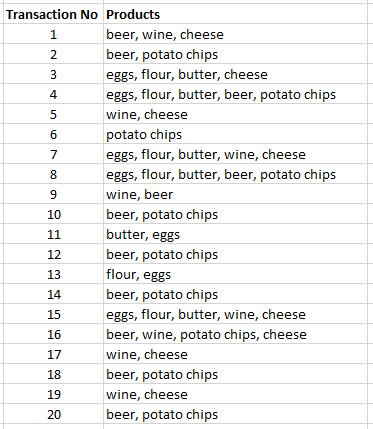
**Eclat/dEclat:**

Frequent itemsets and association rules mining

* The **ECLAT** (**Equivalence Class Clustering and bottom-up Lattice Traversal**) algorithm is a data mining algorithm for association rule mining designed to solve customer bucket analysis problems. The goal is to understand which products from the bucket are commonly bought together.
* Eclat uses depth first search (DFS) algorithm instead of breadth first search (BFS) in **apriori** algorithm.
* There are two ways to organize data in relational databases:
* **Row oriented** – traditional way of storing data, that stores data records in rows and splitting it by one or several columns
* **Column-oriented** (also known as **columnar** or **C-store**) – stores data by field, keeping all of the data associated with a field next to each other
* The Apriori and FP-growth algorithms require data to be in the row format (sometimes called “horizontal format”). In contrast, **the ECLAT algorithm is designed to deal with the data stored in the column-oriented format** (sometimes called “vertical format”).
* Here’s an example of transactions datasets stored in both formats:



* **An example use case for the ECLAT algorithm:**

Let’s now introduce an example use case to make the topic a little bit more practical and applied. In this article, we will take a small dataset of transactions of a night store. For each transaction, we simply have a list of products.

The original data set looks as follows: 🡪

As you can already see, the customers of this night store mainly buy beer, potato chips, wine and the like, together with some other products.

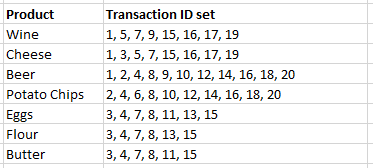
* **ECLAT: faster but fewer metrics:**

If you are already familiar with other algorithms for frequent itemset mining or associative rules mining, there is a small thing that may surprise you. Whereas most algorithms contain a number of key metrics about their rules, ECLAT does not do the same. For example, **ECLAT** does not give you the **Confidence and** **Lift metrics** that are essential for interpretation in the alternative models. On the other hand, this allows the model to be faster: the user has a choice to make between speed and having more metrics.

* **How does the ECLAT algorithm work?**

Let us now get into the **ECLAT** algorithm by replicating the steps of the algorithm by hand.

**Step 1 — List the Transaction ID (TID) set of each product**

The first step is to make a list that contains, for each product, a list of the Transaction IDs in which the product occurs. This list is represented in the following table.

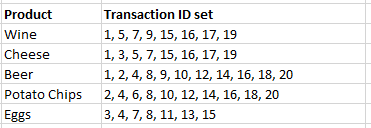
The ECLAT Algorithm. The Transaction ID (TID) sets for each product.

These transaction ID lists are is called the Transaction ID Set, also called TID set.

**Step 2 — Filter with minimum support**

The next step is to decide on a value called the **minimum support.**The minimum support will serve to filter out products that do not occur often enough to be considered.

In the current example, we will choose a value of 7 for the minimum support. As you can see in the table of Step 1, there are two products that have a TID set that contains less than 7 transactions: Flour and Butter. Therefore, we will filter them out, and we obtain the following table:

The ECLAT Algorithm. Filtering out products that do not reach minimum support.

**Step 3 — Compute the Transaction ID set of each product pair**

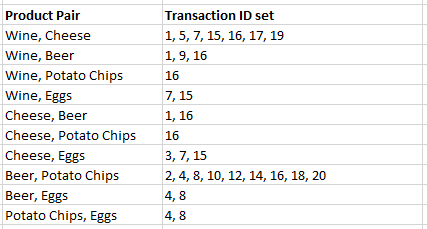
We now move on to pairs of products. We will basically repeat the same thing as in step 1, but now for product pairs.

The interesting thing about the ECLAT algorithm is that this step is done using the **Intersection of the two original sets**. This makes it different from the Apriori algorithm.

The ECLAT algorithm is faster because it is **much simpler to identify the intersection of the set** of transactions IDs than to *scan each individual transaction* for the presence of pairs of products (as Apriori does). You can see in the below image how it's easy to filter out the transaction IDs that are common between the product pair Wine and Cheese:

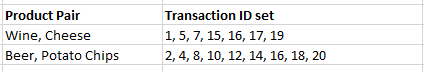
The Eclat Algorithm

The ECLAT Algorithm. Finding the intersection of Transactions IDs is easier than scanning the whole database.

When doing the intersection for each product pair (ignoring the products that did not reach support individually) this gives the following table:

The Transaction ID sets for all product pairs that are still in the race.

**Step 4 — Filter out the pairs that do not reach** minimum **support**

As before, we need to filter out results that do not reach the minimum support of 7. This leaves us with only two remaining product pairs: Wine & Cheese and Beer & Potato Chips.

The ECLAT Algorithm. There are two product pairs that meet support.

**Step 5— Continue as long as you can make new pairs above support**

From this point on, you repeat the steps as long as possible. For the current example, if we create the product pairs of three products, you’ll find that there aren’t any groups of three that reach the minimum support level. Therefore, the association rules will be those obtained in the previous step.