Segment cache architecture

Each Mondrian segment cache node, regardless of which configuration it uses, loads the segments required to answer a given query into system memory. This cache space is called the query cache, and it is composed of hard Java references to the segment objects. Each individual node must have enough memory space available to answer any given query. This might seem like a big limitation, but Mondrian uses deeply optimized data structures which usually take no more than a few megabytes, even for queries returning thousands of rows.

Once the query finishes, Mondrian will usually try to keep the data locally, using a weak reference to the segment data object. A weak reference is a special type of Java object reference which doesn't force the JVM to keep this object in memory. As the Mondrian node keeps answering queries, the JVM might decide to free up that space for something more important, like answering a particularly big query. This cache is referred to as the local cache.

The local cache can be switched on or off by editing the Pentaho Analysis EE configuration file and modifying the value (set it to true or false) of the DISABLE_LOCAL_SEGMENT_CACHE property. Setting this property will not affect the query cache.

Note: The segment cache features explained in this section are for very large OLAP deployments.

Parent Topic

- Work with data
  
  You can refine your Pentaho relational metadata and multidimensional Mondrian data models. You can also learn how to work with big data.

Child Topics

- How the Analysis engine uses memory
- Cache control and propagation

How the Analysis engine uses memory

This is the order in which Mondrian will try to obtain data for a required segment once a query is received:

Procedure

1. The node will parse the query and figure out which segments it must load to answer that particular query
2. It checks into the local cache, if enabled.
3. If the data could not be loaded from the local cache, it checks into the external segment cache, provided by the Pentaho Analysis plugin, and it places a copy inside the query cache.
4. If the data is not available from the external cache, it loads the data form SQL and places it into the query cache.

5. If the data was loaded form SQL, it places a copy in the query cache and it sends it to the external cache to be immediately shared with the other Mondrian nodes.

6. The node can now answer the query.

7. Once the query is answered, Mondrian will release the data from the query cache.

8. If the local cache is enabled, a weak reference to the data is kept there.

**Cache control and propagation**

The CacheControl API allows you to modify the contents of the cache of a particular node. It controls both the data cache and the OLAP schema member cache. The API is documented in the Mondrian project documentation at [http://mondrian.pentaho.com](http://mondrian.pentaho.com).

When flushing a segment region on a node, that node will propagate the change to the external cache by using the SegmentCache SPI. If the nodes are not using the local cache space, then the next node to pick up a query requiring that segment data will likely fetch it again through SQL. Once the data is loaded from SQL, it will again be stored in the external segment cache.

You should not use the local cache space when you are using the external cache. For this reason, it is disabled by default in Pentaho Analysis Enterprise Edition.
Using the local cache space on a node can improve performance with increased data locality, but it also means that all the nodes have to be notified of that change. Mondrian nodes do not propagate the cache control operations among the members of a cluster. If you deploy a cluster of Mondrian nodes and do not propagate the change manually across all of them, then some nodes will answer queries with stale data.

Parent Topic

• Segment cache architecture