

---

# RDBToOnto User Guide

Version 1.2 beta

From Relational Databases to Fine-Tuned Populated Ontologies

---

**Farid Cerbah (Dassault Aviation)**

farid.cerbah@dassault-aviation.fr

**Abstract.**

RDBToOnto is a tool that allows to automatically generate ontologies from relational databases. A prominent feature of this tool is the ability to produce highly structured ontologies by exploiting structuring patterns hidden in the data. Though automated to a large extent, the process can be constrained in many ways through a friendly user interface.

RDBToOnto can automatically extract data from MySQL, Oracle, Microsoft Access and Excel sources.

Use instructions are fully described in this guide.

**Keyword list:** Ontology learning, relational databases, OWL

<b>Document Id.</b>	TAO/2008/D7.2a1/v1.2
<b>Project</b>	TAO IST-2004-026460
<b>Web links</b>	<a href="http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html">http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html</a>

## Contents

<b>1</b>	<b>Animated demos</b>	<b>3</b>
<b>2</b>	<b>What is RDBToOnto?</b>	<b>3</b>
<b>3</b>	<b>Conversion Projects</b>	<b>5</b>
<b>4</b>	<b>Starting RDBToOnto</b>	<b>6</b>
<b>5</b>	<b>Preparing the input data</b>	<b>6</b>
<b>6</b>	<b>Creating a new project</b>	<b>7</b>
<b>7</b>	<b>Opening a project</b>	<b>8</b>
<b>8</b>	<b>Saving the project and the generated ontology</b>	<b>8</b>
<b>9</b>	<b>Running the process</b>	<b>8</b>
<b>10</b>	<b>Feedback on work progress</b>	<b>9</b>
<b>11</b>	<b>Constraining the process</b>	<b>9</b>
11.1	Selecting the converter . . . . .	10
11.2	Populating the ontology . . . . .	10
11.3	Adding inverse properties . . . . .	10
11.4	Excluding tables . . . . .	10
11.5	Global categorization options . . . . .	11
<b>12</b>	<b>Adding local constraints</b>	<b>11</b>
12.1	Creating a local constraint . . . . .	12
12.1.1	Activating and deactivating local constraints . . . . .	13
12.1.2	Setting class name . . . . .	13
12.1.3	Instance naming . . . . .	13
12.1.4	Excluding columns . . . . .	14
12.1.5	Setting local categorization constraints . . . . .	14
<b>13</b>	<b>Database optimization</b>	<b>14</b>
13.1	Editing inclusion dependencies . . . . .	14

*CONTENTS* 2

13.2 Including optimization step in the process . . . . . 16

**A Preparing the data for the readers 17**

A.1 Getting data from MySQL databases . . . . . 17

A.2 Getting data from Oracle databases . . . . . 17

A.3 Getting data from Microsoft Access databases . . . . . 18

A.4 Getting data from Excel spreadsheets . . . . . 18

## 1 Animated demos

As a complement to this user guide, animated demos are provided to get a quick and illustrated overview of RDBToOnto:

- Commented survey of the functionalities on a typical example  
<http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto-demo.html>
- Another example including a database optimization step  
<http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto-demobis.html>

(note, however, that new functionalities included in 1.2 version are not illustrated in these demos).

## 2 What is RDBToOnto?

RDBToOnto is a highly configurable tool that can be used to derive fine-tuned populated ontologies from relational databases. Its main features are:

- **Generation of populated ontologies with rich structure:** The scope of this tool is not restricted to perform literal translations from source relational schemas and data tables to OWL ontologies. The main implemented method included in the platform allows identifying structuring patterns “hidden” in the data, and more particularly concept hierarchies.
- **Automated database readers for common database formats:** To feed the generation process with the data, some database readers are included (see appendix A) while allowing users to add new ones.
- **Constraints on the ontology generation process:** To guide the generation process, global and local constraints can be added by the user (specifying how to derive class hierarchies, defining naming patterns for classes and instances, excluding tables, etc).
- **A complete user interface** to ease the definition of ontology derivation projects, all required information are managed through a dedicated user interface. The resulting ontology can be progressively refined by iteratively adjusting the global and local constraints through the user interface.
- **Database optimization:** If required, an optimization module can be invoked to improve the input database, more particularly to eliminate redundancies.
- **Easy integration of new processing components:** Though RDBToOnto implements a complete ontology learning process, it is worth noting that it is primarily a platform allowing the implementation of new ontology learning algorithms. New implemented methods can be integrated and the user interfaces can be extended to handle specific constraints.

**Caution**

Some limitations of the tool should be stressed:

- RDBToOnto has been successfully tested on several medium-size databases (tables with ten of thousands entries). However, the included database readers are not optimized to deal with large databases.
- Though tested on several databases, the database optimization component should be considered being at the stage of “Alpha” version and the data should be carefully checked when this optimizer is invoked.

