

# DATA WAREHOUSING

RAMESWARA REDDY.K.V  
ASST.PROFESSOR  
CSE DEPARTMENT

# What is Data Warehousing

Data Warehousing is an architectural construct of information systems that provides users with current and historical decision support information that is hard to access or present in traditional operational data stores

## The need for data warehousing

- Business perspective
  - In order to survive and succeed in today's highly competitive global environment
- Decisions need to be made quickly and correctly
- The amount of data doubles every 18 months, which affects response time and the sheer ability to comprehend its content
- Rapid changes

# Business Problem Definition

Providing the organizations with a sustainable competitive

Advantage

- Customer retention
- Sales and customer service
- Marketing
- Risk assessment and fraud detection

# Business problem and data warehousing

Classified into

## **Retrospective analysis:**

Focuses on the issues of past and present events.

## **Predictive analysis:**

Focuses on certain events or behavior based on historical information

Further classified into

## **Classification:**

Used to classify database records into a number of predefined classes based on certain criteria.

## **Clustering:**

Used to segment a database into subsets or clusters based on a set of attributes

## **Association**

It identify affinities among the collection as reflected in the examined records.

## **Sequencing**

This techniques helps identify patterns over time, thus allowing , for example, an analysis of customers purchase during separate visits.

## **Operational and Informational Data Store**

### **Operational Data**

Focusing on transactional function such as bank card withdrawals and deposits

- Detailed
- Updateable
- Reflects current



ODS

Data warehouse

---

volatile

nonvolatile

every current data

current and historical data

detailed data

precalculated summaries

---

## **Informational Data**

Informational data, is organized around subjects such as customer, vendor, and product. What is the total sales today?.

Focusing on providing answers to problems posed by decision makers

- Summarized
- Nonupdateable

## **Operational data store.**

An operational data store (ODS) is an architectural concept to support day-to-day operational decision support and constrains current value data propagated from operational applications.

A data warehouse is a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions.  
[WH Inmon]

## **Subject Oriented**

Data warehouses are designed to help to analyze the data. For example, to learn more about your company's sales data, building a warehouse that concentrates on sales

## **Integrated**

The data in the data warehouse is loaded from different sources that store the data in different formats and focus on different aspects of the subject. The data has to be checked, cleansed and transformed into a unified format to allow easy and fast access.

## **Nonvolatile**

Nonvolatile means that, once entered into the warehouse, data should not change. After inserting data in the data warehouse it is neither changed nor removed. Data warehouse requires two operations in data accessing

- Initial loading of data
- Access of data

## **Time Variant**

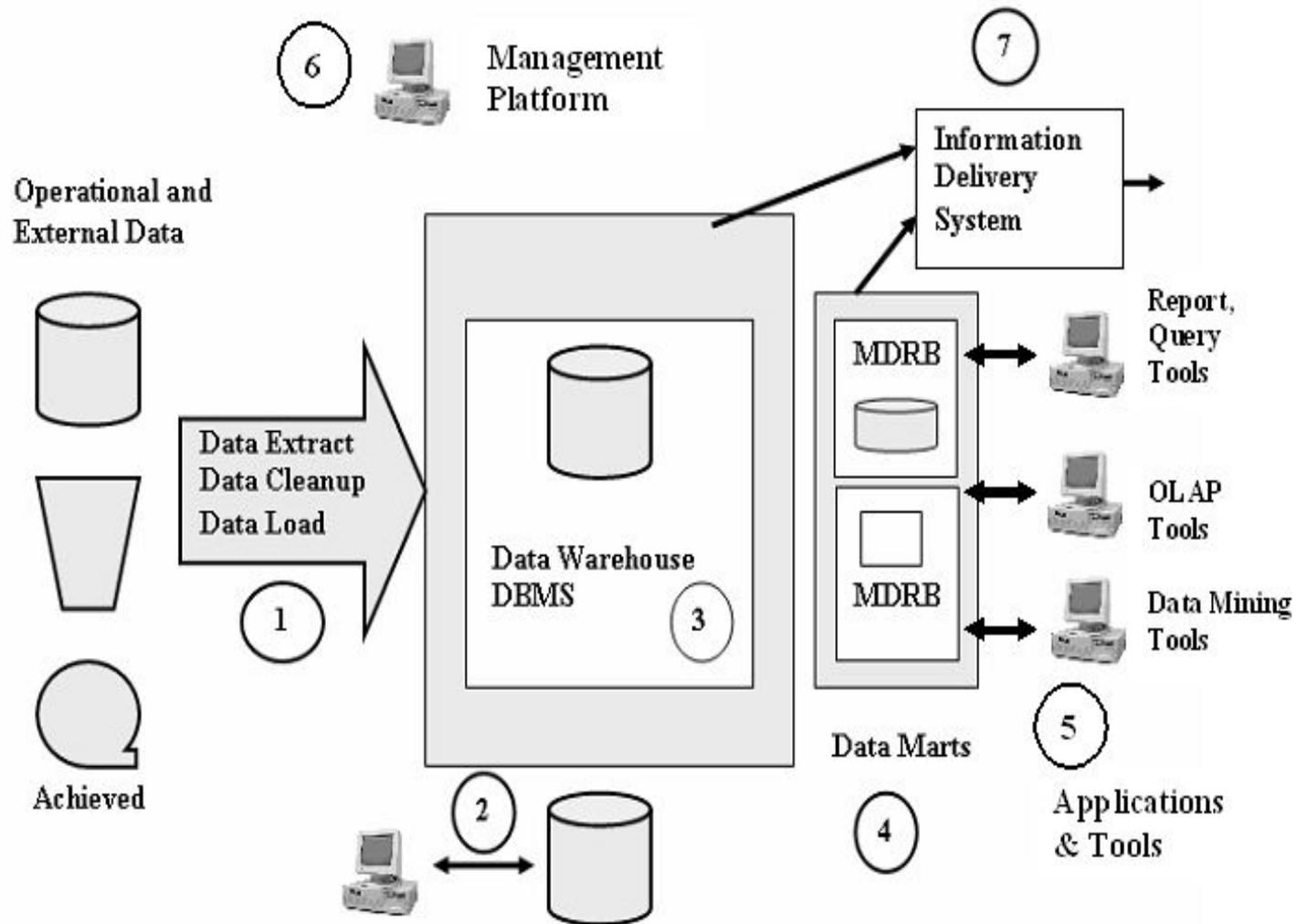
In order to discover trends in business, analysts need large amounts of data. A data warehouse's focus on change over time is what is meant by the term time variant.

Provides information from historical perspective

# Data Warehouse Architecture

Seven data warehouse components

- Data sourcing, cleanup, transformation, and migration tools
- Metadata repository
- Warehouse/database technology
- Data marts
- Data query, reporting, analysis, and mining tools
- Data warehouse administration and management
- Information delivery system



# Data Warehousing Components

Operational data and processing is completely separate from data warehouse processing.

## Data Warehouse Database

It is an important concept (Marked as 2 in the diagram) in the Warehouse environment.

In addition to transaction operation such as ad hoc query processing, and the need for flexible user view creation including aggregation, multiple joins, and drill-down.

- Parallel relational database designs that require a parallel computing platform.
- Using new index structures to speed up a traditional RDBMS.
- Multidimensional database (MDDBS) that are based on proprietary database technology or implemented using already familiar RDBMS.

## Sourcing, Acquisition, Cleaning, and Transformation tools

To perform all of the conversions, summarizations, key changes, structural changes, and condensations needed to transform disparate data into information

- Removing unwanted data from operational database
- Converting to common data names and definitions
- Calculating summarizes and derived data.
- Establishing default for missing data.
- Accommodating source data definition changes.
  - *Database heterogeneity.* DBMS are very different in data model, data access language, data navigation, operation, concurrency, integrity, recovery etc.,.
  - *Data heterogeneity.* This is the difference in the way data is defined and used in different models, different attributes for the same entity.

# Metadata

data about data

Used for building, maintaining, and using the data warehouse

Classified into

## Technical metadata

About warehouse data for use by warehouse designers and administrators when carrying out warehouse development and management tasks

- Information about data sources
- Transformation, descriptions, i.e., the mapping methods from operational databases into the warehouse and algorithms used to convert, enhance or transform data.
- Warehouse objects and data structure definitions for data targets.
- The rules used to perform data cleanup and data enhancement.

- Data mapping operations when capturing data from source systems and applying to the target warehouse database.
- Access authorization, backup history, archive history, information delivery history, data acquisition history, data access etc.,

## Business metadata

Gives perspective of the information stored in the data warehouse

- Subject areas and information object type, including queries, reports, images, video, and / or audio clips.
- Internet home pages.
- Other information to support all data warehouse components.
- Data warehouse operational information e.g., data history, ownership, extract, audit trail, usage data.

Metadata management is provided via a metadata repository and accompanying software.

The important functional components of the metadata repository is the information directory. This directory helps integrate, maintain, and view the contents of the data warehousing system

## **Access Tools**

Front-end tools, ad hoc request, regular reports, and custom applications are the primary delivery of the analysis.

*Alerts*, which let a user know when a certain event has occurred

The tools divided into five main groups.

- Data query and reporting tools
- Application development tools
- Executive information system (EIS) tools
- On-line analytical processing tools
- Data mining tools

## Query and reporting tools

This category can be further divided into two groups.

- Reporting tools
- Managed query tools

Reporting tools can be divided into production reporting tools and desktop report writers.

- Production reporting tools will let companies generate regular operational reports or support high-volume batch jobs.
- Report writers, on the other hand, are inexpensive desktop tools designed for end users.

Managed query tools shield end users from the complexities of SQL and database structures by inserting a metalayer between users and the database

## Applications

Applications developed using a language for the users

# OLAP

**Based on the concepts of multidimensional database**

## **Data mining**

To discovery meaningful new correlations, patterns, and trends by digging into (mining) large amount of data stored in warehouse using artificial-intelligence (AI) and statistical and mathematical techniques

*Discover knowledge.* The goal of knowledge discovery is to determine the following things.

- Segmentation
- Classification
- Association
- Preferencing

*Visualize data.* Prior to any analysis, the goal is to “humanize” the mass of data they must deal with and find a clever way to display the data.

*Correct data.* While consolidating massive database may enterprise find that the data is not complete and invariably contains erroneous and contradictory information. Data mining techniques can help identify and correct problems in the most consistent way possible.

## **Data visualization**

Presenting the output of all the previously mentioned tools  
Colors, shapes, 3-D images, sound, and virtual reality

## **Data Marts**

Data store that is subsidiary to data warehouse

It is partition of data that is created for the use of dedicated group of users

Placed on the data warehouse database rather than placing it as separate store of data.



In most instance, the data mart is physically separate store of data and is normally resident on separate database server.

1. Extremely urgent user requirements.
2. The absence of a budget for a full dwh strategy.
3. The absence of a sponsor for an enterprise wide decision support strategy.
4. The decentralization of business units.
5. The attraction of easy to use tools and a mind sized project.

## **Data Warehouse administration and Management**

- Managing data warehouse includes
- Security and priority management
- Monitoring updates form multiple sources
- Data quality checks
- Managing and updating metadata
- Auditing and reporting data warehouse usage and status
- Replicating, sub setting, and distributing data
- Backup and recover
- Data warehouse storage management

## **Information delivery system**

The information delivery system distributes warehouse stored data and other information objects to other data warehouse and end-user products such as spread sheets and local databases.

Delivery of information may be based on time of day, or a completion of an external event.

# Building a Data Warehouse

## Business considerations

## Return on Investment

## Approach

The information scope of the data warehouse varies with the business requirements, business priorities, and magnitude of the problem

Two data warehouses

Marketing

Personnel

- The top-down approach
  - Building an enterprise data warehouse with subset data marts.
- The bottom-up approach
  - Resulted in developing individual data marts, which are then integrated into the enterprise data warehouse.

## **Organizational issues**

A data warehouse implementation is not truly a technological issue; rather, it should be more concerned with identifying and establishing information requirements, the data sources fulfill these requirements, and timeliness.

## **Design considerations**

A data Warehouse's design point is to consolidate from multiple, often heterogeneous sources into a query database. The main factors include

- Heterogeneity of data sources, which affects data conversion, quality, timeliness
- Use of historical data, which implies that data may be “old”.
- Tendency of databases to grow very large

## **Data content**

- A data warehouse may contain details data, but the data is cleaned up and transformed to fit the warehouse model, and certain transactional attributes of the data are filtered out.
- The content and structure of the data warehouse are reflected in its data model. The data model is the template that describes how information will be organized within the integrated warehouse framework.

## **Metadata**

A data warehouse design should ensure that there is mechanism that populates and maintains the metadata repository, and that all access paths to the data warehouse have metadata as an entry point.

## **Data distribution**

One of the challenges when designing a data warehouse is to know how the data should be divided across multiple servers and which users should get access to which types of data.



The data placement and distribution design should consider several options, including data distribution by subject area, location, or time.

## **Tools**

Each tool takes a slightly different approach to data warehousing and often maintain its own version of the metadata which is placed in a tool-specific, proprietary metadata repository.

The designers of the tool have to make sure that all selected tools are compatible with the given data warehouse environment and with each other.

## **Performance considerations**

Rapid query processing is highly desired feature that should be designed into the data warehouse.

Design warehouse database to avoid the majority of the most expensive operations such as multitable search and joins

## **Nine decisions in the design of data warehouse**

1. Choosing the subject matter.
2. Deciding what a fact table represents.
3. Identifying and confirming the dimensions.
4. Choosing the facts.
5. Storing precalculations in the fact table.
6. Rounding out the dimension tables.
7. Choosing the duration of the database.
8. The need to track slowly changing dimensions.
9. Deciding the query priorities and the query modes

## Technical Considerations

- The hardware platform that would house the data warehouse
- The database management system that supports the warehouse database.
- The communications infrastructure that connects the warehouse, data marts, operational systems, and end users.
- The hardware platform and software to support the metadata repository.
- The systems management framework that enables centralized management and administration. of the entire environment

## Hardware platforms

Data warehouse server is its capacity for handling the volumes of data required by decision support applications, some of which may require a significant amount of historical data.

This capacity requirement can be quite large

The data warehouse residing on the mainframe is best suited for situations in which large amounts of data

The data warehouse server has to be able to support large data Volumes and complex query processing.

### **Balanced approach.**

An important design point when selecting a scalable computing platform is the right balance between all computing components

### **Data warehouse and DBMS specialization**

The requirements for the data warehouse DBMS are performance, throughput, and scalability because the database large in size and the need to process complex ad hoc queries in a relatively in short time.

The database that have been optimized specifically for data warehousing.

## Communications infrastructure

Communications networks have to be expanded, and new hardware and software may have to be purchased to meet out the cost and efforts associated with bringing access to corporate data directly to the desktop.

## Implementation Considerations

Data warehouse implementation requires the integration of many products within a data warehouse.

- The steps needed to build a data warehouse are as follows.
- Collect and analyze business requirements.
- Create a data model and a physical design for the data warehouse.
- Define data warehouse.
- Choose the database technology and platform for the warehouse.
- Extract the data from the operational databases, transform it, clean it up, and load it into the database.

- Choose the database access and reporting tools.
- Choose database connectivity software.
- Choose data analysis and presentation software.
- Update the data warehouse.

### **Access tools**

Suit of tools are needed to handle all possible data warehouse access needs and the selection of tools based on definition of deferent types of access to the data

- Simple tabular form reporting.
- Ranking.
- Multivariable analysis.
- Time series analysis.
- Data visualization, graphing, charting and pivoting.
- Complex textual search.

- Statistical analysis.
- Artificial intelligence techniques for testing of hypothesis, trend discovery, definition and validation of data cluster and segments.
- Information mapping
- Ad hoc user-specified queries
- Predefined repeatable queries
- Interactive drill-down reporting and analysis.
- Complex queries with multitable joins, multilevel sub queries, and sophisticated search criteria.

## **Data extraction, cleanup, transformation and migration**

Data extraction decides the ability to transform, consolidate, integrate, and repair the data should be considered

- The ability to identify data in the data source environments that can be read by the conversion tool is important
- Support for flat files, indexed files
- The capability to merge data from multiple data stores is required in many installations.
- The specification interface to indicate the data to be extracted and conversion criteria is important.
- The ability to read information from data dictionaries or import information from repository products is desired.
- The code generated by the tool should be completely maintainable from within the development environment.
- Selective data extraction of both data elements and records enables users to extract only the required data.

- A field-level data examination for the transformation of data into information is needed.
- The ability to perform data-type and character-set translation is a requirement when moving data between incompatible systems.
- The capability to create summarization, aggregation, and derivation records and fields is very important
- The data warehouse database management should be able to perform the load directly from the tool, using the native API available with the RDBMS.
- Vendor stability and support for the product are items that must be carefully evaluated.

## **Data placement strategies**

As a data warehouse grows, there are at least two options for data placement. One is to put some of the data in the data warehouse into another storage media e.g., WORM, RAID, or photo-optical technology.

The second option is to distribute the data in the data warehouse across multiple servers

## **Data replication**

Data that is relevant to a particular workgroup in a localized database can be a more affordable solution than data warehousing

Replication technology creates copies of databases on a periodic bases, so that data entry and data analysis can be performed separately

## **Metadata**

Metadata is the roadmap to the information stored in the warehouse

The metadata has to be available to all warehouse users in order to guide them as they use the warehouse.

## User sophistication levels

*Casual users*

*Power users.*

*Experts*

## Integrated Solutions

A number of vendors participated in data warehousing by providing a suit of services and products that go beyond one particular Component of the data warehouse.

**Digital Equipment Corp.** Digital has combined the data modeling, extraction and cleansing capabilities of Prism Warehouse Manager with the copy management and data replication capabilities of Digital's ACCESSWORKS family of database access servers in providing users with the ability to build and use information warehouse

**Hewlett-Packard.** Hewlett-Packard's client/server based HP open warehouse comprises multiple components, including a data management architecture, the HP-UX operating system HP 9000 computers, warehouse management tools, and the HP information Access query tool

● **IBM.** The IBM information warehouse framework consists of an architecture; data management tools; OS/2, AIX, and MVS operating systems; hardware platforms, including mainframes and servers; and a relational DBMS (DB2).

● **Sequent.** Sequent computer systems Inc.'s DecisionPoint Program is a decision support program for the delivery of data warehouses dedicated to on-line complex query processing (OLCP). Using graphical interfaces users query the data warehouse by pointing and clicking on the warehouse data item they want to analyze. Query results are placed on the program's clipboard for pasting onto a variety of desktop applications, or they can be saved on to a disk.

## **Benefits of Data Warehousing**

Data warehouse usage includes

- Locating the right information
- Presentation of Information (reports, graphs).
- Testing of hypothesis
- Sharing and the analysis

## Tangible benefits

- Product inventory turnover is improved
- Cost of product introduction are decreased with improved selection of target markets.
- More cost-effective decision making is enabled by increased quality and flexibility of market analysis available through multilevel data structures, which may range from detailed to highly summarized.
- Enhanced asset and liability management means that a data warehouse can provide a “big” picture of enterprise wide purchasing and inventory patterns.

## **Intangible benefits**

The intangible benefits include.

- Improved productivity, by keeping all required data in a single location and eliminating the redundant processing
- Reduced redundant processing.
- Enhance customer relations through improved knowledge of individual requirement and trends.
- Enabling business process reengineering.

## Relational Database Technology for Data Warehouse

The Data warehouse environment needs

- Speed up
- Scale-p

Parallel hardware architectures, parallel operating systems and parallel database management systems will provide the requirement of warehouse environment.

### Types of parallelism

#### *Interquery parallelism*

Threads (or process) handle multiple requests at the same time.

#### *Intraquery parallelism*

scan, join, sort, and aggregation operations are executed concurrently in parallel.

Intraquery parallelism can be done in either of two ways

### *Horizontal parallelism*

Database is partitioned across multiple disks, and parallel processing occurs within a specific task that is performed concurrently on different sets of data.

### *Vertical parallelism*

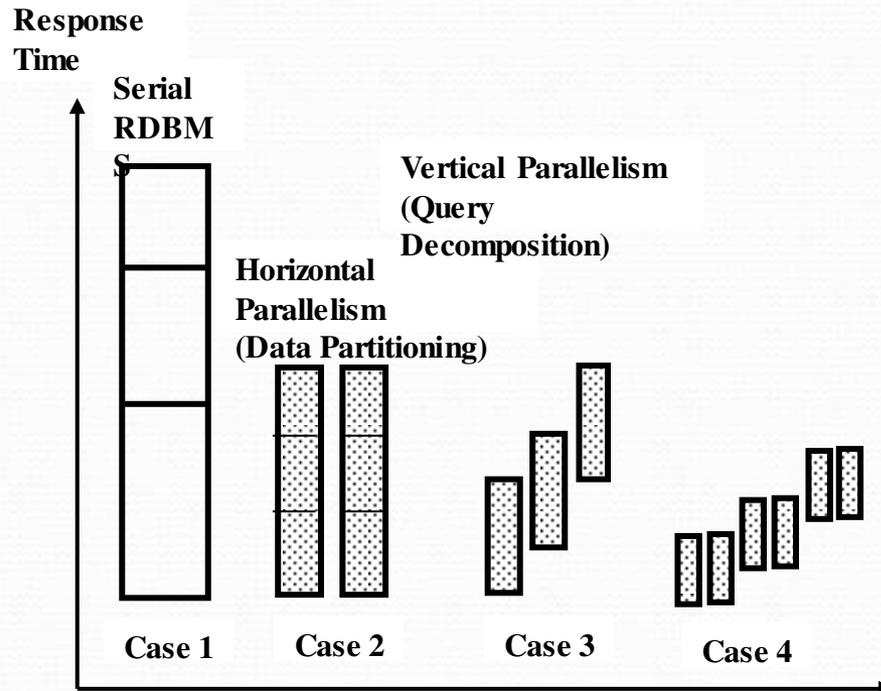
An output from one task (e.g., scan) becomes an input into another task (e.g., join) as soon as records become available.

## **Data Partitioning**

Spreads data from database tables across multiple disks so that I/O operations such as read and write can be performed in parallel.

### *Random partitioning*

It includes data striping across multiple disks on a single server. Another option for random partitioning is round-robin partitioning. In which each new record is placed on the next assigned to the database.



## *Intelligent partitioning*

DBMS knows where a specific record is located and does not waste time searching for it across all disks.

- *Hash partitioning*. A hash algorithm is used to calculate the partition number (hash value) based on the value of the partitioning key for each row.

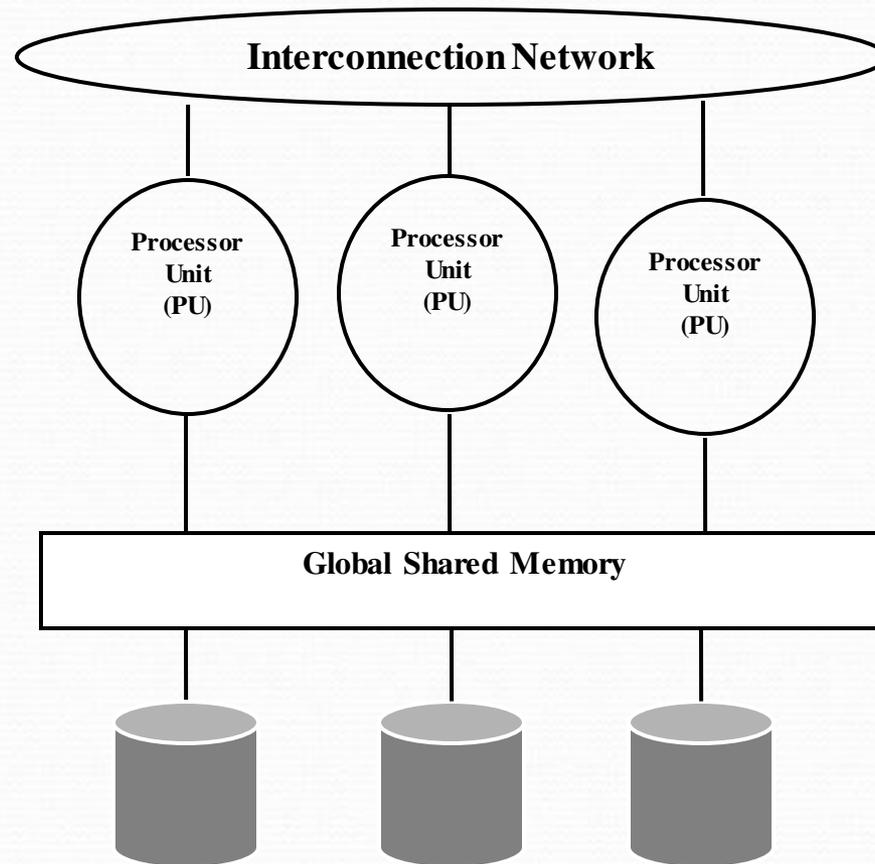
- *Key range partitioning*. Rows are placed and located in the partitions according to the value of the partitioning key (all rows with the key value form A to K are in partition 1, L to T are in partition 2 etc.).
- *Schema partitioning*. an entire table is placed on one disk, another table is placed on a different disk, etc. This is useful for small reference tables that are more effectively used when replicated in each partition rather than spread across partitions.
- *User-defined partitioning*. This is a partitioning method that allows a table to be partitioned on the basis of a user-defined expression.

## **Database Architecture for Parallel Processing**

### **Shared-memory architecture- SMP (Symmetric Multiprocessors)**

Multiple database components executing SQL statements communicate with each other by exchanging messages and data via the shared memory.

Scalability can be achieved through process-based multitasking or thread-based multitasking.



## Shared-disk architecture

The entire database shared between RDBMS servers, each of which is running on a node of a distributed memory system.

Each RDBMS server can read, write, update, and delete records from the same shared database

Implemented by using distribute lock manager (DLM)

### **Disadvantage.**

All nodes are reading and updating the same data, the RDBMS and its DLM will have to spend a lot of resources synchronizing multiple buffer pools.

It may have to handle significant message traffic in a highly utilized REBMS environment.

## **Advantages.**

It reduce performance bottlenecks resulting from data skew (an uneven distribution of data), and can significantly increases system availability.

It eliminates the memory access bottleneck typical of large SMP systems, and helps reduce DBMS dependency on data partitioning.

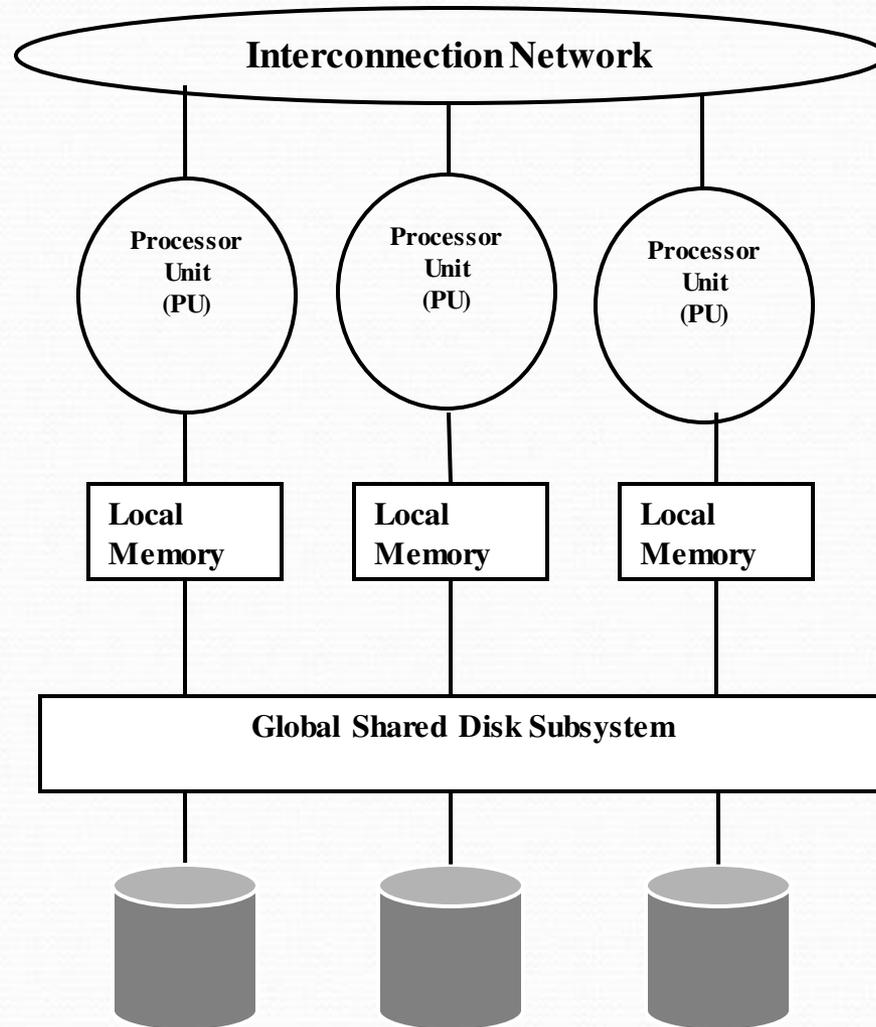


Figure 4.3 Distributed-memory shared-disk architecture

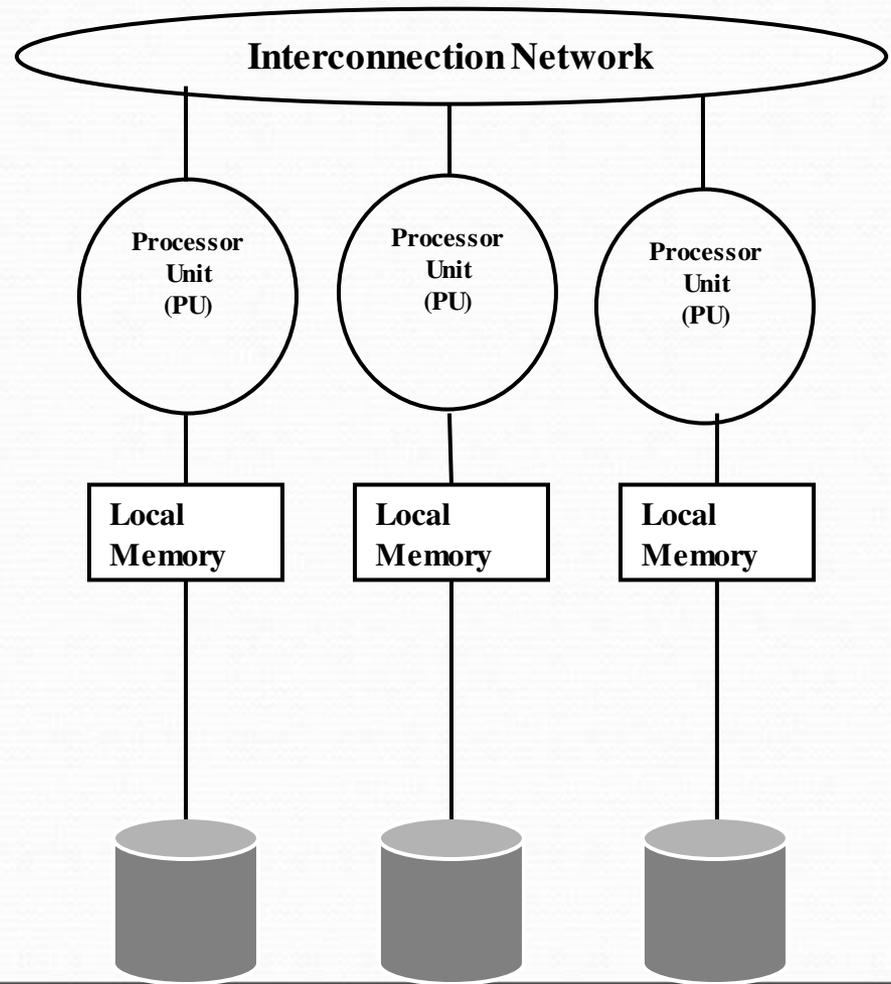
## Shared-nothing architecture

Each processor has its own memory and disk, and communicates with other processors by exchanging messages and data over the interconnection network.

### Disadvantages.

It is most difficult to implement.

It requires a new programming paradigm



## **Combined architecture**

Combined hardware architecture could be a cluster of SMP nodes combined parallel DBMS architecture should support interserver parallelism of distributed memory MPPs and intraserver parallelism of SMP nodes.

## **Parallel RDBMS features**

*Scope and techniques of parallel DBMS operations*

*Optimizer implementation*

*Application transparency*

*The parallel environment*

*DBMS management tool*

## Alternative Technologies

Number of vendors are working on other solutions improving performance in data warehousing environments.

- Advanced database indexing products
- Specialized RDBMSs designed specifically for data warehousing.
- Multidimensional databases

SYBASE IO is an example of a product that uses a bitmapped index structure of the data stored in the SYBASE DBMS.

# Parallel DBMS Vendors

## Oracle

Oracle supports parallel database processing with its add-on oracle parallel server option(OPS) and parallel query option(PQO)

### **Architecture.**

Virtual shared-disc capability.

Process-based approach

Facilitate the inter query parallelism

PQO supports parallel operations such as index build, database load, backup, and recovery.

### **Data partitioning**

It supports random striping of data across multiple disks.

Oracle supports dynamic data repartitioning

## **Parallel operations**

Generates a parallel plan

The oracle PQO query coordinator breaks the query into sub queries

Parallelize the creation of indexes, database load, backup, and recovery

PQO supports both horizontal and vertical parallelism

## **Informix**

### **Architecture.**

Support shared-memory, shared-disk, and shared-nothing models.  
It is thread based architecture.

### **Data partitioning.**

Round-robin, schema, charts, key range, and user-defined partitioning methods . Both data and index can be partitioned

## **Parallel Operations.**

Executes queries in parallel.

## **IBM**

Client/Server database product-DB2 parallel Edition

## **Architecture.**

DB2 PE is a shared-nothing architecture in which all data is partitioned across processor nodes.

Each node is aware of the other nodes and how the data is partitioned

## **Data partitioning.**

Allow a table to span multiple nodes.

The master system catalog for each database is stored on one node and cached on every other node.

## **Parallel operations.**

All database operations are fully parallelized

## **Sybase**

Sybase has implemented its parallel DBMS functionality in a parallel product called DYBASE MPP.

## **Architecture.**

It is a shared-nothing systems that partitions data across multiple SQL servers and supports both function shipping and data repartitions.

Open server application that operates on top of existing SQL servers.

All the knowledge about the environment, data partitions, and parallel query execution is maintained by SYBASE MPP software.

SYBASE MPP consists of specialized servers.

- Data server the smaller executable unit of parallelism that consists of SQL server, split server (performs joins across nodes), and control server (coordination of execution and communication)
- DBA server handles optimization, DDL statements, security and global systems catalog.
- Administrative server a graphical user interface for managing SYBASE MPP.

### **Data partitioning.**

supports hash, key range, and schema partitioning, indexes partitioning.

### **Parallel operations.**

All SQL statements and utilities in parallel across SQL servers

### **Microsoft**

SQL server architecture is shared-everything design optimized for SMP systems. SQL server is tightly integrated with the NT operating systems threads

# DBMS Schemas for Decision Support

**Data Layout for best access**

**Multidimensional Data Model**

**Star Schema**

Two groups: facts and dimension

Facts are the core data element being analyzed

e.g.. items sold

dimensions are attributes about the facts

e.g. date of purchase

The star schema is designed to overcome this limitation in the two-dimensional relational model.

**DBA Viewpoint**

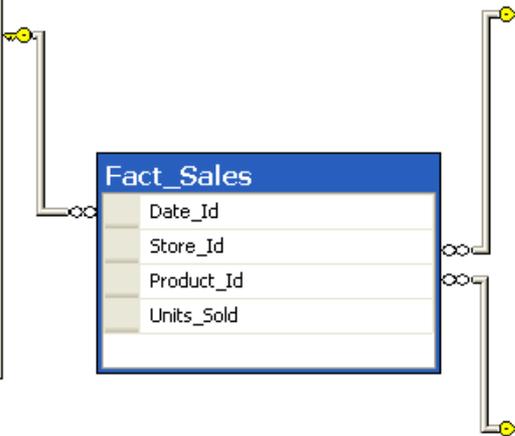
The fact table contains raw facts. The facts are typically additive and are accessed via dimensions.

Dim_Date	
🔑 Id	
Date	
Day	
Day_of_Week	
Month	
Month_Name	
Quarter	
Quarter_Name	
Year	

Fact_Sales	
Date_Id	
Store_Id	
Product_Id	
Units_Sold	

Dim_Store	
🔑 Id	
Store_Number	
State_Province	
Country	

Dim_Product	
🔑 Id	
EAN_Code	
Product_Name	
Brand	
Product_Category	



The dimension tables contain a non-compound primary key and are heavily indexed.

Dimension tables appear in constraints and GROUP BY Clauses, and are joined to the fact tables using foreign key references.

Once the star schema database is defined and loaded, the queries that answer simple and complex questions.

## **Potential Performance Problems with star schemas**

The star schema suffers the following performance problems.

### **Indexing**

Multipart key presents some problems in the star schema model.

(day->week-> month-> quarter-> year )

- It requires multiple metadata definition( one for each component) to design a single table.

- Since the fact table must carry all key components as part of its primary key, addition or deletion of levels in the hierarchy will require physical modification of the affected table, which is time-consuming processed that limits flexibility.
- Carrying all the segments of the compound dimensional key in the fact table increases the size of the index, thus impacting both performance and scalability.

## **Level Indicator**

The dimension table design includes a level of hierarchy indicator for every record.

Every query that is retrieving detail records from a table that stores details and aggregates must use this indicator as an additional constraint to obtain a correct result.

The user is not and aware of the level indicator, or its values are in correct, the otherwise valid query may result in a totally invalid answer.

Alternative to using the level indicator is the **snowflake** schema  
Aggregate fact tables are created separately from detail tables  
Snowflake schema contains separate fact tables for each level of aggregation

## **Other problems with the star schema design**

### *Pairwise Join Problem*

5 tables require joining first two tables, the result of this join with third table and so on.

The intermediate result of every join operation is used to join with the next table.

Selecting the best order of pairwise joins rarely can be solve in a reasonable amount of time.

Five-table query has  $5!=120$  combinations

This problem is so serious that some databases will not run a query that tries to join too many tables.

## **STARjoin and STARindex**

A STARjoin is a high-speed, single-pass, parallelizable multitable join and is introduced by Red Brick's RDBMS.

STARindexes to accelerate the join performance

STARindexes are created in one or more foreign key columns of a fact table.

Traditional multicolumn references a single table where as the STARindex can reference multiple tables

With multicolumn indexes, if a query's WHERE Clause does not contain on all the columns in the composite index, the index cannot be fully used unless the specified columns are a leading subset.

The STARjoin using STARindex could efficiently join the dimension tables to the fact table without penalty of generating the full Cartesian product.

The STARjoin algorithm is able to generate a Cartesian product in regions where there are rows of interest and bypass generating Cartesian products over region where there are no rows.

10 to 20 times faster than traditional pairwise join techniques

## **Bit mapped Indexing**

### **SYBASE IQ**

#### **Overview.**

Data is loaded into SYBASE IQ, it converts all data into a series of bitmaps; which are then highly compressed and stored in disk.

SYBASE IQ indexes do not point to data stored elsewhere all data is contained in the index structure.

## Data Cardinality.

Bitmap indexes are used to queries against low-cardinality data—that is data in which the total number of potential values is relatively low.

For example, state code data cardinality is 50 and gender cardinality is only 2(male and female).

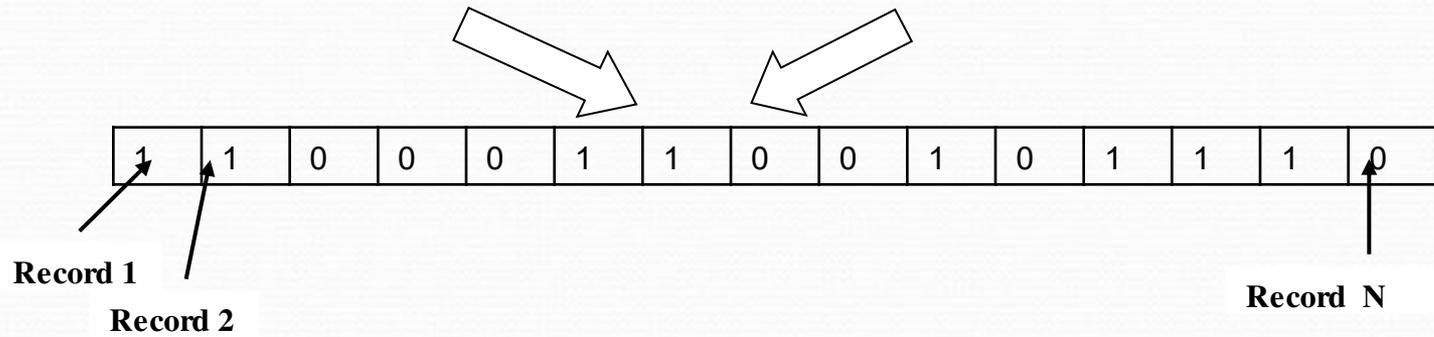
For low cardinality data, each distinct value has its own bitmap index consisting of a bit for every row in the table.

The bit map index representation is a 10000 bit long vector which has its bits turned ON (value of 1) for every record that satisfies “gender=”M” condition.

Bitmap indexes can become cumbersome and even unsuitable for high cardinality data where the range of potential value is high.

SYBASE IQ high cardinality index starts at 1000 distinct values.

Emp-Id	Gender	Last Name	First Name	Address
104345	M	Karthik	Ramasamy	10, North street
104567	M	Visu	Pandian	12, Pallavan street
104788	F	Mala	Prathap	123, Koil street



## Index Types.

The SYBASE IQ provides five index techniques. One is a default index called the *Fast projection index* and the other is either a low-or high-cardinality index.

## Performance.

SYBASE IQ technology achieves very good performance in ad hoc queries for several reasons.

- **Bitwise Technology.** This allows rapid response to queries containing various data type, supports data aggregation and grouping.
- **Compression.** SYBASE IQ uses sophisticated algorithm to compress data into bitmapping SYBASE IQ can hold more data in memory minimizing expensive I/O operations.

	E-Id	Gender	Name
Row		Read	→

E-Id	Gender	Name
1	1	0
1	1	1
0	1	0
1	0	1

**Traditional Row-based processing**

**SYBASE IQ Column-wise processing**

- **Optimized memory-based processing.** SYBASE IQ caches data columns in memory according to the nature of user's queries.
- **Columnwise processing.** SYBASE IQ scans columns not rows. For the low selectivity queries (those that select only a few attributes from a multi attribute row) the technique of scanning by columns drastically reduces the amount of data the engine has to search.
- **Low Overhead.** As an engine optimized for decision support, SYBASE IQ does not carry an overhead associated with traditional OLTP designed RDBMS performance.
- **Large Block I/O.** Block size high in SYBASE IQ can be tuned from 512 bytes to 64 bytes, so that the system can read as much information as necessary in a single I/O.
- **Operating-system-level parallelism.** SYBASE IQ breaks low-level operations like sorts, bitmap manipulation, load and I/O into non blocking operation's that the operating systems can schedule independently and in parallel.
- **Prejoin and ad hoc join Capabilities.** SYBASE IQ allows users to take advantage of know join relationships between tables by defining them in advance and building indexes between tables.

## Shortcoming of Indexing.

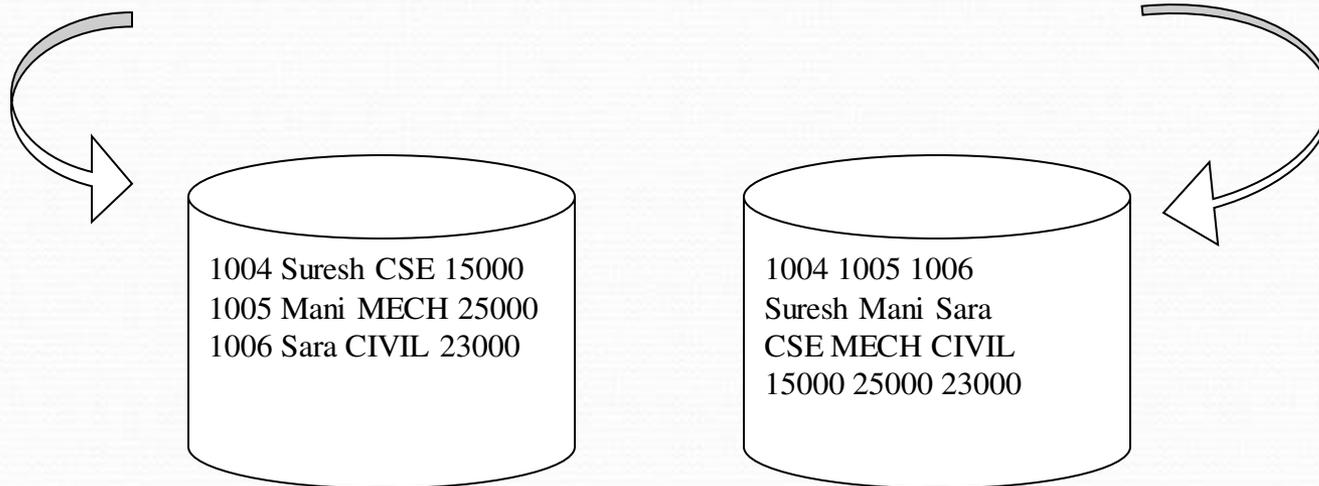
Some of the tradeoffs of the SYBASE IQ are as follows

- *No Updates.* SYBASE IQ does not support updates, and therefore is unsuitable
- *Lack of core RDBMS features.* SYBASE IQ does not support all the backup and recovery and also does not support stored procedures, data integrity checker, data replication, complex data types.
- *Less advantage for planned queries.* SYBASE IQ advantages are most obvious when running ad hoc queries
- *High memory Usage.* SYBASE IQ takes advantage of available system memory to avoid expensive I/O operations

## Column Local Storage

Performance in the data warehouse environment can be achieved by storing data in memory in column wise instead to store one row at a time and each row can be viewed and accessed as single record.

Emp-id	Emp-Name	Dept	Salary
1004	Suresh	CSE	15000
1005	Mani	MECH	25000
1006	Sara	CIVIL	23000



## **Complex Data types**

The warehouse environment support for datatypes of complex like text, image, full-motion video, some and large objects called binary large object (BLOBs) other than simple such as alphanumeric.

## Tools Requirements

The tools that enable sourcing of the proper data contents and formats from operational and external data stores into the data warehouse to perform a number of important tasks that include

- Data transformation from one format to another on the basis of possible differences between the source and the target platform.
- Data transformation and calculation based on the application of the business rules that force certain transformations. Examples are calculating age from the date of birth, or replacing a possible numeric gender code with a more meaningful “male” and “female” and data cleanup, repair, and enrichment, which may include corrections to the address field based on the value of the postal zip code.

- Data conversations and integration, which may include combining several source records into a single record to be loaded into the warehouse.
- Metadata synchronization and management, which includes storing and/or updating metadata definitions about source data files, transformation actions, loading formats, and events etc.,
- When implementing a data warehouse, several selection criteria that affect the tools ability to transform, consolidate, integrate, and repair the data should be considered.
- The ability to identify data in the source environments that can be read by the conversion tool is important.
  - Support for flat files, indexed files
  - The capability to merge data from multiple data stores
  - The ability to read information from data dictionaries or important information from repository products is desired

- The code generated by the tool should be completely maintainable from within the development environment.
- Selective data extraction of both data elements and records enables users to extract only the required data.
- A field-level data examination for the transformation of data into information is needed.
- The ability to perform data-type and character-set translation is a requirement when moving data between incompatible systems.
- The capability to create summarization, aggregation, and derivation records and fields is very important.
- The data warehouse database management system should be able to perform the load directly from the tool, using the native API available with the RDBMS.
- Vendor stability and support for the product are items that must be carefully evaluated.

## **Vendor Approaches**

The integrated solutions can fall into one of the categories described below

Code generators

Database data replication tools

Rule-driven dynamic transformation engines capture data from source systems at user-defined intervals, transform the data, and then send and load the results into a target environment, typically a data mart

## **Access to Legacy Data**

Many organizations develop middleware solutions that can manage the interaction between the new applications and growing data warehouses on one hand and back-end legacy systems in the other hand.

A three architecture that defines how applications are partitioned to meet both near-term integration and long-term migration objectives.

- The data layer provides data access and transaction services for management of corporate data assets.
- The process layer provides services to manage automation and support for current business process.
- The user layer manages user interaction with process and /or data layer services.

## **Vendor Solutions**

### **Prism Solutions**

Provides a comprehensive solution of data warehousing by mapping source data to a target database management system to be used as warehouse.

Warehouse Manager generates code to extract and integrate data, create and manage metadata, and build a subject-oriented, historical base.

Prism Warehouse Manager can extract data from multiple source environments including DB2, IDMS, IMS, VSAM, RMS and sequential files under UNIX or MVS. Target databases include ORACLE SYBASE, and INFIRMIX

## **SAS Institute**

SAS tools to serve all data warehousing functions.

Its data repository function can act to build the informational database.

SAS Data Access Engine serve as extraction tools to combine common variables, transform data representation forms for consistency, consolidate redundant data, and use business rules to produce computed values in the warehouse.

SAS engines can work with hierarchical and relational databases and sequential files

# **Carleton Corporation's PASSPORT and MetaCenter.**

## **PASSPORT.**

PASSPORT is sophisticated metadata-driven, data-mapping and data-migration facility.

PASSPORT Workbench runs as a client on various PC platforms in the three-tiered environment, including OS/2 and Windows.

The product consists of two components.

The first, which is mainframe-based, collects the file, record, or table layouts for the required inputs and outputs and converts them to the Passport Data Language (PDL).



Overall, PASSPORT offers

- A metadata dictionary at the core of the process.
- Robust data conversion, migration, analysis, and auditing facilities.
- The PASSPORT Workbench that enables project development on a workstations, with uploading of the generated application to the source data platform.
- Native interfaces to existing data files and RDBMS, helping users to leverage existing legacy applications and data.
- A comprehensive fourth-generation specification language and the full power of COBOL.

The MetaCenter.

The MetaCenter, developed by Carleton Corporation in partnership with Intellidex System, Inc., is an integrated tool suite that is designed to put users in control of the data warehouse.

It is used to manage

- Data extraction
- Data transformation
- Metadata capture
- Metadata browsing
- Data mart subscription
- Warehouse control center functionality
- Event control and notification

## **Vality Corporation**

Vality Corporation's Integrity data reengineering tool is used to investigate, standardize, transform, and integrate data from multiple operational systems and external sources.

- Data audits
- Data warehouse and decision support systems
- Customer information files and house holding applications
- Client/server business applications such as SAP, Oracle, and Hogan
- System consolidations
- Rewrites of existing operational systems

## **Transformation Engines**

### **Informatica**

Informatica's product, the PowerMart suite, captures technical and business metadata on the back-end that can be integrated with the metadata in front-end partner's products. PowerMart creates and maintains the metadata repository automatically.

It consists of the following components

*PowerMart Designer* is made up of three integrated modules- Source Analyzer, Warehouse Designer, and Transformation Designer

*PowerMart Server* runs on a UNIX or Windows NT platform.

The *Information Server Manager* is responsible for configuring, scheduling, and monitoring the Information Server.

The *Information Repository* is the metadata integration hub of the Informatica PowerMart Suite.

*Informatica PowerCapture* allows a data mart to be incrementally refreshed with changes occurring in the operational system, either as they occur or on a scheduled basis.

