Engine Settings For Both System-Defined And User-Defined Aggregates

There settings that you can use to influence the behavior of the AtScale engine with regard to system-defined and user-defined aggregates. The settings apply to all of the aggregates.

For descriptions of the different types of aggregate, see Types of Aggregate Tables in AtScale.

Settings For Enabling And Configuring Incremental Builds

Use these settings to turn on incremental builds, specify the maximum allowed duration of an incremental build per aggregate, and specify the maximum number of partitions per aggregate.

AGGREGATE.INCREMENTALUPDATE.ENABLED

Set to **True** to use incremental builds for all of the aggregates for a model when the fact dataset uses an incremental indicator. Full builds are still done for user-defined aggregates that are joins or unions of two or more fact tables.

AGGREGATE.INCREMENTALUPDATE.ALLPARTITIONMATERIALIZATIONS.DURATION

Specify the maximum length of time to allow for an incremental build of an aggregate table.

AGGREGATE.INCREMENTALUPDATE.MAXCONSECUTIVESTATICPARTITIONS

Specify the maximum number of partitions to allow for each incrementally built aggregate. When this threshold is exceeded, the partitions are consolidated. Lower values relative to the default result in slower consolidations and faster queries. Higher values result in faster consolidation and slower queries.

Settings For Enabling And Prioritizing Builds Of Aggregate Instances

Use these settings to enable prioritization, specify whether new aggregates should be prioritized over existing ones, and specify whether user-defined aggregates should be prioritized of system-defined ones.

AGGREGATE.QUEUE.PRIORITY.MODE

This setting accepts integer values 0-2:

- 0: Prioritizes building new instances before existing instances that are part of a batch build.
- 1: Prioritizes building batch instances before new instances.
- 2: Prioritizes building aggregate instances in order of definition timestamp (FIFO).

The default value is 1.

AGGREGATE.QUEUE.PRIORITY.USERDEFINED.ENABLED

Set to **True** to prioritize the building of user-defined aggregates over the building of system-defined aggregates.

Settings For Configuring Reattempts To Build Aggregate Instances

These settings apply when an instance of an aggregate fails to be built during the aggregate build process for a model.

AGGREGATES.BATCH.RETRY.MAXATTEMPTSPERAGGREGATE

Specify how often to reattempt to build a single aggregate when the build of that aggregate fails during a batch aggregate build for a model. The default is three times.

AGGREGATES.BATCH.RETRY.MAXATTEMPTSPERBATCH

Specify how often to reattempt to build aggregates during a single batch aggregate build. The default is 5 times.

These two settings interact as in the following example:

Step in the build process	Restarts for Aggregate A	Restarts for Aggregate B	Restarts for Aggregate C	Total restarts for the batch
A batch build of the aggregates in a model starts.	0	0	0	0
The build for Agg A fails and restarts.	1	0	0	1
The build for Agg A again fails and restarts.	2	0	0	2
The build for Agg B fails and restarts.	2	1	0	3

The build for Agg C fails and restarts.	2	1	1	4
The build for Agg C again fails and restarts.	2	1	2	5
The build for Agg C again. The batch build fails as a whole because the max number of retries has been reached.	2	1	1	5

Moreover, if the value of AGGREGATES.BATCH.RETRY.MAXATTEMPTSPERAGGREGATE is reached before the value of AGGREGATES.BATCH.RETRY.MAXATTEMPTSPERBATCH during a batch aggregate build, the build fails and ends.

Whenever a new build starts, the counters for both settings are reset to 0. To use the example in the table above, when a new batch build starts, the counter for the number of restarts for each aggregate table is set to 0. The counter for the total number of restarts for the batch is also reset to 0.

Settings Specific For Data Warehouses

The aggregate settings described here should be applied only when you use the corresponding data warehouse.

Aggregates.Create.Usectas.Snowflake

This setting is specific for the Snowflake data warehouse. When enabled (default), it allows using a faster method for creating aggregates for very large data sets. In case of performance issues you can try disabling or enabling this setting.

Aggregates.Create.Iris.Nojourn.Enabled

This setting is specific for the InterSystems IRIS data warehouse, versions 2022.2 or newer. It allows you to set whether to use the %NOJOURN parameter when creating aggregate tables. The default value is **False**.

Settings For Large Table Optimization

The AtScale engine features settings that help optimizing aggregates for cloud-based data warehouses. When these features are enabled, system-generated aggregates will leverage distribution keys in the databases (sometimes referred to clustering in certain databases). AtScale will look at the available columns in the system-defined aggregate and select the most appropriate column to distribute the aggregate. This will most likely be a join key, but in the absence of an available key, will use several dimensional value columns, if the database allows for it.

Note: Modifying these settings does not require an engine restart.

Aggregates.LargeTableOptimization.Enabled

Enables consideration of optimizations for large aggregate tables, such as column-based clustering or distribution. The default value is **False**.

