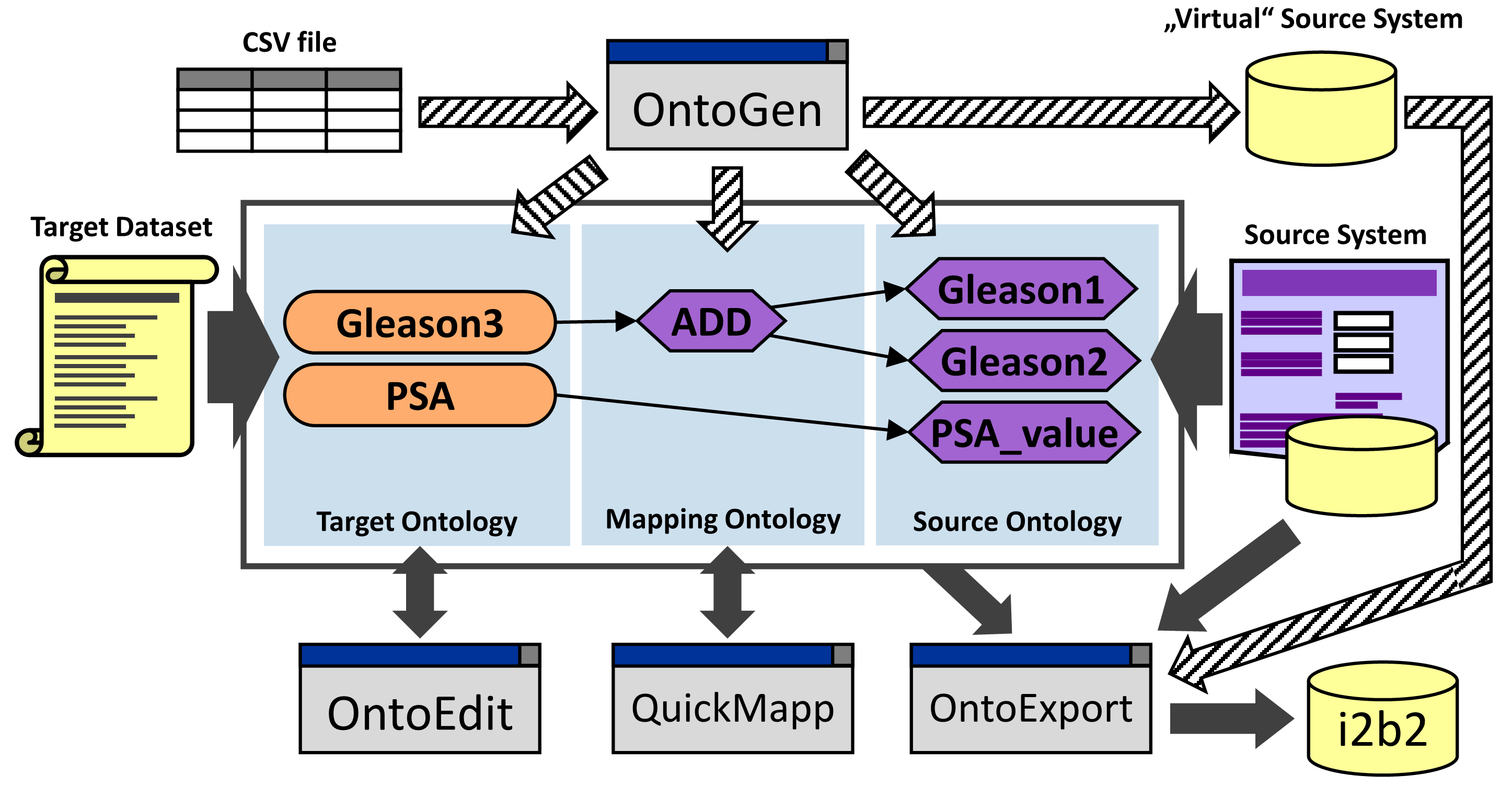
# Loading CSV Files into an i2b2 Installation – The Ontology Way

(Version #1, 12.08.2011, Sebastian Mate)

## Introduction

The “i2b2 OntoImport Suite” is a set of prototypical tools, developed in Erlangen, for the integration of (theoretically) distinct, heterogeneous, medical data sources. The idea is to describe the source and target systems, as well as the data integration (what to do with the date) with OWL ontologies, which are then automatically translated into SQL statements. Besides accessing generic medical systems, for example an EMR (which is quite difficult), the suite includes a tool “OntoGen” which allows CSV files to be used as a “virtual source system”. This is described in this document.

