

---

# Data Science (CS3206)

## Association Rules

# In today's discussion...

---

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

# What Is Frequent Pattern Analysis?

- **Frequent pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?—Milk and Bread
  - What are the subsequent purchases after buying a PC?
    - Digital camera, memory card sequential pattern
  - What kinds of DNA are sensitive to new drug?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, DNA sequence analysis.

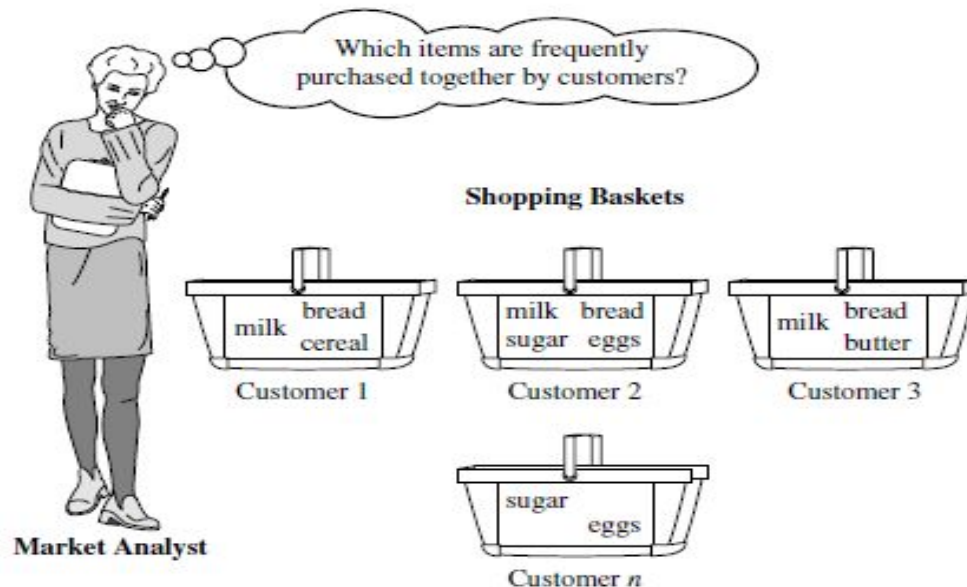
# Why Is Freq. Pattern Mining Important?

---

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data analysis 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

# What Is Frequent Pattern Analysis?

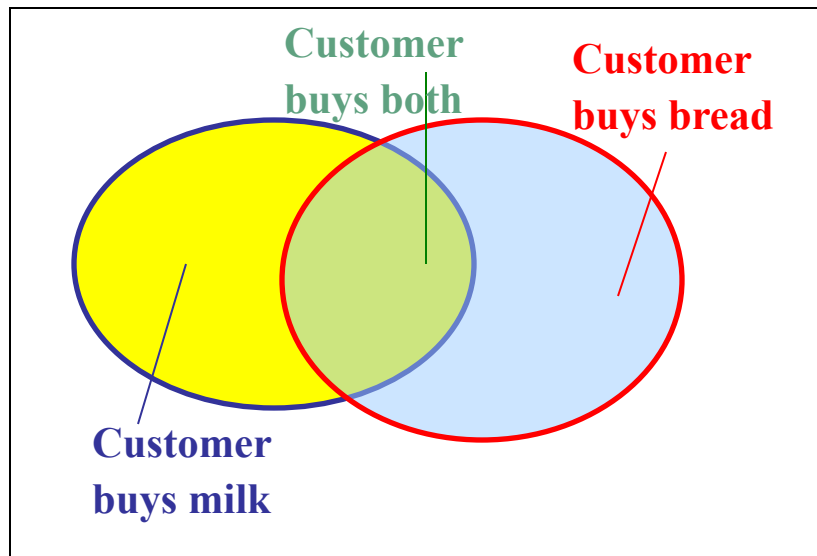
- Frequent pattern mining searches for recurring relationships in a given data set.
- Market basket analysis
  - analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”



Market basket analysis.

# Basic Concepts: Frequent Patterns

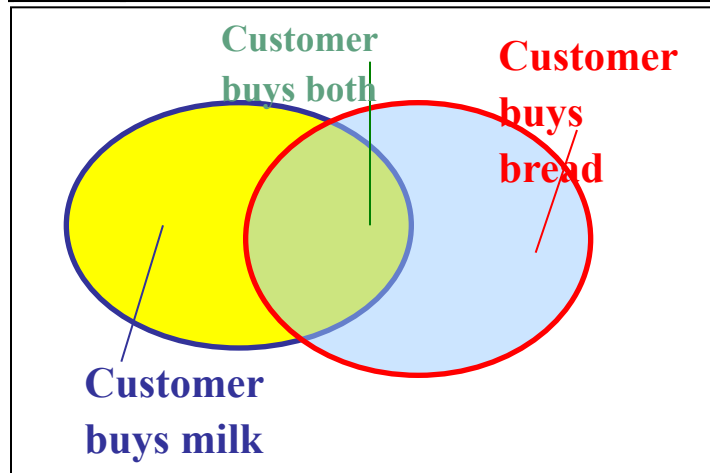
Tid	Items bought
10	Milk, Nuts, Bread
20	Milk, Coffee, Bread
30	Milk, Bread, Eggs
40	Nuts, Eggs, Tea
50	Nuts, Coffee, Bread, Eggs, Tea



- **itemset**: A set of one or more items
- **k-itemset**  $X = \{x_1, \dots, x_k\}$   
 $\{\text{Milk}\} \{\text{Nuts}\} \{\text{Bread}\} \{\text{Milk, Bread}\} \{\text{Milk, Nuts}\}$
- **(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	Milk, Nuts, Bread
20	Milk, Coffee, Bread
30	Milk, Bread, Eggs
40	Nuts, Eggs, Tea
50	Nuts, Coffee, Bread, Eggs, Tea



- Find all the association rules  $A \Rightarrow B$  with minimum support and confidence
  - support**,  $s$ , **probability** that a transaction contains  $A \cup B$   
(i.e., the union of sets  $A$  and  $B$  say, or, both  $A$  and  $B$ , it contains every item in  $A$  and  $B$ )
  - confidence**,  $c$ , **conditional probability** that a transaction having  $A$  also contains  $B$

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

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

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}.$$

# Basic Concepts: Association Rules

Tid	Items bought
10	Milk, Nuts, Bread
20	Milk, Coffee, Bread
30	Milk, Bread, Eggs
40	Nuts, Eggs, Tea
50	Nuts, Coffee, Bread, Eggs, Tea

Let  $minsup = 50\%$ ,  $minconf = 50\%$  domain expert

Freq. Pat.: Milk, Nuts, Bread, Eggs, Tea  
 $\{\text{Milk, Bread}\}$  :2-itemset

Freq. Pat.: Milk:3, Nuts:3, Bread:4, Eggs:3, Tea:2  
 $\{\text{Milk, Bread}\}$ :3

A support of 60% for means that 60% of all the transactions under analysis show that Milk & Bread are purchased together.

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence(A \Rightarrow B) = P(B|A).$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

A confidence of 100% means that 100% of the customers who purchased Milk also bought the Bread

Association rules: (many more!)

- $\text{Milk} \square \text{Bread}$  (60%, 100%)  
 $(\{\text{Milk, Bread}\}:3/\text{total\_transactions}:5=60\%, \{\text{Milk, Bread}\}:3/\text{Milk}:3=100\%)$
- $\text{Bread} \square \text{Milk}$  (60%, 75%)  
 $(\{\text{Milk, Bread}\}:3/\text{total\_transactions}:5=60\%, \{\text{Milk, Bread}\}:3/\text{Bread}:4=75\%)$



# Association Rule

---

- A two-step process:

- 1. Find all frequent itemsets:**

frequency(itemsets)  $\geq$  minimum support count, *minsup*.

- 2. Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

- A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support (*min sup*) threshold, especially when *min sup* is set low.

# The Downward Closure Property

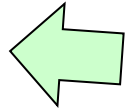
---

- The **downward closure** property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{milk, bread, nuts}** is frequent, so is **{milk, bread}**
  - i.e., every transaction having {milk, bread, nuts} also contains {milk, bread}

# Frequent Itemset Mining Methods

---

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



# Apriori: A Candidate Generation & Test Approach

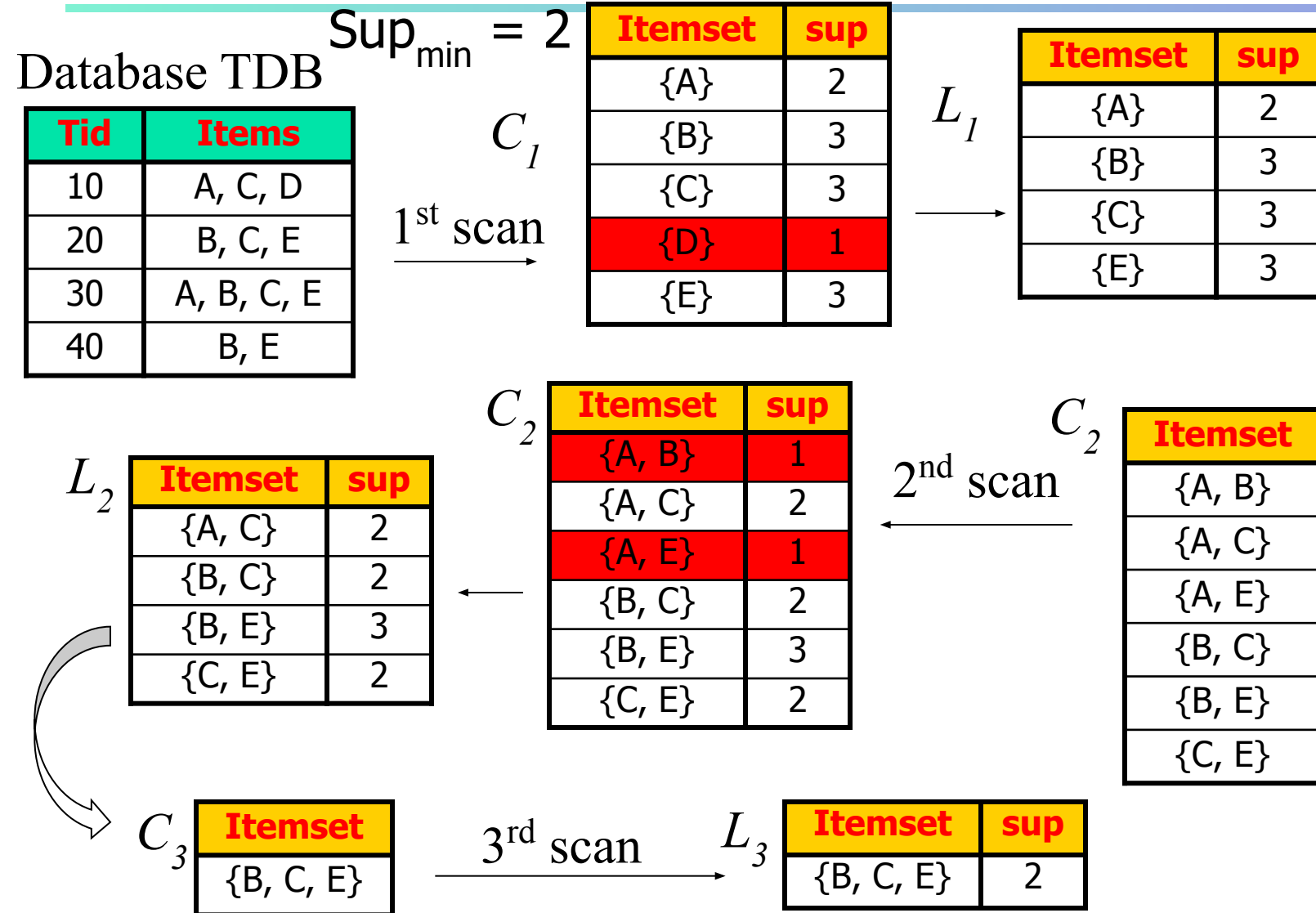
---

- 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  $k$

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **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**  $\bigcup_k L_k$ ;

# 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 (joinable if their first  $(k - 2)$  items are in common):  $L_3 * L_3$ 
    - $abcd$  from  $abc$  and  $abd$
    - $acde$  from  $acd$  and  $ace$
  - Pruning:
    - $acde$  is removed because  $ade$  is not in  $L_3$
  - $C_4 = \{abcd\}$

# Generating Association Rules from Frequent Itemsets

---

- Once the frequent itemsets from transactions in a database  $D$  have been found, it is straightforward to generate strong association rules from them (where *strong association* rules satisfy both minimum support and minimum confidence).
- Association rules can be generated as follows:
  - For each frequent itemset  $l$ , generate all nonempty subsets of  $l$ .
  - For every nonempty subset  $s$  of  $l$ , output the rule

$$“s \Rightarrow (l - s)” \text{ if } \frac{\text{support\_count}(l)}{\text{support\_count}(s)} \geq \text{min\_conf}$$



# Generating Association Rules from Frequent Itemsets

- The data contain frequent itemset  $X = \{B, C, E\}$
- The nonempty subsets of  $X$  are  $\{B, C\}$ ,  $\{B, E\}$ ,  $\{C, E\}$ ,  $\{B\}$ ,  $\{C\}$  and  $\{E\}$
- The resulting association rules are as shown below:

- $\{B, C\} \Rightarrow E$  confidence =  $2/2 = 100\%$
- $\{B, E\} \Rightarrow C$  confidence =  $2/3 = 66\%$
- $\{C, E\} \Rightarrow B$  confidence =  $2/2 = 100\%$
- $E \Rightarrow \{B, C\}$  confidence =  $2/3 = 66\%$
- $C \Rightarrow \{B, E\}$  confidence =  $2/3 = 66\%$
- $B \Rightarrow \{C, E\}$  confidence =  $2/3 = 66\%$

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

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

Itemset	sup
{B, C, E}	2

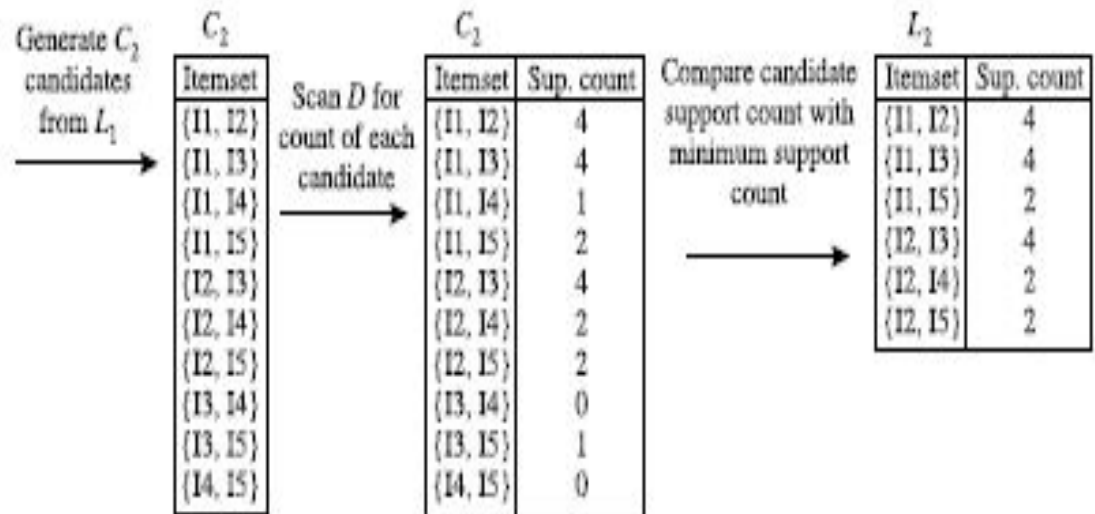
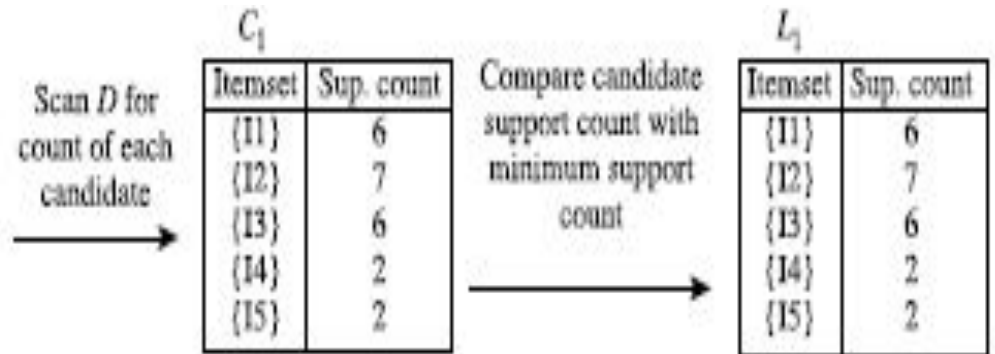
$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}$$

Confidence there are 2 transactions having B and C items, 2 times when customer purchased B and C, also took E. i.e. 100% chances are there of purchasing E with B and C

# Example of Apriori

Transactional Data for an *AlI*Electronics Branch

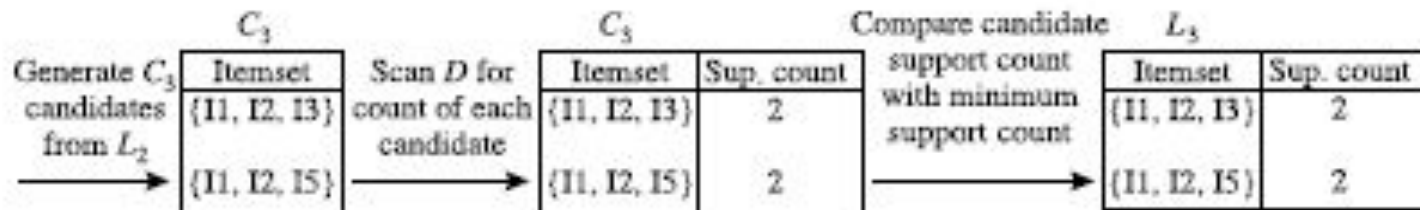
TID	List of Item_IDs
T100	11, 12, 15
T200	12, 14
T300	12, 13
T400	11, 12, 14
T500	11, 13
T600	12, 13
T700	11, 13
T800	11, 12, 13, 15
T900	11, 12, 13



$min\_sup=2$

(absolute support ,  
relative support =  $2/9=22\%$ )

# Example of Apriori



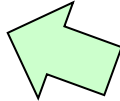
Consider  $x = [I1, I2, I5]$

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

Consider  $x = [I1, I2, I3]$

# 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

# 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

# Hash-based technique

- When scanning each transaction in the database to generate the frequent 1-itemsets,  $L_1$ , generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different buckets of a hash table structure, and increase the corresponding bucket counts

Transactional Data for an *AlI*Electronics Branch

TID	List of Item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Create hash table  $H_2$   
using hash function  
 $h(x, y) = ((\text{order of } x) \times 10 + (\text{order of } y)) \bmod 7$

$H_2$

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3}

- Hash table,  $H_2$ , for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining  $L_1$ . If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in  $C_2$ .

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

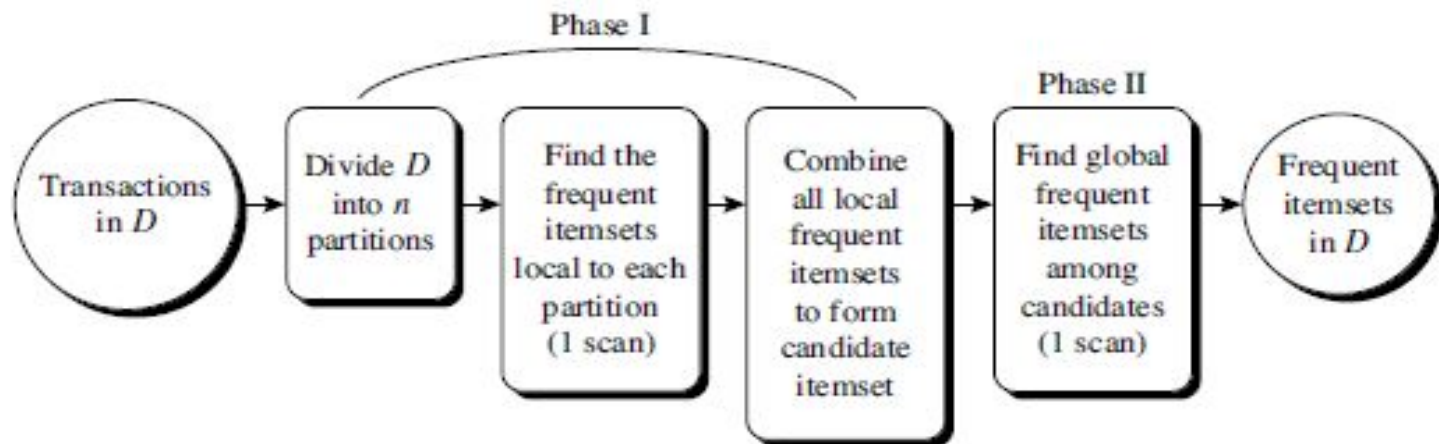
# Transaction reduction

---

- Reducing the number of transactions scanned in future iterations
- A transaction that does not contain any frequent  $k$ -itemsets *cannot contain any* frequent  $(k + 1)$ -itemsets.
- Such a transaction can be marked or removed from further consideration

# Partition: Scan Database Only Twice

- partitioning the data to find candidate itemsets
- 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





