Data Mining & Data Warehouse

Dr. Raed Ibraheem Hamed

University of Human Development,
College of Science and Technology
Department of Information Technology

2016 – 2017

Road map

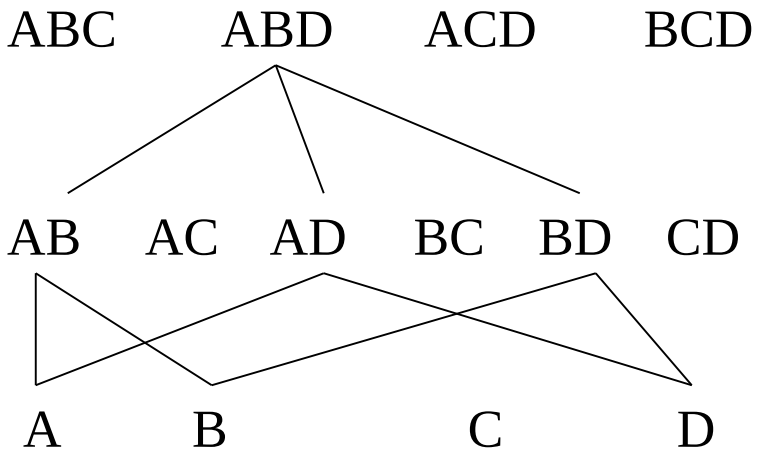
- The Apriori algorithm
- Step 1: Mining all frequent itemsets
- Definition of Apriori Algorithm
- Definition (contd.)
- Steps to Perform Apriori Algorithm
- The Apriori Algorithm — Example-1
- The Apriori Algorithm — Example-2
- **Step 1**: Generating 1-itemset Frequent Pattern
- **Step 2**: Generating 2-itemset Frequent Pattern
- **Step 3**: Generating 3-itemset Frequent Pattern
- **Step 4**: Generating 4-itemset Frequent Pattern
- **Step 5:** Generating Association Rules from Frequent Itemsets

The Apriori algorithm Key Concepts :

1. Frequent Itemsets: The sets of item which has minimum support (denoted by **Li** for ith-Itemset).
2. Apriori Property: Any subset of frequent itemset must be frequent.
3. Join Operation: To find **Lk** , a set of candidate k-itemsets is generated by joining Lk-1 with itself.

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: any subsets of a frequent itemset are also frequent itemsets



Definition of Apriori Algorithm

- In computer science and data mining, **Apriori** is a classic algorithm for learning association rules.
- Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).
- The algorithm attempts to find subsets which are common to at least a minimum number C of the itemsets.

Definition (contd.)

- Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as **candidate generation**, and groups of candidates are tested against the data.

