

Mining Frequent Patterns, Associations, and Correlation

Prepared by Dr. Amjad Qtaish

DS601
Data Mining

Lecture 06



Lecture: #06

Mining Frequent Patterns, Associations, and Correlation



Dr. Amjad Qtaish

Faculty of Informatics
Midocean University





Mining Frequent Patterns, Associations, and Correlation: Basic Concepts and Methods

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





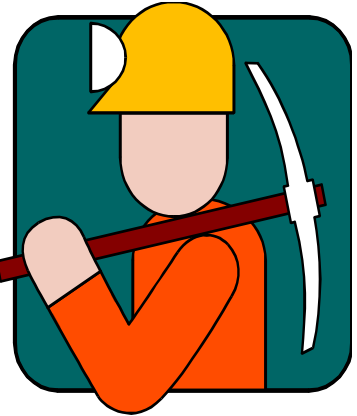
What Is Frequent Pattern Analysis?



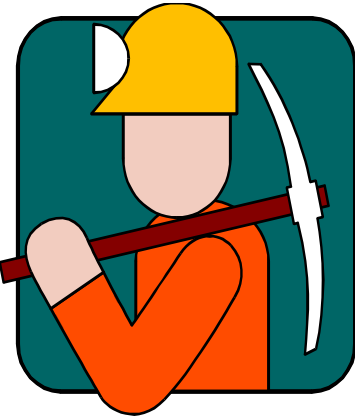
- **Frequent pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- **Motivation:** Finding inherent regularities 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.



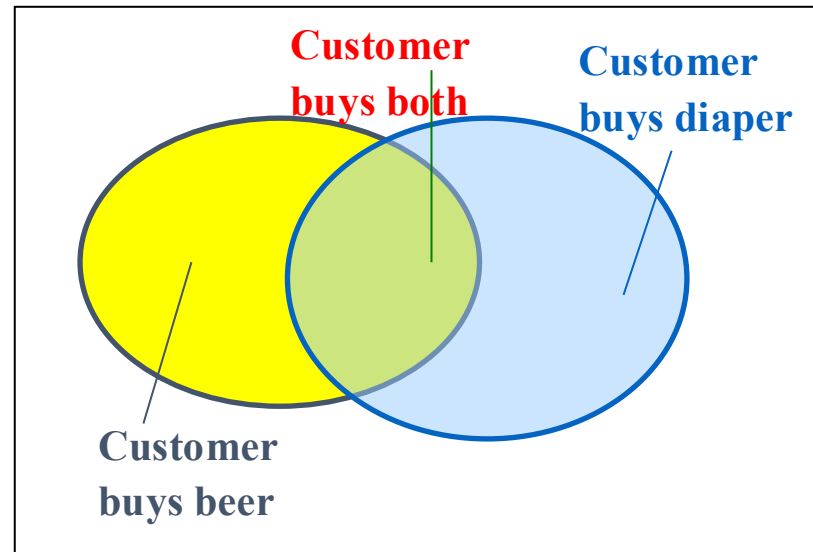
Why Is Freq. Pattern Mining Important?



- 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



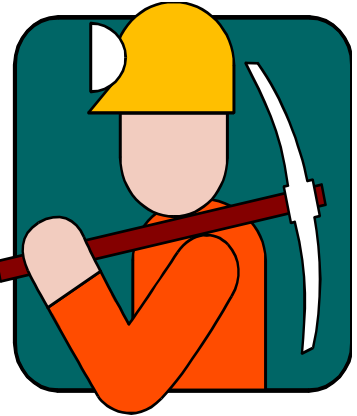
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



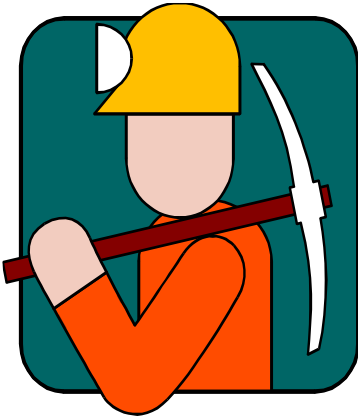
- **itemset**: A set of one or more items
- **k-itemset** $X = \{x_1, \dots, x_k\}$
- **(absolute) support**, or, **support count** of X : Frequency or occurrence of an itemset X
- **(relative) support**, s , is the fraction of transactions that contains X (i.e., the probability that a transaction contains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



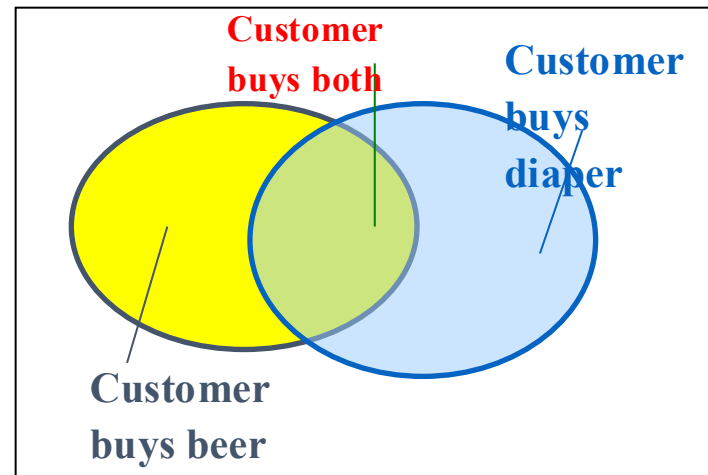
Computer -> antivirus software [support D 2%,confidence D 60%]



- Rule **support** and **confidence** are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules.
- A **support** of 2% for the above Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A **confidence** of 60% means that 60% of the customers who purchased a computer also bought the software



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

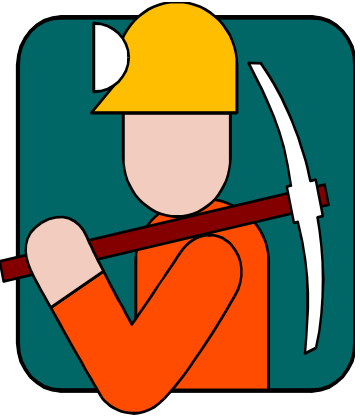


- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - support**, s , probability that a transaction contains $X \cup Y$
 - confidence**, c , conditional probability that a transaction having X also contains Y

Let $\text{minsup} = 50\%$, $\text{minconf} = 50\%$

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

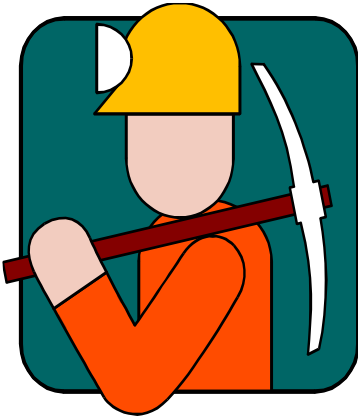
- Association rules: (many more!)
 - $\text{Beer} \rightarrow \text{Diaper}$ (60%, 100%)
 - $\text{Diaper} \rightarrow \text{Beer}$ (60%, 75%)



- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \dots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = 1.27 \cdot 10^{30}$ sub-patterns! **permutations**
- Solution: Mine *closed patterns* and *max-patterns* instead
- An itemset X is **closed** if X is *frequent* and there exists *no super-pattern* $Y \supset X$, 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 $Y \supset X$
- Closed pattern is a lossless compression of freq. patterns
 - Reducing the # of patterns and rules



Closed Patterns and Max-Patterns



$\{A\} = 4$; not closed due to $\{A,E\}$
 $\{B\} = 2$; not frequent $\Rightarrow$ ignore
 $\{C\} = 5$; not closed due to $\{C,E\}$
 $\{D\} = 4$; closed, and not maximal due to e.g.
 $\{A,D\}$
 $\{E\} = 6$; closed, but not maximal due to e.g.
 $\{D,E\}$

$\{A,B\} = 1$; not frequent $\Rightarrow$ ignore
 $\{A,C\} = 3$; not closed due to $\{A,C,E\}$
 $\{A,D\} = 3$; not closed due to $\{A,D,E\}$
 $\{A,E\} = 4$; closed, but not maximal due to
 $\{A,D,E\}$
 $\{B,C\} = 2$; not frequent $\Rightarrow$ ignore
 $\{B,D\} = 0$; not frequent $\Rightarrow$ ignore
 $\{B,E\} = 2$; not frequent $\Rightarrow$ ignore
 $\{C,D\} = 3$; not closed due to $\{C,D,E\}$
 $\{C,E\} = 5$; closed, but not maximal due to
 $\{C,D,E\}$
 $\{D,E\} = 4$; closed, but not maximal due to
 $\{A,D,E\}$

My dataset:

1: **A,B,C,E**
2: **A,C,D,E**
3: **B,C,E**
4: **A,C,D,E**
5: **C,D,E**
6: **A,D,E**

minsupp=50%

$\{A,B,C\} = 1$; not frequent $\Rightarrow$ ignore
 $\{A,B,D\} = 0$; not frequent $\Rightarrow$ ignore
 $\{A,B,E\} = 1$; not frequent $\Rightarrow$ ignore
 $\{A,C,D\} = 2$; not frequent $\Rightarrow$ ignore
 $\{A,C,E\} = 3$; maximal frequent
 $\{A,D,E\} = 3$; maximal frequent
 $\{B,C,D\} = 0$; not frequent $\Rightarrow$ ignore
 $\{B,C,E\} = 2$; not frequent $\Rightarrow$ ignore
 $\{C,D,E\} = 3$; maximal frequent

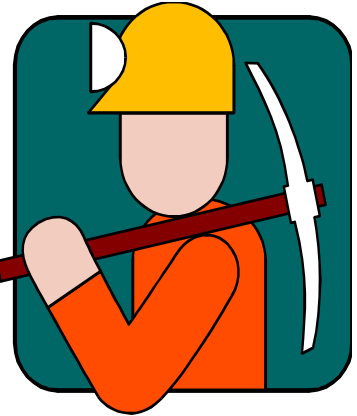
$\{A,B,C,D\} = 0$; not frequent $\Rightarrow$ ignore
 $\{A,B,C,E\} = 1$; not frequent $\Rightarrow$ ignore
 $\{B,C,D,E\} = 0$; not frequent $\Rightarrow$ ignore



Mining Frequent Patterns, Associations, and Correlation: Basic Concepts and Methods

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

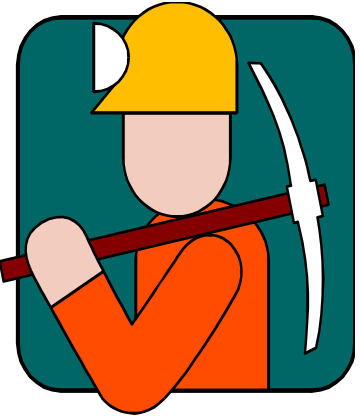




- Apriori: A Candidate Generation-and-Test Approach 
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Close Frequent Patterns and Maxpatterns



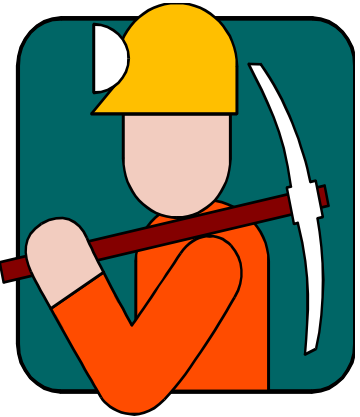
The Downward Closure Property and Scalable Mining Methods



- 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
 - Freq. pattern growth
 - Vertical data format approach



Apriori: A Candidate Generation & Test Approach



- Apriori pruning principle: If there is **any** itemset which is infrequent, its superset should not be generated/tested!
- **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



Database TDB

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

$Sup_{min} = 2$

C_1
1st scan

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

L_1

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

L_2

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

C_2

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

2nd scan

C_2

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

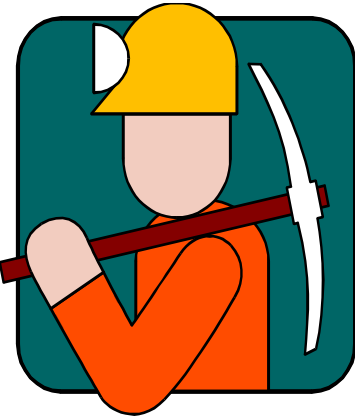
C_3

Itemset	sup
{B, C, E}	2

3rd scan

L_3

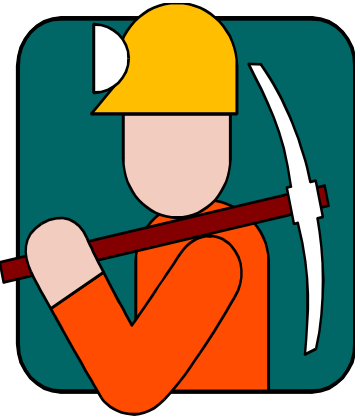
Itemset	sup
{B, C, E}	2



- 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
 - $C_4 = \{abcd\}$



Apriori-Example 2



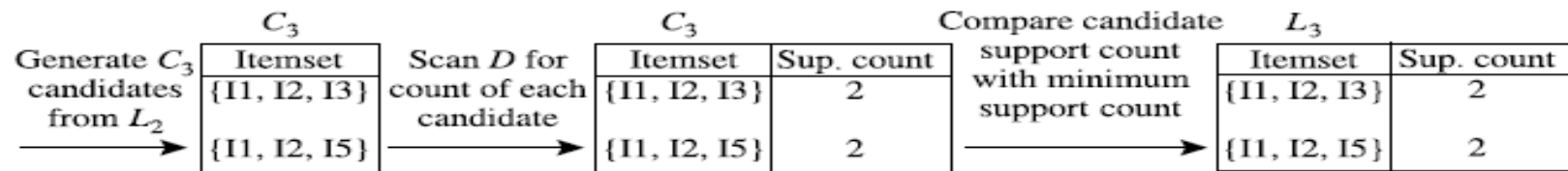
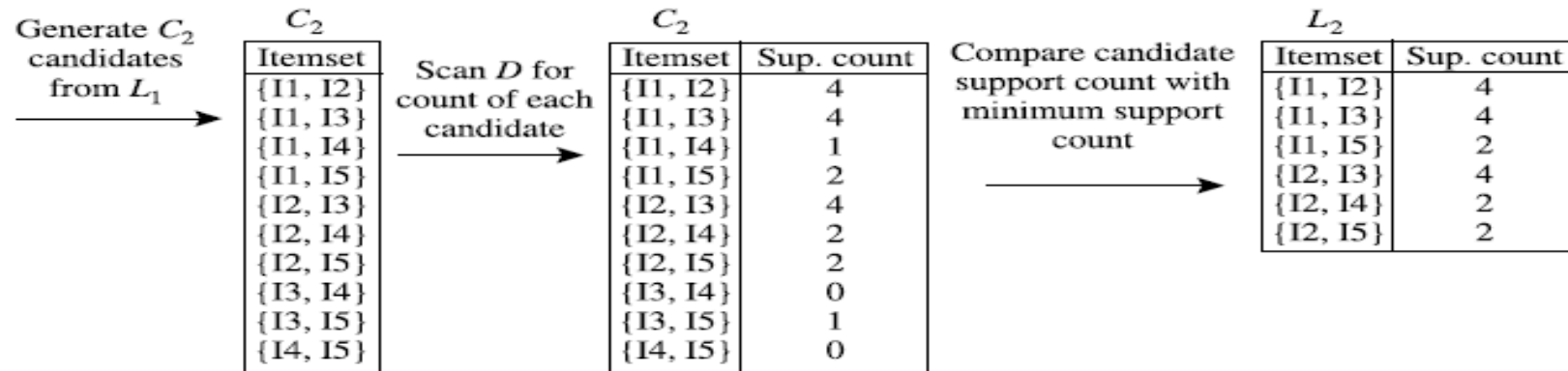
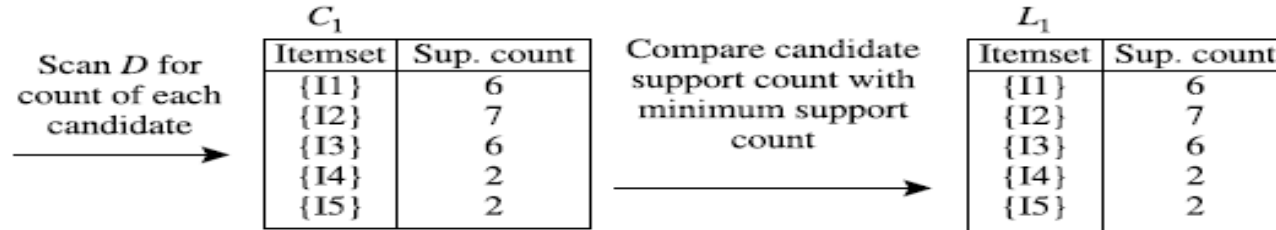
- Let's look at a concrete example, based on the *AllElectronics* transaction database, D , of the following Table. There are nine transactions in this database, that is, $|D| = 9$. We use Next slide to illustrate the Apriori algorithm for finding frequent itemsets in D .

