

Rational Data Architect skills series, Part 3: Discover schema relationships with Rational Data Architect

Create schema mappings semi-automatically

Skill Level: Intermediate

[Torsten Bittner \(tbittner@us.ibm.com\)](mailto:tbittner@us.ibm.com)
Software Engineer
EMC

19 Dec 2006

You can use IBM® Rational® Data Architect to define data mappings. When working with large schemas, it can be very cumbersome to manually create mappings. Rational Data Architect offers a discovery component to semi-automatically identify potential mappings. This tutorial provides an introduction to the relationship discovery component of Rational Data Architect.

Section 1. Before you start

In this tutorial learn how to use the discovery component of Rational Data Architect to semi-automatically create mappings between relational and XML data sources. This tutorial is the third in a [series](#) about Rational Data Architect.

About this tutorial

This tutorial shows you, step by step, how to:

- Invoke the Rational Data Architect (RDA) lexical similarity discovery algorithm to discover potential schema mappings based on the similarity

of column names.

- Define a glossary model that contains words, abbreviations, and synonyms using the RDA Glossary Model editor.
- Use the glossary model information with the semantic name algorithm to discover additional matches.
- Set up relationship discovery to use data samples.
- Find potential mappings with algorithms that use data samples.

Objectives

After taking this tutorial, you should be able to use the Rational Data Architect discovery component to find potential schema mappings.

Prerequisites

Product name change

On December 16th, 2008 IBM announced that as of Version 7.5.1, Rational Data Architect is renamed to [InfoSphere Data Architect](#) to feature its role in InfoSphere Foundation Tools.

This tutorial assumes familiarity with relational databases, preferably DB2®. Familiarity with the Rational Data Architect Mapping Editor is beneficial, but not required. For reference, consult these developerWorks articles:

- Part 1: [Access and integrate enterprise metadata with Rational Data Architect](#)
- Part 2: [Generate SQL/XML queries with Rational Data Architect](#)
- [Use Rational Data Architect to integrate data sources](#)

System requirements

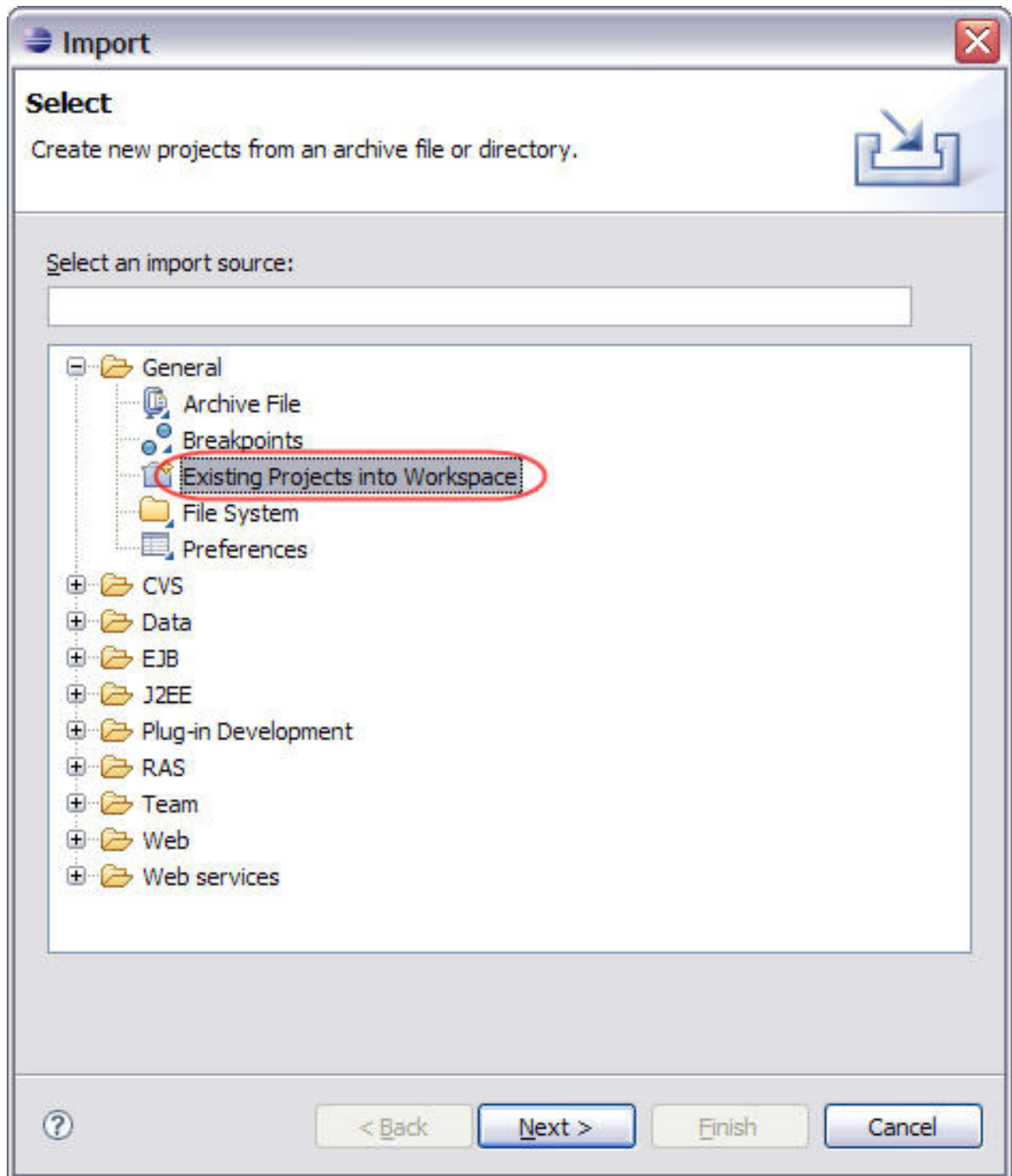
To execute the steps described in this tutorial, you need to have Rational Data Architect V7 and IBM DB2® Enterprise 9 installed. You can download trial versions of IBM Rational Data Architect V7 and DB2 V9.1 (see [Resources](#)).

Setup steps

1. Install [DB2 V9.1](#)..

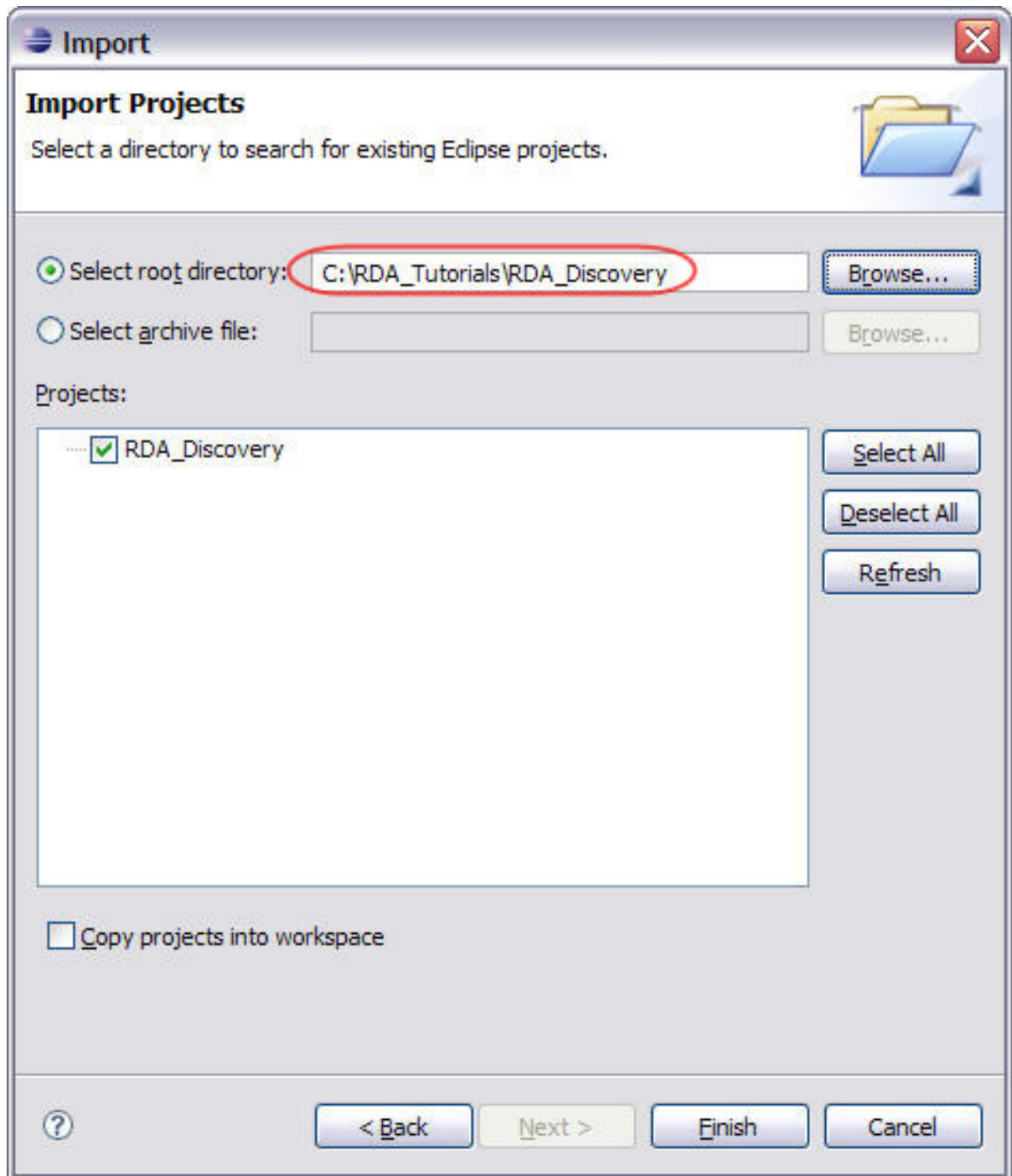
2. Install [Rational Data Architect V7](#).
3. Unzip the package, [ar-rdamapcode.zip](#) from the file into a folder (for example, C:\RDA_Tutorials). This step creates the RDA_Discovery folder.
4. Start Rational Data Architect and specify the folder where you unpacked the package as the location for your workspace (for example, C:\RDA_Tutorials).
5. The RDA_Discovery folder in the package is a Rational Data Architect data project folder. In Rational Data Architect you have to import it into your workspace. From the File menu, select **Import**.
6. Select the **Existing Project into Workspace** wizard.

Figure 1. Import wizard selection



7. Click **Next**. Browse to the location where you extracted ar-rdamapcode.zip (for example, C:\RDA_Tutorials).

Figure 2. Import Project wizard



8. Click **Finish**. As a result, you see the RDA_Discovery project with a set of data models, a glossary model, and mapping models files in your workspace as shown in [Figure 3](#). (If you don't see the Database Explorer, make sure that you are in the Data perspective, shown in [Figure 4](#).)

Figure 3. Data Project Explorer after project import

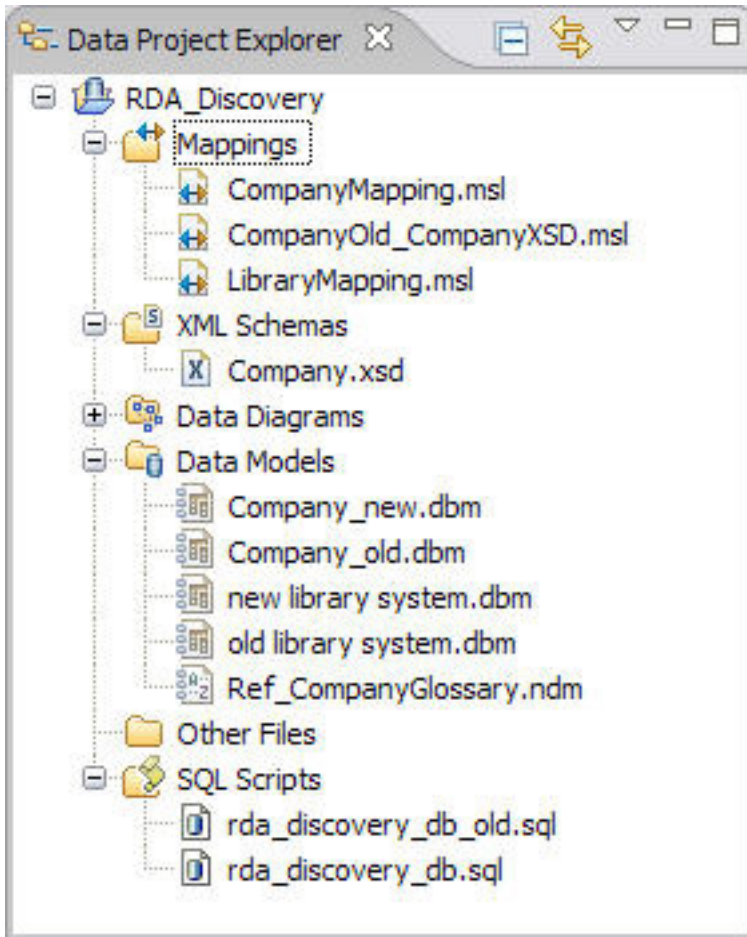
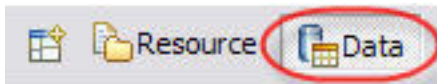


