

## Association Rules Fundamentals



Elena Baralis, Silvia Chiusano  
Politecnico di Torino



### Association rules

#### Objective

- extraction of frequent correlations or pattern from a transactional database

Tickets at a supermarket counter

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk

#### Association rule

diapers  $\Rightarrow$  beer

- 2% of transactions contains both items
- 30% of transactions containing diapers also contains beer



2



### Association rule mining

- A collection of transactions is given
  - a transaction is a set of items
  - items in a transaction are not ordered
- Association rule
  - $A, B \Rightarrow C$ 
    - A, B = items in the **rule body**
    - C = item in the **rule head**
- The  $\Rightarrow$  means co-occurrence
  - not causality
- Examples
  - cereals, cookies  $\Rightarrow$  milk
  - age < 40, life-insurance = yes  $\Rightarrow$  children = yes
  - customer, relationship  $\Rightarrow$  data, mining



3



### Definitions

- Itemset** is a set including one or more items
  - Example: {Beer, Diapers}
- k-itemset** is an itemset that contains k items
- Support count** (#) is the frequency of occurrence of an itemset
  - Example:  $\#(\{\text{Beer}, \text{Diapers}\}) = 2$
- Support** is the fraction of transactions that contain an itemset
  - Example:  $\text{sup}(\{\text{Beer}, \text{Diapers}\}) = 2/5$
- Frequent itemset** is an itemset whose support is greater than or equal to a **minsup** threshold

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk



4



### Rule quality metrics

- Given the association rule
  - $A \Rightarrow B$ 
    - A, B are itemsets
- Support** is the fraction of transactions containing both A and B
  - $$\frac{\# \{A, B\}}{|T|}$$
    - |T| is the cardinality of the transactional database
    - a priori probability of itemset AB
    - rule frequency in the database
- Confidence** is the frequency of B in transactions containing A
  - $$\frac{\text{sup}(A, B)}{\text{sup}(A)}$$
    - conditional probability of finding B having found A
    - "strength" of the " $\Rightarrow$ "



5



### Rule quality metrics: example

- From itemset {Milk, Diapers} the following rules may be derived
- Rule: Milk  $\Rightarrow$  Diapers
  - support
    - $\text{sup} = \# \{ \text{Milk}, \text{Diapers} \} / \# \text{trans.} = 3/5 = 60\%$
  - confidence
    - $\text{conf} = \# \{ \text{Milk}, \text{Diapers} \} / \# \{ \text{Milk} \} = 3/4 = 75\%$
- Rule: Diapers  $\Rightarrow$  Milk
  - same support
    - $s = 60\%$
  - confidence
    - $\text{conf} = \# \{ \text{Milk}, \text{Diapers} \} / \# \{ \text{Diapers} \} = 3/3 = 100\%$

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk



6

### Association rule extraction

- Given a set of transactions  $T$ , association rule mining is the extraction of the rules satisfying the constraints
  - support  $\geq \text{minsup}$  threshold
  - confidence  $\geq \text{minconf}$  threshold
- The result is
  - complete (*all* rules satisfying both constraints)
  - correct (*only* the rules satisfying both constraints)
- May add other more complex constraints

DBG 7

### Association rule extraction

- Brute-force approach
  - enumerate all possible permutations (i.e., association rules)
  - compute support and confidence for each rule
  - prune the rules that do not satisfy the *minsup* and *minconf* constraints
- Computationally *unfeasible*
- Given an itemset, the extraction process may be split
  - first generate frequent itemsets
  - next generate rules from each frequent itemset
- Example
  - Itemset  
 $\{\text{Milk, Diapers}\}$  sup=60%
  - Rules  
 $\text{Milk} \Rightarrow \text{Diapers}$  (conf=75%)  
 $\text{Diapers} \Rightarrow \text{Milk}$  (conf=100%)

DBG 8

### Association rule extraction

- Extraction of frequent itemsets
  - many different techniques
    - level-wise approaches (Apriori, ...)
    - approaches without candidate generation (FP-growth, ...)
    - other approaches
  - most computationally expensive step
    - limit extraction time by means of support threshold
- Extraction of association rules
  - generation of all possible binary partitioning of each frequent itemset
  - possibly enforcing a confidence threshold

DBG 9

### Frequent Itemset Generation

Given  $d$  items, there are  $2^d$  possible candidate itemsets

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

DBG 10

### Frequent Itemset Generation

- Brute-force approach
  - each itemset in the lattice is a *candidate* frequent itemset
  - scan the database to count the support of each candidate
    - match each transaction against every candidate
  - Complexity  $\sim O(|T| \cdot 2^d \cdot w)$ 
    - $|T|$  is number of transactions
    - $d$  is number of items
    - $w$  is transaction length

DBG 11

### Improving Efficiency

- Reduce the *number of candidates*
  - Prune the search space
    - complete set of candidates is  $2^d$
- Reduce the *number of transactions*
  - Prune transactions as the size of itemsets increases
    - reduce  $|T|$
- Reduce the *number of comparisons*
  - Equal to  $|T| \cdot 2^d$
  - Use efficient data structures to store the candidates or transactions

DBG 12

### The Apriori Principle

*"If an itemset is frequent, then all of its subsets must also be frequent"*

- The support of an itemset can never exceed the support of any of its subsets
- It holds due to the antimonotone property of the support measure
  - Given two arbitrary itemsets A and B  
if  $A \subseteq B$  then  $\text{sup}(A) \geq \text{sup}(B)$
- It reduces the number of candidates

DBG 13

### The Apriori Principle

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Found to be Infrequent

Pruned supersets

DBG 14

### Apriori Algorithm [Agr94]

- Level-based approach
  - at each iteration extracts itemsets of a given length k
- Two main steps for each level
  - (1) Candidate generation
    - Join Step
      - generate candidates of length k+1 by joining frequent itemsets of length k
    - Prune Step
      - apply Apriori principle: prune length k+1 candidate itemsets that contain at least one k-itemset that is not frequent
  - (2) Frequent itemset generation
    - scan DB to count support for k+1 candidates
    - prune candidates below minsup

