Demonstration of final mediation access tools and services

Project Number: FP6-2005-IST-026996

Deliverable id: D7.5

Deliverable name: Demonstration of final mediation access tools and services

Submission Date: 27/11/2008

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D7.5
Document name:	Demonstration of final mediation access tools and services
Document type (PU, INT, RE)	RE
Version:	1.0
Submission date:	27/11/2008
Editor: Organisation: Email:	Luis Martín Universidad Politécnica de Madrid lmartin@infomed.dia.fi.upm.es

Document type PU = public, INT = internal, RE = restricted

ABSTRACT:

The ACGT Data Access Infrastructure provides a set of user tools that aid in the process of accessing disparate heterogeneous data. These tools are, namely, the Semantic Mediator, the Mapping Tool and the Query Builder. This document includes a simple guide for each of them and shows their behaviour in a simple use case.

KEYWORD LIST: clinical trials, database integration, web services, semantic mediation, ontologies

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Editor**
0.1	01/11/2008	Draft	Luis Martin
0.2	14/11/2008	Draft	Luis Martin
0.3	20/11/2008	Draft	Luis Martín
1.0	20/11/2008	Final	Luis Martín

List of contributors:

- Luis Martín, UPM
- Alberto Anguita, UPM
- Daniel Mencia, UPM
- Enrique Díez, UPM
- Ana Jiménez, UPM

Contents

1. Introduction

The ACGT Data Access Infrastructure (ACGT-DAI) is comprised by a set of services working together in the task of providing seamless, integrated access to heterogeneous sources of information. The architecture of the ACGT-DAI can be divided into two different layers. While the lower one, called ACGT Data Access Services, is devoted to solve syntactic and technological problems in data, the upper layer (ACGT Semantic Mediation Layer, ACGT-SM) performs data integration and semantic mediation. Figure 1 depicts the architecture of the ACGT-SM.

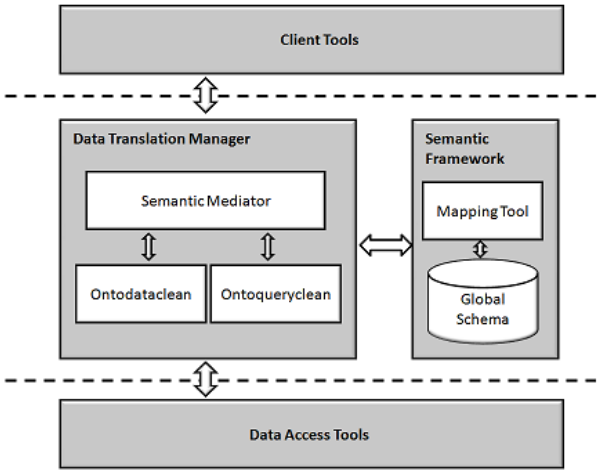

Figure 1: Architecture of the semantic mediation layer

The problem of offering seamless and unified access to heterogeneous sources of information is not a simple one. In fact, the range of issues that the tools and services this document describes deal with belong to very different nature, and goes from solving semantic heterogeneities in databases to ease end users the process of formulating a query. In figure 1, the Semantic Mediation Layer is comprised two different macro-components, the semantic framework, that includes components implied in the semantic homogenization of data—i.e. the Mapping tool and the ACGT Master Ontology—, and the data translation manager, grouping the components that deal with the different aspects of data integration. As can be observed, the problem of aiding end users in the process of building queries is not addressed by any tool at this level. The reason is that in the ACGT architecture the query builder is considered an end user tool, so it belongs to the client tools layer. However, given the importance of the problem tackled by this component, and its relation with the mediation process, it is regarded to be a part of the semantic access infrastructure, and described in detail in this document.

The structure of this document is as follows: Section 2 includes a simple user manual for each of the three tools. Section 3 describes the behaviour of the tools in a simple use case.

2. Semantic mediation tools and services

2.1. The ACGT semantic mediator

The Semantic Mediator is the core component of the ACGT Semantic Mediation layer. It is in charge of accepting queries in terms of the ACGT Master Ontology and translating them into terms of the physical databases included in the integration platform. The Semantic Mediator can be accessed as an OGSA-DAI service, making it available to any terminal connected to the internet. Three different services comprise the Semantic Mediator:

- *SemanticMediator*: this is the main service offered by the Semantic Mediator (and it is called also as the resource). It offers a SPARQL interface for performing queries in terms of the Master Ontology. The received queries are handled by the Semantic Mediator accordingly to the existing database mappings, so a new query for each underlying data source is produced and their results are properly merged and sent back to the user
- *MappingList*: this service allows retrieving the content of all mapping files included in the Semantic Mediator. This service is used by the Query Tool.
- *updateMappingList*: through this service, new mappings can be included in the integration platform. This service is used by the Mapping Tool.

OGSA-DAI services are invoked using perform documents. The complete specification of this type of documents is described in [OGSADAI]. Each of the three services of the Semantic Mediator requires a different set of arguments, and produces a different type of result. Next subsections depict the details and interface of the three services.

SemanticMediator

This service accepts a variable number of arguments, depending on how the user wants to retrieve the results. The first approach requires two arguments, as described below:

- query: the content of the SPARQL query, in terms of the Master Ontology
- type: how the results must be retrieved. If the value of this argument is “CSV”, then no more arguments are required, and the result of this invocations will be directly the content of the SPARQL Result Set containing the integrated results for the given query
- SemanticMediatorOutput: the name to which the results will be associated

This method of invocation is used by the Query Tool. Figure 2 gives an example of perform document in this case.

```
<?xml version="1.0" encoding="UTF-8" ?>
<perform xmlns="http://ogsadai.org.uk/namespaces/2005/10/types">
  <documentation>Simple query across all tables in the database.</documentation>
  <SemanticMediatorWIP name="myActivityInstance">
    <query>PREFIX acgt: <http://www.ifomis.org/acgt/1.0#> PREFIX xsd:
      <http://www.w3.org/2001/XMLSchema#> PREFIX bfo_span:
      <http://www.ifomis.org/bfo/1.1/span#> PREFIX bfo_snap:
      <http://www.ifomis.org/bfo/1.1/snap#> PREFIX ro:
      <http://www.ifomis.org/obo/ro/1.0#> SELECT ?patientID ?birthDate ?regDate
      WHERE { ?patient acgt:hasIdentifier ?patientID . ?patient a acgt:HumanBeing . ?
      patientID a xsd:string . ?patient2 acgt:hasBirthDate ?birthDate . ?patient2 a
      acgt:HumanBeing . ?birthDate a xsd:date . ?registration acgt:hasPatient ?
      patient3 . ?registration a acgt:Registration . ?patient3 a acgt:HumanBeing . ?
      registration2 acgt:hasProcessEnd ?endOfReg . ?endOfReg acgt:hasDate ?regDate . ?
      registration2 a acgt:Registration . ?endOfReg a bfo_span:ProcessBoundary . ?
      regDate a xsd:date . FILTER ( ?patient = ?patient2 ) FILTER ( ?patient = ?patient3 )
      FILTER ( ?registration = ?registration2 ) } LIMIT 5</query>
    <type>CSV</type>
    <SemanticMediatorOutput name="myResults" />
  </SemanticMediatorWIP>
</perform>
```

Figure 2: Perform document of the first service

When the value of the argument type is “DMS”, two more arguments must be provided. In this case the result of the query will be uploaded to the DMS server. The two additional arguments allow specifying where in the DMS the results will be uploaded:

- dir: the directory where the file of results will be uploaded
- file: the name of the file of results

This approach is used by the ACGT Workflow enactor, as the results are used as input of other services.

MappingList

As it was mentioned, this services allows a user to retrieve the complete contain of all the mapping files contained by the Semantic Mediator. This service requires only one argument named MappingListOutput, which indicates the name to which the results will be associated.