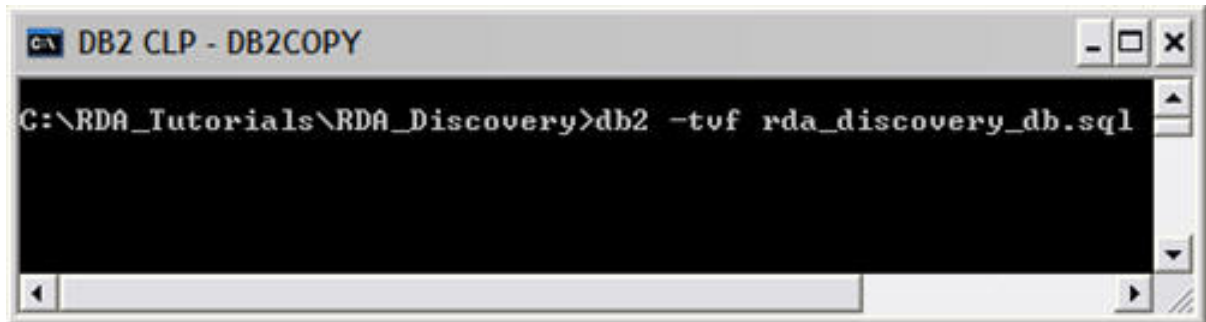
Figure 4. Data perspective



9. Some of the relationship discovery algorithms require sample data. The script file `rda_discovery_db.sql` creates a DB2 database LIBRARY and inserts a set of sample data. To deploy the script in your DB2 database, start the DB2 command window (Windows menu **Start > IBM DB2 > Command Line Tools > Command Window**).
10. Go to the RDA_Discovery folder that you extracted from `ar-rdamapcode.zip`.
11. To create the database LIBRARY, define primary and foreign keys, and insert sample data, run this command:

```
db2 -tvf rda_discovery_db.sql
```

Figure 5. Creating LIBRARY database

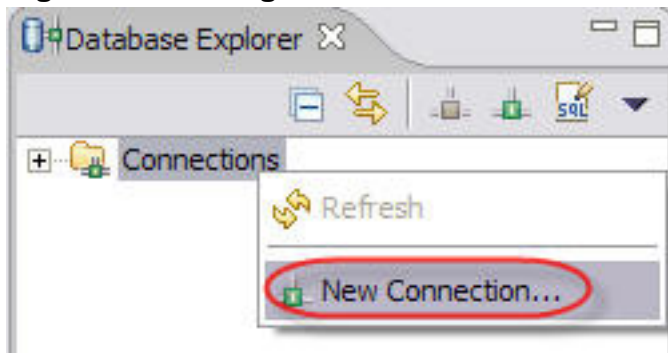


12. To create the database OLD_LIB, define primary and foreign keys, and insert sample data, run the command:

```
db2 -tvf rda_discovery_db_old.sql
```

13. Connect to the database LIBRARY in Rational Data Architect. In the Database Explorer, right-click **Connection** and select **New Connection**. (If you don't see the Database Explorer, make sure that you are in the Data perspective, shown in [Figure 4](#).)

Figure 6. Creating new database connection



14. Specify the connection information according to your environment, similar to [Figure 7](#).

Figure 7. Database connection settings

The screenshot shows the 'Connection identification' dialog box. The 'Connection Name' is 'LIBRARY'. The 'JDBC driver' is 'IBM DB2 Universal'. The 'Connection URL details' section shows 'Database: LIBRARY', 'Host: localhost', and 'Port number: 50000'. The 'User information' section shows 'User ID: tbittner' and 'Password: *****'. A 'Test Connection' button is at the bottom.

Connection identification

Use default naming convention

Connection Name: LIBRARY

Select a database manager:

- 5.1
 - DB2 UDB
 - V7.2
 - V8.1
 - V8.2
 - V9.1
 - DB2 UDB iSeries
 - DB2 UDB zSeries
 - Derby
 - Generic JDBC
 - Informix
 - MySql
 - Orade
 - SQL Server
 - Sybase

JDBC driver: IBM DB2 Universal

Connection URL details

Database: LIBRARY

Host: localhost

Port number: 50000

Use client authentication

JDBC driver class: com.ibm.db2.jcc.DB2Driver

Class location: C:\IBM\SQLLIB\java\db2jcc_license_cu.

Connection URL: jdbc:db2://localhost:50000/LIBRARY:re

User information

User ID: tbittner

Password: *****

Test Connection

15. Click **Test Connection** to check whether all parameters are set correctly. If the test is successful, click **Finish**.
16. Repeat steps 13 to 15 for the database OLD_LIB using the same connection settings except for the database name.

Section 2. Scenario overview and problem description

The scenario for this tutorial is closely related to scenarios in previously published tutorials. The goal is to use the Rational Data Architect mapping editor for data integration (see [Part 1](#)) and query generation (see [Part 2](#)).

The mapping editor significantly reduces the amount of time it takes to manually write SQL or SQL/XML queries. The editor also allows the user to capture data relationships and publish the information in a report.

However, a new problem arises when using the mapping editor. The manual creation of each mapping, one after the other, is time consuming. Especially when working with large schemas on the source and target side, it is difficult to find the columns that are to be mapped.

Rational Data Architect addresses this problem with the relationship discovery component. The idea is to create mappings semi-automatically. The tool discovers potential mappings for the user, who only has to accept or reject the suggested match.

The discovery component works based on the assumption that source and target elements that are involved in a mapping have certain similarities. These similarities fall into two categories.

Similarity of the metadata

This category refers to the information contained in the data model, such as the physical database model, logical database model, or XML schema model. The elements that are mapped between these data models are columns, entities, and XML elements. In many cases the source and target element of a mapping will have the same name. Besides, the name of a source element could be an abbreviation or a synonym of a target element. The discovery algorithms in Rational Data Architect that are based on similarity of the metadata are:

- Lexical similarity
- Semantic name

Similarity of instance data

This category refers to the data that is contained, for example, in the column of a physical database that the user wants to map. The discovery component will gather a data sample from the physical data source and the target. It then compares the two data samples and attempts to discover a relationship. In Rational Data Architect the following algorithms are based on similarity of instance data:

- Signature
- Distributions
- Regular Expressions

This tutorial explains some of the algorithms and how to use them. For simplicity, this tutorial comes with a set of data models that are used to invoke the different algorithms (see [Download](#)).

The metadata-driven discovery uses the physical data models COMPANY_old.dbm and COMPANY_new.dbm. In a typical data integration scenario, one goal is to migrate data from an old schema into a newer schema. The column names and tables in this example are chosen to illustrate the capabilities of the discovery component, rather than simulating a schema as it is used in an enterprise environment.

For the data-driven discovery the database model "old library system.dbm" is used as a source. The physical database that contains the data is OLD_LIB. The target database model "new library system.dbm" is essentially the same as the source model. The difference is the instance data that is contained in the physical database LIBRARY, which is different compared to the data in OLD_LIB.

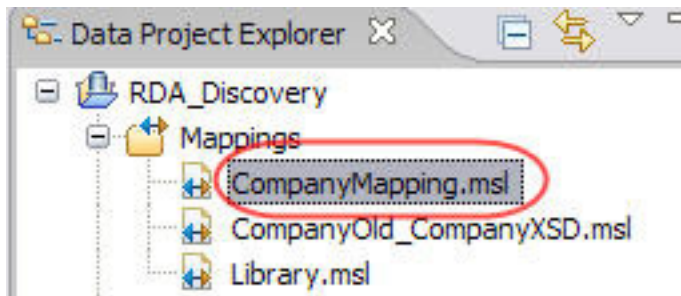
Section 3. Discovering mappings with lexical similarity algorithm

In this section, you'll invoke relationship discovery. Step by step we'll open an existing mapping model that has no mappings defined, run **Find Best Fit** and **Find Similar** discovery with the lexical similarity algorithm, and analyze the discovery results.

Invoking Find Best Fit discovery

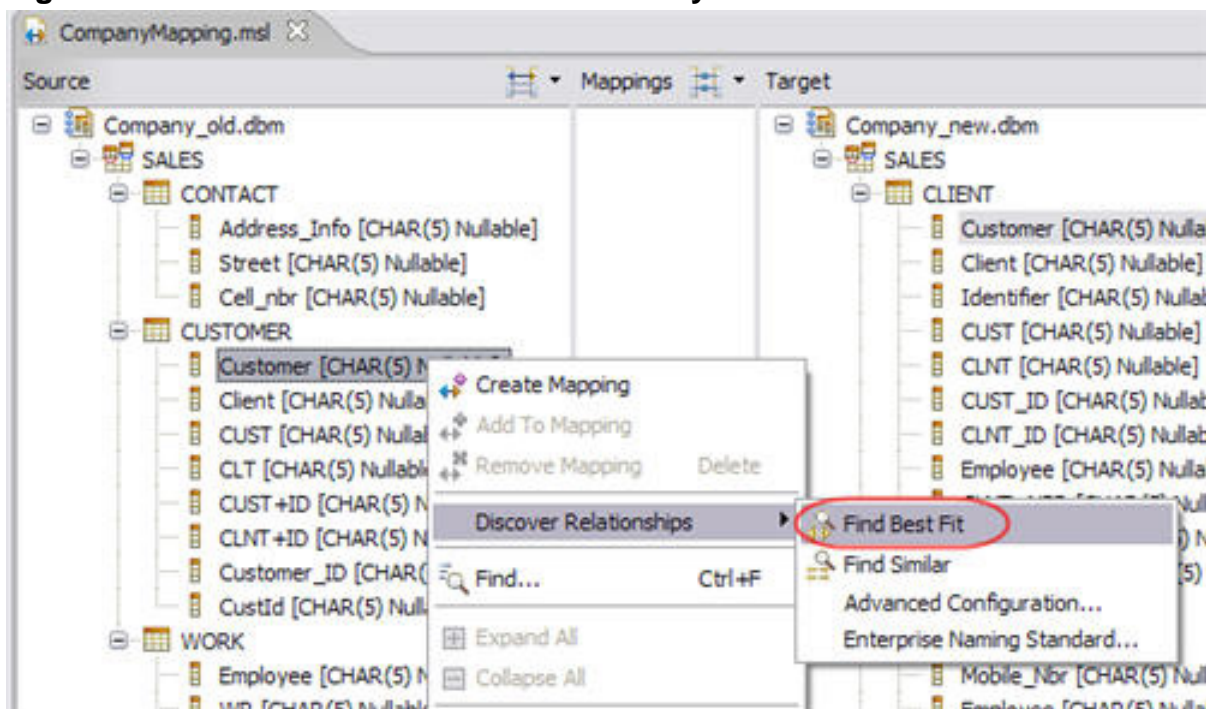
1. Change to the Data perspective.
2. In the Data Project Explorer, open the mapping model, **CompanyMapping.msl** as shown in [Figure 8](#). This opens up the Rational Data Architect mapping editor. On the left side is the source database model, Company_old.dbm, and on the right side is the target database model, Company_new.dbm.

Figure 8. Opening the CompanyMapping.msl mapping model



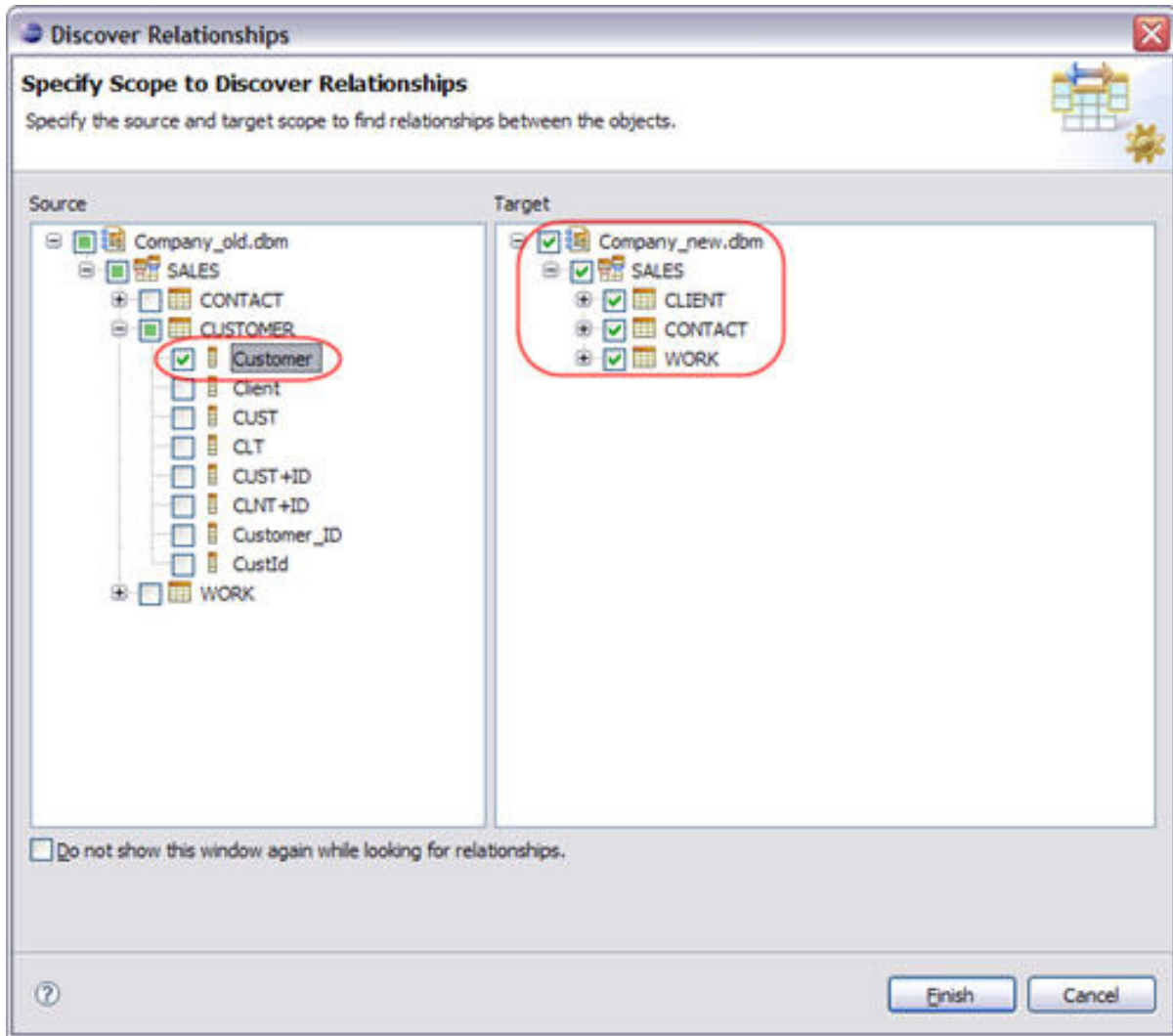
3. To invoke discovery, right-click on the column **Customer** in the table **CUSTOMER**. From the context menu, select **Discover Relationships > Find Best Fit**.