The role of OntoGen is illustrated with the dashed arrows in the image below:



The overall goal is to load the CSV file (upper left) into the i2b2 database (lower right). The first thing OntoGen does is to read the whole CSV file and to perform a column-to-row transformation. That means it converts the CSV file into the EAV (entity-attribute-value) format. It creates an Output.sql file, which can be uploaded into an Oracle database and acts a “Virtual” Source System (upper right). The final data integration is performed by OntoExport (lower right), which processes all the “ontology stuff” in the middle (blue). Because this “ontology stuff” is required by OntoExport, the second thing OntoGen does is to create this “ontology stuff”. Modifying the ontologies (e. g. with OntoEdit and QuickMapp) is not necessarily required, but in most cases desired. This will be explained later. Please note that in this scenario, there is no real “Source System” (EMR). We use the “Virtual Source System” instead – our CSV file, stored in EAV format in our database.

## Step-by-Step – how to do it!

### Annotating the CSV file

Let’s assume that we have this CSV file to start with (this is the example CSV file which is included in the package):



For every patient, there is a new row. The patient’s ID is stored in the first column. The other columns contain various attributes (first row) and values (below the first row).

Besides loading the actual data (white cells) into the i2b2 database, we also have to construct an “i2b2 ontology” from the CSV file’s metadata. Here, the metadata basically are the column’s headlines in the first row (blue) and in some cases the aggregated (unique) values in the columns below. OntoGen uses this information to create the “Target Ontology” (see figure in Introduction section), which is later transformed into the “i2b2 Ontology”. However, it cannot guess the data type and does not know which columns should be aggregated. Therefore, we need to add additional information to the table.

The first thing to do is to add the data type in the second column:



As shown in the table, the data type has to be added in the second row. The column with the patient IDs has to be annotated with “PatientID”.

Valid options are:

* **Cell is empty:** the values in the rows below are being aggregated. For example, for the “Sex” column, this would result in four entries in the i2b2 Ontology: “Male”, “Female”, “M” and “F”.
* **Cell is not empty:**
  + **Integer, PosInteger, Float, PosFloat, String:** These are valid i2b2 datatypes. For the numeric types, this results in the proper creation of the XML field of the i2b2 ontology table. The values are not being aggregated. For example the tool would create a single “Age” i2b2 ontology entry, which can be queried by value.
  + **Anything else (like “Date” in the example):** This would result in a single i2b2 ontology entry, the values are not being aggregated. This is true for the “Date of LabValue” column in the example above.

**The default strategy is:** If you want separate entries in the i2b2 ontology, just leave the cell in the second row empty. If it’s an i2b2 data type, put that one in. If it’s something else, enter something that is not an i2b2 data type (see above).

The second thing to do is to add the other missing columns:

The whole system uses an internal EAV-like data representation to process the facts data. Therefore, it is absolutely necessary to make sure that the CSV file contains several special columns:



Besides the PatientID column, which was already shown, make sure that there are also columns which are annotated with “GlobalDate” and “DocumentID”.

“GlobalDate” contains a timestamp which is, by default, valid for all values in this row. Later, we will show how to link the “Date of Lab Value” to the “Lab Value” column in QuickMapp, which overrides the “default” date relationship in this example.

“DocumentID” is an ascending number. Calculations between data records can only be performed on the same DocumentID. For this tabular representation, these are all other entries in this row. For CSV files just make sure that such a column exists.

Save the table in CSV format as “Input.csv” and put this file into the program directory of the OntoImport Suite.

### Using OntoGen

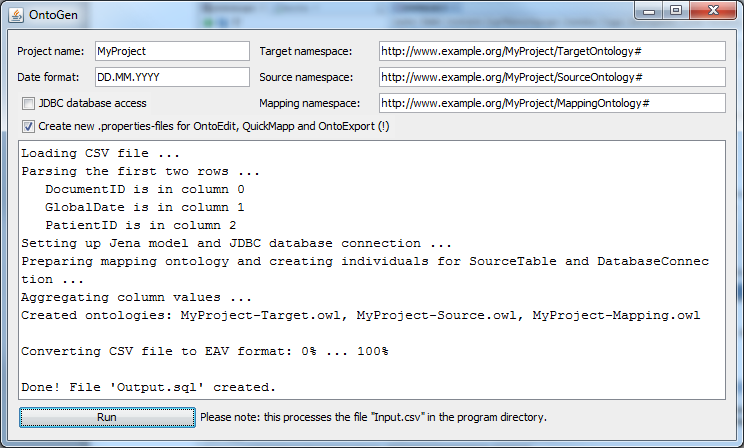
Remember: OntoGen does two things. The first thing is to convert the CSV file to a EAV representation. The second thing is to create the tree ontologies.

First, create a new i2b2 project with i2b2 Wizard. In our example, we call it “MyProject” (as it is suggested in i2b2 Wizard when you create a new i2b2 project).

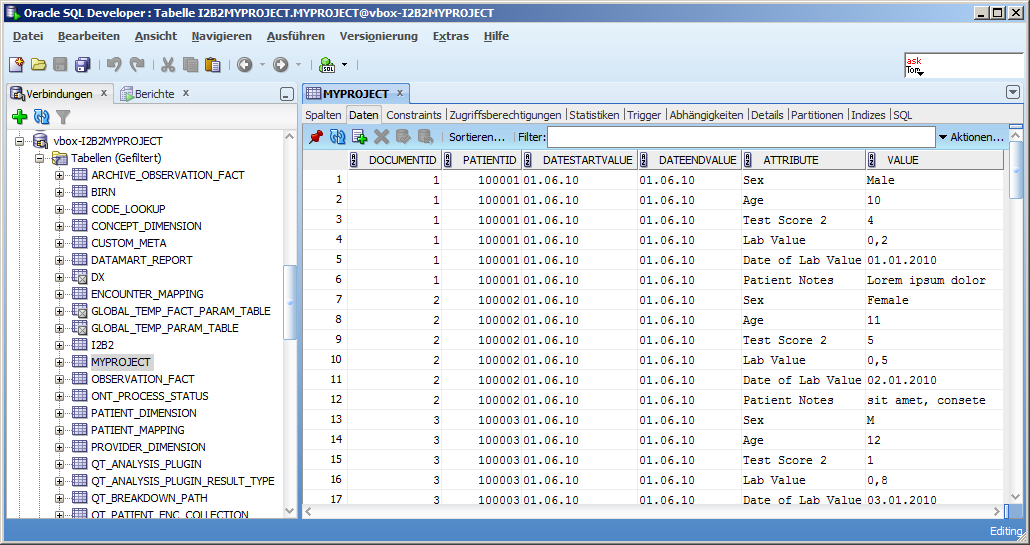
Optional: edit the file ExportConnection.properties to match your new i2b2 project. The schema is always “I2B2” + project name if you use i2b2 Wizard. In our example, this is “I2B2MYPROJECT”.

It is our goal to upload the CSV file (converted to EAV by OntoGen) into the i2b2 project schema. This new table then acts as our “virtual source system”.

Run OntoGen. If you want to, modify “Project Name” and the namespaces. Make sure that that “Date format” matches the date format in the “GlobalDate” type column of the CSV file. Check the “Create new .properties-files” checkbox. If you have edited the ExportConnection.properties file, you can also check “JDBC database access”. Click “Run”.



If “JDBC database access” was checked, OntoGen should have created a new table inside the i2b2 project schema. Its name matches the entered “Project Name”. If “JDBC access” was not checked, please open the created file “Oputput.sql” in the program directory and run the SQL file for example in SQLDeveloper inside i2b2 project schema. This is what the result should look like:



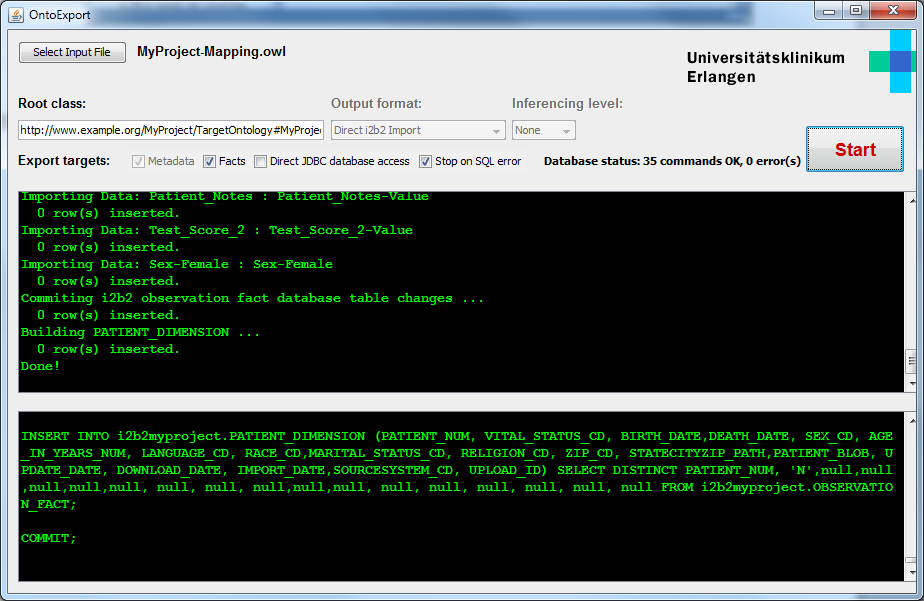
Also note that OntoGen has created three OWL files which are located in the main program directory: MyProject-Target.owl, MyProject-Source.owl and MyProject-Mapping.owl. These files are all required by OntExport to perform the whole ETL process (please refer to the illustration on the first page). The other tools OntoEdit and QuickMapp can be used to tweak the mapping, but for now, we skip this and directly use OntoExport to perform the data export.

### Using OntoExport

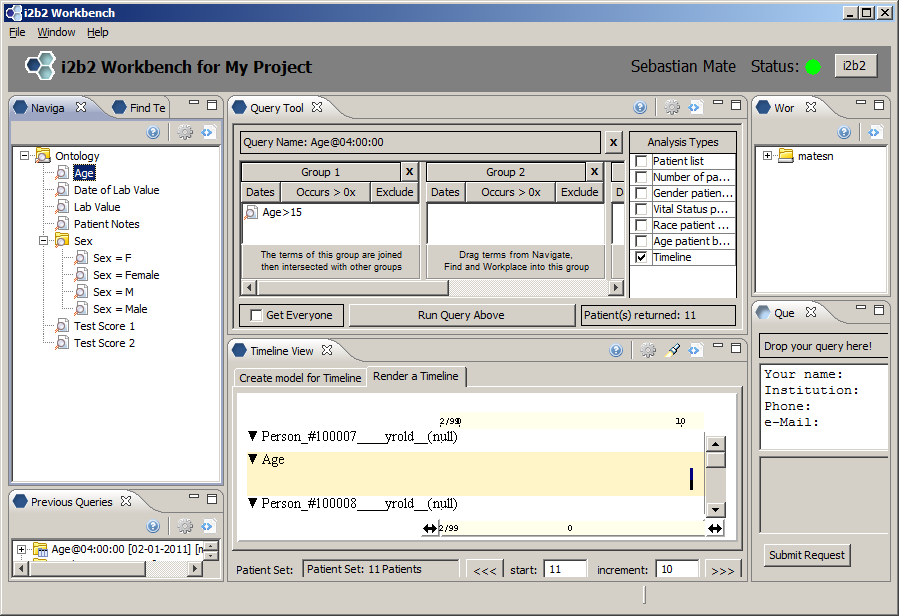
OntoExport reads the three ontologies and loads the data into the i2b2 database. The “Target Ontology” is used to build the i2b2 ontology. The “Source Ontology” contains the description of our source system (which is, in our example, the MYPROJECT table in our i2b2 project schema). For example, it describes how to access this database to get the desired data records belonging to a medical concept (for example: give me all patients with Sex = F). The “Mapping Ontology” defines the mappings and possible data transformation rules between the “Target” and the “Source Ontology”. It can be edited with QuickMapp, which will be shown later. The created “Mapping Ontology” only contains simple 1:1 mappings since the concepts in the “Source” and “Target Ontology” are basically the same.

Optional: edit the file ExportConnection.properties to match your new i2b2 project. The schema is always “I2B2” + project name if you use i2b2 Wizard. In our example, this is “I2B2MYPROJECT”.

Run OntoExport. Make sure that “Facts” is checked and then press “Start”:



If “Direct JDBC database access” was checked, OntoExport should have done everything automatically and the i2b2 database should be populated with the CSV file’s data. If “Direct JDBC database access” was not checked, run the SQL statements from the lower output console inside your i2b2 project schema, for example in SQLDeveloper. To check if everything is working, launch i2b2 and log in:

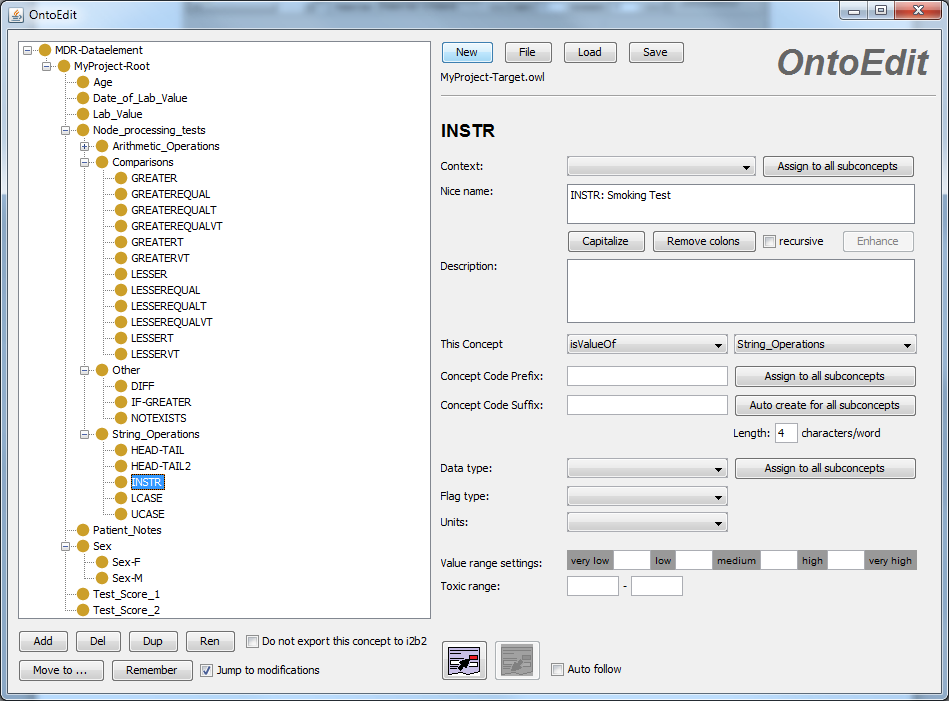


Congratulations, you have successfully imported the data from the CSV file! ☺

### OntoEdit and QuickMapp overview

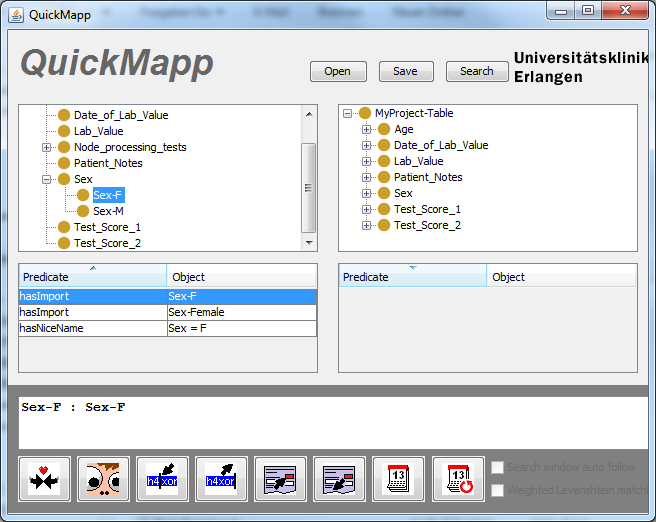
As mentioned above, the “Mapping Ontology” created by OntoGen only contains simple 1:1 mappings, which correspond to the CSV file’s structure (column headlines and aggregated values). To demonstrate the mapping capabilities of the system, predefined test mappings are supplied inside the *TestMappings* subdirectory. Copy the two OWL files from this directory into the main program directory to replace the existing “Target” and Mapping Ontology”. Please note that this only works if you have followed this document and have used “MyProject” as project name.

To edit the “Target Ontology”, run OntoEdit and press “Load”. You can use this tool to create/modify concepts inside the “Target Ontology”. The right side allows you to enter i2b2-Ontology specific parameters:



All concepts in the “Node\_processing\_tests” tree have been manually created to demonstrate/test some data transformation functions.

Also note that the Sex concept now only contains “Sex-F” and “Sex-M” as sub-concepts. “Sex-Male” and “Sex-Female” have been deleted, because we can map the latter concepts to the previous concepts. In QuickMapp, you can see this because two mappings have been created:



Highlighted are the two statements:

**Sex-F** *hasImport* **Sex-F.  
 Sex-F** *hasImport* **Sex-Female.**

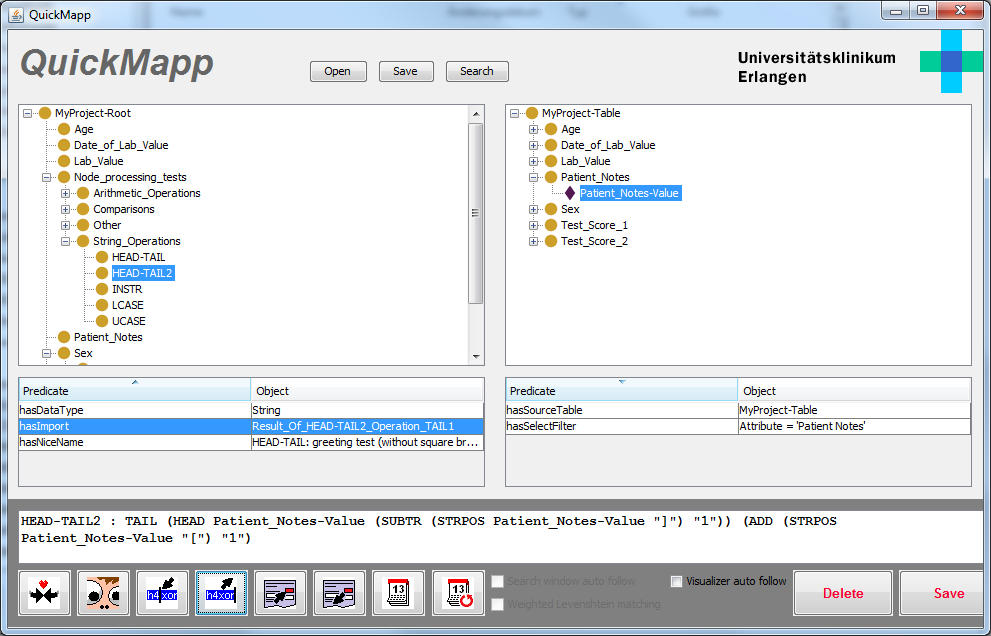
The concepts in the „Source Ontology“

The concepts in the „Target Ontology“

This means that we combine/merge the data records from “Sex-F” and “Sex-Female” on the right side (source system) to one concept in the i2b2 system (target system).

The created relationship between the concepts is stored in the “Mapping Ontology”. These mappings are simple mappings. In the illustration on the first page, the relation **PSA** *hasImport* **PSA\_value** is shown.