Aggregates.LargeTableOptimization.MinimumEstimatedRows

The minimum estimated row count required for the aggregate system to consider applying optimizations such as clustering or distribution. The default value is 100000.

Aggregates.LargeTableOptimization.DistributionKeyColumn.MinimumCardinality

The minimum cardinality required for the highest cardinality dimensional attribute to be used as a table distribution or clustering key. The default value is 30.

Sample Optimization Results

The information in this section is based on tests performed by AtScale. Consider that the results in your setup, with your data, and modified settings can be different.

BigQuery

- Tests performed with 27GB aggregate table, and 96MB dimension table.
- Using distribution for the aggregate table results in almost half the query time.
- Adding distribution on the dimension table seems to add minor improvement to the performance.
- With smaller tables, there is no noticeable difference in performance with or without distribution.

Databricks

Enabling distribution leads to queries that are several times slower.

Other Settings For Both System-Defined And User-Defined Aggregates

A number of features of system-defined and user-defined aggregates are enabled or affected by single settings.

AGGREGATES.CREATE.BUILDFROMEXISTING

Set to **True** to allow new aggregate tables to be built from data that is in an existing aggregate table. This option does not affect subsequent builds.

The AtScale engine continuously assesses the quality of the aggregate-table definitions that it has generated. If it determines that a new definition is needed, by default the first instance of that definition is built from a query against raw data, even if that definition is based on a current aggregate-table definition.

Use this setting to allow the first instance of a new definition to be built from the data that is already in an instance of another definition. Allowing the first instance to be built in this way speeds up the build process.

For example, suppose that the engine decides to supersede the aggregate-table definition AggDef1 by creating the new definition AggDef2, which is based on AggDef1. If this setting is set to **True**, the build of the first instance of AggDef2 will include data from the current instance of AggDef1. If the instance requires data that is not in the current instance of AggDef1, the engine queries raw data to gather it.

Non-incremental aggregates tables can be built only from non-incremental aggregate tables, while incremental aggregate tables can be built only from incremental aggregate tables.

The default value is **True**.

AGGREGATE.CREATE.SECURITYDIMENSIONS.ENABLED

Controls whether or not aggregates should be built when an attribute is part of a security dimension. The default value is **False**.

AGGREGATES.CREATION.TIMEOUT

Specify the maximum length of time to allow per DDL statement that the engine uses to create an aggregate instance. Aggregates that are refreshed with full builds require one DDL statement. Aggregates that are refreshed with incremental builds require one DDL statement per partition.

AGGREGATES.DROP.PURGE.ENABLED

This setting should only be enabled in specific circumstances.

AGGREGATES.DROP.TIMEOUT

Specify the maximum length of time to allow per DDL statement that the engine uses to drop an aggregate instance. The default value is 1.minute; restart is required. The setting is valid for SQL data sources. BigQuery ignores it, and uses its default 50 seconds timeout.

AGGREGATE.INCREMENTALUPDATES.IMMUTABLE.ENABLED

Set to **True** to enable incremental builds and rebuilds of aggregates that use joins on rarely changing dimensions. For more about such incremental builds, see About Incremental Rebuilds.

AGGREGATES.ORC.COMPRESS

Specify which compression method to use. This setting is applicable only if you set the value of AGGREGATES.TABLECONFIG.PREFERREDSTORAGEFORMAT to "orc".

AGGREGATES.TABLECACHE.ENABLED

To use this feature with AtScale, you must have a special AtScale license.

Enable the in-memory aggregate caching feature. The default value is **False**.

AGGREGATES.TABLECACHE.EXTRASUPPORTEDFUNCTIONS.ENABLED

Allow the use of the in-memory aggregate cache when the query contains functions from the set of extra functions (for example, LIKE). The default value is **True**. Can be used when the AGGREGATES.TABLECACHE.ENABLED setting (see above) is set to **True**.

AGGREGATES.TABLECONFIG.PREFERREDSTORAGEFORMAT

Specify the storage format for data in aggregate tables, if you have a preference. Possible values: orc, parquet, rcfile, textfile, none. Specify "none" to allow the engine to decide which format to use.

TABLES.CREATE.PARTITIONS.ENABLED

Set to **True** to enable the AtScale engine to partition table types that can be partitioned. The default value is **True**.

Partitioned aggregates are not supported when the data warehouse that you are using is an instance of Google BiqQuery.

If the value of this setting is not **True**, then the AGGREGATES.CREATE.PARTITION.USERDEFINEDAGGREGATE.ENABLED and AGGREGATES.CREATE.PARTITION.SYSTEMDEFINEDAGGREGATE.ENABLED engine settings have no effect, even when they are enabled. For details, see Engine Settings For User-Defined Aggregates Only and Engine Settings For System-Defined Aggregates Only, respectively.