Mining Association Rules

Yi-Cheng Chen (陳以錚)
Dept. of Computer Science &
Information Engineering,
Tamkang University

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

Market-Basket transactions

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

Example of Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {Milk},

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2

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

- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

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**

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

An Example

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

- if minimum support is 2, minimum confidence is 2/3
- frequent itemset
  - {A}, {B}, {C}, {E}, {A,C}, {B,C}, {B,E}, {C,E}, {B,C,E}
- strong rule
  - {B, E}$\rightarrow$C (2/3)
  - C$\rightarrow$A (2/3), A$\rightarrow$C (2/2)

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

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**

Example of Rules:

{Milk,Diaper} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk,Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Beer} (s=0.4, c=0.5)

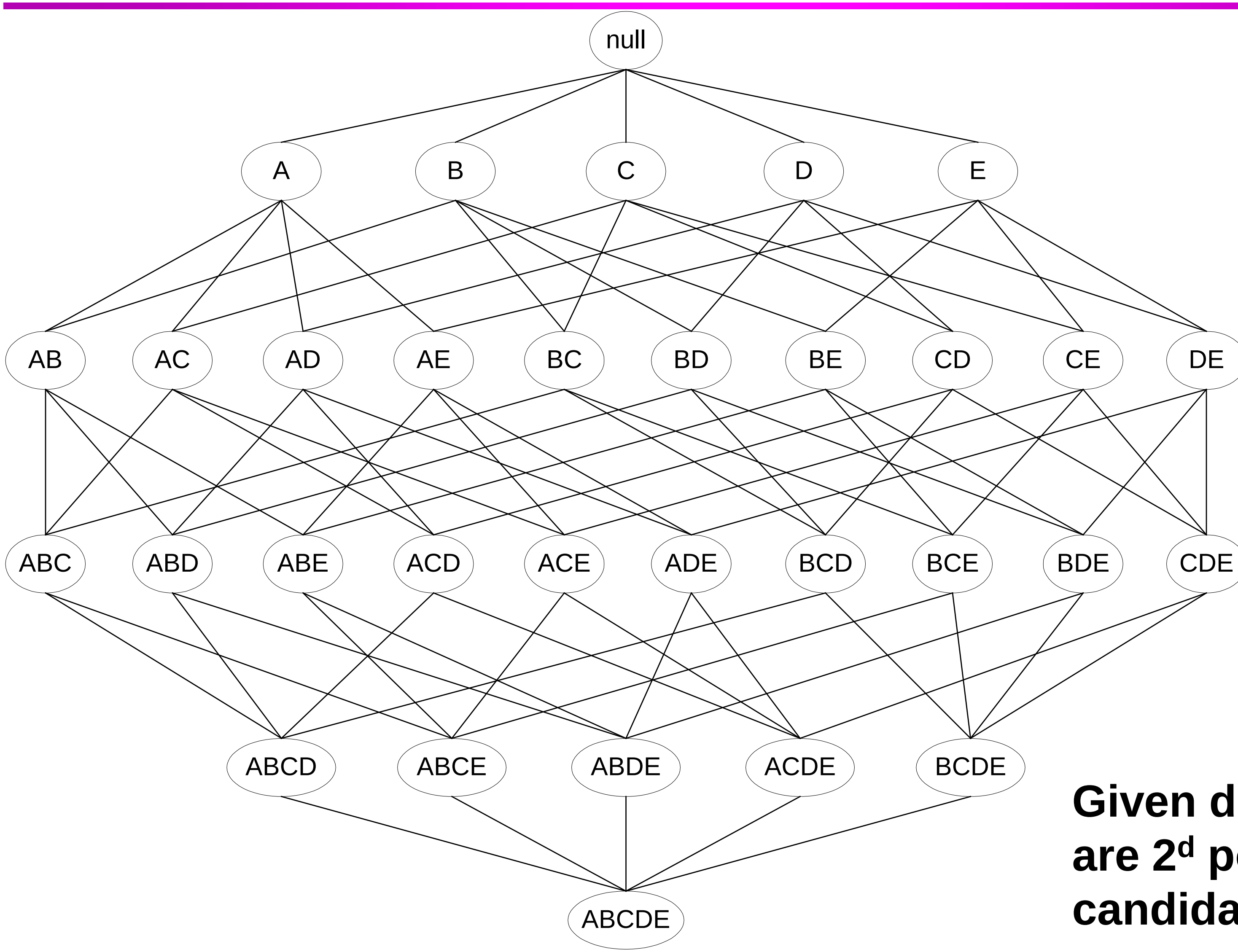
Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence

Basic Approach of Association Rule

- Process of association rule mining
  - step 1: find all frequent itemsets
  - step 2: generate strong association rules from frequent itemsets

Frequent Itemset Generation



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

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

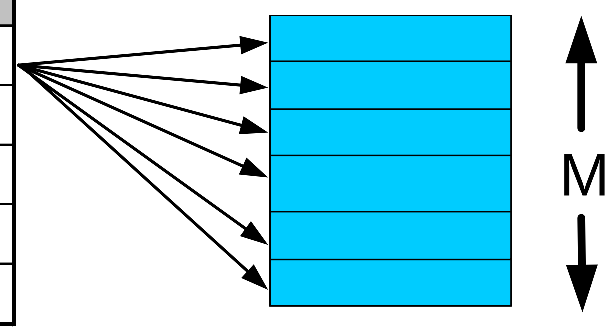
Transactions

List of Candidates

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

N

w



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Apriori Algorithm

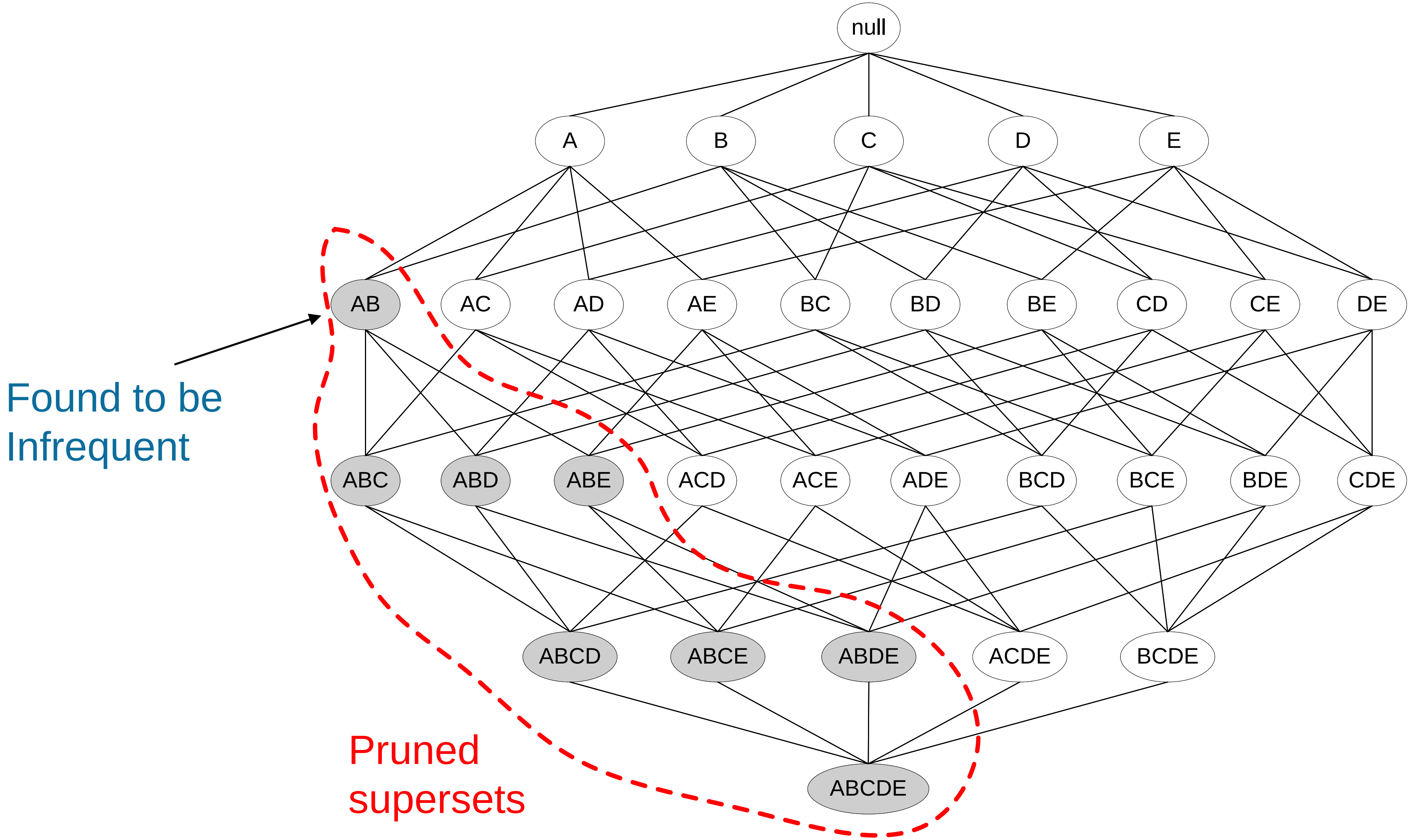
Apriori Algorithm

- Observation: Apriori property
  - all non-empty subsets of a frequent itemset must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets

Illustrating Apriori Principle



Apriori Algorithm

- Notations
  - frequent k-itemset (denoted as L_k): satisfy minimum support
  - candidate k-itemset (denoted as C_k): possible frequent k-itemsets
- Level-wise approach
  - (k-1)-itemsets are used to explore k-itemsets
  - join $C_k = L_{k-1} \otimes L_{k-1} = \{A \otimes B \mid A, B \in L_{k-1}, |A \cap B| = k-2\}$
  - prune C_k by subset test
  - generate L_k by scanning transaction DB

Min. support 50% (i.e., 2tx's)

BE=>C conf.:66%

Database TDB

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

1^{st} 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}

2^{nd} 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}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

Algorithm Apriori Candidate Generation

join step

insert into C_k

select $p.item_1$, $p.item_2$…, $p.item_{k-1}$, $q.item_{k-1}$

from L_{k-1} p, L_{k-1} q

where $p.item_1 = q.item_1$,…, $p.item_{k-2} = q.item_{k-2}$,

$p.item_{k-1} < q.item_{k-1}$;

prune step

for all itemsets $c \in C_k$ do

for all (k-1)-subsets s of c do

if ($s \notin L_{k-1}$) then

delete c from C_k

Important Details of Apriori

- How to generate candidates?
  - **Step 1: self-joining** L_k
  - **Step 2: pruning**
  - **Example of Candidate-generation**
  - L_3=**{***abc, abd, acd, ace, bcd***}**
  - **Self-joining:** 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***}**

Algorithm Apriori

```
L_1={frequent 1-itemsets};
for (k=2; L_{k-1} ≠ 0; k++) do begin
     C_k=apriori-gen(L_{k-1});
     for each transactions t ∈D do begin //scan DB
      C_t=subset(C_k, t) //get the subsets of t that are candidates
      for each candidate c ∈C_t do
           c.count++;
     end
     L_k={c ∈C_k|c.count ≥ minsup}
end
Answer=∪_k L_k;
```

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - 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