updateMappingList

This service allows updating the existing mappings or adding a new mapping file in the integration platform. This process must be carried out in order to reflect any updates in the schemas of the integrated data sources, or to include new databases. Again, only one argument, named updateMappingListInput, is required. This argument must be the complete contain of a mapping file (following the mapping format). In case the given mapping corresponds to an already existing mapping (when the identifier of the mapping and the URL of the associated schema match up), the Semantic Mediator will take care of substituting the old version with the newer version. If the given mapping did not previously exist, it will be added. Figure 3 gives an example of a new mapping being added to the Semantic Mediator.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <perform xmlns="http://ogsadai.org.uk/namespaces/2005/10/types">
    <!-- create session to allow notification of asynchronous delivery completion -->
    <session timeout="300000" />
  - <updateMappingList name="Op">
      <updateMappingListInput><!-- acgt: http://www.ifomis.org/acgt/1.0 --> <!-- bfop1:
        http://www.ifomis.org/bfo/1.1/snap --> <!-- bfop2: http://www.ifomis.org/bfo/1.1/span --> <!-- ro:
        http://www.ifomis.org/obo/ro/1.0 --> <!-- owl: http://www.w3.org/2002/07/owl --> <!-- xsd:
        http://www.w3.org/2001/XMLSchema --> <!-- top_vocab:
        http://gridnode.ehv.campus.philips.com/temp/TOP_CRF_Norm/1.0/vocab --> <mapping> <dbinfo>
        <dbid>TOPdb</dbid>
        <wrapperurl>http://paracelso.dia.fi.upm.es:8080/wsrf/services/ogsadai/DataService</wrapperurl>
        <description>TOP Clinical Trial database</description> </dbinfo> <!-- QUERY 1 --> <map> <!--
        HumanBeing hasIdentifier string --> <!-- patient patient_PT Thing --> <entrydescription> HumanBeings
        with their identifiers </entrydescription> <path_list> <src_paths> <path> <int_entity
        composingid="compID01"> http://www.ifomis.org/acgt/1.0#HumanBeing </int_entity> <rest>
        <int_link> http://www.ifomis.org/acgt/1.0#hasIdentifier </int_link> <int_entity
        composingid="compID02"> http://www.w3.org/2001/XMLSchema#string </int_entity> </rest>
        </path> </src_paths> <target_paths> <path> <int_entity composingid="compID01">
        jdbc:mysql://localhost:3306/top#aeae </int_entity> <rest> <int_link>
        jdbc:mysql://localhost:3306/top#aeae_pt </int_link> <int_entity composingid="compID02">
        http://www.w3.org/2002/07/owl#Thing </int_entity> </rest> </path> </target_paths> </path_list>
        </map> </mapping></updateMappingListInput>
    </updateMappingList>
  </perform>
```

Figure 3: Perform document for adding a new mapping

It is worth mentioning that, although deployed as a public OGSA-DAI service, all three services require specific certificates to be successfully invoked. This way, only authorized users can make use of the Semantic Mediator.

2.2. The mapping tool

The Mapping Tool plays an important role within the ACGT Semantic Mediation layer, as it offers the possibility of including new data sources in the integration platform. This process involves establishing relations between elements of the schema of the source to be integrated and elements of the ACGT Master Ontology—which acts as global schema for the mediator. These relations are called mappings, and the process of establishing mappings is called mapping process. The mappings are used by the mediator in the task of translating integrated queries into queries for the underlying databases.

The mapping process often requires some degree of expertise on both the domain of the data being mapped and the inner technical characteristics of the mappings being produced. To this end, the Mapping Tool incorporates a series of features aimed at facilitating this task and reducing as much as possible the user's workload. These are listed below:

- Web-based friendly user interface: the interface offers an intuitive approach to all available actions. Access is offered through Internet, avoiding installation requirements at the user terminal
- Extensive and powerful mapping editor: a complete editor allows creating and editing mappings with all necessary detail
