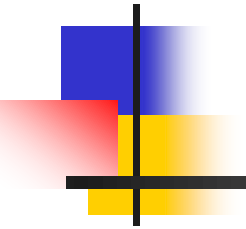


Data Mining – Chapter 6

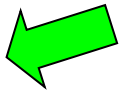
Kyuseok Shim

Seoul National University

http://kdd.snu.ac.kr/~shim

Extended from the slides of the book "Data Mining: Concepts and Techniques (3rd ed.)" provided by Jiawei Han, Micheline Kamber, and Jian Pei

Chapter 5: 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.

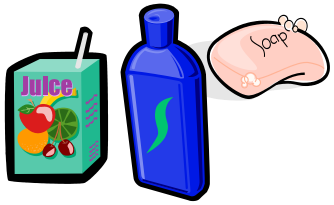
Why Is Freq. Pattern Mining Important?

- 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

Frequent Pattern Mining

- Association rules
  - Apriori
  - FP-tree
- Sequential patterns
  - Apriori
  - PrefixSpan

Association Rules



- 데이터 상호간의 연관 규칙을 찾아내는 기술
- '{라면, 우유}->{커피}'
  - 라면과 우유를 산 사람은 커피도 같이 산다
  - 지지도 (support)
    - 전체 소비자 중에서 그 규칙을 구성하는 물품을 구매한 소비자의 비율
    - 50% - 4명 중 **라면**,**우유**,**커피**를 구매한 사람은 2명
  - 신뢰도 (confidence)
    - 규칙의 왼쪽에 있는 물품을 산 소비자 중에서 오른쪽에 있는 물품들을 산 소비자의 비율
    - 66.7% - **라면**과 **우유**를 산 사람들은 3명인데 그 중에서 **커피**를 산 사람은 2명

사용 사례

- 고객들의 물품 구매 패턴을 분석한 결과에 기반하여
  - 연관 물품 쿠폰이나 할인 행사 제공
  - 온라인 서점에서 다른 구매자들이 구매한 책 정보를 함께 제공

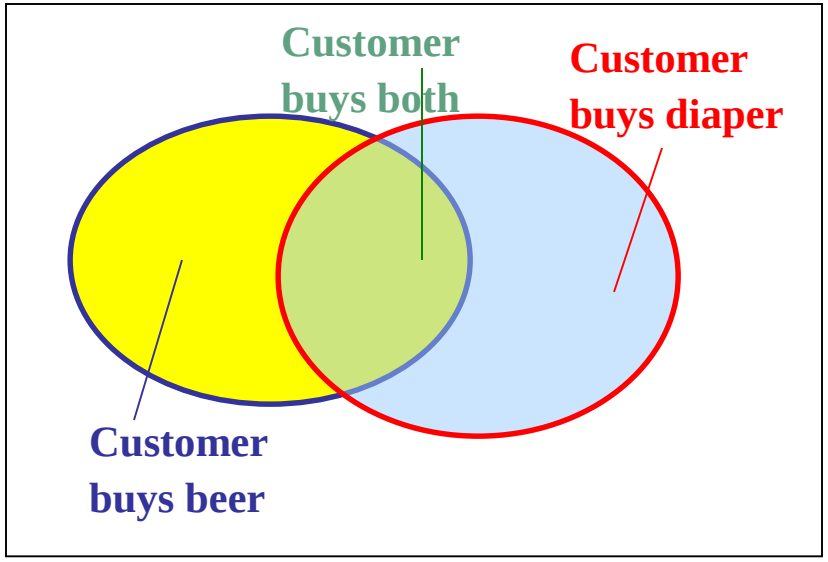
<예: 아마존(amazon.com)의 상품 추천>

사용 사례

- 고객들의 물품 구매 패턴을 분석한 결과에 기반하여
  - Cross-selling (교차 판매) – 서로 다른 카타고리 상품을 추천하여 판매
    - 자동차 보험과 생명보험을 함께 판매하는 온라인 보험회사에서 10 억짜리 생명보험을 가입한 사람에게 자동차보험 대물배상 3억원 추천
    - 11번가에서 디지털카메라를 구매할 때에 명품지갑 추천
  - 백화점의 Package 구매 상품 조합 결정, **물건 진열 순서** 결정 등
  - 효과적인 상품 **카탈로그 디자인**

Basic Concepts: Frequent Patterns

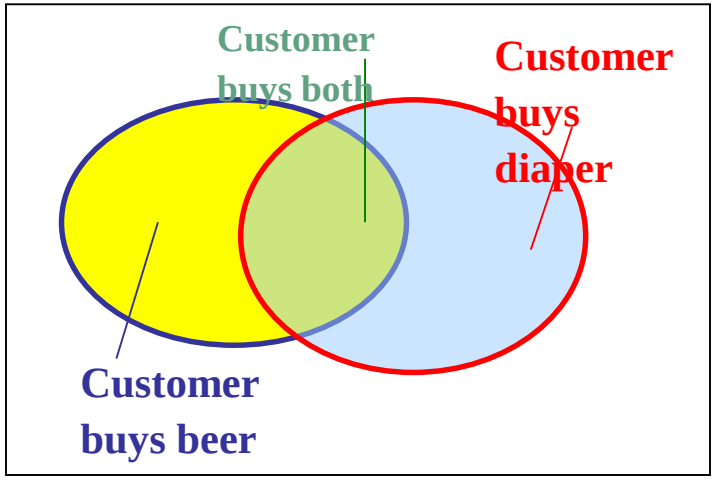
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- 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 $X \cup Y$
  - 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

Association Rules 찾는 방법

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

최소지지도=40%
최소신뢰도 = 100%

- **스텝 1**
  - **최소지지도 를 만족하는 frequent itemset들을 모두 찾음**

Itemset	Sup
a	3
b	3
c	3
e	3
f	2

Itemset	Sup
a,c	2
a,f	2
b,c	2
b,e	3
c,e	2

Itemset	Sup
b,c,e	2

- **스텝 2**
  - **모든 frequent itemset 으로부터 룰 생성**
  - {b,c,e} 에서 아래 룰들을 다 만든 후에 신뢰도를 체크함
    - {b}->{c,e} (X)
    - {c}->{b,e}
    - {e}->{b,c}
    - {b,c}->{e} (O)
    - {b,e}->{c}
    - {c,e}->{b} (O)

Naïve Counting of All Itemsets

Transactions

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

Itemsets & Counts

Itemset	Count
A	1
C	1
D	1
A,C	1
A,D	1
C,D	1
A,C,D	1

Naïve Counting of All Itemsets

Transactions

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

Itemsets & Counts

Itemset	Count
A	1
C	2
D	1
A,C	1
A,D	1
C,D	1
A,C,D	1
B	1
E	1
B,C	1
B,E	1
C,E	1
B,C,E	1

Naïve Counting of All Itemsets

Transactions

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

Itemsets & Counts

Itemset	Count
A	2
C	3
D	1
A,C	2
A,D	1
C,D	1
A,C,D	1
B	2
E	2
B,C	2
B,E	2
C,E	2
B,C,E	2

Itemset	Count
A,B	1
A,E	1
A,B,C	1
A,B,E	1
A,B,C,E	1

Naïve Counting of All Itemsets

Transactions

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

Itemsets & Counts

Itemset	Count
A	2
C	3
D	1
A,C	2
A,D	1
C,D	1
A,C,D	1
B	3
E	3
B,C	2
B,E	3
C,E	2
B,C,E	2

Itemset	Count
A,B	1
A,E	1
A,B,C	1
A,B,E	1
A,B,C,E	1

Naïve Counting of All Itemsets

Transactions

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

$Sup_{min}=2$

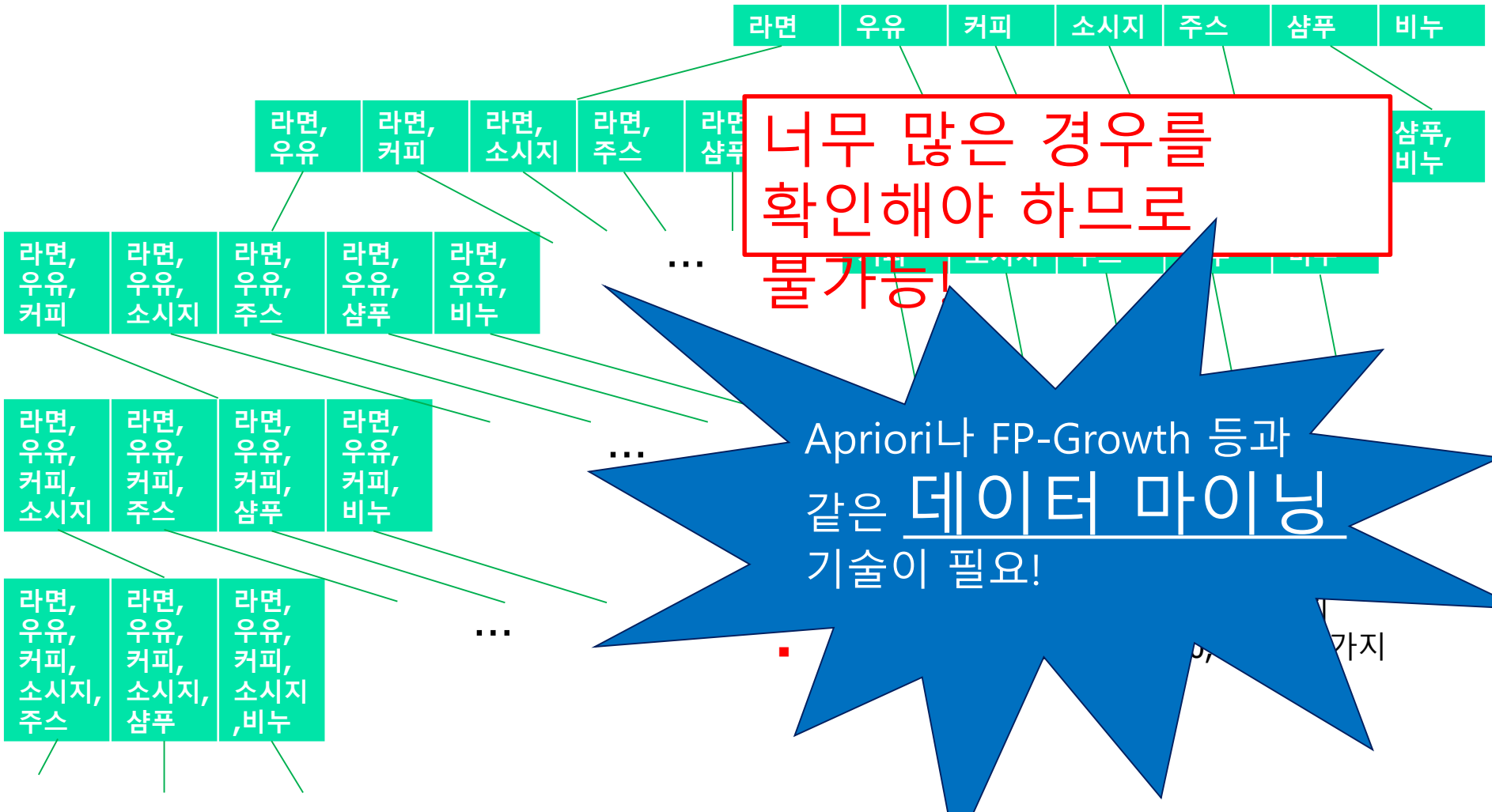
Frequent itemsets

Itemset	Count
A	2
C	3
D	1
A,C	2
A,D	1
C,D	1
A,C,D	1
B	3
E	3
B,C	2
B,E	3
C,E	2
B,C,E	2

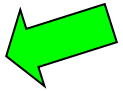
Itemset	Count
A,B	1
A,E	1
A,B,C	1
A,B,E	1
A,B,C,E	1

We may need 2^n itemset entries for counts !

Checking All Combinations?

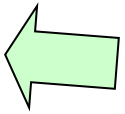


Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

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

Scalable Frequent Itemset Mining Methods

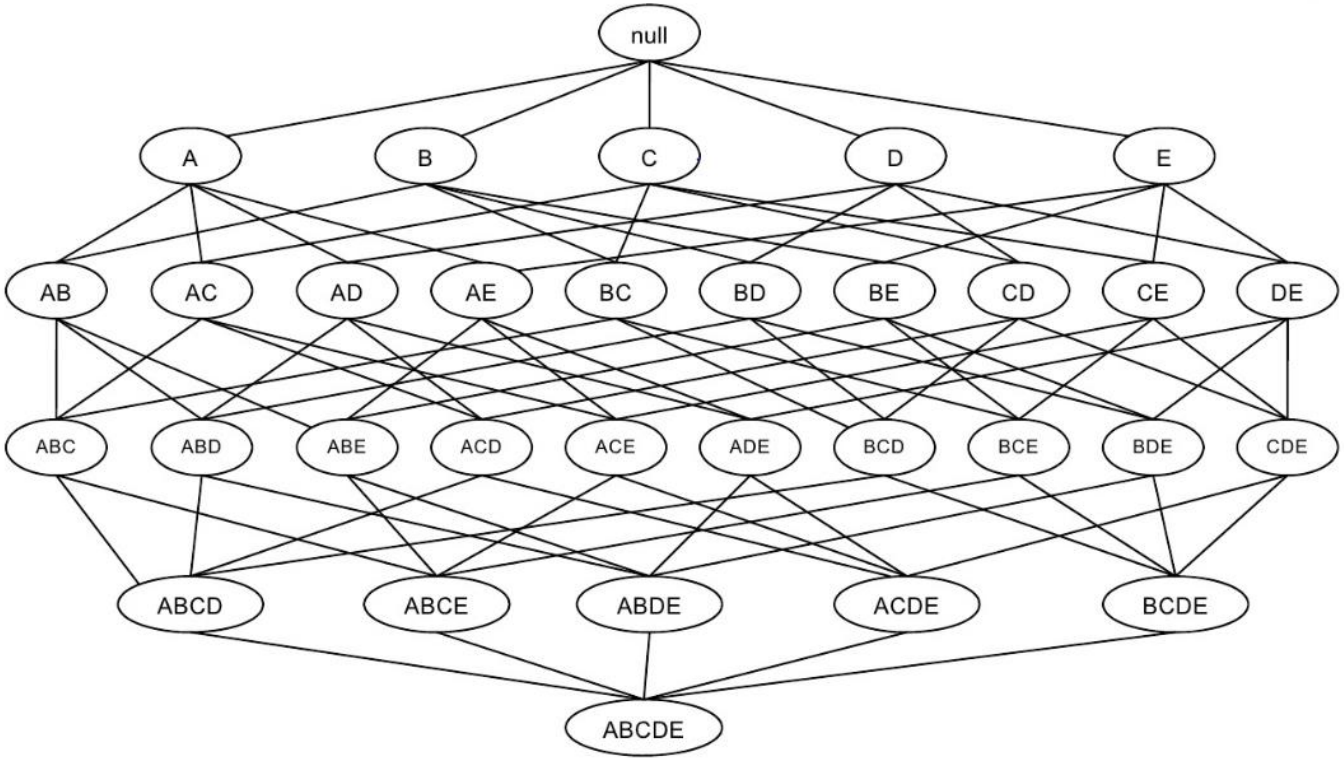
- Apriori: A Candidate Generation-and-Test Approach



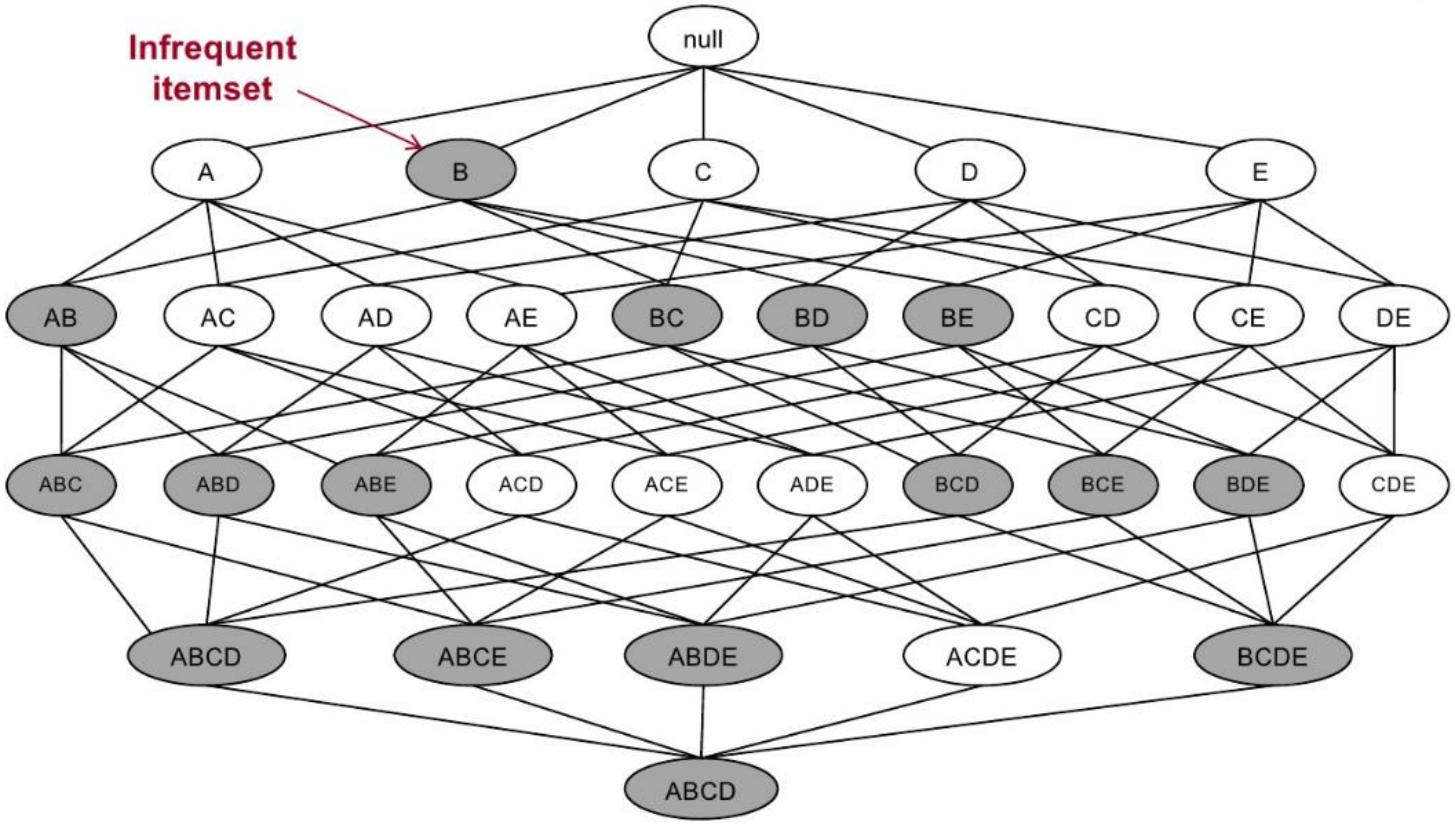
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format

Naïve Counting

- Given d items, there are 2^d itemsets



Candidate Itemset Generation by Apriori



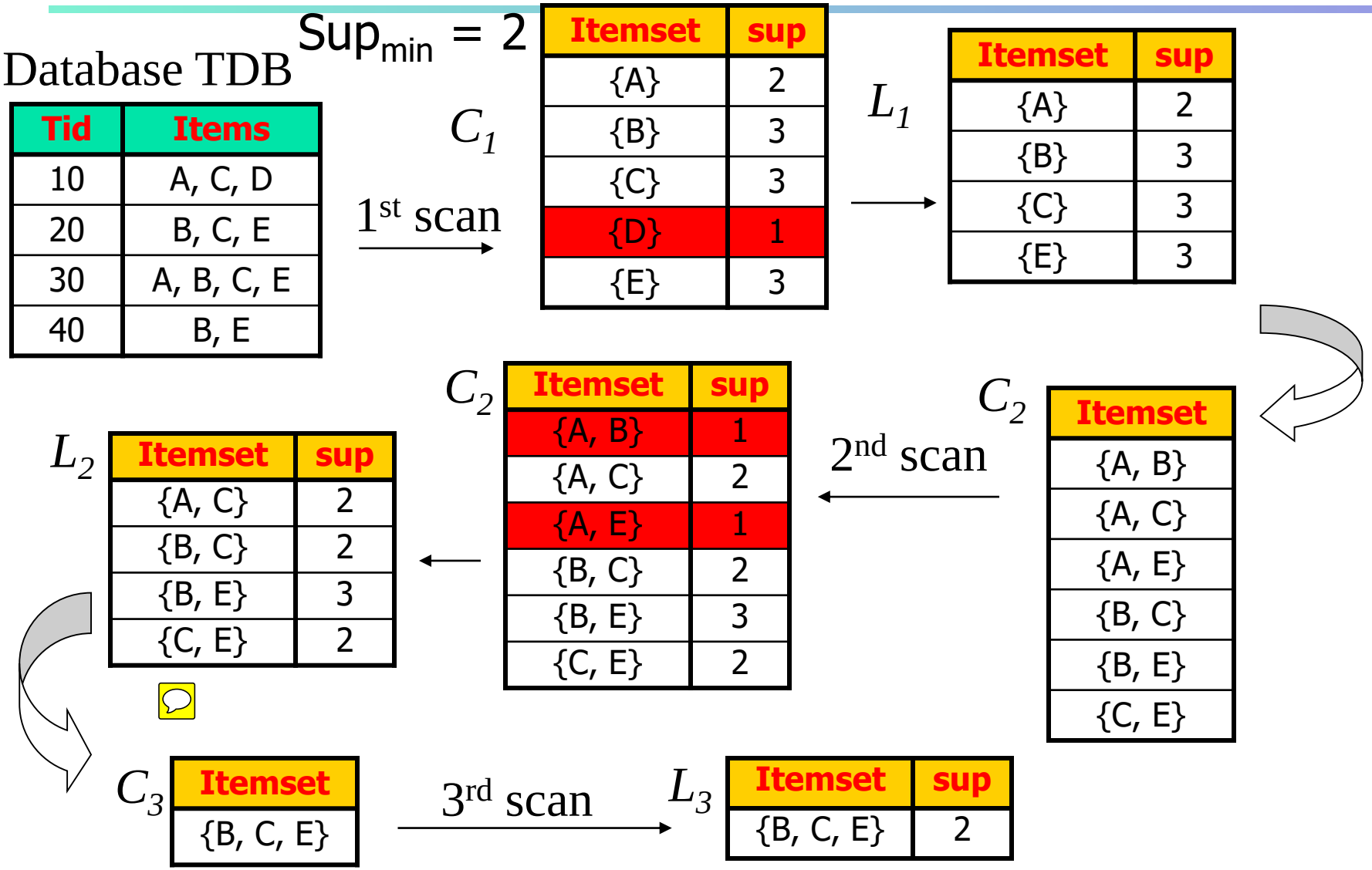
The Downward Closure Property and Scalable Mining Methods

