Università di Pisa

Data Mining and Machine Learning

**Bioinspired computational methods**

**Biological data mining**

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

***Francesco Marcelloni***

Department of Information Engineering
University of Pisa
ITALY

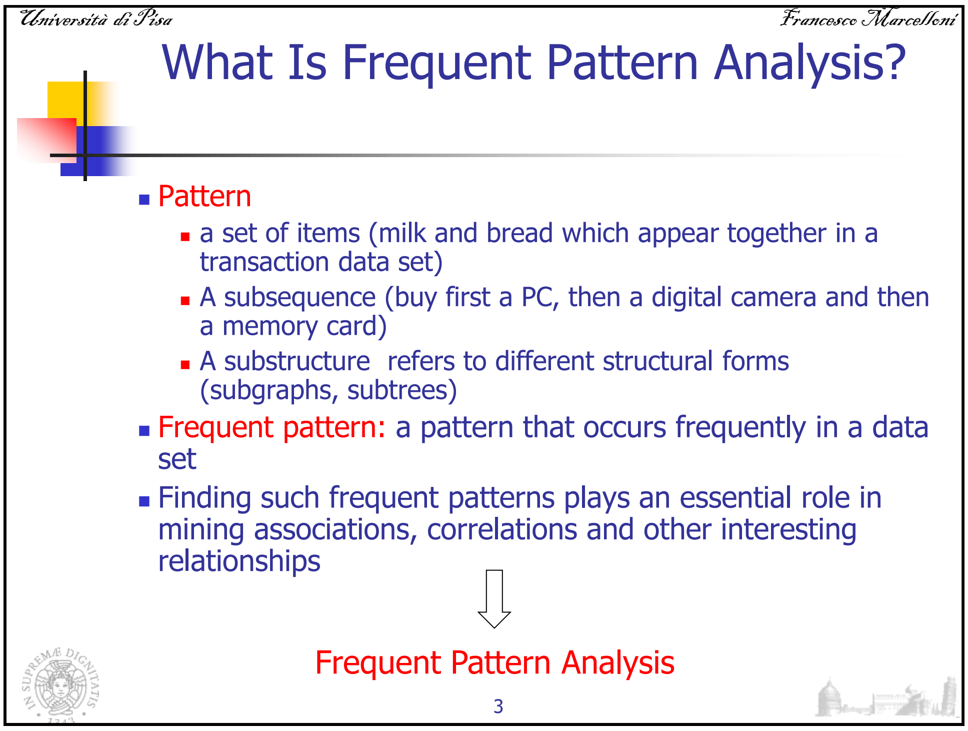
Some slides belong to the collection

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign
Simon Fraser University

Università di Pisa

## Mining Frequent Patterns, Association and Correlations

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

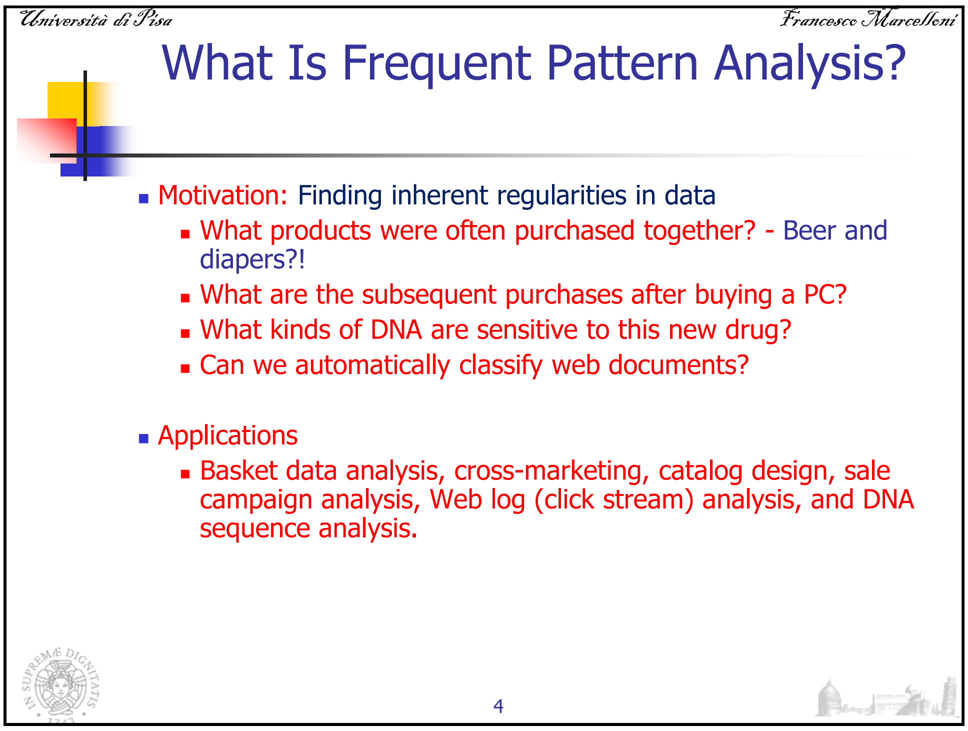


## What Is Frequent Pattern Analysis?

- Pattern
  - a set of items (milk and bread which appear together in a transaction data set)
  - A subsequence (buy first a PC, then a digital camera and then a memory card)
  - A substructure  refers to different structural forms (subgraphs, subtrees)
- Frequent pattern: a pattern that occurs frequently in a data set
- Finding such frequent patterns plays an essential role in mining associations, correlations and other interesting relationships

$\Downarrow$

Frequent Pattern Analysis

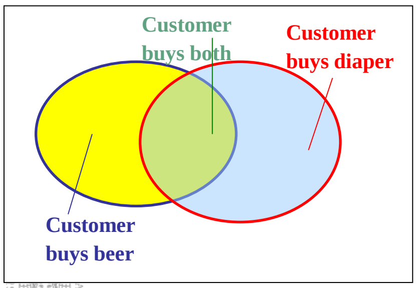


## What Is Frequent Pattern Analysis?

- 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?

| 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 *I*: A set of one or more items $\{I_1, I_2, \ldots, I_m\}$
- k-itemset $X = \{I_1, I_2, \ldots, I_k\}$
- (absolute) support, or, support count of X: Frequency or occurrence of an itemset X
- (relative) support s: 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
  - Broad applications

## Association rule Some definition

- Transaction T: set of items such that $T \subseteq I$
- An association rule is an implication of the form X => Y, where
  - $X \subset I$
  - $Y \subset I$
  - $X \cap Y = \emptyset$
- Support of X => Y in the transaction database D: percentage of transactions in D that contain $X \cup Y$ (i.e., the union of sets X and Y)
- Supp (X => Y) = $P(X \cup Y)$ = probability that a transaction contains $X \cup Y$
- Conf (X => Y) = $P(Y \mid X) = supp(X \cup Y) / supp(X)$ = conditional probability that a transaction having X also contains Y

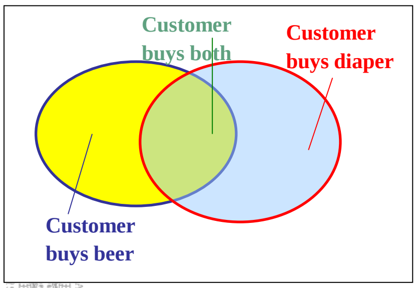
## Association rule
## Some definition

- Each item has a **boolean variable** representing the presence or absence of that item
- Each basket can be represented by a Boolean vector of values assigned to these variables
- Boolean vectors analysed for buying patterns that reflect items that are frequently associated or purchased together.
- Computer =>antivirus_software

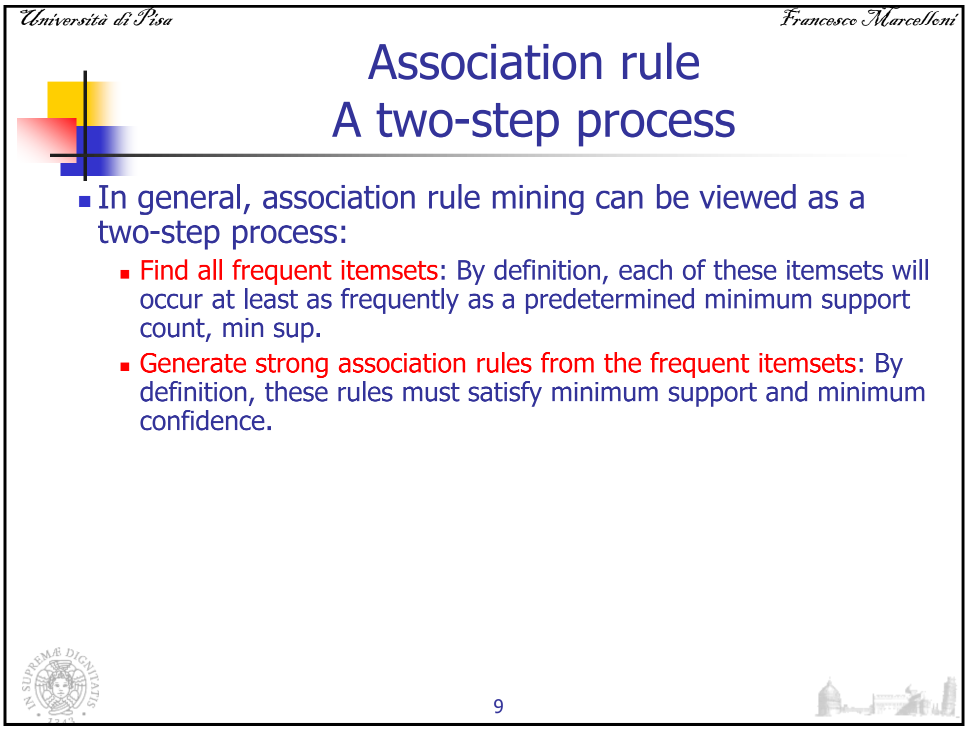
  [support=2%,confidence =60%]
- **Support** -> reflects uselfulness (in the 2% of all transactions, purchased together).
- **Confidence** -> reflects certainty (60% of the customers who purchased a computer also bought the software)

## Why Is Freq. Pattern Mining Important?

| 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 |

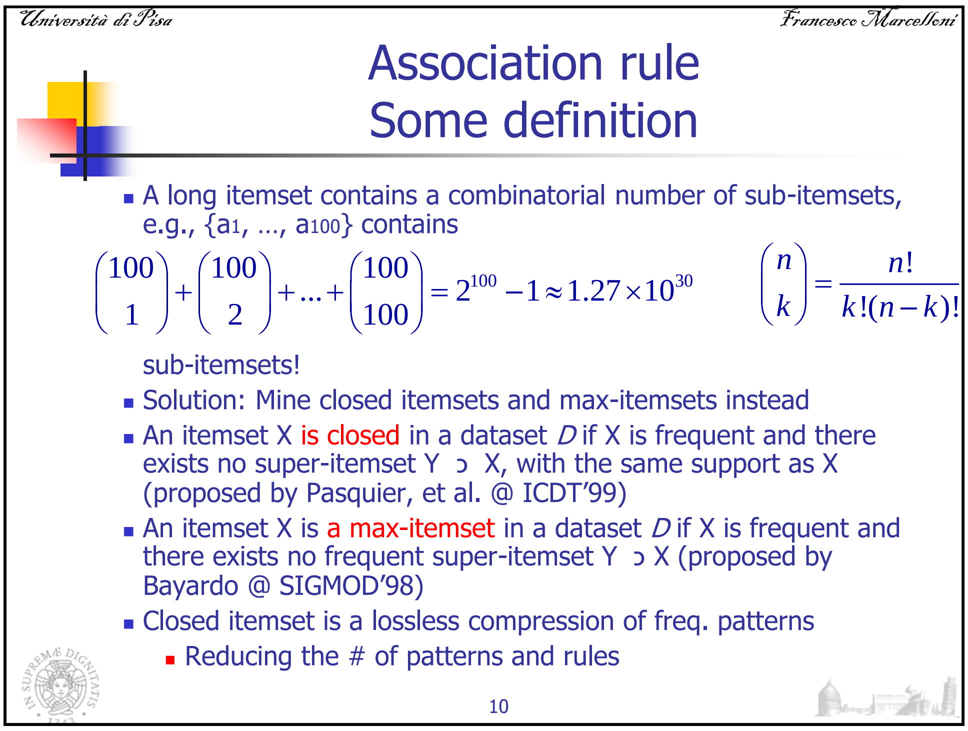


- Find all the rules X => Y with minimum support and confidence
- Let minsup = 50%, minconf = 50%
- Freq. itemsets: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3
- Association rules:
  - Beer => Diaper (60%, 100%)
  - Diaper => Beer (60%, 75%)



## Association rule A two-step process

- In general, association rule mining can be viewed as a two-step process:
  - Find all frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
  - Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.



