

Teiid Designer User Guide

1

7.4.0

1. Introduction	1
1.1. What is Teiid Designer?	1
1.2. Why Use Teiid Designer?	2
1.3. Metadata Overview	2
1.3.1. What is Metadata	2
1.3.2. Editing Metadata vs. Editing Data	3
1.3.3. Metadata Models	3
1.3.4. Business and Technical Metadata	4
1.3.5. Design-Time and Runtime Metadata	5
1.3.6. Source and View Metadata	6
1.4. Models 101	10
1.4.1. What Are Models	10
1.4.2. How is a Model Defined?	11
1.4.3. Model Classes and Types	12
1.4.4. Models and VDBs	12
1.4.5. Models and Connection Profiles	12
1.4.6. Model Validation	12
1.4.7. Testing Your Models	13
2. Teiid Designer Perspectives	15
2.1. Teiid Designer Perspective	15
2.2. Opening a Perspective	17
2.3. Further information	19
3. Teiid Designer Main Menu	21
3.1. File Menu	21
3.2. Edit Menu	24
3.3. Refactor Menu	26
3.4. Navigate Menu	27
3.5. Search Menu	27
3.6. Project Menu	29
3.7. Metadata Menu	30
3.8. Run Menu	31
3.9. Window Menu	31
3.10. Help Menu	32
4. Teiid Designer Views	35
4.1. Model Explorer View	36
4.1.1. Selection-Based Action Menus	38
4.2. Outline View	39
4.2.1. Outline Tree View	39
4.2.2. Outline Thumbnail View	40
4.3. Teiid View	41
4.4. Properties View	45
4.5. Description View	47
4.6. Editors	48
4.7. Problems View	51

4.7.1. Toolbar Menu	52
4.7.2. Context Menu	52
4.8. Search Results View	53
4.9. Datatype Hierarchy View	55
4.10. Teiid Model Classes View	56
4.11. System Catalog View	57
5. Editors	59
5.1. Model Editor	61
5.1.1. Diagram Editor	61
5.1.2. Table Editor	73
5.1.3. Simple Datatypes Editor	78
5.1.4. Semantic Editor	80
5.1.5. Source Editor	80
5.1.6. Model Object Editors	80
5.2. VDB Editor	80
5.2.1. Editing Data Roles	82
5.2.2. Editing Translator Overrides	83
6. Importers	87
6.1. Import DDL into Relational Model	87
6.2. Import From JDBC	90
6.3. Import Metadata From Text File	97
6.3.1. Import Relational Tables Text Importer	99
6.3.2. Import Relational Virtual Tables Text Importer	100
6.3.3. Import Datatypes Text Importer	102
6.4. Import WSDL into Relational Source Model	104
6.5. Import WSDL Into Web Service	109
6.5.1. Import WSDL From Workspace Location	110
6.5.2. Import WSDL From File System Location	117
6.5.3. Import WSDL From URL	124
6.6. XSD Schema File	131
7. New Model Wizards	137
7.1. Creating New Relational Source Model	138
7.1.1. Generate File Translator Procedures	139
7.1.2. Generate Web Service Translator Procedures	140
7.1.3. Copy From Existing Model	141
7.2. Creating New Relational View Model	142
7.2.1. Copy From Existing Model	143
7.2.2. Transform From Existing Model	144
7.2.3. Create From XML Schema	144
7.3. Creating XML Document View Model	145
7.3.1. Copy From Existing Model	146
7.3.2. Build XML Documents From XML Schema	147
7.4. Creating XML Schema Model	150
7.4.1. Copy From Existing Model	151

7.5. Creating Web Service View Model	151
7.5.1. Copy From Existing Model	152
7.5.2. Build From Existing WSDL File(s) or URL	153
7.5.3. Build From Relational Models	153
7.5.4. Build From XML Document View Models	156
7.6. Creating New Extensions Model	158
7.6.1. Copy From Existing Model	159
8. Creating and Editing Model Objects	161
8.1. Creating New Model Objects	161
8.1.1. New Child Action	161
8.1.2. New Sibling Action	164
8.1.3. New Association Action	166
8.2. Model Object Editors	168
8.2.1. Transformation Editor	170
8.2.2. Input Set Editor (XML)	182
8.2.3. Choice Editor (XML)	184
8.2.4. Recursion Editor (XML)	187
8.2.5. Operation Editor	191
8.3. Manage Extension Properties	192
9. Editing Models and Projects	195
9.1. Rename A Model	195
9.2. Move Model	196
9.3. Save Copy of Model	198
9.4. Clone Project	200
10. Testing Your Models	205
10.1. Manage Connection Profiles	205
10.1.1. Set Connection Profile for Source Model	205
10.1.2. View Connection Profile for Source Model	205
10.1.3. Remove Connection Profile from Source Model	206
10.2. Previewing Data For a Model	207
10.2.1. Preview Relational Table or View	208
10.2.2. Preview Relational Table With Access Pattern	208
10.2.3. Preview Relational Procedure	209
10.2.4. Preview Web Service Operation	210
10.2.5. Sample SQL Results for Preview Data	211
10.3. Testing With Your VDB	212
10.3.1. Creating Data Sources	212
10.3.2. Execute VDB from Model Explorer	214
10.3.3. Deploy VDB from Model Explorer	215
10.3.4. Executing a Deployed VDB	215
11. Web Service Wizards	221
11.1. Generating a JBossWS-CXF War	221
11.2. Generating a RESTEasy War	224
12. Searching	231

12.1. Finding Model Objects	232
12.2. Search Models Via Relationship Properties	233
12.3. Search Transformation SQL	235
12.4. Search Models Via Metadata Properties	237
13. User Preferences	239
13.1. Teiid Designer Preferences	239
13.1.1. Diagram Preferences	240
13.1.2. Diagram Printing Preferences	242
13.1.3. Editor Preferences	243
13.1.4. Validation Preferences	246

Introduction

The Teiid Designer User's Guide provides detailed descriptions of Teiid Designer features and functionality.

1.1. What is Teiid Designer?

Teiid Designer is an Eclipse-based graphical modeling tool for modeling, analyzing, integrating and testing multiple data sources to produce Relational, XML and Web Service Views that expose your business data.

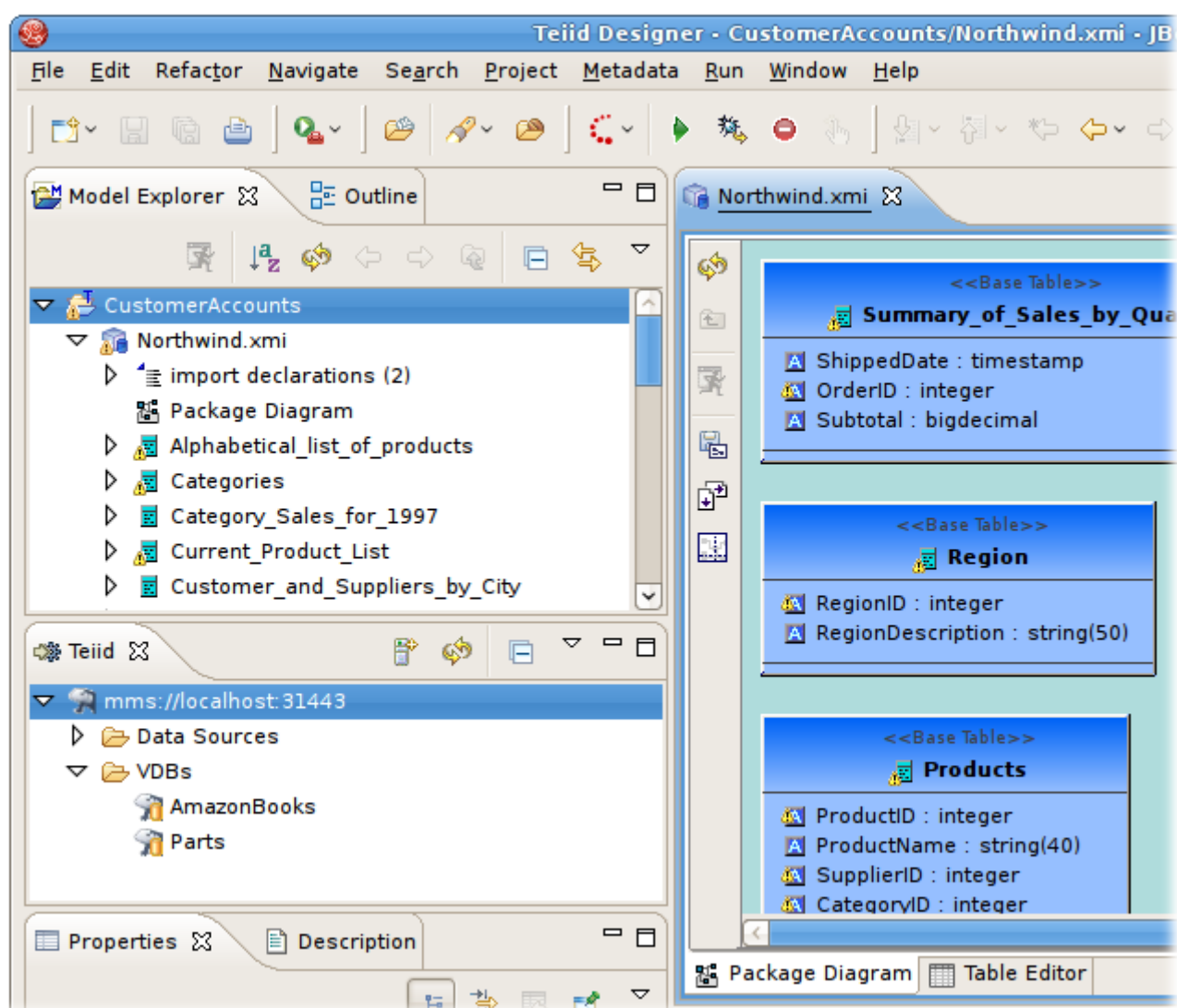


Figure 1.1. Teiid Designer

1.2. Why Use Teiid Designer?

Teiid Designer is a visual tool that enables rapid, model-driven definition, integration and testing of data services without programming. With Teiid Designer, not only do you map from data sources to target formats using a visual tool, but you can also:

- resolve semantic differences
- create virtual data structures at a physical or logical level
- use declarative interfaces to integrate, aggregate, and transform the data on its way from source to a target format which is compatible and optimized for consumption by your applications

- resolve semantic differences

This allows you to abstract the structure of the information you expose to and use in your applications from the underlying physical data structures. With Teiid Designer, data services are defined quickly, the resulting artifacts are easy to maintain and reuse, and all the valuable work and related metadata are saved for later reference.

You can use Teiid Designer to integrate multiple sources, and access them using the common data access standards:

- Web Services / SOAP / XML
- JDBC / SQL
- ODBC / SQL

Teiid Designer is an integral part of the Teiid Designer enterprise-class system for providing data services for service-oriented architectures.

1.3. Metadata Overview

1.3.1. What is Metadata

Metadata is data about data. A piece of metadata, called a meta object in the Teiid Designer, contains information about a specific information structure, irrespective of whatever individual data fields that may comprise that structure.

Let's use the example of a very basic database, an address book. Within your address book you certainly have a field or column for the ZIP code (or postal code number). Assuming that the address book services addresses within the United States, you can surmise the following about the column or field for the ZIP code:

- Named ZIPCode

- Numeric
- A string
- Nine characters long
- Located in the StreetAddress table
- Comprised of two parts: The first five digits represent the five ZIP code numbers, the final four represent the ZIP Plus Four digits if available, or 0000 if not
- Formatted only in integer numeric characters. Errors will result if formatted as 631410.00 or 6314q0000

This definition represents metadata about the ZIP code data in the address book database. It abstracts information from the database itself and becomes useful to describe the content of your enterprise information systems and to determine how a column in one enterprise information source relates to another, and how those two columns could be used together for a new purpose

You can think of this metadata in several contexts:

- What information does the metadata contain? (see [Section 1.3.4, “Business and Technical Metadata”](#))
- What data does the metadata represent? (see [Section 1.3.6, “Source and View Metadata”](#))
- How will my organization use and manage this metadata? (see [Section 1.3.5, “Design-Time and Runtime Metadata”](#))

1.3.2. Editing Metadata vs. Editing Data

The Teiid Designer helps you to create and describe an abstract graphic representation of your data structure of your data in the original data sources. It also describes whether those data sources are composed of Relational databases, text files, data streams, legacy database systems, or some other information type.

The Teiid Designer allows you to create, edit, and link these graphically-represented meta objects that are really a description of your data, and not the data itself.

So when this documentation describes the process of creating, deleting, or editing these meta objects, **remember** that you are not, in fact, modifying the underlying data.

1.3.3. Metadata Models

A **metadata model** represents a collection of metadata information that describes a complete structure of data.

In a previous example we described the field ZIPCode as a **metadata object** in an address book database. This **meta object** represents a single distinct bit of metadata information. We alluded

to its parent table, `StreetAddress`. These **meta objects**, and others that would describe the other tables and columns within the database, would all combine to form a **Source Metadata** model for whichever enterprise information system hosts all the objects.

You can have **Source Models** within your collection of **metadata models**. These model physical data storage locations. You can also have **View Models**, which model the business view of the data. Each contains one type of metadata or another. For more information about difference between Source and View metadata, (see [Section 1.3.6, “Source and View Metadata”](#)).

NOTE: For detailed information about creating models from your metadata, see [Section 1.4, “Models 101”](#)

1.3.4. Business and Technical Metadata

Metadata can include different types of information about a piece of data.

- **Technical metadata** describes the information required to access the data, such as where the data resides or the structure of the data in its native environment.
- **Business metadata** details other information about the data, such as keywords related to the meta object or notes about the meta object.

Note that the terms **technical and business metadata**, refer to the content of the metadata, namely what type of information is contained in the metadata. Don't confuse these with the terms “physical” and “view” metadata that indicate what the metadata represents. For more information, (see [Section 1.3.6, “Source and View Metadata”](#)).

1.3.4.1. Technical Metadata

Technical metadata represents information that describes how to access the data in its original native data storage. Technical metadata includes things such as datatype, the name of the data in the enterprise information system, and other information that describes the way the native enterprise information system identifies the meta object

Using our example of an address book database, the following represent the technical metadata we know about the ZIP code column:

- Named `ZIPCode`
- Nine characters long
- A string
- Located in the `StreetAddress` table
- Uses SQL Query Language

These bits of information describe the data and information required to access and process the data in the enterprise information system.

1.3.4.2. Business Metadata

Business metadata represents additional information about a piece of data, not necessarily related to its physical storage in the enterprise information system or data access requirements. It can also represent descriptions, business rules, and other additional information about a piece of data.

Continuing with our example of the ZIP Code column in the address book database, the following represents business metadata we may know about the ZIP code:

- The first five characters represent the five ZIP code numbers, the final four represent the ZIP Plus Four digits if available, or 0000 if not
- The application used to populate this field in the database strictly enforces the integrity of the data format

Although the first might seem technical, it does not directly relate to the physical storage of the data. It represents a business rule applied to the contents of the column, not the contents themselves.

The second, of course, represents some business information about the way the column was populated. This information, although useful to associate with our definition of the column, does not reflect the physical storage of the data.

1.3.5. Design-Time and Runtime Metadata

Teiid Designer software distinguishes between design-time metadata and run-time metadata. This distinction becomes important if you use the Teiid Designer Server. Design-time data is laden with details and representations that help the user understand and efficiently organize metadata. Much of that detail is unnecessary to the underlying system that runs the Virtual Database that you will create. Any information that is not absolutely necessary to running the Virtual Database is stripped out of the run-time metadata to ensure maximum system performance.

1.3.5.1. Design-Time Metadata

Design-time metadata refers to data within your local directory that you have created or have imported. You can model this metadata in the Teiid Designer, adding **Source** and **View** metadata.

1.3.5.2. Runtime Metadata

Once you have adequately modeled your enterprise information systems, including the necessary technical metadata that describes the physical structure of your sources, you can use the metadata for data access.

To prepare the metadata for use in the Teiid Designer Server, you take a snapshot of a metadata model for the Teiid Designer Server to use when resolving queries from your client applications. This run-time metadata represents a static version of design-time metadata you created or imported. This snapshot is in the form of a **Virtual Database** definition, or **VDB**.

As you create this **runtime metadata**, the Teiid Designer:

- derives the runtime metadata from a consistent set of metadata models.
- creates a subset of design-time metadata, focusing on the technical metadata that describes the access to underlying enterprise information systems.
- optimizes runtime metadata for data access performance.

You can continue to work with the design-time metadata, but once you have created a runtime metadata model, it remains static.

1.3.6. Source and View Metadata

In addition to the distinction between business and technical metadata, you should know the difference between **Source Metadata** and **View Metadata**.

Source and View metadata refer to what the metadata represents, not its content.

Source Metadata directly represents metadata for an enterprise information system and captures exactly where and how the data is maintained. Source Metadata sounds similar to technical metadata, but Source Metadata can contain both technical and business metadata. When you model Source Metadata, you are modeling the data that your enterprise information systems contain.

View Metadata, on the other hand, represent tailored views that **transform** the **Source Metadata** into the terminology and domain of different applications. **View Metadata**, too, can contain both technical and business metadata. When you model **View Metadata**, you're modeling the data as your applications (and your enterprise) ultimately use it.

1.3.6.1. Modeling Your Source Metadata

When you model the **Source Metadata** within your enterprise information systems, you capture some detailed information, including:

- Identification of datatype
- Storage formats
- Constraints
- Source-specific locations and names

The **Source Metadata** captures this detailed technical metadata to provide a map of the data, the location of the data, and how you access it.

This collection of **Source Metadata** comprises a direct mapping of the information sources within your enterprise. If you use the Teiid Designer Server for information integration, this technical metadata plays an integral part in query resolution.

For example, our ZIPCode column and its parent table StreetAddress map directly to fields within our hypothetical address book database.

To extend our example, we might have a second source of information, a comma-separated text file provided by a marketing research vendor. This text file can supply additional demographic information based upon address or ZIP code. This text file would represent another Enterprise Information System (EIS), and the meta objects in its Source Model would describe each comma-separated value.

1.3.6.2. Modeling Your View Metadata

When you create **View Metadata**, you are not describing the nature of your physical data storage. Instead, you describe the way your enterprise uses the information in its day-to-day operations.

View Metadata derives its classes and attributes from other metadata. You can derive **View Metadata** from **Source Metadata** that describes the ultimate sources for the metadata or even from other View Metadata. However, when you model **View Metadata**, you create special “views” on your existing enterprise information systems that you can tailor to your business use or application expectations. This **View Metadata** offers many benefits:

- You can expose only the information relevant to an application. The application uses this **View Metadata** to resolve its queries to the ultimate physical data storage.
- You can add content to existing applications that require different views of the data by adding the **View Metadata** to the existing **View Metadata** that application uses. You save time and effort since you do not have to create new models nor modify your existing applications.
- Your applications do not need to refer to specific physical enterprise information systems, offering flexibility and interchangeability. As you change sources for information, you do not have to change your end applications.
- The **View Metadata** models document the various ways your enterprise uses the information and the different terminology that refers to that information. They do so in a central location.

Our example enterprise information sources, the address book database, and the vendor-supplied comma-delimited text file, reside in two different native storage formats and therefore have two Source Metadata models. However, they can represent one business need: a pool of addresses for a mass mailing.

By creating a **View Metadata** model, we could accurately show that this single View Table, the AddressPool, contains information from the two enterprise information systems. The **View Metadata** model not only shows from where it gets the information, but also the SQL operations it performs to select its information from its source models.

This **View Metadata** can not only reflect and describe how your organization uses that information, but, if your enterprise uses the Teiid Designer Server, your applications can use the **View Metadata** to resolve queries.

To create this **View Metadata**, you create a view and define a **transformation** for that view, a special query that enables you to select information from the source (or even other view) metadata models. For more information, see “[Section 8.2.1, “Transformation Editor”](#).”

1.3.6.3. Modeling Metadata Transformations

Section on Modeling transformations.

1.3.6.3.1. Metadata Transformations

By modeling View Metadata, you can illustrate the business view of your enterprise information sources. View Metadata models not only describe that business view, but also illustrate how the meta objects within the View Metadata models derive their information from other metadata models.

Let’s return to the example of our address book database and the vendor’s comma-separated list. We want to generate the View Metadata model, Address Pool, from these enterprise information systems.

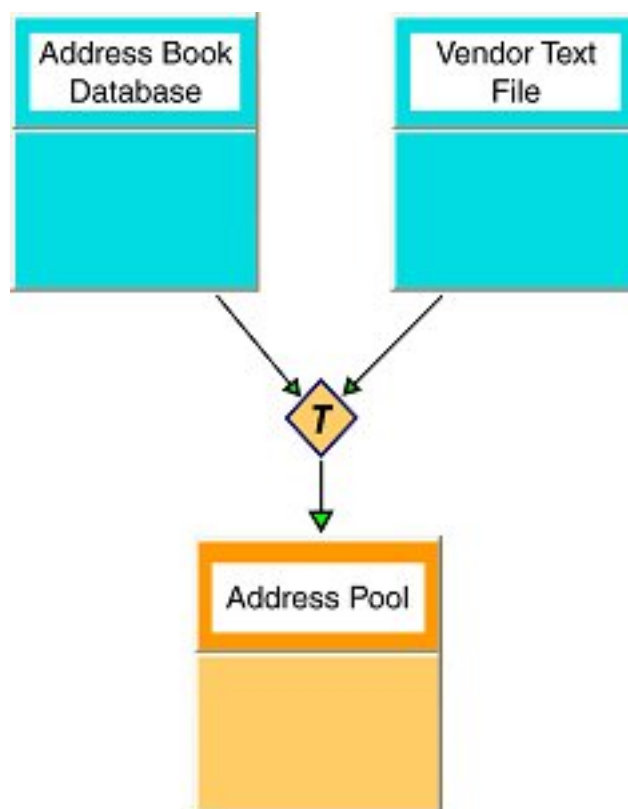


Figure 1.2. Data Flow for View Transformations

The transformation that joins these metadata models to create the virtual Address Pool metadata model contains a SQL query, called a union, that determines what information to draw from the source metadata and what to do with it.

The resulting Address Pool contains not only the address information from our Address Book database, but also that from our vendor-supplied text file.

1.3.6.3.2. SQL in Transformations

Transformations contain SQL queries that SELECT the appropriate attributes from the information sources.

For example, from the sources the transformation could select relevant address columns, including first name, last name, street address, city, state, and ZIP code. Although the metadata models could contain other columns and tables, such as phone number, fax number, e-mail address, and Web URL, the transformation acts as a filter and populates the Address Pool metadata model with only the data essential to building our Address Pool.

You can add other SQL logic to the transformation query to transform the data information. For example, the address book database uses a nine-character string that represents the ZIP Plus Four. The transformation could perform any SQL-supported logic upon the ZIPCode column to substring this information into the format we want for the Address Pool View metadata model.

1.3.6.3.3. Mapping XML Transformations

When you model View Metadata, you can also create a View XML Document model. This View Document lets you select information from within your other data sources, just like a regular View Metadata model, but you can also map the results to tags within an XML document.

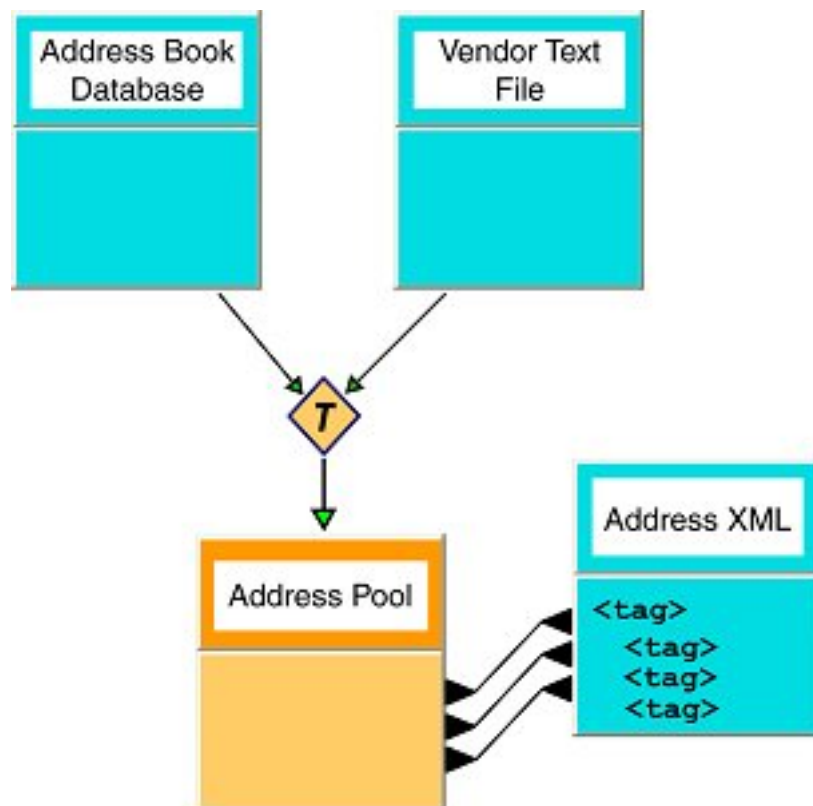


Figure 1.3. Data Flow for XML Transformations

In this example, the Address Pool View Metadata model still selects its information from the Address Book Database and the Vendor Text File, but it also maps the resulting columns into tags in the Address XML document.

1.4. Models 101

1.4.1. What Are Models

A model is a representation of a set of information constructs. A familiar model is the relational model, which defines tables composed of columns and containing records of data. Another familiar model is the XML model, which defines hierarchical data sets.

In Teiid Designer, models are used to define the entities, and relationships between those entities, required to fully define the integration of information sets so that they may be accessed in a uniform manner, using a single API and access protocol. The file extension used for these models is `.xmi` (Example: `NorthwindOracle.xmi`) which adheres to the XMI syntax defined by the OMG.

Below is an example of the partial contents of a model file.

```
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xm
  <mmcore:ModelAnnotation xmi:uuid="mmuuid:b0355f00-413b-1079-9d18-8acf4a712f"
    <modelImports xmi:uuid="mmuuid:2d815780-4140-1079-9d18-8acf4a712f"
    <modelImports xmi:uuid="mmuuid:2e663940-4140-1079-9d18-8acf4a712f"
  </mmcore:ModelAnnotation>
  <relational:BaseTable xmi:uuid="mmuuid:b20e64c0-413b-1079-9d18-8acf4a712f"
    <columns xmi:uuid="mmuuid:bb5ac3c0-413b-1079-9d18-8acf4a712f"
      <type href="http://www.w3.org/2001/XMLSchema#long"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c0-413b-1079-9d18-8acf4a712f"
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c1-413b-1079-9d18-8acf4a712f"
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c2-413b-1079-9d18-8acf4a712f"
      <type href="http://www.metamatrix.com/metamodels/SimpleDataModel#string"/>
    </columns>
    <primaryKey xmi:uuid="mmuuid:d481f940-413b-1079-9d18-8acf4a712f"
  </relational:BaseTable>
```

Figure 1.4. Sample Model File

Model files should never be modified "by hand". While it is possible to do so, there is the possibility that you may corrupt the file such that it cannot be used within the Teiid Designer system.

The fundamental models in Teiid Designer define the structural and data characteristics of the information contained in data sources. These are referred to as source models (represented by



). Teiid Designer uses the information in source models to federate the information in multiple sources, so that from a user's viewpoint these all appear to be in a single source.

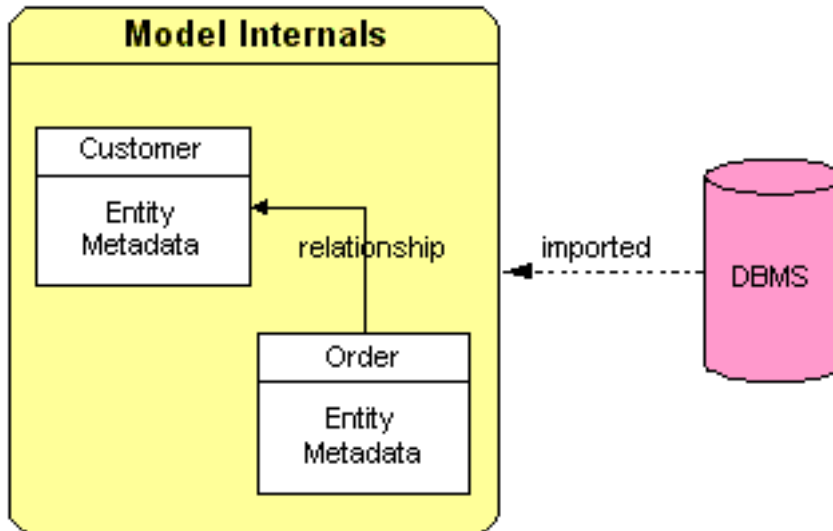


Figure 1.5. Model Internals

In addition to source models, Teiid Designer provides the ability to define a variety of view models (represented by



). These can be used to define a layer of abstraction above the physical (or source) layer, so that information can be presented to end users and consuming applications in business terms rather than as it is physically stored. Views are mapped to sources using transformations between models. These business views can be in a variety of forms:

- Relational Tables and Views
- XML
- Web services
- Functions

For full list of supported model types see [Chapter 7, New Model Wizards](#)

A third model type, logical, provides the ability to define models from a logical or structural perspective.

1.4.2. How is a Model Defined?

Models are defined using Teiid Designer in various ways:

- Created via importing source data characteristics. (see [Chapter 6, Importers](#))

- Manual creation via [Chapter 7, New Model Wizards](#)
- Transforming or copying from one model into another (see [Chapter 7, New Model Wizards](#) options)
- Various custom actions

1.4.3. Model Classes and Types

Teiid Designer can be used to model a variety of classes of models. Each of these represent a conceptually different classification of models.

- **Relational** - Model data that can be represented in table – columns and records – form. Relational models can represent structures found in relational databases, spreadsheets, text files, or simple Web services.
- **XML** - Model that represents the basic structures of XML documents. These can be “backed” by XML Schemas. XML models represent nested structures, including recursive hierarchies.
- **XML Schema** - W3C standard for formally defining the structure and constraints of XML documents, as well as the datatypes defining permissible values in XML documents.
- **Web Services** - which define Web service interfaces, operations, and operation input and output parameters (in the form of XML Schemas).
- **Model Extensions** - for defining property name/value extensions to other model classes.
- **Function** - The Function metamodel supports the capability to provide user-defined functions, including binary source jars, to use in custom transformation SQL statements.

1.4.4. Models and VDBs

Models used for data integration are packaged into a virtual database (VDB). The models must be in a complete and consistent state when used for data integration. That is, the VDB must contain all the models and all resources they depend upon. VDB's are the transport mechanism to expose both the metadata to query against on a server and the corresponding references to the data sources and connections required to perform the actual DB queries. (See the [Section 5.2, “VDB Editor”](#) section)

1.4.5. Models and Connection Profiles

A connection profile provides the connectivity to a data source and is defined by the DTP framework. Designer utilizes this framework from within its JDBC Importer.

1.4.6. Model Validation

Models must be in a valid state in order to be used for data access. Validation of a single model means that it must be in a self-consistent and complete state, meaning that there are no "missing

pieces" and no references to non-existent entities. Validation of multiple models checks that all inter-model dependencies are present and resolvable.

Models must always be validated when they are deployed in a VDB for data access purposes.

Teiid Designer will automatically validate your models whenever the user Saves (Note: the "Models > Validate Automatically" option must be checked). When editing models, the editor tabs will display a "*" to indicate that the model has unsaved changes.

1.4.7. Testing Your Models

Designing and working with data is often much easier when you can see the information you're working with. The Teiid Designer's **Preview Data** feature makes this possible and allows you to instantly preview the information described by any object, whether it's a physical table or a virtual view. In other words, you can test the views with actual data by simply selecting the table, view, procedure or XML document. The preview functionality insures that data access behavior in the Teiid Designer will reliably match when the VDB is deployed to the Server. Previewing information is a fast and easy way to sample the data. Of course, to run more complicated queries like what your application likely uses, simply execute the VDB in the Teiid Designer and type in any query or SQL statement.

After creating your models, you can test them by using the **Preview Data** action



By selecting a desired table object and executing the action, the results of a simple query will be displayed in the Data Tools SQL Results view. This action is accessible throughout the Teiid Designer in various view toolbars and context menus.

Previewable objects include:

- Relational table or view, including tables involving access patterns.
- Relational procedure.
- Web Service operation.
- XML Document staging table.



Note

If attempting to preview a relational access pattern, a web service operation or a relational procedure with input parameters, a dialog will request values for required parameters.

Teiid Designer Perspectives

Teiid Designer utilizes [Eclipse's](http://www.eclipse.org) [http://www.eclipse.org] Workbench environment which controls visual layout via perspectives. A perspective defines the initial set and layout of views and editors. Within the application window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific set of tasks.

Perspectives also control what appears in certain menus and toolbars. They define visible action sets, which you can change to customize a perspective. You can save a perspective that you build in this manner, making your own custom perspective that you can open again later.

2.1. Teiid Designer Perspective

The Teiid Designer perspective provides access to fundamental model editing and management capabilities. This perspective includes 5 main UI components or groups of components as shown below. They include:

- [Section 4.1, “Model Explorer View”](#)
- [Section 4.3, “Teiid View”](#) - Teiid instance view. Provides view of contents for connected instances of installed Teiid runtime.
- [Section 5.1, “Model Editor”](#) - Custom editors targeted for ".xmi" metadata model files.
- [Section 4.4, “Properties View”](#) - Standard property values for selected workbench objects.
- *Miscellaneous Views* - Includes Problems view, Message Log view and the Data Tools SQL Results view (opened if Preview Data action is performed)

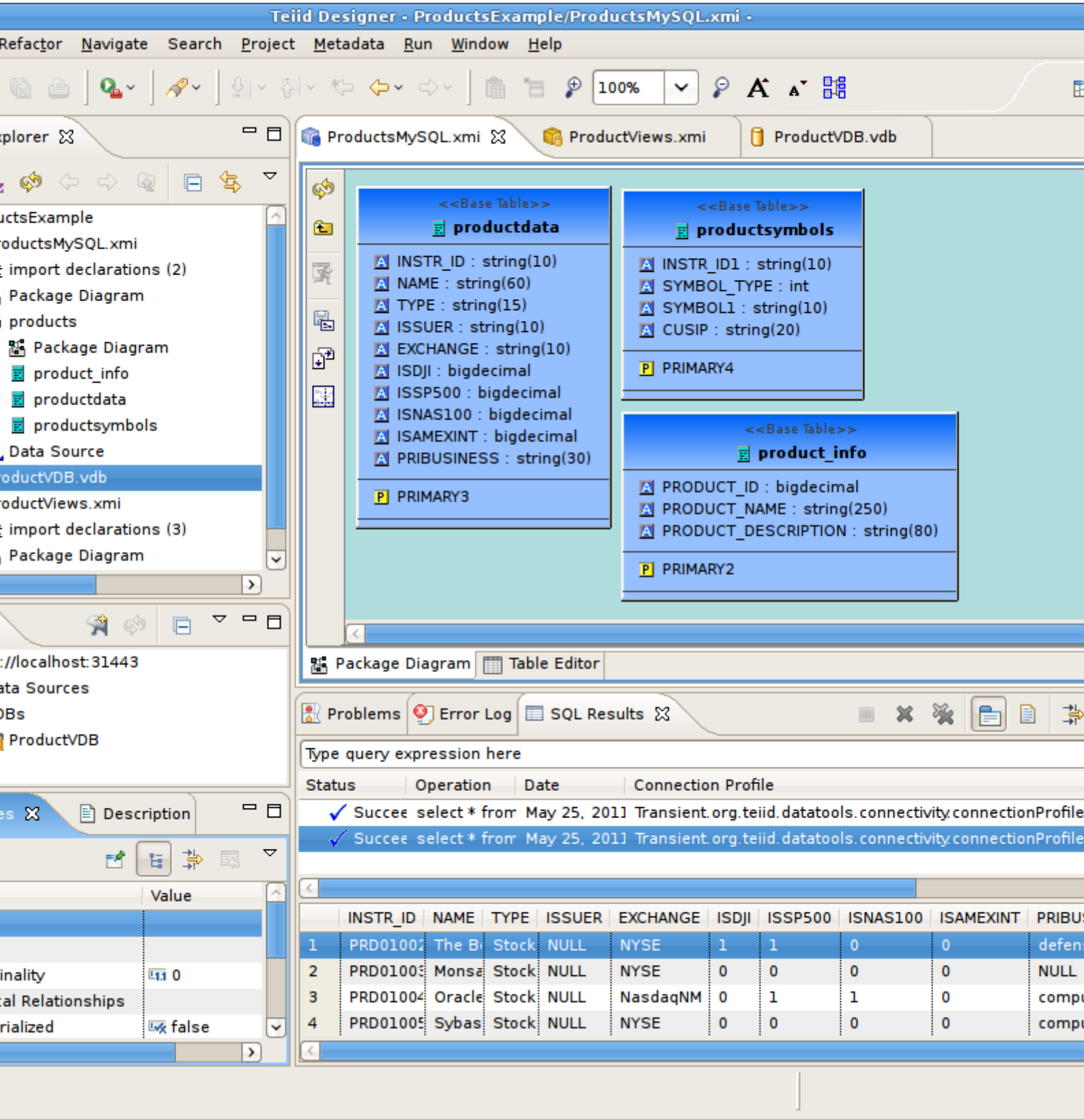


Figure 2.1. Teiid Designer Perspective Layout

2.2. Opening a Perspective

There are two ways to open a perspective:

- Using the Open Perspective button



on the shortcut bar.

- Choosing a perspective from the **Window > Open Perspective** menu.
- To open a perspective by using the shortcut bar button:

- **Step 1** - Click on the Open Perspective button



- **Step 2** - A menu appears showing the same choices as shown on the **Window > Open Perspective** menu. Choose Other from the menu.

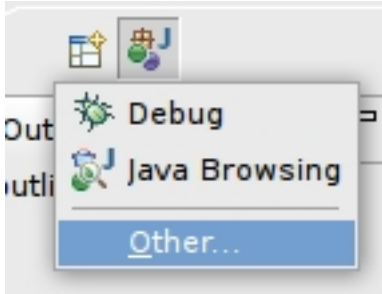


Figure 2.2. Perspectives Menu

- **Step 3** - In the **Select Perspective** dialog choose Teiid Designer and click **OK**.

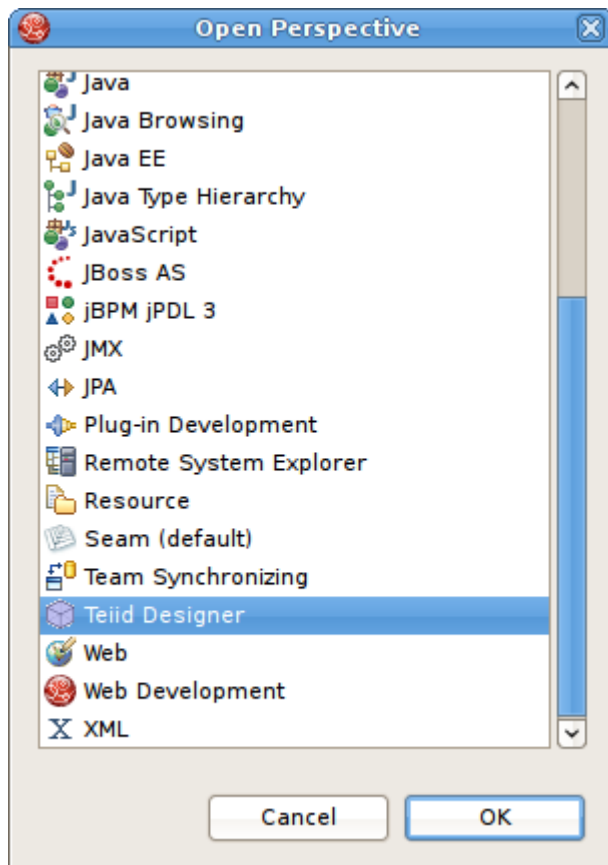


Figure 2.3. Select Perspective Dialog

The Teiid Designer perspective is now displayed.

There are few additional features of perspectives to take note of.

- The title of the window will indicate which perspective is in use.



Figure 2.4. Workbench Window Title Bar

- The shortcut bar may contain multiple perspectives. The perspective button which is pressed in, indicates that it is the current perspective.
- To display the full name of the perspectives, right click the perspective bar and select **Show Text** and conversely select **Hide Text** to only show icons.
- To quickly switch between open perspectives, select the desired perspective button. Notice that the set of views is different for each of the perspectives.

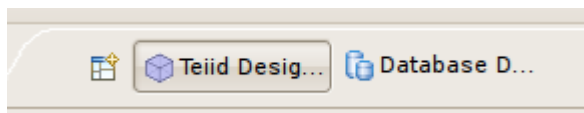


Figure 2.5. Workbench Window Title Bar

2.3. Further information

For more details on perspectives, views and other Eclipse workbench details, see formal [Eclipse Documentation](http://help.eclipse.org/helios/index.jsp) [http://help.eclipse.org/helios/index.jsp].

Teiid Designer Main Menu

There are 8 categories of actions on Teiid Designer's main menu bar.

- These categories include:
 - [Section 3.1, “File Menu”](#) - Resource management actions.
 - [Section 3.2, “Edit Menu”](#) - Standard edit actions including undo/redo.
 - [Section 3.3, “Refactor Menu”](#) - Resource actions (i.e. Rename, Move, etc...).
 - [Section 3.5, “Search Menu”](#) - Find data within your workspace.
 - [Section 3.6, “Project Menu”](#) - Model level actions.
 - [Section 3.7, “Metadata Menu”](#) - Custom metadata-related actions.
 - [Section 3.9, “Window Menu”](#) - Change perspectives or add/remove views to your perspective.
 - [Section 3.10, “Help Menu”](#) - Access available Teiid Designer help documents, Teiid Designer SQL Support Guide and Eclipse Overview information.

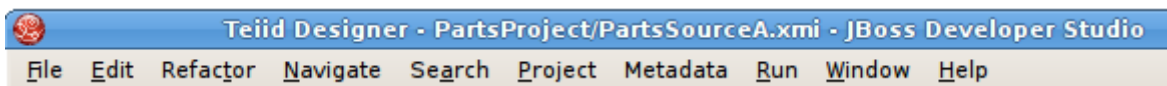


Figure 3.1. Application Main Menu

3.1. File Menu

The **File** menu provides actions to manage your workspace resources.

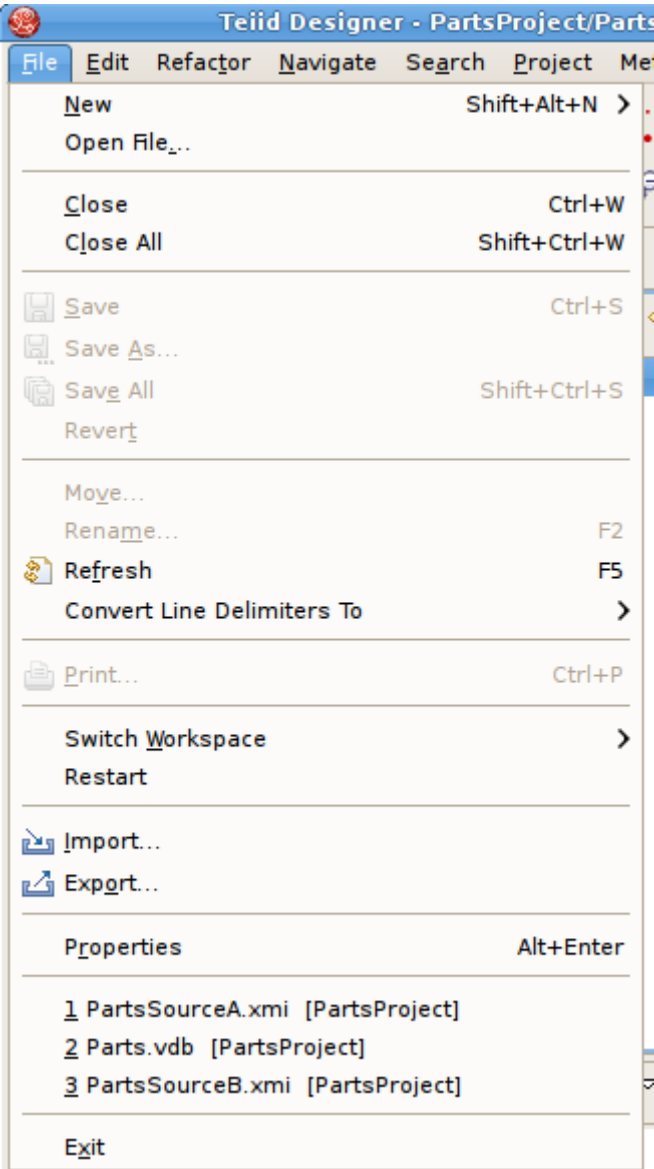


Figure 3.2. File Menu

The **New** > sub-menu provides specific actions to create various generic workspace resources as well as Teiid Designer models and VDBs.

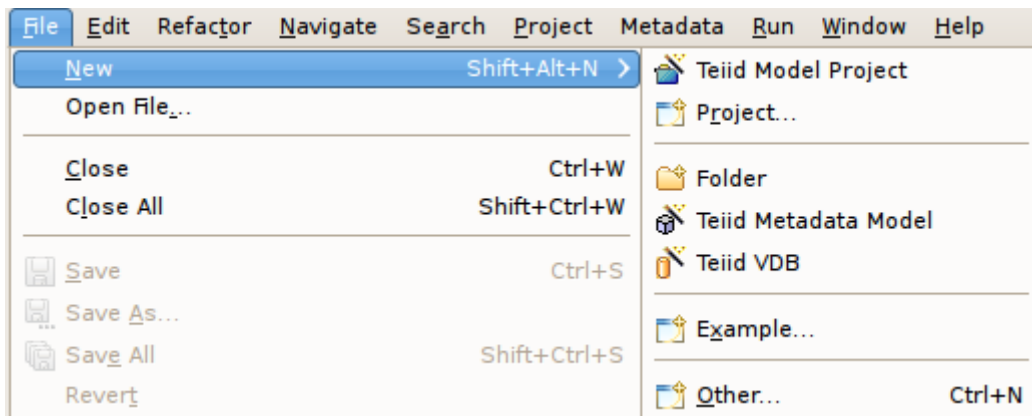








Figure 3.3. File Menu

- The **File** menu contains the following actions:
 -  **New > Model Project** - Create user a new model project.
 -  **New > Folder** - Create new folder within an existing project or folder.
 -  **New > Model** - Create a new model of a specified model type and class using the [Chapter 7, New Model Wizards](#).
 -  **New > Virtual Database Definition** - Create a new VDB, or Virtual Database Definition.
 - **Open File** - Enables you to open a file for editing - including files that do not reside in the Workspace.
 - **Close (Ctrl+W)** - Closes the active editor. You are prompted to save changes before the file closes.
 - **Close All (Shift+Ctrl+W)** - Closes all open editors. You are prompted to save changes before the files close.
 -  **Save (Ctrl+S)** - Saves the contents of the active editor.
 -  **Save As** - Enables you to save the contents of the active editor under another file name or location.



Save All (Shift+Ctrl+S) - Saves the contents of all open editors.

- **Move...** - Launches a Refactor > Move resource dialog..
- **Rename... (F2)** - Launches a Refactor > Rename resource dialog if resource selected, else in-line rename is preformed.
- **Refresh** - Refreshes the resource with the contents in the file system.
- **Convert Line Delimiters To** - Alters the line delimiters for the selected files. Changes are immediate and persist until you change the delimiter again - you do not need to save the file.



Print (Ctrl+P) - Prints the contents of the active editor. In the Teiid Designer, this action prints the diagram in the selected editor. Allows control over orientation (portrait or landscape), scaling, margins and page order. User can also specify a subset of the pages to print (i.e., 2 through 8).

- **Switch Workspace** - Opens the **Workspace Launcher**, from which you can switch to a different workspace. This restarts the **Workbench**.
- **Restart** - Exits and restarts the **Workbench**.



Import - Launches the **Import Wizard** which provides several ways to construct or import models..



Export - Launches the **Export Wizard** which provides options for exporting model data.

- **Properties (Alt+Enter)** - Opens the **Properties** dialog for the currently selected resource. These will include path to the resource on the file system, date of last modification and its writable or executable state.
- **Most Recent Files List** - Contains a list of the most recently accessed files in the **Workbench**. You can open any of these files from the **File** menu by simply selecting the file name.
- **Exit** - Closes and exits the **Workbench**.

3.2. Edit Menu

The **Edit** menu provides actions to manage the content, structure and properties of your model and project resources. The figure below represents the Edit menu presented when a metadata model is selected.

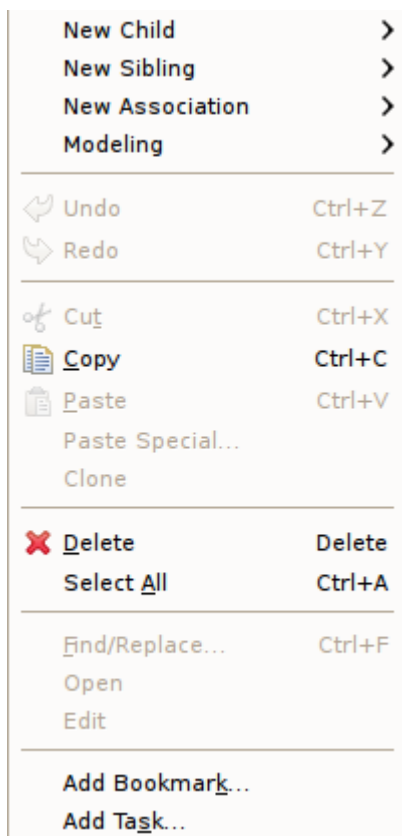








Figure 3.4. Edit Menu

- The **Edit** menu contains the following actions:
 - **New > Child** - This menu is created dynamically to support the creation of whatever types of child objects can be created under the selected object.
 - **New > Sibling** - This menu is created dynamically to support the creation of whatever types of sibling objects can be created under the same parent as the selected object
 - **New > Association** - This menu is created dynamically to support the creation of whatever types of associations can be created with the selected object.
 - **Modeling >** - This menu is created dynamically. Various modeling operations are presented based on selected model object type.
 - 

Undo - Reverses the effect of the most recent command.
 - 

Redo - Reapplies the most recently undone command.

-  **Cut** - Deletes the selected object(s) and copies it to the clipboard.
-  **Copy** - Copies the selected object(s) to the clipboard.
-  **Paste** - Pastes the contents of the clipboard to the selected context.
- **Paste Special...** - Provides additional paste capabilities for complex clipboard objects.
- **Clone** - Duplicates the selected object in the same location with the same name. User is able to rename the new object right in the tree.
-  **Delete** - Deletes the selected object(s).
- **Select All** - Select All objects in current view.
- **Rename** - Allows a user to rename an object in the tree.
- **Find/Replace** - Launches dialog that can be used to search in the current text view, such as a **Transformation Editor**.
- **Open** - Opens the selected object in the appropriate editor.
- **Edit** - Opens the selected object in the appropriate specialized editor, such as the Choice Editor or Recursion Editor..
- **Add Bookmark...** - This command adds a bookmark in the active file on the line where the cursor is currently displayed.
- **Add Task...** - This command adds a task in the active file on the line where the cursor is currently displayed.

3.3. Refactor Menu

The **Refactor** menu provides Teiid Designer specific actions for file-level changes to the models.

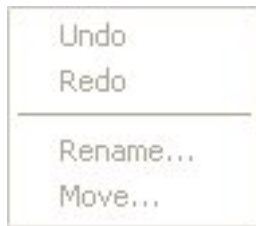


Figure 3.5. Refactor Menu

- The **Refactor** menu contains the following actions:
 - **Undo** - Undo the last refactor command.
 - **Redo** - Redo the last undone refactor command.
 - **Move** - Move a model from one container (folder or project) to another.
 - **Rename** - Rename a model.

3.4. Navigate Menu

Teiid Designer currently does not contribute actions to the Navigate menu. See Eclipse documentation for details.

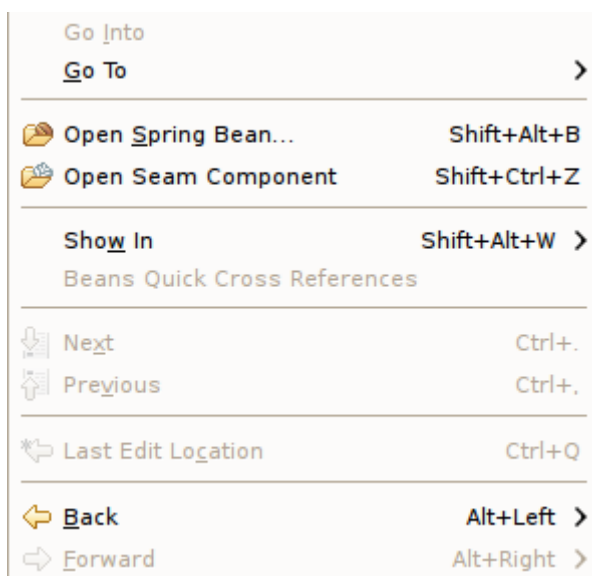


Figure 3.6. Navigate Menu

3.5. Search Menu

The **Search** menu presents several specific search options.

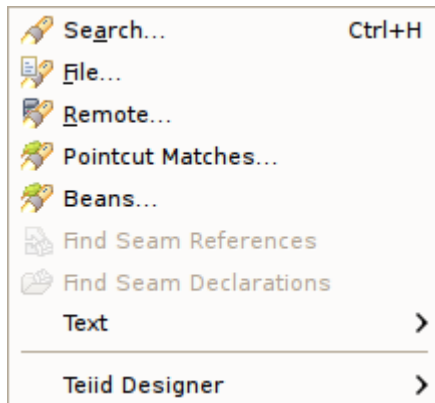


Figure 3.7. Search Menu

Teiid Designer contributes a sub-menu (i.e. Teiid Designer >) to the main search menu, as shown above.

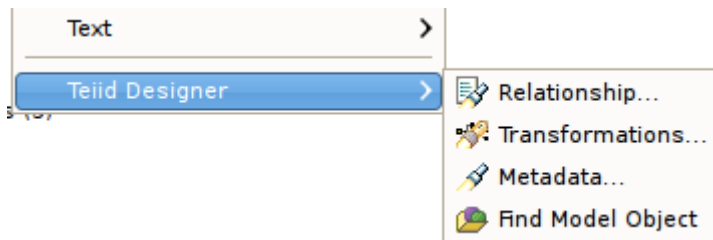


Figure 3.8. Search Menu

- The individual actions in the Teiid Designer sub-menu are described below:



Relationship... - Launches the Search dialog and auto-selects the Relationships tab. User can search for models in the workspace by specifying a relationship type, participant locations, and/or names containing specified text. Search results appear in the [Section 4.8, "Search Results View"](#) view, and double-clicking a result will open that model in the appropriate editor.



Transformations... - Launches the Transformation Search dialog. User can search models in the workspace for matching SQL text. Search results appear in the dialog and user can select and view SQL as well as open desired transformations for editing.



Metadata... - Launches the Search dialog. User can search for models in the workspace by specifying an Object Type, and/or a Data Type, and/or a property value. Search results

appear in the [Section 4.8, “Search Results View”](#) view, and double-clicking a result will open that model in the appropriate editor.



Find Model Object - Launches the Find Model Object dialog, which can be used to find an object in the workspace by specifying all or part of its name. Selecting the object will open it in the appropriate editor.

3.6. Project Menu

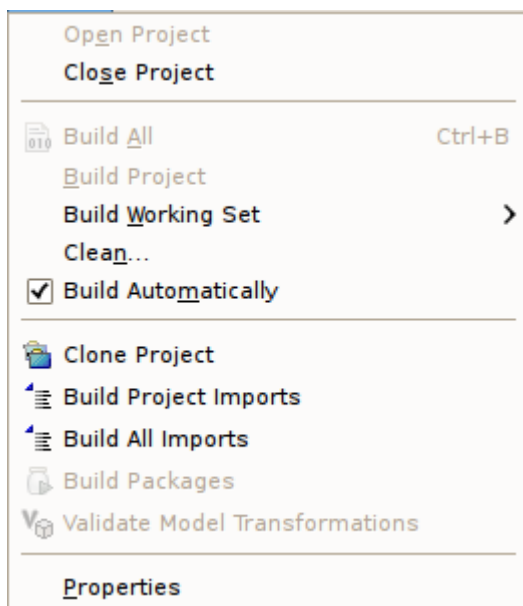



Figure 3.9. Project Menu

- The individual actions in the Project menu are described below:
 - **Open Project** - Launches the **Open Project** dialog.
 - **Close Project** - Closes the currently selected project(s).
 -  **Build All** - Validates the contents of the entire workspace. Any errors or warnings will appear in the **Problems View**.
 - **Build Project** - Validates the contents of the selected project(s). Any errors or warnings will appear in the **Problems View**.
 - **Build Working Set** - Validates the contents of the selected working set. Any errors or warnings will appear in the **Problems View**.
 - **Clean..** - Launches the Clean dialog.

- **Build Automatically** - Sets the **Build Automatically** flag on or off. When on, a check-mark appears to the left of this menu item. When this is turned on, validation of changes is done automatically each time a **Save** is done.
- **Clone Project** - Launches the **Clone Project** dialog.
- **Build Project Imports** - Reconciles all model import dependencies for models contained within the selected project.
- **Build All Imports** - Reconciles all model import dependencies for models contained within the workspace.
- **Build Packages** - TBD
- **Validate Model Transformations** - Revalidates all transformations for the selected view model.
- **Properties** - Displays the operating system's file properties dialog for the selected file.

3.7. Metadata Menu

The **Metadata** menu provides Teiid Designer-specific actions.

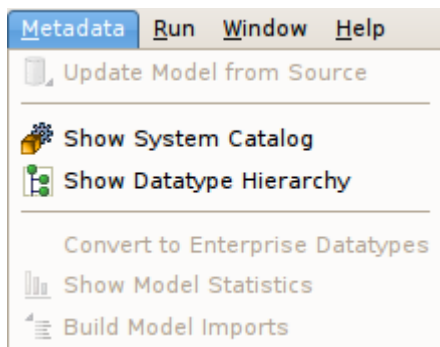


Figure 3.10. Metadata Menu

- The **Metadata** menu contains the following actions:
 - **Update Model from Source** - If the selected model is a relational source model that was originally created via JDBC Import, then the model will be updated based on changes in the database schema.
 - **Show System Catalog** - Opens the [Section 4.11, “System Catalog View”](#).
 - **Show Datatype Hierarchy** - Opens the [Section 4.11, “System Catalog View”](#).
 - **Re-resolve References** - Analyzes references within models to other model components.

- **Convert to Enterprise Datatypes** - Adds an additional property to simple datatypes within your selected schema model to label them as enterprise datatypes.
- **Show Model Statistics** - Opens the **Model Statistics** dialog for the selected model.
- **Build Model Imports** - Reconciles all model import dependencies for the selected model.

3.8. Run Menu

Teiid Designer currently does not contribute actions to the Run menu. See Eclipse documentation for details.

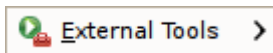


Figure 3.11. Window Menu

3.9. Window Menu

The **Window** menu shown below contains no Teiid Designer-specific actions. See Eclipse Workbench documentation for details.



Figure 3.12. Window Menu

The Preferences... action launches the Preferences dialog, which can be used to set preferences and default values for many features of Teiid Designer.

Note that these menu items may vary depending on your set of installed Eclipse features and plugins.

If you wish to customize a perspective to include one or more Teiid Designer views, select the Show View > Other... action and expand the Teiid Designer category to show the available views.

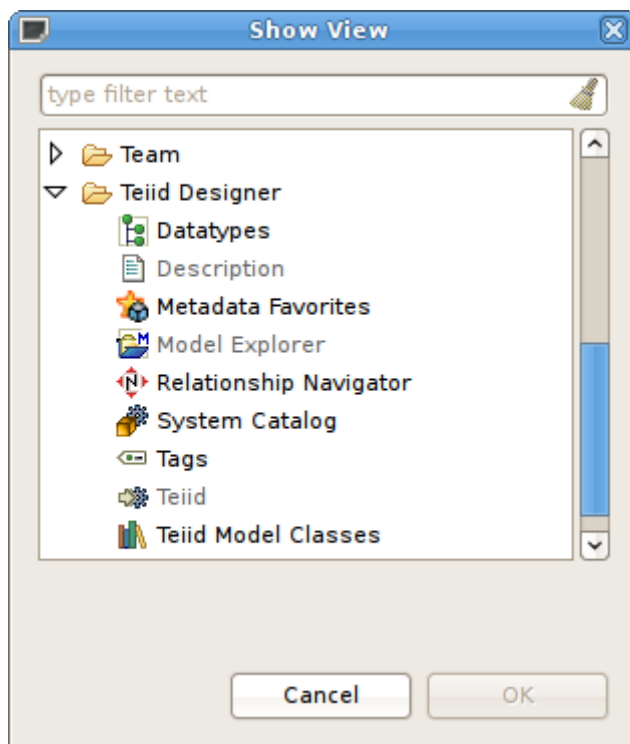


Figure 3.13. Show View Dialog

3.10. Help Menu

The **Help Menu** shown below contains no Teiid Designer-specific actions. See Eclipse Workbench documentation for details.

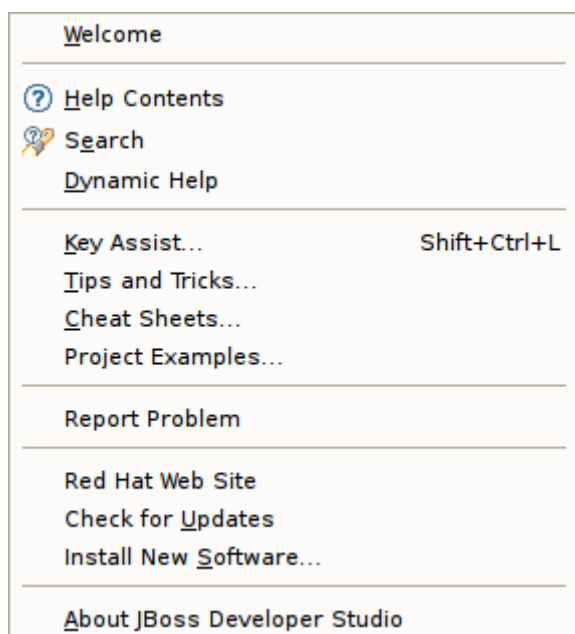




Figure 3.14. Help Menu

- The individual actions are described below:
 - *Welcome* - Shifts to the Welcome perspective, which contains links to documentation, examples and 'how-to' starting points.
 - *Help*
Contents 
- Launches the Help Window. All of Designer's online documentation is accessible from there as well.
 - *Search* 
- Launches the Help Search view, which can be used to search for phrases in the documentation.
 - *Dynamic Help* - Opens the docked dynamic help view.
 - *Key Assist (Ctrl-Shift-L) ...* - Launches a dialog describing existing key assist bindings.
 - *Tips and Tricks...* - Launches a dialog to select one of any contributed "Tips and Tricks" help pages.
 - *Cheat Sheets...* - Launches a dialog to select one of any contributed Eclipse cheat sheets.
 - *Project Examples...* - A JBoss contributed action which provides quick access to import various project examples into your workspace.
 - *Report Problem* - A JBoss contributed action which provides simple problem reporting.
 - *Check for Updates...* - provides access to retrieve updates to installed Eclipse software.
 - *Install New Software...* - provides access to install new software into your workbench.
 - *About XXXXX* - Launches the About dialog.

Teiid Designer Views

Views are dockable windows which present data from your models or your modeling session in various forms. Some views support particular [Chapter 5, Editors](#) and their content is dependent on workspace selection. This section summarizes most of the views used and available in Teiid Designer.

Below is a short summary of Teiid Designer views. The full list is presented in the main menu's *View > Show View* dialog.

Designer

- [Section 4.1, “Model Explorer View”](#)
- [Section 4.9, “Datatype Hierarchy View”](#)
- [Section 4.5, “Description View”](#)
- [Section 4.7, “Problems View”](#)
- [Section 4.3, “Teiid View”](#)
- [Section 4.11, “System Catalog View”](#)
- [Section 4.2, “Outline View”](#)
- [Section 4.4, “Properties View”](#)
- [Section 4.8, “Search Results View”](#)

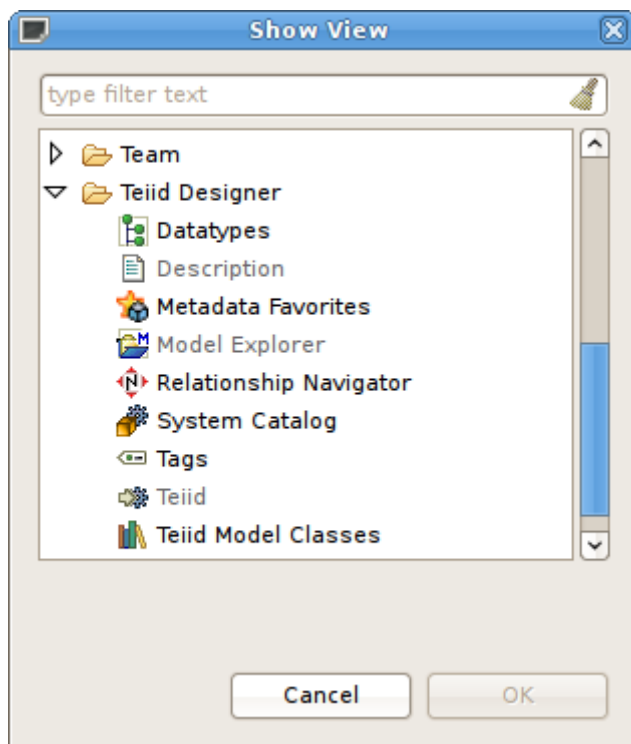


Figure 4.1. Eclipse Show View Dialog

4.1. Model Explorer View

Teiid Designer allows you manage multiple projects containing multiple models and any corresponding or dependent resources. The *Model Explorer* provides a simple file-structured view of these resources.

The **Model Explorer** (shown below) is comprised of a toolbar and a tree view.

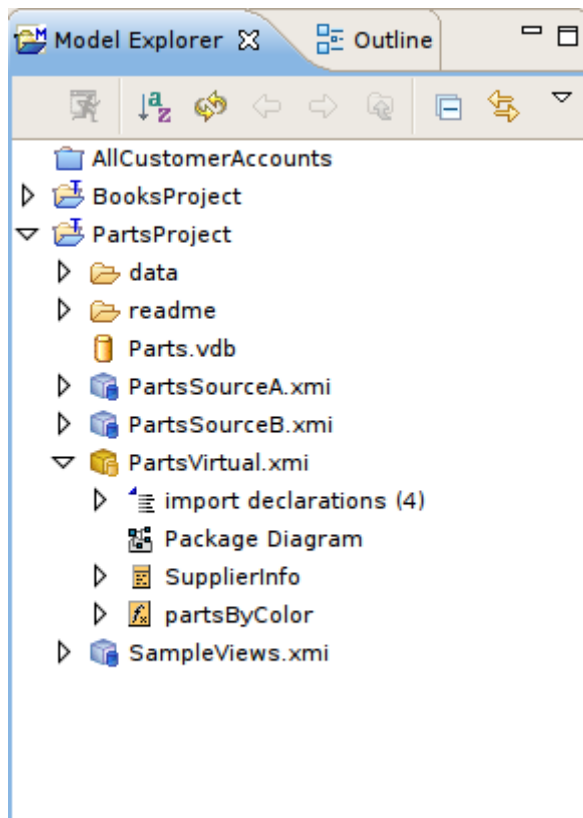











Figure 4.2. Model Explorer View

The toolbar consists of nine common actions:

- 
Preview Data - Executes a simple preview query (SELECT * FROM).
- 
Sort Model Contents - Sorts the contents of the models based on object type and alphabetizing.
- 
Refresh Markers - Refreshes error and warning markers for objects in tree.
- 
Back - Displays the last "Go Into" location. (See Eclipse Help)
- 
Forward - Displays the next "Go Into" location. (See Eclipse Help)
- 
Up - Navigates up one folder/container location. (See Eclipse Help)

- 
Collapse All - Collapses all projects.
- 
Link with Editor - When object is selected in an open editor, this option auto-selects and reveals object in Model Explorer.
- 
Additional Actions

The additional actions are shown in the following figure:

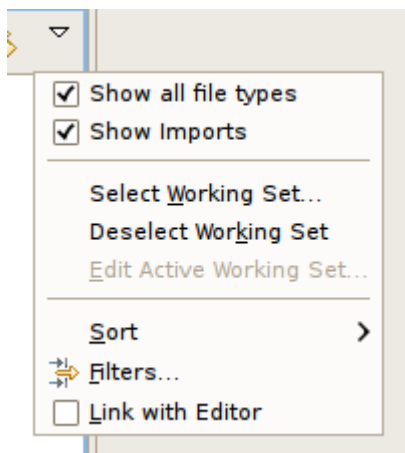


Figure 4.3. Additional Actions

If **Show Model Imports** is checked, the imports will be displayed directly under a model resource as shown below.

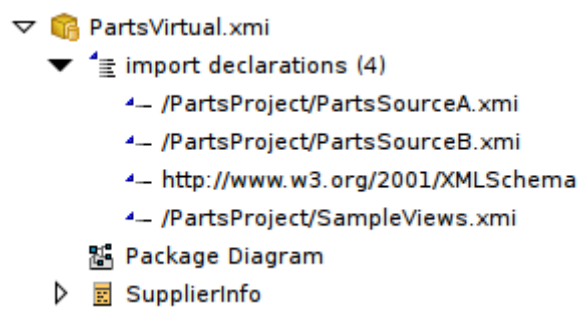


Figure 4.4. Show Model Imports Action

4.1.1. Selection-Based Action Menus

Selecting specific objects in the **Model Explorer** provides a context from which the Teiid Designer presents a customized menu of available actions.

Selecting a **view model**, for instance, results in a number of high-level options to manage edit model content, perform various operations and provides quick access to other important actions available in Teiid Designer. These may include specialized actions based on model type.

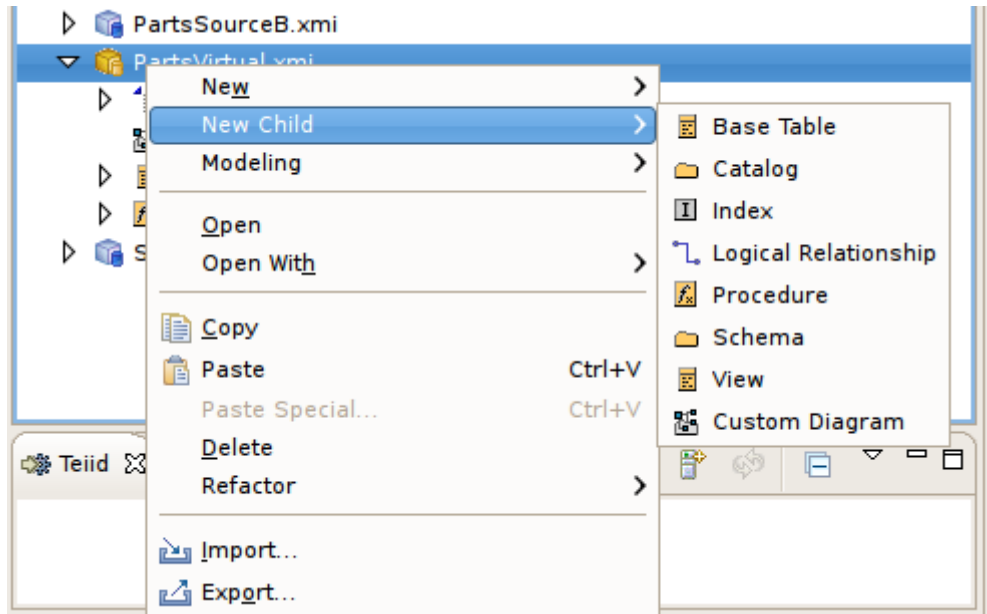


Figure 4.5. Sample Context Menu

4.2. Outline View

The Outline View is a utility view which provides both a tree view dedicated to a specific model (open in an editor) and a scaled thumbnail diagram representative of the diagram open in the corresponding Diagram Editor.

You can show the Outline View by clicking on its tab. If there is no open editors, the view indicates that Outline is not available. If a Model Editor is open, then the root of the displayed tree will be the model for the editor that is currently in focus in Teiid Designer (tab on top).

4.2.1. Outline Tree View

This tree view provides the same basic editing and navigation behavior as the Model Explorer. One additional capability is the drag and drop feature which provides re-ordering and re-parenting of objects in a model.

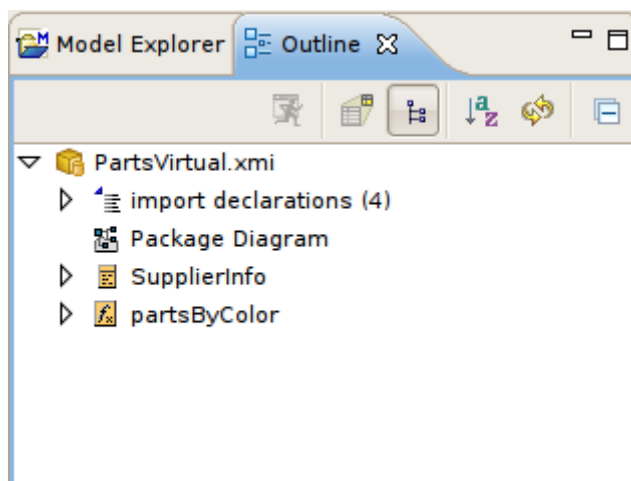


Figure 4.6. Outline View

4.2.2. Outline Thumbnail View

The Outline View also offers you a way to view a thumbnail sketch of your diagram regardless of its size. To view this diagram thumbnail from the Outline panel, click the Diagram Overview button



at the top of the view. The diagram overview displays in the Outline View.

