DB MANAGER

– DEVELOPER MANUAL –

INDEX

SINT Technology S.r.l.
Via delle Calandre 63
50041 Calenzano (FI)
Italy

1 INTRODUCTION

SINT Technology DB Manager is a VI toolkit for LabVIEW development environment (both 32 and 64 bit) that provides an easy to use interface between LabVIEW and multiple database engines (Oracle, MySQL and MS SQL server).

Toolkit doesn't require ODBC configuration, resulting in an easier client machine configuration and LabVIEW developed application distribution.

Due to the easy to extend code architecture further database engines support may be requested contacting the SINT Technology technical support (§6) if needed.

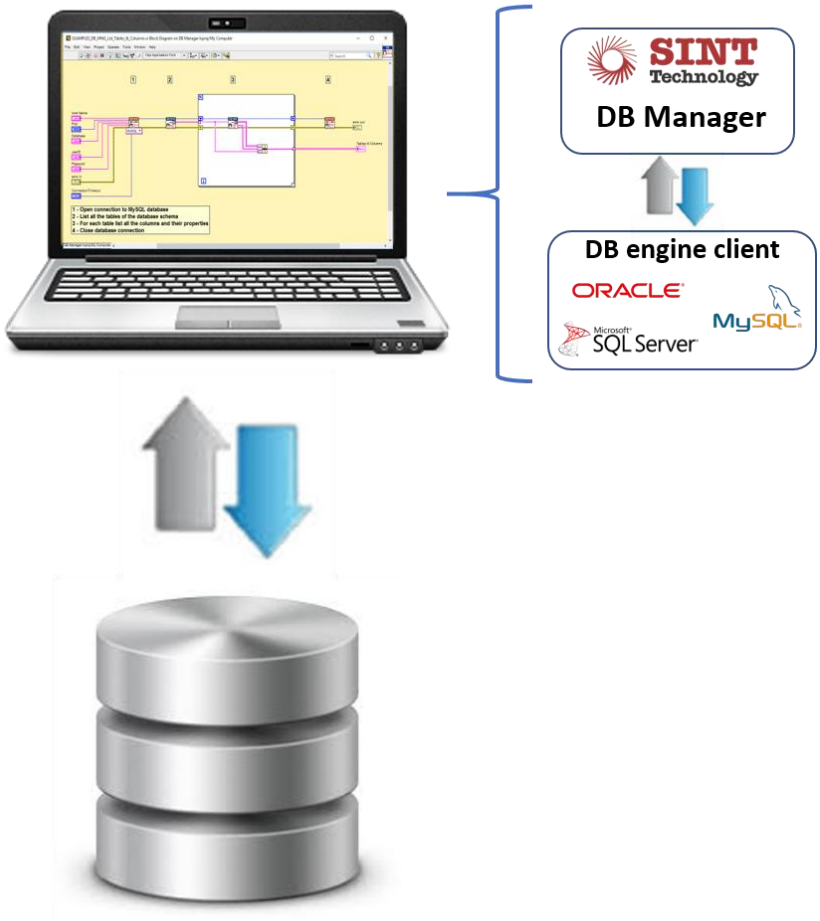

Figure 1: SINT Technology DB Manager data flow

The manual requires that you have basic understanding of the LabVIEW environment, your computer operating system and SQL.

2 SYSTEM REQUIREMENTS

• Operative System	Windows 7, 8, 8.1, 10
• LabVIEW version	2015 or later
• VI Package Manager from JKI	Package installation
• Database specific client provider	Provides specific database engine connectivity (It is mandatory that Labview bitness and DB client provider bitness must match i.e. Labview 64 bit works with 64 bit DB client provider)
• Tested on database	Oracle 11gR2 MySQL 6.3 SQL server 2014

