7BCEE1A – DATA MINING AND DATA WAREHOUSING

UNIT – 1

CHAPTER 1: INTRODUCTION

- **What is a data ware house?**

CHAPTER 2: DELIVERY PROCESS

- **Data warehouse delivery method**

CHAPTER 3: SYSTEM PROCESSES

- **Introduction**
- **Overview**
- **Typical Process Flow within a Data warehouse**
- **Extract and Load Process**
- **Clean and Transform Data**
- **Backup and Archive Process**
- **Query Management Process**

CHAPTER 4: PROCESS ARCHITECTURE

- **Introduction**
- **Load Manager**
- **Warehouse Manager**
- **Query Manager**

CHAPTER 1: INTRODUCTION

<u>What is a Data Warehouse?</u>

- ⇨ A data warehouse is a database, which is kept separate from the organization's operational database.
- ⇨ There is no frequent updating done in a data warehouse.
- ⇨ It possesses consolidated historical data, which helps the organization to analyze its business.
- ⇨ A data warehouse helps executives to organize, understand, and use their data to take strategic decisions.
- ⇨ Data warehouse systems help in the integration of diversity of application systems.
- ⇨ A data warehouse system helps in consolidated historical data analysis.

CHAPTER 2: DELIVERY PROCESS

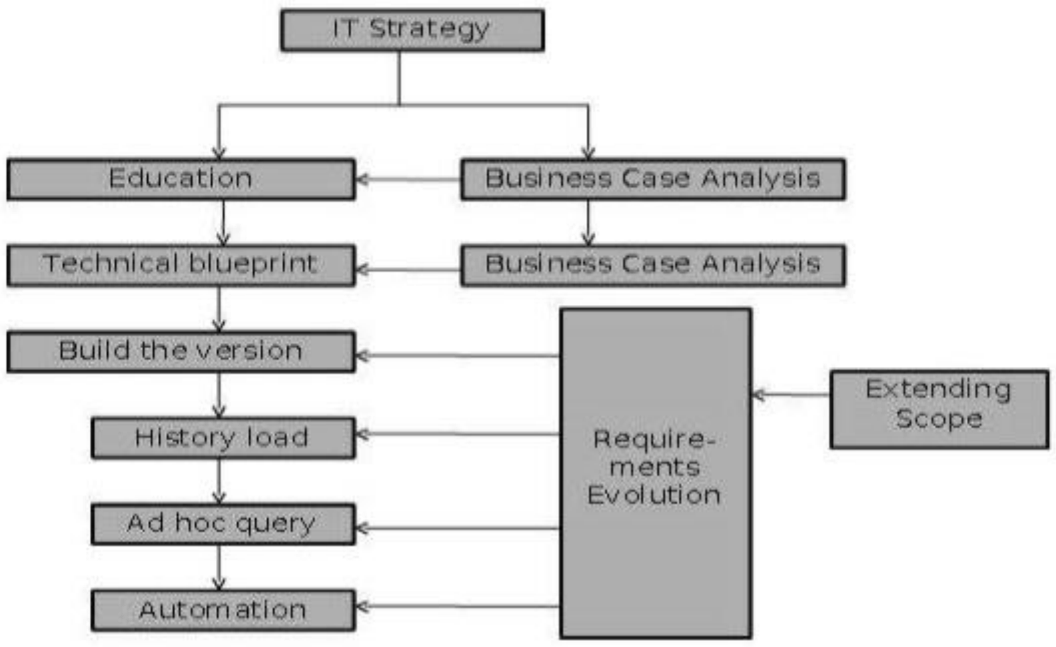
DATA WAREHOSE DELIVERY PROCESS:

- ⇨ A data warehouse is never static; it evolves as the business expands. As the business evolves, its requirements keep changing and therefore a data warehouse must be designed to ride with these changes. Hence a data warehouse system needs to be flexible.
- ⇨ Ideally there should be a delivery process to deliver a data warehouse. However data warehouse projects normally suffer from various issues that make it difficult to complete tasks and deliverables in the strict and ordered fashion demanded by the waterfall method. Most of the times, the requirements are not understood completely. The architectures, designs, and build components can be completed only after gathering and studying all the requirements.

DELIVERY METHOD:

- ⇨ The delivery method is a variant of the joint application development approach adopted for the delivery of a data warehouse. We have staged the data warehouse delivery process to minimize risks. The approach that we will discuss here does not reduce the overall delivery time-scales but ensures the business benefits are delivered incrementally through the development process.

- ⇨ The delivery process is broken into phases to reduce the project and delivery risk.
- ⇨ The following diagram explains the stages in the delivery process –



IT STRATEGY

- ⇨ Data warehouse are strategic investments that require a business process to generate benefits. IT Strategy is required to procure and retain funding for the project.

BUSINESS CASE

- ⇨ The objective of business case is to estimate business benefits that should be derived from using a data warehouse. These benefits may not be quantifiable but the projected benefits need to be clearly stated. If a data warehouse does not have a clear business case, then the business tends to suffer from credibility problems at some stage during the delivery process. Therefore in data warehouse projects, we need to understand the business case for investment.

Education and Prototyping

⇨ Organizations experiment with the concept of data analysis and educate themselves on the value of having a data warehouse before settling for a solution. This is addressed by prototyping. It helps in understanding the feasibility and benefits of a data warehouse. The prototyping activity on a small scale can promote educational process as long as −

- The prototype addresses a defined technical objective.
- The prototype can be thrown away after the feasibility concept has been shown.
- The activity addresses a small subset of eventual data content of the data warehouse.
- The activity timescale is non-critical.

The following points are to be kept in mind to produce an early release and deliver business benefits.

- Identify the architecture that is capable of evolving.
- Focus on business requirements and technical blueprint phases.
- Limit the scope of the first build phase to the minimum that delivers business benefits.
- Understand the short-term and medium-term requirements of the data warehouse.

BUSINESS REQUIREMENTS

- ⇨ To provide quality deliverables, we should make sure the overall requirements are understood. If we understand the business requirements for both short-term and medium-term, then we can design a solution to fulfil short-term requirements. The short-term solution can then be grown to a full solution.
- ⇨ The following aspects are determined in this stage –

- The business rule to be applied on data.
- The logical model for information within the data warehouse.
- The query profiles for the immediate requirement.
- The source systems that provide this data.

TECHNICAL BLUEPRINT:

- ⇨ This phase need to deliver an overall architecture satisfying the long term requirements. This phase also deliver the components that must be implemented in a short term to derive any business benefit. The blueprint need to identify the followings.

  - The overall system architecture.
  - The data retention policy.
  - The backup and recovery strategy.
  - The server and data mart architecture.
  - The capacity plan for hardware and infrastructure.
  - The components of database design.

BUILDING THE VERSION:

- ⇨ In this stage, the first production deliverable is produced. This production deliverable is the smallest component of a data warehouse. This smallest component adds business benefit.

HISTORY LOAD

- ⇨ This is the phase where the remainder of the required history is loaded into the data warehouse. In this phase, we do not add new entities, but additional physical tables would probably be created to store increased data volumes.
- ⇨ Let us take an example. Suppose the build version phase has delivered a retail sales analysis data warehouse with 2 months' worth of history. This information will allow the user to analyze only the recent trends and address the short-term issues. The user in this case cannot identify annual and seasonal trends. To help him do so, last 2 years' sales history could be loaded from the archive. Now the 40GB data is extended to 400GB.
- ⇨ Note – The backup and recovery procedures may become complex, therefore it is recommended to perform this activity within a separate phase.

AD HOC QUERY

- ⇨ In this phase, we configure an ad hoc query tool that is used to operate a data warehouse. These tools can generate the database query.
- ⇨ Note – It is recommended not to use these access tools when the database is being substantially modified.

AUTOMATION

- ⇨ In this phase, operational management processes are fully automated. These would include –

- Transforming the data into a form suitable for analysis.
- Monitoring query profiles and determining appropriate aggregations to maintain system performance.
- Extracting and loading data from different source systems.
- Generating aggregations from predefined definitions within the data warehouse.
- Backing up, restoring, and archiving the data.

EXTENDING SCOPE

- ⇨ In this phase, the data warehouse is extended to address a new set of business requirements. The scope can be extended in two ways,

- By loading additional data into the data warehouse.
- By introducing new data marts using the existing information.

REQUIREMENTS EVOLUTION

- ⇨ From the perspective of delivery process, the requirements are always changeable. They are not static. The delivery process must support this and allow these changes to be reflected within the system.
- ⇨ This issue is addressed by designing the data warehouse around the use of data within business processes, as opposed to the data requirements of existing queries.

- The architecture is designed to change and grow to match the business needs, the process operates as a pseudo-application development process, where the new requirements are continually fed into the development activities and the partial deliverables are produced. These partial deliverables are fed back to the users and then reworked ensuring that the overall system is continually updated to meet the business needs.

CHAPTER 3: SYSTEM PROCESSES

INTRODUCTION

⇨ We have a fixed number of operations to be applied on the operational databases and we have well-defined techniques such as **use normalized data**, **keep table small**, etc. These techniques are suitable for delivering a solution. But in case of decision-support systems, we do not know what query and operation needs to be executed in future. Therefore techniques applied on operational databases are not suitable for data warehouses.

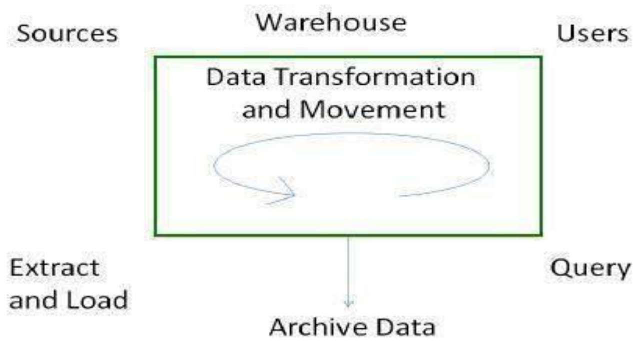
⇨ In this chapter, we will discuss how to build data warehousing solutions on top open-system technologies like Unix and relational databases.

TYPICAL PROCESS FLOW WITHIN A DATA WAREHOUSE:

⇨ There are four major processes that contribute to a data warehouse –

- **Extract and load the data.**
- **Cleaning and transforming the data.**
- **Backup and archive the data.**
- **Managing queries and directing them to the appropriate data sources**.

Typical Process flow within a Data Warehouse



EXTRACT AND LOAD PROCESS

- ⇨ Data extraction takes data from the source systems. Data load takes the extracted data and loads it into the data warehouse.

Controlling the Process

- ⇨ Controlling the process involves determining when to start data extraction and the consistency check on data. Controlling process ensures that the tools, the logic modules, and the programs are executed in correct sequence and at correct time.

When to Initiate Extract

- ⇨ Data needs to be in a consistent state when it is extracted, i.e., the data warehouse should represent a single, consistent version of the information to the user.
- ⇨ For example, in a customer profiling data warehouse in telecommunication sector, it is illogical to merge the list of customers at 8 pm on Wednesday from a customer database with the customer subscription events up to 8 pm on Tuesday. This would mean that we are finding the customers for whom there are no associated subscriptions.

Loading the Data

- ⇨ After extracting the data, it is loaded into a temporary data store where it is cleaned up and made consistent.
- ⇨ **Note** – Consistency checks are executed only when all the data sources have been loaded into the temporary data store.

CLEAN AND TRANSFORM PROCESS

⇨ Once the data is extracted and loaded into the temporary data store, it is time to perform Cleaning and Transforming. Here is the list of steps involved in Cleaning and Transforming –

- Clean and transform the loaded data into a structure
- Partition the data
- Aggregation

Clean and Transform the Loaded Data into a Structure

⇨ Cleaning and transforming the loaded data helps speed up the queries. It can be done by making the data consistent –

- within itself.
- with other data within the same data source.
- with the data in other source systems.
- with the existing data present in the warehouse.

⇨ Transforming involves converting the source data into a structure. Structuring the data increases the query performance and decreases the operational cost. The data contained in a data warehouse must be transformed to support performance requirements and control the ongoing operational costs.

Partition the Data

⇨ It will optimize the hardware performance and simplify the management of data warehouse. Here we partition each fact table into multiple separate partitions.

Aggregation

- ⇨ Aggregation is required to speed up common queries. Aggregation relies on the fact that most common queries will analyze a subset or an aggregation of the detailed data.

BACKUP AND ARCHIVE THE DATA

- ⇨ In order to recover the data in the event of data loss, software failure, or hardware failure, it is necessary to keep regular backups. Archiving involves removing the old data from the system in a format that allow it to be quickly restored whenever required.
- ⇨ For example, in a retail sales analysis data warehouse, it may be required to keep data for 3 years with the latest 6 months data being kept online. In such as scenario, there is often a requirement to be able to do month-on-month comparisons for this year and last year. In this case, we require some data to be restored from the archive.

QUERY MANAGEMENT PROCESS

- ⇨ This process performs the following functions –

- manages the queries.
- helps speed up the execution time of queris.
- directs the queries to their most effective data sources.

- ensures that all the system sources are used in the most effective way.
- monitors actual query profiles.

⇨ The information generated in this process is used by the warehouse management process to determine which aggregations to generate. This process does not generally operate during the regular load of information into data warehouse.

CHAPTER 4: PROCESS ARCHITECTURE

INTRODUCTION:

⇨ **Data Warehouse Architecture** is complex as it's an information system that contains historical and commutative data from multiple sources. There are 3 approaches for constructing Data Warehouse layers: Single Tier, Two tier and Three tier. This 3 tier architecture of Data Warehouse is explained as below.

Single-tier architecture

⇨ The objective of a single layer is to minimize the amount of data stored. This goal is to remove data redundancy. This architecture is not frequently used in practice.

Two-tier architecture

⇨ Two-layer architecture is one of the Data Warehouse layers which separates physically available sources and data warehouse. This architecture is not expandable and also not supporting a large number of end-users. It also has connectivity problems because of network limitations.

Three-Tier Data Warehouse Architecture

⇨ This is the most widely used Architecture of Data Warehouse.
⇨ It consists of the Top, Middle and Bottom Tier.

1. **Bottom Tier:** The database of the Datawarehouse servers as the bottom tier. It is usually a relational database system. Data is

cleansed, transformed, and loaded into this layer using back-end tools.

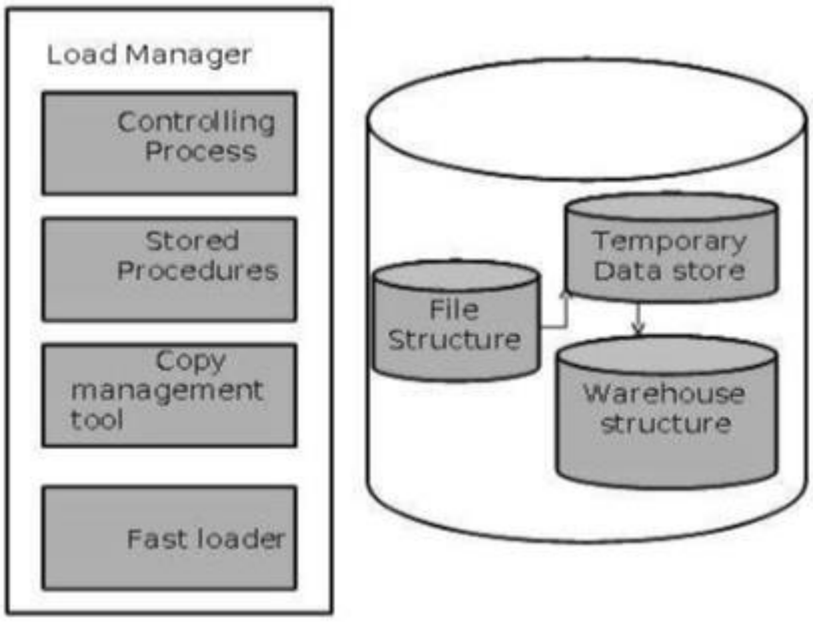
2. **Middle Tier:** The middle tier in Data warehouse is an OLAP server which is implemented using either ROLAP or MOLAP model. For a user, this application tier presents an abstracted view of the database. This layer also acts as a mediator between the end-user and the database.
3. **Top-Tier:** The top tier is a front-end client layer. Top tier is the tools and API that you connect and get data out from the data warehouse. It could be Query tools, reporting tools, managed query tools, Analysis tools and Data mining tools.

LOAD MANAGER:

- ⇨ This component performs the operations required to extract and load process.
- ⇨ The size and complexity of the load manager varies between specific solutions from one data warehouse to other.
- ⇨ The load manager performs the following functions −

- Extract the data from source system.
- Fast Load the extracted data into temporary data store.
- Perform simple transformations into structure similar to the one in the data warehouse.

Load Manager Architecture



Extract Data from Source

⇨ The data is extracted from the operational databases or the external information providers. Gateways is the application programs that are used to extract data. It is supported by underlying DBMS and allows client program to generate SQL to be executed at a server. Open Database Connection(ODBC), Java Database Connection (JDBC), are examples of gateway.

Fast Load

⇨ In order to minimize the total load window the data need to be loaded into the warehouse in the fastest possible time.

⇨ The transformations affects the speed of data processing.

⇨ It is more effective to load the data into relational database prior to applying transformations and checks.

- ⇨ Gateway technology proves to be not suitable, since they tend not be performant when large data volumes are involved.

Simple Transformations

- ⇨ While loading it may be required to perform simple transformations. After this has been completed we are in position to do the complex checks. Suppose we are loading the EPOS sales transaction we need to perform the following checks:

  - Strip out all the columns that are not required within the warehouse.
  - Convert all the values to required data types.

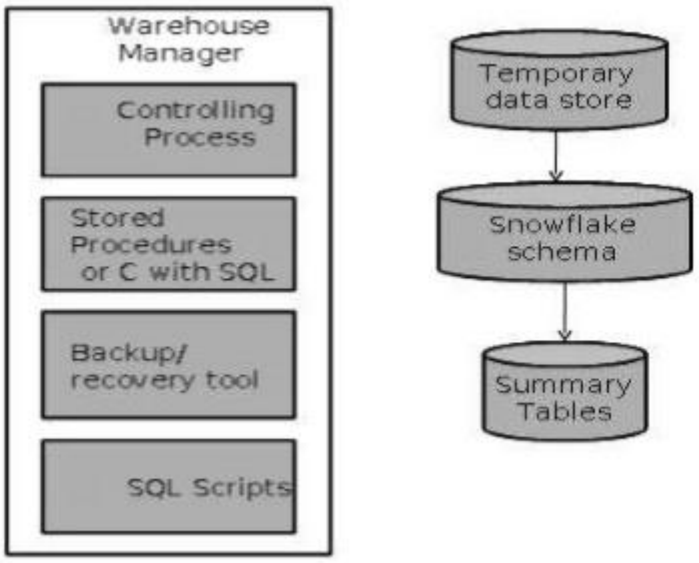
WAREHOUSE MANAGER:

- ⇨ A warehouse manager is responsible for the warehouse management process. It consists of third-party system software, C programs, and shell scripts.
- ⇨ The size and complexity of warehouse managers varies between specific solutions.
- ⇨ **Warehouse Manager Architecture**

  A warehouse manager includes the following –

  - The controlling process
  - Stored procedures or C with SQL
  - Backup/Recovery tool
  - SQL Scripts

Warehouse Manager Architecture



⇨ **Operations Performed by Warehouse Manager**

- A warehouse manager analyzes the data to perform consistency and referential integrity checks.
- Creates indexes, business views, partition views against the base data.
- Generates new aggregations and updates existing aggregations. Generates normalizations.
- Transforms and merges the source data into the published data warehouse.
- Backup the data in the data warehouse.
- Archives the data that has reached the end of its captured life.

QUERY MANAGER:

⇨ **Operations Performed by Query Manager**

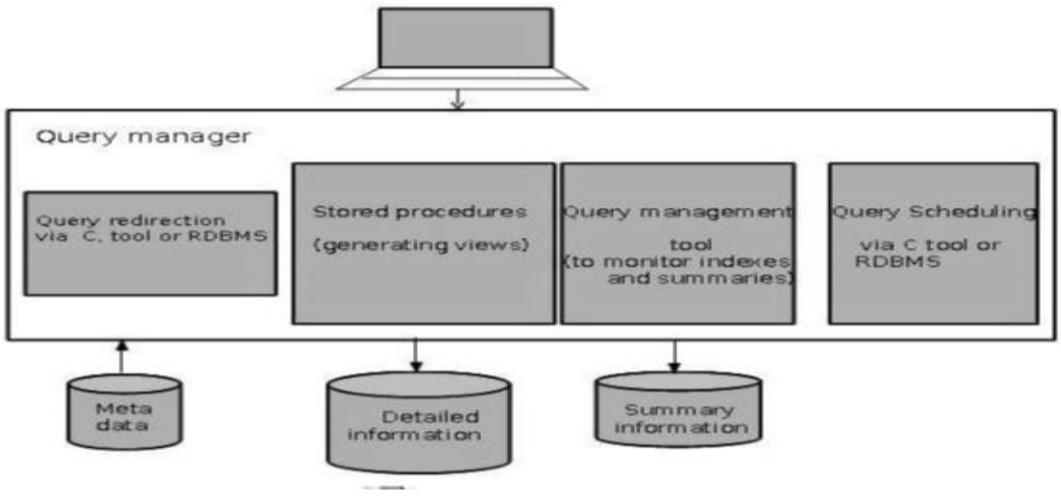
- Query manager is responsible for directing the queries to the suitable tables.
- By directing the queries to appropriate tables, the speed of querying and response generation can be increased.
- Query manager is responsible for scheduling the execution of the queries posed by the user.

⇨ **Query Manager Architecture**

The following screenshot shows the architecture of a query manager. It includes the following:

- Query redirection via C tool or RDBMS
- Stored procedures
- Query management tool
- Query scheduling via C tool or RDBMS
- Query scheduling via third-party software

Query Manager Architecture


Query manager
Query redirection via C. tool or RDBMS
Stored procedures (generating views)
Query management tool (to monitor indexes and summaries)
Query Scheduling via C tool or RDBMS
Meta data
Detailed information
Summary information

7BCEE1A – DATA MINING AND DATA WAREHOUSING

UNIT – 2

CHAPTER 10: SYSTEM AND DATA WAREHOUSE PROCESS MANAGERS

- Introduction
- Why you need tools to Manage a Data Warehouse
- System Managers
- Data Warehouse Process Managers
- Load Manager
- Warehouse Manager
- Query Manager

CHAPTER 17: CAPACITY PLANNING

- Introduction
- Process
- Estimating the Load

CHAPTER 18: TUNING THE DATA WAREHOUSE

- Introduction
- Assessing Performance
- Tuning the Data Load
- Tuning Queries

CHAPTER 10: SYSTEM AND DATA WAREHOUSE PROCESS MANAGERS

INTRODUCTION

- ⇨ The manager will consist of tools and procedures required to manage it area of responsibility.
- ⇨ The **system manager** refers to areas of management of the underlying hardware, operating system or systems themselves, such as scheduling and backup recovery.
- ⇨ The data warehouse **process manager** refers more specifically to the data warehouse itself and the components (load, warehouse, query) of the data warehouse architecture.