## **Constellar**

The Constellar Hub is designed to handle the movement and transformation of data for both data migration and data distribution in an operational system, and for capturing operational data for loading a data warehouse.



Constellar employs a hub and spoke architecture to manage the flow of data between source and target systems.

Hubs that perform data transformation based on rules defined and developed using Migration Manager

Each of the spokes represents a data path between a transformation hub and a data source or target.

A hub and its associated sources and targets can be installed on the same machine, or may run on separate networked computers.

# Metadata

The metadata contains

- The location and description of warehouse system and data components
- Names, definition, structure, and content of the warehouse and end-user views.
- Identification of authoritative data sources.
- Integration and transformation rules used to populate the data warehouse; these include the mapping method from operational databases into the warehouse, and algorithms used to convert, enhance, or transform data
- Integration and transforms rules used to deliver data to end-user analytical tools.
- Subscription information, which includes a history of warehouse updates, refreshments, snapshots, versions, ownership authorizations, and extract audit trail
- Security authorizations, access control lists, etc.

Metadata is used for building, maintaining, managing, and using the data warehouse.

## Metadata Interchange Initiative

A Metadata standard developed for metadata interchange format and its support mechanism.

The goal of the standard include

- Creating a vendor-independent, industry-defined and application programming interface (API) for metadata.
- Enabling users to control and manage the access and manipulation of metadata in their unique environments through the use of interchange-standard compliant tools
- Allowing users to build tool configurations that meet their needs and to incrementally adjust those configurations as necessary to add or subtract tools without impact on the interchange standards environment.
- Enabling individual tools to satisfy their specific metadata access requirements freely and easily within the context of an interchange model
- Defining a clean, simple interchange implementation infrastructure that will facilitate compliance and speed up adoption by minimizing the amount of modification required to existing tools to achieve and maintain interchange standards compliance.
- Creating a process and procedures not only for establishing and maintaining the interchange standards specification but also for extending and updating it over time as required by evolving industry and user needs.

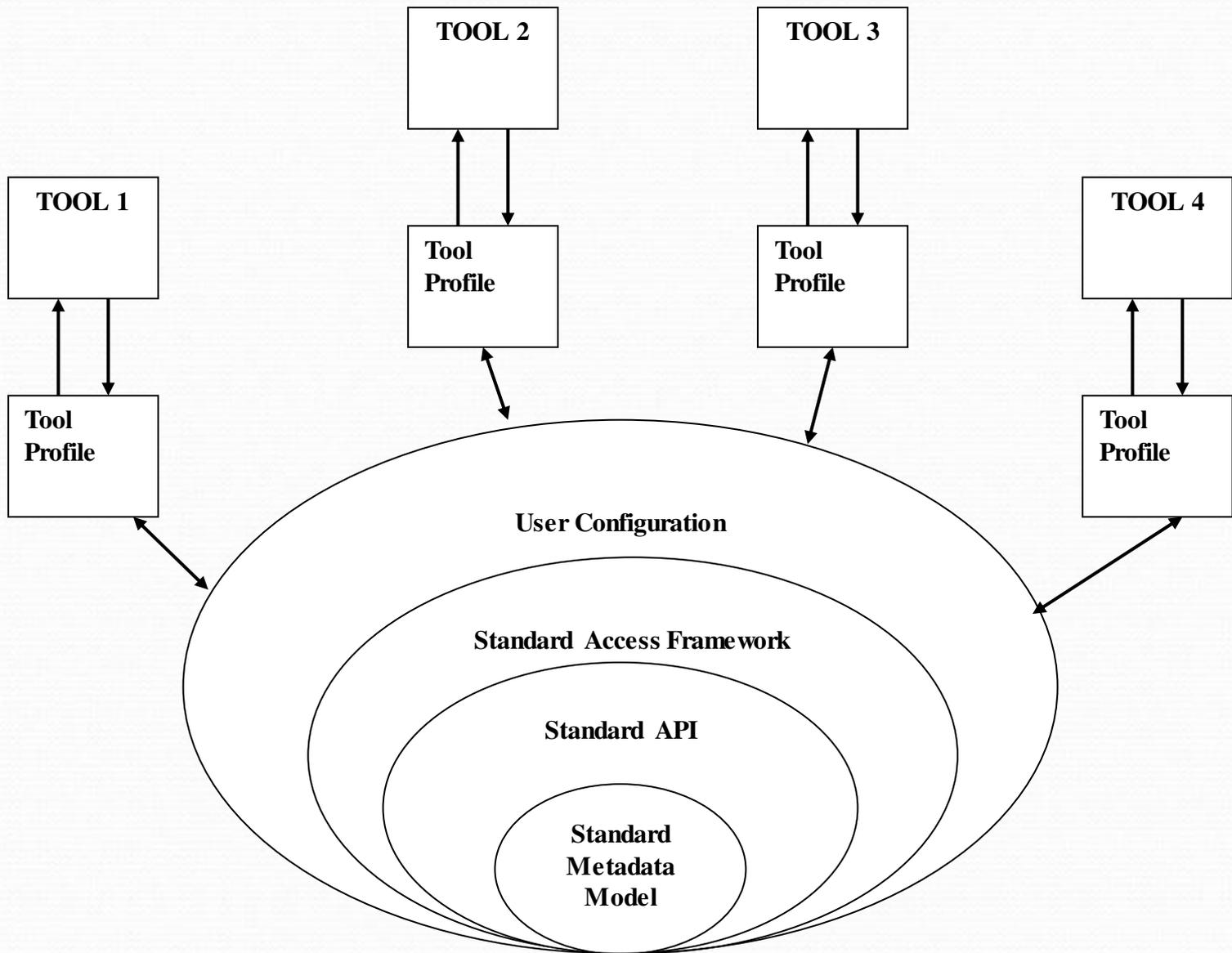
## Metadata Interchange Standard framework.

Implementation of the interchange standard metadata model must assume that the metadata itself may be stored in any type of storage facility or format; relational tables, ASCII files, fixed-format or customized-format repositories, and so on.

The components of the Metadata Interchange Standard Framework are

- The *Standard Metadata Model*, which refers to the ASCII file format used to represent the metadata that is being exchanged.
- The *Standard Access Framework*, which describes the minimum number of API functions a vendor must support.
- *Tool Profile*, which is provided by each tool vendor. The Tool Profile is a file that describes what aspects of the interchange standard metamodel a particular tool supports.
- The *User Configuration*, which is a file describing the legal interchange paths for metadata in the user's environment. This file allows customers to constrain the flow of metadata from tool to tool in their specific environments.

This framework defines the means by which various tool vendors will enable metadata interchange.



## **Metadata Repository**

The metadata itself is housed in and managed by the metadata repository.

Metadata repository management software can be used to map the source data to the target database, generate code for data transformations, integrate and transform the data, and control moving data to the warehouse.

Metadata defines the contents and location of data in the warehouse, relationships between the operational databases and the data warehouse, and the business views of the warehouse data that are accessible by the end-user tools.

A data warehouse design should ensure that there is a mechanism that populates and maintains the metadata repository, and that all access paths to the data warehouse have metadata as an entry point.

# Metadata Management

Metadata define all data elements and their attributes, data sources and timing, and the rules that govern data use and data transformations.

The metadata also has to be available to all warehouse users in order to guide them as they use the warehouse.

A well-thought-through strategy for collecting, maintaining, and distributing metadata is needed for a successful data warehouse implementation.

## Metadata Trends

The process of integrating external and internal data into the warehouse faces a number of challenges

- Inconsistent data formats
- Missing or invalid data
- Different level of aggregation
- Semantic inconsistency (e.g., different codes may mean different things from different suppliers of data)
- Unknown or questionable data quality and timeliness

# Reporting and Query tools and Applications

## Five categories of tools

- Reporting
- Managed Query
- Executive information systems
- On-line analytical processing
- Data mining

## Reporting tools

### *Production reporting tools*

Generate regular operational reports

Include third-generation languages such as COBOL, specialized fourth-generation languages such as Information builders.

### *Report writers*

For end users

E.g.. Segate Crystal report

Having graphical interfaces.

Pull groups of data from a variety of data sources and integrate them in a single report.

- **Managed query tools**

- Shield the end users from the complexities of SQL and database by inserting a metalayer between users and the databases.
- Supports point-and-click creation of SQL.
- Three tiered architecture to improve scalability.

- **Executive information system tools**

- EIS tools used to build customized, graphical decision support application .
- E.g. Pilot Software, Inc's Lightship, Platinum Technology's Forest and Trees.
- Building packaged applications that address functions, such as sales, budgeting, and marketing.

- **OLAP tools**

- An intuitive way to view corporate data.
- Aggregate data along common business subjects or dimension and allow to navigate through the hierarchies and dimensions with the click of a mouse button.
- Drill down, across, or up levels in each dimension or pivot and swap out dimensions to change their view of the data.

- E.g. Cognos' PowerPlay, Brio Technology, Inc's BrioQuery.
- **Data mining tools**
- Statistical and AI algorithms to analyze the correlation of variables in the data and interesting patterns and relationships to investigate.
- E.g. IBM's Intelligent Miner, DataMind Corp's DataMind.
- **The Need for Applications**
- The complexity of the questions grows, the tools may become inefficient.
- The various access types to the data stored in a data warehouse
  - Simple tabular form reporting
  - Ad hoc user-specified queries
  - Predefined repeatable queries
  - Complex queries with multitable joins, multilevel subqueries
  - Ranking
  - Multivariable analysis
  - Time series analysis
  - Data visualization, graphing, charting, and pivoting
  - Complex textual search
  - Statistical analysis

- AI techniques for testing of hypothesis, trends discovery
- Information mapping
- Interactive drill-down reporting and analysis
- Three distinct type of reporting
  1. Creation and viewing of standard reports – Routine delivery of report
  2. Definition and creation of ad hoc reports – managers and business users to quickly create their own reports and get quick answers
  3. Data exploration – Users can easily “surf” through data without a preset path.
- The above said reasons may require applications often take the form of custom-developed screens and reports that retrieve frequently used data and format it in a predefined standardized way.
- **Cognos Impromptu**
- **Overview**
- Product from Cognos Corporation.
- An enterprise solution for interactive database reporting.

- Object oriented architecture, control and administrative consistency across all users and reports
- Graphical user interface
- Ease of deployment
- Low cost
- Support both single user and multiusers
- **The Impromptu Information Catalog.**
- A LAN based repository of business knowledge and data access rules.
- Protects the database from repeated queries and unnecessary processing. Presents the database in a way that reflects how the business is organized,
- And uses the terminology of the business.
- Enables business-relevant reporting through business rules
- **Object-oriented architecture**
- Inheritance-based administration and distributed catalogs.
- Changes to business rules, permission sets, and query activities cascade automatically throughout the enterprise.

- Management functionality through the use of governors
- Governor can control
  - Query activity
  - Processing location
  - Database connections
  - Reporting permissions
  - User profiles
  - Client/server balancing
  - Database transactions
  - Security by value
  - Field and table security
- **Reporting**
  - Easy build and run their own reports
  - Contains predefined templates for mailing, labels, invoices, sales reports, and custom automation.

- Provides special reporting options
- ***Picklists and prompts***
- creating report for which users can select from lists of values called *picklist*.
- Reports containing too many values for a single variable, Impromptu offers prompt.
- It allows to supply value at run time
- ***Custom templates***
- Users can apply their data to the placeholders contained in the template
- Templates standard logic, calculations, and layout complete the report automatically in the user's choice of format
- ***Exception reporting***
- Ability to report high light values that lie outside accepted ranges.
- Three types of exception report
- ***Conditional filters***. Only those values that are outside defined threshold, or define ranges to organize data for quick evaluation. E.g. Sales under Rs.10000.

- **Conditional highlighting.** Formatting data on the basis of data values.

- E.g. Sales over Rs. 10000 always appear blue.

- **Conditional display.** Display report object under certain conditions.

- E.g. Sales graph only if the sales are below a predefined value.

- **Interactive reporting**

- Unifying query and reporting in a single interface.

- **Frames.**

- Frames are building blocks that may be used to produce reports.

- Frames formatted with fonts, border, colors, shading, etc.,

- Frames combined to create complex reports

- Templates can be created with multiframes.

- List frames

- Form frames

- Cross-tab frames

- Chart frames

- Text frames

- Picture frames

- OLE frames

- Impromptu's design is tightly integrated with the Microsoft Windows environment.

## ● **Impromptu Request Server.**

- sending query process to the server.
- Request server will execute the request, generating the result on the server.
- After the producing the result it notifies the client, so that client to do other things.
- supports data maintained in ORACLE 7.x and SYBASE

## ● **Supported database**

- Support ORACLE, SQL server, SYBASE SQL server, MDI DB2 Gateway, Informix, dBase, Paradox.

## ● **Applications**

- Organizations build applications for several reasons
  - A legacy DSS is still being used, and the reporting facilities appear adequate
  - An organization has made a large investment in a particular application
  - A new tool may require an additional investment, software, and the infrastructure
  - A particular reporting requirement may be too complicated for an available reporting tools to handle

- **PowerBuilder**

- Object-oriented applications, including encapsulation, polymorphism, inheritance and GUI objects.
- Once object created and tested and it can be reused by other applications
- Ability to interface with a wide variety of DBMS.

- ***Object orientation***

- Supports many object-oriented features
  - Inheritance
  - Data abstraction
  - Encapsulation
  - Polymorphism

- ***Windows facilities***

- PowerBuilder supports Windows facilities
  - DDE
  - OLE
  - MDI

## *Features*

- PowerBuilder windows and controls can contain program scripts that execute in response to different events that can be detected by PowerBuilder
- PowerBuilder controls are buttons, radio buttons, bush buttons, list box, check boxes, combo boxes, text fields menus, edit fields, and pictures
- Supports events such as *clicked*, *double clicked*
- Client/server application can be constructed using PowerBuilder painters
  - *Application Painter*.
    - First identifies basic details and components of new or existing applications
    - Application icon displays a hierarchical view of the application structure
    - All levels can be expanded or contracted with a click of the right mouse button.
    - Creating and naming new applications, selection of an application icon, setting of the library search path, and defining of default text characteristics.
    - Supports all events
    -

- It also used to run or debug the application
  - ***Window Painter***
    - Used to create and maintain PowerBuilder window objects.
    - Supports creation main application window, pop-up, dialog, and MDI.
    - Operations are performed by drag and drop and click operations.
    - PowerScript Painter – allows to select from a list of events and global and local variables.
    - Object browser – displays attributes of any object, data type and structures.
    - ***Data Windows Painter***
      - Dynamic objects that provide access to databases and other data sources such as ASCII files.
      - Applications use this to connect to multiple databases and files, as well as import and export data in a variety of formats such as dBase, Excel, Lotus.
      - It also supports stored procedure.
      - It allows developers to select a number of presentation styles from the list of tabular, grid, label, and free form.

- It also allows a user-specified number of rows to be displayed.
- **QueryPainter** – used to generate of SQL statements that can be stored in PowerBuilder libraries.
- Thus, using Application Painter, Window Painter, and DataWindows Painter facilitates, a simple client/server application can be constructed literally in minutes.
- A rich set of SQL functions is supported, including CONNECT/DISCONNECT, DECLARE, OPEN, and CLOSE cursor, FETCH, and COMMIT/ROLLBACK.
- PowerBuilder supplies server other painter.
  - **Database Painter** – used to pick table from the list box and examine and edit join conditions and predicates, key fields, extended attribute, display formats and other database attributes.
  - **Structure Painter** – Creation and modification of data structures and groups of related data elements
  - **Preference Painter** – Configuration tool that is used to examine and modify configuration parameters. For the PowerBuilder environment
  - **Menu Painter** – Creates menus

- **Function Painter** – Assists developers in creating functions calls and parameters using combo boxes.
- **Library Painter** - Manages the library in which the application components reside.
- **User object Painter** – Allows Developers to create custom controls.
- **Help Painter** – Built-in help system
- **Forté**
- It is three tiered architecture – Client, Application business logic, and Data server.
- Rapid development, testing, and deployment of distributed client/server applications across any enterprise.
- *Application partitioning.*
- Forté allows to build logical application that is independent of the underlying environment.
- Developers build an application as if it were to run entirely on a single machine.
- Forté automatically splits apart the application to run across the clients and servers that constitute the deployment environment.

- Support tunable application partitioning
- ***Shared-application services***
- With Forté , developers build collection of application components
- The components can include client functionality such as data presentation and other desktop processing.
- Shared-application services form the basis for a three-tiered application architecture
- Business events
- Automate the notification of significant business occurrences so that appropriate actions can be taken immediately by users.
- Forté detects the event, and sends notification to all the application components that have expressed interest in that event.
- It supports three functional components
  - ***Application Development Facility (ADF)***
  - Distributed object computing framework
  - To define user interfaces and application logic
  - Includes GUI designer and Transactional object-oriented language (TOOL)

- ***System Generation Facility (SGF)***

- Assists developers in partitioning the application, generating executables for distribution.

- ***Distributed Execution Facility (DEF)***

- Tools for managing applications at runtime, including system administration support, a distributed object manager to handle communications between applications partitions, and a performance monitor.

- **Web and Java integration**

- Integration with Java
- ActiveX and ActiveX server support
- Forté servers can be called from OLE
- Support for the ability to call Forté Application servers from C++ modules
- An option to generate and compile C++ code

- **Portability and supported platforms**

- Forté provides transparent portability across the most common client/server platforms for both development and deployment.

- Data General AViiON, Digital Alpha, Open VMS, UNIX, HP 9000, IBM RS/6000, Sun SPARC, and Window NT. Desktop GUI support includes Macintosh, Motif, and Windows.

- **Information Builder**

- The products from Information builder  
Cactus and FOCUS

- **Cactus**

- Client/server environment

- create, test and deploy business applications spanning the  
Internet

- Three-tiered environment and application of any size and scope.

- It builds highly reusable components

- Object-based visual programming environment

- Access to ActiveX , VX, and OLE controls.

- ***Web-enabled access***

- Application development for the Web with no prior  
knowledge of HTML.

- The developer can focus on the business problem rather  
than the underlying technology.

- ***Components and features***
  - Cactus Workbench – the front-end interface that provides access to the tool suite via iconic toolbars, push buttons, and menus.
  - Application Manager – in integrated application repository that manages the application development
  - Partition Manager
  - Object browser
  - Maintain – the proprietary language of cactus
  - File painter – used to build the database access objects
  - Application packager – used at deployment
  - EDA/Client – “message layer” for tier-to-tier communications.
  - Cactus Servers
  - Cactus OCX
- **Focus Fusion**
- For multidimensional database technology for OLAP and data warehousing.

- FOCUS Fusion provides
  - ***Fast query and reporting***
    - Its advanced indexing, parallel query and rollup facilities
  - ***Comprehensive, graphics-based administration facilities***
    - Database applications easy to build
  - ***Integrated copy management facilities***
    - Automatic data refresh from any source into Fusion
  - ***Open access via industry-standard protocols***
    - Through ANSI SQL, ODBC, and HTTP

# On-Line Analytical Processing (OLAP)

- **Need for OLAP**
- Market analysis and financial forecasting requires a multidimensional schema
- Required to process large numbers of records from very large data sets.
- The multidimensional nature is the key driver for OLAP
- Relational database and SQL have some limitations
  - E.g.. Full table scan, multiple join, aggregations and sorting and computing this require the resources may not available all the time
- RDBMS weakness in analyzing Time Series and complex mathematical functions
- RDBMS suffer response time and SQL functionality
- OLAP is a continuous, iterative, and interactive process.
  - E.g. Sales person performance affect monthly revenue numbers
- All these reasons make the need for OLAP

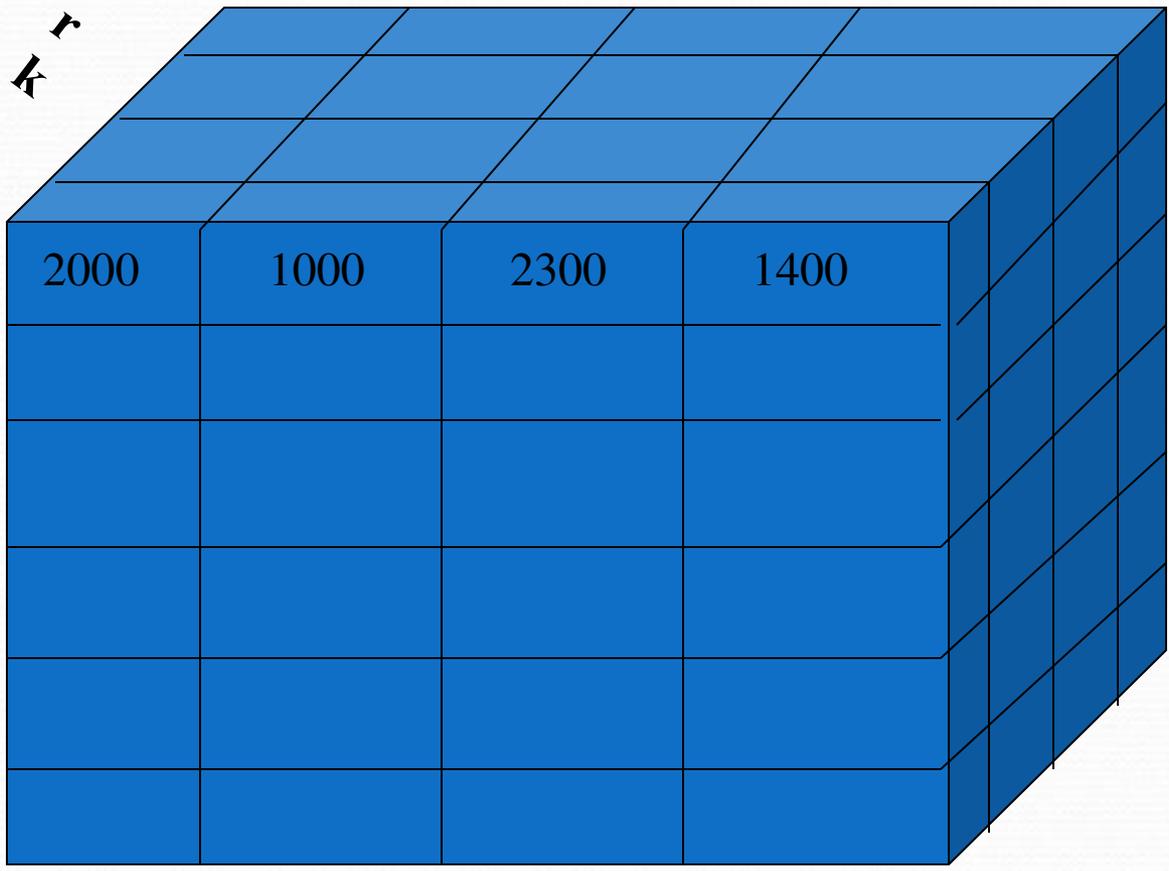
## • **Multidimensional Data Model**

- Business problems are multidimensional nature
- E.g.
  - How much revenue did the new product generate?
  - How much revenue did the new product generate by month, in north division, by sales office, relative to the previous version – a five dimensional query
- Hence Multidimensional data model viewed as cube
- The cube can be converted into table by dimensions with other values like sales numbers, unit price
- The response time of the multidimensional query depends on how many cells have to be added on the fly
- The number of dimensions increases, the number of cells in the table increases exponentially.
- The solution is to Build an efficient multidimensional database is to preaggregate all logical subtotals and totals along all dimensions

- Dimensions are hierarchical in nature
  - E.g. Time dimension – years, quarters, months, weeks, and days
  - Region – country, state, city etc.
  - drill-down – from annual sales to weekly sales and so on.
- properly handling *sparse* data
  - not every cell has a meaning across all dimensions
  - cells having duplicate data
- The multidimensional database to skip empty or repetitive cells can greatly reduce the size of cube and the amount of processing
- Dimensional hierarchy, sparse data management, and preaggregation are keys, they reduce the size of the database

S T E K R A M

Products



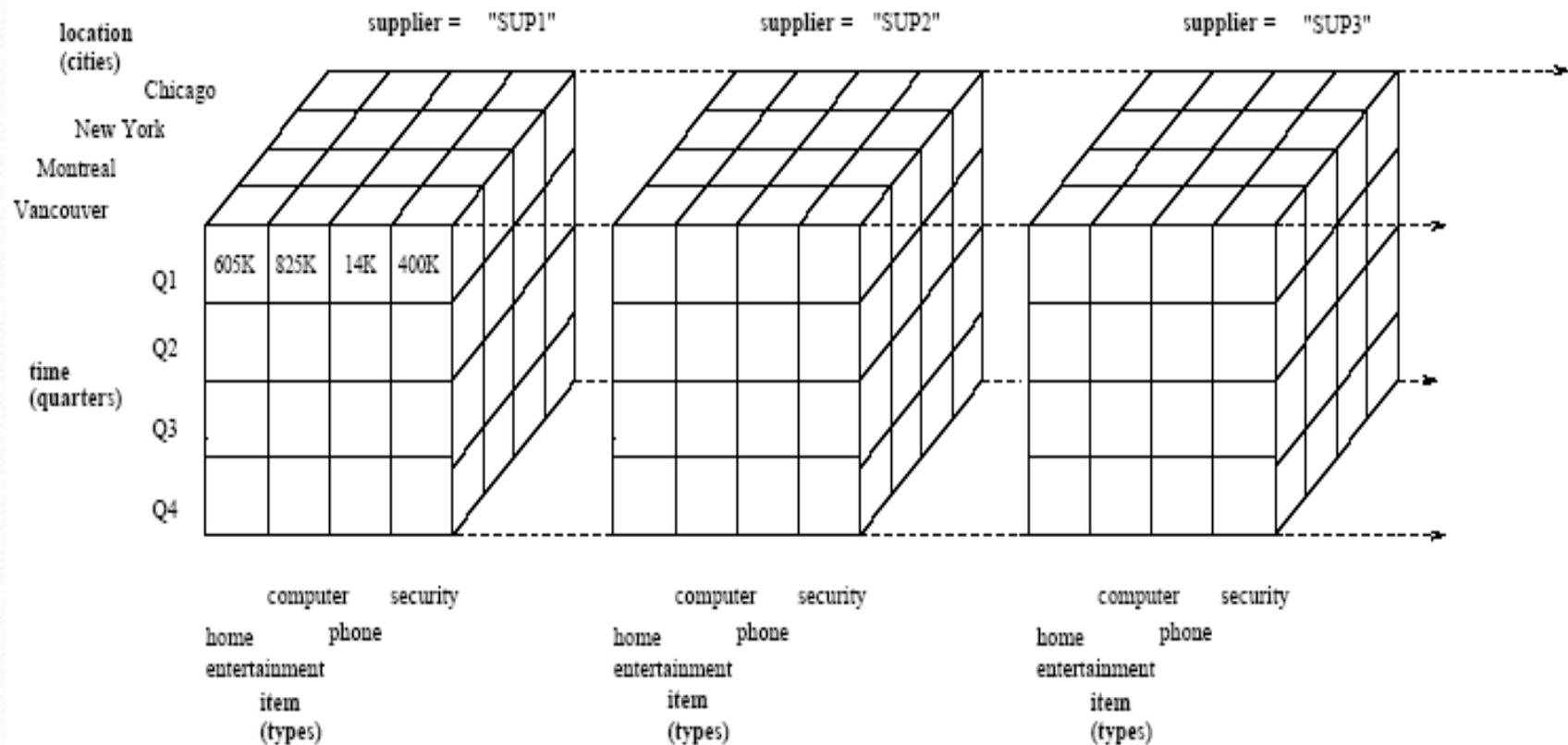
Q1

Q2

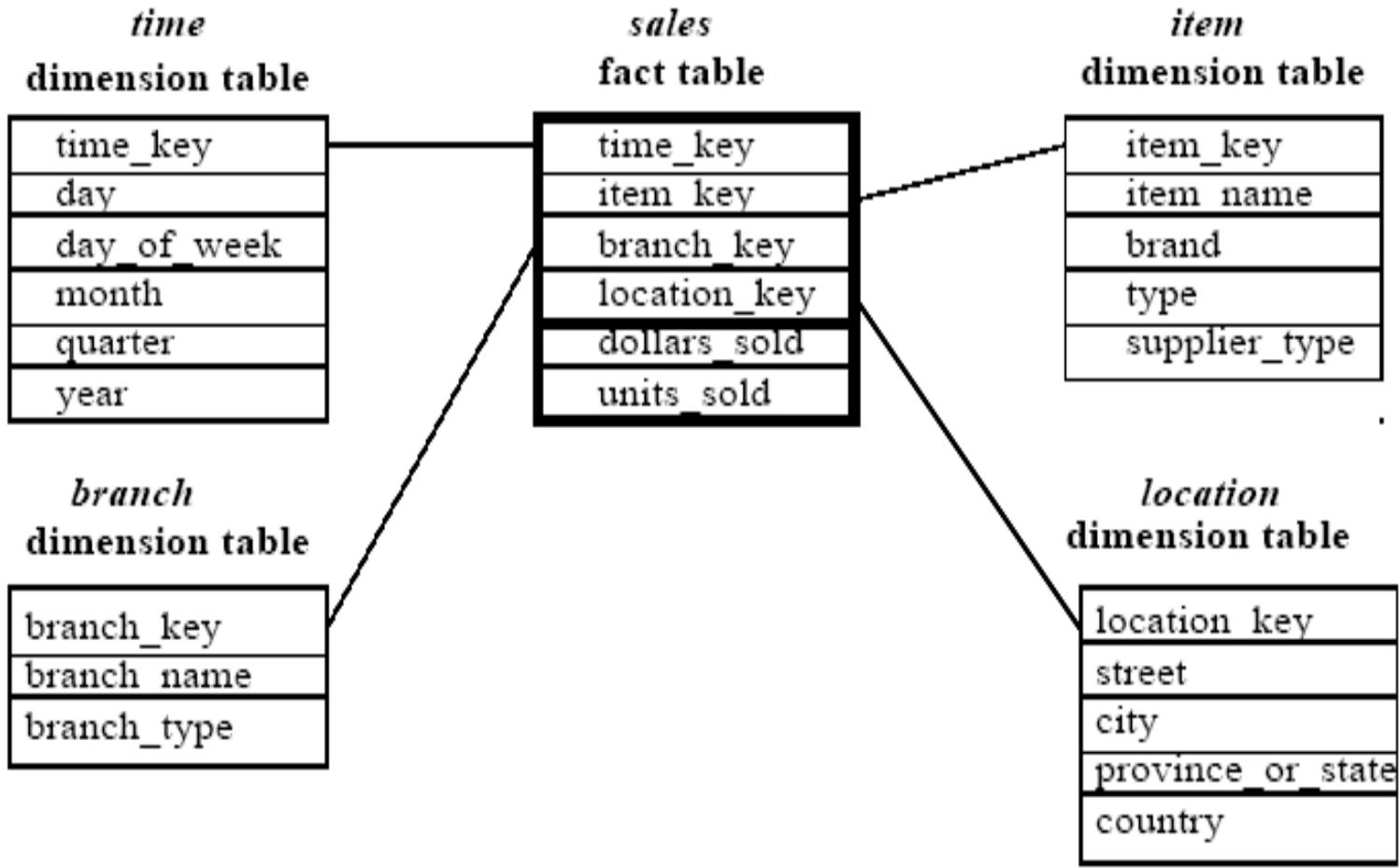
Q3

Q4

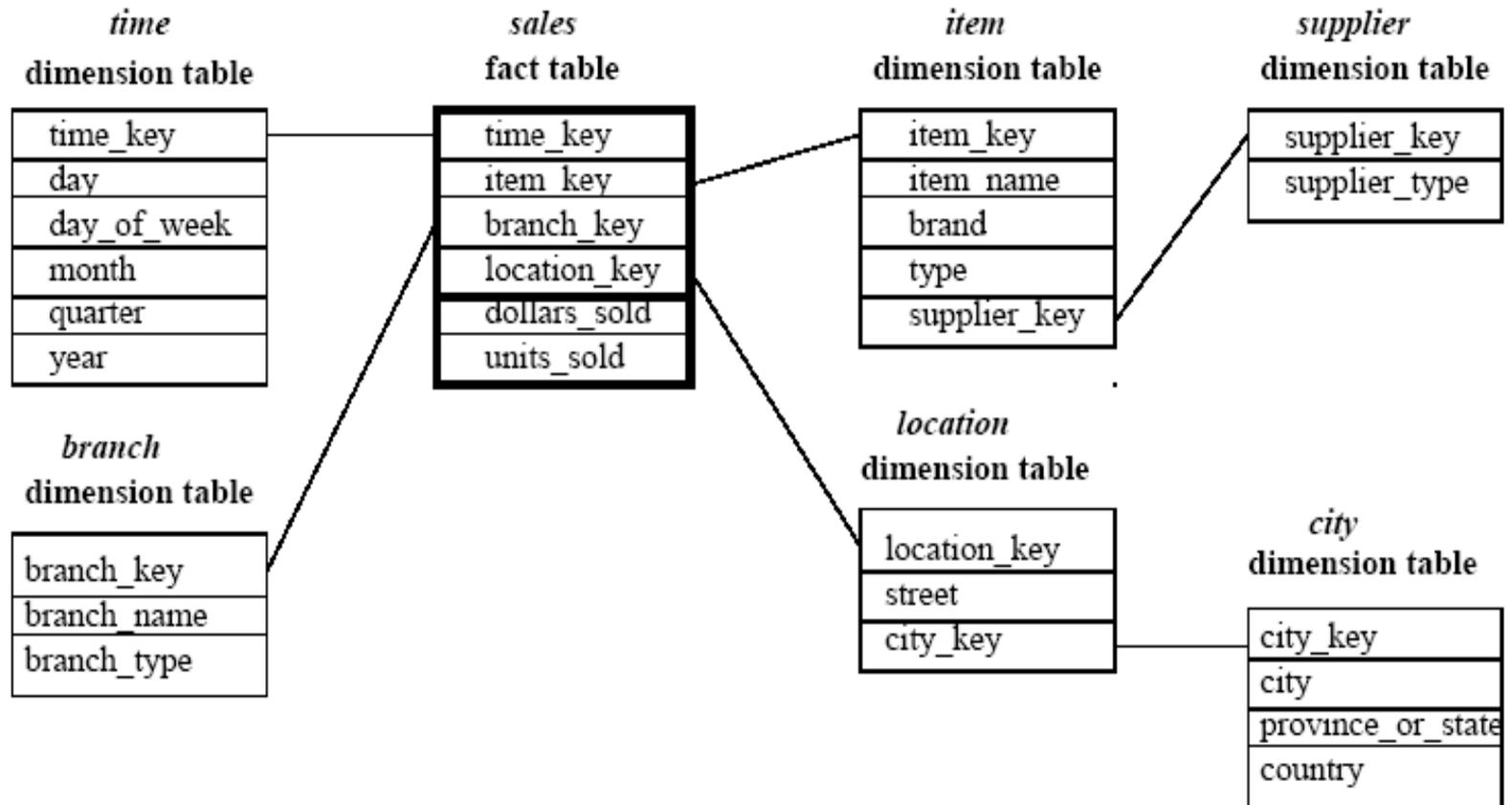
		location (cities)					
		Chicago	New York	Montreal	Vancouver		
time (quarters)	Q1	605K	825K	14K	400K	682	698
	Q2	680	952	31	512	728	789
	Q3	812	1023	30	501	784	870
	Q4	927	1038	38	580	925	984
		computer	home	phone	entertainment	security	item (types)



A 4-D data cube representation of sales data, according to the dimensions *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars\_sold*.



Star schema of a data warehouse for sales.



Snowflake schema of a data warehouse for sales.

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

	location = "Vancouver"				location = "Montreal"				location = "New York"				location = "Chicago"			
t i m e	item				item				item				item			
	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	605K	825K	14K	400K	818K	746K	43K	591K	1087K	968K	38K	872K	854K	882K	89K	623K
Q2	680K	952K	31K	512K	894K	769K	52K	682K	1130K	1024K	41K	925K	943K	890K	64K	698K
Q3	812K	1023K	30K	501K	940K	795K	58K	728K	1034K	1048K	45K	1002K	1032K	924K	59K	789K
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 sales data for *AllElectronics*, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars\_sold*.

1. **roll-up:** The roll-up operation (also called the “drill-up” operation by some vendors) performs aggregation on a data cube, either by *climbing-up a concept hierarchy* for a dimension or by *dimension reduction*. Figure 2.10 shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for *location* given in Figure 2.7. This hierarchy was defined as the total order  $street < city < province\_or\_state < country$ . The roll-up operation shown aggregates the data by ascending the *location* hierarchy from the level of *city* to the level of *country*. In other words, rather than grouping the data by city, the resulting cube groups the data by country.

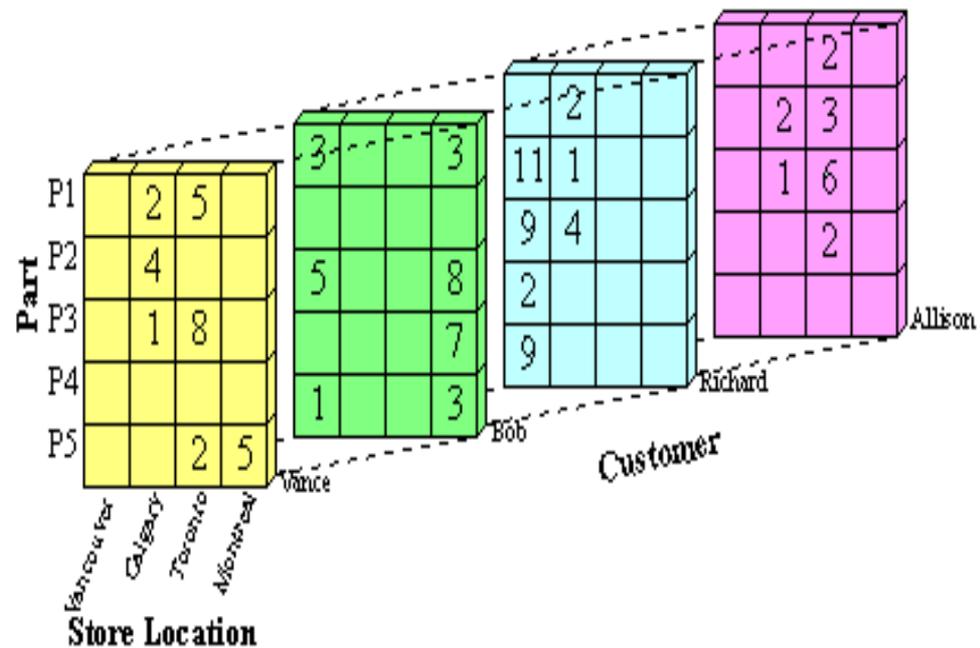
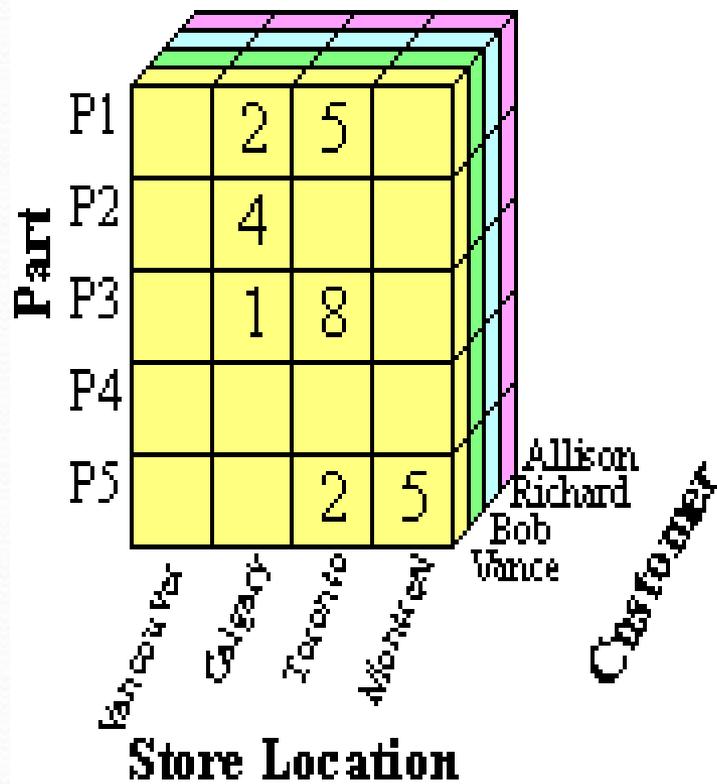
When roll-up is performed by dimension reduction, one or more dimensions are removed from the given cube. For example, consider a sales data cube containing only the two dimensions *location* and *time*. Roll-up may be performed by removing, say, the *time* dimension, resulting in an aggregation of the total sales by location, rather than by location and by time.

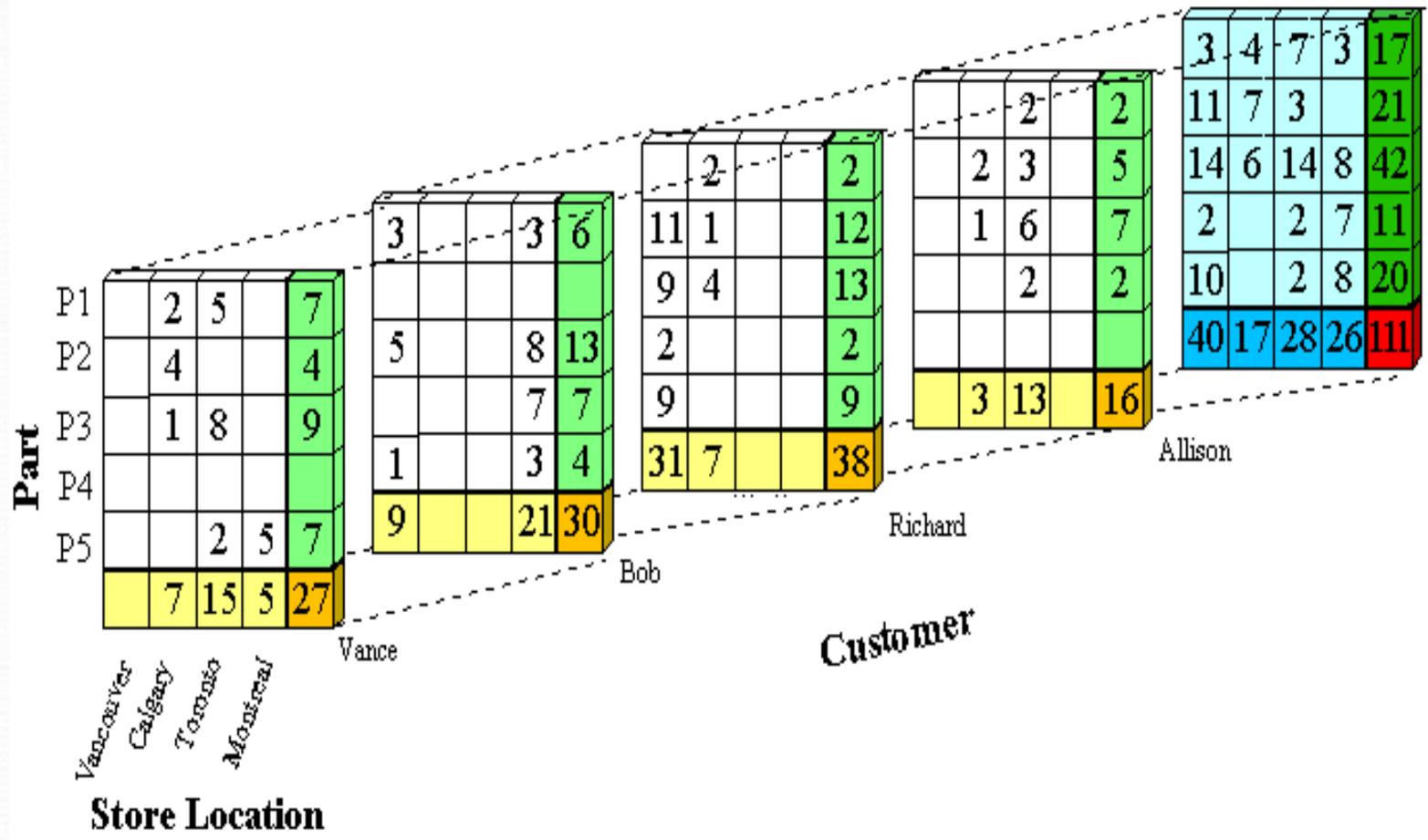
2. **drill-down:** Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either *stepping-down a concept hierarchy* for a dimension or *introducing additional dimensions*. Figure 2.10 shows the result of a drill-down operation performed on the central cube by stepping

down a concept hierarchy for *time* defined as  $day < month < quarter < year$ . Drill-down occurs by descending the *time* hierarchy from the level of *quarter* to the more detailed level of *month*. The resulting data cube details the total sales per month rather than summarized by quarter.

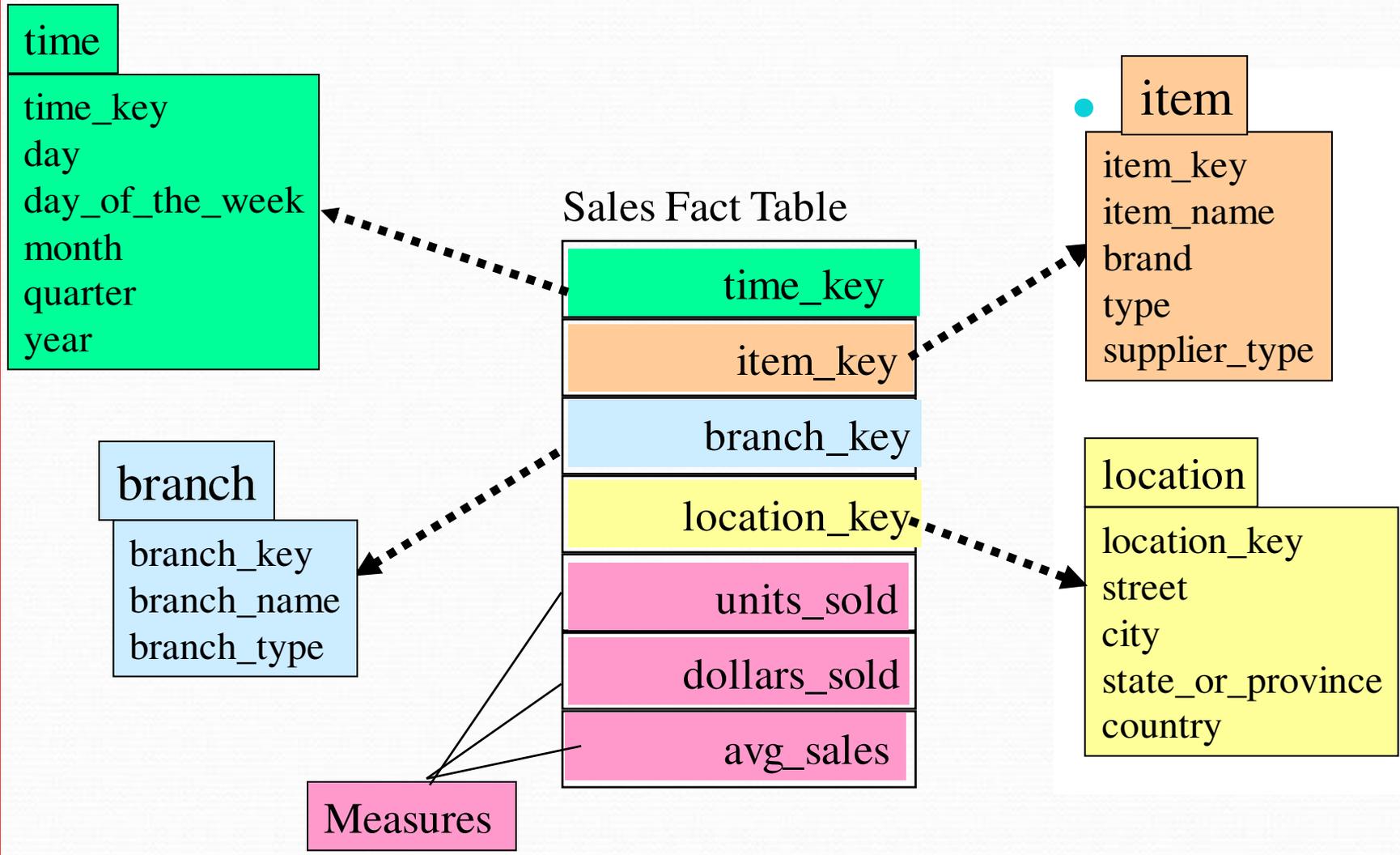
Since a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube. For example, a drill-down on the central cube of Figure 2.10 can occur by introducing an additional dimension, such as *customer\_type*.

- 3. slice and dice:** The *slice* operation performs a selection on one dimension of the given cube, resulting in a subcube. Figure 2.10 shows a slice operation where the sales data are selected from the central cube for the dimension *time* using the criteria  $time = "Q2"$ . The *dice* operation defines a subcube by performing a selection on two or more dimensions. Figure 2.10 shows a dice operation on the central cube based on the following selection criteria which involves three dimensions: ( $location = "Montreal" \text{ or } "Vancouver"$ ) and ( $time = "Q1" \text{ or } "Q2"$ ) and ( $item = "home \text{ entertainment}" \text{ or } "computer"$ ).
- 4. pivot (rotate):** *Pivot* (also called "*rotate*") is a visualization operation which rotates the data axes in view in order to provide an alternative presentation of the data. Figure 2.10 shows a pivot operation where the *item* and *location* axes in a 2-D slice are rotated. Other examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

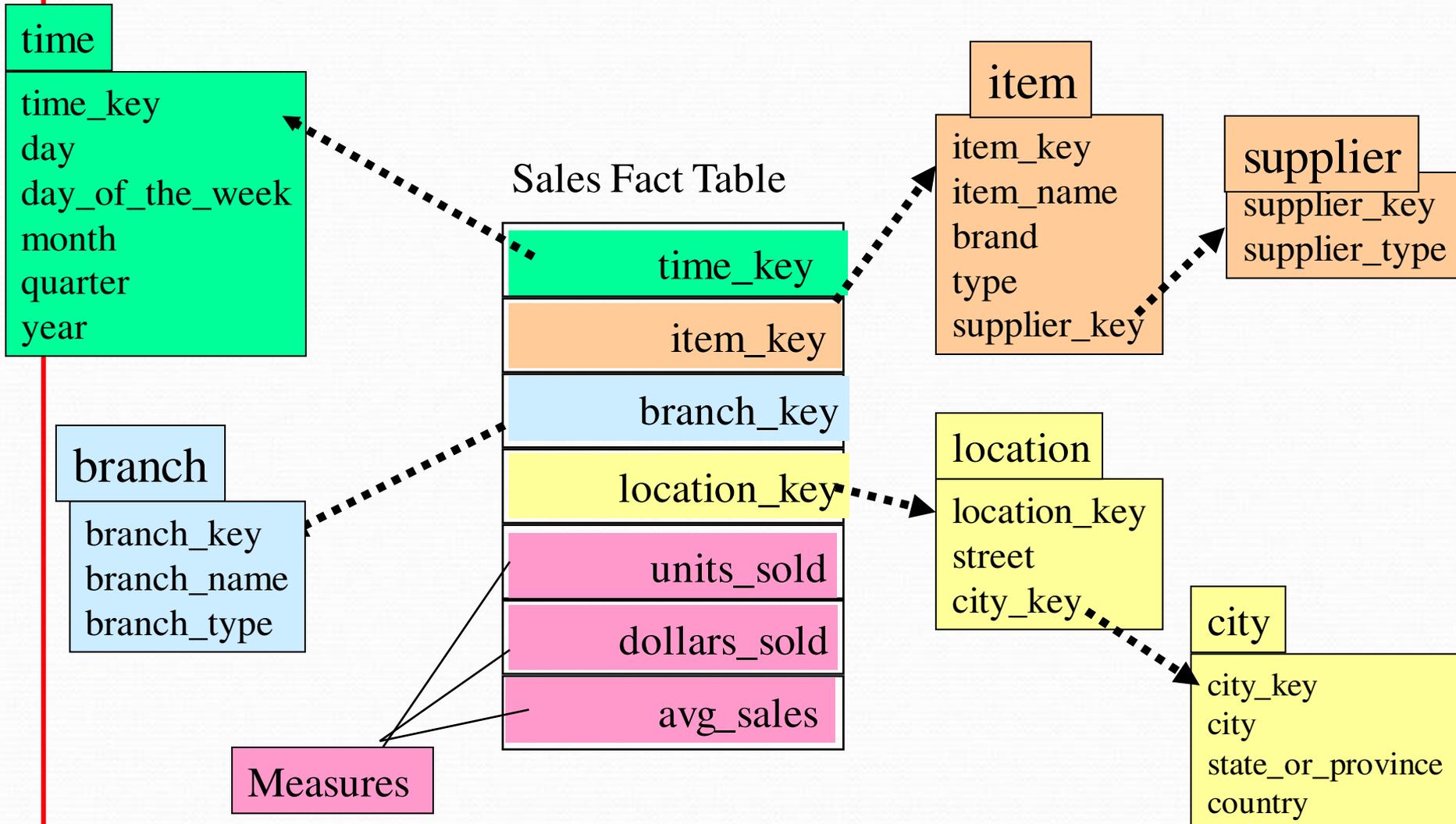




# Example of Star Schema



# Example of Snowflake Schema



# A Concept Hierarchy: Dimension (location)



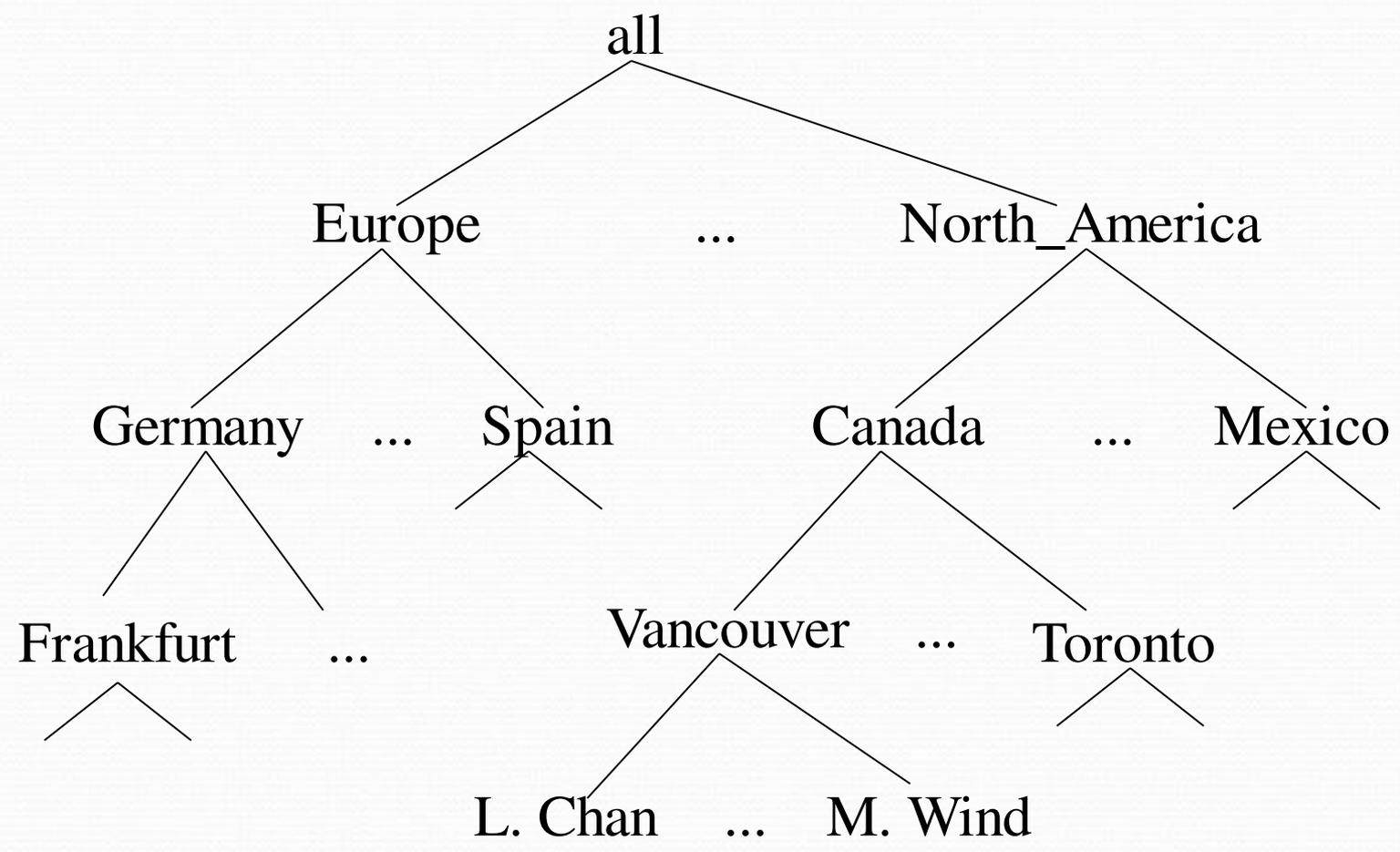
**all**

**region**

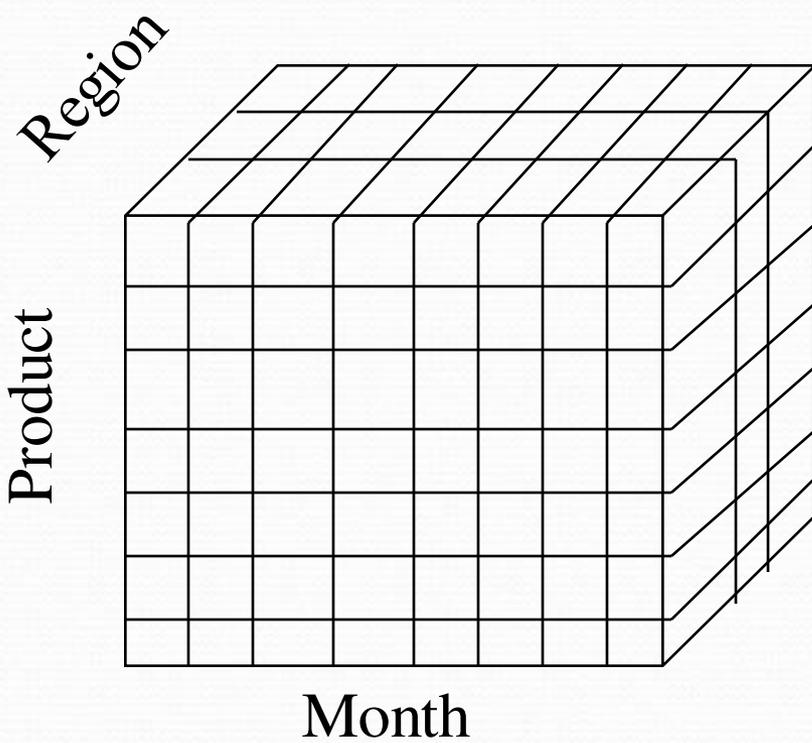
**country**

**city**

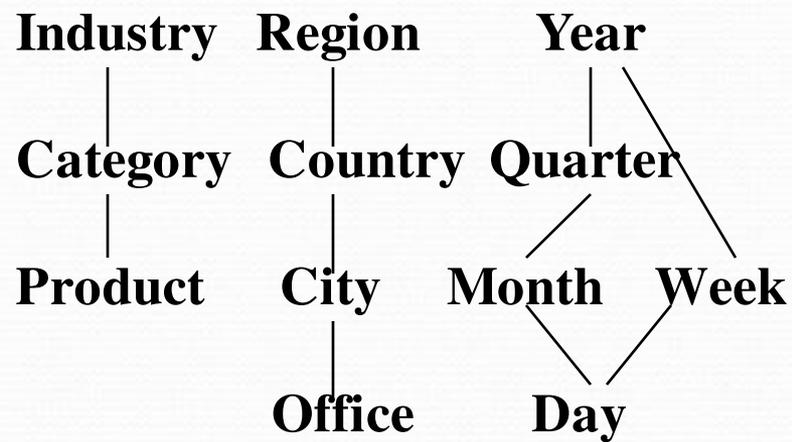
**office**



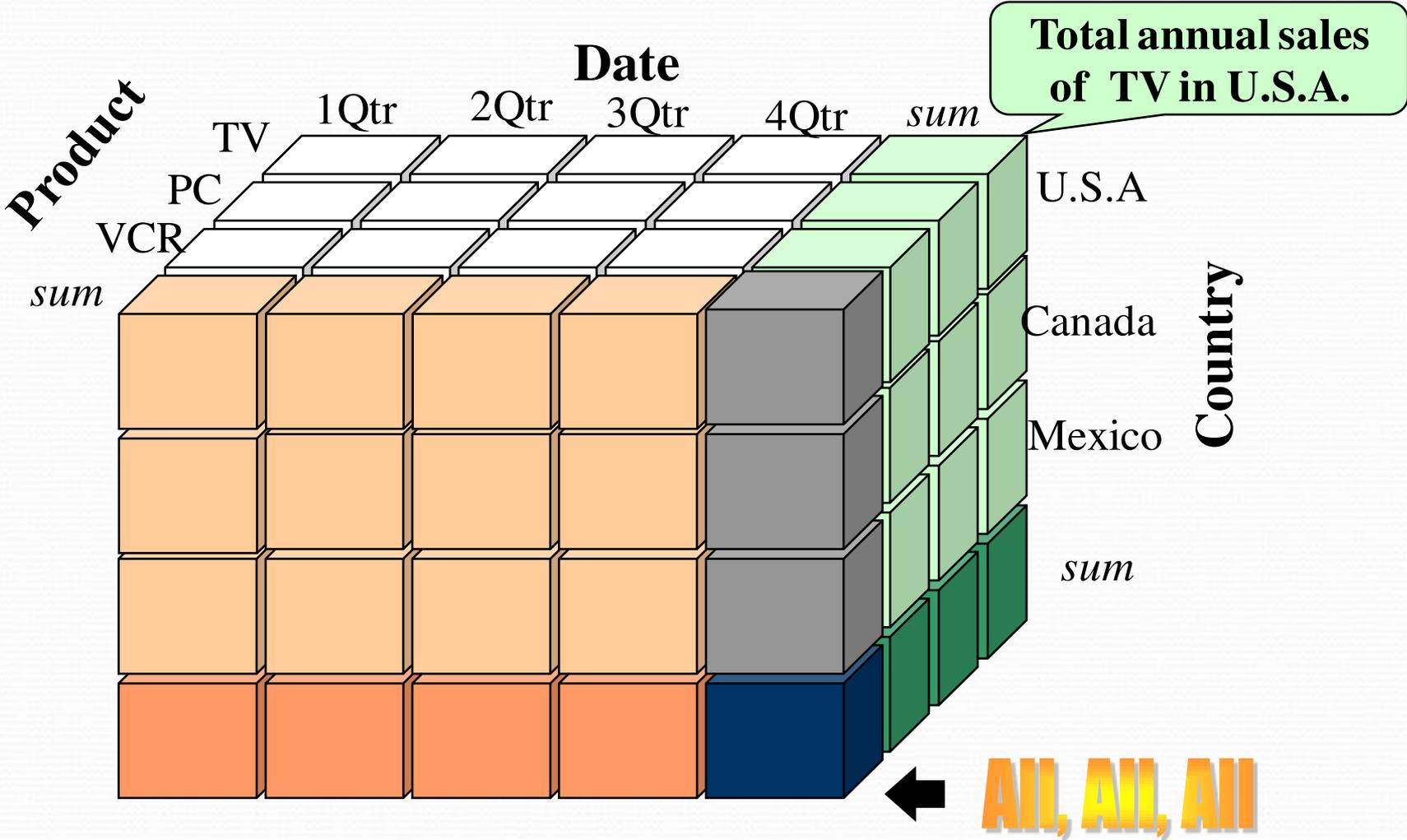
- Sales volume as a function of product, month, and region



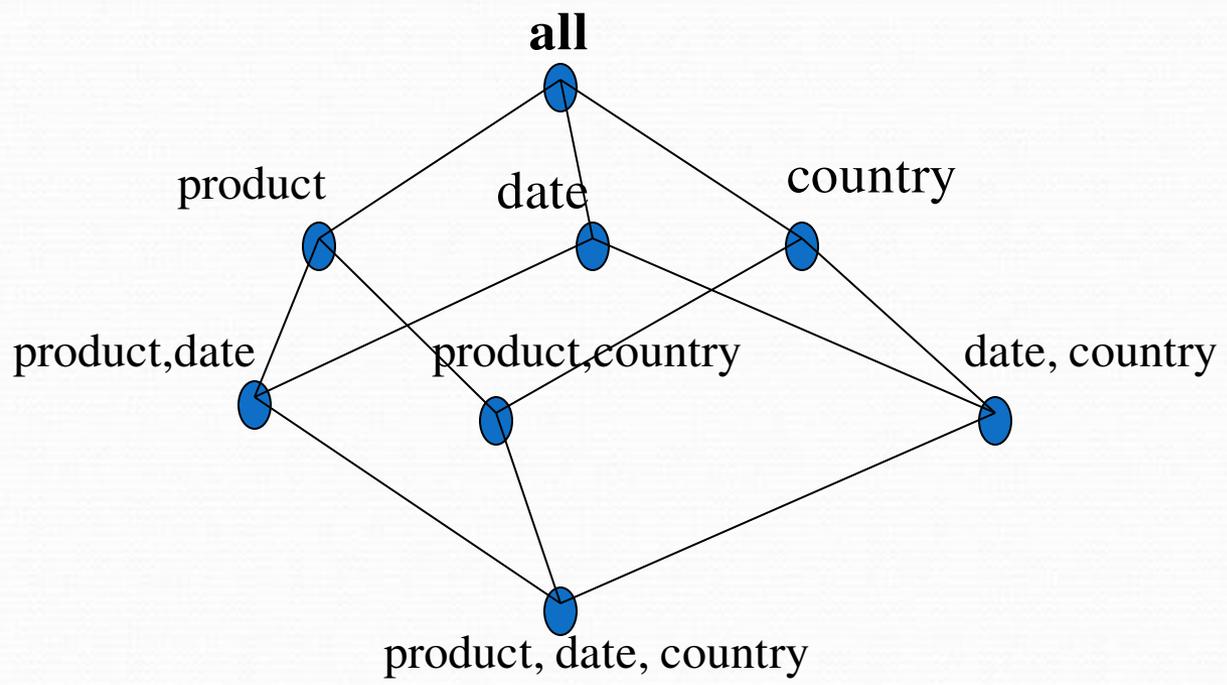
**Dimensions: Product, Location, Time**  
**Hierarchical summarization paths**



# A Sample Data Cube



# Cuboids Corresponding to the Cube



0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D(base) cuboid

## • **Categorization of OLAP tools**

### • **MLOP**

- Specialized data structures used for organize, navigate, and analyze data in an aggregated form
- Tight coupling with the application layer and presentation layer.
- Recently MLOP vendors provide APIs for OLAP operations.
- Data structures use array technology and, improved storage techniques to minimize the disk space requirements through sparse data management.
- Excellent performance when the data is utilized as designed.
- Some products treat time as a special dimension for time series analysis and other products provide strong analytical capabilities
- Applications requiring iterative and comprehensive time series analysis.
- Several challenges face users considering the implementation of applications with MLOP products.

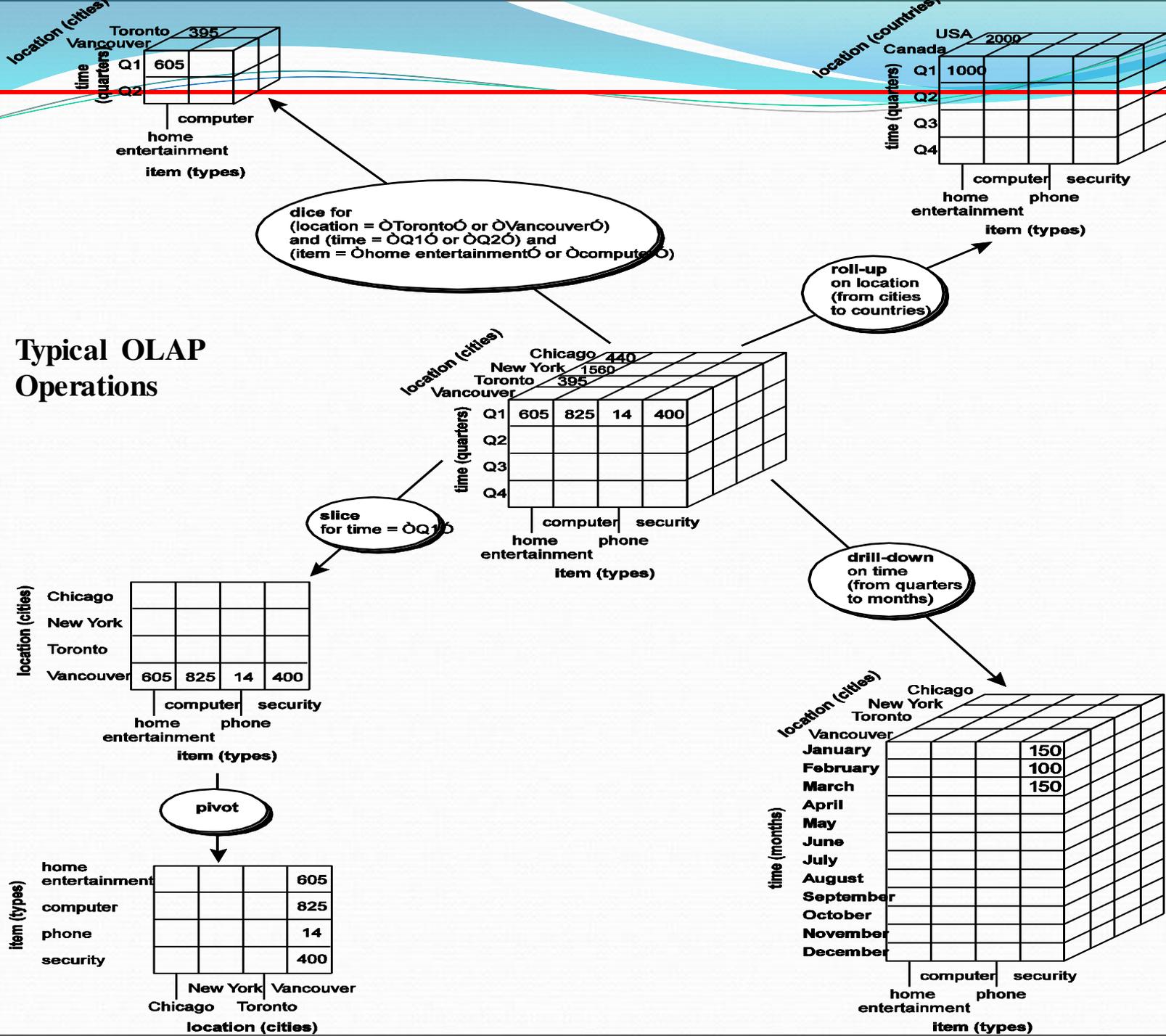
- Limitation in the ability of data structures to support multiple subject areas of data and the detail data required by many analysis applications
- Limitation in the way data can be navigated and analyzed, because the data is structured around the navigation and analysis requirements known at the data structures built.
- MLOP products require a different set of skills and tools for the database administrator of support

- With specialized multidimensional data storage and RDBMS technology, providing user with a facility that tightly “couples” the data multidimensional data structures (MDDs) with data maintained in and RDBMS.
- The MDDs to dynamically obtain detail data maintained in an RDBMS.
- For example sales to be stored and maintained in a persistent structure, will reduce the overhead of performing calculations and building aggregation during application initialization.

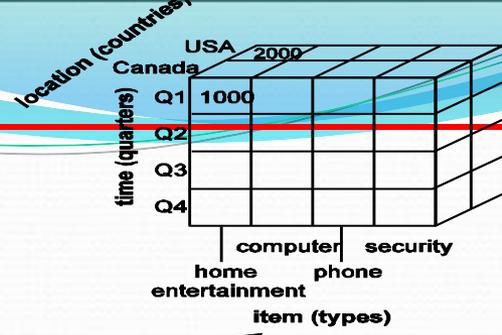
## Typical OLAP Operations

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

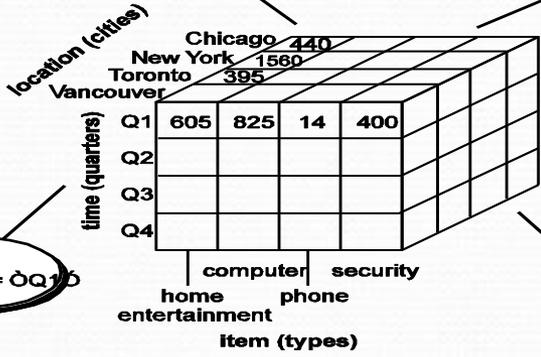
# Typical OLAP Operations



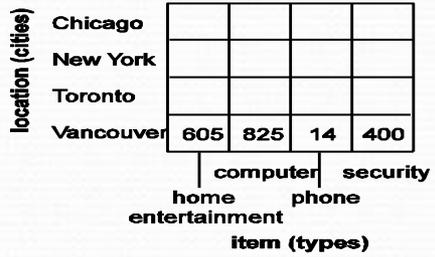
**dice** for  
(location = 'Toronto' or 'Vancouver')  
and (time = 'Q1' or 'Q2') and  
(item = 'home entertainment' or 'computer')



**roll-up**  
on location  
(from cities to countries)



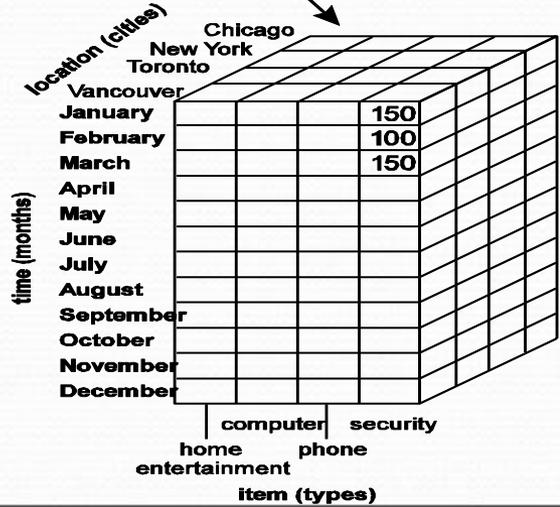
**slice**  
for time = 'Q1'

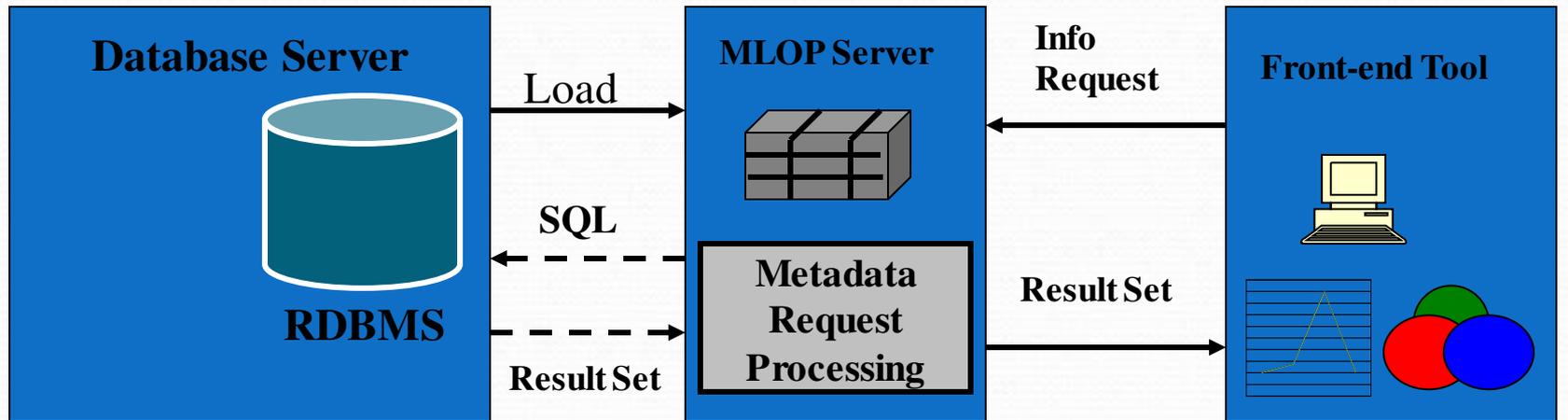


**pivot**



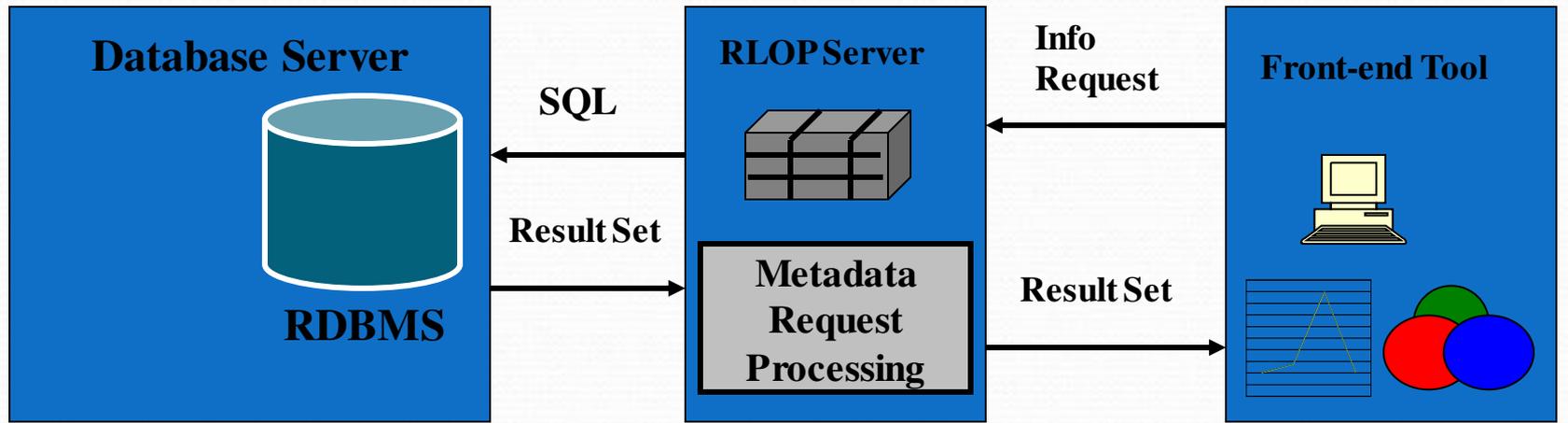
**drill-down**  
on time  
(from quarters to months)





- **RLOP**

- Support RDBMS products directly through a dictionary layer of metadata, bypassing any requirement for creating a static multidimensional data structure.
- Multidimensional views of the two-dimensional relational table to be created without the need to structure the data around the desired view.
- Creation of multiple SQL statements to handle user request
- It is undergoing some technological development
- Movement toward pure middleware technology that provides facilities to simplify development of multidimensional applications
- Further blurring of the lines that delineate RLOAP and hybrid-OLAP products.
-

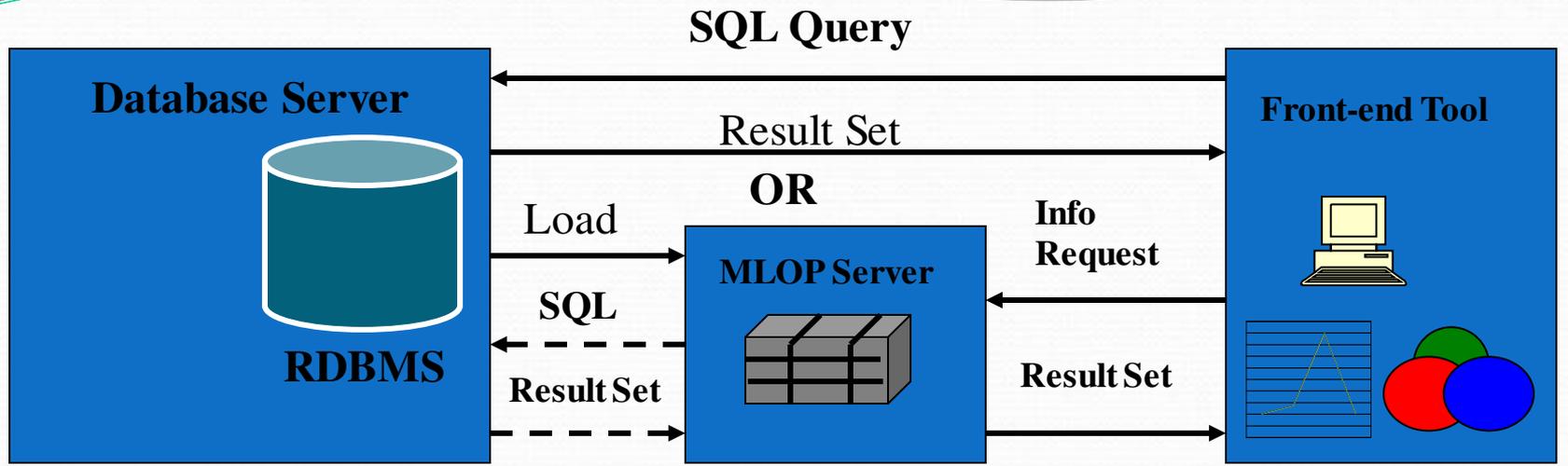


## • **Managed query environment (MQE)**

- Ability to perform limited analysis capability, either directly against RDBMS products, or by leveraging an intermediate MLOP server.
- Some products developed features to provide “datacube” and “slice and dice” analysis capabilities.
- Query executed and the selected data from the DBMS, which then delivers the requested data to the desktop, where it is placed into a datacube.
- The datacube can be stored and maintained locally in the desktop.
- Once the data is in the datacube, users can perform multidimensional analysis.
- The tools can work with MLOP servers, and the data from the relational DBMS can be delivered to the MLOP server, and from there to the desktop.
- With metadata definitions that assist users in retrieving the correct set of data that makes up the datacube.
- Each user to build a custom datacube, the lack of data consistency among users, and the relatively small amount of data that can be efficiently maintained are significant.

### • Examples

- Cognos Software’s PowerPlay, Andyne Software’s Pablo, Dimensional Insight’s CrossTarget, and Speedware’s Media.



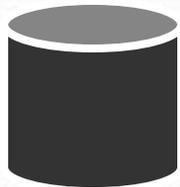
## ● **State of the Market**

- OLAP tools provide way to view the corporate data
- The tools aggregate data along common business subjects or dimensions and then let the users navigate through the hierarchies and dimensions.
- Some tools preaggregate data in special multidimensional database.
- Some other tools work directly against relational data and aggregate data on the fly.
- Some tools process OLAP data on the desktop instead of server.
- Leading database vendors incorporate OLAP functionality in their database kernels.
- Cognos PowerPlay
- IBI FOCUS Fusion
- Pilot Software

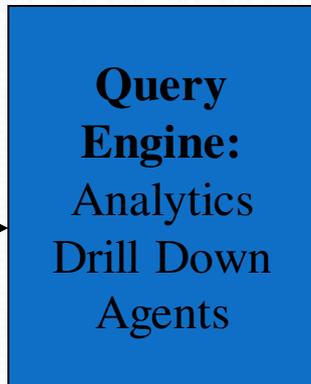
## • **OLAP Tools and the Internet**

- The Internet/WWW and data warehouse are tightly bound together
  - The Internet is a virtually free resource which provides a universal connectivity within and between companies
  - The Web eases complex administrative tasks of managing distributed environments
  - The Web allows companies to store and manage both data and applications on server that can be centrally managed, maintained and updated
- First-generation Web sites – The client can access the decision support report through static HTML pages via web browsers.
- Second-generation Web sites – Interactive and CGI (HTML gateway)
- Third-generation Web sites – Java Applets, and Web based application servers
- Vendors approaches for deploying tools on the Web include
  - *HTML publishing*
  - *Helper applications*
  - *Plug-ins*
  - *Server-centric components*
  - *Java and ActiveX applications*

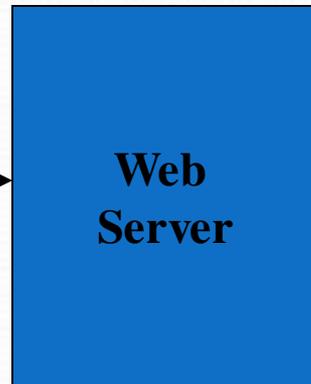
**Structured  
Content**



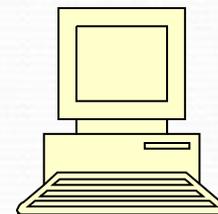
**SQL**



**CGI**

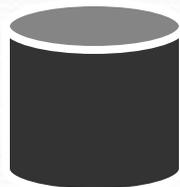
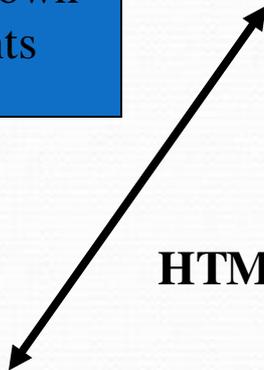


**HTML**



**Web Browser**

**HTML**



**Unstructured  
Content**

- Tools from Internet/Web implementations

- **Arbor Essbase Web**

- It includes OLAP manipulations
- Drill up, down, and across
- pivot, slice and dice
- Fixed and dynamic reporting also data entry
- It doesn't have client package.

- **Information Advantage WebOLAP**

- Server-centric
- Powerful analytical engine that generates SQL to pull data from relational database
- Provide client based package
- Data store and the analytical engine are separate
-

- MicroStrategy DSS Web
  - DSS server
  - relational OLAP server
  - DSS Architect data modeling tool
  - Dss Executive design tool for building executive information system
- Brio technology
  - Support OLAP applications on the Web
  - Its own server `brio.query.server`

U

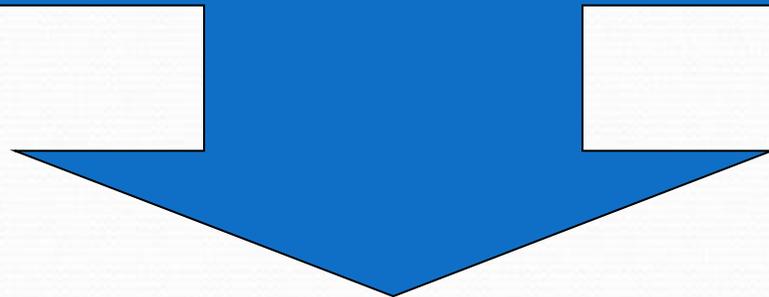
N

I

T

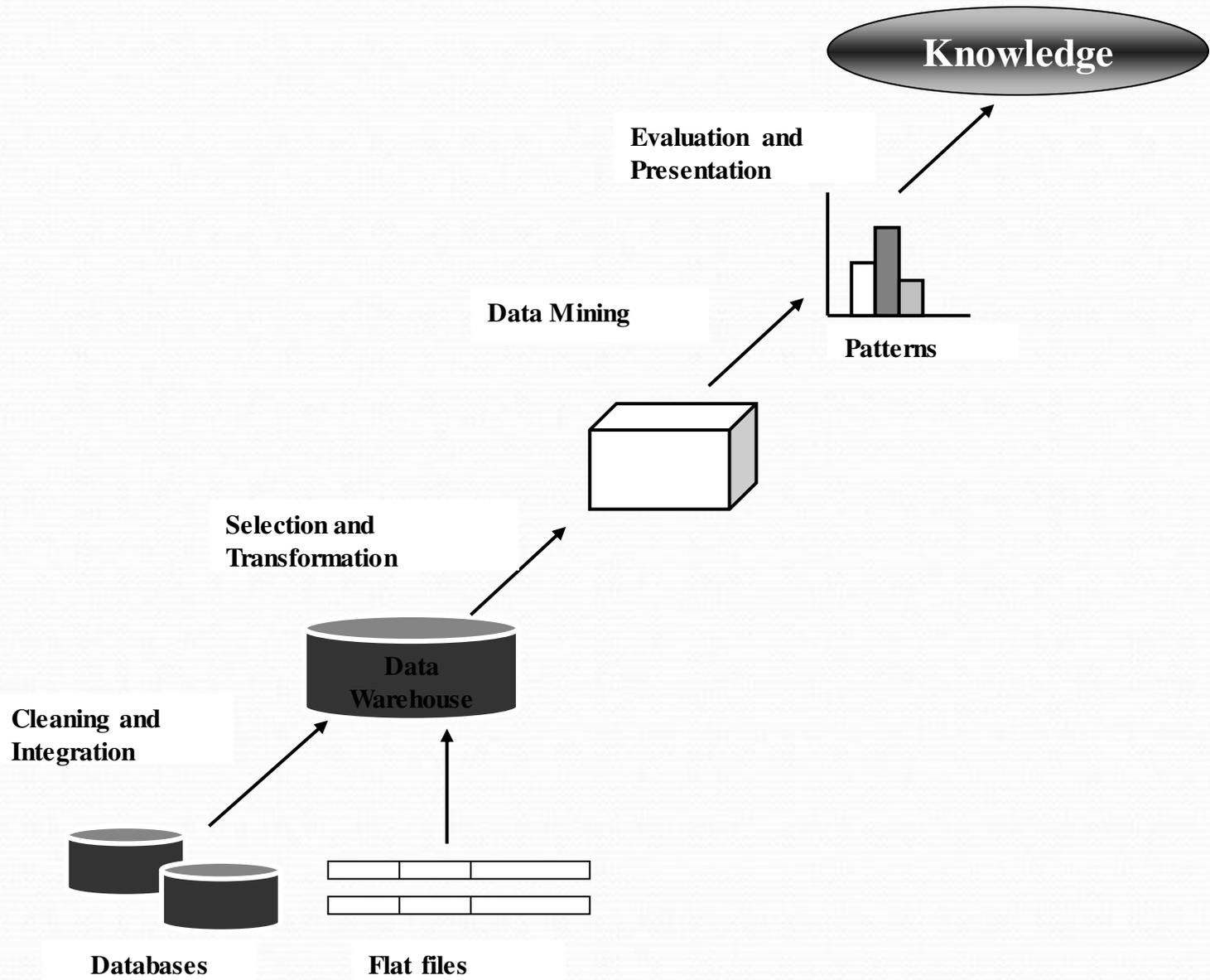
III

# Data Mining



# Introduction

- Extracting or “mining” knowledge from large amounts of data
- “Knowledge mining from data”.
- Knowledge mining, knowledge extraction, data/pattern analysis, data archaeology
- Data mining is a step in the process of knowledge discovery.
- Knowledge discovery is a process consists of iterative sequence of steps.
  1. Data cleaning - to remove noise and inconsistent data
  2. Data integration – where multiple data sources may be combined
  3. Data selection – where relevant to the analysis task are retrieved form the database
  4. Data transformation – where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
  5. Data mining – methods are applied to extract data patterns
  6. Patter evaluation – to identify patterns representing knowledge based on some interestingness measure
  7. Knowledge presentation – techniques are used to present the mined knowledge to the user
-



## • **Types of Data**

- Data mining can be performed on any kind of data repository including data streams. It includes the following data sources
- Database-oriented data sets and applications
  - Relational database, data warehouse, transactional database
- Advanced data sets and advanced applications
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data (incl. bio-sequences)
  - Structure data, graphs, social networks and multi-linked data
  - Object-relational databases
  - Heterogeneous databases and legacy databases
  - Spatial data and spatiotemporal data
  - Multimedia database
  - Text databases
  - The World-Wide Web

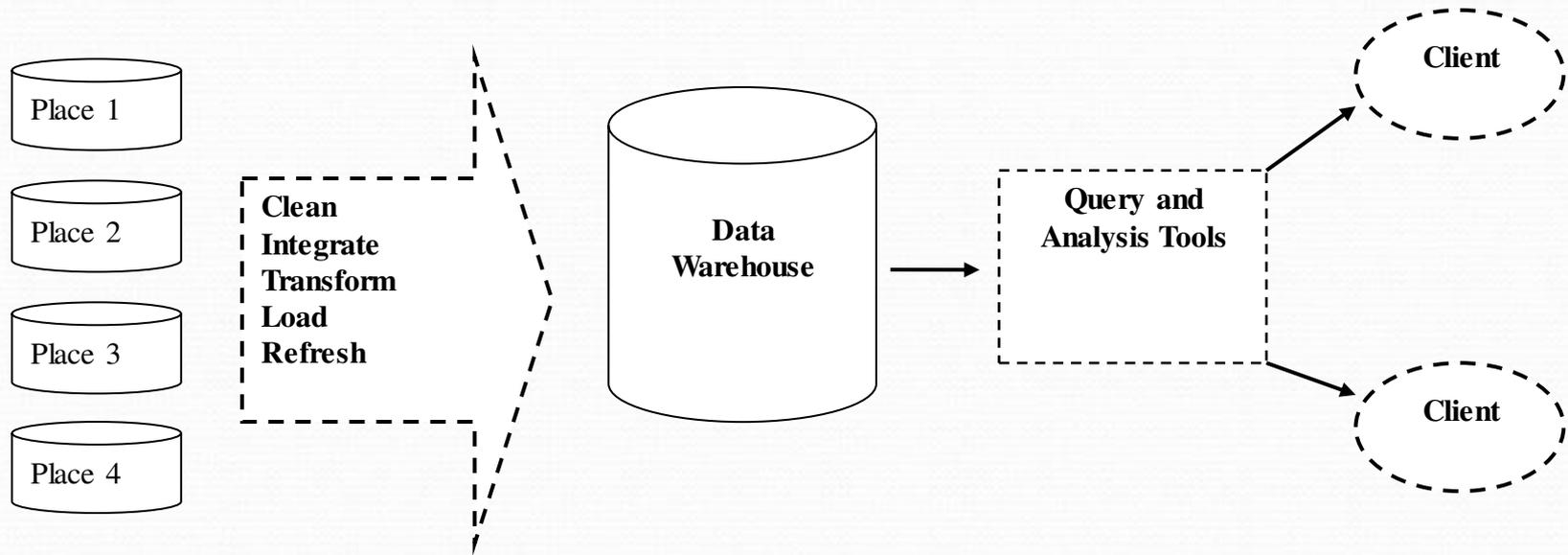
## • Relational Databases

<b>Cust_ID</b>	<b>Name</b>	<b>Address</b>	<b>Age</b>	<b>Income</b>	<b>Category</b>
123 ----	M.Kannan -----	123, south st, -----	34 --	34000 -----	2 -----

## Data Warehouses

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site.

Data warehouse are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing.



## Transactional Databases

A transactional database consists of a file where each record represents a transaction. A transaction typically includes a unique transaction identity number (trans\_ID) and a list of the items making up the transaction.

## ● **Advanced Data and Information Systems and Advanced Applications**

### ● ***Object-Relational Databases***

- A set of **variables** that describe the object (also called attributes)
- A set of **messages** that the object can use to communicate with other objects
- A set of **methods**, where each method holds the code to implement a message.

### ● ***Temporal Databases, Sequence Databases, and Time-Series Databases***

- Temporal database typically stores relational data that including time-related attributes.
- Data mining techniques can be used to find the characteristics of object evolution or the trend of changes for objects in the database.

### ● ***Spatial Databases and Spatiotemporal Databases***

- Spatial database contain spatial-related information
- Geographic database, very large-scale integration or computed-aided design databases, and medical and satellite image databases.
- Geographic databases are commonly used in vehicle navigation and dispatching systems.

- ***Text Databases and Multimedia Databases***

- Text databases are databases that contain word descriptions for objects.
- These word descriptions are usually not simple keywords
- By mining text data, one may uncover general and concise descriptions of the text documents, keyword or content associations
- Multimedia databases store image, audio, and video data.
- Content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces that recognize spoken commands

- ***Heterogeneous Databases and Legacy Databases***

- A heterogeneous database consists of a set of interconnected, autonomous component database

- ***Data Streams***

- Data flow in and out of an observation platform (or window) dynamically
- Power supply, network traffic, stock exchange, telecommunication, Web click streams video surveillance, and weather or environment monitoring



- ***The World Wide Web***

- Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining).
- Automated Web page clustering and classification help group and arrange Web pages in a multidimensional manner based on their contents.
- Web community analysis helps identify hidden Web social networks and communities and observe their evolution.

## • **Data Mining Functionalities**

- Data mining tasks can be classified into two categories
  - Descriptive mining – Characterize the general properties of the data in the database.
  - Predictive mining – Perform inference on the current data in order to make prediction.
- **Concepts/Class Description: Characterization and Discrimination**
  - Data can be associated with classes or concepts
  - Data characterization is a summarization of the general characteristics or features of target class of data.
  - The data corresponding to the user-specified class are typically collected by a database query.
  - The output of data characterization can be pie charts, bar charts, curves, multidimensional data cubes, and multidimensional table, including crosstabs.
- *Data characterization* is a summarization of the general characteristics or features of target class of data. The data corresponding to the user-specified class are typically collected by a database query.

- There are several methods for effective data summarization and characterization. Simple data summaries based on statistical measures.
- An attribute-oriented induction technique can be used to perform data generalization and characterization without step-by-step user interaction.
- The output of data characterization can be presented in various formats. Examples include pie charts, bar charts, curves, multidimensional data cubes, and multidimensional table, including corstabs.
- ***Data discrimination*** is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes.
- The target and contrasting classes can be specified by the user, and the corresponding data objects retrieved through database queries.
- For example, the user may like to compare the general features of software products whose sales increased by 10% in the last year with those whose sales decreased by at least 30% during the same period.

## ● **Mining Frequent Patterns, Associations, and Correlations**

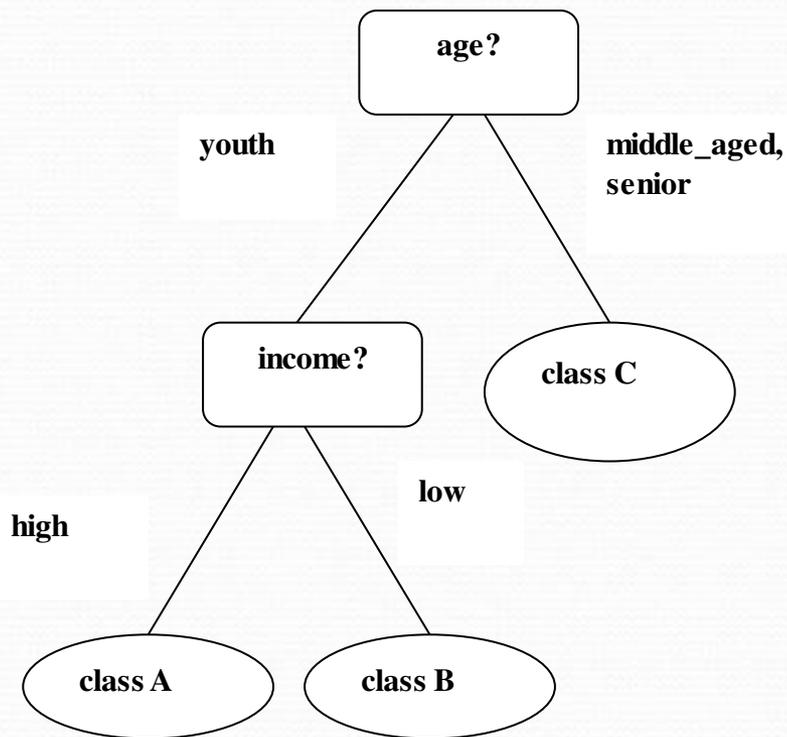
- **Frequent patterns**, are patterns that occur frequently in data. There many kinds of frequent patterns, including **itemsets**, **subsequences**, and **substructures**.
- A **frequent itemset** typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread.
- A frequently occurring **subsequence**, such as the pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is a (frequent) sequential pattern.
- A **substructure** can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern. Mining frequent patterns lead to discovery of interesting associations and correlations within data.

## • **Classification and Prediction**

- *Classification* is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purposes of being able to use the method to predict the class of objects whose class label is unknown.
- The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).
- A decision tree is a flow-chart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions.
- Decision trees can easily be converted to classification rules.
- *Prediction* is used to predict missing or unavailable numerical data values rather than class labels. Regression analysis is a statistical methodology that is most often used for numeric prediction.

- Age(X,"youth") AND income(X,"high")
- Age(X,"youth") AND income(X,"low")
- Age(X,"middle\_aged")
- Age(X,"senior")

class(X,"A")  
class(X,"B")  
class(X,"C")  
class(X,"C")



- **Cluster Analysis**

- Clustering analyzes data objects without consulting a known class label. In general, the data labels are not present in the training data simple because they are not known to begin with. Clustering can be used to generate such labels.
- The objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity.

- **Outlier Analysis**

- A database may contain data objects that do not comply with the general behavior or model of the data. These data objects are **outliers**.
- Most data mining methods discard outliers as noise or exceptions.
- However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones.
- The analysis of outlier data is referred to as **outlier mining**.

- **Example** : Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of extremely large amounts for a given account number in comparison to regular charges incurred by the same account.

- **Evolution Analysis**

- Data **evolution analysis** describes and models regularities or trends for objects whose behavior changes over time.

- **Example:** A data mining study of stock exchange data may identify stock evolution regularities for overall stocks and for the stocks of particular companies.

- **Interestingness of Pattern**

- A data mining system has the potential to generate thousands of patterns, or rules. But only a small fraction of the patterns potentially generated would actually be of interest to any given user.

- An interesting pattern represents **knowledge**.

- Several objective measures of pattern interestingness exist.
- An objective measure for association rules of the form  $S \implies Y$  is rule **support**

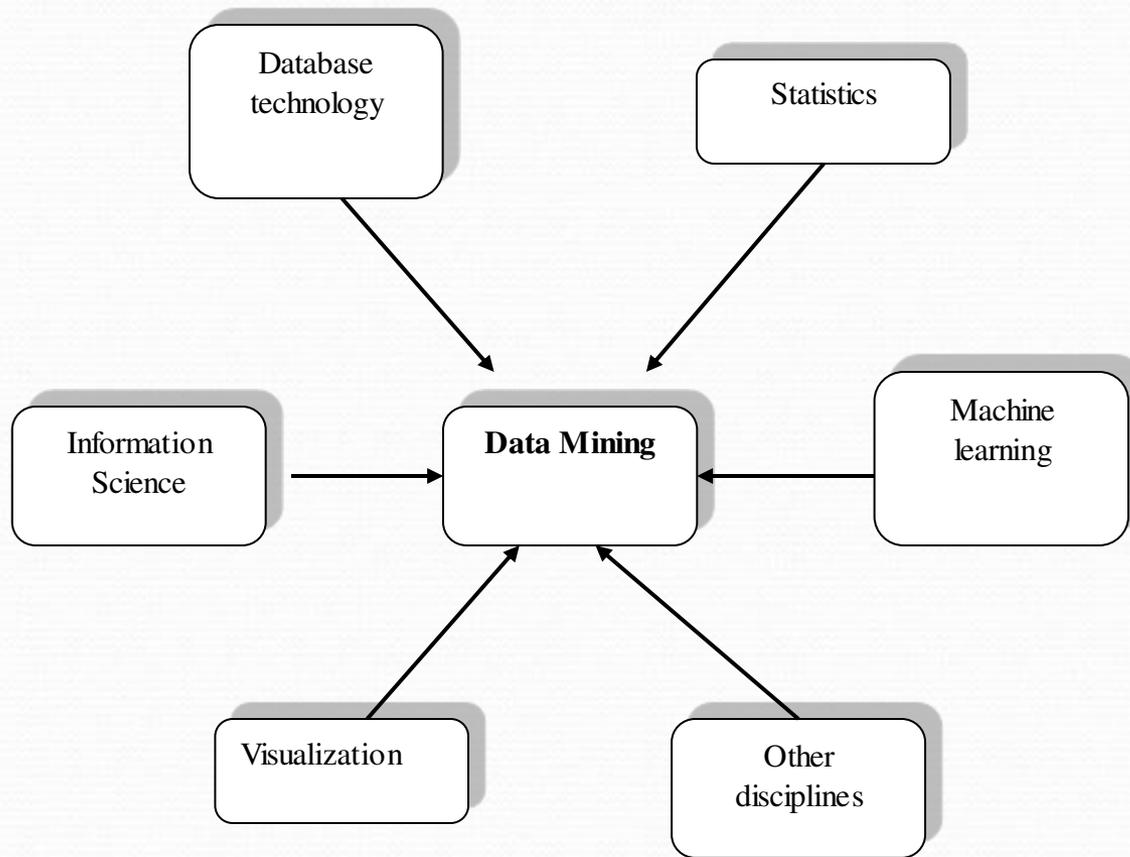
- Another objective measure of association rules is **confidence**
- $\text{support}(X \implies Y) = P(XUY)$
- $\text{confidence}(X \implies Y) = P(Y/X)$

- No. of tuples containing both X and Y
- $\text{support}(X \implies Y) = \frac{\text{No. of tuples containing both X and Y}}{\text{total number of tuples}}$

- $\text{confidence}(X \implies Y) = \frac{\text{No. of tuples\_containing both X and Y}}{\text{Number of tuples containing X}}$

## • **Classification of Data Mining Systems**

- Data mining is an interdisciplinary field, including database systems, statistics, machine learning, visualization, and information science
- Data mining systems can be categorized according to various criteria
- ***Classification according to the kinds of databases mined:***
- A data mining system can be classified according to the kinds of databases mined.
- If classifying according to the special types of data handles, time-series, text stream data, multimedia data mining systems, or World Wide Web mining system.
- ***Classification according to the kinds of techniques utilized:***
- Data mining systems can be categorized according to the underlying data mining techniques employed.
- ***Classification according to the applications adopted:***
- Data mining systems can also be categorized according to the applications they adapt. For example, data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on.



## • **Data Mining Task Primitives**

- A data mining query is defined in terms of **data mining task primitives**. These primitives allow the user interactively communicate with the data mining system during discovery in order to direct the mining process, or examine the findings from different angles or depths.
- The data mining primitives specify the following.
- ***The set of task-relevant data to be mined:*** This specifies the portions of the database or the set of data in which the user is interested. This includes the database attributes or data warehouse dimensions of interest.
- ***The kind of knowledge to be mined:*** This specifies the data mining functions to be performed, such as characterization, discrimination, association or correlation analysis, classification, prediction, clustering, outlier analysis, or evolution analysis.

- The *background knowledge* to be used in the discovery process: This knowledge about the domain to be mined is useful for guiding the knowledge discovery
  - process and for evaluating the patterns found.
- The *interestingness measures and thresholds* for pattern evaluation: They may be used to guide the mining process or , after discovery, to evaluate the discovered patterns. Different kinds of knowledge may have different interestingness measure.
- The expected *representation for visualizing* the discovered patterns: This refers to the form in which discovered patterns are to be displayed, which may include rules, tables, charts, graphs, decision trees, and cubes.

## ● **Integration of a Data Mining System with a Database or Data Warehouse System**

- The possible integration schemes are as follows.

### ● **No coupling:**

- Data mining system will not utilize any function of a Database or Data warehouse system. It may fetch data from a particular source (such as a file system), process data using some data mining algorithms, and then the mining results in another file.

### ● **Loose coupling:**

- Data mining system will use some facilities of a Database or Data warehouse system fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data warehouse.

- **Semitight coupling:**

- Besides linking a Data mining system to Database /Data warehouse system, efficient implementations of a few essential data mining primitives can be provided in the Database/Data warehouse system.

- These primitives can include sorting, indexing, aggregation, histogram analysis, multiway join, and precomputation of some essential statistical measure, such as sum, count, max, min, standard deviation, and so on.

- **Tight coupling:**

- Data mining system is smoothly integrated into the Database/Data warehouse system. The data mining subsystem is treated as one functional component of an information system.

## • **Major Issues in Data Mining**

- The issues in data mining regarding mining methodology are given below.

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

- ***Mining different kinds of knowledge in databases:*** Because different users can be interested in different kinds of knowledge, data mining should cover a wide spectrum of data analysis and knowledge discovery tasks, including data characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis.

- ***Interactive mining of knowledge at multiple levels of abstraction:***

- The data mining process should be interactive.

- Interactive mining allows users to focus the search for patterns, providing and refining data mining requests based on returned results. Specifically, knowledge should be drilling down, rolling up, and pivoting through the data space and knowledge space interactively

- ***Incorporation of background knowledge:***

- Domain knowledge related to databases, such as integrity constraints and deduction rules, can help focus and speed up a data mining process, or judge the interestingness of discovered patterns.

- ***Data mining query languages and ad hoc data mining:***

- Data mining query languages need to be developed to allow users to describe ad hoc data mining tasks by facilitating the specification of the relevant sets of data for analysis, the domain knowledge, the kinds of knowledge to be mined, and the conditions and constraints to be enforced on the discovered patterns.

- ***Presentation and visualization of data mining results:***

- Discovered knowledge should be expressed in high-level languages, visual representations, or other expressive forms so that the knowledge can be easily understood and directly usable by humans.

- ***Handling noisy or incomplete data:***

- The data stored in a database may reflect noise, exceptional cases, or incomplete data objects. When mining data regularities, these objects may confuse the process, causing the knowledge model constructed to overfit the data.

- ***Pattern evaluation-the interestingness problem:***

- A data mining system can uncover thousands of patterns.

# Why Data Preprocessing?

- Data in the real world is dirty
  - **incomplete:** lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=" "
  - **noisy:** containing errors or outliers
    - e.g., Salary="-10"
  - **inconsistent:** containing discrepancies in codes or names
    - e.g., Age="42" Birthday="03/07/1997"
    - e.g., Was rating "1,2,3", now rating "A, B, C"
    - e.g., discrepancy between duplicate records

## Why Is Data Dirty?

- Incomplete data may come from
  - “Not applicable” data value when collected
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments
  - Human or computer error at data entry
  - Errors in data transmission
- Inconsistent data may come from
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

# Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

## Multi-Dimensional Measure of Data Quality

- A well-accepted multidimensional view:
  - Accuracy
  - Completeness
  - Consistency
  - Timeliness
  - Believability
  - Value added
  - Interpretability
  - Accessibility
- Broad categories:
  - Intrinsic, contextual, representational, and accessibility

# Major Tasks in Data Preprocessing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
  - Part of data reduction but with particular importance, especially for numerical data

# Forms of Data Preprocessing

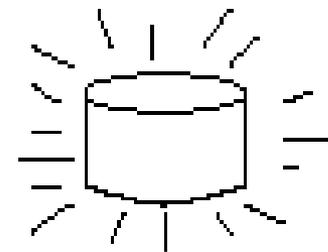
## Data Cleaning

[water to clean dirty-looking data]

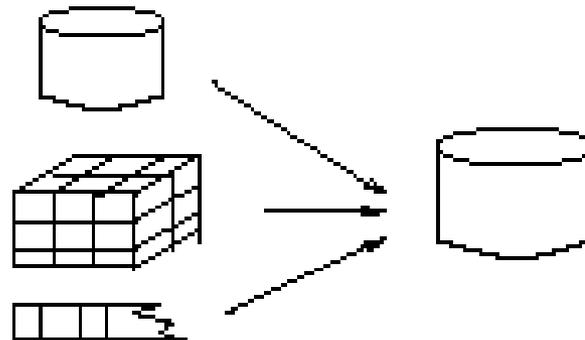


[show soap suds on data]

[‘clean’-looking data]



## Data Integration



## Data Transformation

-2, 32, 100, 59, 48



-0.02, 0.32, 1.00, 0.59, 0.48

## Data Reduction

	A1	A2	A3	...	A126
T1					
T2					
T3					
T4					
...					
T2000					



	A1	A3	...	A115
T1				
T4				
...				
T1456				

# Measuring the Central Tendency

## • Mean (algebraic measure) (sample vs. population):

- Weighted arithmetic mean:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \mu = \frac{\sum x}{N}$$

- Trimmed mean: chopping extreme values

## • Median: A holistic measure

$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

- Middle value if odd number of values, or average of the middle two values otherwise
- Estimated by interpolation (for *grouped data*):

## • Mode

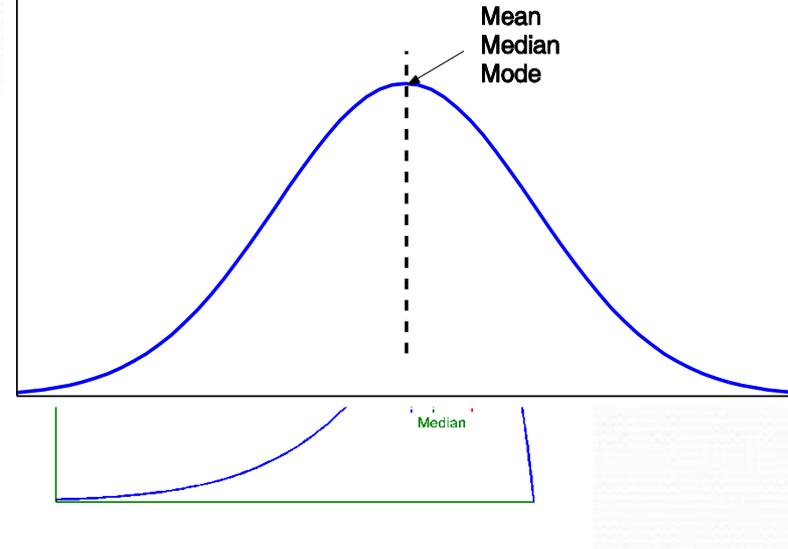
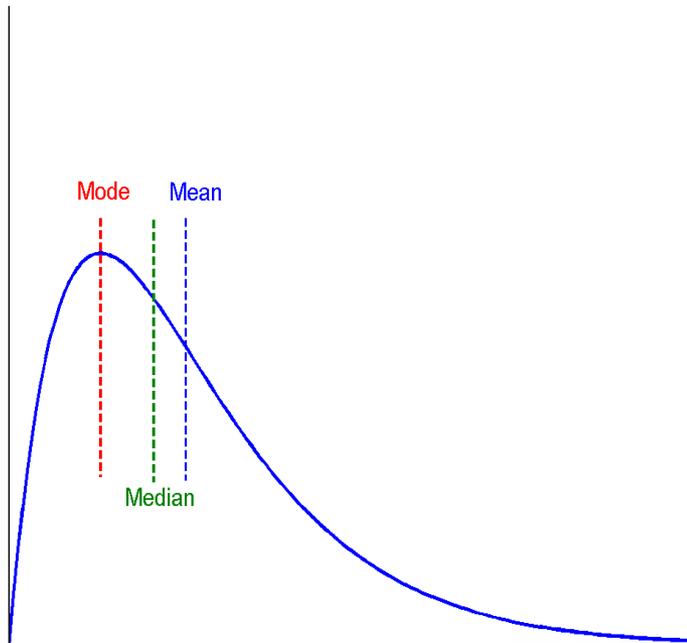
- Value that occurs most frequently in the data
- Unimodal, bimodal, trimodal
- Empirical formula:

$$median = L_1 + \left( \frac{n/2 - (\sum f)l}{f_{median}} \right) c$$

$$mean - mode = 3 \times (mean - median)$$

# Symmetric vs. Skewed Data

- Median, mean and mode of symmetric, positively and negatively skewed data



# Measuring the Dispersion of Data

- Quartiles, outliers and boxplots

- **Quartiles:**  $Q_1$  (25<sup>th</sup> percentile),  $Q_3$  (75<sup>th</sup> percentile)
- **Inter-quartile range:**  $IQR = Q_3 - Q_1$
- **Five number summary:** min,  $Q_1$ , M,  $Q_3$ , max
- **Boxplot:** ends of the box are the quartiles, median is marked, whiskers, and plot outlier individually

- **Outlier:** usually, a value higher/lower than  $1.5 \times IQR$   
$$\frac{1}{N} \sum_{i=1}^n (x_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^n x_i^2 - \mu^2$$

- Variance and standard deviation (*sample: s, population:  $\sigma$* )

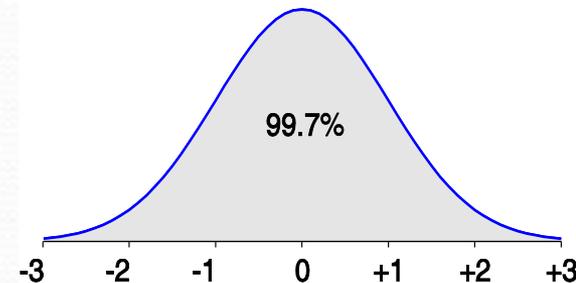
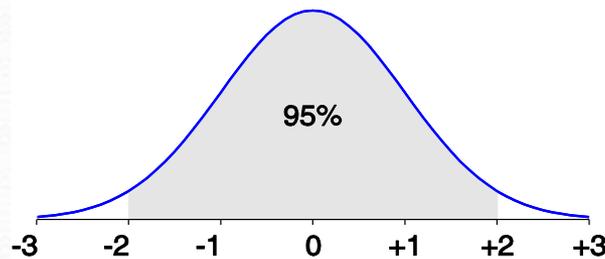
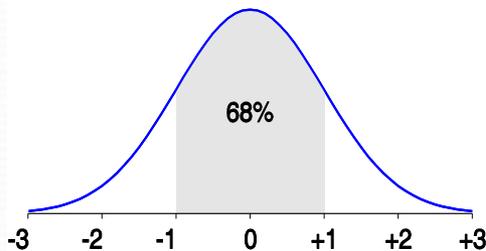
- **Variance:** (algebraic, scalable computation)

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

- **Standard deviation**  $s$  (*or*  $\sigma$ ) is the square root of variance  $s^2$  (*or*  $\sigma^2$ )

# Properties of Normal Distribution Curve

- The normal (distribution) curve
  - From  $\mu - \sigma$  to  $\mu + \sigma$ : contains about 68% of the measurements ( $\mu$ : mean,  $\sigma$ : standard deviation)
  - From  $\mu - 2\sigma$  to  $\mu + 2\sigma$ : contains about 95% of it
  - From  $\mu - 3\sigma$  to  $\mu + 3\sigma$ : contains about 99.7% of it



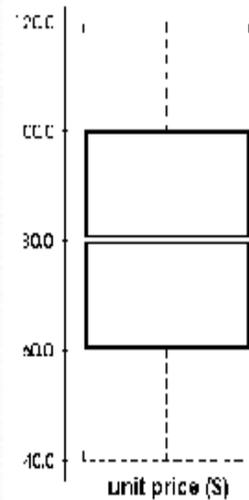
# Boxplot Analysis

- Five-number summary of a distribution:

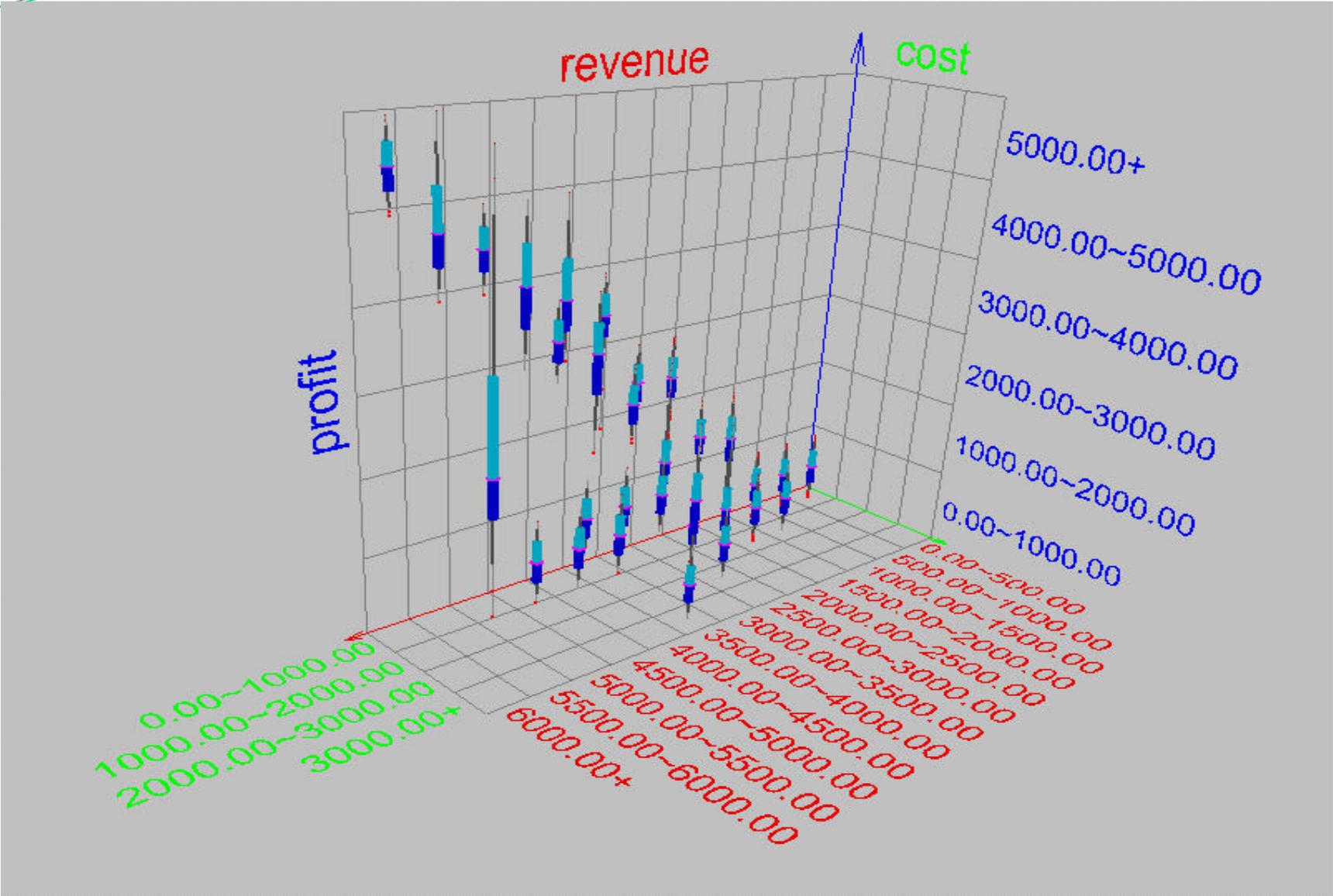
Minimum, Q1, M, Q3, Maximum

- Boxplot

- Data is represented with a box
- The ends of the box are at the first and third quartiles, i.e., the height of the box is IRQ
- The median is marked by a line within the box
- Whiskers: two lines outside the box extend to Minimum and Maximum

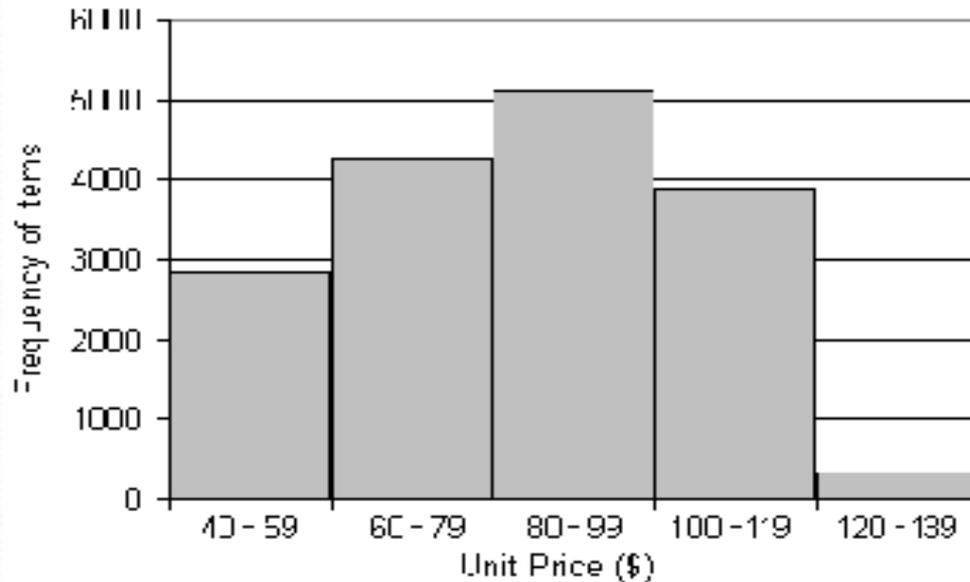


# Visualization of Data Dispersion: Boxplot Analysis



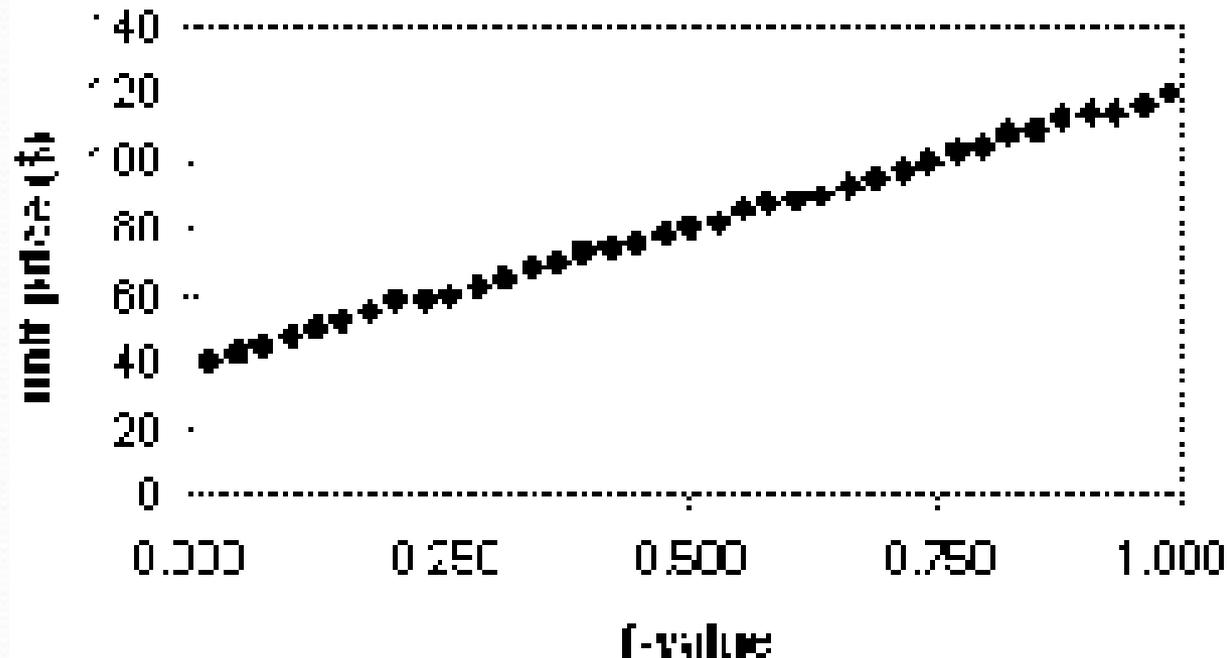
# Histogram Analysis

- Graph displays of basic statistical class descriptions
  - Frequency histograms
    - A univariate graphical method
    - Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data



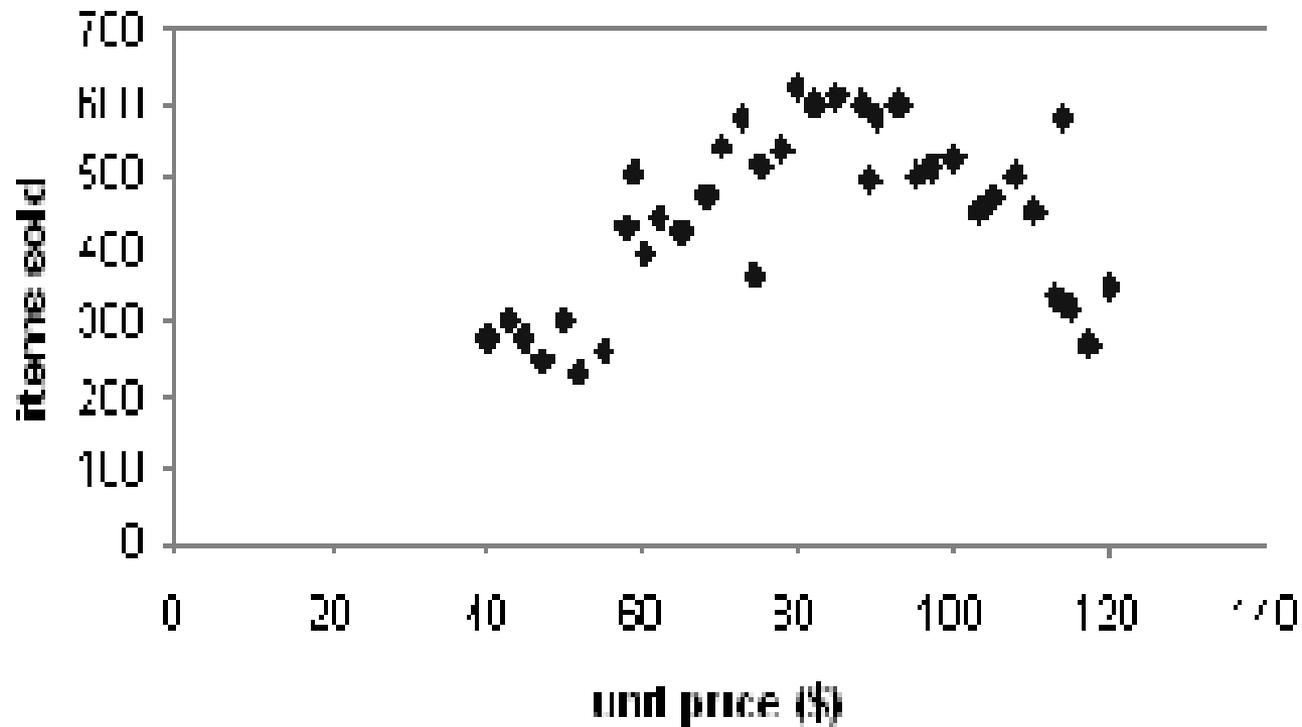
# Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots **quantile** information
  - For a data  $x_i$  data sorted in increasing order,  $f_i$  indicates that approximately  $100 f_i\%$  of the data are below or equal to the value  $x_i$

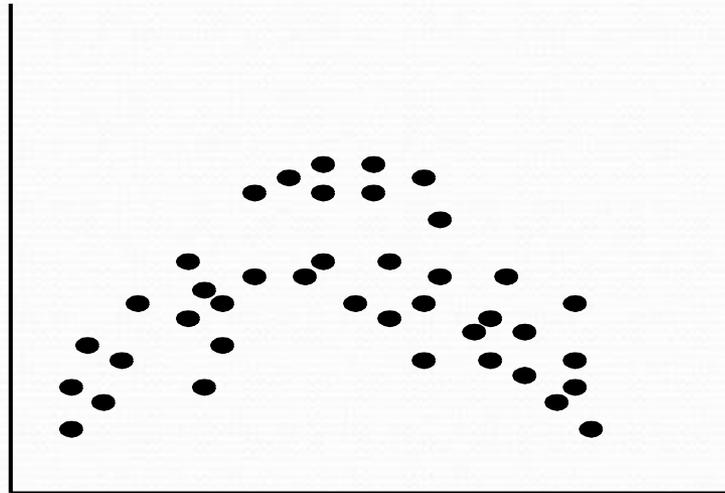
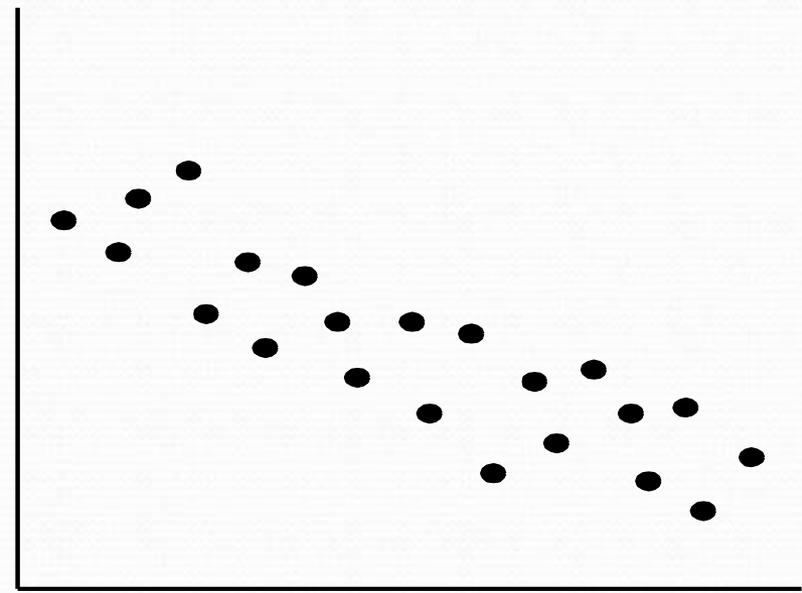
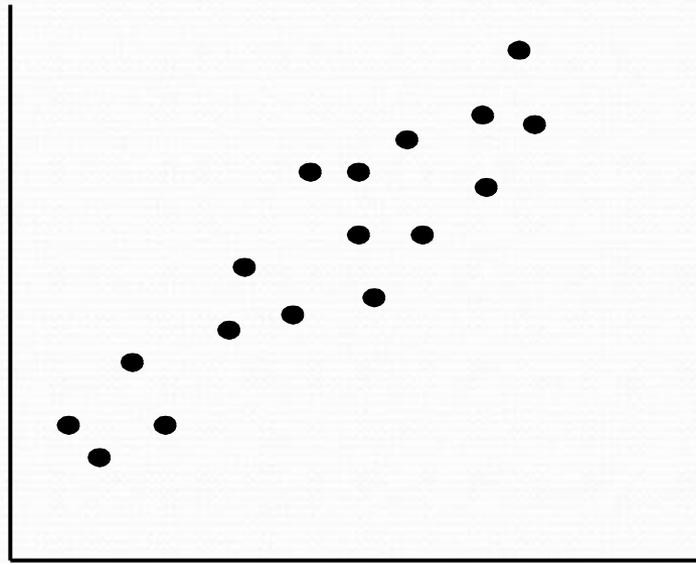


# Scatter plot

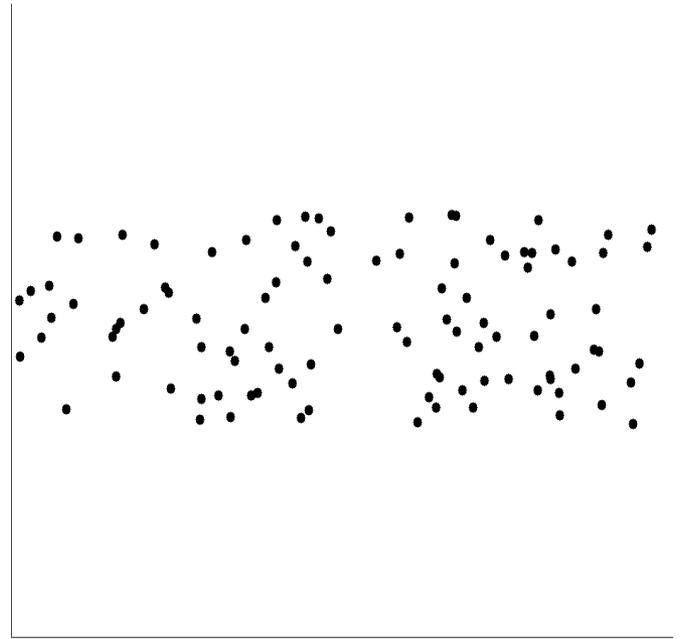
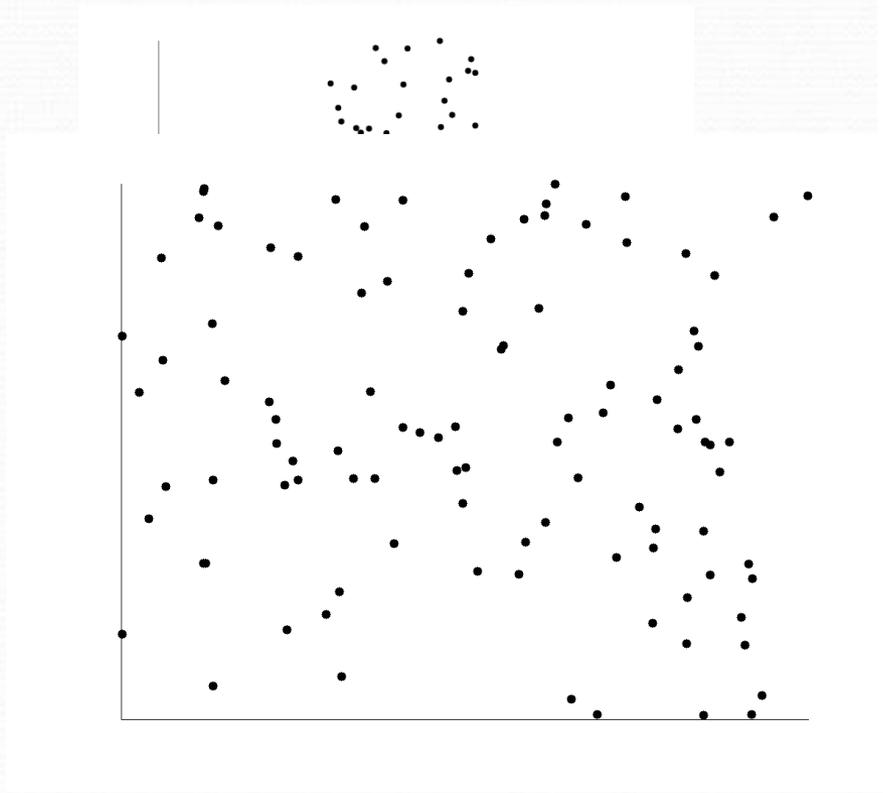
- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane



# Positively and Negatively Correlated Data



# Not Correlated Data



## Data Cleaning

- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data
  - Resolve redundancy caused by data integration

## ● **Missing Data**

### ● Data is not always available

- E.g., many tuples have no recorded value for several attributes, such as customer income in sales data

### ● Missing data may be due to

- equipment malfunction
- inconsistent with other recorded data and thus deleted
- data not entered due to misunderstanding
- certain data may not be considered important at the time of entry
- not register history or changes of the data

### ● Missing data may need to be inferred.

## • **How to Handle Missing Data?**

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant: e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

- **Noisy Data**

- Noise: random error or variance in a measured variable

- Incorrect attribute values may due to

- faulty data collection instruments

- data entry problems

- data transmission problems

- technology limitation

- inconsistency in naming convention

- Other data problems which requires data cleaning

- duplicate records

- incomplete data

- inconsistent data

## • How to Handle Noisy Data?

### • Binning

- first sort data and partition into (equal-frequency) bins
- then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.

### • Regression

- smooth by fitting the data into regression functions

### • Clustering

- detect and remove outliers

### • Combined computer and human inspection

- detect suspicious values and check by human (e.g., deal with possible outliers)

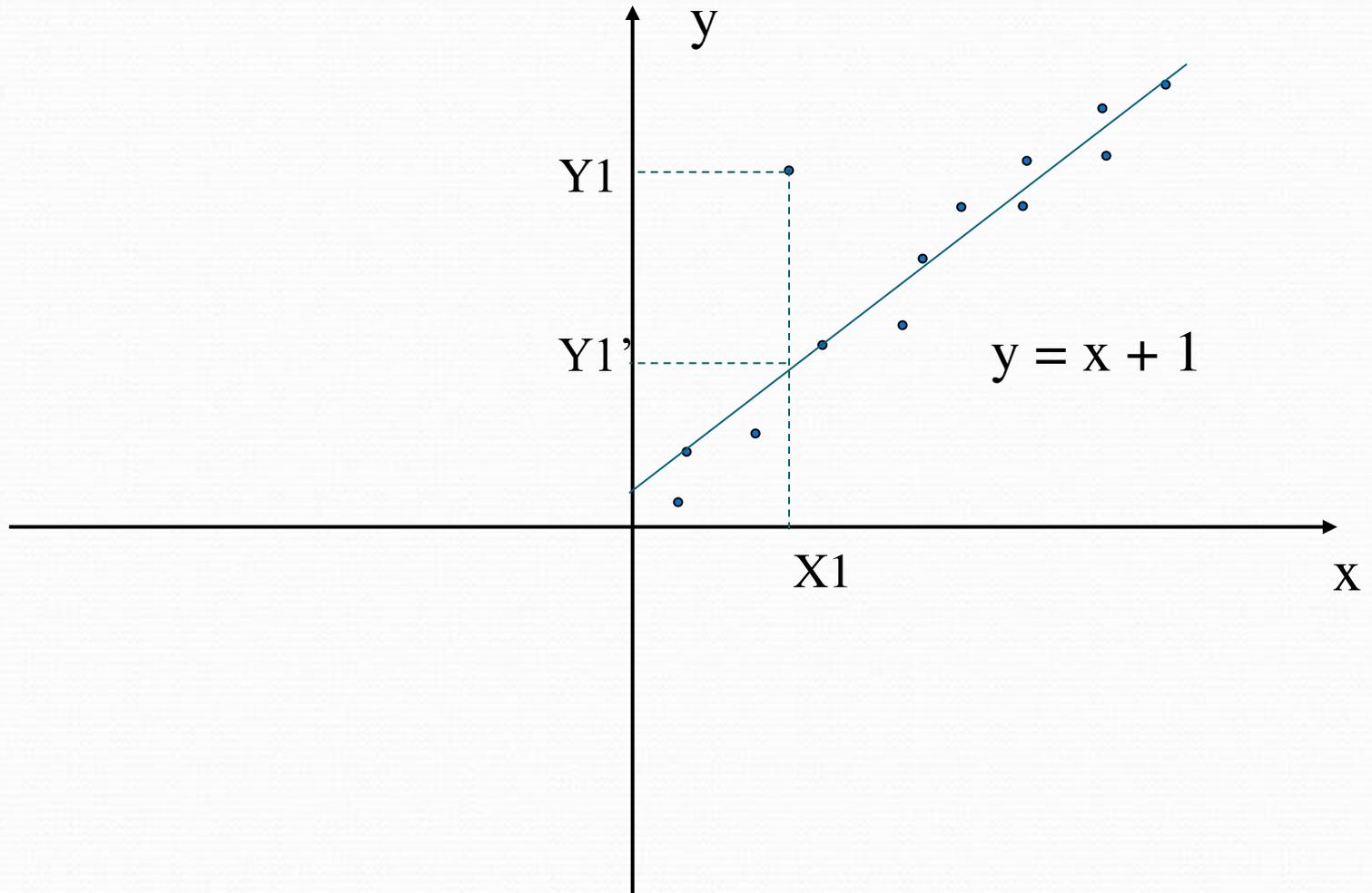
- **Equal-width** (distance) partitioning
  - Divides the range into  $N$  intervals of equal size: uniform grid
  - if  $A$  and  $B$  are the lowest and highest values of the attribute, the width of intervals will be:  $W = (B - A)/N$ .
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well
- **Equal-depth** (frequency) partitioning
  - Divides the range into  $N$  intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

## Binning Methods for Data Smoothing

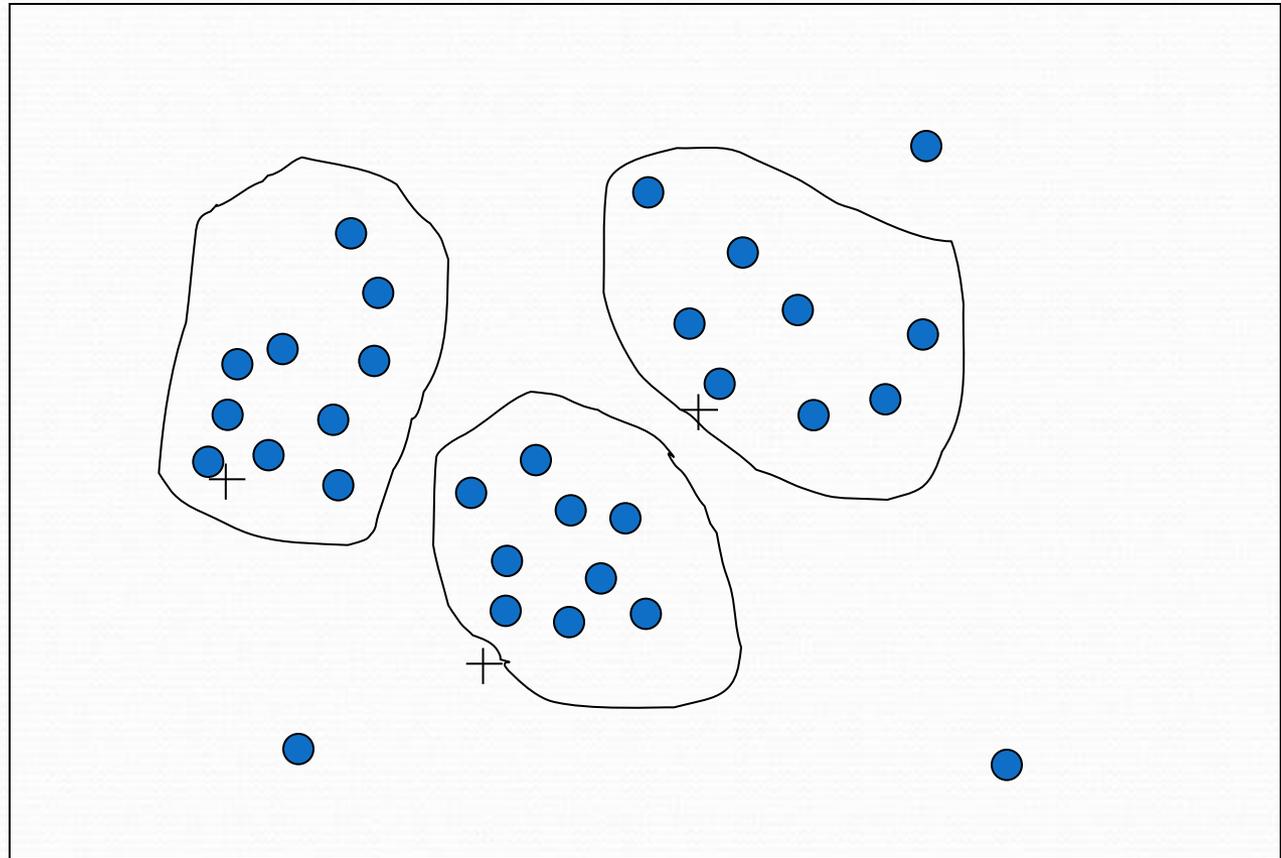
Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

- \* Partition into equal-frequency (equi-depth) bins:
  - - Bin 1: 4, 8, 9, 15
  - - Bin 2: 21, 21, 24, 25
  - - Bin 3: 26, 28, 29, 34
- \* Smoothing by bin means:
  - - Bin 1: 9, 9, 9, 9
  - - Bin 2: 23, 23, 23, 23
  - - Bin 3: 29, 29, 29, 29
- \* Smoothing by bin boundaries:
  - - Bin 1: 4, 4, 4, 15
  - - Bin 2: 21, 21, 25, 25
  - - Bin 3: 26, 26, 26, 34

# Regression



# Cluster Analysis



- Data integration:
  - Combines data from multiple sources into a coherent store
- Schema integration: e.g.,  $A.cust-id \equiv B.cust-\#$ 
  - Integrate metadata from different sources
- **Entity identification problem:**
  - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
  - For the same real world entity, attribute values from different sources are different
  - Possible reasons: different representations, different scales, e.g., metric vs. British units

- Redundant data occur often when integration of multiple databases
  - *Object identification*: The same attribute or object may have different names in different databases
  - *Derivable data*: One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{\sum (A - \bar{A})(B - \bar{B})}{(n-1)\sigma_A\sigma_B} = \frac{\sum (AB) - n\bar{A}\bar{B}}{(n-1)\sigma_A\sigma_B}$$

where  $n$  is the number of tuples,  $\bar{A}$  and  $\bar{B}$  are the respective means of  $A$  and  $B$ ,  $\sigma_A$  and  $\sigma_B$  are the respective standard deviation of  $A$  and  $B$ , and  $\sum(AB)$  is the sum of the  $AB$  cross-product.

- If  $r_{A,B} > 0$ ,  $A$  and  $B$  are positively correlated ( $A$ 's values increase as  $B$ 's). The higher, the stronger correlation.
- $r_{A,B} = 0$ : independent;  $r_{A,B} < 0$ : negatively correlated

- $\chi^2$  (chi-square) test

$$\chi^2 = \sum \frac{(\textit{Observed} - \textit{Expected})^2}{\textit{Expected}}$$

- The larger the  $\chi^2$  value, the more likely the variables are related
- The cells that contribute the most to the  $\chi^2$  value are those whose actual count is very different from the expected count
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones

- Why data reduction?
  - A database/data warehouse may store terabytes of data
  - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
  - Data cube aggregation:
  - Dimensionality reduction — e.g., remove unimportant attributes
  - Data Compression
  - Numerosity reduction — e.g., fit data into models
  - Discretization and concept hierarchy generation

## Data Cube Aggregation

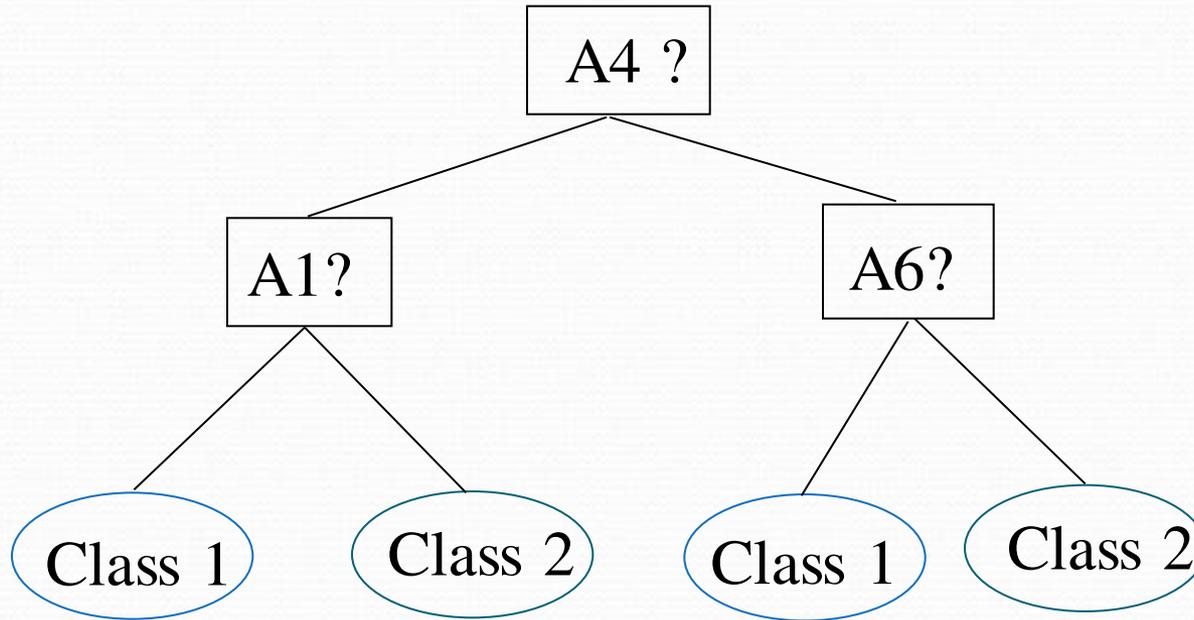
- The lowest level of a data cube (base cuboid)
  - The aggregated data for an **individual entity of interest**
  - E.g., a customer in a phone calling data warehouse
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
  - Step-wise forward selection
  - Step-wise backward elimination
  - Combining forward selection and backward elimination
  - Decision-tree induction

## Example of Decision Tree Induction

Initial attribute set:

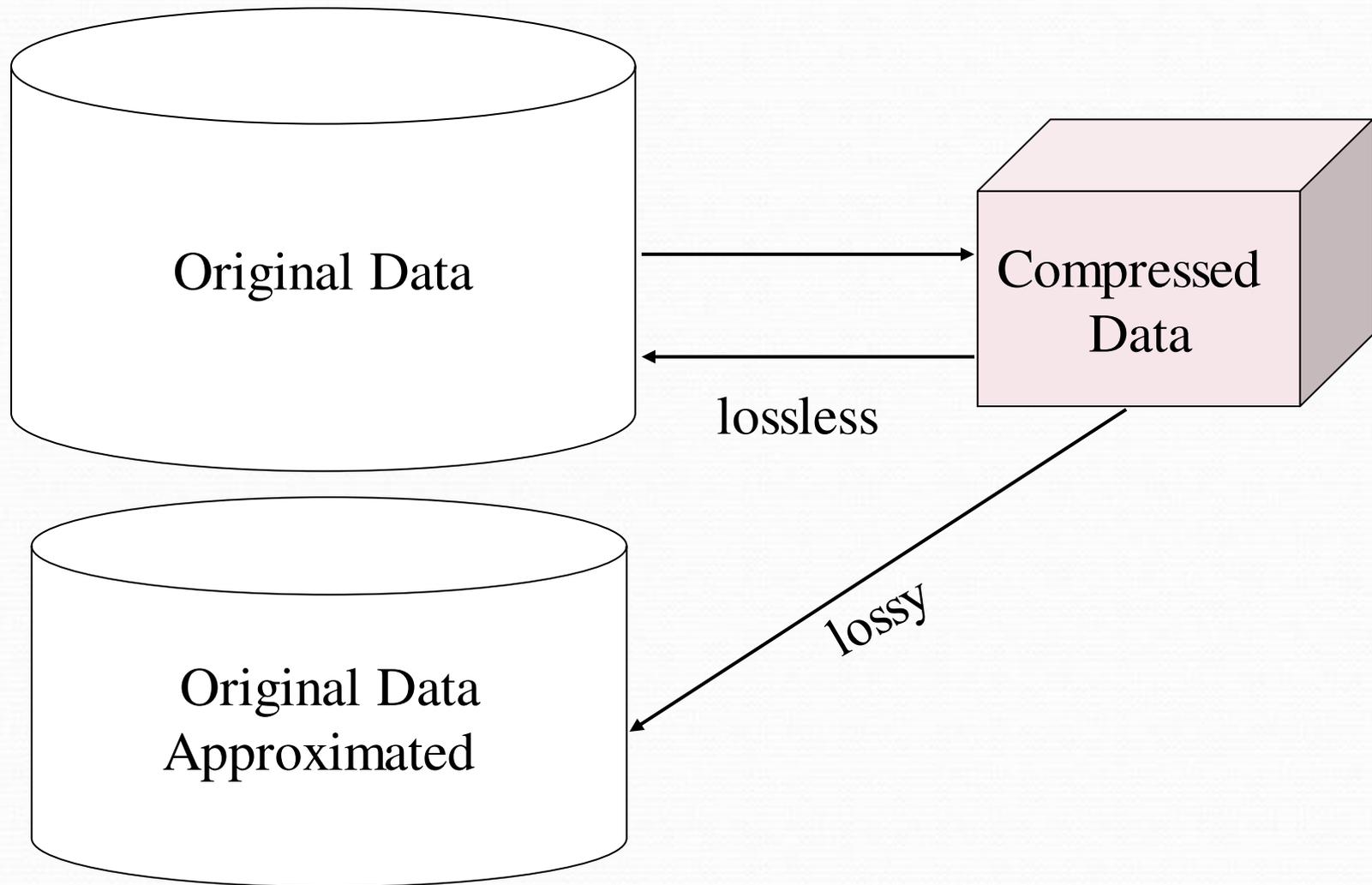
{A1, A2, A3, A4, A5, A6}



-----> Reduced attribute set: {A1, A4, A6}

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless
  - But only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time

# Data Compression

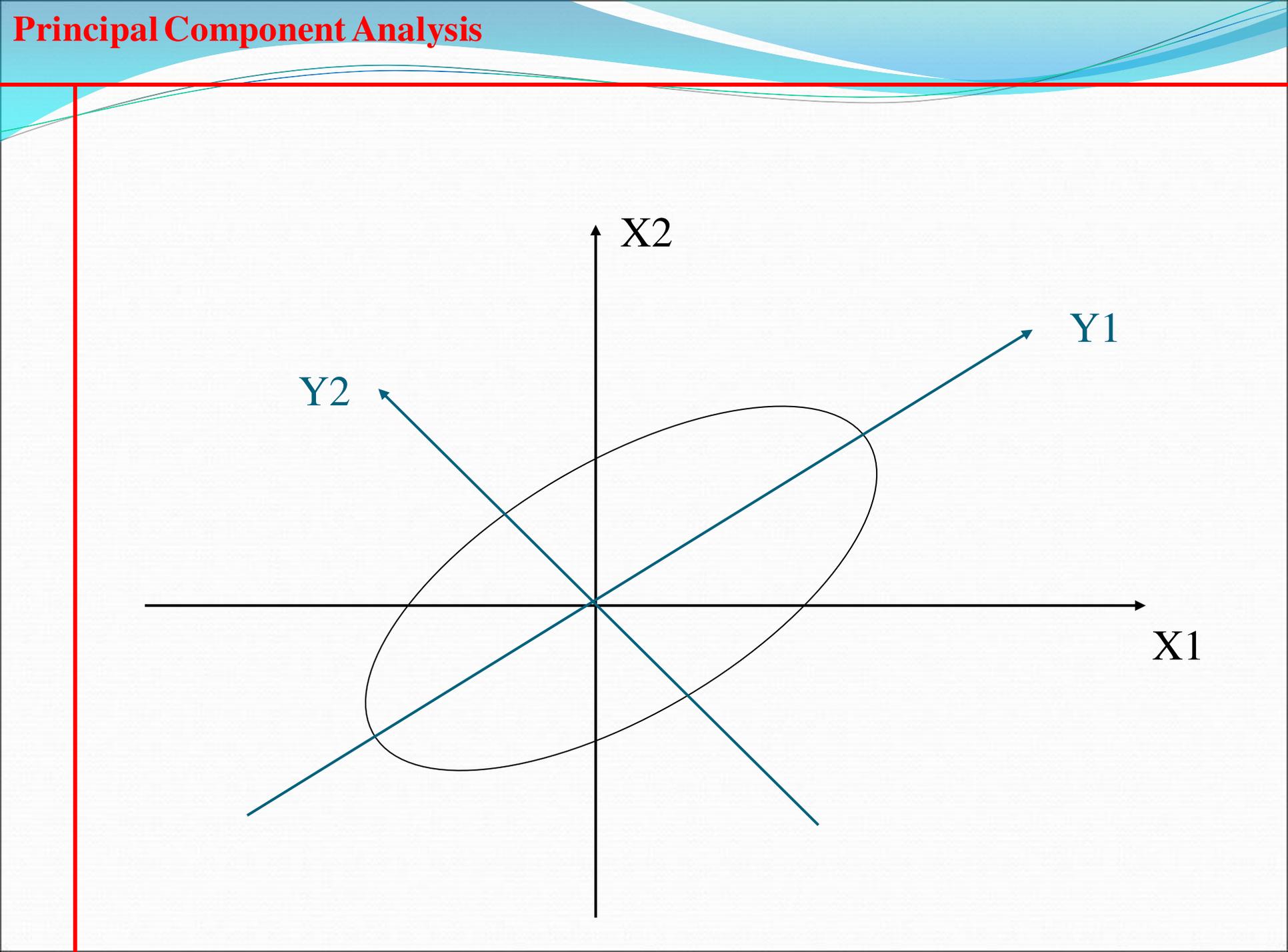


## Dimensionality Reduction: Wavelet Transformation

- Discrete wavelet transform (DWT): linear signal processing, multi-resolutional analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
  - Length,  $L$ , must be an integer power of 2 (padding with 0's, when necessary)
  - Each transform has 2 functions: smoothing, difference
  - Applies to pairs of data, resulting in two set of data of length  $L/2$
  - Applies two functions recursively, until reaches the desired length

## Dimensionality Reduction: Principal Component Analysis (PCA)

- Given  $N$  data vectors from  $n$ -dimensions, find  $k \leq n$  orthogonal vectors (*principal components*) that can be best used to represent data
- Steps
  - Normalize input data: Each attribute falls within the same range
  - Compute  $k$  orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the  $k$  principal component vectors
  - The principal components are sorted in order of decreasing “significance” or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only
- Used when the number of dimensions is large



# Principal Component Analysis

$X_2$

$Y_1$

$Y_2$

$X_1$

## Numerosity Reduction

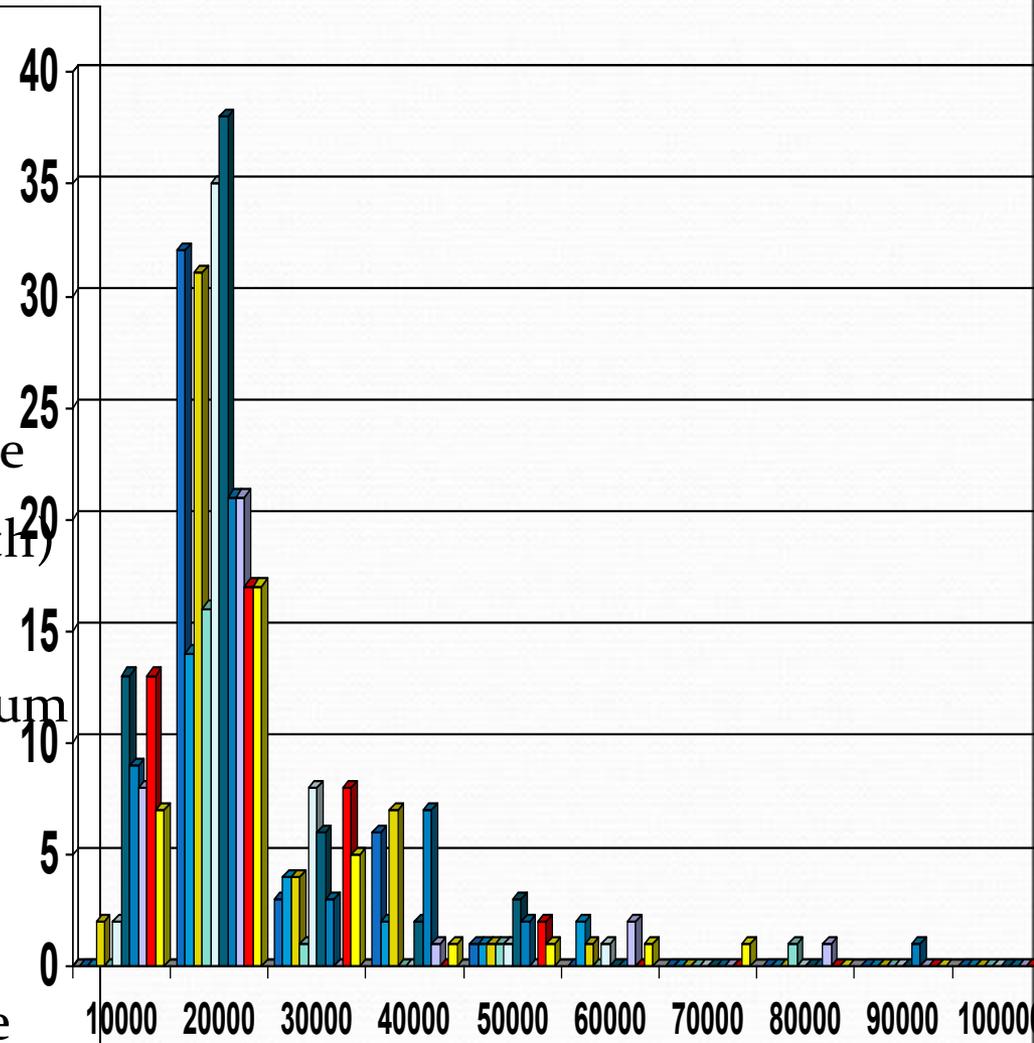
- Reduce data volume by choosing alternative, smaller forms of data representation
- Parametric methods
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Example: Log-linear models—obtain value at a point in  $m$ -D space as the product on appropriate marginal subspaces
- Non-parametric methods
  - Do not assume models
  - Major families: histograms, clustering, sampling

- Linear regression: Data are modeled to fit a straight line
  - Often uses the least-square method to fit the line
- Multiple regression: allows a response variable  $Y$  to be modeled as a linear function of multidimensional feature vector
- Log-linear model: approximates discrete multidimensional probability distributions

# Data Reduction Method (2):

## Histograms

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)
  - V-optimal: with the least *histogram variance* (weighted sum of the original values that each bucket represents)
  - MaxDiff: set bucket boundary between each pair for pairs have the  $\beta-1$  largest differences



## Data Reduction Method (3): Clustering

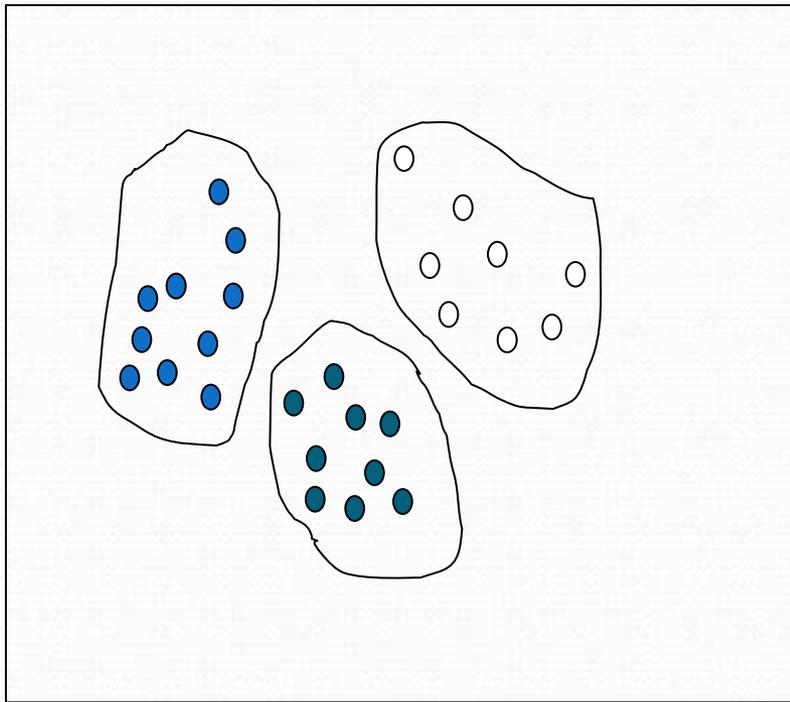
- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms

## Data Reduction Method (4): Sampling

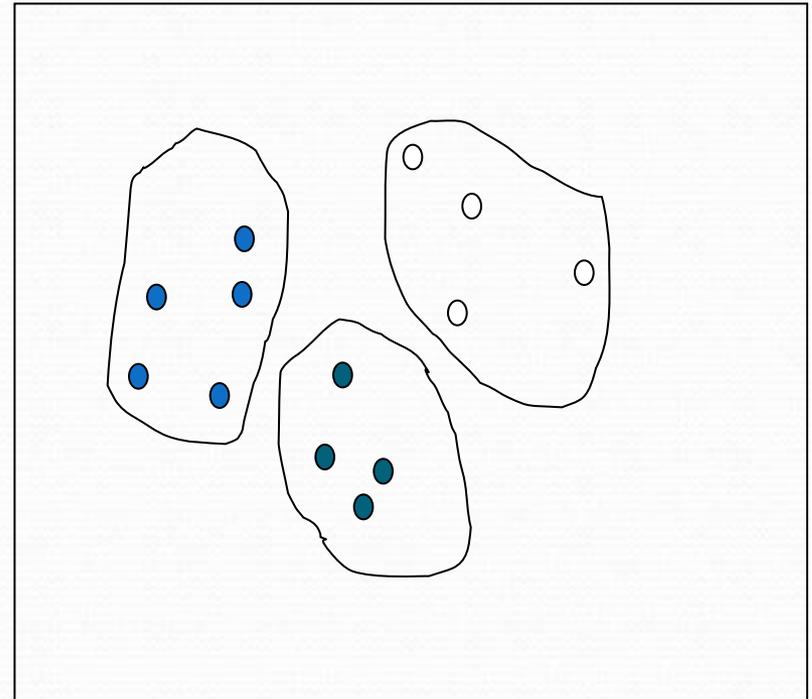
- Sampling: obtaining a small sample  $s$  to represent the whole data set  $N$
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a **representative** subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Note: Sampling may not reduce database I/Os (page at a time)

# Sampling: Cluster or Stratified Sampling

Raw Data



Cluster/Stratified Sample



- Three types of attributes:
  - Nominal — values from an unordered set, e.g., color, profession
  - Ordinal — values from an ordered set, e.g., military or academic rank
  - Continuous — real numbers, e.g., integer or real numbers
- Discretization:
  - Divide the range of a continuous attribute into intervals
  - Some classification algorithms only accept categorical attributes.
  - Reduce data size by discretization
  - Prepare for further analysis

# Discretization and Concept Hierarchy

- Discretization
  - Reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
- Concept hierarchy formation
  - Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as young, middle-aged, or senior)

- Typical methods: All the methods can be applied recursively
  - Binning (covered above)
    - Top-down split, unsupervised,
  - Histogram analysis (covered above)
    - Top-down split, unsupervised
  - Clustering analysis (covered above)
    - Either top-down split or bottom-up merge, unsupervised
  - Entropy-based discretization: supervised, top-down split
  - Interval merging by  $\chi^2$  Analysis: unsupervised, bottom-up merge
  - Segmentation by natural partitioning: top-down split, unsupervised

## Entropy-Based Discretization

- Given a set of samples  $S$ , if  $S$  is partitioned into two intervals  $S_1$  and  $S_2$  using boundary  $T$ , the information gain after partitioning is

$$I(S, T) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2)$$

- Entropy is calculated based on class distribution of the samples in the set. Given  $m$  classes, the entropy of  $S_1$  is

$$\text{Entropy}(S_1) = -\sum_{i=1}^m p_i \log_2(p_i)$$

where  $p_i$  is the probability of class  $i$  in  $S_1$

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization
- The process is recursively applied to partitions obtained until some stopping criterion is met
- Such a boundary may reduce data size and improve

## Segmentation by Natural Partitioning

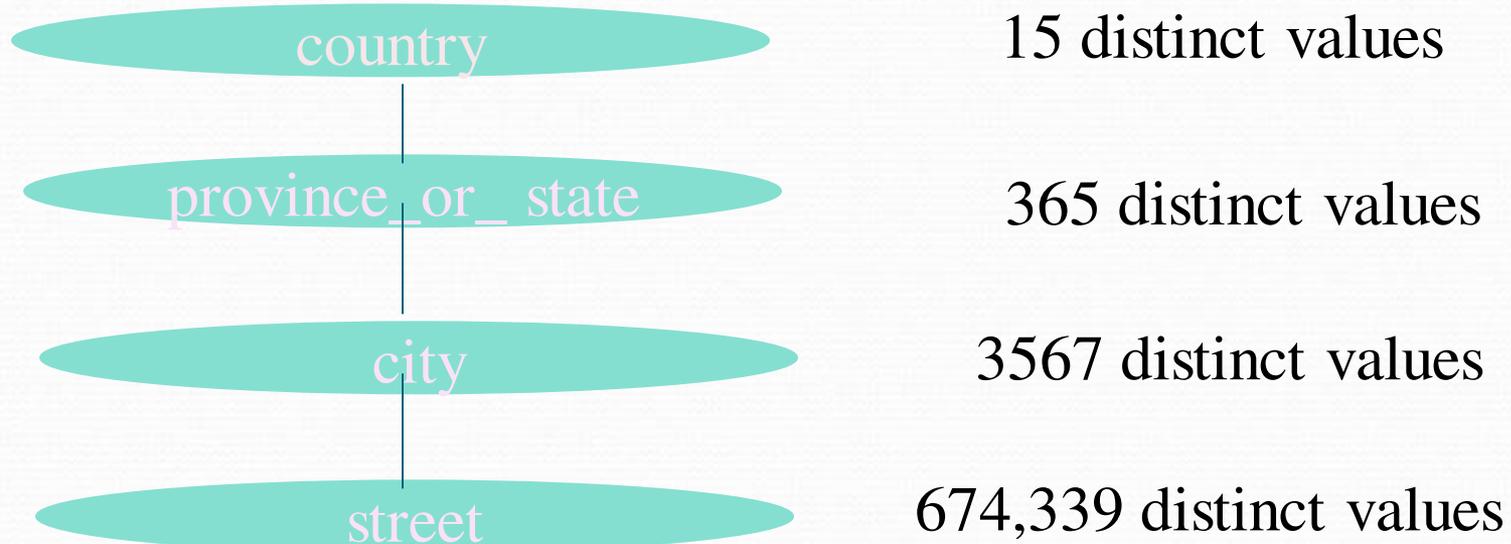
- A simply 3-4-5 rule can be used to segment numeric data into relatively uniform, “natural” intervals.
  - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
  - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
  - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals



- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - street < city < state < country
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only street < city, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {street, city, state, country}

## Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year



U  
N  
I  
T  
IV

# Association Rule Mining and Classification



# What Is Association Mining?

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples.
  - Rule form: “Body  $\rightarrow$  Head [support, confidence]”.
  - $\text{buys}(x, \text{“diapers”}) \rightarrow \text{buys}(x, \text{“beers”}) [0.5\%, 60\%]$
  - $\text{major}(x, \text{“CS”}) \wedge \text{takes}(x, \text{“DB”}) \rightarrow \text{grade}(x, \text{“A”}) [1\%, 75\%]$

# Association Rule: Basic Concepts

$I = \{ I_1, I_2, I_3, \dots, I_m \}$  set of items

$T$  – transactions such that  $T \subseteq I$ .

- **TID** – Transaction ID
- Set of items called **itemset**
- Itemset contains  $k$ -items then it is  **$k$ -itemset**

## ■ **Support count**

The occurrence frequency of an itemset is the number of transactions that contain the itemset.

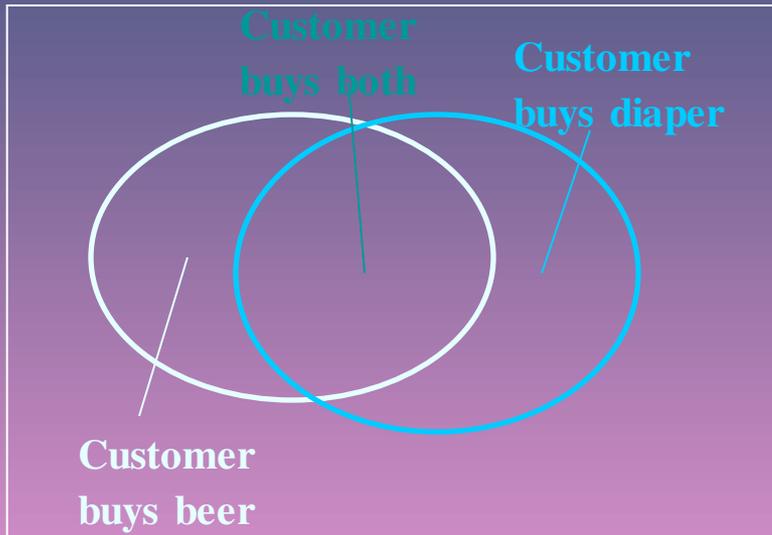
## ■ **Frequent itemset**

The support count satisfies a predefined minimum support threshold

# Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - \*  $\Rightarrow$  *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics*  $\Rightarrow$  \* (What other products should the store stocks up?)
  - Attached mailing in direct marketing
  - Detecting “ping-pong”ing of patients, faulty “collisions”

# Rule Measures: Support and Confidence



- Find all the rules  $X \& Y \Rightarrow Z$  with minimum confidence and support
  - support,  $s$ , probability that a transaction contains  $\{X \cup Y \cup Z\}$
  - confidence,  $c$ , conditional probability that a transaction having  $\{X \cap Y\}$  also contains  $Z$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

*Let minimum support 50%, and minimum confidence 50%, we have*

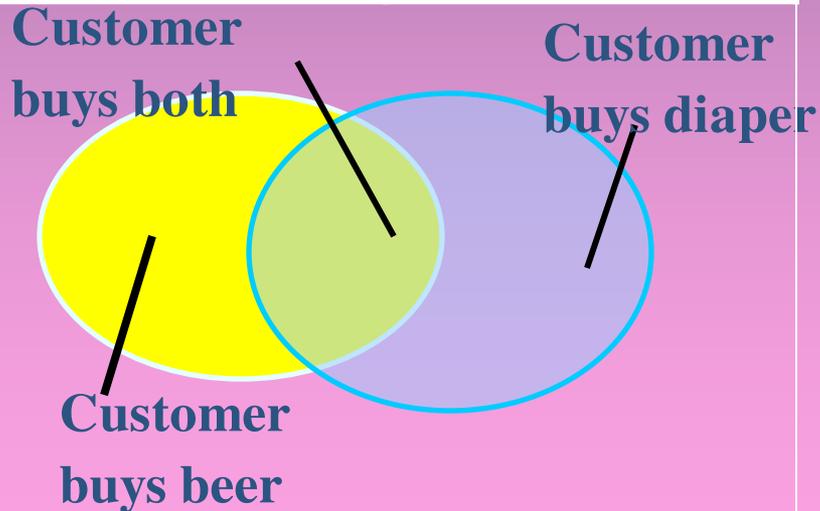
- $A \Rightarrow C$  (50%, 66.6%)
- $C \Rightarrow A$  (50%, 100%)

# Association Rule Mining: A Road Map

- Boolean vs. quantitative associations (Based on the types of values handled)
  - $\text{buys}(x, \text{"SQLServer"}) \wedge \text{buys}(x, \text{"DMBook"}) \rightarrow \text{buys}(x, \text{"DBMiner"})$  [0.2%, 60%]
  - $\text{age}(x, \text{"30..39"}) \wedge \text{income}(x, \text{"42..48K"}) \rightarrow \text{buys}(x, \text{"PC"})$  [1%, 75%]
- Single dimension vs. multiple dimensional associations (see ex. Above)
- Single level vs. multiple-level analysis
  - What brands of beers are associated with what brands of diapers?
- Various extensions
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

# Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset  $X = \{x_1, \dots, x_k\}$
- Find all the rules  $X \rightarrow Y$  with minimum support and confidence
  - support,  $s$ , probability that a transaction contains  $X \cup Y$
  - confidence,  $c$ , conditional probability that a transaction having  $X$  also contains  $Y$

Let  $sup_{min} = 50\%$ ,  $conf_{min} = 50\%$

Freq. Pat.:  $\{A:3, B:3, D:4, E:3, AD:3\}$

Association rules:

$A \rightarrow D$  (60%, 100%)

$D \rightarrow A$  (60%, 75%)

# What Is Frequent Pattern Analysis?

---

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

# Why Is Freq. Pattern Mining Important?

---

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

# Closed Patterns and Max-Patterns

---

- A long pattern contains a combinatorial number of sub-patterns, e.g.,  $\{a_1, \dots, a_{100}\}$  contains  $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = 1.27 \cdot 10^{30}$  sub-patterns!
- Solution: Mine *closed patterns* and *max-patterns* instead
- An itemset  $X$  is **closed** if  $X$  is *frequent* and there exists *no super-pattern*  $Y \supset X$ , with the same support as  $X$
- An itemset  $X$  is a **max-pattern** if  $X$  is frequent and there exists no frequent super-pattern  $Y \supset X$  Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

# Closed Patterns and Max-Patterns

---

- Exercise.  $DB = \{ \langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle \}$ 
  - $Min\_sup = 1$ .
- What is the set of **closed itemset**?
  - $\langle a_1, \dots, a_{100} \rangle: 1$
  - $\langle a_1, \dots, a_{50} \rangle: 2$
- What is the set of **max-pattern**?
  - $\langle a_1, \dots, a_{100} \rangle: 1$
- What is the set of **all patterns**?
  - !!

# Scalable Methods for Mining Frequent Patterns

---

- The **downward closure** property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

# Apriori: A Candidate Generation-and-Test Approach

---

- Apriori pruning principle: If there is **any** itemset which is infrequent, its superset should not be generated/tested!  
(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - **Generate** length  $(k+1)$  **candidate** itemsets from length  $k$  **frequent** itemsets
  - **Test** the candidates against DB
  - Terminate when no frequent or candidate set can be generated

# The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1<sup>st</sup> scan

$C_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

$L_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

$C_2$

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

2<sup>nd</sup> scan

$C_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

$L_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

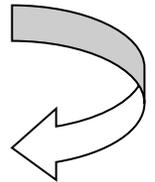
$C_3$

Itemset
{B, C, E}

3<sup>rd</sup> scan

$L_3$

Itemset	sup
{B, C, E}	2



# The Apriori Algorithm

---

- Pseudo-code:

$C_k$ : Candidate itemset of size  $k$

$L_k$ : frequent itemset of size  $k$

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **do begin**

$C_{k+1} =$  candidates generated from  $L_k$ ;

**for each** transaction  $t$  in database do

increment the count of all candidates in  $C_{k+1}$

that are contained in  $t$

$L_{k+1} =$  candidates in  $C_{k+1}$  with min\_support

**end**

**return**  $\cup_k L_k$ ;

# Important Details of Apriori

---

- How to generate candidates?
  - Step 1: self-joining  $L_k$
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - $L_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining:  $L_3 * L_3$ 
    - $abcd$  from  $abc$  and  $abd$
    - $acde$  from  $acd$  and  $ace$
  - Pruning:
    - $acde$  is removed because  $ade$  is not in  $L_3$
  - $C_4 = \{abcd\}$

# How to Generate Candidates?

---

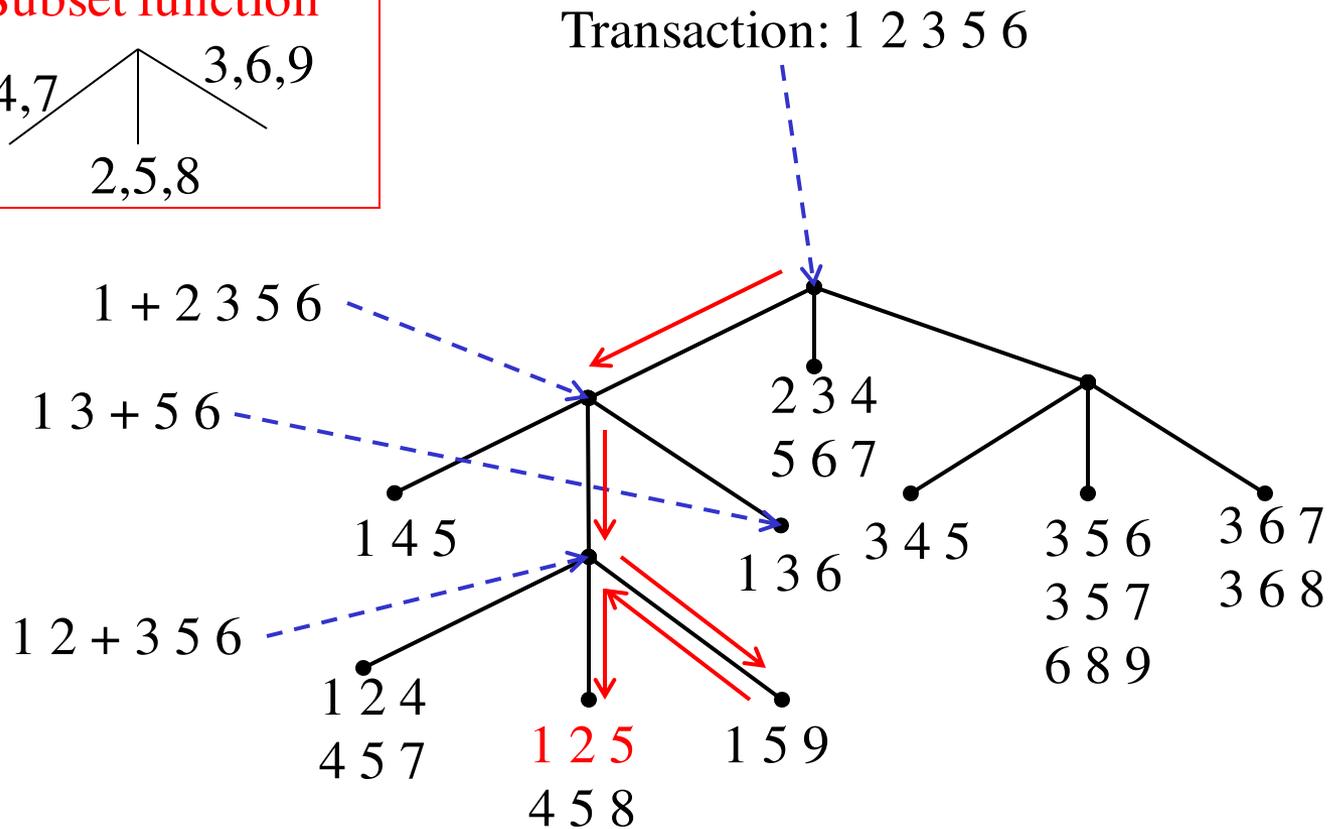
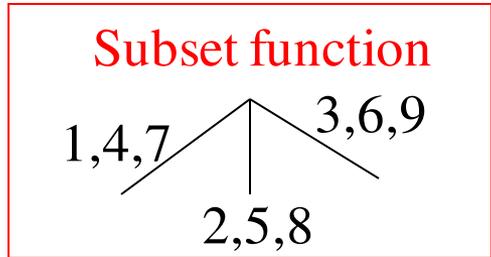
- Suppose the items in  $L_{k-1}$  are listed in an order
- Step 1: self-joining  $L_{k-1}$ 
  - insert into  $C_k$
  - select  **$p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$**
  - from  **$L_{k-1} p, L_{k-1} q$**
  - where  **$p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} <$**   
 **$q.item_{k-1}$**
- Step 2: pruning
  - forall ***itemsets*  $c$  in  $C_k$**  do
    - forall ***(k-1)-subsets*  $s$  of  $c$**  do
      - if ( $s$  is not in  $L_{k-1}$ ) then delete  $c$  from  $C_k$**

# How to Count Supports of Candidates?

---

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf node* of hash-tree contains a list of itemsets and counts
  - *Interior node* contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

# Example: Counting Supports of Candidates



# Efficient Implementation of Apriori in SQL

---

- Hard to get good performance out of pure SQL (SQL-92) based approaches alone
- Make use of object-relational extensions like UDFs, BLOBs, Table functions etc.
  - Get orders of magnitude improvement
- S. Sarawagi, S. Thomas, and R. Agrawal. [Integrating association rule mining with relational database systems: Alternatives and implications](#). In SIGMOD'98

# Challenges of Frequent Pattern Mining

---

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

# Partition: Scan Database Only Twice

---

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. *An efficient algorithm for mining association in large databases*. In *VLDB'95*

# DHP: Reduce the Number of Candidates

---

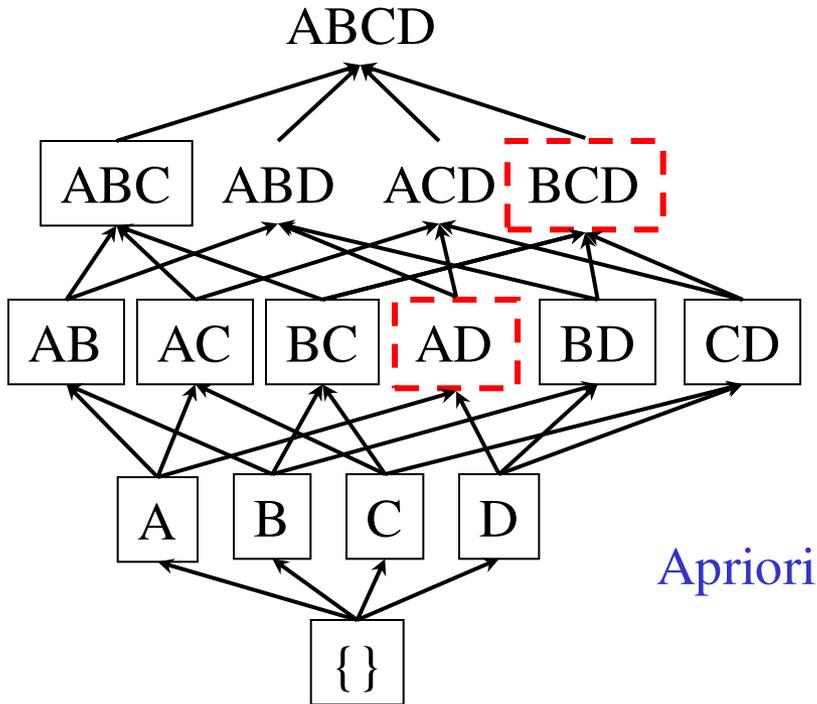
- A  $k$ -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries: {ab, ad, ae} {bd, be, de} ...
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold
- J. Park, M. Chen, and P. Yu. *An effective hash-based algorithm for mining association rules*. In *SIGMOD'95*

# Sampling for Frequent Patterns

---

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, ..., etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. *Sampling large databases for association rules*. In *VLDB'96*

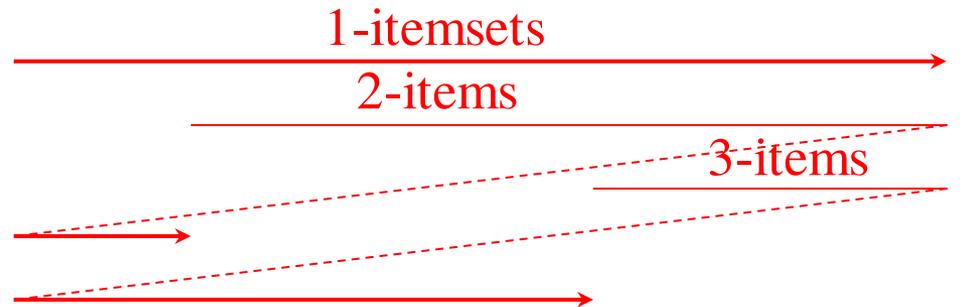
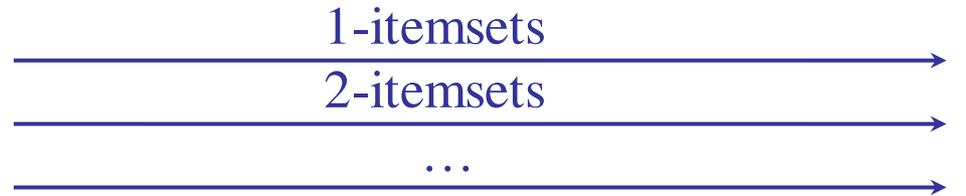
# DIC: Reduce Number of Scans



Itemset lattice

S. Brin R. Motwani, J. Ullman, and S. Tsur. *Dynamic itemset counting and implication rules for market basket data*. In *SIGMOD'97*

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



DIC

# Bottleneck of Frequent-pattern Mining

---

- Multiple database scans are **costly**
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset  $i_1 i_2 \dots i_{100}$ 
    - # of scans: **100**
    - # of Candidates:  $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = \mathbf{1.27 * 10^{30} !}$
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

# Mining Frequent Patterns Without Candidate Generation

---

- Grow long patterns from short ones using local frequent items
  - “abc” is a frequent pattern
  - Get all transactions having “abc”: DB|abc
  - “d” is a local frequent item in DB|abc → abcd is a frequent pattern

# Construct FP-tree from a Transaction Database

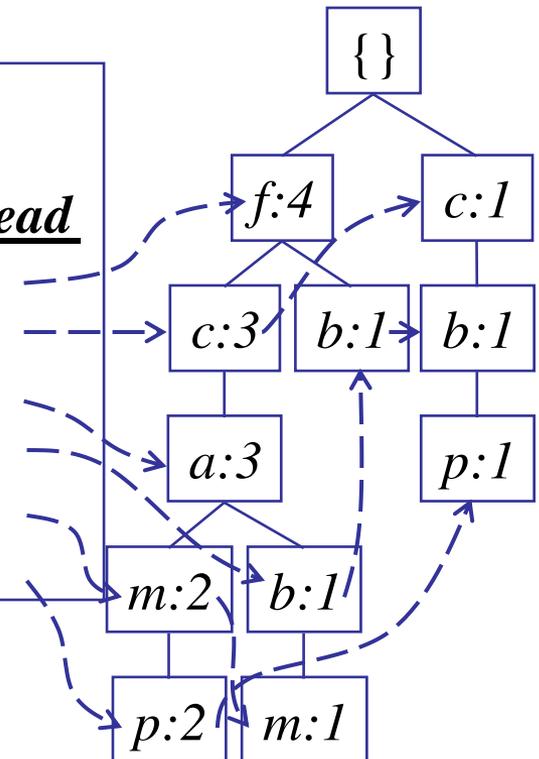
<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

*min\_support = 3*

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

<b>Header Table</b>	
<u><i>Item frequency head</i></u>	
<i>f</i>	4
<i>c</i>	4
<i>a</i>	3
<i>b</i>	3
<i>m</i>	3
<i>p</i>	3

**F-list**=f-c-a-b-m-p



# Benefits of the FP-tree Structure

---

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - For Connect-4 DB, compression ratio could be over 100

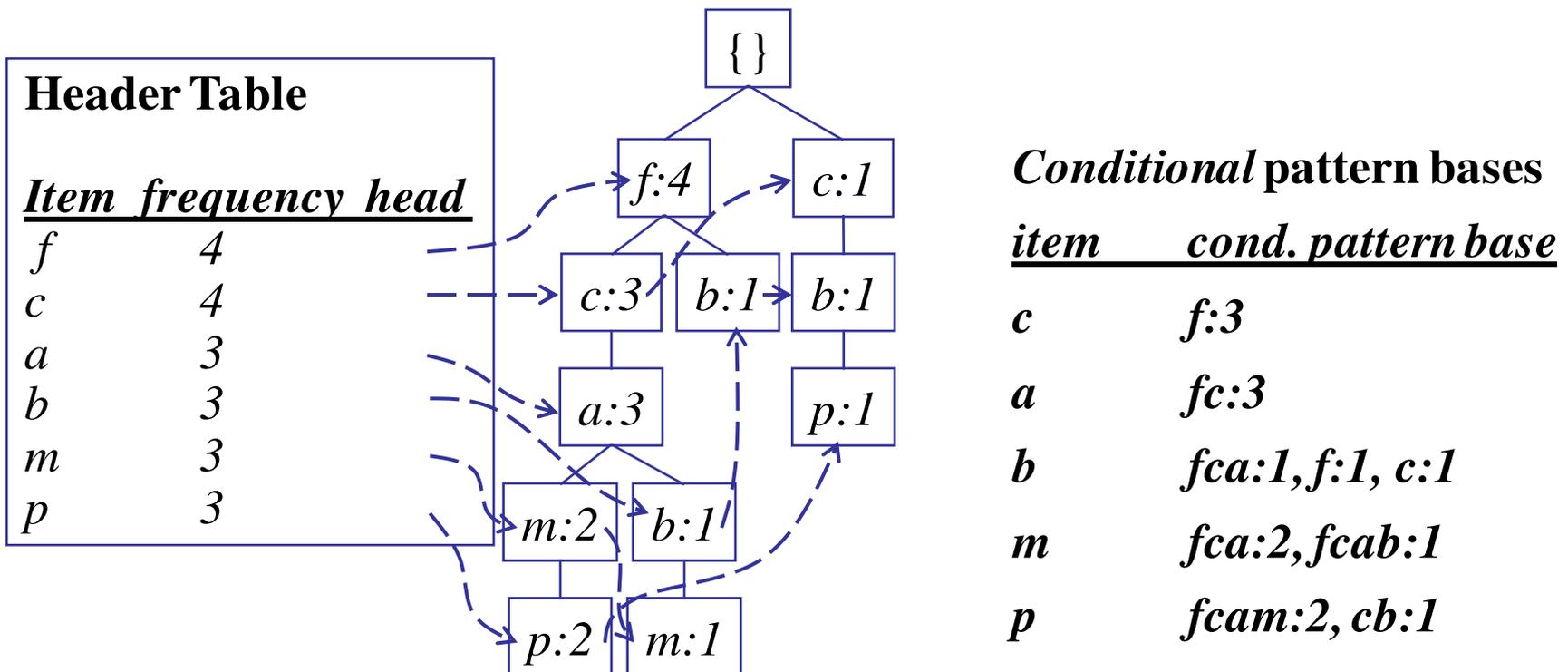
# Partition Patterns and Databases

---

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list=f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - ...
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundancy

# Find Patterns Having P From P-conditional Database

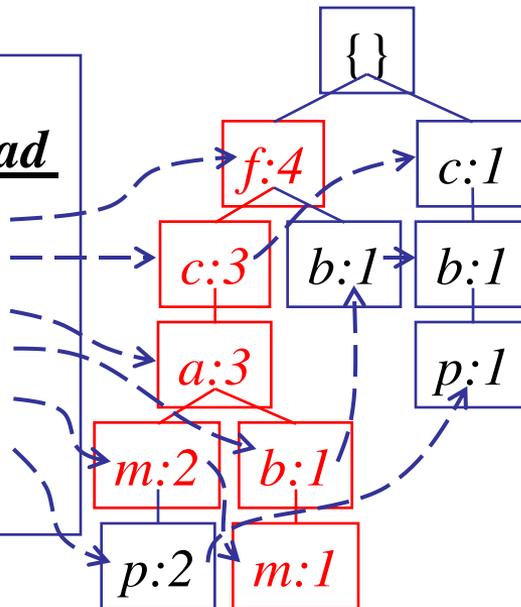
- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item  $p$
- Accumulate all of *transformed prefix paths* of item  $p$  to form  $p$ 's conditional pattern base



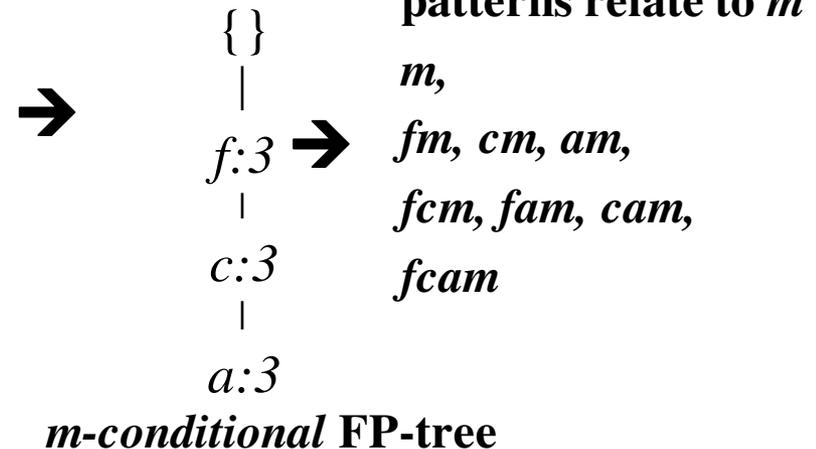
# From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

Header Table	
<u>Item frequency head</u>	
<i>f</i>	4
<i>c</i>	4
<i>a</i>	3
<i>b</i>	3
<i>m</i>	3
<i>p</i>	3

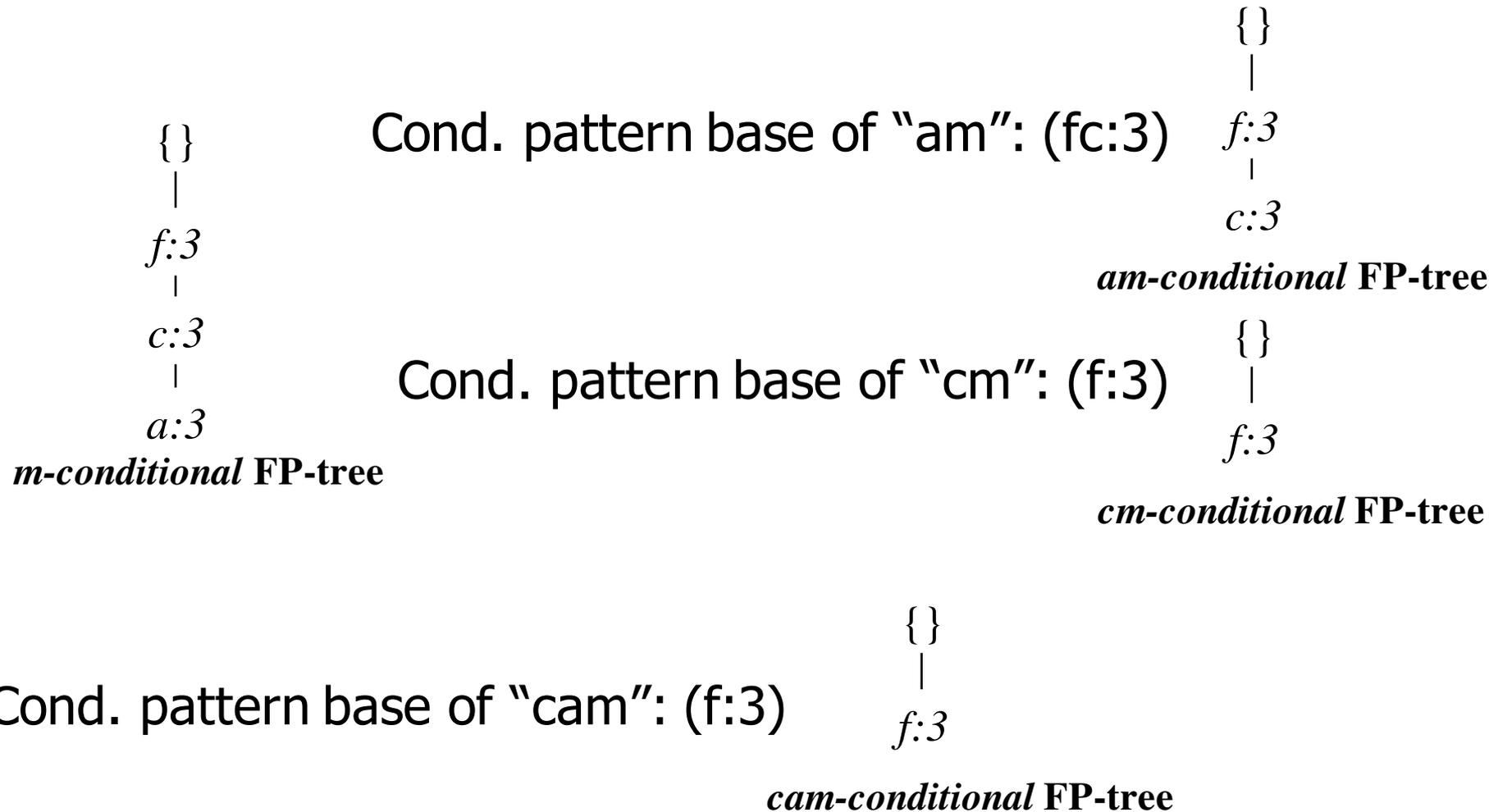


*m*-conditional pattern base:  
*fca:2, fcab:1*



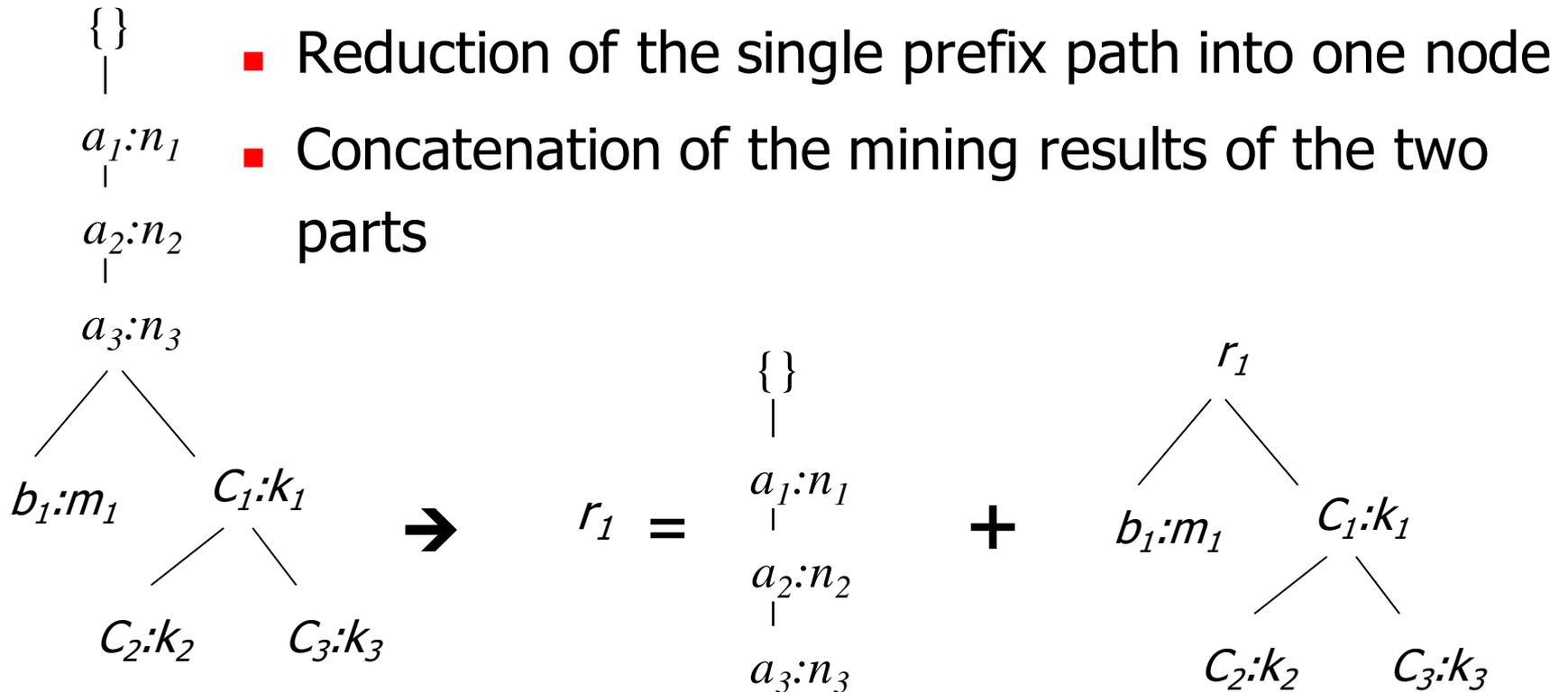
# Recursion: Mining Each Conditional FP-tree

---



# A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts



# Mining Frequent Patterns With FP-trees

---

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

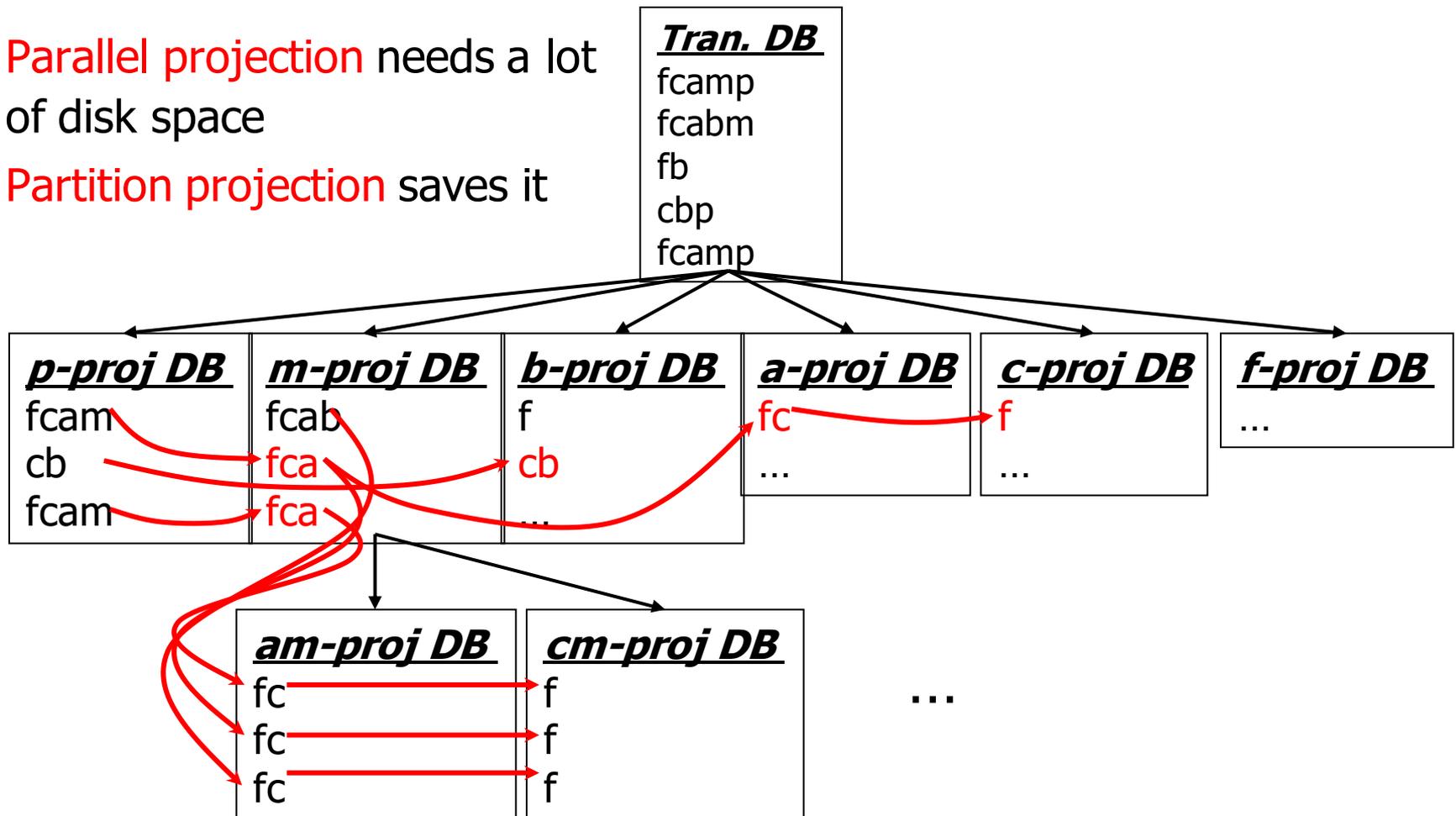
# Scaling FP-growth by DB Projection

---

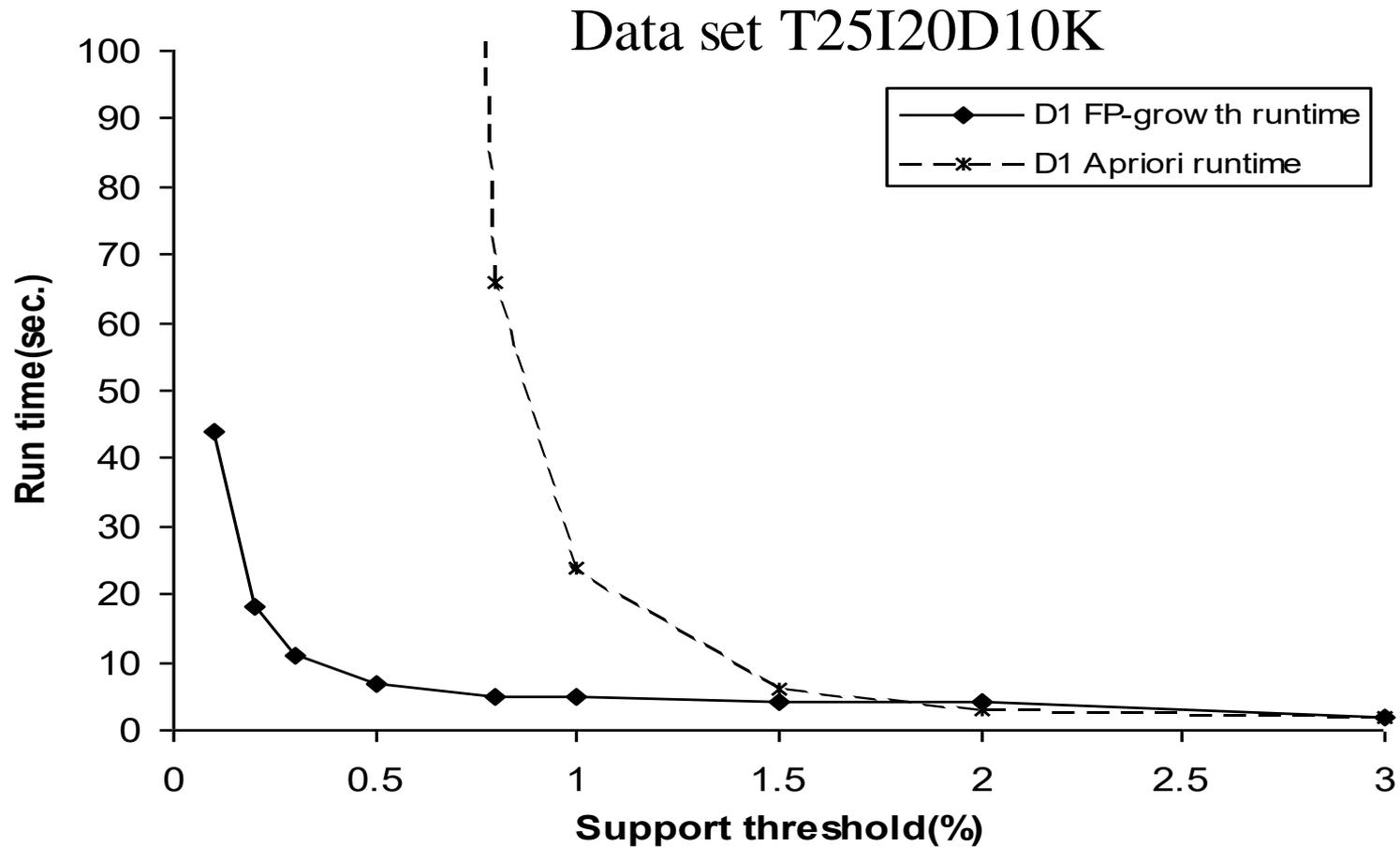
- FP-tree cannot fit in memory?—DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- **Parallel projection** vs. **Partition projection** techniques
  - Parallel projection is space costly

# Partition-based Projection

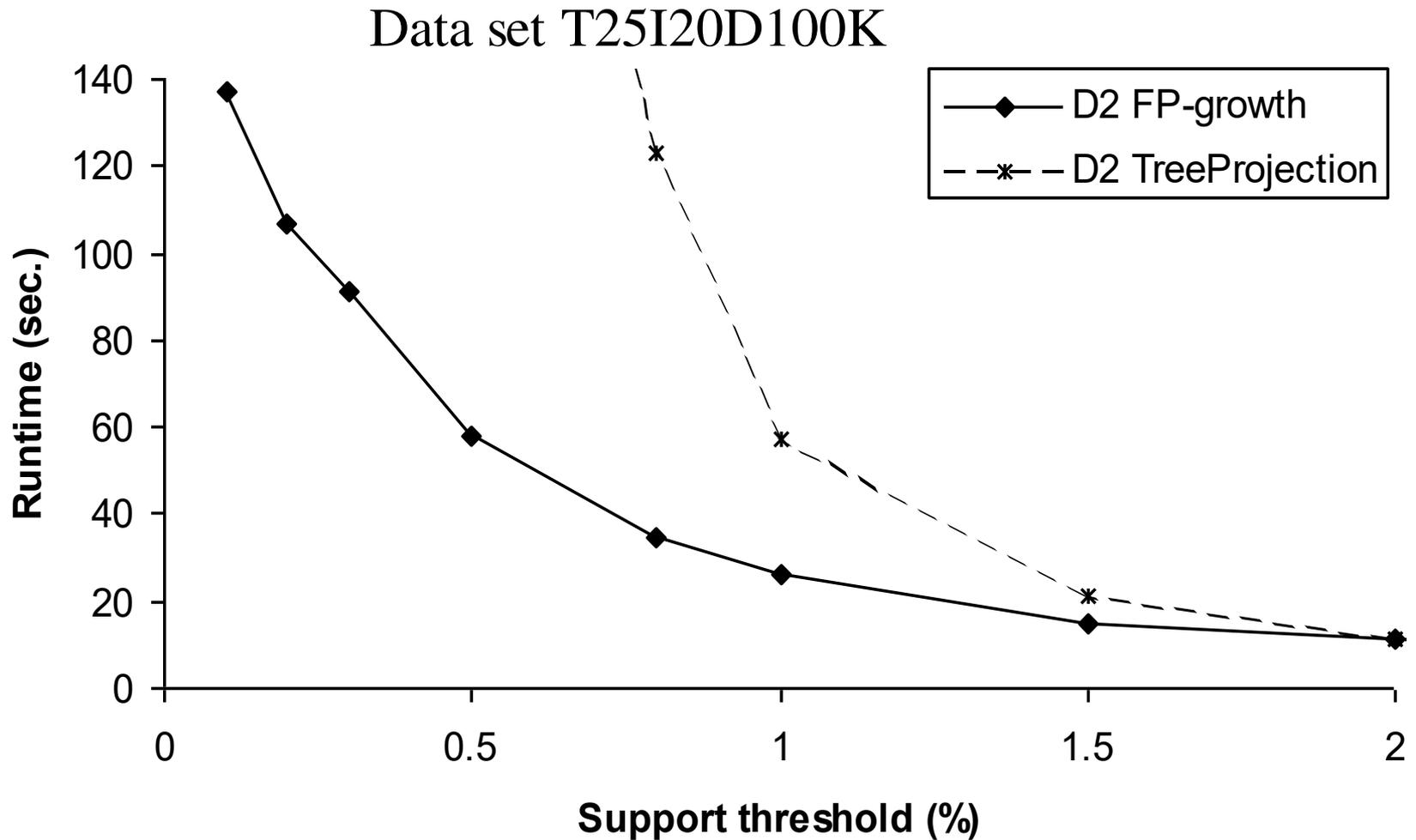
- **Parallel projection** needs a lot of disk space
- **Partition projection** saves it



# FP-Growth vs. Apriori: Scalability With the Support Threshold



# FP-Growth vs. Tree-Projection: Scalability with the Support Threshold



# Why Is FP-Growth the Winner?

---

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases
- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

# Implications of the Methodology

---

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD'00)
- Mining sequential patterns
  - FreeSpan (KDD'00), PrefixSpan (ICDE'01)
- Constraint-based mining of frequent patterns
  - Convertible constraints (KDD'00, ICDE'01)
- Computing iceberg data cubes with complex measures
  - H-tree and H-cubing algorithm (SIGMOD'01)

# MaxMiner: Mining Max-patterns

- 1<sup>st</sup> scan: find frequent items

- A, B, C, D, E

- 2<sup>nd</sup> scan: find support for

- AB, AC, AD, AE, **ABCDE**

- BC, BD, BE, **BCDE**

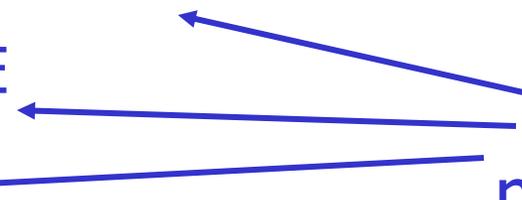
- CD, CE, **CDE**, ~~DE~~

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

- R. Bayardo. **Efficiently mining long patterns from databases**. In *SIGMOD'98*

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F

Potential  
max-patterns



# Mining Frequent Closed Patterns: CLOSET

- Flist: list of all frequent items in support ascending order

- Flist: d-a-f-e-c

- Divide search space

- Patterns having d

- Patterns having d but no a, etc.

- Find frequent closed pattern recursively

- Every transaction having d also has cfa → cfad is a frequent closed pattern

- J. Pei, J. Han & R. Mao. "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00.

Min\_sup=2

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f

# CLOSET+: Mining Closed Itemsets by Pattern-Growth

---

- Itemset merging: if  $Y$  appears in every occurrence of  $X$ , then  $Y$  is merged with  $X$
- Sub-itemset pruning: if  $Y \supset X$ , and  $\text{sup}(X) = \text{sup}(Y)$ ,  $X$  and all of  $X$ 's descendants in the set enumeration tree can be pruned
- Hybrid tree projection
  - Bottom-up physical tree-projection
  - Top-down pseudo tree-projection
- Item skipping: if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
- Efficient subset checking

# CHARM: Mining by Exploring Vertical Data Format

---

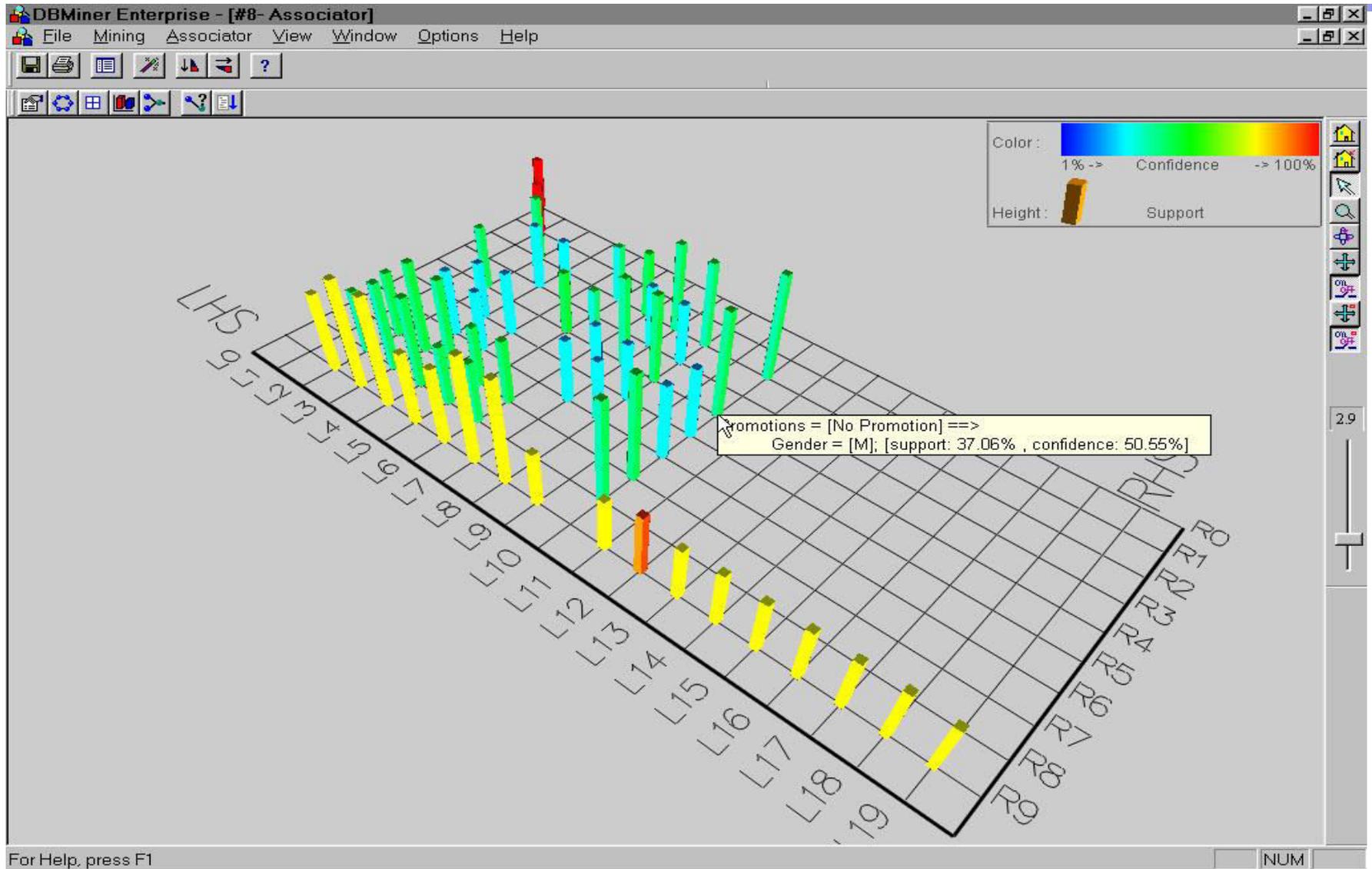
- Vertical format:  $t(AB) = \{T_{11}, T_{25}, \dots\}$ 
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - $t(X) = t(Y)$ : X and Y always happen together
  - $t(X) \subset t(Y)$ : transaction having X always has Y
- Using **diffset** to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$ ,  $t(XY) = \{T_1, T_3\}$
  - Diffset  $(XY, X) = \{T_2\}$
- Eclat/MaxEclat (Zaki et al. @KDD'97), VIPER(P. Shenoy et al.@SIGMOD'00), CHARM (Zaki & Hsiao@SDM'02)

# Further Improvements of Mining Methods

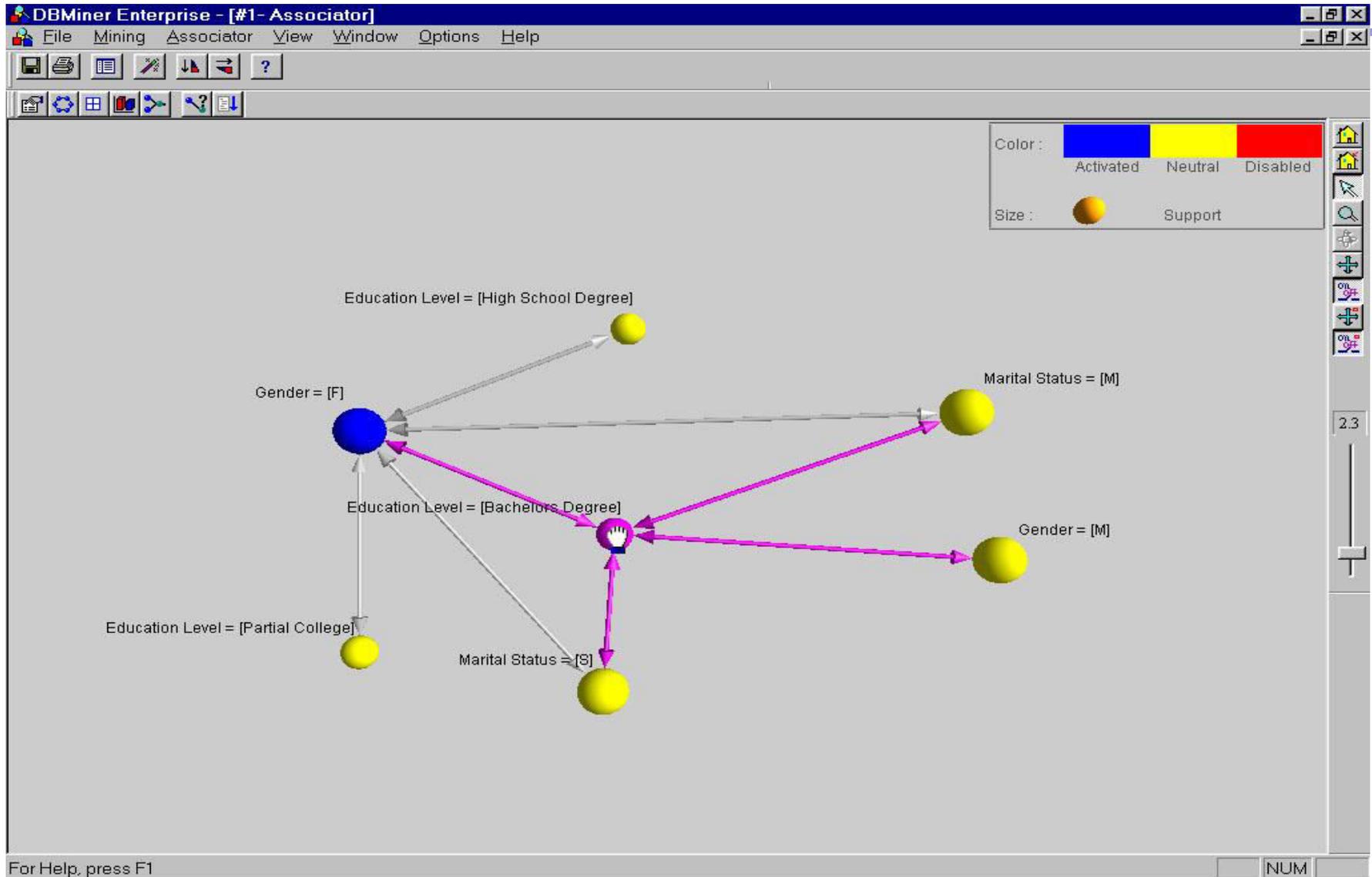
---

- AFOPT
  - A “push-right” method for mining condensed frequent pattern (CFP) tree
- Carpenter
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining

# Visualization of Association Rules: Plane Graph



# Visualization of Association Rules: Rule Graph





# Mining Various Kinds of Association Rules

---

- Mining multilevel association
- Mining multidimensional association
- Mining quantitative association
- Mining interesting correlation patterns

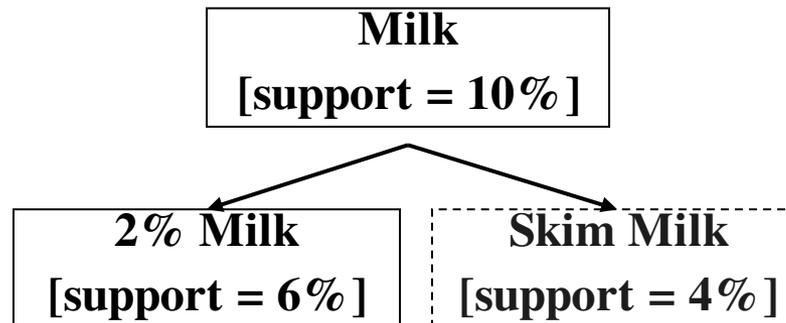
# Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB'95, Han & Fu@VLDB'95)

uniform support

Level 1  
min\_sup = 5%

Level 2  
min\_sup = 5%



reduced support

Level 1  
min\_sup = 5%

Level 2  
min\_sup = 3%

# Multi-level Association: Redundancy Filtering

---

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk  $\Rightarrow$  wheat bread [support = 8%, confidence = 70%]
  - 2% milk  $\Rightarrow$  wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.

# Mining Multi-Dimensional Association

---

- Single-dimensional rules:

$\text{buys}(X, \text{"milk"}) \Rightarrow \text{buys}(X, \text{"bread"})$

- Multi-dimensional rules:  $\geq 2$  dimensions or predicates

- Inter-dimension assoc. rules (*no repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{occupation}(X, \text{"student"}) \Rightarrow \text{buys}(X, \text{"coke"})$

- hybrid-dimension assoc. rules (*repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{buys}(X, \text{"popcorn"}) \Rightarrow \text{buys}(X, \text{"coke"})$

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: numeric, implicit ordering among values—discretization, clustering, and gradient approaches

# Mining Quantitative Associations

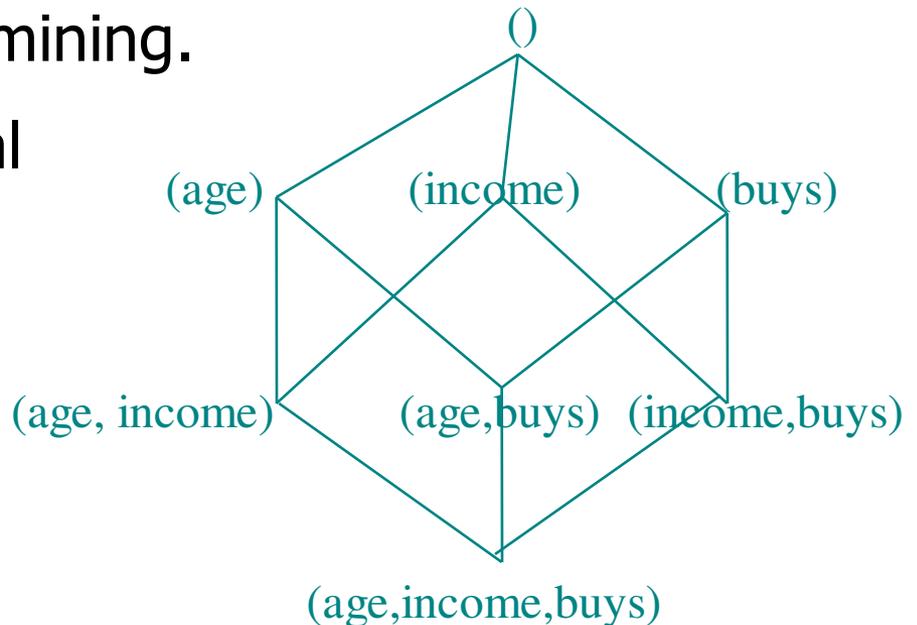
---

- Techniques can be categorized by how numerical attributes, such as **age** or **salary** are treated
  1. Static discretization based on predefined concept hierarchies (data cube methods)
  2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
  3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
    - one dimensional clustering then association
  4. Deviation: (such as Aumann and Lindell@KDD99)  
Sex = female => Wage: mean=\$7/hr (overall mean = \$9)

# Static Discretization of Quantitative Attributes

---

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent  $k$ -predicate sets will require  $k$  or  $k+1$  table scans.
- Data cube is well suited for mining.
- The cells of an  $n$ -dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.





# Mining Other Interesting Patterns

---

- Flexible support constraints
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized  $\text{sup}_{\min}$  specification and application
- Top-K closed frequent patterns
  - Hard to specify  $\text{sup}_{\min}$ , but top-k with  $\text{length}_{\min}$  is more desirable
  - Dynamically raise  $\text{sup}_{\min}$  in FP-tree construction and mining, and select most promising path to mine

# Interestingness Measure: Correlations (Lift)

- *play basketball*  $\Rightarrow$  *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball*  $\Rightarrow$  *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: **lift**

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

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B, C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89 \quad lift(B, \neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

# Are *lift* and $\chi^2$ Good Measures of Correlation?

- "*Buy walnuts*  $\Rightarrow$  *buy milk* [1%, 80%]" is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD'02)

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

$$all\_conf = \frac{sup(X)}{\max\_item\_sup(X)}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	$\sim m, c$	c
No Coffee	m, $\sim c$	$\sim m, \sim c$	$\sim c$
Sum(col.)	m	$\sim m$	$\Sigma$

$$coh = \frac{sup(X)}{|universe(X)|}$$

DB	m, c	$\sim m, c$	m $\sim c$	$\sim m\sim c$	lift	all-conf	coh	$\chi^2$
A1	1000	100	100	10,000	9.26	0.91	0.83	9055
A2	100	1000	1000	100,000	8.44	0.09	0.05	670
A3	1000	100	10000	100,000	9.18	0.09	0.09	8172
A4	1000	1000	1000	1000	1	0.5	0.33	0

# Which Measures Should Be Used?

- **lift** and  $\chi^2$  are not good measures for correlations in large transactional DBs
- **all-conf** or **coherence** could be good measures (Omiecinski@TKDE'03)
- Both **all-conf** and **coherence** have the downward closure property
- Efficient algorithms can be derived for mining (Lee et al. @ICDM'03sub)

symbol	measure	range	formula
$\phi$	$\phi$ -coefficient	-1 ... 1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
$Q$	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},\bar{B}) + P(A,\bar{B})P(\bar{A},B)}$
$Y$	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
$k$	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
$PS$	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A, B) - P(A)P(B)$
$F$	Certainty factor	-1 ... 1	$\max\left(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)}\right)$
$AV$	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
$K$	Klosgen's Q	-0.33 ... 0.38	$\sqrt{P(A, B)} \max(P(B A) - P(B), P(A B) - P(A))$
$g$	Goodman-kruskal's	0 ... 1	$\frac{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}{2 - \max_j P(A_j) - \max_k P(B_k)}$
$M$	Mutual Information	0 ... 1	$\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i) \log P(A_i) \log P(A_i), -\sum_i P(B_i) \log P(B_i) \log P(B_i))}$
$J$	J-Measure	0 ... 1	$\max(P(A, B) \log\left(\frac{P(B A)}{P(B)}\right) + P(\bar{A}\bar{B}) \log\left(\frac{P(\bar{B} \bar{A})}{P(\bar{B})}\right), P(A, B) \log\left(\frac{P(A B)}{P(A)}\right) + P(\bar{A}\bar{B}) \log\left(\frac{P(\bar{A} \bar{B})}{P(\bar{A})}\right))$
$G$	Gini index	0 ... 1	$\max(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2, P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2)$
$s$	support	0 ... 1	$P(A, B)$
$c$	confidence	0 ... 1	$\max(P(B A), P(A B))$
$L$	Laplace	0 ... 1	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
$IS$	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
$\gamma$	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
$\alpha$	all_confidence	0 ... 1	$\frac{\max(P(A), P(B))}{P(A,B)}$
$o$	odds ratio	0 ... $\infty$	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(A,\bar{B})}$
$V$	Conviction	0.5 ... $\infty$	$\max\left(\frac{P(A)P(\bar{B})}{P(\bar{A}\bar{B})}, \frac{P(B)P(\bar{A})}{P(\bar{B}\bar{A})}\right)$
$\lambda$	lift	0 ... $\infty$	$\frac{P(A,B)}{P(A)P(B)}$
$S$	Collective strength	0 ... $\infty$	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$
$\chi^2$	$\chi^2$	0 ... $\infty$	$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$

# Constraint-based (Query-Directed) Mining

---

- Finding **all** the patterns in a database **autonomously**? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an **interactive** process
  - User directs what to be mined using a **data mining query language** (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides **constraints** on what to be mined
  - System optimization: explores such constraints for efficient mining—**constraint-based mining**

# Constraints in Data Mining

---

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.'02
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < \$10) triggers big sales (sum > \$200)
- Interestingness constraint
  - strong rules:  $\text{min\_support} \geq 3\%$ ,  $\text{min\_confidence} \geq 60\%$

# Constrained Mining vs. Constraint-Based Search

---

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding **all patterns** satisfying constraints vs. finding **some (or one) answer** in constraint-based search in AI
  - **Constraint-pushing** vs. **heuristic search**
  - It is an interesting research problem on how to integrate them
- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

# Anti-Monotonicity in Constraint Pushing

TDB (min\_sup=2)

- Anti-monotonicity
  - *When an itemset  $S$  **violates** the constraint, so does any of its superset*
  - $sum(S.Price) \leq v$  is **anti-monotone**
  - $sum(S.Price) \geq v$  is **not anti-monotone**
- Example. C:  $range(S.profit) \leq 15$  is **anti-monotone**
  - Itemset  $ab$  violates C
  - So does every superset of  $ab$

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

# Monotonicity for Constraint Pushing

TDB (min\_sup=2)

- Monotonicity
  - *When an itemset  $S$  **satisfies** the constraint, so does any of its superset*
  - $sum(S.Price) \geq v$  is **monotone**
  - $min(S.Price) \leq v$  is **monotone**
- Example. C:  $range(S.profit) \geq 15$ 
  - Itemset  $ab$  satisfies C
  - So does every superset of  $ab$

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

# Succinctness

---

- Succinctness:
  - Given  $A_1$ , the set of items satisfying a succinctness constraint  $C$ , then any set  $S$  satisfying  $C$  is based on  $A_1$ , i.e.,  $S$  contains a subset belonging to  $A_1$
  - Idea: Without looking at the transaction database, whether an itemset  $S$  satisfies constraint  $C$  can be determined based on the selection of items
  - $\min(S.Price) \leq v$  is succinct
  - $\sum(S.Price) \geq v$  is not succinct
- Optimization: If  $C$  is succinct,  $C$  is pre-counting pushable

# The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

$L_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

$C_2$

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

$C_2$

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

$L_2$

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

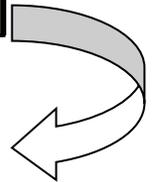
$C_3$

itemset
{2 3 5}

Scan D

$L_3$

itemset	sup
{2 3 5}	2



# Naïve Algorithm: Apriori + Constraint

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

$L_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
<del>{5}</del>	<del>3</del>

$C_2$

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

$C_2$

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

$L_2$

itemset	sup
{1 3}	2
<del>{2 3}</del>	<del>2</del>
<del>{2 5}</del>	<del>3</del>
<del>{3 5}</del>	<del>2</del>

$C_3$

itemset
{2 3 5}

Scan D

$L_3$

itemset	sup
<del>{2 3 5}</del>	<del>2</del>

**Constraint:**  
**Sum{S.price} < 5**

# The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
<del>{5}</del>	<del>3</del>

$C_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
<del>{5}</del>	<del>3</del>

$L_1$

itemset	sup
{1 3}	2
<del>{2 3}</del>	<del>2</del>
<del>{2 5}</del>	<del>3</del>
<del>{3 5}</del>	<del>2</del>

$L_2$

itemset	sup
{1 2}	1
{1 3}	2
<del>{1 5}</del>	<del>1</del>
<del>{2 3}</del>	<del>2</del>
<del>{2 5}</del>	<del>3</del>
<del>{3 5}</del>	<del>2</del>

$C_2$

itemset
{1 2}
{1 3}
<del>{1 5}</del>
{2 3}
<del>{2 5}</del>
<del>{3 5}</del>

$C_2$

itemset
<del>{2 3 5}</del>

$C_3$

itemset	sup
<del>{2 3 5}</del>	<del>2</del>

$L_3$

**Constraint:**  
**Sum{S.price} < 5**

# The Constrained Apriori Algorithm: Push a Succinct Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

$C_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

Scan D

$L_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

$C_2$

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
<del>{2 3}</del>	<del>2</del>
<del>{2 5}</del>	<del>3</del>
<del>{3 5}</del>	<del>2</del>

Scan D

$C_2$

itemset
{1 2}
{1 3}
{1 5}
<del>{2 3}</del>
<del>{2 5}</del>
<del>{3 5}</del>

not immediately to be used

$L_2$

itemset	sup
{1 3}	2
<del>{2 3}</del>	<del>2</del>
<del>{2 5}</del>	<del>3</del>
<del>{3 5}</del>	<del>2</del>

$C_3$

itemset
<del>{2 3 5}</del>

$L_3$

itemset	sup
<del>{2 3 5}</del>	<del>2</del>

**Constraint:**  
 $\min\{S.price\} \leq 1$

# Converting “Tough” Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C:  $\text{avg}(S.\text{profit}) \geq 25$ 
  - Order items in value-descending order
    - $\langle a, f, g, d, b, h, c, e \rangle$
  - If an itemset  $afb$  violates C
    - So does  $afbh, afb^*$
    - It becomes **anti-monotone!**

TDB (min\_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

# Strongly Convertible Constraints

- $\text{avg}(X) \geq 25$  is convertible anti-monotone w.r.t. item **value descending** order  $R$ :  $\langle a, f, g, d, b, h, c, e \rangle$ 
  - If an itemset  $af$  violates a constraint  $C$ , so does every itemset with  $af$  as prefix, such as  $afd$
- $\text{avg}(X) \geq 25$  is convertible monotone w.r.t. item **value ascending** order  $R^{-1}$ :  $\langle e, c, h, b, d, g, f, a \rangle$ 
  - If an itemset  $d$  satisfies a constraint  $C$ , so does itemsets  $df$  and  $dfa$ , which having  $d$  as a prefix
- Thus,  $\text{avg}(X) \geq 25$  is **strongly convertible**

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

# Can Apriori Handle Convertible Constraint?

---

- A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset  $df$  violates constraint  $C: \text{avg}(X) \geq 25$
  - Since  $adf$  satisfies  $C$ , Apriori needs  $df$  to assemble  $adf$ ,  $df$  cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

Item	Value
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

# Mining With Convertible Constraints

- C:  $\text{avg}(X) \geq 25$ ,  $\text{min\_sup}=2$
- List items in every transaction in value descending order R:  $\langle a, f, g, d, b, h, c, e \rangle$ 
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, ...
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

Item	Value
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

TDB ( $\text{min\_sup}=2$ )

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

# Handling Multiple Constraints

---

- Different constraints may require different or even conflicting item-ordering
- If there exists an order  $R$  s.t. both  $C_1$  and  $C_2$  are convertible w.r.t.  $R$ , then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

# What Constraints Are Convertible?

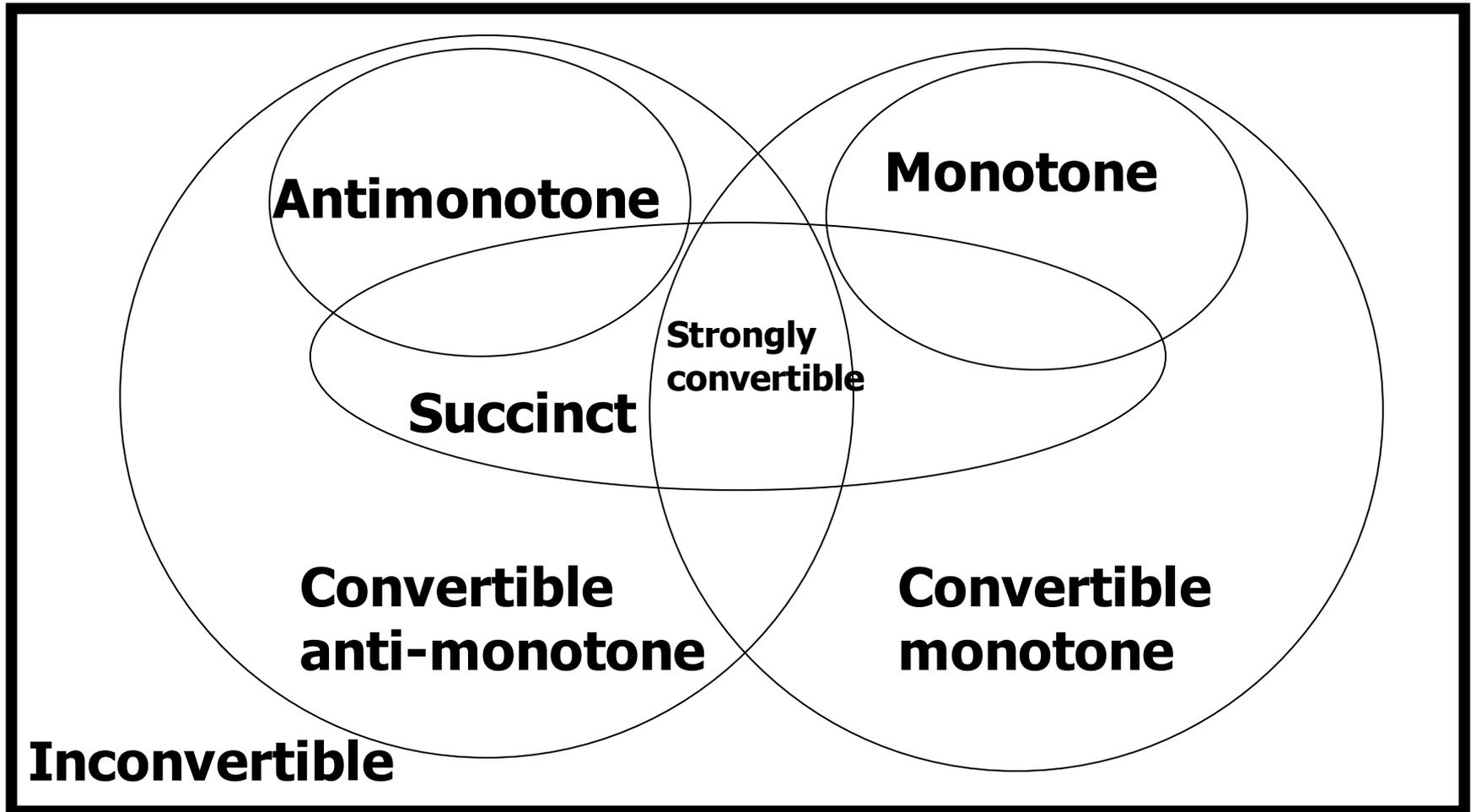
Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
$\text{avg}(S) \leq, \geq v$	Yes	Yes	Yes
$\text{median}(S) \leq, \geq v$	Yes	Yes	Yes
$\text{sum}(S) \leq v$ (items could be of any value, $v \geq 0$ )	Yes	No	No
$\text{sum}(S) \leq v$ (items could be of any value, $v \leq 0$ )	No	Yes	No
$\text{sum}(S) \geq v$ (items could be of any value, $v \geq 0$ )	No	Yes	No
$\text{sum}(S) \geq v$ (items could be of any value, $v \leq 0$ )	Yes	No	No
.....			

# Constraint-Based Mining—A General Picture

Constraint	Antimonotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$\text{count}(S) \leq v$	yes	no	weakly
$\text{count}(S) \geq v$	no	yes	weakly
$\text{sum}(S) \leq v \ (a \in S, a \geq 0)$	yes	no	no
$\text{sum}(S) \geq v \ (a \in S, a \geq 0)$	no	yes	no
$\text{range}(S) \leq v$	yes	no	no
$\text{range}(S) \geq v$	no	yes	no
$\text{avg}(S) \theta v, \theta \in \{=, \leq, \geq\}$	convertible	convertible	no
$\text{support}(S) \geq \xi$	yes	no	no
$\text{support}(S) \leq \xi$	no	yes	no

# A Classification of Constraints

---



# Classification vs. Prediction

---

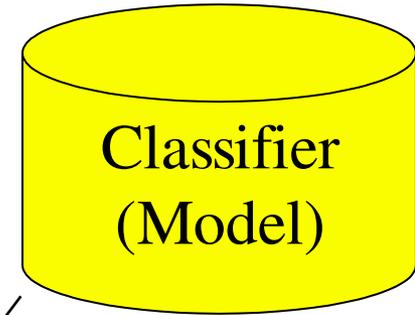
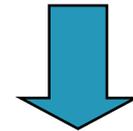
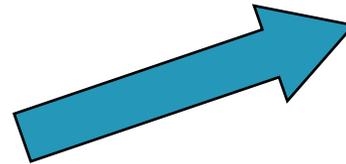
- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (**class labels**) in a classifying attribute and uses it in classifying new data
- **Prediction**
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

# Classification—A Two-Step Process

---

- **Model construction**: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
  - The set of tuples used for model construction is **training set**
  - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage**: for classifying future or unknown objects
  - **Estimate accuracy** of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known

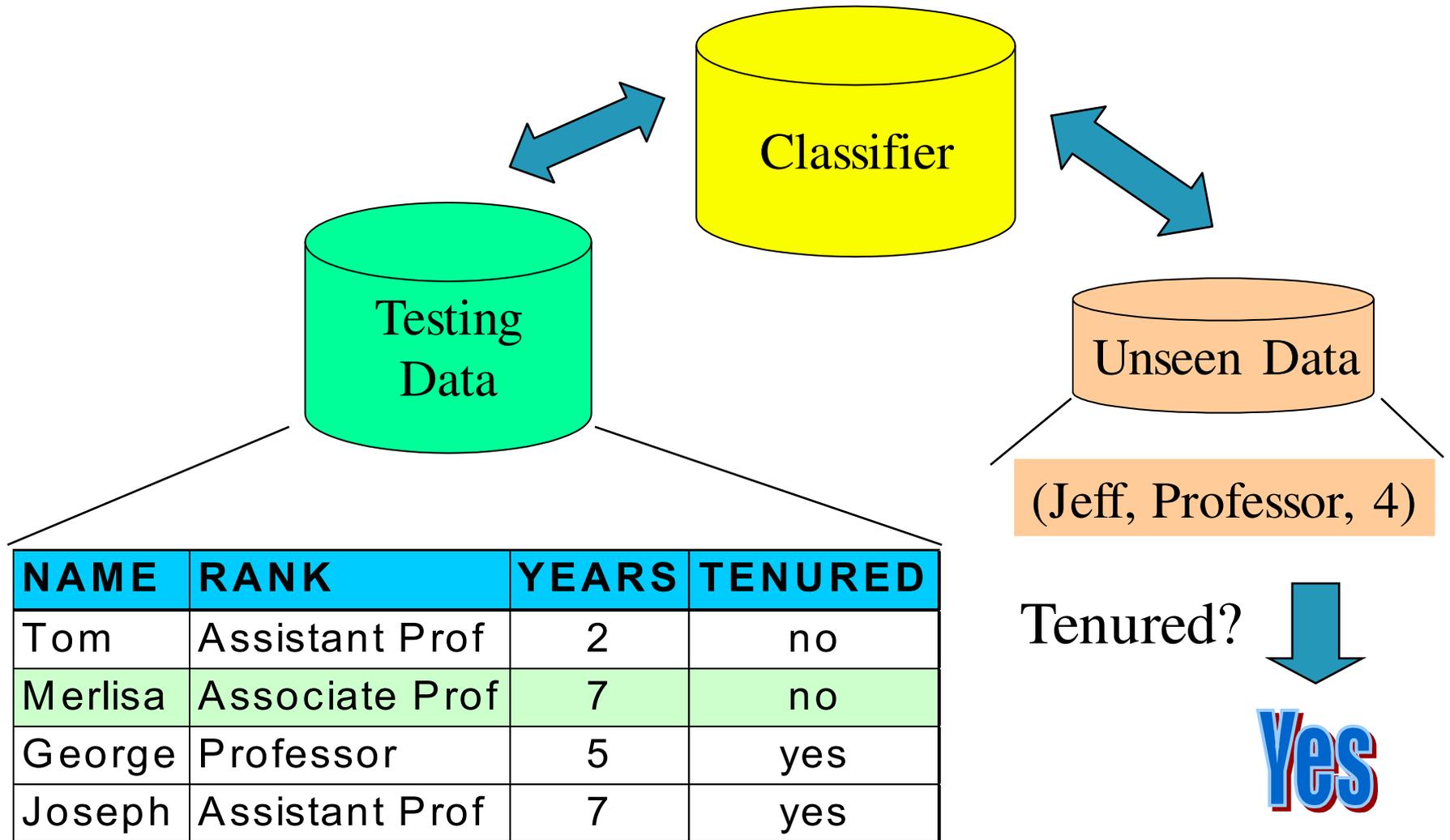
# Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor'  
OR years > 6  
THEN tenured = 'yes'

# Process (2): Using the Model in Prediction



# Supervised vs. Unsupervised Learning

---

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

# Issues: Data Preparation

---

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

# Issues: Evaluating Classification Methods

---

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

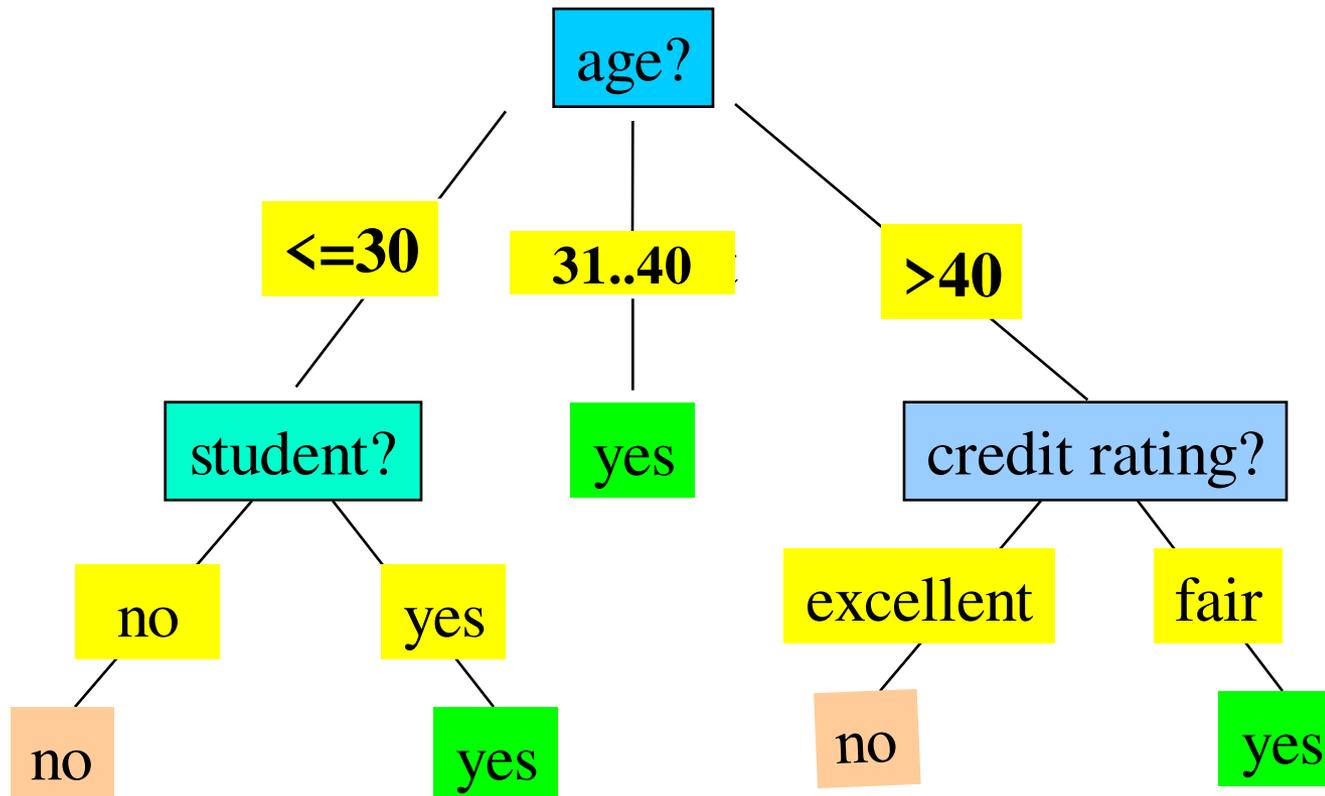
# Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

