For purposes of this discussion, we use the following terms:

* Partition. A group in some sort of aggregation. For example, in SELECT SUM(Sales) GROUP BY Region and SELECT COUNT(Clicks) GROUP BY URL, the data are partitioned by Region and URL, respectively.
* tau = maximum number of times a single user can appear in one partition.
* Delta = maximum number of partitions that a user can appear in.

Here are some examples of queries with different tau and Delta:

1. SELECT COUNT DISTINCT(SalesPerson) GROUP BY Region;
   1. tau = 1, Delta = 1
   2. Each sales person in in one region, and we count distinct
2. SELECT COUNT DISTINCT(Users) GROUP BY URL;
   1. tau = 1, Delta = infinite
   2. Each user might browse several URLs. In practice, we probably reservoir sample to cap Delta = 300 or something like that
3. SELECT COUNT(Orders) GROUP BY Region;
   1. tau = ~10, Delta = ~1
   2. Each user might have multiple orders per region. In practice, we could cap this to some upper bound. A user might also have orders in multiple regions, so Delta might be bigger, but in practice we can set some small upper bound.
4. SELECT COUNT(Clicks) GROUP BY URL;
   1. tau = ~10, Delta = infinite
   2. As in example #2 above, each user could browse hundreds of URLs, and we will usually reservoir sample Delta to cap at something like 300. Additionally, each user may have more than one click per URL, and we would usually set an upper bound such as 10.

These examples just deal with counts, where sensitivity for the count is the same as tau. For something like a SUM, we would multiply the sensitivity of the column by tau. We will set that aside for the rest of this discussion.

**Relation to Burdock**

Burdock has a parameter ‘max\_contrib’ that maps roughly to tau. We support only queries of type 1 and 3, where tau is fixed and small, and Delta == 0. Additionally, we support queries of type 2, where tau == 1 and Delta is very large, via preprocessing with DPSU.

Note that all 3 of our supported scenarios require that at least one of Delta or tau == 1. Therefore, we could use the ‘max\_contrib’ parameter for all 3, as follows:

1. For queries of type 1 and 3, we already rewrite the query and use validator to ensure that Delta == 1. If users issue a query that looks like this, we could assume that max\_contrib -> tau
2. For queries of type 2, we could assume max\_contrib -> Delta. The sampling happens in preprocessing via DPSU, so we turn off that part of the validator when running the postprocessing.

Question: Is it really possible to infer user intent, just by looking at the query? We might need a flag anyway.

OK, assuming we handle those 3 scenarios, we have two additional edge cases to think about.

First, for queries of type 3, what if we want the user to be able to have more than one order per-partition, and more than one partition? For example, the user has 10 orders, and 3 of them are in US, and the other 7 are in Europe. There are two possible ways to handle it:

1. Let user specify tau and Delta independently. Reservoir sample per-partition to enforce tau, and then figure out some clever way to drop records if they end up in more than Delta partitions.
2. Assume that Delta and tau can borrow from one another. In other words, set max\_contrib = 10, which allows 10 orders in one partition, or 1 each in 10, or 7+3, etc. To handle this, we can simply reservoir sample upfront once, and we need only one parameter.

Question: Option B is simpler to explain, but we waste some noise. I think this is probably OK.

For the last scenario, queries of type 4, we would need two parameters. We could offload this to DPSU postprocessing, so main burdock engine doesn’t have to know about it.

Note that all other query types are a subset of type 4, so we could effectively implement just one option and then use that for everything else. However, the algorithm we have requires in-memory processing of all records via DPSU, so it doesn’t fit well with our approach of offloading aggregation to SQL engines.