Figure 9. Invocation of Find Best Fit discovery



4. In the Discover Relationship scoping wizard, make sure that on the Source side the column **Customer** is checked and on the Target side all columns are checked. Click **Finish**.

Figure 10. Discover Relationship scoping wizard for Find Best Fit

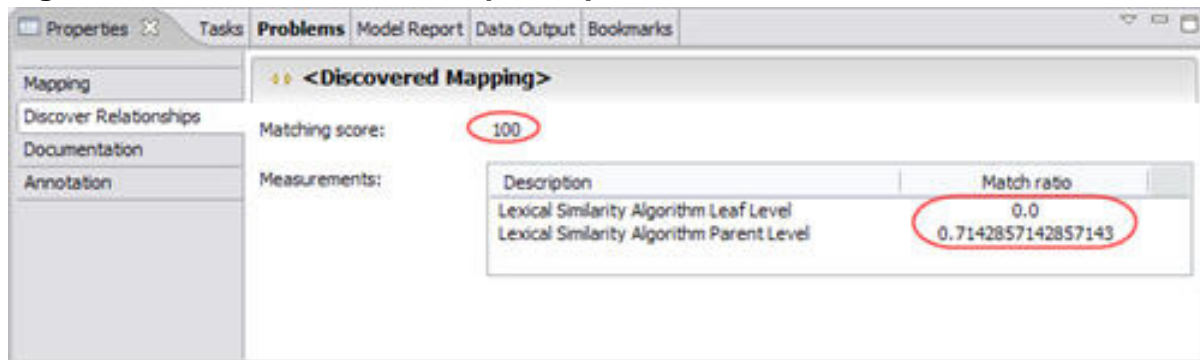


5. In the mapping editor you now see a yellow discovered line pointing from the source column **Customer** to the target column **Customer**. Hover the mouse over the yellow line and you see a matching score value of 100%.

Figure 11. Discovery matching score mouseover information



6. Right-click on the discovered line and select **Properties**. In the Properties View, open the **Discover Relationships** tab. In the discovery properties you find more detailed information about the discovered match.

Figure 12. Discover Relationships Properties view

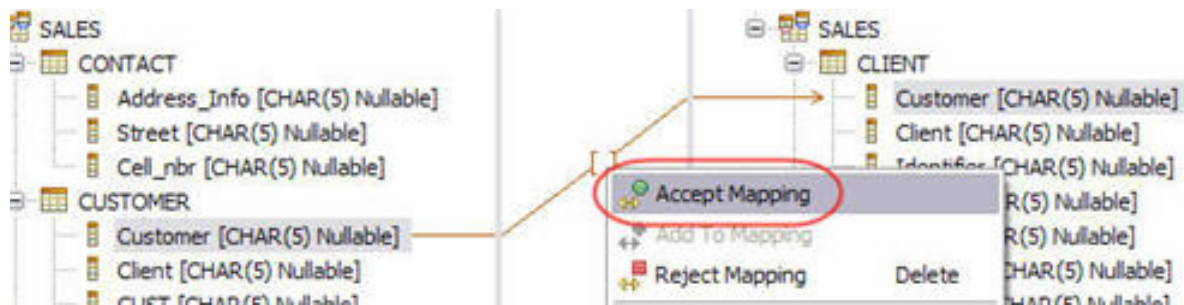
- The **Matching score** is the same that you saw when hovering your mouse over the mapping line. This value is relative compared to other matches that are discovered for the same source column. When more than one match is discovered for one source column, all matches are ranked. A higher ranking results in a higher matching score. More than one result can have the same matching score if their similarity to the source column is the same.

In the **Measurements** table in [Figure 12](#), the name of the discovery algorithm is shown together with a **Match ratio** value. The match ratio is an absolute value between 0 and 1 that shows the distance of the source and target column. Distance means that if two columns are very closely related, the value is low. When discovery identifies two columns as the same, the match ratio is 0. When two columns are very different, the match ratio is higher. A value of 1 means no match. The match ratio is used for the ranking that determines the matching score.

The table contains two match ratio values. The **Leaf Level** value refers to the similarity of leaf level elements in the schema tree view, such as column names when working with physical database models. **Parent Level** refers to the direct parent element of the leaf level element, such as the table name in the physical database model scenario.

- Right-click on the yellow mapping line in the mapping editor and select **Accept Mapping**. The yellow discovery line turns into a blue mapping line.

Figure 13. Accepting a discovered match



Invoking Find Similar discovery

Find Best Fit versus Find Similar

Find Best Fit usually gives you a jump-start on finding the best potential mappings. When invoking discovery with a big scope on source and target, Find Best Fit is the preferred choice. Otherwise the number of returned matches is too big and hard to handle in the mapping editor.

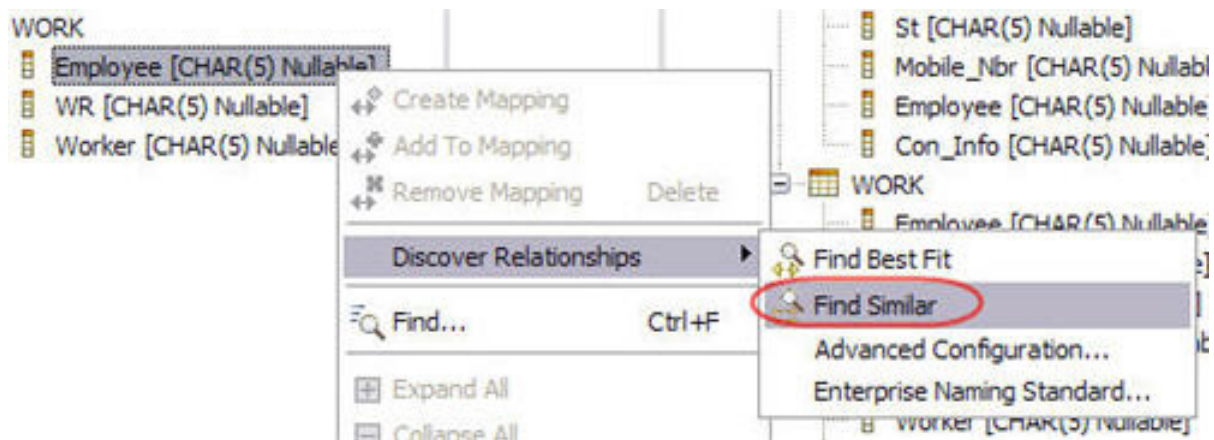
The downside of Find Best Fit is that you're potentially missing some very good mapping candidates. If you don't want to miss these candidates, but want to be able to handle the results, invoke Find Similar discovery with a small scope on the source side and a large scope on the target side.

Invoking relationship discovery using Find Best Fit returns the match with the highest matching score for the selected source column. Other matches, even if they have the same matching score, are automatically disregarded and do not show up in the mapping editor.

Typically you want discovery to return more than one result so you have the flexibility to choose from the suggestions. Invoking discovery using the Find Similar method lets you do this.

1. Right-click on the column **WORK.Employee**, and select **Discover Relationships > Find Similar** from the context menu, as shown in [Figure 14](#).

Figure 14. Invoking Find Similar discovery



2. In the discovery scoping wizard, make sure **WORK.Employee** under **Source** is selected, and that all elements under **Target** are selected. Set the additional scoping parameters to discover at most 5 elements **For each selected source element**. Click **Finish**.

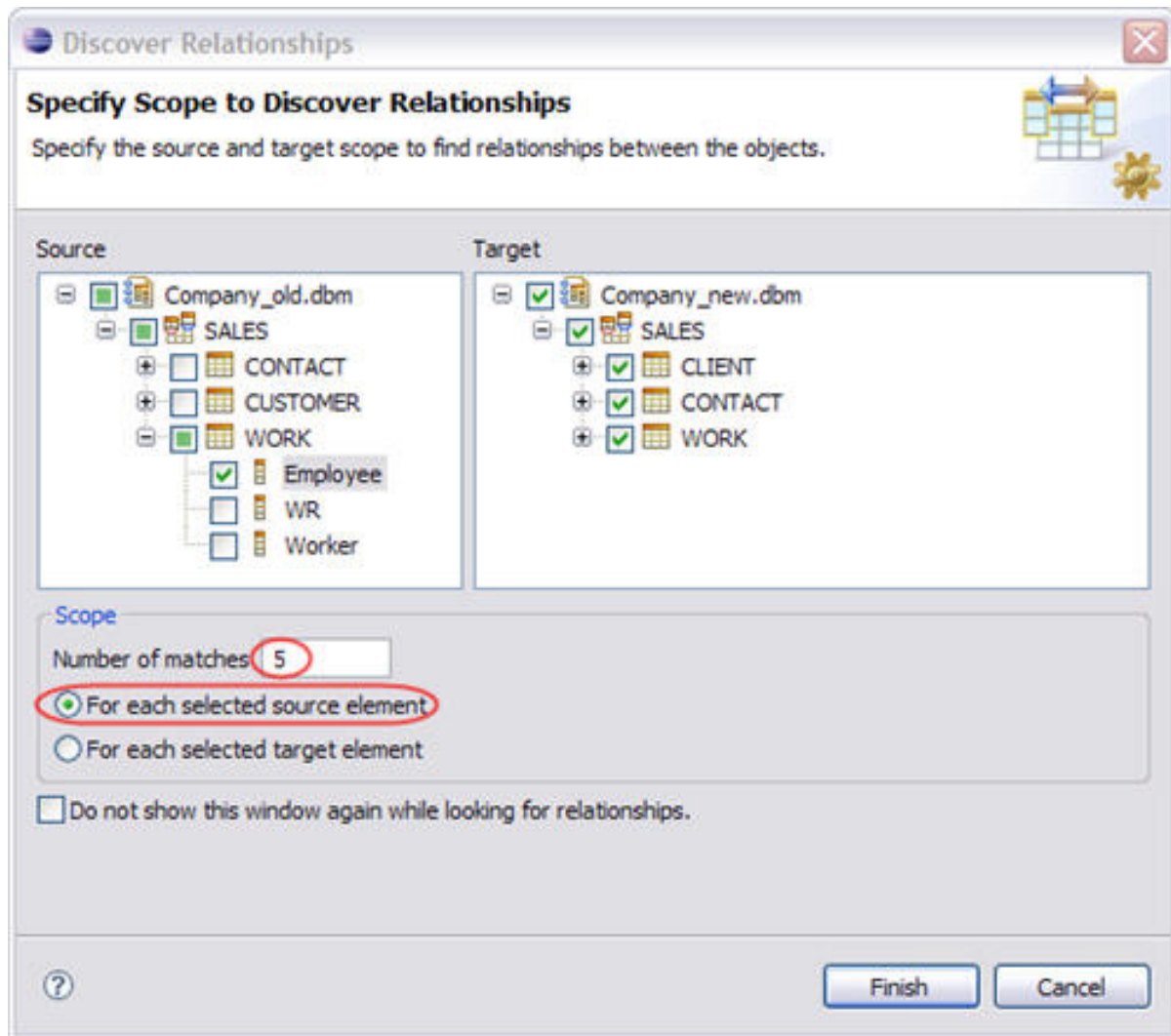
About scoping

It's important to wisely choose the scope for discovery invocation. It has a huge impact on the time of the discovery run and on the amount of returned results. The trade-off is between spending time defining the scope and spending time sorting out unwanted matches.

Good practice is to limit the scope on the side that drives your mapping and to have a wide scope on the opposite side. Let's say you want to migrate all the data in your enterprise to a certain schema. This schema is going to be your target schema. Since you want to fill every column in this schema, your mapping is target driven.

Good practice in this scenario is to invoke discovery for a subset of the target tables, sort out unwanted matches, and continue with the next set of tables.

