Unit-2

Data Mining And Analytics (SRM Institute of Science and Technology)

DATA MINING AND ANALYTICS

- UNIT II

Unit ii

Course Code	18CSE355T	Course Name	DATA MINING AND ANALYTICS	Course Category	E	Professional Elective	L	T	P	C
							3	0	0	3

Pre-requisite Courses	Nil	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department	CSE	Data Book / Codes/Standards	Nil		

Course Learning Rationale (CLR):	The purpose of learning this course is to:	Learning		
		1	2	3
		Level of Thinking (Bloom)	Expected Proficiency (%)	Expected Attainment (%)
CLR-1 :	Understand the concepts of Data Mining			
CLR-2 :	Familiarize with Association rule mining			
CLR-3 :	Familiarize with various Classification algortihms			
CLR-4 :	Understand the concepts of Cluster Analysis			
CLR-5 :	Familiarize with Outlier analysis techniques			
CLR-6 :	Familiarize with applications of Data mining in different domains			

Course Learning Outcomes (CLO):	At the end of this course, learners will be able to:	Learning		
		1	2	3
		Level of Thinking (Bloom)	Expected Proficiency (%)	Expected Attainment (%)
CLO-1 :	Gain knowledge about the concepts of Data Mining	2	80	85
CLO-2 :	Understand and Apply Association rule mining techniques	2	75	80
CLO-3 :	Understand and Apply various Classification algorithms	2	85	80
CLO-4 :	Gain knowledge on the concepts of Cluster Analysis	2	80	75
CLO-5 :	Gain knowledge on Outlier analysis techniques	2	75	85
CLO-6 :	Understand the importance of applying Data mining concepts in different domains	2	80	85

Program Learning Outcomes (PLO)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Engineering Knowledge	Problem Analysis	Design & Development	Analysis, Design, Research	Modern Tool Usage	Society & Culture	Environment & Sustainability	Ethics	Individual & Team Work	Communication	Project Mgt. & Finance	Life Long Learning	PSO - 1	PSO - 2	PSO – 3
CLO-1 :															
CLO-2 :															
CLO-3 :															
CLO-4 :															
CLO-5 :															
CLO-6 :															

Duration (hour)		9	9	9	9	9
S-1	SLO-1	Why Data mining? What is Data mining ?	Mining frequent patterns: Basic concepts	Classification: Basic concepts	Cluster Analysis: Introduction	Outliers: Introduction
	SLO-2	Kinds of data meant for mining	Market Basket Analysis	General approach to Classification	Requirements and overview of different categories	Challenges of outlier detection
S-2	SLO-1	Kinds of patterns that can be mined	Frequent itemsets, Closed itemsets	Decision tree induction	Partitioning method: Introduction	Outlier detection methods: Introduction
	SLO-2	Applications suitable for data mining	Association rules-Introduction	Algorithm for Decision tree induction	k-means	Supervized and Semi-supervized methods
S-3	SLO-1	Issues in Data mining	Apriori algorithm-theoritical approach	Numerical example for Decision tree induction	k-medoids	Unsupervized methods
	SLO-2	Data objects and Attribute types	Apply Apriori algorithm on dataset-1	Attribute selection measure	Hierarchical method: Introduction	
S-4	SLO-1	Statistical descriptions of data	Apply Apriori algorithm on dataset-2	Tree pruning	Agglomerative vs. Divisive method	Statistical and Proximity based methods
	SLO-2		Generating Association rules from frequent itemsets	Scalability and Decision tree induction	Distance measures in algorithmic methods	
S-5	SLO-1	Need for data preprocessing and data quality	Improving efficiency of Apriori	Bayes' Theorem	BIRCH technique	Statistical approaches
	SLO-2			Naïve Bayesian Classification		
S-6	SLO-1	Data cleaning	Pattern growth approach	IF-THEN rules for classification	DBSCAN technique	Statistical data mining
	SLO-2	Data integration		Rule extraction from a decision tree		
S-7	SLO-1	Data reduction	Mining frequest itemsets using Vertical data format	Metrics for evaluating classifier performance	STING technique	Data mining and recommender systems
	SLO-2		Strong rules vs. weak rules	Cross validation		
S-8	SLO-1	Data transformation	Association analysis to Correlation analysis	Bootstrap	CLIQUE technique	Data mining for financial data analysis
	SLO-2			Ensemble methods-Introduction		
S-9	SLO-1	Data cube and its usage	Comparison of pattern evaluation measures	Bagging and Boosting	Evaluation of clustering techniques	Data mining for Intrusion detection
	SLO-2			Random Forests: Introduction		

Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

What Is Frequent Pattern Analysis?

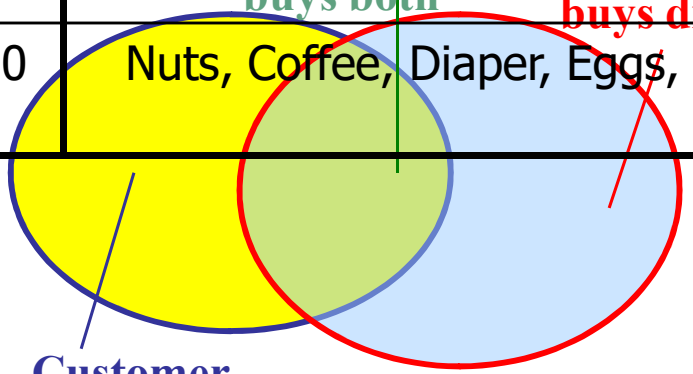
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What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

Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns

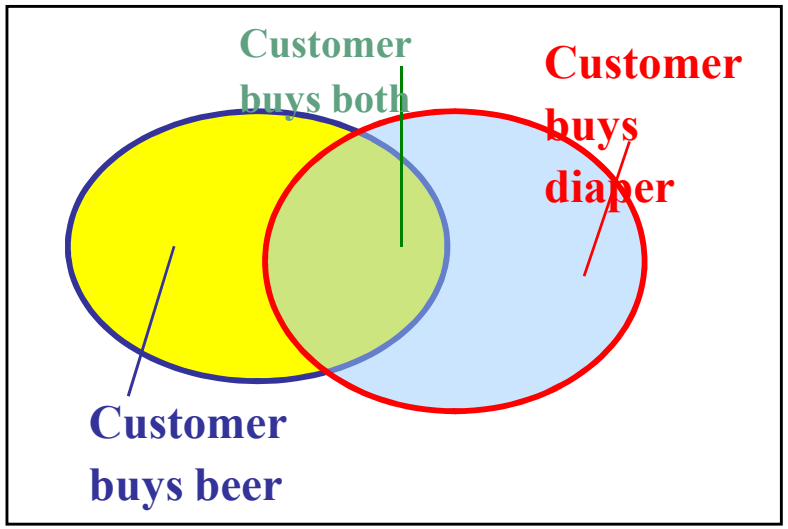
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{1}{1}\binom{0}{0}\binom{0}{0} = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y Ↄ X, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y Ↄ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.
- What is the set of closed itemset?
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Computational Complexity of Frequent Itemset Mining

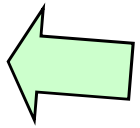
- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions
- The worst case complexty vs. the expected probability
  - Ex. Suppose Walmart has 10^4 kinds of products
    - The chance to pick up one product 10^{-4}
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?

Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach



- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format

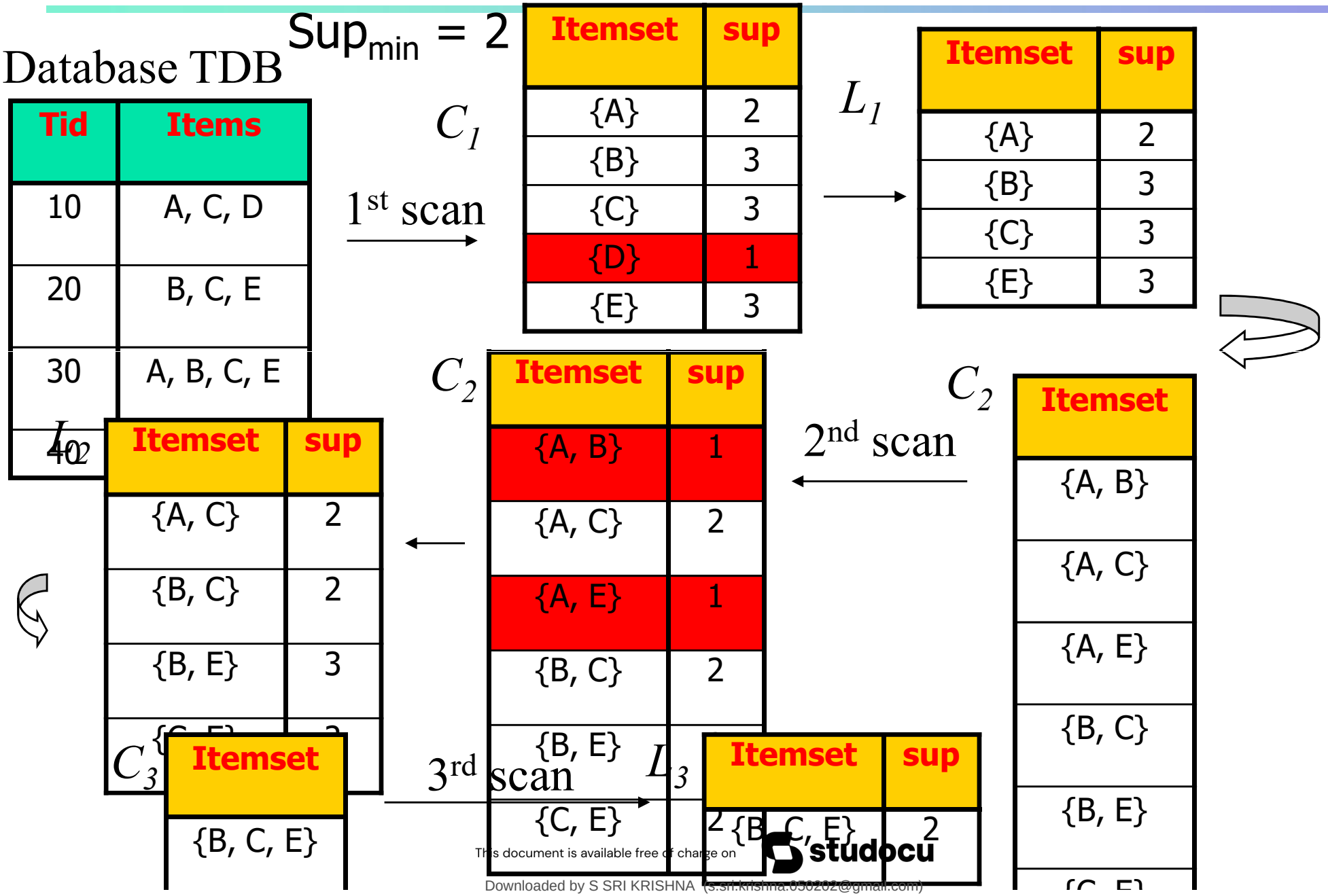
The Downward Closure Property and Scalable Mining Methods

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

Apriori: A Candidate Generation & Test Approach

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

The Apriori Algorithm—An Example



The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
    C_{k+1} = candidates generated from L_k;
    for each transaction t in database do
        increment the count of all candidates in C_{k+1} that are contained in t
    L_{k+1} = candidates in C_{k+1} with min_support
    end
return $\cup_k L_k$;

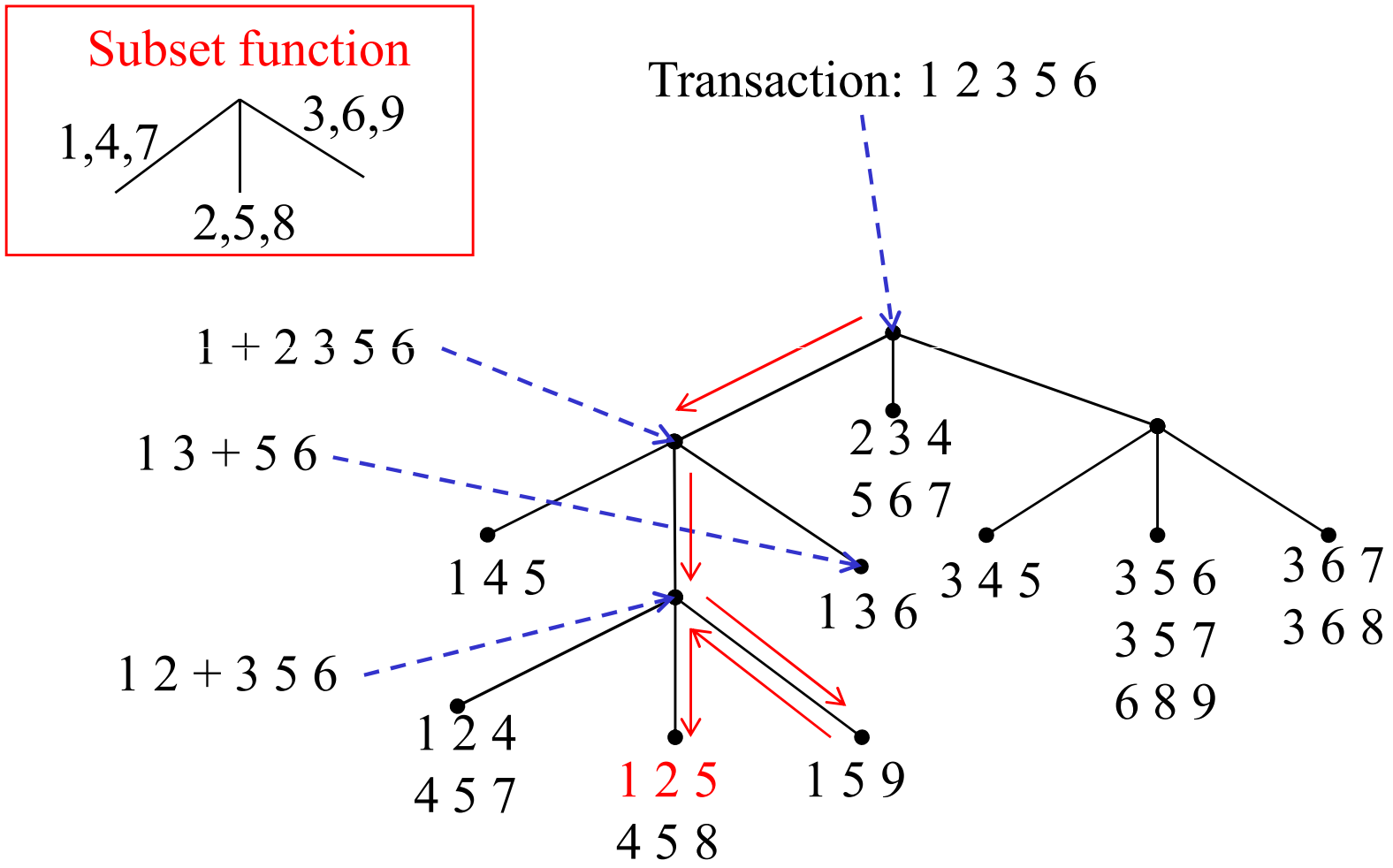
Implementation of Apriori

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

How to Count Supports of Candidates?