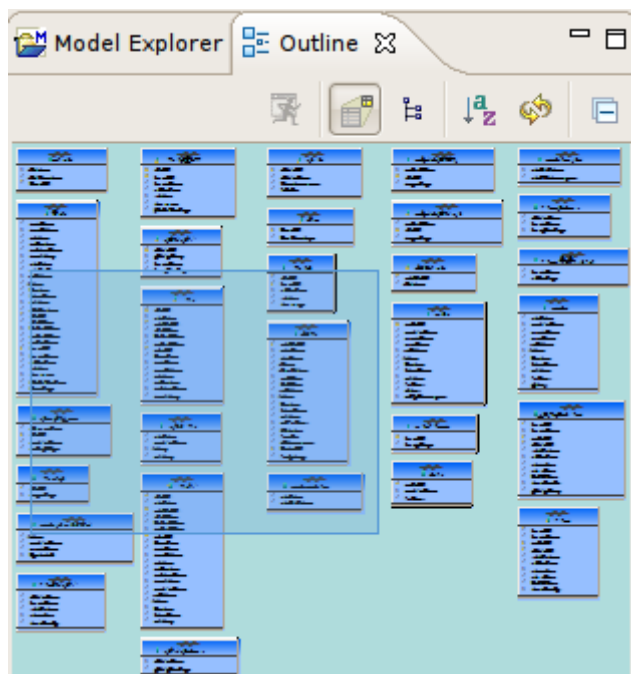
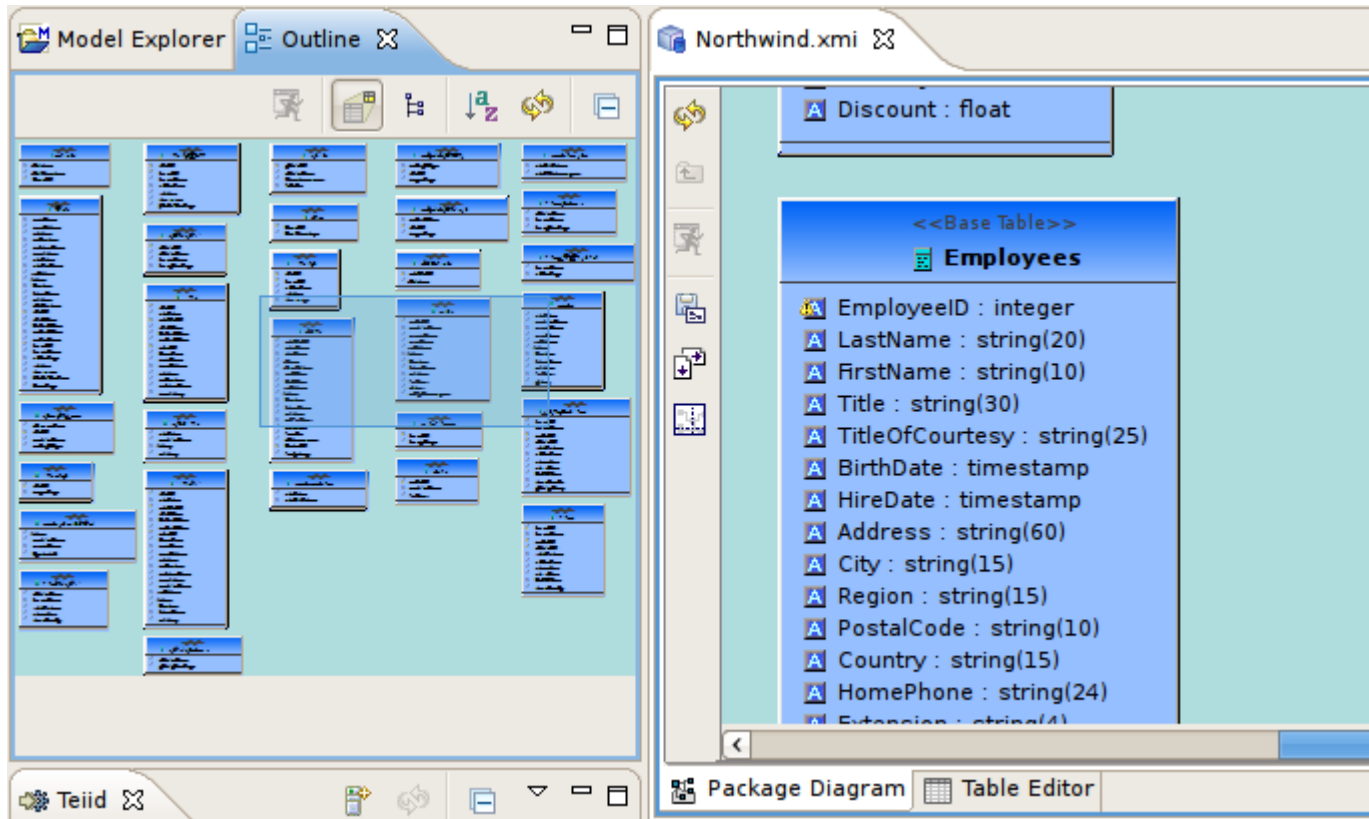


Figure 4.7. Outline View

The view contains a thumbnail of your entire diagram. The shaded portion represents the portion visible in the Diagram Editor view.

To move to a specific portion of your diagram, click the shaded area and drag to the position you want displayed in the Diagram Editor view.



4.3. Teiid View

The Teiid View provides a means to display and manage Teiid server instances and their contents within Designer.

To show the Teiid View click **"Views > Show View > Other..."** to display the Show View dialog. Choose **"Teiid Designer > Teiid"** view and hit **OK**.

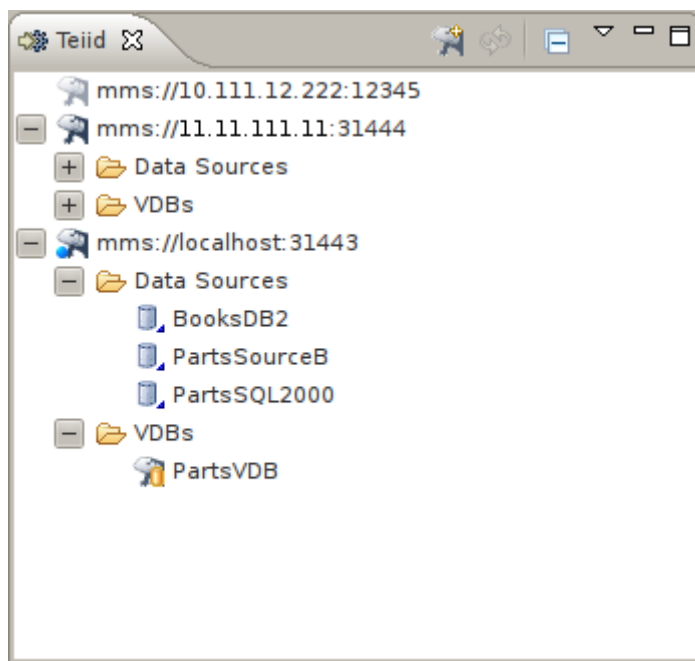


Figure 4.8. Teiid View

To create a new Teiid instance, either right-click select **New Teiid Instance** action or click the same action button,



, in the toolbar.

You'll get the **New Teiid Instance** dialog shown below.

New Teiid Instance

Enter Teiid Instance Information
Define the Teiid Instance connection information.

Host: localhost

Label

☒ Use host URL as label
☐ Use a custom label:

Teiid Admin Connection Info

Port number: 31443
User name: admin
Password: *****
URL: mms://localhost:31443

☒ Save
☒ SSL
☒ Set as default Teiid instance on 'Finish' Test

Teiid JDBC Connection Info

Port number: 31000
User name: user
Password: ****
URL: jdbc:teiid:<vdbname>@mm://localhost:31000

☒ Save
☐ SSL

Cancel Finish









Figure 4.9. New Teiid Instance Dialog

The dialog contains two three sections. The top panel contains host name and an option to customize the new Teiid instance label in the teiid view. The second panel **Teiid Admin Connection Info**, pertains to the connection information required to connect to the admin URL of your installed Teiid instance. The operations Designer exposes deal with deploying and undeploying VDBs as well as managing your test data sources required by those VDBs to successfully connect and query data through the Teiid runtime engine.

The third panel, **Teiid JDBC Connection Info**, provides for entering the connection information for that same Teiid instance. This information is required for Designer to make JDBC connections during execution of the Preview Data feature.

Enter valid **Host**, **Port**, **User name** and **Password** information, edit any options and click *Finish*.

Actions available in this view include:

-  **New Teiid Instance** - Create a new instance of a running Teiid server
-  **Teiid Server Properties** - View and edit properties of an existing Teiid instance
-  - Reconnect and refresh contents of the selected Teiid instance
-  **Delete** - Disconnect and delete the selected Teiid instance
-  **Execute VDB** - Creates a JDBC Teiid connection profile and opens the Data Tools Database Development perspective
-  **Undeploy VDB** - Removes the selected VDB from the Teiid instance
-  **Create Data Source** - Launches the New Data Source wizard
-  **Delete Data Source** - Removes the selected Data Source from the Teiid instance

If you chose the **Use a custom label** option, the text entry field will enable as shown below.

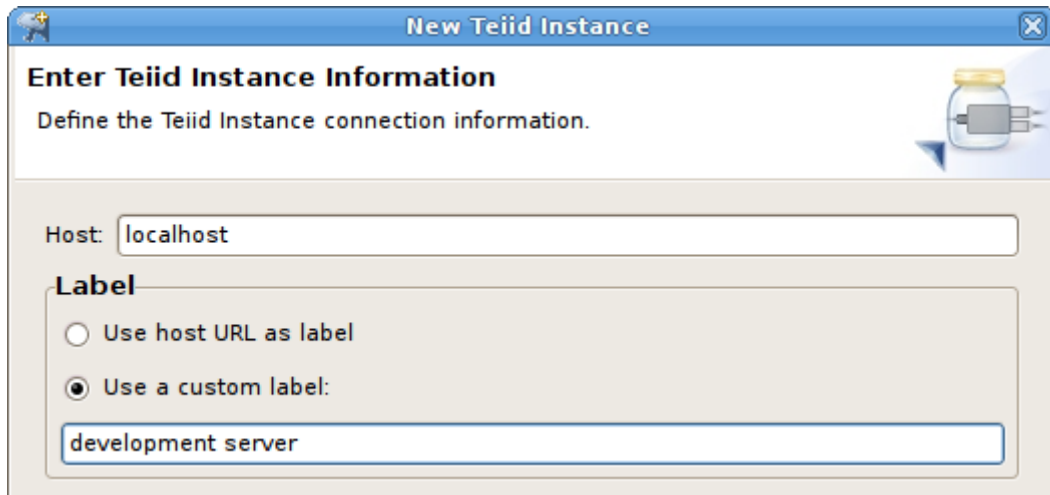


Figure 4.10. Use a Custom Label Option

4.4. Properties View

The **Properties View** provides editing capabilities for the currently selected object in Teiid Designer. The selection provided by whichever view or editor is currently in focus will determine the its contents.

To edit a property, click a cell in the Value column. As in the **Table Editor**, each cell provides a UI editor specific to the property type.

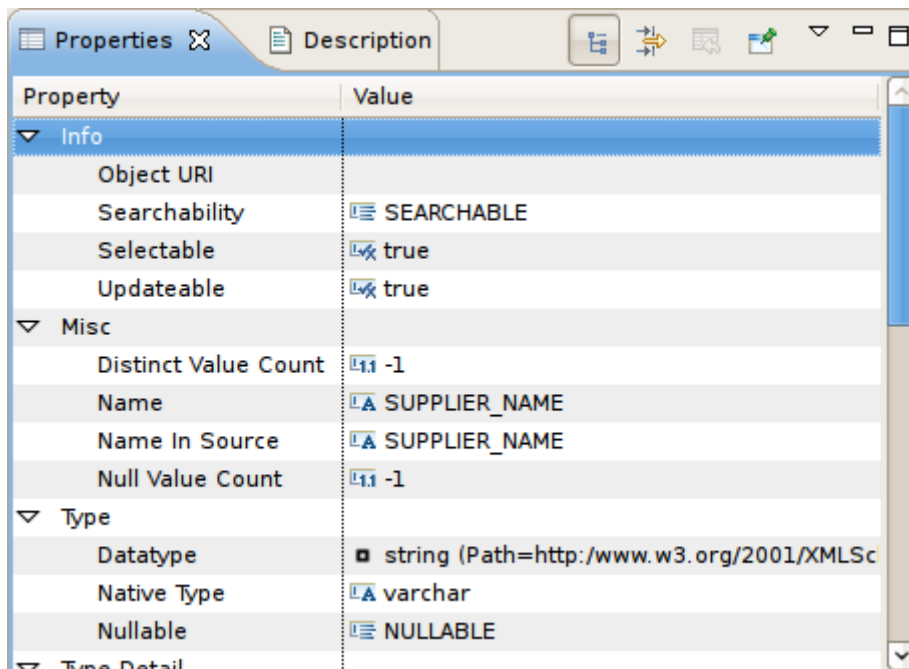


Figure 4.11. Properties View

If the model for the object being edited is not open in an editor, a dialog may appear confirming the attempt to modify the model and asking the user to confirm or cancel. This dialog can be prevented

by checking the preference Always open editor without prompting. You can re-set/uncheck this property via the Teiid Designer's main preference page.

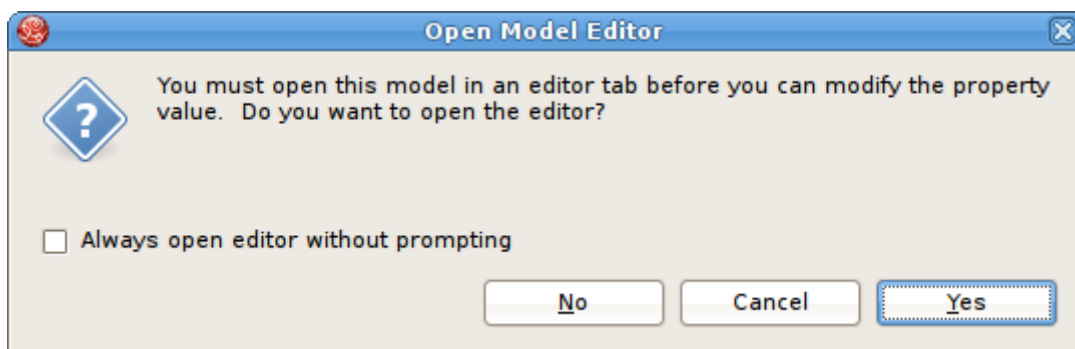


Figure 4.12. Open Model Editor Dialog

Properties can also be edited via a right-click menu presented below.

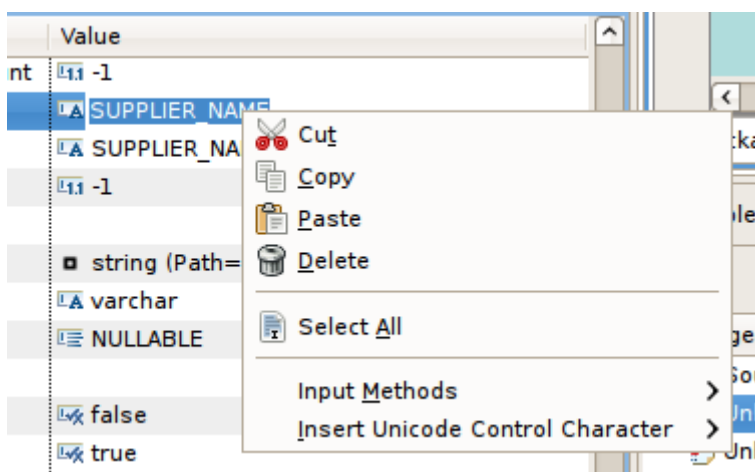





Figure 4.13. Open Model Editor Dialog

The **Properties** toolbar contains the following actions:

-  **Show Categories** - toggles between categorized properties and flat alphabetical properties list.
-  **Show Advanced Properties** - shows/hide advanced properties (if available).
-  **Restore Default Value** - for a selected property, this action will reset the current to a default value (if available).

4.5. Description View

The **Description View** provides a means to display and edit (add, change or remove) a description for any model or model object. To show the Description View click **Views > Show View > Other...** to display the [Figure 4.1, “Eclipse Show View Dialog”](#) dialog. Choose **Teiid Designer > Description** view and hit **OK** .

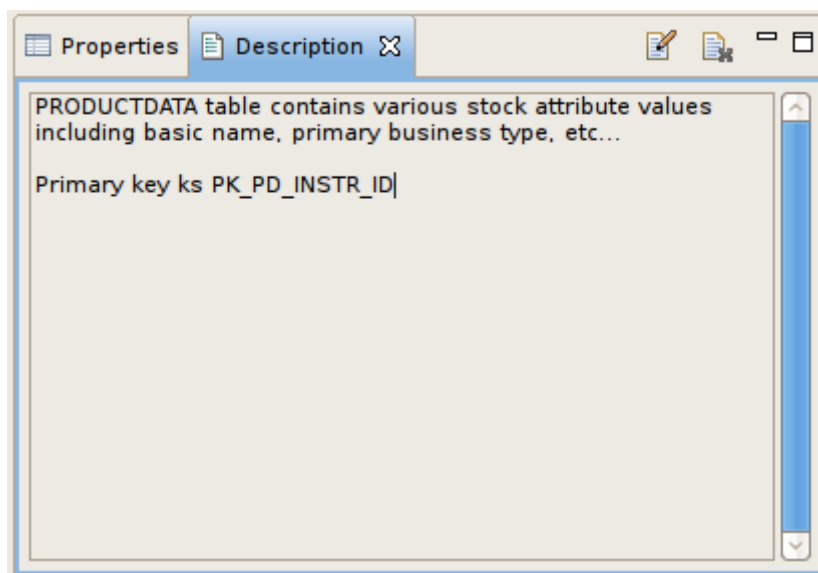


Figure 4.14. Description View

You can click the edit description action in the toolbar or right-click select "Edit" in the context menu to bring up the Edit Description dialog. edit actions.

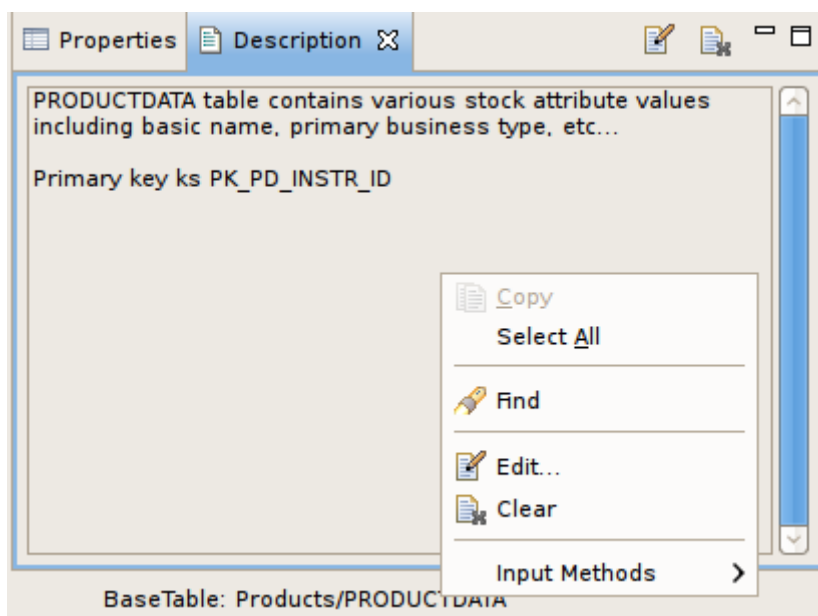


Figure 4.15. Description View Context Menu

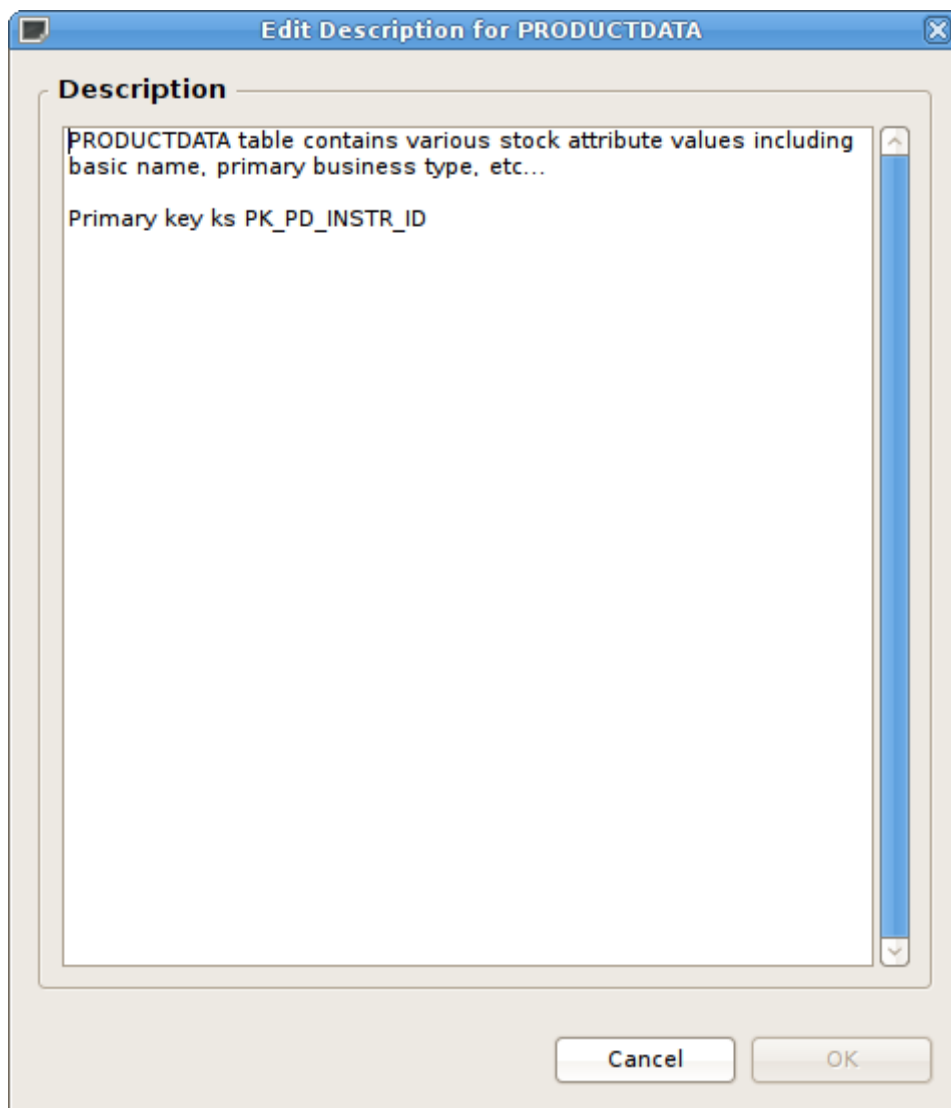


Figure 4.16. Edit Description Dialog

4.6. Editors

Editors are the UI components designed to assist editing your models and to maintain the state for a given model or resource in your workspace. When editing a model, the model will be opened in a **Model Editor**. Editing a property value, for instance, will require an open editor prior to actually changing the property.

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for Teiid Designer may contain operations that are applicable to the active editor (and removed when editor becomes inactive).

Tabs in the editor area indicate the names of models that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.

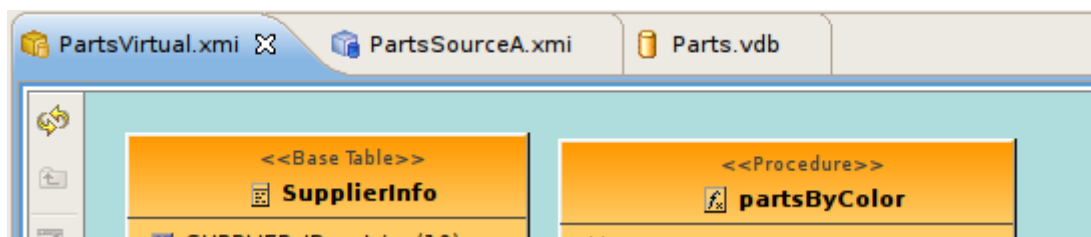


Figure 4.17. Editor Tabs

By default, editors are stacked in the editors area, but you can choose to tile them vertically, and or horizontally in order to view multiple models simultaneously.

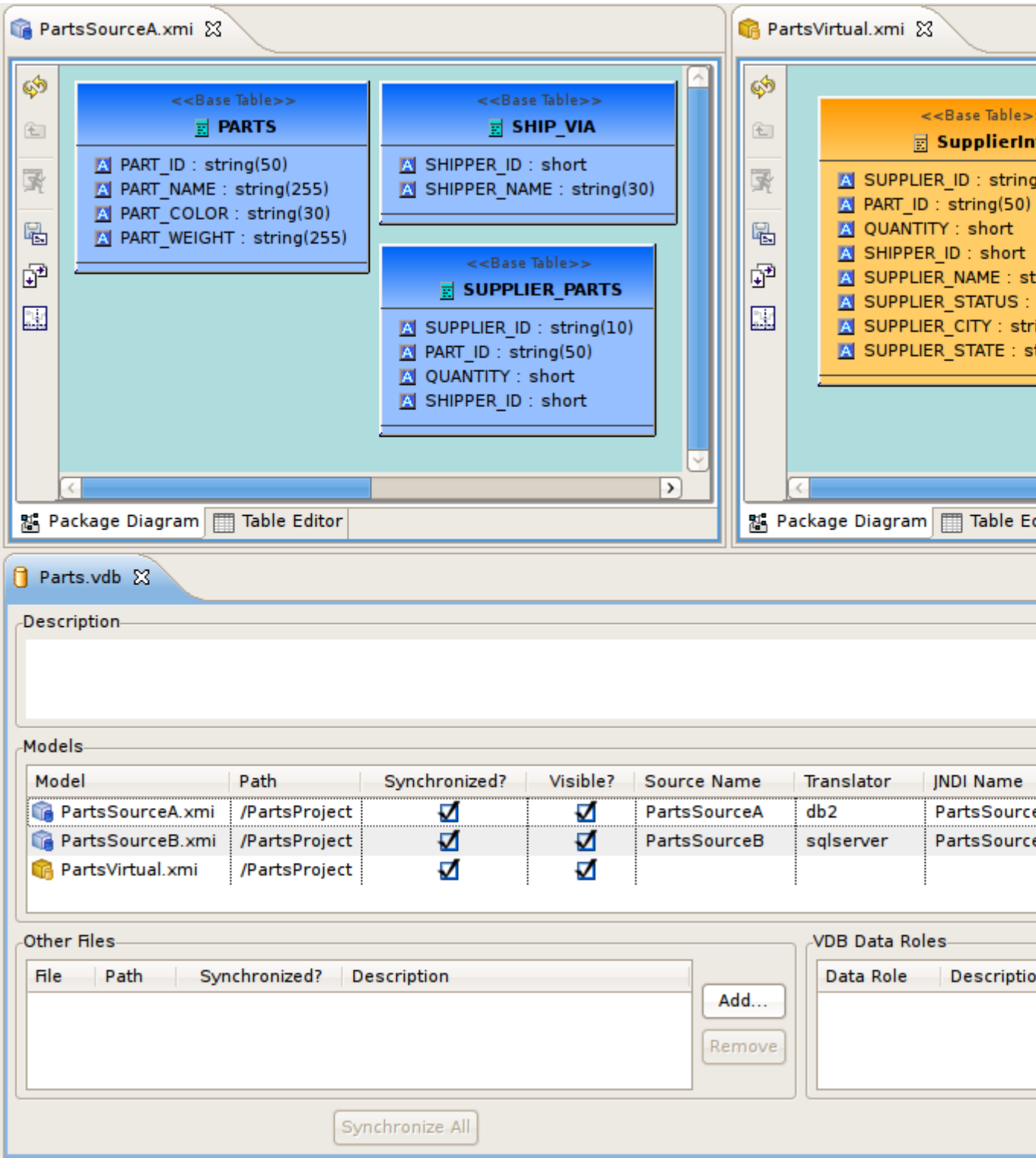


Figure 4.18. Viewing Multiple Editors

The Teiid Designer provides main editor views for XMI models and VDBs.

The Model Editor contains sub-editors which provide different views of the data or parts of data within an XMI model. These sub-editors, specific to model types are listed below.

- **Diagram Editor** - All models except XML Schema models.
- **Table Editor** - All models.
- **Simple Datatypes Editor** - XML Schema models only.
- **Semantics Editor** - XML Schema models only.
- **Source Editor** - XML Schema models only.

The *VDB Editor* is a single page editor containing panels for editing description, model contents and data roles.

In addition to general Editors for models, there are detailed editors designed for editing specific model object types. These object editors include:

- **Transformation Editor** - Manages Transformation SQL for Relational View Base Tables, Procedures and XML Web Service Operations.
- **Choice Editor** - Manages properties and criteria for XML choice elements in XML Document View models.
- **Input Editor** - Manages Input Set parameters used between Mapping Classes in XML Document View models.
- **Recursion Editor** - Manages recursion properties for recursive XML Elements in XML Document View models.
- **Operation Editor** - Manages SQL and Input Variables for Web Service Operations.

4.7. Problems View

The **Problems View** displays validation errors, warnings, or information associated with a resource contained in open projects within your workspace.

Description	Resource	Path	Location	Type
Errors (1 item)				
SQL statement is empty	Books.xml	/BooksProject	bookCollection	Pro
Warnings (1 item)				
The type referenced by column book_edition must be marked as an enterprise datatype	Books.xml	/BooksProject	bookCollection/t	Pro

Figure 4.19. Problems View

By default, the **Problems View** is included in the Teiid Designer perspective. If the **Problems View** is not showing in the current perspective click **Views > Show View > Other > Teiid Designer > Problems**.

There are 5 columns in the view's table which include:

1. **Description** - A description of the problem preceded by a severity icon (i.e., error



warning



or



info

).

2. **Resource** - The name of the resource.
3. **Path** - The project name.
4. **Location** - The object within the resource that has a validation error.
5. **type** - Type of validation item.

4.7.1. Toolbar Menu

Click the upside-down triangle



icon to open the View Menu icon to see various options including sorting, filtering, displayed columns and much more.

4.7.2. Context Menu

Additional actions are available by selecting a problem and right-click to open a context menu.

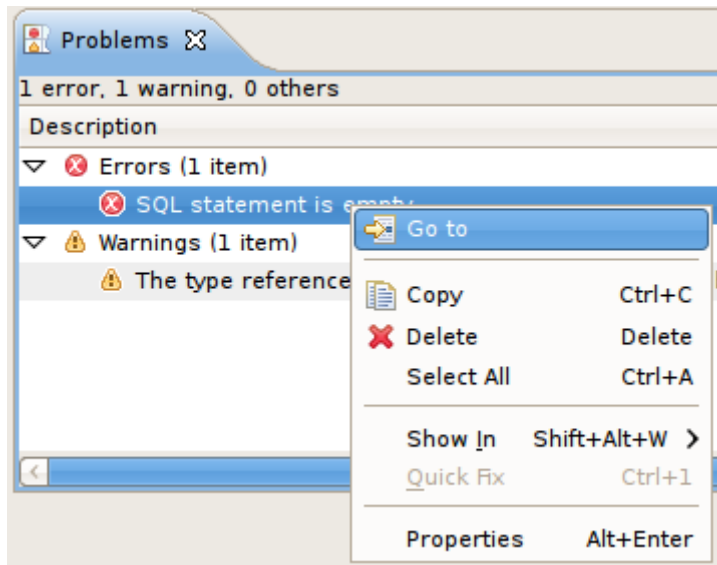


Figure 4.20. Problems View Context Menu

- **Go To** - will open the appropriate editor and select the affected/referenced object.
- **Show In Navigator** - Opens the Basic > Navigator view (if not open) and expands file system tree and reveals applicable resource.
- **Copy** - Copies the problem information to the system clipboard.
- **Paste** - Pastes the problem information located in the system clipboard (if applicable) into the cursor location for a specified text editor.
- **Delete** - Deletes the selected problem rows (if applicable).
- **Select All** - selects all problems in the table.
- **Quick Fix** - (Not yet implemented in Teiid Designer).
- **Properties** - displays a dialog containing additional information.

4.8. Search Results View

Below is an example set of search results. The view contains rows representing matches for your search parameters. You can double-click a entry and the object will be opened and selected in an editor and/or the Model Explorer if applicable.

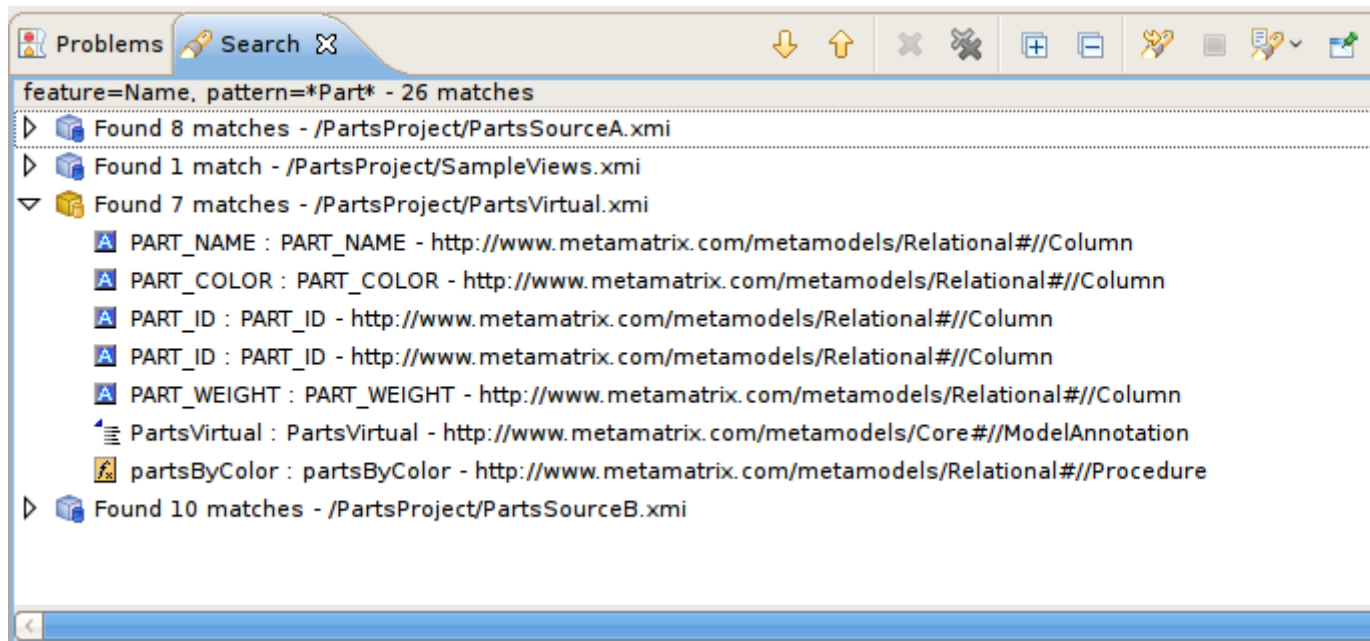








Figure 4.21. Search Results View

The toolbar actions for the **Search Results** view are:

- 
Show Next Match - Navigates down one row in the view.
- 
Show Previous Match - Navigates up one row in the view.
- 
Remove Selected Matches - Removes selected results from the view.
- 
Remove All Matches - Clears the view.
- 
Search - Launches the **MoTeiid Designerearch Dialog**.
- 
Previous Search Results - Select previous search results from history.

You can also perform some of these actions via the right-click menu:

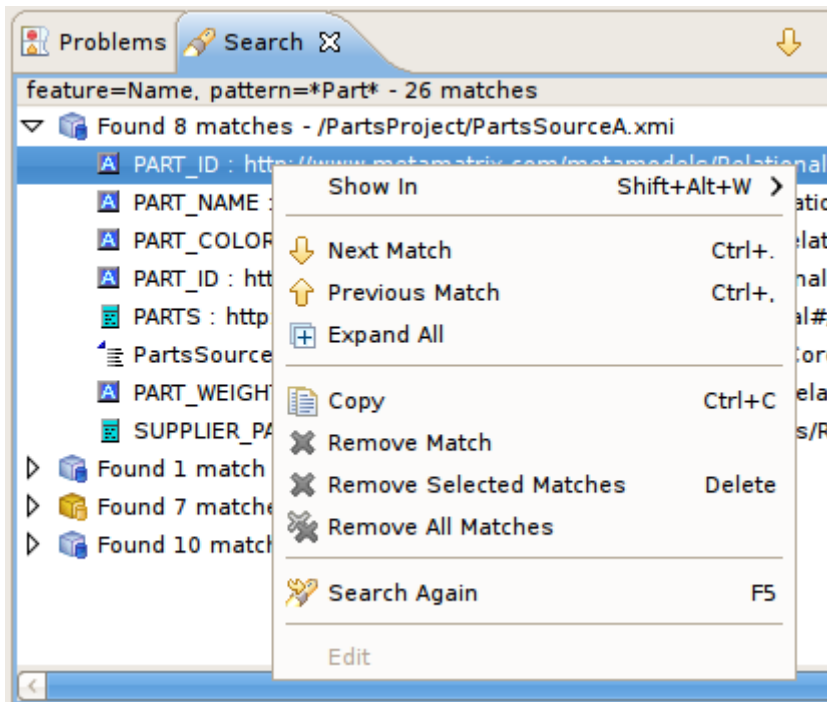


Figure 4.22. Search Results Context Menu

4.9. Datatype Hierarchy View

To open **Teiid Designer's Datatype Hierarchy** view, select the main menu's **Views > Show View > Other...** and select the **Teiid Designer > Datatypes** view in the dialog.

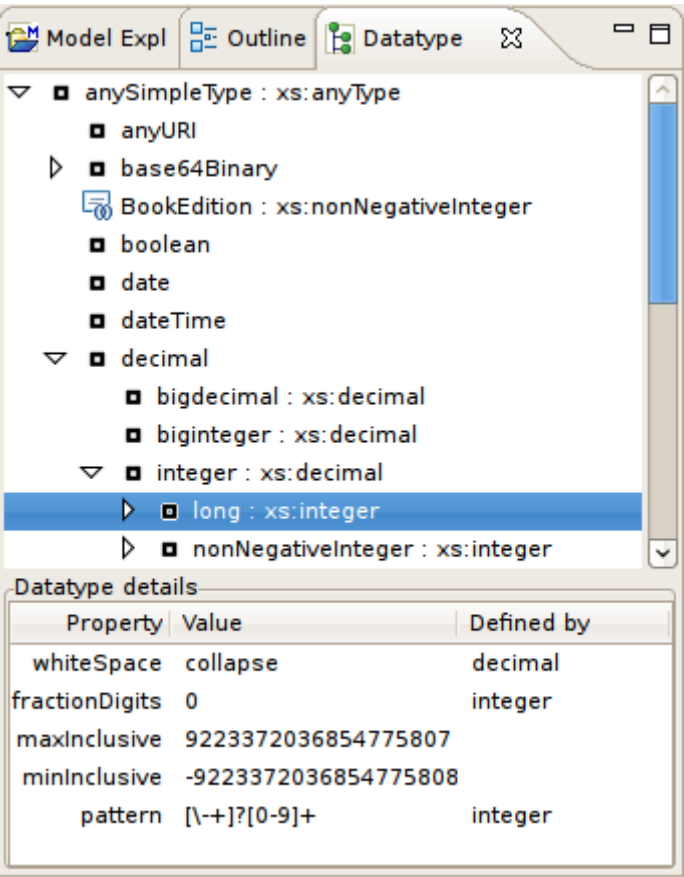


Figure 4.23. Datatype Hierarchy View

4.10. Teiid Model Classes View

The Model Classes View provides a hierarchical EMF-centric view of the various metamodel classes available within Teiid Designer. This view is primarily for informational purposes, but can be used as a reference if creating relationships or searching your workspace for specific metamodel constructs.

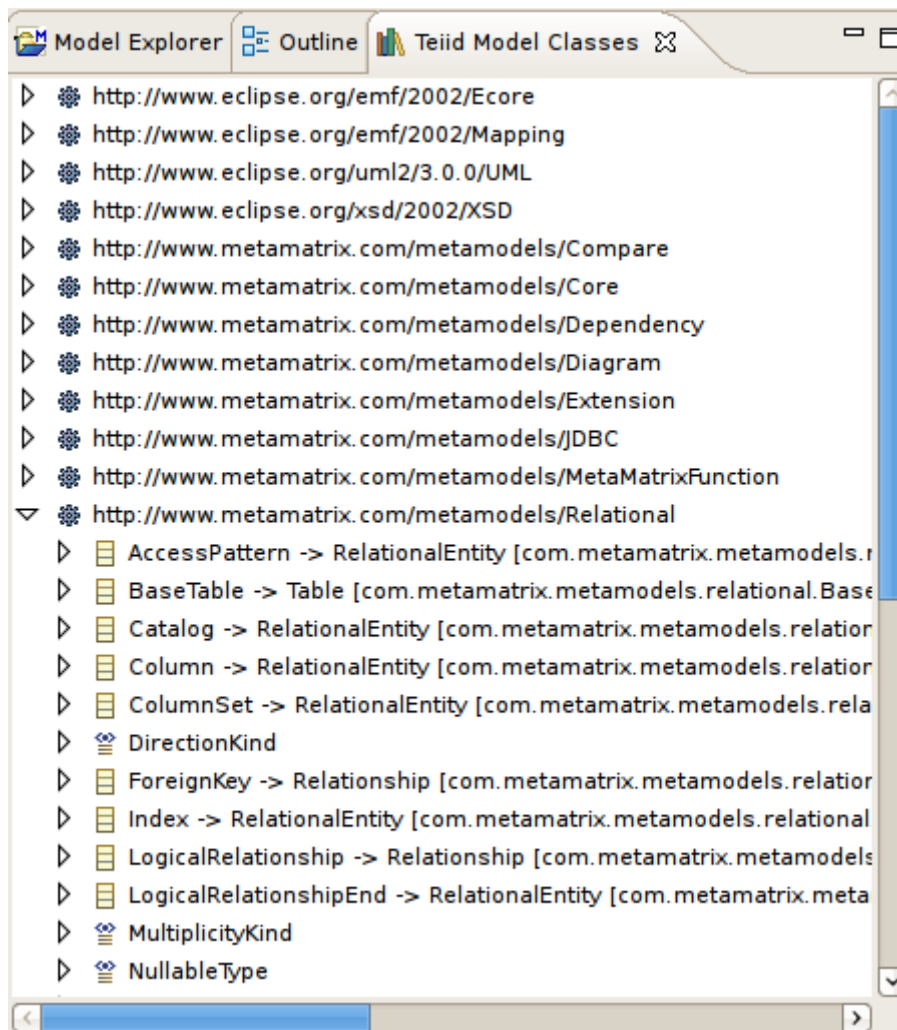


Figure 4.24. Datatype Hierarchy View

4.11. System Catalog View

To open **Teiid Designer's System Catalog** view, select the main menu's **Views > Show View > Other...** and select the **Teiid Designer > System Catalog** view in the dialog..

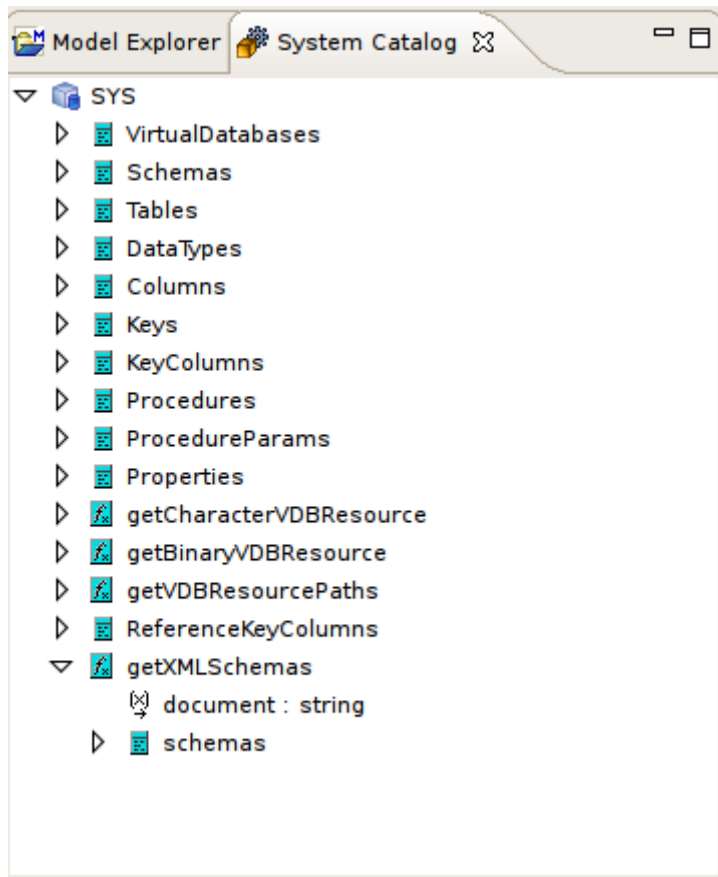


Figure 4.25. System Catalog View

Editors

Editors are the UI components designed to assist editing your models and to maintain the state for a given model or resource in your workspace. When editing a model, the model will be opened in a **Model Editor**. Editing a property value, for instance, will require an open editor prior to actually changing the property.

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for Teiid Designer may contain operations that are applicable to the active editor (and removed when editor becomes inactive).

Tabs in the editor area indicate the names of models that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.

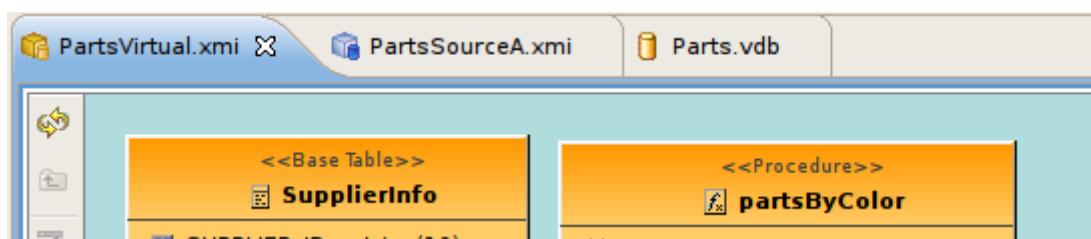


Figure 5.1. Editor Tabs

By default, editors are stacked in the editors area, but you can choose to tile them vertically, and or horizontally in order to view multiple models simultaneously.

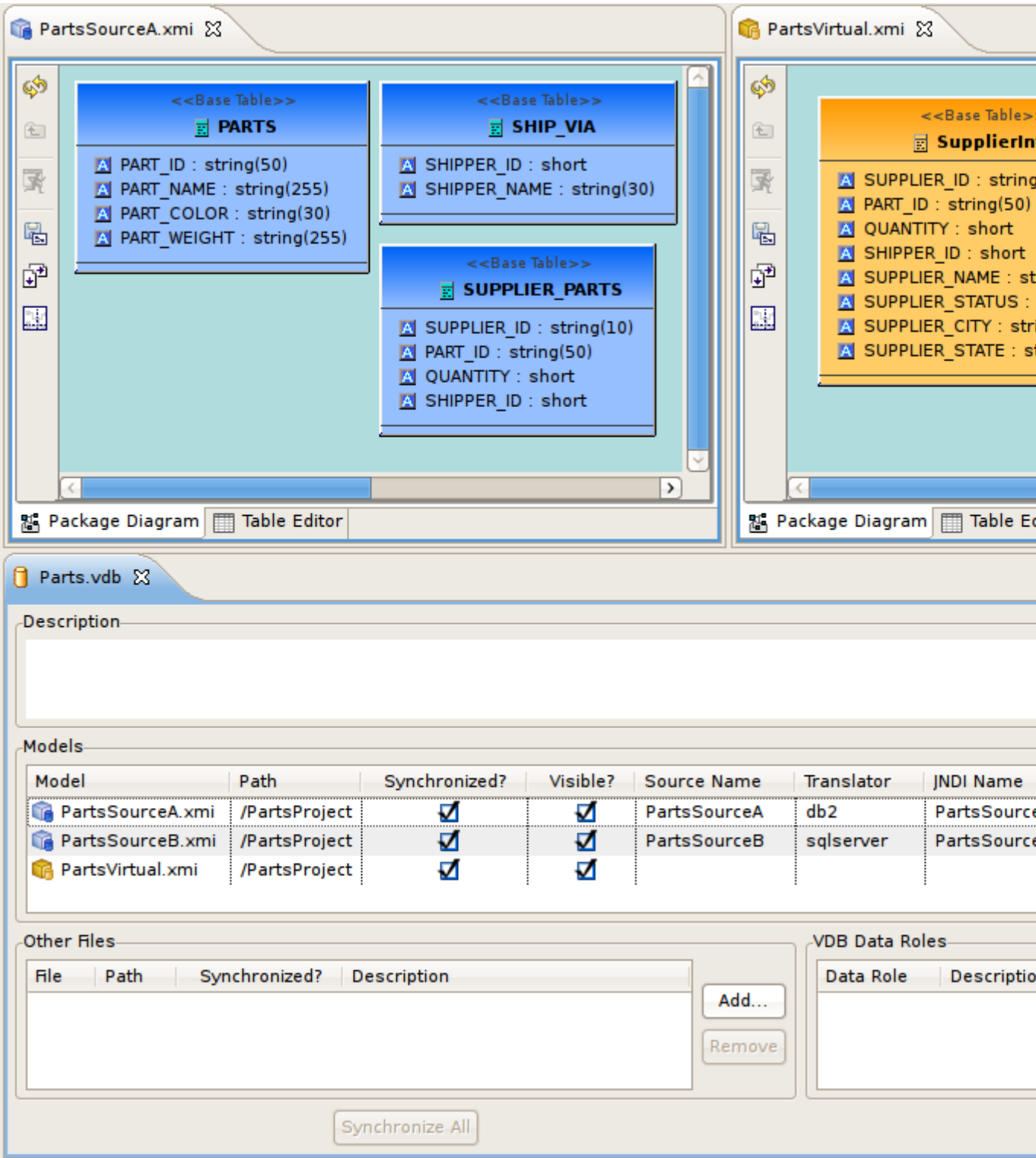


Figure 5.2. Viewing Multiple Editors

The Teiid Designer provides main editor views for XMI models and VDBs.

The Model Editor contains sub-editors which provide different views of the data or parts of data within an XMI model. These sub-editors, specific to model types are listed below.

- **Diagram Editor** - All models except XML Schema models.
- **Table Editor** - All models.
- **Simple Datatypes Editor** - XML Schema models only.
- **Semantics Editor** - XML Schema models only.
- **Source Editor** - XML Schema models only.

The *VDB Editor* is a single page editor containing panels for editing description, model contents and data roles.

In addition to general Editors for models, there are detailed editors designed for editing specific model object types. These "object" editors include:

- **Transformation Editor** - Manages Transformation SQL for Relational View Base Tables, Procedures and XML Web Service Operations.
- **Choice Editor** - Manages properties and criteria for XML choice elements in XML Document View models.
- **Input Editor** - Manages Input Set parameters used between Mapping Classes in XML Document View models.
- **Recursion Editor** - Manages recursion properties for recursive XML Elements in XML Document View models.
- **Operation Editor** - Manages SQL and Input Variables for Web Service Operations.

5.1. Model Editor

The Model Editor is comprised of sub-editors which provide multiple views of your data. The Diagram Editor provides a graphical while the Table Editor provides spreadsheet-like editing capabilities. This section describes these various sub-editors.

5.1.1. Diagram Editor

The Diagram Editor provides a graphical view of the a set of model components and their relationships.

Several types of diagrams are available depending on model type. They include:

- 
Package Diagram



Custom Diagram



Transformation Diagram



Mapping Diagram



Mapping Transformation Diagram

You can customize various diagram visual properties via Diagram Preferences.

Each diagram provides actions via the Main toolbar, diagram toolbar and selection-based context menus. These actions will be discussed below in detail for each diagram type.

When a **Diagram Editor** is in focus, a set of common diagram actions is added to the application's main toolbar.



Figure 5.3. Main Toolbar Diagram Actions

- The actions include:



Zoom In



Zoom to Level



Zoom Out



Increase Font Size



Decrease Font Size



Perform Diagram Layout

5.1.1.1. Package Diagram

The Package Diagram provides a graphical view of the contents of a model container, be it the model itself, a relational catalog or schema.

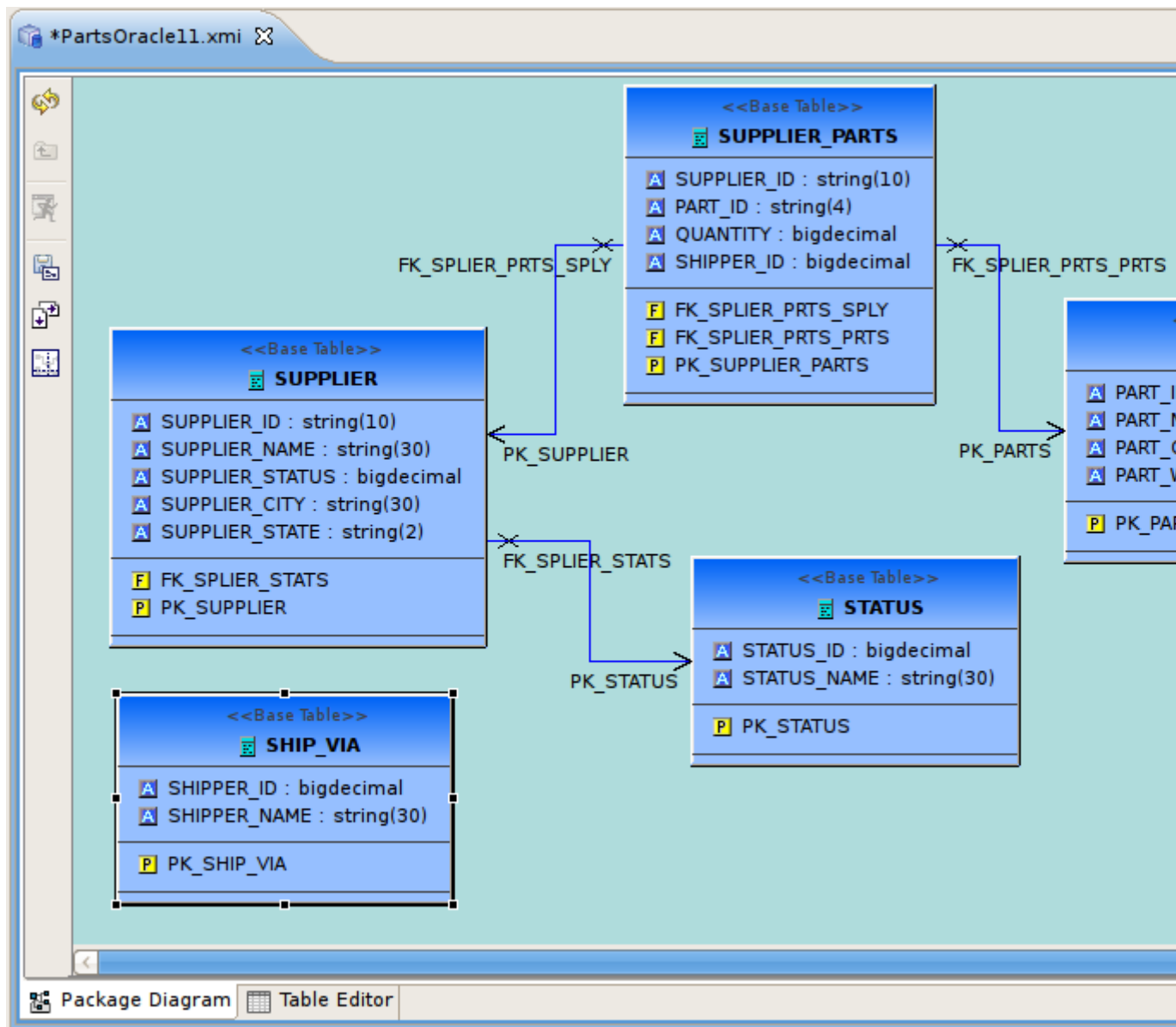









Figure 5.4. Package Diagram Example

- Package Diagram toolbar actions include:

- 
Refresh Diagram - Re-draws diagram.
- 
Show Parent Diagram - Navigates to diagram for parent object (if available).
- 
Preview Data - Executes a simple preview query (SELECT * FROM).
- 
Save Diagram as Image - Save the diagram image to file in JPG or BMP format.
- 
Modify Diagram Printing Preferences - Modify page layout information for printing diagrams. Includes margins, orientation, etc...
- 

Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Context menus provide a flexible means to edit model data, especially from Package Diagrams. Each Package Diagram represents the contents of some container (i.e. Model, Category, Schema, etc...), so New Child, New Sibling and New Association actions are almost always available in addition to standard Edit actions (Delete, Cut, Copy, Paste, Rename, Clone).

A sample context menu for a relational base table is shown below.

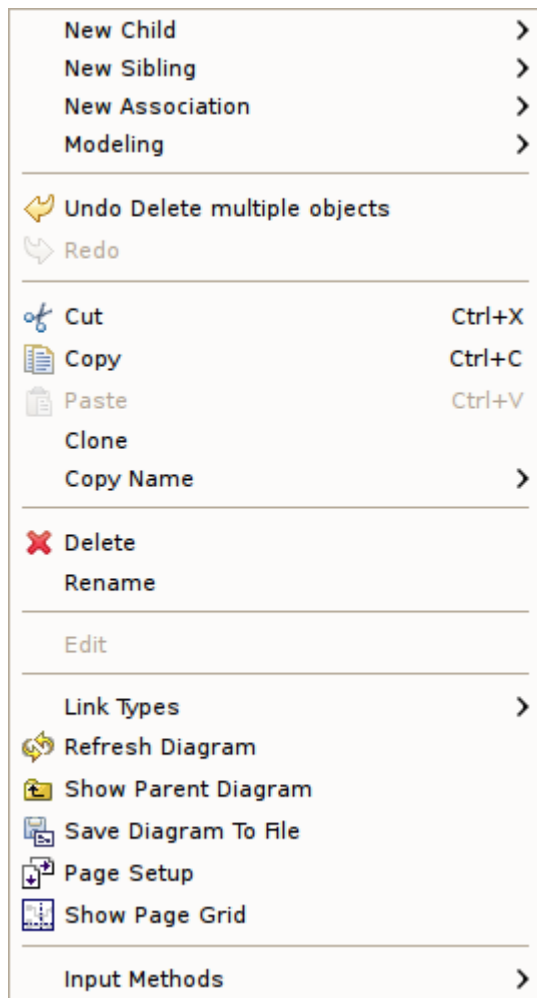


Figure 5.5. Package Diagram Context Menu

5.1.1.2. Custom Diagram

The **Custom Diagram** represents a view of user-defined model objects. Unlike **Package Diagrams**, **Custom Diagrams** can contain objects that are not only unrelated, but can be from different containers and even models.

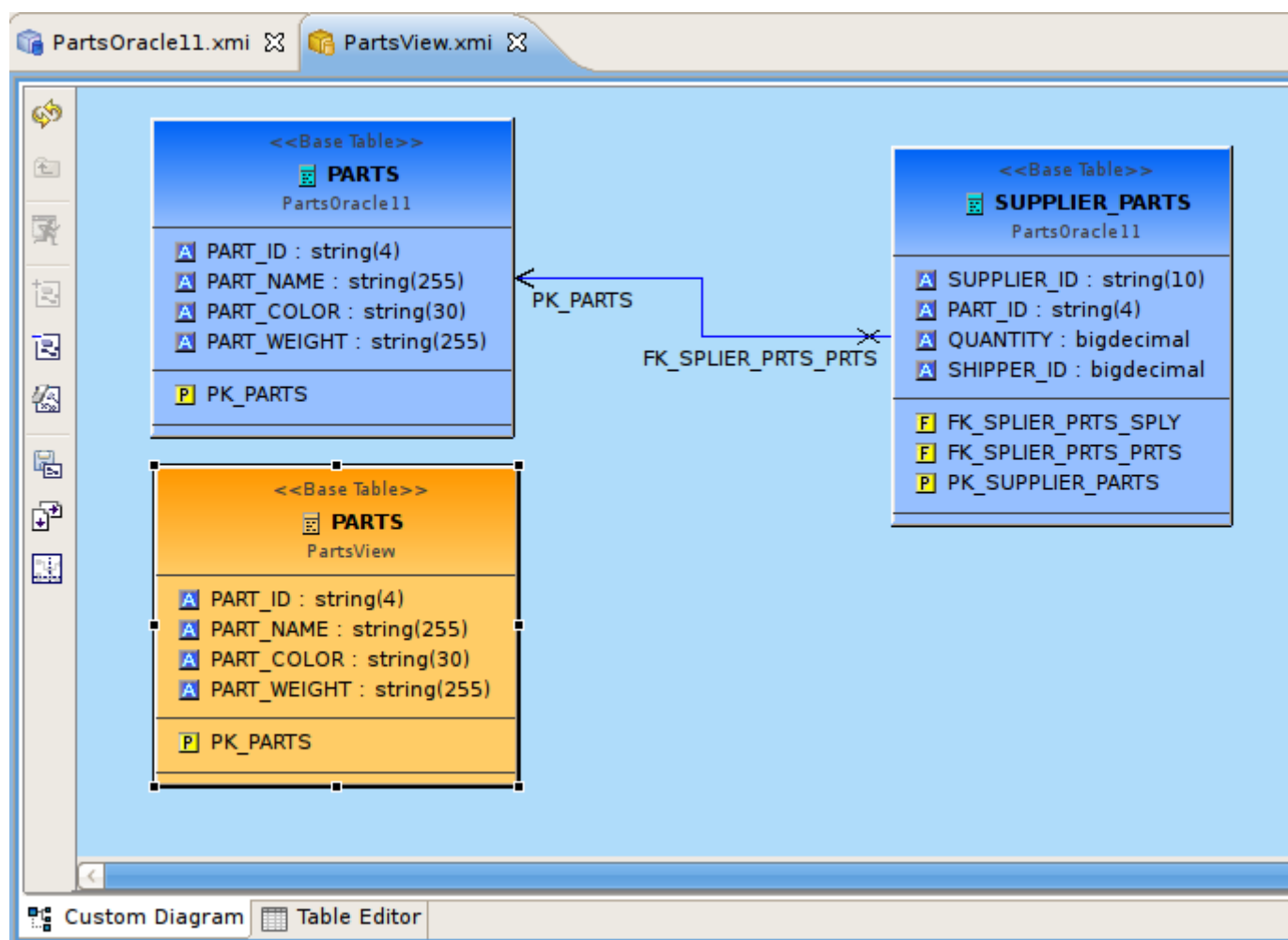












Figure 5.6. Package Diagram Example

- Custom Diagram toolbar actions include:
 -  **Refresh Diagram** - Re-draws diagram.
 -  **Show Parent Diagram** - Navigates to diagram for parent object (if available).
 -  **Preview Data** - Executes a simple preview query (SELECT * FROM).
 -  **Add To Diagram** - Add objects selected in Model Explorer to diagram.

-  **Remove From Diagram** - Removed objects selected in diagram from diagram.
-  **Clear Diagram** - Remove all objects from diagram.
-  **Save Diagram as Image** - Save the diagram image to file in JPG or BMP format.
-  **Modify Diagram Printing Preferences** - Modify page layout information for printing diagrams. Includes margins, orientation, etc...
- 
 **Show/Hide Page Grid** - Show current page boundaries as grid in diagram.

Since **Custom Diagrams** do not represent the contents of container objects(i.e. Model, Category, Schema, etc...) its **context menus** are limited to adding/removing objects from diagram and basic diagram-related display options.

5.1.1.3. Transformation Diagram

The **Transformation Diagram** represents a view of the relationships defined by the source inputs described in a view table's SQL transformation.

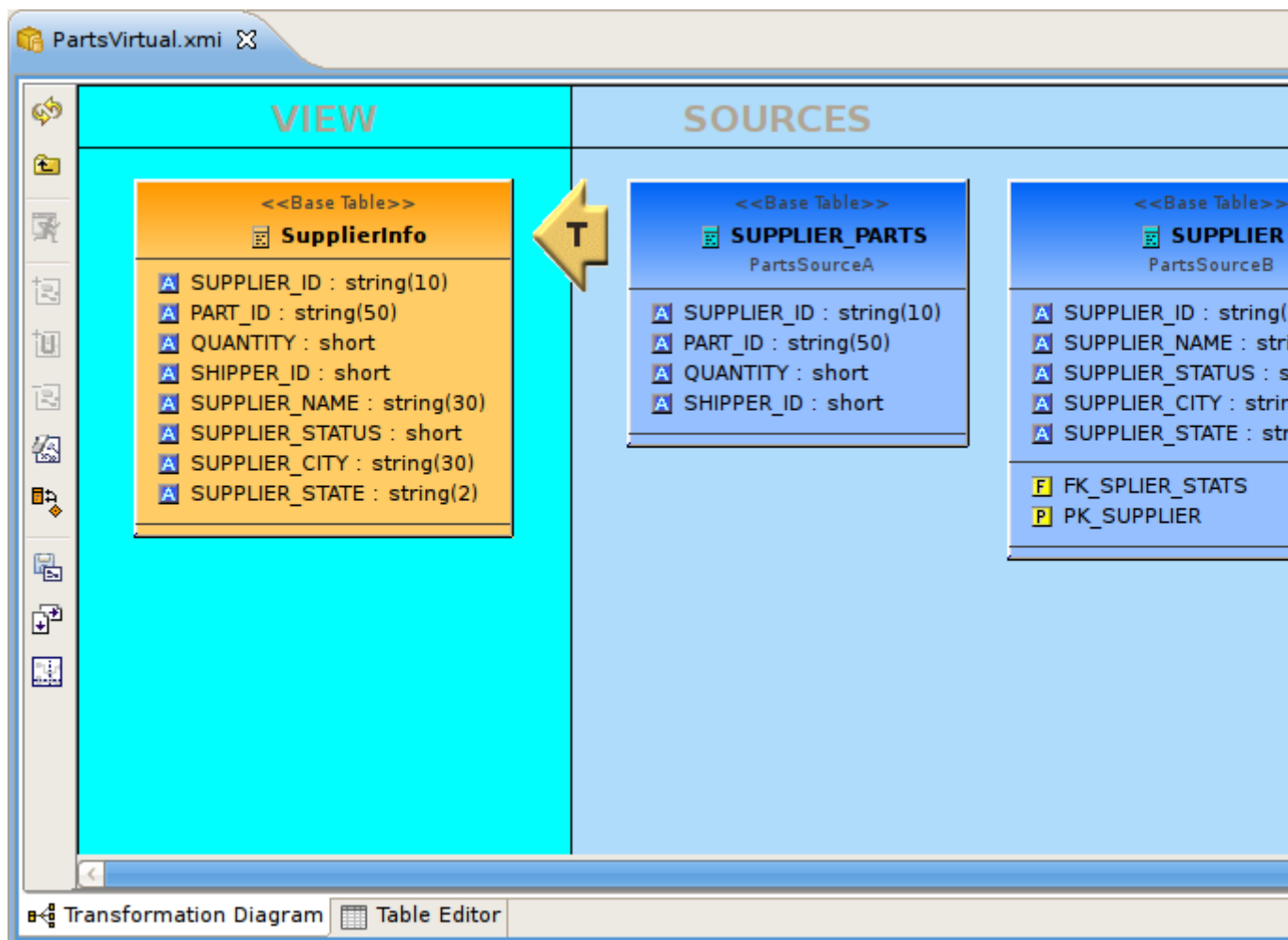






Figure 5.7. Transformation Diagram Example

- Transformation Diagram toolbar actions include:
 -  **Refresh Diagram** - Re-draws diagram.
 -  **Show Parent Diagram** - Navigates to diagram for parent object (if available).
 -  **Preview Data** - Executes a simple preview query (SELECT * FROM).
 -  **Add Transformation Sources** - Add selected sources to transformation.



Add Union Transformation Sources - Add selected sources as union sources.



Remove Transformation Sources - Removed sources selected in diagram from transformation.



Clear Transformation - Remove all sources from transformation.



Open Transformation Reconciler dialog



Save Diagram as Image - Save the diagram image to file in JPG or BMP format.



Modify Diagram Printing Preferences - Modify page layout information for printing diagrams. Includes margins, orientation, etc...



Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Context menus for the

5.1.1.4. Mapping Diagram

The **Mapping Diagram** represents a view of the mapping between virtual mapping class columns and XML document elements. This mapping defines how source data is transformed from row-based results into XML formatted text.

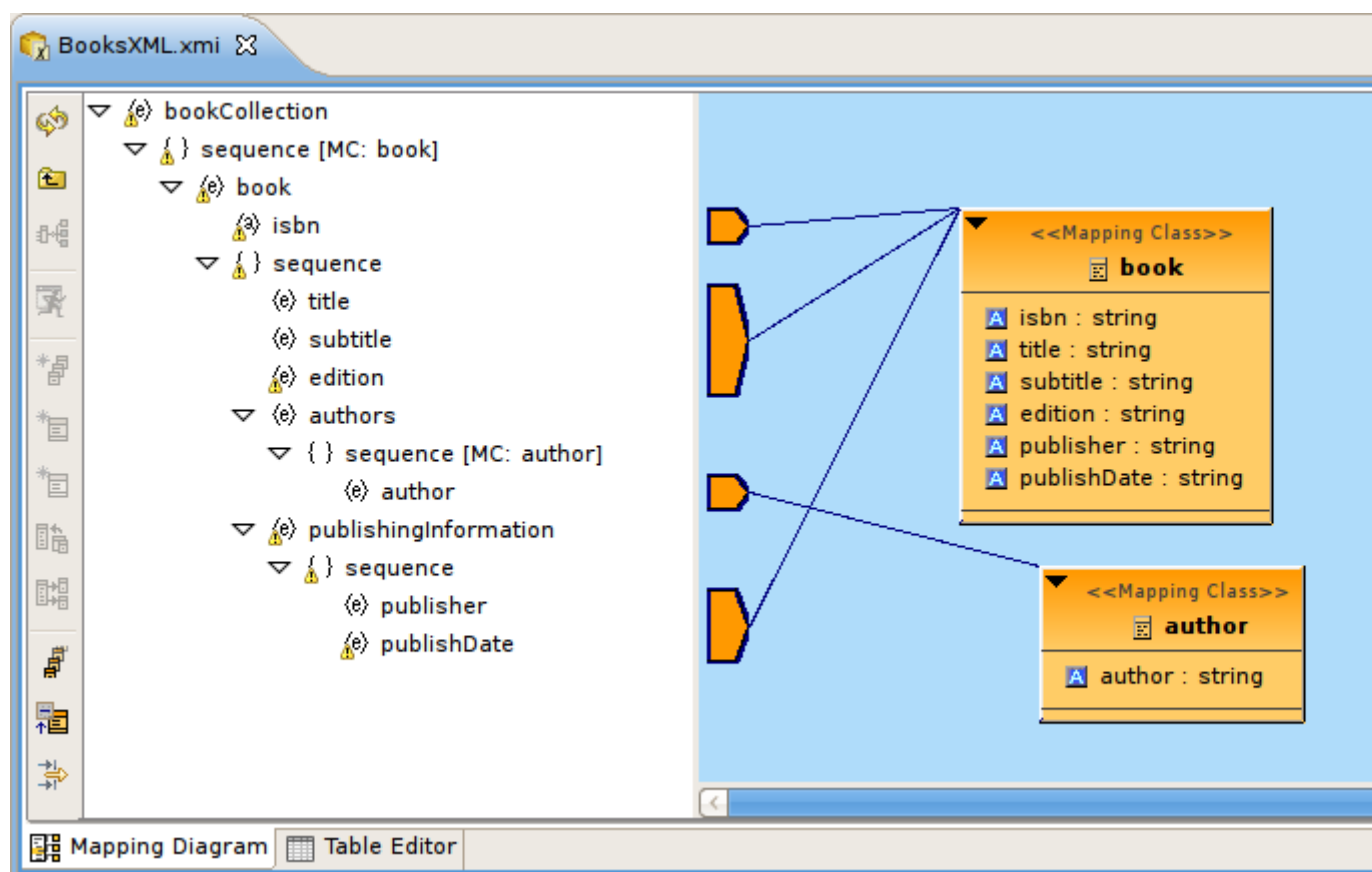


Figure 5.8. Mapping Diagram Example

- Mapping Diagram toolbar actions include:



Refresh Diagram - Re-draws diagram.



Show Parent Diagram - Navigates to diagram for parent object (if available).



Show Mapping Transformation Diagram - Show detailed mapping transformation diagram for selected mapping class.



Preview Data - Executes a simple preview query (SELECT * FROM).



Generate Mapping Classes - Generate mapping classes for the selected XML document root element.



New Mapping Class - Insert new mapping class referenced to the selected XML document element or attribute..



New Staging Table - Insert new staging table referenced to the selected XML document element or attribute.



Merge Mapping Classes - Merge selected mapping classes.



Split Mapping Class - Split selected mapping class.



Display All Mapping Classes



Show Mapping Class Columns



Filter Displayed Mapping Classes with Selection

Context menus for Mapping Diagrams provide Edit capability to the mapping class in addition to mapping class manipulation actions (i.e. Merge Mapping Classes, Split Mapping Class, etc..)

5.1.1.5. Mapping Transformation Diagram

The **Mapping Transformation Diagram** is identical to a Transformation Diagram except for displaying an Input Set and possibly Staging Tables as sources for the Mapping Class's transformation.

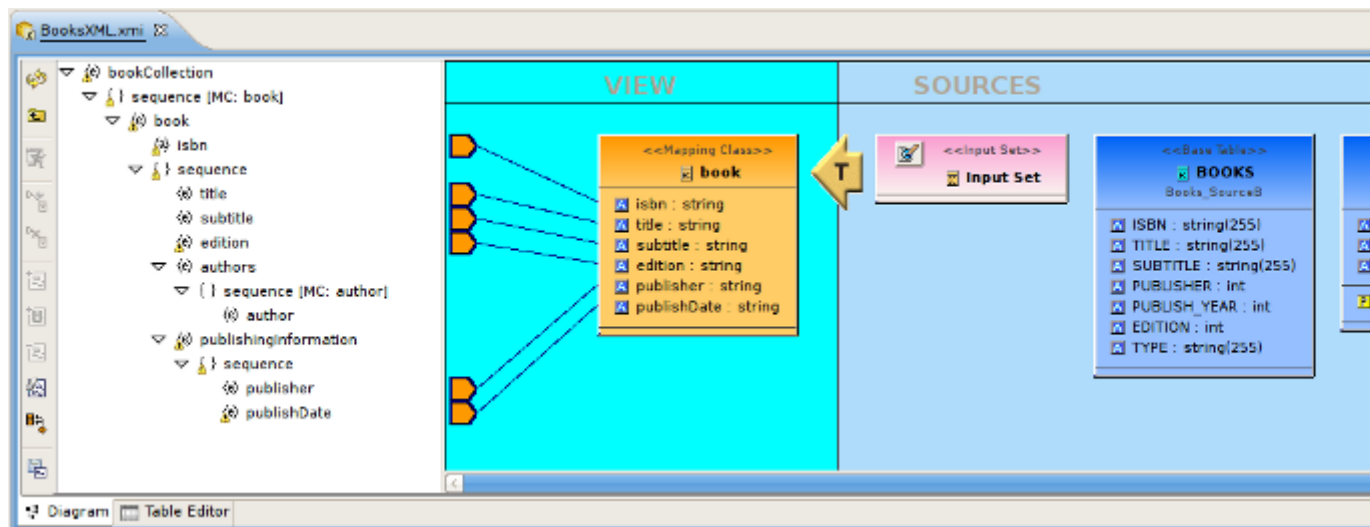














Figure 5.9. Mapping Transformation Diagram Example

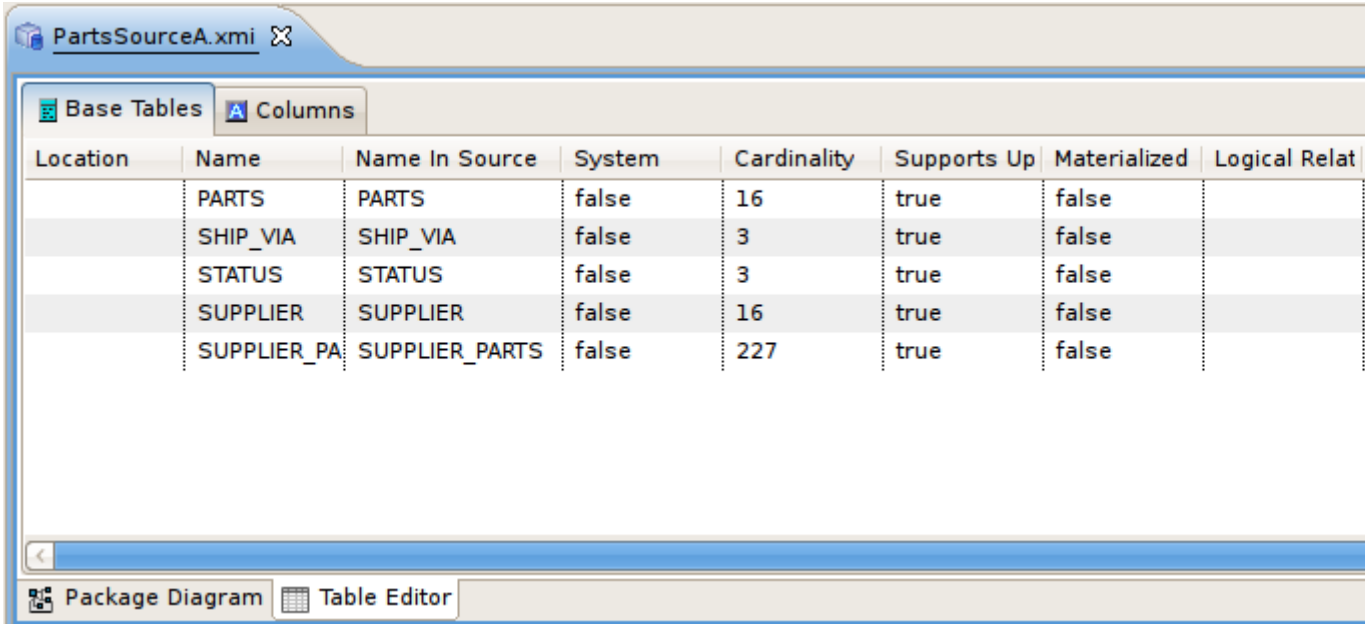
- Mapping Transformation Diagram toolbar actions include:
 -  **Refresh Diagram** - Re-draws diagram.
 -  **Show Parent Diagram** - Navigates to diagram for parent object (if available).
 -  **Preview Data** - Executes a simple preview query (SELECT * FROM).
 -  **New Mapping Link** - Create a mapping link between selected mapping extent (i.e. XML element or attribute) and mapping class column.
 -  **Remove Mapping Link** - Delete mapping link between selected mapping extent (i.e. XML element or attribute) and mapping class column.
 -  **Add Transformation Sources** - Add selected sources to transformation.
 -  **Add Union Transformation Sources** - Add selected sources as union sources.