Transactional Data for an *AllElectronics* Branch

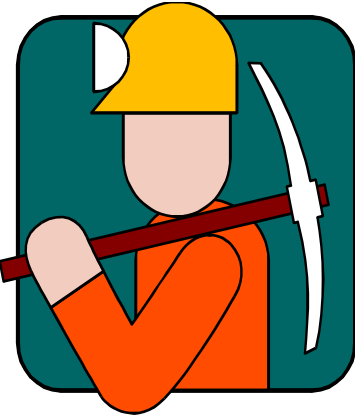
<i>TID</i>	<i>List of item IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



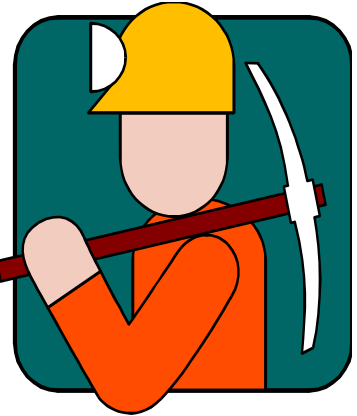
Apriori-Example 2 (Solution)



Generation of the candidate itemsets and frequent itemsets, where the minimum support count is 2.



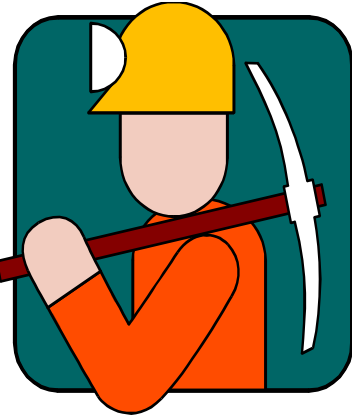
- Open Weka -> explorer
 - Load the fp.arff to the explorer
 - Click Associate tab
 - Apriori
 - Start
-
- Paper : Using Apriori with WEKA for Frequent Pattern Mining, Paresh Tanna



- Apriori: A Candidate Generation-and-Test Approach
- FPGrowth: A Frequent Pattern-Growth Approach 
- 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
 - 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 $DB|abc \rightarrow abcd$ is a frequent pattern



Construct FP-tree from a Transaction Database



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

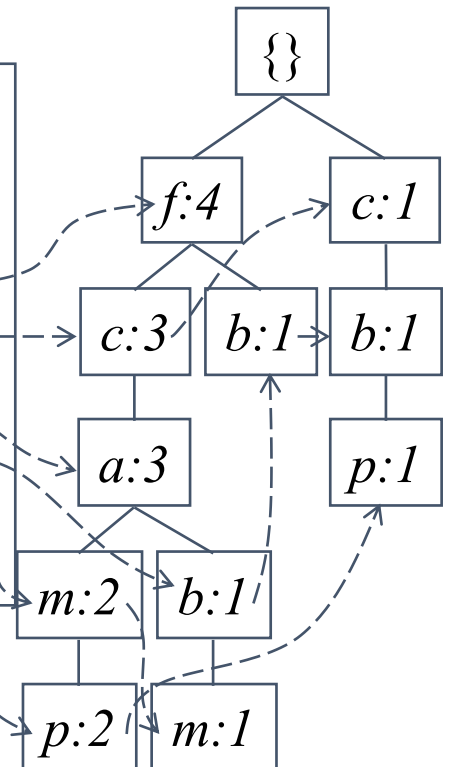
$min_support = 3$

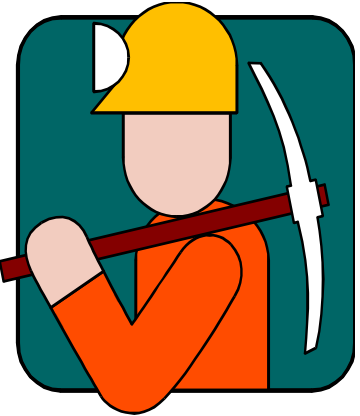
1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