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

Counting Supports of Candidates Using Hash Tree



Candidate Generation: An SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in L_{k-1} are listed in an order
  - Step 1: self-joining L_{k-1}

    insert into C_k

    select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

    from $L_{k-1}\ p, L_{k-1}\ q$

    where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
  - Step 2: pruning

    forall ***itemsets c in*** C_k do

    forall ***(k-1)-subsets s of c*** do

    if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k
- Use object-relational extensions like UDFs, BLOBs, and Table functions for efficient implementation [See: S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98]

Scalable Frequent Itemset Mining Methods

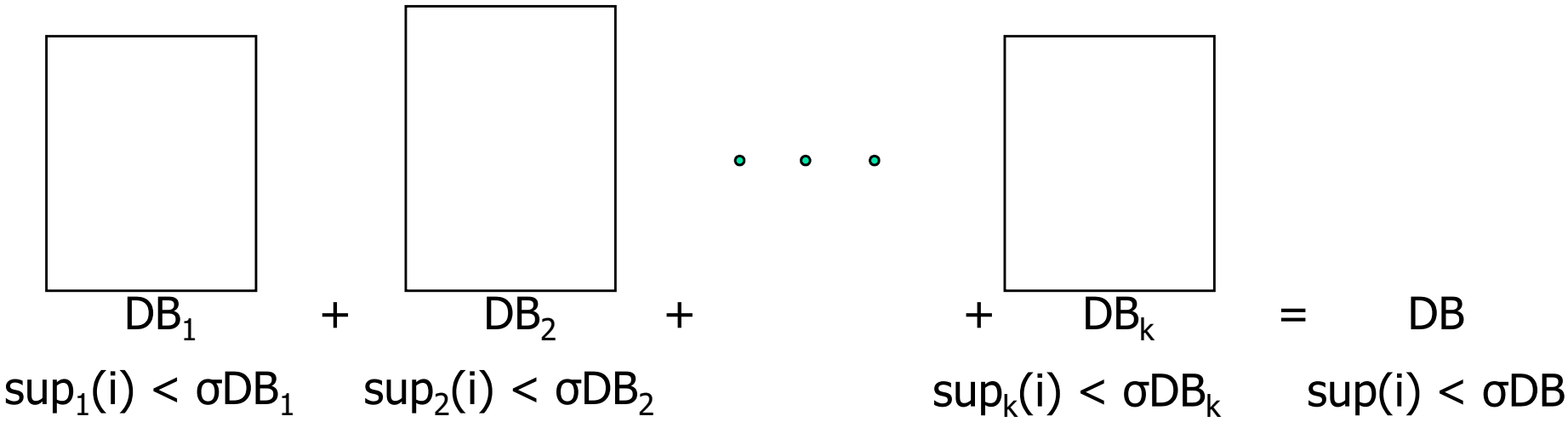
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Further Improvement of the Apriori Method

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

Partition: Scan Database Only Twice

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



DHP: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries
    - {ab, ad, ae}
    - {bd, be, de}
    - …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
. . .	. . .
102	{yz, qs, wt}

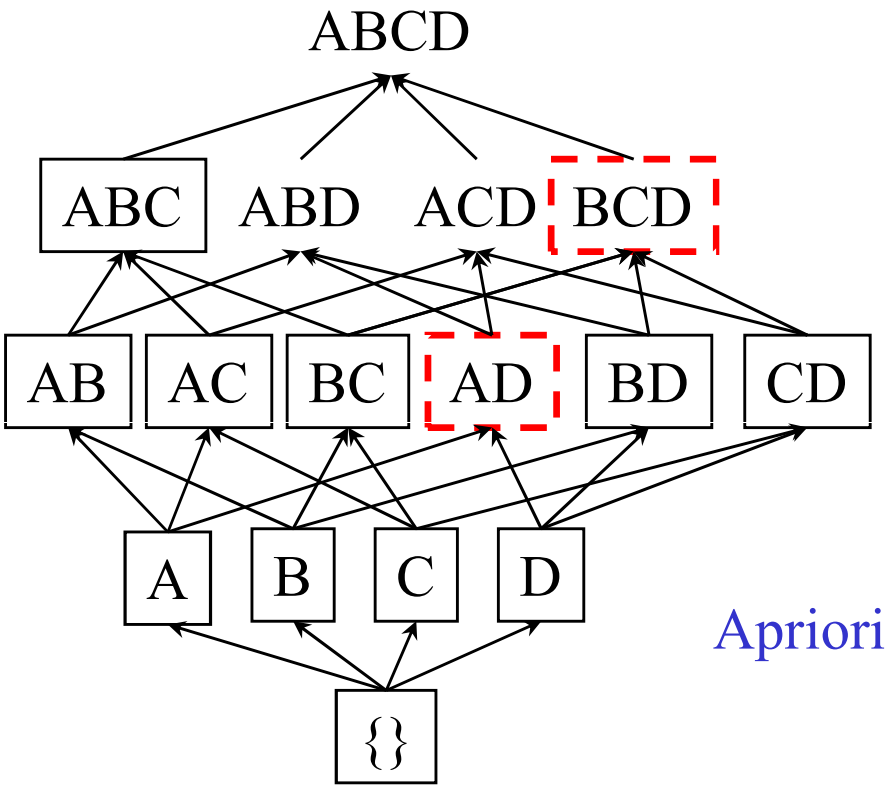
Hash Table

- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

Sampling for Frequent Patterns

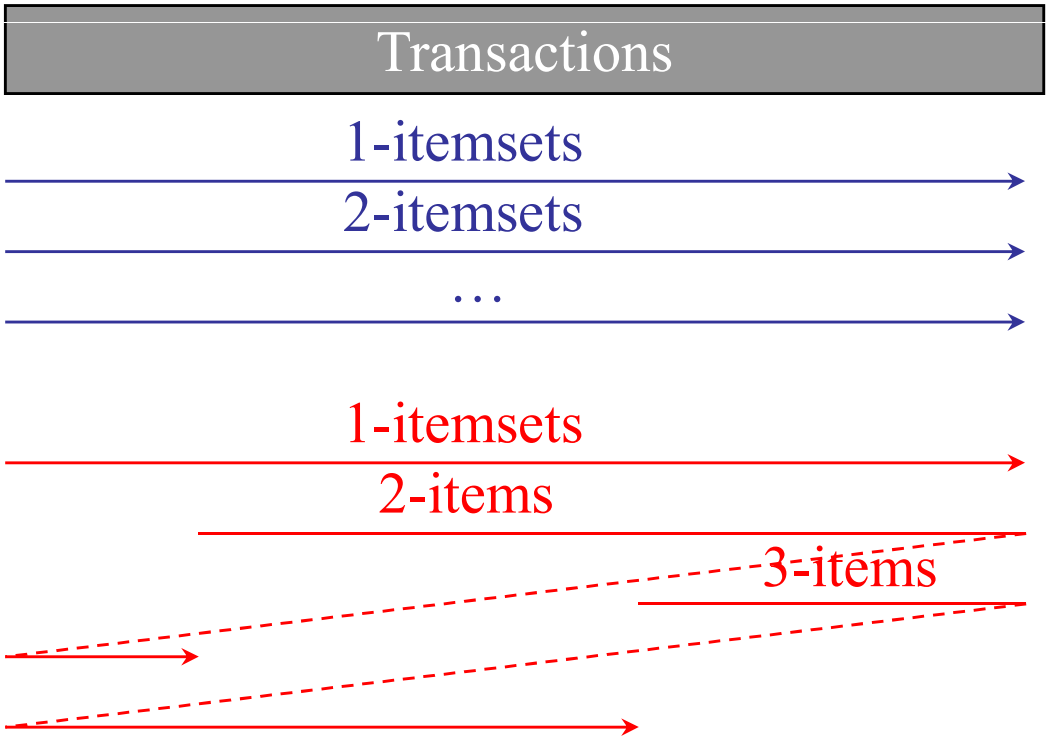
- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find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

DIC: Reduce Number of Scans



Itemset lattice

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



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

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

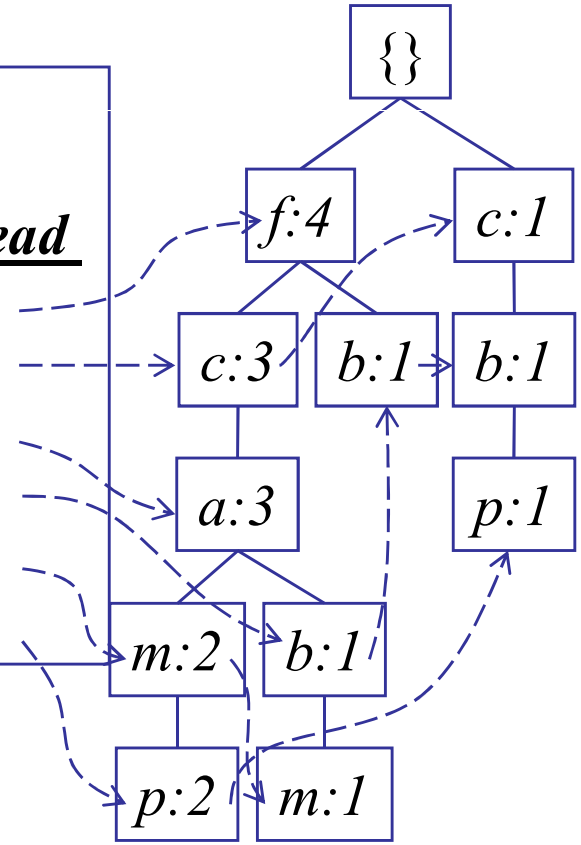
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

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

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



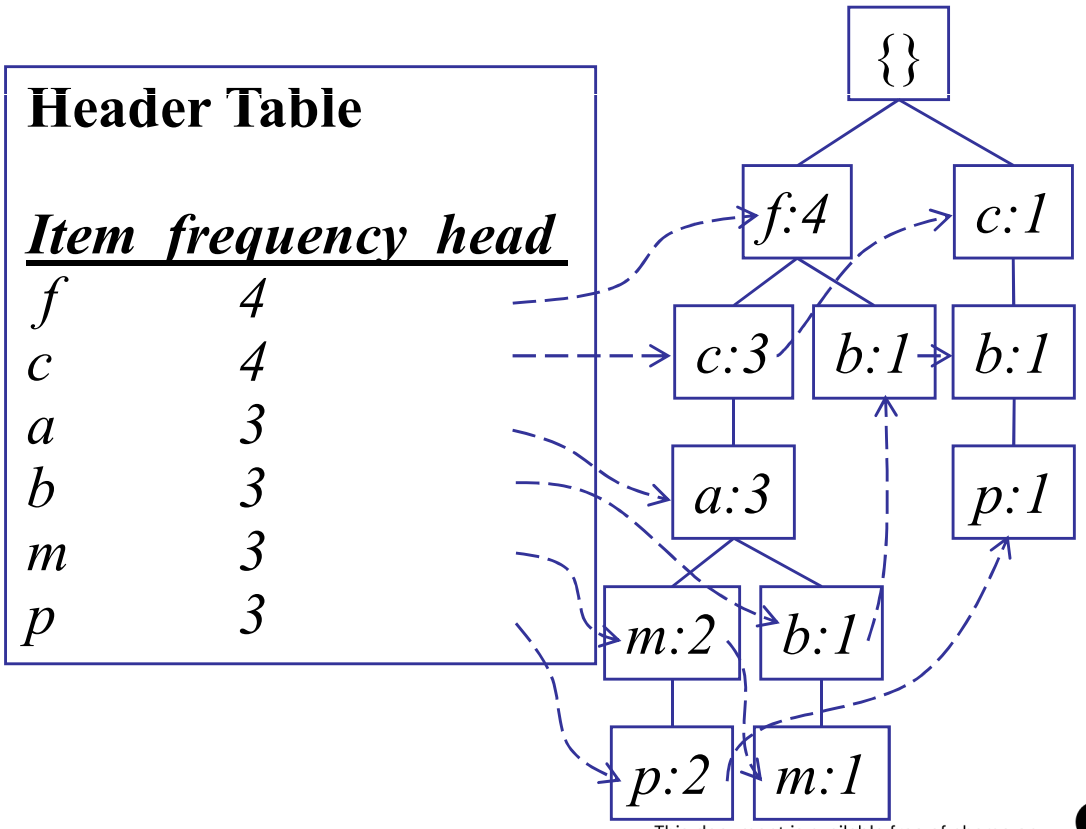
F-list = f-c-a-b-m-p

Partition Patterns and Databases

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

Find Patterns Having P From P-conditional Database

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

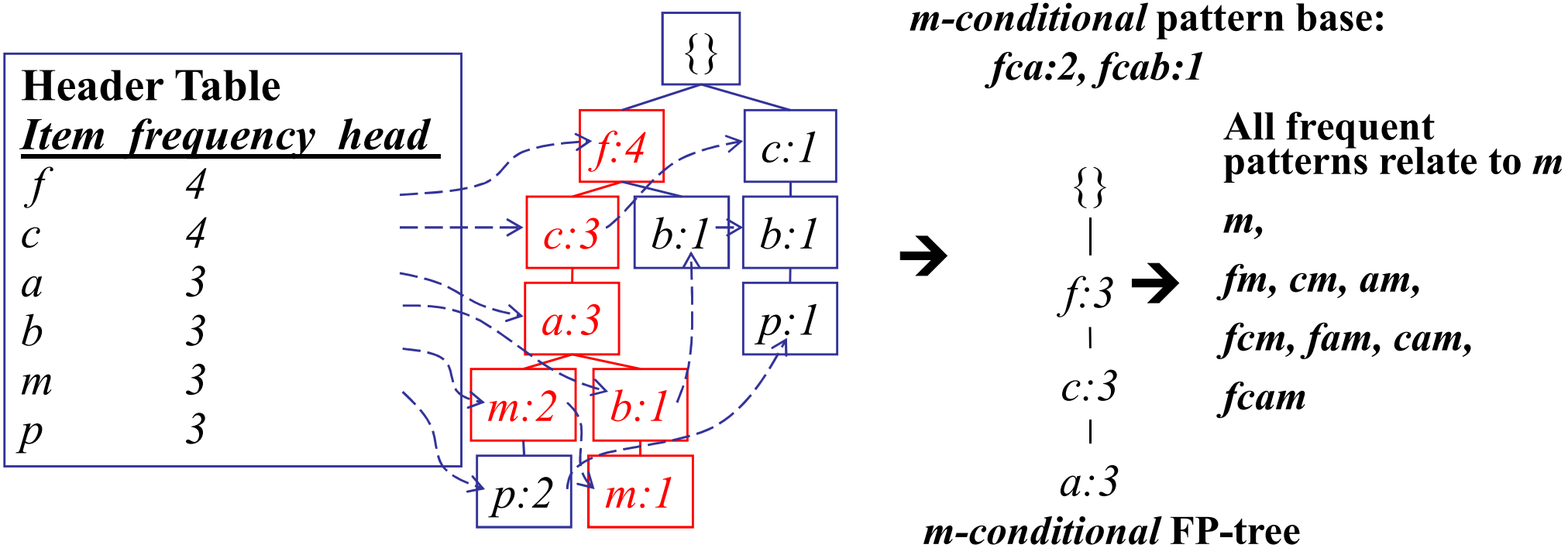


***Conditional* pattern bases**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

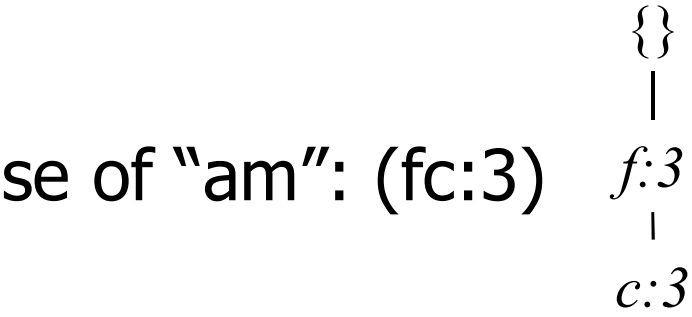
From Conditional Pattern-bases to Conditional FP-trees

