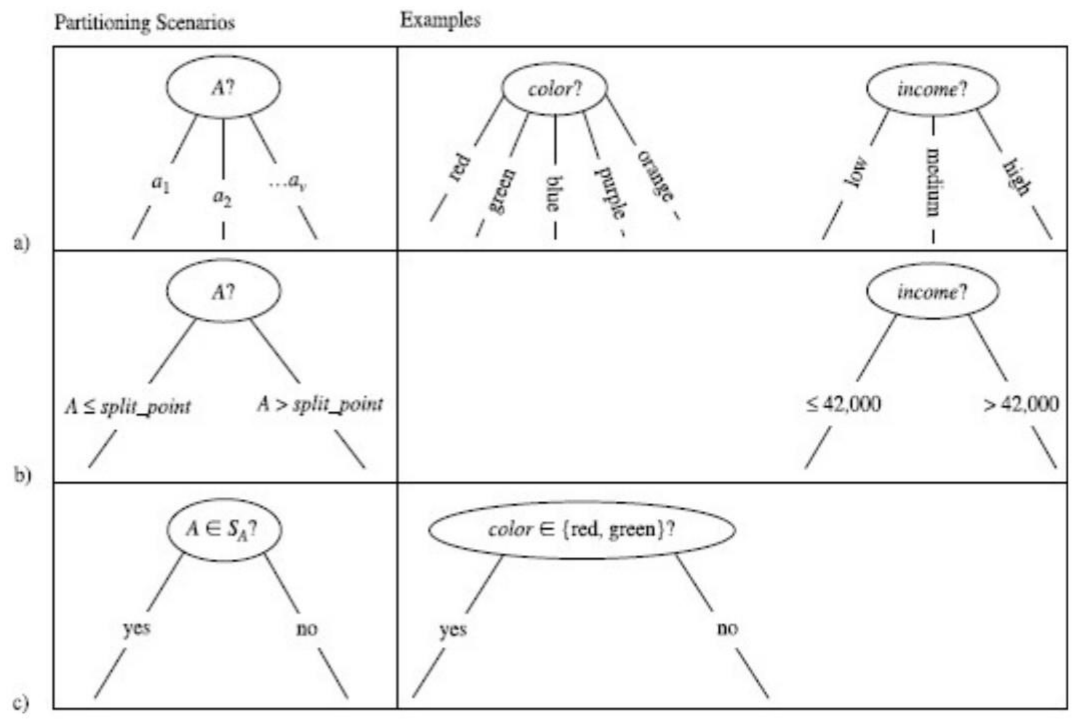
- In this case, the outcomes of t he test at node N corresponddirectly to the known values of A.
- A branch is created for each k nown value, aj, of A and labeled with that • value.

2 A is continuous-valued:

In this case, the test at node N h as two possible outcomes, corresponding to the conditions A <=split point and A >split point, respectively

3 A is discrete-valued and a binary tree must be produced:

If A is discrete-valued and a binary tree must be produced, then the test is of the form A 2 SA, where SA is the Splitting of A.



If A is Discrete valued (b)If A is continuous valued (c) IfA is discrete-valued and a binary tree must be produced:

(b)<u>Attribute Selection Measures</u>

An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes. Information gain

ID3 uses information gain as its attribute selection measure. The attribute with the highest information gain is chosen as the splitting attribute for node N.

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

Info(D) = -m i=1pi log2(pi);

where pi is the probability that an arbitrary tuple inDbelongs to classCi Info(D) is also known as the entropy of D.

How much moreinformation would we still need (after the partitioning) in order to arrive at an exact classification? This amount is measured by

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

The term jDj j jDj acts as the weight of the jth partition
Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the newrequirement (i.e., obtained after partitioning on A). That is,
Gain(A) = Info(D)-InfoA(D):

Table 6.1 Class-labeled training tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Induction of a decision tree using information gain. Table 6.1 presents a training set, D, of class-labeled tuplesThe class label attribute, buys computer, has two distinct values (namely, fyes, nog); therefore, there are two distinct classes (that is, m = 2). Let class C1 correspond to yes and class C2 correspond to no. There are nine tuples of class yes and five tuples of class no. A (root) node N is created for the tuples in D to compute the expected information needed to classify a tuple in D:

the expected information needed to classify a tuple in D if the tuples are partitioned according to age is

Info(D) = -9/14 log2 (9/14)-5/14 log2 5/14 = 0:940 bits:

the expected information needed to classify a tuple in D if the tuples are partitioned according to age is

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ &+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.694 \text{ bits.} \end{aligned}$$

Hence, the gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Similarly, we can compute $Gain(income) = 0.029$ bits, $Gain(student) = 0.151$ bits, and $Gain(credit\_rating) = 0.048$ bits. Because *age* has the highest information gain among the attributes, it is selected as the splitting attribute. Node *N* is labeled with *age*, and

Gain ratio

For example, consider an attribute that acts as a unique identifier, such as product ID. A split on product ID would

result in a large number of partitions (as many as there are values), each one containing just one tuple.