Figure 15. Discovery Find Similar scoping wizard

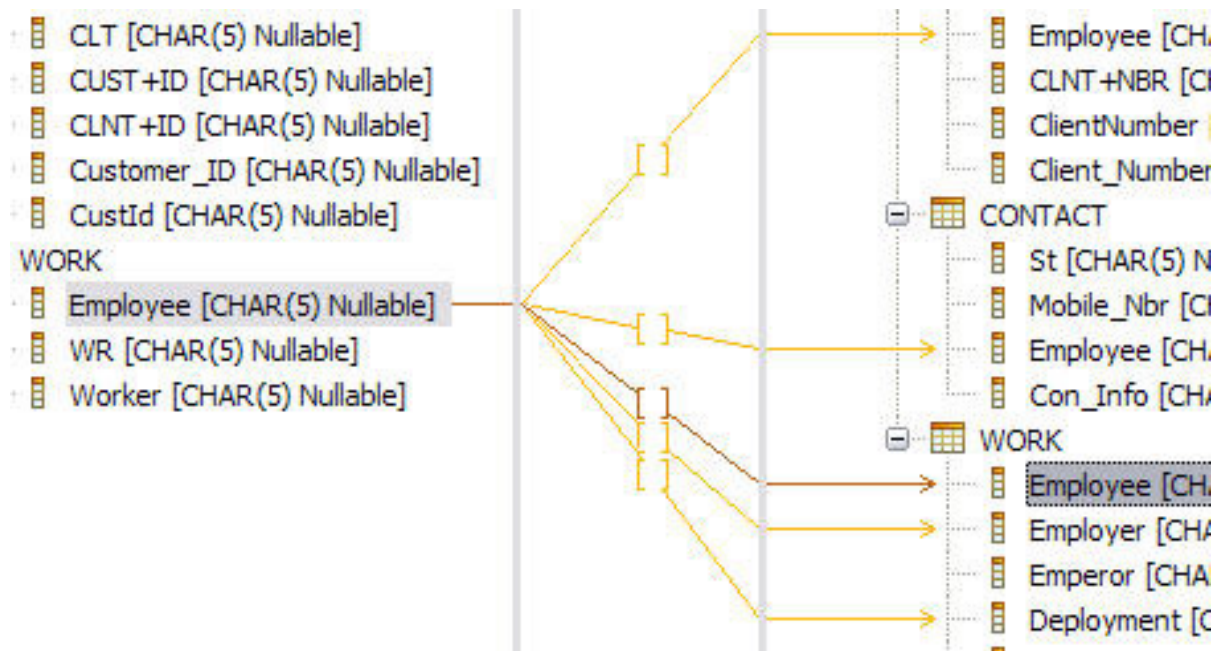


3. The following matches are returned:

- WORK.Employee**
- CONTACT.Employee**
- CLIENT.Employee**
- WORK.Employer**
- WORK.Deployment**

The different matching scores of the three Employee columns result from their different table names (parent-level match ratio).

Figure 16. Discovery Find Similar results

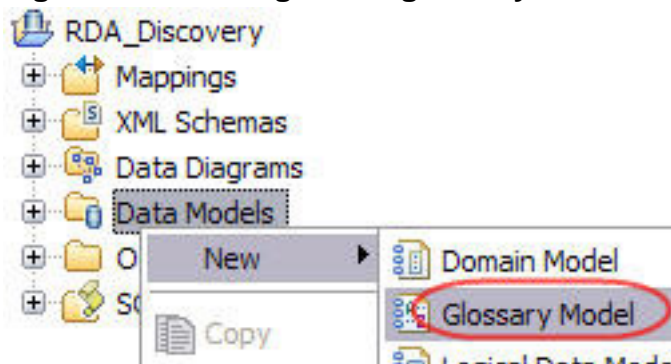


Section 4. Creating a glossary model

In this section you use the Rational Data Architect Glossary Model editor to create a glossary model. This model defines words with abbreviations and synonyms. The information is used to find additional matches in the succeeding section.

1. Right-click the **Data Models** folder in the Data Project Explorer. Select **New > Glossary Model** from the context menu.

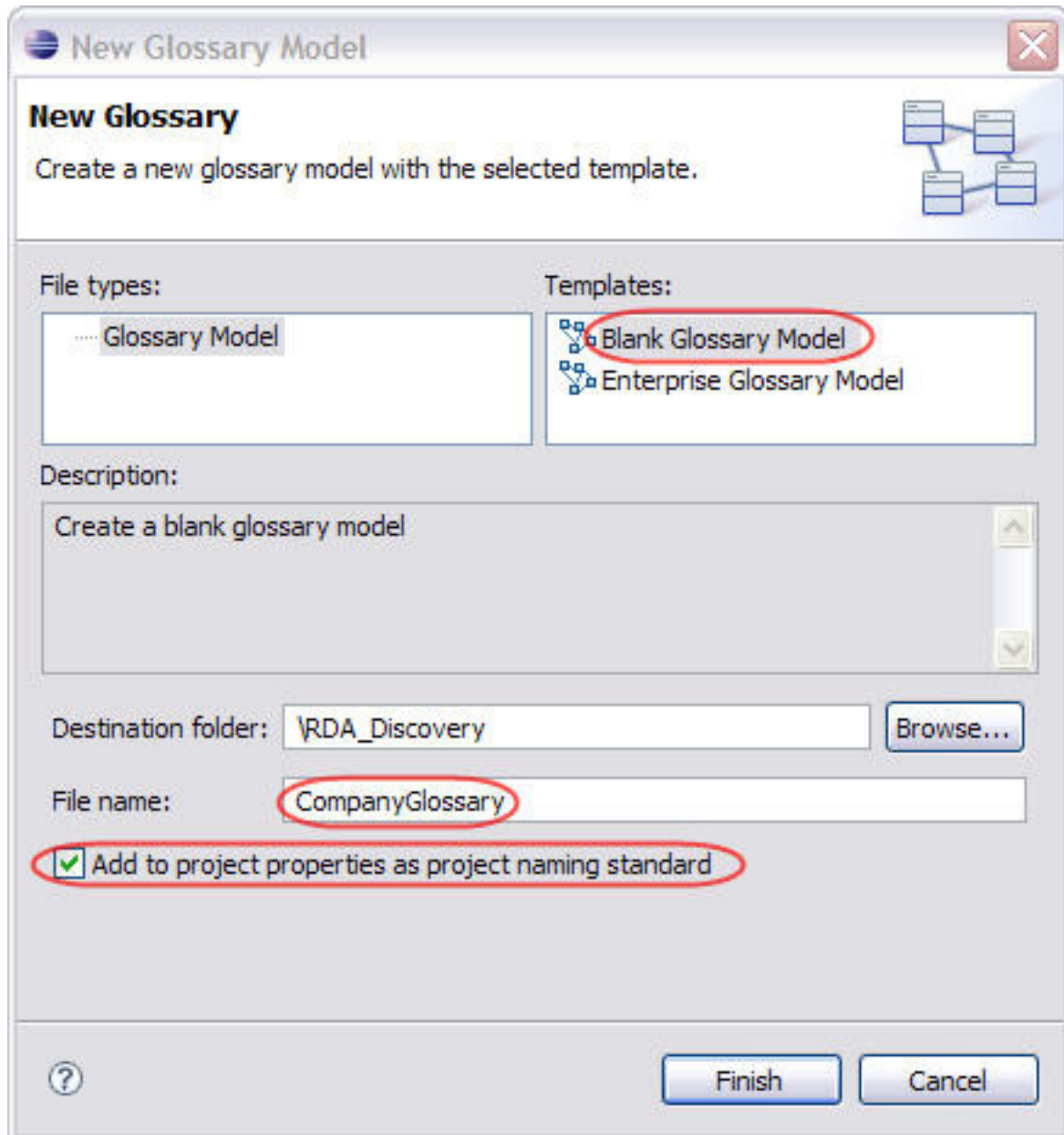
Figure 17. Creating a new glossary model



2. In the New Glossary Model Wizard, make sure to select the template

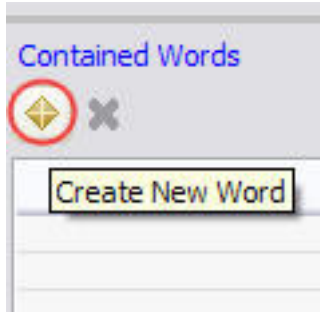
Blank Glossary Model and type `CompanyGlossary` for the file name.
Check **Add to project properties as project naming standard**.

Figure 18. Glossary Model Wizard



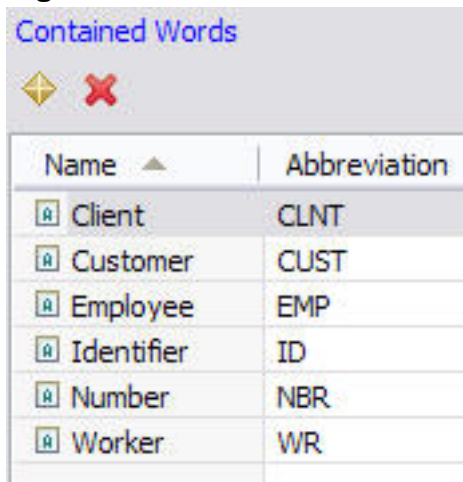
3. Click **Finish**. The glossary model editor opens with an empty glossary model.
4. In the **Contained Words** section, click **Create New Word**.

Figure 19. Creating a new word



5. Add the following words with their abbreviations to the glossary model.

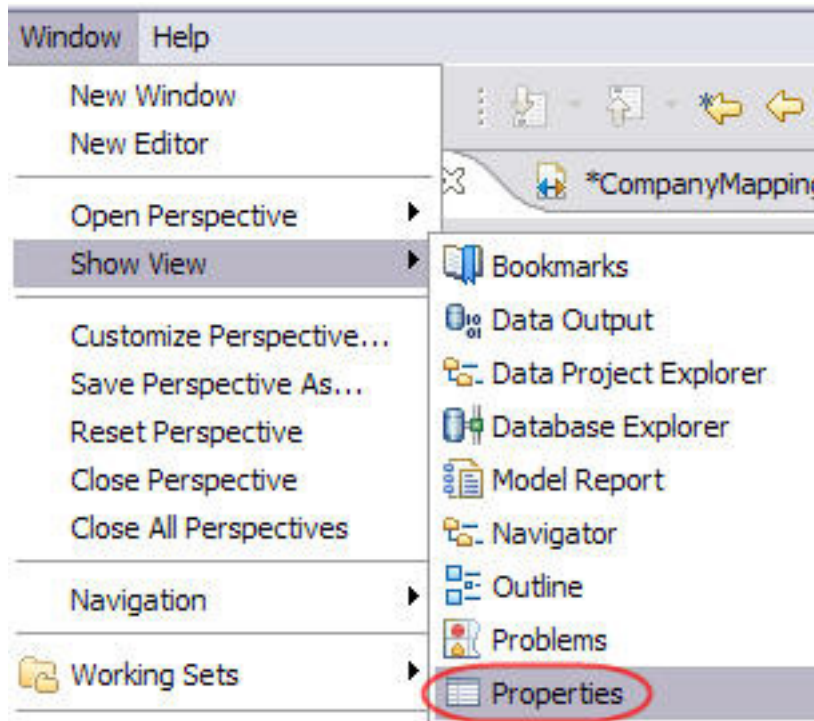
Figure 20. Words with abbreviations in glossary model



Name ▲	Abbreviation
Client	CLNT
Customer	CUST
Employee	EMP
Identifier	ID
Number	NBR
Worker	WR

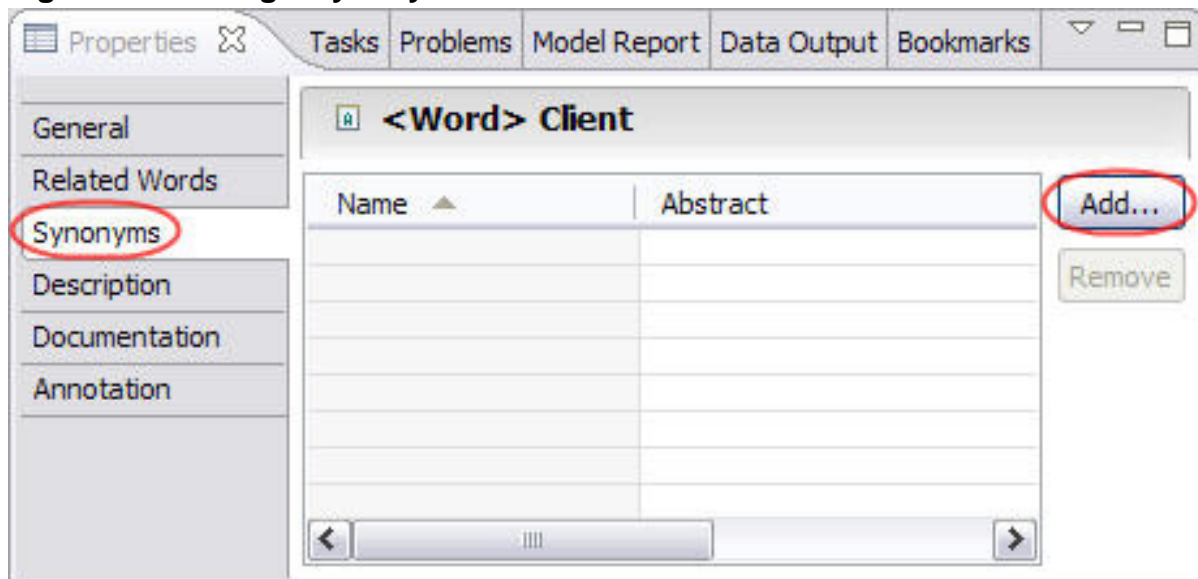
6. Open the Properties view (Menu **Window > Show View > Properties**).

