Association Rules

Data mining and data warehousing

Association Rule Mining

Topics

- Basic concepts of Association Rules
- Rule strength measures
- Basic Algorithms
    - Apriori Algorithm
    - FP-Growth Algorithm
    - Other Approaches
    - Interestingness Measures
    - Sequential Pattern Mining
- Summary

Association rule mining

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Clothes and Milk !
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to new drug?
  - Can we automatically recommend next web document?
- Applications
  - Basket data analysis, Cross-marketing, Rack arrangement, Sale campaign analysis
  - DNA sequence analysis
  - Web log (click stream) analysis
  - planning public services (education, health, transport, funds) as well as public business(for setup new factories, shopping malls or banks and even marketing particular products) from census data.

What is Association Rule?

- Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository.

- An example of an association rule would be "If a customer buys a packet of bread, he is 80% likely to also purchase milk.“

  bread → milk

Association rule mining

- Frequent pattern
  - A pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal et al. in 1993 in the context of frequent itemsets and association rule mining
- An important data mining model studied extensively

Association rule mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction
- Initially used for Market Basket Analysis to find how items purchased by customers are related

  Bread → Milk [Sup = 5%, Conf = 70%]

The model: Data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*
- Transaction t: a set of items, and $t \subseteq I$
- Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$

Transaction data: Supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}

  t2: {apple, cake, salt, yogurt}

  … …

  tn: {biscuit, cake, milk}

- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data: a set of documents

- **Text document data set, each document is treated as a "bag" of keywords**

doc1:	Student, Teach, School
doc2:	Student, School
doc3:	Teach, School, City, Game
doc4:	Baseball, Basketball
doc5:	Basketball, Player, Spectator
doc6:	Baseball, Coach, Game, Team
doc7:	Basketball, Team, City, Game

- **Web page data set**

Session1: PageA.html, PageB.html, PageC.html

Session1: PageC.html, PageD.html, PageE.html

Session1: PageA.html, PageC.html, PageD.html

The model: Rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$
- An association rule is an implication of the form:

  $X \rightarrow Y$, where $X, Y \subset I$, *and* $X \cap Y = \emptyset$

- An itemset is a set of items
  - E.g., X = {milk, bread, cereal} is an itemset
- A *k*-itemset is an itemset with *k* items
  - E.g., {milk, bread} is a 2-itemset

  {milk, bread, cereal} is a 3-itemset

Rule Strength Measures

- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability
  - Support
  - Confidence

Support

- This measure gives an idea of how frequent an *itemset* is in all the transactions.
- *itemset1* = {bread}
- *itemset2* = {shampoo}.
    - t(bread).count >> t(shampoo).count
    - *Support(Itemset1)*> support(Itemset2)
- *itemset1* = {bread, butter}
- *itemset2* = {bread, shampoo}.
    - T(bread, butter).count >> t(bread, shampoo).count
    - *Support(Itemset1)*> support(Itemset2)

Absolute and Relative Support

- T1 = {A,A,C}
- T2 = {A,X}
- What is the support of A ? Is it 3 or 2 ?

- **absolute support** of A,
  - i.e. the absolute number of transactions which contains A, is 2
- **relative support** of A,
  - i.e. the relative number of transactions which contains A, is 2/2=1

Support

- Mathematically,
  - support : the fraction of the total number of transactions in which the itemset occurs.

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

- support helps to identify the rules tobe considered or not for further analysis
  - E,g, to consider only the itemsets which occur at least 50 times out of a total of 10,000 transactions
  - i.e. support = 0.005.
- *Itemset* with very low support do not have enough information on the among the items contained in the set
  - So no conclusions can be drawn from such a rule.

Support and Confidence

- Support
  - The rule holds with support *sup* in *T* (the transaction data set having n transactions) if sup% of transactions contain *X* ∪ *Y*
  - *sup* = Pr(*X* ∪ *Y*)

$$\text{sup} = \frac{(X \cup Y).count}{n}$$

    - Relative Support
  - The frequency count of an itemset *X U Y*, denoted by (*XUY).count*, in a data set *T* is the number of transactions
    - Count/Absolute Support

Logic behind Support

- Logic of the support calculation is
  - to consider only item(sets) which appear frequently enough **in *different* transactions**
  - **to** be sure that the resulting rules are based on an actual patterns
    - not appear due to chance (i.e. the strange behavior of just a few customers).
  - To make predictions about the likes/dislikes of future customers.

Logic behind Support

- If a pattern is based only on two customers, the applicability is ... questionable.
- E.g.
  - Customer C1 bought A a thousand times (once),
  - Customer C2 bought it just to try it out
  - Other 1000 customers (C3..C1002) visited the store but none bought A
- Absolute support for A = 2, not 1001
- Relative support = 2/1003 = 0.00192
  not 1001/1003 = 0.9990
-

Confidence

- defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents.
- E.g. To answer the question
  - of all the transactions containing say, {Potato Chips}, how many also had {Milk} on them?
  - {Potato Chips} → {Milk} should be a high or low confidence rule.
- It is the conditional probability of occurrence of consequent given the antecedent.

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

Rule strength measures

- Confidence
  - The rule holds in *T* with confidence *conf* if % of transactions that contain *X* also contain *Y*
  - *X*□ *Y*
  - *conf* = Pr(*Y* | *X*)

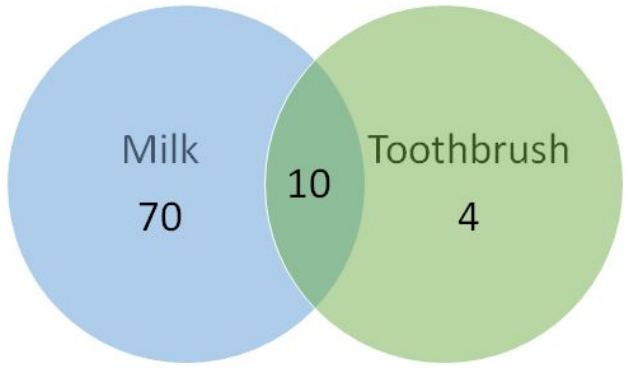
$$confidence = \frac{(X \cup Y).count}{X.count}$$

$$confidence = \frac{Support(X \cup Y)}{Support(X)}$$

- E.g Confidence of {Butter} → {Bread}
  - fraction of transactions having butter also had bread
  - Very high i.e. a value close to 1
- {Yogurt} → {Milk}? High again
- {Toothbrush} → {Milk}? Not so sure?
  - Confidence for this rule will also be high since {Milk} is such a frequent itemset and would be present in every other transaction.
  - What about {Milk} → {Toothbush}? Not so sure?

  - *It does not matter what you have in the antecedent (X) for such a frequent consequent.*
  - *The confidence for an association rule having a very frequent consequent (Y) will always be high.*

Confidence : Limitation



Total transactions = 100

both milk and toothbrush = 10

milk but no toothbrush = 70

toothbrush but no milk = 4.

- Confidence for
  - {Toothbrush} → {Milk}
  - 10/(10+4) = 0.7
- Looks like a high confidence value.
- But we know intuitively that these two products have a weak association and there is something misleading about this high confidence value.
- *Lift* is introduced to overcome this challenge.

Lift

- Lift controls for the *support* (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}.
- *Lift* is a very literal term given to this measure.
- Think of it as the *lift* that {X} provides to our confidence for having {Y} on the cart.
- To rephrase, *lift* is the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}.

Lift

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

- Lift = Confidence(X□Y) / Y.count
  - In cases where {X} actually leads to {Y} on the cart, value of lift will be greater than 1.
- {Toothbrush} → {Milk} rule.
  - Probability of having milk on the cart with the knowledge that toothbrush is present (Confidence)
  - 10/(10+4) = 0.7
- {Milk}
  - consider the probability of having milk on the cart without any knowledge about toothbrush =: 80/100 = 0.8

Lift

- having toothbrush on the cart actually reduces the probability of having milk on the cart to 0.7 from 0.8!
- This will be a lift of 0.7/0.8 = 0.87.
- Now that's more like the real picture.
  - A value of lift less than 1 shows that having toothbrush on the cart does not increase the chances of occurrence of milk on the cart in spite of the rule showing a high confidence value.
  - If lift > 1 indicates high association between {Y} and {X}.
  - More the value of lift, greater are the chances of preference to buy {Y} if the customer has already bought {X}.

  - *Lift* is the measure that will help store managers to decide product placements on aisle.

Problem of Association

- Once quantify the importance of association of products within an itemset, the next step is to generate rules from the entire list of items and identify the most important ones.
  - E.g Supermarkets will have thousands of different products in store
  - Just 10 products may l lead to 57000 rules!!
  - this number increases exponentially with the increase in number of items.
  - Finding lift values for each of these will get computationally very very expensive.
  - How to deal with this problem?
  - How to come up with a set of most important association rules to be considered?
  - *Apriori algorithm* comes to our rescue for this.

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf)
- **Key Features**
  - Completeness: find all rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds
  - Mining with data on hard disk (not in memory)

An example

t1: Bread, Cake, Milk
t2: Bread, Cheese
t3: Cheese, Boots
t4: Bread, Cake, Cheese
t5: Bread, Cake, Clothes, Cheese, Milk
t6: Cake, Clothes, Milk
t7: Cake, Milk, Clothes

- Transaction data
- Assume:
  minsup = 30%
  minconf = 80%
- An example frequent *itemset*: {Cake, Clothes, Milk} [sup = 3/7]
- Association rules from the itemset:

  Clothes ⟶ Milk, Cake [sup = 3/7, conf = 3/3]

  … …

  Clothes, Cake ⟶ Milk, [sup = 3/7, conf = 3/3]

Assumption

- A simplistic view of shopping baskets transactions
  - Some important information not considered e.g.
    - The quantity of each item purchased
    - The price paid
- Assume all data are categorical
  - Examples:
    - Item Purchased or not ?
    - ID numbers, eye color {brown, black, etc.}, zip codes
    - Height in {tall, medium, short}

Many mining algorithms

- A large number of them!!
- Use of different strategies and data structures
- Resulting sets of rules are all the same
- Computational efficiencies and memory requirements may be different

The Apriori algorithm

- **The best known algorithm**
- **Two steps**:
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets)
  - Use frequent itemsets to generate rules

- E.g., a frequent itemset

  {Cake, Clothes, Milk} [sup = 3/7]

  and one rule from the frequent itemset

  Clothes → Milk, Cake [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

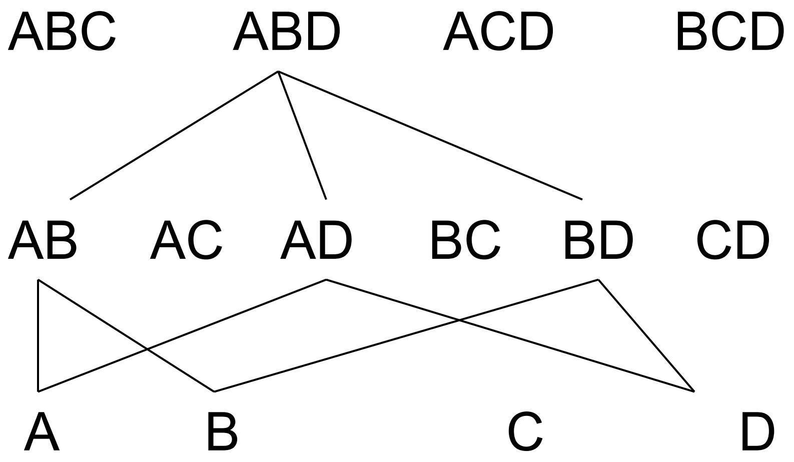
- A frequent *itemset* is an itemset whose support is ≥ minsup
- Key idea
  - The apriori property (downward closure property)
    - Any subsets of a frequent itemset are also frequent itemsets

If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**

i.e., every transaction having

{beer, diaper, nuts}

also contains {beer, diaper}



Anti-monotone property of support

- All subsets of a frequent itemset must also be frequent.
- {Bread, Egg}.count>= {Bread, Egg, Vegetables}.*count*
- If sup({Bread, Egg, Vegetables}) (30/100) = 0.3 > minsup, then {Bread, Egg} >= 0.3
- This is called the **anti-monotone property of support** where if we drop out an item from an itemset, support value of new itemset generated will either be the same or will increase.

The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on
  - In each iteration *k*, only consider itemsets that contain some *k*-1 frequent itemset

- Find frequent itemsets of size 1: F_1
- For $k = 2$
  - C_k = candidates of size *k*: those itemsets of size *k* that could be frequent, given F_{k-1}
  - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once)

The Apriori Algorithm—An Example

Database T

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

$Sup_{min} = 2$

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database T

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

C_1

$\xrightarrow{1\text{st scan}}$

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

$\xrightarrow{\text{1st scan}}$ C_1

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

$\xrightarrow{}$ L_1

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

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}

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

1st scan →

C_1

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

→ 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

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

1st scan $\longrightarrow$

C_1

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

$\longrightarrow$ 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 $\longleftarrow$

C_2

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

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}

$\leftarrow$ 2nd scan

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

$\xrightarrow{\text{1st scan}}$ C_1

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

$\longrightarrow$ 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}

$\xleftarrow{\text{2nd scan}}$ C_2

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

$\longleftarrow$ L_2

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

C_3

Itemset
{B, C, E}

The Apriori Algorithm—An Example

Database TDB

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

$Sup_{min} = 2$

1st scan

C_1

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

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

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm

C_k: Candidate itemset of size k
F_k : frequent itemset of size k

```
Algorithm Apriori(T)
    $C_1 \leftarrow$ init-pass($T$);
    $F_1 \leftarrow \{f \mid f \in C1, f.\text{count}/n \geq minsup\}$;    // n: no. of transactions in T
    for ($k = 2$; $F_{k-1} \neq \varnothing$; $k$++) do
        $C_k \leftarrow$ candidate-gen($F_{k-1}$);
        for each transaction $t \in T$ do
            for each candidate $c \in C_k$ do
                if $c$ is contained in $t$ then
                    $c.count$++;
            end
        end
        $F_k \leftarrow \{c \in C_k \mid c.count/n \geq minsup\}$
    end
return $F \leftarrow \bigcup_k F_k$;
```

Apriori candidate generation

- Function takes F_{k-1} and returns a superset (called the candidates) of the set of all frequent *k*-itemsets
- It has two steps
  - *join* step: Generate all possible candidate itemsets C_k of length *k* **(In sorted order)**
  - *prune* step: Remove those candidates in C_k that cannot be frequent

Implementation of Apriori

- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4 = {*abcd*}

Assignment example

1. 2-Itemset={{A, C}, {B, C}, {B, E}, {C, E}}

- **3-itemset ?**

2. 2-Itemset={{I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}}}

- **3-itemset ?**

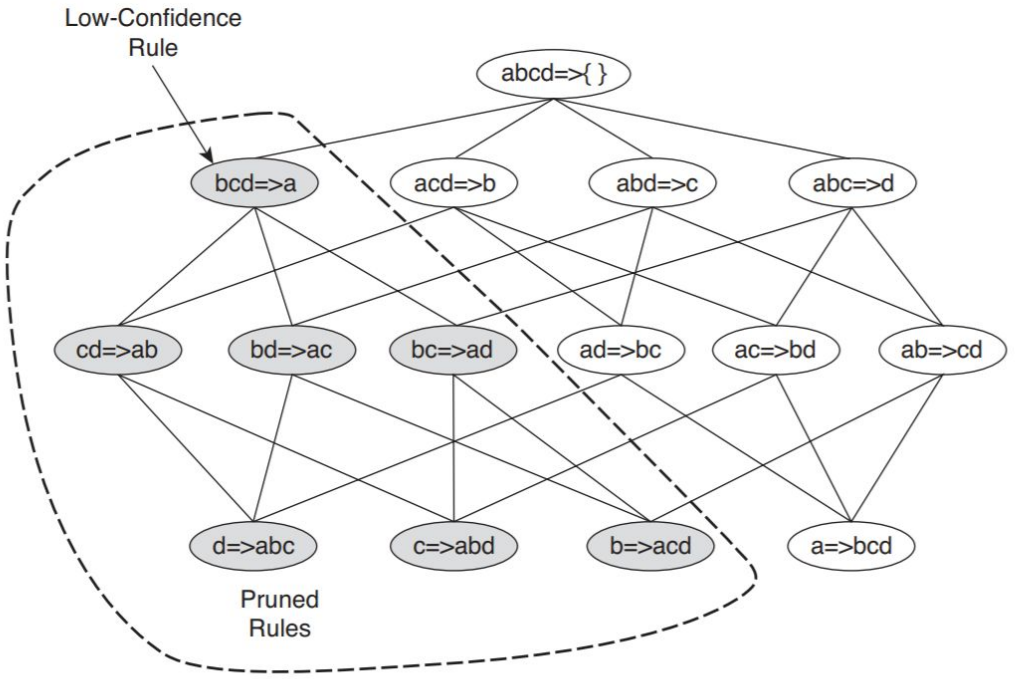
Candidate-gen function

```
Function candidate-gen($F_{k-1}$)
    $C_k \leftarrow \varnothing$;
    forall $f_1, f_2 \in F_{k-1}$
        with $f_1 = \{i_1, \ldots, i_{k-2}, i_{k-1}\}$
        and $f_2 = \{i_1, \ldots, i_{k-2}, i'_{k-1}\}$
        and $i_{k-1} < i'_{k-1}$ do
      $c \leftarrow \{i_1, \ldots, i_{k-1}, i'_{k-1}\}$;        // join $f_1$ and $f_2$
      $C_k \leftarrow C_k \cup \{c\}$;
      for each $(k-1)$-subset $s$ of $c$ do
        if ($s \notin F_{k-1}$) then
          delete $c$ from $C_k$;        // prune
      end
    end
    return $C_k$;
```

Comments on Confidence

- Note :
  - support of all the rules generated from same itemset remains the same
  - difference occurs only in the denominator calculation of confidence.
  - As number of items in X decrease, support{X} increases (as follows from the anti-monotone property of support) and hence the confidence value decreases.
- An intuitive explanation
  - F1 = {(butter),(egg, milk, bread)}
  - F2 = {(milk, butter, bread),(egg)}
  - Butter.count>> {(milk, butter, bread).count
- So conf(F1) < conf(F2)

Rule prunning



- start with a frequent itemset {a,b,c,d}
- start forming rules with just one consequent.
- Remove the rules failing to satisfy the minconf condition.
- start forming rules using a combination of consequents from the remaining ones.
- Keep repeating until only one item is left on antecedent. This process has to be done for all frequent itemsets.