3 FUNCTIONS

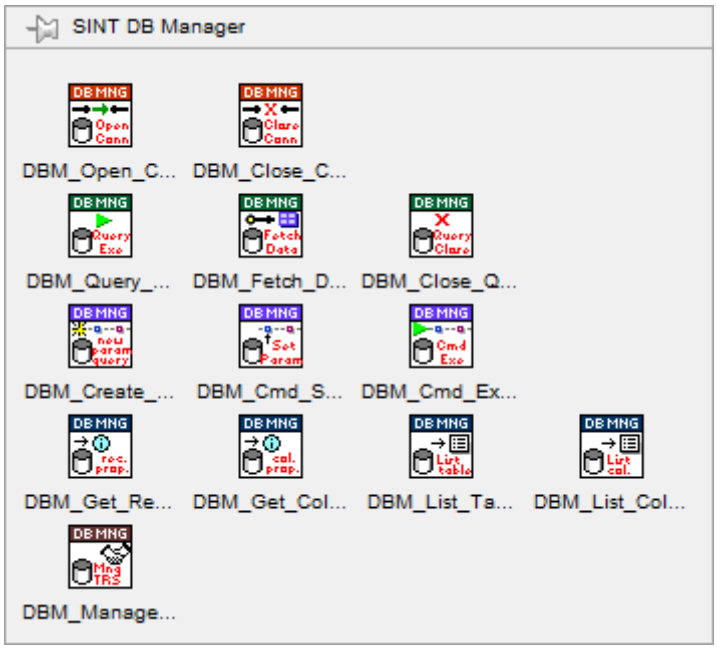

Figure 2: LabVIEW functions palette

The LabVIEW SINT Technology DB Manager palette provides tools for:

- Connecting to database engines (orange icon VIs)
- Perform SQL queries (green icon VIs)
- Perform parametrized queries (violet icon VIs)
- Retrieve database properties (blue icon VIs)
- Manage transactions (brown icon VIs)

In the following all the VIs of the DB Manager toolkit is described, showing up some example of their most common uses

3.1 Database connection management

Orange icons VIs provides tools for instantiate and destroy connection to the selected database engine.

3.1.1 Open connection to a specified database

Before perform any action on a database is required to open the communication channel to it, through the DBM_Open_Connection_Polymorphic.vi and selecting the polymorphic instance based on the database engine to be engaged.

3.1.1.1 Open connection (MySQL)

Open the connection to the MySQL database engine.

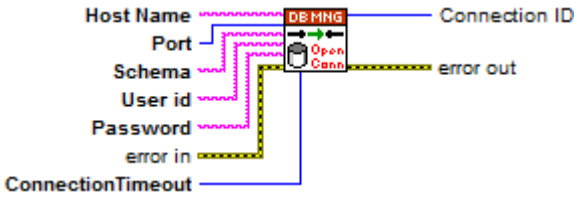

Figure 3: MySQL instance of DBM_Open_Connection.vi polymorphic VI

Parameters:

- **Host name**: specifies the address of MySQL database engine where the database is hosted.
- **Port**: specifies the number of the TCP/IP port that database server use to receive connection requests. The default value is 3306.
- **Schema**: specifies the database schema to connect to.
- **User ID**: specifies the user ID needed to access the database. You might not need to specify a userID.
- **Password**: specifies the password required to access the database for security purposes. You might not need to specify a password.
- **Connection timeout (s)**: specifies the number of seconds to wait for a connection to open, before cancelling the attempt and generating an error. The default value is 10.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection ID**: ID of DB connection
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to

error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.1.1.2 Open connection (Oracle)

Open the connection to the Oracle database engine

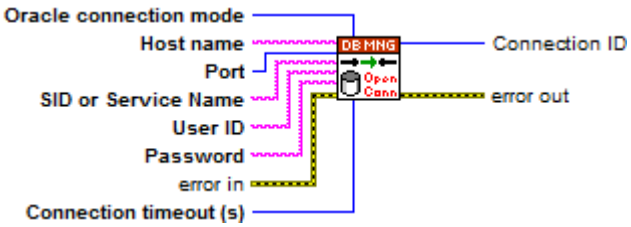

Figure 4: Oracle instance of DBM_Open_Connection.vi polymorphic VI

Parameters:

- **Host name**: specifies the address of Oracle database engine where the database is hosted.
- **Port**: specifies the number of the TCP/IP port that database server use to receive connection requests. The default value is 1521.
- **SID or Service Name**: specifies the name to use as SID or Service Name, according to the value of **Oracle connection mode**.
- **User ID**: specifies the user ID needed to access the database. You might not need to specify a userID.
- **Password**: specifies the password required to access the database for security purposes. You might not need to specify a password.
- **Oracle connection mode**: specifies the connection mode to use to connect and how to consider **SID or Service Name** string. The default value is "Use Service Name"
- **Connection timeout (s)**: specifies the number of seconds to wait for a connection to open, before cancelling the attempt and generating an error. The default value is 10.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection ID**: ID of DB connection
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.1.1.3 Open connection (SQL Server)

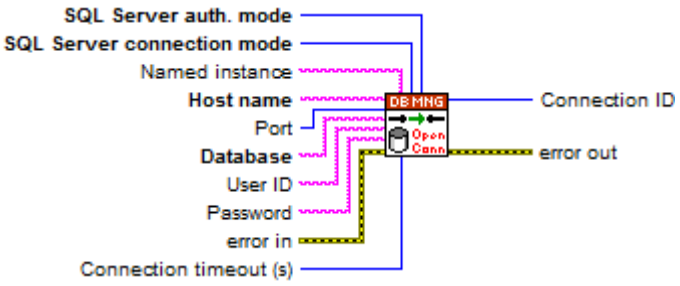

Figure 5: SQL server instance of DBM_Open_Connection.vi polymorphic VI

Parameters:

- **SQL Server auth. mode**: specifies the authentications system to use for connection. The following options are available:
  - Windows authentication: with this option authentication is performed using Windows user information (no further information required)
  - SQL authentication: with this option authentication is performed using SQL user informations. (**User ID** and **Password** parameters need to be specified).
- **SQL Server connection mode**: specifies how to connect to server. The following options are available:
  - Named instance: with this options connection is performed to "**Host name\Named instance**"
  - Port: with this option connection is performed to "**Host name**,**Port**"
- **Host Name**: specifies the address of SQL Server database engine where the database is hosted.
- **Port**: specifies the number of the TCP/IP port that database server use to receive connection requests. The default value is 1433.
- **Database**: specifies the database name to connect to
- **User ID**: specifies the user ID needed to access the database. You might not need to specify a userID.
- **Password**: specifies the password required to access the database for security purposes. You might not need to specify a password.
- **Connection timeout (s)**: specifies the number of seconds to wait for a connection to open, before cancelling the attempt and generating an error. The default value is 10.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection ID**: ID of DB connection
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to

error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.1.2 Close connection

When all actions are performed on database, the communication channel to it is no more required and can be closed through the DBM_Close_Connection.vi, releasing client and server machines resources.

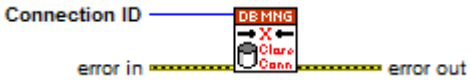

Figure 6: DBM_Close_Connection.VI

Parameters:

- **Connection ID**: ID of DB connection to close
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **error out**: contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

3.2 SQL Queries management

Green icons VIs provides tools for performing SQL queries on the connected database.

3.2.1 Execute SQL query

The DBM_Query_Execute.vi executes an SQL query and returns a query instance that you must eventually close with the FNC_DB_MANAGER_Close_Query VI.

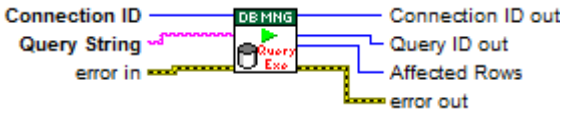

Figure 7: DBM_Execute.vi

Parameters:

- **Connection ID**: ID of DB connection
- **Query String**: specifies the SQL statement to execute.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection ID out**: ID of DB connection
- **Query ID out**: ID of the Recordset
- **Affected Rows**: returns the number of records in the Recordset
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.2.2 Fetch Data

DBM_Fetch_Data.vi retrieves data from specified Query ID. It allows to process a number of rows specified in the "Rows to fetch" control in order to reduce the memory usage in case of a large number of SELECT query affected rows.

Figura 1: DBM_Fetch_Data.vi

Parameters:

- **Query ID**: ID of the recordset to fetch data from.

- **Rows to fetch**: specifies the number of records to retrieve. The default is –1, indicating it will retrieve all records.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **results**: two-dimensional array containing data retrieved
- **Query ID out**: ID of the recordset to fetch data from.
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.2.3 Close Query

Once the SQL query is sent through the DBM_Query_Execute.vi and processed through the DBM_Fetch_Data.vi, the DBM_Close_Qeury.vi allows to close the query instance releasing the client machine used memory.

Figure 8: DBM_Close_Query.vi

Parameters:

- **Query ID**: ID of the recordset to close. Closing **Query ID**, the memory allocated for the query will be deallocated
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.3 Parametrized queries management

Violet icons VIs provides tools for performing parametrized queries on the connected database.

3.3.1 Create parametrized query

The DBM_Create_Parametrized_Query.vi creates a parameterized **SQL query** and returns a **Command ID out** that you must close eventually. The number of elements in the **parameters** array must match the number of parameters you specify in **SQL query**. For stored procedures, set **Stored procedure? (F)** to TRUE and **SQL query** to the name of the procedure you want to execute.

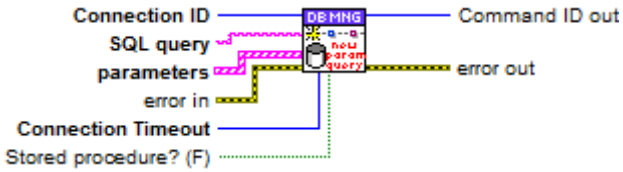

Figura 2: DBM_Create_Parametrized_Query.vi

Parameters:

- **Connection ID**: ID of DB connection to use to create parameterized query
- **SQL query**: Parameterized query to create
- **parameters**: Array of parameter informations describing characteristics of parameters declared in **SQL query** string
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection Timeout**: specifies the time, in seconds, to wait while attempting to execute a command. The default is 30. If the function waits <B>Connection Timeout</B> and the attempt to execute the command is not finished, the attempt is cancelled and an error is returned.
- **Stored procedure? (F)**: specifies how **SQL query** has to be evaluated. If FALSE (default), it has to be evaluated as a textual definition of a command. If TRUE, it has to be evaluated as a stored procedure name.
- **Command ID out**: ID of the created parametric command.
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.3.2 Set parameter value

The DBM_Cmd_Set_Parameter_Value.vi allows to set parameter value for the parametrized query the **Command ID** refers to.

