

DogmaModeler – a tool for ontology specification

Project ref.	IST- 2001-38248
Project Acronym	FFPOIROT
Project full title	Financial Fraud Prevention –Oriented Information Resources using Ontology Technology
Availability	Restricted



INFORMATION SOCIETY TECHNOLOGIES

Security (distribution level)	Restricted
Contractual date of delivery	M9
Actual date of delivery	May 2003
Document number	ffpoirot.TechNote.03.002
Document name	DogmaModeler – a tool for ontology specification
Type	Report
Status & version	V. 1.0
Number of pages	33
WP contributing to the deliverable	
WP/Task responsible	STARLab VUB
Other contributors	
Author	Mustafa Jarrar, Andriy Lisovoy
Editor	Gang Zhao
EC Project Officer	Johan Hagman

Table of Contents

Table of Contents	3
Introduction.....	4
PURPOSE OF THE DOCUMENT	4
DOCUMENT ORGANISATION.....	4
REVISION HISTORY	4
Dogma approach to ontology engineering.....	5
ONTOLOGY BASE.....	5
<i>Lexons</i>	5
<i>Context</i>	5
COMMITMENTS.....	5
APPLICATION DEPENDENCY AND ONTOLOGY MODELING.....	6
Functionalities of DogmaModeler	7
References	12

Introduction

Purpose of the document

This technical report documents the major functionalities as adapted for the FF POIROT project of *DogmaModeler*.

Document Organisation

Chapter 1	Introduction
Chapter 2	Dogma approach to ontology engineering
Chapter 3	User functionalities of DogmaModeler
Chapter 4	Illustration of main operations
Chapter 5	References

Revision history

Version	Author	Date	Comment
0.1	Gang Zhao	24 October 03	Created Chapter 1 based on Mustafa Jarrar's publication.
0.2	Gang Zhao	26 October 03	Created Chapter 2 based on Mustafa jarrar's publication.
0.3	Gang Zhao	28 October 03	Created Chapter 3 from Andriy Lisovoy's user manual draft
1.0	Gang Zhao	1 November 03	Finalised.

Dogma approach to ontology engineering

Unlike ontology engineering proposals, which consider an ontology as one “unit”, holding both conceptual relations and rules together, we decompose an ontology into an ontology base and a layer of ontological commitments.

Ontology base

The ontology base holds conceptual relations, as domain knowledge. The commitment layer consists of a set of ontological commitments, where each commitment holds ontology rules, which *formally and explicitly* provide an interpretation of an application or task in terms of the domain knowledge. Ontology rules are mostly application/task-dependent knowledge, i.e. strongly influenced by the intended use of the knowledge and requirements at hand. Therefore, as a result of the decomposition, the generality of the knowledge in the ontology base level is increased, while rules influenced by requirements at hand are kept separated in the commitment layer. Hence, a conceptual schema can be seen as an ontological commitment defined in terms of the domain knowledge.

Lexons

The ontology base consists of “plausible” intuitive domain fact types, represented and organized as a set of context-specific binary conceptual relations, called *lexons*. A lexon is formally described as a 5-tuple of the form $\langle \gamma: \text{Term}_1, \text{Role}_1, \text{Role}_2, \text{Term}_2 \rangle$, where γ is a context identifier, used to group lexons that are intuitively and informally “related” in an intended conceptualization of a domain. For each context γ and Term T, the pair (γ, T) is assumed to refer to a *concept*.

Context

At the domain level, in our approach, we have introduced the notion of context; so that a term within a context refers *intuitively* to a concept. The intuition that a context provides here is: *a set of implicit or maybe tacit assumptions that are necessary for all instances of a concept in all its applications*. In other words, a context is an abstract identifier that refers to implicit and tacit assumptions in a domain, and that maps a term to its intended meaning (i.e. concept) within these assumptions. Notice that a context in our approach is not explicit formal knowledge. In practice, we define a context by referring to a source (e.g. a set of documents, laws and regulations, informal description of “best practice”, etc.), which, by *human understanding*, is assumed to “contain” the necessary assumptions.

Commitments

The layer of ontological commitments mediates between the ontology base and its applications. Each commitment consists of: (1) an *ontological view* that specifies which lexons from the ontology base are relevant to this commitment, i.e. selection of lexons, and (2) rules that formally constrain and specify the intended meaning of the selected lexons. For simplicity, one can see a commitment as a conceptual schema, where its conceptual relations correspond to lexons in the ontology base. Applications that use (or more precisely, “commit to”) a certain commitment must satisfy all rules declared in this commitment. In other words, any possible world, for an application, must conform to the rules declared in its commitment(s). Each ontological commitment corresponds to an explicit *instance* of an (intentional) first order *interpretation* of the domain knowledge in the ontology base. In other words, it is the role of commitments to provide the formal interpretation(s) of the lexons. Therefore, the lexons in an ontology base are free of a *particular* formal interpretation. This allows different formalizations and interpretations, even if sometimes they disagree about certain things, to co-exist as different commitments in the commitment layer and to share what they have in

common. In our approach, ontological commitments are not restricted to be expressed in a certain specification language. Commitments can be modelled using the ORM graphical notation.

Application dependency and ontology modelling

We suppose that the ontology base –as intuitive domain knowledge- is free of any particular formal interpretation; or rather, lexons are assumed (by human understanding) to be “true within their context’s source”. The formal interpretation of the lexons is provided through ontological commitments, which are explicit and formal (and thus machine-understandable) knowledge.

As a result and as we have illustrated before, we enable the use of conceptual modelling methods for modelling ontological commitments. The application-independency level of an ontology is increased, by separating the commitments (mostly application/task-dependent knowledge) from the ontology base (intuitive domain knowledge). In other words, the interaction problem has “neglect able” influence on the generality of the ontology base level, because ontology modellers are *prevented* from entering their application-specific rules at this level.

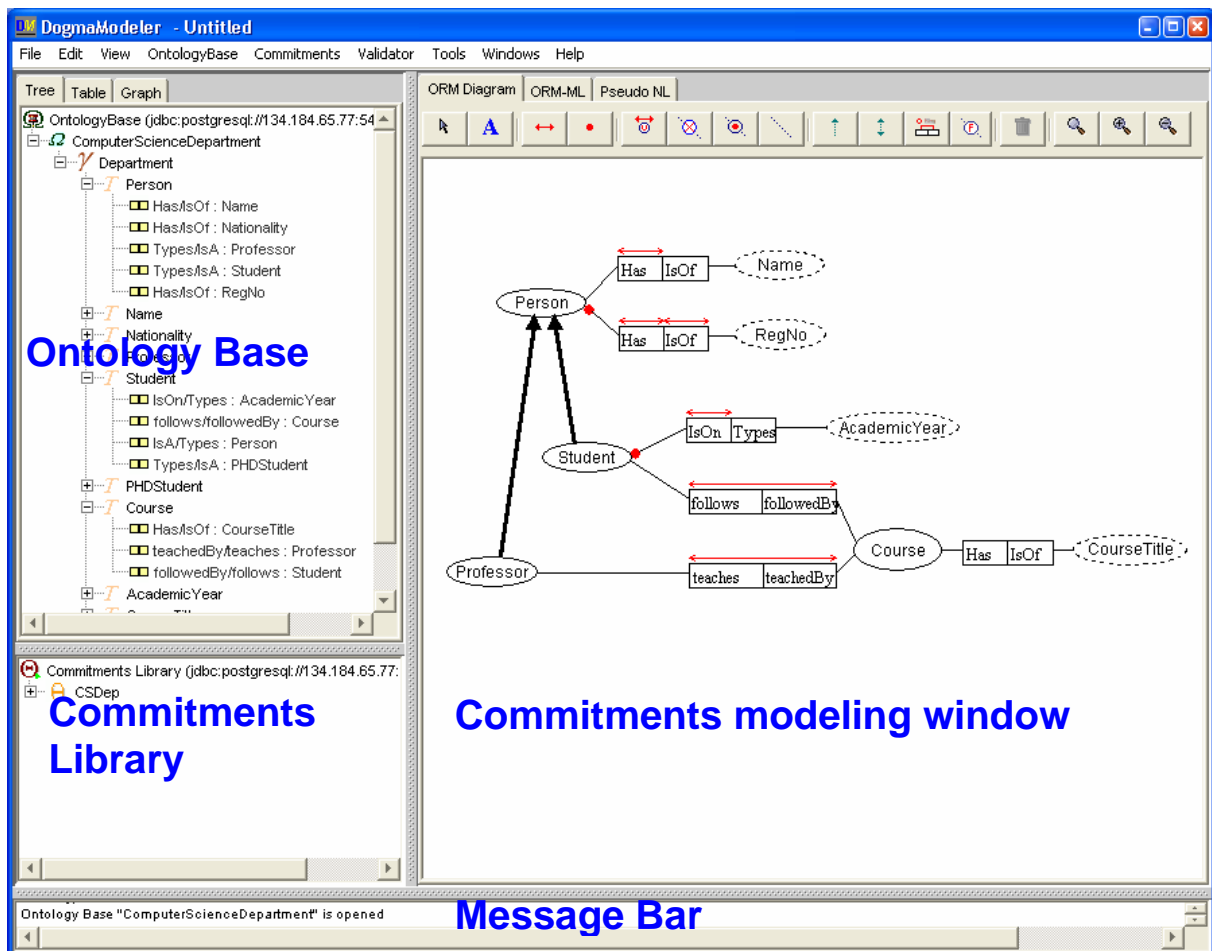
In accordance to the given independency discussion, we emphasize that modelling ontological commitments should not be specific to certain needs, they should be made more generic (e.g. describing application kinds, generic tasks, etc.), and seen as reusable components of knowledge. In our approach, the ontological commitments that are specific to a limited number of applications do not affect the independency of other commitments in the same commitment layer. Rather, commitments -especially large ones- can even be modularized into smaller and interrelated commitments, so that the general purpose –i.e. reusable- parts can be separated from the more specific parts. And therefore, not only the ontology base (i.e. lexons and the intuitive definitions of their terms) can be shared and reused among commitments, but also the ontological commitments themselves can be modularized and seen as a set of reusable knowledge components.