The attribute with the maximum gain ratio is selected as the splitting attribute.

Example 6.2 Computation of gain ratio for the attribute income. A test on income splits the data of Table 6.1 into three partitions, namely low, medium, and high, containing four, six, and

four tuples, respectively. To compute the gain ratio of income, we first use Equation (6.5) to obtain

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right).$$

$$= 0.926.$$

From Example 6.1, we have Gain(income) = 0.029. Therefore, GainRatio(income) =0.029/0.926 = 0.031.

Gini index

The Gini index is used in CART. Usin g the notation described above, the Gini index measures the impurity of D, a data partition or set of training tuples, as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

where pi is the probability that a tuple in D belongs to class Ci and is estimated by jCi,Dj/jDj. The sum is computed over m classes. if income has three possible values, namely flow,medium, highg, then the possible subsets are flow, medium, highg, flow, mediumg, flow,highg, fmedium, highg, flowg, fmediumg, fhighg, and {} if a binary split on A partitions D into D1 and D2, thegini index of D given that partitionin g is

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2).$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

Induction of a decision tree using gini index. LetDbe the training data of Table 6.1where th ere are nine tuples belonging to the class buys computer = yes and the remaining five tuples belong to the clas s buys computer = no. A (root) node N is created for the tuples in D

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

attribute income and consider each of the possible splitting subsets. Consider the subset fl ow, mediumg. This would result in 10 tuples in partition D1 satisfying the condition “income 2 flow, medium

$$\begin{aligned}
&Gini_{income \in \{low, medium\}}(D) \\
&= \frac{10}{14}Gini(D_1) + \frac{4}{14}Gini(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
&= 0.450 \\
&= Gini_{income \in \{high\}}(D).
\end{aligned}$$

.”

Similarly, the Gini index values for splits on the remaining subsets are: 0.315 (for the subs ets flow, highg and fmediumg) and 0.300 (for the subsets fmedium, highg and flowg). Therefore, the best binary split for attribute income is on fmedium, highg (or flowg) because it minimizes the gini index.Evaluating the attribute,we obtain fyouth, seniorg (or fmiddle agedg) as t he best split for agewith a Gini index of 0.375.

Bayesian Classification:

- **Bayesian classifiers are statistical classifiers.**
- **They can predictclass membe rship probabilities, such as the probability that a given tuple belongs toa particular class.**
- **Bayesian classification is based on Bayes’ theorem.**

Bayes' Theorem:

- **Let X be a data tuple. In Bayesian terms, X is considered —evidence.‖and it is descr bed by measurements made on a set of n attributes.**

- **For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the —e vidence‖ or observed data tuple X.**

- **P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.**

- **Bayes' theorem is useful in that it providesa way of calculating the posterior probab ility,**

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

P(H|X), from P(H), P(X|H), an d P(X).

Naïve Bayesian Classification:

The naïve Bayesian classifier, or si mple Bayesian classifier, works as follows:

1. Let D be a training set of tup les and their associated class labels. As usual, each tuple is represented by an n-dimensi onal attribute vector, X = (x1, x2, …,xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, …, An.

2. Suppose that there are m classes, C1, C2, …, Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditi oned on X.

That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Thus we maximize P(CijX). The classCifor which P(CijX) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

3. As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = …= P(Cm), and we would therefore maximize P(X| Ci). Otherwise, we maximize P(X|Ci)P(Ci).

4. Given data sets with many attributes, it would be extremely computationally expensiveto compute P(X|Ci). In order to reduce computation in evaluating P(X|Ci), the

naive assumption of class conditional independence is made. This presumes that the values of the attributes areconditionally independent of one another, given the class label of the tuple. Thus,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$
$$= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i).$$

We can easily estimate the probabilities P(x1|Ci), P(x2|Ci), : : : , P(xn|Ci) fromthe trainingtuples. For eachattribute, we look at whether the attribute is categorical or continuous-valued. Forinstance, to compute P(X|Ci), we consider the following:

If $A_k$is categorical, then P(x_k|Ci) is the number of tuples of class Ciin D havingthe value $x_k$for A_k, divided by |$C_{i,D}$| the number of tuples of class $C_i$in D.

If $A_k$is continuous-valued, then we need to do a bit more work, but the calculationis pretty straightforward.