-  **Remove Transformation Sources** - Removed sources selected in diagram from transformation.
-  **Clear Transformation** - Remove all sources from transformation.
-  **Open Transformation Reconciler dialog**
-  **Save Diagram as Image** - Save the diagram image to file in JPG or BMP format.
-  **Modify Diagram Printing Preferences** - Modify page layout information for printing diagrams. Includes margins, orientation, etc...

Context menus for **Mapping Transformation Diagrams** identical capabilities to the Transformation Diagram with the addition of managing and editing Input Sets.

5.1.2. Table Editor

The **Table Editor** provides a table-based object type structured view of the contents of a model. The figure below shows a relational model viewed in the **Table Editor**. Common object types are displayed in individual folders/tables. All base tables, for instance, are shown in one table independent of their parentage.



Location	Name	Name In Source	System	Cardinality	Supports Up	Materialized	Logical Relat
	PARTS	PARTS	false	16	true	false	
	SHIP_VIA	SHIP_VIA	false	3	true	false	
	STATUS	STATUS	false	3	true	false	
	SUPPLIER	SUPPLIER	false	16	true	false	
	SUPPLIER_PA	SUPPLIER_PARTS	false	227	true	false	

Figure 5.10. Table Editor Example

You can customize Table Editor properties via Table Editor Preferences.

These are the primary features of the Table Editor:


- Edit existing properties.
- Add, remove or edit objects, via the main Edit menu and context menu (Cut, Copy, Paste, Clone, Delete, Rename, Insert Rows).
- Paste information from your clipboard into the table.
- Print your tables.

When a **Table Editor** is in focus, the **Insert Table Rows** action



is added to the application's main toolbar.

A few Table Editor actions are contributed to the right-click menu for selected table rows. These actions, described and shown below include:

-  **Paste** - Paste common spreadsheet data (like Microsoft Excel) to set object properties.
- **Table Editor Preferences** - Change table editor preferences, including customizing visible properties.

- **Insert Rows** - Create multiple new sibling objects.



Table - Refreshes the contents of the current Table Editor to insure it is in sync with the model.

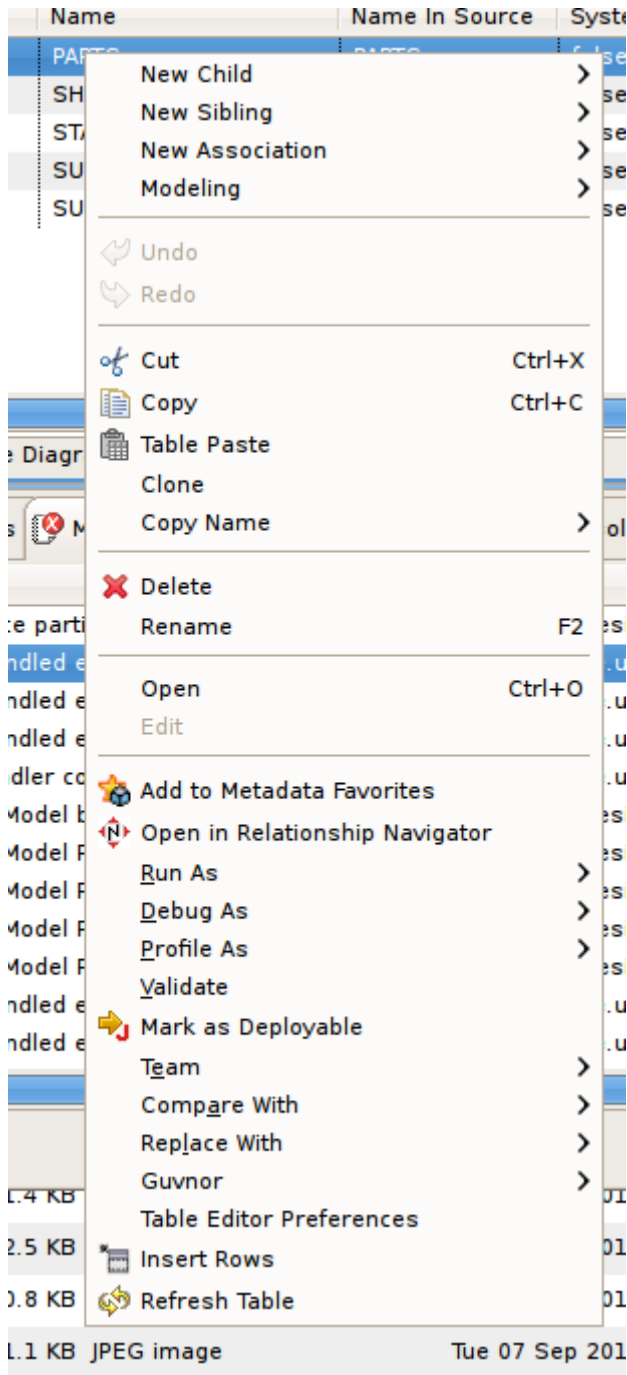
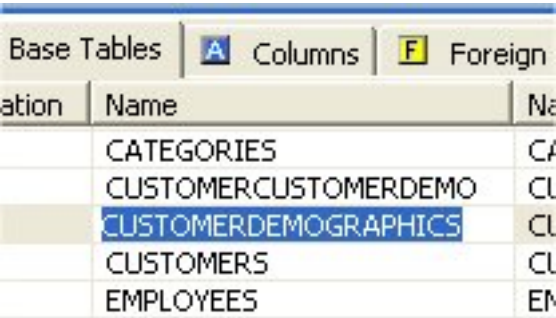


Figure 5.11. Table Editor Example

5.1.2.1. Editing Properties

You can edit properties for an object by double-clicking a table cell.

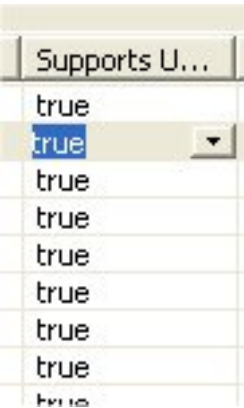
For String properties, the table cell will become an in-place text editor field.



Base Tables	A Columns	F Foreign
ation	Name	Na
	CATEGORIES	CA
	CUSTOMERCUSTOMERDEMO	CU
	CUSTOMERDEMOGRAPHICS	CU
	CUSTOMERS	CU
	EMPLOYEES	EM

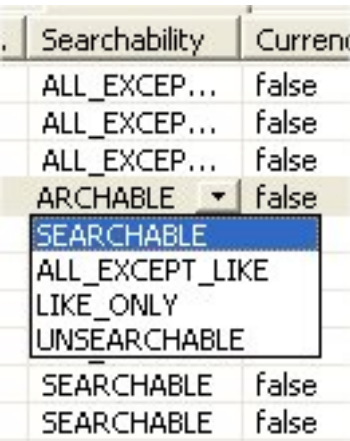
Figure 5.12. Editing String Property

If a property is of a boolean (true or false) type or has multiple, selectable values, a combo box will be displayed to change the value.



Supports U...
true
true
true
true
true
true
true
true
true
true

Figure 5.13. Editing Boolean Value



Searchability	Current
ALL_EXCEP...	false
ALL_EXCEP...	false
ALL_EXCEP...	false
ARCHABLE	false
SEARCHABLE	false
ALL_EXCEPT_LIKE	
LIKE_ONLY	
UNSEARCHABLE	
SEARCHABLE	false
SEARCHABLE	false

Figure 5.14. Editing Multi-Value Property

For multi-valued properties where the available values are dynamic (i.e. can change based on available models or data), a picker-button ("....") will be displayed.

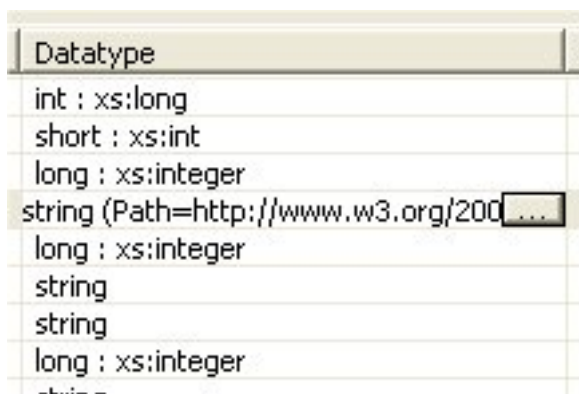


Figure 5.15. Editing Multi-Value With Picker

An example of of this type is the relational column datatype property. Editing via the table cell and clicking the "..." button for datatype will display the following dialog.



Figure 5.16. Editing Datatype Values

5.1.2.2. Inserting Table Rows

The Insert Rows action provides an additional way to create objects in a model. Insert Rows action performs the same function as Insert Sibling action, but allows you to create multiple children at the same time. All new rows will correspond to an object of the same type as the selected object and be located under the same parent as the selected object.

To Insert Rows in a table:

Step 1: Select a table row to insert rows after.

Step 2: Right-click select "Insert Rows" action or select the Insert Rows action on the main toolbar. The following dialog will be displayed.

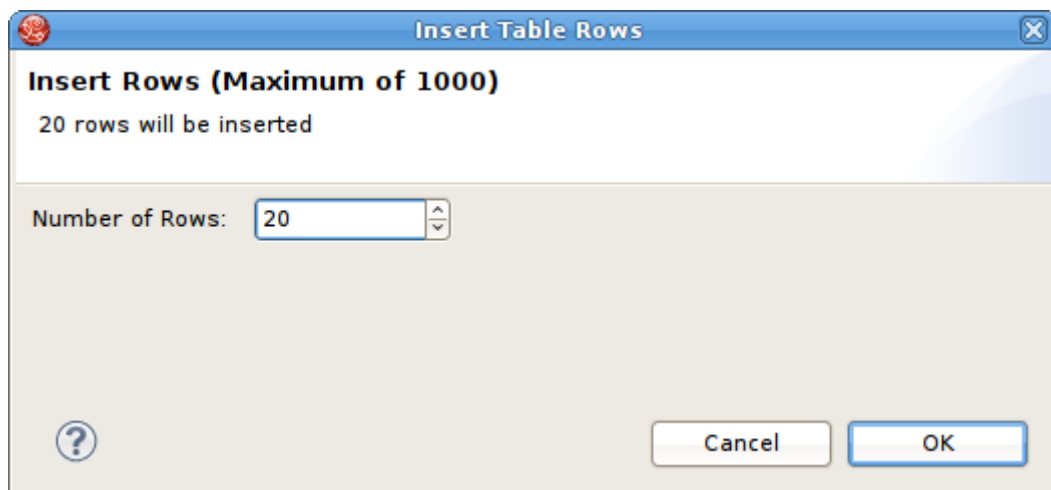


Figure 5.17. Editing String Property

Step 3: Edit the Number of Rows value in the dialog, or use the up/down buttons to change the value.

Step 4: Select OK in dialog.

The desired number of rows (new model objects) will be added after the original selected table row.

5.1.3. Simple Datatypes Editor

The Simple Datatype Editor provides a form-based properties view of XML Schema data.

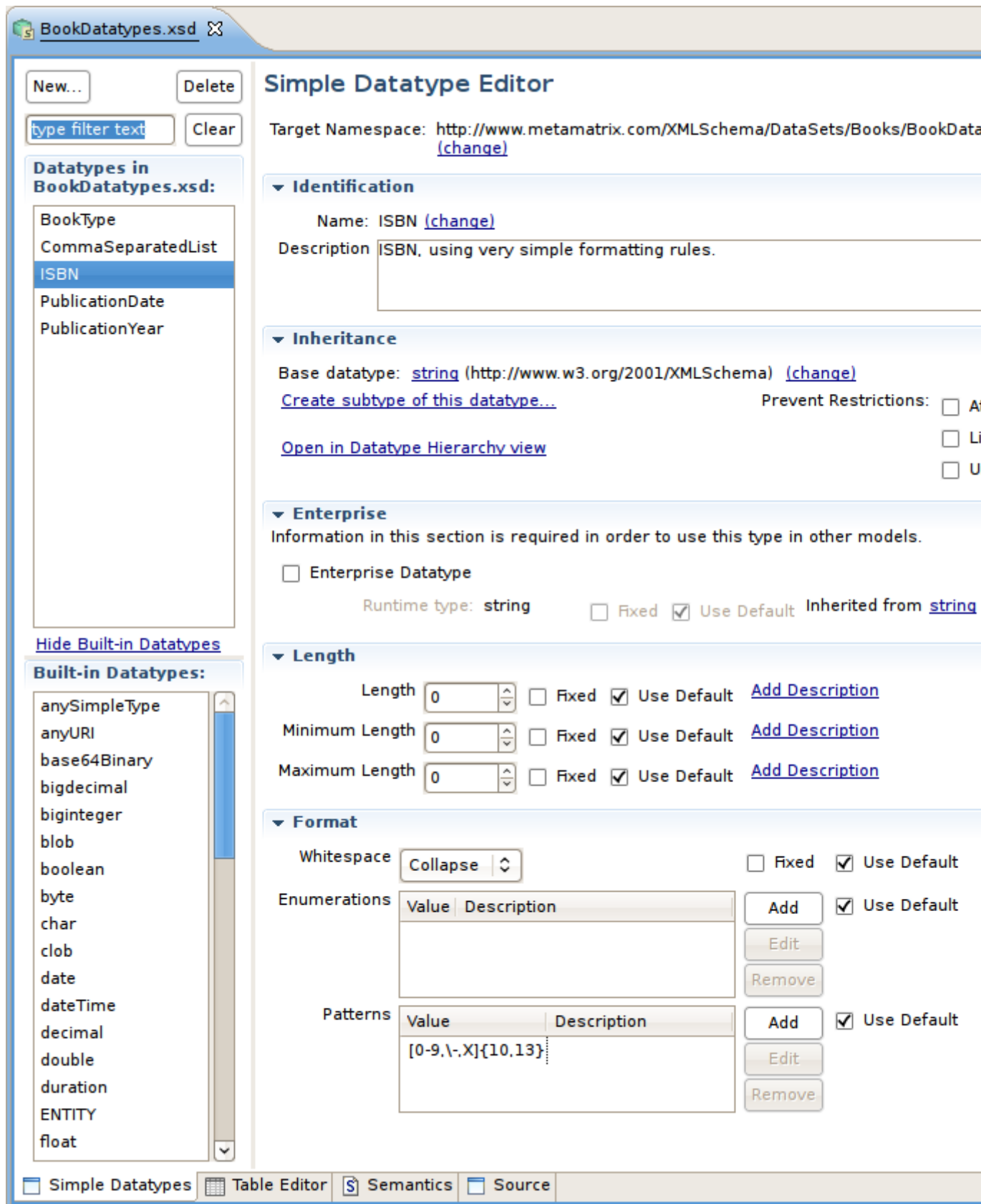


Figure 5.18. Editing String Property

5.1.4. Semantic Editor

The Semantic Editor is a tree based editor for XML Schema elements and attributes.

5.1.5. Source Editor

The Source Editor is a simple text editor which is aware of XML Schema formatting rules.

5.1.6. Model Object Editors

The Model Object Editors represent specialized sub-editors which are available for specific model object types.

- For details, select a specific editor listed below:

- [Section 8.2.1, “Transformation Editor”](#)

[Section 8.2.2, “Input Set Editor \(XML\)”](#)

[Section 8.2.3, “Choice Editor \(XML\)”](#)

[Section 8.2.4, “Recursion Editor \(XML\)”](#)

[Section 8.2.5, “Operation Editor”](#)

5.2. VDB Editor

A **VDB**, or **virtual database** is a container for components used to integrate data from multiple data sources, so that they can be accessed in a federated manner through a single, uniform API. A **VDB** contains models, which define the structural characteristics of data sources, views, and Web services. The **VDB Editor**, provides the means to manage the contents of the **VDB** as well as its deployable (validation) state.

The VDB Editor, shown below, contains a upper and lower panels. The upper panel contains the Models tab and an Other Files tab. The lower panel contains tabs for managing Data Roles, the VDB Description and Translator Overrides.

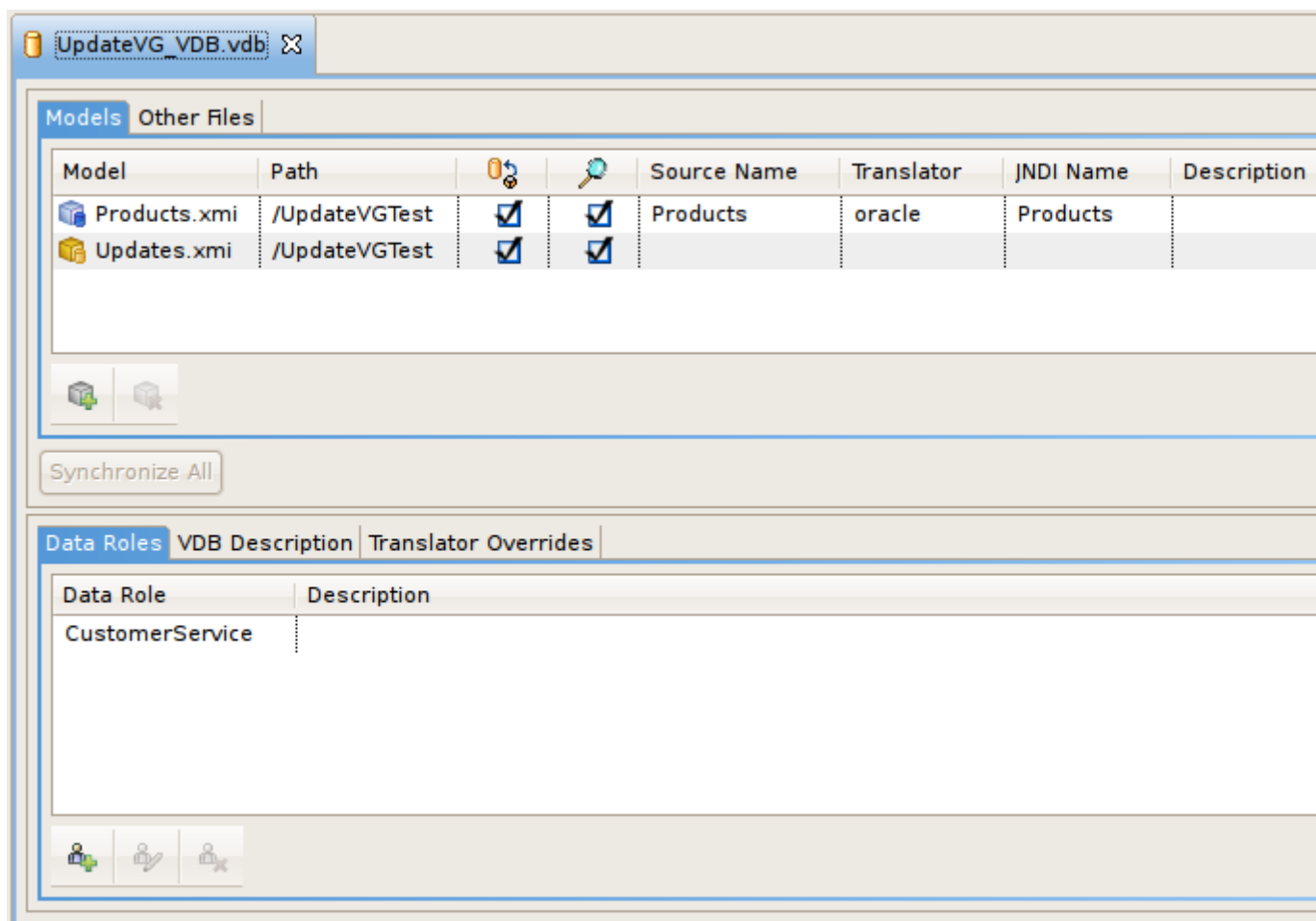


Figure 5.19. VDB Editor

You can manage your VDB contents by using the *Add* or *Remove* models via the buttons at the right.

Set individual model visibility via the *Visibility* checkbox for each model. This provides low level data access security by removing specific models and their metadata contents from schema exposed in GUI tools.

In order for a VDB to be fully queryable the "Source Name", "Translator" and "JNDI Names" must have valid values and represent deployed artifacts on your Teiid server.

If you have Designer runtime plugins installed, and have a Teiid server running, you can select a source model in the VDB Editor and right-click select "Change Translator" or "Change JNDI Data Source" which will allow you to select any applicable artifacts on your server.

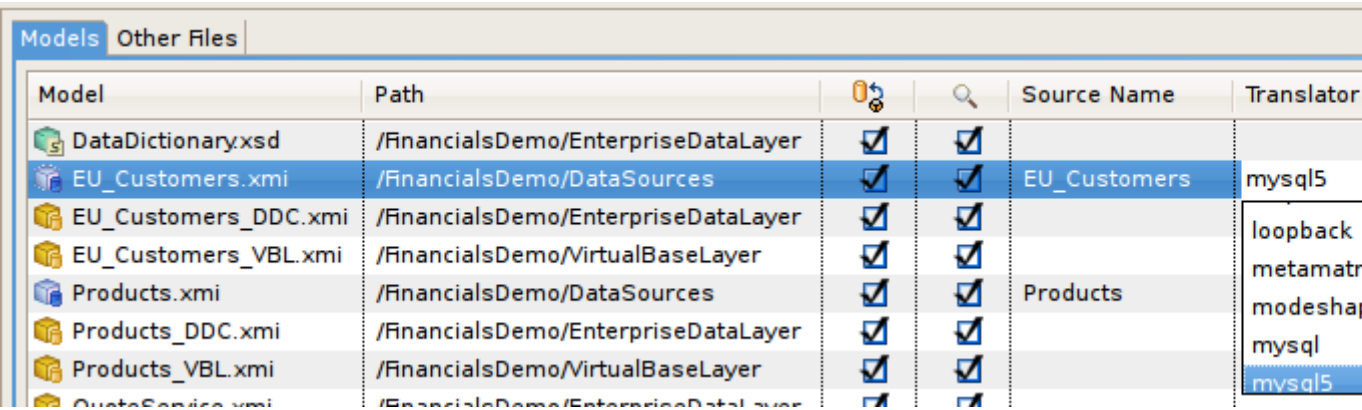


Figure 5.20. Change Translator or Data Source Actions

If you have a default Teiid server instance defined and connected the translator and JNDI table cells will contain drop-down lists of available translator and JNDI names available on that server.

5.2.1. Editing Data Roles

Teiid Designer provides a means to create, edit and manage data roles specific to a VDB. Once deployed within a Teiid server with the security option turned on (by default) any query run against this VDB via a Teiid JDBC connection will adhere to the data access permissions defined by the VDB's data roles.

The *VDB Editor* contains a *VDB Data Roles* section consisting of a List of current data roles and *New...* , *Edit...* and *Remove* action buttons.

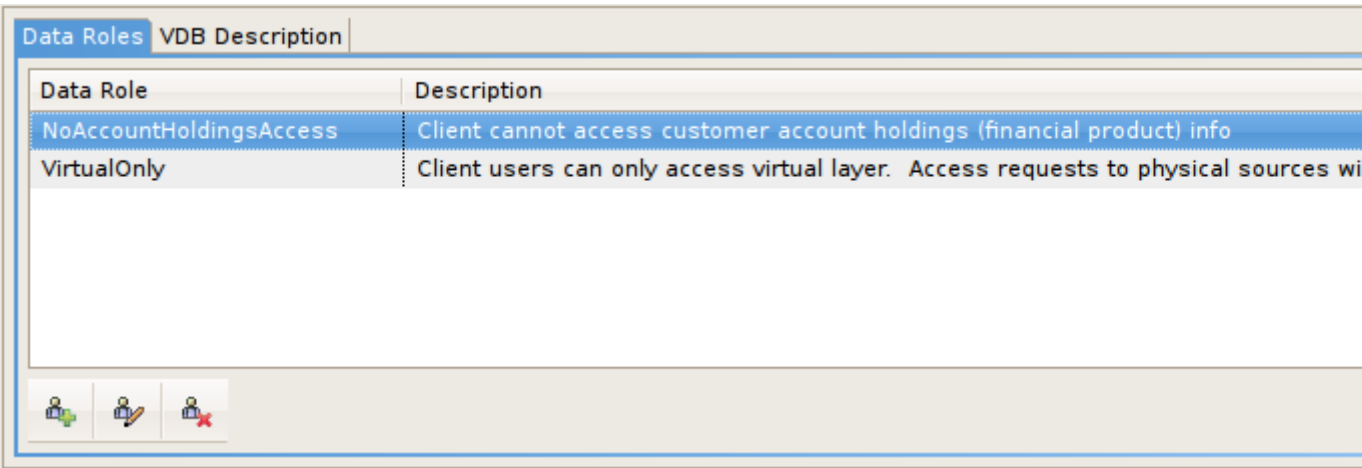


Figure 5.21. VDB Data Roles Panel

Clicking *New...* or *Edit...* will launch the *New VDB Data Role* editor dialog. Specify a unique data role name, add a optional description and modify the individual model element CRUD values by check or unchecking entries in the models section.

New VDB Data Role

Select Finish to save data role

Name:

Description:

☒ Apply this role to All Users

Mapped Role Names

Relational Models

Model	Create	Read	Update	Delete
PartsSourceA.xmi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PARTS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SHIP_VIA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
STATUS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SUPPLIER	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SUPPLIER_PARTS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PartsVirtual.xmi	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PartsSourceB.xmi	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

System Tables Access

☒ Allow this role to access SYSADMIN model

Figure 5.22. VDB Data Roles Tab

5.2.2. Editing Translator Overrides

Teiid Designer provides a means to create, edit and manage translator override properties specific to a VDB via the Translator Overrides tab. A translator override is a set of non-default properties

targeted for a specific source model's data source. So each translator override requires a target translator name like "oracle", db2, mysql, etc. and a set of non-default key-value property sets.

The *VDB Editor* contains a *Translator Overrides* section consisting of a List of current translator overrides on the left, a properties editor panel on the right and *Add (+)* and *Remove (-)* action buttons on the lower part of the panel.

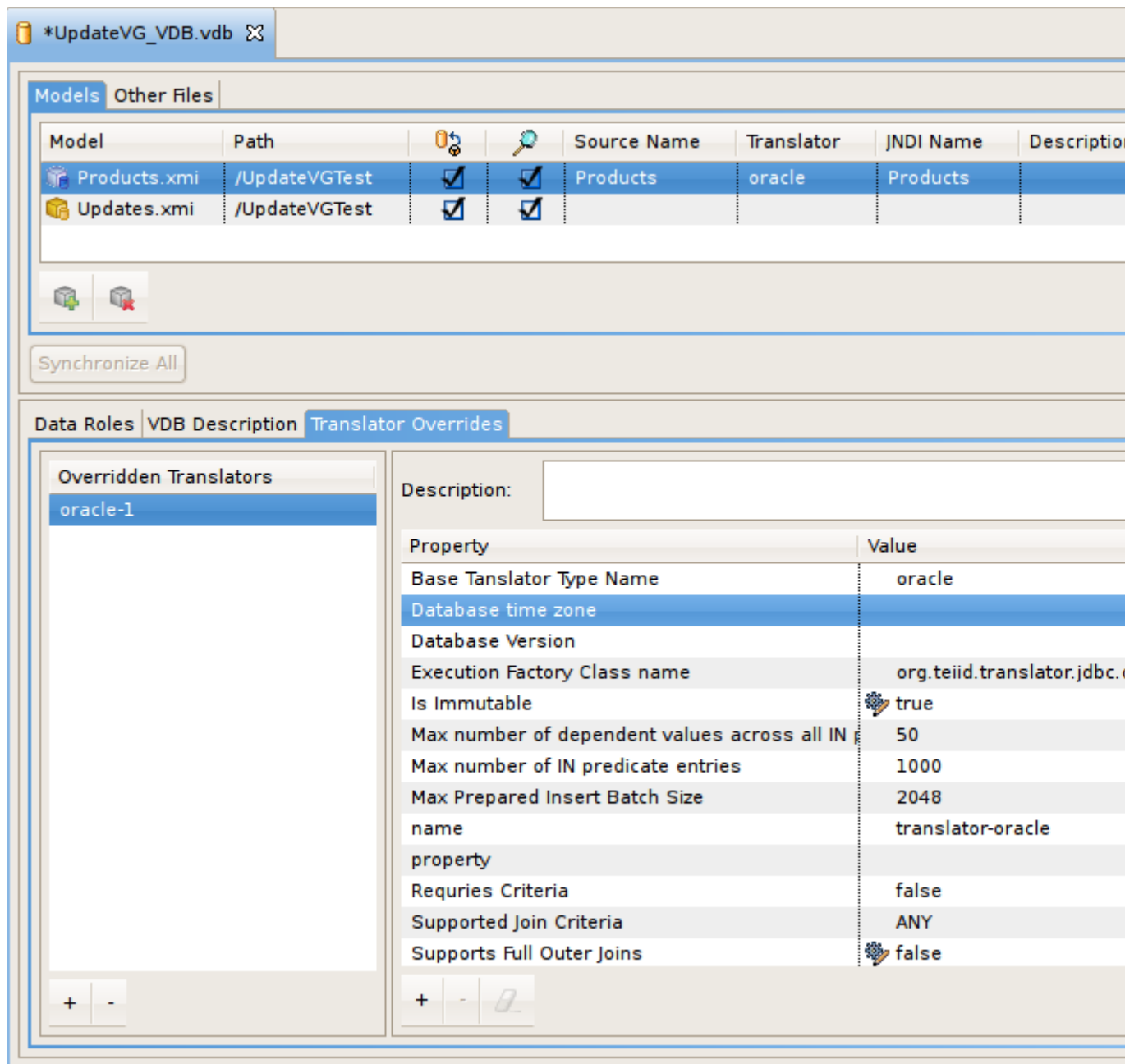


Figure 5.23. VDB Translator Overrides Tab

To override a specific translator type, select the add translator action (+). If a default Teiid server instance is connected and available the Add Translator Override dialog (below) is presented, the user selects an existing translator type and clicks OK. Note that the override is only applicable to

sources within the VDB, so be sure and select a translator type that corresponds to one of the VDB's source models. The properties panel on the right side of the panel will contain editable cells for each property type based on the data-type of the property. (i.e. boolean, integer, string, etc.).

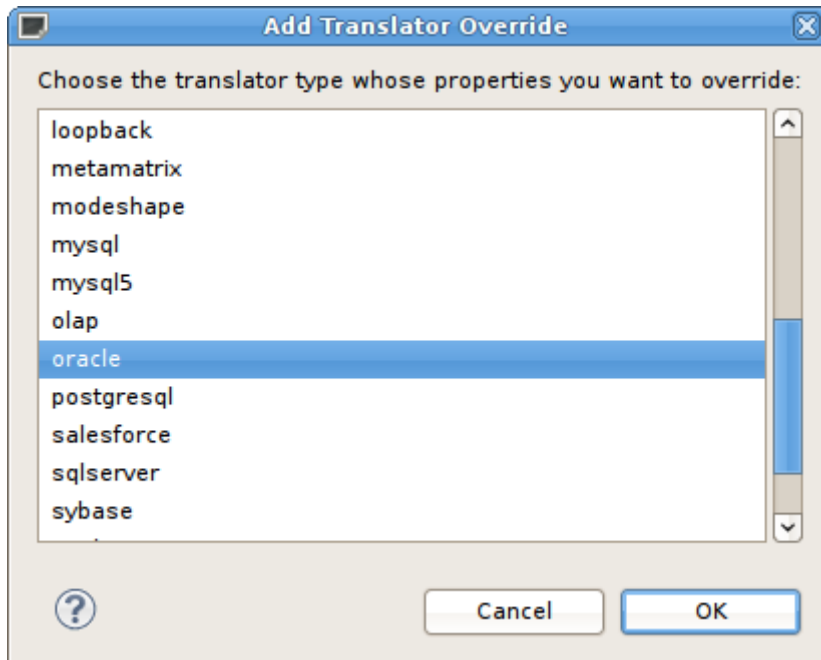


Figure 5.24. Add Translator Override Dialog

If no default Teiid server instance is available, the "Add New Translator Override" dialog is presented. Enter a unique name for the translator override (i.e. "oracle_override"), a valid translator type name (i.e. "oracle") and click OK. The properties panel on the right side of the panel will allow adding, editing and removing key-value string-based property sets. When editing these properties all values will be treated as type string.

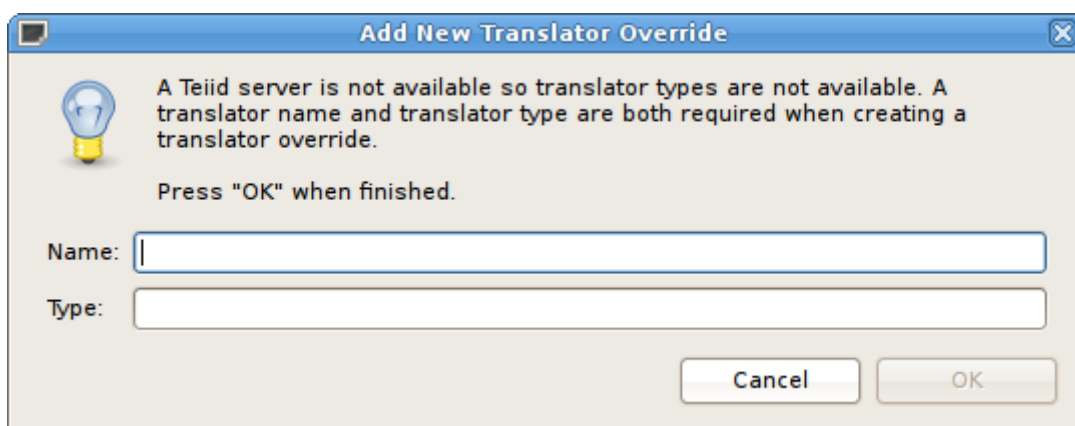


Figure 5.25. Add New Translator Override Dialog

Importers

The **Import Wizard** provides a means to create a model based on the structure of a data source, to convert existing metadata (i.e. **WSDL** or **XML Schema**) into a source model or to load existing metadata files into the current VDB.

To launch the **Import Wizard**, choose the **File > Import** action or select a project, folder or model in the tree and right-click choose "Import..."

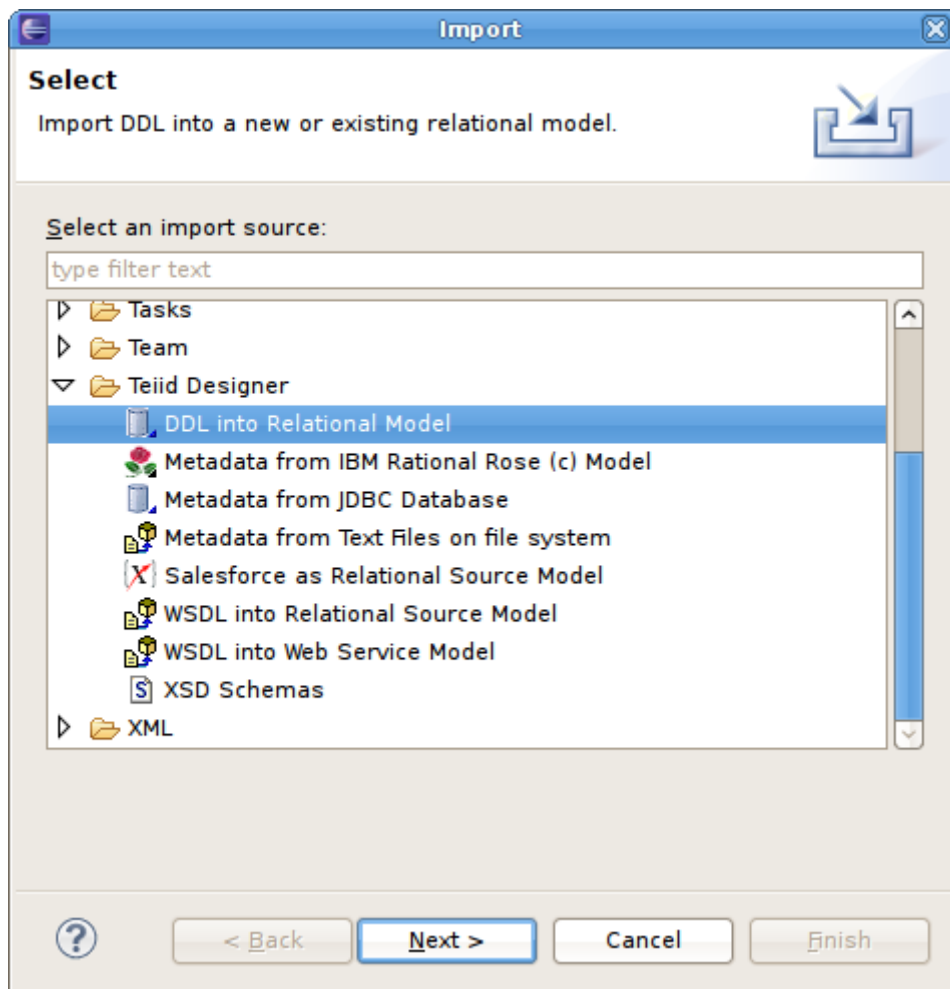


Figure 6.1. Import Wizard

6.1. Import DDL into Relational Model

Source relational models can be created by importing DDL.

- You can create relational source models from your DDL using the steps below.

- **Step 1** - In **Model Explorer** choose the **File > Import** action



in the toolbar or select a project, folder or model in the tree and choose **Import...**

- **Step 2** - Select the import option **Teiid Designer > DDL into Relational Model** and click **Next>**

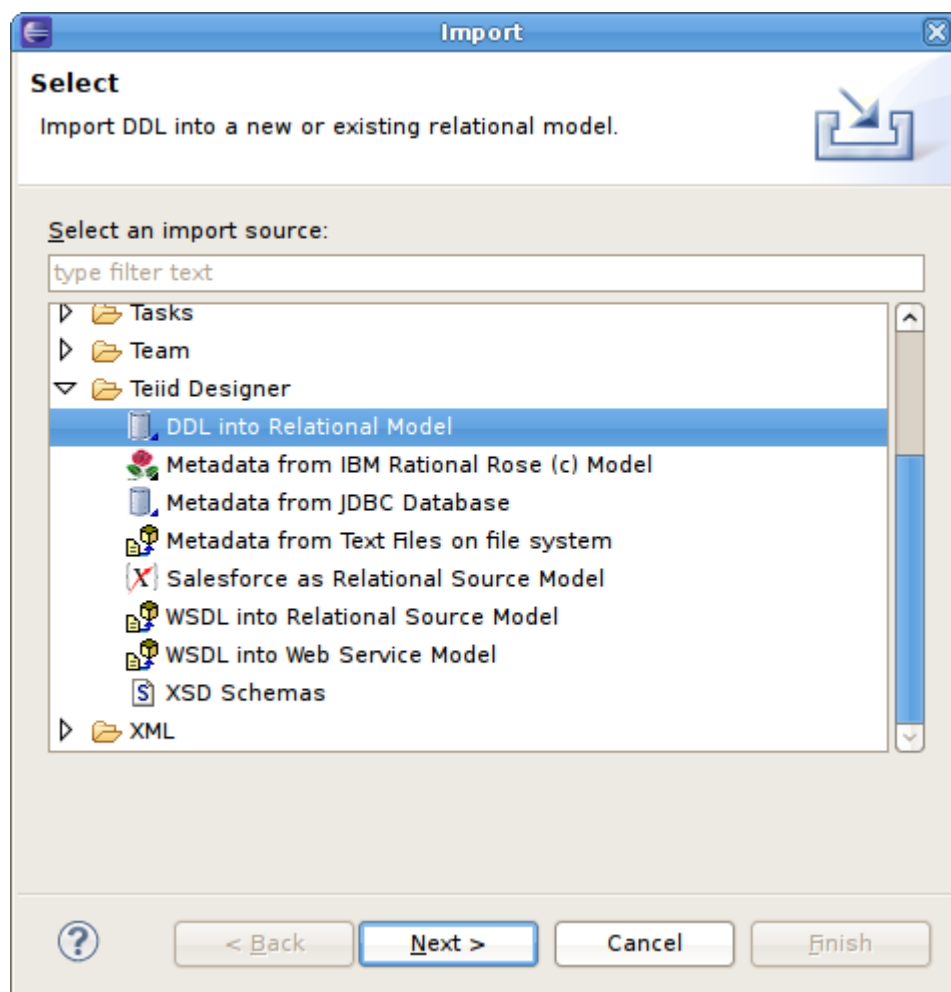


Figure 6.2. Import DDL into Relational Model

- **Step 3** - Select existing DDL from either **Choose from file system...** or **Choose from workspace...** set the Model folder location, enter or select valid model name, set Model type (Source Model or View Model), set desired options and click **NEXT>** (or **FINISH>** if enabled)

Provide DDL source

Update relational source model "ProductsSQLServer" using DDL file "/home/blafond/Runtime-Workspaces/test_7_1_0_E/DDLTest/ProductsSQLServer.ddl".

DDL file: Choose from file system...
 Choose from workspace...

Model folder: Choose...

Model name: Choose...

Model type:

☐ Create model entities for DDL defined by unsupported DML (e.g., Views)

▼ DDL file contents

```

CREATE TABLE ProductData
(
  ProductID      VARCHAR(10) NOT NULL,
  ProductName    VARCHAR(60),
  ProductType    VARCHAR(15),
  ISSUER         VARCHAR(10),
  EXCHANGE       VARCHAR(10),
  DJICOMPONENT   NUMERIC(1) NOT NULL,
  SP500COMPONENT NUMERIC(1) NOT NULL,
  NAS100COMPONENT NUMERIC(1) NOT NULL,
  AMEXINTCOMPONENT NUMERIC(1) NOT NULL,
  PrimaryBusiness VARCHAR(30)
)
GO

-- (generated from ProductSymbols)
  
```

? < Back Next > Cancel Finish

Figure 6.3. DDL Import Options

- **Step 4** - If *NEXT>* is pressed, a difference report is presented for viewing or de-selecting individual relational entities. Press *FINISH>* to complete.

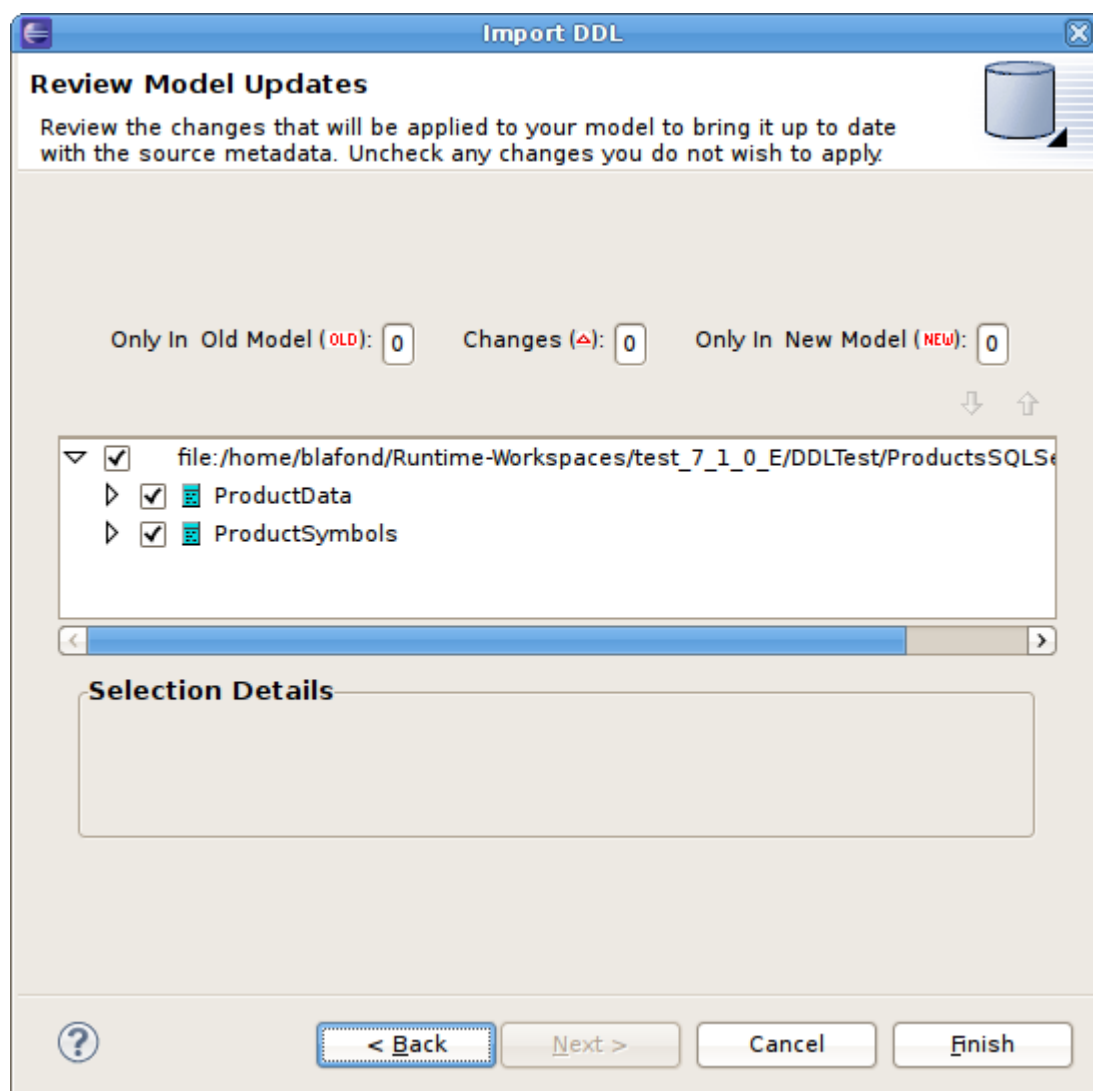


Figure 6.4. Review DDL Updates Dialog

6.2. Import From JDBC

- You can create relational source models from your JDBC source schema data using the steps below.



Note

Depending the detail provided in the database connection url information and schema, Steps 5 through 7 may not be required.

- **Step 1** - In **Model Explorer** choose the **File > Import** action



in the toolbar or select a project, folder or model in the tree and choose **Import...**

- **Step 2** - Select the import option **Metadata Modeling > Metadata from JDBC Database** and click **Next>**

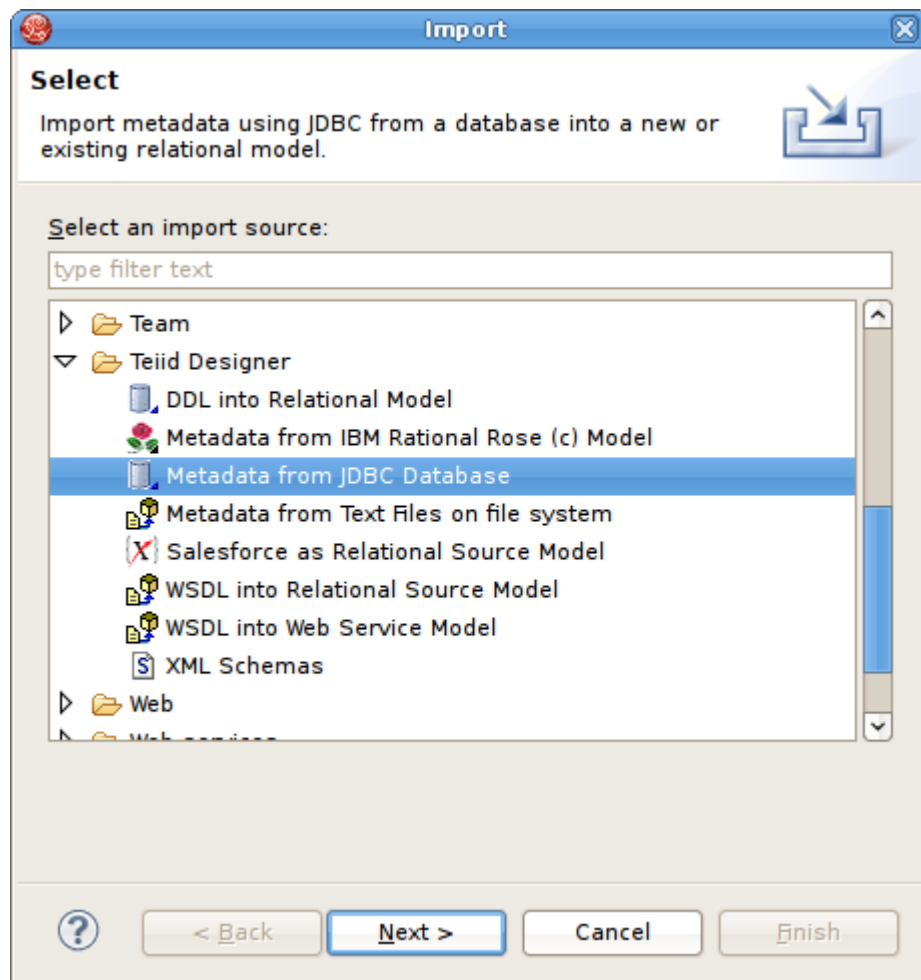


Figure 6.5. Import Metadata from JDBC Database

- **Step 3** - Select existing or previous connection profile from the drop-down selector or press **New...** button to launch the **New Connection Profile** dialog (See Eclipse Data Tools documentation) or **Edit...** to modify/change an existing connection profile prior to selection.

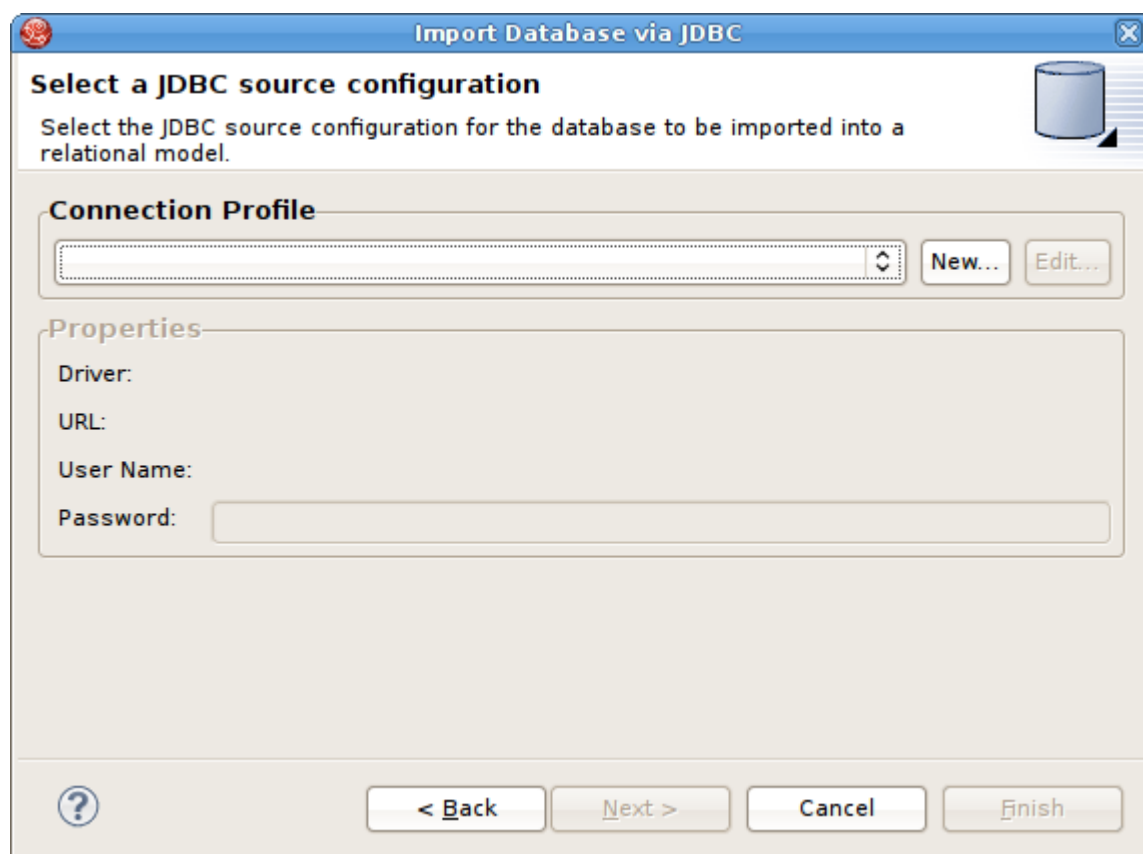


Figure 6.6. Select JDBC Source Configuration Dialog

- **Step 4** - After selecting a *Connection Profile*, input password (if not provided). Press **Next>** (or **Finish>** if enabled)

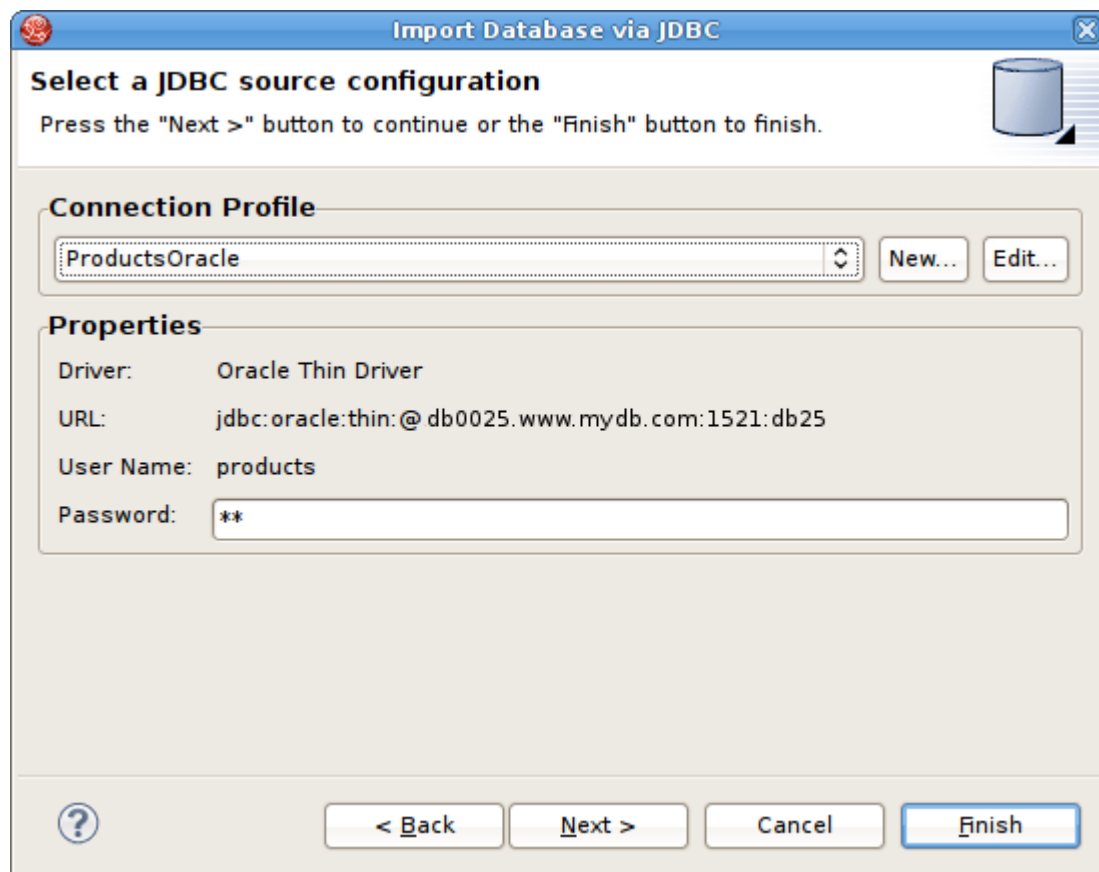


Figure 6.7. Select JDBC Source Configuration Dialog

- **Step 5** - On the **Select Database Metadata** page, select the types of objects in the database to import. Press **Next>** (or **Finish>** if enabled).

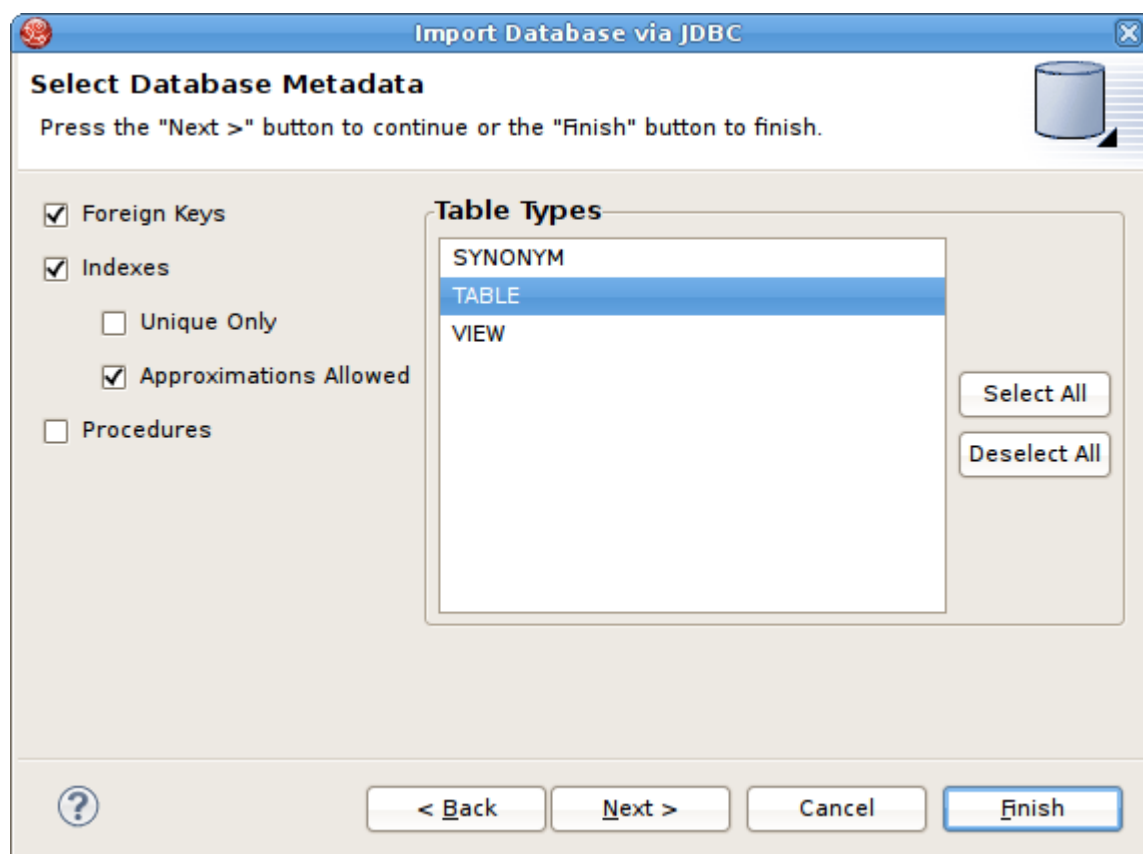


Figure 6.8. Select Database Metadata Dialog

- **Step 6** - On the **Select Database Objects** page, view the contents of the schema, or change selections. Select which database schema objects will be used to construct relational objects. Press **Next>** (or **Finish>** if enabled)

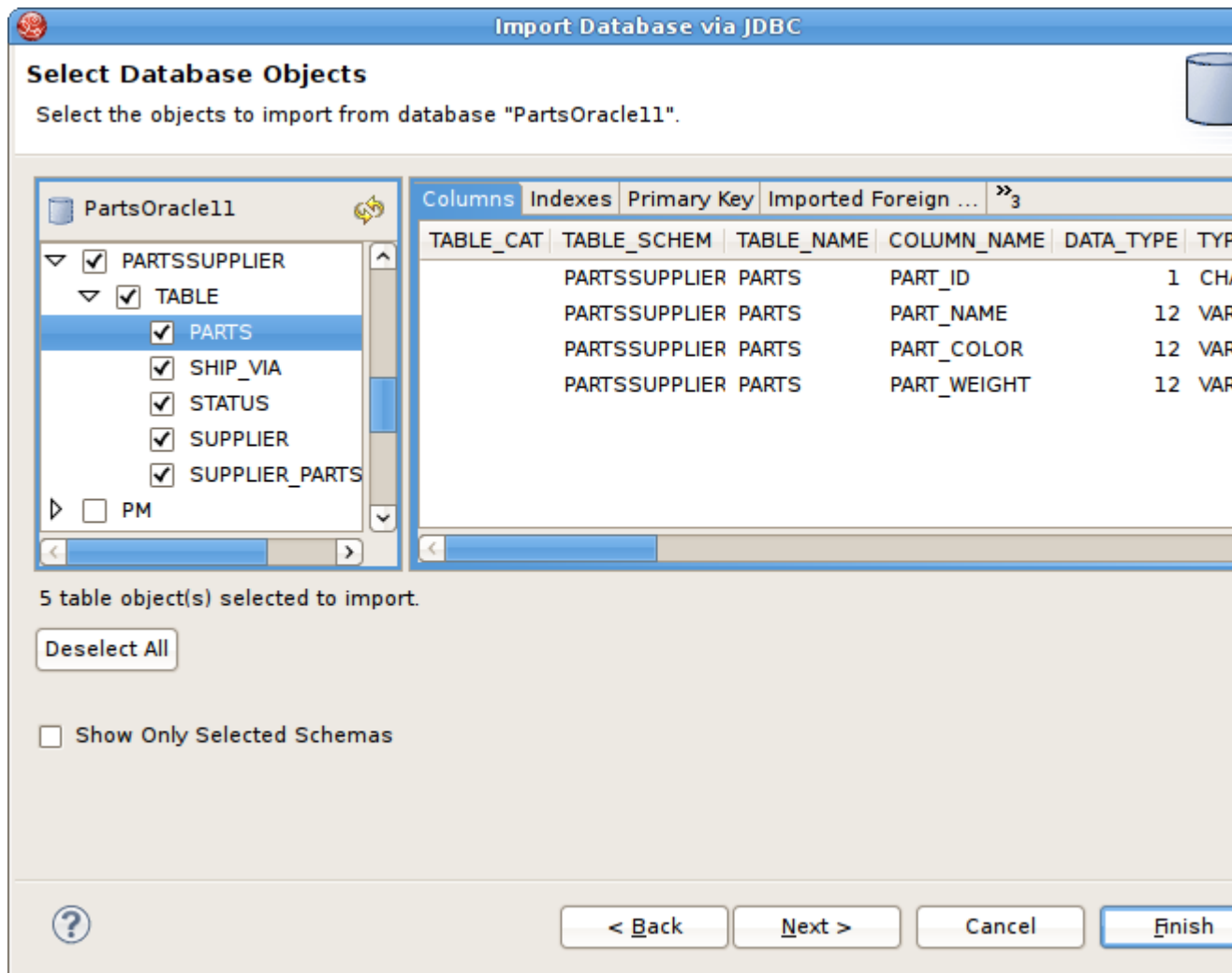


Figure 6.9. Select Database Options Dialog

- **Step 7** - On the **Specify Import Options** page, specify desired **Model Name** as well as any other options used to customize the constructed relational objects. Press **Finish>** to complete.

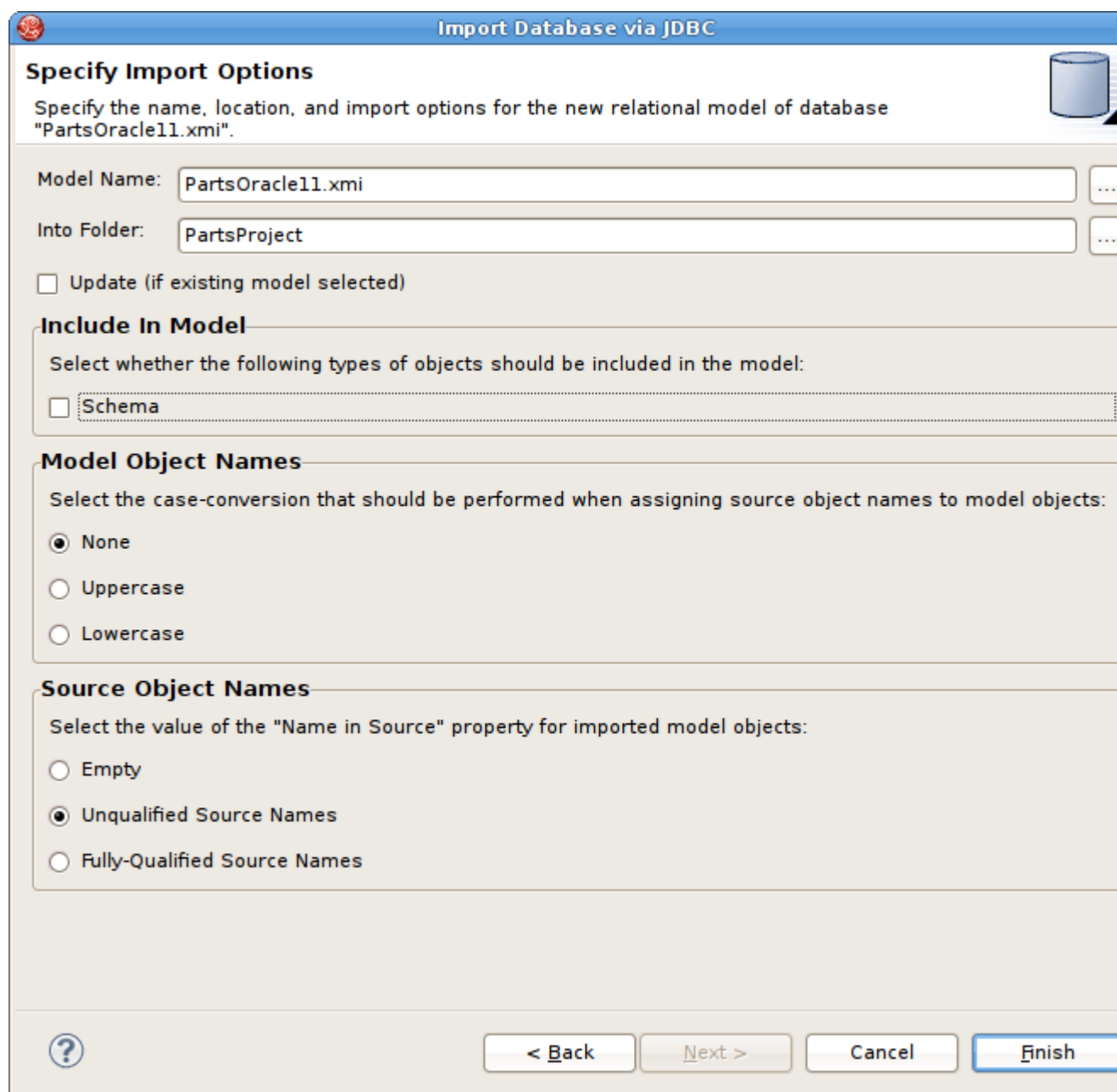


Figure 6.10. Specify Import Options Dialog

During the *Finish* processing, a monitor will be displayed providing feedback on the import progress.

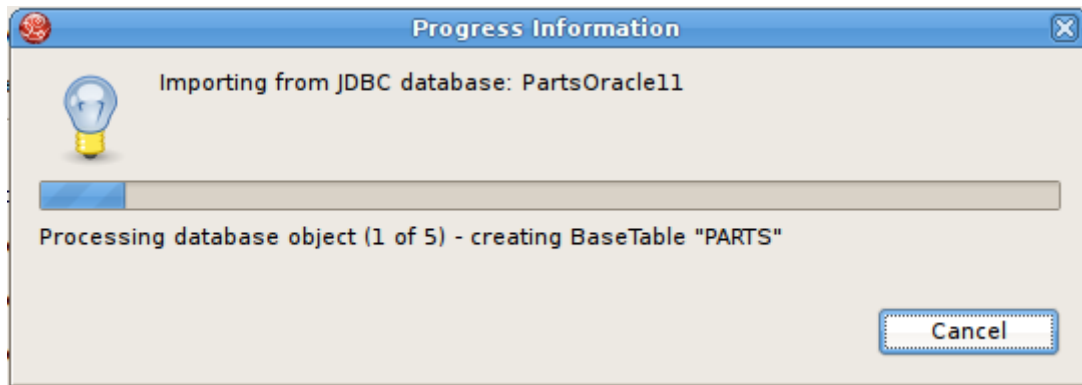

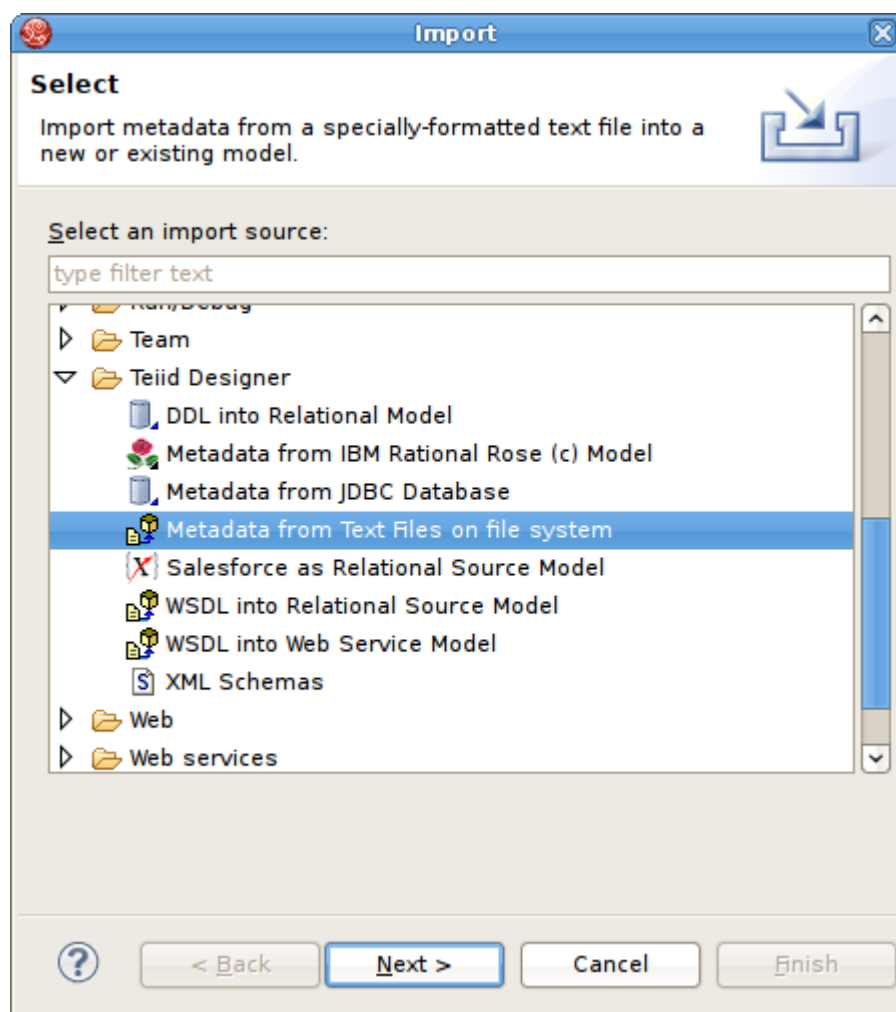


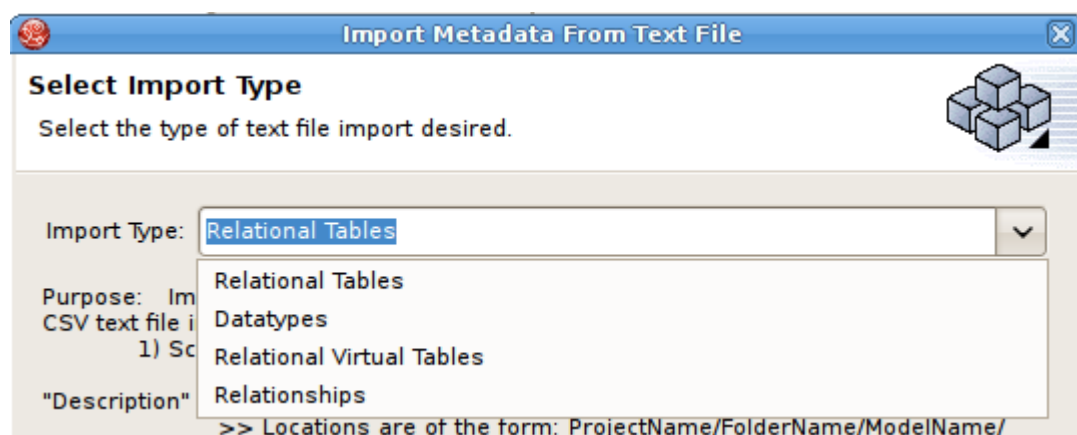
Figure 6.11. JDBC Import Progress Dialog

6.3. Import Metadata From Text File

- The Teiid Designer provides various import options for translating specific text file metadata into models. This is accomplished via the **Import > Metadata from Text Files on file system** option.
- **Step 1** - In **Designer** choose the **File > Import** action  in the toolbar or select a project, folder or model in the tree and choose **Import...**
- **Step 2** - Select the import option **Metadata Modeling > Metadata from Text Files on file system** and click **Next>**

**Figure 6.12. Import Metadata From Text Files**

- **Step 3** - Select an import type via the drop-down menu shown below.

**Figure 6.13. Import Wizard**

- These steps required for each type are defined below:
 - [Section 6.3.1, "Import Relational Tables Text Importer"](#)
 - [Section 6.3.2, "Import Relational Virtual Tables Text Importer"](#)
 - [Section 6.3.3, "Import Datatypes Text Importer"](#)

6.3.1. Import Relational Tables Text Importer

- To create relational tables from imported text file metadata:
 - Perform **Steps 1 through 3** (above) and select the **Relational Tables** import type, then click **Next >**

Import Metadata From Text File

Select Import Type
Select the type of text file import desired.

Import Type:

Purpose: Imports relational schema, catalogs, tables, columns, and indexes from a CSV text file into a relational database.

- 1) Schema, Catalog, and Table data is expected to be of the form:
 - >> TYPE (i.e. SCHEMA, CATALOG, or TABLE), Name, "Description" (Optional), Location (Optional)
 - >> Locations are of the form: ProjectName/FolderName/ModelName/SchemaName.
 - >> If the project, folder, model or schema/catalog containers do not exist, they will be created.
- 2) The Column data is expected to be in the form:
 - >> COLUMN, ColumnName, JDBCType, Length, "Description"
- 3) Column data rows for each table must appear immediately following the table data row.
- 4) The Index data is expected to be of the form:
 - >> INDEX, IndexName, Type, Uniqueness, Tablespace, Column
- 5) Index data rows must appear immediately following the table column data rows.