DBG 15

### Apriori Algorithm [Agr94]

- 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  $C_{k+1}$  satisfying minsup
  end
return  $\cup_k L_k$ ;

```

DBG 16

### Generating Candidates

- Sort  $L_k$  candidates in lexicographical order
- For each candidate of length k
  - Self-join with each candidate sharing same  $L_{k-1}$  prefix
  - Prune candidates by applying Apriori principle
- Example: given  $L_3 = \{abc, abd, acd, ace, bcd\}$ 
  - Self-join
    - $abcd$  from  $abc$  and  $abd$
    - $acde$  from  $acd$  and  $ace$
  - Prune by applying Apriori principle
    - $acde$  is removed because  $ade$  is not in  $L_2$
    - $C_4 = \{abcd\}$

DBG 17

### Apriori Algorithm: Example

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

minsup > 1

DBG 18

### Generate candidate 1-itemsets

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

1st DB scan

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

minsup > 1

DBG

19

### Prune infrequent candidates in $C_1$

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

1st DB scan

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

$L_1 \equiv C_1$

minsup > 1

- All itemsets in set  $C_1$  are frequent according to minsup > 1

DBG

20

### Generate candidates from $L_1$

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

→

itemsets
{A,B}
{A,C}
{A,D}
{A,E}
{B,C}
{B,D}
{B,E}
{C,D}
{C,E}
{D,E}

DBG

21

### Count support for candidates in $C_2$

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

→

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{B,E}	1
{C,D}	3
{C,E}	2
{D,E}	2

2nd DB scan

DBG

22

### Prune infrequent candidates in $C_2$

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

→

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
<del>{B,E}</del>	<del>1</del>
{C,D}	3
{C,E}	2
{D,E}	2

2nd DB scan

→

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

$L_2$

DBG

23

### Generate candidates from $L_2$

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

→

itemsets
{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{C,D,E}

$C_3$

DBG

24

### Apply Apriori principle on $C_3$

$L_2$

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

$C_3$

itemsets
{A,B,C}
{A,B,D}
<del>{A,B,E}</del>
<del>{A,C,D}</del>
<del>{A,C,E}</del>
<del>{A,D,E}</del>
<del>{B,C,D}</del>
<del>{C,D,E}</del>

- Prune {A,B,E}
- Its subset {B,E} is infrequent ({B,E} is not in  $L_2$ )

DBG 25

### Count support for candidates in $C_3$

$L_2$

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

$C_3$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
<del>{A,B,E}</del>	
<del>{A,C,D}</del>	
<del>{A,C,E}</del>	
<del>{A,D,E}</del>	
<del>{B,C,D}</del>	
<del>{C,D,E}</del>	

3rd DB scan

DBG 26

### Prune infrequent candidates in $C_3$

$L_2$

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

$C_3$

itemsets
{A,B,C}
{A,B,D}
<del>{A,B,E}</del>
<del>{A,C,D}</del>
<del>{A,C,E}</del>
<del>{A,D,E}</del>
<del>{B,C,D}</del>
<del>{C,D,E}</del>

3rd DB scan

$L_3$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

- {A,C,E} and {C,D,E} are actually infrequent
- They are discarded from  $C_3$

DBG 27

### Generate candidates from $L_3$

$L_3$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

$C_4$

itemsets
{A,B,C,D}

DBG 28

### Apply Apriori principle on $C_4$

$L_3$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

$C_4$

itemsets
{A,B,C,D}

- Check if {A,C,D} and {B,C,D} belong to  $L_3$
- $L_3$  contains all 3-itemset subsets of {A,B,C,D}
- {A,B,C,D} is potentially frequent

DBG 29

### Count support for candidates in $C_4$

$L_3$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

$C_4$

itemsets
{A,B,C,D}

4th DB scan

$C_4$

itemsets	sup
{A,B,C,D}	1

DBG 30

### Prune infrequent candidates in $C_4$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

→  $C_4$  (4th DB scan)

itemsets	sup
{A,B,C,D}	1

→  $L_4 = \emptyset$

- {A,B,C,D} is actually infrequent
- {A,B,C,D} is discarded from  $C_4$

DBG 31

### Final set of frequent itemsets

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

minsup > 1

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

$L_1$

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

$L_3$

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

$L_2$

DBG 32

### Counting Support of Candidates

- Scan transaction database to count support of each itemset
  - total number of candidates may be large
  - one transaction may contain many candidates
- Approach [Agr94]
  - candidate itemsets are stored in a *hash-tree*
    - leaf node of hash-tree contains a list of itemsets and counts
    - interior node contains a hash table
  - subset function finds all candidates contained in a transaction
    - match transaction subsets to candidates in hash tree

DBG 33

### Performance Issues in Apriori

- Candidate generation
  - Candidate sets may be huge
    - 2-itemset candidate generation is the most critical step
    - extracting long frequent itemsets requires generating all frequent subsets
- Multiple database scans
  - $n + 1$  scans when longest frequent pattern length is  $n$

DBG 34

### Factors Affecting Performance

- Minimum support threshold
  - lower support threshold increases number of frequent itemsets
    - larger number of candidates
    - larger (max) length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases in dense data sets
  - may increase max length of frequent itemsets and traversals of hash tree
    - number of subsets in a transaction increases with its width

DBG 35

### Improving Apriori Efficiency

- Hash-based itemset counting [Yu95]
  - A  $k$ -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction [Yu95]
  - A transaction that does not contain any frequent  $k$ -itemset is useless in subsequent scans
- Partitioning [Sav96]
  - Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

DBG 36

### Improving Apriori Efficiency

- Sampling [Toi96]
  - mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic Itemset Counting [Motw98]
  - add new candidate itemsets only when all of their subsets are estimated to be frequent

DBG 37

### FP-growth Algorithm [Han00]

- Exploits a main memory compressed representation of the database, the FP-tree
  - high compression for dense data distributions
    - less so for sparse data distributions
  - complete representation for frequent pattern mining
    - enforces support constraint
- Frequent pattern mining by means of FP-growth
  - recursive visit of FP-tree
  - applies divide-and-conquer approach
    - decomposes mining task into smaller subtasks
- Only two database scans
  - count item supports + build FP-tree

DBG 38

### FP-tree construction

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

minsups > 1

- (1) Count item support and prune items below minsup threshold
- (2) Build Header Table by sorting items in decreasing support order

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

DBG 39

### FP-tree construction

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

minsups > 1

- (1) Count item support and prune items below minsup threshold
- (2) Build Header Table by sorting items in decreasing support order
- (3) Create FP-tree
  - For each transaction  $t$  in DB
    - order transaction  $t$  items in decreasing support order
      - same order as Header Table
    - insert transaction  $t$  in FP-tree
      - use existing path for common prefix
      - create new branch when path becomes different

DBG 40

### FP-tree construction

Transaction

TID	Items
1	{A,B}

Sorted transaction

TID	Items
1	{B,A}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree

```

graph TD
    Root["{}"] --> B["B:1"]
    B --> A["A:1"]
  
```

DBG 41

### FP-tree construction

Transaction

TID	Items
2	{B,C,D}

Sorted transaction

TID	Items
2	{B,C,D}

Header Table

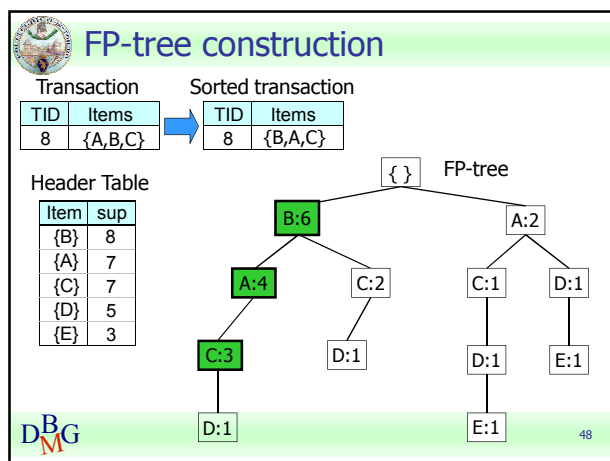
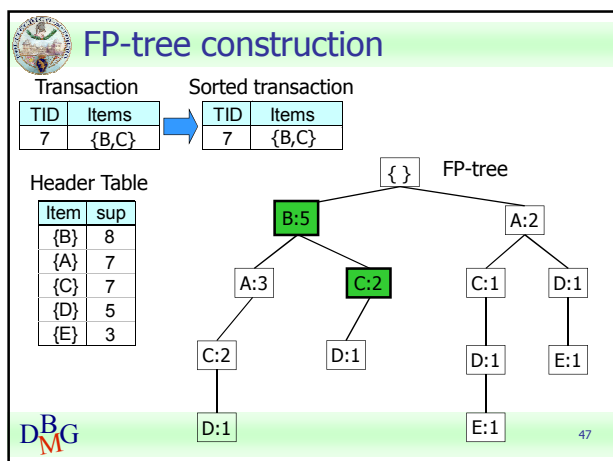
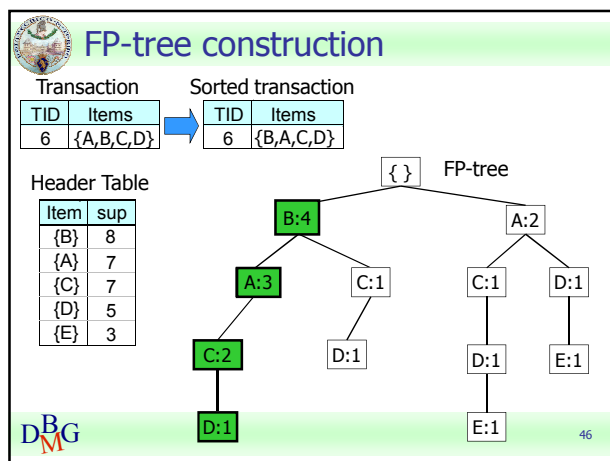
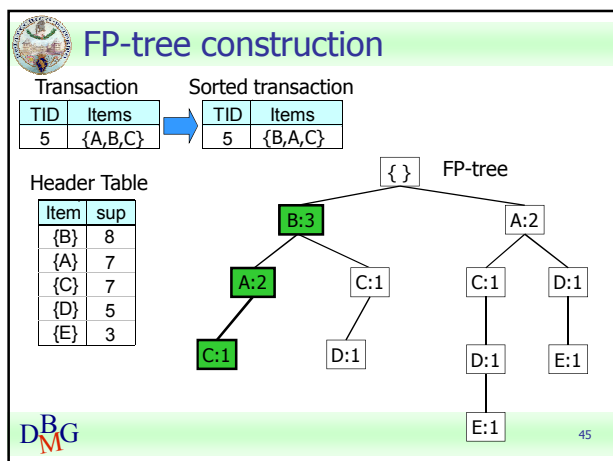