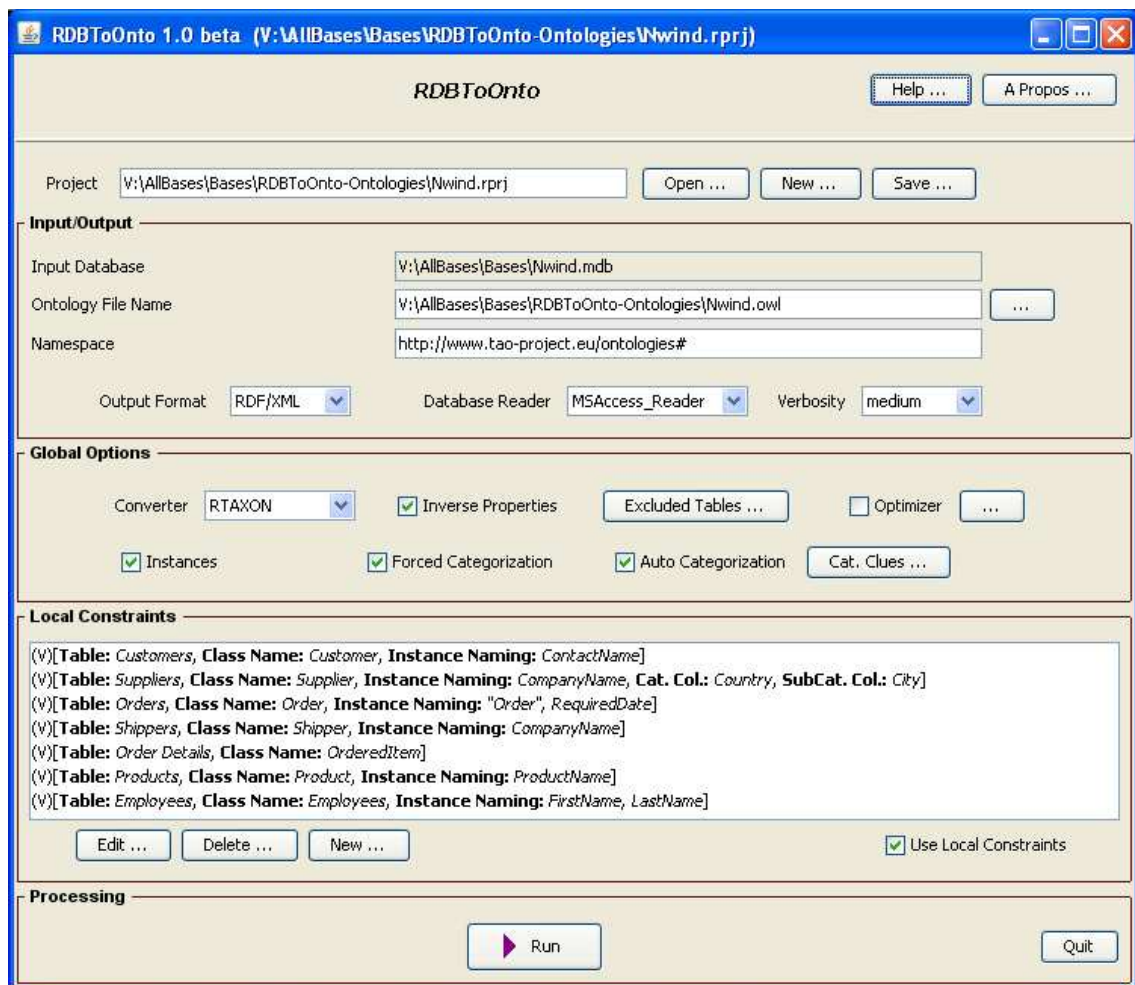


Figure 1: RDBToOnto main window

### 3 Conversion Projects

Each conversion task handled by RDBToOnto is represented by a *project* file that gathers all information related to the task at hand. This includes:

- The input database specification (complete path name and reader)
- The output ontology specification (complete path and format)
- Setting of the various global and local constraints

A project is an XML file with .rprj extension.

When starting a conversion task, a new project is created with default values.

## 4 Starting RDBToOnto

Run the command:

- RDBToOnto.bat on Windows
- RDBToOnto.sh on Linux

### Arguments

When starting the tool from a shell, arguments can be given to the script.

Usage: RDBToOnto [-h] [-gui|-nogui] [-cfg <F1.cfg>] [<F2.rprj>]

- -h: displays this outline of the arguments.
- -gui or -nogui : use the user interface or not.
- -cfg <F1.cfg>: use <F1.cfg> configuration file instead of the default one.
- <F2.rprj>: open <F2.rprj> project.

### Notes

- All arguments are optional.
- The -nogui option allows running the tool in batch mode without showing the user interface. With the -nogui option, project file argument is mandatory.

### Examples of launching commands

- RDBToOnto
- RDBToOnto.bat D:\Home\Conversion\MyProject.rprj
- RDBToOnto.sh -cfg D:/RTO/Conf/MyConf.cfg MyProject.rprj

## 5 Preparing the input data

RDBToOnto includes database automated readers for MySQL, Oracle, Microsoft Access databases and Excel spreadsheets.

Minimal preparation is required for these types of input data:

- For MySQL (see appendix A.1) and Oracle (see appendix A.2), a server should be running providing access to the databases to process.
- For Microsoft Access databases, data read permission should be given to table "MSysRelationships" (see appendix A.3).
- For Excel, "tables" in the spreadsheets should be clearly delimited (see appendix A.4).

## 6 Creating a new project

To create a new project:

- Click on Open button (or Ctrl + N).
- A “Select Input Database” panel will be displayed.
- In this panel, the database file can be provided either by typing full path name or database identifier through the “Input Database” text field or by selecting the database file (click on “...” button to get a file explorer).

If required, login/password information can also be entered from this panel.

### Examples of database identifiers for MySQL:

- jdbc:mysql://localhost/world
- jdbc:mysql://mysql.comp.ac.uk:4085/product\_database?user=aUser&password=aPass
- Or jdbc:mysql://mysql.comp.ac.uk:4085/product\_database and login information entered separately through the interface.

### Example for Oracle:

- To process database with name HR and password HR, enter: jdbc:oracle:thin:@localhost in Input Database field, HR in User field, and HR in Password field.

(see also appendix A.2)

- Once the database is selected, just press Ok button.

RDBToOnto will attempt to get access to the database. If this attempt is successful, a number of initialization operations are performed:

- **Reading of the database schema**

Elements of the schema are now made available and can be manipulated through the interface (to set constraints on tables).

- **Assignment of file names to the project and output ontology**

Both project and ontology file names are derived from the database name.

The names and locations pre-assigned to the project and the ontology can be changed by the user through the interface, before starting the generation process.