Functionalities of DogmaModeler

This section briefly outlines our DogmaModeler Tool prototype for ontology engineering. Its implementation is based on the approach described in the paper [1].

User interface windows

DogmaModeler supports functionalities for modelling, browsing, and managing both the ontology base and the commitments. It supports modelling ontological commitments using the ORM graphical notation, and it generates the corresponding ORM-ML [1] automatically. In addition, DogmaModeler supports verbalization of ontological commitments into pseudo natural language. Fig. 13 shows a screenshot of DogmaModeler with demonstrates its three main windows: the ontology base window, the commitment modelling window, and the commitment library window. We will describe these windows in what follows.




DogmaModeler consists of four window panes:

- Ontology base window. It is working space for editing and creating lexons in an ontology base.

- Commitment modeling window. It is the working space for editing and creating commitments using such conceptual modeling language as ORM
- Commitment Library window. It represents all commitments relevant to the lexons shown in the ontology base window.
- Message window. It shows system messages to the user.

Ontology base window

Before building ontological commitments, ontology builders should define their lexons in the ontology base window, in case it is empty. This window presents the set of lexons – $\{< \gamma : \text{Term}_1, \text{Role}, \text{Term}_2 >\}$ - in a tree-like structure. The first level, (Ω) represents ontology bases (e.g. Biblio-Ontologybase). In the second level, each node (γ) represents a context (e.g. Books). Within a context, each node (\mathcal{T}), in the third level, represents a term; while nodes (\square) in the fourth level, represent the set of (Role, Term_2) for that term.

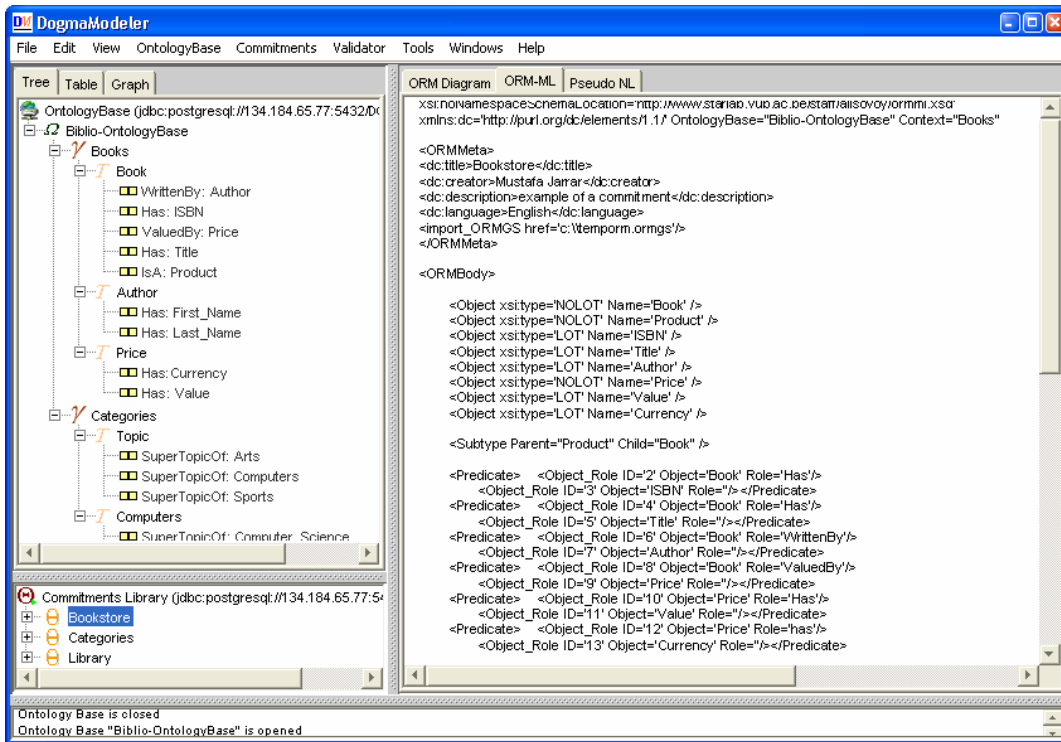
Notice that level 0 () in the tree represents an ontology base server, where the content of ontology bases is hosted and managed. All transactions on the ontology base (e.g. creating contexts, editing lexons, etc.) will be transmitted, verified and executed on the server side. As one can see in Fig. 13, DogmaModeler is connected with our DogmaServer¹, which stores and serves the ontology base and the commitment layer.

Commitment modelling window

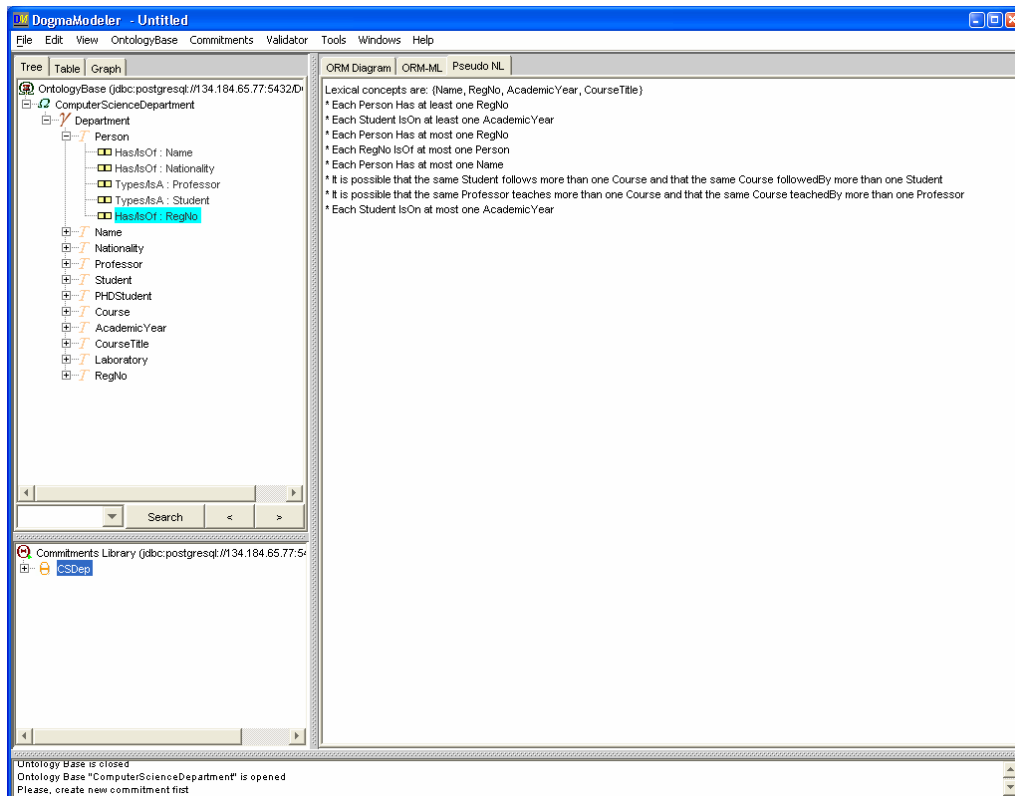
This window consists of three panels: ORM, ORM-ML, and Pseudo NL. To build an ontological commitment, ontology builders can drag and drop lexons from the ontology base window into the ORM panel (i.e. defining the ontological view). When doing so, lexons will be mapped automatically into ORM fact types. Then, in order to define rules on these lexons, ontology builders can use the ORM family of constraints; see icons in the top of the ORM panel.

Remark: mapping lexons as intuitive domain knowledge into ORM fact types that have predefined formal semantics is done as the following: a Term within a context is mapped directly into an Object Type in ORM, Roles within a context are also mapped directly into ORM roles. While in case of ORM Subtype relations that have specific “build-in” semantics, commitment builders need to customize the “Graph settings” window, in order to specify which roles should be mapped. Further, DogmaModeler does not support ORM unary roles and nested fact types.