## Association rule Some definition

- A long itemset contains a combinatorial number of sub-itemsets, e.g., $\{a_1, \ldots, a_{100}\}$ contains

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30} \qquad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

sub-itemsets!

- Solution: Mine closed itemsets and max-itemsets instead
- An itemset X is closed in a dataset *D* if X is frequent and there exists no super-itemset Y ↄ X, with the same support as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-itemset in a dataset *D* if X is frequent and there exists no frequent super-itemset Y ↄ X (proposed by Bayardo @ SIGMOD'98)
- Closed itemset is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

## Association rule
## Some definition

- Let *C* be the set of **closed frequent itemsets** for a data set *D* satisfying a minimum support threshold, *min_sup*.
- Let *M* be the set of **maximal frequent itemsets** for *D* satisfying min_sup.

- Suppose that we have the support count of each itemset in *C* and *M*.
- **Notice that *C* and its count information can be used to derive the whole set of frequent itemsets**.
- Thus, we say that *C* contains complete information regarding its corresponding frequent itemsets.

## Association rule
## Some definition

Exercise.

DB = {<a1, …, a100>, < a1, …, a50>}

Min_sup = 1.

- What is the set of closed itemset?
  - <a1, …, a100>: 1
  - < a1, …, a50>: 2
- What is the set of max-itemset?
  - <a1, …, a100>: 1
- The set of closed frequent itemsets contains complete information regarding the frequent itemsets

  For instance, from *C*, we can derive {$a_2$, $a_{45}$ : 2};

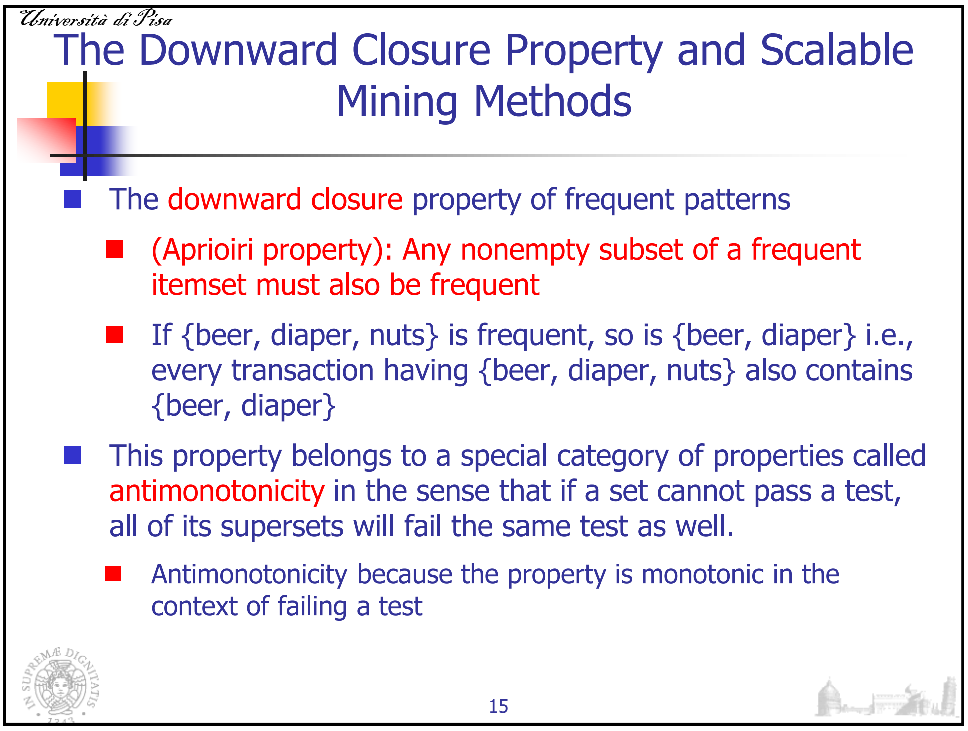
  From D, we can only infer that {$a_2$, $a_{45}$ : 1}

## Computational Complexity of Frequent Itemset Mining

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

## 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

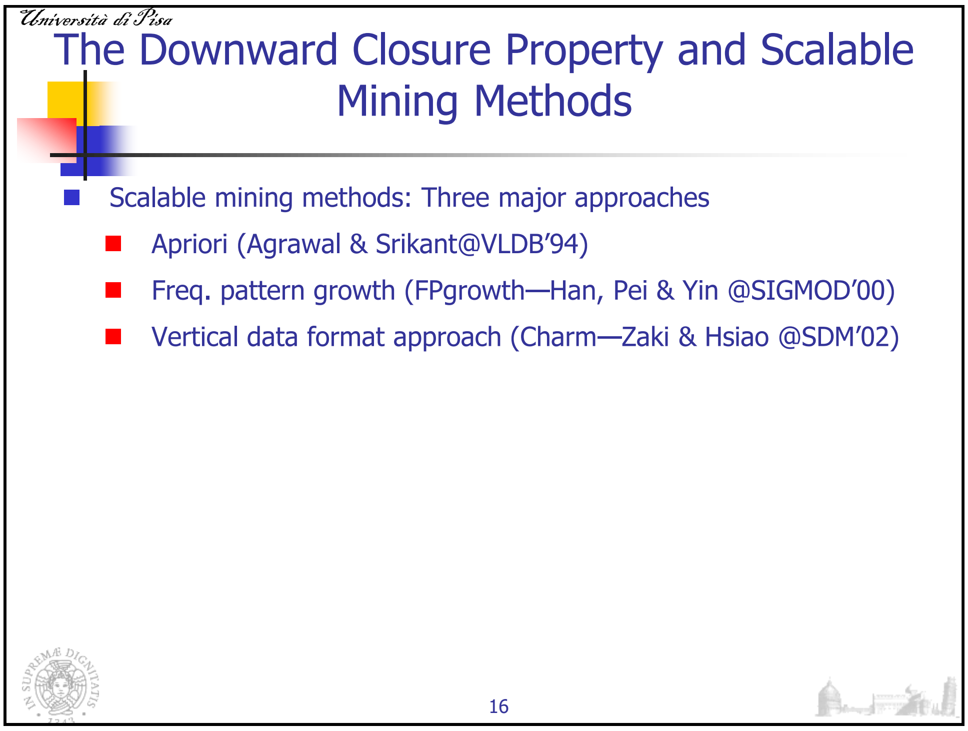

Università di Pisa

## The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - (Aprioiri property): Any nonempty subset of a frequent itemset must also be frequent
  - If {beer, diaper, nuts} is frequent, so is {beer, diaper} i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- This property belongs to a special category of properties called antimonotonicity in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well.
  - Antimonotonicity because the property is monotonic in the context of failing a test

15

15

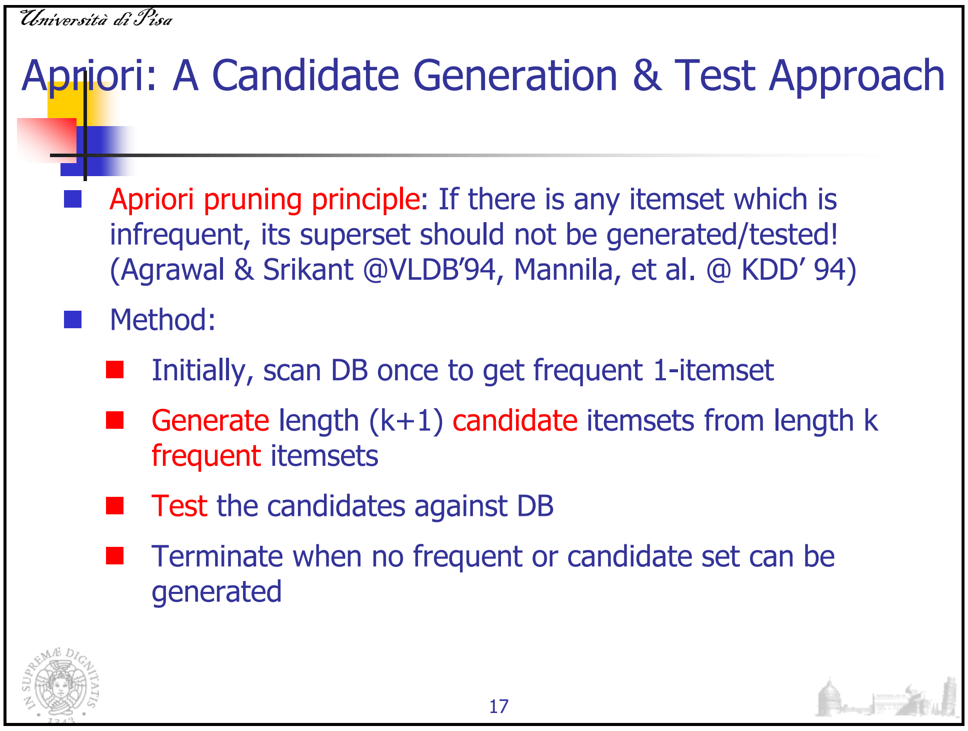

Università di Pisa

## The Downward Closure Property and Scalable Mining Methods

- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

16

16



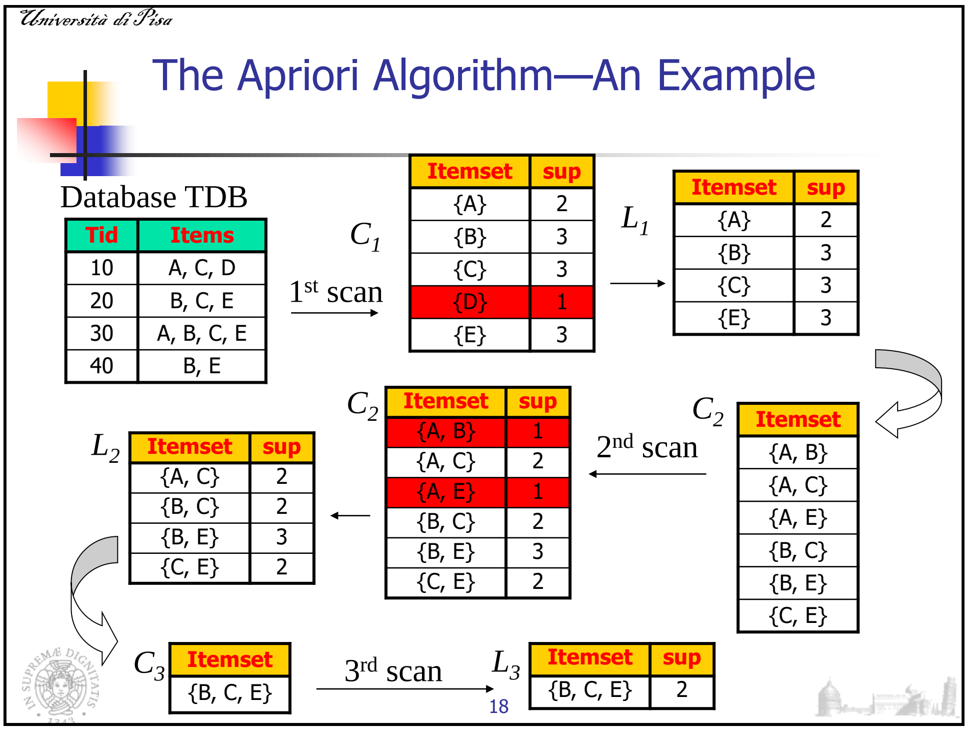
*Università di Pisa*

## 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

17

17



*Università di Pisa*

## 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 |

1st scan $\rightarrow$ $C_1$

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

$\rightarrow$ $L_1$

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

$C_2$

| Itemset |
|---|
| {A, B} |
| {A, C} |
| {A, E} |
| {B, C} |
| {B, E} |
| {C, E} |