<u>Item</u>	<u>frequency</u>	<u>head</u>
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

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

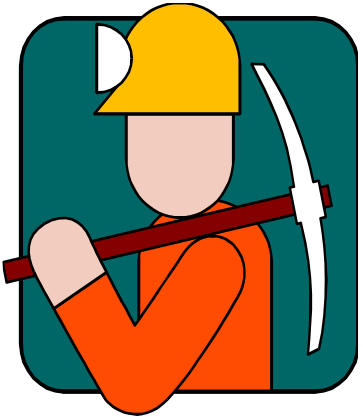




- Frequent patterns can be partitioned into subsets according to f-list
 - F-list = f-c-a-b-m-p
 - Patterns containing p
 - Patterns having m but no p
 - ...
 - Patterns having c but no a nor b, m, p
 - Pattern f
- Completeness and non-redundancy

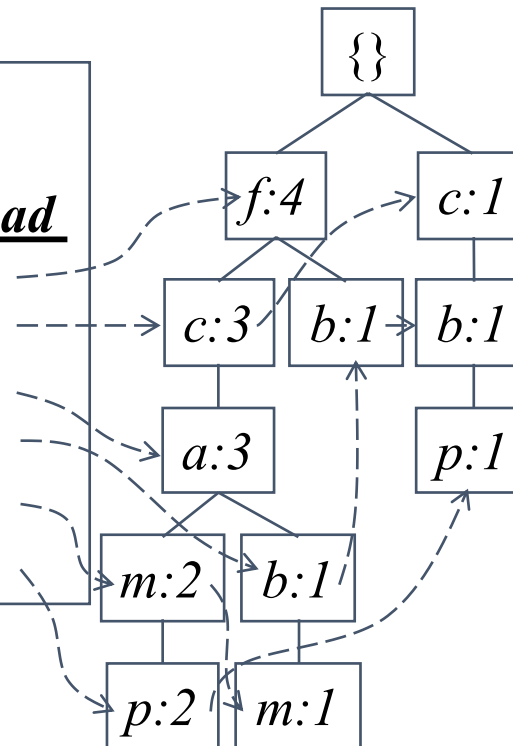


Find Patterns Having P From P-conditional Database



- 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

Header Table	
<i>Item</i>	<i>frequency head</i>
f	4
c	4
a	3
b	3
m	3
p	3



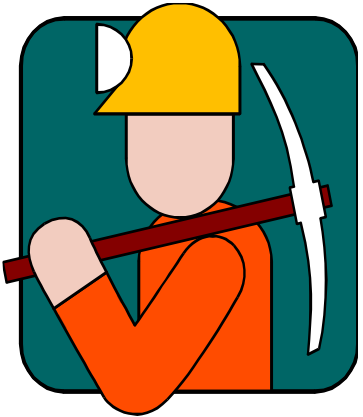
Conditional pattern bases

<i>item</i>	<i>cond. pattern base</i>
c	$f:3$
a	$fc:3$
b	$fca:1, f:1, c:1$
m	$fca:2, fcab:1$
p	$fcam:2, cb:1$

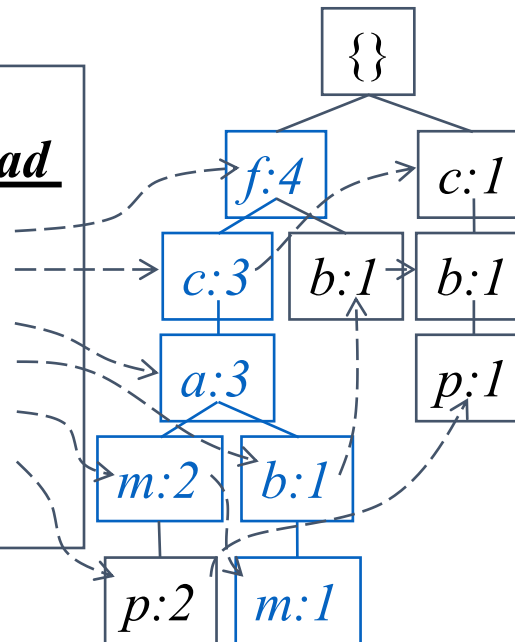


From Conditional Pattern-bases to Conditional FP-trees

- 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



Header Table		
<i>Item</i>	<i>frequency</i>	<i>head</i>
<i>f</i>	4	
<i>c</i>	4	
<i>a</i>	3	
<i>b</i>	3	
<i>m</i>	3	
<i>p</i>	3	