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



Recursion: Mining Each Conditional FP-tree

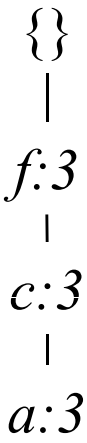
Cond. pattern base of "am": (fc:3)



***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)

***cm-conditional* FP-tree**



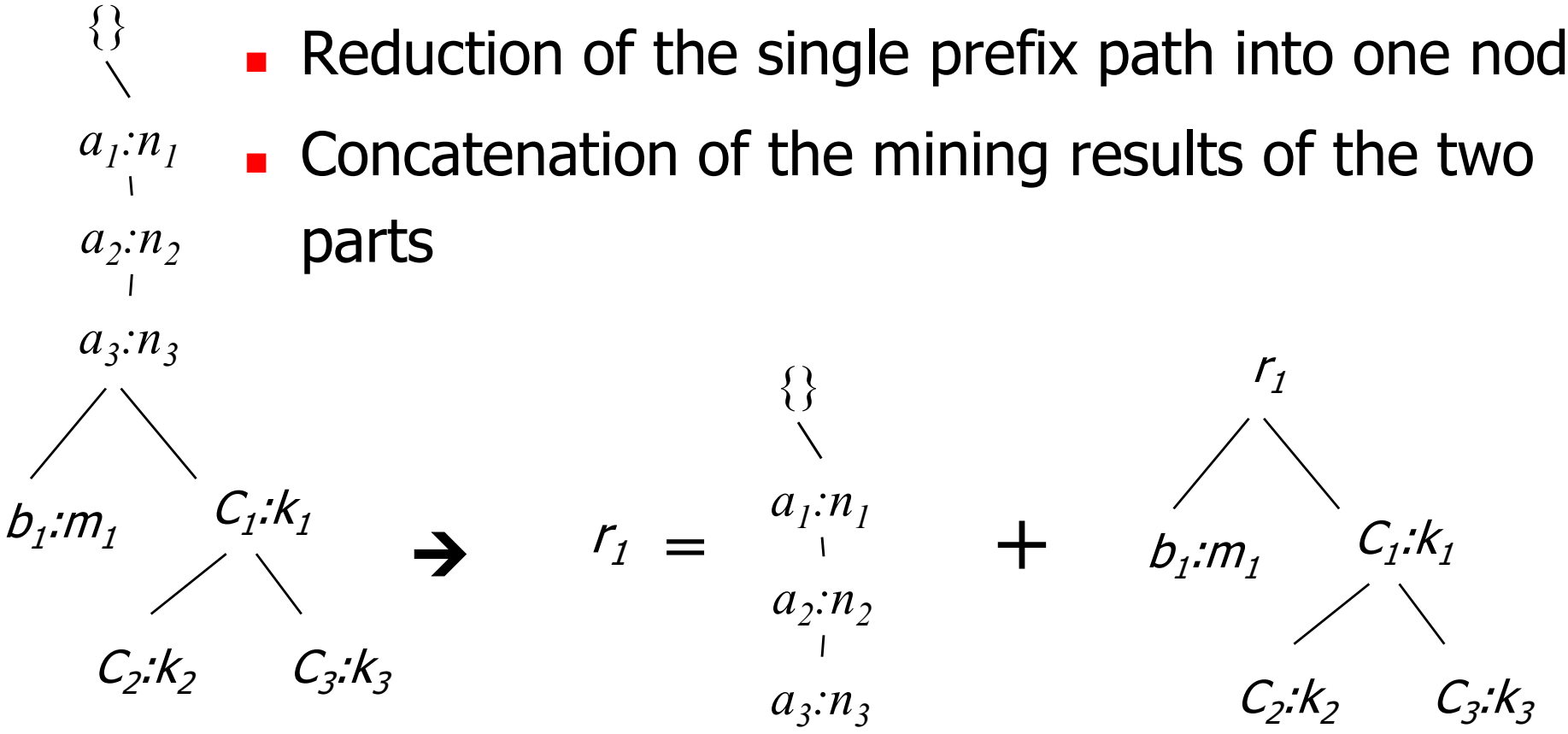
***m-conditional* FP-tree**

Cond. pattern base of "cam": (f:3)

***cam-conditional* FP-tree**

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



Benefits of the FP-tree Structure

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

The Frequent Pattern Growth Mining Method

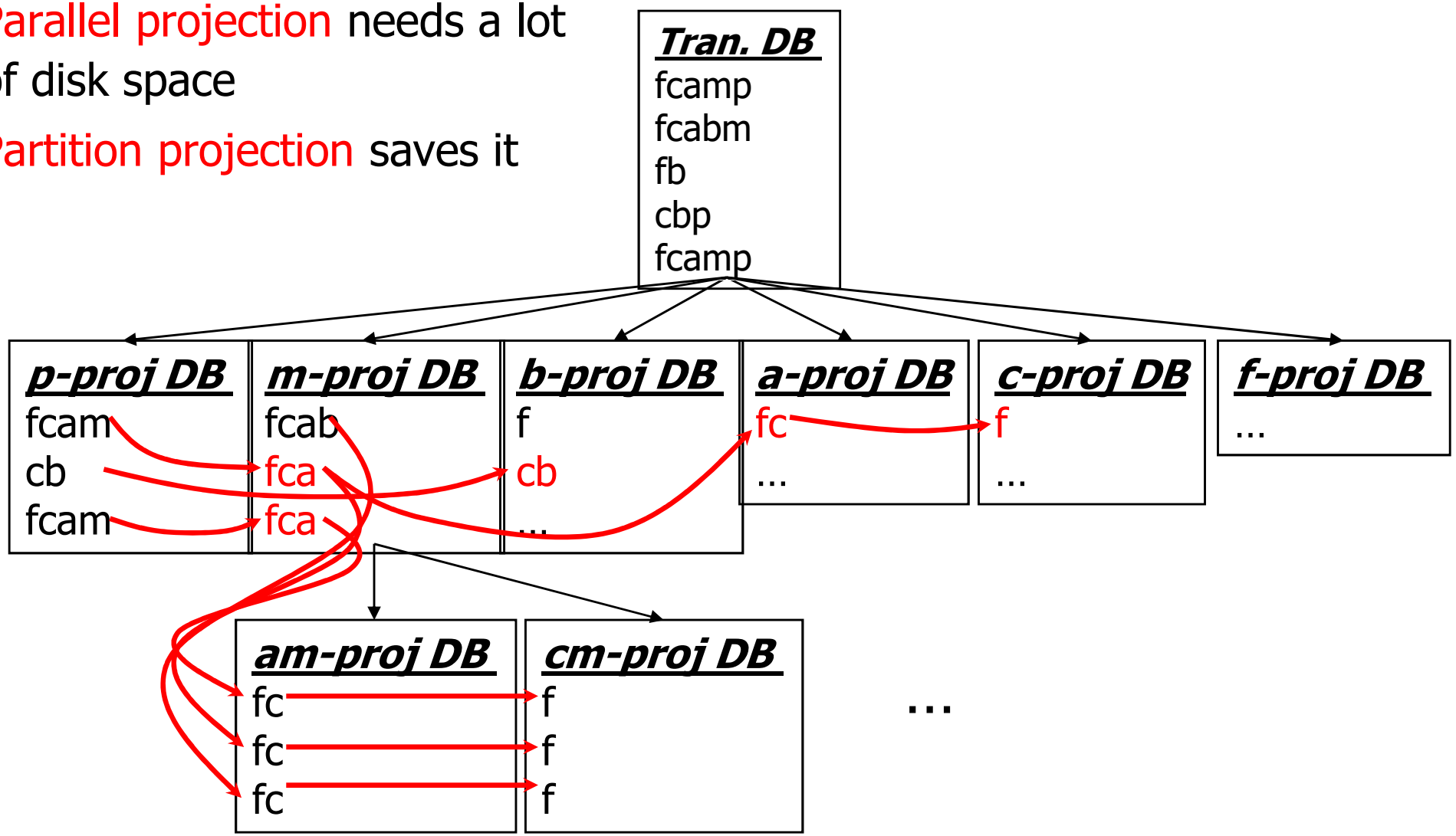
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting 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

Scaling FP-growth by Database Projection

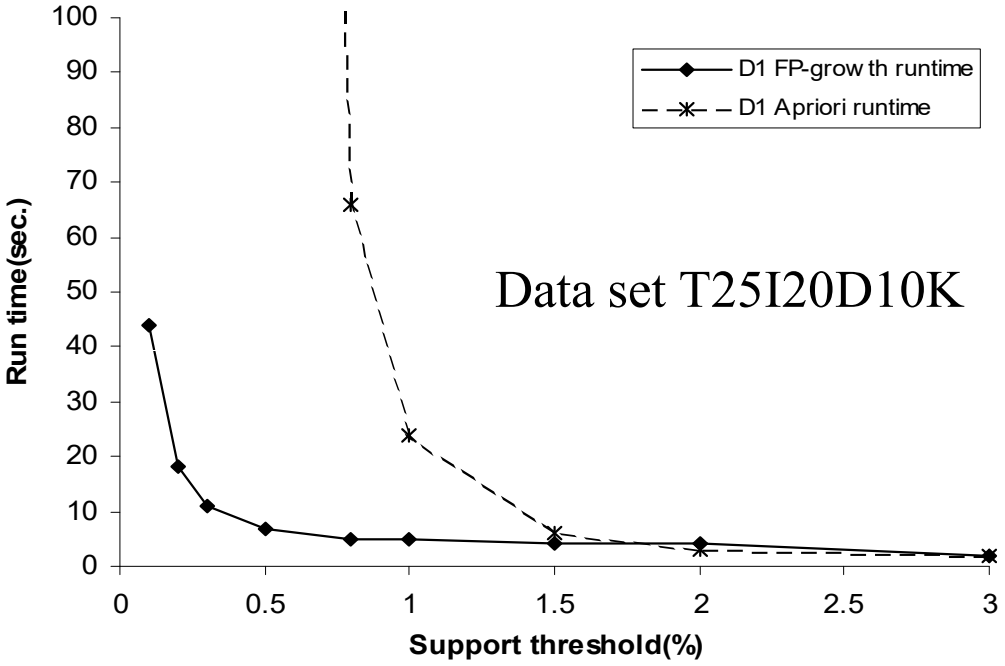
- What about if FP-tree cannot fit in memory?
  - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. partition projection techniques
  - Parallel projection
    - Project the DB in parallel for each frequent item
    - Parallel projection is space costly
    - All the partitions can be processed in parallel
  - Partition projection
    - Partition the DB based on the ordered frequent items
    - Passing the unprocessed parts to the subsequent partitions

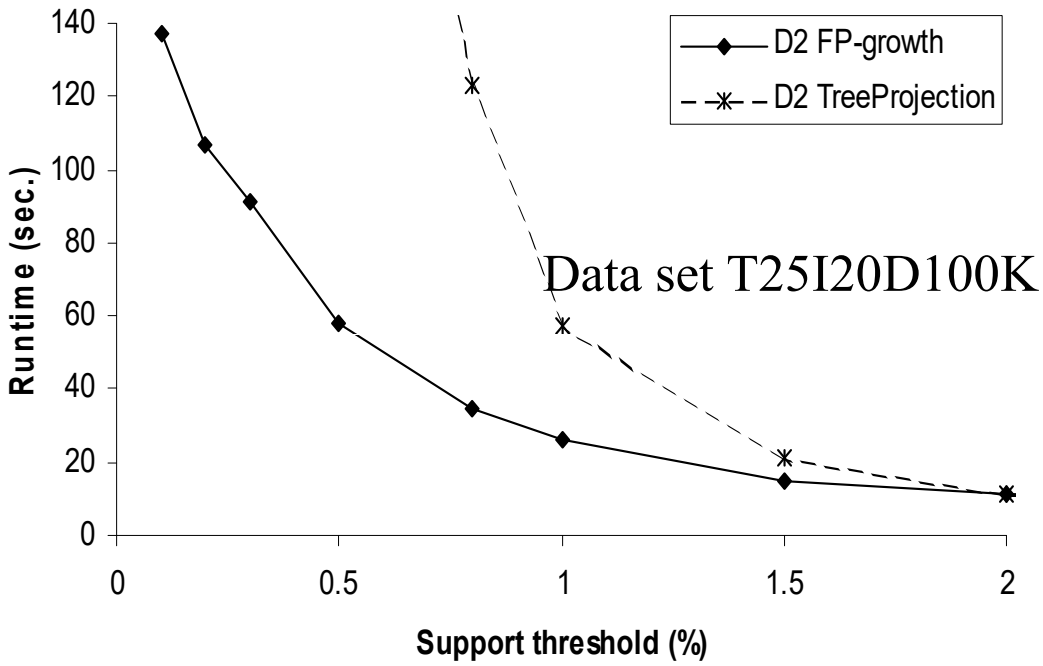
Partition-Based Projection

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



Performance of FPGrowth in Large Datasets





FP-Growth vs. Apriori

FP-Growth vs. Tree-Projection

Advantages of the Pattern Growth Approach

- 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
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching
- A good open-source implementation and refinement of FPGrowth
  - FPGrowth+ (Grahne and J. Zhu, FIMI'03)

Further Improvements of Mining Methods

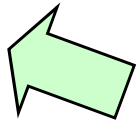
- AFOPT (Liu, et al. @ KDD'03)
  - A "push-right" method for mining condensed frequent pattern (CFP) tree
- Carpenter (Pan, et al. @ KDD'03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining
- FPgrowth+ (Grahne and Zhu, FIMI'03)
  - Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003
- TD-Close (Liu, et al, SDM'06)

Extension of Pattern Growth Mining Methodology

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD'00), FPclose, and FPMax (Grahne & Zhu, Fimi'03)
- Mining sequential patterns
  - PrefixSpan (ICDE'01), CloSpan (SDM'03), BIDE (ICDE'04)
- Mining graph patterns
  - gSpan (ICDM'02), CloseGraph (KDD'03)
- Constraint-based mining of frequent patterns
  - Convertible constraints (ICDE'01), gPrune (PAKDD'03)
- Computing iceberg data cubes with complex measures
  - H-tree, H-cubing, and Star-cubing (SIGMOD'01, VLDB'03)
- Pattern-growth-based Clustering
  - MaPle (Pei, et al., ICDM'03)
- Pattern-Growth-Based Classification
  - Mining frequent and discriminative patterns (Cheng, et al, ICDE'07)

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format



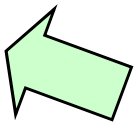
- Mining Close Frequent Patterns and Maxpatterns

ECLAT: Mining by Exploring Vertical Data Format

- Vertical format: t(AB) = {T_{11}, T_{25}, …}
  - tid-list: list of trans.-ids containing an itemset
- Deriving frequent patterns based on vertical intersections
  - t(X) = t(Y): X and Y always happen together
  - t(X) ⊂ t(Y): transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - t(X) = {T_1, T_2, T_3}, t(XY) = {T_1, T_3}
  - Diffset (XY, X) = {T_2}
- Eclat (Zaki et al. @KDD'97)
- Mining Closed patterns using vertical format: CHARM (Zaki & Hsiao@SDM'02)

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns



Mining Frequent Closed Patterns: CLOSET

- Flist: list of all frequent items in support ascending order
  - Flist: d-a-f-e-c
- Divide search space
  - Patterns having d
  - Patterns having d but no a, etc.
- Find frequent closed pattern recursively
  - Every transaction having d also has *cfa* → *cfad* is a frequent closed pattern
- J. Pei, J. Han & R. Mao. "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00.

Min_sup=2

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

CLOSET+: Mining Closed Itemsets by Pattern-Growth

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

MaxMiner: Mining Max-Patterns

- 1st scan: find frequent items
  - A, B, C, D, E
- 2nd scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE, DE

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

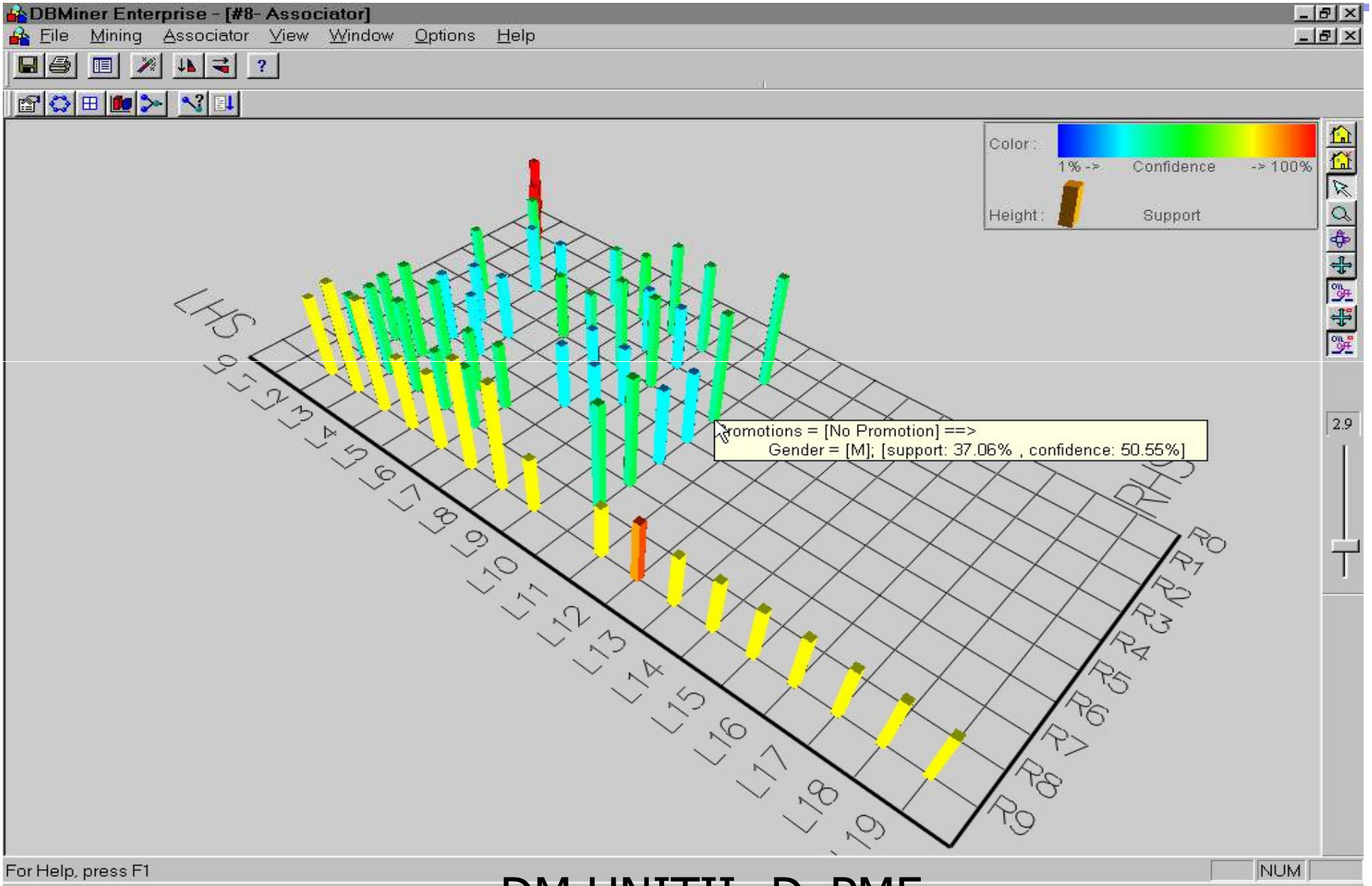
Potential max-patterns

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
- R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD'98*

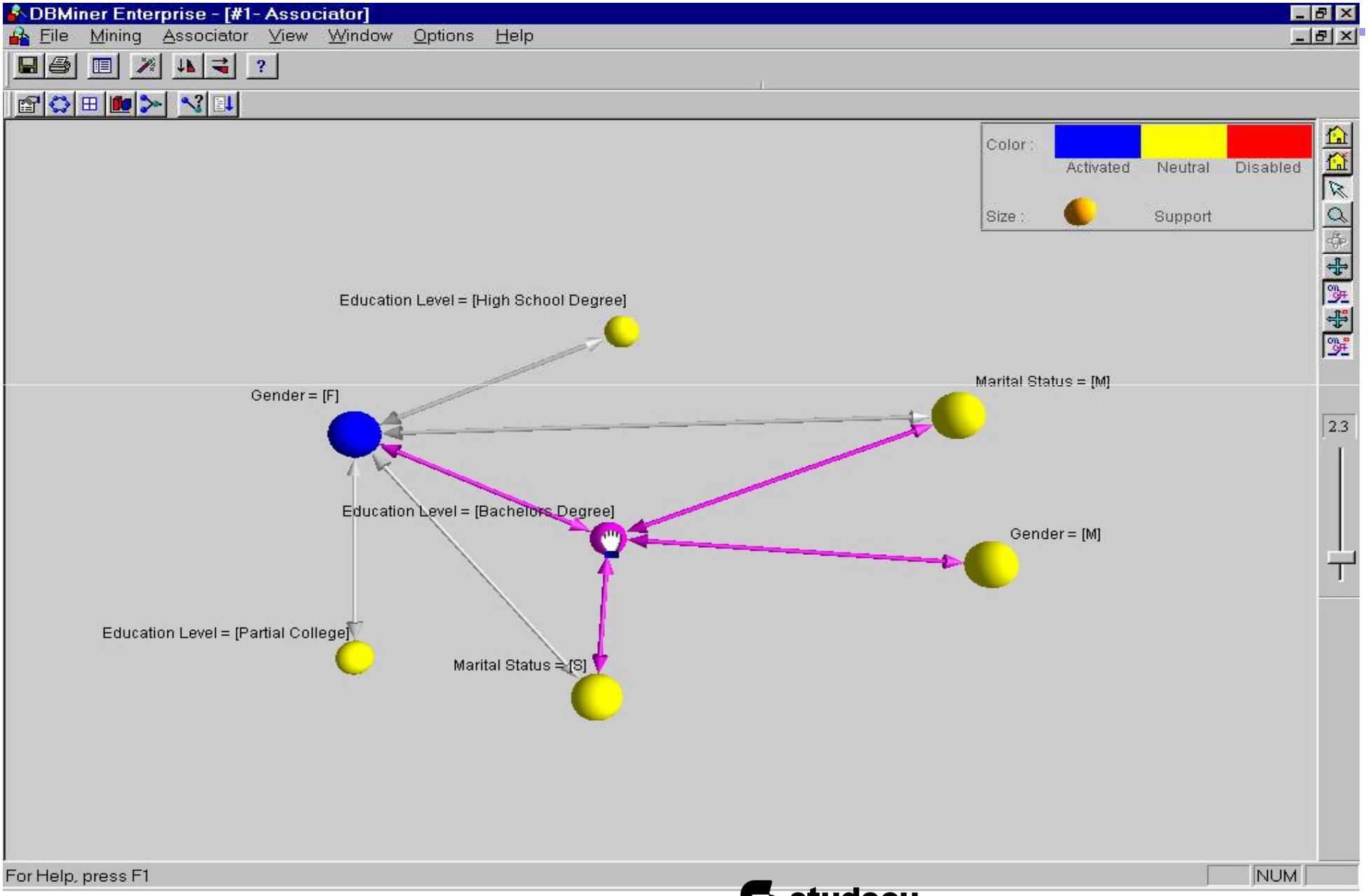
CHARM: Mining by Exploring Vertical Data Format

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

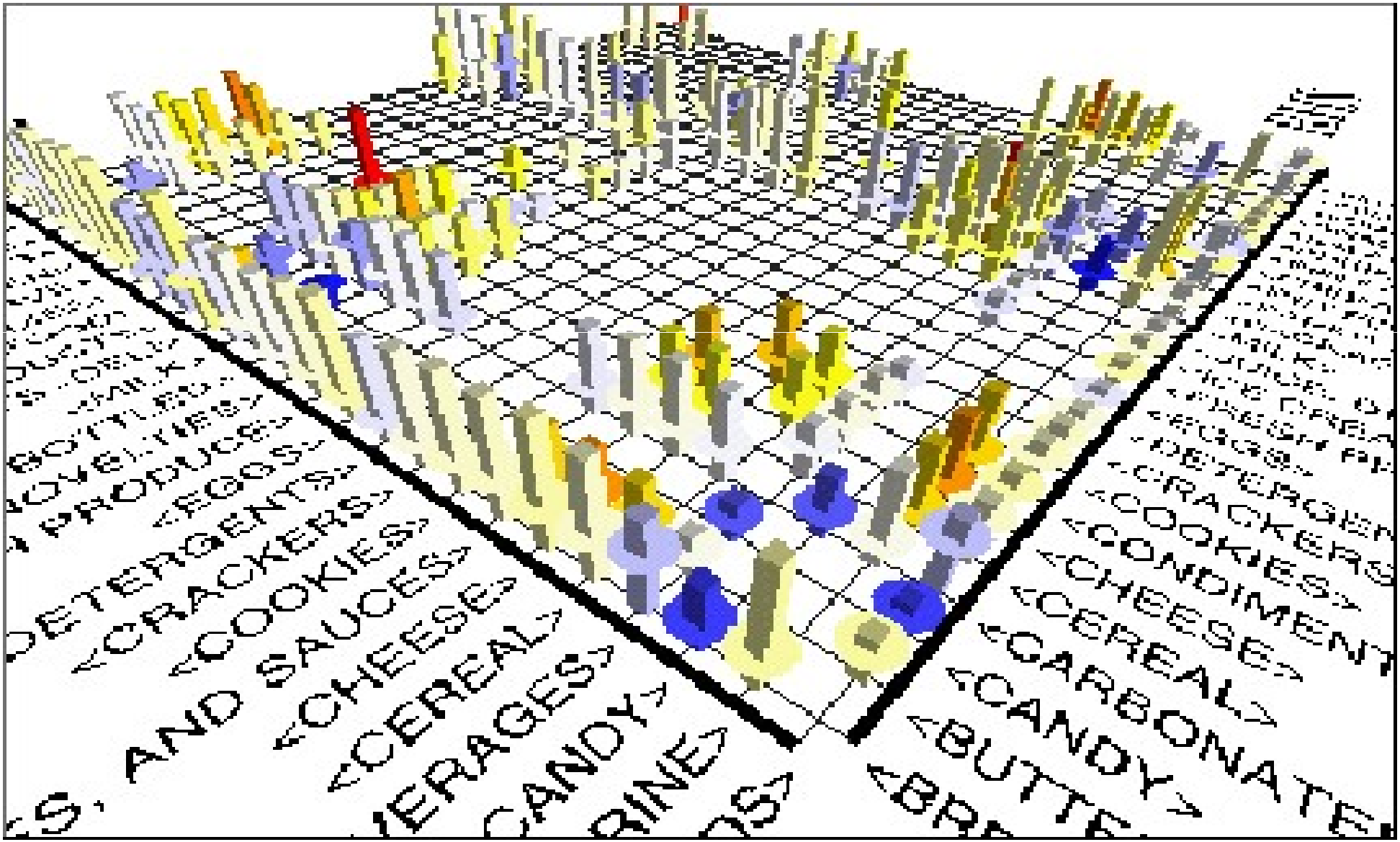
Visualization of Association Rules: Plane Graph



Visualization of Association Rules: Rule Graph

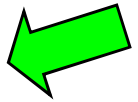


Visualization of Association Rules (SGI/MineSet 3.0)



Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary



Interestingness Measure: Correlations (Lift)

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

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

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

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

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

Are *lift* and χ^2 Good Measures of Correlation?

- "*Buy walnuts* $\Rightarrow$ *buy milk* [1%, 80%]" is misleading if 85% of customers buy milk
- Support and confidence are not good to indicate correlations
- Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD'02)
- Which are good ones?

