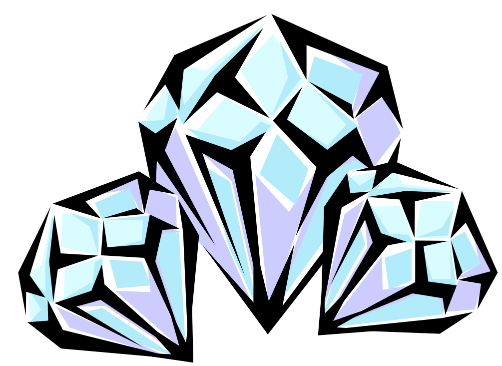
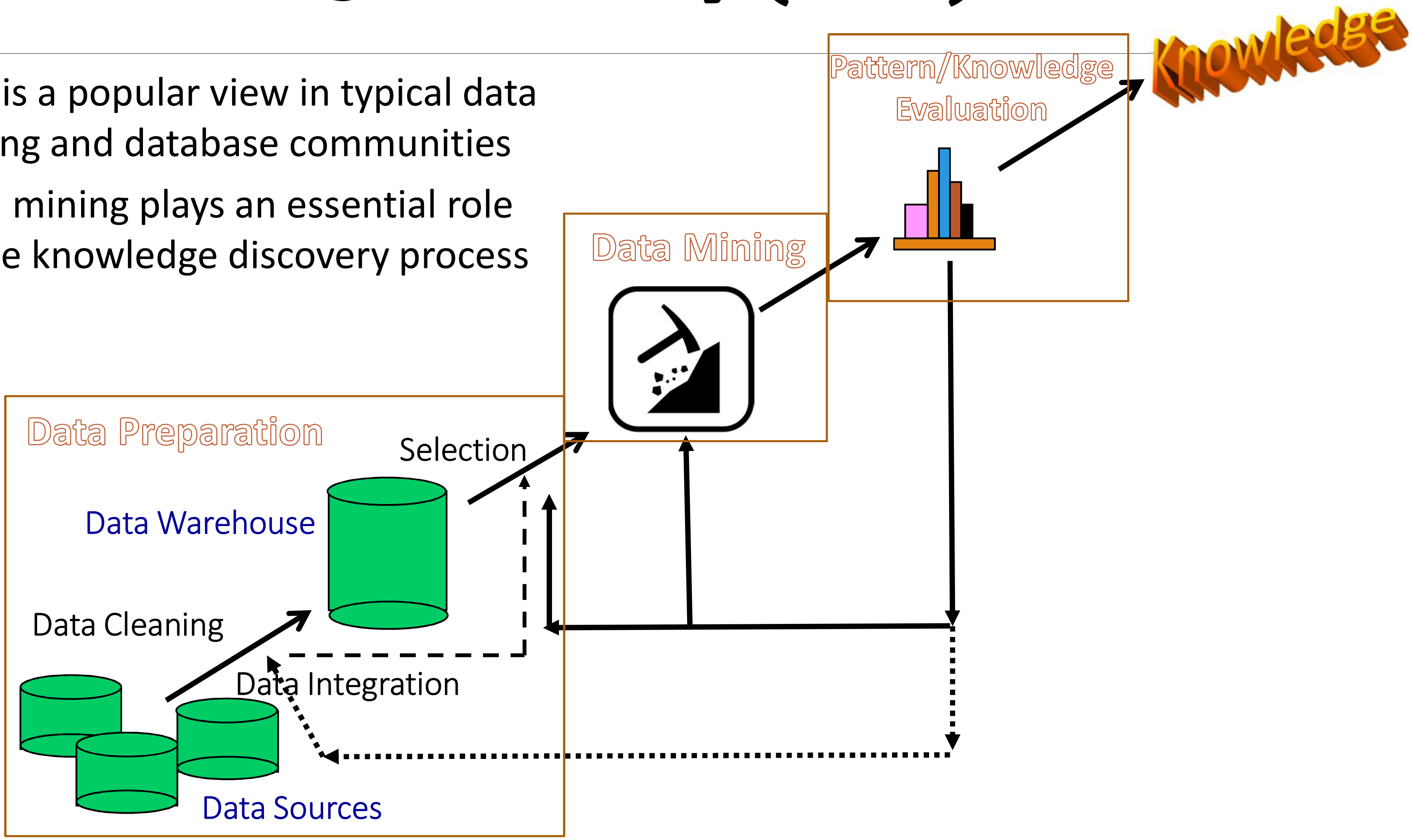
What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (<u>non-trivial,</u> <u>implicit</u>, <u>previously unknown</u> and <u>potentially useful)</u> patterns or knowledge from huge amount of data
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.

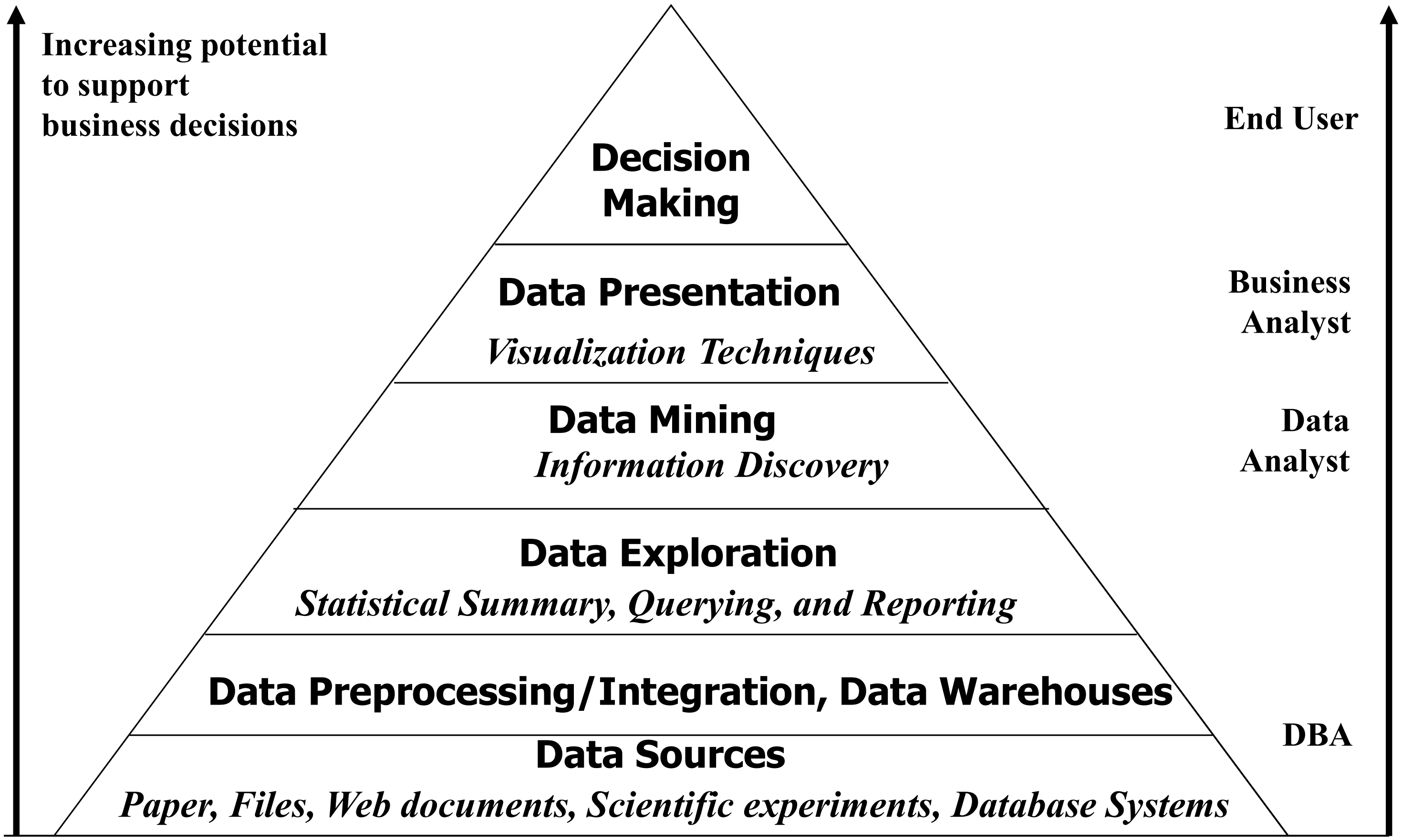


Knowledge Discovery (KDD) Process

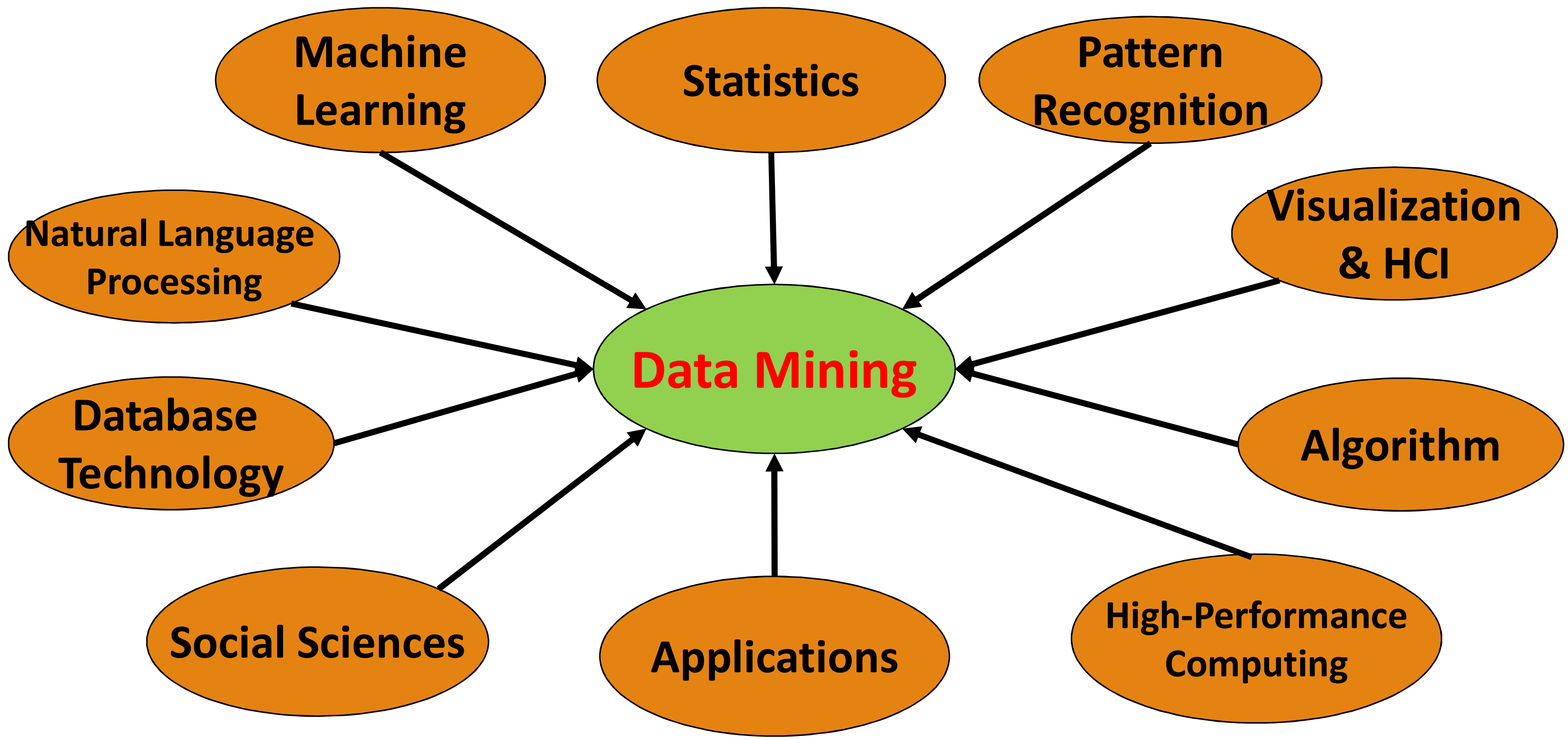
- This is a popular view in typical data mining and database communities
- Data mining plays an essential role in the knowledge discovery process



Data Mining in Business Intelligence



Data Mining: Confluence of Multiple Disciplines



Recommender System

- Product recommendation (Amazon, EBay)
- Search recommendation (Google, Bing)
- Video/music/post recommendation (Netflix, Pandora, Pinterst)
- Friend recommendation (Facebook, twitter)
- Job recommendation (linkIn)

