Transactions with SQL functions

By default our SQL functions work in auto commit mode. If you like to use transactions, you need to turn off auto commit mode. You may have a script to connect to a database and after the SQL.Connect call is successful, you turn off auto commit:

```
Set Variable [ $r ; Value: MBS( "SQL.SetAutoCommit"; $Connection; "off" ) ]
```

Now you can for example run INSERT SQL commands for multiple records. If all functions returned success, you may commit on the end:

```
Set Variable [ $result2 ; Value: MBS("SQL.Commit"; $Connection) ]
```

Or in case one of them failed, you can do a rollback:

```
Set Variable [ $result2 ; Value: MBS("SQL.Rollback"; $Connection) ]
```

With all the error checking, the script is quite long. We have an example here to do two inserts and if nothing goes wrong, it will commit, otherwise it will do a rollback:

```
Set Variable [ $Connection ; Value: MBS("SQL.NewConnection") ]
# Tell plugin where PostgreSQL library is
Set Variable [ $result ; Value: MBS("SQL.SetConnectionOption"; $Connection;
"LIBPQ.LIBS"; Query::PathToSQLLibrary) ]
# Connect to database in read/write/create mode. Creates new file if none exists.
Set Variable [ $result ; Value: MBS("SQL.Connect"; $Connection;
Query::SQLConnectionString; Query::Username; Query::Password;
"PostgreSQL") ]
If [ $result ≠ "OK" ]
        Show Custom Dialog [ "Error: " & $result ]
        Set Variable [ $result ; Value: MBS("SQL.FreeConnection"; $Connection) ]
        Halt Script
End If
# auto commit off
Set Variable [ $r ; Value: MBS( "SQL.SetAutoCommit"; $Connection; "off" ) ]
# Create insert command with setting values
Set Variable [ $Command ; Value: MBS("SQL.NewCommand"; $Connection;
"INSERT INTO Persons (FirstName, LastName, City) VALUES
(:FirstName, :LastName, :City)") ]
If [ MBS("IsError") = 0 ]
        Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command;
"FirstName"; "John") ]
        If [ MBS("IsError") = 0 ]
```

```
                Set Variable [ $result ; Value: MBS("SQL.SetParamAsText";
$Command; "LastName"; "Meyer") ]
                If [ MBS("IsError") = 0 ]
                        Set Variable [ $result ; Value: MBS("SQL.SetParamAsText";
$Command; "City"; "Boston") ]
                        If [ MBS("IsError") = 0 ]
                                Set Variable [ $result ; Value: MBS("SQL.Execute";
$Command) ]
                                If [ MBS("IsError") = 0 ]
                                        Set Variable [ $result ; Value:
MBS("SQL.SetParamAsText"; $Command; "FirstName"; "Monica") ]
                                        If [ MBS("IsError") = 0 ]
                                        Set Variable [ $result ; Value:
MBS("SQL.SetParamAsText"; $Command; "LastName"; "Jones") ]
                                        If [ MBS("IsError") = 0 ]
                                                Set Variable [ $result ; Value:
MBS("SQL.SetParamAsText"; $Command; "City"; "San Francisco") ]
                                                If [ MBS("IsError") = 0 ]
                                                        Set Variable [ $result ; Value:
MBS("SQL.Execute"; $Command) ]
                                                        If [ MBS("IsError") = 0 ]
                                                                # all okay, so let's commit
                                                                Set Variable [ $result2 ; Value:
MBS("SQL.Commit"; $Connection) ]
                                                        End If
                                                End If
                                        End If
                                        End If
                                End If
                        End If
                End If
        End If
        Set Variable [ $result2 ; Value: MBS("SQL.FreeCommand"; $Command) ]
        # if anything got wrong, do a rollback
        Set Variable [ $result2 ; Value: MBS("SQL.Rollback"; $Connection) ]
End If
Set Variable [ $result2 ; Value: MBS("SQL.FreeConnection"; $Connection) ]
```

As you see we do both Commit and Rollback on success. But for the rollback there is nothing to do, so there will be nothing happening. We can easily test it by changing the parameter name "City" to "test" to get an error that the name is wrong. Or change the table name Persons to Person. Any typo in a field, parameter or table name or in the SQL syntax should trigger the rollback.

All those If lines look strange and we may have something better. For next plugin we add ClearErrors and HadErrors functions. You call MBS("ClearErrors") early in the script to reset counter to zero. Then you do various MBS calls, which may or may not cause errors. On the end you can query MBS("HadErrors") to see if you got an error. This returns the count and if we use wring table name in SQL, we get two errors from the SQL.Execute calls. So if count is zero, we call SQL.Commit here and otherwise SQL.Rollback:

```
Set Variable [ $Connection ; Value: MBS("SQL.NewConnection") ]
# Tell plugin where PostgreSQL library is
Set Variable [ $result ; Value: MBS("SQL.SetConnectionOption"; $Connection;
"LIBPQ.LIBS"; Query::PathToSQLLibrary) ]
# Connect to database in read/write/create mode. Creates new file if none exists.
Set Variable [ $result ; Value: MBS("SQL.Connect"; $Connection;
Query::SQLConnectionString; Query::Username; Query::Password;
"PostgreSQL") ]
If [ $result ≠ "OK" ]
        Show Custom Dialog [ "Error: " & $result ]
        Set Variable [ $result ; Value: MBS("SQL.FreeConnection"; $Connection) ]
        Exit Script
End If
# let the plugin count errors
Set Variable [ $r ; Value: MBS( "ClearErrors" ) ]
# auto commit off
Set Variable [ $r ; Value: MBS( "SQL.SetAutoCommit"; $Connection; "off" ) ]
# Create insert command with setting values
Set Variable [ $Command ; Value: MBS("SQL.NewCommand"; $Connection;
"INSERT INTO Persons (FirstName, LastName, City) VALUES
(:FirstName, :LastName, :City)") ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command;
"FirstName"; "John") ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command;
"LastName"; "Meyer") ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command; "City";
"Boston") ]
Set Variable [ $result ; Value: MBS("SQL.Execute"; $Command) ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command;
"FirstName"; "Monica") ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command;
"LastName"; "Jones") ]
Set Variable [ $result ; Value: MBS("SQL.SetParamAsText"; $Command; "City";
"San Francisco") ]
Set Variable [ $result ; Value: MBS("SQL.Execute"; $Command) ]
If [ MBS("HadErrors") = 0 ]
        # all okay, so let's commit
```

```
        Set Variable [ $result2 ; Value: MBS("SQL.Commit"; $Connection) ]
Else
        # we had errors!
        Set Variable [ $result2 ; Value: MBS("SQL.Rollback"; $Connection) ]
End If
Set Variable [ $result2 ; Value: MBS("SQL.FreeCommand"; $Command) ]
Set Variable [ $result2 ; Value: MBS("SQL.FreeConnection"; $Connection) ]
```

As you see we have the repeated calls without a ton of IF lines and later just check if one of the functions failed. If you like to know what failed, check the Trace function to write a log file.
The same Commit/Rollback handling should work for MySQL, MariaDB, Microsoft SQL Server, SQLite, PostgreSQL and many more database types.

Please try the new plugin once available later today and do not hesitate to contact us with your questions.