Working with WKT Strings in MarushkaDesign

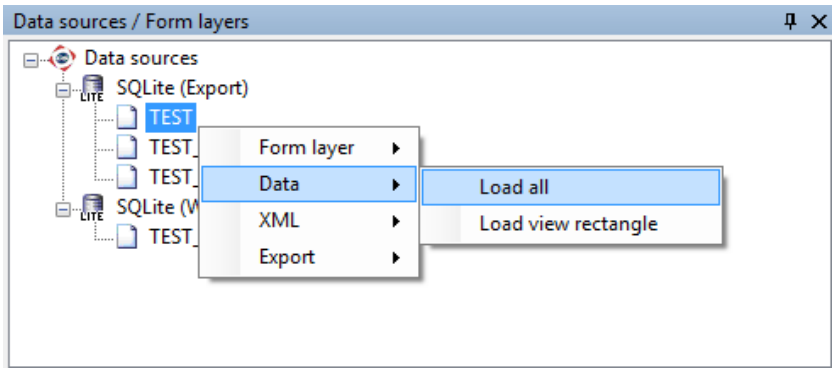
CONTENTS

1 Aim of the Example

In this example we will demonstrate how to work with WKT strings in MarushkaDesign. This example was created in version 4.0.1.0, so it does not have to be compatible with older versions..

2 Working with Example

- Unzip the **WorkingWithWKTString_CZ.zip** into **c:\MarushkaExamples** folder. The target folder must be respected due to interconnection of paths with the project. In the case of placing the files in the different folder, it would not be possible to work with an example.
- Open the **WorkingWithWKTString_CZ.xml** in MarushkaDesign environment.
- Select form layer TEST in SQLite (Export) data store, in context menu choose Data – Load all:



- In map window choose „Fit all“:

- Launch the local web server:

3 A Brief Description of the Example in MarushkaDesign

MarushkaDesign is designed so that the default files (which are the most frequently used), are WKB (Well Known Binary). You can read more about WKB geometry here.

MarushkaDesign also supports WKT (Well Known Text) format, a text alternative to WKB. WKT is a **text** language for representing objects of vector geometry. You can read more about WKT here.

At the start user receives data, e.g. in Excel. The data contain information on GPS coordinates and any additional information that user want to display in these coordinates.

For effective publication of spatial data, it needs to be spatially indexed (files with an extension FRX for WKB file, virtual table SPATKEY in SQL data stores.

If data are not spatially indexed, in MarushkaDesign environment it is possible to “load data” and subsequently export them to another data store.

In the following paragraphs, we will demonstrate how to load data using WKT format from a simple table named TEST with columns `ID|CITY|LATITUDE|LONGTITUDE`, where rows represent individual elements of point geometry. Subsequently we will convert this data to a new geometric table with a spatial index and in WKB format. With such a modified data we are now able to efficiently publish the data. During this conversion, we will also perform transformation of elements from WGS84 to Mercator.

Connect the data store to MarushkaDesign and set transformation of coordinate systems from WGS84 to Mercator. Rename this data store to SQLite (Export) for clarity.

Row in the table represents a point at coordinates LATITUDE, LONGITUDE with attribute *city*. We will represent it as a point (text) element of the given coordinates. In WKT format we will enter the geometry as a string `POINT (LATITUDE LONGITUDE)`.

To retrieve the data in MarushkaDesign you must first add into table TEST a column GEOM that defines the geometry (geometry in WKT format) for each record in the table TEST. You can do it:

a) In cases when it is not desirable to modify the database, in the from layer property *Name* you will not define the physical layer (table) name, but the phrase. This phrase must contain the system columns that you want to use and all other columns you want to use (in the properties *DBColumnsToClient, DBWhereClause* etc.). In particular, the phrase:

   `(SELECT ID, LAT, LON, CITY, 'POINT ('||lon||' '||lat||')' GEOM FROM TEST)`, the GEOM column defines geometry in WKT format.

   In example it is form layer TEST.

b) In the case that we can modify the data in the data store, create a new table with a column GEOM, let’s call it TEST_GEOM. The table is then loaded as a common geometric table. Define property *Name* for the form layer as TEST_GEOM, which must correspond with a *Name* of the newly created table. The system will recognize that the column is of the type TEXT or VARCHAR and will try to convert its content from WKT format to WKB format.

   ```
   CREATE TABLE TEST_GEOM AS SELECT ID, LAT, LON, 'POINT ('||lon||'
   '||lat||')' GEOM FROM TEST
   ```

   In example it is form layer TEST_GEOM.

   Table TEST_GEOM contains in column GEOM records type `POINT(LATITUDE LONGITUDE)`. However, the column can include a more complex geometry, such as LINESTRING, POLYGON, GEOMETRYCOLLECTION, etc.

c) In the case that you do not want to create new table, you can use command CREATE VIEW, which will have the same effect as the b). In this example, it is a form layer TEST_VIEW.

```
CREATE VIEW TEST_VIEW AS SELECT [id],[city],[lat],[lon],'POINT
('||lon||' '||lat||')' GEOM FROM TEST
```

In example it is form layer TEST_VIEW.

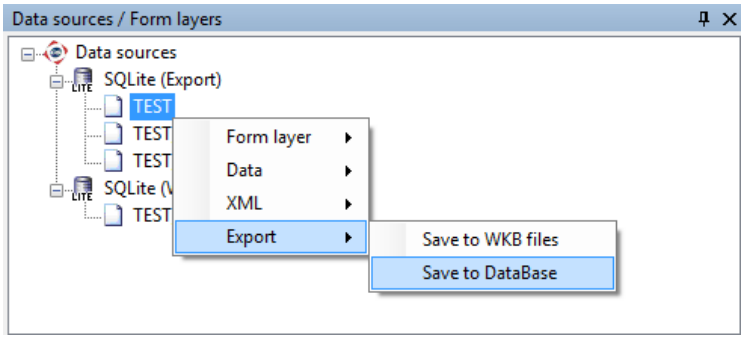
Currently the structures are prepared in the data store. Using function Load all (*Select form layer TEST or TEST_GEOM -> right mouse button click -> Data -> Load All ->* In the map window *All*) – this is done just like in the picture at the beginning of this example – you are able to retrieve the data, therefore you can proceed to the actual conversion.

Create a target geometric table TEST_WKB, it will contain, in addition to the mandatory columns column *City*, it can also contain columns *lat* and *lon*. SQL (founding script) is in the file TEST_WKB.sql.

The conversion of formats between WKT and WKB will the system do by itself. Perform conversion between coordinate systems so that the data store from which are read the data was set to WGS84 -> Mercator. Then connect the same database once more (name it SQLite WKB) and set the coordinate system conversion to Mercator -> Mercator. This conversion between two data stores was performed, because it is not possible to perform Export of data within a data store, which has a different source and target coordinate systems.

Create a physical layer in SQLite (WKB) with name TEST_WKB.

Now perform a right mouse click on form layer TEST, TEST_GEOM or TEST_VIEW and choose *Export -> Save to database* as shown below. After this operation, the dialog will pop up, asking if you want to keep the original ID, here choose no.



Now create a new form layer called TEST_WKB, define its property *Name* as “TEST_WKB”.

Finally select form layer TEST_WKB (*Right mouse button click -> Data -> Load all -> In map window choose All)* as in the picture at the beginning of this example. Then you need to add this layer to the publish layers and at this moment it is prepared for publication.