Appendix: Generic Database Datastore

2013-09-10

Overview

The generic database datastore can be used to provide general access to database tables and views, for example with the `ReadTableFromDataStore()` command. Properly configured, it also will allow reading time series using the `ReadTimeSeriesFromDataStore()` command, and writing time series using the `WriteTimeSeriesToDataStore()` command. The trade-off for generic datastores is that although tables and views can be accessed in a generic way, there is no specific application programming interface (API) to deal with the intricacies of the database and converting tables to more complex data objects like time series may be limited. See also the `TableToTimeSeries()` command, which will convert a table into time series.

The datastore internally corresponds to an Open Database Connectivity (ODBC) connection. The connection can be defined one of two ways:

- Define an ODBC connection using Windows tools. The advantage of this approach is that database authentication occurs through the ODBC connection. The disadvantage is that the connection may use a generic database driver that does not perform as well as vendor drivers. This approach is used when the `DatabaseEngine` and `OdbcName` configuration properties are defined for the datastore.
- Provide connection information via `DatabaseEngine`, `DatabaseServer`, `DatabaseName`, and potentially login configuration properties, and allow the software to use a vendor-specific JDBC (Java Database Connectivity) driver, which is generally optimized for the database software. The disadvantage of this approach is that advanced authentication interfaces have not been implemented (this may or not be an issue depending on the security enabled for the database).

Limitations

The following limitations apply to the generic database datastore:

- Database permissions control which tables and views are accessible and consequently protected tables may not be visible in software or may generate errors if attempts are made to manipulate outside of permissions.
- An attempt is made in the `ReadTableFromDataStore()` command to list tables and views for selection. However, the ability to filter out system tables is limited because some database drivers do not implement required functionality. For example, the SQL Server JDBC driver does not allow generic filtering of system tables and a work-around has been implemented to remove known system table and view names from lists displayed to users.
- Table column properties in TSTool are determined from database column metadata. Although support for common data types has been implemented, some data types may not be fully supported. If a database column type is not supported, the default is to translate the column data to strings in the output table. Additional functionality will be added in the future.
- Although database column properties can specify the width and precision for floating point data, some database metadata is inaccessible, causing data-handling or visualization issues. For example, the SQL Server metadata defaults result in the precision of floating point numbers

(called "precision" in TSTool and "scale" in SQL Server column properties) to be set to zero. The work-around is that any floating point data column that has a precision of zero is treated as having a precision of 6 digits after the decimal point.

Datastore Configuration Files

A datastore is configured by enabling the datastore in the main *TSTool.cfg* configuration file and creating a datastore configuration file for the connection. Configurations are processed at software startup to enable datastores. An example of the TSTool configuration file is shown below. Multiple datastores can be defined using the `[DataStore:DataStoreName]` syntax. Properties for each datastore are specified in an accompanying configuration file described below.

```
# Configuration file for TSTool

...properties omitted…

# Startup datastores (note that datastore name in config file takes precedence)

[DataStore:SomeDatabaseDataStore]
ConfigFile = "SomeDatabaseDataStore.cfg"
```

TSTool Configuration File with Generic Database Datastore Properties

The following illustrates the generic database datastore configuration file format, which in this example is located in the same folder as the TSTool configuration file.

```
# Configuration information for "SomeDatabaseDataStore" datastore (connection).
# The user will see the following when interacting with the datastore:
#
# Type – GenericDatabaseDataStore (required as indicated)
# Name - database identifier for use in applications, for example as the
#     input type/name information for time series identifiers (usually a short string)
# Description - database description for reports and user interfaces (a sentence)
# Enabled – whether the datastore is enabled (default=True)
#
# The following are needed to make the low-level data connection:
#
# DatabaseEngine - the database software (SqlServer)
# OdbcName – ODBC name (specify this OR the following properties)
# DatabaseServer - IP or string address for database server
# DatabaseName - database name used by the server
# SystemLogin – the login to be used for the database connection
# SystemPassword – the password to be used for the database connection
#
# Property values can use the notation "Env:xxxx" to use an environment variable,
# "SysProp:xxxx" to use a JRE system property, or "Prompt" to prompt the user for
# the property value (system console is used – not suitable for TSTool startup from
# the Start menu)

Type = "GenericDatabaseDataStore"
Name = "SomeDatabaseDataStore"
Description = "Database on some server"
Enabled = True
DatabaseEngine = "SqlServer"
# Specify OdbcName…
OdbcName = "OdbcName"
# Or, specify the following…
DatabaseServer = "ServerName"
DatabaseName = "DatabaseName"
SystemLogin = "LoginForConnection"
SystemPassword = "PasswordForConnection"
```

Generic Database Datastore Configuration File

The `DatabaseEngine` can be one of the following values, and is used to control internal database interactions, such as properly formatting date/time strings for SQL statements:

- `Access` – Microsoft Access database
- `Excel` – Microsoft Excel workbook (first row of worksheet should be the column names, column types are determined by scanning rows (independent of the ***Rows to Scan*** value in the ODBC DNS setup); refer to sheet in SQL as "`Select * from [Sheet1$]`")
- `H2` – H2 database
- `Informix` – INFORMIX database
- `MySQL` – MySQL database
- `Oracle` – Oracle database
- `PostgreSQL` – PostgreSQL database
- `SQLServer` – Microsoft SQL Server database

TSTool, which is written in Java, is distributed with the software drivers for the above databases only if datastores have been implemented that use a database product. For example, the State of Colorado's HydroBase database is implemented in SQL Server and consequently the SQL Server driver is distributed with TSTool. Other drivers (e.g., Access via ODBC) depend on installation of the database software, which typically includes the ODBC drivers. Some of the databases listed above have only been used in development and software support may be out of date. If in doubt, contact the software developers and the issue will be evaluated. More databases can be supported if the number of users increases.

The following example illustrates how to configure a datastore for an ODBC DSN connection to an Access database:

```
# Configuration information for Nebraska DNR development database.
# Properties are:
#
# The user will see the following when interacting with the datastore:
#
# Type - required to be GenericDatabaseDataStore
# Name - datastore identifier used in applications, for example as the
#     input type information for time series identifiers (usually a short string)
# Description - datastore description for reports and user interfaces (short phrase)
# DatabaseEngine - the database software
# OdbcName - the Open Database Connectivity Data Source Name (ODBC DSN), configured
#     in Windows Control Panel
#

Type = "GenericDatabaseDataStore"
Name = "ExampleDatabase"
Description = "Example Access Database"
DatabaseEngine = "Access"
OdbcName = "ExampleDatabase"
```

Generic Database Datastore Configuration File Using ODBC DSN Properties

The following example illustrates how to configure a generic datastore for a SQL Server database, using separate database connection properties (NOT using an ODBC DSN). Such configurations may not be suitable because it may be desirable to configure login information in an ODBC DSN. The following is appropriate if a generic read-only service account is configured.

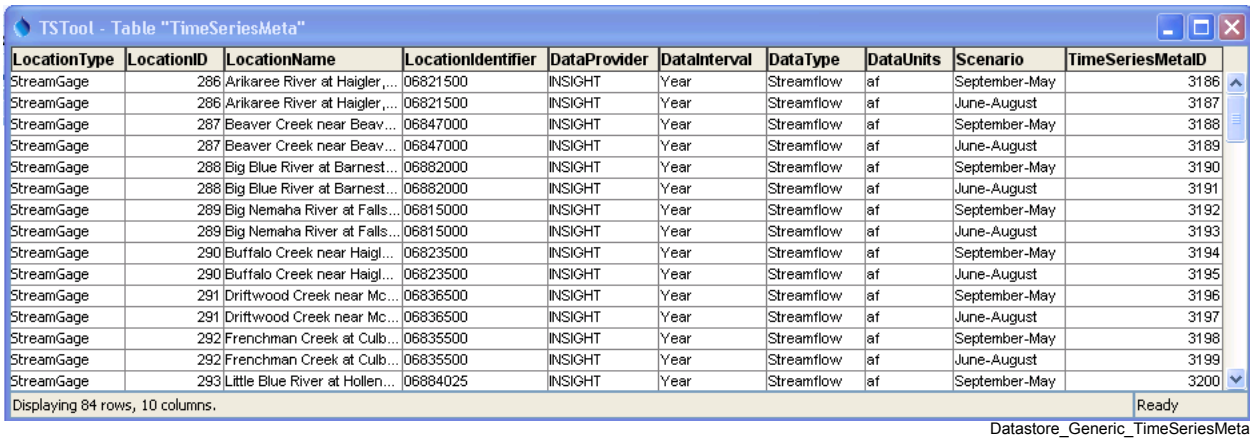
```
# Configuration information for Nebraska DNR development database.
# Properties are:
#
# The user will see the following when interacting with the datastore:
#
# Name - datastore identifier used in applications, for example as the
#     input type information for time series identifiers (usually a short string)
# Description - datastore description for reports and user interfaces (short phrase)
#

Type = "GenericDatabaseDataStore"
Name = "NDNR-Cascade-WaterRights"
Description = "INSIGHT Development Database"
DatabaseEngine = "SqlServer"
DatabaseServer = "xxxxx"
DatabaseName = "WaterRights"
SystemLogin = "guest"
SystemPassword = "guest"
```

