WebMIPMap tutorial

WebMIPMap is a web application specifically tailored for the needs of the Human Brain Project that offers schema mapping utilities. It provides an easy to use web interface where correspondences between schemata can be defined by simply drawing arrows between two tree-form representations and generates declarative representations, under the formalism of TGD rules.

Its purpose is allowing registered users of the HBP to create mappings between the MIP schema and hospital or research data schemata. These mappings can be used to translate hospital and research data schema terms to terms of the MIP schema in case this terminology does not exist in the MIP schema.

# Creating/Loading a new mapping task

To start a **new mapping task** the user clicks on ‘New’ under the ‘Mapping Task’ menu (Figure 1).

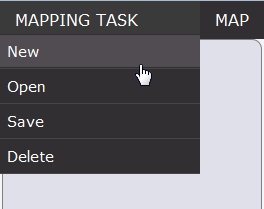


Figure 1 – Creating a new Mapping Task

A wizard dialog is then presented to him/her containing all available input options, first for the source schema and, after having completed all necessary information on the first screen, for the target schema. WebMIPMap can take as input the following types for the source and target schema:

1. a set of CSV files forming the database schema
2. an XSD file containing the schema definition in a hierarchy of elements
3. an SQL file containing the schema creation script
4. a relational database (MySQL/Postgres/Apache Derby) schema itself, given the connection uri to it and the credentials for the connection
5. a set of pre-defined saved schemata

The source and target schemata in a mapping task can have different input types. The user can choose among these options by selecting the corresponding option in the upper drop-down list (Figure 2).

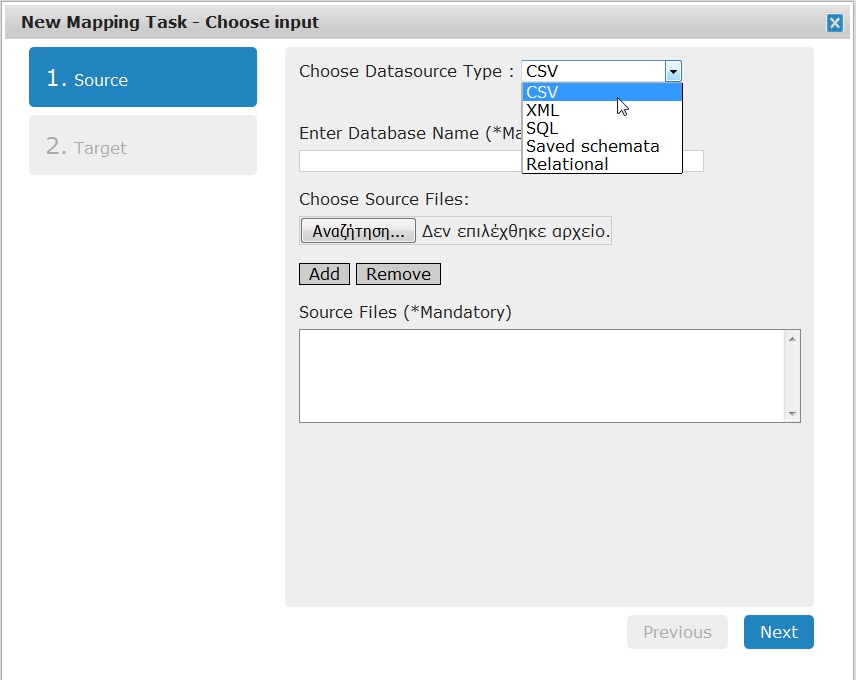


Figure 2 – Input Type Options

* For CSV input (Figure 3), the user must enter a database/schema name and select and add (or remove in case of mistakenly selected files) the CSV files that will compose the source schema, each corresponding to a table. The CSV delimiter used in input files should be the ‘comma’ (,) character.

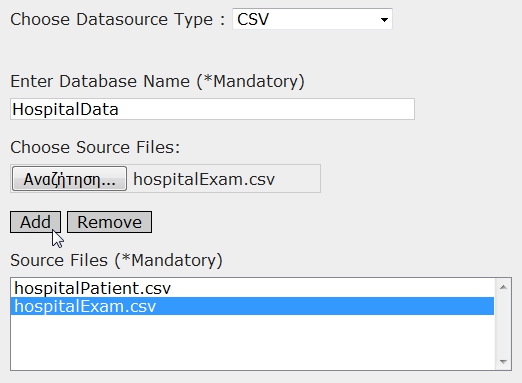


Figure 3 - Choosing CSV files as schema input

* For XML input the user selects just the XML Schema Definition (XSD) file containing the schema definition.
* For SQL input (Figure 4) the user should enter the database / schema name and select the SQL file with the script containing the creation and definition of the schema tables.

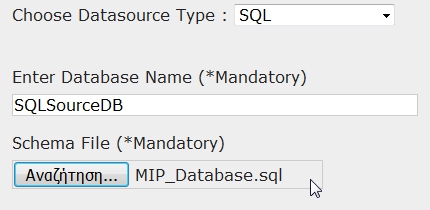


Figure 4 - Choosing SQL script file as schema input

* By selecting ‘Saved schemata’ (Figure 5) the user is presented with a list containing schema definitions uploaded in the server. For, example, the MIP schema can be selected as the target schema so that hospital and research schema terms can be mapped to a common used terminology.

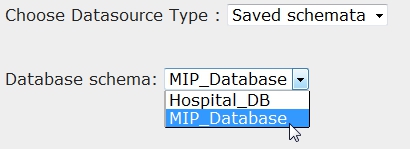


Figure 5 - Choosing between saved schemata

* For Relational input (Figure 6) a driver must first be selected. After that, most of the information in the ‘Uri’ field is filled automatically leaving to the user to fill in the host address and the database name. A username and a password for this database must also be entered.

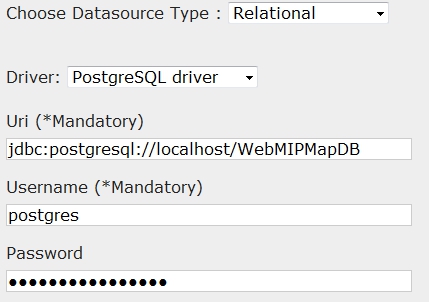


Figure 6 - Providing database information for target relational input

Required fields must be filled or selected before moving to the next step, otherwise the process will not continue and the user will be notified on the lack of information provided.

To **load an existing mapping task** the user selects ‘Open’ under the ‘Mapping Task’ menu and chooses a previously saved mapping task. The user has access only to the mapping tasks he/she has previously created and saved himself/herself.

After information for both the source and target schema has been provided, or a task has been selected for loading, MIPMap creates a tree-form representation for each of the two database schemata, one for the original source schema and one for the target one, taking into consideration existing primary keys.

The newly created mapping task is presented in the Mapping Task Main panel (Figure 7), which consists of three areas. In the left area, the user is presented with the tree-form representation of the source schema, where each leaf node corresponds to a table attribute followed by its data type. Primary key constraints -and possible foreign key saved on the mapping task xml- are also represented in the schema, the first by a ‘key’ icon on the appropriate element and the latter by a grey arrow connecting the referenced elements. The right-most area contains the respective representation for the target schema. The yellow middle area is used for creating constant and function correspondences and it will be thoroughly described in Section 2.1.

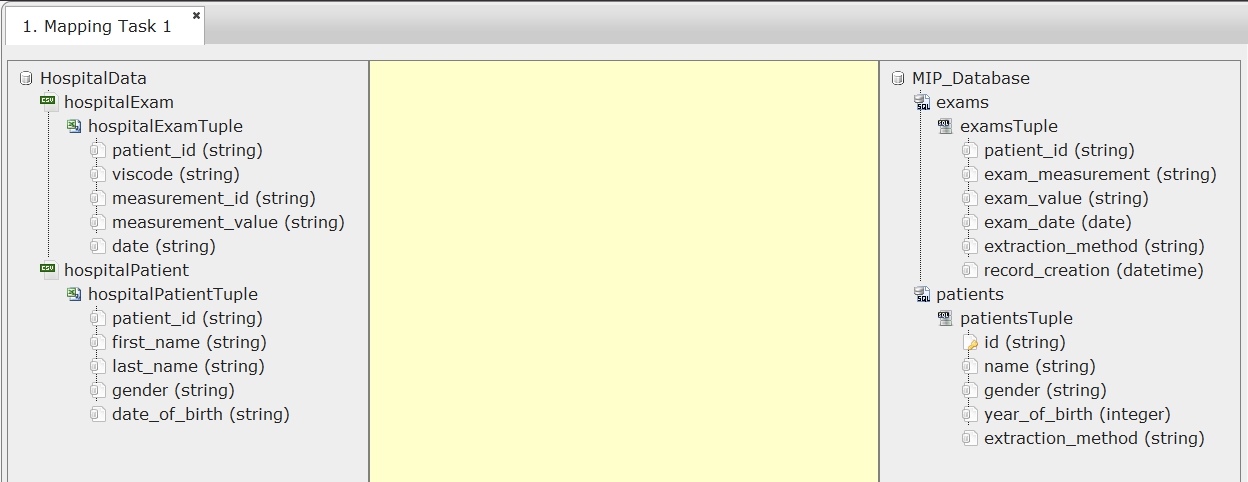


Figure 7 – New mapping task with CSV and saved schema (stored as SQL) input.   
No correspondences haves been drawn yet.

It should also be noted that more than one mapping tasks can be loaded. The ‘Scenarios’ panel on the left-most area of MIPMap offers navigation between the tasks and quick access to some of the most important features of the tool and will be described in detail in Section 5.

# Designing a mapping task

## Correspondences between elements

In the mapping task Main panel the user can drag arrows from one leaf node to another. In case the nodes belong to the same tree (source or target) a foreign key constraint is created, while when the nodes belong to different trees a **correspondence** between the two attributes is created. Note, that a correspondence from a target tree node to a source tree node is not allowed and cannot be created.

An additional option to create simple correspondences is provided (Figure 8). By right clicking on a leaf node the user has the option to set the current node either as source or target of a simple correspondence connection. A node must have been set as source for the connection before another one is selected as target. Additionally, only the last node that has been set as source is taken into account for the connection. This option can be used to create foreign keys too, if the two nodes selected belong to the same tree.

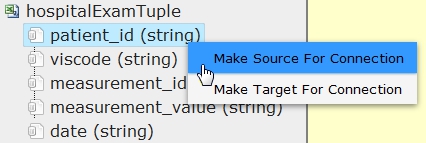


Figure 8 – Selecting a leaf node as the source point of a connection

When the user clicks on a leaf node the correspondences from -or to- that node are highlighted in blue making it easier for her to distinguish the path of the mapped elements (Figure 9).

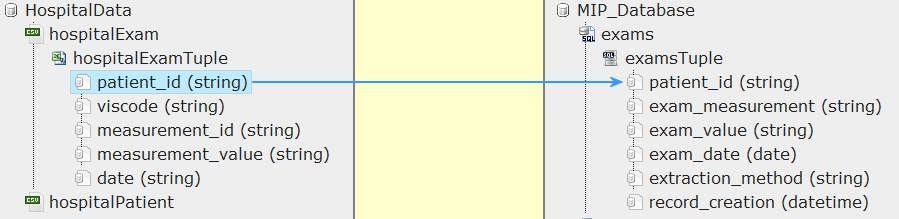


Figure 9 - Highlighted correspondence between two elements

By right clicking in the center area with the yellow background (Manipulation area) the following abilities are offered to the user:

* Define a **constant value** over the target instances (Figure 10). The constant value can either be a given *string* or *number* value or generated from a *function* (the date or date and time of the mapping task, a sequence number etc.). Constant value options are presented after double-clicking on the constant icon: C:\Users\OLAF\Documents\screenshots\screenshot.1452701112.jpg. An example of how this constant value is represented in the Manipulation area is shown in Figure 11.

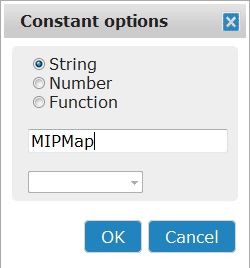


Figure 10 - Constant options

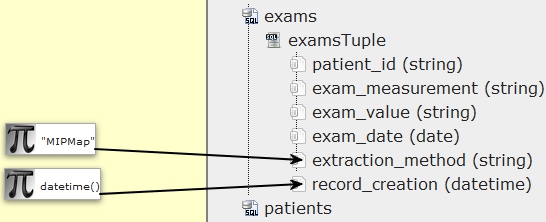


Figure 11 - Constant correspondence examples

* Create a complex **transformation function** (Figure 12) that uses one or more source attributes and maps the result to the target schema. Transformation function options are presented after providing input to and double-clicking on the function icon: C:\Users\OLAF\Documents\screenshots\screenshot.1452701121.jpg

There is a big selection of functions that the user can choose from, from string (Append, Substring, Trim, Lower/Upper case) to mathematical ones (Round, Absolute value, trigonometric functions), even more complex ones, like ‘If’ function, and casting ones (to String, to Integer). The user must first drag *at least one* arrow from a source leaf node to the function icon. More than one source nodes can be used as input to the transformation function. Then, by double clicking on the icon the function menu appears, in which available functions can be selected and inserted into the type area.

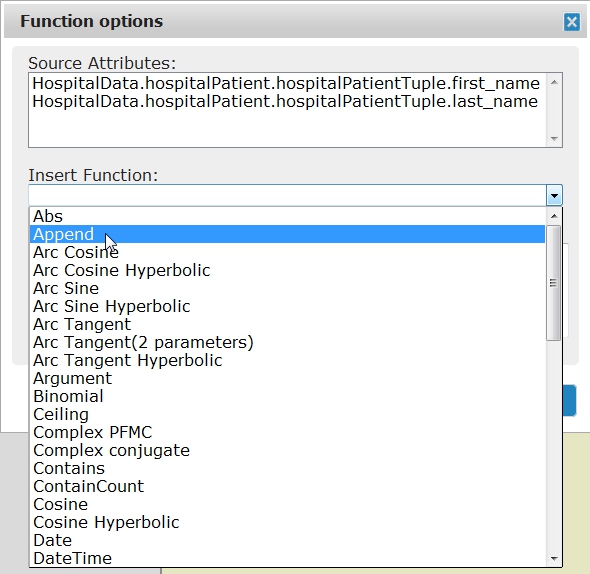


Figure 12 - Selecting a transformation function

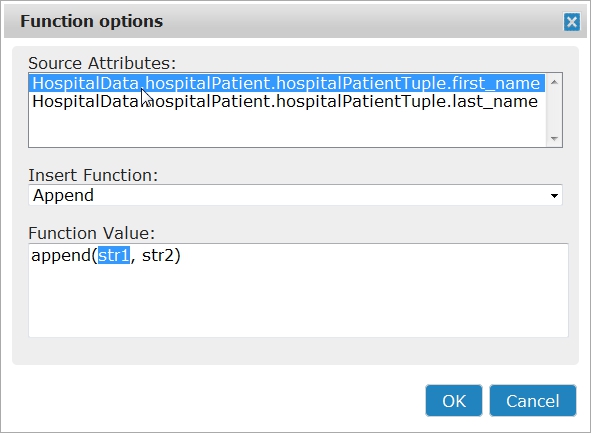


Figure 13 – Inserting Source Attributes to the Function Value textarea

In the type area the user enters the values of the input variables. String values must be enclosed in *double quotes* (” “), not in single quotes. Clicking on an item of the source attributes list adds it on the type area, so that the user will not have to type the whole path. Functions can be used inside other functions too.

*For example, to concatenate first name with last name into one field and separate them with a blank space the transformation function can be:*

append ( HospitalData.hospitalPatient.hospitalPatientTuple.first\_name,   
append (" ", HospitalData.hospitalPatient.hospitalPatientTuple.last\_name) )

*or to get the year from a date string:*

substring( HospitalData.hospitalPatient.hospitalPatientTuple.date\_of\_birth, (  
len (HospitalData.hospitalPatient.hospitalPatientTuple.date\_of\_birth) – 4 ) )

The user then can drag an arrow from the function icon to one *or more* leaf nodes of the target tree to assign its value to it (Figure 14).



Figure 14 – Transformation Function correspondence example

By right clicking on a connection arrow the user can select to delete the specific connection (Figure 15). Deleting a constant or function icon also deletes all connections associated with it. A “Delete all connections” option is also offered by right clicking on the grey area outside the trees.

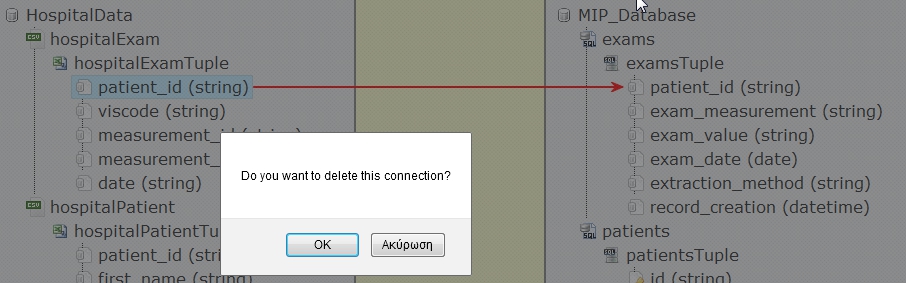


Figure 15 – Deleting an existing connection

## Join conditions and Foreign key constraints

MIPMap also offers users the ability to apply **join** **conditions** (Figure 16) to the source sets by specifying a join dependency between two elements as ‘Mandatory’. Join dependencies are defined with drag-and-drop operations among elements of the same schema and are represented by a grey line –or arrow in case they are not bidirectional and corresponding to a foreign key constraint as previously noted.

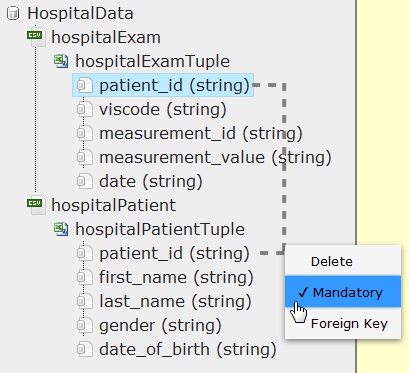


Figure 16 - Applying a mandatory join condition between two attributes

The user can right click on the connecting line and define the dependency as *mandatory* or *optional*, making the line bold, as well as choose if it is bidirectional or not (‘Foreign Key’ option).

## Additional options

In addition, **selection conditions (**Figure **17,** Figure **18)** can be applied to source sets by right clicking on a set node (the ones representing the tables), choosing to ‘Edit Selection Condition’ and typing the appropriate condition which will be then shown next to the table name. Typing the desired column name (leaf node name) is adequate, meaning it isn’t necessary for the user to type the whole tree path to the leaf node. Supported symbols are equality (==, !=), comparison (<, >, <=, >=), arithmetic (+, -, etc) and conditional operators (&&, ||), while, as in transformation functions, string values must be enclosed *in double quotes (“ “)* and *not* in single quotes*.*

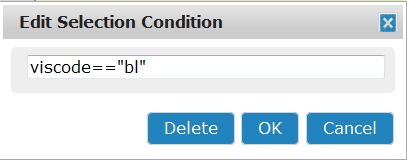


Figure 17 - Typing a selection condition, so that only baseline (“bl”) exams will be translated.

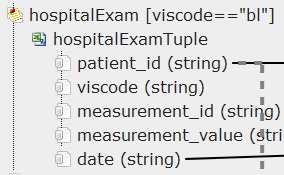


Figure 18 - Created selection condition appearing on node

# TGD Generation

After correspondences between elements have been defined the user can compile the mapping scenario and **generate TGDs** (Tuple Generating Dependencies) [1] by clicking on ‘Generate Transformations’ under the ‘Map’ menu.

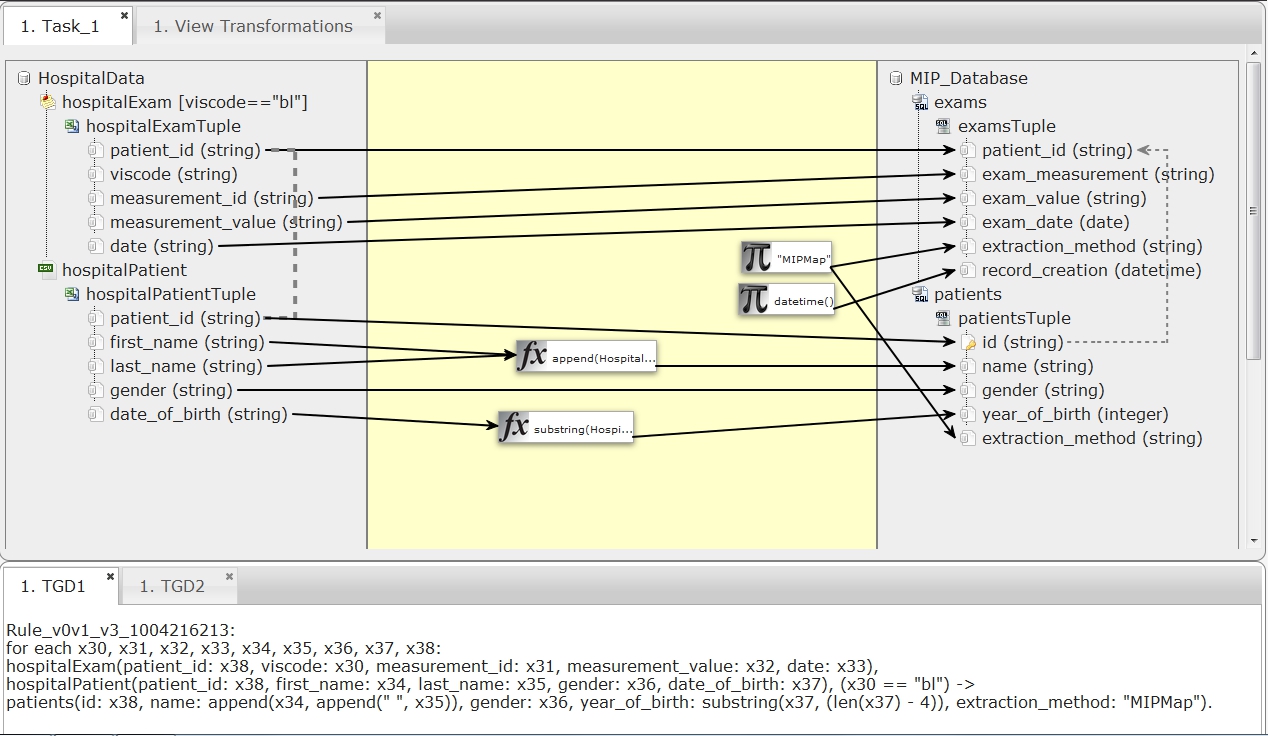


Figure 19 - Full mapping task with generated TGDs

The following two panels are shown (Figure 19):

* The ‘View Transformations Window’ which includes all the details about the schemata, their constraints, the original source-target dependencies and their rewritings to enforce target egds and compute optimal solutions.
* The ‘View TGDs’ panel that appears below the main panel and lists all TGD rules that compose the final mapping between the two schemata.

In case *only constant values* are mapped to a target table a corresponding pseudo-TGD rule, named ‘**Constant FORule**’, is generated and represented in the graphical interface.

If changes in the mapping task have been made (e.g. a new correspondence, constraint etc. is added, edited or removed) and one of those panels is clicked, MIPMap notices the changes and notifies the user asking to refresh its content.

The TGDs that are produced are **rewritten** by the MIPMap engine in order to ensure that the solutions generated are optimal.

The option to export the generated TGD rules is also offered. By clicking on ‘Export TGDs’ under the ‘Map’ menu the user can choose to export and save the TGDs locally on his/her machine.

Once the mapping has been compiled in its logical form, it is possible to **generate the corresponding executable scripts** (SQL or XQuery) from the ‘Scenarios’ panel or from the system bar under the Map menu (Figure 20). The SQL script produced can be used externally for the translation of data between the two schemata that follows exactly the rules specified.

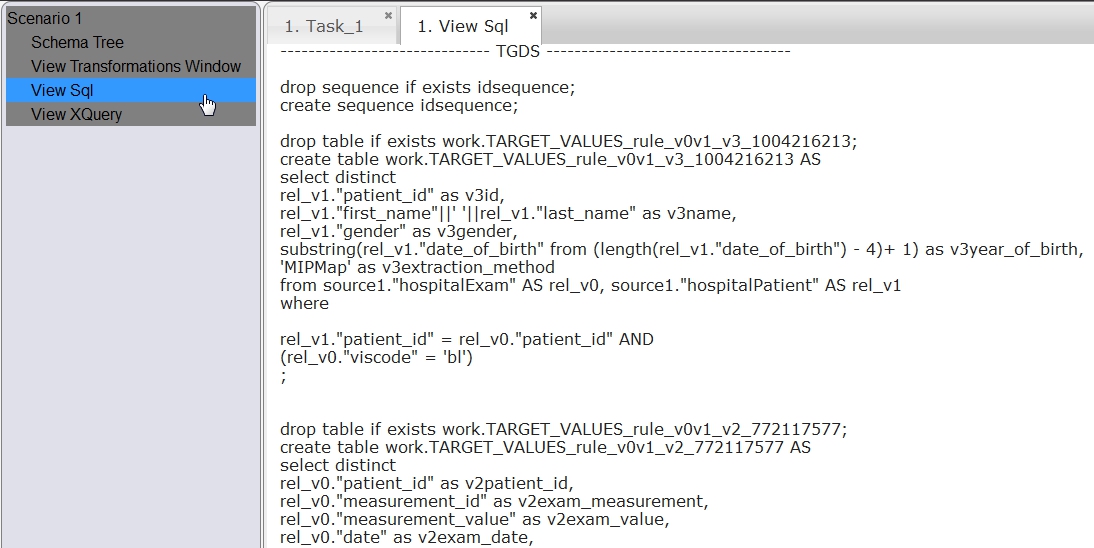


Figure 20 – SQL script generated for external data exchange proccess

# Saving / deleting a mapping task

By clicking on the x icon of the main mapping task area the scenario is closed. Mapping tasks can be **saved** in order to be easily accessed in the future, by clicking on ‘Save’ under the ‘Mapping task’ menu and typing the name of the mapping task the user wishes (Figure 21).

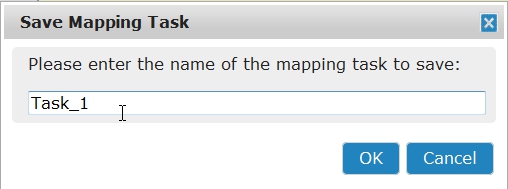


Figure 21 – Saving a mapping task

As it has been mentioned in Section 1, previously saved mapping tasks of the current user can be loaded in the future. The user is also given the option to **‘Delete’ a mapping task**. Deleted mapping tasks are erased from the server. Confirmation is requested in this case, while currently opened mapping tasks must first be closed before selected to be deleted.

# ‘Scenarios’ configuration area

If more than one mapping tasks are opened simultaneously the TGD Generation and Translation actions correspond to the mapping task that is currently selected. By default the last opened mapping task is the one selected and is highlighted at the rightmost ‘Scenarios’ panel. The user can **select a task** as the current one via the Scenarios panel by right clicking on the desired scenario (Figure 22). The option to close the mapping task is also provided.

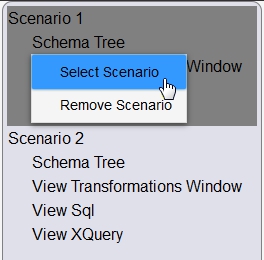


Figure 22 – Selecting between two opened Scenarios

The Transformations, SQL and XQuery script tabs gain focus when clicked on the rightmost Scenarios panel or they can be opened if they are not loaded through the panel as well.

# Crowd-sourcing functionality

* 1. **Save/Open global mapping task**

Administrators will be able to store global mappings that will be made available to all WebMIPMap users. If a user has administrator rights, by clicking on button 'Save Mapping Task As Global' which is located under the tab 'GLOBAL TASKS' can save the active scenario as global. Every WebMIPMap user can load global mapping tasks, by clicking on button 'Open Global Mapping Task' in the same tab option.

* 1. **Save public mapping task**

A user can define public mapping scenarios, i.e. scenarios that can be viewed by other users. After creating a mapping scenario the user can save this scenario in their public folder. By clicking on button 'Save Mapping Task As Public' which is located under the tab 'PUBLIC TASKS' the active scenario will be saved as public.

* 1. **Send Trust Request**

By clicking on button 'Send Request' in tab 'PUBLIC TASKS' appears the following pop window (Figure 23).

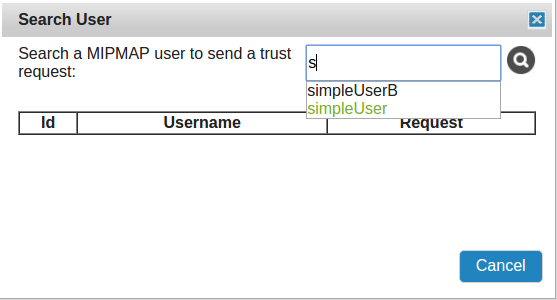


Figure 23 – Search MIPMap users

By inserting a letter or more of a MIPMap user, a list of user names appearing as list. After selecting a user and clicking on the search image button, the selected user added in the table (Figure 24).

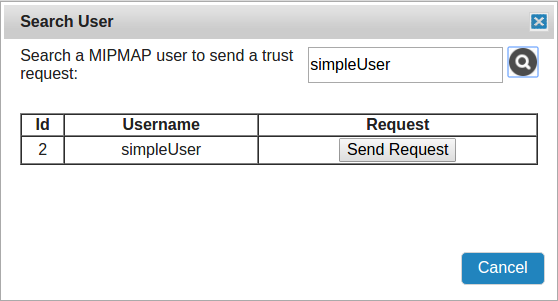


Figure 24 – Select MIPMap users

Then, by clicking on button 'Send Request', a trusted request is sent in the selected user.

* 1. **Pending Requests**

By clicking on button 'Pending Requests' in tab 'PUBLIC TASKS' appears the following pop window (Figure 25).

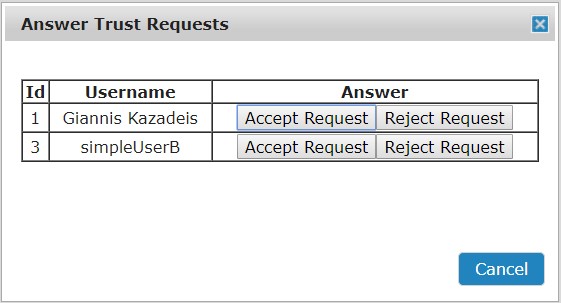


Figure 25 – Pending Requests

Then, by clicking on button 'Accept Request' in any of the trusted users, the respective user is regarded trusted and consequently, have access in current user’s public tasks. On the other hand, if the current user selects to reject a trusted request, the requested user does not acquire access to current user’s public tasks and the request is not visible anymore. The rejected user can re-submit the trusted request in the future.

* 1. **Remove Trusted Users**

By clicking on button 'Remove Trusted Users' in tab 'PUBLIC TASKS' appears the following pop window (Figure 26).

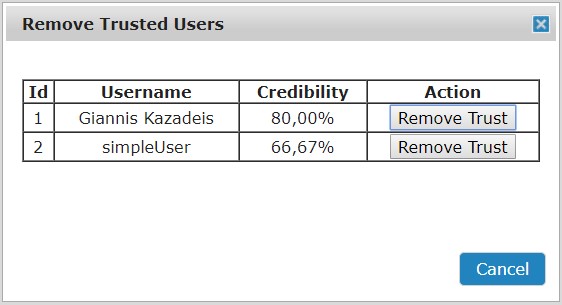


Figure 26 – Remove Trusted Users

Then, by clicking on button 'Remove Trust' in any of the trusted users, the respective user is not regarded trusted anymore and consequently, they have not access in current user’s public tasks.

* 1. **List Trusted User**

By clicking on button 'List Trusted Users' in tab 'PUBLIC TASKS' appears the following pop window (Figure 27).

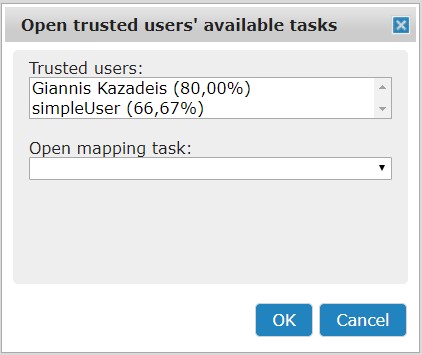


Figure 27 – List Trusted Users

Now, it appears a window which includes the list of trusted users and their credibility. After clicking on a trusted user, selected user’s public mapping tasks appear in the dropdown list under the label ‘Open mapping Task’. By selecting a mapping task from the aforementioned dropdown list and clicking on the button 'OK', the selected mapping task opens in the Mapping Task Main panel and the current user can edit the scenario by adding, changing or removing correspondence of the selected mapping task.

* 1. **Recommend Mapping Task**

By clicking on button 'Recommend Mapping Task' in tab 'PUBLIC TASKS' a recommended mapping task generated by the preferences of current user’s trusted friends. It is worthwhile to mention that in order to activate this functionality, the current user should open a mapping task with or without correspondences from their private or public mapping tasks’ repository. Then, the generated mapping task will be saved in user’s private or public repository depending from which folder the ‘base’ mapping task opened. Further information about this functionality is available in the **“WebMIPMap Report.dock”**.

# References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. G. Mecca, P. Papotti, S. Raunich - Core Schema Mappings. In SIGMOD Conference (2009)