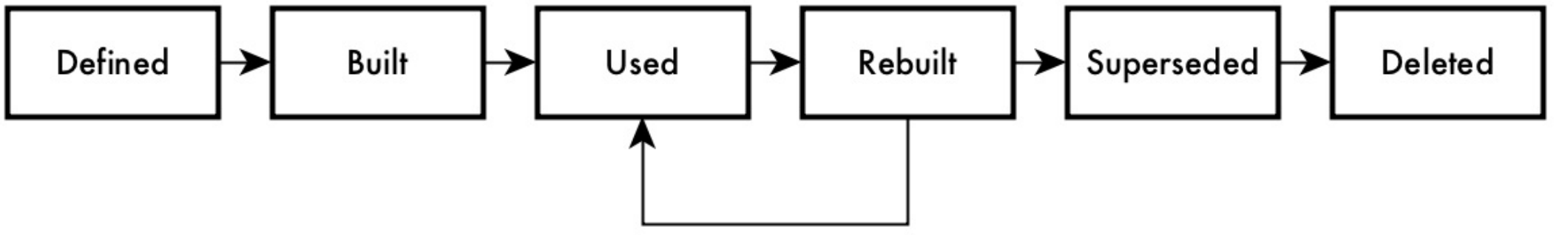
Life Cycle Of Aggregate Tables

The following diagram summarizes the stages in the life cycle of an aggregate table.



Defined

System-Defined Tables

The engine determines that an aggregate table is needed. The determination is based on predicted need or on query history, statistics, and other criteria. The engine creates the definition of the aggregate table.

User-Defined Tables

You define an aggregate table that conforms to one of the three use cases discussed in When to Define Your Own Aggregate Tables.

Built

System-Defined Tables

The next build process for the corresponding model builds the first instance of the aggregate table.

User-Defined Tables

You deploy the definition of an aggregate table after you create that definition. After you do this, the next build process for the model builds the first instance of the new table.

Used

The engine directs queries to the aggregate table, if those queries can be satisfied by it. The engine also continuously evaluates the quality of the aggregate table for such queries.

Rebuilt

Over time, as more data is ingested into the cluster and existing data is updated, the current instance of an aggregate table becomes out of date. Rebuilding the table brings it up to date. Rebuilding is done per model, so all of the aggregate tables for a model are rebuilt (or built, if a table is new) in one batch.

Superseded

System-Defined Tables

Demand-Defined Tables

The engine eventually determines, based on criteria such as query history and statistics, that a different system-defined aggregate table could better satisfy queries. The new table could be entirely new or based on the current aggregate table. The engine creates a definition for the new aggregate table and, after the first instance of this table is built, stops using the most recent instance of the old aggregate table.

The engine could determine that removing an aggregate table would help reduce the overhead of managing the aggregate tables for a model. There are two situations in which the engine might determine this:

- The metrics in the aggregate table are a subset of the metrics that are in a different aggregate table that also has the same dimension columns. In this situation, there is no need to manage the smaller aggregate table.
- Two aggregate tables contain the same dimension columns, but the each contain different metrics. In this situation, the engine might create a new table that is a union of both, replacing the two original tables.

Prediction-Defined Tables

If a catalog is redeployed, changes in its models might cause the engine to generate different prediction-defined aggregates for those models. Changes to the metrics in a fact dataset could result in a different all-member aggregate being defined for that table. Changes to a dimension could result in a different dimension-only aggregate being defined on the dimension; if no dimension-only aggregate is defined on a dimension that has changed, the engine might define one.

User-Defined Tables

The engine never supersedes user-defined aggregate tables.

Deleted

System-Defined Tables

The engine retains the definition and the instances of the old aggregate table for comparisons and for usage statistics. Definitions are retained indefinitely because they have statistics associated with them that are useful, and there is negligible cost involved in retaining them. Eventually, however, the engine deletes instances according to two sets of criteria:

First set of criteria

Desired Number of Instances: This number sets an upper bound on the set of current and inactive instances for the engine to retain. You can change this number. By default, it's 100.

Maximum Number of Excess Instances: Rather than prune every time it exceeds the desired number by building a new instance, the engine can wait until there are a number of instances that are candidates for deletion. You can also change this number. By default, it's 10.

Second set of criteria

When the maximum number of excess instances is reached, the engine uses two criteria to rank all of the instances: which instances were the least recently used and which instances are the least frequently used.

To put all of this together, suppose that the desired number of instances is set to 100 and the maximum number of excess instances is set to 10. When the total number of instances reaches 110, the engine ranks the instances by the two sets of criteria. It then deletes the instances that are ranked 101 through 110.

User-Defined Tables

If you no longer want instances of a particular user-defined aggregate table to be built, you use the Design Center to delete the definition of that table. After doing so, you redeploy the catalog that contains the corresponding model to remove the definition from the list of aggregate-table definitions for the model. You can do this even if instances of the definition still exist on the cluster.

If you want to delete instances of a particular user-defined aggregate table, you must delete them manually, not through the Design Center.