Step 2: Generating rules from frequent itemsets

- Frequent itemsets ≠ association rules
- For each frequent itemset *X*,

  For each proper nonempty subset *A* of *X*,
  - Let *B* = X - *A*
  - A $\longrightarrow$ B is an association rule if
    - Confidence(A $\rightarrow$ B) $\geq$ minconf,

      support(A $\rightarrow$ B) = support(A $\cup$ B) = support(X)

      confidence(A $\rightarrow$ B) = support(A $\cup$ B) / support(A)

Generating Rules: an example

- Suppose {2,3,4} is frequent, with sup=50%
  - Proper nonempty subsets: {2,3}, {2,4}, {3,4}, {2}, {3}, {4}, with sup=50%, 50%, 75%, 75%, 75%, 75% respectively
  - These generate these association rules:
    - $2,3 \rightarrow 4$, confidence=100%
    - $2,4 \rightarrow 3$, confidence=100%
    - $3,4 \rightarrow 2$, confidence=67%
    - $2 \rightarrow 3,4$, confidence=67%
    - $3 \rightarrow 2,4$, confidence=67%
    - $4 \rightarrow 2,3$, confidence=67%
    - All rules have support = 50%

Generating Rules: summary

- To recap, in order to obtain A $\rightarrow$ B, we need to have support(A $\cup$ B) and support(A)
- All the required information for confidence computation has already been recorded in itemset generation
  - No need to see the data *T* any more
- This step is not as time-consuming as frequent itemsets generation

Transaction id	Items
t1	{1, 2, 4, 5}
t2	{2, 3, 5}
t3	{1, 2, 4, 5}
t4	{1, 2, 3, 5}
t5	{1, 2, 3, 4, 5}
t6	{2, 3, 4}

- By applying the algorithm with *minsup* = 0.5, *minconf*= 0.9 and *minlift* = 1

rule 0: 4 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0
rule 1: 3 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0
rule 2: 1 ==> 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2
rule 3: 1 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0
rule 4: 5 ==> 2 support : 0.833(5/6) confidence : 1.0 lift : 1.0
rule 5: 4 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0
rule 6: 1 4 ==> 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.2
rule 7: 4 5 ==> 1 support : 0.5 (3/6) confidence : 1.0 lift : 1.5
rule 8: 1 4 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0
rule 9: 3 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0
rule 10: 1 5 ==> 2 support : 0.66 (4/6) confidence : 1.0 lift : 1.0
rule 11: 1 2 ==> 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2
rule 12: 1 ==> 2 5 support : 0.66 (4/6) confidence : 1.0 lift : 1.2
rule 13: 1 4 5 ==> 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.0
rule 14: 1 2 4 ==> 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.2
rule 15: 2 4 5 ==> 1 support : 0.5 (3/6) confidence : 1.0 lift : 1.5
rule 16: 4 5 ==> 1 2 support : 0.5 (3/6) confidence : 1.0 lift : 1.5
rule 17: 1 4 ==> 2 5 support : 0.5 (3/6) confidence : 1.0 lift : 1.5

For More example

- https://www.digitalvidya.com/blog/apriori-algorithms-in-data-mining/ [How to prepare the input frequency table?]
- https://www.softwaretestinghelp.com/apriori-algorithm/
- https://www.geeksforgeeks.org/apriori-algorithm/

Assignment Exercise: 1

- A database has five transactions.

Let *min sup* = 60% and *min con f* = 80%.

TID	***items bought***
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y}
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

Find all frequent itemsets using Apriori.

Apriori Algorithm

Seems to be very expensive

- Breadth-first (Level-wise) search
- If, K = the size of the largest itemset then makes at most K passes over data
- Very simple and fast
  - Under some conditions, all rules can be found in linear time
- Scale up to large data sets

Apriori Algorithm

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
    - The number of frequent itemsets to be generated is senstive to the minsup threshold
    - When minsup is low, there exist potentially an exponential number of frequent itemsets
    - Example:
      - 10^4 frequent 1-itemsets, generate more than 10^7 candidate 2-itemsets
      - To discover a frequent pattern of size 100, such as $\{a_1, \ldots, a_{100}\}$
        - Generated candidates $2^{100} – 1$ = (Approx.) 10^{30}
  - Tedious workload of support counting for candidates

Improve Efficiency of Aptiori

- **Hash-Based Technique:**
  - hash-based structure (hash table) is used for managing the k-itemsets and its corresponding count.
- **Transaction Reduction:**
  - reduces the number of transactions scanning in iterations.
  - E.g. The transactions which do not contain frequent items are marked or removed.(Preprocessing)
- **Partitioning:**
  - It is proven in some cases that for any itemset to be potentially frequent in the database, it should be frequent in at least one of the partitions of the database.
  - only two database scans to mine the frequent itemsets.
  - (Imbalance partitioning is an issue)

Improve Efficiency of Aptiori

- **Sampling:**
  - picks a random sample S from Database D
  - searches for frequent itemset in S.
  - It may be possible to lose a global frequent itemset. This can be reduced by lowering the min_sup.
- **Dynamic Itemset Counting:**
  - add new candidate itemsets at any marked start point of the database during the scanning of the database.

Apriori Algorithm

- Reducing Complexity of Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Mining Frequent Patterns without Candidate Generation ???

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- The FPGrowth Approach given by J. Han, J. Pei, and Y. Yin, SIGMOD' 00
  - Depth-first search
  - Avoid explicit candidate generation

FPGrowth Approach

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - Highly condensed, but complete for frequent pattern mining
  - Avoid costly database scans
- An efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones called conditional databases
  - Avoid candidate generation: sub-database mining only!

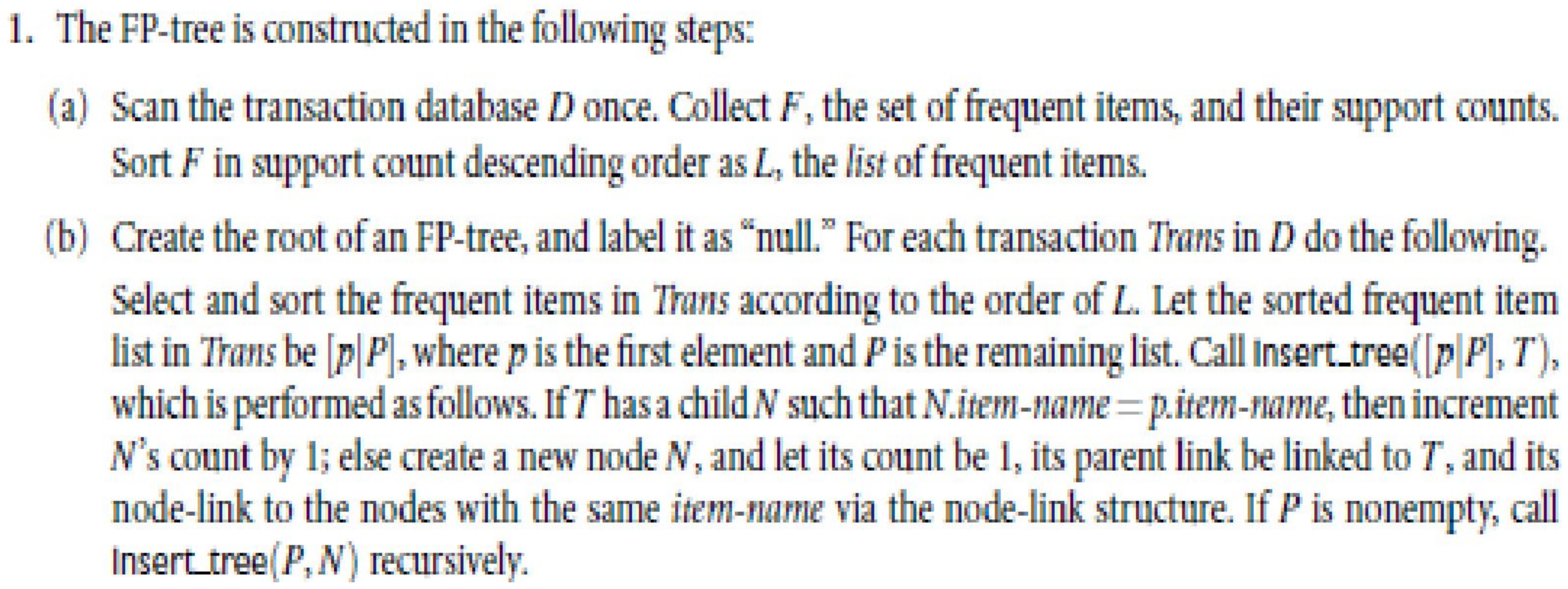
Example

- **Refer the Document**

The FP-Growth Mining Method

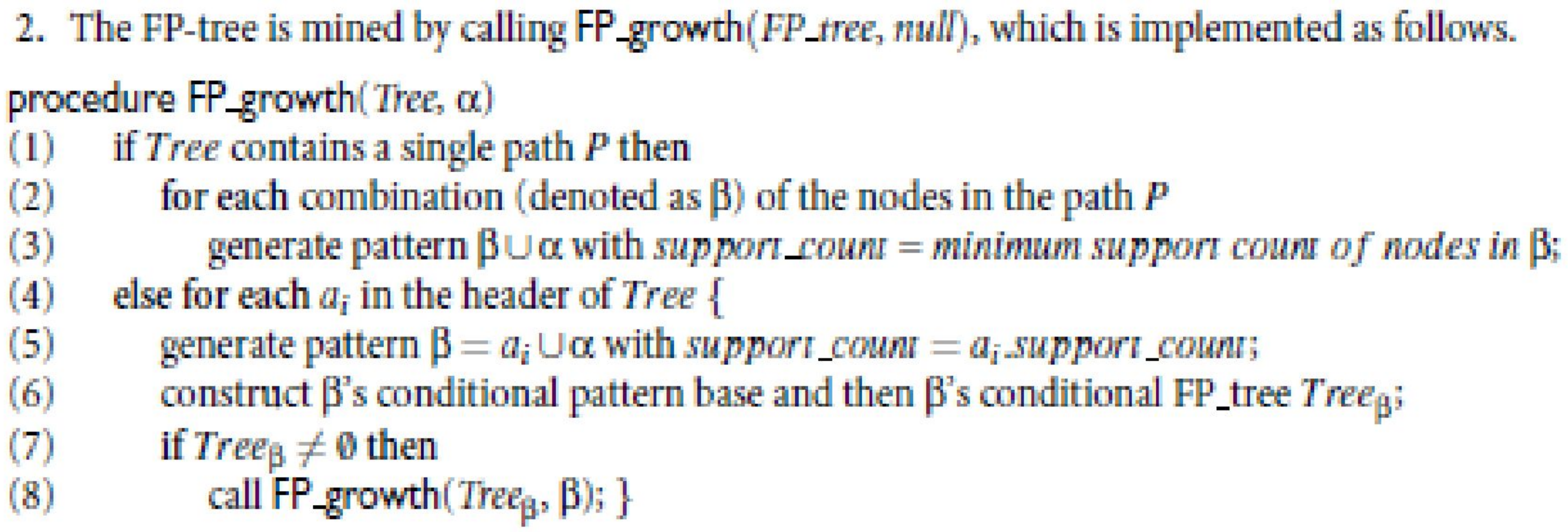
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

FP-Growth Algorithm



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.
   (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-Growth Algorithm Cont…



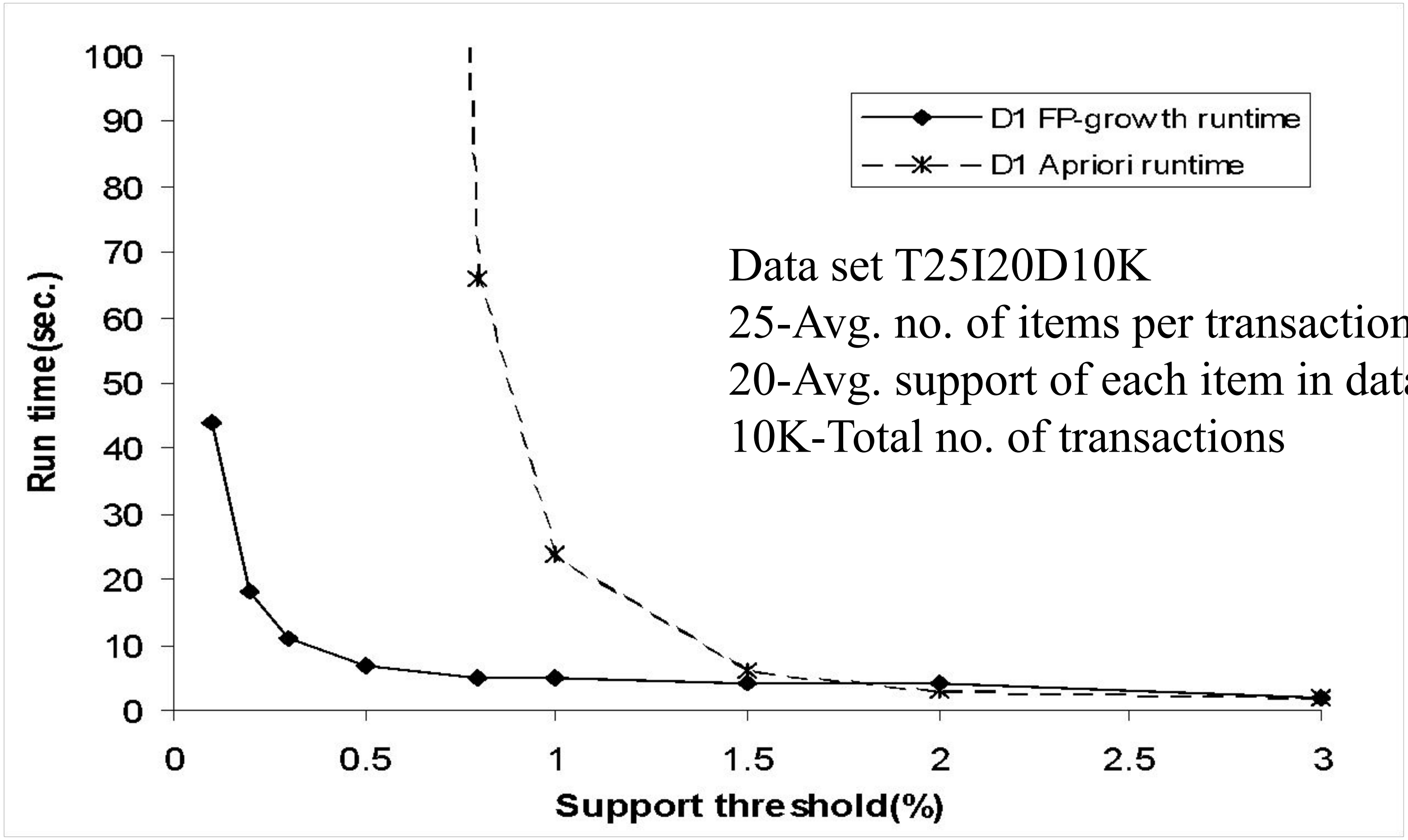
2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

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

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

FP-Growth vs. Apriori: Scalability With the Support Threshold



Data set T25I20D10K
25-Avg. no. of items per transaction
20-Avg. support of each item in dataset
10K-Total no. of transactions

Pros and Cons of FP-Growth

- Divide-and-conquer
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Performance is Faster than Apriori
  - Use compact data structure
  - No candidate generation, no candidate test
  - Eliminate repeated database scans
  - Basic operation is counting and FP-Tree building
- Problem:
  - When the database is large, sometimes unrealistic to construct a main memory based FP-Tree

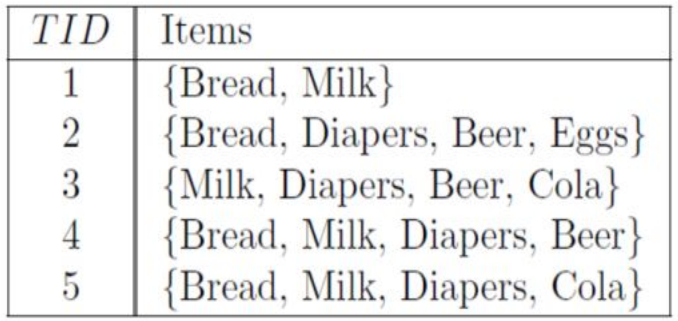
Data Format

- Apriori and FP-Growth
  - { TID: itemset }
    - TID: Transaction ID
    - Itemset: set of items bought in transaction TID
  - Horizontal Data Format
- Alternative way
  - { Item: TID_set }
    - Item: item name
    - TID_set: set of transaction identifiers containing the item
  - Vertical Data Format

- Before that ….

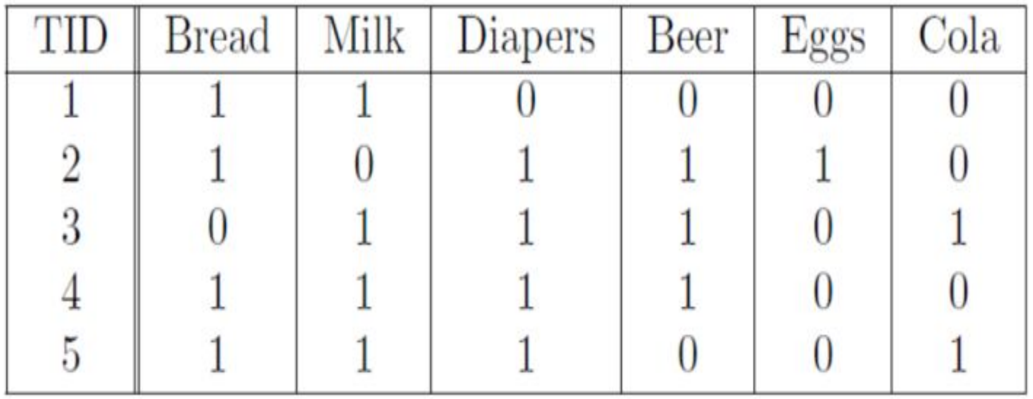
Transection representation : Binary

Table 6.1. An example of market basket transactions.



TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Table 6.2. A binary 0/1 representation of market basket data.

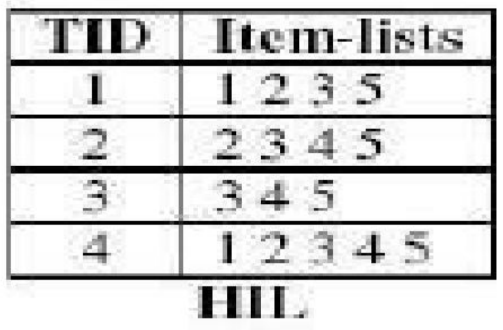


TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Data Layout for Transaction Matrix

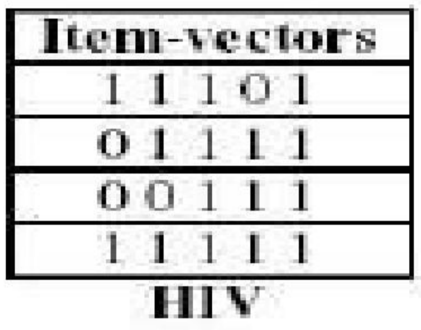
- Horizontal item-list (HIL): The database is represented as a set of transactions, storing each transaction as a list of item identifiers (item-list).
- Horizontal item-vector (HIV): The database is represented as a set of transactions, but each transaction is stored as a bit-vector (item-vector) of 1's and 0's to express the presence or absence of the items in the transaction.
- Vertical tid-list (VTL): The database is organized as a set of columns with each column storing an ordered list (tid-list) of only the transaction identifiers (TID) of the transactions in wich the item exists.
- Vertical tid-vector (VTV): This is similar to VTL, except that each column is stored as a bit-vector (tid-vector) of 1's and 0's to express the presence or absence of the items in the transactions

Data Layout for Transaction Matrix



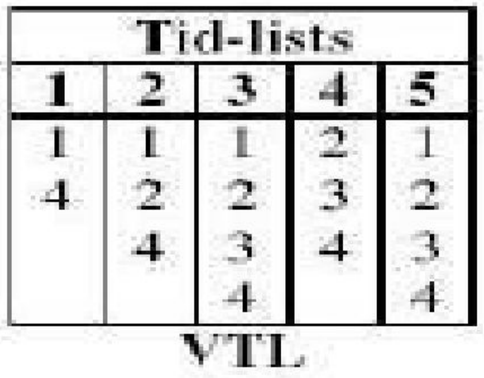
TID	Item-lists
1	1 2 3 5
2	2 3 4 5
3	3 4 5
4	1 2 3 4 5

HIL



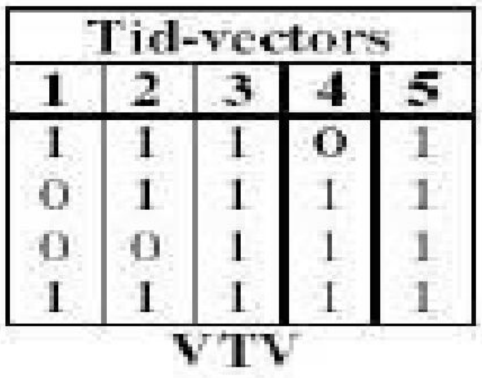
Item-vectors
1 1 1 0 1
0 1 1 1 1
0 0 1 1 1
1 1 1 1 1

HIV



Tid-lists				
1	2	3	4	5
1	1	1	2	1
4	2	2	3	2
	4	3	4	3
		4		4

VTL



Tid-vectors				
1	2	3	4	5
1	1	1	0	1
0	1	1	1	1
0	0	1	1	1
1	1	1	1	1

VTV

Example: Data Layout

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	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

$\downarrow$

TID-list

Mining by Exploring Vertical Data Format

- ECLAT (Equivalence CLASS Transformation)
- Developed by Zaki

ECLAT: Introduction

- efficient and scalable version of the Apriori algorithm.
- While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph.
- This vertical approach of the ECLAT algorithm makes it a faster algorithm than the Apriori algorithm.

ECLAT Algorithm

- Deriving frequent patterns based on vertical intersections
  - t(X) = t(Y): X and Y always happen together
  - t(X) ⊂ t(Y): transaction having X always has Y
- To count itemset AB
  - Intersect TID-list of itemA with TID-list of itemB

ECLAT Algorithm

- Transform the horizontally formatted data to the vertical format by scanning the data set once
- Support count of an itemset
  - The length of the TID_set of the itemset

ECLAT Algorithm

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.

A
1
4
5
6
7
8
9

$\wedge$

B
1
2
5
7
8
10

$\rightarrow$

AB
1
5
7
8

- 3 traversal approaches:
  - top-down, bottom-up and hybrid

ECLAT Algorithm

- Starting with k=1, the Frequent k-itemsets can be used to construct the candidate (k+1) itemsets based on the Apriori property
  - Done by intersection of the TID_sets of the frequent k-itemsets to compute the TID_sets of the corresponding (k+1) itemsets
- This process repeats, with k incremented by 1 each time, until no frequent itemsets or no candidate itemsets can be found

Eclat

- Transactions, originally stored in horizontal format,
- are read from disk and converted to vertical format.

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

$\rightarrow$

Bread
1
2
4
5

Milk
1
3
4
5

Diaper
2
3
4
5

Beer
2
3
4

Coke
3
5

Eggs
2

Eclat

The frequency of each item is counted and the infrequent items and their corresponding vertical lists are deleted from the vertical list.

Bread	Milk	Diaper
1	1	2
2	3	3
4	4	4
5	5	5

Beer	Coke	Eggs
2	3	2
3	5	
4		

Bread	Milk	Diaper
1	1	2
2	3	3
4	4	4
5	5	5

Beer
2
3
4

- Minimum support = 3

Eclat

- The Eclat algorithm is defined recursively.
- The initial call uses all the single items with their tidsets.
- In each recursive call, the function verifies each itemset-tidset pair with all the others pairs to generate new candidates.
- If the new candidate is frequent, it is added to the set. Then, recursively, it finds all the frequent itemsets in the branch.

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}	{Bread, Diaper}	{Bread, Beer}
1	2	2
4	4	4
5	5	

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}	{Bread, Diaper}	{Bread, Beer}
1	2	2
4	4	4
5	5	

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}	{Bread, Diaper}	{Bread, Beer}
1	2	2
4	4	4
5	5	

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Bread, Beer}
2
4

{Bread, Milk, Diaper}
4
5

Eclat

Bread	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Bread, Beer}
2
4

{Bread, Milk, Diaper}
4
5

Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Milk, Diaper}
3
4
5

{Milk, Beer}
3
4

Eclat

	Milk	Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

{Milk, Diaper}
3
4
5

{Milk, Beer}
3
4

Eclat

		Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

		Diaper	Beer
1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

1
2
4
5

1
3
4
5

Diaper
2
3
4
5

Beer
2
3
4

$\Downarrow$

{Diaper, Beer}
2
3
4

Eclat

1	1	2	2
2	3	3	3
4	4	4	4
5	5	5	

Eclat

Bread
1
2
4
5

Milk
1
3
4
5

Diaper
2
3
4
5

Beer
2
3
4

{Bread, Milk}
1
4
5

{Bread, Diaper}
2
4
5

{Milk, Diaper}
3
4
5

{Diaper, Beer}
2
3
4

Eclat

- Uses vertical database – tidset(bitset) intersections.
- Scans the database only once.
- Depth-first search algorithm.

ECLAT Algorithm Summary

- Intersection is more efficient
- Pipelined counting for frequent itemsets
- Advantage
  - Less number of database scan
  - Very fast support counting
  - No need to scan the database to find the support of (k+1) itemsets (for k>=1)
    - Because the TID_set of each k-itemset carries the complete information required for counting each support

ECLAT Algorithm

- Disadvantage
  - Intermediate tid-lists may become too large for memory
  - Long computation time for intersecting the long set

- Performance improvement Idea
  - Using diffset to accelerate mining [CHARM Algorithm]
    - Only keep track of differences of tids
    - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
    - Diffset $(XY, X) = \{T_2\}$

Closed Patterns and Max-Patterns

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

$\binom{100}{1}$ =100 frequent 1-itemsets

$\binom{100}{2}$ 2-frequent item set

$\binom{100}{100}$ 100-frequent itemset

$$\binom{100}{1} + \binom{100}{2} + \ldots \binom{100}{100} =$$

$2^{100} – 1 = 1.27*10^{30}$ sub-patterns!

- Solution: *Mine closed patterns and max-patterns instead*

Max-Patterns

- An itemset X is a max-pattern (maximal) if X is frequent and there exists no frequent super-pattern Y ɔ X

Maximal Frequent Itemset

Definition

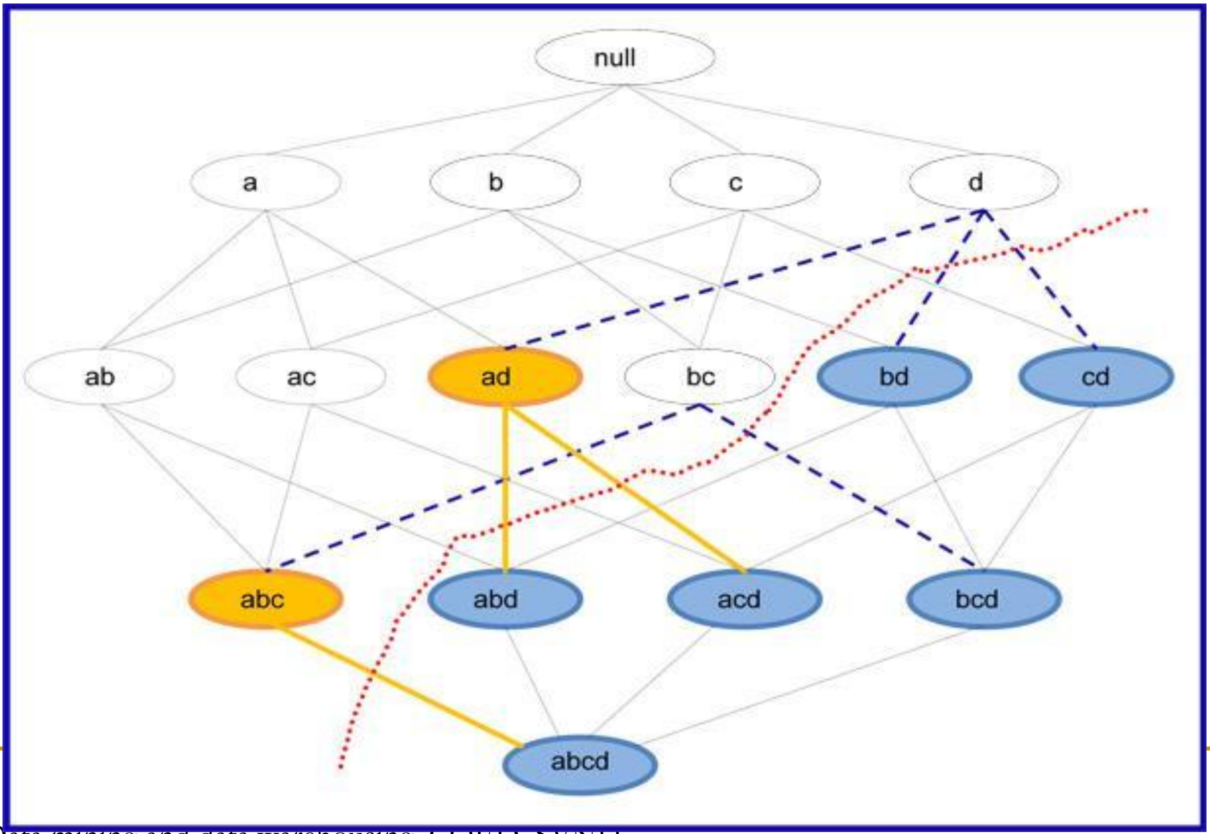
- It is a frequent itemset for which none of its immediate supersets are frequent.

Identification

- Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets.
- Identify all of its immediate supersets.
- If none of the immediate supersets are frequent, the itemset is maximal frequent.

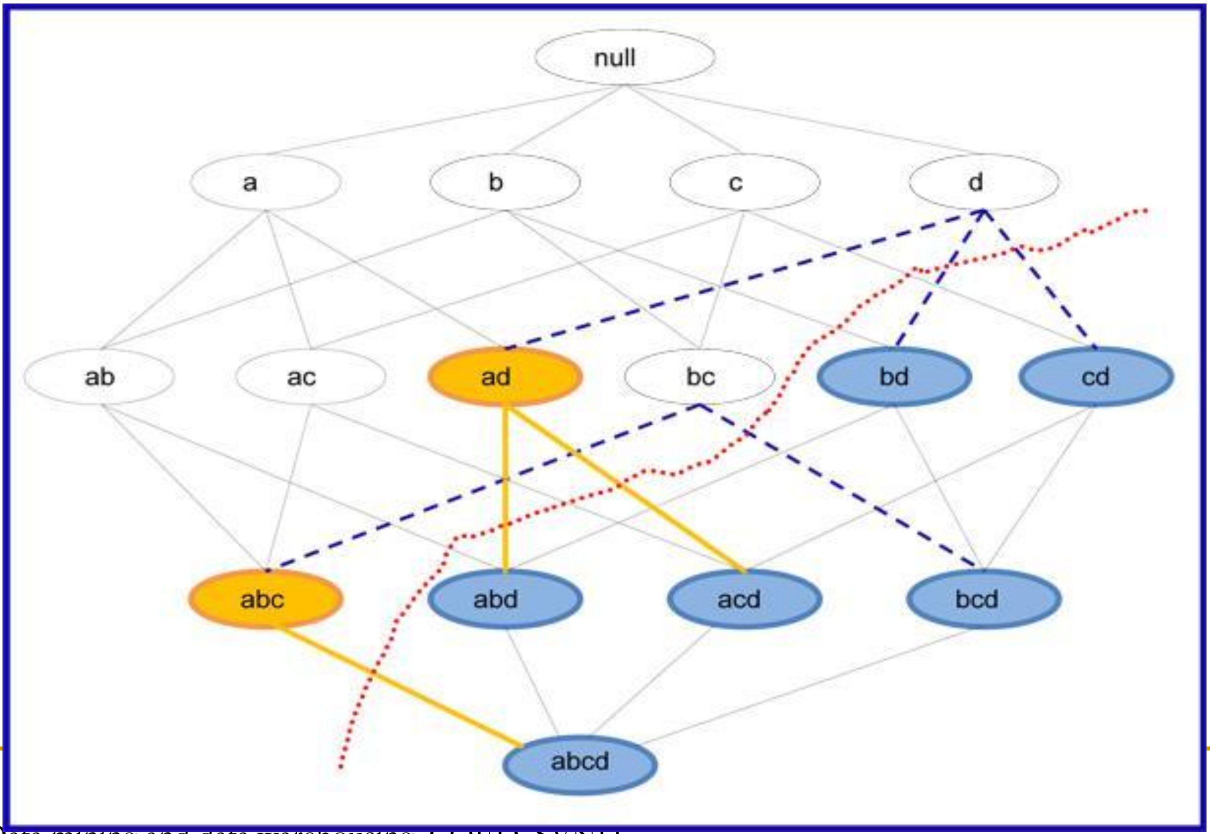
Example

- first identify the frequent itemsets at the border
  - ***d*, *bc*, *ad* and *abc*.**