WHY YOU NEED TOOLS TO M ANAGE A DATA WAREHOUSE

- ⇨ Data warehouses are not just large databases; they are large, complex environments that integrate many different technologies. As such they require a lot of maintenance and management
- ⇨ The traditional approach of having a large team of administrators to manage a data warehouse arena does not work well. The system usage is generally too ad hoc and unpredictable to be manually administrated. Therefore, intelligent tools are required to help the system and database administrators to do their jobs.
- ⇨ The tools required can be divided into the two categories:
  - o **System management tools**
  - o **Data warehouse process management tools**

SYSTEM MANAGERS:

⇨ System management is mandatory for the successful implementation of a data warehouse. The most important system managers are −

- **System configuration manager**
- **System scheduling manager**
- **System event manager**
- **System database manager**
- **System backup recovery manager**

SYSTEM CONFIGURATION MANAGER:

- The system configuration manager is responsible for the management of the setup and configuration of data warehouse.
- The structure of configuration manager varies from one operating system to another.
- In Unix structure of configuration, the manager varies from vendor to vendor.
- Configuration managers have single user interface.
- The interface of configuration manager allows us to control all aspects of the system.

SYSTEM SCHEDULING MANAGER

  - System Scheduling Manager is responsible for the successful implementation of the data warehouse. Its purpose is to schedule ad hoc queries. Every operating system has its own scheduler with some form of batch control mechanism. The list of features a system scheduling manager must have is as follows –

- Work across cluster or MPP boundaries
- Deal with international time differences
- Handle job failure
- Handle multiple queries
- Support job priorities
- Restart or re-queue the failed jobs
- Notify the user or a process when job is completed
- Maintain the job schedules across system outages
- Re-queue jobs to other queues
- Support the stopping and starting of queues
- Log Queued jobs
- Deal with inter-queue processing

.

- Some important jobs that a scheduler must be able to handle are as follows –
    - Daily and ad hoc query scheduling
    - Execution of regular report requirements
    - Data load
    - Data processing
    - Index creation
    - Backup

- Aggregation creation
- Data transformation

SYSTEM EVENT MANAGER

⇨ The event manager is a kind of a software. The event manager manages the events that are defined on the data warehouse system. We cannot manage the data warehouse manually because the structure of data warehouse is very complex. Therefore we need a tool that automatically handles all the events without any intervention of the user.

Events

⇨ Events are the actions that are generated by the user or the system itself. It may be noted that the event is a measurable, observable, occurrence of a defined action.

⇨ Given below is a list of common events that are required to be tracked.

- Hardware failure
- Running out of space on certain key disks
- A process dying
- A process returning an error
- CPU usage exceeding an 805 threshold
- Internal contention on database serialization points
- Buffer cache hit ratios exceeding or failure below threshold
- A table reaching to maximum of its size
- Excessive memory swapping

- A table failing to extend due to lack of space
- Disk exhibiting I/O bottlenecks
- Usage of temporary or sort area reaching a certain thresholds
- Any other database shared memory usage

⇨ The most important thing about events is that they should be capable of executing on their own. Event packages define the procedures for the predefined events. The code associated with each event is known as event handler. This code is executed whenever an event occurs.

SYSTEM AND DATABASE MANAGER:

⇨ System and database manager may be two separate pieces of software, but they do the same job. The objective of these tools is to automate certain processes and to simplify the execution of others. The criteria for choosing a system and the database manager are as follows –

- increase user's quota.
- assign and de-assign roles to the users
- assign and de-assign the profiles to the users
- perform database space management
- monitor and report on space usage
- tidy up fragmented and unused space
- add and expand the space
- add and remove users

- manage user password
- manage summary or temporary tables
- assign or deassign temporary space to and from the user
- reclaim the space form old or out-of-date temporary tables
- manage error and trace logs
- to browse log and trace files
- redirect error or trace information
- switch on and off error and trace logging
- perform system space management
- monitor and report on space usage
- clean up old and unused file directories
- add or expand space.

SYSTEM BACKUP RECOVERY MANAGER:

⇨ The backup and recovery tool makes it easy for operations and management staff to back-up the data. Note that the system backup manager must be integrated with the schedule manager software being used. The important features that are required for the management of backups are as follows –

- Scheduling
- Backup data tracking
- Database awareness

⇨ Backups are taken only to protect against data loss. Following are the important points to remember –

- The backup software will keep some form of database of where and when the piece of data was backed up.
- The backup recovery manager must have a good front-end to that database.
- The backup recovery software should be database aware.
- Being aware of the database, the software then can be addressed in database terms, and will not perform backups that would not be viable.

PROCESS MANAGERS:

⇨ Process managers are responsible for maintaining the flow of data both into and out of the data warehouse. There are three different types of process managers −

  - Load manager
  - Warehouse manager
  - Query manager

DATA WAREHOUSE LOAD MANAGER

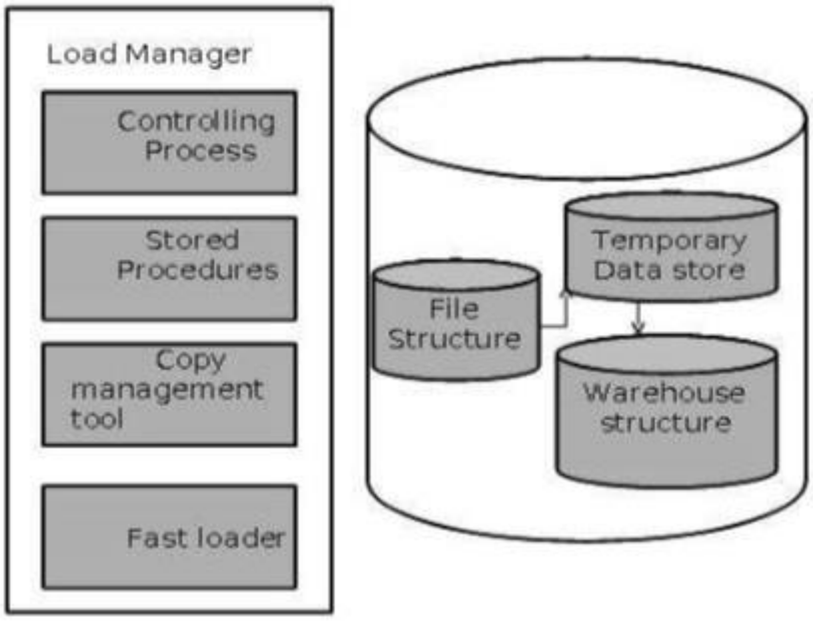
⇨ Load manager performs the operations required to extract and load the data into the database. The size and complexity of a load manager varies between specific solutions from one data warehouse to another.

Load Manager Architecture

⇨ The load manager does performs the following functions −

- Extract data from the source system.

- Fast load the extracted data into temporary data store.
- Perform simple transformations into structure similar to the one in the data warehouse.



Extract Data from Source

The data is extracted from the operational databases or the external information providers. Gateways are the application programs that are used to extract data. It is supported by underlying DBMS and allows the client program to generate SQL to be executed at a server. Open Database Connection (ODBC) and Java Database Connection (JDBC) are examples of gateway.

Fast Load

- In order to minimize the total load window, the data needs to be loaded into the warehouse in the fastest possible time.
- Transformations affect the speed of data processing.

- It is more effective to load the data into a relational database prior to applying transformations and checks.
- Gateway technology is not suitable, since they are inefficient when large data volumes are involved.

Simple Transformations

While loading, it may be required to perform simple transformations. After completing simple transformations, we can do complex checks. Suppose we are loading the EPOS sales transaction, we need to perform the following checks –

- Strip out all the columns that are not required within the warehouse.
- Convert all the values to required data types.

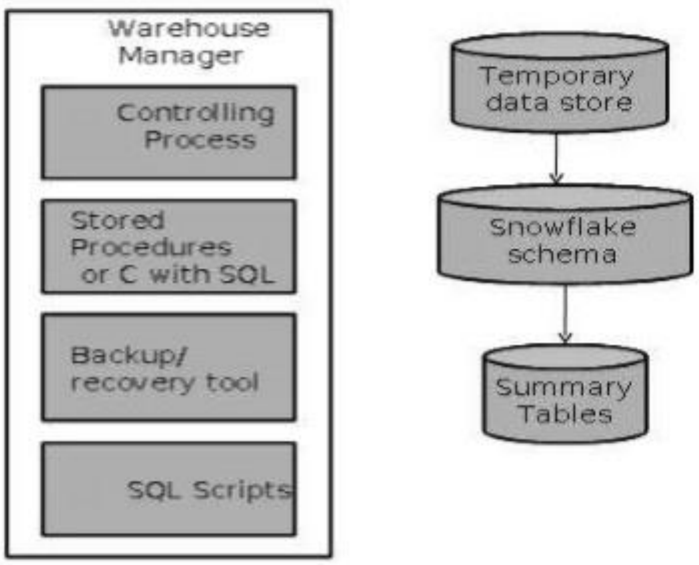
<u>WAREHOUSE MANAGER</u>:

⇨ The warehouse manager is responsible for the warehouse management process. It consists of a third-party system software, C programs, and shell scripts. The size and complexity of a warehouse manager varies between specific solutions.

⇨ **Warehouse Manager Architecture**

A warehouse manager includes the following –

- The controlling process
- Stored procedures or C with SQL
- Backup/Recovery tool
- SQL scripts



Functions of Warehouse Manager

A warehouse manager performs the following functions −

- Analyzes the data to perform consistency and referential integrity checks.
- Creates indexes, business views, partition views against the base data.
- Generates new aggregations and updates the existing aggregations.
- Generates normalizations.
- Transforms and merges the source data of the temporary store into the published data warehouse.
- Backs up the data in the data warehouse.
- Archives the data that has reached the end of its captured life.

.

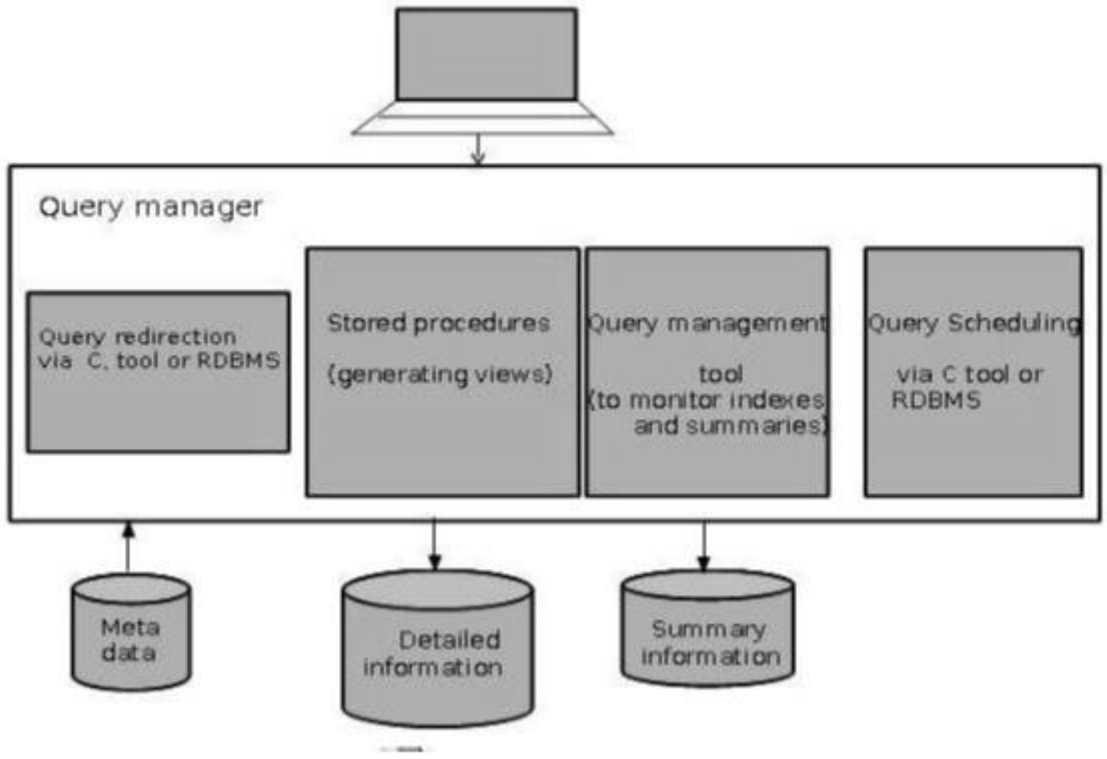
QUERY MANAGER:

⇨ The query manager is responsible for directing the queries to suitable tables. By directing the queries to appropriate tables, it speeds up the query request and response process. In addition, the query manager is responsible for scheduling the execution of the queries posted by the user.

⇨ **Query Manager Architecture**

A query manager includes the following components –

- Query redirection via C tool or RDBMS
- Stored procedures
- Query management tool
- Query scheduling via C tool or RDBMS
- Query scheduling via third-party software


Query manager
Query redirection via C. tool or RDBMS
Stored procedures (generating views)
Query management tool (to monitor indexes and summaries)
Query Scheduling via C tool or RDBMS
Meta data
Detailed information
Summary information

Functions of Query Manager

- It presents the data to the user in a form they understand.
- It schedules the execution of the queries posted by the end-user.
- It stores query profiles to allow the warehouse manager to determine which indexes and aggregations are appropriate.

CHAPTER 17: CAPACITY PLANNING

INTRODUCTION

- ⇨ Any data warehouse solution will grow over time sometimes quite dramatically
- ⇨ It is essential that the components of the solution (hardware, software, and database) are capable of supporting the extended sizes without unacceptable performance loss, or growth of the load window to a point where it affects the use of the system
- ⇨ The capacity plan for a data warehouse is defined within the technical blueprint stage of the process
- ⇨ The business requirements stage should have identified the approximate sizes for data, users, and any other issues that constrain system performance

PROCESS

- ⇨ It have a clear understanding of the usage profiles of all users of the data warehouse like
  - ✓ the number of users in the group;
  - ✓ whether they use ad hoc queries frequently;
  - ✓ whether they use ad hoc queries occasionally at unknown intervals
  - ✓ whether they use ad hoc queries occasionally at regular and predictable times;

- ✓ the average size of query they tend to run;
- ✓ the maximum size of query they tend to run;
- ✓ the elapsed login time per day;
- ✓ the peak time of daily usage;
- ✓ the number of queries they run per peak hour;
- ✓ the number of queries they run per day

⇨ Usage profiles:

- ✓ change over time, and need to be kept up to date
- ✓ useful for growth predictions and capacity planning
- ✓ require an understanding of the business

ESTIMATING THE LOAD

- ⇨ Factors for choosing the hardware: hardware architecture, resilience (flexibility)
- ⇨ consideration of the ultimate size of the data warehouse
- ⇨ extremely costly to change hardware in midstream
- ⇨ different elements that need to be considered like o how much CPU.
- ⇨ how much memory and how much disk
- ⇨ The cost should never be allowed to affect capacity planning

- ⇨ A small growth in requirements in a data warehouse environment can mean adding enough hardware to run a full-sized operational OLTP system

  Example: adding a single byte to a field in the fact data meant adding 4GB of storage for each year's worth of online data.

RULES / STEPS IN ESTIMATING THE LOAD

- ▪ Initial Configuration
- ▪ How Much CPU Bandwidth
- ▪ How Much Memory
- ▪ How Much Disk

INITIAL CONFIGURATION

- ⇨ will not have history information or statistics.
- ⇨ Need to be done on the predicted load on difficult due to an adhoc element.
- ⇨ Phased approach: adhoc element will be small to start off, functionality is phased in
- ⇨ Estimate the configuration based on the known requirements
- ⇨ Need to allow some contingency

HOW MUCH CPU BANDWIDTH

- ⇨ The load can be divided into two distinct phases:
  - **Daily processing**
    - User query processing
  - **Overnight processing**
    - Data transformation and load
    - aggregation and index creation
    - Backup

DAILY PROCESSING

- ⇨ Estimate the time that each query will take
- ⇨ Estimate the size of the largest likely common query

  Example: users will want to query the most recent week or month’s worth of data.

  - ⇨ Know the volume of data that will be involved.
- ⇨ Need to know I/O characteristics of the devices that the data will reside on.
- ⇨ Using S (scan rate) and F (volume of data) you can calculate T, the time in seconds to perform a full table scan as T = F / S
- ⇨ depending on the degree of parallelism, we get o T1 = F / S1, …, Tn = F / Sn

  - where S1 is the scan speed of a single disk or striped disk set, and Sn is the scan speed of all the disksor disk sets

- take the query response time requirements specified in the service level agreement and pick the appropriate T value, Tp this will give you
  - Sp - required scan rate
  - the number of disks or disk sets,
  - P, the required degree of parallelism
- need to estimate Pc, the number of parallel scanning threads that a single CPU will support
- you can estimate your CPU requirement to support a single query with o need to use 2P to allow for other query overheads, sorts etc

$$C_s = \text{Roundup}\left(\frac{2P}{P_s}\right)$$

- To calculate the minimum number of CPUs required overall use the formula

$$C_t = nC_s + 1$$

  - where n is the number of concurrent queries
  - additional 1 is added to allow for the operating system overheads and for all other user processing.

OVERNIGHT PROCESSING

- ⇨ operations like data transformation and load, aggregation and index creation and backup are serialized
- ⇨ they tend to run one at a time, one after the other

DATA TRANSFORMATION

- ⇨ depend on the amount of data processing that is involved
- ⇨ will require more CPU bandwidth than the aggregation and index creation operations when there is an enormous amount of data transformation

BACKUP

- ⇨ will fit into the CPU bandwidth required by the aggregation and index creation
- ⇨ backup use of many parallel streams to speed up
- ⇨ the amount of parallelism will come down for longer backups, and its CPU bandwidth requirement will drop

DATA LOAD

- ⇨ can use massive parallelism to speed up its operation
- ⇨ if you use fewer parallel streams it will use less CPU bandwidth and will take longer to run
- ⇨ likely to fit into whatever CPU capacity is required for the aggregation and index creation
- ⇨ post processing to be carried out on the loaded data

HOW MUCH MEMORY

- ⇨ need to estimate is the minimum requirement
- ⇨ things that affect the amount of memory required are as follows

DATABASE REQUIREMENTS

- ⇨ Database will need memory to cache data blocks as they are used o to cache parsed SQL statements.
- ⇨ These requirements will vary from RDBMS to RDBMS,
- ⇨ Memory for sort space
  - o Each process that performs a sort will require an amount of sort area o parallel technology, can have multiple concurrent queries running

HOW MUCH DISK

- ⇨ Data base requirements
  - o Administration
    - ▪ Fact and Dimension data
    - ▪ Aggregation and Index data
- ⇨ Non Data base requirements
  - o Operating system requirements
  - o Other software requirements
  - o Data warehouse requirements
  - o User requirements

DATA BASE SIZING

- ⇨ ASPECTS TO THE DATABASE SIZING
    - System administration requirements
    - Fact and dimension data
    - Aggregations
    - Indexes
- ⇨ SYSTEM ADMINISTRATION REQUIREMENTS
    - data dictionary
    - journal files
    - rollback space
    - temporary requirements

      Temporary space T=(2n + 1) P

        - where n is the number of concurrent queries allowed, and P is the size of a partition
- ⇨ FACT AND DIMENSION DATA
    - Data that you can actually size
    - Do not attempt to seize the database until the database schema is complete
    - Allow for period variations when sizing partitions
    - When calculating the actual size you will need to know
    - the average size of the data in each field;

- the percentage occupancy of each field;
- the position of each data field in the table;
- the RDBMS storage format of each data type;
- Any table, row and column overheads.

⇨ OTHER FACTORS

- The percentage occupancy of each field is also important
- even empty fields can still take up space in a database record
- database block or page size
- difference between using a 2kB block size and using a 16 kB block size
- size of the index space required for the fact and dimension data
- amount of data that you intend to keep online

⇨ AGGREGATIONS

- allow the same amount of space for aggregations as you will have fact data online

⇨ INDEXES

- allow as much space again for indexing as for the aggregates themselves

⇨ Space required by the database will be

$$\begin{aligned}
\text{Space required} &= F + F_i + D + D_i + A + A_i + T + S \\
&= 3F + F_i + D + D_i + T + S && \text{as } A = A_i = F \\
&< 3.3F + D + D_i + T + S && \text{as } F_i \leqslant 30\%\ F \\
&< 3.5F + T + S && \text{as } D \leqslant 10\%\ F \text{ and } D \approx D_i \\
&\leqslant 3.5F + T && \text{as } S \ll T \text{ and } S \ll F \quad (17.7)
\end{aligned}$$

- F is the size of the fact data
- Fi is the size of the fact data indexation;
- D is the size of the dimension data;
- Di is the size of the dimension data indexation;
- A is the size of the aggregations;
- Ai is the size of the aggregations indexation;
- T is the size of the database temporary or sort area
- S is the database system administration overhead

NON-DATABASE SIZING

⇨ a take the server environment into account

⇨ in cluster or MPP environment, allow space on each node

⇨ example, the operating system will need to be installed on each node

⇨ make allowance for swap space,

⇨ leave enough room for system crash dumps

⇨ other software that has disk space requirements

⇨ RDBMS software, any other third-party tools

⇨ room for the log and trace files

- space for all the code and tools
- log and trace space for each of the managers: load, warehouse, and query
- The load manager need disk space allocated for the source files
- The users will require disk space
- In client-server mode of connection, space to create data extracts or reports
- in direct connection, users may also want directories to save SQL queries and other work

CHAPTER 18: TUNING THE DATA WAREHOUSE

INTRODUCTION

⇨ A data warehouse keeps evolving and it is unpredictable what query the user is going to post in the future. Therefore it becomes more difficult to tune a data warehouse system. In this chapter, we will discuss how to tune the different aspects of a data warehouse such as performance, data load, queries, etc.

Difficulties in Data Warehouse Tuning

⇨ Tuning a data warehouse is a difficult procedure due to following reasons –

- Data warehouse is dynamic; it never remains constant.
- It is very difficult to predict what query the user is going to post in the future.
- Business requirements change with time.
- Users and their profiles keep changing.
- The user can switch from one group to another.
- The data load on the warehouse also changes with time.