- 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

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]

Discovering Rules

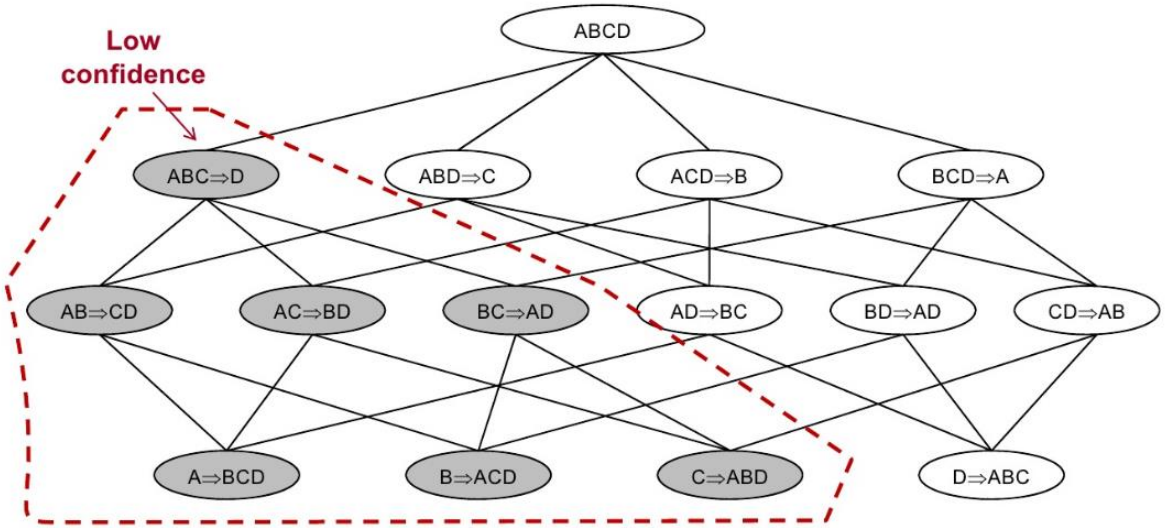
Naïve Algorithm:

for each frequent itemset f **do**
  for each subset c of f **do**
    if (support(f)/support(f-c) ≥ minconf) **then**
      output the rule (f-c)➔ c,
        with confidence = support(f)/support(f-c)
        and support = support(f)

- Given a frequent itemset f
  - Find all non-empty subsets c in f
    - s.t. the rule (f-c)➔ c satisfies the minimum support
  - Output the rule (f-c)➔ c
- Let f = {A,B,C}
  - Candidate itemsets are {A}, {B}, {C}, {A,B}, {A,C}, {B,C}
  - {B,C}->{A}, {A,C}->{B}, {A,B}->{C}, {C}->{A,B},{B}->{A,C},{A}->{B,C}

Can we do better?

- The confidence of rules generated from the same itemset have the anti-monotonicity property
- Let f = {A,B,C,D}
  - Confidence({A,B,C}->{D}) ≥ Confidence({A,B}->{C,D})
  - ≥ Confidence({A}->{B,C,D})



Discovering Rules

- Consider the rule (f-c)➔c
- Now, if c_1 is a subset of c
  - f-c1 is a superset of f-c

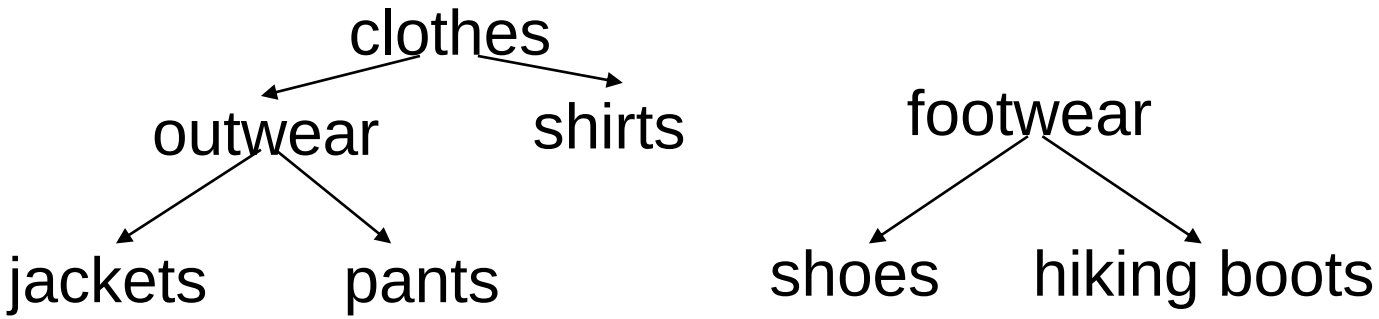
    support(f-c1) ≤ support(f-c)

    support(f)/support(f-c1) ≥ support(f)/support(f-c)

    conf((f-c1)➔c1) ≥ conf((f-c)➔c)
- So, if a consequent c generates a valid rule, so do all subsets of c
- Can use the apriori candidate generation algorithm to limit number of possible rules tested.
- Consider a frequent itemset ABCDE
  - If ACDE➔B and ABCE➔D are the only one-consequent rules with minimum confidence, then ACE ➔ BD is the only other rule that needs to be tested.

Generalized Association Rules

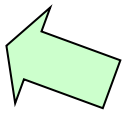
- Hierarchies over items (e.g. UPC codes)



- Associations across hierarchies:
  - The rule clothes => footwear may hold even if clothes => shoes do not hold
- [Srikant, Agrawal 95]
- [Han, Fu 95]

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori



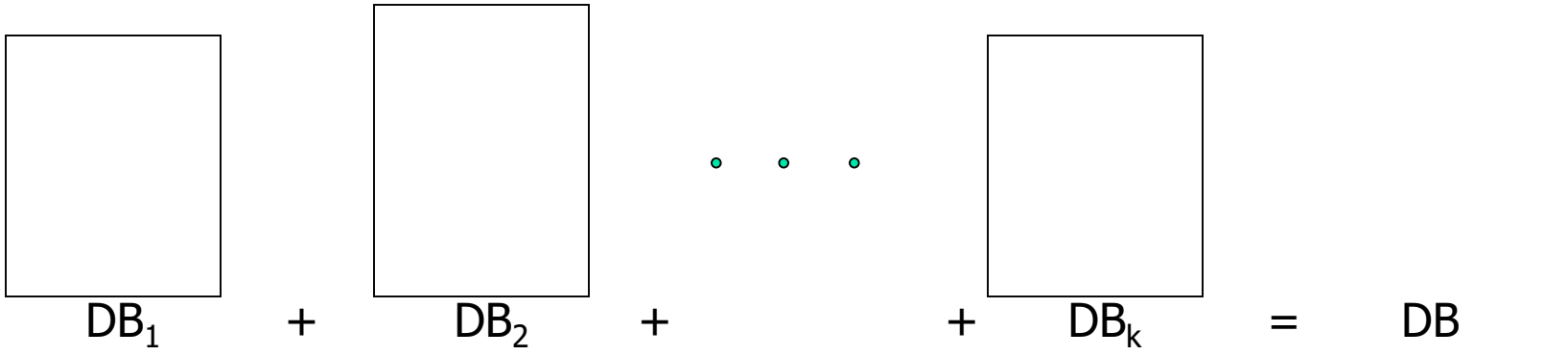
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

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

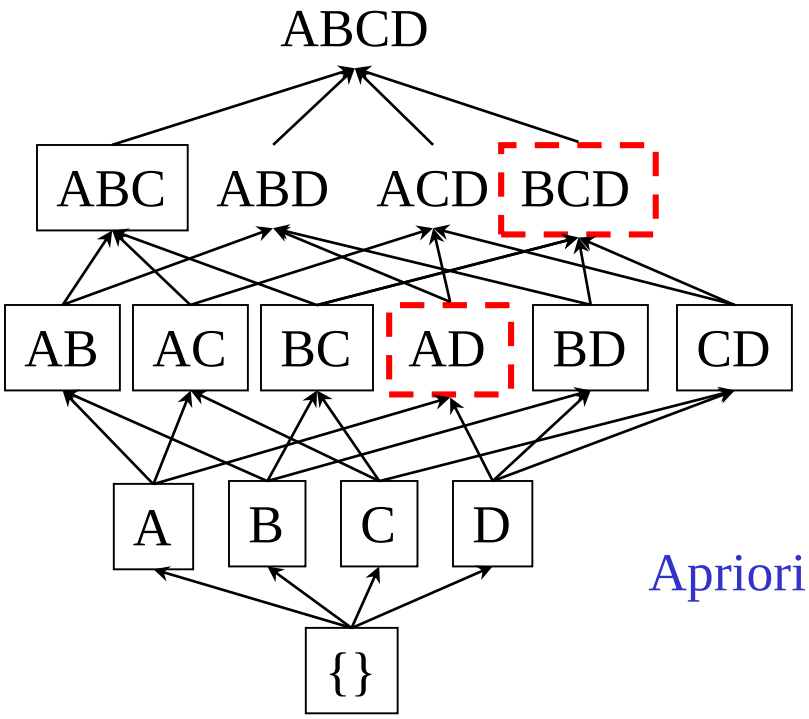
- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries
    - {ab, ad, ae}
    - {bd, be, de}
    - …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is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

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

Hash Table

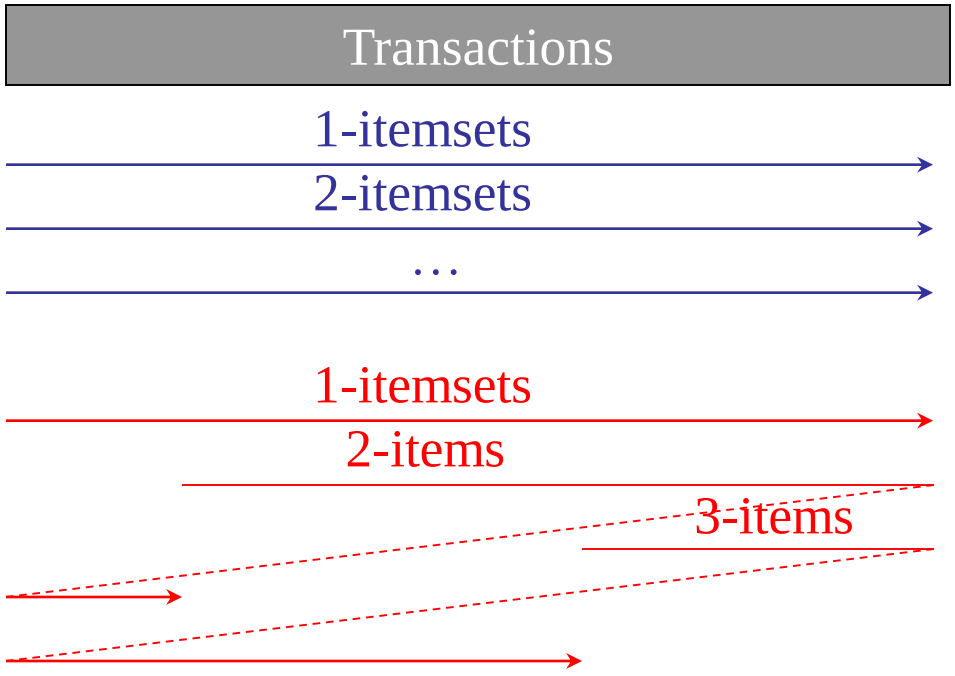
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

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

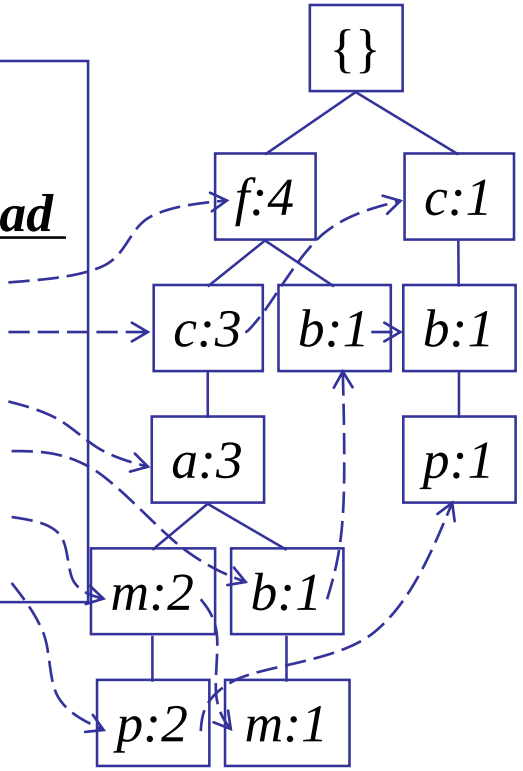
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

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

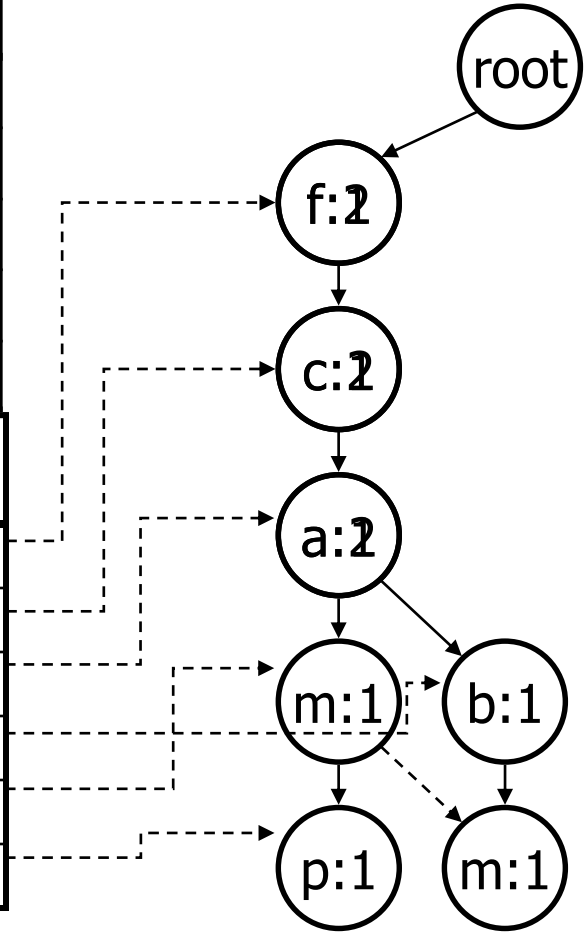
An Example of Construction of FP-tree

Min_sup = 3

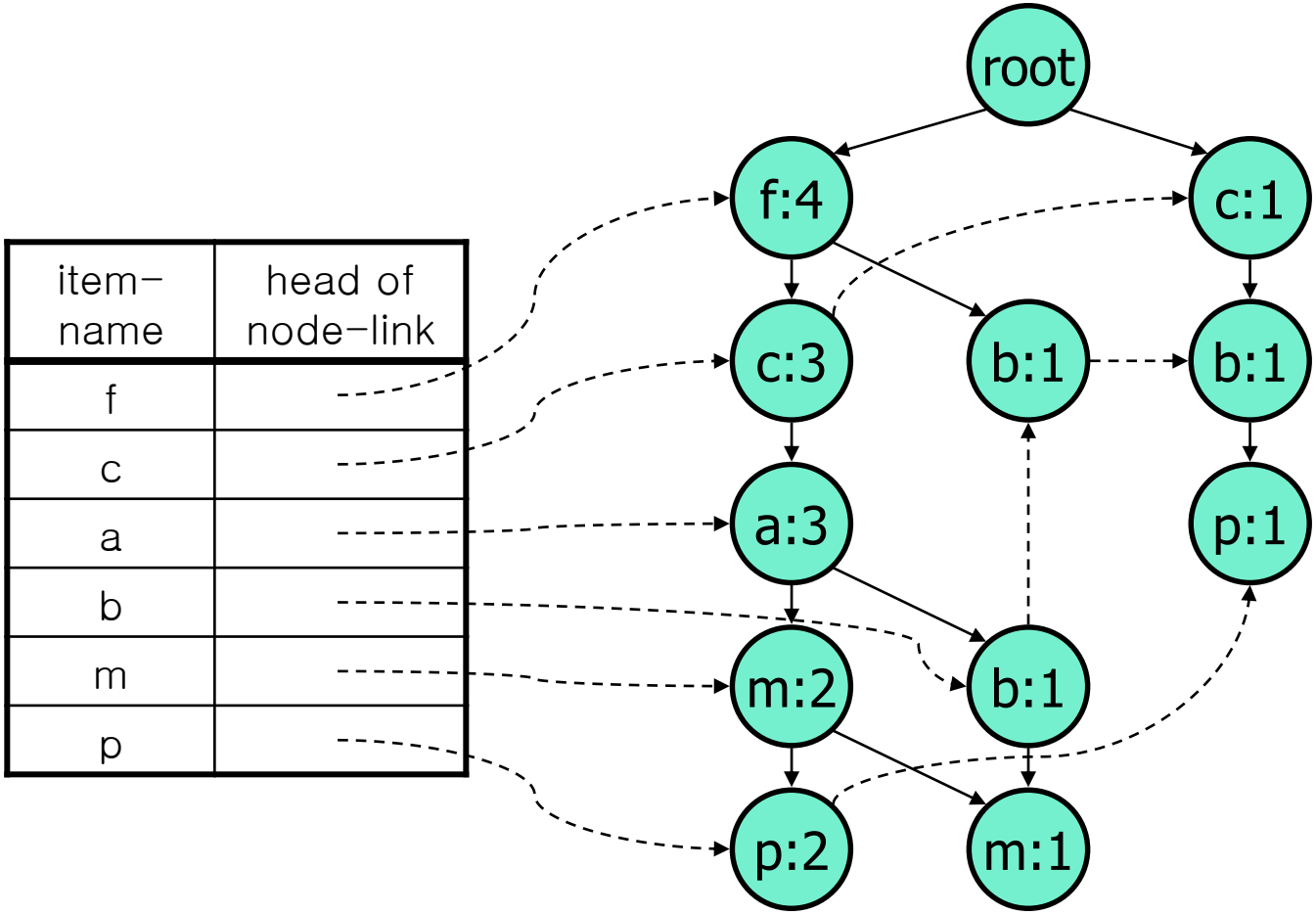
TID	Items bought	(Ordered) frequent items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

item	support
f	4
c	4
a	3
b	3
m	3
p	3

item-name	head of node-link
f	
c	
a	
b	
m	
p	



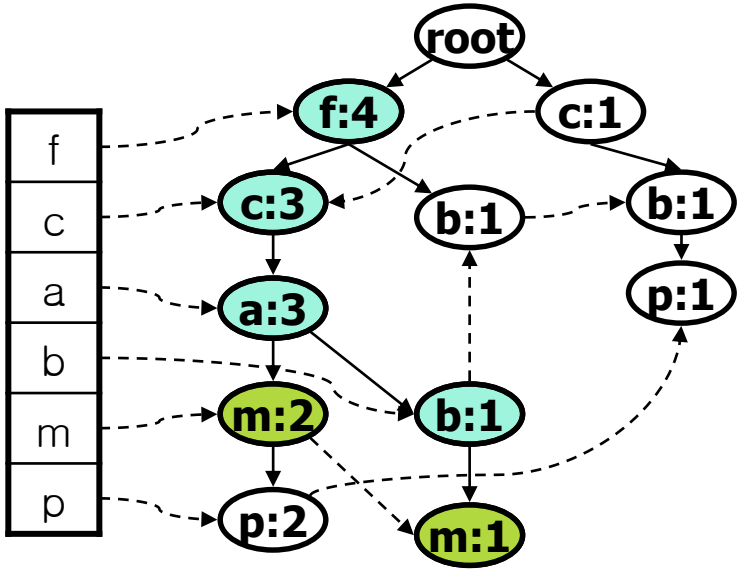
An Example of a complete FP-tree



Conditional Pattern Base

- A sub-pattern base under the condition of existence of a certain pattern

- Example (min_sup = 3)



