Elements of Dashboard Design
- Caches

Cache – Data Model

- The Cache data source allows for easy definition of a memory resident data cache that stores real-time data.

- The Cache data source can store current scalar values or tabular data with optional filters, as well as time-stamped scalar or tabular data (up to a configurable number of entries), which keeps only the most current information and discards older entries.

- A cache history can be configured to time-stamp incoming data or to recognize a time-stamp already included in incoming data. You can also optionally pre-load a cache with data from another source, e.g., if you have historical information stored in a database.

- The Cache data source was specifically designed to enable high-speed analytic processing of real-time data and the comparison of current real-time values against history data, either visually or as input to alert rules that activate automated behavior.

- The Cache data source allows for much higher performance than the traditional means of either performing analytics on information stored in a database or querying it back out of the database when it is necessary to make calculations.

Cache – Setup

To use a cache object, you must

1. Create cache definition objects and attach them to real time data. Cache objects can be attached to tabular data or scalar (type double) data.
2. Configure properties of cache definition object (e.g., specify unique CacheName)
3. Save cache definition objects in a cache definition file (.rtv).
4. Load cache definition files into memory.

Once the cache definitions are in memory, then the individual caches become available as a Cache

Data Source for attaching to graphical objects.

NOTE:

- Every cache must have a unique CacheName.
- Two types of cache objects are available: table caches and double caches
- Caches persist data for the lifetime of the RTView instance.

Cache – Creation and Configuration of Double Cache

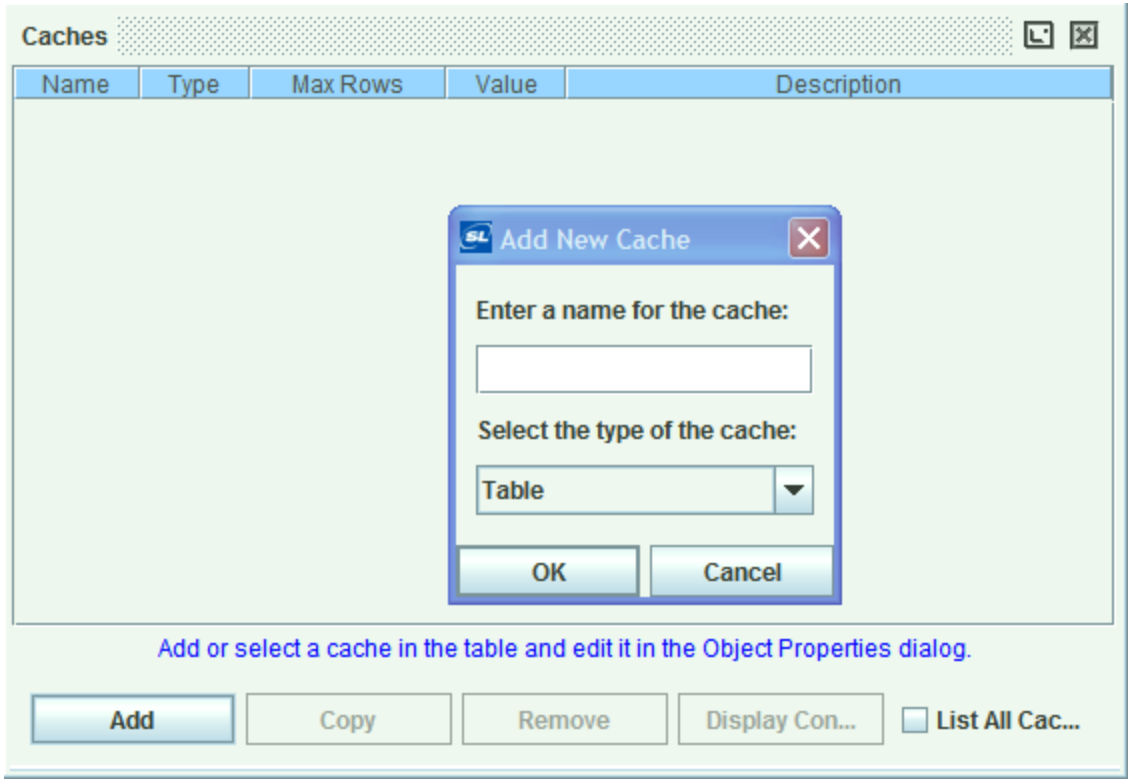
Double Cache: useful for storing real-time scalar data as doubles

Properties

- maxNumberOfRows: Number of rows in the History table. Default is 100. Older entries will be deleted from cache. If 0, no History table is created and only the Current table will be available.
- value: Attach your input scalar data to this property
- cacheName: A unique name for the cache. Required
- initialTable: Input table containing initial values for the cache. This table must contain the same column(s) as the input table attached to valueTable, as well as a timestamp column if timestampColumnName is specified.
- resetTrigger: Determines when the caches will be cleared. Any updates to this property (i.e., change in value) will trigger a clear.
- timestampColumnName: Specify the name of a timestamp column. If the specified timestampColumnName is not found in the incoming data, a column will be inserted and each row will contain the time that row was received.