- Collaborative environment: the environment offers the possibility of two or more users to concurrently collaborate on the same project

Next paragraphs describe the functioning of this tool. The complete mapping process is depicted and viewed through screenshots.

Login window

The first step when accessing the Mapping Tool is logging into the system. This process ensures that only authorized personnel has access to this tool. A username and a password must be provided, as shown in figure 4.

Figure 4: login screen

The first time a user pretends to get access into the tool, a new account must be created. Some personal data together with a username and password are required. At this stage of development, any person can create an account. Future enhancements will be included in order to restrict this process to individuals associated to ACGT. Figure 5 shows the account creation window.

Figure 5. Account creation

Administration window

After a user has logged into the system, a mappings administration screen is presented to him. In this screen a user can see all the active mappings he is either administrator or user of. There are two more windows that contain messages received to the account of the user. The first one lists the requests from other users for joining mappings the user is administrator of. The second one displays the received messages. From this screen the user can perform two actions: create a new mapping or join an existing mapping. Figure 6 shows the administration window.

Figure 6. Mappings administration screen

If the user clicks on the New Mapping button, a small form will appear on the same screen. This form must be filled in with information regarding the name the user prefers for the new mapping and details about the wrapper to be used for this new mapping—i.e. wrapper URL, database identifier and RDF schema URL. After providing all the necessary information, the new mapping will appear in the screen containing mappings where the user is administrator. Figure 7 depicts the creation of a new mapping.

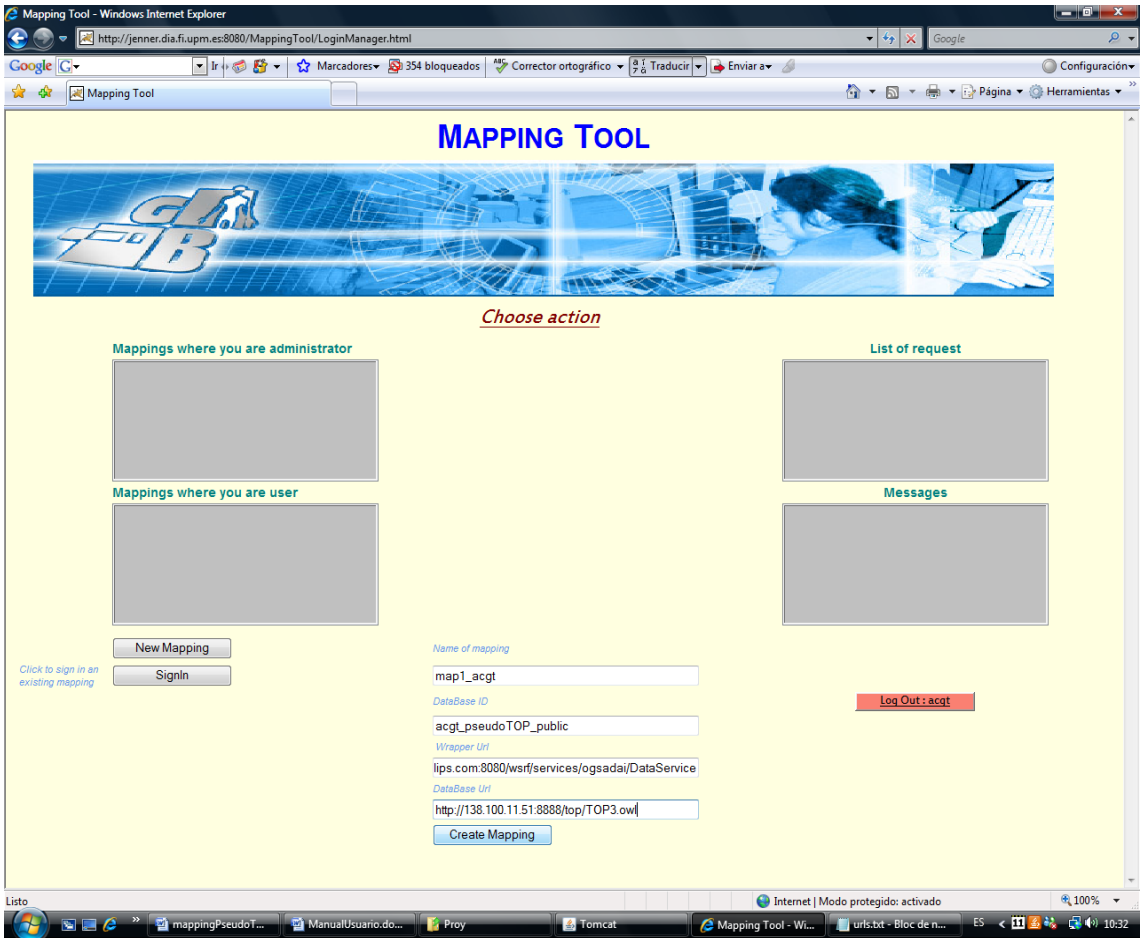

Figure 7. Creating a new mapping

Once a mapping has been created and is listed in the mappings where you are administrator window, the user can start editing it. Clicking on the name of the mapping will make several new actions available, as illustrated in figure 8.

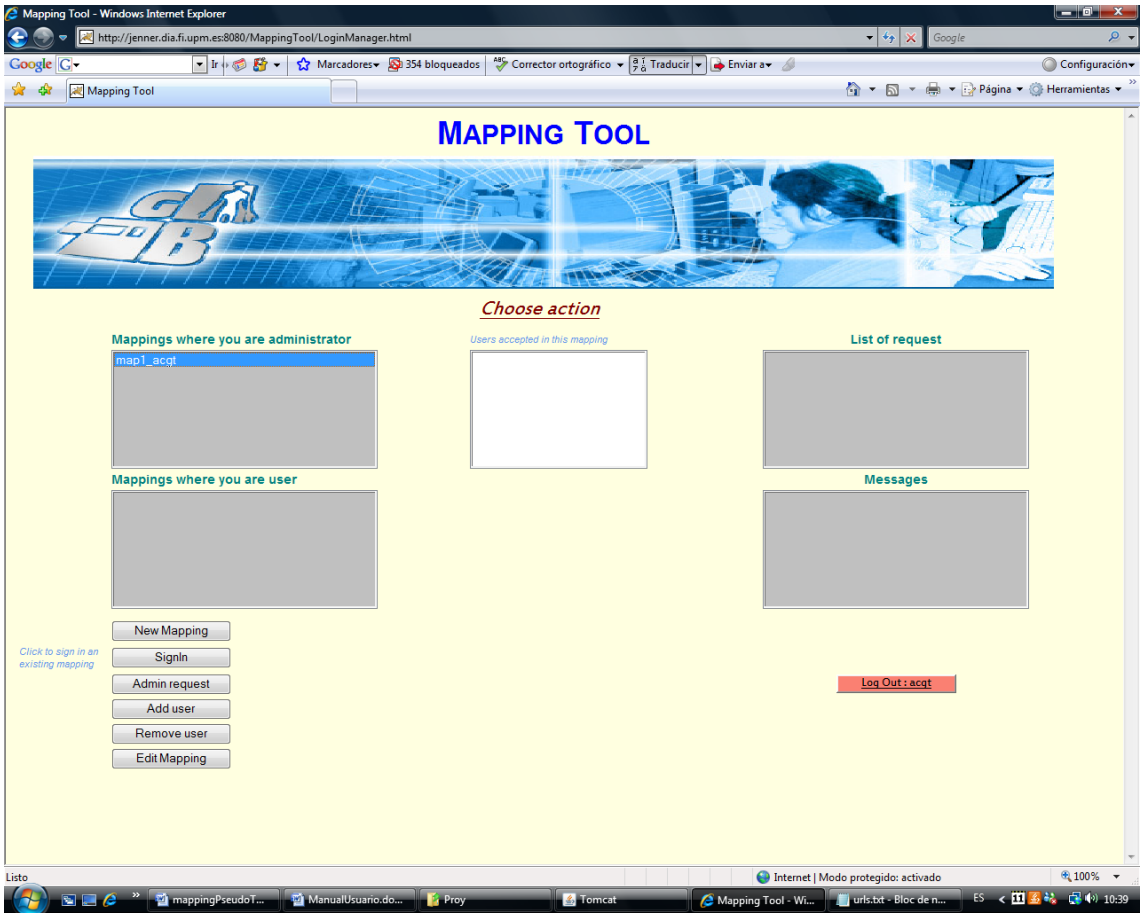

Figure 8. Available actions over an existing mapping

The button “Admin request” allows the user to manage any incoming requests to join this mapping from other users. The buttons “Add user” and “Remove user” allows managing privileges of other users with respect to this mapping. Finally, the button “Edit mapping” allows to build the mapping itself. Selecting this last option will guide us to the mapping manager window.

Mapping Manager window

The mapping manager window lets a user edit an existing mapping by establishing the required relations between elements of the ACGT Master Ontology and elements of the physical database RDF schema. This window is composed of a series of elements necessary for the mapping construction process. The elements under the “Conceptual Area” tag are related to the Master Ontology, while the elements under the “Physical Area” tag relate to the physical schema being mapped. Figure 9 shows the main mapping manager window.

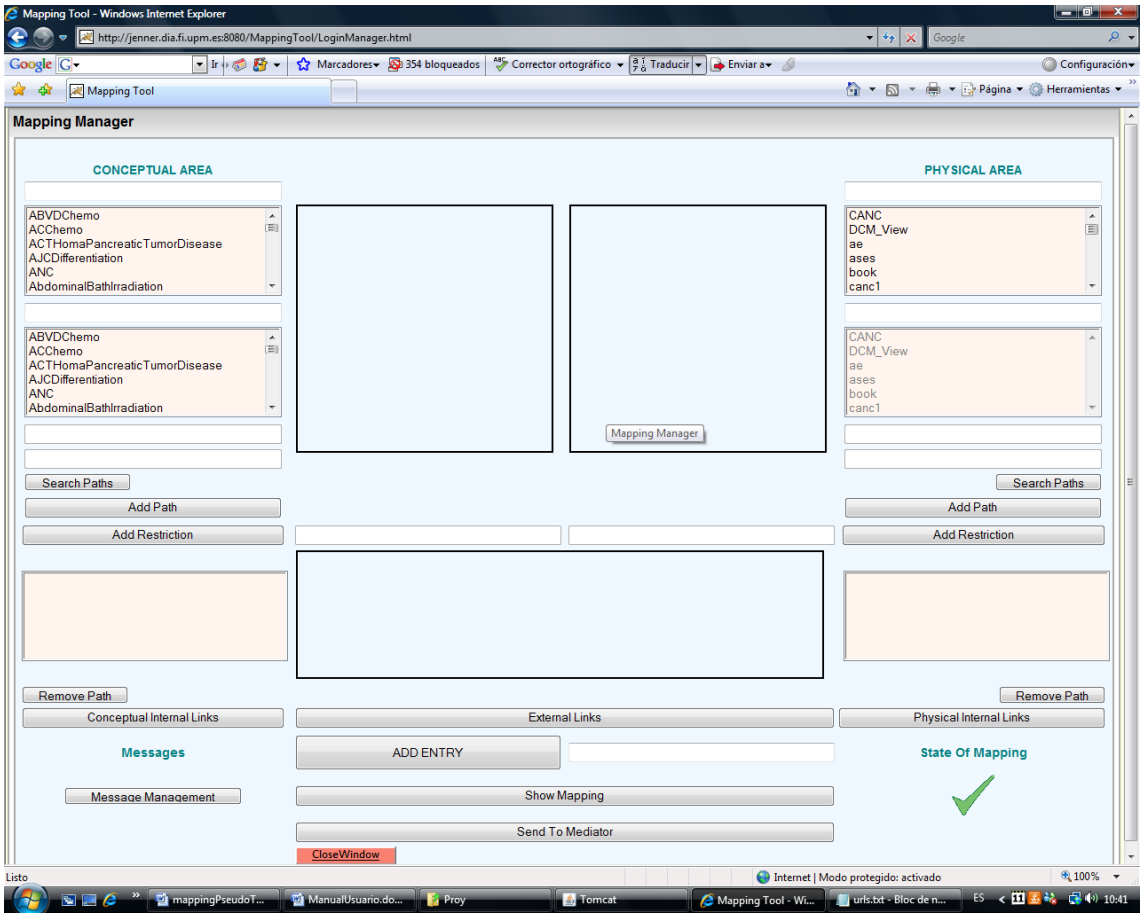

Figure 9. Mapping manager window

Mappings are composed of one or more entries, and each entry is composed of one or more conceptual views and their semantically equivalent physical views. A view is defined as a collection of one or more paths in a schema (for details on the mapping format, please refer to deliverable D7.3). The conceptual area elements and the physical area elements offer the possibility of building entries for the current mapping.

To create an entry, a user must select a class as a starting point for a path. This can be done by selecting a class from the class list (both for conceptual and physical), or by writing part of its name in the textbox right above the class list. After a class has been selected, the class tree window next to the class list will display the relations that have the selected class as domain. Figure 10 shows the creation of a conceptual path.

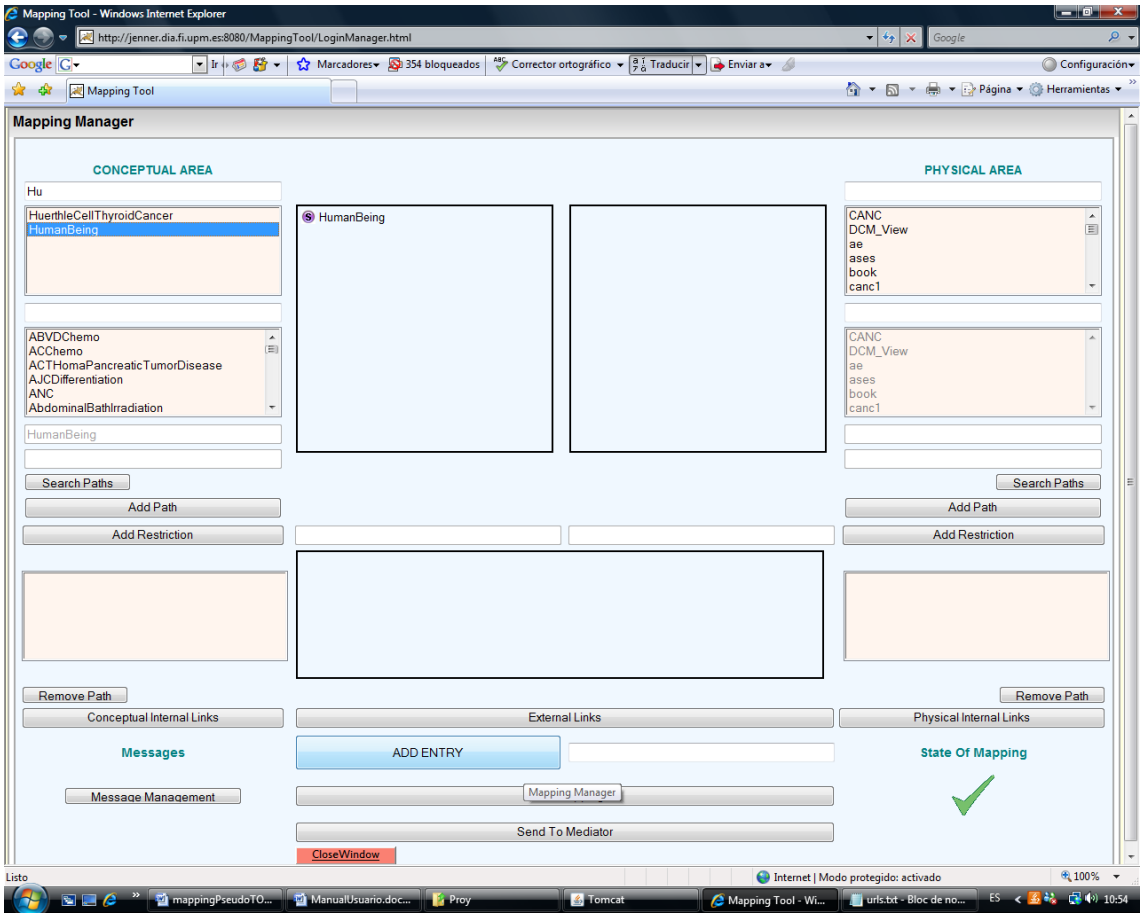