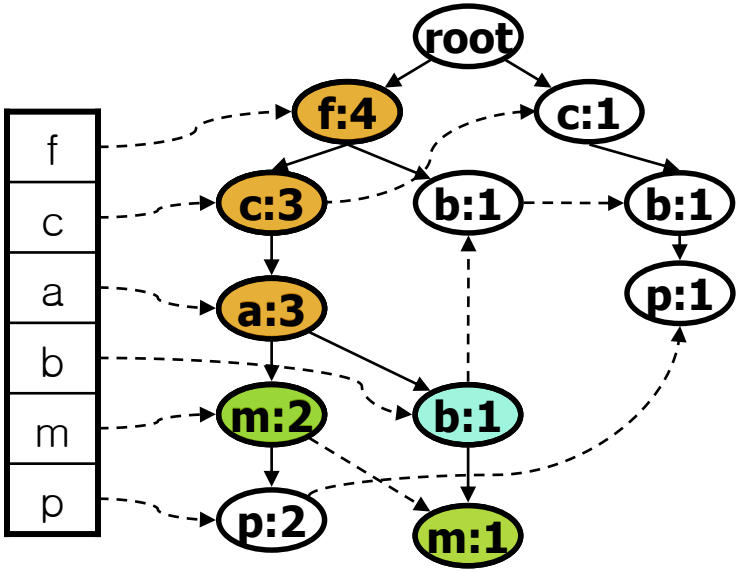
Nodes that contribute m's cond. pattern bases

m's cond. pattern bases

- (fca:2), (fcab:1)

Conditional FP-tree

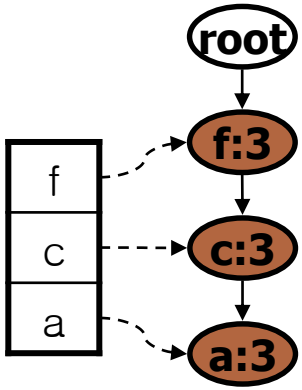
- FP-tree on the conditional pattern bases of a certain item
- If FP-tree consists of single path, all the combinations of items in the path are the freq patterns
- Example (min_sup = 3)



m's cond. pattern bases
- (fca:2), (fcab:1)

The nodes that contribute to m's cond. FP-tree

m's cond. FP-tree
- (fca:3)

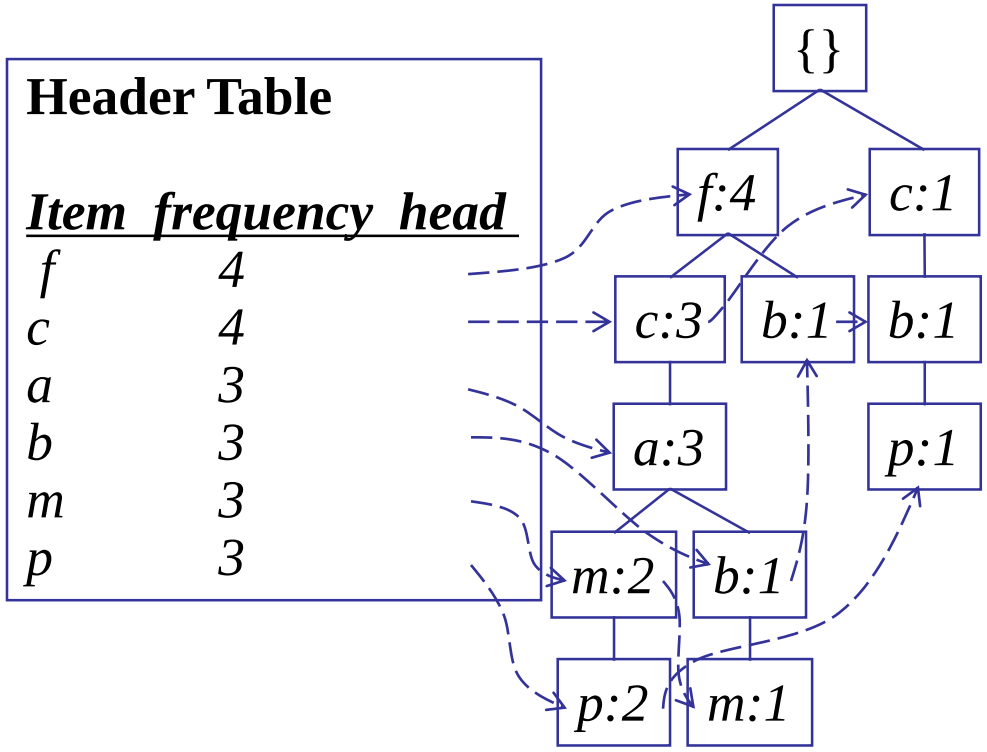


Partition Patterns and Databases

- 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

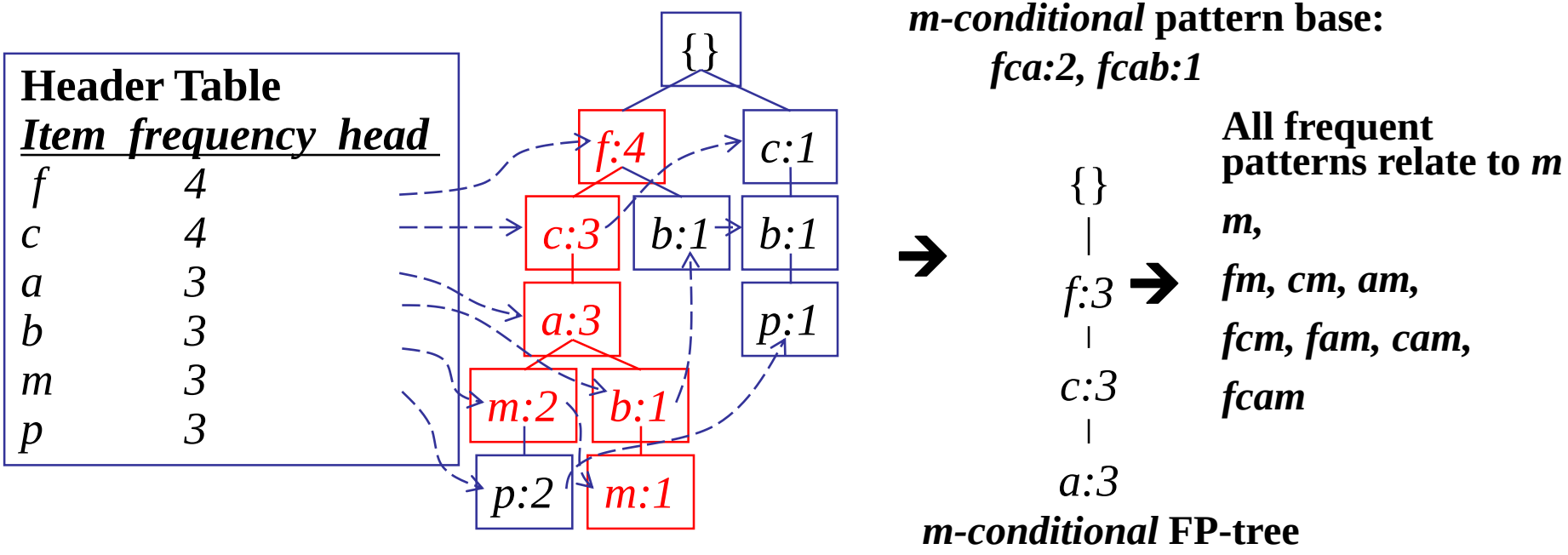


***Conditional* pattern bases**

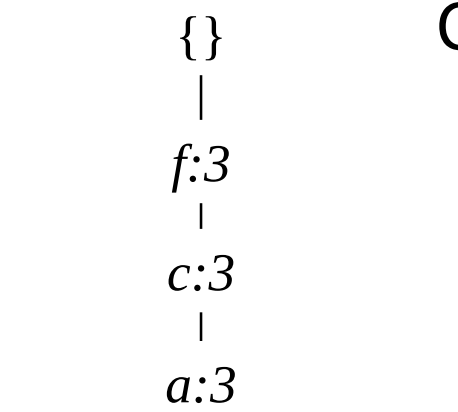
item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

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

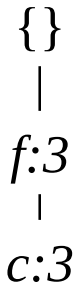


Recursion: Mining Each Conditional FP-tree



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

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

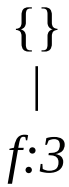


***am-conditional* FP-tree**

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

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

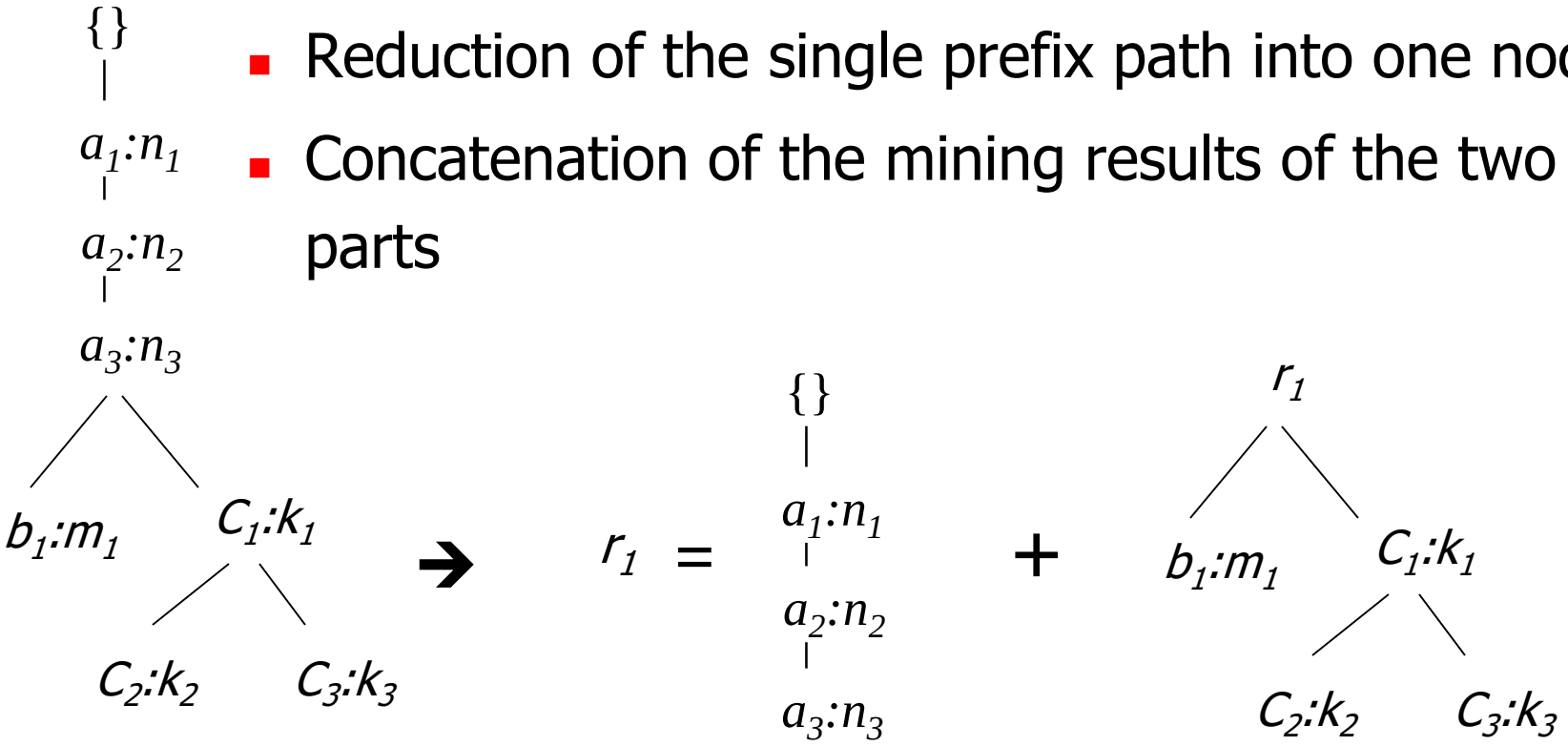
Cond. pattern base of "cam": (f:3)



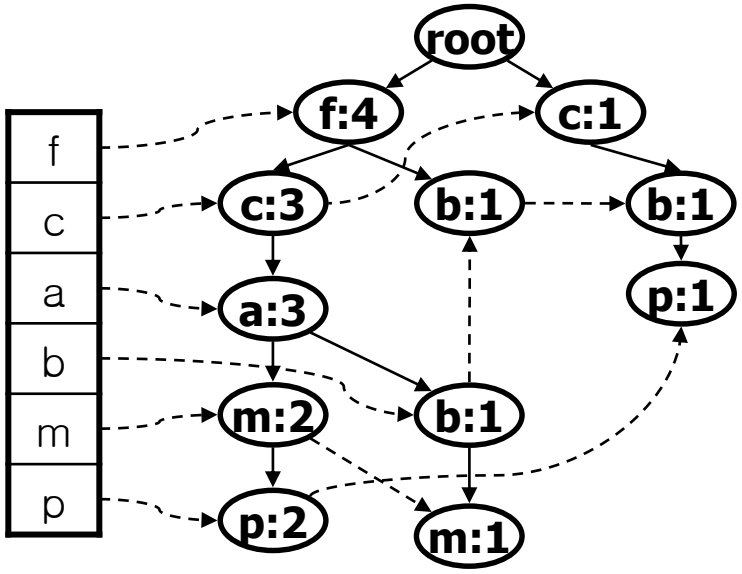
***cam-conditional* FP-tree**

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



An Example of FP-growth



Min_sup=3

item	conditional pattern base	conditional FP-tree	result freq. pattern
p	(fcam:2), (cb:1)	(c:3)	p, cp
m	(fca:2), (fcab:1)	(fca:3)	m, am, cm, fm, cam, fam, fcm, fcam
b	(fca:1), (f:1). (c:1)	none	b
a	(fc:3)	(fc:3)	a, ca, fa, fca
c	(f:3)	(f:3)	c, fc
f	none	none	f

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

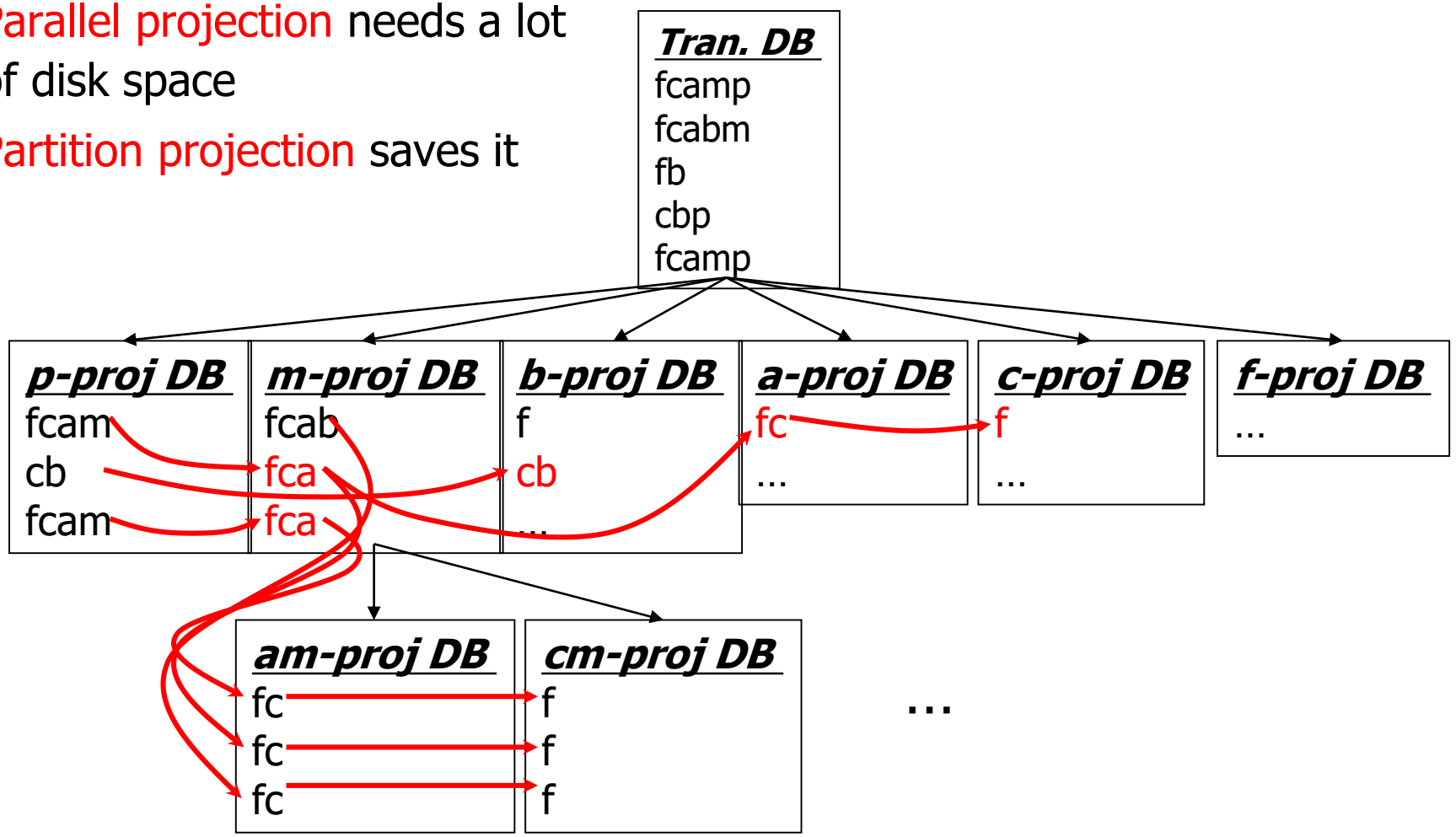
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting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

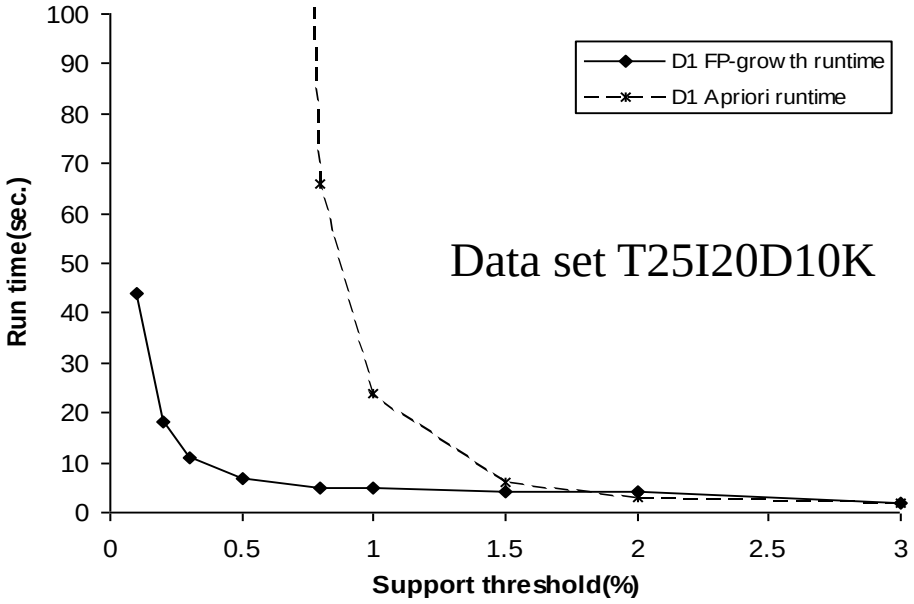
- What about if FP-tree cannot fit in memory?
  - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. partition projection techniques
  - Parallel projection
    - Project the DB in parallel for each 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

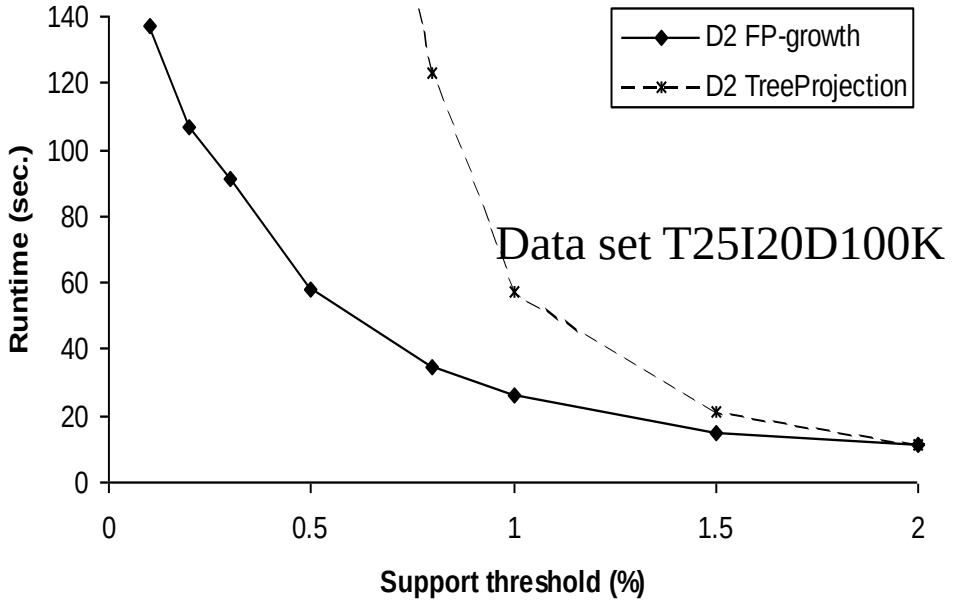
- Parallel projection needs a lot of disk space
- Partition projection saves it



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

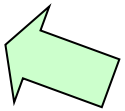
- AFOPT (Liu, et al. @ KDD'03)
  - A "push-right" method for mining condensed frequent pattern (CFP) tree
- Carpenter (Pan, et al. @ KDD'03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining
- FPgrowth+ (Grahne and Zhu, FIMI'03)
  - Efficiently 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

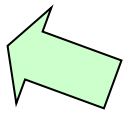
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns



ECLAT: Mining by Exploring Vertical Data Format

- Vertical format: t(AB) = $\{T_{11}, T_{25}, \ldots\}$
  - 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) $\subset$ 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 (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

