

## **Data Mining Introduction**

Data mining is one of the most useful techniques that help entrepreneurs, researchers, and individuals to extract valuable information from huge sets of data.

Data mining is also called ***Knowledge Discovery in Database (KDD)***.

The knowledge discovery process includes Data cleaning, Data integration, Data selection, Data transformation, Data mining, Pattern evaluation, and Knowledge presentation.

### **What is Data Mining?**

- DM is the process of
  - extracting information
  - to identify patterns, trends, and useful data
  - that would allow the business to take the data-driven decision from huge sets of data
- DM is the process of
  - investigating hidden patterns of information to various perspectives
  - for categorization into useful data,
    - which is collected and assembled in particular areas such as data warehouses,
  - for efficient analysis,
  - for helping decision making and other data requirement
  - to eventually cost-cutting and generating revenue.
- DM is the act of
  - automatically searching for large stores of information
  - to find trends and patterns
  - that go beyond simple analysis procedures.
- DM utilizes complex mathematical algorithms for data segments and evaluates the probability of future events.
- DM is also called Knowledge Discovery of Data (KDD) refers to
  - the nontrivial extraction of implicit, previously unknown and potentially useful information from data stored in databases.

## DM in Business:

- DM is a process used by organizations to extract specific data from huge databases to solve business problems.
- It primarily turns raw data into useful information.



## **DM and Data Science**

- Data Mining is an activity which is a part of a broader Knowledge Discovery in Databases (KDD) Process while Data Science is a field of study just like Applied Mathematics or Computer Science.
- Often Data Science is looked upon in a broad sense while Data Mining is considered a niche.
- Some activities under Data Mining such as statistical analysis, writing data flows and pattern recognition can intersect with Data Science. Hence, Data Mining becomes a subset of Data Science.
- Machine Learning in Data Mining is used more in pattern recognition while in Data Science it has a more general use.
- Data Science and Data Mining should not be confused with Big Data Analytics and one can have both Miners and Scientists working on big datasets.

Example : Consider a scenario where you are a major retailer in India. You have 50 stores operating in 10 major cities in India and you have been operational for 10 years.

Scenario 1: You want to study the last 8 years' data to find the number of sales of sweets during festive seasons of 3 cities. It is a job of Data Mining expert. A Data Miner would probably go through historical information stored in legacy systems and employ algorithms to extract trends.

Scenario 2: You want to know which sweets have received more positive reviews. In this case, your sources of data may not be limited to databases, they could extend to social websites or customer feedback messages. It is a job of Data Scientist. A person employed as a Data Scientist is more suited to apply algorithms and conduct this socio-computational analysis.

## Data Science vs Data Mining Comparison Table

Below is the comparison table between Data Science and Data Mining.

Basis for comparison	Data Mining	Data Science
<b>What is it?</b>	A technique	An area
<b>Focus</b>	Business process	Scientific study
<b>Goal</b>	Make data more usable	Building Data-centric products for an organization
<b>Output</b>	Patterns	Varied
<b>Purpose</b>	Finding trends previously not known	Social analysis, building predictive models, unearthing unknown facts, and more
<b>Vocational Perspective</b>	Someone with a knowledge of navigating across data and statistical understanding can conduct data mining	A person needs to understand Machine Learning, Programming, info-graphic techniques and have the domain knowledge to become a data scientist
<b>Extent</b>	Data mining can be a subset of Data Science as Mining activities are part of the Data Science pipeline	Multidisciplinary – Data Science consists of Data Visualizations, Computational Social Sciences, Statistics, Data Mining, Natural Language Processing, et cetera
<b>Deals with (the type of data)</b>	Mostly structured	All forms of data – structured, semi-structured and unstructured
<b>Other less popular names</b>	Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction	Data-driven Science

## **Types of Data Mining**

Data mining can be performed on the following types of data:

### **Relational Database:**

- A relational database is a collection of multiple data sets formally organized by tables, records, and columns from which data can be accessed in various ways without having to recognize the database tables.
- Tables convey and share information, which facilitates data searchability, reporting, and organization.

### **Object-Relational Database:**

- A combination of an object-oriented database model and relational database model is called an object-relational model. It supports Classes, Objects, Inheritance, etc.
- One of the primary objectives of the Object-relational data model is to close the gap between the Relational database and the object-oriented model practices frequently utilized in many programming languages, for example, C++, Java, C#, and so on.

### **Transactional Database:**

- A transactional database refers to a database management system (DBMS) that has the potential to undo a database transaction if it is not performed appropriately. Even though this was a unique capability a very long while back, today, most of the relational database systems support transactional database activities.

### **Data warehouses:**

- A Data Warehouse is the technology that collects the data from various sources within the organization to provide meaningful business insights.
- The huge amount of data comes from multiple places such as Marketing and Finance. The extracted data is utilized for analytical purposes and helps in decision-making for a business organization.
- The data warehouse is designed for the analysis of data rather than transaction processing.

### **Data Repositories:**

- The Data Repository generally refers to a destination for data storage. However, many IT professionals utilize the term more clearly to refer to a specific kind of setup within an IT structure. For example, a group of databases, where an organization has kept various kinds of information.

## Advantages of Data Mining

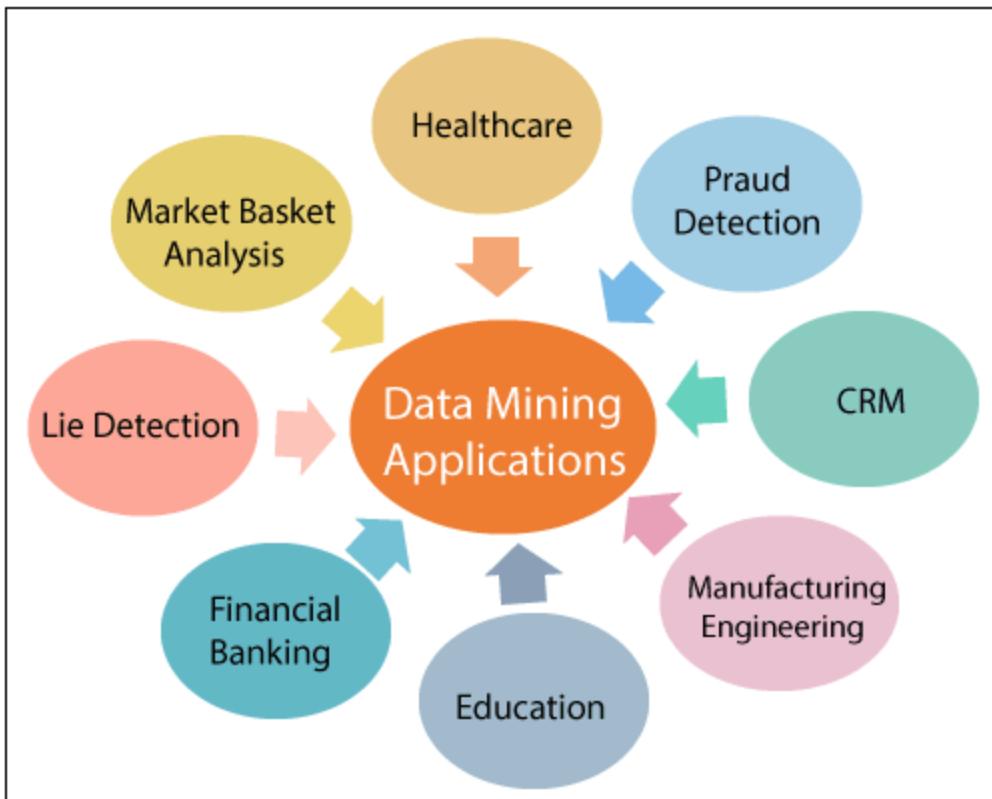
- The Data Mining technique enables organizations to obtain knowledge-based data.
- Data mining enables organizations to make lucrative modifications in operation and production.
- Compared with other statistical data applications, data mining is a cost-efficient.
- Data Mining helps the decision-making process of an organization.
- It Facilitates the automated discovery of hidden patterns as well as the prediction of trends and behaviors.
- It can be induced in the new system as well as the existing platforms.
- It is a quick process that makes it easy for new users to analyze enormous amounts of data in a short time.

## Disadvantages of Data Mining

- There is a probability that the organizations may sell useful data of customers to other organizations for money. As per the report, American Express has sold credit card purchases of their customers to other organizations.
- Many data mining analytics software is difficult to operate and needs advance training to work on.
- Different data mining instruments operate in distinct ways due to the different algorithms used in their design. Therefore, the selection of the right data mining tools is a very challenging task.
- The data mining techniques are not precise, so that it may lead to severe consequences in certain conditions.

## Data Mining Applications

- Data Mining is primarily used by organizations with intense consumer demands- Retail, Communication, Financial, marketing company, determine price, consumer preferences, product positioning, and impact on sales, customer satisfaction, and corporate profits. Data mining enables a retailer to use point-of-sale records of customer purchases to develop products and promotions that help the organization to attract the customer.



These are the following areas where data mining is widely used:

### **Data Mining in Healthcare:**

- Data mining in healthcare has excellent potential to improve the health system. It uses data and analytics for better insights and to identify best practices that will enhance health care services and reduce costs. Analysts use data mining approaches such as Machine learning, Multi-dimensional database, Data visualization, Soft computing, and statistics. Data Mining can be used to forecast patients in each category. The procedures ensure that the patients get intensive care at the right place and at the right time. Data mining also enables healthcare insurers to recognize fraud and abuse.

### **Data Mining in Market Basket Analysis:**

- Market basket analysis is a modeling method based on a hypothesis. If you buy a specific group of products, then you are more likely to buy another group of products. This technique may enable the retailer to understand the purchase behavior of a buyer. This data may assist the retailer in understanding the requirements of the buyer and altering the store's layout accordingly. Using a different analytical comparison of results between various stores, between customers in different demographic groups can be done.

### **Data mining in Education:**

- Education data mining is a newly emerging field, concerned with developing techniques that explore knowledge from the data generated from educational Environments. EDM objectives are recognized as affirming student's future learning behavior, studying the impact of educational support, and promoting learning science. An organization can use data mining to make precise decisions and also to predict the results of the student. With the results, the institution can concentrate on what to teach and how to teach.

### **Data Mining in Manufacturing Engineering:**

- Knowledge is the best asset possessed by a manufacturing company. Data mining tools can be beneficial to find patterns in a complex manufacturing process. Data mining can be used in system-level designing to obtain the relationships between product architecture, product portfolio, and data needs of the customers. It can also be used to forecast the product development period, cost, and expectations among the other tasks.

### **Data Mining in CRM (Customer Relationship Management):**

- Customer Relationship Management (CRM) is all about obtaining and holding Customers, also enhancing customer loyalty and implementing customer-oriented strategies. To get a decent relationship with the customer, a business organization needs to collect data and analyze the data. With data mining technologies, the collected data can be used for analytics.

### **Data Mining in Fraud detection:**

- Billions of dollars are lost to the action of frauds. Traditional methods of fraud detection are a little bit time consuming and sophisticated. Data mining provides meaningful patterns and turning data into information. An ideal fraud detection system should protect the data of all the users. Supervised methods consist of a collection of sample records, and these records are classified as fraudulent or non-fraudulent. A model is constructed using this data, and the technique is made to identify whether the document is fraudulent or not.

### **Data Mining in Lie Detection:**

- Apprehending a criminal is not a big deal, but bringing out the truth from him is a very challenging task. Law enforcement may use data mining techniques to investigate offenses, monitor suspected terrorist communications, etc. This technique includes text mining also, and it seeks meaningful patterns in data, which is usually unstructured text. The information collected from the previous investigations is compared, and a model for lie detection is constructed.

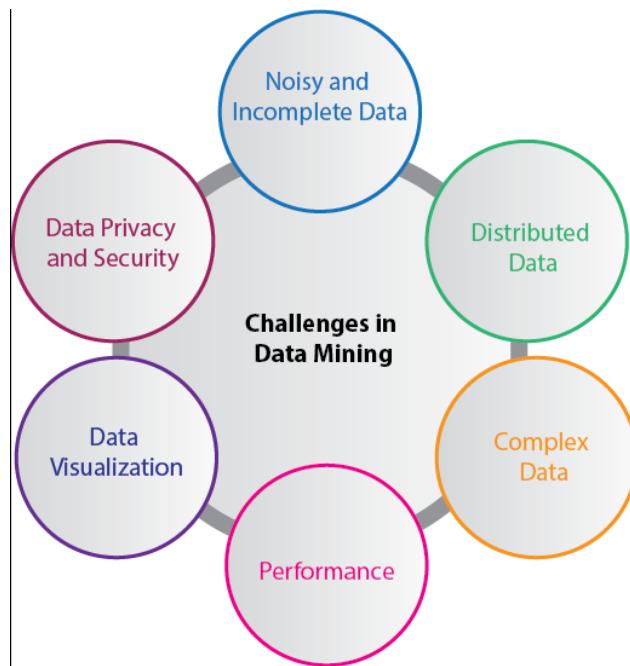
### **Data Mining Financial Banking:**

- The Digitalization of the banking system is supposed to generate an enormous amount of data with every new transaction. The data mining technique can help bankers by solving

business-related problems in banking and finance by identifying trends, casualties, and correlations in business information and market costs that are not instantly evident to managers or executives because the data volume is too large or are produced too rapidly on the screen by experts. The manager may find these data for better targeting, acquiring, retaining, segmenting, and maintain a profitable customer.

## **Challenges of Implementation in Data mining**

Although data mining is very powerful, it faces many challenges during its execution. Various challenges could be related to performance, data, methods, and techniques, etc. The process of data mining becomes effective when the challenges or problems are correctly recognized and adequately resolved.



### **Incomplete and noisy data:**

- The process of extracting useful data from large volumes of data is data mining. The data in the real-world is heterogeneous, incomplete, and noisy. Data in huge quantities will usually be inaccurate or unreliable. These problems may occur due to data measuring instrument or because of human errors. Suppose a retail chain collects phone numbers of customers who spend more than \$ 500, and the accounting employees put the information into their system. The person may make a digit mistake when entering the phone number, which results in incorrect data. Even some customers may not be willing to disclose their phone numbers, which results in incomplete data. The data could get changed due to human or system error. All these consequences (noisy and incomplete data) makes data mining challenging.

### **Data Distribution:**

- Real-worlds data is usually stored on various platforms in a distributed computing environment. It might be in a database, individual systems, or even on the internet. Practically, It is a quite tough task to make all the data to a centralized data repository mainly due to organizational and technical concerns. For example, various regional

offices may have their servers to store their data. It is not feasible to store all the data from all the offices on a central server. Therefore, data mining requires the development of tools and algorithms that allow the mining of distributed data.

### **Complex Data:**

- Real-world data is heterogeneous, and it could be multimedia data, including audio and video, images, complex data, spatial data, time series, and so on. Managing these various types of data and extracting useful information is a tough task. Most of the time, new technologies, new tools, and methodologies would have to be refined to obtain specific information.

### **Performance:**

- The data mining system's performance relies primarily on the efficiency of algorithms and techniques used. If the designed algorithm and techniques are not up to the mark, then the efficiency of the data mining process will be affected adversely.

### **Data Privacy and Security:**

- Data mining usually leads to serious issues in terms of data security, governance, and privacy. For example, if a retailer analyzes the details of the purchased items, then it reveals data about buying habits and preferences of the customers without their permission.

### **Data Visualization:**

- In data mining, data visualization is a very important process because it is the primary method that shows the output to the user in a presentable way. The extracted data should convey the exact meaning of what it intends to express. But many times, representing the information to the end-user in a precise and easy way is difficult. The input data and the output information being complicated, very efficient, and successful data visualization processes need to be implemented to make it successful.

*There are many more challenges in data mining in addition to the problems above-mentioned.*

*More problems are disclosed as the actual data mining process begins, and the success of data mining relies on getting rid of all these difficulties.*

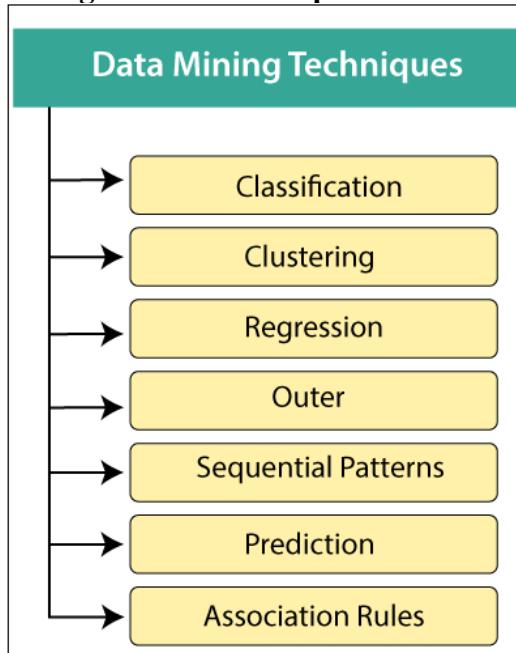
## **Data Mining Techniques**

- Data mining includes the utilization of refined data analysis tools to find previously unknown, valid patterns and relationships in huge data sets.
- These tools can incorporate statistical models, machine learning techniques, and mathematical algorithms, such as neural networks or decision trees.
- Thus, data mining incorporates analysis and prediction.
- Depending on various methods and technologies from the intersection of machine learning, database management, and statistics, professionals in data mining have devoted their objectives to better understanding how to process and make conclusions from the huge amount of data, but what are the methods they use to make it happen?
- Major major data mining techniques are association, classification, clustering, prediction, sequential patterns, and regression.

Data mining techniques can be classified by different criteria, as follows:

- i. **Classification of Data mining frameworks as per the type of data sources mined:**  
This classification is as per the type of data handled. For example, multimedia, spatial data, text data, time-series data, World Wide Web, and so on..
- ii. **Classification of data mining frameworks as per the database involved:**  
This classification based on the data model involved. For example. Object-oriented database, transactional database, relational database, and so on..
- iii. **Classification of data mining frameworks as per the kind of knowledge discovered:**  
This classification depends on the types of knowledge discovered or data mining functionalities. For example, discrimination, classification, clustering, characterization, etc. some frameworks tend to be extensive frameworks offering a few data mining functionalities together..
- iv. **Classification of data mining frameworks according to data mining techniques used:**  
This classification is as per the data analysis approach utilized, such as neural networks, machine learning, genetic algorithms, visualization, statistics, data warehouse-oriented or database-oriented, etc.  
The classification can also take into account, the level of user interaction involved in the data mining procedure, such as query-driven systems, autonomous systems, or interactive exploratory systems.

## Classification of data mining frameworks as per the kind of knowledge discovered:



### 1. Classification:

This technique is used to obtain important and relevant information about data and metadata. This data mining technique helps to classify data in different classes.

### 2. Clustering:

Clustering is a division of information into groups of connected objects. Describing the data by a few clusters mainly loses certain confine details, but accomplishes improvement. It models data by its clusters. Data modeling puts clustering from a historical point of view rooted in statistics, mathematics, and numerical analysis. From a machine learning point of view, clusters relate to hidden patterns, the search for clusters is unsupervised learning, and the subsequent framework represents a data concept. From a practical point of view, clustering plays an extraordinary job in data mining applications. For example, scientific data exploration, text mining, information retrieval, spatial database applications, CRM, Web analysis, computational biology, medical diagnostics, and much more.

In other words, we can say that Clustering analysis is a data mining technique to identify similar data. This technique helps to recognize the differences and similarities between the data.

Clustering is very similar to the classification, but it involves grouping chunks of data together based on their similarities.

### 3. Regression:

Regression analysis is the data mining process is used to identify and analyze the relationship between variables because of the presence of the other factor. It is used to define the probability of the specific variable. Regression, primarily a form of planning and modeling. For example, we

might use it to project certain costs, depending on other factors such as availability, consumer demand, and competition. Primarily it gives the exact relationship between two or more variables in the given data set.

#### 4. Association Rules:

This data mining technique helps to discover a link between two or more items. It finds a hidden pattern in the data set.

Association rules are if-then statements that support to show the probability of interactions between data items within large data sets in different types of databases. Association rule mining has several applications and is commonly used to help sales correlations in data or medical data sets.

The way the algorithm works is that you have various data, For example, a list of grocery items that you have been buying for the last six months. It calculates a percentage of items being purchased together.

These are three major measurements technique:

- **Lift:**  
This measurement technique measures the accuracy of the confidence over how often item B is purchased.  
$$(\text{Confidence}) / (\text{item B}) / (\text{Entire dataset})$$
- **Support:**  
This measurement technique measures how often multiple items are purchased and compared it to the overall dataset.  
$$(\text{Item A} + \text{Item B}) / (\text{Entire dataset})$$
- **Confidence:**  
This measurement technique measures how often item B is purchased when item A is purchased as well.  
$$(\text{Item A} + \text{Item B}) / (\text{Item A})$$

#### 5. Outer detection:

This type of data mining technique relates to the observation of data items in the data set, which do not match an expected pattern or expected behavior. This technique may be used in various domains like intrusion, detection, fraud detection, etc. It is also known as Outlier Analysis or Outlier mining. The outlier is a data point that diverges too much from the rest of the dataset. The majority of the real-world datasets have an outlier. Outlier detection plays a significant role in the data mining field. Outlier detection is valuable in numerous fields like network interruption identification, credit or debit card fraud detection, detecting outlying in wireless sensor network data, etc.

#### 6. Sequential Patterns:

The sequential pattern is a data mining technique specialized for **evaluating sequential data** to discover sequential patterns. It comprises of finding interesting subsequences in a set of sequences, where the stake of a sequence can be measured in terms of different criteria like length, occurrence frequency, etc.

In other words, this technique of data mining helps to discover or recognize similar patterns in transaction data over some time.

## 7. Prediction:

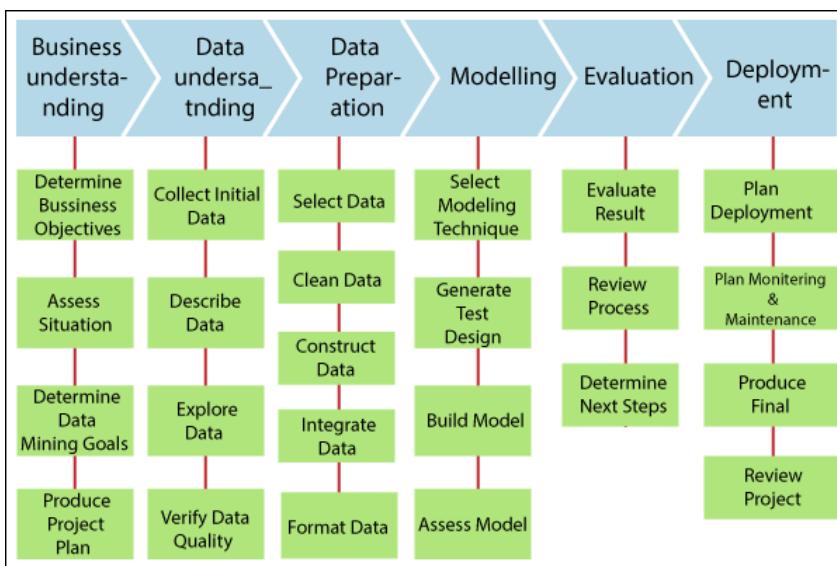
Prediction used a combination of other data mining techniques such as trends, clustering, classification, etc. It analyzes past events or instances in the right sequence to predict a future event.

next → ← prev

## Data Mining Implementation Process

Many different sectors are taking advantage of data mining to boost their business efficiency, including manufacturing, chemical, marketing, aerospace, etc. Therefore, the need for a conventional data mining process improved effectively. Data mining techniques must be reliable, repeatable by company individuals with little or no knowledge of the data mining context. As a result, a cross-industry standard process for data mining (CRISP-DM) was first introduced in 1990, after going through many workshops, and contribution for more than 300 organizations.

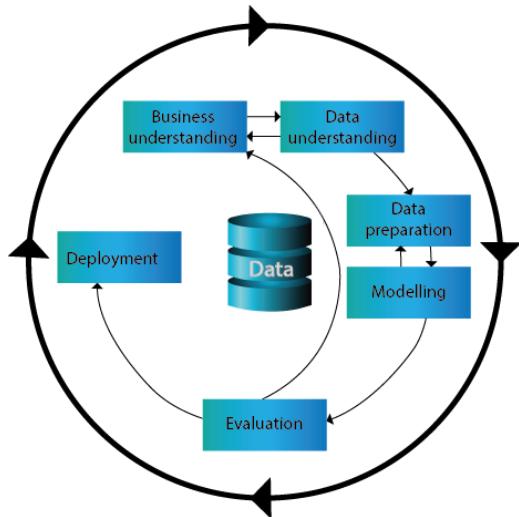
Data mining is described as a process of finding hidden precious data by evaluating the huge quantity of information stored in data warehouses, using multiple data mining techniques such as Artificial Intelligence (AI), Machine learning and statistics.



Let's examine the implementation process for data mining in details:

## The Cross-Industry Standard Process for Data Mining (CRISP-DM)

Cross-industry Standard Process of Data Mining (CRISP-DM) comprises of six phases designed as a cyclical method as the given figure:



### 1. Business understanding:

It focuses on understanding the project goals and requirements from a business point of view, then converting this information into a data mining problem afterward a preliminary plan designed to accomplish the target.

#### Tasks:

- Determine business objectives
- Access situation
- Determine data mining goals
- Produce a project plan

#### Determine business objectives:

- It Understands the project targets and prerequisites from a business point of view.
- Thoroughly understand what the customer wants to achieve.
- Reveal significant factors, at the starting, it can impact the result of the project.

#### Access situation:

- It requires a more detailed analysis of facts about all the resources, constraints, assumptions, and others that ought to be considered.

#### Determine data mining goals:

- A business goal states the target of the business terminology. For example, increase catalog sales to the existing customer.
- A data mining goal describes the project objectives. For example, It assumes how many objects a customer will buy, given their demographics details (Age, Salary, and City) and the price of the item over the past three years.

### **Produce a project plan:**

- It states the targeted plan to accomplish the business and data mining plan.
- The project plan should define the expected set of steps to be performed during the rest of the project, including the latest technique and better selection of tools.

## **2. Data Understanding:**

Data understanding starts with an original data collection and proceeds with operations to get familiar with the data, to data quality issues, to find better insight in data, or to detect interesting subsets for concealed information hypothesis.

### **Tasks:**

- Collects initial data
- Describe data
- Explore data
- Verify data quality

### **Collect initial data:**

- It acquires the information mentioned in the project resources.
- It includes data loading if needed for data understanding.
- It may lead to original data preparation steps.
- If various information sources are acquired then integration is an extra issue, either here or at the subsequent stage of data preparation.

### **Describe data:**

- It examines the "gross" or "surface" characteristics of the information obtained.
- It reports on the outcomes.

### **Explore data:**

- Addressing data mining issues that can be resolved by **querying, visualizing, and reporting**, including:
  - Distribution of important characteristics, results of simple aggregation.
  - Establish the relationship between the small number of attributes.
  - Characteristics of important sub-populations, simple statical analysis.

- It may refine the data mining objectives.
- It may contribute or refine the information description, and quality reports.
- It may feed into the transformation and other necessary information preparation.

### **Verify data quality:**

- It examines the data quality and addressing questions.

### **3. Data Preparation:**

- It usually takes more than 90 percent of the time.
- It covers all operations to build the final data set from the original raw information.
- Data preparation is probable to be done several times and not in any prescribed order.

### **Tasks:**

- Select data
- Clean data
- Construct data
- Integrate data
- Format data

#### **Select data:**

- It decides which information to be used for evaluation.
- In the data selection criteria include significance to data mining objectives, quality and technical limitations such as data volume boundaries or data types.
- It covers the selection of characteristics and the choice of the document in the table.

#### **Clean data:**

- It may involve the selection of clean subsets of data, inserting appropriate defaults or more ambitious methods, such as estimating missing information by modeling.

#### **Construct data:**

- It comprises of Constructive information preparation, such as generating derived characteristics, complete new documents, or transformed values of current characteristics.

#### **Integrate data:**

- Integrate data refers to the methods whereby data is combined from various tables, or documents to create new documents or values.

#### **Format data:**

- Formatting data refer mainly to linguistic changes produced to information that does not alter their significance but may require a modeling tool.

#### **4. Modeling:**

In modeling, various modeling methods are selected and applied, and their parameters are measured to optimum values. Some methods gave particular requirements on the form of data. Therefore, stepping back to the data preparation phase is necessary.

##### **Tasks:**

- Select modeling technique
- Generate test design
- Build model
- Assess model

##### **Select modeling technique:**

- It selects the real modeling method that is to be used. For example, decision tree, neural network.
- If various methods are applied, then it performs this task individually for each method.

##### **Generate test Design:**

- Generate a procedure or mechanism for testing the validity and quality of the model before constructing a model. For example, in classification, error rates are commonly used as quality measures for data mining models. Therefore, typically separate the data set into train and test set, build the model on the train set and assess its quality on the separate test set.

##### **Build model:**

- To create one or more models, we need to run the modeling tool on the prepared data set.

##### **Assess model:**

- It interprets the models according to its domain expertise, the data mining success criteria, and the required design.
- It assesses the success of the application of modeling and discovers methods more technically.
- It Contacts business analytics and domain specialists later to discuss the outcomes of data mining in the business context.

#### **5. Evaluation:**

- At the last of this phase, a decision on the use of the data mining results should be reached.
- It evaluates the model efficiently, and review the steps executed to build the model and to ensure that the business objectives are properly achieved.
- The main objective of the evaluation is to determine some significant business issue that has not been regarded adequately.
- At the last of this phase, a decision on the use of the data mining outcomes should be reached.

**Tasks:**

- Evaluate results
- Review process
- Determine next steps

**Evaluate results:**

- It assesses the degree to which the model meets the organization's business objectives.
- It tests the model on test apps in the actual implementation when time and budget limitations permit and also assesses other data mining results produced.
- It unveils additional difficulties, suggestions, or information for future instructions.

**Review process:**

- The review process does a more detailed evaluation of the data mining engagement to determine when there is a significant factor or task that has been somehow ignored.
- It reviews quality assurance problems.

**Determine next steps:**

- It decides how to proceed at this stage.
- It decides whether to complete the project and move on to deployment when necessary or whether to initiate further iterations or set up new data-mining initiatives.it includes resources analysis and budget that influence the decisions.

**6. Deployment:**

**Determine:**

- Deployment refers to how the outcomes need to be utilized.

**Deploy data mining results by:**

- It includes scoring a database, utilizing results as company guidelines, interactive internet scoring.

- The information acquired will need to be organized and presented in a way that can be used by the client. However, the deployment phase can be as easy as producing. However, depending on the demands, the deployment phase may be as simple as generating a report or as complicated as applying a repeatable data mining method across the organizations.

### **Tasks:**

- Plan deployment
- Plan monitoring and maintenance
- Produce final report
- Review project

### **Plan deployment:**

- To deploy the data mining outcomes into the business, takes the assessment results and concludes a strategy for deployment.
- It refers to documentation of the process for later deployment.

### **Plan monitoring and maintenance:**

- It is important when the data mining results become part of the day-to-day business and its environment.
- It helps to avoid unnecessarily long periods of misuse of data mining results.
- It needs a detailed analysis of the monitoring process.

### **Produce final report:**

- A final report can be drawn up by the project leader and his team.
- It may only be a summary of the project and its experience.
- It may be a final and comprehensive presentation of data mining.

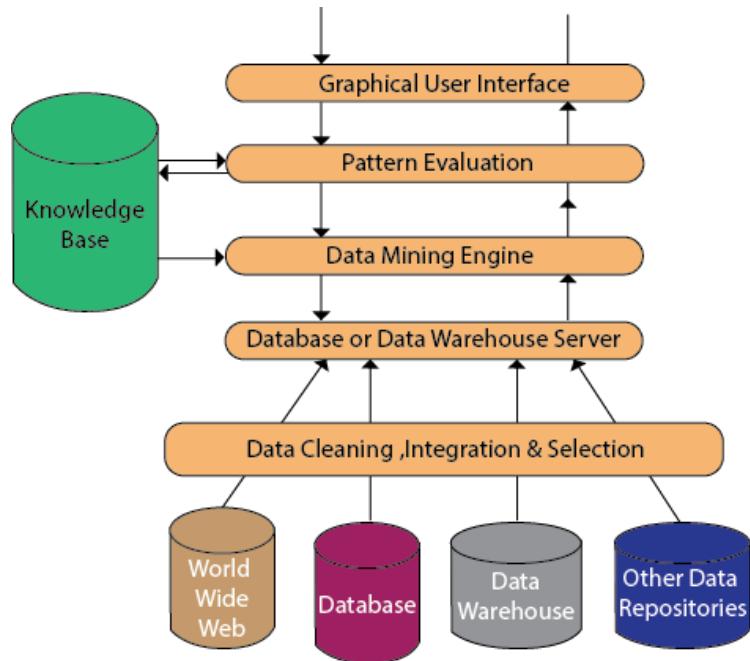
### **Review project:**

- Review projects evaluate what went right and what went wrong, what was done wrong, and what needs to be improved.

## Data Mining Architecture

Data mining is a significant method where previously unknown and potentially useful information is extracted from the vast amount of data. The data mining process involves several components, and these components constitute a data mining system architecture.

The significant components of data mining systems are a data source, data mining engine, data warehouse server, the pattern evaluation module, graphical user interface, and knowledge base.



### Data Source:

The actual source of data is the Database, data warehouse, World Wide Web (WWW), text files, and other documents. You need a huge amount of historical data for data mining to be successful. Organizations typically store data in databases or data warehouses. Data warehouses may comprise one or more databases, text files spreadsheets, or other repositories of data. Sometimes, even plain text files or spreadsheets may contain information. Another primary source of data is the World Wide Web or the internet.

### Different processes:

Before passing the data to the database or data warehouse server, the data must be cleaned, integrated, and selected. As the information comes from various sources and in different formats, it can't be used directly for the data mining procedure because the data may not be complete and accurate. So, the first data requires to be cleaned and unified. More information than needed will be collected from various data sources, and only the data of interest will have to be selected and passed to the server. These procedures are not as easy as we think. Several methods may be performed on the data as part of selection, integration, and cleaning.

### **Database or Data Warehouse Server:**

The database or data warehouse server consists of the original data that is ready to be processed. Hence, the server is cause for retrieving the relevant data that is based on data mining as per user request.

### **Data Mining Engine:**

The data mining engine is a major component of any data mining system. It contains several modules for operating data mining tasks, including association, characterization, classification, clustering, prediction, time-series analysis, etc.

In other words, we can say data mining is the root of our data mining architecture. It comprises instruments and software used to obtain insights and knowledge from data collected from various data sources and stored within the data warehouse.

### **Pattern Evaluation Module:**

The Pattern evaluation module is primarily responsible for the measure of investigation of the pattern by using a threshold value. It collaborates with the data mining engine to focus the search on exciting patterns.

This segment commonly employs stake measures that cooperate with the data mining modules to focus the search towards fascinating patterns. It might utilize a stake threshold to filter out discovered patterns. On the other hand, the pattern evaluation module might be coordinated with the mining module, depending on the implementation of the data mining techniques used. For efficient data mining, it is abnormally suggested to push the evaluation of pattern stake as much as possible into the mining procedure to confine the search to only fascinating patterns.

### **Graphical User Interface:**

The graphical user interface (GUI) module communicates between the data mining system and the user. This module helps the user to easily and efficiently use the system without knowing the complexity of the process. This module cooperates with the data mining system when the user specifies a query or a task and displays the results.

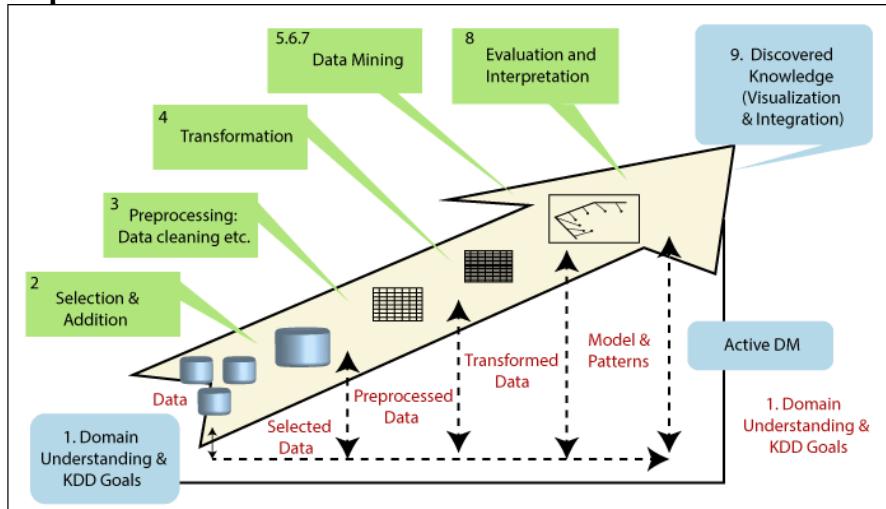
### **Knowledge Base:**

The knowledge base is helpful in the entire process of data mining. It might be helpful to guide the search or evaluate the stake of the result patterns. The knowledge base may even contain user views and data from user experiences that might be helpful in the data mining process. The data mining engine may receive inputs from the knowledge base to make the result more accurate and reliable. The pattern assessment module regularly interacts with the knowledge base to get inputs, and also update it.

## KDD and DM

**Data Mining** also known as Knowledge Discovery in Databases, refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data stored in databases.

### Steps Involved in KDD Process:



#### 1.a Define the objective:

- This is the initial preliminary step.
- It develops the scene for understanding what should be done with the various decisions like transformation, algorithms, representation, etc.
- The individuals who are in charge of a KDD venture need to understand and characterize the objectives of the end-user and the environment in which the knowledge discovery process will occur ( involves relevant prior knowledge).

#### 1.b Preparing DataSet

- Discovering data accessibility, obtaining important data,
- Collecting and integrating all the data for knowledge discovery
- This process is important because of DM learns and discovers from the accessible data.
- If some significant attributes are missing, at that point, then the entire study may be unsuccessful from this respect, the more attributes are considered.
- On the other hand, to organize, collect, and operate advanced data repositories is expensive. The interactive and iterative aspect of the KDD is taking place where process will begin with best available data sets and later expands and observes the impact in terms of knowledge discovery and modeling.

#### 2. Data Cleaning:

Data cleaning is defined as removal of incorrect, incomplete, irrelevant, duplicate or irregularly formatted information.

- 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**.
3. **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.
4. **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.

**NOTE:** After this step, data reliability is improved. It incorporates data clearing, for example, Handling the missing quantities and removal of noise or outliers. It might include complex statistical techniques or use a Data Mining algorithm in this context. For example, when one suspects that a specific attribute of lacking reliability or has many missing data, at this point, this attribute could turn into the objective of the Data Mining supervised algorithm. A prediction model for these attributes will be created, and after that, missing data can be predicted. The expansion to which one pays attention to this level relies upon numerous factors. Regardless, studying the aspects is significant and regularly revealing by itself, to enterprise data frameworks.

5. **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.

In this stage, the creation of appropriate data for Data Mining is prepared and developed. Techniques here incorporate dimension reduction( for example, feature selection and extraction and record sampling), also attribute transformation(for example, discretization of numerical attributes and functional transformation). This step can be essential for the success of the entire KDD project, and it is typically very project-specific. For example, in medical assessments, the quotient of attributes may often be the most significant factor and not each one by itself. In business, we may need to think about impacts beyond our control as well as efforts and transient issues. For example, studying the impact of advertising accumulation. However, if we do not utilize the right transformation at the starting, then we may acquire an amazing effect that insights to us about the transformation required in the next iteration. Thus, the KDD process follows upon itself and prompts an understanding of the transformation required.

6. **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**.

NOTE:

- **Prediction and description**

We are now prepared to decide on which kind of Data Mining to use, for example, classification, regression, clustering, etc. This mainly relies on the KDD objectives, and also on the previous steps. There are two significant objectives in Data Mining, the first one is a prediction, and the second one is the description. Prediction is usually referred to as supervised Data Mining, while descriptive Data Mining incorporates the unsupervised and visualization aspects of Data Mining. Most Data Mining techniques depend on inductive learning, where a model is built explicitly or implicitly by generalizing from an adequate number of preparing models. The fundamental assumption of the inductive approach is that the prepared model applies to future cases. The technique also takes into account the level of meta-learning for the specific set of accessible data.

- **Selecting the Data Mining algorithm**

Having the technique, we now decide on the strategies. This stage incorporates choosing a particular technique to be used for searching patterns that include multiple inducers. For example, considering precision versus understandability, the previous is better with neural networks, while the latter is better with decision trees. For each system of meta-learning, there are several possibilities of how it can be succeeded. Meta-learning focuses on clarifying what causes a Data Mining algorithm to be fruitful or not in a specific issue. Thus, this methodology attempts to understand the situation under which a Data Mining algorithm is most suitable. Each algorithm has parameters and strategies of leaning, such as ten folds cross-validation or another division for training and testing.

- **Utilizing the Data Mining algorithm**

At last, the implementation of the Data Mining algorithm is reached. In this stage, we may need to utilize the algorithm several times until a satisfying outcome is obtained. For example, by turning the algorithms control parameters, such as the minimum number of instances in a single leaf of a decision tree.

7. **Pattern Evaluation:** Pattern Evaluation is defined 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.

Note :

In this step, we assess and interpret the mined patterns, rules, and reliability to the objective characterized in the first step. Here we consider the preprocessing steps as for their impact on the Data Mining algorithm results. For example, including a feature in step 4, and repeat from there. This step focuses on the comprehensibility and utility of the induced model. In this step, the identified knowledge is also recorded for further use. The last step is the use, and overall feedback and discovery results acquire by Data Mining.

8. **Knowledge representation:** Knowledge representation is defined as technique which utilizes visualization tools to represent data mining results.

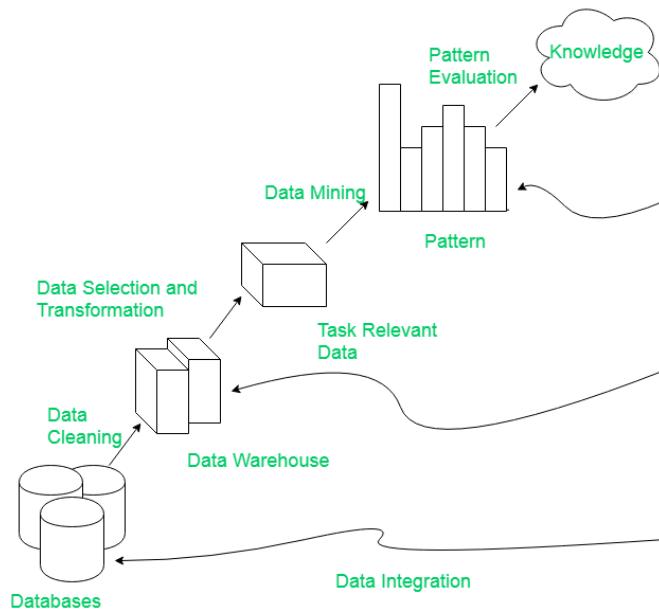
- Generate **reports**.
- Generate **tables**.
- Generate **discriminant rules, classification rules, characterization rules**, etc.

**Note:**

- KDD is an **iterative process** where evaluation measures can be enhanced, mining can be refined, new data can be integrated and transformed in order to get different and more appropriate results.
- **Preprocessing of databases** consists of **Data cleaning** and **Data Integration**
- **Using the discovered knowledge**
  - Now, we are prepared to include the knowledge into another system for further activity. The knowledge becomes effective in the sense that we may make changes to the system and measure the impacts. The accomplishment of this step decides the effectiveness of the whole KDD process. There are numerous challenges in this step, such as losing the "laboratory conditions" under which we have worked. For example, the knowledge was discovered from a certain static depiction, it is usually a set of data, but now the data becomes dynamic. Data structures may change certain quantities that become unavailable, and the data domain might be modified, such as an attribute that may have a value that was not expected previously.

## Data Mining: Data Warehouse Process

KDD incorporating Data Warehouse



**Data Warehouses** are information gathered from multiple sources and saved under a schema that is living on the identical site. It is made with the aid of diverse techniques inclusive of the following processes :

### 1. Data Cleanup:

### 2. Data Integration:

Data integration is the process of integrating data from different assets right into a unified view. The integration manner starts with a startup and includes steps which include refinement, ETL mapping, and conversion. Data integration ultimately permits analytics tools to create powerful and cheap enterprise intelligence.

In a typical data integration procedure, the client sends a request for information to the master server. The master server prepares the vital records from internal and external assets. Extracts facts from sources and then integrates them into a single information set. It is then returned again to the client for use.

### 3. Data Transformation:

Data transformation is the manner of converting information from one layout or shape to another layout or structure. Data Transformation is critical for features which include data integration and information management. Data transformation has different capabilities: you could alternate the records types relying on the desires of your project, enrich or aggregate the records through casting off invalid or duplicate data.

Generally, the technique consists of two stages.

In the **first step**, you should:

- Perform an information search that identifies assets and data types.
- Determine the structure and information changes that occur.

- Mapping data to discover how character fields are mapped, edited, inserted, filtered, and stored.

In the **second step**, you must:

- Extract data from the original source. The size of the supply can range from a connected tool to a dependable useful resource along with a database or streaming resources, including telemetry or logging files from clients who use your web application.
- Send data to the target site.
- The target may be a database or a data warehouse that manages structured and unstructured records.

#### **4. Loading Data:**

Data loading is the manner of copying and loading data from a report, folder or application to a database or similar utility. This is usually done via copying digital data from the source and pasting or loading the records into a data warehouse or processing tools.

Data-loading is used in data extraction and loading methods. Typically, such information is loaded in a different format than the original location of the source.

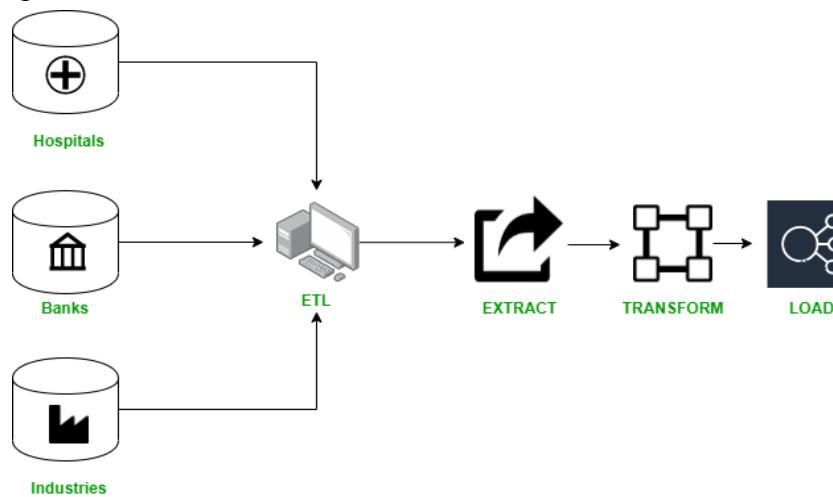
#### **5. Data Refreshing:**

In this process, the data stored in the warehouse is periodically refreshed so that they maintain its integrity.

A data warehouse is a model of Multidimensional data structures that are known as “Data Cube” in which every dimension represents an attribute or different set of attributes in the schema of the data and each cell is used to store the value. Data is gathered from various sources such as Hospitals, Banks, Organizations and many more and goes through a process called ETL(Extract, Transform, Load).

1. **Extract:** This process reads the data from the database of various sources.
2. **Transform:** It transforms the data stored inside the databases into data cubes so that it can be loaded inside the warehouse.
3. **Load:** It is a process of writing the transformed data into the data warehouse.

This process can be seen in the illustration below:



## Data Quality Definitions and Measures

There are many definitions of data quality, in general, data quality is the assessment of how much the data is usable and fits for the following DM task

Having accurate and business-ready data is an absolutely integral component to ensure that DM application users (companies) do not experience the negative impacts that can accompany “bad” or “dirty” data.

There are five components that will ensure data quality; completeness, consistency, accuracy, validity, and timeliness. When each of these components are properly executed, it will result in high-quality data.

1. **Accuracy:** Data collected is correct, relevant and accurately represents what it should. Accuracy is more challenging to remedy than data completeness and consistency. The Inaccuracy may be resulted from Incorrect Logging.

### Incorrect logging

**Note : “Inaccuracy is not Noise” :** In DM Process, people use the term “Noise” for exceptional behavior - not for incorrect logging. If DM algorithm is said to be able to deal with noise, then it can identify (Mine) low-frequent behavior with the help of the main process flow only. It is impossible for discovery algorithms to distinguish incorrect logging from exceptional events.

**What incorrect logging means is that the *recorded data is wrong*.**

Here are two true stories of incorrect data:

- Example : 1
  - In an ERP system, data entries from invoice documents had been scanned automatically. However, because of a mistake in the scanning procedure the invoice ID was interpreted as the invoice date for some of the cases. As a result, activities with a timestamp of the year 2020 appeared in the log data.
- Example 2:
  - In a process improvement project in a hospital the **data showed low utilization rates** for two specific wards. Actually the hospital closed 2 wards but had to re-open them again shortly afterwards, which resulted in the “missing information”

for those two wards. The problem was also due to the manual entries by the staff one day later so the entry date were not matching to the closing and reopening dates of the wards.

- The other inaccuracy may be resulted due to inconsistency in collection process. Eg. One sensor is sensing at the end of time cycle and another is sensing at the begining of the time cycle
2. **Completeness:** Ensuring there are no gaps in the data from what was supposed to be collected and what was actually collected. The Noncompleteness may be result from Insufficient logging

### **Insufficient logging**

While incorrect logging is about wrong data, insufficient logging is about missing data.

Typical problems with missing data are:

- Fields in the database of the information system are simply overwritten. So, old entries are lost and the database only provides information about the current status, but not the overall history of what happened in the past.
- Some systems employ “batch logging” procedures, where, for example, activities are logged once a day (all at once). This way, all changes in-between are lost as well as the ordering of what happened when cannot be reconstructed anymore.
  - Typical OLAP and data mining techniques do not require the whole history of a process, and therefore data warehouses often do not contain all the data that is needed for process mining.
- Another problem is that, ironically, by logging too much data sometimes there is not enough data. I have heard of more than one SAP or enterprise service bus system that does not keep logs longer than one month for the sheer amount of data that would accumulate otherwise. But processes often run longer than one month and, therefore, logs from a larger timeframe would be needed.
- Finally, for specific types of analysis additional data is required. For example, to calculate execution times for activities both start and completion timestamps must be available in the data. For an organizational analysis, the person or the department that performed an activity should be included in the log extract, and so forth.

3. **Consistency:** The types of data must align with the expected versions of the data being collected. Inconsistency means Violation of semantic rules defined over the database.

One of the biggest challenges can be to find the right information and to understand what it means.

In fact, figuring out the semantics of existing IT logs can be anything between really easy and incredibly complicated. It largely depends on how distant the logs are from the actual business logic. For example, the performed business process steps may be recorded directly with their activity name, or you might need a mapping between some kind of cryptic action code and the actual business activity.

**Sol1:** Normally menu driven Data Entry will help preserving semantics

Sol 2: Predefining log time for extracting data

4. **Validity:** Validity is derived from the process instead of the final result. When there is a need to fix invalid data, there is an issue with the process rather than the results (Invalid data will affect results in absolute meaning). This makes it a little trickier to resolve. Any change in the storage-structure (domain, range, normalization etc) creates the need to validate the data before DM process. Co-relationship plays major role in maintaining validity..

### Correlation

Because process mining is based on the *history* of a process, the individual process instances need to be reconstructed from the log data. Correlation is about stitching everything together in the correct way:

- Business processes often span multiple IT systems, and usually each IT system has its own local IDs. One needs to correlate these local process IDs to combine log fragments from the different systems (local ID from system No. 1 and local ID from system No. 2) in order to get a full picture of the process from start to end.
- Even within the same system correlation may be necessary. For example, in an ERP purchase-to-pay process purchase orders are identified by purchase order IDs and later on the invoices are characterized by invoice IDs. To get an end-to-end process perspective, the corresponding purchase order IDs and invoice IDs need to be matched.
- Sometimes, there are hierarchical processes and then activity instances need to be distinguished to correlate lower-level events that belong to these (activity) sub processes.

5. **Timeliness:** The data should be received at the expected time in order for the information to be utilized efficiently. Anything slower becomes an inadequate source of information. With real time data and analytics, companies are better equipped to make more effective and informed decisions. There is a pressing need to eliminate the lag time between when a survey is completed in the field and when it is received. Timing is Important to maintain the timeliness.

Precisely because process mining evaluates the history of performed process instances, the timing is very important for ordering the events within each sequence. If the timestamps are wrong or not precise enough, then it is difficult to create the correct order of events in the history.

**Some of the problems seen with timestamps are:**

- Timestamp resolution is too low. For example, only the date of a performed activity (but not the time) is recorded. But even if the time is recorded, it may be necessary to record it at least with millisecond accuracy if many events follow each other in automated systems.
- Different timestamp granularities on different systems. For example, the timestamps in one system may be rounded to minutes. Another system (which is also executing a part of the process) records events with 1-second resolution. When put together, the order of some of the events may be wrong due to the granularity difference.
- Different clocks on different systems. If multiple computers record data, then these computers can have different system clocks. In the merged log, these time differences then create problems, since they destroy the correct order of events.

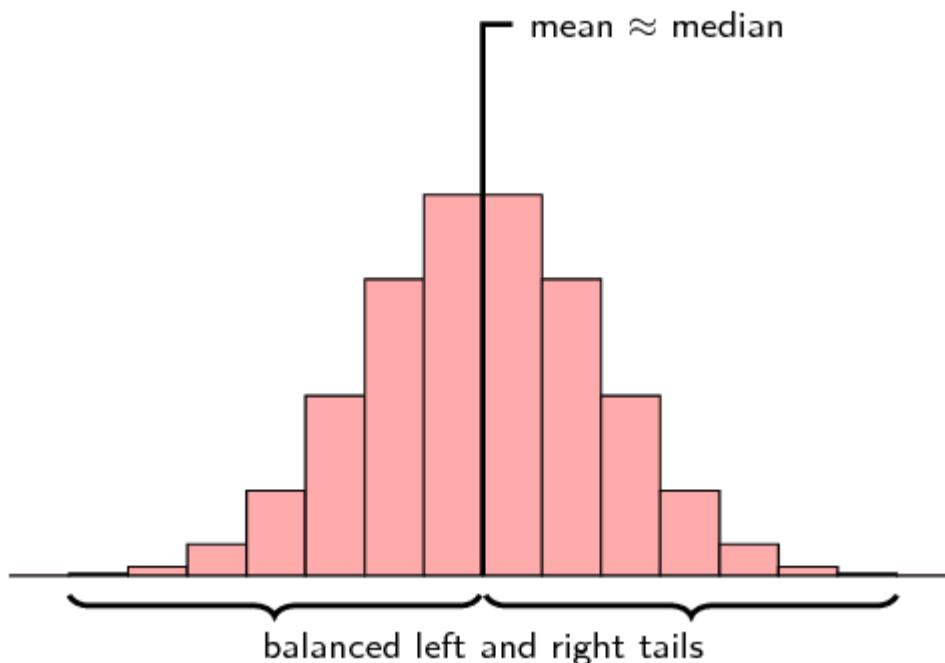
Ideally, timestamps should be precise, not be rounded up or down, and synchronized (if there are multiple systems). If there are differences, it may help to work with offsets. If too many events have the same timestamp, one can try to use the original sequence of events.

## Data Distribution representation using Histogram

### Normal Distribution

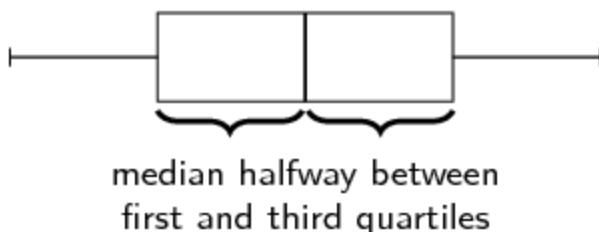
#### Symmetric and skewed data (EMBKD)

A symmetric distribution is one where the left and right hand sides of the distribution are roughly equally balanced around the mean. The histogram below shows a typical symmetric distribution.



For symmetric distributions, the mean is approximately equal to the median. The **tails** of the distribution are the parts to the left and to the right, away from the mean. The tail is the part where the counts in the histogram become smaller. For a symmetric distribution, the left and right tails are equally balanced, meaning that they have about the same length.

The figure below shows the box and whisker diagram for a typical symmetric data set.

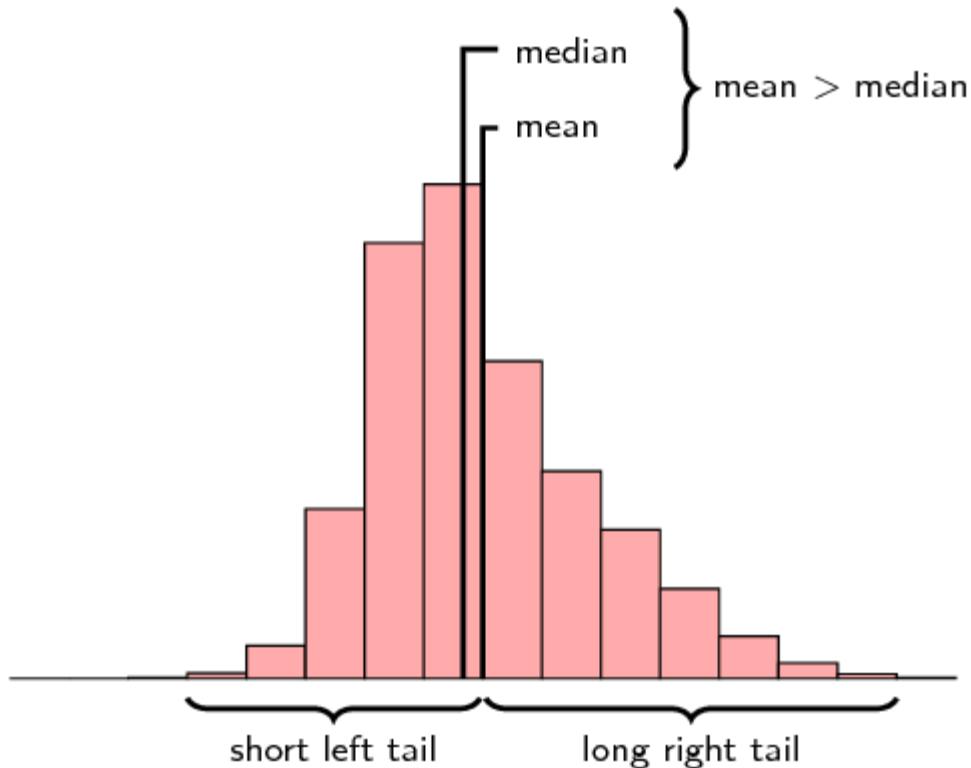


Another property of a symmetric distribution is that its median (second quartile) lies in the middle of its first and third quartiles. Note that

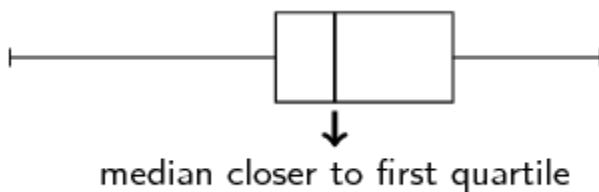
the whiskers of the plot (the minimum and maximum) do not have to be equally far away from the median. In the next section on outliers, you will see that the minimum and maximum values do not necessarily match the rest of the data distribution well.

### Skewed (EMBKG)

A distribution that is **skewed right** (also known as **positively skewed**) is shown below.



Now the picture is not symmetric around the mean anymore. For a right skewed distribution, the mean is typically greater than the median. Also notice that the tail of the distribution on the right hand (positive) side is longer than on the left hand side.

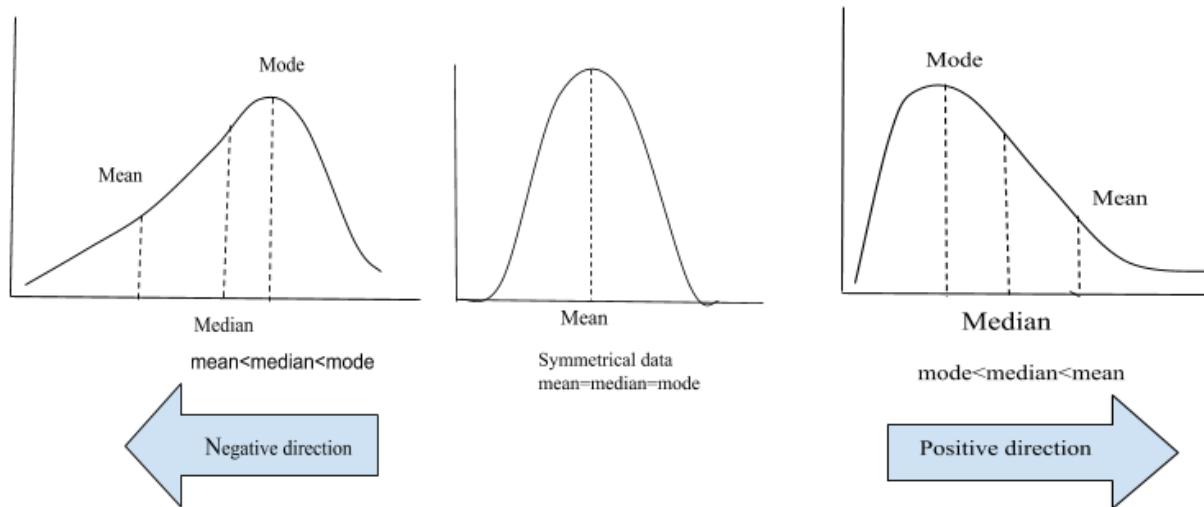


From the box and whisker diagram we can also see that the median is closer to the first quartile than the third quartile. The fact that the right hand side tail of the distribution is longer than the left can also be seen.

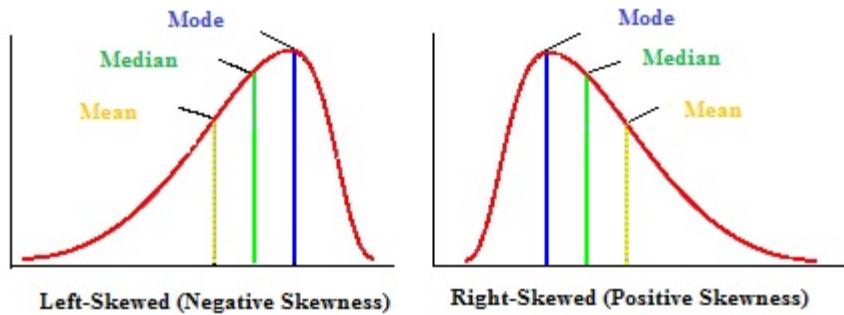
A distribution that is skewed left has exactly the opposite characteristics of one that is skewed right:

- the mean is typically less than the median;
- the tail of the distribution is longer on the left hand side than on the right hand side; and
- the median is closer to the third quartile than to the first quartile.

## Skewness



- The mean, median and mode are all Centrality measures (center of a set of data).
- The skewness of the data can be determined by how these quantities are related to one another
- By studying the shape of the data we can discover the relation between the mean, median and mode
- If the  $\text{mean} > \text{median}$  it indicates that the distribution is positively skewed.  
If the  $\text{mean} < \text{median}$  it indicates that the distribution is negatively skewed



- Karl Pearson's first method uses mode and it's formula
- $S_K = \frac{\text{mean} - \text{mode}}{\text{standard deviation}}$
- However, this method is not considered very stable in data's where mode is made up of too few pieces as it won't be considered a very strong measure of central tendency.
- For example in the first data set below 8 only occurs twice, so while using Pearson's first formula of skewness you have to be cautioned as it won't be a good measure of central tendency.
- Set 1 = [1, 2, 3, 4, 5, 8, 7, 8]
- However in the second set you can see that 8 appears ten times thus, you can use the Pearson's measure of skewness as you know it will give you a more stable and reliable result.
- Set 2 = [1, 5, 6, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8]
- The second formula of skewness uses the median and is denoted (By Karl Pearson)

$$S_K = \frac{3(\text{mean} - \text{median})}{\text{standard deviation}}$$

Karl Pearson's coefficient of skewness lies between -3 and +3.

- If  $SK = 0$  then we can say that the frequency distribution is normal and symmetrical.
- If  $SK < 0$  then we can say that the frequency distribution is negatively skewed.
- If  $SK > 0$  then we can say that the frequency distribution is positively skewed.

## **Example**

Q. The age (in years) of 6 randomly selected students from a class are:

[22, 25, 24, 23, 24, 20]

Find the Karl Pearson's coefficient of skewness.

*Solution:*

**STEP#1**

We will first find the mean.

*REMEMBER: For mean we first add all the data together and then divide it by the total number of numbers.*

$$\bar{X} = \frac{138}{6} = 23 \text{ years}$$

**STEP#2**

We will now find the median.

*REMEMBER: For median we pick the middle value of the set  $M = \frac{x_1 + x_2}{2}$*

$$M = \frac{24 + 23}{2} = 23.5 \text{ years}$$

**STEP#3**

Find the variance first and then take its unroot for Standard deviation.

$$= \frac{1}{5}(3190 - \frac{138^2}{6})$$

$$= \frac{1}{5}(3190 - \frac{19044}{6})$$

$$= \frac{1}{5}(3190 - 3174)$$

$$= \frac{16}{5}$$

$$s_x = \sqrt{s_x^2}$$

$$= \sqrt{3.2}$$

$$= 1.7889 \text{ years}$$

**STEP#4**

Put it all into Pearson's equation to get:

As you can tell the value of SK<0 thus we can say that the data is negatively skewed.

## **What Is the Interquartile Range Rule?**

### **What Is the Interquartile Range?**

Any set of data can be described by its five-number summary. These five numbers, which give you the information you need to find patterns and outliers, consist of (in ascending order):

- The minimum or lowest value of the dataset
- The first quartile  $Q_1$ , which represents a quarter(1/4<sup>th</sup> Position) of the way through the list of all data
- The median of the data set, which represents the midpoint of the whole list of data
- The third quartile  $Q_3$ , which represents three-quarters of the way through the list of all data (3/4<sup>th</sup> Position)
- The maximum or highest value of the data set.

These five numbers tell a person more about their data than looking at the numbers all at once could, or at least make this much easier.

- For example, the range, which is the minimum subtracted from the maximum, is one indicator of how spread out the data is in a set
  - note: the range is highly sensitive to outliers—if an outlier is also a minimum or maximum, the range will not be an accurate representation of the breadth of a data set
  - Range would be difficult to extrapolate otherwise.

## **IQR**

Similar to the range but less sensitive to outliers is the interquartile range.

- The interquartile range is **calculated in much the same way as the range**.
  - All you do to find it is **subtract the first quartile from the third quartile**:

$$\text{IQR} = Q_3 - Q_1.$$

- The interquartile range shows how the data is spread about the median.
- It is less susceptible than the range to outliers and can, therefore, be more helpful.

## **Using the Interquartile Rule to Find Outliers**

Though it's not often affected much by them, the interquartile range can be used to detect outliers. This is done using these steps:

1. Calculate the interquartile range for the data.
2. Multiply the interquartile range (IQR) by 1.5 (a constant used to discern outliers).
3. Add 1.5 x (IQR) to the third quartile. Any number greater than this is a suspected outlier.
4. Subtract 1.5 x (IQR) from the first quartile. Any number less than this is a suspected outlier.

### Note :

Remember that the interquartile rule is only a rule of thumb that generally holds but does not apply to every case. In general, you should always follow up your outlier analysis by studying the resulting outliers to see if they make sense. Any potential outlier obtained by the interquartile method should be examined in the context of the entire set of data.

### IQR Example

Given Set of data: 1, 3, 4, 6, 7, 7, 8, 8, 10, 12, 17.

The five-number summary for this data set is

- minimum = 1,
- first quartile = 4,
- median = 7,
- third quartile = 10 and
- maximum = 17.

\*\*\* You may look at the data and automatically say that 17 is an outlier, but what does the interquartile range rule say?

### IQR for the given Data :

- $Q_3 - Q_1 = 10 - 4 = 6$

Now

- multiply your answer by 1.5 to get  $1.5 \times 6 = 9$ .
- $(1^{\text{st}} \text{ QR} - 9) = 4 - 9 = -5$ . No data is less than this.
- $(3^{\text{rd}} \text{ QR} + 9) = 10 + 9 = 19$ . No data is greater than this.
- Despite the maximum value being five more than the nearest data point, the interquartile range rule shows that it should probably not be considered an outlier for this data set.

## Missing Values and replacement policies

- Most of the effort with the project connected with data is spent on data preparation, sometimes it can take up to 90 percent of the overall time spent on the project.
- Dealing with missing data is one of the most difficult parts in the data preparation phase.
- One of the reasons that it is considered difficult is that there is no best way to deal with missing values.
- In order to understand what to do with missing values found in your dataset, firstly, you need to understand what type of missing values you have. When I first faced the problem of missing data, it was difficult to understand the meaning of their types, that's what I will try to explain in this article with simple and clear examples.

Three kinds of missing data:

- Missing at Random (MAR)
- Missing Completely at Random (MCAR)
- Missing Not at Random (MNAR)

Let's imagine that we are trying to predict the price of the car that is being sold, for example in eBay, the data may look like this:

Model	Year	Color	Mileage	Price
Chevrolet	2014	NaN	10000	50000
Ford	2001	White	NaN	20000
Toyota	2005	Red	NaN	30000
Crysler	2019	Black	0	100000

Let us understand the different types of missing values with Example:

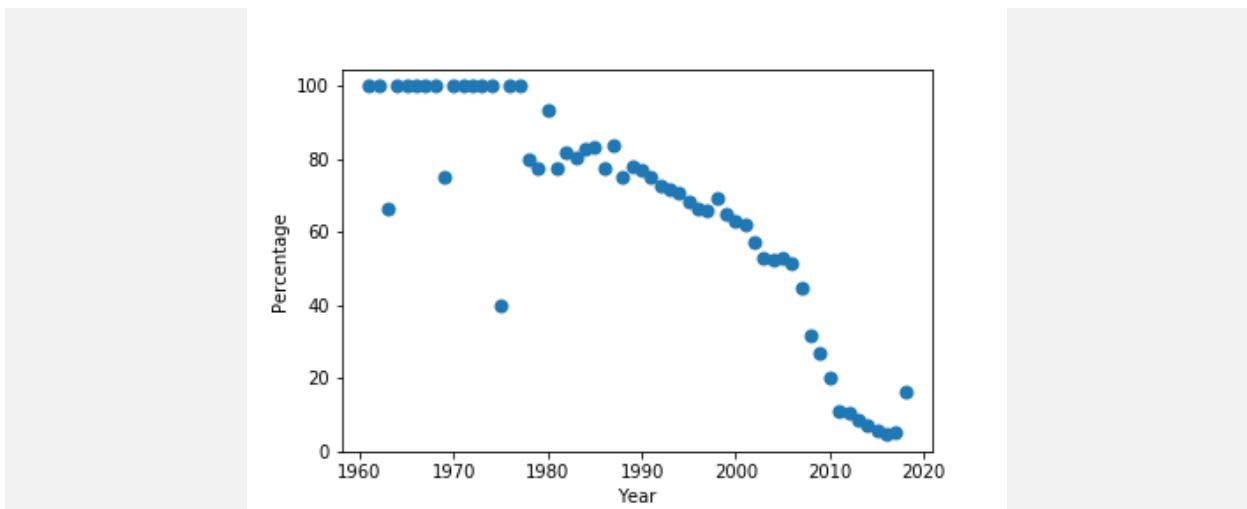
## Missing at Random (MAR)

MAR data — means there is a systematic relationship between the propensity of missing values and the observed data, but not the missing data.

What it means, is that the missingness of data can be predicted by other features in the dataset.

In the table above table

- the mileage has few missing values.
- the manufacturing year of cars with missing values is lower than in other examples
- 



**Graph for Percentage of missing values in the mileage column depending on the year of the car**

- The correlation between the percentage of missing values in the mileage column and the manufacturing year of the car is clearly seen
- It is clearly seen that older the car, more the probability that the mileage will not be provided by the seller of the car.
- we can predict the missingness of the mileage of the car, from its manufacturing year (may be with help of some expert)

## **Missing Completely at Random (MCAR)**

- MCAR means there is no relationship between the missingness of the data and any values, observed or missing.
- This kind of missing values is the easiest to understand.
- The fact that the data is missing has nothing to do neither with observed data nor with non-observed data, it's just missing.
- There is no logic in it.
- In the above given table there is missing value in the color column... random and non systematic one. (May be Someone just forgot to mention the color)

### **Missing Not at Random (MNAR)**

MNAR data is the most complicated one both in terms of finding it and dealing with it. The fact that the data is missing is related to the unobserved data, i.e. the data that we don't have, the missingness is related to factors that we didn't account for.

## Another Example

**MAR :**

- The missing data here is affected only by the complete (observed) variables and not by the characteristics of the missing data itself.
- in other words, for a data point, to be missing is not related to the missing data, but it is related to some of (or all) the observed data, the following example will depict the situation and make it more clear :

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	
26	121	26	
29	91	29	
30	105	30	
30	110	30	
31	98	31	
44	118	44	118
46	93	46	93
48	141	48	141
51	104	51	104
51	116	51	116
54	97	54	97

- We could easily notice that IQ score is missing for youngsters (age < 44 yo), and thus the missing data depends on the observed data, however there is no dependency with the values of the missing column itself.

## Missing Completely at Random (MCAR)

- There's no relationship between whether a data point is missing and any values in the data set (missing or observed).
- The missing data are just a random subset of the data.
- The missingness is nothing to do with any other variable. By the way, data are rarely MCAR.

- following example will depicts this kind of problem :

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	
26	121	26	121
29	91	29	91
30	105	30	
30	110	30	110
31	98	31	
44	118	44	118
46	93	46	93
48	141	48	
51	104	51	
51	116	51	116
54	97	54	

- It is relatively easy to check the assumption that in our example data is missing completely at random. If you can predict any reason for missing data (e.g., using common sense, regression, or some other method) whether based on the complete variable Age or the Missing variable IQ score , then the data is not MCAR !

### Missing Not at Random (MNAR)

- The data will be missing based on the missing column itself , for instance the following example points out the fact that data are missing on IQ score with only the people having a low score .

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	133
26	121	26	121
29	91	29	
30	105	30	
30	110	30	110
31	98	31	
44	118	44	118
46	93	46	
48	141	48	141
51	104	51	
51	116	51	116
54	97	54	

As you can see , it is impossible to detect MNAR cases without knowing the missing values !

## Coping And Dealing With Missing Data Problems (Missing Value Management)

### **Method 1: Deletion**

it falls under two different techniques :

- **Listwise Deletion** : In this method, an entire record is excluded from analysis if any single value is missing , and therefore we have the same N (number of records) for all analysis .
- **Pairwise Deletion** : during our analysis the number of records taken into consideration denoted “N” will vary according to the studied variable (column) , and for instance we could compute the mean for 2 features (Complete VS missing) and while dividing by the number of samples , we end up dividing by different N , one is the total number of rows and the other is the total number on complete values on the missing feature .

### **Example:**

```
X =  
  {. 3 2,  
   8 . 2,  
   1 5 8,  
   1 3 5,  
   2 4 3,  
   4 5 3 };
```

*Listwise deletion* is the operation used by regression procedures to deal with missing values.

During listwise deletion, an observation that contains a missing value in *any* variable is discarded;

no portion of that observation is used when building "cross product" matrices such as the covariance or correlation matrix.

For our example, listwise deletion means that the correlation matrix is formed by using rows 3–6, as follows:

```
/* Listwise deletion matrix:  
   1 5 8,  
   1 3 5,  
   2 4 3,  
   4 5 3 */
```

ListCorr		
1.000	0.492	-0.698
0.492	1.000	0.184
-0.698	0.184	1.000

What happens if you form the matrix that consists of pairwise correlations? That is, form the array C such that  $C[i,j]$  is the correlation between the  $i$ th and the  $j$ th columns of X. The missing values for each pair of variables are deleted based on whether either variable contains a missing value.

Under this pairwise-deletion scheme, each element of C is computed by using different observations:

- The element  $C[1,2]$  is computed by using observations 3–6 because the first observation has a missing value for X1 and the second observation has a missing value for X2.
- The element  $C[1,3]$  is computed by using observations 2–6 because the first observation is missing for X1.
- The element  $C[2,3]$  is computed by using observations 1 and 3–6 because the second observation is missing for X2.

The following SAS/IML statements compute the array of pairwise correlations:

```
/* vectors used for pairwise correlation
R12  R13  R23
      3 2
      8 2
 1 5  1 8  5 8
 1 3  1 5  3 5
 2 4  2 3  4 3
 4 5  4 3  5 3 */
PairCorr = corr(X, "Pearson", "pairwise");
Eigenval = eigval(PairCorr);
print PairCorr[format=6.3], Eigenval;
```

PairCorr		
1.000	0.492	-0.717
0.492	1.000	0.419
-0.717	0.419	1.000

For the matrix of pairwise correlations, one eigenvalue is negative. This indicates that the matrix is not a valid correlation matrix. *There is no multivariate distribution for which this matrix represents the correlation between variables!*

### **Method 2: Single Imputation Methods**

- **Single value imputation :** replacing the missing value with a single value utilizing one strategy such as : Mean , Median , Most Frequent , Mean Person , ... of the corresponding feature .

If there is a dataset that have great outliers, I'll prefer median. E.x.: 99% of household income is below 100, and 1% is above 500.

On the other hand, if we work with wear of clothes that customers give to dry-cleaner (assuming that dry-cleaners' operators fill this field intuitively), I'll fill missings with mean value of wear.

**Mean/Median/Mode Imputation:** For all observations that are non-missing, calculate the mean, median or mode of the observed values for that variable, and fill in the missing values with it. Context & spread of data are necessary pieces of information to determine which descriptor to use.

- Ok to use if missing data is less than 3%, otherwise introduces too much bias and artificially lowers variability of data

	col1	col2	col3	col4	col5		col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN	mean()	2.0	5.0	3.0	6.0	7.0
1	9	NaN	9.0	0	7.0		9.0	11.0	9.0	0.0	7.0
2	19	17.0	NaN	9	NaN		19.0	17.0	6.0	9.0	7.0

### **3. Hot or Cold Deck Imputation**

**“Hot Deck Imputation” :** Find all the sample subjects who are similar on other variables, then *randomly* choose one of their values to fill in.

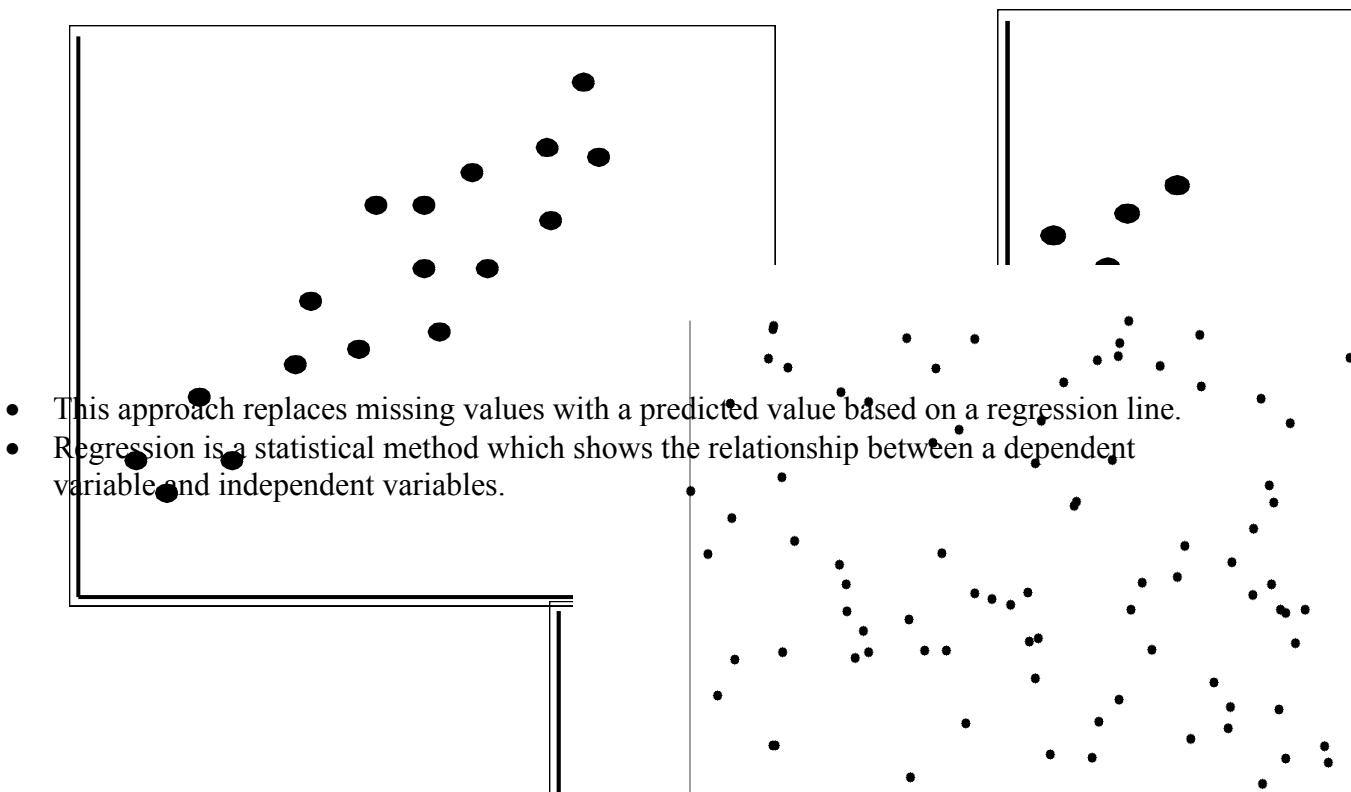
- Good because constrained by pre-existing values, but the randomness introduces hidden variability and is computationally expensive

**“Cold Deck Imputation” :** Systematically choose the value from an individual who has similar values on other variables (e.g. the third item of each collection). This option removes randomness of hot deck imputation.

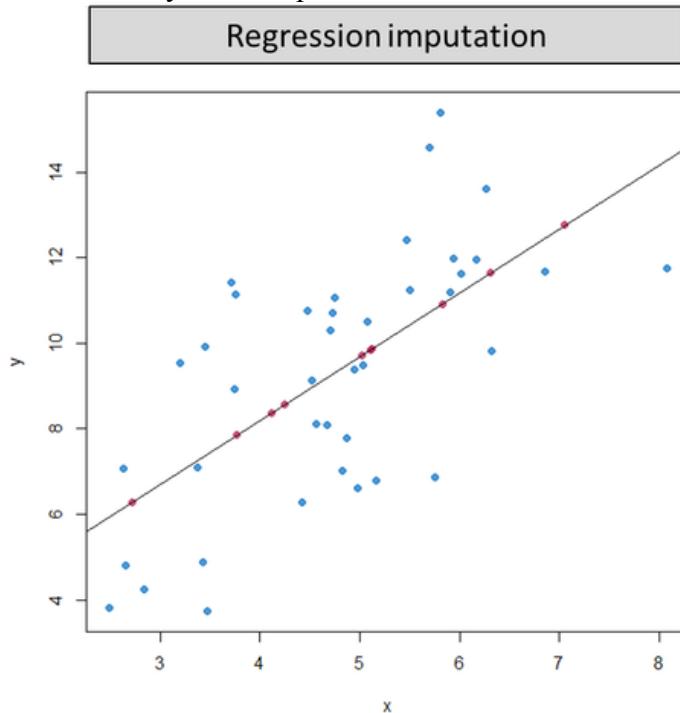
- Positively constrained by pre-existing values, but the randomness introduces hidden variability and is computationally expensive
- **Similarity** : trying to find the closest ( top-N closer ) row(s) to the row containing our missing value , and fix a strategy among them to assign a value to our missing value .

#### 4. Regression Imputation :

- In single regression imputation the imputed value is predicted from a regression equation , we assume that the missing values are in a regression line with a nonzero slope with one of the complete features ( predictors )
- Fill in with the predicted value obtained by regressing the missing variable on other variables; instead of just taking the mean, you're taking the *predicted value*, based on other variables.



- It's expressed as  $y = mx + b$  where m is the slope, b is a constant, x is the independent variable and y is the dependent variable.



- This method assumes that the imputed values fall directly on a regression line with a non-zero slope. As you can see, it is easy to comprehend and seems logical at the same time but it can affect the variability and the distribution of the data to some extent.

**“Stochastic regression imputation” :** The predicted value from a regression, *plus* a random residual value.

- This has all the advantages of regression imputation but adds in the advantages of the random component.

### Challenges of Single value mutation:

- values found in single imputation might be biased by the specific values in the current data set, and
- not represent the total values of the full population.

**Sol : Multiple Mutation**

## Multiple Imputation

Multiple imputation was a huge breakthrough in statistics about 20 years ago because it solved a lot of these problems with missing data (though, unfortunately not all). If done well, it leads to unbiased parameter estimates and accurate standard errors.

While single imputation gives us a *single* value for the missing observation's variable, multiple imputation gives us (you guessed it) *multiple* values for the missing observation's variable and then averages them for the final value.

To get each of these averages, a multiple imputation method would run analyses with 5–10 unique samples of the dataset and run the same predictive analysis on each\*\*. The predicted value at that point would serve as the value for that run; the data signature of these samples change each time, which causes the prediction to be a bit different. The more times you do this, the less biased the outcome will be.

Once you take the mean of these values, it is important to analyze their spread. If they're clustering, they have a low standard deviation. If they're not, variability is high and may be a sign that the value prediction may be less reliable.

While this method is much more unbiased, it is also more complicated and requires more computational time and energy.

# Numeric Data Imputation

## Mean/Median

Imputing with mean/median is one of the most intuitive methods, and in some situations, it may also be the most effective. We basically take the average of the data, or we take the median of the data and replace all missing values with that value.

- Assumptions: Data is missing at random; Missing observations look like the majority of observations
- Advantages: Easy and quick to implement; Preserves the loss of data
- Disadvantages: Co-variance and variance may change; More the missing data, higher the distortion

## Arbitrary Value Method

Here, the purpose is to flag missing values in the data set. You would impute the missing data with a fixed arbitrary value (a random value).

It is mostly used for categorical variables, but can also be used for numeric variables with arbitrary values such as 0, 999 or other similar combinations of numbers.

- Assumptions: Data is not missing at random
- Advantages: Quick and easy to implement; bring out underlying importance of missing values
- Disadvantages: Changes co-variance/variance; may create outliers

## End of Tail Method

This method is similar to the arbitrary value method, however, the arbitrary value here is chosen at the tail-end of the underlying distribution of the variable.

If normally distributed, we use the mean  $+/- 3$  times Standard Deviation.

**If the distribution is skewed, use the IQR proximity rule. [Refer the Document of Data Distribution]**

- Assumptions: Data is not missing at random; Data is skewed at the tail-end
- Advantages: Can bring out the importance of missing values;
- Disadvantages: Changes Co-variance/variance; may create biased data.

## Categorical Data Imputation

### Mode

As the name suggests, you impute missing data with the most frequently occurring value. This method would be best suited for categorical data, as missing values have the highest probability of being the most frequently occurring value.

- Assumptions: Data is missing at random; missing values look like majority
- Advantages: Quick and easy to implement; Suitable for categorical data
- Disadvantages: May create a biased data-set, favoring most frequent value

### Add a Category for Missing Data

This next method is quite straightforward and only works for categorical data. You would create a separate label for missing values – ‘missing’ or it could be anything relevant. The idea is to flag missing values and understand the importance of being missing.

- Assumptions: No assumption
- Advantages: Quick and easy to implement; Helps understand importance of missing data
- Disadvantage: Potentially misunderstood data; Number of missing data should be large enough



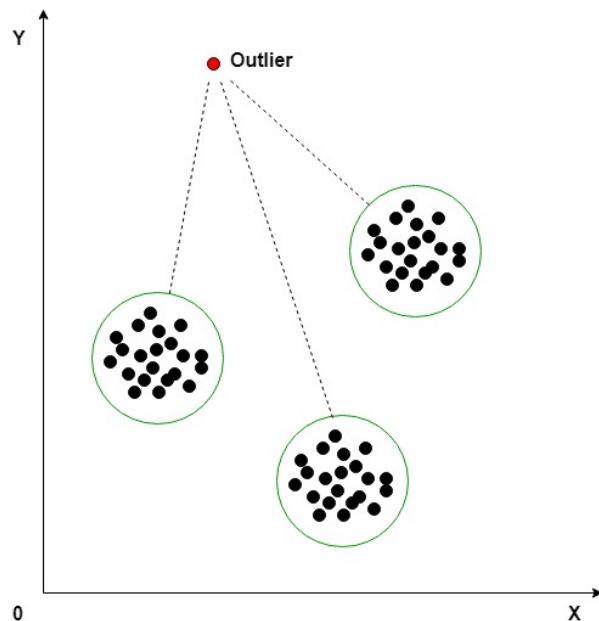
## A Brief Overview of Outlier Detection Techniques

*“Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism” — Hawkins(1980)*

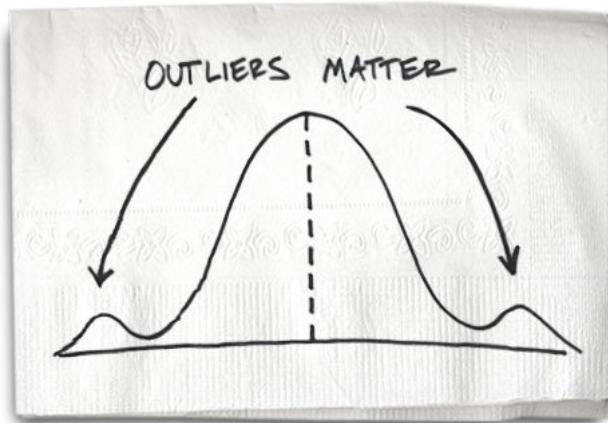
An **outlier** is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining.

### Why outlier analysis?

Most data mining methods discard outliers noise or exceptions, however, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring one and hence, the outlier analysis becomes important in such case.



## Another Way you can explain Outlier:



- Outliers are extreme values that deviate from other observations on data ,
- They may indicate a variability in a
  - measurement,
  - experimental errors or
  - a novelty
    - In the process of producing, collecting, processing and analyzing data, outliers can come from many sources and hide in many dimensions. Those that are not a product of an error are called **novelties..**
- In other words, an outlier is an observation that diverges from an overall pattern on a sample.

### **Most common causes of outliers on a data set:**

- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Experimental errors (data extraction or experiment planning/executing errors)
- Intentional (dummy outliers made to test detection methods)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources)
- Natural (not an error, novelties in data)

### **Outlier detection is important in many applications, such as:**

- Intrusions in communication networks
- Fraud in financial data
- Fake news and misinformation
- Healthcare analysis
- Industry damage detection
- Security and surveillance
- etc

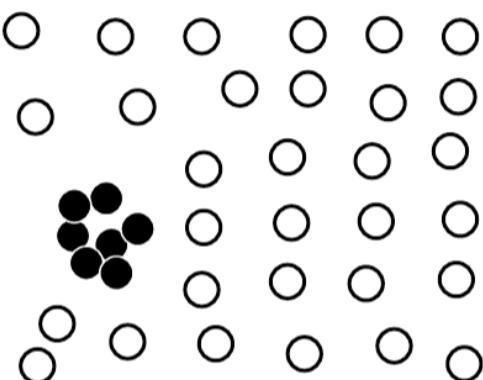
## Categorization of outliers

### 1. Univariate and Multivariate

- Univariate outliers can be found when looking at a distribution of values in a single feature space.
- Multivariate outliers can be found in a n-dimensional space (of n-features). Looking at distributions in n-dimensional spaces can be very difficult for the human brain, that is why we need to train a model to do it for us.

### 2. Depending on the environment:

- Global outlier (Point outlier)— Object significantly deviates from the rest of the data set
- **Contextual outliers:** Object deviates significantly based on a selected context.
  - noise in data, such as punctuation symbols when realizing text analysis
  - background noise signal when doing speech recognition
  - For example, 28 °C is an outlier for a Moscow winter, but not an outlier in another context, 28 °C is not an outlier for a Moscow summer
- **Collective outliers:** Usually, a data set may contain different types of outliers and at the same time may belong to more than one type of outlier.
  - A subset of data objects collectively deviate significantly from the whole data set, even if the individual data objects may not be outliers. For example, a large set of transactions of the same stock among a small party in a short period can be considered as an evidence of market manipulation



- Which and how many features am I taking into account to detect outliers ?  
**(univariate / multivariate)**
- Can I assume a distribution(s) of values for my selected features? **(parametric / non-parametric)**

**Will lead to different types of Detection methods**

# Outlier Detection Methods

- The outlier detection methods differ according to
  - Statistical study regarding normal objects versus outliers
  - Proximity Based
    - Density based
    - Distance based
  - sample of data for analysis is given with domain expert-provided labels that can be used to build an outlier detection model.
  - If expert-labeled examples of normal and/or outlier objects can be obtained, they can be used to build outlier detection models.
  - The methods used can be divided into supervised methods, semi-supervised methods, and unsupervised methods.

## Statistical Methods

### Parametric and Non parametric

Parametric methods involve assumption of some underlying distribution such as normal distribution whereas there is no such requirement with non-parametric approach.

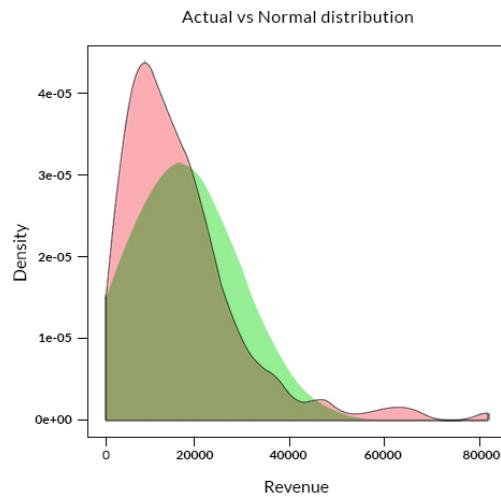
Additionally, you could do a univariate analysis by studying a single variable at a time or multivariate analysis where you would study more than one variable at the same time to identify outliers.

Assume we have the data for Revenue and Operating System for Mobile devices for an app.  
Below is the subset of the data:

OS	Revenue	Device
Android	8473	Mobile
Android	11790	Mobile
Android	7605	Mobile
iOS	15904	Mobile
iOS	19390	Mobile
iOS	56719	Mobile

## Problem : How can we identify outliers in the Revenue?

### Parametric Approach

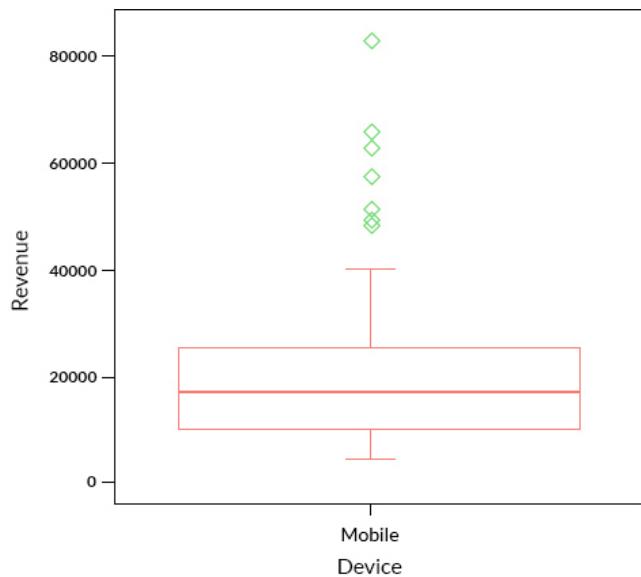


#### In above Graph :

- The x-axis, in the above plot, represents the Revenues and the y-axis, probability density of the observed Revenue value.
- The density curve for the actual data is shaded in ‘pink’,
- the normal distribution is shaded in ‘green’ and
- log normal distribution is shaded in ‘blue’.
- Note :
  - The probability density for the actual distribution is calculated from the observed data,
  - whereas for normal distribution is computed based on the observed mean and standard deviation of the Revenues.
- Outliers could be identified by calculating the probability of the occurrence of an observation or calculating how far the observation is from the mean.
  - For example, observations greater than 3 times the standard deviation from the mean, in case of normal distribution, could be classified as outliers.

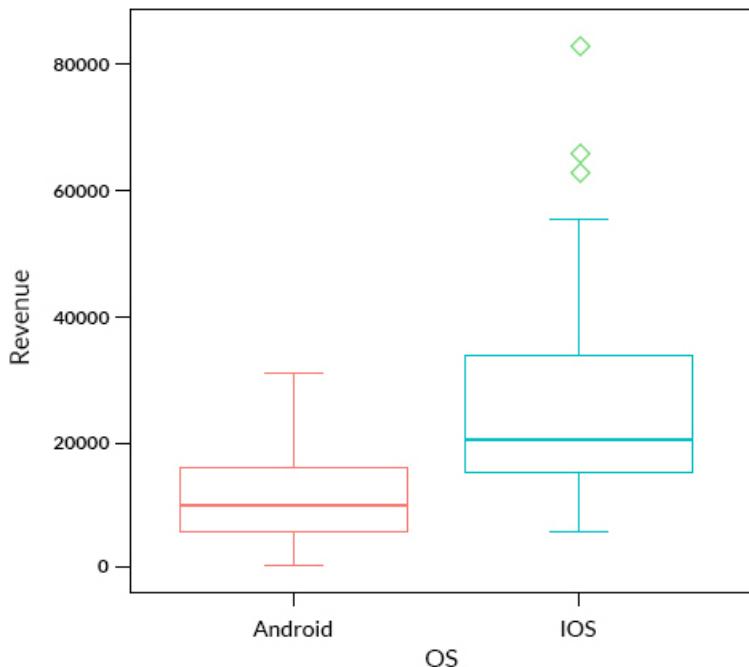
### Non-Parametric Approach

Let's look at a simple non-parametric approach like a box plot to identify the outliers.



In the box plot shown above, we can identify 7 observations, which could be classified as potential outliers, marked in green. These observations are beyond the whiskers.

In the data, we have also been provided information on the OS. Would we identify the same outliers, if we plot the Revenue based on OS??



In the above box plot, we are doing a bivariate analysis, taking 2 variables at a time which is a special case of multivariate analysis.

It seems that there are 3 outlier candidates for iOS whereas there are none for Android. This was due to the difference in distribution of Revenues for Android and iOS users.

So, just analyzing Revenue variable on its own i.e univariate analysis, we were able to identify 7 outlier candidates which dropped to 3 candidates when a bivariate analysis was performed.

### **Closing Thoughts**

- Both Parametric as well as Non-Parametric approach could be used to identify outliers based on the characteristics of the underlying distribution.
- If the mean accurately represents the center of the distribution and the data set is large enough, parametric approach could be used whereas if the median represents the center of the distribution, non-parametric approach to identify outliers is suitable.

Some of the most popular statistical methods for outlier detection are:

- **Z-Score or Extreme Value Analysis (parametric)**

- Probabilistic and Statistical Modeling (parametric)

- Linear Regression Models (PCA, LMS)

- **Proximity Based Models (non-parametric)**

- Information Theory Models

- High Dimensional Outlier Detection Methods (high dimensional sparse data)

## Z-Score

- The z-score or standard score of an observation is a metric that indicates how many standard deviations a data point is from the sample's mean, assuming a gaussian distribution (If not in normal distribution, proper transformation is to be applied).
- This makes z-score a parametric method.
- z-score (also called a standard score) gives you an idea of how far from the mean a data point is. But more technically it's a measure of how many standard deviations below or above the **population mean** a raw score is.
- A z-score can be placed on a normal distribution curve. Z-scores range from -3 standard deviations (which would fall to the far left of the normal distribution curve) up to +3 standard deviations (which would fall to the far right of the normal distribution curve). In order to use a z-score, you need to know the mean  $\mu$  (Population Mean) and standard deviation  $\sigma$ .
- After making the appropriate transformations to the selected feature space of the dataset, the z-score of any data point can be calculated with the following expression:

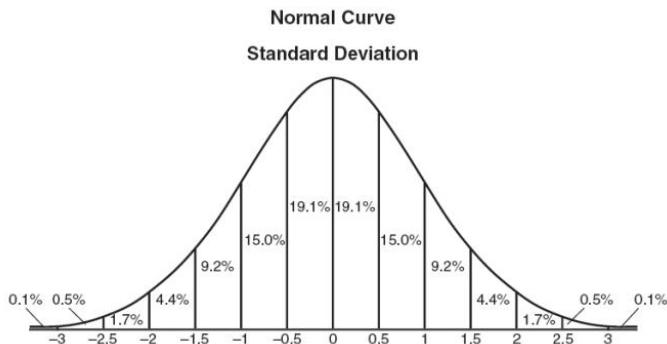
$$z = \frac{x - \mu}{\sigma}$$

- E.g. For the test score of 190. The test has a mean ( $\mu$ ) of 150 and a standard deviation ( $\sigma$ ) of 25. Assuming a normal distribution, your z score would be:

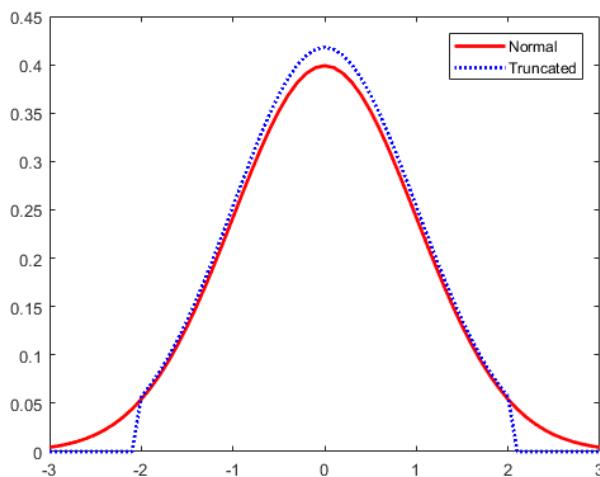
$$z = (x - \mu) / \sigma$$

$$= (190 - 150) / 25 = 1.6.$$

- The z score tells you how many standard deviations from the mean your score is. In this example, your score is 1.6 standard deviations *above* the mean.
- When computing the z-score for each sample on the data set a threshold must be specified. Some good ‘thumb-rule’ thresholds can be: 2.5, 3, 3.5 or more standard deviations.



By ‘tagging’ or removing the data points that lay beyond a given threshold we are classifying data into outliers and not outliers



Z-score is a simple, yet powerful method to get rid of outliers in data if you are dealing with parametric distributions in a low dimensional feature space.

## Population mean (weighted mean):

Population Mean Definition

- The population mean is an **average** of a group (population) characteristic. The group could be a person, item, or thing,
- In statistics, it's actually rare that you can calculate the population mean. That's because asking an entire **population** about something is usually cost prohibitive or too time consuming.
- For example, one veterinary practice probably keeps weight records of all the pets that come in the door, enabling you to calculate the average weight of a dog for that practice (i.e. the population mean for that practice).
- But if you were working for a pet food company who wanted to know the average weight of a dog, you wouldn't be able to track down all the 70 to 80 million dogs in the US and weigh them.
- You would have to take a **sample** (a small portion of the population of dogs) and weigh them. You can then use this figure to *approximate* the population mean.
- The population mean symbol is  $\mu$ .
- formula to find the population mean is:  
$$\mu = (\Sigma * X) / N$$

where:

$\Sigma$  means "the sum of."

X = all the individual items in the group.

N = the number of items in the group.
- **Sample question:** All 57 residents in a nursing home were surveyed to see how many times a day they eat meals.  
1 meal (2 people)  
2 meals (7 people)  
3 meals (28 people)  
4 meals (12 people)  
5 meals (8 people)  
What is the population mean for the number of meals eaten per day?

**Solution:**

**Step 1:** Sum up all of your X values. This is the  $\Sigma X$  portion of the population mean formula.

$$1 + 1 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 3 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5 = 188.$$

Note: You could also sum this with the following formula:

$$(1*2)+(2*7)+(3*28)+(4*12)+(5*8)=188.$$

- **Step 2:** Divide your answer to Step 1 with the number of items in your data set. There are 57 people, so:

$$188 / 57 = 3.29824561404$$

That's an average of 3.3 meals per person, per day.

The population mean is 3.3.

- **Population Mean vs. Sample Mean**

- Figuring out the population mean should feel familiar. You're just taking an average, using the same formula you probably learned in basic math (just with different notation). However, care must be taken to ensure that you are calculating the mean for a population (the whole group) and not a sample (part of the group). The symbols for the two are different:

Population mean symbol =  $\mu$

Sample mean symbol =  $\bar{x}$

## Nonparametric problems (Dbscan)

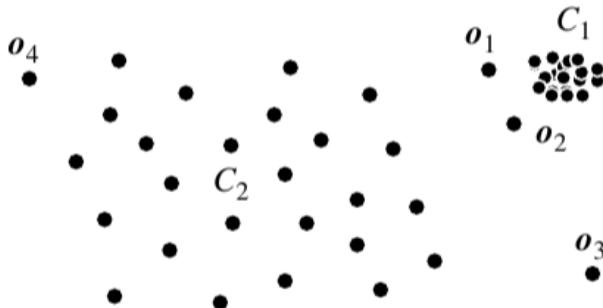
In nonparametric methods for outlier detection, the model of “normal data” is learned from the input data, rather than assuming one a priori. Nonparametric methods often make fewer assumptions about the data and thus can be applicable in more scenarios. As an example, we can use histograms.

Distance Based :

- ❖ Distance-based outlier detection method consults the neighbourhood of an object, which is defined by a given radius. An object is then considered an outlier if its neighborhood does not have enough other points.
- ❖ A distance the threshold that can be defined as a reasonable neighbourhood of the object.

Density Based :

- ❖ Density-based outlier detection method investigates the density of an object and that of its neighbors. Here, an object is identified as an outlier if its density is relatively much lower than that of its neighbors.



- ❖ Consider the example above, distance-based methods are able to detect  $O_3$ , but as for  $O_1$  and  $O_2$  it is not as evident.
- ❖ The idea of density-based is that we need to compare the density around an object with the density around its local neighbors. The basic assumption of density-based outlier detection methods is that the density around a nonoutlier object is similar to the density around its neighbors, while the density around an outlier object is significantly different from the density around its neighbors.

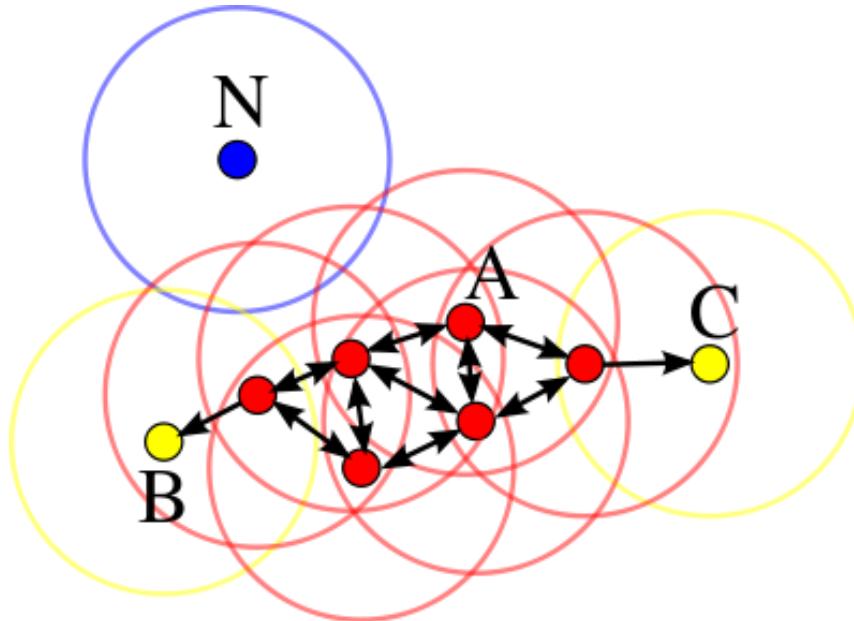
## Dbscan (Density Based Spatial Clustering of Applications with Noise)

In machine learning and data analytics clustering methods are useful tools that help us visualize and understand data better. Relationships between features, trends and populations in a data set can be graphically represented via clustering methods like dbscan, and can also be applied to detect outliers in nonparametric distributions in many dimensions.

Dbscan is a density based clustering algorithm, it is focused on finding neighbors by density (MinPts) on an ‘n-dimensional sphere’ with radius  $\epsilon$ . A cluster can be defined as the maximal set of ‘density connected points’ in the feature space.

Dbscan then defines different classes of points:

- **Core point:** A is a core point if its neighborhood (defined by  $\epsilon$ ) contains at least the same number or more points than the parameter MinPts.
- **Border point:** C is a border point that lies in a cluster and its neighborhood does not contain more points than MinPts, but it is still ‘*density reachable*’ by other points in the cluster.
- **Outlier:** N is an outlier point that lies in no cluster and it is not ‘*density reachable*’ nor ‘*density connected*’ to any other point. Thus this point will have “his own cluster”.



- If A is a core point, it forms a cluster with all the points that are reachable from it. A point Q is reachable from P if there is a path  $p_1, \dots, p_n$  with  $p_1 = p$  and  $p_n = q$ , where each  $p_{i+1}$  is directly reachable from  $p_i$  (all the points on the path must be core points, with the possible exception of q).
- Reachability is a non-symmetric relation since, by definition, no point may be reachable from a non-core point, regardless of distance (so a non-core point may be reachable, but nothing can be reached from it!). Therefore a further notion of *connectedness* is needed to formally define the extent of the clusters found by this algorithm.
- Two points  $p$  and  $q$  are density-connected if there is a point o such that both  $p$  and  $q$  are density-reachable from  $o$ . *Density-connectedness* is symmetric.

A cluster satisfies two properties:

- All points within the cluster are mutually density-connected.

- If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

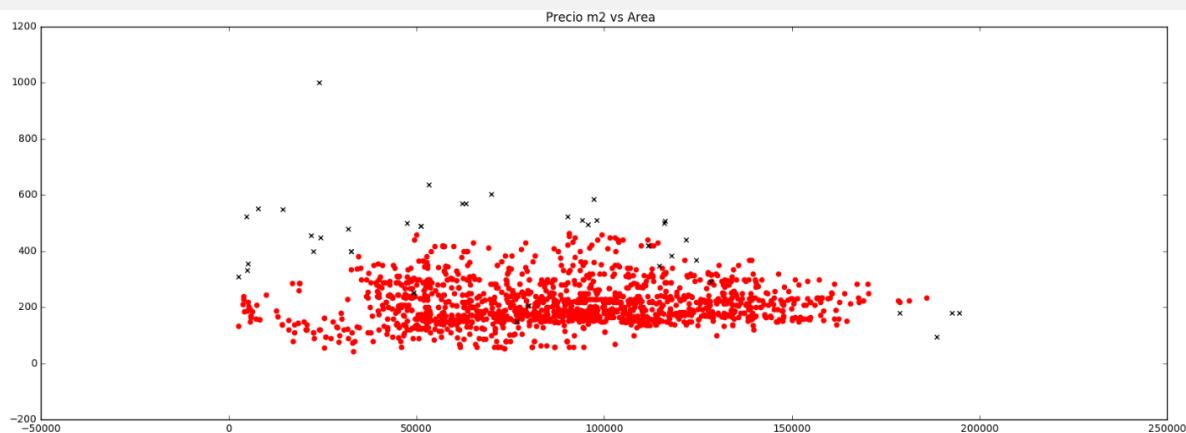
**Steps to be followed:**

- ❖ Before starting DBSCAN , important step is scaling the data, since the radius  $\epsilon$  will define the neighborhoods along with MinPts.
- ❖ After scaling the feature space, is time to choose the spatial metric on which dbscan will perform the clustering. The metric must be chosen depending on the problem, an euclidean metric works well for 2 or 3 dimensions, the manhattan metric can also be useful when dealing with higher dimensional feature spaces 4 or more dimensions.
- ❖ Then, the parameter eps ( $\epsilon$ ) must be chosen accordingly to perform clustering. If  $\epsilon$  is too big many points will be density connected, if its too small the clustering will result in many meaningless clusters. A good approach is to try values ranging from 0.25 to 0.75.

Dbscan is also sensitive to the MinPts parameter, tuning it will completely depend on the problem at hand.

The complexity of dbscan is of  $O(n \log n)$ , it is an effective method with medium sized data sets. Dbscan estimates the number of clusters by itself, there is no need to specify the number of desired clusters, it is an unsupervised machine learning model.

Outliers (noise) will be assigned to the -1 cluster. After tagging those instances, they can be removed or analyzed.



**Real world application of DBSCAN in housing prices (red:normal, black: outliers)**

### **Z-Score pros:**

- It is a very effective method if you can describe the values in the feature space with a gaussian distribution. (Parametric)
- The implementation is very easy using pandas and scipy.stats libraries.

### **Z-Score cons:**

- It is only convenient to use in a low dimensional feature space, in a small to medium sized dataset.
- Is not recommended when distributions can not be assumed to be parametric.

### **Dbscan pros:**

- It is a super effective method when the distribution of values in the feature space can not be assumed.
- Works well if the feature space for searching outliers is multidimensional (ie. 3 or more dimensions)
- Sci-kit learn's implementation is easy to use and the documentation is superb.
- Visualizing the results is easy and the method itself is very intuitive.

### **Dbscan cons:**

- The values in the feature space need to be scaled accordingly.
- Selecting the optimal parameters eps, MinPts and metric can be difficult since it is very sensitive to any of the three params.
- It is an unsupervised model and needs to be re-calibrated each time a new batch of data is analyzed.
  - It can predict once calibrated but is strongly not recommended.

## **Proximity based methods:**

### **Supervised method**

We model an outlier detection as a classification problem. Samples examined by domain experts used for training & testing.

#### **Challenges:**

- Classes are unbalanced. That is, the population of outliers is typically much smaller than that of normal objects. Methods for handling unbalanced classes can be used, **such as oversampling**.
- Catch as many outliers as possible, **i.e., recall is more important than accuracy** (i.e., not mislabeling normal objects as outliers)

### **Unsupervised method**

- In some application scenarios, objects labeled as “normal” or “outlier” are not available. Thus, an unsupervised learning method has to be used.
- Unsupervised outlier detection methods make an implicit assumption: The normal objects are somewhat “clustered.”
- In other words, an unsupervised outlier detection method expects that normal objects follow a pattern far more frequently than outliers.

#### **Challenges:**

- Normal objects may not share any strong patterns, but the collective outliers may share high similarity in a small area
- In case if normal activities are diverse and do not fall into high-quality clusters unsupervised methods may have a high false positive rate and may let many actual outliers be undetected.

The latest unsupervised methods developed smart ideas to tackle outliers directly without explicitly detecting clusters.

## **Semi-Supervised Methods**

In many applications, although obtaining some labeled examples is feasible, the number of such labeled examples is often small. If some labeled normal objects are available:

- Use the labeled examples and the proximate unlabeled objects to train a model for normal objects
- Those not fitting the model of normal objects are detected as outliers

## **Challenges**

- If only some labeled outliers are available, a small number of labeled outliers may not cover the possible outliers well.

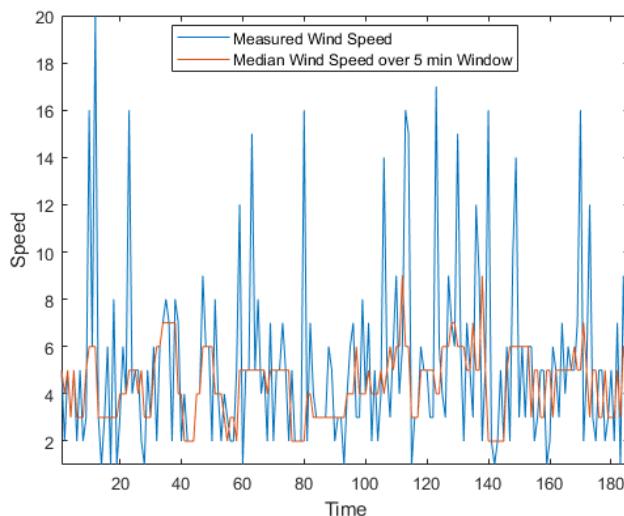


## Data Transformation in Data Mining

- It is done for combining unstructured data with structured data to analyze it later.
- It is also important when the data is transferred to a new cloud **data warehouse**.
  - As the homogeneous and well-structured data are easier to analyze and look for patterns.
  - For example, a company has acquired another firm and now has to consolidate all the business data. The smaller company may be using a different database than the parent firm. Also, the data in these databases may have unique IDs, keys and values. All this needs to be formatted so that all the records are similar and can be evaluated.
- This is why data transformation methods are applied. And, they are described below:

### Data Smoothing

- This method is used for removing the noise from a dataset.
- Noise is referred to as the distorted and meaningless data within a dataset.
- Smoothing uses algorithms to highlight the special features in the data.
- After removing noise, the process can detect any small changes to the data to detect special patterns.
- When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form.
  - Any data modification or trend can be identified by this method.



### Smoothing using binning

#### Unsorted data for price in dollars

Before sorting: 8 16, 9, 15, 21, 21, 24, 30, 26, 27, 30, 34

## **First of all, sort the data**

After Sorting: 8, 9, 15, 16, 21, 21, 24, 26, 27, 30, 30, 34

### **1. Smoothing the data by equal frequency bins**

**Bin 1:** 8, 9, 15, 16

**Bin 2:** 21, 21, 24, 26,

**Bin 3:** 27, 30, 30, 34

- **Smoothing by bin means**

**For Bin 1:**

$$(8 + 9 + 15 + 16 / 4) = 12$$

Bin 1 = 12, 12, 12, 12

**For Bin 2**

$$(21 + 21 + 24 + 26 / 4) = 23$$

Bin 2 = 23, 23, 23, 23

**For Bin 3:**

$$(27 + 30 + 30 + 34 / 4) = 30$$

Bin 3 = 30, 30, 30, 30

- **Smoothing by bin boundaries**

- pick the minimum and maximum value.
- Put the minimum on the left side and maximum on the right side.
- Middle values in bin boundaries move to its closest neighbor value with less distance.

**Before bin Boundary:** Bin 1: 8, 9, 15, 16

Here, 8 is the minimum value and 16 is the maximum value. 9 is near to 8, so 9 will be treated as 8. 15 is more near to 16 and farther away from 8. So, 15 will be treated as 16.

**After bin Boundary:** Bin 1: 8, 8, 16, 16

**Before bin Boundary:** Bin 2: 21, 21, 24, 26,

**After bin Boundary:** Bin 2: 21, 21, 26, 26,

**Before bin Boundary:** Bin 3: 27, 30, 30, 34

**After bin Boundary:** Bin 3: 27, 27, 27, 34

Pros of Smoothing :

- Data smoothing clears the understandability of different important hidden patterns in the data set.
- Data smoothing can be used to help predict trends. Prediction is very helpful for getting the right decisions at the right time.

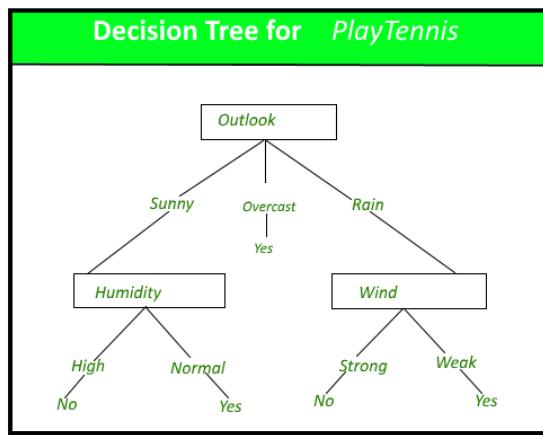
- Data smoothing helps in getting accurate results from the data.

Cons of data smoothing

- Data smoothing doesn't always provide a clear explanation of the patterns among the data.
- It is possible that certain data points being ignored by focusing the other data points.

## Discretization

- This is a process of converting continuous data into a set of data intervals.
- Continuous attribute values are substituted by small interval labels.
- This makes the data easier to study and analyze.
- If a continuous attribute is handled by a **data mining** task, then its discrete values can be replaced by constant quality attributes.
- This improves the efficiency of the task.
- This method is also called data reduction mechanism as it transforms a large dataset into a set of categorical data.
- Discretization also uses **decision tree-based algorithms** to produce short, compact and accurate results when using discrete values.



- It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes.
- Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values.
  - For **example**, (1-10, 11-20) (age:- young, middle age, senior).

## Generalization

- In this process, low-level data attributes are transformed into high-level data attributes using concept hierarchies.
- This conversion from a lower level to a higher conceptual level is useful to get a clearer picture of the data.
- For example, age data can be in the form of (20, 30) in a dataset. It is transformed into a higher conceptual level into a categorical value (young, old).
- Data generalization can be divided into two approaches – data cube process (OLAP) and attribute oriented induction approach (AOI).
- It converts low-level data attributes to high-level data attributes using concept hierarchy.

- For Example Age initially in Numerical form (22, 25) is converted into categorical value (young, old).
- For **example**, Categorical attributes, such as house addresses, may be generalized to higher-level definitions, such as town or country.

### **Attribute construction**

- In the attribute construction method, new attributes are created from an existing set of attributes.
- For example, in a dataset of employee information, the attributes can be employee name, employee ID and address. These attributes can be used to construct another dataset that contains information about the employees who have joined in the year 2019 only.
- This method of reconstruction makes mining more efficient and helps in creating new datasets quickly.
- Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

### **Normalization**

- Also called data pre-processing, this is one of the crucial techniques for **data transformation in data mining**.
- Here, the data is transformed so that it falls under a given range.
- When attributes are on different ranges or scales, data modelling and mining can be difficult. Normalization helps in applying **data mining algorithms** and extracting data faster.
- Data normalization involves converting all data variable into a given range. Techniques that are used for normalization are:

#### **The popular normalization methods are:**

1. Min-max normalization
2. Decimal scaling
3. Z-score normalization

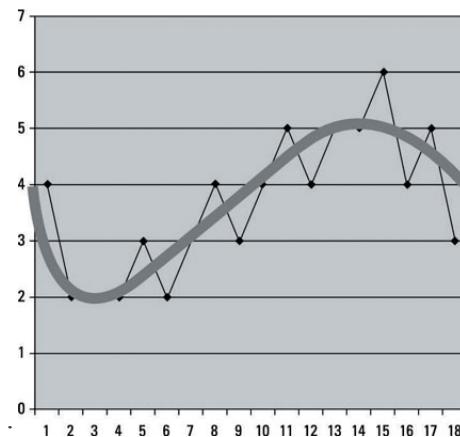
## Data Smoothing

To remove irregularity from the data

- Noise removal from normal data
- Data filtering
- Outlier removal etc.

In time series data (Continuous signal)

- Smoothing is usually done to help us better see patterns (trends in time series data)
- smooth out the irregular roughness to see a clearer signal
- Like other DM procedures, smoothing doesn't result into a model,, but it can be a good first step in describing various components of the series which will help to prepare the model
- The implication behind data smoothing is that the data consists of two parts:
  - part (consisting of the core data points) that signifies overall trends or real trends, and
  - part consisting mostly of deviations (noise) — some fluctuating points that result from some volatility in the data.
  - **Data smoothing seeks to eliminate that second part.**



- Data smoothing operates on several assumptions:
  - That fluctuation in data is likeliest to be noise.
  - That the noisy part of the data is of short duration.
  - That the data's fluctuation, regardless of how varied it may be, won't affect the underlying trends represented by the core data points.
- Noise in data tends to be random and its fluctuations should not affect the overall trends drawn from examining the rest of the data.
- Reducing or eliminating noisy data points can clarify real trends and patterns in the data and improv the data's "signal-to-noise ratio."
  - Smoothing will not accurately predict the exact value e.g. price of the next trade for a given stock, but predict the trend

- predicting a general trend can yield more powerful insights than knowing the actual price or its fluctuations.

### **Smoothing using binning (Already discussed in last class)**

#### **Unsorted data for price in dollars**

Before sorting: 8 16, 9, 15, 21, 21, 24, 30, 26, 27, 30, 34

#### **First of all, sort the data**

After Sorting: 8, 9, 15, 16, 21, 21, 24, 26, 27, 30, 30, 34

#### **1. Smoothing the data by equal frequency bins**

**Bin 1:** 8, 9, 15, 16

**Bin 2:** 21, 21, 24, 26,

**Bin 3:** 27, 30, 30, 34

- **Smoothing by bin means**

**For Bin 1:**

$$(8+9+15+16/4) = 12$$

Bin 1 = 12, 12, 12, 12

**For Bin 2**

$$(21+21+24+26/4) = 23$$

Bin 2 = 23, 23, 23, 23

**For Bin 3:**

$$(27+30+30+34/4) = 30$$

Bin 3 = 30, 30, 30, 30

- **Smoothing by bin boundaries**

- pick the minimum and maximum value.

- Put the minimum on the left side and maximum on the right side.

- Middle values in bin boundaries move to its closest neighbor value with less distance.

**Before bin Boundary:** Bin 1: 8, 9, 15, 16

Here, 8 is the minimum value and 16 is the maximum value. 9 is near to 8, so 9 will be treated as 8. 15 is more near to 16 and farther away from 8. So, 15 will be treated as 16.