Figure 10. The Conceptual area and the Physical area contain the necessary elements for building conceptual paths and physical paths

This view can be expanded by clicking on an item. Relations will be expanded with their range classes, and these will be expanded with relations that contain them as domain. This iterative process allows creating paths, as can be seen in figure 11.

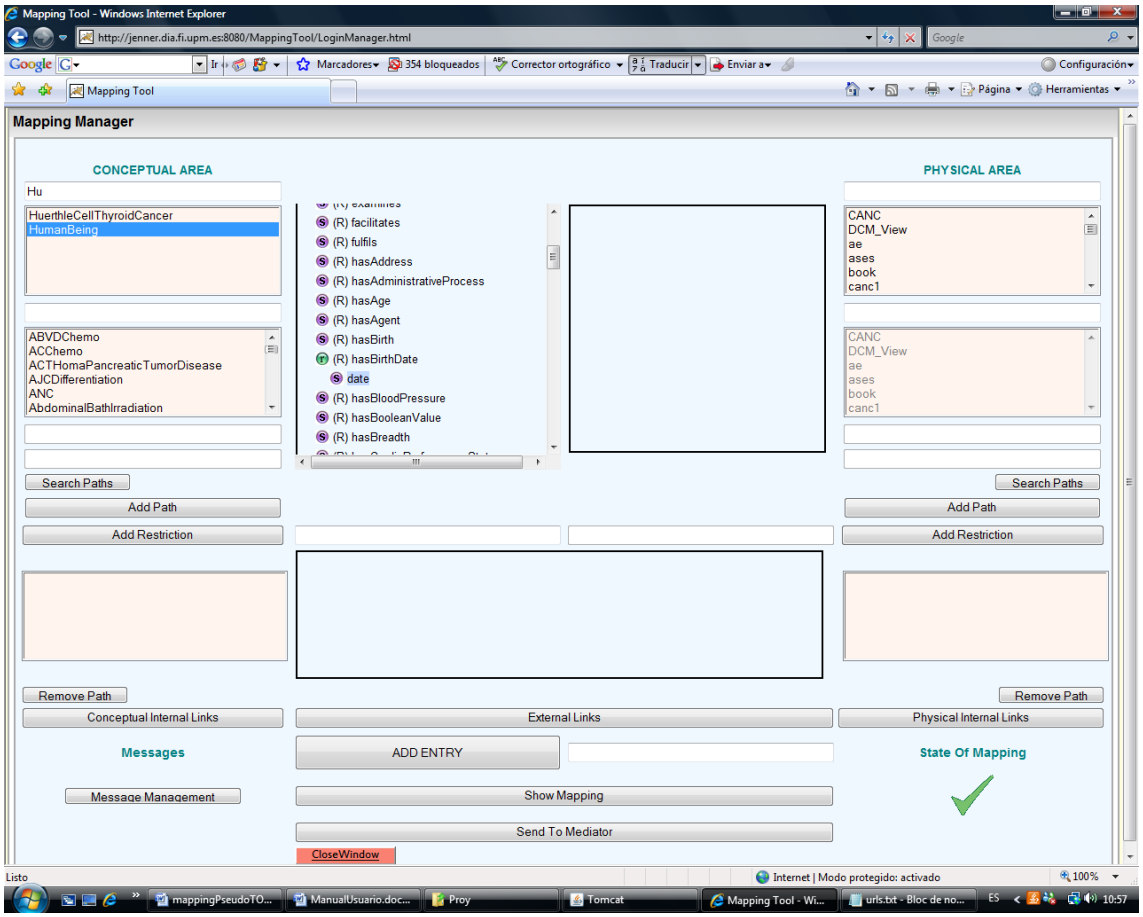

Figure 11. Path creation process

After the path has been built, it can be added to the path list for the current entry by clicking the button “Add Path”. The path will be displayed in the path list. Following this method, paths for the Master Ontology (conceptual paths) and paths for the physical schema (physical paths) can be built and added to the current entry, as shown in figure 12.

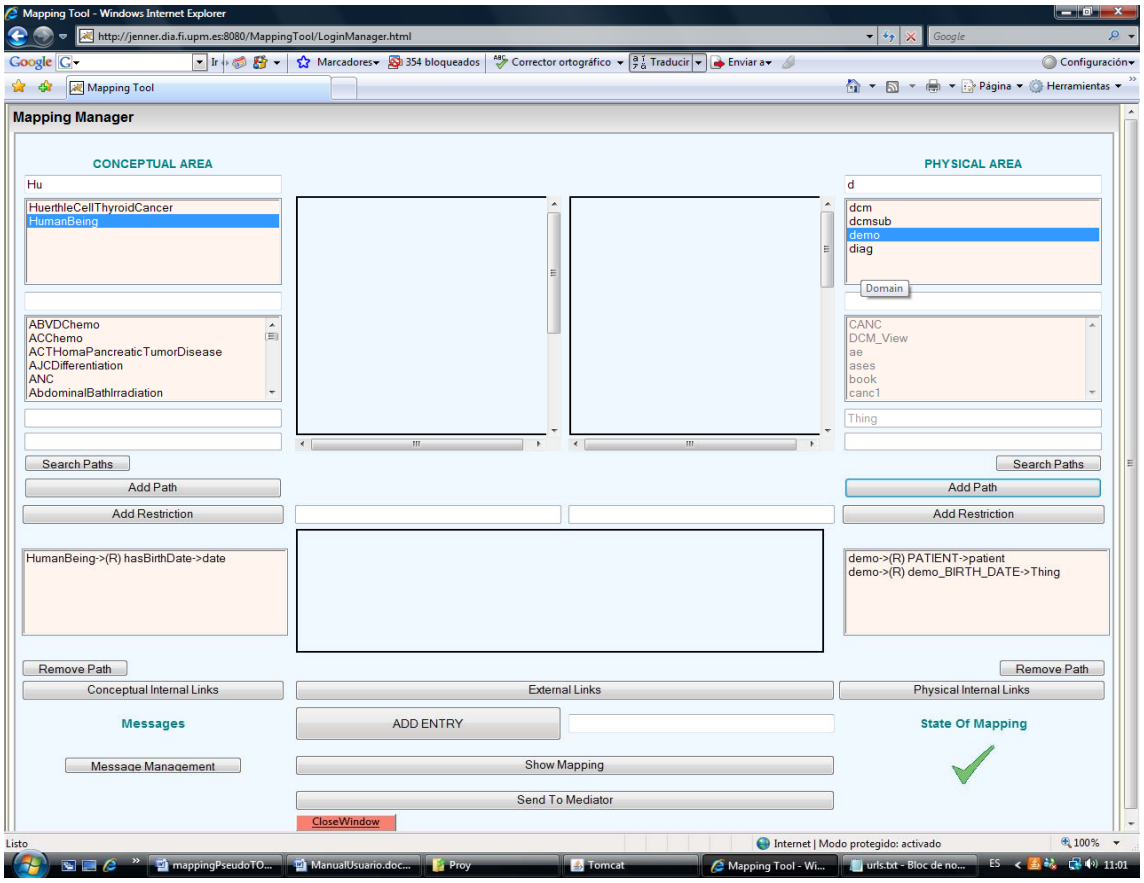

Figure 12. Adding paths into the current entry

Apart from a list of conceptual paths and physical paths, an entry requires information about the links among classes of those paths. There are two types of links: internal and external. Internal links establish equality relations among classes of different conceptual paths or among classes of different physical paths. External links are used to establish semantic equivalence of conceptual classes with physical classes. The buttons “Conceptual internal links” and “Physical internal links” allow creating internal links. When one of these buttons is clicked, the “Bounds Manager” window will pop up. In this window all the conceptual paths or all the physical paths (depending on the button clicked) will be displayed twice. This way, the user will be able to select two conceptual paths or two physical paths. The complete description of the selected paths will be displayed, and the user will have to select which classes are internally linked. Figure 13 shows the creation of internal links for an entry.

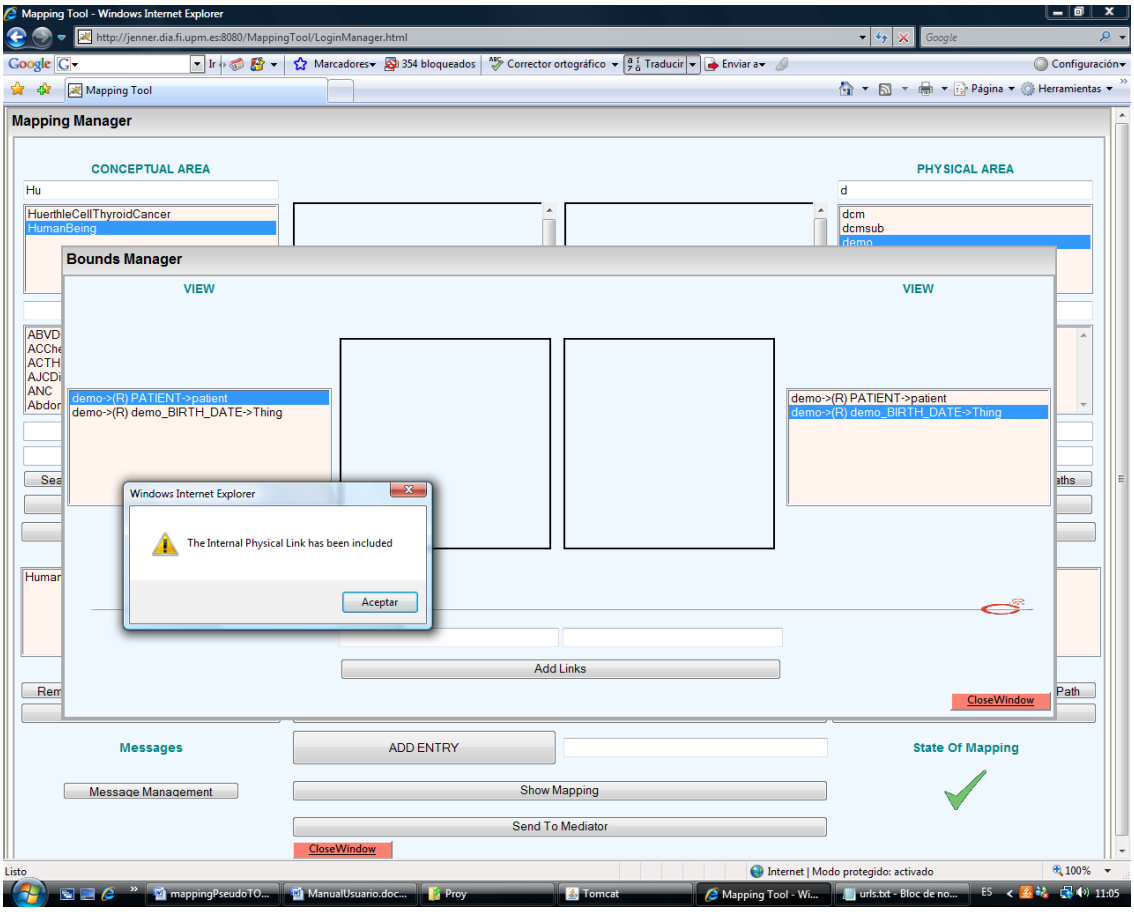

Figure 13. Establishing an internal link between physical paths

In order to create external links, the button “External links” must be clicked. This produces another “Bounds Manager” window to pop up. This time, the left area will show conceptual paths, and the right area physical paths. By selecting one path of each area the user will be able to select which conceptual class gets bounded with which physical class. Figure 14 illustrates this.

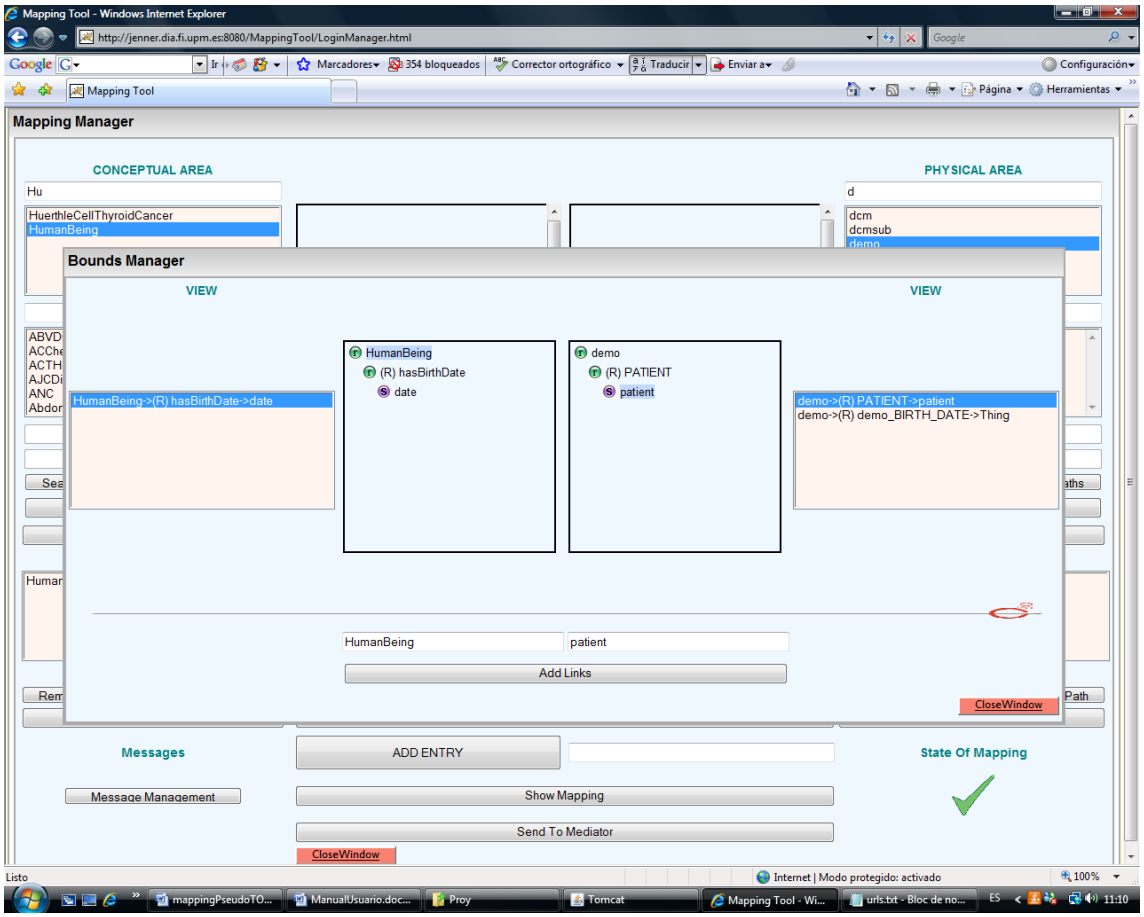

Figure 14. Establishing an external link

After all internal and external links have been created the user must proceed to add the entry into the mapping. To do this, he must simply click on the “Add entry” button. All the fields of the current entry will be reset, and the added entry will be displayed on the entries list. The user will be able to examine the state of the complete mapping by clicking on the “Show mapping” button, as seen in figure 15.

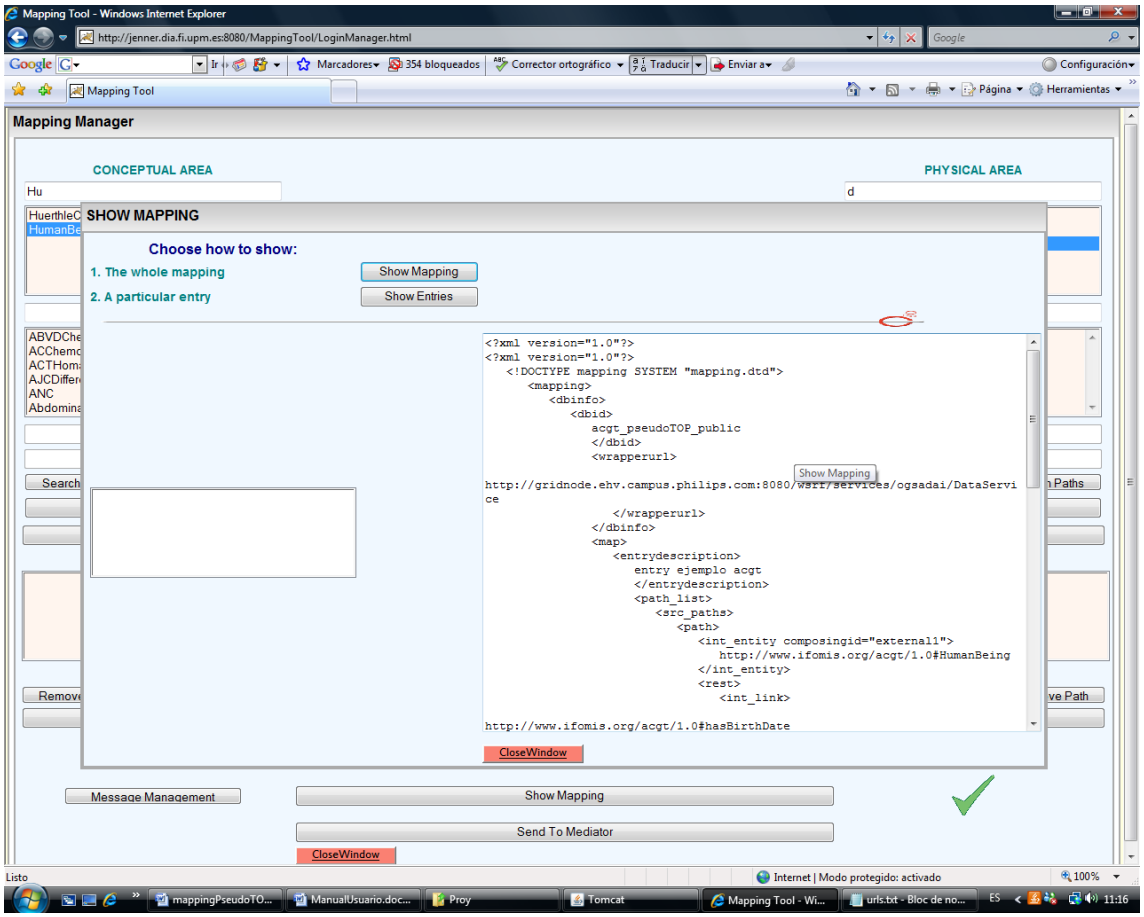

Figure 15. Details of the created mapping can be reviewed in the Show Mapping window

After the user has completed a mapping, he can submit it to the Semantic Mediator. To do this, he only has to click on the “Send to Mediator” button. From this moment, the mediator will be able to integrate data from this source with previously existing repositories, as incoming queries will be translated to the new source as well.

Collaborative approach for the Mapping process

The mapping process involves establishing semantic relations from terms in a schemata with terms contained in the Master Ontology. It therefore implicates that the user must have some degree of expertise on the data being mapped to the Master Ontology. Schema terminology is not always self explicative, and thus the meaning of a specific class might not be clear to a regular user. Furthermore, the Master Ontology, although thoroughly annotated, contains more than one thousand terms, and thus some knowledge about its structure is highly recommended. Even the way elements are mapped together requires some training on the details of the mapping format. Derived from all these circumstances, a user creating a mapping for the Semantic Mediator should have expertise on the three mentioned areas. In order to relax this requirement, the Mapping Tool incorporates features that allow a mapping to be built concurrently by more than one user, thus offering the possibility of adopting a collaborative approach. The basic idea is to let two or more users to be actively working on the same mapping at the same time, each adding his expertise to the process. To do this, the administrator of a mapping must first add more users to a project. There are two ways to achieve this. The administrator can invite other users to collaborate with a mapping, or a user can directly ask an administrator for access privileges on an existing mapping. Figure 16 shows a