The default behavior for selecting the output directory of the created files is:

- if the path does not refer to a local file (i.e. not a file system-based path name), then all data are created in the default output directory. In the initial configuration of RDBToOnto, this is the directory named “output” located just under RDBToOnto main directory. See below how to change the default output directory.

This case applies to MySQL and Oracle databases.

- If the provided path refers to a file in a local directory, the proposed output location is an “RDBToOnto-Ontologies” directory which will be created in the directory that contains the database. See below how to always put the generated data in the default directory.

#### **Changing the default output directory**

Default output directory can be changed by adding an “outputDir” element in the RDBToOnto configuration file ( “RDBToOnto.cfg” file in “conf” directory under RDBToOnto main directory).

Moreover, if “always” attribute is set to “yes”, it will always be proposed to put the data in the default output directory (i.e, RDBToOnto will never try to derive a location from the database path name).

**example:** <outputDir always="yes">C:\temp</outputDir>

#### **Note**

When creating a new project, the extension or prefix (eg, ”jdbc:mysql:” for MySQL) of the database full path is used to determine the appropriate reader. The reader to apply can be changed from the interface.

## **7 Opening a project**

To open a project:

- Enter full path name of the project in the ”Project” field or select the file with a file explorer by clicking on ”...” button (or Ctrl + O).

## **8 Saving the project and the generated ontology**

The project settings are saved in the project file given in ”Project” field by clicking on Save button (or Ctrl + S).

No specific action is needed to save the ontology as it is systematically saved at end of each generation process run.

The user can select the output format from the interface (“Output Format” list in “Input/Output” section of the main window).

## **9 Running the process**

The process is started by clicking on Run button in Processing section of RDBToOnto main window (or Ctrl + O).

A waiting windows is displayed. The process can be interrupted (at the end of main step which is underway) by clicking on the Stop button of the waiting window.

## 10 Feedback on work progress

Details on work progress are given in the running window. When dealing with relatively large databases, some steps can take substantial time, such as reading of the data and ontology population. It could be useful to follow the progress of the process from this window.

From the user interface, “Verbosity” list can be used to the set the message level (the three possible values are low, medium and high).

## 11 Constraining the process

Creation of a project is the only mandatory preliminary step to ontology generation. Once a project has been successfully created from a database, ontology generation can be run to get an ontology.

However, the resulting ontology can be significantly improved by setting the **global options** and adding **local constraints** to provide specific indications on how input tables should be handled.

Global options (see 11.5) allow to:

- Select a converter (see 11.1)
- Decide whether inverse properties should be added (see 11.3)
- Populate the ontology or not (see 11.2)
- Exclude tables (see 11.4)
- Set global categorization options (see 11.5)
- activate/deactivate all local constraints (see 12)
- Decide whether a database optimization step should be included in the process (see 13)

With local constraints attached to tables the user can (see 12):

- Assign names to classes (see 12.1.2)
- Specify how instances of a derived class should be name (see 12.1.2)
- Exclude columns (see 12.1.4)
- Set local categorization options on specific tables (see 12.1.5).

## 11.1 Selecting the converter

The displayed local options depend on the selected converter.

Two converters are included:

- The **RTAXON converter** which implements a comprehensive method allowing to generate populated ontologies with rich taxonomies induced from the table data.
- The **basic converter** that can be exploited to derive an ontology without instances and categorization (can also be seen as a shortcut to a minimalist configuration of the RTAXON converter).

Selecting a converter updates the global option panel to show the options that are relevant for this converter.

Available converters are defined in the configuration file.

RDBToOnto is designed to allow the integration of new converters. This extension capability is described in the development guide.

## 11.2 Populating the ontology

With RTAXON converter, check the Instances box in "global Options" section to populate the ontology.

Ontology population may require substantial processing time on large databases. The overall cost of this step also depends on the complexity of the ontology model and local constraints assigned to tables, and more particularly on the patterns defined to name instances.

