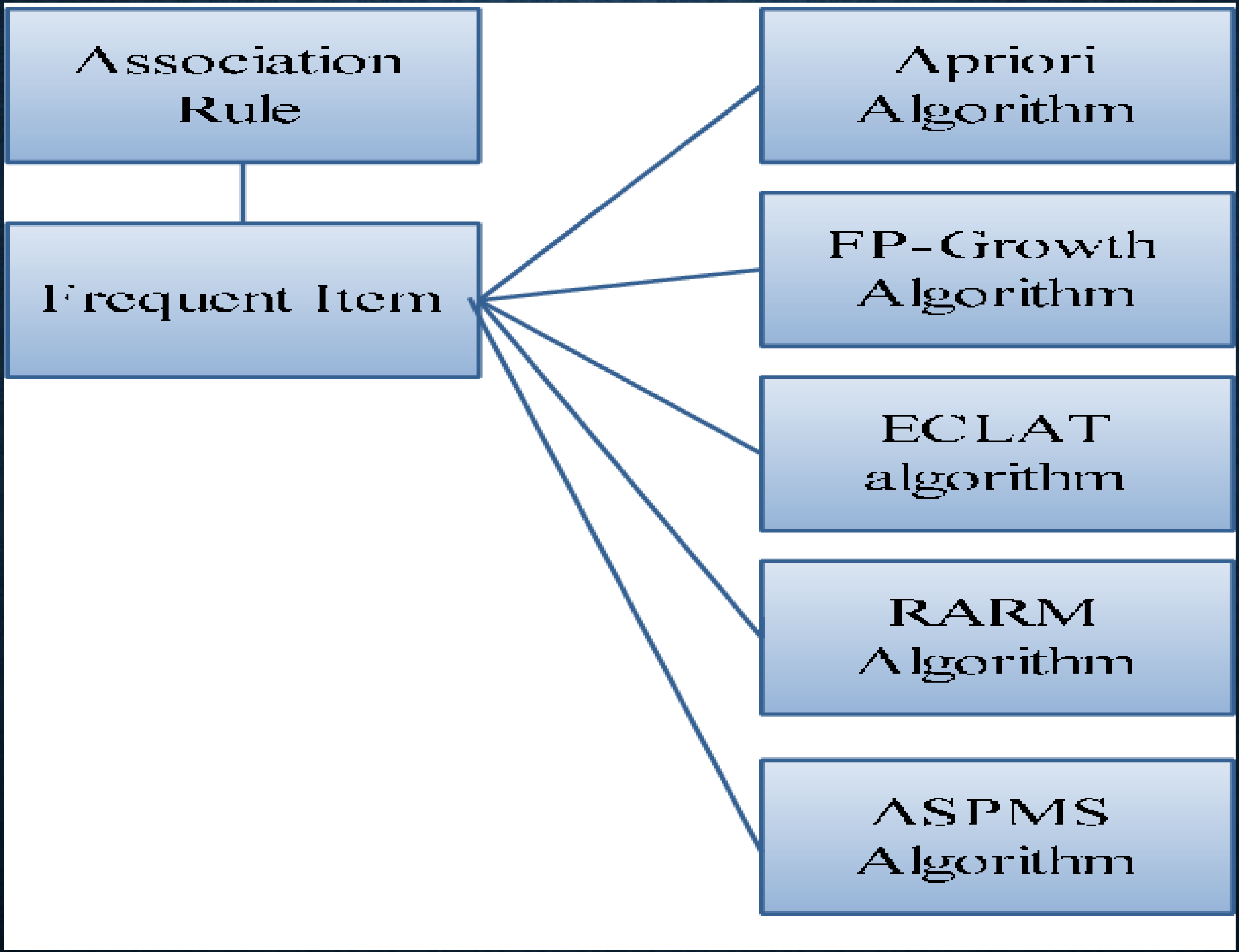
Introduction To **Data Mining**


Association Rule
Frequent Item
Apriori Algorithm
FP-Growth Algorithm
ECLAT algorithm
RARM Algorithm
ASPMS Algorithm

What is Association Rule?

Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently a itemset occurs in a transaction. A typical example is Market Based Analysis.
Market Based Analysis is one of the key techniques used by large relations to show associations between items. It allows retailers to identify relationships between the items that people buy together frequently.

What is Frequent Item?

In frequent mining usually the interesting associations and correlations between item sets in transactional and relational databases are found. In short, Frequent Mining shows which items appear together in a transaction or relation.

APRIOPRI ALGORITHM

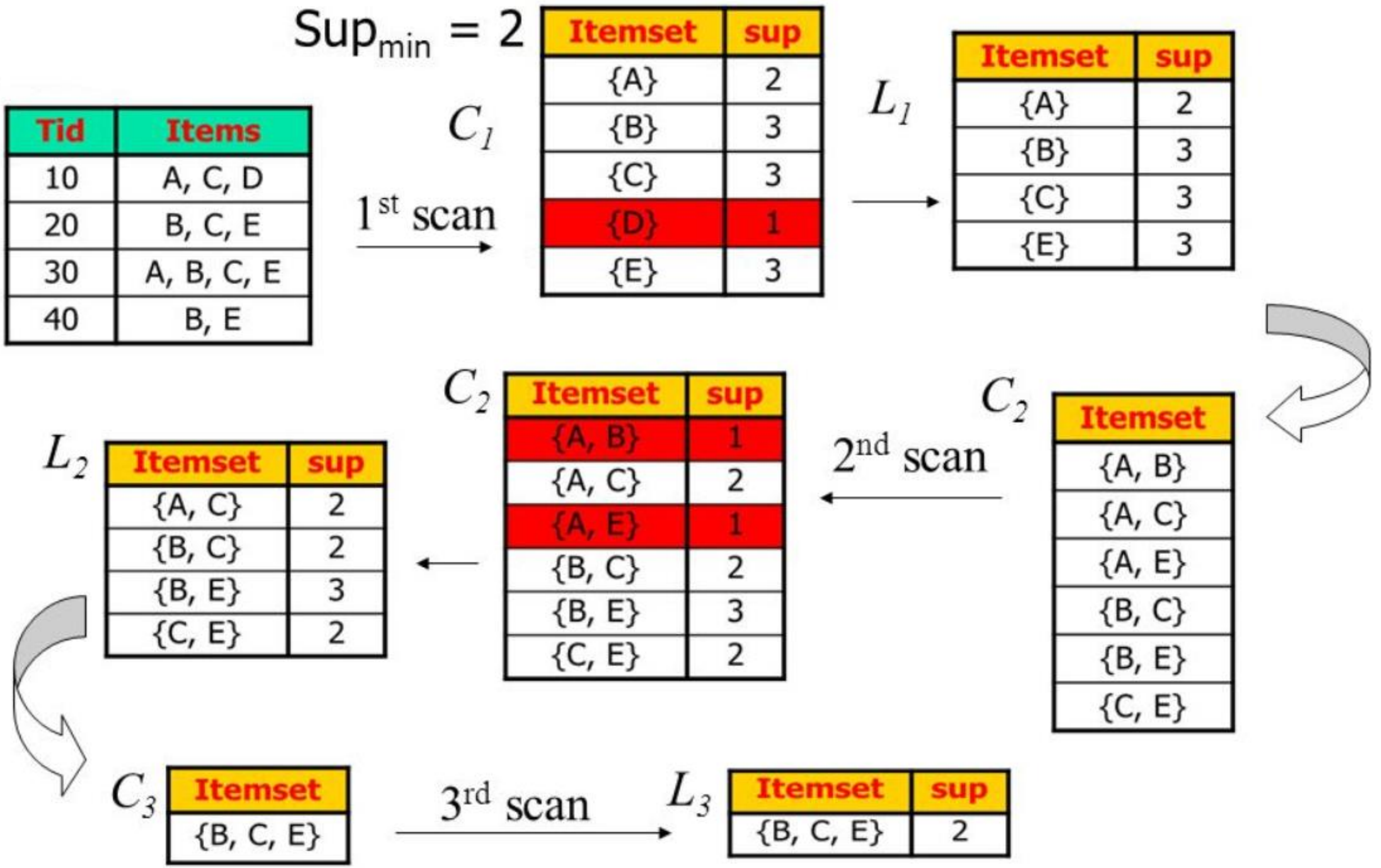
Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.
To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space.

Apriori Property –
All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that

All subsets of a frequent itemset must be frequent(Apriori property).
If an itemset is infrequent, all its supersets will be infrequent.

Limitations of Apriori Algorithm

Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemset, low minimum support or large itemset i.e. it is not an efficient approach for large number of datasets. For example, if there are 10^4 from frequent 1- itemset, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemset that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemset, also it will scan database many times repeatedly for finding candidate itemset. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.



$Sup_{min} = 2$

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

1st scan $\longrightarrow$

C_1

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

L_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 $\longleftarrow$

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan $\longrightarrow$

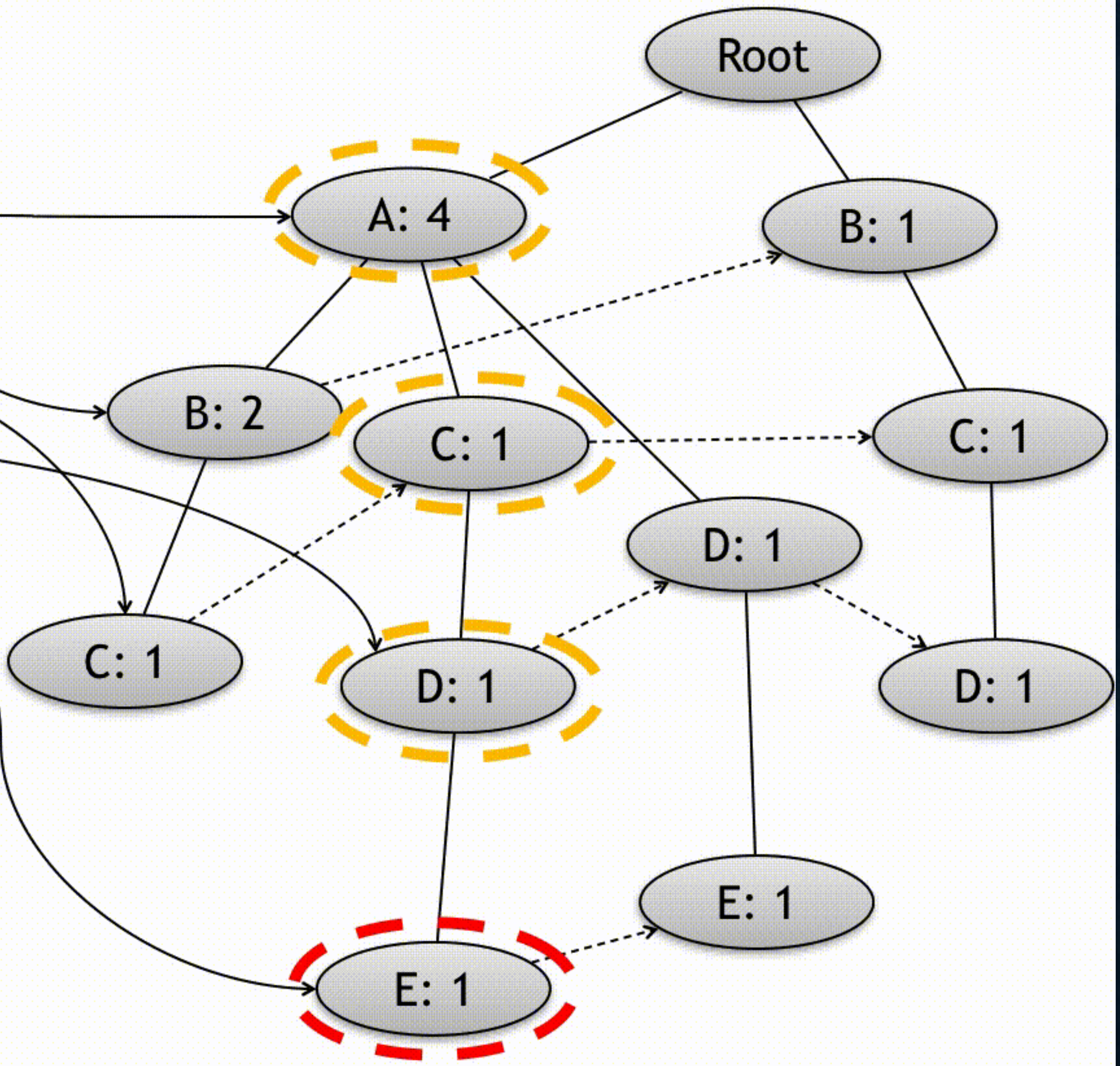
L_3

Itemset	sup
{B, C, E}	2

FP GROWTH

The two primary drawbacks of the Apriori Algorithm are:
1.At each step, candidate sets have to be built.
2.To build the candidate sets, the algorithm has to repeatedly scan the database.
These two properties inevitably make the algorithm slower. To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm. It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure.

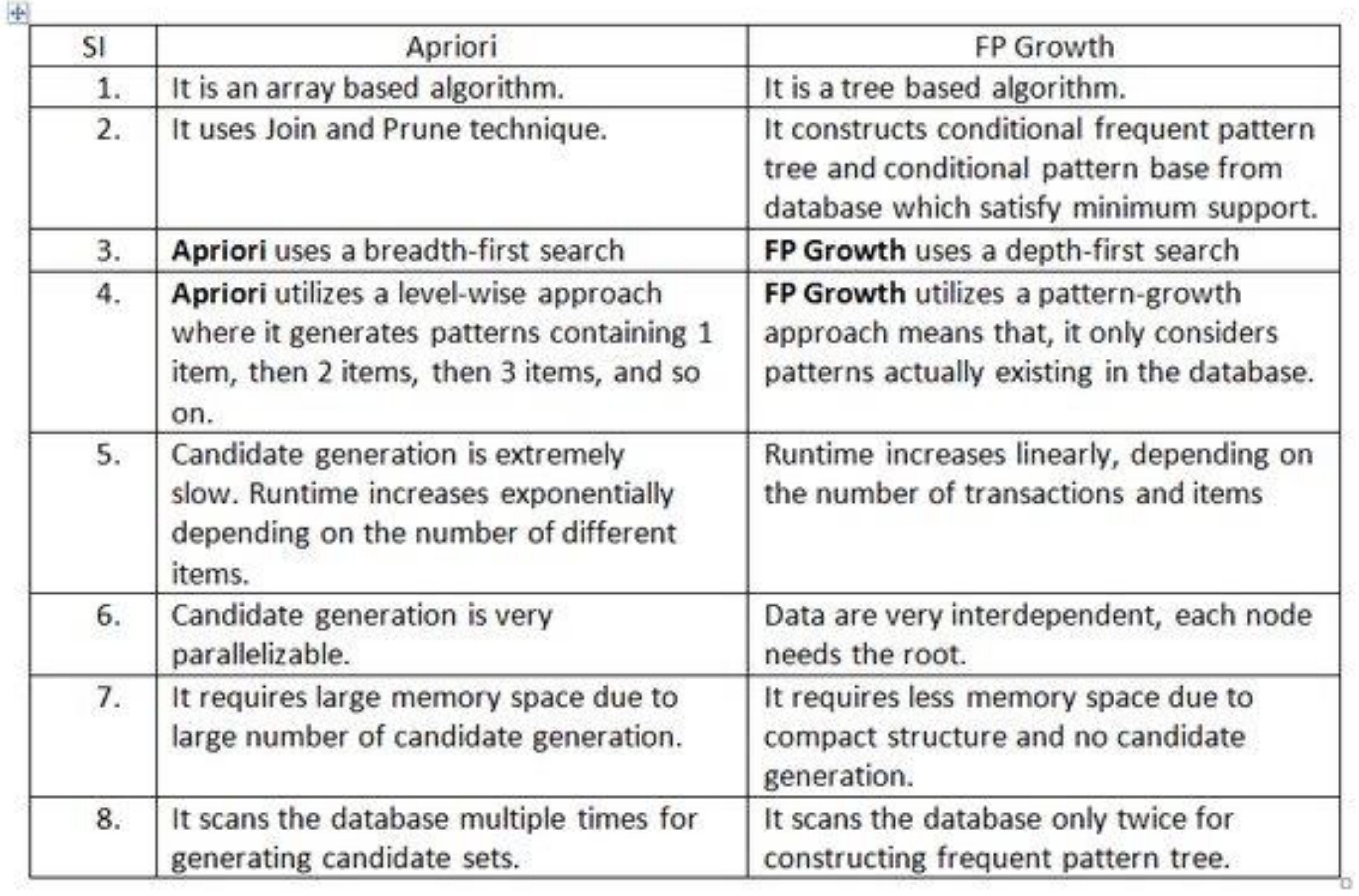
Item	Head of node-link
A	
B	
C	
D	
E	



Cond. FP-tree on "E"

{ (A:1, C:1, D:1, E:1),

}



Sl	Apriori	FP Growth
1.	It is an array based algorithm.	It is a tree based algorithm.
2.	It uses Join and Prune technique.	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy minimum support.
3.	**Apriori** uses a breadth-first search	**FP Growth** uses a depth-first search
4.	**Apriori** utilizes a level-wise approach where it generates patterns containing 1 item, then 2 items, then 3 items, and so on.	**FP Growth** utilizes a pattern-growth approach means that, it only considers patterns actually existing in the database.
5.	Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.	Runtime increases linearly, depending on the number of transactions and items
6.	Candidate generation is very parallelizable.	Data are very interdependent, each node needs the root.
7.	It requires large memory space due to large number of candidate generation.	It requires less memory space due to compact structure and no candidate generation.
8.	It scans the database multiple times for generating candidate sets.	It scans the database only twice for constructing frequent pattern tree.