user examining the available mappings to request joining.

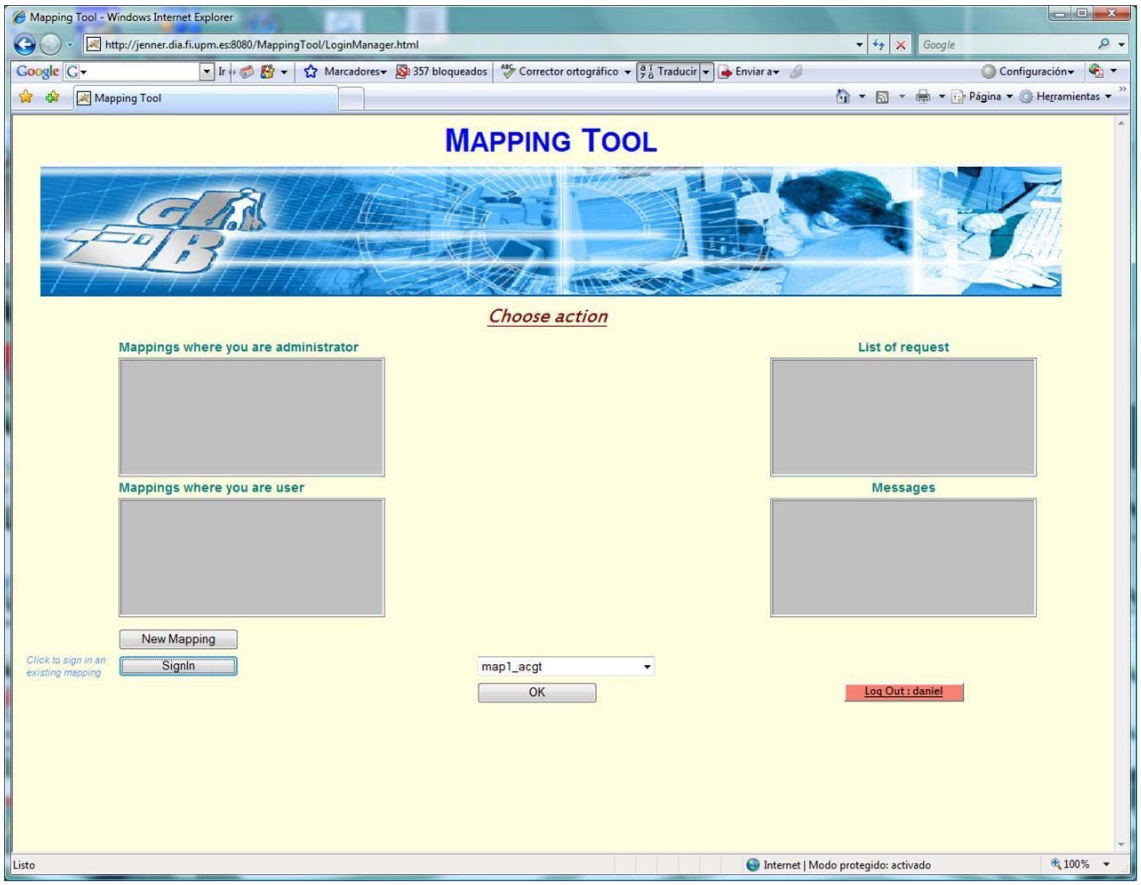

Figure 16. Available mappings for joining

A user named “Daniel” clicks on the “Sign In” button. A list of mappings where he could send collaboration petitions is displayed, and one of the listed mappings is selected. This sends a petition to the administrator of this mapping, which will be displayed in the “List of Requests” element—for each petition, the name of the user who created the petition and the name of the mapping he is asking access to is displayed. This is shown in figure 17.

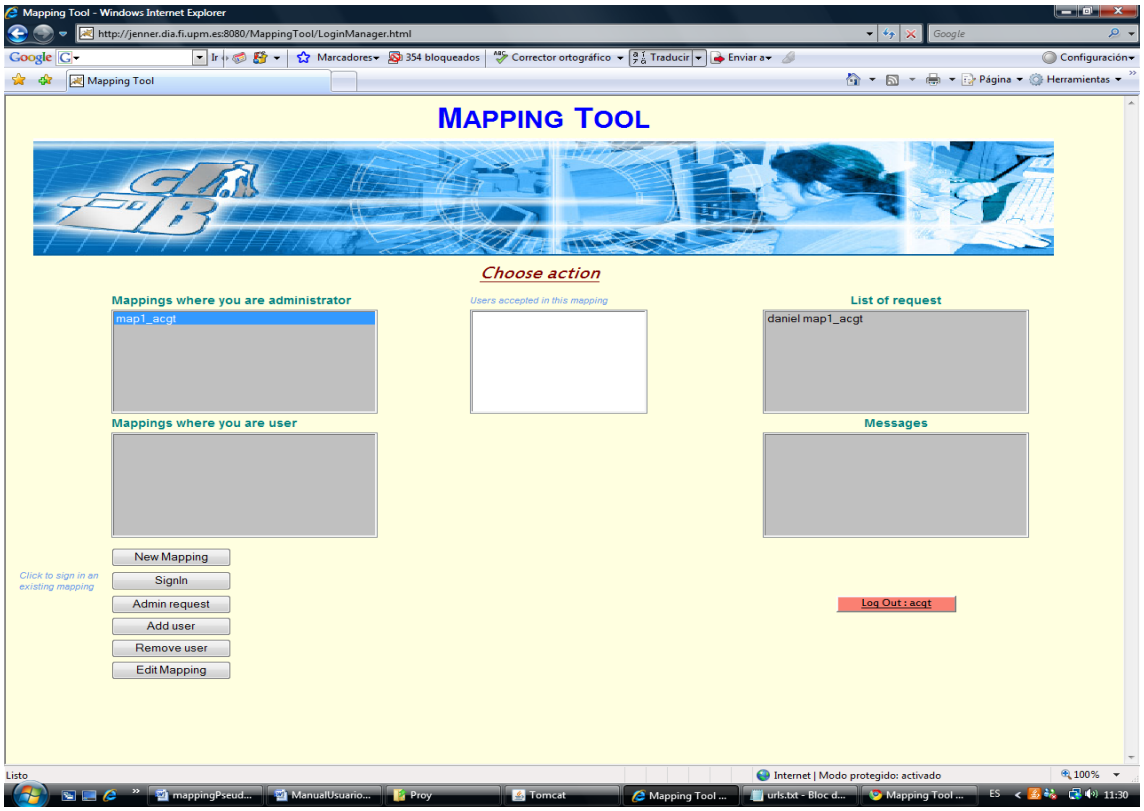

Figure 17. The administrator of a mapping can see the user that have requested access to his mappings.

The administrator of a mapping has the capacity of either accepting or rejecting requests from other users. To do this, he selects a mapping and clicks on “Admin request”. This produces a list of users requesting access for that mapping to appear. The administrator can select any user and submit the preferred action for him, as shown in figure 18.

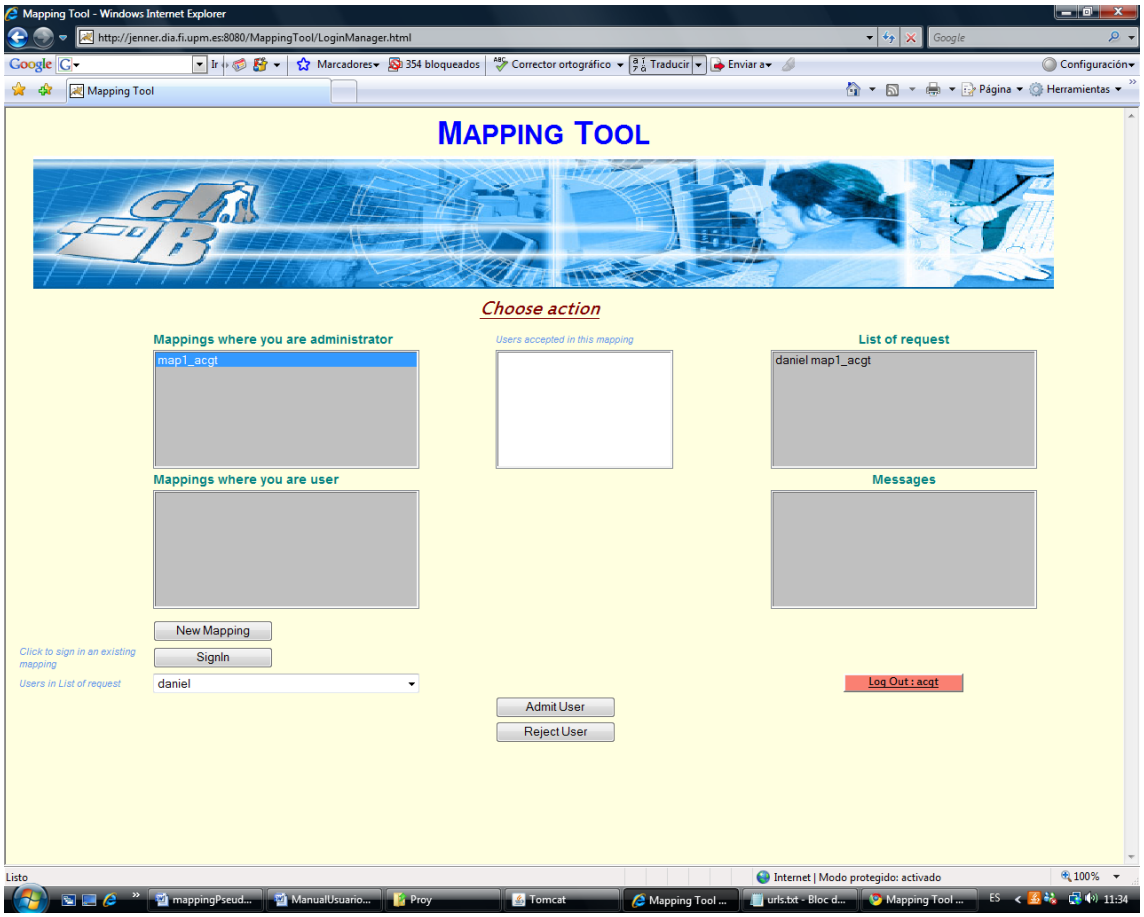

Figure 18. Administration of user's requests

Once a user's request to join a mapping has been accepted, this will appear in the mappings where he is user. From this moment, he will be able to edit it, as shown in previous screens.

2.2.1. The query builder

The query builder is a web based tool with the aim of aiding non expert users in the process of formulating queries for integrated repositories of databases. The query language used by the mediator is SPARQL [SPARQL], a RDF query language with an intermediate level of expressiveness. A query in SPARQL can be divided in three main sections:

- An expression identifying the information to be retrieved. It is comprised by a set of variables matching with classes in the model.
- A set of views representing the subset of the global schema implied in the retrieval process.
- A set of constraints over the elements in the previous sections.

The formulation of a query in SPARQL for the mediator requires the user to know not only the query language, but also global schema in the detail. This kind of knowledge involves a strong training in the technical field—i.e. including technologies and standards such as XML, RDF, OWL and SPARQL—, something that definitely cannot be demanded to end users. This query builder has been designed to overcome these issues.

Login screen

The query tool is accessible through a normal internet browser. The user must have a valid account to use the query tool. Once the user is identified, she is granted access to the query building interface. Figure 19 shows the Query Tool login screen.

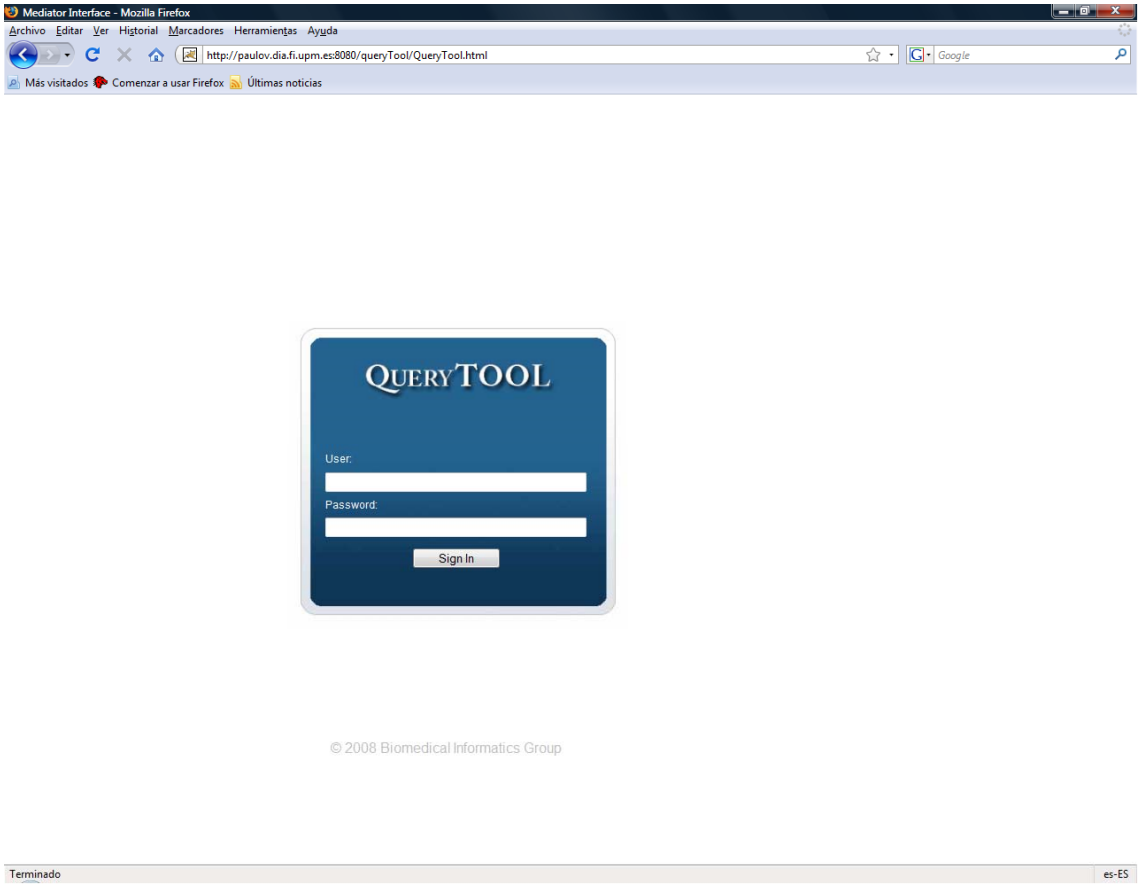

Figure 19: The Query Tool login screen

Query Building Interface

In the Repository section of the query building interface the available repositories are shown to the user. At this level, a user can perform the following tasks with the tool:

1. Change the password.
2. Load a stored query.
3. Load a new repository.
4. Create a query.
5. Exit the system.

This screen is shown in figure 20.

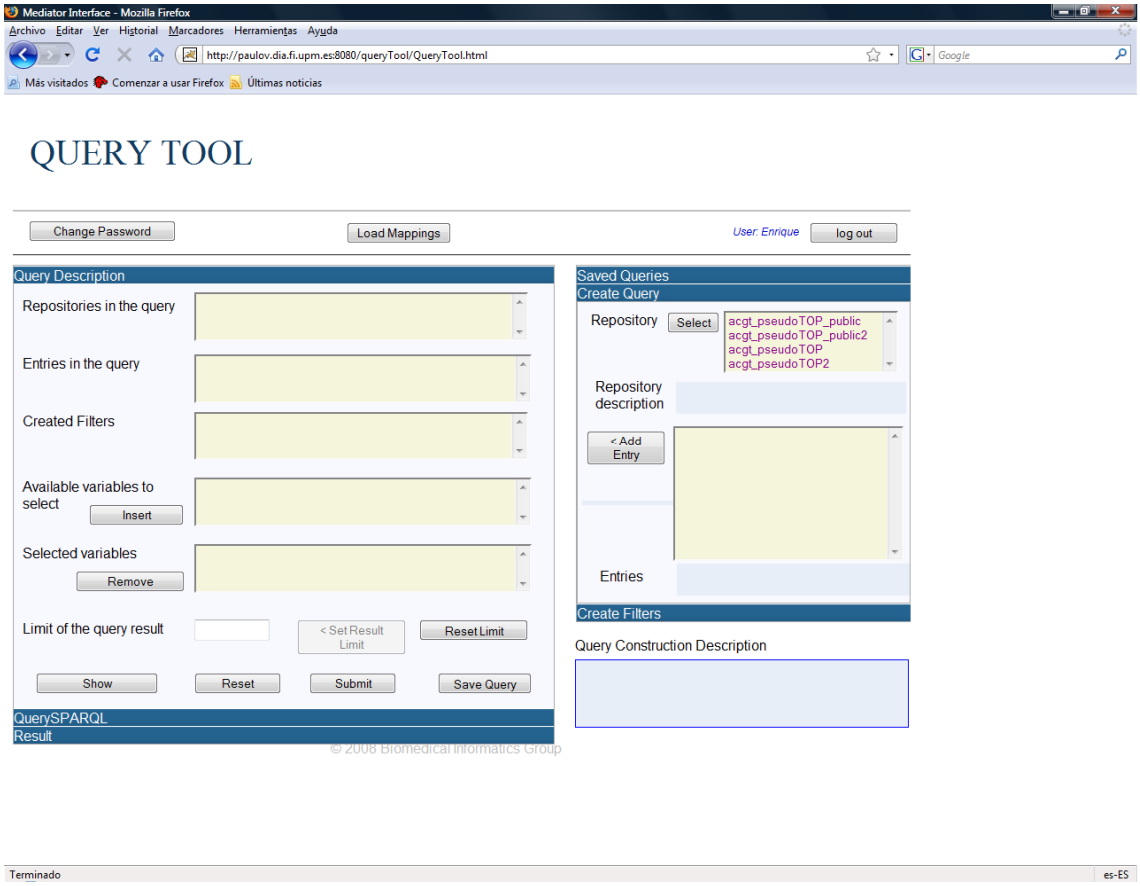

Figure 20: The Query Tool main window

The following paragraphs describe the normal process of creating a query, sending it to the mediator and storing it for further usages.

The first step is selecting the repository among the set available in the Create Query panel. In the panel below the views available in the selected repository are shown. A repository at this level represents a set of integrated databases. These repositories are created by joining sets of mappings, following the requirements of a given user profile, and has a schema that is a subset of the ACGT Master Ontology. Figure 21 illustrates how the available views are displayed when an entry is selected, showing natural language descriptions of the information represented by these subsets of the global schema.