2nd scan $\leftarrow$ $C_2$

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

$\leftarrow$ $L_2$

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

$C_3$

| Itemset |
|---|
| {B, C, E} |

3rd scan $\rightarrow$ $L_3$

| Itemset | sup |
|---|---|
| {B, C, E} | 2 |

18

18

## The Apriori Algorithm (Pseudo-Code)

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

```
$L_1$ = {frequent items};
for ($k$ = 1; $L_k$ !=$\emptyset$; $k$++) 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 $\cup_k L_k$;
```

## The Apriori Algorithm (Observations)

"How is the Apriori property used in the algorithm?"

Let us look at how $L_{k-1}$ is used to find $L_k$ for k>=2.

A two-step process, consisting of join and prune actions.

- The join step: in order to find $L_k$, a set of candidate k-itemsets, donoted $C_k$, is generated by joining $L_{k-1}$ with itself.
  - The notation $l_i[j]$ refers to the jth item in li (e.g., $l_1[k-2]$ refers to the second to the last item in l ).
  - For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.
  - For the (k-1)-itemset, $l_i$ , this means that the items are sorted such that $l_i[1] < l_i[2] < \ldots < l_i[k-1]$.

Università di Pisa

## The Apriori Algorithm (Observations)

- The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of $L_{k-1}$ are joinable if their first (k-2) items are in common.

$$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$$

- The resulting itemset formed by the joining is

$$\{l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]\}$$

- The prune step: any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of a candidate k-itemset is not in $L_{k-1}$, then the candidate cannot be frequent either and so can be removed from $C_k$.

21

21

Università di Pisa

## The Apriori Algorithm (An example)

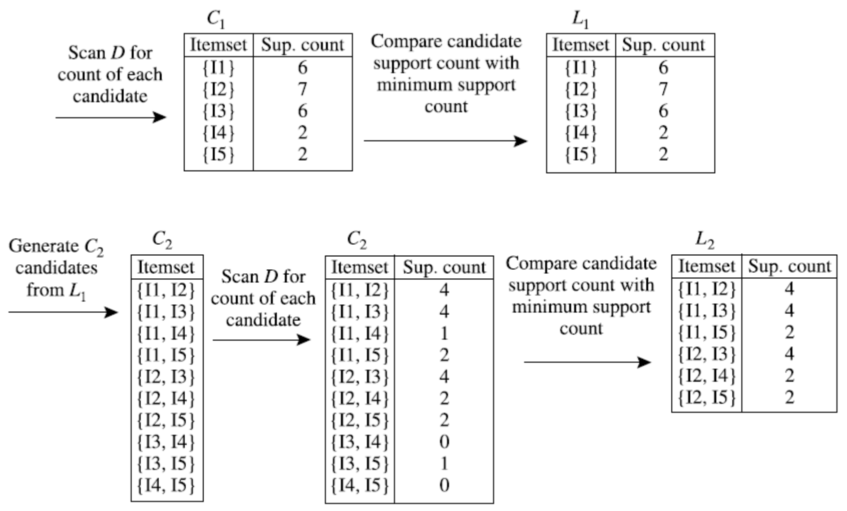
| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Let us suppose that the minimum support is 2

22

22

## The Apriori Algorithm (An example)


Scan $D$ for count of each candidate $\rightarrow$

$C_1$

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

Compare candidate support count with minimum support count $\rightarrow$

$L_1$

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

Generate $C_2$ candidates from $L_1$ $\rightarrow$

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate $\rightarrow$

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count $\rightarrow$

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

23

23

## The Apriori Algorithm (An example)

**(a)** Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

**(b)** Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of $L_2$. Therefore, keep {I1, I2, I3} in $C_3$.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of $L_2$. Therefore, keep {I1, I2, I5} in $C_3$.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I1, I3, I5} from $C_3$.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I3, I4} from $C_3$.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I3, I5} from $C_3$.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I4, I5} from $C_3$.

**(c)** Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

24

## The Apriori Algorithm (An example)

The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, $C_4$. Although the join results in {{I1, I2, I3, I5}}, itemset {I1, I2, I3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets. ■

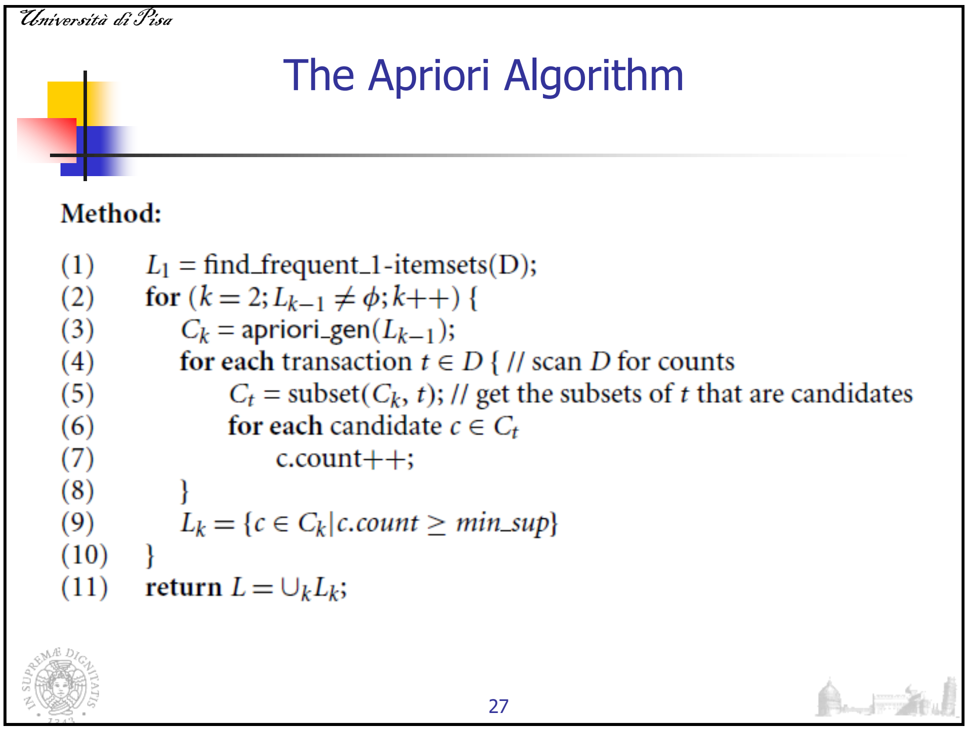
## The Apriori Algorithm

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- *D*, a database of transactions;
- *min_sup*, the minimum support count threshold.

**Output:** *L*, frequent itemsets in *D*.


Università di Pisa

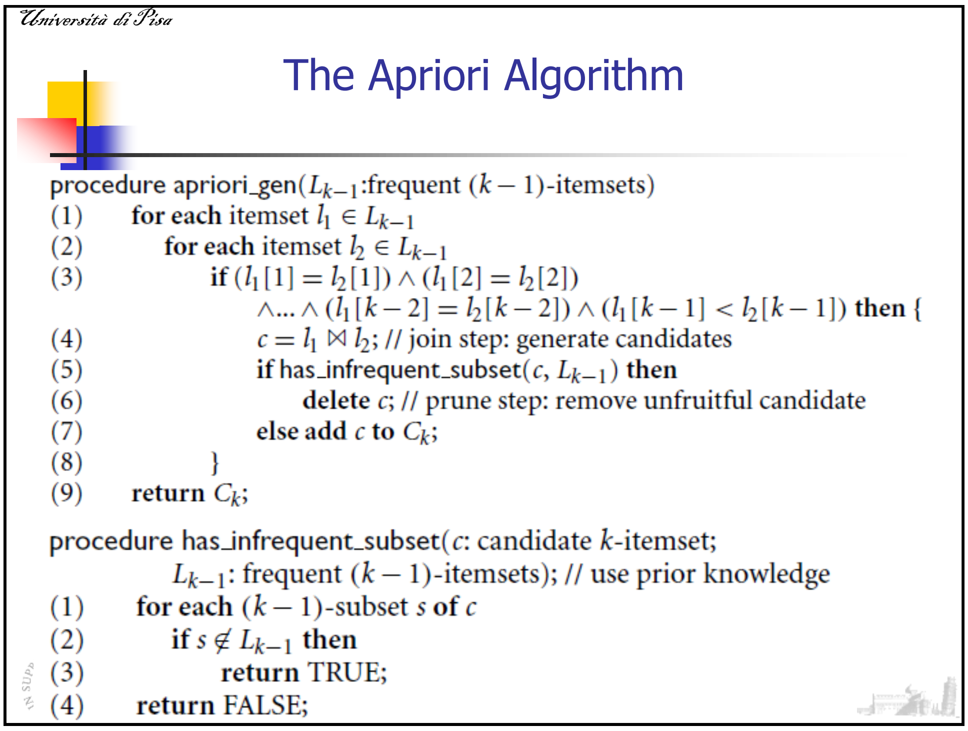
## The Apriori Algorithm

**Method:**

```
(1)   $L_1$ = find_frequent_1-itemsets(D);
(2)   for ($k = 2; L_{k-1} \neq \phi; k{+}{+}$) {
(3)       $C_k$ = apriori_gen($L_{k-1}$);
(4)       for each transaction $t \in D$ { // scan $D$ for counts
(5)           $C_t$ = subset($C_k$, $t$); // get the subsets of $t$ that are candidates
(6)           for each candidate $c \in C_t$
(7)               c.count++;
(8)       }
(9)       $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10)  }
(11)  return $L = \cup_k L_k$;
```

