

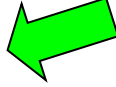
Data Mining:

Concepts and Techniques

(3rd ed.)

— Chapter 6 —

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

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

What Is Frequent Pattern Analysis?

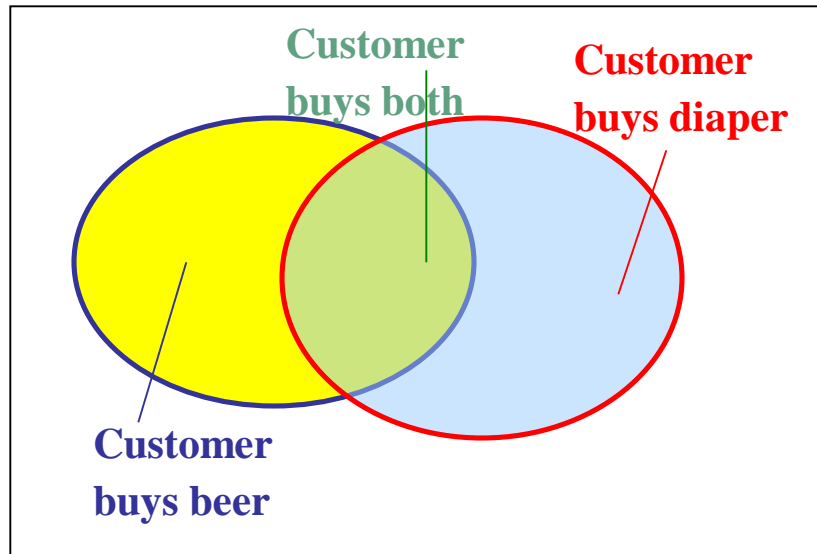
- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?— Beer and diapers?!
 - What are the subsequent purchases after buying a PC?
 - What kinds of DNA are sensitive to this new drug?
 - Can we automatically classify web documents?
- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
 - Association, correlation, and causality analysis
 - Sequential, structural (e.g., sub-graph) patterns
 - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
 - Classification: discriminative, frequent pattern analysis
 - Cluster analysis: frequent pattern-based clustering
 - Data warehousing: iceberg cube and cube-gradient
 - Semantic data compression: fascicles
 - Broad applications

Basic Concepts: Frequent Patterns

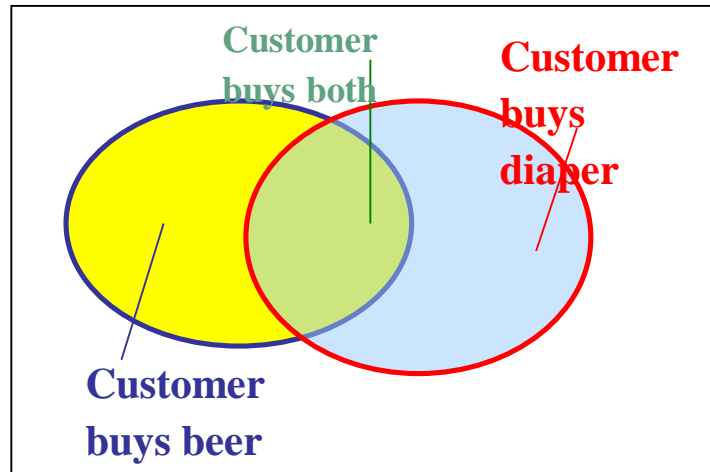
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



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

Basic Concepts: Association Rules

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



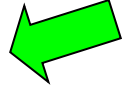
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - **support**, s , **probability** that a transaction contains $X \cup Y$
 - **confidence**, c , **conditional probability** that a transaction having X also contains Y

Let $minsup = 50\%$, $minconf = 50\%$

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

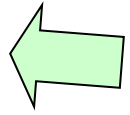
- Association rules: (many more!)
 - $Beer \rightarrow Diaper$ (60%, 100%)
 - $Diaper \rightarrow Beer$ (60%, 75%)