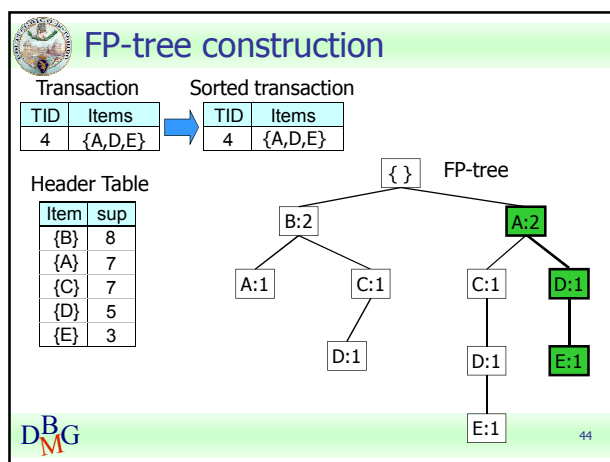
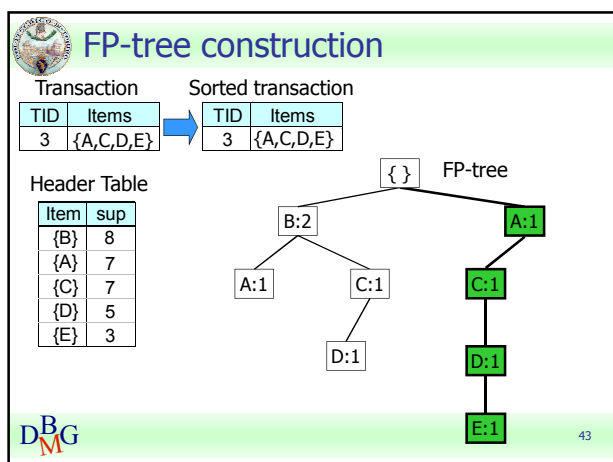
Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree

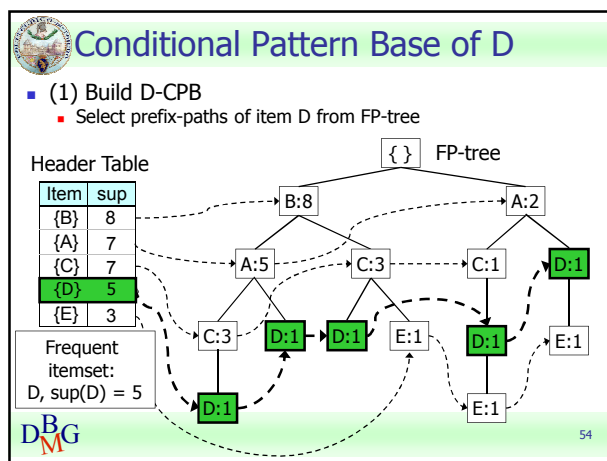
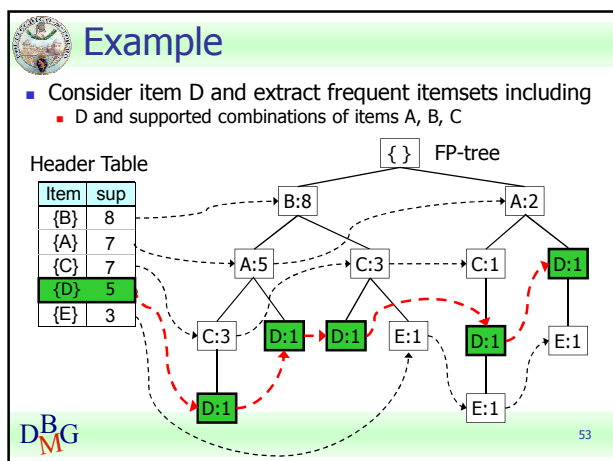
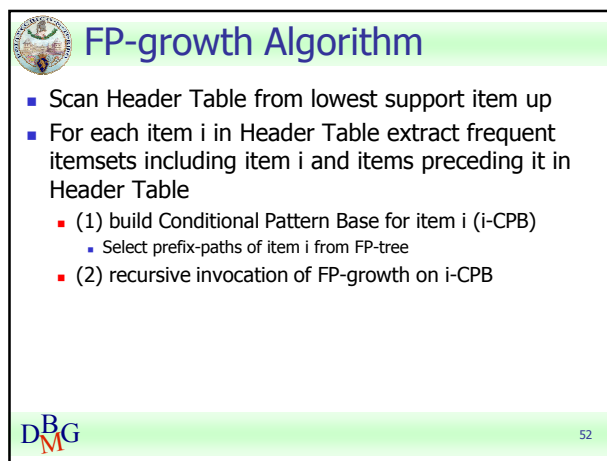
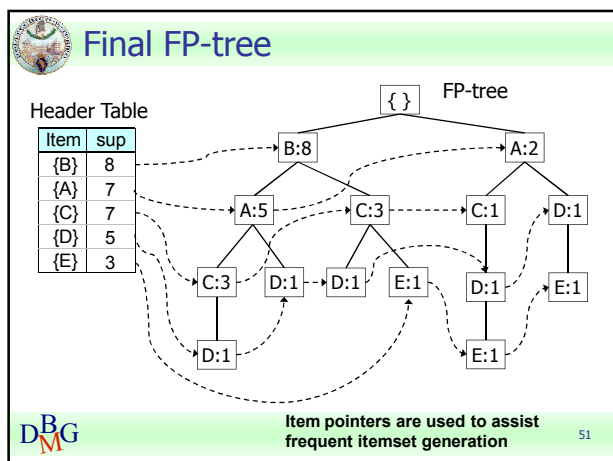
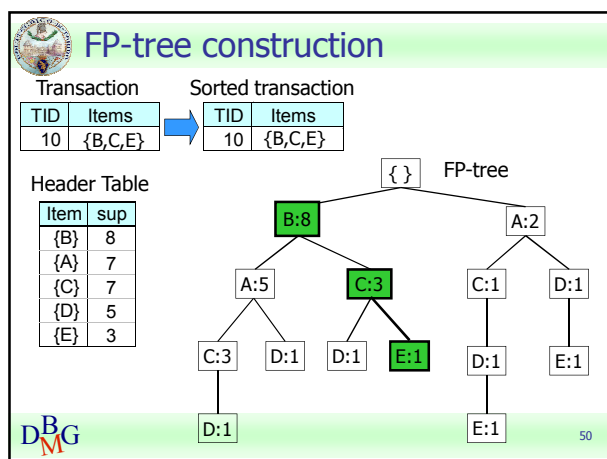
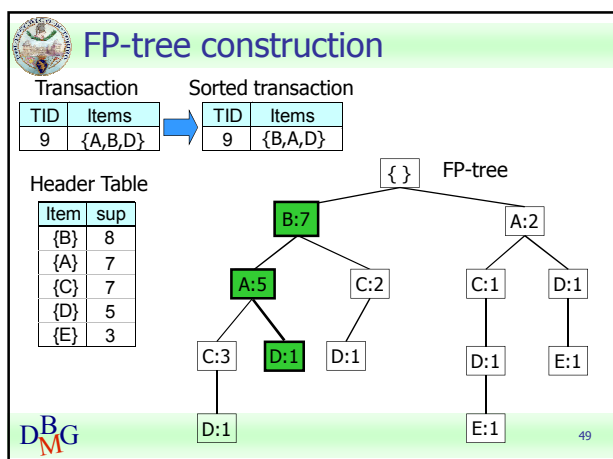
```

