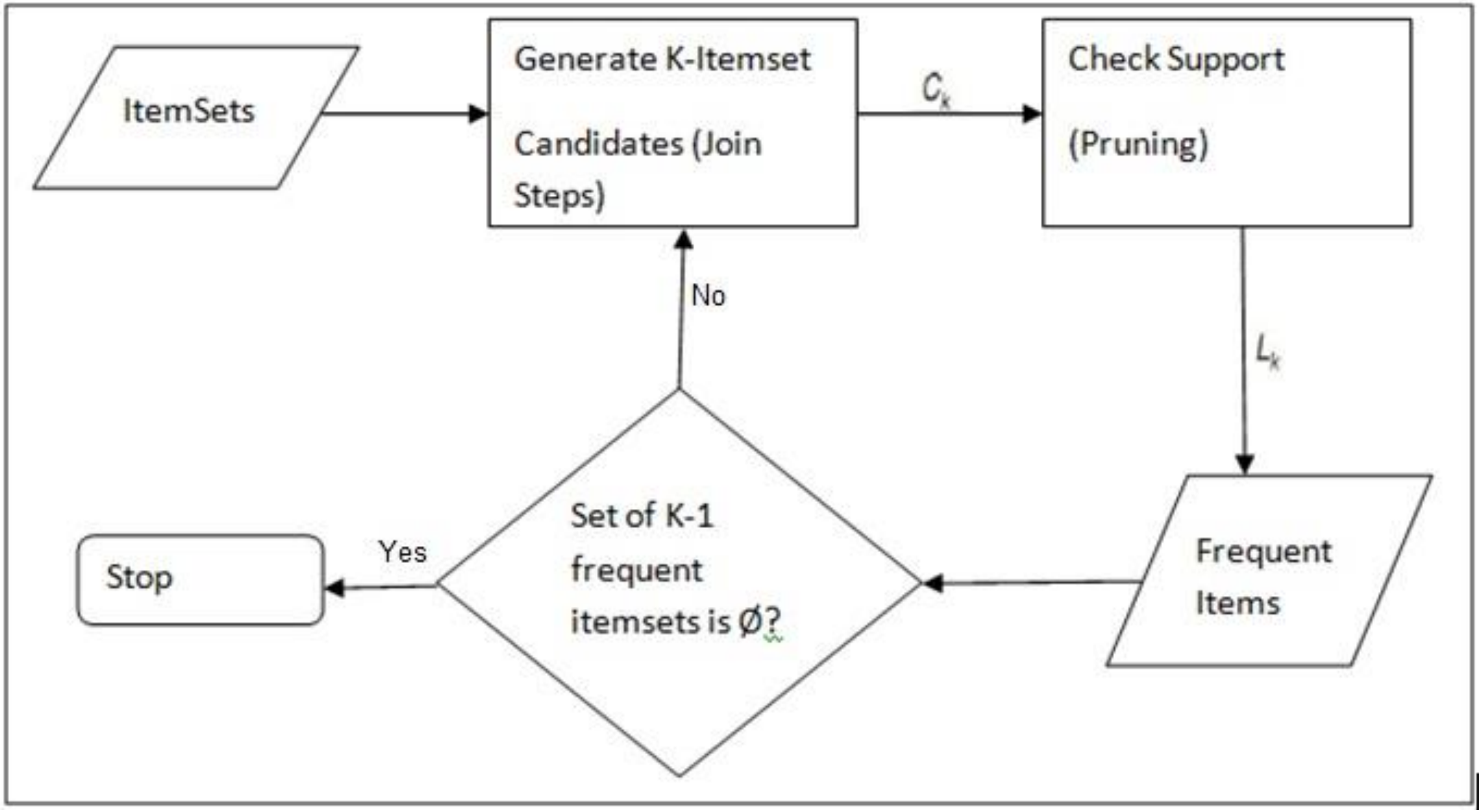
Mining Frequent patterns, Associations & Correlations [Part-2]

Shubham kumar
Dept. of CS&IT
MGCU Bihar

- **Apriori algorithm:**



Apriori Algorithm

Input: **D**, a database of transactions.

min_sup, the minimum support count threshold

Output: L, frequent itemsets in D.

Steps:

[1] L_1 = frequent_1-itemsets(D);

[2] **For** (k = 2; $L_{k-1} \neq \phi$; k++)

[3] C_k = **apriori_gen(L_{k-1})**;

[4] **For** each transaction t ϵ D

[5] Increment the count of all candidates in C_k that contain in t.

[6] L_k = candidates in C_k with min_sup.

[7] **End**

[8] **Return** L = $\cup_K L_k$

apriori_gen(L_{k-1} : frequent (k-1) itemsets)

[1] **For** each itemset $l_1 \in L_{k-1}$

[2] **For** each itemset $l_2 \in L_{k-1}$

[3] **IF** ($l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots\ldots\ldots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$)

[4] **THEN** {

[5] $c = l_1 \bowtie l_2$ // Join step

[6] **For** each (k-1)-subset s of c

[7] **IF** $s \notin L_{k-1}$

[8] **THEN** delete c // Prune step

[9] **ELSE** add c to C_k

[10] **End**

[11] }

[12] **Return** C_k

Generating Association rules

- By using Apriori algorithm we are able to find frequent itmsets from transactional database. (In our example, we got frequent itemsets from database D).
- Now we can generate strong association rules from those frequent itemsets.
- Strong association rules satisfy both minimum support (min_sup) and minimum confidence (min_conf).
- Since frequent itmsets already satisfy the minimum support, therefore we have to focus only on minimum confidence.

As we know, Confidence($A \Longrightarrow B$) = $P(B/A) = \dfrac{\text{support count } (A \cup B)}{\text{support count } (A)}$

- The P(B/A) is the conditional probability that is expressed in terms of itemset support count, where support count(A ∪B) is the number of transactions containing the itemsets A∪B (i.e. items in A **and** items in B), and support count(A) is the number of transactions containing the itemset A.
- Based on the above equation, association rules can be generated as follows:

[1] Generate all nonempty subsets of each frequent itemset l.

[2] For every nonempty subset s of l, output the rule

$s \Rightarrow (l - s)$, IF $\frac{support_{count}(l)}{support_{count}(s)} \geq min\_conf$

where min_conf is the minimum confidence threshold.

- In our example, database D has different transactions which contain different itemsets as shown in figure. what are the association rules that can be generated from X?

D

TID	List of items_IDs
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

By using apriori algorithm,

we found frequent itemset X = {I1, I2, I5}.

The nonempty subsets of X are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}.

hence, the resulting association rules will be :

{I1, I2} ⇒ I5, confidence = 2/4 = 50%

{I1, I5} ⇒ I2, confidence = 2/2 = 100%

{I2, I5} ⇒ I1, confidence = 2/2 = 100%

I1 ⇒ {I2, I5}, confidence = 2/6 = 33%

I2 ⇒ {I1, I5}, confidence = 2/7 = 29%

I5 ⇒ {I1, I2}, confidence = 2/2 = 100%

- If the minimum confidence threshold is, say, 50%, then only the first, second, third, and last rules are output, because these are the only ones generated that are strong.

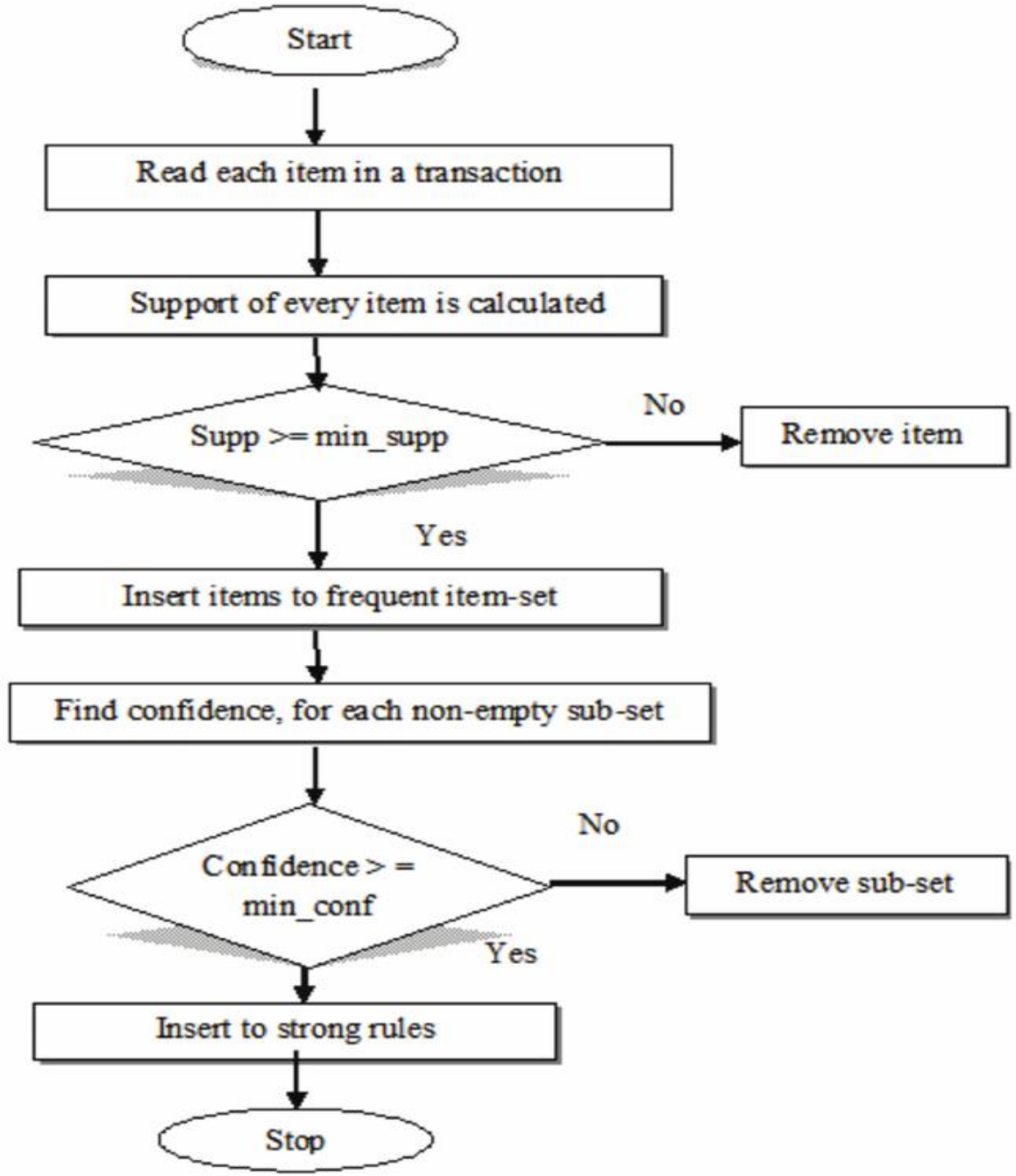
  i.e. $\{I1, I2\} \Rightarrow I5$,

  $\{I1, I5\} \Rightarrow I2$,

  $\{I2, I5\} \Rightarrow I1$,

  and $I5 \Rightarrow \{I1, I2\}$ are strong rules.

- We have realized that the strong rules can be generated easily after fidning frequent itemsets.
- Hence by using the following flowchart strong rules can be generated from the transactional database.


Start
Read each item in a transaction
Support of every item is calculated
Supp >= min_supp
No
Remove item
Yes
Insert items to frequent item-set
Find confidence, for each non-empty sub-set
Confidence > = min_conf
No
Remove sub-set
Yes
Insert to strong rules
Stop

- **Improving the Efficiency of Apriori:**

There are many approaches used to improve the efficiency of the Apriori algorithm.

Hash-based technique (hashing itemsets into corresponding buckets)**:** A hash-based technique can be used to reduce the size of the candidate k-itemsets, C_k , for $k > 1$.

For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L1, we can generate all the 2-itemsets for each transaction, map them into the different buckets of a hash table structure, and increase the corresponding bucket counts.

In the hash table , a 2-itemset with a corresponding bucket count less than the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash-based technique may substantially reduce the number of candidate k-itemsets (especially when k = 2).

- Create hash table H_2 using hash function:

  h(x,y)= ((order of x)*10 + (order of y)) mod 7

H_2

Bucket address	0	1	2	3	4	5	6
Bucket counts	2	2	4	2	2	4	4
Bucket contents	{I1,I4} {I3,I5}	{I1,I5} {I1,I5}	{I2,I3} {I2,I3} {I2,I3} {I2,I3}	{I2,I4} {I2,I4}	{I2,I5} {I2,I5}	{I1,I2} {I1,I2} {I1,I2} {I1,I2}	{I1,I3} {I1,I3} {I1,I3} {I1,I3}

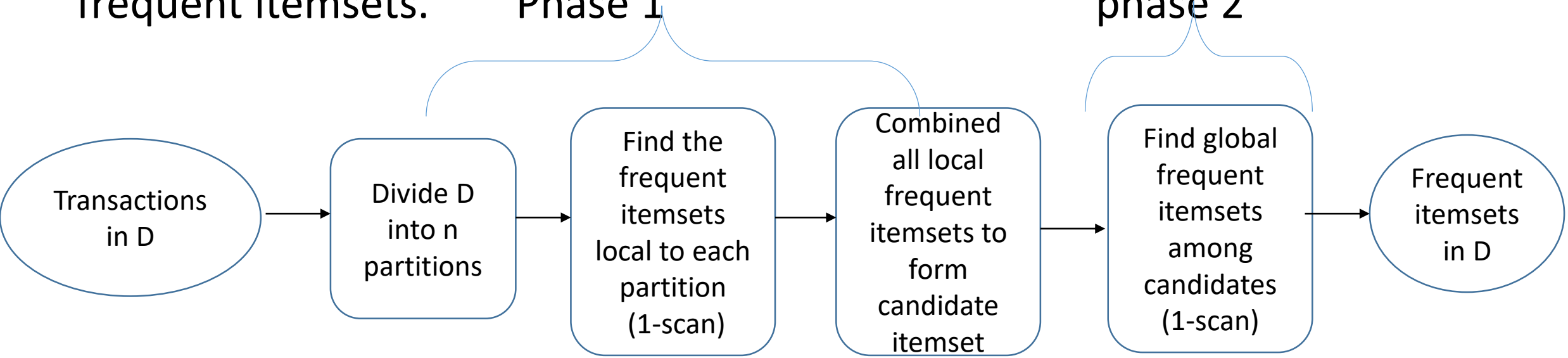
Hash table H_2 for candidate 2-itemsets

This hash table is generated by scanning the database D while determining L_1 .

If the minimum support count is, say 3, then the itemsets in buckets 0,1,3 and 4 cannot be frequent so they should not be included in C_2 .

- **Transaction reduction** (reducing the number of transactions scanned in future iterations)**:** A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k + 1)-itemsets. Therefore, such a transaction can be removed from further consideration because subsequent database scans for j-itemsets, where j > k, will not need to consider such a transaction.

- **Partitioning** (partitioning the data to find candidate itemsets)**:** A partitioning technique can be used that requires just two database scans to mine the frequent itemsets. Phase 1 phase 2



- It consists of two phases. In the first phase, the transactions of D are divided into n non overlapping partitions.
- If the minimum relative support threshold for transactions in D is min_sup, then the minimum support count for a partition is min_sup multiplied by the number of transactions in that partition.
- For each partition, all the local frequent itemsets (i.e., the itemsets frequent within the partition) are found.
- A local frequent itemset may or may not be frequent with respect to the entire database, D.
- However, any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions.
- Therefore, all local frequent itemsets are candidate itemsets with respect to D. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to D.

- In the second phase, a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets.
- Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

Sampling and Dynamic itemset counting are other approaches to improve the efficiency of apriori algorithm.

Reference

- Jiawei Han, Micheline kamber and Jian pei. “DATA MINING concepts and Techniques” 3/e, Elsevier, 2012