Note that, with the default configuration, instances of a given class are systematically added to its parent classes. It is possible to leave that propagation task to an external reasoner by setting "withInstancesInSuperClasses" property to "no" in RDBToOnto configuration file.

## 11.3 Adding inverse properties

When this option is selected, an inverse property is systematically assigned to each derived object property.

If the ontology is populated, inverse properties are also instantiated.

## 11.4 Excluding tables

A view to manage (display, exclude, restore) tables is obtained by clicking on "Excluded Tables" button.

### 11.5 Global categorization options

Categorization is a prominent feature of RDBToOnto, and more specifically of the RTAXON method.

Two global categorization options are provided:

- **Automated categorization:** If selected, mining of potential categorization schemes will be performed in order to automatically identify class hierarchies.

This option is activated/deactivated through "Automated Categorization" toggle switch in "Global Options" section of RDBToOnto main window.

- **Forced categorization:** The user has the ability to impose specific categorization schemes in tables through local constraints (see section 12.1.5).

This option is activated/deactivated through "Forced Categorization" toggle switch in "Global Options" panel.

If the "Forced Categorization" option is not selected, the process will ignore all local categorization constraints set by the user.

#### Note

The identification of relevant categorization sources involves **categorization clues**. A list of common clues are pre-defined in the RDBToOnto configuration file and project-specific clues can be added using the categorization clues editor.

To open this editor, click on "Cat. Clues ..." button in "Global Options" section of RDBToOnto main window. The editor allows:

- Displaying the common clues
- Displaying/Adding/Removing specific clues
- through "Override Clues" toggle switch, specifying if the new clues are intended to override the common clues or if both sources should be jointly considered.

Specific clues are saved in the project file.

## 12 Adding local constraints

Local constraints allow to assign precise prescriptions on tables to further refine the definition of the resulting classes and instances. The defined constraints are displayed in the "Local Constraints" section of RDBToOnto main window.

Flexible editing of the local constraints allows the user to iteratively experiment various processing configurations. Before starting the generation process, it is possible to involve or ignore all or some of the defined local constraints:

- "Use Local Constraints" toggle switch in "Local Constraints" section can be used to involve or ignore the constraints.

