Data Conversion Functional Specification v1.0

1. Introduction

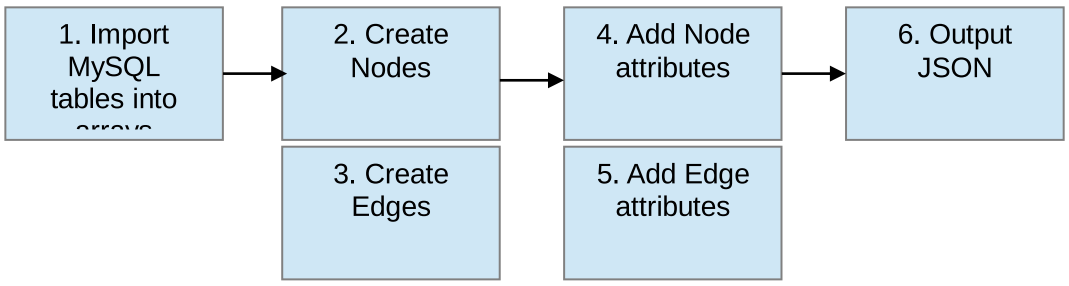
The requirement is for a stand-alone PHP data conversion routine. The routine should convert raw data from MySQL database tables into a JSON graph format composed of nodes (elements) and edges (links).

Whenever it is called, the routine will import a new set of raw data and convert it according to the input parameters provided. The resulting JSON will be output to a visualisation tool for graphical output.

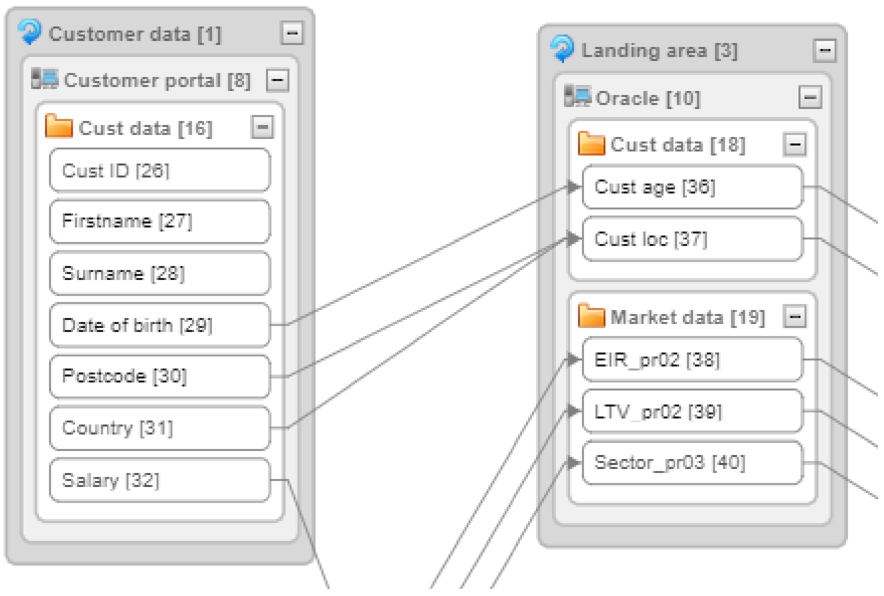
To help explain the functionality, a detailed worked example is provided. This describes each stage of the output (to be provided as a separate OpenOffice Calc spreadsheet).

2. Processing steps

The routine can be called from the command line, or preferably an HTML webpage front-end. It requires a Viewpoint (single string) to be provided as an input parameter.
The following diagram illustrates the main components:



The objective is to create JSON which displays the following type of schematic.
The boxes are Nodes. The connecting lines are Edges.



The box Node structure is composed of 4 parts:
Level 1: An outer Group (eg Customer data [1])
Level 2: An inner Group (eg Customer portal [8])
Level 3: An inner Group (eg Cust data [16])
Level 4: A leaf Node (eg Cust ID [26])

The layout format is fixed and there are always 4 levels. There are always 3 Groups containing one or more Elements. They are all referred to as Nodes. The groups are Group Nodes and the Elements are leaf nodes (as they do not have any children).

In addition, the leaf Node Elements can be linked to to one another. This connecting line is an Edge relationship.

3. Import Raw data and Mapping files

The first step is to load the test data from MySQL database tables, into a set of in-memory arrays and give all records an appropriate index so that they can be processed.

These are provided in worksheets 1-5
These can be considered data dictionaries that contain all the valid values that can be provided in the Mapping file.

The second step is to import the Mapping file, worksheet 6.
Every variation of the output JSON will be a different Mapping file.

The objective of the data conversion routine is to convert these input files into the JSON using a set of input variables.

The input variable for the first test scenario is very simple. It is VIE = “Customer data”

4. Create node_data

This is probably the most complex part of the routine.

As mentioned there are 4 different Node types. Three are Groups and one is the leaf (innermost) node. They are stacked into a box-within-a-box layout. The Group nodes are the 3 outer boxes. The leaf Node is the inner most box. Understanding this is key to creating the correct node structure.

The following rules are used to determine which of the 4 levels an element is:
If it is a Process, it is the outermost node (and a Group)
If it is an Application, it is the second box (and so needs to belong to a Process Group)
If it is a DataSet, it is the third box (and so needs to belong to an Application Group)
If it is a Data Element, it is a Node (and so needs to belong to a DataSet Group)

Nodes have properties. These are shown in the Examples.
As a general rule, everything has an ID and a Name.

5.Create link_data

This is more straightforward as Links are only possible between Data Elements.
Again,the worksheet example describes the attributes that can exist.

6. Output JSON

The final JSON file is constructed by combining the node_data array with the link_data array and adding some additional rows.

7. Design and Test considerations

The design of the routine should be easy to follow and easy to make enhancements to.
The code should be clear, easy to understand and well documented

As far as possible the design should be data-driven and avoid hard-coded parameters.