Implementation Subset Function

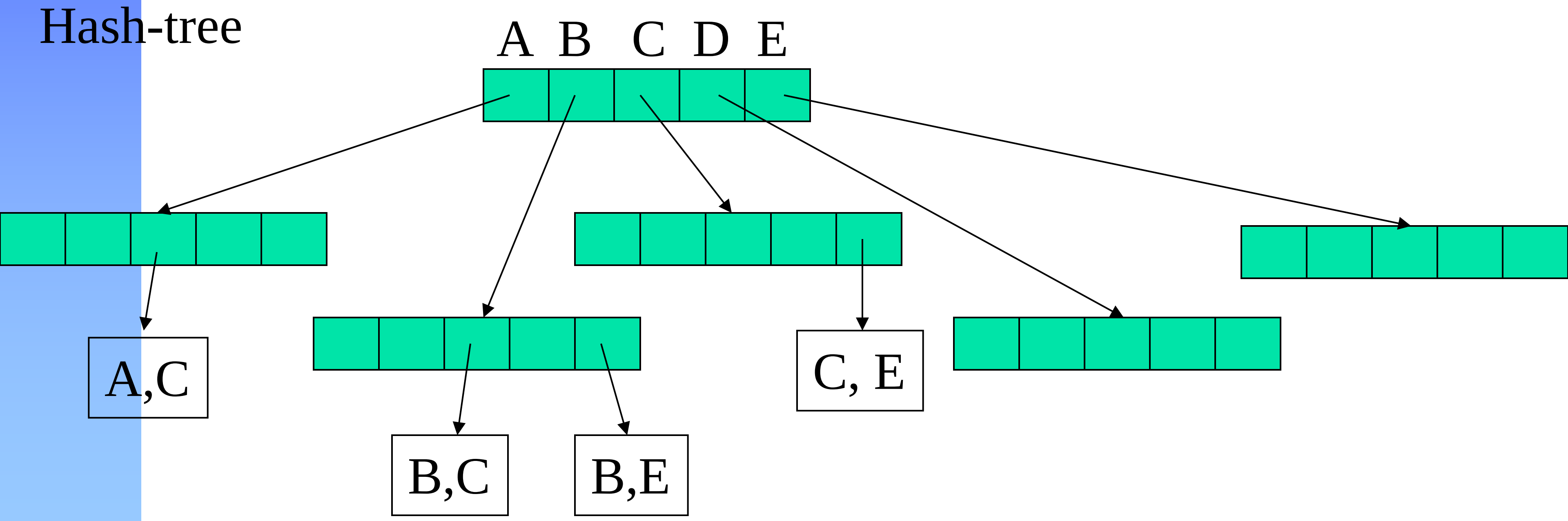
C_2

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

transaction: { B, C, E} has three 2-itemset candidates, i.e.,

{{B,C},{B,E},{C,E}}

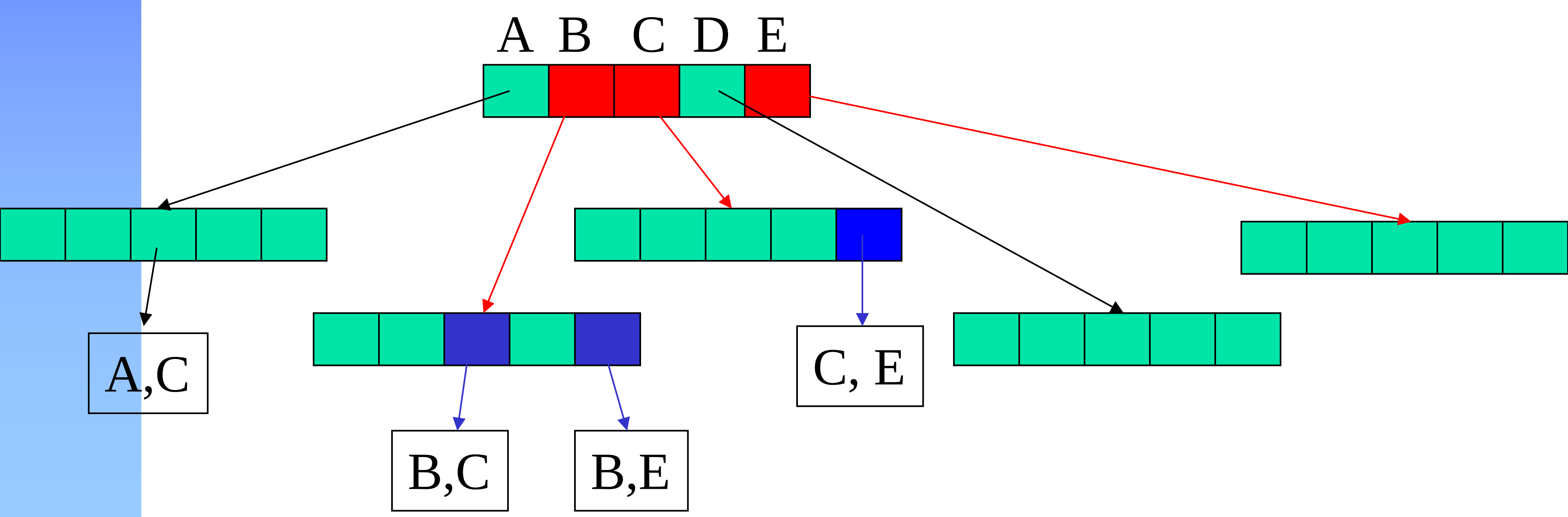
Hash-tree



Implementation Subset Function (cont.)

- For the root, hash on every item in t
- If in the interior node and reach this node by hashing item I, hash on each item that comes after I in t and recursively do.
- If in leaf, add the corresponding itemset into answer set.

{B,C,E}



Rules Generation

```
For each large item m do
  for each subset p of m do
     if (support(m)/support(p))>= min_confidence then
        output the rule p=>(m-p)
         with confidence=support(m)/support(p)
                 support=support(m)
```

m={a,c,d,e,f,g} 2000 tx's
p={a,d} 5000 tx's
{a,d}=>{c,e,f,g} confidence=40%,support=2000 tx's

Redundant Rules

- For the same support and confidence, if we have a rule {a,d}=>{c,e,f,g}, do we have
  - {a,d}=>{c,e,f}
  - {a}=>{c,e,f,g}
  - {a,d,c}=>{e,f,g}
  - {a}=>{d,c,e,f,g}

Improvement of Apriori Algorithm

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

DHP(Direct Hashing & Pruning)

- Observation of performance in association rule mining
  - initial candidate set generation is key issue to improve
  - amount of transaction data that must be scanned
- Major features of DHP
  - efficient generation for frequent itemsets
  - effective reduction on transaction database size
  - option of reducing #(database scan) required.

J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In SIGMOD'95

Efficient Generation of Frequent Itemsets

- Using hashing

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

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

Support=2

$$h(\{x,y\})=((ord(x)*10+ord(y)) \bmod 7$$

	{C,E}				{B,E}		{A,C}
	{C,E}		{B,C}		{B,E}		**{C,D}**
	{A,D}	{A,E}	{B,C}		{B,E}	{A,B}	**{A,C}**
H_2	3	1	2	0	3	1	3
	0	1	2	3	4	5	6

L1={A,B,C,E}
Pruning {A,D} and {C,D}

C_2

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

$C_2=L_1*L_1$

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

Effective Reduction on Transaction Database Size

A transaction is used to determine L_{k+1} only if the transaction consists of K+1 L_k

D_2

TID	Items	
100	A, C, D	{A,C}
200	B, C, E	{B,C}, {B,E},{C,E}
300	A, B, C, E	{A,C},{B,C},{B,E},{C,E}
400	B, E	{B,E}

D_3

TID	Items
200	B, C, E
300	A, B, C, E

TID	Items
200	B, C, E
300	B, C, E

A qualified item appears in at least k of Ck in a transaction.

Partitioning

- Observation: any potential frequent itemset appears as a frequent itemset in at least one of the partitions.
- Transaction DB is divided into non-overlapping partitions
- Partition size is chosen to be resident in main memory

A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

Partitioning Algorithm

1. Divide D into partitions $D^1, D^2, \ldots, D^{p;}$
2. For I = 1 to p do
3. $\quad L^i = \text{Apriori}(D^i);$
4. $C = L^1 \cup \ldots \cup L^p;$
5. Count C on D to generate L;

Partitioning (cont'd)

- Two phases scanning
  - First scan: generates a set of all potentially frequent itemsets
    - Each partition generates the local frequent itemsets
  - Second scan: actual support is measured
    - Collection of local frequent itemset = global candidate itemset
    - Global frequent itemsets are found by scan DB

Partitioning Adv/Disadv

- ***Advantages:***
  - Adapts to available main memory
  - Easily parallelized
  - Maximum number of database scans is two
- ***Disadvantages:***
  - May have many candidates during second scan

Sampling

- Sample the database and apply Apriori to the sample.
- ***Potentially Large Itemsets (PL):*** Large itemsets from sample
- ***Negative Border (BD^-):***
  - Generalization of Apriori-Gen applied to itemsets of varying sizes.
  - Minimal set of itemsets which are not in PL, but whose subsets are all in PL.

H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Borders of frequent itemsets

- Itemset **X** is more ***specific*** than itemset **Y** if **X superset of Y** (notation: **Y<X**). Also, **Y** is more ***general*** than **X** (notation: **X>Y**)

- **The Border:** Let **S** be a collection of frequent itemsets and **P** the lattice of itemsets. The ***border*** **Bd(S)** of **S** consists of all itemsets **X** such that ***all more general itemsets*** than **X** are in **S** and ***no pattern more specific*** than **X** is in **S**.

$$Bd(S) = \left\{ X \in P \,\middle|\, \begin{array}{l} \text{for all } Y \in P \text{ with } Y \prec X \text{ then } Y \in P, \\ \text{and for all } W \in P \text{ with } X \prec W \text{ then } W \notin S \end{array} \right\}$$

Positive and negative border

- **Border**

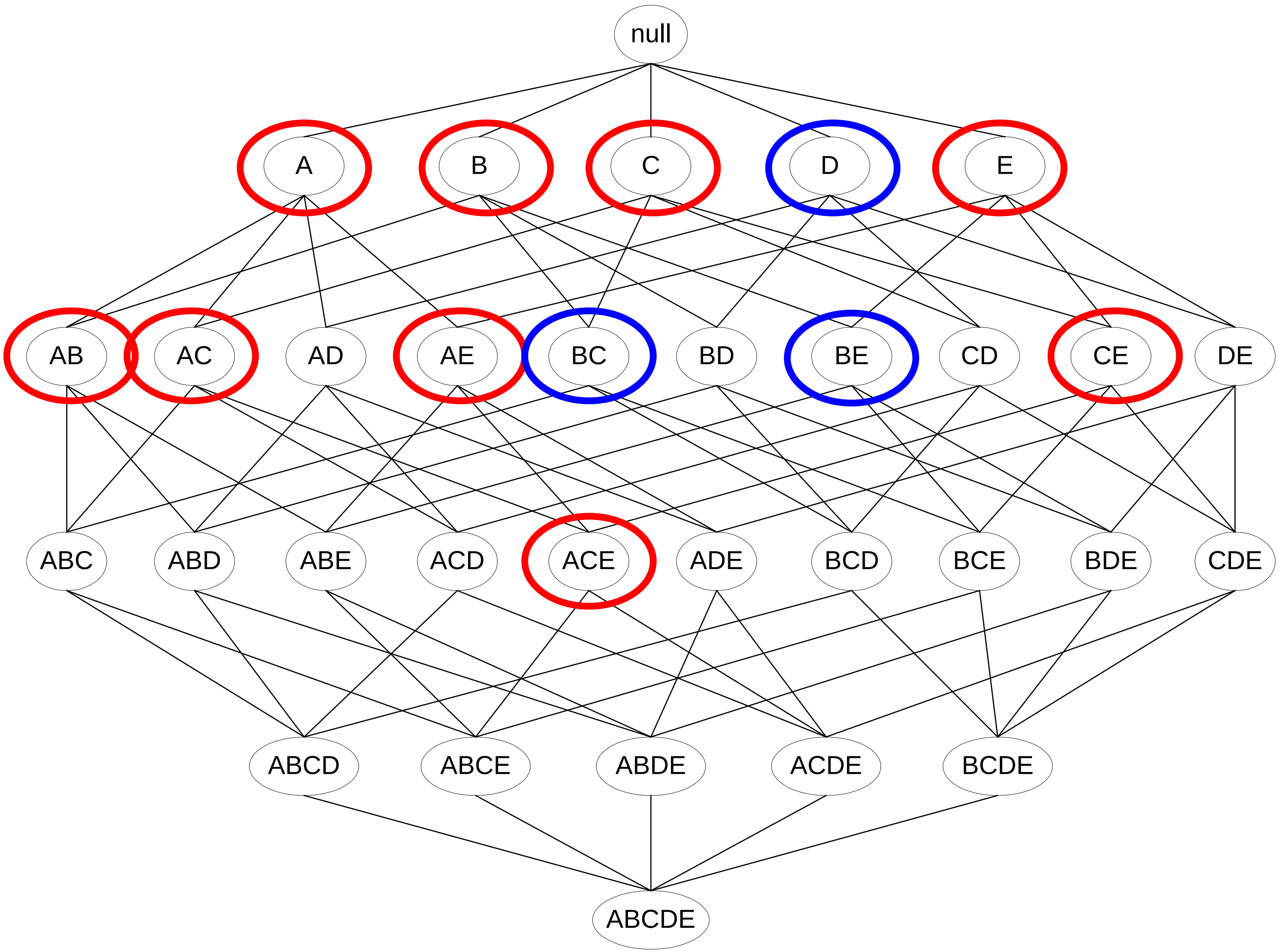
$$Bd(S) = \left\{ X \in P \,\middle|\, \begin{array}{l} \text{for all } Y \in P \text{ with } Y \prec X \text{ then } Y \in S, \\ \text{and for all } W \in P \text{ with } X \prec W \text{ then } W \notin S \end{array} \right\}$$