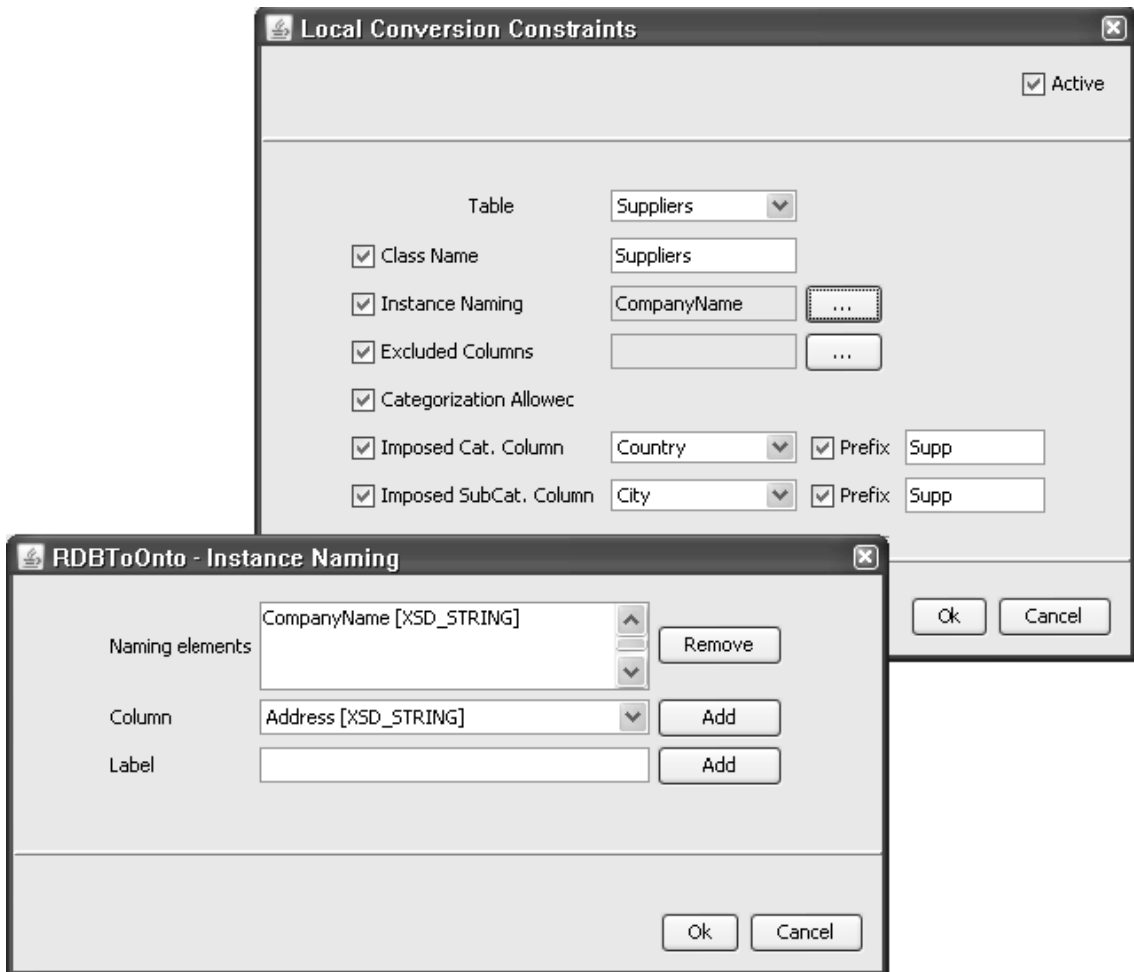


Figure 2: A view of a local constraint panel and of the related instance naming panel

- The constraints can be partially inhibited (see 12.1.1).

## 12.1 Creating a local constraint

To create a new local constraint:

- Click on "New ..." button in "Local Constraints" section.
- A "Local Conversion Constraints" panel is displayed (figure 2).
- Select an item in "Table" list.
- set the constraints (see below).

### 12.1.1 Activating and deactivating local constraints

Local constraints assigned to a table can be activated/deactivated through the Active toggle in "Local Conversion Constraints" panel.

Additionally, to enable partial activation/deactivation of the local constraints, a specific toggle is assigned to each constraint (Class name, Instance Naming, ...) – see figure 2.

In the "Local Constraints" section (RDBToOnto main window) showing all the local constraints, each constraints is prefixed with a symbol indicating its status:

- (V): The constraints on the concerned table are all active.
- (X): The constraints are all inactive.
- (P): Some of the defined constraints are inactive.

### 12.1.2 Setting class name

When selecting a table, the class name field is filled with the table name.

The class name can be adjusted manually.

### 12.1.3 Instance naming

A dedicated editor is obtained by clicking on "..." button aligned with the "Instance Naming" field.

Instance names can be defined by combining content of columns and "constant" labels:

- columns: Items from the "'column'" list can be added in the naming pattern by clicking on corresponding Add button.
- labels: String from the "'column'" list can be added in the naming pattern by clicking on corresponding Add button.

"\_" characters will be systematically inserted between parts of the names composed by applying the naming patterns.

#### Note

Reference to a column of an external table is possible. However, the external table should be linked through a foreign key relationship with the source table.

This additional functionality requires entering a little more complex expressions through the "Label" field.

The typical form is: `ex:<FK>:<DstTable>:<DstColumn>`

- the expression should start with the "ex:" string
- <FK> is the name of the foreign key attribute that refers to the destination/external table

- <DstTable> is the name of the destination table
- <DstColumn> is the name of the destination attribute

**Example:** ex:CustomerID:Customers:ContactName attached to an Orders table.

An instance naming constraint combining label "order" and the above expression allows generating instances with names like "order\_Birgit\_Lang", "order\_Arturo\_Brescia\_1", "order\_Arturo\_Brescia\_2", etc.