ASSESSING PERFORMANCE

⇨ Here is a list of objective measures of performance –

- Average query response time
- Scan rates
- Time used per day query

- Memory usage per process
- I/O throughput rates

⇨ Following are the points to remember.

- It is necessary to specify the measures in service level agreement (SLA).
- It is of no use trying to tune response time, if they are already better than those required.
- It is essential to have realistic expectations while making performance assessment.
- It is also essential that the users have feasible expectations.
- To hide the complexity of the system from the user, aggregations and views should be used.
- It is also possible that the user can write a query you had not tuned for.

TUNING THE DATA LOAD

⇨ Data load is a critical part of overnight processing. Nothing else can run until data load is complete. This is the entry point into the system.

⇨ There are various approaches of tuning data load that are discussed below –

- The very common approach is to insert data using the **SQL Layer**. In this approach, normal checks and constraints need to be performed. When the data is inserted into the table, the code will run to check for enough space to insert the data. If sufficient space is not available, then more space may have to be allocated to these tables. These checks take time to perform and are costly to CPU.

- The second approach is to bypass all these checks and constraints and place the data directly into the preformatted blocks. These blocks are later written to the database. It is faster than the first approach, but it can work only with whole blocks of data. This can lead to some space wastage.
- The third approach is that while loading the data into the table that already contains the table, we can maintain indexes.
- The fourth approach says that to load the data in tables that already contain data, **drop the indexes & recreate them** when the data load is complete. The choice between the third and the fourth approach depends on how much data is already loaded and how many indexes need to be rebuilt.

Integrity Checks

⇨ Integrity checking highly affects the performance of the load. Following are the points to remember –

- Integrity checks need to be limited because they require heavy processing power.
- Integrity checks should be applied on the source system to avoid performance degrade of data load.

TUNING QUERIES

⇨ We have two kinds of queries in data warehouse –

- Fixed queries
- Ad hoc queries

Fixe Queries

⇨ Fixed queries are well defined. Following are the examples of fixed queries –

- regular reports
- Canned queries
- Common aggregations

⇨ Tuning the fixed queries in a data warehouse is same as in a relational database system. The only difference is that the amount of data to be queried may be different. It is good to store the most successful execution plan while testing fixed queries. Storing these executing plan will allow us to spot changing data size and data skew, as it will cause the execution plan to change.

Ad Hoc Queries

⇨ To understand ad hoc queries, it is important to know the ad hoc users of the data warehouse. For each user or group of users, you need to know the following –

- The number of users in the group
- Whether they use ad hoc queries at regular intervals of time
- Whether they use ad hoc queries frequently

- Whether they use ad hoc queries occasionally at unknown intervals.
- The maximum size of query they tend to run
- The average size of query they tend to run
- Whether they require drill-down access to the base data
- The elapsed login time per day
- The peak time of daily usage
- The number of queries they run per peak hour
- It is important to track the user's profiles and identify the queries that are run on a regular basis.
- It is also important that the tuning performed does not affect the performance.
- Identify similar and ad hoc queries that are frequently run.
- If these queries are identified, then the database will change and new indexes can be added for those queries.
- If these queries are identified, then new aggregations can be created specifically for those queries that would result in their efficient execution.

7BCEE1A – DATA MINING AND DATA WAREHOUSING

UNIT – 3

CHAPTER 1: INTRODUCTION

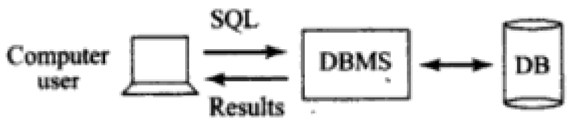
- **Introduction**
- **Basics of Data Mining**
- **Data Mining versus Knowledge Discovery in database**
- **Data Mining Issues**
- **Data Mining Metrics**
- **Social Implications of Data Mining**
- **Data Mining from a Database Perspective**

CHAPTER 1: INTRODUCTION

Introduction:

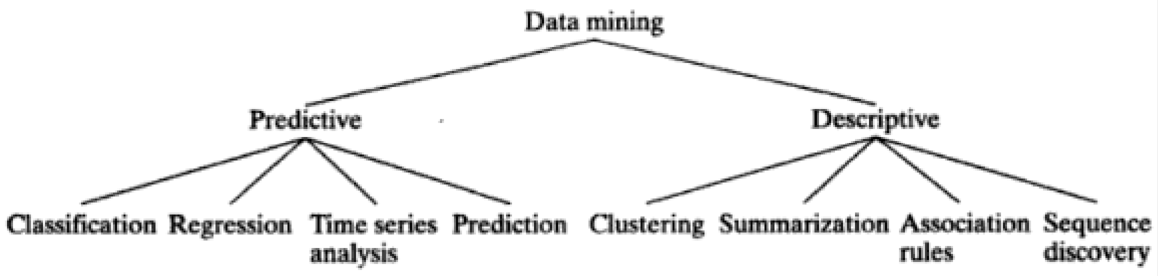
- ⇨ Data mining is defined as finding hidden information in a database. Alternatively, it has been called **exploratory data analysis, data driven discovery,** and **deductive learning.**
- ⇨ Traditional data base queries access a database using a well defined query stated in a language such as SQL. The output of the query consists of the data from the database that satisfies the query. The output is usually a subset of the database.

Database Access



- ⇨ Data Mining access of database differs from this traditional access in several ways,
  - Query
  - Data
  - Output
- ⇨ Data mining involves many different algorithms to accomplish different tasks. These algorithms can be characterized as consisting of three parts:
  - Model
  - Preference
  - Search

Data Mining Models and Tasks



Predictive Model:

It makes a **prediction** about values of data using known results found from different data.

Descriptive Model:

It identifies the **Patterns** or **relationships** in data.

Basic Data Mining Tasks:

Classification:

Classification maps data into predefined **groups** or **classes.** It is often referred to as **Supervised learning** because classes are determined before examining the data.

Example:

An airport security screening station is used to determine: if passengers are potential terrorists or criminals. To do this, the face of each passenger is scanned and its basic pattern (distance between eyes, size and shape of mouth, shape of head, etc.) is identified.

Regression:

Regression is used to map a data into a **real valued prediction variable.** In actuality, regression involves the learning of **function** that does this mapping.

Example:

A college professor wishes to reach a certain level of savings before

her retirement. Periodically, she predicts what her retirement savings will be based on its current value and several past values. She uses a simple linear regression formula to predict this value by fitting past behavior to a linear function and then using this function to predict the values at points in the future. Based on these values, she then alters her investment portfolio.

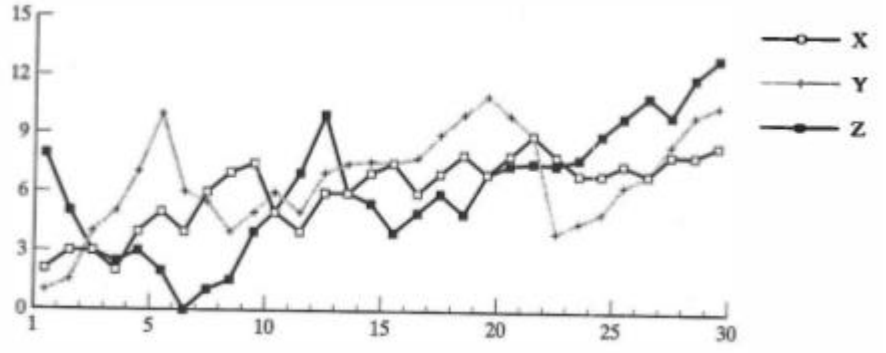
Time Series Analysis:

With time series analysis, the value of an attribute is examined as it varies over time. The values usually are obtained as evenly spaced time points **daily, weekly, hourly…**

Example:

Mr. Smith is trying to determine whether to purchase stock from Companies X, Y, or z. For a period of one month he charts the daily stock price for each company. The following figure shows the time series plot that Mr. Smith has generated. Using this and similar information available from his stockbroker, Mr. Smith decides to purchase stock X because it is less volatile while overall showing a slightly larger relative amount of growth than either of the other stocks. As a matter of fact, the stocks X or Y and Z have a similar behavior. The behavior of Y between days 6 and 20 is identical to that for Z between days 13 and 27.

Time series plot



Prediction:

Prediction can be viewed as a type of classification. But the difference is that prediction is predicting a future state based on past and current data. Prediction applications include, **flooding, speech recognition, machine learning, pattern recognition.**

Example:

Predicting flooding is a difficult problem. One approach uses monitors placed at various; points in the river. These monitors collect data relevant to flood prediction: water level, ' rain amount, time, humidity, and so on. Then the water level at a potential flooding point in the river can be predicted based on the data collected by the sensors upriver from this point. The prediction must be made with respect to the time the data were collected.

Clustering:

Clustering is similar to classification except that the groups are **not predefined**, but rather defined by the **data alone.** Clustering is alternatively referred to as **unsupervised learning** or **segmentation.**

Example:

A certain national department store chain creates special catalogs targeted to various demographic groups based on attributes such as income, location, and physical characteristics of potential customers (age, height, weight, etc.). To determine the target mailings of the various catalogs and to assist in the creation of new, more specific catalogs, the company performs a clustering of potential customers based on the determined attribute values. The results of the clustering exercise are then used by management to create special catalogs and distribute them to the correct target population based on the cluster for that catalog.

Summarization:

Summarization maps data into subsets with associated simple descriptions, Summarization is also called **characterization** or **generalization.** It extracts or derives representative information about the database.

Example:

One of the many criteria used to compare universities by the U.S. News & World Report is the average SAT or ACTS score. This is a summarization used to estimate the type and intellectual level of the student body.

Association Rules:

An association rule is a model that identifies specific types of data associations; these associations are often used in the retail sales community to identify items that are **frequently purchased together**.

Example:

A grocery store retailer is trying to decide whether to put bread on sale. To help determine the impact of this decision, the retailer generates association rules that show what other products are frequently purchased with bread. He finds that 60% of the time that bread is sold so is pretzels and that 70% of the time jelly is also sold. Based on these facts, he tries to capitalize on the association between bread, pretzels, and jelly by placing some pretzels and jelly at the end of the aisle where the bread is placed. In addition, he decides not to place either of these items on sale at the same time.

Sequence Discovery:

Sequential analysis or sequence discovery is used to determine sequential patterns in data. These patterns are based on a time sequence of actions. These patterns are similar to associations in that data are found to be related but the relationship is based on time.

Example:

The Webmaster at the XYZ Corp. periodically analyzes the Web log data to determine how users of the XYZ's Web pages access them. He is interested in determining what sequences of pages are frequently accessed. He determines that 70 percent of the users of page A follow one of the following patterns of behavior: (A, B, C) or (A, D, B, C) or (A, E, B, C). He then determines to add a link directly from page A to page C.

Data Mining versus Knowledge Discovery in Database:

KDD – Knowledge Discovery in Database:

It is the process of finding useful information and patterns in data.

Data Mining:

It is the use of algorithms to extract the information and patterns derived by the KDD process.

KDD is a process that involves many different steps.

- ⇨ **Input of this process :** data
- ⇨ **Output of this process:** useful information desired by the users.

The KDD process consists of the following**:**

1. **Data Cleaning**: Data cleaning is defined as removal of noisy and irrelevant data from collection.

- Cleaning in case of **Missing values**.
- Cleaning **noisy** data, where noise is a random or variance error.
- Cleaning with **Data discrepancy detection** and **Data transformation tools**.

2. **Data Integration**: Data integration is defined as heterogeneous data from multiple sources combined in a common source(DataWarehouse).
    - Data integration using **Data Migration tools**.
    - Data integration using **Data Synchronization tools**.
    - Data integration using **ETL**(Extract-Load-Transformation) process.

3. **Data Selection**: Data selection is defined as the process where data relevant to the analysis is decided and retrieved from the data collection.
    - Data selection using **Neural network**.
    - Data selection using **Decision Trees**.
    - Data selection using **Naive bayes**.
    - Data selection using **Clustering**, **Regression**, etc.

4. **Data Transformation**: Data Transformation is defined as the process of transforming data into appropriate form required by mining procedure.

    Data Transformation is a two step process:

    - **Data Mapping**: Assigning elements from source base to destination to capture transformations.

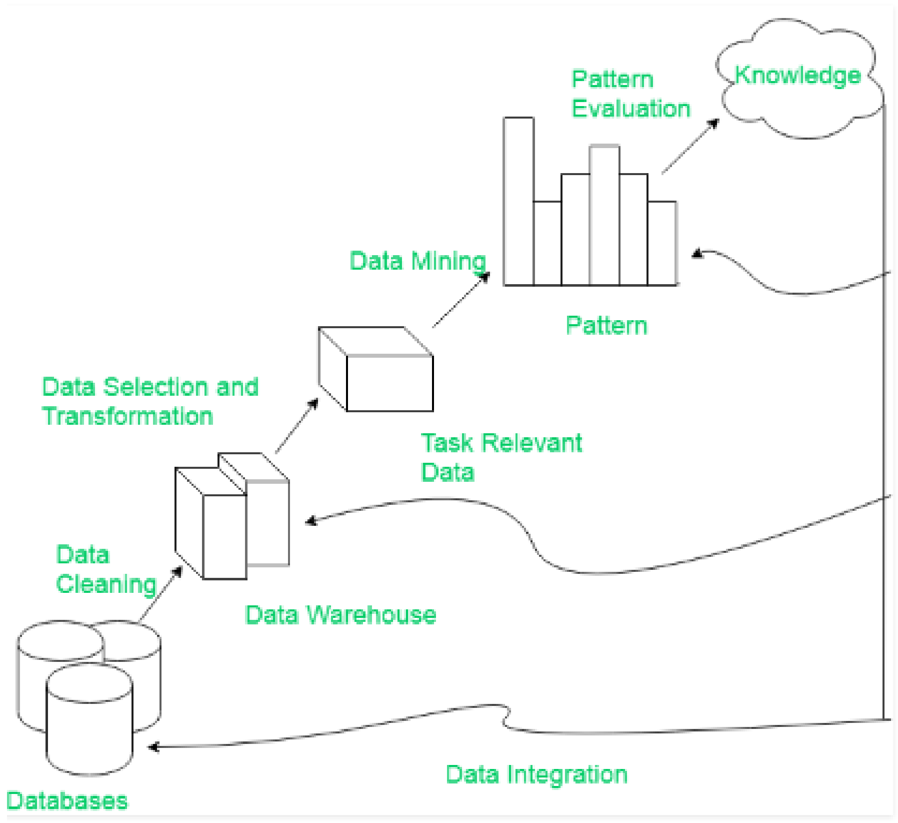
- **Code generation**: Creation of the actual transformation program.

5. **Data Mining**: Data mining is defined as clever techniques that are applied to extract patterns potentially useful.
    - Transforms task relevant data into **patterns**.
    - Decides purpose of model using **classification** or **characterization**.

6. **Pattern Evaluation**: Pattern Evaluation is defined as as identifying strictly increasing patterns representing knowledge based on given measures.
    - Find **interestingness score** of each pattern.
    - Uses **summarization** and **Visualization** to make data understandable by user.

7. **Knowledge representation**: Knowledge representation is defined as technique which utilizes visualization tools to represent data mining results.
    - Generate **reports**.
    - Generate **tables**.
    - Generate **discriminant rules**, **classification rules**, **characterization rules**, etc.

Steps involved in KDD process

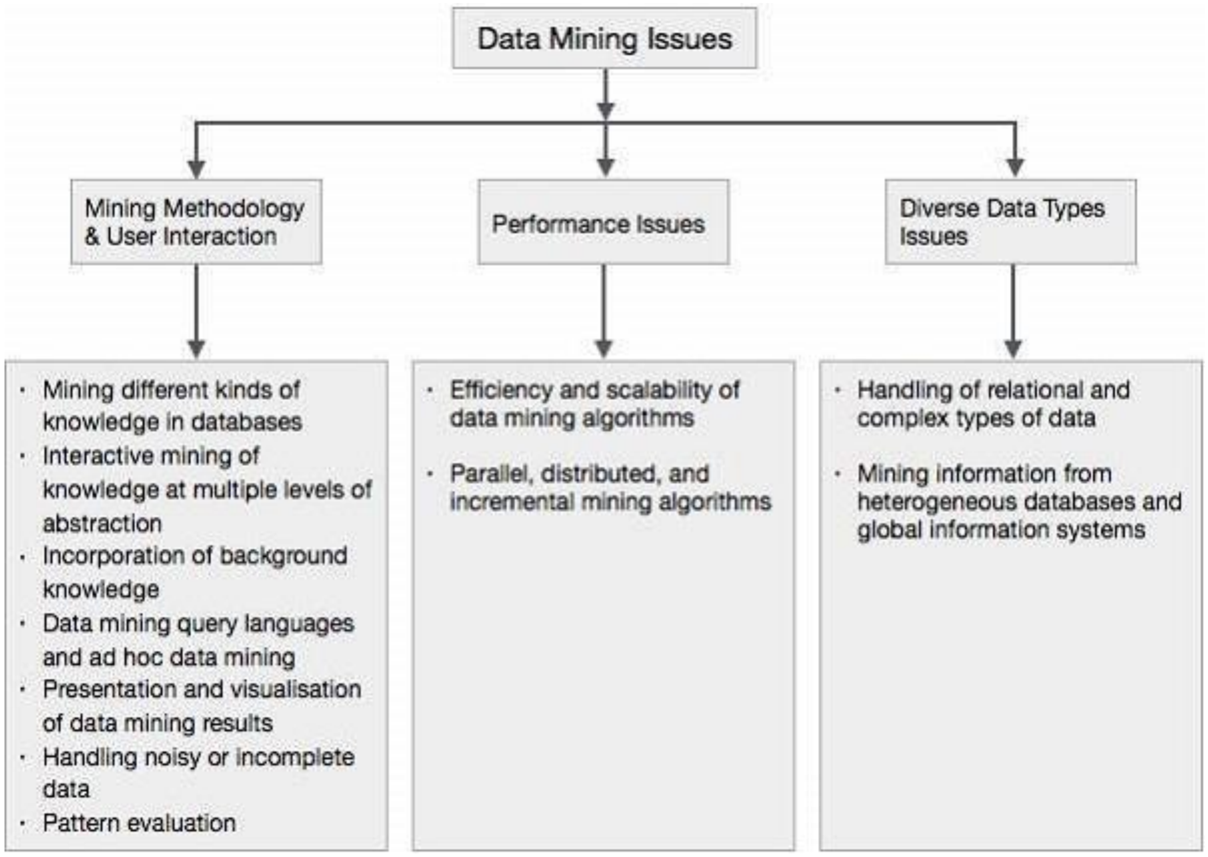


Data Mining Issues:

Data mining is not an easy task, as the algorithms used can get very complex and data is not always available at one place. It needs to be integrated from various heterogeneous data sources. These factors also create some issues. Here in this tutorial, we will discuss the major issues regarding –

- **Mining Methodology and User Interaction**
- **Performance Issues**
- **Diverse Data Types Issues**

The following diagram describes the major issues.


Data Mining Issues
Mining Methodology & User Interaction
Performance Issues
Diverse Data Types Issues
Mining different kinds of knowledge in databases
Interactive mining of knowledge at multiple levels of abstraction
Incorporation of background knowledge
Data mining query languages and ad hoc data mining
Presentation and visualisation of data mining results
Handling noisy or incomplete data
Pattern evaluation
Efficiency and scalability of data mining algorithms
Parallel, distributed, and incremental mining algorithms
Handling of relational and complex types of data
Mining information from heterogeneous databases and global information systems

Mining Methodology and User Interaction Issues

It refers to the following kinds of issues –

- **Mining different kinds of knowledge in databases** – Different users may be interested in different kinds of knowledge. Therefore it is necessary for data mining to cover a broad range of knowledge discovery task.
- **Interactive mining of knowledge at multiple levels of abstraction** – The data mining process needs to be interactive because it allows users to focus the search for patterns, providing and refining data mining requests based on the returned results.
- **Incorporation of background knowledge** – To guide discovery process and to express the discovered patterns, the background

knowledge can be used. Background knowledge may be used to express the discovered patterns not only in concise terms but at multiple levels of abstraction.

- **Data mining query languages and ad hoc data mining** – Data Mining Query language that allows the user to describe ad hoc mining tasks, should be integrated with a data warehouse query language and optimized for efficient and flexible data mining.
- **Presentation and visualization of data mining results** – Once the patterns are discovered it needs to be expressed in high level languages, and visual representations. These representations should be easily understandable.
- **Handling noisy or incomplete data** – The data cleaning methods are required to handle the noise and incomplete objects while mining the data regularities. If the data cleaning methods are not there then the accuracy of the discovered patterns will be poor.
- **Pattern evaluation** – The patterns discovered should be interesting because either they represent common knowledge or lack novelty.

Performance Issues

There can be performance-related issues such as follows –

- **Efficiency and scalability of data mining algorithms** – In order to effectively extract the information from huge amount of data in databases, data mining algorithm must be efficient and scalable.
- **Parallel, distributed, and incremental mining algorithms** – The factors such as huge size of databases, wide distribution of data, and complexity of data mining methods motivate the development of

parallel and distributed data mining algorithms. These algorithms divide the data into partitions which is further processed in a parallel fashion. Then the results from the partitions are merged. The incremental algorithms, update databases without mining the data again from scratch.

Diverse Data Types Issues

- **Handling of relational and complex types of data** – The database may contain complex data objects, multimedia data objects, spatial data, temporal data etc. It is not possible for one system to mine all these kind of data.
- **Mining information from heterogeneous databases and global information systems** – The data is available at different data sources on LAN or WAN. These data source may be structured, semi structured or unstructured. Therefore mining the knowledge from them adds challenges to data mining.

Data Mining Metrics:

Data mining metrics are used to measuring the **effectiveness or usefulness** of a data mining. Different metrics could be used for different techniques and also based on the interest level.

ROI: Return on Investment:

From an overall business or usefulness perspective a measure such as ROI could be used, ROI examines the difference between **what the data mining technique costs** and **what the savings / what the benefits from its use are.** It could be measured as, **increased sales , reduced advertising expenditure** or **both.**

Traditional Metrics:

Our objective is to compare different alternatives to implementing a specific data mining task. The metrics used include the traditional metrics of **space** and **time** based on complexity analysis.

Social Implications of Data Mining:

⇨ Data mining is the process of knowledge discovery where knowledge is gained by analysing the data store in very large repositories, which are analysed from various perspectives and the result is summarized it into useful information.

⇨ Advancements in Statistics, Machine Learning, Artificial Intelligence, Pattern recognition and Computation capabilities have given present day's data mining functionality a new height. Data mining have various applications and these applications have enriched the various fields of human life including business, education, medical, scientific etc.

