Caching

Caching is the process of storing a calculated value for later use.

The *lazy evaluation* strategy employed in Megaladata calculates values only when required. With this strategy, intermediate calculations are not stored in memory and are recalculated each time the platform needs them. This approach reduces the demand for computational resources, specifically random-access memory (RAM). However, if an algorithm uses the same intermediate data multiple times, lazy evaluation can increase the total execution time. In these situations, we recommend saving the intermediate calculations in memory for later use.

Therefore, enable caching to avoid repeated calculations when subsequent handlers and visualizers use the same calculated values multiple times.

While data caching requires additional memory, it can decrease the algorithm's execution time in the situations described above. Megaladata allows for balancing memory usage and execution speed depending on your needs.

Caching calculated expressions

- Enable caching in the Calculator's expressions when using the `Data()` function for recursive computations.
- We recommend that you use caching for functions whose results depend on the time of calculation, for example, `Random()` and `Today()`.

Read on: Configuring Locale