DogmaModeler supports saving ORM-ML into text files, or uploading it into an ontology server. The following graph shows the ORM-ML tab of the commitment modelling window.



The graph below shows the commitments in Pseudo Natural language (fixed-syntax English sentences) of the ORM model.



It is automatically generated by the tool by applying predefined templates to the commitments' content. We believe that this allows non-experts to (help to) check, validate or build the commitment rules and will simplify the modeling process.

Commitment library window

The purpose of this window (Under the ontology base window) is to enhance the reusability, management, and organization of ontological commitments. The current implementation allows ontology builders to access and browse ontological commitments stored in a library (📁). Each node (📁) in the first level of the tree represents a commitment. By expanding a commitment node, the set of lexons and the set of rules -subject to this commitment- will appear in the second level. Advanced features e.g. indexing, modularization/composing, versioning, etc. of ontological commitments are ongoing research issues.

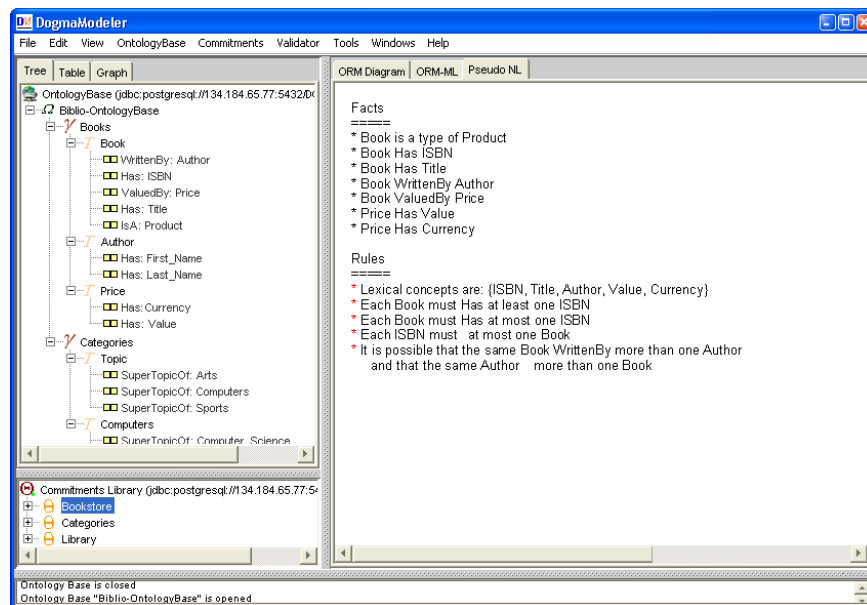
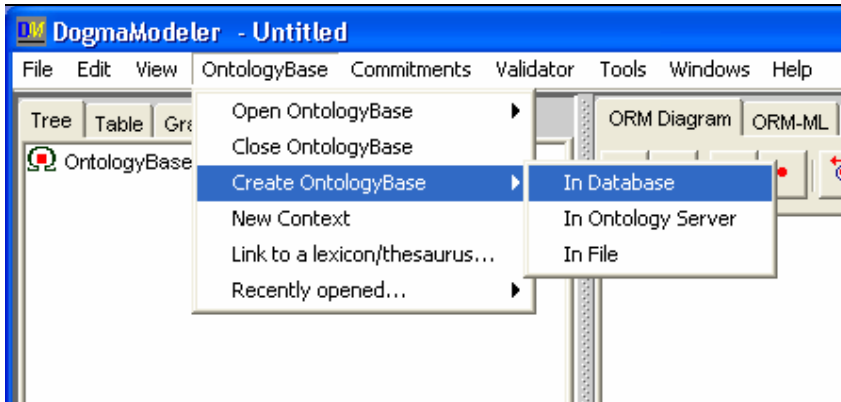


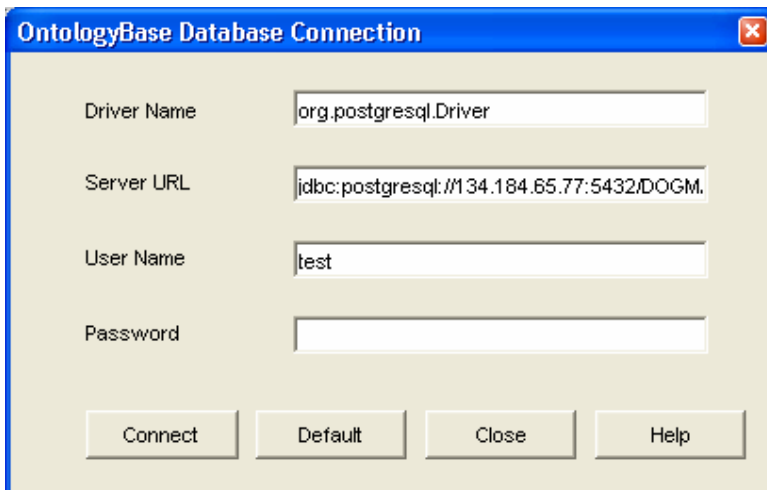
Illustration of main operations

Set up an ontology base

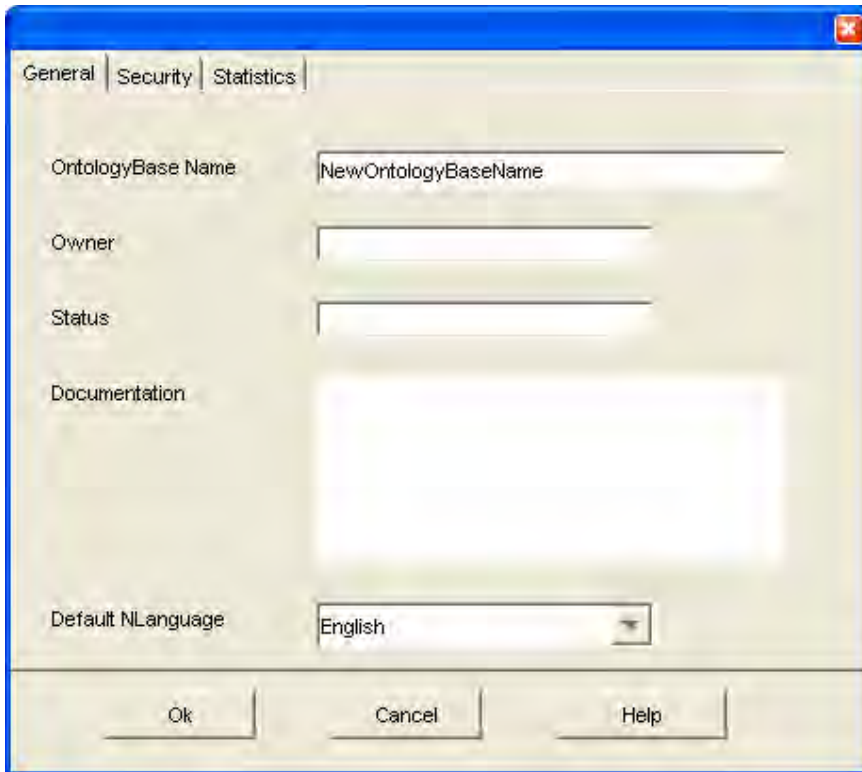
The operations on ontology bases are under the menu, *Ontology Base*.



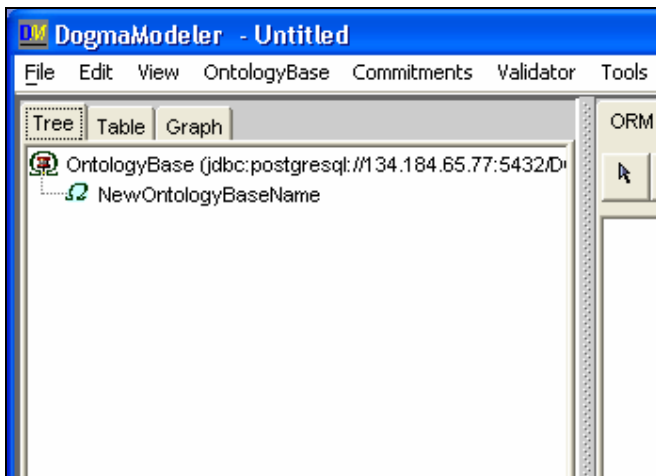
The Ontology Base database connection window will appear. Press "Default" button, if you don't know exactly what information you have to enter. You will be connected according to the last successful connection.