**After bin Boundary:** Bin 1: 8, 8, 16, 16

**Before bin Boundary:** Bin 2: 21, 21, 24, 26,

**After bin Boundary:** Bin 2: 21, 21, 26, 26,

**Before bin Boundary:** Bin 3: 27, 30, 30, 34

**After bin Boundary:** Bin 3: 27, 27, 27, 34

## The Mean value and Standard Error

Case : A manager of a warehouse wants to know how much a typical supplier delivers in 1000 dollar units. He/she takes a sample of 12 suppliers, at random, obtaining the following results:

Supplier	Amount
1	9
2	8
3	9
4	12
5	9
6	12
7	11
8	7
9	13
10	9
11	11
12	10

- ❖ The computed mean or average of the Amount = 10.
- ❖ The manager decides to use this as the estimate for *expenditure of a typical supplier*.

### Is this a good or bad estimate?

#### To estimate the judgement we shall calculate the error in the judgement

- The "error" = true amount spent minus the estimated amount.
- The "error squared" is the error above, squared.
- The "SSE" is the sum of the squared errors.
- The "MSE" is the mean of the squared errors (Lower the value, better the estimate).

## The estimate = 10

Supplier	\$	Error	Error Squared
----------	----	-------	---------------

1	9	-1	1
2	8	-2	4
3	9	-1	1
4	12	2	4
5	9	-1	1
6	12	2	4
7	11	1	1
8	7	-3	9
9	13	3	9
10	9	-1	1
11	11	1	1
12	10	0	0

$$\text{SSE} = 36$$

$$\text{MSE} = 36/12 = 3$$

Same way We will calculate the MSE assuming the estimated value to be 7,9 and 12 instead of

Estimator	7	9	10	12
-----------	---	---	----	----

SSE	144	48	36	84
-----	-----	----	----	----

MSE	12	4	3	7
-----	----	---	---	---

Ideally the estimation of **mean** will give the best result

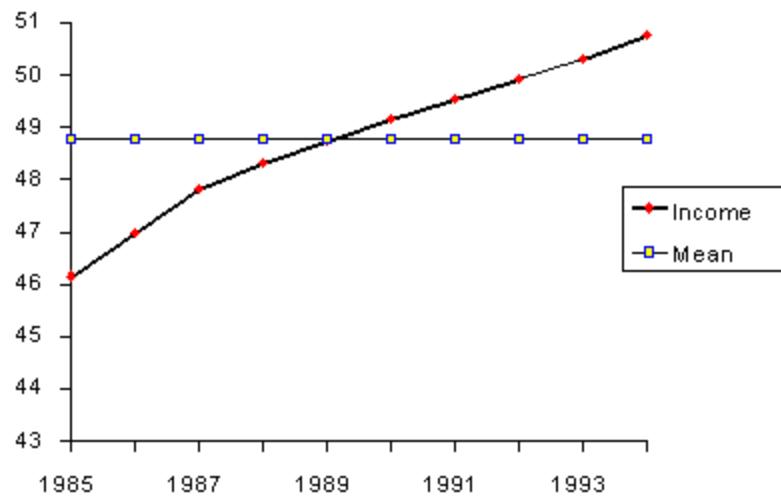
Lets take another example

Data : Income before taxes of a businessman between 1985 and 1994

Year	\$ (millions)	Mean	Error	Squared Error
1985	46.163	48.676	-2.513	6.313
1986	46.998	48.676	-1.678	2.814
1987	47.816	48.676	-0.860	0.739
1988	48.311	48.676	-0.365	0.133
1989	48.758	48.676	0.082	0.007
1990	49.164	48.676	0.488	0.239
1991	49.548	48.676	0.872	0.761
1992	48.915	48.676	0.239	0.057
1993	50.315	48.676	1.639	2.688
1994	50.768	48.676	2.092	4.378

**The MSE = 1.8129.**

Graph showing the Mean and Income line for the data



- *Can we use the mean to forecast income (Can we smooth the Income line with a mean value?) if the goal is to predict the trend?*
- The answer is Big NO

•

### Rather than equal, Different weights must be assigned to the recent past and long past data

Example: Average of the values 3, 4, 5 is 4

$$=3/3 + 4/3 + 5/3$$

$$=1 + 1.3333 + 1.6667 = 4.$$

The multiplier 1/3 is called the *weight*.

General Formula (sum of weights = 1)

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \left(\frac{1}{n}\right) x_1 + \left(\frac{1}{n}\right) x_2 + \dots + \left(\frac{1}{n}\right) x_n$$

# Single Moving Average

- Averaging Successive Subsets
- 
- Data : (Discussed above)
  - 9, 8, 9, 12, 9, 12, 11, 7, 13, 9, 11, 10
  - Let M= size of Subset = 3
  - For first 3 numbers avg =  $(9 + 8 + 9) / 3 = 8.667$ .
- This is "smoothing" is continued by advancing one period and calculating the next average of three numbers, dropping the first number known as Moving average

## Results of Moving Average

Supplier	\$	MA	Error	Error squared
----------	----	----	-------	---------------

1	9	9	0	0
2	8	8	0	0
3	9	8.667	0.333	0.111
4	12	9.667	2.333	5.444
5	9	10.000	-1.000	1.000
6	12	11.000	1.000	1.000
7	11	10.667	0.333	0.111
8	7	10.000	-3.000	9.000
9	13	10.333	2.667	7.111
10	9	9.667	-0.667	0.444
11	11	11.000	0	0
12	10	10.000	0	0

The MSE = 2.42 as compared to 3 in the previous case

### **Central Moving Average :**

Major change: : place the average in the middle of the time interval of three periods  
Note: This works well with odd time periods, but not so good for even time periods.  
Smooth with MA's using  $M = 4$

Interim Steps					
Period	Value	MA Centered	Final	Error	
1	9		9	0	
1.5					
2	8		8	0	
2.5		9.5			
3	9	9.5	9.5	0.5	
3.5		9.5			
4	12	10.0	10	2	
4.5		10.5			
5	9	10.750	10.750	1.75	
5.5		11.0			
6	12		12	0	
6.5					
7	11		11	0	

$$\text{MSE} = 1.04$$

## **Double Moving Averages for a Linear Trend Process**

Moving averages are still not able to handle significant trends when forecasting

- Unfortunately, neither the mean of all data nor the moving average of the most recent M values, when used as forecasts for the next period, are able to cope with a significant trend.
- Double Moving Averages for a Linear Trend Process :
- calculates a second moving average from the original moving average, using the same value for M.
- As soon as both single and double moving averages are available, a computer routine uses these averages to compute a slope and intercept, and then forecasts one or more periods ahead.



## Exponential Smoothing

This is a very popular scheme to produce a smoothed Time Series.

Whereas in Single Moving Averages the past observations are weighted equally,

Exponential Smoothing assigns *exponentially decreasing weights* as the observation get older.

In other words, *recent observations are given relatively more weight in forecasting than the older observations.*

$$S_2 = y_1$$

$$S_3 = \alpha y_2 + (1 - \alpha) S_2;$$

$$S_t = \alpha y_{t-1} + (1 - \alpha) S_{t-1} \quad 0 < \alpha \leq 1 \quad t \geq 3.$$

- This smoothing scheme begins by setting  $S_2$  to  $y_1$ ,
- $S_t$  stands for smoothed observation or EWMA (Exponentially weighted Moving Average),
- $y$  stands for the original observation.
- The subscripts refer to the time periods, 1, 2, ..., n. For the third period,

### Setting the first EWMA

The initial EWMA plays an important role in computing all the subsequent EWMAs.

Setting  $S_2$  to  $y_1$  is one method of initialization. Another way is to set it to the target of the process.

Still another possibility would be to average the first four or five observations.

It can also be shown that the smaller the value of  $\alpha$ , the more important is the selection of the initial EWMA. The user would be wise to try a few methods, (assuming that the software has them available) before finalizing the settings.

## Why is it called "Exponential"?

### Why is it called "Exponential"?

*Expand basic equation*

Let us expand the basic equation by first substituting for  $S_{t-1}$  in the basic equation to obtain

$$\begin{aligned} S_t &= \alpha y_{t-1} + (1 - \alpha) [\alpha y_{t-2} + (1 - \alpha) S_{t-2}] \\ &= \alpha y_{t-1} + \alpha(1 - \alpha) y_{t-2} + (1 - \alpha)^2 S_{t-2}. \end{aligned}$$

*Summation formula for basic equation*

By substituting for  $S_{t-2}$ , then for  $S_{t-3}$ , and so forth, until we reach  $S_2$  (which is just  $y_1$ ), it can be shown that the expanding equation can be written as:

$$S_t = \alpha \sum_{i=1}^{t-2} (1 - \alpha)^{i-1} y_{t-i} + (1 - \alpha)^{t-2} S_2, \quad t \geq 2.$$

*Expanded equation for  $S_5$*

For example, the expanded equation for the smoothed value  $S_5$  is:

$$S_5 = \alpha [(1 - \alpha)^0 y_{5-1} + (1 - \alpha)^1 y_{5-2} + (1 - \alpha)^2 y_{5-3}] + (1 - \alpha)^3 S_2.$$

*Illustrates exponential behavior*

This illustrates the exponential behavior. The weights,  $\alpha(1 - \alpha)^t$  decrease geometrically, and their sum is unity as shown below, using a property of geometric series:

$$\alpha \sum_{i=0}^{t-1} (1 - \alpha)^i = \alpha \left[ \frac{1 - (1 - \alpha)^t}{1 - (1 - \alpha)} \right] = 1 - (1 - \alpha)^t.$$

From the last formula we can see that the summation term shows that the contribution to the smoothed value  $S_t$  becomes less at each consecutive time period.

*Example for  $\alpha = 0.3$*

Let  $\alpha = 0.3$ . Observe that the weights  $\alpha(1 - \alpha)^t$  decrease exponentially (geometrically) with time.

### Value weight

last	$y_1$	0.2100
	$y_2$	0.1470
	$y_3$	0.1029
	$y_4$	0.0720

### What is the "best" value for $\alpha$ ?

*How do you choose the weight parameter?*

The speed at which the older responses are dampedened (smoothed) is a function of the value of  $\alpha$ . When  $\alpha$  is close to 1, dampening is quick and when  $\alpha$  is close to 0, dampening is slow. This is illustrated in the table below.

-----> towards past observations

$\alpha$	$(1 - \alpha)$	$(1 - \alpha)^2$	$(1 - \alpha)^3$	$(1 - \alpha)^4$
0.9	0.1	0.01	0.001	0.0001
0.5	0.5	0.25	0.125	0.0625
0.1	0.9	0.81	0.729	0.6561

We choose the best value for  $\alpha$  so the value which results in the smallest MSE.

Example : Consider the following data set consisting of 12 observations taken over time:

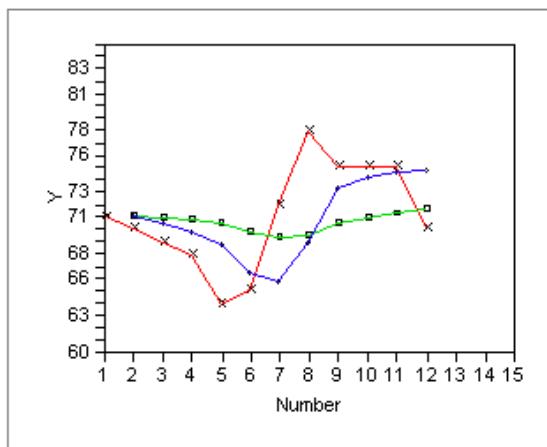
Time	yt	S( $\alpha=0.1$ )	Error	Error squared
1	71			
2	70	71	-1.00	1.00
3	69	70.9	-1.90	3.61
4	68	70.71	-2.71	7.34
5	64	70.44	-6.44	41.47
6	65	69.80	-4.80	23.04
7	72	69.32	2.68	7.18
8	78	69.58	8.42	70.90
9	75	70.43	4.57	20.88
10	75	70.88	4.12	16.97
11	75	71.29	3.71	13.76
12	70	71.67	-1.67	2.79

The sum of the squared errors (SSE) = 208.94.

The mean of the squared errors (MSE) is the SSE /11 = 19.0.

The MSE was again calculated for  $\alpha=0.5$  and turned out to be 16.29, so in this case we would prefer an  $\alpha$  of 0.5. Can we do better? We could apply the proven trial-and-error method. This is an iterative procedure beginning with a range of  $\alpha$  between 0.1 and 0.9. We determine the best initial choice for  $\alpha$  and then search between  $\alpha-\Delta$  and  $\alpha+\Delta$ . We could repeat this perhaps one more time to find the best  $\alpha$  to 3 decimal places.

### Exponential Smoothing: Original and Smoothed Values



Y    x—Original Y    ■—alpha = .1 ♦—alpha = .5

Pros of Smoothing (discussed in last class)

- Data smoothing clears the understandability of different important hidden patterns in the data set.
- Data smoothing can be used to help predict trends. Prediction is very helpful for getting the right decisions at the right time.
- Data smoothing helps in getting accurate results from the data.

Cons of data smoothing

- Data smoothing doesn't always provide a clear explanation of the patterns among the data.
- It is possible that certain data points being ignored by focusing the other data points.

<https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.htm>





## Normalization

- Also called data pre-processing, this is one of the crucial techniques for **data transformation in data mining**.
- Here, the data is transformed so that it falls under a given range.
- When attributes are on different ranges or scales, data modelling and mining can be difficult. Normalization helps in applying **data mining algorithms** and extracting data faster.
- Data normalization involves converting all data variable into a given range

person_name	Salary	Year_of_experience	Expected Position Level
Aman	100000	10	2
Abhinav	78000	7	4
Ashutosh	32000	5	8
Dishi	55000	6	7
Abhishek	92000	8	3
Avantika	120000	15	1
Ayushi	65750	7	5

The attributes salary and year\_of\_experience are on different scale and hence attribute salary can take high priority over attribute year\_of\_experience in the model.

The popular normalization methods are:

1. Min-max normalization
2. Decimal scaling
3. Z-score normalization

## 1. Min-Max Normalization:

It is the linear transformation of the original unstructured data. It scales the data from 0 to 1. It is calculated by the following formula:

$$v' = \frac{v - \text{min}_F}{\text{max}_F - \text{min}_F} (\text{new\_max}_F - \text{new\_min}_F) + \text{new\_min}_F ,$$

where  $V'$  = equivalent value for  $V$

$\text{new\_min} .. \text{new\_max}$  is new range

$\text{min..max}$  is Old range      for the current value of feature  $F$ .

Assume that the minimum and maximum values for the feature  $F$  are \$50,000 and \$100,000 correspondingly. It needs to range  $F$  from 0 to 1. In accordance with min-max normalization,  $v = \$80,000$  is transformed to:

$$v' = \frac{80,000 - 50,000}{100,000 - 50,000} + (1 - 0) + 0 = \frac{3}{5} = 0,6$$

- As you can see this technique enables to interpret the data easily.
- There are no large numbers, only concise data that do not require further transformation and can be used in decision-making process immediately.
- This transforms the original data linearly.

## 2. Z-Score Normalization:

- It is also called zero-mean normalization.
- The essence of this technique is the data transformation by the values conversation to a common scale where an average number equals zero and a standard deviation is one.
- A value is normalized to ' $v'$ ' under the formula:

$$v' = \frac{v - \bar{F}}{\sigma_F},$$

Here  $\bar{F}$  is the mean and  
 $\sigma_F$  is the standard deviation of feature  $F$ .

Example:

On the supposition that the mean of feature is \$65,000 and its standard deviation is \$ 18,000. Applying the z-score normalization we get the following mean of the value equals to \$85,800:

$$(85,800 - 65,000) / 18,000 = 1.1.16$$

### Z-Scores will help in situation like

- Sometimes we want to do more than summarize a bunch of scores.
- Sometimes we want to talk about particular scores within the bunch.
- We may want to tell other people about whether or not a score is above or below average.
- We may want to tell other people how far away a particular score is from average.
- We might also want to compare scores from different bunches of data. We will want to know which score is better.

Z-Scores tell us

- whether a particular score is equal to the mean, below the mean or above the mean of a bunch of scores.
- They can also tell us how far a particular score is away from the mean.
- Is a particular score close to the mean or far away?

### If a Z-Score....

- Has a value of 0, it is equal to the group mean.
- Is positive, it is above the group mean.
- Is negative, it is below the group mean.

- Is equal to +1, it is 1 Standard Deviation above the mean.
- Is equal to +2, it is 2 Standard Deviations above the mean.
- Is equal to -1, it is 1 Standard Deviation below the mean.
- Is equal to -2, it is 2 Standard Deviations below the mean.

### **Z-Scores Can Help Us Understand...**

- How typical a particular score is within bunch of scores.
- If data are normally distributed, approximately 95% of the data should have Z-score between -2 and +2.
- Z-scores that do not fall within this range may be less typical of the data in a bunch of scores.
- Thus Z score will help to identify the outlier

### **Z-Scores Can Help Us Compare...**

- Individual scores from different bunches of data. We can use Z-scores to standardize scores from different groups of data. Then we can compare raw scores from different bunches of data.

### **Advantages of the z score**

- The z-score is very useful when we are understanding the data. Some of the useful facts are mentioned below;  
The z-score is a very useful statistic of the data due to the following facts;  
It allows a data administrator to understand the probability of a score occurring within the normal distribution of the data.
- The z-score enables a data administrator to compare two different scores that are from different normal distributions of the data.

### **Is a higher or lower Z score better?**

- Suppose we have data from two persons. Person A has a high Z score value and person B have low Z Score value. In this case, the higher Z-score indicates that Person A is far away from person B.

### **What does a negative and a positive z score mean?**

- A negative z-score indicates that the data point is below the mean.  
A positive z-score indicates that the data point is above the mean.

### **Why is the mean of Z scores is 0?**

- The standard deviation of the z-scores is always 1 and similarly, the mean of the z-scores is always 1.  
Z-scores values above the 0 represent that sample values are above the mean.  
z-scores values below the 0 represent that sample values are below the mean.

In the case of squared z-scores, the sum of the squared z-scores is always equal to the number of z-score values.

### What is the meaning of the high Z score and low Z score?

- Suppose we have a high z-score value then it means a very low probability of data above this z-score.
- Suppose we have a low z-score value then it means a very low probability of data below this z-score.

### Comparison of Min-Max Normalization and Z-Score Normalization

Min-max normalization	Z-score normalization
Not very well efficient in handling the outliers	Handles the outliers in a good way.
Min-max Guarantees that all the features will have the exact same scale.	Helpful in the normalization of the data but not with the <i>exact</i> same scale.

<https://www.statisticshowto.com/probability-and-statistics/z-score/>

<https://www.codecademy.com/articles/normalization>

<https://statistics.laerd.com/statistical-guides/standard-score-1.php> (For extra reading)

### **3. Decimal Scaling:**

- It normalizes the values of an attribute by changing the position of their decimal points
- The number of points by which the decimal point is moved can be determined by the absolute maximum value of attribute A.
- A value,  $v$ , of attribute A is normalized to  $v'$  by computing

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

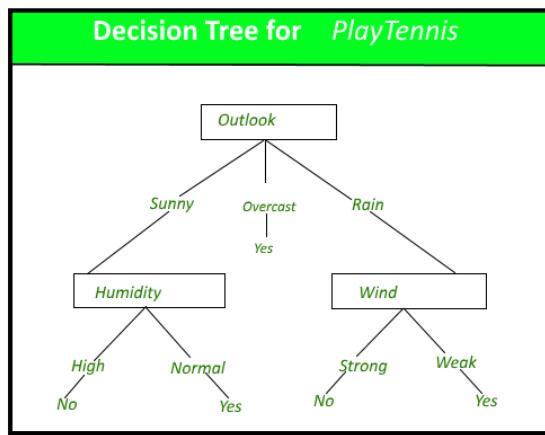
- where  $j$  is the smallest integer such that  $\text{Max}(|v'|) < 1$ .

For example:

- Suppose: Values of an attribute P varies from -99 to 99.
- The maximum absolute value of P is 99.
- For normalizing the values we divide the numbers by 100 (i.e.,  $j = 2$ ) or (number of integers in the largest number) so that values come out to be as 0.98, 0.97 and so on.

## Discretization

- This is a process of converting continuous data into a set of data intervals.
- Continuous attribute values are substituted by small interval labels.
- This makes the data easier to study and analyze.
- If a continuous attribute is handled by a **data mining** task, then its discrete values can be replaced by constant quality attributes.
- This improves the efficiency of the task.
- This method is also called data reduction mechanism as it transforms a large dataset into a set of categorical data.
- Discretization also uses **decision tree-based algorithms** to produce short, compact and accurate results when using discrete values.



- It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes.
- Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values.
  - For **example**, (1-10, 11-20) (age:- young, middle age, senior).

## Transformation thru Discretization (Binning)

Data-mining applications often involve quantitative data (Numerical or Continuous). However, learning from quantitative data is often less effective and less efficient than learning from qualitative data (Categorical or Discrete). Discretization addresses this issue by transforming quantitative data into qualitative data.

- Binning, also known as quantization is used for transforming continuous numeric features into discrete ones (categories).
- Numerical input variables may have a highly skewed or non-standard distribution.
  - This could be caused by outliers in the data, multi-modal distributions, highly exponential distributions, and more.
- Many machine learning algorithms prefer or perform better when numerical input variables have a standard probability distribution.
- Discretization is the process through which we can transform continuous variables, models or functions into a discrete form.
- We do this by creating a set of contiguous intervals (or bins) that go across the range of our desired variable/model/function.
  - Discretization is also known as Binning

## Why Discretization is Important

Mathematical problems with continuous data have an infinite number of degrees of freedom (DoF) but our calculations cannot go on forever. Data Scientists require using Discretization for a number of reasons. Many of the top contributions on Kaggle use discretization for some of the following reasons:

### 1. Fits the problem statement

- Often, it is easier to understand continuous data (such as weight) when divided and stored into meaningful categories or groups.
  - For example, we can divide a continuous variable, weight, and store it in the following groups :  
***Under 100 lbs (light), between 140–160 lbs (mid), and over 200 lbs (heavy)***
- This will make the structure more useful as no objective difference between variables falling under the same category (weight class in our case).
  - In our example, weights of *85 lbs* and *56 lbs* convey the same information (the object is light). Therefore, discretization helps make our data easier to understand if it fits the problem statement.

## **2. Interprets features**

- Continuous features have a smaller chance of correlating with the target variable due to infinite degrees of freedom and may have a complex non-linear relationship. Thus, it may be harder to interpret such a function.
- After discretizing a variable, groups corresponding to the target can be interpreted.

## **3. Incompatible with models/methods**

- Certain models may be incompatible with continuous data, for example, alternative decision-tree models such as a Random-Forest model is not suitable for continuous features.
- Feature engineering methods, for example any entropy-based methods may not work with continuous data, thus we would discretize variables to work with different models & methods.

### **Signal-to-Noise Ratio**

- When we discretize a model, we are fitting it to bins and reducing the impact of small fluctuation in the data. Often, we would consider small fluctuations as noise. We can reduce this noise through discretization.
- This is the process of “smoothing”, wherein each bin smoothens fluctuations, thus reducing noise in the data.

### **Approaches to Discretization**

- Unsupervised:
  - Fixed width binning
    - We manually create fix width bins based on some rules and domain knowledge.
    - on the basis of some domain knowledge, rules or constraints. Just like the name indicates, in fixed-width binning, we have specific fixed widths for each of the bins which are usually pre-defined by the user analyzing the data. Each bin has a pre-fixed range of values which should be assigned to that bin
  - Equal-Width
  - Equal-Frequency
  - Statistical binning or Binning based on Rounding (Divide and round/floor/ceiling)
- Adaptive Binning

- Based on the statistical information of the data boundary is decided
  - K-Means
  - Quantile based
- Supervised:
  - Decision Trees

## **Equal Width Bin:**

In Equal width, we divide the data in equal widths.

$$\text{width} = (\max - \min) / N$$

- N is user defined
- Thus, i<sup>th</sup> interval range will be  $[A + (i-1)w, A + iw]$  where  $i = 1, 2, 3, \dots, N$

Example: Convert the list into 3 Equal Width Bins:

List = 5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215

**Result:** 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3

Width =  $215 - 5 / 3 = 210 / 3 = 70$

Bin 1: 5 to  $70 + 5 = 75$

Bin2 : 75 to  $145 (75+70)$

Bin3: 145 to  $215 (145+70)$

Result:

bin1: 5, 10, 11, 13, 15, 35, 50, 55, 72

bin2: 92

bin3: 204, 215

Challenges:

- How to define N?
- To create the sorted order list
- Skewed data cannot be handled well by this method

## **2- Equal Frequency Binning**

The algorithm divides the data into  $k$  groups which each group contains approximately same number of values. For the both methods, the best way of determining  $k$  is by looking at the histogram and try different intervals or groups.

Challenge:

- Equal frequency may not be possible due to repeated values.

- **Data :** 0, 4, 12, 16, 16, 18, 24, 26, 28

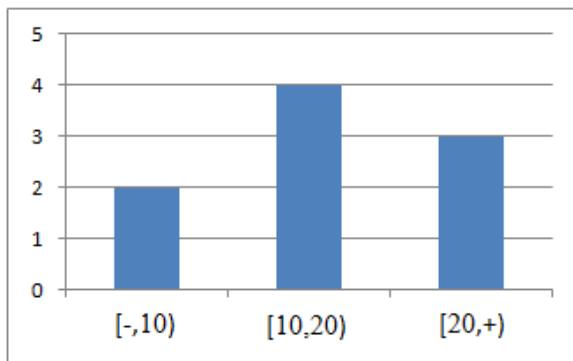
- **Equal width**

- Bin 1: 0, 4 [ -, 10 )
- Bin 2: 12, 16, 16, 18 [ 10, 20 )
- Bin 3: 24, 26, 28 [ 20, + )

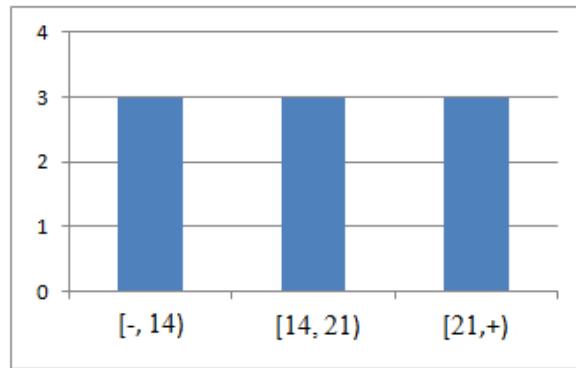
- **Equal frequency**

- Bin 1: 0, 4, 12 [ -, 14 )
- Bin 2: 16, 16, 18 [ 14, 21 )
- Bin 3: 24, 26, 28 [ 21, + )

**Equal width**



**Equal frequency**



The drawback in using fixed-width binning is that

- due to manually deciding the bin ranges, we can end up with irregular bins which are not uniform based on the number of data points or values which fall in each bin.
- Some of the bins might be densely populated and some of them might be sparsely populated or even empty!

## Adaptive Binning

- In Fixed-Width Binning, bin ranges are manually decided. So, we usually end up in creating irregular bins which are not uniform based on the number of data points or values which fall under each bin. Some of the bins might be densely populated and some of them might be sparsely populated or even empty.
  - For example, bins 0, 5 and 8 are empty in our case.
- In Adaptive Binning, data distribution itself decides bin ranges for itself. No manual intervention is required. So, the bins which are created are uniform in terms of number of data points in it.

## Quantile based binning

- It is a good strategy to use for adaptive binning.
- Quantiles are specific values or cut-points which help in partitioning the continuous valued distribution of a specific numeric field into discrete contiguous bins or intervals.
- Thus, **q-Quantiles** help in partitioning a numeric attribute into q equal partitions.
- Popular examples of quantiles include the *2-Quantile* known as the *median* which divides the data distribution into two equal bins,
- *4-Quantiles* known as the *quartiles* which divide the data into 4 equal bins [used in Box plot]
- *10-Quantiles* also known as the *deciles* which create 10 equal width bins. Let's now look at the data distribution for the developer Income field.

Note :

- The algorithm assumes that the data quantiles are (mostly) unique and they divide  $N$  observations into  $k$  groups that each have approximately  $N/k$  observations.
- The repeated value can occupy more than one quantile value
  - If there are more than  $N/k$  repeated values,
  - if a particular value is repeated more than  $2N/k$  times
- For example,
  - The diastolic blood pressure data is given with 5209 observations
  - It is to be divided into 10 groups that each have approximately 10% of the data.
  - The value 80 appears in 711<sup>th</sup> position (which is about 14% of the data)
  - His value will reside in 2<sup>nd</sup> quartile
- If 80 is repeated from 711<sup>th</sup> position to 1500<sup>th</sup> position in such a way that it is appeared in 2<sup>nd</sup> and 3<sup>rd</sup> both quartiles

- If you do not want to split the value 80 across two bins, merge these bins and it will result in nine bins instead 10.  
[\(https://blogs.sas.com/content/iml/2014/11/05/binning-quantiles-rounded-data.html#prettyPhoto\)](https://blogs.sas.com/content/iml/2014/11/05/binning-quantiles-rounded-data.html#prettyPhoto)

## Case Study

- Data set: **2016 FreeCodeCamp Developer\Coder survey**
  - about various attributes pertaining to coders and software developers.

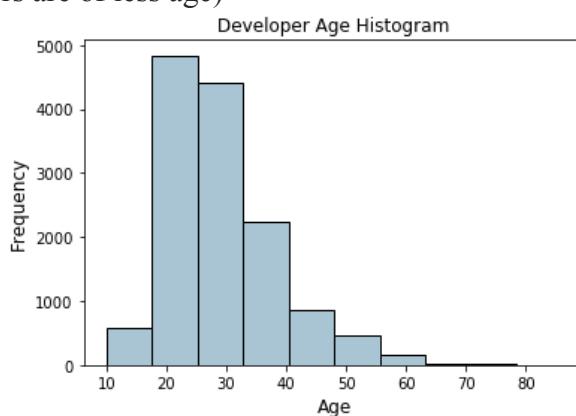
	ID.x	EmploymentField	Age	Income
0	cef35615d61b202f1dc794ef2746df14	office and administrative support	28.0	32000.0
1	323e5a113644d18185c743c241407754	food and beverage	22.0	15000.0
2	b29a1027e5cd062e654a63764157461d	finance	19.0	48000.0
3	04a11e4bcb573a1261eb0d9948d32637	arts, entertainment, sports, or media	26.0	43000.0
4	9368291c93d5d5f5c8cdb1a575e18bec	education	20.0	6000.0

### Sample attributes from the FCC coder survey dataset

- The ID.x variable is basically a unique identifier for each coder\developer who took the survey and the other fields are pretty self-explanatory.

## Fixed-Width Binning

- Just like the name indicates, in fixed-width binning, we have specific fixed widths for each of the bins which are usually pre-defined by the user analyzing the data.
- Each bin has a pre-fixed range of values which should be assigned to that bin on the basis of some domain knowledge, rules or constraints.
- Binning based on rounding is one of the ways, where you can use the rounding operation which we discussed earlier to bin raw values.
- **Feature:Age** frequency distribution shown in the histogram indicate slightly right skewed values (Developers are of less age)



## Histogram depicting developer age distribution

We will now assign these raw age values into specific bins based on the following scheme

Age Range: Bin

-----  
0 - 9 : 0  
10 - 19 : 1  
20 - 29 : 2  
30 - 39 : 3  
40 - 49 : 4  
50 - 59 : 5  
60 - 69 : 6  
... and so on

**Age Bin = Floor (Age/10)**

ID.x	Age	Age_bin_round
1071	6a02aa4618c99fdb3e24de522a099431	17.0
1072	f0e5e47278c5f248fe861c5f7214c07a	38.0
1073	6e14f6d0779b7e424fa3fdd9e4bd3bf9	21.0
1074	c2654c07dc929cdf3dad4d1aec4ffbb3	53.0
1075	f07449fc9339b2e57703ec7886232523	35.0

Binning by rounding

But what if we need more flexibility? What if we want to decide and fix the bin widths based on our own rules\logic? Binning based on custom ranges will help us achieve this.

**Let's define some custom age ranges for binning developer ages using the following scheme.**

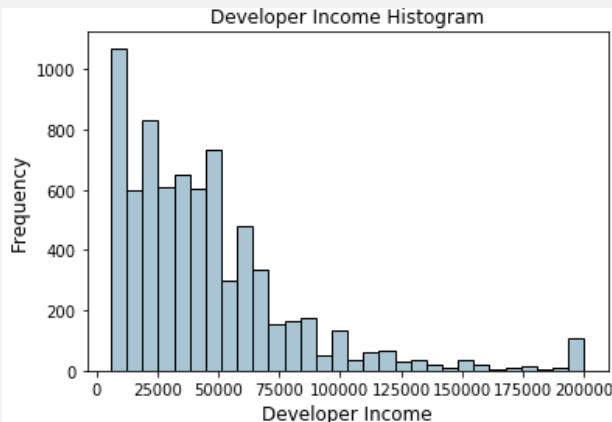
Age Range : Bin

-----  
0 - 15 : 1  
16 - 30 : 2  
31 - 45 : 3  
46 - 60 : 4  
61 - 75 : 5  
75 - 100 : 6

ID.x	Age	Age_bin_round	Age_bin_custom_range	Age_bin_custom_label
1071	6a02aa4618c99fdb3e24de522a099431	17.0	(15, 30]	2
1072	f0e5e47278c5f248fe861c5f7214c07a	38.0	(30, 45]	3
1073	6e14f6d0779b7e424fa3fdd9e4bd3bf9	21.0	(15, 30]	2
1074	c2654c07dc929cdf3dad4d1aec4ffbb3	53.0	(45, 60]	4
1075	f07449fc9339b2e57703ec7886232523	35.0	(30, 45]	3

Custom binning scheme for developer ages

### Adaptive Binning



Histogram depicting developer income distribution

- The above distribution depicts a right skew in the income with lesser developers earning more money and vice versa.
- Let's take a *4-Quantile* or a *quartile* based adaptive binning scheme.

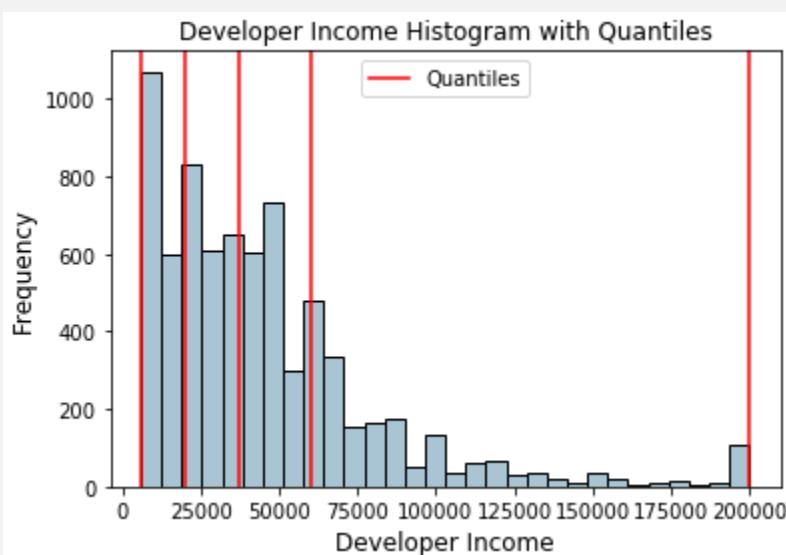
### quartiles obtained :

- Income range [0..200000]
- quantile\_list = [0, .25, .5, .75, 1.]

Quantile	Value
0.00	6000.0 (smallest value)
0.25	20000.0
0.50	37000.0
0.75	60000.0
1.00	200000.0

	ID.x	Age	Income	Income_quantile_range	Income_quantile_label
4	9368291c93d5d5f5c8cdb1a575e18bec	20.0	6000.0	(5999.999, 20000.0]	0-25Q
5	dd0e77eab9270e4b67c19b0d6bbf621b	34.0	40000.0	(37000.0, 60000.0]	50-75Q
6	7599c0aa0419b59fd11ffede98a3665d	23.0	32000.0	(20000.0, 37000.0]	25-50Q
7	6dff182db452487f07a47596f314bddc	35.0	40000.0	(37000.0, 60000.0]	50-75Q
8	9dc233f8ed1c6eb2432672ab4bb39249	33.0	80000.0	(60000.0, 200000.0]	75-100Q

Quantile based bin ranges and labels for developer incomes



Histogram depicting developer income distribution with quartile values

- The red lines in the distribution above depict the quartile values and our potential bins.
- This will result in uniform distribution of the values

## **Kmeans clustering binning**

K-means clustering-based binning

For the k-means clustering, we implement the proposed technique in following steps:

1. Initial groups are obtained using the k-means algorithm on the first lot (Training Data).
2. The 1<sup>st</sup> instance from the second lot (Test Data set) is obtained and the Euclidean distances between this vector and centroid of each group obtained from the first phase is calculated. The device is assigned to the group to which it is nearest.
3. To follow the k-means algorithm in this case,
  - The average sum of distance of this point and each point of that group to the centroid should be calculated.
    - This process will be repeated for every group and the device should be assigned to the group, which gives the minimum average sum.
    - Clearly, such an approach will increase the complexity of binning process to a high degree and make it unusable in production environment.
  - Another approach for the best approximation is : the device is assigned to whichever groups centroid it is closest.
4. The centroid of the group to which the device is added, is recalculated and updated for that group.
5. This procedure is carried out for all the Data of the new Data\_set

### **Limitations:**

- K-Means doesn't improve the value spread
- It can handle outliers, however a centroid bias may exist.
- Can be combined with categorical encoding

**Extra Reading :** [https://www.cs.colostate.edu/~malaiya/turakhia\\_paper.pdf](https://www.cs.colostate.edu/~malaiya/turakhia_paper.pdf)

## **Tree based**

- Decision Tree does not improve the value spread
- It can handle outliers well as trees are robust to outliers.
- Creates monotonic relationships

Extra reading : <https://hiweller.github.io/colordistance/binning-methods.html>

## **Data Reduction in Data Mining**

**The method of data reduction may achieve a condensed description of the original data which is much smaller in quantity but keeps the quality of the original data.**

**Methods of data reduction:**

### **Reducing Rows**

- **Missing value removal**
- **Data Cube Aggregation**

### **Reducing Columns**

- **Feature reduction (Dimension reduction)**
- **Feature selection (Feature extraction)**
- **Data Compression**
- **Numerosity Reduction**
- **Discretization & Concept Hierarchy Operation**

## **Reduction in Rows**

### **1. Data Cube Aggregation: (Reduction in Rows)**

- This technique is used to aggregate data in a simpler form.
- For example, imagine that information you gathered for your analysis for the years 2012 to 2014, that data includes the revenue of your company every three months. They involve you in the annual sales, rather than the quarterly average,
- So we can summarize the data in such a way that the resulting data summarizes the total sales per year instead of per quarter. It summarizes the data.

### **2. Missing Values Ratio.**

- Data columns with too many missing values are unlikely to carry much useful information. Thus data columns with number of missing values greater than a given threshold can be removed. The higher the threshold, the more aggressive the reduction.

## **Reduction of Dimensions:**

- Whenever we come across any data which is weakly important, then we use the attribute required for our analysis. It reduces data size as it eliminates outdated or redundant features.

An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not.

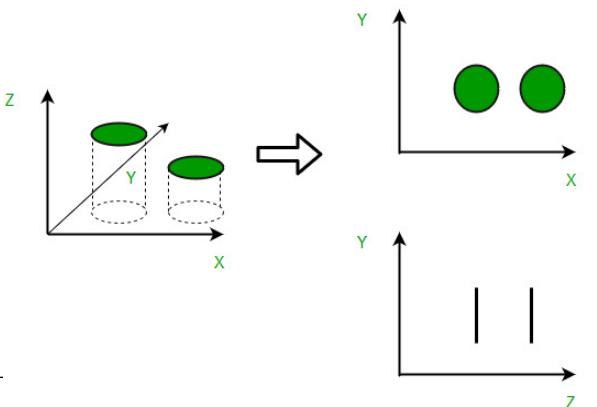
- This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc.
- However, some of these features may overlap.

In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems.

A 3-D classification problem can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line.

The below figure illustrates this concept, where a 3-D feature space is split into two 1-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further.

Dimensionality Reduction



Feature Selection Can be :

- i. **Univariate Feature Selection**
- ii **Multi Variate**

**Univariate Feature Selection:** This technique involves more of a manual kind of work. Visiting every feature and checking its importance with the target. There are some great tricks that you should keep under your sleeve to implement Univariate Feature Selection.

- proper **domain knowledge** required ‘
- **Checking the variance** (yes the ever confusing bias-variance trade-off :) of all features. The thumb rule here is set a threshold value (say a feature with 0 variance means it has the same value for every sample so such a feature would not bring any predictive power to the model) remove feature accordingly.
- **Use of Pearson Correlation:** It might be the most applicable technique of three.

So, in a nutshell, it gives us the inter-dependence between the target variable and a feature.

When you have a lot of features (like hundreds or thousands of them), then it really becomes impossible to go & manually check for every one of them or if you don't have enough domain knowledge then you got to trust this following technique. So put in layman's term it is nothing but selecting multiple features at once.

## Multivariate Feature Reduction

### Feature Reduction (Extracting New feature):

This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions

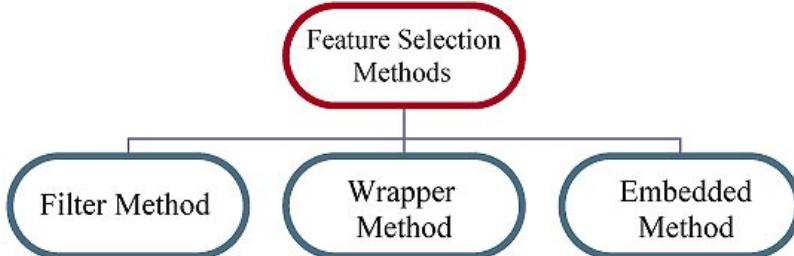
The various methods used for dimensionality reduction include

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

# Multivariate Feature Selection:

In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem.

**Multivariate Feature Selection is broadly divided into three categories:**



## Filter Method :

→ Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms.

→ Filter methods apply some ranking over features. The ranking denotes how ‘useful’ each feature is likely to be for classification. Once this ranking has been computed, a feature set composed of the best N features is created.

→ Features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. (The correlation is a subjective term here).



1. **Pearson’s Correlation:** Oh yes! Pearson Correlation is Filter method. We have already discussed it.
2. **Variance Thresholds:** This too, we have already discussed.
3. **Linear discriminant analysis (LDA):** The goal is to project a dataset onto a lower-dimensional space with good class-separability and also reduce computational costs. In simple word LDA brings all the higher dimensional variable (which we can’t plot and analyse) onto 2D graph & while doing so removes the useless feature.

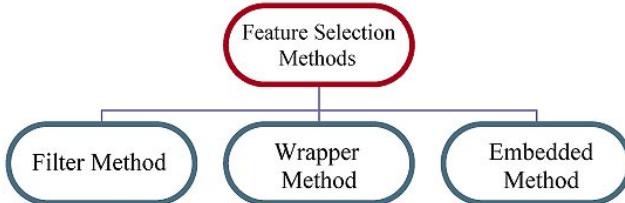
→ LDA is also ‘**Supervised Dimension Reduction**’ technique and more kind of **Feature Extraction** than **Selection** (as it is creating kind of a new variable by reducing its dimension). So it works only on labelled data.

→ It maximizes the *separability* between classes.

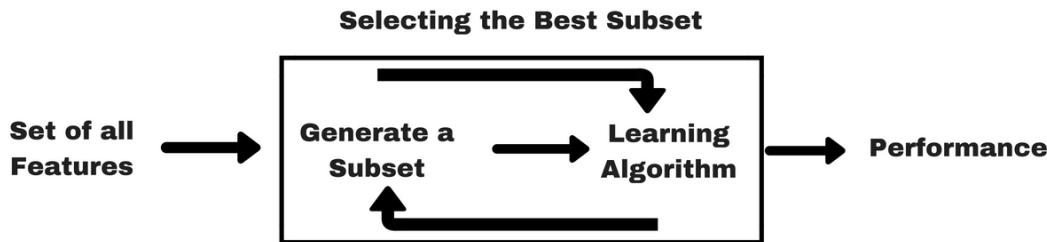
4. **ANOVA ( Analysis of variance)** It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature (Response or target). It provides a statistical test of whether the means of several groups are equal or not.
5. **Chi-Square:** It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

**Note :** filter method do not remove multi-collinearity. So, you must deal with multi-collinearity of features as well before training models for your data.

Feature\Response	Continuous	Categorical
Continuous	Pearson's Correlation	LDA
Categorical	Anova	Chi-Square



## Wrapper Method:



- Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset.
  - Wrapper methods are so called because they wrap a classifier up in a feature selection algorithm. Typically a set of features is chosen; the efficiency of this set is determined; some perturbation is made to change the original set and the efficiency of the new set is evaluated.
  - The problem with this approach is that feature space is vast and looking at every possible combination would take a large amount of time and computation.
  - The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.
1. **Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
  2. **Backward Elimination:** In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on the removal of features.
  3. **Recursive Feature Elimination (RFE):** It works by recursively removing attributes and building a model on those attributes that remain. It uses an external estimator that assigns weights to features (for example, the coefficients of a linear model) to identify which attributes (and the combination of attributes) contribute the most to predicting the target attribute.
    - It is a greedy optimization algorithm which aims to find the best performing feature subset.
    - It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration.
    - It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination

## **Embedded Method:**

→ Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.

→ So this are not any kind of special feature selection or extraction techniques and they also help in avoiding overfitting.

1. Select k-best in Random Forest
2. Gradient boosting machine (GBM)

## **Difference between Filter and Wrapper method**

<b>Filter Method</b>	<b>Wrapper Method</b>
Measure the relevance of features by their correlation with dependent variable	Measure the usefulness of a subset of feature by actually training a model on it.
Much faster compared to wrapper methods as they do not involve training the models	Wrapper methods are computationally very as they involve training the models
Use statistical methods for evaluation of a subset of features	Wrapper methods use cross validation.

### **3. Data Compression:**

- The data compression technique reduces the size of the files using different encoding mechanisms (Huffman Encoding & run-length Encoding). We can divide it into two types based on their compression techniques.
  - **Lossless Compression –**  
Encoding techniques (Run Length Encoding) allows a simple and minimal data size reduction. Lossless data compression uses algorithms to restore the precise original data from the compressed data.
  - **Lossy Compression –**  
Methods such as Discrete Wavelet transform technique, PCA (principal component analysis) are examples of this compression. For e.g., JPEG image format is a lossy compression, but we can find the meaning equivalent to the original the image. In lossy-data compression, the decompressed data may differ to the original data but are useful enough to retrieve information from them.

### **4. Numerosity Reduction:**

- In this reduction technique the actual data is replaced with mathematical models or smaller representation of the data instead of actual data, it is important to only store the model parameter.
- Or non-parametric method such as clustering, histogram, sampling.

### **5. Discretization & Concept Hierarchy Operation:**

- Techniques of data discretization are used to divide the attributes of the continuous nature into data with intervals.
- We replace many constant values of the attributes by labels of small intervals. This means that mining results are shown in a concise, and easily understandable way.
  - **Top-down discretization –**  
If you first consider one or a couple of points (so-called breakpoints or split points) to divide the whole set of attributes and repeat of this method up to the end, then the process is known as top-down discretization also known as splitting.
  - **Bottom-up discretization –**  
If you first consider all the constant values as split-points, some are discarded through a combination of the neighbourhood values in the interval, that process is called bottom-up discretization.
- Concept Hierarchies:
  - It reduces the data size by collecting and then replacing the low-level concepts (such as 43 for age) to high-level concepts (categorical variables such as middle age or Senior).

Some interesting Feature reduction techniques are

- **Low Variance Filter.**
- data columns with little changes in the data carry little information.
- Thus all data columns with variance lower than a given threshold are removed.
- Note: variance is range dependent; therefore normalization is required before applying this technique.

- **High Correlation Filter.**

- Data columns with very similar trends are also likely to carry very similar information.
- In this case, only one of them will suffice to feed the machine learning model.
- Here we calculate the correlation coefficient between numerical columns and between nominal columns as the **Pearson's Product Moment Coefficient** and the **Pearson's chi square value** respectively.
- Pairs of columns with correlation coefficient higher than a threshold are reduced to only one. A word of caution: correlation is scale sensitive; therefore column normalization is required for a meaningful correlation comparison.

### **Random Forests / Ensemble Trees.**

- Decision Tree Ensembles, also referred to as random forests, are useful for feature selection in addition to being effective classifiers.
- One approach to dimensionality reduction is to generate a large and carefully constructed set of trees against a target attribute and then use each attribute's usage statistics to find the most informative subset of features.
- Specifically, we can generate a large set (2000) of very shallow trees (2 levels), with each tree being trained on a small fraction (3) of the total number of attributes.
- If an attribute is often selected as best split, it is most likely an informative feature to retain. A score calculated on the attribute usage statistics in the random forest tells us – relative to the other attributes – which are the most predictive attributes.

### **Principal Component Analysis (PCA).**

- **Principal Component Analysis (PCA)** is a statistical procedure that orthogonally transforms the original  $n$  coordinates of a data set into a new set of  $n$  coordinates called principal components.
- As a result of the transformation, the first principal component has the largest possible **variance**; each succeeding component has the highest possible variance under

the constraint that it is **orthogonal** to (i.e., uncorrelated with) the preceding components.

- Keeping only the first  $m < n$  components reduces the data dimensionality while retaining most of the data information, i.e. the variation in the data.
- Notice that the PCA transformation is sensitive to the relative scaling of the original variables. Data column ranges need to be normalized before applying PCA.
- Also notice that the new coordinates (PCs) are not real system-produced variables anymore. Applying PCA to your data set loses its interpretability. If interpretability of the results is important for your analysis, PCA is not the transformation for your project

- **Backward Feature Elimination.** In this technique, at a given iteration, the selected classification algorithm is trained on  $n$  input features. Then we remove one input feature at a time and train the same model on  $n-1$  input features  $n$  times. The input feature whose removal has produced the smallest increase in the error rate is removed, leaving us with  $n-1$  input features. The classification is then repeated using  $n-2$  features, and so on. Each iteration  $k$  produces a model trained on  $n-k$  features and an error rate  $e(k)$ . Selecting the maximum tolerable error rate, we define the smallest number of features necessary to reach that classification performance with the selected machine learning algorithm.
- **Forward Feature Construction.** This is the inverse process to the Backward Feature Elimination. We start with 1 feature only, progressively adding 1 feature at a time, i.e. the feature that produces the highest increase in performance. Both algorithms, Backward Feature Elimination and Forward Feature Construction, are quite time and computationally expensive. They are practically only applicable to a data set with an already relatively low number of input columns.

○

For reference

- **Step-wise Forward Selection –**

The selection begins with an empty set of attributes later on we decide best of the original attributes on the set based on their relevance to other attributes. We know it as a p-value in statistics.

Suppose there are the following attributes in the data set in which few attributes are redundant.

Initial attribute Set: {X1, X2, X3, X4, X5, X6}

Initial reduced attribute set: {}

Step-1: {X1}

Step-2: {X1, X2}

Step-3: {X1, X2, X5}

Final reduced attribute set: {X1, X2, X5}

- **Step-wise Backward Selection –**

This selection starts with a set of complete attributes in the original data and at each point, it eliminates the worst remaining attribute in the set.

Suppose there are the following attributes in the data set in which few attributes are redundant.

Initial attribute Set: {X1, X2, X3, X4, X5, X6}

Initial reduced attribute set: {X1, X2, X3, X4, X5, X6 }

Step-1: {X1, X2, X3, X4, X5}

Step-2: {X1, X2, X3, X5}

Step-3: {X1, X2, X5}

Final reduced attribute set: {X1, X2, X5}

- **Combination of forward and Backward Selection –**

It allows us to remove the worst and select best attributes, saving time and making the process faster.

# Itemset Mining using Horizontal and Vertical Data Format

A. Subashini, Assistant Professor, Government Arts

College,C.Mutlur,Chidambaram,Tamilnadu,India,subanandh31@gmail.com

M. Karthikeyan, Assistant Professor, Department of Computer and Information science,

Annamalai university,Tamilnadu,India,karthiaucse@gmail.com

**Abstract** In data mining, Item set mining is an essential subfield. It is made to determine patterns which are interesting and useful in transaction database. The frequent item set mining task is to discover collections of items that appear frequently composed in transactions made by customers and the next task of infrequent item set mining is to discover rare items that appear in transaction purchased by customers rarely. The task of Item set mining is used to detect frequently co-occurring item set in database. In supermarket product, retailer wants sufficient information are in need to decide the placement of products, promotion strategies and improving the profit of the supermarket and customer satisfaction. Market Basket analysis can help retailer to plan which items to put on sale at reduced rates. In this study, supermarket datasets are mined from association rule mining methods comparing time and memory usage of using existing Apriori based algorithms in order to generate frequent Itemsets and rare item sets, so users will be able to reduce the time of decision making, improve the performance and operation, and increase the profit of their organizations.

**Keywords -** *Apriori algorithm , Data mining ,Frequent itemsets, rare Itemsets.*

## I. INTRODUCTION

Data Mining is considered as a vital phase in the discovery of knowledge rather than the process itself [1]. There are many other important steps which come along the entire process. Data cleaning and integration, selecting data which comes under the data pre-processing techniques. The data selected is mined and the knowledge discovered is evaluated and presented to the user. The user should be able to interpret the information discovered by the data mining techniques and this is why knowledge presentation is also of prime importance in the data process. Association rule mining (ARM) is in several application such as inventory control, mobile mining, educational mining, market and risk management, telecommunication networks, graph mining, etc. The patterns discovered are based on repeated items in the dataset, though from these data some data can be either irrelevant or obvious. Uninteresting Itemsets sometimes and too many itemsets generated while low support threshold is given, but a high MST misses few rare itemsets. The database in real world may have items that are of varying occurrences. Some items look frequently in transactions and some of them look infrequently. The frequent itemsets by setting high or equal minimum support threshold value and the rare itemsets can also be interesting, by setting a low support threshold values. The database of market basket is only an

occurrence of data with boolean attributes indicating whether an item is present in the transaction or not. If an item is absence in a transaction, it is denoted by '0' and its presence denoted by '1'. The analysis remains unaltered even when there are more values for an attribute other than true or false.

In data mining ,Market Basket Analysis is the best example for itemset mining ,buying habits of the customer can be analysis by the departmental manager with an item x or item y. This process helps the departmental manager to make a plan for effective marketing strategies. Definition of data mining is given as "A process of non-trivial extraction of implicit, previously unknown and potentially useful information from the data stored in a database". One traditional approach to discover the relationships via support and confidence was proposed by Agrawal, Imielinski and Swami in 1993[3]. Using this approach, item sets that can be observed frequently are identified. The minimum value sets by user is called "minimum support threshold" to segregate the frequent item sets from infrequent item sets.

## II. LITERATURE SURVEY

Association Rule Mining (ARM) was introduced by Aggarwal et al. which is the area of Data Mining, discovering interesting and hidden relationships in large

datasets [3],[9].In [4], the authors developed an improved Apriori version known as FP-growth method that will help to provide more efficiency than exclusive one and overcome the problems of traditional Apriori algorithm. The test results of the algorithm confirmed that the requirements have higher efficiency in relation to time, less storage space and CPU usage as compared to Apriori Algorithm. In[11],supermarket dataset is in packtpub.com website supermarket-dataset is downloaded for this analysis. In [5], the authors consider both the existence and non-appearance of items as a basis for generating rules. They measured the association rules using chi-square test is tested on census data. In [6], the authors have generated the frequent item sets using Genetic Algorithm which is very simple and efficient. The advantage of this algorithm is that they perform global search and its time complexity is less as compared to other algorithms. But this approach is not feasible for identifying the negative and rare Item sets.

In [2], the author gave the brief study of various Data Mining algorithms for mining frequent item sets using association rules. Among all algorithms, FP-Growth algorithm is the most capable method to find the frequent item sets. The conditional structure was constructed by FP\_growth to find relevant item sets without candidate generation. FP\_growth consumes less memory so, the efficiency of the algorithm is effective but is unable to identify rare item sets. The major drawback of this method is that it generates the huge number of rules. To generate MRI's Apriori-Rare [7] algorithm. In [8], the authors derived a method known as FP-growth, based on FP-tree. The first probe of the database develops a list of frequent items in which items in the descending order are compacted into a frequent-pattern tree or FP-tree. The FP-tree is extracted to generate item sets. The limitation is based on the minimum support threshold. The proposal of frequent itemset mining, hundreds of algorithms have been proposed on various kinds of extensions and applications, ranging from scalable data mining methodologies, to handling a various data types, different extended mining tasks, and a several of new applications (Han, Cheng, Xin, & Yan, 2007).

In [10], the authors proposed a Hash-Based algorithm, which is specifically operational for the generation of candidate set for large item sets. In addition, the generation of smaller candidate sets enables the user to effectively trim the transaction database at much earlier stage of the iterations and thus reduce the computational cost for later stages. This algorithm is not effective for graphical data. Minimal Infrequent Itemset algorithm is used for mining minimal infrequent itemsets. An itemset is said to be minimal infrequent itemset if itemset is lesser than or equal to maximum. An Open-Source Data Mining Library by Dr.philippe-fournier-viger. Therefore, this paper analyzes a number of FIM and RareIM algorithms to

provide an overview of the FIM and Rare itemset comparing the different algorithms of itemset count, maximum memory used and time complexity. The previous works done on Frequent itemsets and Rare itemsets algorithms are presented in next Section .In result and discussion presents a table which provides a comparison of the fundamental and significant Frequent and Rare algorithms that have been proposed by other researchers.

### III. PROPOSED METHOD

*Association Rule Mining*—ARM is the task of identifying meaningful implication rules of the form from the transaction dataset that is  $I = \{I_1, I_2, \dots, I_m\}$  and  $D =$ the task relevant data,be a set of database transaction , $X \rightarrow Y$  exhibited in a data set (i.e., relation), where  $X$  and  $Y$  are subsets of the items  $I$ (i.e., possible distinct values of columns of a data set) and  $X \cap Y = \emptyset$  (Agrawal,Imieliski, & Swami, 1993). The degree to which a rule is meaningful is defined by:

- i) support, the number of times both items  $X$  and  $Y$  are found in the data set, and
- ii) confidence, the number of times that  $X \rightarrow Y$  holds true relative to all occurrences of  $X$ .

Mining association rules usually involves two steps

- i) identifying frequent item sets (i.e.,  $X \cup Y$  that meets a minimum support threshold), and that not meets minimum support then identifying as Rare itemset.
- ii) deriving association rules from the item sets that meet a level of confidence.

Apriori searches the space of all patterns in an iterative bottom-up breadth-first manner. Each iteration obtains counts for its current set of candidate patterns and removes from further consideration any candidate patterns that are not frequent or cannot be frequent. Apriori has proved to be efficient for mining frequent patterns of small length. However, for long patterns Apriori can be I/O intensive since each iteration requires a full scan of the data set.

*Data formats*—Itemset mining for frequent and rare, there are two data formats to be implemented. Horizontal and vertical data format, the horizontal data format is the same as that stored in a database. The horizontal data format is converted into vertical data format, with the transaction identifiers grouped for each item. The vertical data format are more effective than those based on the horizontal data format in Itemsets mining algorithms, Because the database scans only once and compute the supports of itemsets fast, it takes more memory for additional information, like Tid\_sets is disadvantage.

#### A. APRIORI ALGORITHM

Apriori (Agrawal and Srikant 1994) is an algorithm that mines the frequent itemsets for generating Boolean association rules. The technique used level\_wise iterative

search to discover  $(k+1)$ -itemsets from  $k$ -itemsets. An example of transactional data from T100 to T900 that contains of product items I1,I2,I3,I4,I5 being purchased at different transactions is shown in Table 1.

#### Algorithm: Apriori

**Input:**  $D$ , database of transactions

**min\_sup:** Minimum Support Threshold

**Output:**  $L$ , Frequent itemsets in  $D$

1. the database Scan to get the support of each 1-itemset, compare with min\_sup and get the frequent 1-itemset.
2. Use  $L_{K-1}$ , Join  $L_{K-1}$  to generate the candidate  $k$ -itemset.
3. Scan the database to get the support of each candidate  $k$ -itemset, compare with min\_sup and get the frequent  $k$ -itemset.
4. Repeat the steps 2 to 3 until candidate itemset is null. If null, generate all subsets for each frequent itemset.

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

Table 1 Sample of transactional data.

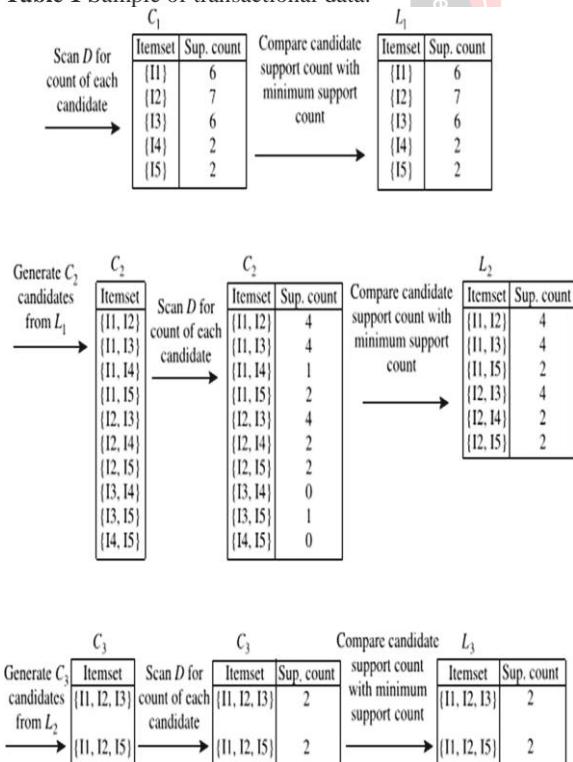


Fig. 1 Generation of candidate itemsets and frequent itemsets.

First, scanned the database to identify all the 1-itemsets by counting each of them and taking those as frequent 1-itemset that satisfy the minimum support threshold

which was given by the user. The description of each frequent itemset needs to scan the entire database until no more frequent  $k$ -itemsets is possible to be known. According to Fig. 1, the user specified minimum support threshold used is 2. Therefore, only the records that fulfill a minimum support count of 2 will be included into the next cycle of algorithm processing. The apriori algorithm of candidate itemsets reduces the size considerably and provides a noble performance. Though, it is still suffering from two acute limitations. In fig.1 first, a large number of candidate itemsets may still need to be generated if the total count of a frequent  $k$ -itemsets increases. Then, the entire database is required to be scanned repeatedly and a huge set of candidate items are required to be verified using the pattern matching technique.

#### B. APRIORITID ALGORITHM

AprioriTID[13] efforts to improve the performance of Apriori by avoiding multiple DB hits in the valuation process. “After the first pass, AprioriTID did not use the database for counting the support of candidate itemsets” [11]. “The process of candidate itemset generation is the same like the original Apriori algorithm. Alternatively data can also be presented in item-TID\_set format, where item is an item name and TID\_set is the set of transaction identifiers containing the item. This format is known as vertical data format. In the following table you can see the vertical data format of the example, shown in table 2.

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

#### Vertical data format

Itemset	TID_set
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T300, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

#### The 2-itemsets in vertical data format

Itemset	TID_set
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

#### The 3-itemsets in vertical data format

Table 2. Generation of itemset using Vertical data format

First we alter the horizontal form of data to the vertical data format by scanning once the data set. The support count of an itemset is the length of the TID\_set. Beginning

with  $k=1$  the frequent  $k$ -itemsets can be used to build the candidate  $(k+1)$ -itemsets. This procedure recaps with  $k$  incremented by 1 all time until no frequent Itemsets can be found. Benefits of this algorithm: Enhanced than Apriori in the generation of candidate  $(k+1)$ -itemset from frequent  $k$ -itemsets. There is no need to scan the database to find the support count of  $(k+1)$  itemsets (for  $k \geq 1$ ). This is because the TID\_set of each  $k$ -itemset carries the complete information required for counting such support. The difficulty of this algorithm involve in the TID\_set being extended, taking sizeable memory space as well as computation time for intersecting the long sets.

#### C. APRIORI\_RARE ALGORITHM

The objective of this algorithm is to generate the frequent as well as rare itemsets. It is the modification of Apriori algorithm to generate frequent and rare itemsets. It uses a Supportcount(sub-routine) to find the support count of a given itemset. The advantage of this algorithm is that it restores all the minimal rare itemsets. However, Apriori\_Rare fails to find all the rare itemsets.

---

#### *Algorithm : Apriori\_Rare*

---

```

 $C_1 \leftarrow$  All 1-itemsets
 $k \leftarrow 1$ 
while ( $C_k$  not Null) do
    Supportcount( $C_k$ )
     $R_k =$  Rare items (Supportcount < Minsup)
     $F_k =$  Frequent items (Supportcount > Minsup)
     $C_{k+1} =$  AprioriGen( $F_k$ )
     $k = k+1$ 
End of While
end
 $F = R_k$ 

```

---

Support count method: counts the support of candidate itemsets.

Apriori\_Gen function: Generates the candidates and then uses the Apriori property(all non\_empty subsets of a frequent itemset must also be frequent) to eliminate those having a subset that is not frequent.

If the support of a candidate is less than the minsup(minimumsupport),then instead of pruning it we will save it in the rare itemset( $R_k$ ).

#### D. APRIORIRARE\_TID ALGORITHM

The objective of this algorithm is to generate the rare itemsets. There is an alternative implementation of AprioriRare is called "AprioriRare\_TID". This implementation is based on AprioriTID instead of the standard Apriori algorithm. The key difference is that the identifiers of transactions where patterns are found are

kept in memory to avoid scanning the database. This can be faster on some datasets.

#### IV. RESULTS AND DISCUSSION

Various apriori based existing algorithm used to discover frequent and rare itemset mining techniques have been studied and reported in this paper. A general comparison among these techniques also has been reported in table.3. To address the limitations of these techniques, an effective Apriori based frequent and rare itemset finding technique has been presented. In this section we present the results of tests. First, we provide results that we obtained on a real-life supermarket dataset. In Fig.3 that shows the efficiency of memory usage of Frequent and Rare algorithms for supermarket dataset. The efficiency of memory usage of algorithms shown in Fig.4. Then, we demonstrate that which approach is computationally efficient for discover frequent and rare itemsets. Thus, a series of computational times and memory consumption resulting from the application of existing algorithms to well-known datasets is presented. In fig (3&4),Apriori algorithm is efficient for frequent itemset while comparing to Apriori and AprioriTID the both maximum memory and run time is less for the supermarket dataset and for Rare itemset AprioriRare\_TID is efficient while comparing to AprioriRare and AprioriRare\_TID maximum memory in system occupied is less. All the experiments were carried out on an Intel CORE i5 machine running under windows10 operating system with 8GB RAM. Algorithms are implemented in the java platform.

Table3 Efficiency of Frequent and Rare itemsets Algorithms for time consumption and memory usage.

Efficiency	FREQUENT ITEMSET MINING ALGORITHMS		RARE ITEMSET MINING ALGORITHMS	
	APRIORI TID	APRIORI	APRIORI RARE	APRIORIRARE TID
No.of itemset count	34	34	256	276
max_mem ory(mb)	37	28	39	11
total_time(ms)	207	76	79	115

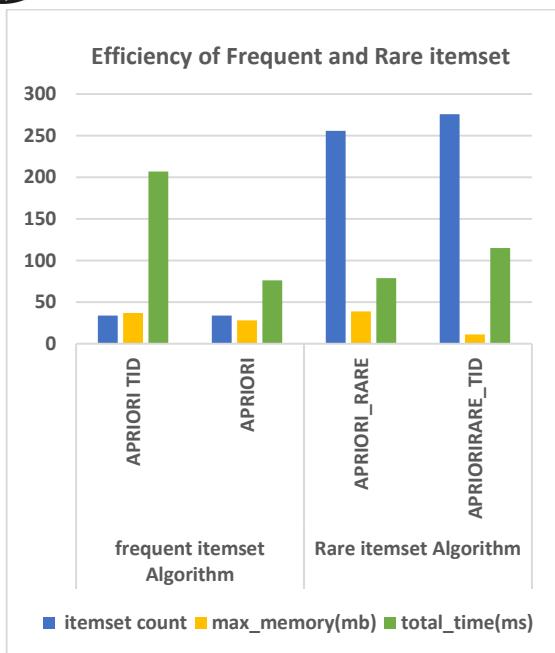


Fig. 2 Comparing Efficiency of Frequent and Rare itemsets Algorithms.

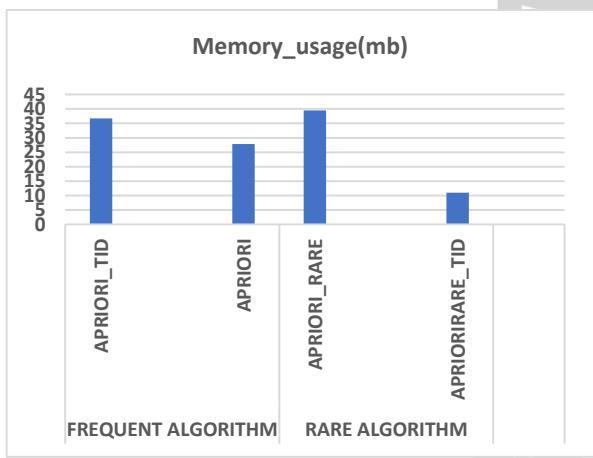


Fig. 3 Memory usage of existing Frequent and Rare algorithms for supermarket dataset.

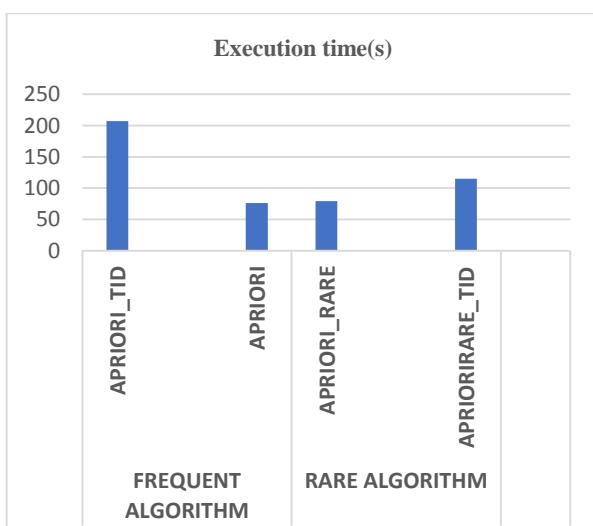


Fig. 4 Execution time of usage of existing Frequent and Rare algorithms for supermarket dataset.

## V CONCLUSION

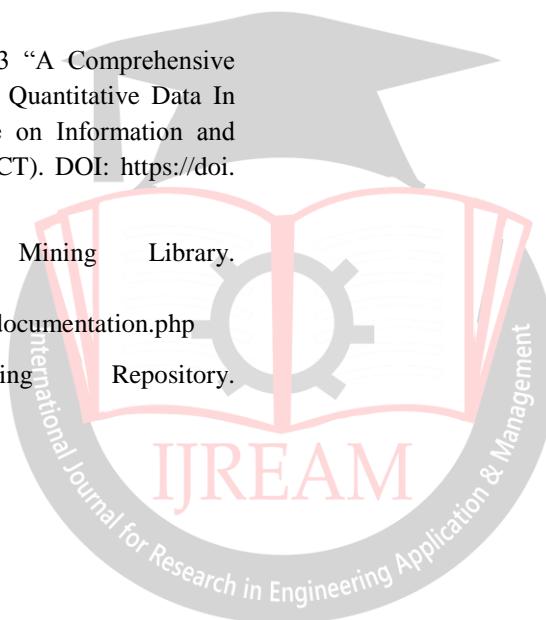
Itemset mining is an active field of research having several applications. This paper has presented the problem of frequent and rare itemset mining, discussed the main techniques for exploring the search space of itemsets and time consumption, employed by itemset mining algorithms. In summary, Apriori algorithm is efficient method for large dataset to discover frequent and rare itemset while comparing to Vertical and Horizontal data format. Experimental results show in Fig.2 that the Apriori and AprioriTID algorithms in terms of efficiency. Evaluation of Apriori, AprioriRare, AprioriTID and AprioriRare\_TID algorithms ensures that it is able to mine a transaction data set within a shorter run time with less memory consumption, so customers will be able to decrease the time of decision creating, improve the performance and action, and increase the profit of their organizations.

## REFERENCES

- [1] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers, Burlington (2010). ISBN 1-55860-901-6.
- [2] Arpan Shah et al. (2014), 'A Collaborative Approach of Frequent Item Set Mining: A Survey', International Journal of Computer Applications, Vol 107, No 8 <https://doi.org/10.5120/18775-0088>
- [3] Agrawal, R., Imielinski, T., Swami,A.N.:Mining association rules between sets of items in large databases. In: ACM SIGMOD International Conference on Management of Data, Washington, D.C. (1993).
- [4] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Inkeri Verkamo, A.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAIPress, Menlo Park (1996).
- [5] Manisha Kundal & Dr Parminder Kaur (2015), 'Various Frequent itemset based on Data Mining Technique', International Research Journal of Engineering and Technology, Vol 02, No 3 .
- [6] Ghosh S. et al. (2010), 'Mining frequent item sets using Genetic Algorithm', International Journal of Artificial Intelligence and Applications, Vol 1, No 4.
- [7] Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Patras, Greece, vol. 1, pp. 305–312, October 2007.
- [8] Sharma A. et al. (2012), 'A Survey of Association Rule Mining Using Genetic Algorithm', International

Journal of Computer Applications and Information Technology, Vol 1, No 2.

- [9] Brin S. et al. 'Beyond Market Baskets: Generalizing Association Rules to Correlations', in Proceedings of the ACM SIGMOD Conference, pp. 265-276 <https://doi.org/10.1145/253260.253327>.
- [10] Park J. et al. (1995) , 'Effective Hash-Based Algorithm for Mining Association', Proceedings of ACM SIGMOD International Conference on Management of Data, San Jose, CA, pp. 175 – 186 <https://doi.org/10.1145/568271.223813> .
- [11] Kumbhare, et al 2014 "An Overview of Association Rule Mining Algorithms" International Journal of Computer Science and Information Technologies (IJCSIT), Vol (5).
- [12] [https://www.packtpub.com/mapt/book/big\\_data\\_and\\_business\\_intelligence/9781784396589/5/ch05lvlsec33/the-supermarket-dataset](https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781784396589/5/ch05lvlsec33/the-supermarket-dataset).
- [13] Gosain, A and Bhugra, M 2013 "A Comprehensive Survey of Association Rules on Quantitative Data In Data Mining" IEEE Conference on Information and Communication Technologies (ICT). DOI: <https://doi.org/10.1109/cict.2013.6558244>.
- [14] An Open-Source Data Mining Library. <http://www.philippe-fournier-viger.com/spmf/index.php?link=documentation.php>
- [15] UCI Machine Learning <http://archive.ics.uci.edu/ml>.



See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/296873041>

# A Compressed Vertical Binary Algorithm for Mining Frequent Patterns.

Chapter · November 2008

---

CITATIONS

0

READS

336

4 authors, including:



José Hernández-Palancar

Advanced Technologies Applications Center, (CENATAV), Cuba

75 PUBLICATIONS 244 CITATIONS

[SEE PROFILE](#)



Raudel Hernández-León

Centro de Aplicaciones de Tecnologías de Avanzada

44 PUBLICATIONS 91 CITATIONS

[SEE PROFILE](#)



José E. Medina Pagola

University of Information Sciences

64 PUBLICATIONS 338 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



High resolution palmprint feature extraction and matching [View project](#)



Data Mining [View project](#)

---

# A Compressed Vertical Binary Algorithm for Mining Frequent Patterns

J. Hdez. Palancar<sup>1</sup>, R. Hdez. León<sup>1</sup>, J. Medina Pagola<sup>1</sup>, and A. Hechavarria<sup>1</sup>

Advanced Technologies Application Center (CENATAV), 7a # 21812 e/ 218 y 222,  
Rpto. Siboney, Playa, C.P. 12200, Ciudad de la Habana, Cuba.  
`{jpalancar,rhernandez,jmedina,ahechavarria}@cenatav.co.cu`

## 1 Introduction

Mining association rules in transaction databases have been demonstrated to be useful and technically feasible in several application areas [14], [18], [21] particularly in retail sales, and it becomes every day more important in applications that use document databases [11], [16], [17]. Although research in this area has been going on for more than one decade; today, mining such rules is still one of the most popular methods in knowledge discovery and data mining.

Various algorithms have been proposed to discover large itemsets [2], [3], [6], [9], [11], [19]. Of all of them, Apriori has had the biggest impact [3], since its general conception has been the base for the development of new algorithms to discover association rules.

Most of the previous algorithms adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there are many items, when the quantity of items by transaction is high, or the minimum support threshold is quite low. These algorithms need to scan the database several times to check the support of each candidate, and it is a time-consuming task for very sparse and huge databases. The weak points of the Apriori algorithm are these aspects: the candidate generation and the count of each candidate support.

Performance of the algorithm could be significantly improved if we find a way to reduce the computational cost of the tasks above mentioned.

Although in [3] the way in which the transactions are represented is not mentioned by the authors, this aspect influences decisively in the algorithm. In fact, it has been one of the elements used by other authors, including us, in the formulation of new algorithms [4], [6], [7], [9], [19].

We face the problem in the following way:

- Other authors represent the transaction database as sorted lists (or array-based), BTree, Trie, etc., using items that appear in each transaction;

others use horizontal or vertical binary representations. We will use a compressed vertical binary representation of the database.

- The efficiency count of each candidate's support in this representation can be improved using logical operations, which are much faster than working with non-compact forms.

A new algorithm suitable for mining association rules in databases is proposed in this paper; this algorithm is named as CBMine (Compressed Binary Mine).

The discovery of large itemsets (the first step of the process) is computationally expensive. The generation of association rules (the second step) is the easiest of both. The overall performance of mining association rules depends on the first step; for this reason, the comparative effects of the results that we present with our algorithm covers only the first step.

In the next section we give formal definitions about association rules and frequent itemsets. The third section is dedicated to related work. The fourth section contains the description of CBMine algorithm. The experimental results are discussed in the fifth section.

The new algorithm shows significantly better performance than several algorithms, like Bodons Apriori algorithms, and in sparse databases than MAFIA, and in a general way it is applicable to those algorithms with an Apriori-like approach.

## 2 Preliminaries

Let be  $I = \{i_1, i_2, \dots, i_n\}$  a set of elements, called items (we prefer to use the term elements instead of literals [2], [3]). Let  $D$  be a set of transactions, where each transaction  $T$  is a set of items, so that  $T \subseteq I$ . An association rule is an implication of the form  $X \Rightarrow Y$ , where  $X \subset I$ ,  $Y \subset I$ , and  $X \cap Y = \emptyset$ . The association rule  $X \Rightarrow Y$  holds in the database  $D$  with certain quality and a support  $s$ , where  $s$  is the proportion of transactions in  $D$  that contain  $X \cup Y$ . Some quality measures have been proposed, although these are not considered in this work.

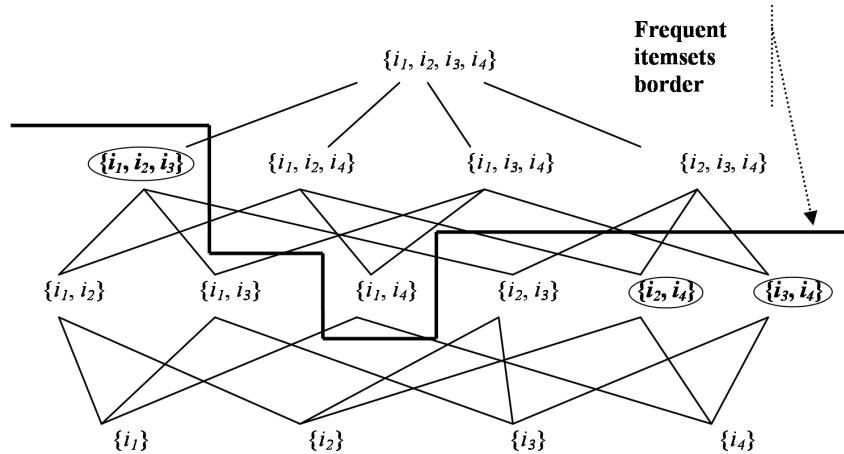
Given a set of transactions  $D$ , the problem of mining association rules is to find all association rules that have a support greater than or equal to the user-specified minimum (called *minsup*) [3]. For example, beer and disposable diapers are items so that  $\text{beer} \Rightarrow \text{diaper}$  is an association rule mined from the database if the co-occurrence rate of beer and disposable diapers (in the same transaction) is not less than *minsup*.

The first step in the discovery of association rules is to find each set of items (called *itemset*) that has co-occurrence rate above the minimum support. An itemset with at least the minimum support is called a *large* itemset or a *frequent* itemset. In this paper, as in others, the term frequent itemset will be used. The size of an itemset represents the number of items contained in the

itemset, and an itemset containing  $k$  items is called a  $k$ -itemset. For example, beer, diaper can be a frequent 2-itemset. If an itemset is frequent and no proper superset is frequent, we say that it is a *maximally* frequent itemset.

Finding all frequent itemsets has received a considerable amount of research effort in all these years because it is a very resource-consuming task. For example, if there is a frequent itemset with size  $l$ , then all  $2^l - 1$  nonempty subsets of the itemset have to be generated.

The set of all subsets of  $I$  (the powerset of  $I$ ) naturally forms a lattice, called the *itemset lattice* [10], [22]. For example, consider the lattice of subsets of  $I = \{i_1, i_2, i_3, i_4\}$ , shown in Fig. 1 (the empty set has been omitted). Each maximal frequent itemset of the figure is in bold face and in an ellipse.



**Fig. 1.** Lattice of subsets of  $I = \{i_1, i_2, i_3, i_4\}$

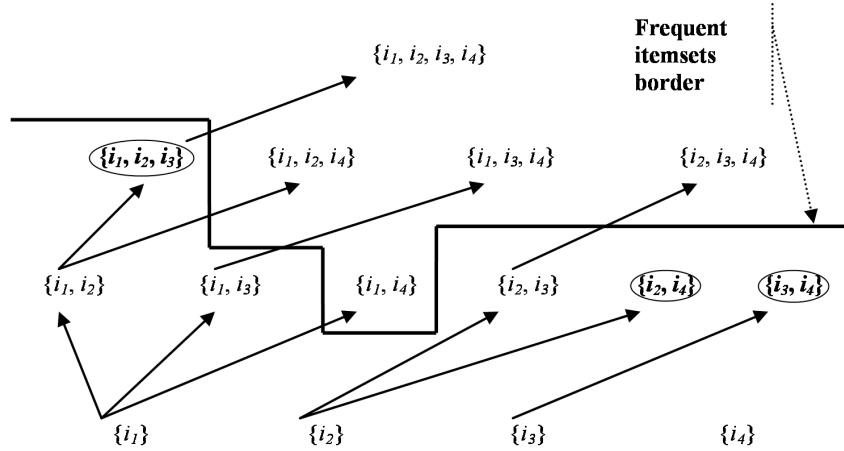
Due to the *downward closure* property of itemset support - meaning that any subset of a frequent itemset is frequent - there is a border, so that all frequent itemsets lie below the border, while all infrequent itemsets lie above it. The border of frequent itemsets is shown with a bold line in Fig. 1.

An optimal association mining algorithm must only evaluate the frequent itemsets traversing the lattice in some way. This one can be done considering an equivalence class approach. The equivalence class of an itemset  $a$ , expressed as  $E(a)$ , is given as:

$$E(a) = \{b : |a| = k, |b| = k, \text{Prefix}_{k-1}(b) = \text{Prefix}_{k-1}(a)\}, \quad (1)$$

where  $\text{Prefix}_k(c)$  is the prefix of size  $k$  of  $c$ , i.e., its  $k$  first items in a lexicographical order.

Assuming equivalence classes, the itemset lattice of Fig. 1 can be structured in a forest, shown in Fig. 2, clustering itemsets of same equivalence classes.



**Fig. 2.** Search forest of subsets of  $I = \{i_1, i_2, i_3, i_4\}$

In order to traverse the itemset space, a convenient strategy should be chosen. Today's common approaches employ either breath-first search or depth-first search. In a breadth strategy the support values of all  $(k-1)$ -itemsets are determined before counting the support values of  $k$ -itemsets, and in a depth one recursively descends following the forest structure defined through equivalence classes [10].

The way itemsets are represented is decisive to compute their supports. Conceptually, a database is a two-dimensional matrix where the rows represent the transactions and the columns represent the items. This matrix can be implemented in the following four different formats [20]:

- Horizontal item-list (**HIL**): The database is represented as a set of transactions, storing each transaction as a list of item identifiers (item-list).
- Horizontal item-vector (**HIV**): The database is represented as a set of transactions, but each transaction is stored as a bit-vector (item-vector) of 1's and 0's to express the presence or absence of the items in the transaction.
- Vertical tid-list (**VTL**): The database is organized as a set of columns with each column storing an ordered list (tid-list) of only the transaction identifiers (TID) of the transactions in which the item exists.
- Vertical tid-vector (**VTV**): This is similar to VTL, except that each column is stored as a bit-vector (tid-vector) of 1's and 0's to express the presence or absence of the items in the transactions.

Many association rule mining algorithms have opted for a list-based (horizontally or vertically) layout since, in general, this format takes less space than the bit-vector approach. In other way, it could be noticed that computing the

TID	Item-lists	Item-vectors
1	1 2 3 5	1 1 1 0 1
2	2 3 4 5	0 1 1 1 1
3	3 4 5	0 0 1 1 1
4	1 2 3 4 5	1 1 1 1 1

Tid-lists					Tid-vectors				
1	2	3	4	5	1	2	3	4	5
1	1	1	2	1	1	1	1	0	1
4	2	2	3	2	0	1	1	1	1
	4	3	4	3	0	0	1	1	1
		4	4	4	1	1	1	1	1

**Fig. 3.** Examples of database layouts

supports of itemsets is simpler and faster with the vertical layout (VTL or VTV) since it involves only the intersections of tid-lists or tid-vectors.

In a general point of view, rule mining algorithms employ a combination of a traverse strategy (breadth-first or depth-first) and a form of the database layout. Examples of algorithms for horizontal mining with a breadth strategy previously presented are Apriori, AprioriTID, DIC [3] and with a depth strategy are different version applying FP-Trees [1]. Other algorithms considering a VTL layout are Partition, with a breadth strategy [10], and Eclat, with a depth strategy [22].

In this paper we evaluate a compressed form of the VTV layout, improving the performance of the itemset generation, applicable to those algorithms with an Apriori-like approach.

The problem of mining frequent itemsets was first introduced by Agrawal *et al.* [2]. To achieve efficient mining frequent patterns, an antimonotonic property of frequent itemsets, called the Apriori heuristic, was formulated in [3]. The Apriori heuristic can dramatically prune candidate itemsets.

Apriori is a breadth-first search algorithm, with a HIL organization, that iteratively generates two kinds of sets:  $C_k$  and  $L_k$ . The set  $L_k$  contains the large itemsets of size  $k$  ( $k$ -itemsets). Meanwhile,  $C_k$  is the set of candidate  $k$ -itemsets, representing a superset of  $L_k$ . This process continues until a null set  $L_k$  is generated.

The set  $L_k$  is obtained scanning the database and determining the support for each candidate  $k$ -itemset in  $C_k$ . The set  $C_k$  is generated from  $L_{k-1}$  with the following procedure:

$$C_k = \{c | Join(c, L_{k-1}) \wedge Prune(c, L_{k-1})\} \quad (2)$$

where:

$$Join(\{i_1, \dots, i_k\}, L_{k-1}) \equiv \langle \{i_1, \dots, i_{k-2}, i_{k-1}\} \in L_{k-1} \wedge \{i_1, \dots, i_{k-2}, i_k\} \in L_{k-1} \rangle, \quad (3)$$

$$Prune(c, L_{k-1}) \equiv \langle \forall s[s \subset c] \wedge |s| = k-1 \rightarrow s \in L_{k-1} \rangle. \quad (4)$$

Observe that the Join step (3) takes two ( $k-1$ )-itemsets of a same equivalence class to generate a  $k$ -itemsets.

The Apriori algorithm is presented in Fig. 4.

---

**Algorithm:** Apriori

---

**Input:** Database

**Output:** Large itemsets

---

```

1)  $L_1 = \{\text{large 1-itemsets}\};$ 
2) for ( $k = 2; L_{k-1} \neq \emptyset; k++$ ) do begin
3)    $C_k = \text{apriori\_gen}(L_{k-1});$  // New candidates
4)   forall transaction  $t \in D$  do begin
5)      $C_t = \text{subset}(C_k, t);$  // Candidates in  $t$ 
6)     forall candidate  $c \in C_t$  do
7)        $c.\text{count}++;$ 
8)     end
9)    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\};$ 
10)  end
11) end
12) Answer =  $\cup_k L_k;$ 
```

---

**Fig. 4.** Pseudo code of Apriori Algorithm

The procedure `apriori_gen` used in step 3 is described in (2).

### 3 Related work

The vertical binary representations (VTV) and the corresponding support counting method have been investigated by other researchers [7], [8], [10], [12], [22].

Zaki *et al.* proposed several algorithms using vertical binary representations in 1997 [22]. Their improvements are obtained clustering the database and applying an Apriori-like method with simple tid-vectors.

Gardarin *et al.* proposed two breadth-first search algorithms using vertical binary representations named N-BM and H-BM in 1998 [8]. N-BM considers simple (uncompressed) vertical binary representations for itemsets. Meanwhile, H-BM uses, besides, an auxiliary bit-vector, where each bit represents a group of bits of the original bit-vector. In order to save the memory, every 1-itemset has both, while every large itemset keeps only the auxiliary bit-vector. H-BM first performs the AND between auxiliary bit-vectors and only non-zero

groups are considered to the final count. However, as in large itemsets only auxiliary bit-vectors are stored; the bit values of the items considered in the itemset need to be checked.

Burdick *et al.* proposed a depth-first search algorithm using vertical binary representation named MAFIA in 2001 [7]. They use bit-vectors, and it is an efficient algorithm; but only for finding maximal frequent itemsets, and especially in dense databases. Mining only maximal frequent itemsets has the following deficiency: From them we know that all its subsets are frequent, but we do not know the exact value of the supports of these subsets. Therefore, we can not obtain all the possible association rules from them.

Shenoy *et al.* proposed another breath-first search algorithm, called VIPER, using vertical representations. Although VIPER used a compressed binary representation on disk, when these compressed vectors are processed in memory they are converted right away into tid-lists, not considering advantages of boolean operations over binary formats [20].

Many other researchers have proposed other vertical binary algorithms, although the above mentioned are to the best of our knowledge the most representatives.

The method we present in this paper, CBMine, obtains all frequent itemsets faster than these well-known Apriori and vertical binary implementations, outperforming them considerably, especially for sparse databases.

## 4 CBMine algorithm

A new method applied to Apriori-like algorithms, named CBMine (Compressed Binary Mine), is analyzed in this section.

### 4.1 Storing the transactions

Let us call the itemset that is obtained by removing infrequent items from a transaction the filtered transaction. The size of the filtered transactions is declared to be "substantially smaller than the size of database". Besides, all frequent itemsets can be determined even if only filtered transactions are available.

The set of filtered transactions can be represented as an  $m \times n$  matrix where  $m$  is the number of transactions and  $n$  is the number of frequent items (see Fig. 5 for an  $8 \times 5$  matrix). We can denote the presence or absence of an item in each transaction by a binary value (1 if it is present, else 0).

This representation has been considered as a logical view of the data. Nevertheless, some researchers have employed it for counting the support for an item and for generating the set of 1-frequent itemsets [15].

To reduce I/O cost and speed up the algorithm, the filtered transactions could be stored in main memory instead of on disk. Although this is a reasonable solution, any data structure could require a considerable - and probably a prohibitive - memory space for large databases.

Tid	Items
1	1 2 3 5
2	2 3 4 5
3	3 4 5
4	1 2 3 4 5
5	4 5
6	2 3 4
7	2 4 5
8	1 2 4 5

1 2 3 4 5
1 1 1 0 1
0 1 1 1 1
0 0 1 1 1
1 1 1 1 1
0 0 0 1 1
0 1 1 1 0
0 1 0 1 1
1 1 0 1 1

Fig. 5. Horizontal layout of the database

1 2 3 4 5
1 1 1 0 1
0 1 1 1 1
0 0 1 1 1
1 1 1 1 1
0 0 0 1 1
0 1 1 1 0
0 1 0 1 1
1 1 0 1 1

Item	Tid-vector	Array
1	1 0 0 1 0 0 0 1	{0x91}
2	1 1 0 1 0 1 1 1	{0xD7}
3	1 1 1 1 0 1 0 0	{0xF4}
4	0 1 1 1 1 1 1 1	{0x7F}
5	1 1 1 1 1 0 1 1	{0xFB}

Fig. 6. Vertical binary representation of a transaction database (word size = 8)

Considering the standard binary representation of the filtered transactions, we propose to represent these transactions vertically and store them in main memory as an array of integer numbers (a VTV organization). It should be noticed that these numbers are not defined by row but by column (see Fig. 6). The reasons for this orientation will be explained later on.

If the maximum number of transactions were not greater than a word size, the database could be stored as a simple set of integers; however, a database is normally much greater than a word size, and in many cases very much greater. For that reason, we propose to use a list of words (or integers) to store each filtered item.

Let  $T$  be the binary representation of a database, with  $n$  filtered items and  $m$  transactions. Taking from  $T$  the columns associated to frequent items, each item  $j$  can be represented as a list  $I_j$  of integers (integer-list) of word size  $w$ , as follows:

$$I_j = \{W_{1,j}, \dots, W_{q,j}\}, q = \lceil m/w \rceil, \quad (5)$$

where each integer of the list can be defined as:

$$W_{s,j} = \sum_{r=1}^{\min(w, m - (s-1)*w)} 2^{(w-r)} * t_{((s-1)*w+r),j}. \quad (6)$$

The upper expression  $\min(w, m - (s-1)*w)$  is included to consider the case in which the transaction number  $(s-1)*w + r$  doesn't exist due to the fact that it is greater than  $m$ .

This binary representation for items, as noted Burdick *et al.*, naturally extends to itemsets [7]. Suppose we have an integer-list  $A_X$  for an itemset  $X$ , the integer-list  $A_{X \cup \{j\}}$  is simply the bitwise *AND* of the integer-lists  $A_X$  and  $I_j$ . If an itemset has a single item then its integer-list is  $I_j$ .

The weakness of the vertical binary representation is the memory spending, especially in sparse databases. An alternative representation is considering only non null integers. This compressed integer-lists could be represented as an array  $CA$  of pairs  $\langle s, B_s \rangle$  with  $1 \leq s \leq q$  and  $B_s \neq 0$ .

## 4.2 Algorithm

CBMine is a breadth-first search algorithm with a VTV organization, considering compressed integer-lists for itemset representation.

This algorithm iteratively generates a prefix list  $PL_k$ . The elements of this list have the format:  $\langle Prefix_{k-1}, CA_{Prefix_{k-1}}, Suffixes_{Prefix_{k-1}} \rangle$ , where  $Prefix_{k-1}$  is a  $(k-1)$ -itemset,  $CA_{Prefix_{k-1}}$  is the corresponding compressed integer-list, and  $Suffixes_{Prefix_{k-1}}$  is the set of all suffix items  $j$  of  $k$ -itemsets extended with the same  $Prefix_{k-1}$ , where  $j$  is lexicographically greater than every item in the prefix and the  $k$ -itemsets extended are frequent. This representation not only reduces the required memory space to store the integer-lists but also eliminates the Join step described in (3).

CBMine guarantees a significant memory reduction. Other algorithms with VTV representation have an integer-list for each itemset, meanwhile CBMine has an integer-list only for each equivalent class (see Fig. 7).

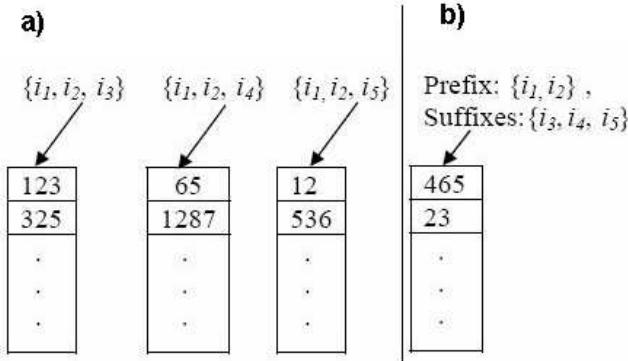
The Prune step (4) is optimized generating  $PL_k$  as a sorted list by the prefix field and, for each element, by the suffix field.

In order to determine the support of an itemset with a compressed integer-list  $CA$ , the following expression is considered:

$$Support(CA) = \sum_{\langle s, B_s \rangle \in CA} BitCount(B_s), \quad (7)$$

where  $BitCount(B_s)$  represents a function that calculates the Hamming Weight of each  $B_s$ .

Although this algorithm uses compressed integer-lists of non null integers ( $CA$ ) for itemset representation, in order to improve the efficiency, we maintain the initial integer-lists (including the null integers)  $I_j = \{W_{1,j}, \dots, W_{q,j}\}$  associated with each large 1-itemset  $j$ . This consideration allows directly accessing in  $I_j$  the integer position defined in  $CA$ .



**Fig. 7.** a) others algorithms with VTV representations, b) CBMine

The above consideration allows defining the formulae (8). Notice that this function represents a significant difference and improvement respect other methods.

$$CompAnd(CA, I_j) = \{ < s, B'_s > : < s, B_s > \in CA, \\ B'_s = B_s \text{ and } W_{s,j}, B'_s \neq 0 \}. \quad (8)$$

It could also be noticed that the cardinality of  $CA$  is reduced with the increment of the size of the itemsets due to the downward closure property. It allows the improvement of the above processes (7) and (8).

The CBMine algorithm is presented in Fig. 8.

The step 2 of the pseudo code shown in Fig. 8 (the process for  $k = 2$ ) is performed in a similar way to that for  $k \geq 3$ , except the Prune procedure because it is unnecessary in this case. This procedure Prune, used in step 9, is similar to (4). Notice that this algorithm only scans the database once in the first step.

## 5 Experimental results

Here we present the experimental results of an implementation of the CBMine algorithm. It was compared with the performance of two Apriori implementations made by Ferenc Bodon (Simple and Optimized algorithms) [5], and MAFIA [7].

Four known databases were used: T40I10D100K, T10I4D100K, generated from the IBM Almaden Quest research group, Chess and Pumsb\*, prepared by Roberto Bayardo from the UCI datasets and PUMSB (see Table 1).

Our tests were performed on a PC with a 2.66 GHz Intel P4 processor and 512 Mbytes of RAM. The operating system was Windows XP. Running times were obtained using standard C functions. In this paper, the runtime includes both CPU time and I/O time.

---

**Algorithm:** CBMine

---

**Input:** Database**Output:** Large itemsets

```
1)  $L_1 = \{\text{large 1-itemsets}\}; // \text{Scanning the database}$ 
2)  $PL_1 = \{\langle \text{Prefix}_1, CA_{\text{Prefix}_1}, Suffixes_{\text{Prefix}_1} \rangle\};$ 
3) for ( $k = 3; PL_{k-1} \neq \emptyset; k++$ ) do
4)   forall  $\langle \text{Prefix}, CA, Suffixes \rangle \in PL_{k-1}$  do
5)     forall item  $j \in Suffixes$  do begin
6)        $\text{Prefix}' = \text{Prefix} \cup \{j\};$ 
7)        $CA' = \text{CompAnd}(CA, I_j);$ 
8)       forall  $(j' \in Suffixes) \&& (j' > j)$  do
9)         if  $\text{Prune}(\text{Prefix}' \cup \{j'\}, PL_{k-1}) \&&$ 
10)             $\text{Support}(\text{CompAnd}(CA', I_{j'})) \geq \text{minsup}$ 
11)         then  $Suffixes' = Suffixes' \cup \{j'\};$ 
12)       if  $Suffixes' \neq \emptyset$ 
13)         then  $PL_k = PL_k \cup \{\langle \text{Prefix}', CA', Suffixes' \rangle\};$ 
14)       end
15)   Answer =  $\cup_k L_k; // L_k \text{ is obtained from } PL_k$ 
```

---

**Fig. 8.** Pseudo code of CBMine Algorithm**Table 1.** Database characteristics

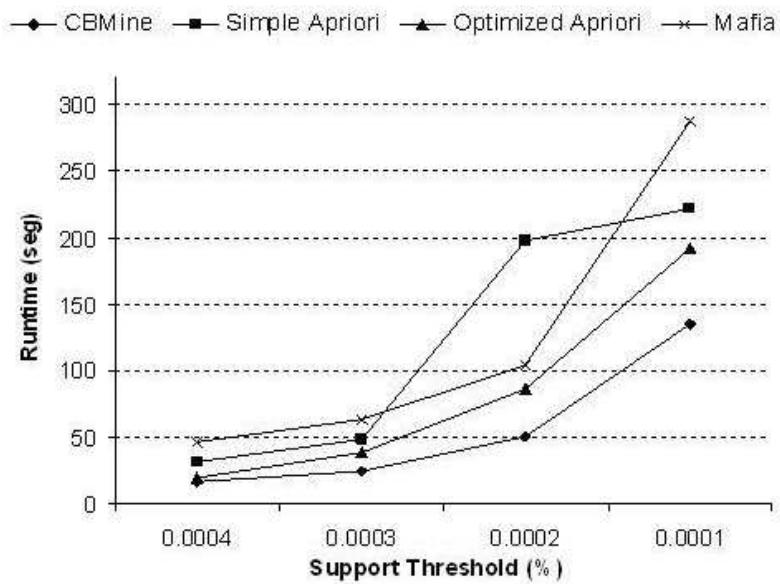
	T10I4D100K	T40I10D100K	Pumsb*	Chess
AvgTS	10.1	39.54	50	37
MaxItems	870	942	2.087	75
Transactions	100000	100000	49046	3196

Table 2 and the following graphics present the test results of the Apriori implementations, MAFIA and CBMine with these databases. Each test was carried out 3 times; the tables and graphics show the averages of the results. Bodon's Apriori implementations were versions on April 26th, 2006 [5]. MAFIA implementation was a version 1.4 on February 13th, 2005; it was run with "-fi" option in order to obtain all the frequent itemsets [13].

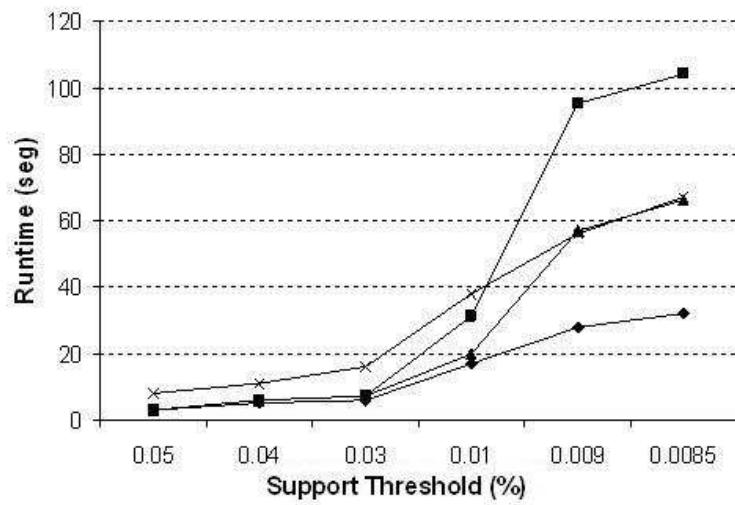
CBMine beats the other implementations in almost all the times. It performs best results independently of the support threshold in sparse databases (T40I10D100K and T10I4D100K). Nevertheless, we have verified that MAFIA beats CBMine for low thresholds in less sparse databases (Chess and Pumsb\*).

**Table 2.** Performance results (in seconds)

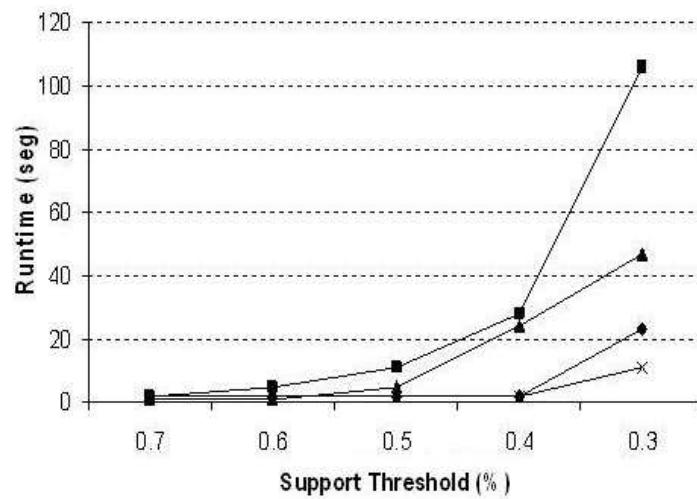
databases	<i>minsup</i>	CBMine	Simple-Apriori	Optimized-Apriori	MAFIA
T10I4D100K	0.0004	17	32	20	52
	0.0003	25	49	39	64
	0.0002	51	198	86	104
	0.0001	135	222	192	287
T40I10D100K	0.0500	3	3	3	8
	0.0400	5	6	6	11
	0.0300	6	7	7	16
	0.0100	17	31	20	38
	0.0090	28	95	57	56
	0.0085	32	104	66	67
Pumsb*	0.7	2	2	1	2
	0.6	2	5	1	2
	0.5	2	11	5	2
	0.4	2	28	24	2
	0.3	23	106	47	11
Chess	0.9	0	3	2	0
	0.8	0	8	2	0
	0.7	1	45	3	1
	0.6	2	92	22	2
	0.5	16	163	53	7



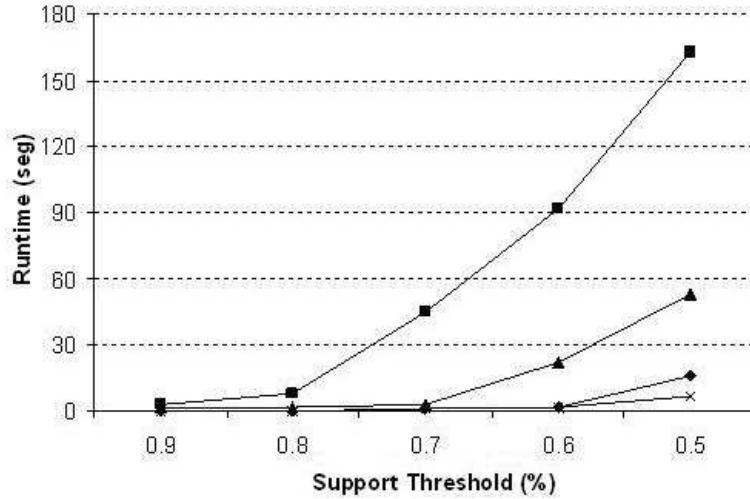
**Fig. 9.** T10I4D100K database



**Fig. 10.** T40I10D100K database



**Fig. 11.** Pumsb\* database



**Fig. 12.** Chess database

## 6 Conclusions

The discovery of frequent objects (itemsets, episodes, or sequential patterns) is one of the most important tasks in data mining. The ways databases and its candidates are stored cause a crucial effect on running times and memory requirements.

In this paper we have presented a compressed vertical binary approach for mining several kinds of databases. Our experimental results show that the inclusion of this representation in Apriori-like algorithms makes them more efficient and scalable.

We presented a method that obtains frequent itemset faster than other well-known Apriori implementations and vertical binary implementations, outperforming them considerably, especially for sparse databases.

There are many other issues to be analyzed using a vertical compressed binary approach. This is our goal, and these issues will be included in further papers.

## References

1. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1347–1362, 2005. Member-Gosta Grahne and Student Member-Jianfei Zhu.
2. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Ja-

- jodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
3. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.
  4. Ferenc Bodon. Surprising results of trie-based fim algorithms. In Bart Goethals, Mohammed J. Zaki, and Roberto Bayardo, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'04)*, volume 126 of *CEUR Workshop Proceedings*, Brighton, UK, 1. November 2004.
  5. Ferenc Bodon. Trie-based apriori implementation for mining frequent itemsequences. In Bart Goethals, Siegfried Nijssen, and Mohammed J. Zaki, editors, *Proceedings of ACM SIGKDD International Workshop on Open Source Data Mining (OSDM'05)*, pages 56–65, Chicago, IL, USA, August 2005.
  6. Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13–15, 1997, Tucson, Arizona, USA*, pages 255–264. ACM Press, 05 1997.
  7. Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 443–452, Washington, DC, USA, 2001. IEEE Computer Society.
  8. G. Gardarin, P. Pucheral, and F. Wu. Bitmap based algorithms for mining association rules, 1998.
  9. Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.
  10. Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.
  11. John D. Holt and Soon M. Chung. Multipass algorithms for mining association rules in text databases. *Knowl. Inf. Syst.*, 3(2):168–183, 2001.
  12. Tsau Young Lin. Data mining and machine oriented modeling: A granular computing approach. *Appl. Intell.*, 13(2):113–124, 2000.
  13. Calimlim M. and Gehrke J. Himalaya data mining tools: Mafia. <http://himalaya-tools.sourceforge.net>, May 2006.
  14. Fayyad U. M., Piatetsky-Shapiro G., and Smyth P. From data mining to knowledge discovery: An overview. In Fayyad U. M., Piatetsky-Shapiro G., Smyth P., and Uthurusamy R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, 1996.
  15. Gopalan R. P. and Sucayho Y. G. High performance frequent patterns extraction using compressed fp-tree. In *Proceedings of the SIAM International Workshop on High Performance and Distributed Mining*, Orlando, USA, 2004.
  16. Feldman R. and Hirsh H. *Machine Learning and Data Mining: Methods and Applications*, chapter Finding Associations in Collections of Text, pages 223–240. John Wiley and Sons, 1998.
  17. Feldman R., Dagen I., and Hirsh H. Mining text using keyword distributions. *Journal of Intelligent Information Systems*, 10(3):281–300, 1998.

18. Chen M. S., Han J., and Yu P. S. Data mining: An overview from a data-base perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.
19. Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *The VLDB Journal*, pages 432–444, 1995.
20. Pradeep Shenoy, Jayant R. Haritsa, S. Sundarshan, Gaurav Bhalotia, Mayank Bawa, and Devavrat Shah. Turbo-charging vertical mining of large databases. pages 22–33, 2000.
21. Cheung D. W., Han J., Ng V. T., and Wong C Y. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the 12th IEEE International Conference on Data Engineering*, pages 106–114. IEEE, 1996.
22. Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. Technical Report TR651, 1997.

# Horizontal Format Data Mining with Extended Bitmaps

Buddhika De Alwis<sup>1</sup>, Supun Malinga<sup>2</sup>, Kathiravelu Pradeeban<sup>3</sup>, Denis Weerasiri<sup>4</sup>, Shehan Perera<sup>5</sup>

<sup>1,2,3,4</sup> WSO2 Inc., Sri Lanka.

<sup>5</sup> Department of Computer Science and Engineering,  
University of Moratuwa, Sri Lanka.

<sup>1</sup>buddhikac@wso2.com

<sup>2</sup>supunm@wso2.com

<sup>3</sup>pradeeban@wso2.com

<sup>4</sup>denis@wso2.com

<sup>5</sup>shehan@uom.lk

**Abstract:** Analysing the data warehouses to foresee the patterns of the transactions often needs high computational power and memory space due to the huge set of past history of the data transactions. With the fragmented data along with the current trend of distributed systems, most of the fundamental algorithms that are initially proposed to find the association among the itemsets in the data warehouses are inefficient either in throughput or the utilization of the resources.

Apriori algorithm is a mostly learned and implemented algorithm that mines the data warehouses to find the associations. However, Apriori is generally not an optimized algorithm. More variations, improvements, and alternatives have been suggested to overcome the inefficiency of Apriori algorithm, either as a whole or to specific sets of data. In any case, a fraction of improvement in the algorithm often improves the mining considerably. Frequent item set mining with vertical data format has been proposed as an improvement over the basic Apriori algorithm, which mines the data sets of vertical form, opposed to the typical horizontal format data as in case of Apriori.

In this paper we are proposing an algorithm as an alternative to Apriori algorithm, which will use bitmap indices in conjunction with a horizontal format data set converted to a vertical format data structure to mine frequent itemsets leveraging efficiencies of bitmap based operations and vertical format data orientation.

**Keywords:** Data mining, Association Rule, Apriori, Vertical format mining, Bitmap Indices, Data Analysis, Data Warehousing.

## I. Introduction

Data Mining is an emerging concept or a tool in database concepts, which is young, yet powerful and promising [1], [2]. Knowledge discovery, prediction, clustering, and classifications through pattern recognition are some of the key design aspects of data mining. Most of the existing techniques on the associations are based on analysing the data warehouses - the existing bulk data of transaction history. Given the existence of a set of items, association rules enable the prediction of the existence of one or more other items based on the knowledge gathered by classifying the data warehouses.

Foreseeing the user behaviour from the customer analysis of the past years is a topic of interest in this user-oriented marketing era. Database Engineers and scientists have been working on the associations of the transactions and derived many algorithms for effective decision making. Most of the existing techniques are based on analysing the data warehouses - the existing bulk data of transaction history [1].

The choice of the algorithm to retrieve the relationship between the variables for a given application is a challenging task, which is often a compromise where the accuracy, efficiency, latency, throughput, and security matter, as resources are limited. Several algorithms leading to optimal and sub-optimal conclusion have been developed and practiced on the datasets to extract patterns and gather the association among the variables or items. Given the existence of a set of items, association rules enable the prediction of the existence of one or more other items based on the knowledge gathered by classifying the data warehouses.

Association rule learning is mostly explained by one of its common applications – retrieving the association between the items that customers purchase. As an analogy, it is referred as keeping track of the customers' baskets or market basket analysis [1]. Association rules are commonly used in mining web usage [3], intrusion detection [4], and bioinformatics [5].

Statistical bias caused by suggesting a hypothesis by non-representative data or a narrow sample to match the hypothesis is defined as data-snooping bias, which often leads to a wrong decision in scientific calculations which include a highly distributed network [6]. This leads to a wrong decision in calculations, hence these factors also should be taken into consideration when choosing the algorithm.

Apriori algorithm [1] is considered the fundamental algorithm for mining for associations. Hybrid algorithms converging the classification and association rule mining have also been suggested [7]. With the explosive growth of the data base size, scalability for the data mining algorithms becomes crucial to be able to work with very large data sets effectively. In this paper, a hybrid frequent item-set mining method extending Apriori algorithm is proposed.

## II. Preliminaries

Traditionally, the algorithms of data analysis assumed that the input database contains a relatively smaller number of records. It becomes impossible to deploy the databases fully in the main memory, with the explosion of growth in their size. Extracting data from the hard drive is considerably slower than accessing the data located in memory. Hence the data mining algorithms should be scalable to be able to work with very large databases effectively. An algorithm is called 'scalable', if sustained capacity of main memory, with an increase in the number of records in the input database, its execution time increases linearly. Hence efficient scalable algorithms for data mining in very large data sets are widely studied.

### A. Market Basket Analysis

Finding frequent item sets plays an important role in data mining as the first step in determining association rules. Association rule learning is mostly explained by market basket analysis [1] – retrieving the association between the items that customers purchase. Here products or sets of items (item-sets) which occur in many transactions are found.

*Table 1.* Transactional data

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

Each of the patterns of behaviour of buyers identified through this analysis can be used to place the goods in the shops or restructure pages in the catalogues.

A set consists of i goods, called i-set-element (i-item-set). The percentage of transactions containing a given set, called the support (provision) set. It is believed that in order for a set of interest, its support must be above a user-defined minimum, such sets are called 'frequent'. Table 1 [8] describes several transactions (T100, T200, ..), stored in a relational database. Corresponding column mentions the relevant list of item ids for the particular transaction. As an example "T200" transaction contains "I2" and "I4" item ids.

In frequent item-set mining, we derive rules based on two measurements called minimum support and the minimum confidence that reflect the usefulness and certainty of the discovered rules. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold [1].

### B. Algorithm Apriori

This algorithm defines several stages. At the i-th stage identifies all frequent i-element sets. Each stage consists of two steps: the formation of candidates (candidate generation) and the counting of candidates (candidate counting). At the step of forming the candidates, algorithm creates a lot of candidates from the i-element sets. At the step of counting candidates algorithm scans the database of transactions, computing support sets of candidates. After scanning, the algorithm discards the candidates, ensuring that less than a user-defined minimum and saves only the common i-element sets.

During the 1<sup>st</sup> phase of the selected set of candidates contains all 1-element sets. Algorithm calculates their support during the step counting candidates. Thus, after the first phase all frequent 1-element sets are known. Reasoning in a straight line, a candidate can "burn" all pairs of goods. However, Apriori reduces the number of sets of candidate sifting - a priori - those candidates who may not be frequent, based on information received at previous stages of the information on which of the sets are the most abundant. Screenings are based on the simple assumption that if the set is frequent, all its subsets must also be frequent. Thus, before the counting of candidates step algorithm can reject any candidate set, a subset of which is not frequent. This process is continued until the number of frequent n-item-sets becomes zero, where n determines the no. of children in the item-set.

Consider the database presented in Table 1. Suppose that the minimum support count threshold is 2. That is, to be a frequent item-set, there should be at least two transactions, consisted of the particular item-set. In the first stage, all the products individually are sets of candidates and counted during the counting step, the candidate. At the second stage, the candidate may be only a couple of items, each of which is frequently encountered. For example, initially all the sets of single items ( $\{I1\}$ ,  $\{I2\}$ ,  $\{I3\}$ ,  $\{I4\}$  and  $\{I5\}$ ) have a support count of 6, 7, 6, 2, and 2 respectively. So initially all five items become frequent item-sets.

Thus, the second stage of the algorithm will form the following list of sets of candidates. Table 2 shows the item sets along with their support counts. Now frequent 2-item-sets are  $\{I1, I2\}$ ,  $\{I1, I3\}$ ,  $\{I1, I5\}$ ,  $\{I2, I3\}$ ,  $\{I2, I4\}$  and  $\{I2, I5\}$  as the other item-sets don't have the minimum support count.

*Table 2.* Item sets and the support counts

Itemset	Sup. Count
$\{I1, I2\}$	4
$\{I1, I3\}$	4
$\{I1, I4\}$	1
$\{I1, I5\}$	2
$\{I2, I3\}$	4
$\{I2, I4\}$	2
$\{I2, I5\}$	2
$\{I3, I4\}$	0
$\{I3, I5\}$	1
$\{I4, I5\}$	0

Likewise this process is continued until the no of frequent n-item-sets become zero, where n determines the no. of children in the item-set.

Apriori counts not only the provision of all frequent sets, but also ensuring those sets of candidates, which could not be discarded a priori. The set of all sets of candidates, which are rare, but whose software calculates the Apriori algorithm, called the negative region (negative border). Thus, the set belongs to a negative area if it can not be attributed to the frequent, but so are its subsets.

### C. Algorithm Optimization

Many optimizations to the primary data mining algorithms are proposed over the time. They focus on a narrow sub set of datasets or cater the mining of the data broadly.

#### 1) Pruning

Mannila et al. proposed pruning techniques to reduce the size of candidate set [9] as an optimization. Pruning strategy for the algorithm is based on a characteristic: a frequent item-set is a set, if and only if all its subsets are frequent.

#### 2) Dynamic hashing and pruning

Park et al. suggested a hash based algorithm (DHP), which generates candidate large itemsets efficiently, while reducing the size of the transaction data base [10]. Number of candidate itemsets generated by DHP is smaller than that generated using existing algorithms, making it efficient for large itemsets, specifically for the large 2-itemsets.

#### 3) Parallel Data mining

Park et al., extended DHP into a parallel data mining algorithm, facilitating parallel generation of the candidate itemsets and parallel determination of large itemsets [11].

#### 4) Transaction Reduction

Agrawal et al. proposed this algorithm to reduce the number of transactions scanned in the future iterations [12], [13]. A transaction that does not contain any frequent k-itemsets can not contain any frequent (K +1) itemsets, hence can be removed.

#### 5) Partition

Savasere et al. designed a Partition-based highly parallel algorithm [14], which logically divides the data base into several disjoint blocks. Each considered separately a sub-block and it generates all the frequent sets. The generated frequency of collection is used to generate all possible frequent sets, and followed by the final calculation of the degree of support of these itemsets. Sub-block size is chosen to allow each sub-block be placed in the main memory, each stage scanned only once.

This algorithm is highly parallel, and can be allocated to each sub-block of a processor that generates a frequency set. Frequency set is resulting, after the end of each cycle. The processor to generate the overall communication between the candidate k-itemsets. Usually here the communication process is a major bottleneck in algorithm execution time; while on the other hand, each individual processor generates frequent set time is also a bottleneck. Other methods are shared between multiple processors in a hash tree to generate frequent sets. More information on the

generated set of parallel frequency method in the literature [14] found.

#### 6) Sampling

A subset of the sample is taken for mining in the sampling based mining techniques, in the algorithm proposed by Mannila et al. [9]. Toivonen further developed this algorithm [15], where the rules are produced with the first extracted sample from the whole dataset. The remaining of the dataset is used to authenticate the dataset that is chosen. Sampling based algorithms significantly reduced the I/O costs, nevertheless with inaccurate results often and distortions in the data, known as data-skew.

#### 7) Dynamic itemset counting

Dynamic itemset counting technology [16] proposed by Brin et al. marks the starting point of the database divided into blocks. The results of algorithm need fewer database scans than Apriori.

#### 8) Equivalence CLASS Transformation (ECLAT)

Algorithm developed by Zaki [17] is used to mine the data sets efficiently using vertical data formats.

#### 9) Vertical format data mining

Apriori algorithm mines frequent patterns from a horizontal data format which represents the items categorized into particular transactions, where vertical data format represents data as transactions categorized into particular items.

Hence, for a particular item, there is a set of corresponding transaction ids. Instead of horizontal data format, Apriori can be extended to use vertical data format for efficient mining. Table III shows the transactional data represented in Table I, in the vertical format.

As can be seen from the Table 3, the vertical format data mining only has to parse the dataset once to get the itemsets. For the itemset generation from the 2<sup>nd</sup> itemset, it only needs to refer the previous itemset. This eliminates the need to parse through the dataset each time to count the frequency of itemsets, for each round. Hence, relative to the algorithms developed for the use on databases with the horizontal layout of data, the algorithms developed for the vertical representation tend to be more optimal.

Table 3. Transactional data represented in the vertical format

Item ID	List of TIDs	Support Count
I1	T100, T400, T500, T700, T800, T900	6
I2	T100, T200, T300, T400, T600, T800, T900	7
I3	T300, T500, T600, T700, T800, T900	6
I4	T200, T400	2
I5	T100, T800	2

#### 10) VIPER

VIPER [18], [19] is a general purpose algorithm, with no special requirement for the underlying database, based on

the vertical format representation. VIPER also uses compressed bit-vectors (referred to as 'snakes') to store data, with optimized generation of snakes, intersection, counting, and storage. VIPER is one of the algorithms motivated by the vertical format mining, as vertical format mining is more efficient than its horizontal format mining.

### III. Horizontal Format Data Mining with Extended Bitmaps

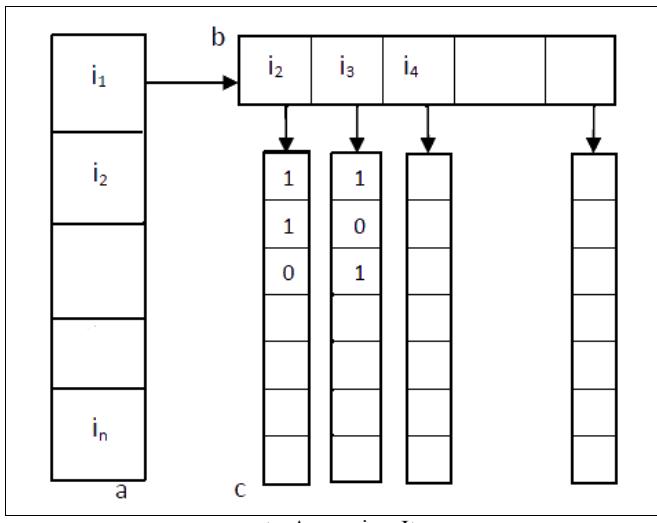
We propose the algorithm 'Horizontal Format Data Mining with Extended Bitmaps', for mining a data set in a horizontal format, by converting it to a data structure of vertical format along with bitmaps indices to store the itemset data.

#### A. Parsing the dataset to create 1st itemset and bitmaps

The first and only parsing of the dataset is done here.

- Creating the master array.
- Creating bitmaps relevant to each item in master array. Here bitmaps are used to store the individual bits compactly, exploiting the bit level parallelism effectively. In a bit map, 1 indicates the existence.

As in Figure 1, masterArray is an array of Item objects of all items in the dataset. Under each item, there is an array of other items occurring together with it in transactions which we call from, which we refer from now on as AssociatedItems. Under each AssociatedItem, we store a bit vector indicating the presence of the AssociatedItem for every transaction where the Item is found. Basically, length of bit vectors give the number of transactions that its root Item in masterArray is found. If the AssociatedItem is in the transaction, bit vector value for that transaction will be 1 (true), otherwise 0 (false).



**Figure 1.** Main data storage structure of the extended vertical data mining.

#### B. Remove redundant associated items

Here for better memory utilisation, the redundant duplicate mappings (AssociatedItems) are pruned from the main data structure such as, Item  $i_1$  mapped to AssociatedItem  $i_2$  and Item  $i_2$  mapped to AssociatedItem  $i_1$  in the masterArray.

#### C. Pruning itemsets according to apriori property

Association mining is carried out solely in this step, in three major phases. The core logic of Apriori property is used; but implementation is done using the manipulation of bit vectors.

##### 1) 1<sup>st</sup> frequency itemset extraction

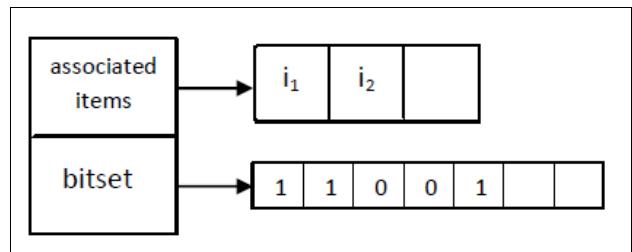
The items not satisfying the minimum support are removed from the masterArray [1].

##### 2) 2<sup>nd</sup> frequency itemset extraction

Here we iterate through the associatedItemArrays for each Item in masterArray. For each AssociatedItem, the cardinality of its bit vector is compared with the minimum support value. AssociatedItems having less than the minimum support are removed.

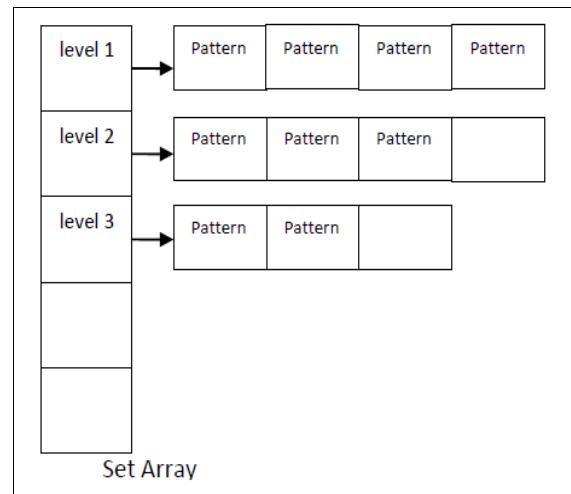
##### 3) Extracting frequency itemsets above two

For mining frequency item sets of three or above, we start intersecting bit vectors of AssociatedItems, compare result bit vectors' cardinality with minimum support value, and remove the AssociatedItems as Apriori property suggests. This will take place for increasing frequency itemset values and stop when no more frequent item sets are present. In this recurring step, two main data structures [Figure 2, Figure 3] are used.



**Figure 2.** Pattern structure

Pattern structure Figure 2 is used to store each resultant bit vector from intersection of bit vectors and the itemset ids for those intersected bit vectors' belonging items. As Pattern is also a representation of item sets, we can use it as a frequent itemset itself.



**Figure 3.** Candidates structure

Candidates structure Figure 3, is used to store the frequent item sets of each level of itemset size. Here each candidate level is an array of patterns. Each level indicates the

frequency itemset level. Hence frequent itemsets of length one will be under level one, frequent itemsets of length two will be under level two and so on. Once the above steps are completed, the frequent item sets are retrieved by iterating through the candidate structure.

## IV. Analysis

We discussed the algorithm, "Horizontal format data mining with extended bitmaps". Here we are going to analyze the algorithm with a simple data set.

Table 4. Sample data set for the analysis

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I1, I2
T400	I2, I5

Table 4 shows the data set to be considered for this analysis. Here we are taking a very simple data set for the ease of analysis. In this analysis, we will use the proposed algorithm to find the frequent item sets with the minimum support = 2.

### A. Building the master array and the bitmaps

We first build the master array with all the items in the sample data set. Then go through the first transaction  $T100 = \{I1, I2, I5\}$ .

The associated item array is built with the items in the first transaction. I2 and I5 are linked to I1 in the master array, as the associated items in the associated item array for I1. Similarly, I5 is stored in the associated item array of I2, as shown in Figure 4.

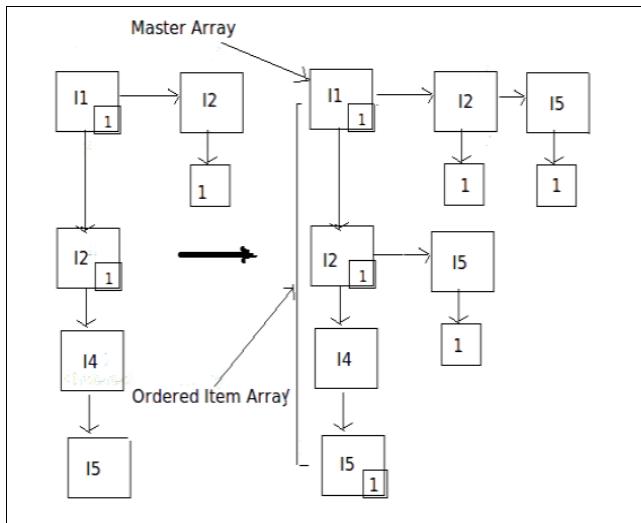


Figure 4. Data structure after going through T100

We avoid the redundancy by storing I1 in the associated item array of I2 for the same transaction, as  $\{I1, I2\}$  and  $\{I2, I1\}$  are identical for the transaction. Hence associated item arrays built with  $T100: \{I1, I2, I5\}$  and  $\{I2, I5\}$ .

Under each associated item in the respective associated item arrays, a bit is used to indicate the existence of the item.

The items in the transaction are counted and stored in the master array. Here I1, I2, and I5 are indicated by '1' in the master array, to show that the item exists once. Similarly for the next transaction  $T200 = \{I2, I4\}$ , I2 is updated with the count of 2, and I4 with 1, in the master array.

Associated Item Array of I2 is updated. Here I4 is linked to I5 in the array of I2.  $\{I2, I5, I4\}$ .

A '0' is used under I4 of the associated item array of I2 to indicate that for  $T100$ , only I5 was associated to I2. Similarly the '0' for  $T200$  under the bitmap of I5, in the associated item array of I2 show that I5 wasn't associated with I2.

From Figure 5, we can read that,

$$T100 = \{I1, I2 (1), I5 (1)\} = \{I1, I2, I5\}$$

$$T100 = \{I2, I5 (1), I4 (0)\} = \{I2, I5\}$$

$$T200 = \{I2, I5 (0), I4 (1)\} = \{I2, I4\}$$

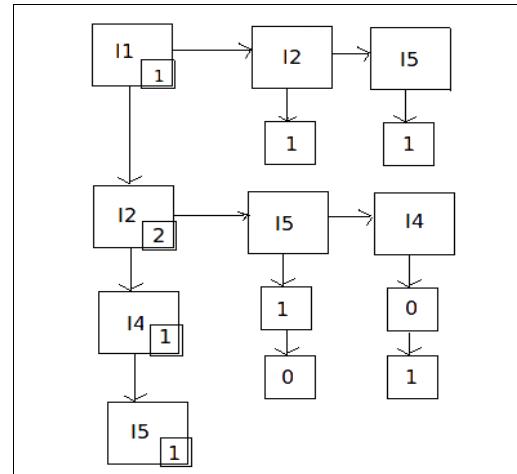


Figure 5. Data structure after T200

Figure 6 shows the data structure in our notation, updated with  $T300 = \{I1, I2\}$ .

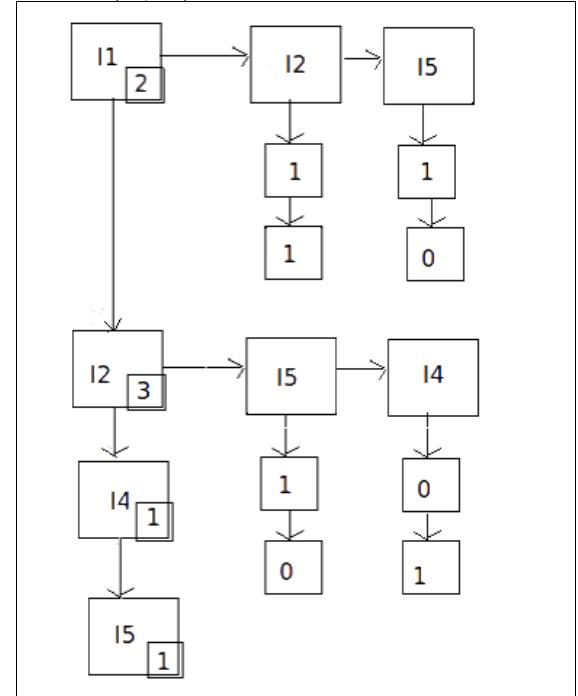


Figure 6. Data structure after T300

Figure 7. shows the final data structure, with T400, where all the master array is updated with all the transactions, and the relevant bit maps are created for each item in the master array. Reading the associated item arrays for each item in the master array shows the below.

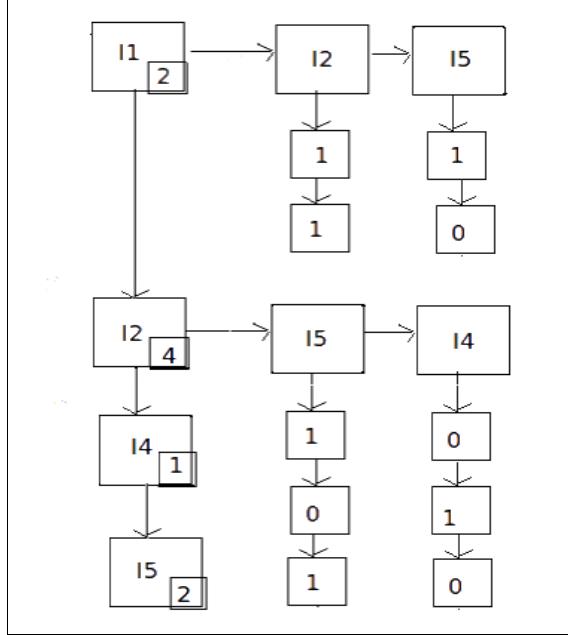
$$I1, I2 (1), I5 (1) = \{I1, I2, I5\}$$

$$I1, I2 (1), I5 (0) = \{I1, I2\}$$

$$I2, I5 (1), I4 (0) = \{I2, I5\}$$

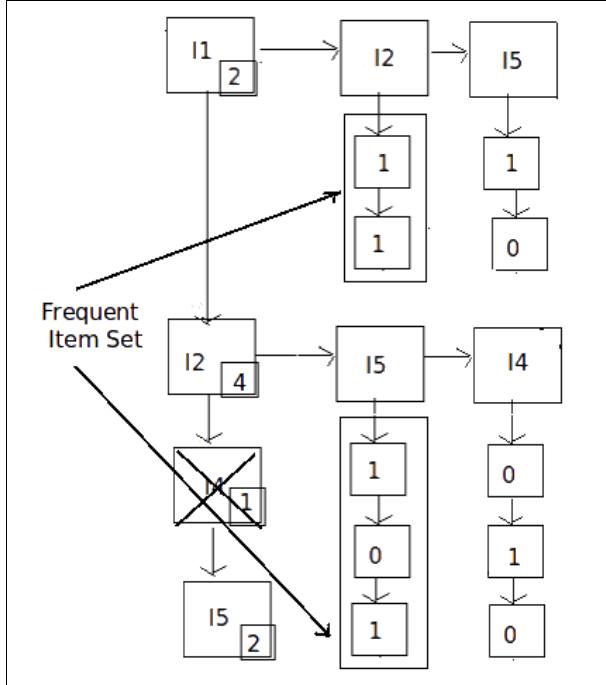
$$I2, I5 (0), I4 (1) = \{I2, I4\}$$

$$I2, I5 (1), I4 (0) = \{I2, I5\}$$



**Figure 7.** Final Data structure

#### B. Pruning itemsets



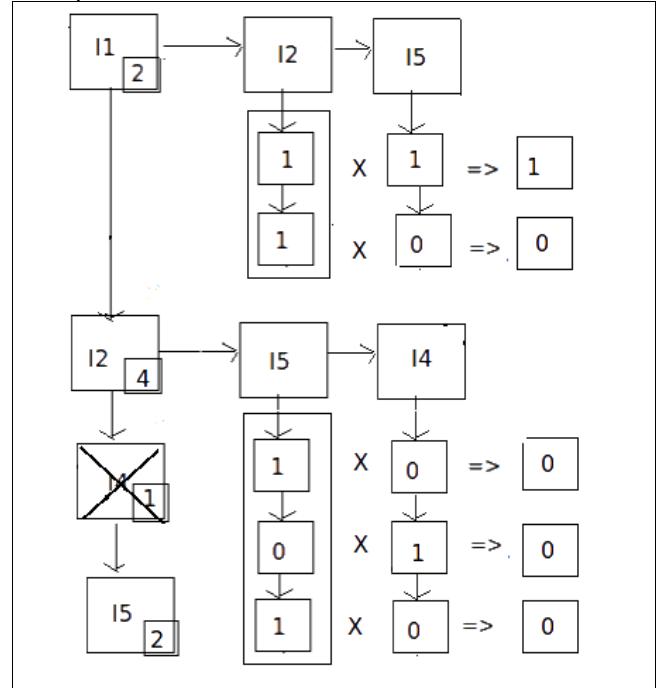
**Figure 8.** Frequent itemset

As shown in Figure 8, since our minimum support is 2, I4 is removed from further calculations, as it doesn't satisfy the minimum support level.

For each AssociatedItem, the cardinality of its bit vector is compared with the minimum support value. AssociatedItems having less than the minimum support are removed. Hence only {I1, I2} and {I2, I5} remain.

Now we have to extract the frequency itemsets above two. We start intersecting the bit vectors of AssociatedItems. Intersecting {I1, I2} with {I1, I5} and {I2, I5} with {I2, I4}. We compare result bit vectors' cardinality with minimum support value 2. Here, as depicted in Figure 9, the intersection shows that {I1, I2, I5} appears once, and {I2, I5, I4} doesn't appear as a frequent set.

Since these do not satisfy the minimum support level of 2, we stop here.



**Figure 9.** Final Frequent Item Sets

From the algorithm, we derive that {I1, I2} and {I2, I5} are the frequent sets that satisfy the minimum support.

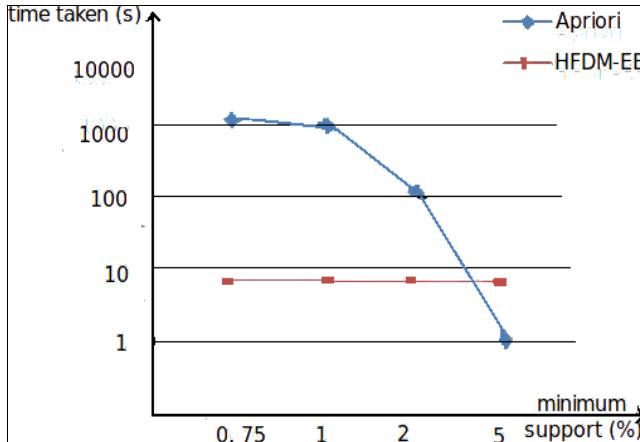
## V. Experimental Study

The algorithm was implemented and benchmarked in a system with 2 GB memory and 2.5 Ghz core 2 Duo Processor, against an implementation of Apriori algorithm for the datasets chosen from Frequent Itemset Mining Dataset Repository [20], along with different values of minimum support, as both the algorithms mine the datasets in horizontal form. Here both Apriori and the Horizontal Format Data-mining with Extended Bitmaps algorithm are designed for the databases having the horizontal layout.

*Table 5. T= 10; I = 4; D = 10K*

Minimum Support (%)	T10I4D10K	
	Apriori (sec)	HFDM-EB (sec)
0.75	1934.4	10.6
1	1365.4	10.5
2	238.4	10.4
5	1.6	9.6

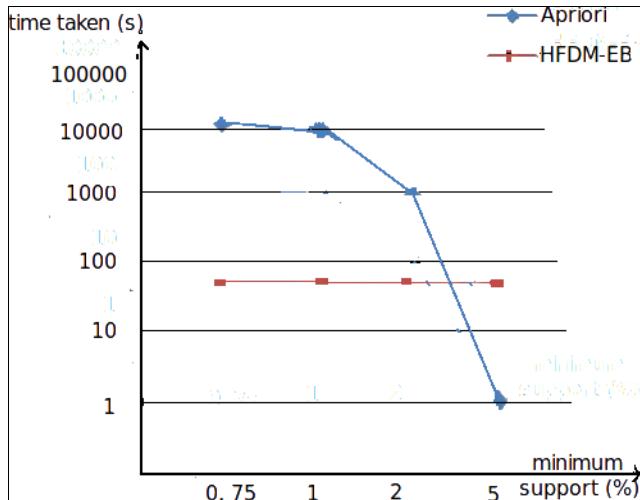
Table 5, Table 6, and Table 7 compare the performance of the algorithms. The minimum support vs time taken are plotted for ( $T = 10, I = 4, D = 10K$ ), ( $T = 10, I = 4, D = 100K$ ), and ( $T = 40, I = 10, D = 100K$ ), which are shown in Figure 10, Figure 11, and Figure 12 respectively. Time is shown in the log scale.



**Figure 10.**  $T = 10; I = 4; D = 10K$

*Table 6.*  $T = 10; I = 4; D = 100K$

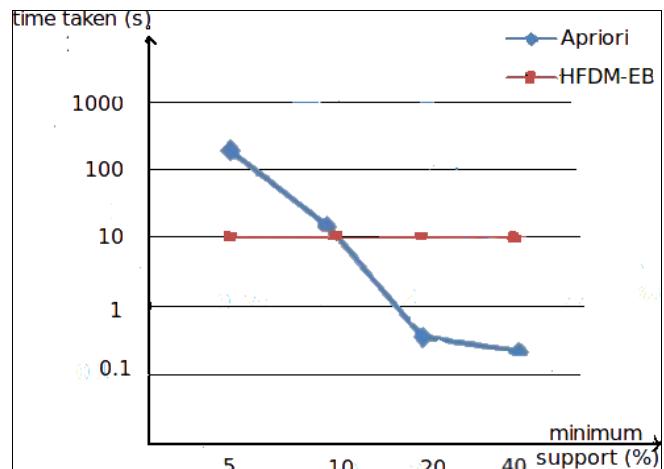
Minimum Support (%)	T10I4D100K	
	Apriori (sec)	HFDM-EB (sec)
0.75	21039.1	134.2
1	12600.5	132.1
2	2365.1	131.7
5	1.3	132.1



**Figure 11.**  $T = 10; I = 4; D = 100K$

*Table 7.*  $T = 40; I = 10; D = 100K$

Minimum Support (%)	T40I10D100K	
	Apriori (sec)	HFDM-EB (sec)
5	232	10.9
10	19	10.8
20	0.5	10.6
40	0.3	10.4



**Figure 12.**  $T = 40; I = 10; D = 100K$

The time taken to build the bitmap is independent of the number of frequent item sets. But Apriori's time drops drastically when the number of frequent items is low. Thus Apriori was having a higher performance at high support levels where number of frequent item sets found is low.

## VI. Conclusion and future work

Large companies for decades accumulated data on their customers, suppliers, products and services. Due to high rate of development of e-commerce working in Web start-ups can turn into a huge enterprise in a matter of months, rather than something those years. And, as a consequence, will grow rapidly and their databases.

Data mining, also called 'knowledge discovery in databases' [21] provides organizations with the tools developed to analyse the large collection of information to find trends, patterns and relationships that can help in making strategic decisions. In this paper we have proposed an efficient algorithm for Association Rule Mining, which recovers the associations as the Apriori on a data set in horizontal format, utilizing the bitmaps.

We have implemented the algorithm in Java, which may be more efficient, if implemented in C or a lower level language, so that we can control the memory allocation, in the most optimal way for the algorithm, as we want.

For each data item, a bitmap is created for each associated item. If there are  $n$  associated items for a data item, then the number of candidate sets generated is  $n(n-1)$ . So there can be redundant bitmaps created for the same data item pairs. Currently, redundant pairs are pruned after creating the vertical format. But, if there is a dynamic pruning mechanism to prune redundant data item pairs while creating the vertical format memory can be well optimized.

The algorithm lends well to Map Reduce like distributed data mining since mining of each data item is independent of others. Each master array index is self contained, and hence can be mined in parallel. So this algorithm can be enhanced to work in a distributed environment with or without a shared memory. Here data structure generation becomes the Map phase, where the result accumulation becomes the Reduce phase, as in Map-Reduce.

For some larger data sets that are having many items per transaction, the algorithm fails to withstand due to utilizing prohibitive amount of memory. It can be mitigated by using

compressed bitmap [22] implementation instead of plain bitmaps, so the memory is utilized better.

## References

- [1] Jiawei Han and Micheline Kamber. "Data Mining, Concepts and Techniques," 2nd Edition, 2006.
- [2] Abraham Silberschatz, Henry F.Korth, and S.Sudarshan. "Database System Concepts," 5th Edition. McGraw-Hill, Inc., New York, San Francisco, Washington, DC, 2005.
- [3] Jaideep Srivastava , Robert Cooley , Mukund Deshpande, Pang-Ning Tan. "Web Usage Mining: Discovery and Applications of Usage," Department of Computer Science and Engineering, University of Minnesota. ACM SIGKDD Explorations Newsletter, v.1 n.2, January 2000.
- [4] Wenke Lee , Salvatore J. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th conference on USENIX Security Symposium*, p.6-6, January 26-29, 1998, San Antonio, Texas.
- [5] Elisabeth Georgii, Lothar Richter, Ulrich Rückert and Stefan Kramer. "Analyzing microarray data using quantitative association rules," *Bioinformatics* 2005, 21(Suppl 2):ii123-ii129.
- [6] Ioannidis, John P. A. (August 30, 2005). "Why Most Published Research Findings Are False". *PLoS Medicine* (San Francisco: Public Library of Science) 2 (8). doi:10.1371/journal.pmed.0020124. ISSN 1549-1277.
- [7] Behrouz Minaei-Bidgoli1, William F. Punch. "Using Genetic Algorithms for Data Mining - Optimization in an Educational Web-based System," Genetic Algorithms Research and Applications Group (GARAGe). Department of Computer Science & Engineering. Michigan State University. Online: [www.lon-capu.org/papers/v90-gapaper.pdf](http://www.lon-capu.org/papers/v90-gapaper.pdf) [Accessed: 25th December, 2009]
- [8] Wei-Qing Sun, Cheng-Min Wang, Tie-Yan Zhang, and Yan Zhang. 2009. "Transaction-item association matrix-based frequent pattern network mining algorithm in large-scale transaction database." *W. Trans. on Comp.* 8, 8 (Aug. 2009), 1327-1336.
- [9] H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithm for discovering association rules. *AAAI Workshop on Knowledge Discovery in Databases*, pp 181-192, Jul. 1994.
- [10] J.S. Park, M.S. Chen, and PS Yu. "An effective hash-based algorithm for mining association rules". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp 175-186, May 1995.
- [11] J.S. Park, M.S. Chen, and P.S. Yu. "Efficient parallel data mining of association rules," *4th International Conference on Information and Knowledge Management*, Baltimore, Maryland, Nov. 1995.
- [12] R. Agrawal, and R. Srikant. "Fast algorithms for mining association rules," in *Proceedings of. 1994 Int. Conf. Very Large Databases (VLDB'94)*, Sep. 1994.
- [13] J. Han and Y. Fu. "Discovery of multiple-level association rules from large databases," in *Proceedings of. Int. Conf. Very Large Databases (VLDB'95)*, pp 402-431, Sep. 1995.
- [14] A. Savasere, E. Omiecinski, and S. Navathe. "An efficient algorithm for mining association rules in large databases," in *Proceedings of the 21st International Conference on Very Large Database*, pp 432-443, Sep. 1995.
- [15] H. Toivonen. "Sampling large databases for association rules," in *Proceedings of the 22nd International Conference on Very Large Database*, Bombay, India, pp 134-145, Sep. 1996.
- [16] S. Brin, R. Motwani, JD Ullman, and S. Tsur. "Dynamic itemset counting and implication rules for market basket data," in *Proceedings of ACM SIGMOD International Conference On the Management of Data*, pp 255-264, May 1997.
- [17] M. J. Zaki. "Scalable algorithms for association mining," *IEEE Trans. Knowledge and Data Engineering*, 12:372-390, 2000.
- [18] Shenoy, P. and Haritsa, J. and Sudarshan, S. and Bhalotia, G. and Bawa, M. and Shah, D. (2000) "VIPER: A Vertical Approach to Mining Association Rules." In: *ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, May 16-18, 2000 , Dallas, Texas.
- [19] Shenoy, P. and Haritsa, J. and Sudarshan, S. and Bhalotia, G. and Bawa, M. and Shah, D. "Turbo-Charging Vertical Mining of Large Databases".
- [20] Frequent Itemset Mining Dataset Repository. [Online]. Available: <http://fimi.cs.helsinki.fi/data/> [Accessed: 25th December, 2009]
- [21] UM Fayyad et al., Eds. "Advances in Knowledge Discovery and Data Mining," AAAI/MIT Press, Menlo Park, Calif., 1996. *Advances in Knowledge Discovery and Data Mining*, AAAI / MIT Press, Menlo Park, Calif., 1996.
- [22] Kesheng Wu, Ekow Otoo and Arie Shoshani. "Optimizing Bitmap Indices with Efficient Compression." *ACM Transactions on Database Systems*, v31, pages 1 - 38, March, 2006.

## Author Biographies

**Buddhika De Alwis** received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa in 2010. He is currently a software engineer at WSO2 Inc. His research interests include data mining, distributed and cloud computing.

**A. Supun Malinga** is a software Engineer at WSO2 Inc. He received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa. His research interests include Distributed computing, SOA, middleware.

**Kathiravelu Pradeeban** obtained his B.Sc (Hons) in Computer Science and Engineering from the University of Moratuwa. He is a software engineer at WSO2 Inc., currently working with WSO2 Cloud Middleware Platform. His research interests include distributed computing, data mining, and SOA web services.

**W.A. Denis Dhananjaya Weerasiri** is a software engineer at WSO2 Inc. He received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa in 2010. His research interests include cloud computing, distributed computing and business process management.

**Amal Shehan Perera** is a senior lecturer in computer science at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. He served as a senior visiting research scholar at North Dakota State University in 2009. He has multiple research publications and a co-owner of a US patent. His areas of interest include data mining, database systems, and software engineering. He won the ACM KDD Cup 2006 for data mining in collaboration with Prof. William Perrizo. He obtained his MSc and PhD from North Dakota State University under the guidance of Prof. Perrizo and his BSc from University of Colombo, Sri Lanka.

# Association Analysis (3)

# FP-Tree/FP-Growth Algorithm

- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets.

## Building the FP-Tree

1. Scan data to determine the support count of each item.  
Infrequent items are discarded, while the frequent items are sorted in decreasing support counts.
2. Make a second pass over the data to construct the FP-tree.  
As the transactions are read, before being processed, their items are sorted according to the above order.

# First scan – determine frequent 1-itemsets, then build header

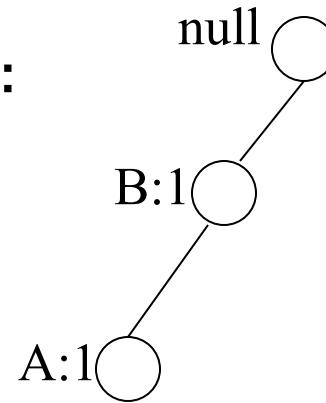
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

B	8
A	7
C	7
D	5
E	3

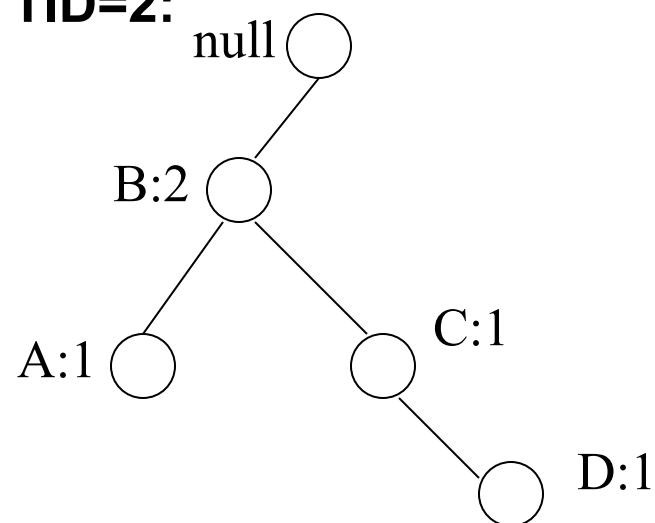
# FP-tree construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



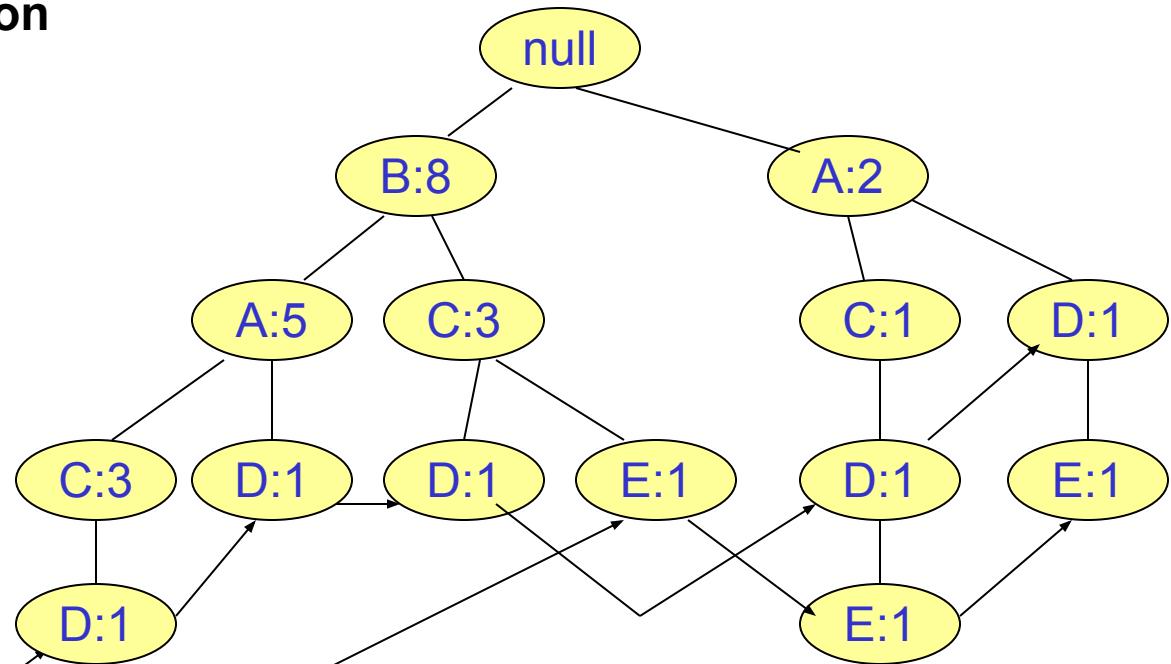
After reading TID=2:



# FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

**Transaction Database**



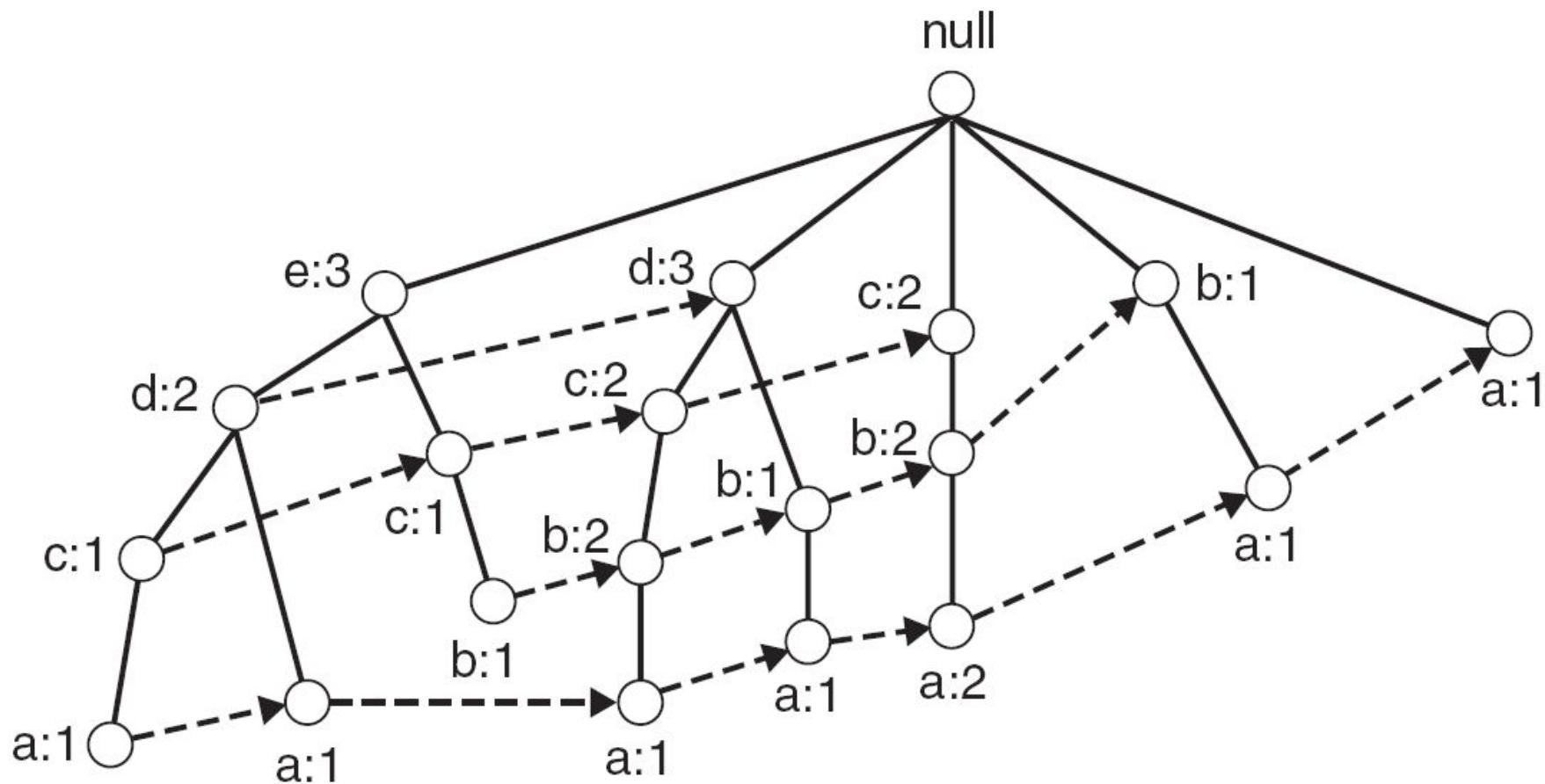
**Header table**

Item	Pointer
B	8
A	7
C	7
D	5
E	3

Chain pointers help in quickly finding all the paths of the tree containing some given item.

# FP-Tree size

- The size of an FP-tree is typically smaller than the size of the uncompressed data because many transactions often share a few items in common.
- **Best-case** scenario:
  - All transactions have the same set of items, and the FP-tree contains only a single branch of nodes.
- **Worst-case** scenario:
  - Every transaction has a unique set of items.
  - As none of the transactions have any items in common, the size of the FP-tree is effectively the same as the size of the original data.
- The size of an FP-tree also depends on how the items are ordered.
  - If the ordering scheme in the preceding example is reversed,
    - i.e., from lowest to highest support item, the resulting FP-tree probably is denser (shown in next slide).
    - Not always though...ordering is just a heuristic.

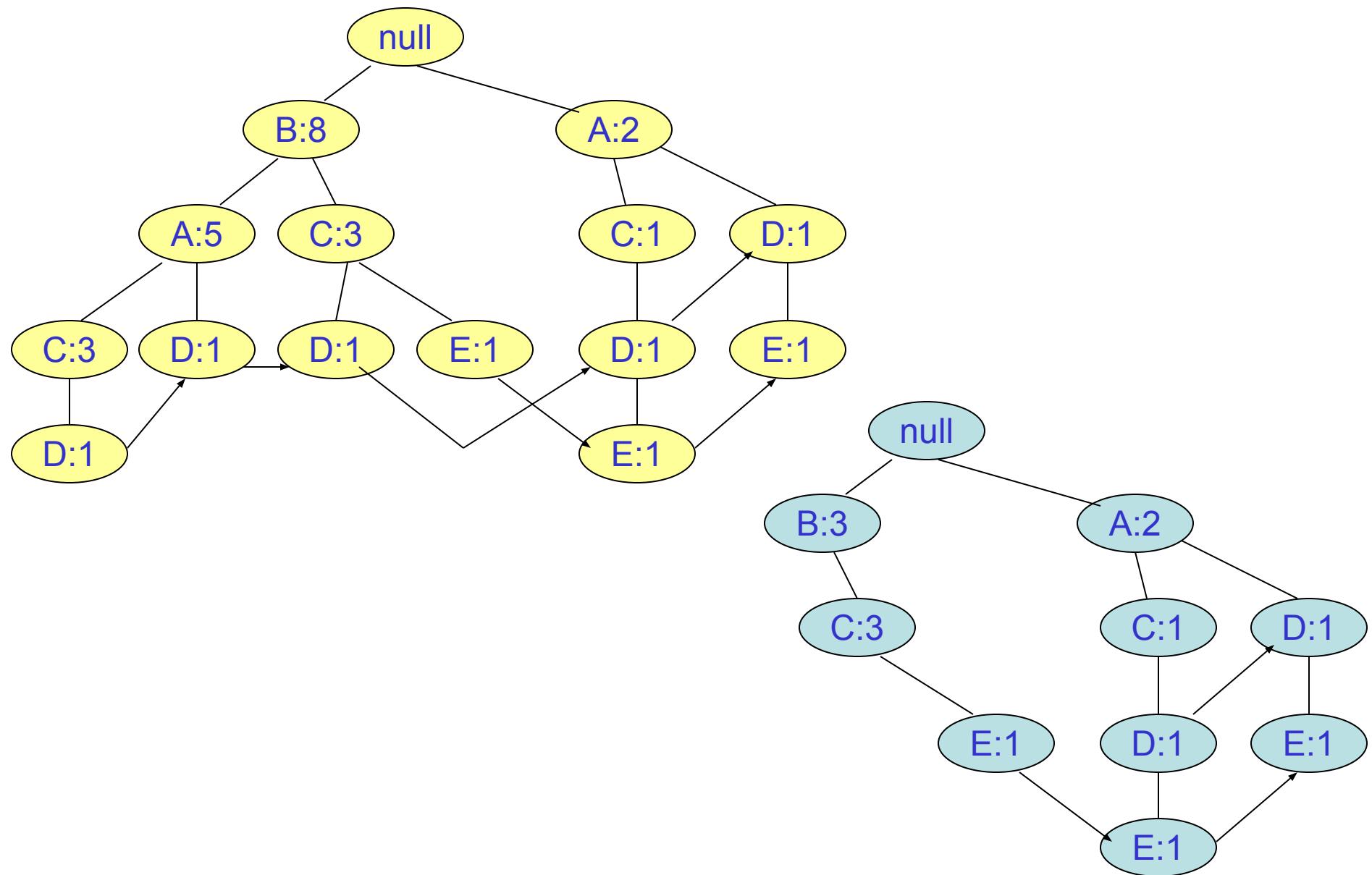


An FP-tree representation for the data set with a different item ordering scheme.

# FP-Growth (I)

- FP-growth generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion.
- Given the example tree, the algorithm looks for **frequent itemsets** ending in **E** first, followed by **D**, **C**, **A**, and finally, **B**.
- Since every transaction is mapped onto a path in the FP-tree, we can derive the **frequent itemsets** ending with a particular item, say, **E**, by examining only the paths containing node **E**.
- These paths can be accessed rapidly using the pointers associated with node **E**.

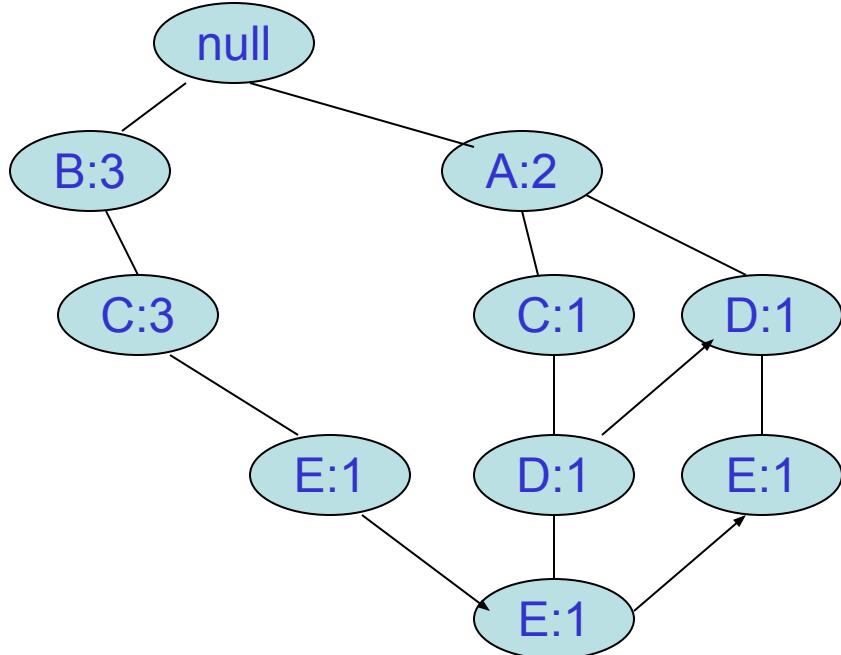
# Paths containing node E



# Conditional FP-Tree for E

- We now need to build a conditional FP-Tree for **E**, which is the tree of itemsets ending in **E**.
- **It is not** the tree obtained in previous slide as result of deleting nodes from the original tree.
- Why? Because the order of the items change.
  - In this example, **C** has a higher count than **B**.

# Conditional FP-Tree for E



The set of paths containing E.

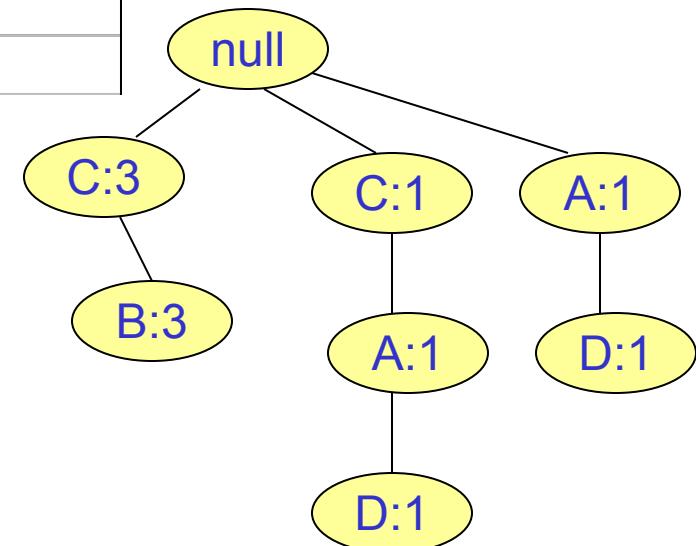
Insert each path (after truncating E) into a new tree.

Header table

Item	Pointer
C	4
B	3
A	2
D	2

The new header

The conditional  
FP-Tree for E



Adding up the counts for D we get 2, so {E,D} is frequent itemset.

We continue recursively.

Base of recursion: When the tree has a single path only.

# FP-Tree Another Example

Transactions

A B C E F O

A C G

E I

A C D E G

A C E G L

E J

A B C E F P

A C D

A C E G M

A C E G N

Freq. 1-Itemsets.  
Supp. Count  $\geq 2$

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	

Transactions with items sorted based on frequencies, and ignoring the infrequent items.

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

A C D

A C E G

A C E G

# FP-Tree after reading 1<sup>st</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

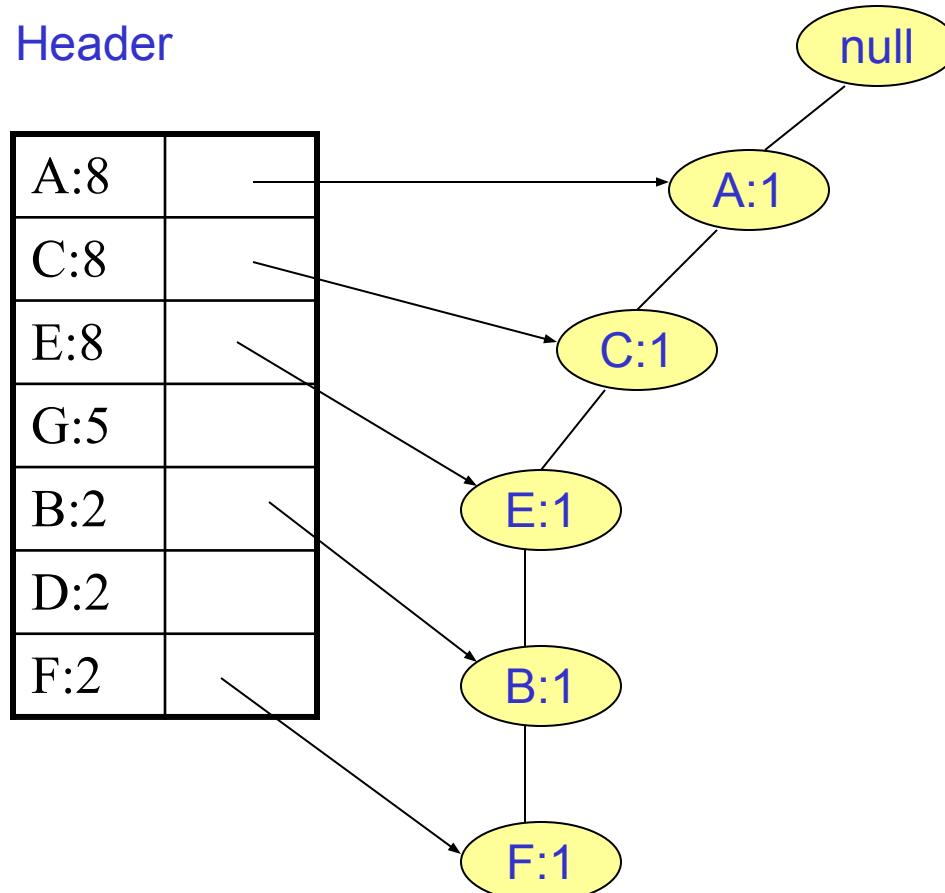
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 2<sup>nd</sup> transaction

A C E B F

**A C G**

E

A C E G D

A C E G

E

A C E B F

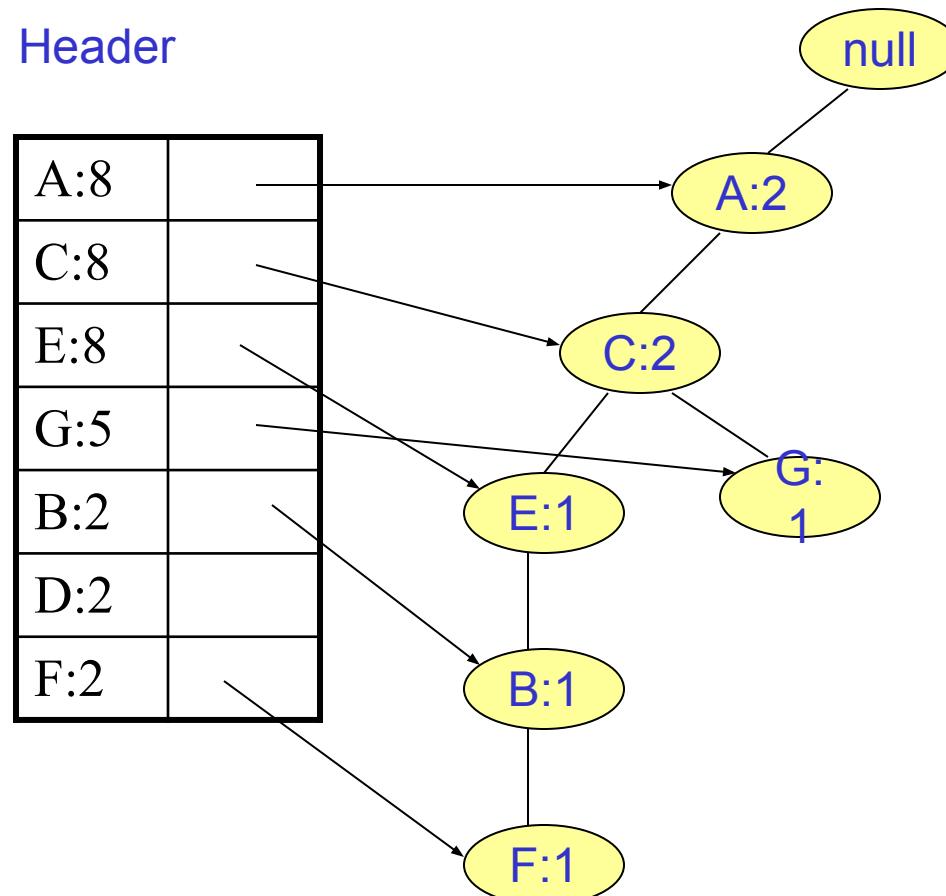
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 3<sup>rd</sup> transaction

A C E B F

A C G

**E**

A C E G D

A C E G

E

A C E B F

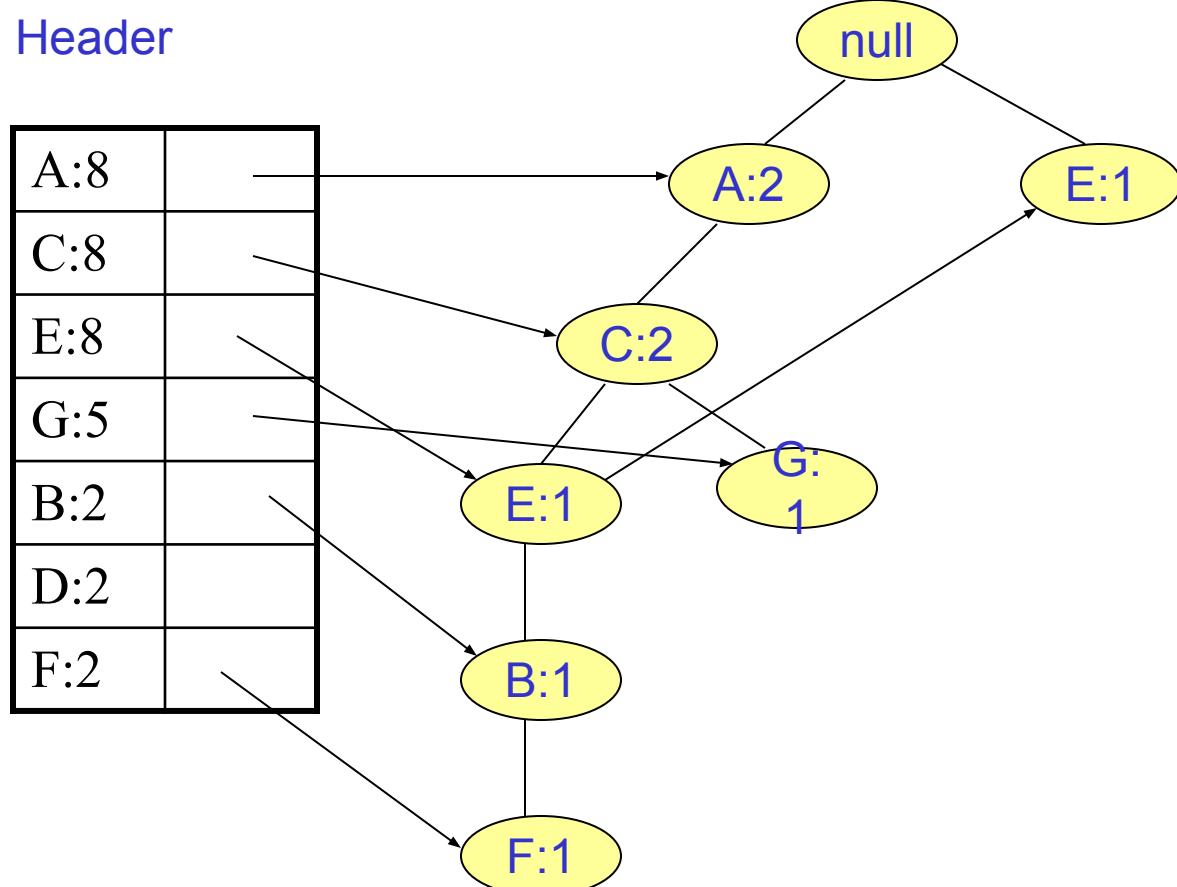
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 4<sup>th</sup> transaction

A C E B F

A C G

E

**A C E G D**

A C E G

E

A C E B F

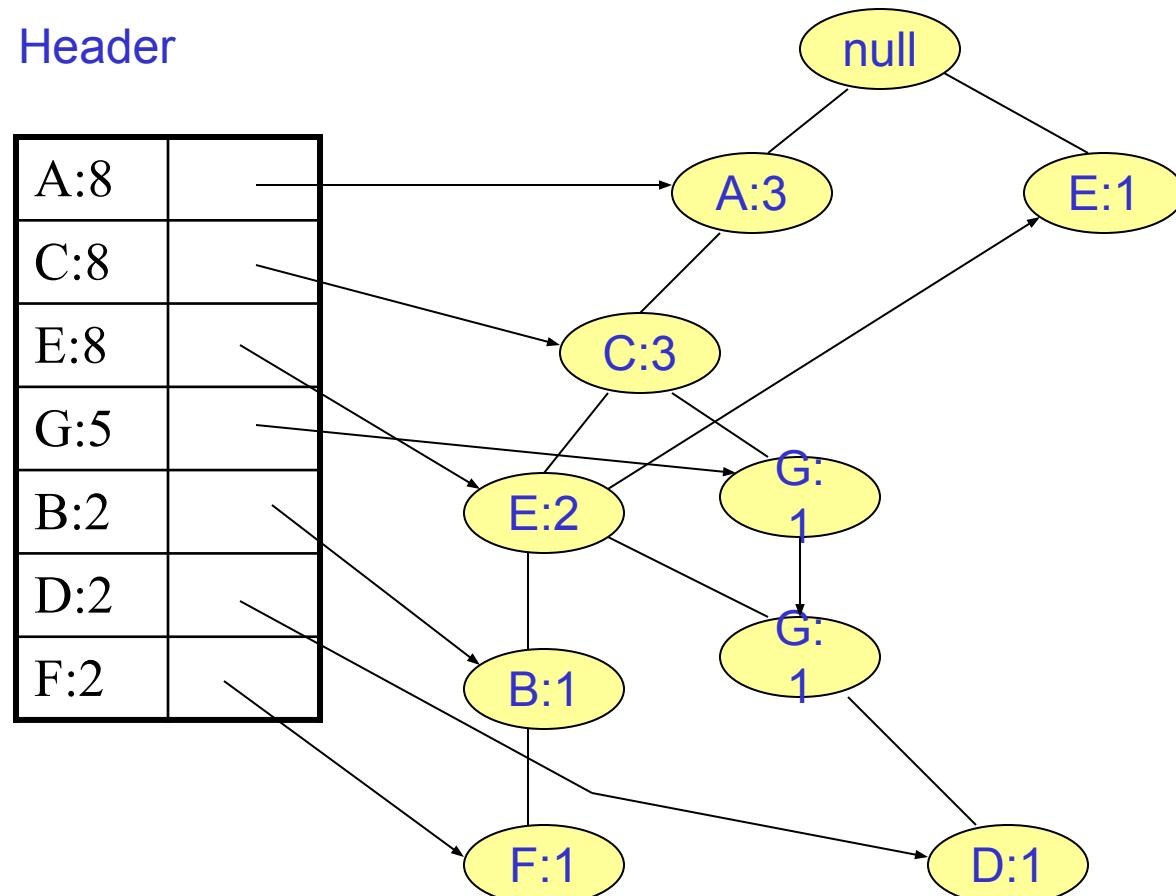
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 5<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

**A C E G**

E

A C E B F

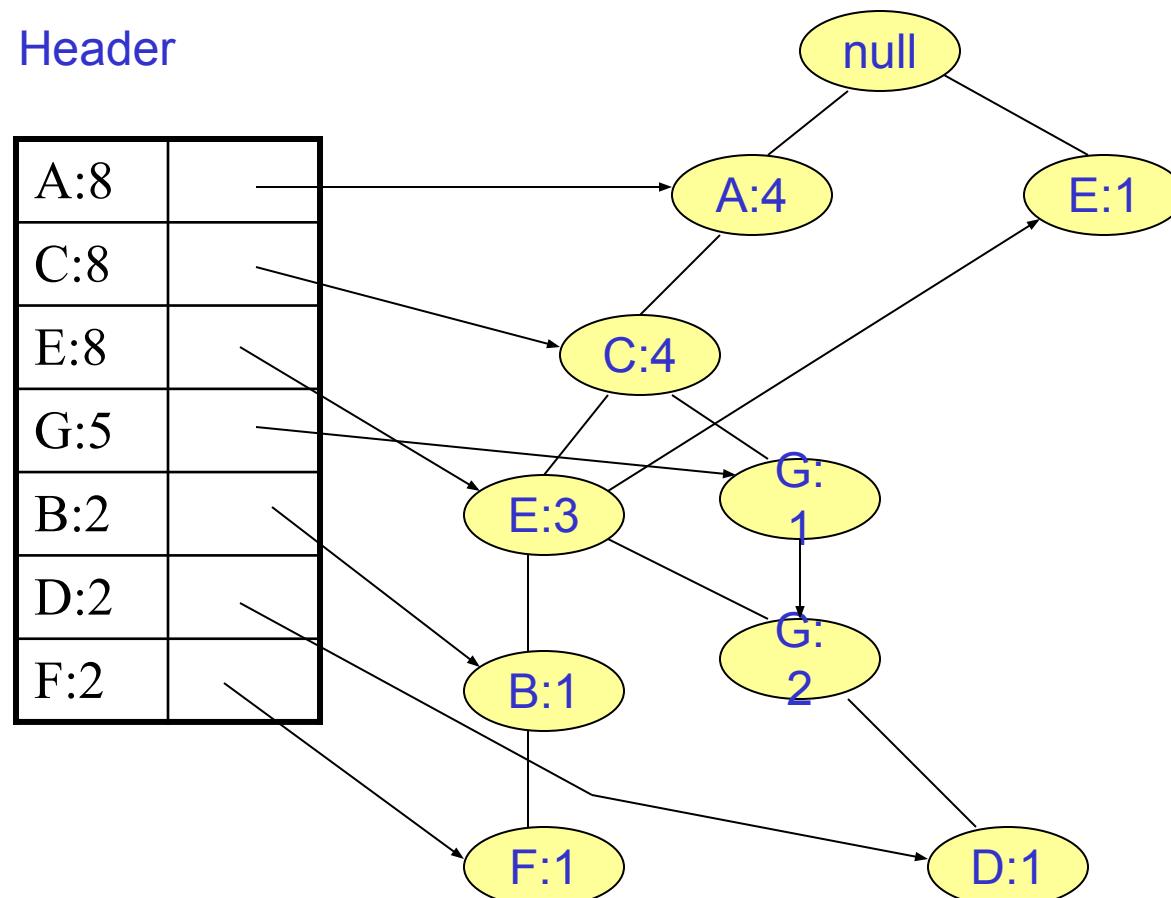
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 6<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

**E**

A C E B F

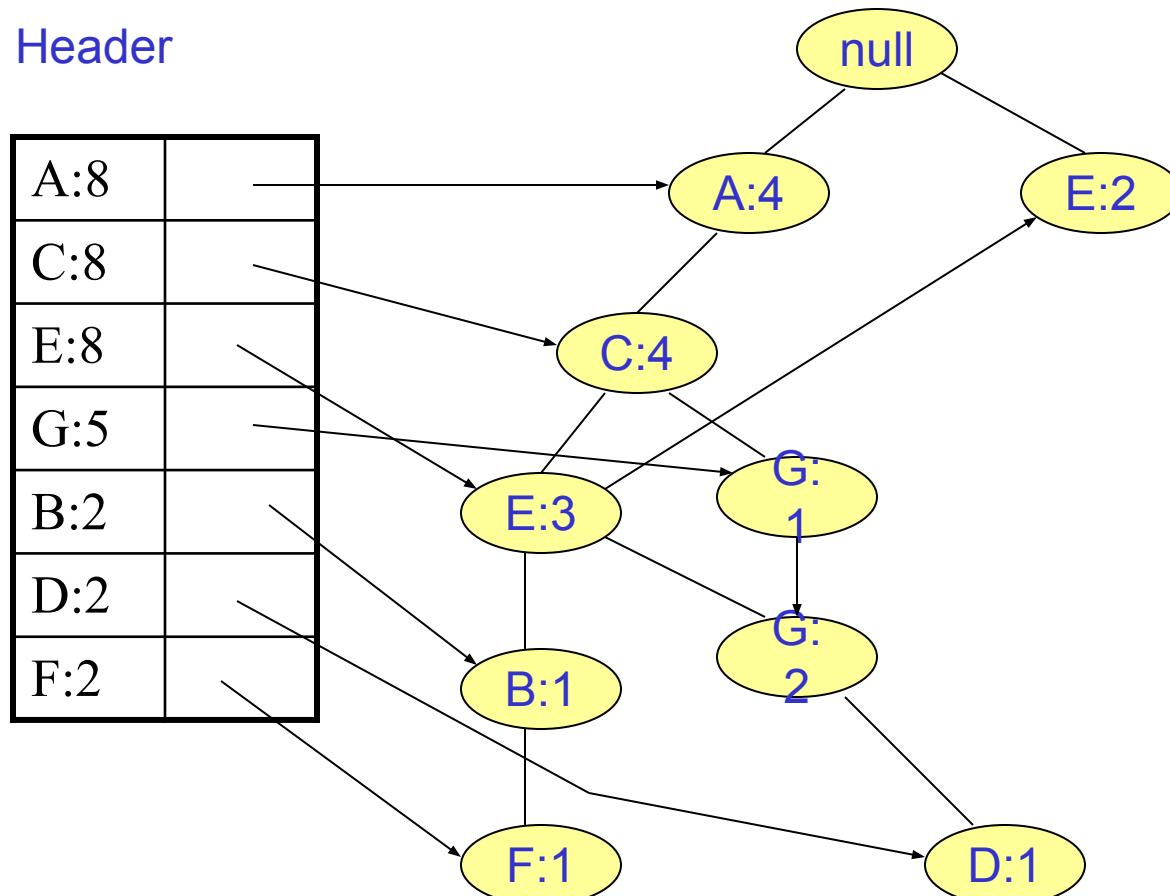
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 7<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