Cache – Creation and Configuration of Table Cache

Table Cache: useful for storing real-time tabular data



Cache – Creation and Configuration of Table Cache

Properties

- maxNumberOfRows: Number of rows in the History table. Default is 100. Older entries will be deleted from cache. If 0, no History table is created and only the Current table will be available.
- valueTable: Attach your input tabular data to this
- cacheName: A unique name for the cache. Required.
- deleteTable: For an indexed table cache, this table determine which rows will be removed from the cache tables. Rows will be removed in the caches if their index column values match those of a row in the deleteTable.
- indexColumnNames: A semicolon (;) separated list of column names to be used as index columns.
- initialTable: Input table containing initial values for the cache. This table must contain the same column(s) as the input table attached to valueTable, as well as a timestamp column if timestampColumnName is specified.
- resetTrigger: Determines when the caches will be cleared. Any updates to this property (i.e., change in value) will trigger a clear.

Cache – Properties

Properties

- timestampColumnName: Specify the name of a timestamp column. If the specified timestampColumnName is not found in the incoming data, a column will be inserted and each row will contain the time that row was received.

- rowExpirationMode Specifies whether to expire rows in the current table and, if so, the mode for doing so. Possible values are:

  Off Row expiration is disabled.

  Mark An Expired column is added to the current table and its value is set to true in each expired row.

  Delete Expired rows are deleted from the current table.

- rowExpirationTime: This property is available when the rowExpirationMode property is set to Mark or Delete. This property specifies the amount of time that rows will be kept in the table according to their timestamp. Specify the duration in seconds, or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows:

  y years (365 days)

  M months (31 days) w weeks (7 days)

  d days h hours

  m minutes s seconds

Cache – Properties

Properties (cont)

- historyColumnNames: Enter a semicolon (;) separated list of column names from the cache current table to be stored in the cache history. The order in which the column names are specified does not affect the order in which they appear in the history table, they appear in the same order as they do in the current table.

- historyTimeSpan: Previously named timeSpan, this property specifies the amount of time that rows will be kept in the History table according to their timestamp. Specify the duration in seconds or specify a number followed by a single character indicating the desired time interval (e.g. 15m for 15 minutes). Valid characters are as follows:

  y years (365 days)

  M months (31 days) w weeks (7 days)

  d days h hours

  m minutes s seconds

Exercises

Ex 1: Data Source for Cache

1. The cache exercise assumes a JMS simulated data source.

   cd %RTV_HOME%\demos\emsdemo

   run_exchange_simulator

   cd %RTV_HOME%\demos\Class

2. Bring up the Builder
3. Tools->Options->JMS

   In the “JMS Connections” Tab set connection to “default”

   The “JMS Messages” tab, add

   “Tucon.Orders”, “Tucon.Trades”, “Tucon.Quotes”

4. Save. From the “Confirm” dialog pick NO, to save locally.

Exercises

Ex 2: Create a cache object for JMS

1. In the builder, add a cache with Tools>Cache

   name of cache: JmsSimData

   type Table

2. Set the object properties of the cache:

   valueTable: attach to JMS data source.

   indexColumnNames:Symbol;Account

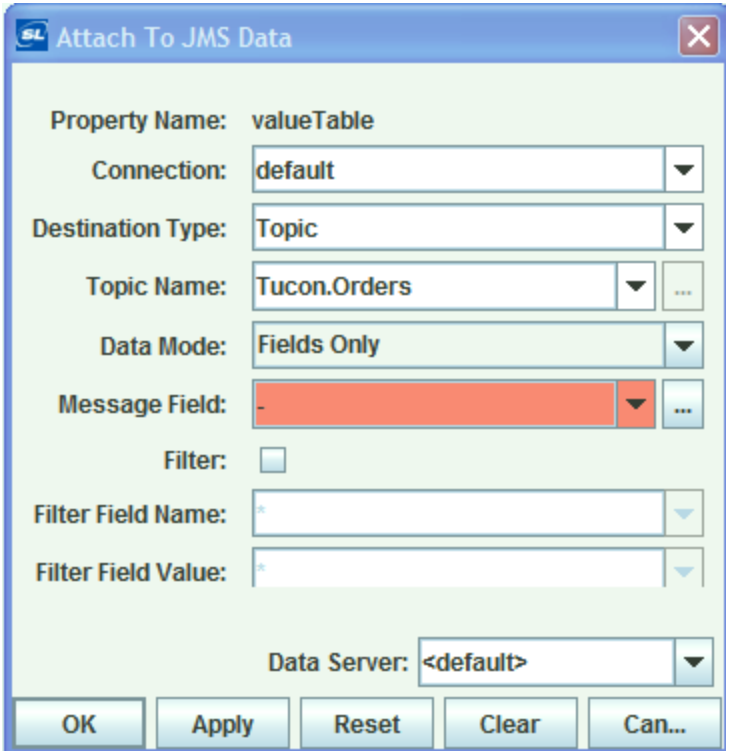
   maxNumberOfRows:1000

3. Tools->Options->Caches ,

   Add sim_jms_cache.rtv.

   Apply and Save.

4. File>Save As sim_jms_cache.rtv.



Exercises

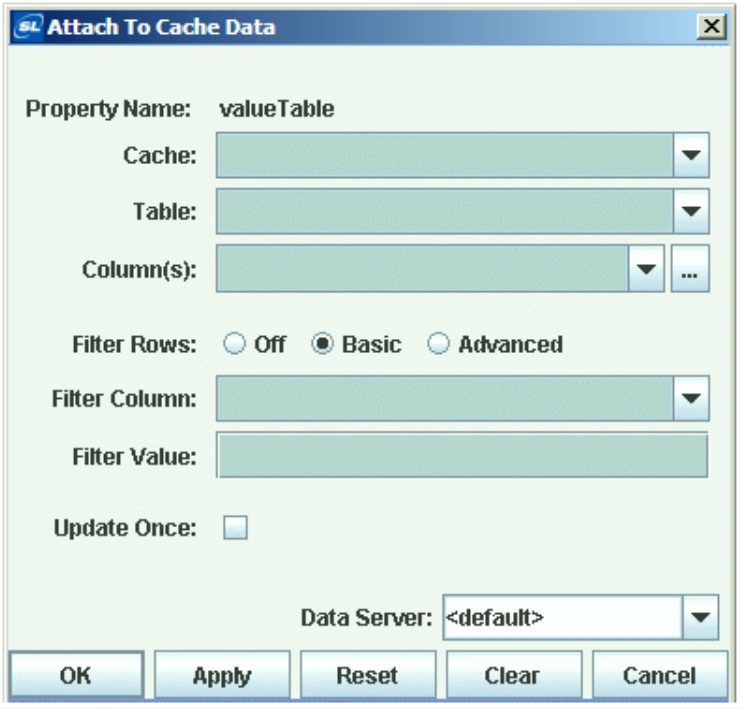
Ex 3: Register the cache definition file

1. In the builder, select Tools>Options>Caches
2. Add the file from Ex 2.
3. Save settings

The settings will be saved as CACHEOPTIONS.ini

Cache – Data Source

Caches act as another data source within RTView. From the Object Properties window you can access the Attach to Cache Data dialog to connect an object property to this in-memory data. Once a property has been attached to cache data, it receives continuous updates.



Cache – Debugging

Debugging Flag

-cachedstrace:N shows various cache loading metrics

where N=1,…,5

Debug Cache Tables

Specific cache attachments will show debug tables

Cache: RTViewDs

Table: Tables

Fields: *

will list all caches in memory and the # of rows and columns and % full

Exercises

Ex 4: Attach to a cache

1. Create a new session, File> New.
2. Add 2 tables from the object palette
3. Attach one table to cache current table
4. Attach the other table to cache history table
5. Save As class_tables.rtv

Ex 5: Create cache definitions for other data sources (Hawk, EMS, etc)

1. Repeat Ex1-Ex4 for other data sources covered earlier. Assign appropriate index columns.

NOTE: For Hawk sources, it is recommended that Hawk internal caching is disabled and that Hawk index columns are turned on.

Cache – Data Compaction

- Primary compaction operates on in-memory cache tables, condensing a cache history table by periodically applying a calculation (for example, average, min, max, etc.) to each column. It then stores a single result row in another cache table.

- Primary compaction is configured using the Data Compaction: Primary (in-memory) properties. This feature complements the Historian data compaction, or secondary compaction feature, which operates on database tables.

- Secondary compaction can be configured to compact rows in database tables as the data "ages". Secondary compaction is configured using the Data Compaction: Secondary (Historian) properties.

Cache – Data Compaction Properties

Properties

- condenseRowsFlag: If selected, enables the primary compaction feature for the cache. This property makes the following tables available (in addition to the tables available by default, the Current and History tables.):

  current_condensed: This table stores the most recent row of condensed data for each unique index column value combination.

  history_condensed: This table stores condensed data history, limited by the cache historyTimeSpan and/or maxNumberOfHistoryRows properties

- condenseRowsGroupBy: A string that specifies the calculation to be performed on each data column in the cache table

- condenseRowsInterval:The time interval at which the cache history is condensed. This property is available when the condenseRowsFlag property is selected.

  Specify a value using the following format:

  NNu

  where NN is a number and u is a single character. Valid characters are as follows:

  w weeks (7 days)

  d days h hours

  m minutes s seconds

  For example, to specify a ten minute interval: 10m

Cache – Historian

You must specify your Cache definition (.rtv) file as an Historian data configuration (,rtv) file in order use the following object properties.

Properties

- currentTableName:Enter the name of a table in your Historian database in which to store the cache's current data table. If you specify a currentTableName and no table by that name exists in your database, then one will be created for you the first time you run the Historian.

- databaseName:Name of your Historian database that contains history and/or current tables for this cache. If left blank, the database name RTVHISTORY is assumed.

- historyTableName:Enter the name of a table in your Historian database in which to store the cache history data table. If you specify a historyTableName and no table by that name exists in your database, one is created for you the first time you run the Historian