#### 12.1.4 Excluding columns

A view to manage (display, exclude, restore) columns is obtained by clicking on corresponding "..." button.

#### 12.1.5 Setting local categorization constraints

Through "Categorization Allowed" toggle button, categorization on the concerned table can be allowed or inhibited. This option applies to both automated and forced categorization (11.5).

If the user wants to apply forced categorization on a specific table, a categorization column and optionally a sub-categorization column should be selected from "Imposed Cat. Column" and "Imposed SubCat. Column" lists respectively (see figure 2). Furthermore, prefixes can be defined through the Prefix fields. The defined prefixes will be added to names of the generated subclasses.

## 13 Database optimization

To ensure that the target ontology will capture all relational aspects of the source data, it could be appropriate to involve the database optimization component in the transformation process. The optimizer provides an editor to define *inclusion dependencies* and applies them to reduce redundancies by adding new primary key/foreign key relationships<sup>1</sup>. Typically, an inclusion dependency holds when data from a *source table* is exploited to feed columns of another table, called *including table*.

**Caution:** Though tested on several databases, this component should be considered being at the stage of "Alpha" version and the data should be carefully checked when this optimizer is invoked. Behavior of the optimizer when dealing with wrong or partial user defined dependencies has not been investigated in depth yet.

### 13.1 Editing inclusion dependencies

To start the editor, click on "..." button located on the left side of "Optimizer" check box in "Global Options" section.

---

<sup>1</sup>First version of the optimizer user interface was implemented by Matthieu Beyou.

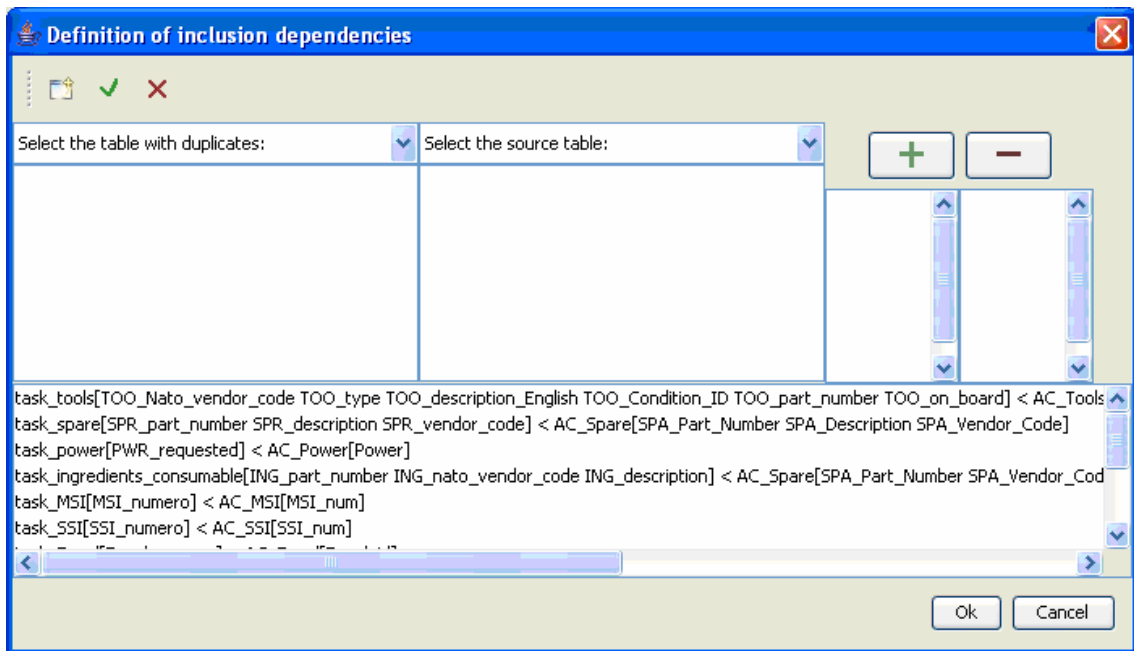


Figure 3: Inclusion dependency editor

It is probably easier to understand the usage of this editor through the provided animated demo (<http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto-demobis.html>).

