Chapter 6. Frequent Pattern Mining: Concepts and Apriori

Meng Jiang
Data Science

Definition

- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets

Definition (cont.)

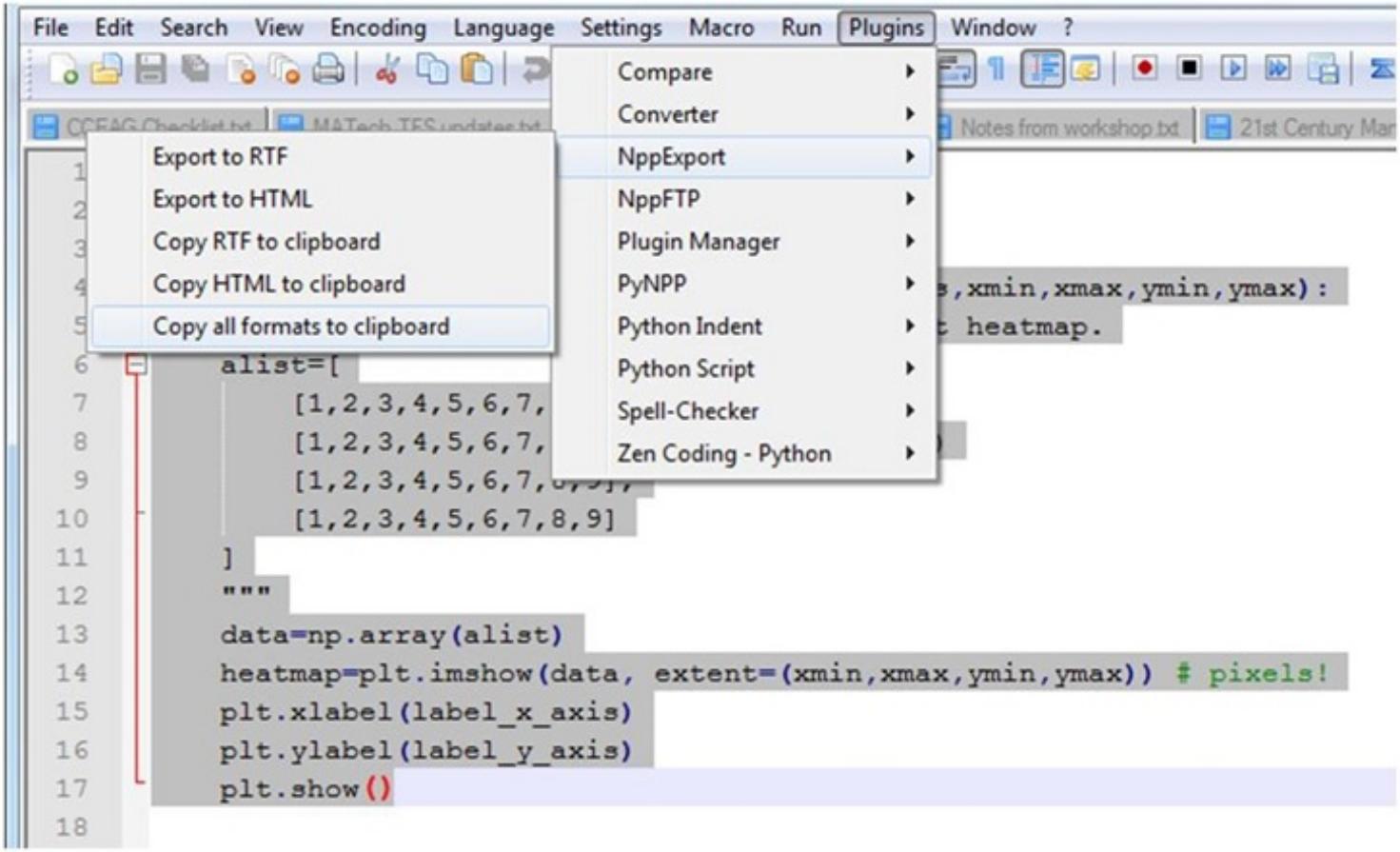
- Pattern discovery: Uncovering patterns from massive data
- Motivation examples:
  - What products were often purchased together?

Definition (cont.)

– What are the subsequent purchases after buying an iPad?

Definition (cont.)

– What code segments likely contain copy-and-paste bugs?



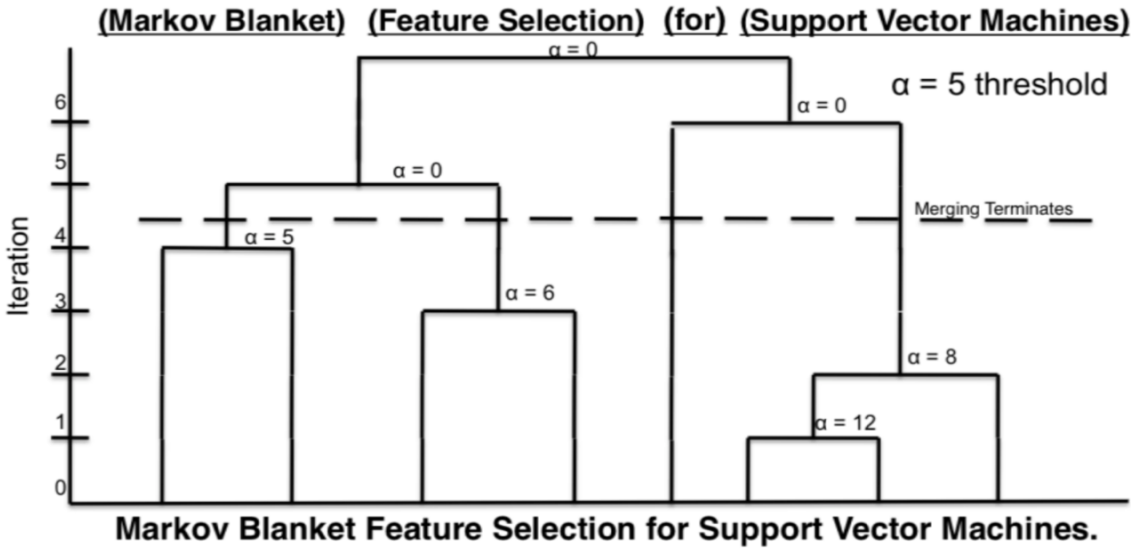
Definition (cont.)

– What word sequences likely form phrases in this corpus?

	Topic 1	*Topic 2*	*Topic 3*	*Topic 4*	*Topic 5*
unigrams	coffee	food	room	store	good
	ice	good	parking	shop	food
	cream	place	hotel	prices	place
	flavor	ordered	stay	find	burger
	egg	chicken	time	place	ordered
	chocolate	roll	nice	buy	fries
	breakfast	sushi	place	selection	chicken
	tea	restaurant	great	items	tacos
	cake	dish	area	love	cheese
	sweet	rice	pool	great	time
n-grams	ice cream	spring rolls	parking lot	grocery store	mexican food
	iced tea	food was good	front desk	great selection	chips and salsa
	french toast	fried rice	spring training	farmer's market	food was good
	hash browns	egg rolls	staying at the hotel	great prices	hot dog
	frozen yogurt	chinese food	dog park	parking lot	rice and beans
	eggs benedict	pad thai	room was clean	wal mart	sweet potato fries
	peanut butter	dim sum	pool area	shopping center	pretty good
	cup of coffee	thai food	great place	great place	carne asada
	iced coffee	pretty good	staff is friendly	prices are reasonable	mac and cheese
	scrambled eggs	lunch specials	free wifi	love this place	fish tacos

Definition (cont.)

– What word sequences likely form phrases in this corpus?



Markov Blanket Feature Selection for Support Vector Machines.

[Markov blanket] [feature selection] for [support vector machines]
[knowledge discovery] using [least squares] [support vector machine] [classifiers]
…[support vector] for [machine learning]…

Why Is It Important?

- Finding inherent regularities in a data set
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering

Frequent Patterns (Itemsets)

- Itemset: A set of one or more items
- k-itemset: $X = \{x_1, \ldots, x_k\}$
- (*absolute*) *support* (*count*) of X: Frequency or the number of occurrences of an itemset X

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

Frequent Patterns (Itemsets)

- (*relative*) *support*, *s:* The fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if the support of X is no less than a *minsup* threshold

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

Let *minsup = 50%*

Freq. 1-itemsets:

Beer: 3 (60%); Nuts: 3 (60%)

Diaper: 4 (80%); Eggs: 3 (60%)

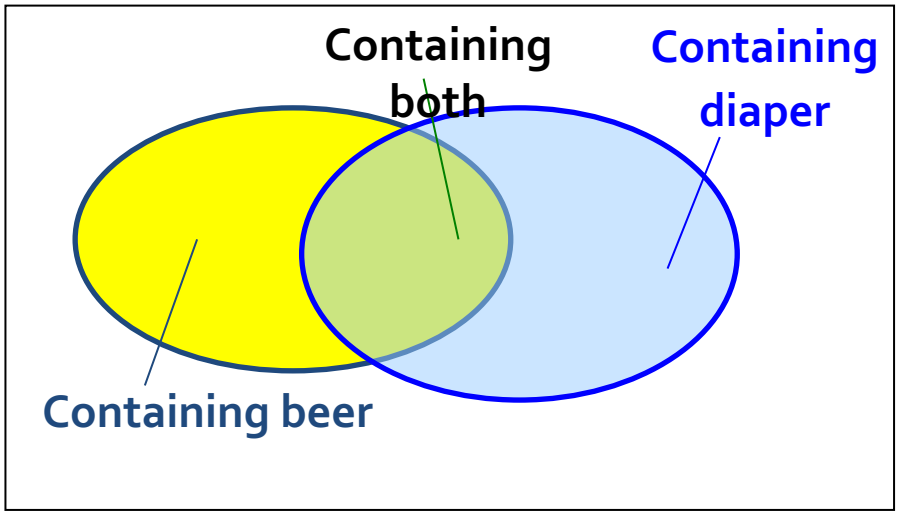
Freq. 2-itemsets:

{Beer, Diaper}: 3 (60%)

From Frequent Itemsets to Association Rules

- Association rules: $X \rightarrow Y$ (s, c)
  - Support, *s*: The probability that a transaction contains $X \cup Y$
  - Confidence, *c: The* conditional probability that a transaction containing X also contains *Y*
  - c = sup($X \cup Y$) / sup(X)

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



Note: Itemset: $X \cup Y$, a subtle notation!

From Frequent Itemsets to Association Rules

- **Association rule mining**: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence
- Frequent itemsets: Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3
- Association rules: Let *minconf = 50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

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

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Assuming (absolute) *minsup* = 1

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Assuming (absolute) *minsup* = 1

1-itemsets: $\{a_1\}$: 2, $\{a_2\}$: 2, …, $\{a_{50}\}$: 2, $\{a_{51}\}$: 1, …, $\{a_{100}\}$: 1,

2-itemsets: $\{a_1, a_2\}$: 2, …, $\{a_1, a_{50}\}$: 2, $\{a_1, a_{51}\}$: 1 …, …, $\{a_{99}, a_{100}\}$: 1, …

99-itemsets: $\{a_1, a_2, \ldots, a_{99}\}$: 1, …, $\{a_2, a_3, \ldots, a_{100}\}$: 1

100-itemset: $\{a_1, a_2, \ldots, a_{100}\}$: 1

- In total: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1$ sub-patterns!

A too huge set for any computer to compute or store!

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* $Y \supset X$, ***with the same support as X***
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ↄ X, ***with the same support as X***
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, \ldots, a_{50}\}$: 2"; P_2: "$\{a_1, \ldots, a_{100}\}$: 1"

Expressing Patterns in Compressed Form: Closed Patterns

- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to find from the result: "$\{a_2, \ldots, a_{40}\}$: 2" and "$\{a_5, a_{51}\}$: 1" are frequent patterns

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X, ~~with the same support as X~~
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X, ~~with the same support as X~~
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?
    - One: P: "$\{a_1, \ldots, a_{100}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- Max-pattern is a lossy compression!
  - We only know $\{a_1, \ldots, a_{40}\}$ is frequent
  - But we do not know the real support of $\{a_1, \ldots, a_{40}\}$, …, any more!
- Thus in many applications, mining closed-patterns is more desirable than mining max-patterns

The Downward Closure Property of Frequent Patterns: Apriori

- Observation: From $TDB_{1:}$ T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - We get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
  - Also, its subsets are all frequent: $\{a_1\}$, $\{a_2\}$, …, $\{a_{50}\}$, $\{a_1, a_2\}$, …, $\{a_1, \ldots, a_{49}\}$, …
  - There must be some hidden relationships among frequent patterns!

The Downward Closure Property of Frequent Patterns: Apriori

- The downward closure (also called "Apriori") property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - **Apriori: Any subset of a frequent itemset must be frequent.**

The Downward Closure Property of Frequent Patterns: Apriori

- **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

- 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)

Frequent Itemset 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)

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

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

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

1st scan

C_1

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

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

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

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

1st scan

C_1

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

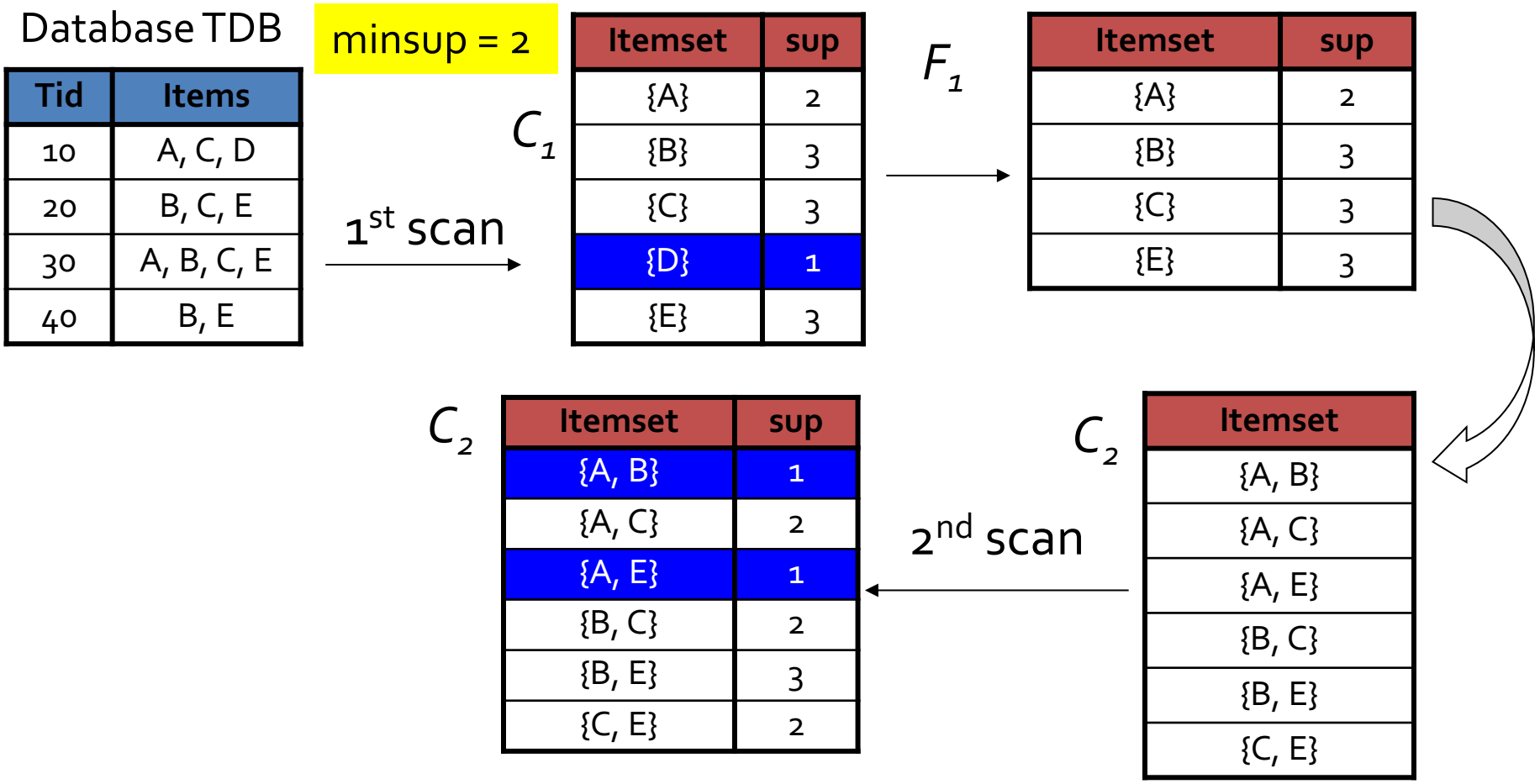
F_1

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

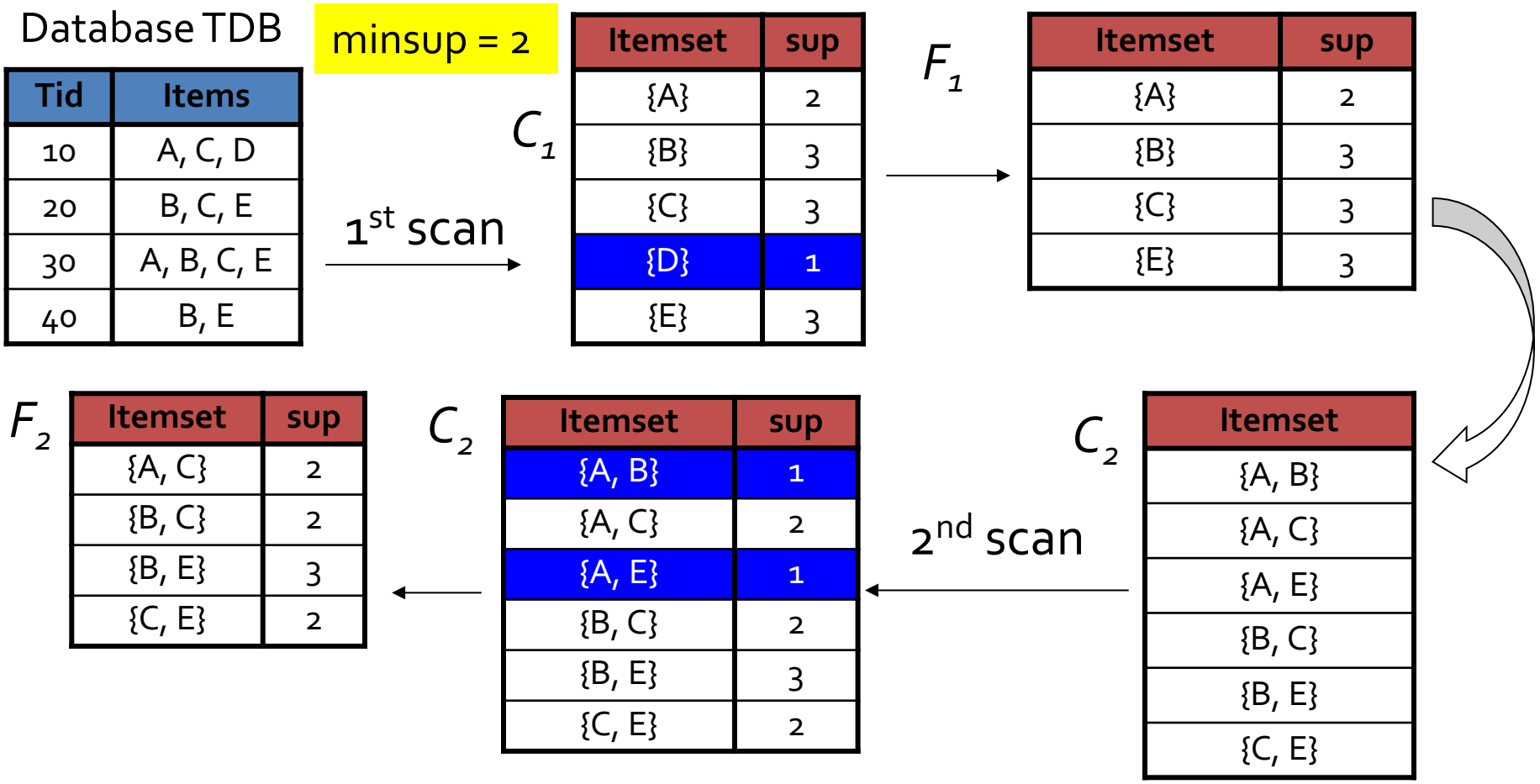
C_2

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

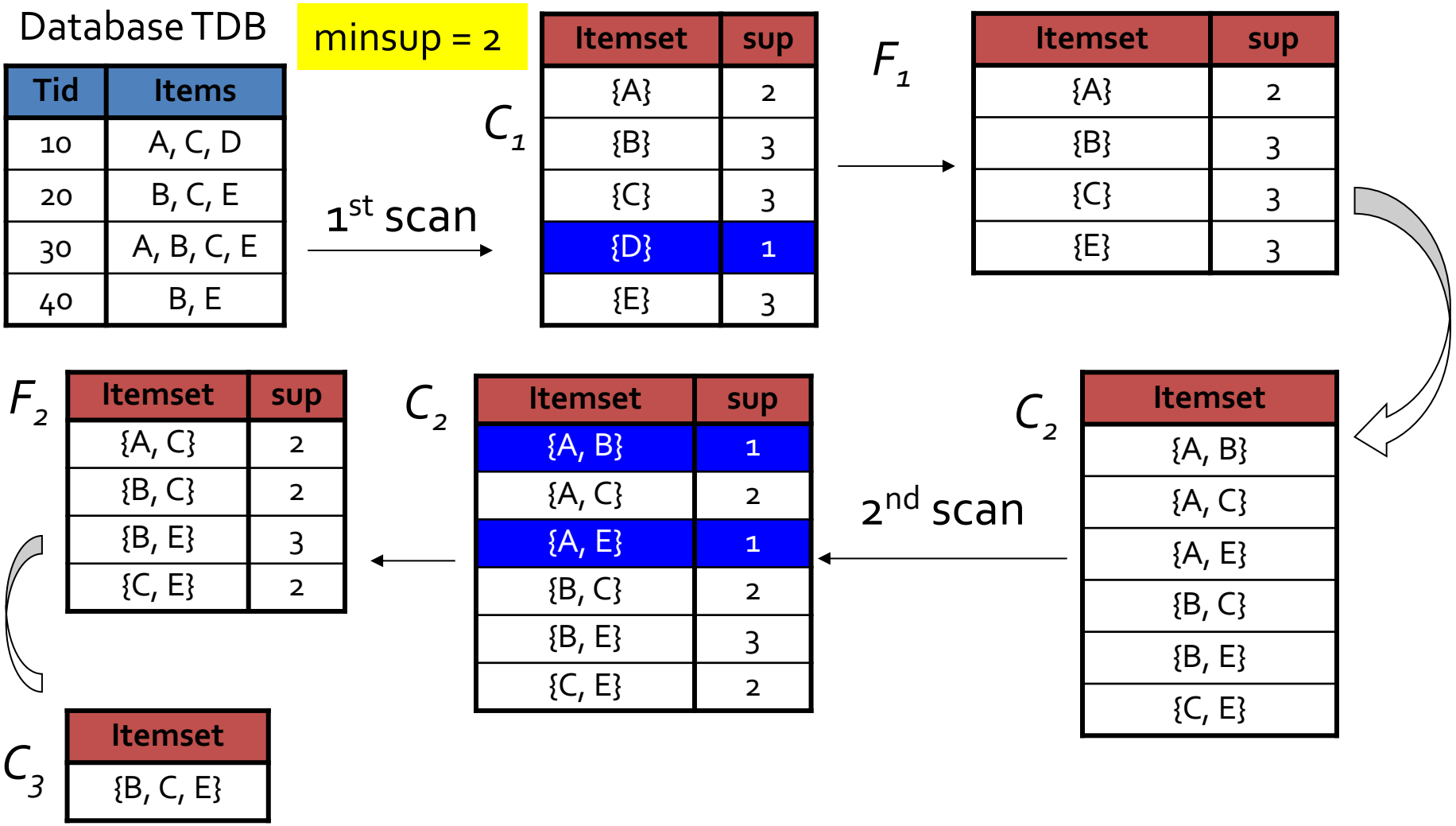
The Apriori Algorithm: An Example



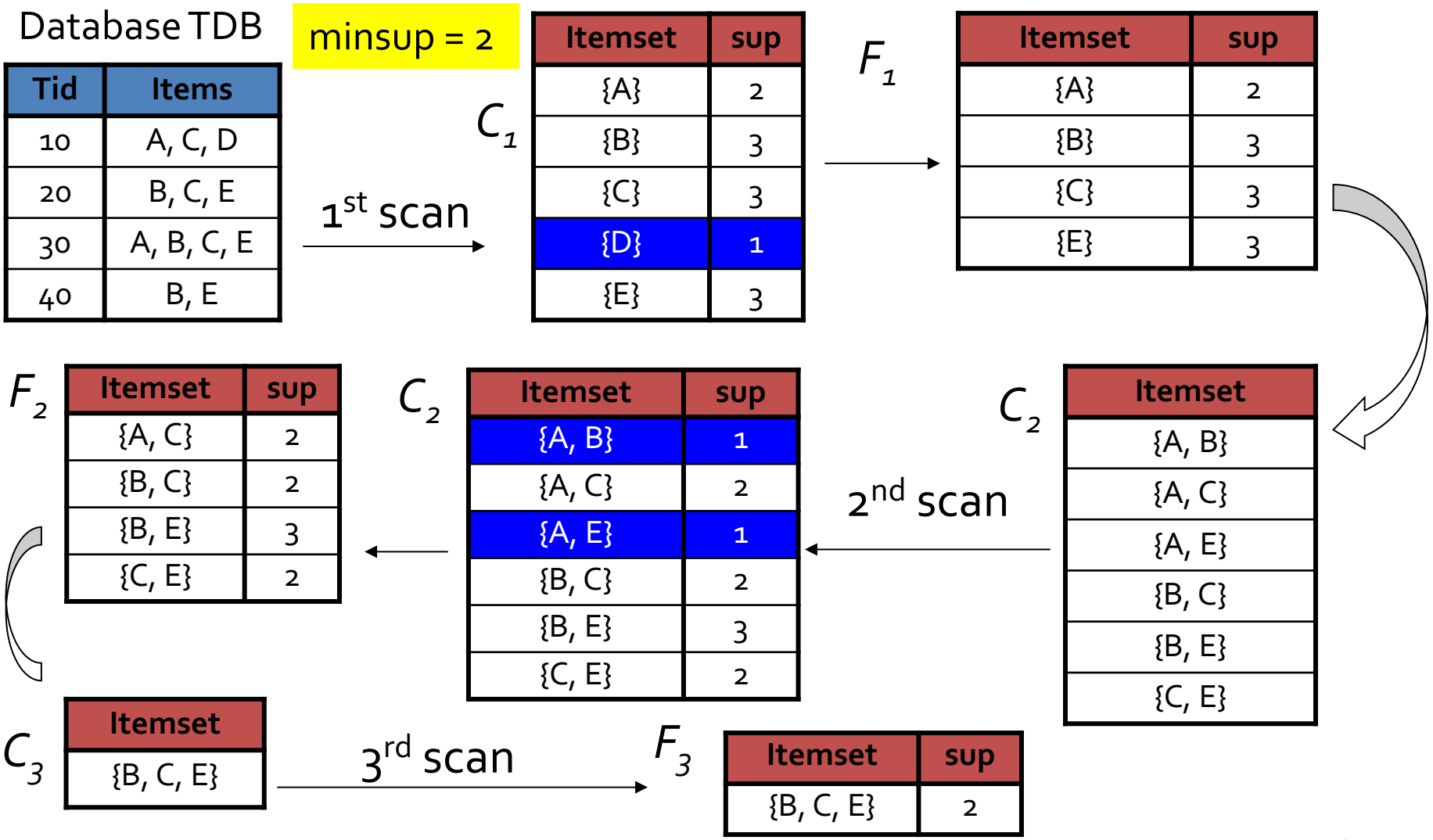
The Apriori Algorithm: An Example



The Apriori Algorithm: An Example

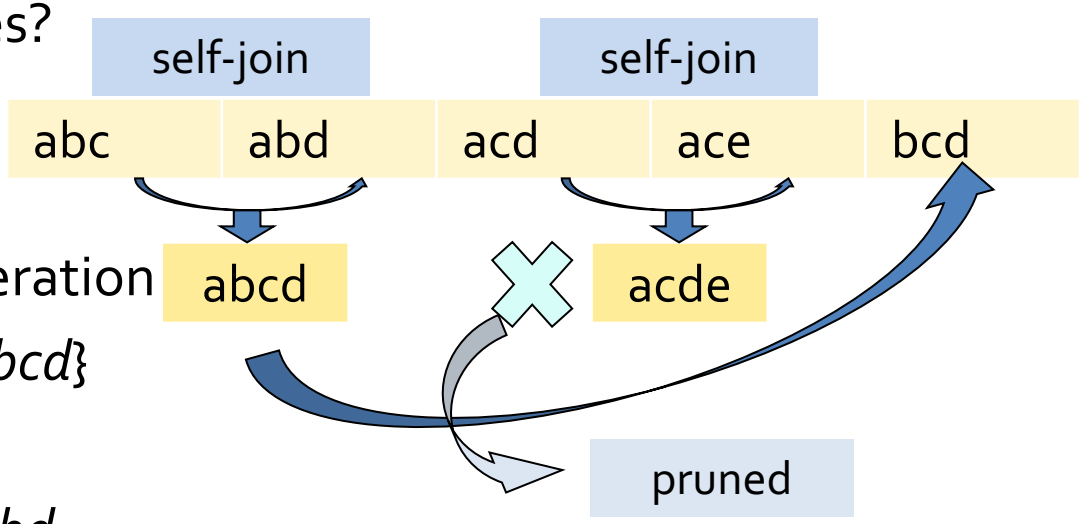


The Apriori Algorithm: An Example



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

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

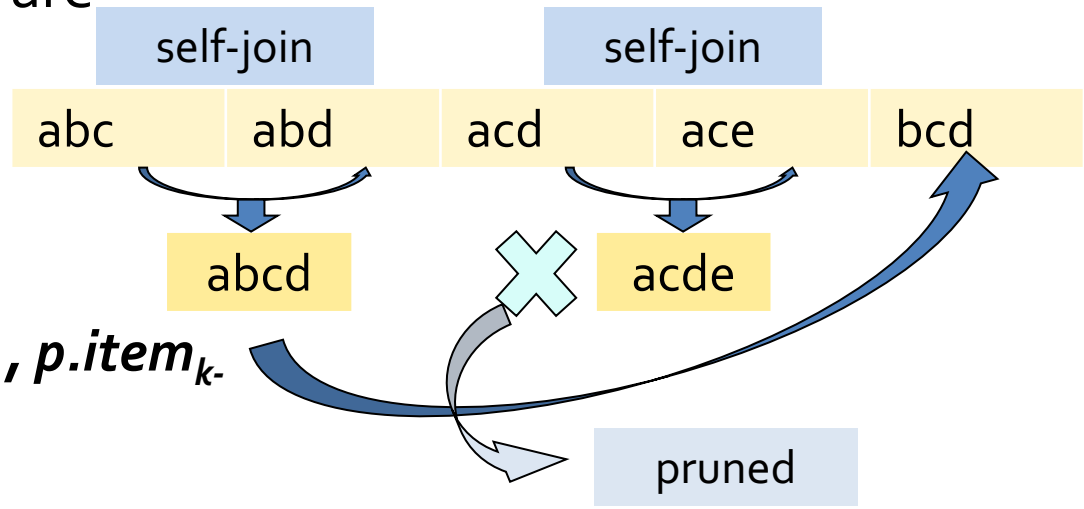
  where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

- 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

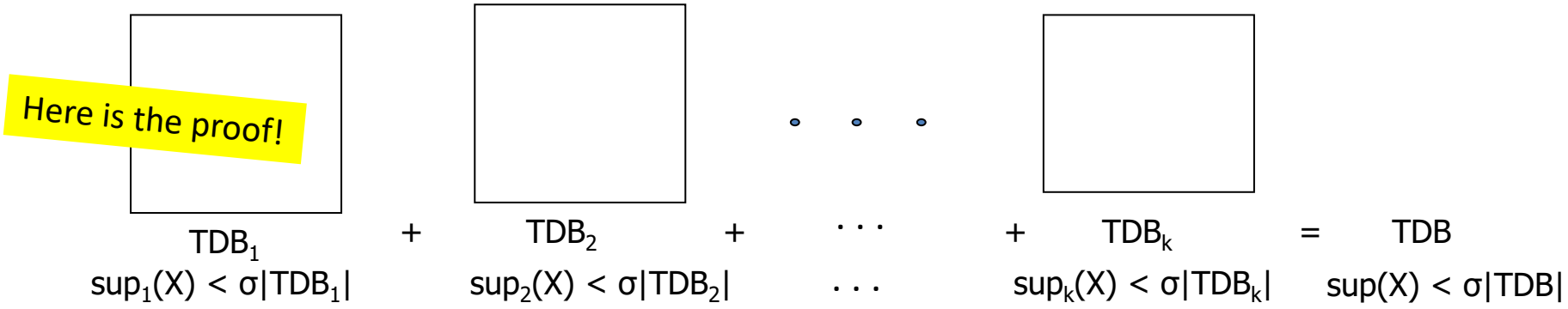


Apriori: Improvements and Alternatives

- 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)
  - Hypecube decomposition (e.g., LCM: Uno, et al., 2004)

Partitioning for Parallelization

- 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 and find local frequent patterns
  - Scan 2: Consolidate global frequent patterns

Discussion

- Do you believe in the association: Diapers → Beer?

References

- 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
- 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

References (cont.)

- 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
- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010