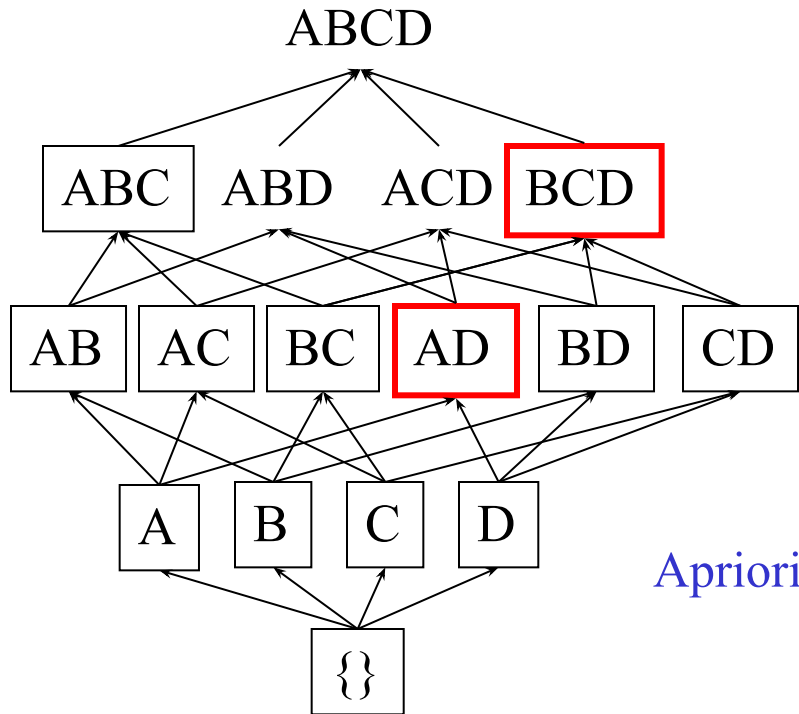
# Sampling for Frequent Patterns

---

- Select a sample of original database, mine frequent patterns within sample using Apriori
  - trade off some degree of accuracy against **efficiency**
  - possible that we will miss some of the global frequent itemsets
- use a lower support threshold than minimum support to find the frequent itemsets local to S
- Scan database again to find missed frequent patterns
- H. Toivonen. [Sampling large databases for association rules](#). In *VLDB'96*

# DIC: Reduce Number of Scans

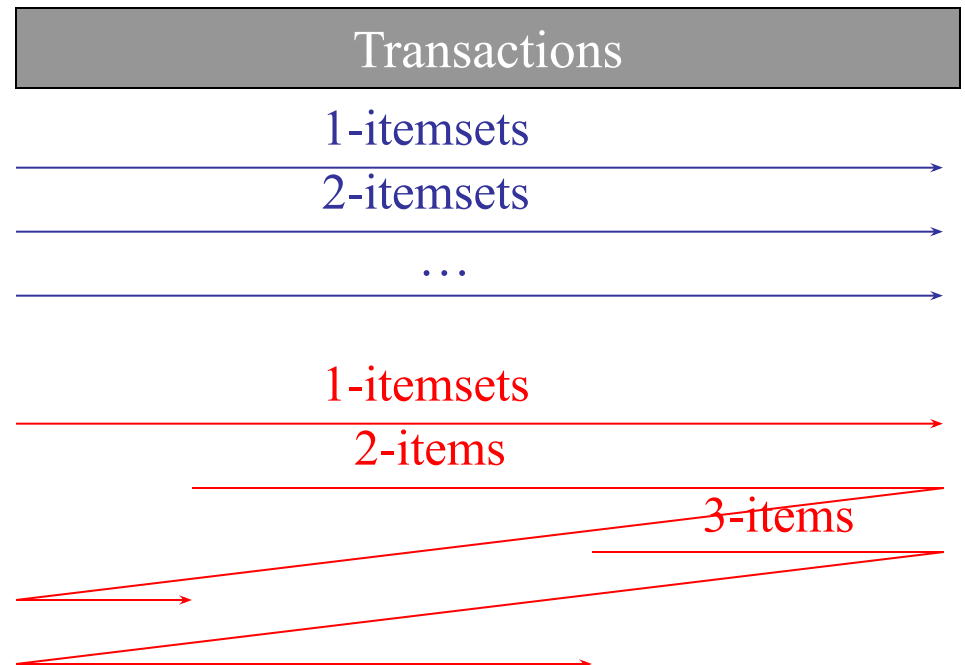
## Dynamic Itemset Counting



Itemset lattice

Apriori

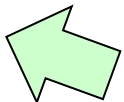
- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



S. Brin R. Motwani, J. Ullman, and S. Tsur. **DIC** 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

# 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  $\square$  abcd is a frequent pattern

# Construct FP-tree from a Transaction Database

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

*min\_support* = 3

Step 1 — List down the frequencies of the individual items in the dataset

Item	Frequency
f	4
a	3
c	4
d	1
g	1
i	1
m	3
p	3
b	3
l	2
o	2
h	1
j	1
k	1
s	1
e	1
n	1

# Construct FP-tree from a Transaction Database

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

***min\_support = 3***

Step 2 — Eliminate frequencies lower than minimum support and arrange the remaining frequencies in descending order

Item	Frequency
f	4
c	4
a	3
b	3
m	3
p	3

# Construct FP-tree from a Transaction Database

---

Step 3 — Order items in every transaction in frequency descending order and items with support lesser than minimum support eliminated.

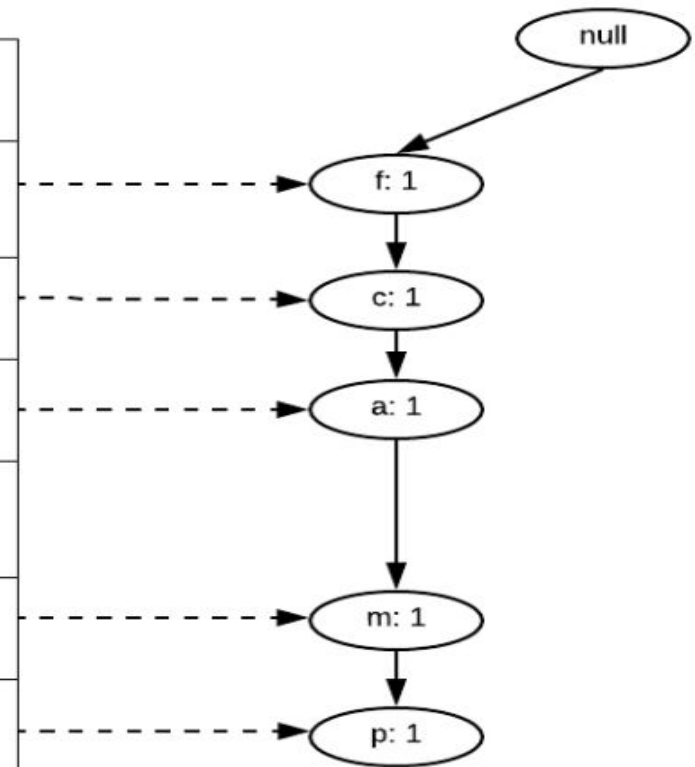
<i><b>TID</b></i>	<i><b>Items bought</b></i>	<i><b>(ordered) frequent items</b></i>
<b>100</b>	<b>{f, a, c, d, g, i, m, p}</b>	<b>{f, c, a, m, p}</b>
<b>200</b>	<b>{a, b, c, f, l, m, o}</b>	<b>{f, c, a, b, m}</b>
<b>300</b>	<b>{b, f, h, j, o, w}</b>	<b>{f, b}</b>
<b>400</b>	<b>{b, c, k, s, p}</b>	<b>{c,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

# Construct FP-tree from a Transaction Database

Step 4 — Use the *Frequent Items ordered* column to build FP-Tree.  
initialize our FP-Tree with a root node represented by *null*  
Iterate transaction by transaction.

<u>TID</u>	<u>(ordered) frequent items</u>
100	{f, c, a, m, p}
200	{f, c, a, b, m}
300	{f, b}
400	{c, b, p}
500	{f, c, a, m, p}

Item	Frequency	Node Link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



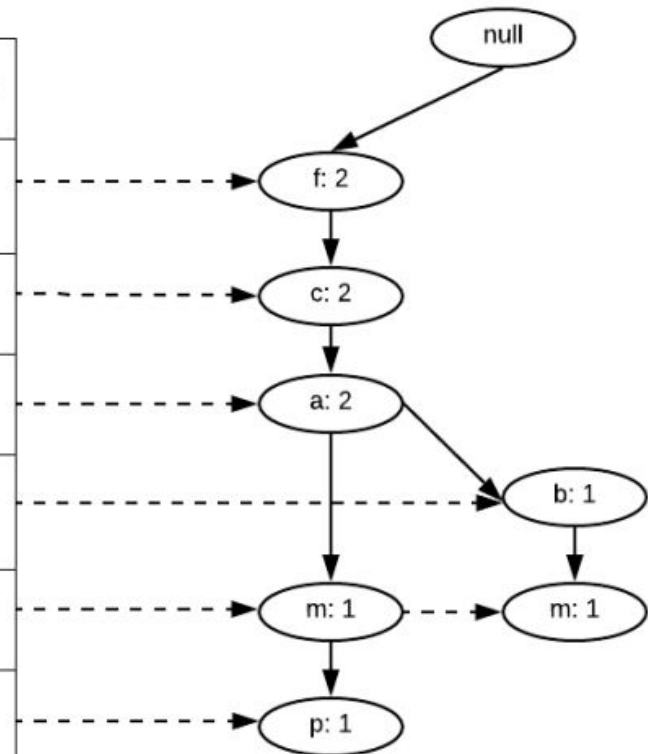


# Construct FP-tree from a Transaction Database

Step 5 — After the first transaction move onto next transaction in the dataset

<u>TID</u>	<u>(ordered) frequent items</u>
100	{f, c, a, m, p}
200	{f, c, a, b, m}
300	{f, b}
400	{c, b, p}
500	{f, c, a, m, p}

Item	Frequency	Node Link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

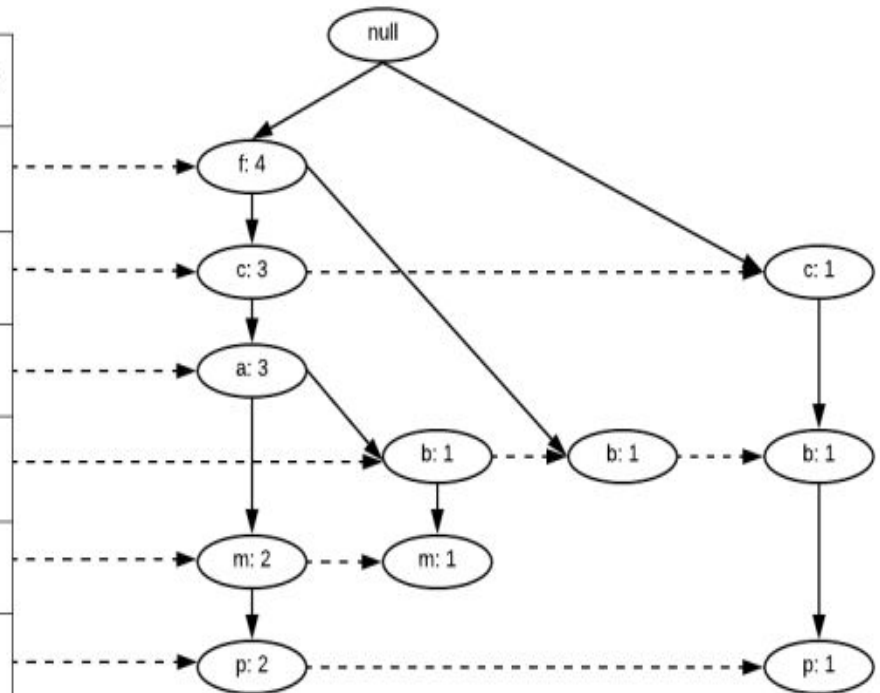


# Construct FP-tree from a Transaction Database

Step 6 — repeat this process for all the transactions in the dataset  
after traversing the whole dataset, FP-Tree and header table as shown in the below figure

<u>TID</u>	<u>(ordered) frequent items</u>
100	{f, c, a, m, p}
200	{f, c, a, b, m}
300	{f, b}
400	{c, b, p}
500	{f, c, a, m, p}

Item	Frequency	Node Link
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

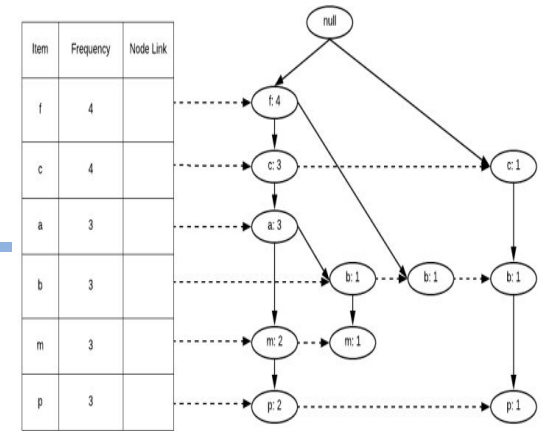


# Construct FP-tree from a Transaction Database

---

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
  - First, create the root of the tree, labeled with “null.”
  - A branch is created for each transaction

# Find Frequent Patterns



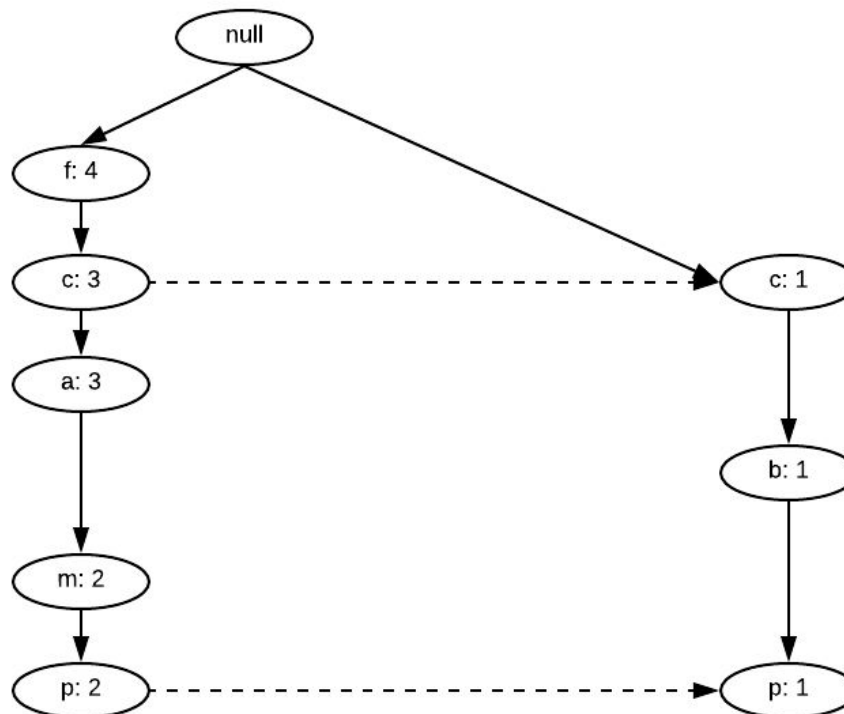
- FP-growth algorithm generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion
- The algorithm will iterate the header table in a reverse order.
- First it will find all the frequent items ending in **p**, then **m**, **b**, **a**, **c**, and finally, **f**
- Derive the frequent itemsets ending with a particular item, say **p**, by examining only the paths containing node **p**. These paths can be accessed rapidly using the pointers associated with node **p**
- FP-growth finds all the frequent itemsets ending with a particular suffix by employing a divide-and-conquer strategy to split the problem into smaller subproblems.
- For example, suppose we are interested in finding all frequent itemsets ending in **p**. First check whether the itemset **{p}** itself is frequent. If it is frequent, consider the subproblem of finding frequent itemsets ending in **mp**, followed by **bp**, **ap**, **cp**, and **fp**. In turn, each of these subproblems is further decomposed into smaller subproblems. By merging the solutions obtained from the subproblems, all the frequent itemsets ending in **p** can be found