A continuous-valued attribute is typically assumed tohave a Gaussian distribution with a mean µ and standard deviation , defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

5.In order to predict the class label of X, P(XjCi)P(Ci) is evaluated for each class Ci. The classifier predicts that the class label of tuple X is the class Ciif and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

The training data are in Table 6.1. The data tuples are described by the attributes age, income, student, and credit rating. The class label attribute, buys computer, has two distinct values (namely, fyes, nog). Let C1 correspond to the class buys computer = yes and C2 correspond to buys computer = no. The tuple we wish to classify is X = (age = youth, income = medium, student = yes, credit rating = fair) We need to maximize P(XjCi)P(Ci), for i = 1, 2.

**P(Ci), the prior probability of each
class, can be computed based on the training tuples:
P(buys computer = yes) = 9=14 = 0:643
P(buys computer = no) = 5=14 = 0:357
To compute PXjCi), for i = 1, 2, we compute the following conditional probabilities:
P(age = youth j buys computer = yes) = 2=9 = 0:222
P(age = youth j buys computer = no) = 3=5 = 0:600
P(income = medium j buys computer = yes) = 4=9 = 0:444
P(income = medium j buys computer = no) = 2=5 = 0:400
P(student = yes j buys computer = yes) = 6=9 = 0:667
P(student = yes j buys computer = no) = 1=5 = 0:200
P(credit rating = fair j buys computer = yes) = 6=9 = 0:667
P(credit rating = fair j buys computer = no) = 2=5 = 0:400
Using the above probabilities, we obtain
P(Xjbuys computer = yes) = P(age = youth j buys computer = yes) _
P(income = medium j buys computer = yes) _**

P(student = yes j buys computer = yes) _
P(credit rating = fair j buys computer = yes)
= 0:222_0:444_0:667_0:667 = 0:044.

Similarly,
P(Xjbuys computer = no) = 0:600_0:400_0:200_0:400 = 0:019.
To find the class, Ci, that maximizes P(XjCi)P(Ci), we compute
P(Xjbuys computer = yes)P(buys computer = yes) = 0:044_0:643 = 0:028
P(Xjbuys computer = no)P(buys computer = no) = 0:019_0:357 = 0:007
Therefore, the naïve Bayesian classifier predicts buys computer = yes for tuple X.

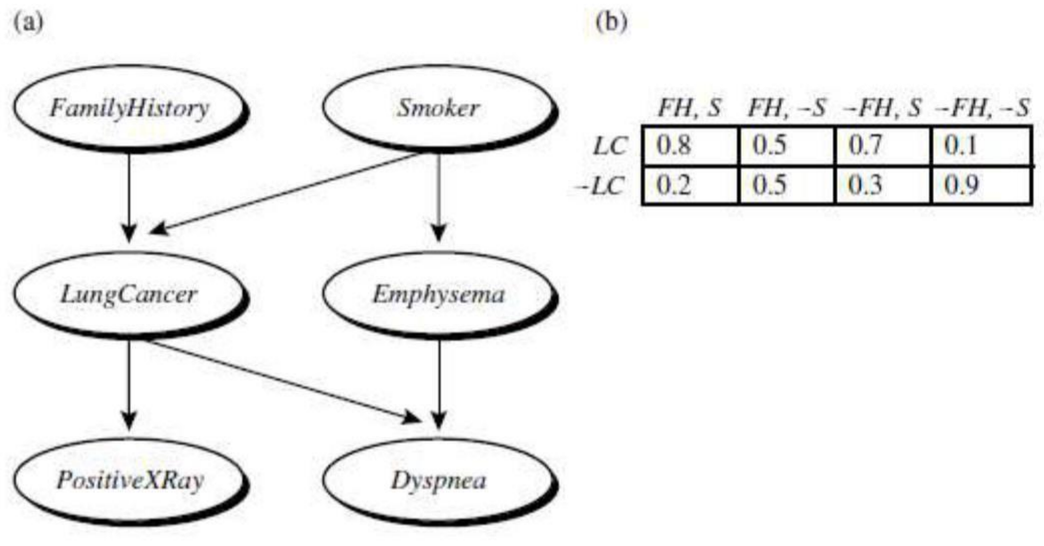
Bayesian Belief Networks

Bayesian belief networks specify joint conditional probability distributions Trained Bayesian belief networks can be used for classification. Bayesian belief networks are also known as belief networks, Bayesian networks, and probabilistic networks. For brevity, we will refer to them as belief networks. A belief network is defined by two components—a directed acyclic graph and a set of conditional probability tables (Figure 6.11). Each node in the directed acyclic graph represents a random variable. The variables may be discrete or continuous-valued. Each arc represents a probabilistic dependence. If an arc is drawn from a node Y to a node Z, thenY is a parent orimmediate predecessor of Z, and Z is a descendant ofY.