This follows an example of Quinlan's ID3 (Playing Tennis)

# Output: A Decision Tree for "*buys\_computer*"

---



# Algorithm for Decision Tree Induction

---

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a **top-down recursive divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
  - There are no samples left

# Attribute Selection Measure: Information Gain (ID3/C4.5)

---

- Select the attribute with the highest information gain
- Let  $p_i$  be the probability that an arbitrary tuple in  $D$  belongs to class  $C_i$ , estimated by  $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in  $D$ :

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

- **Information** needed (after using  $A$  to split  $D$  into  $v$  partitions) to classify  $D$ :

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

- **Information gained** by branching on attribute  $A$

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

# Attribute Selection: Information Gain

- Class P: buys\_computer = "yes"

- Class N: buys\_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

age	$p_i$	$n_i$	$I(p_i, n_i)$
$\leq 30$	2	3	0.971
31...40	4	0	0
$> 40$	3	2	0.971

$\frac{5}{14} I(2,3)$  means "age  $\leq 30$ " has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	high	no	excellent	no
31...40	high	no	fair	yes
$> 40$	medium	no	fair	yes
$> 40$	low	yes	fair	yes
$> 40$	low	yes	excellent	no
31...40	low	yes	excellent	yes
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$> 40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
$> 40$	medium	no	excellent	no

# Computing Information-Gain for Continuous-Value Attributes

---

- Let attribute  $A$  be a continuous-valued attribute
- Must determine the *best split point* for  $A$ 
  - Sort the value  $A$  in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i + a_{i+1})/2$  is the midpoint between the values of  $a_i$  and  $a_{i+1}$
  - The point with the *minimum expected information requirement* for  $A$  is selected as the split-point for  $A$
- Split:
  - $D_1$  is the set of tuples in  $D$  satisfying  $A \leq \text{split-point}$ , and  $D_2$  is the set of tuples in  $D$  satisfying  $A > \text{split-point}$

# Gain Ratio for Attribute Selection (C4.5)

---

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $\text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}(A)$
- Ex.  $\text{SplitInfo}_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$ 
  - $\text{gain\_ratio}(\text{income}) = 0.029/0.926 = 0.031$
- The attribute with the maximum gain ratio is selected as the splitting attribute

# Gini index (CART, IBM IntelligentMiner)

- If a data set  $D$  contains examples from  $n$  classes, gini index,  $gini(D)$  is defined as

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

where  $p_j$  is the relative frequency of class  $j$  in  $D$

- If a data set  $D$  is split on  $A$  into two subsets  $D_1$  and  $D_2$ , the  $gini$  index  $gini_A(D)$  is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest  $gini_{split}(D)$  (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

# Gini index (CART, IBM IntelligentMiner)

- Ex. D has 9 tuples in buys\_computer = "yes" and 5 in "no"

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

- Suppose the attribute income partitions D into 10 in  $D_1$ : {low, medium} and 4 in  $D_2$

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

but  $gini_{\{medium, high\}}$  is 0.30 and thus the best since it is the lowest

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

# Comparing Attribute Selection Measures

---

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

# Other Attribute Selection Measures

---

- CHAID: a popular decision tree algorithm, measure based on  $\chi^2$  test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to  $\chi^2$  distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

# Overfitting and Tree Pruning

---

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

# Enhancements to Basic Decision Tree Induction

---

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

# Classification in Large Databases

---

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

# Scalable Decision Tree Induction Methods

---

- **SLIQ** (EDBT'96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory
- **SPRINT** (VLDB'96 — J. Shafer et al.)
  - Constructs an attribute list data structure
- **PUBLIC** (VLDB'98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)
- **BOAT** (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples

# Scalability Framework for RainForest

---

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: **AVC (Attribute, Value, Class\_label)**
- **AVC-set** (of an attribute  $X$ )
  - Projection of training dataset onto the attribute  $X$  and class label where counts of individual class label are aggregated
- **AVC-group** (of a node  $n$ )
  - Set of AVC-sets of all predictor attributes at the node  $n$

# Rainforest: Training Set and Its AVC Sets

## Training Examples

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

## AVC-set on *Age*

Age	Buy_Computer	
	yes	no
<=30	3	2
31..40	4	0
>40	3	2

## AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

## AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

## AVC-set on *credit\_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

# Data Cube-Based Decision-Tree Induction

---

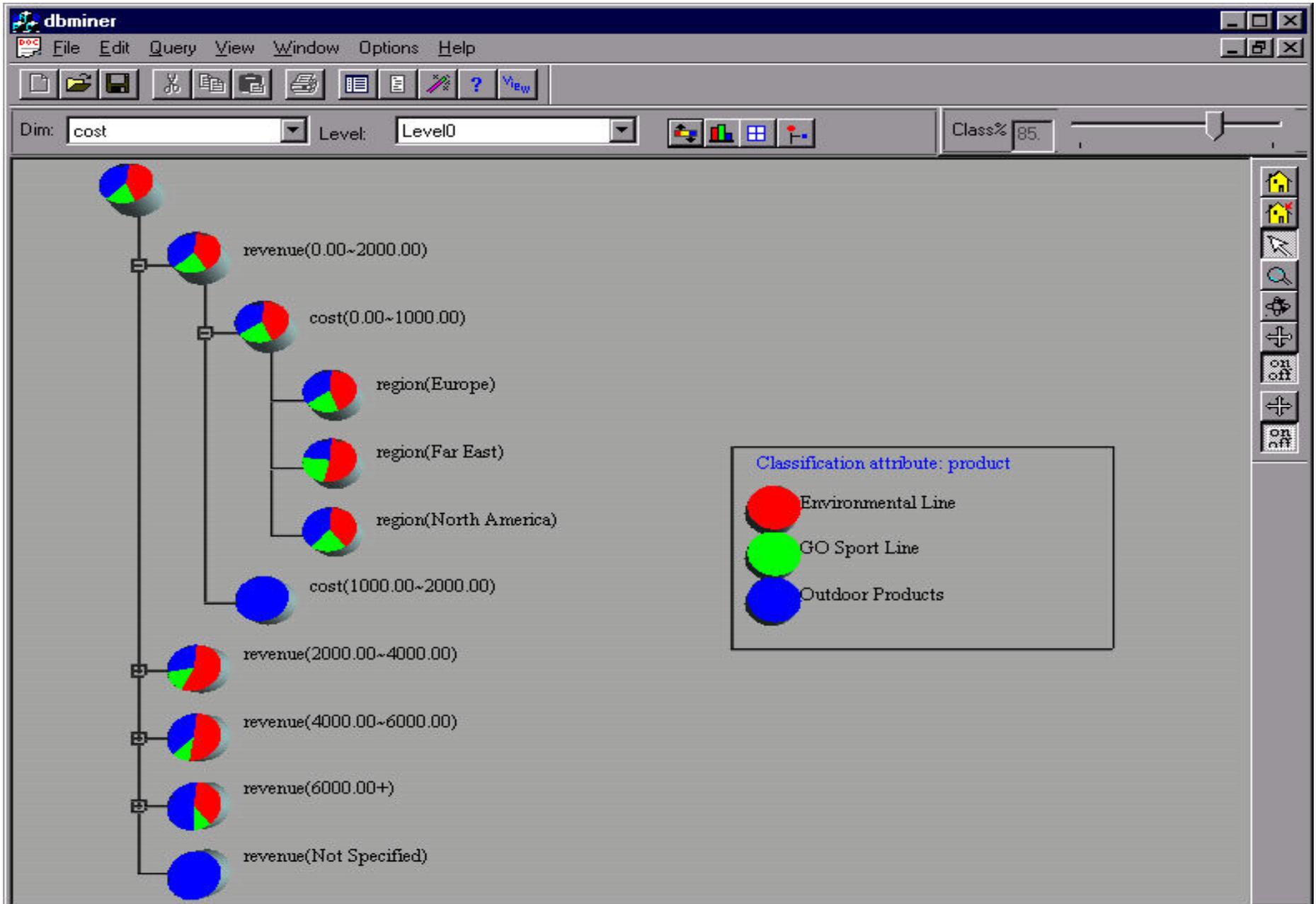
- Integration of generalization with decision-tree induction
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level

# BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

---

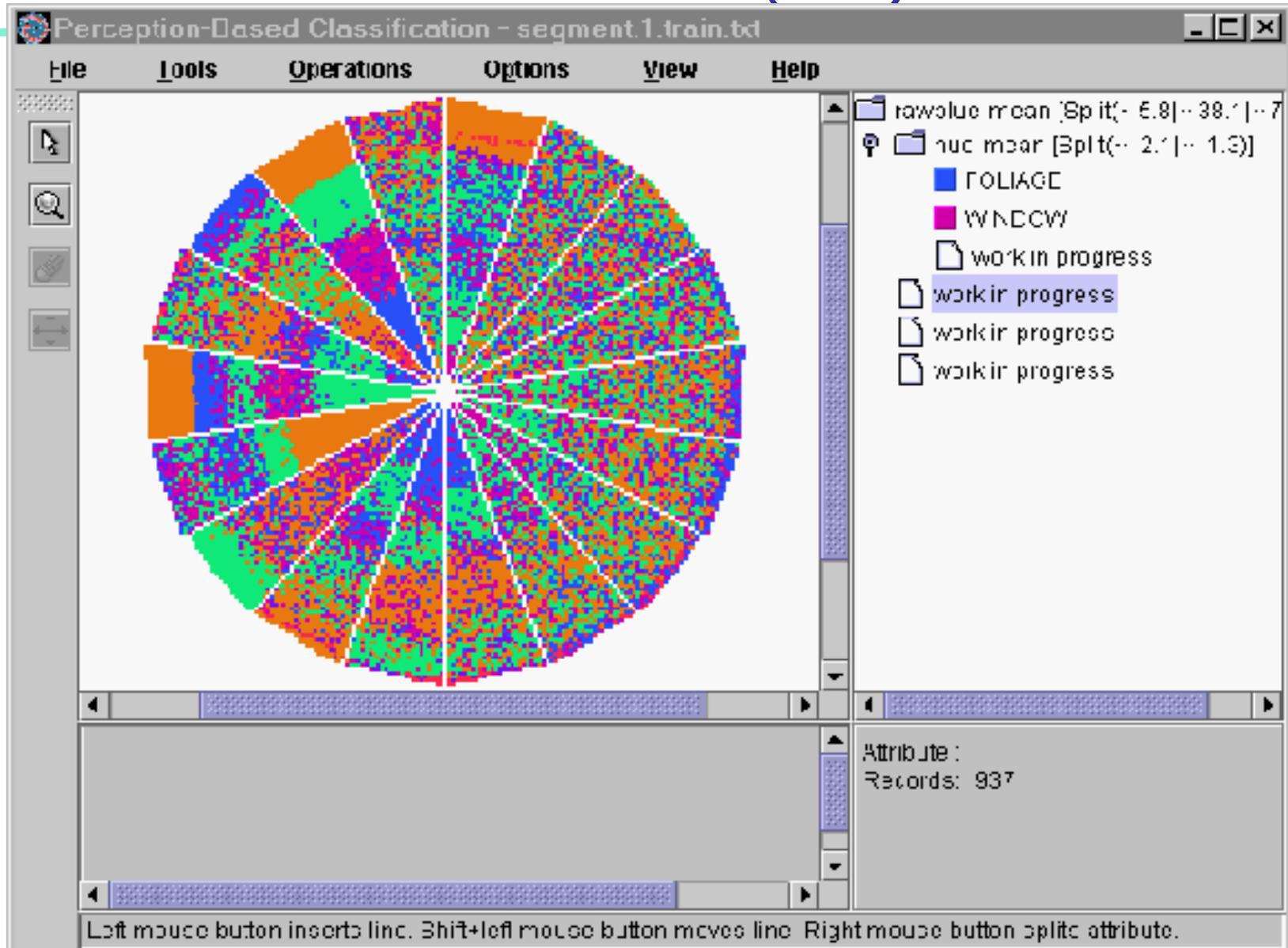
- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree  $T'$ 
  - It turns out that  $T'$  is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

# Presentation of Classification Results





# Interactive Visual Mining by Perception-Based Classification (PBC)



# Bayesian Classification: Why?

---

- A statistical classifier: performs *probabilistic prediction*, *i.e.*, predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

# Bayesian Theorem: Basics

---

- Let  $\mathbf{X}$  be a data sample ("*evidence*"): class label is unknown
- Let  $H$  be a *hypothesis* that  $X$  belongs to class  $C$
- Classification is to determine  $P(H|\mathbf{X})$ , the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$
- $P(H)$  (*prior probability*), the initial probability
  - E.g.,  $\mathbf{X}$  will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$ : probability that sample data is observed
- $P(\mathbf{X}|H)$  (*posteriori probability*), the probability of observing the sample  $\mathbf{X}$ , given that the hypothesis holds
  - E.g., Given that  $\mathbf{X}$  will buy computer, the prob. that  $X$  is 31..40, medium income

# Bayesian Theorem

---

- Given training data  $\mathbf{X}$ , *posteriori probability of a hypothesis*  $H$ ,  $P(H|\mathbf{X})$ , follows the Bayes theorem

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

- Informally, this can be written as  
posteriori = likelihood x prior/evidence
- Predicts  $\mathbf{X}$  belongs to  $C_2$  iff the probability  $P(C_i|\mathbf{X})$  is the highest among all the  $P(C_k|\mathbf{X})$  for all the  $k$  classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

# Towards Naïve Bayesian Classifier

---

- Let  $D$  be a training set of tuples and their associated class labels, and each tuple is represented by an  $n$ -D attribute vector  $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are  $m$  classes  $C_1, C_2, \dots, C_m$ .
- Classification is to derive the maximum posteriori, i.e., the maximal  $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

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

- Since  $P(\mathbf{X})$  is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

# Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

- This greatly reduces the computation cost: Only counts the class distribution
- If  $A_k$  is categorical,  $P(x_k | C_i)$  is the # of tuples in  $C_i$  having value  $x_k$  for  $A_k$  divided by  $|C_i, D|$  (# of tuples of  $C_i$  in  $D$ )
- If  $A_k$  is continuous-valued,  $P(x_k | C_i)$  is usually computed based on Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$

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

and  $P(x_k | C_i)$  is

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

# Naïve Bayesian Classifier: Training Dataset

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Class:

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

Data sample

X = (age <=30,

Income = medium,

Student = yes

Credit\_rating = Fair)

# Naïve Bayesian Classifier: An Example

- $P(C_i)$ :  
 $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute  $P(X|C_i)$  for each class  
 $P(\text{age} = \text{"<=30"} \mid \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$   
 $P(\text{age} = \text{"<= 30"} \mid \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$   
 $P(\text{income} = \text{"medium"} \mid \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$   
 $P(\text{income} = \text{"medium"} \mid \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$   
 $P(\text{student} = \text{"yes"} \mid \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{student} = \text{"yes"} \mid \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$   
 $P(\text{credit\_rating} = \text{"fair"} \mid \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{credit\_rating} = \text{"fair"} \mid \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
- **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**

$$P(X|C_i) : P(X|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) * P(C_i) : P(X|\text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$$

**Therefore, X belongs to class ("buys\_computer = yes")**

# Avoiding the 0-Probability Problem

---

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts

# Naïve Bayesian Classifier: Comments

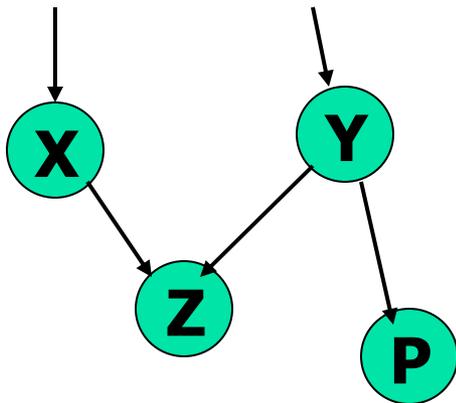
---

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.  
Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

# Bayesian Belief Networks

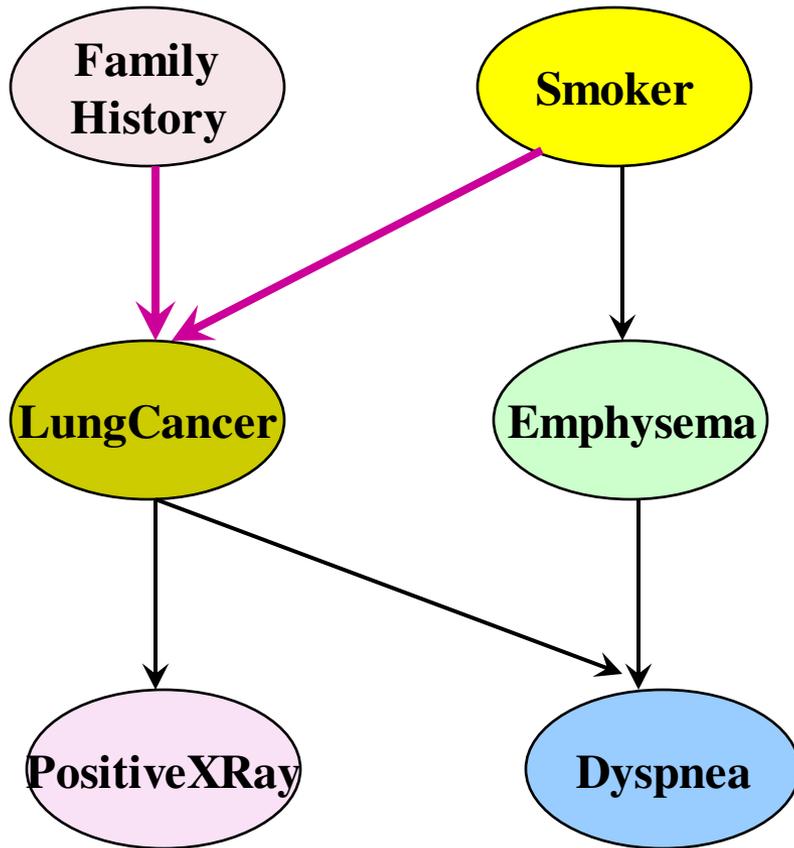
---

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops or cycles

# Bayesian Belief Network: An Example



The **conditional probability table (CPT)** for variable LungCancer:

	(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

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of  $\mathbf{X}$ , from CPT:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | Parents(Y_i))$$

**Bayesian Belief Networks**

# Training Bayesian Networks

---

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref. D. Heckerman: Bayesian networks for data mining

# Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules

R: IF *age* = youth AND *student* = yes THEN *buys\_computer* = yes

- Rule antecedent/precondition vs. rule consequent

- Assessment of a rule: *coverage* and *accuracy*

- $n_{\text{covers}}$  = # of tuples covered by R

- $n_{\text{correct}}$  = # of tuples correctly classified by R

$\text{coverage}(R) = n_{\text{covers}} / |D|$  /\* D: training data set \*/

$\text{accuracy}(R) = n_{\text{correct}} / n_{\text{covers}}$

- If more than one rule is triggered, need **conflict resolution**

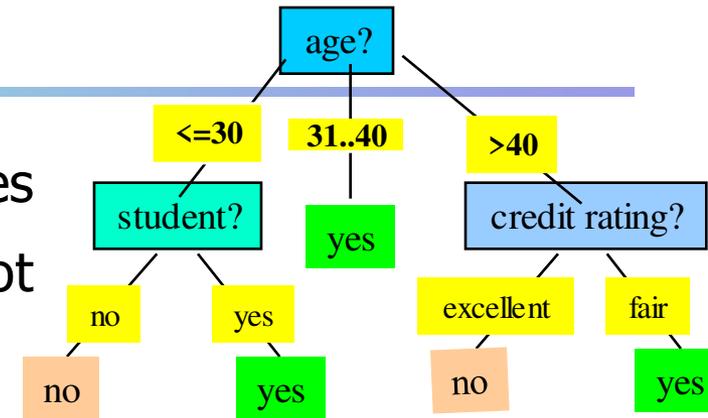
- Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the *most attribute test*)

- Class-based ordering: decreasing order of *prevalence or misclassification cost per class*

- Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

# Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive
- Example: Rule extraction from our *buys\_computer* decision-tree



IF *age* = young AND *student* = *no*

THEN *buys\_computer* = *no*

IF *age* = young AND *student* = *yes*

THEN *buys\_computer* = *yes*

IF *age* = mid-age

THEN *buys\_computer* = *yes*

IF *age* = old AND *credit\_rating* = *excellent* THEN *buys\_computer* = *yes*

IF *age* = young AND *credit\_rating* = *fair* THEN *buys\_computer* = *no*

# Rule Extraction from the Training Data

---

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class  $C_i$  will cover many tuples of  $C_i$  but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

# How to Learn-One-Rule?

- Star with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info\_gain by extending condition

$$FOIL\_Gain = pos' \times \left( \log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg} \right)$$

It favors rules that have high accuracy and cover many positive tuples

- Rule pruning based on an independent set of test tuples

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

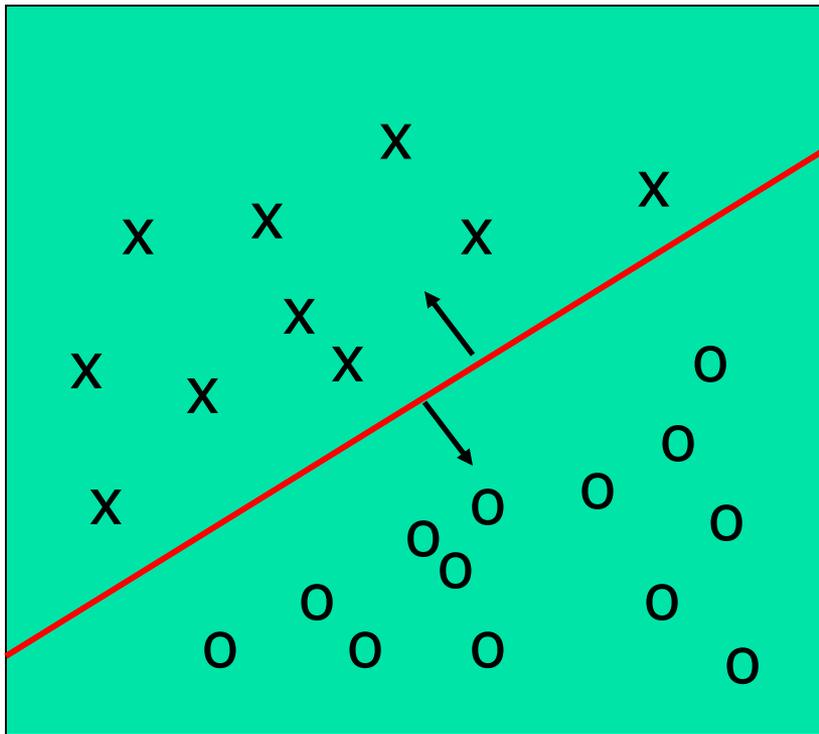
If  $FOIL\_Prune$  is higher for the pruned version of R, prune R

# Classification: A Mathematical Mapping

---

- **Classification:**
  - predicts categorical class labels
- E.g., Personal homepage classification
  - $x_i = (x_1, x_2, x_3, \dots)$ ,  $y_i = +1$  or  $-1$
  - $x_1$  : # of a word "homepage"
  - $x_2$  : # of a word "welcome"
- Mathematically
  - $x \in X = \mathfrak{R}^n$ ,  $y \in Y = \{+1, -1\}$
  - We want a function  $f: X \rightarrow Y$

# Linear Classification



- Binary Classification problem
- The data above the red line belongs to class 'x'
- The data below red line belongs to class 'o'
- Examples: SVM, Perceptron, Probabilistic Classifiers

# Discriminative Classifiers

---

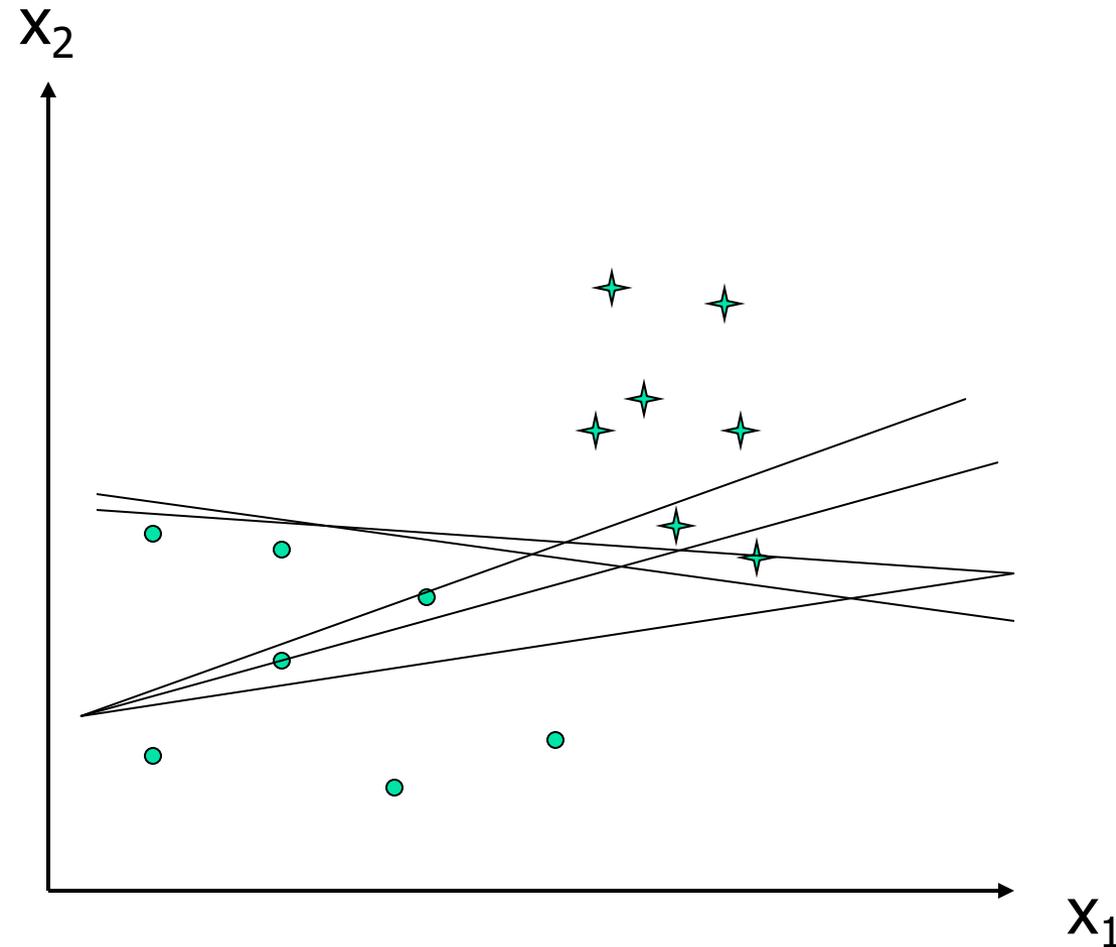
## ■ Advantages

- prediction accuracy is generally high
  - As compared to Bayesian methods – in general
- robust, works when training examples contain errors
- fast evaluation of the learned target function
  - Bayesian networks are normally slow

## ■ Criticism

- long training time
- difficult to understand the learned function (weights)
  - Bayesian networks can be used easily for pattern discovery
- not easy to incorporate domain knowledge
  - Easy in the form of priors on the data or distributions

# Perceptron & Winnow



- Vector:  $x, w$

- Scalar:  $x, y, w$

Input:  $\{(x_1, y_1), \dots\}$

Output: classification function  $f(x)$

$f(x_i) > 0$  for  $y_i = +1$

$f(x_i) < 0$  for  $y_i = -1$

$f(x) \Rightarrow wx + b = 0$

or  $w_1x_1 + w_2x_2 + b = 0$

- Perceptron: update  $W$  additively

- Winnow: update  $W$  multiplicatively

# Classification by Backpropagation

---

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where 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
- Also referred to as **connectionist learning** due to the connections between units

# Neural Network as a Classifier

---

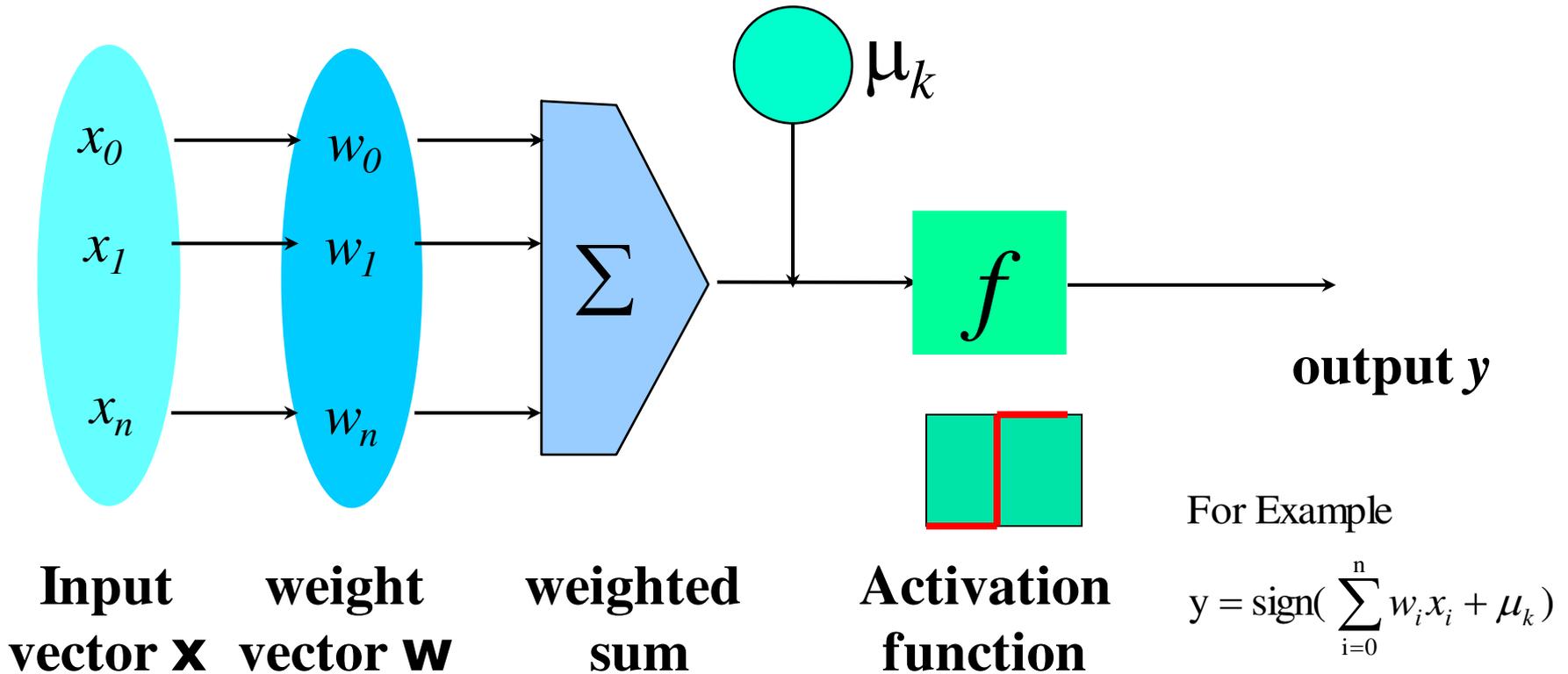
## ■ Weakness

- Long training time
- Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
- Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network

## ■ Strength

- High tolerance to noisy data
- Ability to classify untrained patterns
- Well-suited for continuous-valued inputs and outputs
- Successful on a wide array of real-world data
- Algorithms are inherently parallel
- Techniques have recently been developed for the extraction of rules from trained neural networks

# A Neuron (= a perceptron)



- The  $n$ -dimensional input vector  $\mathbf{x}$  is mapped into variable  $y$  by means of the scalar product and a nonlinear function mapping

# A Multi-Layer Feed-Forward Neural Network

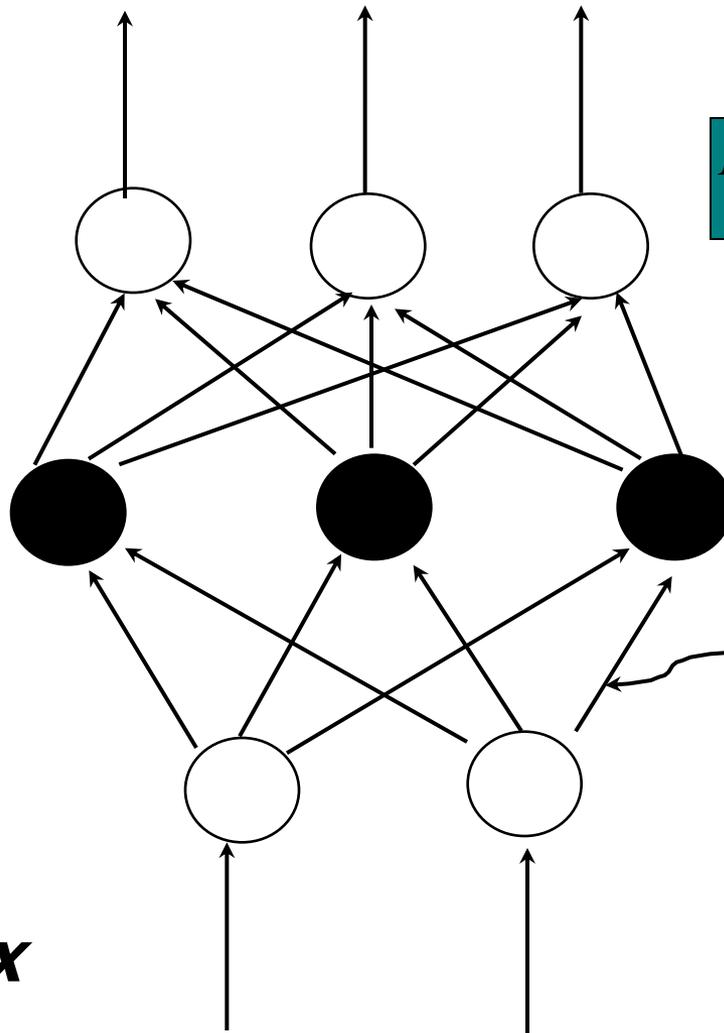
Output vector

Output layer

Hidden layer

Input layer

Input vector:  $X$



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

$$\theta_j = \theta_j + (l) Err_j$$

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

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

$$O_j = \frac{1}{1 + e^{-I_j}}$$

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

# How A Multi-Layer Neural Network Works?

---

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- 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
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

# Defining a Network Topology

---

- First decide the **network topology**: # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

# Backpropagation

---

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the “**backwards**” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “**backpropagation**”
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

# Backpropagation and Interpretability

---

- Efficiency of backpropagation: Each epoch (one iteration through the training set) takes  $O(|D| * w)$ , with  $|D|$  tuples and  $w$  weights, but # of epochs can be exponential to  $n$ , the number of inputs, in the worst case
- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

# SVM—Support Vector Machines

---

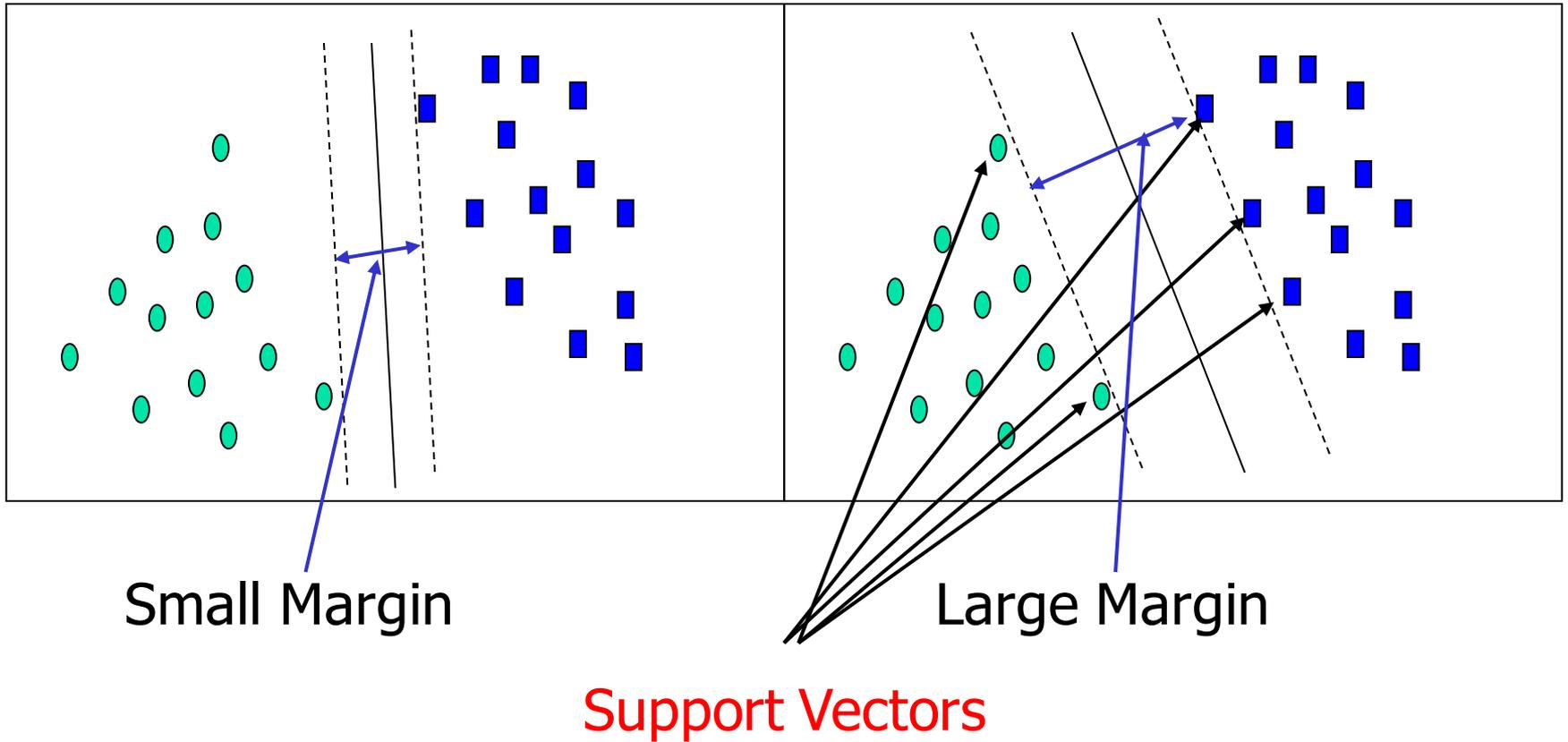
- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., “decision boundary”)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors (“essential” training tuples) and margins (defined by the support vectors)

# SVM—History and Applications

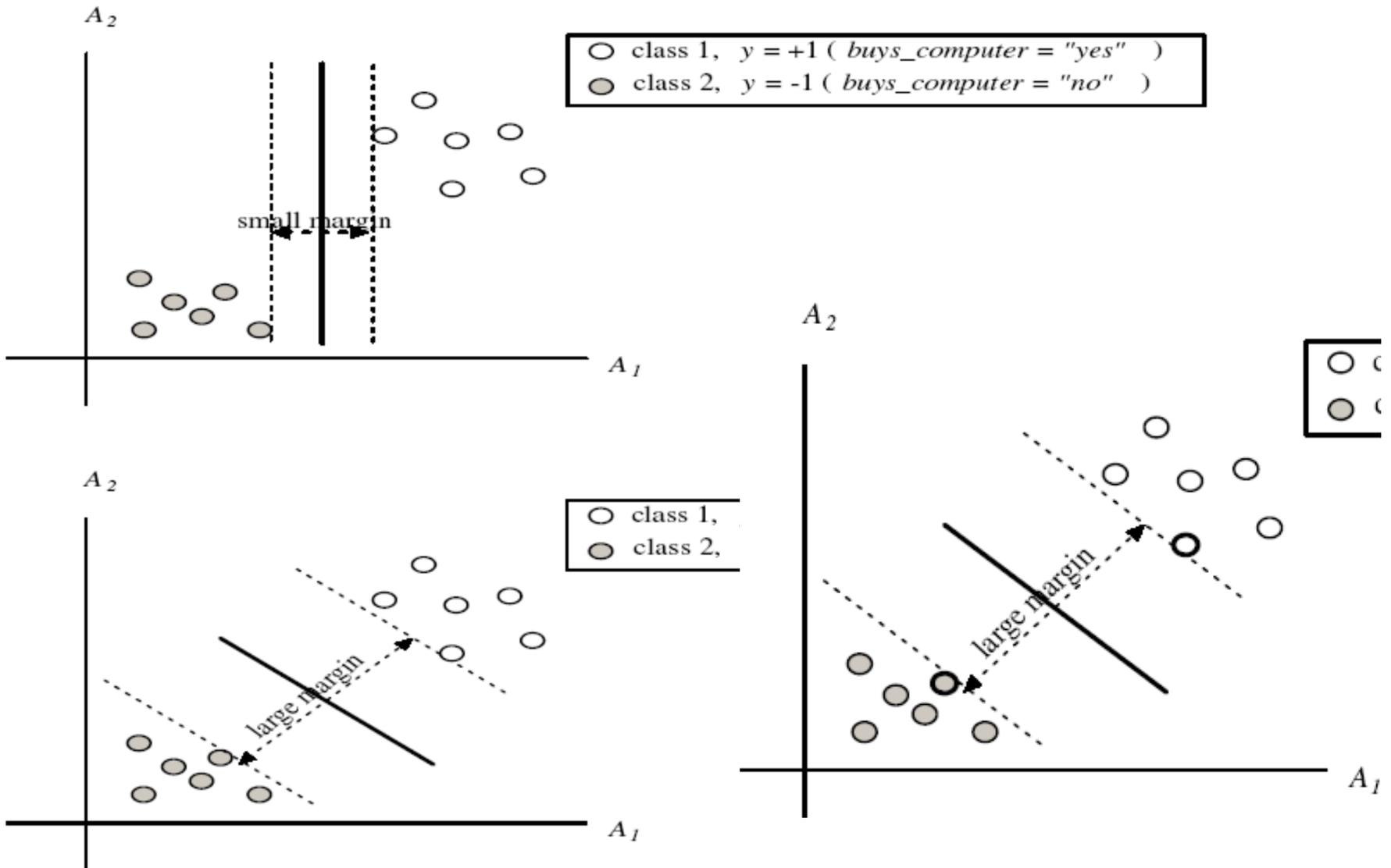
---

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

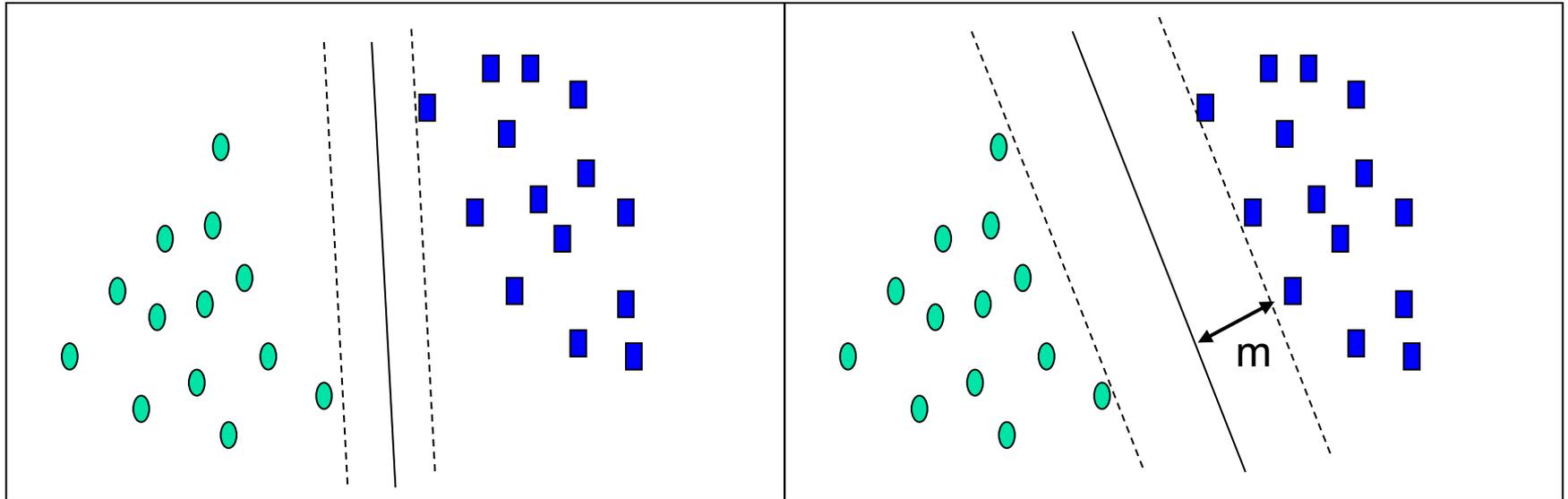
# SVM—General Philosophy



# SVM—Margins and Support Vectors



# SVM—When Data Is Linearly Separable



Let data  $D$  be  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{|D|}, y_{|D|})$ , where  $\mathbf{x}_i$  is the set of training tuples associated with the class labels  $y_i$

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

# SVM—Linearly Separable

- A separating hyperplane can be written as

$$\mathbf{W} \bullet \mathbf{X} + b = 0$$

where  $\mathbf{W} = \{w_1, w_2, \dots, w_n\}$  is a weight vector and  $b$  a scalar (bias)

- For 2-D it can be written as

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- The hyperplane defining the sides of the margin:

$$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1, \text{ and}$$

$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1$$

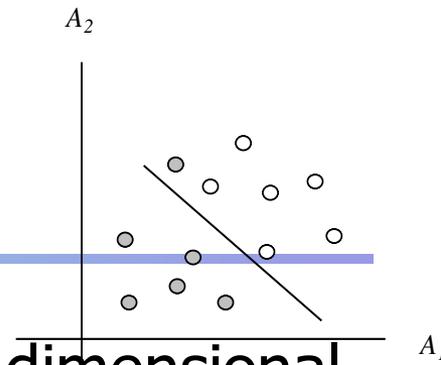
- Any training tuples that fall on hyperplanes  $H_1$  or  $H_2$  (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints  $\rightarrow$  *Quadratic Programming (QP)*  $\rightarrow$  Lagrangian multipliers

# Why Is SVM Effective on High Dimensional Data?

---

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples — they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

# SVM—Linearly Inseparable



- Transform the original input data into a higher dimensional space

**Example 6.8** Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector  $\mathbf{X} = (x_1, x_2, x_3)$  is mapped into a 6D space  $Z$  using the mappings  $\phi_1(\mathbf{X}) = x_1, \phi_2(\mathbf{X}) = x_2, \phi_3(\mathbf{X}) = x_3, \phi_4(\mathbf{X}) = (x_1)^2, \phi_5(\mathbf{X}) = x_1x_2$ , and  $\phi_6(\mathbf{X}) = x_1x_3$ . A decision hyperplane in the new space is  $d(\mathbf{Z}) = \mathbf{WZ} + b$ , where  $\mathbf{W}$  and  $\mathbf{Z}$  are vectors. This is linear. We solve for  $\mathbf{W}$  and  $b$  and then substitute back so that we see that the linear decision hyperplane in the new ( $\mathbf{Z}$ ) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(\mathbf{Z}) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \quad \blacksquare \end{aligned}$$

- Search for a linear separating hyperplane in the new space

# SVM—Kernel functions

---

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a kernel function  $K(\mathbf{X}_i, \mathbf{X}_j)$  to the original data, i.e.,  $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \cdot \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

Polynomial kernel of degree  $h$  :  $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$

Gaussian radial basis function kernel :  $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$

Sigmoid kernel :  $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$

- SVM can also be used for classifying multiple ( $> 2$ ) classes and for regression analysis (with additional user parameters)

# Scaling SVM by Hierarchical Micro-Clustering

---

- SVM is not scalable to the number of data objects in terms of training time and memory usage
- “Classifying Large Datasets Using SVMs with Hierarchical Clusters Problem” by Hwanjo Yu, Jiong Yang, Jiawei Han, KDD’03
- CB-SVM (Clustering-Based SVM)
  - Given limited amount of system resources (e.g., memory), maximize the SVM performance in terms of accuracy and the training speed
  - Use micro-clustering to effectively reduce the number of points to be considered
  - At deriving support vectors, de-cluster micro-clusters near “candidate vector” to ensure high classification accuracy

# CB-SVM: Clustering-Based SVM

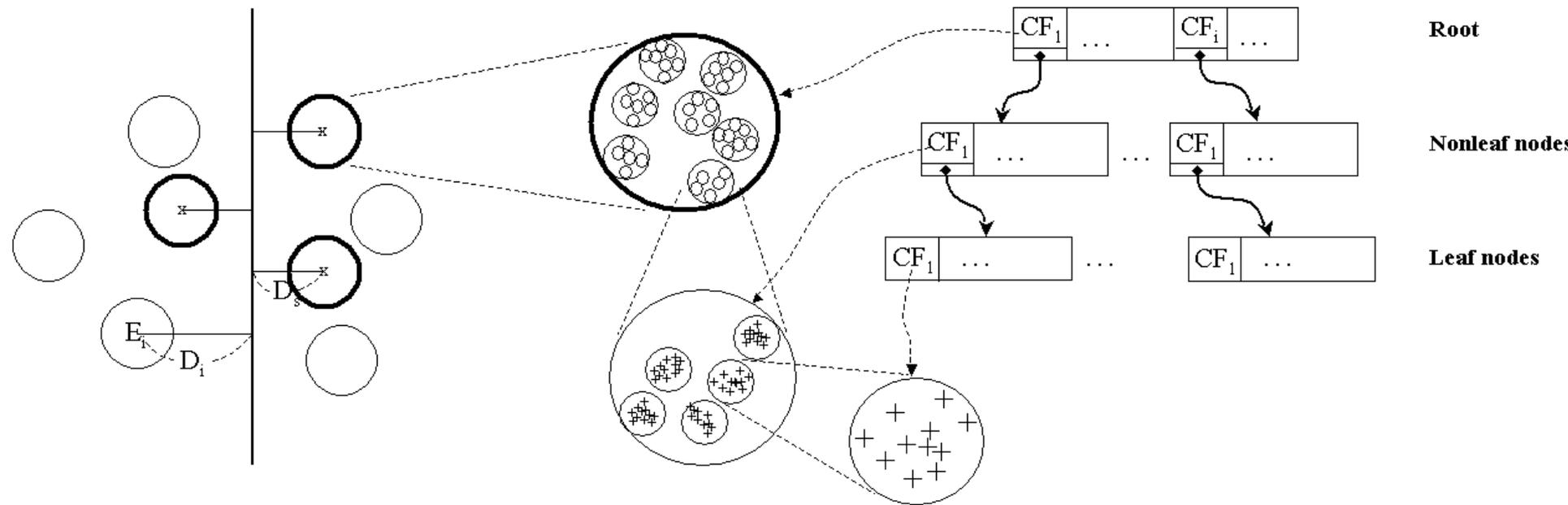
---

- Training data sets may not even fit in memory
- Read the data set once (minimizing disk access)
  - Construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory
  - The statistical summary maximizes the benefit of learning SVM
- The summary plays a role in indexing SVMs
- Essence of Micro-clustering (Hierarchical indexing structure)
  - Use micro-cluster hierarchical indexing structure
    - provide finer samples closer to the boundary and coarser samples farther from the boundary
  - Selective de-clustering to ensure high accuracy