- **Positive border:** Itemsets in the border that are also frequent (belong in **S**)

$$Bd^{+}(S) = \left\{ X \in S \,\middle|\, \text{for all } Y \in P \text{ with } X \prec Y \text{ then } Y \notin S \right\}$$

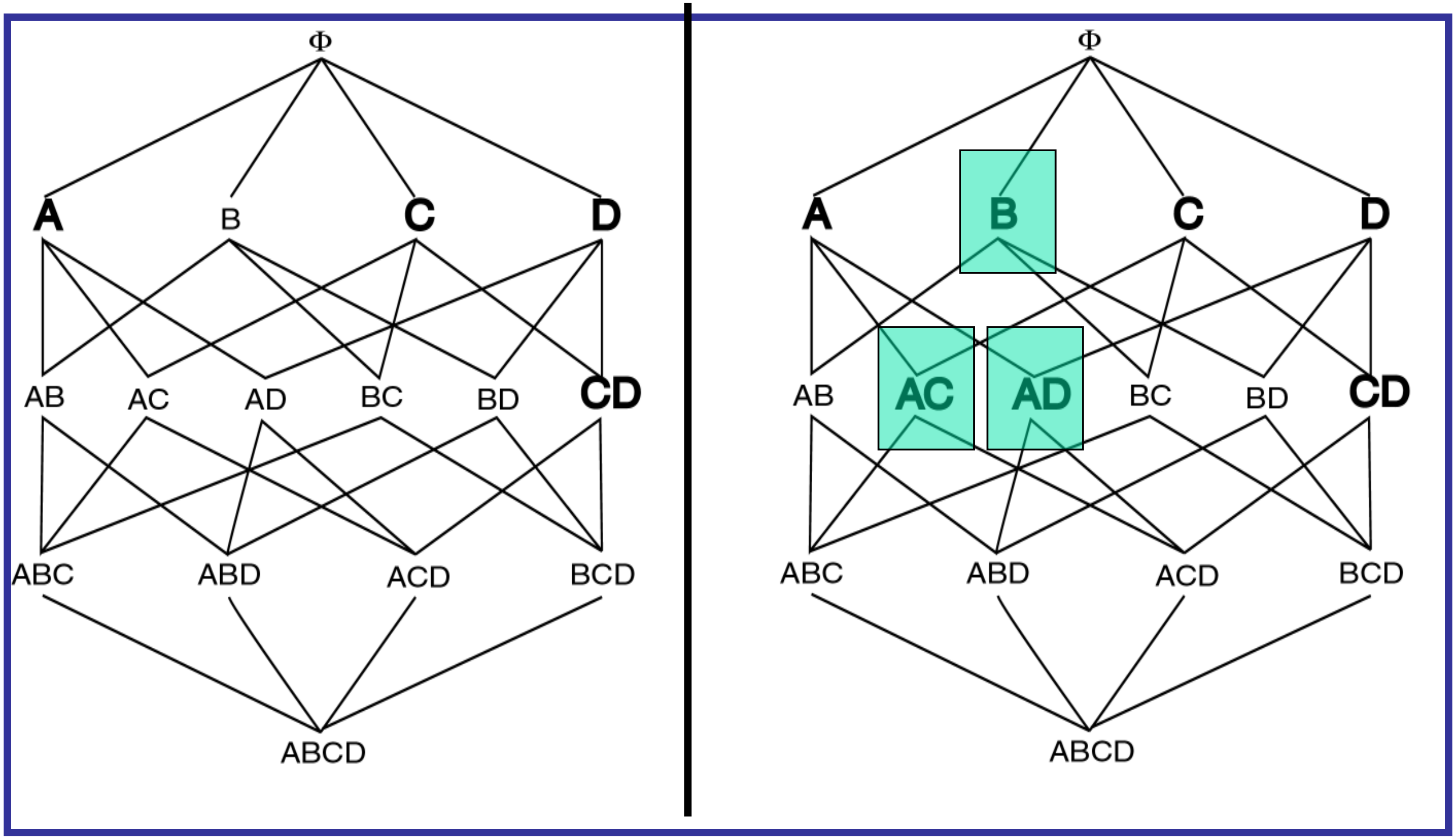
- **Negative border:** Itemsets in the border that are not frequent (do not belong in **S**)

$$Bd^{-}(S) = \left\{ X \in P \setminus S \,\middle|\, \text{for all } Y \in P \text{ with } Y \prec X \text{ then } Y \in S \right\}$$



Bd-(S) = {{D},{B,C},{B,E}}, Bd+(S) = {{A,B},{A,C,E}}

Negative Border Example



PL

PL $\cup$BD^{-}(PL)

Sampling Algorithm

1. D_s = sample of Database D;
2. PL = Large itemsets in D_s using smaller support;
3. C = PL $\cup$ BD^-(PL);
4. Count C in Database using support;
5. ML = large itemsets in BD^-(PL);
6. If ML = $\varnothing$ then done
7. else
8. C = repeated application of BD^-;
9. Count C in Database;

Sampling Adv/Disadv

- ***Advantages:***
  - Reduces number of database scans to one in the best case and two in worst
  - Scales better
- ***Disadvantages:***
  - Potentially large number of candidates in second pass

Apriori Example

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

Sampling Example

- Find AR assuming s = 20%
- D_s = { t_1,t_2}
- Smalls = 10%
- PL = {{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}
- BD^-(PL)={{Beer},{Milk}}
- ML = {{Beer}, {Milk}}
- Repeated application of BD^- generates all remaining itemsets

Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $C^{100}_1 + C^{100}_2 + \ldots + C^{100}_3 = 2^{100}-1 = 1.27*10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

FP-Growth

- Motivation
  - Mining in main memory to reduce #(DB scans)
  - Without candidate generation
  - More frequently occurring items will have better chances of sharing item than less frequently occurring items

J. Han, J. Pei, and Y. Yin: "Mining frequent patterns without candidate generation". In Proc. ACM-SIGMOD'2000, pp. 1-12, Dallas, TX, May 2000.

FP-Growth (cont'd)

- Frequent pattern Growth
- Divide-and-conquer strategy
- Algorithm
  - Phase 1: Construct FP-Tree (frequent-pattern tree)
  - Phase 2: FP-Growth (frequent pattern growth)
    - Divide FP-tree into conditional FP-tree (conditional DB), each associated with one frequent item
    - Mine each such DB separately

FP-Trees Construction

- Step 1: Find frequent 1-item, sorted items in frequency descending order by scanning DB

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

min_support = 3

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

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

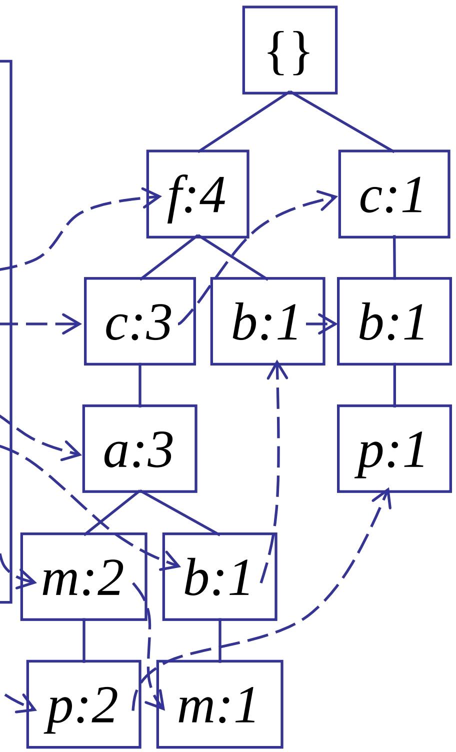
FP-Trees Construction (cont.)

Step 2: Scan DB and construct the FP-tree

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

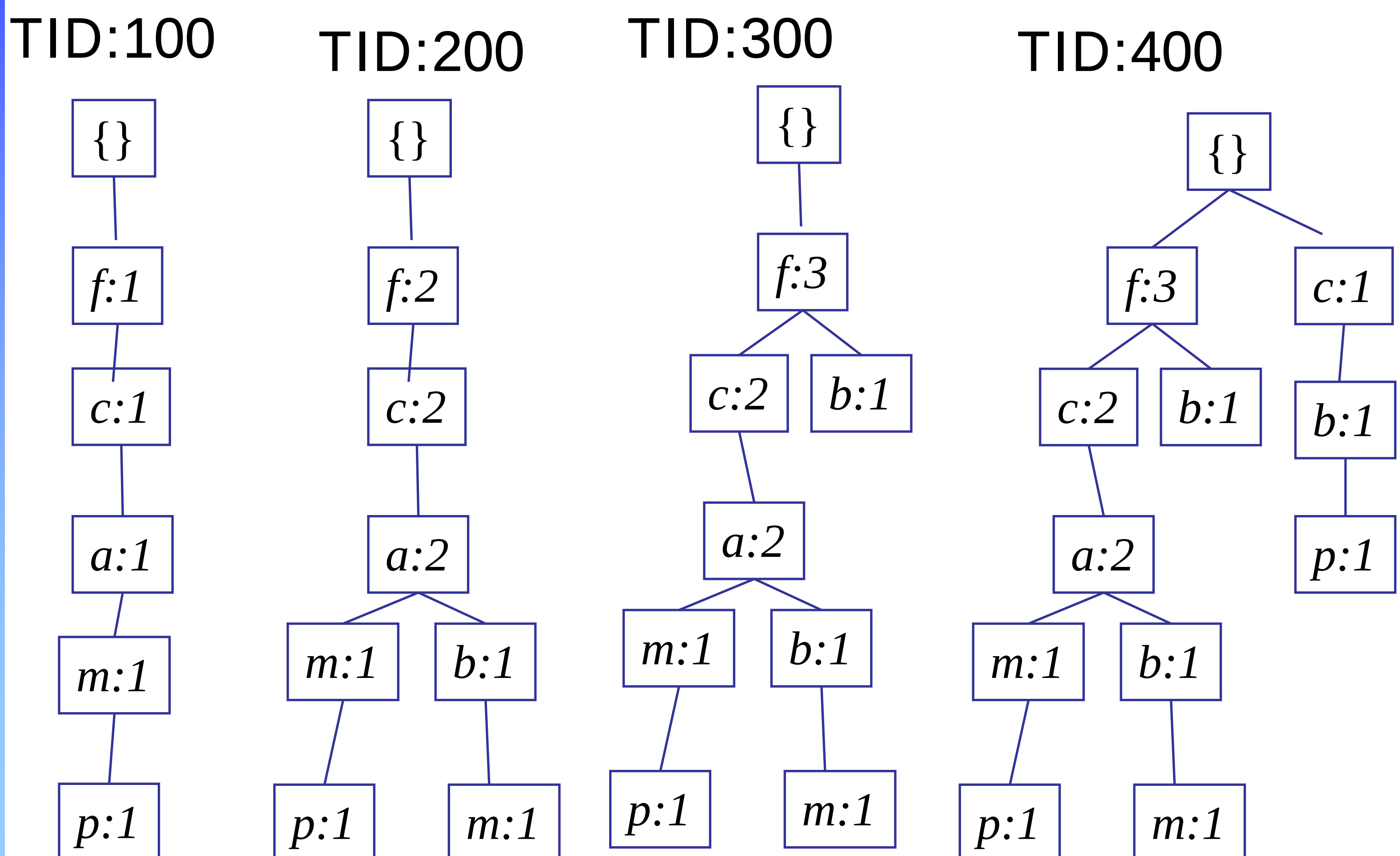
Header Table

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



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

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



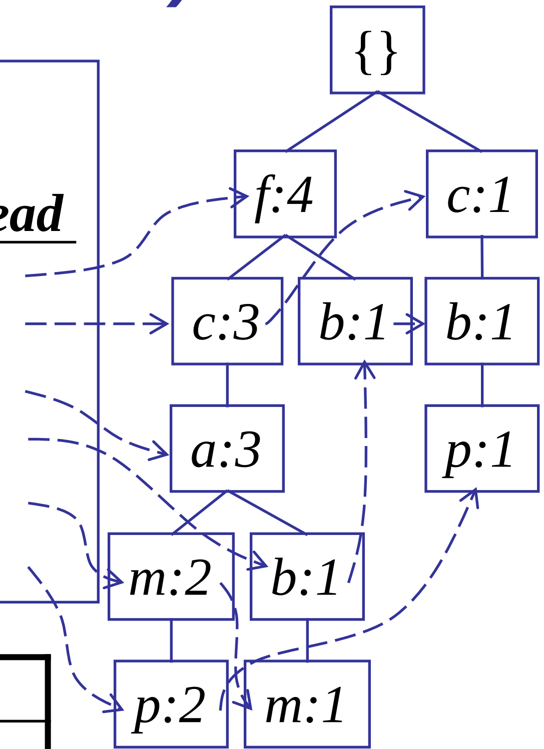
FP-Growth Overview

- Start from each frequent 1-pattern (initial suffix pattern), construct conditional pattern base
  - Conditional base: the set of prefix paths in FP-tree co-occurring with the suffix pattern
- Constructs corresponding conditional FP-tree
- Mining recursively on such tree.
- Pattern growth is achieved by the concatenation of suffix pattern with frequent patterns generated from conditional FP-tree