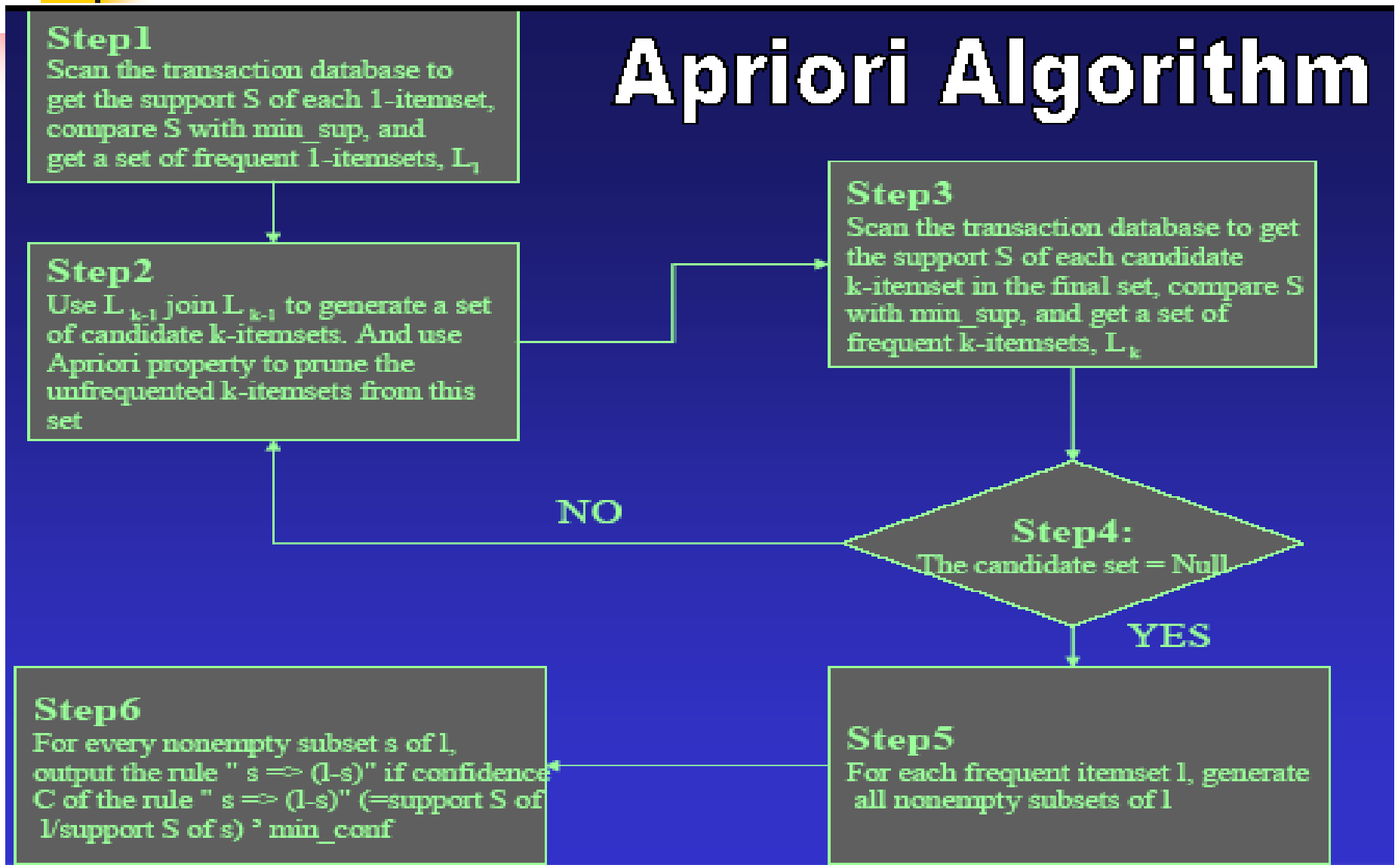
- The algorithm terminates when no further successful extensions are found.

Apriori Algorithm

Uses a ***Level-wise search***, where ***k*-itemsets** (An itemset that contains *k* items is a ***k-itemset***) are used to explore (*k*+1)-itemsets, to mine frequent itemsets from transactional database for Boolean association rules.

First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the set of frequent 2-itemsets, which is used to fine L3, and so on, until no more frequent *k*-itemsets can be found.

Steps to Perform Apriori Algorithm



The Apriori Algorithm: Example

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

- Consider a database, D , consisting of 9 transactions.
- Suppose min.support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern

Scan D for count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

- In the first iteration of the algorithm, each item is a member of the set of candidate.
- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets **satisfying minimum support**.

Step 2: Generating 2-itemset Frequent Pattern

Generate C_2 candidates from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

Compare candidate support count with minimum support count

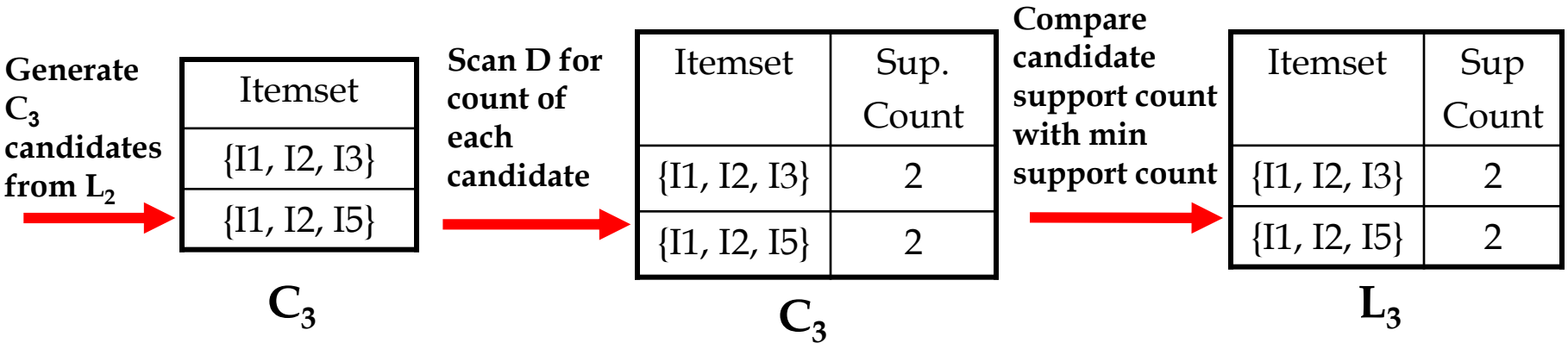
Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 2: Generating 2-itemset Frequent Pattern [Cont.]

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses L_1 *Join* L_1 to generate a candidate set of 2-itemsets, C_2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.
- Note: We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern



- The generation of the set of candidate 3-itemsets, C_3 , involves use of the Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 3: Generating 3-itemset Frequent Pattern [Cont.]

- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that **four candidates cannot possibly be frequent**. How ?
- For example , lets take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, We will keep {I1, I2, I3} in C_3.
- Lets take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L_2 and hence it is not frequent violating Apriori Property. Thus We will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L_3, consisting of those candidates 3-itemsets in C_3 having minimum support.

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.

- Thus, $C_4 = \varphi$, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.

- What's Next ?

  These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

- Procedure:
  - For each frequent itemset "***l***", generate all nonempty subsets of ***l***.
  - For every nonempty subset ***s*** of ***l***, output the rule "**s → (l-s)**" if **support_count(l) / support_count(s) >= min_conf** where min_conf is minimum confidence threshold.

- Back To Example:

  We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
  - Lets take ***l*** = {I1,I2,I5}.
  - Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

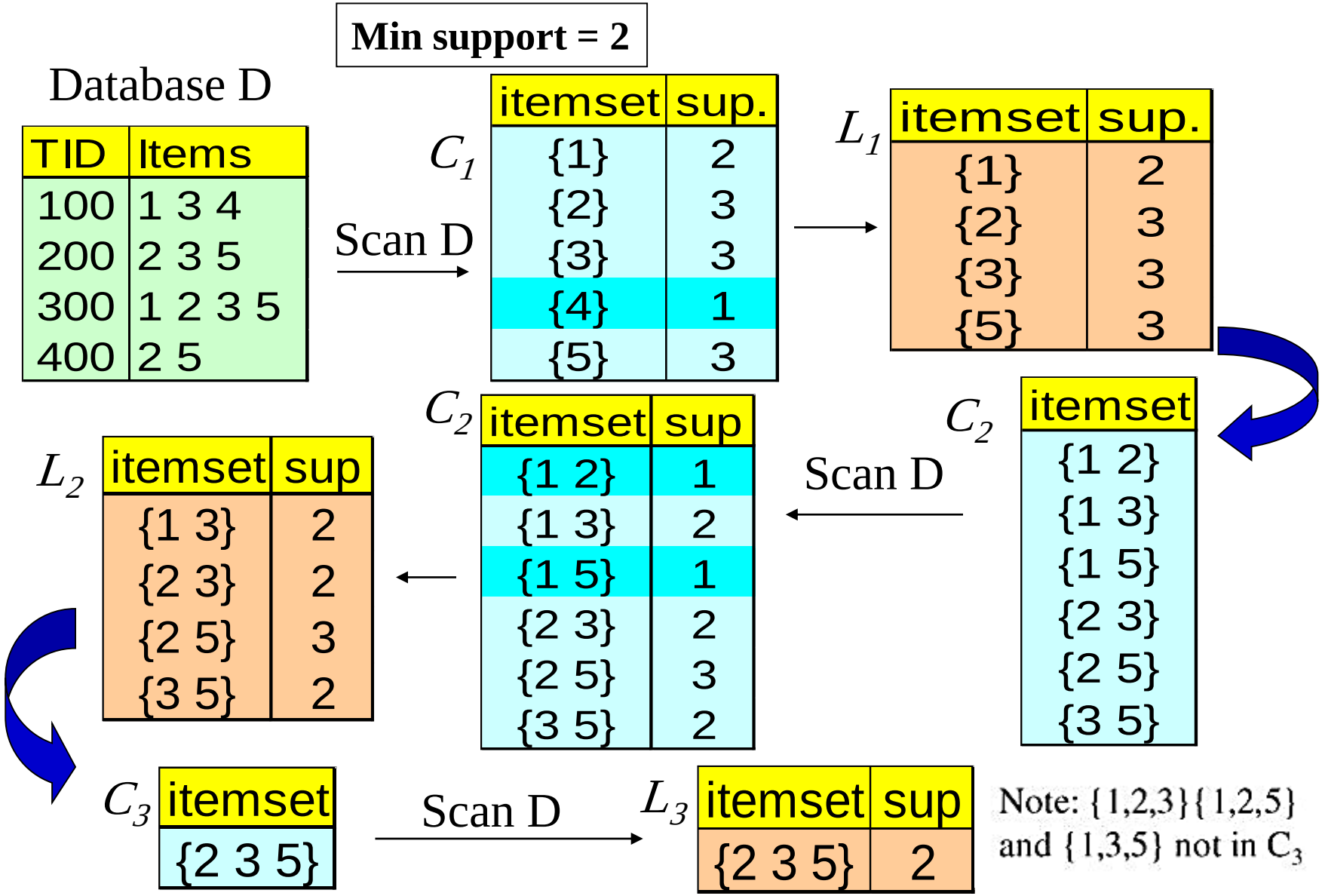
- Let minimum confidence threshold is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.
  - R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - R3: I2 ^ I5 → I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

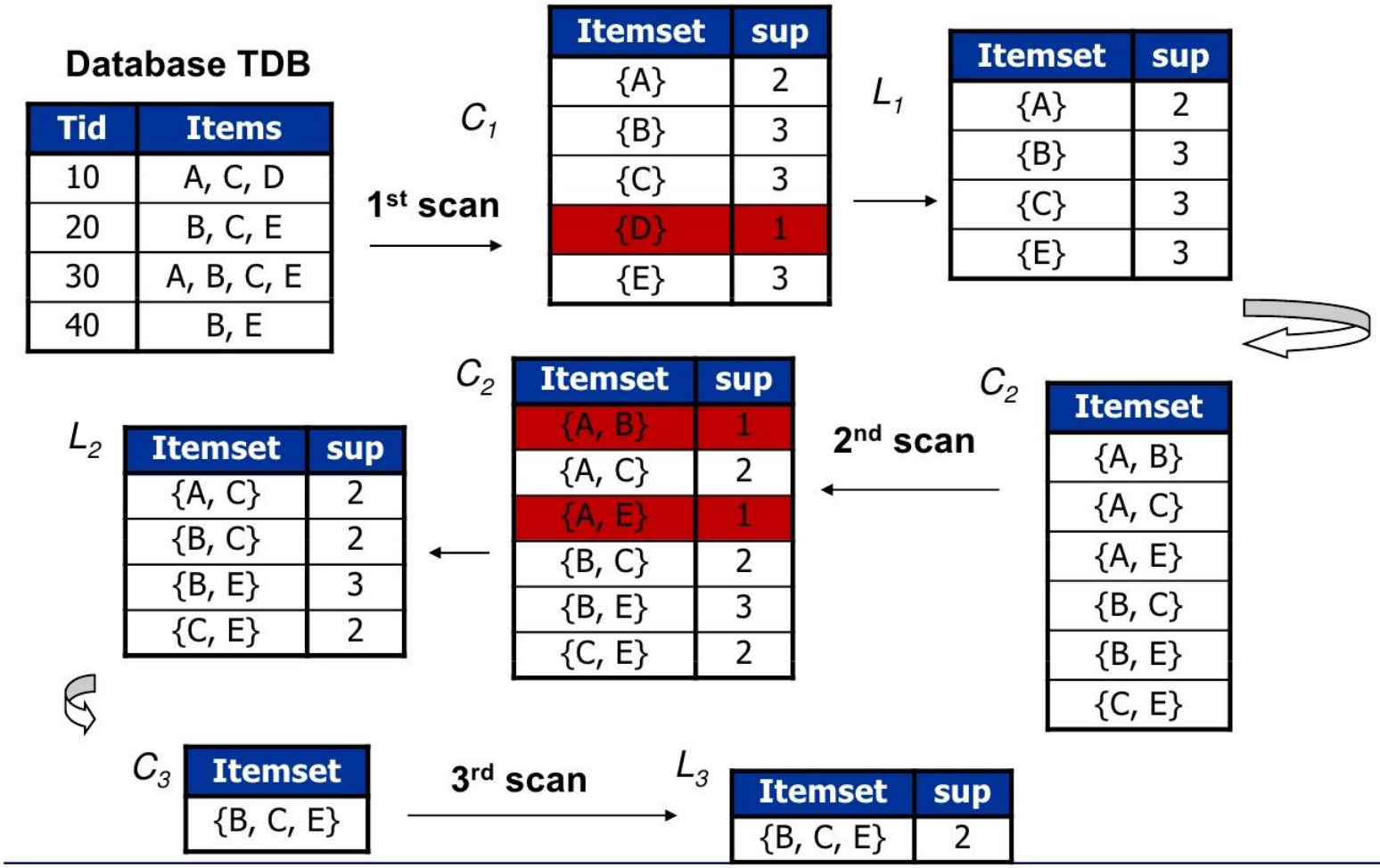
- R4: I1 → I2 ^ I5
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- R5: I2 → I1 ^ I5
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- R6: I5 → I1 ^ I2
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.

In this way, We have found three strong association rules.

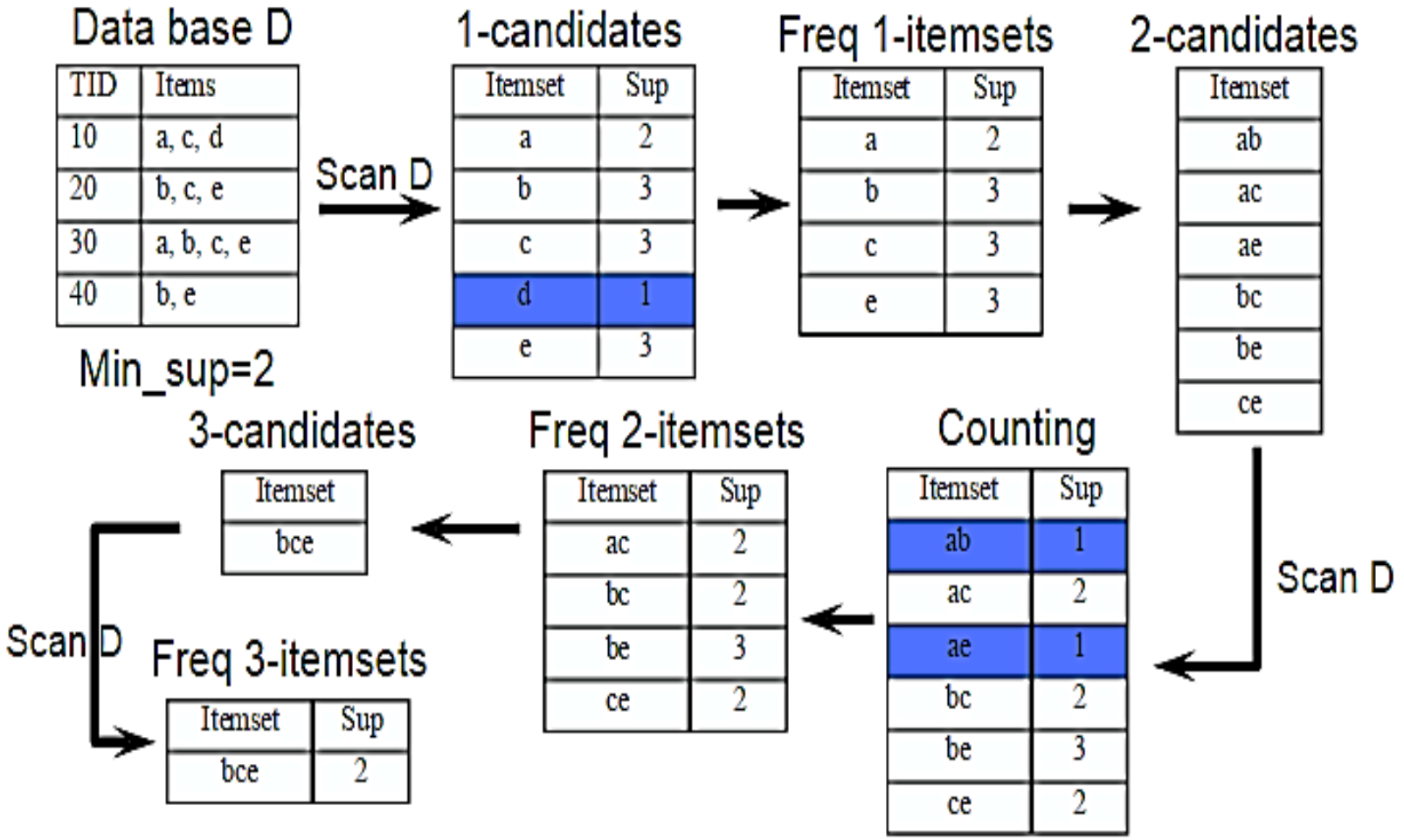
The Apriori Algorithm — Example



Example of Apriori Run



Apriori algorithm example



Thank you