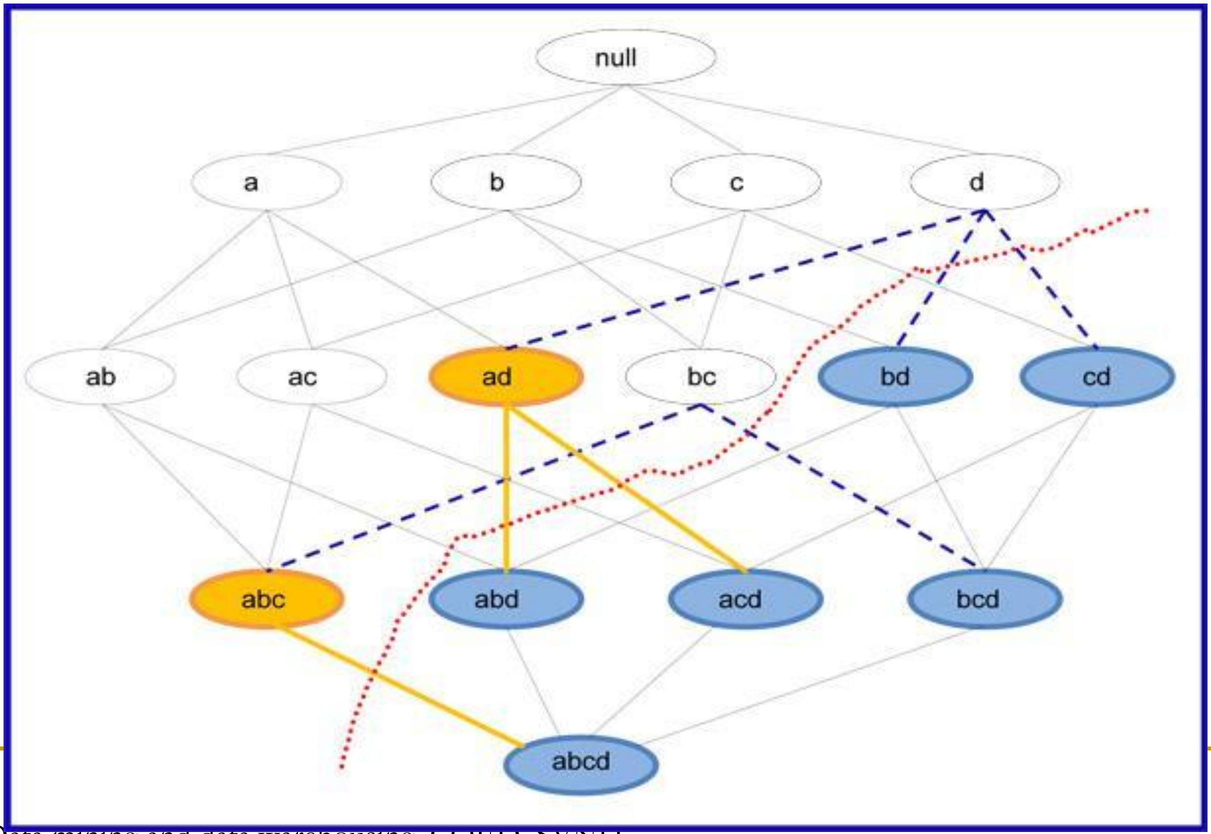
Example

- lattice is divided into two groups
  - items above the red line (Demarcation) that are blank are frequent itemsets
  - and the blue ones below the red dashed line are infrequent.



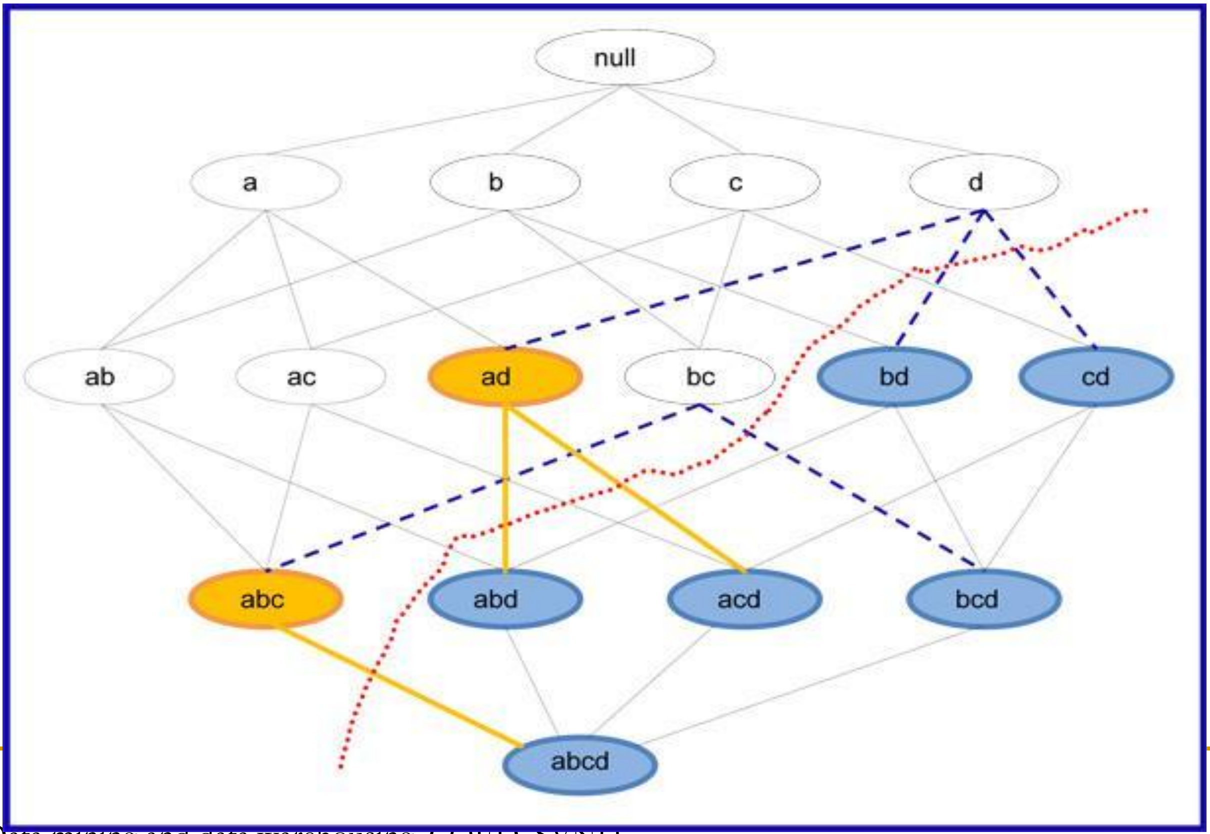
Example

- lattice is divided into two groups
  - items above the red line (Demarcation) that are blank are frequent itemsets
  - and the blue ones below the red dashed line are infrequent.



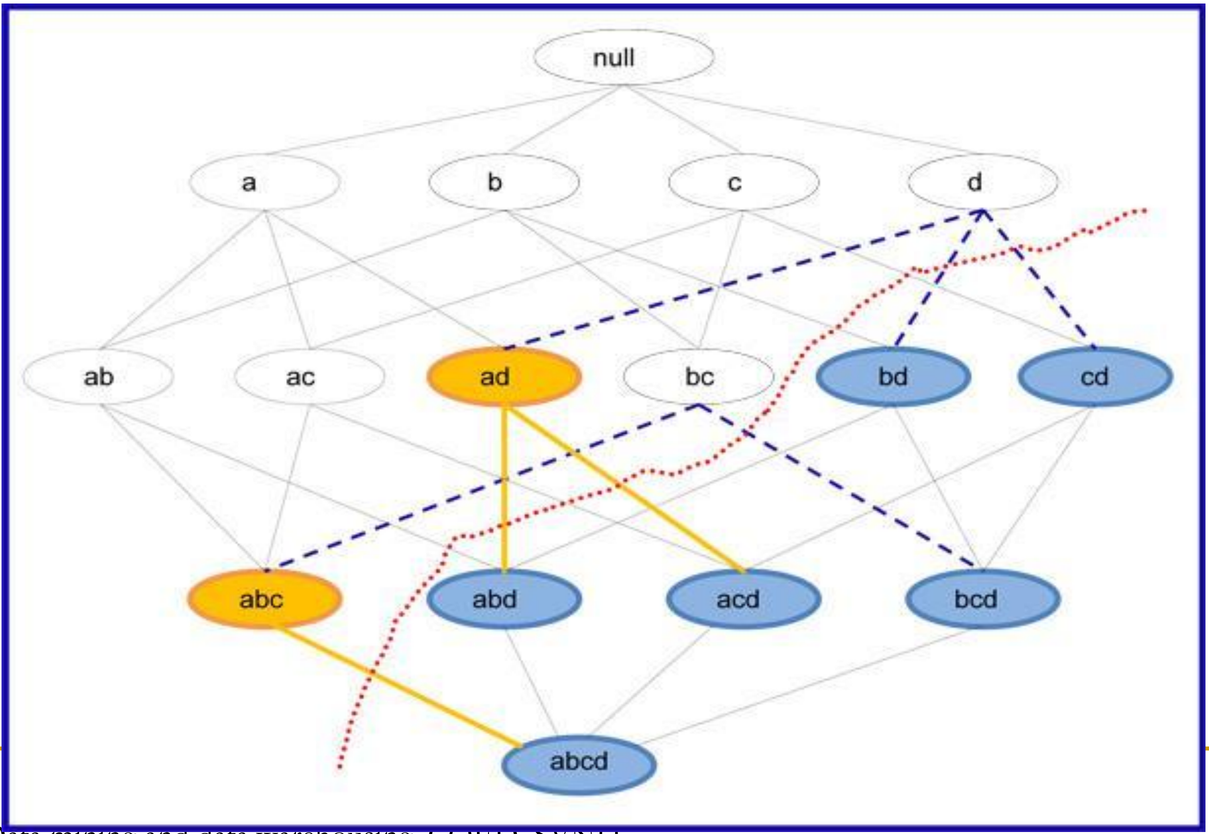
Example

- lattice is divided into two groups
  - items above the red line (Demarcation) that are blank are frequent itemsets
  - and the blue ones below the red dashed line are infrequent.



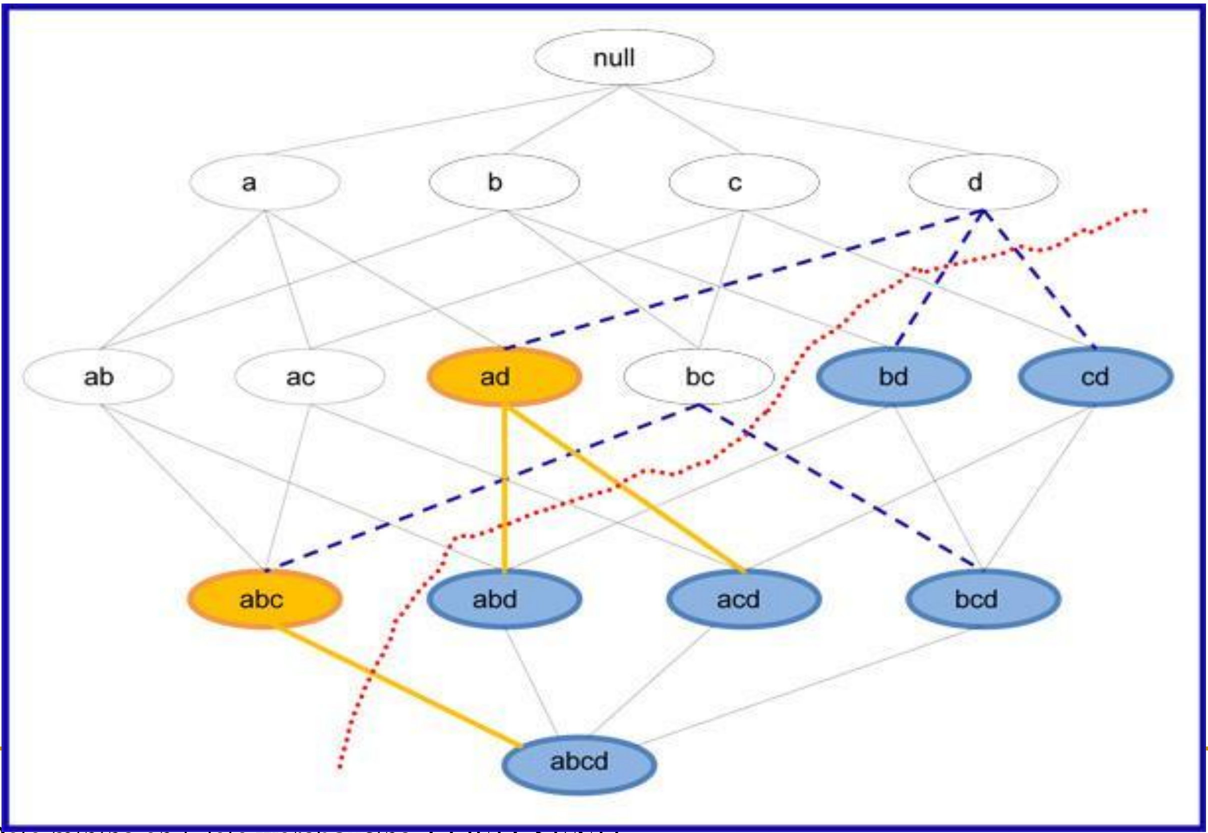
Example

- Identify their immediate supersets,
  - blue dashed line



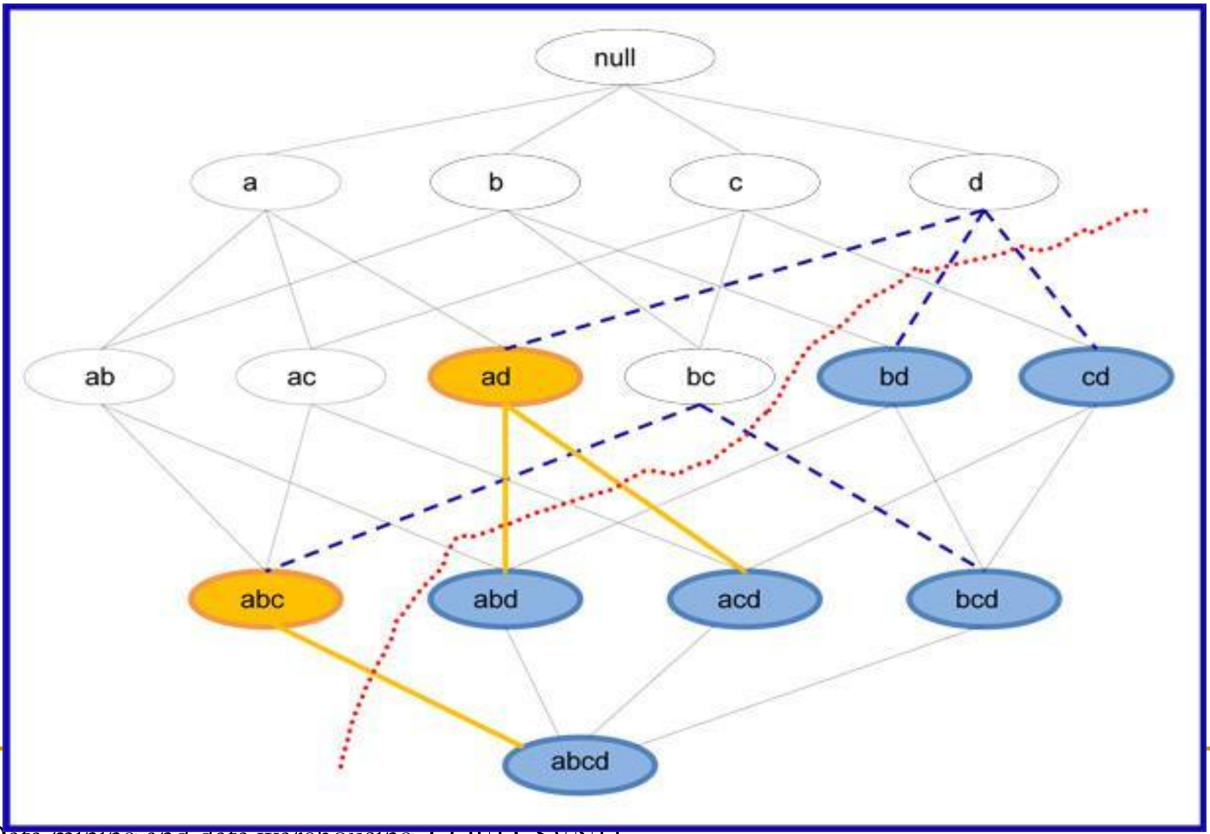
Example

- for d :
  - bd and cd are nonfrequent
  - ad is frequent
  - d is not maximal frequent,



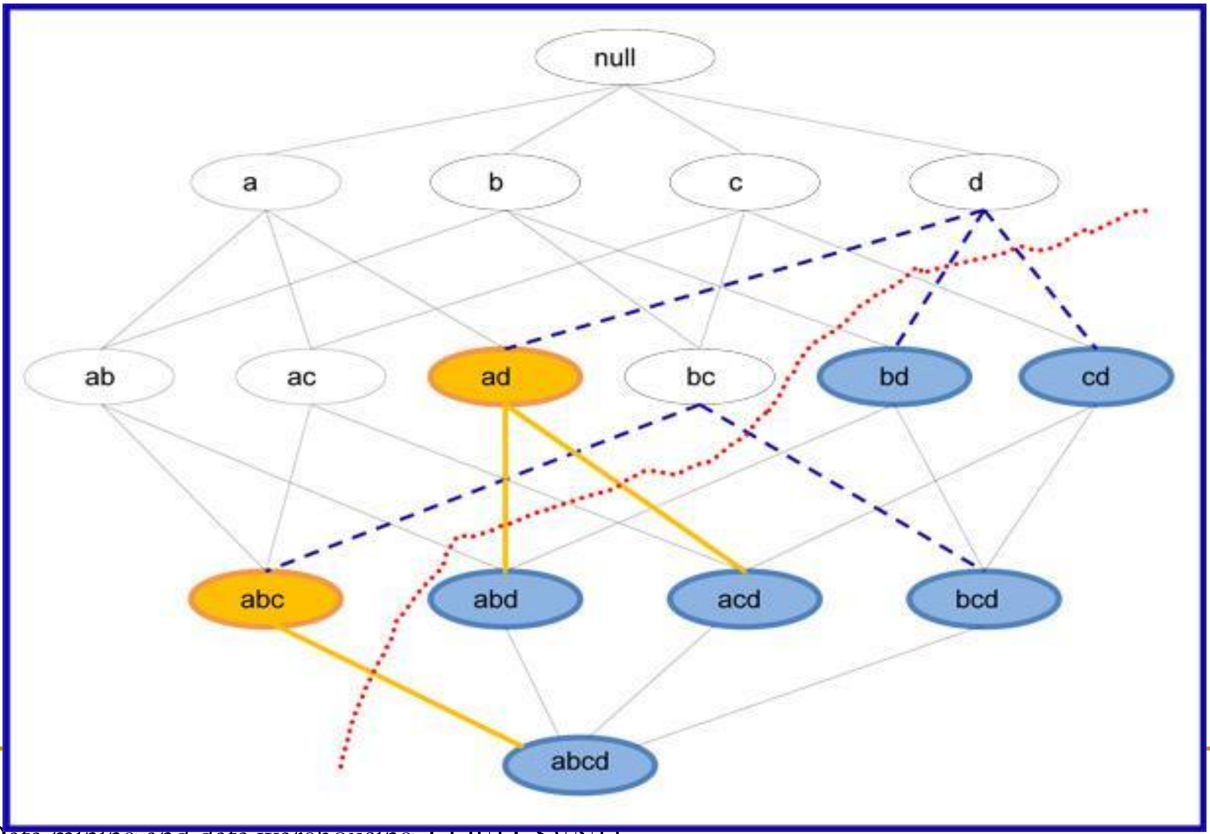
Example

- for *bc* :
  - *abc* is frequent
  - *bcd in non frequent*
  - *bc* is NOT maximal frequent.