In the displayed editor (figure 3):

- the top-left buttons respectively correspond to “Creation of a new dependency”, “Validation of current dependency”, and “Deletion of current dependency”.
- The two lists in the left part can be used to select the including table and the source table.
- The two lists in the right part will include the attributes pairs from including tables and source table that are relevant for the dependency under definition.
- The list at the bottom is the defined inclusion dependencies.
- When a table is selected, either including table or source table, its attributes are displayed in the corresponding list.
- Pairs of attributes from the displayed including table/source table attributes can be selected and, by clicking on “+” button, the pairs are added in the lists of relevant attribute pairs.
- The definition of the inclusion dependency is validated by clicking on the dependency validation button (second button in the top-left button panel).

### **13.2 Including optimization step in the process**

To include an optimization step in the process, check the “Optimizer” box in “Global Options” section.

#### **Note**

In a session on a given project, the optimization is run only once, even though the process can be run several times (for instance, with different constraint settings). However, if new inclusion dependencies are added, the project should be saved and reload before running the process again.

## A Preparing the data for the readers

### A.1 Getting data from MySQL databases

The only requirement is a running MySQL server providing access to the databases to process.

A typical database identifier starts with "jdbc:mysql:".

#### Examples:

- jdbc:mysql://localhost/world
- jdbc:mysql://mysql.comp.ac.uk:4085/product\_database?user=aUser&password=hiddenPass
- Or jdbc:mysql://mysql.comp.ac.uk:4085/product\_database and login information entered separately through the interface.

### A.2 Getting data from Oracle databases

This is an alpha version of the reader which has been successfully tested on a limited number of databases (not tested yet on Linux/Unix systems). Specific features of Oracle like object and spatial types are not supported in this version.

The only requirement is a running Oracle server providing access to the databases to process.

A typical database identifier starts with "jdbc:oracle:thin:".

#### Examples:

- To process database with name HR and password HR, enter through the Select Input Database Panel (which appears after clicking on New ... Button):
  - jdbc:oracle:thin:@localhost in Input Database field,
  - HR in User field,
  - HR in Password field.

Other examples of Oracle database Urls:

- jdbc:oracle:thin:hr/hr@localhost
- jdbc:oracle:thin:hr/hr@//testserver.oracle.com

The login information should always be provided.

### A.3 Getting data from Microsoft Access databases

All is needed is to give read data permission to the MSysRelationships system table:

#### Step 1: Display system tables

- Open the input database with MS Access.
- In Tools menu, choose the Options entry.
- Within the Options dialog, select the View tab and look for the Show section.
- Check the box next to the System objects option and close the options dialog. System tables should now be displayed.

#### Step 2: Grant access permission to MSysRelationships table

- Choose the User and Group Permissions entry from Security sub-menu of Tools menu.
- Enable Read Data permission for MSysRelationships table.

The database can now be handled by the “MSAccess\_Reader” reader.

Note that, for MS Access, another reader based on XML exports is provided in the the samples directory (see development guide for details).

### A.4 Getting data from Excel spreadsheets

The (MSExcel\_Reader) reader<sup>2</sup> handles MS Excel documents with multiple spreadsheets.

**Caution:** To be properly extracted by the reader, **the input should be carefully prepared**. Tables in spreadsheets should be clearly delimited:

- No data should be included in cells outside the table zone (e.g. no title to introduce the table).
- The first line of the table should be for the names of the columns.
- The input document can include several spreadsheets but each spreadsheet should contain only one table.

---

<sup>2</sup>This reader is based on Apache POI (<http://poi.apache.org>).