- collaborative filtering
- content-based filtering
- hybrid

Commerce, Profiling and Finance

- Planning and Forecasting
  - Dynamic pricing
  - Ads bidding
- Profiling
  - User profiling
    - Churn Prediction: knowing which users are going to stop using your platform in the future.
  - Product profiling
- Fintech
  - Stock market
    - Sentiment analysis

Urban Planning

- Energy and power
- Traffic prediction and management
  - Parking detection
  - Traffic control
- Transportation sharing system
  - Uber
  - Bike-sharing
- Pollution
  - Air quality prediction

Medicine and Healthcare

- Disease prediction
  - Computer Aided Detection
  - EHR
  - Risk prediction
  - Disease progression prediction
- Healthcare
  - Epidemic and outbreak prediction
  - Food safety
- Medicine study
  - Drug discovery and prediction
- Bioinformatics

Other Sciences and Applications

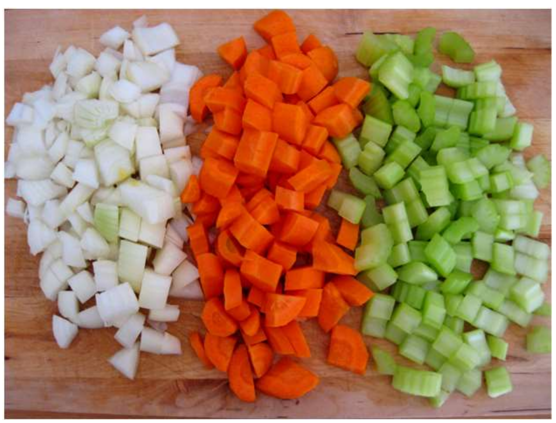
- Education
  - MOOC (massive open online course)
- Political science and Social science
  - Fake news
  - Crime and terrorist detection
  - Disaster detection
  - Opinion mining
  - Social influence
- Environmental Science
  - Climate

Pattern Discovery: Basic Concepts

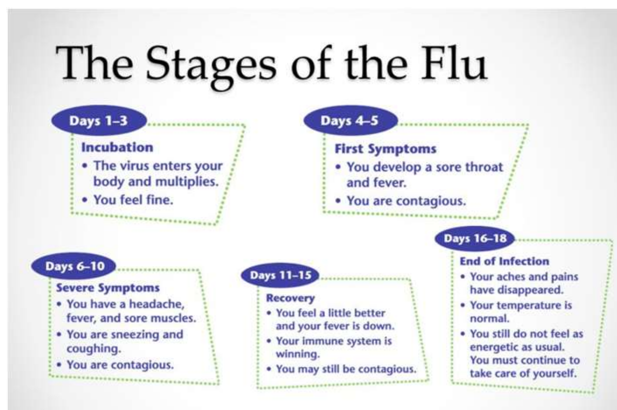
- What Is Pattern Discovery?   Why Is It Important?
- Basic Concepts: Frequent Patterns and Association Rules
- Compressed Representation: Closed Patterns and Max-Patterns

What are Patterns?

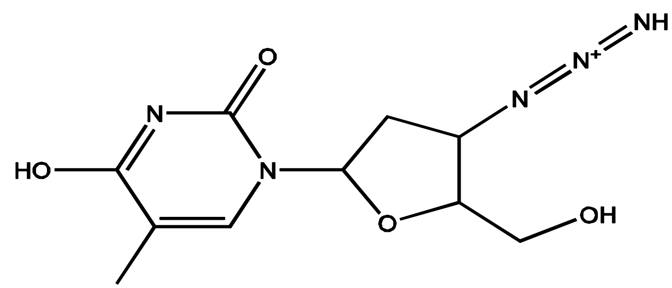
- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets



Frequent item set



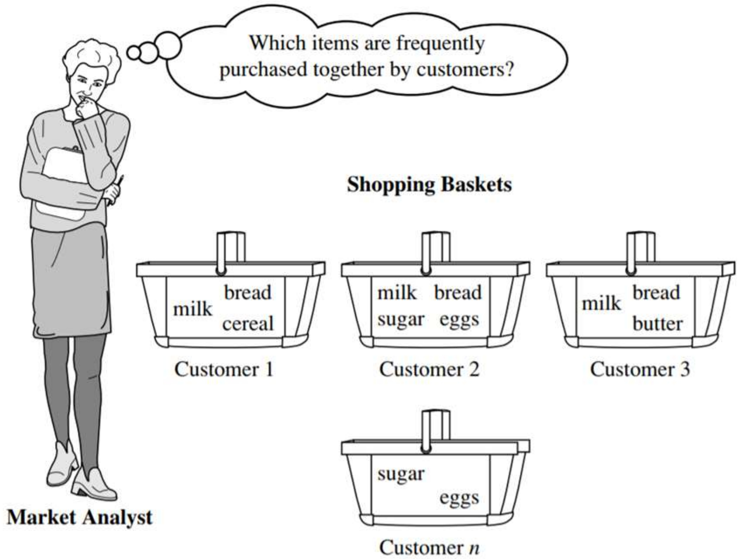
Frequent sequences



Frequent structures

What Is Pattern Discovery?

- Pattern discovery: Uncovering patterns from massive data sets
- It can answer questions such as:
  - What products were often purchased together?
  - What are the subsequent purchases after buying an iPad?



Pattern Discovery: Why Is It Important?

- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining **sequential**, structural (e.g., sub-graph) patterns
- Broad applications
  - Market basket analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, biological sequence analysis
  - Many types of data: spatiotemporal, multimedia, time-series, and stream data

Basic Concepts: Transactional Database

- Transactional Database (TDB)
  - Each transaction is associated with an identifier, called a TID.
  - May also have counts associated with each item sold

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

Basic Concepts: k-Itemsets and Their Supports

- Itemset: A set of one or more items

  $$\mathrm{I} = \{I_1, I_2, \cdots, I_m\}$$

- k-itemset: An itemset containing k items:

  $X = \{x_1, \ldots, x_k\}$

  - Ex. {Beer, Nuts, Diaper} is a 3-itemset

- Absolute support (count)
  - sup{X} = occurrences of an itemset X
    - Ex. sup{Beer} = 3
    - Ex. sup{Diaper} = 4
    - Ex. sup{Beer, Diaper} = 3
    - Ex. sup{Beer, Eggs} = 1

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

- Relative support
  - *s*{*X*} = The fraction of transactions that contains X (i.e., the probability that a transaction contains X)
  - Ex. s{Beer} = 3/5 = 60%
  - Ex. s{Diaper} = 4/5 = 80%
  - Ex. s{Beer, Eggs} = 1/5 = 20%

Basic Concepts: Frequent Itemsets (Patterns)

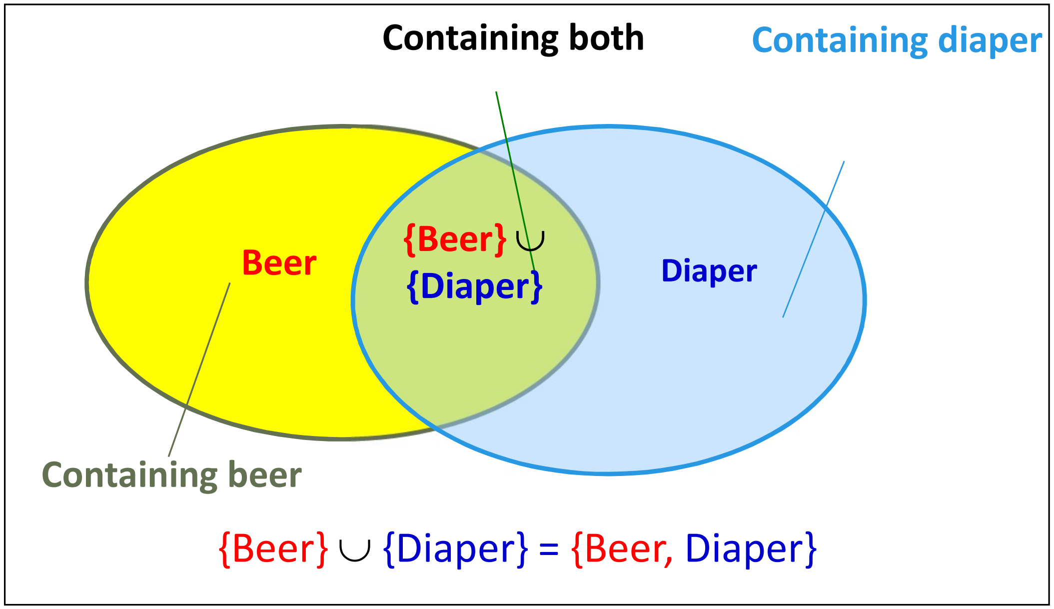
- An itemset (or a pattern) X is *frequent* if the support of X is no less than a *minsup* threshold σ
- Let σ = *50%* (σ: *minsup* threshold) for the given 5-transaction dataset
  - All the frequent 1-itemsets:
    - Beer: 3/5 (60%); Nuts: 3/5 (60%); Diaper: 4/5 (80%); Eggs: 3/5 (60%)
  - All the frequent 2-itemsets:
    - {Beer, Diaper}: 3/5 (60%)
  - All the frequent 3-itemsets?
    - None