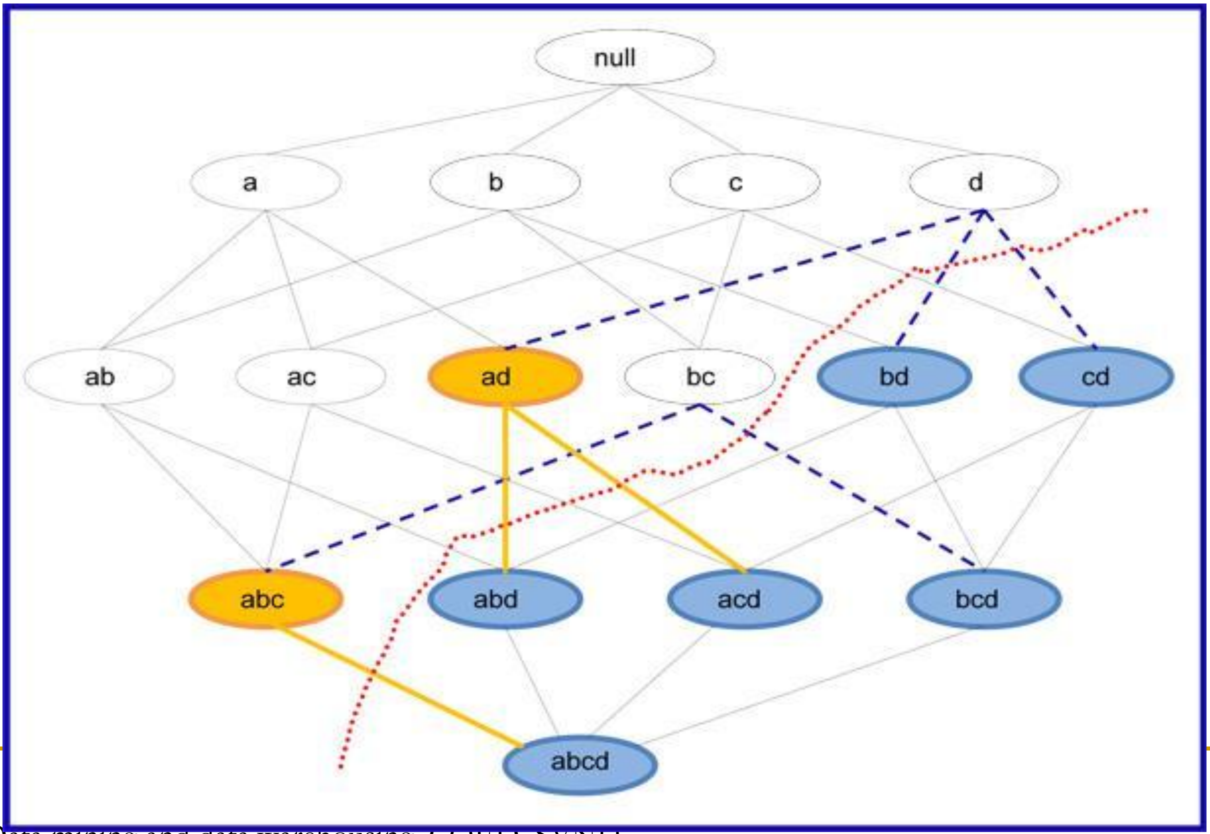
Example

- for ad
  - abd and acd are infrequent
  - ad maximal



Example

- for abc
  - *abcd* is infrequent
  - Abc maximal



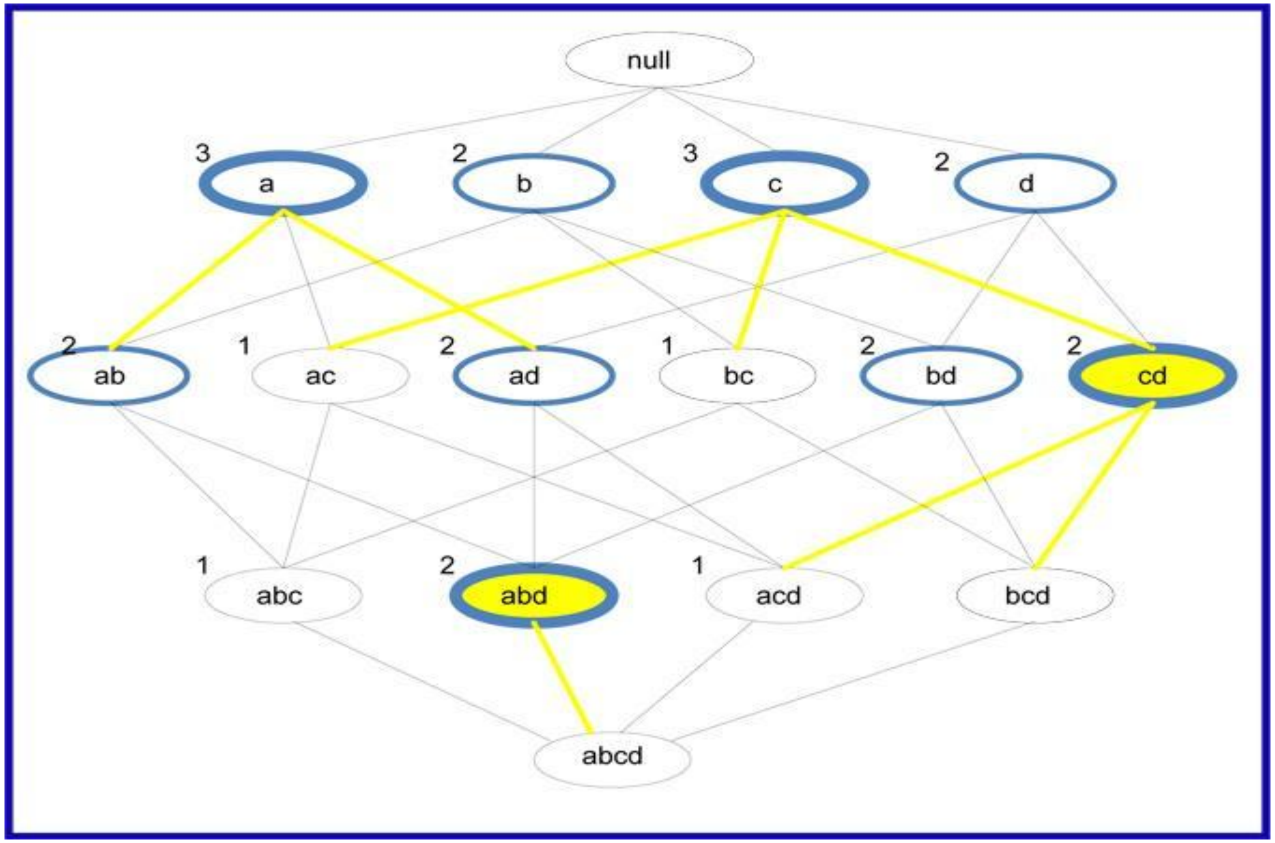
Closed Patterns

- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X
- It is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

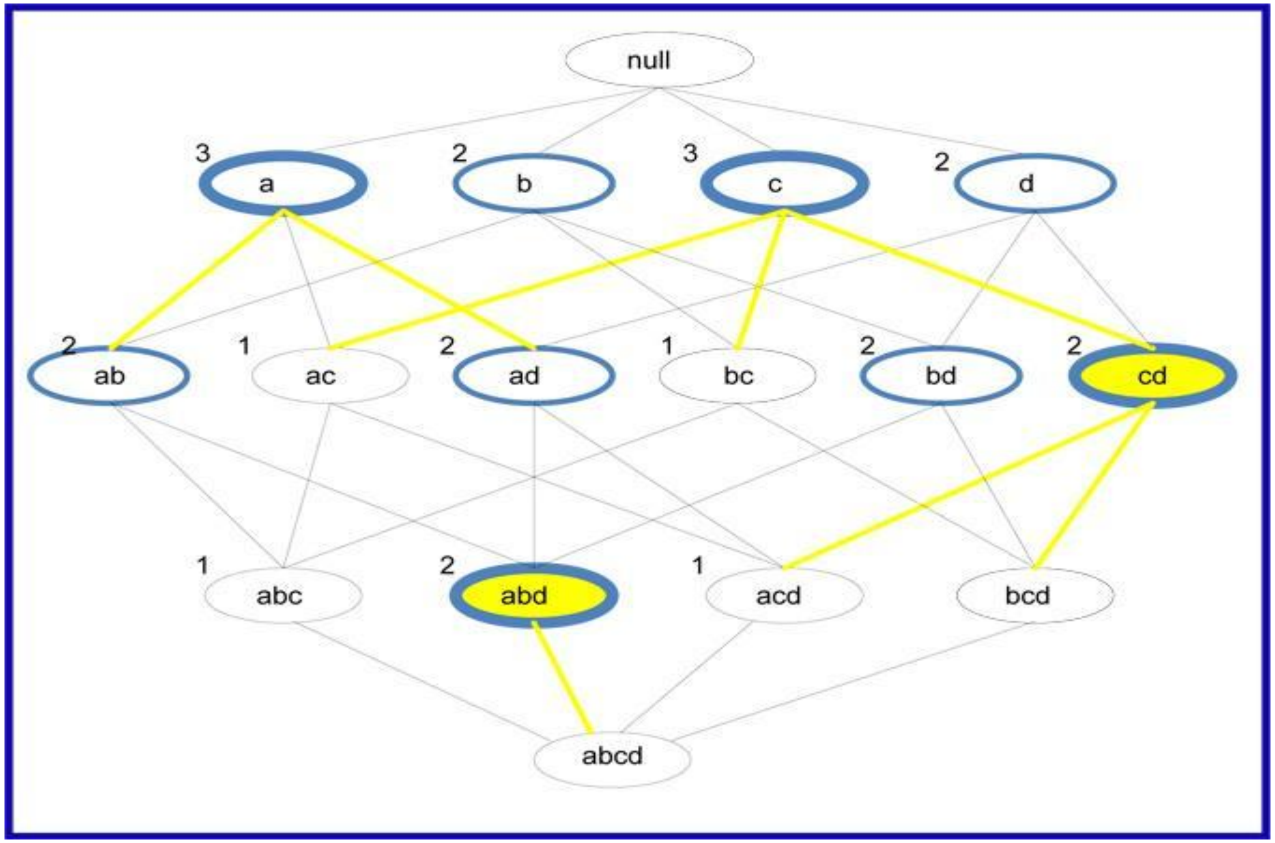
Closed Frequent Itemset

- ***Defination***
  - An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset. (all superset must be with less than the item support)
  - It is a frequent itemset that is both closed and its support is greater than or equal to minsup.
- ***Identification***
  - Method1: Identify all frequent itemsets then check for superset with the same support. If found then disqualify otherwise item is closed
  - Method2: first identify the closed itemsets and then use the minsup to determine which ones are frequent.

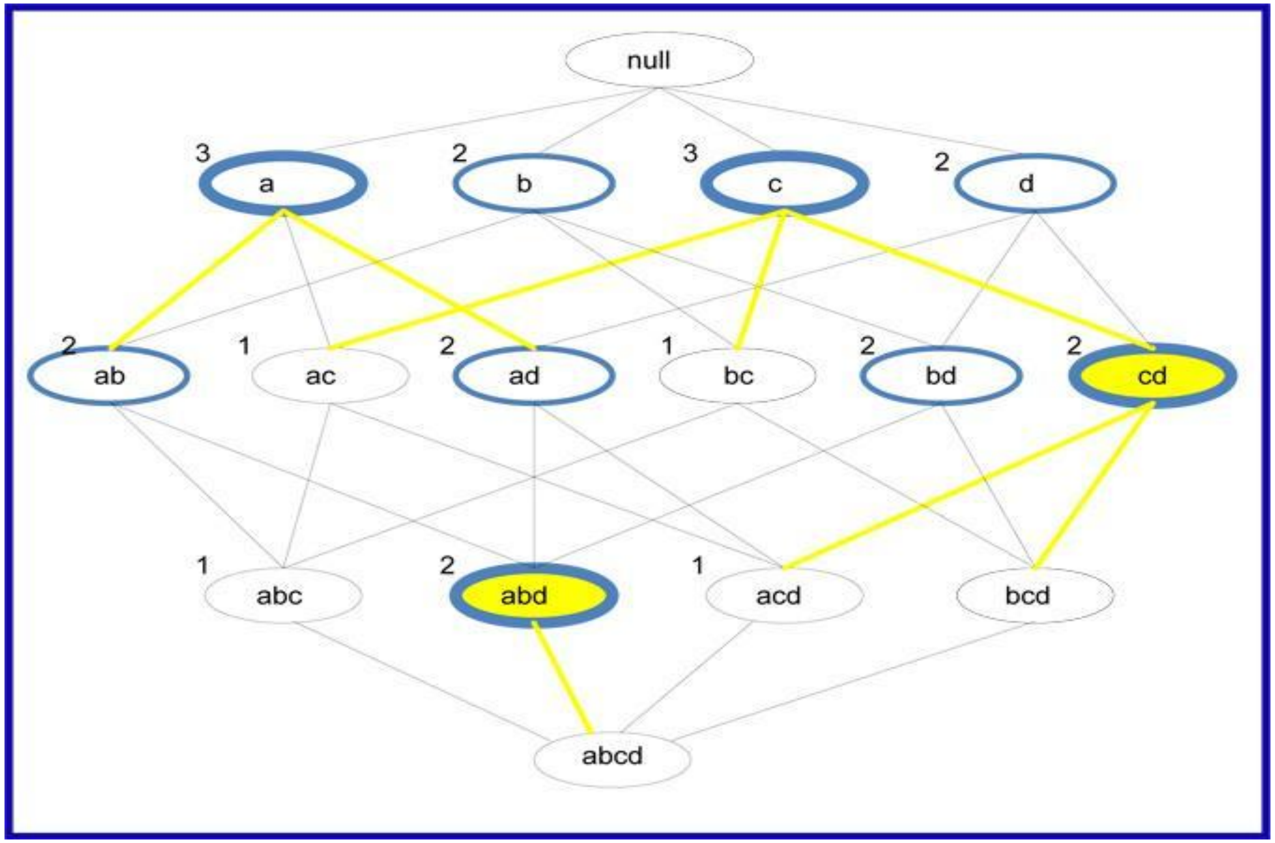
- The itemsets that are circled with blue are the frequent itemsets.
- The itemsets that are circled with the thick blue are the closed frequent itemsets.
- The itemsets that are circled with the thick blue and have the yellow fill are the maximal frequent itemsets.



