FeSCADA – Database query

Introduction

A new utility called “Database query” was added to FeSCADA software. With this feature the user can setup and execute database queries with SQL statements. The output of different queries can be linked with program tags and the tags values sent to PLCs or other controllers on the plant floor level.

This chapter will cover the following topics:

1. Description – useful situations
2. Query trigger types
3. Basic SQL queries
4. Tags and SQL queries
5. Data conversion: database data types
6. Data conversion: tags data types
7. Application examples
8. Conclusions

1) Description – useful situations

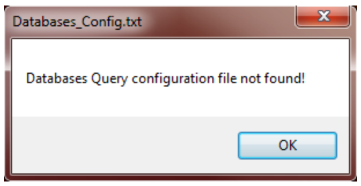
In a typical SCADA application the data is flowing upward from the plant floor to the control and administration network. The SCADA software is reading from PLCs and is recording in databases. The recorded data is then analyzed with other software for production output or for OEE performance.

But there are cases when the information has to flow from the control and administration network downward to the PLCs. For example if the recipes of a production machine are in a database, on a computer from the control network. Or if a production management software, or a remote operator, is planning a production run and has to start the machines on the plant floor at certain hours of the day, based on orders from the customers. Or if the PLC is working in an Automated Storage and Retrieval System (ASRS) and has to know the first empty location, from thousands, to control the servo-motors for positioning at that location, to store the product.

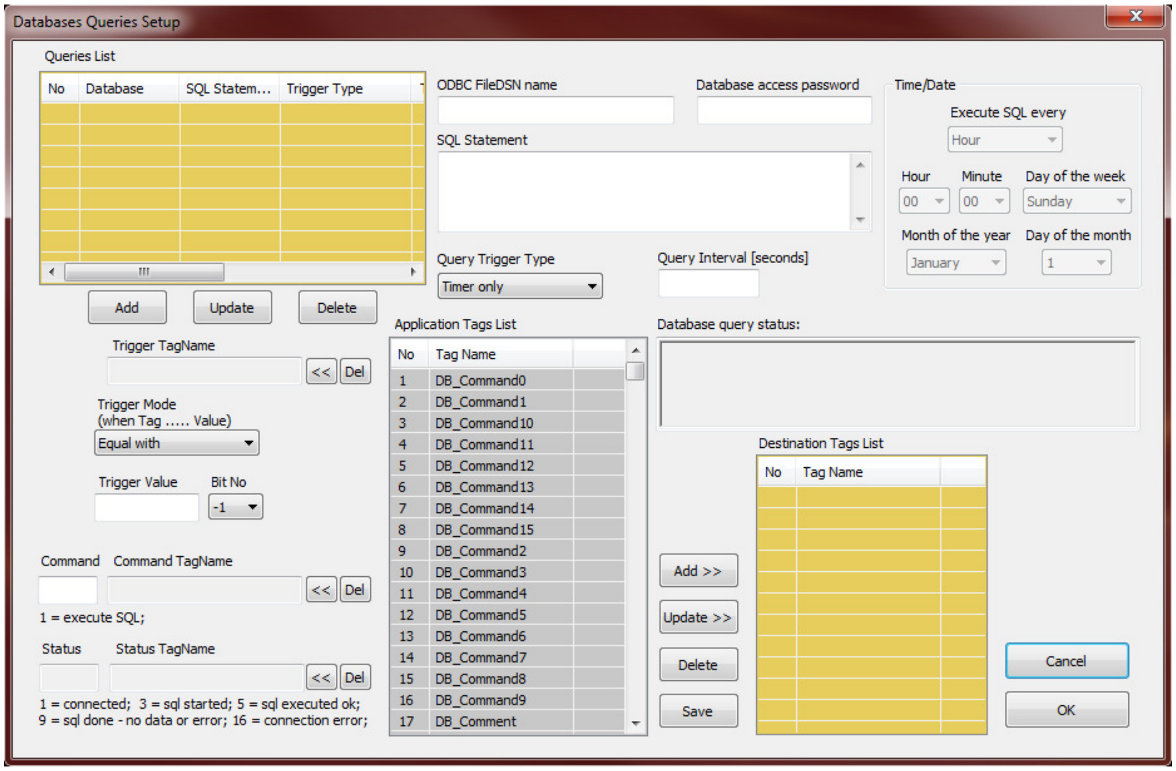
A second observation is that in the present times the Internet is used everywhere. It is in any work office and on every smart phone. The websites are using more and more databases to increase the interaction with the customers. The online store website backed by databases is probably the most prevalent business application on the Internet. The power of the Internet is that one can be anywhere in the world and have access to his own server or website. Also, it became fairly easy to setup and use an Apache server and to write PHP applications with which to read/write data from/to a MySQL database. Either for industrial applications or for home automation hobby projects.