Figura 3: DBM_Cmd_Set_Parameter_Value.vi

Parameters:

- **Command ID**: ID of the parametric query to set parameter to
- **Parameter index**: specifies the parameter to set.
- **Value**: specifies the parameter value to set.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **Command ID out**: ID of the parametric query to set parameter to
- **error out**: contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

3.3.3 Command execute

The DBM_Cmd_Execute.vi send the parametrized query specified in the **Command ID** to the connected database.

Figure 9: DBM_Cmd_Execute.vi

Parameters:

- **Command ID**: ID of the parametric command to execute
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Query ID out**: ID of the Recordset
- **Affected Rows**: returns the number of records in the Recordset
- **error out**: contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

3.4 Database properties

Blue icons VIs provides tools for retrieving connected database properties.

3.4.1 Get recordset properties

The DBM_Get_Recordset_Properties.vi allows to retrieve properties of the recordset specified by **Query ID.**

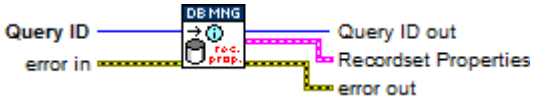

Figure 10: DBM_Get_Recordset_Properties

Parameters:

- **Query ID**: ID of the recordset to get properties from
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **Query ID out**: ID of the recordset to get properties from
- **Recordset Properties**: returns the following properties of the recordset.
  - column count: indicates the number of columns in the recordset
  - record index: specifies the ordinal position of the current record in the recordset (0-indexed)
  - record count: specifies the number of records in a recordset
  - bof?: assume the value of TRUE if the current record position is before the first record, otherwise FALSE
  - eof?: assume the value of TRUE if the current record position is after the last record, otherwise FALSE
  - cursor type: specifies the cursor type of the recordset
  - cache size: specifies the number of records that can be cached
  - state: describes if the recordset is open, closed, connecting, executing or retrieving data
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.4.2 Get column properties

The DBM_Get_Column_Properties.vi allows to retrieve column properties of the recordset specified by **Query ID.**

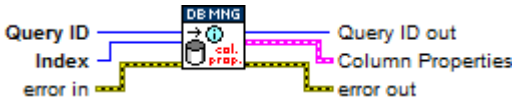

Figure 11: DBM_Get_Column_Properties.vi

Parameters:

- **Query ID**: ID of the recordset to retrieve properties from
- **Index**: Index of the column to get the properties from
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.
- **Query ID out**: ID of the recordset to retrieve properties from
- **Column Properties**: returns an array containing information about the columns in the recordset.
    - name: returns the name of the column.
    - database data type: returns the type of data in the column.
    - defined size: returns the defined size in bytes of the column.
    - actual size: returns the actual size of the column's value
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.4.3 List table

The DBM_List_Table.vi lists the tables of the database schema the **Connection ID** refers to.

Figure 12: DBM_List_Table.vi

Parameters:

- **Connection ID**: ID of DB connection
- **Table**: specifies the name of the table to search.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in error out.
- **Connection ID out**: ID of DB connection
- **Columns**: returns the array containing the list of columns of **Table**
- **Column Properties**: returns an array containing information about the columns in the database table.
    - name: returns the name of the column.
    - database data type: returns the type of data in the column.

    - defined size: returns the defined size in bytes of the column.
    - actual size: returns the actual size of the column's value.
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

3.5 Transactions management

The brown icon VI DBM_Manage_Transaction.vi provides tools for the database transactions management.