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

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

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



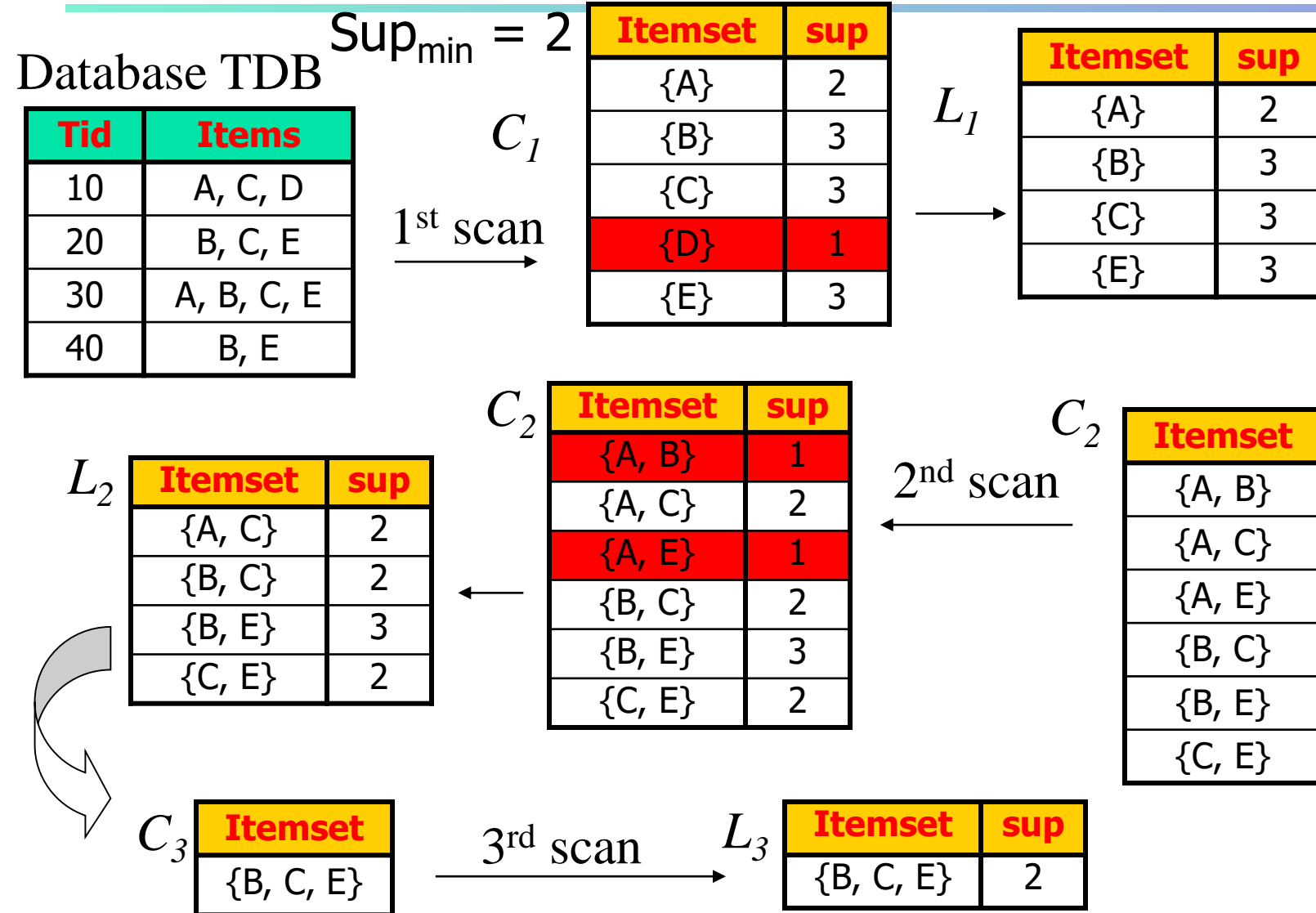
The Downward Closure Property and Scalable Mining Methods

- The **downward closure** property of frequent patterns
 - Any subset of a frequent itemset must be frequent
 - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
 - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
 - Apriori (Agrawal & Srikant@VLDB'94)
 - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
 - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is **any** itemset which is infrequent, its superset should not be generated/tested!
(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
 - Initially, scan DB once to get frequent 1-itemset
 - **Generate** length $(k+1)$ **candidate** itemsets from length k **frequent** itemsets
 - **Test** the candidates against DB
 - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