# Find Frequent Patterns

**Step 1** — Header table would be iterated in a reverse order, so first, those frequent itemsets would be searched for which end with the item **p**.

Gather all the paths ending in node **p**.

These paths are known as the **prefix paths**. The below figure shows the prefix paths for node **p**.



# Find Frequent Patterns

**Step 2** — Update the support count of the nodes to represent those paths which contain node **p**.

For example, {**f**: 4, **c**: 3, **a**: 3, **m**: 2, **p**: 2} contains many paths without node **p** like {**f**, **c**, **a**}, so we have to update the support counts.

Add the support count of node **p** to all of its parent nodes till the root node. *fcam:2, cb:1*

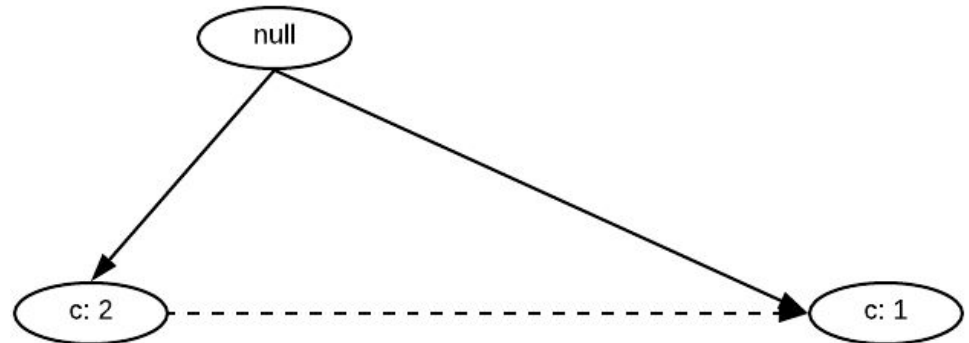
Once the paths are updated with new support counts, eliminate all those items whose support count is less than the minimum support count, in this case, 3.

The support count of items will be calculated by adding all the support counts of nodes containing that item in the prefix paths. *fam:2, c:3*

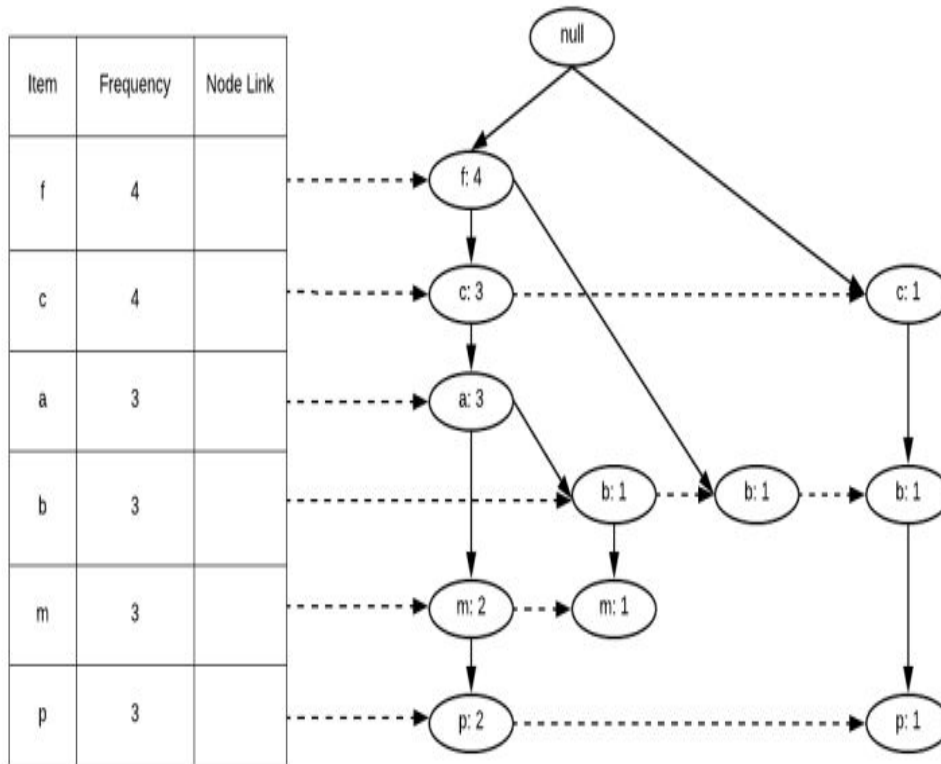
Conditional FP-Tree for node **p**.

From conditional FP-Tree, {**c**, **p**} frequent itemset

{**p** - 3}, {**c**, **p** - 3}



# From Conditional Pattern-bases to Conditional FP-trees



***Conditional pattern bases***

***itemcond. pattern base***

***c f:3***

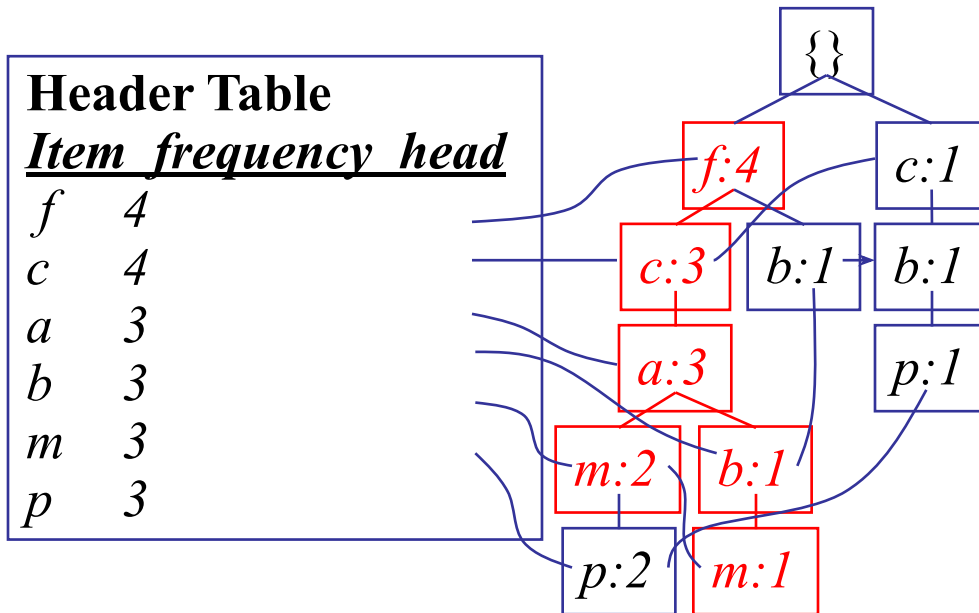
***a fc:3***

***b fca:1, f:1, c:1***

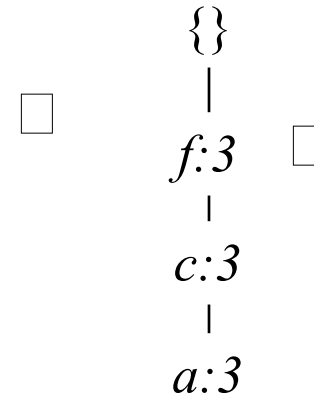
***m fca:2, fcab:1***

# 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



*m*-conditional pattern base:  
 $fca:2, fcab:1 \Rightarrow fca:3 \ b:1$

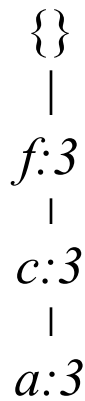


All frequent patterns relate to *m*  
 $\{m - 3\}, \{f, m - 3\},$   
 $\{c, m - 3\}, \{a, m - 3\},$   
 $\{f, a, m - 3\},$   
 $\{c, a, m - 3\}$   
 $\{c, f, m - 3\}$   
 $\{c, f, a, m - 3\}$

*m*-conditional FP-tree

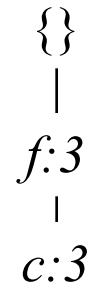


# Recursion: Mining Each Conditional FP-tree



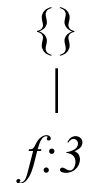
*m*-conditional FP-tree

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



*am*-conditional FP-tree

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



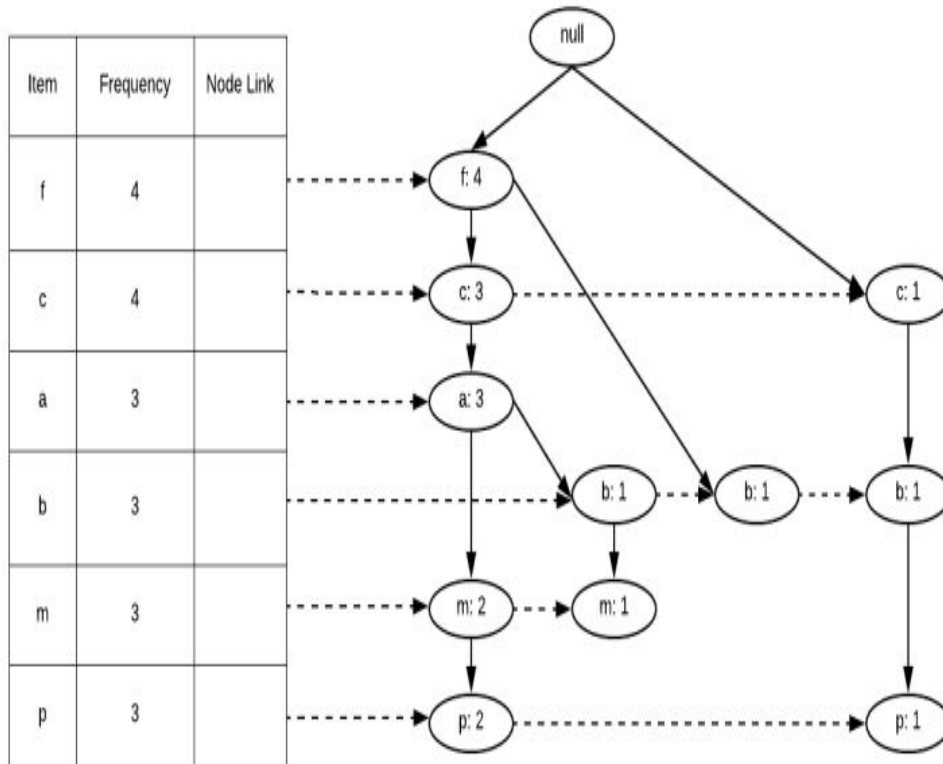
*cm*-conditional FP-tree

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



*cam*-conditional FP-tree

# From Conditional Pattern-bases to Conditional FP-trees



**{b - 3}**

**{a - 3}, {f, a - 3}, {c, a - 3},  
{c, f, a - 3}**

**{f - 4}, {c, f - 3}**

**{c - 4}**

# FP Growth

Transactional Data for an *AllElectronics* Branch

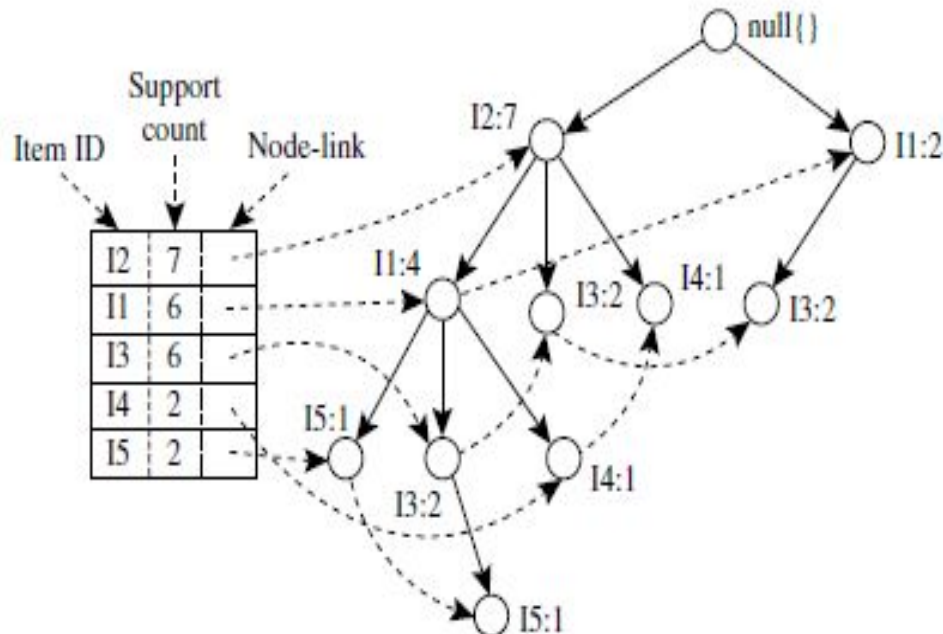
TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

T100	I2, I1, I5
T200	I2, I4
T300	I2, I3
T400	I2, I1, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I2, I1, I3, I5
T900	I2, I1, I3