PositiveXRay is independent of whether the patient has a family history of lung cancer or is a smoker, given

that we know the patient has lung cancer. In other words, once we know the outcome of the variable LungCancer, then the variables FamilyHistory and Smoker do not provideany additional information regarding PositiveXRay. The arcs also show that the variable LungCancer is conditionally independent of Emphysema, given its parents, FamilyHistory and Smoker. Figure 6.11(b) shows a CPT for the variable LungCancer. The conditional probability for each known value of LungCancer is given for each possible combination of values of its parents. For instance, from the upper leftmost and bottom rightmost entries, respectively, we see that

P(LungCancer = yes j FamilyHistory = yes, Smoker = yes) = 0.8
P(LungCancer = no j FamilyHistory = no, Smoker = no) = 0.9



	FH, S	FH, ¬S	¬FH, S	¬FH, ¬S
LC	0.8	0.5	0.7	0.1
¬LC	0.2	0.5	0.3	0.9

<u>Rule-Based classification:</u>

Using 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-hand side)of a rule isknownas the rule antecedent or precondition.The "THEN"-part (or right-hand side) is the rule consequent. In the rule antecedent, the condition consists of one or more attribute tests (such as 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 (that is, all of 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; ncorrect be the number of tuples correctly classified by R; and jDj be the number of tuples in D. We can define the coverage and accuracy of R as

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

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

Table 6.1. a customer will buy a computer. Consider rule R1 above, which covers 2 of the 14 tuples. It can correctly classify both tuples. Therefore, coverage(R1) = 2/14 = 14:28% and accuracy

(R1) = 2/2 = 100%.

If a rule is satisfied by X, the rule is said to be triggered. For example, suppose we have X=

(age = youth, income = medium, student = yes, credit rating = fair).

We would like to classify X according to buys computer. X satisfies R1, which triggers the rule.If R1 is the only rule satisfied, then the rule fires by returning the class prediction for X.

If more than one rule is triggered, we need a conflict resolution strategy to figure out which rule gets to fire and assign its class prediction to X. There are many possible strategies. We look at two, namely size ordering and rule ordering.The size ordering scheme assigns the highest priority to the triggering rule that has the "toughest" requirements, where toughness is measured by the rule antecedent size. That is, the triggering rule with the most attribute tests is fired.

The rule ordering scheme prioritizes the rules beforehand. The ordering may be classbased or rule-based.With class-based ordering, the classes are sorted in order of decreasing "importance," such as by decreasing order of prevalence. That is, all of the rules for the most prevalent (or most frequent) class come first, the rules for the next prevalent class come next, and so on.

With rule-based ordering, the rules are organized into one long priority list, according to some measure of rule quality such as accuracy, coverage, or size (number of attribute tests in the rule antecedent), or based on advice from domain experts. When rule ordering is used, the rule set is known as a decision list. With rule ordering, the triggering rule that appears earliest in the list has highest priority, and so it gets to fire its class prediction.

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 Extracting classification rules from a decision tree. The decision tree of Figure 6.2 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 6.2 are

R1: IF age = youth AND student = no THEN buys computer = no
R2: IF age = youth AND student = yes THEN buys computer = yes
R3: IF age = middle aged THEN buys computer = yes
R4: IF age = senior AND credit rating = excellent THEN buys computer = yes
R5: IF age = senior AND credit rating = fair THEN buys computer = no

Rule Induction Using a Sequential Covering Algorithm

IF-THEN rules can be extracted directly from the training data (i.e., without having to generate a decision tree first) using a sequential covering algorithm. The name comes from the notion that the rules are learned sequentially (one at a time), where each rule for a given class will ideally cover many of the tuples of that class (and hopefully none of the tuples of other classes). Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.

```
Input:
D, a data set class-labeled tuples;
Att vals, the set of all attributes and their possible values.
Output: A set of IF-THEN rules.
Method:
(1) Rule set = fg; // initial set of rules learned is empty
(2) for each class c do
(3) repeat
(4) Rule = Learn One Rule(D, Att vals, c);
(5) remove tuples covered by Rule from D;
(6) until terminating condition;
(7) Rule set = Rule set +Rule; // add new rule to rule set
(8) endfor
(9) return Rule Set;
```

Figure

Basic sequential covering algorithm.

