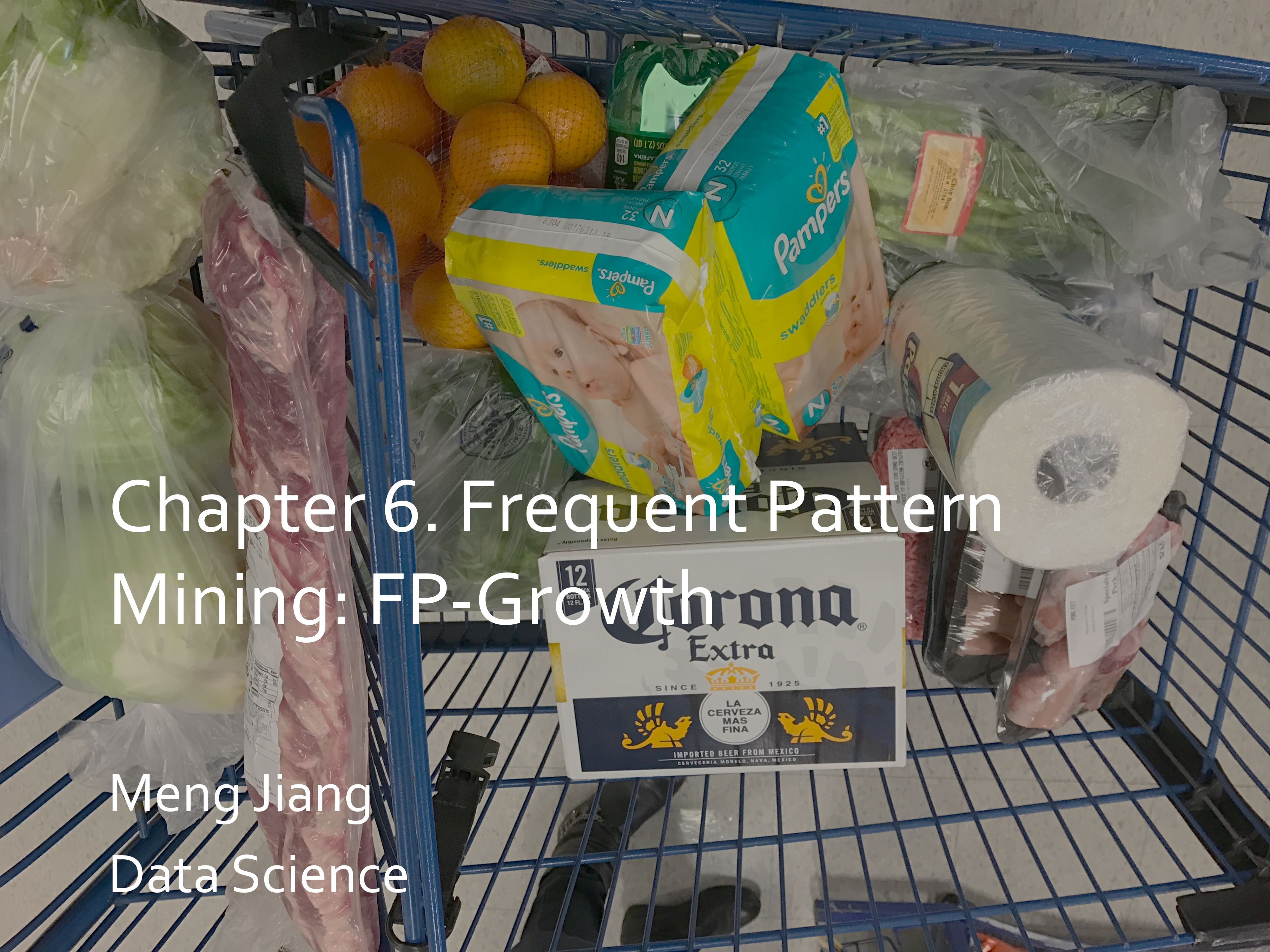
Chapter 6. Frequent Pattern Mining: FP-Growth

Meng Jiang
Data Science



Review Last Lecture

- **Frequent itemsets (patterns):** Itemset, k-itemset, (absolute/relative) support, minimum support
- **Association rules:** Support, confidence
- **Closed patterns** and **Max-patterns**
- **Apriori**
  – The downward closure property: Any subset of a frequent itemset must be frequent
  – Algorithm: Level-wise, candidate generation, test
  – Partition for parallelization

Today's Lecture

- Frequent itemset mining
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
    - **Direct hashing and pruning: DHP** (Park, Chen, Yu@SIGMOD'95)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)
- Closed itemset mining
  - **Pattern growth-based approach: CLOSET+** (Wang et al. @KDD'03)

Today's Learning Goals

- Describe DHP, Eclat, FPGrowth, and CLOSET+
- Implement **FPGrowth**
  - Solve the *frequent itemset mining* problem **by hand** if the database is small, say, 10 transactions
  - Solve the problem by **programming** given an arbitrary size of transaction database and minimum support

Review: The Apriori Algorithm

Database TDB

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

minsup = 2

1st scan

C_1

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

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

F_2

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

C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

Review: The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k
F_k : Frequent itemset of size k

K := 1;
F_k := {frequent items}; // frequent 1-itemset

Issue: Too many candidates!!!

While (F_k != ∅) **do {** // when F_k is non-empty
   C_{k+1} := 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

Today's Lecture

- Frequent itemset mining
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
    - **Direct hashing and pruning: DHP** (Park, Chen, Yu@SIGMOD'95)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)
- Closed itemset mining
  - **Pattern growth-based approach: CLOSET+** (Wang et al. @KDD'03)

Improving Efficiency of Apriori

- Bottlenecks
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - **Shrink number of candidates**
  - Reduce passes of transaction database scans
  - Reduce number of transactions
  - Facilitate support counting of candidates

Direct Hashing and Pruning (DHP)

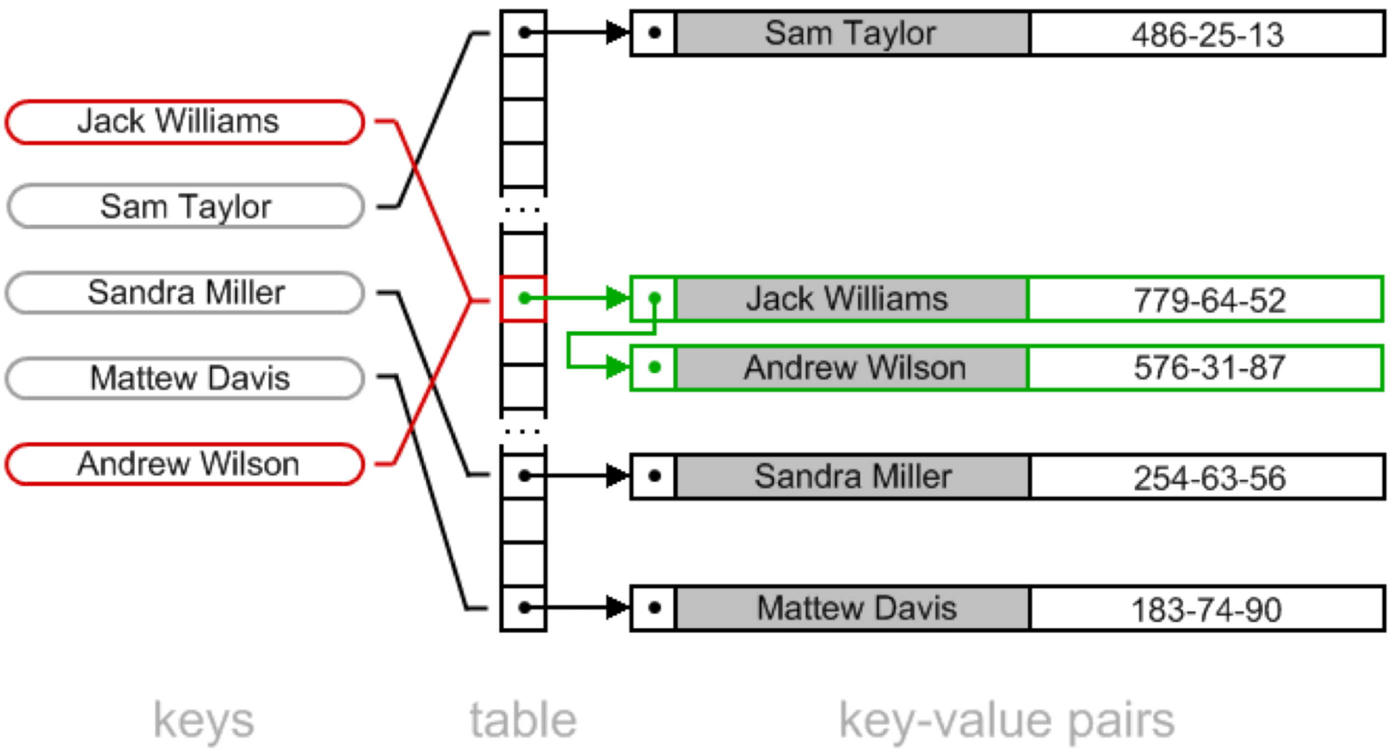
- DHP (Direct Hashing and Pruning): Hash k-itemsets into buckets and a k-itemset whose bucket count is below the threshold cannot be frequent
- Especially useful for 2-itemsets
  - Generate a hash table of 2-itemsets during the scan for 1-itemset
    - Hash entries
      - {ab, ad, ae}
      - {bd, be, de}
      - ...

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

Hash Table

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

*Hash Table



Sam Taylor
486-25-13
Jack Williams
Sam Taylor
Sandra Miller
Mattew Davis
Andrew Wilson
Jack Williams
779-64-52
Andrew Wilson
576-31-87
Sandra Miller
254-63-56
Mattew Davis
183-74-90
keys
table
key-value pairs

DHP (cont.)

- Especially useful for 2-itemsets
  - Generate a hash table of 2-itemsets during the scan for 1-itemset
  - If the count of a bucket is below minimum support count, the itemsets in the bucket should not be included in candidate 2-itemsets
    - 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 (e.g., 50)**

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

Hash Table

Today's Lecture

- Frequent itemset mining
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
    - **Direct hashing and pruning: DHP** (Park, Chen, Yu@SIGMOD'95)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)
- Closed itemset mining
  - **Pattern growth-based approach: CLOSET+** (Wang et al. @KDD'03)

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: $t(e) = \{T_{10}, T_{20}, T_{30}\}$; $t(a) = \{T_{10}, T_{20}\}$; $t(ae) = \{T_{10}, T_{20}\}$

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

ECLAT (cont.)

- 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., $t(ac) \subset t(ce)$)

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

ECLAT (cont.)

- 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

Today's Lecture

- Frequent itemset mining
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
    - **Direct hashing and pruning: DHP** (Park, Chen, Yu@SIGMOD'95)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)
- Closed itemset mining
  - **Pattern growth-based approach: CLOSET+** (Wang et al. @KDD'03)

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

J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In SIGMOD'00.

FPGrowth (cont.)

- 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

FPGrowth: Example

1. Scan DB once, find single item frequent pattern:

Let min_support = 3

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

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

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

TID	Items in the Transaction
100	*{f, a, c, d, g, i, m, p}*
200	*{a, b, c, f, l, m, o}*
300	*{b, f, h, j, o, w}*
400	*{b, c, k, s, p}*
500	*{a, f, c, e, l, p, m, n}*

TID	Ordered, frequent items
100	*{f, c, a, m, p}*
200	*{f, c, a, b, m}*
300	*{f, b}*
400	*{c, b, p}*
500	*{f, c, a, m, p}*

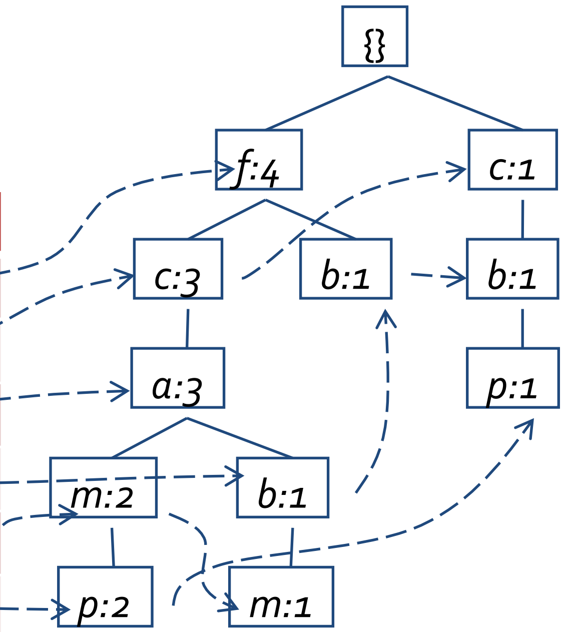
FPGrowth: Example (cont.)

3. Scan DB again, construct FP-tree

TID	Ordered, frequent items
100	{*f, c, a, m, p*}
200	{*f, c, a, b, m*}
300	{*f, b*}
400	{*c, b, p*}
500	{*f, c, a, m, p*}

Header Table

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



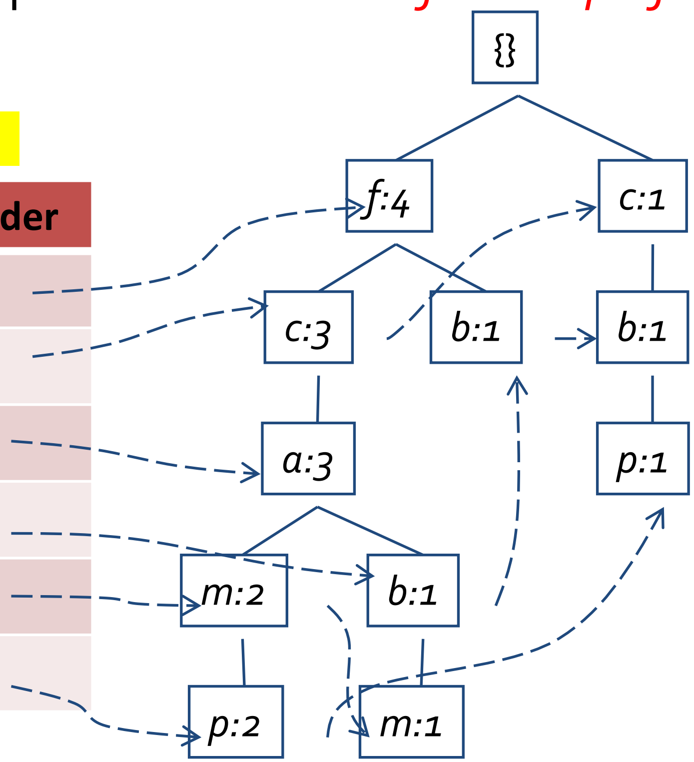
Divide and Conquer Based on Patterns and Data

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

- 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**

Item	*Conditional 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***

Mine Each Conditional Pattern-Base Recursively

Conditional pattern bases

<u>*item*</u>	<u>*cond. 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***

min_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & mine it

p-conditional PB: ***fcam:2, cb:1*** $\rightarrow$ ***c: 3***

m-conditional PB: ***fca:2, fcab:1*** $\rightarrow$ ***fca: 3***

b-conditional PB: ***fca:1, f:1, c:1*** $\rightarrow$ $\boldsymbol{\phi}$

a-conditional PB: ***fc:3*** $\rightarrow$ ***fc:3***

c-conditional PB: ***f:3*** $\rightarrow$ ***f:3***

Mine Each Conditional Pattern-Base Recursively

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & mine it

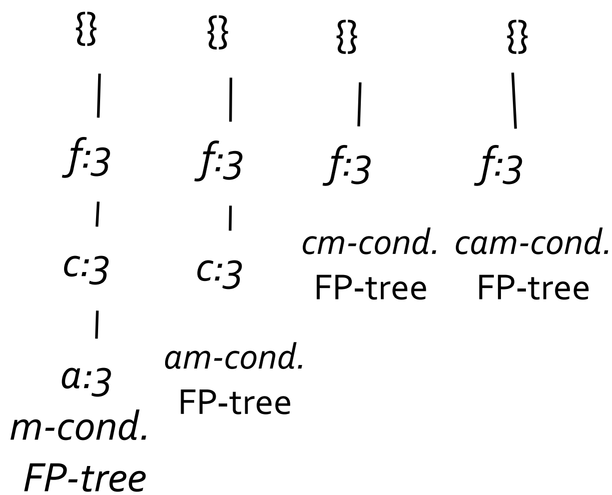
p-conditional PB: ***fcam:2, cb:1*** → ***c: 3***

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

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

a-conditional PB: ***fc:3*** → ***fc:3***

c-conditional PB: ***f:3*** → ***f:3***



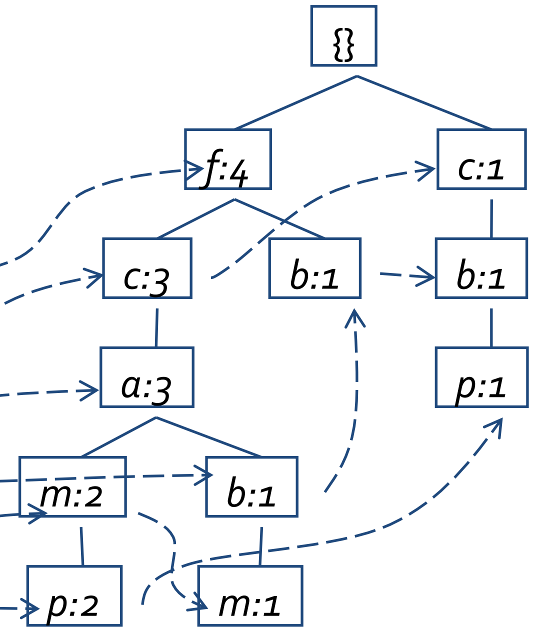
Actually, for single branch FP-tree, all frequent patterns can be generated in one shot

m: 3
fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3
fcam: 3

Can you find all the frequent itemsets?

TID	Ordered, frequent items
100	*{f, c, a, m, p}*
200	*{f, c, a, b, m}*
300	*{f, b}*
400	*{c, b, p}*
500	*{f, c, a, m, p}*

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



Conditional **pattern bases**

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*

……

Answer:
f:4, a:3, c:4, b:3, m:3, p:3;
fa:3, fc:3, fm:3, ac:3, am:3, cm: 3, cp:3;
fcm: 3, fam:3, cam: 3;
fcam: 3.

Handout Exercise: FPGrowth vs. Apriori

Database TDB

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

minsup = 2

1^{st} scan

C_1

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

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

F_2

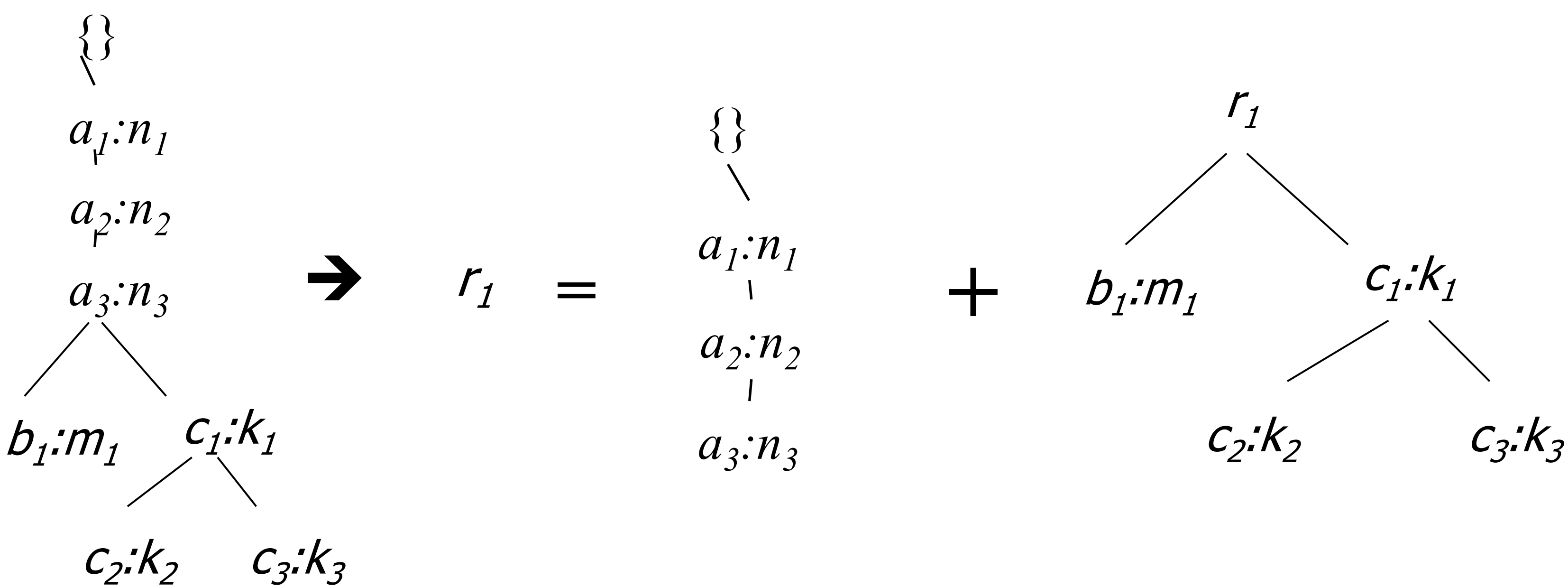
Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

Itemset

Answer: 1-itemsets: A:2, B:3, C:3, E:3; 2-itemsets: AC:2, BC: 2, BE: 3, CE: 2; 3-itemset: BCE: 2.

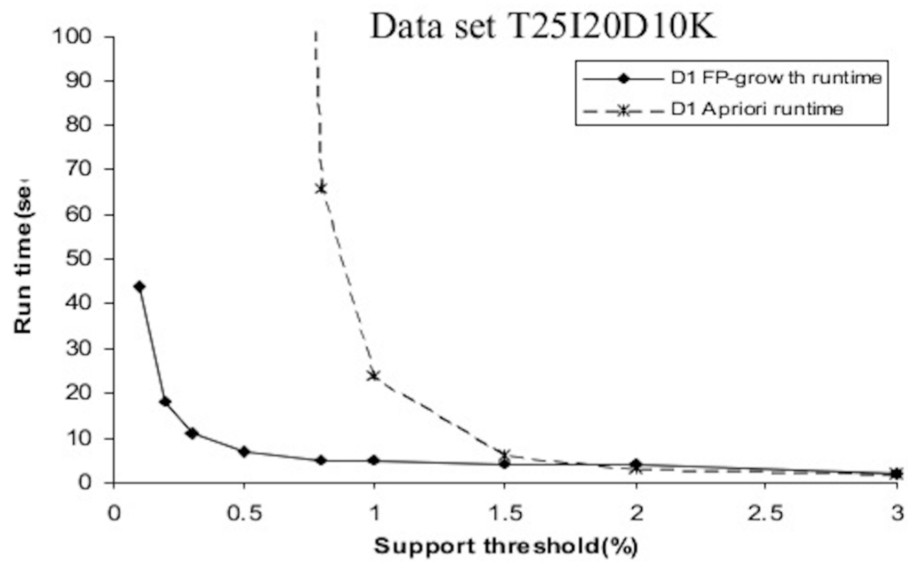
A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree has a shared single prefix-path
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



Scaling FP-growth by Database Projection

- More efficient than Apriori



- What if FP-tree cannot fit in memory? — DB projection
  - Project the DB based on patterns
  - Construct & mine FP-tree for each projected DB

Scaling FP-growth by Database Projection (cont.)

- **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

Trans. DB

f_2 f_3 f_4 g h
f_3 f_4 i j
f_2 f_4 k
f_1 f_3 h
…

Assume only f's are freq. & the freq. item ordering is: f_1-f_2-f_3-f_4

Parallel projection

f_4-proj. DB	**f_3-proj. DB**
f_2 f_3	f_2
f_3	f_1
f_2	…
…	

Partition projection

f_4-proj. DB	**f_3-proj. DB**
f_2 f_3	f_1
f_3	…
f_2	
…	

f_2 will be projected to f_3-proj. DB only when processing f_4-proj. DB

Today's Lecture

- Frequent itemset mining
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
    - **Direct hashing and pruning: DHP** (Park, Chen, Yu@SIGMOD'95)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)
- Closed itemset mining
  - **Pattern growth-based approach: CLOSET+** (Wang et al. @KDD'03)

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

TID	Items
1	acdef
2	abe
3	cefg
4	acdf

Let min_support = 2

a:3, ~~b:1,~~ c:3, d:2, e:3, f:3~~, g:1~~

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

d-proj. db: {acef, acf}

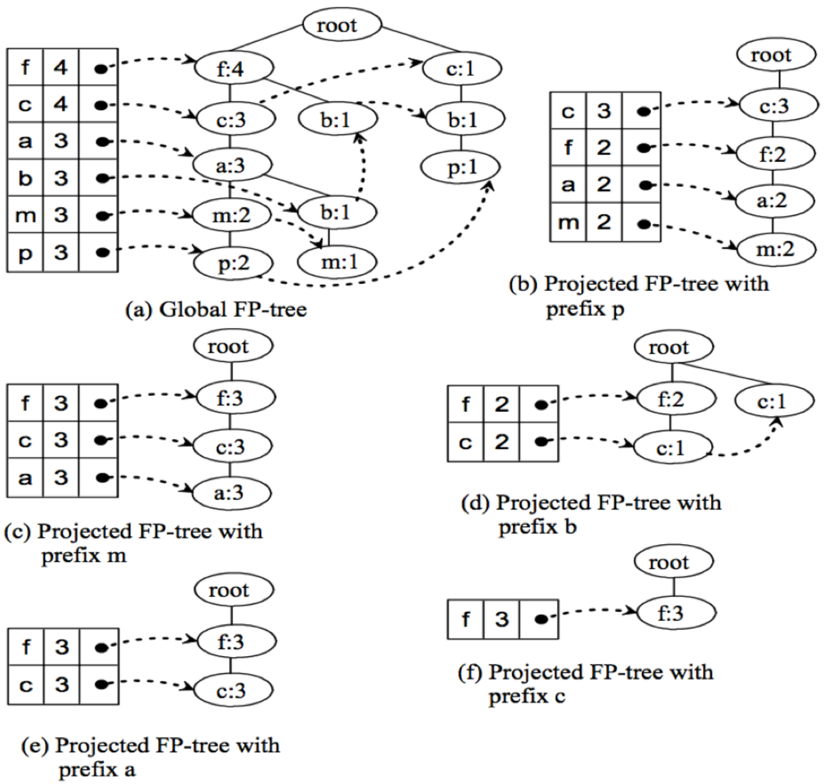
→ acfd-proj. db: {e}, thus we get: acfd:2

J. Wang, J. Han, and Jian Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In KDD'03. https://www.cs.umd.edu/class/spring2018/cmsc644/wang03closet.pdf

CLOSET+ (cont.)

- Many other tricks (but not detailed here), such as
  - Hybrid tree projection
    - **Bottom-up physical tree-projection**

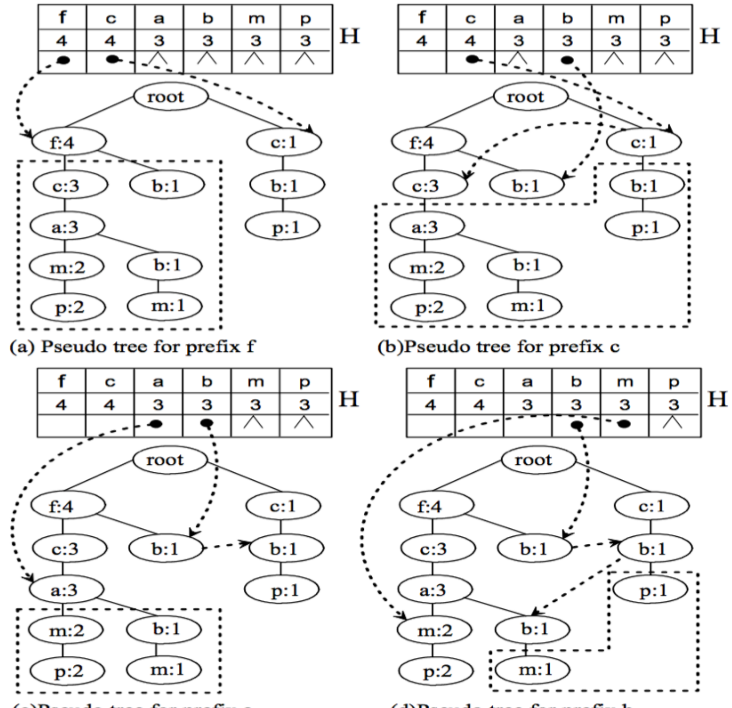
TID	Ordered, frequent items
100	*{f, c, a, m, p}*
200	*{f, c, a, b, m}*
300	*{f, b}*
400	*{c, b, p}*
500	*{f, c, a, m, p}*



CLOSET+ (cont.)

- Many other tricks (but not detailed here), such as
  - Hybrid tree projection
    - Bottom-up physical tree-projection
    - **Top-down pseudo tree-projection**

TID	Ordered, frequent items
100	*{f, c, a, m, p}*
200	*{f, c, a, b, m}*
300	*{f, b}*
400	*{c, b, p}*
500	*{f, c, a, m, p}*



CLOSET+ (cont.)

- 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**
    - **Two-level hash-indexed result tree**

Have Some Fun:
Frequent Pattern Mining *Research*

References (I) Basic Concepts

- R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in Proc. of SIGMOD'93
- R. J. Bayardo, “Efficiently mining long patterns from databases”, in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules”, in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery, 15(1): 55-86, 2007

References (II) Efficient Pattern Mining Methods

- 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