



# Chapter 6. Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

Meng Jiang

CS412 Summer 2017:

Introduction to Data Mining



Data warehouse (OLAP) vs operational database (OLTP)

Transactions = tuples = samples = instances...

Data cube, data cuboid, cells

Dimension, dimension levels, dimension values

Ex. 3 transactions: (9/15/16, Urbana), (9/15/16, Urbana), (9/15/16, Chicago) → data warehouse. Data cubes. One of cubes = sales

Cube: dimension tables and fact tables (measures).

Dimension = ("time", "location")

concept hierarchy of each dimension

time: year-month-date (dimension levels) vs dimension values

location: country-state-city (dimension levels)

One measure = count.

Two base cells: (9/15/16, Urbana): 2, (9/15/16, Chicago)

Base cuboids: (date, city) → (month, city)... not empty

Aggregate cuboids: (\*, city), (year, \*); Apex cuboid: (\*, \*)

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

- **Basic Concepts**
- Frequent Pattern (Itemset) Mining Methods
- Pattern Evaluation Methods

# Pattern Discovery: Definition

- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets
- Pattern discovery: Uncovering patterns from massive data
- Motivation examples:
  - What products were often purchased together?
  - What are the subsequent purchases after buying an iPad?
  - What code segments likely contain copy-and-paste bugs?
  - What word sequences likely form phrases in this corpus?

# Pattern Discovery: Why Is It Important?

- Finding inherent regularities in a data set
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering
- Broad applications
  - Market basket analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, biological sequence analysis

# Frequent Patterns (Itemsets)

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

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

Let *minsup* = 50%

Freq. 1-itemsets:

Beer: 3 (60%); Nuts: 3 (60%)

Diaper: 4 (80%); Eggs: 3 (60%)

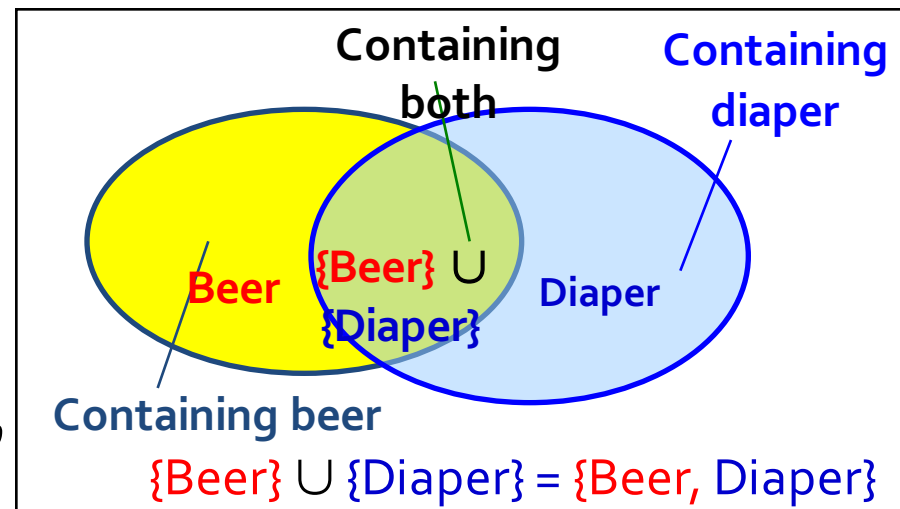
Freq. 2-itemsets:

{Beer, Diaper}: 3 (60%)

# From Frequent Itemsets to Association Rules

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

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



Note: Itemset:  $X \cup Y$ , a subtle notation!

# Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
  - How many frequent itemsets does the following  $TDB_1$  contain?
    - $TDB_1: T_1: \{a_1, \dots, a_{50}\}; T_2: \{a_1, \dots, a_{100}\}$
    - Assuming (absolute)  $minsup = 1$
    - Let's have a try
- 1-itemsets:  $\{a_1\}: 2, \{a_2\}: 2, \dots, \{a_{50}\}: 2, \{a_{51}\}: 1, \dots, \{a_{100}\}: 1,$
- 2-itemsets:  $\{a_1, a_2\}: 2, \dots, \{a_1, a_{50}\}: 2, \{a_1, a_{51}\}: 1 \dots, \dots, \{a_{99}, a_{100}\}: 1, \dots$
- 99-itemsets:  $\{a_1, a_2, \dots, a_{99}\}: 1, \dots, \{a_2, a_3, \dots, a_{100}\}: 1$
- 100-itemset:  $\{a_1, a_2, \dots, a_{100}\}: 1$
- In total:  $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$  sub-patterns!

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



# Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns**: A pattern (itemset)  $X$  is **closed** if  $X$  is *frequent*, and there exists *no super-pattern*  $Y \supset X$ , **with the same support as  $X$** 
  - Let Transaction DB  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does  $TDB_1$  contain?
    - Two:  $P_1: "\{a_1, \dots, a_{50}\}: 2"$ ;  $P_2: "\{a_1, \dots, a_{100}\}: 1"$
- **Closed pattern** is a **lossless compression** of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say:  $"\{a_2, \dots, a_{40}\}: 2"$ ,  $"\{a_5, a_{51}\}: 1"$

# Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns**: A pattern  $X$  is a **max-pattern** if  $X$  is frequent and there exists no frequent super-pattern  $Y \supset X$ , ~~with the same support as  $X$~~
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$
  - Suppose  $minsup = 1$ . How many max-patterns does  $TDB_1$  contain?
    - One:  $P: \{a_1, \dots, a_{100}\}: 1$
- **Max-pattern** is a **lossy compression**!
  - We only know  $\{a_1, \dots, a_{40}\}$  is frequent
  - But we do not know the real support of  $\{a_1, \dots, a_{40}\}$ , ..., any more!
- Thus in many applications, mining closed-patterns is more desirable than mining max-patterns

- Transaction, itemset
- Pattern (itemset), min\_sup (abs. rel.)
- Frequent pattern (itemset)
- Association rules: item  $\rightarrow$  item (sup, conf.)
  - Min\_sup, min\_conf
- Number of frequent patterns: huge
- Compression:
  - Closed pattern (lossless)
  - Max pattern (lossy)

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

- Basic Concepts
- **Frequent Pattern (Itemset) Mining Methods**
- Pattern Evaluation Methods

# Efficient Pattern Mining Methods

- The Downward Closure Property of Frequent Patterns
- **The Apriori Algorithm**
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- **FPGrowth: A Frequent Pattern-Growth Approach**



# The Downward Closure Property of Frequent Patterns

- Observation: From  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$ 
  - We get a frequent itemset:  $\{a_1, \dots, a_{50}\}$
  - Also, its subsets are all frequent:  $\{a_1\}, \{a_2\}, \dots, \{a_{50}\}, \{a_1, a_2\}, \dots, \{a_1, \dots, a_{49}\}, \dots$
  - There must be some hidden relationships among frequent patterns!
- The **downward closure (also called “Apriori”)** property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - **Apriori: Any subset of a frequent itemset must be frequent**
- Efficient mining methodology
  - If **any subset of an itemset S** is infrequent, then there is no chance for S to be frequent—why do we even have to consider S!?