m-conditional pattern base:
fca:2, fcab:1

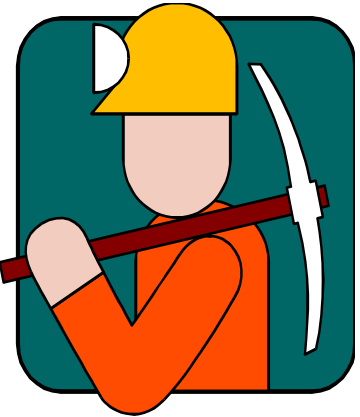


{
|
f:3
|
c:3
|
a:3



All frequent
patterns relate to *m*
m,
fm, cm, am,
fc m, fam, cam,
fcam

m-conditional FP-tree



$\{$
|
 $f:3$
|
 $c:3$
|
 $a:3$

m-conditional FP-tree

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

$\{$
|
 $f:3$
|
 $c:3$

am-conditional FP-tree

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

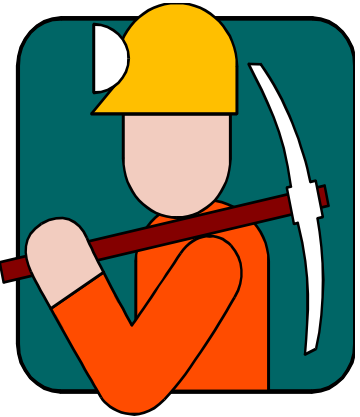
$\{$
|
 $f:3$

cm-conditional FP-tree

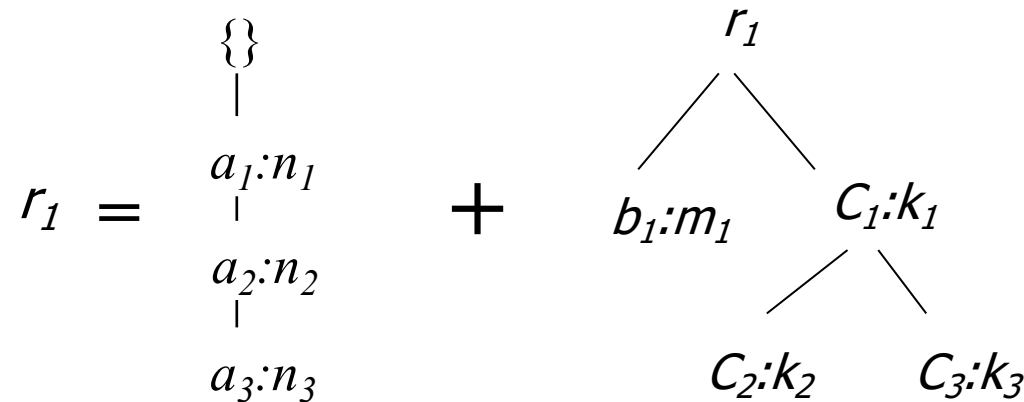
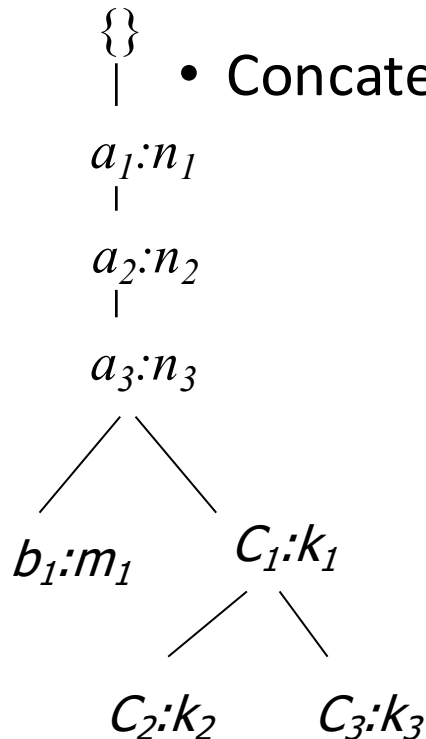
Cond. pattern base of "cam": (f:3)

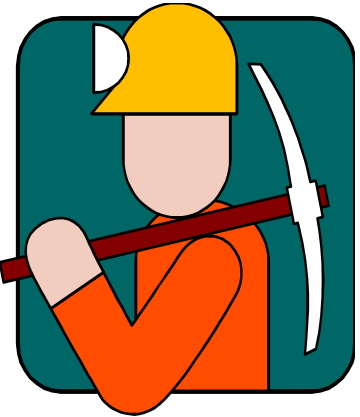
$\{$
|
 $f:3$

cam-conditional FP-tree

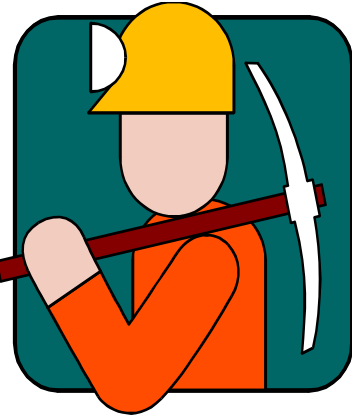


- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
 - Reduction of the single prefix path into one node
 - Concatenation of the mining results of the two parts

