Università degli Studi di Milano

Master Degree in Computer Science

Information Management course

Teacher: Alberto Ceselli

Lecture 11: 20/11/2012

Data Mining: Concepts and Techniques

(3rd ed.)

— Chapter 6 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University
©2011 Han, Kamber & Pei. All rights reserved.

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?

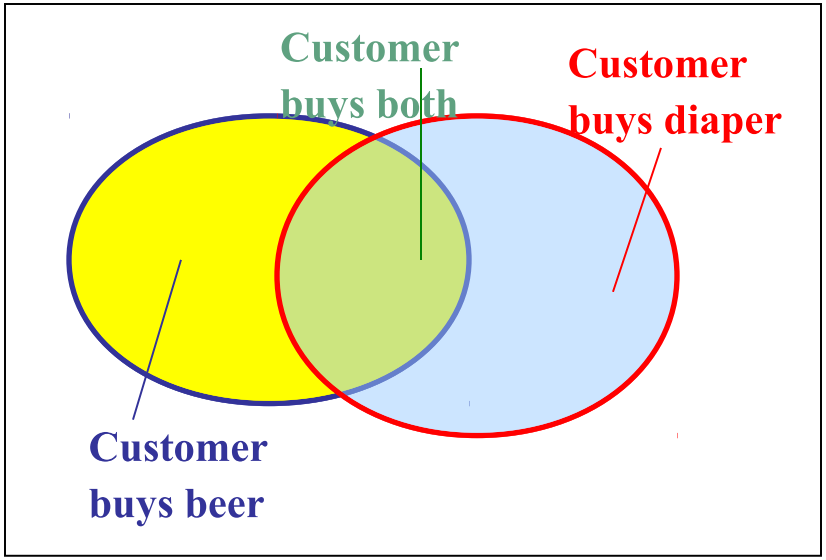
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami (1993) in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What 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 …, 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

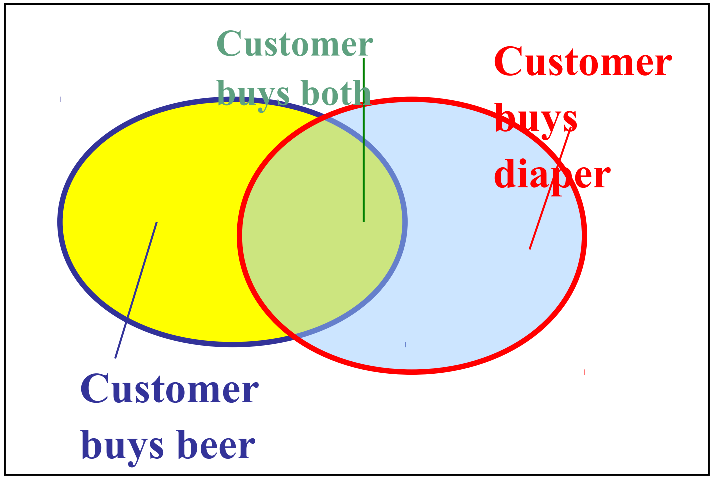
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: number of occurrences of an itemset X in the dataset
- *(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* → *Y* <u>*fixing a*</u> minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ 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* → *Diaper* (60%, 100%)
- *Diaper* → *Beer* (60%, 75%)

Closed Patterns and Max-Patterns

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

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 \cdot 10^{30}$$

  sub-patterns!
- Idea: restrict to *closed and maximal* patterns
  - An itemset X is a closed p. if X is *frequent* and there exists *no super-pattern* Y ⊃ X, *with the same support* as X
  - An itemset X is a maximal p. if X is frequent and there exists no frequent superpattern Y ⊃ X
- Closed pattern is a lossless compression of freq. Patterns: reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise.

  DB = {$<a_1, \ldots, a_{100}>$, $< a_1, \ldots, a_{50}>$}

  - Min_sup = 1.
- What is the set of closed itemset?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of maximal pattern?
  - $<a_1, \ldots, a_{100}>$: 1
- What is the set of all patterns? !!