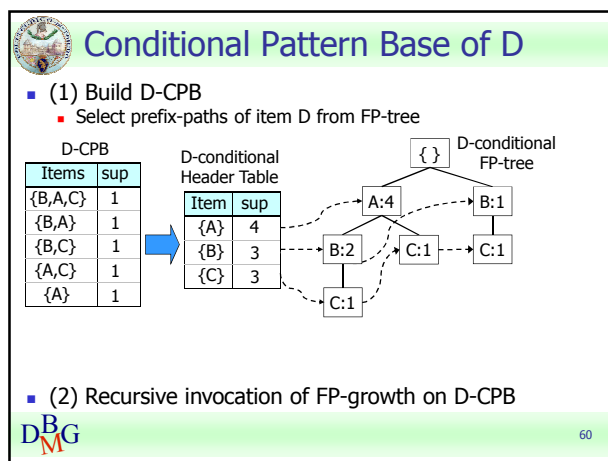
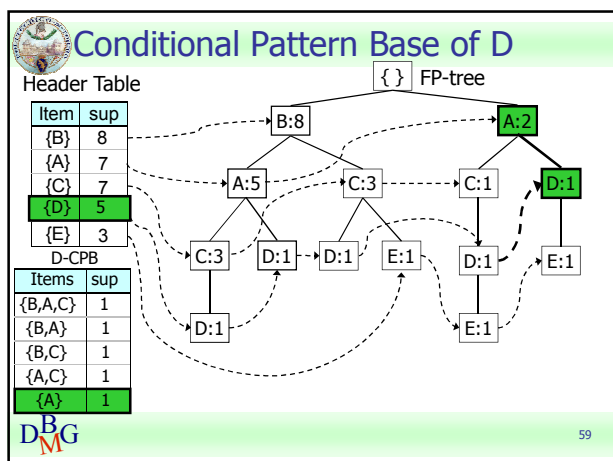
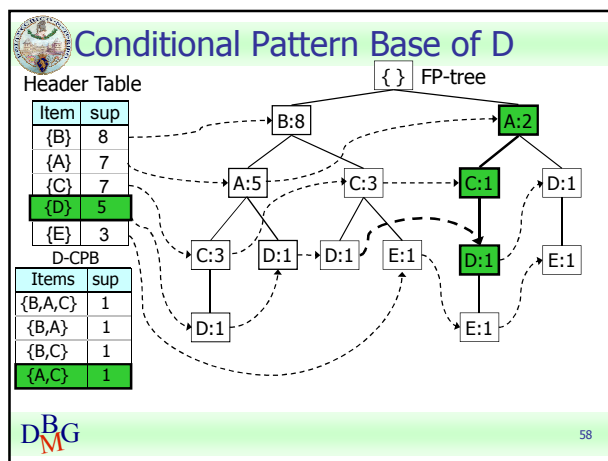
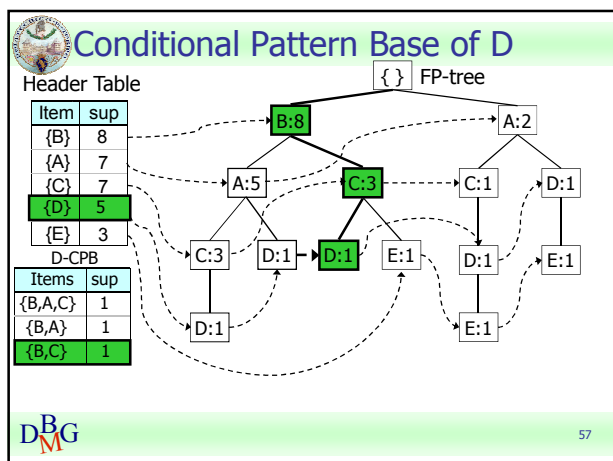
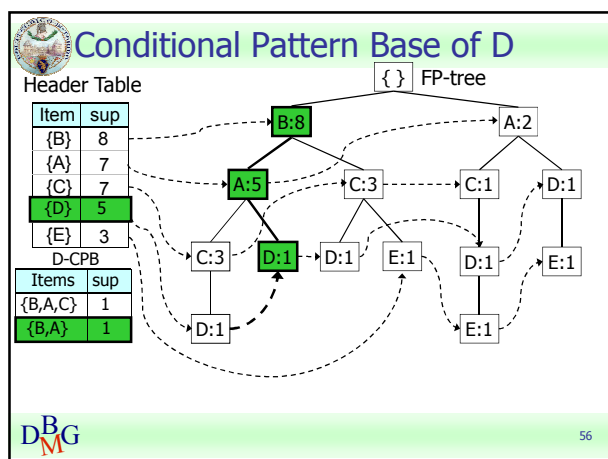
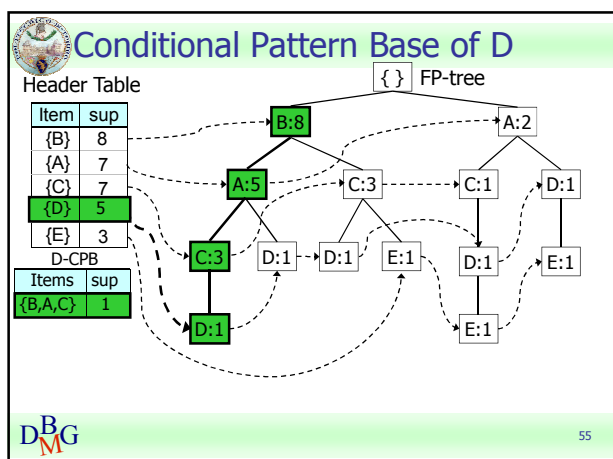
graph TD
    Root["{}"] --> B["B:2"]
    B --> A["A:1"]
    B --> C["C:1"]
    C --> D["D:1"]
  
```

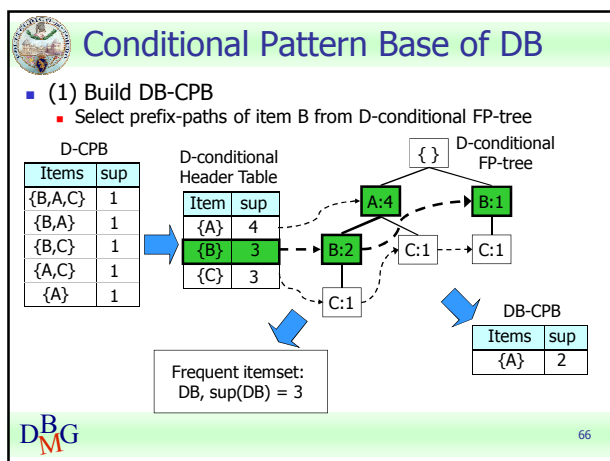
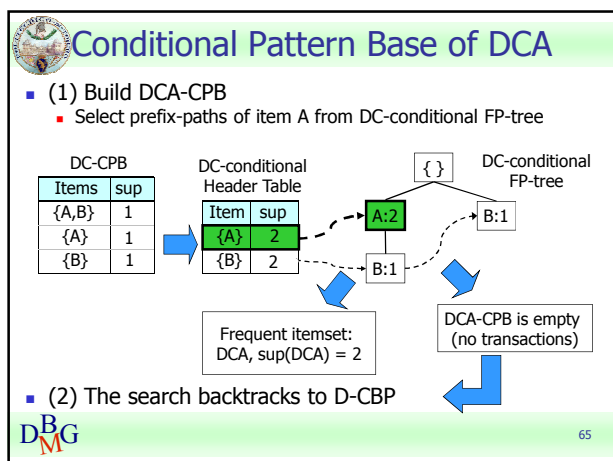
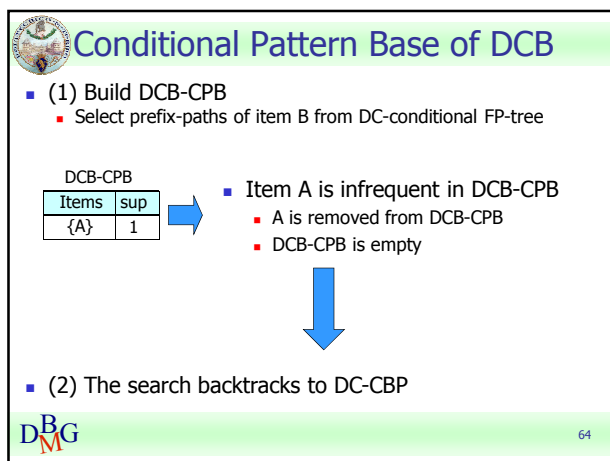
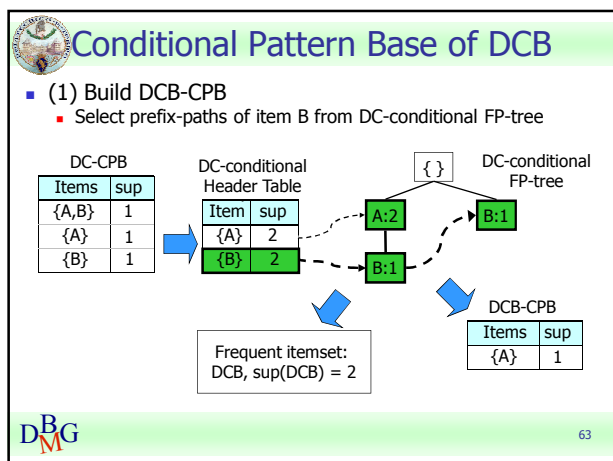
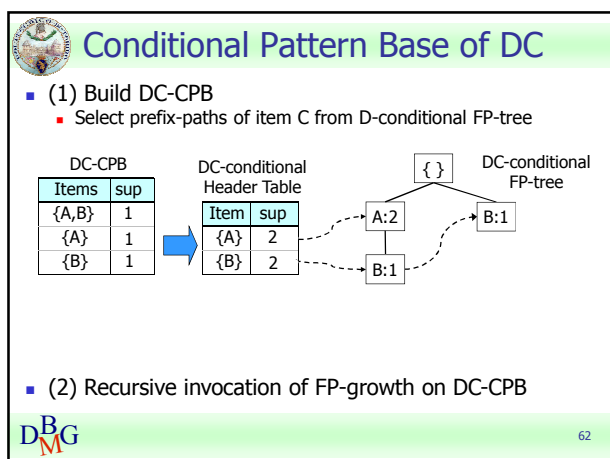
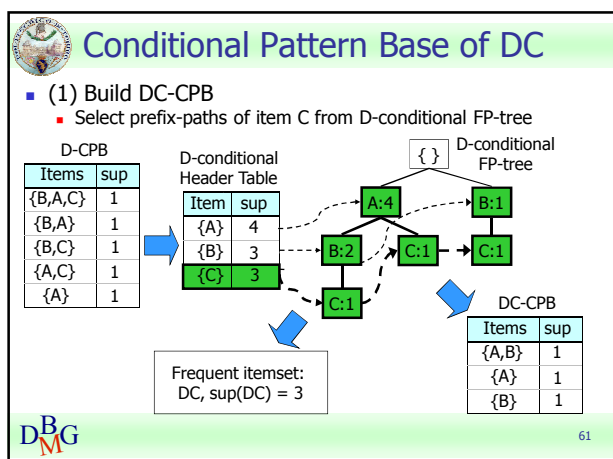
DBG 42











### Conditional Pattern Base of DB

- (1) Build DB-CPB
  - Select prefix-paths of item B from D-conditional FP-tree

DB-CPB

Items	sup
{A}	2

DB-conditional Header Table

Items	sup
{A}	2

DB-conditional FP-tree

{}

A:2

- (2) Recursive invocation of FP-growth on DB-CPB

DBG

### Conditional Pattern Base of DBA

- (1) Build DBA-CPB
  - Select prefix-paths of item A from DB-conditional FP-tree

DB-CPB

Items	sup
{A}	2

DB-conditional Header Table

Items	sup
{A}	2

DB-conditional FP-tree

{}

A:2

Frequent itemset: DBA, sup(DBA) = 2

DBA-CPB is empty (no transactions)

The search backtracks to D-CPB

- (2) The search backtracks to D-CPB

DBG

### Conditional Pattern Base of DA

- (1) Build DA-CPB
  - Select prefix-paths of item A from D-conditional FP-tree

D-CPB

Items	sup
{B,A,C}	1
{B,A}	1
{B,C}	1
{A,C}	1
{A}	1

D-conditional Header Table

Item	sup
{A}	4
{B}	3
{C}	3

D-conditional FP-tree

{}

A:4

B:1

B:2

C:1

C:1

DA-CPB is empty (no transactions)