The Apriori Algorithm (Pseudo-Code)

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \emptyset; k++$) **do begin**

C_{k+1} = candidates generated from L_k ;

for each transaction t in database **do**

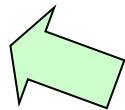
increment the count of all candidates in C_{k+1} that
are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

Scalable Frequent Itemset Mining Methods

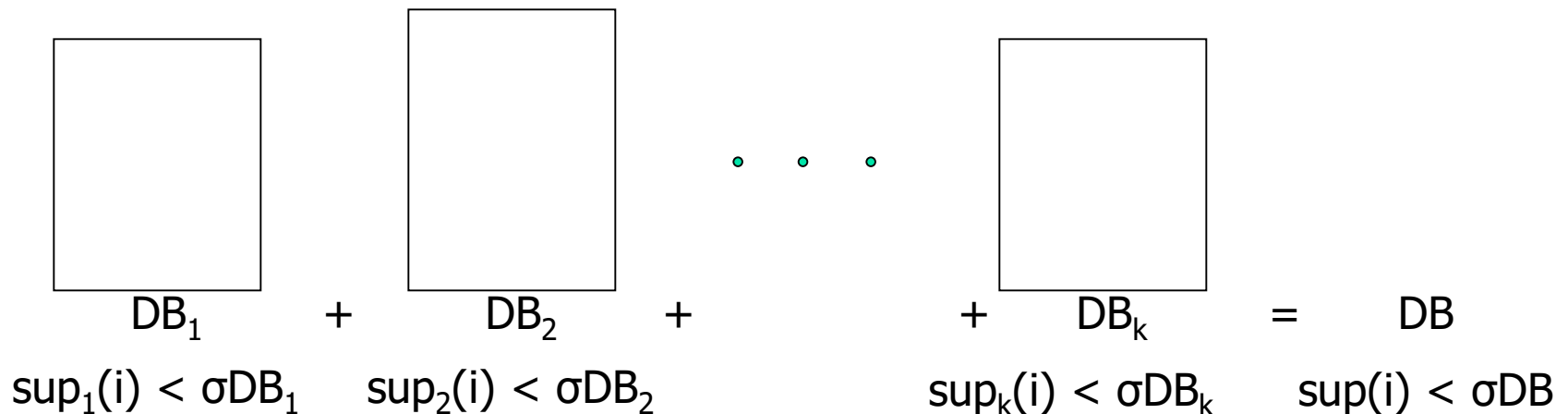
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori 
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Further Improvement of the Apriori Method

- Major computational challenges
 - Multiple scans of transaction database
 - Huge number of candidates
 - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
 - Reduce passes of transaction database scans
 - Shrink number of candidates
 - Facilitate support counting of candidates

Partition: Scan Database Only Twice

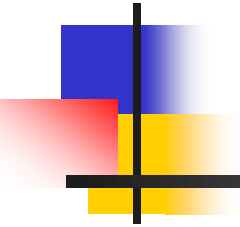
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
 - Scan 1: partition database and find local frequent patterns
 - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



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*

Dynamic Itemset Counting (DIC)



DIC





- Alternative to Apriori Itemset Generation
- Itemsets are dynamically added and deleted as transactions are read
- Relies on the fact that for an itemset to be frequent, all of its subsets must also be frequent, so we only examine those itemsets whose subsets are all frequent

Algorithm stops after every M transactions to add more itemsets.

Train Analogy

- There are stations every M transactions. The passengers are itemsets. Itemsets can get on at any stop as long as they get off at the same stop in the next pass around the database.
- Only itemsets on the train are counted when they occur in transactions. At the very beginning we can start counting 1-itemsets, at the first station we can start counting some of the 2-itemsets. At the second station we can start counting 3-itemsets as well as any more 2-itemsets that can be counted and so on.

Itemset counting

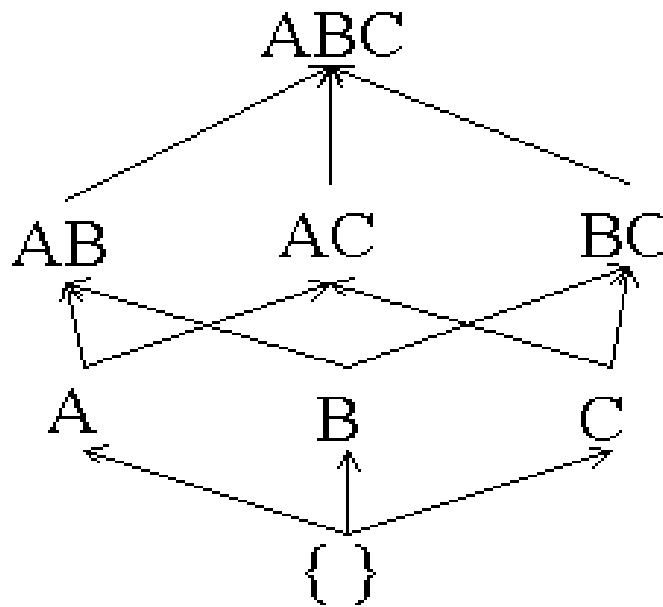
- Solid box:  confirmed frequent itemset - an itemset we have finished counting and exceeds the support threshold *minsupp*
- Solid circle:  confirmed infrequent itemset - we have finished counting and it is below *minsupp*
- Dashed box:  suspected frequent itemset - an itemset we are still counting that exceeds *minsupp*
- Dashed circle:  suspected infrequent itemset - an itemset we are still counting that is below *minsupp*

Example Transaction

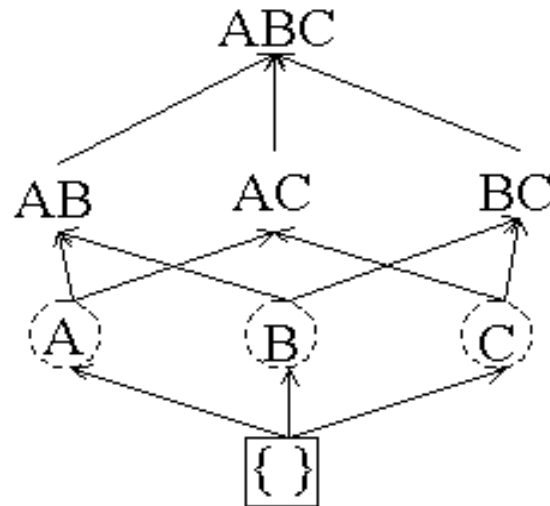
<i>TID</i>	<i>A</i>	<i>B</i>	<i>C</i>
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

minsupp = 25% and M = 2.

Itemset Lattice for the Example Transactions



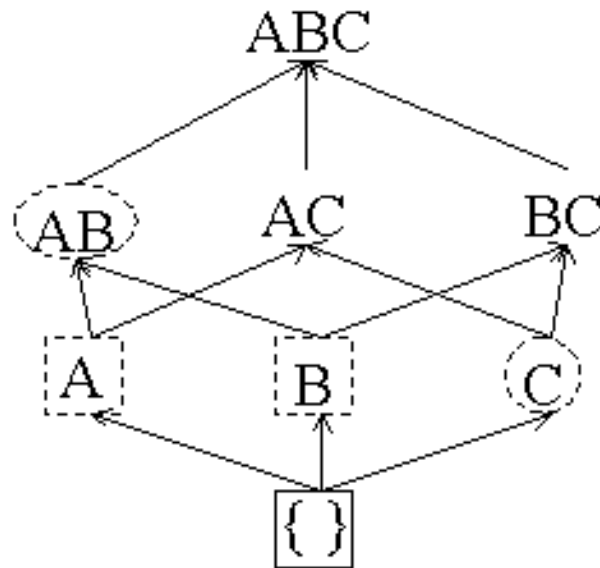
Itemset Lattice before any transactions are read



Counters: $A = 0, B = 0, C = 0$

Empty itemset is marked with a solid box. All 1-itemsets are marked with dashed circles

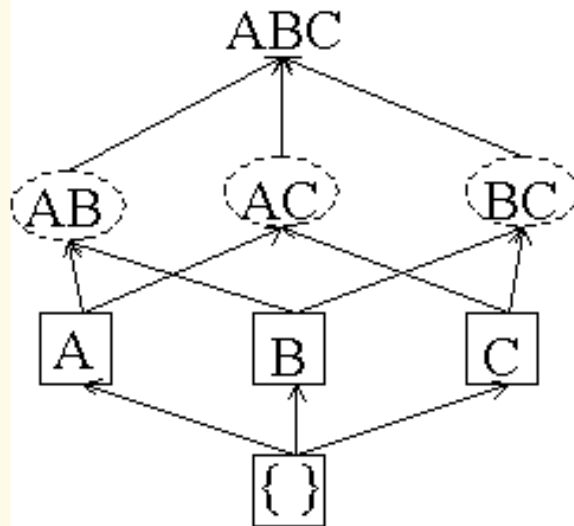
After M Transactions are read



Counters: $A = 2$, $B = 1$, $C = 0$, $AB = 0$

We change A and B to dashed boxes because their counters are greater than minsup (1) and add a counter for AB because both of its subsets are boxes.

After 2M Transactions are read

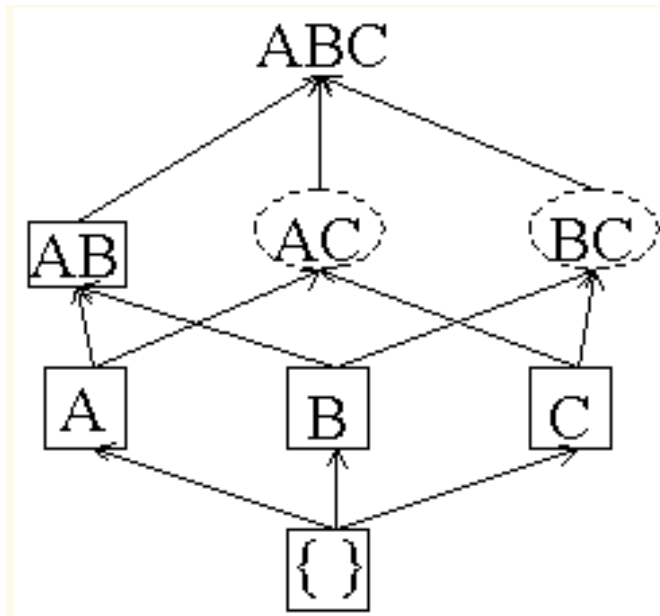


Counters: $A = 2$, $B = 2$, $C = 1$, $AB = 0$,

$AC = 0$, $BC = 0$

C changes to a square because its counter is greater than minsup. A, B and C have been counted all the way through so we stop counting them and make their boxes solid. Add counters for AC and BC because their subsets are all boxes.

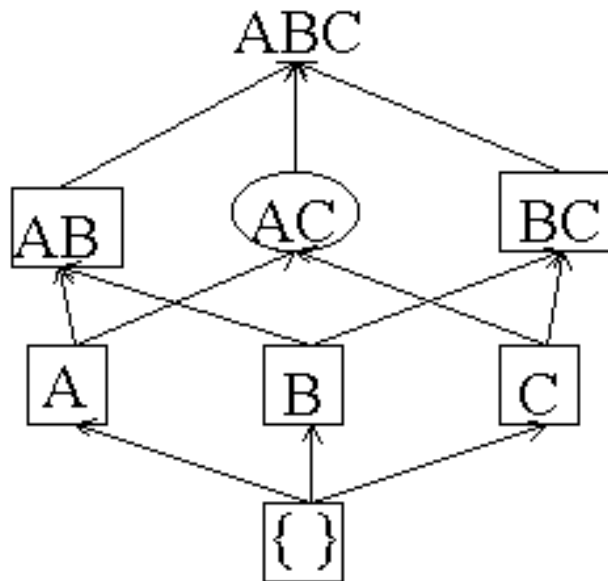
After 3M Transactions are read



Counters: $A = 2$, $B = 2$, $C = 1$, $AB = 1$, $AC = 0$, $BC = 0$

AB has been counted all the way through and its counter satisfies minsup so we change it to a solid box.

After 4M Transactions are read



Counters: $A = 2$, $B = 2$, $C = 1$, $AB = 1$, $AC = 0$, $BC = 1$

AC and BC are counted all the way through. We do not count ABC because one of its subsets is a circle. There are no dashed itemsets left so the algorithm is done.