*A sharp knife for pruning!*

# Apriori Pruning and Scalable Mining Methods

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not even be generated! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Scalable mining Methods: Three major approaches
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li @KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)

# Apriori: A Candidate Generation & Test Approach

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

# The Apriori Algorithm (Pseudo-Code)

$C_k$ : Candidate itemset of size  $k$

$F_k$ : Frequent itemset of size  $k$

$K := 1$ ;

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

**While** ( $F_k \neq \emptyset$ ) **do** { // when  $F_k$  is non-empty

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

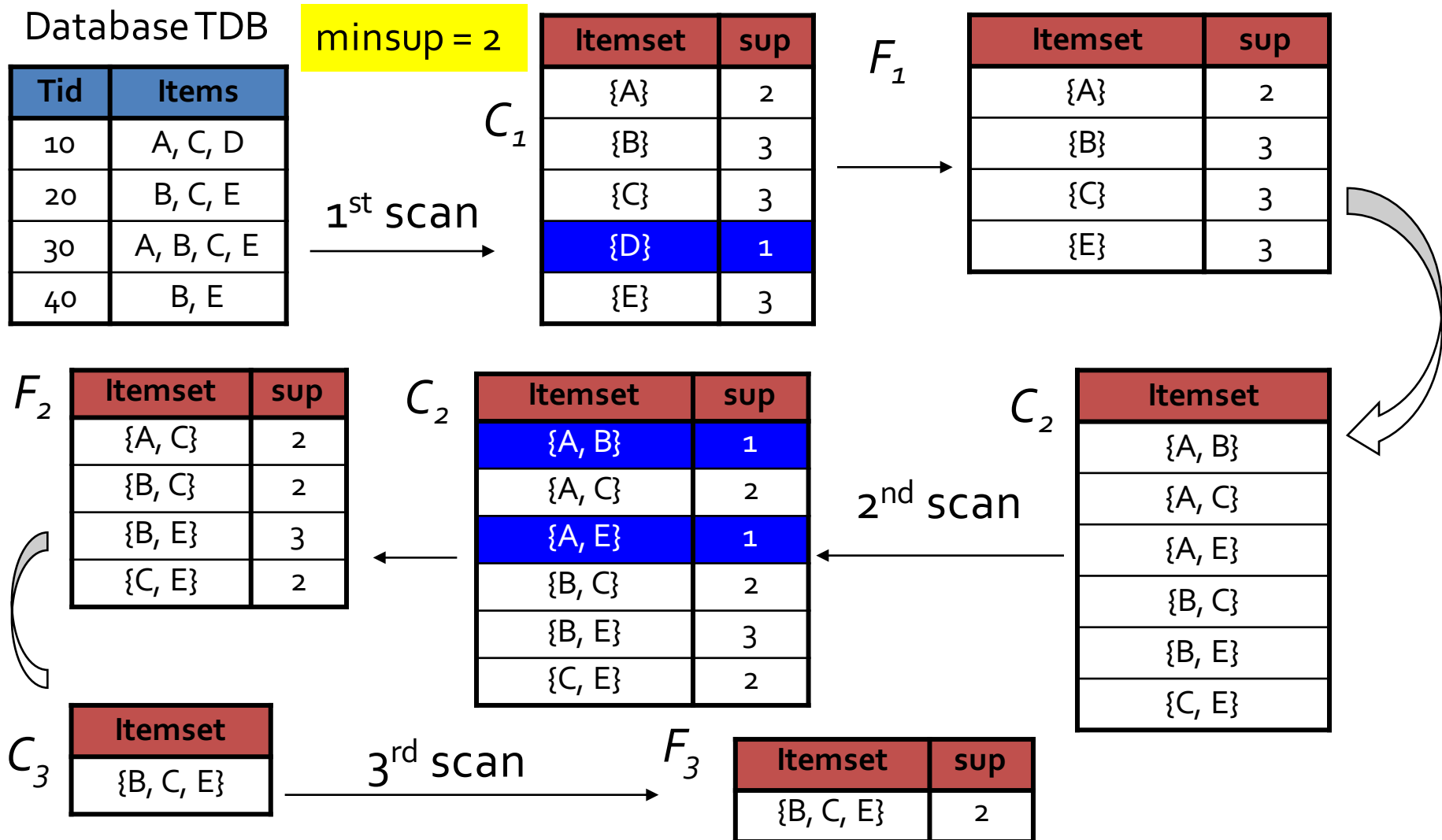
    Derive  $F_{k+1}$  by counting candidates in  $C_{k+1}$  with respect to  $TDB$  at  
    minsup;

$k := k + 1$

}

**return**  $\bigcup_k F_k$  // return  $F_k$  generated at each level

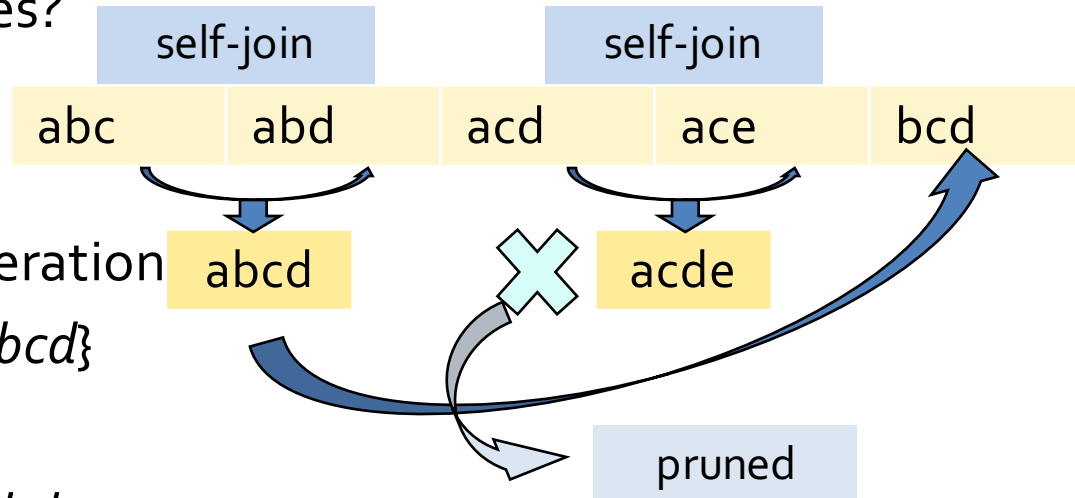
# The Apriori Algorithm: An Example





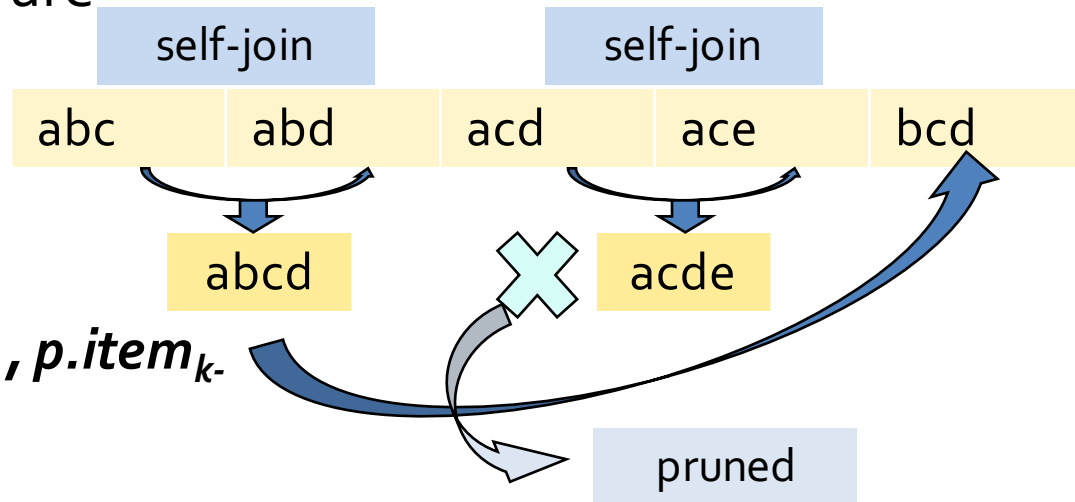
# Apriori: Implementation Tricks

- How to generate candidates?
  - Step 1: self-joining  $F_k$
  - Step 2: pruning
- Example of candidate-generation
  - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining:  $F_3 * F_3$ 
    - $abcd$  from  $abc$  and  $abd$
    - $acde$  from  $acd$  and  $ace$
  - Pruning:
    - $acde$  is removed because  $ade$  is not in  $F_3$
  - $C_4 = \{abcd\}$



# Candidate Generation: An SQL Implementation

- Suppose the items in  $F_{k-1}$  are listed in an order
- Step 1: self-joining  $F_{k-1}$   
insert into  $C_k$   
select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$   
from  $F_{k-1}$  as  $p, F_{k-1}$  as  $q$   
where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning  
for all *itemsets*  $c$  in  $C_k$  do  
for all  $(k-1)$ -subsets  $s$  of  $c$  do  
if ( $s$  is not in  $F_{k-1}$ ) then delete  $c$   
from  $C_k$



# Apriori: Improvements and Alternatives

- Reduce passes of transaction database scans
  - Partitioning (e.g., Savasere, et al., 1995)
  - Dynamic itemset counting (Brin, et al., 1997)
- Shrink the number of candidates
  - Hashing (e.g., DHP: Park, et al., 1995)
  - Pruning by support lower bounding (e.g., Bayardo 1998)
  - Sampling (e.g., Toivonen, 1996)
- Exploring special data structures
  - Tree projection (Agarwal, et al., 2001)
  - H-miner (Pei, et al., 2001)
  - Hypercube decomposition (e.g., LCM: Uno, et al., 2004)

# Partitioning: Scan Database Only Twice

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



- Method: (A. Savasere, E. Omiecinski and S. Navathe, VLDB'95)
  - Scan 1: Partition database (how?) and find local frequent patterns
  - Scan 2: Consolidate global frequent patterns (how to?)
- Why does this method guarantee to scan TDB only twice?

# Direct Hashing and Pruning (DHP)

- DHP (Direct Hashing and Pruning): Reduce the number of candidates (J. Park, M. Chen, and P. Yu, SIGMOD'95)
- Observation: 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

Itemsets	Count
{ab, ad, ae}	35
{bd, be, de}	298
.....	...
{yz, qs, wt}	58

**Hash Table**



# Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- Tid-List: List of transaction-ids containing an itemset
- Vertical format:  $t(e) = \{T_{10}, T_{20}, T_{30}\}$ ;  $t(a) = \{T_{10}, T_{20}\}$ ;  $t(ae) = \{T_{10}, T_{20}\}$

**A transaction DB in Horizontal  
Data Format**

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

**The transaction DB in Vertical  
Data Format**

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

# Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- Tid-List: List of transaction-ids containing an itemset
- Vertical format:  $t(e) = \{T_{10}, T_{20}, T_{30}\}$ ;  $t(a) = \{T_{10}, T_{20}\}$ ;  $t(ac) = \{T_{10}, T_{20}\}$
- Properties of Tid-Lists
  - $t(X) = t(Y)$ : X and Y always happen together (e.g.,  $t(ac) = t(d)$ )
  - $t(X) \subset t(Y)$ : transaction having X always has Y (e.g.,  $t(ac) \subset t(ce)$ )
- **Deriving frequent patterns based on vertical intersections**

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

# Diffset Based Mining

- ECLAT: Using **diffset** to accelerate mining
  - Only keep track of differences of tids