27

27

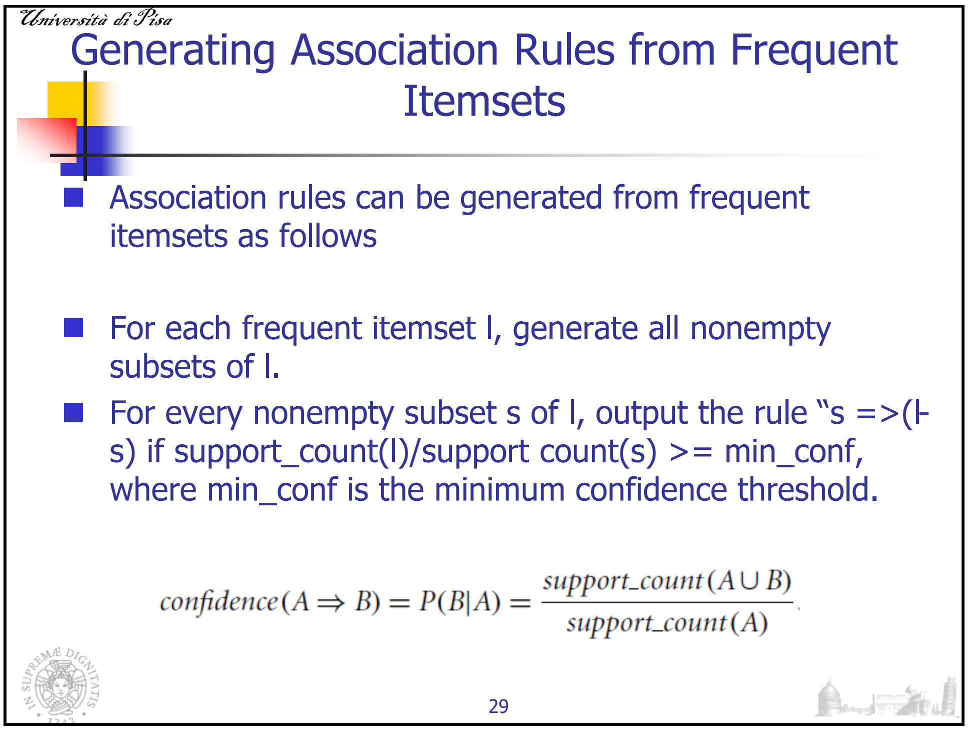

Università di Pisa

## The Apriori Algorithm

```
procedure apriori_gen($L_{k-1}$:frequent $(k-1)$-itemsets)
(1)   for each itemset $l_1 \in L_{k-1}$
(2)       for each itemset $l_2 \in L_{k-1}$
(3)           if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$
                  $\wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
(4)               $c = l_1 \bowtie l_2$; // join step: generate candidates
(5)               if has_infrequent_subset($c$, $L_{k-1}$) then
(6)                   delete $c$; // prune step: remove unfruitful candidate
(7)               else add $c$ to $C_k$;
(8)           }
(9)   return $C_k$;

procedure has_infrequent_subset($c$: candidate $k$-itemset;
              $L_{k-1}$: frequent $(k-1)$-itemsets); // use prior knowledge
(1)   for each $(k-1)$-subset $s$ of $c$
(2)       if $s \notin L_{k-1}$ then
(3)           return TRUE;
(4)   return FALSE;
```

28


Università di Pisa

## Generating Association Rules from Frequent Itemsets

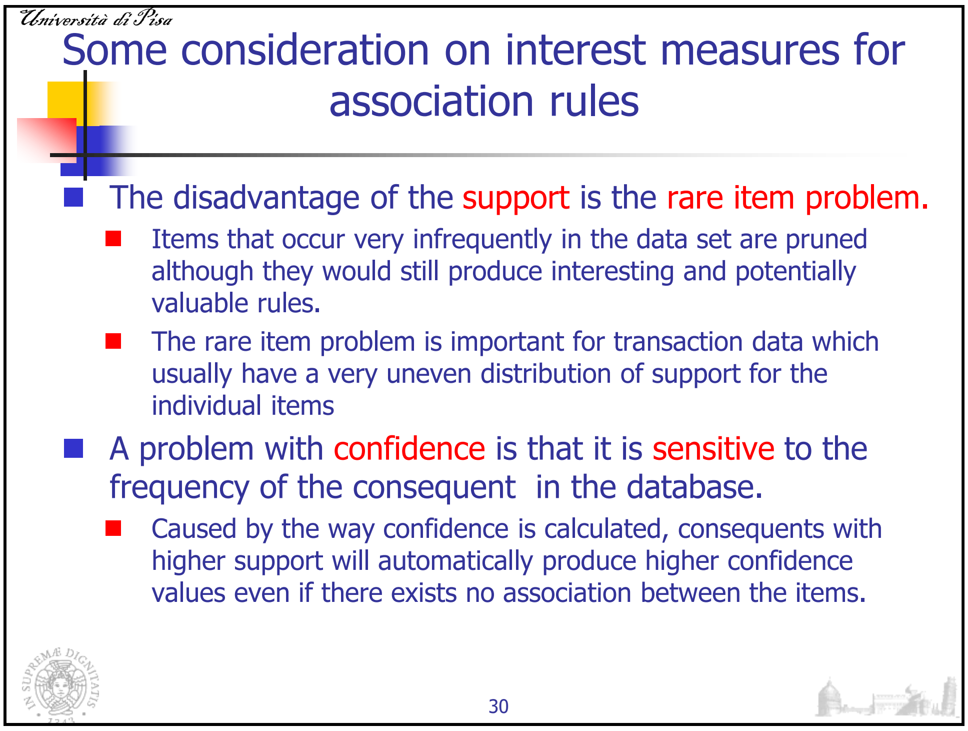
- Association rules can be generated from frequent itemsets as follows

- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "s =>(l-s) if support_count(l)/support count(s) >= min_conf, where min_conf is the minimum confidence threshold.

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

29

29


Università di Pisa

## Some consideration on interest measures for association rules

- The disadvantage of the support is the rare item problem.
  - Items that occur very infrequently in the data set are pruned although they would still produce interesting and potentially valuable rules.
  - The rare item problem is important for transaction data which usually have a very uneven distribution of support for the individual items
- A problem with confidence is that it is sensitive to the frequency of the consequent in the database.
  - Caused by the way confidence is calculated, consequents with higher support will automatically produce higher confidence values even if there exists no association between the items.

30

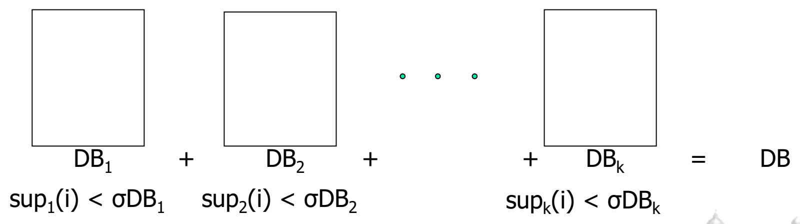
30

## Scalable Frequent Itemset Mining Methods

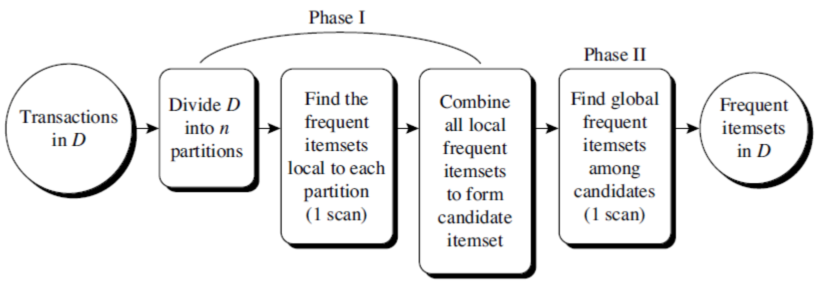
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth:  A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

## Further Improvement of the Apriori Method

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

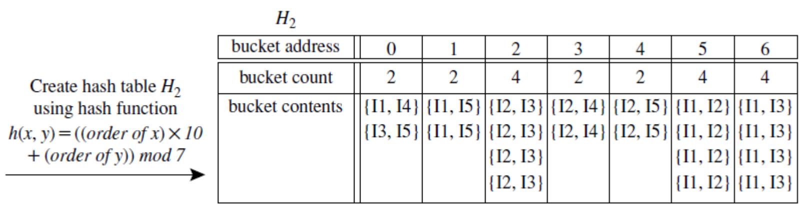


## Further Improvement of the Apriori Method


Phase I
Phase II
Transactions in D
Divide D into n partitions
Find the frequent itemsets local to each partition (1 scan)
Combine all local frequent itemsets to form candidate itemset
Find global frequent itemsets among candidates (1 scan)
Frequent itemsets in D

## DHP: Reduce the Number of Candidates

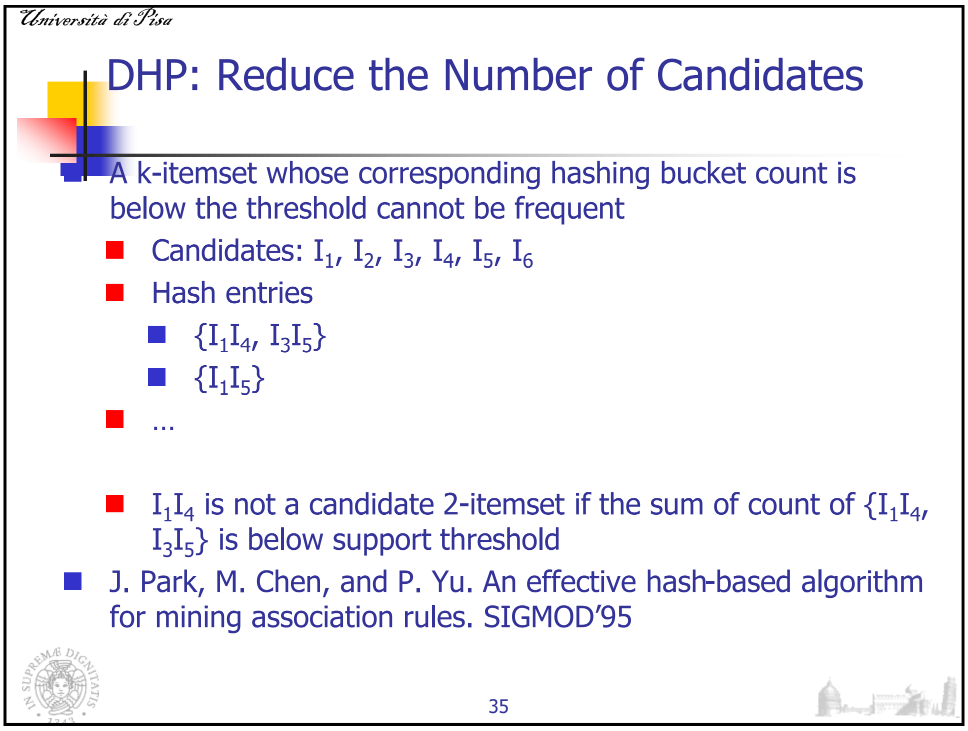
- When scanning each transaction in the database to generate the frequent 1-itemsets, $L_1$, we can generate all the 2-itemsets $C_k$, for each transaction, hash (i.e., map) them into the different buckets of a hash table structure, and increase the corresponding bucket counts.