A sample of typical input data is shown below:

Sample File Format:

```
CATALOG, Catalog_1, "Catalog_1 Description", Project_1/MyModel_1
SCHEMA, Schema_1, "Schema_1 Description", Project_1/MyModel_1/Catalog_1
```

Figure 6.14. Select Import Type - Relational Tables

- **Step 4** - In the next page, you'll need to provide a source text file containing the metadata formatted to the specifications on the previous page.

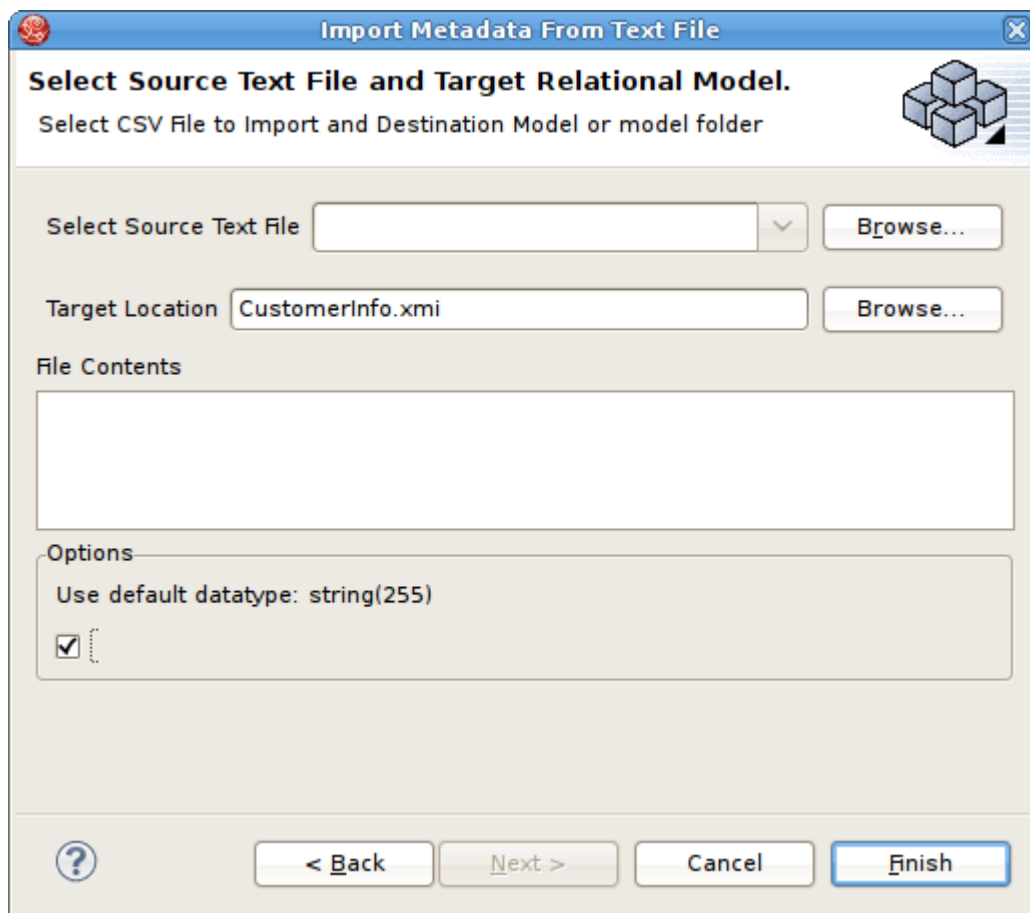
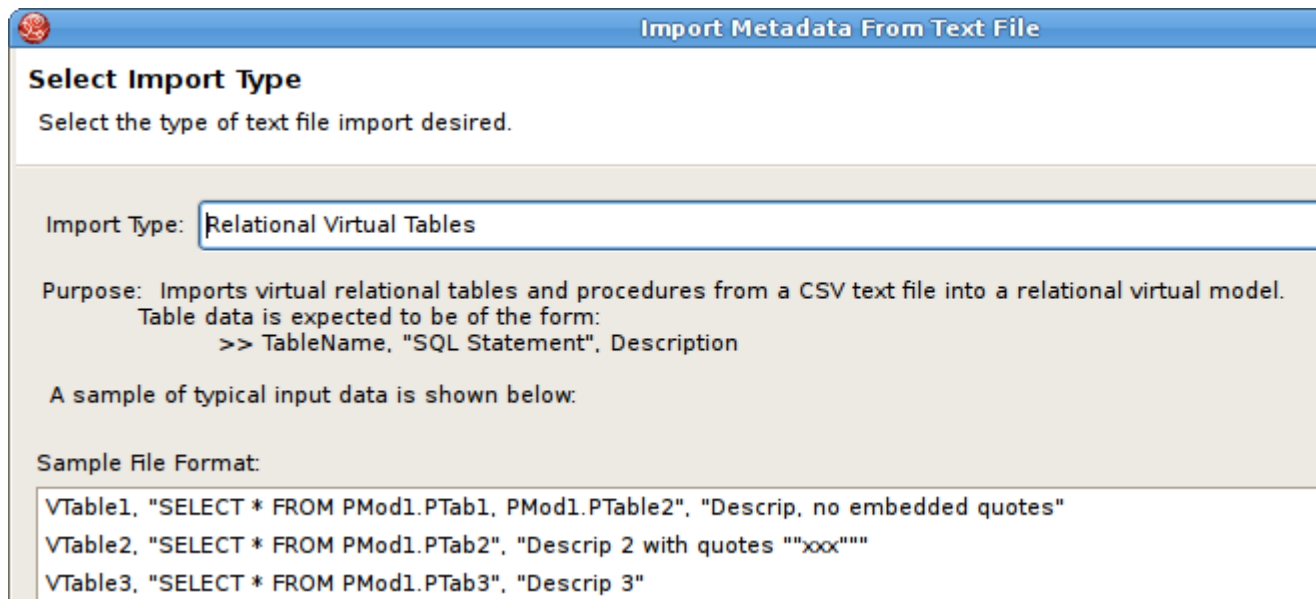


Figure 6.15. Select Source Text File and Target Relational Model

- **Step 5** - Select an existing relational model as the target location for your new relational components using the **Browse...** button to open the Relational Model Selector Dialog. Select a relational model from your workspace or specify a unique name to create a new model.
- **Step 6** - Select any additional options and choose **Finish**.

6.3.2. Import Relational Virtual Tables Text Importer

- To create relational virtual tables from imported text file metadata:
 - Perform **Steps 1 through 3** (above) and select the **Relational Virtual Tables** import type, then click **Next >**



The screenshot shows a dialog box titled "Import Metadata From Text File". Inside, the "Select Import Type" section is active, with the instruction "Select the type of text file import desired." Below this, the "Import Type:" dropdown menu is set to "Relational Virtual Tables". The "Purpose:" section explains that it imports virtual relational tables and procedures from a CSV text file into a relational virtual model, with a note that table data is expected to be of the form: >> TableName, "SQL Statement", Description. A sample of typical input data is shown below, under the heading "Sample File Format:". The sample data consists of three lines: VTable1, "SELECT * FROM PMod1.PTab1, PMod1.PTable2", "Descrip, no embedded quotes", VTable2, "SELECT * FROM PMod1.PTab2", "Descrip 2 with quotes ""xxx""", and VTable3, "SELECT * FROM PMod1.PTab3", "Descrip 3".

Import Metadata From Text File

Select Import Type
Select the type of text file import desired.

Import Type:

Purpose: Imports virtual relational tables and procedures from a CSV text file into a relational virtual model.
Table data is expected to be of the form:
>> TableName, "SQL Statement", Description

A sample of typical input data is shown below:

Sample File Format:

```
VTable1, "SELECT * FROM PMod1.PTab1, PMod1.PTable2", "Descrip, no embedded quotes"
VTable2, "SELECT * FROM PMod1.PTab2", "Descrip 2 with quotes ""xxx""
VTable3, "SELECT * FROM PMod1.PTab3", "Descrip 3"
```

Figure 6.16. Select Import Type - Relational Virtual Tables

- **Step 4** - In the next page, you'll need to provide a source text file containing the metadata formatted to the specifications on the previous page.

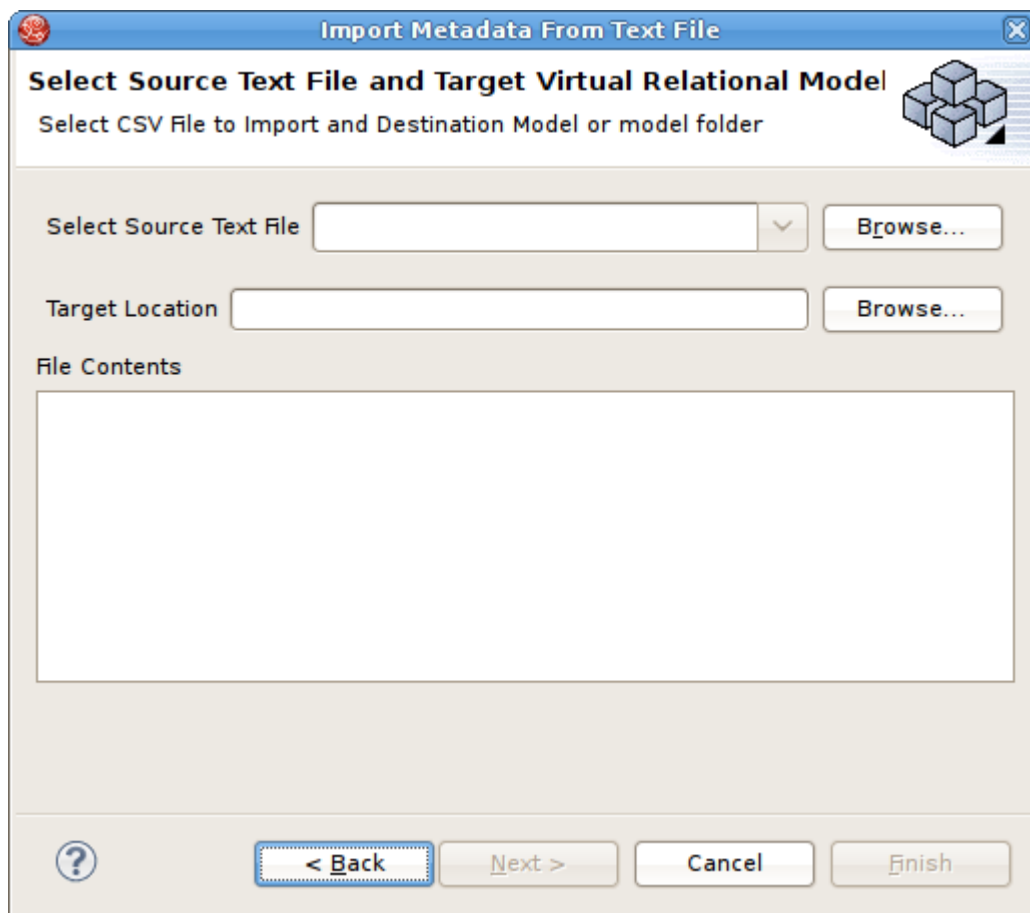
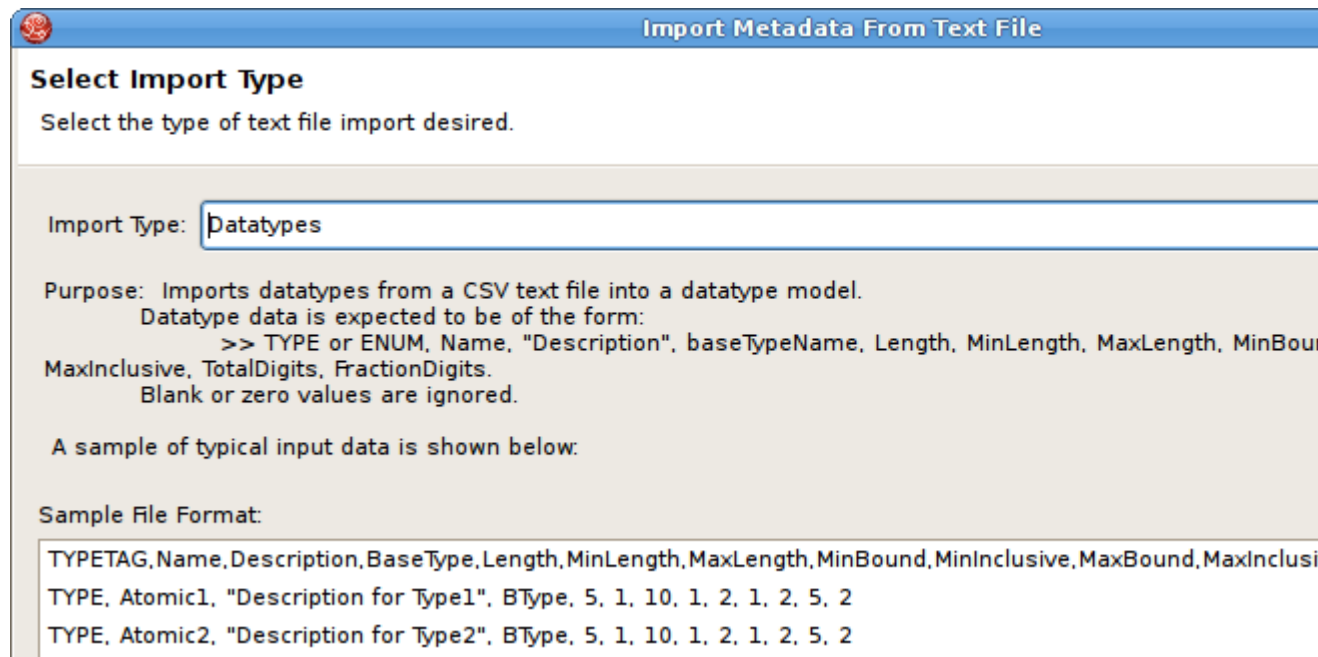


Figure 6.17. Select Source Text File and Target Virtual Relational Model

- **Step 5** - Select an existing relational virtual model as the target location for your new model components using the **Browse...** button to open the Virtual Model Selector Dialog. Select a virtual relational model from your workspace or specify a unique name to create a new model.
- **Step 6** - Select **Finish**.

6.3.3. Import Datatypes Text Importer

- To create datatypes from imported text file metadata:
 - Perform **Steps 1 through 3** (above) and select the **Datatypes** import type, then click **Next >**



Select Import Type

Select the type of text file import desired.

Import Type:

Purpose: Imports datatypes from a CSV text file into a datatype model.
Datatype data is expected to be of the form:
>> TYPE or ENUM, Name, "Description", baseTypeName, Length, MinLength, MaxLength, MinBound, MinInclusive, TotalDigits, FractionDigits, MaxBound, MaxInclusive.
Blank or zero values are ignored.

A sample of typical input data is shown below:

Sample File Format:

```
TYPETAG,Name,Description,BaseType,Length,MinLength,MaxLength,MinBound,MinInclusive,MaxBound,MaxInclusive
TYPE, Atomic1, "Description for Type1", BType, 5, 1, 10, 1, 2, 1, 2, 5, 2
TYPE, Atomic2, "Description for Type2", BType, 5, 1, 10, 1, 2, 1, 2, 5, 2
```

Figure 6.18. Select Import Type - Datatypes

- **Step 4** - In the next page, you'll need to provide a source text file containing the metadata formatted to the specifications on the previous page.

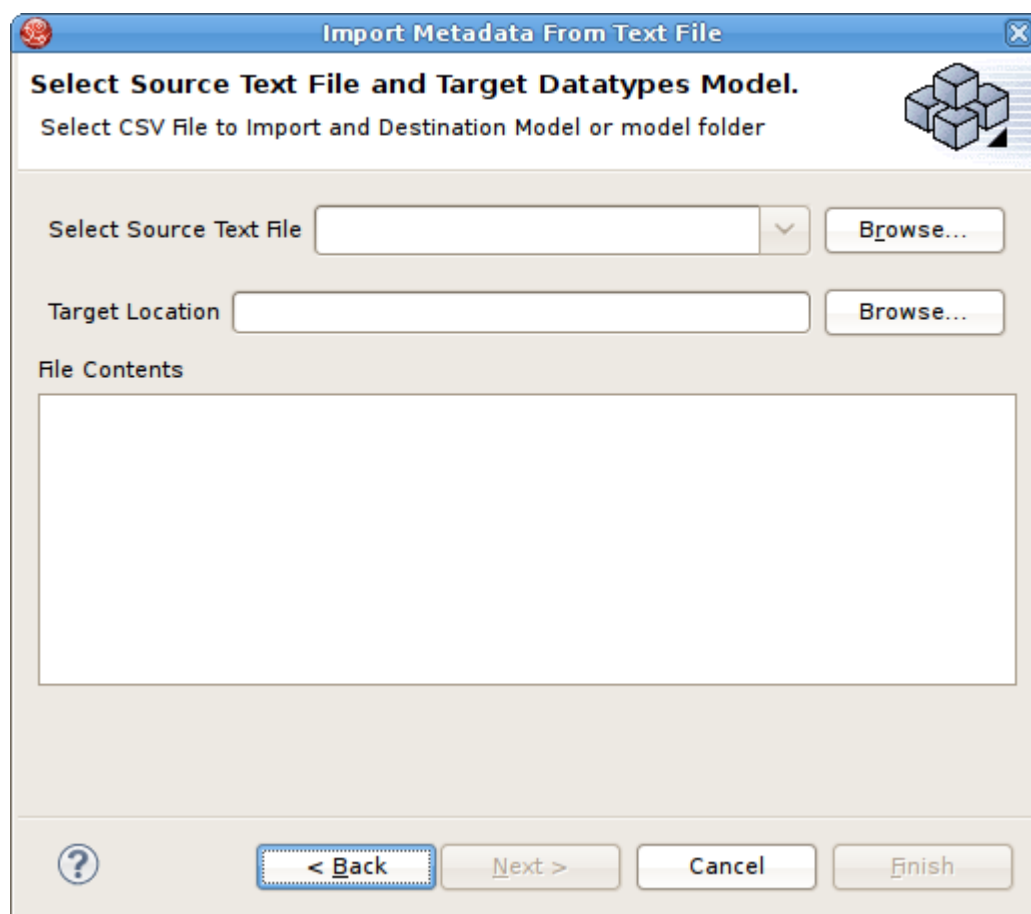


Figure 6.19. Select Source Text File and Datatypes Model

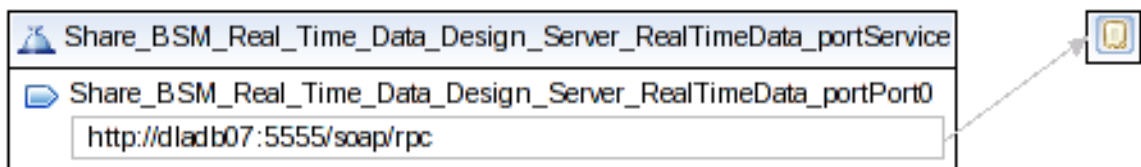
- **Step 5** - Select an existing datatype model as the target location for your new model components using the **Browse...** button to open the Datatypes Model Selector Dialog. Select a datatypes model from your workspace or specify a unique name to create a new model.
- **Step 6** - Select any additional options and choose **Finish**.

6.4. Import WSDL into Relational Source Model

You can create a Relational model by selecting an existing *Web Services Connection Profile* defined by a WSDL file in your workspace or defined by a URL. Designer will interpret the WSDL, locate any associated or dependent XML schema files, generate a physical model to invoke the service, and generate virtual models containing procedures to build and parse the XML declared as the service messages.

- The sample WSDL for this section defines the following:
 - A single **wsdl:service** named `Share_BSM_Real_Time_Data_Design_Server_RealTimeData_portService`

- A single **wsdl:port** named Share_BSM_Real_Time_Data_Design_Server_RealTimeData_portPort0"
- A single **wsdl:portType** named Share_BSM_Real_Time_Data_Design_Server_RealTimeData_portPortType
- A single **wsdl:operation** named OnHand



To create relational models from **WSDL** use the steps below.

- **Step 1** - In **Model Explorer** choose the **File > Import** action in the toolbar or select a project, folder or model in the tree and choose **Import...**
- **Step 2** - Select the import option **Teiid Designer > WSDL into Relational Source Model** and click **Next>**

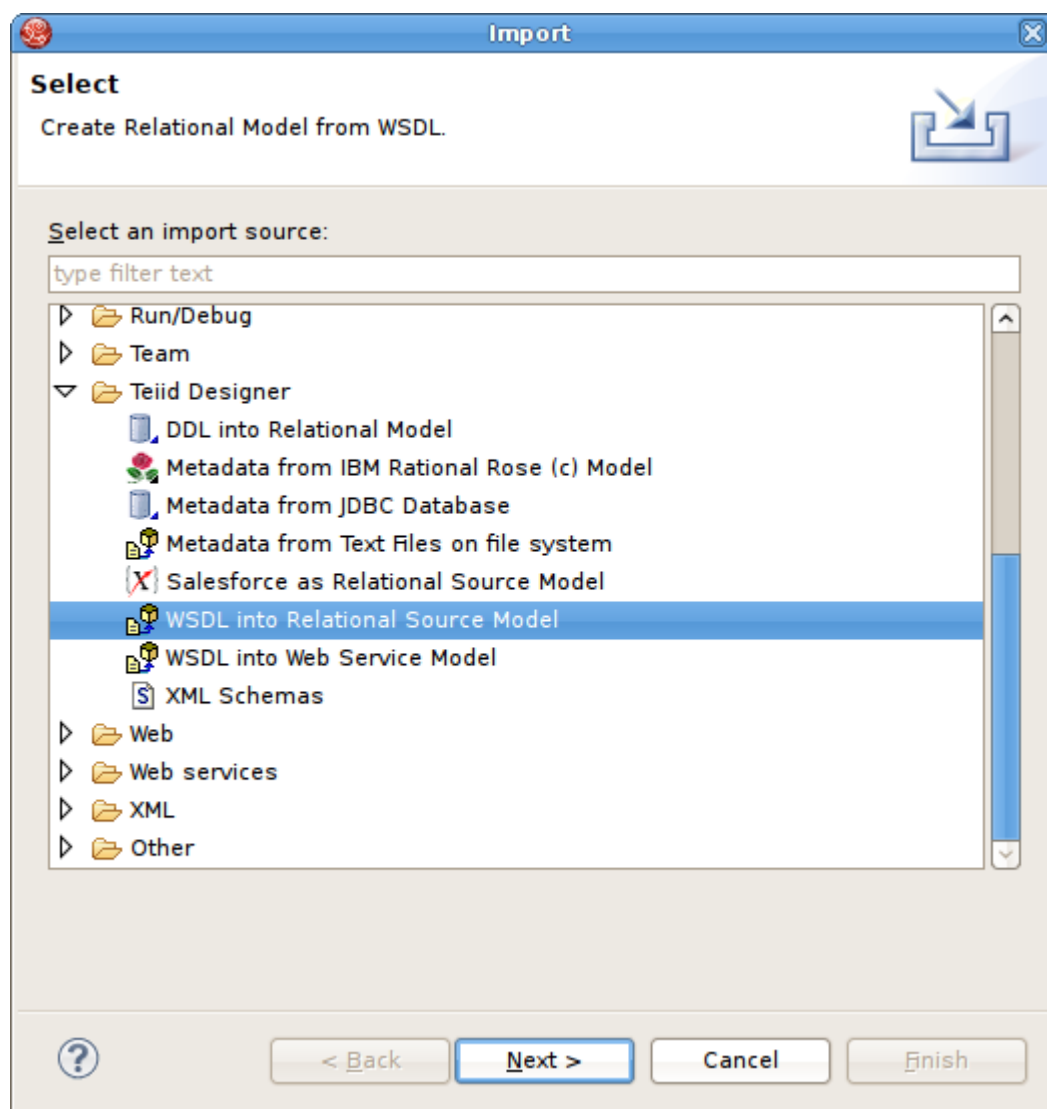


Figure 6.20. Import WSDL as Relational Source Model

- **Step 3** - On the next page select an existing Web Service Connection Profile from the list, or click the **New** Button to create a new profile.

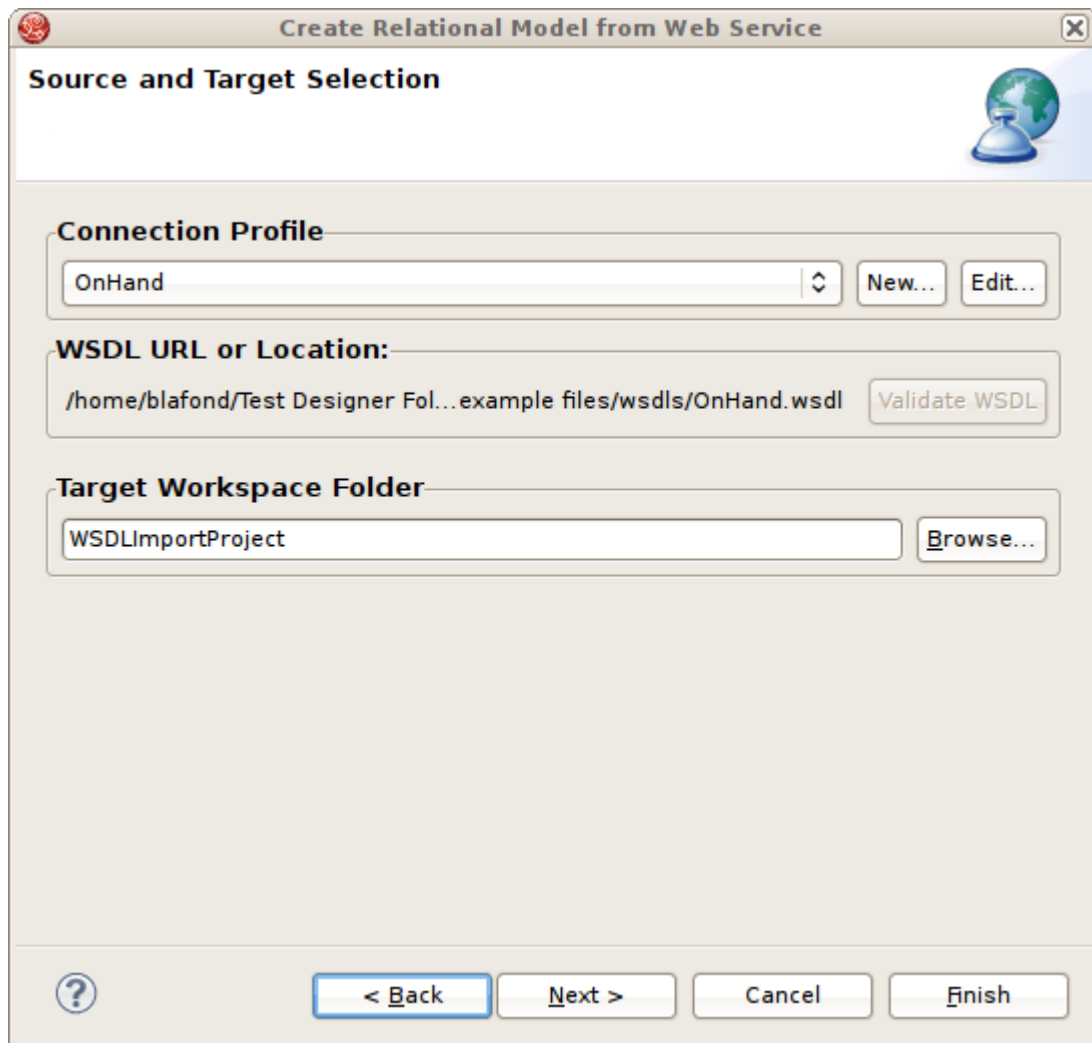


Figure 6.21. Source and Target Selection Dialog

- **Step 4** - When the connection profile is selected, you'll be required to validate your **WSDL** via the **Validate WSDL** button (see above figure). If the WSDL is not valid, or does not comply with WS-I 1.0, the validation failures will be displayed in an error dialog.
- **Step 5** - Select a model project in the **Target Workspace Folder** field via the **Browse...** button. Click **Next>** when the target is defined.
- **Step 6** - The next page allows you to selected individual **Web Service Operations** to model. The default behavior of this page selects **all** available operations in the tree. Operations can be de-selected if they are not being modeled. The Selection Details editor displays static information about the operation such as the names of the input and output messages, and faults thrown by the operation.

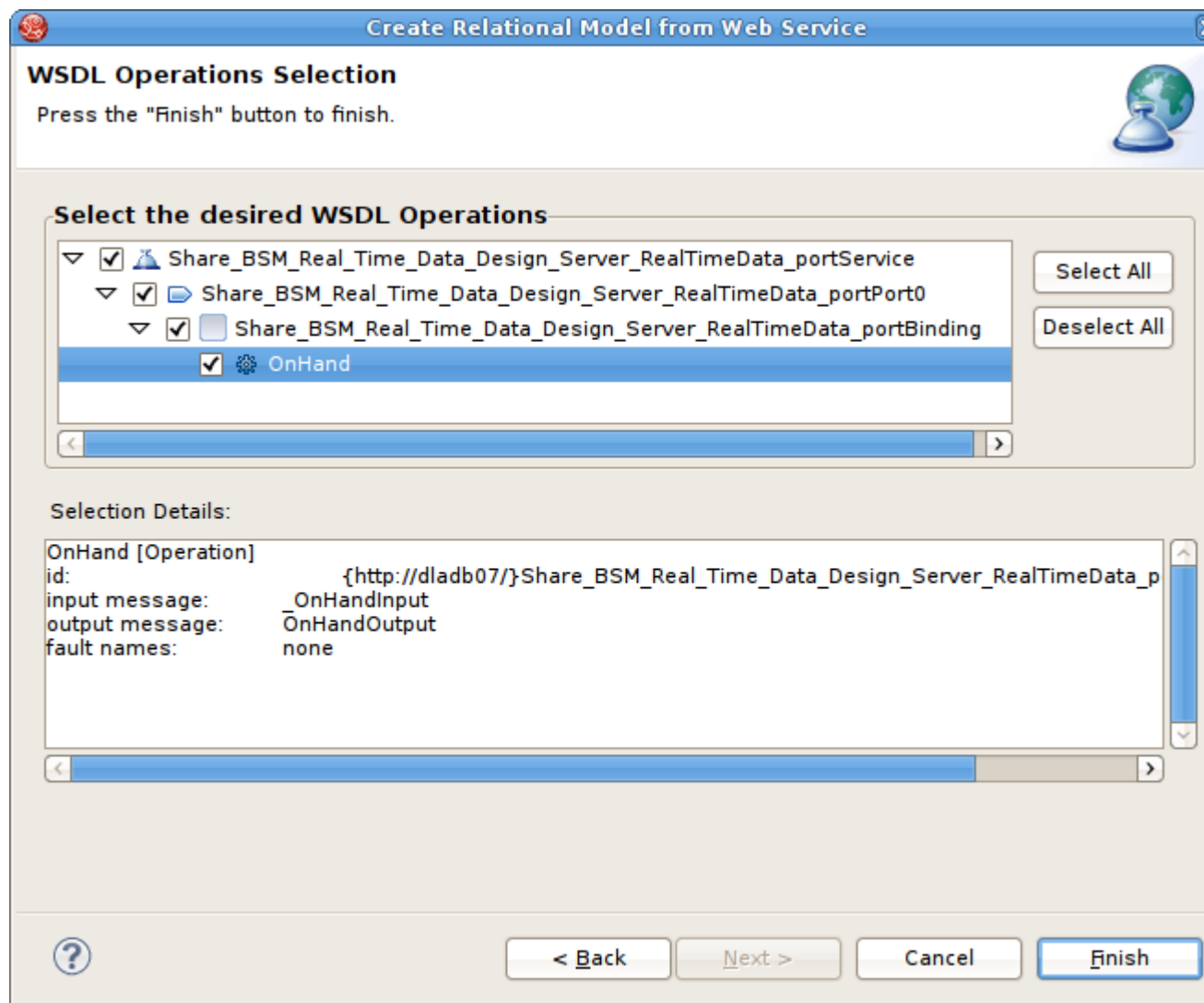


Figure 6.22. WSDL Operations Selection Dialog

- **Step 7** - Click **Finish**. After generation the new models can be found in the specified location in your workspace.

In the **Model Explorer** you can see the importer created the following:

- A single **physical model** containing a single procedure called invoke. This model and procedure correspond to the single port declared in the WSDL. For a WSDL with multiple **wsdl:port** declarations the importer would create a physical model with an invoke procedure for each **wsdl:port** declaration. This facilitates binding each physical model to its own Teiid WebService Connector and allowing the connector to supply the endpoint.
- A single **view model** containing a **catalog** named after the operation declared in the WSDL. For a WSDL with multiple **wsdl:operation** declarations, the importer would create multiple catalogs. The **catalog** contains procedure declarations that use the **SQLXML** calls supported by Teiid to create and parse the XML defined as the operation messages.

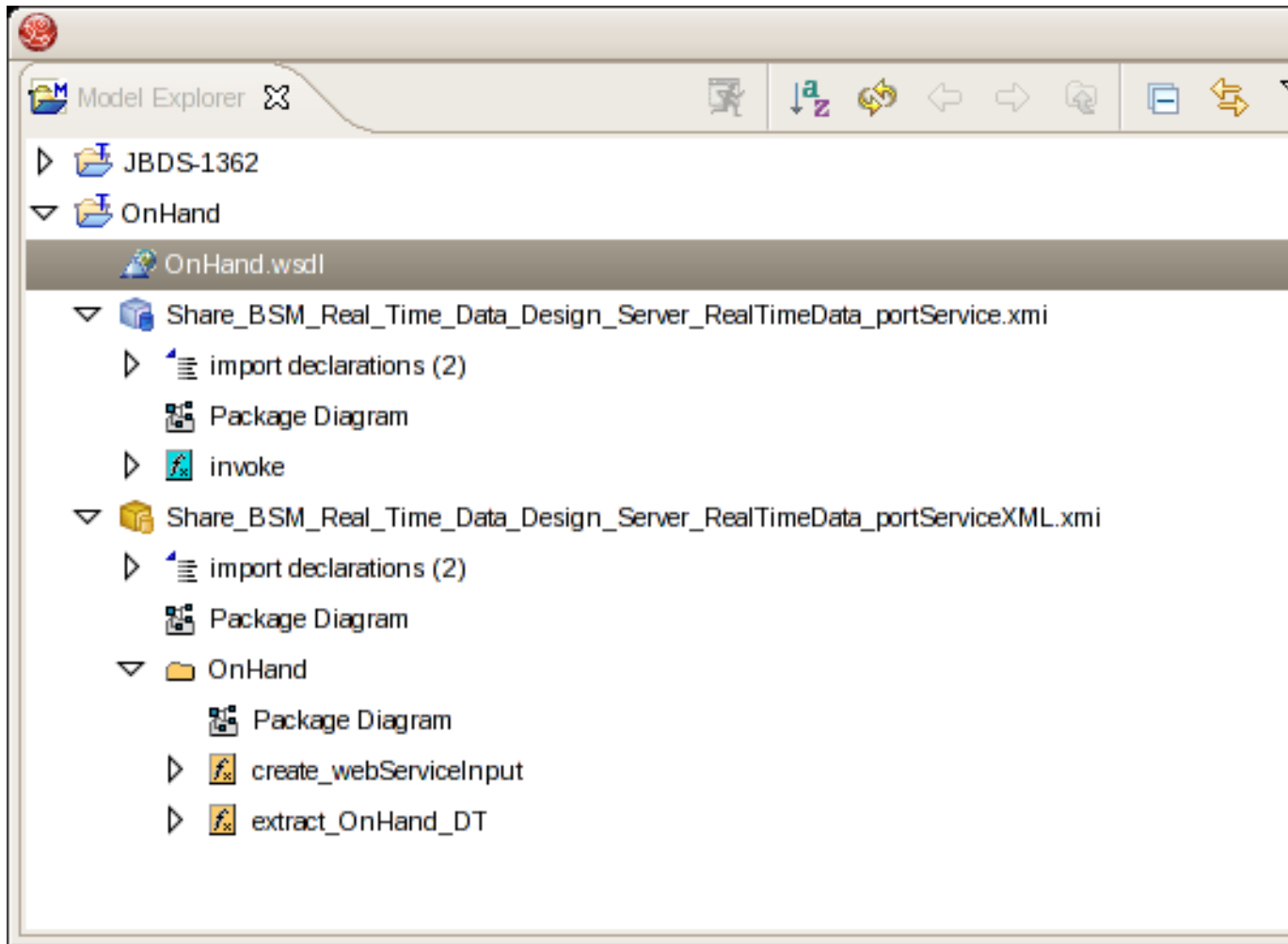


Figure 6.23. Importing the WSDL file into the Model Explorer

The user can now use the procedures defined by the importer in a higher level view model that builds the request, executes the service via the invoke procedure, and extracts the results from the response.

6.5. Import WSDL Into Web Service


You can create a **Web Service** model by selecting a **WSDL** file in your workspace, importing **WSDL** files from the file system or by defining a URL. The Teiid Designer will interpret the **WSDL**, locate any associated or dependent **XML Schema** files, generate an **XML View** of the schema components and create a **Web Service** model representing the interfaces and operations defined in the **WSDL**.

- There are three options for selecting the **WSDL** for your **Web Service** generation
 - **Workspace Location**
 - **File System Location**

- URL

Detailed steps for each of these options is described below, as well as a description of how the wizard handles **WSDL** errors.

6.5.1. Import WSDL From Workspace Location

- You can create a **Web Service** model by selecting a **WSDL** file from your workspace.
- **Step 1** - Choose the **File Import** choose the **File > Import** action

in the toolbar or select a project, folder or model in the tree and choose **Import...**
- **Step 2** - Select the import option **Metadata Modeling > WSDL into Web Service Model** option shown below and click **Next>**

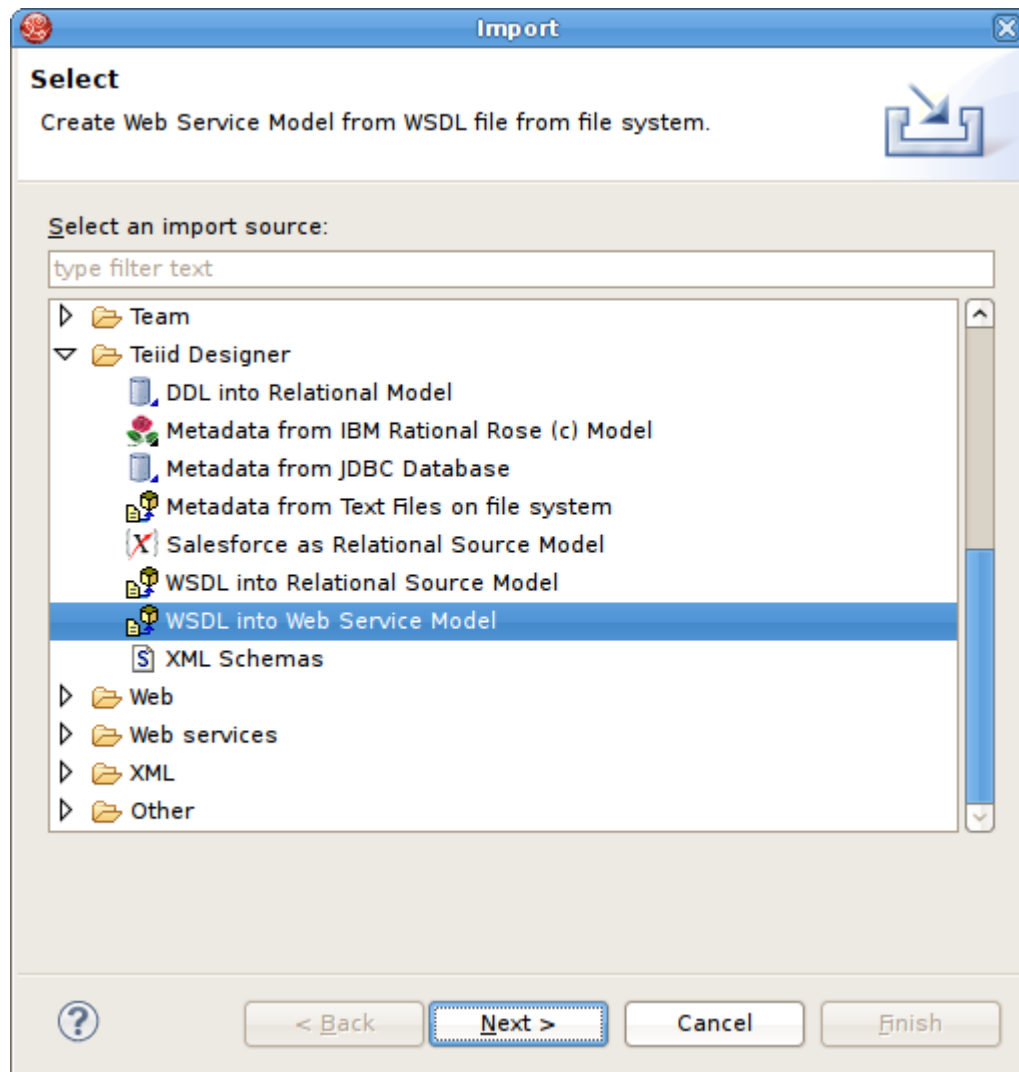


Figure 6.24. WSDL File Into Web Service Dialog

- **Step 3** - Input a valid name for your **Web Service** model and select the **Workspace...** button. Locate your workspace **WSDL** file in the selection dialog and click **OK>**. Click **Next>** to continue.

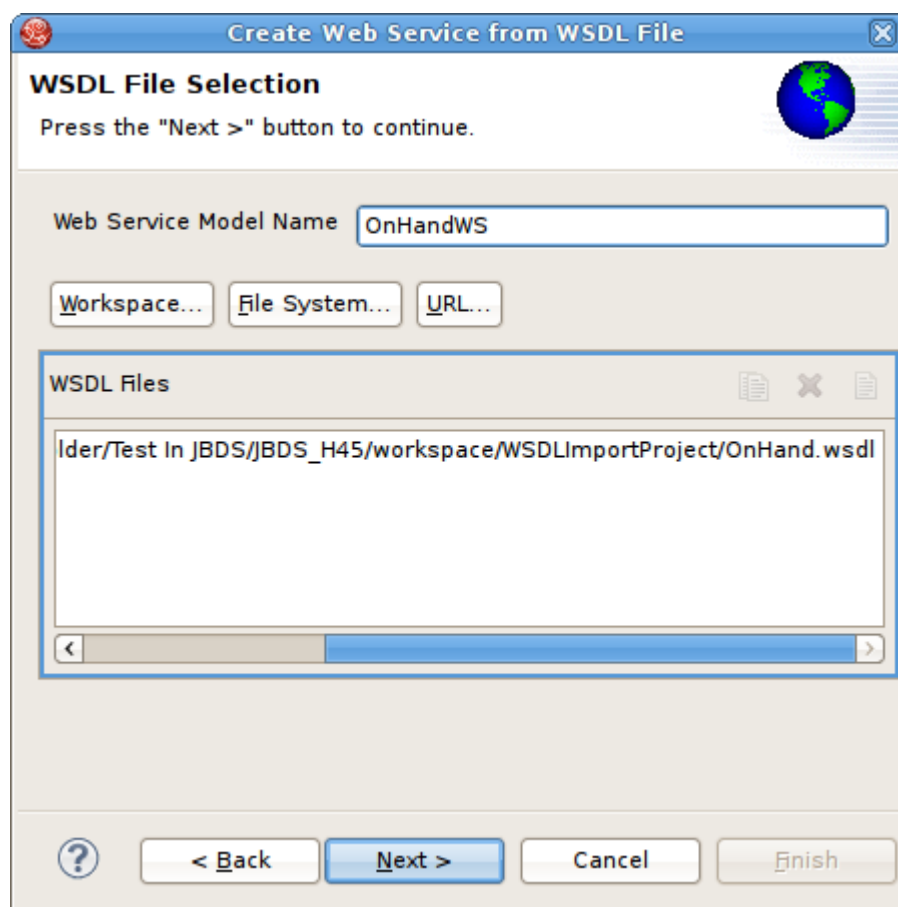


Figure 6.25. WSDL File Selection Dialog

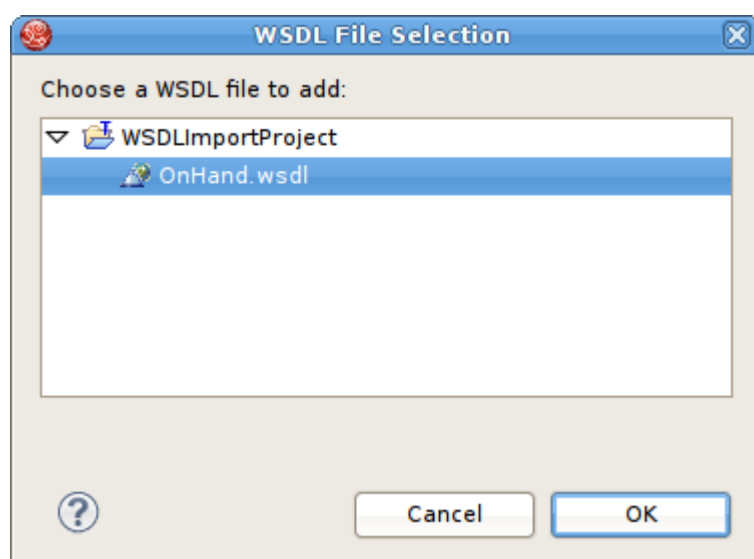


Figure 6.26. WSDL File Workspace Selection Dialog

**Note**

- If no **WSDL** is selected or specified then the importer will only create an empty **Web Service** model. No **XML Schema** or **XML View** models will be generated.
- Any referenced files (**WSDLs** or schemas) must either be embedded in the **WSDL** file or exist on your file system.

- **Step 4** - The next page is titled **Namespace Resolution**. This page identifies successful and errant **WSDL** namespace resolution. The main **WSDL** document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main **WSDL** file imports other **WSDL** files that cannot be resolved. If no errors, select **Next** to proceed, or **Finish** (if enabled) to complete with default options.

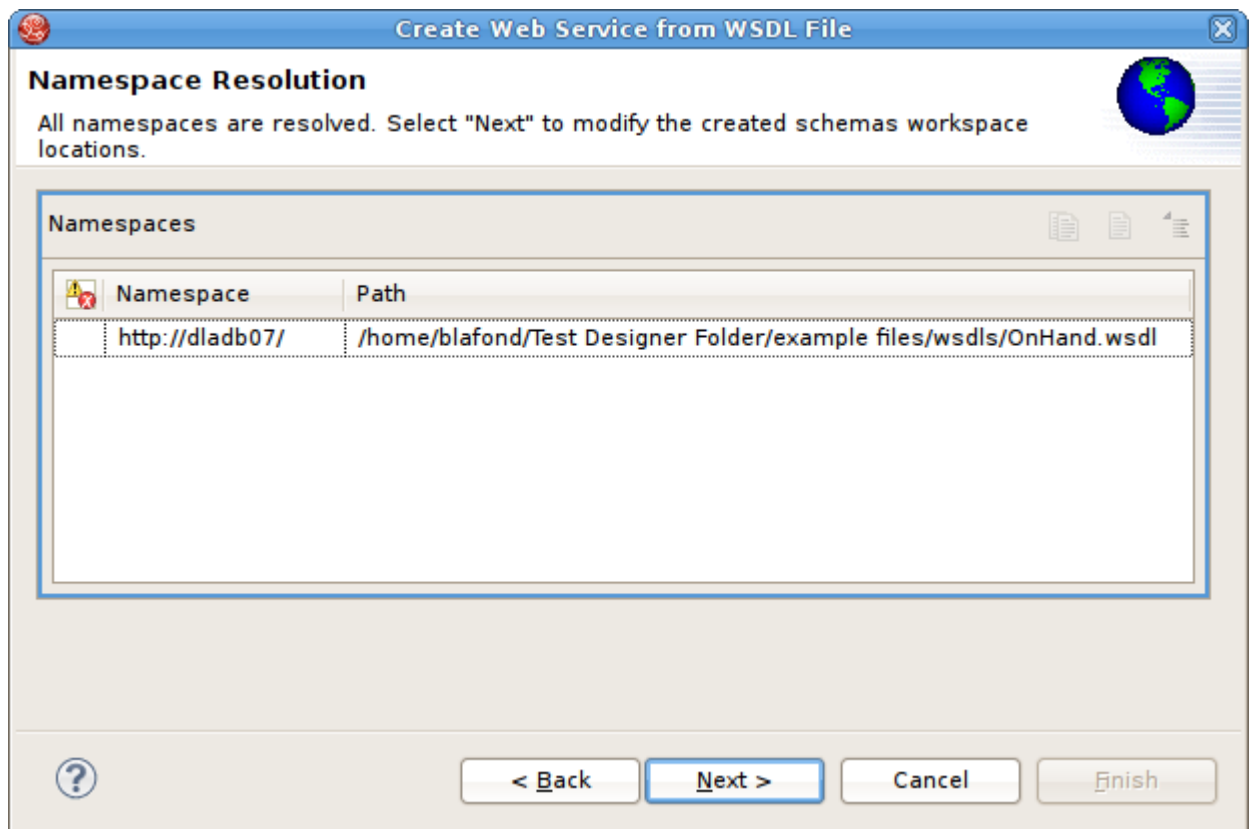


Figure 6.27. Namespace Resolution Dialog

- **Step 5** - The next page **WSDL Operations Selection** allows customizing the resulting content of your **Web Service** model by selecting/deselecting various operations and interfaces in the following dialog.

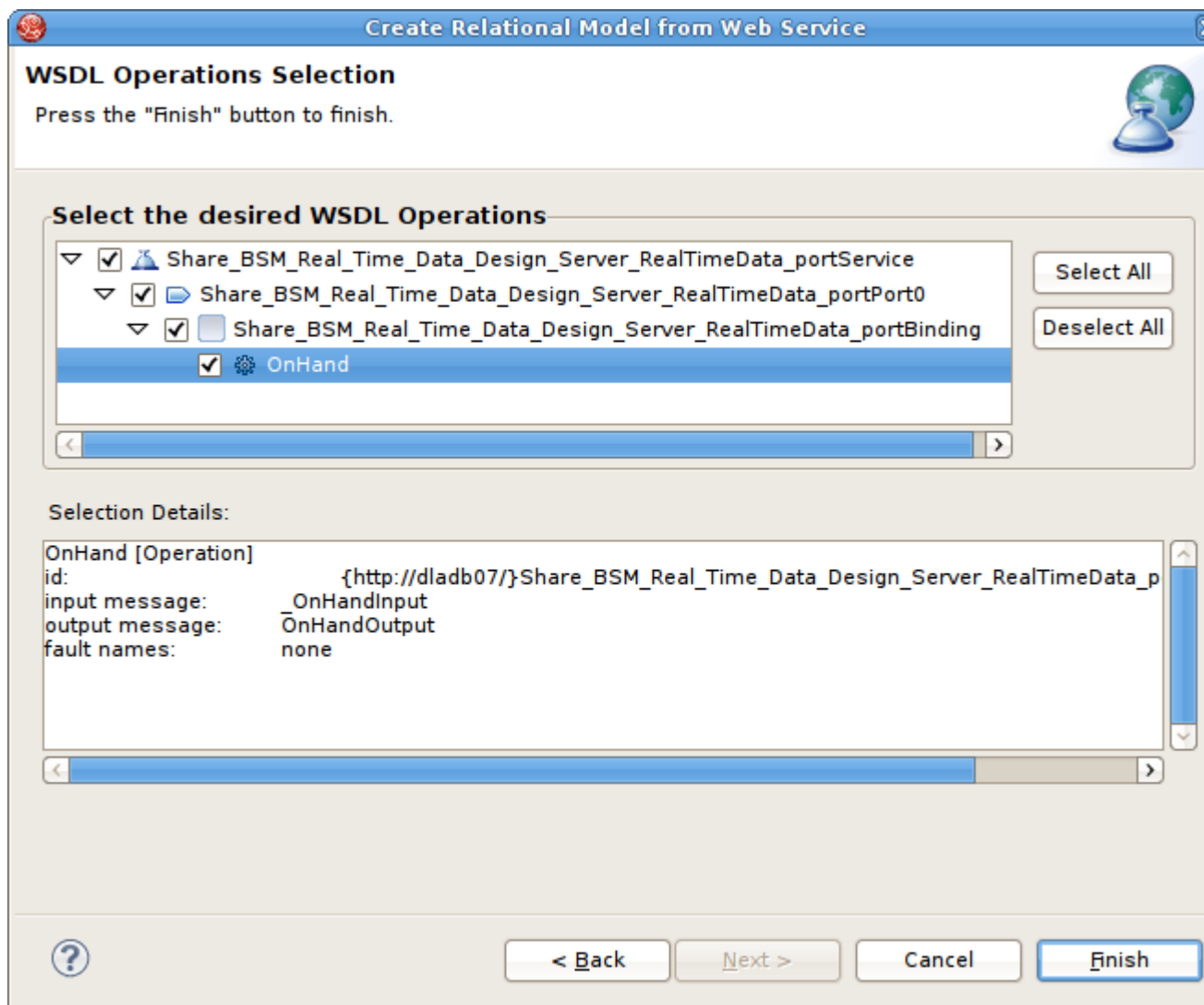


Figure 6.28. Namespace Resolution Dialog

- **Step 6** - The next page is titled **Schema Workspace Location Selection**. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select **Next** to proceed, or **Finish** to complete with default option

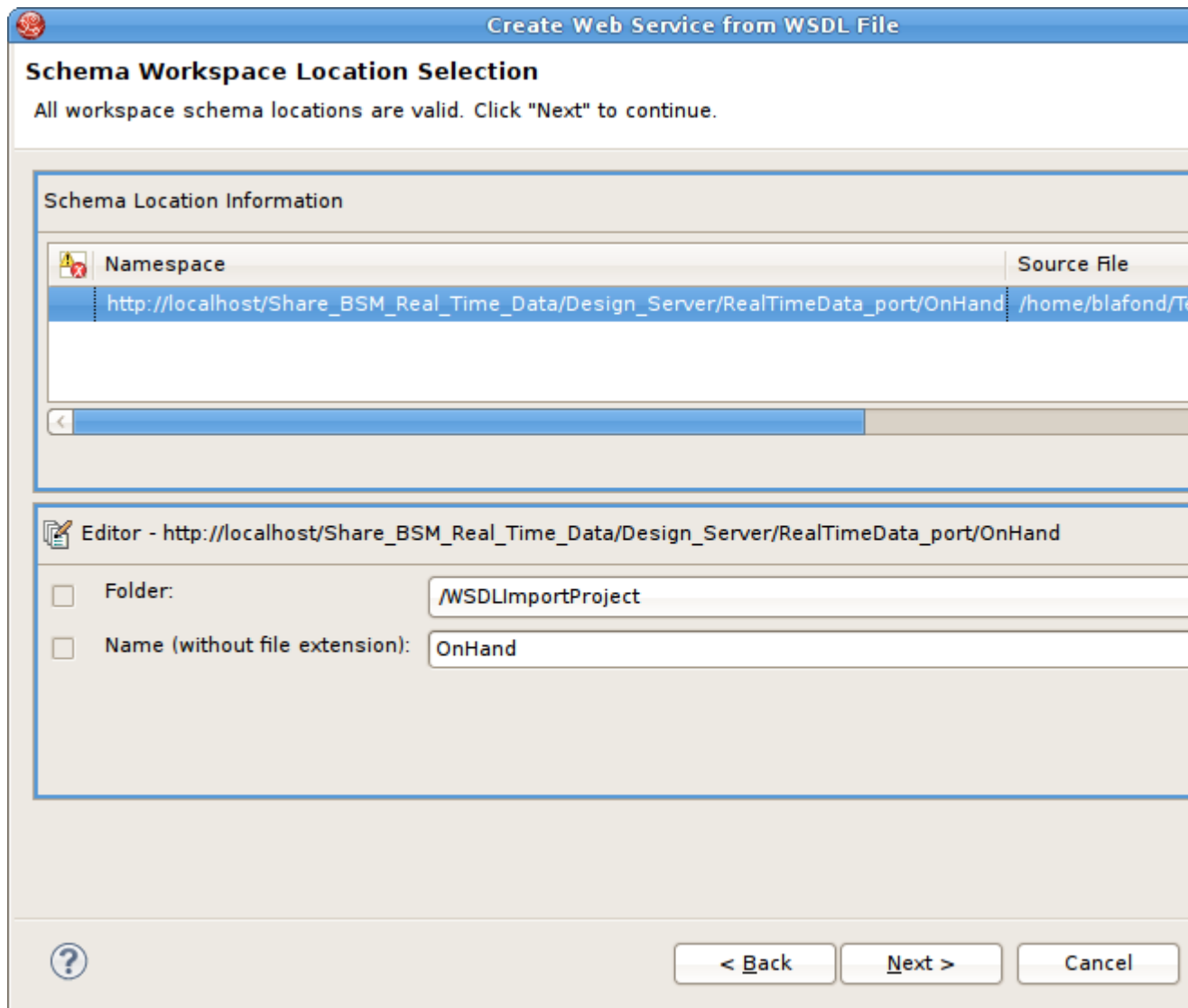


Figure 6.29. Namespace Resolution Dialog

- **Step 7** - The last page titled **XML Model Generation** allows you to change the name of the **XML View** model if the **Generate virtual XML document model** is checked. Input desired name or use the default name provide. Select **Finish** to complete.

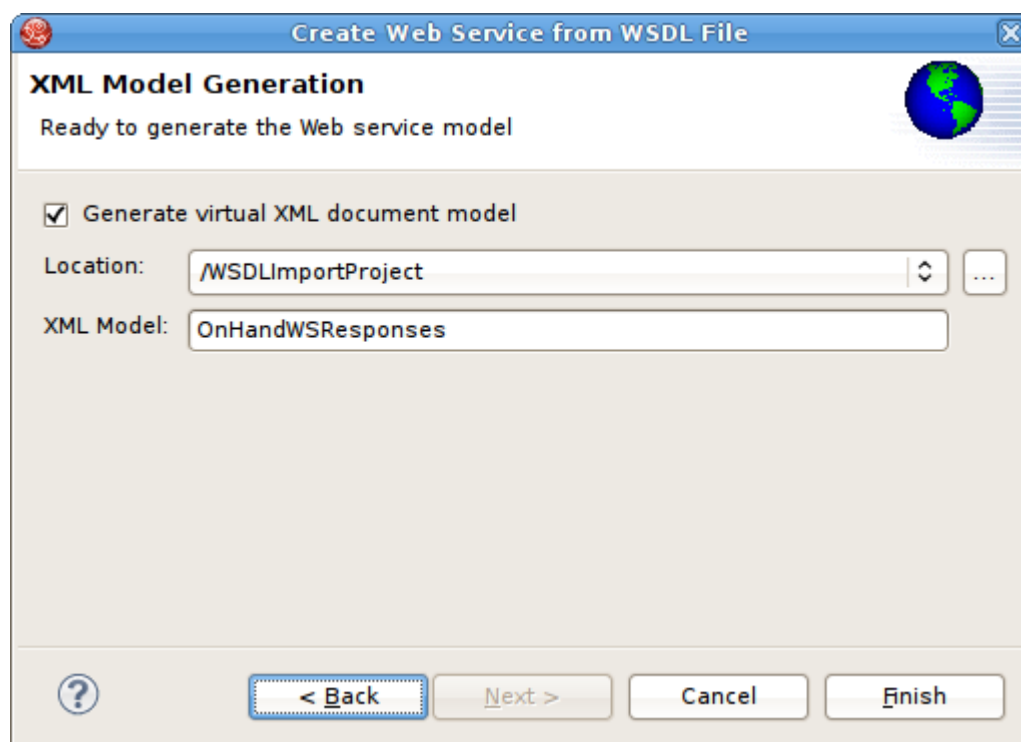


Figure 6.30. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, an error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

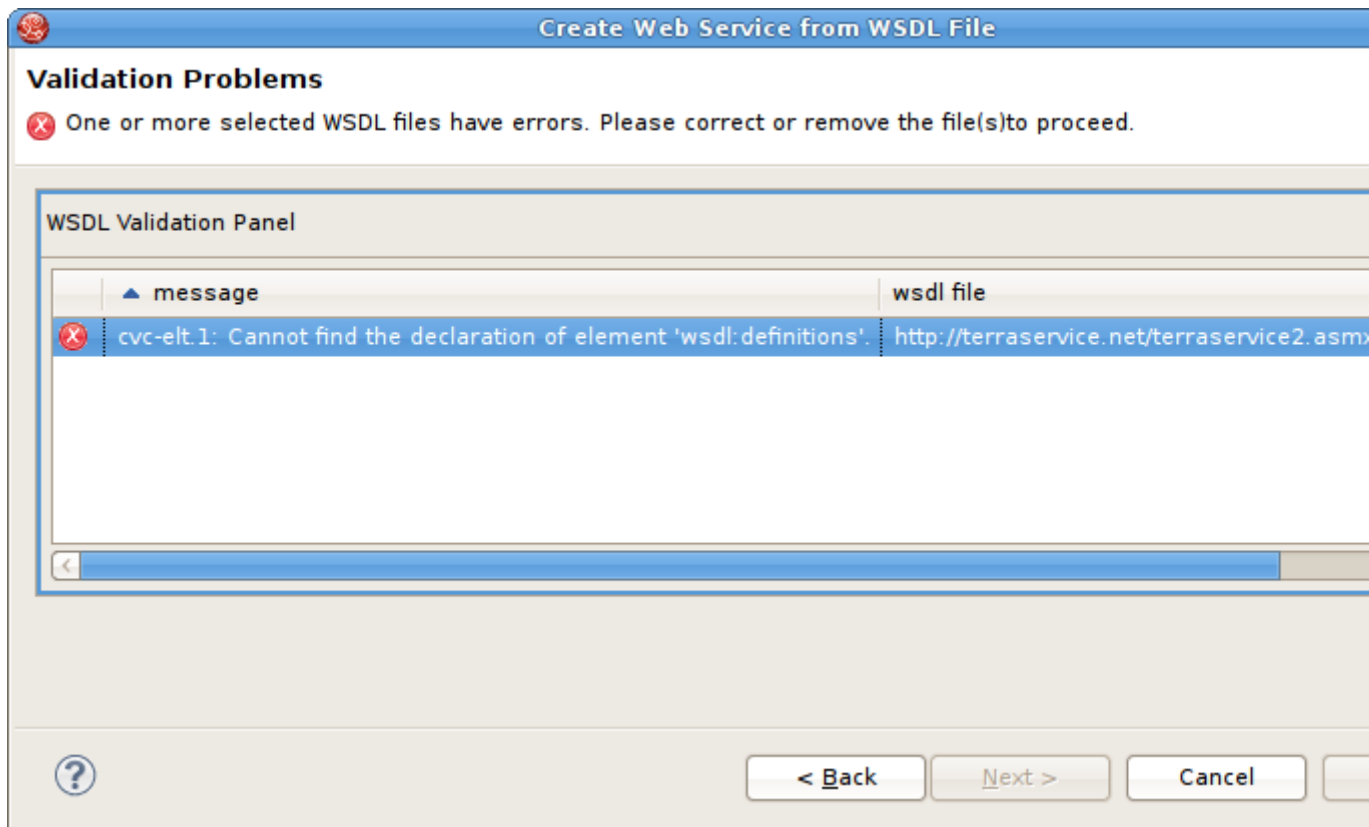



Figure 6.31. WSDL Validation Problems Dialog

6.5.2. Import WSDL From File System Location

- You can create a **Web Service** model by selecting a **WSDL** file from your local file system.
 - Step 1** - Choose the **File Import** choose the **File > Import** action  in the toolbar or select a project, folder or model in the tree and choose **Import...**
 - Step 2** - Select the import option **Metadata Modeling > WSDL File into Web Service Model** and click **Next>**

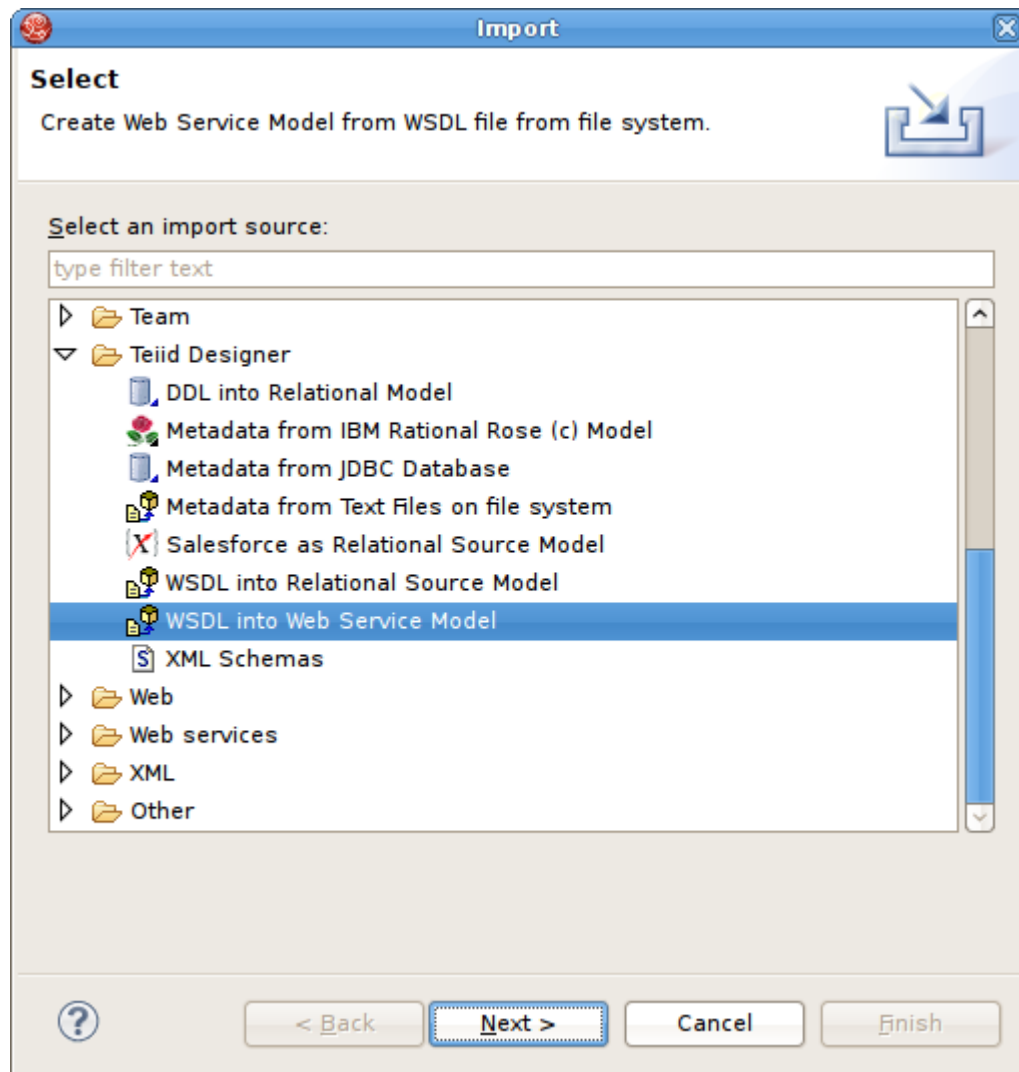


Figure 6.32. WSDL File Into Web Service Dialog

- **Step 3** - Input a valid name for your **Web Service** model and select the **File System...** button. Locate your file system **WSDL** file in the selection dialog and click **OK>**.

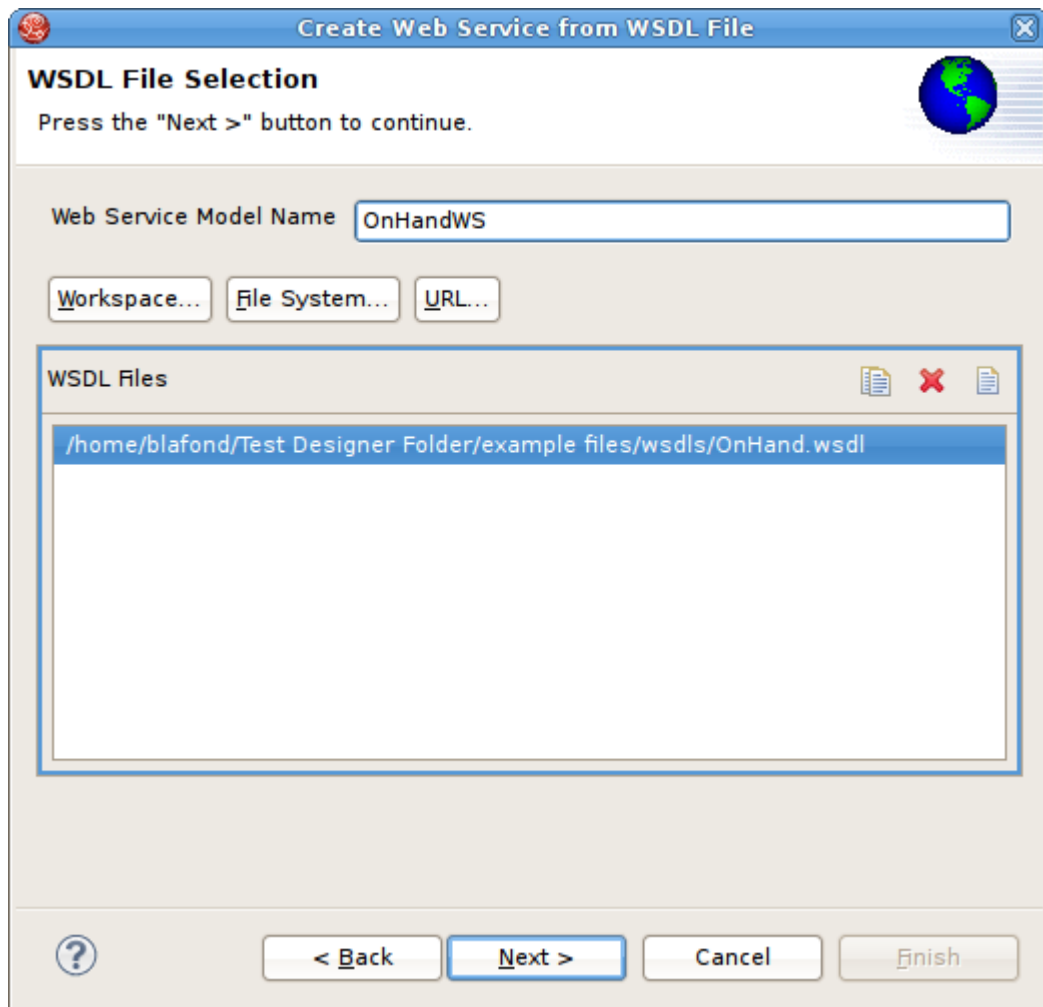


Figure 6.33. WSDL File Selection Dialog



Note

- If no **WSDL** is selected or specified then the importer will only create an empty **Web Service** model. No **XML Schema** or **XML View** models will be generated.
- Any referenced files (**WSDLs** or schemas) must either be embedded in the **WSDL** file or exist on your file system.

- **Step 4** - The next page is titled **Namespace Resolution**. This page identifies successful and errant **WSDL** namespace resolution. The main **WSDL** document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main **WSDL** file imports other **WSDL** files that cannot be resolved. If no errors, select **Next** to proceed, or **Finish** (if enabled) to complete with default options.

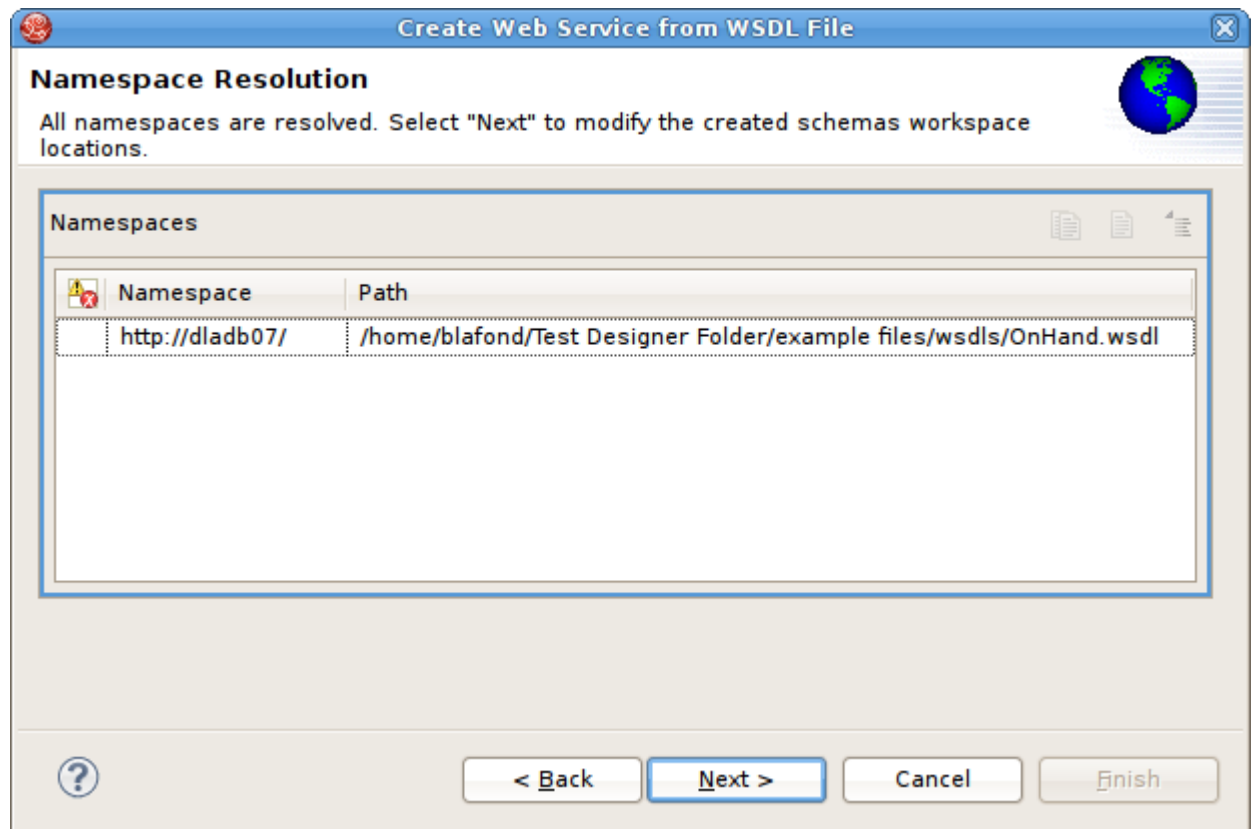


Figure 6.34. Namespace Resolution Dialog

- **Step 5** - The next page **WSDL Operations Selection** allows customizing the resulting content of your **Web Service** model by selecting/deselecting various operations and interfaces in the following dialog.

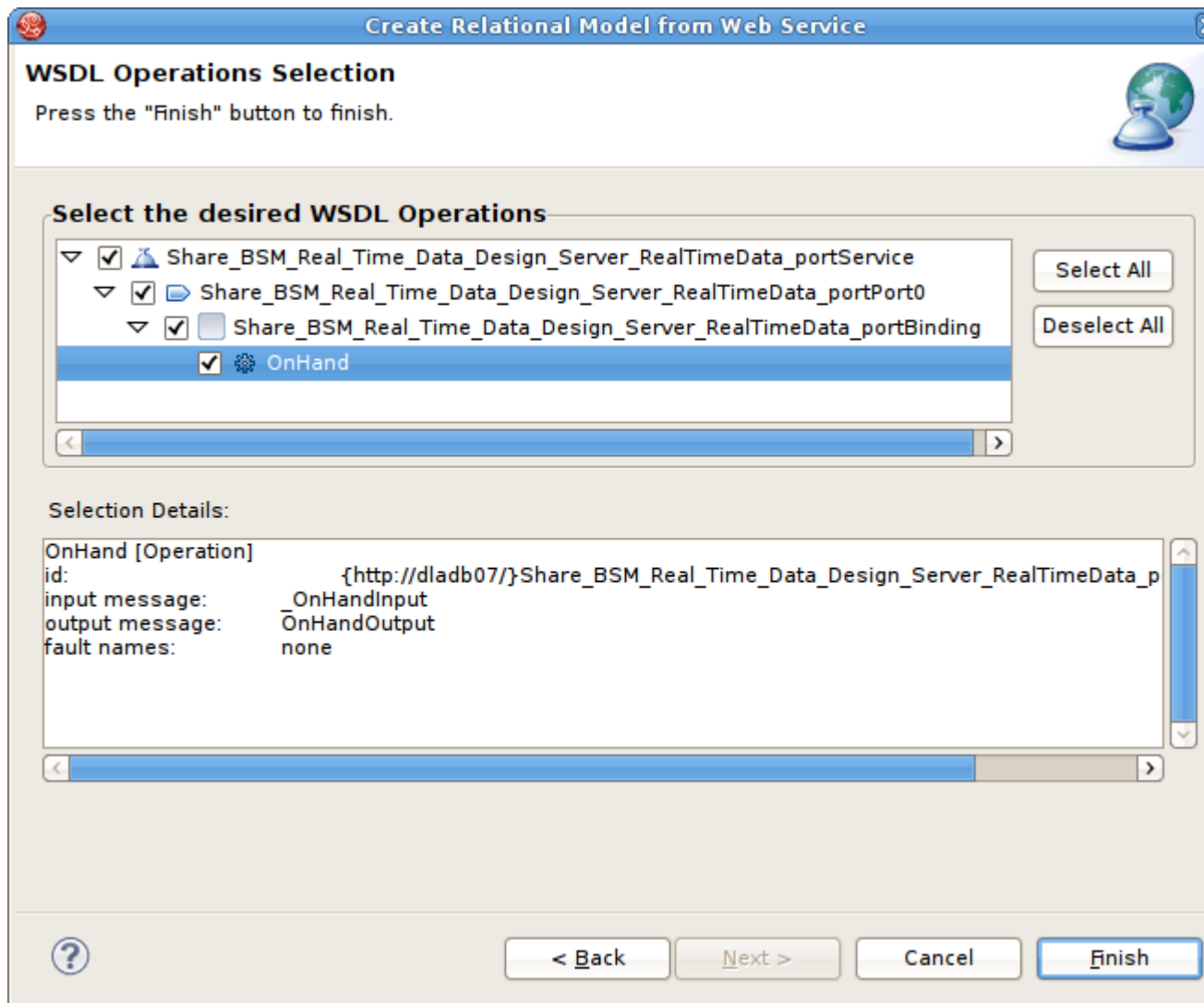


Figure 6.35. Namespace Resolution Dialog

- **Step 6** - The next page is titled **Schema Workspace Location Selection**. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select **Next** to proceed, or **Finish** to complete with default option

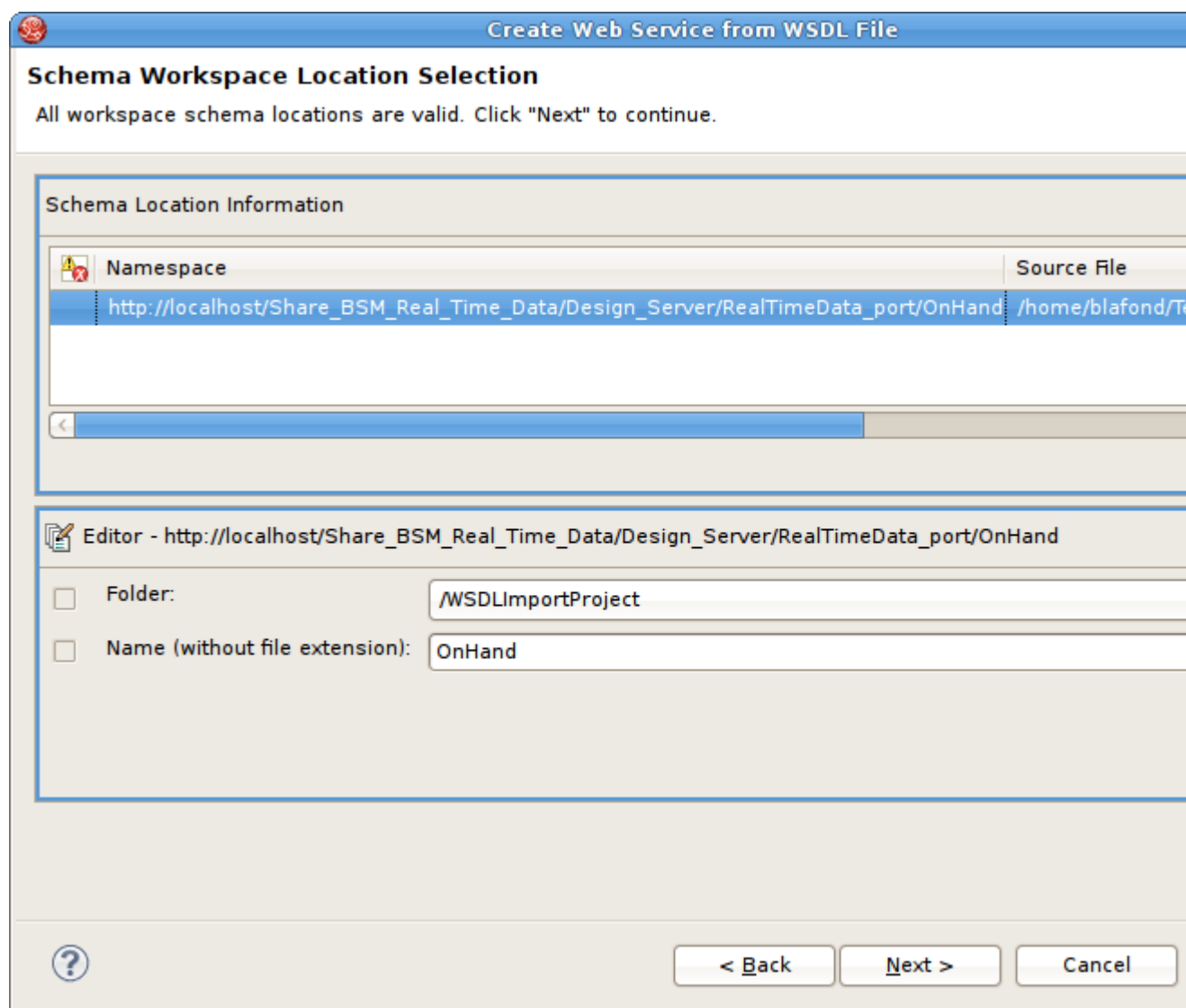


Figure 6.36. Namespace Resolution Dialog

- **Step 7** - The last page titled **XML Model Generation** allows you to change the name of the **XML View** model if the **Generate virtual XML document model** is checked. Input desired name or use the default name provide. Select **Finish** to complete.

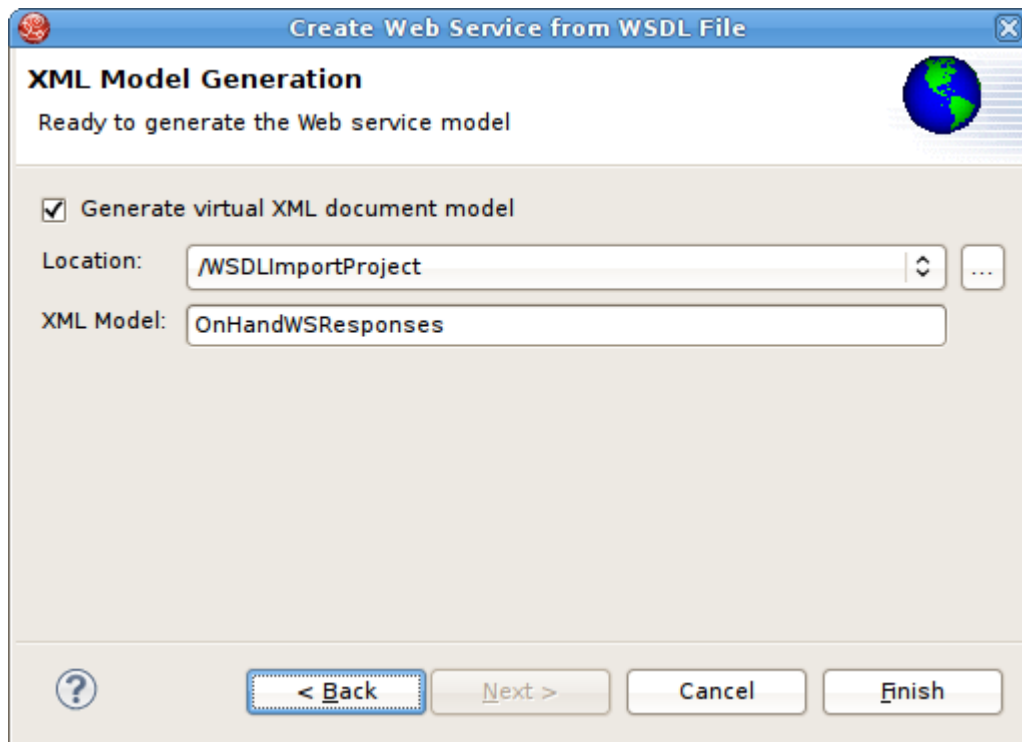


Figure 6.37. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, an error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

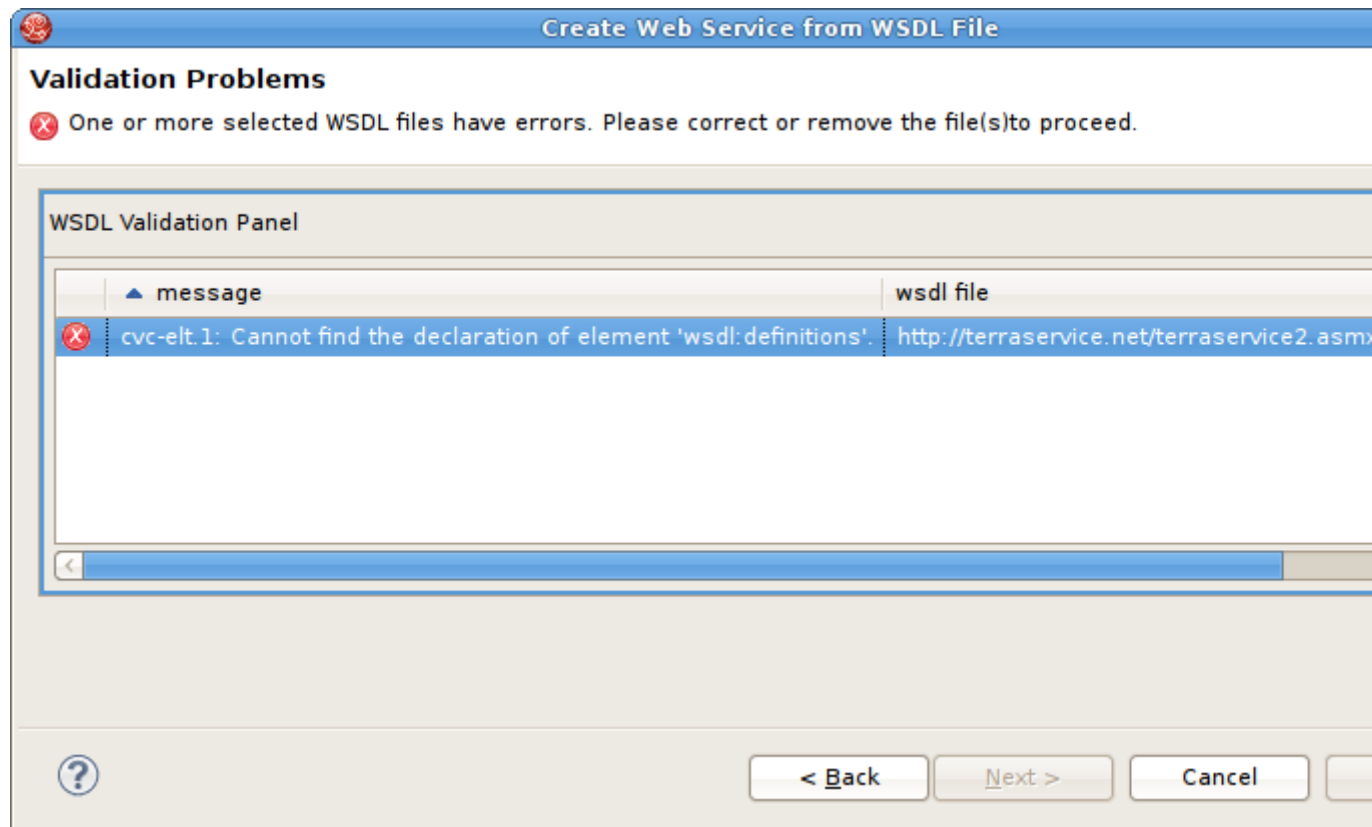



Figure 6.38. WSDL Validation Problems Dialog

6.5.3. Import WSDL From URL

- You can create a **Web Service** model by selecting a **WSDL** file based on a URL.
 - **Step 1** - Choose the **File Import** choose the **File > Import** action  in the toolbar or select a project, folder or model in the tree and choose **Import...**
 - **Step 2** - Select the import option **Metadata Modeling > WSDL File into Web Service Model** and click **Next>**

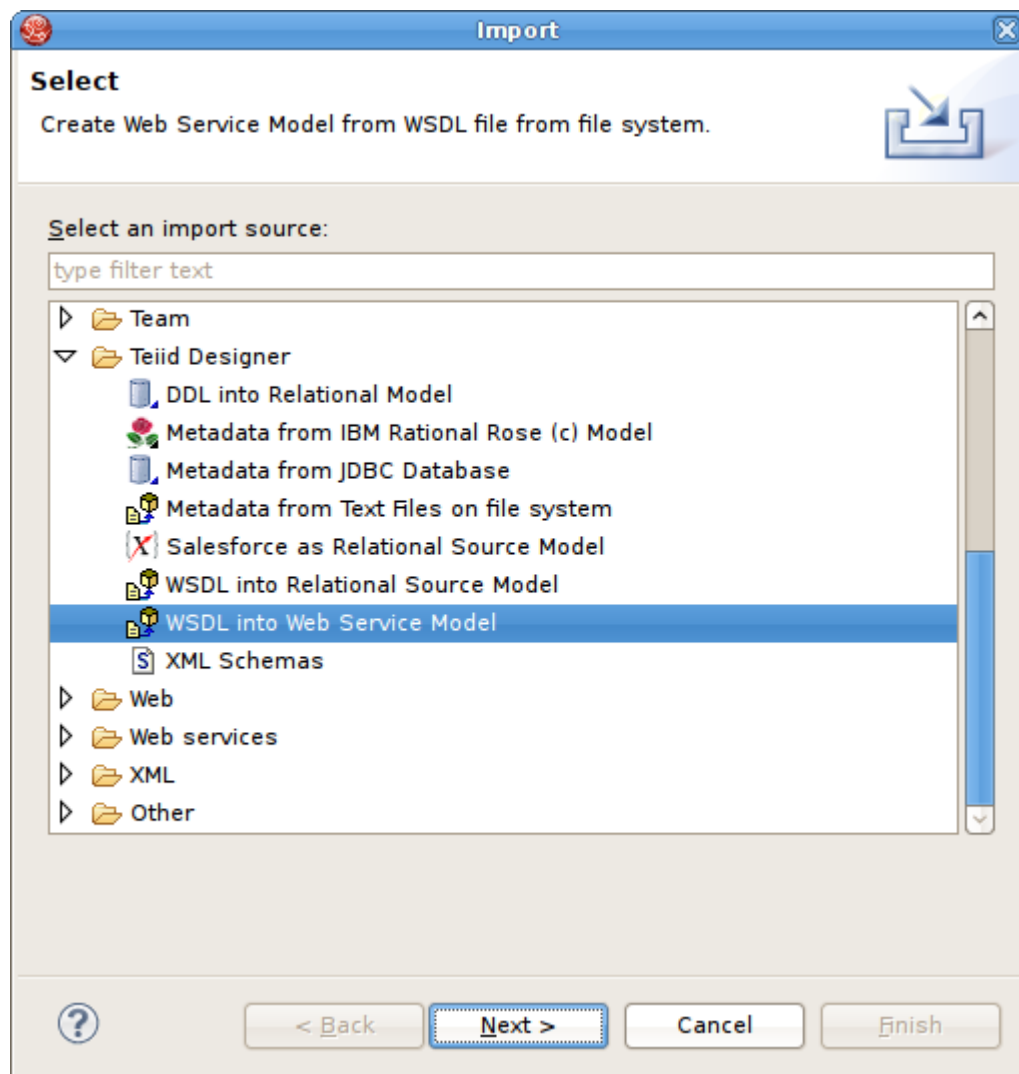


Figure 6.39. WSDL File Into Web Service Dialog

- **Step 3** - Input a valid name for your **Web Service** model and select the **URL...** button. Enter a valid WSDL URL in the dialog click **OK**>. Click **Next**> to continue.

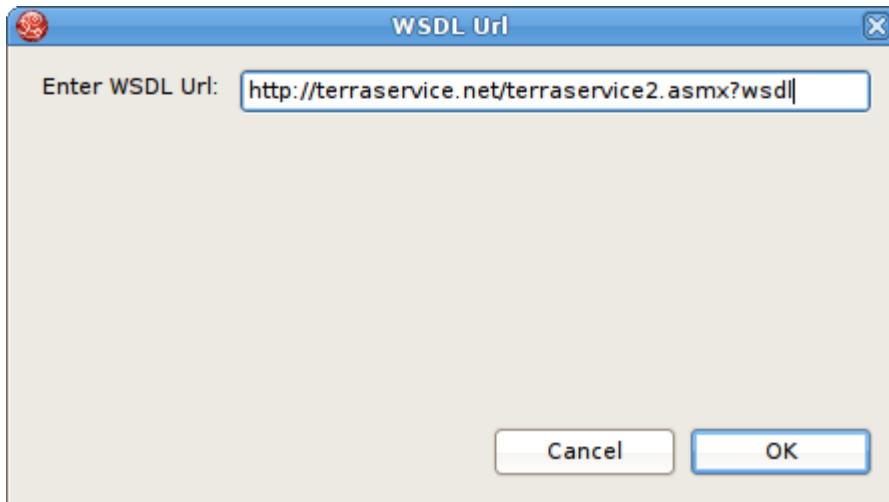


Figure 6.40. WSDL URL Dialog



Note

- If no **WSDL** is selected or specified then the importer will only create an empty **Web Service** model. No **XML Schema** or **XML View** models will be generated.
 - Any referenced files (**WSDLs** or schemas) must either be embedded in the **WSDL** file or exist on your file system.
-
- **Step 4** - The next page is titled **Namespace Resolution**. This page identifies successful and errant **WSDL** namespace resolution. The main **WSDL** document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main **WSDL** file imports other **WSDL** files that cannot be resolved. If no errors, select **Next** to proceed, or **Finish** (if enabled) to complete with default options.

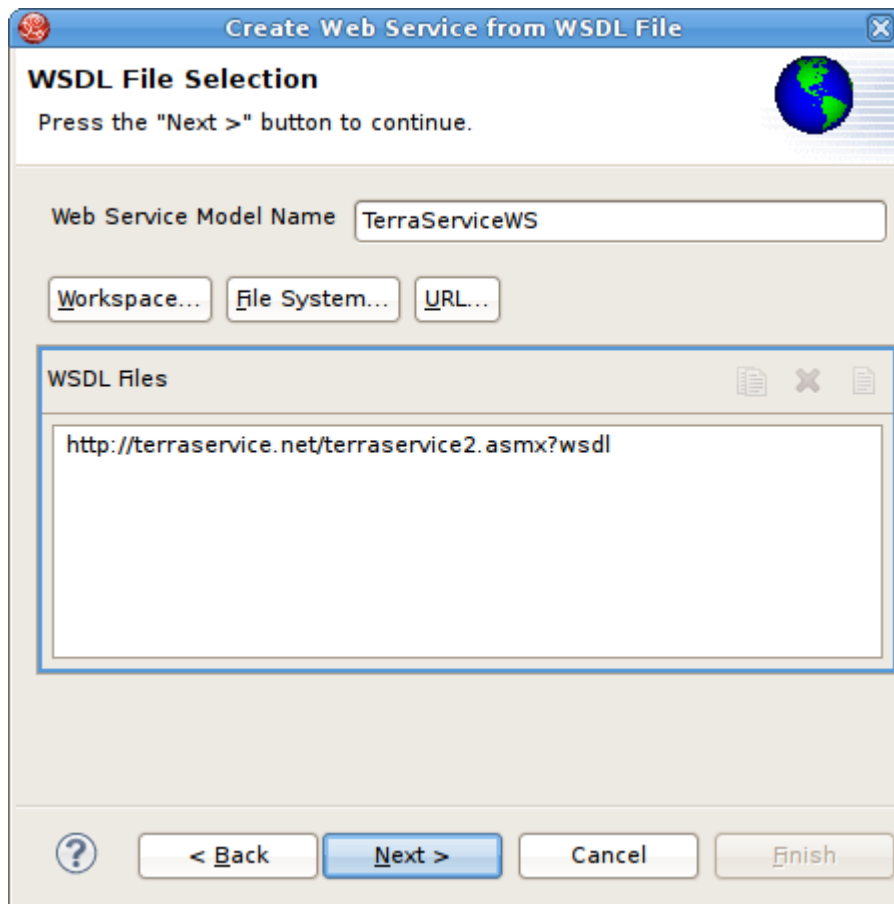


Figure 6.41. Namespace Resolution Dialog

- **Step 5** - The next page **WSDL Operations Selection** allows customizing the resulting content of your **Web Service** model by selecting/deselecting various operations and interfaces in the following dialog.

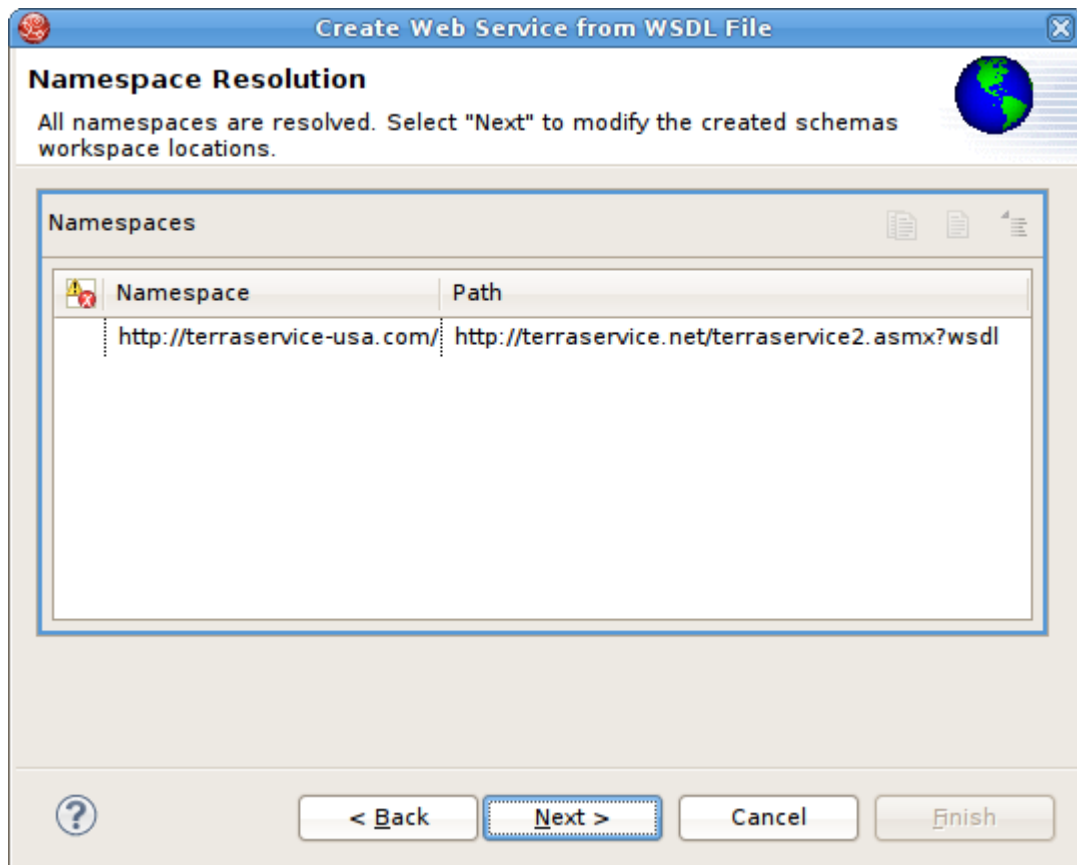


Figure 6.42. Namespace Resolution Dialog

- **Step 6** - The next page is titled **Schema Workspace Location Selection**. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select **Next** to proceed, or **Finish** to complete with default option

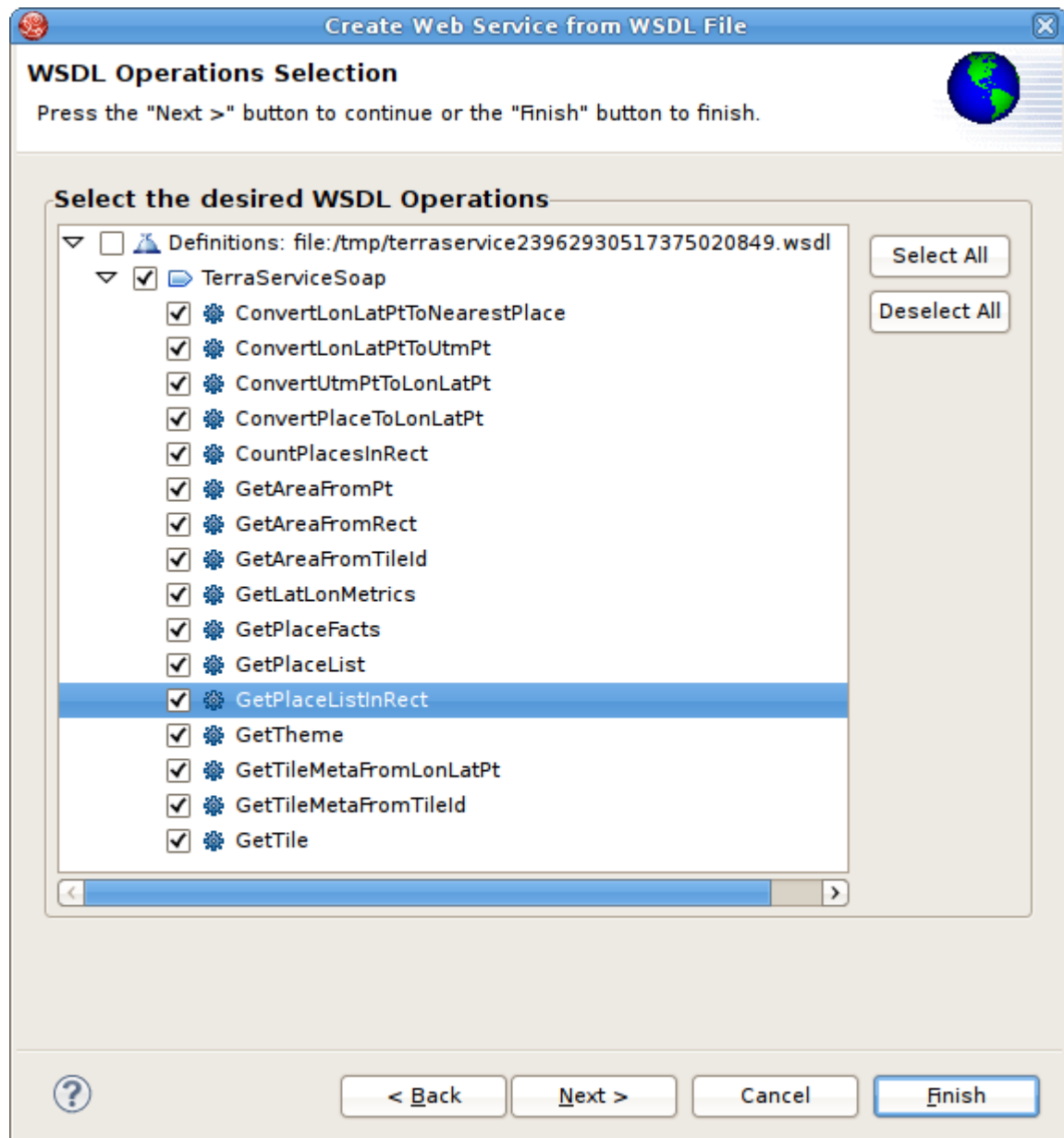


Figure 6.43. Namespace Resolution Dialog

- **Step 7** - The last page titled **XML Model Generation** allows you to change the name of the **XML View** model if the **Generate virtual XML document model** is checked. Input desired name or use the default name provide. Select **Finish** to complete.

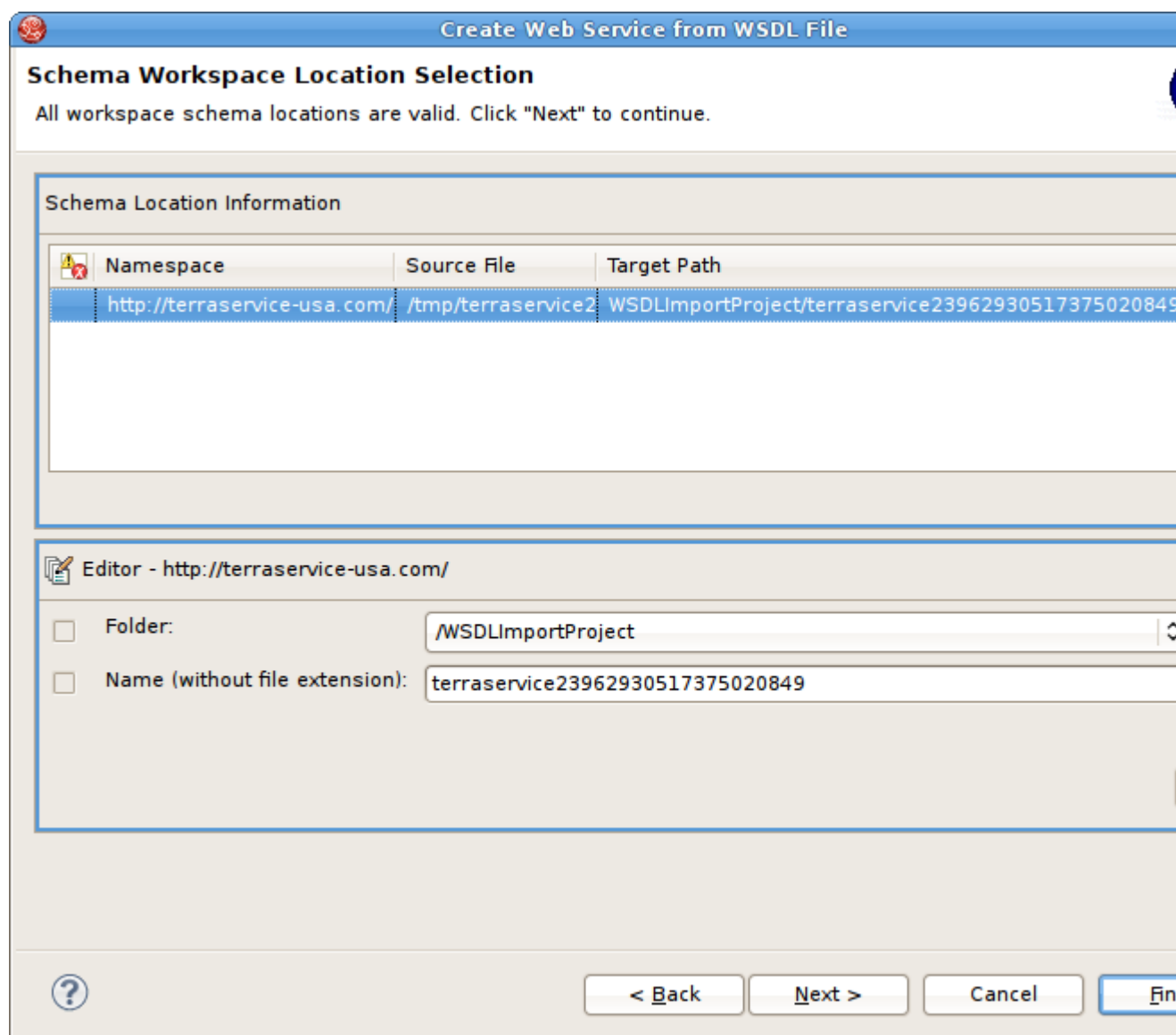


Figure 6.44. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, an error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

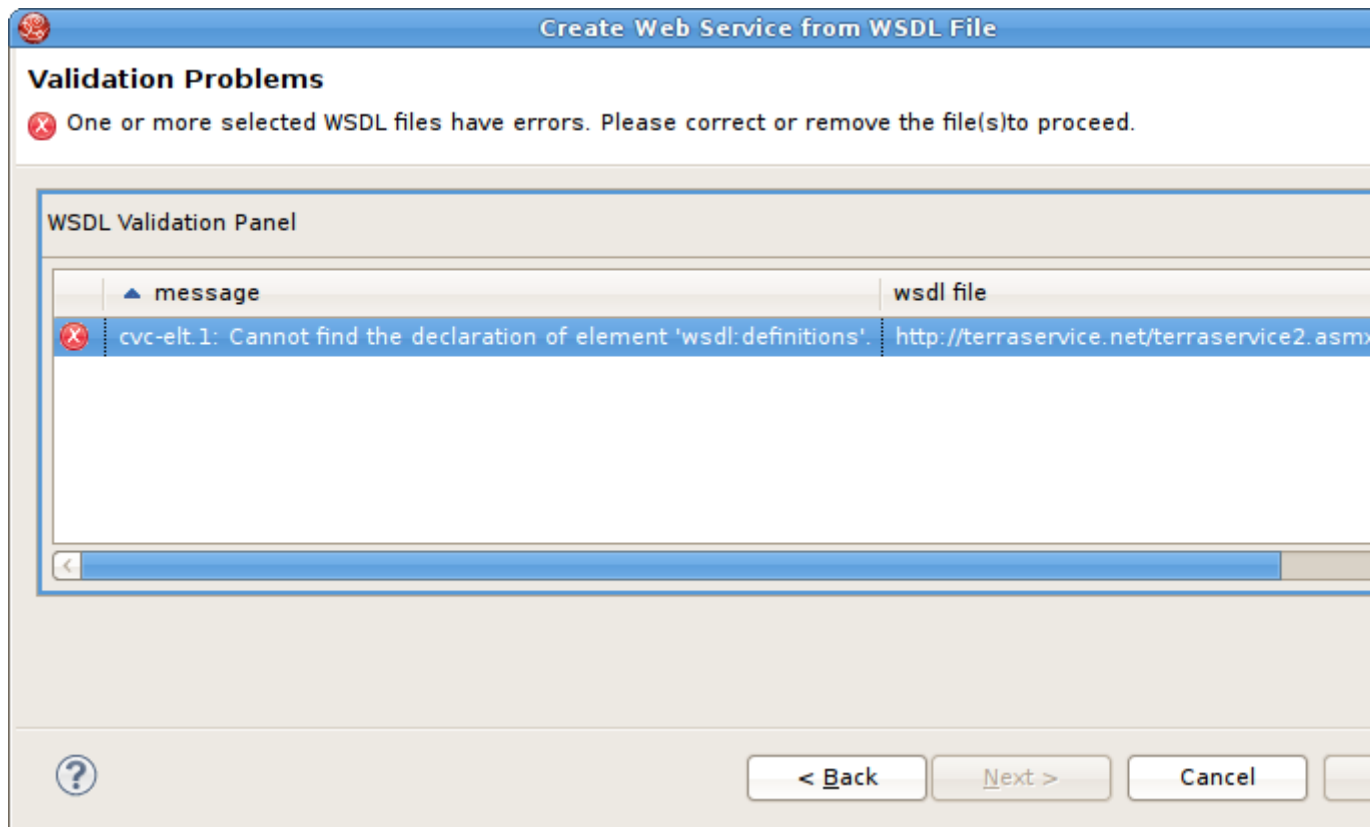



Figure 6.45. WSDL Validation Problems Dialog

6.6. XSD Schema File

- You can import **XML Schema** file (XSD) files using the steps below.
 - Step 1** - In **Model Explorer** choose the **File > Import** action  in the toolbar or select a project, folder or model in the tree and choose **Import...**
 - Step 2** - Select the import option **Metadata Modeling > XSD Schema files on file system** and click **Next>**

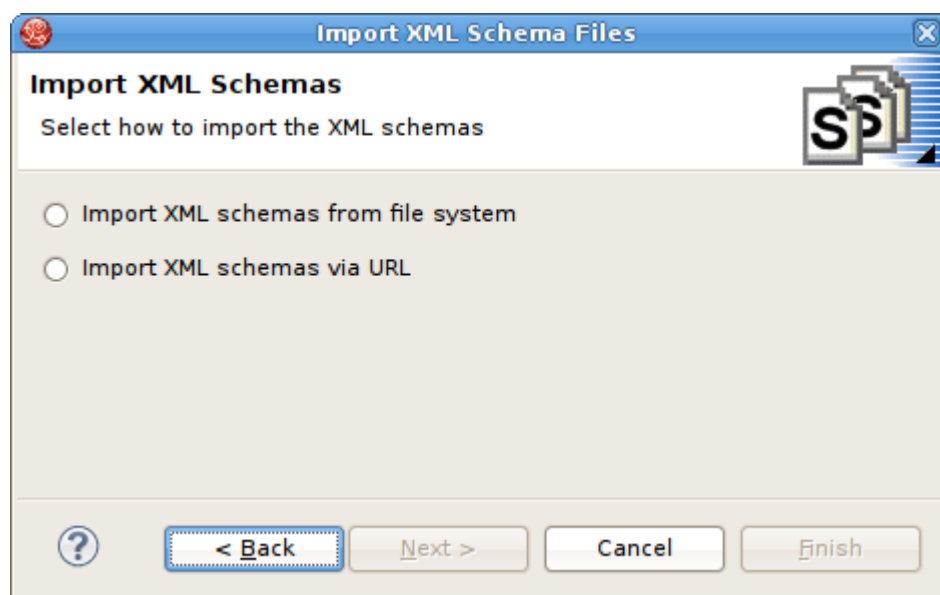


Figure 6.46. Import XSD Schemas Dialog

- **Step 3** - Select either **Import XSD Schemas from file system** or **Import XSD Schemas via URL** and click **Next >**
- **Step 4a** - If importing from file system, the **Import XSD Files** dialog is displayed. Click on the **Browse** button to find the directory that contains the XSD file(s) you wish to import.
 - To select all of the XSD files in the directory, click the checkbox next to the folder in the left panel.
 - To select individual XSD files, click the checkboxes next to the files you want in the right panel



Figure 6.47. Select XSD From File System

- **Step 4b** - If importing from URL, select the *Import XML Schemas via URL* option and click *OK* to display the final *Add XML Schema URLs* wizard page.

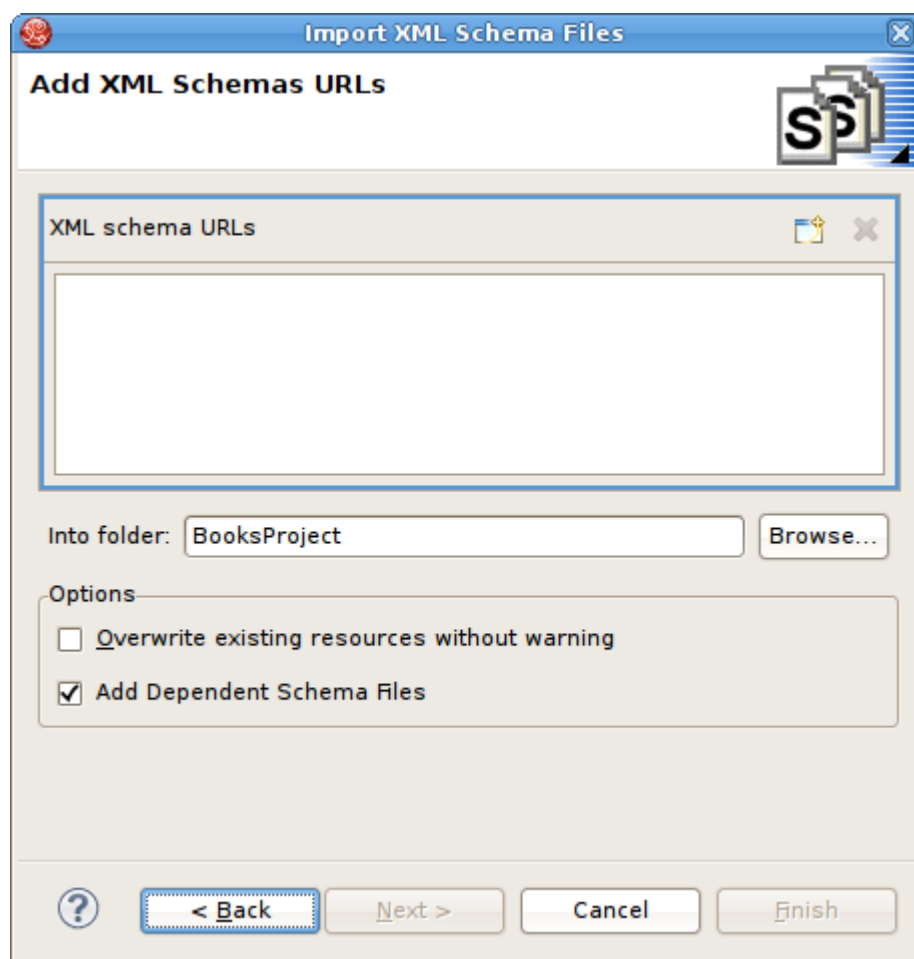
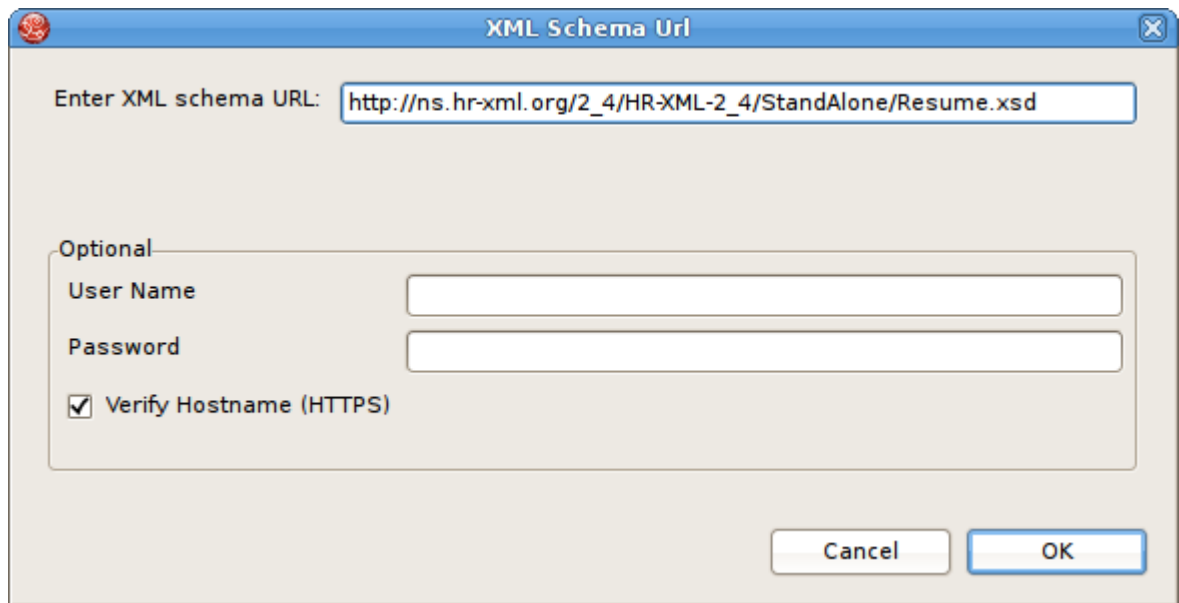


Figure 6.48. Add XML Schema URLs Dialog

- **Step 5** - Click the Add XML Schema URL button



Enter a valid schema URL. Click OK. Schema will be validated and resulting entry added to the list of XML Schema URLs.



The image shows a dialog box titled "XML Schema Url". It has a blue title bar with a close button in the top right corner. The main area is light gray. At the top, it says "Enter XML schema URL:" followed by a text box containing the URL "http://ns.hr-xml.org/2_4/HR-XML-2_4/StandAlone/Resume.xsd". Below this, there is a section labeled "Optional" in a smaller font. Inside this section, there are two text boxes: "User Name" and "Password". Below these, there is a checkbox labeled "Verify Hostname (HTTPS)" which is checked. At the bottom right of the dialog, there are two buttons: "Cancel" and "OK".

Figure 6.49. Add XSD Schema URLs

The schema URL is now displayed in the XML Schema URLs list.

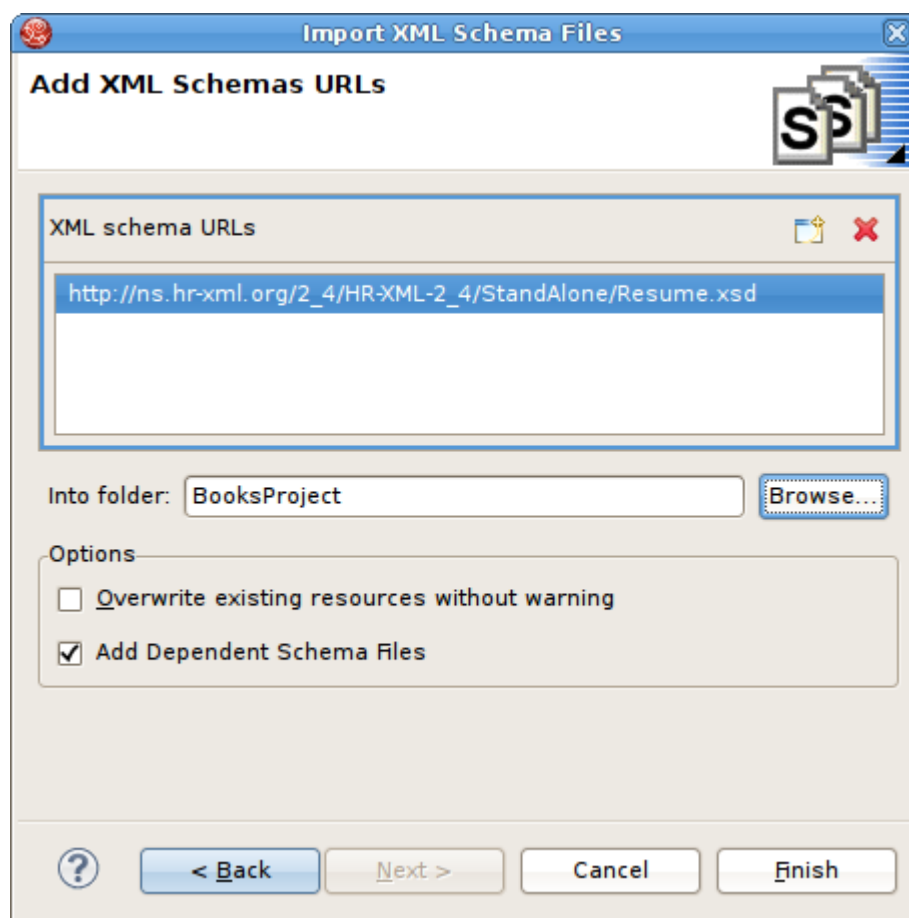


Figure 6.50. Add XSD Schema URLs

- **Step 6** - Click **Finish**.



Note

XSD files may have dependent files. This importer will determine these dependencies and import these as well if Add Dependent Schema Files is checked

New Model Wizards

Models are the primary resource used by the Teiid Designer. Creating models can be accomplished by either directly importing existing metadata or by creating them using one of several **New Model** wizard options. This section describes these wizards in detail.

- The Teiid Designer currently supports the following types of models:



Section 7.1, “Creating New Relational Source Model”



Section 7.2, “Creating New Relational View Model”



Section 7.3, “Creating XML Document View Model”



Section 7.4, “Creating XML Schema Model”



Section 7.5, “Creating Web Service View Model”



Section 7.6, “Creating New Extensions Model”

Use one of the following options to launch the New Model Wizard.

New Model Wizard

- Choose the **File** > **New...** > **Metadata Model** action



- Select a project or folder in the *Figure 4.2, “Model Explorer View”* and choose the same action in the right-click menu.

- Select the **New** button on the main toolbar



and select the **Metadata Model** action



Note

Model names are required to be unique within Designer. When specifying model names in new model wizards and dialogues error messages will be presented and you will be prevented from entering an existing name.

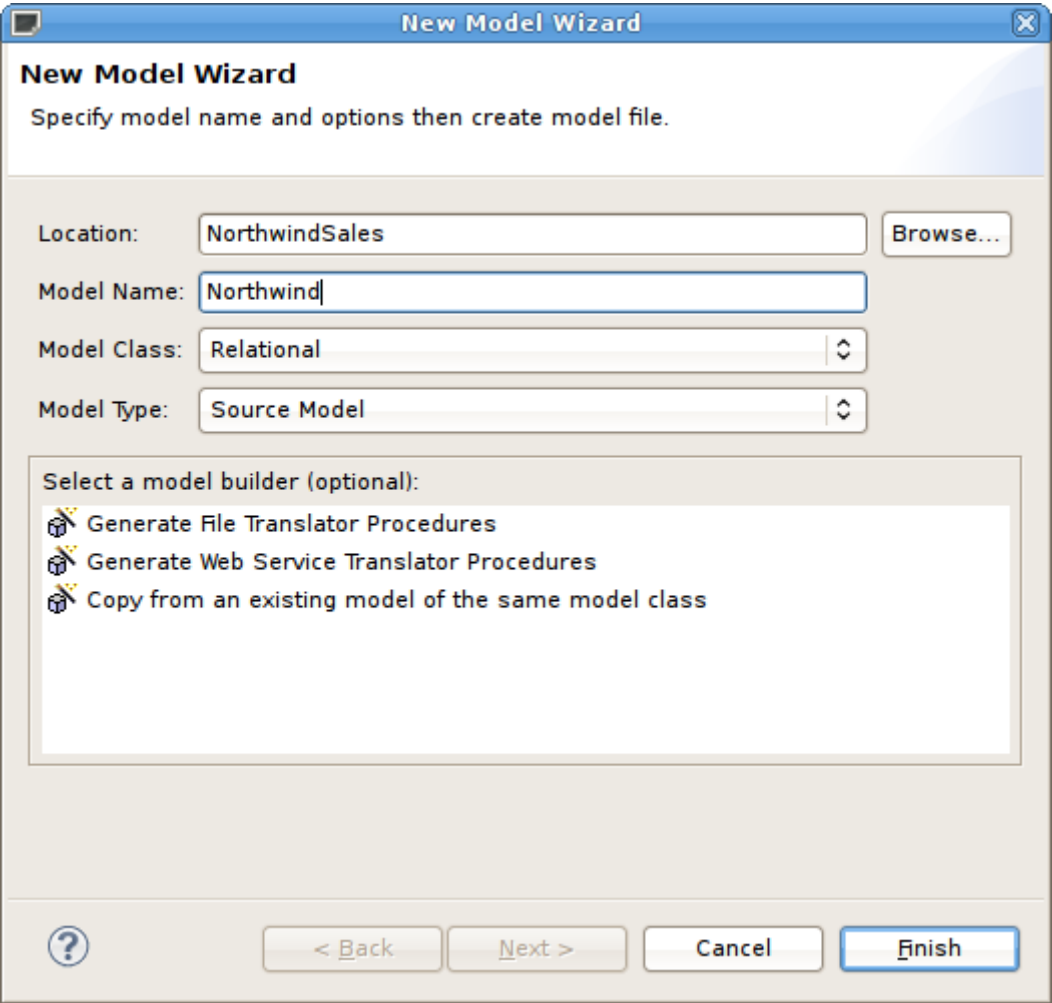


Figure 7.1. Import Wizard Selection Dialog

7.1. Creating New Relational Source Model

Create New Relational Source Model

- To create a new empty relational source model:

- **Step 1** - [New Model Wizard](#)
- **Step 2** - Specify a unique model name.
- **Step 3** - Select **Relational** option from **Model Class** drop-down menu.
- **Step 4** - Select **Source Model** from **Model Type** drop-down menu.
- **Step 5** - Click **Finish**.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty relational source model, the following builder options are available:
 - Copy from existing model of the same model class.

7.1.1. Generate File Translator Procedures

This builder option allows construction of a relational model containing one or more of the procedures required for accessing file-based data via a file translator.

- To create a new relational model containing file translator procedures, complete [Create New Relational Source Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Generate File Translator Procedures** and click **Next >**. The **Generate File Translator Procedures** dialog will be displayed.
 - **Step 6** - Check one or more of the **Available File Translator Procedures** Click **Finish**

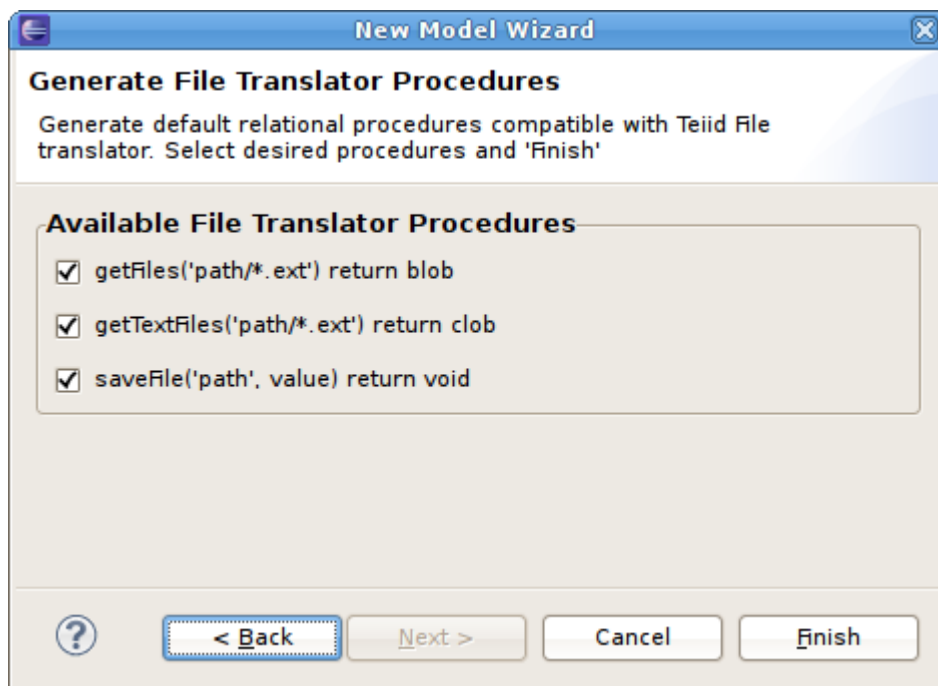


Figure 7.2. Generate File Translator Procedures Dialog

7.1.2. Generate Web Service Translator Procedures

This builder option allows construction of a relational model containing one or more of the procedures required for accessing web-service-based XML data via a web s translator.

- To create a new relational model containing file translator procedures, complete [Create New Relational Source Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Generate Web Service Translator Procedures** and click **Next >**. The **Generate File Translator Procedures** dialog will be displayed.
 - **Step 6** - Check one ore more of the **Available Web Services Translator Procedures** Click **Finish**

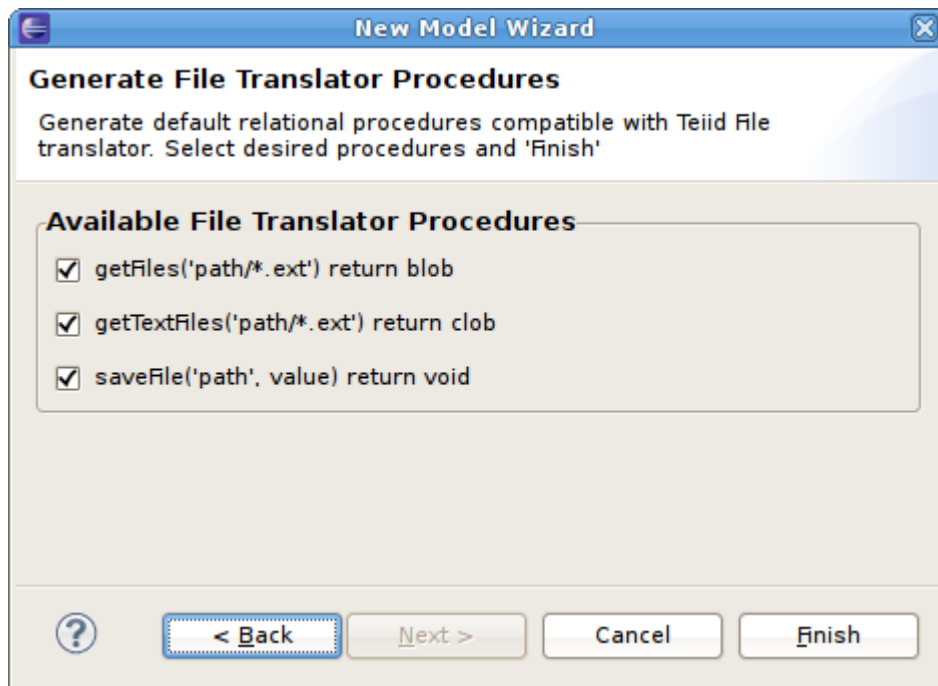


Figure 7.3. Generate Web Service Translator Procedures Dialog

7.1.3. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another relational source model, complete [Create New Relational Source Model](#) above and continue with these additional steps:
- **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.
- **Step 6** - Select an existing relational model from the workspace using the browse button.



Note

An existing model will be pre-selected if a relational model in the workspace is selected in the [Figure 4.2, "Model Explorer View"](#) prior to starting the new model wizard.

- **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

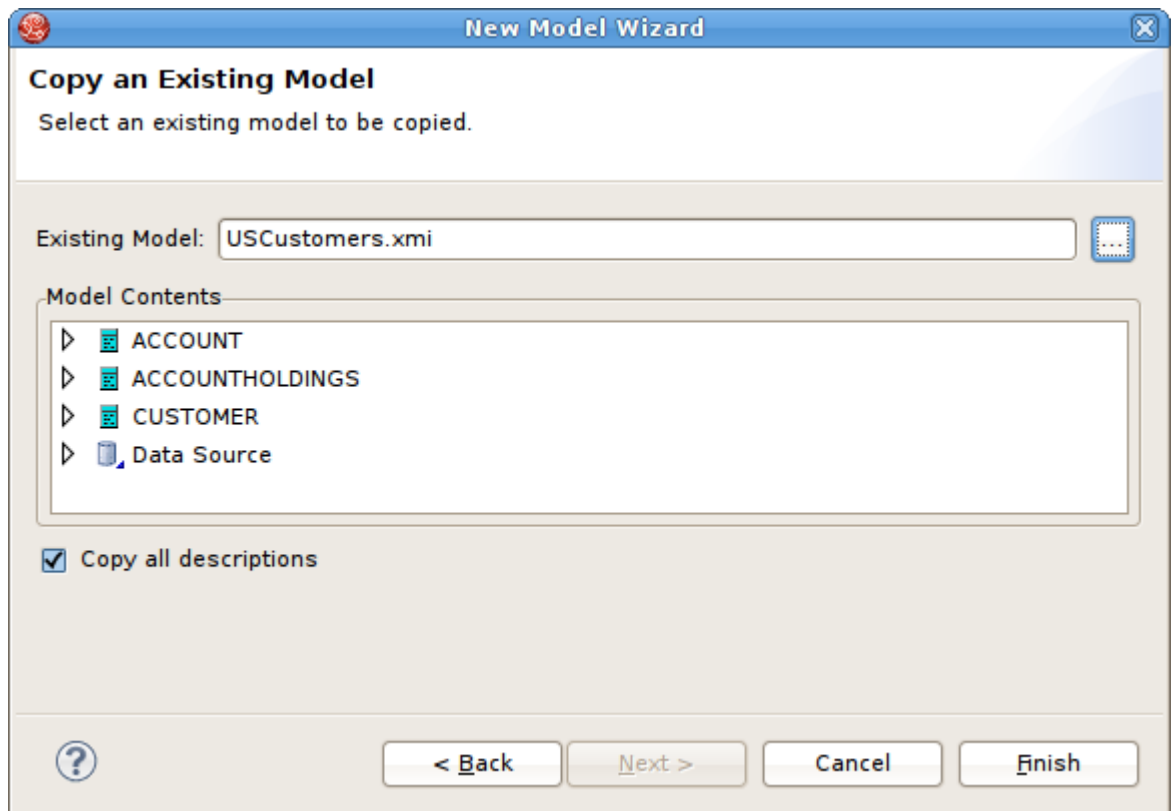


Figure 7.4. Copy An Existing Model Dialog

7.2. Creating New Relational View Model

Create New Relational View Model

- To create a new empty relational view model:
 - **Step 1** - [New Model Wizard](#)
 - **Step 2** - Specify a unique model name.
 - **Step 3** - Select **Relational** option from **Model Class** drop-down menu.
 - **Step 4** - Select **View Model** from **Model Type** drop-down menu.
 - **Step 5** - Click **Finish**.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty relational view model, the following builder options are available:
 - Copy from existing model of the same model class.
 - Transform from existing model.

7.2.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another relational view model, complete [Create New Relational View Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.
 - **Step 6** - Select an existing relational model from the workspace using the browse button.



- **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

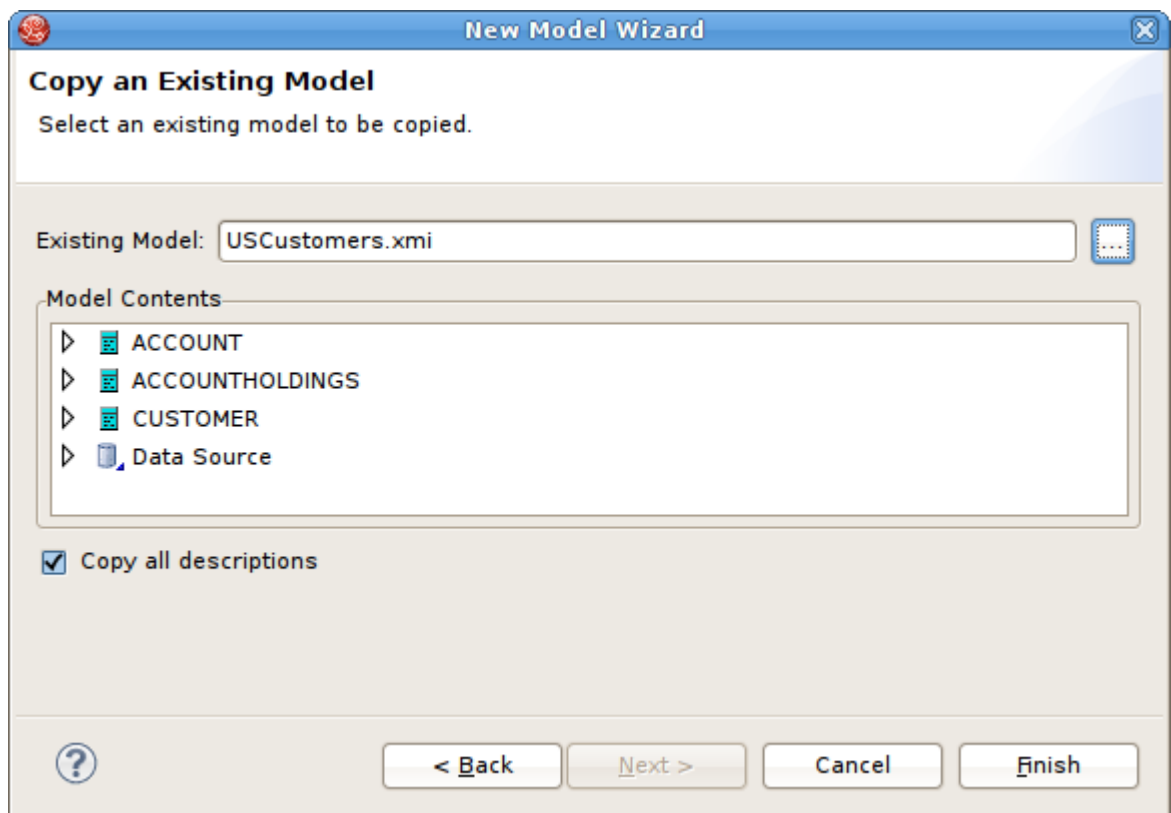


Figure 7.5. Copy An Existing Model Dialog

7.2.2. Transform From Existing Model

This option is only applicable for creating a relational view model from a relational source model with the added feature of creating default transformations (SELECT * FROM SourceModel.Table_X) for each source table. The steps are the same as for the [Section 7.2.1, “Copy From Existing Model”](#) described above.

There is an additional option in the second page of the wizard which can automatically set the relational table's supports update property to false. If this is unchecked the default value will be true.

7.2.3. Create From XML Schema

In the Teiid Designer you can use your existing schema models to generate relational view tables.

- To create a new relational view model from XML schema:
 - **Step 1** - Select any **XML Schema model (XSD)** in in the [Figure 4.2, “Model Explorer View”](#) tree.
 - **Step 2** - Right-click to display the context menu and select the **Modeling > Create Relational View Model from Schema** action.
 - **Step 3** - In the **Create Virtual Tables from XSD Schema Wizard** dialog, specify a relational view model name.

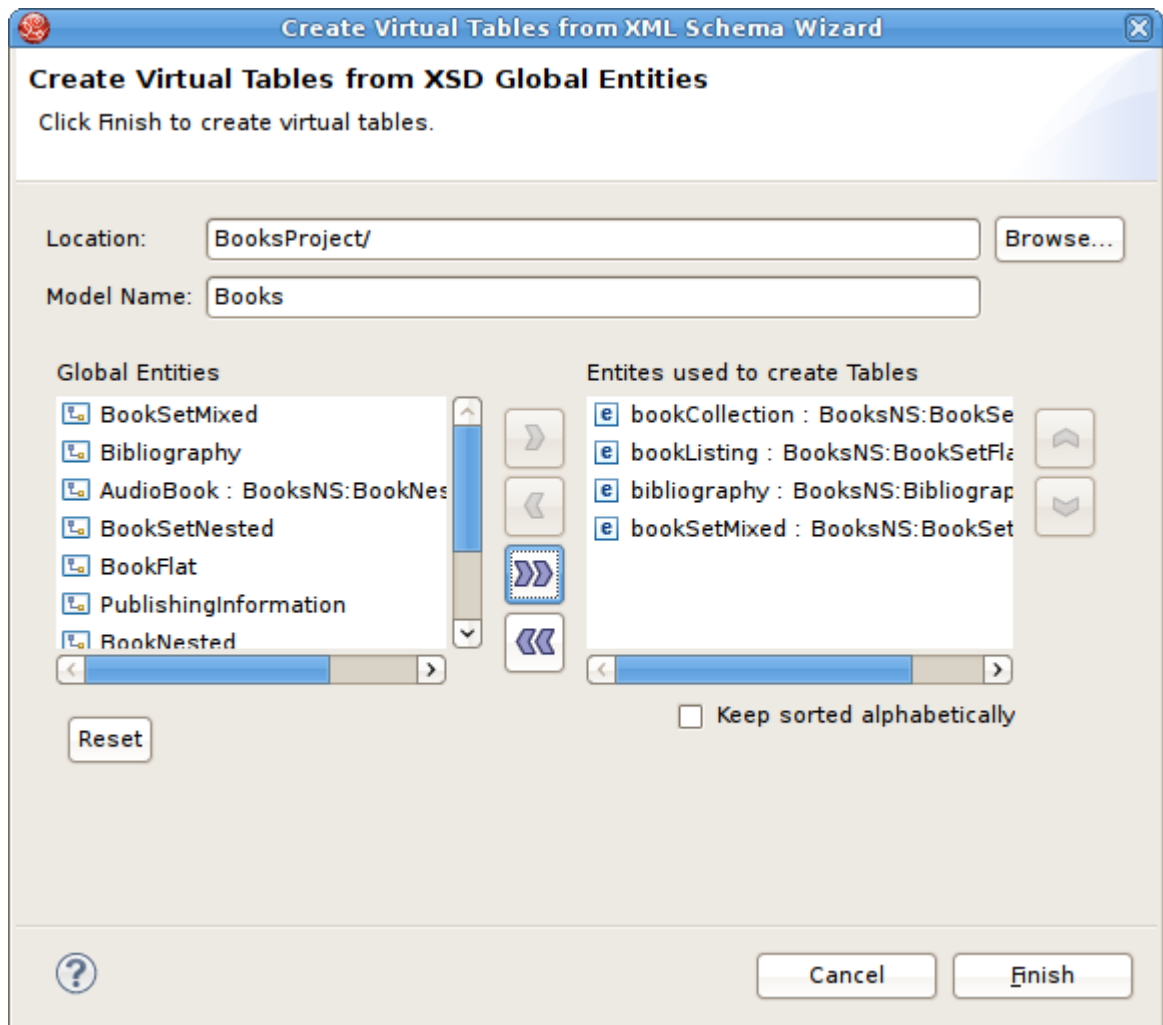


Figure 7.6. Create Virtual Tables From XML Schema Dialog

- **Step 4** - Select one or more global schema entities and move from left panel to the right panel. Click **Finish** when done.

When model generation is complete the new model will be opened in an editor for your viewing.

7.3. Creating XML Document View Model

Create XML Document View Model

- To create a new empty xml document view model:
 - **Step 1** - [New Model Wizard](#)
 - **Step 2** - Specify a unique model name.

- **Step 3** - Select **XML** option from **Model Class** drop-down menu.
- **Step 4** - Select **View Model** from **Model Type** drop-down menu.
- **Step 5** - Click **Finish**.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty xml document view model, the following builder options are available:
 - Copy from existing model of the same model class.
 - Build XML documents from XML schema.

7.3.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another xml document view model, complete [Create XML Document View Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.
 - **Step 6** - Select an existing relational model from the workspace using the browse button.



- **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

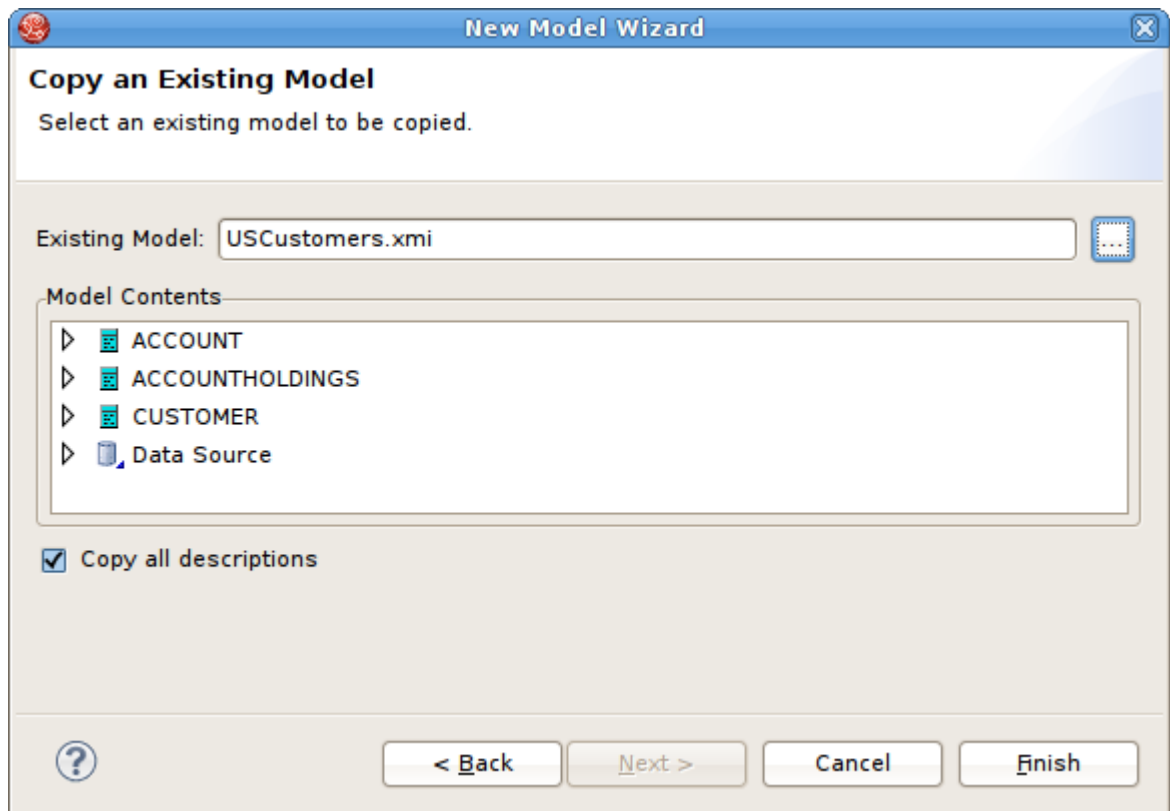


Figure 7.7. Copy An Existing Model Dialog

7.3.2. Build XML Documents From XML Schema

This option creates an XML View document model based on a selected XML schema and its dependencies.

- To create a new xml document view model by from XML schema, complete [Create XML Document View Model](#) above and continue with these additional steps:
- **Step 5** - Select the model builder labeled **Build XML documents from XML schema** and click **Next >**. The **Select XML Schema** dialog will be displayed.
- **Step 6** - Select an existing schema model from the workspace using the browse button.



Note

An existing model will be pre-selected if an XSD model in the workspace is selected in the VDB explorer prior to starting the new model wizard. The

schema must be found in the workspace so if you need to get one or more into the workspace use the XSD Schemas on file system importer.

