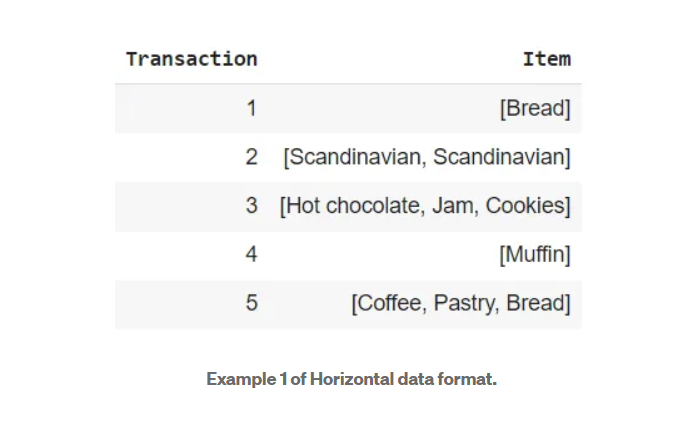
**Eclat and FP Growth:**

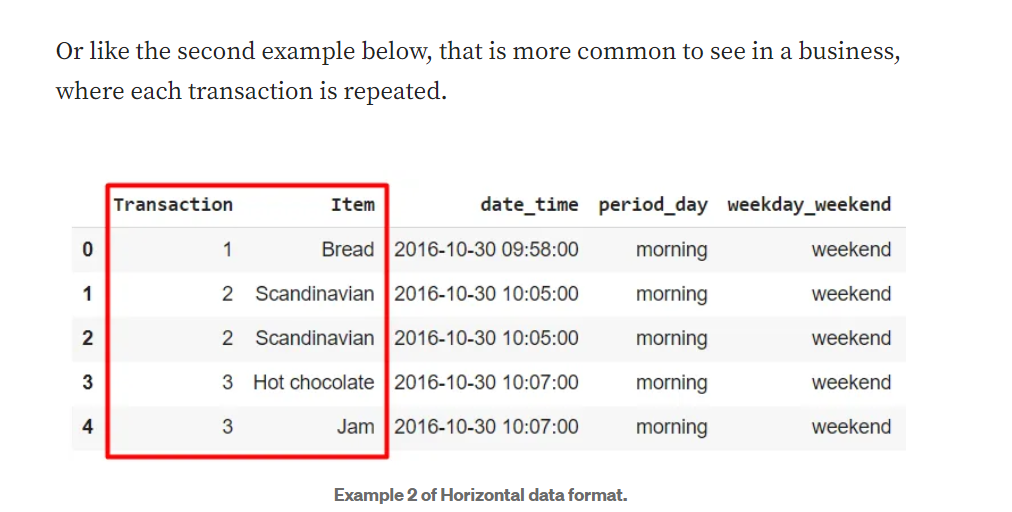
The ECLAT algorithm is not the first algorithm for **association rule mining.** The foundational algorithm in the domain is [**the Apriori algorithm**](https://towardsdatascience.com/the-apriori-algorithm-5da3db9aea95). Since the Apriori algorithm is the first algorithm that was proposed in the domain, it has been improved upon in terms of computational efficiency (i.e. they made faster alternatives).

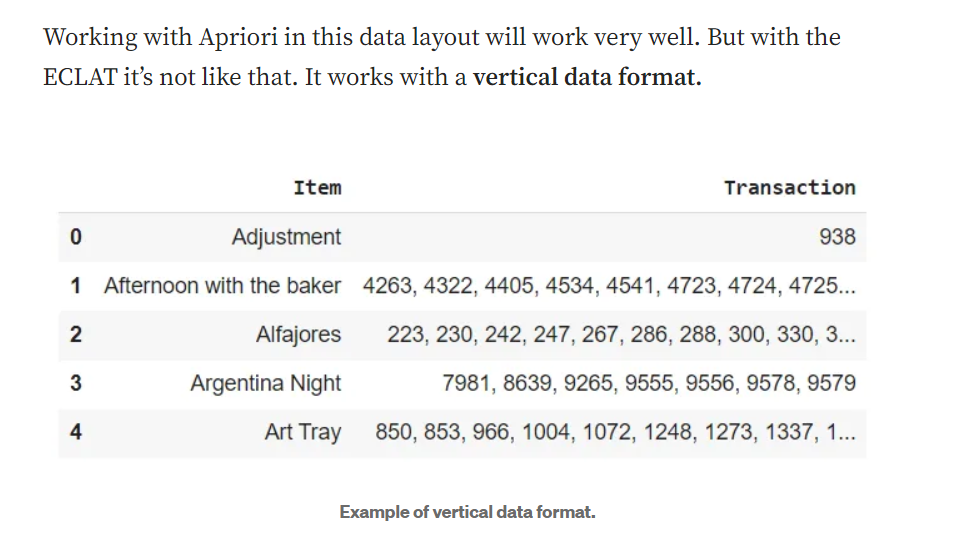
**ECLAT vs FP Growth vs Apriori**

There are two faster alternatives to the Apriori algorithm that are state-of-the-art: one of them is [**FP Growth**](https://towardsdatascience.com/the-fp-growth-algorithm-1ffa20e839b8) and the other one is **ECLAT**. Between FP Growth and ECLAT there is no obvious winner in terms of execution times: it will depend on different data and different settings in the algorithm.

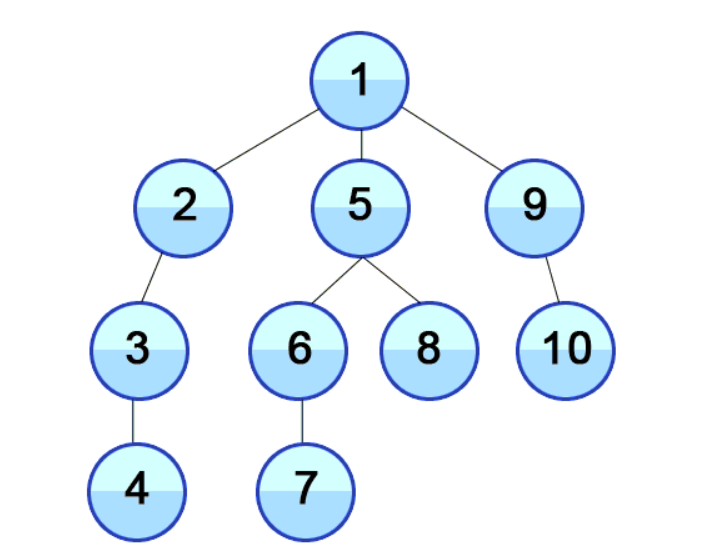
The first thing that is important to say is that there are 2 kinds of data formats to apply the Association Rules Algorithm. **Horizontal data format** and **vertical data format**. For example, is pretty common we used the horizontal format layout, where each transaction is represented by an ID and a list of items besides.

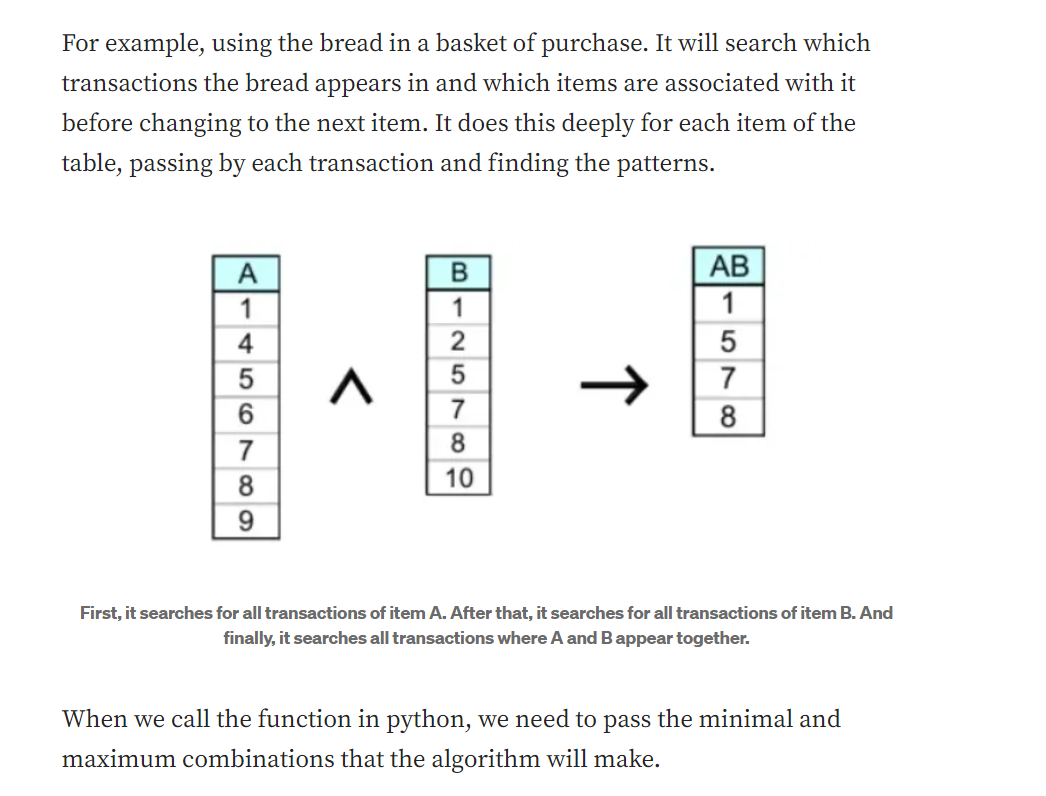






This format for ECLAT is necessary because it does the association using the Depth-First Search of a graph (DFS). What does it mean? This means that it finds all the combinations of items of the most item more frequently before of change to another combination.







## Well, what are the advantages of this algorithm?

1. The Eclat algorithm has low memory requirements compared to Apriori because it uses the vertical layout (DFS method).
2. The Eclat algorithm does not the repeated scan of the data to calculate the support, thus, it is faster than Apriori.
3. At each stage of the generated database, the Eclat algorithm uses the current generated dataset to learn frequent items, unlike the Apriori which scans the original database repeatedly. Since the Eclat scans over the database once, it is much faster than the Apriori algorithm.

## Conclusion

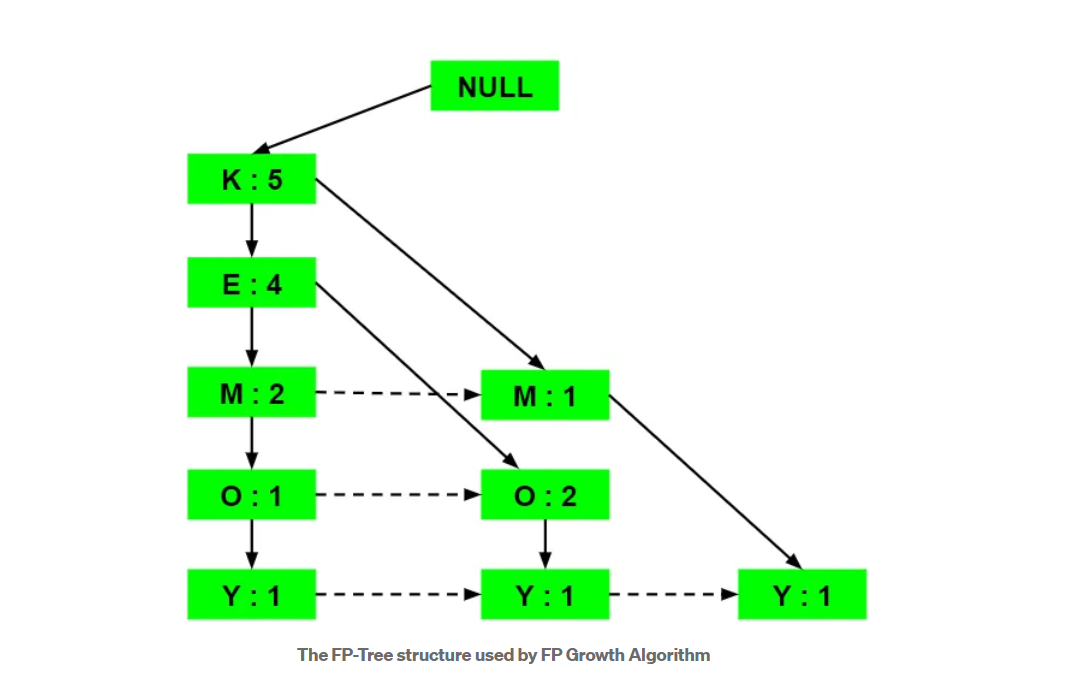
The ECLAT is faster than Apriori if we use it in the small or medium dataset. However, when we talk about the large dataset is possible that Apriori performs better. This happens because the ECLAT algorithm consumes more space in memory than Apriori and when we leave a large dataset intermediate results of vertical item lists become too large for memory, thus affecting the algorithm scalability.  
So, we can conclude that ECLAT is good in small or medium datasets and in situations where we won't need a lot of measures like lift or confidence.

**FP Growth**

FP Growth means Frequency Pattern Growth, and can be considered a version improve from Apriori as it is faster and more efficient to obtain the same goal. But there is another point, while Apriori needs to scan the database multiple times to check the support count, the FP Growth needs only twice. Furthermore, it uses a different structure known as FP-Tree to store the information.

The first node represents the “null” and each other node represents an item. As the proposed algorithm is to find the frequent itemsets, this structure (FP-Tree) allows it to be faster than Apriori.

Look at the image below. The K, E, M, O, and Y are the item. You can imagine them as supermarket items like butter, bread, coffee, rice, and milk. The number in front of them represents the frequency of each item, for example, K appeared in 5 purchases. You can realize that the image below has the dashes arrows, which means that items are linked and appear in other order.



FP Growth Algorithm is the method of finding frequent patterns without candidate generation. It constructs an FP Tree rather than using the generate and test strategy of Apriori. The focus of the FP Growth algorithm is on fragmenting the paths of the items and mining frequent patterns.

This method has an advantage over Apriori as it does not require scanning the database to find the support of itemsets. This is because the Transaction set will carry the count of occurrence of each item in the transaction (support). The bottleneck comes when there are many transactions taking huge memory and computational time for intersecting the sets.