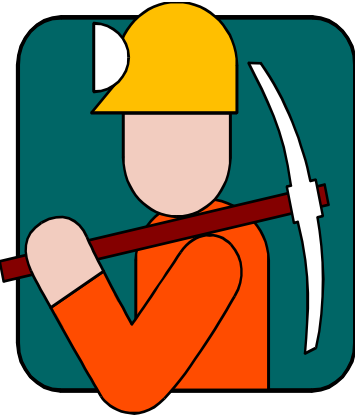




- **Completeness**
 - Preserve complete information for frequent pattern mining
 - Never break a long pattern of any transaction
- **Compactness**
 - Reduce irrelevant info—infrequent items are gone
 - Items in frequency descending order: the more frequently occurring, the more likely to be shared
 -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



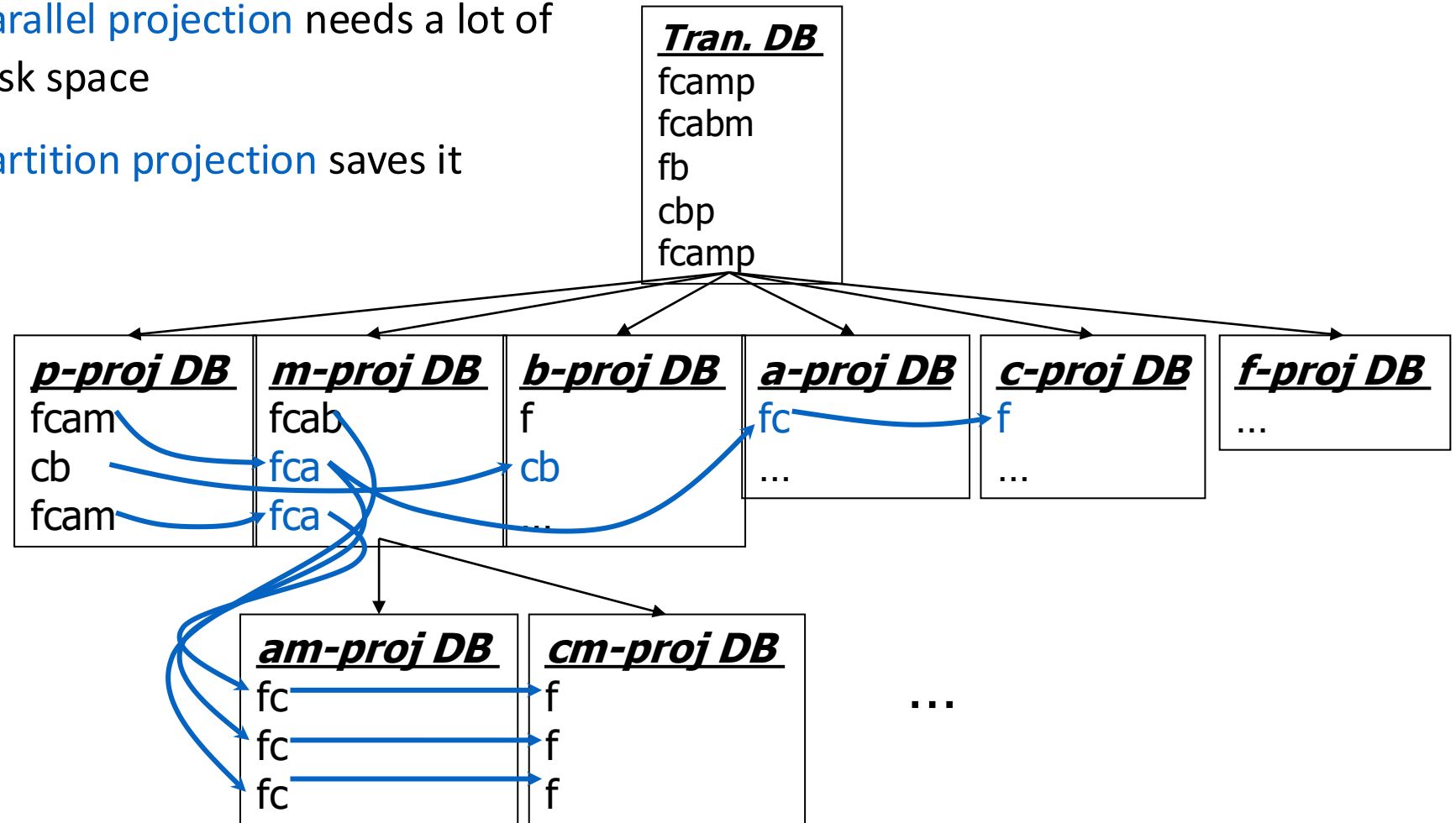
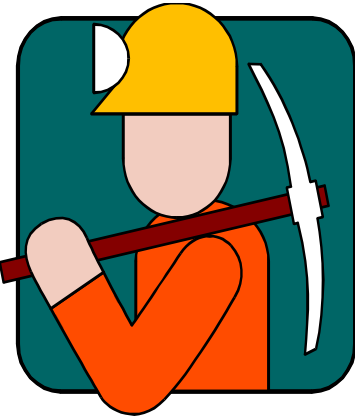
- **Idea:** Frequent pattern growth
 - Recursively grow frequent patterns by pattern and database partition
- **Method**
 - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
 - Repeat the process on each newly created conditional FP-tree
 - Until the resulting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

