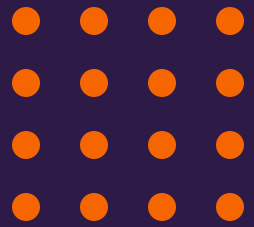
School of COMPUTING
Clemson University

Frequent Item Set Mining

Aaron J. Masino, PhD
Associate Professor, School of Computing



School of COMPUTING Clemson University

Outline

- Concept & motivation
- Frequent item sets & association rules
- Algorithms for finding frequent item sets

School of COMPUTING
Clemson University

Concept and motivation

School of COMPUTING Clemson University

Market basket analysis

- Transaction database of customer purchases

Transaction ID	Items purchased
0	butter, bread, milk, sugar
1	butter, flour, milk, sugar
2	butter, eggs, milk, salt
3	eggs
4	butter, flour, milk, salt, sugar

- Identify items that are purchased frequently together
- Create rules that indicate if certain items are purchased then certain other items are likely to be purchased
- Applications
  - Improve store layout
  - Focused marketing / add-on sales
  - Inventory management

Items	Frequency
butter	4
milk	4
butter, milk	4
sugar	3
butter, sugar	3
…	

Going beyond market baskets

- Plagiarism detection
  - Sentences are “baskets”
  - Documents are “items”
  - Two (or more) documents appear together frequently (i.e., are “in” multiple sentences) may indicate one (or more) are plagiarized
- Adverse drug interactions
  - Patients are “baskets”
  - Drugs and effects are “items”
  - Drugs that appear together with a given effect frequently may indicate an adverse reaction – this requires that “absence” of the effect is also recorded

Market basket generalization and formalism

- Items $I = \{i_0, i_1, \cdots, i_m\}$ - the set of all items
- Itemset $X \subseteq I$ – a subset of I
  - There are 2^m possible itemsets if there are m distinct items
- Transaction T a collection of items from I (*basket)*
- Database D : a collection of transactions
- For itemset X, define the support, $S(X) = |\{T \in D | X \subseteq T\}|$ - the number of transactions in the database that contain the itemset

School of COMPUTING
Clemson University

Frequent item sets
Association rules

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Database

Transactions

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Milk

$t_s = 3$

X	Support	Frequent
…		

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Transactions

Database

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Milk

$t_s = 3$

X	Support	Frequent
Bread	3	yes
…		

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Transactions

Database

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Milk

$t_s = 3$

X	Support	Frequent
Bread	3	yes
Coke	3	yes
…		

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Transactions

Database

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Milk

$t_s = 3$

X	Support	Frequent
Bread	3	yes
Coke	3	yes
Milk	4	yes
…		

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Transactions

Database

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Milk

$t_s = 3$

X	Support	Frequent
Bread	3	yes
Coke	3	yes
Milk	4	yes
Diaper	2	no
Beer, Bread	2	no
…		

Frequent itemsets

An itemset, $X \subseteq I$, is *frequent* if and only if (iff) it is contained in at least t_s transactions of a database D, that is iff

$$S(X) \geq t_s$$

where t_s is called the *support threshold*

Transactions

Database

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

$t_s = 3$

X	Support	Frequent
Bread	3	yes
Coke	3	yes
Milk	4	yes
Diaper	2	no
Beer, Bread	2	no
Coke, Milk	3	yes
…		

Association rules

- An *association rule* is an *if-then* rule generated from frequent itemsets

$$r : X \Longrightarrow Y$$

(read as X *implies* Y, or *if* X *then* Y) where X and Y are itemsets

  - X is the *antecedent*
  - Y is the *consequent*
- **Interpretation:** If a transaction (basket) contains the items in X then it is <u>likely</u> to contain the items in Y
- **Support:** the minimum of the support of the itemsets X and Y

$$S(r) = min[S(X), S(Y)]$$

School of COMPUTING Clemson University

Association rule confidence

- **Confidence:** The probability that the *consequent* follows if the *antecedent* is true for a given rule, r, defined by:

$$C(r) = \frac{S(X \cup Y)}{S(X)}$$

Association rule confidence

- **Confidence:** The probability that the *consequent* follows if the *antecedent* is true for a given rule, r, defined by:

$$C(r) = \frac{S(X \cup Y)}{S(X)}$$

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- Consider the rule

$$r: \{milk, coke\} \Longrightarrow \{bread\}$$

  - $X = \{milk, coke\}$ (antecedent)
  - $Y = \{bread\}$ consequent
  - $X \cup Y = \{milk, coke, bread\}$

$$C(r) = \frac{S(X \cup Y)}{S(X)} = \frac{1}{3}$$

Association rule confidence

- **Confidence:** The probability that the *consequent* follows if the *antecedent* is true for a given rule, r, defined by:

$$C(r) = \frac{S(X \cup Y)}{S(X)}$$

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- Consider the rule

$$r: \{milk, coke\} \Longrightarrow \{diaper\}$$

  - $X = \{milk, coke\}$ (antecedent)
  - $Y = \{diaper\}$ consequent
  - $X \cup Y = \{milk, coke, diaper\}$

$$C(r) = \frac{S(X \cup Y)}{S(X)} = \frac{2}{3}$$

Association rule interestingness

- Some rules will have high confidence even though antecedent is not a result of the consequent (correlation does NOT equal causality)
- For example, milk and bread may occur together frequently simply because they are common grocery items not because purchasing milk directly influences the purchase of bread
- Define the *interest*, $I(r)$, of rule, $r : X \Longrightarrow Y$ by

$$I(r) = \left| C(r) - \frac{S(Y)}{|D|} \right|$$

  where $|D|$ is the number of transactions (baskets) in the database
- Threshold for *interesting rules* typically set to 0.5

Association rule interestingness

- Define the *interest*, $I(r)$, of rule, $r : X \Longrightarrow Y$ by

$$I(r) = \left| C(r) - \frac{S(Y)}{|D|} \right|$$

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- Consider the rule

$$r: \{milk, coke\} \Longrightarrow \{bread\}$$

- $C(r) = \frac{S(X \cup Y)}{S(X)} = \frac{1}{3}$
- $S(Y) = 3$
- $|D| = 5$
- $I(r) = \left| \frac{1}{3} - \frac{3}{5} \right| \approx 0.27$

School of COMPUTING Clemson University

Association rule mining

- Goal: find all association rules with support greater than t_s and confidence greater than t_c
- General approach
  1. Find all frequent itemsets above threshold – this is the hard part
  2. Rule generation
     a. For every frequent itemset, I, do
        i. For every subset, A, in I, generate rule $A \Longrightarrow I \backslash A$ (note since I is frequent A must be frequent and above threshold
  3. Output rules with confidence above threshold

School of COMPUTING Clemson University

Association rule mining example

$B_1 = \{m, c, b\}$ $B_2 = \{m, p, j\}$

$B_3 = \{m, c, b, n\}$ $B_4 = \{c, j\}$

$B_5 = \{m, p, b\}$ $B_6 = \{m, c, b, j\}$

$B_7 = \{c, b, j\}$ $B_8 = \{b, c\}$

- Support threshold $s = 3$, confidence $c = 0.75$
- Step 1) Find frequent itemsets:
  - {b,m} {b,c} {c,m} {c,j} {m,c,b}
- Step 2) Generate rules:
  - ~~b → m: c=4/6~~
  - m → b: c=4/5

b → c: c=5/6

...

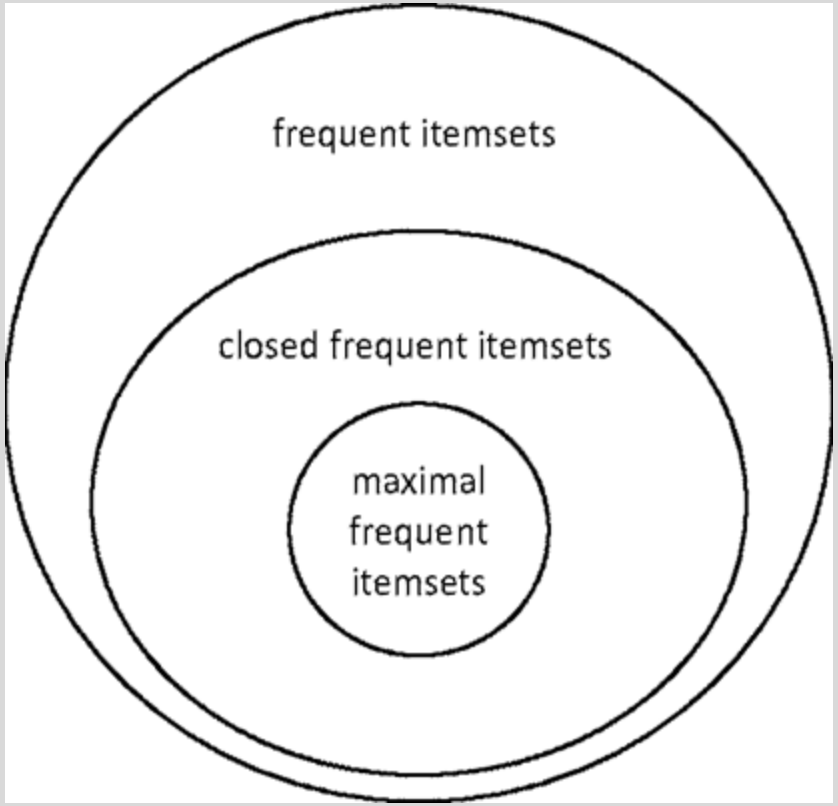
~~b,c → m: c=3/5~~

b,m → c: c=3/4

~~b → c,m: c=3/6~~

Pruning association rules

- Closed frequent itemset: A frequent itemset is closed if no immediate superset has the same support
- Maximal frequent itemsets: A frequent itemset is *maximal* if no immediate superset (addition of 1 item) is frequent
- Can choose to keep only rules that include maximal or closed itemsets



School of COMPUTING Clemson University

Association rule maximal & closed examples

	Support	Maximal (s=3)	Closed
A	4	No	No
B	5	No	Yes
C	3	No	No
AB	4	Yes	Yes
AC	2	No	No
BC	3	Yes	Yes
ABC	2	No	Yes

Consider $t_s = 3$

A and C are frequent but have immediate supersets with the same support

Not frequent

Not frequent

Association rule maximal & closed examples

	Support	Maximal (s=3)	Closed
A	4	No	No
B	5	No	Yes
C	3	No	No
AB	4	Yes	Yes
AC	2	No	No
BC	3	Yes	Yes
ABC	2	No	Yes

Consider $t_s = 3$

A, B and C are frequent but have immediate supersets which are also frequent.

School of COMPUTING
Clemson University

Finding Frequent Itemsets

Naïve approach to frequent itemset identification

- Create a data structure (e.g., a Python dictionary) to hold distinct itemsets and their counts
- Read the stored transactions in the database and compute and store the itemset counts
- How much memory do we need for the data structure?
  - For a database with m unique items, there are 2^m possible itemsets
  - Let's just consider itemsets with 2 items of which there are $m(m-1)/2$ (why divide by 2? ba = ab in this context)
  - Assume 4-byte integers to store the count for each itemset
  - Walmart $m \approx 10^5$ require 20 GB of memory
  - Amazon $m \approx 10^7$ requires 200 TB of memory

A-Priori Algorithm

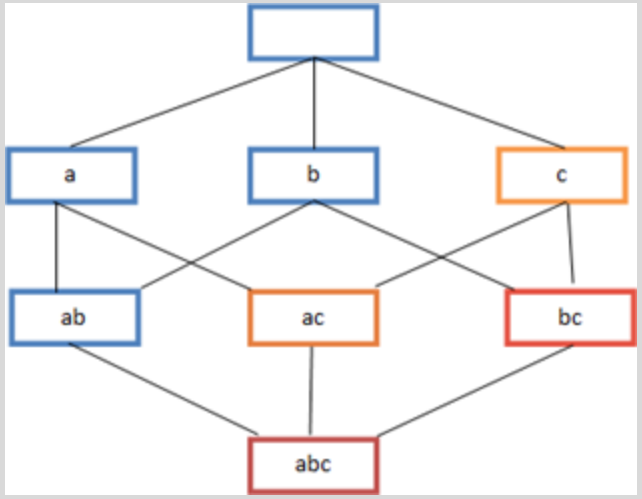
A two-pass approach called A-Priori limits the need for main memory

Key idea: monotonicity

- If itemset I appears at least s times, so does every subset J of I

Contrapositive for pairs:

- If itemset I does not appear in s transactions (baskets), then no larger itemset that contains I can appear in s transactions (baskets)

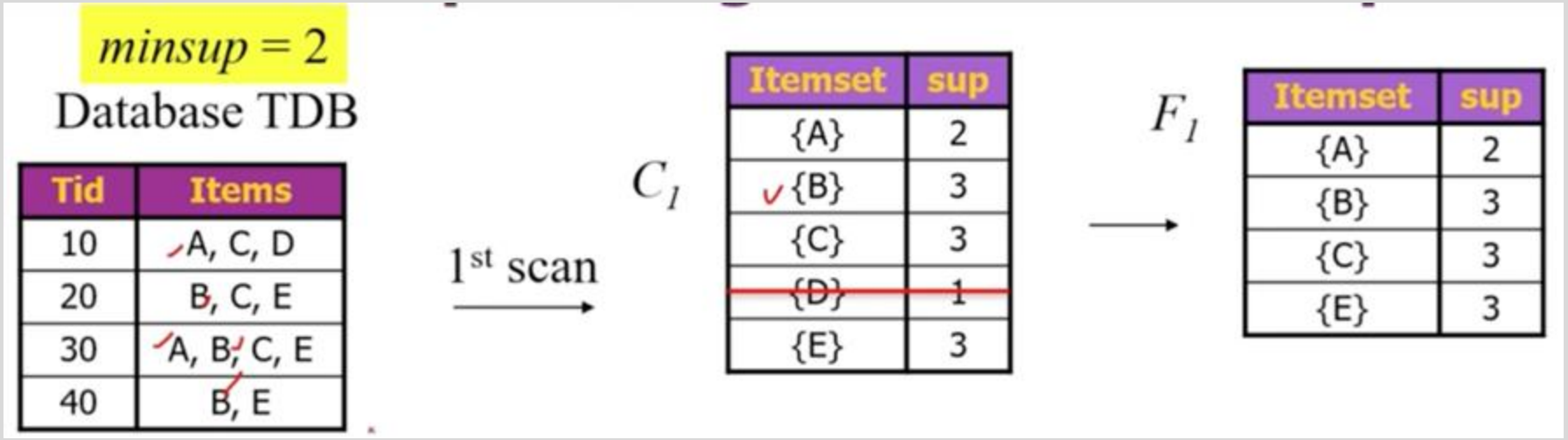


A-Priori Algorithm – pass one

Pass 1: Read transactions and count in main memory the number of occurrences of each individual item

- Requires only memory proportional to number of individual items

Items that appear $\geq s$ times are the frequent items

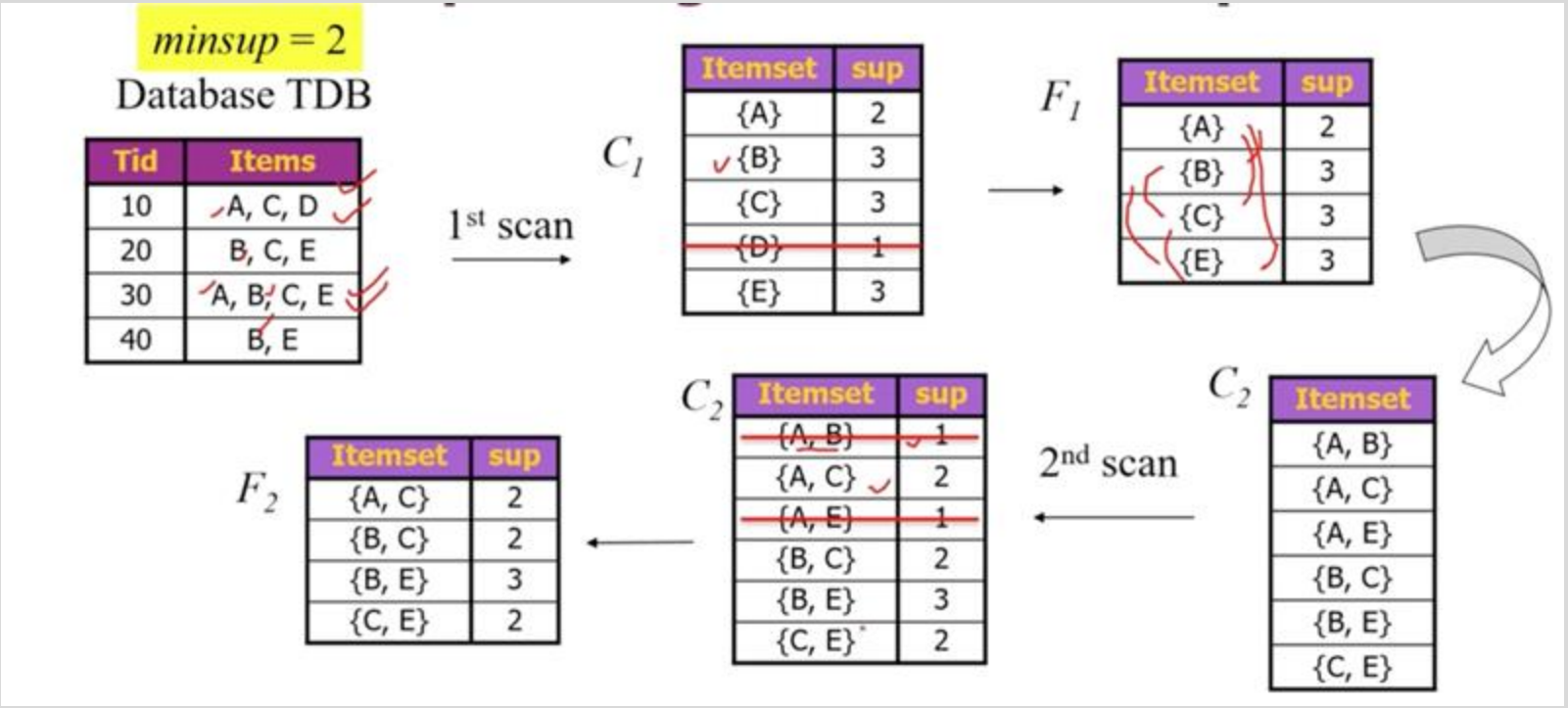


https://www.youtube.com/watch?v=VBYNmolsa1M

A-Priori Algorithm – pass two

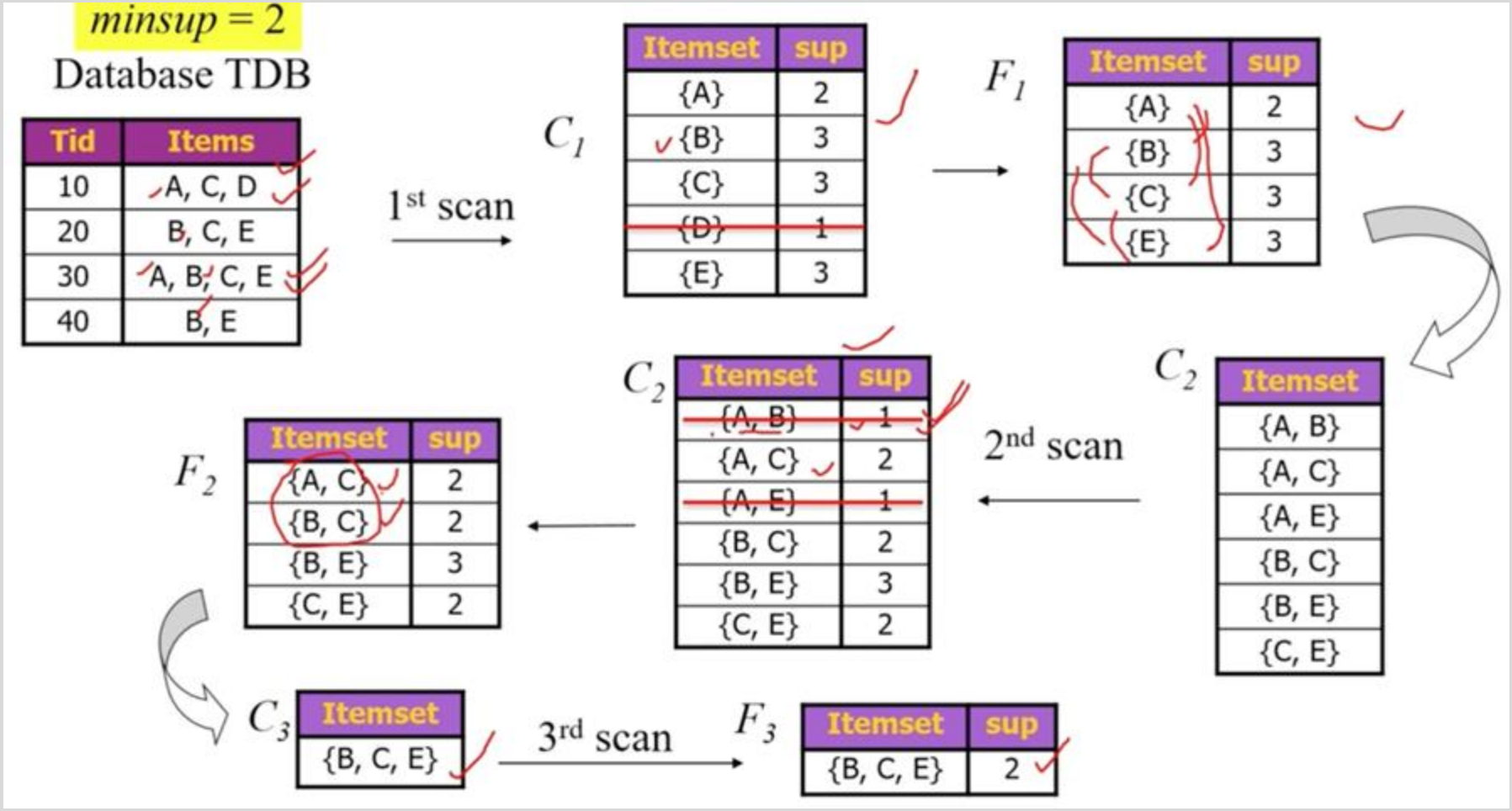
Pass 2: Read baskets again and keep track of the count of only those pairs where both elements are frequent (from Pass 1)

- Requires memory (for counts) **proportional to square of the number of frequent items** (not the square of total number of items)
- Plus a list of the frequent items (so you know what must be counted)



A-Priori Algorithm – pass three and beyond

Note that we skip items that are not frequent for C3 such as A,B and A,E



PCY (Park-Chen-Yu) Algorithm

Observation:

- In pass 1 of A-Priori, most memory is idle
- We store only individual item counts
- Can we use the idle memory to reduce memory required in pass 2?

Pass 1 of PCY: In addition to item counts, maintain a hash table with as many buckets as fit in memory

Keep a count for each bucket into which pairs of items are hashed

- For each bucket just keep the count, not the actual pairs that hash to the bucket!

School of COMPUTING Clemson University

PCY Algorithm – First Pass

```
FOR (each basket) :
    FOR (each item in the basket):
        add 1 to item's count;

    FOR (each pair of items in the basket):
        hash the pair to a bucket;
        add 1 to the count for that bucket;
```

New in PCY

PCY Hash Function

- Assume each item has a unique integer identifier
- Define hash function as $(i * j)\%10$
- What happens in first pass?

Transactions	Pairs	Hash Values
1, 2, 3	(1,2), (2,3), (1,3)	2, 6, 3
1, 3	(1, 3)	3
1, 2, 10	(1,2), (1,10), (2,10)	2, 0, 0
1, 2, 3, 10	(1,2),(1,3),(1,10), (2,3), (2,10), (3,10)	2, 3, 0, 6, 0, 0

Hash Bucket	Count
0	5
2	3
3	3
6	2

PCY second pass

- Assume we require support of s
- Convert hash bucket to bit vector
  - 1 indicates count $\geq s$, 0 otherwise
  - Requires 1/32 as much memory as storing counts
- On second pass, any examined pair that maps to a bucket with a 0 bit is discarded
  - In this example, pairs (1,3), (2,3), (2,8), (1,8) ignored
  - Just because a pair maps to a bit 1 bucket does NOT mean it is frequent
  - Frequent pairs: (1,2), (1,10)

$s = 3$

Hash Bucket	Count	Bit
0	7	1
2	3	1
3	2	0
6	2	0
8	1	0

Transactions	Pairs	Hash Values
1, 2, 3	(1,2), (2,3), (1,3)	2, 6, 3
1, 3, 10	(1, 3), (1,10), (3,10)	3, 0, 0
1, 2, 10	(1,2), (1,10), (2,10)	2, 0, 0
1, 2, 8, 10	(1,2),(1,8),(1,10), (2,8), (2,10), (8,10)	2, 8, 0, 6, 0, 0

Many other algorithms

- Random sampling
- SON
- Toivonen
- …
- https://link.springer.com/article/10.1007/s10462-018-9629-z