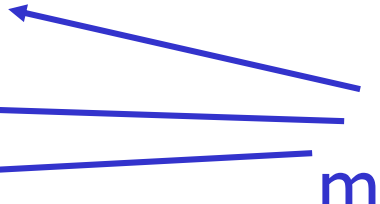
CLOSET+: Mining Closed Itemsets by Pattern-Growth

- 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

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

- 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

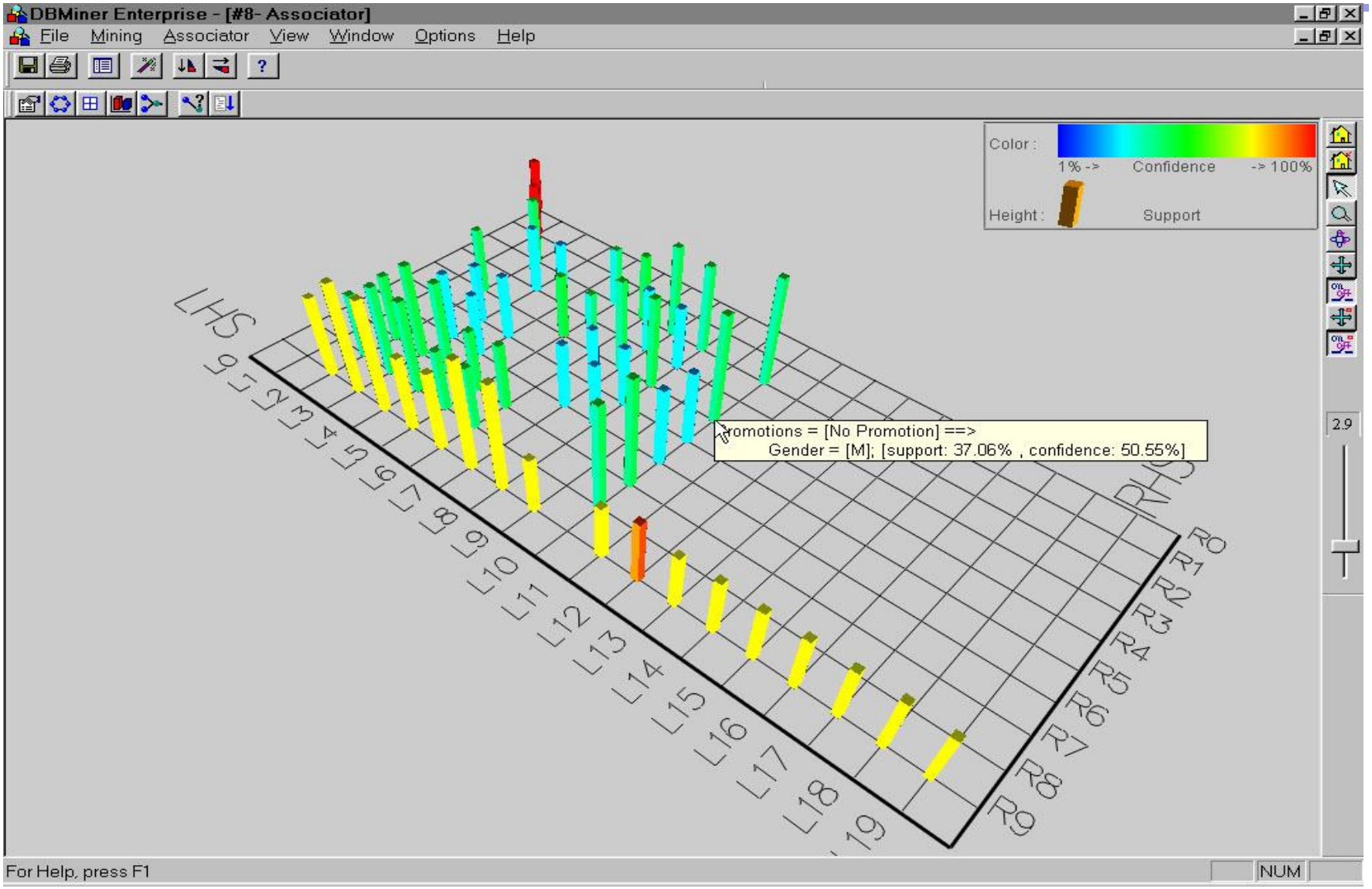


- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
- R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD'98*

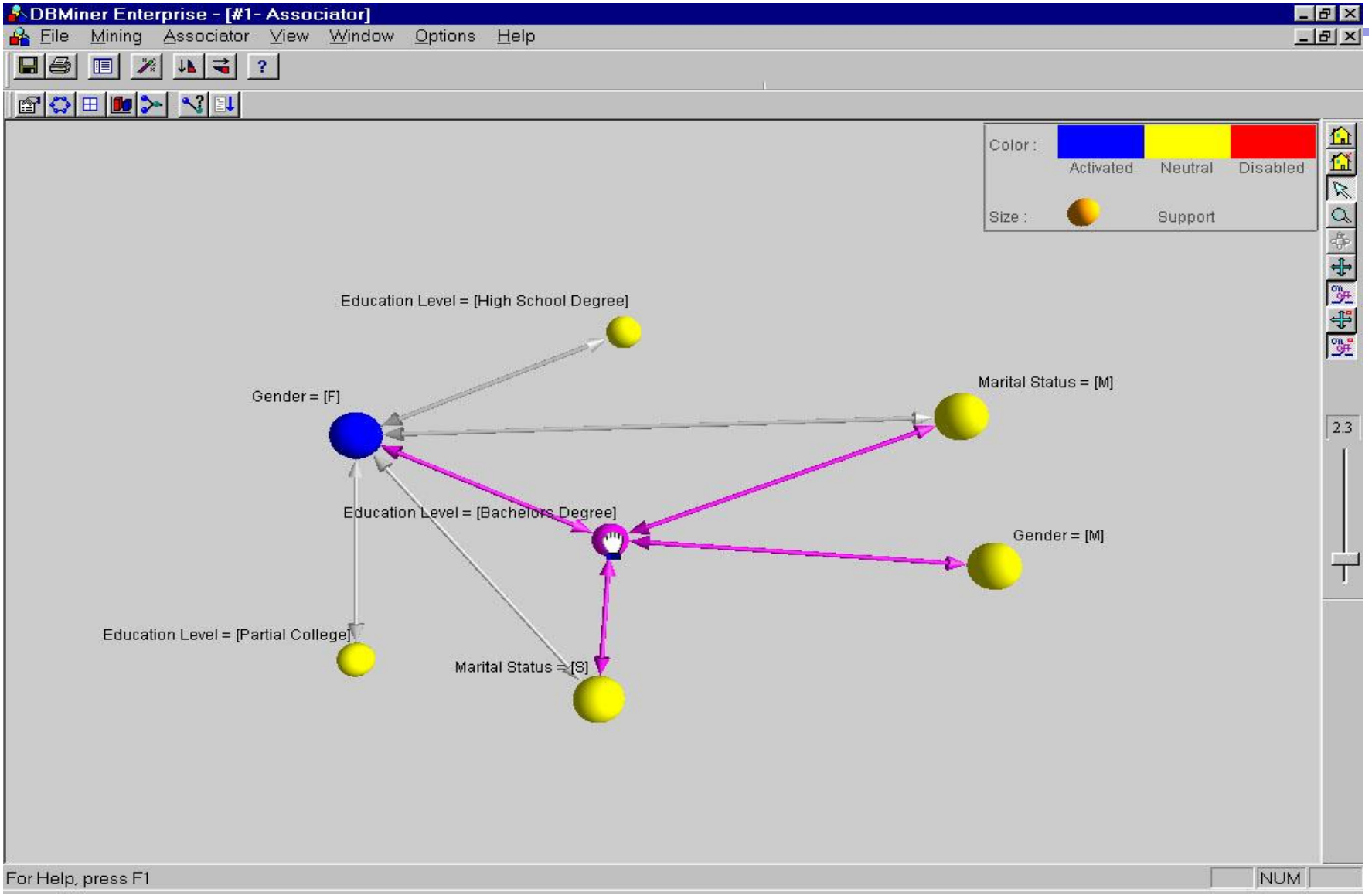
CHARM: Mining by Exploring Vertical Data Format

- 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)

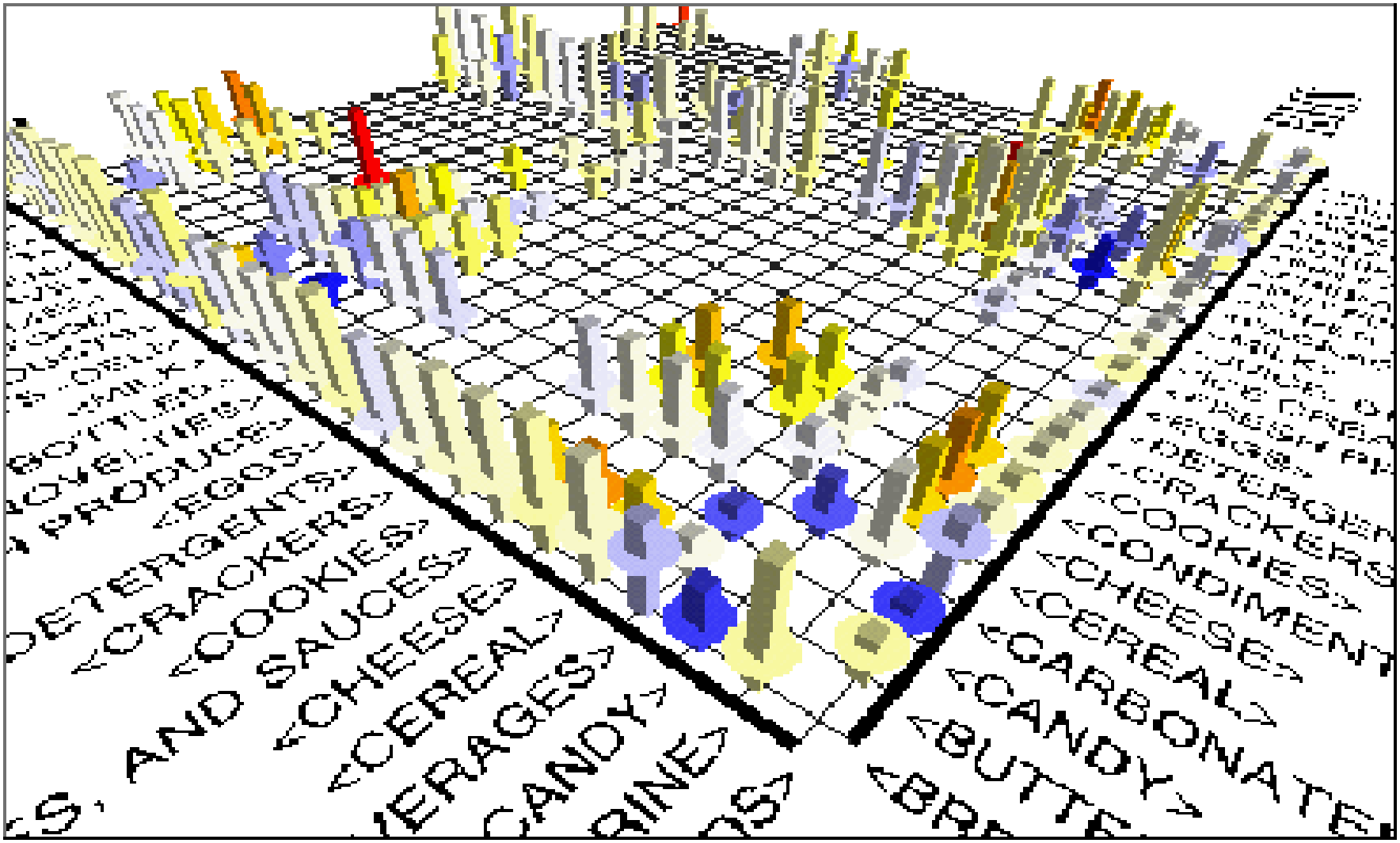
Visualization of Association Rules: Plane Graph



Visualization of Association Rules: Rule Graph

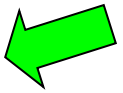


Visualization of Association Rules (SGI/MineSet 3.0)



Chapter 5: 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)

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

- If lift < 1, then the occurrence of A is negatively correlated with the occurrence of B
  - The occurrence of one likely leads to the absence of the other one.
- If lift > 1, then A and B are positively correlated
  - The occurrence of one implies the occurrence of the other.
- If lift = 1, then A and B are independent
  - There is no correlation between them.

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	$\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$
M	Mutual Information	0 ...1	$\frac{\Sigma_i\Sigma_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\Sigma_i P(A_i)\log P(A_i)\log P(A_i),-\Sigma_i P(B_i)\log P(B_i)\log P(B_i))}$
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 ...∞	$\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$

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 c

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

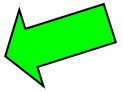
$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

- 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

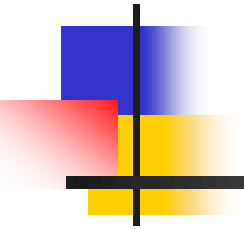
Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

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



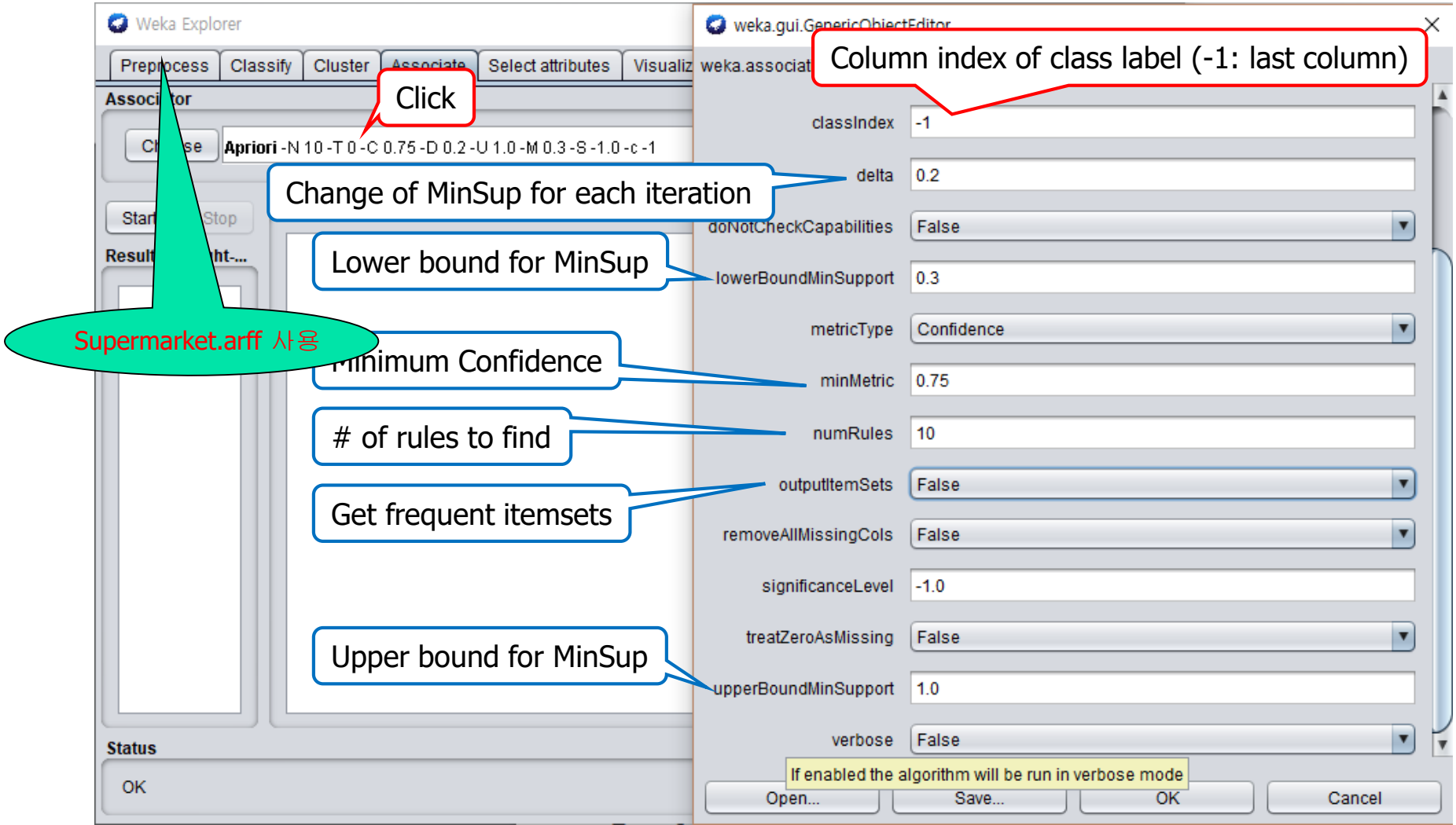
Summary

- 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



Weka - Association Rule

Association Rule - Apriori Algorithm



Association Rule - Apriori Algorithm

Apriori -N 10 -T 0 -C 0.75 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1

- Setting
  - Number of rules: 10
  - Upper bound for MinSup: 1.0
  - Lower bound for MinSup: 0.3
  - Minimum confidence: 0.75
  - Delta : 0.05

- Result
  - Minimum support: 0.35
  - # of rules: 10

```
Minimum support: 0.35 (1619 instances)
Minimum metric <confidence>: 0.75
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 22

Size of set of large itemsets L(2): 36

Size of set of large itemsets L(3): 3

Best rules found:

 1. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    conf:(0.83)
 2. milk-cream=t vegetables=t 2025 ==> bread and cake=t 1658    conf:(0.82)
 3. fruit=t vegetables=t 2207 ==> bread and cake=t 1791    conf:(0.81)
 4. margarine=t 2288 ==> bread and cake=t 1831    conf:(0.8)
 5. biscuits=t 2605 ==> bread and cake=t 2083    conf:(0.8)
 6. milk-cream=t 2939 ==> bread and cake=t 2337    conf:(0.8)
 7. tissues-paper prd=t 2247 ==> bread and cake=t 1776    conf:(0.79)
 8. fruit=t 2962 ==> bread and cake=t 2325    conf:(0.78)
 9. baking needs=t 2795 ==> bread and cake=t 2191    conf:(0.78)
10. frozen foods=t 2717 ==> bread and cake=t 2129    conf:(0.78)
```

Association Rule - Apriori Algorithm

Apriori -N 10 -T 0 -C 0.75 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1

- Setting
  - Number of rules: 10
  - Upper bound for MinSup: 1.0
  - Lower bound for MinSup: 0.3
  - Minimum confidence: 0.75
  - Delta : 0.2
- Result
  - Minimum support: 0.3
  - # of rules: 10

```
Minimum support: 0.3 (1388 instances)
Minimum metric <confidence>: 0.75
Number of cycles performed: 4

Generated sets of large itemsets:

Size of set of large itemsets L(1): 25

Size of set of large itemsets L(2): 69

Size of set of large itemsets L(3): 20

Best rules found:

 1. biscuits=t vegetables=t 1764 ==> bread and cake=t 1487    <conf:(0.84)> l
 2. total=high 1679 ==> bread and cake=t 1413    <conf:(0.84)> lift:(1.17) le
 3. biscuits=t milk-cream=t 1767 ==> bread and cake=t 1485    <conf:(0.84)> l
 4. biscuits=t fruit=t 1837 ==> bread and cake=t 1541    <conf:(0.84)> lift:(
 5. biscuits=t frozen foods=t 1810 ==> bread and cake=t 1510    <conf:(0.83)>
 6. frozen foods=t fruit=t 1861 ==> bread and cake=t 1548    <conf:(0.83)> li
 7. frozen foods=t milk-cream=t 1826 ==> bread and cake=t 1516    <conf:(0.83
 8. baking needs=t milk-cream=t 1907 ==> bread and cake=t 1580    <conf:(0.83
 9. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    <conf:(0.83)> lift
10. baking needs=t biscuits=t 1764 ==> bread and cake=t 1456    <conf:(0.83)>
```

Rule set changes

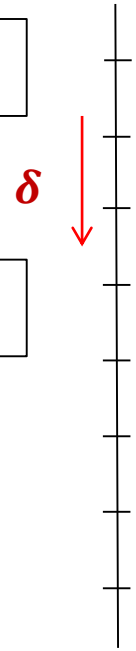
Association Rule - Apriori Algorithm

- $\delta = 0.2$

Upper bound = 1.0

# of rules = 10

MinSup = 1.0



Found 10 rules??

-> NO!