The search ends

Frequent itemset: DA, sup(DA) = 4

DBG

### Frequent itemsets with prefix D

- Frequent itemsets including D and supported combinations of items B,A,C

Example DB

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

minsup > 1

itemsets	sup
{D}	5
{A,D}	4
{B,D}	3
{C,D}	3
{A,B,D}	2
{A,C,D}	2
{B,C,D}	2

DBG

### Other approaches

- Many other approaches to frequent itemset extraction
  - some covered later
- May exploit a different database representation
  - represent the tidset of each item [Zak00]

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

	A	B	C	D	E
1	1	1	2	2	1
2	4	2	3	4	3
3	5	5	4	5	6
4	6	7	8	9	
5	7	8	9		
6	8	10			
7	9				
8					
9					
10					

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

DBG

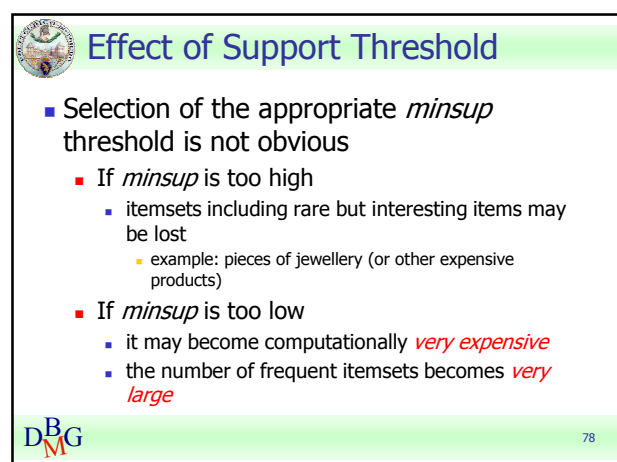
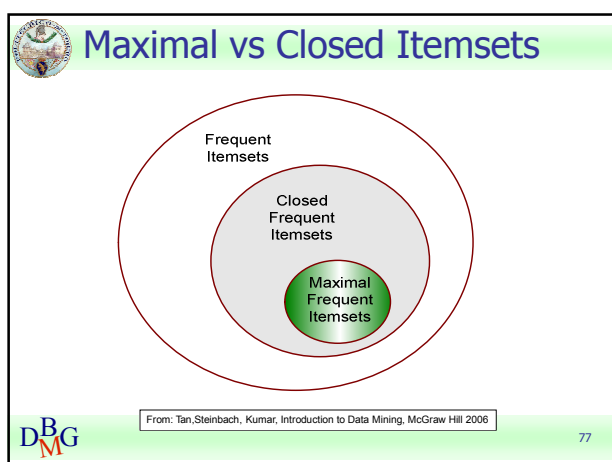
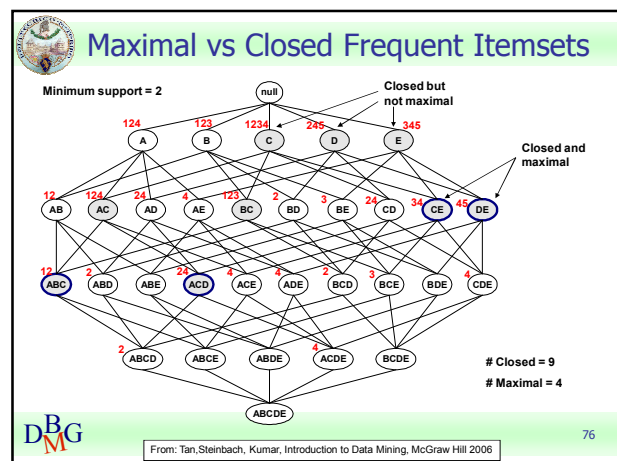
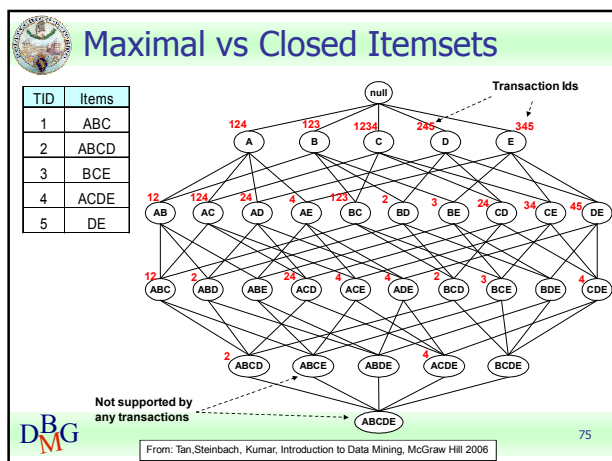
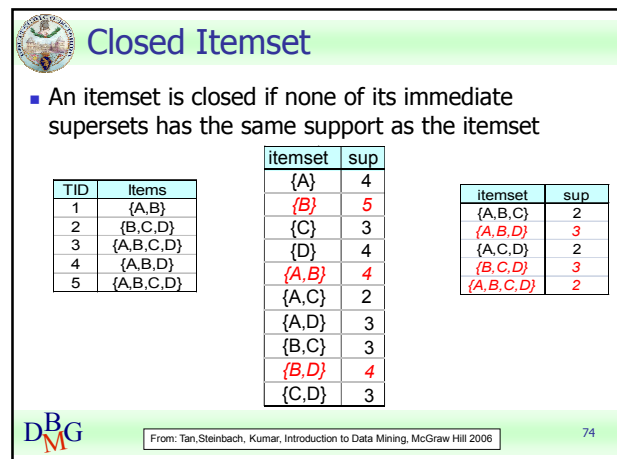
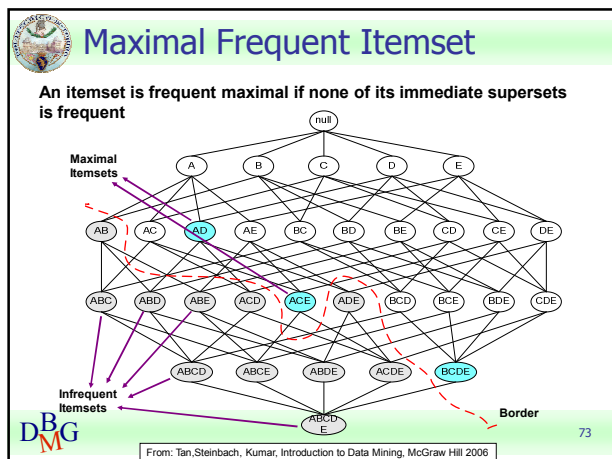
### Compact Representations

- Some itemsets are redundant because they have identical support as their supersets

- Number of frequent itemsets  $= 3 \times \sum_{k=1}^m \binom{10}{k}$
- A compact representation is needed

DBG

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



### Interestingness Measures

- A large number of pattern may be extracted
  - rank patterns by their interestingness
- Objective measures
  - rank patterns based on statistics computed from data
  - initial framework [Agr94] only considered support and confidence
    - other statistical measures available
- Subjective measures
  - rank patterns according to user interpretation [Silb98]
    - interesting if it contradicts the expectation of a user
    - interesting if it is actionable

DBG 79

### Confidence measure: always reliable?

- 5000 high school students are given
  - 3750 eat cereals
  - 3000 play basket
  - 2000 eat cereals and play basket
- Rule
 

play basket  $\Rightarrow$  eat cereals  
sup = 40%, conf = 66,7%

is misleading because eat cereals has sup 75% (>66,7%)
- Problem caused by high frequency of rule head
  - negative correlation

	basket	not basket	total
cereals	2000	1750	3750
not cereals	1000	250	1250
total	3000	2000	5000

DBG 80

### Correlation or lift

$r: A \Rightarrow B$

$$\text{Correlation} = \frac{P(A,B)}{P(A)P(B)} = \frac{\text{conf}(r)}{\text{sup}(B)}$$

- Statistical independence
  - Correlation = 1
- Positive correlation
  - Correlation > 1
- Negative correlation
  - Correlation < 1

DBG 81

### Example

- Association rule
 

play basket  $\Rightarrow$  eat cereals

has corr = 0.89

  - negative correlation
- but rule
 

play basket  $\Rightarrow$  not (eat cereals)

has corr = 1,34

DBG 82

#	Measure	Formula
1	$\phi$ -coefficient	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's $\lambda$	$\frac{\sum_{i=1}^m \max_j P(a_{ij}, B_k) - \sum_{i=1}^m \max_j P(a_{ij}, B_k) - \max_k P(a_{ik})}{\sum_{i=1}^m \max_j P(a_{ij}, B_k) - \max_k P(a_{ik})}$
3	Odds ratio ( $\alpha$ )	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(A,\bar{B})P(\bar{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(A,\bar{B})P(\bar{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}} = \frac{\alpha-1}{\alpha+1}$