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

- Why do these itemsets (shown on the left) form the complete set of frequent *k*-itemsets (patterns) for any *k*?
- **Observation**: We may need an efficient method to mine a complete set of frequent patterns

From Frequent Itemsets to Association Rules

- Compared with itemsets, association rules can be more telling
  - Ex. *Diaper → Beer*
    - *Buying diapers may likely lead to buying beers*



Note: X ∪ Y: the union of two itemsets

- The set contains both X and Y

Association Rules

- How do we compute the strength of an association rule $X \rightarrow Y$ (Both *X* and *Y* are itemsets)?
- We first compute the following two metrics, s and c.
  - Support of $X \cup Y$
    - Ex. s{Diaper, Beer} = 3/5 = 0.6 (i.e., 60%)
  - Confidence of $X \rightarrow Y$
    - The *conditional probability* that a transaction containing X also contains *Y:*

      c = sup(*X, Y)* / sup(X)
    - Ex. *c* = sup{Diaper, Beer}/sup{Diaper} = ¾ = 0.75

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

- In pattern analysis, we are often interested in those rules that dominate the database, and these two metrics ensure the popularity and correlation of X and Y.

Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c) such that s ≥ *minsup* and c ≥ *minconf*
- Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3
- Let *minconf = 50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

(Q: Are these all rules?)

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

- **Observations:**
  - Mining association rules and mining frequent patterns are very close problems
  - Scalable methods are needed for mining large datasets

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain (minsup = 2)?
  - $TDB_{1:}$ T_1: {a_1, …, a_{50}}; T_2: {a_1, …, a_{100}}
  - Let's have a try

1-itemsets: {a_1}: 2, {a_2}: 2, …, {a_{50}}: 2

2-itemsets: {a_1, a_2}: 2, …, {a_1, a_{50}}: 2, {a_2, a_3}: 2 …, …, {a_{49}, a_{50}}: 2,

…, …, …, …

49-itemsets: {a_1, a_2, …, a_{49}}: 2, …, {a_2, a_3, …, a_{50}}: 2

50-itemset: {a_1, a_2, …, a_{50}}: 2

A too huge set for any one to compute or store!

- The total number of frequent itemsets:

$$\binom{50}{1} + \binom{50}{2} + \cdots + \binom{50}{50} = 2^{50} - 1$$

Expressing Patterns in Compressed Form

- How to reduce the redundancy of the list of all the frequent itemsets?

- Solution 1: **Closed patterns**: A pattern (itemset) X is closed if X is *frequent,* and there exists *no super-pattern* Y ⊃ X, *with the same support* as X
  - Ex. TDB_1: T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$; T_3: $\{a_1, …, a_{10}\}$
  - Suppose *minsup* = 2. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, …, a_{50}\}$: 2"; P_2: "$\{a_1, …, a_{10}\}$: 3"

Expressing Patterns in Compressed Form: Closed Patterns

- Closed pattern is a lossless compression of frequent patterns
  - Lossless: no information loss
  - Reduces the # of patterns but does not lose the support information!
  - Given P_1: "{a_1, …, a_{50}}: 2"; P_2: "{a_1, …, a_{10}}: 3";
  - You will still be able to say: "{a_2, …, a_{40}}: 2", "{a_5, a_1}: 3"