Generic Database Datastore Configuration File Using Database Connection Properties

The `ReadTimeSeriesFromDataStore()` and `WriteTimeSeriesToDataStore()` commands recognize additional datastore configuration properties, which allow the commands to read time series from and write time series to a general database design. To use these command, time series metadata in the database are mapped to TSTool time series identifiers (`LocationType:Location.DataSource.DataType.Interval.Scenario`) and other key properties can be provided (e.g., data units). Core tables in a compatible database design contain the following data:

- Definitions, such as data types, data units, data source (providers)
- Locations
- Time series metadata (typically relationships to the above)
- Time series data records (associated with a time series metadata record)

Relationships between the above tables that use database keys can be complicated to configure. Consequently, it is recommended that a database view be configured to provide time series metadata as a list of time series and associated properties. For performance reasons, it may also be appropriate to copy the view to a "materialized" table and create indexes on the table. The following figure illustrates time series metadata from a database view:


TSTool - Table "TimeSeriesMeta"

LocationType	LocationID	LocationName	LocationIdentifier	DataProvider	DataInterval	DataType	DataUnits	Scenario	TimeSeriesMetaID
StreamGage	286	Arikaree River at Haigler,...	06821500	INSIGHT	Year	Streamflow	af	September-May	3186
StreamGage	286	Arikaree River at Haigler,...	06821500	INSIGHT	Year	Streamflow	af	June-August	3187
StreamGage	287	Beaver Creek near Beav...	06847000	INSIGHT	Year	Streamflow	af	September-May	3188
StreamGage	287	Beaver Creek near Beav...	06847000	INSIGHT	Year	Streamflow	af	June-August	3189
StreamGage	288	Big Blue River at Barnest...	06882000	INSIGHT	Year	Streamflow	af	September-May	3190
StreamGage	288	Big Blue River at Barnest...	06882000	INSIGHT	Year	Streamflow	af	June-August	3191
StreamGage	289	Big Nemaha River at Falls...	06815000	INSIGHT	Year	Streamflow	af	September-May	3192
StreamGage	289	Big Nemaha River at Falls...	06815000	INSIGHT	Year	Streamflow	af	June-August	3193
StreamGage	290	Buffalo Creek near Haigl...	06823500	INSIGHT	Year	Streamflow	af	September-May	3194
StreamGage	290	Buffalo Creek near Haigl...	06823500	INSIGHT	Year	Streamflow	af	June-August	3195
StreamGage	291	Driftwood Creek near Mc...	06836500	INSIGHT	Year	Streamflow	af	September-May	3196
StreamGage	291	Driftwood Creek near Mc...	06836500	INSIGHT	Year	Streamflow	af	June-August	3197
StreamGage	292	Frenchman Creek at Culb...	06835500	INSIGHT	Year	Streamflow	af	September-May	3198
StreamGage	292	Frenchman Creek at Culb...	06835500	INSIGHT	Year	Streamflow	af	June-August	3199
StreamGage	293	Little Blue River at Hollen...	06884025	INSIGHT	Year	Streamflow	af	September-May	3200

Displaying 84 rows, 10 columns. Ready

Datastore_Generic_TimeSeriesMeta

Time Series Metadata View

The following example illustrates datastore properties that are used to describe the database design so that TSTool can read and write time series. If configured, TSTool also will provide time series browsing features in the main window.

```
# Indicate the table that contains time series metadata
# Use a view that handles the foreign keys so that this configuration is simpler
TimeSeriesMetaTable = "view_TimeSeriesMeta_All"

# Columns in the metadata table for the time series identifier parts
# Only the TimeSeriesMetaTable_MetaIdColumn column contains a primary key
TimeSeriesMetaTable_LocationTypeColumn = "LocationType"
TimeSeriesMetaTable_LocationIdColumn = "LocationIdentifier"
TimeSeriesMetaTable_DescriptionColumn = "LocationName"
TimeSeriesMetaTable_DataSourceColumn = "DataProvider"
TimeSeriesMetaTable_DataTypeColumn = "DataType"
TimeSeriesMetaTable_DataIntervalColumn = "DataInterval"
TimeSeriesMetaTable_ScenarioColumn = "Scenario"
TimeSeriesMetaTable_DataUnitsColumn = "Units"
TimeSeriesMetaTable_MetaIdColumn = "TimeSeriesMetaID"

# Data table to use for time series data, in this case based on the interval
# (could also use the data type if DBA wanted to split out that way)
# Time series formatting strings (e.g., %I = interval like Year) and time series
# properties (e.g., ${TS:Property}) can be used to specify the data table.
TimeSeriesDataTable = "TimeSeriesData%I"

# Properties that describe columns for time series data
# The column names can vary based on date/time precision, etc.
TimeSeriesDataTable_MetadataIdColumn = "TimeSeriesMetaID"
TimeSeriesDataTable_DateTimeColumn = "
TimeSeriesDataYear:Year,TimeSeriesDataMonth:Date,TimeSeriesDataDay:Date"
TimeSeriesDataTable_ValueColumn = "Value"
TimeSeriesDataTable_FlagColumn = "Flag"
```

Generic Database Datastore Time Series Configuration Properties

The `TimeSeriesMetaTable` table/view is used with time series identifier information as a lookup to determine the internal database key for the time series data records, indicated by `TimeSeriesMetaTable_MetaIdColumn` in the metadata table and `TimeSeriesDataTable_MetadataIdColumn` in time series data table(s). Once the internal key for the time series is determined, it is used to read or write the individual time series records.

The following table summarizes configuration properties that are used for time series metadata.

Configuration Properties for Time Series Metadata

Property	Description	Default
`TimeSeriesMetaTable`	The name of the database table/view that contains time series metadata, essentially a catalog of time series in the database.	None – must be specified.
`TimeSeriesMetaTable_LocationTypeColumn`	The name of the column in the time series metadata table for the location type. Typically location type abbreviations are stored in a separate table and are joined to the metadata view.	Location type will not be used to identify time series.
`TimeSeriesMetaTable_LocationIdColumn`	The name of the column in the time series metadata table for the location identifier. A string is typically used that allows lookup in location-specific tables.	
`TimeSeriesMetaTable_DescriptionColumn`	The name of the column in the time series metadata table for the description, typically the location name..	
`TimeSeriesMetaTable_DataSourceColumn`	The name of the column in the time series metadata table for the data source. Typically data source abbreviations are stored in a separate table and are joined to the metadata view.	Location source will not be used to identify time series.
`TimeSeriesMetaTable_DataTypeColumn`	The name of the column in the time series metadata table for the data type. Typically data type abbreviations are stored in a separate table and are joined to the metadata view.	None – must be specified.
`TimeSeriesMetaTable_IntervalColumn`	The name of the column in the time series metadata table for the interval (containing `Day`, `Month`, etc. as per time series identifiers). Typically data interval abbreviations are stored in a separate table and are joined to the metadata view.	None – must be specified.
`TimeSeriesMetaTable_ScenarioColumn`	The name of the column in the time series metadata table for the scenario (containing blanks or strings). Typically scenario abbreviations are stored in a separate table and are joined to the metadata view.	Scenario will not be used to identify time series.
`TimeSeriesMetaTable_DataUnitsColumn`	The name of the column in the time series metadata table for the data units (containing blanks or strings). Typically units abbreviations are stored in a separate table and are joined to the metadata view.	Units for time series will be blank.
`TimeSeriesMetaTable_MetadataIdColumn`	The name of the column in the time series metadata table for the metadata primary key, which is the relationship to data records.	None – must be specified.

The following table summarizes configuration properties that are used for time series data tables.

Configuration Properties for Time Series Data

Property	Description	Default
`TimeSeriesDataTable`	The name of the database table/view that contains time series data records. To facilitate multiple data tables, this property can be specified with the following special format values, which will cause the table name to be dynamically determined. Because the value must be defined for reading and writing, dynamic values are confined to data that are available in both cases: • `%I` – use the time series data interval • `%T` – use the time series data type In the future additional options for specifying the time series table may be provided.	None – must be specified.
`TimeSeriesDataTable_MetaIdColumn`	The name of the column in the time series data table for the metadata ID, which has the same value as the metadata ID in the metadata table (`TimeSeriesMetaTable_MetadataIdColumn`) and groups time series records.	None – must be specified.
`TimeSeriesDataTable_DateTimeColumn`	The name of the column in the time series data table for the date/time. Specify a single value to indicate the date/time column is the same for all time series data tables, or use the following syntax to indicate column names for each table: `DataTable1:DateTimeColumn1, DataTable2:DateTimeColumn2,…` The following constraints apply: • If the time series interval is `Year` and the column type is integer, the values are assumed to be the 4-digit year in the time series. • Otherwise if the column type is timestamp, the date/time values are used directly and are handled according to the time series date/time precision.	None – must be specified.
`TimeSeriesDataTable_ValueColumn`	The name of the column in the time series data table for the data values, currently must be the same for all time series data tables. Null values in the column are interpreted as missing data.	None – must be specified.
`TimeSeriesDataTable_FlagColumn`	The name of the column in the time series data table for the data flags, currently must be the same for all time series data tables.	Data flags will not be used with the time series.

This page is intentionally blank.