Classification by Back propagation:

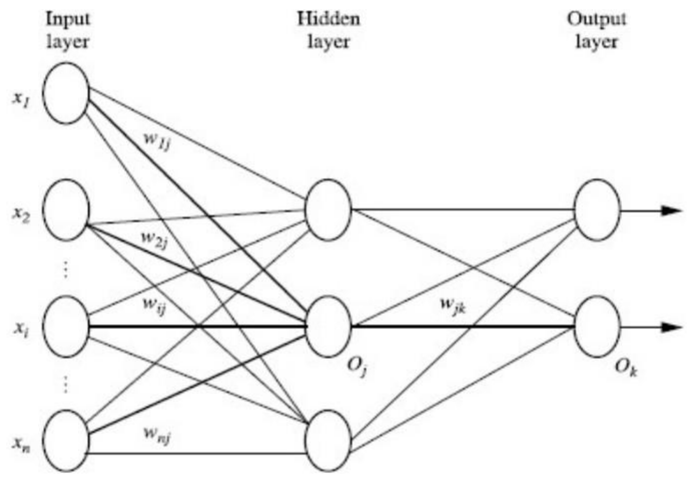
"What is backpropagation?" Backpropagation is a neural network learning algorithm. a neural network is a set of connected input/output units in which each connection has a weight associated with it.During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples. Neural network learning is also referred to as connectionist learning due to the connections between units.

A Multilayer Feed Forward Neural Network:

The backpropagation algorithm perfor ms learning on a multilayer feed-forward neural net work. It iteratively learns a set of weights for prediction of the class label of tuples. A multilayer feed-forward neural network consists of an input layer, one or more hidden layers, and an output layer. An example of a multilayer fe ed-forward network .

Each layer is made up of units. The inp uts to the network correspond to the attributes measured for each training tuple. The inputs are fed simultaneously into the units making up the input layer. These inp uts pass through the input layer and are then weighted and f ed simultaneously to a second layer of “neuronlike” units, known as a hidden layer. The outputs of the hidden layer units can be input to another hidden layer, and so on.The numberof hidden layers is arbitrary, although in p ractice, usually onlyone is used. The weighted outp uts of the last hidden layer are input to units making up the o utput layer, which emits the network’s prediction fo r given tuples. The units in the input layer are called input units . The units in the hidden layers and output layer are sometimes referred to as neurodes, due to theiir symbolic biological basis, or as output units. T he multilayer neural network shown in Figure 6.15 has two layers. of output units. Therefore, we say that it is a two-layer neural network.

The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer. It is fully conne cted in that each unit provides input to each unit in the next forward layer.



Classification by Backpropagation:

- Backpropagation is a neural n etwork learning algorithm.
- A neural network is a set of connected input/output units inwhich each connectio n has a weightassociated with it.
- During the learning phase, th e network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples.
- Neural network learning is al so referred to as connectionist learning due to the connections between units.
- Neural networks involve l ong training times and are therefore more suitable for applicationswhere this is feasible.
- Backpropagation learns by iteratively processing a data set of training tuples, comparing the network’s pr ediction for each tuple with the actual known target value.

- The target value may be t he known class label of the training tuple (for classification problems) or a continuous v alue (for prediction).
- For each training tuple, th e weights are modified so as to minimize the mean squared errorbetween the network's prediction and the actual target value. These modifications are made in the ―backwards‖ direction, that is, from the output layer, throu gh each hidden layer down to the firrst hidden layer hence the name is backpropagation.
- Although it is not guaranteed, in general the weights will eventually converge, and the learning process stops.

Advantages:

- It include their high tolerance of noisy data as well as their ability to classify patterns on which they have not been trained.
- They can be used when you may have little knowledge of the relationships between attributesand classes.
- They are well-suit ed for continuous-valued inputs and outputs, unlike most decision tree algorithms.
- They have been successful on a wide array of real-world data, including handwritten character recognition, pathology and laboratory medicine, and training a com puter to pronounce English text.
- Neural network algorithms ar e inherently parallel; parallelization techniques can be used to speed up the computation pr ocess.
-

Process:

Initialize the weights:

The weights in the network are initialized to small random numbers ranging from-1.0 to 1.0, or -0.5 to 0.5. Each unit has a bias associated with it. The biases are similarly initialized to small random numbers.

Each training tuple, X, is processed by the following steps.

Propagate the inputs forward:

First, the training tuple is fed to the input layer of thenetwork. The inputs pass through the input units, unchanged. That is, for an input unitj, its output, Oj, is equal to its input value, Ij. Next, the net input and output of eachunit in the hidden and output layers are computed. The net input to a unit in the hiddenor output layers is computed as a linear combination of its inputs.