FP-Growth Overview (cont'd)

Header Table

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



min_support = 3

Item	Cond. Pattern base	Cond. FP-tree	Frequent patterns
c	f:3	f:3	fc:3
a	fc:3	fc:3	fca:3,fa:3,ca:3
b	fca:1,f:1,c:1		
m	fca:2,fcab:1	fca:3	fm:3,cm:3,am:3,fcm:3,fam:3,cam:3,fcam:3
p	fcam:2,cb:1	c:3	cp:3

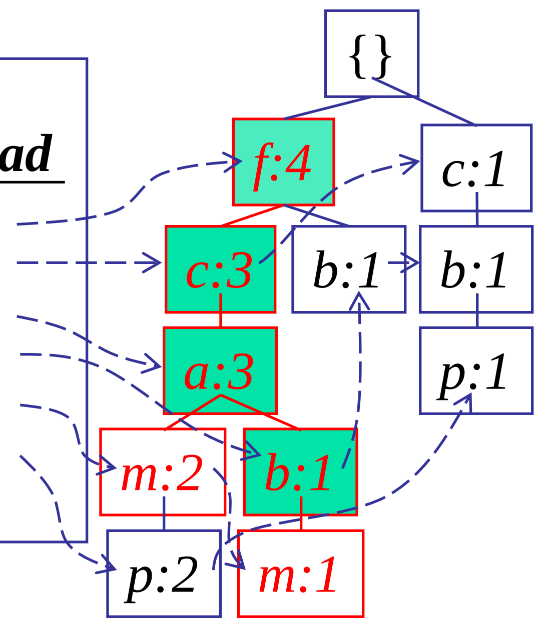
Construct Conditional FP-tree

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base

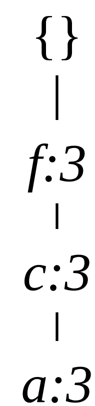
min_support = 3

Header Table

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



m-conditional **pattern base:**
fca:2, fcab:1



m-conditional **FP-tree**

All frequent patterns concerning *m*

fm, cm, am,
fcm, fam, cam,
fcam

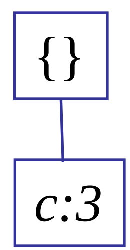
Construct Conditional FP-tree (cont'd)

- p's cond. pattern base: fcam:2,cb:1
  - Accumulate the count for each item
    - f:2 c:3,a:2,m:2,b:1
  - Sort frequent items in count descending order
  - Construct cond. FP tree

Min_support=3

Before	Sorted
fcam:2	c:2
cb:1	c:1

➔



➔

All frequent patterns relating *p*

cp

Principles of Frequent Pattern Growth

- Pattern growth property
  - Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.
- "*abcdef* " is a frequent pattern, if and only if
  - "*abcde* " is a frequent pattern, and
  - "*f* " is frequent in the set of transactions containing "*abcde* "

Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Leads to focused search of smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

Presentation of Association Rules

Presentation of Association Rules

	Body	Implies	Head	Supp (%)	Conf (%)	F	G	H	I
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4				
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	20.46	29.05				
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04				
4	cost(x) = '0.00~1000.00'	==>	revenue(x) = '1000.00~1500.00'	10.45	14.84				
5	cost(x) = '0.00~1000.00'	==>	region(x) = 'United States'	22.56	32.04				
6	cost(x) = '1000.00~2000.00'	==>	order_qty(x) = '0.00~100.00'	12.91	69.34				
7	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '0.00~500.00'	28.45	34.54				
8	order_qty(x) = '0.00~100.00'	==>	cost(x) = '1000.00~2000.00'	12.91	15.67				
9	order_qty(x) = '0.00~100.00'	==>	region(x) = 'United States'	25.9	31.45				
10	order_qty(x) = '0.00~100.00'	==>	cost(x) = '0.00~1000.00'	59.17	71.86				
11	order_qty(x) = '0.00~100.00'	==>	product_line(x) = 'Tents'	13.52	16.42				
12	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	23.88				
13	product_line(x) = 'Tents'	==>	order_qty(x) = '0.00~100.00'	13.52	98.72				
14	region(x) = 'United States'	==>	order_qty(x) = '0.00~100.00'	25.9	81.94				
15	region(x) = 'United States'	==>	cost(x) = '0.00~1000.00'	22.56	71.39				
16	revenue(x) = '0.00~500.00'	==>	cost(x) = '0.00~1000.00'	28.45	100				
17	revenue(x) = '0.00~500.00'	==>	order_qty(x) = '0.00~100.00'	28.45	100				
18	revenue(x) = '1000.00~1500.00'	==>	cost(x) = '0.00~1000.00'	10.45	96.75				
19	revenue(x) = '500.00~1000.00'	==>	cost(x) = '0.00~1000.00'	20.46	100				
20	revenue(x) = '500.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	19.67	96.14				
21									
22									
23	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
24	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
25	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
26	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
27	cost(x) = '0.00~1000.00' AND order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	33.23				

Sheet1

Maximal and closed

Max-patterns

- Max-pattern: frequent patterns without proper frequent super pattern
  - BCDE, ACD are max-patterns
  - BCD is not a max-pattern

Min_sup=2

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

R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.

What is Closed Itemset?

- O and I are finite sets of transactions and items respectively
- f(O):items common to all transactions $o \in O$
- g(I): transactions relate to all items $i \in I$

What is Closed Itemset?

- Ex:

  $f(\{100,300\})=\{AC\}$

  $g(\{BE\})=\{200,300,400\}$

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

What is Closed Itemset?

- Galois closure operators : h = f o g
- An itemset $C \in I$ is a closed itemset if $h(C)=f(g(C))=C$
- Ex: $h(AC)=f(g(AC))=AC$

  so AC is a closed itemset

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

What is Closed Itemset?

- Ex: $h(\{B,C\})=f(g(\{B,C\}))=f(200,300)=\{B,C,E\}$

So {B,C} is not a closed itemset.

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

Why ? Please go over this paper.

Maximal Frequent Itemset

Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

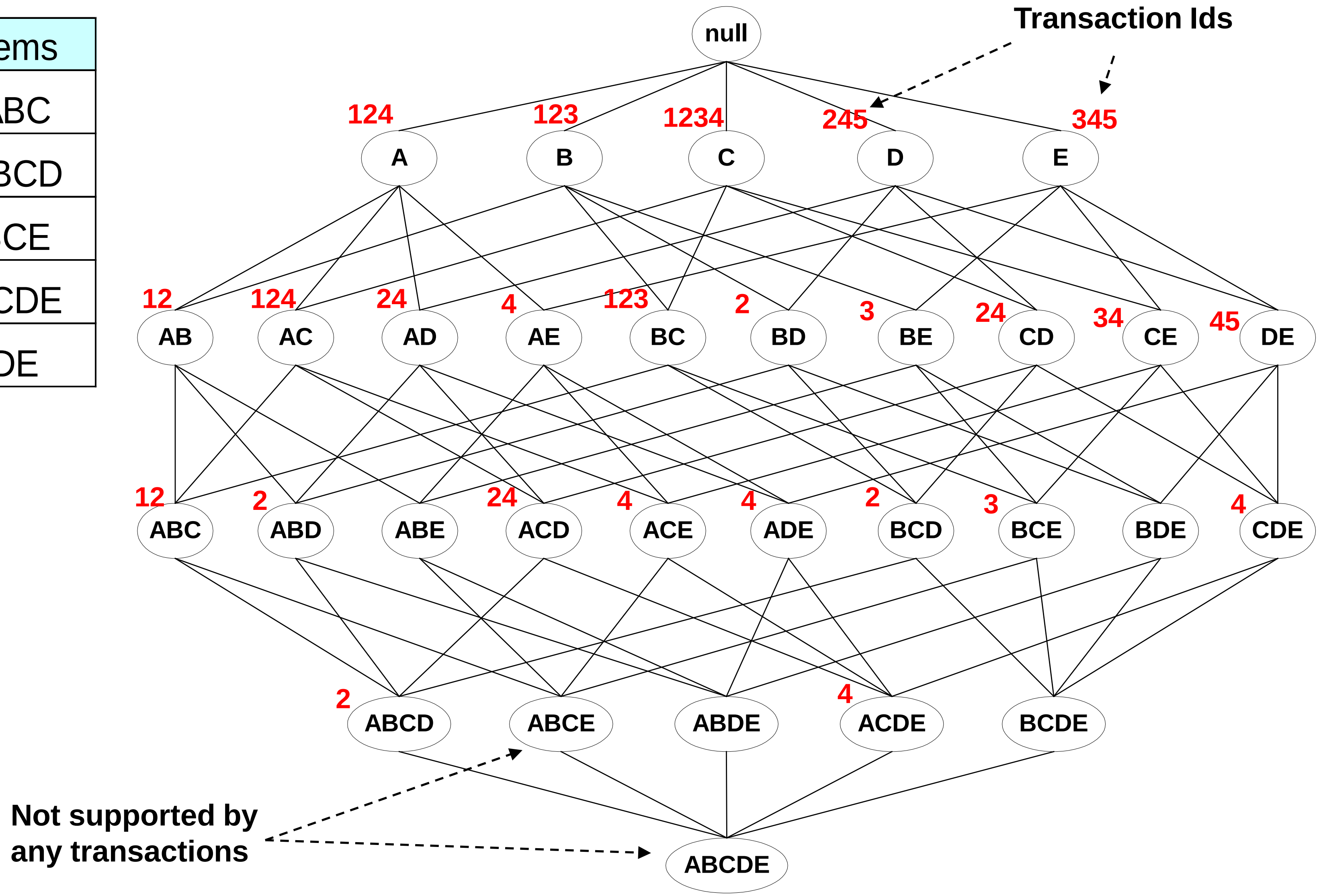
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

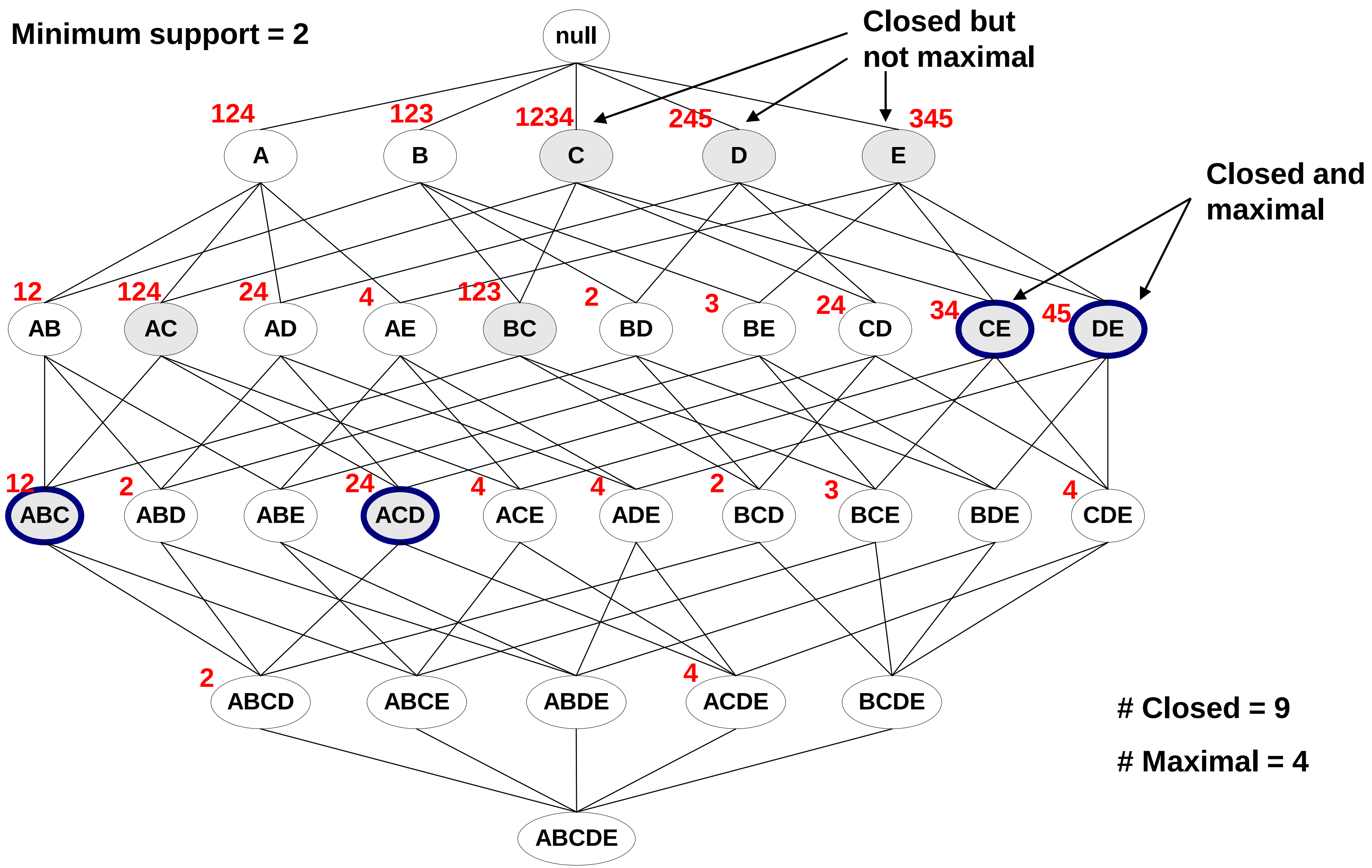
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

