
The background of the slide is a complex, abstract composition. It features a grid of small, light-colored plus signs on a pale pinkish-grey background. Overlaid on this is a network of thin, reddish-brown lines connecting various points, some of which are small green dots. A large, semi-transparent white trapezoidal shape is positioned in the center, serving as a backdrop for the title text. The text is in a bold, black, sans-serif font. There are also some faint, larger-scale geometric patterns, including a series of vertical lines on the right side and a cluster of orange and brown dots on the left side.

The Downward Closure Property of Frequent Patterns

The Downward Closure Property of Frequent Patterns

- ❑ Observation: From $TDB_1: T_1: \{a_1, \dots, a_{50}\}; T_2: \{a_1, \dots, a_{100}\}$
 - ❑ We get a frequent itemset: $\{a_1, \dots, a_{50}\}$
 - ❑ Also, its subsets are all frequent: $\{a_1\}, \{a_2\}, \dots, \{a_{50}\}, \{a_1, a_2\}, \dots, \{a_1, \dots, a_{49}\}, \dots$
 - ❑ There must be some hidden relationships among frequent patterns!
- ❑ The **downward closure (also called “Apriori”)** property of frequent patterns
 - ❑ If **$\{\text{beer, diaper, nuts}\}$** is frequent, so is **$\{\text{beer, diaper}\}$**
 - ❑ Every transaction containing $\{\text{beer, diaper, nuts}\}$ also contains $\{\text{beer, diaper}\}$
 - ❑ Apriori: Any subset of a frequent itemset must be frequent
- ❑ Efficient mining methodology
 - ❑ If **any subset of an itemset S** is infrequent, then there is no chance for S to be frequent—why do we even have to consider S !?  A sharp knife for pruning!

Apriori Pruning and Scalable Mining Methods

- ❑ Apriori pruning principle: If there is any itemset which is infrequent, its superset should not even be generated! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- ❑ Scalable mining Methods: Three major approaches
 - ❑ Level-wise, join-based approach: Apriori (Agrawal & Srikant@VLDB'94)
 - ❑ Vertical data format approach: Eclat (Zaki, Parthasarathy, Ogihara, Li @KDD'97)
 - ❑ Frequent pattern projection and growth: FPgrowth (Han, Pei, Yin @SIGMOD'00)

The background of the slide is a complex, abstract composition. It features a dark, reddish-brown base with a network of thin, light-colored lines forming a web-like structure. Scattered throughout are small, colorful dots in shades of green, blue, and orange. A prominent white banner with a slight 3D effect and a drop shadow runs horizontally across the center of the image. On the left side, there is a smaller, semi-transparent inset image showing a different pattern of dots and lines. The overall aesthetic is technical and modern.

The Apriori Algorithm

Apriori: A Candidate Generation & Test Approach

- ❑ Outline of Apriori (level-wise, candidate generation and test)
 - ❑ Initially, scan DB once to get frequent 1-itemset
 - ❑ Repeat
 - ❑ Generate length-(k+1) candidate itemsets from length-k frequent itemsets
 - ❑ Test the candidates against DB to find frequent (k+1)-itemsets
 - ❑ Set $k := k + 1$
 - ❑ Until no frequent or candidate set can be generated
 - ❑ Return all the frequent itemsets derived

The Apriori Algorithm (Pseudo-Code)

C_k : Candidate itemset of size k

F_k : Frequent itemset of size k

$K := 1$;

$F_k := \{\text{frequent items}\}$; // frequent 1-itemset

While ($F_k \neq \emptyset$) **do** { // when F_k is non-empty

$C_{k+1} := \text{candidates generated from } F_k$; // candidate generation

 Derive F_{k+1} by counting candidates in C_{k+1} with respect to TDB at minsup;

$k := k + 1$

}

return $\cup_k F_k$ // return F_k generated at each level

The Apriori Algorithm—An Example

Database TDB

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

minsup = 2

C_1

1st scan

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

F_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

F_2

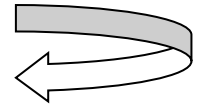
Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

2nd scan

C_2

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



C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

Apriori: Implementation Tricks

□ How to generate candidates?

□ Step 1: self-joining F_k

□ Step 2: pruning

□ Example of candidate-generation

□ $F_3 = \{abc, abd, acd, ace, bcd\}$

□ Self-joining: $F_3 * F_3$

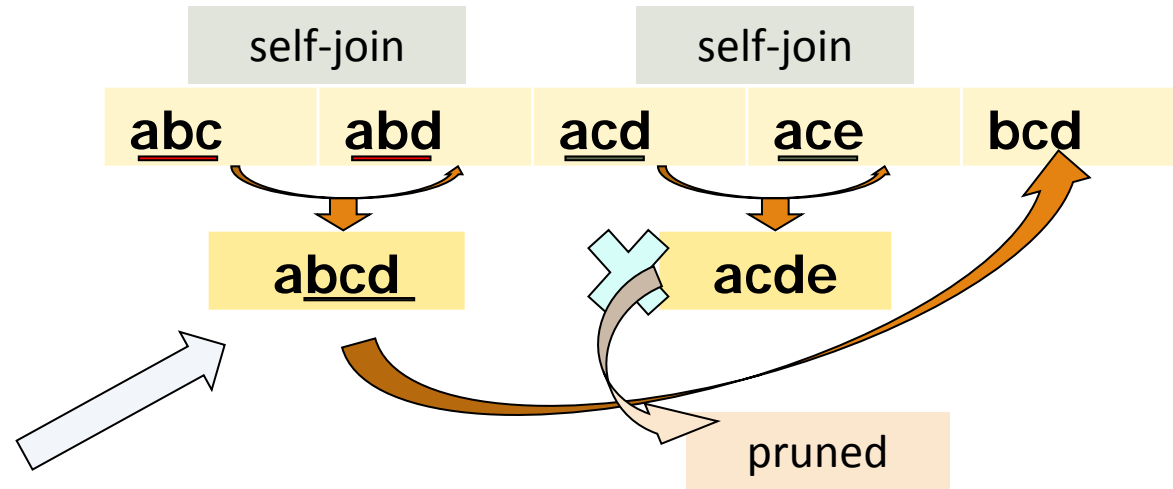
□ $abcd$ from abc and abd

□ $acde$ from acd and ace

□ Pruning:

□ $acde$ is removed because ade is not in F_3

□ $C_4 = \{abcd\}$



Candidate Generation: An SQL Implementation

- Suppose the items in F_{k-1} are listed in an order

- Step 1: self-joining F_{k-1}
insert into C_k

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

from F_{k-1} as p, F_{k-1} as q

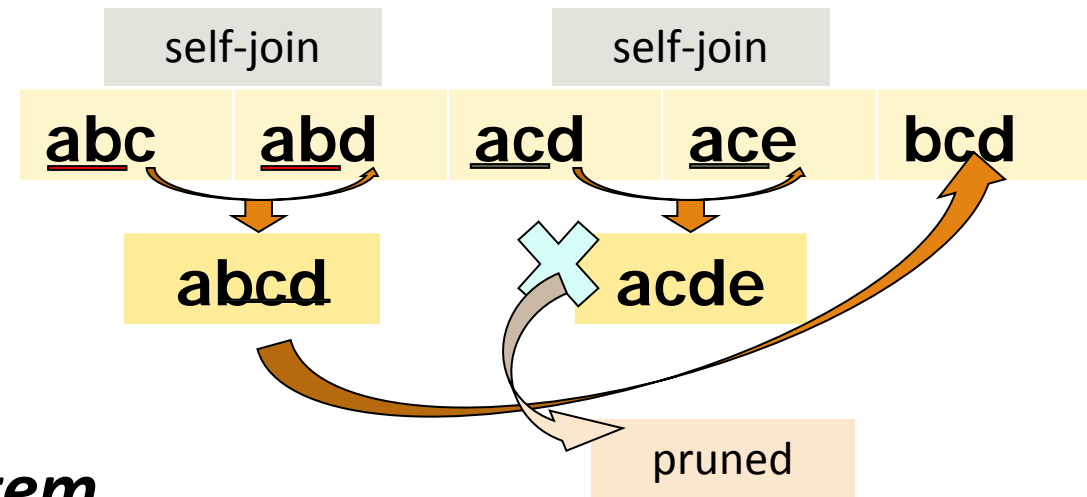
where $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

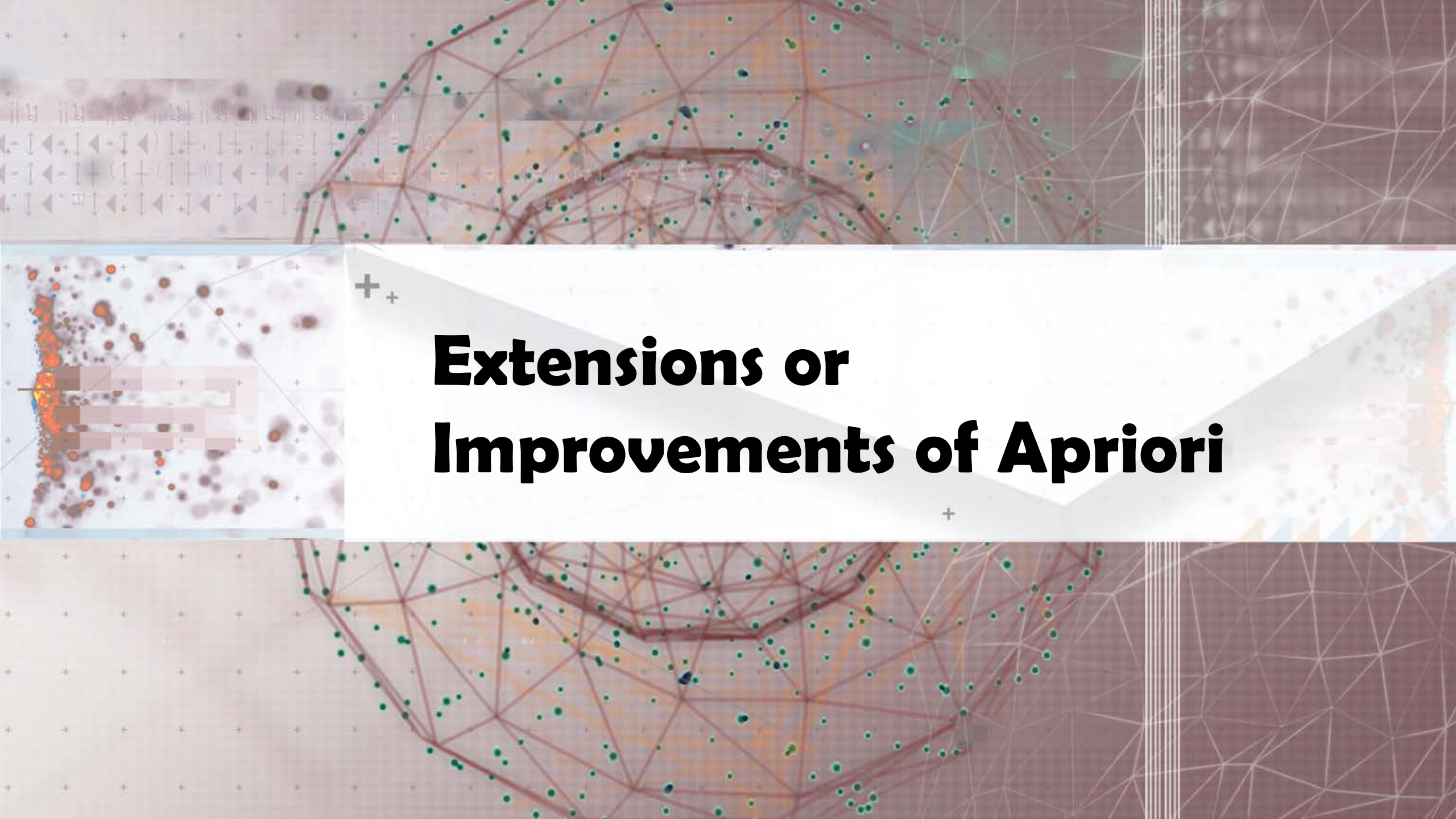
- Step 2: pruning

for all *itemsets* c in C_k do

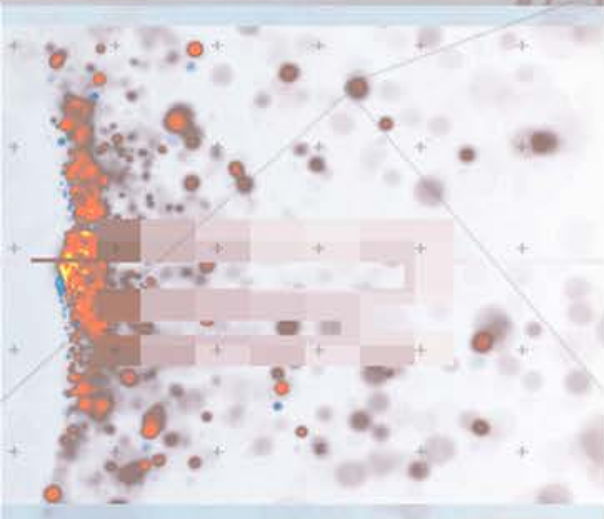
for all *(k-1)-subsets* s of c do

if (s is not in F_{k-1}) then delete c from C_k





The background features a complex network of thin, light-colored lines forming a web-like structure. Scattered throughout are numerous small, colored dots in shades of green, blue, and orange. A prominent, darker, reddish-brown geometric shape, resembling a stylized 'X' or a complex polygon, is centered in the upper half. The overall aesthetic is technical and data-driven.

Extensions or Improvements of Apriori



Apriori: Improvements and Alternatives

- ❑ Reduce passes of transaction database scans
 - ❑ Partitioning (e.g., Savasere, et al., 1995) 
 - ❑ Dynamic itemset counting (Brin, et al., 1997)
- ❑ Shrink the number of candidates
 - ❑ Hashing (e.g., DHP: Park, et al., 1995) 
 - ❑ Pruning by support lower bounding (e.g., Bayardo 1998)
 - ❑ Sampling (e.g., Toivonen, 1996)
- ❑ Exploring special data structures
 - ❑ Tree projection (Agarwal, et al., 2001)
 - ❑ H-miner (Pei, et al., 2001)
 - ❑ Hypercube decomposition (e.g., LCM: Uno, et al., 2004)

To be discussed in subsequent slides

To be discussed in subsequent slides

Partitioning: Scan Database Only Twice

- Theorem: *Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB*



- Method: (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)
 - Scan 1: Partition database (how?) and find local frequent patterns
 - Scan 2: Consolidate global frequent patterns (how to?)
- Why does this method guarantee to scan TDB only twice?

Direct Hashing and Pruning (DHP)

- ❑ DHP (Direct Hashing and Pruning): Reduce the number of candidates (J. Park, M. Chen, and P. Yu, SIGMOD'95)
- ❑ Observation: A k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

- ❑ Candidates: a, b, c, d, e

- ❑ Hash entries

- ❑ {ab, ad, ae}

- ❑ {bd, be, de}

- ❑ ...

Itemsets	Count
{ab, ad, ae}	35
{bd, be, de}	298
.....	...
{yz, qs, wt}	58

Hash Table

- ❑ Frequent 1-itemset: a, b, d, e

- ❑ ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

The background features a complex network of red lines connecting green and blue dots, resembling a data graph. On the left, there is a smaller inset showing a grid of orange and red dots. The central text is overlaid on a white, angular geometric shape.

Mining Frequent Patterns by Exploring Vertical Data Format

Exploring Vertical Data Format: ECLAT

- ❑ ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- ❑ Tid-List: List of transaction-ids containing an itemset
- ❑ Vertical format: $t(e) = \{T_{10}, T_{20}, T_{30}\}$; $t(a) = \{T_{10}, T_{20}\}$; $t(ae) = \{T_{10}, T_{20}\}$
- ❑ Properties of Tid-Lists
 - ❑ $t(X) = t(Y)$: X and Y always happen together (e.g., $t(ac) = t(d)$)
 - ❑ $t(X) \subset t(Y)$: transaction having X always has Y (e.g., $t(ac) \subset t(ce)$)
- ❑ Deriving frequent patterns based on vertical intersections
- ❑ Using **diffset** to accelerate mining
 - ❑ Only keep track of differences of tids
 - ❑ $t(e) = \{T_{10}, T_{20}, T_{30}\}$, $t(ce) = \{T_{10}, T_{30}\} \rightarrow \text{Diffset}(ce, e) = \{T_{20}\}$

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

The background features a complex geometric pattern of thin, light-colored lines forming a network of triangles and polygons. Overlaid on this are several semi-transparent rectangular panels. The top-left panel shows a grid of small, colorful dots (green, blue, orange) connected by lines. The bottom-left panel displays a dense cluster of orange and red dots. The right side of the image has a vertical strip of white lines. A large, white, angular shape serves as a backdrop for the central text.

FPGrowth: A Pattern Growth Approach

FPGrowth: Mining Frequent Patterns by Pattern Growth

- ❑ Idea: Frequent pattern growth (FPGrowth)
 - ❑ Find frequent single items and partition the database based on each such item
 - ❑ Recursively grow frequent patterns by doing the above for each partitioned database (also called *conditional database*)
 - ❑ To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- ❑ Mining becomes
 - ❑ Recursively construct and mine (conditional) FP-trees
 - ❑ Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Example: Construct FP-tree from a Transactional DB

TID	Items in the Transaction	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}

1. Scan DB once, find single item frequent pattern: **Let min_support = 3**

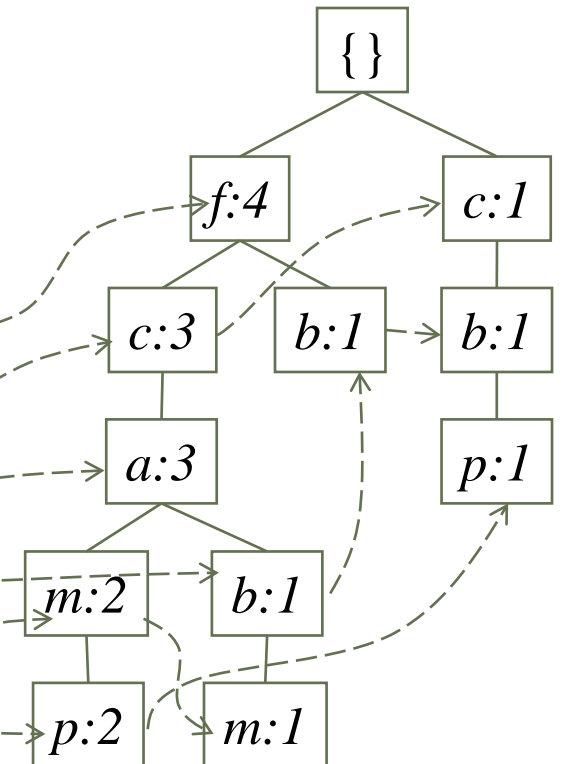
f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

F-list = f-c-a-b-m-p

3. Scan DB again, construct FP-tree

Header Table		
Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

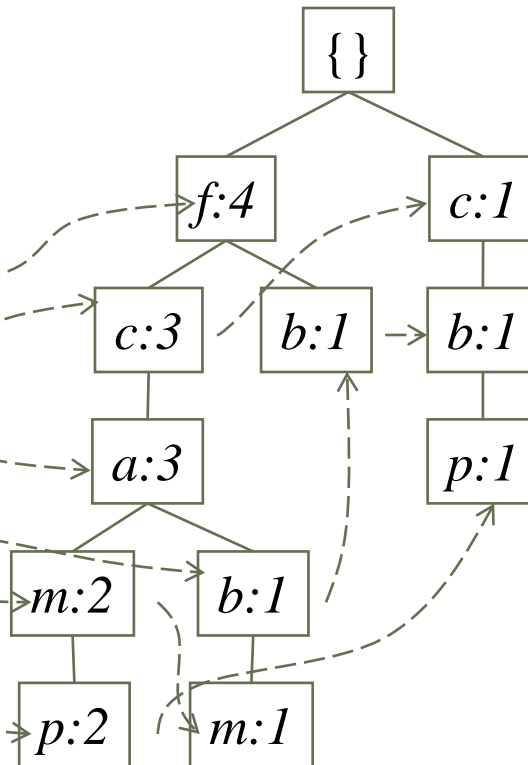


Divide and Conquer Based on Patterns and Data

- Pattern mining can be partitioned according to current patterns
 - Patterns containing p: p's conditional database: $fcam:2, cb:1$
 - Patterns having m but no p: m's conditional database: $fca:2, fcab:1$
 -
- p's conditional pattern base: *transformed prefix paths* of item p

min_support = 3

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Conditional pattern bases

<u>Item</u>	<u>Conditional pattern base</u>
c	$f:3$
a	$fc:3$
b	$fca:1, f:1, c:1$
m	$fca:2, fcab:1$
p	$fcam:2, cb:1$

Mine Each Conditional Pattern-Base Recursively

Conditional pattern bases

<i>item</i>	<i>cond. pattern base</i>
<i>c</i>	<i>f:3</i> min_support = 3
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

□ For each conditional pattern-base

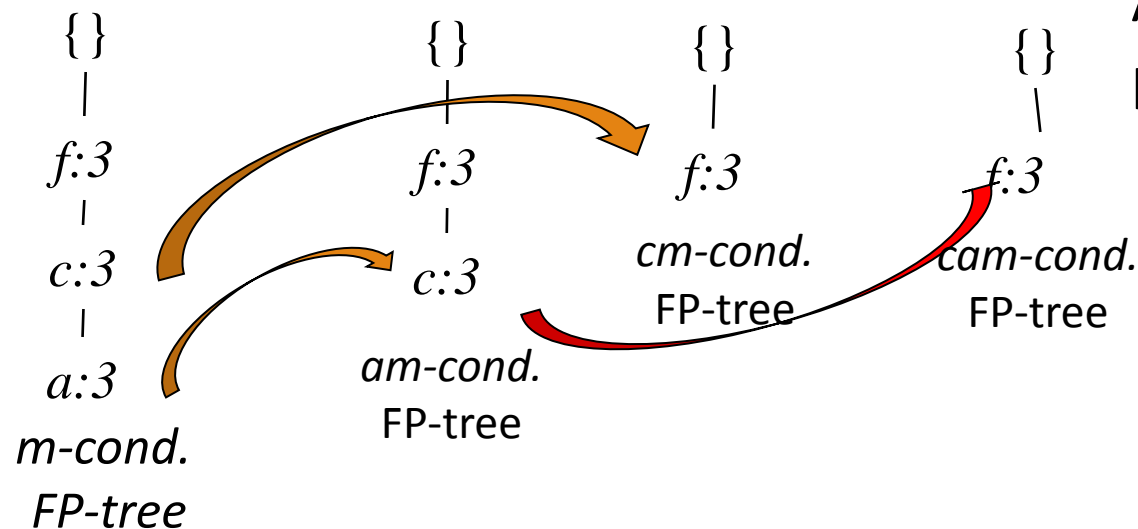
□ Mine single-item patterns

□ Construct its FP-tree & mine it

p-conditional PB: *fcam:2, cb:1* → *c: 3*

m-conditional PB: *fca:2, fcab:1* → *fca: 3*

b-conditional PB: *fca:1, f:1, c:1* → ϕ



Actually, for single branch FP-tree, all frequent patterns can be generated in one shot

m: 3

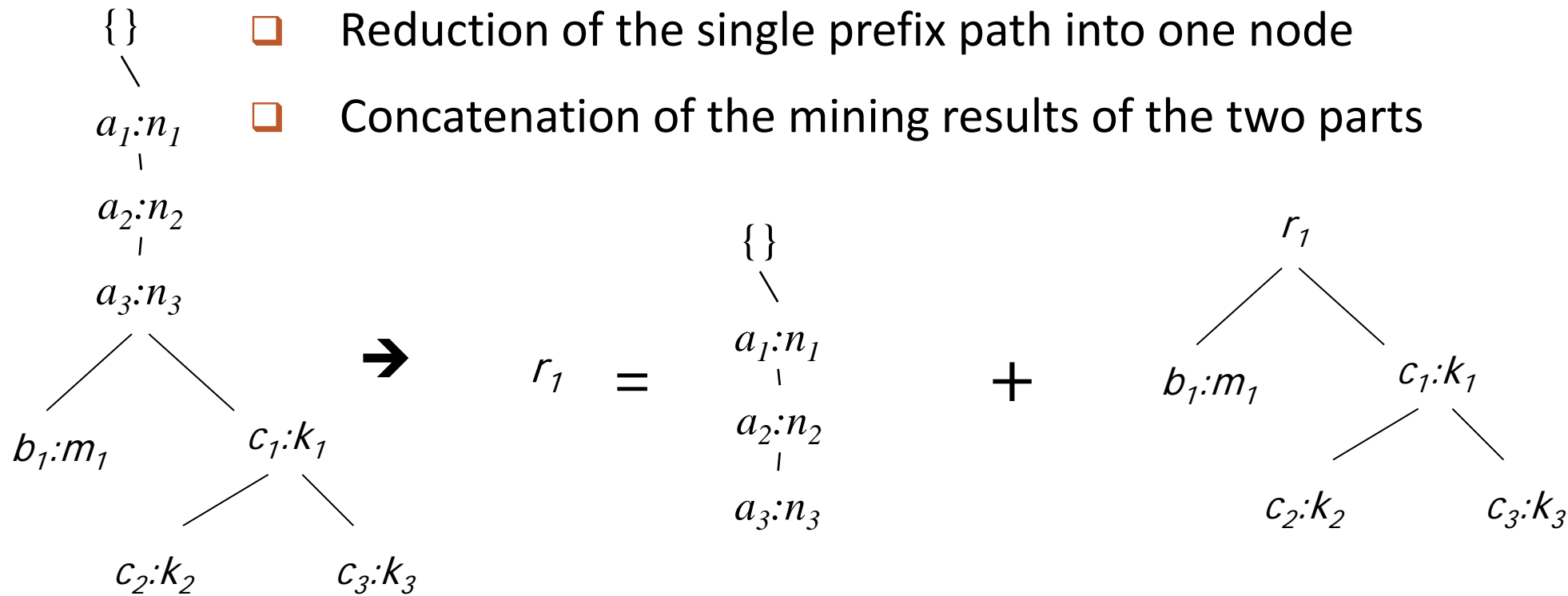
fm: 3, cm: 3, am: 3

fcm: 3, fam:3, cam: 3

fcam: 3

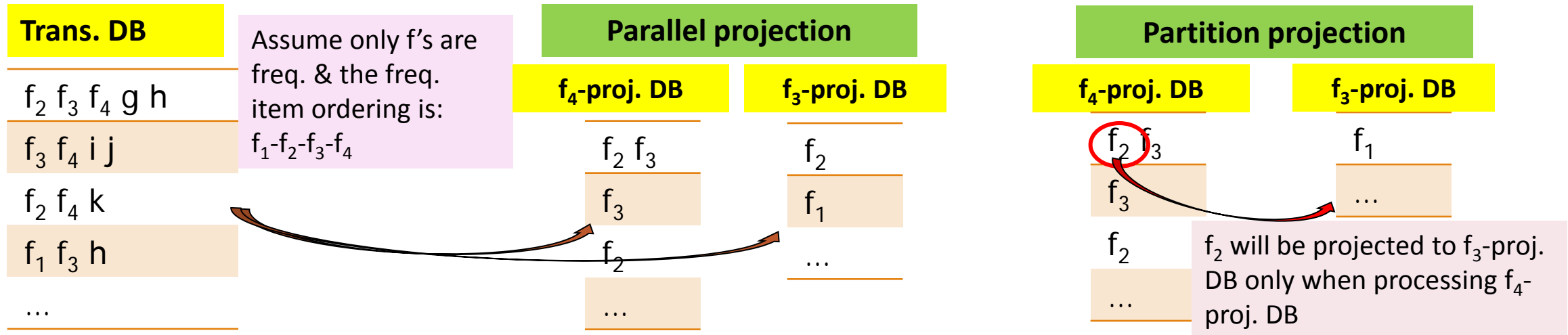
A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts



Scaling FP-growth by Database Projection

- What if FP-tree cannot fit in memory? — DB projection
 - Project the DB based on patterns
 - Construct & mine FP-tree for each projected DB
- Parallel projection** vs. **partition projection**
 - Parallel projection: Project the DB on each frequent item
 - Space costly, all partitions can be processed in parallel
 - Partition projection: Partition the DB in order
 - Passing the unprocessed parts to subsequent partitions



The background of the slide is a complex, abstract composition. It features a central white banner with a subtle 3D effect, flanked by two large, light gray triangular shapes that point towards the center. The background is filled with a dense network of thin, light gray lines forming a triangular mesh. Overlaid on this are various patterns: a grid of small gray plus signs on the left, a series of purple arrows pointing left in the upper left, and a cluster of orange and red dots on the left side. The overall color palette is muted, with earthy tones and soft grays.

Mining Closed Patterns

CLOSET+: Mining Closed Itemsets by Pattern-Growth

- ❑ Efficient, *direct* mining of closed itemsets
- ❑ Ex. Itemset merging: If Y appears in every occurrence of X, then Y is merged with X
 - ❑ d-proj. db: {acef, acf} → acfd-proj. db: {e}, thus we get: acfd:2
- ❑ Many other tricks (but not detailed here), such as
 - ❑ Hybrid tree projection
 - ❑ Bottom-up physical tree-projection
 - ❑ Top-down pseudo tree-projection
 - ❑ Sub-itemset pruning
 - ❑ Item skipping
 - ❑ Efficient subset checking
- ❑ For details, see J. Wang, et al., “CLOSET+:”, KDD'03

TID	Items
1	acdef
2	abe
3	cefg
4	acdf

Let minsupport = 2

a:3, c:3, d:2, e:3, f:3

F-List: a-c-e-f-d

Recommended Readings

- ❑ R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, VLDB'94
- ❑ A. Savasere, E. Omiecinski, and S. Navathe, “An efficient algorithm for mining association rules in large databases”, VLDB'95
- ❑ J. S. Park, M. S. Chen, and P. S. Yu, “An effective hash-based algorithm for mining association rules”, SIGMOD'95
- ❑ S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications”, SIGMOD'98
- ❑ M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “Parallel algorithm for discovery of association rules”, Data Mining and Knowledge Discovery, 1997
- ❑ J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD'00
- ❑ M. J. Zaki and Hsiao, “CHARM: An Efficient Algorithm for Closed Itemset Mining”, SDM'02
- ❑ J. Wang, J. Han, and J. Pei, “CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets”, KDD'03
- ❑ C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, “Frequent Pattern Mining Algorithms: A Survey”, in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014