E

**A C E B F**

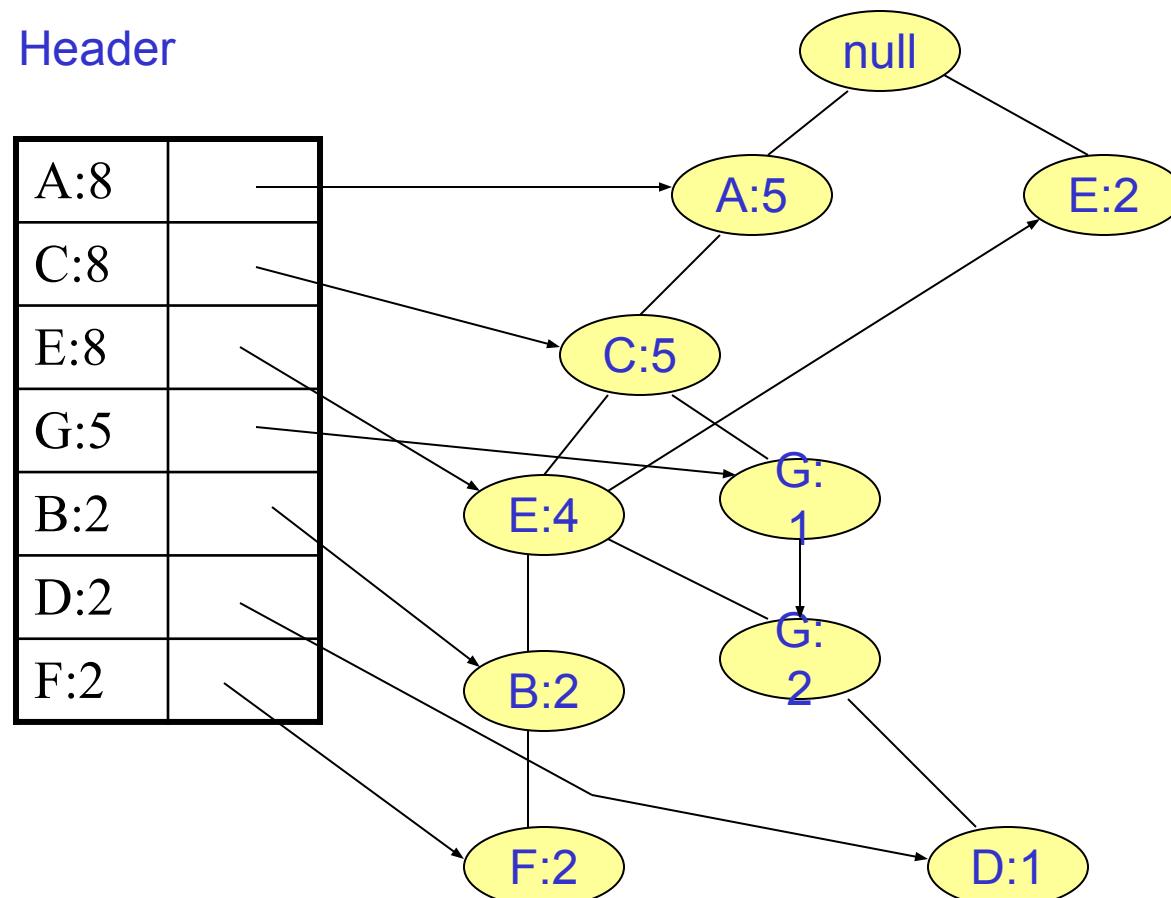
A C D

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 8<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

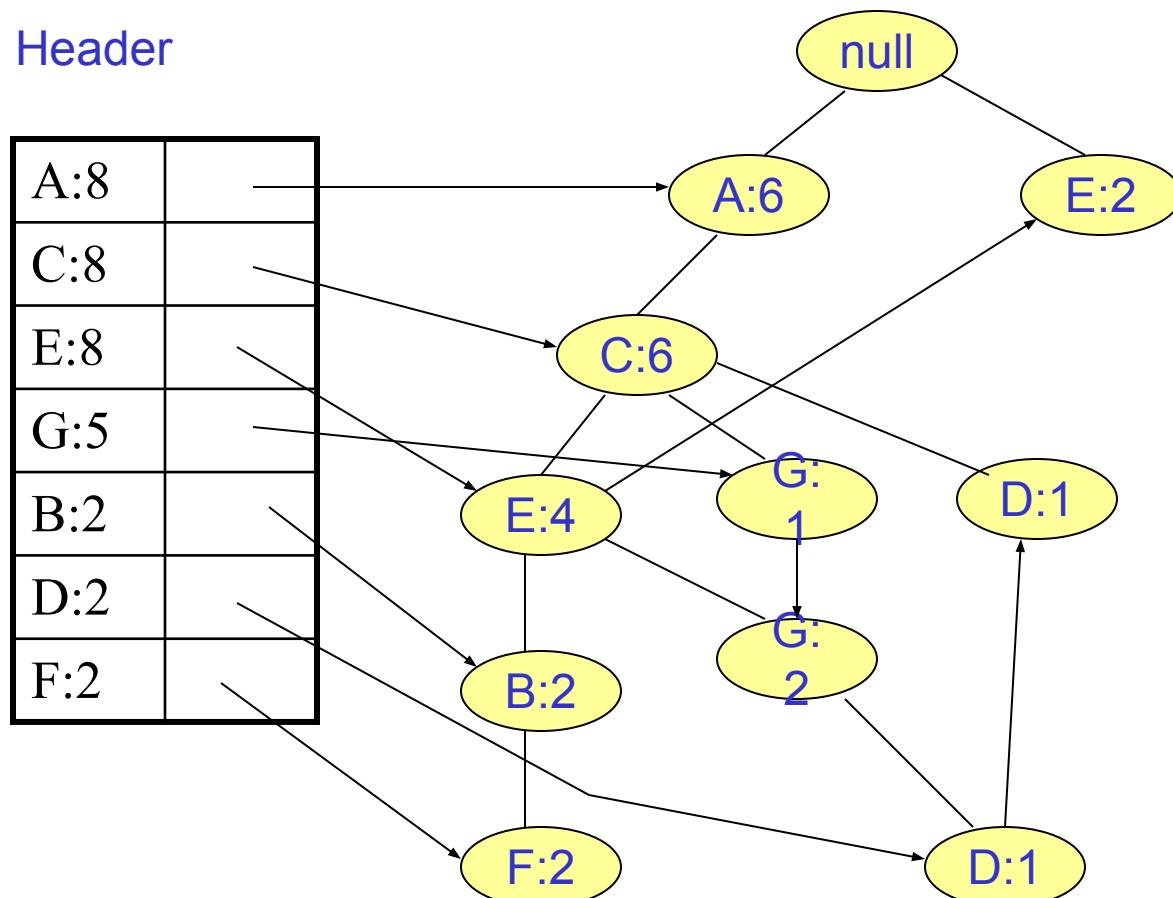
**A C D**

A C E G

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 9<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

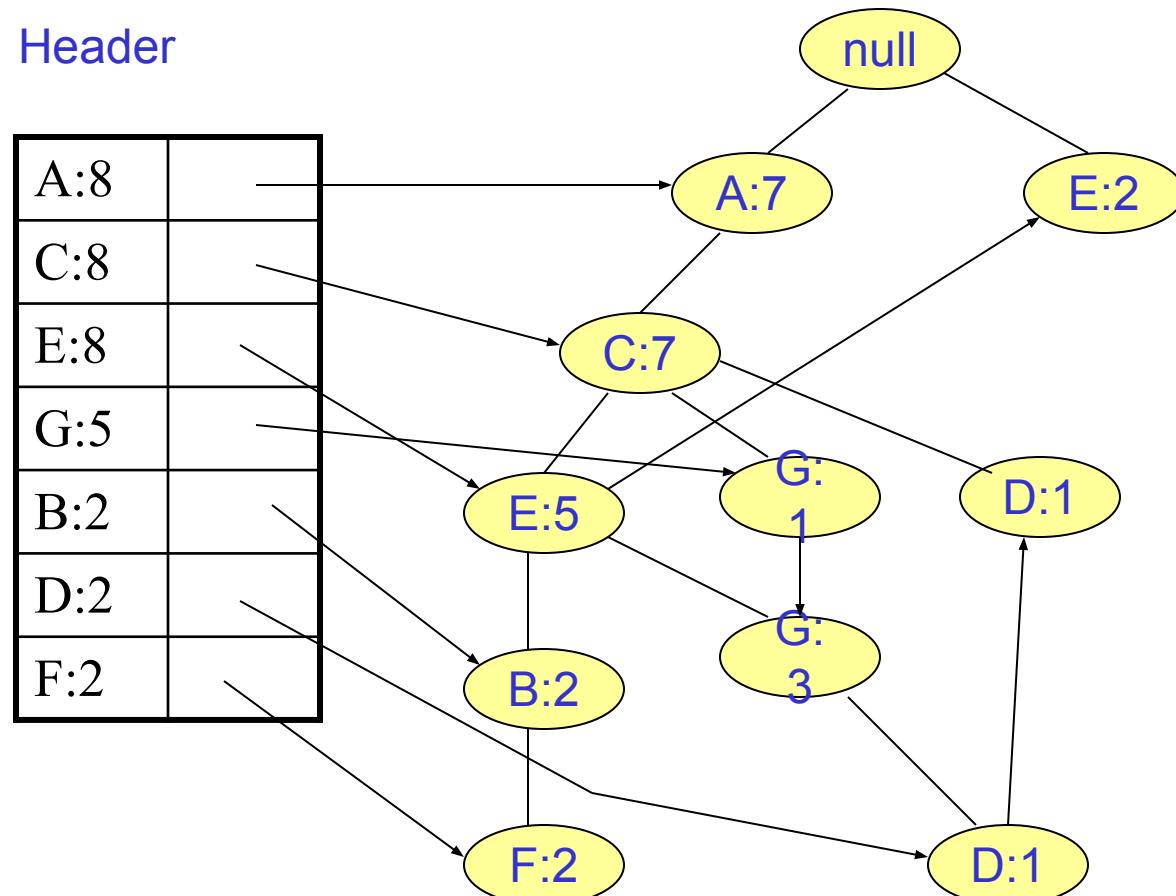
A C D

**A C E G**

A C E G

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# FP-Tree after reading 10<sup>th</sup> transaction

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

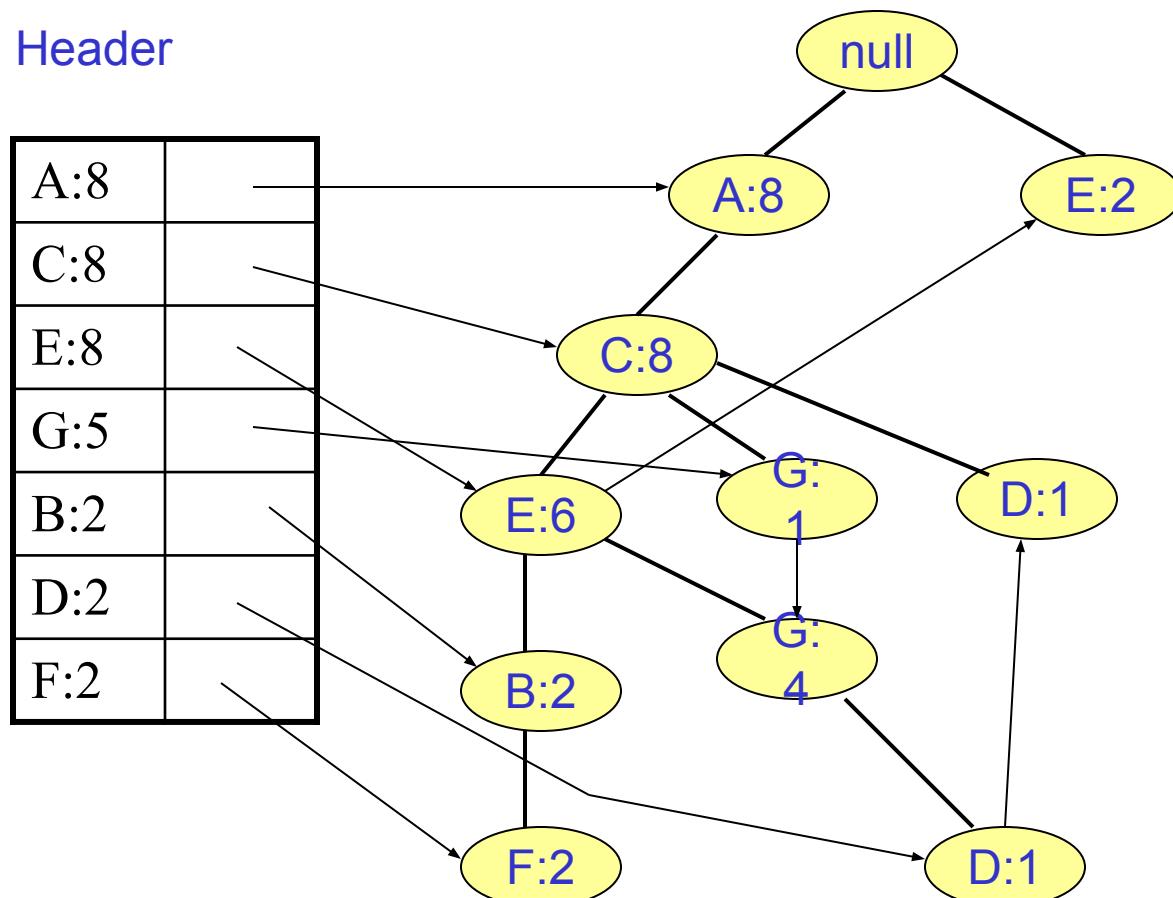
A C D

A C E G

**A C E G**

Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



# Conditional FP-Trees

Build the conditional FP-Tree for each of the items.

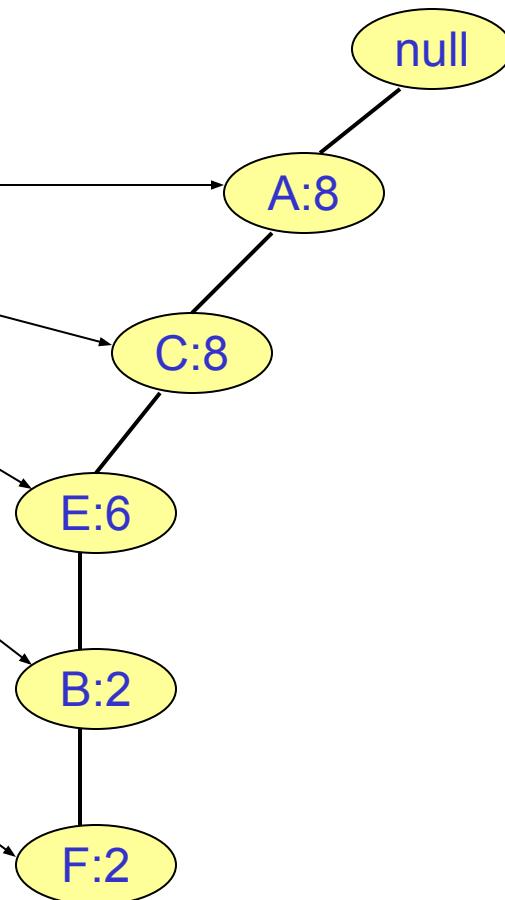
For this:

1. Find the paths containing on focus item. With those paths we build the conditional FP-Tree for the item.
2. Read again the tree to determine the new counts of the items along those paths. Build a new header.
3. Insert the paths in the conditional FP-Tree according to the new order.

# Conditional FP-Tree for F

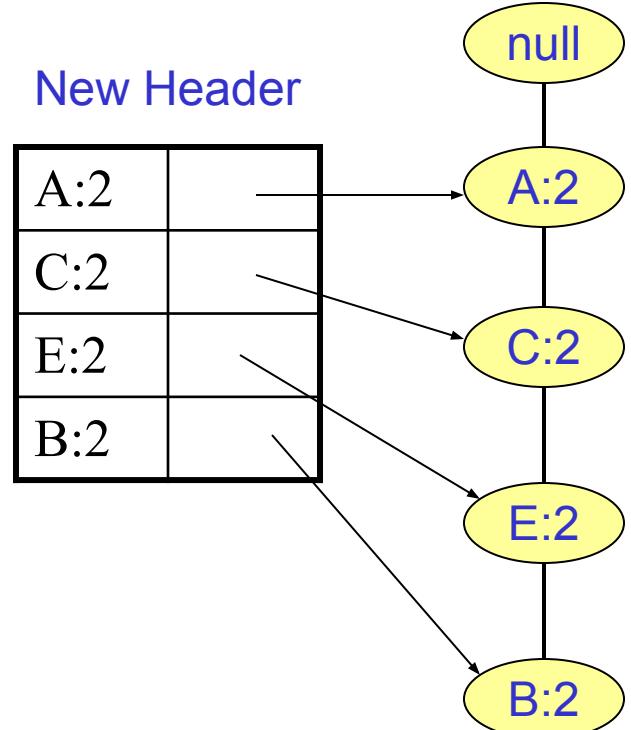
Header

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



New Header

A:2	
C:2	
E:2	
B:2	

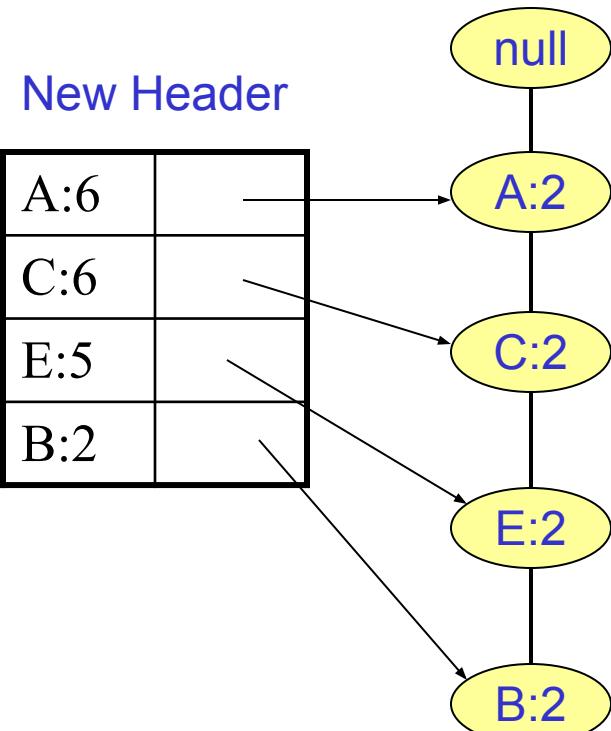


There is only a single path containing F

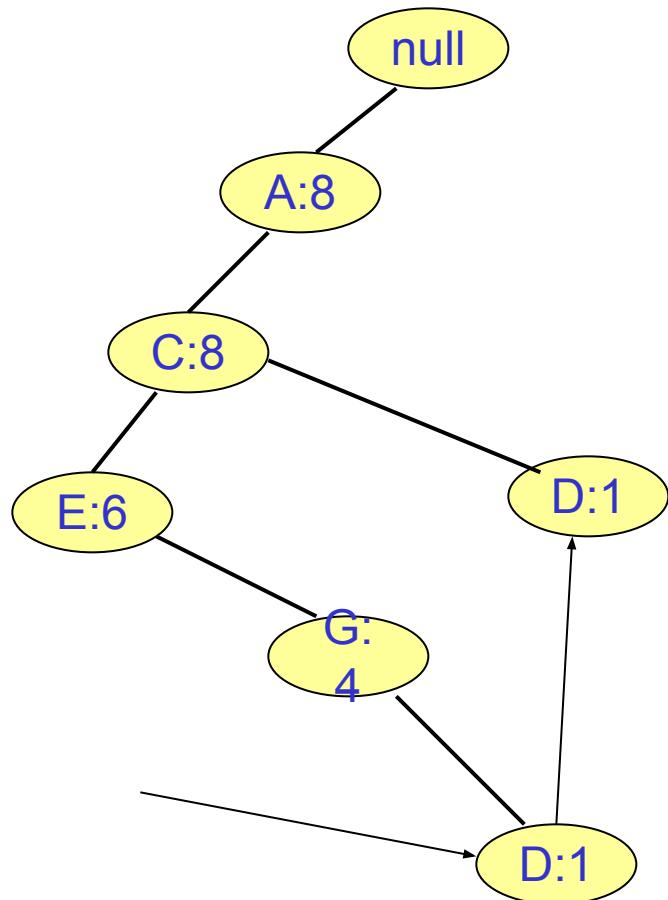
# Recursion

- We continue recursively on the conditional FP-Tree for F.
- However, when the tree is just a single path it is the **base case** for the recursion.
- So, we just produce all the subsets of the items on this path merged with F.

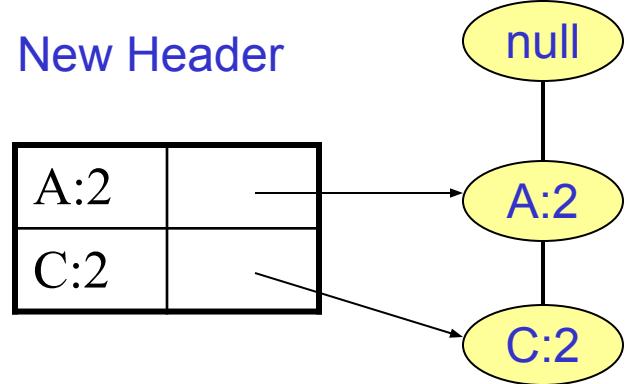
$\{F\}$   $\{A,F\}$   $\{C,F\}$   $\{E,F\}$   $\{B,F\}$   
 $\{A,C,F\}$ , ...,  
 $\{A,C,E,F\}$



# Conditional FP-Tree for D



Paths containing D after updating the counts



The other items are removed as infrequent.

The tree is just a single path; it is the **base case** for the recursion.

So, we just produce all the subsets of the items on this path merged with D.

{D} {A,D} {C,D} {A,C,D}

**Exercise:** Complete the example.

FP tree algorithm:

to identify frequent patterns in the area of Data Mining

AIM :

- To Draw a FP tree
- To identify frequent patterns

**Problem :** Find all frequent itemsets and frequent patterns in the following database using FP-growth algorithm. Take minimum support as 30%.

TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B, E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C

Table 1 - Snapshot of the Database

### **Step 1 - Calculate Minimum support**

First should calculate the minimum support count. Question says minimum support should be 30%. It calculate as follows:

Minimum support count( $30/100 * 8$ ) = **2.4**

As a result, 2.4 appears but to empower the easy calculation it can be rounded to to the ceiling value. Now,

Minimum support count is **ceiling**( $30/100 * 8$ ) = **3**

### **Step 2 - Find frequency of occurrence**

Now time to find the frequency of occurrence of each item in the Database table.

Item	Frequency
A	5
B	6
C	3
D	6
E	4

Table2 -Frequency of Occurrence

### Step 3 - Prioritize the items

Item B got the highest priority (**1**)

At the same time you have opportunity to drop the items which not fulfill the minimum support requirement.

E.g. If Database contain F which has frequency 1, then you can drop it.

Note : We have no items with MinSup

\*Some people display the frequent items using list instead of table. The frequent item list for the above table will be **B:6, D:6, A: 5, E:4, C: 3.**

### Step 4 -Order the items according to priority

As you see in the Table 3 new column added to the Table 1.

Last column indicate the ordered list

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B, E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Table 3 - New version of the Table 1

### Step 5 –Draw the FP-Tree

As a result of previous steps we got a ordered items table (Table 3). Now it's time to draw the FP tree.

### Row 1:

FP trees have 'null' node as the root node.

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E

Start with root node first and attach the items of the row 1 one by one respectively. (See the Figure 1) And write their occurrences in front of it.

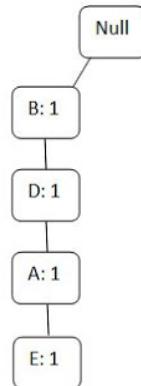


Figure 1- FP tree for Row 1

### Row 2:

Then update the above tree (Figure 1) by entering the items of row 2.

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C

Update the frequency count of repeated visit of nodes

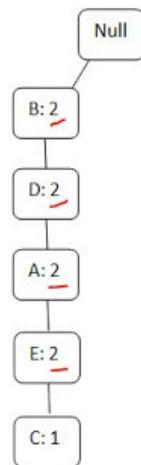


Figure 2- FP tree for Row 1,2

### Row 3:

	TID	Items	Ordered Items
In	1	E, A, D, B	B,D,A,E
	2	D, A, C, E, B	B,D,A,E,C
•	3	C, A, B, E	B,A,E,C

- overtaking D in existing path
- draw another A and connect it to B and then connect new E to that A and new C to new E. See Figure 3.

row 3 you have to visit B,A,E and C respectively.

Can't connect B to existing A

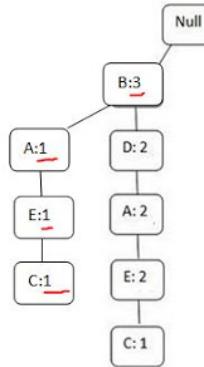


Figure 3 - After adding third row

**Row 4:**

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B, E	B,A,E,C
4	B, A, D	B,D,A
5	D	D

Then row 4 contain B,D,A. Now we can just rename the frequency of occurrences in the existing branch. As B:4,D,A:3.

**Row 5:**

In fifth raw have only item D. Now we have opportunity draw new branch from 'null' node. See Figure 4.

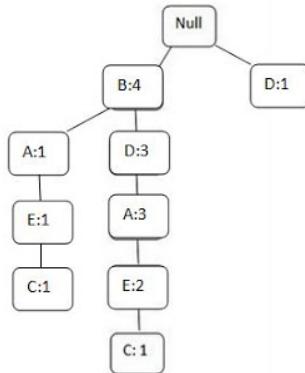


Figure 4- Connect D to null node

**Row 7:**

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B, E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Attach two new nodes A and E to the D node which hanging on the null node.  
Then mark D,A,E as D:2,A:1 and E:1.

**Row 8 :**

Attach new node C to B. Change the traverse times.(B:6,C:1)

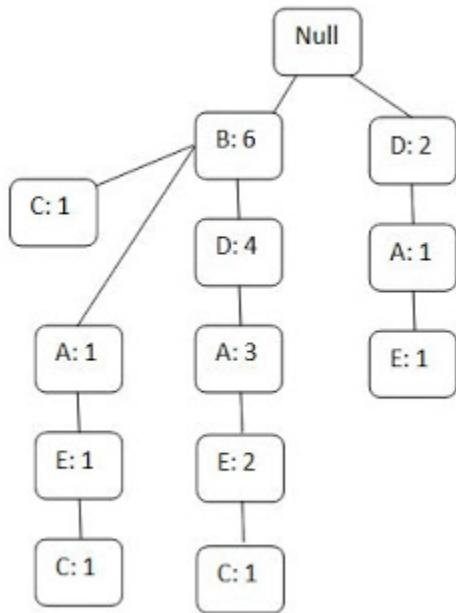


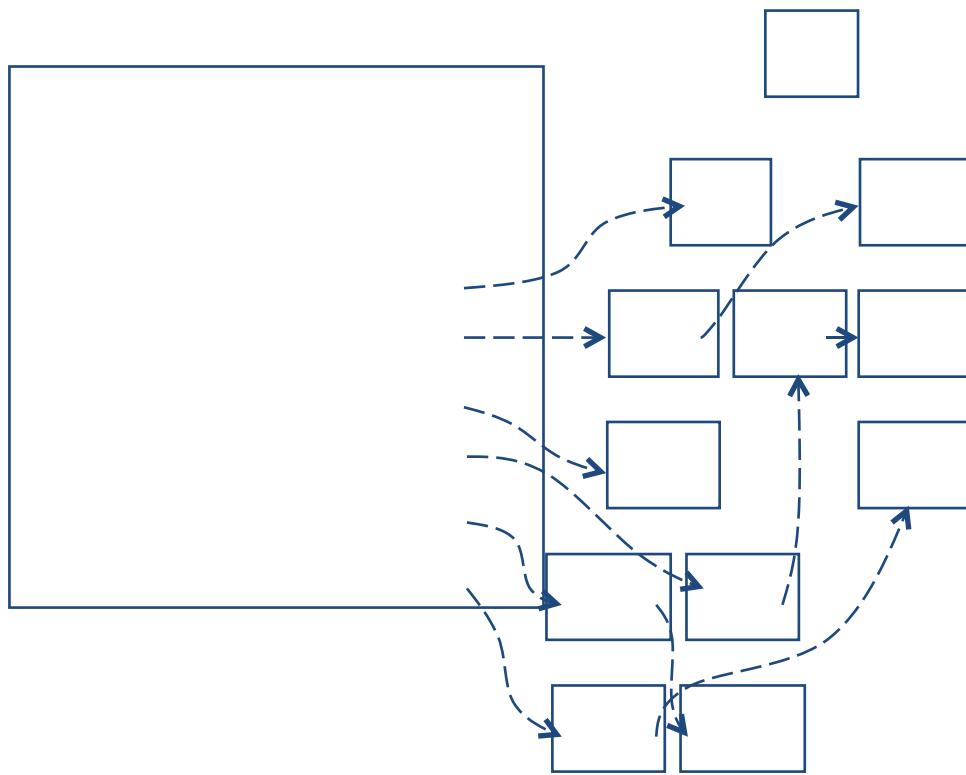
Figure 5 - Final FP tree

**Step 6 - Validation**

How we know is this correct?

Count the frequency of occurrence of each item of the FP tree and compare it with Table 2. If both counts equal, then it is positive point to indicate your tree is correct.

## Linked based DS



## How to identify frequent patterns from FP tree?

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B, E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Item	Frequency
A	5
B	6
C	3
D	6
E	4

Table2 -Frequency of Occurrence

B:6, D:6, A: 5, E:4, C: 3.

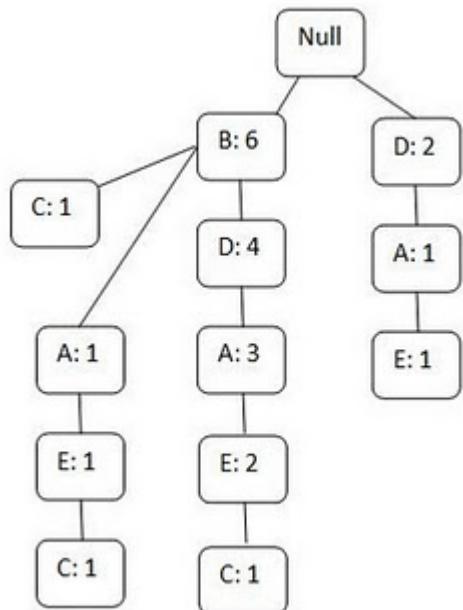


Figure 1

The Final FP tree created is as in Fig1.

**Minimum support count is 3.**

Orders list of Items of the DB in order of their frequency :

**B:6**  
**D:6**  
**A:5**  
**E:4**  
**C:3**

To observe the FPs we have to go through bottom to top, from C to B.

## FP using C

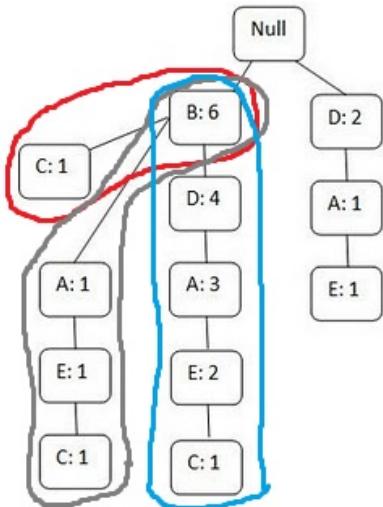


Figure 2

### **Step 1: Find the conditional pattern base for C:3.**

Here c.count = 3 which is mentioned as C:3

In Figure 2 there are 3 Cs with count = 1 for each.

Traverse bottom to top and get the branches which have Cs with the occurrence of C.

3 branches are derived

- **BDAE: 1 [in Blue]**
- B: 1 - [in Red]**
- BAE: 1 – [in Brown]**

These branches regard as Conditional Pattern Base for C. You have to note three facts.

1. Even we traverse bottom to top we write the branches in top to bottom manner.
2. C is not there.
3. 1 came after each branch to represent the frequency of occurrence of C in each branch.

Note: To ensure you correctly got all the occurrences of C in FP tree, add occurrences of C in each branch and compare with all the occurrences of C in FP tree; for this case **1+1+1 = 3** it is correct.

## Step 2: Find F-list from the Conditional Pattern Base of C.

The list of items are B:3,D:1,A:2,E:2

B:3 only eligible for F-list due (Minimum Support Count=3)

F-list = **B:3**

## Step 3: Draw the Conditional FP tree for C (figure 3)

When drawing the conditional FP tree you have to make sure to get Conditional Pattern Base as Ordered Items as described in or Data Table Table 3.

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B, E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Now we can find **C:3, BC:3** as identified FPs from tree.

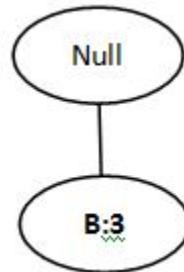


Figure 3

If you answer a question paper the answer will be  
Conditional pattern base for C:3  
**BDAE: 1; B: 1; BAE: 1**  
**F-list: B:3**

Conditional FP-tree for C (Draw Figure 3.)

Frequent patterns

**C:3, BC:3**

## FP using E

B:6

D:6

A:5

E:4

C:3

Step 1: Find Conditional Pattern Base for E:4.

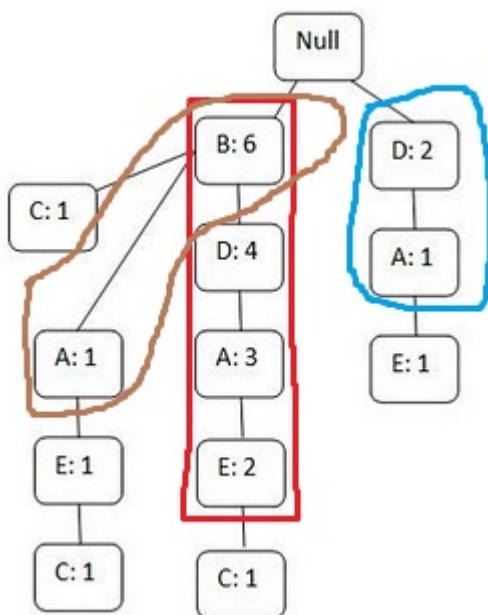


Figure 4

**BDA: 2** - Red line

**BA:1** - Brown line

**DA:1** - Blue line

Step 2: Prepare F-List

F-list =

**A:4**

**B:3**

**D:3**

Final Conditional Pattern Base = ABD : 2, AB: 1, AD: 1

Note:

- 3 comes to B by 2 Bs coming from BDA: 2 and 1 B coming from BA:1.
- And 4 As and 3 Ds also according to that manner.
- Why A appears before the B in F-list even B appears before previously? That have a reason. We got a list from this Conditional Pattern Base as B:3, D:3, A:4. But in F-list

they should appear according to their frequency of occurrences in Conditional Pattern Base. So A coming to front and others going to back. Then it is easy to use for creating conditional FP tree for E.

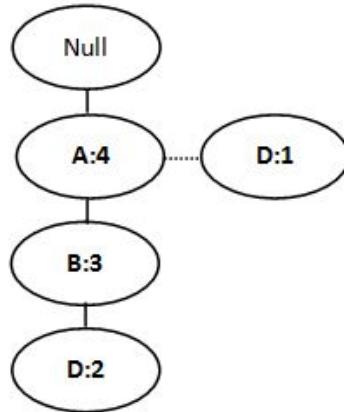


Figure 5

Repeat the same task for this tree (Conditional tree for E so E is silent)

Items in this FP tree are

A:4

B:3

D:3

Let's go bottom to top.

### DE

We are going to identify FPs for DE. Because of we use conditional FP tree for E, E comes to back.

Conditional pattern base for DE:3

A:1, AB:2

List of Items are L A:3, B:2

F-list: A:3 [B.count < min Supp]

Conditional FP-tree for DE : Figure 6

Frequent patterns

DE:3, ADE:3 [This final list of paeern should follow the original sequence of table 3]

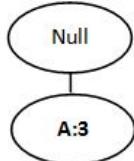


Figure 6

Same way

BE

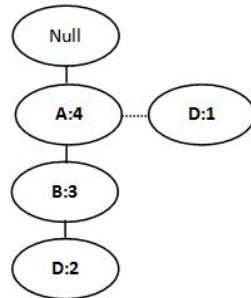


Figure 5

Conditional pattern base for BE:3

A:3

F-list: A:3

Conditional FP-tree for DE : Figure 6

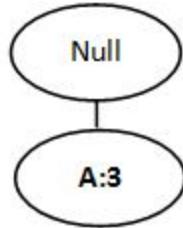


Figure 6

**Frequent patterns**

BE:3, ABE:3

AE (from fig 5)

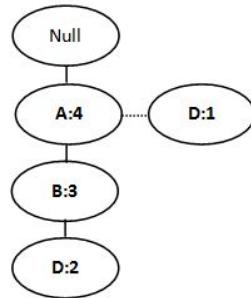


Figure 5

Conditional pattern base for AE:4

NULL

Conditional FP-tree for AE

NULL

**Frequent patterns**

**AE:4**

We find all FPs of conditional FP tree for E. Next step is remind the previous or main tree. And before leave E we can write E:4 also as a FP. Let's move to A.

**A**

B:6

D:6

**A:5**

E:4

C:3

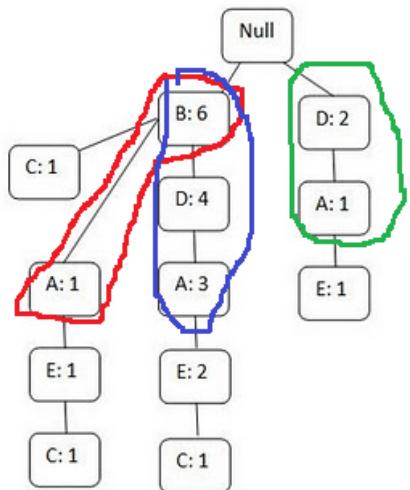


Figure 1

Conditional pattern base for A:5

BD: 3; B:1; D:1

F-list: B:4, D:4

Conditional FP-tree for A : Figure 7

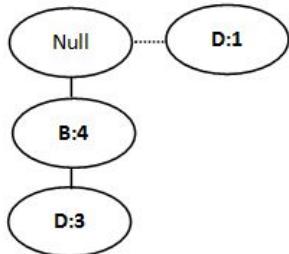


Figure 7

Then we will go on the conditional FP tree for A as like as previous trip.

**AD**

Conditional pattern base for AD:4

B:3

F-list :B:3

Conditional FP-tree for AD : Figure 3

**Frequent patterns**

**AD:4, ABD:3**

**AB**

Conditional pattern base for AB:4

NULL

**Frequent patterns**

**AB:4**

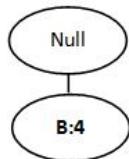


Figure 8

And also can get A:5 as a FP.

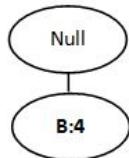


Figure 8

**D**

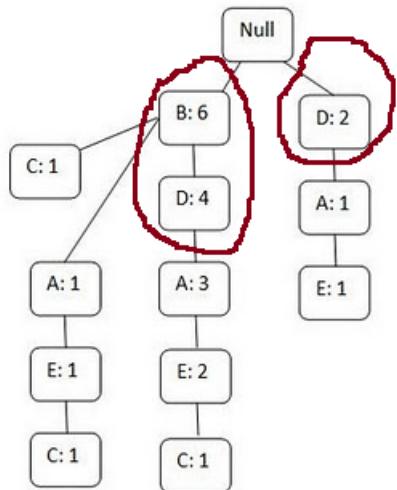
B:6

**D:6**

A:5

E:4

C:3



Conditional pattern base for D:6

B: 4

F-list : B:4

Conditional FP-tree for D : Figure 8

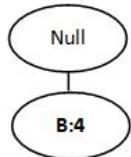


Figure 8

Frequent patterns

D:6, BD:4

Then last item, B.

**B**

**B:6**

**D:6**

**A:5**

**E:4**

**C:3**

Conditional pattern base for B:6

NULL

Conditional FP-tree for B

NULL

Frequent patterns

B:6

The identified frequent itemsets from the Database with minimum support as 30% are  
**C:3, BC:3, DE:3, ADE:3 , BE:3, ABE:3, AE:4, E:4, AD:4, ABD:3, AB:4, A:5, D:6, BD:4, B:6.**

# Association Rules

---

Data mining and data warehousing

Association Rule Mining

# Topics

- Basic concepts of Association Rules
- Rule strength measures
- Basic Algorithms
  - Apriori Algorithm
  - FP-Growth Algorithm
  - Other Approaches
  - Interestingness Measures
  - Sequential Pattern Mining
- Summary

# Association rule mining

## ■ Motivation: Finding inherent regularities in data

- What products were often purchased together?— Clothes and Milk !
- What are the subsequent purchases after buying a PC?
- What kinds of DNA are sensitive to new drug?
- Can we automatically recommend next web document?

## ■ Applications

- Basket data analysis, Cross-marketing, Rack arrangement, Sale campaign analysis
- DNA sequence analysis
- Web log (click stream) analysis
- planning public services (education, health, transport, funds) as well as public business(for setup new factories, shopping malls or banks and even marketing particular products) from census data.

# What is Association Rule?

- Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository.
- An example of an association rule would be "If a customer buys a packet of bread, he is 80% likely to also purchase milk."

bread → milk

# Association rule mining

- Frequent pattern
  - A pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal et al. in 1993 in the context of **frequent itemsets** and **association rule mining**
- An important data mining model studied extensively

# Association rule mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction
- Initially used for **Market Basket Analysis** to find how items purchased by customers are related  
Bread → Milk [Sup = 5%, Conf = 70%]

# The model: Data

- $I = \{i_1, i_2, \dots, i_m\}$ : a set of *items*
- Transaction  $t$ : a set of items, and  $t \subseteq I$
- Transaction Database  $T$ : a set of transactions  
 $T = \{t_1, t_2, \dots, t_n\}$

# Transaction data: Supermarket data

- Market basket transactions:

- t1: {bread, cheese, milk}

- t2: {apple, cake, salt, yogurt}

- ...      ...

- tn: {biscuit, cake, milk}

- Concepts:

- An *item*: an item/article in a basket
  - $I$ : the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

# Transaction data: a set of documents

- **Text document data set, each document is treated as a “bag” of keywords**

doc1: Student, Teach, School

doc2: Student, School

doc3: Teach, School, City, Game

doc4: Baseball, Basketball

doc5: Basketball, Player, Spectator

doc6: Baseball, Coach, Game, Team

doc7: Basketball, Team, City, Game

- **Web page data set**

Session1: PageA.html, PageB.html, PageC.html

Session1: PageC.html, PageD.html, PageE.html

Session1: PageA.html, PageC.html, PageD.html

# The model: Rules

- A transaction  $t$  contains  $X$ , a set of items (**itemset**) in  $I$ , if  $X \subseteq t$
- An **association rule** is an implication of the form:  
$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$
- An **itemset** is a set of items
  - E.g.,  $X = \{\text{milk, bread, cereal}\}$  is an itemset
- A  **$k$ -itemset** is an itemset with  $k$  items
  - E.g.,  $\{\text{milk, bread}\}$  is a 2-itemset  
 $\{\text{milk, bread, cereal}\}$  is a 3-itemset

# Rule Strength Measures

- An association rule is a pattern that states when  $X$  occurs,  $Y$  occurs with certain probability
  - Support
  - Confidence

# Support

- This measure gives an idea of how frequent an *itemset* is in all the transactions.
- $itemset1 = \{\text{bread}\}$
- $itemset2 = \{\text{shampoo}\}$ .
  - $t(\text{bread}).\text{count} >> t(\text{shampoo}).\text{count}$
  - $\text{Support}(Itemset1) > \text{support}(Itemset2)$
- $itemset1 = \{\text{bread, butter}\}$
- $itemset2 = \{\text{bread, shampoo}\}$ .
  - $T(\text{bread, butter}).\text{count} >> t(\text{bread, shampoo}).\text{count}$
  - $\text{Support}(Itemset1) > \text{support}(Itemset2)$

# Absolute and Relative Support

- $T_1 = \{A, A, C\}$
- $T_2 = \{A, X\}$
- What is the support of A ? Is it 3 or 2 ?
  
- **absolute support** of A,
  - i.e. the absolute number of transactions which contains A, is 2
- **relative support** of A,
  - i.e. the relative number of transactions which contains A, is  $2/2=1$

# Support

- Mathematically,
  - support : the fraction of the total number of transactions in which the itemset occurs.
$$Support(\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both } X \text{ and } Y}{\text{Total number of transactions}}$$
- support helps to identify the rules to be considered or not for further analysis
  - E.g, to consider only the itemsets which occur at least 50 times out of a total of 10,000 transactions
  - i.e. support = 0.005.
- *Itemset* with very low support do not have enough information on the among the items contained in the set
  - So no conclusions can be drawn from such a rule.

# Support and Confidence

## ■ Support

- The rule holds with support  $sup$  in  $T$  (the transaction data set having  $n$  transactions) if  $sup\%$  of transactions contain  $X \cup Y$
- $sup = \Pr(X \cup Y)$

$$sup = \frac{(X \cup Y).count}{n}$$

- Relative Support
- The frequency count of an itemset  $X \cup Y$ , denoted by  $(X \cup Y).count$ , in a data set  $T$  is the number of transactions
  - Count/Absolute Support

# Logic behind Support

- Logic of the support calculation is
  - to consider only item(sets) which appear frequently enough **in *different* transactions**
  - to be sure that the resulting rules are based on actual patterns
    - not appear due to chance (i.e. the strange behavior of just a few customers).
  - To make predictions about the likes/dislikes of future customers.

# Logic behind Support

- If a pattern is based only on two customers, the applicability is ... questionable.
- E.g.
  - Customer C1 bought A a thousand times (once),
  - Customer C2 bought it just to try it out
  - Other 1000 customers (C3..C1002) visited the store but none bought A
- Absolute support for A = 2, not 1001
- Relative support =  $2/1003 = 0.00192$   
not  $1001/1003 = 0.9990$
-

# Confidence

- defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents.
- E.g. To answer the question
  - of all the transactions containing say, {Potato Chips}, how many also had {Milk} on them?
  - $\{ \text{Potato Chips} \} \rightarrow \{ \text{Milk} \}$  should be a high or low confidence rule.
- It is the conditional probability of occurrence of consequent given the antecedent.

$$\text{Confidence}(\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both } X \text{ and } Y}{\text{Transactions containing } X}$$

# Rule strength measures

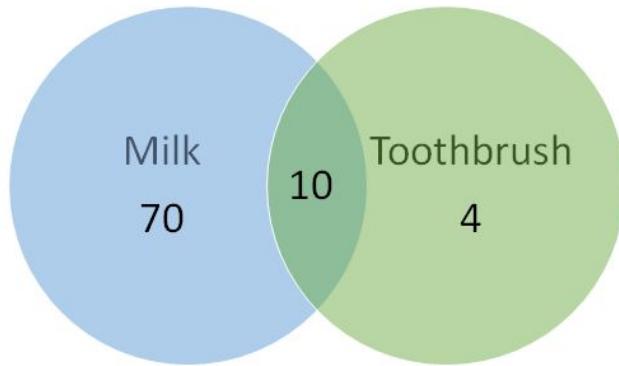
## ■ Confidence

- The rule holds in  $T$  with **confidence**  $conf$  if % of transactions that contain  $X$  also contain  $Y$
- $X \square Y$
- $conf = \Pr(Y | X)$   
$$confidence = \frac{(X \cup Y).count}{X.count}$$

$$confidence = \frac{Support(X \cup Y)}{Support(X)}$$

- E.g Confidence of  $\{\text{Butter}\} \rightarrow \{\text{Bread}\}$ 
  - fraction of transactions having butter also had bread
  - Very high i.e. a value close to 1
- $\{\text{Yogurt}\} \rightarrow \{\text{Milk}\}$ ? High again
- $\{\text{Toothbrush}\} \rightarrow \{\text{Milk}\}$ ? Not so sure?
  - Confidence for this rule will also be high since  $\{\text{Milk}\}$  is such a frequent itemset and would be present in every other transaction.
  - What about  $\{\text{Milk}\} \rightarrow \{\text{Toothbrush}\}$ ? Not so sure?
- *It does not matter what you have in the antecedent (X) for such a frequent consequent.*
- *The confidence for an association rule having a very frequent consequent (Y) will always be high.*

# Confidence : Limitation



Total transactions = 100  
both milk and toothbrush = 10  
milk but no toothbrush = 70  
toothbrush but no milk = 4.

- Confidence for
  - $\{\text{Toothbrush}\} \rightarrow \{\text{Milk}\}$
  - $10/(10+4) = 0.7$
- Looks like a high confidence value.
- But we know intuitively that these two products have a weak association and there is something misleading about this high confidence value.
- *Lift* is introduced to overcome this challenge.

# Lift

- Lift controls for the *support* (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}.
- *Lift* is a very literal term given to this measure.
- Think of it as the \*lift\* that {X} provides to our confidence for having {Y} on the cart.
- To rephrase, *lift* is the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}.

# Lift

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions \ containing \ both \ X \ and \ Y) / (Transactions \ containing \ X)}{Fraction \ of \ transactions \ containing \ Y}$$

- Lift = Confidence(X ⊑ Y) / Y.count
  - In cases where {X} actually leads to {Y} on the cart, value of lift will be greater than 1.
- {Toothbrush} → {Milk} rule.
  - Probability of having milk on the cart with the knowledge that toothbrush is present (Confidence)
  - $10/(10+4) = 0.7$
- {Milk}
  - consider the probability of having milk on the cart without any knowledge about toothbrush  $=: 80/100 = 0.8$

# Lift

- having toothbrush on the cart actually reduces the probability of having milk on the cart to 0.7 from 0.8!
- This will be a lift of  $0.7/0.8 = 0.87$ .
- Now that's more like the real picture.
  - A value of lift less than 1 shows that having toothbrush on the cart does not increase the chances of occurrence of milk on the cart in spite of the rule showing a high confidence value.
  - If  $\text{lift} > 1$  indicates high association between  $\{Y\}$  and  $\{X\}$ .
  - More the value of lift, greater are the chances of preference to buy  $\{Y\}$  if the customer has already bought  $\{X\}$ .
  - *Lift* is the measure that will help store managers to decide product placements on aisle.

# Problem of Association

- Once quantify the importance of association of products within an itemset, the next step is to generate rules from the entire list of items and identify the most important ones.
  - E.g Supermarkets will have thousands of different products in store
  - Just 10 products may lead to 57000 rules!!
  - this number increases exponentially with the increase in number of items.
  - Finding lift values for each of these will get computationally very very expensive.
  - How to deal with this problem?
  - How to come up with a set of most important association rules to be considered?
  - *Apriori algorithm* comes to our rescue for this.

# Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf)
- **Key Features**
  - **Completeness:** find all rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds
  - Mining with data on **hard disk** (not in memory)

# An example

- Transaction data
- Assume:

$\text{minsup} = 30\%$

$\text{minconf} = 80\%$

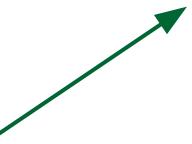
- An example frequent *itemset*: {Cake, Clothes, Milk}  
[sup = 3/7]
- Association rules from the itemset:

Clothes  $\rightarrow$  Milk, Cake [sup = 3/7, conf = 3/3]

...

...

Clothes, Cake  $\rightarrow$  Milk, [sup = 3/7, conf = 3/3]



t1: Bread, Cake, Milk  
t2: Bread, Cheese  
t3: Cheese, Boots  
t4: Bread, Cake, Cheese  
t5: Bread, Cake, Clothes, Cheese, Milk  
t6: Cake, Clothes, Milk  
t7: Cake, Milk, Clothes

# Assumption

- A simplistic view of shopping baskets transactions
  - Some important information not considered e.g.
    - The quantity of each item purchased
    - The price paid
- Assume all data are categorical
  - Examples:
    - Item Purchased or not ?
    - ID numbers, eye color {brown, black, etc.}, zip codes
    - Height in {tall, medium, short}

# Many mining algorithms

- A large number of them!!
- Use of different strategies and data structures
- Resulting sets of rules are all the same
- Computational efficiencies and memory requirements may be different

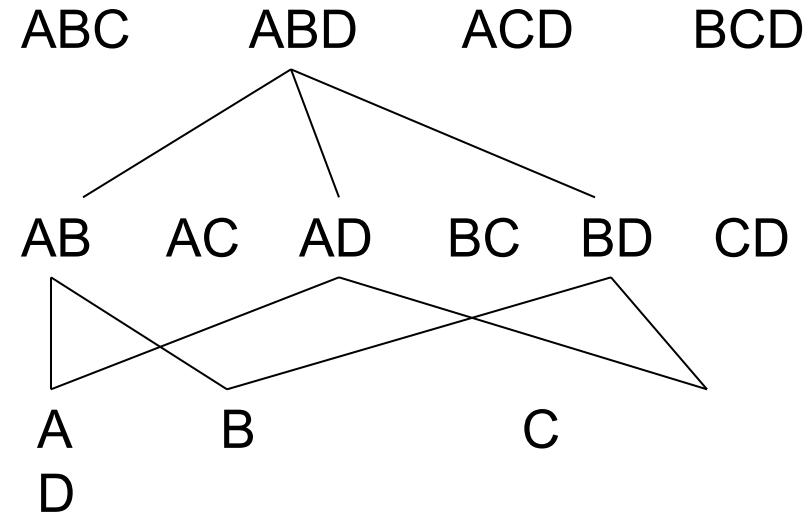
# The Apriori algorithm

- **The best known algorithm**
- **Two steps:**
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets)
  - Use frequent itemsets to **generate rules**
- E.g., a frequent itemset  
    {Cake, Clothes, Milk}     [ $\text{sup} = 3/7$ ]  
    and one rule from the frequent itemset  
    Clothes  $\rightarrow$  Milk, Cake    [ $\text{sup} = 3/7$ ,  $\text{conf} = 3/3$ ]

# Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is  $\geq \text{minsup}$
- Key idea
  - The apriori property (downward closure property)
    - Any subsets of a frequent itemset are also frequent itemsets

If **{beer, diaper, nuts}** is frequent,  
so is **{beer, diaper}**  
i.e., every transaction having  
**{beer, diaper, nuts}**  
also contains **{beer, diaper}**



# Anti-monotone property of support

- All subsets of a frequent itemset must also be frequent.
- $\{\text{Bread, Egg}\}.count \geq \{\text{Bread, Egg, Vegetables}\}.count$
- If  $\text{sup}(\{\text{Bread, Egg, Vegetables}\}) (30/100) = 0.3 > \text{minsup}$ , then  $\{\text{Bread, Egg}\} \geq 0.3$
- This is called the **anti-monotone property of support** where if we drop out an item from an itemset, support value of new itemset generated will either be the same or will increase.

# The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on
  - In each iteration  $k$ , only consider itemsets that contain some  $k-1$  frequent itemset
- Find frequent itemsets of size 1:  $F_1$
- For  $k = 2$ 
  - $C_k$  = candidates of size  $k$ : those itemsets of size  $k$  that could be frequent, given  $F_{k-1}$
  - $F_k$  = those itemsets that are actually frequent,  $F_k \subseteq C_k$  (need to scan the database once)

# The Apriori Algorithm—An Example

Database T

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

$$\text{Sup}_{\min} = 2$$

# The Apriori Algorithm—An Example

$$\text{Sup}_{\min} = 2$$

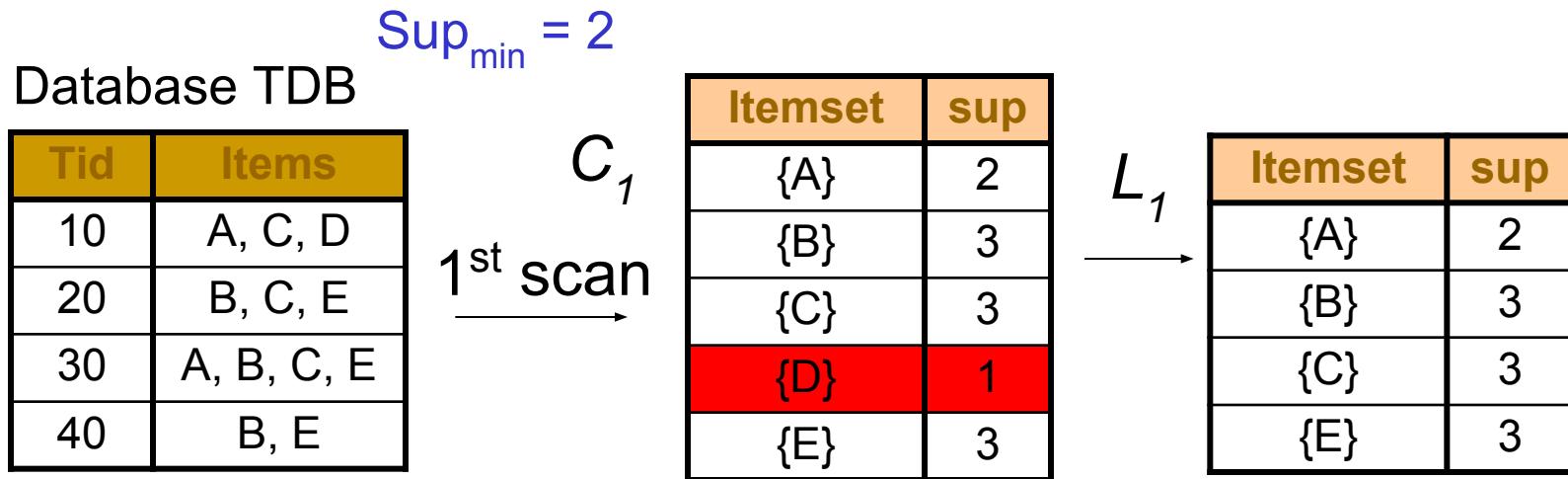
Database T

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

$C_1$   
1<sup>st</sup> scan

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

# The Apriori Algorithm—An Example



# The Apriori Algorithm—An Example

$$\text{Sup}_{\min} = 2$$

Database TDB

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

$C_1$   
1<sup>st</sup> scan

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

$L_1$   
→

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

↔  
 $C_2$

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

# The Apriori Algorithm—An Example

$$\text{Sup}_{\min} = 2$$

Database TDB

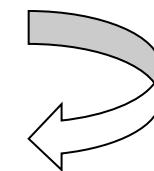
Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$C_1$   
1<sup>st</sup> scan

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

$L_1$   
→

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



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

2<sup>nd</sup> scan  
←

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

$C_2$

# The Apriori Algorithm—An Example

$$\text{Sup}_{\min} = 2$$

Database TDB

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

$C_1$

1<sup>st</sup> scan →

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

$L_1$

→

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

$C_2$

2<sup>nd</sup> scan ←

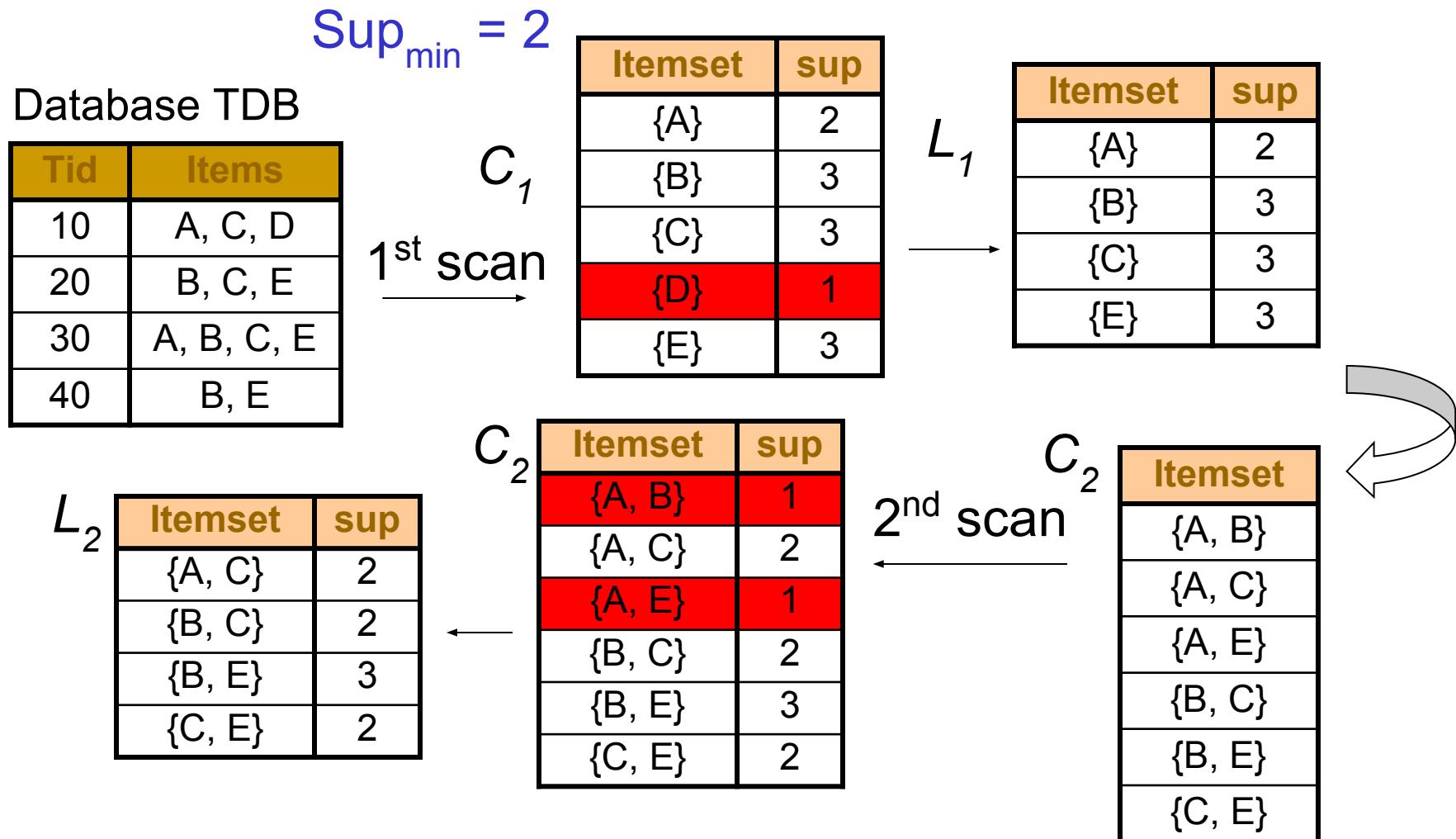
Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

$C_2$

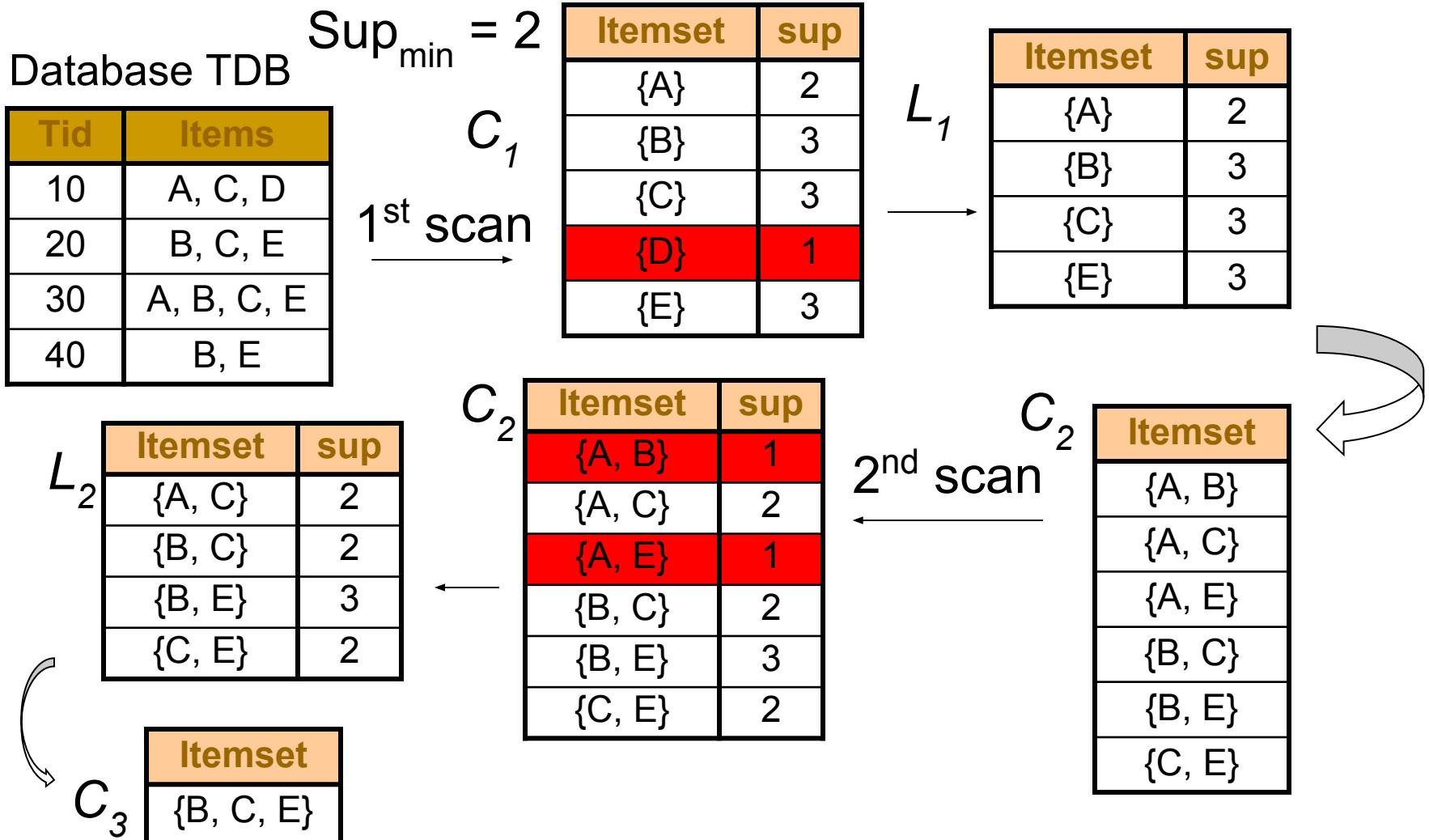
←

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

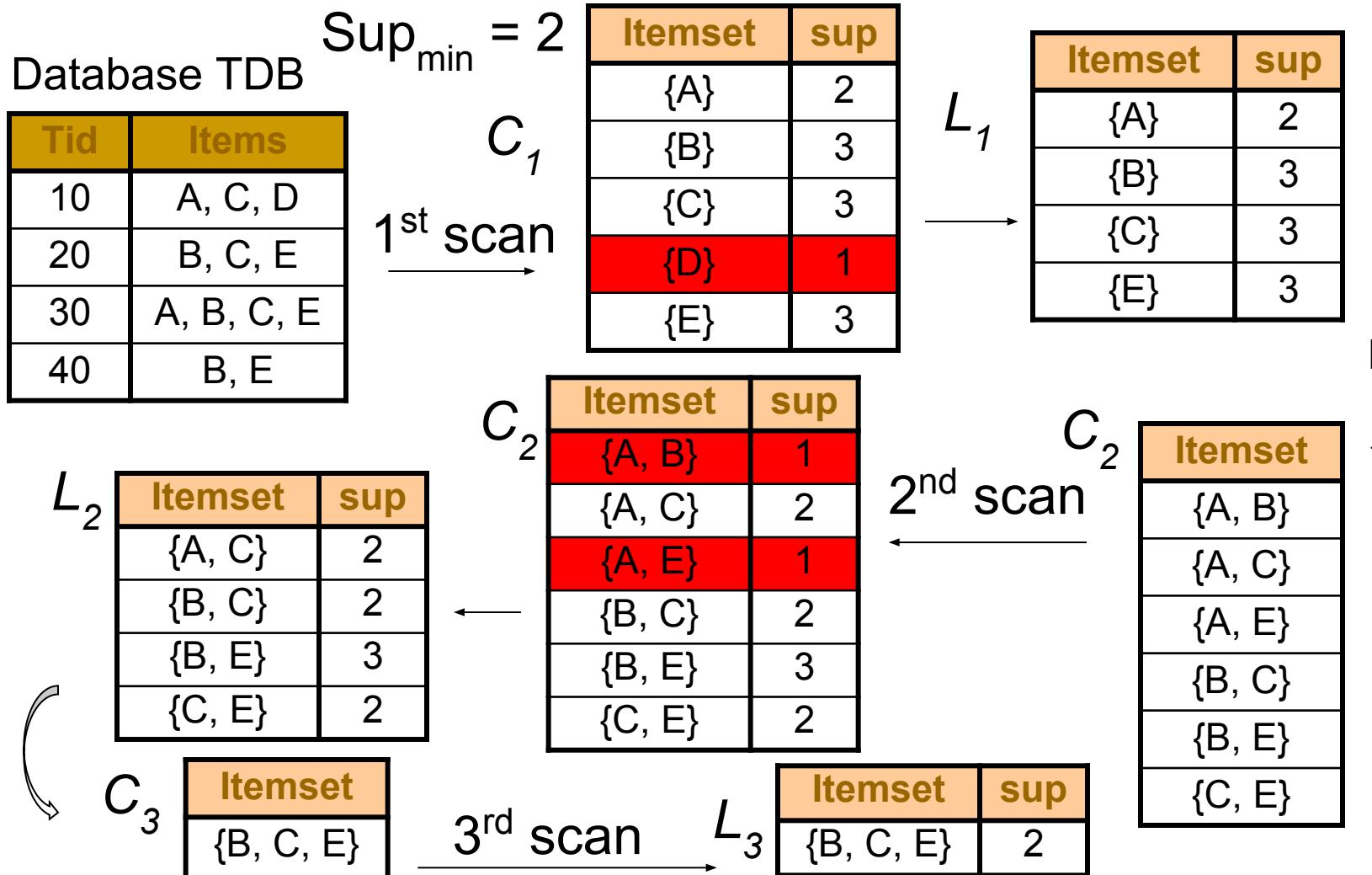
# The Apriori Algorithm—An Example



# The Apriori Algorithm—An Example



# The Apriori Algorithm—An Example



# The Apriori Algorithm

$C_k$ : Candidate itemset of size k

$F_k$ : frequent itemset of size k

## Algorithm Apriori( $T$ )

```
 $C_1 \leftarrow \text{init-pass}(T);$ 
 $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq \text{minsup}\}; \quad // n: \text{no. of transactions in } T$ 
for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do
     $C_k \leftarrow \text{candidate-gen}(F_{k-1});$ 
    for each transaction  $t \in T$  do
        for each candidate  $c \in C_k$  do
            if c is contained in  $t$  then
                 $c.\text{count}++;$ 
            end
        end
    end
     $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{minsup}\}$ 
end
return  $F \leftarrow \bigcup_k F_k;$ 
```

# Apriori candidate generation

- Function takes  $F_{k-1}$  and returns a superset (called the candidates) of the set of all frequent  $k$ -itemsets
- It has two steps
  - *join step*: Generate all possible candidate itemsets  $C_k$  of length  $k$  (**In sorted order**)
  - *prune step*: Remove those candidates in  $C_k$  that cannot be frequent

# Implementation of Apriori

## ■ Example of Candidate-generation

- $L_3 = \{abc, abd, acd, ace, bcd\}$
- Self-joining:  $L_3 * L_3$ 
  - $abcd$  from  $abc$  and  $abd$
  - $acde$  from  $acd$  and  $ace$
- Pruning:
  - $acde$  is removed because  $ade$  is not in  $L_3$
- $C_4 = \{abcd\}$

# Assignment example

1. **2-Itemset**= $\{\{A, C\}, \{B, C\}, \{B, E\}, \{C, E\}\}$ 
  - **3-itemset ?**
  
2. **2-Itemset**= $\{\{I1,I2\}, \{I1,I3\}, \{I1,I5\}, \{I2,I3\}, \{I2,I4\}, \{I2,I5\}\}$ 
  - **3-itemset ?**

# Candidate-gen function

**Function** candidate-gen( $F_{k-1}$ )

$C_k \leftarrow \emptyset;$

**forall**  $f_1, f_2 \in F_{k-1}$

with  $f_1 = \{i_1, \dots, i_{k-2}, i_{k-1}\}$

and  $f_2 = \{i_1, \dots, i_{k-2}, i'_{k-1}\}$

and  $i_{k-1} < i'_{k-1}$  **do**

$c \leftarrow \{i_1, \dots, i_{k-1}, i'_{k-1}\};$  // join  $f_1$  and  $f_2$

$C_k \leftarrow C_k \cup \{c\};$

**for** each  $(k-1)$ -subset  $s$  of  $c$  **do**

**if** ( $s \notin F_{k-1}$ ) **then**

    delete  $c$  from  $C_k;$  // prune

**end**

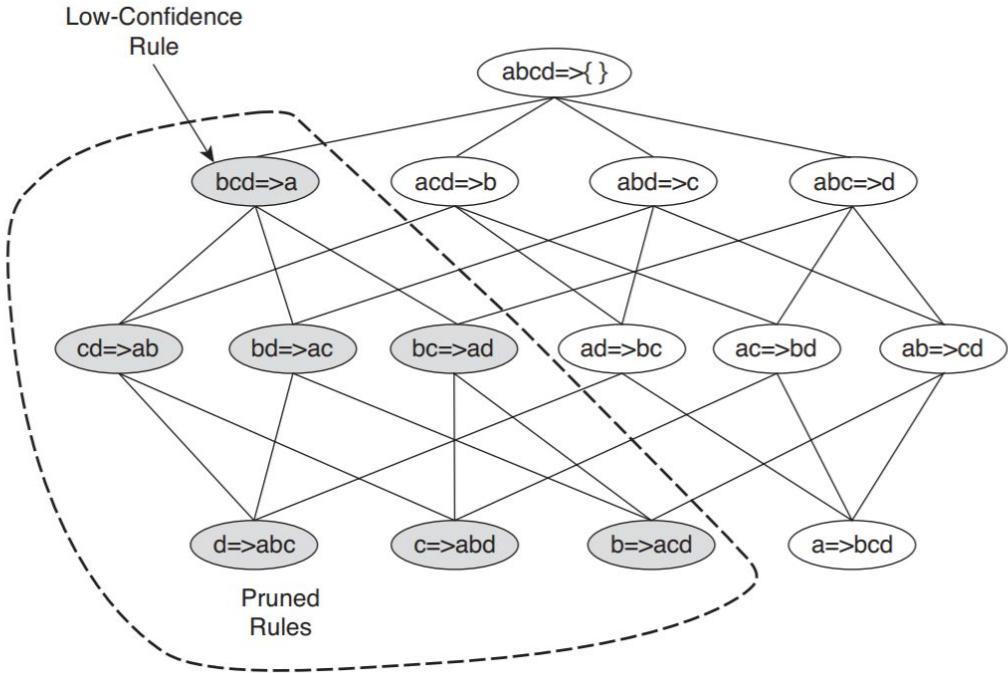
**end**

**return**  $C_k;$

# Comments on Confidence

- Note :
  - support of all the rules generated from same itemset remains the same
  - difference occurs only in the denominator calculation of confidence.
  - As number of items in X decrease, support{X} increases (as follows from the anti-monotone property of support) and hence the confidence value decreases.
- An intuitive explanation
  - $F1 = \{(butter), (egg, milk, bread)\}$
  - $F2 = \{(milk, butter, bread), (egg)\}$
  - Butter.count >> {(milk, butter, bread).count}
- So  $\text{conf}(F1) < \text{conf}(F2)$

# Rule pruning



- start with a frequent itemset {a,b,c,d}
- start forming rules with just one consequent.
- Remove the rules failing to satisfy the minconf condition.
- start forming rules using a combination of consequents from the remaining ones.
- Keep repeating until only one item is left on antecedent. This process has to be done for all frequent itemsets.

## Step 2: Generating rules from frequent itemsets

- Frequent itemsets  $\neq$  association rules
- For each frequent itemset  $X$ ,  
For each proper nonempty subset  $A$  of  $X$ ,
  - Let  $B = X - A$
  - $A \rightarrow B$  is an association rule if
    - $\text{Confidence}(A \rightarrow B) \geq \text{minconf}$ ,
    - $\text{support}(A \rightarrow B) = \text{support}(A \cup B) = \text{support}(X)$
    - $\text{confidence}(A \rightarrow B) = \text{support}(A \cup B) / \text{support}(A)$

# Generating Rules: an example

- Suppose  $\{2,3,4\}$  is frequent, with sup=50%
  - Proper nonempty subsets:  $\{2,3\}$ ,  $\{2,4\}$ ,  $\{3,4\}$ ,  $\{2\}$ ,  $\{3\}$ ,  $\{4\}$ , with sup=50%, 50%, 75%, 75%, 75%, 75% respectively
  - These generate these association rules:
    - $2,3 \rightarrow 4$ , confidence=100%
    - $2,4 \rightarrow 3$ , confidence=100%
    - $3,4 \rightarrow 2$ , confidence=67%
    - $2 \rightarrow 3,4$ , confidence=67%
    - $3 \rightarrow 2,4$ , confidence=67%
    - $4 \rightarrow 2,3$ , confidence=67%
    - All rules have support = 50%

# Generating Rules: summary

- To recap, in order to obtain  $A \rightarrow B$ , we need to have  $\text{support}(A \cup B)$  and  $\text{support}(A)$
- All the required information for confidence computation has already been recorded in itemset generation
  - No need to see the data  $T$  any more
- This step is not as time-consuming as frequent itemsets generation

Transaction id	Items
t1	{1, 2, 4, 5}
t2	{2, 3, 5}
t3	{1, 2, 4, 5}
t4	{1, 2, 3, 5}
t5	{1, 2, 3, 4, 5}
t6	{2, 3, 4}

■ By applying the algorithm with  $minsup = 0.5$ ,  $minconf = 0.9$  and  $minlift = 1$

rule 0: 4 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0  
 rule 1: 3 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0  
 rule 2: 1 ==> 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2  
 rule 3: 1 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0  
 rule 4: 5 ==> 2 support : 0.833(5/6) confidence : 1.0 lift : 1.0  
 rule 5: 4 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0  
 rule 6: 1 4 ==> 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.2  
 rule 7: 4 5 ==> 1 support : 0.5 (3/6) confidence : 1.0 lift : 1.5  
 rule 8: 1 4 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0  
 rule 9: 3 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0  
 rule 10: 1 5 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0  
 rule 11: 1 2 ==> 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2  
 rule 12: 1 ==> 2 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2  
 rule 13: 1 4 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0  
 rule 14: 1 2 4 ==> 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.2  
 rule 15: 2 4 5 ==> 1 support : 0.5 (3/6) confidence : 1.0 lift : 1.5  
 rule 16: 4 5 ==> 1 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.5  
 rule 17: 1 4 ==> 2 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.5

# For More example

- [https://www.digitalvidya.com/blog/apriori-algo\\_rithms-in-data-mining/](https://www.digitalvidya.com/blog/apriori-algo_rithms-in-data-mining/) [How to prepare the input frequency table?]
- [https://www.softwaretestinghelp.com/apriori-a\\_lgorithm/](https://www.softwaretestinghelp.com/apriori-a_lgorithm/)
- <https://www.geeksforgeeks.org/apriori-algorit hm/>

# Assignment Exercise: 1

- A database has five transactions.

Let  $\min sup = 60\%$  and  $\min con f = 80\%$ .

**TID    items bought**

T100 {M, O, N, K, E, Y}

T200 {D, O, N, K, E, Y}

T300 {M, A, K, E}

T400 {M, U, C, K, Y}

T500 {C, O, O, K, I ,E}

Find all frequent itemsets using Apriori.

# Apriori Algorithm

Seems to be very expensive

- Breadth-first (Level-wise) search
- If,  $K$  = the size of the largest itemset then makes at most  $K$  passes over data
- Very simple and fast
  - Under some conditions, all rules can be found in linear time
- Scale up to large data sets

# Apriori Algorithm

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
    - The number of frequent itemsets to be generated is sensitive to the minsup threshold
    - When minsup is low, there exist potentially an exponential number of frequent itemsets
    - Example:
      - $10^4$  frequent 1-itemsets, generate more than  $10^7$  candidate 2-itemsets
      - To discover a frequent pattern of size 100, such as  $\{a_1, \dots, a_{100}\}$ 
        - Generated candidates  $2^{100} - 1 = (\text{Approx.}) 10^{30}$
    - Tedious workload of support counting for candidates

# Improve Efficiency of Aptiori

## ■ Hash-Based Technique:

- hash-based structure (hash table) is used for managing the k-itemsets and its corresponding count.

## ■ Transaction Reduction:

- reduces the number of transactions scanning in iterations.
- E.g. The transactions which do not contain frequent items are marked or removed.(Preprocessing)

## ■ Partitioning:

- It is proven in some cases that for any itemset to be potentially frequent in the database, it should be frequent in at least one of the partitions of the database.
- only two database scans to mine the frequent itemsets.
- (Imbalance partitioning is an issue)

# Improve Efficiency of Aptiori

- **Sampling:**
  - picks a random sample S from Database D
  - searches for frequent itemset in S.
  - It may be possible to lose a global frequent itemset. This can be reduced by lowering the min\_sup.
- **Dynamic Itemset Counting:**
  - add new candidate itemsets at any marked start point of the database during the scanning of the database.

# Apriori Algorithm

- Reducing Complexity of Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

# Mining Frequent Patterns without Candidate Generation ???

# Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- The FP-Growth Approach given by J. Han, J. Pei, and Y. Yin, SIGMOD' 00
  - Depth-first search
  - Avoid explicit candidate generation

# FPGrowth Approach

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - Highly condensed, but complete for frequent pattern mining
  - Avoid costly database scans
- An efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones called conditional databases
  - Avoid candidate generation: sub-database mining only!

# Example

- **Refer the Document**

# The FP-Growth Mining Method

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

# FP-Growth Algorithm

1. The FP-tree is constructed in the following steps:

- (a) Scan the transaction database  $D$  once. Collect  $F$ , the set of frequent items, and their support counts. Sort  $F$  in support count descending order as  $L$ , the list of frequent items.
- (b) Create the root of an FP-tree, and label it as “null.” For each transaction  $Trans$  in  $D$  do the following. Select and sort the frequent items in  $Trans$  according to the order of  $L$ . Let the sorted frequent item list in  $Trans$  be  $[p|P]$ , where  $p$  is the first element and  $P$  is the remaining list. Call `Insert_tree([p|P], T)`, which is performed as follows. If  $T$  has a child  $N$  such that  $N.item-name = p.item-name$ , then increment  $N$ ’s count by 1; else create a new node  $N$ , and let its count be 1, its parent link be linked to  $T$ , and its node-link to the nodes with the same *item-name* via the node-link structure. If  $P$  is nonempty, call `Insert_tree(P, N)` recursively.

# FP-Growth Algorithm Cont...

2. The FP-tree is mined by calling `FP_growth(FP_tree, null)`, which is implemented as follows.

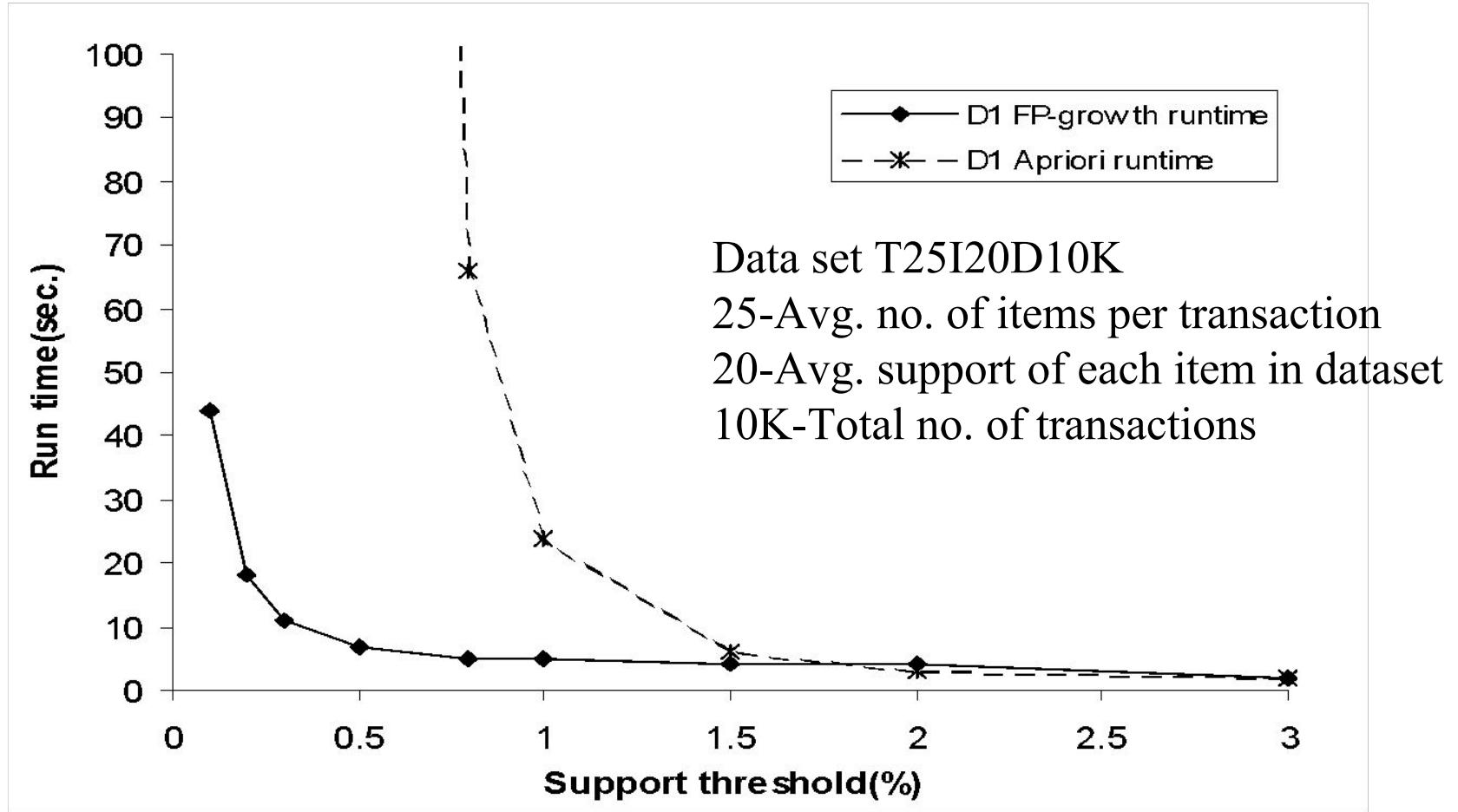
**procedure** `FP_growth(Tree, α)`

- (1) if *Tree* contains a single path *P* then
- (2)     for each combination (denoted as  $\beta$ ) of the nodes in the path *P*
- (3)         generate pattern  $\beta \cup \alpha$  with *support\_count* = *minimum support count of nodes in β*;
- (4)     else for each  $a_i$  in the header of *Tree* {
- (5)         generate pattern  $\beta = a_i \cup \alpha$  with *support\_count* =  $a_i.support\_count$ ;
- (6)         construct  $\beta$ 's conditional pattern base and then  $\beta$ 's conditional FP-tree  $Tree_{\beta}$ ;
- (7)         if  $Tree_{\beta} \neq \emptyset$  then
- (8)             call `FP_growth(Tree $_{\beta}$ , β); }`

# Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database

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



# Pros and Cons of FP-Growth

- Divide-and-conquer
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Performance is Faster than Apriori
  - Use compact data structure
  - No candidate generation, no candidate test
  - Eliminate repeated database scans
  - Basic operation is counting and FP-Tree building
- Problem:
  - When the database is large, sometimes unrealistic to construct a main memory based FP-Tree

# Data Format

- Apriori and FP-Growth
  - { TID: itemset }
    - TID: Transaction ID
    - Itemset: set of items bought in transaction TID
  - Horizontal Data Format
- Alternative way
  - { Item: TID\_set }
    - Item: item name
    - TID\_set: set of transaction identifiers containing the item
  - Vertical Data Format



- Before that ....



# Transection representation : Binary

**Table 6.1.** An example of market basket transactions.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

**Table 6.2.** A binary 0/1 representation of market basket data.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

# Data Layout for Transaction Matrix

- Horizontal item-list (HIL): The database is represented as a set of transactions, storing each transaction as a list of item identifiers (item-list).
- Horizontal item-vector (HIV): The database is represented as a set of transactions, but each transaction is stored as a bit-vector (item-vector) of 1's and 0's to express the presence or absence of the items in the transaction.
- Vertical tid-list (VTL): The database is organized as a set of columns with each column storing an ordered list (tid-list) of only the transaction identifiers (TID) of the transactions in which the item exists.
- Vertical tid-vector (VTV): This is similar to VTL, except that each column is stored as a bit-vector (tid-vector) of 1's and 0's to express the presence or absence of the items in the transactions

# Data Layout for Transaction Matrix

TID	Item-lists
1	1 2 3 5
2	2 3 4 5
3	3 4 5
4	1 2 3 4 5

HIL

Item-vectors
1 1 1 0 1
0 1 1 1 1
0 0 1 1 1
1 1 1 1 1

HIV

Tid-lists				
1	2	3	4	5
1	1	1	2	1
4	2	2	3	2
4	3	4	4	3
	4		4	4

VTL

Tid-vectors				
1	2	3	4	5
1	1	1	0	1
0	1	1	1	1
0	0	1	1	1
1	1	1	1	1

VTV

# Example: Data Layout

Horizontal  
Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

TID-list

# Mining by Exploring Vertical Data Format

- ECLAT (Equivalence CLASS Transformation)
- Developed by Zaki

# ECLAT: Introduction

- efficient and scalable version of the Apriori algorithm.
- While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph.
- This vertical approach of the ECLAT algorithm makes it a faster algorithm than the Apriori algorithm.

# ECLAT Algorithm

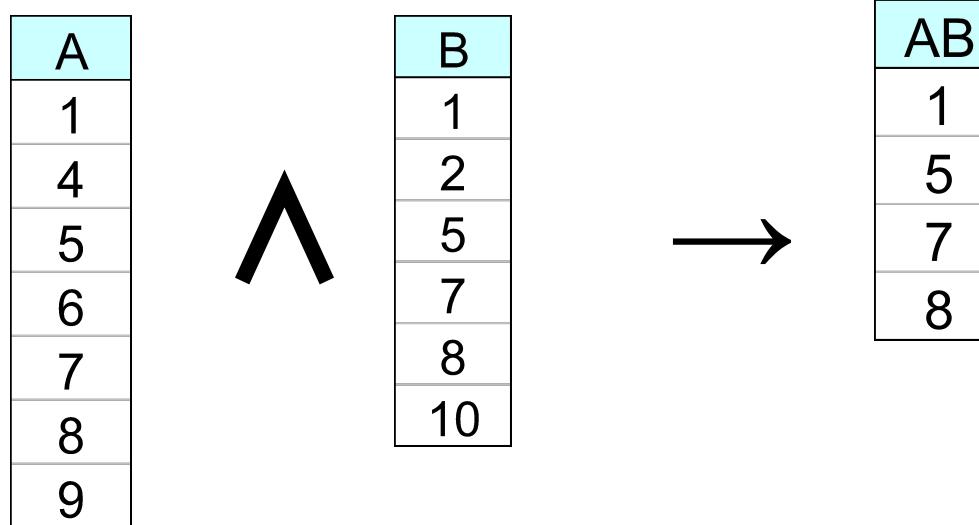
- Deriving frequent patterns based on vertical intersections
  - $t(X) = t(Y)$ : X and Y always happen together
  - $t(X) \subset t(Y)$ : transaction having X always has Y
- To count itemset AB
  - Intersect TID-list of itemA with TID-list of itemB

# ECLAT Algorithm

- Transform the horizontally formatted data to the vertical format by scanning the data set once
- Support count of an itemset
  - The length of the TID\_set of the itemset

# ECLAT Algorithm

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.



- 3 traversal approaches:
  - top-down, bottom-up and hybrid

# ECLAT Algorithm

- Starting with  $k=1$ , the Frequent  $k$ -itemsets can be used to construct the candidate  $(k+1)$  itemsets based on the Apriori property
  - Done by intersection of the TID\_sets of the frequent  $k$ -itemsets to compute the TID\_sets of the corresponding  $(k+1)$  itemsets
- This process repeats, with  $k$  incremented by 1 each time, until no frequent itemsets or no candidate itemsets can be found

# Eclat

- Transactions, originally stored in horizontal format,
- are read from disk and converted to vertical format.

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



Bread	Milk	Diaper
1	1	2
2	3	3
4	4	4
5	5	5

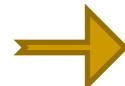
  

Beer	Coke	Eggs
2	3	2
3	5	
4		

# Eclat

The frequency of each item is counted and the infrequent items and their corresponding vertical lists are deleted from the vertical list.

Bread	Milk	Diaper
1	1	2
2	3	3
4	4	4
5	5	5



Bread	Milk	Diaper
1	1	2
2	3	3
4	4	4
5	5	5

Beer	Coke	Eggs
2	3	2
3	5	
4		

Beer
2
3
4

- Minimum support = 3

# Eclat

---

- The Eclat algorithm is defined recursively.
  - The initial call uses all the single items with their tidsets.
  - In each recursive call, the function verifies each itemset-tidset pair with all the others pairs to generate new candidates.
  - If the new candidate is frequent, it is added to the set. Then, recursively, it finds all the frequent itemsets in the branch.
-

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	
5	5	5	4



{Bread, Milk}	{Bread, Diaper}	{Bread, Beer}
1	2	2
4	4	4
5	5	

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	
5	5	5	4



{Bread, Milk}	{Bread, Diaper}	{Bread, Beer}
1	2	2
4	4	4
5	5	

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Bread, Beer}
2
4

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Bread, Beer}
2
4



{Bread, Milk, Diaper}
4
5

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Bread, Beer}
2
4



{Bread, Milk, Diaper}
4
5

# Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Milk, Diaper}	{Milk, Beer}
3	3
4	4
5	

# Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Milk, Diaper}	{Milk, Beer}
3	3
4	4
5	

# Eclat

		Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

		Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

		Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	



{Diaper, Beer}
2
3
4

# Eclat

1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

# Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}	{Bread, Diaper}
1	2
4	4
5	5

{Milk, Diaper}
3
4
5

{Diaper, Beer}
2
3
4

# Eclat

---

- Uses vertical database – tidset(bitset) intersections.
  - Scans the database only once.
  - Depth-first search algorithm.
-

# ECLAT Algorithm Summary

- Intersection is more efficient
- Pipelined counting for frequent itemsets
- Advantage
  - Less number of database scan
  - Very fast support counting
  - No need to scan the database to find the support of  $(k+1)$  itemsets ( for  $k \geq 1$  )
    - Because the TID\_set of each k-itemset carries the complete information required for counting each support

# ECLAT Algorithm

- Disadvantage
  - Intermediate tid-lists may become too large for memory
  - Long computation time for intersecting the long set
- Performance improvement Idea
  - Using **diffset** to accelerate mining [CHARM Algorithm]
    - Only keep track of differences of tids
    - $t(X) = \{T_1, T_2, T_3\}$ ,  $t(XY) = \{T_1, T_3\}$
    - Diffset  $(XY, X) = \{T_2\}$

# Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g.,  $\{a_1, \dots, a_{100}\}$  contains

$$\binom{100}{1} = 100 \text{ frequent 1-itemsets}$$

$$\binom{100}{2} \text{ 2-frequent item set}$$

$$\binom{100}{100} \text{ 100-frequent itemset}$$

$$\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} =$$

$$2^{100} - 1 = 1.27 \cdot 10^{30} \text{ sub-patterns!}$$

- Solution: Mine *closed patterns* and *max-patterns instead*

# Max-Patterns

- An itemset  $X$  is a **max-pattern (maximal)** if  $X$  is frequent and there exists no frequent super-pattern  $Y \supset X$

# Maximal Frequent Itemset

## ***Definition***

- It is a frequent itemset for which none of its immediate supersets are frequent.

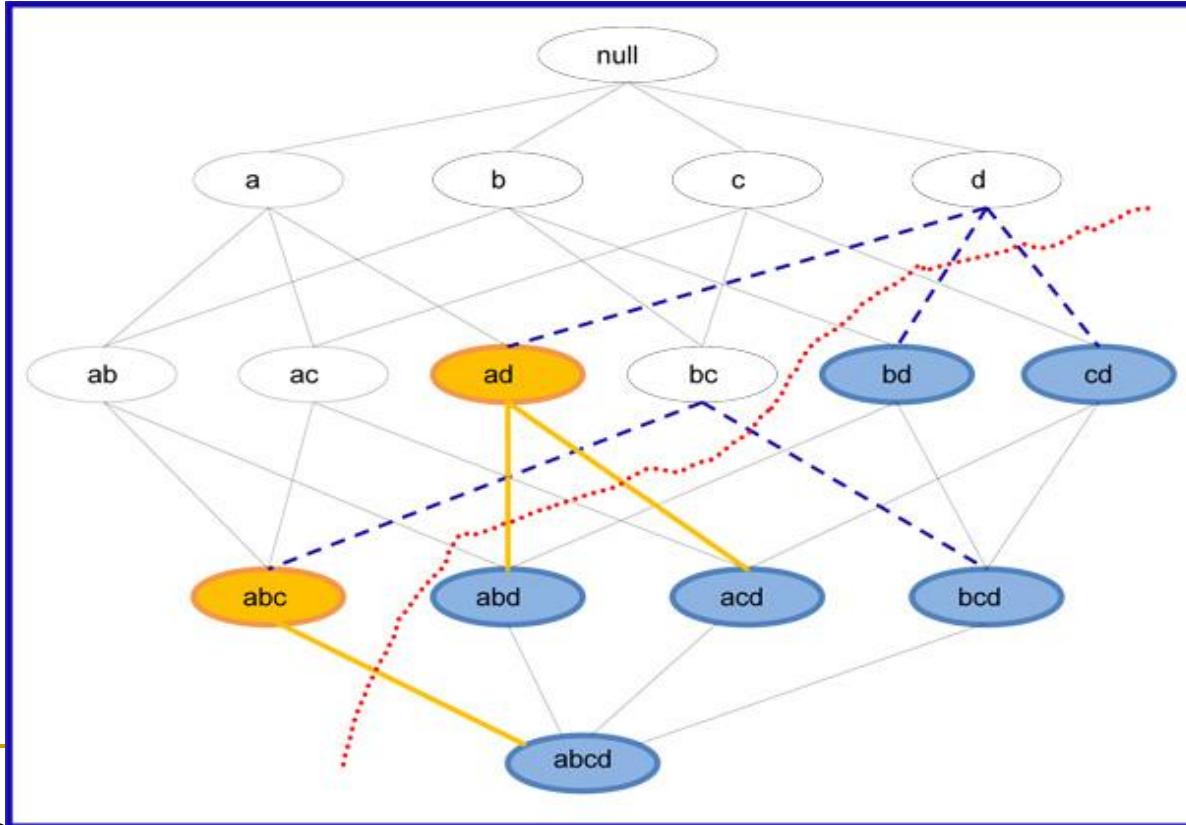
## ***Identification***

- Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets.
- Identify all of its immediate supersets.
- If none of the immediate supersets are frequent, the itemset is maximal frequent.

# Example

- first identify the frequent itemsets at the border

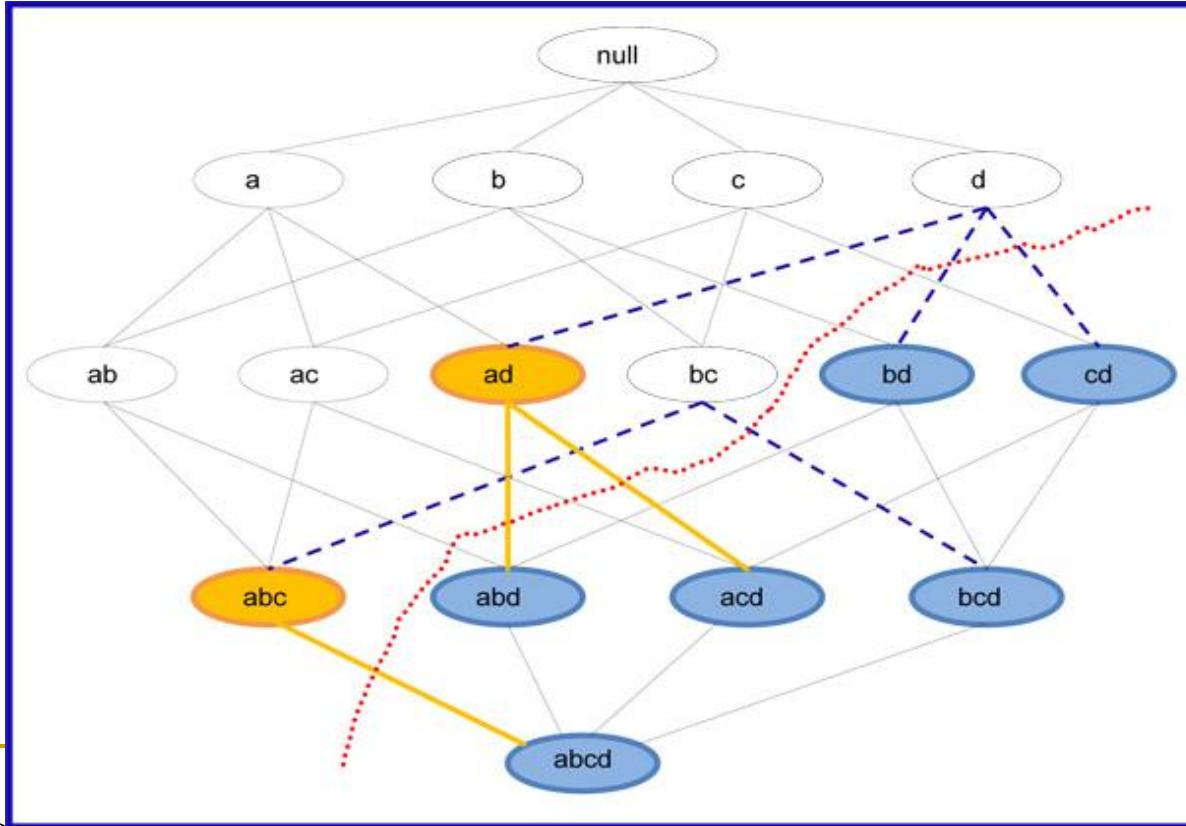
- d, bc, ad and abc.***



# Example

■ lattice is divided into two groups

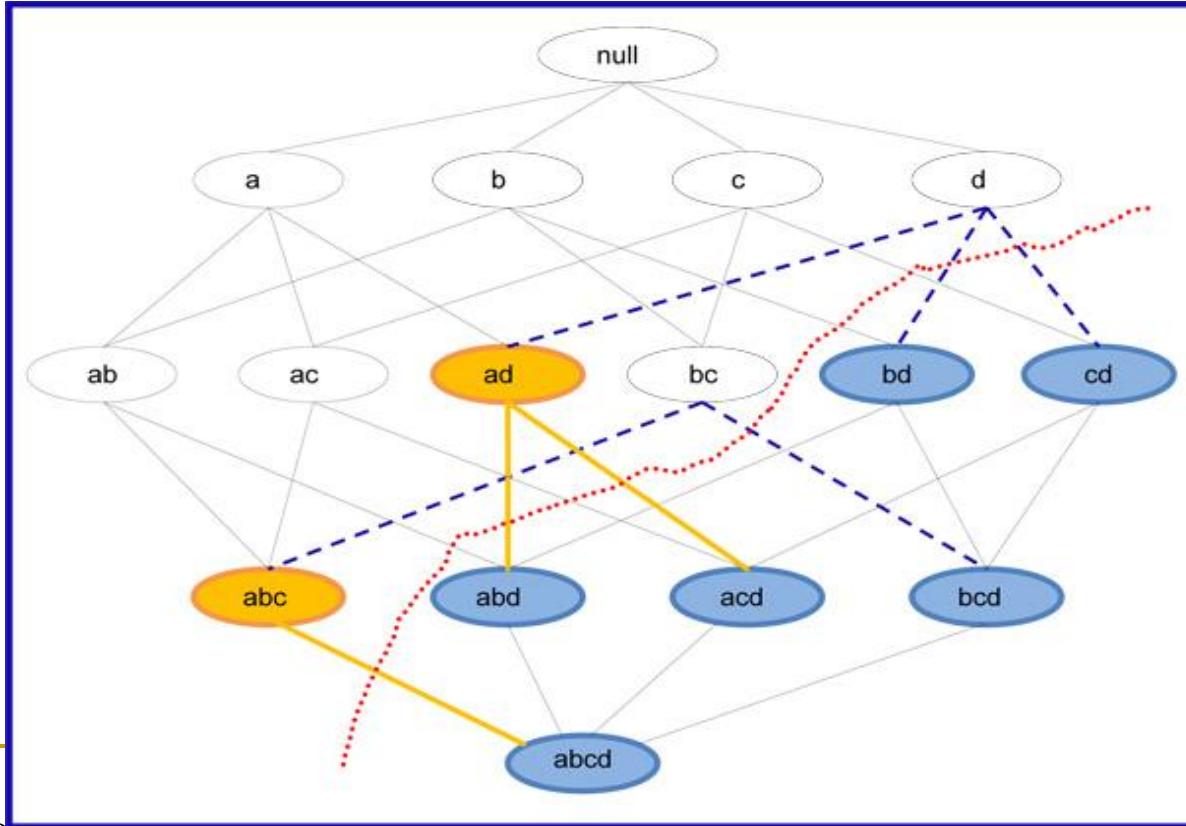
- items above the red line (Demarcation) that are blank are frequent itemsets
- and the blue ones below the red dashed line are infrequent.



# Example

■ lattice is divided into two groups

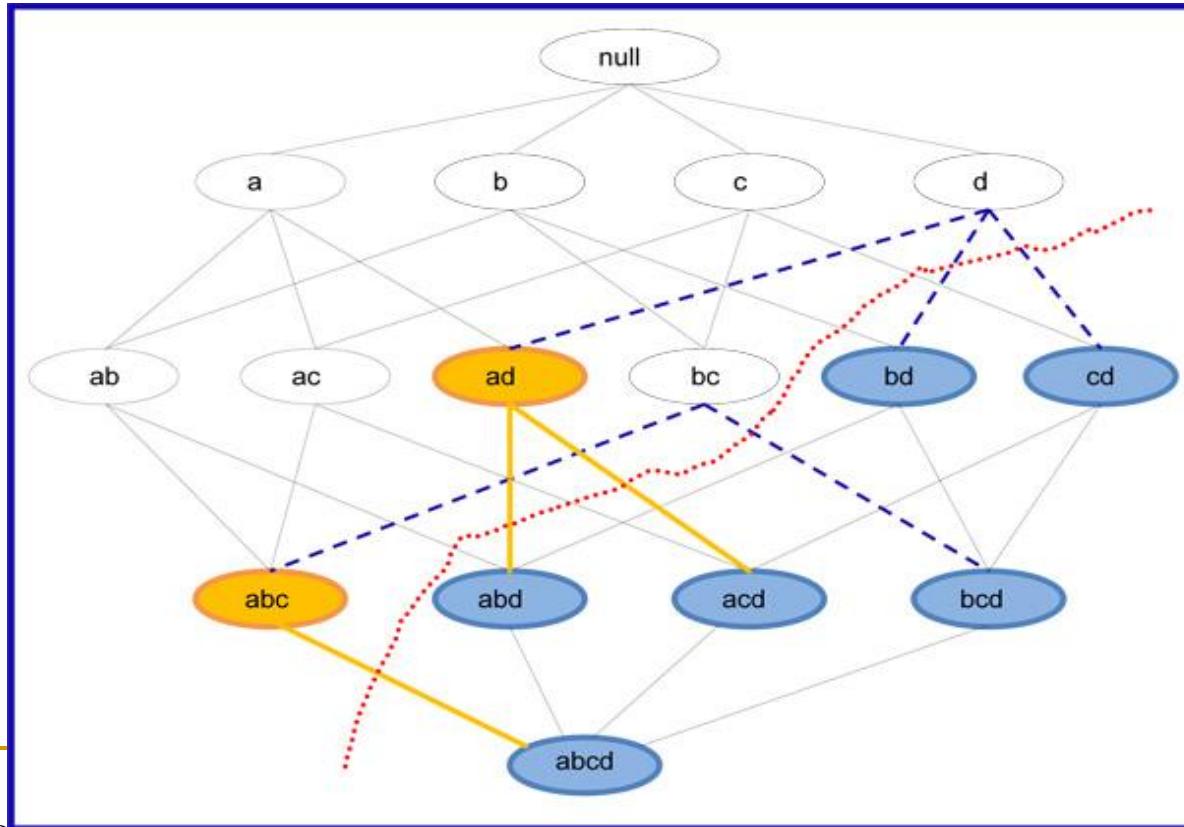
- items above the red line (Demarcation) that are blank are frequent itemsets
- and the blue ones below the red dashed line are infrequent.



# Example

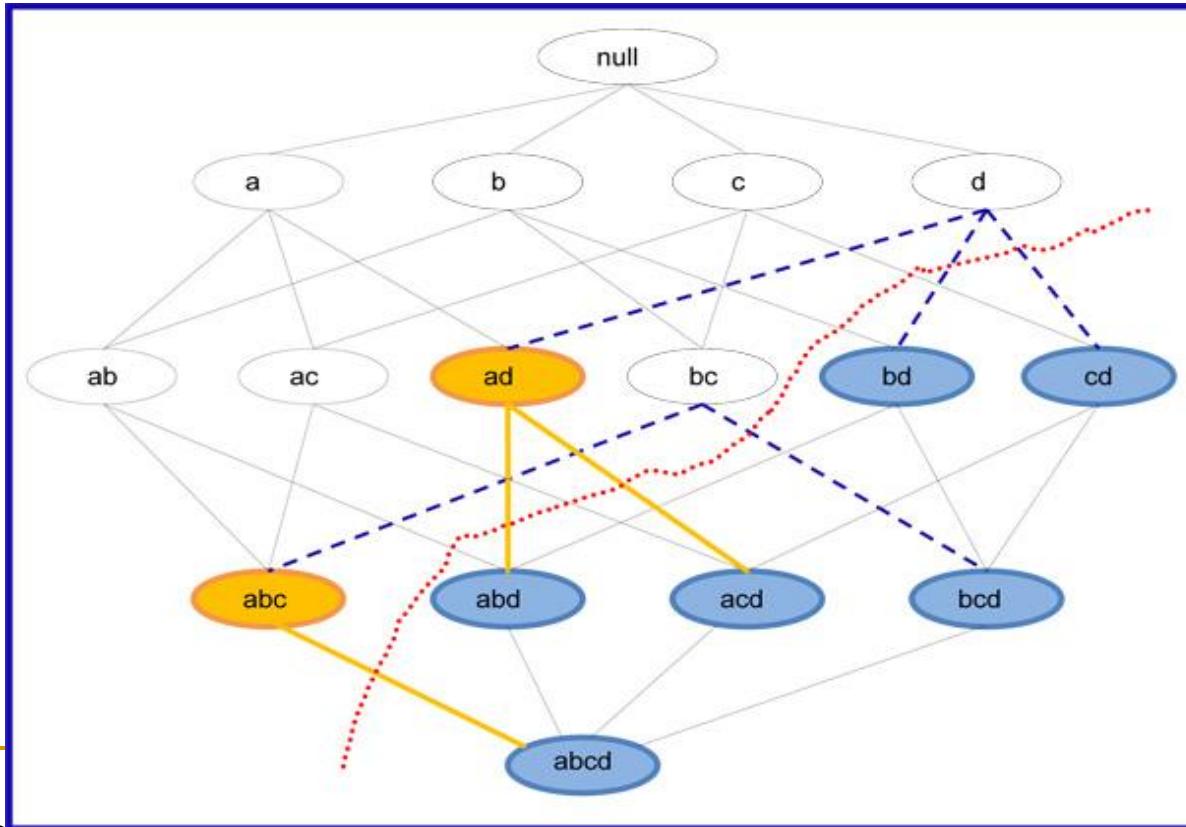
■ lattice is divided into two groups

- items above the red line (Demarcation) that are blank are frequent itemsets
- and the blue ones below the red dashed line are infrequent.



# Example

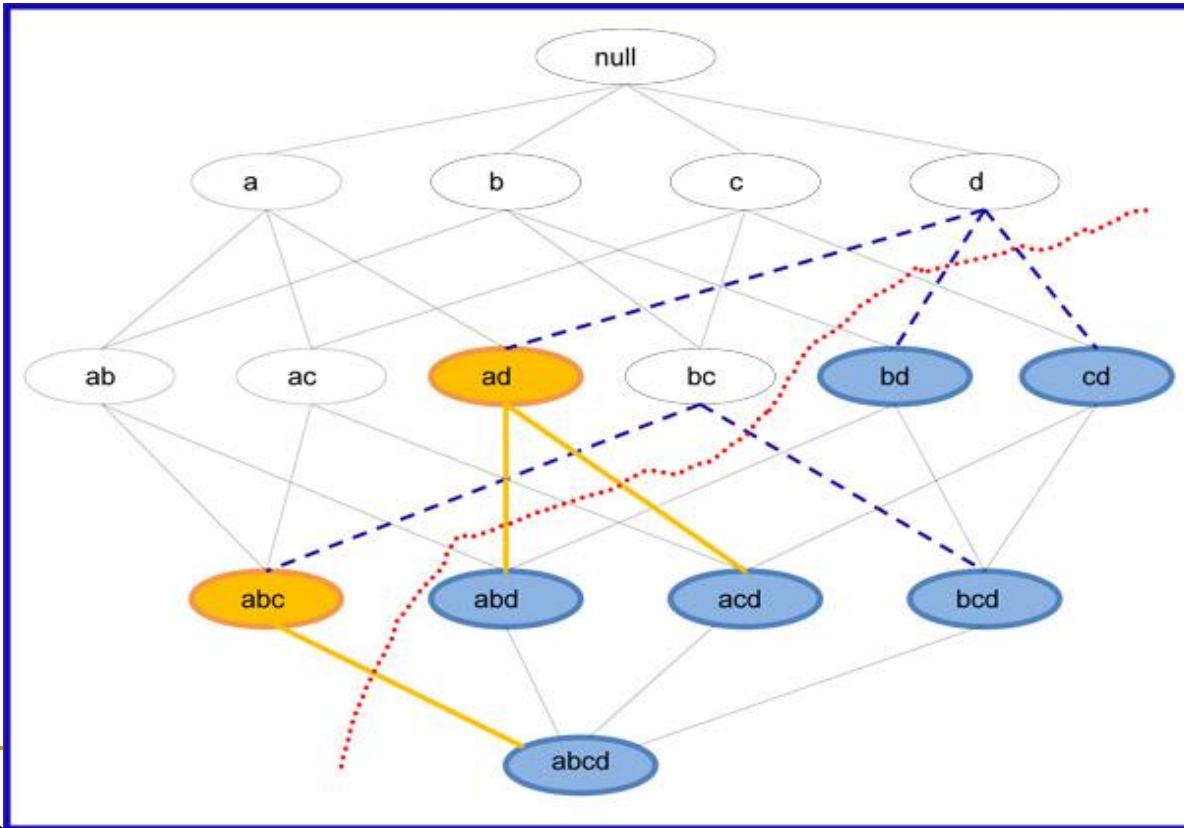
- Identify their immediate supersets,
  - blue dashed line



# Example

■ for d :

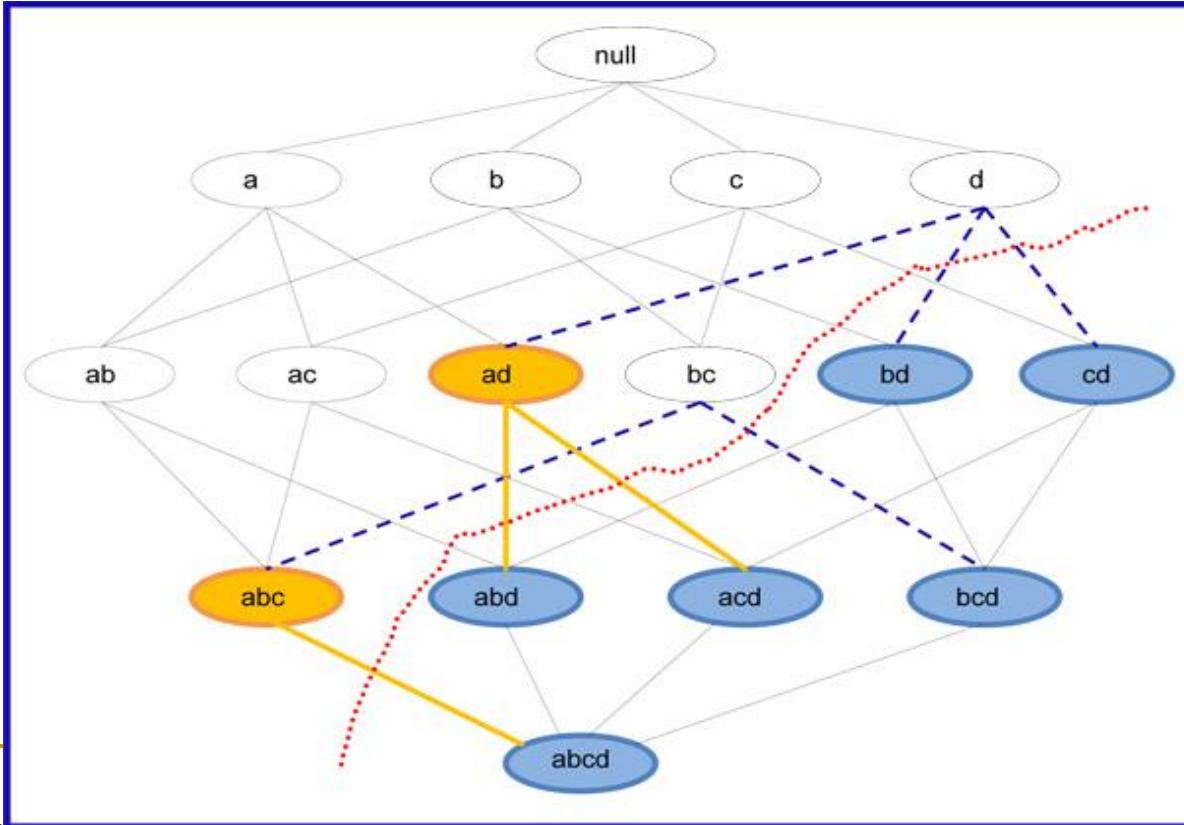
- bd and cd are nonfrequent
- ad is frequent
- d is not maximal frequent,



# Example

■ for  $bc$  :

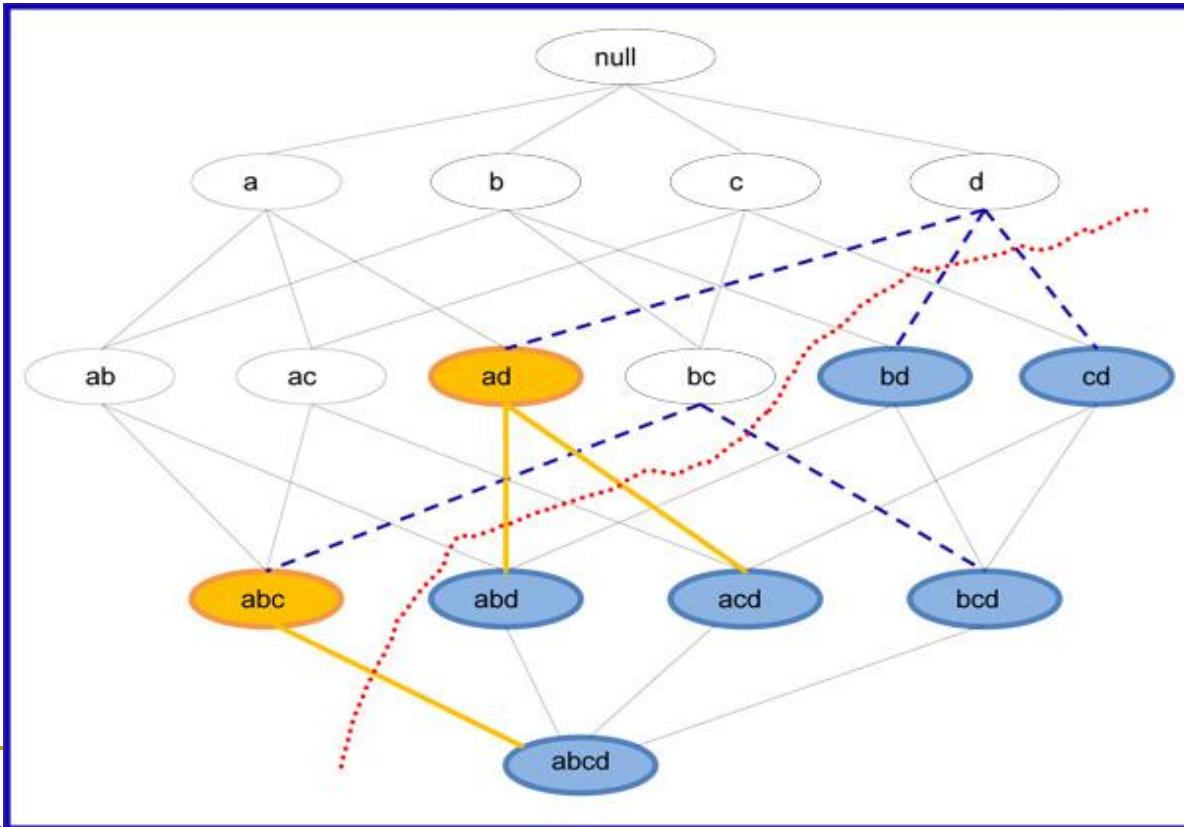
- $abc$  is frequent
- $bcd$  is non frequent
- $bc$  is NOT maximal frequent.



# Example

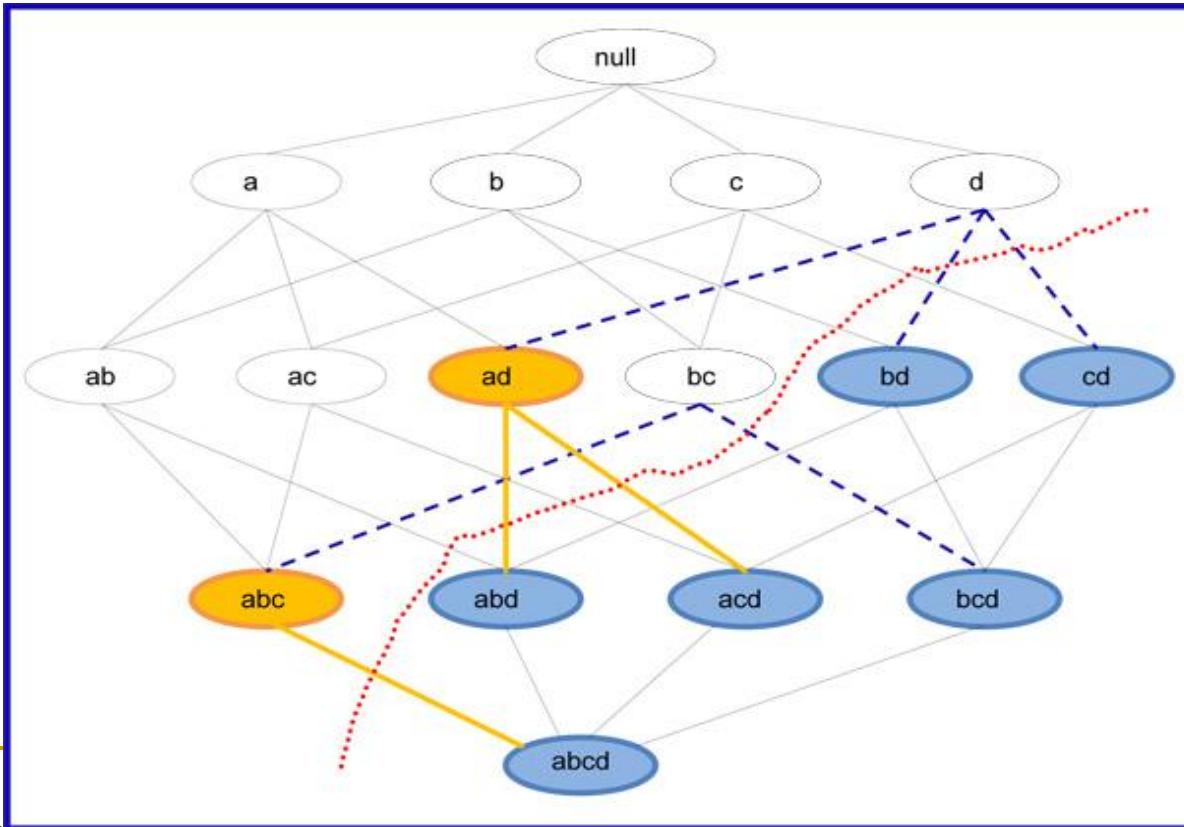
■ for ad

- abd and acd are infrequent
- ad maximal



# Example

- for abc
  - $abcd$  is infrequent
  - $ABC$  maximal



# Closed Patterns

- An itemset  $X$  is **closed** if  $X$  is *frequent* and there exists *no super-pattern*  $Y \supset X$ , *with the same support* as  $X$
- It is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

# Closed Frequent Itemset

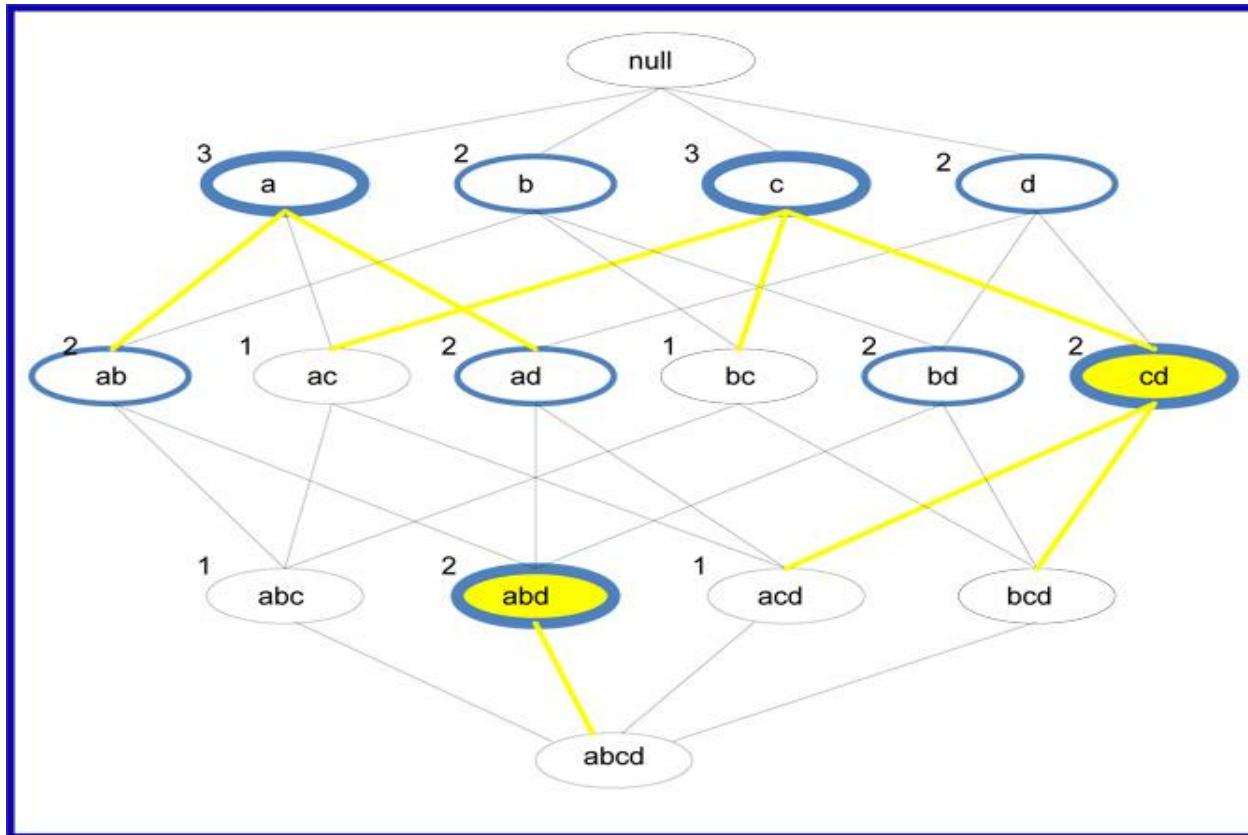
## ■ ***Defination***

- An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset. (all superset must be with less than the item support)
- It is a frequent itemset that is both closed and its support is greater than or equal to minsup.

## ■ ***Identification***

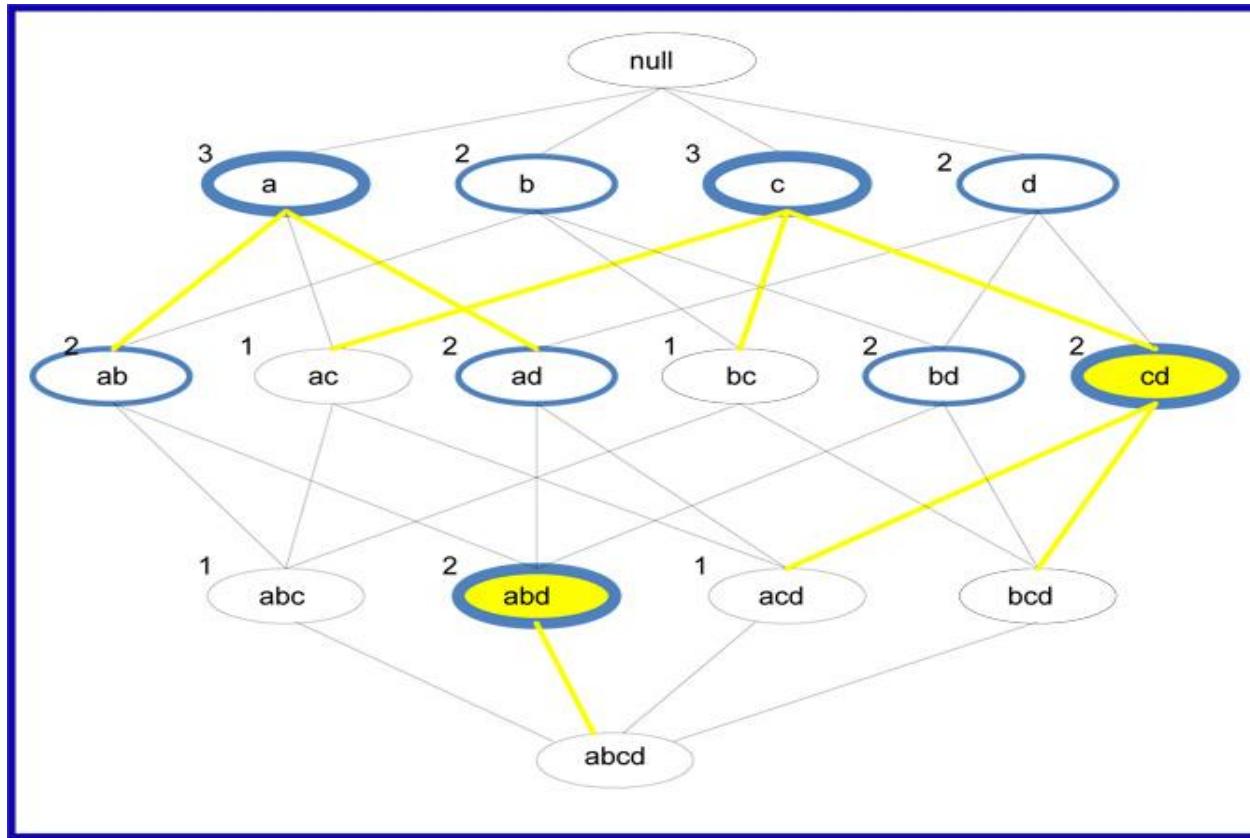
- Method1: Identify all frequent itemsets then check for superset with the same support. If found then disqualify otherwise item is closed
- Method2: first identify the closed itemsets and then use the minsup to determine which ones are frequent.

- The itemsets that are circled with blue are the frequent itemsets.
- The itemsets that are circled with the thick blue are the closed frequent itemsets.
- The itemsets that are circled with the thick blue and have the yellow fill are the maximal frequent itemsets.



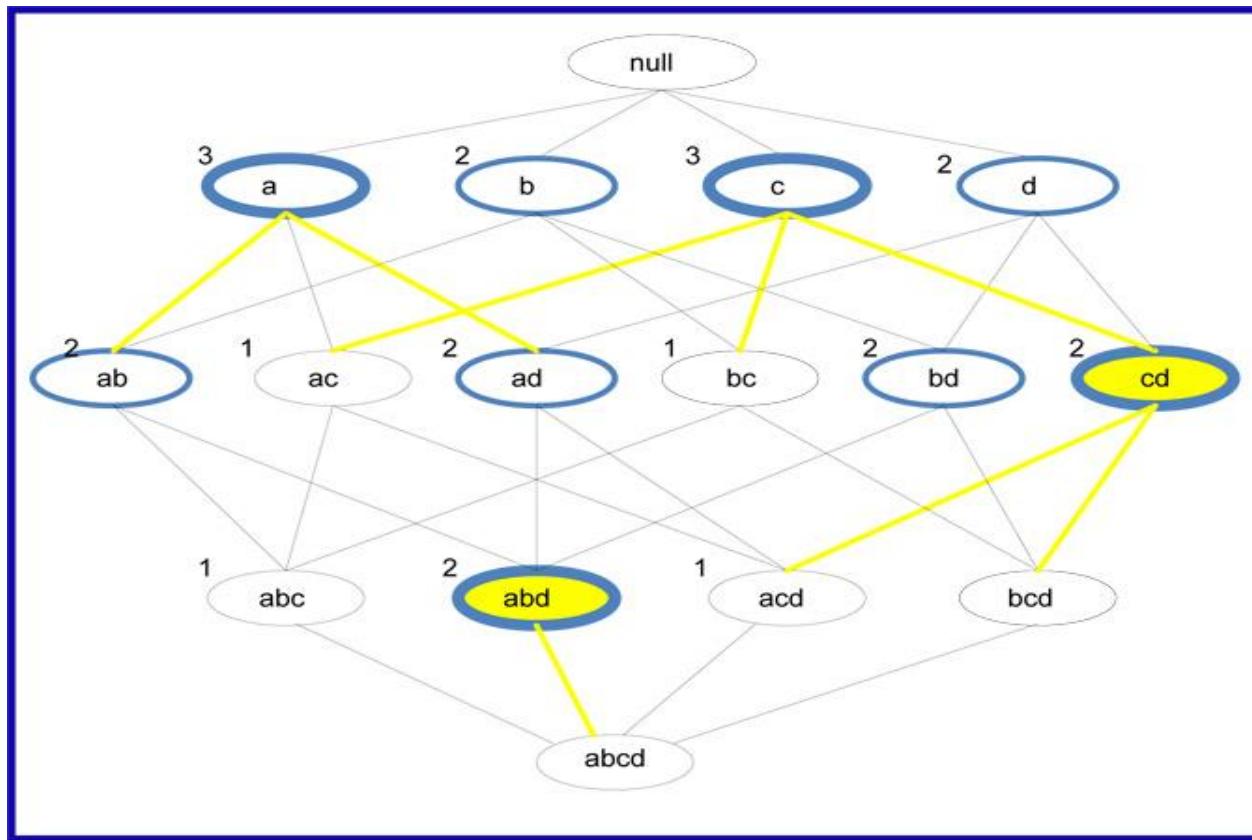
- For ad

- frequent itemset
- support of ad = support of **abd**
- so it is NOT a closed frequent itemset

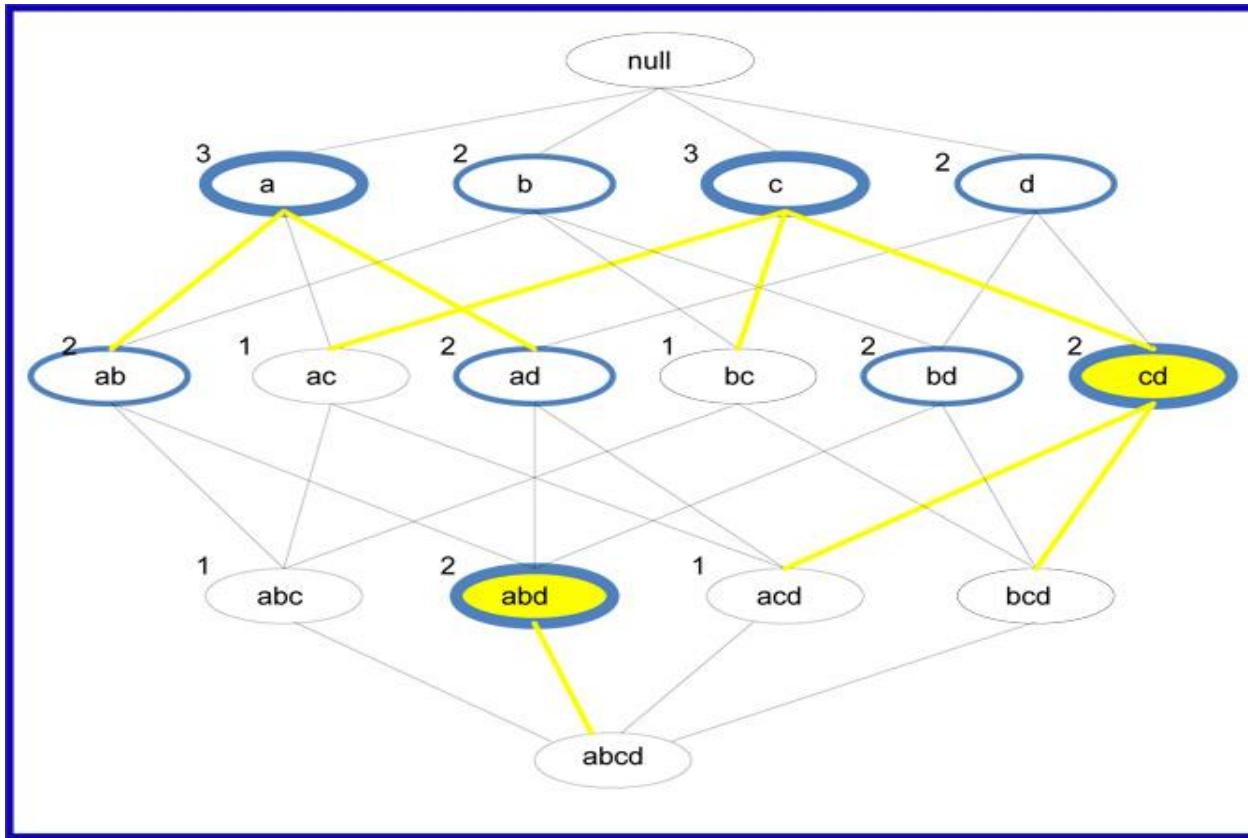


- For ad

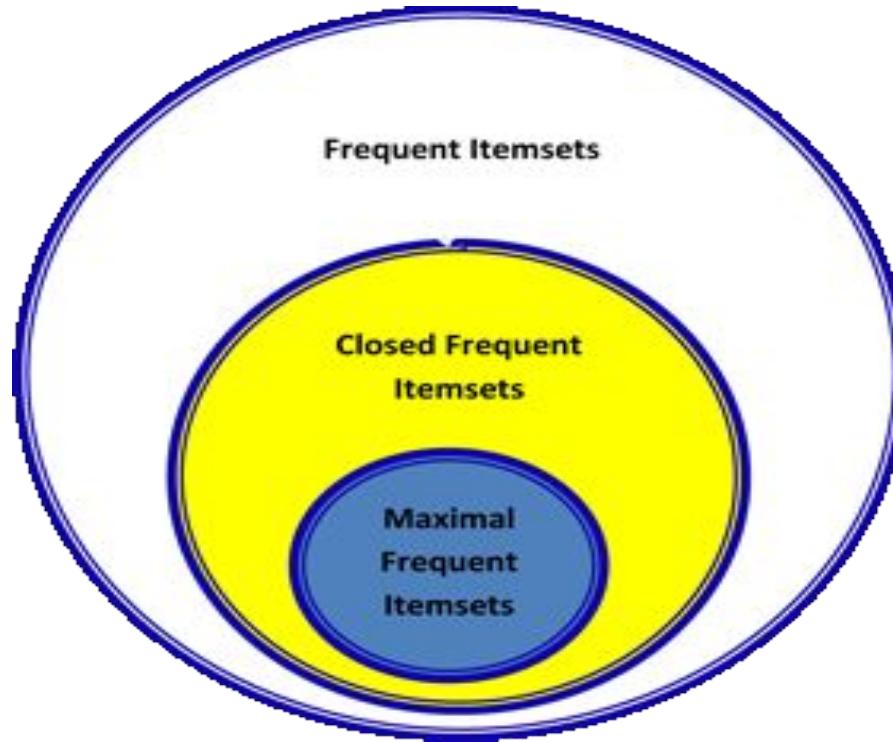
- c is frequent itemset
- support of C >
  - Support of ac, bc and cd



- 9 frequent itemsets,
- 4 of them are closed frequent itemsets
- out of these 4, 2 of them are maximal frequent itemsets.



# Relationship between three measures



- Closed frequent itemsets are more widely used than maximal frequent itemset
  - Find closed
  - Then find closed frequent

# Max-Patterns - Example

Transaction Database

1:  $\{a, d, e\}$   
2:  $\{b, c, d\}$   
3:  $\{a, c, e\}$   
4:  $\{a, c, d, e\}$   
5:  $\{a, e\}$   
6:  $\{a, c, d\}$   
7:  $\{b, c\}$   
8:  $\{a, c, d, e\}$   
9:  $\{b, c, e\}$   
10:  $\{a, d, e\}$

Frequent Item Set

1 item	2 items	3 items
$\{a\}$ : 7	$\{a, c\}$ : 4	$\{a, c, d\}$ : 3
$\{b\}$ : 3	$\{a, d\}$ : 5	$\{a, c, e\}$ : 3
$\{c\}$ : 7	$\{a, e\}$ : 6	$\{a, d, e\}$ : 4
$\{d\}$ : 6	$\{b, c\}$ : 3	
$\{e\}$ : 7	$\{c, d\}$ : 4	
	$\{c, e\}$ : 4	
	$\{d, e\}$ : 4	

MinSup = 3

Find Itemset with none of superset are nonfrequent ( $\text{sup} > 3$ )

The maximal item sets are  $\{b, c\}$   $\{a, c, d\}$   $\{a, c, e\}$   $\{a, d, e\}$

- Every frequent itemset is a subset of at least one of these sets

# Closed Patterns - Example

Transaction Database

1:  $\{a, d, e\}$   
2:  $\{b, c, d\}$   
3:  $\{a, c, e\}$   
4:  $\{a, c, d, e\}$   
5:  $\{a, e\}$   
6:  $\{a, c, d\}$   
7:  $\{b, c\}$   
8:  $\{a, c, d, e\}$   
9:  $\{b, c, e\}$   
10:  $\{a, d, e\}$

Frequent Item Set

1 item	2 items	3 items
$\{a\}$ : 7	$\{a, c\}$ : 4	$\{a, c, d\}$ : 3
$\{b\}$ : 3	$\{a, d\}$ : 5	$\{a, c, e\}$ : 3
$\{c\}$ : 7	$\{a, e\}$ : 6	$\{a, d, e\}$ : 4
$\{d\}$ : 6	$\{b, c\}$ : 3	
$\{e\}$ : 7	$\{c, d\}$ : 4	
	$\{c, e\}$ : 4	
	$\{d, e\}$ : 4	

- $\{b\}$  is a subset of  $\{b, c\}$  both have a support of 3
- $\{c\}$  is subset of  $\{b, c\}$  but support ( $c$ ) > support( $\{b, c\}$ )
- $\{d, e\}$  is a subset of  $\{a, d, e\}$  both have a support of 4

All frequent item sets are **Closed except {b} and {d, e}**

# Closed Patterns and Max-Patterns

- $DB = \{<a_1, \dots, a_{100}>, <a_1, \dots, a_{50}>\}$ 
  - $\text{Min\_sup} = 1.$

Tid	Items
T1	A1,A2...A100
T2	A1,a2,...A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?

- all  $\text{Supp}(\text{superset}) < \text{sup}(item)$

# Closed Patterns and Max-Patterns

- $DB = \{\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle\}$ 
  - $\text{Min\_sup} = 1.$

Tid	Items
T1	A1,A2...A100
T2	A1,a2,...A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all  $\text{Supp}(\text{superset}) < \text{sup}(item)$
  - $\langle a_1, \dots, a_{100} \rangle: 1$
  - $\langle a_1, \dots, a_{50} \rangle: 2$

# Closed Patterns and Max-Patterns

- $DB = \{\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle\}$ 
  - $\text{Min\_sup} = 1.$

Tid	Items
T1	A1,A2...A100
T2	A1,a2,...A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all  $\text{Supp}(\text{superset}) < \text{sup}(item)$
  - $\langle a_1, \dots, a_{100} \rangle$ : 1
  - $\langle a_1, \dots, a_{50} \rangle$ : 2
- What is the set of max-pattern?

# Closed Patterns and Max-Patterns

- $DB = \{\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle\}$ 
  - $\text{Min\_sup} = 1.$

Tid	Items
T1	A1,A2...A100
T2	A1,a2,...A50

Item	Support
A1..A50	2
A51..A100	1

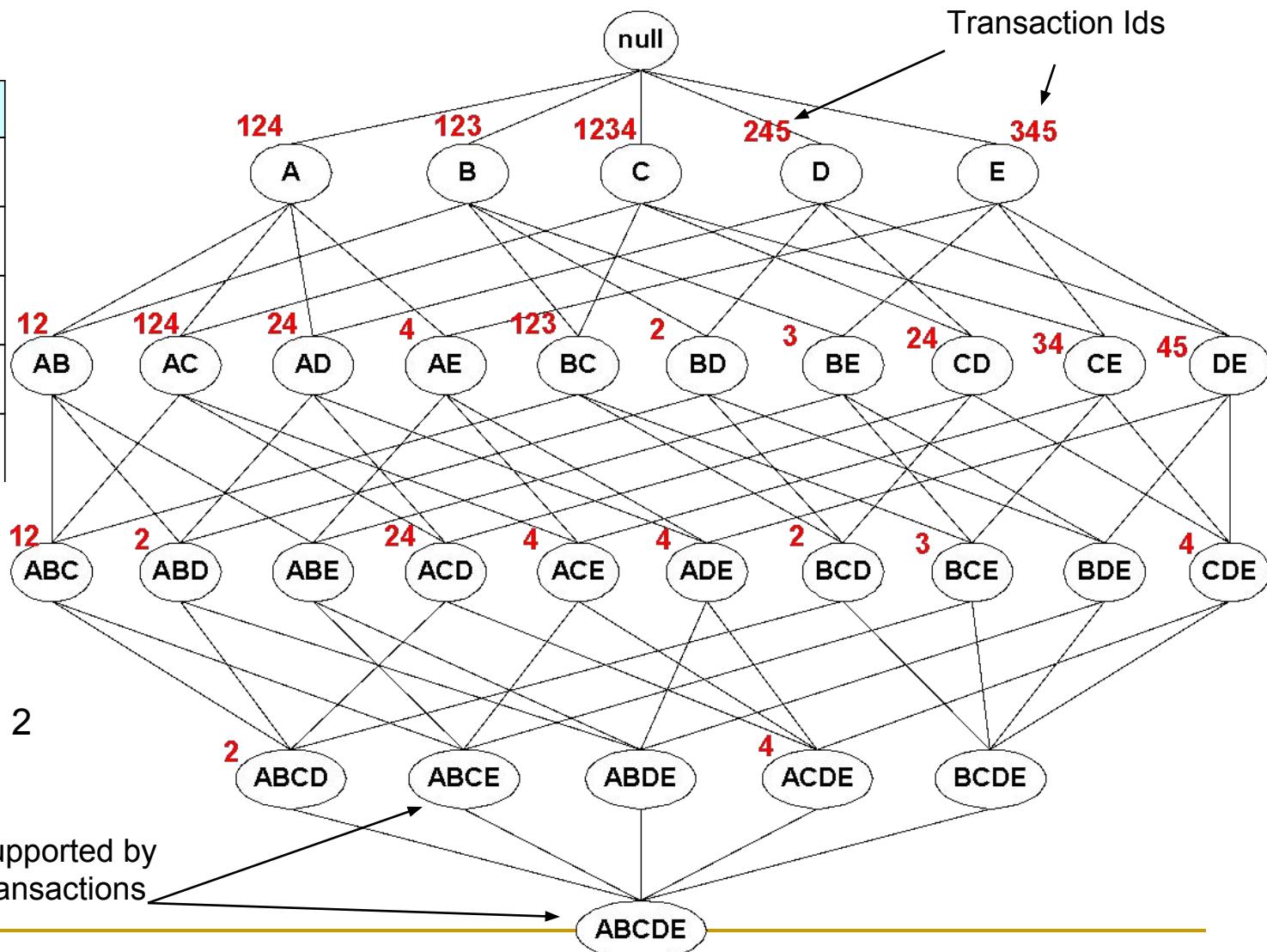
- What is the set of closed itemset?
  - all  $\text{Supp}(\text{superset}) < \text{sup}(item)$
  - $\langle a_1, \dots, a_{100} \rangle: 1$
  - $\langle a_1, \dots, a_{50} \rangle: 2$
- What is the set of max-pattern?
  - $\langle a_1, \dots, a_{100} \rangle: 1$

# Mine the Closed and Max-Patterns

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

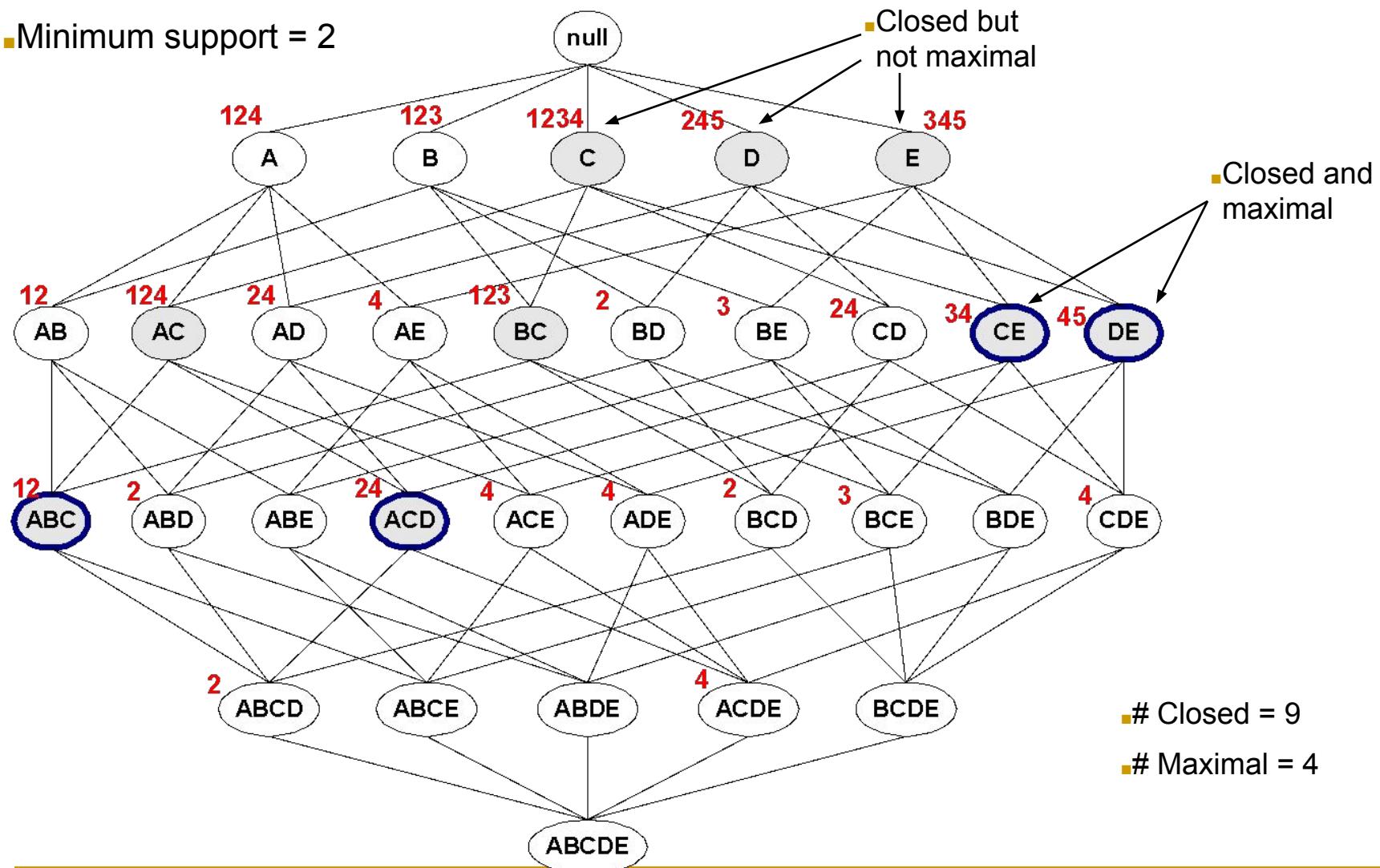
Minimum support = 2

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



# Maximal vs Closed Frequent Itemsets

■ Minimum support = 2



# Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g.,  $\{a_1, \dots, a_{100}\}$  contains

$$\binom{100}{1} = 100 \text{ frequent 1-itemsets}$$

$$\binom{100}{2} \text{ 2-frequent item set}$$

$$\binom{100}{100} \text{ 100-frequent itemset}$$

$$\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} =$$

$$2^{100} - 1 = 1.27 \cdot 10^{30} \text{ sub-patterns!}$$

- Solution: Mine *closed patterns* and *max-patterns instead*

# Closed Patterns and Max-Patterns

- $DB = \{\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle\}$ 
  - $\text{Min\_sup} = 1.$

Tid	Items
T1	A1,A2...A100
T2	A1,a2,...A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all  $\text{Supp}(\text{superset}) < \text{sup}(item)$
  - $\langle a_1, \dots, a_{100} \rangle: 1$
  - $\langle a_1, \dots, a_{50} \rangle: 2$
- What is the set of max-pattern?

- $\langle a_1, \dots, a_{100} \rangle: 1$

# How to Mine Closed Frequent Itemsets

- From the set of closed frequent itemsets, we can easily derive the set of frequent itemsets and their support.
- In practice, it is more desirable to mine the set of closed frequent itemsets rather than the set of all frequent itemsets in most cases.

# How to Mine Closed Frequent Itemsets

- Example and Exercise - a naïve approachFirst mine the complete set of frequent itemsets and then remove every frequent itemset that is a proper subset of, and carries the same support as, an existing frequent itemset
- It is quite costly
- Search for closed frequent itemsets directly during the mining process

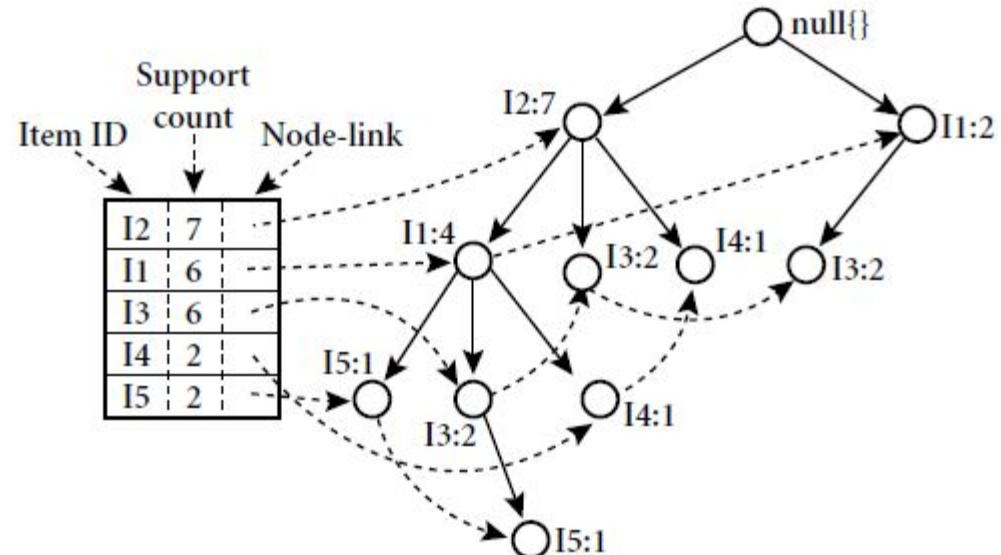
# How to Mine Closed Frequent Itemsets

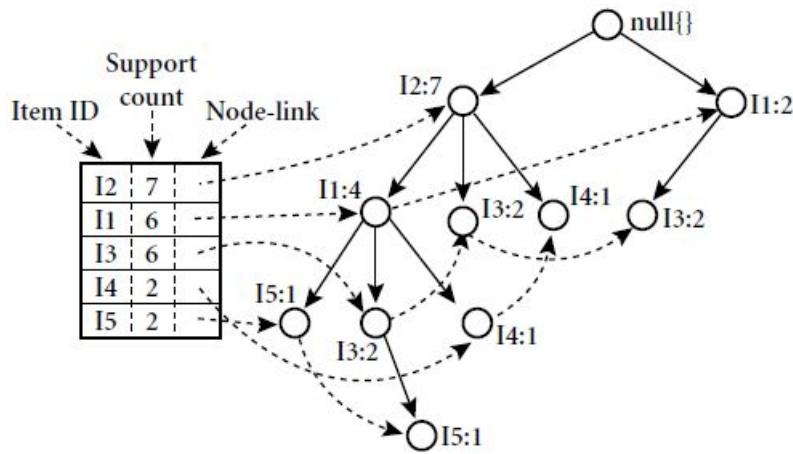
- Requires to prune the search space as soon as we can identify the case of closed itemsets during mining
  - Item merging: *If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y, then  $X \cup Y$  forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y.*
- Pruning
  - Item Merging
    - if Y appears in every occurrence of X, then Y is merged with X
  - Sub-itemset Pruning
  - Item Skipping

# Example:

Transactional data for an *AllElectronics* branch.

TID	List of item IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3





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

<i>Item</i>	<i>Conditional Pattern Base</i>	<i>Conditional FP-tree</i>	<i>Frequent Patterns Generated</i>
I5	$\{\{I2, I1: 1\}, \{I2, I1, I3: 1\}\}$	$\langle I2: 2, I1: 2 \rangle$	$\{I2, I5: 2\}, \{I1, I5: 2\}, \{I2, I1, I5: 2\}$
I4	$\{\{I2, I1: 1\}, \{I2: 1\}\}$	$\langle I2: 2 \rangle$	$\{I2, I4: 2\}$
I3	$\{\{I2, I1: 2\}, \{I2: 2\}, \{I1: 2\}\}$	$\langle I2: 4, I1: 2 \rangle, \langle I1: 2 \rangle$	$\{I2, I3: 4\}, \{I1, I3: 4\}, \{I2, I1, I3: 2\}$
I1	$\{\{I2: 4\}\}$	$\langle I2: 4 \rangle$	$\{I2, I1: 4\}$

# Example from FP-Growth

<u>TID</u>	<u>Items bought</u>	<u>(ordered) frequent items</u>	<u>min_support = 3</u>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}	
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}	
300	{b, f, h, j, o, w}	{f, b}	
400	{b, c, k, s, p}	{c, b, p}	
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}	

M's conditional database is

$\{\{fca:2\}, \{fcab:1\}\}$

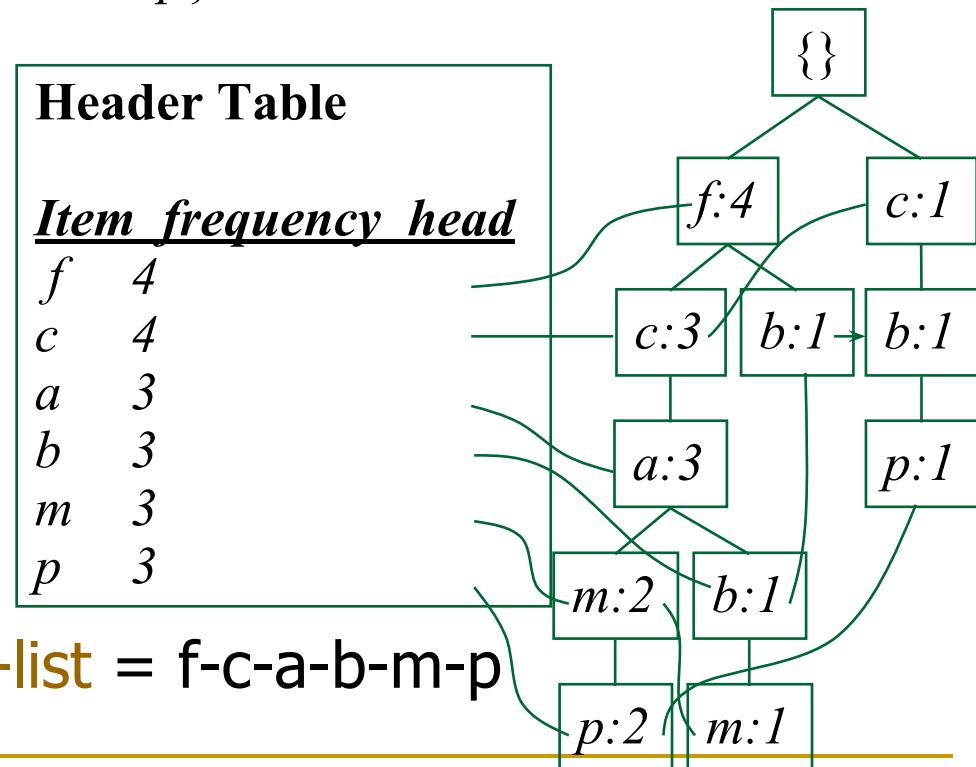
Meaning is each of transactions contains itemset {f,c,a}, this can be merged with {m} to form the closed itemset  $\{fcam:3\}$

**Header Table**

Item frequency head

f	4
c	4
a	3
b	3
m	3
p	3

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



# Mining Closed Itemsets

## ■ Sub-itemset pruning

- if  $Y \supset X$ , and  $\text{sup}(X) = \text{sup}(Y)$ ,  $X$  and all of  $X$ 's descendants in the set enumeration tree can be pruned
  - $DB = \{\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle\}$ , where  $\text{minSup}=2$
  - The projection of the first item,  $a_1$ , derives the frequent itemset,  $\{a_1, \dots, a_{50}\}$ : $2\}$ , based on the itemset merging optimization
  - Because  $\text{support}(\{a_2\}) = \text{support}(\{a_1, \dots, a_{50}\})$  and  $\{a_2\}$  is a proper subset of  $\{a_1, \dots, a_{50}\}$ 
    - No need to examine  $a_2$  and its projected database
  - Similar pruning can be done for  $a_3 \dots a_{50}$
  - Mining of closed itemsets in this data set terminates after mining  $a_1$ 's projected database

# Mining Closed Itemsets

## ■ Item skipping

- if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
  - DB = { $\langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle$ }, where minSup=2
  - $a_2$  is in  $a_1$ 's projected database and has the same support as  $a_2$  in global header table,  $a_2$  can be pruned from the global header table
  - Similar pruning can be done for  $a_3 \dots a_{50}$
  - No need to mine anything more after mining  $a_1$ 's projected database

# Mining Closed Itemsets

- Need of efficient closure checking
  - Check whether the newly found itemset is a subset of an already found closed itemset with the same support
  - Check whether the newly found itemset is a superset of an already found closed itemset with the same support
  - Uses Pattern-Tree structure
    - Similar to the FP-Tree except that all of the closed itemsets found are stored explicitly in the corresponding tree branches
- Algorithms: CLOSET, CLOSET+

# MaxMiner: Mining Max-Patterns

- Extend the closed item methods with maximal frequent itemset
- 1<sup>st</sup> scan: find frequent items
  - A, B, C, D, E
- 2<sup>nd</sup> scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, DE, CDE
- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

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

Potential  
max-patterns

# Problems with the Association Mining

- Single minsup
  - It assumes that all items in the data are of the **same nature** and/or have **similar frequencies**
- Not true
  - In many applications, some items appear very frequently in the data, while others rarely appear
    - e.g., in a supermarket, people buy *food processor* and *cooking pan* much less frequently than they buy *bread* and *milk*

# Rare Item Problem

- If the frequencies of items vary a great deal, there will be **two problems**
  - If **minsup** is set too high, those rules that involve rare items will not be found
  - To find rules that involve both frequent and rare items, **minsup has to be set very low**
    - This may cause **combinatorial explosion** because those frequent items will be associated with one another in all possible ways

# Mining Various Kinds of Rules or Regularities

- Multi-level
  - At different level of abstraction
- Multi-Dimensional
  - More than one dimension (e.g. items bought by Student)
- Quantitative association rules
  - Involve numeric attributes that have an implicit ordering among values (e.g. age)
- Constraint Based

# ML/MD Associations with Flexible Support Constraints

- Why flexible support constraints?
  - Real life occurrence frequencies vary greatly
    - Diamond, watch, pens in a shopping basket
  - Uniform support may not be an interesting model
- A flexible model
  - The lower-level, the more dimension combination, and the long pattern length, usually the smaller support
  - General rules should be easy to specify and understand
  - Special items and special group of items may be specified individually and have higher priority

# Multiple-level Association Rules

- Items often form hierarchy
- Uniform support
  - The same minimum support threshold is used when mining at each level of abstraction
  - Only one minimum support is required
  - Search procedure is simplified

uniform support

Level 1  
 $\text{min\_sup} = 5\%$

Milk  
[support = 10%]

Level 2  
 $\text{min\_sup} = 5\%$

2% Milk  
[support = 6%]

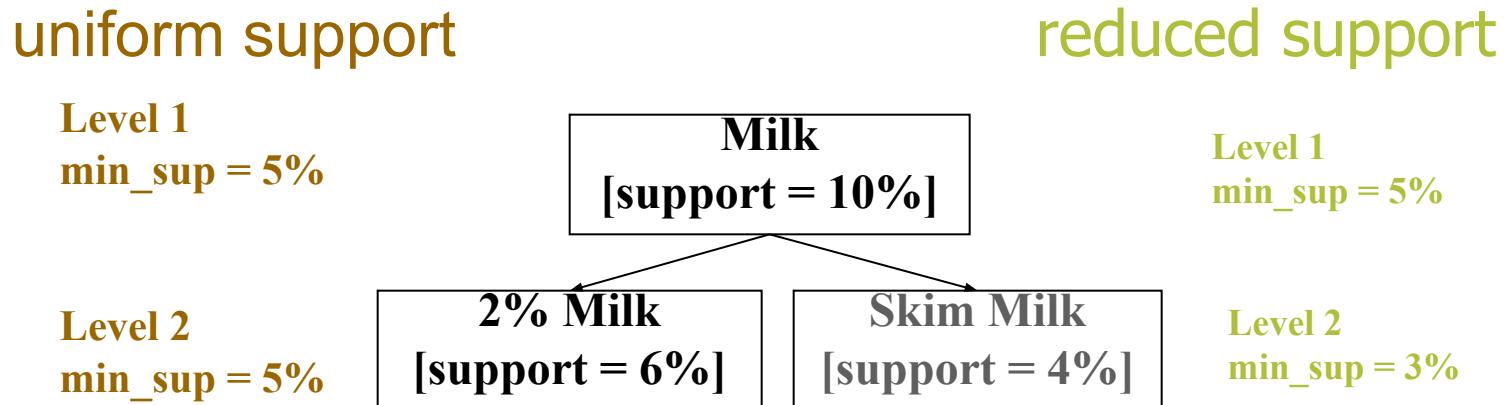
Skim Milk  
[support = 4%]

# Multiple-level Association Rules

- Uniform support
  - Apriori can be adopted with the knowledge that the ancestor is a superset of its descendants and searching is avoided for any item whose ancestors do not have minimum support
  - Difficulties
    - Items at lower levels of abstraction will not occur as frequently as those at higher levels of abstraction
    - If minsup is too high, may miss some meaningful associations at low abstraction levels
    - If minsup is too low, may generate many uninteresting associations occurring at high abstraction levels

# Multiple-level Association Rules

- Reduced Minimum support at lower levels
  - Each level of abstraction has its own minimum support threshold
  - The deeper the level of abstraction, the smaller the corresponding threshold is



# Multiple-level Association Rules

- Item or Group based Minimum Support
  - As the groups are more important, it is desirable to set up user-specific, item or group based minimal support thresholds when mining
  - A group/combination can be visualize and set the low support threshold for the group ( e.g. Laptop and Flash Drive) to checkout the association pattern containing items in this categories

# Multi-level Association: Redundancy Filtering

- Apriori cannot be apply directly for reduced support and group support
- Problem: Some rules may be redundant due to “ancestor” relationships between items
  - Example
    - milk  $\Rightarrow$  wheat bread [support = 8%, confidence = 70%]
    - 2% milk  $\Rightarrow$  wheat bread [support = 2%, confidence = 72%]
  - Here, the first rule is an ancestor of the second rule
    - A rule R1 is an ancestor of a rule R2, if R1 can be obtained by replacing the items in R2 by their ancestors in a hierarchy

# Multi-level Association: Redundancy Filtering

- Example
  - milk  $\Rightarrow$  wheat bread [support = 8%, confidence = 70%]
  - 2% milk  $\Rightarrow$  wheat bread [support = 2%, confidence = 72%]
- A rule can be considered redundant if its support is close to the “expected” value, based on the rule’s ancestor, and thus remove that rule
  - First rule says, 8% support and 70% confidence and about one-quarter of milk is ‘2% milk’
  - Second rule is having approx 70% confidence and 2% milk are also samples of milk and a support quarter share in milk (i.e.  $8\% \times \frac{1}{4}$ )
  - So, Rule 2 is not interesting because does not offer any additional information and is less general than Rule 1

# Multi-dimensional Association

- Single-dimensional/Single-predicate/IntraDimensional rules

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

Predicate: buys as a dimension

- Multi-dimensional/Interdimensional rules

- $\geq 2$  dimensions or predicates
  - MD Association, searches for the frequent predicate sets
  - A K-predicate set is a set containing k conjunctive predicates
    - e.g. {age, buys, occupation} is a 3-predicate set

# Multi-dimensional Association

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

age(X,"19-25")  $\wedge$  occupation(X,“student”)  $\Rightarrow$   
buys(X,“coke”)

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

age(X,"19-25")  $\wedge$  buys(X, “popcorn”)  $\Rightarrow$  buys(X,  
“coke”)

# Multi-dimensional Association

- Categorical Attributes
  - Finite number of possible values
  - No ordering among values
  - e.g. brand, color, occupation
  - Also called Nominal attributes
    - Their values are “names of things”
- Quantitative Attributes
  - Numeric
  - Implicit ordering among values
  - e.g. age, income, price

# Multi-dimensional Association

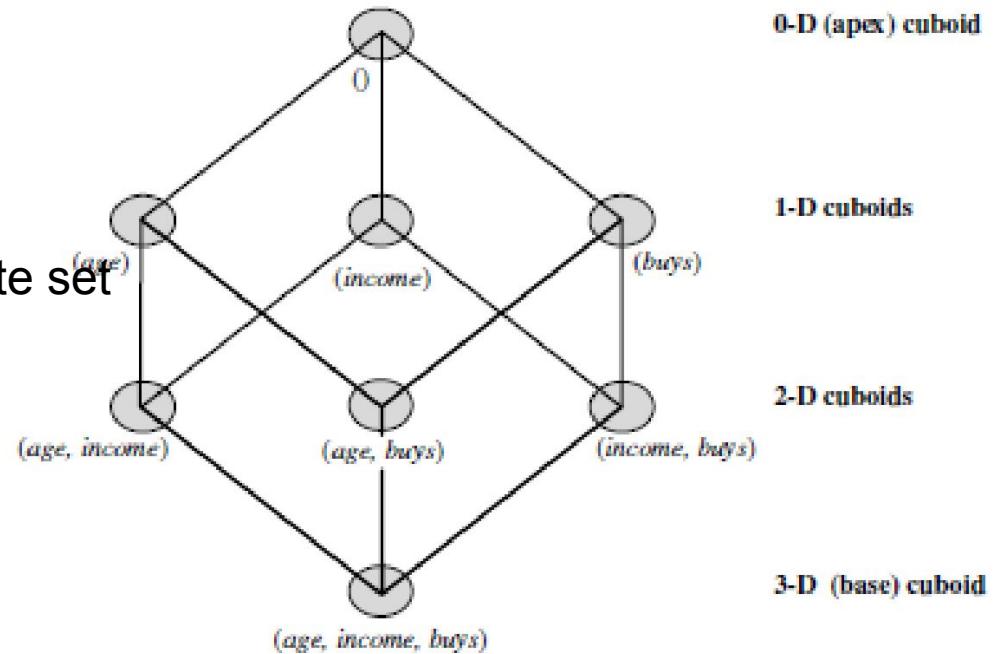
- Techniques for mining MD Asso. regarding the treatment of Quantitative attributes
  - Using Static Discretization of Quantitative Attributes
  - Quantitative attributes are discretized using predefined concept hierarchy
    - Occurs before mining
    - A concept hierarchy for attribute may be used to replace the original numeric values by labels
      - e.g. for income, interval labels as "0...20K", "21K...30K", ...
      - Here, discretization is static and predetermined
    - The discretized numeric attributes, with their interval labels can be treated as categorical attributes (where each interval is considered as a category)

# Multi-dimensional Association

- Techniques for mining MD Asso. regarding the treatment of Quantitative attributes
  - Dynamic Discretization of Quantitative Attributes
  - Quantitative attributes are discretized or clustered into “bins” based on the distribution of the data
    - These bins may be further combined during the mining process
    - Discretization process is dynamic and established so as to satisfy some mining criteria, such as maximizing the confidence of the rules mined

# Multi-dimensional Association

- Differed from the Single-dimensional Association which searches for frequent itemsets
- MD Association, searches for the frequent predicate sets
  - A K-predicate set is a set containing k conjunctive predicates
  - Implemented
    - Data Cube
      - Stores aggregates  
(such as counts)
    - 3-D Cube □
    - N-D Cube for N-predicate set
    - Detail later on



# Mining Various Kinds of Rules or Regularities

- Multi-level
  - At different level of abstraction
- Multi-Dimensional
  - More than one dimension (e.g. item buys with customer age)
- Quantitative association rules
  - Involve numeric attributes that have an implicit ordering among values (e.g. age)
- Constraint Based

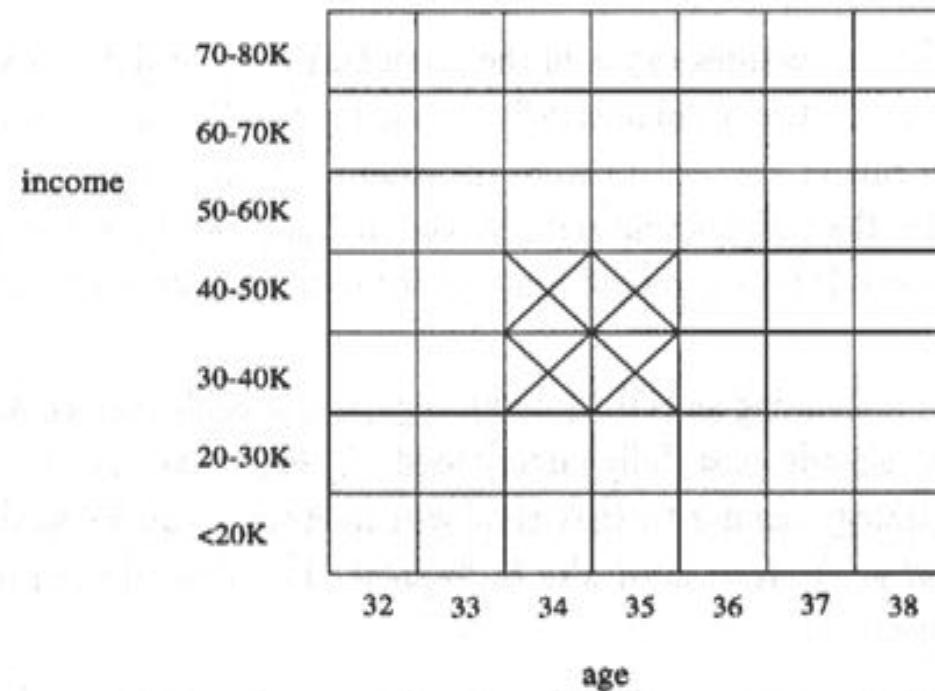
# Mining Quantitative Association Rules

- The Association rule in the form of
  - Quantitative attribute (s)  Categorical attribute (s)
- Example of 2-D Quantitative Association Rule
  - $A_{\text{quan1}} \wedge A_{\text{quan2}} \Rightarrow A_{\text{cat}}$
  - $\text{Age}(X, "34-35") \wedge \text{Income}(X, "30K - 50K") \quad \square \quad \text{Buys}(X, "HDTV")$
- Like Multi Dimension, here also Numeric attributes discretized dynamically

# Mining Quantitative Association Rules

## ■ Association Rule Clustering System (ARCS)

- An approach that Maps pairs of Quantitative attributes onto a 2-D grid for tuples satisfying a given Categorical attribute condition
- The grid is searched for clusters of points from which the association rules are generated



# Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FP-Growth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns



## ■ Mining Association Rule

- Frequent Pattern {A,B}
- Possible Association Rule  $A \square B$ ,  $B \square A$
- Support and confidence of the rule:

$$\text{support}(A \Rightarrow B) = P(A \cup B)$$

$$\text{confidence}(A \Rightarrow B) = P(B | A)$$

- Strong association rule with minimum thresholds.
- No enough!

# Misleading “strong” association rule

- Suppose we are interested in analyzing transactions at *AllElectronics* with respect to the purchase of computer games and videos. Let *game* refer to the transactions containing computer games, and *video* refer to those containing videos. Of the 10,000 transactions analyzed, the data show that 6000 of the customer transactions included computer games, while 7500 included videos, and 4000 included both computer games and videos. Suppose that a data mining program for discovering association rules is run on the data, using a minimum support of, say, 30% and a minimum confidence of 60%. The following association rule is discovered:

$\text{buys}(X, \text{"computer games"}) \Rightarrow \text{buys}(X, \text{"videos"})$

- P(videos) [support = 40%, confidence = 66%].

# From Association Rule to Correlation Rule

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

**Lift** is a simple correlation measure that is given as follows. The occurrence of itemset  $A$  is **independent** of the occurrence of itemset  $B$  if  $P(A \cup B) = P(A)P(B)$ ; otherwise, itemsets  $A$  and  $B$  are **dependent** and **correlated** as events. This definition can easily be extended to more than two itemsets. The **lift** between the occurrence of  $A$  and  $B$  can be measured by computing

$$\text{lift}(A, B) = \frac{P(A \cup B)}{P(A)P(B)}. \quad (6.8)$$

- Life=1: independent
- Life>1: positive correlated
- Life<1: negetive correlated

**Table 6.6**  $2 \times 2$  Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

	<i>game</i>	$\overline{\text{game}}$	$\Sigma_{\text{row}}$
<i>video</i>	4000	3500	7500
$\overline{\text{video}}$	2000	500	2500
$\Sigma_{\text{col}}$	6000	4000	10,000

- $P(V) = .75$
- $P(G) = .6$
- $P(VG) = .40$
- $\text{Life}(VG) = .40 / (.75 * .6) < 1$
- Negative correlated

# Interestingness Measure: Correlations ( $\chi^2$ )

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

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

	game	$\overline{game}$	$\Sigma_{row}$
video	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
$\Sigma_{col}$	6,000	4,000	10,000

$$e_{ij} = \frac{count(A = a_i)count(B = b_j)}{N}$$

$$\begin{aligned}\chi^2 &= \sum \frac{(observed - expected)^2}{expected} = \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} \\ &\quad + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000} = 555.6.\end{aligned}$$

## Interestingness Measure: Correlations ( $\chi^2$ )

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

# Interestingness Measure: Correlations (all\_confidence)

$$all\_conf(A, B) = \frac{sup(A \cup B)}{\max\{sup(A), sup(B)\}} = \min\{P(A|B), P(B|A)\},$$

	<i>game</i>	$\overline{game}$	$\Sigma_{row}$
<i>video</i>	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
$\Sigma_{col}$	6,000	4,000	10,000

- All\_confidence(g,v)=0.53

# Interestingness Measure: Correlations (max confidence)

$$\text{max\_conf}(A, B) = \max\{P(A | B), P(B | A)\}.$$

	<i>game</i>	$\overline{\text{game}}$	$\Sigma_{\text{row}}$
<i>video</i>	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{\text{video}}$	2,000 (1,500)	500 (1,000)	2,500
$\Sigma_{\text{col}}$	6,000	4,000	10,000

## Interestingness Measure: Correlations (cosine)

$$\text{cosine}(A, B) = \frac{P(A \sqcap B)}{\sqrt{P(A)P(B)}}$$

	<i>game</i>	$\overline{\text{game}}$	$\Sigma_{\text{row}}$
<i>video</i>	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{\text{video}}$	2,000 (1,500)	500 (1,000)	2,500
$\Sigma_{\text{col}}$	6,000	4,000	10,000

- $\text{cosine}(g, v) = 0.6$

	$milk$	$\overline{milk}$	$\Sigma_{row}$
$coffee$	$mc$	$\overline{mc}$	$c$
$\overline{coffee}$	$m\bar{c}$	$\overline{m\bar{c}}$	$\bar{c}$
$\Sigma_{col}$	$m$	$\overline{m}$	$\Sigma$

- Null-transactions: is a transaction that does
- not contain any of the itemsets being examined
- A measure is null-invariant if its value
- Is free from the influence of null-transactions.

Data Set	$mc$	$\overline{mc}$	$m\bar{c}$	$\overline{m\bar{c}}$	$all\_conf.$	$cosine$	$lift$	$X^2$
$A_1$	1,000	100	100	100,000	0.91	0.91	83.64	83,452.6
$A_2$	1,000	100	100	10,000	0.91	0.91	9.26	9,055.7
$A_3$	1,000	100	100	1,000	0.91	0.91	1.82	1,472.7
$A_4$	1,000	100	100	0	0.91	0.91	0.99	9.9
$B_1$	1,000	1,000	1,000	1,000	0.50	0.50	1.00	0.0
$C_1$	100	1,000	1,000	100,000	0.09	0.09	8.44	670.0
$C_2$	1,000	100	10,000	100,000	0.09	0.29	9.18	8,172.8
$C_3$	1	1	100	10,000	0.01	0.07	50.0	48.5

# Constraint-Based Mining

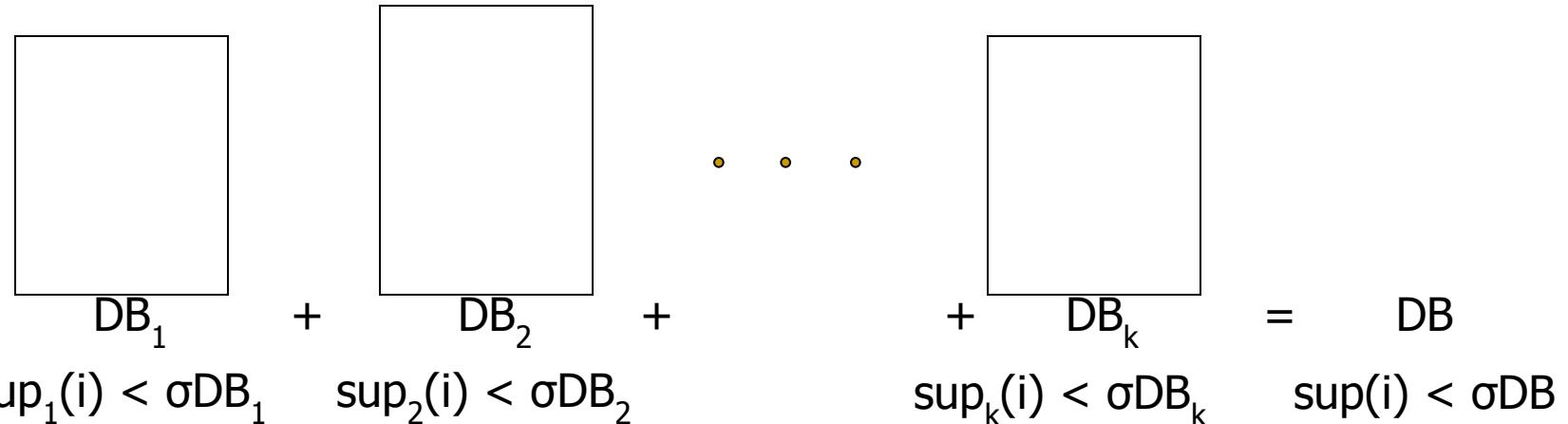
- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: Classification, Association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98
  - Dimension/level constraints:
    - In relevance to region, price, brand, customer category
  - Rule constraints
    - Small sales (price < \$10) triggers big sales (sum > \$200)
  - Interestingness constraints:
    - Strong rules ( $\text{min\_support} \geq 3\%$ ,  $\text{min\_confidence} \geq 60\%$ )

# Apriori Algorithm

- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

# Partition: Scan Database Only Twice

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



# Sampling for Frequent Patterns

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

# Association Rules - Summary

- Basic Concepts
  - Frequent Itemsets and Association Rules
  - Measures
- Applications of frequent pattern and associations
  - Market Basket
  - Weblog mining
  - Bioinformatics
- Efficient and Scalable Frequent Itemset Mining Methods
  - The Apriori Algorithm
    - Finding Frequent Itemsets Using Candidate Generation
    - Generating Association Rules from Frequent Itemsets
  - Improving the Efficiency of Apriori
  - Mining Frequent Itemsets without Candidate Generation
  - Mining Frequent Itemsets Using Vertical Data Format
- Are All the Pattern Interesting?—Pattern Evaluation Methods
  - Strong Rules Are Not Necessarily Interesting
  - From Association Analysis to Correlation Analysis
  - Selection of Good Measures for Pattern Evaluation
- Sequential Pattern Mining

# Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

# Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing:02.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
- G. Liu, H. Lu, W. Lou, J. X. Yu. On Computing, Storing and Querying Frequent Patterns. KDD'03.
- G. Grahe and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003

# Ref: Vertical Format and Row Enumeration Methods

- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. DAMI:97.
- Zaki and Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.
- C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. KDD'02.
- F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki , CARPENTER: Finding Closed Patterns in Long Biological Datasets. KDD'03.
- H. Liu, J. Han, D. Xin, and Z. Shao, Mining Interesting Patterns from Very High Dimensional Data: A Top-Down Row Enumeration Approach, SDM'06.

# Ref: Mining Correlations and Interesting Rules

- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97.
- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98.
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02.
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03.
- T. Wu, Y. Chen and J. Han, “Association Mining in Large Databases: A Re-Examination of Its Measures”, PKDD'07

# Ref: Freq. Pattern Mining Applications

- Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. ICDE'98.
- H. V. Jagadish, J. Madar, and R. Ng. Semantic Compression and Pattern Extraction with Fascicles. VLDB'99.
- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.
- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT'02.

# Mine the Closed and Max-Patterns

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

Min\_sup = 2

Min\_conf = 50%