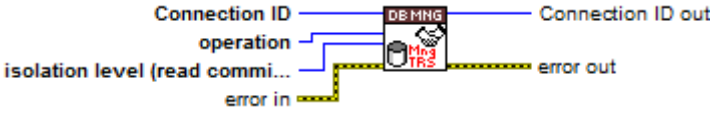

Figure 13: DBM_Manage_Transaction.vi

Parameters:

- **Connection ID**: ID of DB connection
- **operation**: specifies the operation you want to perform on the transaction.
  - begin: Starts a transaction.
  - commit: Commits the transaction to the database.
  - rollback: Discards the changes made to the database and restores the state of the database to the point when the transaction began.
- **isolation level (read committed)**: specifies the isolation level used for the transaction. You only need to set **isolation level (read committed)** if other transactions might be pending at this same time.
  - chaos: Transactions can overwrite each other.
  - read uncommitted: You cannot see uncommitted records from another transaction. Dirty reads, non-repeatable reads, and phantom reads are all possible.
  - read committed: This transaction cannot see changes made by other transactions until they are committed. Dirty reads are not possible, but non-repeatable reads and phantom reads are possible.
  - repeatable read: You cannot see any changes in records without requerying the database. Dirty reads and non-repeatable reads are not possible, but phantom reads are possible.
  - serializable: The transaction occurs in complete isolation. Dirty reads, non-repeatable reads, and phantom reads are not possible.
- **error in**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. This node runs normally only if no error occurred before
- **Connection ID out**: ID of DB connection
- **error out**: describes error conditions that occur before this node runs. The default is no error. If an error occurred before this node runs, the node passes the error in value to error out. If an error occurs while this node runs, it runs normally and merge its own error status with error in before to set it in error out.

4 Examples

SINT Technology DB Manager toolkit provides example VIs to show how it can be used to accomplish common tasks. Each example VI is configured for a MySQL DB connection, but they can be easily adapted to connect to the other database engine simply selecting the appropriate database engine polymorphic instance of the DBM_Open_Connection.vi and giving the required input parameters.

Both front panel and block diagram of the example VIs include descriptive labels in order to guide the user to run the example properly.

4.1 Query SELECT

This example shows how to use SINT Technology DB Manager VIs to execute a SELECT query and fetch data resulting from the query itself. The fields have to be filled with proper information of the database used to run the example correctly.