P: itemset; X, Y: items

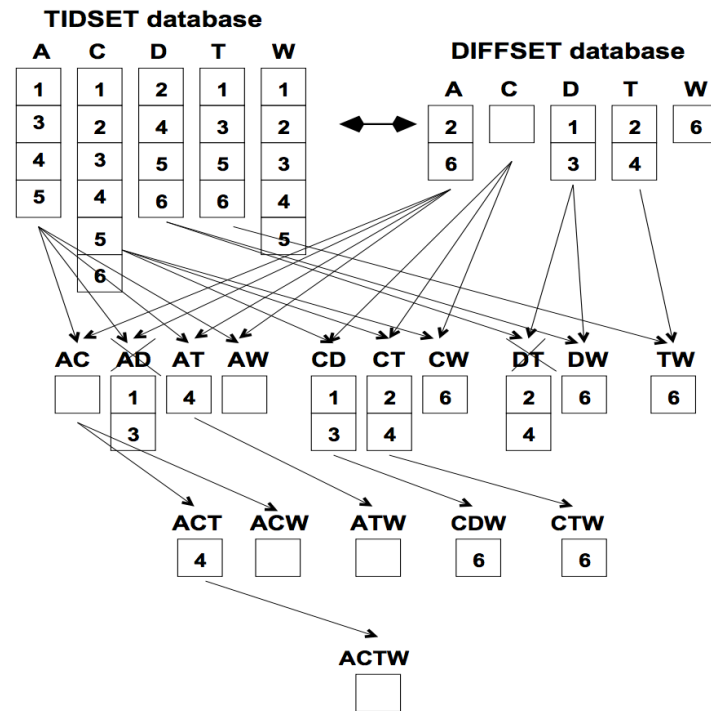
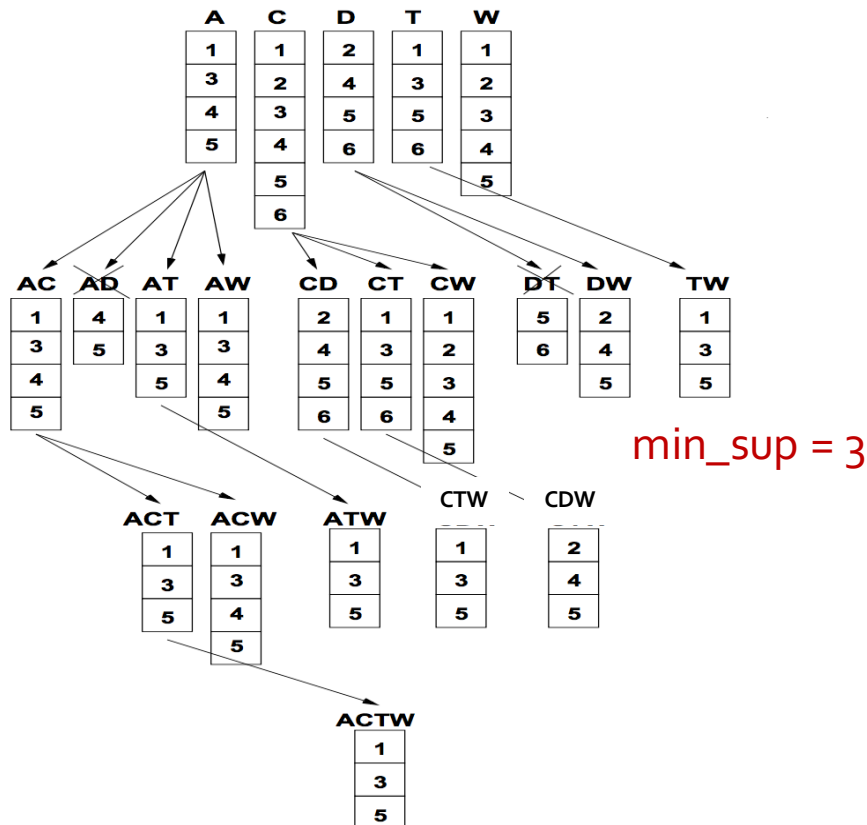
$$d(XY) = t(X) - t(Y)$$

$$= d(Y) - d(X)$$

$$d(PXY) = t(PX) - t(PY)$$

$$= d(PY) - d(PX)$$

$$\sigma(PX) = \sigma(P) - |d(PX)|$$



# FPGrowth: Mining Frequent Patterns by Pattern Growth

- Idea: Frequent pattern growth (FPGrowth)
  - Find frequent single items and partition the database based on each such item
  - Recursively grow frequent patterns by doing the above for each partitioned database (also called *conditional database*)
  - To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- Mining becomes
  - Recursively construct and mine (conditional) FP-trees
  - Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

# Example: Construct FP-tree from a Transactional DB

TID	Items in the Transaction	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}

Answer:

f:4, a:3, c:4, b:3, m:3, p:3;  
 fm: 3, cm: 3, am: 3, cp:3;  
 fcm: 3, fam:3, cam: 3;  
 fcam: 3.