Figure 21. Opening the properties view



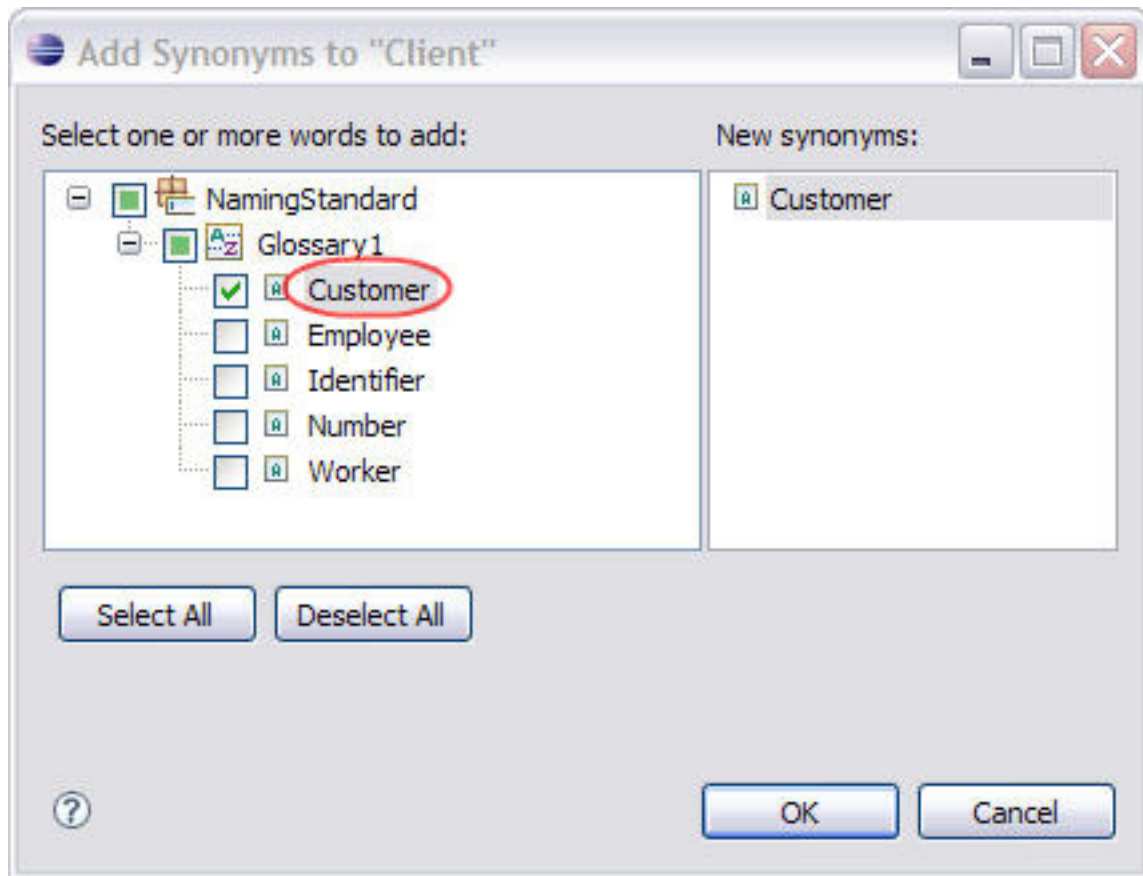
7. Click on the word **Client** to select it in the glossary model editor. In the properties, view select **Synonyms**, then click **Add**.

Figure 22. Adding a synonym to a word



8. Check the word **Customer**, and click **OK**.

Figure 23. Selecting a synonym



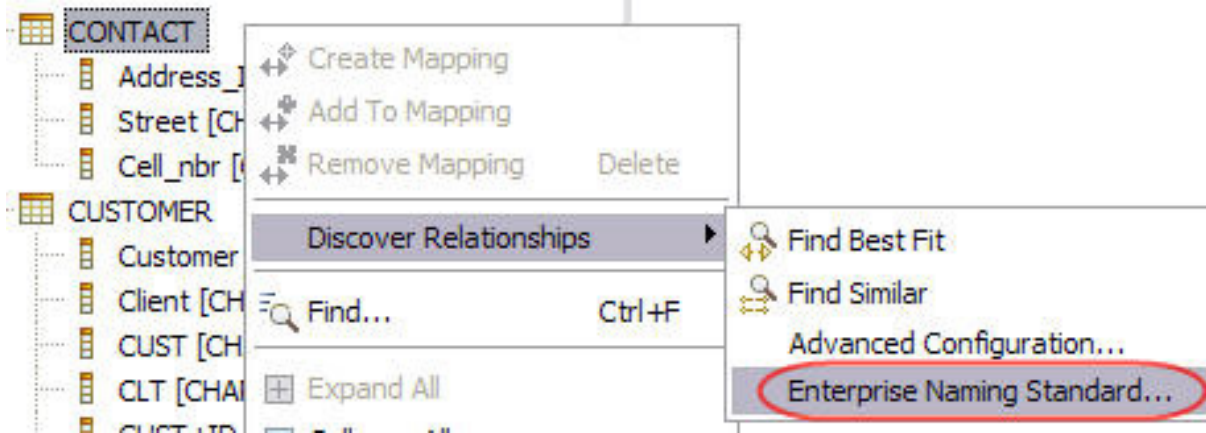
9. Repeat steps 7 and 8 to add the synonym relationships Employee/Worker and Identifier/Number.

Section 5. Discovering mappings using abbreviations

In this section you use the abbreviations from the glossary model, CompanyGlossary.ndm, to discover relationships.

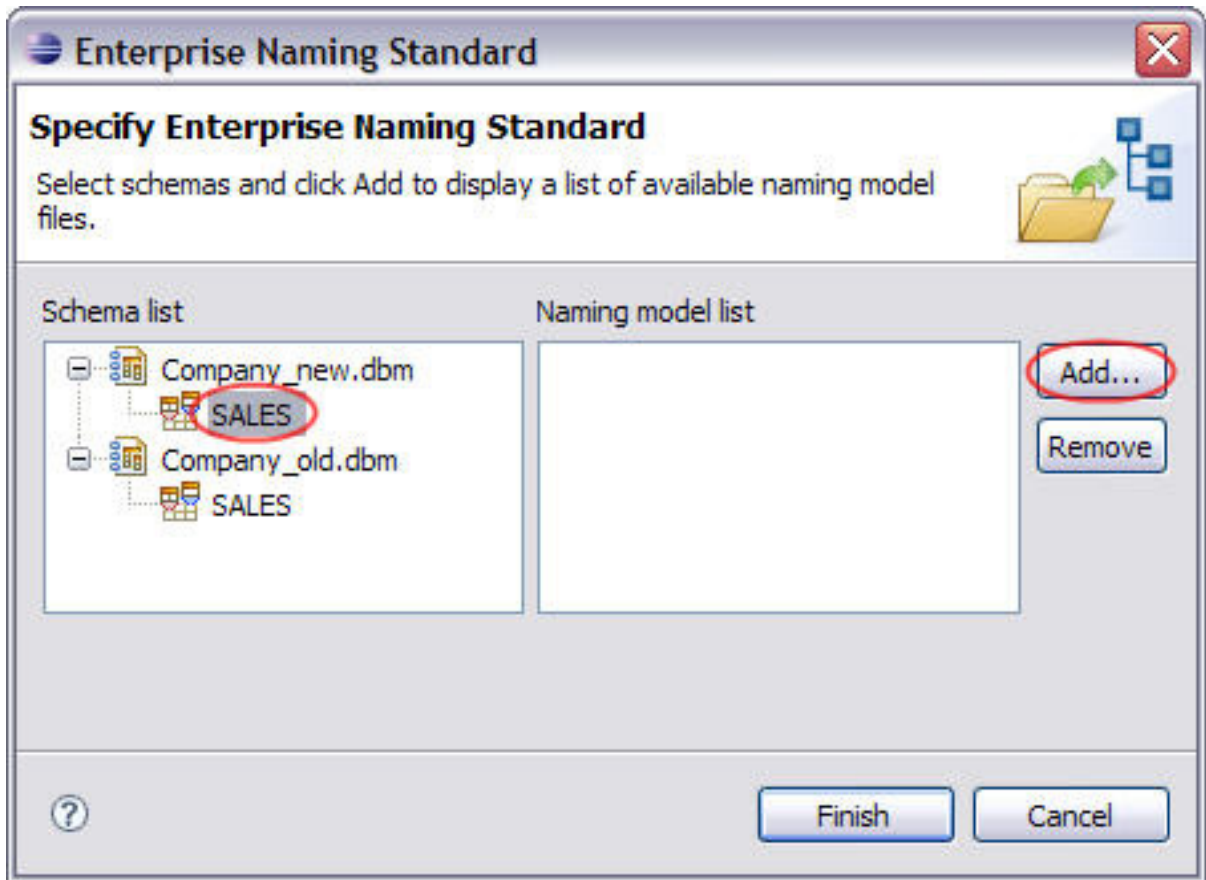
1. Open the mapping model CompanyMapping.msl.
2. Right-click in the mapping editor and select **Discover Relationships > Enterprise Naming Standard** from the context menu.

Figure 24. Invoking the Enterprise Naming Standard wizard



3. In the Enterprise Naming Standard wizard, select the schema **SALES** in Company_new.dbm, and click **Add**.

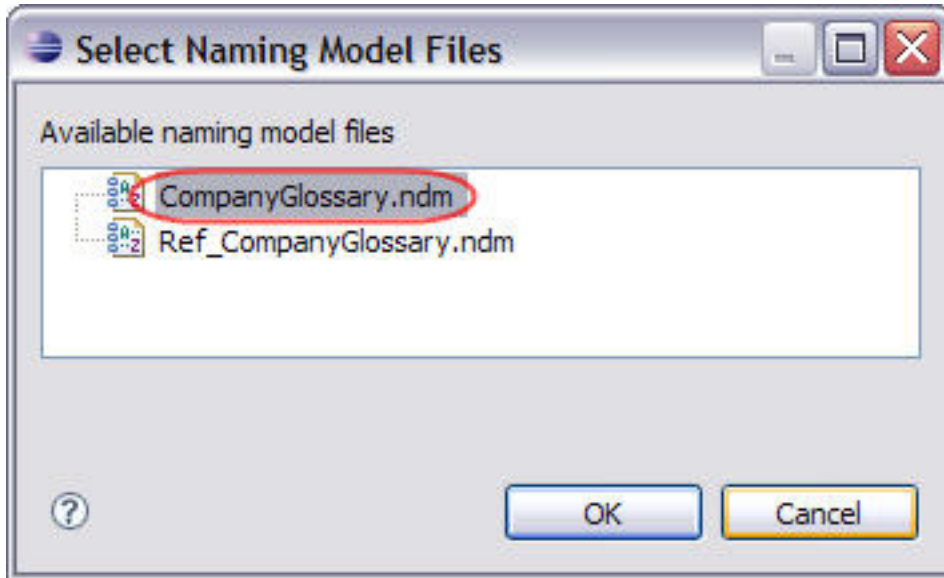
Figure 25. Adding a glossary model to a schema



4. Select the glossary model **CompanyGlossary.ndm** from the list and click **OK**. Click **Finish** to close the Enterprise Naming Standard Wizard. This informs the mapping editor about the abbreviations that are defined in the

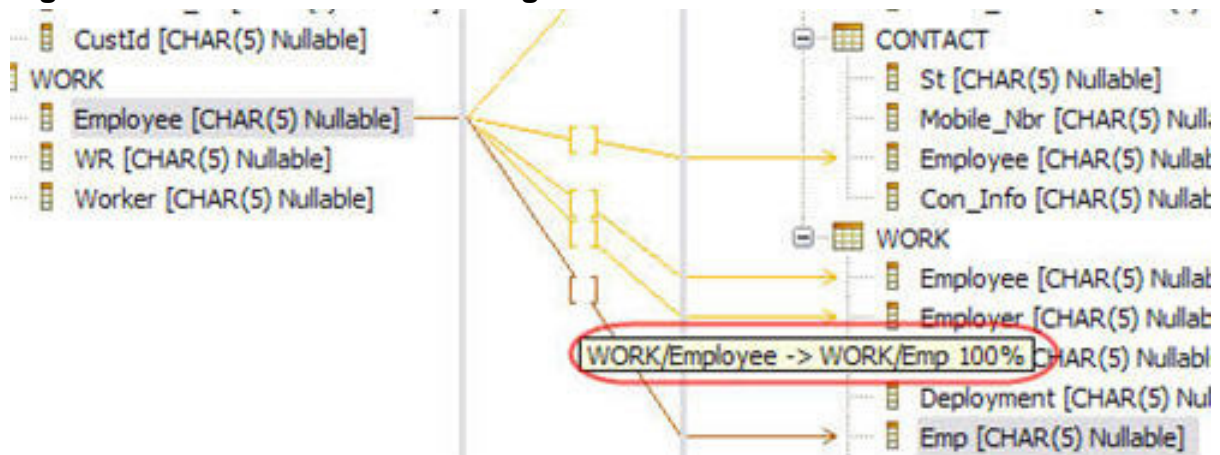
glossary model.

Figure 26. Selecting a glossary model for abbreviation usage



5. In the mapping editor, invoke Find Similar discovery for the source column WORK.Employee (see Figure 16).
6. Instead of the target column Deployment, as in the previous run, now discovery returns the column **Emp**. Since the abbreviation is defined by the user, the matching score is 100%.

Figure 27. Discovered match using abbreviations

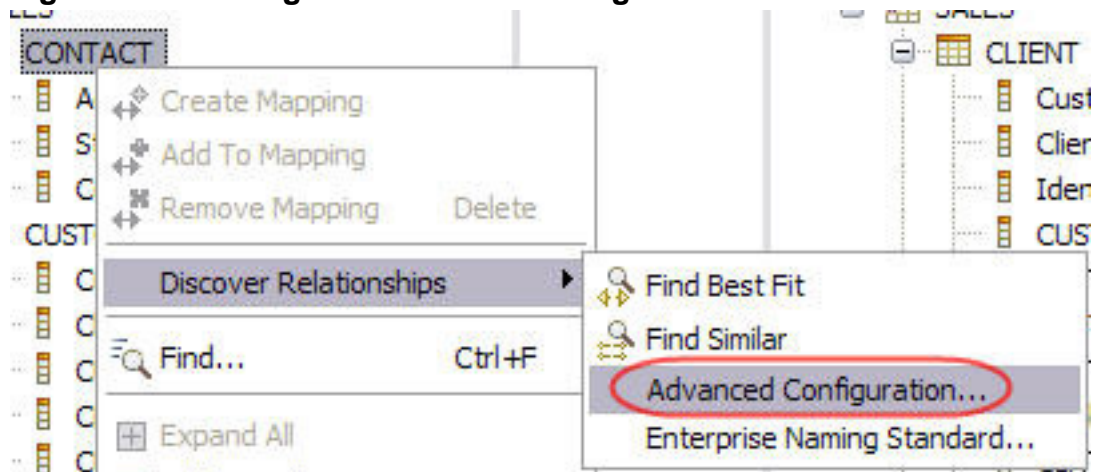


Section 6. Discovering mappings using synonyms

In this section you use the semantic name algorithm, which has the same capabilities as the lexical similarity algorithm. This algorithm also takes synonym information into consideration. The synonyms are defined in a thesaurus. The semantic name algorithm supports three types of thesauri:

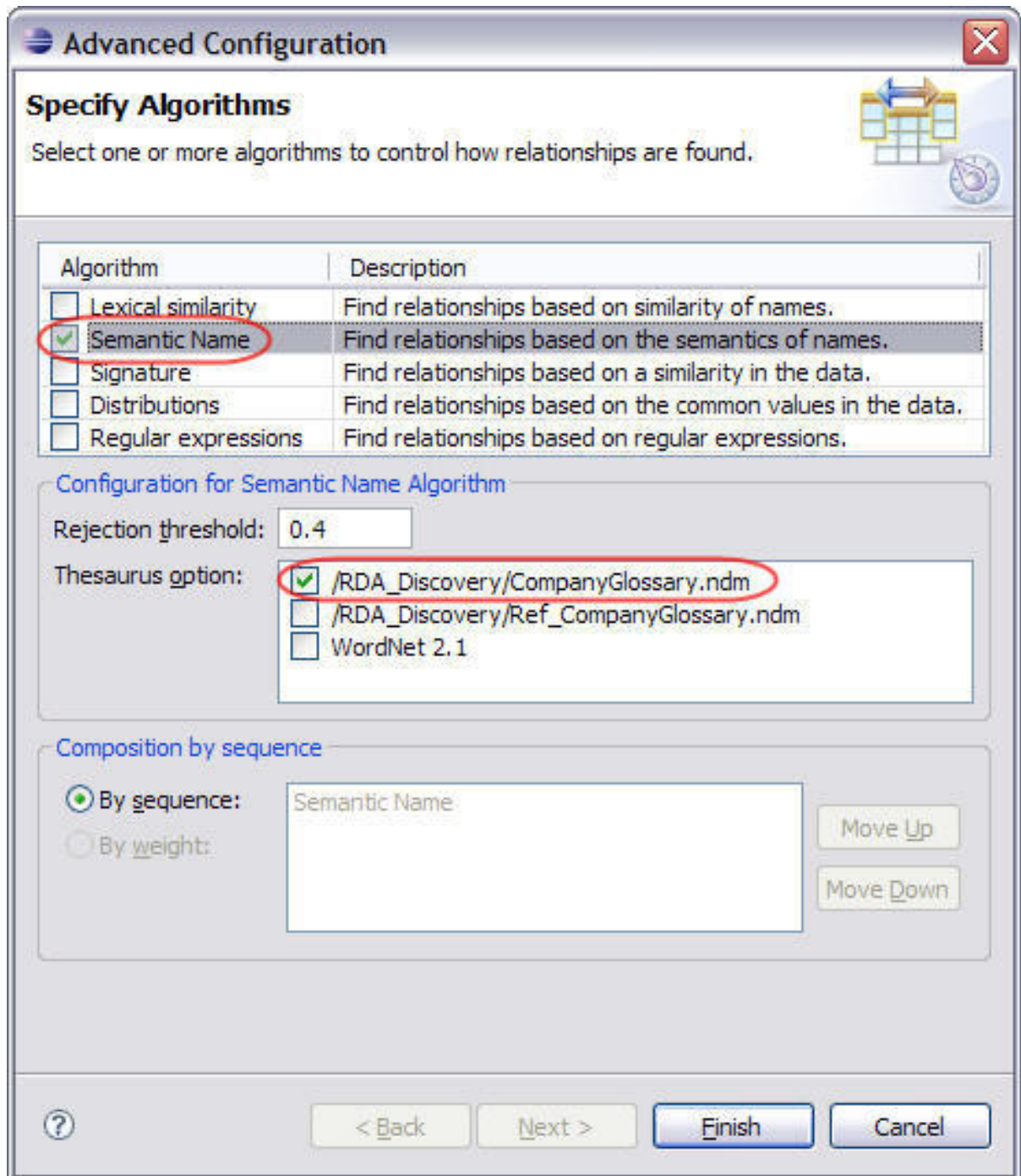
- RDA glossary models
 - [WordNet Thesaurus](#)
 - [SureWord Thesaurus](#)
1. Open the mapping model, **CompanyMapping.msl**.
 2. Right-click in the mapping editor and select **Discover Relationships > Advanced Configuration** from the context menu.

Figure 28. Invoking the Advanced Configuration wizard



3. The Advanced Configuration wizard lets you choose your preferred discovery algorithm. Make sure to uncheck the **Lexical Similarity** algorithm and check the **Semantic Name** algorithm. In the configuration section of the Semantic Name algorithm, there is a list of all thesauri available for the current project. By default, the glossary model that is associated with the project (see [Figure 18](#)) is preselected. When Sureword and WordNet are installed on your system, they show up automatically in the list. Make sure that **CompanyGlossary.ndm** is checked and click **Finish**.

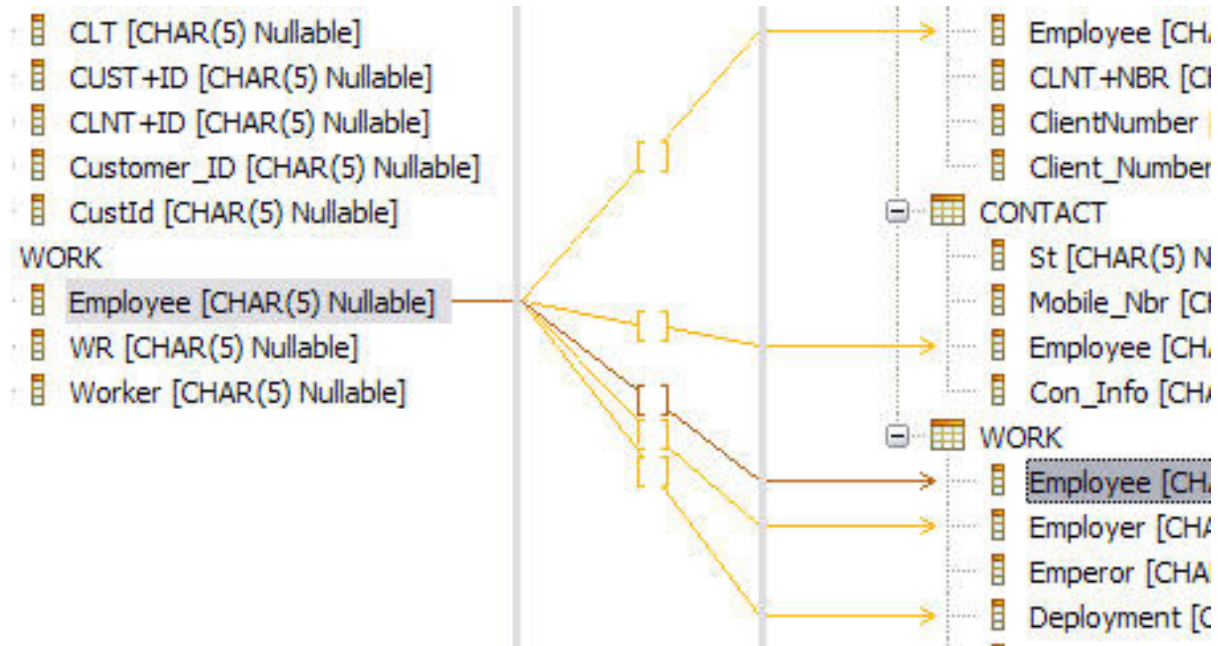
Figure 29. Advanced Configuration for Semantic Name algorithm



- Invoke Find Similar discovery for the source column WORK.Employee (see [Figure 27](#)). In the result, shown below, there are now the target columns **Worker** and **WR**. Worker was discovered because it's a synonym of Employee, and WR is defined as an abbreviation for Worker in the glossary model CompanyGlossary.ndm. Since this glossary model is still defined as the Enterprise Naming Standard for the schema

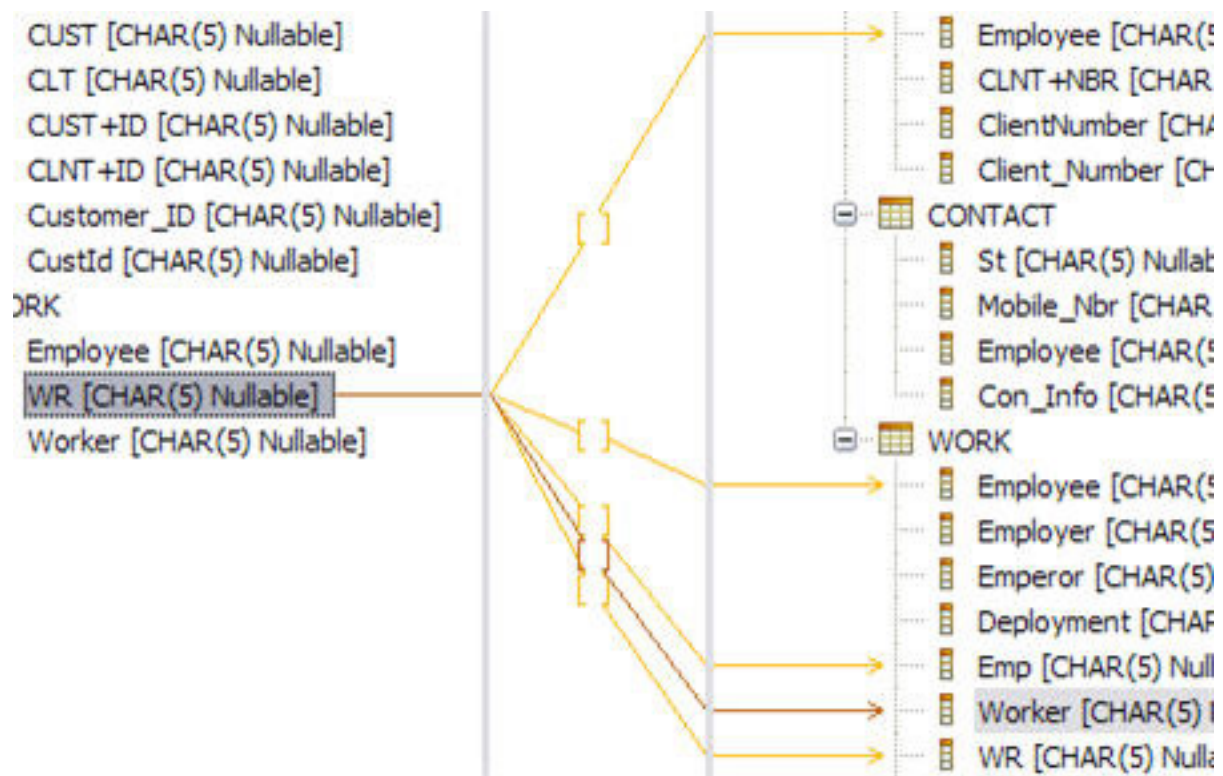
Company_new.dbm/SALES (see Figure 18), WR is discovered as well.

Figure 30. Matches found using synonym discovery



4. Invoke Find Similar discovery for the source column WORK.WR. The only discovered match is the column **WORK.WR** on the target side. Right-click the center pane of the mapping editor and select **Reject All Mappings** from the context menu.
5. Right-click in the mapping editor and select **Discover Relationships > Enterprise Naming Standard** from the context menu. Add the CompanyGlossary.ndm file to the schema Company_old.dbm/SALES (similar to Figure 25 and Figure 26).
6. Invoke Find Similar discovery for the source column WORK.WR once again. Now WR is recognized as an abbreviation of Worker. Consequently, the discovery result includes abbreviations and synonyms for Worker. The result is the same as for the source column WORK.Employee, as shown in Figure 31.

Figure 31. Matches found using abbreviations and synonyms combined

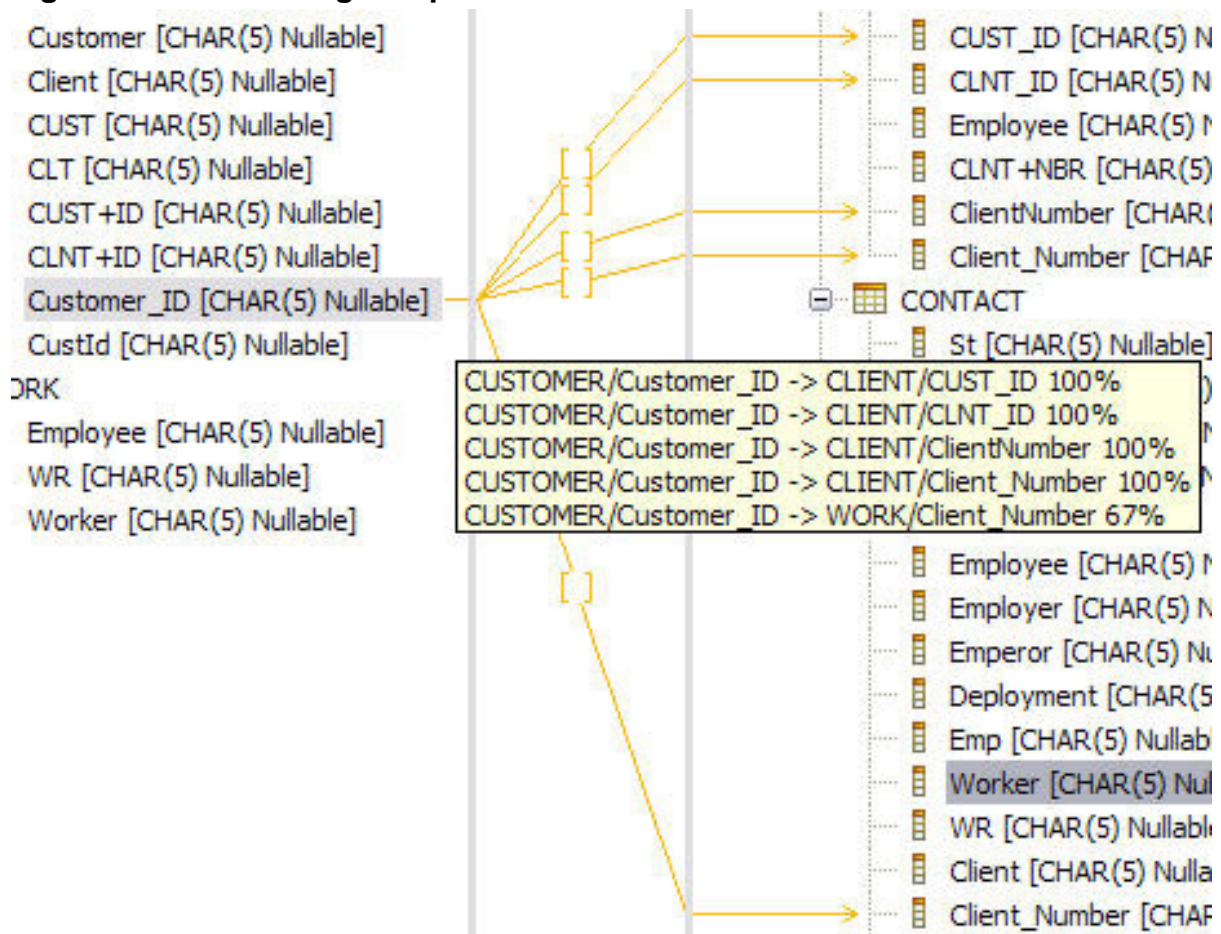


Section 7. Discovering mappings between composed words

In data schemas you often find composed words, such as column names like CUSTOMER_NUMBER or EMPLOYEE_ID. The semantic name algorithm discovers similarities of composed words. In this section you'll learn how to use this feature.

1. Open the mapping model, **CompanyMapping.msl**.
2. Configure the mapping editor to use the semantic name algorithm with CompanyGlossary.ndm as the thesaurus, and configure the enterprise naming standard CompanyGlossary.dbm for both the source and target schemas.
3. Invoke Find Similar discovery on the source column **CUSTOMER.Customer_ID**. As shown in [Figure 32](#), several composed words are returned as the result. The result includes abbreviations and synonyms of the single words.

Figure 32. Discovering composed words



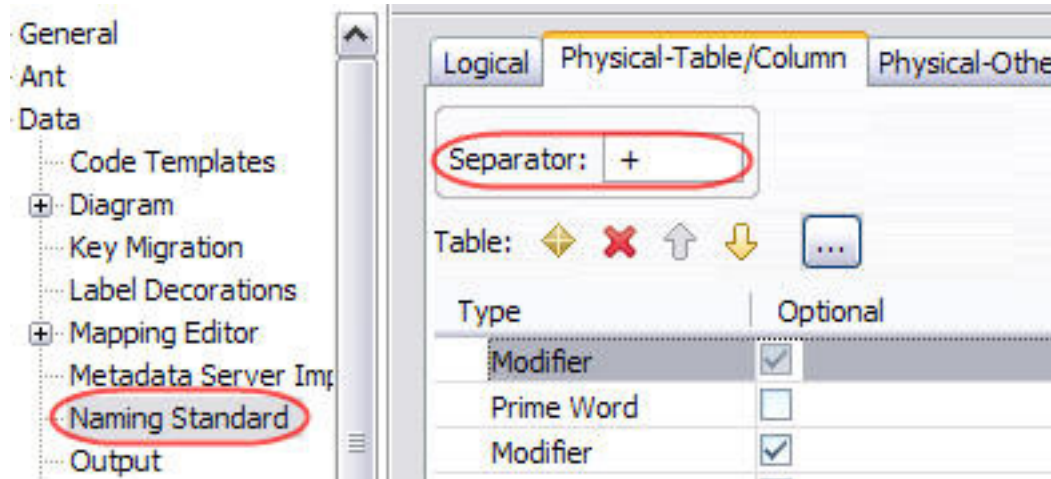
4. There are different ways to separate words from each other. By default, the semantic name algorithm recognizes the following separators between words:

- Space " "
- Underscore "_"
- Pipe symbol "|"
- Semicolon ";"
- Apostrophe "'"
- Comma ","
- Camelcase words (for example, CustomerNumber)

You can use a different separator as well. Open the menu **Window > Preferences** and navigate to **Data > Naming Standard**. The tabs **Logical** and **Physical-Table/Column** have a text field for separators. The separators you define there are both considered during discovery.

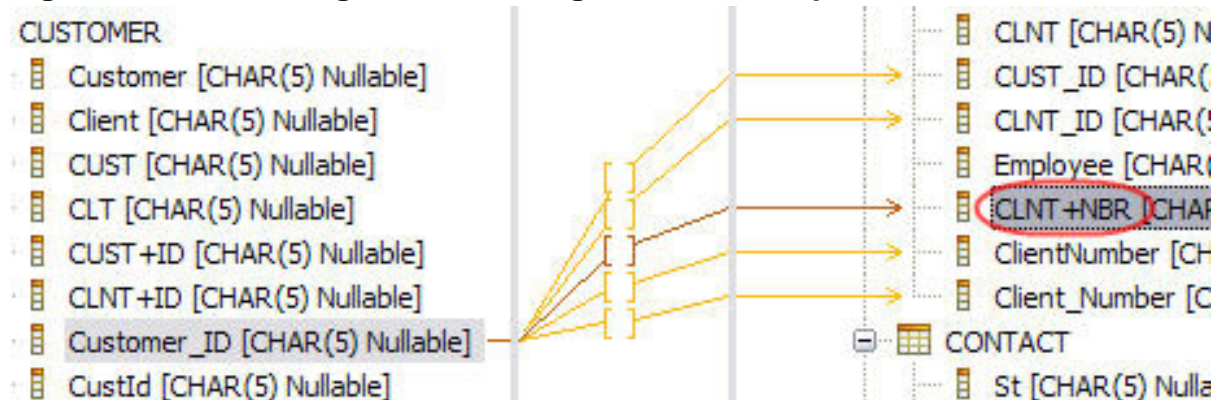
Change the **Separator** for Physical-Table/Column to "+" (plus sign).

Figure 33. Defining a custom word separator



5. Invoke Find Similar discovery on the source column **CUSTOMER.Customer_ID** again. The target column **CLNT+NBR** is now also part of the result.

Figure 34. Discovering columns using the custom separator "+"



The functions of the lexical similarity and semantic name algorithm are also available when mapping a physical database model to an XML schema. Feel free to try them out with the mapping model `CompanyOld_CompanyXSD.msl` that is part of the tutorial package.

Section 8. Discovering mappings using data samples

The matches discovered so far in this tutorial are all based on metadata similarities.

This section shows how to configure and invoke discovery using data samples. Algorithms that use data samples are only available when mapping physical database models as source and target.

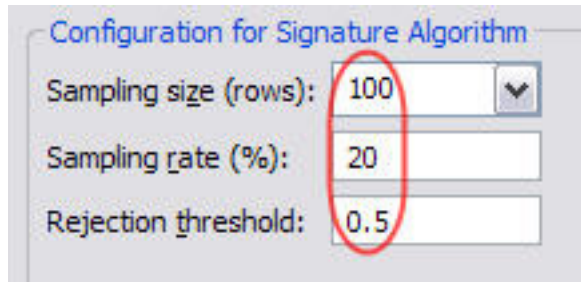
1. Open the mapping model LibraryMapping.msl in the Mappings folder. In this scenario, you have two databases that already contain data and you want to integrate them.
The source database is defined by the model "old library system.dbm" and the target is defined by "new library system.dbm." For simplicity in this tutorial, both schemas are identical. However, the data contained in the physical databases LIBRARY and OLD_LIB (created in [Figure 5](#)) is different.
2. Invoke the discovery advanced configuration wizard (right-click in the mapping editor and select **Discover Relationships > Advanced Configuration** from the context menu).
3. Check the **Signature** algorithm from the list and uncheck all other algorithms.

Figure 35. Signature algorithm

Algorithm	Description
<input type="checkbox"/> Lexical similarity	Find relationships based on similarity of names.
<input type="checkbox"/> Semantic Name	Find relationships based on the semantics of names.
<input checked="" type="checkbox"/> Signature	Find relationships based on a similarity in the data.
<input type="checkbox"/> Distributions	Find relationships based on the common values in the data.
<input type="checkbox"/> Regular expressions	Find relationships based on regular expressions.

4. In the **Configuration for Signature Algorithm** you can configure the number of rows that you want to include in your sample. A bigger sample can help make the discovery results more accurate, but increases the discovery time.
The **Sampling rate** defines the percentage of rows that are sampled (for example, 20% means that 200 out of 1000 rows are sampled). **Sampling size (rows)** is the additional setting to specify the maximum value of sampled rows. Make sure the **Sampling size** value is set to 100 rows, and **Sampling rate** is set to 20%.

Figure 36. Signature algorithm configuration



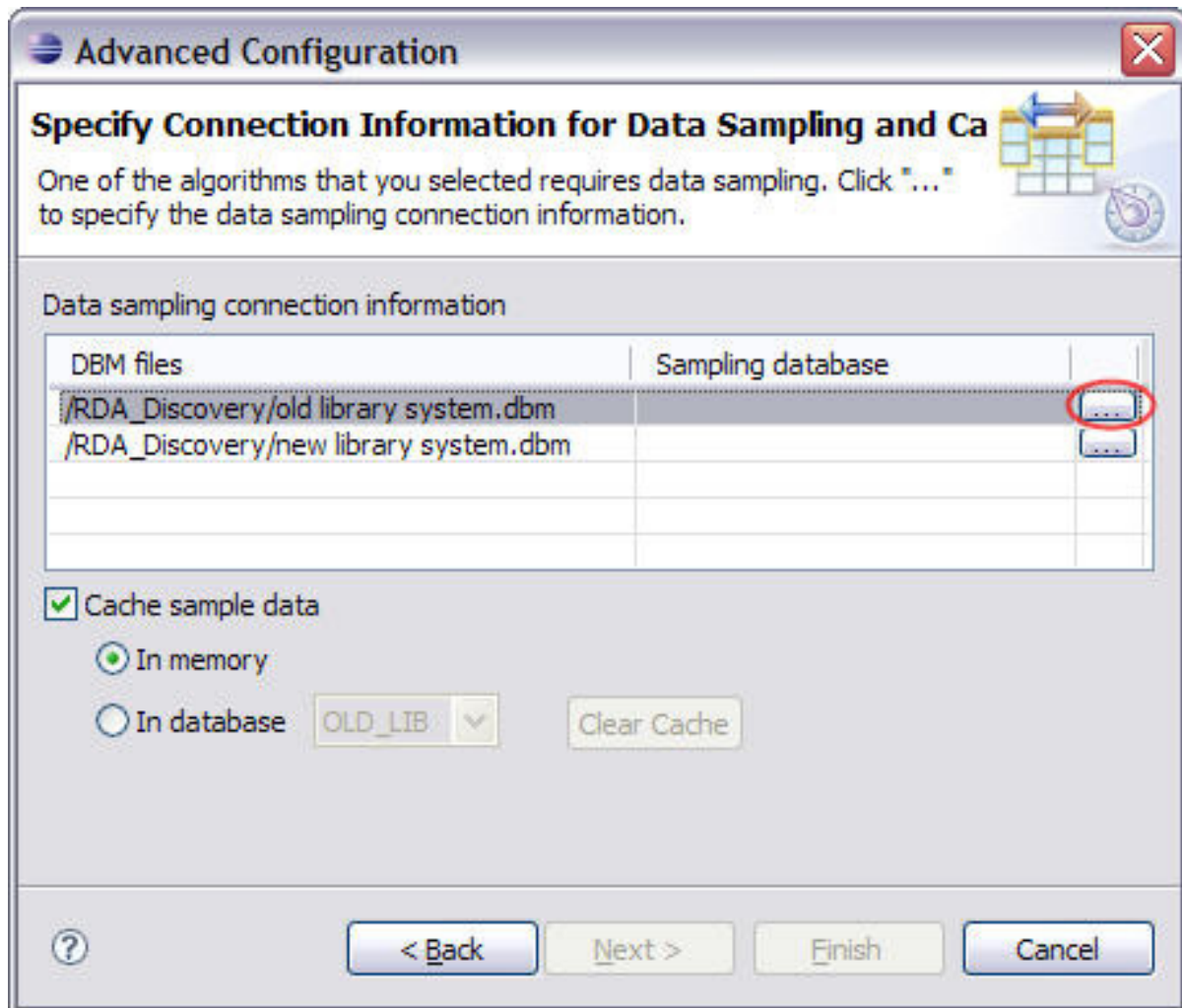
5. Another value that you can set in the configuration for all discovery algorithms is the **Rejection Threshold**. This value serves as a filter. The mapping editor only shows yellow discovered lines for matches with a match ratio lower than the specified rejection threshold value. Make sure that the **Rejection threshold** value for the signature algorithm is set to 0.5, as shown above, and click **Next**.

Best practices

If you find yourself often adjusting the discovery advanced configuration default settings, change these defaults in the preferences (Menu **Window > Preferences**, navigate to **Data > Mapping Editor > Discover Relationships**).

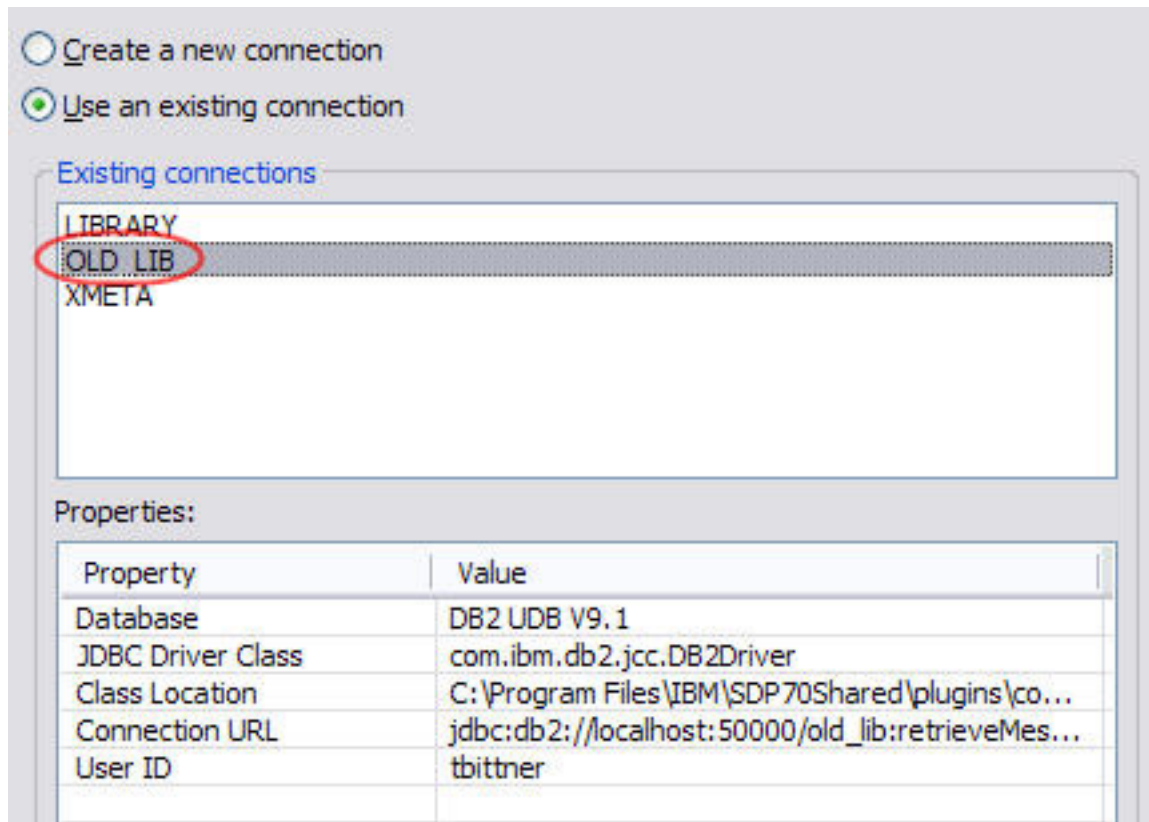
6. The next wizard page asks you to specify the database connection for both source and target schema. Click on the ... button next to /RDA_Discovery/old library system.dbm, as shown here.

Figure 37. Data sampling parameters



7. Select **OLD_LIB** from the list and click **Finish**.

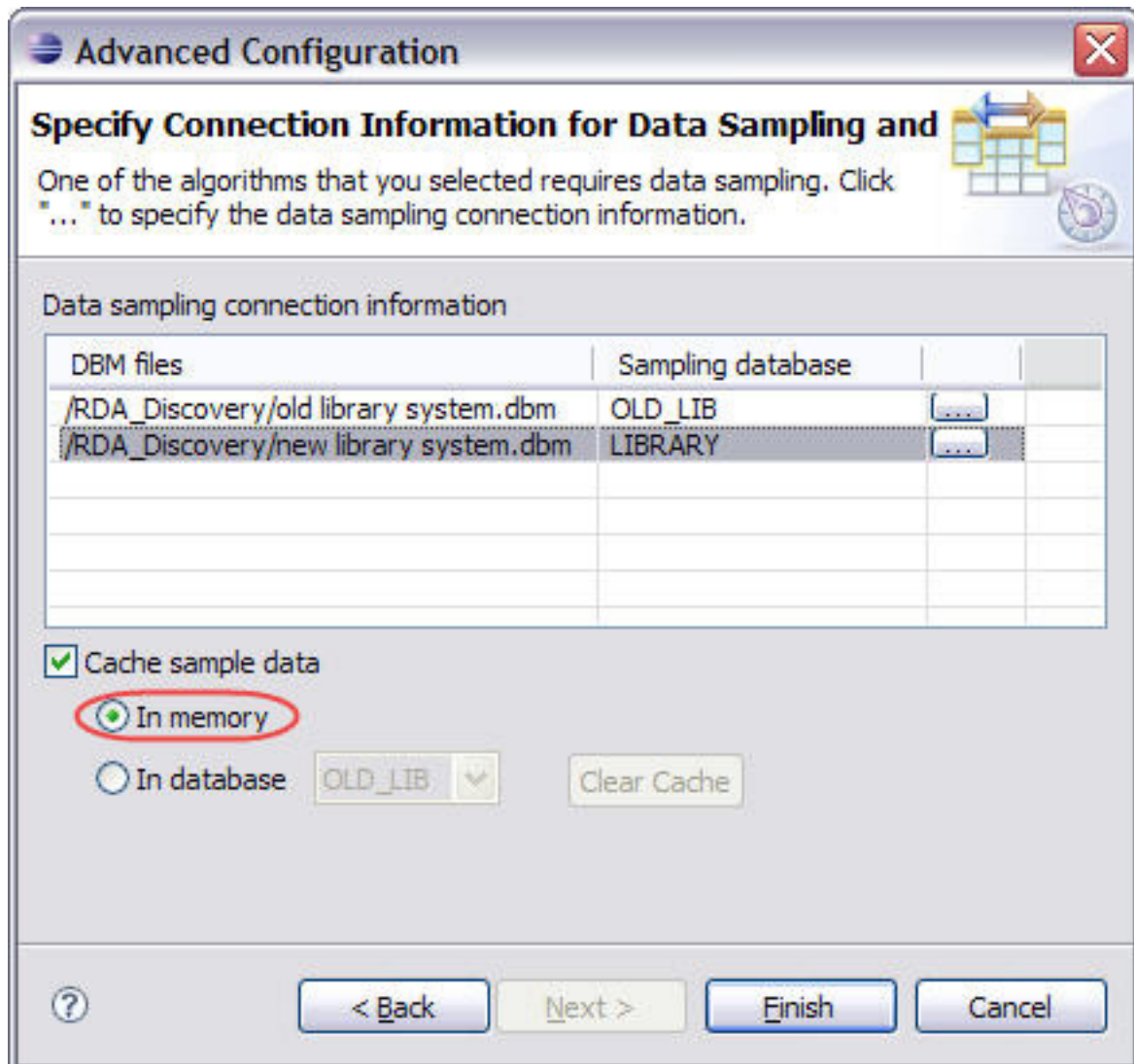
Figure 38. Selecting sampling database



8. Repeat steps 6 and 7 for /RDA_Discovery/new library system.dbm and the database LIBRARY.
9. Discovery offers three options for caching the sample data:
 - No caching -- The data is gathered from the data source for each discovery run.
 - Memory caching -- The data is cached in memory during the first discovery run. Consecutive discovery runs on the same tables perform faster. This setting is recommended for discovery runs with up to 50 tables.
 - Database caching -- The data is cached in a relational database during the first discovery run. Consecutive discovery runs perform faster. This setting is recommended for discovery runs with more than 50 tables.

Make sure that the **In memory** caching option is checked. Click **Finish**.

Figure 39. Selecting in-memory caching



10. Invoke Find Similar discovery on the source column **BOOK_AUTHORS.AUTHORNAME**.

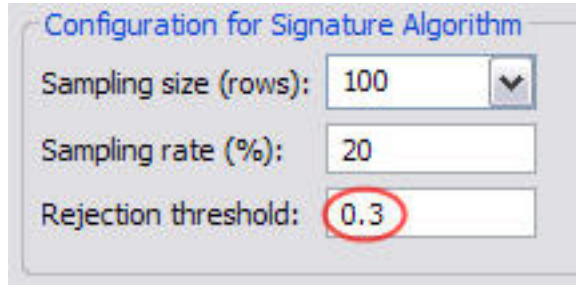
11. The following matches are returned:

```
BOOK_AUTHORS.AUTHORNAME
BOOK.PUBLISHER_NAME
STUDENT.NAME
LOCATION.NAME
BOOK.TITLE
```

12. Reject all mappings.

13. Open the discovery advanced configuration wizard and set the **Rejection threshold** value for the signature algorithm to 0.3, as shown in [Figure 40](#).

Figure 40. Adjusting the rejection threshold value to filter results



14. Invoke Find Similar discovery on the source column **BOOK_AUTHORS.AUTHORNAME** again. The match BOOK.TITLE is not part of the result anymore. It was filtered because its match ratio value is higher than 0.3.

Section 9. Summary

In this tutorial, you learned how to use the Rational Data Architect discovery component to discover schema relationships.

You can now invoke several metadata-driven and data-driven algorithms to discover similarities of schemas. You created your own glossary model and used it to discover similarities using abbreviations and synonyms. You configured discovery to use data samples and invoked the signature algorithm to discover similarities of data.

The Rational Data Architect relationship discovery component helps to significantly reduce the time to create complex mappings in the Rational Data Architect mapping editor.

Downloads

Description	Name	Size	Download method
RDA_Discovery.zip	ar-rdamapcode.zip	48KB	HTTP

[Information about download methods](#)

Resources

Learn

- Check out the other parts of this series:
 - [Part 1: Access and integrate enterprise metadata with Rational Data Architect](#) (Jul 2006): Understand how RDA capabilities can enhance the federation capabilities of WebSphere Information Integrator.
 - [Part 2: Generate SQL/XML queries with Rational Data Architect](#) (Sep 2006): Get an introduction to the SQL/XML generation component of Rational Data Architect.
- ["Use Rational Data Architect to integrate data sources"](#) (developerWorks, Mar 2006): Explore a tool-supported process for federation design in five steps.
- [developerWorks resource page for DB2 for Linux, UNIX, and Windows](#): Read articles and tutorials and connect to other resources to expand your DB2 skills.
- [developerWorks Architecture zone](#): Get the resources you need to advance your skills in the architecture arena.

Get products and technologies

- Download a free trial version of [Rational Data Architect](#).
- Download a free trial version of [DB2 Enterprise 9](#).
- Download a free trial version of [DB2 Enterprise Server Edition, V8.2](#).
- Now you can use DB2 for free. Download [DB2 Express-C](#), a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edition and provides a solid base to build and deploy applications.
- [DB2 Enterprise 9](#) is the result of a five-year development project that transformed traditional (static) database technology into an interactive data server that merges the high performance and ease of use of DB2 with the self-describing benefits of XML.

Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).
- Talk more with other developers at the [developerWorks open source forum](#).

About the author

Torsten Bittner



Torsten Bittner works as a software engineer in Information Management, IBM Software Group. He carries a diploma degree in computer science from the University of Rostock, Germany. His development responsibilities include the Rational Data Architect mapping editor discovery and the query generation component.