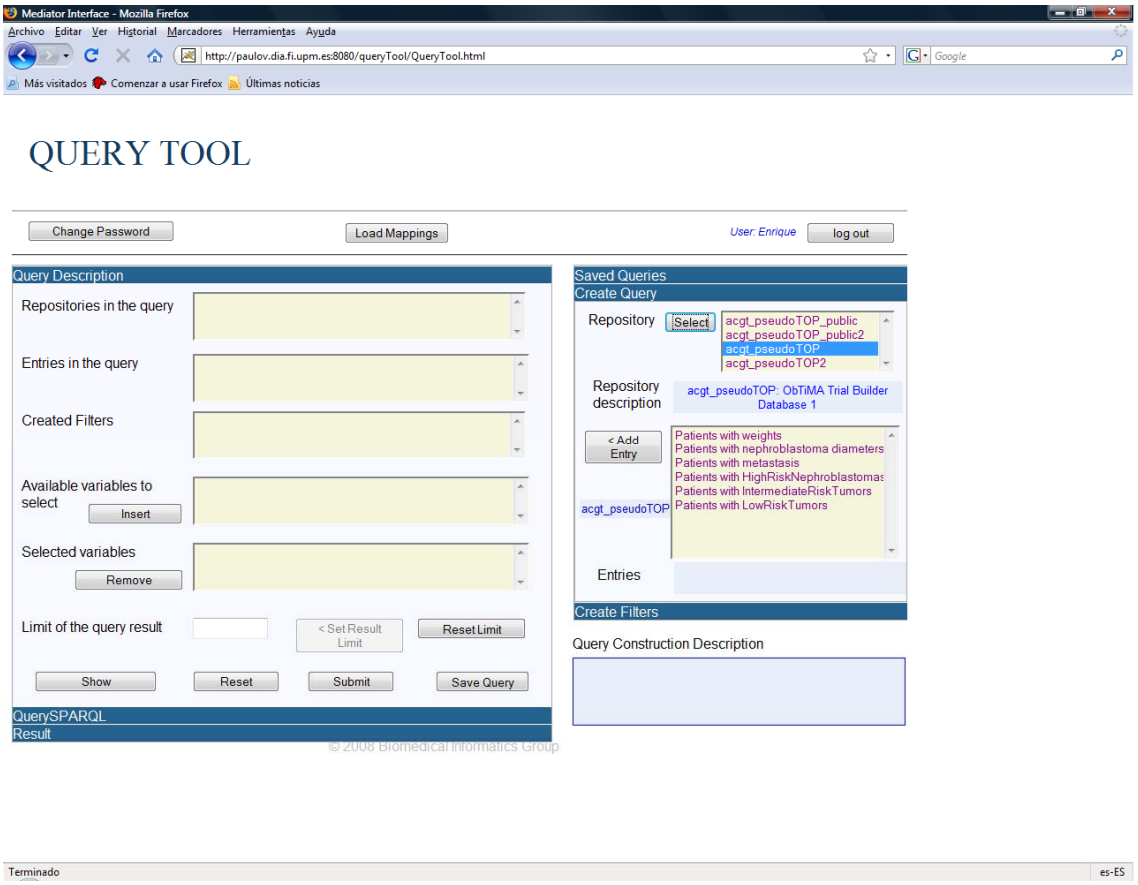

Figure 21: When a repository is selected, its associated views are displayed

Once the natural language description of the views are displayed, the user selects the set of them representing the schema containing the information that she needs to retrieve. When a view is selected, it is shown in the entries panel of the Query Description section. The available variables (the ones that have a real link with information in the databases) are displayed as well, as shown in figure 22.

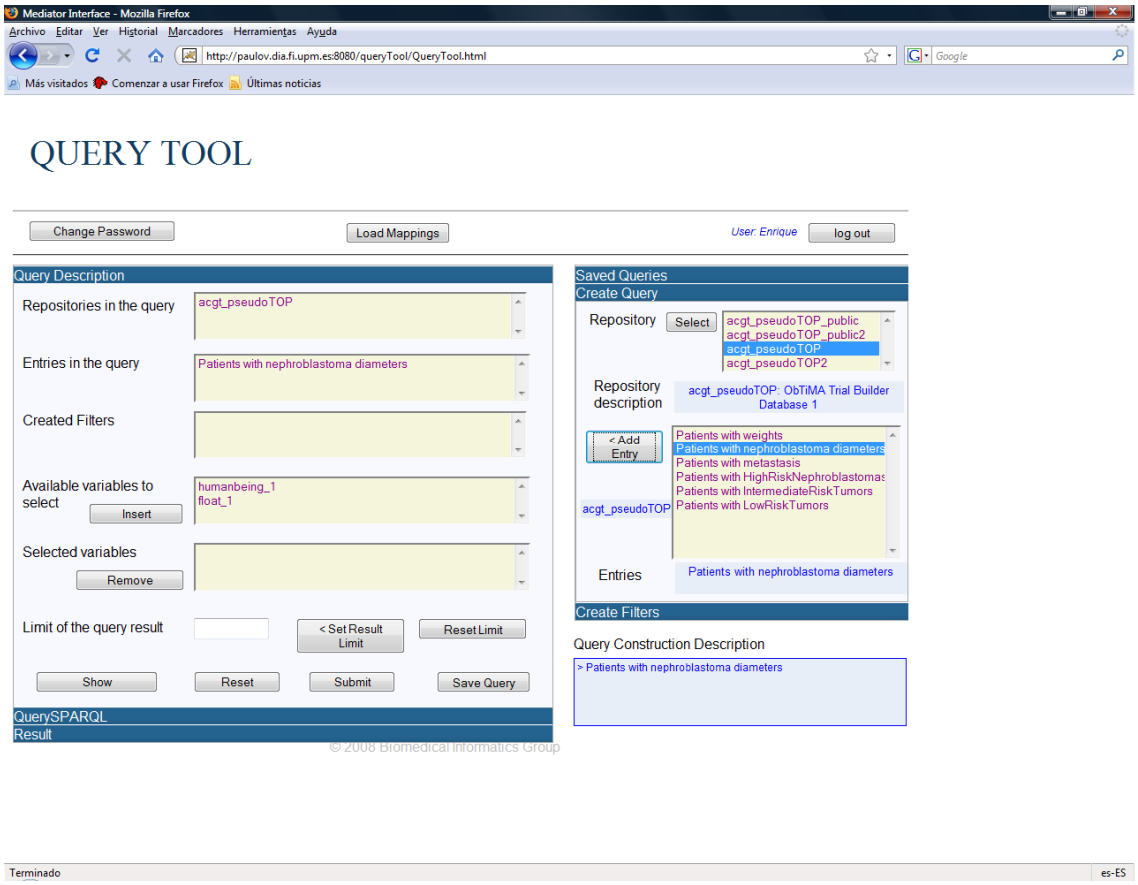

Figure 22: Entries are added to the constructed query

At this point, the user must select the variables she wants to be included in the final query. It is worth saying that these variables comprise the structure of the information that will be retrieved, so this is a crucial point in the construction of the query. When the user selects a variable, it disappear from the “available variables panel” to be included in the “selected variables panel”, as can be seen in figure 23.

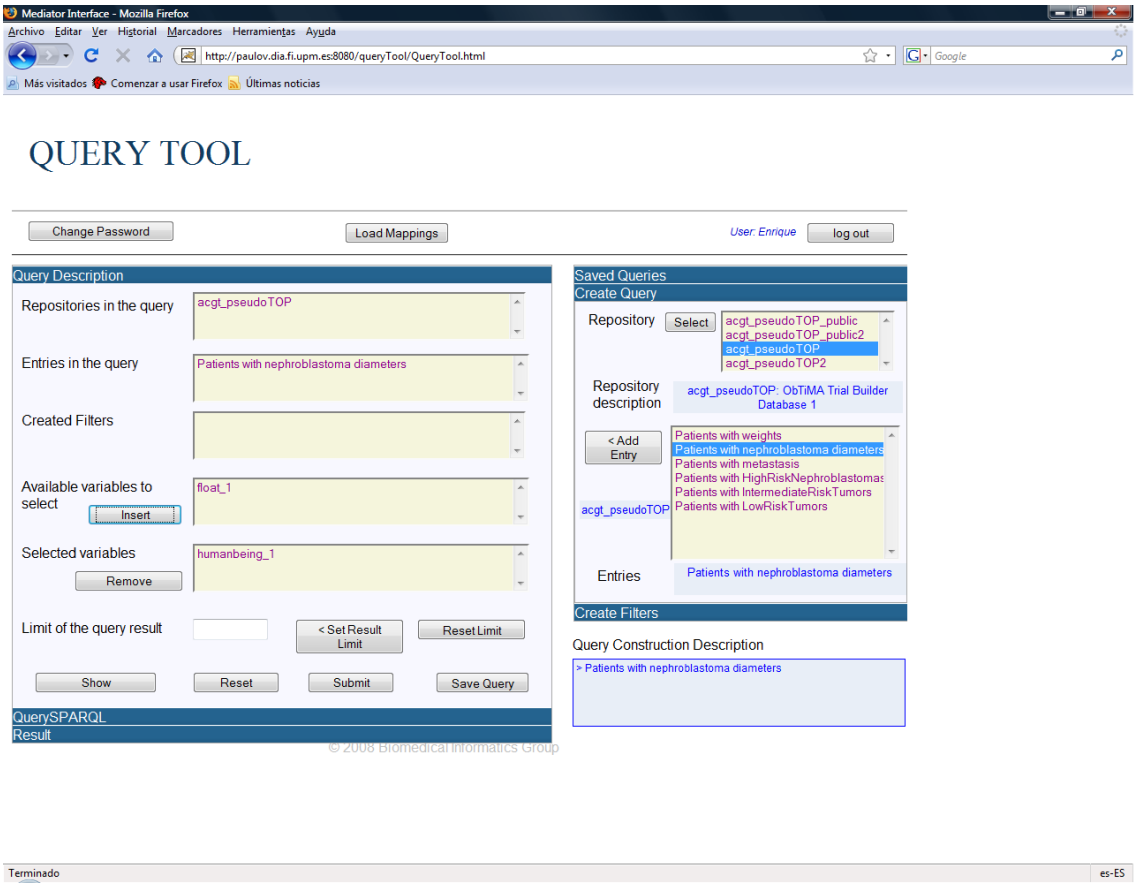

Figure 23: The available variables to retrieve must be selected

Next step is establishing constraints over the retrieved information. To this end, the “Create Filters” includes the option of selecting variables and different operators. It is also possible to include literal values in the filters. A limit for the retrieved values can also be set. All these features can be observed in figure 24.

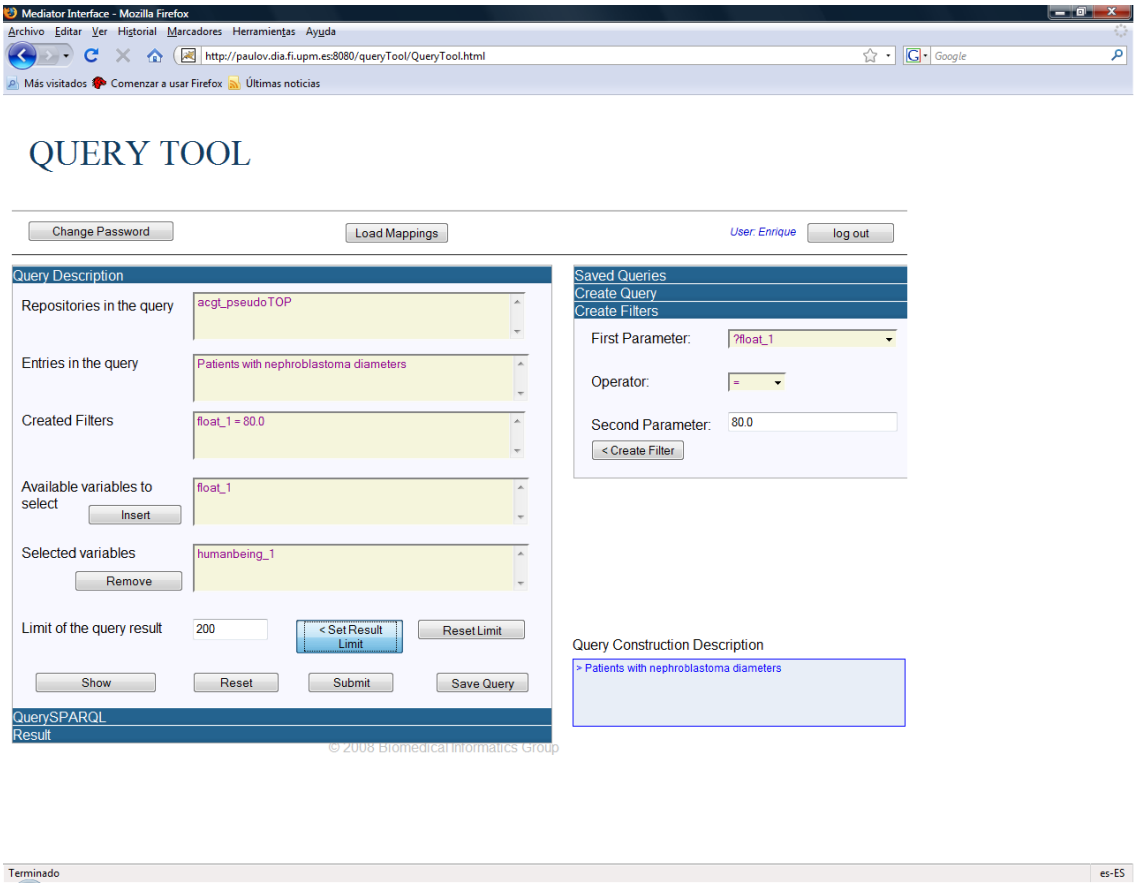

Figure 24: Additional constrains can be added to the query

Although the Query Builder has been designed to hide the complexity of the SPARQL query, there always exists the possibility of consulting the actual code of the query in any moment during the construction process. This functionality will allow expert users to modify the query manually if they need a more personalized configuration. Another advantage is that a query containing only the views can be used as a template in some kind of environments. The way the query code is displayed by the query builder is shown in figure 25.

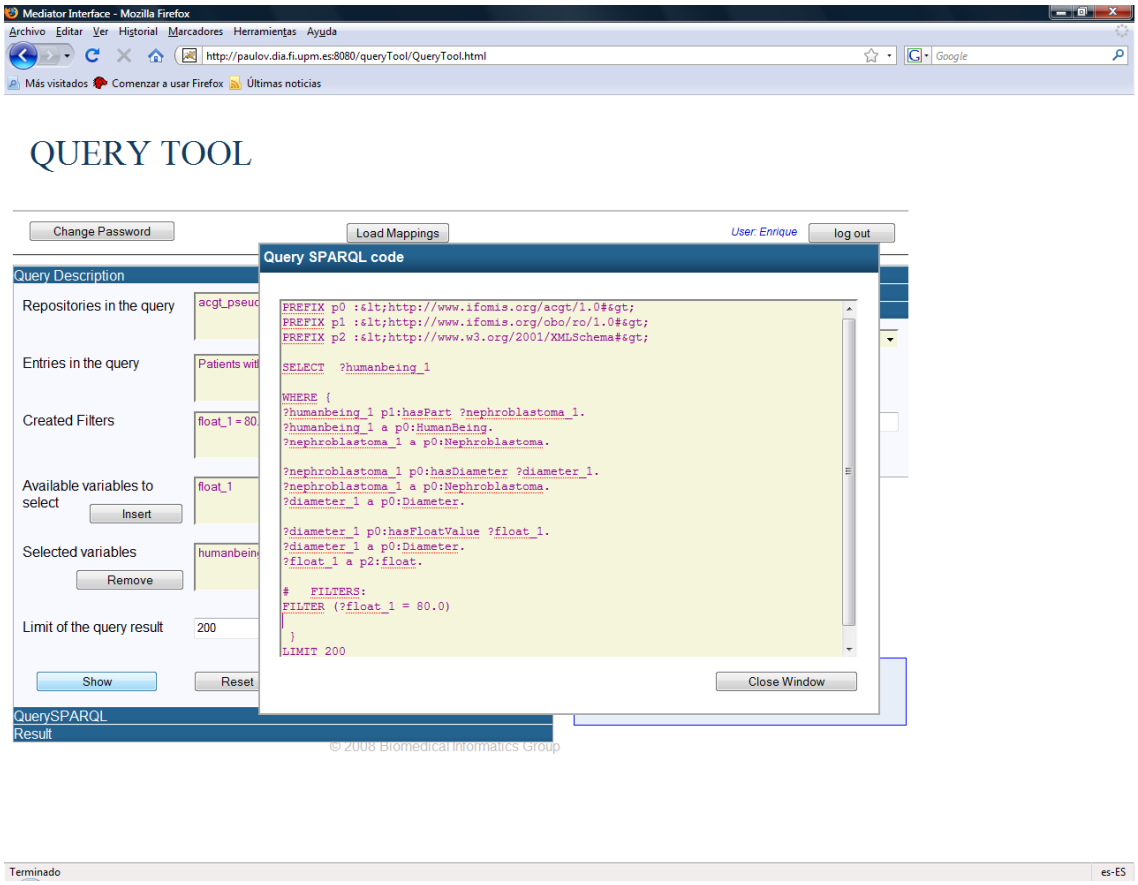

Figure 25: The SPARQL form of the query can be reviewed

The user can also store the query for further usages using the button “Save Query”. A saved query is identified by a natural language description provided by the user. This identifier is supposed to be unique, so duplicate names are not allowed. The user is requested to introduce the clinical trial and a description for it, but these fields are not mandatory. Figure 26 shows this functionality.

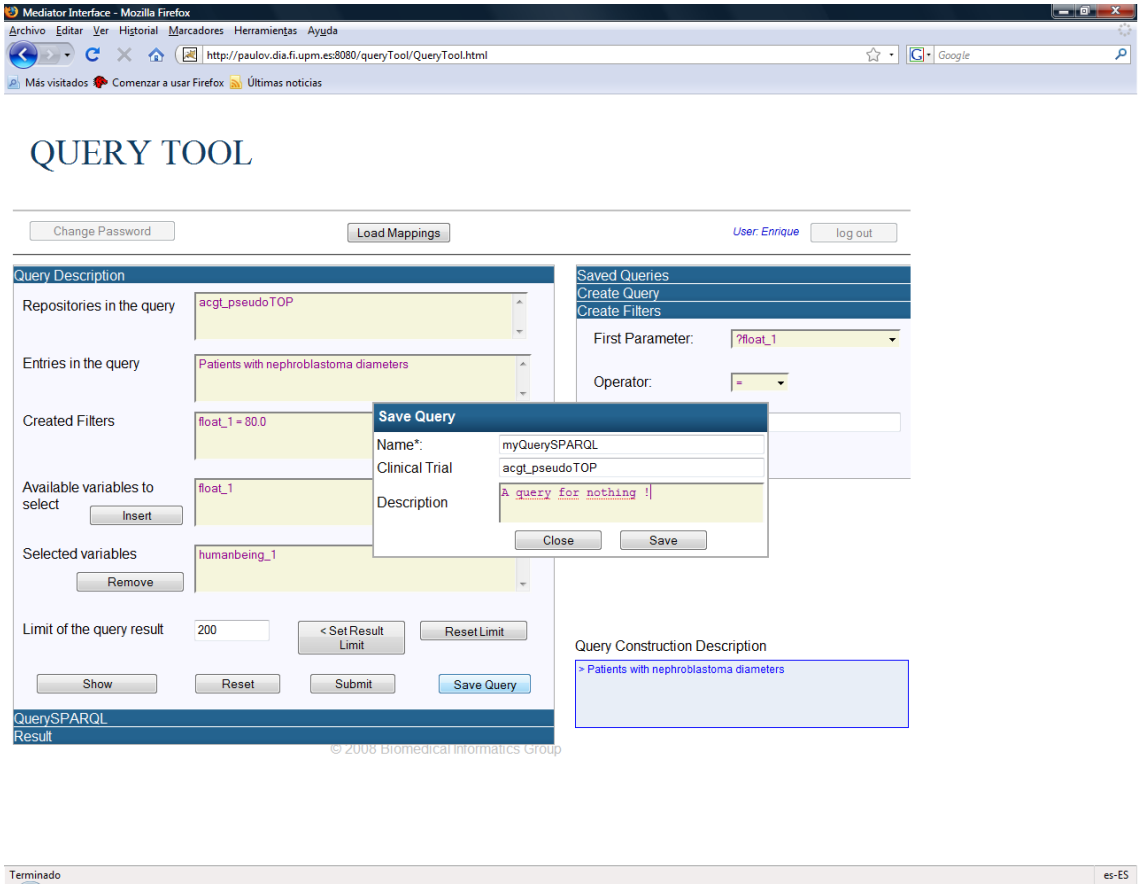

Figure 26: Saving queries for subsequent access

Once the query building process has finished, the user can submit the query. When the query is submitted, it is sent to the mediator and the retrieved results are displayed in the results panel, as shown in figure 27.

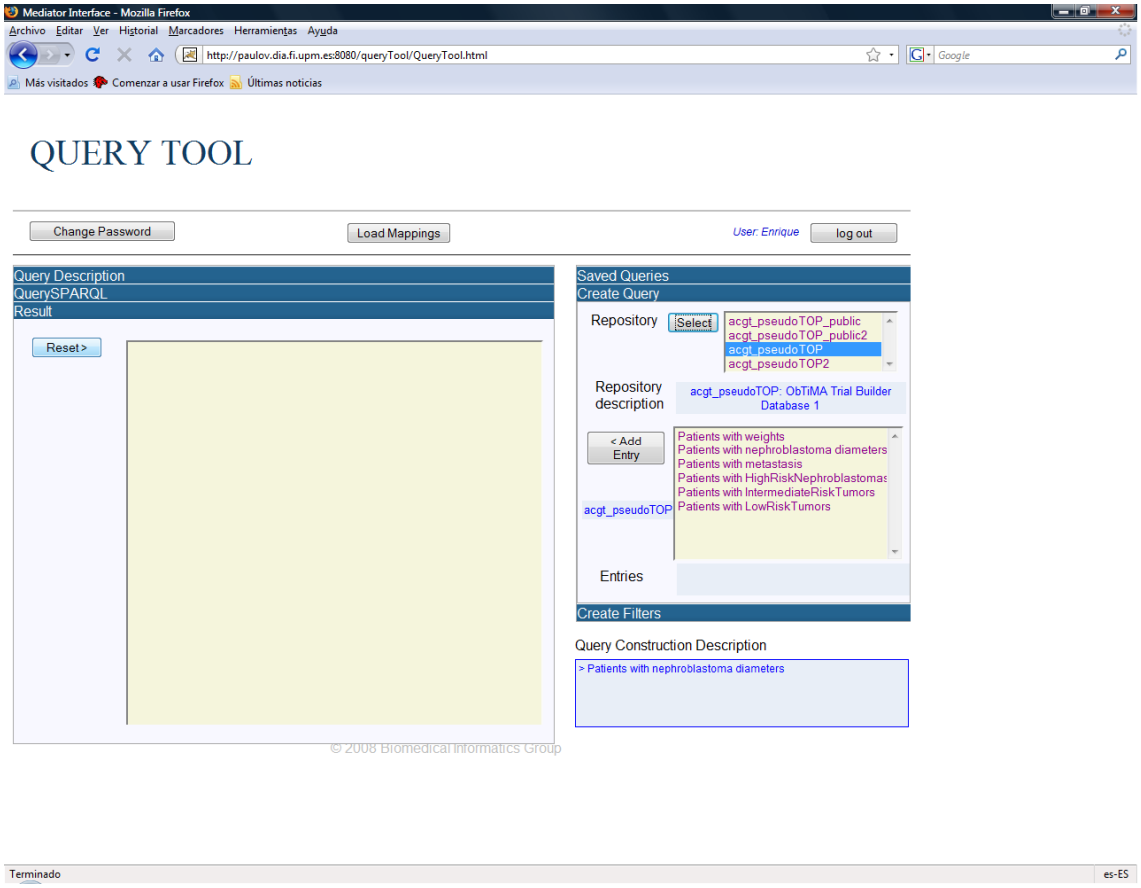

Figure 27: Results of submitted queries are displayed in the Result window

3. Case Study: the Pseudo-TOP scenario

This section describes the process of integrating a new data set in the mediator, and the subsequent query building procedure, using the tools developed to this end. The case study presented is the same that was presented in deliverable D7.4 “Consolidated approach for semantic mediation and integration of heterogeneous data sources for clinical trials”. In D7.4 this case was used to describe the mediation process, illustrating the suitability of the approach by solving different heterogeneity cases.

The selected case involves accessing to the TOP trial database. This relational database contains clinical information related to patients in the clinical trial. The main requirement for integrating a new data source in the system is that the global schema, in this case the ACGT Master Ontology, covers the semantic information needed to establish the proper mappings. However, this is not always possible. When a semantic element found in the new data source is not included in the ontology, the issue must be reported to the ontology engineers maintaining the model. In order to address this issue, an ontology submission system is being developed. This system will facilitate communication for ontology updating, something closely related to the normal mapping process.