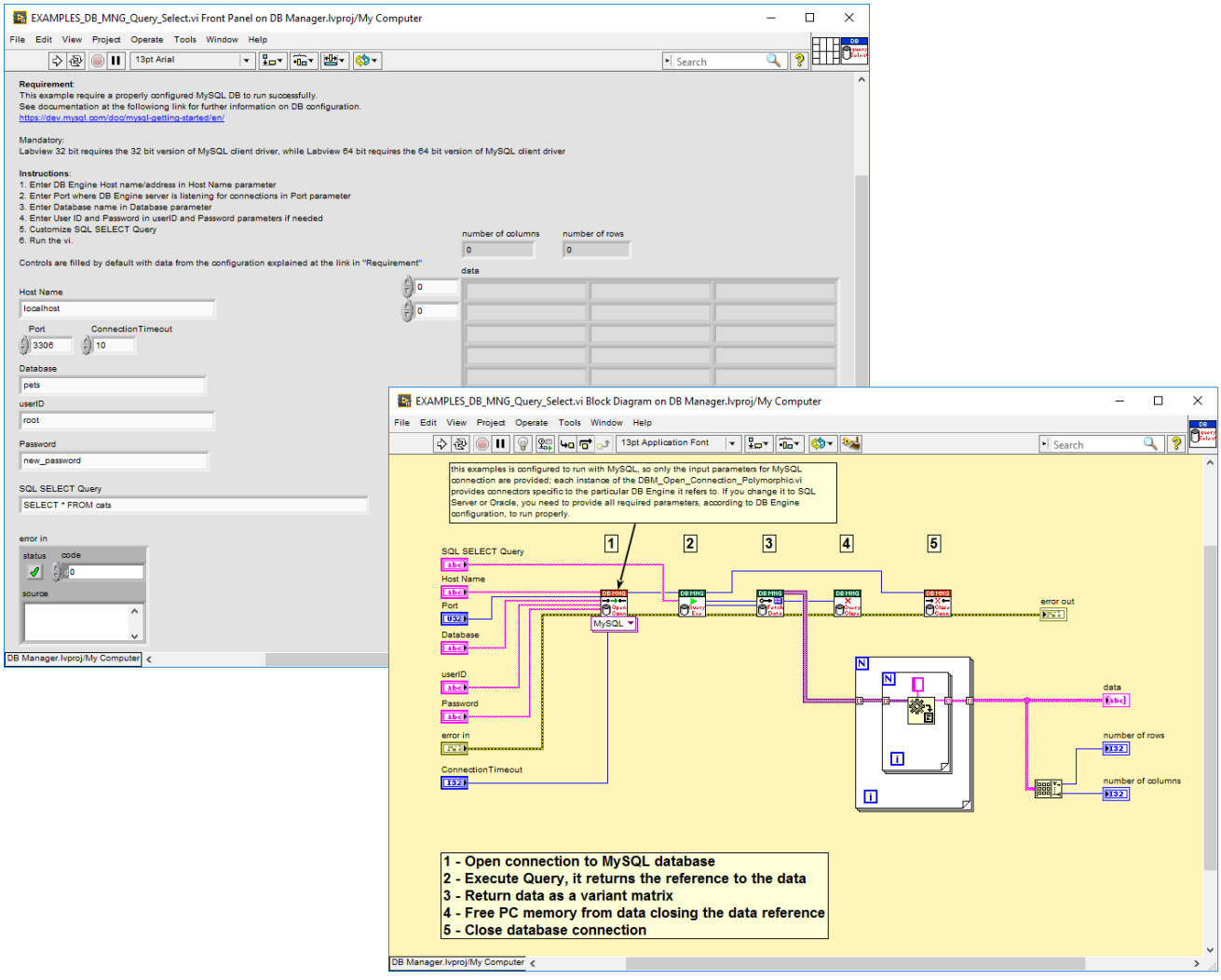

Figure 14: EXAMPLES_DB_MNG_Query_Select.vi

4.2 Query INSERT

This example shows how to use SINT DB Manager Vis to execute an INSERT query and shows also a simple use of transaction. The fields have to be filled with proper information of the database used to run the example correctly.

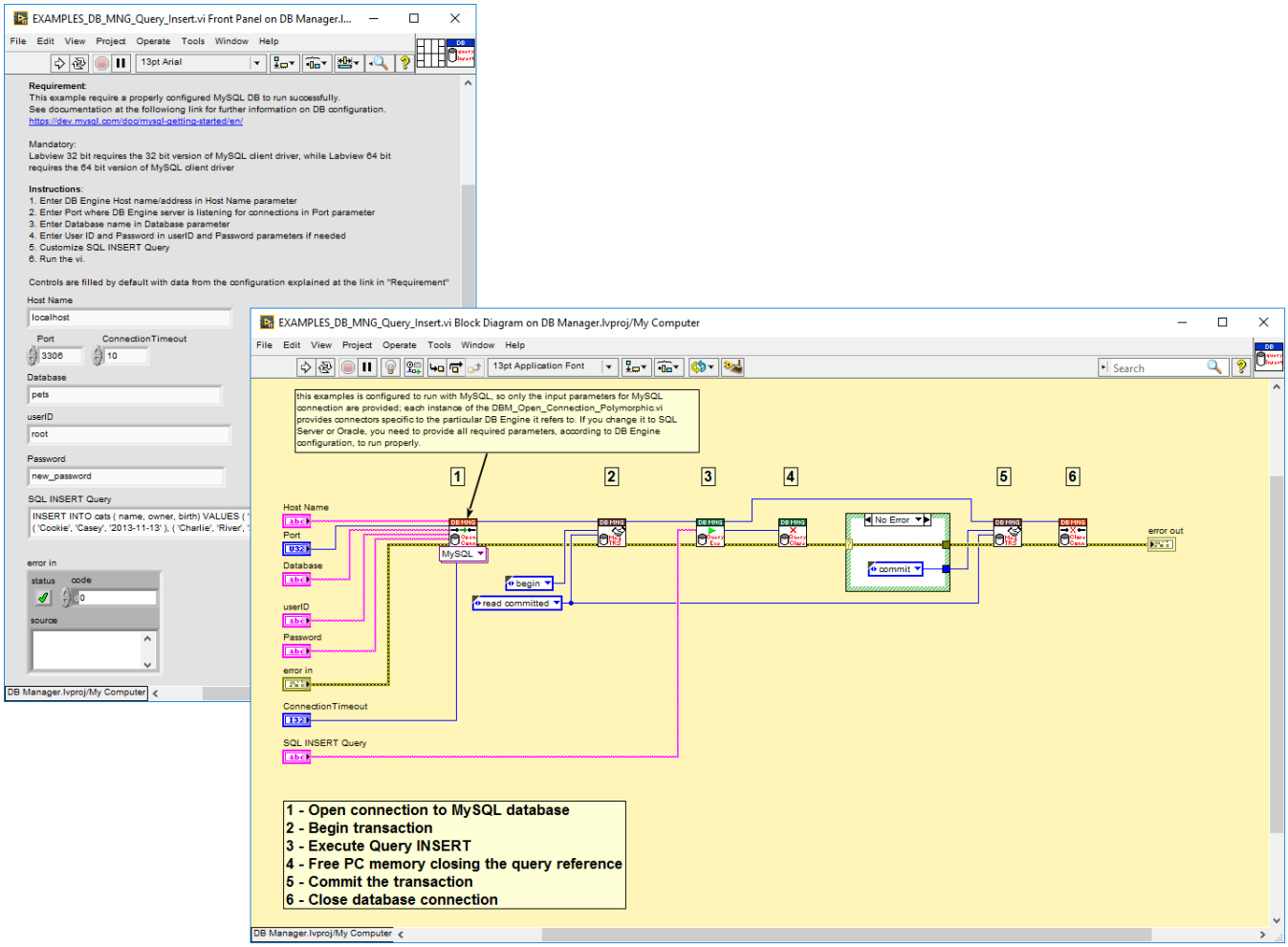

Figure 15: EXAMPLES_DB_MNG_Query_Insert.vi

4.3 Parametric query

This example shows how to use SINT DB Manager Vis to implement a simple parametric query and shows also a simple use of transaction. The fields have to be filled with proper information of the database used to run the example correctly.

