**Description of the programming logic to import networks data from the spreadsheet to the WaMDaM database**

Adel Abdallah, Jan 12, 2017

**Next task:**

1. Update the following few changes on the previous package
   1. Update importing data to the tables Sources and Methods to read both of them from one spreadsheet called “1.2\_Sources&Methods”
   2. Update importing data to the tables Datasets and ObjectTypes to read both of them from one spreadsheet called “2.1\_Datasets&ObjectTypes”
   3. Update the table called “2.2\_Attributes” to read data from a different starting point. Point 2.4 below talks about the Global Attributes Table
   4. Add an empty “ReadME” text file inside each folder in the package
   5. Create a new folder called “Data” that includes the sqlite database file and the used spreadsheet input files. I want the browser to point to that folder when looking for input files.
   6. Update the package name to Jan12
2. Load data for five tables and connect them: Instances, Connections, Mapping, DataValuesMapper, and ScenarioMapping.
   1. Implement the rules defined in this document for Datasets, Object Types, Attributes, and Scenarios
   2. Load data into the “Instances” and “Connections” tables
   3. Connect the network data into the “Mapping” and the ScenarioMapping tables. This includes many steps that are described in the document on GitHub named: Task\_Networks\_Jan12.dox.
   4. Add an import function to the Global Attributes table. This table is almost identical to the Attributes table in excel but it used the Dataset reference instead of the ObjetcType reference as described later in the document. Data from the Global attributes will be populated in the same table in the databases but there is some different approach on how to connect it with Instances as described later in the document.

**Task description**

Importing network’s spreadsheet data into WaMDaM SQLite is a complex task in the Wizard which involve looking up and mapping foreign keys. This document describes the logic of importing the data and explains what the business rules are. The center of this data loading step that maps tables together in WaMDaM is a table called “Mapping”. This table is invisible to users. The Mapping table is populated when users load networks, scenarios, then node and link “instances” and relate them with Attributes, a source, and a method. In this step, the user can define a new network which comprise node and links instances but without populating data values for the attributes of instances. This leaves the red tables in the ER diagram empty for now.

**Software business rules**

1. Implement a software business rule for this action: after the user imports/loads a “dataset” entry, the Wizard must automatically populate one ObjectType entry for the dataset with these specific dummy values in the table below

|  |  |
| --- | --- |
| **ObjectType Table fields** | **Value** |
| [ObjectType](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_ObjectType.html) | ScenarioDummyObjectType |
| [ObjectCode](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_ObjectCode.html) | ScenarioDummyObjectType |
| [ObjectTopologyCV](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_ObjectTopologyCV.html) | network |
| MapColor | null |
| MapSymbol | null |
| [Description](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_Description.html) | The purpose of this ObjectType is first to connect and help query a scenario and network for a dataset. Second, this ObjectType allows users to create attributes that are global to the network or scenario. Like define a global attribute that applies to all the nodes and links of the network. |
| [ObjectTypeCVID](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_ObjectTypeCVID.html) | null |
| [ObjectCategoryID](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_ObjectCategoryID.html) | null |
| [DatasetID](http://amabdallah.github.io/test2/columns/DatasetStructure_ObjectTypes_DatasetID.html) | [ ] look up the new datasetID that is just created |

Then after the ScenarioDummyObjectType is created, the Wizard creates one attribute called ScenarioDummyObjectType with the value in the table below for the dummy ObjectType that is just created.

|  |  |
| --- | --- |
| **Attributes Table fields** | **Value** |
| AttributeName | ScenarioDummyAttribute |
| ObjectTypeID | [ ] look up the ObjectTypeID for the ScenarioDummyObjectType |
| AttributeCode | ScenarioDummyAttribute |
| UnitCVID | Dimensionless |
| AttributeTypeCVID | dummy |
| AttributeNameCVID | null |
| AttributeCategoryID | null |
| ModelInputOrOutput | null |
| AttributeDescription | The purpose of this Attribute is to connect and help query a scenario and network for a dataset. |

Then after the ScenarioDummyAttribute is created, the Wizard will be ready to import ObjecTypes and Attributes. The Wizard will create another dummy Attribute for each ObjectType created in the database. The dummy attribute is called ObjectTypeDummyAttribute as populated with the values in the table below. Once all the ObjectTypes and their Attributes are loaded into the database, the Wizard and users can load the network and its scenarios (see point 2 below).

|  |  |
| --- | --- |
| **Attributes Table fields** | **Value** |
| AttributeName | ObjectTypeDummyAttribute |
| ObjectTypeID | [ ] Use the ObjectTypeID that the for ObjectTypeDummyAttribute |
| AttributeCode | ObjectTypeDummyAttribute |
| UnitCVID | Dimensionless |
| AttributeTypeCVID | dummy |
| AttributeNameCVID | null |
| AttributeCategoryID | null |
| ModelInputOrOutput | null |
| AttributeDescription | The purpose of this Attribute is to connect and help query all the instances that belong to one ObjectType. |

1. After the Wizard loads a new MasterNetwork and at least one scenario, it must populate these four tables in order, with these specific dummy values.

Create a dummy instance table(s) with these values for each scenario

|  |  |
| --- | --- |
| **Instances Table fields** | **Value** |
| InstanceName | [use the same scenario name +”DummyInstance”] |
| InstanceCode | [use the same scenario name +”DummyInstance”] |
| Longitude\_x | null |
| Latitude\_y | null |
| Description | Dummy instance to help connect scenarios and networks with their dataset through the ScenarioDummyAttribute and ScenarioDummyObjectType |
| InstanceNameCVID | null |

Populate the DataValuesMapper table for each scenario or dummy instance created with these values

|  |  |
| --- | --- |
| **DataValuesMapper Table fields** | **Value** |
| [DataValuesMapperID](http://amabdallah.github.io/test2/columns/DataValues_DataValuesMapper_DataValuesMapperID.html) | [ ] look up the highest existing integer value and add one. If the table is empty, then start with [1] |

Create a dummy Mapping table(s) with these values for each scenario

|  |  |
| --- | --- |
| **Mapping Table fields** | **Value** |
| AttributeID | [ ] look up the AttributeID for the ScenarioDummyAttribute |
| InstanceID | [ ] look up the InstanceID for the dummy “ScenarioDummyInstance” |
| SourceID | Look up the SourceName entered in the Scenarios spreadsheet |
| MethodID | Look up the SourceName entered in the Scenarios spreadsheet |
| DataValuesMapperID | [ ] look up the highest existing integer value and add one. If the table is empty, then start with [1] |

Create a dummy ScenarioMappingtable(s) with these values for each scenario

|  |  |
| --- | --- |
| **ScenarioMapping Table fields** | **Value** |
| ScenarioID | Look up the ScenarioID for each scenario in the network |
| MappingID | [ ] look up the MappingID for the combination of Attribute “ScenarioDummyAttribute”, ObjectType “ScenarioDummyObjectType”, and the Instance [same scenario name +”DummyInstance”] |

After populating the ScenarioMapping Table, the circle complete between a dataset and its scenarios and networks. Next, the Wizard/users will be able to import node and link Instances.

**Loading node Instances**

1. Populate the Instances Table from the provided data in the Nodes spreadsheet
2. Get the InstanceID, the SourceID, MethodID, AttributeID (through the provided ObjectName and the default dummy attribute for that Object which is called “ObjectTypeDummyAttribute”, and get the next available DataValueMapperID and populate all of them for one node instance into the Mapping Table.
3. Get the MappingID from point (2) and the provided ScenarioID and create an entry in the ScenarioMapping table

After finishing point (3), a node instance would be connected to its objectType, and network. Use a for-loop over each provided node entry in the spreadsheet table to all of them.

Implement this check: if the same node, with the same: source, method, Attribute which is called “ObjectTypeDummyAttribute”,Type are provided again but with only a different scenario name that is provided, then the Wizard will look up if the these combinations exist in the Mapping Table. If so, then the Wizard only populates the ScenarioMapping Table with the same MappingID but different scenarioID. This concept is very important in WaMDaM which allows the reuse of data across scenarios. It also allows users later to quickly query and compare if there is a difference between two scenarios.

**Loading link instances**

Follow the same loading steps and checks for the node instances above. However, the link spreadsheet contains additional data about the start and end nodes of each link. These extra data will be populated in the Connections Table in a following step. The start and end nodes are just foreign keys of other “node” instances while the LinkID is a foreign key to the Link InstanceID. The Instances Table in WaMDaM works as a supper class that shares the common metadata between nodes and links and the Connections Table stores the special extra metadata for “link” instances. Later we will implement a validation check that a link instance must have two different start and end node instances. In other words, a link cannot exist without both a start and end nodes.

**Loading global attributes**

Besides the attributes for each ObjectType, there are attributes for the entire Dataset that apply to the entire scenario. Example global attributes are like budget constraints or objective function values for the whole scenario.

The global attributes are populated similar to the regular attributes except that they are associated with the dummy ObjecType defined earlier called “ScenarioDummyObjectType”. Users won’t see the ScenarioDummyObjectType. Instead they will provide the Dataset name and the Wizard will look up the ID of the ScenarioDummyObjectType based on the provided dataset name. In the Mapping Table, you can use the ScenarioDummyInstance for all the global attributes.

**To maybe implement later (not included in this task)**

1. Each added instance that has the same ObjectInstancesID will inherit all the attributes of the same Object Type. Then, the Wizard will populate the Mapping table with all the Attributes for each new loaded instance. At the same time, once a new entry is added to the Mapping Table, the Wizard automatically creates a new entry in the ScenarioMapping Table that connects the instance, with an attribute, and a scenario. The ScenarioMapping table helps sharing data between scenarios without duplications. Only unique data differences between scenarios will be different between two scenarios.
2. The AttributeTypeCodeCV as defined for each Attribute will specify the data type for each attribute which will be implemented to all instance of the same Object Type. The Wizard needs to implement a software business rule that allows users to only populate one of the eight data types as chosen in the Attribute table.