Each such unit has anumber of inputs to it that are, in fact, the outputs of the units conn ected to

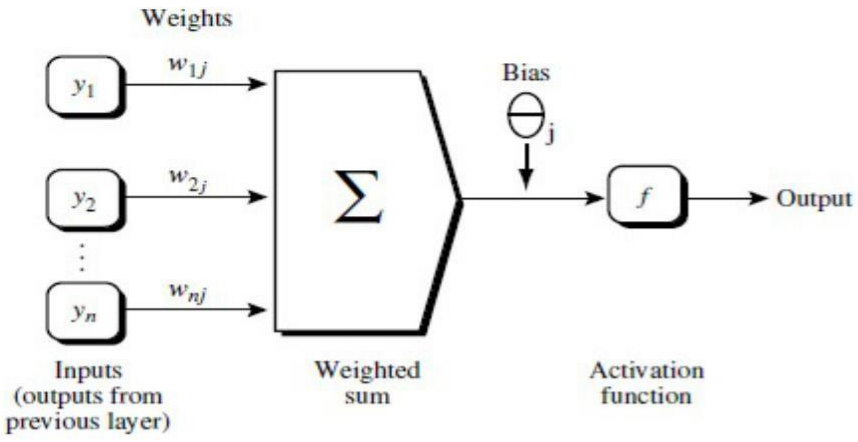
it in theprevious layer. Each connection has a weight. To compute the net input to the unit, each input connected to the unit ismultiplied by its correspondingweight, and this is summed.

$$I_j = \sum_i w_{ij} O_i + \theta_j,$$

where$w_{i,j}$is the weight of the connection from unit iin the previous layer to unit j; $O_i$is the output of unit ifrom the previous layer

Ɵjis the bias of the unit & it actsas a threshold in that it serves to vary the activity of the unit.

Each unit in the hidden and output layers takes its net input and then applies an activation function to it.



Backpropagate the error:

The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction. For a unit j in the output layer, the error Err $_j$is computed by

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

where$O_j$is the actual output of unit j, and $T_j$is the known target value of the giventraining tuple.

The error of a hidden layerunit j is

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

where$w_{jk}$is the weight of the connection from unit j to a unit k in the next higher layer, and$Err_k$is the error of unit k.

Weights are updatedby the following equations, where Dwi j is the change in weight $w_{i\,j}$:

$$\Delta w_{ij} = (l) Err_j O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

. Support Vector Machines

Support Vector Machines, a promising new method for the classification of both linear and nonlinear data. In a nutshell, a support vector machine (or SVM) is an algorithm that works as follows. It uses a nonlinear mapping to transform
the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (that is, a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mappingto a sufficiently high dimension, data from two classes can always be separated bya hyperplane. The SVM finds this hyperplane using support vectors ("essential" training tuples) and margins.

The Case When the Data Are Linearly Separable

Let the data set D be given as (X1, y1),(X2, y2), : : : , (XjDj, yjDj),whereXi is the set of training tupleswith associated class labels, yi.Each yi can take one of two values, either+1 or-1 (i.e., yi belongs{+1,-1}), corresponding to the classes buys computer = yes and buys computer = no, respectively. To aid in visualization, let's consider an example based on two input attributes, A1 and A2, as shown in Figure 6.20. From the graph, we see that the 2-D data are linearly separable (or "linear," for short) because a straight line can be drawn to separate all of the tuples of class+1 from all of the tuples of class-1.
There are an infinite number of separating lines that could be drawn.We want to find the "best" one, that is, one that (we hope) will have the minimum classification error on previously unseen tuples.How can we find this best line?Note that
if our datawere 3-D (i.e.,with three attributes),wewould want to find the best separating plane. Generalizing to n dimensions, we want to find the best hyperplane.We will use the term"hyperplane" to refer to the decision boundary that we are seeking, regardless of the number of input attributes. So, in other words, how can we find the best hyperplane? An SVM approaches this problem by searching for the maximum marginal hyperplane.