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

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

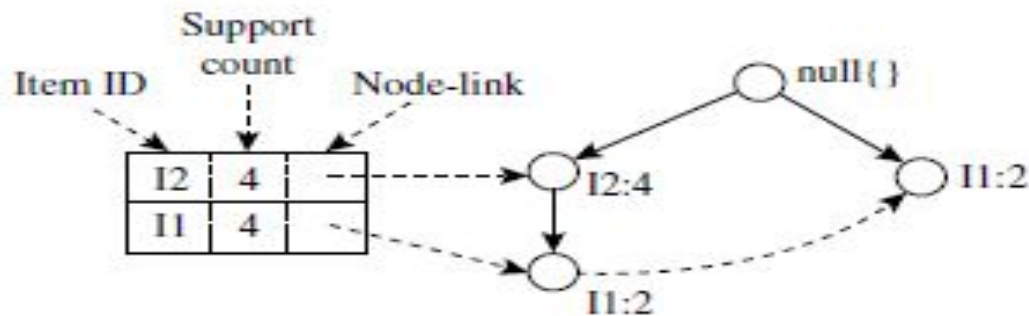
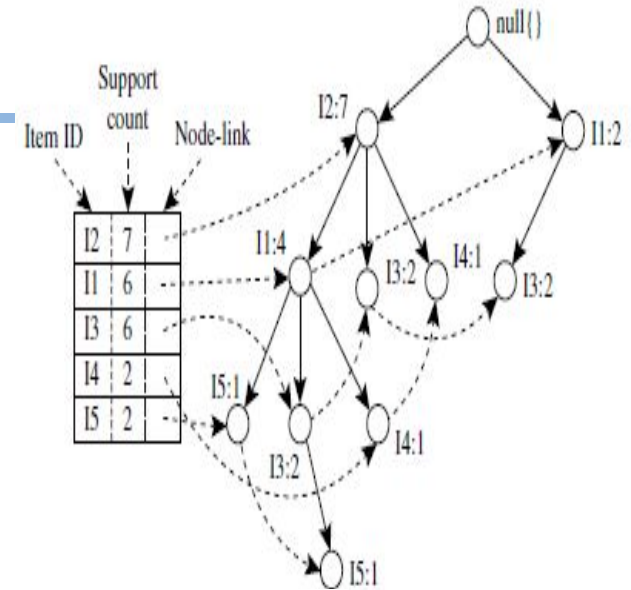
I2	7
I1	6
I3	6
I4	2
I5	2



# FP Growth

Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	$\langle I2: 2, I1: 2 \rangle$	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	$\langle I2: 2 \rangle$	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	$\langle I2: 4, I1: 2 \rangle, \langle I1: 2 \rangle$	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	$\langle I2: 4 \rangle$	{I2, I1: 4}



The conditional FP-tree associated with the conditional node I3.

# 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 Method

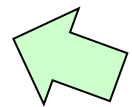
---

- 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

# Scalable Frequent Itemset Mining Methods

---

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



Eclat (Equivalence Class Transformation)

# Mining by Exploring Vertical Data Format

Transactional Data for an *Allelectronics* Branch

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

The Vertical Data Format  
of the Transaction Data  
[one scan of data set]

minimum support count = 2

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

Perform mining on this data set by  
intersecting the TID sets of every  
pair of frequent single items.  
[10 intersections]

2-Itemsets in Vertical Data Format

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

Generate 3-itemset

3-Itemsets in Vertical Data Format

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

Eliminate {I1,I4}  
& {I3,I5}



# Mining by Exploring Vertical Data Format

---

- Given are the following six transactions on items {A;B;C;D;E}
- Use the Apriori algorithm to compute all frequent item sets, and their support, with minimum support 2. Clearly indicate the steps of the algorithm, and the pruning that is performed.

tid	items
1	<i>A, B, C</i>
2	<i>A, B, C</i>
3	<i>B, C</i>
4	<i>B, C, D</i>
5	<i>B, C, D, E</i>
6	<i>E</i>

# Mining by Exploring Vertical Data Format

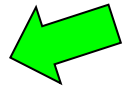
---

- Vertical format:  $t(AB) = \{T_{11}, T_{25}, \dots\}$ 
  - 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 [Equivalence Class Transformation] (Zaki et al. @KDD'97)
- Mining Closed patterns using vertical format: CHARM (Zaki & Hsiao@SDM'02)

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

---

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



# Pattern Evaluation Methods

- minimum support and confidence thresholds *help weed out* or exclude the exploration of a good number of uninteresting rules
- *when mining at low support thresholds or mining for long patterns*; many of the rules generated are not interesting to the users

## A misleading “strong” association rule

Total transactions	10000
Computer games	6000
Videos	7500
Both	4000

minimum support=30%

minimum confidence=60%.

*buys computer games* → *buys videos*

[*support* = 40%(4000/10000), *confidence* = 66%(4000/6000)]

Misleading rule because the probability of purchasing videos is 75%, which is even larger than 66%.

In fact, computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other.

# Interestingness Measure: Correlations (Lift)

- The support and confidence measures are insufficient at filtering out uninteresting association rules
- To tackle this weakness, a correlation measure can be used to augment the support–confidence framework for association rules

$$A \Rightarrow B \text{ [support, confidence, correlation]}$$

- **Lift is a simple correlation measure**

- The occurrence of itemset  $A$  is *independent* of the occurrence of itemset  $B$  if  $P(A \cup B) = P(A)P(B)$ ;
- otherwise, itemsets  $A$  and  $B$  are *dependent* and correlated as events

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

# Interestingness Measure: Correlations (Lift)

---

- **Lift is a simple correlation measure**

- $< 1$ , then the occurrence of  $A$  is ***negatively correlated*** with the occurrence of  $B$ , meaning that the occurrence of one likely leads to the absence of the other one.
- $> 1$ , then  $A$  and  $B$  are ***positively correlated***, meaning that the occurrence of one implies the occurrence of the other
- $= 1$ , then  $A$  and  $B$  are *independent* and there is no correlation between them

It assesses the degree to which the occurrence of one “lifts” the occurrence of the other

Given the current market conditions, the sale of games is said to increase or “lift” the likelihood of the sale of videos by a factor of the value returned formula.

# Interestingness Measure: Correlations (Lift)

2 × 2 Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

	game	$\overline{\text{game}}$	$\Sigma_{\text{row}}$
video	4000	3500	7500
$\overline{\text{video}}$	2000	500	2500
$\Sigma_{\text{col}}$	6000	4000	10,000

$\overline{\text{game}}$  refer to the transactions that do not contain computer games

$\overline{\text{video}}$  refer to the transactions that do not contain videos

$$P(\{\text{game}\}) = 0.60 \text{ and } P(\{\text{video}\}) = 0.75$$

$$P(\{\text{game}, \text{video}\}) = 0.40$$

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

$$\begin{aligned} \text{Lift} &= P(\{\text{game}, \text{video}\}) / P(\{\text{game}\}) P(\{\text{video}\}) \\ &= 0.40 / 0.60 * 0.75 = 0.89 \end{aligned}$$

<1, there is a negative correlation between the occurrence of {game} and {video}

The **numerator** is the likelihood of a customer **purchasing both**, while the **denominator** is what the likelihood would have been if the **two purchases were completely independent**

# Interestingness Measure: Correlations (Lift)

- *play basketball*  $\Rightarrow$  *eat cereal* [40%, 66.7%]
- *play basketball*  $\Rightarrow$  *not eat cereal* [20%, 33.3%]

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

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

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

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



# Interestingness Measure: $\chi^2$

Contingency Table, Now with the Expected Values

	game	$\overline{\text{game}}$	$\Sigma_{\text{row}}$
video	4000 (4500)	3500 (3000)	7500
$\overline{\text{video}}$	2000 (1500)	500 (1000)	2500
$\Sigma_{\text{col}}$	6000	4000	10,000

Expected frequency calculation

$$e_{ij} = \frac{\text{count}(A = a_i) \times \text{count}(B = b_j)}{N},$$

$$\chi^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

$$= \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000}$$

$$= 555.6.$$

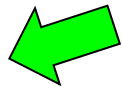
$\chi^2 > 1$ , and the observed value of the slot (*game*, *video*) = 4000, which is less than the expected value of 4500,  
*buying game and buying video are **negatively correlated**.*

*Consistent with the conclusion derived from the analysis of the lift*

# 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