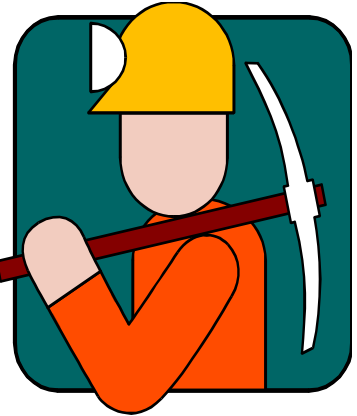


- What about if FP-tree cannot fit in memory?
 - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- **Parallel projection** vs. **partition projection** techniques
 - Parallel projection
 - Project the DB in parallel for each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

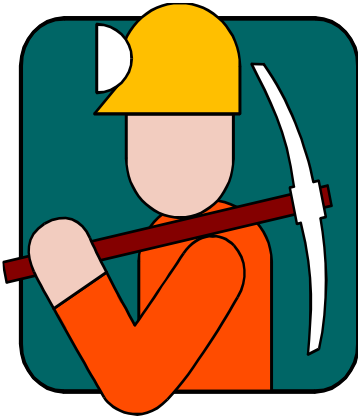


- Parallel projection needs a lot of disk space
- Partition projection saves it



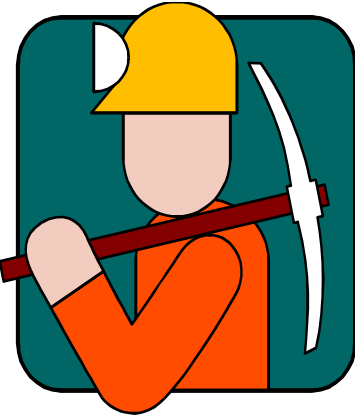


- Apriori: A Candidate Generation-and-Test Approach
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Close Frequent Patterns and Maxpatterns 

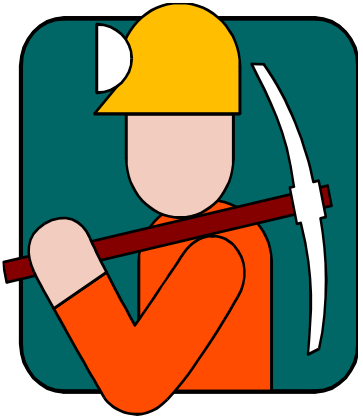


- Flist: list of all frequent items in support ascending order
 - Flist: d-a-f-e-c
- Divide search space
 - Patterns having d
 - Patterns having d but no a, etc.
- Find frequent closed pattern recursively
 - Every transaction having d also has *cfa* $\rightarrow$ *cfad* is a frequent closed pattern
-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

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



- Itemset merging: if Y appears in every occurrence of X , then Y is merged with X
- Sub-itemset pruning: if $Y \supset X$, and $\text{sup}(X) = \text{sup}(Y)$, X and all of X 's descendants in the set enumeration tree can be pruned
- 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



- 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

Potential
max-patterns

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

- R. Bayardo. Efficiently mining long patterns from databases.

SIGMOD'98

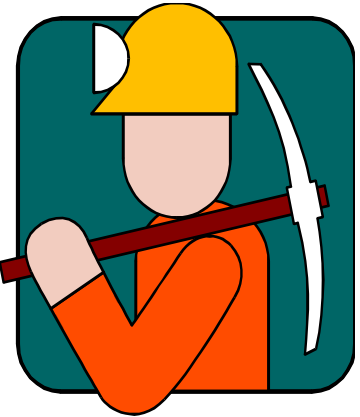
Tid	Items
10	A, B, C, D, E
20	B, C, D, E,
30	A, C, D, F



Mining Frequent Patterns, Associations, and Correlation: Basic Concepts and Methods

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





- *play basketball* $\Rightarrow$ *eat cereal* [40%(2000/5000), 66.7%(2000/3000)] 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



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

	<i>milk</i>	$\overline{milk}$	Σ_{row}
<i>coffee</i>	<i>mc</i>	$\overline{mc}$	<i>c</i>
$\overline{coffee}$	$\overline{mc}$	$\overline{mc}$	$\overline{c}$
Σ_{col}	<i>m</i>	$\overline{m}$	Σ

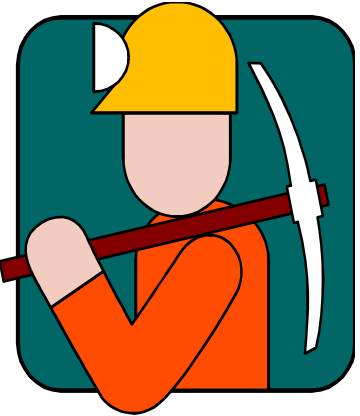
<i>Data</i>	<i>mc</i>	$\overline{mc}$	$\overline{mc}$	$\overline{mc}$	<i>all_conf.</i>	<i>max_conf.</i>	<i>Kulc.</i>	<i>cosine</i>	IR
D_1	10,000	1,000	1,000	100,000	0.91	0.91	0.91	0.91	0.0
D_2	10,000	1,000	1,000	100	0.91	0.91	0.91	0.91	0.0
D_3	100	1,000	1,000	100,000	0.09	0.09	0.09	0.09	0.0
D_4	1,000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



Mining Frequent Patterns, Associations, and Correlation: Basic Concepts and Methods

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



Thank You