In all the above examples the SCADA application is the software that has to read from a database and send the data to the PLCs (or Arduinos). The “Database Query” feature of FeSCADA was developed especially for this purpose.

To open the **Database Query Setup** dialog window, select from the menu **Utils->Database Query**. A notification message will inform if no data was previously configured. Press **OK** to open the dialog window. The list of the database **Queries** is on the top right corner of the window. The **Application Tags List** is in the middle. The **Destination Tags List** is in the bottom right corner.



Databases Query configuration file not found!



1 = execute SQL;

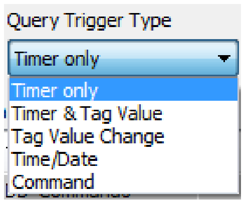
1 = connected; 3 = sql started; 5 = sql executed ok;
9 = sql done - no data or error; 16 = connection error;

No	Tag Name
1	DB_Command0
2	DB_Command1
3	DB_Command10
4	DB_Command11
5	DB_Command12
6	DB_Command13
7	DB_Command14
8	DB_Command15
9	DB_Command2
10	DB_Command3
11	DB_Command4
12	DB_Command5
13	DB_Command6
14	DB_Command7
15	DB_Command8
16	DB_Command9
17	DB_Comment

In the settings of the **Database Queries** dialog window there are many options common to the previously discussed data-logging, recipes, ramps, and timers utilities of FeSCADA. The **Add**, **Update** and **Delete** buttons are used to manage the queries list. The **Trigger TagName** is working the same as for data-logging utility. The **ODBC FileDSN name** is the name of the file with database connection settings. The **Database access password** text box is used when the database is requiring a password. The user will write the query in the **SQL Statement** text box. The **Query Interval [seconds]** and the **Time/Date** group of options have the same functionality as for data-logging, emails, and timers. Press **Update** if changes are made. Press **OK** to exit the **Database Query Setup** dialog saving the new changes. Restart FeSCADA for the new queries, or new changes, to become active.

2) Query trigger types

The **Query Trigger Type** has the options: *Timer only*, *Timer & Tag Value*, *Tag Value Change*, *Time/Date*, and *Command*. The first 4 options are working the same as for the data-logging utility. When the condition is true the query will be executed.



The *Command* option is working with the **Command TagName** and **Status TagName**. To setup the command and status tags select tags from **Application Tags List** and press the **<<** buttons. Press **Update** button if changes are made. If the **Comand Tag Name** is not set, the user can type the values 1 and 0 in **Command** text box to manually trigger the query execution.

Whenever the **Command TagName** has a value of 1 (one) the query is executed and the result is shown in the **Status TagName**.

Command	Description
0, <> 1	Nothing
1	**Start** the SQL query execution

Status	Description
-1, 16	Connection error
0	Not connected
1	Connection OK
3	SQL query started
5	SQL executed OK
9	SQL done – No data
17	SQL done – SQL Error

The *Command* tag has to toggle from 1 to 0 (or <> 1) and back to 1 to restart the SQL query.

Database query status text box will show the results of the database connection or of the last query execution.

Status	**Database query status** (examples)
0	MySQL2 - Not Connected
-1	MySQL3 - Connection error
1	ExcelDB - Connection OK
5	TextDB - Connection OK - Reading OK
9	TextDB - Connection OK - Reading OK - No Data
17	Test_SQL - [Microsoft][SQL Native Client][SQL Server]Invalid object name 'Peoples1'.
17	Test_SQL - [Microsoft][SQL Native Client][SQL Server]Invalid column name 'ID2'.

In a normal run the *Command* is set to 1 to execute the query and the *Status* tag is monitored. When the *Status* tag is equal with 5 (five) then the new data is valid and can be used for other purposes.

3) Basic SQL queries

Structured **Q**uery **L**anguage (**SQL**) is a specific programming language designed for managing data held in relational databases. It became a standard (ANSI – 1986, ISO – 1987) for databases access. A Windows ODBC driver for a particular database is able to interpret the SQL statements, regardless of the database name or version (Oracle, MySQL, Microsoft SQL, etc.).

With SQL statements one can create tables in a database, can insert new recordings, update older recordings, and many more. In the table below there are some examples of SQL statements.

SQL statements (examples)
CREATE TABLE Peoples (Name char(30), Title char(20), Age INT);
INSERT INTO Peoples (Name, Title, Age) VALUES ('John', 'Mr.', 32);
UPDATE Peoples SET Age=34, Title='Manager' WHERE Name LIKE 'John';
DROP TABLE Peoples;

With the instructions above the **Database Query** utility from FeSCADA can replicate the fuctionality of the **Data Logging** feature. It offers more flexibility but the setup is more cumbersome and the user has to pay attention to more details.

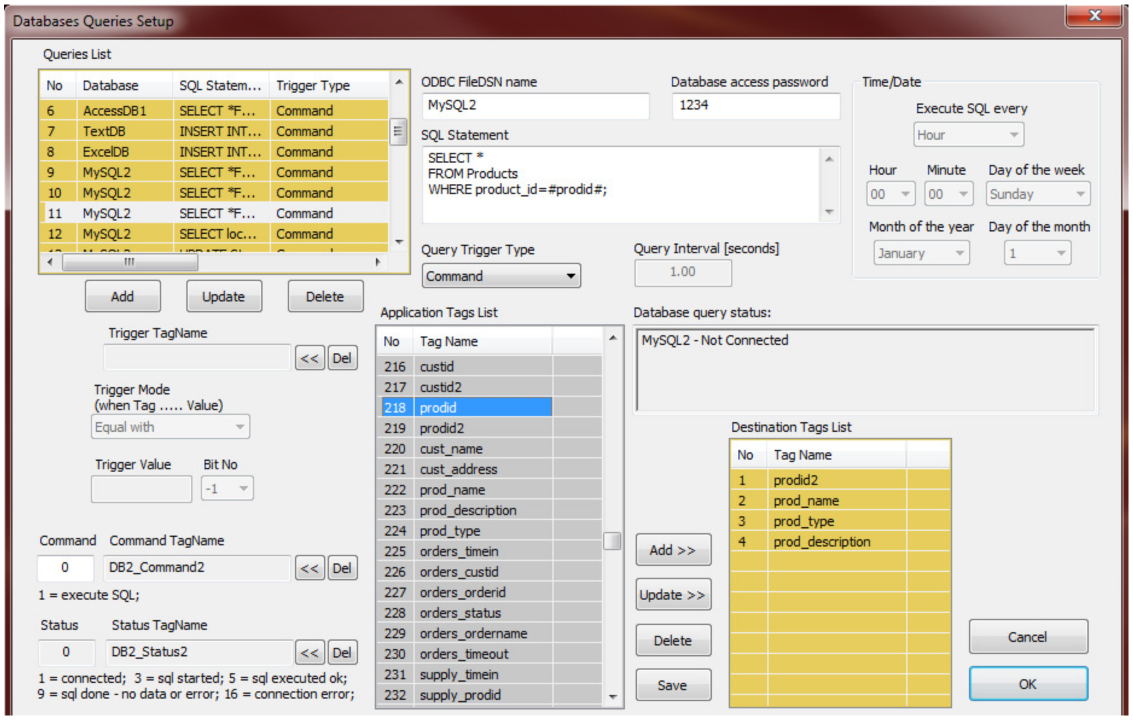
By far, the most important SQL statement is the SELECT… one. And this instruction is making possible the reading of data from a database. Examples of SELECT… statement are given below.

SELECT … SQL statements (examples)
SELECT * FROM Peoples WHERE 'Name' LIKE 'John';
SELECT * FROM Customers WHERE customer_id=10;
SELECT location FROM Storage WHERE product_id=0;
SELECT * FROM Supply WHERE qty1>qty2;
SELECT count(*) as Total_Emply_Locations FROM Storage WHERE product_id=0;
SELECT location FROM Storage WHERE product_id=7 GROUP BY last_updated_time LIMIT 1;

The data returned by SELECT… statement has to be interpreted and converted to be used with the tags defined in FeSCADA.

4) Tags and SQL queries

There are two ways in which the tags can be used with SQL queries. The first way is to include tags values into the SQL statements. The value of the tag will be replaced in the SQL statement at the moment of SQL execution. To insert a tag in the SQL statement place the cursor where you want the tag and double click the tag in the **Application Tags List**. The tag name will be inserted in the SQL statement in the format: *#TagName#.* In the picture below, "*prodid*" tag value will be used in the SQL statement.



Queries List

No	Database	SQL Statem…	Trigger Type
6	AccessDB1	SELECT *F…	Command
7	TextDB	INSERT INT…	Command
8	ExcelDB	INSERT INT…	Command
9	MySQL2	SELECT *F…	Command
10	MySQL2	SELECT *F…	Command
11	MySQL2	SELECT *F…	Command
12	MySQL2	SELECT loc…	Command

ODBC FileDSN name: MySQL2

Database access password: 1234

SQL Statement:

```
SELECT *
FROM Products
WHERE product_id=#prodid#;
```

Query Trigger Type: Command

Query Interval [seconds]: 1.00

Time/Date — Execute SQL every: Hour; Hour: 00; Minute: 00; Day of the week: Sunday; Month of the year: January; Day of the month: 1

Trigger TagName:

Trigger Mode (when Tag ….. Value): Equal with

Trigger Value: ; Bit No: -1

Command: 0; Command TagName: DB2_Command2

1 = execute SQL;

Status: 0; Status TagName: DB2_Status2

1 = connected; 3 = sql started; 5 = sql executed ok;
9 = sql done - no data or error; 16 = connection error;

Application Tags List

No	Tag Name
216	custid
217	custid2
218	prodid
219	prodid2
220	cust_name
221	cust_address
222	prod_name
223	prod_description
224	prod_type
225	orders_timein
226	orders_custid
227	orders_orderid
228	orders_status
229	orders_ordername
230	orders_timeout
231	supply_timein
232	supply_prodid

Database query status: MySQL2 - Not Connected

Destination Tags List

No	Tag Name
1	prodid2
2	prod_name
3	prod_type
4	prod_description

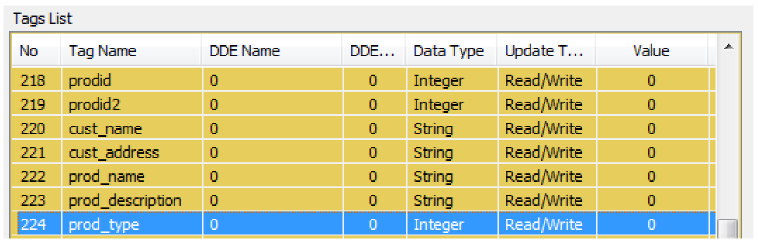
The second way to use tags with SQL queries is to retrieve the data returned by the SELECT… statement. For this the user has to select tags from **Application Tags List** and to insert them in **Destination Tags List** in the order in which the SELECT… statement will return the data.

The example below is the table "Products" in the database defined by the DSN file MySQL2.

prodid	name	type	description
1	Milk	1	2 percent
2	Butter	1	Not salted
3	Bread	2	Sour dough
4	Cheese	1	Cow
5	Eggs	3	Brown, big
6	Meat	4	Beef

The query: *SELECT * FROM Products;* will return 6 datasets and in each dataset we will have 4 variables: *prodid* – integer, *name* – string, *type* – integer, and *description* – string. In the **Destination Tags List** we have selected 4 tags to retrieve the data: *prodid2*, *prod_name*, *prod_type*, and *prod_description*.

FeSCADA will read the first dataset available and will convert the data to match the destination tags. After that it will ignore the next datasets. This is because the destination tags can take only one value at a time (for a reading). The execution of the above SELECT… statement will give the result:



Tags List

No	Tag Name	DDE Name	DDE…	Data Type	Update T…	Value
218	prodid	0	0	Integer	Read/Write	0
219	prodid2	0	0	Integer	Read/Write	0
220	cust_name	0	0	String	Read/Write	0
221	cust_address	0	0	String	Read/Write	0
222	prod_name	0	0	String	Read/Write	0
223	prod_description	0	0	String	Read/Write	0
224	prod_type	0	0	Integer	Read/Write	0

prodid2	*1*
prod_name	*'Milk'*
prod_type	*1*
prod_description	'2 percent'

To read a specific dataset the SELECT… statement has to be more precise. For example, if we want to read the dataset 4 the SELECT… statement has to be like this: *SELECT * FROM Products WHERE prodid=4;* or: *SELECT * FROM Products WHERE name LIKE 'Cheese';*

5) Data conversion: database data types

In FeSCADA we have only 3 data types: integer, float and string. The data types used in databases are more numerous and more complicated. In general it is possible to convert these databases data types to the basic integer, float and string types. Another observation is that the majority of the PLCs on the market have 16 and 32 bits registers.

After a query execution FeSCADA will check the data type of every column returned by the SELECT … statement. If the data type is numeric (integer or real) it will convert its value to three data types: an integer value, a float value and a string. If the data type is a string it will remain a string. If the data is of DATE, TIME or TIMESTAMP types it will be converted to a string. See the table below.

SQL data type	Data Type	FeSCADA data type conversion
SQL_CHAR SQL_VARCHAR	Char(xx) String	String (max 255 chars)
SQL_SMALLINT	Integer (32 bits)	Integer (Float, String)
SQL_INTEGER	Long (64 bits)	Integer (Float, String)
SQL_NUMERIC SQL_DECIMAL SQL_REAL	Float (32 bits)	Float (Integer, String)
SQL_FLOAT SQL_DOUBLE	Double (64 bits)	Float (Integer, String)
SQL_TYPE_DATE SQL_TYPE_TIME SQL_TYPE_TIMESTAMP	Date structure Time structure Timestamp structure	String (max 255 chars) “YYYY-MM-DD” “HH:MM:SS” “YYYY-MM-DD HH:MM:SS”

6) Data conversion: tags data types

In the next step the program will check the data type of the tag associated with every column. If the tag is an integer, the integer conversion from the column is copied into the tag’s value. If it is a float, or it is an integer that it is scaled, the real number of the column conversion is copied into the tag’s value. If the tag is a string, the string conversion of the column is copied into the tag.

The table below shows the data types conversion from database columns to FeSCADA tags.

Column data type	FeSCADA tag data type ->	Integer	Float	String
Char(xx) String	String (max 255 chars)	---	---	Yes
Integer (32 bits) Long (64 bits)	Integer	Yes	Yes	Yes
Float (32 bits) Double (64 bits)	Float	Yes	Yes	Yes
Date structure Time structure Timestamp structure	String (max 255 chars) “YYYY-MM-DD” “HH:MM:SS” “YYYY-MM-DD HH:MM:SS”	---	---	Yes

The number of tags in the **Destination Tags List** can be bigger or smaller than the number of columns returned by the SELECT … statement. The first returned column is assigned to the first tag, the second column to the second tag, etc.

7) Application examples

There are many possible applications that can be done using the **Database Query** utility of FeSCADA software. Recipes selection, turning on and off PLC outputs from a Web page, start/stop machines from a production management software, automation of a storage system, etc.

Another family of applications is for home automation projects. The user can start/stop the heating system, lights and sprinklers from a web page available on his own smart phone. The only concern is for the authentication and security of the website access.

8) Conclusions

With FeSCADA software it is possible to read data from a database and to send the result downward to the plant floor level, to the PLCs and other industrial controllers. The SELECT… query is used in an SQL statement. Tags can be used in building the SQL statement. The data returned by SELECT… queries is converted and copied to selected destination tags.