In the present document, we first show the process of including the new dataset in the system using the mapping tool. The mapping process, described in section 2.3, presents a variety of special issues in this case study. At the mapping level, it presents a range of issues that the mapping tool is able to cope with—e.g. internal link of paths. The mapping process implies a study of the correspondences between views in the schemas to match, in this case the Master Ontology and the TOP trial schema. Both are represented in RDFS, but this issue is hidden by the mapping tool, facilitating the process of navigating the schemas. Secondly, we show the process of building and submitting a query to retrieve data from the new source. This is done using the query builder. This tool hides the complexity of the query language—i.e. SPARQL—, providing functionalities that permit non-expert users to retrieve information using the mediator.

3.1. Mapping creation for the Psudo-TOP scenario

The first step for the execution of this scenario involves creating a new mapping of the physical pseudo-TOP database with the Master Ontology. The Mapping Tool is used in this process, assisting the user or users implicated with this task.

In this mapping, four entries are created. These entries reflect information regarding patient’s personal data—identifiers and birth dates—and details about the registration of patients in the TOP clinical trial. The mapping will allow the subsequent query of this information through the mediator.

The first entry in the mapping relates the patient’s identifiers in the Master Ontology and in the physical database. In the Master Ontology, a direct path relates the class *HumanBeing* and the datatype *string* through the relation *hasIdentifier*. In the physical database, this information is stored in the class *patient* and *Thing*, related through *patient_PT*. Therefore, an entry that relates those two paths must be created, and two external links—one between *HumanBeing* and *patient* and another between *string* and *Thing*—must be added. Figure 28 illustrates this entry.

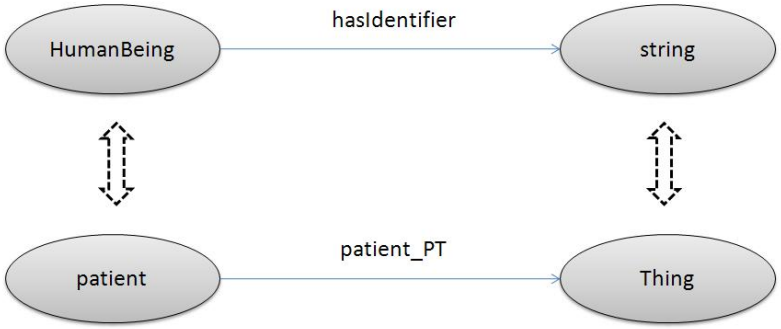

Figure 28: The first entry of the mapping

This entry is created in the Mapping Tool by selecting the appropriate classes and relations. Figure 29 depicts this process.

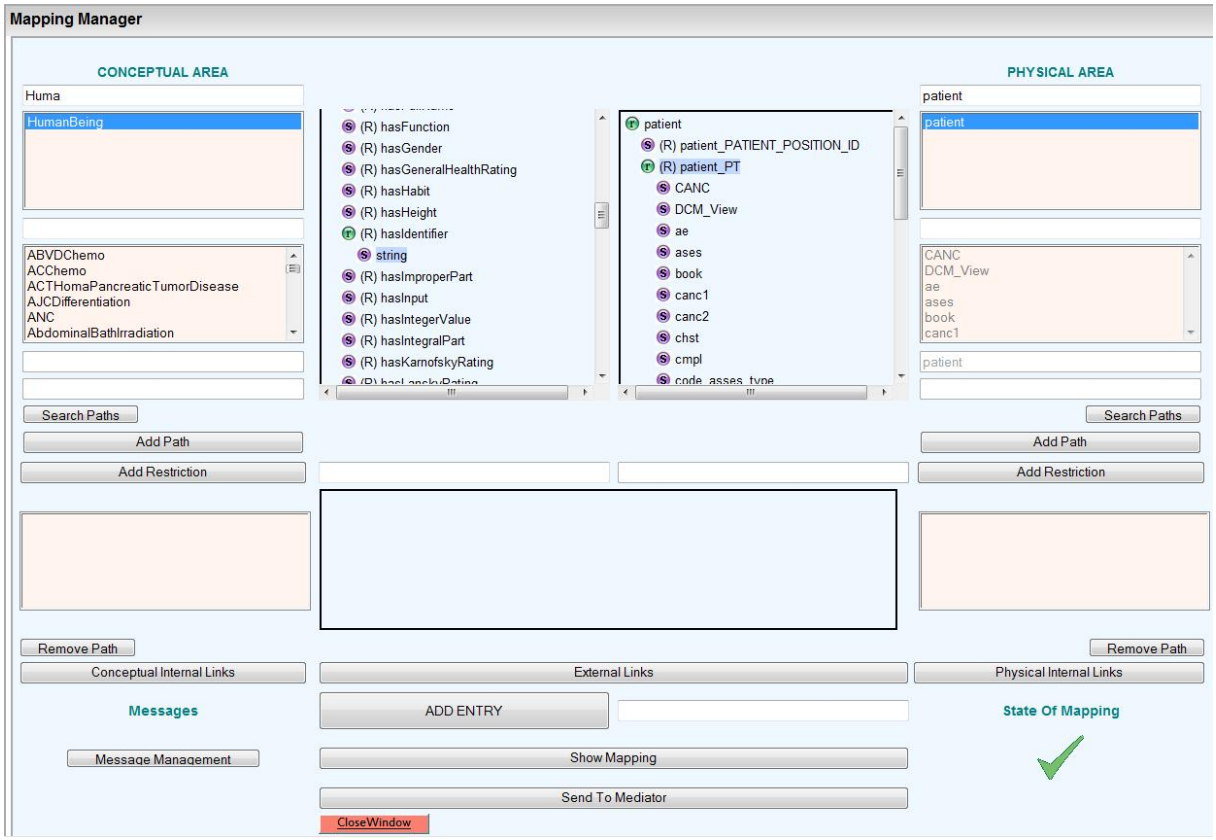

Figure 29: The entry is created through the mapping process

After the paths have been specified and the external links have been established, a natural language description is provided. The aim of this description is express in natural language the conceptual paths included in the entry. This description will be later shown in the Query Tool and will aid users during the query construction process. In this case, the description provided is “Patients and identifiers”.

Another entry included in this mapping reflects the fact that the patients of a clinical trial have an associated date of birth. Once again, there is a direct path in the Master Ontology to specify this information. The class *HumanBeing* is related with the datatype *date* through the relation *hasBirthDate*. In the physical database, on the other hand, this same information is separated in two different paths that must be linked together. The first the personal data item

of a clinical trial patient with the patient itself (demo → PATIENT → patient), and the second path relates the personal data item with a birth date (demo → demo_BIRTH_DATE → Thing). Figure 30 depicts this entry.

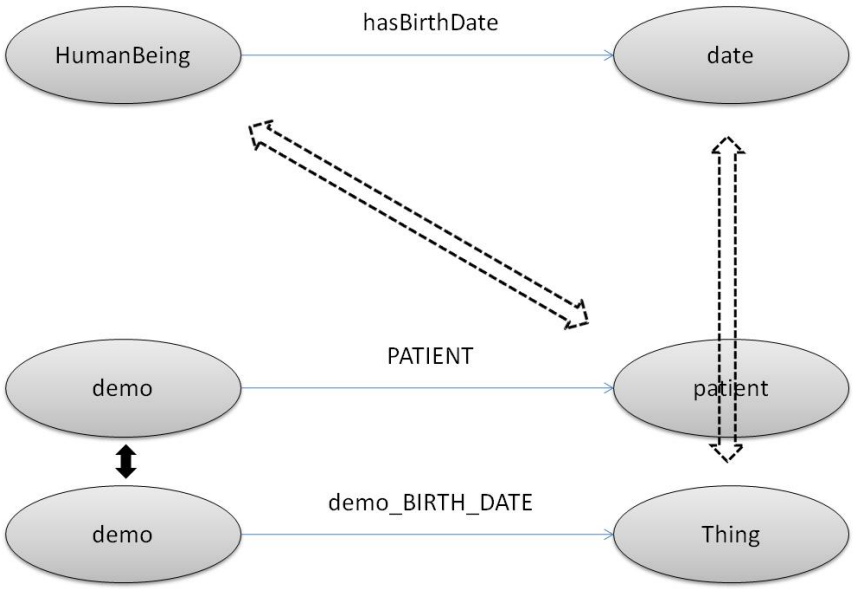

Figure 30: The second entry involves two physical paths

The entry is created as before. The internal links are created through the Bounds Manager window. Figure 31 depicts this case.

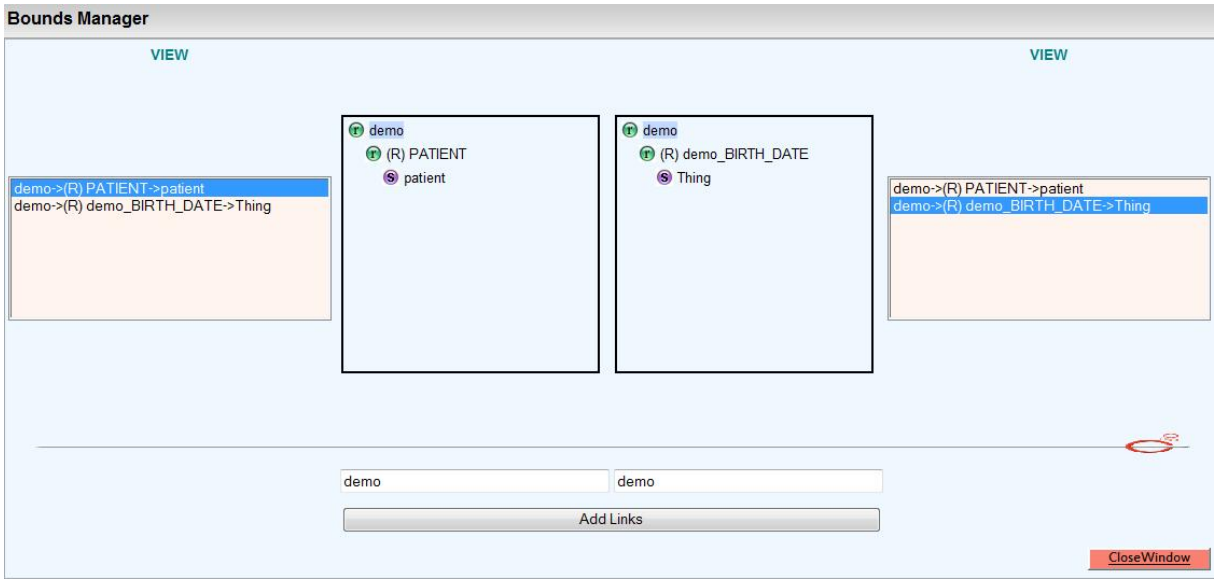

Figure 31: Establishing internal links in an entry

Other two entries involving the patient's registrations and the registration's dates are created to complete the mapping. The final state of the mapping can be consulted, as shown in figure 32.