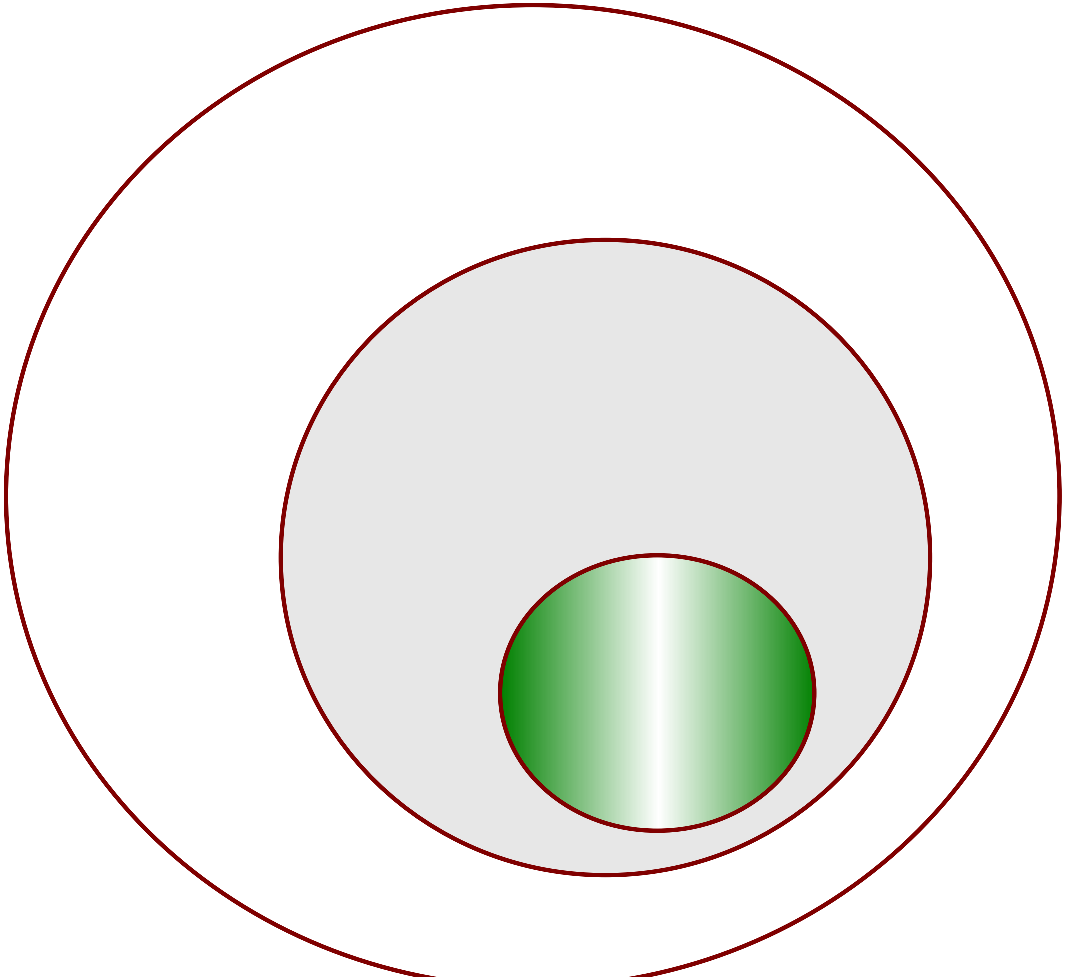
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs Closed Frequent Itemsets



Maximal vs Closed Itemsets



Closed Association Rules

- Large number of frequent itemsets (especially when the support threshold is **low**) and a huge number of association rules
- **Frequent closed itemset**: An itemset *X* is a **closed itemset** if there exists no itemset Y such that
  - Y is a proper superset of X
  - every transaction containing *X* also contains Y

N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99, 398-416, Jerusalem, Israel, Jan. 1999.

Closed Association Rules

- **Association rule on frequent closed itemsets:** Rule $X \Rightarrow Y$ is an association rule on frequent closed itemsets if
  - (1) both X and $X \cup Y$ are frequent closed itemsets.
  - (2) there does not exist frequent closed itemset Z such that $X \subset Z \subset (X \cup Y)$.
  - (3) the confidence of the rule passes the given min. conf

Closed Association Rules (cont'd)

TID	Items
100	*a,c,d,e,f*
200	*a,b,e*
300	*c,e,f*
400	*a,c,d,f*
500	*c,e,f*

Given minimum support 2

Total frequent itemsets:20:

{a},{c},{d},{e},{f},{a,c},{a,d},{a,e},{a,f}, {c,d},{c,e},{c,f},{d,f},{e,f},{a,c,d},{a,c,f}, {a.d.f},{c,d,f},{c.e.f},**{a,c,d,f}**

Closed frequent itemsets:

{a, c, d, f}, {c, e, f}, {a, e}, {c, f}, {a}, {e}

Given minimum confidence 50%,

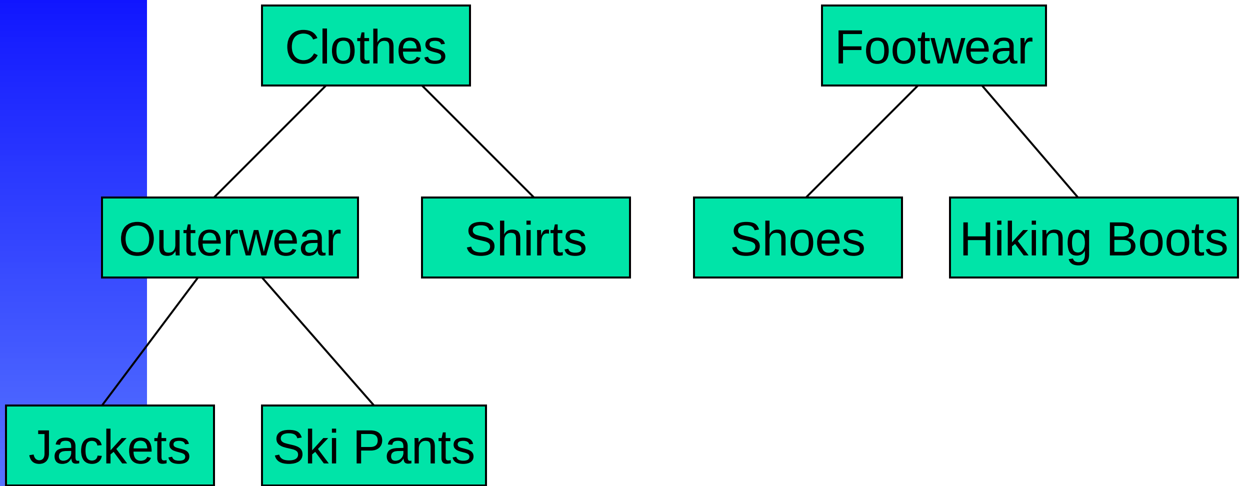
Closet association rule

{c, f} $\Rightarrow$ {a, d} (2,50%), {a} $\Rightarrow$ {c, d, f} (2,67%),

{e} $\Rightarrow$ {c, f} (3,75%), {c, f} $\Rightarrow$ {e} (3,75%),

{e} $\Rightarrow$ {a} (2,50%), {a} $\Rightarrow$ {e} (2,67%)

Multilevel Association Rules



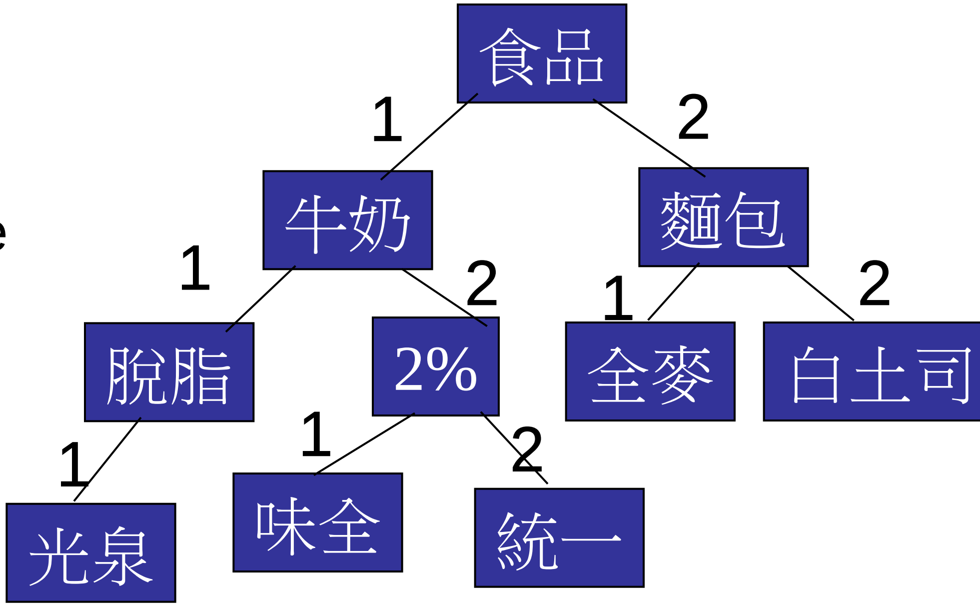
Freq. pattern	Support
Jacket	2
Outerwear	3
Clothes	4
Shoes	2
Hiking Boots	2
Footwear	4
OW, HB	2
Clothes, HB	2
OW, FW	2
Clothes, FW	2

Tx	Items bought
100	Shirt
200	Jacket, Hiking Boots
300	Ski Pants, Hiking Boots
400	Shoes
500	Shoes
600	Jacket

	sup(30%)	conf(60%)
OW -> HB	33%	66%
OW -> FW	33%	66%
HB -> OW	33%	100%
HB -> Clothes	33%	100%
Jacket -> HB	16%	50%
Ski Pants -> HB	16%	100%

Multiple-Level Association Rules

- Items often form hierarchy
- Items at the lower level are expected to have lower support
- Rules regarding itemsets at appropriate levels could be quite useful
- Transaction database can be encoded based on dimensions and levels



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Mining Multi-Level Associations

- A top down, progressive deepening approach:
  - First find high-level strong rules:

    milk $\rightarrow$ bread [20%, 60%].
  - Then find their lower-level "weaker" rules:

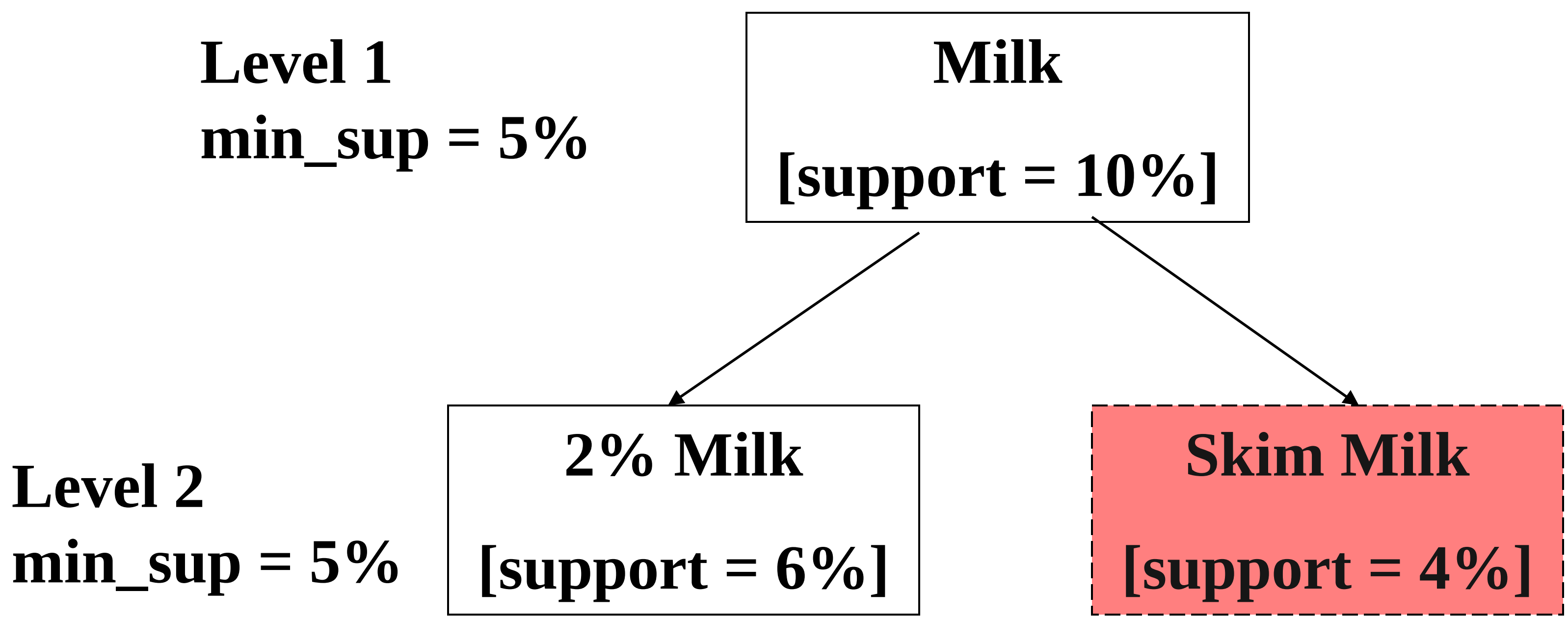
    2% milk $\rightarrow$ wheat bread [6%, 50%].
- Variations at mining multiple-level association rules
  - Level-crossed association rules:

    2% *milk* $\rightarrow$ *wheat bread*
  - Association rules with multiple, alternative hierarchies:

    2% *milk* $\rightarrow$ *bread*

Uniform Support

Multi-level mining with uniform support



Uniform Support (cont'd)

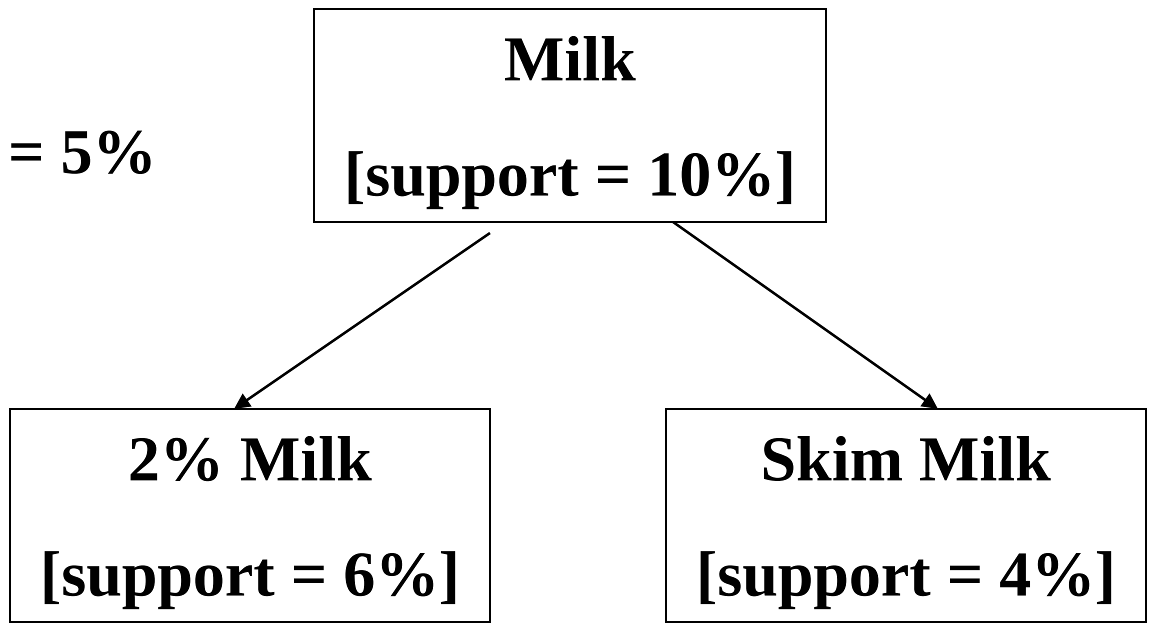
- Uniform Support: the same minimum support for all levels
  - + No need to examine itemsets containing any item whose ancestors do not have minimum support.
  - – Lower level items do not occur as frequently. If support threshold
    - too high $\Rightarrow$ miss low level associations.
    - too low $\Rightarrow$ generate too many high level associations.

Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 3%



Search strategies in Reduced Support

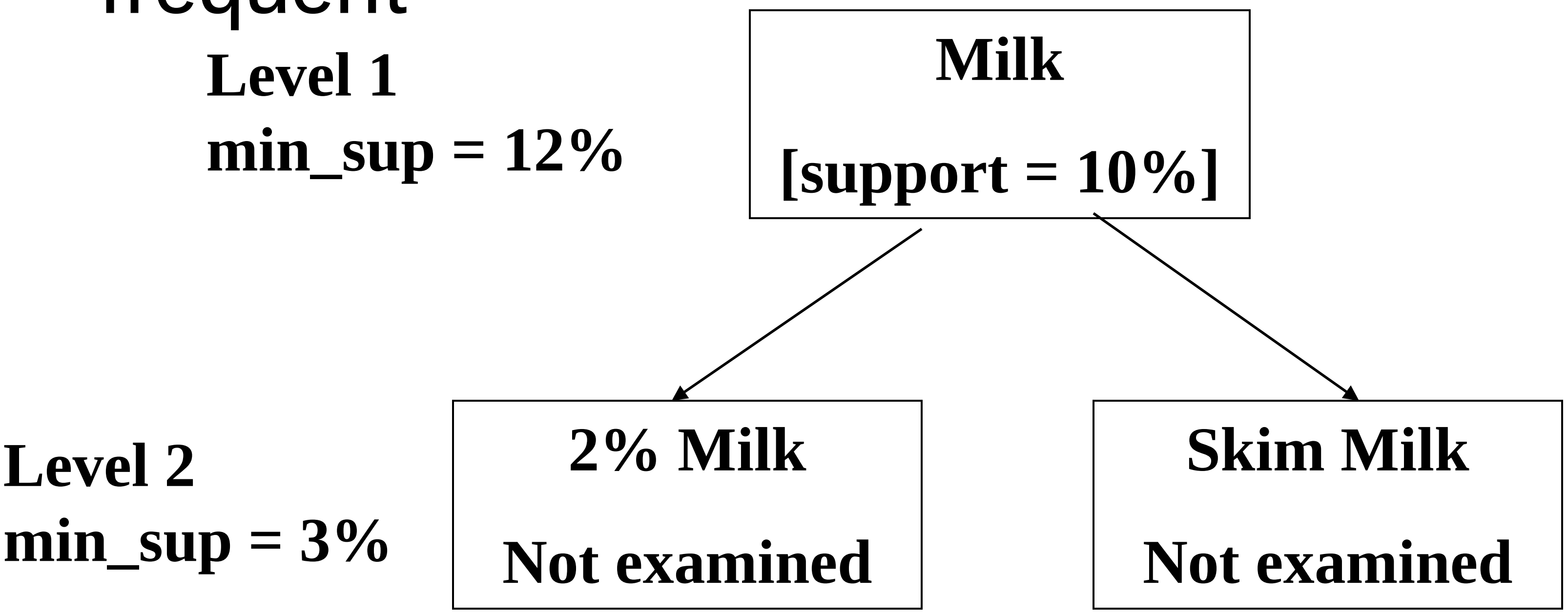
- Reduced Support: reduced minimum support at lower levels
  - 4 search strategies:
    - Level-by-level independent
    - Level-cross filtering by single item
    - Level-cross filtering by k-itemset
    - Controlled level-cross filtering by single item

Level by Level Independent

- Full breadth search
- No background knowledge of frequent itemsets is used to pruning
- Each node is examined, regardless of whether or not its parent node is found to be frequent.

Level-cross Filtering by Single Item

- An item at the i-th level is examined iff its parent node at the (i-1)-th level is frequent



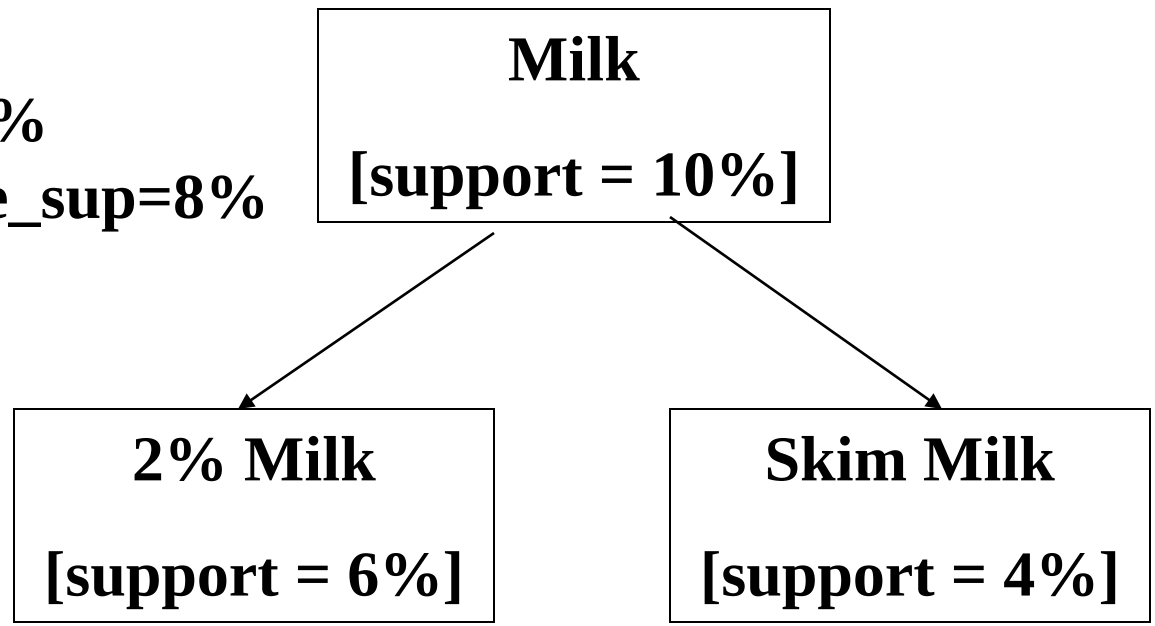
Level-cross Filtering by K-itemset

- A k-itemset at the i-th level is examined iff its corresponding parent k-itemset at the (i-1)-th level is frequent

Controlled Level-cross Filtering by Single Item

Level 1
min_sup = 12%
Level_passage_sup=8%

Level 2
min_sup = 3%



ML 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
  - Special items and special group of items may be specified individually and have higher priority

Quantitative Association Rules

Multidimensional Association Rules

- Single dimensional association rule
  - E.g.: buys(bread) $\wedge$ buys(milk) $\Rightarrow$ buys(butter)
- Multidimensional association rule
  - E.g.: age(34-35) $\wedge$ income(30K-50K) $\Rightarrow$ buys(HDTV)
- Attributes types
  - Categorical
    - finite number of possible values, no ordering among values
  - Numerical
    - numeric, implicit ordering among values

Example of Quantitative Association Rules

Record ID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	yes	2

Rule	Support	Confidence
<Age:30..39>and<Married:Yes>==><NumCars:2>	40%	100%
<Age:20..29>==><NumCars:0..1>	60%	100%

Mapping to Boolean Association Rules Problem

TID	Age:20-29 **(A)**	Age:30-40 **(B)**	Married: Yes **(C)**	Married: No **(D)**	NumCars :0 **(E)**	Numcars :1 **(F)**	NumCars :2 **(G)**
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

TID	Items
100	A,D,F
200	A,C,F
300	A,D,E
400	B,C,G
500	B,C,G

Dilemma of Discretization

- Min_Support
  - Increasing the number of intervals results in lower support for any single interval.
- Min_Conf
  - Some rules may have minimum confidence only when an itemset in the antecedent consists of a small interval.

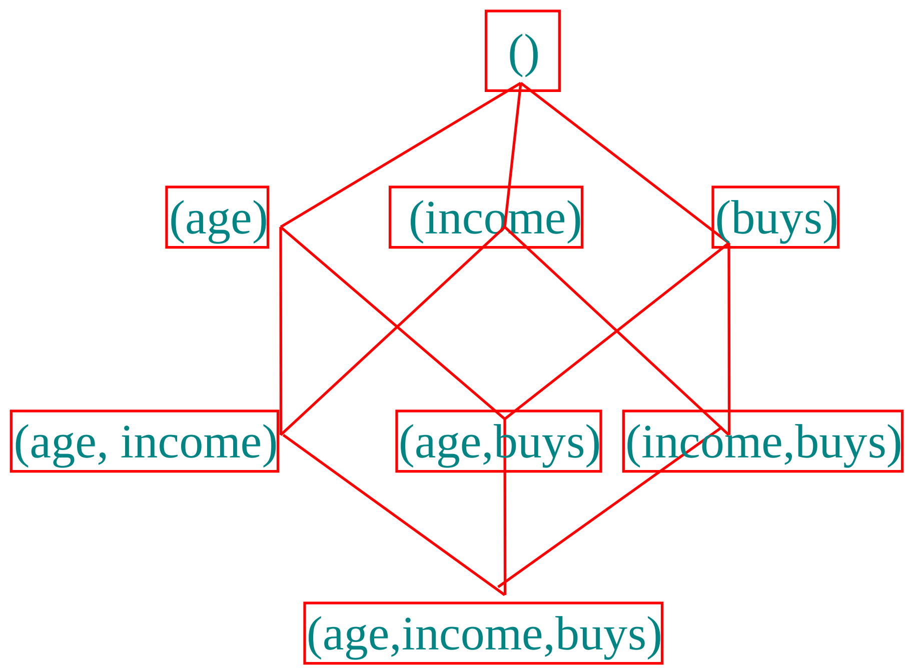
Rule A=>C
confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$)

Mining Quantitative Association Rules

- Approaches
  - Static discretization of quantitative attributes
  - Quantitative association rule (discretized based on distribution of data)
  - Distance-based association rule (discretized based on semantic meaning of interval)

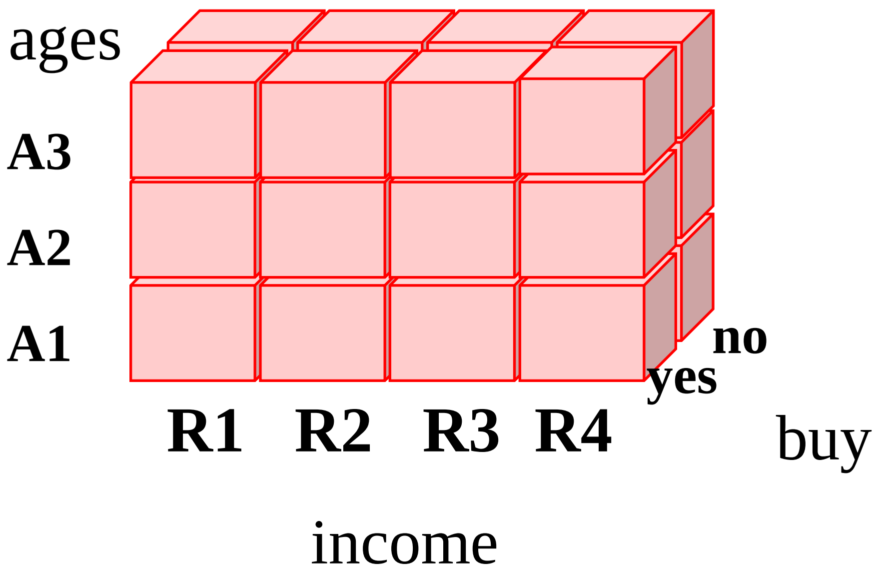
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.



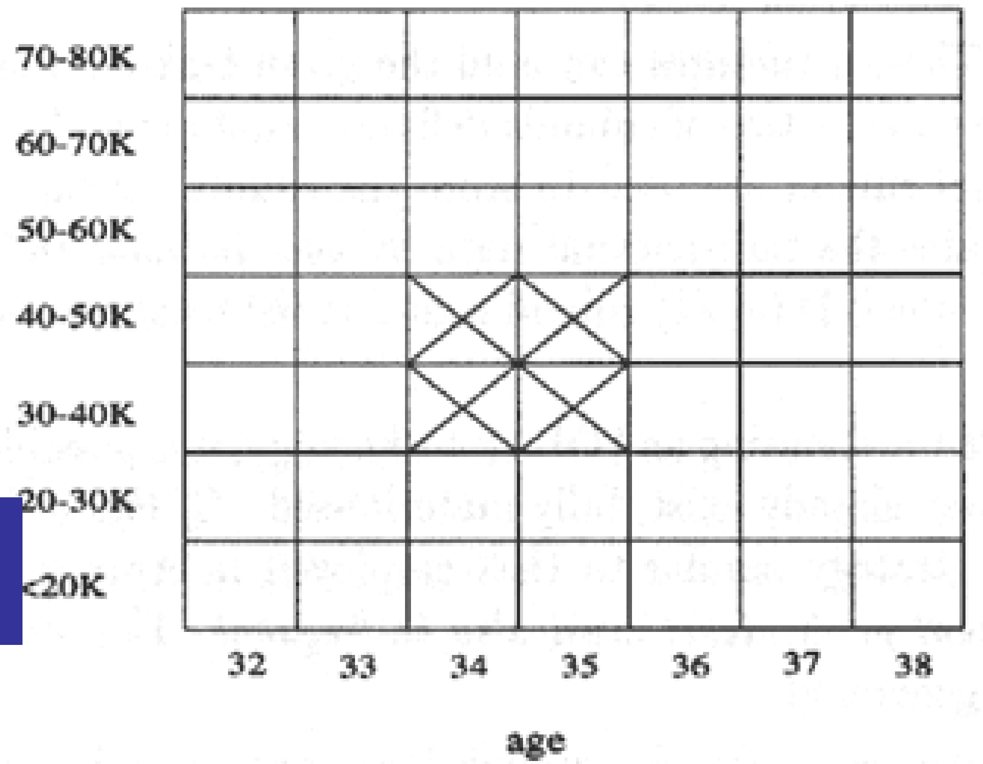
Static Discretization of Quantitative Attributes (cont'd)

- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the n-predicate sets.



Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized.
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster "adjacent"
  - association rules to form general rules using a 2-D grid.



age(34-35) $\wedge$ income(30K-50K) $\Rightarrow$ buys(high resolution TV)

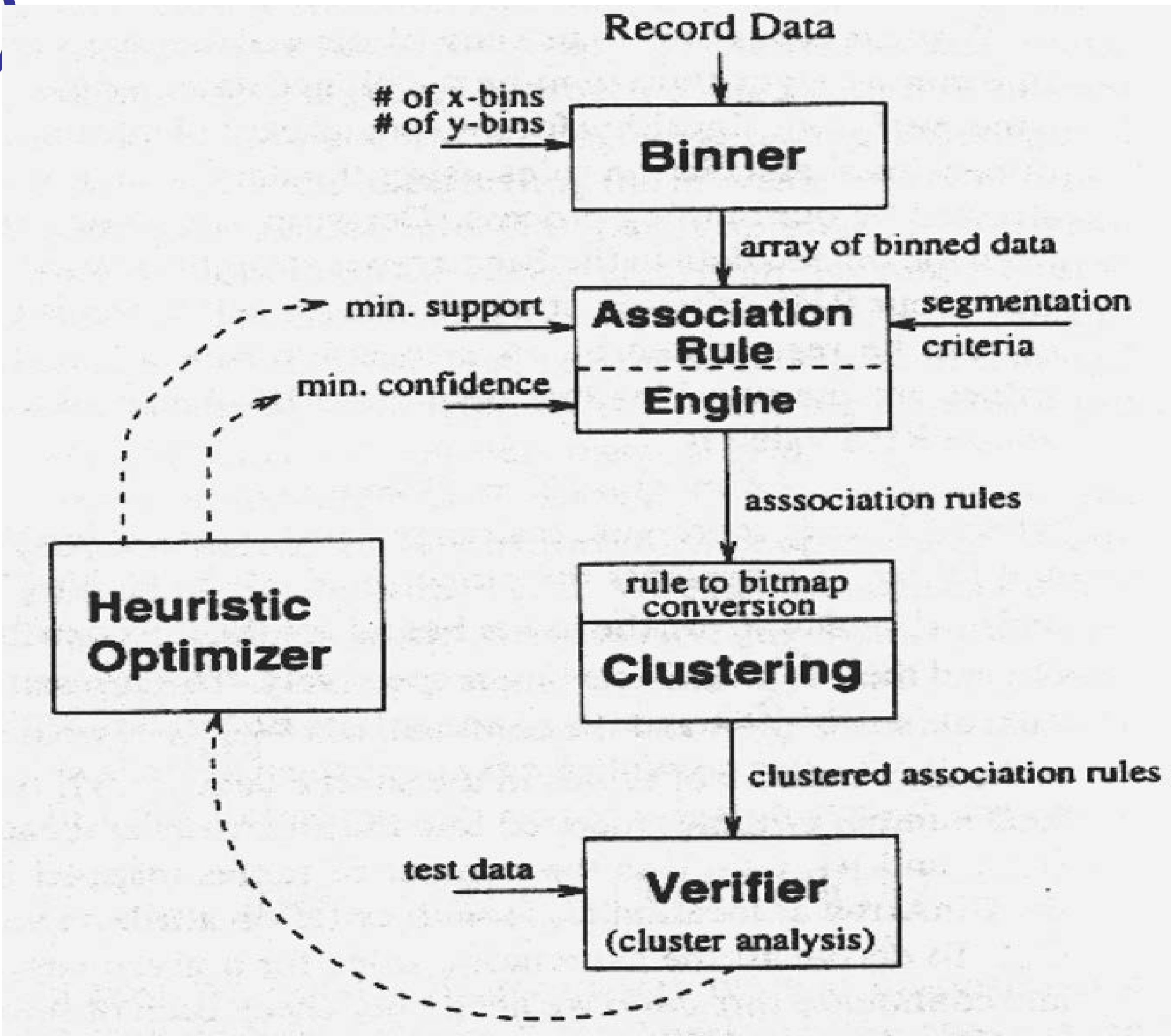
ARCS (Association Rule Clustering System)

1. Binning

2. Find frequent predicateset

3. Clustering

4. Optimize



B. Lent, A. N. Swami and J. Widom, "Clustering Association Rules". ICDE 97.

R. Srikant and R. Agrawal, “Mining Quantitative Association Rules in Large Relational Tables”. ACM SIGMOD96.

Example

Min-Support=40%=2 records
Min-confidence=50%

People

Record ID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	yes	2

Binning

Age	Married
20..24:1 25..29:2 30..34:3 35..39:4	Yes:1 No:2

After Mapping attributes

Record ID	Age	Married	NumCars
100	1	2	1
200	2	1	1
300	2	2	0
400	3	1	2
500	4	1	2

Frequent Itemset (Sample)	Support
{Age:25..29}	2
{Age:30..39}	2
{Married:Yes}	3
{Married:No}	2
{NumCars:1}	2
{NumCars:2}	2
{<Age:30..39>,<Married:Yes>}	2

Rules: Sample

Rule	Support	Confidence
<Age:30..39>and<Married:Yes>=><NumCars:2>	40%	100%
<Age:20..29>=><NumCars:0..1>	60%	100%

Mining Distance-based Association Rules

- Different binning methods

Price($)	Equi-width (width $10)	Equi-depth (depth 2)	Distance-based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

- Distance-based partitioning, more meaningful discretization considering:
  - density/number of points in an interval
  - “closeness” of points in an interval

Clusters and Distance Measurements

S[X] is a set of N tuples $t_1, t_2, \ldots, t_N$, projected on the attribute set X. The diameter of S[X]:

$$d(S[X]) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} dist_x(t_i[X], t_j[X])}{N(N-1)}$$

$dist_x$:distance metric, e.g. Euclidean distance or Manhattan

Clusters and Distance Measurements(Cont.)