Computational Complexity of Frequent Itemset Mining

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

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: Candidate Generate&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 is 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 Generate & 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

$Sup_{min} = 2$

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

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items};

for (k = 1; L_k !=$\varnothing$; 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 enough support

end

return $\cup_k L_k$;

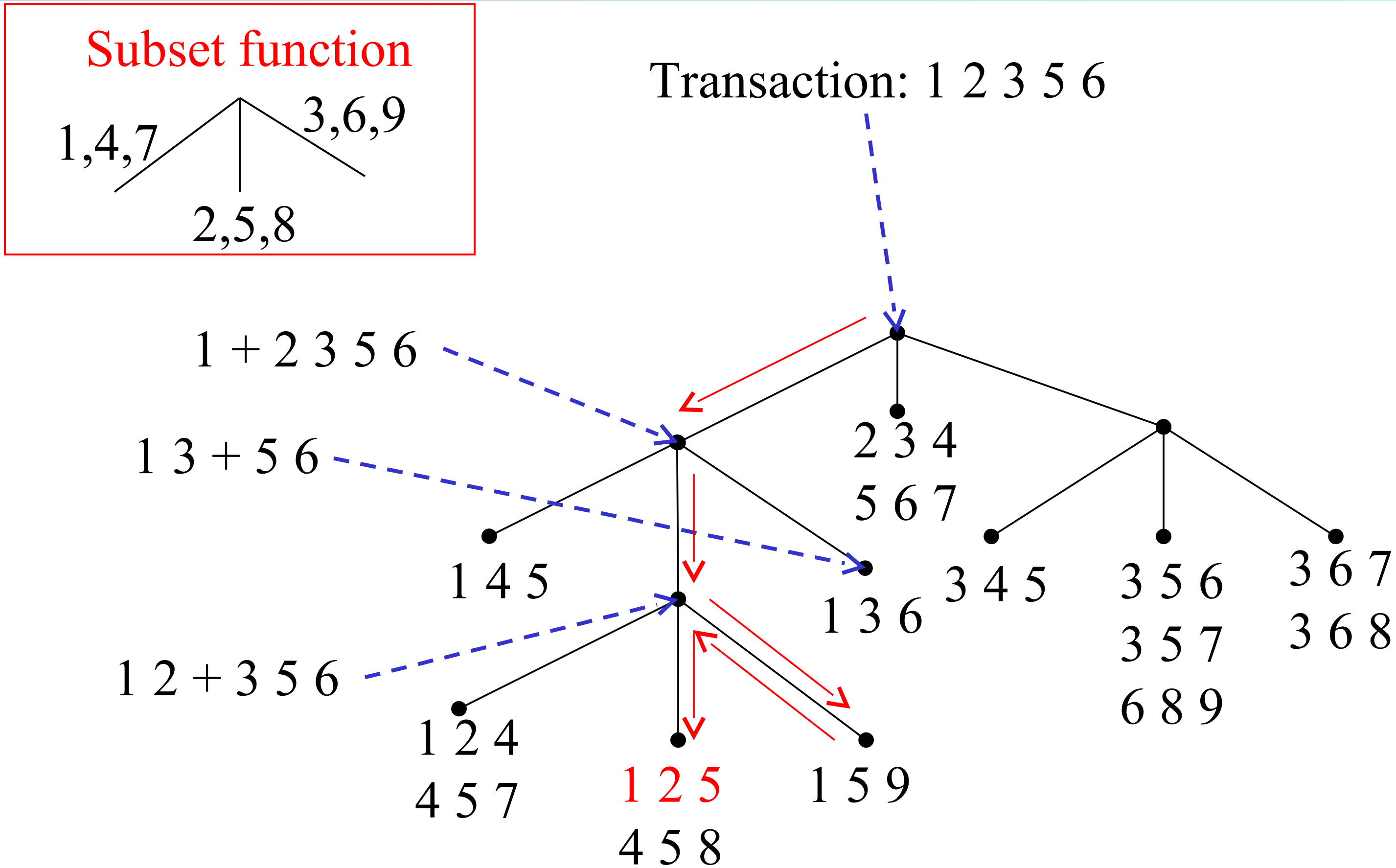
Implementation of Apriori

- 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
  - $C_4 = \{abcd\}$

How to Count Supports of Candidates?

- Why counting supports of candidates is a problem?
  - The total number of candidates can be very huge
  - Each 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
  - *Subset function*: finds all the candidates contained in a transaction

Counting Supports of Candidates Using Hash Tree



Build: store only frequent candidates and their count; do it incrementally while building L_k

Query for a candidate: visit the tree;

Query for an itemset: perform a visit for each sub-itemset;

Generating Association Rules from frequent itemsets

- When all frequent itemsets are found, generate *strong* association rules:
  - Pick each frequent itemset f, generate all its nonempty subsets
  - For each such subset s, test the rule
    - $s \rightarrow (f \setminus s)$
  - support($s \rightarrow (f \setminus s)$) is above the threshold (as f is frequent by construction)
  - confidence ($s \rightarrow (f \setminus s)$) = $P((f \setminus s) \mid s) =$

    count(f) / count(s)
    - count(f) and count(s) are known, and so checking is quick

Candidate Generation: An SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in L_{k-1} are listed in an order
  - Step 1: self-joining L_{k-1}

    insert into $\boldsymbol{C_k}$

    select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

    from $\boldsymbol{L_{k-1}\ p, L_{k-1} q}$

    where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
  - Step 2: pruning

    forall ***itemsets c in*** $\boldsymbol{C_k}$ do

    forall ***(k-1)-subsets s of c*** do

    if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k
- Use object-relational extensions like UDFs, BLOBs, and Table functions for efficient implementation [See: S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98]

Scalable Frequent Itemset Mining Methods

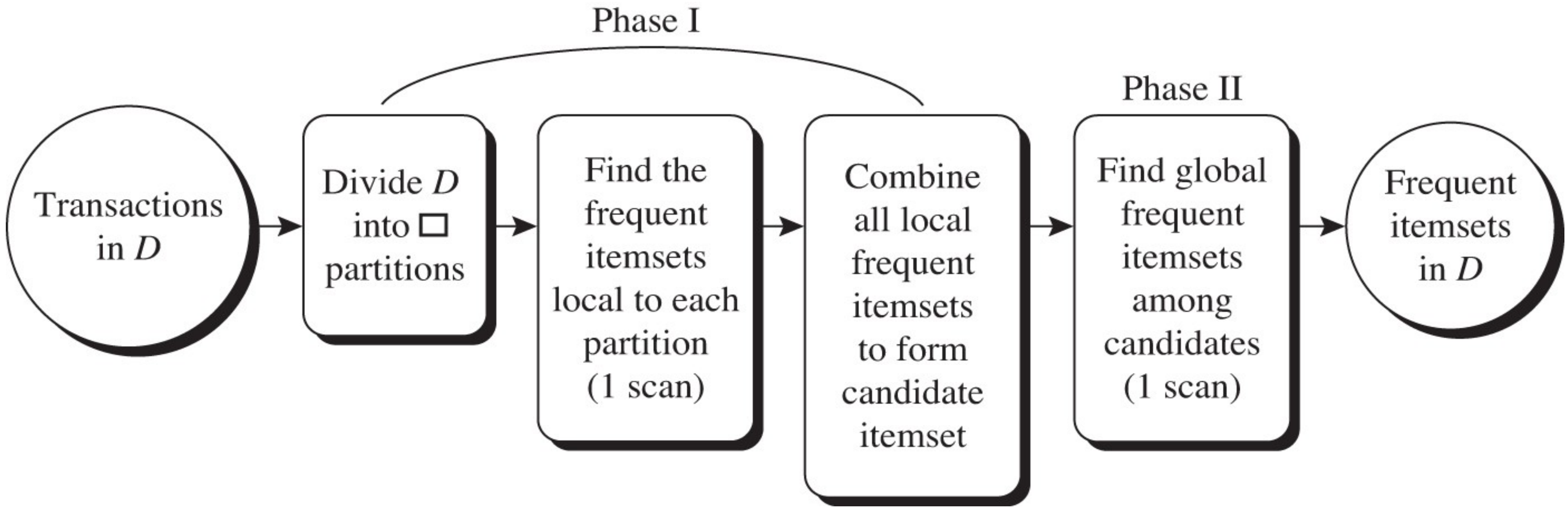
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

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

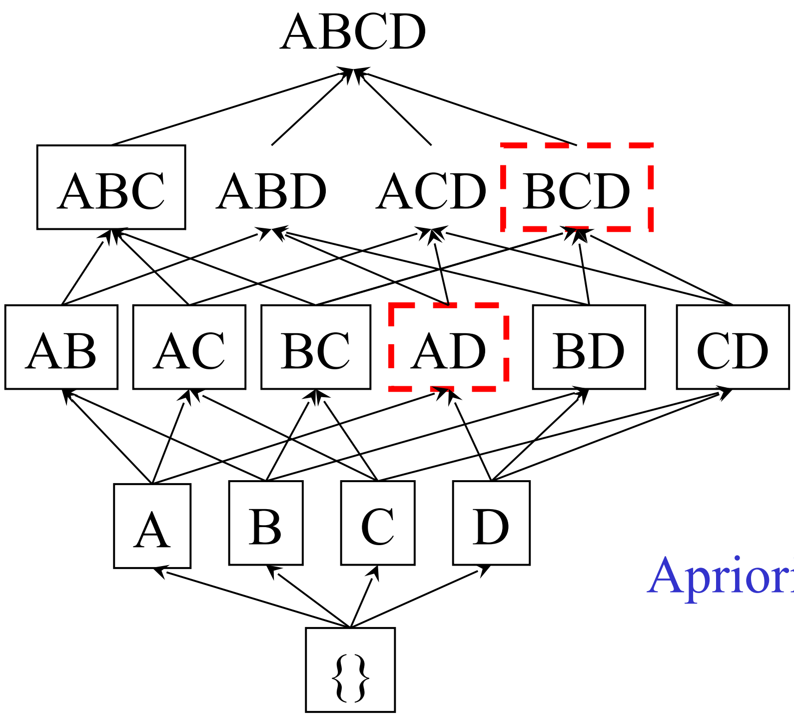
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe *'95*



Sampling for Frequent Patterns

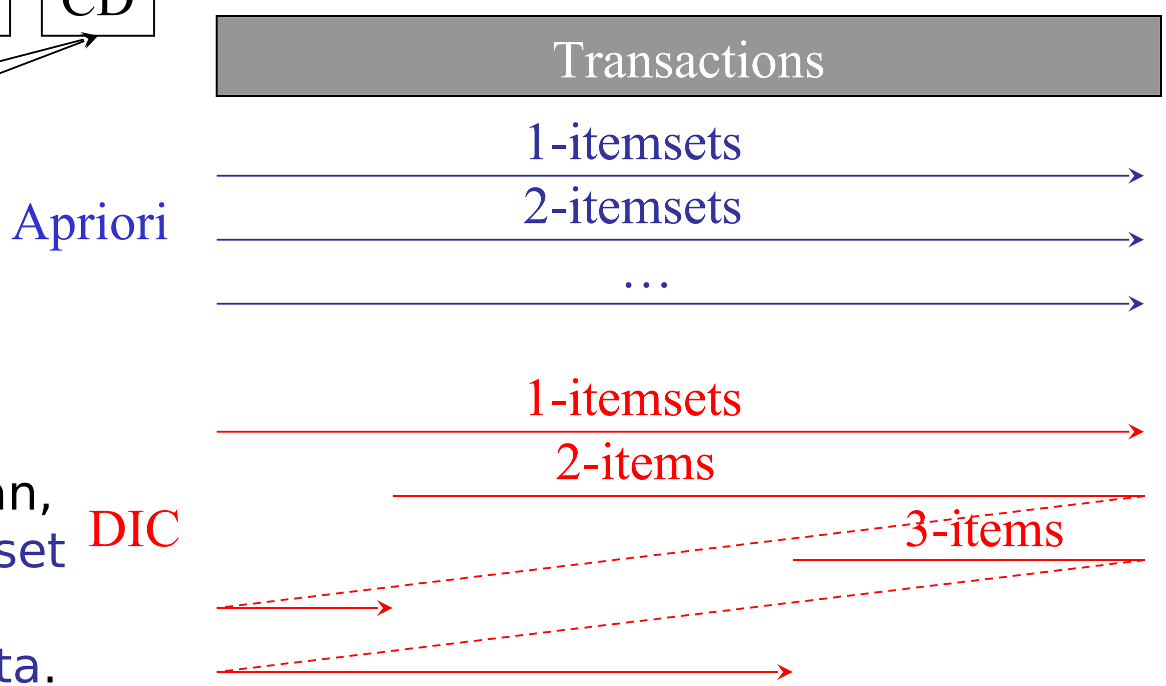
- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find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

Dynamic Itemset Counting: Reduce Number of Scans



Itemset lattice

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *SIGMOD'97*