SQL tuning methodology

A common SQL tuning methodology is to:

1. Verify that the execution path is the optimal for the query. If not, either use the tuning directives (such as hints on Oracle) or identify why the native optimizer failed to pick the optimal path.
2. If the query is still slow, then look at adding indexes.
3. If the query is still slow, then you know you are going to have to look at the architecture.

Ask yourself:

- What information is the query trying to get?
- Is this information necessary?
- Are there alternative ways to get this information?

DB Optimizer's SQL Tuner can help with steps 1 and 2. Step 3 must be done by a developer or DBA, but knowing that steps 1 and 2 are already validated can indicate to management that step 3 is necessary and therefore allocate sufficient resources for step 3.

How do we know if the native database optimizer chose the optimal path? How long would it take to check this by hand?

DB Optimizer's SQL Tuner is a solid fast sanity test to verify the plan chosen by the native database SQL optimizer. Tuner quickly generates as many alternative paths as possible and allows the user to execute them to see if there are more efficient execution paths. DB Optimizer's SQL Tuner is successful at tuning queries that have a sub-optimal execution path.

A query has a sub-optimal execution path when the database optimizer has miscalculated the cost of the various possible access paths and mistakenly chosen a bad path. The access path calculations are sometimes miscalculated because of the following reasons:

- **The table/index statistics are missing or wrong**. For example, the number or rows is missing or way off.
- **The data is skewed, for example, the number of orders with an open status is usually low compared to all the orders that have a closed status because the work is complete**. For example, orders get filled every day, but only a few are open and needing to be processed. Looking for open orders should probably use an index and return fewer rows than looking for closed orders which should probably just do a full table scan.
- **The predicates used are correlated**. The optimizer treats two predicate filters on a table as more selective than just one, but this is not always the case such is the case in the query, how many Swedes speak Swedish which basically returns the same number of results as just asking for the number of Swedes alone. Another example is how many Swedes speak Swahili, which is probably more selective than the optimizer would guess.
- **There is a bug in the optimizer**.

DB Optimizer's SQL tuner takes a query and tries to produce as many execution paths as possible. These alternative execution paths can then be run to see if there is a faster or less resource expensive execution path. The execution of each alternative case is timed and if the execution exceeds 1.5 X the original case then its execution is stopped and we move on to the next case. This avoids wasting time and resources on execution plans that are clearly sub-optimal.