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} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {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
- **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

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

Definition: Association Rule

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

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:

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

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

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

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} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

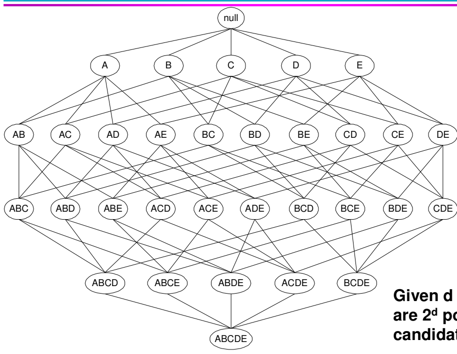
Observations:

- 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
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

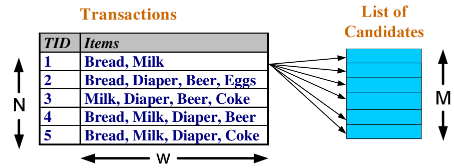
Frequent Itemset Generation



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



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

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

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

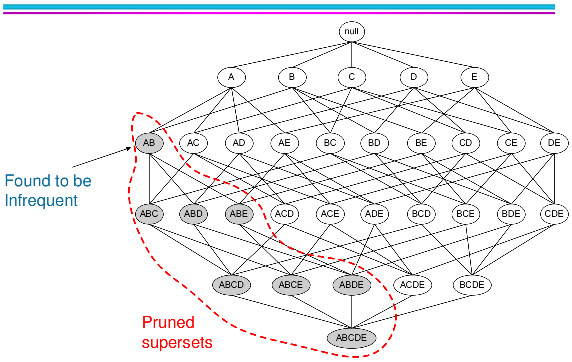
Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets 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
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Pairs (2-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
6 + 6 + 1 = 13

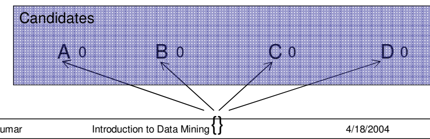
Frequent Itemset Mining

- 2 strategies:
  - Breadth-first: Apriori
    - Exploit monotonicity to the maximum
  - Depth-first strategy: Eclat
    - Prune the database
    - Do not fully exploit monotonicity

Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

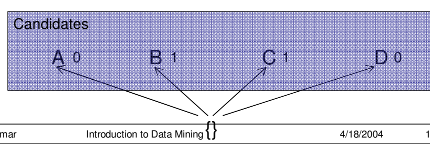
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

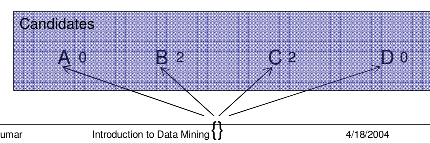
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

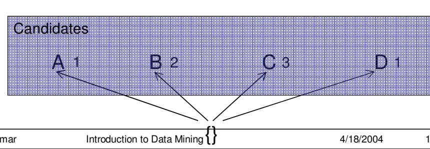
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

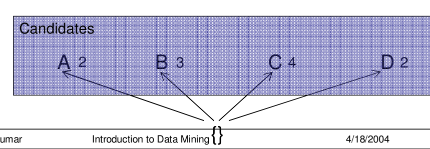
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

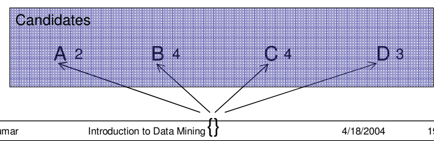
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

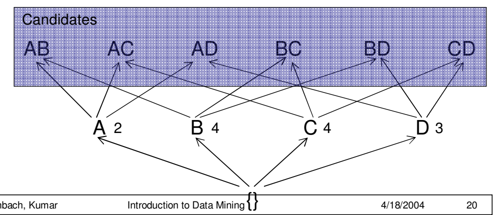
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

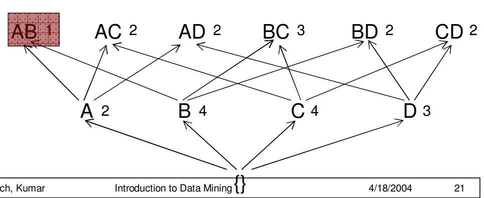
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

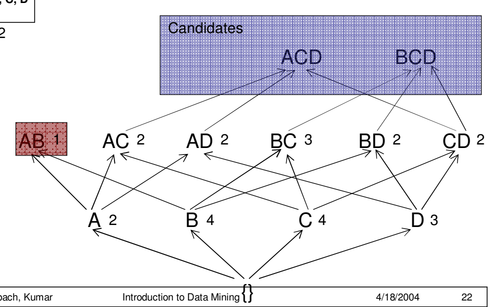
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

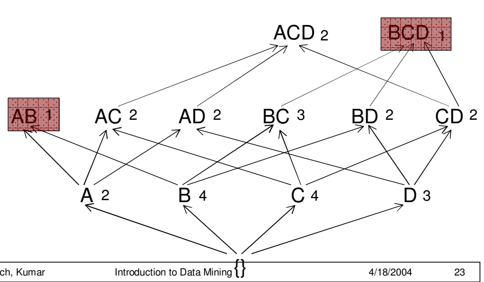
minsup=2



Apriori

1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

minsup=2



Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Frequent Itemset Mining

- 2 strategies:
  - Breadth-first: Apriori
    - Exploit monotonicity to the maximum
  - Depth-first strategy: Eclat
    - Prune the database
    - Do not fully exploit monotonicity

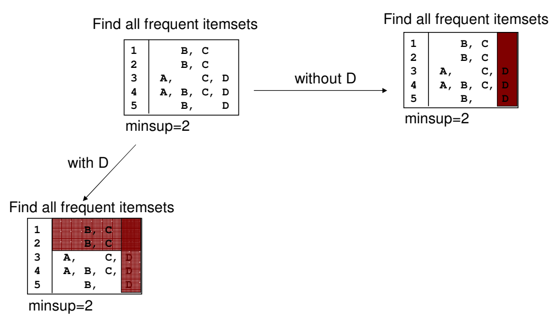
Depth-First Algorithms

Find all frequent itemsets

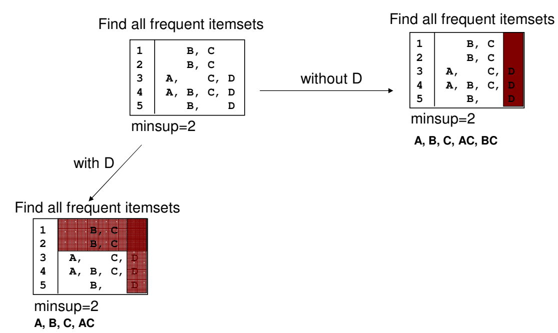
1	B, C
2	B, C
3	A, C, D
4	A, B, C, D
5	B, D

minsup=2

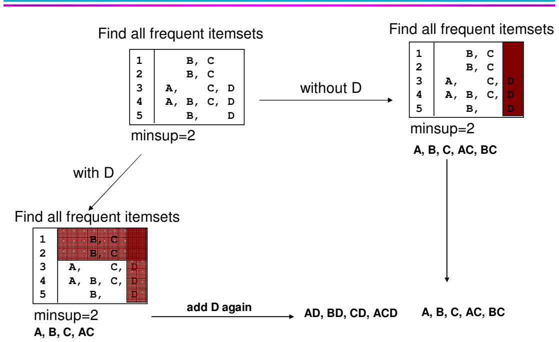
Depth-First Algorithms



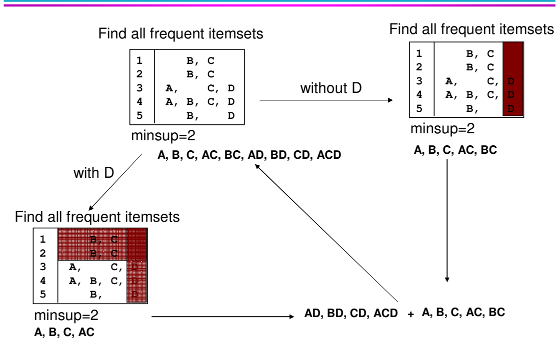
Depth-First Algorithms



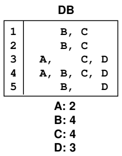
Depth-First Algorithms



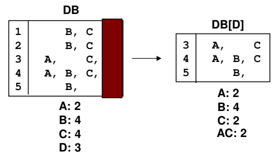
Depth-First Algorithms



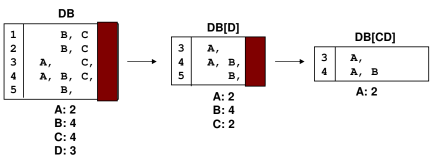
Depth-First Algorithm



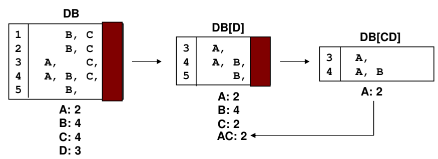
Depth-First Algorithm



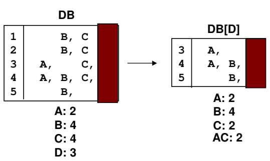
Depth-First Algorithm



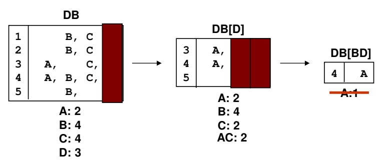
Depth-First Algorithm



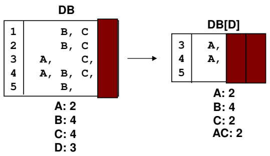
Depth-First Algorithm



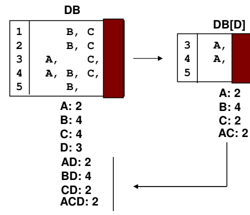
Depth-First Algorithm



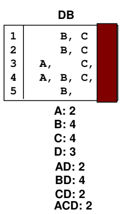
Depth-First Algorithm



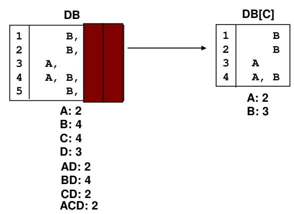
Depth-First Algorithm



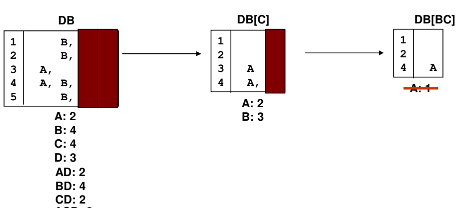
Depth-First Algorithm



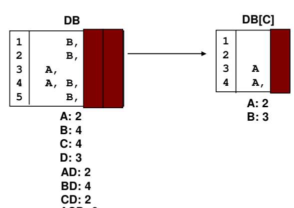
Depth-First Algorithm



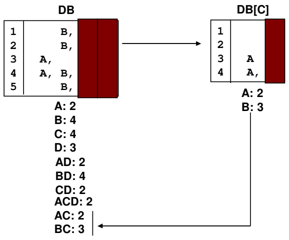
Depth-First Algorithm



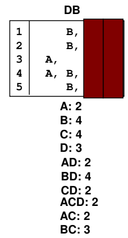
Depth-First Algorithm



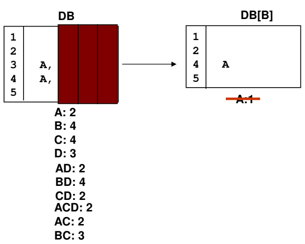
Depth-First Algorithm



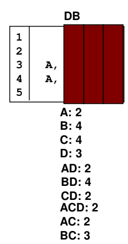
Depth-First Algorithm



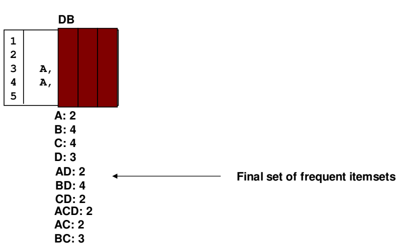
Depth-First Algorithm



Depth-First Algorithm



Depth-First Algorithm



ECLAT

- For each item, store a list of transaction ids (tids)

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				

TID-list

ECLAT

- 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

- Depth-first traversal of the search lattice
- Advantage: very fast support counting
- Disadvantage: intermediate tid-lists may become too large for memory

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

ABC $\rightarrow$D,	ABD $\rightarrow$C,	ACD $\rightarrow$B,	BCD $\rightarrow$A,
A $\rightarrow$BCD,	B $\rightarrow$ACD,	C $\rightarrow$ABD,	D $\rightarrow$ABC
AB $\rightarrow$CD,	AC $\rightarrow$ BD,	AD $\rightarrow$ BC,	BC $\rightarrow$AD,
BD $\rightarrow$AC,	CD $\rightarrow$AB,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

    c(ABC $\rightarrow$D) can be larger or smaller than c(AB $\rightarrow$D)

  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm