Technical Note

Database Drivers' Options

Author: R&D Department

Publication date: September 10, 2007

Revision date: September 2010

Database Drivers' Options

Overview

You can change the behaviour of REP++ database drivers using options. These options can be set in two ways:

- Method A: Using a switch in the SQLD.INI file.
- Method B: Using the **Connection.SetOption()** method.

This technical note describes the two methods along with the new options.

NOTE: this document applies to REP++ starting from version 7.0.

Method A

You can insert a section in the SQLD.INI file (bin\sdwin32 directory) with the name 'CONNECTION:*ConnectionName*' where *ConnectionName* is the name of the connection. This section can contain the option for the related database driver in the form of 'OptionName=*OptionValue*'. The option name must be one of the valid options for the related database driver described below. For compatibility purposes, the database prefix 'xxx_' can be removed from the name.

Example:

```
[CONNECTION:MySqlServer]
SS_CursorMode=Static
MultiRowSize=128
TransMode=Auto
```

You can use the *Login* trace to see if the options are correctly set.

Method B

You can also use the **SetOption()** method from the **Connection** object. This method uses two parameters, the option name and the option value. The full option name must be specified. The method returns the following code:

0	(ERR_NO_ERR)	Parameter set.
12	(ERR_BAD_PARAM)	Bad parameter value for this option.
67	(ERR_NOT_SUPPORTED)	This option is not supported by this database driver.

Example:

```
app.DataConnection.SetOption("SS_CursorMode", "DefResSet");
```

Options

General

The following options can be used on all database drivers.

MultiRowSize

Sets the amount of memory allocated for multi-row fetching. The value is expressed in KB. The value range must be between 0 and 1024. The default value is 0, which lets the driver decide on the amount of memory to allocate (actually 16 KB). The buffer is allocated only if the multi-row fetching option is specified when the SQL command is compiled. Changing the value of this option has an impact on newly compiled SQL commands. Commands already compiled (or kept in the cache) are not modified.

NbMaxCursor

Specifies the maximum number of SQL commands that can be opened at a given time for this connection. The value range must be between 4 and 16384. The default value is 32.

Oracle V8 Driver

ORA_NumStrict

Indicates if the database specified by this connection supports transparent cast from CHAR to NUM type. If Oracle is used as a gateway to a foreign database (like DB/2) and this database does not support automatic conversion, this option must be set to *true*. The default value for this option is *false*.

- Auto: Same as *false*.
- True: Strict.
- False: Not Strict.

ORA_NumWithComma

Indicates if the driver converts the comma contained in the received numeric value to a period. Set this option to *true* if the language setting of the database uses the comma as the decimal separator. The default value of this option is *false*.

- Auto: Same as *false*.
- True: Converts the comma to a period.
- False: Does not convert.

SQL Server OLEDB and ODBC Drivers

SS_CursorMode

Sets the type of cursor used to fetch data. SQL Server supports many options to optimize the way the data is returned from a request. You can use a default result set (firehose cursor) or one of the cursor flavors. The REP++ driver for SQL Server (using OLEDB) supports the following options:

- Auto: Use a fast forward cursor if connected to a SQL Server 2000 or older database. A default result set is used if connected to a SQL Server 2005 or newer database. This is the default option.
- Static: Use a static cursor.
- FastFoward: Use a fast forward cursor.

- DefResSet: Use a default result set.

In all cases, the driver uses a default result set if the SQL command processed is a stored procedure or a SELECT accessing BLOB values.

The fastest method to access data is normally achieved using a default result set. However, this method also has severe drawbacks (prior to SQL Server 2005).

With versions prior to SQL Server 2005, when using the default result set method, only one statement can be executed at a time for each SQL Server connection. If you use a SELECT statement, all rows must be fetched before you can execute another SQL statement of any kind. If you try to execute a second SQL statement while another one is still active, the OLEDB driver silently creates a second connection. When this second statement terminates, the connection is automatically closed. Note that these automatically opened connections are not pooled. This behaviour can cause scalability problems.

A new option (MARS – Multiple Active Result Sets) has been added starting with SQL Server 2005. This option allows multiple active result sets at a time. With this option on, the driver does not silently open new connections when more than one statement needs to be executed at the same time.

Using multi-row fetching can improve dramatically the performance when fetching a large result set with a static or fast forward cursor. In this case, specifying a large **MultiRowSize** value will have a significant impact on the performance.

The multi-row fetching parameter has no impact on performance, but might have a negative impact if a large value is specified for the **MultiRowSize** option. This is not a big surprise knowing that SQL Server already caches all the result sets on the client side. In this case, we are doing double caching, which increases overhead without benefits.

For more information about these different options, please refer to the SQL Server documentation.

SS_StartTrans

Starts a new transaction if not already started. Only for OLEDB.

SS_TransMode

Indicates when to start a new transaction. By default, a transaction is automatically started when a non-SELECT statement is processed. Only for OLEDB.

- Auto: Starts a transaction when a non-SELECT statement is executed, if a transaction is not already started.
- All: Starts a transaction when a SQL statement is executed, if a transaction is not already started.

SQLite

DBFileName

Database file name.

TmpTable

Indicates where the temporary tables are kept.

- *File*: Temporary tables are kept in a file.
- *Memory*: Temporary tables are kept in memory.

CacheSize

Cache size, in bytes.