When you are connected, you will need to specify new Ontology Base name and its properties:

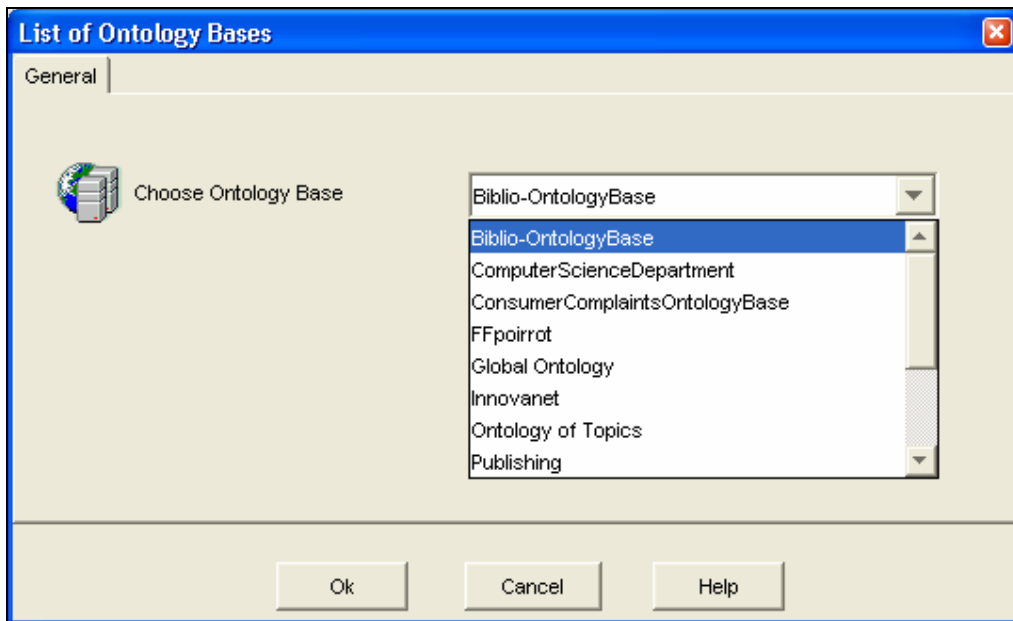


The name of the Ontology Base must be unique. When finished you will see your new empty Ontology Base.

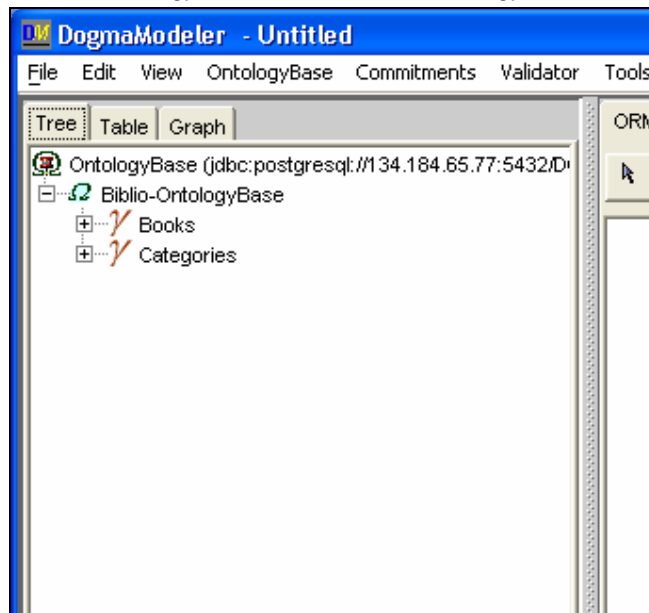


Open an ontology base

You will need to connect to the database of ontologies as if you set up a new ontology base described in the above. Once done, you will be presented a list of ontologies.

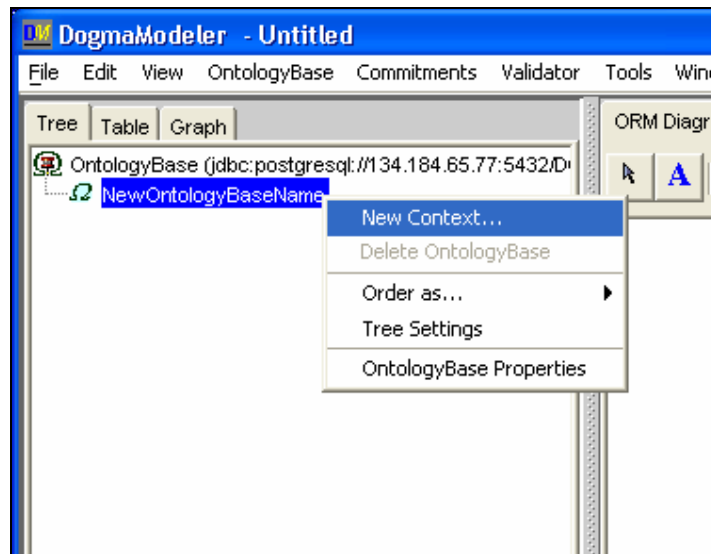


When you make your choice, the ontology will be shown in the ontology base window, as below.

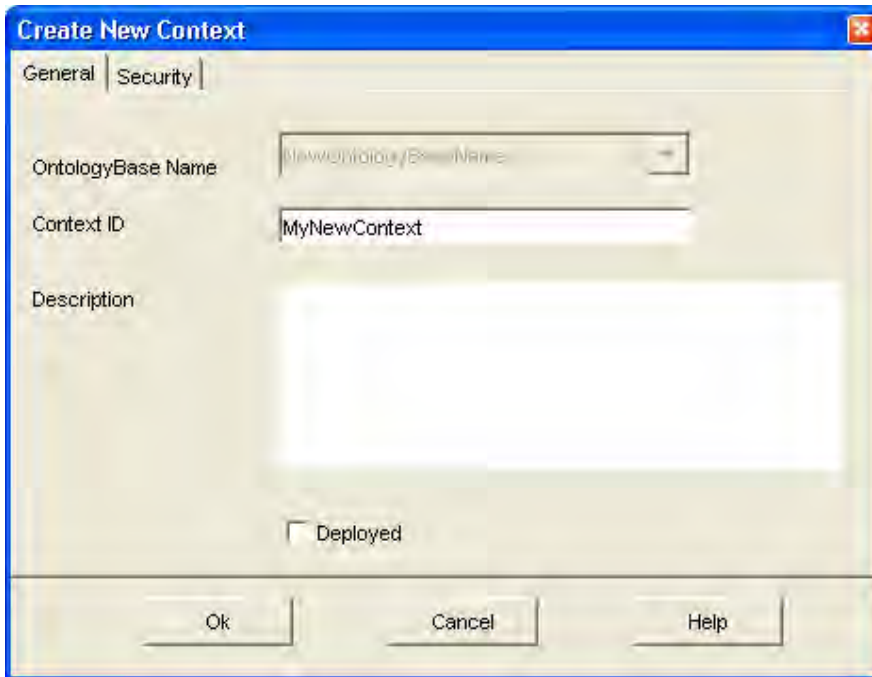


Creating a context

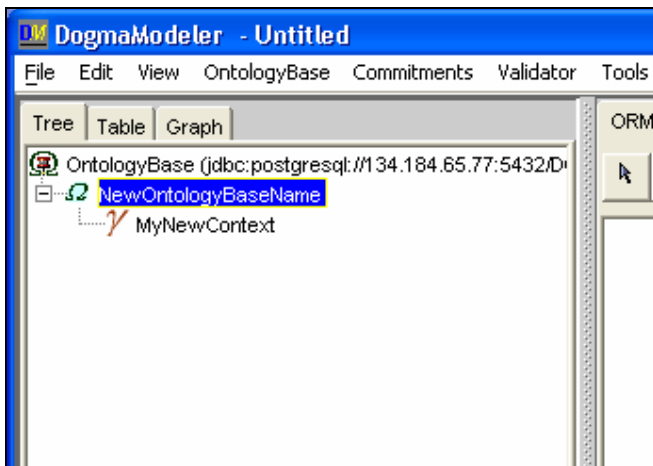
With a right-click on the mouse on an ontology base node you can choose “new Context”.



This allows you to create new context within the ontology.

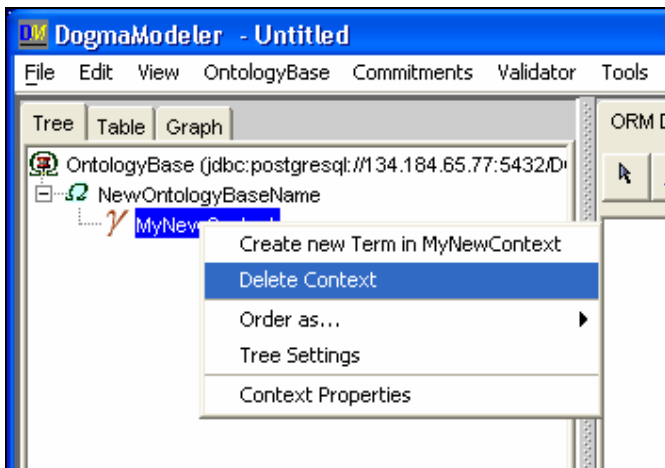


The new context will appear in the ontology base window:



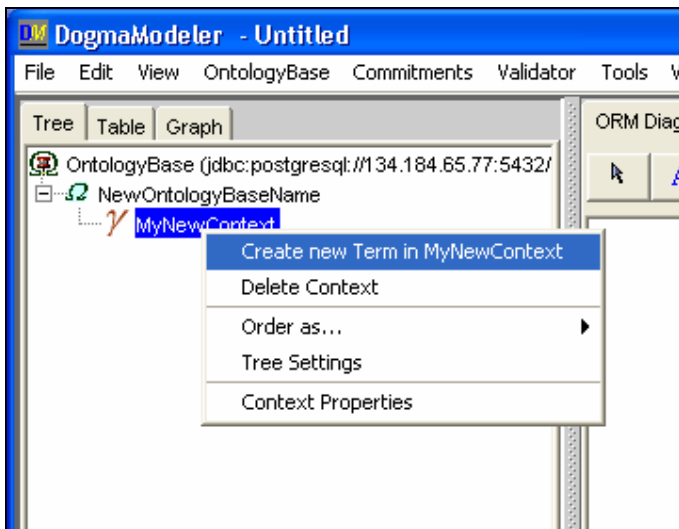
Delete a context

You can delete the context by right-click on the context node and select "Delete Context". Note that the action "Delete Context" will delete all terms and lexons in this context as well.

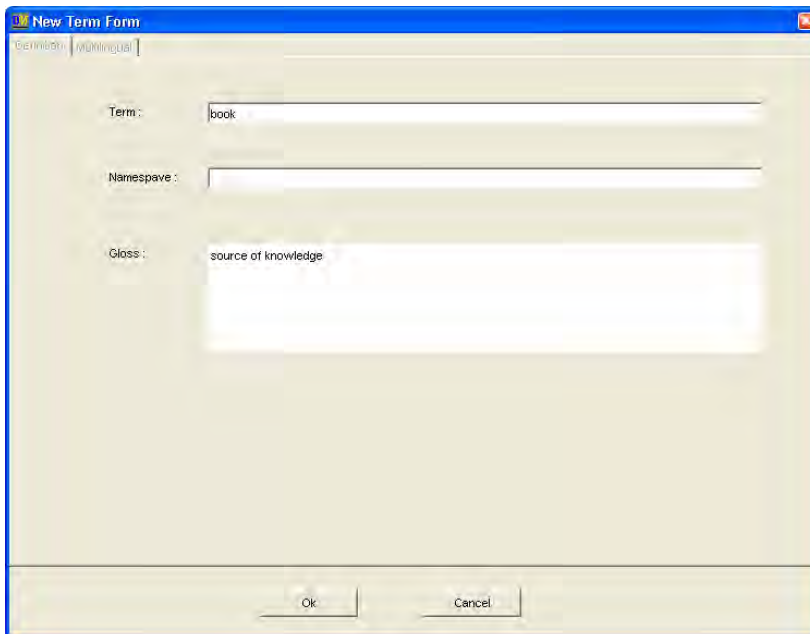


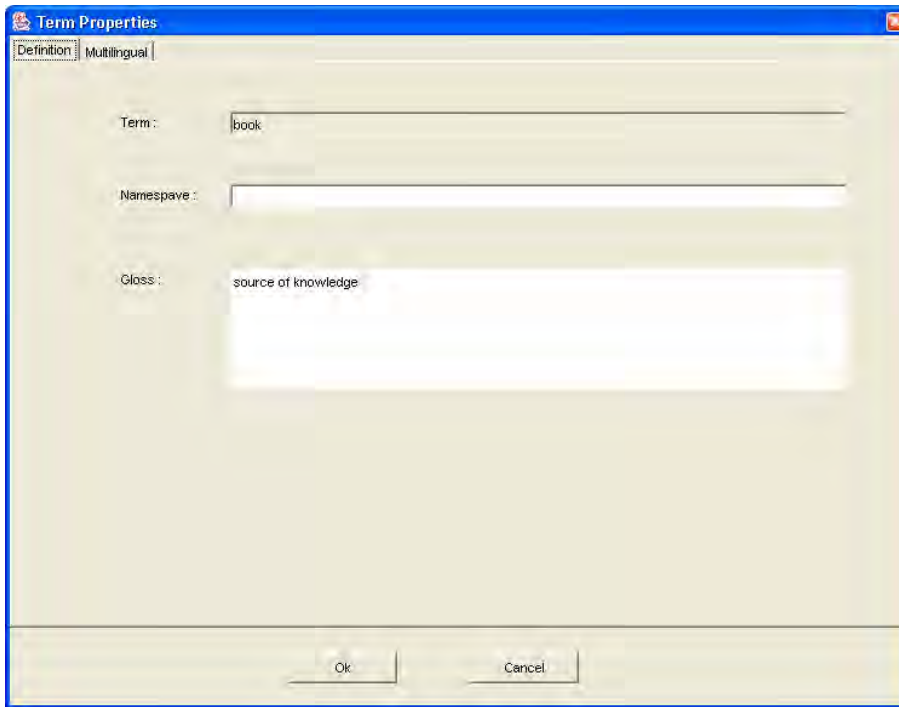
Creating a term

Once a context is created, terms and lexons can be created within the context. "Create new term" can be brought up by a right-click on the context.

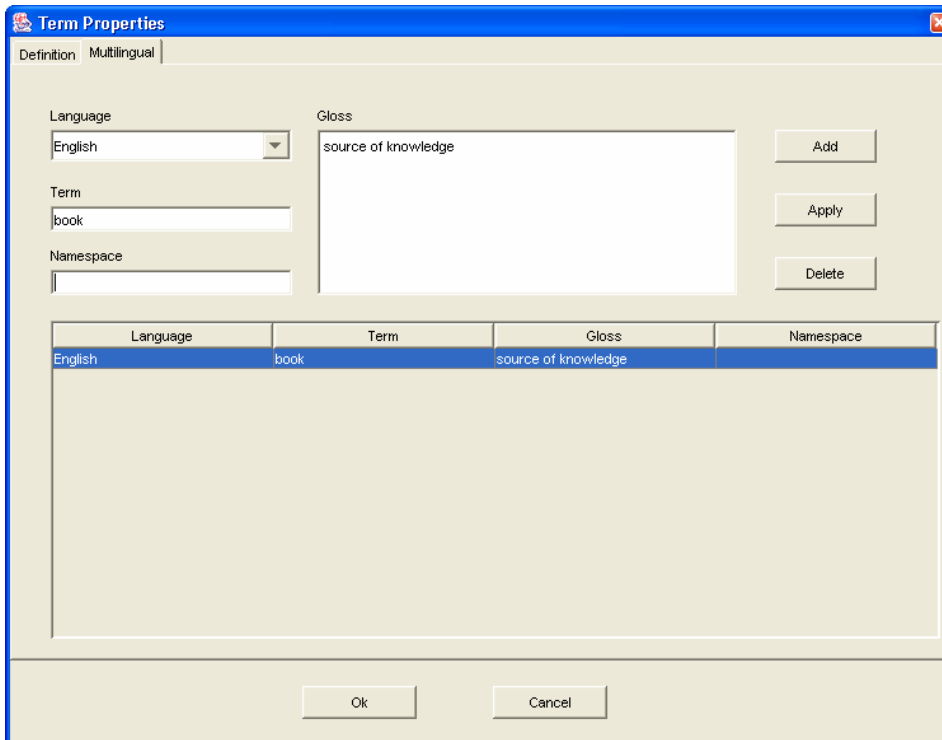


A term is created in the term property form. The following graph shows the definition tab.

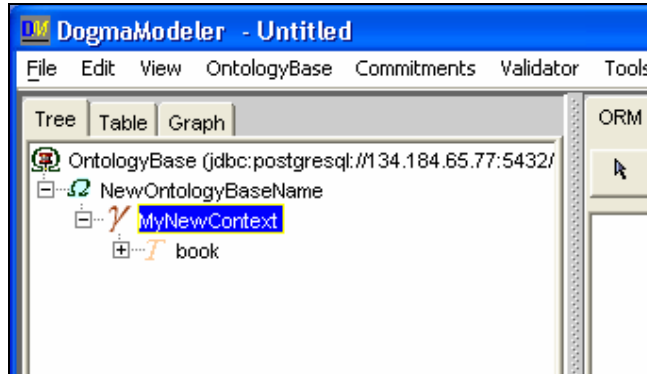




With Multilingual tab, you may add, delete or edit multilingual expressions of the term

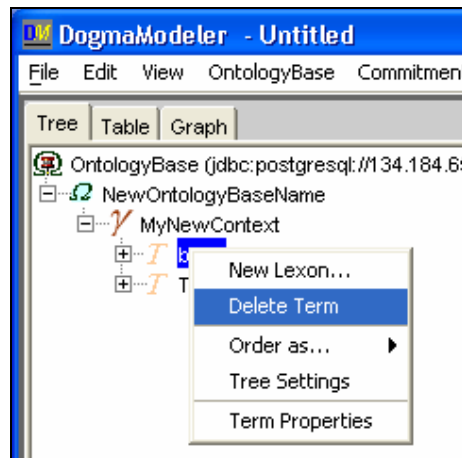


A new term will appear in a chosen context:



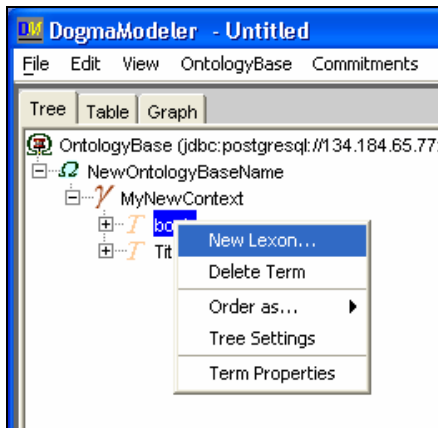
Delete a term

Right click on the term to delete and choose "Delete Term".