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

$H_2$

| 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, I3} | {I2, I4} {I2, I4} | {I2, I5} {I2, I5} | {I1, I2} {I1, I2} {I1, I2} {I1, I2} | {I1, I3} {I1, I3} {I1, I3} {I1, I3} |

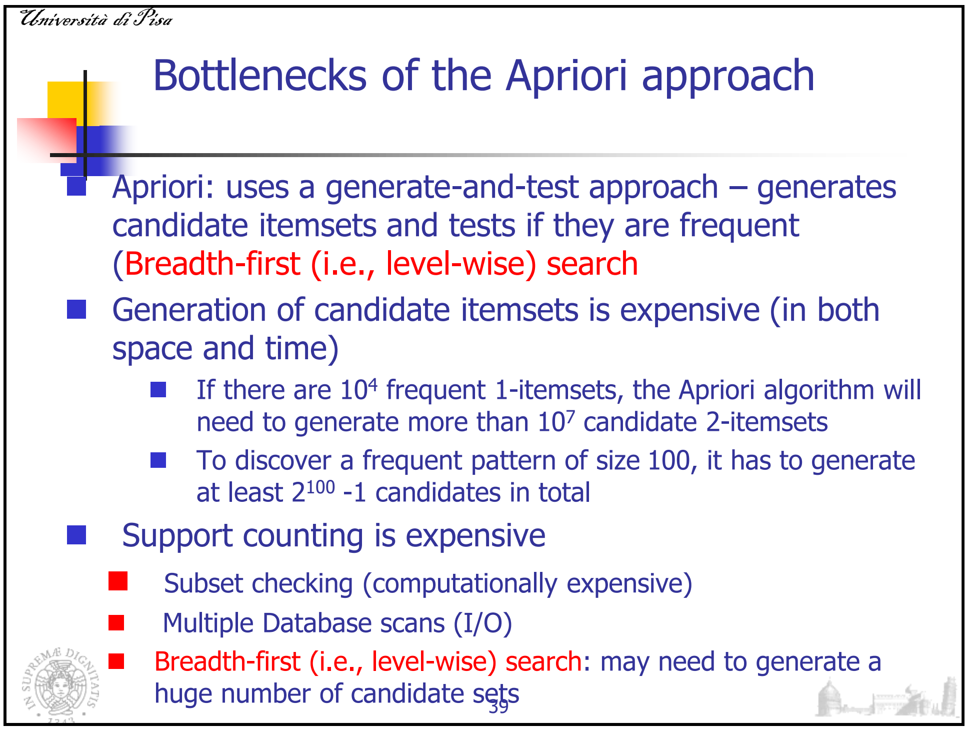

Università di Pisa

## DHP: Reduce the Number of Candidates

- A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: $I_1$, $I_2$, $I_3$, $I_4$, $I_5$, $I_6$
  - Hash entries
    - $\{I_1I_4, I_3I_5\}$
    - $\{I_1I_5\}$
  - …
  - $I_1I_4$ is not a candidate 2-itemset if the sum of count of $\{I_1I_4, I_3I_5\}$ is below support threshold
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95

35

35

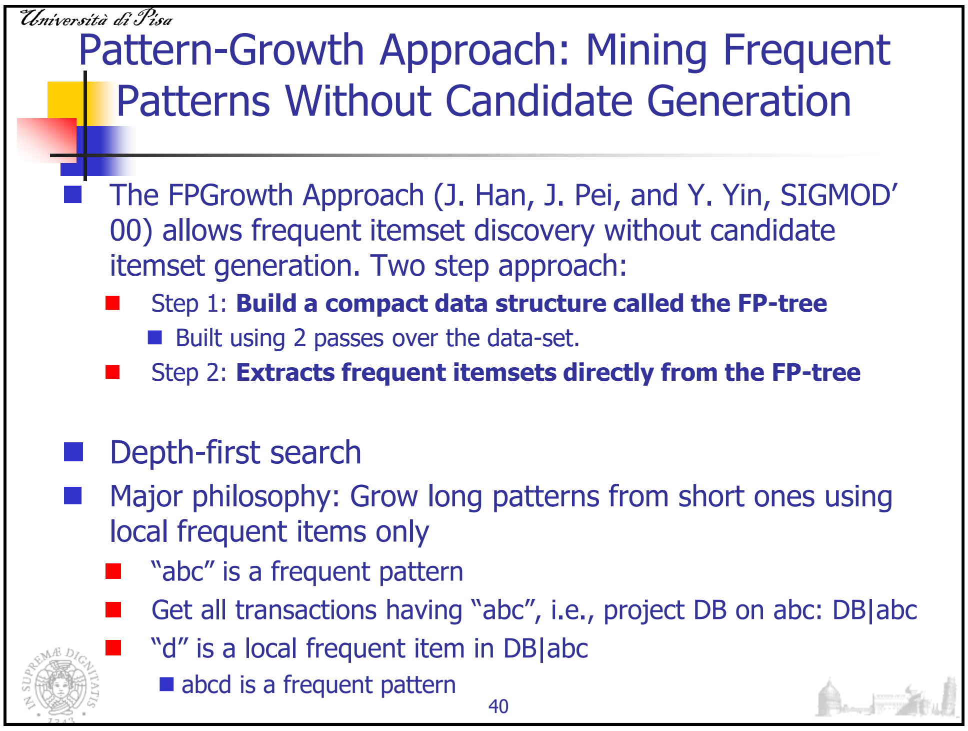

Università di Pisa

## Bottlenecks of the Apriori approach

- Apriori: uses a generate-and-test approach – generates candidate itemsets and tests if they are frequent (Breadth-first (i.e., level-wise) search
- Generation of candidate itemsets is expensive (in both space and time)
  - If there are $10^4$ frequent 1-itemsets, the Apriori algorithm will need to generate more than $10^7$ candidate 2-itemsets
  - To discover a frequent pattern of size 100, it has to generate at least $2^{100}$ -1 candidates in total
- Support counting is expensive
  - Subset checking (computationally expensive)
  - Multiple Database scans (I/O)
  - Breadth-first (i.e., level-wise) search: may need to generate a huge number of candidate sets

39

39

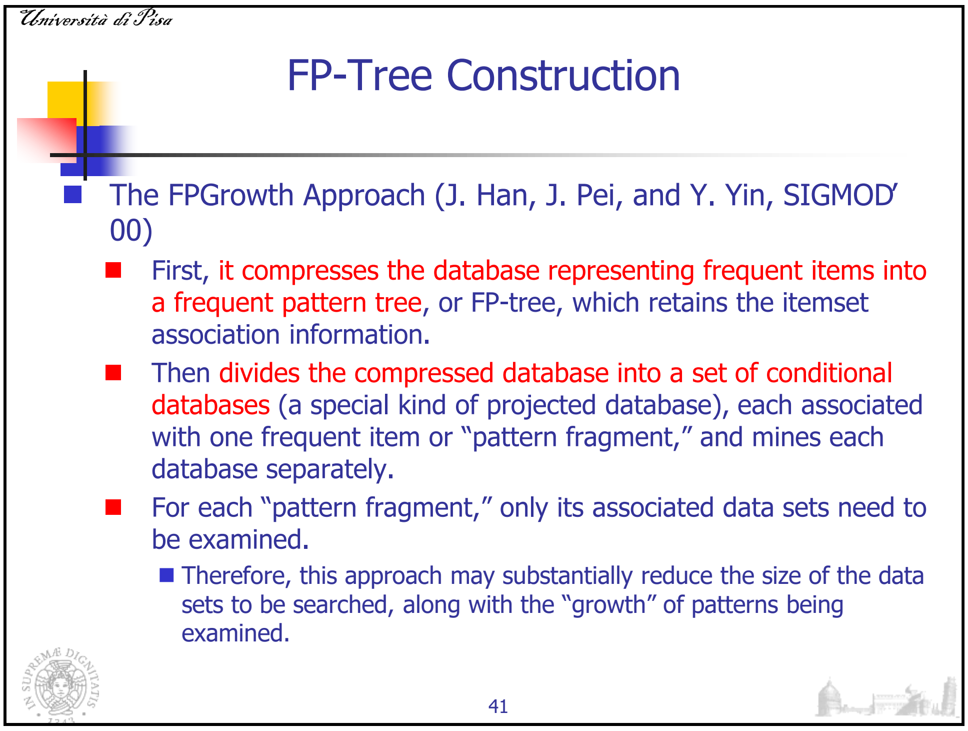

Università di Pisa

## Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00) allows frequent itemset discovery without candidate itemset generation. Two step approach:
  - Step 1: **Build a compact data structure called the FP-tree**
    - Built using 2 passes over the data-set.
  - Step 2: **Extracts frequent itemsets directly from the FP-tree**
- Depth-first search
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc
    - abcd is a frequent pattern

40

40


Università di Pisa

## FP-Tree Construction

- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - First, it compresses the database representing frequent items into a frequent pattern tree, or FP-tree, which retains the itemset association information.
  - Then divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each database separately.
  - For each "pattern fragment," only its associated data sets need to be examined.
    - Therefore, this approach may substantially reduce the size of the data sets to be searched, along with the "growth" of patterns being examined.

41

41

## Step 1: FP-Tree Construction

- FP-Tree is constructed using 2 passes over the data-set:

Pass 1:

- Scan data and find support for each item.
- Discard infrequent items.
- Sort frequent items in decreasing order based on their support.

- Use this order when building the FP-Tree, so common prefixes can be shared.

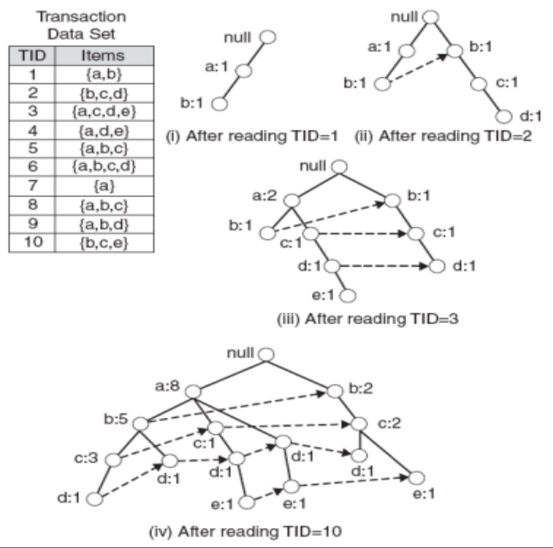
## Step 1: FP-Tree Construction

Pass 2:

- Nodes correspond to items and have a counter
  1. FP-Growth reads 1 transaction at a time and maps it to a path
  2. Fixed order is used, so paths can overlap when transactions share items (when they have the same prefix ).
     - In this case, counters are incremented
  3. Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines)
     - The more paths that overlap, the higher the compression. FP-tree may fit in memory.
  4. Frequent itemsets extracted from the FP-Tree.

Università di Pisa

## Step 1: FP-Tree Construction


Transaction Data Set

| TID | Items |
|---|---|
| 1 | {a,b} |
| 2 | {b,c,d} |
| 3 | {a,c,d,e} |
| 4 | {a,d,e} |
| 5 | {a,b,c} |
| 6 | {a,b,c,d} |
| 7 | {a} |
| 8 | {a,b,c} |
| 9 | {a,b,d} |
| 10 | {b,c,e} |

Università di Pisa

## Step 1: FP-Tree Size

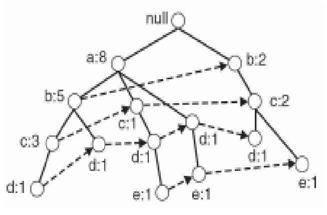
- The FP-Tree usually has a smaller size than the uncompressed data - typically many transactions share items (and hence prefixes).
  - Best case scenario: all transactions contain the same set of items.
    - 1 path in the FP-tree
  - Worst case scenario: every transaction has a unique set of items (no items in common)
    - Size of the FP-tree is at least as large as the original data.
    - Storage requirements for the FP-tree are higher - need to store the pointers between the nodes and the counters.

- The size of the FP-tree depends on how the items are ordered
- Ordering by decreasing support is typically used but it does not always lead to the smallest tree (it's a heuristic).