⇨ For most of us, data mining has become a part of our daily lives. Data mining, affecting everyday things from the products stocked at our local supermarket, to the ads we see while surfing the Internet, to crime prevention.

⇨ Data mining can offer the individual many benefits by improving customer service and satisfaction, and lifestyle, in general. Data mining is present in many aspects of our daily lives, whether we realize it or not. It affects how we shop, work, search for information, and can even influence our leisure time, health, and well-being. In this section, we look at examples of such ubiquitous (or ever-present) data mining. Some demerits of data mining are,

- **Unauthorized use of confidential data**
- **Violation of accessing protected data**
- **Guidelines violations**

Data mining from a database perspective:

Data mining can be studied from various perspectives. They are,

- ⇨ **IR Researcher**
  - An IR researcher probably would concentrate on the use of data mining techniques to **access text data**.
- ⇨ **Statistician**
  - A statistician might look primarily at the **historical techniques**.
- ⇨ **Machine learning specialist**
  - A machine learning specialist might be interested primarily in **data mining algorithms that learn.**
- ⇨ **Algorithm Researcher**
  - An algorithm researcher would be interested in studying and comparing algorithms based on **type** and **complexity.**

Here we are concerned about the following implementation issues of data mining on database perspective,

Scalability: Algorithm that does not scale up to perform well massive real world datasets is of limited application.

Real-World data: Real-World data are noisy and have many missing attribute values. Algorithms should be able to work even in the presence of these problems.

Update: Many data mining algorithms work with static datasets. This is not a realistic assumption.

Ease of Use: Although some algorithms may work well, they may not be well received by users.

7BCEE1A – DATA MINING AND DATA WAREHOUSING

UNIT – 4

CHAPTER 2: RELATED CONCEPTS

- Database / OLTP Systems
- Fuzzy sets and Fuzzy Logic
- Information Retrieval
- Decision Support System
- Dimensional Modeling
- OLAP
- Web Search Engines

CHAPTER 3: DATA MINING TECHNIQUES

- Introduction
- A Statistical Perspective on Data Mining
- Similarity Measures
- Decision Trees
- Neural Networks
- Genetic Algorithms

CHAPTER 2: RELATED CONCEPTS

Database / OLTP Systems:

- **Database** is a collection of data usually associated with some organization or enterprise. Data in a database are usually viewed to have a particular structure/schema with which it is associated.

  For example: Personnel database.

  Table schema – It consists of collection of records and **fields**.

- **DBMS** Data Base Management System is software used to access database.

- **Data Model –** Data stored in a database are often viewed in an abstract manner is called data model.
  - **ER Model** is the common model used ER diagram to represent the data. The basic component of ER model is **entity** and **relationship.**
  - **Relational Model** being composed of relations. In a mathematical perspective a relation is viewed as a **subset of Cartesian product.**

- **On-Line Transaction Processing (OLTP) System** refers to the system that manages transaction oriented applications. These systems are designed to support on-line transaction and process query quickly on the Internet.

For example: POS (point of sale) system of any supermarket is a OLTP System.

Fuzzy Sets and Fuzzy Logic:

- **Set** – A set is normally thought of as a collection of objects. It can be defined by enumerating the set.

  Example: F= {1,2,3,4,5} -> Enumeration representation

  F= {x | x ∈ z^+ and x<=5 } -> Membership representation

- **Fuzzy Set -** A fuzzy set is a set F, in which the set membership function, f is a real valued function with output in the range [0,1].
  - An element x is said to belong to F with probability f(x).
  - An element x to be in ¬F with probability 1-f(x).

  Query against Set:

  The query against set will produce the **Boolean result.**

  Example: select name from R where salary > 100000

  Query against Fuzzy Set:

  Suppose we wished to find the names of employees who are all tall. The results of this query are **fuzzy.**

- **Fuzzy Logic -** is reasoning with uncertainty. That is instead of a two valued logic (true, false) , there are multiple values (true, false, maybe).

  Fuzzy logic has been used in database system to retrieve data with **imprecise / missing values.**

Fuzzy logic uses the following operators to perform the specific operations.

Operator	**Operation**
˥	Negation
^	Intersection
v	Union

- Fuzzy logic uses rules and membership functions to estimate a continuous function. Fuzzy logic is valuable to develop control systems for such things as **elevators, trains** and **heating systems.**
- Most real world classification problems are fuzzy.

  Example: Loan approval problem based on **income** and **loan amount requested.**

  Black and White Classification:

  The loan office may take the decision by,

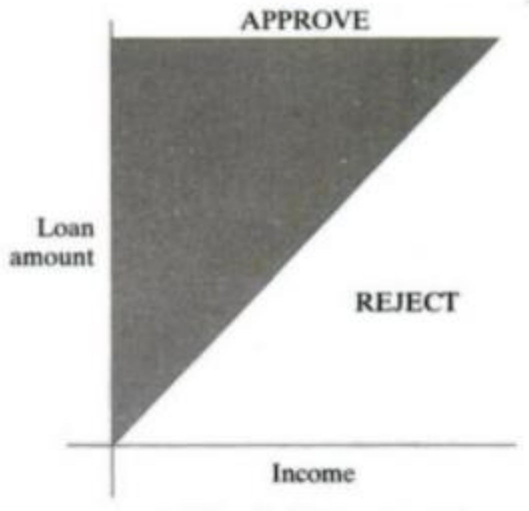
  - Simply approving the loan request on or above the line.
  - Rejecting the loan that fall below the line.

  Fuzzy Classification:

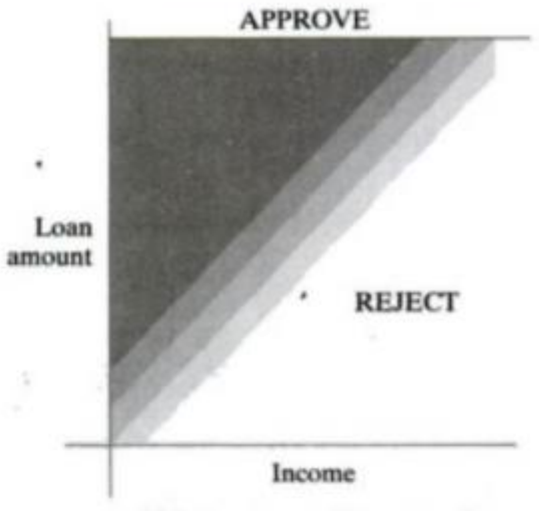
  Here we could classify the individuals into multiple classes:

  - Approve
  - Reject
  - Probably Approve
  - Probable Reject
  - Unknown

$$\left.\begin{array}{l}\text{Probably Approve}\\ \text{Probable Reject}\\ \text{Unknown}\end{array}\right\} \rightarrow \textbf{Fuzzy classes}$$



(a) Simplistic loan approval



(b) Loan approval is not precise

Information Retrieval:

- Information Retrieval involves retrieving desired information from textual data. The historical development of IR was based on effective use of **libraries.**
- In IR systems, documents are represented by document surrogates consisting of data, such as **identifiers, title, authors, dates, abstracts, extracts, reviews** and **keywords.**
- An IR system consists of a set of documents **D= { D1, D2, ..., Dn}.** The input is a query **q,** often stated a **list of keywords.** The similarity between the query and each document is then calculated: **sim(q, Di).**
- The effectiveness of the system is processing the query is measured by looking at **precision** and **recall.**

$$\text{Precision} = \frac{|\text{Relevant and Retrieved}|}{|\text{Retrieved}|}$$

$$\text{Recall} = \frac{|\text{Relevant and Retrieved}|}{|\text{Relevant}|}$$

Precision:

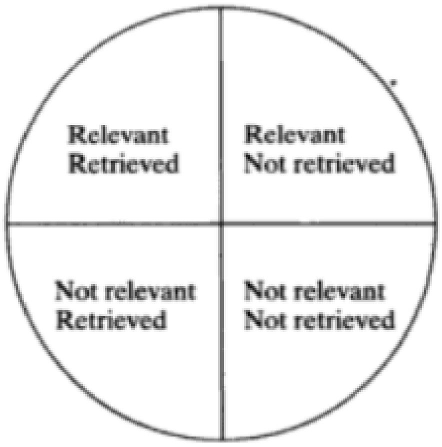
It is used to answer the question "**Are all documents retrieved one that I am interested in?**"

Recall:

It is used to answer the question "**Have all relevant documents been retrieved?"**

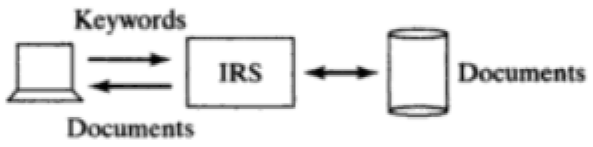
- The four possible query results available with IR queries
  - Relevant Retrieved
  - Not Relevant Not Retrieved → **Desirable outcome**
  - Relevant Not Retrieved
  - Not Relevant Retrieved → **Error situations**

IR query results



- Many similarity measures have been proposed for use in information retrieval.
    - **Sim(q, Di)** → query to document similarity
    - **Sim(Di, Dj)** → document to document similarity
    - **Sim(qi, qj)** → query to query similarity

Information retrieval query



- The **Inverse Document Frequency (IDF)** is often used by similarity measures. Here the similarity measure is inversely proportional to the total number of documents that contain it.

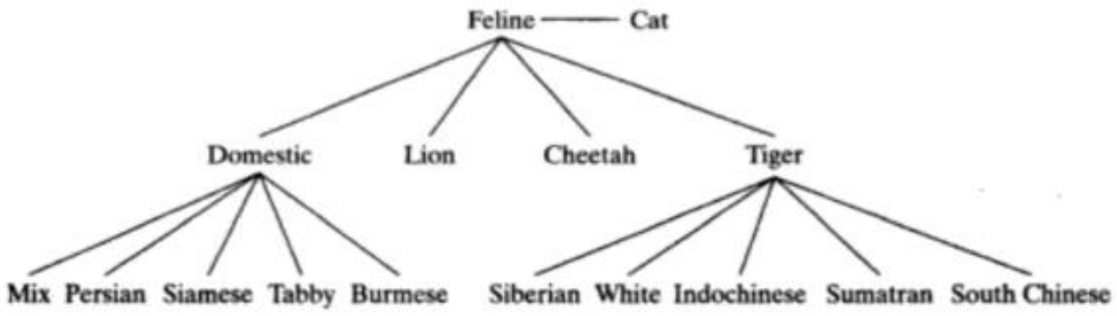
$$IDF_k = lg \frac{n}{|\text{documents containing } k|} + 1$$

n – Number of documents k – given keyword

- **Concept Hierarchy** is used in information retrieval systems to show the relationship between various keywords.

  Example: wish to find all documents about cats

Concept Hierarchy



Decision Support System:

- Decision Support Systems (**DSS**) are comprehensive computer systems and related tools that assist managers in **making decisions** and **solving problems.**
- The goal is to improve the decision making process by providing specific information needed by management.
- The DSS differ from database management system in that more adhoc queries and customized information may be provided.
- Recently, the **Executive Information System (EIS),** and **Executive Support System (ESS)** has evolved as well. These systems all aim at developing the business structure and computer techniques to better provide information needed by management to make **effective business decisions.**
- DSS may be use **data mining tools** to make **intelligent business decisions** that impact the entire company / enterprise.

Dimensional Modeling:

- Dimensional modeling is a different way to **view** and **interrogate** data in a database. A **dimension** is a collection of logically related **attributes** and is viewed as an **axis** for modeling the data.

  Example: The time dimension could be divided into many granularities:

  millennium, century, decade, year, month, date, hour, minute, second,..
- Here we use the three type of dimension modelling view,
  - **Relational view**

- **Cube view**
- **Aggregation hierarchy/Lattice/Directed path view**

➢ **Relational view:**

For example we take the following relation with three dimensions,

Product ID	Location ID	Date
123	Dallas	022900
123	Houston	020100
150	Dallas	031500
150	Dallas	031500
150	Fort worth	021000
150	Chicago	012000
200	Seattle	030100
300	Rochester	021500
500	Bradenton	022000
500	Chicago	012000

Same data

Determining the key for this relation could be difficult because it is possible for the **same product to be sold multiple times on the same day and at same location.** In this case, product 150 was sold at two times in Dallas on the same day.

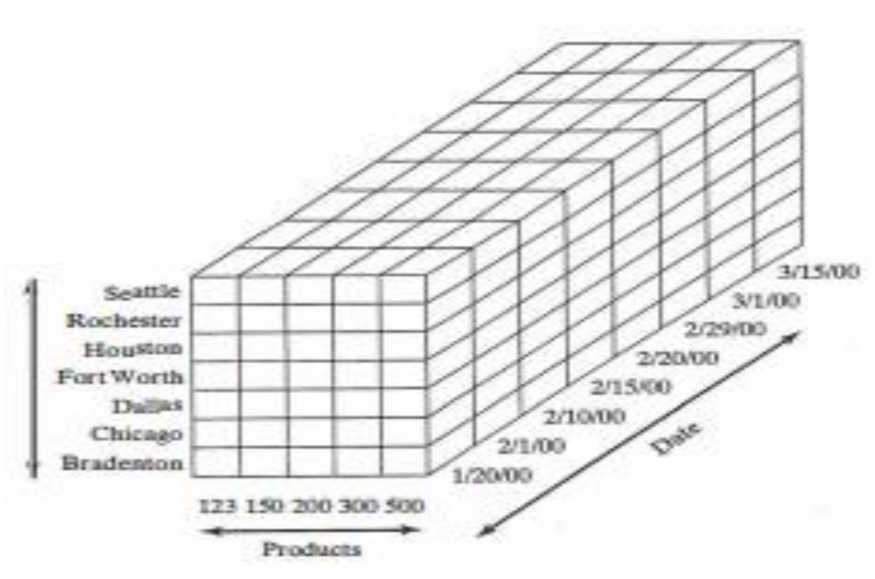
➢ **Cube view:**

The same multi dimensional may also be viewed as a **cube.** Each dimension is an **axis** for the cube. This cube has one fact for each unique combination of dimension values.

- **productID ->** 123, 150, 200, 300, 500 (5 unique entries)
- **LocationID ->** Dallas, Houston, Fort worth, Chicago, Seattle, Rochester, Bradenton (7 unique entries)
- **Date ->**022900, 020100,031500, 021000, 012000, 030100,021500,022000 (8 unique entries)

So, this cube could have **8*7*5=200** facts stored.

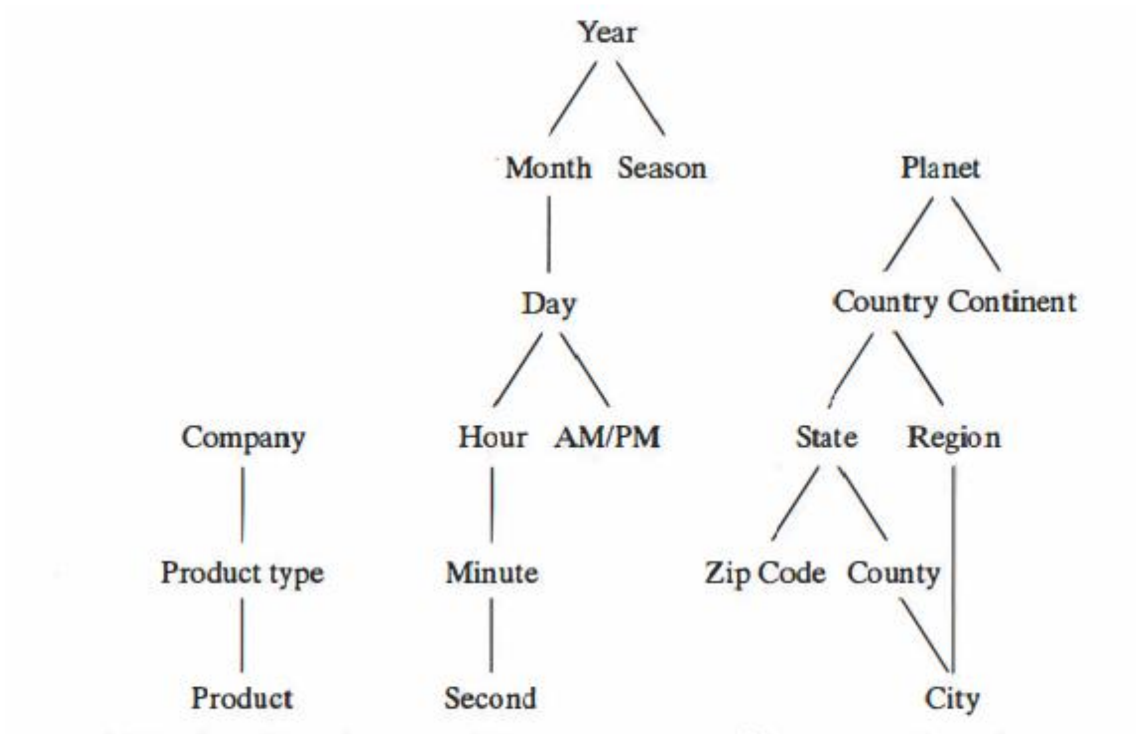
Cube View



- **Aggregation hierarchy / Lattice / Directed path view:**
  - The levels of dimension may support **Partial order** or **Total order** and can be viewed via a directed path, a hierarchy or lattice.
  - To refer to the **order relationship** among different levels in the dimension we use **<.**

Example:

Aggregation Hierarchy



(a) Product dimension (b) Time dimension (c) Location dimension

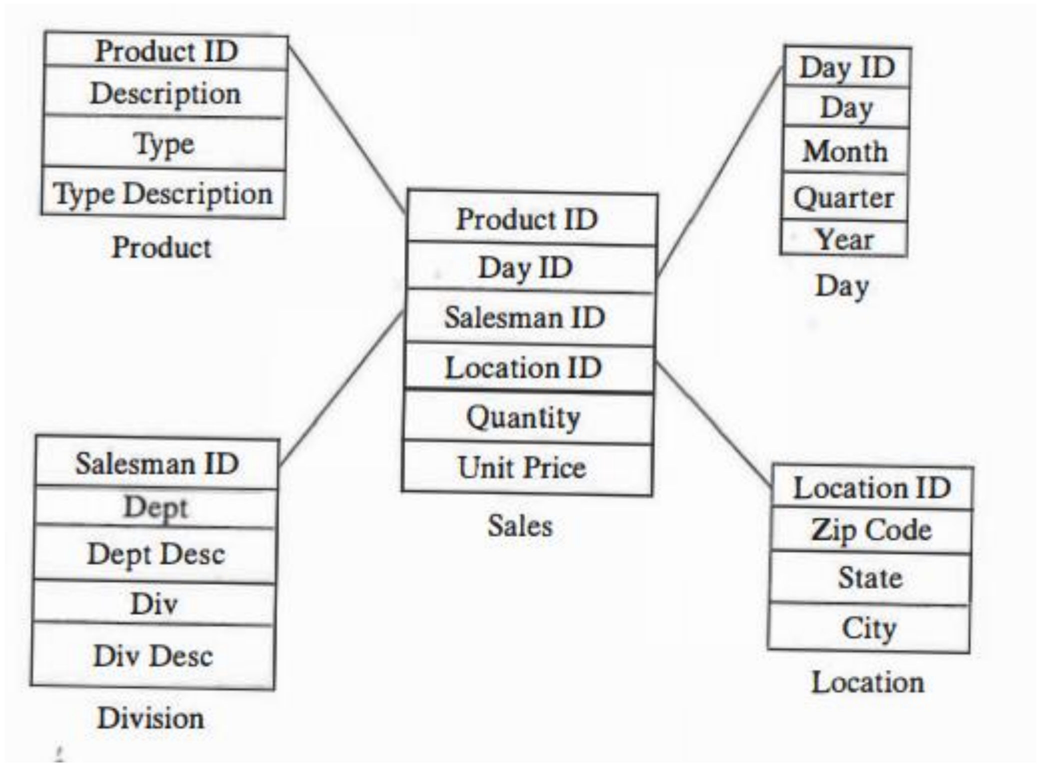
- Here, **Day < Month** but **Day $\not<$ Season.** The aggregation can be applied to levels that can be found in the **same path** as defined by the **<** relationship.

Additive dimension: If we add the sales data for all **24 hours** in a day, we get the sales data for that **day.**

Non-Additive dimension: If we were to sum up the sales data for **all zip codes**, we would not get sales data for the **country.**

- **Multidimensional Schema:**
    - The different type of multidimensional schemas are, **star schema, snowflake schema,** and **fact constellation schema.**
    - A **star schema** shows data as a collection of two types: **facts** and **dimensions.** At the centre of the star, the data being examined, the facts are shown in **fact tables / major tables.** On the outside of the facts, each dimension is shown separately in **dimension tables / minor tables.**

Star Schema



    - Access to the fact table from a dimension table can be accomplished via a join between a dimension table and the fact table on particular dimension values.

- **Indexing:**
    - With multidimensional data, indices help to reduce the overhead of scanning the extremely large tables.
    - There are two type of indices used in multidimensional data.
        - **Bitmap indices:**

          Each tuple in the table is represented by a predefined bit so that a table with **n tuples** would be represented by a vector of **n bits.**

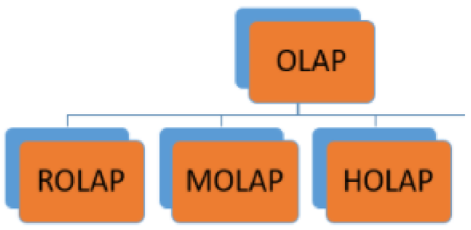
        - **Join indices:**

          It support joins by pre computing tuples from tables that **join together** and **pointing** to the tuples in those tables.

OLAP:

- **Online Analytical Processing (OLAP)** is a category of software that allows users to analyze information from multiple database systems at the same time. It is a technology that enables analysts to extract and view business data from different points of view.

- Analysts frequently need to group, aggregate and join data. These operations in relational databases are resource intensive. With OLAP data can be pre-calculated and pre-aggregated, making analysis faster.
- OLAP databases are divided into one or more cubes. The cubes are designed in such a way that creating and viewing reports become easy.
- **Types of OLAP systems:**



Type of OLAP	Explanation
Relational OLAP(ROLAP):	ROLAP is an extended RDBMS along with multidimensional data mapping to perform the standard relational operation.
Multidimensional OLAP (MOLAP)	MOLAP Implements operation in multidimensional data.
Hybrid Online Analytical Processing (HOLAP)	In HOLAP approach the aggregated totals are stored in a multidimensional database while the detailed data is stored in the relational database. This offers both data efficiency of the ROLAP model and the performance of the MOLAP model.

- **Basic analytical operations of OLAP**

Four types of analytical operations in OLAP are:

1. **Roll-up**
2. **Drill-down**
3. **Slice and dice**
4. **Pivot (rotate)**

1) Roll-up:

Roll-up is also known as "consolidation" or "aggregation." The Roll-up operation can be performed in 2 ways

1. Reducing dimensions
2. Climbing up concept hierarchy. Concept hierarchy is a system of grouping things based on their order or level.

2) Drill-down

In drill-down data is fragmented into smaller parts. It is the opposite of the rollup process. It can be done via

- Moving down the concept hierarchy
- Increasing a dimension

3) Slice:

Here, one dimension is selected, and a new sub-cube is created.

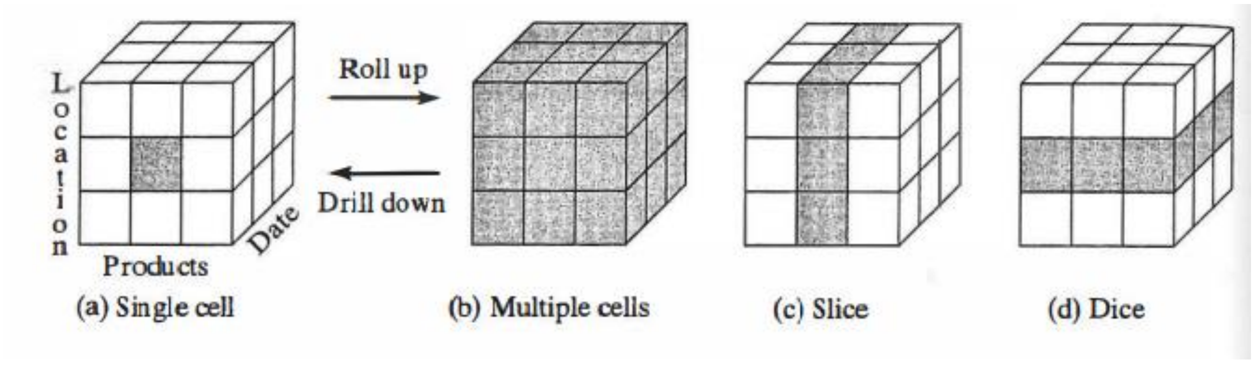
Dice:

This operation is similar to a slice. The difference in dice is you select 2 or more dimensions that result in the creation of a sub-cube.

4) Pivot

In Pivot, you rotate the data axes to provide a substitute presentation of data.

OLAP Operations



Web Search Engines:

- **Search Engine** refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web.
- User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user.
- **Search Engine Components:**

  Generally there are three basic components of a search engine as listed below:

  1. Web Crawler
  2. Database
  3. Search Interfaces

  - **Web crawler**

    It is also known as **spider** or **bots.** It is a software component that traverses the web to gather information.

- **Database**

  All the information on the web is stored in database. It consists of huge web resources.

- **Search Interfaces**

  This component is an interface between user and the database. It helps the user to search through the database.

➢ Conventional search engines suffer from following issues:

- **Abundance** – Most of the data of the web are of **no interest** to most people.
- **Limited coverage** – Search engine often provide results from a subset of the web pages.
- **Limited query** – Most search engines provide access based only on **simple keyword based searching.**
- **Limited customization** – query results are often determined only by the query itself.

CHAPTER 3: DATA MINING TECHNIQUES

Introduction:

- There are many different methods used to perform data mining tasks. These techniques not only require **specific types of data structures,** but also imply **certain types of algorithmic approaches.**
- In generally the data mining techniques are classified as,
    - **Parametric Model technique**
    - **Non-Parametric Model technique**

Parametric Model:

It describes the relationship between **input** and **output** through the use of **algebraic equations** where some parameters are not specified. These unspecified parameters are determined by providing input examples.

Non-Parametric Model:

A Non-Parametric model is one that is **data driven.** Here **no explicit equations** are used to determine the model. This means that the modelling process adapts to the **data at hand.** Non parametric model techniques include **neural networks, decision trees,** and **genetic algorithms.**

A Statistical Perspective on Data Mining:

Some of the statistical concepts that are basis for data mining techniques are,

- **Point Estimation**
- **Models Base on Summarization**
- **Bayes Theorem**
- **Hypothesis Testing**
- **Regression and Correlation**

Point Estimation:

⇨ Point estimation refers to the process of estimating a population parameter, **O**, by an estimate of the parameter, **Ô**. This can be done to estimate **mean, variance, standard deviation, or any other statistical parameter.**

⇨ Often the estimate of the parameter for a general population may be made by actually calculating the parameter value for a population sample. An estimator technique may also be used to estimate (predict) the value of missing data. The **bias** of an estimator is the difference between the expected value of the estimator and the actual value:

$$\textbf{Bias} = E(\hat{O})-O$$

An unbiased estimator is one whose bias is 0.

- One measure of the effectiveness of an estimate is the **mean squared error(MSE)**, which is defined as the expected value of the squared difference between the estimate and the actual value:

  MSE(G) = E(G - 8)2

- The **root mean square (RMS)** may also be used to estimate error or as another statistic to describe a distribution.

$$\mathrm{RMS} = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n}}$$

- An alternative use is to estimate the magnitude of the error. The **root mean square error (RMSE)** is found by taking the square root of the MSE.

- A popular estimating technique is the **jackknife estimate**. With this approach, the estimate of a parameter, e, is obtained by omitting one value from the set of observed

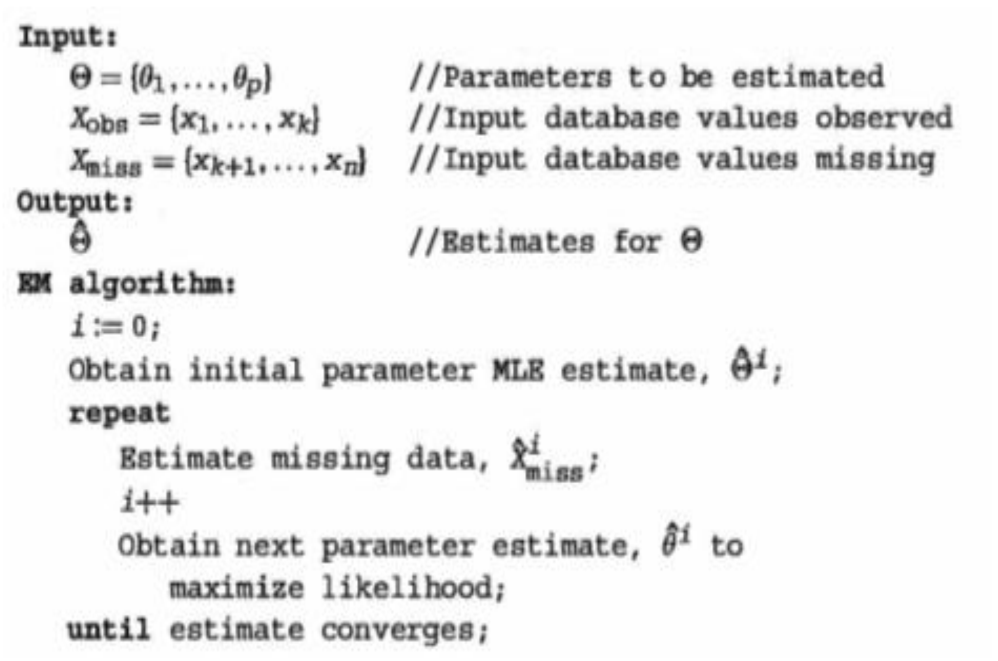
$$\hat{\theta}_{(.)} = \frac{\sum_{j=1}^{n} \hat{\theta}_{(j)}}{n}$$

- Another technique for point estimation is called the **maximum likelihood estimate (MLE)**. Likelihood can be defined as a value proportional to the actual probability that with a specific distribution the given sample exists. The likelihood function, L, is thus defined as,

$$L(\Theta \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i \mid \Theta)$$

- The **expectation-maximization (EM) algorithm** is an approach that solves the estimation problem with incomplete data. The EM algorithm finds an MLE for a parameter (such as a mean) using a two-step process: **estimation** and **maximization**.

EM -Algorithm



```
Input:
   $\Theta = \{\theta_1, \ldots, \theta_p\}$          //Parameters to be estimated
   $X_{obs} = \{x_1, \ldots, x_k\}$        //Input database values observed
   $X_{miss} = \{x_{k+1}, \ldots, x_n\}$   //Input database values missing
Output:
   $\hat{\Theta}$                          //Estimates for $\Theta$
EM algorithm:
   $i := 0$;
   Obtain initial parameter MLE estimate, $\hat{\Theta}^i$;
   repeat
      Estimate missing data, $X^i_{miss}$;
      $i$++
      Obtain next parameter estimate, $\hat{\theta}^i$ to
         maximize likelihood;
   until estimate converges;
```

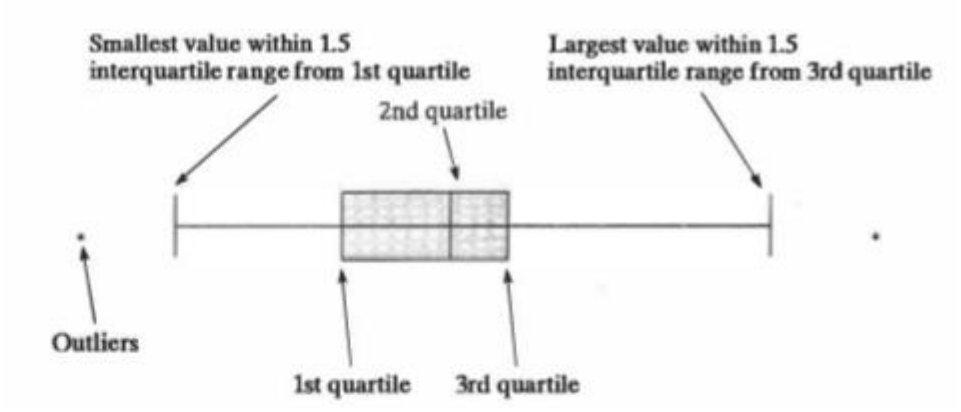
Models Based on Summarization:

- There are many basic concepts that provide an abstraction and summarization of the data as a whole. The basic well-known statistical concepts such as **mean, variance, standard deviation, median, and mode** are simple models of the underlying population. Fitting a population to a specific frequency distribution provides an even better model of the data.

- There are also many well-known techniques to display the structure of the data graphically. For example, a h**istogram** shows the distribution of the data.

⇨ A **box plot** is a more sophisticated technique that illustrates several different features of the population at once. The total range of the data values is divided into **four equal parts called quartiles**. The box in the center of the figure shows the range between the first, second, and third quartiles. The line in the box shows the median.

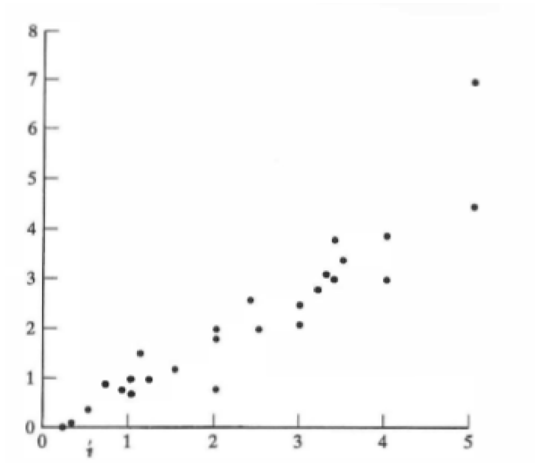
Box plot example



Smallest value within 1.5 interquartile range from 1st quartile
Largest value within 1.5 interquartile range from 3rd quartile
2nd quartile
Outliers
1st quartile
3rd quartile

⇨ Another visual technique to display data is called a **scatter diagram**. This is a graph on a two-dimensional axis of points representing the relationships between x and y values.

Scatter diagram example



Bayes Theorem:

Bayes Rule or Bayes Theorem

$$P(h_1 \mid x_i) = \frac{P(x_i \mid h_1)P(h_1)}{P(x_i \mid h_1)P(h_1) + P(x_i \mid h_2)P(h_2)}$$

- ⇨ Here P(h1 I Xi) is called the **posterior probability**, while P(h1) is the **prior probability** associated with hypothesis h 1. P (xi) is the probability of the occurrence of data value Xi and P(xi I h1) is the conditional probability that, given a hypothesis, the tuple satisfies it.

  Where there are m different hypotheses we have:

$$P(x_i) = \sum_{j=1}^{m} P(x_i \mid h_j)P(h_j)$$

  Thus, we have

$$P(h_1 \mid x_i) = \frac{P(x_i \mid h_1)P(h_1)}{P(x_i)}$$

Hypothesis Testing:

- ⇨ Hypothesis testing attempts to find a model that explains the observed data by first creating a hypothesis and then testing the hypothesis against the data.

- This is in contrast to most data mining approaches, which create the model from the actual data without guessing what it is first. The actual data itself drive the model creation.
- hypothesis usually is verified by examining a data sample. If the hypothesis holds for the sample, it is assumed to hold for the population in general. Given a population, the initial (assumed) hypothesis to be tested, **Ho**, is called the **null hypothesis.** Rejection of the null hypothesis causes another hypothesis, **Ht**, called the **alternative hypothesis**, to be made.
- One technique to perform hypothesis testing is based on the use of **the chi-squared statistic**. A hypothesis is first made, and then the observed values are compared based on this hypothesis. Assuming that 0 represents the observed data and E is the expected values based on the hypothesis, the chi-squared statistic is defined as:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

Regression and Correlation:

- Both bivariate regression and correlation can be used to evaluate the strength of a relationship between two variables.
- **Regression** is generally used to predict future values based on past values by fitting a set of points to a curve.
- Linear regression assumes that a linear relationship exists between the input data and the output data. The common formula for a linear relationship is used in this model:

$$y = c_0 + c_1 x_1 + \cdots + c_n x_n$$

- Here there are n input variables, which are called **predictors** or **regressor**s; one output variable (the variable being predicted), which is called the response; and n + 1 constants, which are chosen during the modeling process to match the input examples (or sample). This is sometimes called **multiple linear regression** because there is more than one predictor.
- **Correlation**, however, is used to examine the degree to which the values for two variables behave similarly.
- Two different data variables, X and Y, may behave very similarly. Correlation is the problem of determining how much alike the two variables actually are. One standard formula to measure linear correlation is the correlation coefficient r. Given two variables, X and Y, the correlation coefficient is a real value r E [-1, 1]. A positive number indicates a **positive correlation**, whereas a negative number indicates a **negative correlation.**
- The value for r is defined as,

$$r = \frac{\sum (x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum (x_i - \bar{X})^2 \sum (y_i - \bar{Y})^2}}$$

where X and Y are the means for X and Y, respectively. Suppose that X = (2, 4, 6, 8, 10). If Y = X, then r = 1. When Y = (1, 3, 5, 7, 9), r = 1. If Y = (9, 7, 5, 3, 1), r = -1.

SIMILARITY MEASURES:

- The similarity between two tuples t; and t1, sim(t;, t1), in a database Dis a mapping from D x D to the range [0, 1]. Thus, sim(t;, t1) E [0, 1].
- The objective is to define the similarity mapping such that documents that are more alike have a higher similarity value. Thus, the following are desirable characteristics of a good similarity measure:
  - Vt; ED , sim(t;, t;) = 1
  - Vt;, t1 ED , sirn(t;, fJ) = 0 if t; and t1 are not alike at all
  - Vt;, fJ , tk ED , sim(t;, fJ) < sim(t;, tk) if t; is more like tk than it is like t1
- Here are some of the more common similarity measures used in traditional IR systems and more recently in Internet search engines:

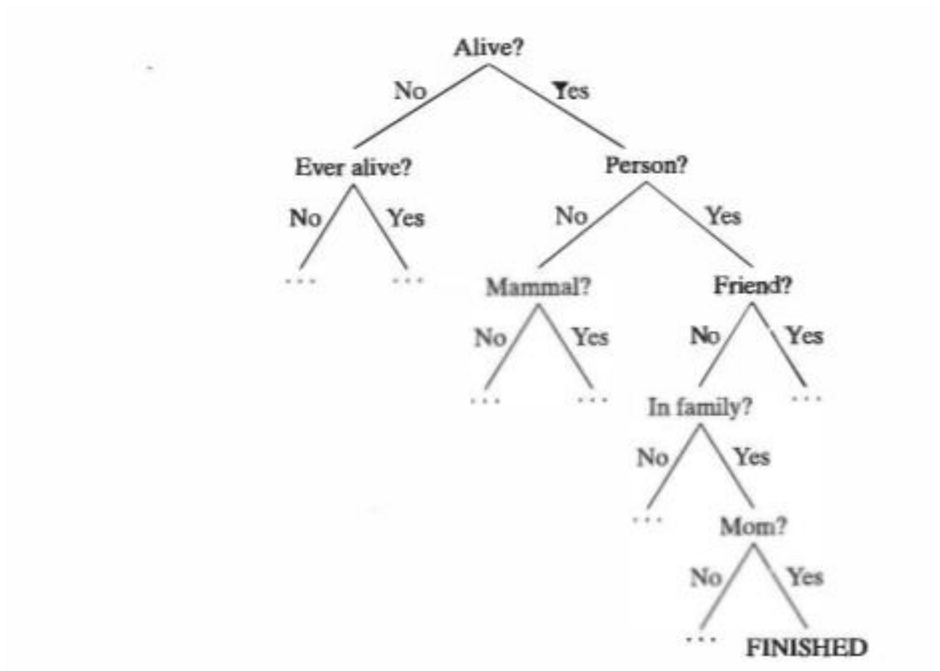
- **Dice:** $\mathrm{sim}(t_i, t_j) = \dfrac{2\sum_{h=1}^{k} t_{ih}t_{jh}}{\sum_{h=1}^{k} t_{ih}^2 + \sum_{h=1}^{k} t_{jh}^2}$
- **Jaccard:** $\mathrm{sim}(t_i, t_j) = \dfrac{\sum_{h=1}^{k} t_{ih}t_{jh}}{\sum_{h=1}^{k} t_{ih}^2 + \sum_{h=1}^{k} t_{jh}^2 - \sum_{h=1}^{k} t_{ih}t_{jh}}$
- **Cosine:** $\mathrm{sim}(t_i, t_j) = \dfrac{\sum_{h=1}^{k} t_{ih}t_{jh}}{\sqrt{\sum_{h=1}^{k} t_{ih}^2 \sum_{h=1}^{k} t_{jh}^2}}$
- **Overlap:** $\mathrm{sim}(t_i, t_j) = \dfrac{\sum_{h=1}^{k} t_{ih}t_{jh}}{\min\left(\sum_{h=1}^{k} t_{ih}^2, \sum_{h=1}^{k} t_{jh}^2\right)}$

- **Distance** or **dissimilarity measures** are often used instead of similarity measures. As implied, these measure how "unlike" items are. Traditional distance measures may be used in a two-dimensional space. These include,

- **Euclidean:** $\text{dis}(t_i, t_j) = \sqrt{\sum_{h=1}^{k} (t_{ih} - t_{jh})^2}$
- **Manhattan:** $\text{dis}(t_i, t_j) = \sum_{h=1}^{k} | (t_{ih} - t_{jh}) |$

Decision Trees:

- A decision tree is a predictive modeling technique used in classification, clustering, and prediction tasks. Decision trees use a **"divide and conquer"** technique to split the problem search space into subsets.
- **Example:** "Twenty Questions" game .



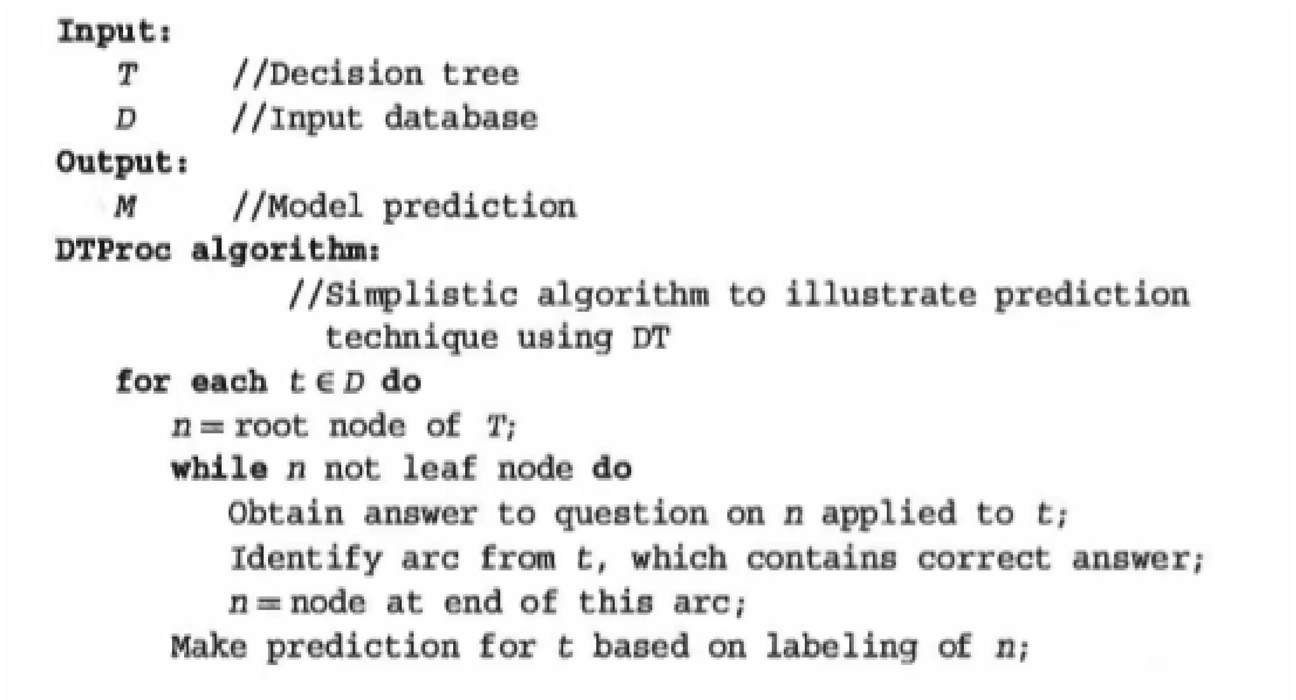
This tree has as the root the first question asked. Each subsequent level in the tree consists of questions at that stage in the game.

- **Definition:**
  - **A decision tree (DT)** is a tree where the root and each internal node is labeled with a question. The arcs emanating from each

node represent each possible answer to the associated question. Each leaf node represents a prediction of a solution to the problem under consideration.

- An algorithm to create the tree.
- An algorithm that applies the tree to data and solves the problem under consideration.