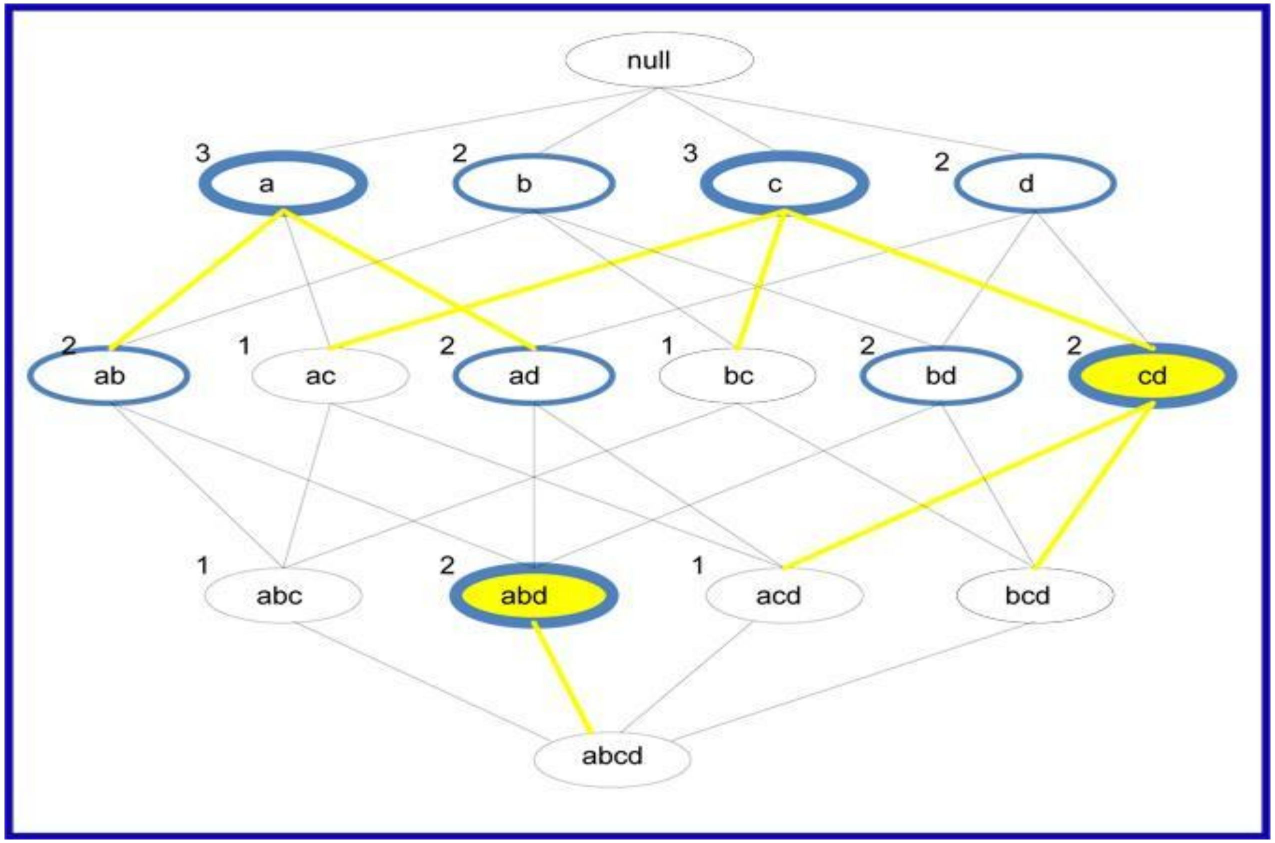
- For **ad**
  - frequent itemset
  - support of ad = support of **abd**
  - so it is NOT a closed frequent itemset



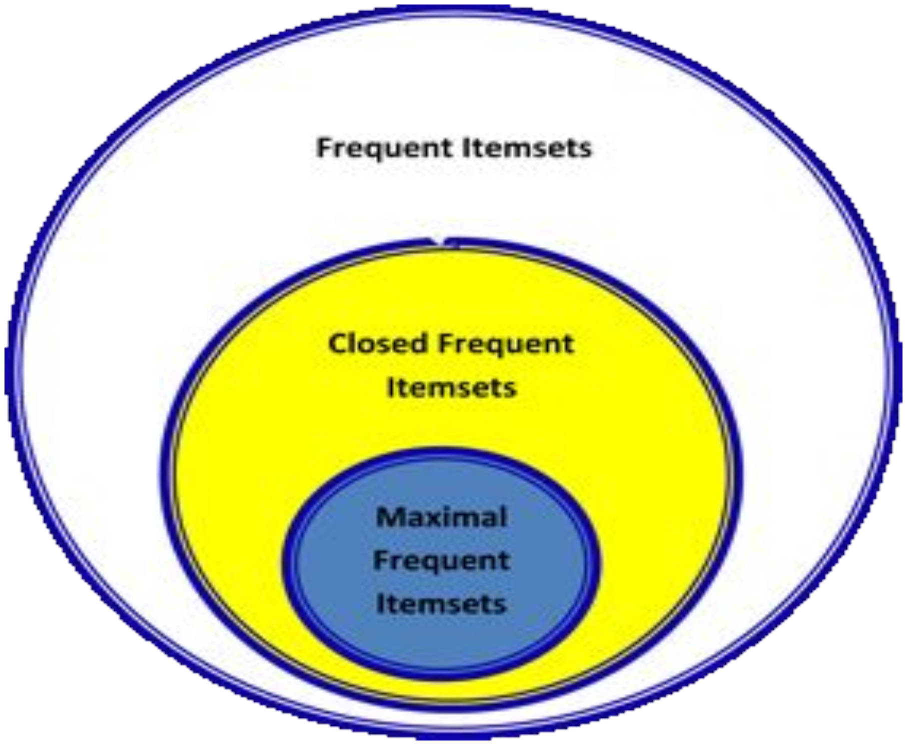
- For **ad**
  - c is frequent itemset
  - support of C >
    - Support of ac, bc and cd



- 9 frequent itemsets,
- 4 of them are closed frequent itemsets
- out of these 4, 2 of them are maximal frequent itemsets.



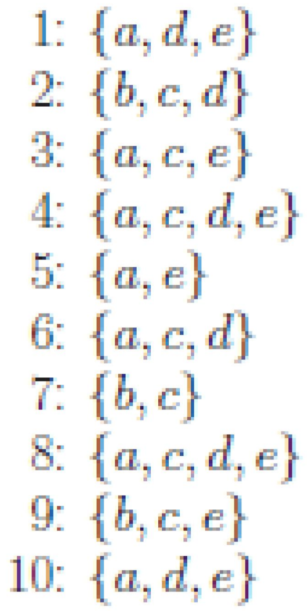
Relationship between three measures



- Closed frequent itemsets are more widely used than maximal frequent itemset
  - Find closed
  - Then find closed frequent

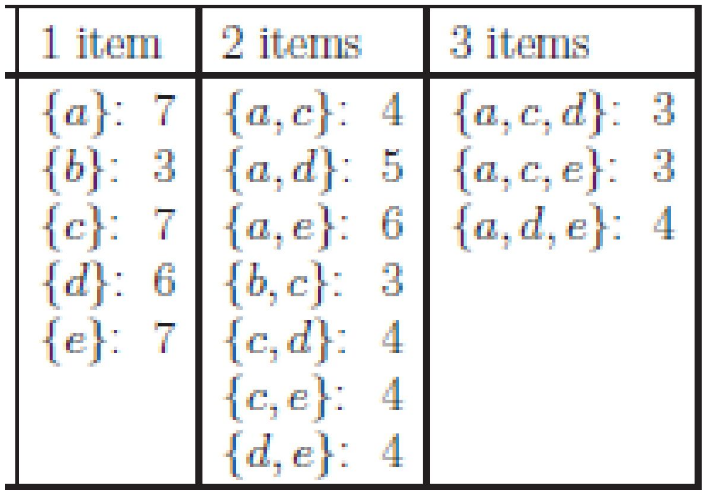
Max-Patterns - Example

Transaction Database



1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{b, c, e\}$
10:	$\{a, d, e\}$

Frequent Item Set



1 item	2 items	3 items
$\{a\}$: 7	$\{a, c\}$: 4	$\{a, c, d\}$: 3
$\{b\}$: 3	$\{a, d\}$: 5	$\{a, c, e\}$: 3
$\{c\}$: 7	$\{a, e\}$: 6	$\{a, d, e\}$: 4
$\{d\}$: 6	$\{b, c\}$: 3	
$\{e\}$: 7	$\{c, d\}$: 4	
	$\{c, e\}$: 4	
	$\{d, e\}$: 4	

MinSup = 3

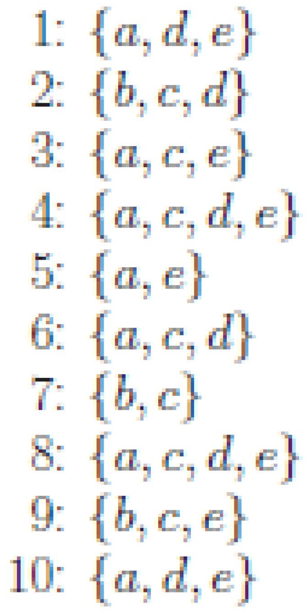
Find Itemset with none of supeset are nonfrequent (sup>3)

The maximal item sets are {b,c} {a,c,d} {a,c,e} {a,d,e}

- Every frequent itemset is a subset of at least one of these sets

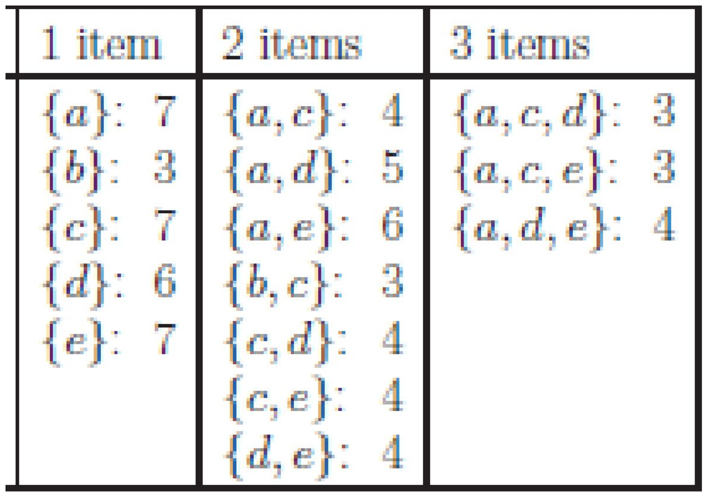
Closed Patterns - Example

Transaction Database



1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{b, c, e\}$
10:	$\{a, d, e\}$

Frequent Item Set



1 item	2 items	3 items
$\{a\}$: 7	$\{a, c\}$: 4	$\{a, c, d\}$: 3
$\{b\}$: 3	$\{a, d\}$: 5	$\{a, c, e\}$: 3
$\{c\}$: 7	$\{a, e\}$: 6	$\{a, d, e\}$: 4
$\{d\}$: 6	$\{b, c\}$: 3	
$\{e\}$: 7	$\{c, d\}$: 4	
	$\{c, e\}$: 4	
	$\{d, e\}$: 4	

- {b} is a subset of {b,c} both have a support of 3
- {c} is subset of {b,c} but support (c) > suppot({b,c}
- {d,e} is a subset of {a,d,e} both have a support of 4

All frequent item sets are Closed **except {b} and {d, e}**

Closed Patterns and Max-Patterns

- DB = $\{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$
  - Min_sup = 1.

Tid	Items
T1	A1,A2…A100
T2	A1,a2,…A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all Supp(superset) < sup(item)

Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.

Tid	Items
T1	A1,A2…A100
T2	A1,a2,…A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all Supp(superset) < sup(item)
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
-

Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.

Tid	Items
T1	A1,A2…A100
T2	A1,a2,…A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all Supp(superset) < sup(item)
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?

Closed Patterns and Max-Patterns

- DB = {$<a_1, \ldots, a_{100}>$, $< a_1, \ldots, a_{50}>$}
  - Min_sup = 1.

Tid	Items
T1	A1,A2…A100
T2	A1,a2,…A50

Item	Support
A1..A50	2
A51..A100	1

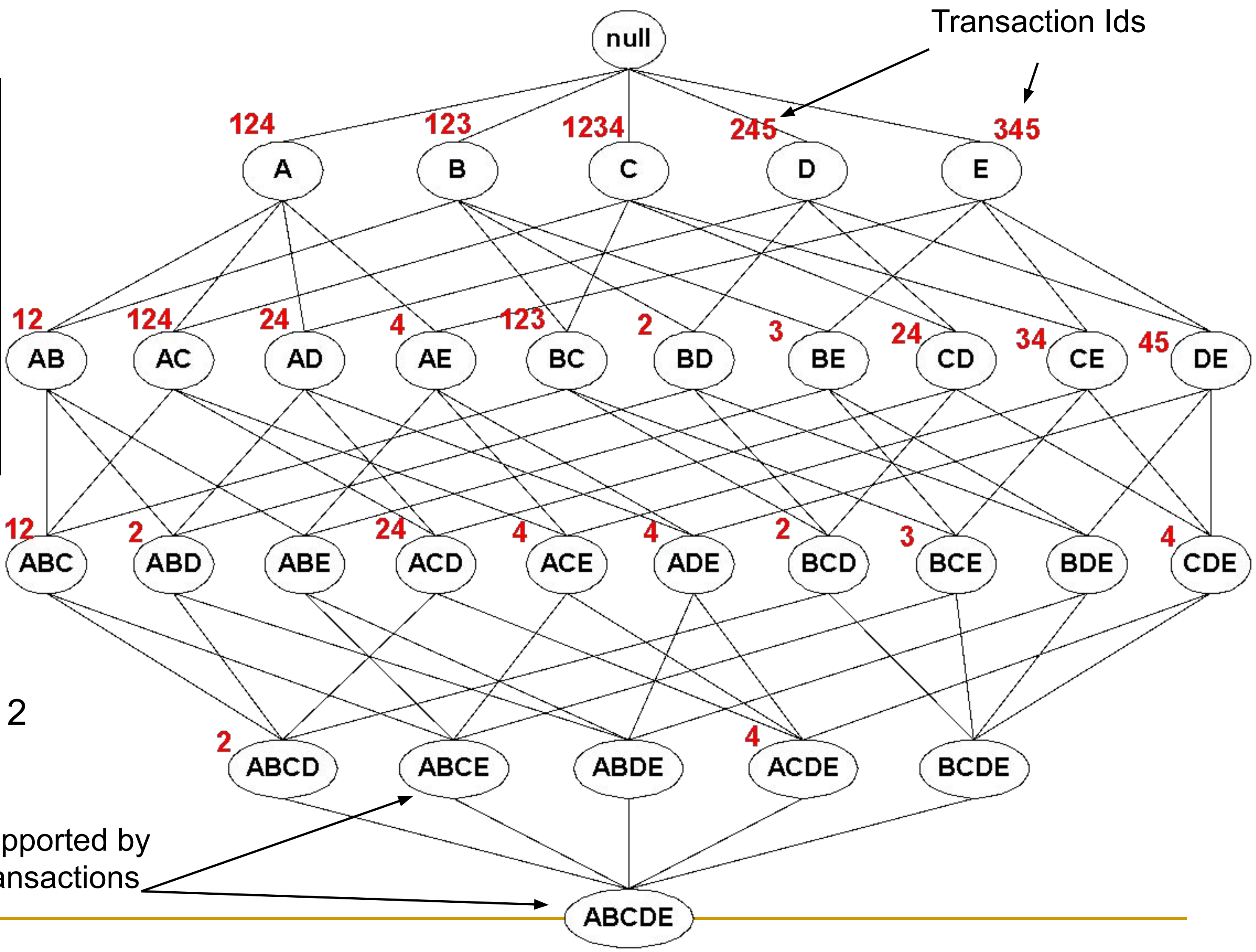
- What is the set of closed itemset?
  - all Supp(superset) < sup(item)
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of max-pattern?
  - $<a_1, \ldots, a_{100}>$: 1

Mine the Closed and Max-Patterns

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

Minimum support = 2

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

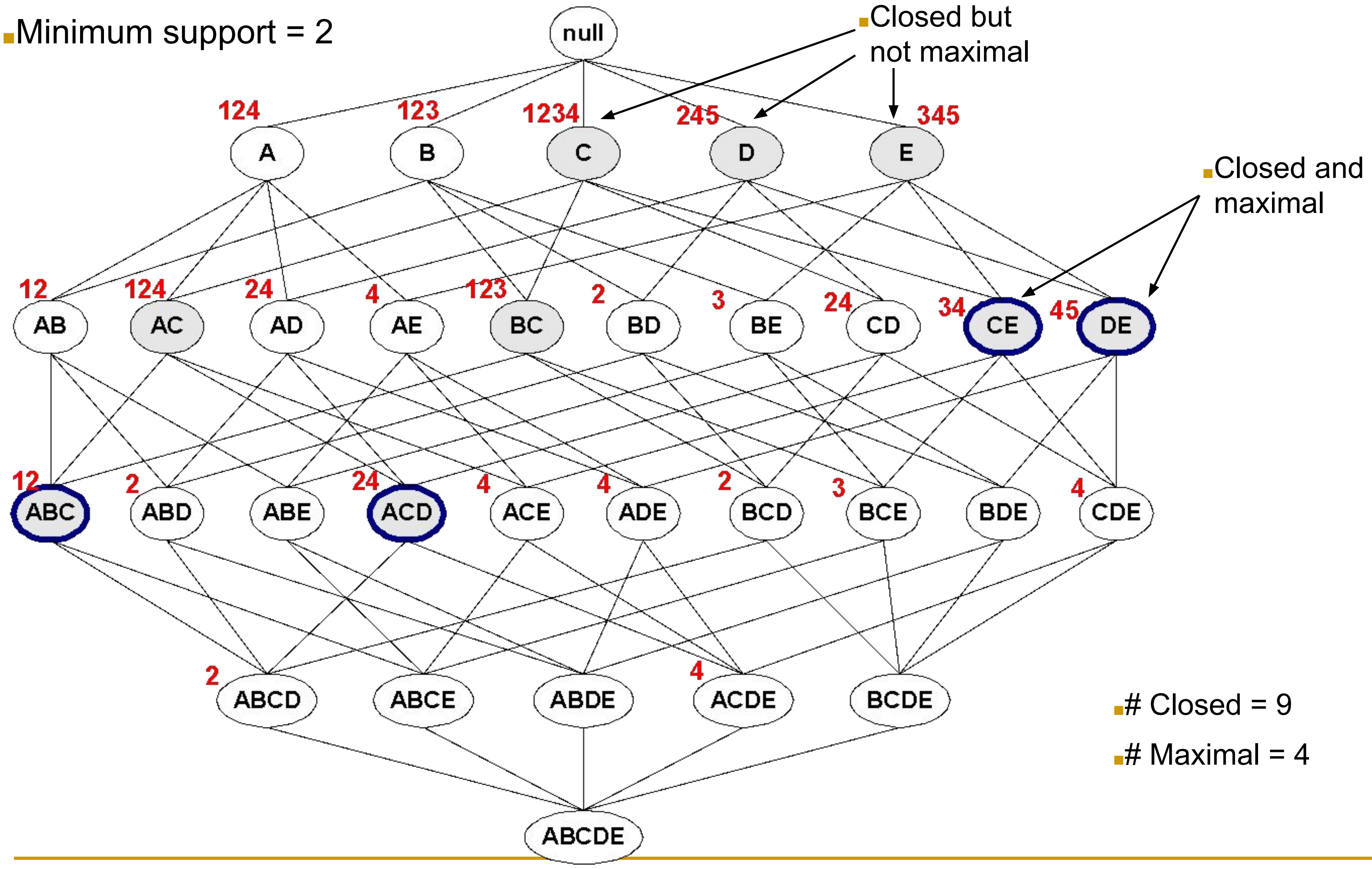


Minimum support = 2

- Not supported by any transactions

Maximal vs Closed Frequent Itemsets

- Minimum support = 2



- # Closed = 9
- # Maximal = 4

Closed Patterns and Max-Patterns

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

  $\binom{100}{1}$ =100 frequent 1-itemsets

  $\binom{100}{2}$ 2-frequent item set

  $\binom{100}{100}$ 100-frequent itemset

  $\binom{100}{1} + \binom{100}{2} + \ldots \binom{100}{100} =$

  $2^{100} - 1 = 1.27 \cdot 10^{30}$ sub-patterns!

- Solution: *Mine closed patterns and max-patterns instead*

Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.

Tid	Items
T1	A1,A2…A100
T2	A1,a2,…A50

Item	Support
A1..A50	2
A51..A100	1

- What is the set of closed itemset?
  - all Supp(superset) < sup(item)
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1

How to Mine Closed Frequent Itemsets

- From the set of closed frequent itemsets, we can easily derive the set of frequent itemsets and their support.
- In practice, it is more desirable to mine the set of closed frequent itemsets rather than the set of all frequent itemsets in most cases.

How to Mine Closed Frequent Itemsets

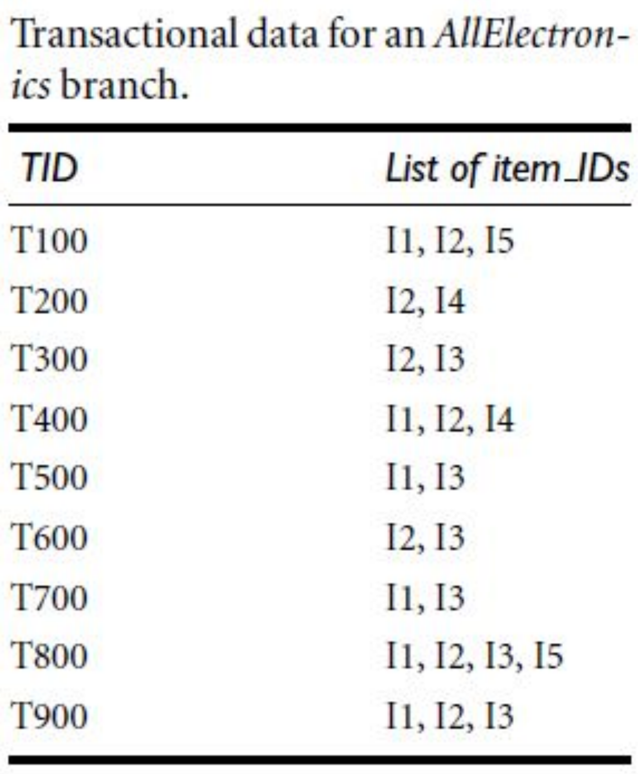
- Example and Exercise - a naïve approachFirst mine the complete set of frequent itemsets and then remove every frequent itemset that is a proper subset of, and carries the same support as, an existing frequent itemset
- It is quite costly
- Search for closed frequent itemsets directly during the mining process

How to Mine Closed Frequent Itemsets

- Requires to prune the search space as soon as we can identify the case of closed itemsets during mining
  - Item merging: *If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y, then X U Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y.*

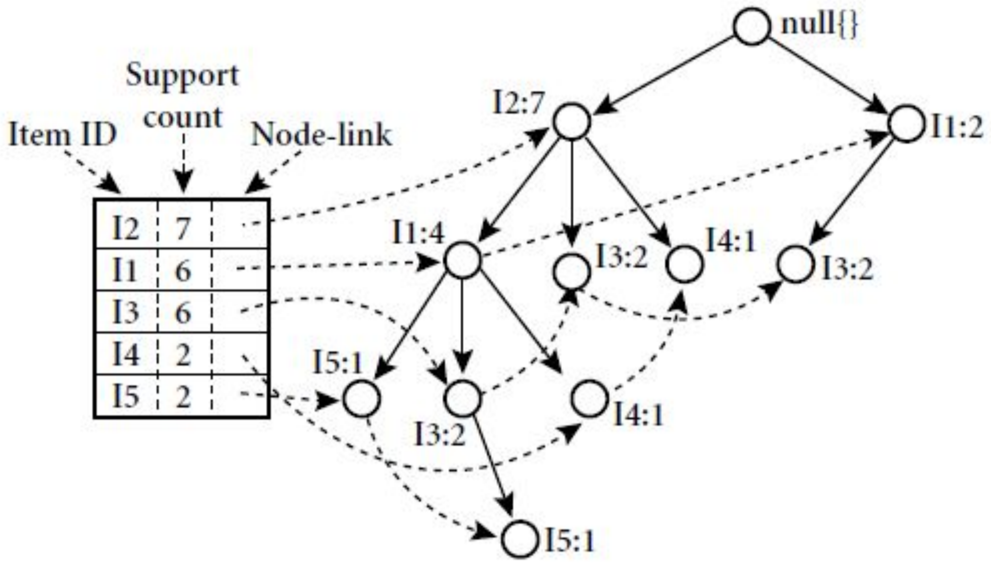
- Pruning
  - Item Merging
    - if Y appears in every occurrence of X, then Y is merged with X
  - Sub-itemset Pruning
  - Item Skipping

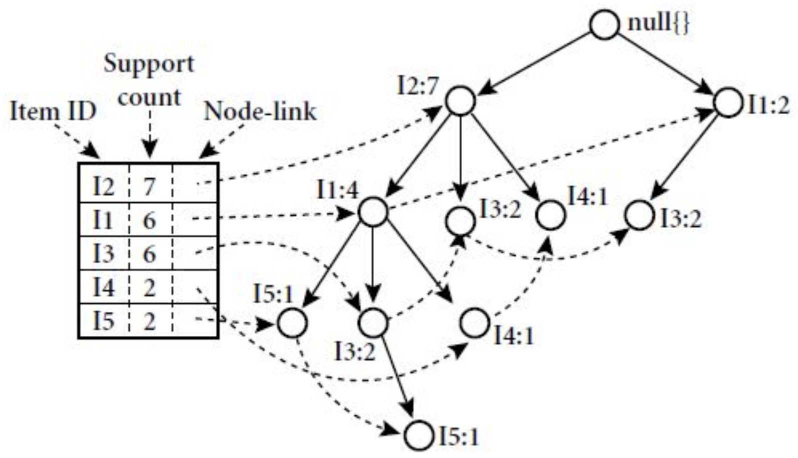
Example:



Transactional data for an *AllElectronics* branch.

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





Mining the FP-tree by creating conditional (sub-)pattern bases.

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

Example from FP-Growth

TID	***Items bought***	***(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}*

min_support = 3

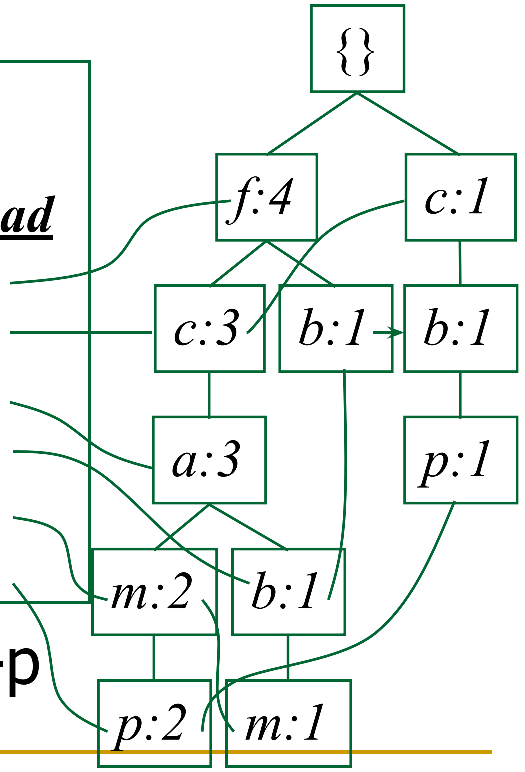
M's conditional database is

{{fca:2},{ fcab:1}}

Meaning is each of transactions contains itemset {f,c,a}, this can be merged with {m} to form the closed itemset *{fcam:3}*

Header Table

Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



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

Mining Closed Itemsets

- Sub-itemset pruning
  - if Y ɔ X, and sup(X) = sup(Y), X and all of X's descendants in the set enumeration tree can be pruned
    - DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}, where minSup=2
    - The projection of the first item, a_1 derives the frequent itemset, {a_1, …, a_{50} :2}, based on the itemset merging optimization
    - Because support({a_2}) = support({a_1, …, a_{50}}) and {a_2} is a proper subset of {a_1, …, a_{50}}
      - No need to examine a_2 and its projected database
    - Similar pruning can be done for a_3… a_{50}
    - Mining of closed itemsets in this data set terminates after mining a_1's projected database

Mining Closed Itemsets

- Item skipping
  - if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
    - DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}, where minSup=2
    - a_2 is in a_1's projected database and has the same support as a_2 in global header table, a_2 can be pruned from the global header table
    - Similar pruning can be done for a_3… a_{50}
    - No need to mine anything more after mining a_1's projected database

Mining Closed Itemsets

- Need of efficient closure checking
  - Check whether the newly found itemset is a subset of an already found closed itemset with the same support
  - Check whether the newly found itemset is a superset of an already found closed itemset with the same support
  - Uses Pattern-Tree structure
    - Similar to the FP-Tree except that all of the closed itemsets found are stored explicitly in the corresponding tree branches
- Algorithms: CLOSET, CLOSET+

MaxMiner: Mining Max-Patterns

- Extend the closed item methods with maximal frequent itemset
- 1st scan: find frequent items
  - A, B, C, D, E
- 2nd scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, DE, CDE

Potential max-patterns

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

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

Problems with the Association Mining

- Single minsup
  - It assumes that all items in the data are of the same nature and/or have similar frequencies
- Not true
  - In many applications, some items appear very frequently in the data, while others rarely appear

  e.g., in a supermarket, people buy *food processor* and *cooking pan* much less frequently than they buy *bread* and *milk*

Rare Item Problem

- If the frequencies of items vary a great deal, there will be two problems
  - If minsup is set too high, those rules that involve rare items will not be found
  - To find rules that involve both frequent and rare items, minsup has to be set very low
    - This may cause combinatorial explosion because those frequent items will be associated with one another in all possible ways

Mining Various Kinds of Rules or Regularities

- Multi-level
  - At different level of abstraction
- Multi-Dimensional
  - More than one dimension (e.g. items bought by Student)
- Quantitative association rules
  - Involve numeric attributes that have an implicit ordering among values (e.g. age)
- Constraint Based

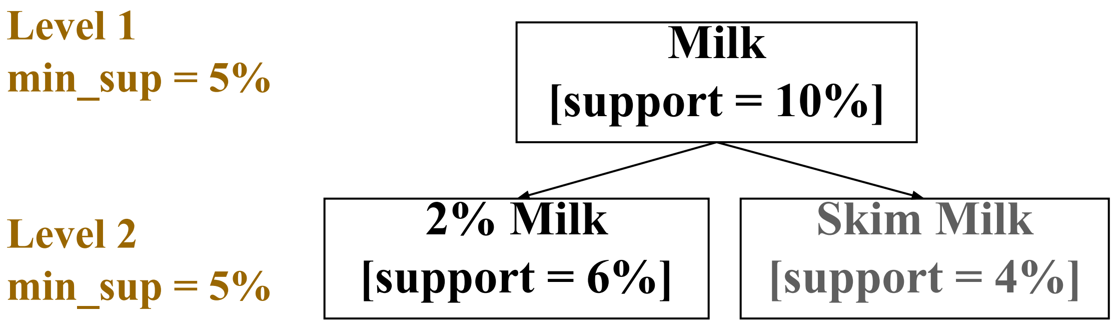
ML/MD Associations with Flexible Support Constraints

- Why flexible support constraints?
  - Real life occurrence frequencies vary greatly
    - Diamond, watch, pens in a shopping basket
  - Uniform support may not be an interesting model
- A flexible model
  - The lower-level, the more dimension combination, and the long pattern length, usually the smaller support
  - General rules should be easy to specify and understand
  - Special items and special group of items may be specified individually and have higher priority

Multiple-level Association Rules

- Items often form hierarchy
- Uniform support
  - The same minimum support threshold is used when mining at each level of abstraction
  - Only one minimum support is required
  - Search procedure is simplified

uniform support

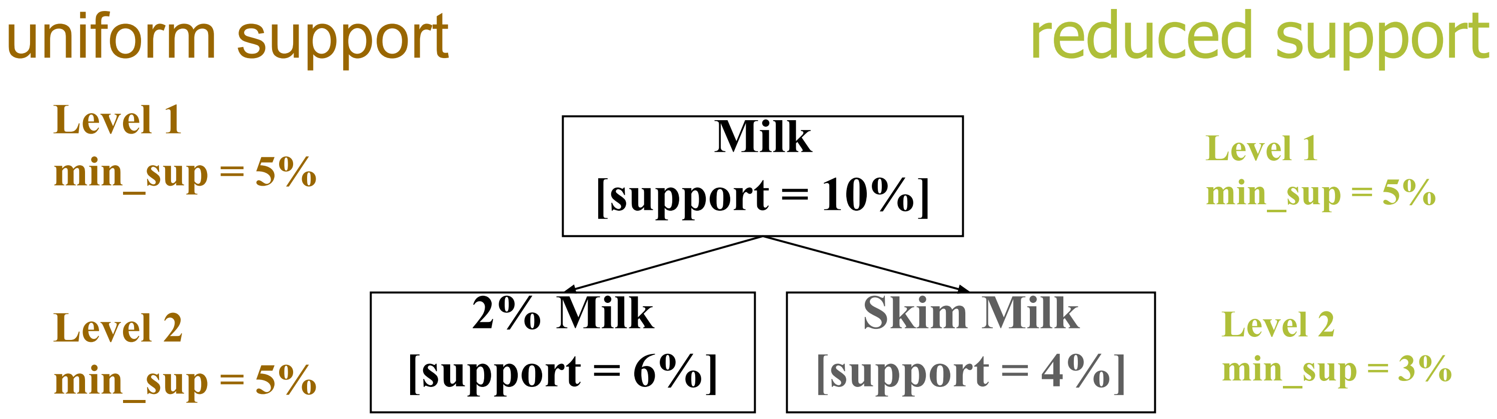


Multiple-level Association Rules

- Uniform support
  - Apriori can be adopted with the knowledge that the ancestor is a superset of its descendants and searching is avoided for any item whose ancestors do not have minimum support
  - Difficulties
    - Items at lower levels of abstraction will not occur as frequently as those at higher levels of abstraction
    - If minsup is too high, may miss some meaningful associations at low abstraction levels
    - If minsup is too low, may generate many uninteresting associations occurring at high abstraction levels

Multiple-level Association Rules

- Reduced Minimum support at lower levels
  - Each level of abstraction has its own minimum support threshold
  - The deeper the level of abstraction, the smaller the corresponding threshold is



Multiple-level Association Rules

- Item or Group based Minimum Support
  - As the groups are more important, it is desirable to set up user-specific, item or group based minimal support thresholds when mining
  - A group/combination can be visualize and set the low support threshold for the group (e.g. Laptop and Flash Drive) to checkout the association pattern containing items in this categories

Multi-level Association: Redundancy Filtering

- Apriori cannot be apply directly for reduced support and group support
- Problem: Some rules may be redundant due to "ancestor" relationships between items
  - Example
    - milk ⇒ wheat bread [support = 8%, confidence = 70%]
    - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]
  - Here, the first rule is an ancestor of the second rule
    - A rule R1 is an ancestor of a rule R2, if R1 can be obtained by replacing the items in R2 by their ancestors in a hierarchy

Multi-level Association: Redundancy Filtering

- Example
  - milk ⇒ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]
- A rule can be considered redundant if its support is close to the “expected” value, based on the rule’s ancestor, and thus remove that rule
  - First rule says, 8% support and 70% confidence and about one-quarter of milk is ‘2% milk’
  - Second rule is having approx 70% confidence and 2% milk are also samples of milk and a support quarter share in milk (i.e. 8% x ¼)
  - So, Rule 2 is not interesting because does not offer any additional information and is less general than Rule 1

Multi-dimensional Association

- Single-dimensional/Single-predicate/IntraDimensional rules

    buys(X, “milk”) ⇒ buys(X, “bread”)

    Predicate: buys as a dimension

- Multi-dimensional/Interdimensional rules
    - ≥ 2 dimensions or predicates
    - MD Association, searches for the frequent predicate sets
    - A K-predicate set is a set containing k conjuctive predicates
        - e.g. {age, buys, occupation} is a 3-predicate set

Multi-dimensional Association

- Inter-dimension assoc. rules (*no repeated predicates*)

  age(X,"19-25") ∧ occupation(X,"student") ⇒ buys(X,"coke")

- hybrid-dimension assoc. rules (*repeated predicates*)

  age(X,"19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")

Multi-dimensional Association

- Categorical Attributes
  - Finite number of possible values
  - No ordering among values
  - e.g. brand, color, occupation
  - Also called Nominal attributes
    - Their values are "names of things"
- Quantitative Attributes
  - Numeric
  - Implicit ordering among values
  - e.g. age, income, price

Multi-dimensional Association

- Techniques for mining MD Asso. regarding the treatment of Quantitative attributes
  - Using Static Discretization of Quantitative Attributes
  - Qunatitative attributes are discretized using predefined concept hierarchy
    - Occurs before mining
    - A concept hierarchy for attribute may be used to replace the original numeric values by labels
      - e.g. for income, interval labels as "0...20K', "21K...30K", ...
      - Here, discretization is static and predetermined
    - The discretized numeric attributes, with their interval labels can be treated as categorical attributes (where each interval is considered as a category)

Multi-dimensional Association

- Techniques for mining MD Asso. regarding the treatment of Quantitative attributes
  - Dynamic Discretization of Quantitative Attributes
  - Qunatitative attributes are discretized or clusterd into "bins" based on the distribution of the data
    - These bins may be further combined during the mining process
    - Discretization process is dynamic and established so as to satisfy some mining criteria, such as maximizing the confidence of the rules mined

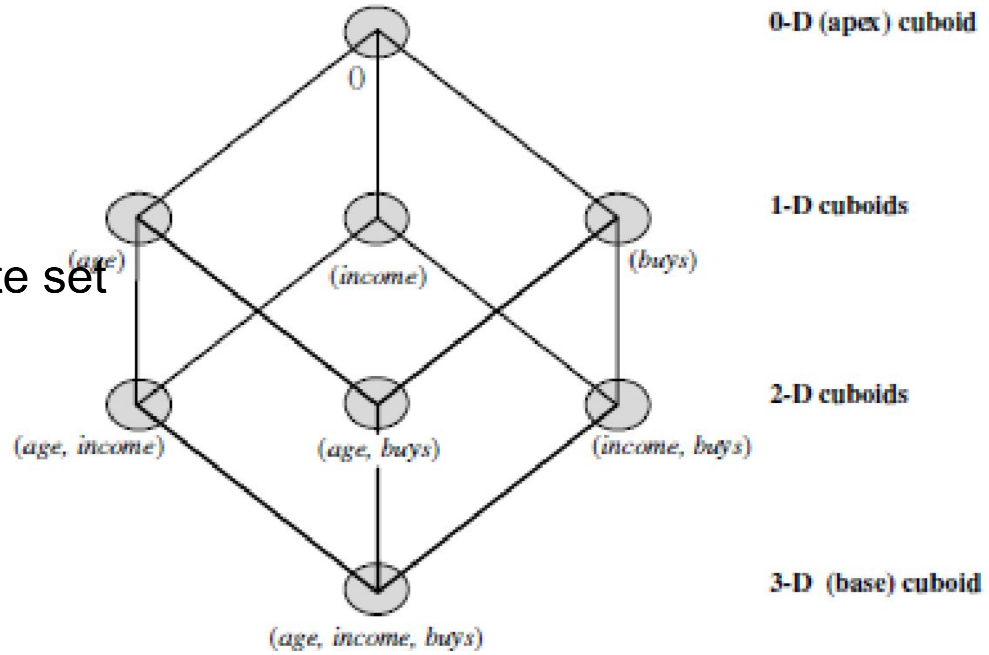
Multi-dimensional Association

- Differed from the Single-dimensional Association which searches for frequent itemsets
- MD Association, searches for the frequent predicate sets
  - A K-predicate set is a set containing k conjuctive predicates
  - Implemented
    - Data Cube
      - Stores aggregates (such as counts)

    3-D Cube

    N-D Cube for N-predicate set

    Detail later on



Mining Various Kinds of Rules or Regularities

- Multi-level
    - At different level of abstraction
- Multi-Dimensional
    - More than one dimension (e.g. item buys with customer age)
- Quantitative association rules
    - Involve numeric attributes that have an implicit ordering among values (e.g. age)
- Constraint Based

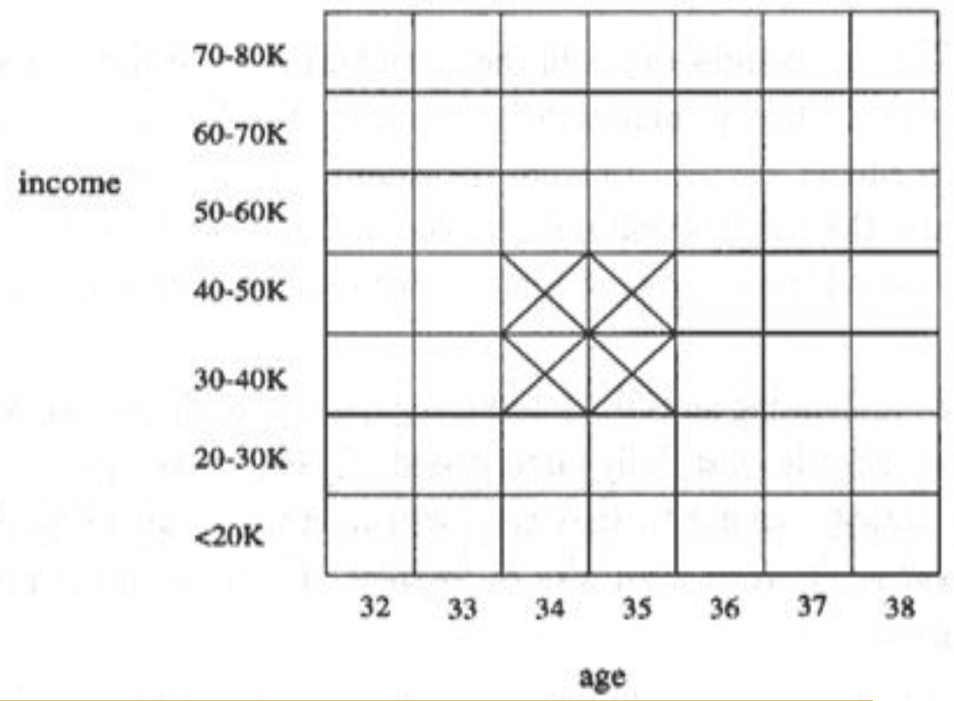
Mining Quantitative Association Rules

- The Association rule in the form of

  Quantitative attribute (s) Categorical attribute (s)
- Example of 2-D Quantitative Association Rule
  - $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
  - Age(X,"34-35") ∧ Income(X,"30K - 50K") Buys(X,"HDTV")
- Like Multi Dimension, here also Numeric attributes discretized dynamically

Mining Quantitative Association Rules

- Association Rule Clustering System (ARCS)
  - An approach that Maps pairs of Quantitative attributes onto a 2-D grid for tuples satisfying a given Categorical attribute condition
  - The grid is searched for clusters of points from which the association rules are generated
- Cluster "adjacent" association rules to form general rules using a 2-D grid



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

Mining Association Rule

- Frequent Pattern {A,B}
- Possible Association Rule A□B, B□A
- Support and confidence of the rule:

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

$$confidence(A \Rightarrow B) = P(B \mid A)$$

- Strong association rule with minimum thresholds.
- No enough!

Misleading "strong" association rule

- Suppose we are interested in analyzing transactions at *AllElectronics with respect to the purchase of* computer games and videos. Let *game refer to the transactions containing computer games, and video* refer to those containing videos. Of the 10,000 transactions analyzed, the data show that 6000 of the customer transactions included computer games, while 7500 included videos, and 4000 included both computer games and videos. Suppose that a data mining program for discovering association rules is run on the data, using a minimum support of, say, 30% and a minimum confidence of 60%. The following association rule is discovered:

$$buys(X, \text{"computer games"}) \Rightarrow buys(X, \text{"videos"})$$

$$[support = 40\%, confidence = 66\%].$$

- P(videos)

From Association Rule to Correlation Rule

$$A \Rightarrow B \; [support, confidence, correlation].$$

Lift is a simple correlation measure that is given as follows. The occurrence of itemset A is **independent** of the occurrence of itemset B if $P(A \cup B) = P(A)P(B)$; otherwise, itemsets A and B are **dependent** and **correlated** as events. This definition can easily be extended to more than two itemsets. The **lift** between the occurrence of A and B can be measured by computing

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}. \tag{6.8}$$

- Life=1: independent
- Life>1: positive correlated
- Life<1: negetive correlated

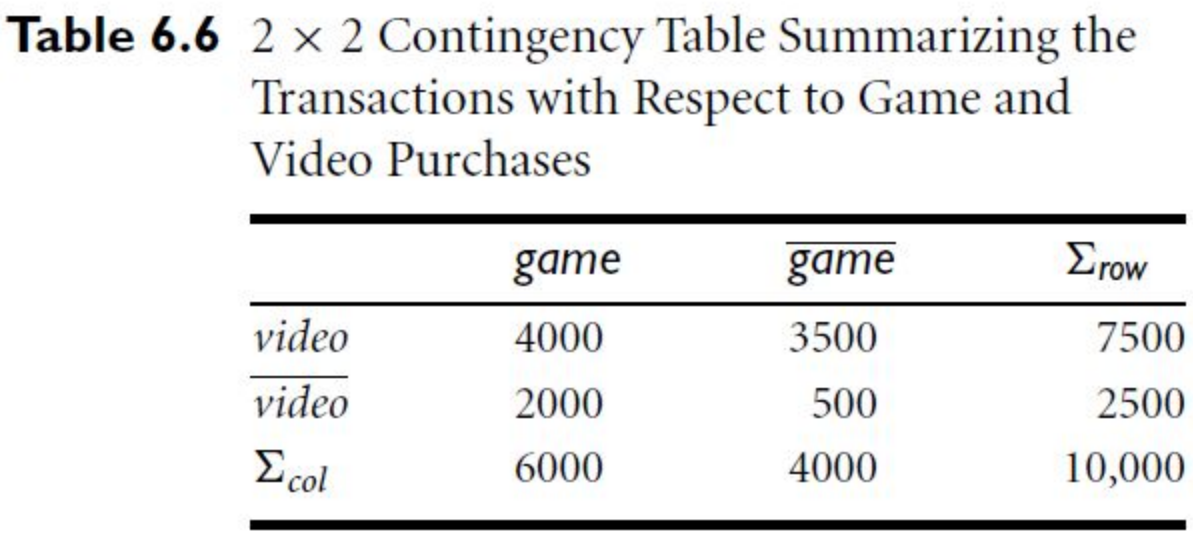


Table 6.6 2 × 2 Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

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

- P(V)=.75
- P(G)=.6
- P(VG) = .40
- Life(VG) = .40/(.75*.6)<1
- Negative correlated

Interestingness Measure: Correlations (χ^2)

- X^2 (chi-square) test

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

	$game$	$\overline{game}$	Σ_{row}
$video$	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

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

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected} = \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000} = 555.6.$$

Interestingness Measure: Correlations (χ^2)

- The larger the X^2 value, the more likely the variables are related
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

Interestingness Measure: Correlations (all_confidence)

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

	$game$	$\overline{game}$	Σ_{row}
$video$	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

- All_confidence(g,v)=0.53

Interestingness Measure: Correlations (**max confidence**)

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

	$game$	$\overline{game}$	Σ_{row}
$video$	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

Interestingness Measure: Correlations (cosine)

$$\cos ine(A,B) = \frac{P(A \cup B)}{\sqrt{P(A)P(B)}}$$

	game	$\overline{game}$	Σ_{row}
video	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

- cosine(g,v)=0.6

	$milk$	$\overline{milk}$	Σ_{row}
$coffee$	mc	$\overline{m}c$	c
$\overline{coffee}$	$m\overline{c}$	$\overline{mc}$	$\overline{c}$
Σ_{col}	m	$\overline{m}$	Σ

- Null-transactions: 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.

Data Set	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	all_conf.	cosine	lift	X^2
A_1	1,000	100	100	100,000	0.91	0.91	83.64	83,452.6
A_2	1,000	100	100	10,000	0.91	0.91	9.26	9,055.7
A_3	1,000	100	100	1,000	0.91	0.91	1.82	1,472.7
A_4	1,000	100	100	0	0.91	0.91	0.99	9.9
B_1	1,000	1,000	1,000	1,000	0.50	0.50	1.00	0.0
C_1	100	1,000	1,000	100,000	0.09	0.09	8.44	670.0
C_2	1,000	100	10,000	100,000	0.09	0.29	9.18	8,172.8
C_3	1	1	100	10,000	0.01	0.07	50.0	48.5

Constraint-Based Mining

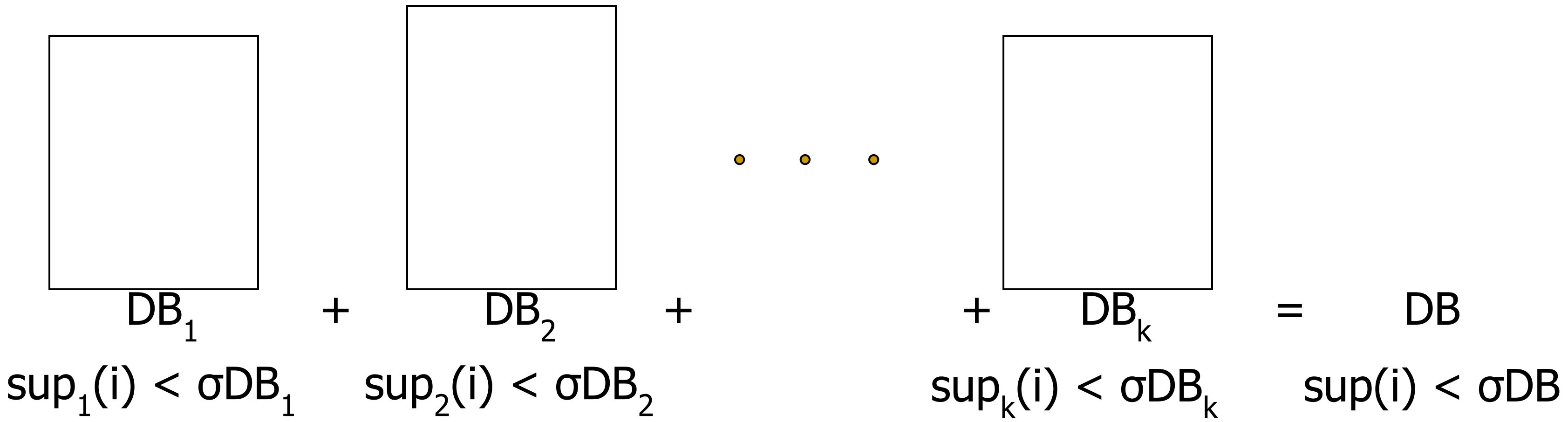
- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: Classification, Association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98
  - Dimension/level constraints:
    - In relevance to region, price, brand, customer category
  - Rule constraints
    - Small sales (price < \$10) triggers big sales (sum > \$200)
  - Interestingness constraints:
    - Strong rules (min_support ≥ 3%, min_confidence ≥ 60%)

Apriori Algorithm

- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

- 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*



Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
    - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules In *VLDB'96*

Association Rules - Summary

- Basic Concepts
  - Frequent Itemsets and Association Rules
  - Measures
- Applications of frequent pattern and associations
  - Market Basket
  - Weblog mining
  - Bioinformatics
- Efficient and Scalable Frequent Itemset Mining Methods
  - The Apriori Algorithm
    - Finding Frequent Itemsets Using Candidate Generation
    - Generating Association Rules from Frequent Itemsets
  - Improving the Efficiency of Apriori
  - Mining Frequent Itemsets without Candidate Generation
  - Mining Frequent Itemsets Using Vertical Data Format
- Are All the Pattern Interesting?—Pattern Evaluation Methods
  - Strong Rules Are Not Necessarily Interesting
  - From Association Analysis to Correlation Analysis
  - Selection of Good Measures for Pattern Evaluation
- Sequential Pattern Mining

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'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.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing:02.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
- G. Liu, H. Lu, W. Lou, J. X. Yu. On Computing, Storing and Querying Frequent Patterns. KDD'03.
- G. Grahne and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003

Ref: Vertical Format and Row Enumeration Methods

- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. DAMI:97.
- Zaki and Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.
- C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. KDD'02.
- F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki , CARPENTER: Finding Closed Patterns in Long Biological Datasets. KDD'03.
- H. Liu, J. Han, D. Xin, and Z. Shao, Mining Interesting Patterns from Very High Dimensional Data: A Top-Down Row Enumeration Approach, SDM'06.

Ref: Mining Correlations and Interesting Rules

- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97.
- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98.
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02.
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03.
- T. Wu, Y. Chen and J. Han, "Association Mining in Large Databases: A Re-Examination of Its Measures", PKDD'07

Ref: Freq. Pattern Mining Applications

- Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. ICDE'98.
- H. V. Jagadish, J. Madar, and R. Ng. Semantic Compression and Pattern Extraction with Fascicles. VLDB'99.
- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.
- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT'02.

Mine the Closed and Max-Patterns

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f

Min_sup = 2

Min_conf =50%