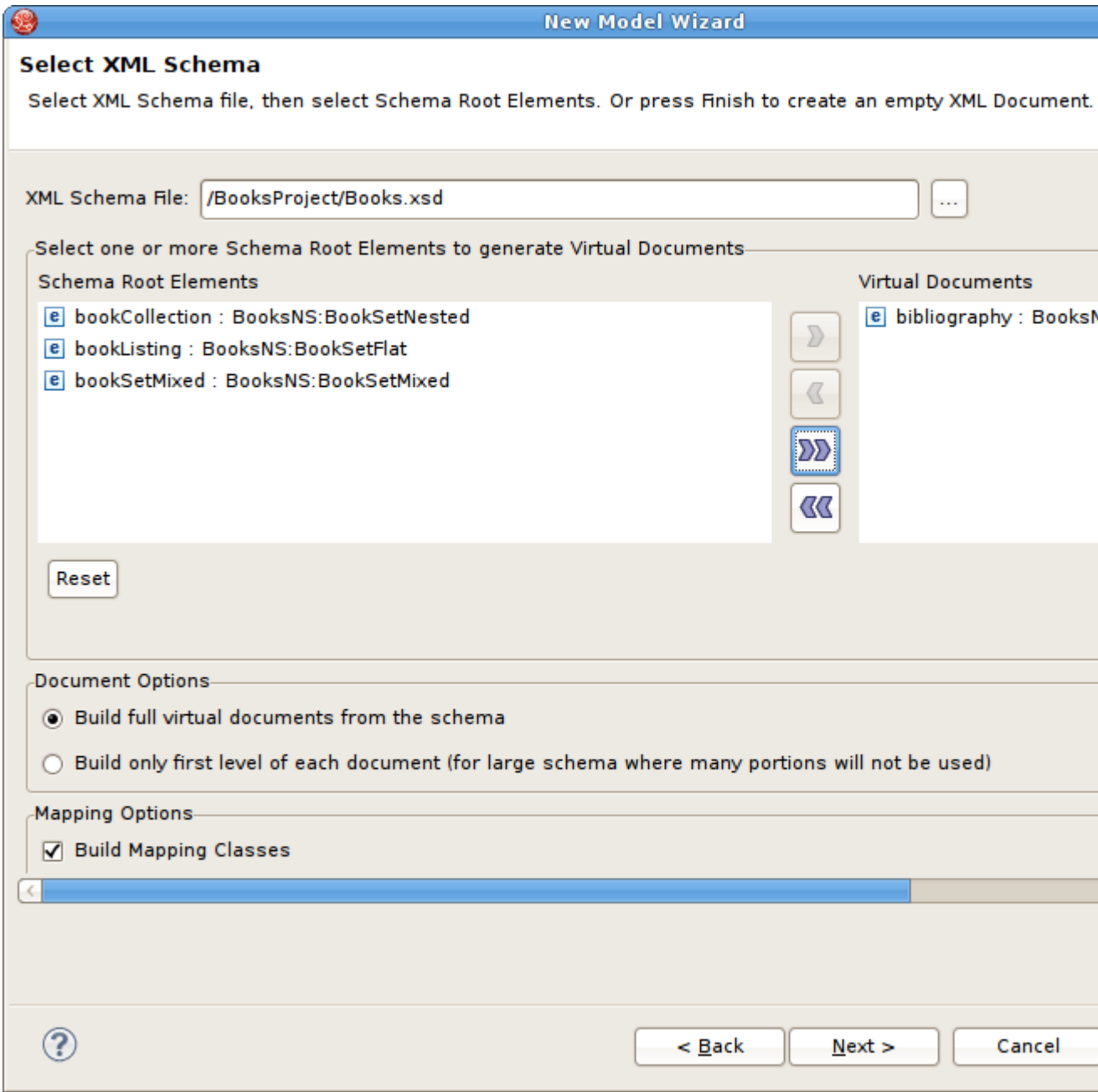


Figure 7.8. Select XML Schema Dialog

- **Step 7** - Move the available schema root elements you want to become virtual documents in the new model over to the **Virtual Documents** list by using the arrow button



for selected elements or the



button to move all elements.

- **Step 8** - Select the appropriate document options and mapping options. Click **Finish**
- **Step 9** - Click **Finish** to create a model of all selected document entities or (optional) click **Next >** to view **Selected Documents Statistics** page which shows document entity statistics and gives you an idea the size of the model being created.

New Model Wizard

Selected Documents Statistics

This is an overview of the documents to be generated. Select Next to preview and edit the document, or Finish to create the document with default settings.

Documents: 1

Elements: 7

Recursive Elements: 0

Complex Subtype Elements: 0

Attributes: 0

Total entity count: 13

? < Back Next > Cancel Finish

Figure 7.9. Selected Documents Statistics Dialog

- **Step 10** - (Optional) Click **Finish** to create a model of all selected document entities or click **Next >** to view **Preview Generated Documents** page that allows you to exclude document specific entities then click **Finish**.



Note

For deeply nested schema, your total entity count may be large. If so, displaying the preview may take some time.

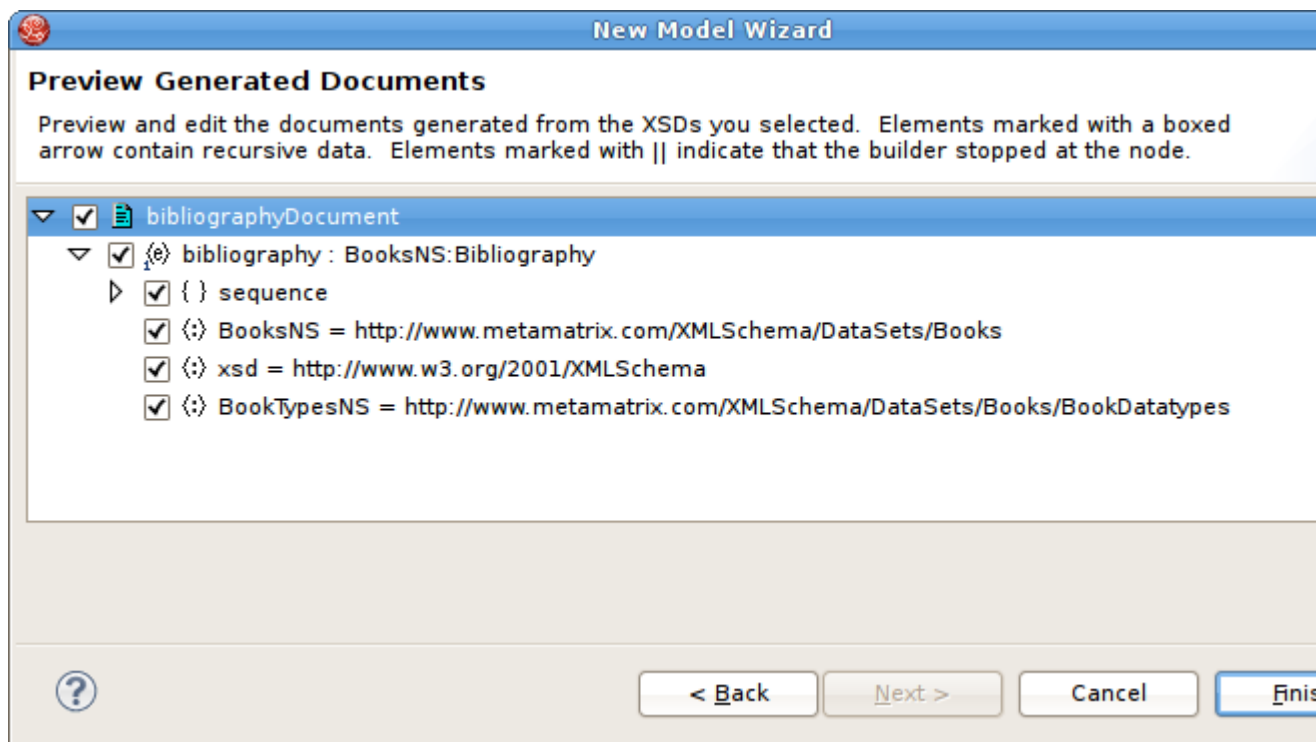


Figure 7.10. Preview Generated Documents Dialog

7.4. Creating XML Schema Model

Create XML Schema Model

- To create a new empty xml schema (.xsd) model:
 - **Step 1** - [New Model Wizard](#)
 - **Step 2** - Specify a unique model name.
 - **Step 3** - Select **XML Schema (XSD)** option from **Model Class** drop-down menu.
 - **Step 4** - Select **Datatype Model** from **Model Type** drop-down menu.
 - **Step 5** - Click **Finish**.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty xml schema model, the following builder option is available:

- Copy from existing model of the same model class.

7.4.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another xml schema model, complete [Create XML Schema Model](#) above and continue with these additional steps:

- **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.

- **Step 6** - Select an existing relational model from the workspace using the browse button.



- **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

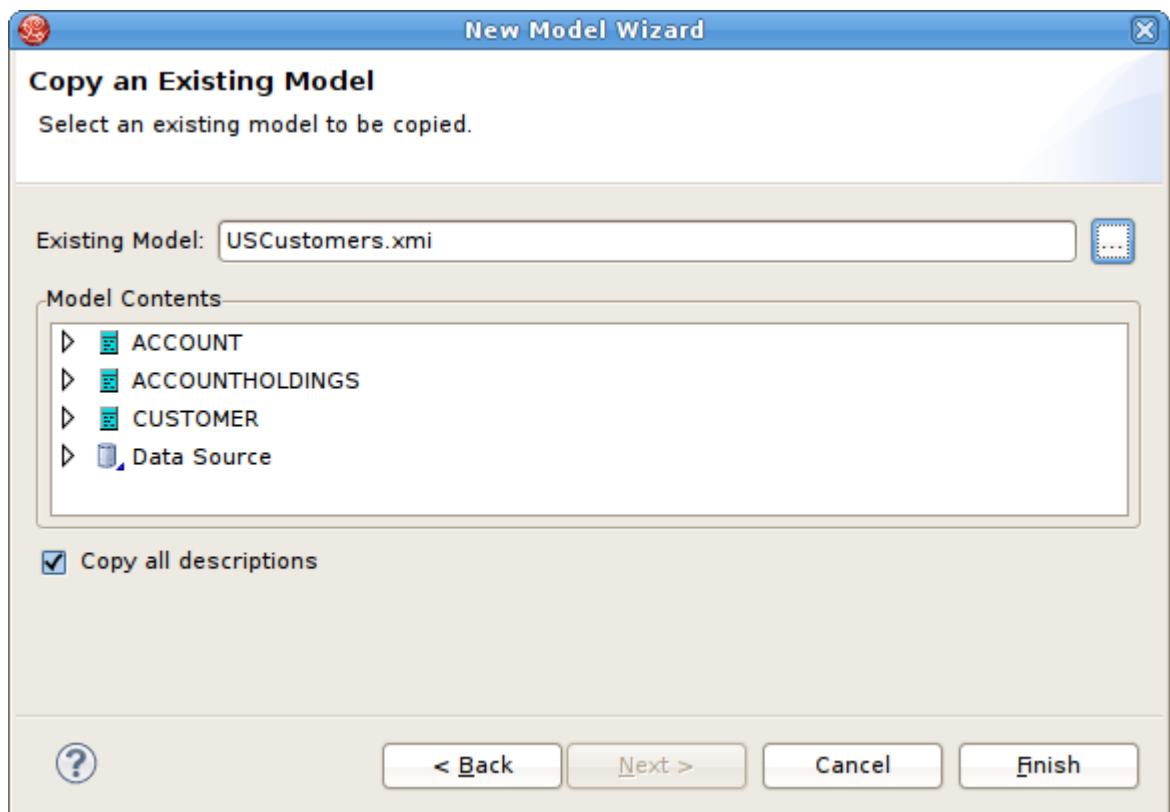


Figure 7.11. Copy An Existing Model Dialog

7.5. Creating Web Service View Model

Create Web Service View Model

- To create a new empty web service view model:
 - **Step 1** - [New Model Wizard](#)
 - **Step 2** - Specify a unique model name.
 - **Step 3** - Select **Web Service** option from **Model Class** drop-down menu.
 - **Step 4** - Select **View Model** from **Model Type** drop-down menu.
 - **Step 5** - Click **Finish**.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty web service view model, the following builder options are available:
 - Copy from existing model of the same model class.
 - Build from existing WSDL file(s) or URL.

7.5.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another web service view model, complete [Create Web Service View Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.
 - **Step 6** - Select an existing relational model from the workspace using the browse button.



- **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

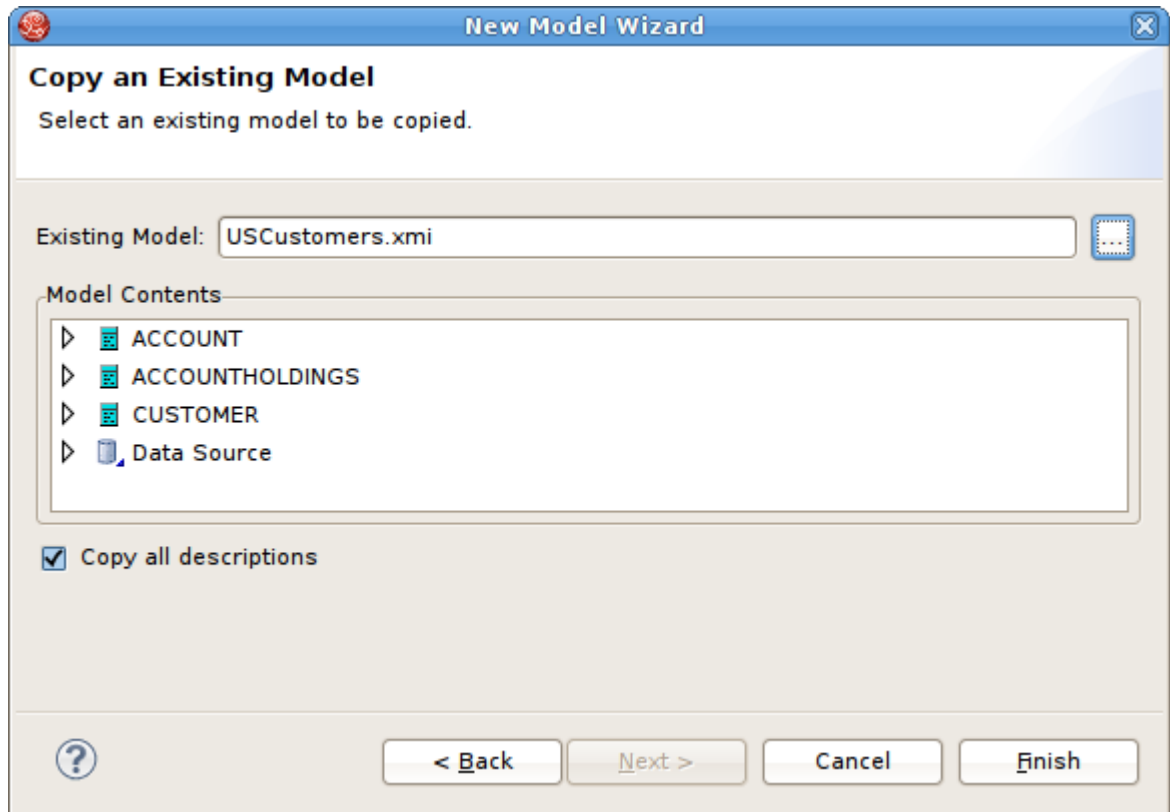


Figure 7.12. Copy An Existing Model Dialog

7.5.2. Build From Existing WSDL File(s) or URL

This builder option creates a Web service model based on a user-defined WSDL file and its referenced schemas. In addition, applicable XML schema files and XML View document models (optional) are created.

- To create a new relational model by copying contents from another web service view model, complete [Create Web Service View Model](#) above and continue with these additional steps:
- **Step 5** - Select the model builder labeled **Build from existing WSDL file(s) or URL** and click **Next >**.
- The remaining wizard steps are identical to those found using the [Section 6.5, "Import WSDL Into Web Service"](#) action option.

7.5.3. Build From Relational Models

This method is recommended for experienced users for consistent and rapid deployment of **Web** services designed to query relational sources. It provides detailed control of all **Web** service interfaces, operations and required transformations from **XML Views**

- To create a Web service model from relational models or objects:

- **Step 1** - Select any combination of relational models, tables and/or procedures in the [Figure 4.2, “Model Explorer View”](#) tree..



Note

It is recommended that the user selects single source models, which enables auto-naming of input/output schema and Web service models in \ **Step 3**.

- **Step 2** - Right-click select **Modeling** > **Create Web Service** action

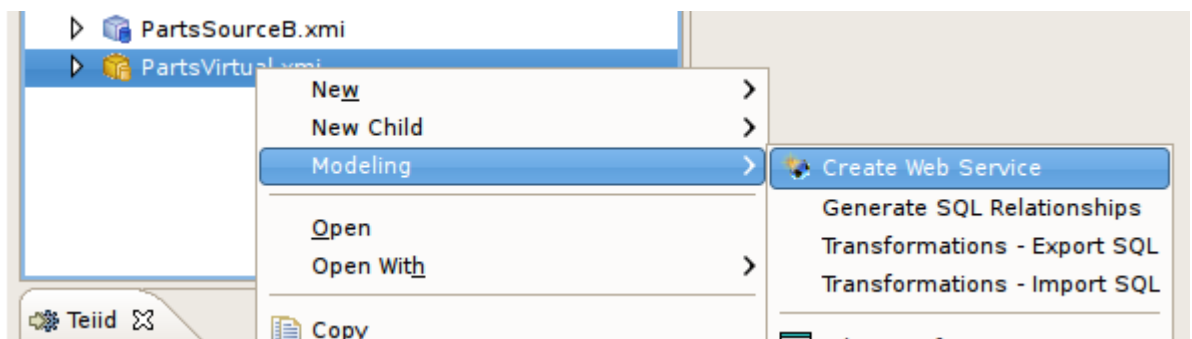
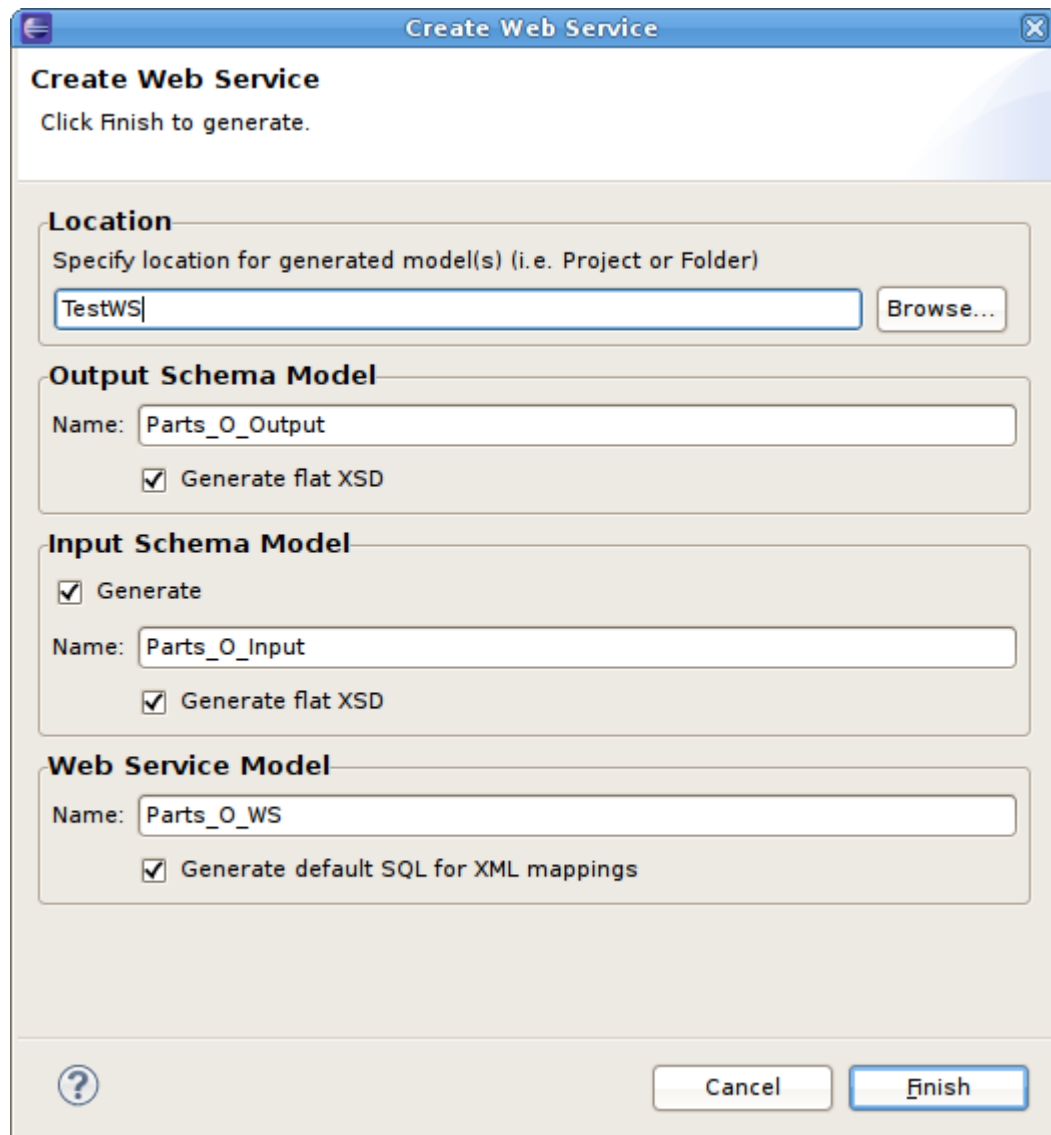


Figure 7.13. Create Web Service Action

- **Step 3** - In the **Create Web Service** dialog, specify file names for the generated **Input Schema** file, **Output Schema** file and **Web service** model. Change options as desired. Click **Finish** when done.



The image shows a 'Create Web Service' dialog box with a blue title bar and a close button. The main area is divided into four sections: 'Location', 'Output Schema Model', 'Input Schema Model', and 'Web Service Model'. The 'Location' section has a text field with 'TestWS' and a 'Browse...' button. The 'Output Schema Model' section has a 'Name' field with 'Parts_O_Output' and a checked 'Generate flat XSD' checkbox. The 'Input Schema Model' section has a checked 'Generate' checkbox, a 'Name' field with 'Parts_O_Input', and a checked 'Generate flat XSD' checkbox. The 'Web Service Model' section has a 'Name' field with 'Parts_O_WS' and a checked 'Generate default SQL for XML mappings' checkbox. At the bottom, there is a help icon (question mark in a circle), a 'Cancel' button, and a 'Finish' button.

Create Web Service

Click Finish to generate.

Location

Specify location for generated model(s) (i.e. Project or Folder)

TestWS Browse...

Output Schema Model

Name: Parts_O_Output

☒ Generate flat XSD

Input Schema Model

☒ Generate

Name: Parts_O_Input

☒ Generate flat XSD

Web Service Model

Name: Parts_O_WS

☒ Generate default SQL for XML mappings

? Cancel Finish

Figure 7.14. Create Web Service Dialog

- **Step 4** - When model generation is complete, a confirmation dialog should appear. Click **OK**.

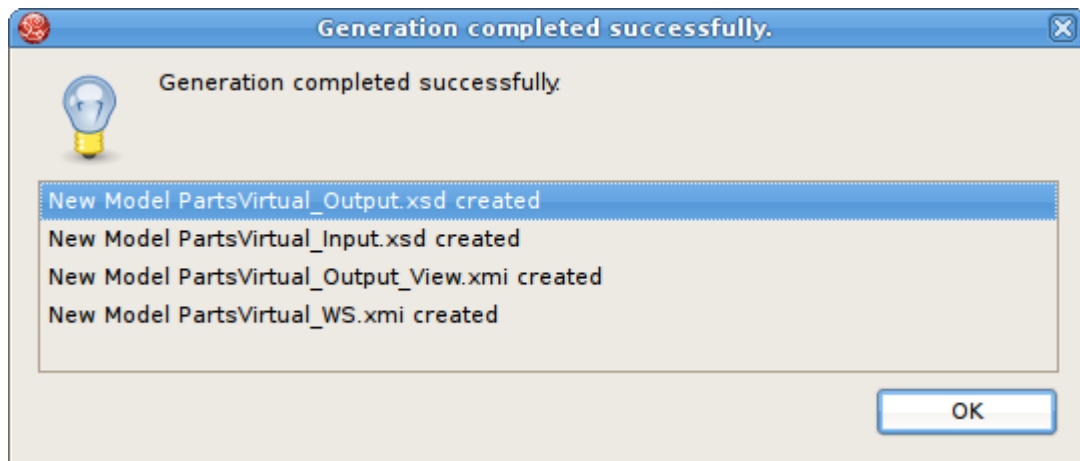

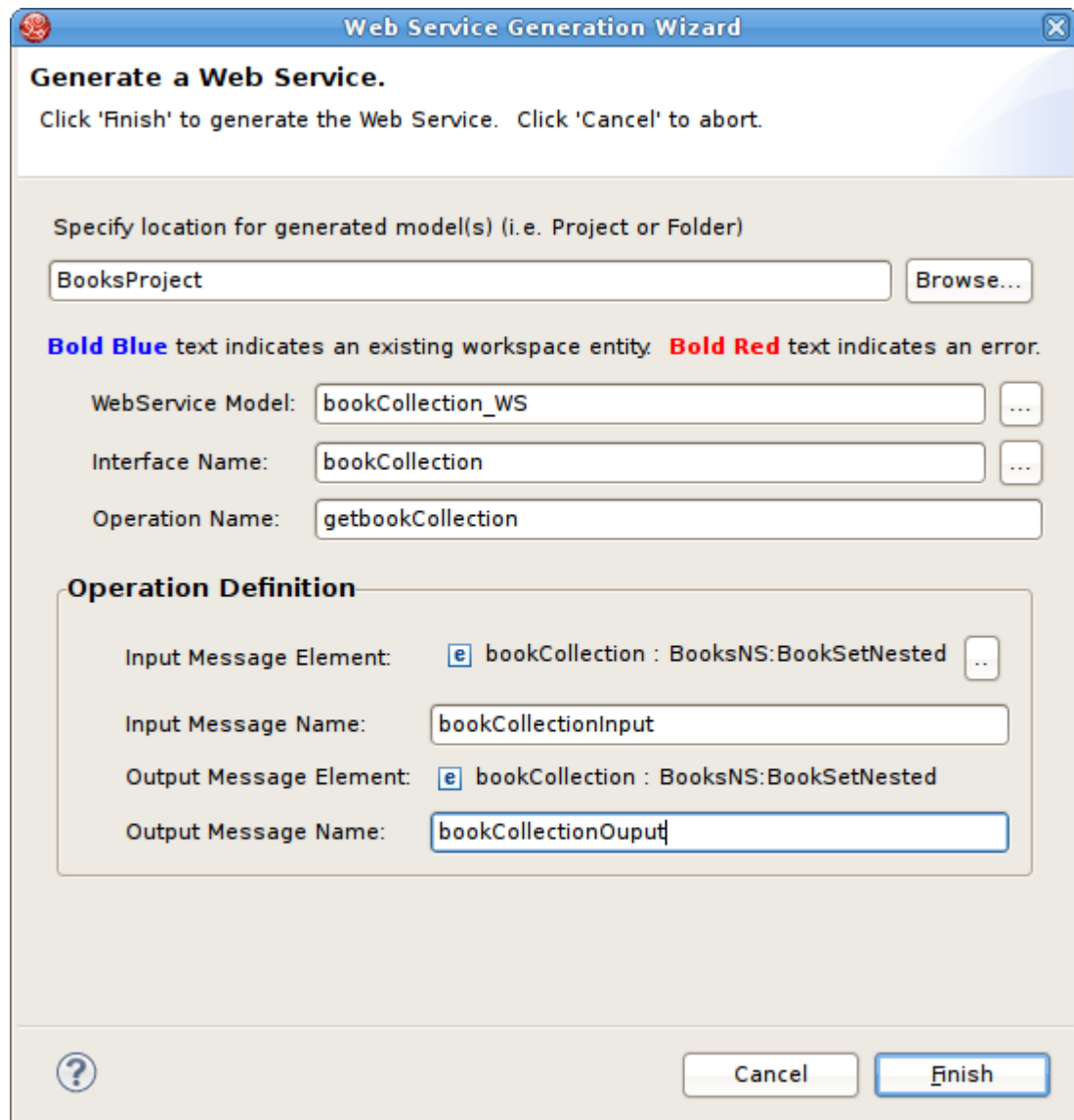


Figure 7.15. Generation Completed Dialog

7.5.4. Build From XML Document View Models

Web Service models and their corresponding **Interfaces** and **Operations** can be generated in Teiid Designer from **XML View** model components. Namely, XML View Documents and XML View Document roots.

- To create a new Web service model from XML components::
 - **Step 1** - Select either a single XML Document or single **XML Document root** in [Figure 4.2, "Model Explorer View"](#).
 - **Step 2** - Right-click select **Modeling > Create Web Service** action 
 - **Step 3** - Fill in missing properties in **Web Service Generation Wizard** shown below.



The image shows a 'Web Service Generation Wizard' dialog box. At the top, it says 'Generate a Web Service.' and 'Click 'Finish' to generate the Web Service. Click 'Cancel' to abort.' Below this, there's a section 'Specify location for generated model(s) (i.e. Project or Folder)' with a text field containing 'BooksProject' and a 'Browse...' button. A legend states: 'Bold Blue text indicates an existing workspace entity. Bold Red text indicates an error.' The main configuration section has three rows: 'WebService Model:' with 'bookCollection_WS' and an ellipsis button; 'Interface Name:' with 'bookCollection' and an ellipsis button; and 'Operation Name:' with 'getbookCollection'. Below this is an 'Operation Definition' section with four rows: 'Input Message Element:' with a blue 'e' icon and 'bookCollection : BooksNS:BookSetNested' and an ellipsis button; 'Input Message Name:' with 'bookCollectionInput'; 'Output Message Element:' with a blue 'e' icon and 'bookCollection : BooksNS:BookSetNested'; and 'Output Message Name:' with 'bookCollectionOutput'. At the bottom, there's a help icon (question mark in a circle), a 'Cancel' button, and a 'Finish' button.

Web Service Generation Wizard

Generate a Web Service.
Click 'Finish' to generate the Web Service. Click 'Cancel' to abort.

Specify location for generated model(s) (i.e. Project or Folder)

BooksProject

Bold Blue text indicates an existing workspace entity. **Bold Red** text indicates an error.

WebService Model: bookCollection_WS

Interface Name: bookCollection

Operation Name: getbookCollection

Operation Definition

Input Message Element: e bookCollection : BooksNS:BookSetNested

Input Message Name: bookCollectionInput

Output Message Element: e bookCollection : BooksNS:BookSetNested

Output Message Name: bookCollectionOutput

Figure 7.16. Generate A Web Service Dialog

- **Step 4** - Click **Finish** to generate model. When model generation is complete, a confirmation dialog should appear. Click **OK**.

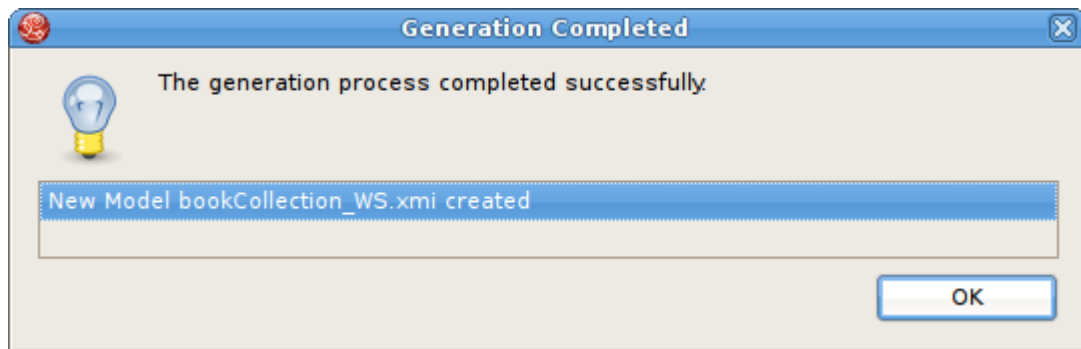


Figure 7.17. Generation Completed Dialog



Note

Users can change the **Web Service Model** and **Interface Name** values (via "...") buttons) to use existing **Web service** model components. This will create a new operation in an existing model.

7.6. Creating New Extensions Model

Create New Extensions Model

- To create a new empty extensions model:
 - **Step 1** - [New Model Wizard](#)
 - **Step 2** - Specify a unique model name.
 - **Step 3** - Select **Model Extensions** option from **Model Class** drop-down menu.
 - **Step 4** - Select **Model Class Extension** from **Model Type** drop-down menu.
 - **Step 5** - Click **Finish**.




Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty extensions model, the following builder option is available:
 - Copy from existing model of the same model class.

7.6.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another extensions model, complete [Create New Extensions Model](#) above and continue with these additional steps:
 - **Step 5** - Select the model builder labeled **Copy from existing model of the same model class** and click **Next >**. The **Copy Existing Model** dialog will be displayed.
 - **Step 6** - Select an existing relational model from the workspace using the browse button.

 - **Step 7** - Check the **Copy all descriptions** option if desired. Click **Finish**

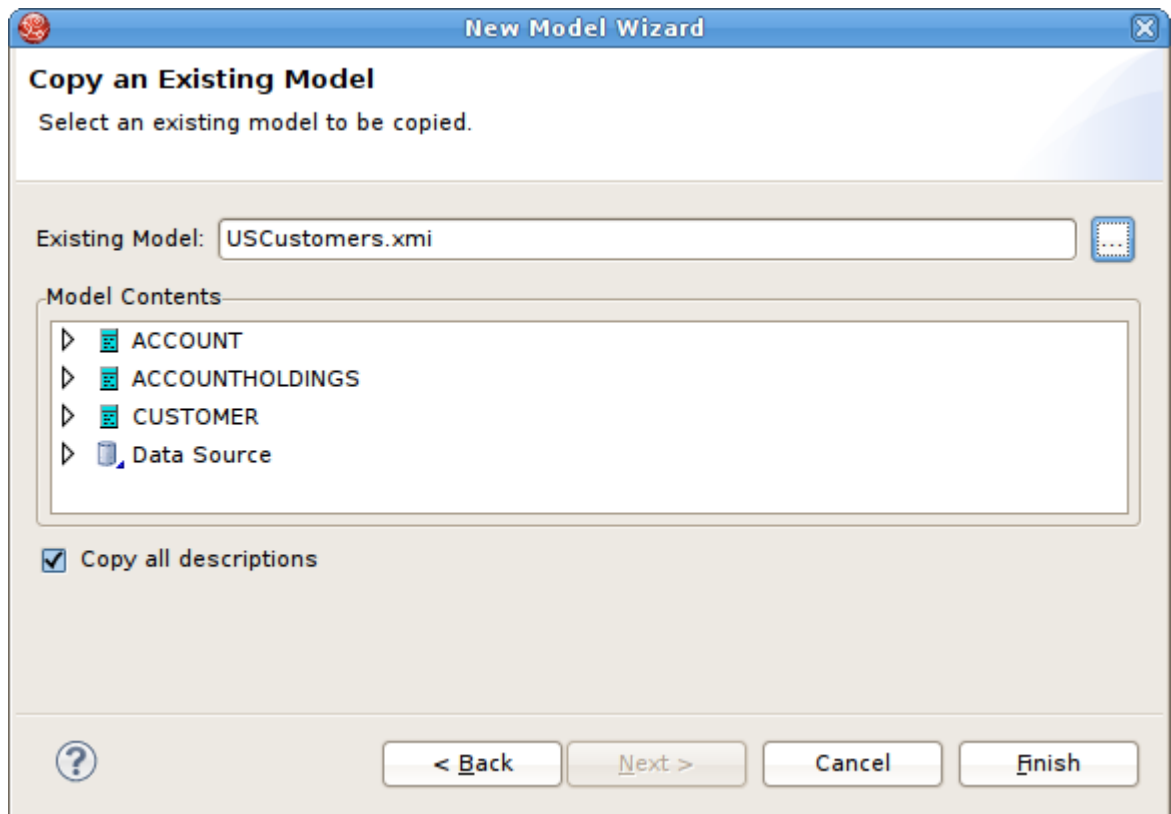


Figure 7.18. Copy An Existing Model Dialog

Creating and Editing Model Objects

This section summarizes Teiid Designer features for creating and editing existing model objects contained in your models.

8.1. Creating New Model Objects

As discussed in the introduction, [Section 1.3.3, “Metadata Models”](#) provide a framework to model various types of metadata. Each metamodel type has a set of parent-child relationships that establish constraints on what can be created and where. You cannot, for example, create a column attribute in a stored procedure, nor can you create a mapping class column in a Web service operation's output message.

The Teiid Designer provides a common set of actions to create new children of these models as well as children of children.

- You can create new model objects directly in the [Figure 4.2, “Model Explorer View”](#), [Section 5.1.1, “Diagram Editor”](#) or [Section 5.1.2, “Table Editor”](#) using the following actions:
 - **New Child Action**
 - **New Sibling Action**
 - **New Association Action**

8.1.1. New Child Action

- To create new child model objects in the [Figure 4.2, “Model Explorer View”](#):
 - **Step 1** - Select the parent object to which you want to add a child. For example, you can add a package to a package or an attribute to a class.
 - **Step 2** - Right-click on a container object. From the pop-up menu, select **New Child**. You can now select the child object you would like to add.

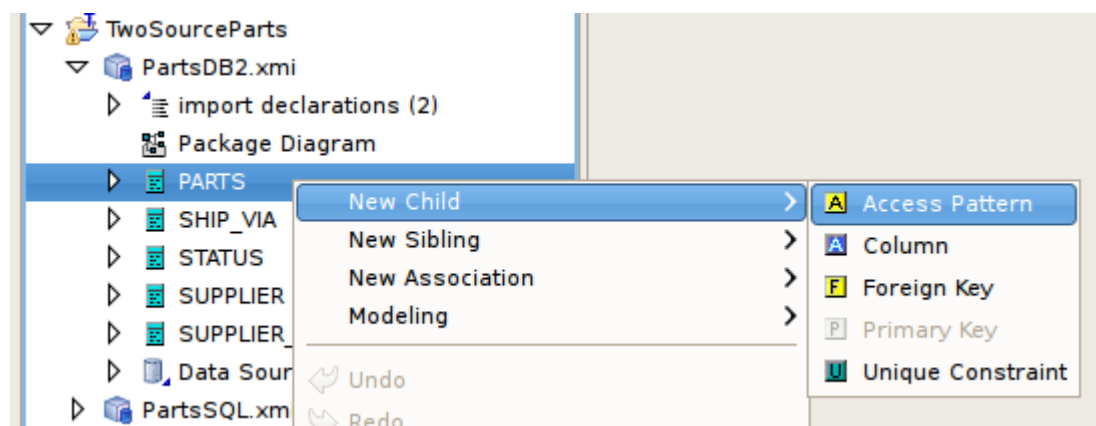


Figure 8.1. New Child Action In Model Explorer

- **Step 3** - The new model object displays on the [Figure 4.2, “Model Explorer View”](#) and is highlighted for renaming.

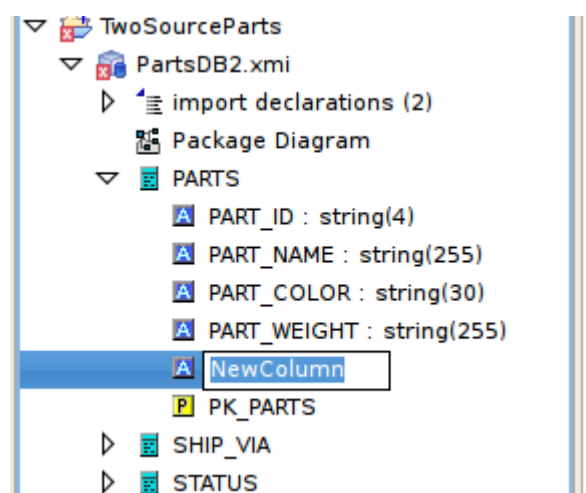


Figure 8.2. New Model Object In Explorer

- To create new child model objects in the [Section 5.1.1, “Diagram Editor”](#):
- **Step 1** - Select the parent object to which you want to add a child. For example, you can add a package to a package or an attribute to a class.
- **Step 2** - Right-click on a container object. From the pop-up menu, select **New Child**. You can now select the child object you would like to add.

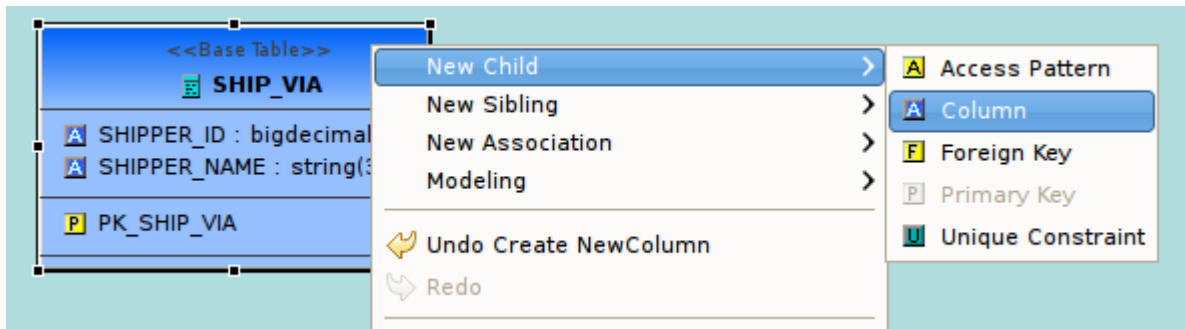


Figure 8.3. New Child Action In Diagram

- **Step 3** - The new model object displays on the diagram and is highlighted for renaming.

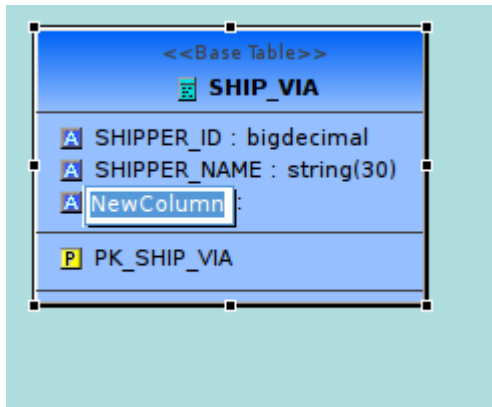


Figure 8.4. New Model Object In Diagram

- To create new child model objects in the [Section 5.1.2, "Table Editor"](#):
- **Step 1** - Select the row for the parent object to which you want to add a child. For example to add a column, click the **Base Table** tab and select base table row.
- **Step 2** - Right-click on a table row. From the pop-up menu, select **New Child**. You can now select the child object you would like to add.

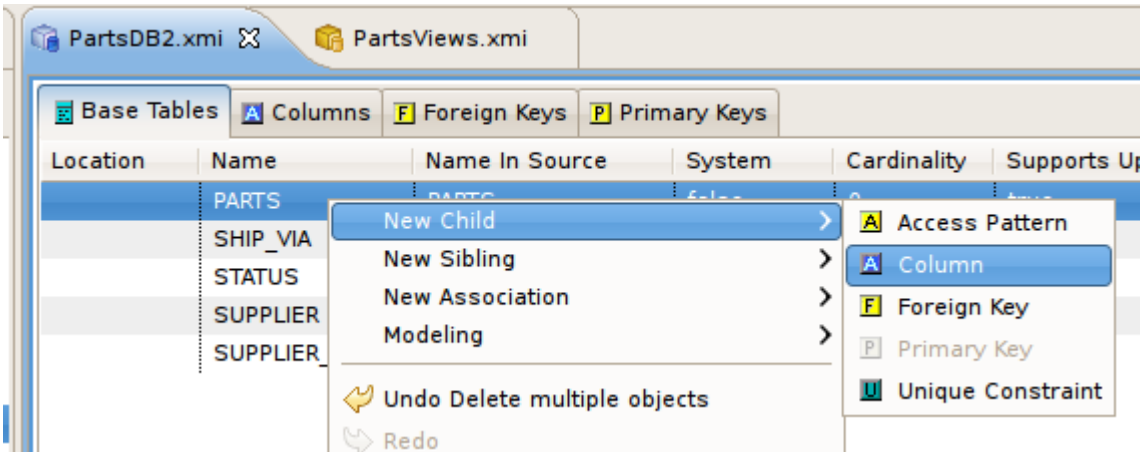


Figure 8.5. New Child Action In Table Editor

- **Step 3** - The selected tab in the **Table Editor** changes to the tab for the child object type, the new model object row is displayed and the row's name table cell is highlighted for renaming.

8.1.2. New Sibling Action

- To create new sibling model objects in the [Figure 4.2, "Model Explorer View"](#):
- **Step 1** - Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.
- **Step 2** - Right-click on that object. From the pop-up menu, select **New Sibling**. You can now select the sibling object you would like to add.

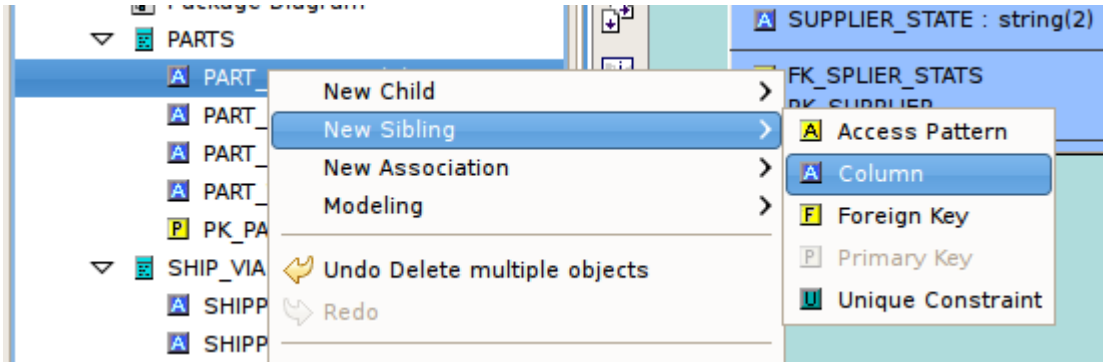


Figure 8.6. New Sibling Action In Model Explorer

- **Step 3** - The new model object displays on the [Figure 4.2, "Model Explorer View"](#) and is highlighted for renaming.
- To create new sibling model objects in the [Section 5.1.1, "Diagram Editor"](#):

- **Step 1** - Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.
- **Step 2** - Right-click on that object. From the pop-up menu, select **New Sibling**. You can now select the sibling object you would like to add.



Figure 8.7. New Sibling Action In Diagram

- **Step 3** - The new model object displays on the diagram and is highlighted for renaming.
- To create new sibling model objects in the [Section 5.1.2, "Table Editor"](#):
 - **Step 1** - Select the row for the object to which you want to add a sibling. For example, you can add a column sibling to a column.
 - **Step 2** - Right-click on a row. From the pop-up menu, select **New Sibling**. You can now select the sibling object you would like to add.

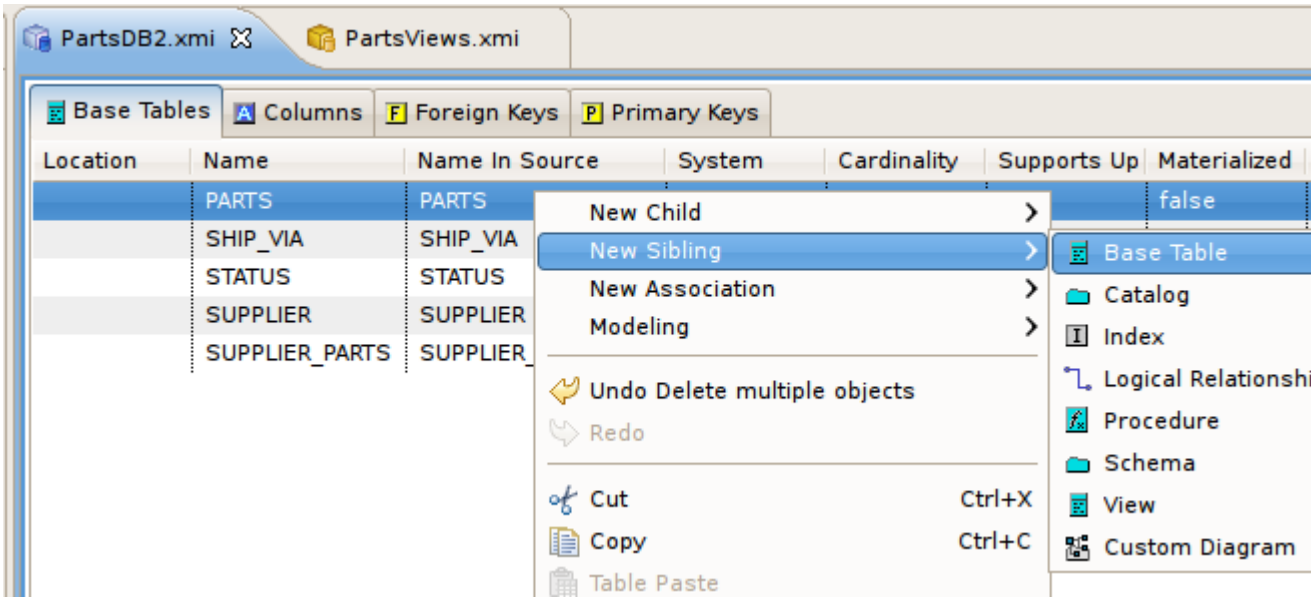


Figure 8.8. New Sibling Action In Table Editor

- **Step 3** - The selected tab in the **Table Editor** changes to the tab for the child object type, the new model object row is displayed and the row's name table cell is highlighted for renaming.

8.1.3. New Association Action

- To create new associations between model objects in the [Figure 4.2, "Model Explorer View"](#):
- **Step 1** - Select two objects you wish to associate. For example, select columns in different base tables.
- **Step 2** - Right-click. From the pop-up menu, select **New Association > Foreign Key Relationship**.

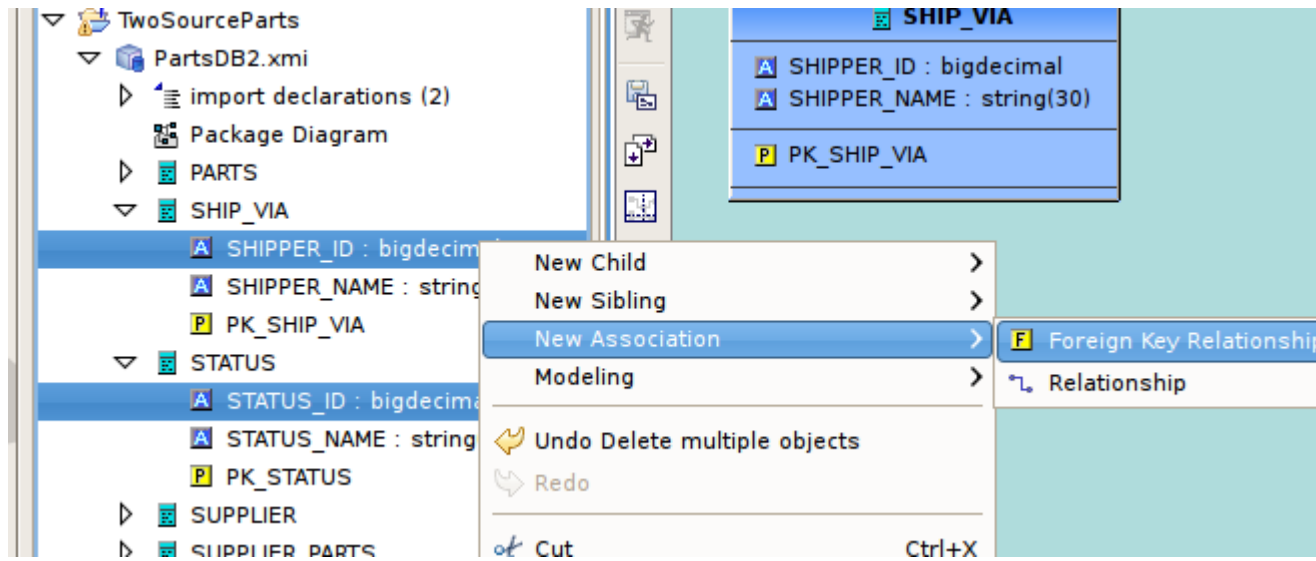


Figure 8.9. New Association Action In Model Explorer

- **Step 3** - The new relationship link is displayed in the diagram.

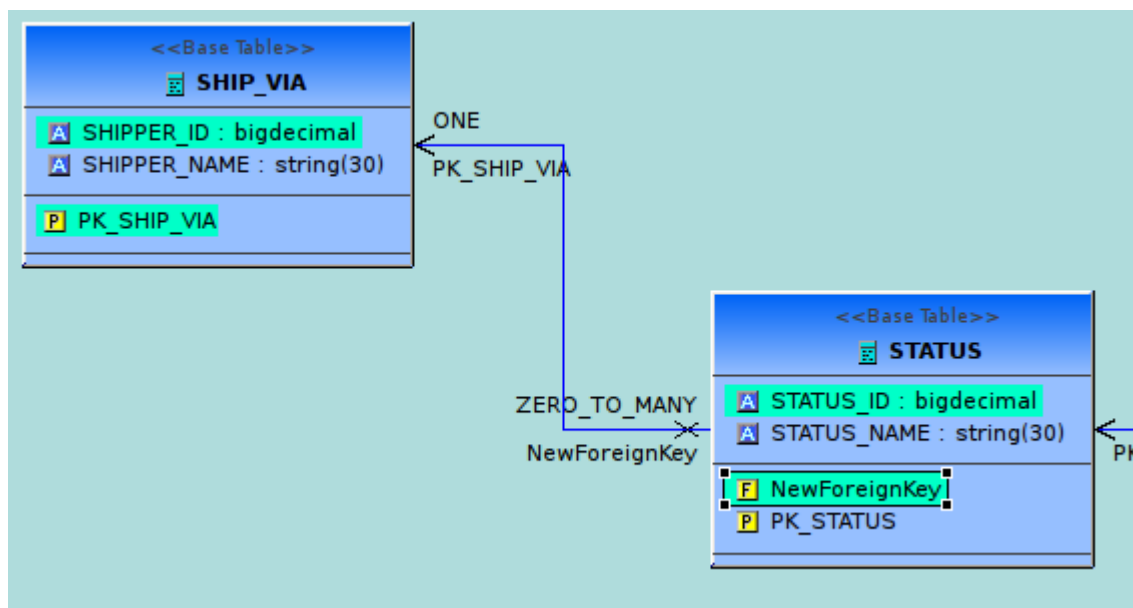






Figure 8.10. New Association In Diagram

- To create new associations between model objects in the [Section 5.1.1, "Diagram Editor"](#):
- **Step 1** - Select two objects you wish to associate. For example, select columns in different base tables.
- **Step 2** - Right-click. From the pop-up menu, select **New Association > Foreign Key Relationship**.

- **Step 3** - The new relationship link is displayed in the diagram. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.
OR
 - **Step 1** - Select a column in table.
 - **Step 2** - Drag the column to another table and drag over a column and drop onto this column. The target column should highlight in Yellow.
 - **Step 3** - The new relationship link is displayed in the diagram. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.
- To create new associations between model objects in the [Section 5.1.2, "Table Editor"](#):
 - **Step 1** - Select two objects you wish to associate. For example, select columns in different base tables.
 - **Step 2** - Right-click. From the pop-up menu, select **New Association > Foreign Key Relationship..**
 - **Step 3** - New Foreign Key and Primary Key objects will be added to the contents of their respective tabs in the Table Editor. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.

8.2. Model Object Editors

The primary actions for editing model objects are:

- 
Cut - Deletes the selected object(s) and copies it to the clipboard.
- 
Copy - Copies the selected object(s) to the clipboard.
- 
Paste - Pastes the contents of the clipboard to the selected context.
- **Clone** - Duplicates the selected object in the same location with the same name; user is able to rename the new object right in the tree.
- 
Delete - Deletes the selected object(s).

- **Rename** - Allows a user to rename an object.

These actions are presented in Teiid Designer's main **Edit** menu and also in the right-click context menus for model objects selected in the [Figure 4.2, “Model Explorer View”](#), [Section 5.1.1, “Diagram Editor”](#) and [Section 5.1.2, “Table Editor”](#).

Modeling Sub-Menu. In addition to the New Child/Sibling/Association menus available for object creation Designer provides a *Modeling* > sub-menu which presents various object-specific actions which can be performed.

If you select a source table, for instance, the modeling menu below would be presented:

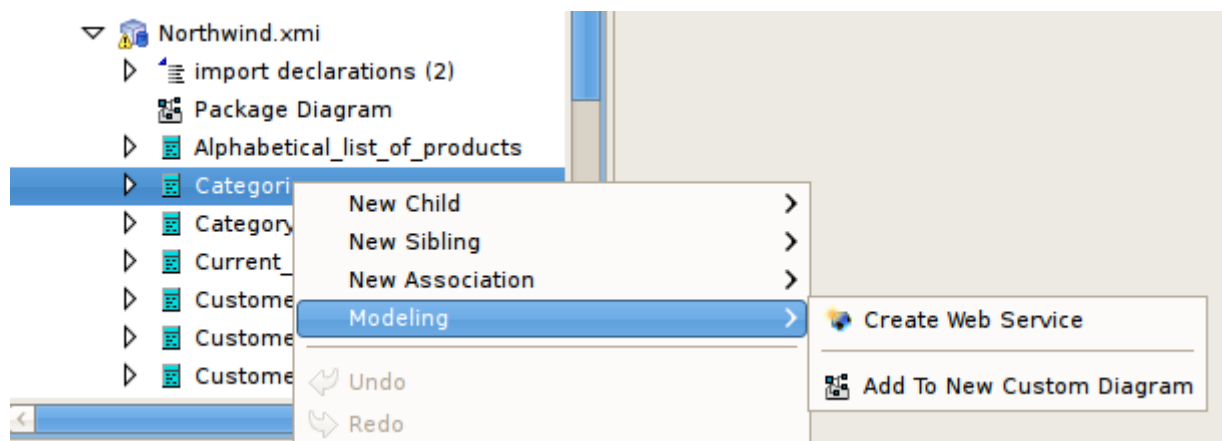


Figure 8.11. Modeling Sub-Menu for Source Table

If a view table is selected, the menu would reflect the actions related to virtual operations:

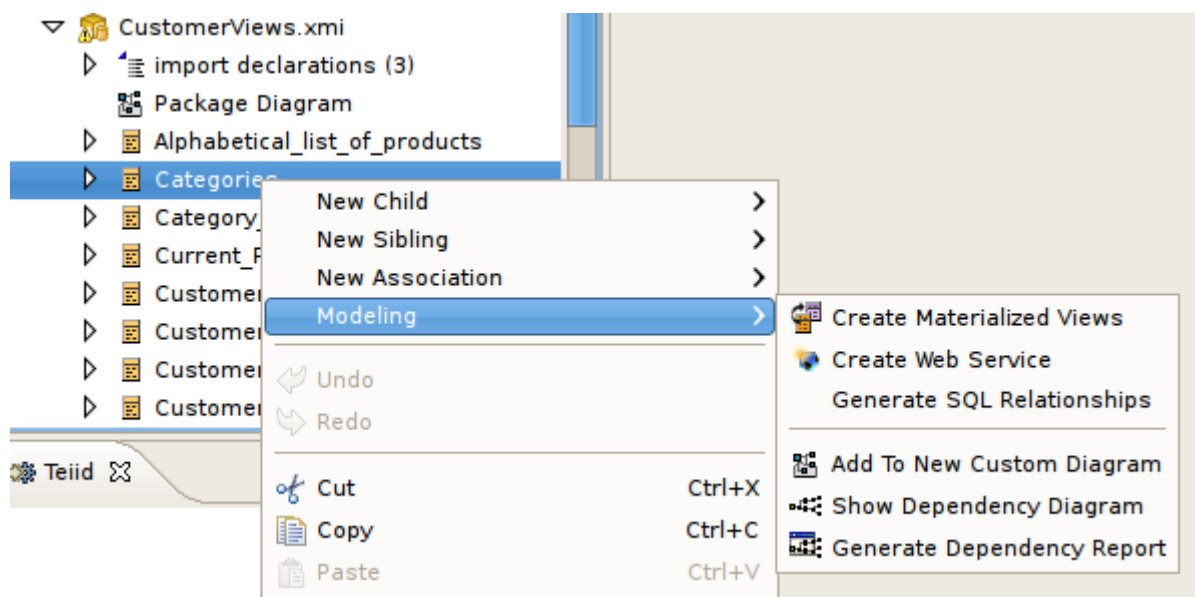


Figure 8.12. Modeling Sub-Menu for Source Table

Teiid Designer also provides specialized object editors to handle complex model objects and their unique properties. These objects include:

- [Section 8.2.1, “Transformation Editor”](#)
- [Section 8.2.2, “Input Set Editor \(XML\)”](#)
- [Section 8.2.3, “Choice Editor \(XML\)”](#)
- [Section 8.2.4, “Recursion Editor \(XML\)”](#)
- [Section 8.2.5, “Operation Editor”](#)

This section describes these editors in detail.

8.2.1. Transformation Editor

The **Teiid Designer's Transformation Editor** enables you to create the query transformations that describe how to derive your virtual metadata information from physical metadata sources or other virtual metadata and how to update the sources.

The **Transformation Editor** provides a robust set of tools you can use to create these SQL queries. You can use these tools, or you can simply type a SQL query into the Transformation Editor.

To edit a transformation you can:

- Double-click Edit
 - A relational view table or procedure in the Model Explorer or Diagram Editor
 - A transformation node in a transformation diagram or mapping transformation diagram
 - A mapping class in a mapping diagram or mapping transformation diagram
- Right-click Edit action on selected object in the Model Explorer, Diagram Editor or Table Editor
 - A relational view table or procedure
 - A transformation node in a transformation diagram or mapping transformation diagram
 - A mapping class in a mapping diagram or mapping transformation diagram

If a Model Editor is not currently open for the selected object's model, a Model Editor will be opened.

After the corresponding transformation diagram is opened in the Diagram Editor, the Transformation Editor is displayed in the lower section of the Diagram Editor.

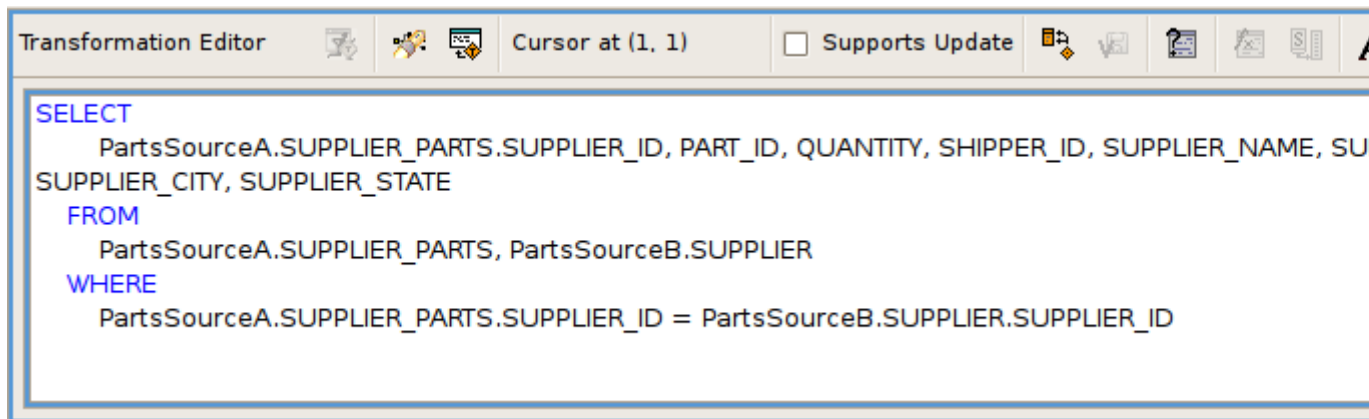


Figure 8.13. Editing String Property

If this virtual class supports updates, the tabs on the bottom of the **Transformation Editor** allow you to enter SQL for each type of query this virtual class supports. If this virtual class does not support updates, only the **SELECT** tab is available.

You can enter separate SQL queries on each available tab to accommodate that type of query.

Within the **Transformation Editor**, you can:

- Disable specific update transformation types on this virtual class.
- Build or edit a criteria clause to use in your transformation.
- Build or edit an expression to use in your transformation.
- Find and replace a string within your transformation.
- Validate the transformation to ensure its content contains no errors.
- Reconcile target attributes to ensure the symbols in your transformation match the attributes in your virtual metadata class.

You can also set preferences that impact the display of your **Transformation Editor**. For more information, see [Section 13.1.3.3, "Transformation Editor Preferences"](#)

- The **Transformation Editor** toolbar actions are summarized below.



Preview Virtual Data - executes a simple preview query for the target table or procedure of the transformation being edited.



Search Transformations - provides a simple way select and edit another transformation based SQL text search criteria.



Edit Transformation - provides a simple way to change which transformation to edit without searching in a diagram or the Model Explorer. Simply click the action and select from a list of views, tables, procedures or operations from the currently edited model.



Cursor Position (line, column) - shows the current line and column position of the insertion cursor. For example, Cursor Position(1,4) indicates that the cursor is presently located at column 4 of line 1.



Supports Update - checkbox allows you to enable or disable updates for the current transformation target. If 'Supports Update' is checked, the editor shows four tabs at the bottom for the Select, Update, Insert and Delete transformations. If 'Supports Update' is unchecked, all updates are disabled and only the Select transformation is displayed.



Reconcile - allows you to resolve any discrepancies between the transformation symbols and the target attributes. Pressing this button will display the "Reconcile Virtual Target Attributes" dialog box in which you can resolve discrepancies.



Save/Validate - saves edits to the current transformation and validates the transformation SQL. Any Warning or Error messages will be displayed at the bottom of the editor in the messages area. If the SQL validates without error, the message area is not displayed.



Criteria Builder - allows you to build a criteria clause in your transformation. The button will enable if the cursor position is within a query that allows a criteria. Pressing the button will launch the Criteria Builder dialog. If the Criteria Builder is launched inside an existing criteria, that criteria will be displayed for edit, otherwise the Criteria Builder will be initially empty.



Expression Builder - allows you to build an expression within your transformation. The button will enable if the cursor position is at a location that allows an expression. Pressing the button will launch the Expression Builder dialog. If the Expression Builder is launched inside an existing expression, that expression will be displayed for edit, otherwise the Expression Builder will be initially empty.



Expand Select * - allows you to expand a "SELECT *" clause into a SELECT clause which

contains all of the SELECT symbols. The button will enable only if the cursor is within a query that contains a SELECT * clause that can be expanded.



Increase Font Size - increases the font size of all editor text by 1.



Decrease Font Size - decreases the font size of all editor text by 1.



Show/Hide Messages - toggles the display of the message area at the bottom of the transformation editor.



Optimize SQL - when toggled 'ON', will use the short names of all SQL symbols that can be optimized. Some symbol names may remain fully qualified in the event of a duplicate name or if the optimizer is unable to optimize it. When the action is toggled 'OFF', all symbol names will be fully-qualified.



Import SQL Text - allows you to import a sql statement from a text file on your file system. Pressing this button will display an import dialog in which you can navigate to the file.



Export SQL Text - allows you to export the currently displayed SQL statement into a text file on your file system. Pressing this button will display an export dialog in which you can choose the location for export.

- **Close "X"** - closes the transformation editor.

8.2.1.1. Using the Criteria Builder

The Transformation Editor's **Criteria Builder** offers you a quick, graphical means to build criteria clauses in your transformations based on meta objects in your diagram. If you launch the **Criteria Builder** with your cursor within an existing criteria in your transformation SQL, the builder will open in Edit mode. If your cursor is not in an existing criteria location, the builder will open in create mode and allow you to create it from scratch.

This procedure provides an example of building a criteria clause using the **Criteria Builder**. When building your own criteria, you can mix and match the values and constants with whatever logic you need to build powerful and complex criteria.

- **To use the Criteria Builder:**

- **Step 1** - In the Transformation Editor, click the **Launch Criteria Builder** button.



- **Step 2** - The **Criteria Builder** displays.

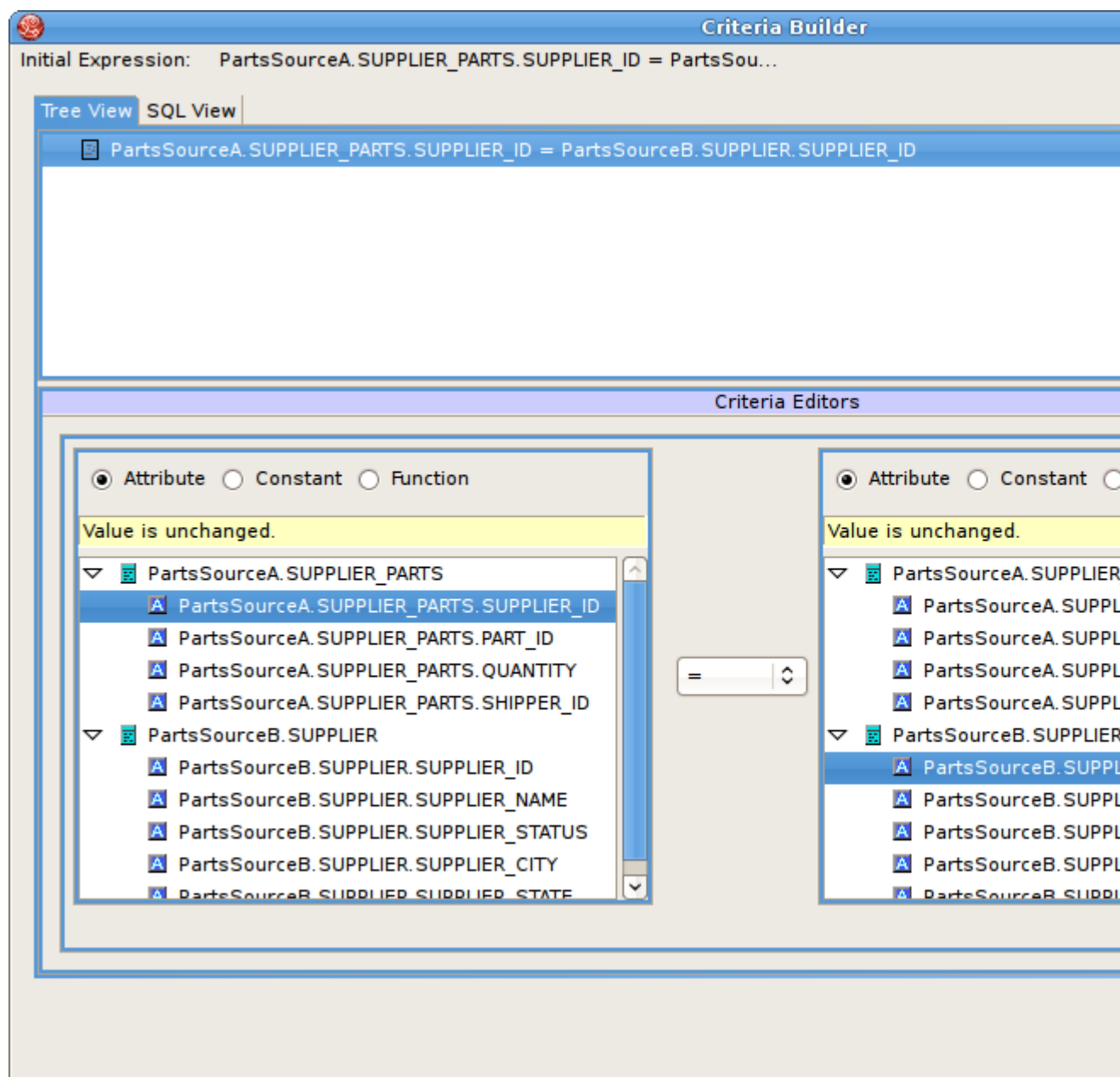


Figure 8.14. Editing String Property

The two tabs at the top, **Tree View** and **SQL View**, show the current contents of the criteria you have built.

The **Criteria Editor** at the bottom allows you to build a criteria clause. To build a criteria clause, you must add information to the left side of the predicate, select a comparison operator, and add a value to the right side.

- **Step 3** - The radio buttons on either side of the **Predicate Editor** let you choose what type of content to place in that side of your predicate. Click the radio button of the type of content you want to place in your criteria. You can click:
 - **Attribute** to add an attribute to the predicate. If you click the **Attribute** radio button, the Predicate Editor looks like this:

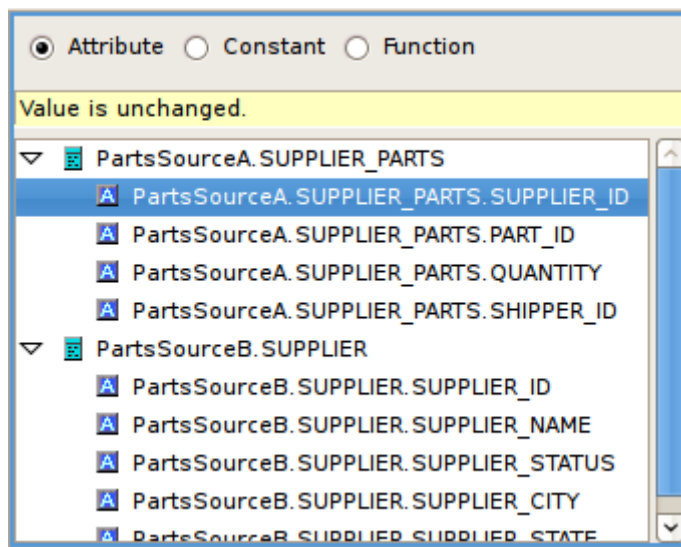


Figure 8.15. Attribute Panel

From the tree, select the attribute you want to add to the expression. You can select an attribute from any of the source classes in the transformation.

- **Constant** to add a hard-wired constant value to the predicate. If you click this radio button, the **Predicate Editor** looks like this:

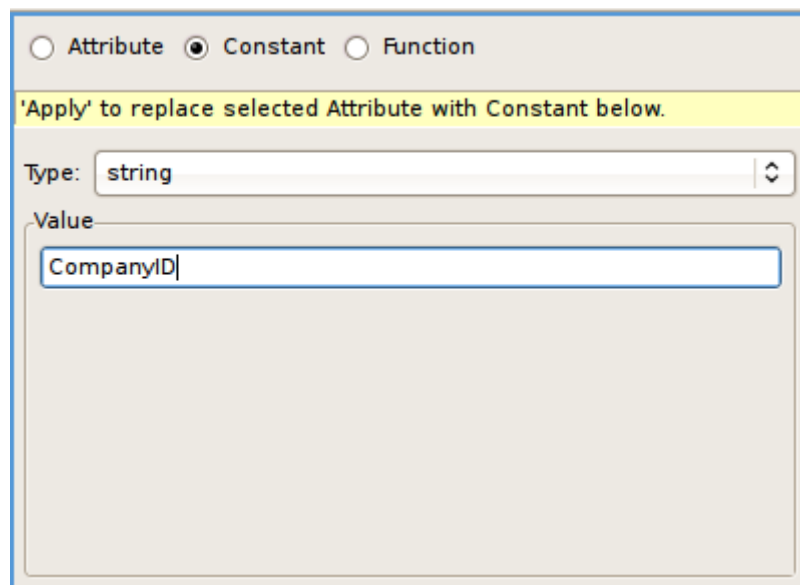


Figure 8.16. Constants Panel

Select the datatype for this constant from the Type drop-down list and enter the value in the Value edit box.

- **Function** to add a function. If you click the Function radio button, the Predicate Editor looks like this:

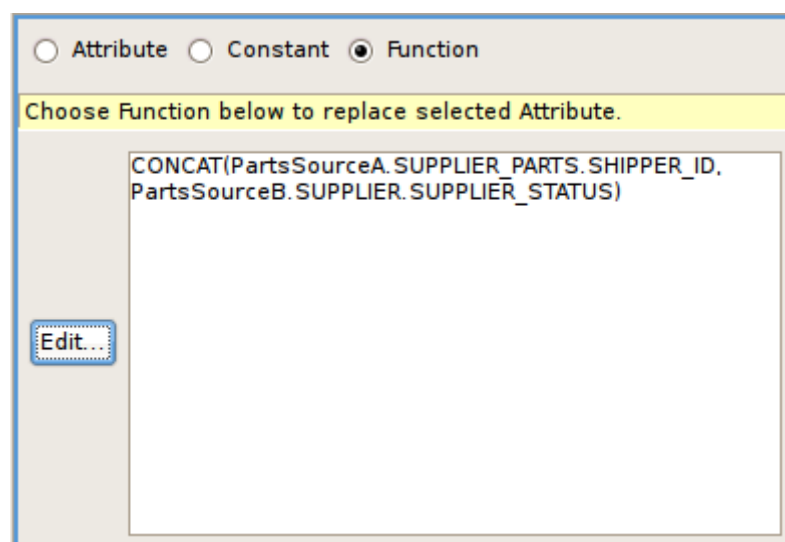


Figure 8.17. Functions

Click the Edit button to use the Expression Builder to construct a function to use in the predicate of your SQL Criterion. For more information about the Expression Builder, see [Section 8.2.1.2, “Using the Expression Builder”](#)

- **Step 4** - Set a value left side of the predicate and, when necessary, the right side of the predicate. If the right side of the predicate does not require a value of some sort, the Criteria Builder will not let you enter one.
- **Step 5** - Click Apply.
- **Step 6** - When you have created both a Left Expression and a Right Expression in the Predicate Editor, click Apply to add the criterion to the tree view at the top of the dialog box.

The criteria clause displays in the Criteria tree.

You can create complex criteria by joining other criteria with this one. To join criteria with this one, select the criteria in the Criteria tree and click:

- Delete to remove the selected criterion.
- AND to create a new criterion that must also be true.
- OR to create a new criterion that can be true instead of the selected criterion.
- NOT to establish negative criterion.

If you join a criterion to the one you just completed, you build the expression the same way, using the Expression Editors panel and the Predicate Editor panel. You can create complex, nested criteria by judicious use of the AND and OR buttons.

Once you have created the complete criteria you want, click OK to add it to your transformation.


8.2.1.2. Using the Expression Builder

The **Transformation Editor's Expression Builder** offers you a quick, graphical means to build expressions in your transformations. This **Expression Builder** lets you create:

- Attributes by selecting an attribute.
- Constants by selecting the datatype and value.
- Functions from both the standard Teiid Designer SQL functions and your enterprise's custom user-defined functions. If you select a function before you launch the Expression Builder, you can use the Expression Builder to edit the selected function; otherwise, you can create a new function from scratch.

- To use the **Expression Builder**:

- **Step 1** - In the **Transformation Editor**, click the location where you want to insert the function.

- **Step 2** - Click the **Expression Builder** button. 

The SQL Expression Builder displays.

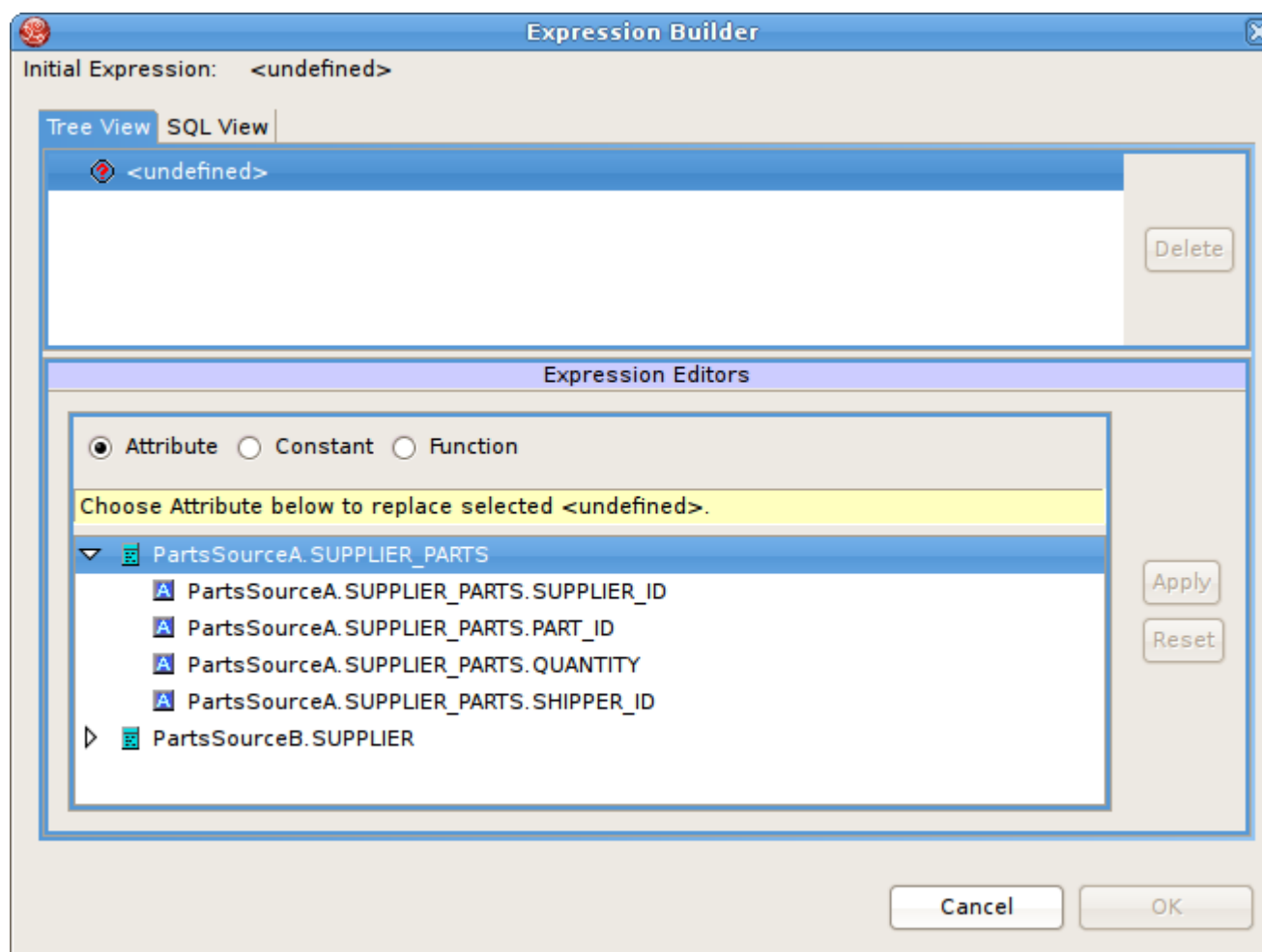


Figure 8.18. Expression Builder

The two tabs at the top, Tree View and SQL View, show the current contents of the expression you have built. To build an expression, you must specify the type of expression you want to build and populate it. In most cases, you will use the Expression Builder to construct a complex expression.

- **Step 3** -Click the Function radio button to add a function.



Note

You can simply add constants and attributes as expressions by themselves using the **Attribute** or **Constant** radio buttons; however, the **Expression Editor** is most useful for functions.

- **Step 4** - The Expression Editor displays the Function editor.

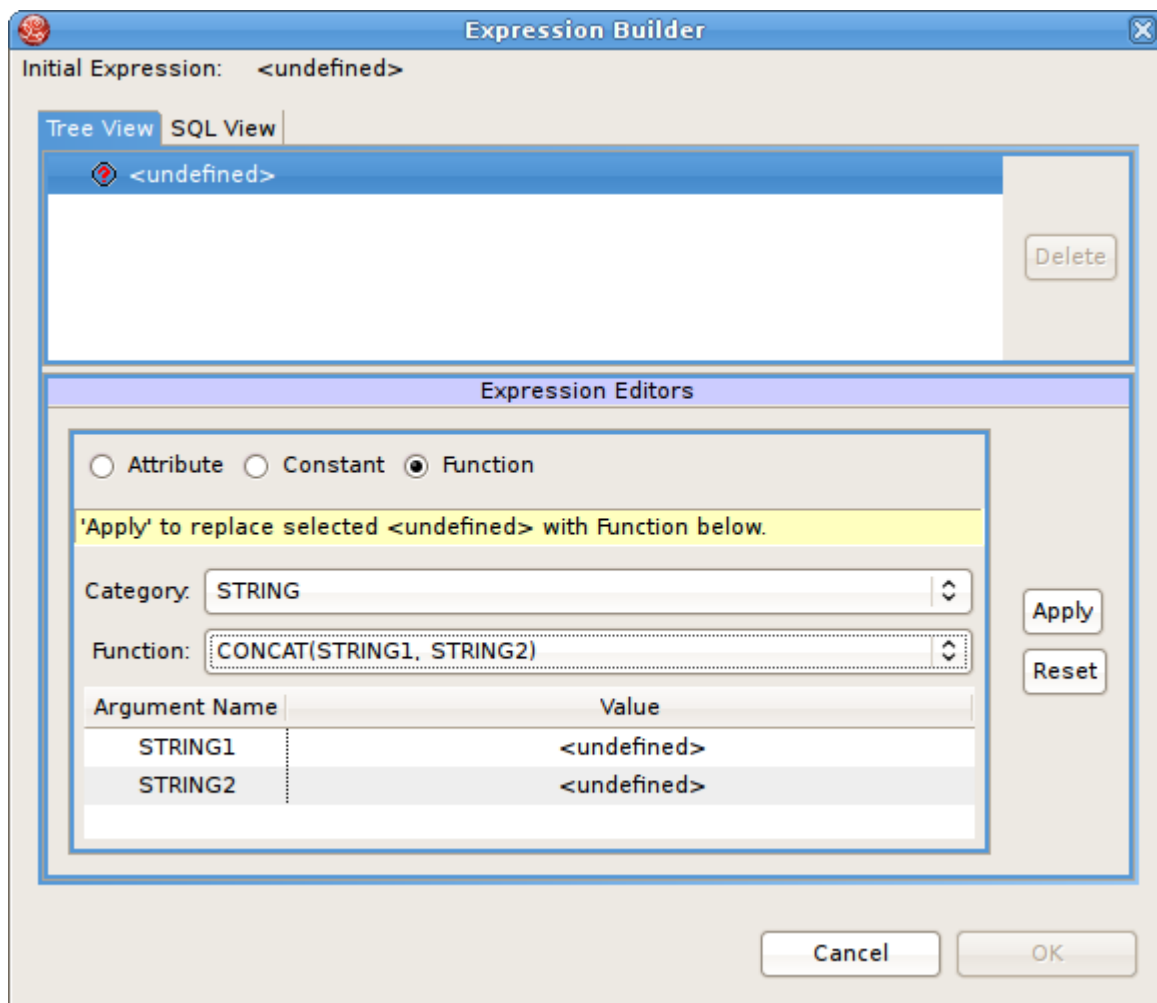


Figure 8.19. Function Panel Selected

From the Category drop-down list, choose the type of function you want to add. By default, the Teiid Designer System offers the following categories:

- **Conversion** for functions that convert one datatype into another.
- **Datetime** for functions that handle date or time information.
- **Miscellaneous** for other functions.
- **Numeric** for mathematic and other numeric functions.
- **String** for string manipulation functions.



Note

Any additional categories represent those containing user-defined functions your site has created.

- **Step 5** - From the **Function** drop-down list, select the function you want. The table beneath the drop-down lists displays the number of arguments required for this function.
- **Step 6** - Click **Apply**.
- **Step 7** - Your function displays in the tree at the top. Sub nodes display for each argument you need to set for this function.

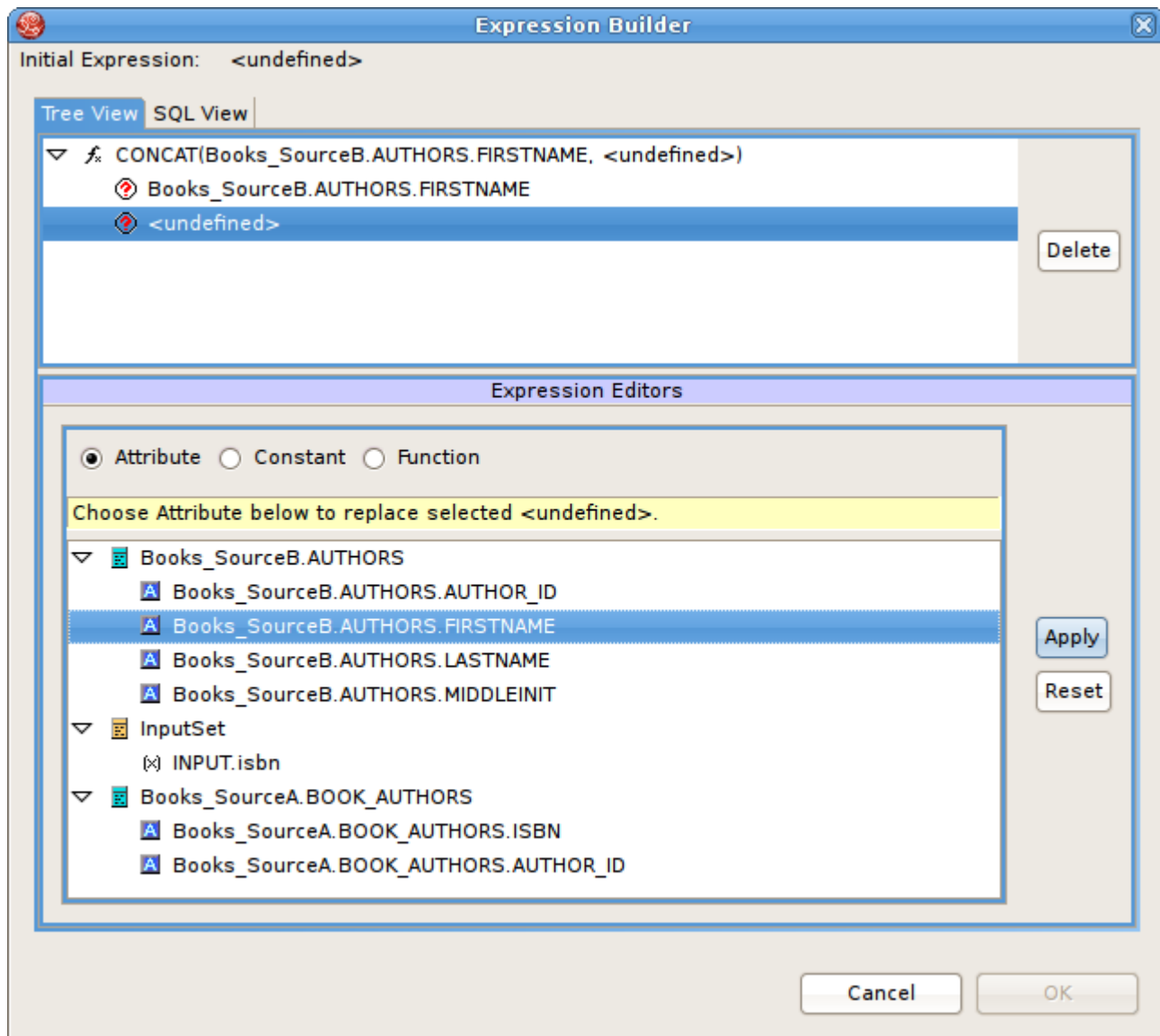


Figure 8.20. New Blank Function Created

You need to set an attribute or constant value for each sub node in the tree to specify the arguments this function needs. You can also nest another function in the tree using the **Function** editor.

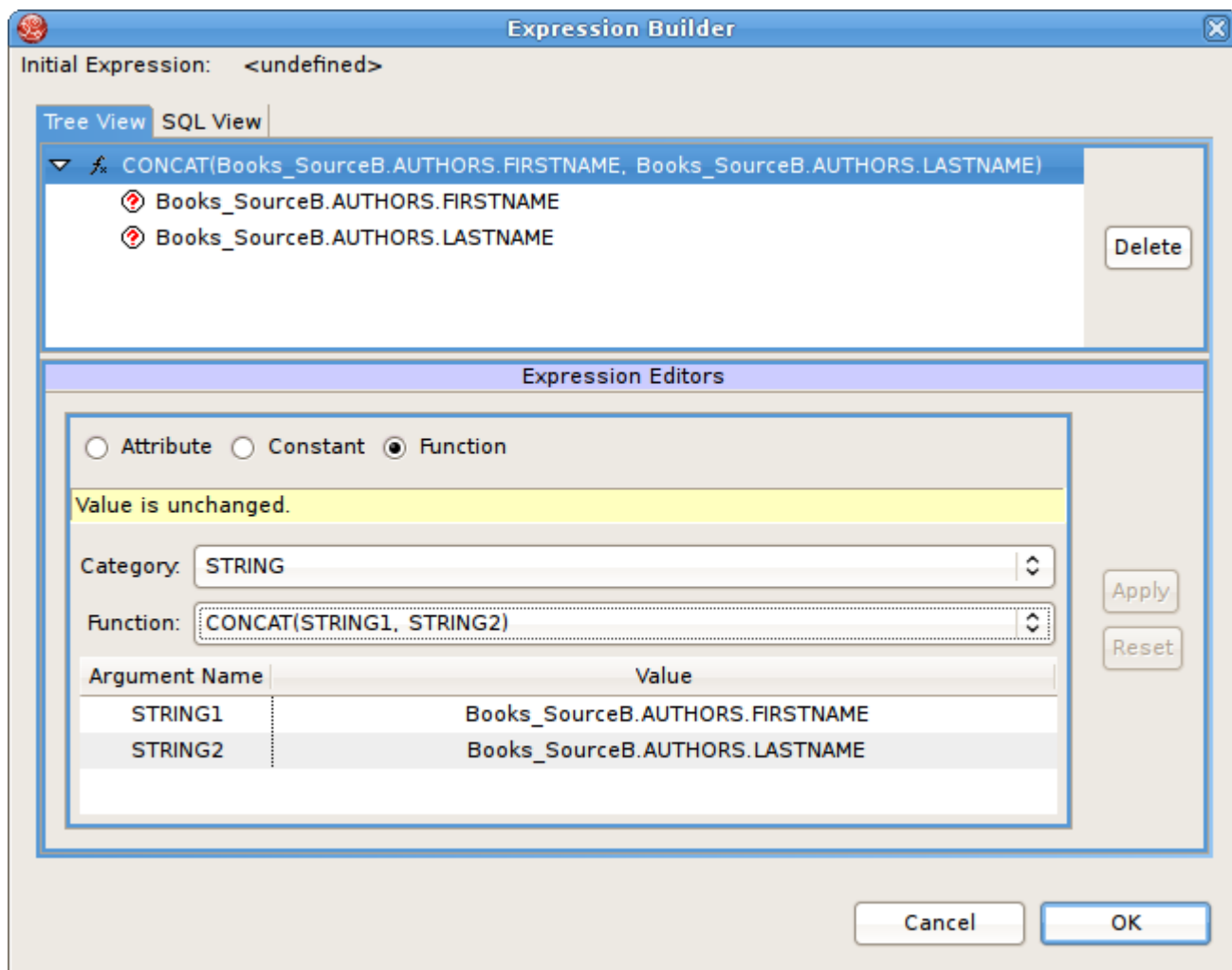


Figure 8.21. Nested Function Example

- **Step 8** - Click each sub node in the tree and use the editors at the bottom of the dialog box to apply an attribute, constant, or function value to it.
- **Step 9** - When you have added values to all nodes, as shown below, click **OK**. to add this expression to your query or **Cancel** to close the dialog box without inserting the expression.

If the **OK** button does not enable, you have not added a value to all nodes in the tree. You can also nest functions within your expressions by selecting an argument and selecting a function for that argument. The nested function displays in the tree beneath your root function and its arguments display as well. Using the Expression Builder and nested functions, you can create complex logic within your query transformations.

8.2.2. Input Set Editor (XML)

The **Input Set** represents a special class that contains attributes from a parent mapping class. When you create mapping classes for an **XML Document** model, the Teiid Designer automatically

adds an **Input Set** to all XML transformation diagrams for mapping classes beneath the highest node in the Document meta object.

The **Input Set** proves especially useful for information integration using the **Teiid Designer Server**. Through the **Input Set**, you can access a row of data generated by any XML transformation in a mapping class higher in the XML document's hierarchy. You can use **Input Set** attributes, which are individual columns from the rows of data, within the criteria of an XML transformation query of the child mapping class.

You cannot use the **Input Set** attributes within the SELECT portion of the XML transformation query.

To use an **Input Set**, you must use the **Input Set Editor** to bind attributes from parent classes.

Once you have created an **Input Set**, you can use the attributes within it as source material for the XML transformation diagram's query.

The **Input Set** only serves to enable data flow between nested mapping classes. If you use the **Teiid Designer Server** for data access, your applications cannot directly query an **Input Set**. **Input Sets** only display in the XML transformation diagram to which they belong. **Input Sets** do not display on the [Figure 4.2, "Model Explorer View"](#) and you cannot use them as you would a normal class, such as for source classes in other transformations.

To open the **Input Set Editor**, either double-click the input set in the **Mapping Transformation Diagram** or click the edit button on the **Input Set** in the diagram. (see below)

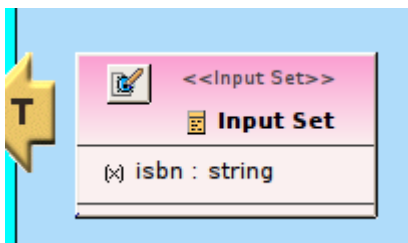


Figure 8.22. Edit Input Set Button

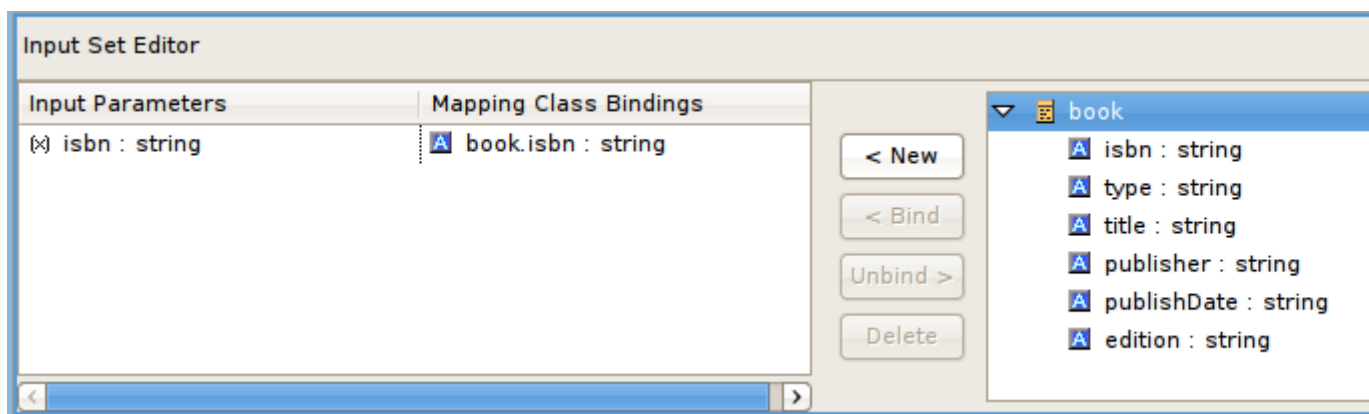


Figure 8.23. Input Set Editor Panel

The **Input Parameters** table contains a list of mapping attributes within the input set and the mapping attributes bound to input set mapping attributes. The tree on the right displays the parent mapping classes and the attributes available from each.

Using the **Input Set Editor**, you can:

- **Add** a mapping attribute from a parent mapping class to the **Input Set**. In the tree on the right, select the symbol for which you want to create an attribute and click **New**. The item displays in the **Input Parameters** and **Mapping Class Bindings** table.
- **Delete** a mapping attribute from the **Input Set**. Click the row in the **Input Parameters** and **Mapping Class Bindings** table that you want to delete and click **Delete**. The Teiid Designer removes this row from the table and this mapping attribute from your **Input Set**.
- **Bind and Unbind Input Parameters**.

Once you have created the mapping attributes within the **Input Set** that you need, you can use the **Input Set Parameters** within a mapping class transformation to produce mapping attributes you can map to your XML document.

8.2.3. Choice Editor (XML)

Within an XML Document model, a choice compositor defines all possible document data structures (sometimes called fragments) that can appear at that location in an XML instance document. When the Teiid Designer Server populates an XML instance document at runtime based upon your virtual XML document, it will choose the first fragment that matches the criteria you specify within the **Choice Editor**.

To view the choice editor, right-click on the choice node in the mapping diagram's XML Document tree view and select **Edit** from the right-click pop up menu.

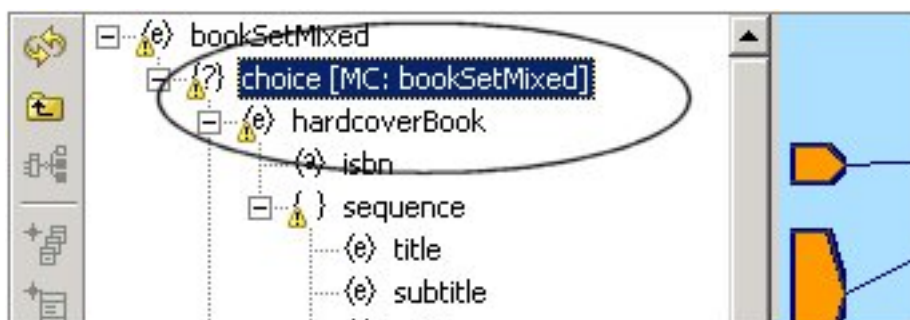


Figure 8.24. Opening The Choice Editor

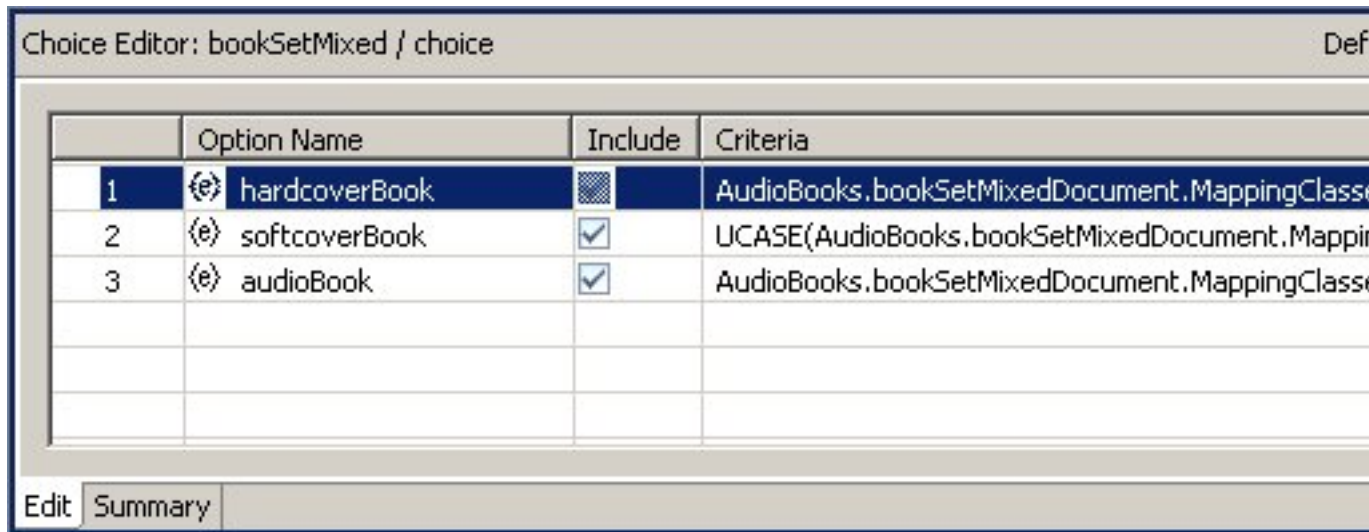


Figure 8.25. The Choice Editor

The table on this panel displays fragment options for the choice, each represented by the top node of the document fragment.

The Summary tab, shown below, displays a SQL-like version of the current choice criteria.

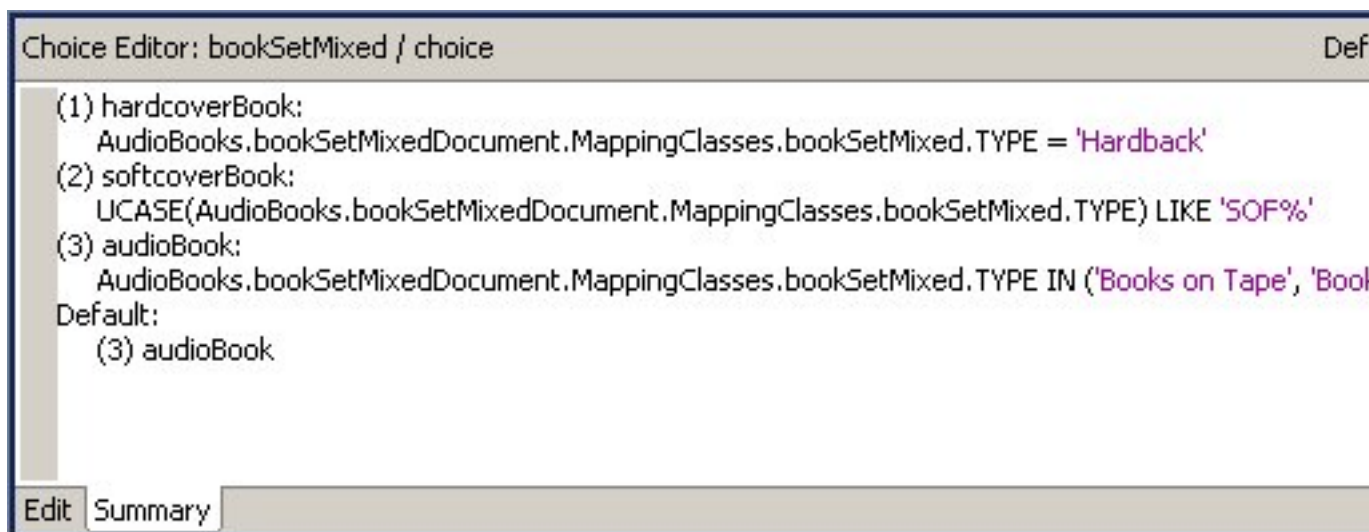


Figure 8.26. Choice Criteria Summary

8.2.3.1. Using the Choice Editor

You should address each choice option by performing one of the following:

- **Specify a criteria** statement for the Teiid Designer Server to apply in order to determine which elements or elements to insert into the result document.

- **Exclude or include** the option's fragment from the document.
- Set the elements' criteria **test order**.
- **Set a default action** that occurs if none of the criteria you set is met.

8.2.3.2. Excluding Fragments

The XML Schema upon which you based the XML Document model determines the nature of the options available to the choice. A schema you share with other, external sources (such as business partners) might include information that you do not want to include within XML files.

For example, Sample Financial Services shares an XML schema with its partners Example Mutual Insurance, Illustrative Brokerage, and FinancialPartners.com. The partners created the schema broadly, to cover all possibilities for information they might need to interchange. As such, the customer information XML document might include a choice compositor based on a list of all products all companies offer.

However, Sample Financial does not offer a credit card; so it could exclude those elements from the XML documents its Teiid Designer Server creates since it will never have credit card information for an XML document.

The table on the **Choice Editor** contains the **Include** column. By default, all elements specified by the schema are included. You can click to remove the checkmark beside any element you do not want to include within your XML documents generated by this virtual XML document metadata model. By removing the checkmark, you are not removing the element from the XML Document model; you are merely telling the Teiid Designer Server that it will never use this element as part of the choice.

You cannot edit criteria for excluded elements. However, if you exclude an option for which you have established a criteria, Teiid Designer will retain the criteria if you want to include the option in the future.

8.2.3.3. Editing Choice Criteria

- To edit the criteria for a choice element:
 - **Step 1** - In the table on the **Choice Editor** panel, select the element you want to edit..
 - **Step 2** - Click **Edit Criteria** button to launch the **Criteria Builder** dialog.
 - **Step 3** - Use the **Criteria Builder** to create the conditions for which the Teiid Designer Server will test to determine whether to choose this option in the XML instance document.
 - **Step 4** - Click OK. The criteria you set displays both in the table and in the summary tab.

You must set a criterion for each option in your document unless you have selected to exclude that option or specify that option will be the default option.

8.2.3.4. Setting Choice Element Order

To edit the criteria for a choice element:

The Teiid Designer Server evaluates the choice criteria in the order in which they appear, and when one choice criteria is met, the Teiid Designer Server populates the XML instance document with that option. The Teiid Designer Server might not test all criteria for all options, so their order matters a great deal.

Therefore, the order in which your options appear within the choice criteria often determines what information appears ultimately in your XML instance documents. You can reorder the option list within the choice to set the order in which the Teiid Designer Server tests the criteria.

To set this order, select an element in the table and use the



or



button to move it into a new position in the table. The new order displays both in the table and in the Choice Criteria box and reorders the XML document as well.

8.2.3.5. Setting a Default Choice Action

The default action represents the course the Teiid Designer Server should take if none of the criteria you set evaluates to true.

You can set this default using the combo box available in the **Choice Editor's** toolbar to:

- Any of the options within the table except those you have excluded from the document.
- **THROW** to throw a Teiid Designer Server exception.
- **RECORD** to record the Teiid Designer Server exception.
- **DISCARD** to place no element within the XML instance document.

Note: You must set a default action for your choice criteria.

8.2.4. Recursion Editor (XML)

Some **XML schemas** define data structures that contain self-referencing elements or datatypes. When generating XML documents, such data structures can produce an endless repetition of nested tags. This self-nesting pattern is known as **recursion**.

When generating virtual documents from **XML Schema**, the Teiid Designer detects recursive data structures in the XML Schema model and halts the recursive nesting pattern after two cycles. These two cycles serve different purposes when mapping the document:

- The **first cycle** can be thought of as an “**entry condition**” for the **recursion**. The mapping class located at this node defines a normal mapping transformation like that of any other in the document model.
- The **second cycle** defines a mapping transformation that will be performed repeatedly until conditions are met that will halt the document instance being generated by the Teiid Designer Server. This fragment of the document model is called the **recursive fragment**. The mapping transformation for this fragment is no different from the first, except that you can access the first cycle's mapping class attributes, plus you have the opportunity to specify the conditions that will halt the recursion.

You can recognize a mapping class located at the second, recursive document fragment by the looping arrow button in the top-left-hand corner of the diagram object as shown below.

When you model a virtual document based on an **XML Schema** model containing recursion, you can choose whether to treat the nested fragments as recursive. You should only use recursion when the data access pattern from your data source(s) is also recursive; in other words, when the same query transformation should be executed over and over to generate and map the nested document's data content.

By default, the Teiid Designer does not mark the recursive fragments in document models to execute recursively in the Teiid Designer Server. To take advantage of this behavior, you must open the **Recursion Editor** in the recursive mapping class [Section 5.1.1.5, “Mapping Transformation Diagram”](#), mark the transformation query as recursive, and specify the recursion limit properties.

8.2.4.1. The Editor

The Recursion Editor lets you enable and limit recursion. The Recursion Editor button only displays on mapping classes, which have recursive patterns. For example, if you have an element named Employee which contains a element named Supervisor which itself contains an Employee element nested within it, you might need to limit the number of times the elements are nested within the document.

You can set the following conditions to limit the recursion:

- A fixed number of results to the query.
- A SQL-based criteria limit condition.
- A combination of both.

To open the Recursion Editor, click on the Recursion Editor button



on the displayed mapping class.

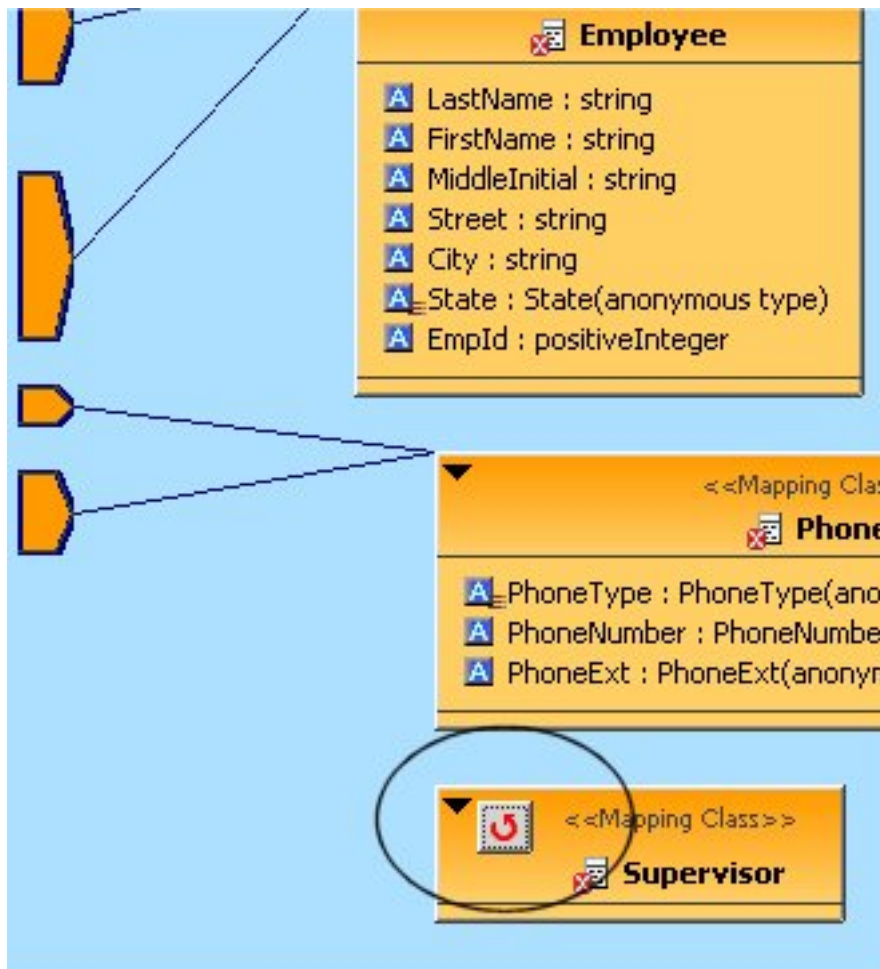


Figure 8.27. Open Recursion Editor Button

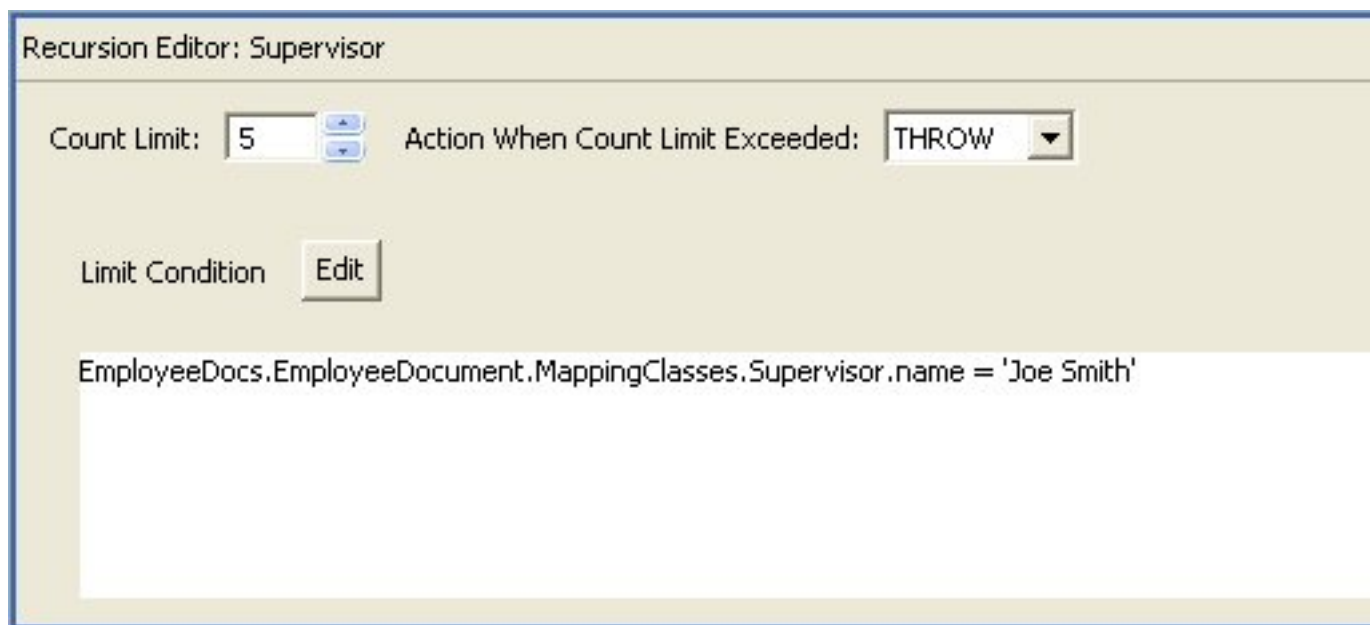


Figure 8.28. Recursion Editor

- To edit recursion properties:
 - **Step 1** - Click the **Enable Recursion** check box if you want the Teiid Designer Server to perform the query you specify to generate the nested tags within the **XML document**.
 - **Step 2** - Click the arrows beside the **Count Limit** box to limit the number of times to recursively perform the query. If you do not set a **Limit Condition** in the text area, the recursion finishes when the query reaches this limit. You can only set this limit to a maximum supported by your Teiid Designer Server. For more information about this limit, contact your system administrator
 - **Step 3** - Click the **Action When Count Limit Exceeded** drop down menu to instruct the Teiid Designer Server what to do if it encounters more results for the query than the count limit before it reaches the limit condition.
 - **Step 4** - Click the **Edit** button to launch the SQL [Section 8.2.1.1, “Using the Criteria Builder”](#) to build a limiting condition for this recursion.



Note

The Teiid Designer Server will evaluate this condition each time it recursively performs this query. If this criteria clause evaluates false, the Teiid Designer Server performs the query recursively again unless it has reached the **Count Limit**. If the criteria evaluates true, the Teiid Designer Server performs the mapping for the current level and ends its recursive loop.

When you have created the criteria, it displays in the **Limit Condition** box.

When the Teiid Designer Server dynamically populates your XML documents at runtime, it will use the recursion specifications you entered here.

8.2.5. Operation Editor

Editing of **Web Service** Operation transformations is simplified via the **Operation Editor**. When editing a Web Service model, an additional editor tab labeled "**Operation Editor**" is available. This editor, shown below is comprised of:

- **Operations** section showing a tree view of Interfaces and Operations contained within the Web Service model.
- **Input Variables** section providing editing of desired Input Variable declarations.
- **Procedure** section providing SQL editing of the procedure.

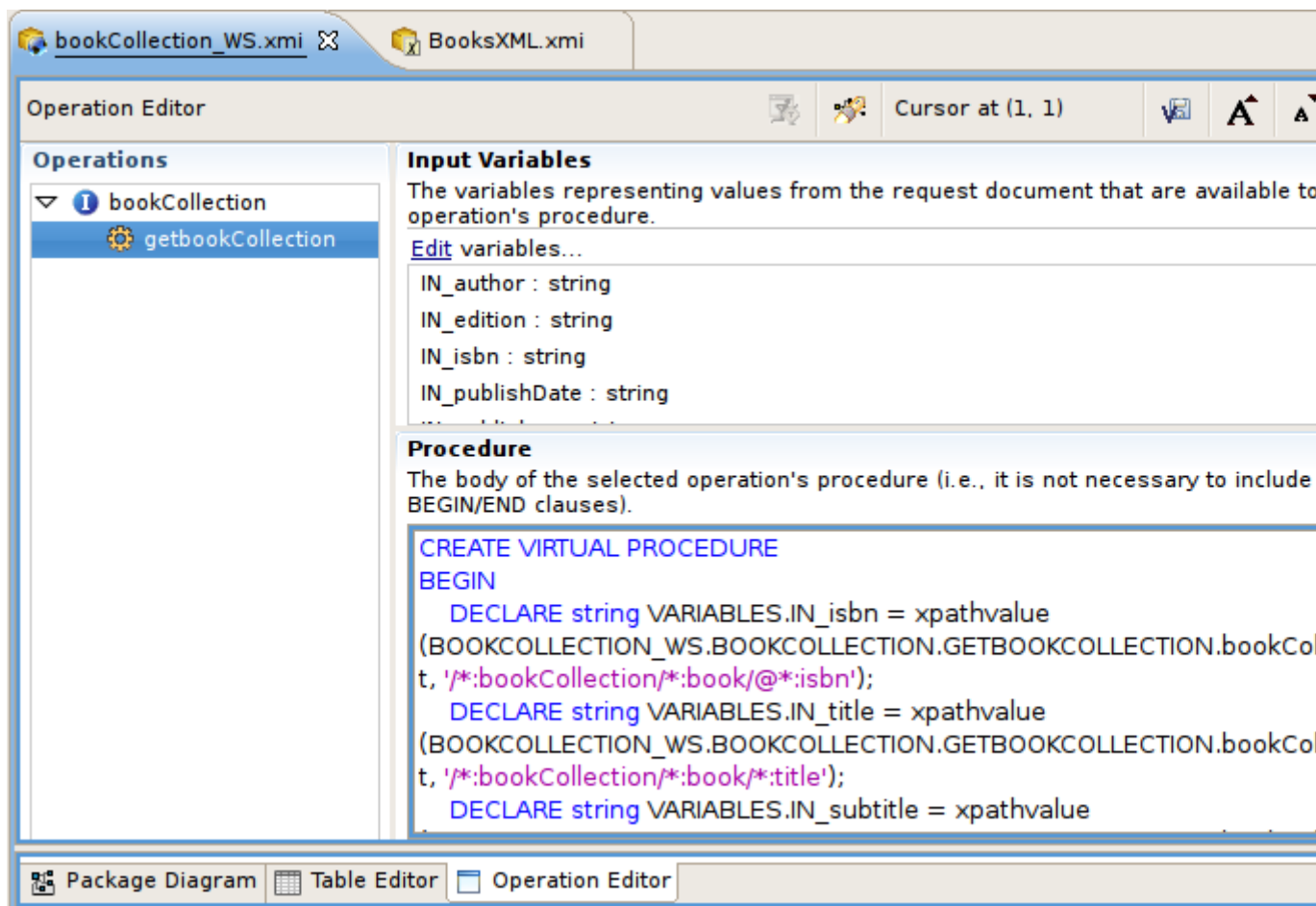


Figure 8.29. Operation Editor

The **Operations** section contains all interfaces and operations currently defined in the model.

Selecting an operation will display the variables related to the input parameter's content in the **Input Variables** section and the body of its procedure (minus the CREATE VIRTUAL PROCEDURE BEGIN - END keywords and the input variable declarations and assignments) in the **Procedure section**.

When pasting in SQL, do not include the **CREATE VIRTUAL PROCEDURE BEGIN - END** keywords. Input variables will be automatically generated when the Content via Element property is set on an operation's input parameter. Input variables may be edited using the Edit link in the **Input Variables** section, and may only represent XPath values to single attributes and elements within the input contents; other variable declarations and assignments must be typed directly into the **Procedure** section. Clicking the *Edit* link will display the following dialog:

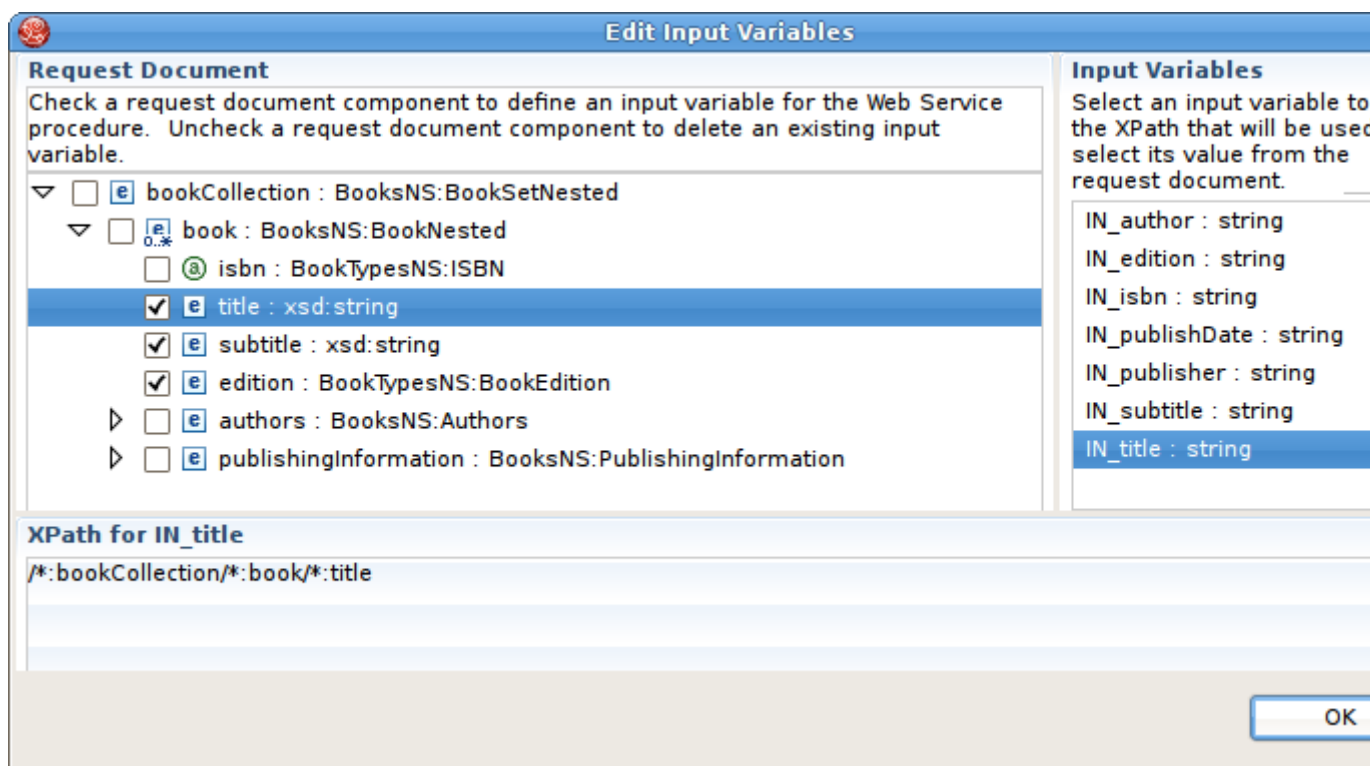


Figure 8.30. Edit Input Variables Dialog

8.3. Manage Extension Properties

In order to facilitate custom property usage for relational models and objects such as tables, columns, and procedures, a dialog to create extended properties was added. These properties are exposed in Teiid as indexed metadata and can be read by your custom translator.

In 7.4, you can add, edit and delete these extended properties by selecting a relational model or model object using the **Modeling > Manage Extended Properties** action. The menu option and dialog is shown below

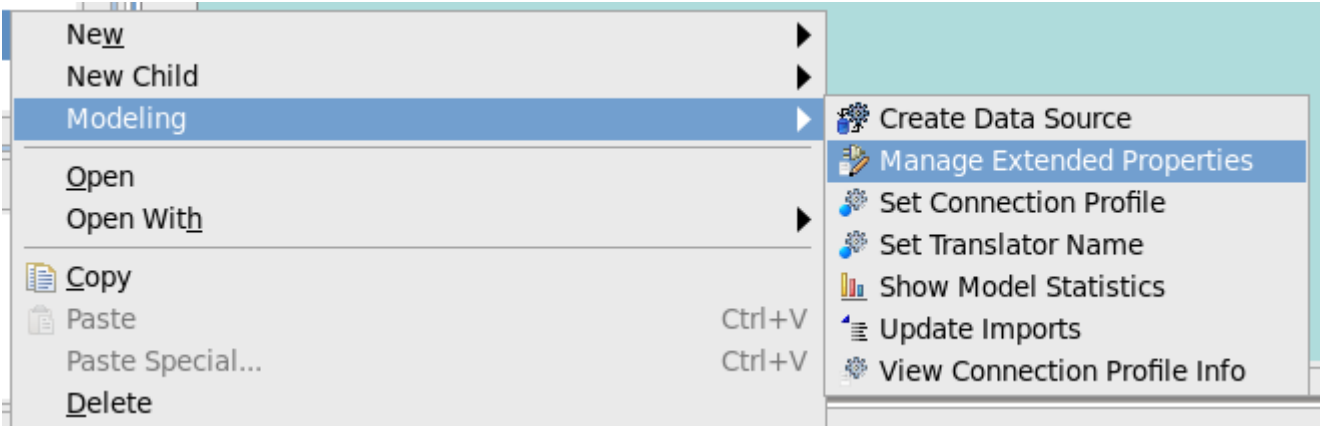


Figure 8.31. Manage Extended Properties Menu Option

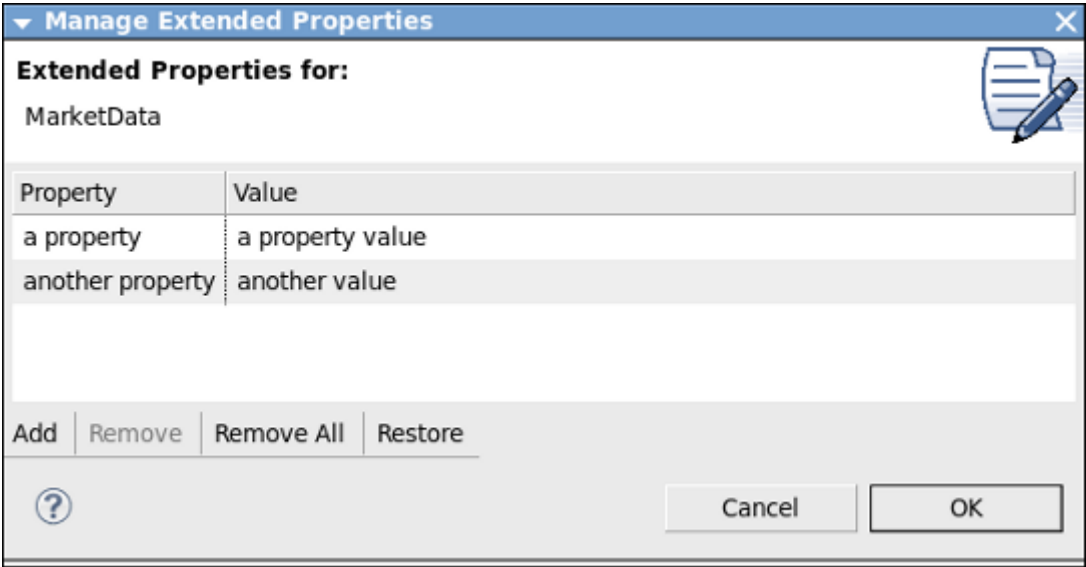


Figure 8.32. Manage Extended Properties Dialog

Editing Models and Projects

Teiid Designer offers three basic model edit actions: **Rename**, **Move** and **Save As...** and one project-related action, **Clone Project**. These actions are described below.

9.1. Rename A Model

- To rename a model in your workspace:
 - **Step 1** - Select a model in the [Figure 4.2, “Model Explorer View”](#).
 - **Step 2** - Right-click select the **Refactor > Rename** action.

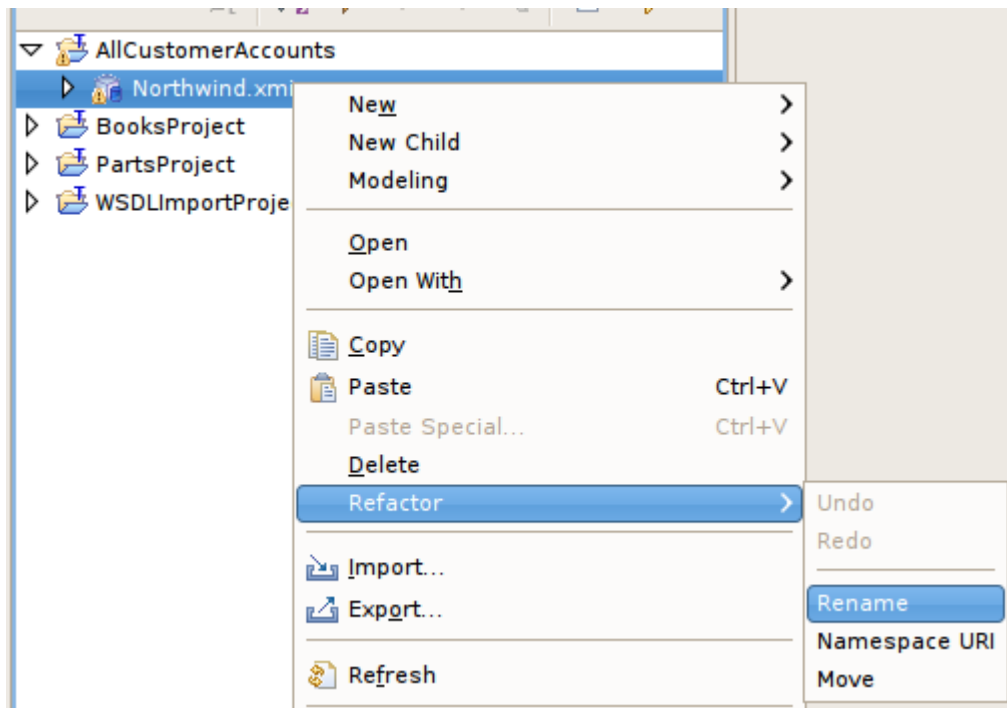


Figure 9.1. Refactor Rename Action In Model Explorer

- **Step 3** - Specify unique model name in the **Rename Model File** dialog. Click **OK**.

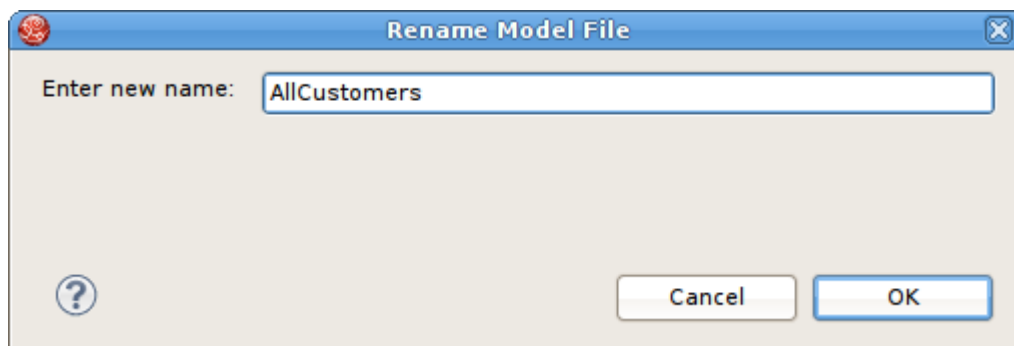


Figure 9.2. Rename Model File Dialog



Note

Renaming a model that is a dependency to another model will automatically change the model imports for those models. If source model `CustomerSource` is renamed to `OldCustomerSource`, for instance, the import statement for the view model `CustomerAccounts` which imports `CustomerSource` will be changed to reflect the new name.

9.2. Move Model

- To move a model in your workspace:
 - **Step 1** - Select a model in the [Figure 4.2, “Model Explorer View”](#).
 - **Step 2** - Right-click select the **Refactor > Move** action.

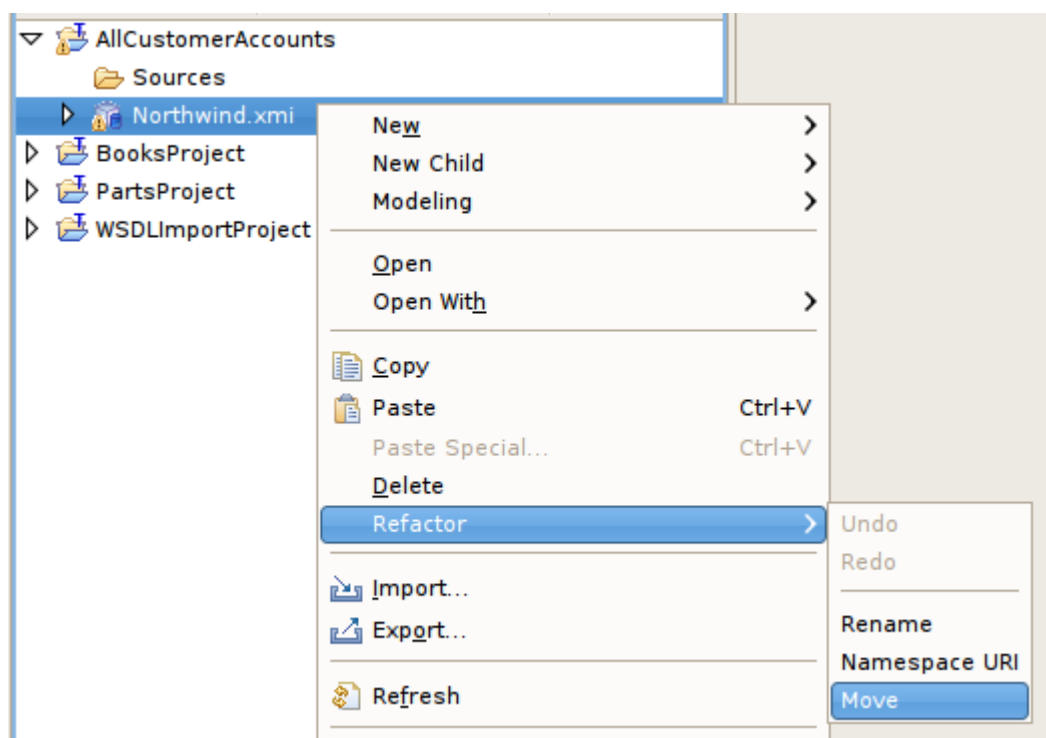


Figure 9.3. Refactor Move Action In Model Explorer

- **Step 3** - Select a new location (i.e. Project or Folder) and click **OK**.

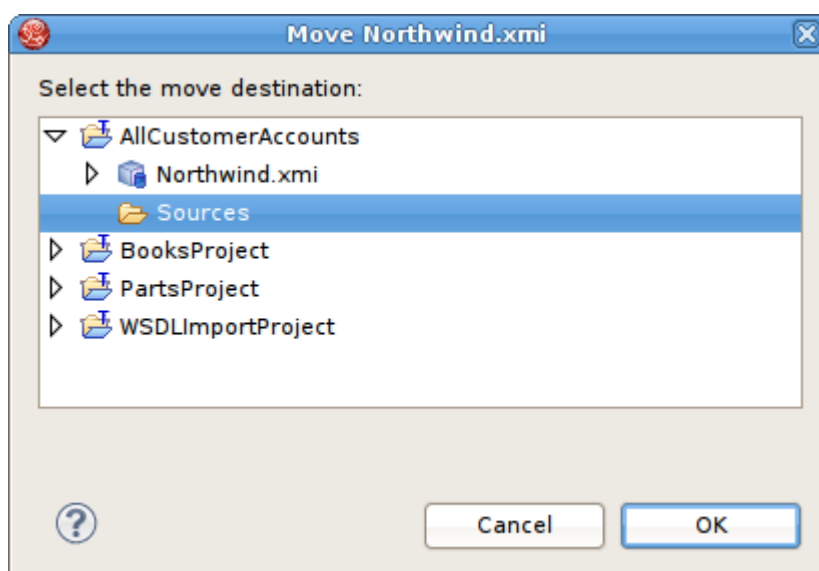


Figure 9.4. Move Model Dialog

9.3. Save Copy of Model

The **Save As...** action performs a similar function as the **Refactor > Rename** action except the renamed model is a structural copy of the original model.



Note

Each model object maintains its own unique ID, so copying a model will result in an exact structural copy of your original model but with re-generated unique object IDs. Be aware that locating and copying your models via your local file system may result in runtime errors within Designer. Each model is expected to be unique and duplicate models are not permitted.

- To create a duplicate model using **Save As...**:
 - **Step 1** - Open the model you wish to copy in a **Model Editor** by double-clicking the model in [Figure 4.2, “Model Explorer View”](#) or right-click select **Open** action.
 - **Step 2** - Select the editor tab for the model you opened.



Figure 9.5. Select Editor Tab

- **Step 3** - Select **File > Save As...** action to open the **Save Model As** dialog.

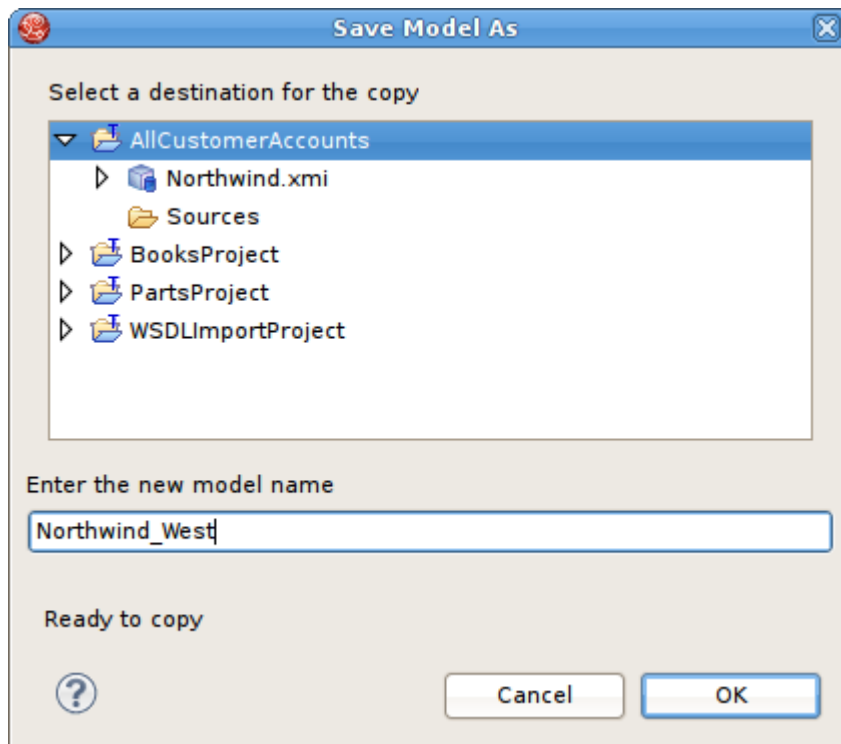


Figure 9.6. Save Model As Dialog

- **Step 4** - Enter a unique model name in the new model name text field and click **OK**.
- **Step 5** - If dependent models are detected, the **Save Model As - Import References** dialog is presented to give you the opportunity to change any of the dependent models imports to reference the new model or not.

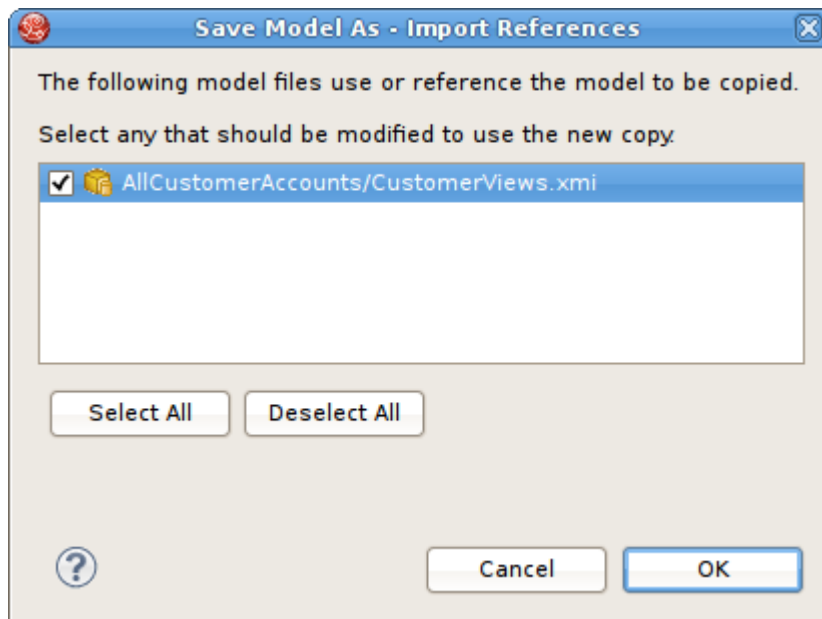


Figure 9.7. Save Model As Dialog

9.4. Clone Project

Because each instance of a model contains a unique ID and each object in each model contains a unique ID, copying a project is a delicate task. For this reason, the Clone Project action was created to manage the creation of exact structural copies of all models in the source project.

- The following lists specific rules and limitations for this action.
 - This action clones a complete model project containing any number of model (XMI or XSD) files organized in a user-defined directory structure.
 - All object references (UUIDs) within the original project will be replaced with new unique references.
 - Any model dependencies or internal object references are refactored to reflect the dependencies within the cloned project.
 - Any model references to models in projects external to the original project will NOT be replaced.
 - Only XMI and XSD files are cloned. All other file types in your project will NOT be processed nor copied into your newly cloned project including VDBs
 - If one or more editors that require "save" are open, the user will be asked to save them before continuing with the cloning process.
- To clone a model project::

- **Step 1** - Select an existing model project in the [Figure 4.2, “Model Explorer View”](#).
- **Step 2** - Right-click select the **Clone Project** action or select the **Project > Clone Project** action located in the Teiid Designer's main menu bar.

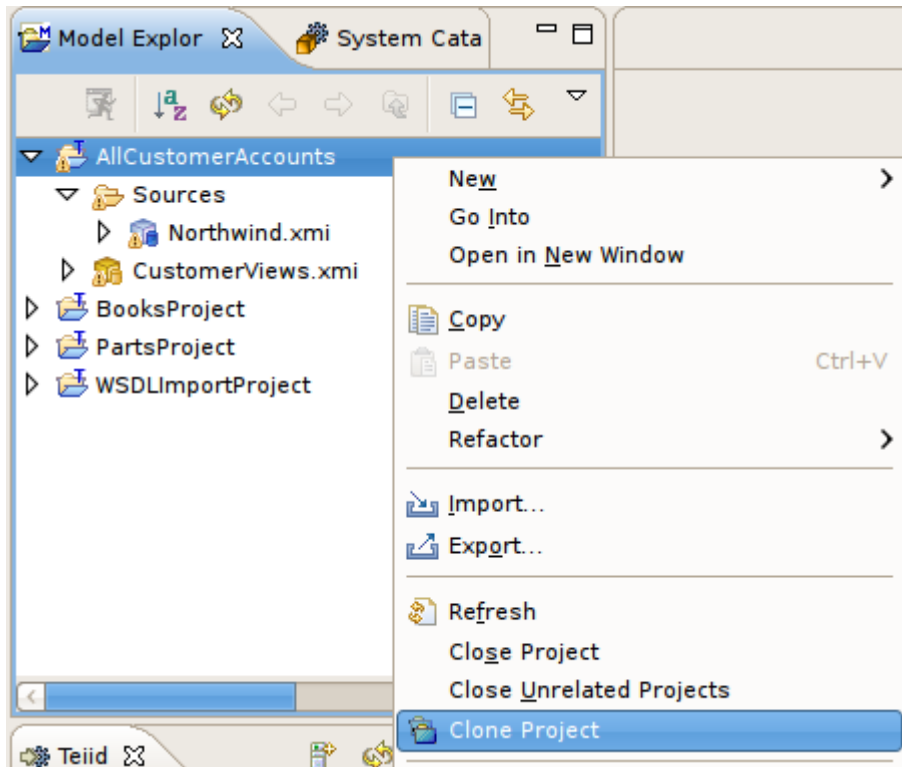


Figure 9.8. Clone Project In Context Menu

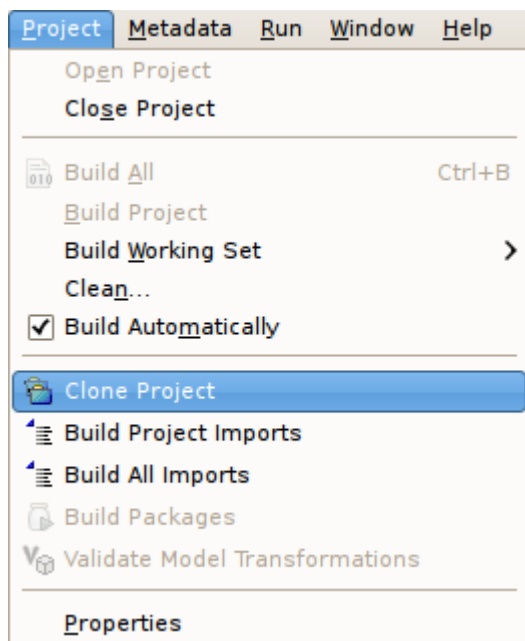


Figure 9.9. Clone Project In Project Menu

- **Step 3** - On the **Clone Project** wizard page, provide a name for your new project.

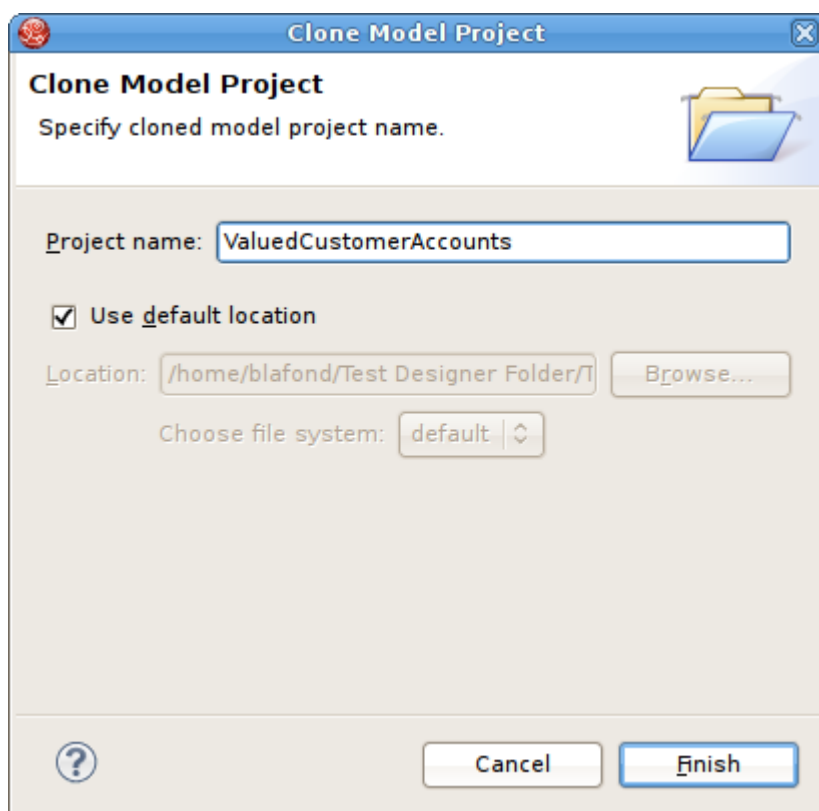


Figure 9.10. Clone Project In Project Menu

- **Step 4** - (Optional) If you wish to create your cloned project in a location other than your default workspace location, uncheck the **Use default location** check-box and specify (type in or browse to) a new directory location on your local file system.
- **Step 5** - Click **Finish** to generate your new project.

Testing Your Models

As described briefly in Chapter 1, you can test your models in Teiid Designer by using the Preview Data action



or test your models via your deployable VDB. These two options will be described in detail in this chapter as well as managing your required connection profiles.

10.1. Manage Connection Profiles

Teiid Designer utilizes the Eclipse Datatools' Connection Profile framework for connection management. Connection Profiles provide a mechanism to connect to JDBC and non-JDBC sources to access metadata for constructing metadata source models. Teiid Designer also provides a custom Teiid connection profile template designed as a JDBC source to a deployed VDB.

By selecting various Teiid Designer Import options, any applicable Connection Profiles you have defined in your Database Development perspective will be available to use as your import source. From these import wizards you can also create new connection profiles or edit existing connection profiles without leaving the wizard.