DTProc Algorithm



```
Input:
   T     //Decision tree
   D     //Input database
Output:
   M     //Model prediction
DTProc algorithm:
            //Simplistic algorithm to illustrate prediction
              technique using DT
   for each t ∈ D do
      n = root node of T;
      while n not leaf node do
         Obtain answer to question on n applied to t;
         Identify arc from t, which contains correct answer;
         n = node at end of this arc;
      Make prediction for t based on labeling of n;
```

- The complexity of the algorithm is straightforward to analyze. For each tuple in the database, we search the tree from the root down to a particular leaf. At each level, the maximum number of comparisons to make depends on the branching factor at that level. So the complexity depends on the product of the number of levels and the maximum branching factor.

Neural Networks:

- The **Artificial Neural Network (ANN)** bases its assimilation of data on the way that the human brain processes information. The brain has billions of cells called neurons that process information in the form of electric signals. External information, or stimuli, is received, after which the brain processes it, and then produces an output.
- Similarly, ANN receives input through a large number of processors that operate in parallel and are arranged in tiers. The first tier receives the raw input data, which it then processes through nodes that are interconnected and have their own packages of knowledge and rules. The processor then passes it on to the next tier as output. Each successive tier of processors and nodes receives the output from the tier preceding it and further processes it; rather than having to process the raw data anew every time.
- Neural networks modify themselves as they learn from their robust initial training and then from ongoing self-learning that they experience by processing additional information. A simple learning model applied by neural networks is the process of weighting input streams in favour of those most likely to be accurate. That means a preference is put on the input streams that have a higher weight; and the higher the weight, the more influence that unit has on another. The process of reducing predictable errors through weight, is done through gradient descent algorithms. Finally, output units are the end part of the process; this is where the network responds to the data that was put in initially, and can now be processed.

Artificial Neural Network (ANN)

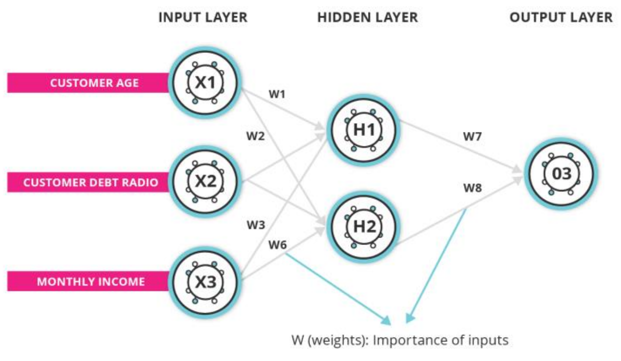


Image source: Mahanta, J. (Jul, 2017). 'Introduction to neural networks, advantages and applications'. Retrieved from Towards Data Science.

➢ Definition:

A neural network (NN) is a directed graph, F = (V, A) with vertices V = {1, 2, ... , n} and arcs A= {(i, j} 11::::: i, j s n}, with the following restrictions:

- V is partitioned into a set of input nodes, VI, hidden nodes, V H, and output nodes, Vo.
- The vertices are also partitioned into layers {1, ... , k} with all input nodes in layer 1 and output nodes in layer k. All hidden nodes are in layers 2 to k - 1 which is called the hidden layers.
- Any arc (i, j} must have node i in layer h-1 and node j in layer h.
- Arc (i, j} is labeled with a numeric value WiJ· 5. Node i is labeled with a function fi .

- **Activation Functions**:

⇨ The output of each node i in the NN is based on the definition of a function f;, activation function, associated with it. An activation function is sometimes called a **processing element function** or a **squashing function**.

⇨ An activation function may also be called a **firing rule**, relating it back to the workings of the human brain. When the input to a neuron is large enough, it fires, sending an electrical signal out on its axon (output link).

⇨ Activation functions may be **unipolar**, with values in [0, 1], or **bipolar**, with values in [-1, 1].

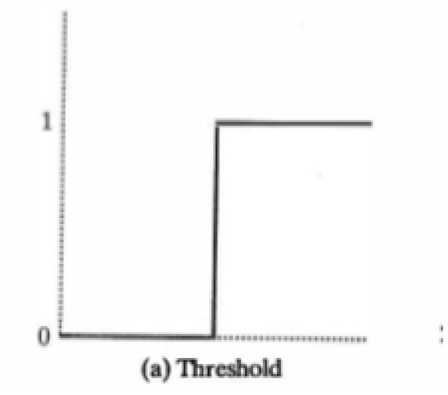
- **Linear:** A linear activation function produces a linear output value based on the input. The following is a typical activation function,

  $$f_i(S) = cS$$

  Here c is a constant positive value. With the linear function, the output value has no limits in terms of maximum or minimum values.

- **Threshold or step:** The output value is either a 1 or 0, depending on the sum of the products of the input values and their associated weights.

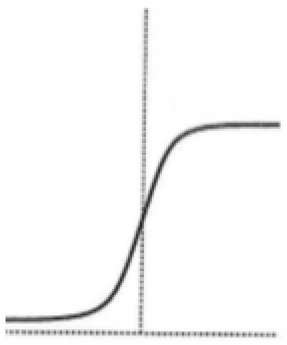
$$f_i(S) = \left\{ \begin{array}{ll} 1 & \text{if } S > T \\ 0 & \text{otherwise} \end{array} \right\}$$



(a) Threshold

- **Sigmoid:** This is an "S"- shaped curve with output values between -1 and 1.

$$f_i(S) = \frac{1}{(1 + e^{-cS})}$$


(b) Sigmoid

Here c is a constant positive value that changes the slope of the function.

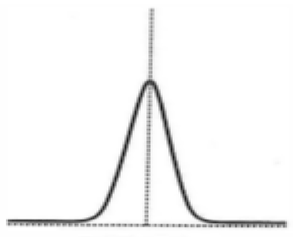
- **Hyperbolic tangent:** A variation of the sigmoid function is the hyperbolic tangent function shown here

$$f_i(S) = \frac{(1 - e^{-S})}{(1 + e^{-cS})}$$

This function has an output centered at zero, which may help with learning.

- **Gaussian:** The Gaussian function, is a bell-shaped curve with output values in the range [0, 1]. A typical Gaussian function is

$$f_i(S) = e^{\frac{-S^2}{v}}$$


(c) Gaussian

Here s is the mean and v is the predefined positive variance of the function.

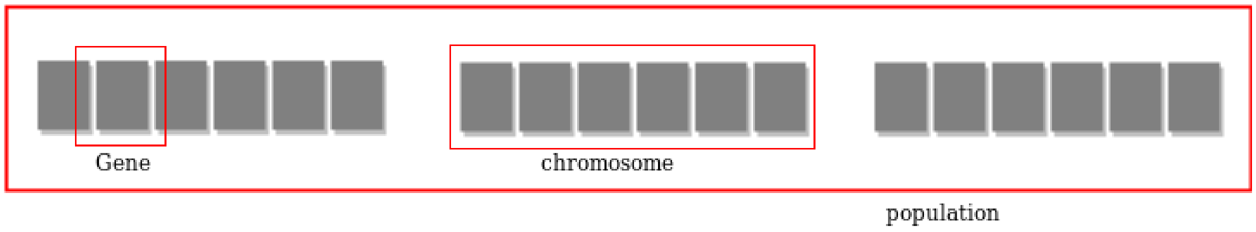
Genetic Algorithms:

- Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. **They are commonly used to generate high-quality solutions for optimization problems and search problems.**
- Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate "survival of the fittest" among individual of consecutive generation for solving a problem. **Each generation consist of a population of individuals** and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.
- Genetic algorithms are based on an analogy with genetic structure and behavior of chromosome of the population. Following is the foundation of GAs based on this analogy –

1. Individual in population compete for resources and mate

2. Those individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from “fittest” parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. Thus each successive generation is more suited for their environment.

- The population of individuals are maintained within search space. Each individual represent a solution in search space for given problem. Each individual is coded as a finite length vector (analogous to chromosome) of components. These variable components are analogous to Genes. Thus a chromosome (individual) is composed of several genes (variable components).



- A Fitness Score is given to each individual which **shows the ability of an individual to “compete”**. The individual having optimal fitness score (or near optimal) are sought.
- The GAs maintains the population of n individuals (chromosome/solutions) along with their fitness scores.The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce **better offspring** by combining chromosomes of parents. The population size is static so the room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals

eventually creating new generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solutions will arrive while least fit die.

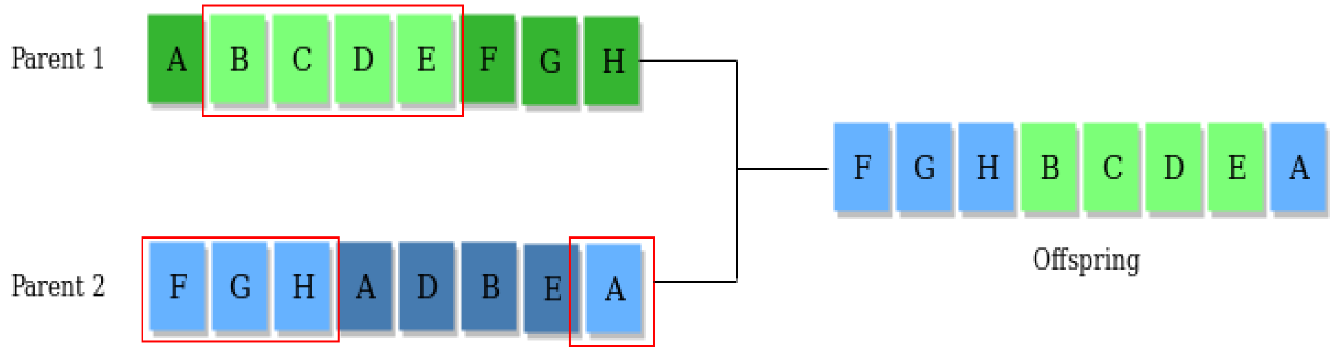
- Each new generation has on average more “better genes” than the individual (solution) of previous generations. Thus each new generations have better **“partial solutions”** than previous generations. Once the offsprings produced having no significant difference than offspring produced by previous populations, the population is converged. The algorithm is said to be converged to a set of solutions for the problem.

- **Operators of Genetic Algorithms**

Once the initial generation is created, the algorithm evolves the generation using following operators –

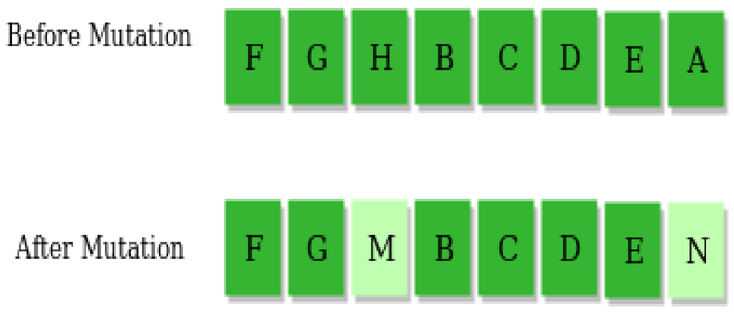
1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass there genes to the successive generations.

2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring). For example –



3) Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence.

For example –



➢ The whole algorithm can be summarized as –

```
1) Randomly initialize populations p
2) Determine fitness of population
3) Until convergence repeat:
    a) Select parents from population
    b) Crossover and generate new population
    c) Perform mutation on new population
    d) Calculate fitness for new population
```

<u>**7BCEE1A – DATA MINING AND DATA WAREHOUSING**</u>

<u>**UNIT – 5**</u>

CHAPTER 6: ASSOCIATION RULES

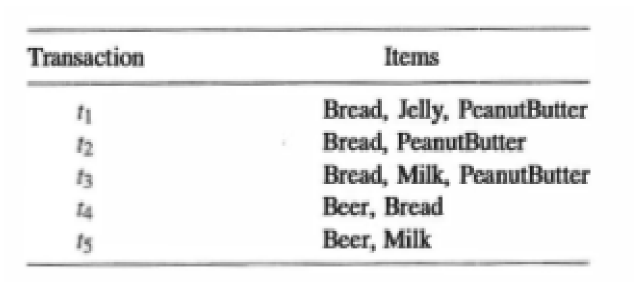
- **Introduction**
- **Large Itemsets**
- **Basic Algorithms**
- **Parallel and Distributed Algorithms**
- **Comparing Approaches**
- **Incremental Rules**
- **Advanced Association Rules**
- **Measuring the Quality of Rules**

CHAPTER 6: ASSOCIATION RULES

Introduction:

- Association rules are used to show the relationships between data items.
- The purchasing of one product when another product is purchased represents an association rule. Association rules are frequently used by retail stores to assist **in marketing, advertising, floor placement,** and **inventory control.**

Sample Data to Illustrate Association Rules



Transaction	Items
t_1	Bread, Jelly, PeanutButter
t_2	Bread, PeanutButter
t_3	Bread, Milk, PeanutButter
t_4	Beer, Bread
t_5	Beer, Milk

- Here, there are five transactions **{t1, t2,t3, t4, t5}** and five items **{Beer, Bread, Jelly, Milk, Peanut Butter}.**
- **Association Rule:**

Given a set of items **I={I1,I2,...,Im}** and a database of transaction **D={t1, t2,...,tn}** where **ti=(Ii1, Ii2,...,Iik)** and **Iij ∈ I ,** an association rule is an implication of the form **X => Y** where **X,Y ⊂ I** are sets of items called **itemsets** and **X∩Y =∅.**

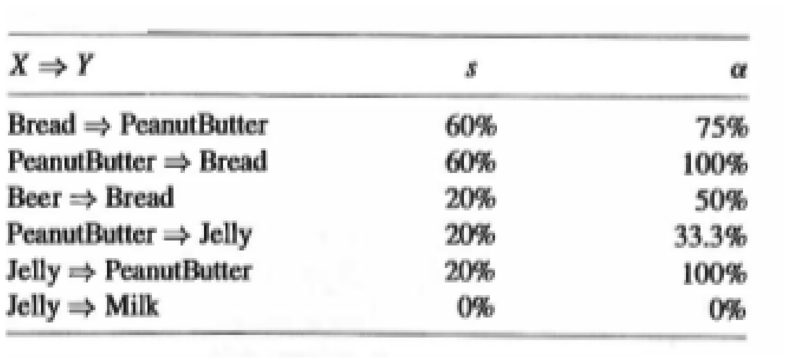
- **Support:**

The support (s) for an association rule X => Y is the percentage of transactions in the database that contain X U Y.

- **Confidence** or **Strength:**

  The confidence or strength (a) for an association rule X => Y is the ratio of the number of transactions that contain X U Y to the number of transactions that contain X.

Support and Confidence for Some Association Rules



$X \Rightarrow Y$	s	α
Bread ⇒ PeanutButter	60%	75%
PeanutButter ⇒ Bread	60%	100%
Beer ⇒ Bread	20%	50%
PeanutButter ⇒ Jelly	20%	33.3%
Jelly ⇒ PeanutButter	20%	100%
Jelly ⇒ Milk	0%	0%

- **Association Rule Problem:**

  Given a set of items **I={I1,I2,...,Im}** and a database of transaction **D={t1, t2,...,tn}** where **ti=(Ii1, Ii2,...,Iik)** and **Iij ∈ I, ,** the association rule problem is to **identify all association rules X => Y** with a minimum support and confidence. These values (s, a) are given as input to the problem.

Large Itemsets:

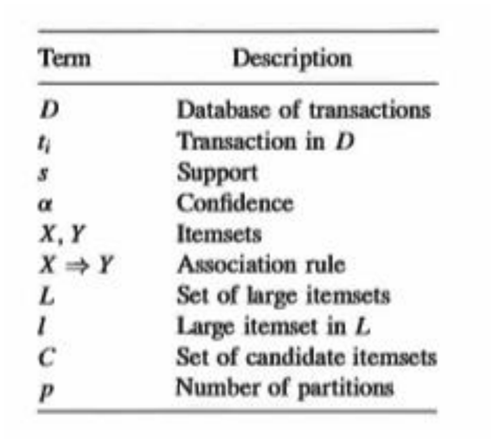
- The most common approach to finding association rules is to break up the problem into two parts:

  1. Find large itemsets.

  2. Generate rules from frequent itemsets.

  An itemset is any subset of the set of all items, I.

Association Rule Notation



Term	Description
D	Database of transactions
t_i	Transaction in D
s	Support
α	Confidence
X, Y	Itemsets
$X \Rightarrow Y$	Association rule
L	Set of large itemsets
l	Large itemset in L
C	Set of candidate itemsets
p	Number of partitions

- **A large (frequent) itemset** is an itemset whose number of occurrences is above a threshold, s. We use the notation L to indicate the complete set of large itemsets and l to indicate a specific large itemset.
- Once the large itemsets have been found, we know that any interesting association rule, X => Y, must have XU Y in this set of frequent itemsets. The **subset of any large itemset is also large.**
- Finding large itemsets generally is quite easy but very costly. The naive approach would be to count all itemsets that appear in any transaction. Given a set of items of size **m**, there are **2^m** subsets. Since we are not interested in the empty set, the potential number of large itemseti is then **2^m - 1**.
- Most association rule algorithms are based on smart ways to reduce the number of itemsets to be counted. These potentially large itemsets are called **candidates**, and the set of all counted (potentially large) itemsets is the **candidate itemset (c).** Duuring the counting process the following data structures to be used common
    - **trie**
    - **hash tree**

ARGen Algorithm

```
Input:
   D     //Database of transactions
   I     //Items
   L     //Large itemsets
   s     //Support
   α     //Confidence
Output:
   R     //Association Rules satisfying s and α
ARGen algorithm:
   $R = \emptyset$;
   for each $l \in L$ do
      for each $x \subset l$ such that $x \neq \emptyset$ do
         if $\frac{\text{support}(l)}{\text{support}(x)} \geq \alpha$ then
            $R = R \cup \{x \Rightarrow (l - x)\}$;
```

Example:

Suppose that the input support and confidence are **s = 30%** and **a = 50%,** respectively. Using this value of s, we obtain the following set of large itemsets:

L = {{Beer}, {Bread}, {Milk}, {PeanutButter}{Bread, PeanutButter}}

We now look at what association rules are generated from the last large itemset. Here **l = {Bread, PeanutButter}**. There are two nonempty subsets of l: {Bread} and {PeanutButter}. With the first one we see:

$$\frac{\text{support}(\{\text{Bread, PeanutButter}\})}{\text{support}(\{\text{Bread}\})} = \frac{60}{80} = 0.75$$

This means that the confidence of the association rule **Bread=>PeanutButter** is **75%.**

Likewise with the second large itemset,

$$\frac{\text{support}(\{\text{Bread, PeanutButter}\})}{\text{support}(\{\text{PeanutButter}\})} = \frac{60}{60} = 1$$

This means that the confidence of the association rule

PeanutButter=> Bread is **100%,** and this is a valid association rule.

Basic Algorithms:

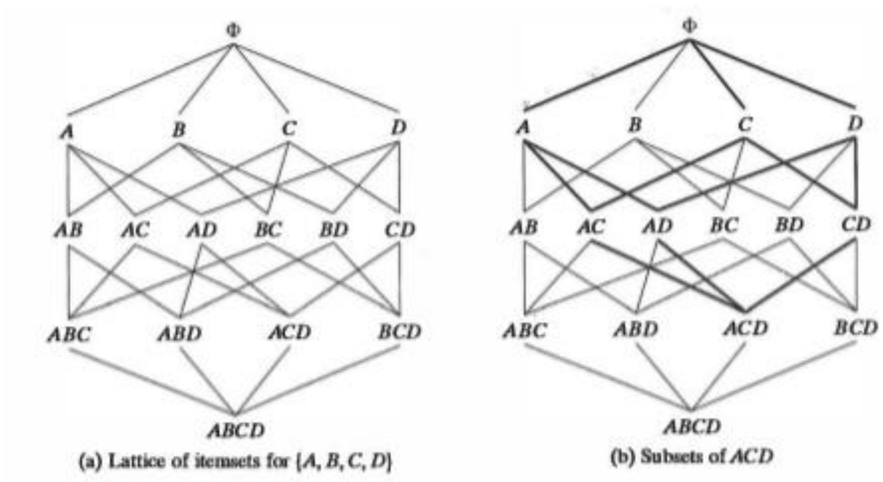
Apriori Algorithm:

- The Apriori algorithm is the most well known association rule algorithm and is used in most commercial products. It uses the following property, which we call the large itemset property: **Any subset of a large itemset must be large.**
- The large itemsets are also said to be **downward closed** because if an itemset satisfies the minimum support requirements, so do all of its subsets.

  Example:

  In this case there are four items {A, B, C, D}. The lines in the lattice represent the subset relationship, so the large itemset property says that any set in a path above an itemset must be large if the original itemset is large. The nonempty subsets of ACD are seen as {AC, AD, CD, A, C, D}. If ACD is large, so is each of these subsets. If any one of these subsets is small, then so is ACD.

Downward closure

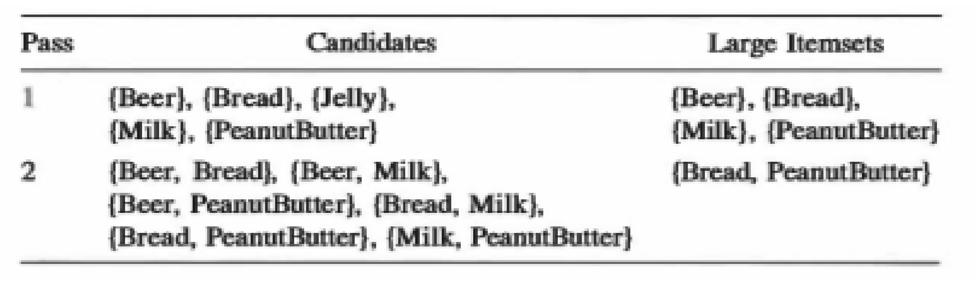


- The basic idea of the Apriori algorithm is to generate candidate itemsets of a particular size and then scan the database to count these to see if they are large. During scan i, candidates of size i, C; are counted. Only those candidates that are large are used to generate candidates for the next pass. That is L; are used to generate C;+l· An itemset is considered as a candidate only if all its subsets also are large. To generate candidates of size i + 1, joins are made of large itemsets found in the previous pass.

