SCHNEIDER ELECTRIC

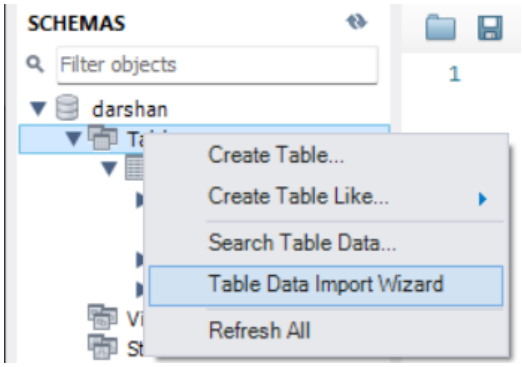
NODE RED FLOW - MySQL

- DARSHAN S

OBJECTIVE :

To create a Node-RED flow consists of several interconnected nodes designed to interact with a Modbus server, process the data, publish it to an MQTT broker, and store the data in a MySQL database and also make changes in the existing table with SQL Queries.

IMPORT EXCEL INTO MySQL TABLE :



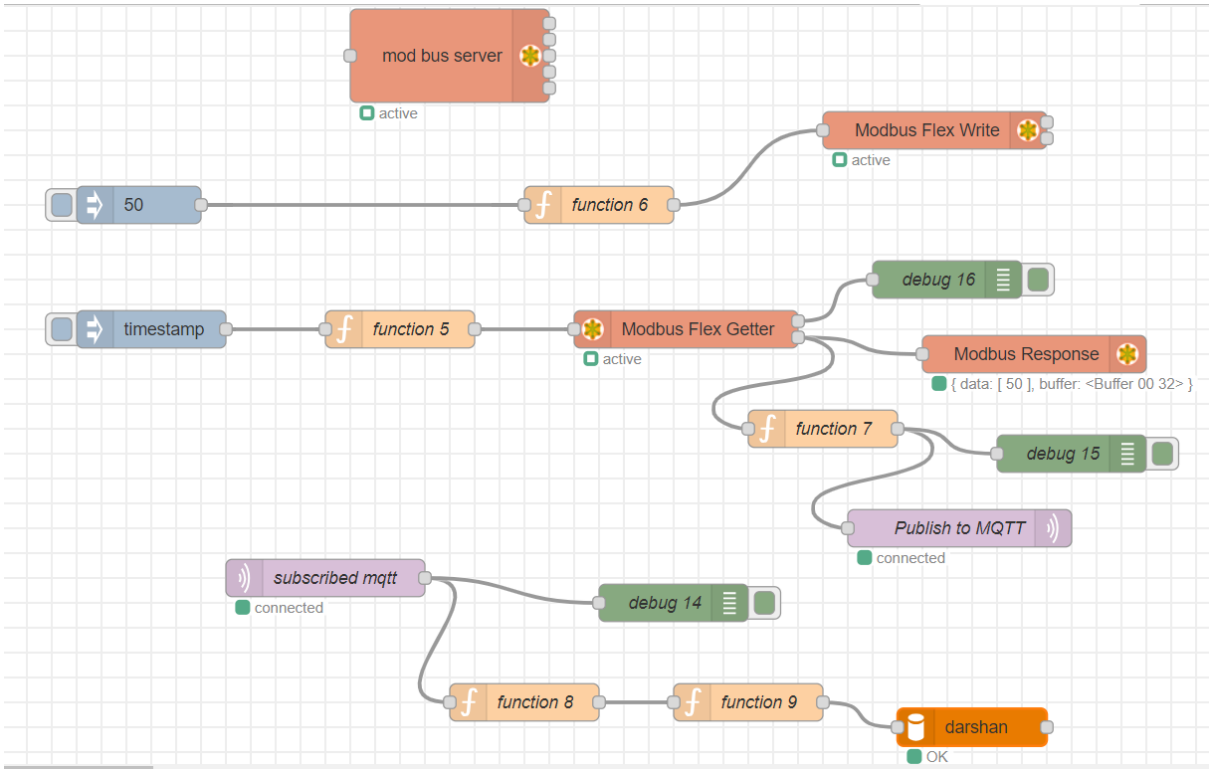
Select the Table Data Import Wizard option from your required database in MySQL Workbench.

Browse the CSV(MS – DOS) excel file which you need to import. Then continue and Finish.

EXAMPLE DATA (100 rows) :

```
mysql> select*from employee;
+------+------------+------------+--------+---------------+------+------------+------+
| Sno  | First Name | Last Name  | Gender | Country       | Age  | Date       | Id   |
+------+------------+------------+--------+---------------+------+------------+------+
|    1 | Dulce      | Abril      | Female | United States |   32 | 15/10/2017 | 1562 |
|    2 | Mara       | Hashimoto  | Female | Great Britain |   25 | 16/08/2016 | 1582 |
|    3 | Philip     | Gent       | Male   | France        |   36 | 21/05/2015 | 2587 |
|    4 | Kathleen   | Hanner     | Female | United States |   25 | 15/10/2017 | 3549 |
|    5 | Nereida    | Magwood    | Female | United States |   58 | 16/08/2016 | 2468 |
|    6 | Gaston     | Brumm      | Male   | United States |   24 | 21/05/2015 | 2554 |
|    7 | Etta       | Hurn       | Female | Great Britain |   56 | 15/10/2017 | 3598 |
|    8 | Earlean    | Melgar     | Female | United States |   27 | 16/08/2016 | 2456 |
|    9 | Vincenza   | Weiland    | Female | United States |   40 | 21/05/2015 | 6548 |
|   10 | Fallon     | Winward    | Female | Great Britain |   28 | 16/08/2016 | 5486 |
|   11 | Arcelia    | Bouska     | Female | Great Britain |   39 | 21/05/2015 | 1258 |
|   12 | Franklyn   | Unknow     | Male   | France        |   38 | 15/10/2017 | 2579 |
|   13 | Sherron    | Ascencio   | Female | Great Britain |   32 | 16/08/2016 | 3256 |
```

FLOW:



Modbus Setup

Modbus Server

(`modbus-server`)

- **Purpose**: Simulates a Modbus server that you can read from and write to.
- **Configuration**: It runs on `localhost` (127.0.0.1) at port `10502`.

Modbus Write Operation

Triggering the Write Operation

1. **Inject Node for Triggering Write:**
   - **Purpose**: Initiates the write operation.
   - **Configuration**: Sends a payload of `2` to trigger the write operation.
2. **Function Node (`function 6`):**
   - **Purpose**: Prepares the Modbus write request message.
   - **Code:**

```
msg.payload = {
    value: msg.payload,
    'fc': 6,
    'unitid': 1,
    'address': 0,
    'quantity': 1,
```

```
};
return msg;
```

Modbus Read Operation Explained

Triggering the Read Operation

1. **Inject Node for Triggering Read:**
    - **Purpose**: Initiates the read operation at specified intervals or events.
    - **Configuration**: Sends a Modbus read request to fetch data.
2. **Function Node (`function 5`):**
    - **Purpose**: Constructs the Modbus read request message.
    - **Function**

```
msg.payload = {
    'fc': 3,
    'unitid': 1,
    'address': 0,
    'quantity': 1,
};
return msg;
```

NOTE : *Modbus, primarily designed for industrial communication, typically handles numerical data. There is no direct mapping of string variables to Modbus space. So if you need to read those data, then your program can break up the strings into individual bytes and store them inside the DM space so that they become accessible via Modbus command.*

FUNCTION 7 :

Function node `function 7` in the Node-RED flow plays a crucial role in processing the response data received from the Modbus read operation before it is further utilized or published.

The primary purpose of `function 7` is to handle and manipulate the payload received from the Modbus read operation (`modbus-flex-getter` node). It ensures that the data is formatted or prepared appropriately for subsequent actions, such as publishing via MQTT (`mqtt out` node) or performing additional processing.

Here basically we have to change the array value got from the Flex-Getter into a simple integer to input in the query of MySQL.

Code :

```
// Check if the data array exists and has at least one element
if (msg.payload.data && msg.payload.data.length > 0) {
    // Extract the first value from the data array
    msg.payload = msg.payload.data[0];
} else {
    // Handle the case where data is not present or empty
    msg.payload = null;
}

return msg;
```

Eclipse Mosquitto Broker in MQTT Communication

Eclipse Mosquitto is an open-source MQTT broker that facilitates the communication between different MQTT clients. It handles the publish/subscribe messaging pattern used in the MQTT protocol, enabling devices and applications to communicate with each other effectively.

Role in This Flow

In this Node-RED flow, the Eclipse Mosquitto broker plays a crucial role in managing the MQTT communication between different nodes. It serves as the central point where messages are published and from which they are subscribed.

Key Points

1. **Broker Configuration**:
    - **Broker**: `localhost`
    - **Port**: `1883`
    - **Client ID**: Not specified (auto-generated by the client)
    - **Protocol Version**: MQTT version 3.1.1
2. **Publishing Data**:
    - **Node**: `Publish to MQTT`
    - **Topic**: `modbus/data`
    - **Description**: This node publishes Modbus data to the `modbus/data` topic on the Mosquitto broker. Whenever the `Publish to MQTT` node receives a message, it sends that message to the Mosquitto broker, which then makes it available to all clients subscribed to that topic.
3. **Subscribing to Data**:
    - **Node**: `subscribed mqtt`
    - **Topic**: `modbus/data`
    - **Description**: This node subscribes to the `modbus/data` topic on the Mosquitto broker. Whenever a message is published to this topic, the broker forwards the message to this node. The node then processes the message through its connected nodes.

4. **Message Flow**:
    - **Publish**: After reading data from the Modbus server, the `function 7` node processes the payload and forwards it to the `Publish to MQTT` node. This node then publishes the data to the `modbus/data` topic on the Mosquitto broker.
    - **Subscribe**: The `subscribed mqtt` node subscribes to the same `modbus/data` topic. When the Mosquitto broker receives a message on this topic, it forwards the message to the `subscribed mqtt` node. The message is then processed by `function 8` and other connected nodes.

FUNCTION 8 :

Purpose

The purpose of this function is to take the payload received from the MQTT topic and use it to construct an SQL query. This query is then sent to the MySQL node to update a specific record in a database.

```
// Extract the value from the MQTT payload
var value = msg.payload;

// Construct the SQL query
var sql = `DELETE FROM employee where age >= ${value}`;

// Set up the message payload for the MySQL node
msg.topic = sql;
return msg;
```

- Using the extracted value, the function constructs an SQL `DELETE` query.
- The query deletes the employee data whose ages are above 50.

This function node automates the process of updating a MySQL database with values received via MQTT. For example, if a sensor value is published to the `modbus/data` topic, this function can update the database with the new sensor reading, enabling real-time data logging and monitoring.

FUNCTION 9: Validating and Filtering Payload

Node Details

- **Node**: `function 9`
- **Function Name**: Validate Payload Type

Purpose

The purpose of this function is to ensure that only messages with numeric payloads are forwarded. If the payload is not a number, the message is discarded.

```
if (typeof msg.payload === 'number') {
    return msg; // Only send the message if the payload is a number
} else {
    return null; // Discard the message if the payload is not a number
}
```

MySQL Node: Executing SQL Queries

Node Details

- **Node**: `mysql`
- **Name**: MySQL Database Node
- **Configuration**:
  - **Host**: `127.0.0.1`
  - **Port**: `3306`
  - **Database**: `my_database`
  - **Charset**: `UTF8`

Purpose

The MySQL node is responsible for executing SQL queries against a MySQL database. In this flow, it updates a specific record in the database with the value received from the MQTT message.

SUMMARY :

Here's a brief summary of the Node-RED flow:

1. **Modbus Server Node**: Simulates a Modbus server.
2. **Inject Node**: Triggers Modbus read operations.
3. **Function Nodes**: Prepare Modbus requests and handle MQTT and Modbus responses.
4. **Modbus Flex Getter Node**: Reads data from the Modbus server.
5. **MQTT Nodes**: Publish and subscribe to MQTT topics for data transmission.
6. **MySQL Node**: Updates a MySQL database with data received from MQTT.
7. **Debug Nodes**: Display debug messages for monitoring and troubleshooting.

This flow integrates Modbus communication, MQTT messaging, and MySQL database operations within Node-RED for industrial automation and data management tasks.