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

Null-Invariant Measures

Table 6: Properties of interestingness measures. Note that none of the measures satisfies all the properties.

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
ϕ	ϕ-coefficient	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Goodman-Kruskal's	$0 \cdots 1$	Yes	No	No	Yes	No	No*	Yes	No
α	odds ratio	$0 \cdots 1 \cdots \infty$	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual Information	$0 \cdots 1$	Yes	Yes	Yes	No**	No	No*	Yes	No
J	J-Measure	$0 \cdots 1$	Yes	No	No	No**	No	No	No	No
G	Gini index	$0 \cdots 1$	Yes	No	No	No**	No	No*	Yes	No
s	Support	$0 \cdots 1$	No	Yes	No	Yes	No	No	No	No
c	Confidence	$0 \cdots 1$	No	Yes	No	No**	No	No	No	Yes
L	Laplace	$0 \cdots 1$	No	Yes	No	No**	No	No	No	No
V	Conviction	$0.5 \cdots 1 \cdots \infty$	No	Yes	No	No**	No	No	Yes	No
I	Interest	$0 \cdots 1 \cdots \infty$	Yes*	Yes	Yes	Yes	No	No	No	No
IS	Cosine	$0 \cdots \sqrt{P(A,B)} \cdots 1$	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	$-0.25 \cdots 0 \cdots 0.25$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	$-1 \cdots 0 \cdots 1$	Yes	Yes	Yes	No**	No	No	Yes	No
AV	Added value	$-0.5 \cdots 0 \cdots 1$	Yes	Yes	Yes	No**	No	No	No	No
S	Collective strength	$0 \cdots 1 \cdots \infty$	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	$0 \cdots 1$	No	Yes	Yes	Yes	No	No	No	Yes
K	Klosgen's	$(\frac{2}{\sqrt{3}}-1)^{1/2}[2-\sqrt{3}-\frac{1}{\sqrt{3}}] \cdots 0 \cdots \frac{2}{3\sqrt{3}}$	Yes	Yes	Yes	No**	No	No	No	No

where: P1: $O(\mathbf{M}) = 0$ if $det(\mathbf{M}) = 0$, *i.e.*, whenever A and B are statistically independent.
P2: $O(\mathbf{M_2}) > O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [k \ -k; \ -k \ k]$.
P3: $O(\mathbf{M_2}) < O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [0 \ k; \ 0 \ -k]$ or $\mathbf{M_2} = \mathbf{M_1} + [0 \ 0; \ k \ -k]$.
O1: Property 1: Symmetry under variable permutation.
O2: Property 2: Row and Column scaling invariance.
O3: Property 3: Antisymmetry under row or column permutation.
O3': Property 4: Inversion invariance.
O4: Property 5: Null invariance.
Yes*: Yes if measure is normalized.
No*: Symmetry under row or column permutation.
No**: No unless the measure is symmetrized by taking $\max(M(A,B), M(B,A))$.

Comparison of Interestingness Measures

- Null-(transaction) invariance is crucial for correlation analysis
- Lift and χ^2 are not null-invariant
- 5 null-invariant measures

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

Measure	Definition	Range	Null-Invariant
$\chi^2(a,b)$	$\sum_{i,j=0,1} \frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0,\infty]$	No
$Lift(a,b)$	$\frac{P(ab)}{P(a)P(b)}$	$[0,\infty]$	No
$AllConf(a,b)$	$\frac{sup(ab)}{max\{sup(a),sup(b)\}}$	$[0,1]$	Yes
$Coherence(a,b)$	$\frac{sup(ab)}{sup(a)+sup(b)-sup(ab)}$	$[0,1]$	Yes
$Cosine(a,b)$	$\frac{sup(ab)}{\sqrt{sup(a)sup(b)}}$	$[0,1]$	Yes
$Kulc(a,b)$	$\frac{sup(ab)}{2}(\frac{1}{sup(a)}+\frac{1}{sup(b)})$	$[0,1]$	Yes
$MaxConf$(a,b)	$max\{\frac{sup(ab)}{sup(a)},\frac{sup(ab)}{sup(b)}\}$	$[0,1]$	Yes

Table 3. Interestingness measure definitions.

Null-transactions w.r.t. m and ~m

Kulczynski measure (1927)

Null-invariant

Data set	mc	$\overline{mc}$	$m\overline{c}$	$\overline{mc}$	χ^2	$Lift$	$AllConf$	$Coherence$	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	90557	9.26	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0	1	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	670	8.44	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	24740	25.75	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	8173	9.18	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	965	1.97	0.01	0.01	0.10	0.5	0.99

Table 2. Example data sets.

Subtle: They disagree

Analysis of DBLP Coauthor Relationships

Recent DB conferences, removing balanced associations, low sup, etc.

ID	Author a	Author b	$sup(ab)$	$sup(a)$	$sup(b)$	$Coherence$	$Cosine$	$Kulc$
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Table 5. Experiment on DBLP data set.

Advisor-advisee relation: Kulc: high, coherence: low, cosine: middle

- Tianyi Wu, Yuguo Chen and Jiawei Han, "Association Mining in Large Databases: A Re-Examination of Its Measures", Proc. 2007 Int. Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD'07), Sept. 2007

Which Null-Invariant Measure Is Better?

- IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications

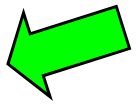
$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

- Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is balanced & neutral
  - D_5 is imbalanced & neutral
  - D_6 is very imbalanced & neutral

Data	*mc*	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	*all_conf.*	*max_conf.*	*Kulc.*	*cosine*	IR
D_1	10,000	1,000	1,000	100,000	0.91	0.91	0.91	0.91	0.0
D_2	10,000	1,000	1,000	100	0.91	0.91	0.91	0.91	0.0
D_3	100	1,000	1,000	100,000	0.09	0.09	0.09	0.09	0.0
D_4	1,000	1,000	1,000	100,000	0.5	0.5	0.5	0.5	0.0
D_5	1,000	100	10,000	100,000	0.09	0.91	0.5	0.29	0.89
D_6	1,000	10	100,000	100,000	0.01	0.99	0.5	0.10	0.99

Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary



Summary

- Basic concepts: association rules, support-confident framework, closed and max-patterns
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (ECLAT, CHARM, ...)
- Which patterns are interesting?
  - Pattern evaluation methods

Ref: Basic Concepts of Frequent Pattern Mining

- (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93
- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98
- (Closed-pattern) N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

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, 2002.
- G. Grahne and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. FIMI'03
- B. Goethals and M. Zaki. An introduction to workshop on frequent itemset mining implementations. *Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03),* Melbourne, FL, Nov. 2003
- 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

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.
- M. J. Zaki and C. J. 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

- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97.
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94.
- R. J. Hilderman and H. J. Hamilton. *Knowledge Discovery and Measures of Interest*. Kluwer Academic, 2001.
- 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, "Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework", Data Mining and Knowledge Discovery, 21(3):371-397, 2010

UNIT - II

Mining frequent patterns: Basic concepts-Market Basket Analysis-Frequent itemsets, Closed itemsets-Association rules-Introductio-Apriori algorithm-theoritical approach-Apply Apriori algorithm on dataset-1-Apply Apriori algorithm on dataset-2-Generating Association rules from frequent item sets -Improving efficiency of Apriori-Pattern growth approach-Mining frequest itemsets using Vertical data format-Strong rules vs. weak rules-Association analysis to Correlation analysis-Comparison of pattern evaluation measures

1.Mining Frequent Patterns

- Frequent patterns are patterns (e.g., item sets, subsequences, or substructures) that appear frequently in a data set.
- a set of items, such as milk and bread, that appear frequently together in a transaction data set is a frequent item set.
- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern.
- A substructure can refer to different structural forms, such as subgraphs, subtrees, which may be combined with item sets or subsequences.
- If a substructure occurs frequently, it is called a (frequent) structured pattern.

1.Mining Frequent Patterns

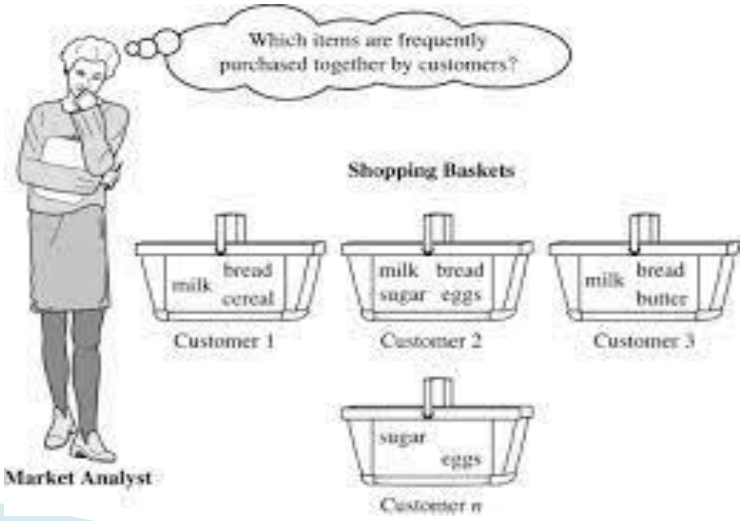
- It helps in data classification, clustering, and other data mining task.
- Frequent pattern mining searches for recurring relationships in a given data set.

Market Basket Analysis: A Motivating Example

- Frequent item set mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- It helps in many business decision-making processes such as catalog design, cross-marketing, and customer shopping behavior analysis.

1.Mining Frequent Patterns: market basket analysis

- analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets".
- discovery of these associations can help retailers develop marketing strategies.



1.Mining Frequent Patterns: market basket analysis

- learn more about the buying habits of customers market basket analysis may be performed on the retail data of customer transactions at store.
- use the results to plan marketing or advertising strategies, or in the design of a new catalog.
- help to design different store layouts.

one strategy

- items that are frequently purchased together can be placed in closeness to further encourage the combined sale of such items.

Ex:

Computer hardware and software

1.Mining Frequent Patterns: market basket analysis

Alternative strategy

- placing hardware and software at opposite ends of the store may tempt customers who purchase such items to pick up other items along the way.

ex:

Expensive computer and antivirus.

- Market basket analysis can also help retailers plan which items to put on sale at reduced prices.

Ex:

Computer and printer

1.Mining Frequent Patterns: market basket analysis

- each item has a Boolean variable, representing the presence or absence of that item.
- Each basket can then be represented by a Boolean vector of values assigned to these variables.
- The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently purchased together.
- These patterns can be represented in the form of association rules.
- computer ⇒ antivirus software [support = 2%,confidence = 60%].

1.Mining Frequent Patterns: market basket analysis

- They respectively reflect the usefulness and certainty of discovered rules.
- A support of 2% for Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.
- association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.
- These thresholds can be a set by users or domain experts.
- Additional analysis can be performed to discover interesting statistical correlations between associated items.

2.Frequent Item sets, Closed Item sets, and Association Rules

- A set of items is referred to as an **item set.**
- An item set that contains *k* items is a ***k*-item set**.
- The set {*computer, antivirus software}* is a **2-itemset**.
- The occurrence frequency of an item set is the number of transactions that contain the item set.
- This is also known, simply, as the **frequency, support count,** or **count** of the item set.

$$\textbf{support}(A \Rightarrow B) = P(A \cup B)$$

$$\textbf{confidence}(A \Rightarrow B) = P(B|A)$$

2.Frequent Item sets, Closed Item sets, and Association Rules

- **Support :** represents the popularity of that product of all the product transactions.
- **Support is an indication of how frequently the items appear in the data.**
- **5% Support** means total 5% of transactions in database.
  - Support(A -> B) = Support count(A ∪ B)
- **Confidence:** indicates the number of times the if–then statements are found true.
- **Measures how often items in Y appear in transactions that contain X**
- A confidence of 60% means that 60% of the customers who purchased a milk and bread also bought butter.
- Confidence(A -> B) = Support count(A ∪ B) / Support count(A)
- If a rule satisfies both minimum support and minimum confidence,

it is a strong rule.

Support_count(X) : Number of transactions in which X appears.

If X is A **union** B then it is the number of transactions in which A and B both are present.

2.Frequent Item sets, Closed Item sets, and Association Rules

- Association rule mining finds interesting associations and relationships among large sets of data items.
- This rule shows how frequently a item set occurs in a transaction.
- Ex: Market Based Analysis.
- An implication expression of the form X -> Y, where X and Y are any 2 item sets.

{milk,bread}-- > egg.

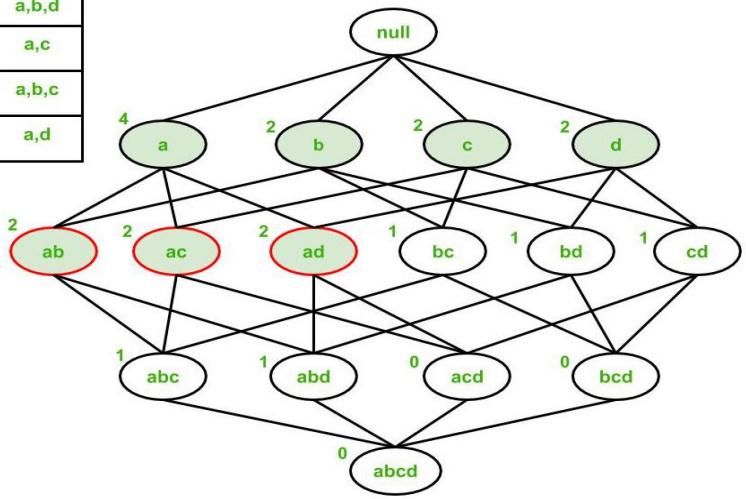
- association rule mining can be viewed as a **two-step process**:
- 1. Find all frequent item sets: each of these item sets will occur at least as frequently as a predetermined minimum support count, *min_sup*.
- 2. Generate strong association rules from the frequent item sets: these rules must satisfy minimum support and minimum confidence.

2.Frequent Item sets, Closed Item sets, and Association Rules

- **Maximal Item set:** An item set is maximal frequent if none of its supersets are frequent.
- Steps:
- Examine the frequent item sets that appear at the border between the infrequent and frequent item sets.
- Identify all of its immediate supersets.
- If none of the immediate supersets are frequent, the item set is maximal frequent.