### Using QuickMapp



**Properties of selected node in Trgt. Ont.**

**Properties of selected node in Src. Ont.**

**Mapping Expression Editor**

**Source Ontology**

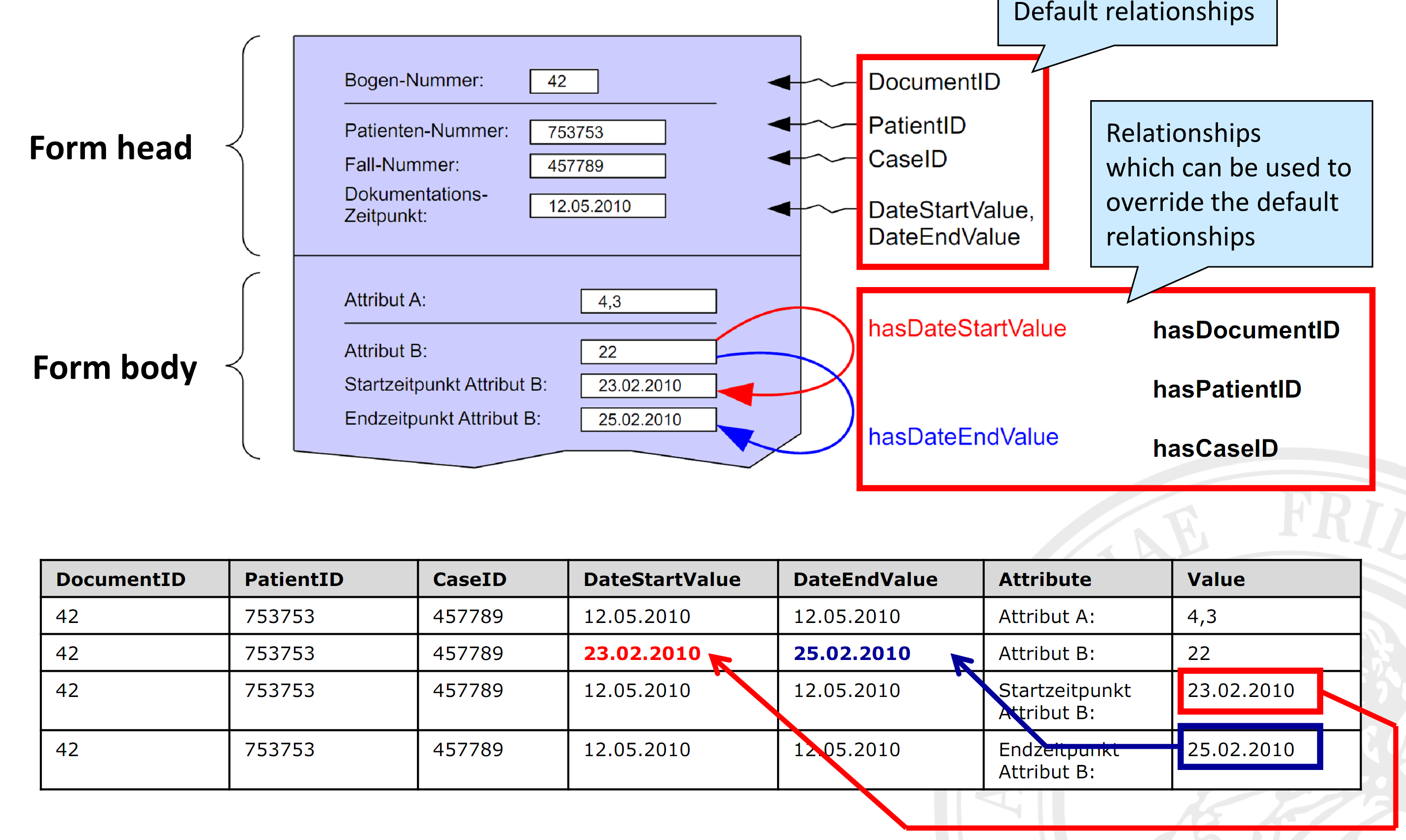
**Target Ontology**

Buttons, from left to right:

* Create a simple mapping expression between the selected concepts in the Target and Source Ontology.
* Visualize mapping tree (not implemented).
* Insert the concept’s name from the selected concept of the Source Ontology into the Editor.
* Jump to the selected concept node (inside the mapping editor).
* Show the concept in the form visualizer (requires the SoarianFormRenderer, which is specific for the UK-Erlangen)
* Select the Ontology node which matches the selected form element from the SoarianFormRenderer.
* Create a *hasDate* relationship. This replaces the default timestamp with the timestamp that is documented in a different “concept”. See next section!
* Recursively create a *hasDate* relationship for all selected concepts and sub-concepts of the source ontology.
* Delete the statement / mapping expression, which is shown inside the mapping expression editor from the Mapping Ontology.
* Save the statement / mapping expression, which is shown inside the mapping expression editor in the Mapping Ontology.

### Overriding the Date Relation

The data model, on which the whole systems operates, is designed to allow “overriding” attributes of data records:



However, the current implementation only has support for a “hasDate” relation, which combines the “hasDateStartValue” and “hasDateEndValue” relationship from the theoretical approach. This is, hower, sufficient in most cases. It will be improved in future versions of the tools.

To continue with the initial example: we want to override the “MySheet Date” timestamp for the “Lab Value” concept with the timestamps in the column “Date of Lab Value”.



What we need to do is to define the following statement inside the mapping ontology:

**Lab\_Value** *hasDate* **Date\_of\_Lab\_Value.**

In QuickMapp, this can be done with the “calendar” buttons. Or simply enter the relationship into the expression editor and click on “Save”:

