Introduction to Data Mining

Frequent Pattern Mining and Association Analysis

Li Xiong

Slide credits: Jiawei Han and Micheline Kamber
George Kollios

Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- Mining association rules
- Association mining to correlation analysis
- Constraint-based association mining

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - Frequent sequential pattern
  - Frequent structured pattern
- 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?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Frequent Itemset Mining

- Frequent itemset mining: frequent set of items in a transaction data set
- Agrawal, Imielinski, and Swami, SIGMOD 1993
  - SIGMOD Test of Time Award 2003

"This paper started a field of research. In addition to containing an innovative algorithm, its subject matter brought data mining to the attention of the database community … even led several years ago to an IBM commercial, featuring supermodels, that touted the importance of work such as that contained in this paper. "

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In SIGMOD '93.

Apriori: Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In VLDB '94.

Basic Concepts: Transaction dataset

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

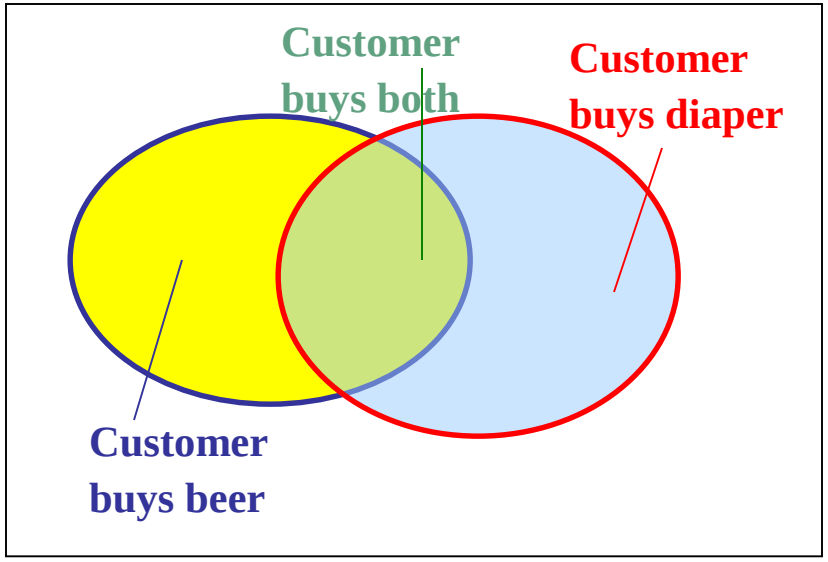
Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Itemset: $X = \{x_1, \ldots, x_k\}$ (k-itemset)
- Frequent itemset: X with minimum support count
  - Support count (absolute support): count of transactions containing X

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset: $X = \{x_1, \ldots, x_k\}$ (k-itemset)
- Frequent itemset: X with minimum support count
  - Support count (absolute support): count of transactions containing X
- Association rule: $A \rightarrow B$ with minimum support and confidence
  - Support: probability that a transaction contains $A \cup B$

    $s = P(A \cup B)$
  - Confidence: conditional probability that a transaction having A also contains B

    $c = P(B \mid A)$

Illustration of Frequent Itemsets and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Frequent itemsets (minimum support count = 3) ?
- Association rules (minimum support = 50%, minimum confidence = 50%) ?

Illustration of Frequent Itemsets and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Frequent itemsets (minimum support count = 3) ?

  {A:3, B:3, D:4, E:3, AD:3}

- Association rules (minimum support = 50%, minimum confidence = 50%) ?

  A → D (60%, 100%)

  D → A (60%, 75%)

Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- Mining association rules
- Association mining to correlation analysis
- Constraint-based association mining

Scalable Methods for Mining Frequent Patterns

- Frequent itemset mining methods
  - Apriori
  - Fpgrowth
- Closed and maximal patterns and their mining methods

Frequent itemset mining

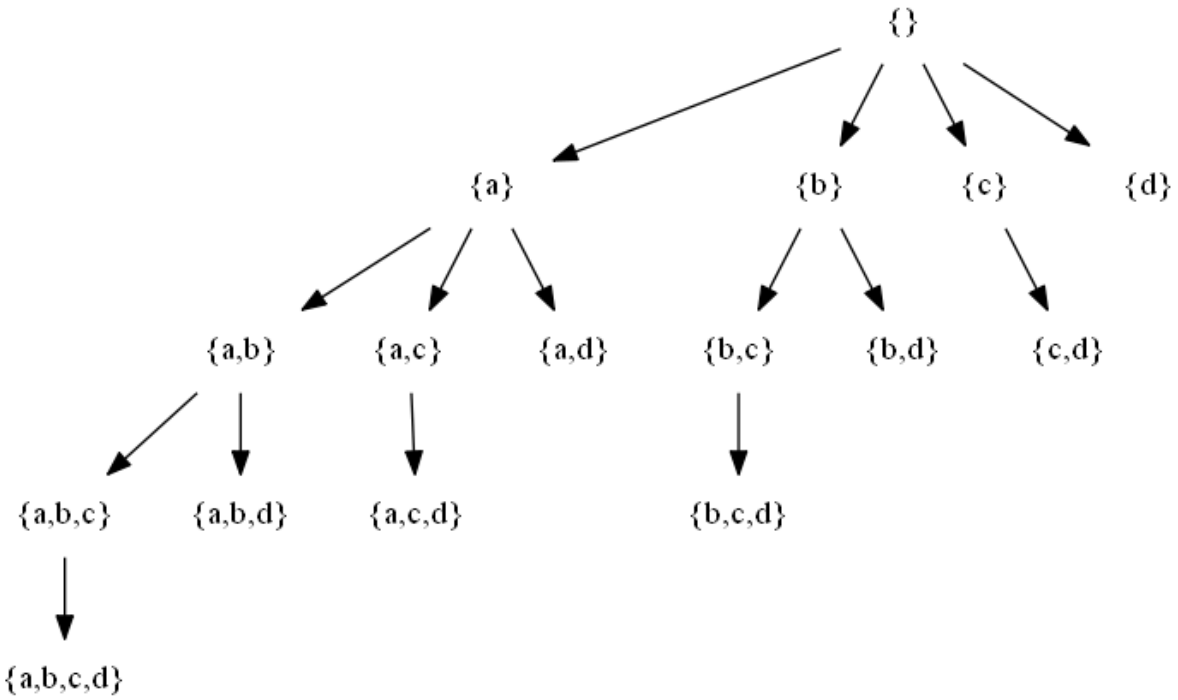
- Brute force approach

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Frequent itemset mining

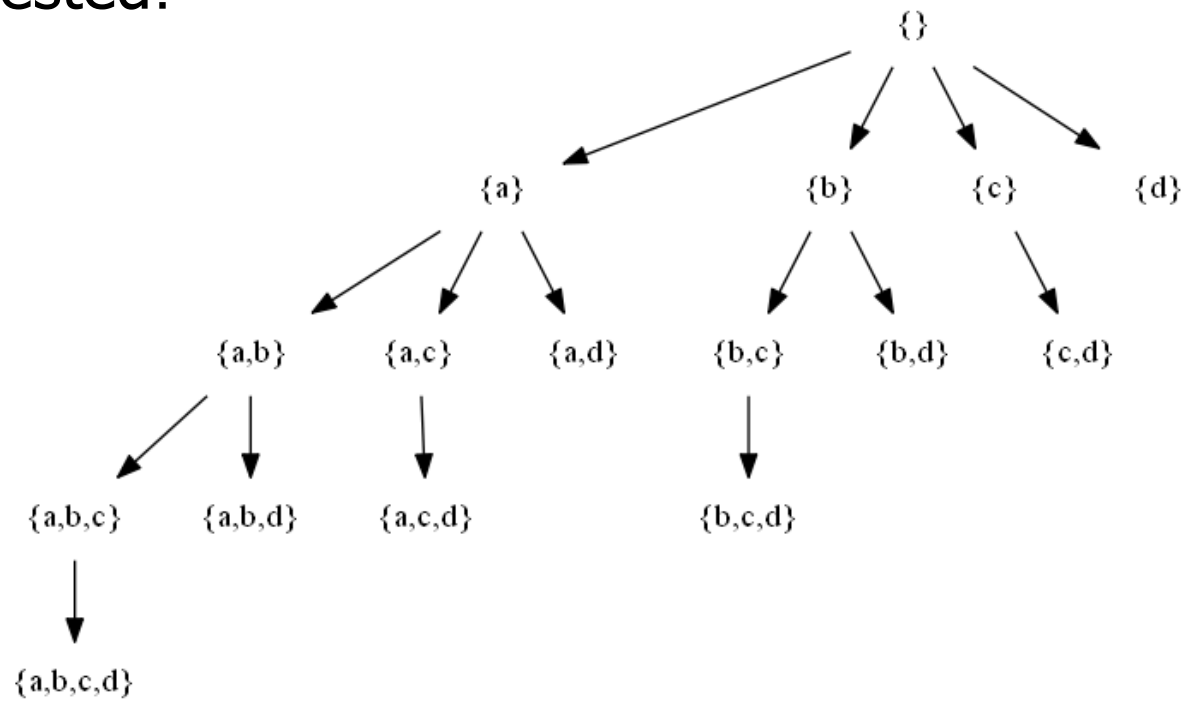
- Brute force approach
- Set enumeration tree for all possible itemsets
- Tree search
  - Apriori – BFS, FPGrowth - DFS

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



Apriori

- BFS based
- Apriori pruning principle: if there is any itemset which is infrequent, its superset must be infrequent and should not be generated/tested!



Apriori: Level-Wise Search Method

- Level-wise search method (BFS):
  - 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

- Pseudo-code:

  C_k: Candidate k-itemset
  L_k : frequent k-itemset

  L_1 = frequent 1-itemsets;
  for (k = 2; L_{k-1} !=∅; k++)
  C_k = generate candidate set from L_{k-1};
  for each transaction t in database
  find all candidates in C_k that are subset of t;
  increment their count;
  L_k = candidates in C_k with min_support
  return $\cup_k L_k$;

The Apriori Algorithm—An Example

Sup_{min} = 2

Transaction DB

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 → 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

Details of Apriori

- How to generate candidate sets?
- How to count supports for candidate sets?

Candidate Set Generation

C_k = generate candidate set from L_{k-1};

- Step 1: self-joining L_{k-1}: assuming items and itemsets are sorted in order, joinable only if the first k-2 items are in common

$$\begin{aligned}
&\textbf{insert into } C_k \\
&\textbf{select } p.\text{item}_1,\ p.\text{item}_2,\ \ldots,\ p.\text{item}_{k-1},\ q.\text{item}_{k-1} \\
&\textbf{from } L_{k-1}\ p,\ L_{k-1}\ q \\
&\textbf{where } p.\text{item}_1 = q.\text{item}_1,\ \ldots,\ p.\text{item}_{k-2} = q.\text{item}_{k-2}, \\
&\qquad p.\text{item}_{k-1} < q.\text{item}_{k-1};
\end{aligned}$$

- Step 2: pruning: prune if it has infrequent subset

Example: Generate C_4 from L_3={*abc, abd, acd, ace, bcd*}

- Step 1: Self-joining: $L_3 * L_3$
  - *abcd* from *abc* and *abd; acde* from *acd* and *ace*
- Step 2: Pruning:
  - *acde* is removed because *ade* is not in L_3

C_4={*abcd*}

How to Count Supports of Candidates?

for each transaction *t* in database
find all candidates in C_k that are subset of *t*;
increment their count;

- For each subset s in t, check if s is in C_k
  - The total number of candidates can be very large
  - One transaction may contain many candidates

How to Count Supports of Candidates?

for each transaction *t* in database
find all candidates in C_k that are subset of *t*;
increment their count;

- For each subset s in t, check if s is in C_k
  - Linear search
  - Hash-tree (prefix tree with hash function at interior node) – used in original paper
  - Hash-table - recommended

DHP: Reducing number of candidates

Database D

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

On the fly

C_1	count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

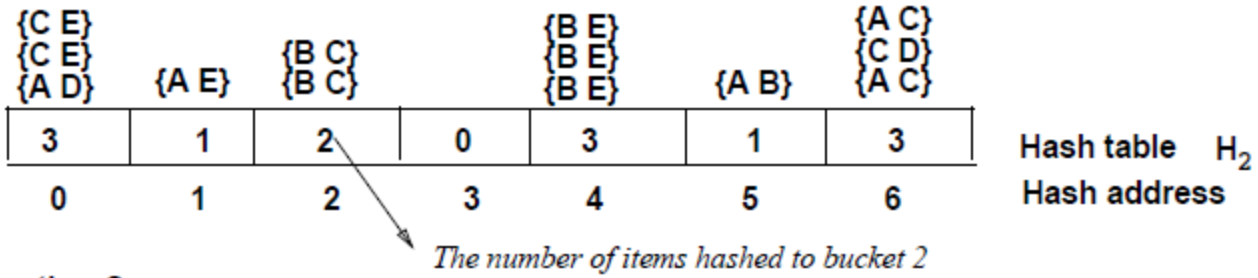
→ L_1: {A}, {B}, {C}, {E}

minimum support, s = 2

Making a hash table

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

h{{x y}) = ((order of x)*10 + (order of y)) mod 7;



Generating C_2

$L_1 \times L_1$

# in a bucket with the itemset	
{A B}	1
{A C}	3
{A E}	1
{B C}	2
{B E}	3
{C E}	3

→ C_2: {A C}, {B C}, {B E}, {C E}

Assignment 1

- Implementation and evaluation of Apriori
- Performance competition!

Improving Efficiency of Apriori

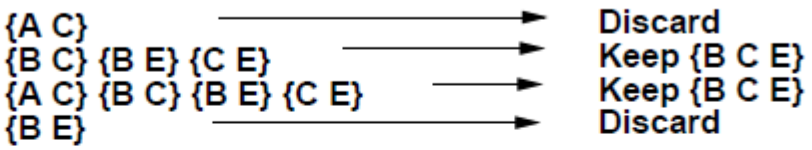
- Bottlenecks
  - Huge number of candidates
  - Multiple scans of transaction database
  - Support counting for candidates
- Improving Apriori: general ideas
  - Shrink number of candidates
  - Reduce passes of transaction database scans
  - Reduce number of transactions

Reducing size and number of transactions

- Discard infrequent items
  - If an item is not frequent, it won't appear in any frequent itemsets
  - If an item does not occur in at least k frequent k-itemset, it won't appear in any frequent k+1-itemset
    - Implementation: if it does not occur in at least k candidate k-itemset, discard
- Discard a transaction if all items are discarded

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

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



D_3 = {<200, B C E>, <300, B C E>}

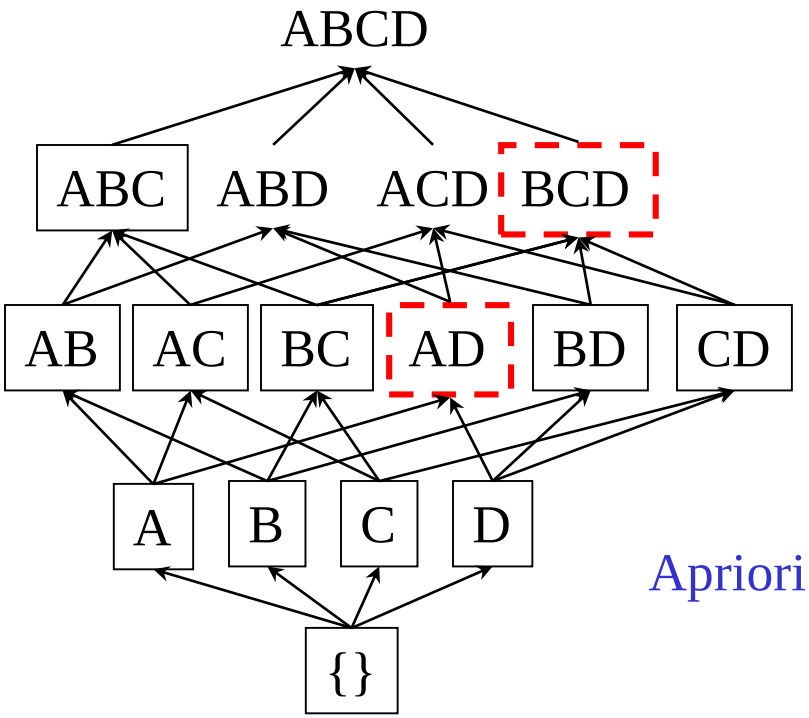
C_2	count
{A C}	2
{B C}	2
{B E}	3
{C E}	2

→

L_2
{A C}
{B C}
{B E}
{C E}

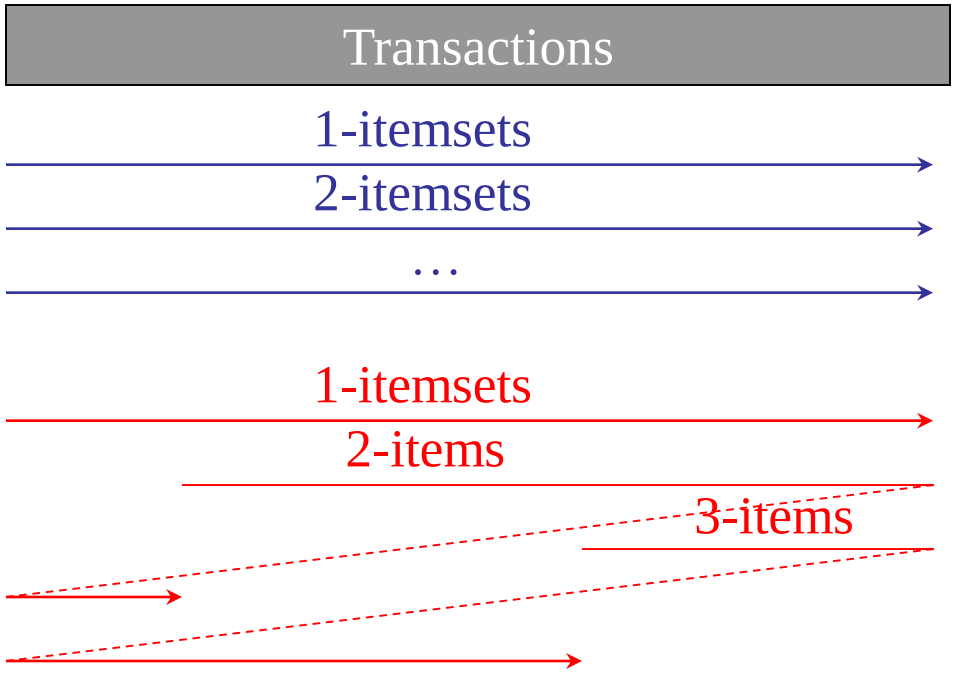
s = 2

DIC: Reduce Number of Scans



Itemset lattice

- DIC (Dynamic itemset counting): partition DB into blocks, add new candidate itemsets at partition points
  - 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



Apriori

DIC

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

Partitioning: Reduce Number of Scans

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database in *n* partitions and find local frequent patterns (minimum support count?)
  - Scan 2: determine global frequent patterns from the collection of all local frequent patterns

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

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within samples using Apriori
- Scan database once to verify frequent itemsets found in sample
- Use a lower support threshold than minimum support
- Tradeoff accuracy against efficiency

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

Scalable Methods for Mining Frequent Patterns

- Frequent itemset mining methods
  - Apriori
  - **FPgrowth**
- Closed and maximal patterns and their mining methods

Mining Frequent Patterns Without Candidate Generation

- Apriori: Breadth first search in set enumeration tree
- FP-Growth: Depth first search in set enumeration tree
- Basic idea: Find (grow) long patterns from short ones recursively
  - "abc" is a frequent pattern
  - All transactions having "abc": DB|abc (conditional DB)
  - "d" is a local frequent item in DB|abc, then abcd is a frequent pattern
- Details:
  - Data structure to find conditional DB - FP-tree (trie)
  - Sort items in the set-enumeration (pattern) tree

Construct FP-tree from a Transaction Database

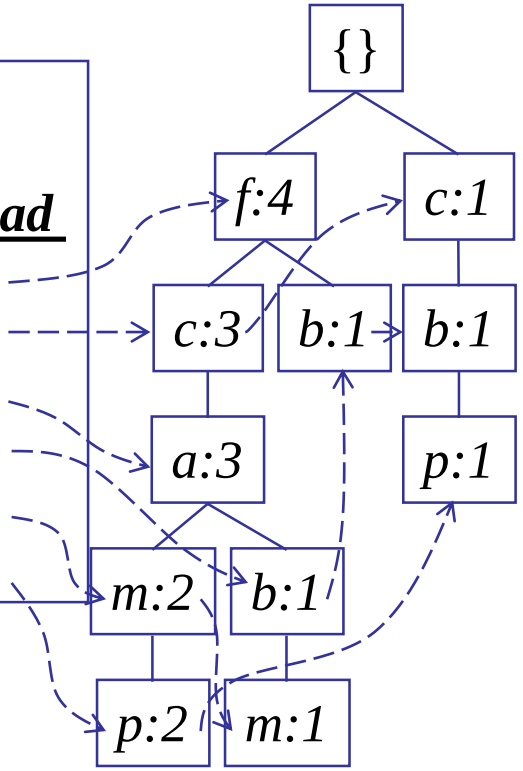
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree (prefix tree)

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

Possible Patterns – set enumeration tree

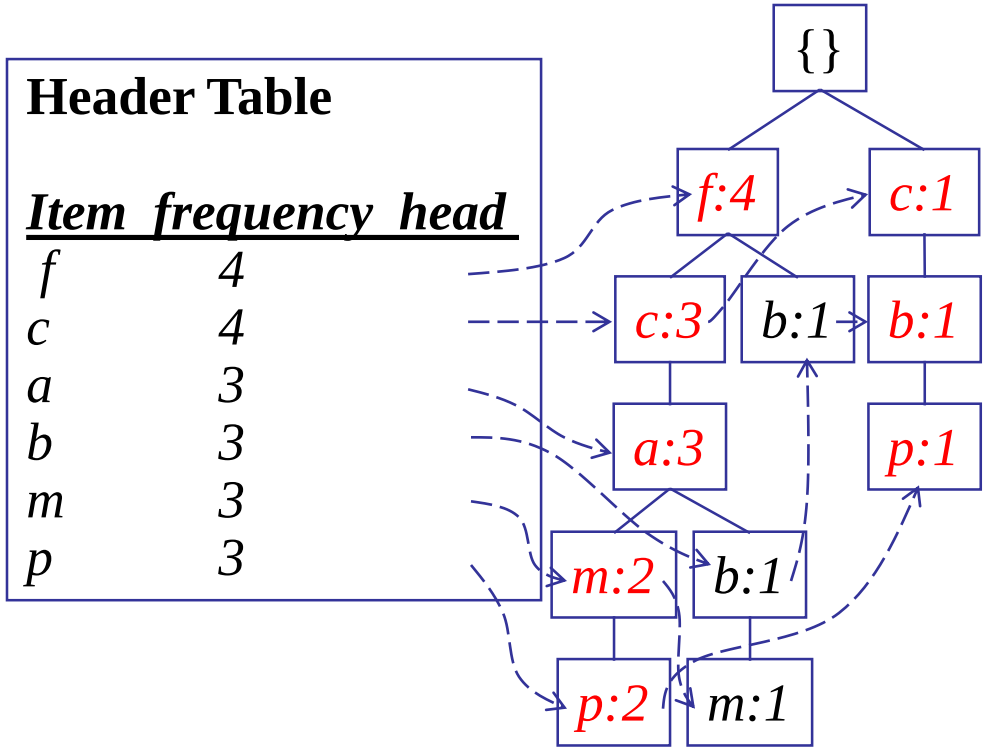
- Frequent patterns can be partitioned into subsets according to f-list: f-c-a-b-m-p (assuming items are sorted) – essentially set enumeration tree
  - Patterns containing p (patterns ending with p)
  - Patterns having m but no p (patterns ending with m)
  - …
  - Patterns having c but no a nor b, m, p (patterns ending with c)
  - Pattern having f but no c, a, b, m, p (patterns ending with f)
- Completeness and non-redundancy
- Ordering of the items: from least frequent items to frequent items, offers better selectivity and pruning

Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item (least frequent first), construct its conditional DB, and then its conditional FP-tree (with only frequent items)
  - Repeat the process recursively on the new 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

Find Patterns Ending with P

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional DB (conditional pattern base)



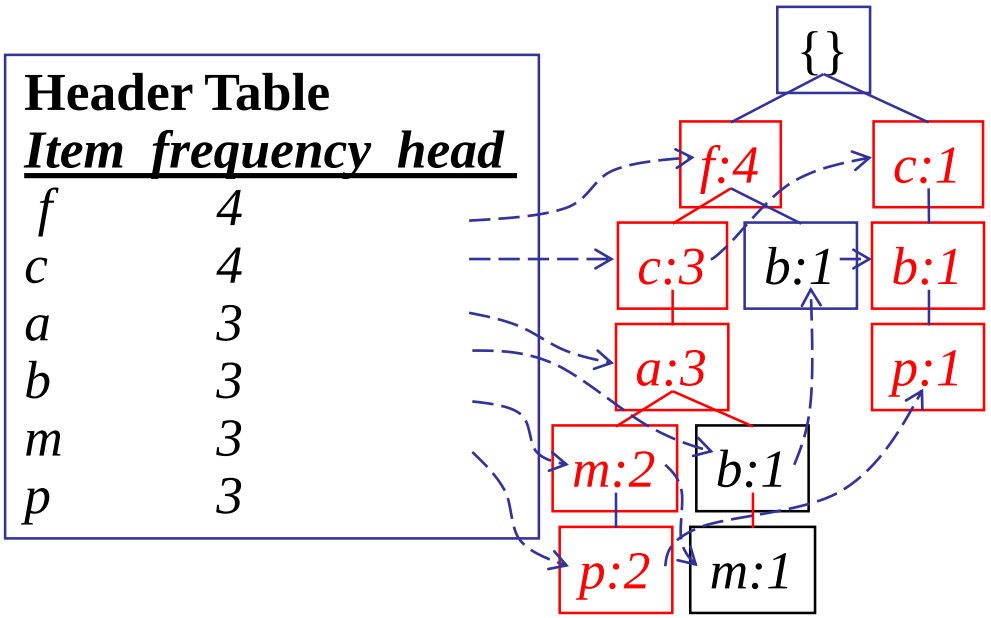
***min_support* = 3**

***Conditional* pattern base**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

From Conditional DB to Conditional FP-trees

- Accumulate the count for each item in the base
- Construct the FP-tree for the frequent items of the conditional DB
- Repeat the process recursively on the new conditional FP-tree until the resulting FP-tree is empty, or only one path



***p-conditional* DB:**

fcam:2, cb:1

p-conditional FP-tree (min-support =3)

{}

|

c:3

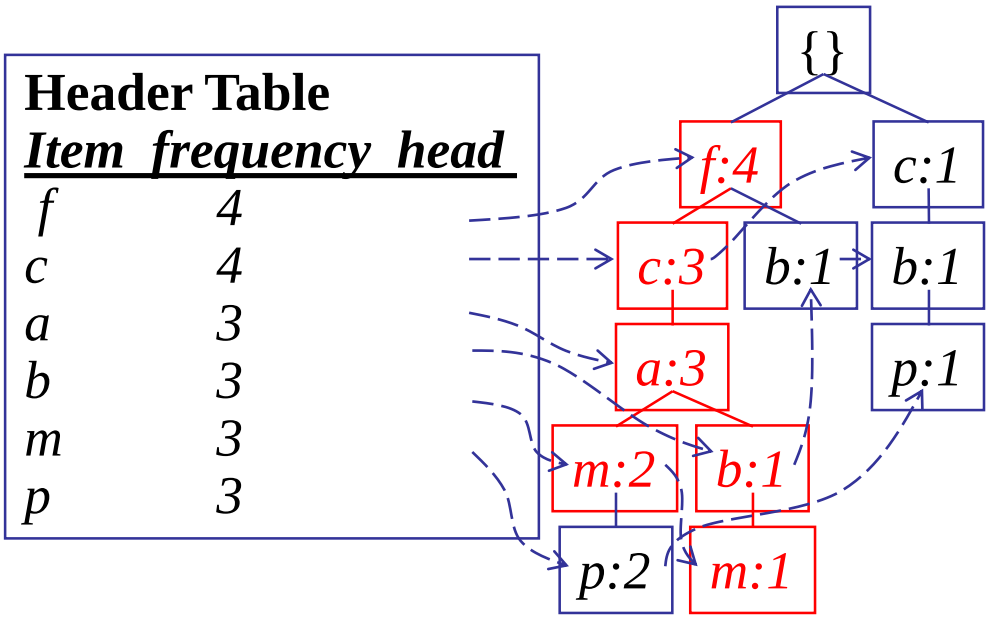
All frequent patterns containing p

p,

cp

Finding Patterns Ending with m

- Construct m-conditional DB, then its conditional FP-tree
- Repeat the process recursively on the new conditional FP-tree



***m-conditional* pattern base:**

fca:2, fcab:1

m-conditional FP-tree
(min-support =3):

{}
|
f:3
|
c:3
|
a:3

All frequent patterns ending with m:

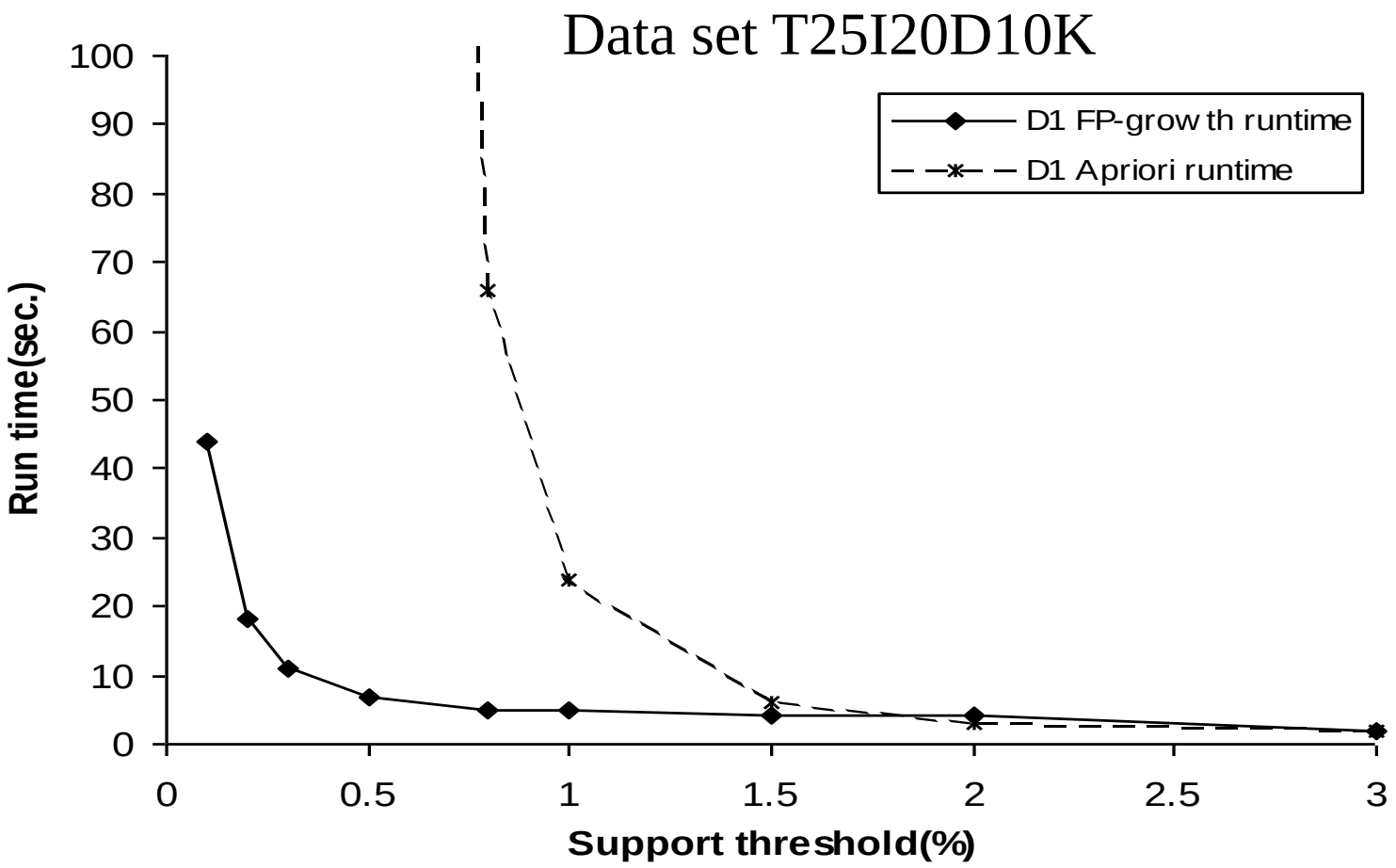
m,

fm, cm, am,

fcm, fam, cam,

fcam

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



Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - Decompose both mining task and DB and leads to focused search of smaller databases
  - Search least frequent items first for depth search, offering good selectivity
- 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

Scalable Methods for Mining Frequent Patterns

- Scalable mining methods for frequent patterns
  - Apriori (Agrawal & Srikant@VLDB'94) and variations
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
- Closed and maximal patterns and their mining methods
  - Concepts
  - Max-patterns: MaxMiner, MAFIA
  - Closed patterns: CLOSET, CLOSET+, CARPENTER
- FIMI Workshop

Closed Patterns and Max-Patterns

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

Closed Patterns and Max-Patterns

- Solution: *Mine "boundary" patterns*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ↄ X, *with the same support* as X (Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ↄ X (Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns and support counts

Max-patterns

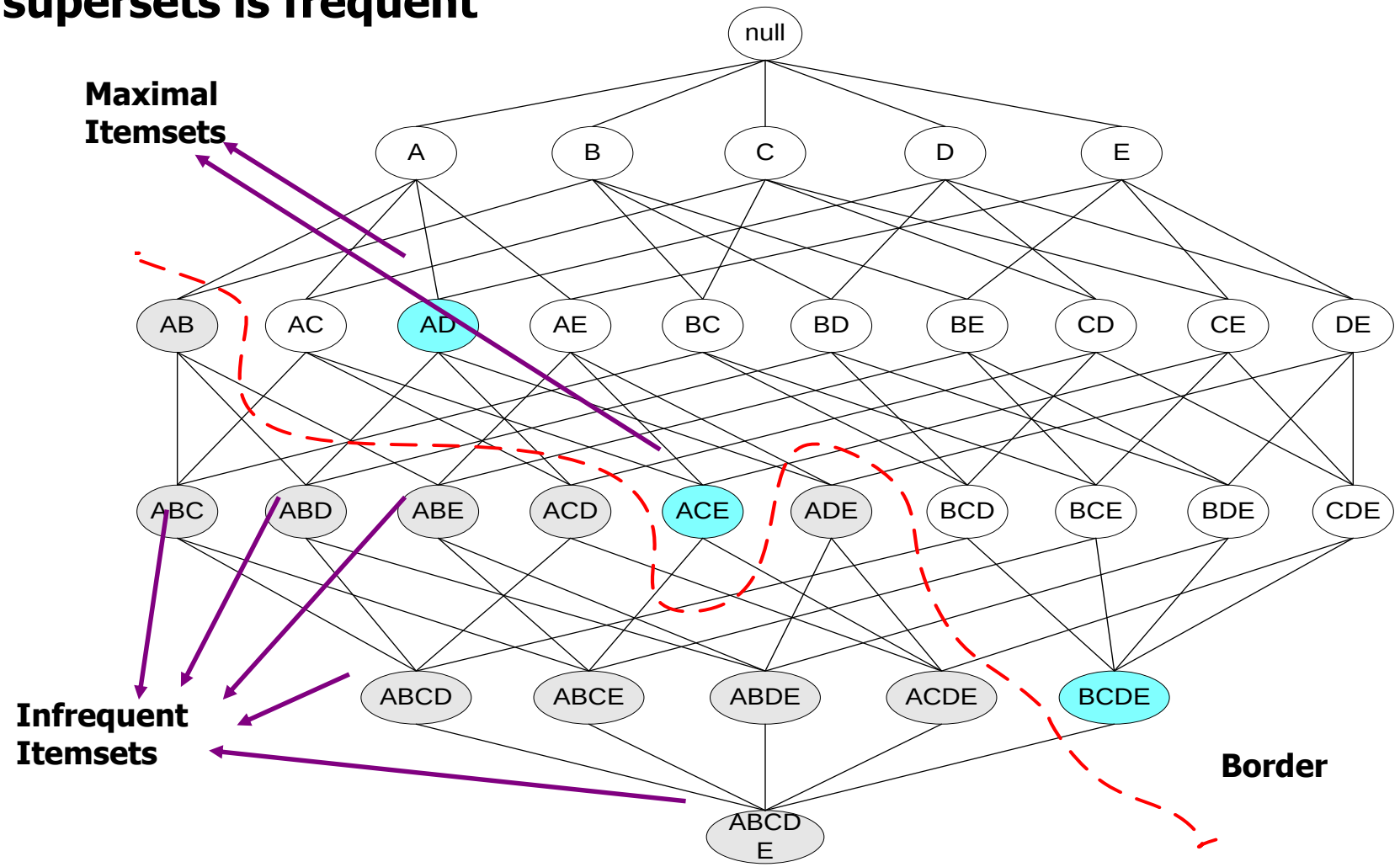
- Frequent patterns without frequent super patterns
  - BCDE (2), ACD (2) are max-patterns
  - BCD (2) is not a max-pattern

Min_sup=2

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

Max-Patterns Illustration

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed Patterns

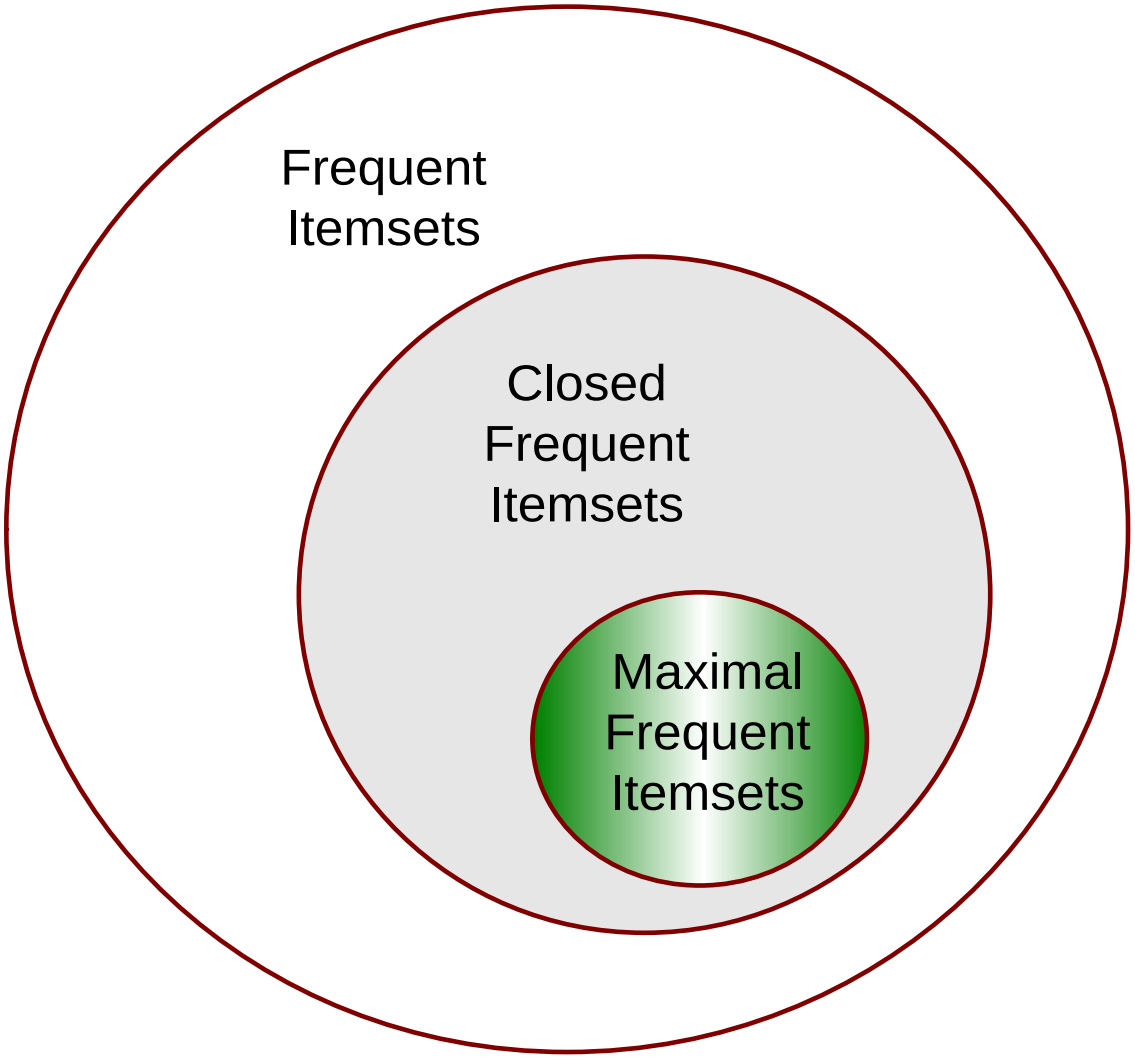
- An itemset is closed if none of its immediate supersets has the same support as the itemset (min_sup = 2)

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



Exercise: Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  min_sup = 1
- What is the set of closed itemset?

- What is the set of max-pattern?

- What is the set of all patterns?

Exercise: Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  min_sup = 1.
- What is the set of closed itemset?

  <a_1, …, a_{100}>: 1
  < a_1, …, a_{50}>: 2
- What is the set of max-pattern?

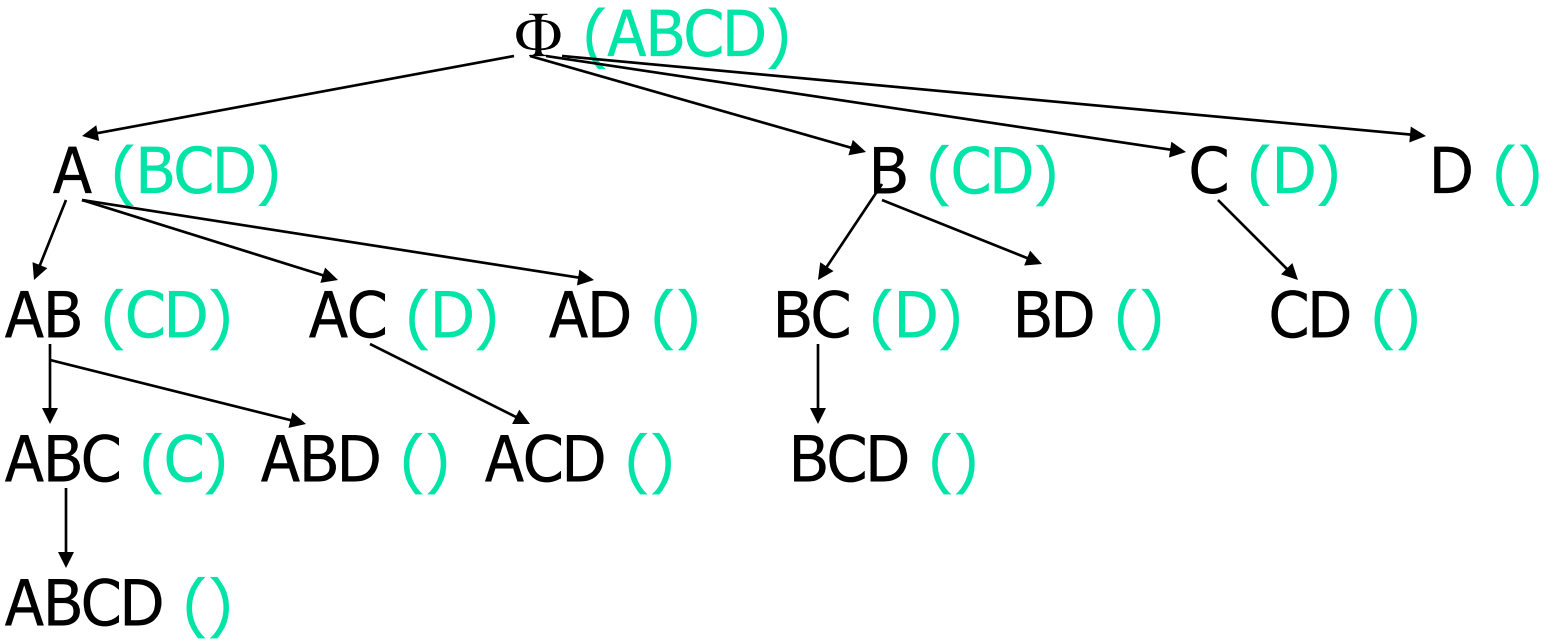
  <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Scalable Methods for Mining Frequent Patterns

- Scalable mining methods for frequent patterns
  - Apriori (Agrawal & Srikant@VLDB'94) and variations
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
- Closed and maximal patterns and their mining methods
  - Concepts
  - Max-pattern mining: MaxMiner, MAFIA
  - Closed pattern mining: CLOSET, CLOSET+, CARPENTER

MaxMiner: Mining Max-patterns

- R. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD'98*
- Idea: generate the complete set-enumeration tree one level at a time, prune if possible



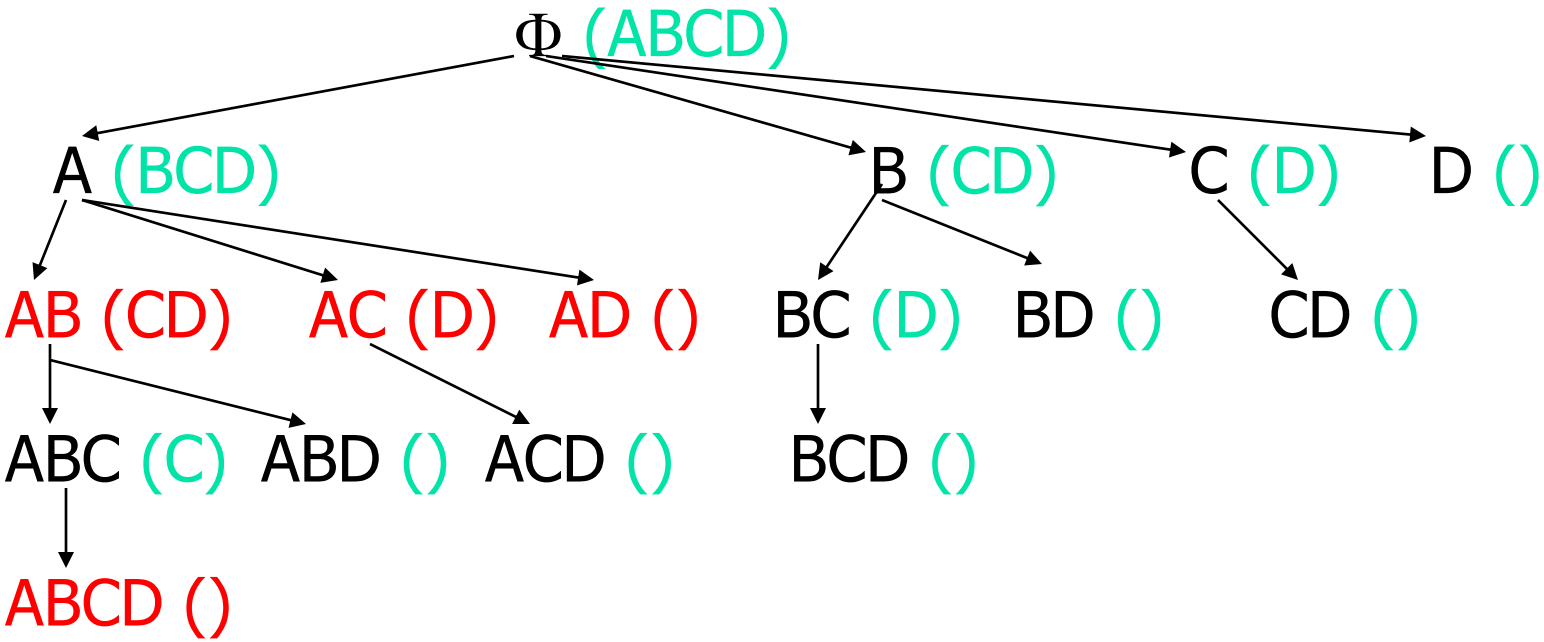
Algorithm MaxMiner

- Initially, generate one node N= Φ (ABCD), where h(N)=Φ and t(N)={A,B,C,D}.
- Recursively expanding N
  - Local pruning
    - If h(N)$\cup$t(N) (the leaf node) is frequent, do not expand N (prune entire subtree). (bottom-up pruning)
    - If for some i$\in$t(N), h(N)$\cup${i} (immediate child node) is NOT frequent, remove i from t(N) before expanding N (prune subbranch i). (top-down pruning)
  - Global pruning

Local Pruning Techniques (e.g. at node A)

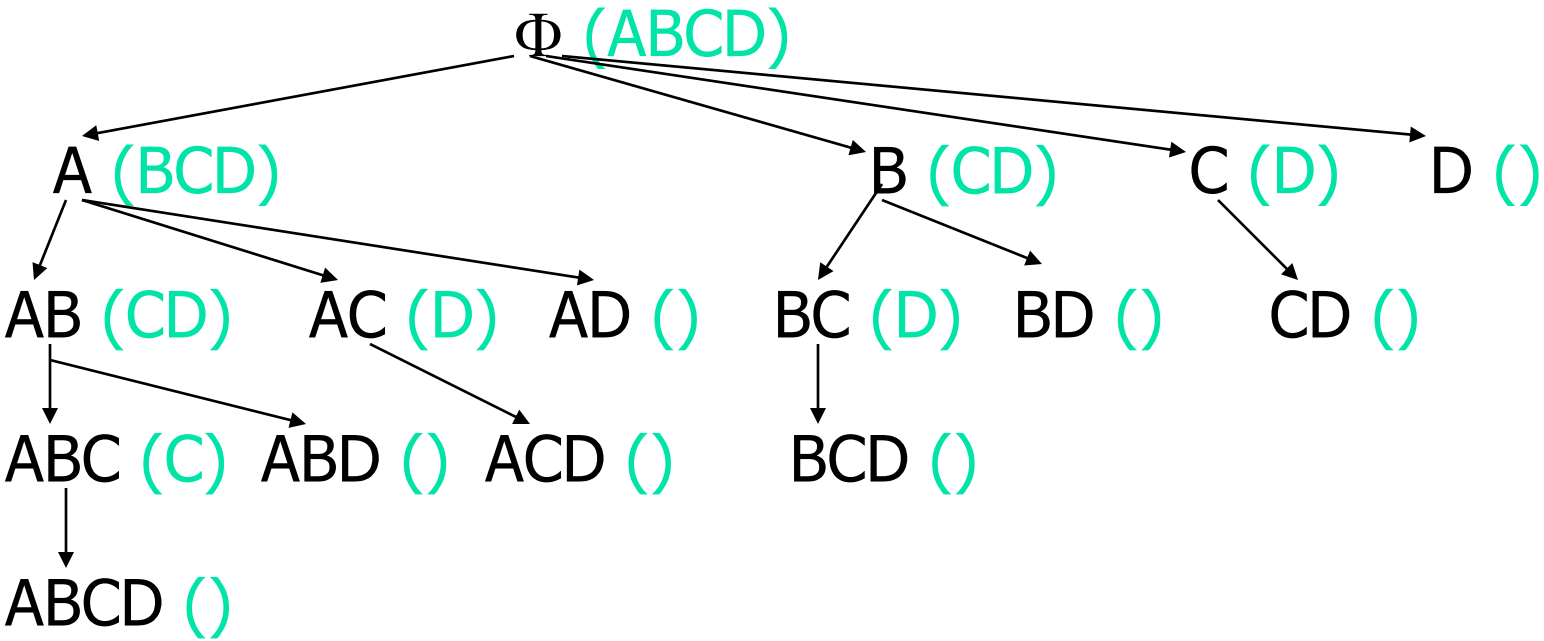
Check the frequency of ABCD and AB, AC, AD.

- If ABCD is frequent, prune the whole sub-tree.
- If AC is NOT frequent, prune C from the parenthesis before expanding (prune AC branch)

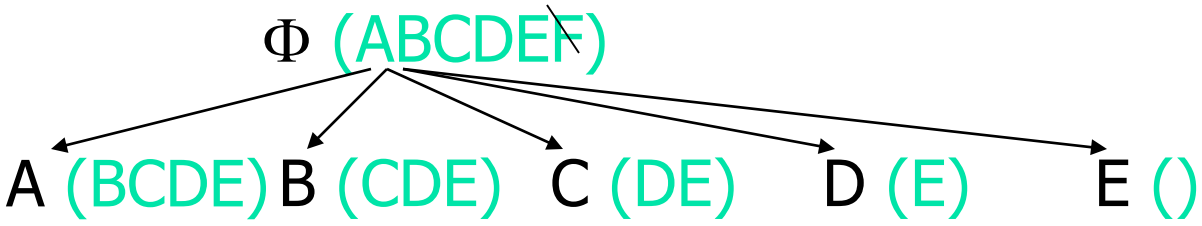


Global Pruning Technique (across sub-trees)

- When a max pattern is identified (e.g. BCD), prune all nodes (e.g. C, D) where h(N)∪t(N) is a sub-set of it (e.g. BCD).



Example



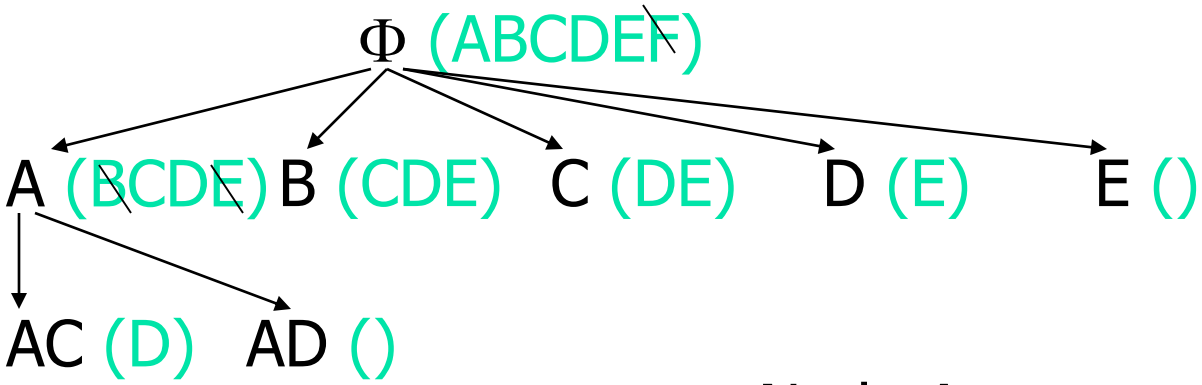
Items	Frequency
~~ABCDEF~~	~~0~~
A	2
B	2
C	3
D	3
E	2
~~F~~	~~1~~

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

Min_sup=2

Max patterns:

Example



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

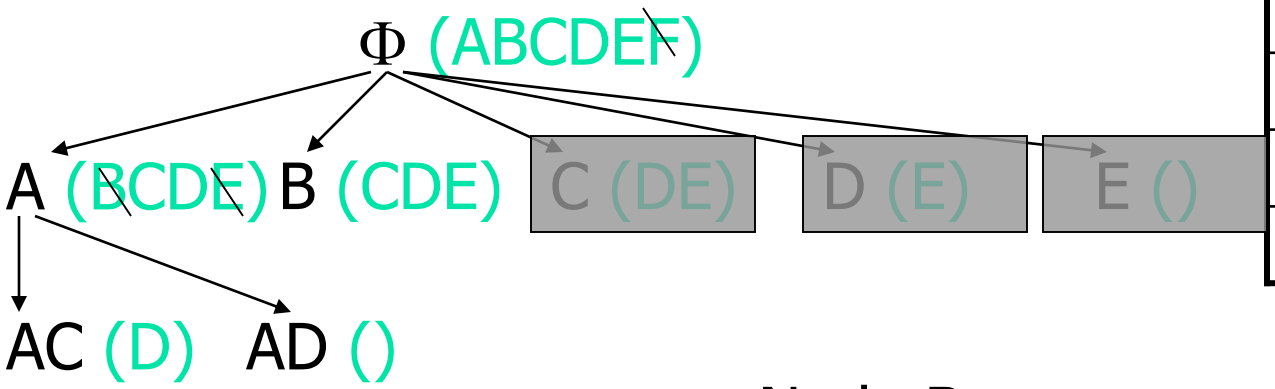
Min_sup=2

Node A

Items	Frequency
~~ABCDE~~	~~1~~
~~AB~~	~~1~~
AC	2
AD	2
~~AE~~	~~1~~

Max patterns:

Example



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

Min_sup=2

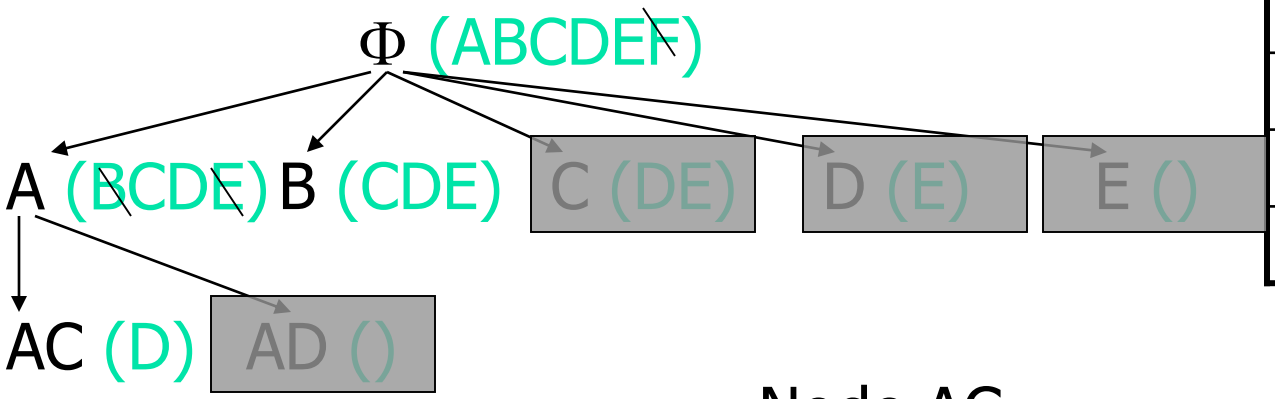
Node B

Items	Frequency
BCDE	2
BC	
BD	
BE	

Max patterns:

BCDE

Example



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

Min_sup=2

Node AC

Items	Frequency
ACD	2

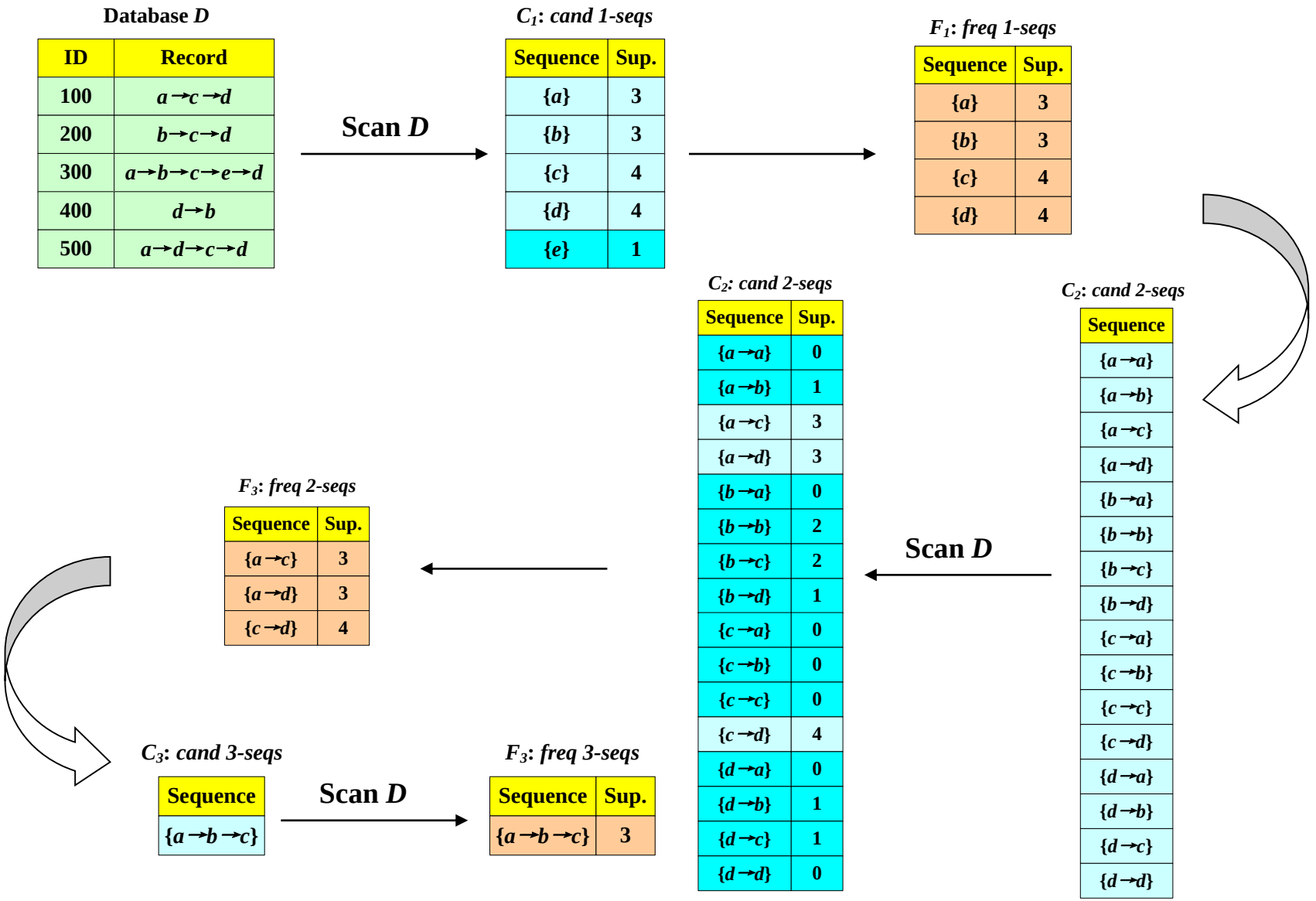
Max patterns:

BCDE
ACD

Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- Frequent sequence mining and graph mining
  - Apriori based: GSP (EDBT 96)
  - FP-Growth based: prefixSpan (ICDE 01)
- Mining various kinds of association rules
- From association mining to correlation analysis
- Summary

GSP Algorithm (Generalized Sequential Pattern)



Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Fpgrowth (Projection-based)
  - Max and closed pattern mining
- Sequence mining