  Example:

  Find the candidate and large itemset with **s = 30%** and **a = 50%.**

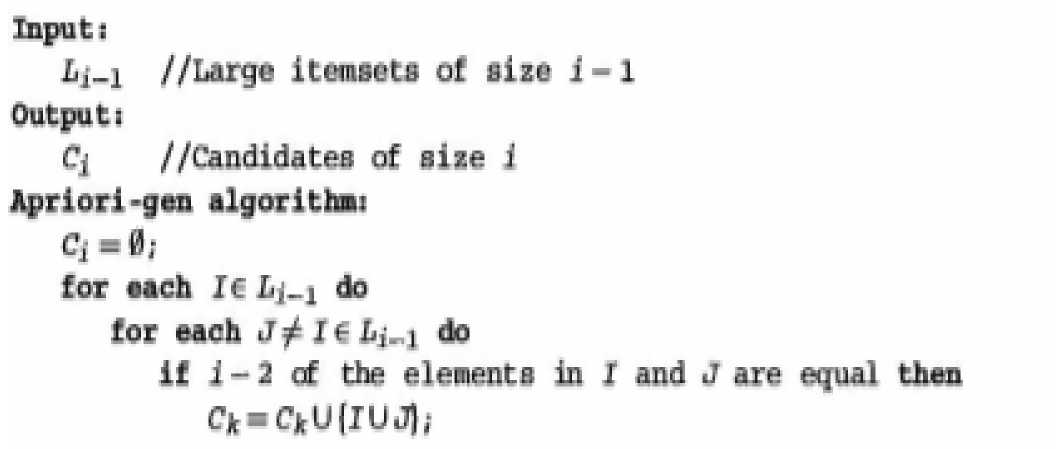
Large Itemsets using Apriori



Pass	Candidates	Large Itemsets
1	{Beer}, {Bread}, {Jelly}, {Milk}, {PeanutButter}	{Beer}, {Bread}, {Milk}, {PeanutButter}
2	{Beer, Bread}, {Beer, Milk}, {Beer, PeanutButter}, {Bread, Milk}, {Bread, PeanutButter}, {Milk, PeanutButter}	{Bread, PeanutButter}

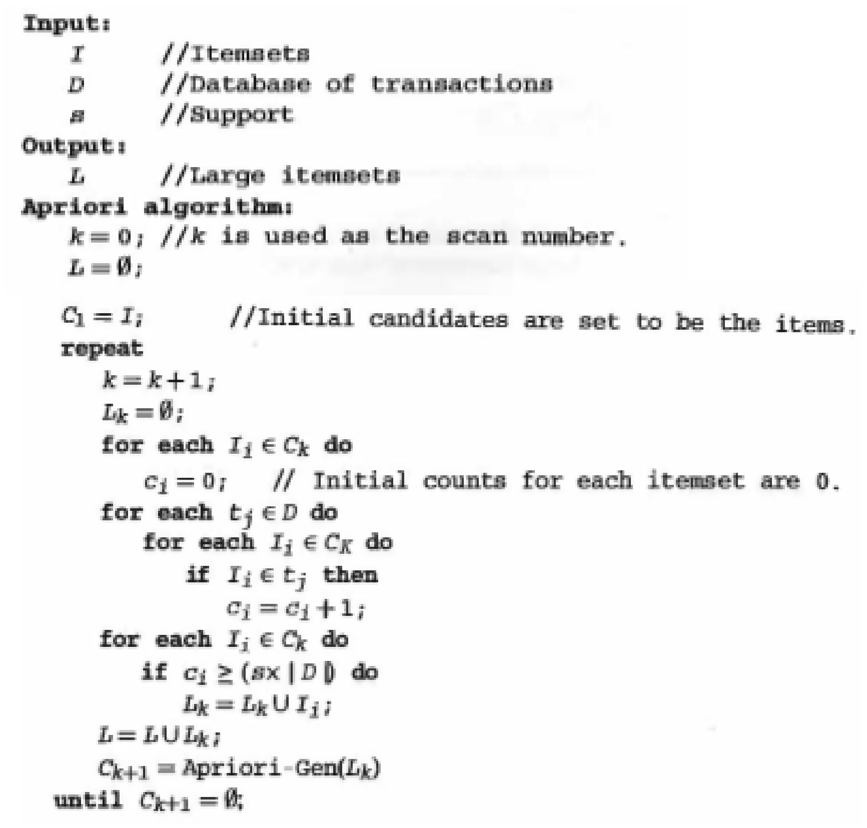
There are no candidates of size three because there is only one large itemset of size two.

Apriori-gen Algorithm (to genereate candidates)



```
Input:
   $L_{i-1}$   //Large itemsets of size $i-1$
Output:
   $C_i$    //Candidates of size $i$
Apriori-gen algorithm:
   $C_i = \emptyset$;
   for each $I \in L_{i-1}$ do
      for each $J \neq I \in L_{i-1}$ do
         if $i-2$ of the elements in $I$ and $J$ are equal then
            $C_k = C_k \cup \{I \cup J\}$;
```

Apriori Algorithm (to genereate large itemset)



```
Input:
    $I$      //Itemsets
    $D$      //Database of transactions
    $s$      //Support
Output:
    $L$      //Large itemsets
Apriori algorithm:
    $k = 0$; //$k$ is used as the scan number.
    $L = \emptyset$;
    $C_1 = I$;        //Initial candidates are set to be the items.
    repeat
       $k = k + 1$;
       $L_k = \emptyset$;
       for each $I_i \in C_k$ do
          $c_i = 0$;    // Initial counts for each itemset are 0.
       for each $t_j \in D$ do
          for each $I_i \in C_K$ do
             if $I_i \in t_j$ then
                $c_i = c_i + 1$;
       for each $I_i \in C_k$ do
          if $c_i \ge (s \times |D|)$ do
             $L_k = L_k \cup I_i$;
       $L = L \cup L_k$;
       $C_{k+1}$ = Apriori-Gen($L_k$)
   until $C_{k+1} = \emptyset$;
```

Sampling Algorithm:

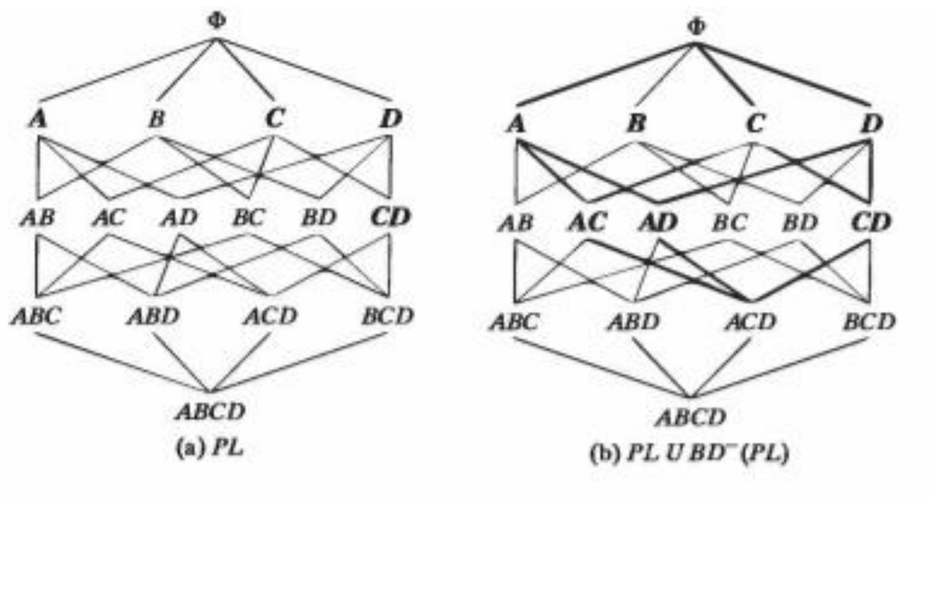
- To facilitate efficient counting of itemsets with large databases, sampling of the database may be used. The original sampling algorithm **reduces the number of database scans** to,
  - **one in the best case**
  - **two in the worst case.**
- The database sample is drawn such that it can be memory-resident. Then any algorithm, such as Apriori, is used to find the large itemsets for the sample. These are viewed as potentially large (PL) itemsets and used as candidates to be counted using the entire database**.**

- **Additional candidates** are determined by applying the negative border function, BD-, against the large itemsets from the sample. The entire set of candidates is then **C = BD$^-$(PL) U PL**.
- **Negative Border Function BD$^-$** is a generalization of the Apriori-Gen algorithm. It is defined as the minimal set of itemsets that are not in PL, but whose subsets are all in PL.

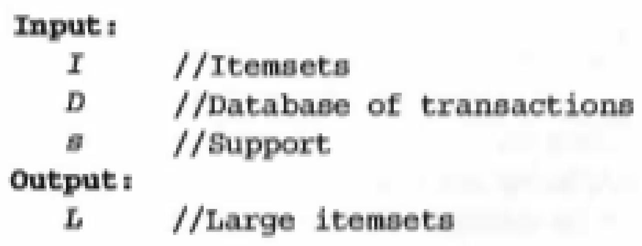
Example:

Suppose the set of items is {A, B, C, D}. The set of large itemsets found to exist in a sample of the database is PL = {A, C, D, CD}. The first scan of the entire database, then, generates the set of candidates as follows: C = BD -(PL) UPL = {B,AC,AD} U {A, C, D, CD }. Here we add AC because both A and C are in PL. Likewise we add AD. We could not have added ACD because neither AC nor AD is in PL.

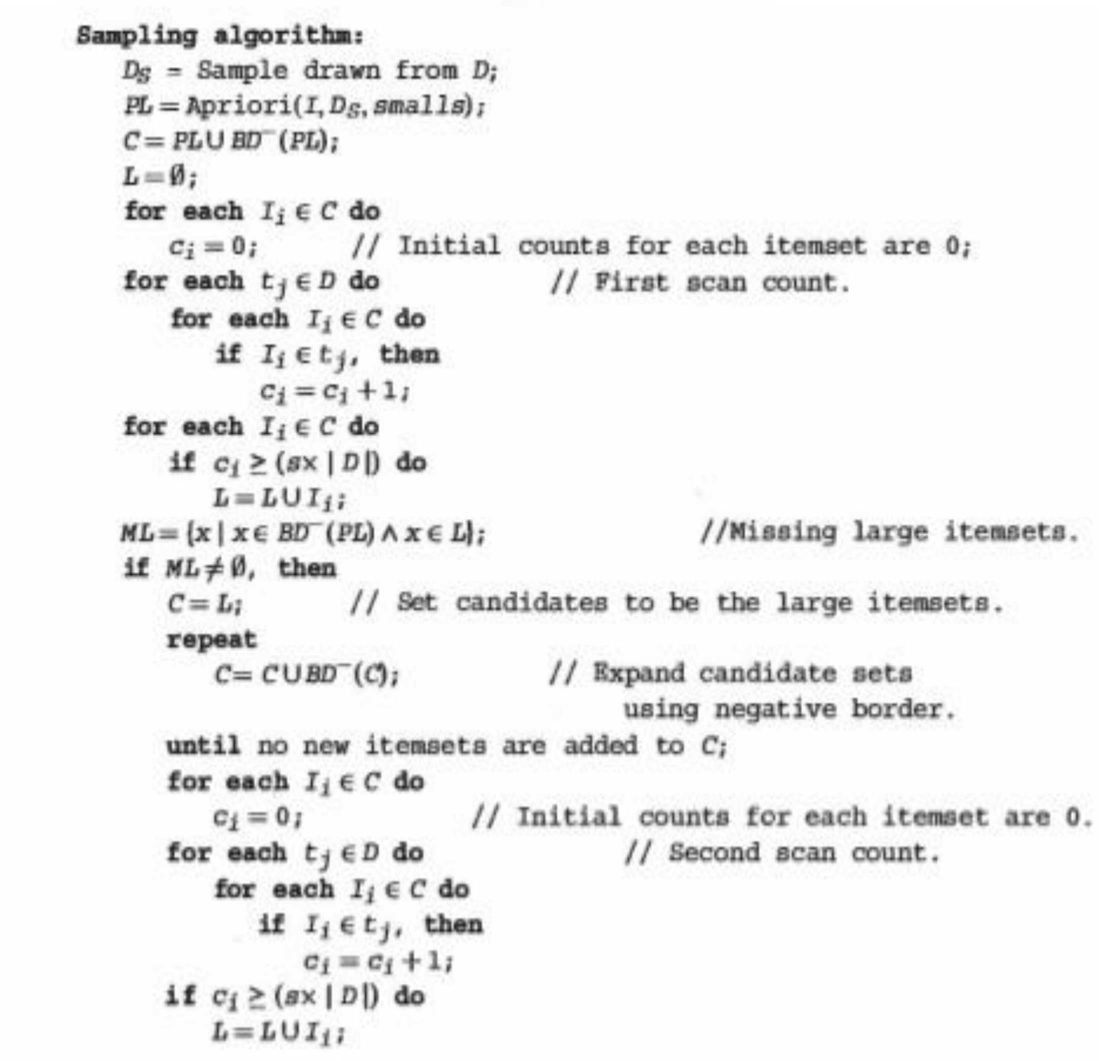
Negative border



Sampling Algorithm



```
Input:
    I       //Itemsets
    D       //Database of transactions
    s       //Support
Output:
    L       //Large itemsets
```



```
Sampling algorithm:
    $D_S$ = Sample drawn from $D$;
    $PL = \text{Apriori}(I, D_S, smalls)$;
    $C = PL \cup BD^-(PL)$;
    $L = \emptyset$;
    for each $I_i \in C$ do
       $c_i = 0$;       // Initial counts for each itemset are 0;
    for each $t_j \in D$ do            // First scan count.
       for each $I_i \in C$ do
          if $I_i \in t_j$, then
             $c_i = c_i + 1$;
    for each $I_i \in C$ do
       if $c_i \geq (s \times |D|)$ do
          $L = L \cup I_i$;
    $ML = \{x \mid x \in BD^-(PL) \land x \in L\}$;            //Missing large itemsets.
    if $ML \neq \emptyset$, then
       $C = L$;         // Set candidates to be the large itemsets.
       repeat
          $C = C \cup BD^-(C)$;          // Expand candidate sets
                                          using negative border.
       until no new itemsets are added to $C$;
       for each $I_i \in C$ do
          $c_i = 0$;            // Initial counts for each itemset are 0.
       for each $t_j \in D$ do            // Second scan count.
          for each $I_i \in C$ do
             if $I_i \in t_j$, then
                $c_i = c_i + 1$;
       if $c_i \geq (s \times |D|)$ do
          $L = L \cup I_i$;
```

Partitioning Algorithm:

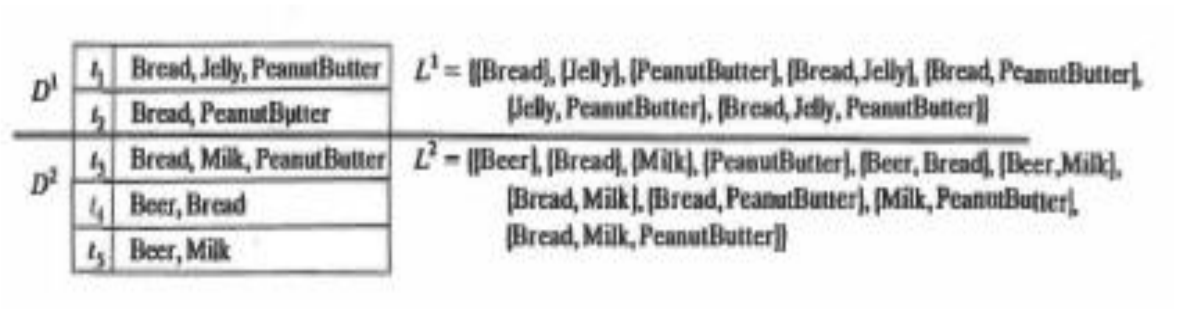
- Various approaches to generating large itemsets have been proposed based on a partitioning of the set of transactions. In this case, **D** is divided into **p** partitions. **D^1, D^2, ... , D^p**. Partitioning may improve the performance of finding large 1temsets in several ways:

- By taking advantage of the large itemset property, we know that a large itemset must be large in at least one of the partitions. This idea can help to design algorithms more efficiently than those based on looking at the entire database.
- Partitioning algorithms may be able to adapt better to limited main memory. Each partition can be created such that it fits into main memory
- By using partitioning, parallel and/or distributed algorithms can be easily created, where each partition could be handled by a separate machine.
- Incremental generation of association rules may be easier to perform by treating the current state of the database as one partition and treating the new entries as a second partition.

Partition Algorithm

```
Input:
   $I$       //Itemsets
   $D = \{D^1, D^2, \ldots, D^p\}$        //Database of transactions divided
                                            into partitions
   $s$       //Support
Output:
   $L$       //Large itemsets
Partition algorithm:
   $C = \emptyset$;
   for $i = 1$ to $p$ do  //Find large itemsets in each partition.
      $L^i = \text{Apriori}(I, D^i, s)$;
      $C = C \cup L^i$;
   $L = \emptyset$;
   for each $I_i \in C$ do
      $c_i = 0$;      //Initial counts for each itemset are 0.
   for each $t_j \in D$ do //    Count candidates during second scan.
      for each $I_i \in C$ do
         if $I_i \in t_j$, then
            $c_i = c_i + 1$;
   for each $I_i \in C$ do
      if $c_i \geq (s \times |D|)$ do
         $L = L \cup I_i$;
```

Partitioning example



D^1	t_1	Bread, Jelly, PeanutButter	$L^1 = \{\{Bread\}, \{Jelly\}, \{PeanutButter\}, \{Bread, Jelly\}, \{Bread, PeanutButter\}, \{Jelly, PeanutButter\}, \{Bread, Jelly, PeanutButter\}\}$
	t_2	Bread, PeanutButter	
D^2	t_3	Bread, Milk, PeanutButter	$L^2 = \{\{Beer\}, \{Bread\}, \{Milk\}, \{PeanutButter\}, \{Beer, Bread\}, \{Beer, Milk\}, \{Bread, Milk\}, \{Bread, PeanutButter\}, \{Milk, PeanutButter\}, \{Bread, Milk, PeanutButter\}\}$
	t_4	Beer, Bread	
	t_5	Beer, Milk	

Here the database is partitioned into two parts, the first containing two transactions and the second with three transactions. Using a **support of 10%,** the resulting large itemsets **L** ^{1} and **L** ^{2} are shown. If the items are uniformly distributed across the partitions, then a large fraction of the itemsets will be large. However, if the data are not uniform, there may be a large percentage of false candidates.

Parallel and Distributed Algorithms:

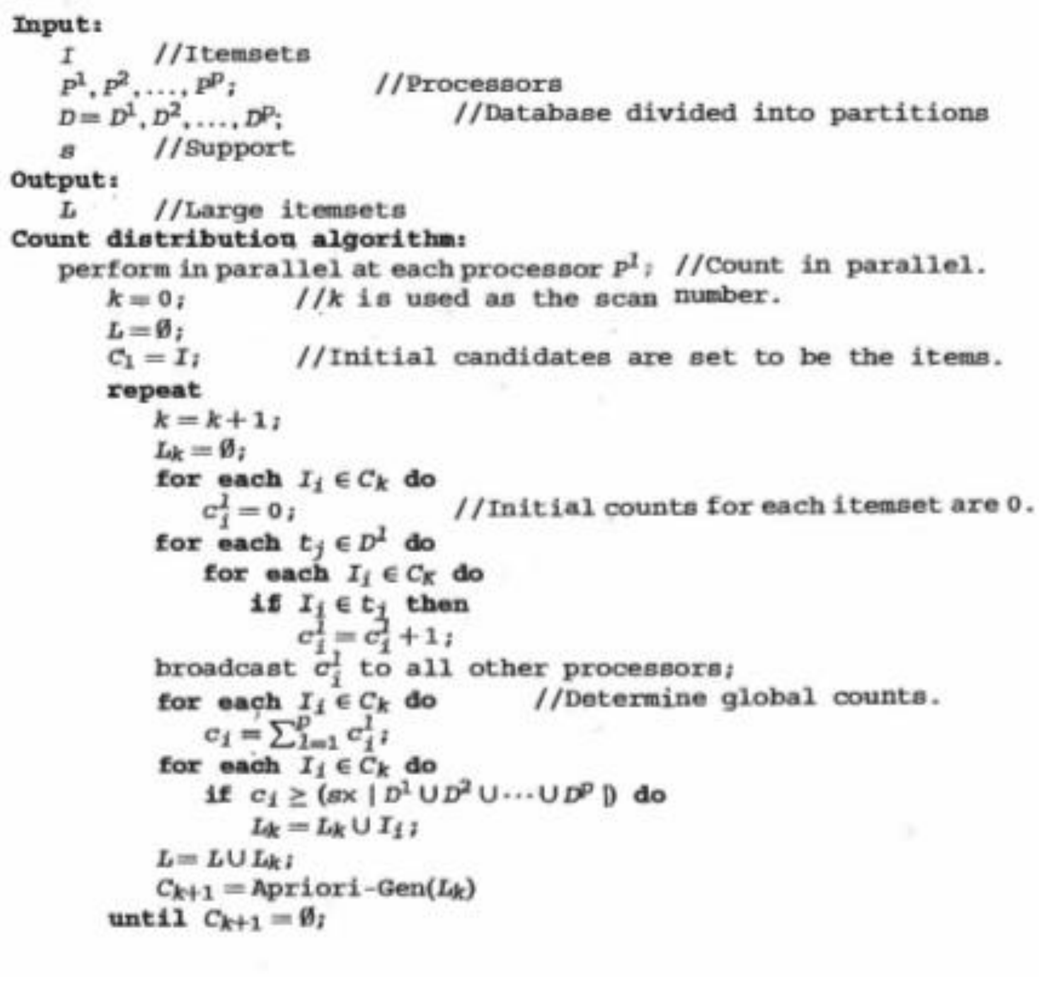
- ⇨ Most parallel or distributed association rule algorithms strive to parallelize either the **data**, known as **data parallelism**, or the **candidates**, referred to as **task parallelism.**
- ⇨ **Data parallelism Vs Task parallelism:**
    - With task parallelism, the candidates are partitioned and counted separately at each processor.
    - The data parallelism algorithms have reduced communication cost over the task, because only the initial candidates (the set of items) and the local counts at each iteration must be distributed.
    - With task parallelism, not only the candidates but also the local set of transactions must be broadcast to all other sites.

- However, the data parallelism algorithms require that memory at each processor be large enough to store all candidates at each scan.
- The task parallelism approaches can avoid this because only the subset of the candidates that are assigned to a processor during each scan must fit into memory.

Data Parallelism:

One data parallelism algorithm is the **count distribution algorithm (CDA)**. The database is divided into **p partitions**, one for each processor. Each processor counts the candidates for its data and then broadcasts its counts to all other processors. Each processor then determines the global counts. These then are used to determine the large itemsets and to generate the candidates for the next scan.

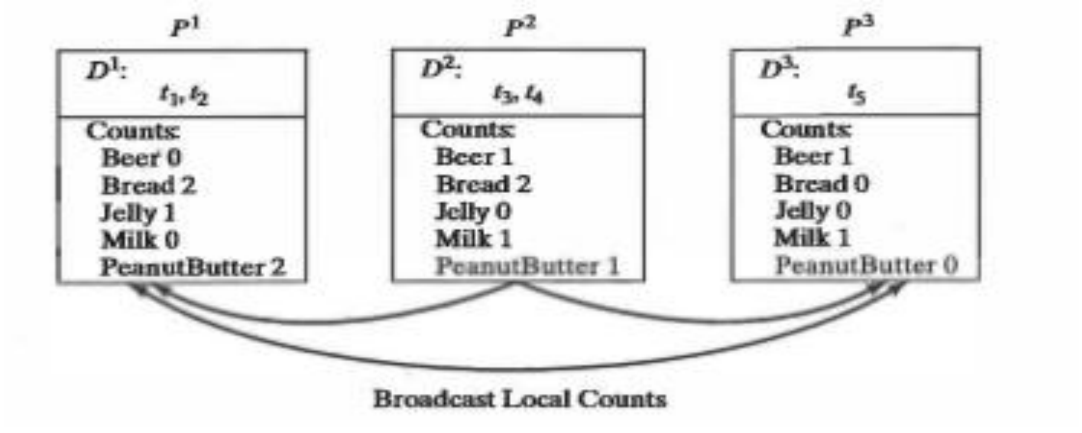
Count Distribution Algorithm



```
Input:
   $I$      //Itemsets
   $P^1, P^2, \dots, P^p$;          //Processors
   $D = D^1, D^2, \dots, D^p$;           //Database divided into partitions
   $s$      //Support
Output:
   $L$      //Large itemsets
Count distribution algorithm:
   perform in parallel at each processor $P^l$; //Count in parallel.
      $k = 0$;         //k is used as the scan number.
      $L = \emptyset$;
      $C_1 = I$;        //Initial candidates are set to be the items.
      repeat
         $k = k + 1$;
         $L_k = \emptyset$;
         for each $I_i \in C_k$ do
            $c_i^l = 0$;            //Initial counts for each itemset are 0.
         for each $t_j \in D^l$ do
            for each $I_i \in C_k$ do
               if $I_i \in t_j$ then
                  $c_i^l = c_i^l + 1$;
         broadcast $c_i^l$ to all other processors;
         for each $I_i \in C_k$ do       //Determine global counts.
            $c_i = \sum_{l=1}^{p} c_i^l$;
         for each $I_i \in C_k$ do
            if $c_i \geq (s \times |D^1 \cup D^2 \cup \cdots \cup D^p|)$ do
               $L_k = L_k \cup I_i$;
         $L = L \cup L_k$;
         $C_{k+1}$ = Apriori-Gen($L_k$)
      until $C_{k+1} = \emptyset$;
```

Example:

Data parallelism using CDA



This figure illustrates the approach used by the CDA algorithm using the grocery store data. Here there are three processors. The first two transactions are counted at **P^1**, the next two at **P^2**, and the last one at **P^3**. When the local counts are obtained, they are then broadcast to the other sites so that global counts can be generated.

Task Parallelism:

The **data distribution algorithm (DDA)** demonstrates task parallelism. Here the candidates as well as the database are partitioned among the processors. Each processor in parallel counts the candidates given to it using its local database partition.

Following our convention, we use

- **C^l_k** to indicate the candidates of size k examined at processor P^1.
- **L^l_k** are the local large k-itemsets at processor l.

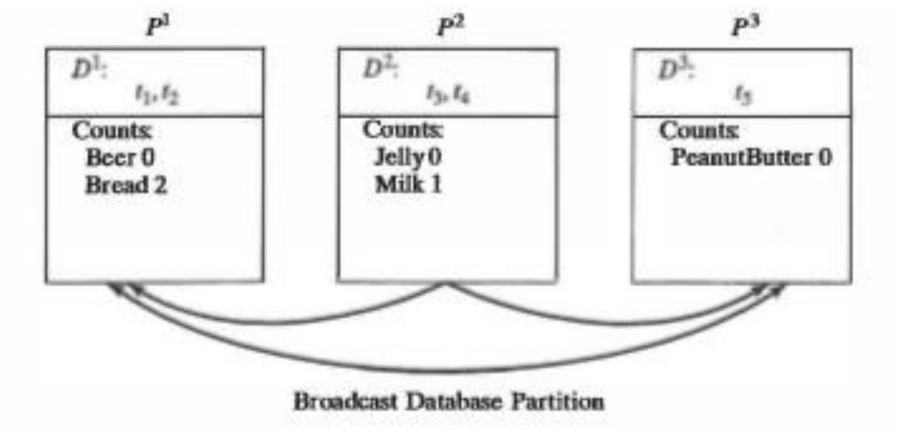
Then each processor broadcasts its database partition to all other processors. Each processor then uses this to obtain a global count for its data and broadcasts this count to all other processors. Each processor then can determine globally large itemsets and generate the next candidates. These candidates then are divided among the processors for the next scan.

Data Distribution Algorithm

```
Input:
   $I$      //Itemsets
   $P^1, P^2, \dots, P^p$;          //Processors
   $D = D^1, D^2, \dots, D^p$;          //Database divided into partitions
   $s$      //Support
Output:
   $L$      //Large itemsets
Data distribution algorithm:
   $C_1 = I$;
   for each $1 \le l \le p$ do      //Distribute size 1 candidates
                                       to each processor.
      determine $C_1^l$ and distribute to $P^l$;
   perform in parallel at each processor $P^l$ //Count in parallel.
      $k = 0$;        //$k$ is used as the scan number.
      $L = \emptyset$;
      repeat
         $k = k + 1$;
         $L_k^l = \emptyset$;
         for each $I_i \in C_k^l$ do
            $c_i^l = 0$;         //Initial counts for each itemset are 0.
         for each $t_j \in D^l$ do
            for each $I_i \in C_k^l$ do
               if $I_i \in t_j$, then
                  $c_i^l = c_i^l + 1$;          //Determine local counts.
         broadcast $D^l$ to all other processors;
         for every other processor $m \ne l$ do
            for each $t_j \in D^m$ do
               for each $I_i \in C_k^l$ do
                  if $I_i \in t_j$, then
                     $c_i^l = c_i^l + 1$;          //Determine
                                                      global counts.
            if $c_i \ge (s \times | D^1 \cup D^2 \cup \cdots \cup D^p |)$ do
               $L_k^l = L_k^l \cup I_i$;
            broadcast $L_k^l$ to all other processors;
            $L_k = L_k^1 \cup L_k^2 \cup \cdots \cup L_k^p$;          //Global large
                                                      $k$-itemsets.
            $C_{k+1} =$ Apriori-gen($L_k$)
            $C_{k+1}^l \subset C_{k+1}$;          //Determine next set of
                                                  local candidates.
         until $C_{k+1}^l = \emptyset$ ;
```

Example:

Task parallelism using DDA



This figure illustrates the approach used by the DDA algorithm using the grocery store data. Here there are three processors. **P¹** is counting **Beer** and **Bread**, **P²** is counting **Jelly** and **Milk**, and **P³** is counting **PeanutButter**. The first two transactions initially are counted at P^1, the next two at P^2, and the last one at P^3. When the local counts are obtained, the database partitions are then broadcast to the other sites so that each site can obtain a global count.

Comparing Approaches:

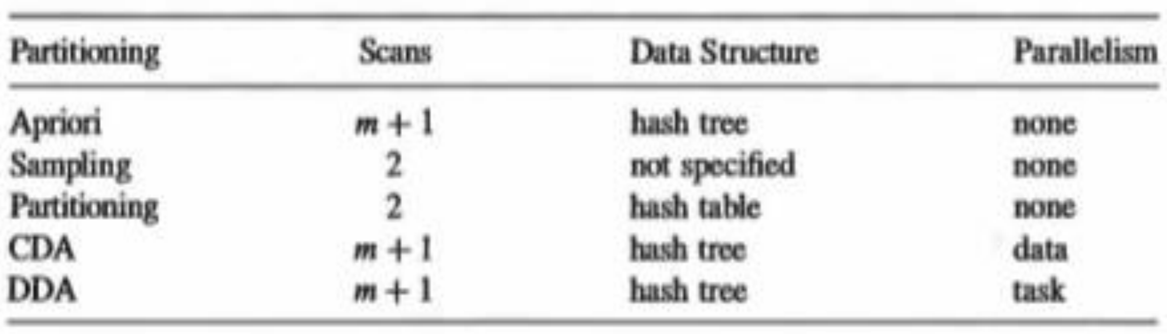
Algorithms can be classified along the following dimensions:

- **Target:** The algorithms we have examined generate all rules that satisfy a given support and confidence level. Alternatives to these types of algorithms are those that generate some subset of the algorithms based on the constraints given.
- **Type:** Algorithms may generate regular association rules or more advanced association rules.

- **Data type:** We have examined rules generated for data in categorical databases. Rules may also be derived for other types of data such as plain text.
- **Data source:** Our investigation has been limited to the use of association rules for market basket data. This assumes that data are present in a transaction. The absence of data may also be important.
- **Technique:** The most common strategy to generate association rules is that of finding large itemsets. Other techniques may also be used.
- **Itemset strategy:** Itemsets may be counted in different ways. The most naive approach is to generate all itemsets and count them. As this is usually too spaceintensive, the bottom-up approach used by Apriori, which takes advantage of the large itemset property, is the most common approach. A top-down technique could also be used.
- **Transaction strategy:** To count the itemsets, the transactions in the database must be scanned. All transactions could be counted, only a sample may be counted, or the transactions could be divided into partitions.
- **Itemset data structure:** The most common data structure used to store the candidate itemsets and their counts is a hash tree. Hash trees provide an effective technique to store, access, and count itemsets. They are efficient to search, insert, and delete itemsets. A hash tree is a multiway search tree where the branch to be taken at each level in the tree is determined by applying a hash function as opposed to comparing key values to branching points in the node. A leaf node in the hash tree contains the candidates that hash to it, stored in sorted order.

- **Transaction data structure:** Transactions may be viewed as in a flat file or as a TID list, which can be viewed as an inverted file.
- **Optimization:** These techniques look at how to improve on the performance of an algorithm given data distribution (skewness) or amount of main memory.
- **Architecture:** Sequential, parallel, and distributed algorithms have been proposed.
- **Parallelism strategy:** Both data parallelism and task parallelism have been used.

Comparison of Association Rule Algorithms



Partitioning	Scans	Data Structure	Parallelism
Apriori	$m+1$	hash tree	none
Sampling	2	not specified	none
Partitioning	2	hash table	none
CDA	$m+1$	hash tree	data
DDA	$m+1$	hash tree	task

Incremental Rules:

⇨ All algorithms discussed so far assume a static database. However, in reality we cannot assume this. With these prior algorithms, generating association rules for a **new database** state requires a **complete rerun of the algorithm**.

⇨ Several approaches have been proposed to address the issue of how to maintain the association rules as the underlying database changes. Most of the proposed approaches have addressed the issue of how to modify the association rules as inserts are performed on the database. These incremental updating approaches concentrate on determining the large itemsets for **D U db** where **D** is a **database state** and **db** are **updates** to it and where the large itemsets for D, L are known.

- One incremental approach, **fast update (FUP),** is based on the Apriori algorithm. Each iteration, k, scans both db and D with candidates generated from the prior iteration, k - 1, based on the large itemsets at that scan. In addition, we use as part of the candidate set for scan k to be L_k found in D.
- For each scan k of db, L_k plus the counts for each itemset in L_k are used as input. When the count for each item in L_k is found in db, we automatically know whether it will be large in the entire database without scanning D.

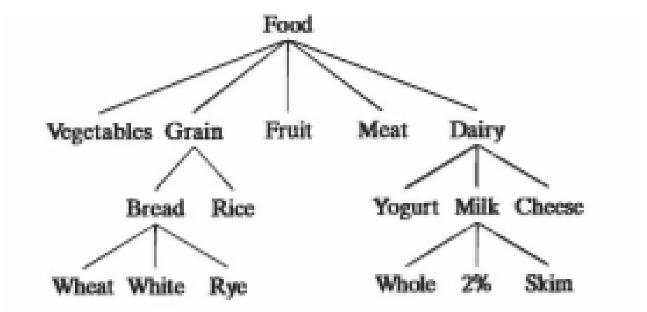
Advanced Association Rules:

Several techniques that have been proposed to generate association rules that are more complex than the basic rules.

Generalized Association Rules:

- Using a **concept hierarchy** that shows the set relationship between different items, generalized association rules allow rules at different levels.
- Association rules could be generated for any and all levels in the hierarchy. A generalized association rule, **X => Y**, is defined like a regular association rule with the restriction that no item in Y may be above any item in X.

Example: **Concept hierarchy**



This figure shows a partial concept hierarchy for food. This hierarchy shows that Wheat Bread is a type of Bread, which is a type of grain. An association rule of the . form **Bread => PeanutButter has a lower support and threshold than one of the form Grain => PeanutButter**. There obviously are more transactions containing any type of grain than transactions containing Bread. Likewise, **Wheat Bread => Peanutbutter has a lower threshold and support than Bread => PeanutButter**.

Multiple-Level Association Rules:

- A **variation of generalized rules** are **multiple-level association rules**. With multiple-level rules, itemsets may occur from any level in the hierarchy: Using a variation of the Apriori algorithm, the concept hierarchy is traversed in a top-down manner and large itemsets are generated. When large itemsets are found at level i, large itemsets are generated for level i + 1. Large k-itemsets at one level in the concept hierarchy are used as candidates to generate large k-itemsets for children at the next level.
- The minimum support required for association rules may vary based on level m the hierarchy. We would expect that the frequency of itemsets at higher levels i . s . much greater than the frequency of itemsets at lower levels. Thus, for the reduced minimum support concept, the following rules apply:
    - The minimum support for all nodes in the hierarchy at the same level is identical.
    - If $\mathbf{a_i}$ is the minimum support for level i in the hierarchy and $\mathbf{a_{i-1}}$ is the minimum support for level i - 1, then $\mathbf{a_{i-1} > a_i}$.

Quantitative Association Rules:

⇨ The association rule algorithms discussed so far assume that the data are categorical. A quantitative association rule is one that involves categorical and quantitative data. An example of a quantitative rule is:

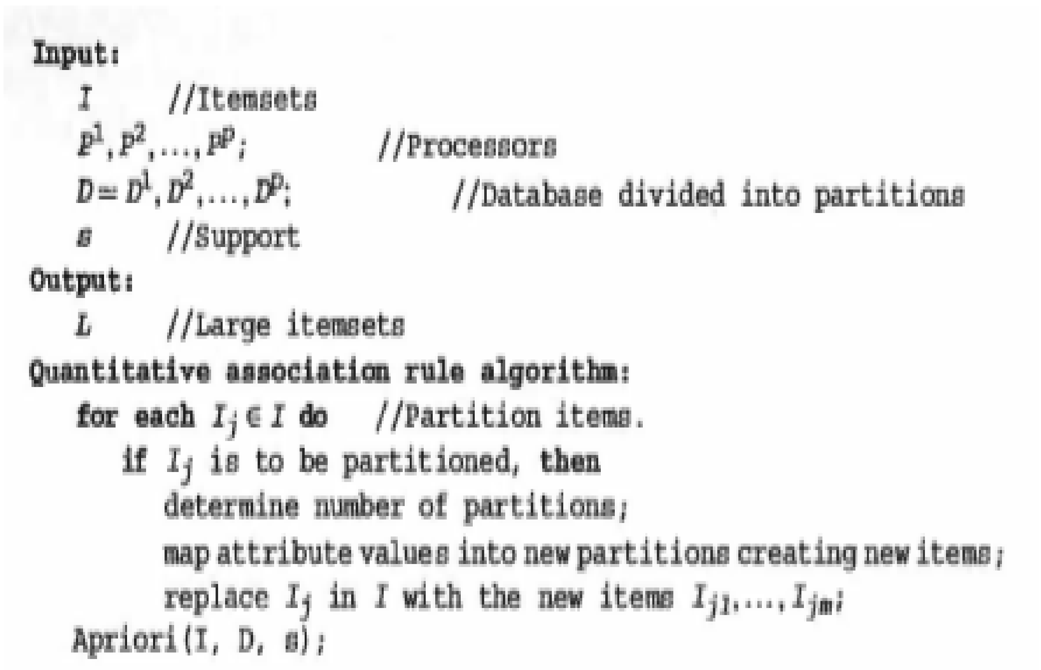
A customer buys wine for between
\$30 and \$50 a bottle => she also buys caviar

This differs from a traditional association rule such as:

A customer buys wine => she also buys caviar.

⇨ The cost quantity has been divided into an interval. In these cases, the items are not simple literals. For example, instead of having the items **{Bread, Jelly},** we might have the items **{(Bread:[O ... 1]), (Bread:(l ... 2]), (Bread:(2 ...∞)), (Jelly:[O ... 1.5]), (Jelly:(l.5 ... 3]), (Jelly:(3 ... ∞))}.**

Quantitative Association Rule Algorithm



```
Input:
   $I$      //Itemsets
   $P^1, P^2, \ldots, P^p$;         //Processors
   $D = D^1, D^2, \ldots, D^p$;           //Database divided into partitions
   $s$      //Support
Output:
   $L$      //Large itemsets
Quantitative association rule algorithm:
   for each $I_j \in I$ do   //Partition items.
      if $I_j$ is to be partitioned, then
         determine number of partitions;
         map attribute values into new partitions creating new items;
         replace $I_j$ in $I$ with the new items $I_{j1}, \ldots, I_{jm}$;
   Apriori(I, D, s);
```

- Because we have divided what was one item into several items, the minimum support and confidence used for quantitative rules may need to be lowered. The minimum support problem obviously is worse with a large number of intervals. Thus, an alternative solution would be to combine adjacent intervals when calculating support. Similarly, when there are a small number of intervals, the confidence threshold may need to be lowered.

Using Multiple Minimum Supports:

- When looking at large databases with many types of data, using one minimum support value can be a problem. Different items behave differently. It certainly is easier to obtain a given support threshold with an attribute that has only two values than it is with an attribute that has hundreds of values. It might be more meaningful to find a rule of the form

  SkimMilk => WheatBread

  with a **support of 3%** than it is to find

  Milk => Bread

  with a **support of 6%.**
- If a larger support is used, we might **miss out on generating meaningful association rules.** This problem is called the **rare item problem**. If the minimum support is too high, then rules involving items that rarely occur will not be generated. If it is set too low, then too many rules may be generated, many of which are not important.
- One approach, **MISapriori**, allows a different support threshold to be indicated for each item. Here **MIS** stands for **minimum item support.** The minimum support for a rule is the minimum of all the minimum

supports for each item in the rule. An interesting problem occurs when multiple minimum supports are used.

Example:

Suppose we have three items, {A, B, C}, with minimum supports MIS(A) = 20%, MIS(B) = 3%, and MIS(C) = 4%. Because the support for A is so large, it may be small, while both AB and AC may be large because the required support for AB min(M/S (A),MIS(B)) = 3% and AC = min(MIS (A), MIS (C)) = 4%.

Correlation Rules:

⇨ A correlation rule is defined as a set of itemsets that are correlated. The motivation for developing these correlation rules is that negative correlations may be useful.

⇨ Correlation satisfies **upward closure** in the itemset lattice. Thus**, if a set is correlated, so is every superset of it.**

Example:

Suppose there are two items, {A, B} where A ::::} B has a support of 15% and a confidence of 60%. Because these values are high, a typical association rule algorithm probably would deduce this to be a valuable rule. However, if the probability to purchase item B is 70%, then we see that the probability of purchasing B has actually gone down, presumably because A was purchased. Thus, there appears to be a negative correlation between buying A and buying B. The correlation can be expresed as,

$$\text{correlation}(A \Longrightarrow B) = \frac{P(A,B)}{P(A)\,P(B)}$$

which in this case is: **0.15/(0.25x0.7) = 0.857**. Because this correlation value is lower than 1, it indicates a **negative correlation** between A and B.

Measuring the Quality of Rules:

Some of the following metrics are used to measure the quality of an association rule,

- **Support**
- **Confidence**
- **Lift / Interest**
- **Conviction**
- **Surprise**
- **Chi-Squared statistic**

Support:

The support (s) for an association rule X => Y is the percentage of transactions in the database that contain X U Y.

Confidence or **Strength:**

The confidence or strength (a) for an association rule X => Y is the ratio of the number of transactions that contain X U Y to the number of transactions that contain X.

Lift / Interest:

With correlation rules, we saw that correlation may be used to measure the relationship between items in a rule. This may also be expressed as the lift or interest

$$\text{interest}(A \longrightarrow B) = \frac{P(A, B)}{P(A)\, P(B)}$$

This measure takes into account both P(A) and P(B). A problem with this measure is that it is symmetric. Thus, there is no difference between the value for interest **(A => B)** and the value for interest**(B => A).**

Conviction:

As with lift, conviction takes into account both P(A) and P(B). From logic we know that implication A→ B = ¬(A^ ¬B). To take into account the negation, the conviction measure inverts this ratio. The formula for conviction is,

$$\mathrm{conviction}(A \Longrightarrow B) = \frac{P(A)\ P(\neg B)}{P(A, \neg B)}$$

Conviction has a value of **1 if A and B are not related**. Rules that always hold have a value of ∞.

Surprise:

Here surprise is a measure of the **changes of correlations between items over time.** For example, if you are aware that beer and pretzels are often purchased together, it would be a surprise if this relationship actually lowered significantly. Thus, this rule **beer => pretzel** would be of interest even if the confidence decreased.

Chi-Squared statistic:

The chi squared statistic can be calculated in the following manner. Suppose the set of items is **I = {I1, I2 ... , I m}**. Given any possible itemset X, it also is viewed as a subset of the Cartesian product. The chi squared statistic is then calculated for X as,

$$\chi^2 = \sum_{X \in I} \frac{(O(X) - E[X])^2}{E[X]}$$

Example: **Contingency Table**

	B	$\bar{B}$	Total
A	15	10	25
$\bar{A}$	55	20	75
Total	70	30	100

$E[AB] = 17.5$, $E[A\bar{B}] = 7.5$, $E[\bar{A}B] = 52.5$, and $E[\bar{A}\bar{B}] = 22.5$. Using these values, we calculate **x²** for this example as,

$$\chi^2 = \sum_{X \in I} \frac{(O(X) - E[X])^2}{E[X]} = \frac{(15 - 17.5)^2}{17.5} + \frac{(10 - 7.5)^2}{7.5} + \frac{(55 - 52.5)^2}{52.5} + \frac{(20 - 22.5)^2}{22.5} = 1.587 \quad ($$

If all values were **independent**, then the chi squared statistic should be **0**.