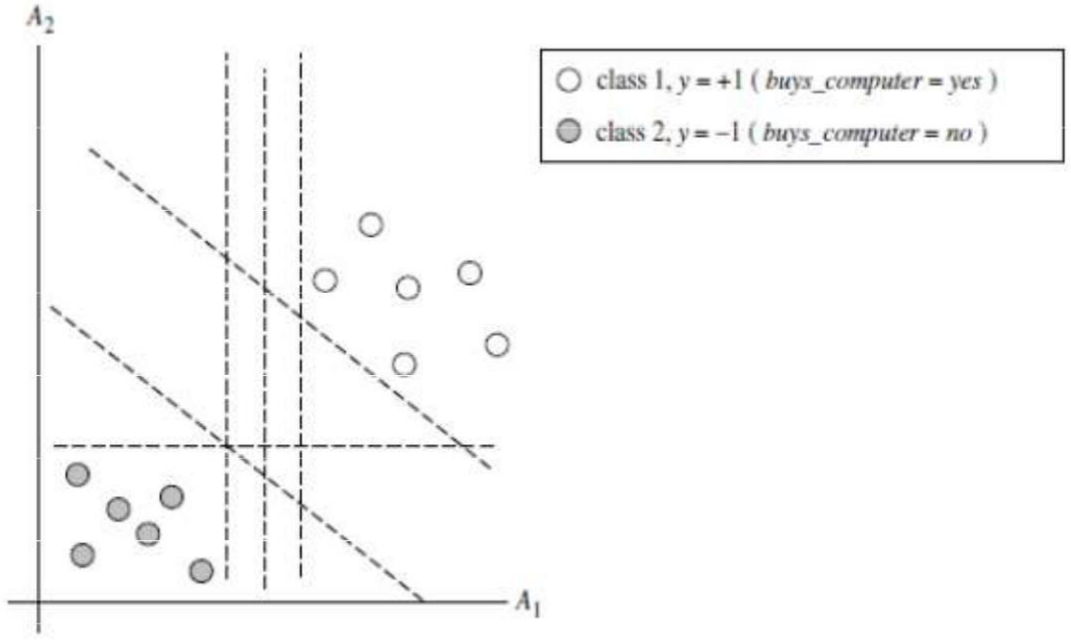


Figure 6.20 The 2-D training data are linearly separable. There are an infinite number of (possible) separating hyperplanes or "decision boundaries." Which one is best?

Hyper plane with the larger margin to be more accurate at classifying future data tuples than the hyperplane with the

smaller margin. This is why (during the learning or training phase), the SVM searches for the hyperplane with the largest margin, that is, the maximum marginal hyper plane (MMH). The associated margin gives the largest separation between classes. Getting to an informal definition of margin, we can say that the shortest distance from a hyperplane to one side of its margin is equal to the shortest distance from the hyper plane to the other side of its margin, where the "sides" of the margin are parallel to the hyperplane. When dealing with the MMH, this distance is, in fact, the shortest distance from the MMH to the closest training tuple of either class. A separating hyper plane can be written as

W_X+b = 0;

whereW is a weight vector, namely,W = fw1, w2, : : : , wng; n is the number of attributes; and b is a scalar, often referred to as a bias. To aid in visualization, let's consider two input attributes, A1 and A2.

Training tuples are 2-D, e.g., X = (x1, x2), where x1 and x2 are the values of attributes A1 and A2, respectively, for X. If we think of b as an additional weight, w0, we can rewrite the above separating hyper plane as