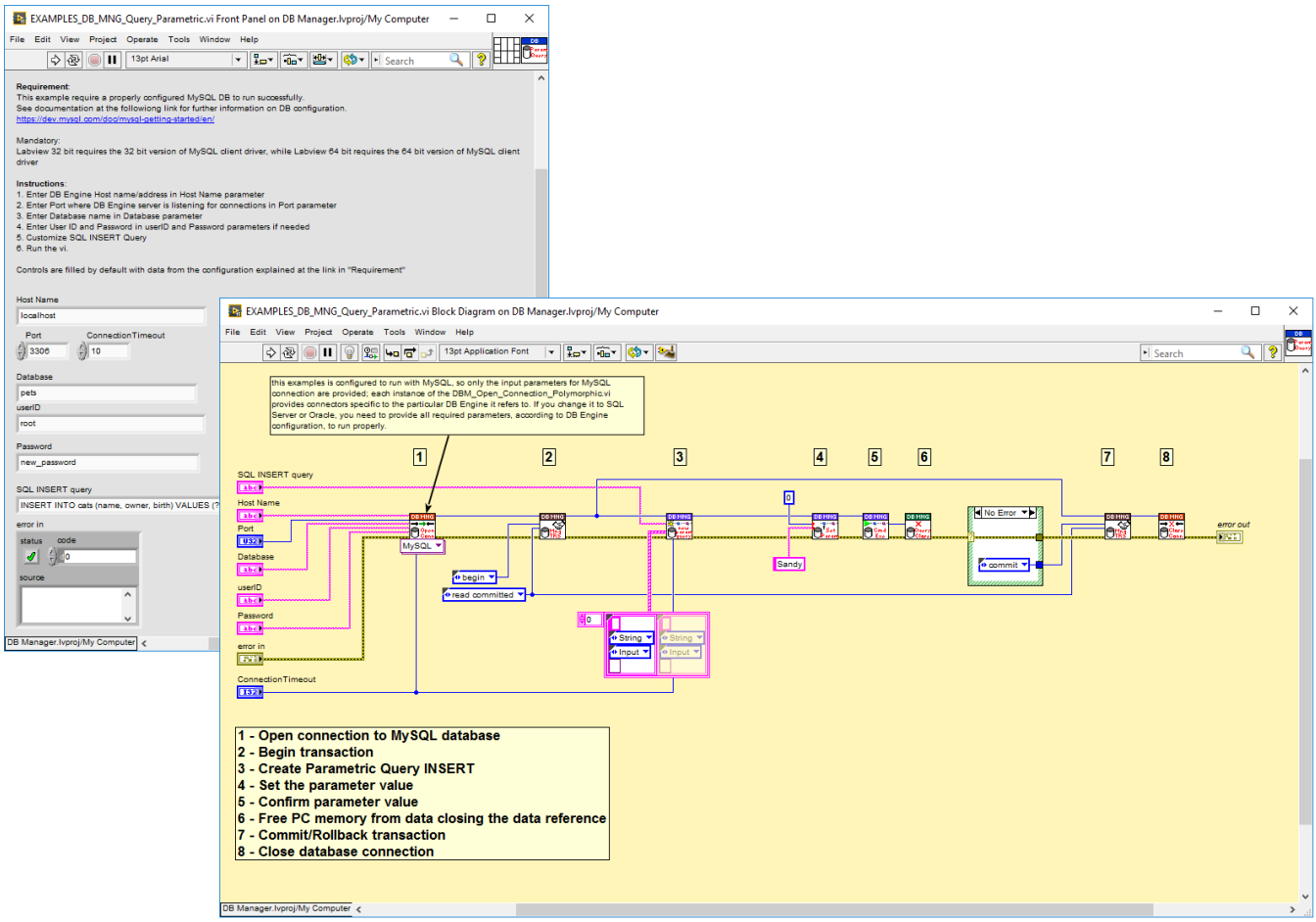

Figure 16: EXAMPLES_DB_MNG_Query_Parametric.vi

4.4 Retrieve database properties

This example shows how to use SINT DB Manager Vis to retrieve the list of tables with the list of their columns and related column properties. The fields have to be filled with proper information of the database used to run the example correctly