6	Kappa ( $\kappa$ )	$\frac{P(A,B) - P(A)P(B)}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
7	Mutual Information ( $M$ )	$\sum_{i=1}^m \sum_{j=1}^n P(a_{ij}) \log \frac{P(a_{ij})}{P(A_i)P(B_j)}$
8	J-Measure ( $J$ )	$\max \left( P(A,B) \log \left( \frac{P(A,B)}{P(A)P(B)} \right) + P(\bar{A},\bar{B}) \log \left( \frac{P(\bar{A},\bar{B})}{P(\bar{A})P(\bar{B})} \right), P(A,\bar{B}) \log \left( \frac{P(A,\bar{B})}{P(A)P(\bar{B})} \right) + P(\bar{A},B) \log \left( \frac{P(\bar{A},B)}{P(\bar{A})P(B)} \right) \right)$
9	Gini index ( $G$ )	$\max \left( P(A)[P(A A)^2 + P(B A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2, P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2 \right)$
10	Support ( $s$ )	$P(A,B)$
11	Confidence ( $c$ )	$\frac{P(A,B)}{P(A)}$
12	Laplace ( $L$ )	$\frac{P(A,B) + 1}{N P(A) + 2}$
13	Conviction ( $V$ )	$\frac{P(A,\bar{B})}{P(A)P(\bar{B})} - \frac{P(A,B)}{P(A)P(B)}$
14	Interest ( $I$ )	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine ( $IS$ )	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's ( $PS$ )	$P(A,B) - P(A)P(B)$
17	Certainty factor ( $F$ )	$\frac{P(A,B) - P(A)P(B)}{1 - P(A)}$
18	Added Value ( $AV$ )	$\max(P(B A) - P(B), P(A B) - P(A))$
19	Collective strength ( $S$ )	$\frac{P(A,B) - P(A)P(B)}{P(A)P(B) - P(A)P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
20	Jaccard ( $J$ )	$\frac{P(A,B)}{P(A) + P(B) - P(A,B)}$
21	Kingem ( $K$ )	$\sqrt{P(A,B) \max(P(B A) - P(B), P(A B) - P(A))}$

DBG