TID	Items
1	a,b,d
2	a,c
3	a,b,c
4	a,d

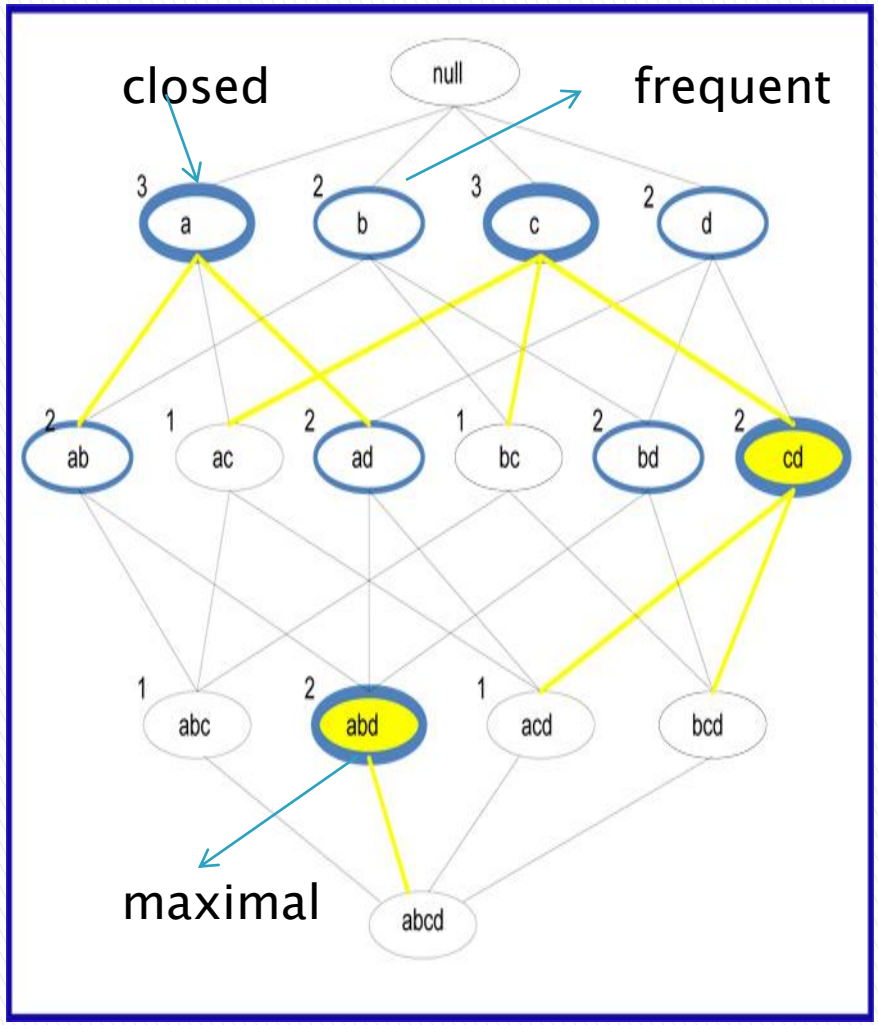
- Sub count=2.



2.Frequent Item sets, Closed Item sets, and Association Rules

- **Closed Item set:** An item set is closed in a data set if there exists no superset that has the same support count as this original item set.
- Sub count=2

a(3)={ab(2),ac(1),ad(2)}// closed



2. Frequent Item sets, Closed Item sets, and Association Rules

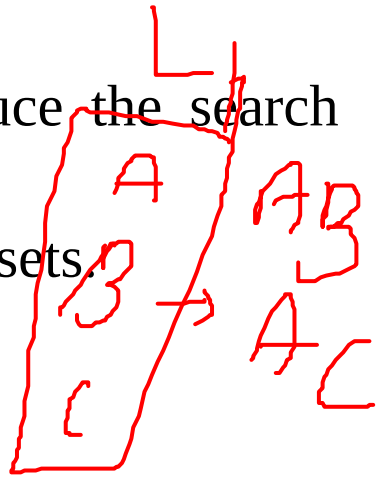
- **K- Item set:** Item set which contains K items is a K-item set.
- So it can be said that an item set is frequent if the corresponding support count is greater than minimum support count.

2.Frequent Item sets, Closed Item sets, and Association Rules

- A major challenge in mining frequent item sets from a large data set is the fact that such mining often generates a huge number of item sets satisfying the minimum support (min sup) threshold, especially when min sup is set low.
- This is because if an item set is frequent, each of its subsets is frequent as well.
- For example, a frequent item set of length 100, such as {a1, a2,..., a100}, contains $\binom{100}{1} = 100$

3. Apriori Algorithm

- Apriori algorithm was the first algorithm that was proposed for frequent item set mining.
- In 1994 it was improved by R Agarwal and R Srikant and came to be known as Apriori.
- This algorithm uses two steps "join" and "prune" to reduce the search space.
- It is an iterative approach to discover the most frequent item sets.
- The probability that item I is not frequent is if:
- P(I) < minimum support threshold, then I is not frequent.
- P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to item set.
- If an item set set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored.



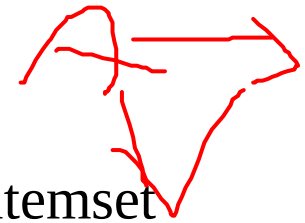
3.Apriori Algorithm

- **Join Step**: This step generates (K+1) item set from K-item sets by joining each item with itself.
- **Prune Step**: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate item sets
- This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved.
- A minimum support threshold is given in the problem or it is assumed by the user.
- **#1)** In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.
- **#2)** Let there be some minimum support, min_sup (eg 2). The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

3. Apriori Algorithm

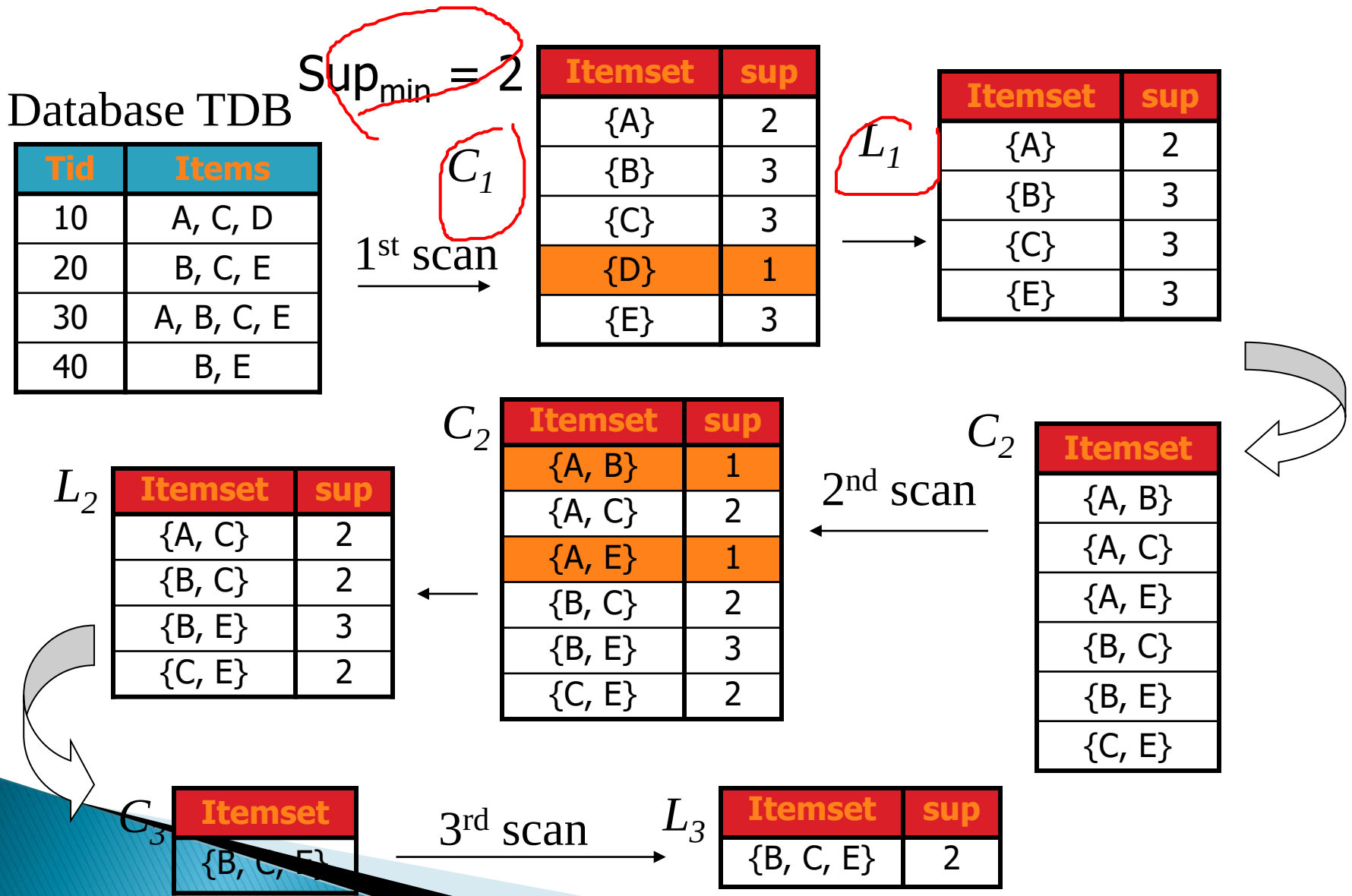
- **#3)** Next, 2-itemset frequent items with min_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.
- **#4)** The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –itemsets with min-sup only.
- **#5)** The next iteration will form 3 –itemsets using join and prune step. This iteration will follow antimonotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min_sup.
- If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.
- **#6)** Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.

3. Apriori Method



- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate item sets from length k frequent item sets
  - Terminate when no frequent or candidate set can be generated
- To improve the efficiency of the level-wise generation of frequent item sets, the apriori property is used to reduce the search space
  - All non-empty subsets of a frequent itemset must also be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
- large itemset: itemset with support > s
- candidate itemset: itemset that may have support > s

3. The Apriori Algorithm—An Example



3. Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- 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*}

3. Algorithm: Apriori

Find frequent item sets using an iterative level-wise approach based on candidate generation.

Input: D, a database of transactions; min sup, the minimum support count threshold.

Output: L, frequent item sets in D.

Method:

(1) L_1 = find frequent 1-itemsets(D);

(2) for ($k = 2; L_{k-1} \neq \varphi; k++$) {

(3) C_k = apriori gen(L_{k-1});

(4) for each transaction $t \in D$ { // scan D for counts

(5) C_t = subset(C_k , t); // get the subsets of t that are candidates

(6) for each candidate $c \in C_t$

(7) c.count++;

(8) }

(9) $L_k = \{c \in C_k \mid c.count \geq min\ sup\}$

(10) }

(11) return $L = \cup_k L_k$;

procedure apriori gen(L_k-1:frequent (k − 1)-itemsets)

(1) for each itemset $l_1 \in L_{k-1}$

(2) for each itemset $l_2 \in L_{k-1}$

(3) if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {

(4) $c = l_1 \bowtie l_2$; // join step: generate candidates

(5) if has infrequent subset(c, L_{k-1}) then

(6) delete c; // prune step: remove unfruitful candidate

(7) else add c to C_k ;

(8) }

(9) return C_k ;

4.Generating Association Rules from Frequent Itemsets

- Once the frequent item sets from transactions in a database D have been found, can generate strong association rules from them.
- where strong association rules satisfy both minimum support and minimum confidence.
- This can be done using for confidence.
- confidence($A \Rightarrow B$) = $P(B|A)$ = support count($A \cup B$)/ support count(A) .
- The conditional probability is expressed in terms of itemset support count, where support count($A \cup B$) is the number of transactions containing the itemsets $A \cup B$, and support count(A) is the number of transactions containing the itemset A.

4.Generating Association Rules from Frequent Itemsets

- Based on this equation, association rules can be generated as follows:
- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "s ⇒ (l − s)"

  if (support count(l) / support count(s)) ≥ min conf, where min conf is the minimum confidence threshold.
- Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support.
- Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.
- Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.

4.Generating Association Rules Example

- Example: Given the following table and the frequent item set X = {I1, I2, I5}, generate the association rules.

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

- The nonempty subsets of X are:

  {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, {I5}

- The resulting association rules are:

$$
\begin{aligned}
\{I1, I2\} &\Rightarrow I5, && confidence = 2/4 = 50\% \\
\{I1, I5\} &\Rightarrow I2, && confidence = 2/2 = 100\% \\
\{I2, I5\} &\Rightarrow I1, && confidence = 2/2 = 100\% \\
I1 &\Rightarrow \{I2, I5\}, && confidence = 2/6 = 33\% \\
I2 &\Rightarrow \{I1, I5\}, && confidence = 2/7 = 29\% \\
I5 &\Rightarrow \{I1, I2\}, && confidence = 2/2 = 100\%
\end{aligned}
$$

- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules are output, because these are the only ones generated that are strong

5.Improving efficiency of apriori algorithm

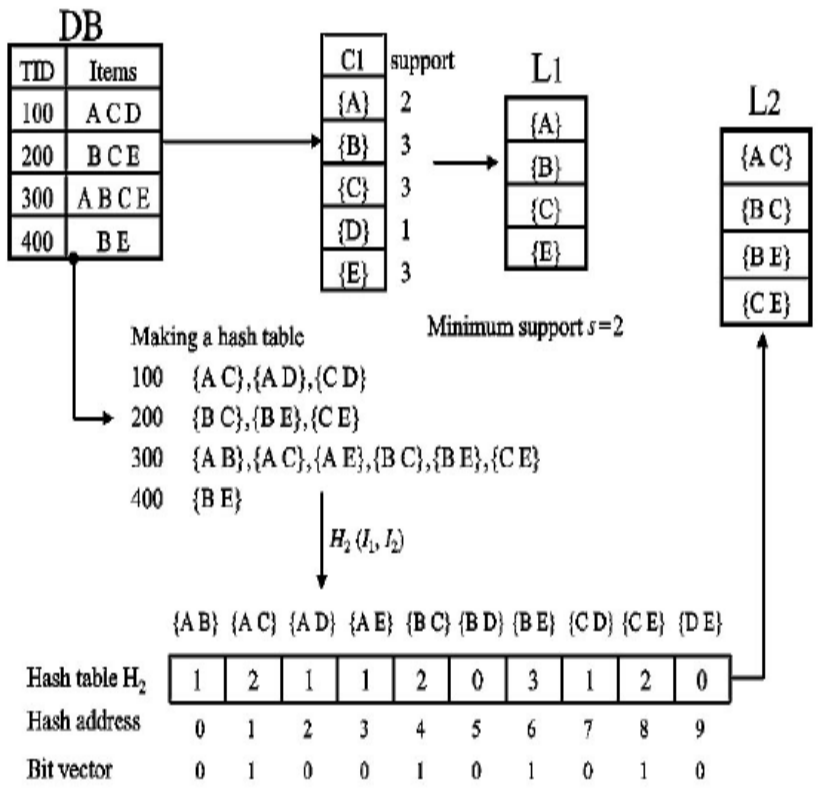
(i)Hash-based technique (hashing itemsets into corresponding buckets):

A hash-based technique can be used to reduce the size of the candidate k-itemsets, C_k , for k > 1.

For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L1, we can generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different buckets of a hash table structure, and increase the corresponding bucket counts .

A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set.