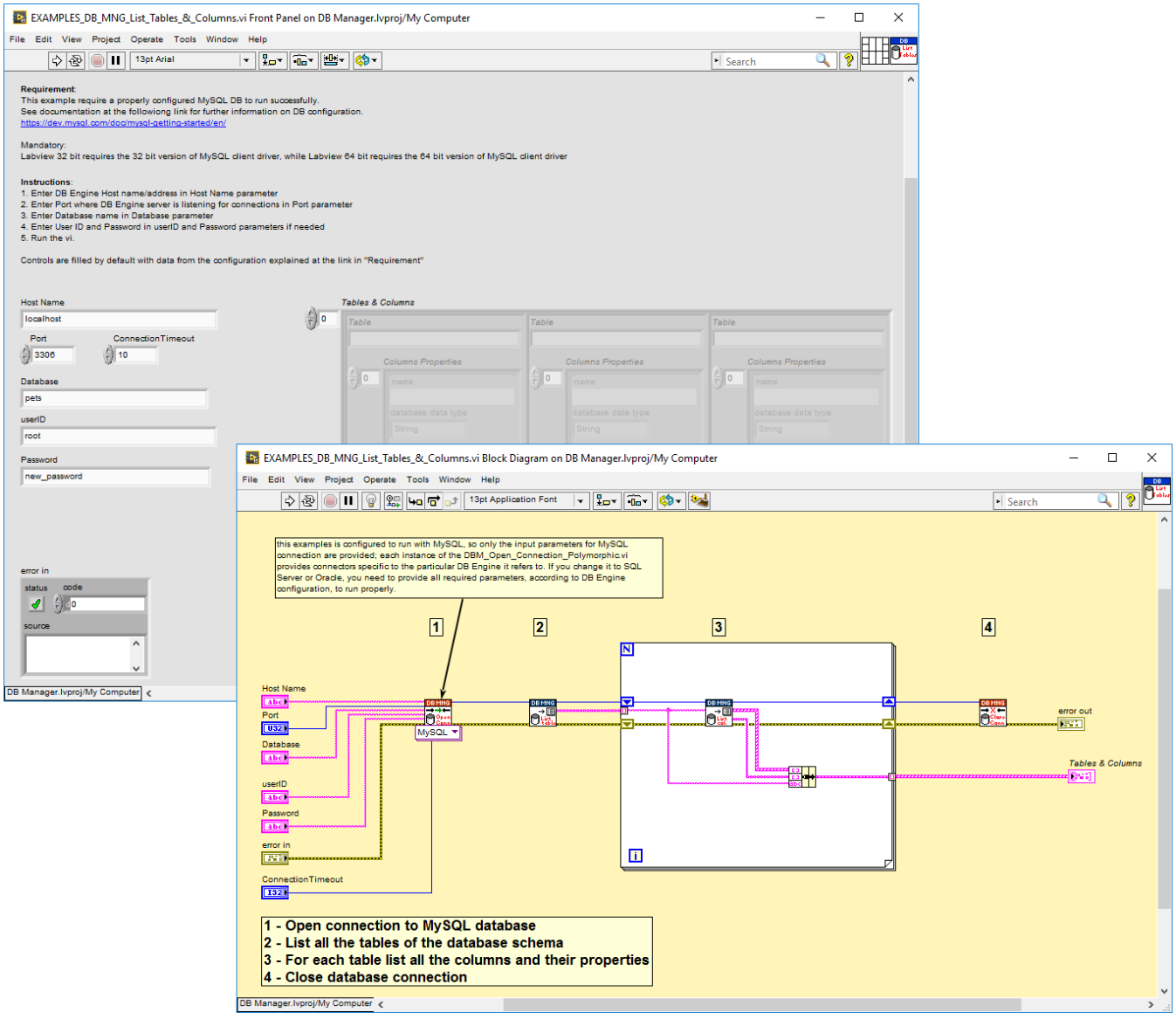

Figure 17: EXAMPLES_DB_MNG_List_Tables_&_Columns.vi

5 Evaluation and Licensing

Evaluation version is fully functional during the 30 days evaluation period. A message pop-up with the remaing evaluation period is displayed at the first call of a toolkit function when the project runs. If the evaluation period is expired every call of a toolkit function returns error.

Full licensed version is not limited to the 30 days evaluation period and doesn't display message pop-ups.

To purchase SINT DB Manager full licensed version you can follow the following steps:

1. Visit our software solutions page at https://www.sintechnology.com/hardware-software-solutions.html
2. Click on "request a quote" button and fill the request form.

6 Technical support

E-mail: support@sintechnology.com

Telephone: +39 055 8826302 / 8862965

Web site: www.sintechnology.com