The diameter, *d*, assesses the density of a cluster C_X, where

$$d(C_X) \leq d_0^X$$

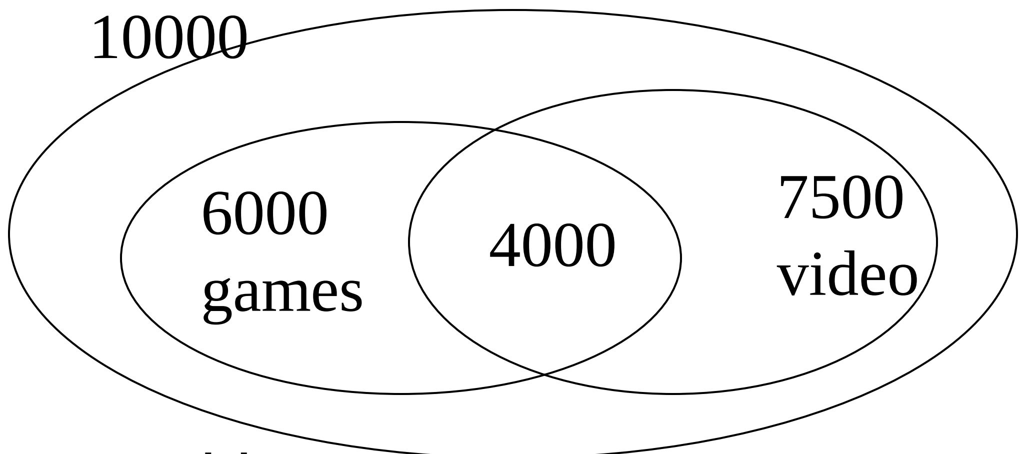
$$|C_X| \geq s_0$$

Finding clusters and distance-based rules

R. J. Miller and Y.Yang,"Association Rules over Interval Data", Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data.

From Association Mining to Correlation Analysis

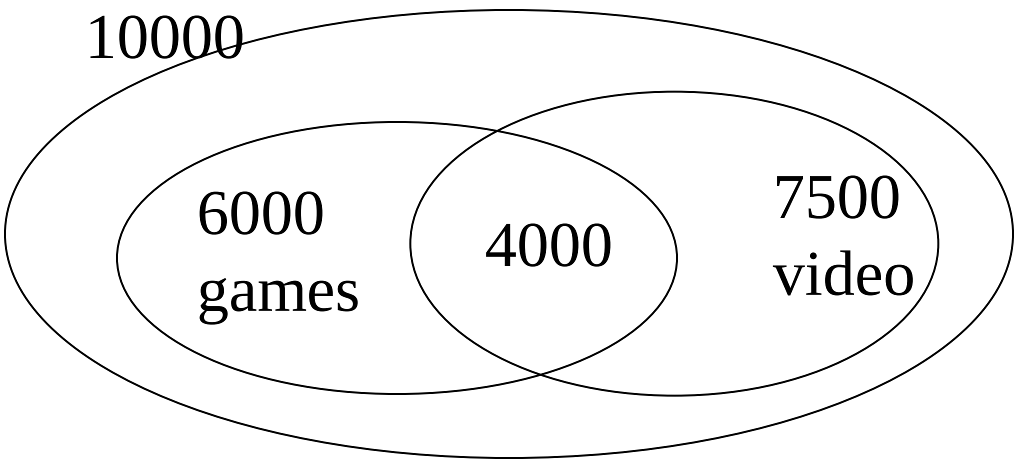
Strong Rules & Interesting



- Games -> Videos,

  support = 4000/10000 = 40%, confidence = 4000/6000 = 66%
- Prob(videos) = 7500/10000 = 75%
- In fact, games & videos are negatively associated
- Purchase of one actually decrease the likelihood of purchasing the other

Correlation Analysis



- Corr(A,B)=P(AUB)/(P(A)P(B))
  - E.g. Corr(games, videos)=0.4/(0.6*0.75)=0.89
  - Corr(A, B)=1, A & B are independent
  - Corr(A, B)<1, occurrence of A is negatively correlated with B
  - Corr(A, B)>1, occurrence of A is positively correlated with B

Mining Association Rules with Weighted Items

- Weighted items
- Weighted support
- Association rule with minimum weighted support
- Given minimum weighted support 0.4

 => {B,E} ((0.3+0.9)*5/7=0.86)

code	Item	Profit	Weight
A	Apple	100	0.1
B	Orange	300	0.3
C	Banana	400	0.4
D	Milk	800	0.8
E	Coca	900	0.9

TID	Items
100	A, B, D, E
200	A, D, E
300	B, D, E
400	A, B, D, E
500	A, C, E
600	B, D, E
700	B, C, D, E

Mining Inter-Transaction Association Rules

- Intra-transaction association rules:

  e.g. When the prices of IBM and SUN go up, at 80% of probability the price of Microsoft goes up on the same day

- Inter-transaction association rules:

  e.g. If the price of IBM and SUN go up, Microsoft's will most likely (80% probability) go up the next day and then drop four days later

Summary

- Frequent pattern mining: mine regularities in databases
- Basic methods
  - Apriori: candidate generation and test
  - Pattern growth: without candidate generation
- Extensions
  - Various patterns: max-patterns, closed patterns
  - ML/MD patterns, quantitative patterns

Association Rules with Constraints

Constrained Association Rules

- Shortcomings of traditional association rules
  - lack of user exploration
  - lack of focus: find associations between itemsets
    - whose types do not overlap
    - total price under $100 to itemsets whose average price is at least $1000

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? --- Making good use of constraints!
- Kinds of constraints used in mining
  - knowledge 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.
  - <u>rule constraints:</u>
    - small sales (price $< \$10$) triggers big sales (sum $> \$200$).
  - Interestingness constraints:
    - strong rules (min_support $\geq 3\%$, min_confidence $\geq 60\%$).

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints
    - meta-rule guided mining.

      e.g. P(x, y) ^ Q(x, w) $\rightarrow$ buys(x, "Education software").
  - Rule (content) constraint
    - constraint-based association query optimization

      e.g. **Sum{S.price < 5}**

Constrained Association Query Optimization Problem

- Given a CAQ *=* { (*S1*, *S2*) / *C* }, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- Proposed approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Property of Constraints

- Property of constrains
  - anti-monotonicity
  - succinctness

Anti-monotonicity

- anti-monotonicity (downward closed)
  - S' is subset of S and S' violates C, so does S.
  - Apriori property is also anti-monotonic
  - Prune unqualified itemsets at each iteration
  - e.g. sum(I.price)<= $100
  - e.g. min(J.price)>=500
  - e.g. avg(I.price)<=100 is not anti-monotonic

Characterization of Anti-Monotonicity Constraints

$S\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	Yes
$v \in S$	no
$S \supseteq V$ $S \subseteq V$ $S = V$	no yes partly
$\min(S) \le v$ $\min(S) \ge v$ $\min(S) = v$	no yes partly
$\max(S) \le v$ $\max(S) \ge v$ $\max(S) = v$	yes no partly
$\text{count}(S) \le v$ $\text{count}(S) \ge v$ $\text{count}(S) = v$	yes no partly
$\text{sum}(S) \le v$ $\text{sum}(S) \ge v$ $\text{sum}(S) = v$	yes no partly
$\text{avg}(S)\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	no
(frequent constraint)	(yes)

Succinctness

- Succinctness:
  - For any set S_1 and S_2 satisfying C, $S_1 \cup S_2$ satisfies C
  - Given A_1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A_1 , i.e., it contains a subset belongs to A_1 ,
- Example :
  - *min(S.Price)* $\leq$ *v* is succinct
  - *sum(S.Price)* $\geq$ *v* is not succinct
    - May have other itemsets not in A1 with their sum being greater than v
- Optimization:
  - If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

Characterization of Constraints by Succinctness

$S\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	Yes
$v \in S$	yes
$S \supseteq V$ $S \subseteq V$ $S = V$	yes yes yes
$\min(S) \le v$ $\min(S) \ge v$ $\min(S) = v$	yes yes yes
$\max(S) \le v$ $\max(S) \ge v$ $\max(S) = v$	yes yes yes
$\mathrm{count}(S) \le v$ $\mathrm{count}(S) \ge v$ $\mathrm{count}(S) = v$	weakly weakly weakly
$\mathrm{sum}(S) \le v$ $\mathrm{sum}(S) \ge v$ $\mathrm{sum}(S) = v$	no no no
$\mathrm{avg}(S)\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	no
(frequent constraint)	(no)

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint

Constraint:

Sum{S.price < 5}

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
~~{5}~~	~~3~~

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price < 5}

The Constrained Apriori Algorithm: Push a Succinct Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

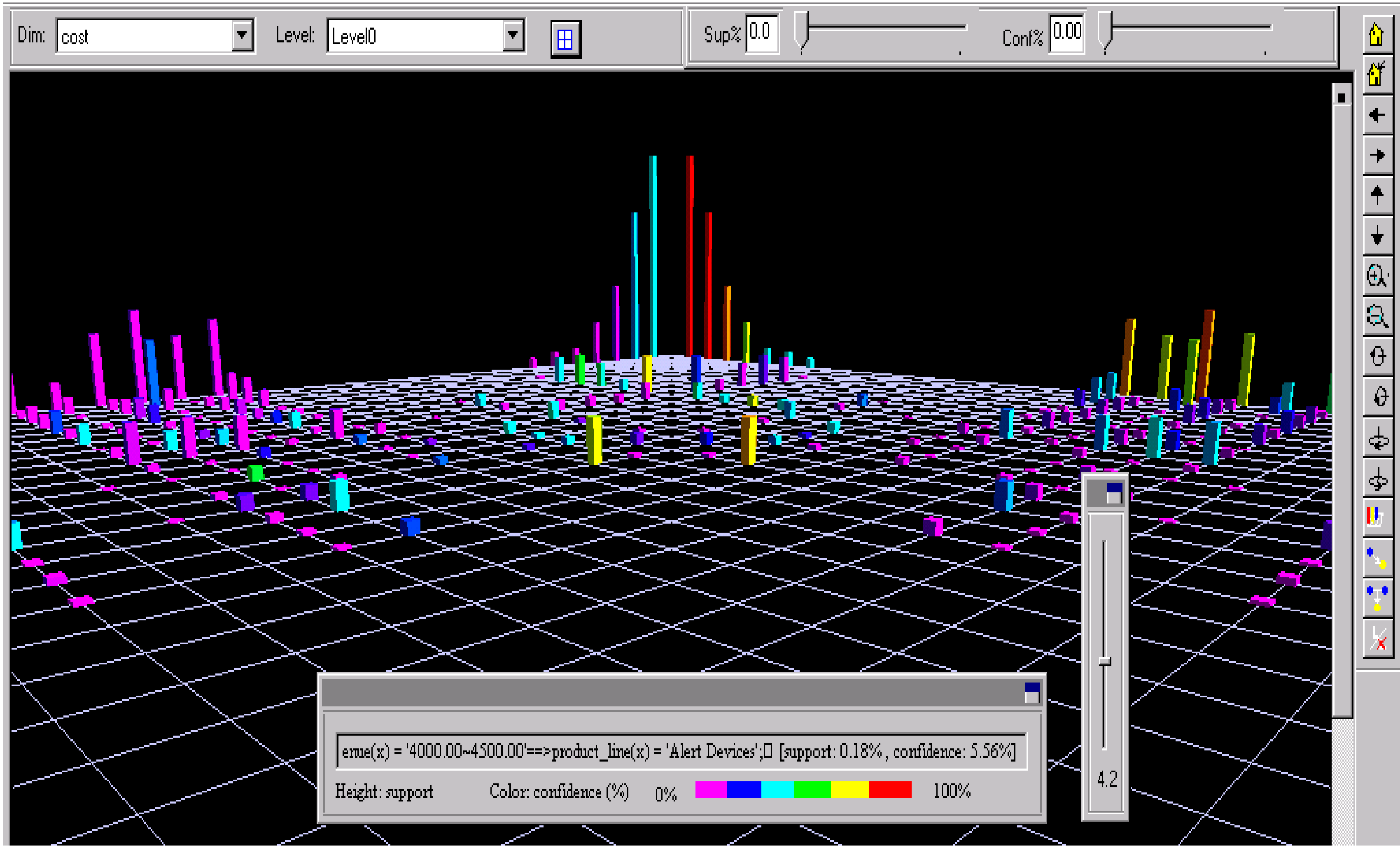
L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price<= 1

Association Rule Visualization



Association Rule Visualization