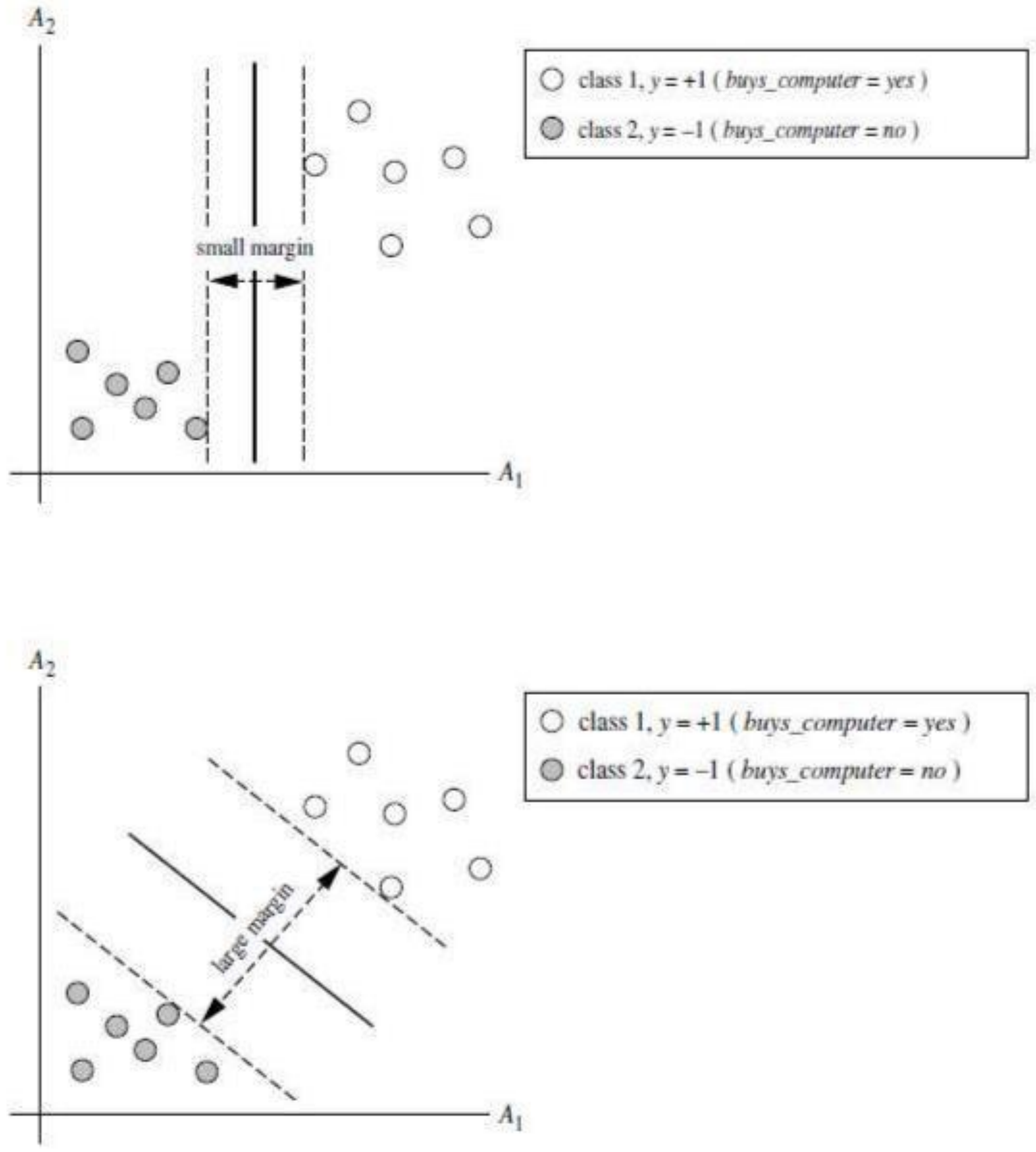
w0+w1x1+w2x2 = 0:

Thus, any point that lies above the separating hyper plane

satisfies w0+w1x1+w2x2 > 0:

Similarly, any point that lies below the separating hyper plane

satisfies w0+w1x1+w2x2 < 0:


A2
class 1, y = +1 (buys_computer = yes)
class 2, y = –1 (buys_computer = no)
small margin
A1
A2
class 1, y = +1 (buys_computer = yes)
class 2, y = –1 (buys_computer = no)
large margin
A1

ure 6.21 Here we see just two possible separating hyperplanes and their associated margins. Which one is better? The one with the larger margin should have greater generalization accuracy.

Lazy Learners (or Learning from Your Neighbors):

The classification methods is a decision tree induction, Bayesian classification, rule-based classification, classification by back propagation, support vector machines, and classification based on association rule mining—are all examples of eager learners. Eager learners, when given a set of training tuples, will construct a generalization (i.e., classification) model before receiving new (e.g., test) tuples to classify.

k-Nearest-Neighbor Classifiers

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple

with training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n-dimensional space. In this way, all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k "nearest neighbors" of the unknown tuple. "Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, say, X1 = (x11, x12, : : : , x1n) and X2 = (x21, x22, : : : , x2n), is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1 and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count.
Typically, we normalize the values of each attribute before using Equation (6.45). This helps prevent attributes with initially large ranges (such as income) from out weighing attributes with initially smaller ranges (such as binary attributes). Min-max normalization, for example, can be used to transforms value v of a numeric attribute A to v0 in the range [0, 1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

where minA and maxA are the minimum and maximum values of attribute A.

For k-nearest-neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors. When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space. Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple. "But how can distance be computed for attributes that not numeric, but categorical, such as color?" The above discussion assumes that the attributes used to describe the tuples are all numeric. For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple X1 with that in tuple X2. If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0.If the two are different (e.g., tuple X1 is blue but tuple X2 is red), then the difference is considered to be 1.

"What about missing values?" In general, if the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference. Suppose that each of the attributes have been mapped to the range [0, 1]. For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing. If A is numeric and missing from both tuples X1 and X2, then the difference is also taken to be 1.

Case-Based Reasoning

Case-based reasoning (CBR) classifiers use a database of problem solutions to solve new problems. Unlike nearest-neighbor classifiers, which store training tuples as points in Euclidean space, CBR stores the tuples or "cases" for problem solving. Business applications of CBR include problem resolution for
customer service help desks, where cases describe product-related diagnostic problems. CBR has also