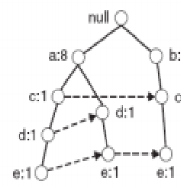
## Step 2: Frequent Itemset Generation

- FP-Growth extracts frequent itemsets from the FP-tree.
- Bottom-up algorithm - from the leaves towards the root
- Divide and conquer: first look for frequent itemsets ending in e, then de, etc. . . then d, then cd, etc. . .
- First, **extract prefix path sub-trees** ending in an item(set). (hint: use the linked lists)

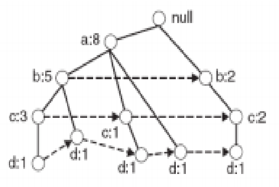
## Prefix path sub-trees (Example)



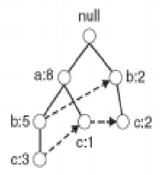
↑ Complete FP-tree



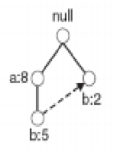
(a) Paths containing node e



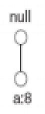
(b) Paths containing node d



(c) Paths containing node c



(d) Paths containing node b

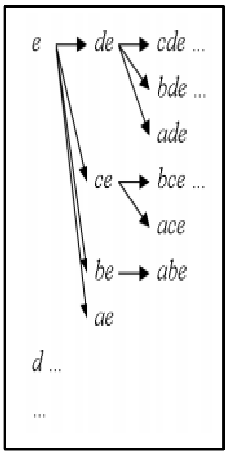


(e) Paths containing node a

Università di Pisa

## Step 2: Frequent Itemset Generation

- Each prefix path sub-tree is processed recursively to extract the frequent itemsets. Solutions are then merged.
  - E.g. the prefix path sub-tree for e will be used to extract frequent itemsets ending in e, then in de, ce, be and ae, then in cde, bde, cde, etc.
  - Divide and conquer approach

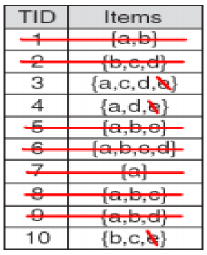


48

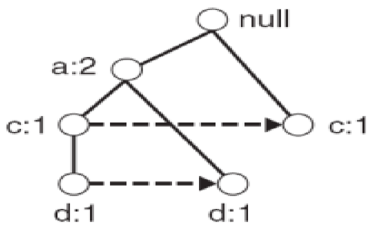
Università di Pisa

## Conditional FP-Tree

- Conditional FP-Tree: FP-Tree that would be built if we only consider transactions containing a particular itemset (and then removing that itemset from all transactions).
- Example: FP-Tree conditional on e.



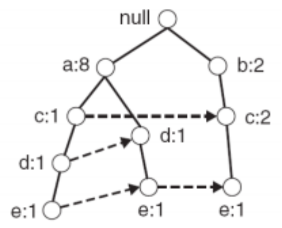
| TID | Items |
|---|---|
| ~~1~~ | ~~{a,b}~~ |
| ~~2~~ | ~~{b,c,d}~~ |
| 3 | {a,c,d,~~e~~} |
| 4 | {a,d,~~e~~} |
| ~~5~~ | ~~{a,b,c}~~ |
| ~~6~~ | ~~{a,b,c,d}~~ |
| ~~7~~ | ~~{a}~~ |
| ~~8~~ | ~~{a,b,c}~~ |
| ~~9~~ | ~~{a,b,d}~~ |
| 10 | {b,c,~~e~~} |



49

## Example

- Let minSup = 2 and extract all frequent itemsets containing e.
  1. Obtain the prefix path sub-tree for e:



## Example

2. Check if e is a frequent item by adding the counts along the linked list (dotted line). If so, extract it.

   Yes, count =3 so {e} is extracted as a frequent itemset.

3. As e is frequent, find frequent itemsets ending in e. i.e. de, ce, be and ae.

## Example

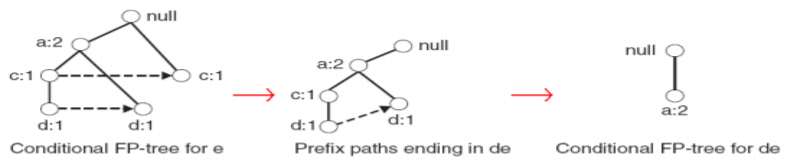
3. Use the conditional FP-tree for e to find frequent itemsets ending in de, ce and ae

Note that be is not considered as b is not in the conditional FP-tree for e.

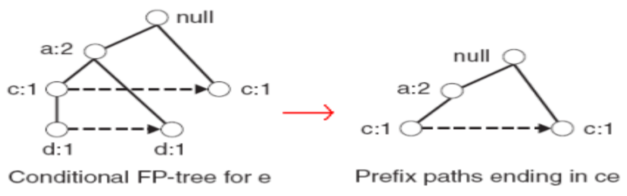
For each of them (e.g. de), find the prefix paths from the conditional tree for e, extract frequent itemsets, generate conditional FP-tree, etc... (recursive)

## Example

Example: e -> de -> ade ({d,e}, {a,d,e} are found to be frequent)

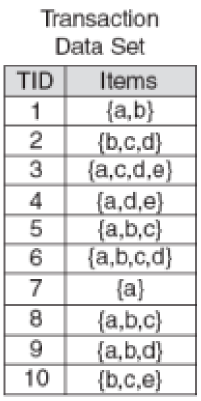

null
a:2
c:1
c:1
d:1
d:1
Conditional FP-tree for e
null
a:2
c:1
d:1
d:1
Prefix paths ending in de
null
a:2
Conditional FP-tree for de

Example: e -> ce ({c,e} is found to be frequent)



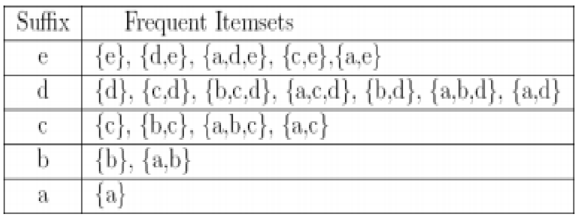
## Result

Frequent itemsets found (ordered by suffix and order in which they are found):



Transaction Data Set

| TID | Items |
|---|---|
| 1 | {a,b} |
| 2 | {b,c,d} |
| 3 | {a,c,d,e} |
| 4 | {a,d,e} |
| 5 | {a,b,c} |
| 6 | {a,b,c,d} |
| 7 | {a} |
| 8 | {a,b,c} |
| 9 | {a,b,d} |
| 10 | {b,c,e} |



| Suffix | Frequent Itemsets |
|---|---|
| e | {e}, {d,e}, {a,d,e}, {c,e},{a,e} |
| d | {d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d} |
| c | {c}, {b,c}, {a,b,c}, {a,c} |
| b | {b}, {a,b} |
| a | {a} |

54

## Discussion

- Advantages of FP-Growth
  - only 2 passes over data-set
  - “compresses” data-set
  - no candidate generation
  - much faster than Apriori
- Disadvantages of FP-Growth

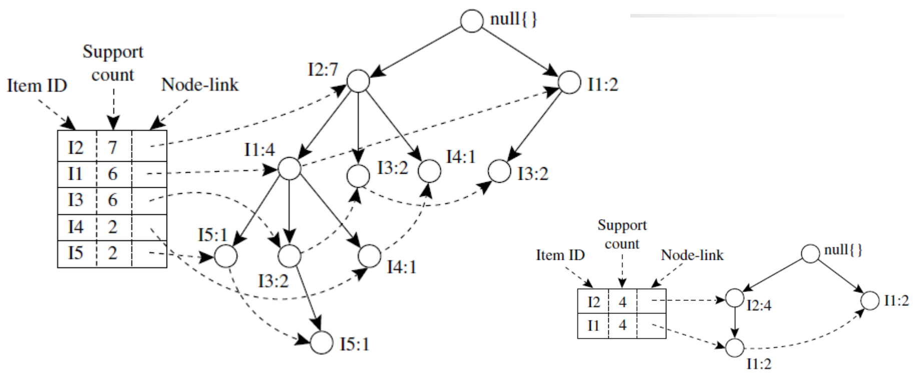
  FP-Tree may not fit in memory!!

  FP-Tree is expensive to build

55

Università di Pisa

## FP-tree: Another Example



| *Item* | *Conditional Pattern Base* | *Conditional FP-tree* | *Frequent Patterns Generated* |
|---|---|---|---|
| I5 | {{I2, I1: 1}, {I2, I1, I3: 1}} | ⟨I2: 2, I1: 2⟩ | {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2} |
| I4 | {{I2, I1: 1}, {I2: 1}} | ⟨I2: 2⟩ | {I2, I4: 2} |
| I3 | {{I2, I1: 2}, {I2: 2}, {I1: 2}} | ⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩ | {I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2} |
| I1 | {{I2: 4}} | ⟨I2: 4⟩ | {I2, I1: 4} |

56

Università di Pisa

## FP-tree: The Algorithm

**Algorithm: FP_growth.** Mine frequent itemsets using an FP-tree by pattern fragment growth.

**Input:**

- *D*, a transaction database;
- *min_sup*, the minimum support count threshold.

**Output:** The complete set of frequent patterns.

**Method:**

**1.** The FP-tree is constructed in the following steps:

(a) Scan the transaction database *D* once. Collect *F*, the set of frequent items, and their support counts. Sort *F* in support count descending order as *L*, the *list* of frequent items.

57

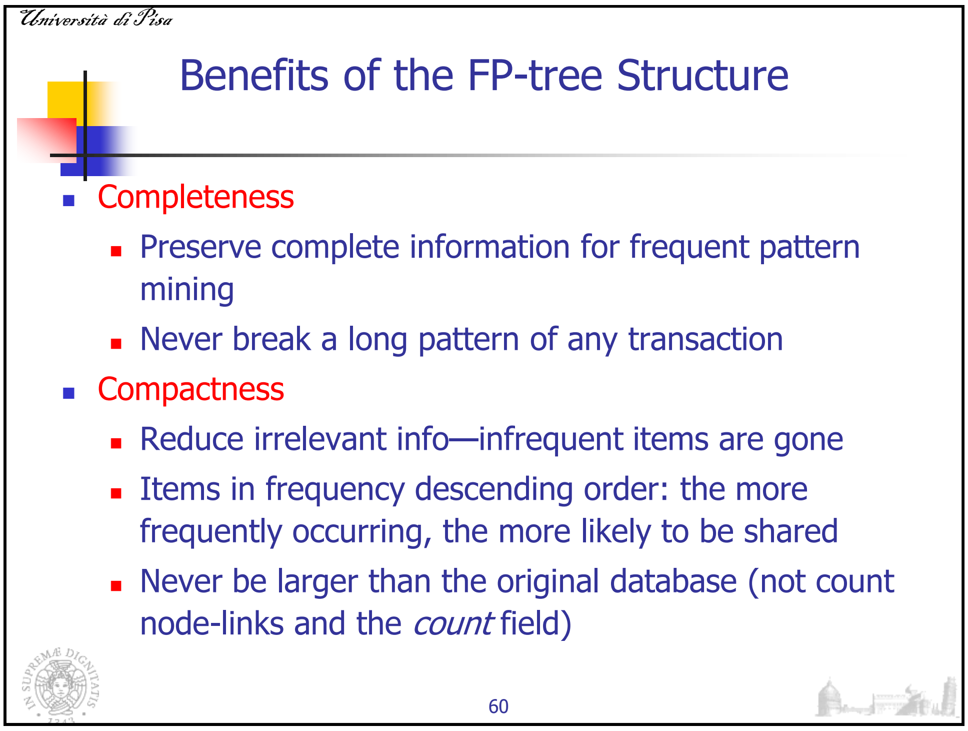
## FP-tree: Another Example

**(b)** Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in *D* do the following.
Select and sort the frequent items in *Trans* according to the order of *L*. Let the sorted frequent item list in *Trans* be $[p|P]$, where $p$ is the first element and $P$ is the remaining list. Call insert_tree($[p|P]$, $T$), which is performed as follows. If $T$ has a child $N$ such that *N.item-name* = *p.item-name*, then increment $N$'s count by 1; else create a new node $N$, and let its count be 1, its parent link be linked to $T$, and its node-link to the nodes with the same *item-name* via the node-link structure. If $P$ is nonempty, call insert_tree($P$, $N$) recursively.