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$; T_3: $\{a_1, …, a_{10}\}$
  - Suppose *minsup* = 2. How many max-patterns does TDB_1 contain?
    - One: P: "$\{a_1, …, a_{50}\}$: 2"

Expressing Patterns in Compressed Form: Max-Patterns

- Max-pattern is a lossy compression!
  - We only any subset of the max-pattern P:{a_1, …, a_{50}} is frequent
  - But we do not know the real support of {a_1, …, a_{10}}, …, any more!
  - More compressed than closed pattern (that is smaller in size)

The Downward Closure Property of Frequent Patterns

- **Frequent** itemset: $\{a_1, \ldots, a_{50}\}$
  - Subsets are all **frequent**: $\{a_1\}, \{a_2\}, \ldots, \{a_{50}\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{49}\}, \ldots$
- Downward closure (Apriori): Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - If any subset of an itemset S is **infrequent**, then there is no chance for S to be frequent.

A sharp knife for pruning!

Apriori: A Candidate Generation & Test Approach

- Outline of Apriori (level-wise, candidate generation and test)
  - Scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-(k+1) candidate itemsets from length-k frequent itemsets
    - Test the candidates against DB to find frequent (k+1)-itemsets
    - Set k := k +1
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

The Apriori Algorithm—An Example

minsup = 2

Database TDB

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

1st scan

C_1

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

F_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}

2nd scan

C_2

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

F_2

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

C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k

F_k : Frequent itemset of size k

K := 1;

F_k := {frequent items}; // frequent 1-itemset

While (F_k != ∅) **do {** // when F_k is non-empty

C_{k+1} := candidates generated from F_k; // candidate generation

Derive F_{k+1} by counting candidates in C_{k+1} with respect to *TDB* at minsup;

k := k + 1

}

return $\cup_k F_k$ // return F_k generated at each level

Candidate Generation (Pseudo-Code)

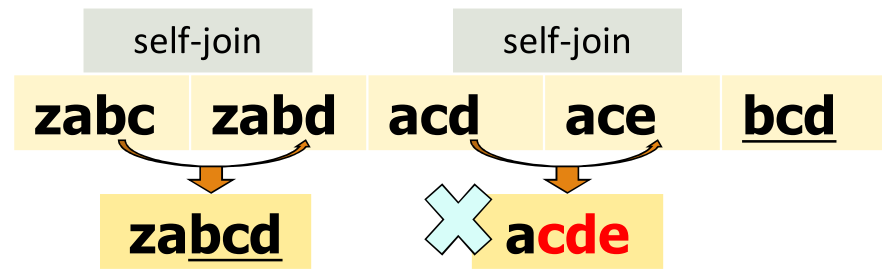
- Suppose the items in F_{k-1} are listed in an order
- // Step 1: Joining

  for each p **in** F_{k-1}

      for each q **in** F_{k-1}

          if $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$ **{**

              c = **join**(p, q)



- // Step 2: pruning

              if has_infrequent_subset(c, F_{k-1})

                  continue // prune

              else add c **to** C_k

          }

Partitioning: Scan Database Only Twice

- Theorem: *Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB*

Here is the proof!

TDB_1 + TDB_2 + ... + TDB_k = TDB

$s_1(X) < \sigma|TDB_1|$ ... $s_2(X) < \sigma|TDB_2|$ ... $s_k(X) < \sigma|TDB_k|$ ... $s(X) < \sigma|TDB|$

Partitioning: Scan Database Only Twice

- Method: Scan DB **twice** (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)
  - Scan 1: Partition database so that each partition can fit in main memory (why?)
    - Mine local frequent patterns in this partition
  - Scan 2: Consolidate global frequent patterns
    - Find global frequent itemset candidates (those frequent in at least one partition)
    - Find the true frequency of those candidates, by scanning TDB_i one more time

Why Mining Frequent Patterns by Pattern Growth?

- Apriori: A ***breadth-first search*** mining algorithm
  - First find the complete set of frequent k-itemsets
  - Then derive frequent (k+1)-itemset candidates
  - Scan DB again to find true frequent (k+1)-itemsets
- Are there depth-first search algorithm?
  - Yes