Such a hash-based technique may substantially reduce the number of candidate k-itemsets examined (especially when k = 2).



5. Improving efficiency of apriori algorithm

(ii)Transaction reduction (reducing the number of transactions scanned in future iterations):

A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k + 1)-itemsets.

Therefore, such a transaction can be marked or removed from further consideration because subsequent database scans for j-itemsets, where j > k, will not need to consider such a transaction.

Improving efficiency of apriori algorithm

(iii) Partitioning (partitioning the data to find candidate item sets):

A partitioning technique can be used that requires just two database scans to mine the frequent item sets.

It consists of two phases.

In phase I, the algorithm divides the transactions of D into n non overlapping partitions.

If the minimum relative support threshold for transactions in D is min sup, then the minimum support count for a partition is min sup × the number of transactions in that partition.

For each partition, all the local frequent itemsets (i.e., the itemsets frequent within the partition) are found.

A local frequent itemset may or may not be frequent with respect to the entire database, D.

Improving efficiency of apriori algorithm

Partitioning (partitioning the data to find candidate item sets):

However, any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions.

Therefore, all local frequent itemsets are candidate itemsets with respect to D. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to D.

In phase II, a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets.

Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

Improving efficiency of apriori algorithm

(iv) Sampling (mining on a subset of the given data):

- The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D.
- we trade off some degree of accuracy against efficiency.
- The S sample size is such that the search for frequent item sets in S can be done in main memory, and so only one scan of the transactions in S is required overall.
- Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets.
- To reduce this possibility, use a lower support threshold than minimum support to find the frequent itemsets local to S (denoted L S).
- The rest of the database is then used to compute the actual frequencies of each itemset in L S

Improving efficiency of apriori algorithm

Sampling (mining on a subset of the given data):

- A mechanism is used to determine whether all the global frequent itemsets are included in L S .
- If L S actually contains all the frequent itemsets in D, then only one scan of D is required.
- Otherwise, a second pass can be done to find the frequent itemsets that were missed in the first pass.
- The sampling approach is especially beneficial when efficiency is of utmost importance such as in computationally intensive applications that must be run frequently

Improving efficiency of apriori algorithm

(v) Dynamic itemset counting (adding candidate itemsets at different points during a scan):

- A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points.
- In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan.
- The technique uses the count-so-far as the lower bound of the actual count.
- If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.

Mining by Exploring Vertical Data Format

Minimum support = 2

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

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}

itemset	TID_set
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, 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}

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

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%]
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%]
- Measure of dependent/correlated events: lift

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

- If lift $=1$, occurrence of item set A is independent of occurrence of item set B, otherwise, they are dependent and correlated

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

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

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

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%]
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%]
- Measure of dependent/correlated events: lift

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

- If lift $=1$, occurrence of itemset A is independent of occurrence of itemset B, otherwise, they are dependent and correlated

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

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

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

A Comparison of Pattern Evaluation Measures

- Researchers have studied many pattern evaluation measures even before the start of in-depth research on scalable methods for mining frequent patterns. Recently, several other pattern evaluation measures have attracted interest.
- All_conf(A,B) = sup($A \cup B$)/ max{sup(A),sup(B)}= min{P(A|B),P(B|A)},
- Given two itemsets, A and B, the max confidence measure of A and B is defined as max conf(A, B) = max{P(A|B),P(B|A)}.
- The max conf measure is the maximum confidence of the two association rules, "A ⇒ B" and "B ⇒ A."
- Given two itemsets, A and B, the Kulczynski measure of A and B (abbreviated as Kulc) is defined as Kulc(A, B) = 1/ 2 (P(A|B) + P(B|A)).

Data Mining:
Concepts and Techniques

UNIT –2

Jiawei Han
Department of Computer Science

Mining Frequent Patterns, Association and Correlations

- *Mining frequent patterns: Basic concepts*
- *Market Basket Analysis Frequent itemsets, Closed itemsets*
- *Association rules-Introduction Apriori algorithm-theoritical approach*
- *Apply Apriori algorithm on dataset-1*
- *Apply Apriori algorithm on dataset-2*
- *Generating Association rules from frequent itemsets*
- *Improving efficiency of Apriori*
- *Pattern growth approach*
- *Mining frequent item sets using Vertical data format*
- *Strong rules vs. weak rules Association analysis to Correlation Analysis*
- *Comparison of pattern evaluation Measures*

What Is Frequent Pattern Analysis?

- Frequent pattern: **Frequent** patterns are itemsets, subsequences, or substructures that appear in a **data** set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread, that appear frequently together in a transaction **data** set, is a **frequent** itemset.
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

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

IMPORTANT TERMS

- Itemset
- Frequent pattern
- Association rule
- Support- how frequently the items appear in the **data**
- Confidence- the number of times the if-then statements are found true.

ASSOCIATION RULES

- if-then statements that help to show the probability of relationships between data items.
- An association rule has two parts: an antecedent (if) and a consequent (then).
- An antecedent is an item found within the data.
- A consequent is an item found in combination with the antecedent.

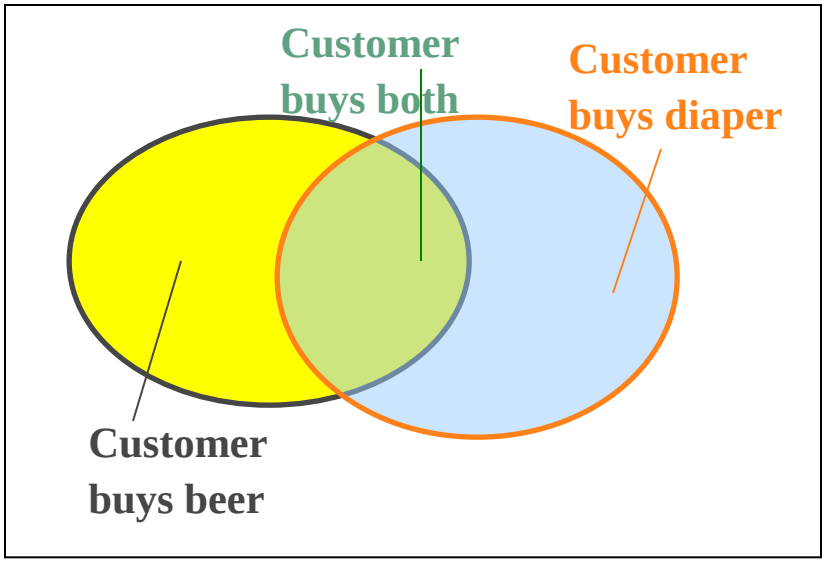
- Association rules criteria

  support **and** ***confidence*** to identify the most important relationships.

- Support is an indication of how frequently the items appear in the data.

- Confidence indicates the number of times the if-then statements are found true.

Transaction-id	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee,Diaper, Eggs, Milk



Basic Concepts: Frequent Patterns and Association Rules

Basic Concepts: Frequent Itemsets (Patterns)

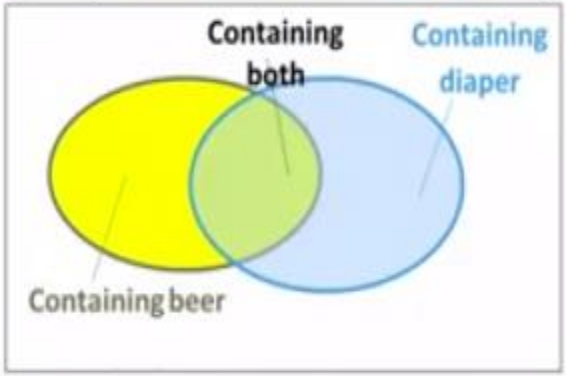
- Itemset: A set of one or more items
- k-itemset: $X = \{x_1, \ldots, x_k\}$
- *(absolute) support (count)* of X: Frequency or the number of occurrences of an itemset X
- *(relative) support, s:* The fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if the support of X is no less than a *minsup* threshold (denoted as σ)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Let *minsup = 50%*
- Freq. 1-itemsets:
  - Beer: 3 (60%); Nuts: 3 (60%)
  - Diaper: 4 (80%); Eggs: 3 (60%)
- Freq. 2-itemsets:
  - {Beer, Diaper}: 3 (60%)

From Frequent Itemsets to Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Note: Itemset: X ∪ Y, a subtle notation!

- Association rules: $X \rightarrow Y$ (s, c)
  - Support, *s*: The probability that a transaction contains X ∪ Y
  - Confidence, *c*: *The* conditional probability that a transaction containing X also contains *Y*
  - c = sup(X ∪ Y) / sup(X)
- **Association rule mining**: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence
- Frequent itemsets: Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3
- Association rules: Let *minconf = 50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

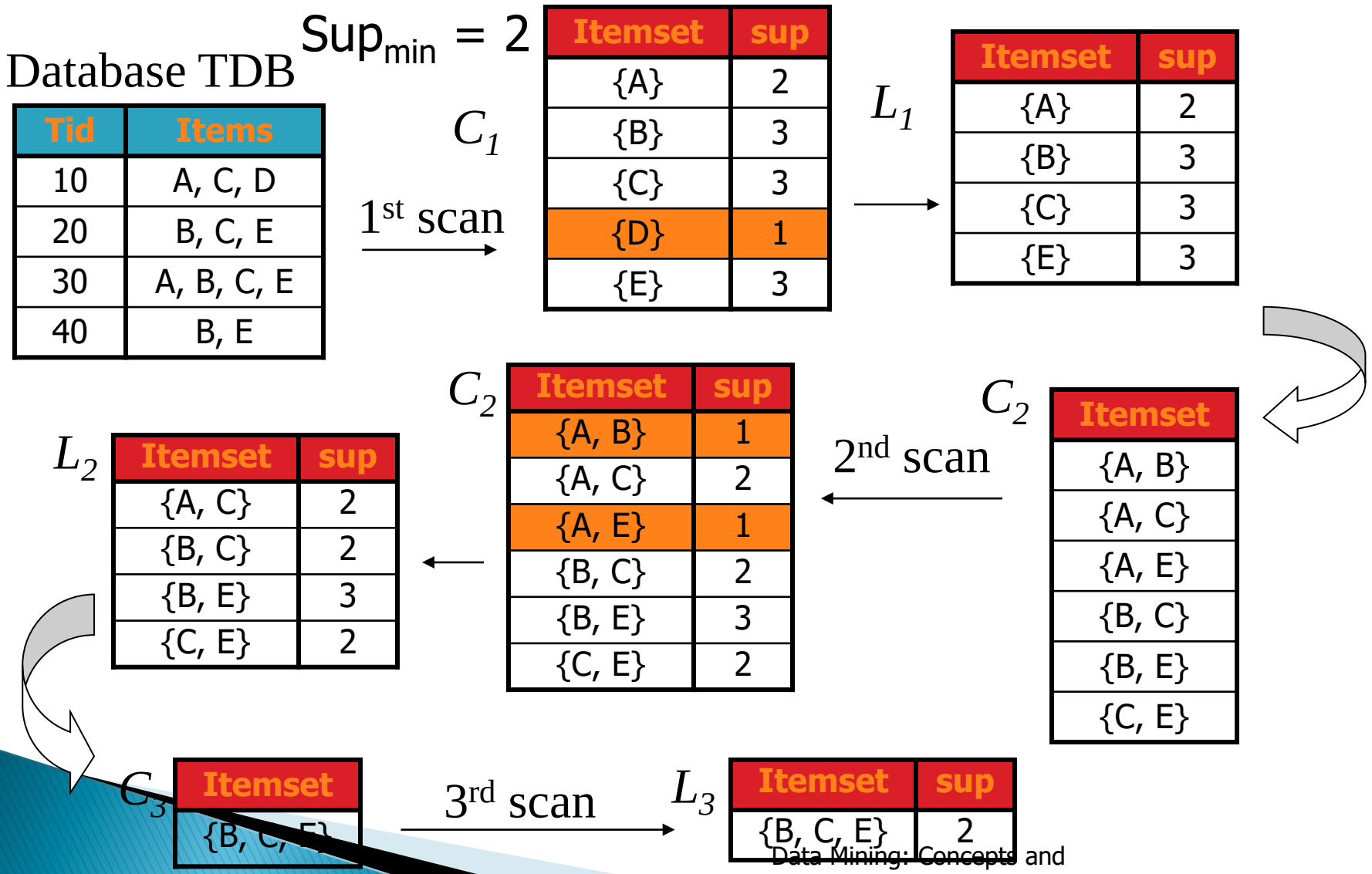
Scalable Methods for Mining Frequent Patterns

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

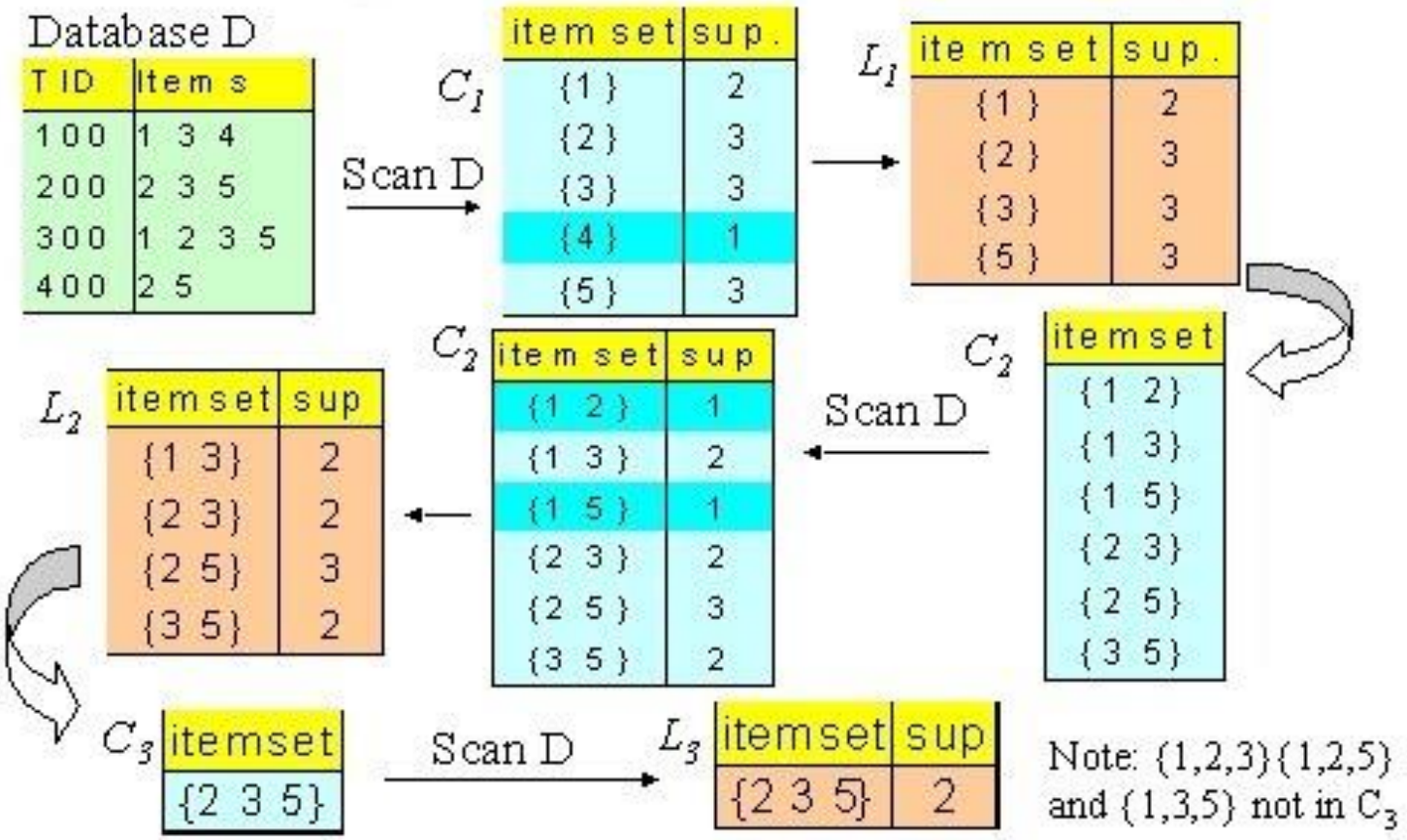
Apriori: A Candidate Generation-and-Test Approach