# CF-Tree: Hierarchical Micro-cluster

Negative clusters

Positive clusters



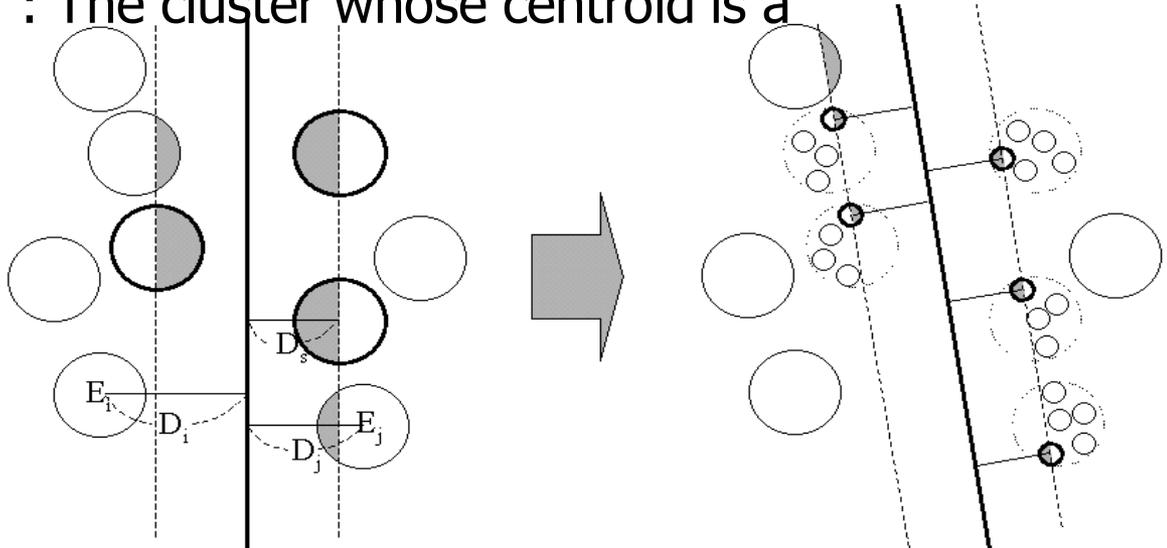
# CB-SVM Algorithm: Outline

---

- Construct two CF-trees from positive and negative data sets independently
  - Need one scan of the data set
- Train an SVM from the centroids of the root entries
- De-cluster the entries near the boundary into the next level
  - The children entries de-clustered from the parent entries are accumulated into the training set with the non-declustered parent entries
- Train an SVM again from the centroids of the entries in the training set
- Repeat until nothing is accumulated

# Selective Declustering

- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster  $E_i$  such that
  - $D_i - R_i < D_s$ , where  $D_i$  is the distance from the boundary to the center point of  $E_i$  and  $R_i$  is the radius of  $E_i$
  - Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
    - "Support cluster": The cluster whose centroid is a support vector



# Experiment on Synthetic Dataset

---

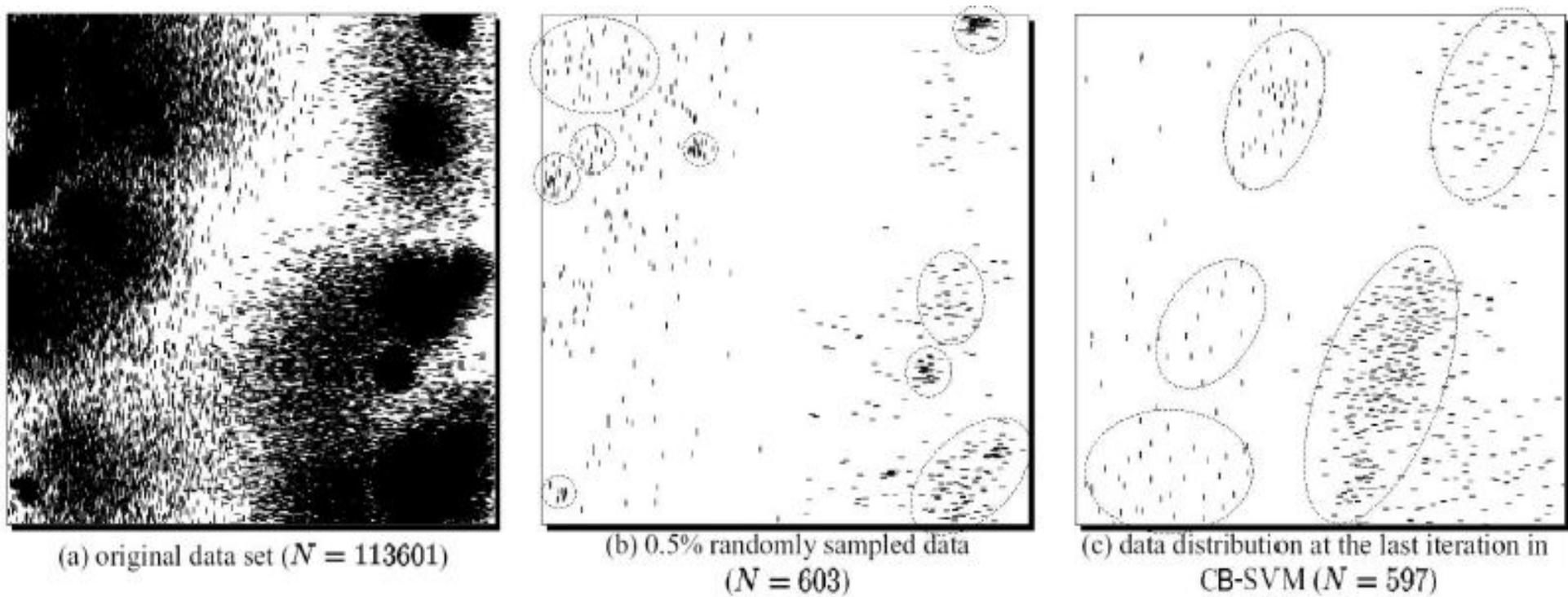


Figure 6: Synthetic data set in a two-dimensional space. ‘|’: positive data; ‘-’: negative data

# Experiment on a Large Data Set

---

S-Rate	# of data	# of errors	T-Time	S-Time
0.0001%	23	6425	0.000114	822.97
0.001%	226	2413	0.000972	825.40
0.01%	2333	1132	0.03	828.61
0.1%	23273	1012	6.287	835.87
1%	230380	1015	1192.793	838.92
5%	1151714	1020	20705.4	842.92
ASVM	307	865	54872.213	
CB-SVM	2893	876	1.639	2528.213

**Table 4: Performance results on the very large data set (# of training data = 23066169, # of testing data = 233890). S-Rate: sampling rate; T-Time: training time; S-Time: sampling time; ASVM: selective sampling**

# SVM vs. Neural Network

---

## ■ SVM

- Relatively new concept
- Deterministic algorithm
- Nice Generalization properties
- Hard to learn – learned in batch mode using quadratic programming techniques
- Using kernels can learn very complex functions

## ■ Neural Network

- Relatively old
- Nondeterministic algorithm
- Generalizes well but doesn't have strong mathematical foundation
- Can easily be learned in incremental fashion
- To learn complex functions—use multilayer perceptron (not that trivial)

# Associative Classification

---

- Associative classification
  - Association rules are generated and analyzed for use in classification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Classification: Based on evaluating a set of rules in the form of
$$P_1 \wedge p_2 \dots \wedge p_l \rightarrow "A_{\text{class}} = C" (\text{conf}, \text{sup})$$
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - In many studies, associative classification has been found to be more accurate than some traditional classification methods, such as C4.5

# Typical Associative Classification Methods

---

- CBA
  - Mine association possible rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- CMAR
  - Classification: Statistical analysis on multiple rules
- CPAR
  - Generation of predictive rules (FOIL-like analysis)
  - High efficiency, accuracy similar to CMAR
- RCBT
  - Explore high-dimensional classification, using top-k rule groups
  - Achieve high classification accuracy and high run-time efficiency

# A Closer Look at CMAR

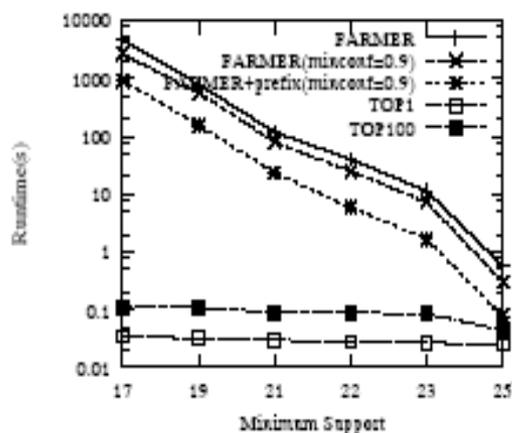
---

- CMAR
- Efficiency: Uses an enhanced FP-tree that maintains the distribution of class labels among tuples satisfying each frequent itemset
- Rule pruning whenever a rule is inserted into the tree
  - Given two rules,  $R_1$  and  $R_2$ , if the antecedent of  $R_1$  is more general than that of  $R_2$  and  $\text{conf}(R_1) \geq \text{conf}(R_2)$ , then  $R_2$  is pruned
  - Prunes rules for which the rule antecedent and class are not positively correlated, based on a  $\chi^2$  test of statistical significance
- Classification based on generated/pruned rules
  - If only one rule satisfies tuple  $X$ , assign the class label of the rule
  - If a rule set  $S$  satisfies  $X$ , CMAR
    - divides  $S$  into groups according to class labels
    - uses a weighted  $\chi^2$  measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - assigns  $X$  the class label of the strongest group

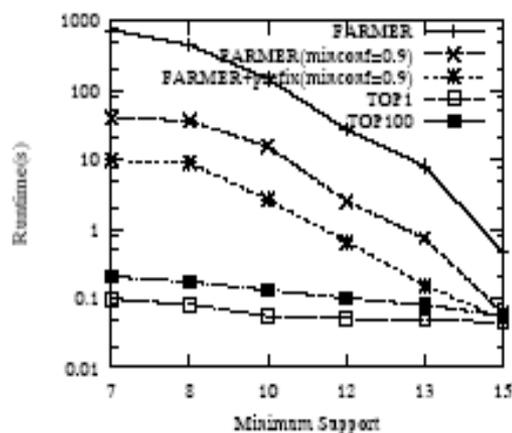
# Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

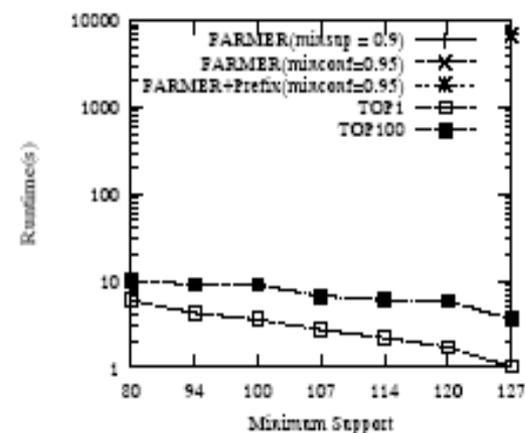
Table 2: Classification Results



(a) ALL-AML leukemia



(b) Lung Cancer



(c) Ovarian Cancer

# Lazy vs. Eager Learning

---

- Lazy vs. eager learning
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

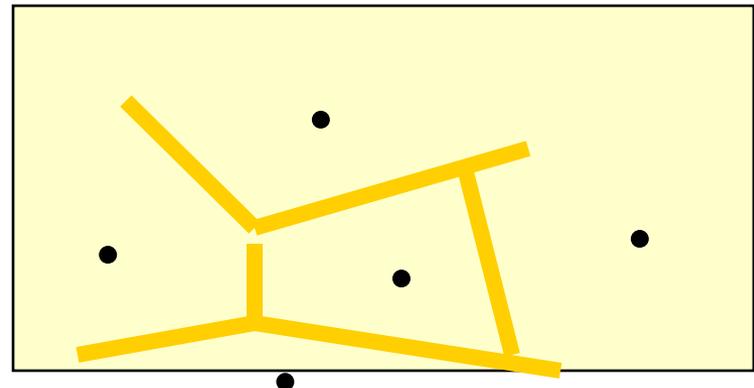
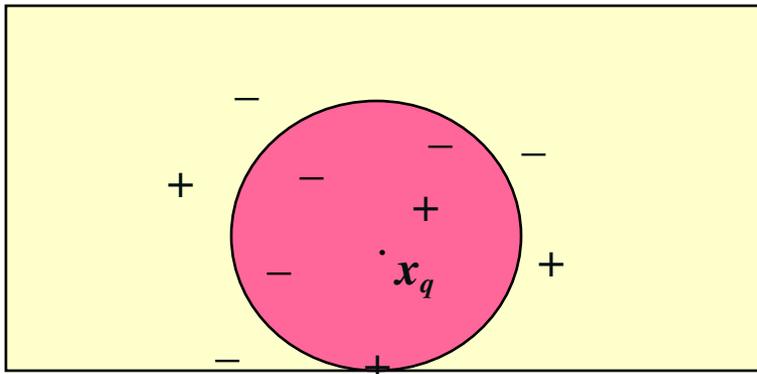
# Lazy Learner: Instance-Based Methods

---

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

# The $k$ -Nearest Neighbor Algorithm

- All instances correspond to points in the  $n$ -D space
- The nearest neighbor are defined in terms of Euclidean distance,  $\text{dist}(\mathbf{X}_1, \mathbf{X}_2)$
- Target function could be discrete- or real- valued
- For discrete-valued,  $k$ -NN returns the most common value among the  $k$  training examples nearest to  $x_q$
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples



# Discussion on the $k$ -NN Algorithm

---

- $k$ -NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the  $k$  nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the  $k$  neighbors according to their distance to the query  $x_q$ 
    - Give greater weight to closer neighbors  $w \equiv \frac{1}{d(x_q, x_i)^2}$
- Robust to noisy data by averaging  $k$ -nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

# Case-Based Reasoning (CBR)

---

- CBR: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

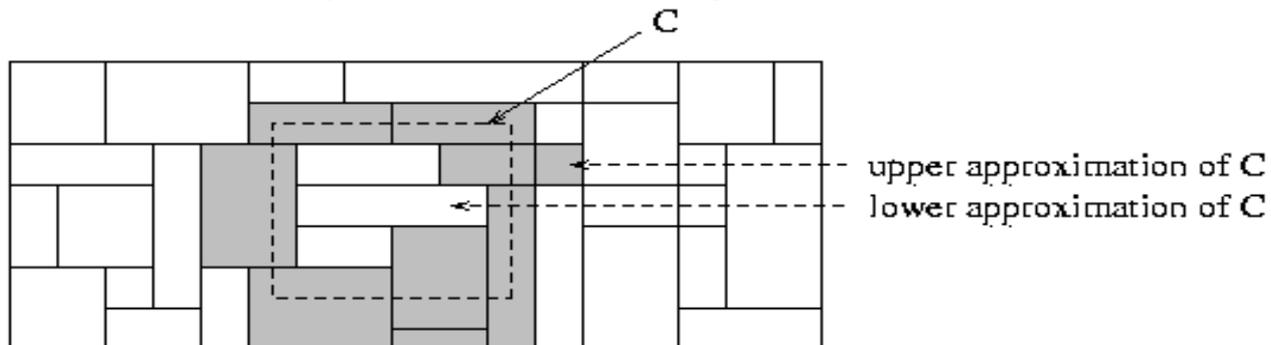
# Genetic Algorithms (GA)

---

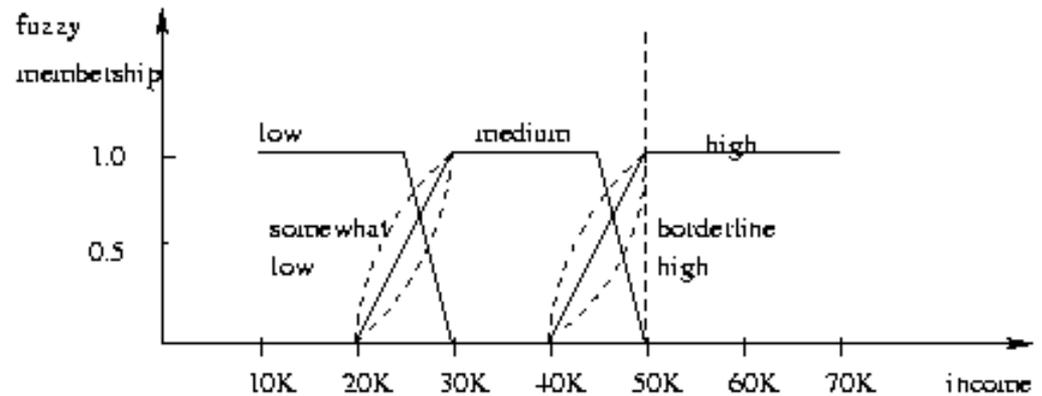
- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if  $A_1$  and  $\neg A_2$  then  $C_2$  can be encoded as 100
  - If an attribute has  $k > 2$  values,  $k$  bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offsprings
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offsprings are generated by *crossover* and *mutation*
- The process continues until a population  $P$  evolves *when each rule in  $P$  satisfies a prespecified threshold*
- Slow but easily parallelizable

# Rough Set Approach

- Rough sets are used to **approximately or “roughly” define equivalent classes**
- A rough set for a given class  $C$  is approximated by two sets: a **lower approximation** (certain to be in  $C$ ) and an **upper approximation** (cannot be described as not belonging to  $C$ )
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



# Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using **fuzzy membership graph**)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined

# What Is Prediction?

---

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

# Linear Regression

- Linear regression: involves a response variable  $y$  and a single predictor variable  $x$

$$y = w_0 + w_1 x$$

where  $w_0$  (y-intercept) and  $w_1$  (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form  $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_{|D|}, y_{|D|})$
  - Ex. For 2-D data, we may have:  $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above

# Nonlinear Regression

---

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

convertible to linear with new variables:  $x_2 = x^2$ ,  $x_3 = x^3$

$$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

# Other Regression-Based Models

---

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of  $y$  is a function of the mean value of  $y$ , not a constant
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels

# Regression Trees and Model Trees

---

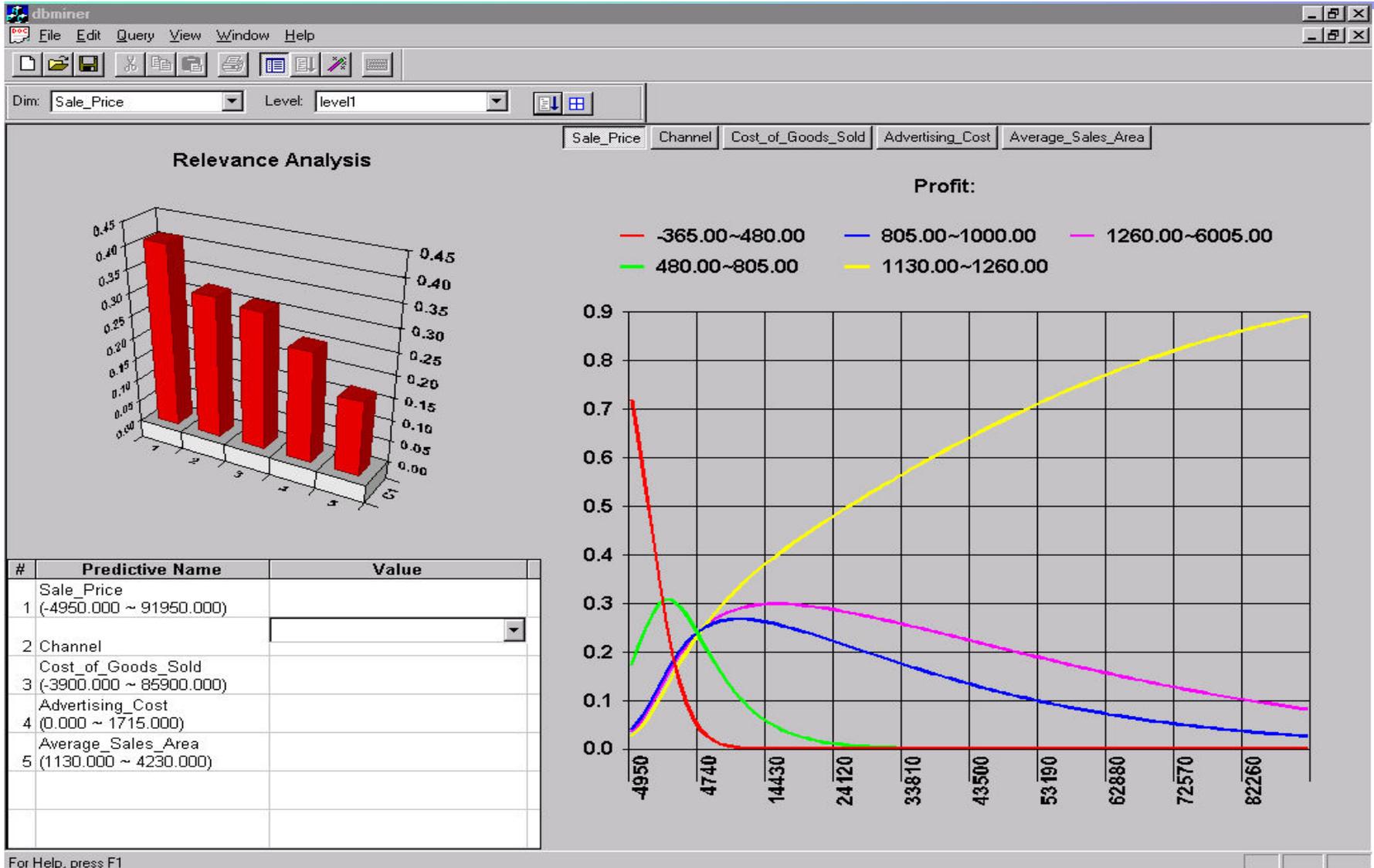
- Regression tree: proposed in CART system
  - CART: Classification And Regression Trees
  - Each leaf stores a *continuous-valued prediction*
  - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

# Predictive Modeling in Multidimensional Databases

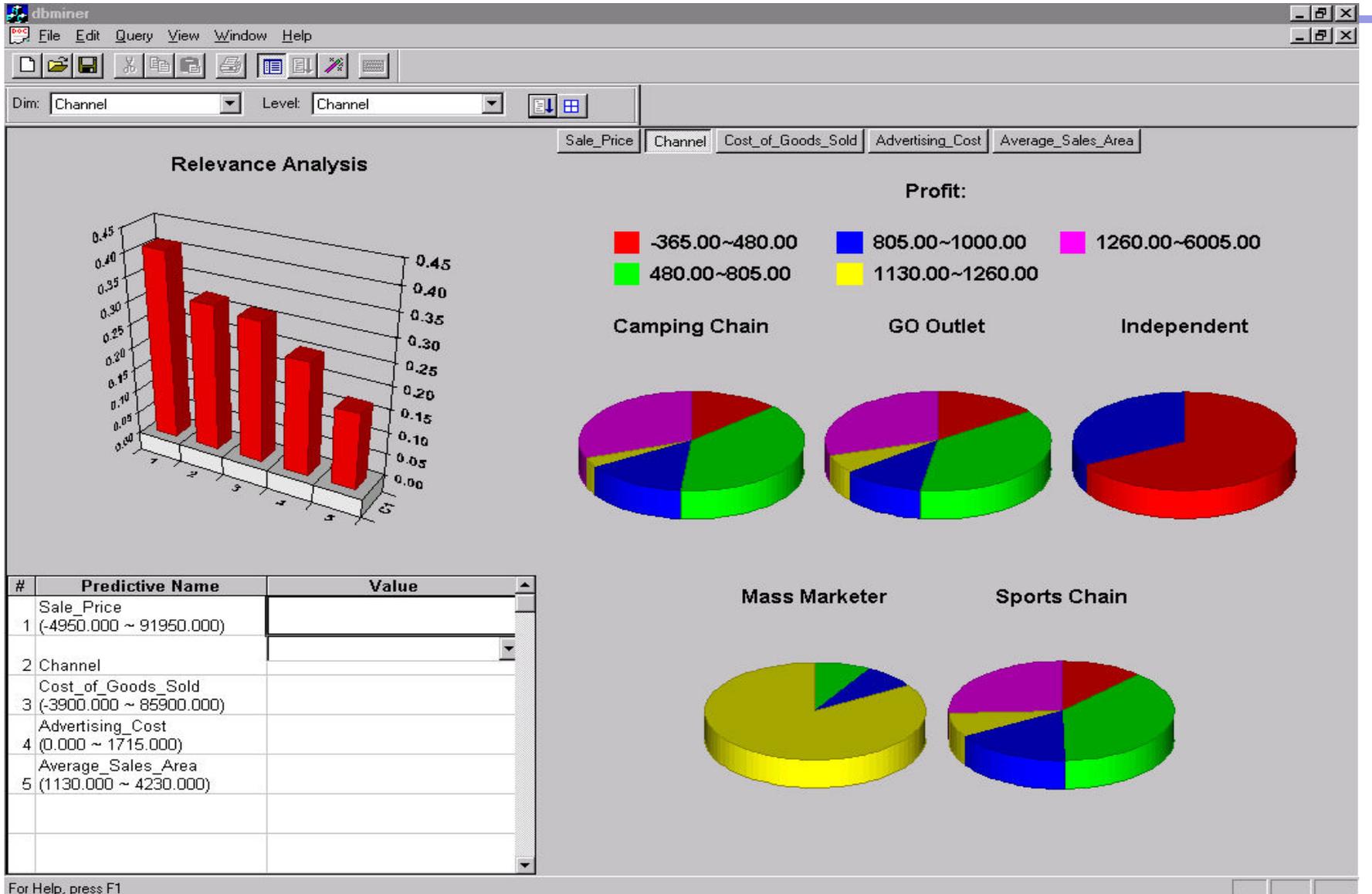
---

- Predictive modeling: Predict data values or construct generalized linear models based on the database data
- One can only predict value ranges or category distributions
- Method outline:
  - Minimal generalization
  - Attribute relevance analysis
  - Generalized linear model construction
  - Prediction
- Determine the major factors which influence the prediction
  - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis

# Prediction: Numerical Data



# Prediction: Categorical Data



# Classifier Accuracy Measures

	$C_1$	$C_2$
$C_1$	True positive	False negative
$C_2$	False positive	True negative

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy of a classifier  $M$ ,  $\text{acc}(M)$ : percentage of test set tuples that are correctly classified by the model  $M$ 
  - Error rate (misclassification rate) of  $M = 1 - \text{acc}(M)$
  - Given  $m$  classes,  $CM_{i,j}$  an entry in a **confusion matrix**, indicates # of tuples in class  $i$  that are labeled by the classifier as class  $j$
- Alternative accuracy measures (e.g., for cancer diagnosis)
  - sensitivity =  $t\text{-pos}/\text{pos}$                     /\* true positive recognition rate \*/
  - specificity =  $t\text{-neg}/\text{neg}$                     /\* true negative recognition rate \*/
  - precision =  $t\text{-pos}/(t\text{-pos} + f\text{-pos})$
  - accuracy =  $\text{sensitivity} * \text{pos}/(\text{pos} + \text{neg}) + \text{specificity} * \text{neg}/(\text{pos} + \text{neg})$
  - This model can also be used for cost-benefit analysis

# Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function:** measures the error betw.  $y_i$  and the predicted value  $y_i'$ 
  - Absolute error:  $|y_i - y_i'|$
  - Squared error:  $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean absolute error:  $\frac{\sum_{i=1}^d |y_i - y_i'|}{d}$     Mean squared error:  $\frac{\sum_{i=1}^d (y_i - y_i')^2}{d}$
  - Relative absolute error:  $\frac{\sum_{i=1}^d |y_i - y_i'|}{\sum_{i=1}^d |y_i - \bar{y}|}$     Relative squared error:  $\frac{\sum_{i=1}^d (y_i - y_i')^2}{\sum_{i=1}^d (y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers

Popularly use (square) root mean-square error, similarly, root relative squared error

# Evaluating the Accuracy of a Classifier or Predictor (I)

---

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout  $k$  times, accuracy = avg. of the accuracies obtained
- Cross-validation ( $k$ -fold, where  $k = 10$  is most popular)
  - Randomly partition the data into  $k$  *mutually exclusive* subsets, each approximately equal size
  - At  $i$ -th iteration, use  $D_i$  as test set and others as training set
  - Leave-one-out:  $k$  folds where  $k = \#$  of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

# Evaluating the Accuracy of a Classifier or Predictor (II)

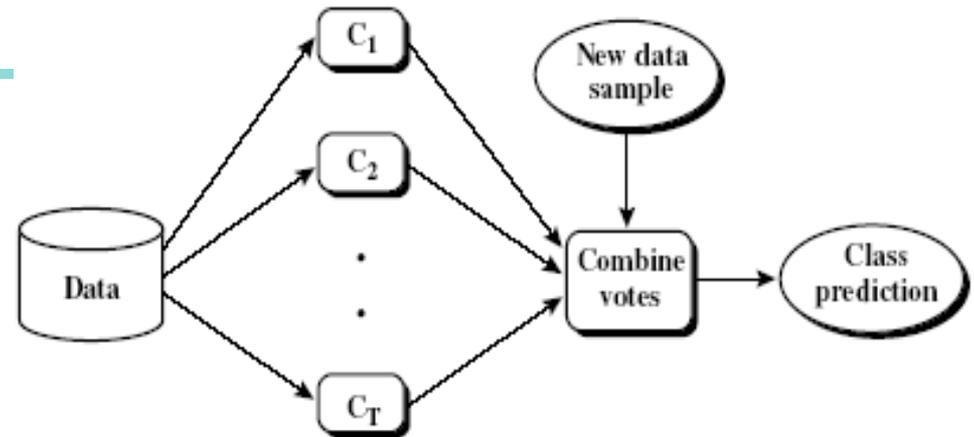
---

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - Suppose we are given a data set of  $d$  tuples. The data set is sampled  $d$  times, with replacement, resulting in a training set of  $d$  samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since  $(1 - 1/d)^d \approx e^{-1} = 0.368$ )
  - Repeat the sampling procedure  $k$  times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^k (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

# Ensemble Methods: Increasing the Accuracy

---



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of  $k$  learned models,  $M_1, M_2, \dots, M_k$ , with the aim of creating an improved model  $M^*$
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

# Bagging: Bootstrap Aggregation

---

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set  $D$  of  $d$  tuples, at each iteration  $i$ , a training set  $D_i$  of  $d$  tuples is sampled with replacement from  $D$  (i.e., bootstrap)
  - A classifier model  $M_i$  is learned for each training set  $D_i$
- Classification: classify an unknown sample  $\mathbf{X}$ 
  - Each classifier  $M_i$  returns its class prediction
  - The bagged classifier  $M^*$  counts the votes and assigns the class with the most votes to  $\mathbf{X}$
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from  $D$
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

# Boosting

---

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of  $k$  classifiers is iteratively learned
  - After a classifier  $M_i$  is learned, the weights are updated to allow the subsequent classifier,  $M_{i+1}$ , to pay more attention to the training tuples that were misclassified by  $M_i$
  - The final  $M^*$  combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

# Adaboost (Freund and Schapire, 1997)

---

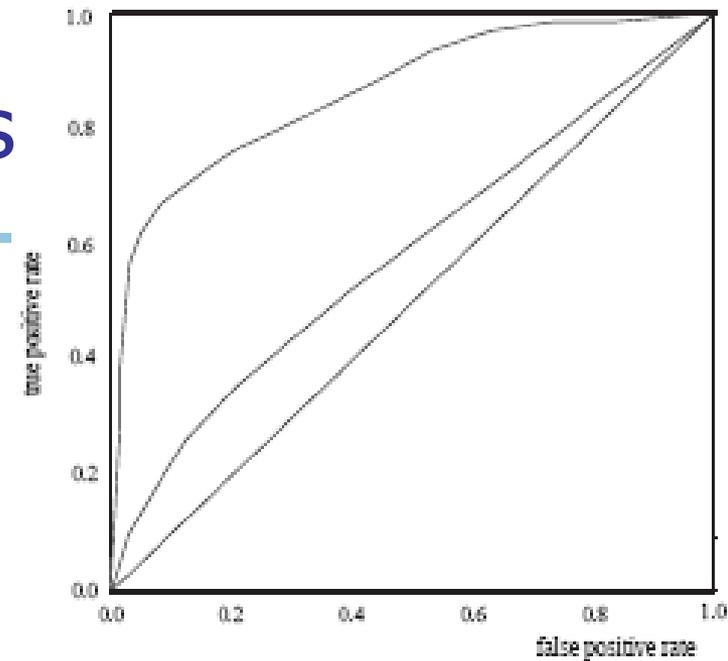
- Given a set of  $d$  class-labeled tuples,  $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the weights of tuples are set the same ( $1/d$ )
- Generate  $k$  classifiers in  $k$  rounds. At round  $i$ ,
  - Tuples from  $D$  are sampled (with replacement) to form a training set  $D_i$  of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model  $M_i$  is derived from  $D_i$
  - Its error rate is calculated using  $D_i$  as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate:  $err(\mathbf{X}_j)$  is the misclassification error of tuple  $\mathbf{X}_j$ . Classifier  $M_i$  error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X}_j)$$

- The weight of classifier  $M_i$ 's vote is  $\log \frac{1 - error(M_i)}{error(M_i)}$

# Model Selection: ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

# Patterns and Models

Rameswara Reddy.K.V

Asst.Professor

CSE Department

# Patterns and Models

- Pattern
- Model
- Visualizing a pattern
- Database
- Record
- Field
- Predictor
- Prediction
- value

# Where are models used?

- Selection
- Acquisition
- Retention
- Extension

# Right Model

- It could always be used to make the correct prediction
- It would not degrade over time
- It could be used with the data at hand not require any extraordinary data collection
- It would be simpler and smaller than the data it was used to model.

# Sampling

- It is a statistical analysis technique used to select, manipulate and analyze a representative subset of data points to identify patterns and trends in the larger data set being examined.
- Random sampling
- Experimental design
- Round robin
- Stratified
- Cluster
- \* Avoiding bias

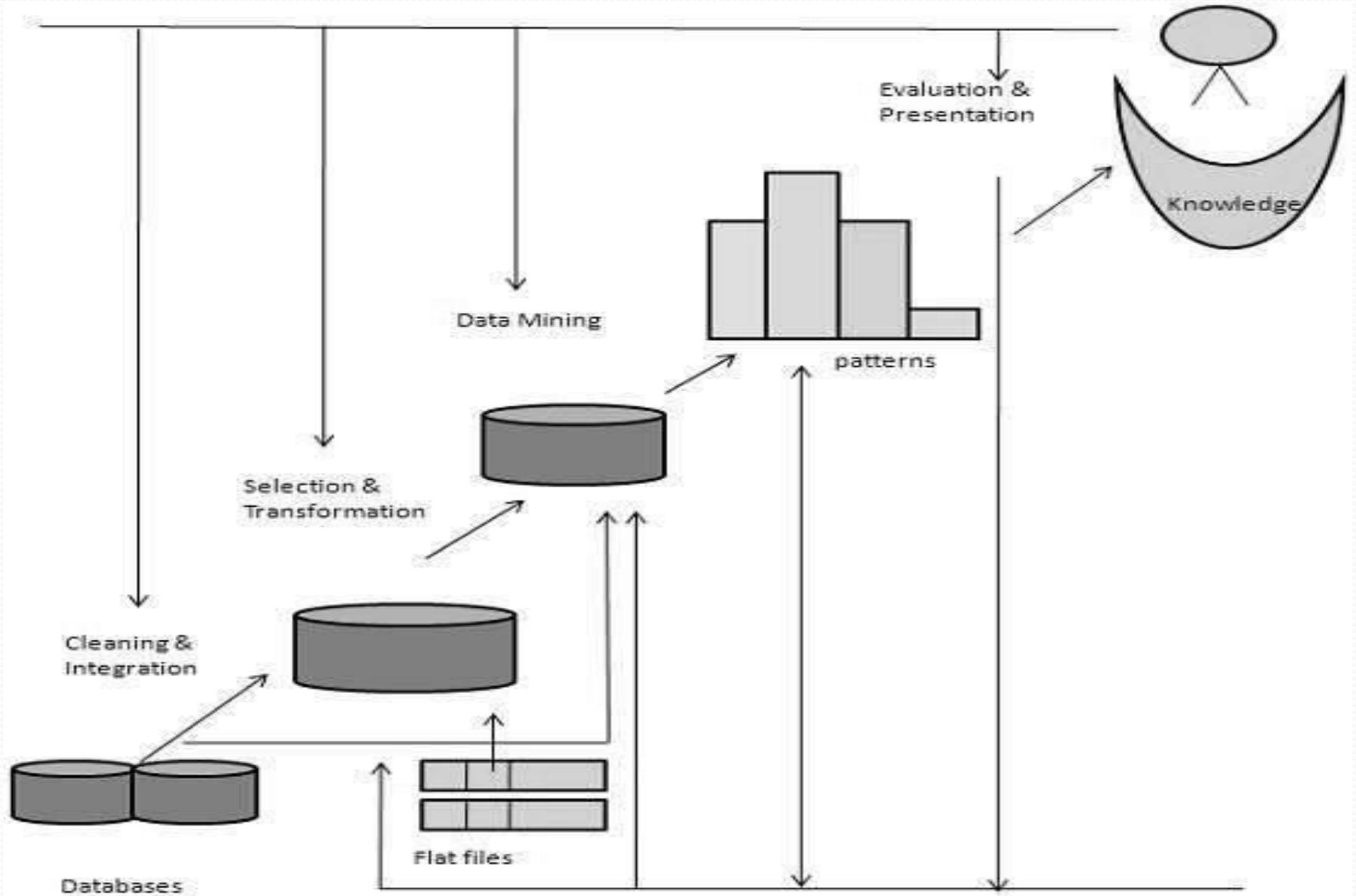
# Data Mining

- Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes. Using a broad range of techniques, you can use this information to increase revenues, cut costs, improve customer relationships, reduce risks and more.

# Applications

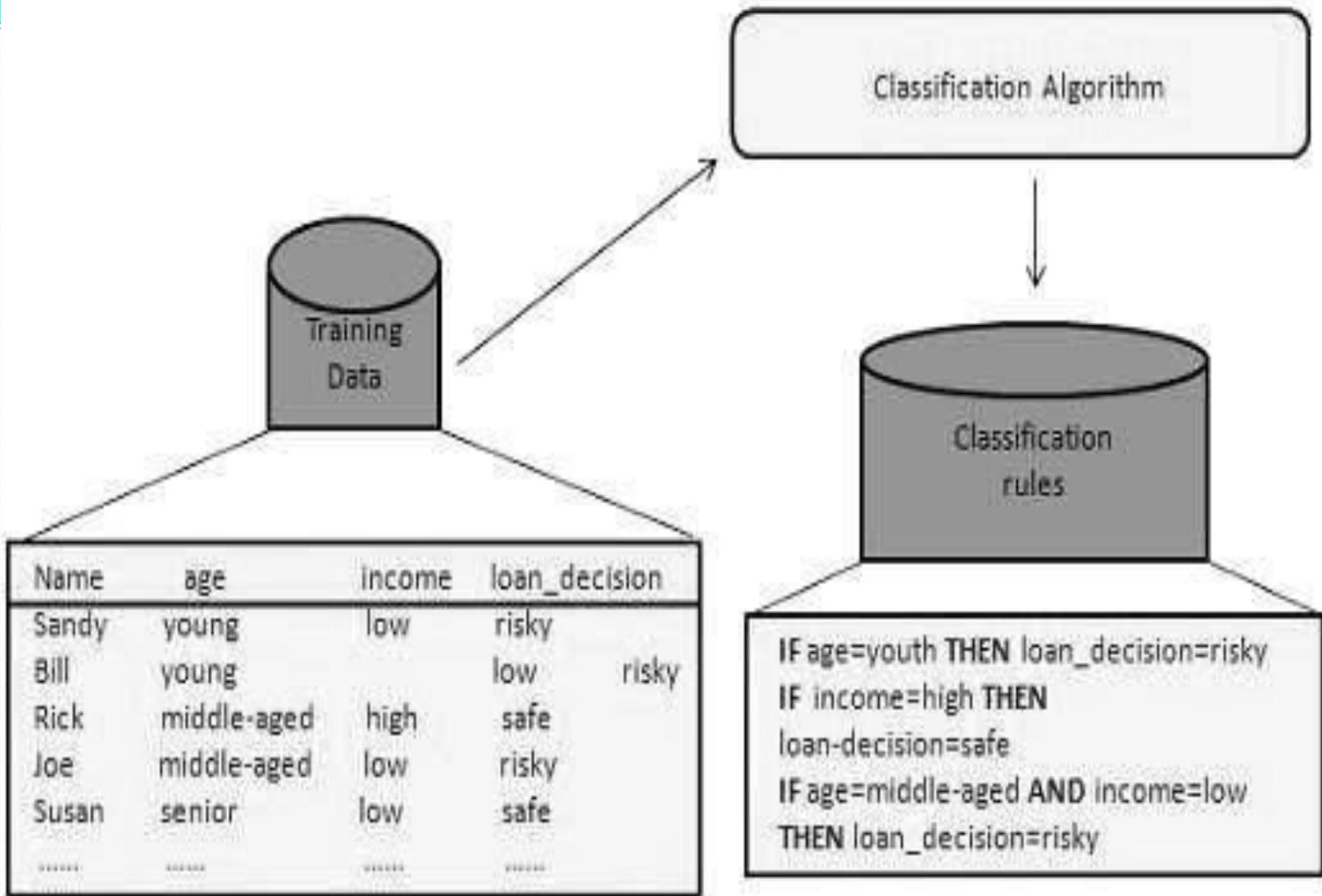
- Data mining is highly useful in the following domains
  - Market Analysis and Management
  - Corporate Analysis & Risk Management
  - Fraud Detection

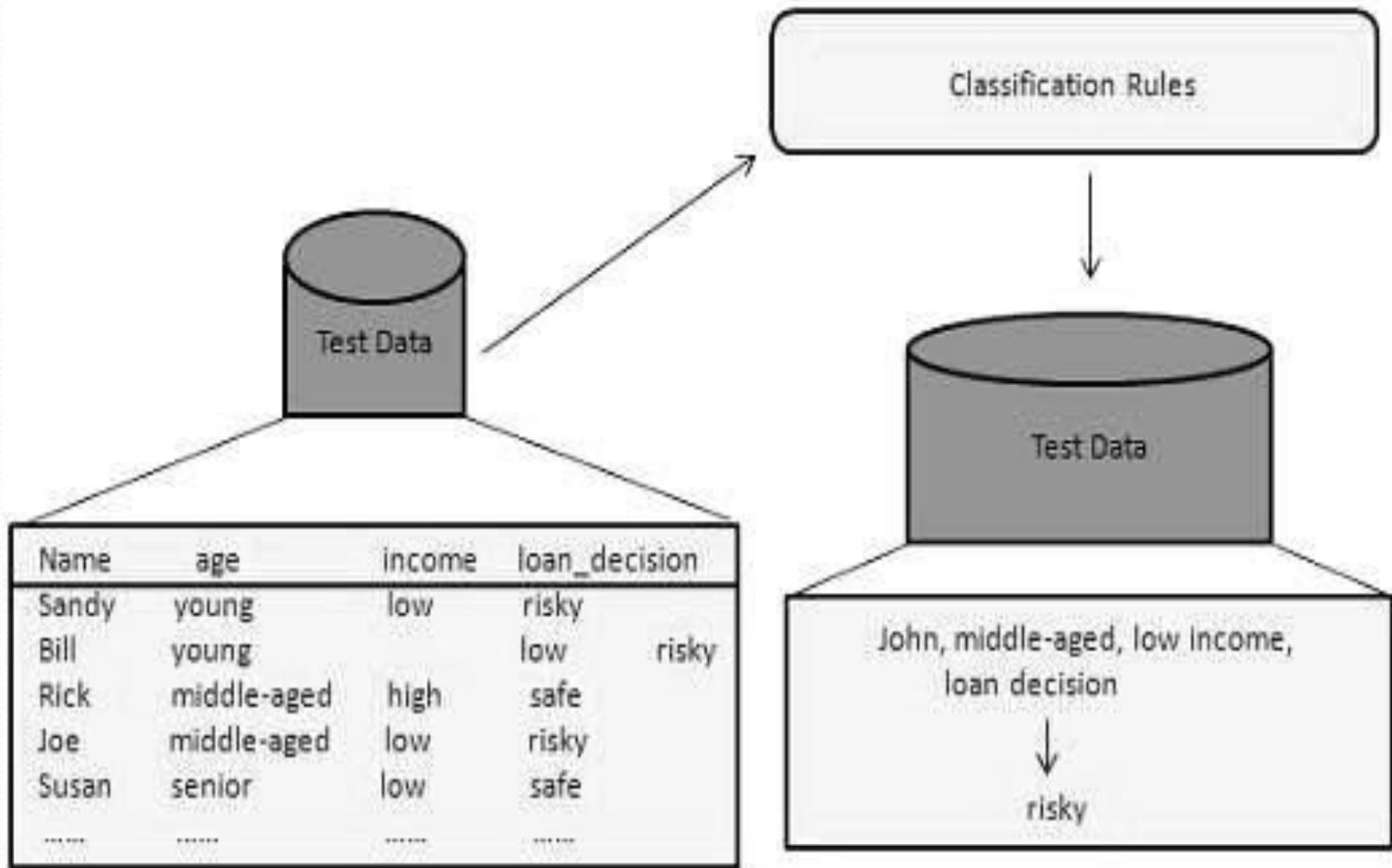
# KDD



- **Data Cleaning** – In this step, the noise and inconsistent data is removed.
- **Data Integration** – In this step, multiple data sources are combined.
- **Data Selection** – In this step, data relevant to the analysis task are retrieved from the database.
- **Data Transformation** – In this step, data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
- **Data Mining** – In this step, intelligent methods are applied in order to extract data patterns.
- **Pattern Evaluation** – In this step, data patterns are evaluated.
- **Knowledge Presentation** – In this step, knowledge is represented.

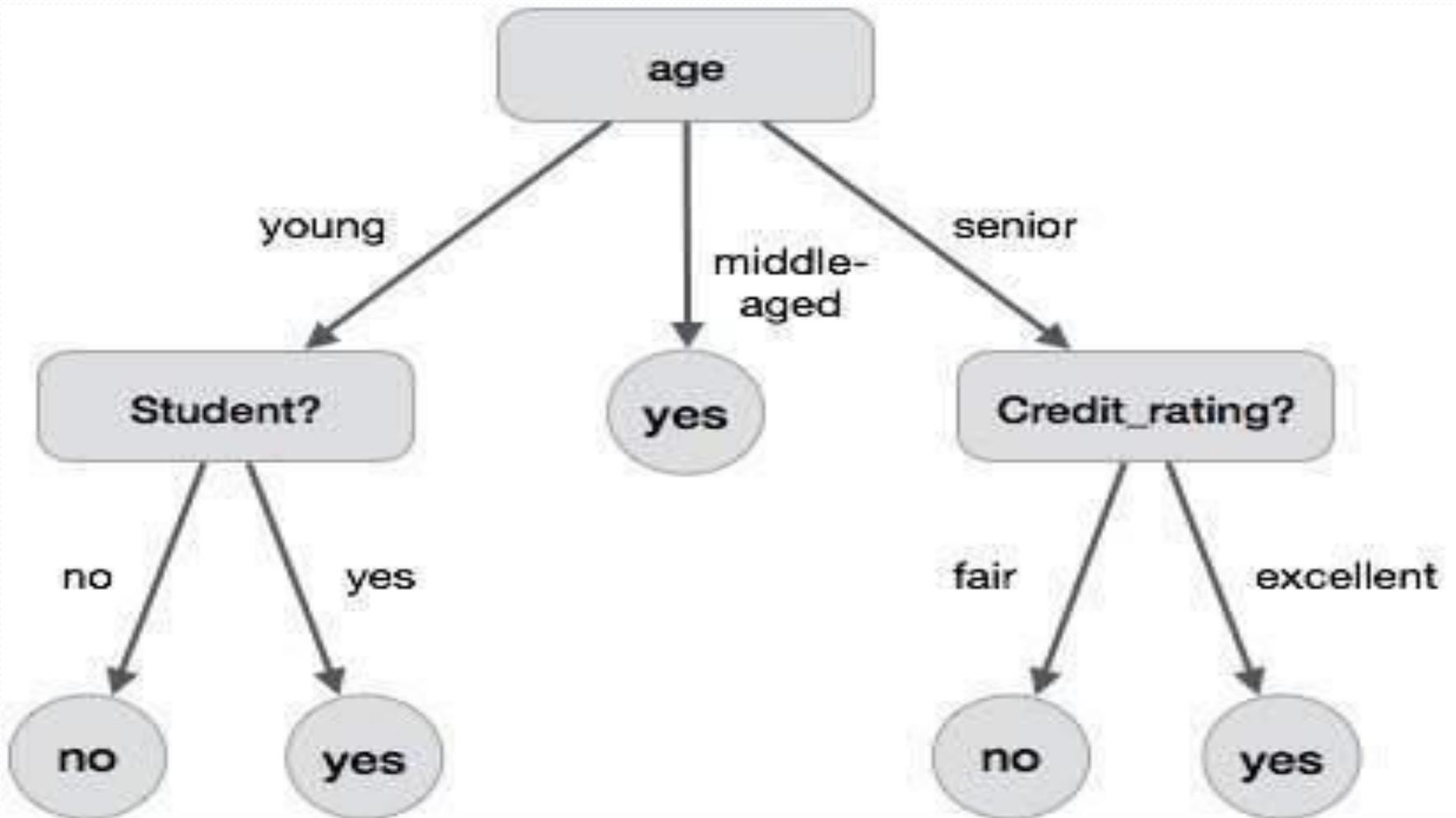
- 
- There are two forms of data analysis that can be used for extracting models describing important classes or to predict future data trends. These two forms are as follows –
  - Classification
  - Prediction
  - Classification models predict categorical class labels; and prediction models predict continuous valued functions.





# Decision Tree

- A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.



- **Algorithm : Generate\_decision\_tree**
- **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 that the data tuples into individual classes. This criterion includes a splitting\_attribute and either a splitting point or splitting subset.
- **Output:** A Decision Tree
- **Method** create a node N;
- if tuples in D are all of the same class, C then return N as leaf node labeled with class C;
- if attribute\_list is empty then return N as leaf node with labeled with majority class in D;  
// majority voting
- apply attribute\_selection\_method(D, attribute\_list) to find the best splitting\_criterion;
- label node N with splitting\_criterion;
- if splitting\_attribute is discrete-valued and multiway splits allowed then // not restricted to binary trees
- attribute\_list -= splitting attribute; // remove splitting attribute for each outcome j of splitting criterion // partition the tuples and grow subtrees for each partition
- let D<sub>j</sub> be the set of data tuples in D satisfying outcome j; // a partition
- if D<sub>j</sub> is empty then
- attach a leaf labeled with the majority class in D to node N;
- else attach the node returned by Generate decision tree(D<sub>j</sub>, attribute list) to node N;
- end for
- return N;

# Attribute selection measures

- Information gain
- Gain ratio
- Gini index