MinSup = 0.8

Iterate until 10 rules found

Lower bound = 0.3

Association Rule - Apriori Algorithm

Apriori -N 10000 -T 0 -C 0.75 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1

- Setting
  - Number of rules: 10000
    - To find all rules that satisfy the given support and confidence
  - Lower bound for minSup: 0.3
  - Minimum confidence: 0.75
- Result
  - Minimum support: 0.3
    - 1388 instances
  - # of rules: 42

```
Minimum support: 0.3 (1388 instances)
Minimum metric <confidence>: 0.75
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 25

Size of set of large itemsets L(2): 69

Size of set of large itemsets L(3): 20

Best rules found:

    1. biscuits=t vegetables=t 1764 ==> bread and cake=t 1487    conf:(0.84)
    2. total=high 1679 ==> bread and cake=t 1413    conf:(0.84)
    3. biscuits=t milk-cream=t 1767 ==> bread and cake=t 1485    conf:(0.84)
    4. biscuits=t fruit=t 1837 ==> bread and cake=t 1541    conf:(0.84)
    5. biscuits=t frozen foods=t 1810 ==> bread and cake=t 1510    conf:(0.83)
    6. frozen foods=t fruit=t 1861 ==> bread and cake=t 1548    conf:(0.83)
    7. frozen foods=t milk-cream=t 1826 ==> bread and cake=t 1516    conf:(0.83)
    8. baking needs=t milk-cream=t 1907 ==> bread and cake=t 1580    conf:(0.83)
    9. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    conf:(0.83)
   10. baking needs=t biscuits=t 1764 ==> bread and cake=t 1456    conf:(0.83)
   11. baking needs=t fruit=t 1900 ==> bread and cake=t 1564    conf:(0.82)
   12. frozen foods=t vegetables=t 1882 ==> bread and cake=t 1548    conf:(0.82)
   13. milk-cream=t vegetables=t 2025 ==> bread and cake=t 1658    conf:(0.82)
   14. baking needs=t vegetables=t 1949 ==> bread and cake=t 1586    conf:(0.81)
   15. fruit=t vegetables=t 2207 ==> bread and cake=t 1791    conf:(0.81)
   16. baking needs=t frozen foods=t 1835 ==> bread and cake=t 1485    conf:(0.81)
   17. margarine=t 2288 ==> bread and cake=t 1831    conf:(0.8)
   18. biscuits=t 2605 ==> bread and cake=t 2083    conf:(0.8)
   19. pet foods=t 1867 ==> bread and cake=t 1490    conf:(0.8)
   20. biscuits=t vegetables=t 1764 ==> fruit=t 1404    conf:(0.8)
   21. milk-cream=t 2939 ==> bread and cake=t 2337    conf:(0.8)
```

Association Rule - Apriori Algorithm

Apriori -N 10000 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1

- Setting
  - Lower bound for minSup: 0.3
  - Minimum confidence: 0.8
- Result
  - Minimum support: 0.3
    - 1388 instances
  - # of rules: 17

```
Minimum support: 0.3 (1388 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 25

Size of set of large itemsets L(2): 69

Size of set of large itemsets L(3): 20

Best rules found:

 1. biscuits=t vegetables=t 1764 ==> bread and cake=t 1487    conf:(0.84)
 2. total=high 1679 ==> bread and cake=t 1413    conf:(0.84)
 3. biscuits=t milk-cream=t 1767 ==> bread and cake=t 1485    conf:(0.84)
 4. biscuits=t fruit=t 1837 ==> bread and cake=t 1541    conf:(0.84)
 5. biscuits=t frozen foods=t 1810 ==> bread and cake=t 1510    conf:(0.83)
 6. frozen foods=t fruit=t 1861 ==> bread and cake=t 1548    conf:(0.83)
 7. frozen foods=t milk-cream=t 1826 ==> bread and cake=t 1516    conf:(0.83)
 8. baking needs=t milk-cream=t 1907 ==> bread and cake=t 1580    conf:(0.83)
 9. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    conf:(0.83)
10. baking needs=t biscuits=t 1764 ==> bread and cake=t 1456    conf:(0.83)
11. baking needs=t fruit=t 1900 ==> bread and cake=t 1564    conf:(0.82)
12. frozen foods=t vegetables=t 1882 ==> bread and cake=t 1548    conf:(0.82)
13. milk-cream=t vegetables=t 2025 ==> bread and cake=t 1658    conf:(0.82)
14. baking needs=t vegetables=t 1949 ==> bread and cake=t 1586    conf:(0.81)
15. fruit=t vegetables=t 2207 ==> bread and cake=t 1791    conf:(0.81)
16. baking needs=t frozen foods=t 1835 ==> bread and cake=t 1485    conf:(0.81)
17. margarine=t 2288 ==> bread and cake=t 1831    conf:(0.8)
```

Association Rule - Apriori Algorithm

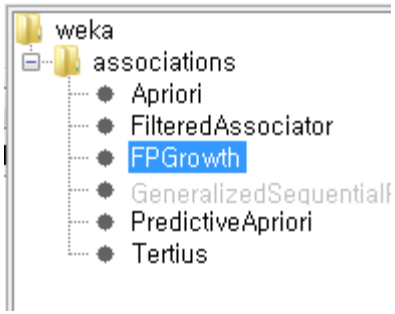
Apriori -N 10000 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.25 -S -1.0 -c -1

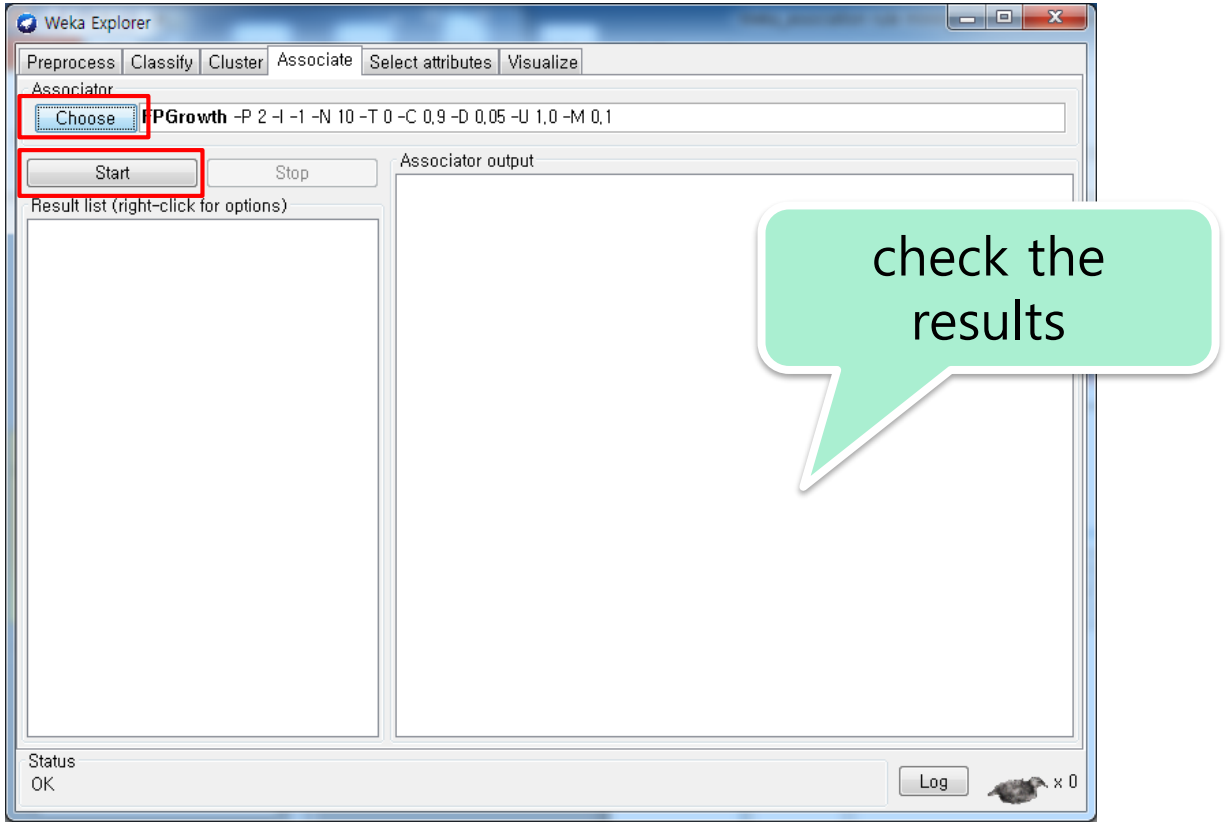
- Setting
  - Lower bound for MinSup: 0.25
  - Minimum confidence: 0.8
- Result
  - Minimum support: 0.25
  - # of rules: 53

```
Size of set of large itemsets L(1): 30

Size of set of large itemsets L(2): 132

Size of set of large itemsets L(3): 79

Size of set of large itemsets L(4): 6

Best rules found:

 1. biscuits=t fruit=t vegetables=t 1404 ==> bread and cake=t 1216    conf:(0.87)
 2. frozen foods=t fruit=t vegetables=t 1451 ==> bread and cake=t 1242    conf:(0.86)
 3. baking needs=t milk-cream=t fruit=t 1365 ==> bread and cake=t 1161    conf:(0.85)
 4. margarine=t fruit=t 1538 ==> bread and cake=t 1301    conf:(0.85)
 5. biscuits=t vegetables=t 1764 ==> bread and cake=t 1487    conf:(0.84)
 6. baking needs=t fruit=t vegetables=t 1489 ==> bread and cake=t 1255    conf:(0.84)
 7. tissues-paper prd=t milk-cream=t 1514 ==> bread and cake=t 1275    conf:(0.84)
 8. total=high 1679 ==> bread and cake=t 1413    conf:(0.84)
 9. biscuits=t milk-cream=t 1767 ==> bread and cake=t 1485    conf:(0.84)
10. baking needs=t milk-cream=t vegetables=t 1392 ==> bread and cake=t 1169    conf:(0.84)
11. biscuits=t fruit=t 1837 ==> bread and cake=t 1541    conf:(0.84)
12. milk-cream=t margarine=t 1549 ==> bread and cake=t 1299    conf:(0.84)
13. biscuits=t margarine=t 1493 ==> bread and cake=t 1251    conf:(0.84)
14. milk-cream=t fruit=t vegetables=t 1571 ==> bread and cake=t 1311    conf:(0.83)
15. biscuits=t frozen foods=t 1810 ==> bread and cake=t 1510    conf:(0.83)
16. margarine=t vegetables=t 1587 ==> bread and cake=t 1322    conf:(0.83)
17. biscuits=t tissues-paper prd=t 1453 ==> bread and cake=t 1209    conf:(0.83)
18. frozen foods=t fruit=t 1861 ==> bread and cake=t 1548    conf:(0.83)
19. frozen foods=t milk-cream=t 1826 ==> bread and cake=t 1516    conf:(0.83)
20. party snack foods=t fruit=t 1592 ==> bread and cake=t 1321    conf:(0.83)
21. tissues-paper prd=t vegetables=t 1559 ==> bread and cake=t 1293    conf:(0.83)
22. baking needs=t milk-cream=t 1907 ==> bread and cake=t 1580    conf:(0.83)
23. tissues-paper prd=t fruit=t 1567 ==> bread and cake=t 1297    conf:(0.83)
24. party snack foods=t milk-cream=t 1541 ==> bread and cake=t 1275    conf:(0.83)
25. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    conf:(0.83)
26. baking needs=t margarine=t 1645 ==> bread and cake=t 1358    conf:(0.83)
27. baking needs=t biscuits=t 1764 ==> bread and cake=t 1456    conf:(0.83)
28. frozen foods=t tissues-paper prd=t 1505 ==> bread and cake=t 1239    conf:(0.82)
29. baking needs=t fruit=t 1900 ==> bread and cake=t 1564    conf:(0.82)
30. sauces-gravy-pkle=t fruit=t 1490 ==> bread and cake=t 1226    conf:(0.82)
31. baking needs=t tissues-paper prd=t 1573 ==> bread and cake=t 1294    conf:(0.82)
32. frozen foods=t vegetables=t 1882 ==> bread and cake=t 1548    conf:(0.82)
33. biscuits=t party snack foods=t 1592 ==> bread and cake=t 1306    conf:(0.82)
34. baking needs=t party snack foods=t 1530 ==> bread and cake=t 1254    conf:(0.82)
35. milk-cream=t vegetables=t 2025 ==> bread and cake=t 1658    conf:(0.82)
36. sauces-gravy-pkle=t milk-cream=t 1487 ==> bread and cake=t 1216    conf:(0.82)
37. bread and cake=t biscuits=t vegetables=t 1487 ==> fruit=t 1216    conf:(0.82)
38. party snack foods=t vegetables=t 1559 ==> bread and cake=t 1273    conf:(0.82)
39. juice-sat-cord-ms=t biscuits=t 1542 ==> bread and cake=t 1259    conf:(0.82)
40. frozen foods=t margarine=t 1531 ==> bread and cake=t 1249    conf:(0.82)
```

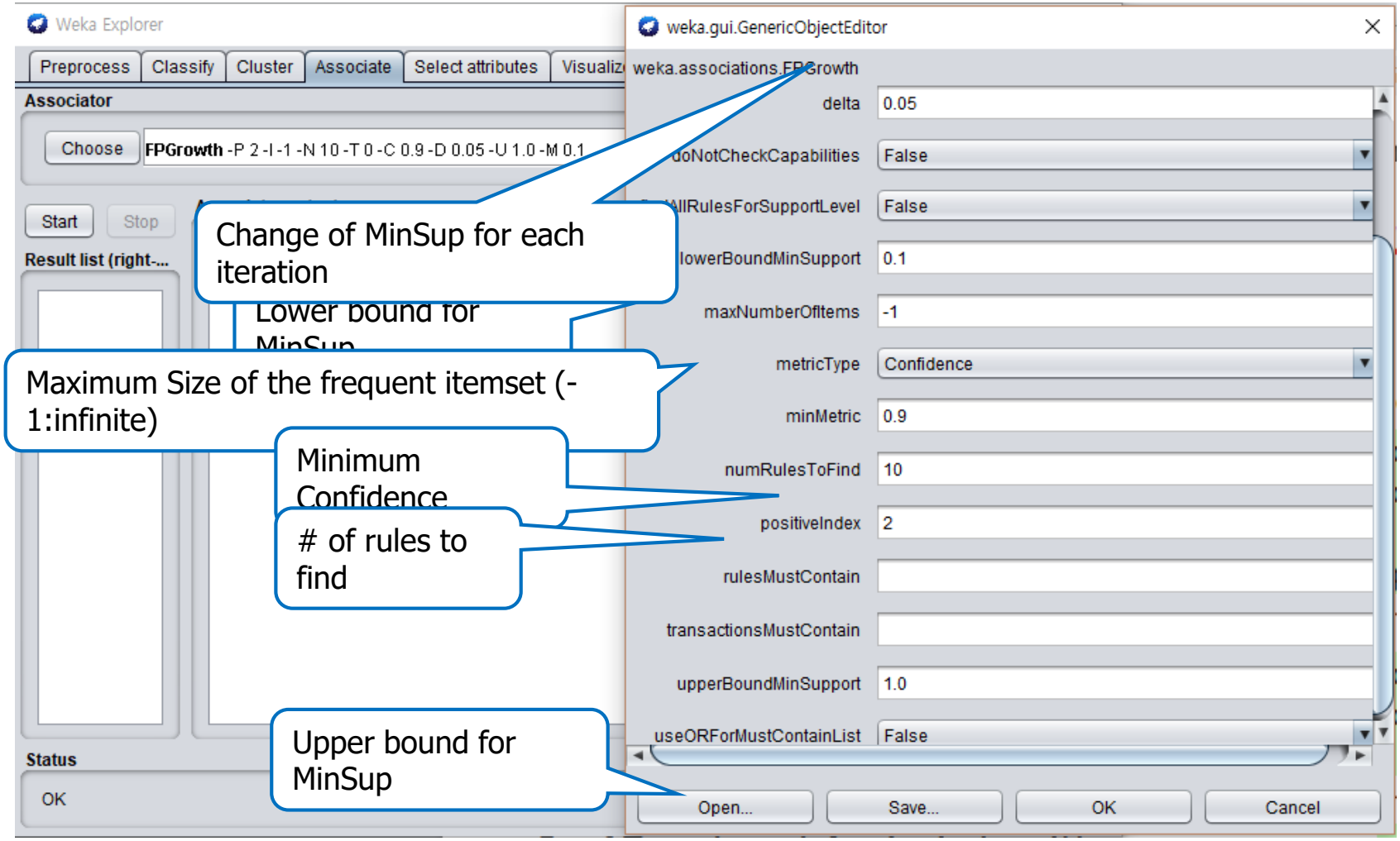
Association Rule - FPGrowth Algorithm

- Select algorithm 'FPGrowth'
- Run the algorithm

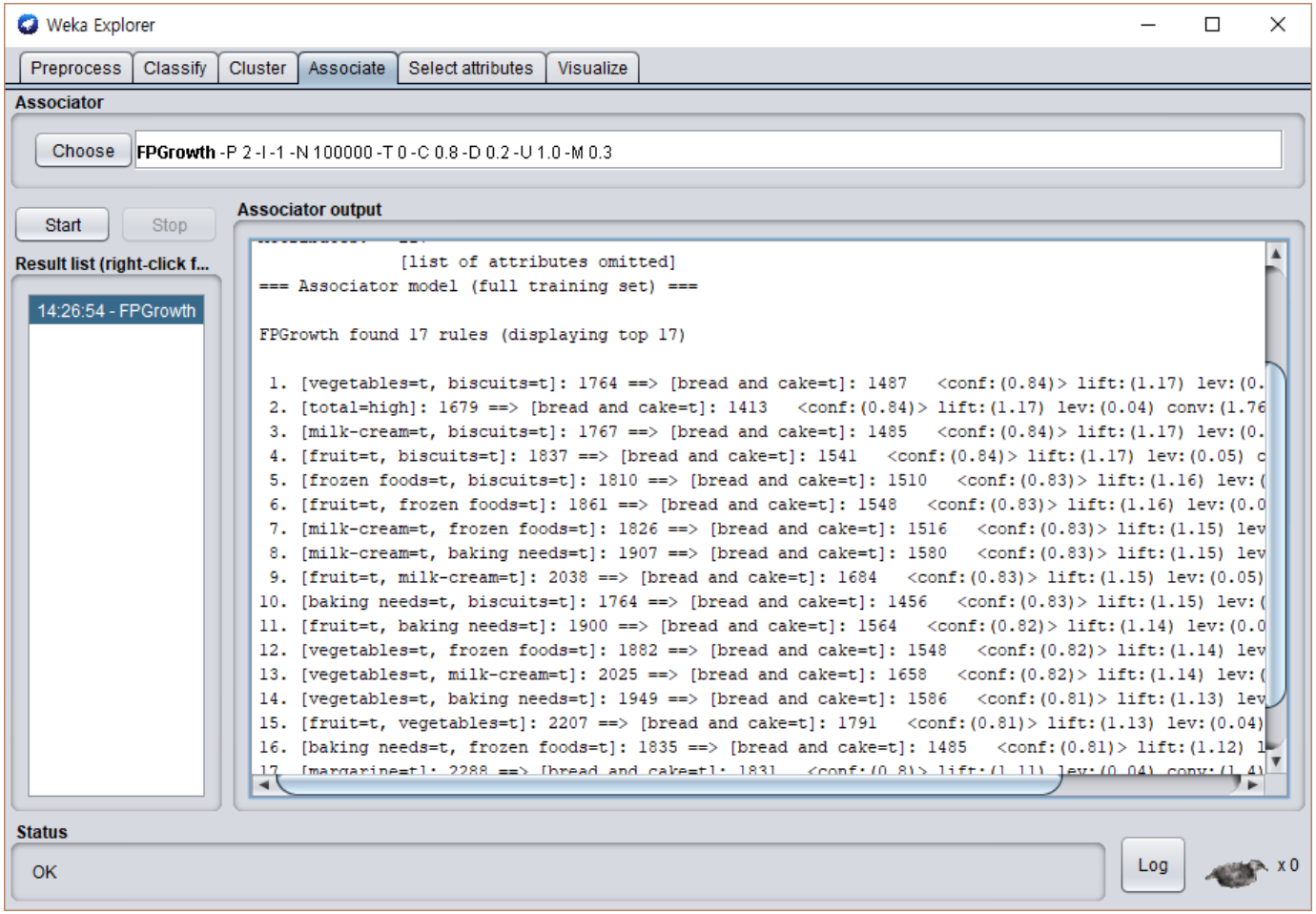


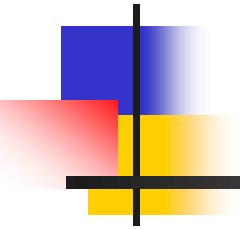


Association Rule - FPGrowth Algorithm

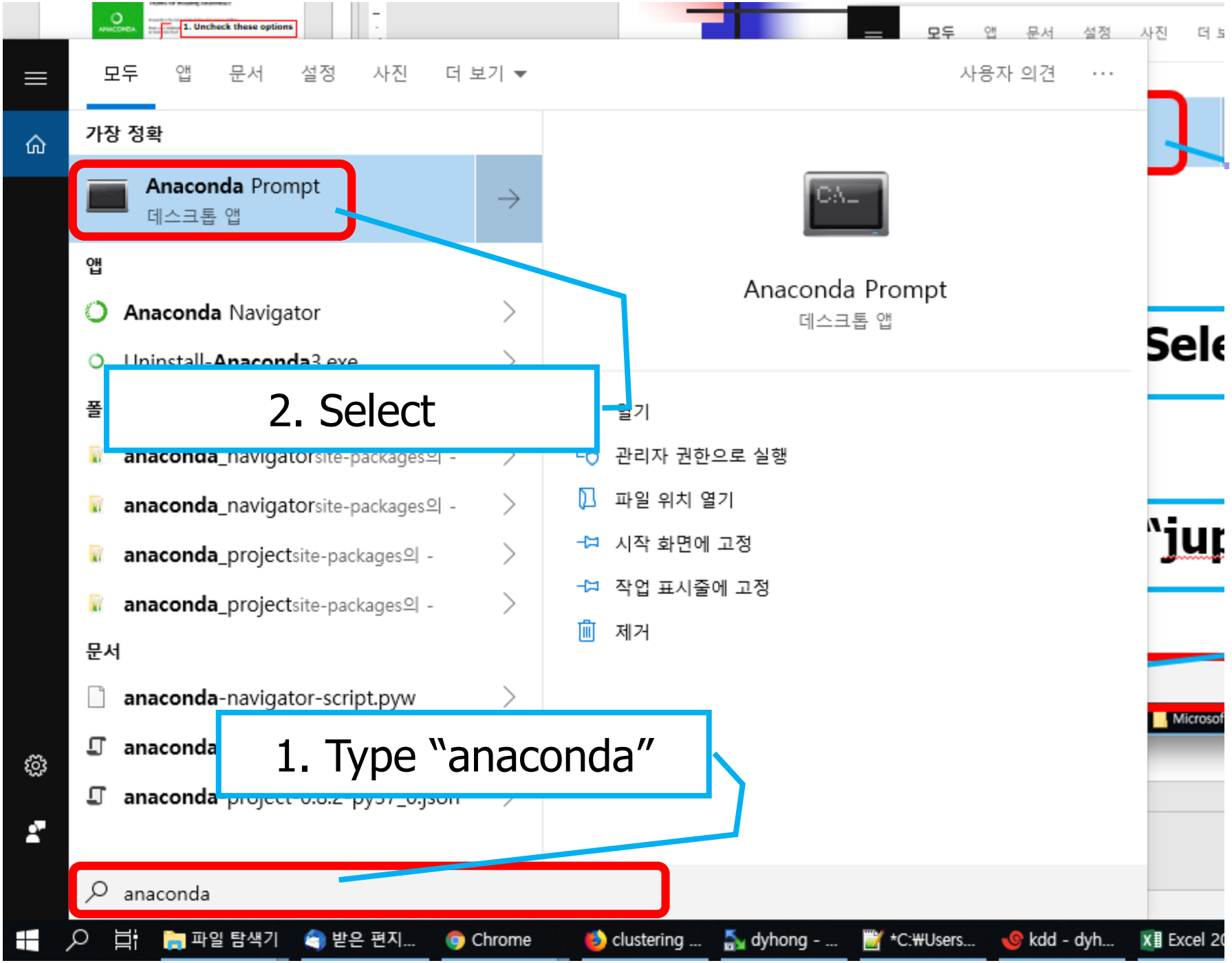


Association Rule - FPGrowth Algorithm





Python - Association Rule


1. Uncheck these options
모두
앱
문서
설정
사진
더 보기
사용자 의견
가장 정확
Anaconda Prompt
데스크톱 앱
앱
Anaconda Navigator
Uninstall-Anaconda3.exe
anaconda_navigatorsite-packages의 -
anaconda_navigatorsite-packages의 -
anaconda_projectsite-packages의 -
anaconda_projectsite-packages의 -
문서
anaconda-navigator-script.pyw
anaconda
anaconda
Anaconda Prompt
데스크톱 앱
열기
관리자 권한으로 실행
파일 위치 열기
시작 화면에 고정
작업 표시줄에 고정
제거
2. Select
1. Type “anaconda”
anaconda
파일 탐색기
받은 편지...
Chrome
clustering ...
dyhong - ...
*C:₩Users...
kdd - dyh...
Excel 2