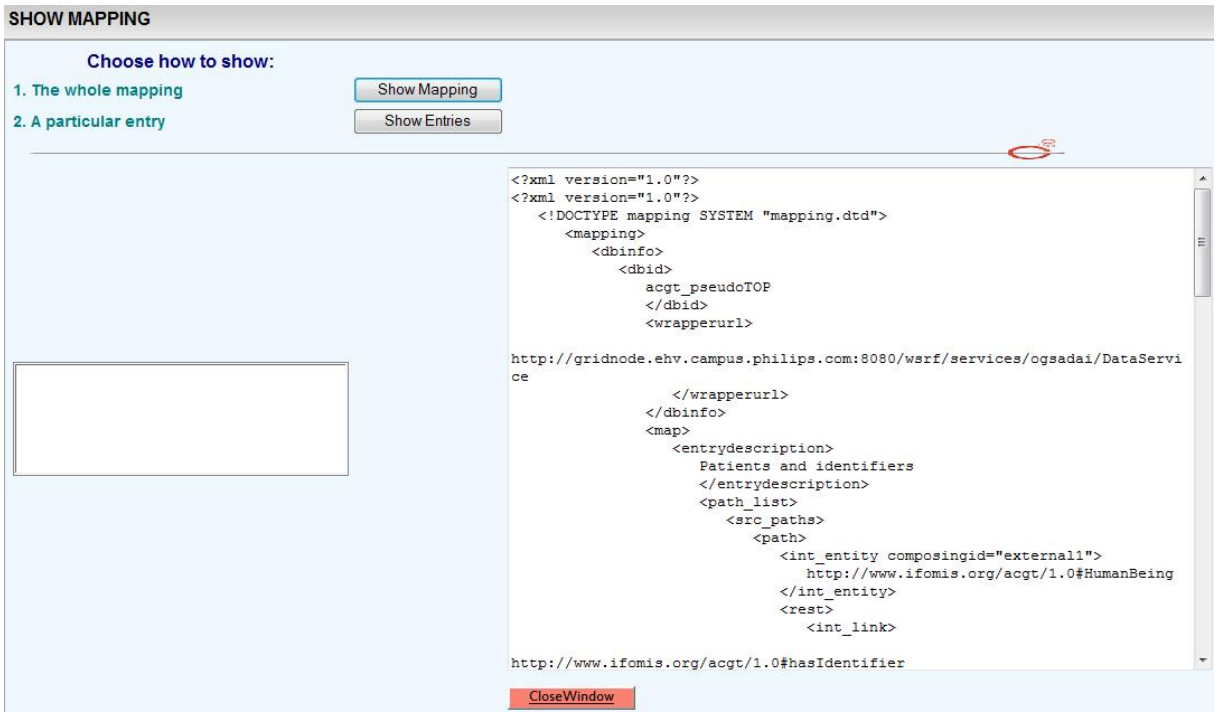

Figure 32: The final state of the mapping

Finally, the user submits the mapping to the Semantic Mediator. Subsequent queries launched against the mediator will be redirected to the pseudoTOP database, in case they contain any of the conceptual views mapped to physical views.

3.2. Query building process in the Pseudo-TOP scenario

In the Pseudo-TOP scenario, as described in D7.4, the user wants to retrieve the list of patients registered in the TOP trial, together with their birth dates and dates of registration. The needed query must include a WHERE clause containing the views related to the information she wants to retrieve. Table XXX depicts the views in SPARQL that must be included and their meanings.

SPARQL view	Meaning
?humanbeing_1 p0:hasIdentifier ?string_1. ?humanbeing_1 a p0:HumanBeing. ?string_1 a p1:string.	*Relation between patients and their identifiers in the clinical trial*
?humanbeing_2 p0:hasBirthDate ?date_2. ?humanbeing_2 a p0:HumanBeing. ?date_2 a p1:date.	*Relation between patients and their birth dates*
?registration_3 p0:hasProcessEnd ?processboundary_3. ?registration_3 a p0:Registration.	*Relation between registrations and registration dates*

?processboundary_3 a p2:ProcessBoundary. ?processboundary_3 p0:hasDate ?date_3. ?processboundary_3 a p2:ProcessBoundary. ?date_3 a p1:date.	
?registration_4 p0:hasPatient ?humanbeing_4. ?registration_4 a p0:Registration. ?humanbeing_4 a p0:HumanBeing.	*Relation between patients and their registrations in the clinical trial*

Table 1: Views of the Pseudo-TOP scenario query

As can be seen, these views are comprised by paths extracted from the Master ontology. An end user would know about what she is seeking (the descriptions in the right column). Formulating these in SPARQL is made easy with the query builder, as it shows natural language descriptions of the views. In the tool, the user only has to select these descriptions and the SPARQL code is automatically included in the query. This is not an easy process, since these views must form a connected graph. The tool automatically includes the links between them when this is possible, and warns the user if it is not. The variables to be included in the SELECT clause must be also selected, in order to configure the shape of the results. Figure 33 shows the behaviour of the query builder in this step for the Pseudo-TOP scenario query.

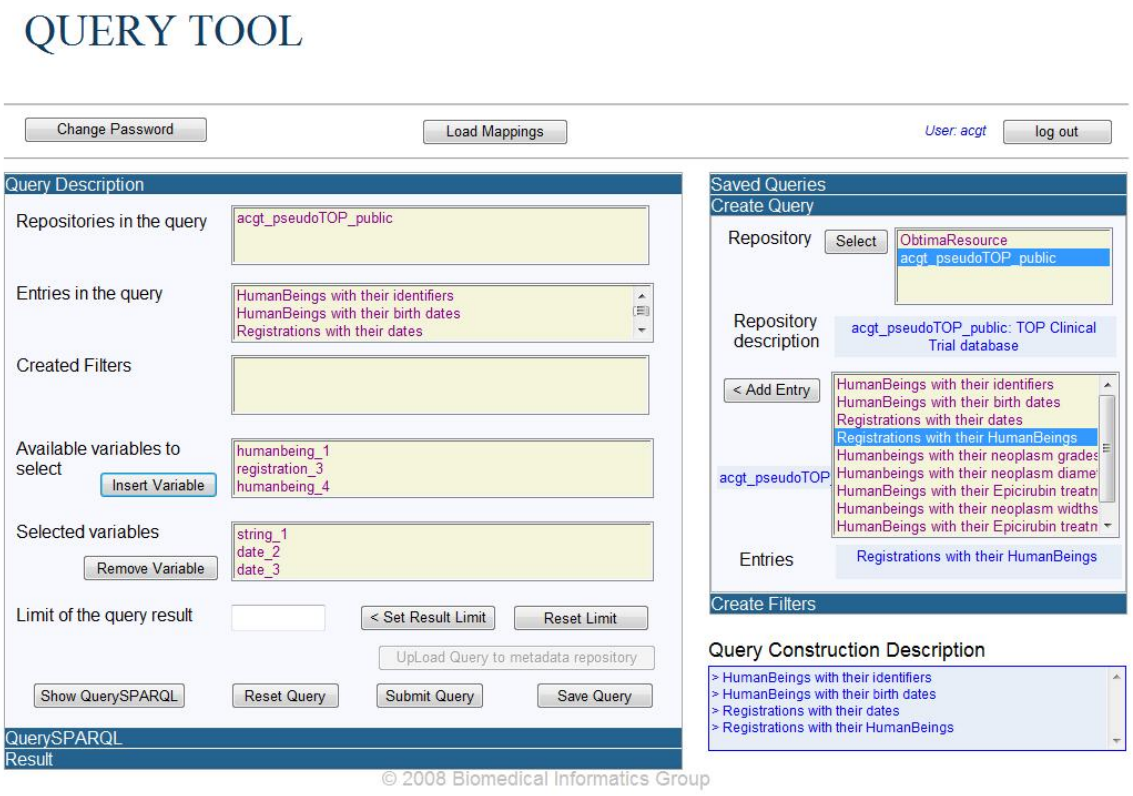

Figure 33: The Query Builder: view selection step in the query construction process

A SPARQL query can also include constraints. In fact, given that it is a need that the graph formed by the views in the SELECT clause is connected, most times the inclusion of such constraints is necessary. In this particular case, the tool automatically includes two of the needed constraints, and the user formulates the third one. The constraints included and their respective meanings are shown in table XXX.

SPARQL constraint	Automatic	Meaning
FILTER (?humanbeing_2 = ?humanbeing_1)	*Yes*	*Establishes that the patients of the first view are the same as the patients in the second view*
FILTER (?registration_4 = ?registration_3)	*Yes*	*Establishes that the registrations of the third view are the same as the registrations in the fourth view*
FILTER (?humanbeing_1 = ?humanbeing_4)	*No*	*Establishes that the patients of the first view are the same as the patients in the fourth view*

Table 2: Constraints of the Pseudo-TOP scenario query

The query builder includes a section that aids in this task. When the user in the query includes a new constraint, the tool generates a description in natural language that clarifies its meaning. Figure 34 illustrates this process in the query builder.

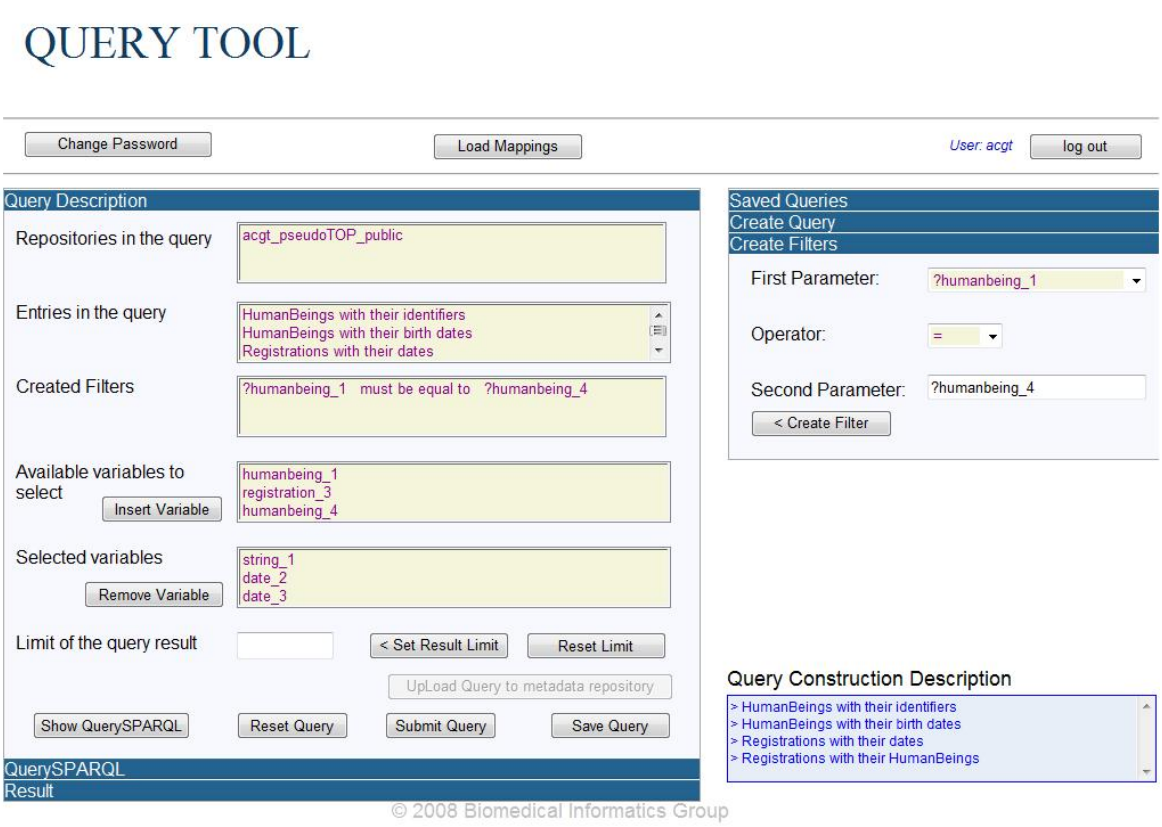

Figure 34: The Query Builder: constraint generation

The query builder hides the SPARQL code in any moment of the query construction process, but it is possible to consult it, and even modify it before submitting the query or storing it.

References

[OGSADAI] OGSADAI Perform documents specification. Available at: http://www.ogsadai.org.uk/documentation/ogsadai-wsi-2.2/doc/interaction/Perform.html

[SPARQL] SPARQL Query Language for RDF. Available at: http://www.w3.org/TR/rdf-sparql-query/