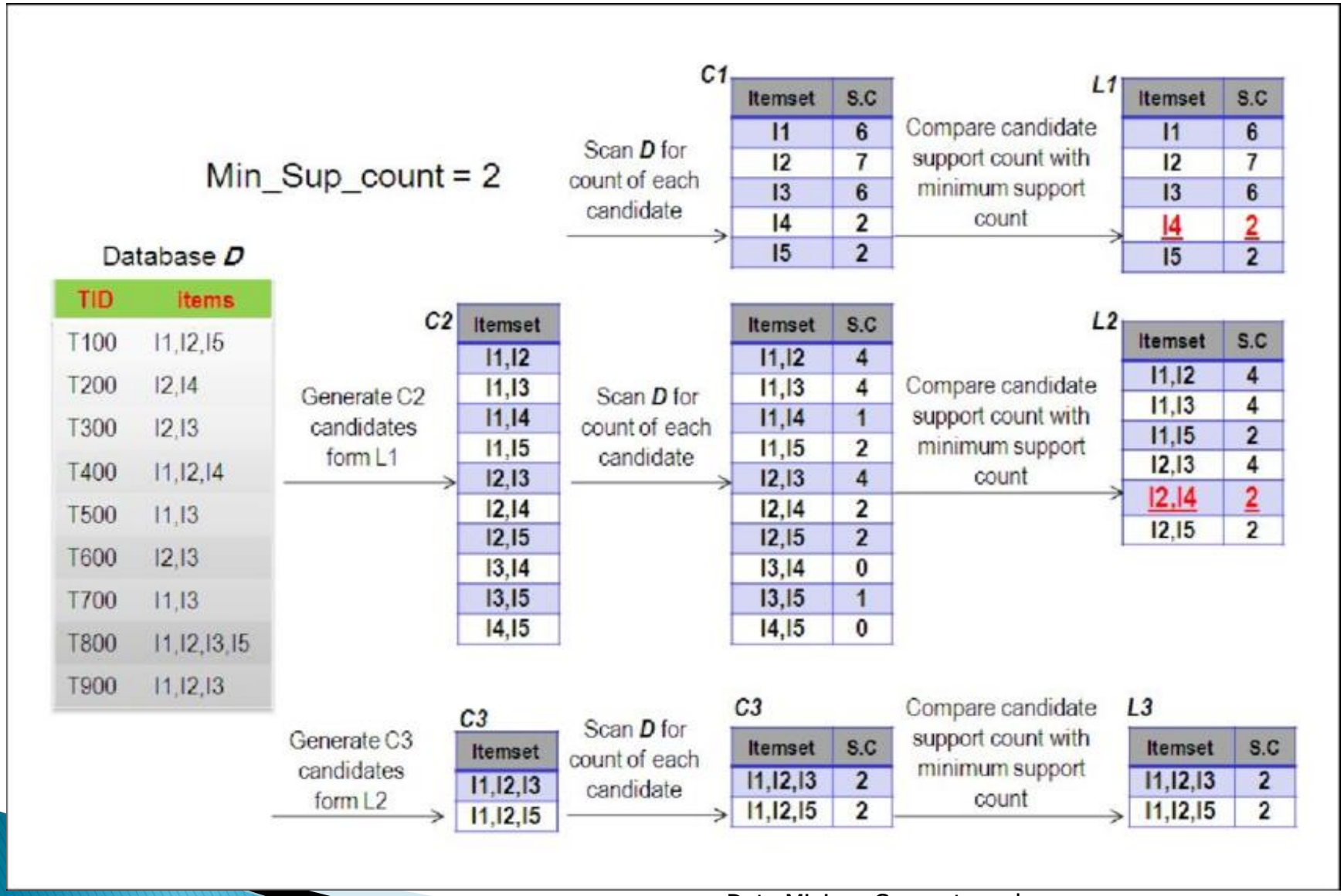
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent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

The Apriori Algorithm—An Example



The Apriori Algorithm -- Example




Min_Sup_count = 2
Database D
TID Items
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
Scan D for count of each candidate
C1
Itemset S.C
I1 6
I2 7
I3 6
I4 2
I5 2
Compare candidate support count with minimum support count
L1
Itemset S.C
I1 6
I2 7
I3 6
I4 2
I5 2
Generate C2 candidates form L1
C2
Itemset
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5
Scan D for count of each candidate
Itemset S.C
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
Compare candidate support count with minimum support count
L2
Itemset S.C
I1,I2 4
I1,I3 4
I1,I5 2
I2,I3 4
I2,I4 2
I2,I5 2
Generate C3 candidates form L2
C3
Itemset
I1,I2,I3
I1,I2,I5
Scan D for count of each candidate
C3
Itemset S.C
I1,I2,I3 2
I1,I2,I5 2
Compare candidate support count with minimum support count
L3
Itemset S.C
I1,I2,I3 2
I1,I2,I5 2

- Join step: is generated by joining with itself
- Prune Step: any (k-1) item set that is not frequent cannot be a subset of a frequent k-item set

The Apriori Algorithm

- Pseudo-code:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
  increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
  return $\cup_k L_k$;

Important Details of Apriori

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

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  forall *itemsets c in* C_k do

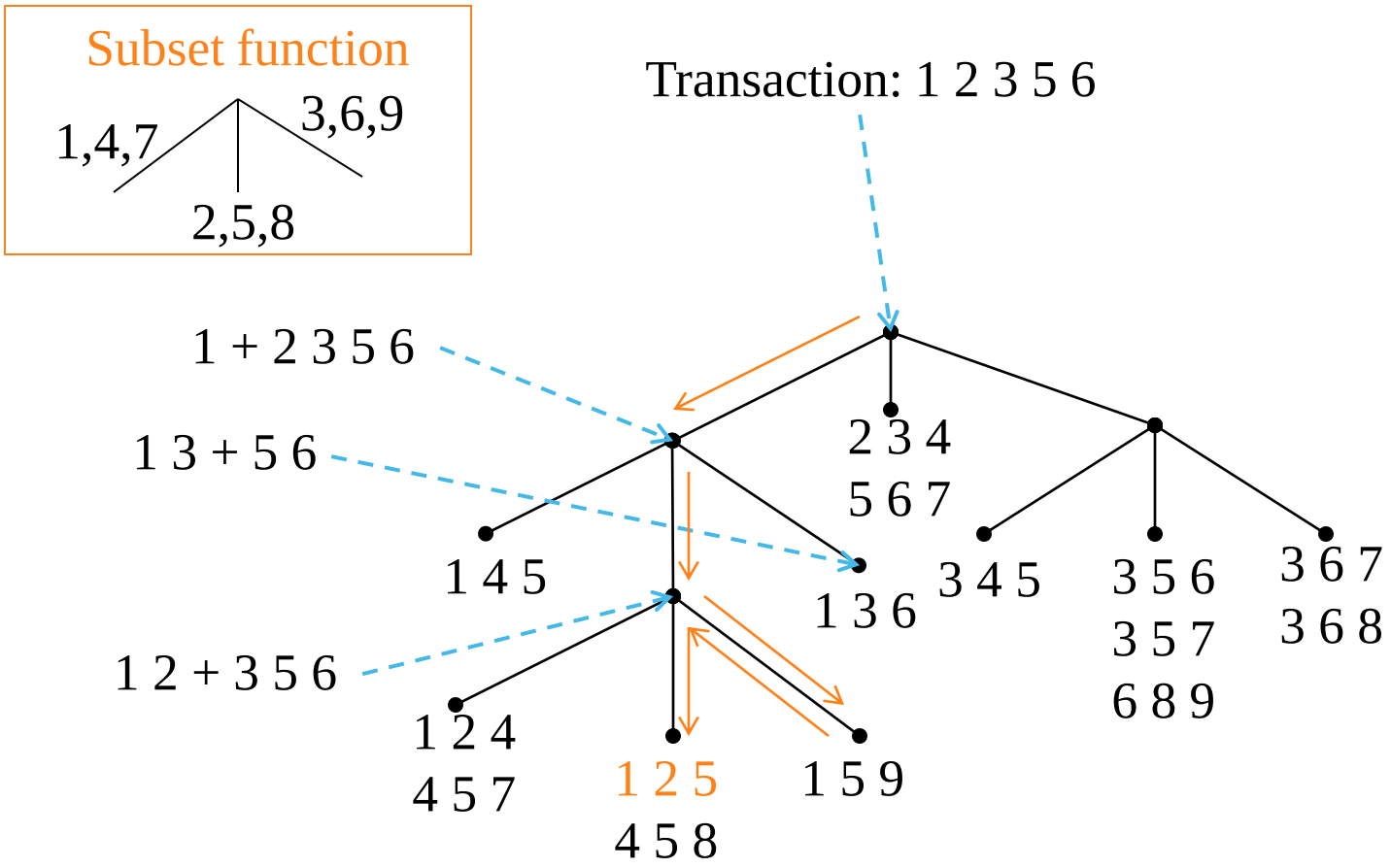
  forall *(k-1)-subsets s of c* do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

How to Count Supports of Candidates?

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

Example: Counting Supports of Candidates



Efficient Implementation of Apriori in SQL

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

Challenges of Frequent Pattern Mining

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

Partition: Scan Database Only Twice

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

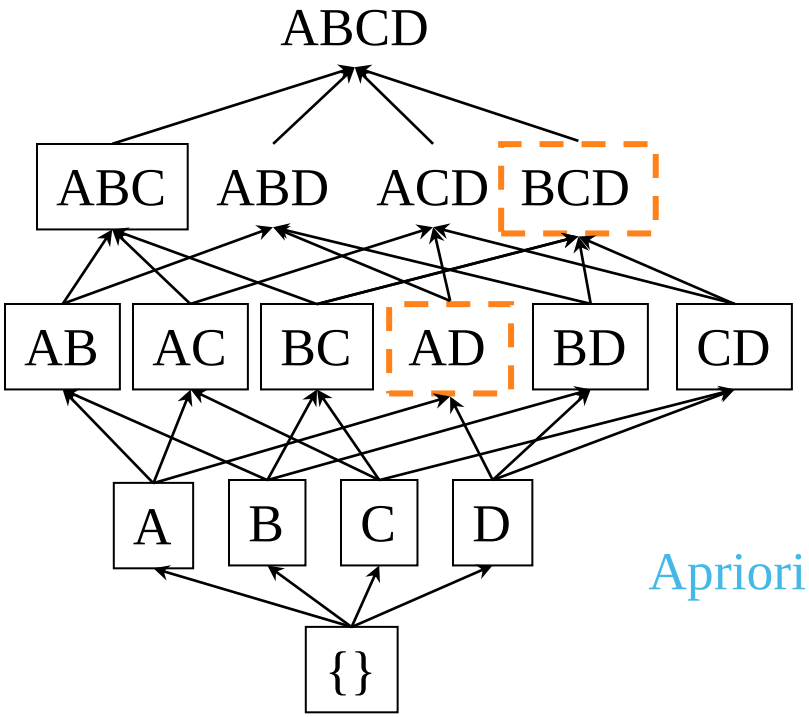
DHP: Reduce the Number of Candidates

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

Sampling for Frequent Patterns

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

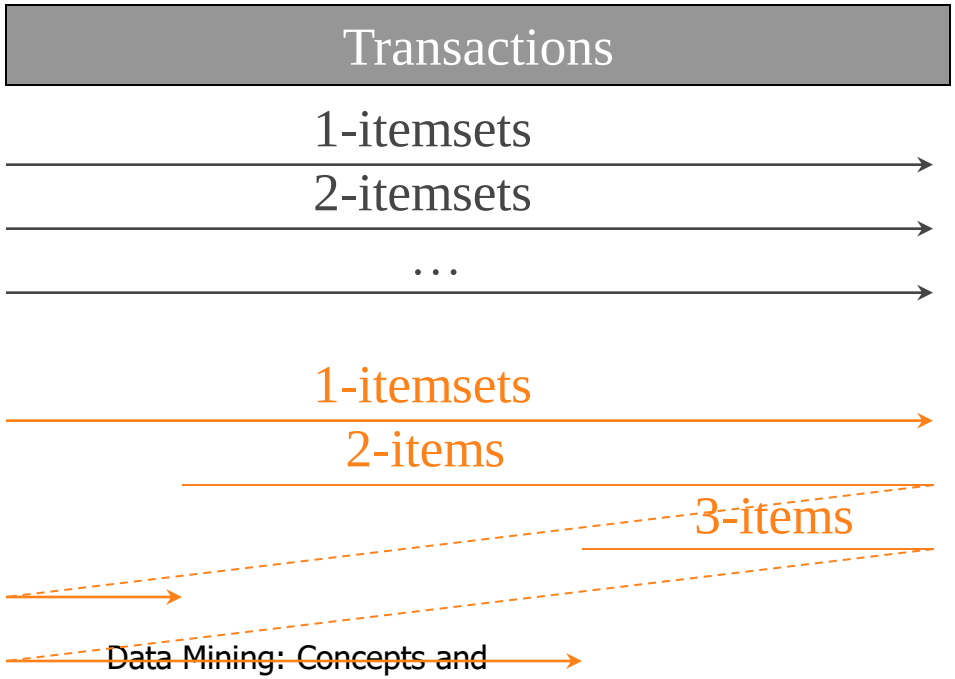
DIC: Reduce Number of Scans



Itemset lattice

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

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



Mining Frequent Patterns Without Candidate Generation

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