Install pyfpgrowth with pip

- In anaconda prompt, type

  pip install pyfpgrowth

and press enter

```
(C:\Users\Ahn\Anaconda3) C:\Users\Ahn\pythonWorkspace\weka>pip install pyfpgrowth
Collecting pyfpgrowth
Installing collected packages: pyfpgrowth
Successfully installed pyfpgrowth-1.0
```

Import Libraries

- pyfpgrowth is an open source package that supports frequent pattern mining and rule mining

In [1]:
```
import pandas as pd
import numpy as np
import pyfpgrowth
```

Load Data

```
In [2]: df = pd.read_csv("supermarket.csv")
        df
```

Out[2]:

	?	?.1	?.2	?.3	?.4	?.5	?.6	?.7	?.8	?.9	...	?.
0	t	?	?	?	?	?	?	?	?	?	...	?
1	?	?	?	?	?	?	?	?	?	?	...	?
2	t	?	?	?	?	?	?	?	?	?	...	?

- Each row is a transaction
- Each column is an item
- 't' if the transaction has the item, '?' if not

name of items are in supermarket_attributes.txt

Preprocessing

- We want the data to have the form :

[[1, 3, 6], [3, 8], … [4, 7, 8, 9]]

  - A 2d-list where each row is the list of items in a transaction

Preprocessing

In [3]:
```
X = df.values
transaction = []
for row in X:
    temp = []
    for i in range(len(row)-1):
        if row[i] == 't':
            temp.append(i)
    transaction.append(temp)
print(len(transaction))
```

```
4626
```

Preprocessing

In [3]:
```
X = df.values
transaction = []
for row in X:
    temp = []
    for i in range(len(row)-1):
        if row[i] == 't':
            temp.append(i)
    transaction.append(temp)
print(len(transaction))
```

If an item is in the transaction, append its index to temp

4626 # of transactions

Preprocessing

In [3]:
```
X = df.values
transaction = []
for row in X:
    temp = []
    for i in range(len(row)-1):
        if row[i] == 't':
            temp.append(i)
    transaction.append(temp)
print(len(transaction))
```

4626 # of transactions

The last column is not an item;
it is the price is higher or lower than 100 dollars, so exclude it

Preprocessing

In [3]:
```
X = df.values
transaction = []
for row in X:
    temp = []
    for i in range(len(row)-1):
        if row[i] == 't':
            temp.append(i)
    transaction.append(temp)
print(len(transaction))
```

Append the list of item indices to the 2d-list

4626 # of transactions

Parameters

In [4]:
```
min_sup = 0.3
confidence = 0.75
```

Association Rule - FPGrowth Algorithm

In [5]:
```
patterns = pyfpgrowth.find_frequent_patterns(transaction,
                                             int(min_sup*len(transaction)))
patterns
```

Out[5]:
```
{(12,): 3329,
 (12, 13): 2190,
 (12, 13, 17): 1455,
 (12, 13, 31): 1484,
 (12, 13, 60): 1579,
 (12, 13, 82): 1563,
 (12, 13, 85): 1585,
 (12, 15): 1868,
 (12, 17): 2082,
```

Association Rule - FPGrowth Algorithm

In [5]:
```
patterns = pyfpgrowth.find_frequent_patterns(transaction,
                                             int(min_sup*len(transaction)))

patterns
```

This function requires the support as the number of occurrences

Out[5]:
```
{(12,): 3329,
 (12, 13): 2190,
 (12, 13, 17): 1455,
 (12, 13, 31): 1484,
 (12, 13, 60): 1579,
 (12, 13, 82): 1563,
 (12, 13, 85): 1585,
 (12, 15): 1868,
 (12, 17): 2082,
```

Association Rule - FPGrowth Algorithm

```
In [5]: patterns = pyfpgrowth.find_frequent_patterns(transaction,
                                                     int(min_sup*len(transaction)))
        patterns
```

```
Out[5]: {(12,): 3329,
         (12, 13): 2190,
         (12, 13, 17): 1455,
         (12, 13, 31): 1484,
         (12, 13, 60): 1579,
         (12, 13, 82): 1563,
         (12, 13, 85): 1585,
         (12, 15): 1868,
         (12, 17): 2082,
               .
               .
               .
```

The result is saved as a dictionary

ex)

```
patterns[(12,)]
```

```
3329
```

```
patterns[(12,15)]
```

```
1868
```

There must be a comma in python tuple object

Association Rule - FPGrowth Algorithm

```
rules = pyfpgrowth.generate_association_rules(patterns,
                                              confidence)
rules
```

```
{(12, 82): ((85,), 0.770223752151463),
 (12, 85): ((82,), 0.7792773182411842),
 (13, 17): ((12,), 0.8252977878615996),
 (13, 31): ((12,), 0.809160305343511 5),
 (13, 85): ((82,), 0.7638603696098563),
 (17, 31): ((12,), 0.8341625207296849),
 (17, 60): ((12,), 0.840317100792752),
```

.

.

.

Association Rule - FPGrowth Algorithm

```
rules = pyfpgrowth.generate_association_rules(patterns,
                                              confidence)
rules
```

```
{(12, 82): ((85,), 0.770223752151463),
 (12, 85): ((82,), 0.7792773182411842),
 (13, 17): ((12,), 0.8252977878615996),
 (13, 31): ((12,), 0.8091603053435115),
 (13, 85): ((82,), 0.763860369609856 3),
 (17, 31): ((12,), 0.834162520729 6849),
 (17, 60): ((12,), 0.84031710079 2752),
```

.
.
.

A rule (12, 82) -> (85)
with confidence 0.77

Association Rule - FPGrowth Algorithm

```
rules = pyfpgrowth.generate_association_rules(patterns,
                                              confidence)
rules
```

```
{(12, 82): ((85,), 0.770223752151463),
 (12, 85): ((82,), 0.7792773182411842),
 (13, 17): ((12,), 0.8252977878615996),
 (13, 31): ((12,), 0.8091603053435115),
 (13, 85): ((82,), 0.7638603696098563),
 (17, 31): ((12,), 0.834162520729684 9),
 (17, 60): ((12,), 0.840317100792752),
```

.
.
.

We want top-k rules with highest confidence! How?

Sorting Dictionary

```
rules.items()
```

```
dict_items([((17, 82), ((12,), 0.8387799564270153)), ((60, 82), ((85,), 0.770741286205
2038)), ((21,), ((12,), 0.773899033297529 5)), ((13, 31), ((12,), 0.8091603053435115)),
((17, 60), ((12,), 0.840317100792752)), ((58,), ((12,), 0.793929712460063 9)), ((136,),
((12,), 0.7923408845738943)), ((17, 31), ((12,), 0.8341625207296849)), ((12, 82), ((8
5,), 0.770223752151463)), ((13, 17), ((12,), 0.8252977878615996)), ((37,), ((12,), 0.7
980717728976968)), ((82, 85), ((12,), 0.8114233907524931)), ((44,), ((12,), 0.75688559
3220339)), ((31, 60), ((12,), 0.8301369863013699)), ((82,), ((12,), 0.784869976359338
1)), ((12, 85), ((82,), 0.7792773182411842)), ((31, 85), ((82,), 0.7708665603402446)),
((13, 85), ((82,), 0.7638603696098563))])
```

- rules.items() returns the (key, value) pairs

Sorting Dictionary

```
rules_sorted = sorted(rules.items(),
                      key=lambda t : t[1][1],
                      reverse = True)
rules_sorted
```

```
[((17, 60), ((12,), 0.840317100792752)),
 ((17, 82), ((12,), 0.8387799564270153)),
 ((17, 31), ((12,), 0.834162520729684 9)),
 ((31, 60), ((12,), 0.8301369863013699)),
 ((13, 17), ((12,), 0.8252977878615996)),
```

Sorting Dictionary

```
rules_sorted = sorted(rules.items(),
                      key=lambda t : t[1][1],
                      reverse = True)
rules_sorted
```

lambda t means that t is the each row of input list

```
[((17, 60), ((12,), 0.840317100792752)),
 ((17, 82), ((12,), 0.838779564270153)),
 ((17, 31), ((12,), 0.834162520729684 9)),
 ((31, 60), ((12,), 0.830136986301369 9)),
 ((13, 17), ((12,), 0.825297787861599 6)),
```

Sorting Dictionary

```
rules_sorted = sorted(rules.items(),
                      key=lambda t : t[1][1],
                      reverse = True)
rules_sorted
```

We will sort the input list with t[1][1] as the key

```
[((17, 60), ((12,), 0.840317100792752)),
 ((17, 82), ((12,), 0.838779564270153)),
 ((17, 31), ((12,), 0.834162520729684 9)),
 ((31, 60), ((12,), 0.830136986301369 9)),
 ((13, 17), ((12,), 0.825297787861599 6)),
```

Sorting Dictionary

```
rules_sorted = sorted(rules.items(),
                      key=lambda t : t[1][1],
                      reverse = True)
rules_sorted
```

```
[((17, 60), ((12,), 0.840317100792752)),
 ((17, 82), ((12,), 0.838779564270153)),
 ((17, 31), ((12,), 0.834162520729684 9)),
 ((31, 60), ((12,), 0.830136986301369 9)),
 ((13, 17), ((12,), 0.825297787861599 6)),
```

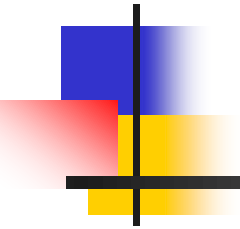
We will sort the input list with descending order

Top-k Rules

```
k = 10
topk_rules = []
for i in range(k):
    temp = []
    temp.append(rules_sorted[i][0])
    temp.append(rules_sorted[i][1][0])
    temp.append(rules_sorted[i][1][1])
    topk_rules.append(temp)
topk_rules
```

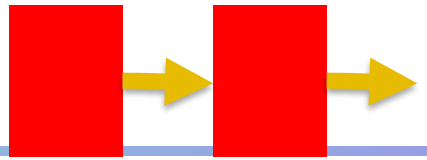
```
[[(17, 60), (12,), 0.840317100792752],
 [(17, 82), (12,), 0.8387799564270153],
 [(17, 31), (12,), 0.8341625207296849],
 [(31, 60), (12,), 0.8301369863013699],
 [(13, 17), (12,), 0.8252977878615996],
 [(82, 85), (12,), 0.8114233907524931],
 [(13, 31), (12,), 0.8091603053435115],
 [(37,), (12,), 0.7980717728976968],
 [(58,), (12,), 0.7939297124600639],
 [(136,), (12,), 0.7923408845738943]]
```

Finally append top-k rules to the result list

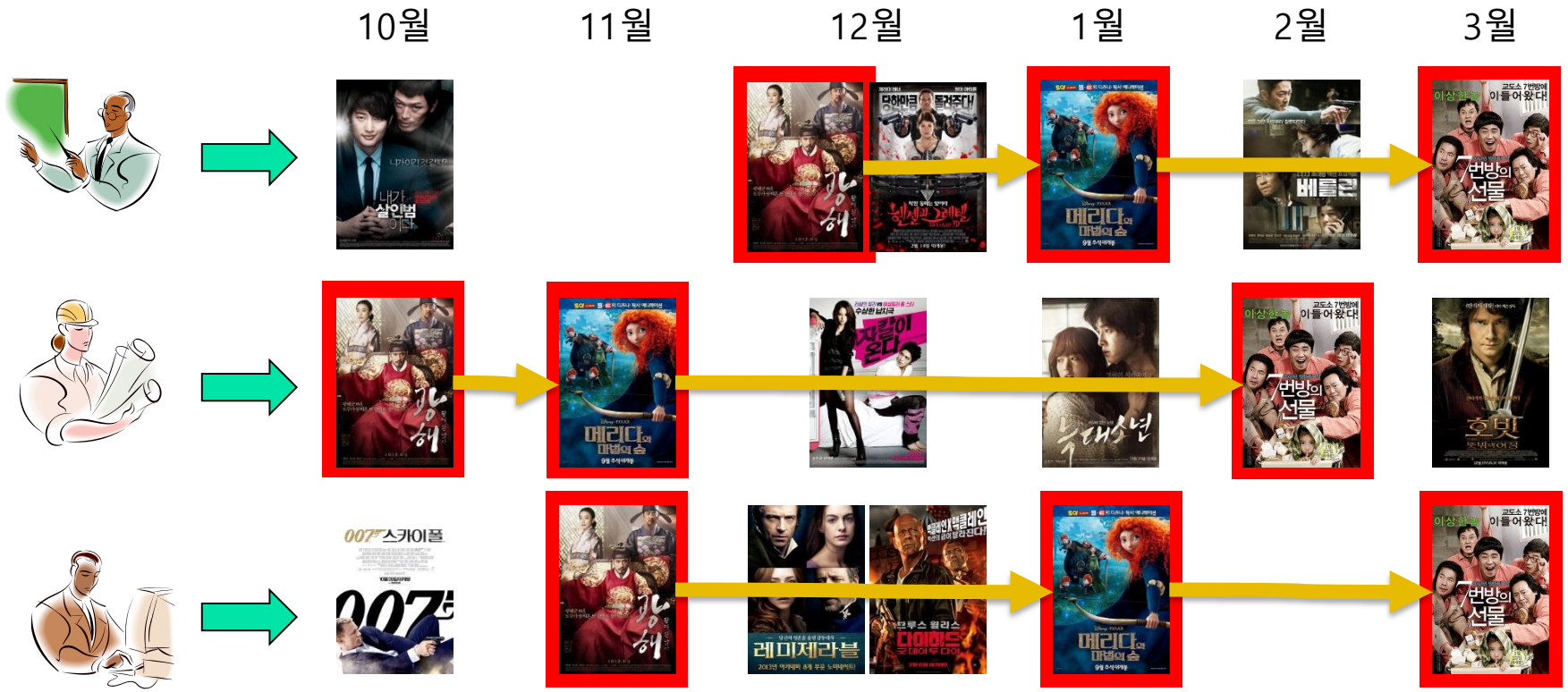


Sequential Pattern Mining

영화 감상 기록



빈번한 패턴



● ● ●

Sequential Patterns

- Sequential Pattern Mining
  - 각 원소 사이에 순서가 있는 시퀀스 데이터 베이스에서 순서를 고려하여 빈번하게 나타나는 패턴을 찾아주는 기술

고객 번호	구매 기록
1	({늑대 소년}, {베를린, 레미제라블})
2	({늑대 소년}, {베를린, 공동경비 구역})
3	({러브레터}, {광해, 호빗}, {스카이폴})
4	({늑대 소년}, {레미제라블}

- ({늑대 소년}) → ({레미제라블})
  - 네 명 중에서 두 명이 갖고 있는 패턴이므로 지지도는 50%
  - 만일 최소 지지도를 50%라고 했다면 아래 룰이 찾아짐
    - ({늑대 소년}) → ({레미제라블})
    - ({늑대 소년}) → ({베를린})

Sequential Patterns

- 의료 데이터마이닝
  - 환자의 과거 테스트, 진단, 치료, 처방등에 관한 히스토리 데이터로부터 당뇨병에 걸리는 환자의 패턴을 이용하여 앞으로 당뇨병 또는 치매에 걸릴 확률이 높은 환자를 알아냄
  - 히스토리 정보를 이용하여 부작용이 발생할 환자를 예측함
  - 입원 환자, 신생아, 마취 사망 가능성 예측 : 기존의 입원 환자/신생아 사례(항생제 등)로부터 패턴을 만들고, 이후 입원 환자/신생아 경우에 예상하는 자료에 사용
    - 연관규칙, 순차패턴 모두 적용 가능

Sequential Patterns

- 사고 발생이나 부품 고장 전 센서를 통해 수집된 데이터를 이용해 순차 패턴 마이닝 으로 원인을 파악
  - 운전자 부주의
    - 사고 예방에 이용
    - 예) 미끄러운 길에서 속도 제한 등
  - 차량 결함
    - 리콜 결정에 이용
    - 신차개발에 반영
    - 예) 빗길에서 배터리 전압이 낮아질 경우 자동차의 최고속도를 제한

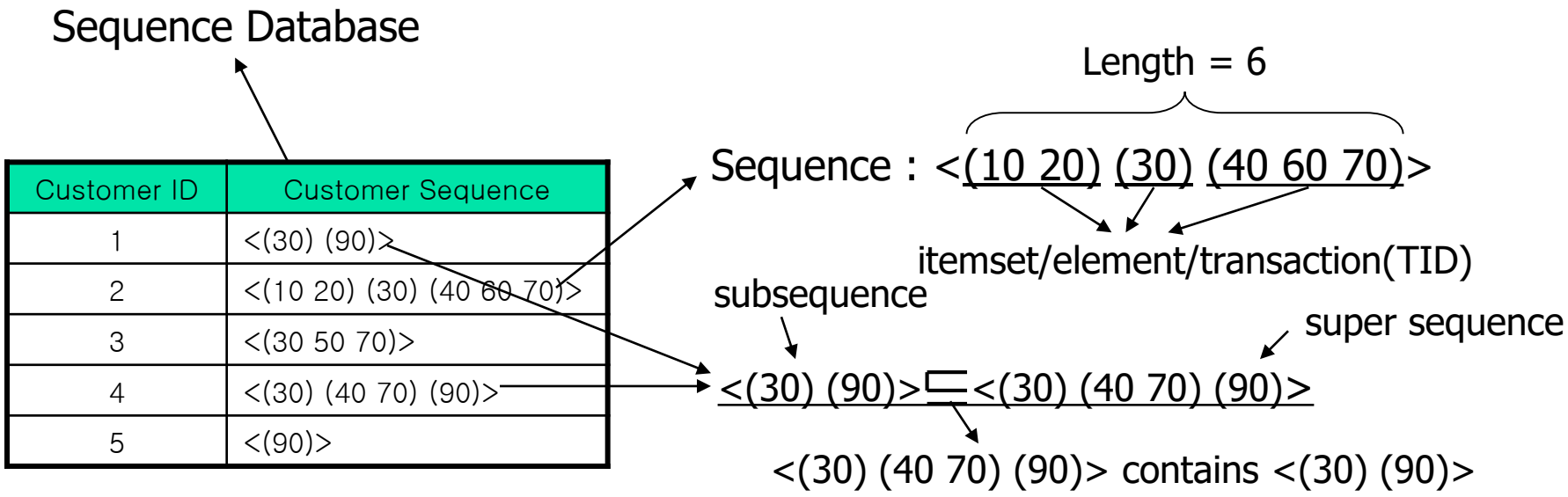
Sequential Patterns

- 고객 구매 패턴
  - 상품 구매 히스토리 데이터를 이용하여 고객의 지금까지 구매한 상품을 보고 다른 상품을 추천함
- 위치 정보 마이닝
  - 사람들 움직임 데이터를 이용, 고객이 오늘 5시에 어디에 있을지 예측
  - 특정한 범죄를 일으킨 사람의 행동반경 패턴을 이용하여 최근 발생한 범죄자의 위치를 추적함
  - 사용자의 운전 습관, Traffic 패턴 기반으로 자동차 Navigation 길 안내

Sequential Patterns

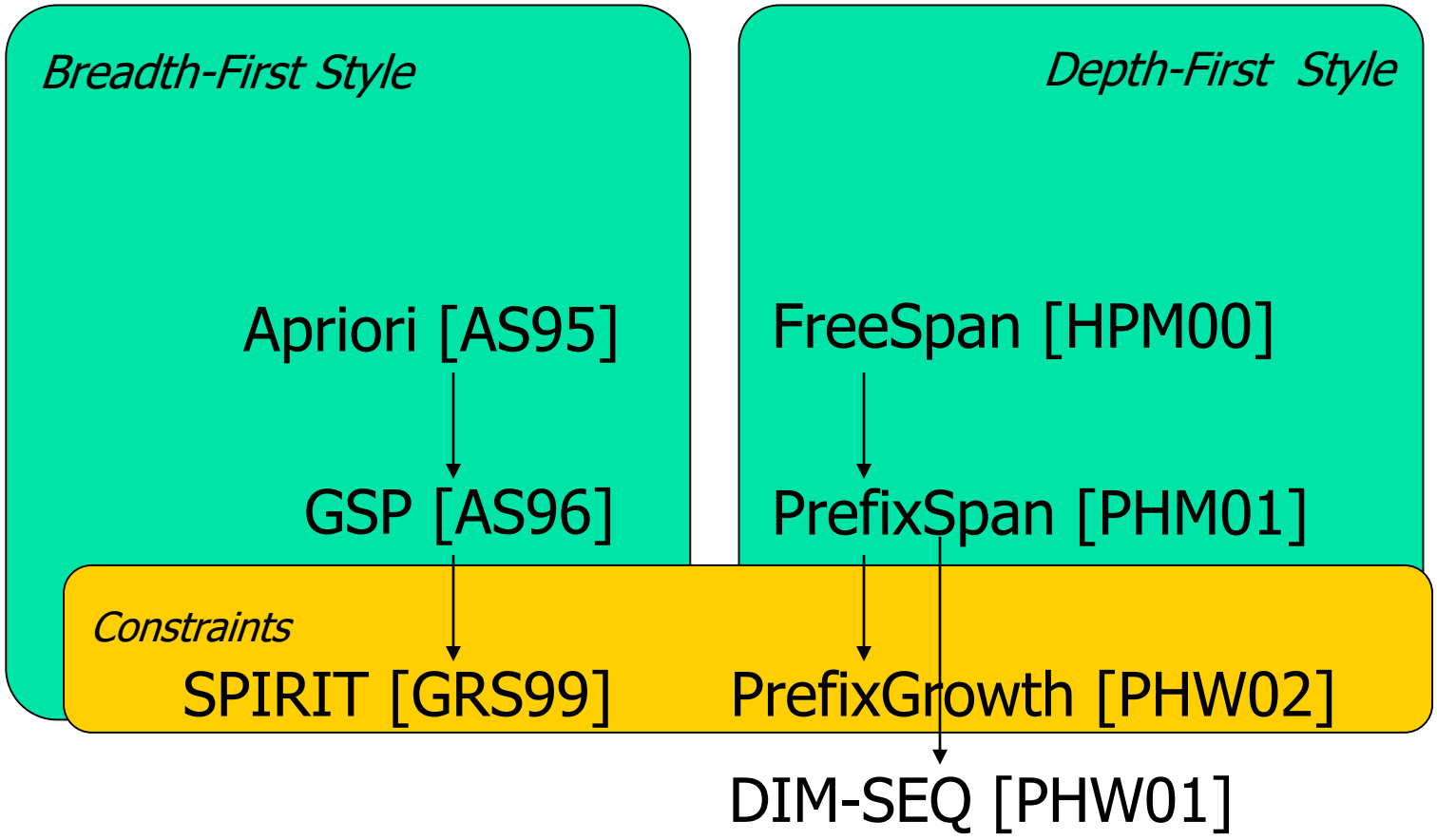
- Given:
  - A sequence of customer transactions
  - Each transaction is a set of items
- Find all maximal sequential patterns supported by more than a user-specified percentage of customers
- Sample Applications
  - E-Commerce
  - Mail order companies
  - Web access log analysis
  - DNA sequences

Sequential Patterns



Sequential patterns with support > 25%
<(30) (90)>
<(30) (40 70)>

Sequential Patterns



GSP: Apriori Style Algorithm

- Finding Length-1 Sequential Patterns
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

min_sup =2

Seq. ID	Sequence
10	<(bd)(c)(b)(ac)>
20	<(bf)(ce)(b)(fg)>
30	<(ah)(bf)(a)(b)(f)>
40	<(be)(ce)(d)>
50	<(a)(bd)(b)(c)(b)(ade)>

Cand	Sup
<a>	**3**
<b>	**5**
<c>	**4**
<d>	**3**
<e>	**3**
<f>	**2**
~~<g>~~	1
~~<h>~~	1

GSP: Generating Length-2 Candidates

51 length-2 Candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<(a)(a)>	<(a)(b)>	<(a)(c)>	<(a)(d)>	<(a)(e)>	<(a)(f)>
<b>	<(b)(a)>	<(b)(b)>	<(b)(c)>	<(b)(d)>	<(b)(e)>	<(b)(f)>
<c>	<(c)(a)>	<(c)(b)>	<(c)(c)>	<(c)(d)>	<(c)(e)>	<(c)(f)>
<d>	<(d)(a)>	<(d)(b)>	<(d)(c)>	<(d)(d)>	<(d)(e)>	<(d)(f)>
<e>	<(e)(a)>	<(e)(b)>	<(e)(c)>	<(e)(d)>	<(e)(e)>	<(e)(f)>
<f>	<(f)(a)>	<(f)(b)>	<(f)(c)>	<(f)(d)>	<(f)(e)>	<(f)(f)>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Without Apriori property, 8*8+8*7/2=92 candidates

Apriori prunes 44.57% candidates

Candidate Generate-and-test: Drawbacks

- A huge set of candidate sequences generated.
  - Especially 2-item candidate sequence.
- Multiple Scans of database needed.
  - The length of each candidate grows by one at each database scan.
- Inefficient for mining long sequential patterns.
  - A long pattern grow up from short patterns
  - The number of short patterns is exponential to the length of mined patterns.

Prefix and Suffix (Projection)

- <(a)>, <(a)(a)>, <(a)(ab)> and <(a)(abc)> are the *prefixes* of sequence <(a)(abc)(ac)(d)(cf)>
- Given sequence <(a)(abc)(ac)(d)(cf)>

Prefix	*Suffix* (Prefix-Based *Projection*)
<(a)>	<(abc)(ac)(d)(cf)>
<(a)(a)>	<(_bc)(ac)(d)(cf)>
<(a)(b)>	<(_c)(ac)(d)(cf)>

PrefixSpan

SID	Sequence Database
10	<(a b) (c)>
20	<(b) (a c)>
30	<(d)(c) (a b)>

Given min_support = 50%

1-frequent sequence
<a>, <b>, <c>

Having prefix <(a)> F

SID	Projected Database
10	<(_b)(c)>
20	(_c)
30	<(_b)>

1-frequent sequence
<(_b)>

Having prefix <(ab)> F

SID	Projected Database
10	<c>

1-frequent sequence
∅

Having prefix <(b)> F

SID	Projected Database
10	<(c)>
20	<(a c)>

1-frequent sequence
<(c)>

Having prefix <(b)(c)> F

SID	Projected Database

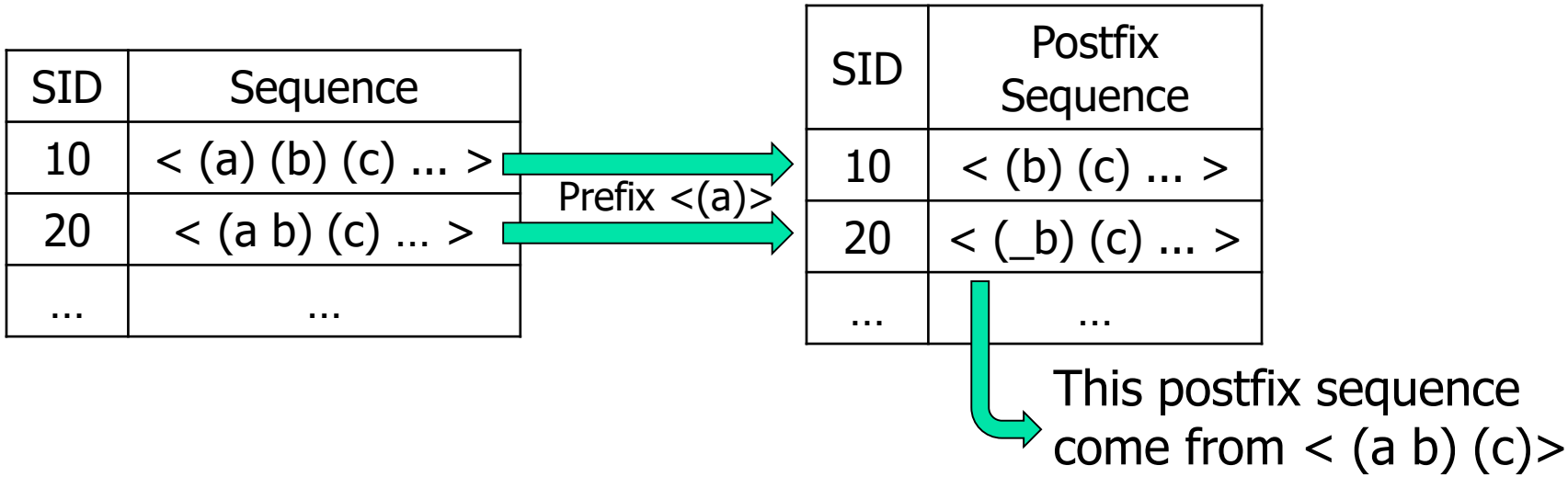
1-frequent sequence
∅

Having prefix <(c)> F

SID	Projected Database
30	<(a b)>

1-frequent sequence
∅

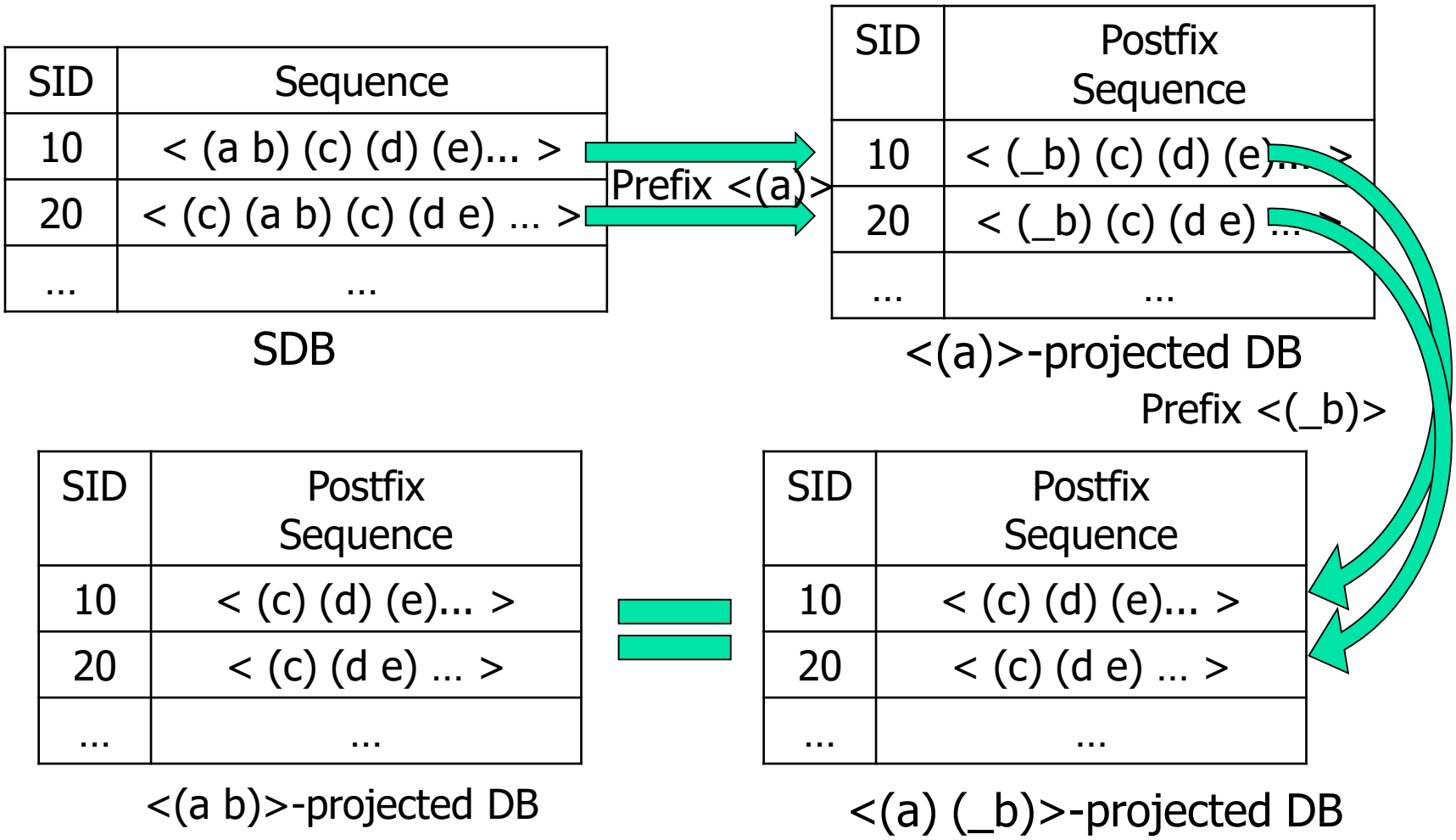
An Example of PrefixSpan



SDB

<(a)>-projected DB

An Example of PrefixSpan



An Example of PrefixSpan

SID	Sequence
10	< (a) (b) (c) (d) (e) >
20	< (a) (c) (d) (b) (c) (e) >
30	< (b) (a) (c) (d) >
40	< (b) (c) (d) (e) >
50	< (a) (b) (e) >
…	…

SDB

SID	Postfix Sequence
10	< (b) (c) (d) (e) >
20	< (c) (d) (b) (c) (e) >
30	< (c) (d) >
50	< (b) (e) >
…	…

<(a)>-projected DB

Support(<(b)>) in <(a)>-projected DB ?

An Example of PrefixSpan

SID	Sequence
10	< (a) (b) (c) (d) (e) >
20	< (a) (c) (d) (b) (c) (e) >
30	< (b) (a) (c) (d) >
40	< (b) (c) (d) (e) >
50	< (a) (b) (e) >
…	…

SDB

SID	Postfix Sequence
10	< (b) (c) (d) (e) >
20	< (c) (d) (b) (c) (e) >
30	< (c) (d) >
50	< (b) (e) >
…	…

<(a)>-projected DB

Support(<(b)>) in <(a)>-projected DB = 3

An Example of PrefixSpan

SID	Sequence
10	< (a) (b) (c) (d) (e) >
20	< (a) (c) (d) (b) (c) (e) >
30	< (b) (a) (c) (d) >
40	< (b) (c) (d) (e) >
50	< (a) (b) (e) >
…	…

SDB

SID	Postfix Sequence
10	< (b) (c) (d) (e) >
20	< (c) (d) (b) (c) (e) >
30	< (c) (d) >
50	< (b) (e) >
…	…

<(a)>-projected DB

Support(<(b)>) in <(a)>-projected DB = 3

This means that Support(<(a) (b)>) in SDB = 3

An Example of PrefixSpan

SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c) (e) (a) >
30	< (b) (a) (d) >
40	< (b) (c d) (e) >
50	< (d) (f) (g) (a c) (e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c) (e) (a) >
30	< (d) >
50	< (_c) (e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ?

An Example of PrefixSpan

SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c) (e) (a) >
30	< (b) (a) (d) >
40	< (b) (c d) (e) >
50	< (d) (f) (g) (a c) (e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c) (e) (a) >
30	< (d) >
50	< (_c) (e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ? = 3

An Example of PrefixSpan

Prefix <(a)>

SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c) (e) (a) >
30	< (b) (a) (d) >
40	< (b) (c d) (e) >
50	< (d) (f) (g) (a c) (e) (a) >
…	…

SDB

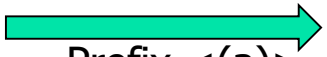
SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c) (e) (a) >
30	< (d) >
50	< (_c) (e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB = 3

This means that Support(<(a c)>) in SDB = 3

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

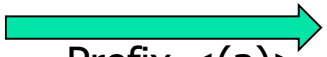
SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ?

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

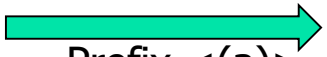
SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ? = 2

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ? = 2

This means that Support(<(a c)>) in SDB = 2

An Example of PrefixSpan

SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

SDB

We miss this transaction!!!

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ? = 2

This means that Support(<(a c)>) in SDB = 2

An Example of PrefixSpan

SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

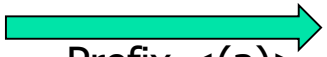
<(a)>-projected DB

We also count this as _c

Support(<(_c)>) in <(a)>-projected DB ? = 2

This means that Support(<(a c)>) in SDB = 2

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a c) (d) (e) >
20	< (a c) (d) (b) (c e) (a) >
30	< (a) (b) (a c) >
40	< (a) (c) (c d) (e) >
50	< (d) (f) (g) (a) (c e) (a) >
…	…

SDB

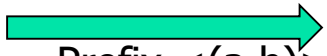
SID	Postfix Sequence
10	< (_c) (d) (e) >
20	< (_c) (d) (b) (c e) (a) >
30	< (b) (a c) >
40	< (c) (c d) (e)>
50	< (c e) (a) >
…	…

<(a)>-projected DB

Support(<(_c)>) in <(a)>-projected DB ? = 3

This means that Support(<(a c)>) in SDB = 3

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

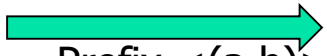
SDB

SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

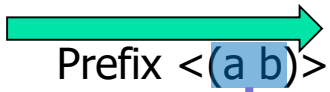
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d?

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

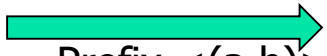
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d? YES

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

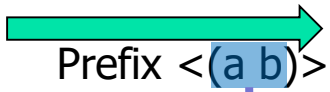
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d?

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d? NO

An Example of PrefixSpan

Prefix <(a b)>

SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

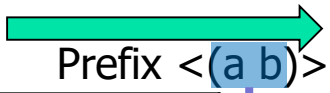
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d?

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

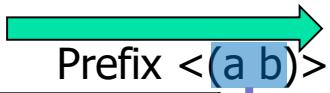
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d? YES

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

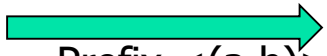
SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB?

Do we count this as _d? YES

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

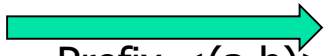
SDB

SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB = 3

An Example of PrefixSpan



SID	Sequence
10	< (b) (e) (a b d) (d) (e) >
20	< (a b) (c) (d) (b) (a b d) (a) >
30	< (a b) (a) (b) (a d) >
40	< (a) (c) (c d) (e) >
50	< (d) (a b c d) (f) (g) (a) (c e) (a) >
…	…

SDB

SID	Postfix Sequence
10	< (_d) (d) (e) >
20	< (c) (d) (b) (a b d) (a) >
30	< (a) (b) (a d) >
50	< (_c d) (f) (g) (a) (c e) (a) >
…	…

<(a b)>-projected DB

Support(<(_d)>) in <(a b)>-projected DB = 3

This means that Support(<(a b d)>) in SDB = 3

Mining Sequential Patterns by Prefix Projections

- Step 1: find length-1 sequential patterns
  - <(a)>, <(b)>, <(c)>, <(d)>, <(e)>, <(f)>
- Step 2: divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:
  - The ones having prefix <(a)>;
  - The ones having prefix <(b)>;
  - …
  - The ones having prefix <(f)>

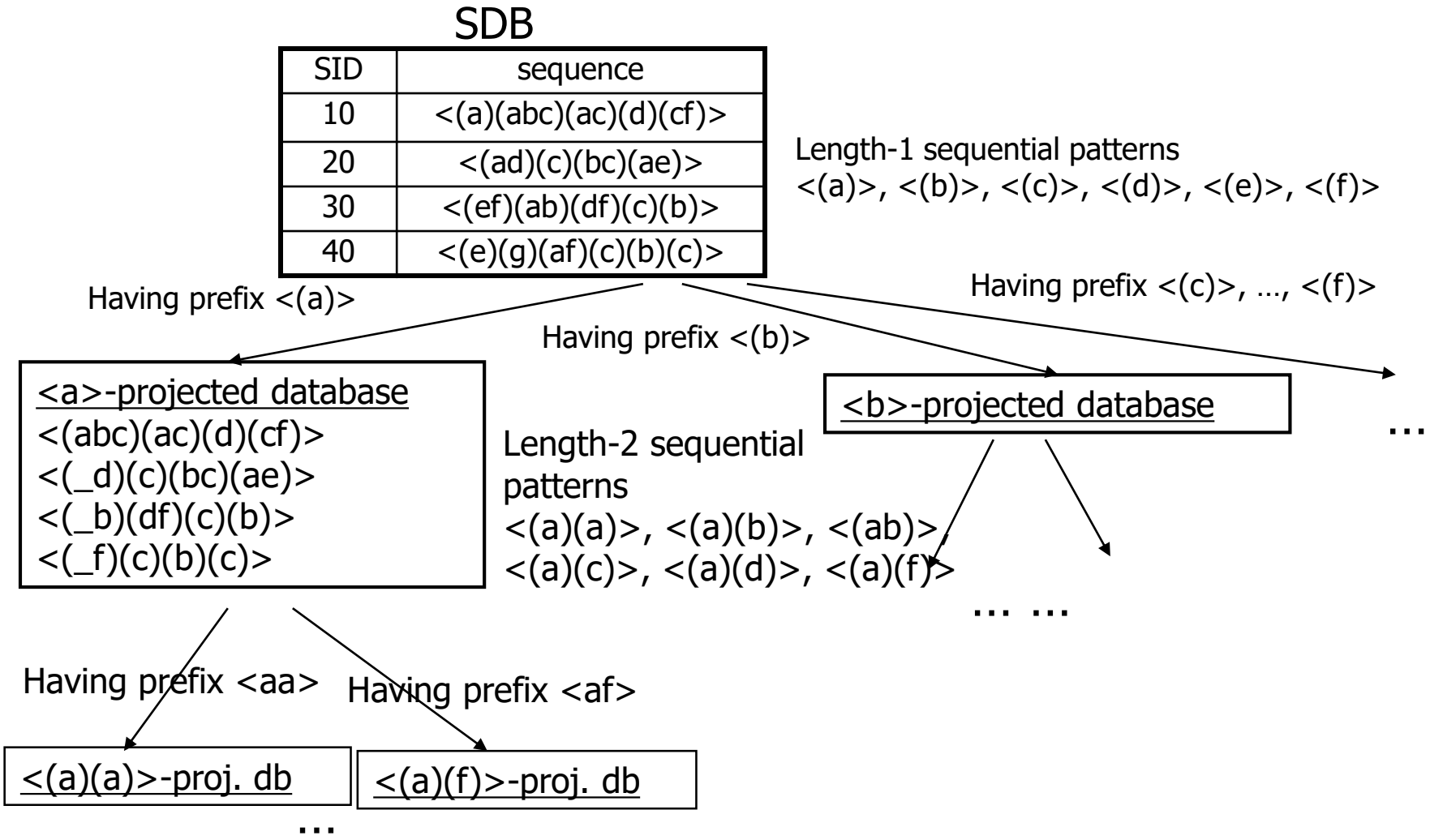
SID	sequence
10	<(a)(abc)(ac)(d)(cf)>
20	<(ad)(c)(bc)(ae)>
30	<(ef)(ab)(df)(c)(b)>
40	<(e)(g)(af)(c)(b)(c)>