## FP-tree: Another Example

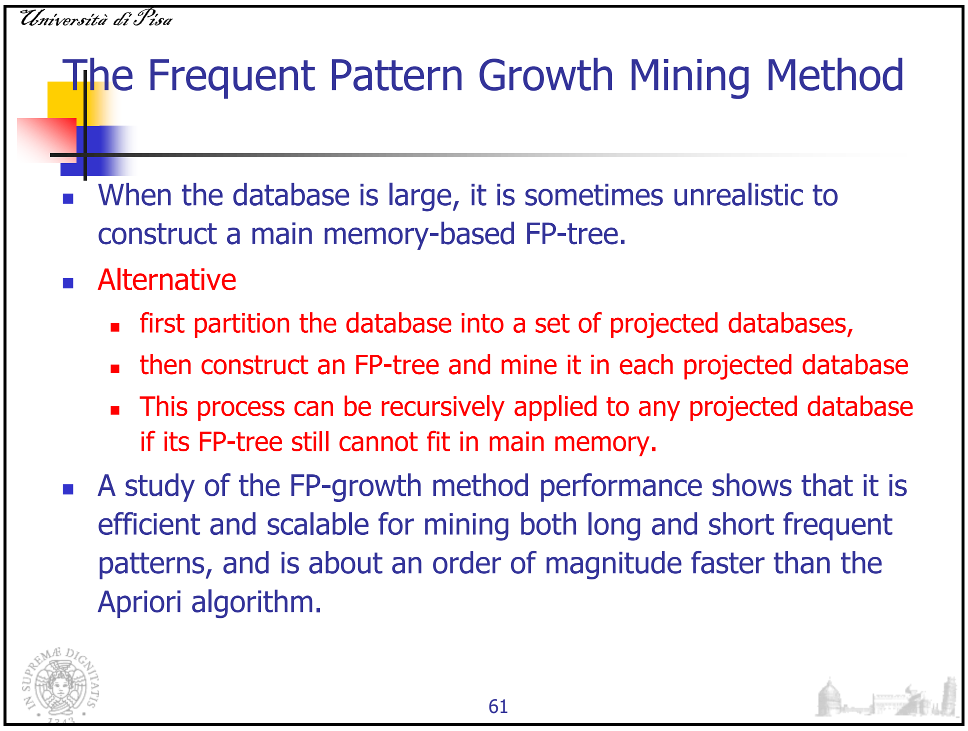
**2.** The FP-tree is mined by calling FP_growth(*FP_tree, null*), which is implemented as follows.

```
procedure FP_growth(Tree, $\alpha$)
(1)   if Tree contains a single path P then
(2)       for each combination (denoted as $\beta$) of the nodes in the path P
(3)           generate pattern $\beta \cup \alpha$ with support_count = minimum support count of nodes in $\beta$;
(4)   else for each $a_i$ in the header of Tree {
(5)       generate pattern $\beta = a_i \cup \alpha$ with support_count = $a_i$.support_count;
(6)       construct $\beta$'s conditional pattern base and then $\beta$'s conditional FP_tree $Tree_\beta$;
(7)       if $Tree_\beta \neq \emptyset$ then
(8)           call FP_growth($Tree_\beta$, $\beta$); }
```



## Benefits of the FP-tree Structure

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

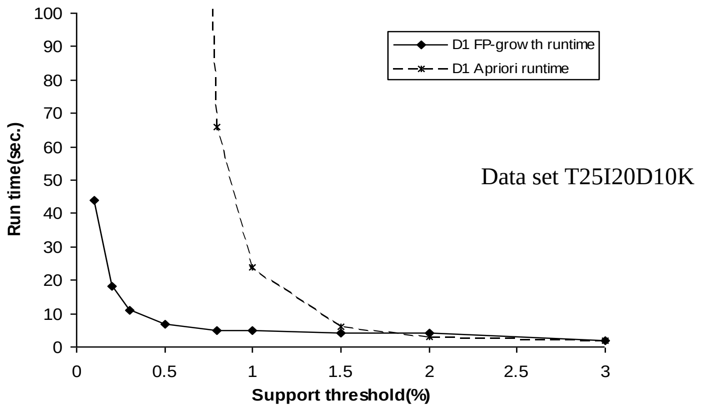


## The Frequent Pattern Growth Mining Method

- When the database is large, it is sometimes unrealistic to construct a main memory-based FP-tree.
- Alternative
  - first partition the database into a set of projected databases,
  - then construct an FP-tree and mine it in each projected database
  - This process can be recursively applied to any projected database if its FP-tree still cannot fit in main memory.
- A study of the FP-growth method performance shows that it is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm.

Università di Pisa

## Performance of FPGrowth in Large Datasets



FP-Growth vs. Apriori

Università di Pisa

## Scalable Frequent Itemset Mining Methods

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

## ECLAT: Mining by Exploring Vertical Data Format

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

## ECLAT: Mining by Exploring Vertical Data Format

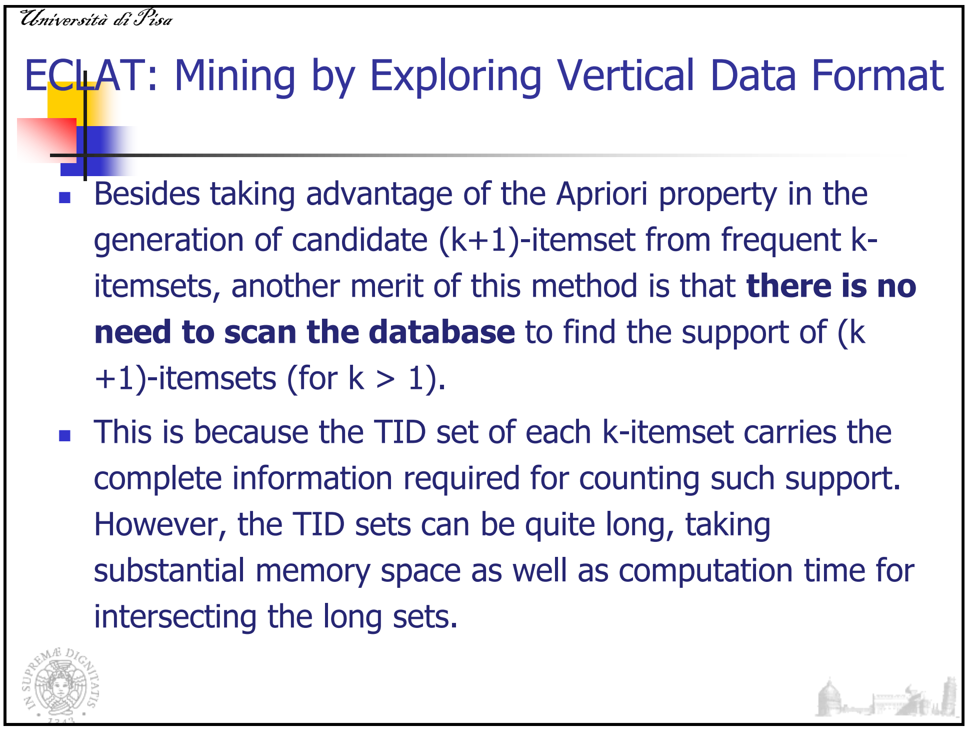
| *itemset* | *TID_set* |
|---|---|
| I1 | {T100, T400, T500, T700, T800, T900} |
| I2 | {T100, T200, T300, T400, T600, T800, T900} |
| I3 | {T300, T500, T600, T700, T800, T900} |
| I4 | {T200, T400} |
| I5 | {T100, T800} |

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} |

3-Itemsets in Vertical Data Format

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

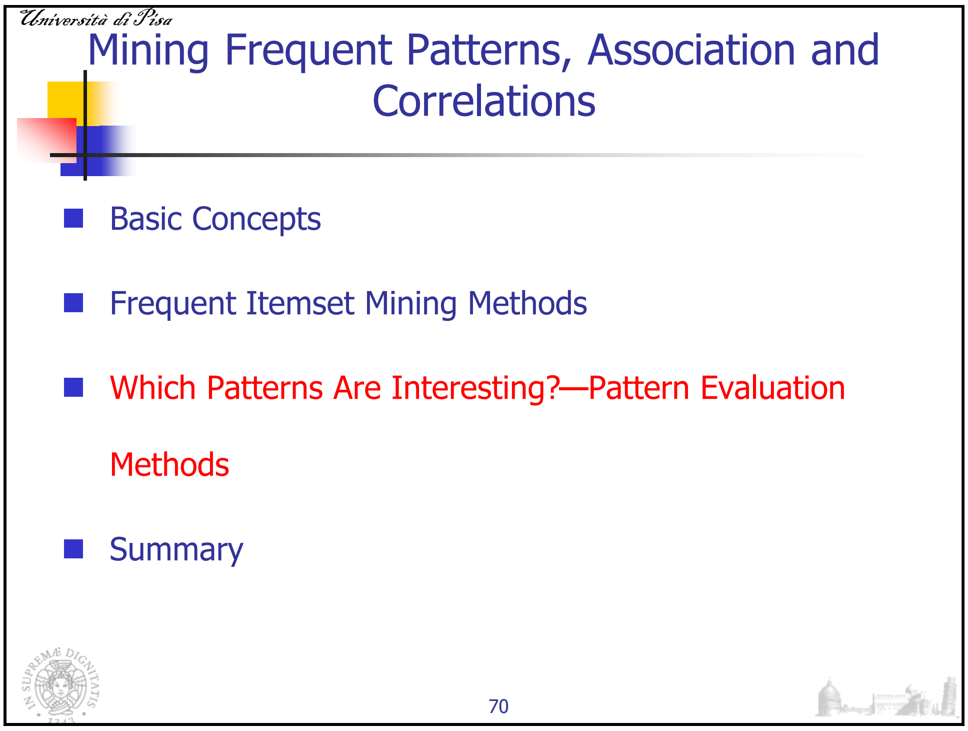


Università di Pisa

## ECLAT: Mining by Exploring Vertical Data Format

- Besides taking advantage of the Apriori property in the generation of candidate (k+1)-itemset from frequent k-itemsets, another merit of this method is that **there is no need to scan the database** to find the support of (k+1)-itemsets (for k > 1).
- This is because the TID set of each k-itemset carries the complete information required for counting such support. However, the TID sets can be quite long, taking substantial memory space as well as computation time for intersecting the long sets.

69



Università di Pisa

## Mining Frequent Patterns, Association and Correlations

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

70

70

Università di Pisa

## Mining Frequent Patterns, Association and Correlations

- 10000 customer transactions analyzed
  - 6000 include computer games
  - 7500 include videos
  - 4000 include computer games and video

$$buys(X, \text{“computer games”}) \Rightarrow buys(X, \text{“videos”})$$
$$[support = 40\%, confidence = 66\%].$$

- **The rule is misleading** because the probability of purchasing videos is 75%, which is even larger than 66%. Computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other

71

71

Università di Pisa

## Interestingness Measure: Correlations (Lift)

- Lift
  - Introduced by S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (ACM SIGMOD '97), pages 265-276, 1997.
  - lift(A -> B) = lift(B -> A) = conf(A -> B)/supp(B) = conf(B -> A)/supp(A) = P(A and B)/(P(A)P(B))
  - Lift measures how many times more often X and Y occur together than expected if they were statistically independent.
  - Lift is not down-ward closed and does not suffer from the rare item problem.
  - Rare itemsets with low counts (low probability) which per chance occur a few times (or only once) together can produce enormous lift values.

72

72

Università di Pisa

## Some consideration on interest measures for association rules

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

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

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

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

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

73

73

Università di Pisa

## Some consideration on interest measures for association rules

- Correlation

$$\chi^2 = \sum \frac{(observed - expected)^2}{expected} = \frac{(2000-2250)^2}{2250} +$$

$$+ \frac{(1750-1500)^2}{1500} + \frac{(1000-750)^2}{750} + \frac{(250-500)^2}{500} = 277.76$$

Negatively correlated

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

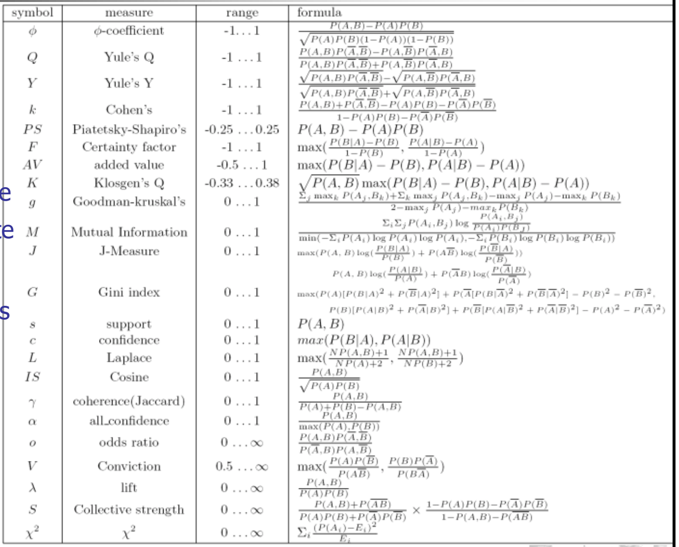
74

74

*Università di Pisa*

## Are *lift* and $\chi^2$ Good Measures of Correlation?

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



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

75

*Università di Pisa*

## Measures used for comparison

- All_confidence [0,1]

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

- Max_confidence [0,1]

$$max\_conf(A,B) = max\{P(A\,|\,B), P(B\,|\,A)\}$$

- Kulczynski [0,1] $\quad Kulc(A,B) = \frac{1}{2}(P(A|B) + P(B|A))$

- Cosine [0,1] $\quad cosine(A,B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$

$$= \sqrt{P(A|B) \times P(B|A)}.$$

76

Università di Pisa

## Null-Invariant Measures

2 × 2 Contingency Table for Two Items

| | milk | $\overline{milk}$ | $\Sigma_{row}$ |
|---|---|---|---|
| coffee | $mc$ | $\overline{m}c$ | $c$ |
| $\overline{coffee}$ | $m\overline{c}$ | $\overline{mc}$ | $\overline{c}$ |
| $\Sigma_{col}$ | $m$ | $\overline{m}$ | $\Sigma$ |

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

Null-invariant

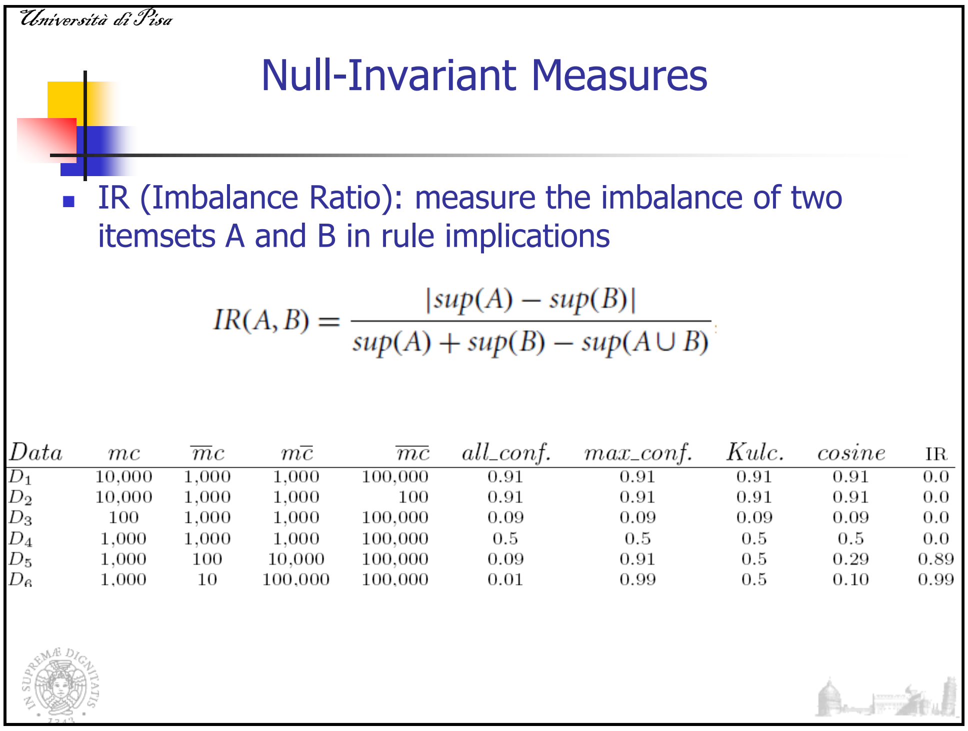
| Data Set | $mc$ | $\overline{m}c$ | $m\overline{c}$ | $\overline{mc}$ | $\chi^2$ | lift | all_conf | max_conf. | Kulc. | cosine |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | 10,000 | 1000 | 1000 | 100,000 | 90557 | 9.26 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_2$ | 10,000 | 1000 | 1000 | 100 | 0 | 1 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_3$ | 100 | 1000 | 1000 | 100,000 | 670 | 8.44 | 0.09 | 0.09 | 0.09 | 0.09 |
| $D_4$ | 1000 | 1000 | 1000 | 100,000 | 24740 | 25.75 | 0.5 | 0.5 | 0.5 | 0.5 |
| $D_5$ | 1000 | 100 | 10,000 | 100,000 | 8173 | 9.18 | 0.09 | 0.91 | 0.5 | 0.29 |
| $D_6$ | 1000 | 10 | 100,000 | 100,000 | 965 | 1.97 | 0.01 | 0.99 | 0.5 | 0.10 |

Subtle: They disagree

Università di Pisa

## Null-Invariant Measures

- "Why are lift and X² so poor at distinguishing pattern association relationships in the previous transactional data sets?"
- A null-transaction is a transaction that does not contain any of the itemsets being examined.
- A measure is null-invariant if its value is free from the influence of null-transactions.
  - Null-invariance is an important property for measuring association patterns in large transaction databases
  - Among the six discussed measures, only lift and $X^2$ are not null-invariant measures.



Università di Pisa

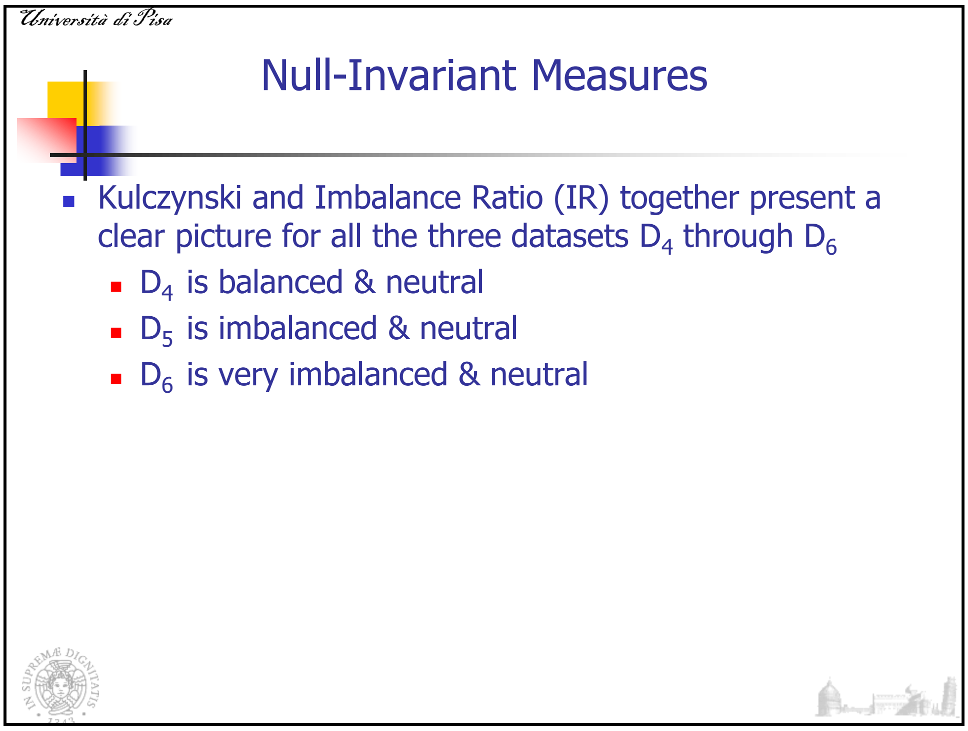
## Null-Invariant Measures

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

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

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

79

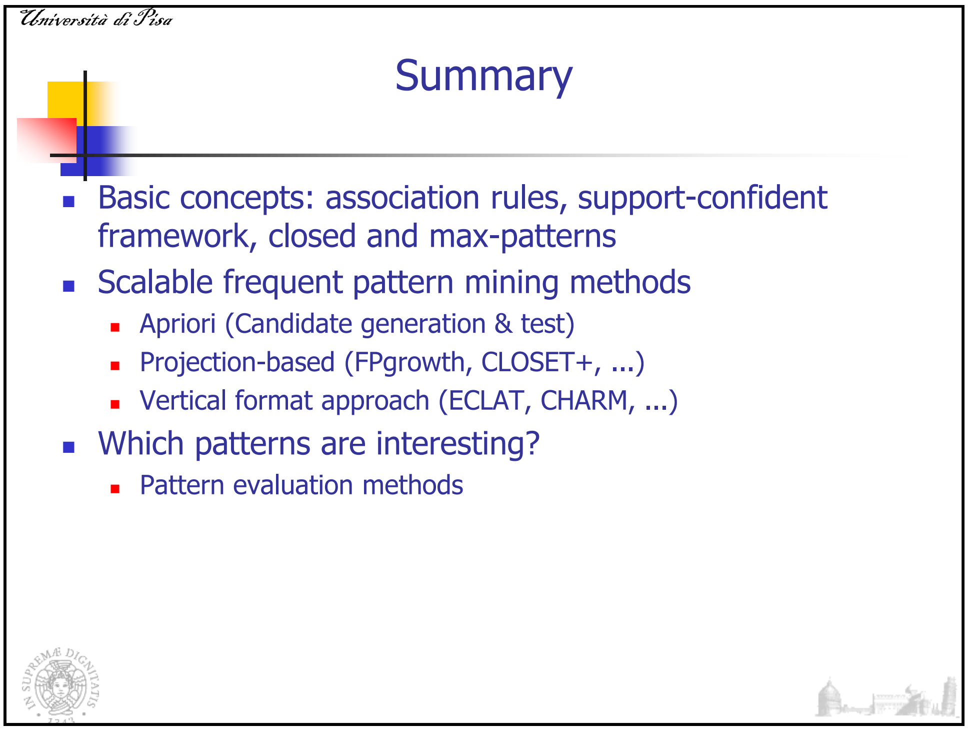


Università di Pisa

## Null-Invariant Measures

- 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

80


Università di Pisa

## 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

81