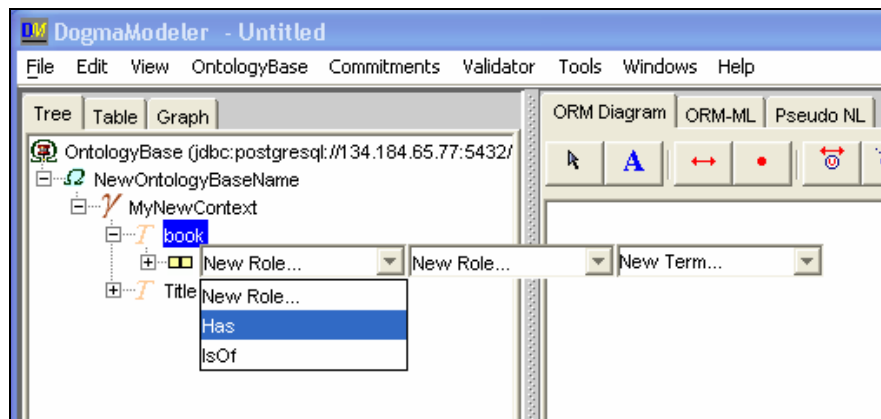


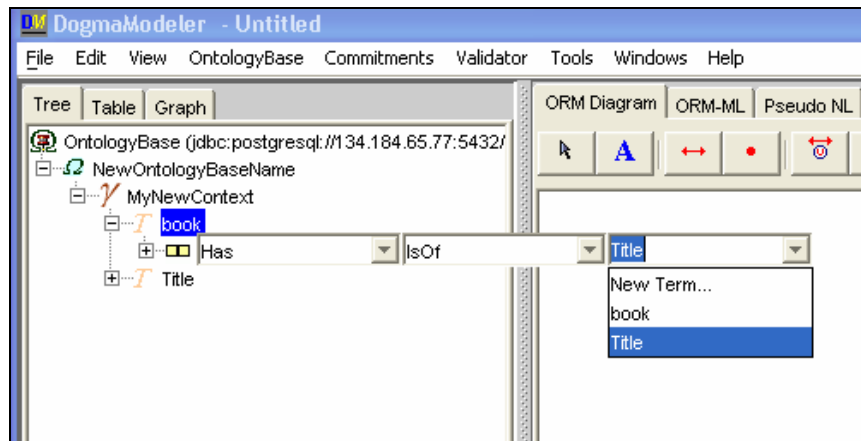
Creating a lexon

You can choose New Lexion by a right click on a term.

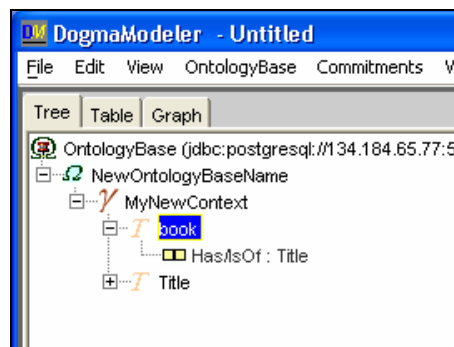


Three comboboxes will appear under the selected term, so the user can choose from existing Roles and Terms. If they are not in the lists, the user needs to add new ones by selecting "New Role"/"New Term":



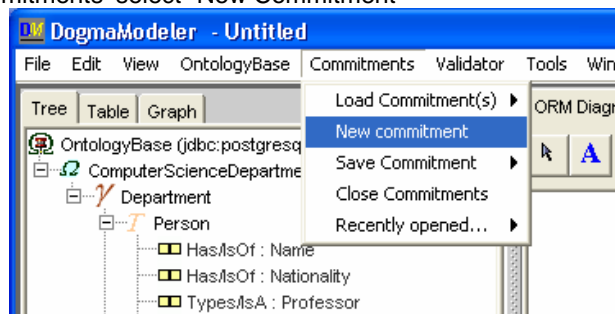


A new lexon will appear under the chosen term:

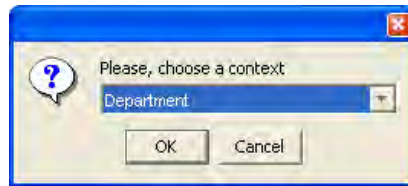


Creating a new commitment

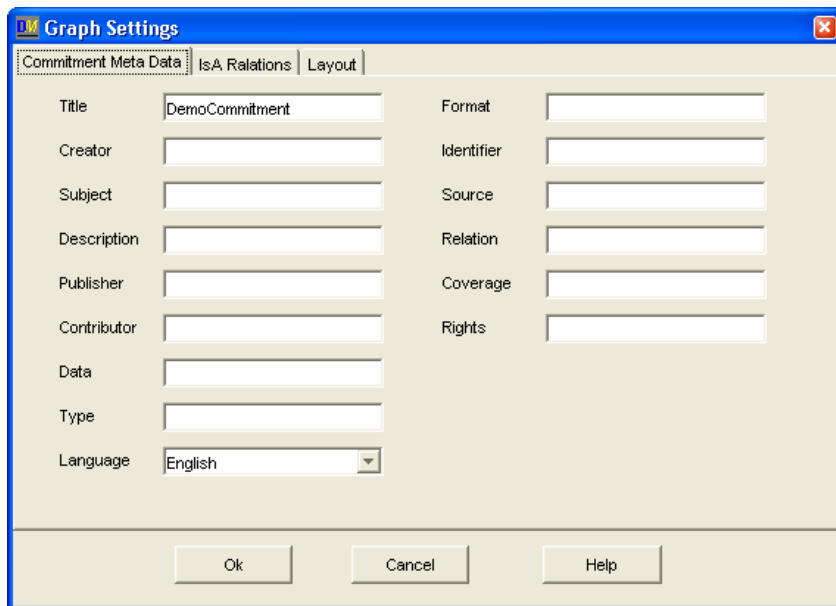
- First, an ontology base should be opened (see chapter – opening an ontology base)
- In the menu “Commitments’ select “New Commitment”

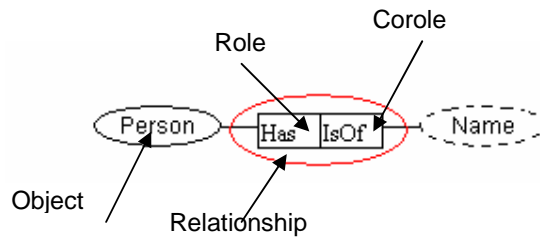


- Then a context for a new commitment should be chosen. Note that afterwards you cannot use lexons from other context



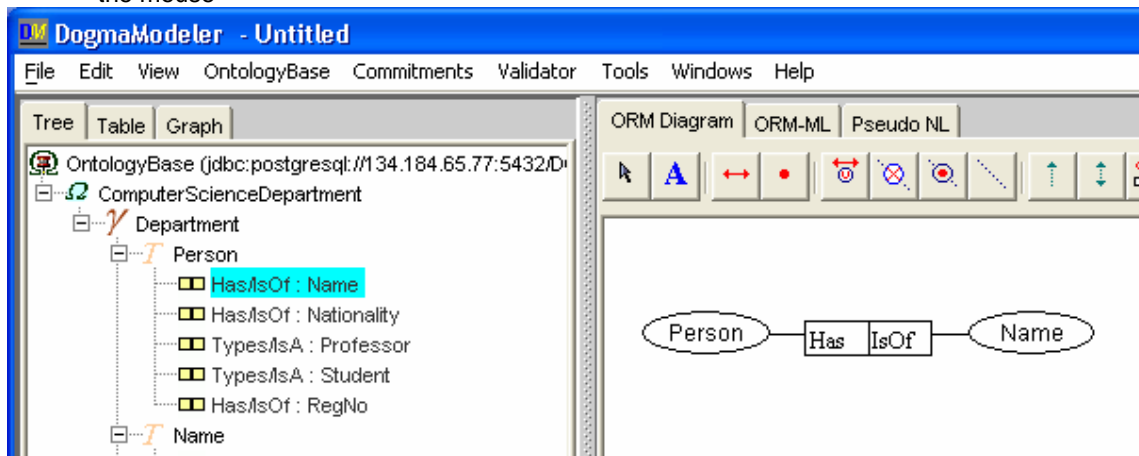
- In "Commitment Meta Data" form you must provide a name for the new commitment