1. Scan DB once, find single item frequent pattern:

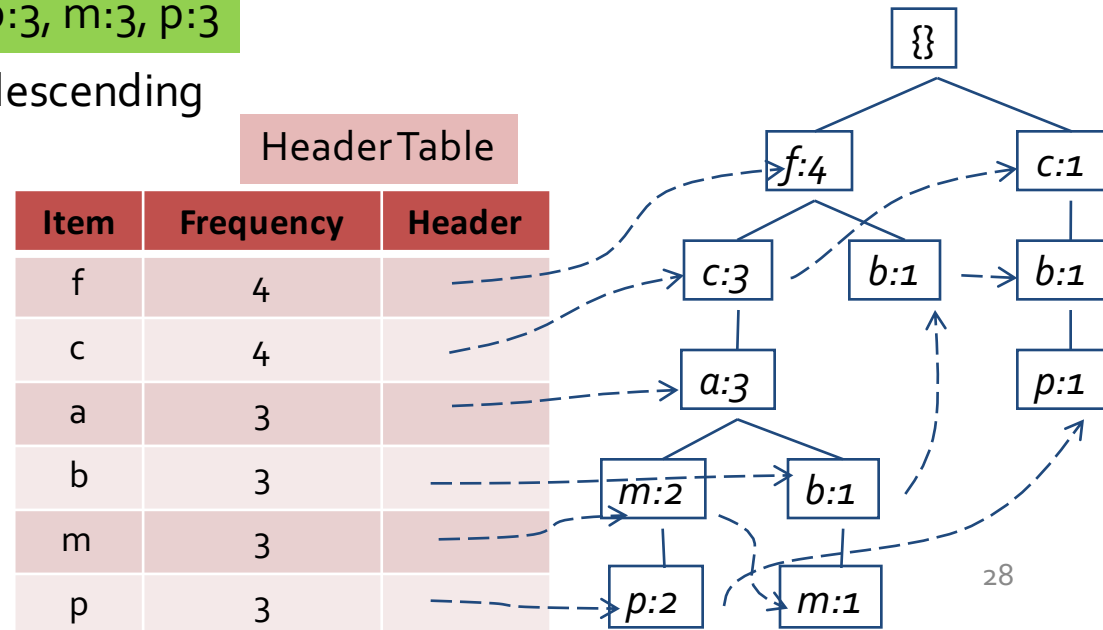
Let min\_support = 3

f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

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

3. Scan DB again, construct FP-tree



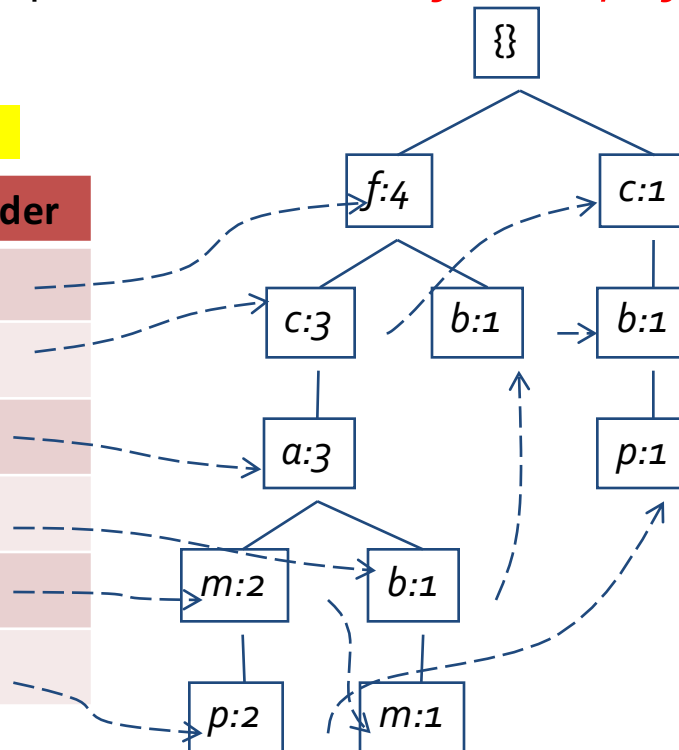


# Divide and Conquer Based on Patterns and Data

- Pattern mining can be partitioned according to current patterns
  - Patterns containing  $p$ :  $p$ 's conditional database:  $fcam:2, cb:1$
  - Patterns having  $m$  but no  $p$ :  $m$ 's conditional database:  $fca:2, fcab:1$
  - .....
- $p$ 's conditional pattern base: *transformed prefix paths* of item  $p$

min\_support = 3

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



**Conditional pattern bases**

Item	Conditional pattern base
c	$f:3$
a	$fc:3$
b	$fca:1, f:1, c:1$
m	$fca:2, fcab:1$
p	$fcam:2, cb:1$

# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

item cond. pattern base

<i>c</i>	<i>f:3</i>
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

min\_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & mine it

*p*-conditional PB: *fcam:2, cb:1* → *c: 3*

*m*-conditional PB: *fca:2, fcab:1* → *fca: 3*

*b*-conditional PB: *fca:1, f:1, c:1* →  $\phi$

*a*-conditional PB: *fc:3* → *fc:3*

*c*-conditional PB: *f:3* → *f:3*

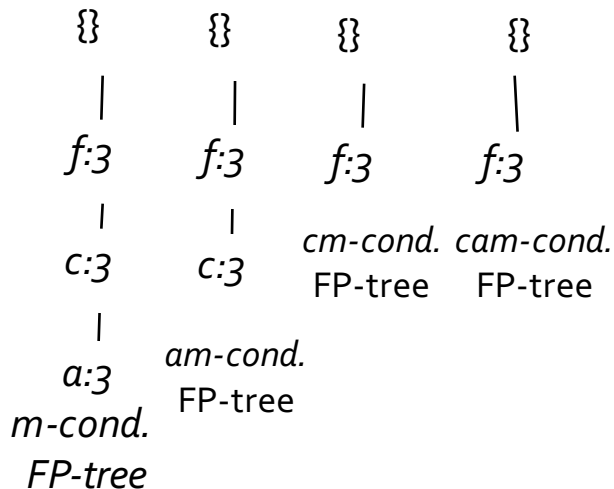
# Mine Each Conditional Pattern-Base Recursively

## 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

min\_support = 3



For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

*p*-conditional PB:  $fcam:2, cb:1 \rightarrow c:3$

*m*-conditional PB:  $fca:2, fcab:1 \rightarrow fca:3$

*b*-conditional PB:  $fca:1, f:1, c:1 \rightarrow \phi$

*a*-conditional PB:  $fc:3 \rightarrow fc:3$

*c*-conditional PB:  $f:3 \rightarrow f:3$

mine( $\langle f:3, c:3, a:3 \rangle | m$ )

$\rightarrow (am:3) + \text{mine}(\langle f:3, c:3 \rangle | am)$

$\rightarrow (cam:3) + (fam:3) + \text{mine}(\langle f:3 \rangle | cam)$

$\rightarrow (fcam:3)$

$\rightarrow (cm:3) + \text{mine}(\langle f:3 \rangle | cm)$

$\rightarrow (fcm:3)$

$\rightarrow (fm:3)$

# Mine Each Conditional Pattern-Base Recursively

## 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

min\_support = 3

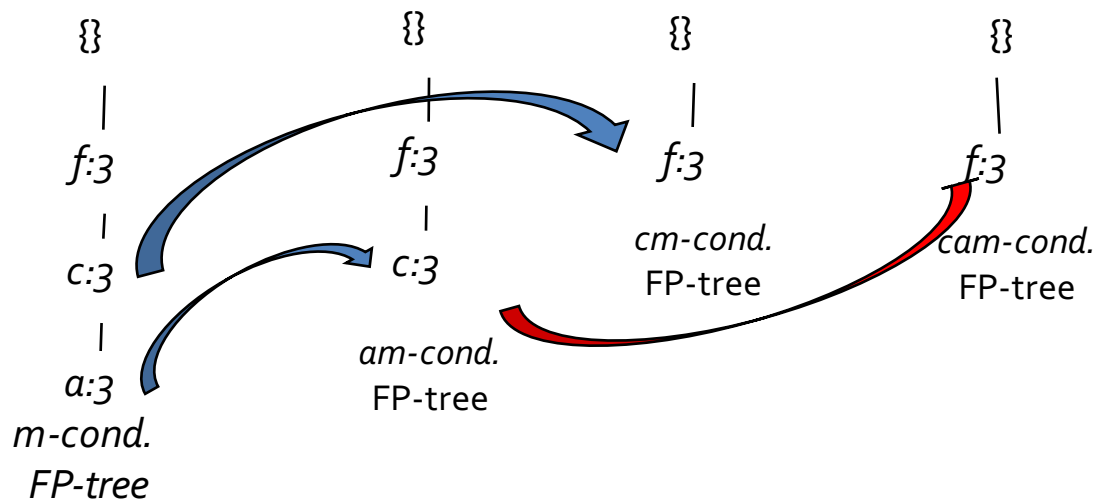
For each conditional pattern-base

- Mine single-item patterns
- Construct its cond. FP-tree & mine it

*p*-conditional PB: *fcam*:2, *cb*:1 → *c*: 3

*m*-conditional PB: *fca*:2, *fcab*:1 → *fca*: 3

*b*-conditional PB: *fca*:1, *f*:1, *c*:1 →  $\phi$



Actually, for single branch FP-tree, all frequent patterns can be generated in one shot

***m*: 3**  
***fm*: 3, *cm*: 3, *am*: 3**  
***fcm*: 3, *fam*: 3, *cam*: 3**  
***fcam*: 3**

# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

item cond. pattern base

<i>c</i>	<i>f:3</i>
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

min\_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

*p*-conditional PB: *fcam:2, cb:1* → *c:3*

*m*-conditional PB: *fca:2, fcab:1* → *fca:3*

*b*-conditional PB: *fca:1, f:1, c:1* →  $\phi$

*a*-conditional PB: *fc:3* → *fc:3*

*c*-conditional PB: *f:3* → *f:3*

# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

item cond. pattern base

<i>c</i>	<i>f:3</i>
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

min\_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

*p*-conditional PB: *fcam:2, cb:1* → *c:3*

*m*-conditional PB: *fca:2, fcab:1* → *fca:3*

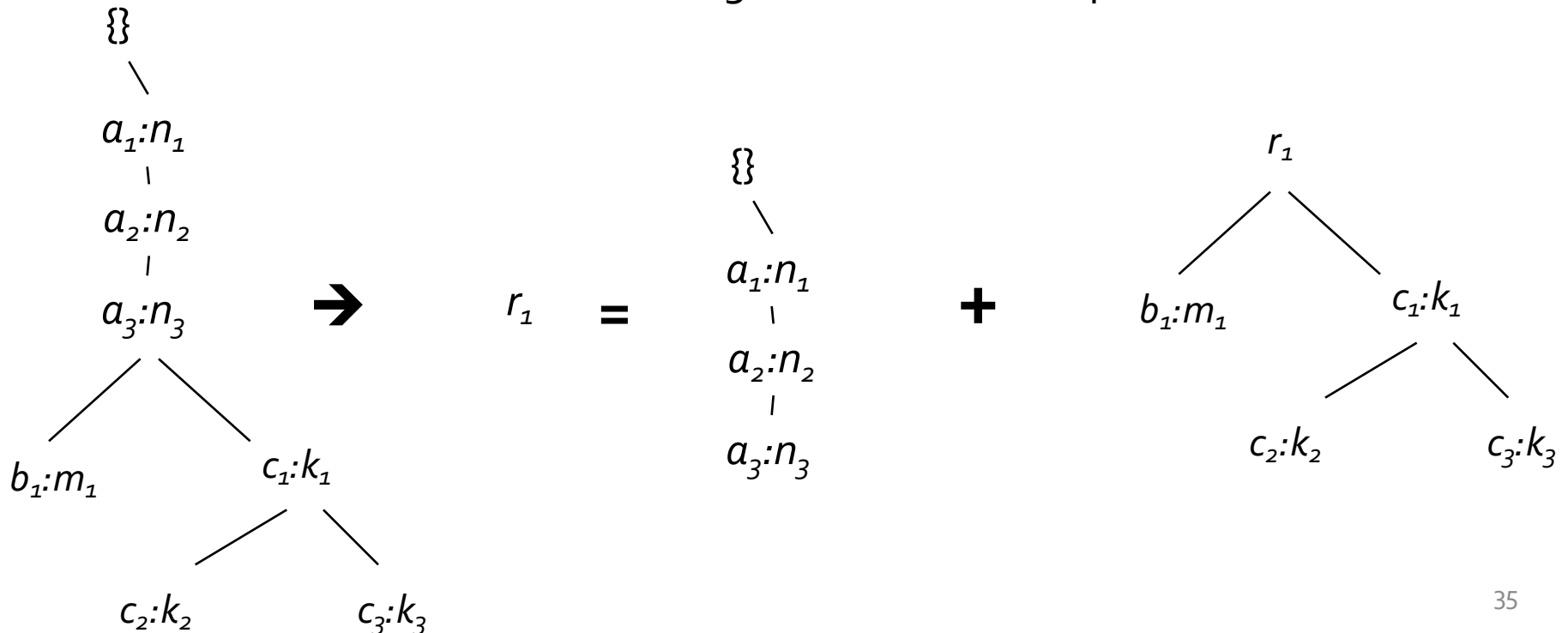
*b*-conditional PB: *fca:1, f:1, c:1* →  $\phi$

*a*-conditional PB: *fc:3* → *fc:3*

*c*-conditional PB: *f:3* → *f:3*

# A Special Case: Single Prefix Path in FP-tree

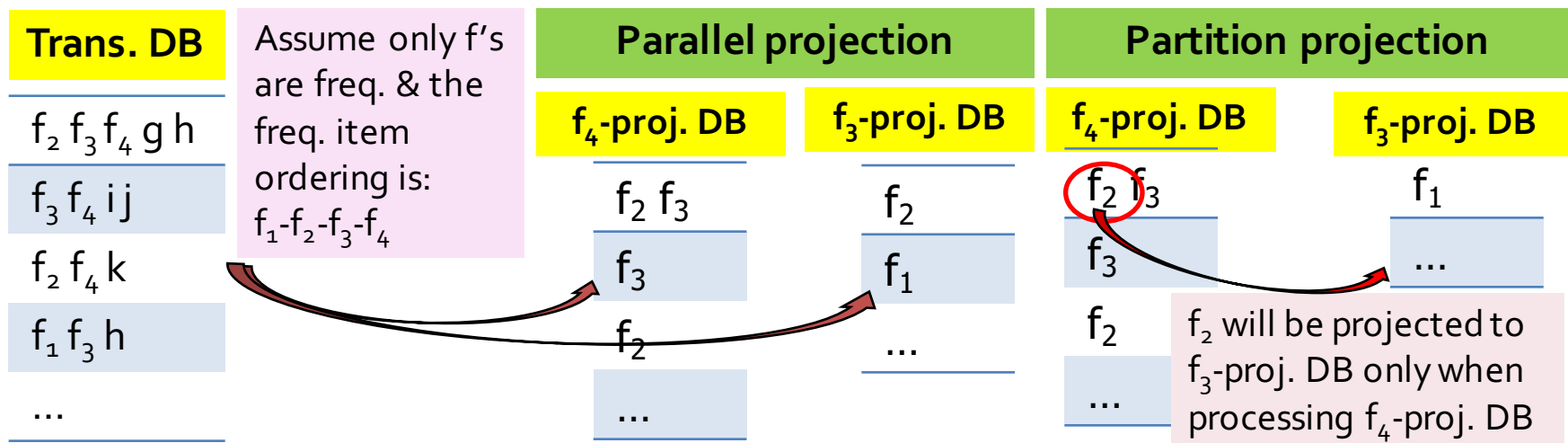
- Suppose a (conditional) FP-tree  $T$  has a shared single prefix-path  $P$
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



# Scaling FP-growth by Database Projection

YouTube: <https://www.youtube.com/watch?v=LXx1xKFgoDg>

- What if FP-tree cannot fit in memory? — DB projection
  - Project the DB based on patterns
  - Construct & mine FP-tree for each projected DB
- **Parallel projection** vs. **partition projection**
  - Parallel projection: Project the DB on each frequent item
    - Space costly, all partitions can be processed in parallel
  - Partition projection: Partition the DB in order
    - Passing the unprocessed parts to subsequent partitions





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

- Basic Concepts
- Frequent Pattern (Itemset) Mining Methods
- **Pattern Evaluation Methods**

# How to Judge if a Rule/Pattern Is Interesting?

- Pattern-mining will generate a large set of patterns/rules
  - Not all the generated patterns/rules are interesting
- Interestingness measures: Objective vs. subjective
  - Objective interestingness measures
    - Support, confidence, correlation, ...
  - Subjective interestingness measures: One man's trash could be another man's treasure
    - Query-based: Relevant to a user's particular request
    - Against one's knowledge-base: unexpected, freshness, timeliness
    - Visualization tools: Multi-dimensional, interactive examination

# Limitation of the Support-Confidence Framework

- Are s and c interesting in association rules: " $A \Rightarrow B$ " [s, c]?
- Example: Suppose one school may have the following statistics on # of students who may play basketball and/or eat cereal:

Be careful!

2-way contingency table

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

- Association rule mining may generate the following:
  - $\text{play-basketball} \Rightarrow \text{eat-cereal}$  [40%, 66.7%] (higher s & c)
- But this strong association rule is misleading: The overall % of students eating cereal is 75% > 66.7%, a more telling rule:
  - $\neg \text{play-basketball} \Rightarrow \text{eat-cereal}$  [35%, 87.5%] (high s & higher c)

# Interestingness Measure: Lift

- Measure of dependent/correlated events: **lift**

$$lift(B, C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

*Lift* is more telling than s & c

- Lift(B, C) may tell how B and C are correlated

- Lift(B, C) = 1: B and C are independent
- > 1: positively correlated
- < 1: negatively correlated

	B	¬B	Σ <sub>row</sub>
C	400	350	750
¬C	200	50	250
Σ <sub>col.</sub>	600	400	1000

- For our example,  $lift(B, C) = \frac{400/1000}{600/1000 \times 750/1000} = 0.89$

$$lift(B, \neg C) = \frac{200/1000}{600/1000 \times 250/1000} = 1.33$$

- Thus, B and C are negatively correlated since  $lift(B, C) < 1$ ;
  - B and ¬C are positively correlated since  $lift(B, \neg C) > 1$

# Interestingness Measure: $\chi^2$

- Another measure to test correlated events:  $\chi^2$

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- General rules

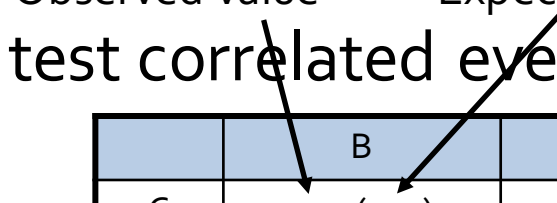
- $\chi^2 = 0$ : independent

- $\chi^2 > 0$ : correlated, either positive or negative, so it needs additional test

- Now,  $\chi^2 = \frac{(400 - 450)^2}{450} + \frac{(350 - 300)^2}{300} + \frac{(200 - 150)^2}{150} + \frac{(50 - 100)^2}{100} = 55.56$

- $\chi^2$  shows B and C are negatively correlated since the expected value is 450 but the observed is only 400
- $\chi^2$  is also more telling than the support-confidence framework

Observed value      Expected value



	B	¬B	$\Sigma_{row}$
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
$\Sigma_{col}$	600	400	1000

# Lift and $\chi^2$ : Are They Always Good Measures?

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!
- But,  $\text{Lift}(B, C) = 8.44 \gg 1$  (Lift shows B and C are strongly positively correlated!)
- $\chi^2 = 670$ : Observed(BC)  $\gg$  expected value (11.85)
- Too many null transactions may “spoil the soup”!

	B	¬B	$\Sigma_{\text{row}}$
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{\text{col.}}$	1100	101000	102100

*null transactions*

**Contingency table with expected values added**

	B	¬B	$\Sigma_{\text{row}}$
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\Sigma_{\text{col.}}$	1100	101000	102100

# Interestingness Measures & Null-Invariance

- *Null invariance*: Value does not change with the # of null-transactions
- A few interestingness measures: Some are null invariant

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
$Lift(A, B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$AllConf(A, B)$	$\frac{s(A \cup B)}{\max\{s(A), s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A, B)$	$\frac{s(A \cup B)}{s(A) + s(B) - s(A \cup B)}$	$[0, 1]$	Yes
$Cosine(A, B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A, B)$	$\frac{1}{2} \left( \frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)} \right)$	$[0, 1]$	Yes
$MaxConf(A, B)$	$\max\left\{ \frac{s(A)}{s(A \cup B)}, \frac{s(B)}{s(A \cup B)} \right\}$	$[0, 1]$	Yes

$\chi^2$  and lift are not null-invariant

Jaccard, cosine, AllConf, MaxConf, and Kulczynski are null-invariant measures

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
$Lift(A, B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$AllConf(A, B)$	$\frac{s(A \cup B)}{\max\{s(A), s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A, B)$	$\frac{s(A \cup B)}{s(A) + s(B) - s(A \cup B)}$	$[0, 1]$	Yes
$Cosine(A, B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A, B)$	$\frac{1}{2} \left( \frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)} \right)$	$[0, 1]$	Yes
$MaxConf(A, B)$	$\max\left\{ \frac{s(A)}{s(A \cup B)}, \frac{s(B)}{s(A \cup B)} \right\}$	$[0, 1]$	Yes

	B	$\neg B$	$\sum_{\text{row}}$
C	100 (11.85)	1000	1100
$\neg C$	1000 (988.15)	100000	101000
$\sum_{\text{col.}}$	1100	101000	102100

Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)

$\chi^2 = 670$ : Observed(BC) >> expected value (11.85)



# Null Invariance: An Important Property

- Why is null invariance crucial for the analysis of massive transaction data?
  - Many transactions may contain neither milk nor coffee!

milk vs. coffee contingency table

	<i>milk</i>	$\neg milk$	$\Sigma_{row}$
<i>coffee</i>	<i>mc</i>	$\neg mc$	<i>c</i>
$\neg coffee$	<i>m</i> $\neg c$	$\neg m$ $\neg c$	$\neg c$
$\Sigma_{col}$	<i>m</i>	$\neg m$	$\Sigma$

- Lift and  $\chi^2$  are not null-invariant: not good to evaluate data that contain too many or too few null transactions!
- Many measures are not null-invariant!

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

Data set	<i>mc</i>	$\neg mc$	<i>m</i> $\neg c$	$\neg m$ $\neg c$	$\chi^2$	<i>Lift</i>
<i>D</i> <sub>1</sub>	10,000	1,000	1,000	100,000	90557	9.26
<i>D</i> <sub>2</sub>	10,000	1,000	1,000	100	0	1
<i>D</i> <sub>3</sub>	100	1,000	1,000	100,000	670	8.44
<i>D</i> <sub>4</sub>	1,000	1,000	1,000	100,000	24740	25.75
<i>D</i> <sub>5</sub>	1,000	100	10,000	100,000	8173	9.18
<i>D</i> <sub>6</sub>	1,000	10	100,000	100,000	965	1.97

# Comparison of Null-Invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?
  - $D_4$ — $D_6$  differentiate the null-invariant measures
  - Kulc (Kulczynski 1927) holds firm and is in balance of both directional implications

2-variable contingency table

	<i>milk</i>	$\neg milk$	$\Sigma_{row}$
<i>coffee</i>	<i>mc</i>	$\neg mc$	<i>c</i>
$\neg coffee$	$m \neg c$	$\neg m \neg c$	$\neg c$
$\Sigma_{col}$	<i>m</i>	$\neg m$	$\Sigma$

All 5 are null-invariant

Data set	<i>mc</i>	$\neg mc$	$m \neg c$	$\neg m \neg c$	<i>AllConf</i>	Jaccard	<i>Cosine</i>	<i>Kulc</i>	<i>MaxConf</i>
$D_1$	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
$D_2$	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
$D_3$	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
$D_4$	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
$D_5$	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
$D_6$	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Subtle: They disagree on those cases

# Analysis of DBLP Coauthor Relationships

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

ID	Author <i>A</i>	Author <i>B</i>	$s(A \cup B)$	$s(A)$	$s(B)$	Jaccard	<i>Cosine</i>	<i>Kulc</i>
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)

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

- Which pairs of authors are strongly related?
  - Use Kulc to find Advisor-advisee, close collaborators

# Imbalance Ratio with Kulczynski Measure

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

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

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

Data set	$mc$	$\neg mc$	$m\neg c$	$\neg m\neg c$	Jaccard	Cosine	Kulc	IR
$D_1$	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
$D_2$	10,000	1,000	1,000	100	0.83	0.91	0.91	0
$D_3$	100	1,000	1,000	100,000	0.05	0.09	0.09	0
$D_4$	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
$D_5$	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
$D_6$	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

# What Measures to Choose for Effective Pattern Evaluation?

- Null value cases are predominant in many large datasets
  - Neither milk nor coffee is in most of the baskets; neither Mike nor Jim is an author in most of the papers; .....
- Null-invariance is an important property
- Lift,  $\chi^2$  and cosine are good measures if null transactions are not predominant
  - Otherwise, Kulczynski + Imbalance Ratio should be used to judge the interestingness of a pattern
- Exercise: 4<sup>th</sup> Credit Project?
  - (“Spam detection”, “SVM”)
  - (“Spam detection”, “Matrix factorization”)
  - (“Link prediction”, “SVM”)
  - (“Link prediction”, “Matrix factorization”)

# Summary

- Basic Concepts:
  - Frequent Patterns, Association Rules, Closed Patterns and Max-Patterns
- Frequent Itemset Mining Methods
  - The Downward Closure Property and The Apriori Algorithm
  - Extensions or Improvements of Apriori
  - Mining Frequent Patterns by Exploring Vertical Data Format
  - FPGrowth: A Frequent Pattern-Growth Approach
  - Mining Closed Patterns
- Which Patterns Are Interesting?—Pattern Evaluation Methods
  - Interestingness Measures: Lift and  $\chi^2$
  - Null-Invariant Measures
  - Comparison of Interestingness Measures

# References

- R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", in Proc. of SIGMOD'93
- R. J. Bayardo, "Efficiently mining long patterns from databases", in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules", in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", VLDB'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
- S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating association rule mining with relational database systems: Alternatives and implications", SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "Parallel algorithm for discovery of association rules", Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", SIGMOD'00

# References (cont.)

- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014
- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- 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
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- 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