Finding Seq. Patterns with Prefix <a>

- Only need to consider projections w.r.t. <(a)>
  - <a>-projected database: <(abc)(ac)(d)(cf)>, <(_d)(c)(bc)(ae)>, <(_b)(df)(c)(b)>, <(_f)(c)(b)(c)>
- Find all the length-2 seq. pat. Having prefix <(a)>: <(a)(a)>, <(a)(b)>, <(ab)>, <(a)(c)>, <(a)(d)>, <(a)(f)>
  - Further partition into 6 subsets
    - Having prefix <(a)(a)>;
    - …
    - Having prefix <(a)(f)>

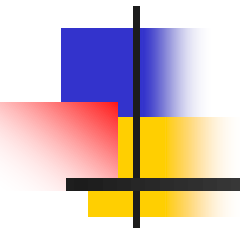
SID	sequence
10	<(a)(abc)(ac)(d)(cf)>
20	<(ad)(c)(bc)(ae)>
30	<(ef)(ab)(df)(c)(b)>
40	<(e)(g)(af)(c)(b)(c)>

Completeness of PrefixSpan



Efficiency of PrefixSpan

- No candidate sequence needs to be generated
- Projected databases keep shrinking
- Major cost of PrefixSpan: constructing projected databases
- Can be improved by pseudo-projections

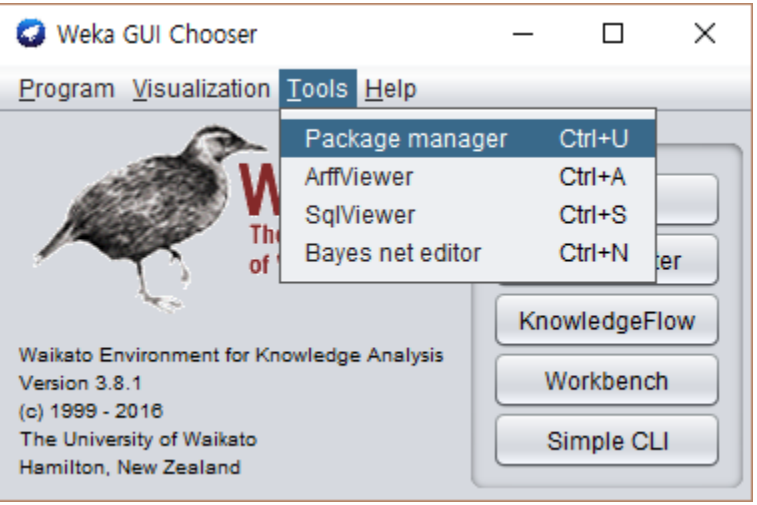


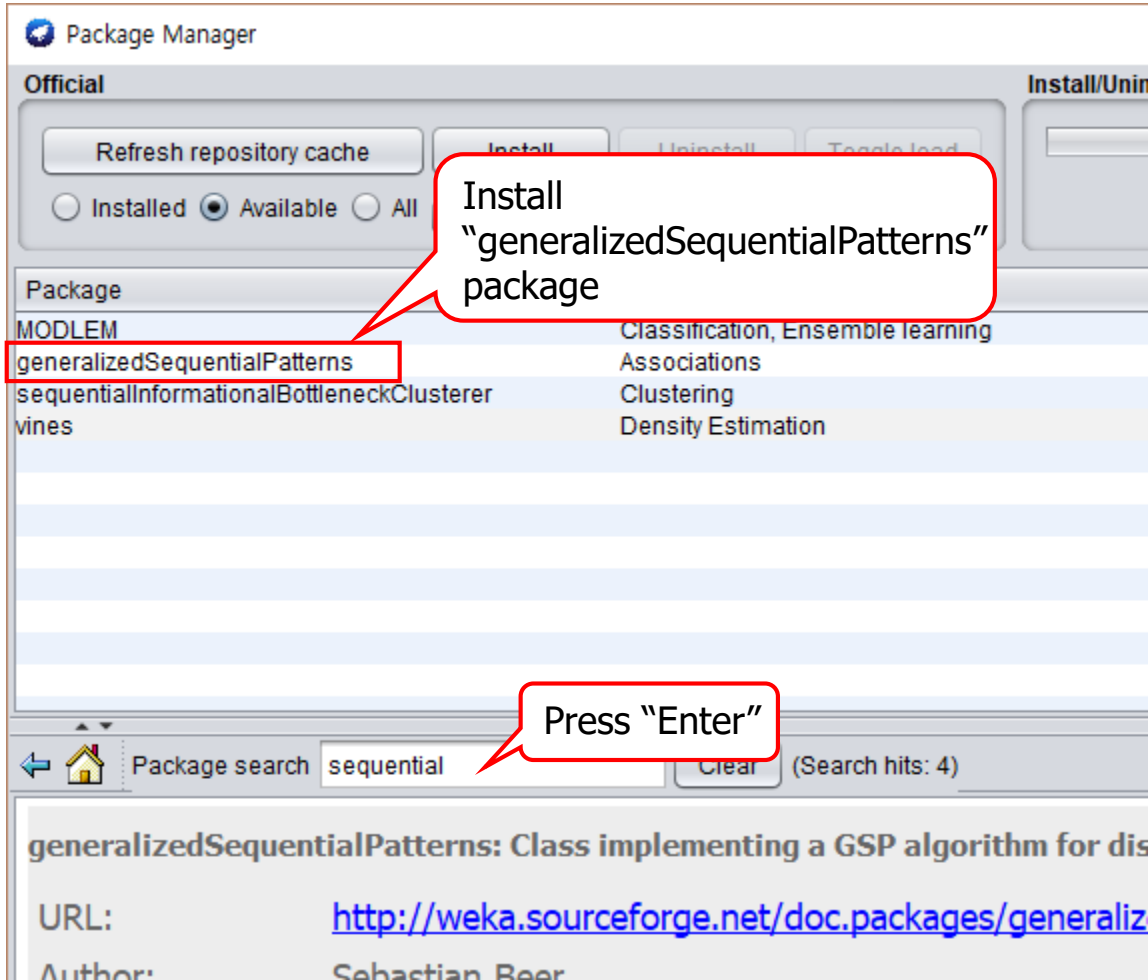
Practicing with Weka

Download the Additional Dataset

- http://kdd.snu.ac.kr/weka/
  - supermarket-seq.arff

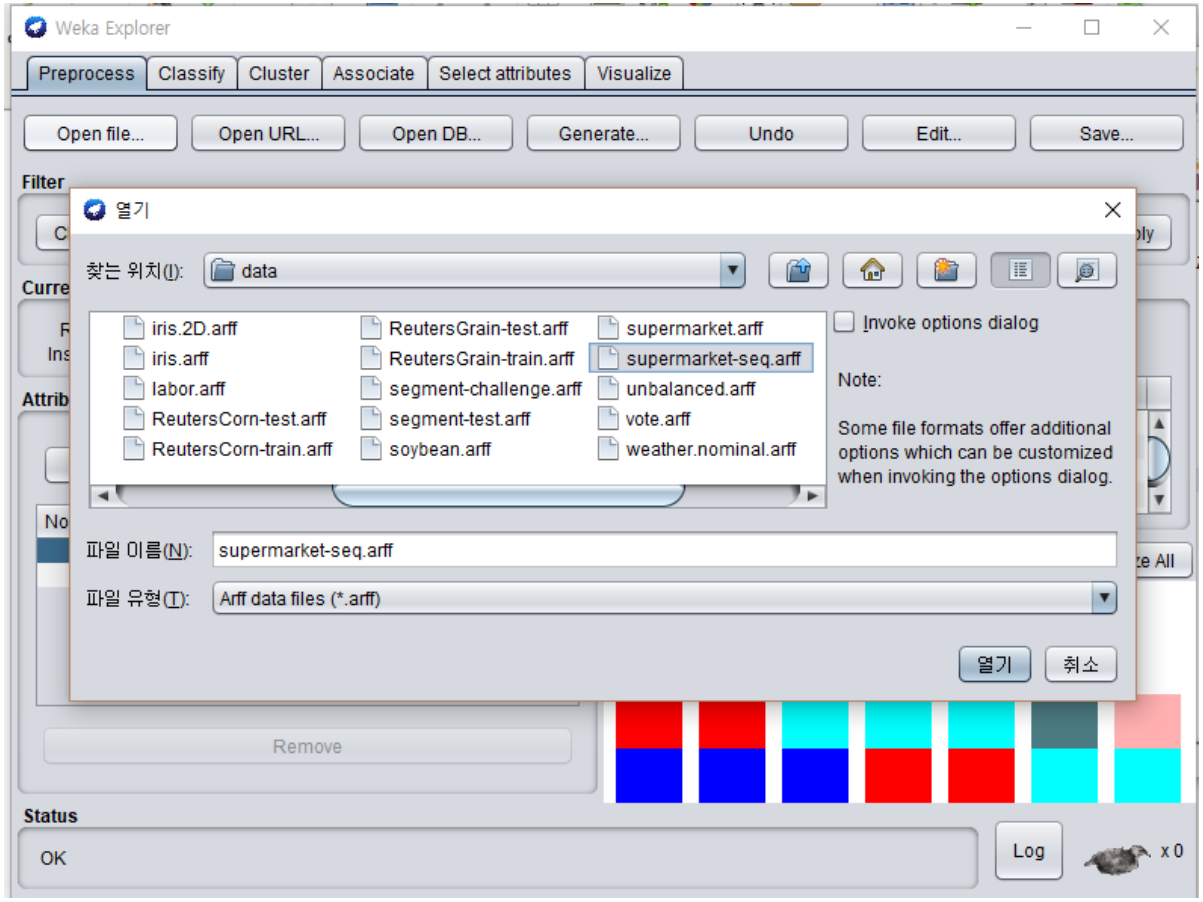
Weka- Sequential Pattern Mining





Weka- Sequential Pattern Mining

- Open “supermarket-seq.arff” file



Weka- Sequential Pattern Mining

Purchase Date	Items
2018/09/24	Milk
2018/09/25	Eggs
2018/09/27	Coffee

Customer 1

Purchase Date	Items
2018/09/18	Milk
2018/09/20	Eggs, Diapers

Customer 2

Purchase Date	Items
2018/09/27	Milk
2018/09/28	Diapers
2018/09/29	Coffee

Customer 3

Purchase Date	Items
2018/09/30	Eggs
2018/10/02	Coffee
2018/10/03	Sugar

Customer 4

Purchase Date	Items
2018/09/24	Eggs, Diapers
2018/09/26	Coffee

Customer 5

Purchase Date	Items
2018/09/25	Coffee
2018/09/26	Sugar
2018/09/28	Diapers

Customer 6

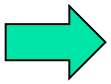
Purchase Date	Items
2018/09/29	Diapers
2018/10/01	Coffee,Sugar

Customer 7

Customer	Product
1	Milk
1	Eggs
1	Coffee
2	Milk
2	Eggs, Diapers
3	Milk
3	Diapers
3	Coffee
4	Eggs
4	Coffee
4	Sugar
5	Eggs, Diapers
5	Coffee
6	Coffee
6	Sugar
6	Diapers
7	Diapers
7	Coffee, Sugar

Weka- Sequential Pattern Mining

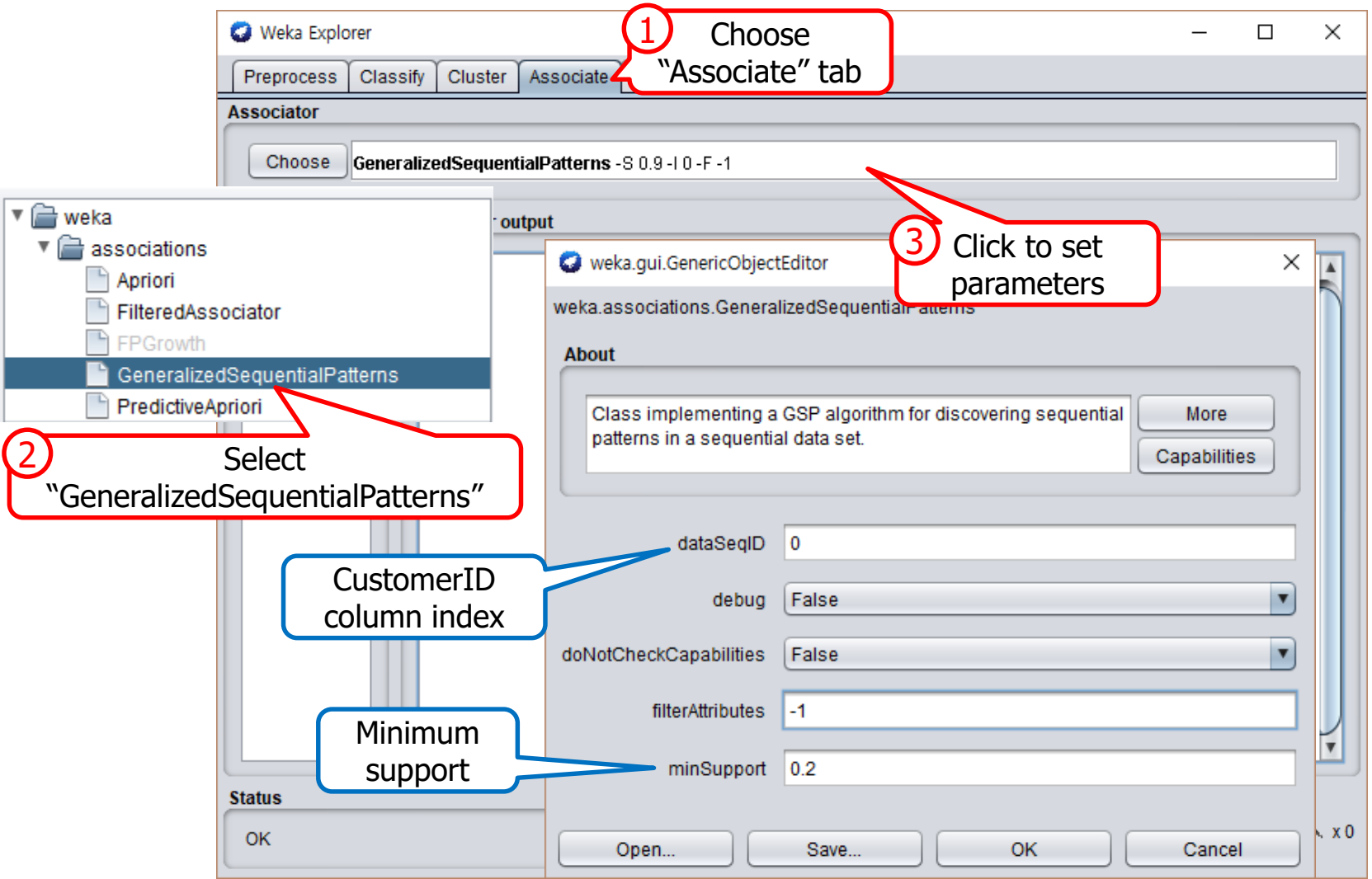
Customer	Product
1	Milk
1	Eggs
1	Coffee
2	Milk
2	Eggs, Diapers
3	Milk
3	Diapers
3	Coffee
4	Eggs
4	Coffee
4	Sugar
5	Eggs, Diapers
5	Coffee
6	Coffee
6	Sugar
6	Diapers
7	Diapers
7	Coffee, Sugar



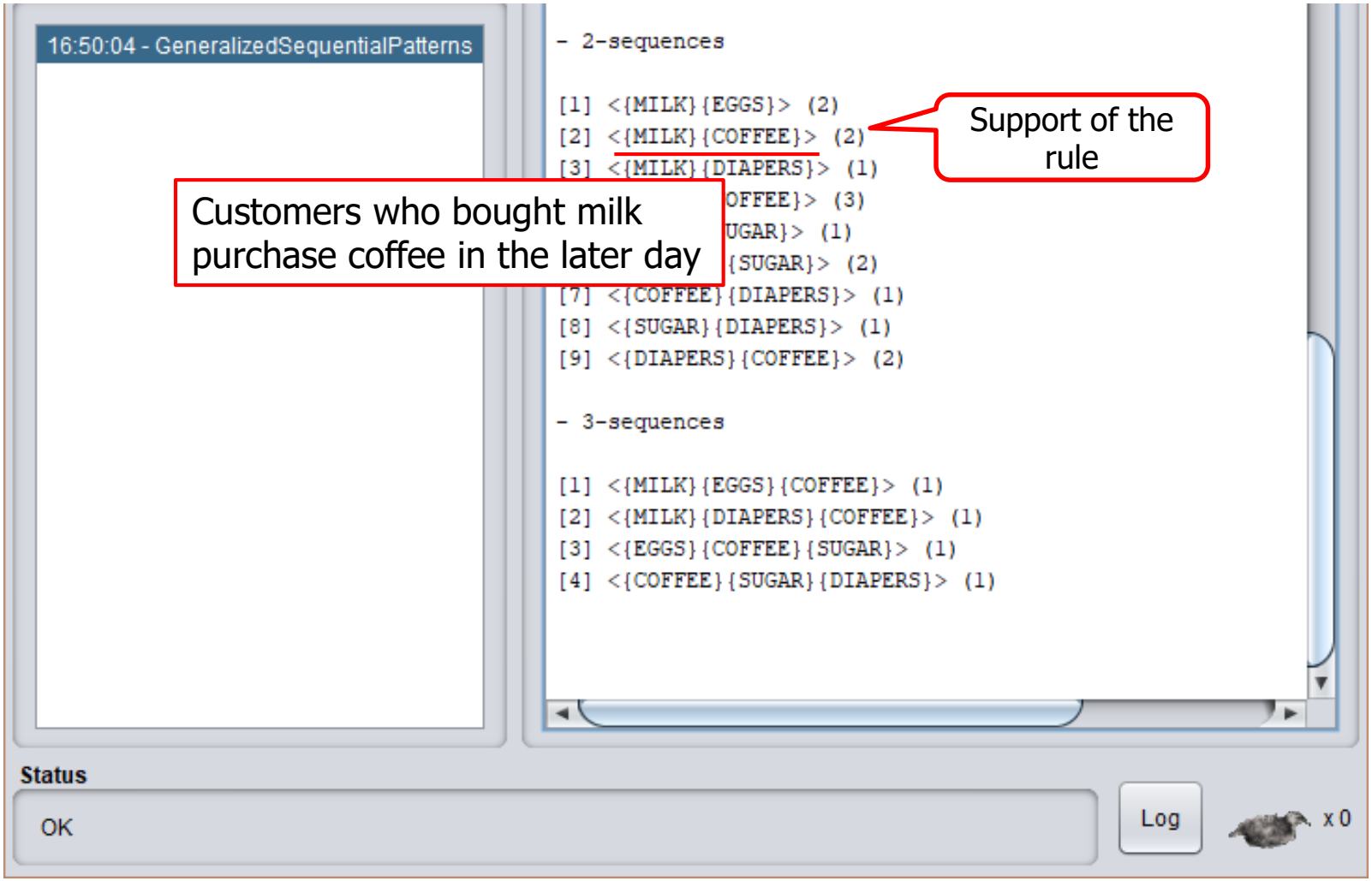
```
1  @relation market_sequence
2  @attribute personID {1,2,3,4,5,6,7}
3  @attribute item {MILK, EGGS, COFFEE, SUGAR, DIAPERS}
4  @data
5  1, MILK
6  1, EGGS
7  1, COFFEE
8  2, MILK
9  2, EGGS,DIAPERS
10 3, MILK
11 3, DIAPERS
12 3, COFFEE
13 4, EGGS
14 4, COFFEE
15 4, SUGAR
16 5, EGGS,DIAPERS
17 5, COFFEE
18 6, COFFEE
19 6, SUGAR
20 6, DIAPERS
21 7, DIAPERS
22 7, COFFEE,SUGAR
```

length : 330 lines : 22 Ln : 1 Col : 1 Sel : 0 | 0 Windows (C

Weka- Sequential Pattern Mining



Weka- Sequential Pattern Mining



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