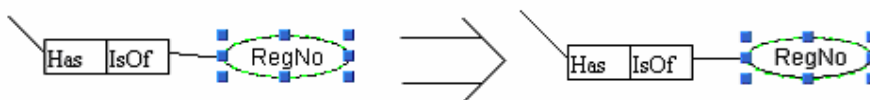


Select lexons

- Set of lexons (from the chosen context) you want to use in a commitment can be dragged and dropped one by one from the ontology base tree to a diagram. For this you press with the left button of the mouse on a lexon node, drag it to a desired place of the diagram and release the left button of the mouse

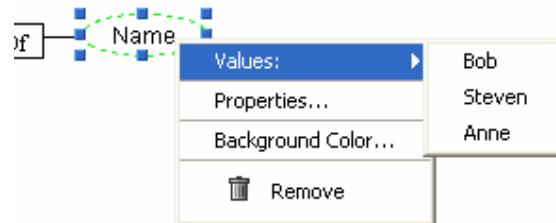


- By pressing ctrl+shift and using arrow keys on your keyboard you are able to move a graphical object pixel by pixel (first select desired object/set of objects);



Set object properties

Click with the right button of the mouse on an object. In the appeared popup, you will see values defined for selected object:

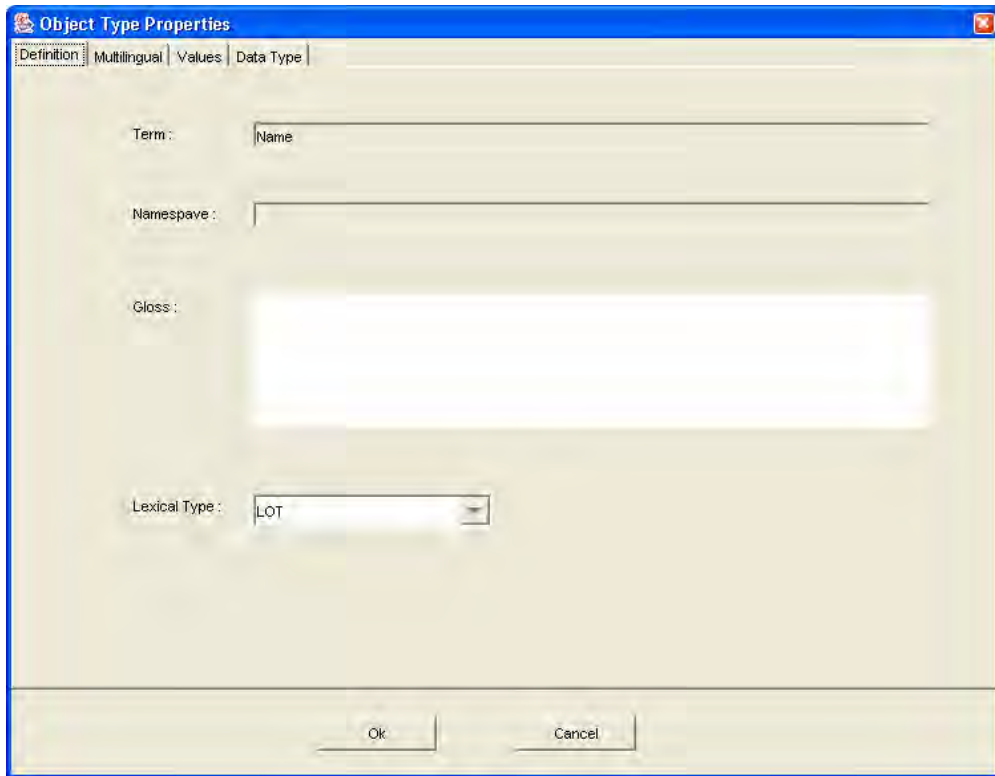


You can select "Properties" to "view/edit" object properties:



The "Object Type Properties" Form will appear with 4 tabs:

- Definition – you may view the definition of the concept represented by this object and you may defined this object as LOT/NOLOT:

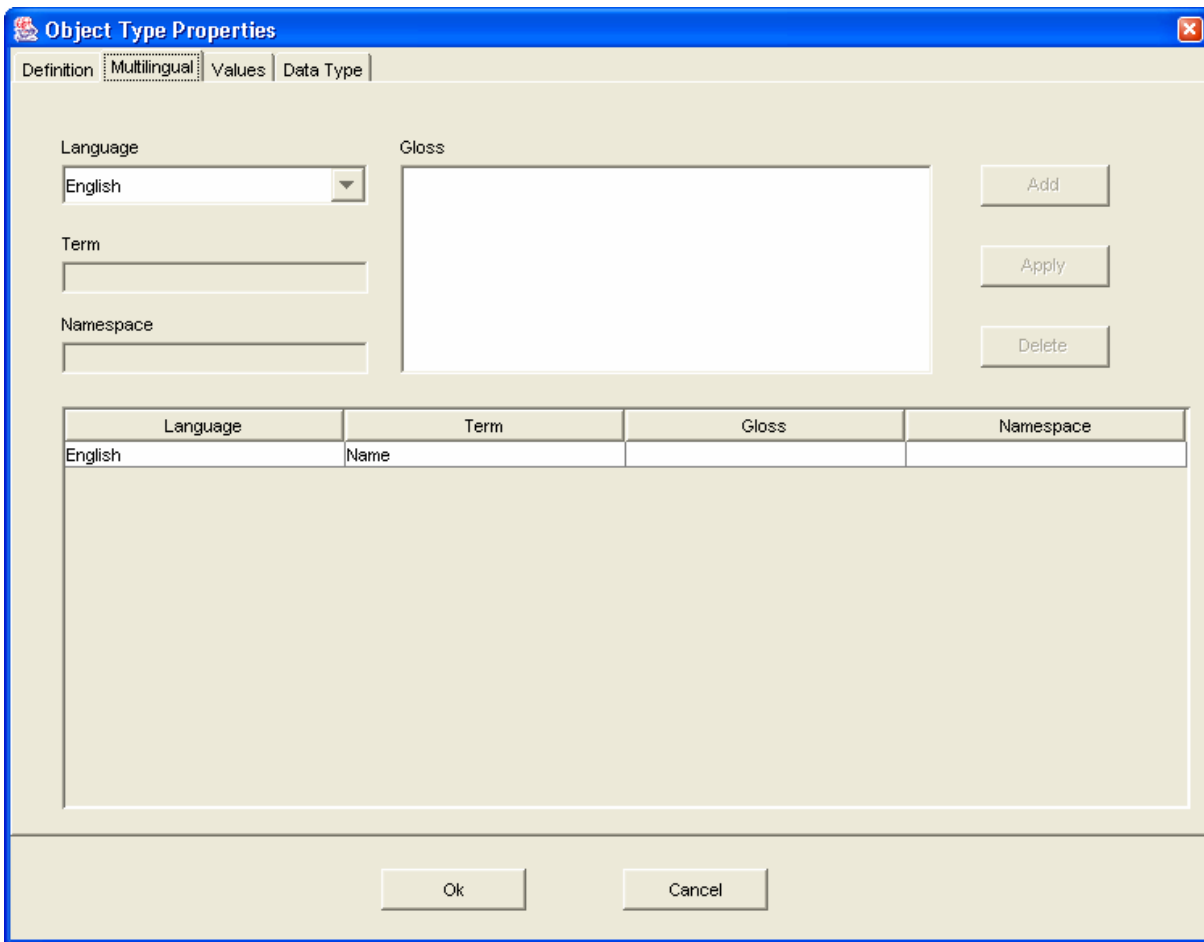


The image shows a software dialog box titled "Object Type Properties". It features a tabbed interface with four tabs: "Definition", "Multilingual", "Values", and "Data Type". The "Definition" tab is currently selected. The dialog contains the following fields:

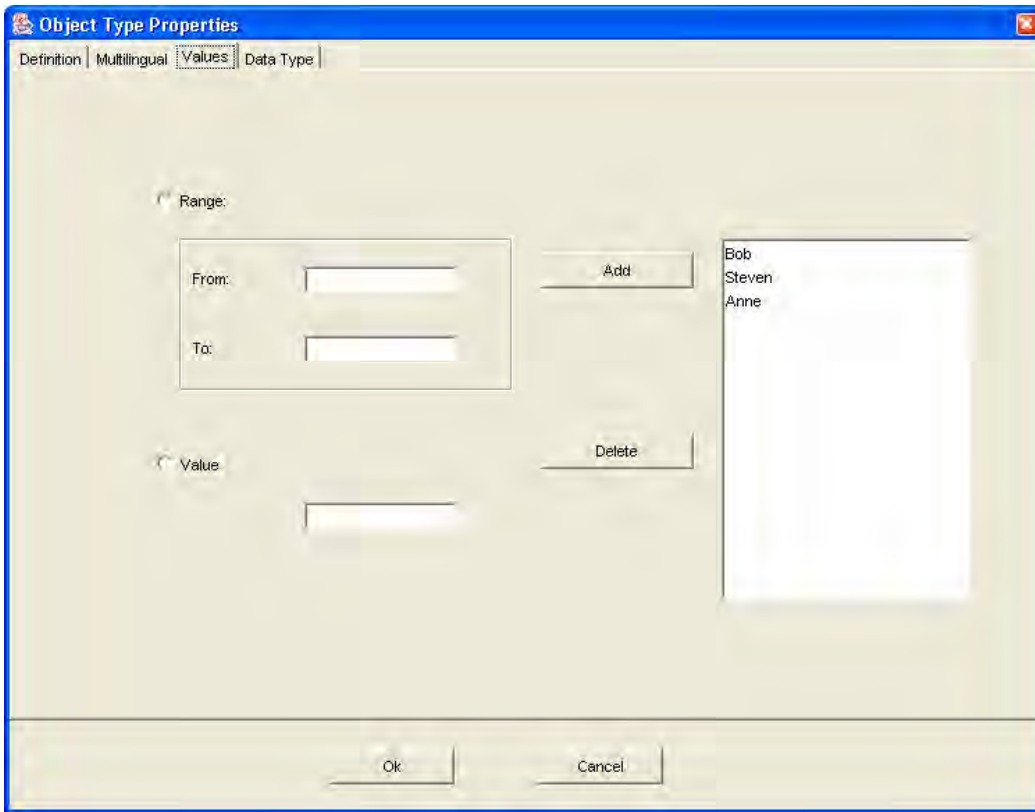
- Term :** A text input field containing the word "Name".
- Namespace :** An empty text input field.
- Gloss :** A large, empty text area for entering a gloss.
- Lexical Type :** A dropdown menu with "LOT" selected.

At the bottom of the dialog, there are two buttons: "Ok" and "Cancel".

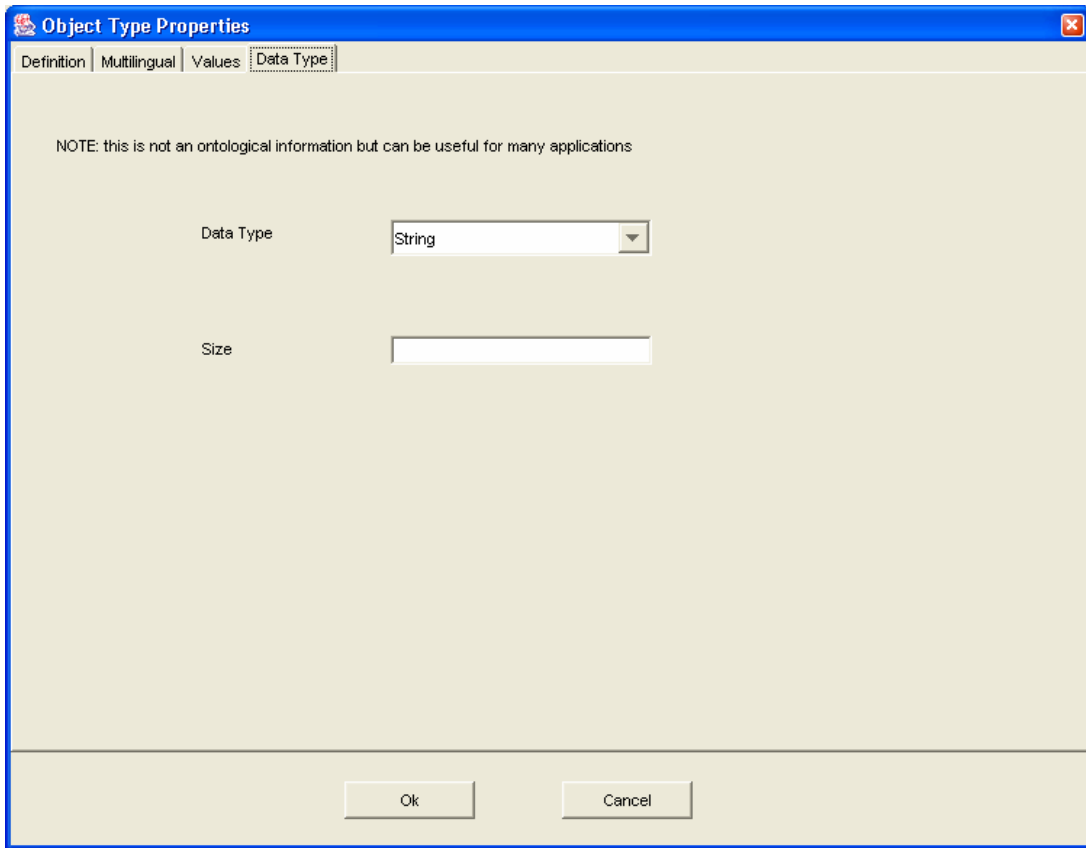
- Multilingual – you may view interpretations of a concept in different languages.



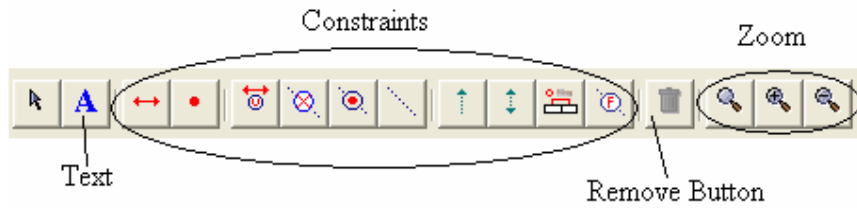
- Values – you may define values for the object



- Data Type – you may define a data type for the selected object.

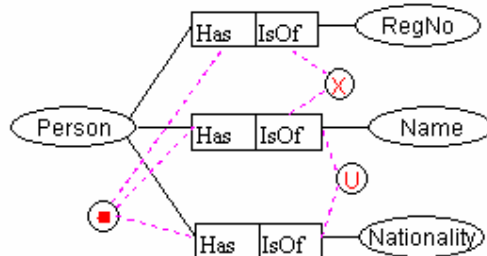
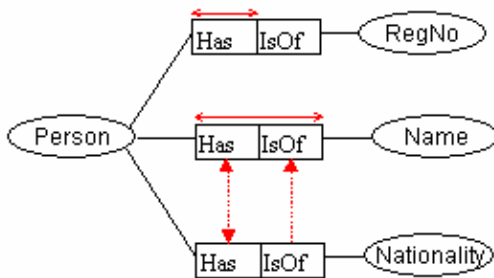


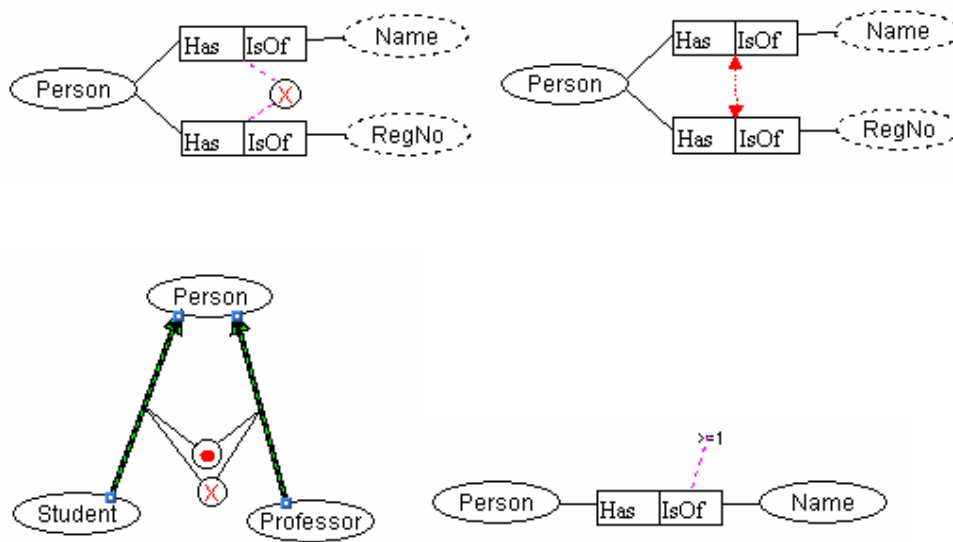
Add constraints



Types of constraints

- Mandatory constraint
- Uniqueness constraint -
- Subset constraint -
- Equality constraint -
- External uniqueness -
- External mandatory -
- Exclusion -
- Total constraint
- Exclusive constraint
- Ring constraint
- Frequency constraint





Save a Commitment

In the menu "Commitments" select "Save Commitment":



If you choose "save (as) to file", it will ask you where you want to save this commitment.

Save to DB will save this commitment into a database you currently connected. After saving into a database you will see it in the commitment library.



References

[1] *Jarrar M., Demy J. and Meersman R., On Using Conceptual Data Modeling for Ontology Engineering. In Aberer K., March S., and Spaccapietra S., (eds): Journal on Data Semantics, Special issue on "Best papers from the ER/ODBASE/COOPIS 2002 Conferences", Vol. 1.1, Springer,(2003)*