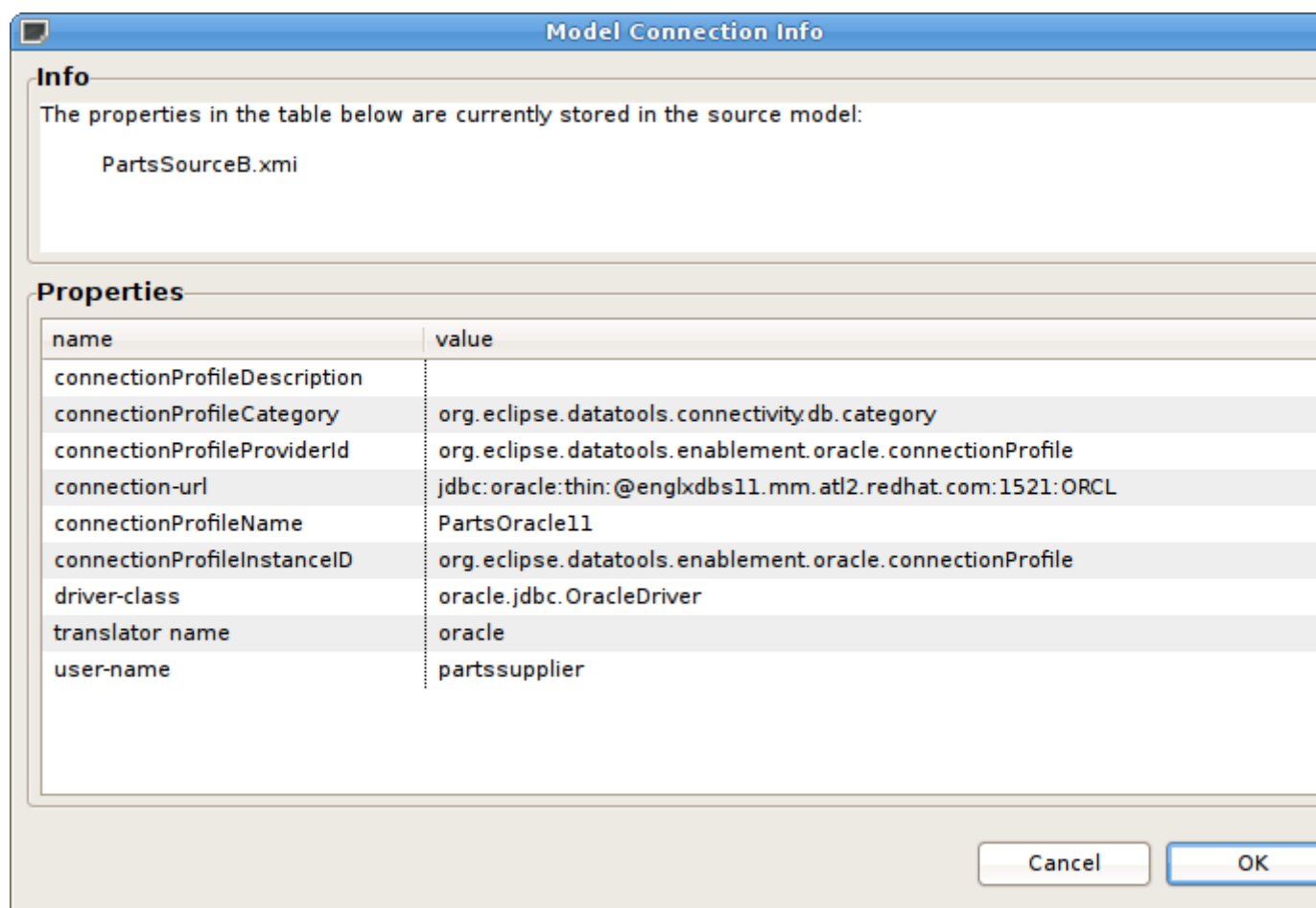
The [Section 4.3, “Teiid View”](#) provides access to running Teiid instances and shows data source and VDB artifacts deployed there. The "Create Data Source" action available on this view utilizes the available and applicable connection profiles.

10.1.1. Set Connection Profile for Source Model

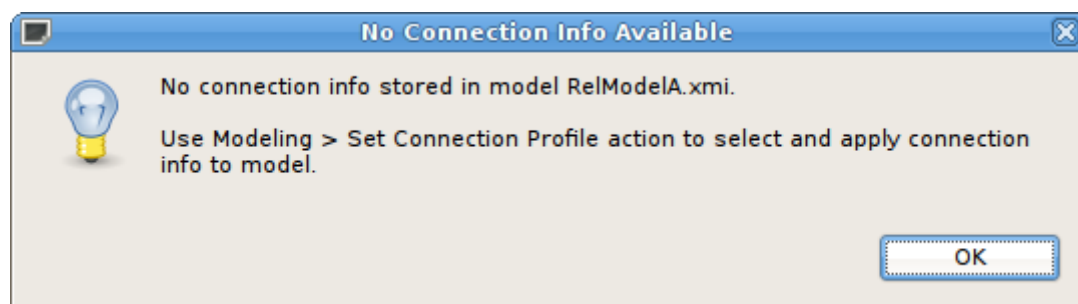
Teiid Designer integrates Data Tools Connection Profiles by persisting pertinent connection information in each source model. This can occur through Importing process or through the **Modeling > Set Connection Profile** action.

10.1.2. View Connection Profile for Source Model

In addition to setting the connection profile on a source model you can also view a source model's connection profile information via the **Modeling > View Connection Info** action which displays the detailed properties of the connection.

**Figure 10.1. Connection Profile Information Dialog**

Note that if a source model has no associated connection profile the following dialog will be displayed.

**Figure 10.2. No Connection Info Dialog**

10.1.3. Remove Connection Profile from Source Model

As a user, you may not want this connection information (i.e. URL, username, etc...) shared through your VDB. Designer provides a means to remove this connection information via a

Modeling > Remove Connection Info action. When adding a source model without connection information will require the user to supply or select the correct translator type.

10.2. Previewing Data For a Model

Designing and working with data is often much easier when you can see the information you're working with. The Designer's Preview Data feature makes this possible and allows you to instantly preview the information described by any object, whether it's a physical table or a virtual view. In other words, you can test the views with actual data by simply selecting the table, view, procedure or XML document. Previewing information is a fast and easy way to sample the data. Of course, to run more complicated queries like what your application likely uses, simply execute the VDB Via DTP and type in any query or SQL statement.

After creating your models, you can test them by using the **Preview Data** action



By selecting a desired table object and executing the action, the results of a simple query will be displayed in the [Figure 10.8, "SQL Results View"](#) view. This action is accessible throughout the Teiid Designer in various view toolbars and context menus.

There are two requirements for previewing your data: the selected object must be one of several previewable model object types and all source models within the model dependency tree must reference a connection profile. Model objects that can be previewed include: relational tables and views (including tables involving access patterns), relational procedures, Web service operations and XML document staging tables.

Note that any virtual table, view or procedure is previewable as long as all "physical" source models reference sufficient connection info. (See [Section 4.3, "Teiid View"](#) view)

After selecting the Preview Data action, Designer will insure that all source models are associated with connection profiles and that all required passwords are set.

If the model selected for preview is a source model and there is insufficient connection info for that model, the following dialog will be displayed and the action terminated.

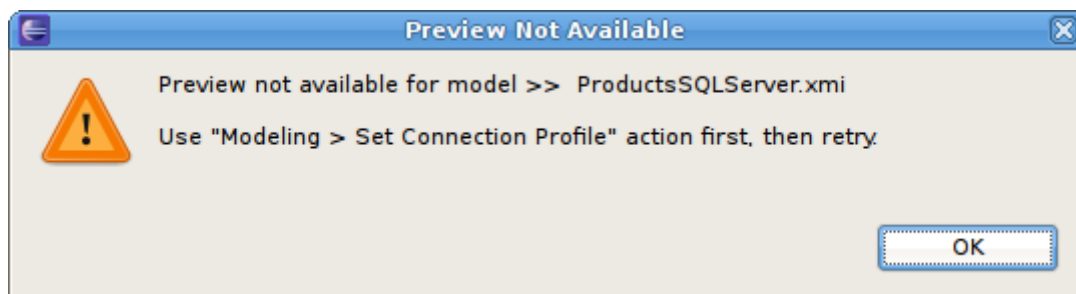


Figure 10.3. Preview Not Available

If any of the source models in the corresponding project require a password that can't be retrieved from an existing connection profile, the user will be queried for each missing password

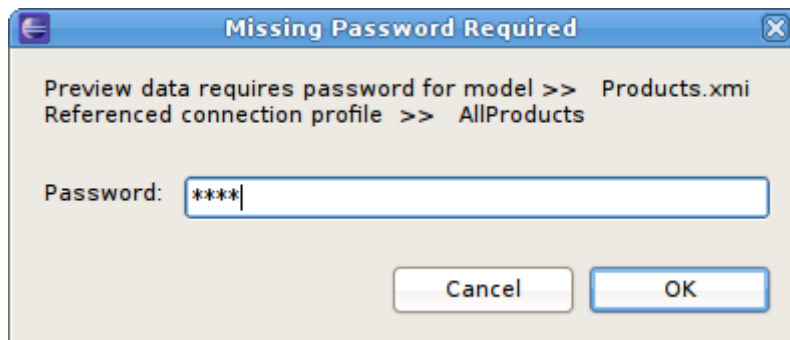


Figure 10.4. Missing Password

Testing Your Transformations


When editing transformation SQL in the Transformation Editor, a special **Preview Data** action is provided in the editor tool-bar




You can change your transformation SQL, re-validate and preview your the data for your modified SQL.

The following sections provide steps for previewing your data. Note that all steps assume that all source models referenced by your models, either directly or through dependencies, are bound to connector bindings.

10.2.1. Preview Relational Table or View

- To preview a relational table, relational view or staging table:
 - **Step 1** - Select a relational table or view in the [Section 4.1, “Model Explorer View”](#) or diagram. The table or view can be in a view model as well as a source model. Staging tables are not visible in the [Section 4.1, “Model Explorer View”](#), so you need to open the mapping diagram and select it there.
 - **Step 2** - Right-click select the **Preview Data** action  You can also select the same action in the tool-bar of either the [Section 4.1, “Model Explorer View”](#) or diagram.
 - **Step 3** - Your query results will be displayed in the [Figure 10.8, “SQL Results View”](#) view. The view will automatically open or get focus if not visible in your perspective.

10.2.2. Preview Relational Table With Access Pattern

- To preview a relational table or view with access pattern:
 - **Step 1** - Select a relational table or view in the [Section 4.1, “Model Explorer View”](#) or diagram that contains an access pattern. The table or view can be in a view model as well as a source model.
 - **Step 2** - Right-click select the **Preview Data** action 

You can also select the same action in the tool-bar of either the [Section 4.1, “Model Explorer View”](#) or diagram.
 - **Step 3** - A column input dialog is presented. Select each access pattern and enter a value for each required column. Note that if data entered does not match the column datatype (String, integer, etc...), an error message will be displayed in the dialog header. When all required values are entered, click the **OK** button to execute the query.

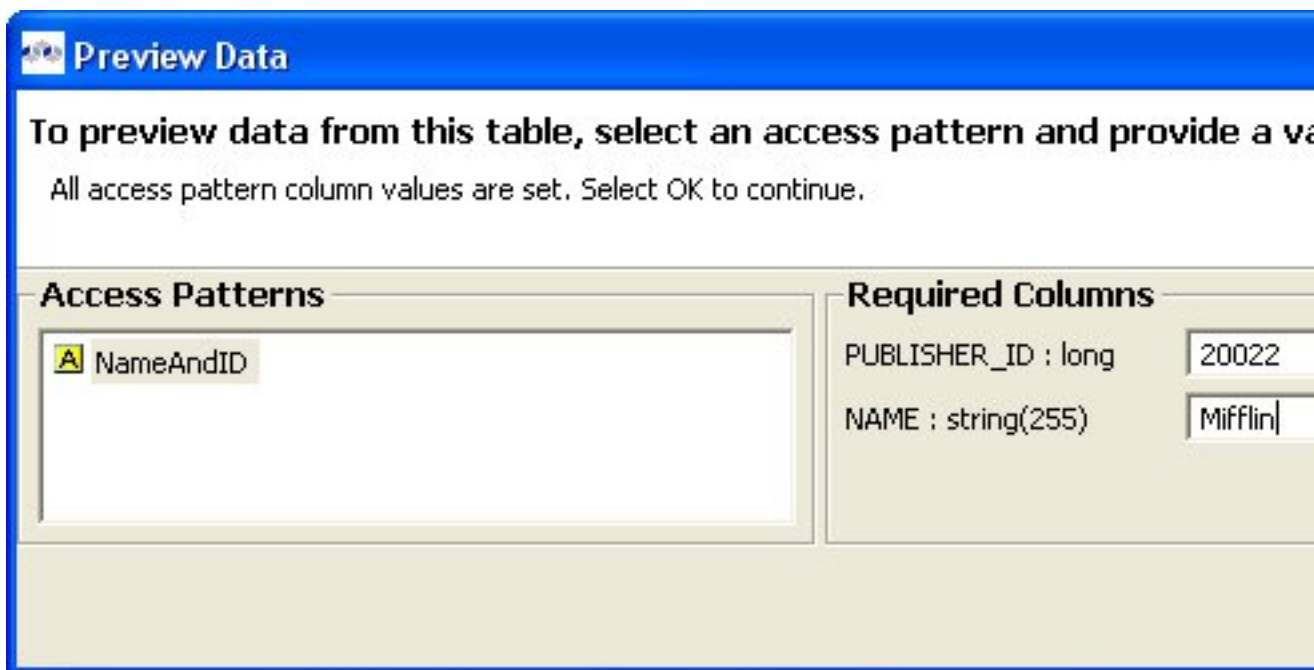


Figure 10.5. Access Pattern Column Input Dialog

- **Step 4** - Your query results will be displayed in the [Figure 10.8, “SQL Results View”](#) view. The view will automatically open or get focus if not visible in your perspective.

10.2.3. Preview Relational Procedure

- To preview a relational procedure:
 - **Step 1** - Select a relational procedure in the [Section 4.1, “Model Explorer View”](#) or diagram. The procedure can be in a view model as well as a source model.

- **Step 2** - Right-click select the **Preview Data** action



You can also select the same action in the tool-bar of either the [Section 4.1, “Model Explorer View”](#) or diagram.

- **Step 3** - An input parameter input dialog is presented. Enter a valid value for each parameter. Note that if data entered does not match the parameter datatype (String, integer, etc...), an error message will be displayed in the dialog header. When all required values are entered, click the **OK** button to execute the query.

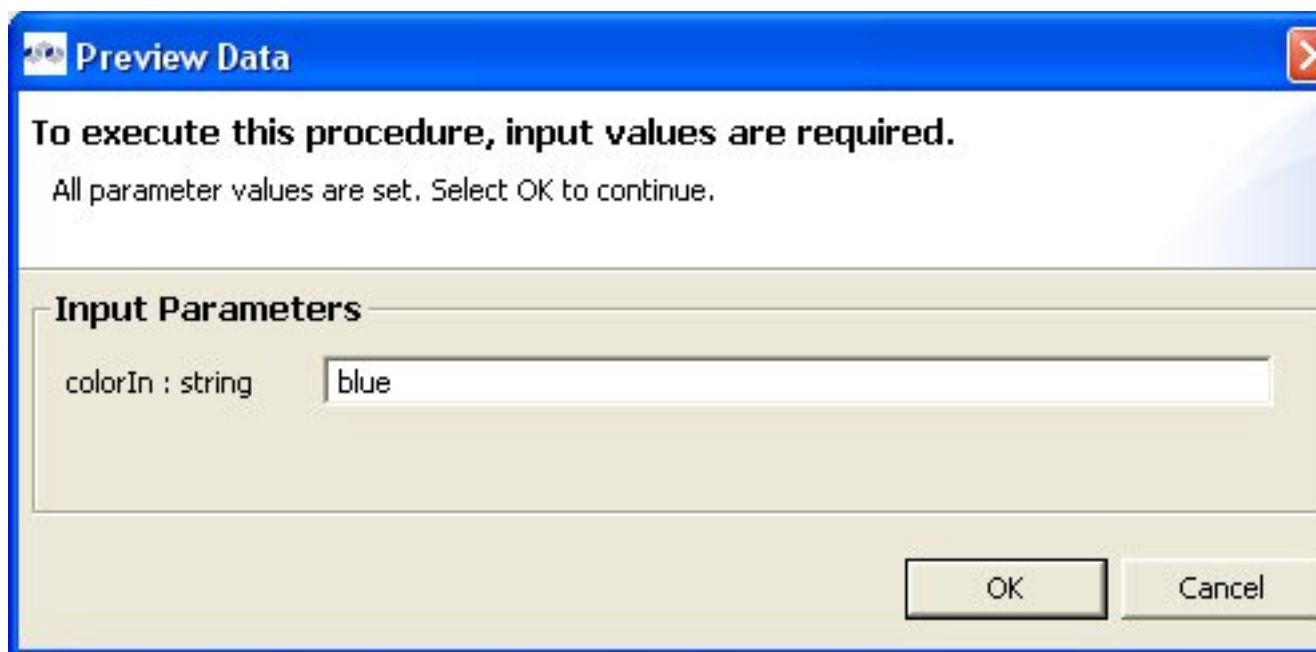


Figure 10.6. Procedure Parameter Input Dialog

- **Step 4** - Your query results will be displayed in the [Figure 10.8, “SQL Results View”](#) view. The view will automatically open or get focus if not visible in your perspective.

10.2.4. Preview Web Service Operation

- To preview a Web service operation:

- **Step 1** - Select a Web service operation in the [Section 4.1, “Model Explorer View”](#) or diagram. The operation can be in a view model as well as a source model.

- **Step 2** - Right-click select the **Preview Data** action



You can also select the same action in the tool-bar of either the [Section 4.1, “Model Explorer View”](#) or diagram.

- **Step 3** - An input parameter input dialog is presented. Enter a valid value for each parameter. Note that if data entered does not match the parameter datatype (String, integer, etc...), an error message will be displayed in the dialog header. When all required values are entered, click the **OK** button to execute the query.

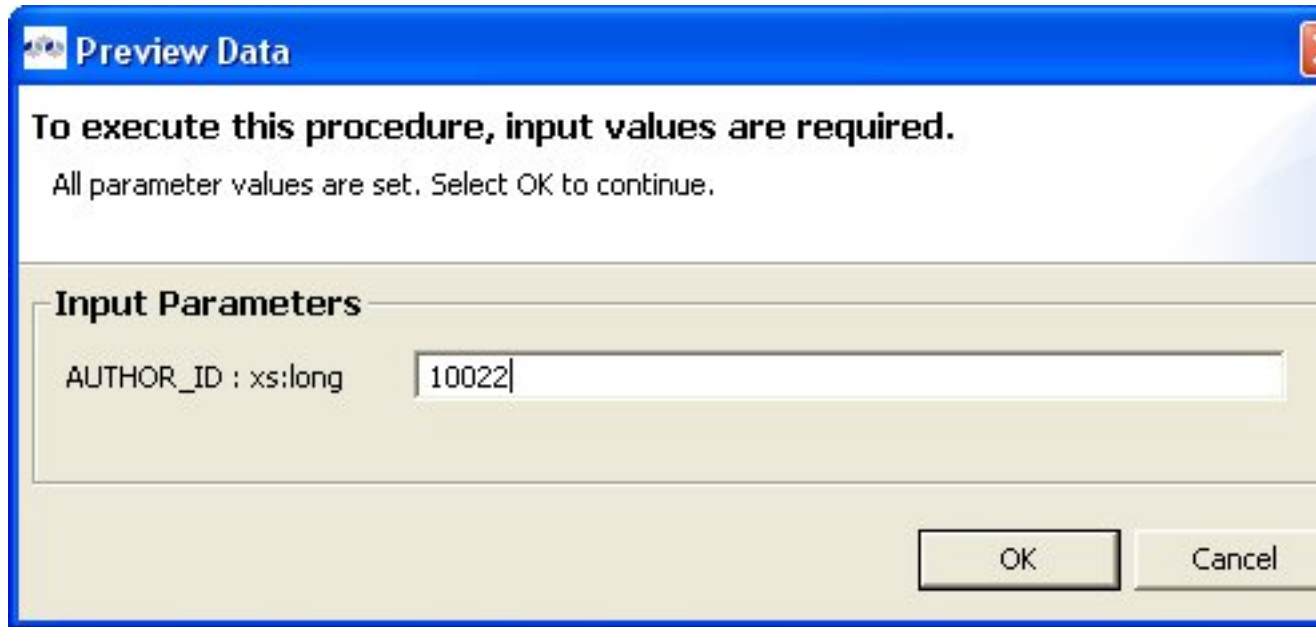


Figure 10.7. Procedure Parameter Input Dialog

- **Step 4** - Your query results will be displayed in the [Figure 10.8, "SQL Results View"](#) view. The view will automatically open or get focus if not visible in your perspective.

10.2.5. Sample SQL Results for Preview Data

Preview Data results are displayed in the Eclipse Datatools SQL Results view as shown below. Note there are a number of display preference and filter options for this view via toolbar buttons and the dropdown menu.

	INSTR_ID	NAME	TYPE
1	PRD01088	Novell Incorporated	Sto
2	PRD01089	Amazon.com, Incorporated	Sto
3	PRD01090	Juniper Networks, Incorporated	Sto
4	PRD01091	Red Hat, Incorporated	Sto
5	PRD01092	Boston Scientific Corporation	Sto
6	PRD01093	Inex Pharmaceuticals, Incorporated	Sto
7	PRD01094	Pfizer, Inc.	Sto
8	PRD01095	Cytovax Biotechnologies Incorporated	Sto
9	PRD01096	Commonwealth Biotechnologies	Sto
10	PRD01097	British Biotechnology plc	Sto
11	PRD01220	Unisys Corporation	Sto
12	PRD01099	Honeywell International	Sto
13	PRD01100	Hilton Hotels Corporation	Sto
14	PRD01101	Hilton Hotels Corporation	Co
15	PRD01102	Mercury Interactive Corporation	Sto
16	PRD01103	Fidelity Freedom Income Fund	Mu
17	PRD01104	Fidelity Freedom 2000 Fund	Mu

Figure 10.8. SQL Results View

10.3. Testing With Your VDB

In Teiid Designer you can execute a VDB to test/query actual data.

The requirements for VDB execution are:

- A deployed VDB backed by valid deployed Data Sources
- An instance of a Teiid Connection Profile configured for the deployed VDB

Teiid Designer simplifies this process via Deploy VDB and Execute VDB actions. Deploy VDB does just that, deploy a selected VDB to a running Teiid instance. Execute VDB performs the VDB deployment, creates a Teiid Connection Profile, opens the Database Development perspective and creates a connection to your VDB.

10.3.1. Creating Data Sources

The mechanism by which VDBs are able to query actual data sources is the Data Source. These are deployed configurations backed by database or source connection jars. Each source model referenced within a VDB requires a JNDI name representing a deployed Data Source.

When creating VDBs you do not need to have deployed data sources on your Teiid server, but if you wish to test your VDB, the data sources need to be present.

Teiid Designer provides a **Create Data Source** action so you can create compatible data sources for your source model. If you wish to create a data source for a specific model you can select that source model in your workspace and select the **Modeling > Create Data Source** action. This will extract the connection profile data from your source model and create a corresponding data source on your default Teiid server.

You can also create data sources from the Teiid view. Select a Teiid server instance in the Teiid view and right-click select the **Create Data Source** action. This will launch the Create Data Source Dialog shown below.

Create Data Source

All inputs are valid. Select Finish to create data source.

Teiid Server: mm://localhost:31443

Name: BooksDB2

Connection Source

☐ Use Model Connection Info

Model:

☒ Use Connection Profile Info

Connection Profile: BooksDB2 New... Edit...

Connection Properties

Name	Value
password	*****
user-name	books
connection-url	jdbc:db2://db000255.org.mydbs.com:50000
driver-class	com.ibm.db2.jcc.DB2Driver

? Cancel Finish

Figure 10.9. Create Data Source Dialog

You can either select an existing Connection Profile from the drop-down list (Use Connection Profile Info option) or check the Use Model Info option and select an existing source model containing connection info.

After creating your new data source it should now be shown in the **Data Sources** folder of the corresponding Teiid server.

10.3.2. Execute VDB from Model Explorer

- If you have a Teiid instance defined and connected in your Teiid view you can:
 - **Step 1** - Right-click a VDB in your Model Explorer select **Modeling > Execute VDB** action. This action will insure your selected VDB is deployed to Teiid, create a Teiid Connection Profile specific for that VDB, open the Database Development perspective and create a connection to your VDB.



Figure 10.10. Execute VDB Action

- **Step 2** - Select your new Teiid connection profile and right-click select **Open SQL Scrapbook**, enter your designer SQL (i.e. `SELECT * FROM TableXXXX`), select all text and right-click select **Execute Selected Text**

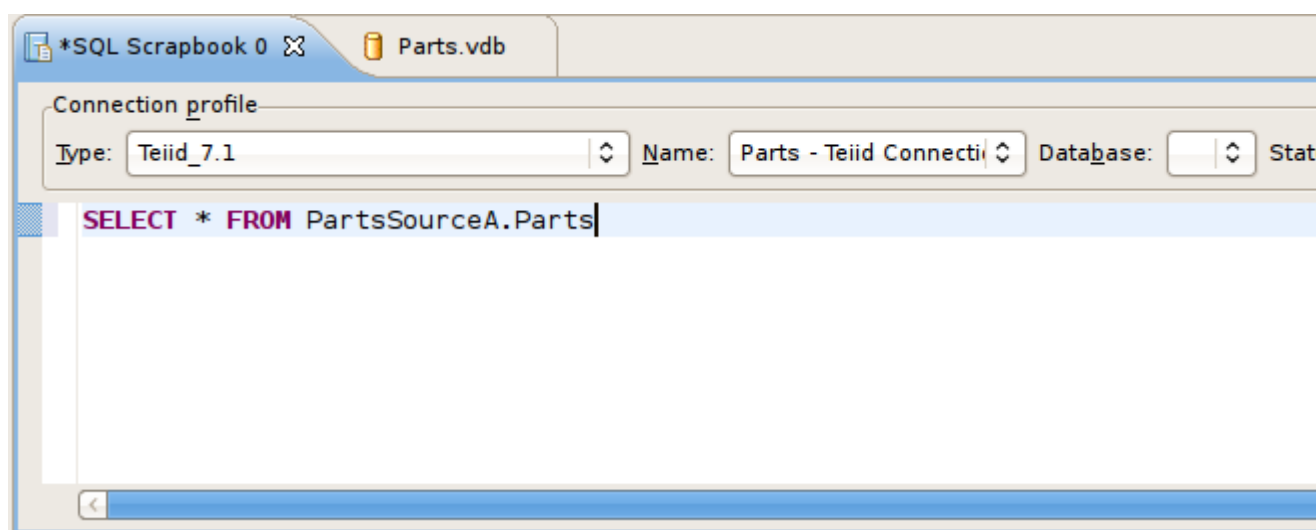


Figure 10.11. SQL Scrapbook Editor

- **Step 3** - Results of query should be displayed in the SQL Results view on the **Result1** tab.

Status	Operation	Date	Connection Profile
✓ Succes	SELECT * FROM PartsSourceA.Parts	Oct 15, 2010	Parts - Teiid Connection

	PART_ID	PART_NAME	PART_COLOR	PART
1	P300	Nut	Red	12
2	P301	Bolt	Green	12
3	P302	Screw	Blue	13
4	P303	Bolt	Green	17
5	P304	Cam	Green	18

Figure 10.12. SQL Results View

10.3.3. Deploy VDB from Model Explorer

You can also deploy your VDB first by selecting it in the Model Explorer and dragging/dropping it onto a connected Teiid instance in the Teiid view, or right-click select **Modeling > Deploy** action.

Once deployed, you can select the VDB in the Teiid View and right-click select the **Execute VDB** action there. This will create a Teiid Connection Profile specific for that VDB, open the Database Development perspective and create a connection to your VDB. Continue with Step's 2 and 3 above.

Note that if you do not have a Teiid instance defined or your default Teiid instance is disconnected, the following dialog will be displayed if the **Modeling > Deploy** action is launched.

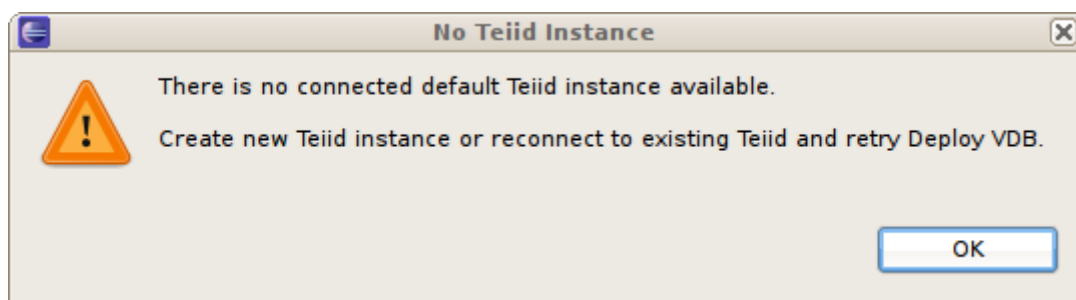


Figure 10.13. No Teiid Instance Defined

10.3.4. Executing a Deployed VDB

To execute a VDB, that's been deployed manually, follow the steps below:

- To execute a VDB, that's been deployed manually, follow the steps below:

- **Step 1** - Open the **Database Development** perspective.
- **Step 2** - Select the **Database Connections** folder and choose the **New** action to display the **New Connection Profile** dialog.

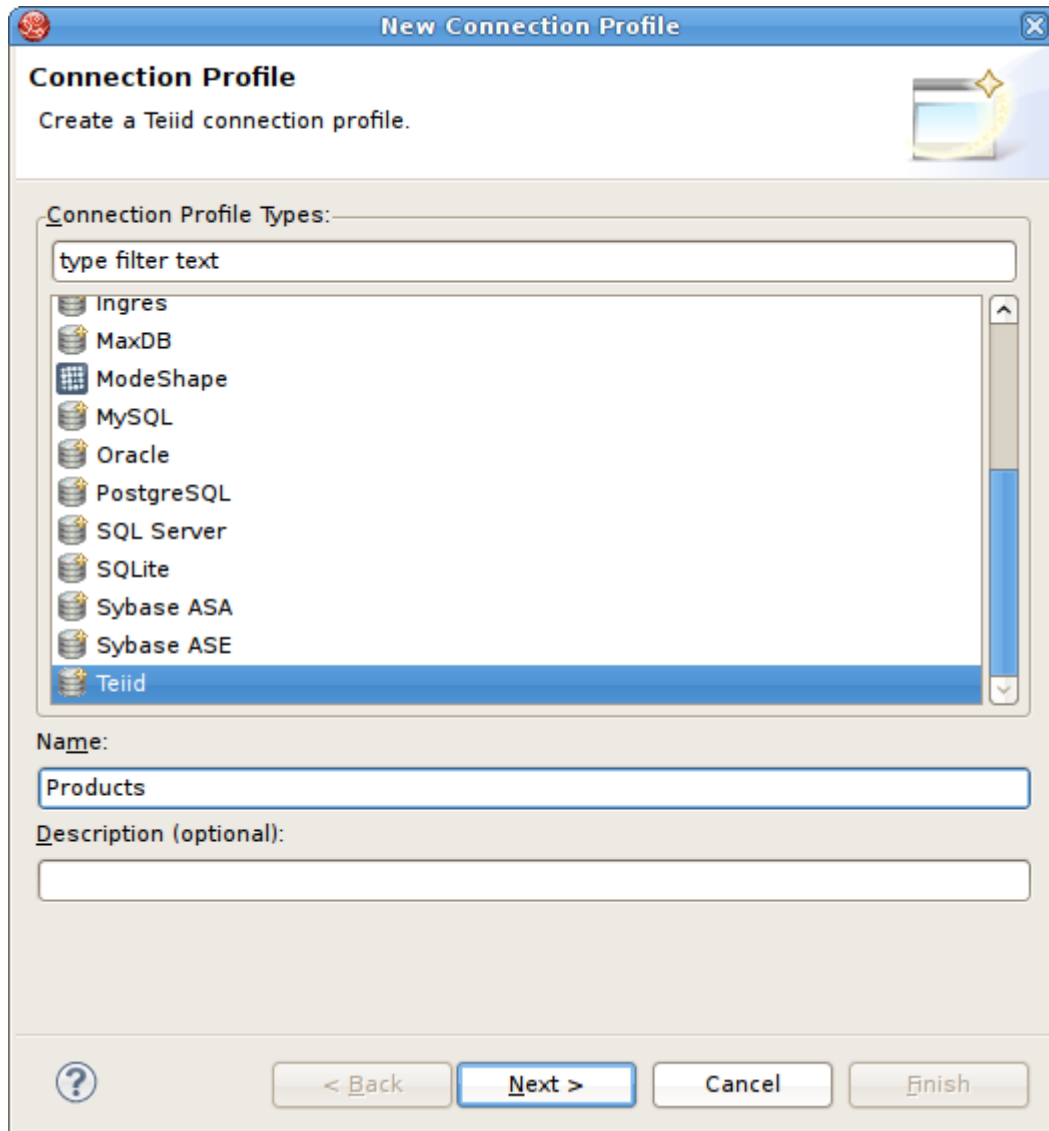


Figure 10.14. New Connection Profile Dialog

- **Step 3** - Enter unique name for your profile, select an existing connection profile type and hit **Next**.
- **Step 4** - In the **Teiid Profile Wizard** page, select the **New Driver Definition** button to locate and select the Teiid client jar on your file system. Configure your URL using your VDB Name, Host, Port, Username (default = "admin") and Password (default = "teiid").

The screenshot shows the 'Teiid Profile Wizard' dialog box. The title bar reads 'Teiid Profile Wizard'. The main heading is 'Specify a Driver and Connection Details'. Below this, a text box says 'Select a driver from the drop-down and provide login details for the connection.' To the right of this text is a small icon of a folder with a star. Below the text is a 'Drivers:' label followed by a drop-down menu showing 'Teiid Server JDBC Driver'. To the right of the drop-down are two small icons: a green plus sign and a blue triangle. Below the 'Drivers:' section is a 'Properties' section with two tabs: 'General' (selected) and 'Optional'. The 'General' tab contains several text boxes: 'VDB Name' with 'Products', 'Host' with 'localhost', 'Port' with '31000', 'Username' with 'admin', and 'Password' with a masked password (seven dots). Below these text boxes are two checkboxes: 'SSL Connection' (unchecked) and 'Save password' (checked). Below the 'Save password' checkbox is a 'Connection URL' label followed by a text box. At the bottom of the 'Properties' section is a 'Test Connection' button. Below the 'Properties' section are two checkboxes: 'Connect when the wizard completes' (checked) and 'Connect every time the workbench is started' (unchecked). At the bottom of the dialog are four buttons: a question mark icon, '< Back', 'Next >', 'Cancel', and 'Finish'.

Figure 10.15. Teiid Connection Profile Dialog

- **Step 5** - Select **Next** to view a summary of your new Teiid Connection Profile.

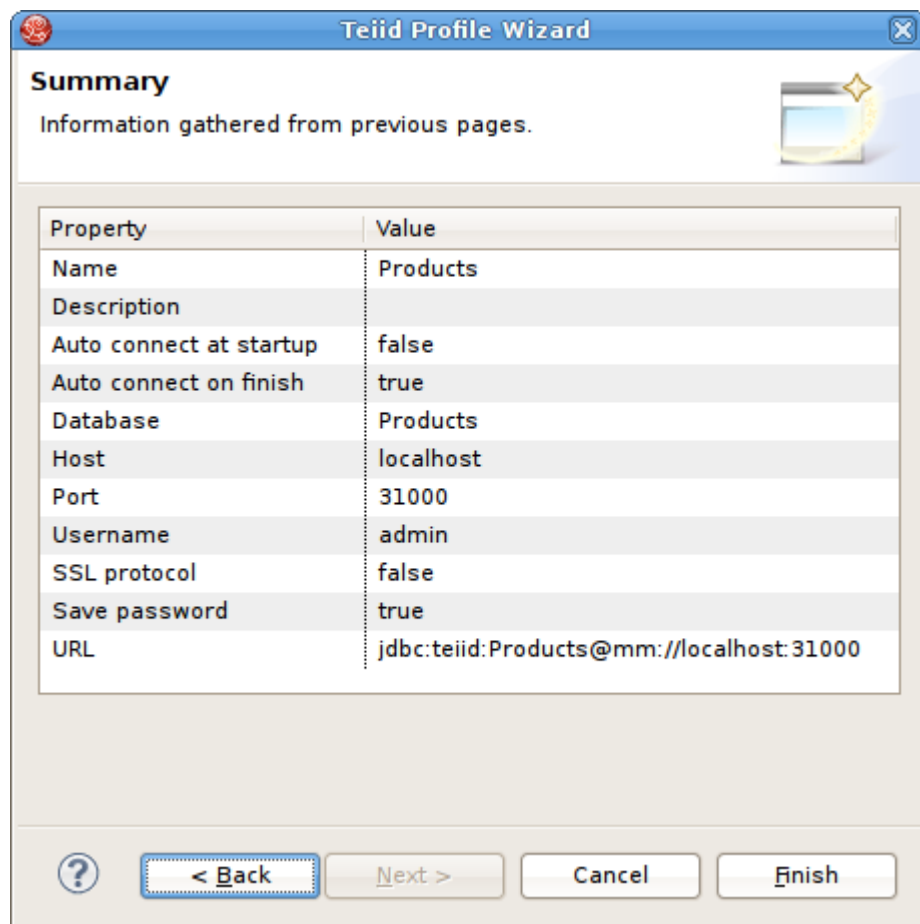


Figure 10.16. Teiid Connection Profile Summary

- **Step 6** - Select **Finish**.
- **Step 7** - Select your new Teiid connection profile and right-click select **Open SQL Scrapbook**, enter your designer SQL (i.e. `SELECT * FROM TableXXXX`), select all text and right-click select **Execute Selected Text**.

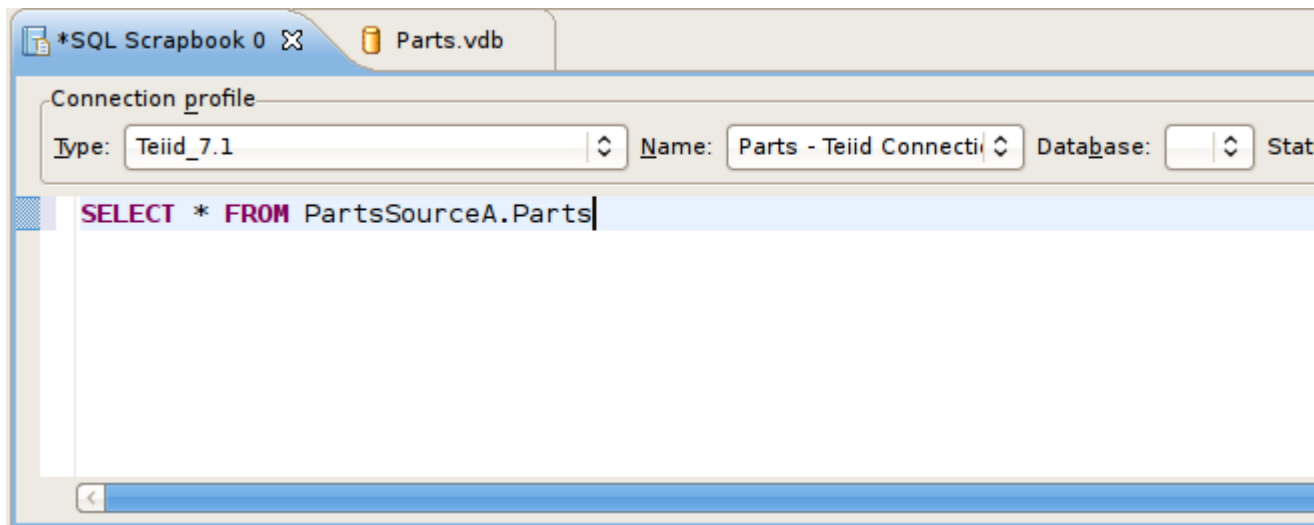


Figure 10.17. SQL Scrapbook Editor

- **Step 8** - Results of query should be displayed in the SQL Results view on the **Result1** tab.

The screenshot shows the SQL Results view. At the top, there is a tab labeled 'SQL Results'. Below the tab, there is a text input field with the placeholder 'Type query expression here'. Below the input field, there is a table with the following columns: 'Status', 'Operation', 'Date', and 'Connection Profile'. The table contains one row with a green checkmark in the 'Status' column, 'Succes SELECT * FROM PartsSourceA.Parts' in the 'Operation' column, 'Oct 15, 2010' in the 'Date' column, and 'Parts - Teiid Connection' in the 'Connection Profile' column. Below the table, there is a scrollable area containing a table with the following columns: 'PART_ID', 'PART_NAME', 'PART_COLOR', and 'PART'. The table contains five rows of data:

	PART_ID	PART_NAME	PART_COLOR	PART
1	P300	Nut	Red	12
2	P301	Bolt	Green	12
3	P302	Screw	Blue	13
4	P303	Bolt	Green	17
5	P304	Cam	Green	18

Figure 10.18. SQL Results View

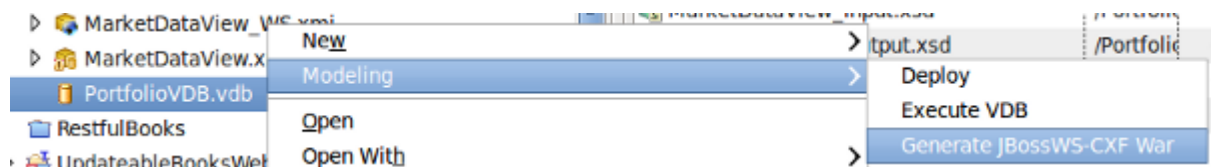
Web Service Wizards

Teiid Designer allows you to expose your VDBs via a SOAP or REST interface. JBossWS-CXF or RESTEasy wars can be generated based on models within your VDBs. This section describes these wizards in detail.

11.1. Generating a JBossWS-CXF War

The Teiid Designer provides web service generation capabilities in the form of a JBossWS-CXF war. Once you have added your **Web Service Models** as described in [Section 7.5, “Creating Web Service View Model”](#) to your **VDB**, deployed the **VDB** to a running Teiid instance and created your VDB's data source, you are ready to expose the web service using the generated war.

- To generate a new JBossWS-CXF war using the VDB:
 - **Step 1** - Right-click on the VDB containing your web service model(s) and select the **Modeling > Generate JBossWS-CXF War** action.



- **Step 2** - Fill in missing properties in **Web Service War Generation Wizard** shown below.

Create Web Service WAR File

Create a WAR file to deploy as a Web Service

Enter the required information, then click OK to create the WAR file.

WAR Creation Information

Name: PortfolioVDB

Host: localhost

Port: 8080

Connection JNDI Name: java:PortfolioVDB

Security Options

When using HTTPBasic security, a local Teiid connection is required using the PassthroughAuthentication property.

☒ None

☐ HTTPBasic

☐ WS-Security (Username Token)

HTTPBasic Options

Security Realm:

Security Role:

WS-Security Username:

WS-Security Password:

Target namespace: http://teiid.org

WAR File Save Location: /home/tejones/WARFiles

Change...

Restore Default

Cancel OK

Figure 11.1. Generate a JBossWS-CXF War Web Service Dialog

Table 11.1. Field Descriptions

Field Name	Description
Name	The name of the generated war file.
Host	The server host name (or IP).
Port	The server port.
Connection JNDI Name	The JNDI connection name to the deployed Teiid source VDB.

Field Name	Description
Security options	<ul style="list-style-type: none"> • None - no username/password required to connect to the VDB through the generated web service. • HTTP Basic - the specified security realm and role will be used. The default realm value is the realm that comes out-of-the-box with Teiid (teiid-security). The role needs to be defined in the appropriate security mechanism. In the case of Teiid, use the teiid-security-roles.properties file. When using HTTPBasic, a local Teiid connection using the PassthroughAuthentication property is required. See the Teiid user's manual for details on PassthroughAuthentication. • WS-Security - a password callback class will be generated for you which will validate that the username/password values you specified in the war generator dialog are passed in. This is meant to be a testing mechanism for your WS-Security enabled web service and your own security mechanism should be implemented in this class. All source code is included in the generated war along with the compiled class files.
Target namespace	This is the target namespace that will be used in the generated WSDL and subsequent generated web service classes.
War File Save Location	The folder where the generated WAR file should be saved.

- **Step 3** - Click **OK** to generate the web service war. When war generation is complete, a confirmation dialog should appear. Click **OK**.

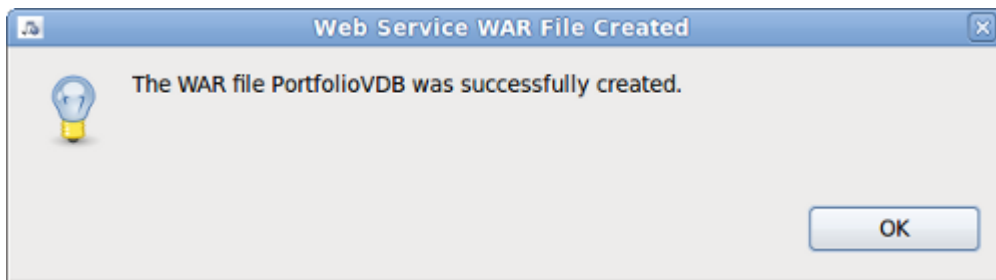
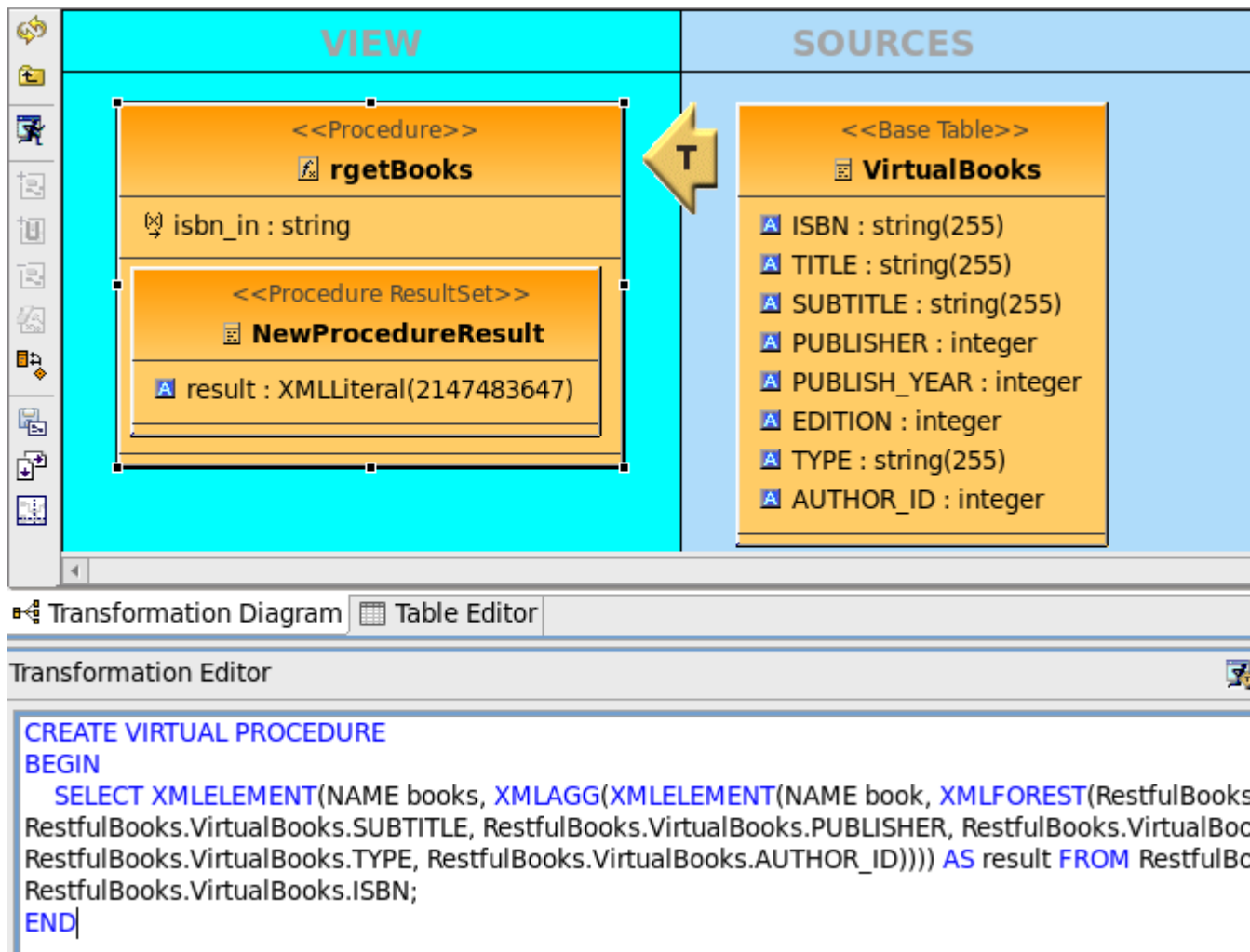


Figure 11.2. Generation Completed Dialog

11.2. Generating a RESTEasy War

In Teiid Designer, it is also possible to expose your VDBs over REST using a generated RESTEasy war. Also, if your target virtual model has update, insert and delete SQL defined, you can easily provide CRUD capabilities via REST. Accepted inputs into the generated REST operations are URI path parameters and/or XML. The only output at this time is an XML document.

- **Step 1** In a virtual model, add a procedure(s) that returns an XMLLiteral object. The target of your procedure can be any models in your VDB. Here is an example procedure that selects from a virtual table (VirtualBooks) and returns the results as an XMLLiteral:



Notice the syntax used to convert the relation table result of the select from VirtualBooks, to an XMLLiteral. All XML functions are documented in the **Scalar Functions** chapter of the **Teiid Reference Guide**.

Here is an example of an update procedure that will insert a row and return an XMLLiteral object:

The screenshot shows the Transformation Editor interface. The top section is divided into two panes: **VIEW** (left) and **SOURCES** (right). A yellow arrow labeled 'T' points from the **SOURCES** pane to the **VIEW** pane.

VIEW Pane:

- Procedure: **rputBook** (type: <<Procedure>>)
- Inputs: isbn_in : string, title_in : string, subtitle_in : string, publisher_in : int, publish_year_in : int, edition_in : int, type_in : string, author_id_in : int
- Procedure ResultSet: **NewProcedureResult** (type: <<Procedure ResultSet>>)
- Output: result : XMLLiteral

SOURCES Pane:

- Table: **VirtualBooks** (type: <<Base Table>>)
- Columns: ISBN : string(255), TITLE : string(255), SUBTITLE : string(255), PUBLISHER : integer, PUBLISH_YEAR : integer, EDITION : integer, TYPE : string(255), AUTHOR_ID : integer

Below the panes are two tabs: **Transformation Diagram** and **Table Editor**. The **Table Editor** tab is active, showing the following SQL code:

```

CREATE VIRTUAL PROCEDURE
BEGIN
  DECLARE integer VARIABLES.insert_count = 0;
  INSERT INTO RestfulBooks.VirtualBooks (RestfulBooks.VirtualBooks.ISBN, RestfulBooks.VirtualBooks.TITLE, RestfulBooks.VirtualBooks.SUBTITLE, RestfulBooks.VirtualBooks.PUBLISHER, RestfulBooks.VirtualBooks.PUBLISH_YEAR, RestfulBooks.VirtualBooks.EDITION, RestfulBooks.VirtualBooks.TYPE, RestfulBooks.VirtualBooks.AUTHOR_ID)
  VALUES (RestfulBooks.rputBook.isbn_in, RestfulBooks.rputBook.title_in, RestfulBooks.rputBook.subtitle_in, RestfulBooks.rputBook.publisher_in, RestfulBooks.rputBook.publish_year_in, RestfulBooks.rputBook.edition_in, RestfulBooks.rputBook.type_in, RestfulBooks.rputBook.author_id_in);
  VARIABLES.insert_count = VARIABLES.ROWCOUNT;
  IF(VARIABLES.insert_count = 1)
  BEGIN
    SELECT XMLCOMMENT('Insert was successful!') AS result;
  END
  ELSE
  BEGIN
    SELECT XMLCOMMENT('Insert failed!') AS result;
  END
END

```

The input format for the REST procedure could be URI parameters, an XML document, or some combination of both. When using an XML document your root node should be **<input>** and the XML nodes should correspond to order of the procedure's input parameters. For example, here is the input for the above insert procedure:

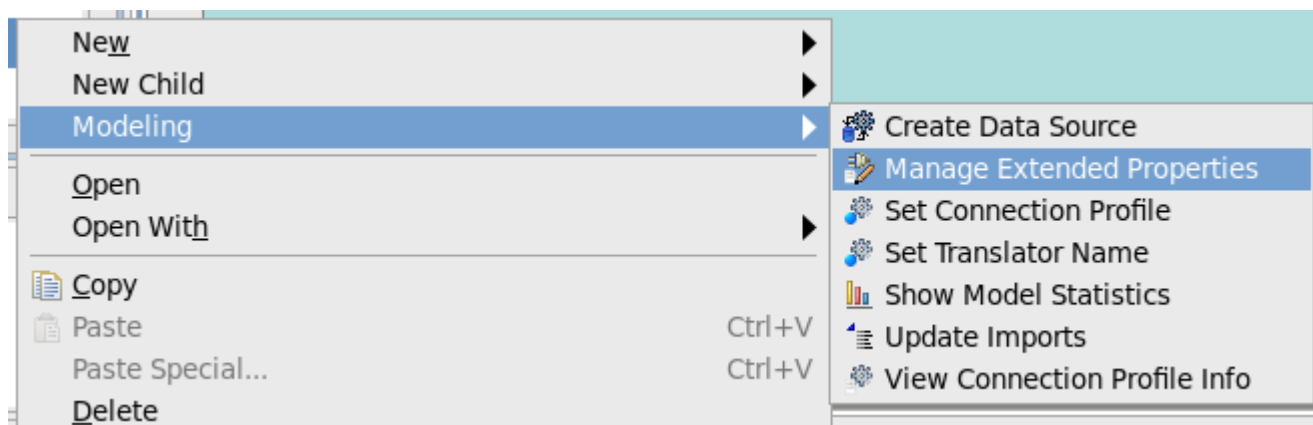

```

<input>
<ISBN>0-13-014714-1-99999</ISBN>
<TITLE>The XML Handbook</TITLE>
<SUBTITLE>Updated Edition</SUBTITLE>
<PUBLISHER>16</PUBLISHER>
<PUBLISH_YEAR>2000</PUBLISH_YEAR>
<EDITION>2</EDITION>
<TYPE>Hardback</TYPE>
<AUTHOR_ID>49</AUTHOR_ID>
</input>

```

Figure 11.3. Sample XML Input

- **Step 2** - Now we need to identify our procedure as REST eligible. To do this we add two extended properties to the procedure via the **Modeling->Manage Extended Properties** context menu option.

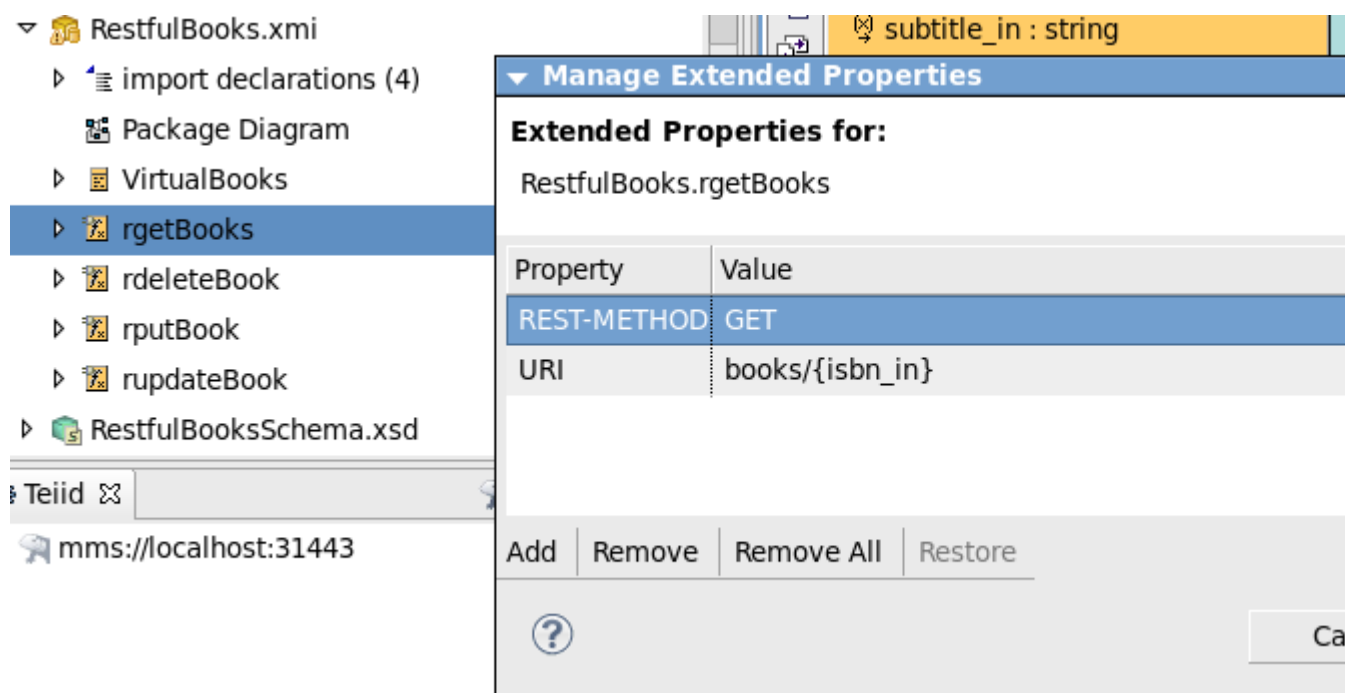


The two required properties are defined in the table below:

Table 11.2. Required Extended Properties for RESTful Procedures

Property Name	Description
REST-METHOD	The HTTP method that will determine the REST mapping of this procedure. Supported methods are: GET, PUT, POST and DELETE
URI	The resource path to the procedure. For example, if you use "books/{isbn}" as your URI value for a procedure, http://{host}:{port}/{war_context}/{model_name}/books/123 would execute this procedure and pass 123 in as a parameter.

Here's what the above example would look like in the Extended Properties dialog:



Note that the generated URI will have the model name included as part of the path, so full URL would look like this: **http://{host}:{port}/{war_context}/{model_name}/books/123**. If you wanted a REST service to return all books, you would write your procedure just as it is above, but remove the input parameter. The URI property would then just be **'books'** (or whatever you want) and the URL would be **http://{host}:{port}/{war_context}/{model_name}/books**.

Once you have added all of your procedures along with the required extended properties, be sure and add the model to your VDB or synchronize if it's already included in the VDB. You will then need to re-deploy the VDB.

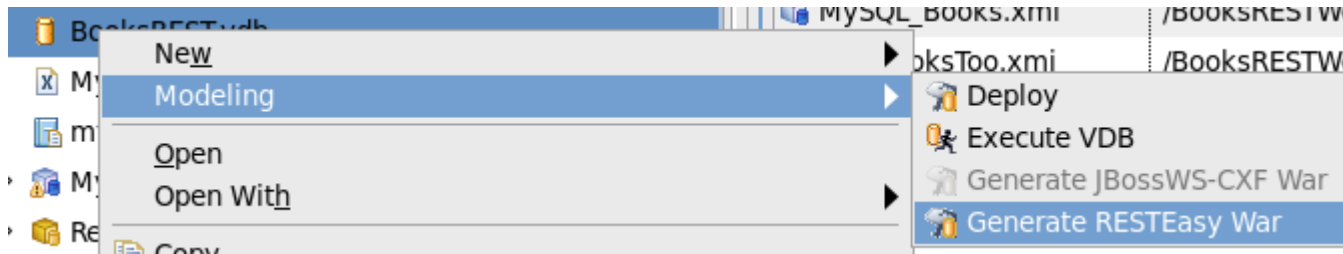


Tip

*If you redeploy your VDB during development, you may receive an **"Invalid Session Exception"** due to a stale connection obtained for the pool. This can be corrected by flushing the data source or, alternatively, you could add a test query to your VDB connection's `-ds.xml` file. This will insure you get a valid connection after redeploying your VDB. The syntax for the test query is as follows: **<check-valid-connection-sql>some arbitrary sql</check-valid-connection-sql>**"*

- **Step 3** - 3. If you have not already done so, you will need to create a data source for your VDB. This can be done in the Teiid View of Designer. Right-click on your deployed VDB and select Create Data Source. The Generate REST WAR dialog will ask you for the **JNDI name** for your created source so that it can connect to your VDB.

- **Step 4** - Right-click on the VDB containing your virtual model(s) with REST eligible procedures and select the **Modeling > Generate RESTEasy War** action. If there are no procedures that are REST eligible, the "Generate RESTEasy War" option will not be enabled.



- **Step 5** - Fill in missing properties in the **REST War Generation Wizard** shown below.

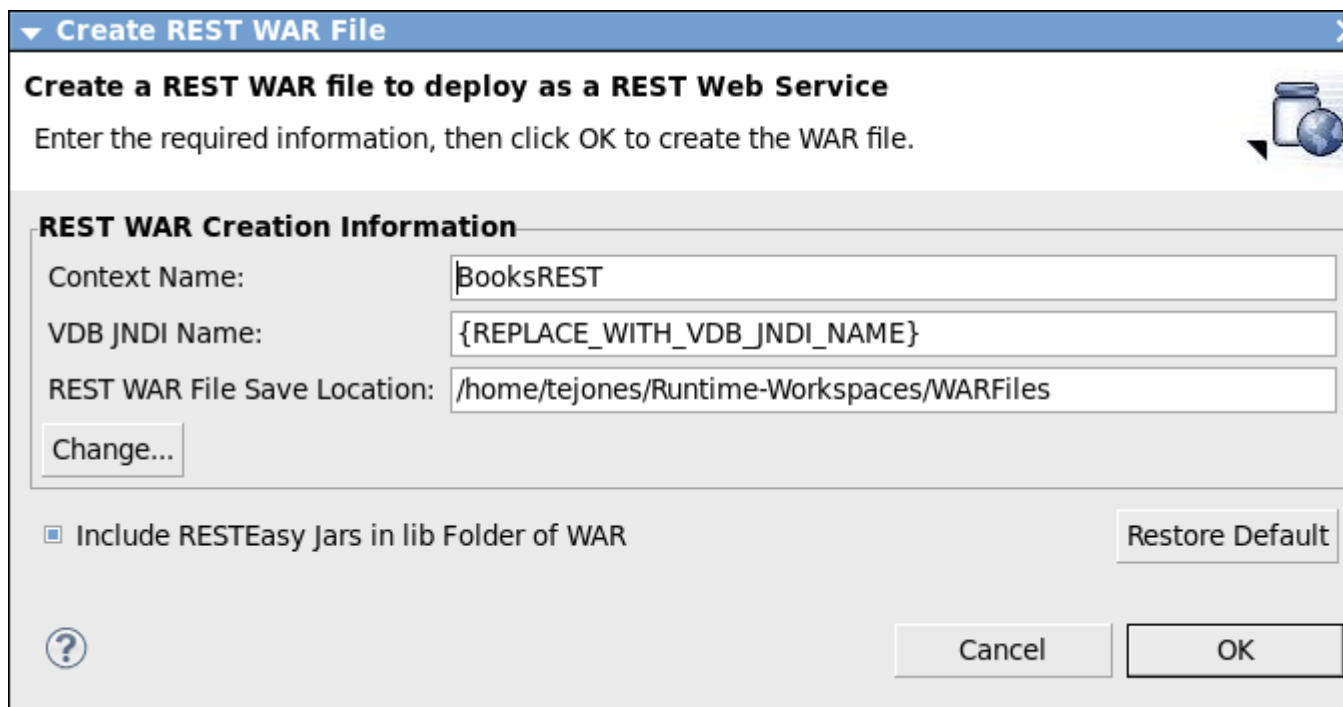


Figure 11.4. Generate a REST WAR War File Dialog

Table 11.3. Field Descriptions

Field Name	Description
Name	The name of the generated war file.
Connection JNDI Name	The JNDI connection name to the deployed Teiid source VDB.
War File Save Location	The folder where the generated WAR file should be saved.
Include RESTEasy Jars in lib Folder of WAR	If selected, the RESTEasy jars and there dependent jars will be included in the lib foled

Field Name	Description
	of the generated WAR. If not selected, the jars will not be included. This should be de-selected in environments where RESTEasy is installed in the classpath of the server installation to avoid conflicts.

- **Step 6** - Click **OK** to generate the REST war. When war generation is complete, a confirmation dialog should appear. Click **OK**.

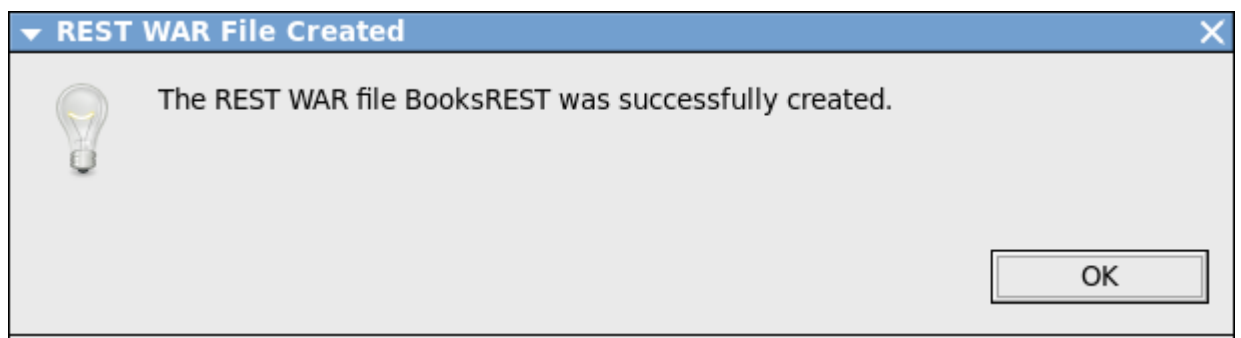


Figure 11.5. Generation Completed Dialog

Searching

Designer provides multiple search actions located via Teiid Designer sub-menu in Eclipses Search menu. **Search** menu.

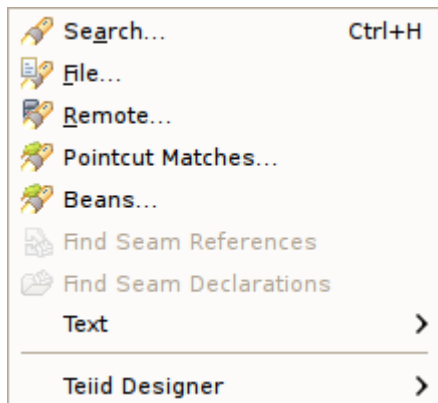


Figure 12.1. Search Options

- The individual actions in the Teiid Designer sub-menu are described below:



Relationship... - Launches the Search dialog and auto-selects the Relationships tab. User can search for models in the workspace by specifying a relationship type, participant locations, and/or names containing specified text. Search results appear in the [Section 4.8, "Search Results View"](#) view, and double-clicking a result will open that model in the appropriate editor.



Transformations... - Launches the Transformation Search dialog. User can search models in the workspace for matching SQL text. Search results appear in the dialog and user can select and view SQL as well as open desired transformations for editing.




Metadata... - Launches the Search dialog. User can search for models in the workspace by specifying an Object Type, and/or a Data Type, and/or a property value. Search results appear in the [Section 4.8, "Search Results View"](#) view, and double-clicking a result will open that model in the appropriate editor.



Find Model Object - Launches the Find Model Object dialog, which can be used to find an object in the workspace by specifying all or part of its name. Selecting the object will open it in the appropriate editor.

12.1. Finding Model Objects

The Teiid Designer provides a name-based search capability to quickly locate and display model objects.

- To find a model object:
 - **Step 1** - Open the **Find Model Object** dialog by either selecting the  action on the main Teiid Designer tool-bar.

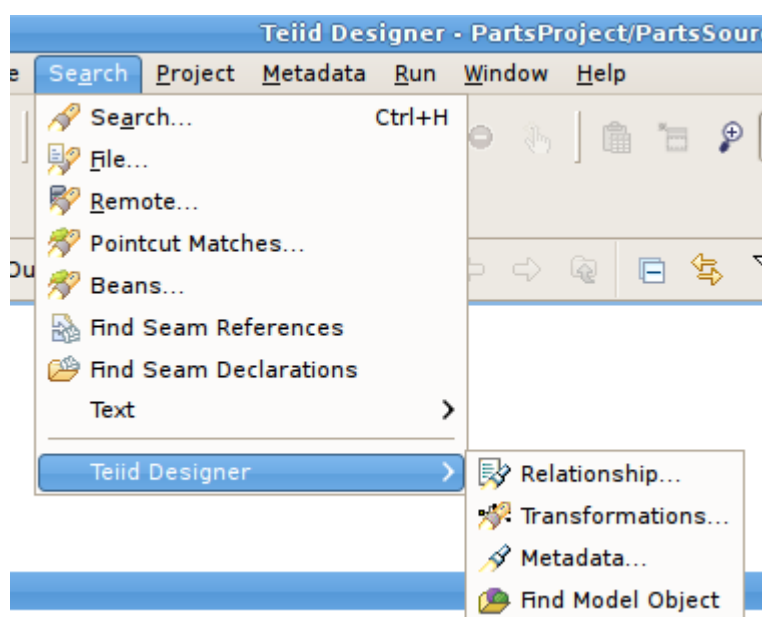



Figure 12.2. Find Model Object Action In Toolbar

or select the same action via the main menu's **Search > Find Model Object**  action.

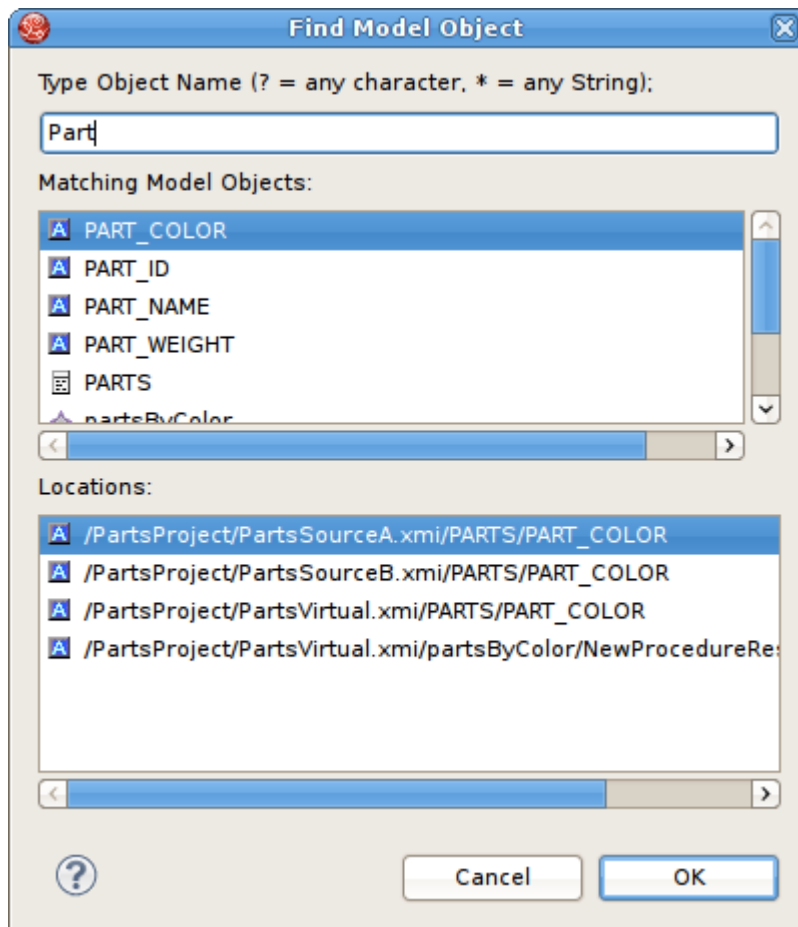


Figure 12.3. Find Model Object Dialog

- **Step 2** - Begin typing a word or partial word in the **Type Object Name** field. Wild-card (*) characters will be honored. As you type, the objects which match the desired name will be displayed in the **Matching Model Objects** list. If there are more than one objects with the same name, the locations or paths of the objects are displayed in the **Locations** list.
- **Step 3** - If more than one object exists with the desired name, select the one of the locations.
- **Step 4** - Click **OK**. If editor is not open for the object's model, an editor will open. The desired object should end up displayed in a diagram (if applicable) and selected.

12.2. Search Models Via Relationship Properties

The Teiid Designer provides a search capability to find model objects that are characterized by one or more relationship property values including relationship type, participants and name.

- To search for relationships:

- **Step 1** - Select **Search > Relationships...** action on the main menu



which opens the **Search** dialog.

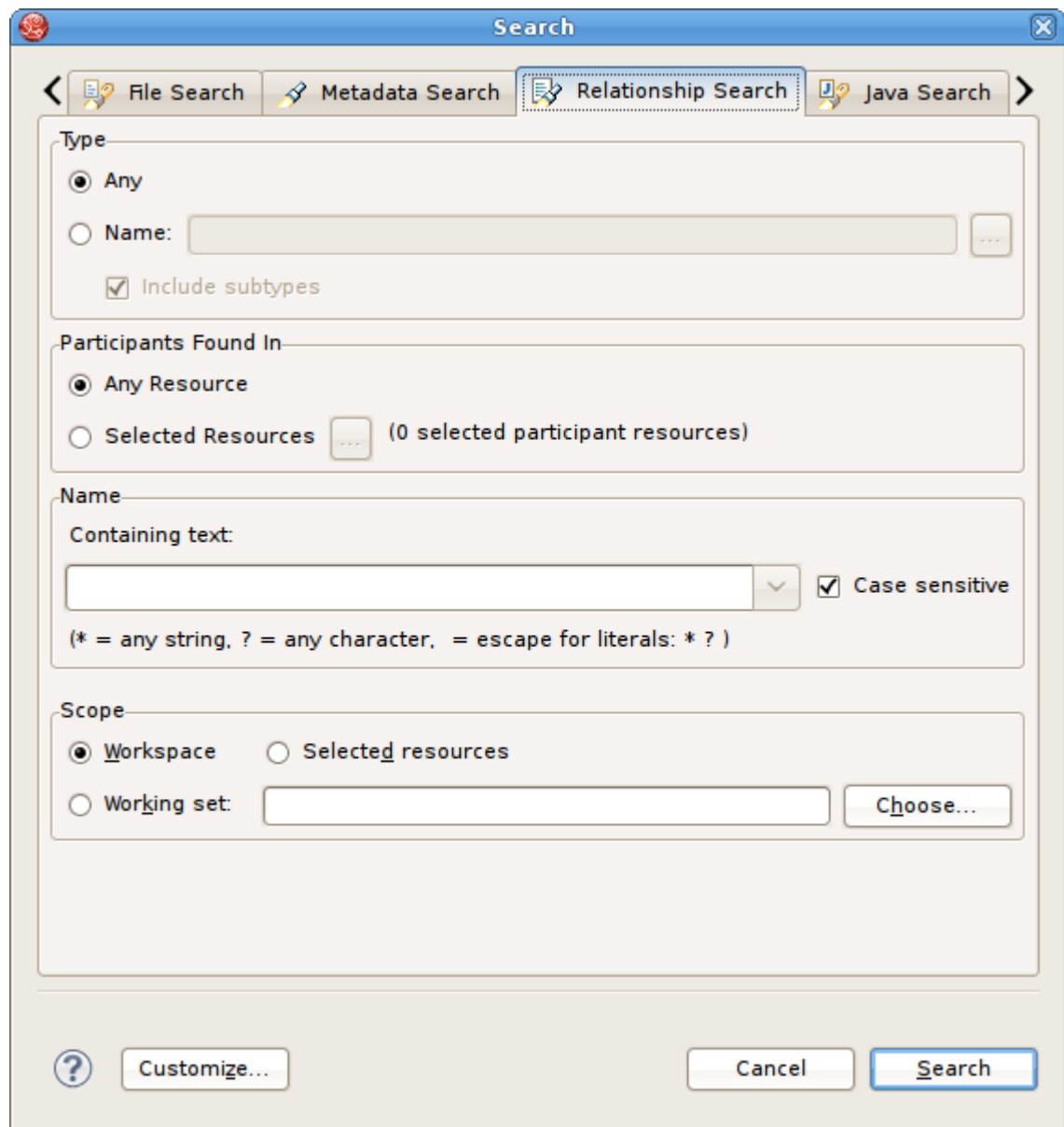


Figure 12.4. Relationship Search Dialog

- **Step 2** - Specify desired search options for **Object Type**, location of **Participants** and **Name**.
- **Step 3** - Click **Search**. The search will be performed and the results will be displayed in the [Section 4.8, “Search Results View”](#). If the view is not yet open, it will be opened automatically.

12.3. Search Transformation SQL

The Teiid Designer provides a search capability to string values present in transformation SQL text.

- To search for string values in your transformations SQL:
- Step 1** - Select **Search > Transformations...** action on the main menu



which opens the **Search Transformations** dialog.

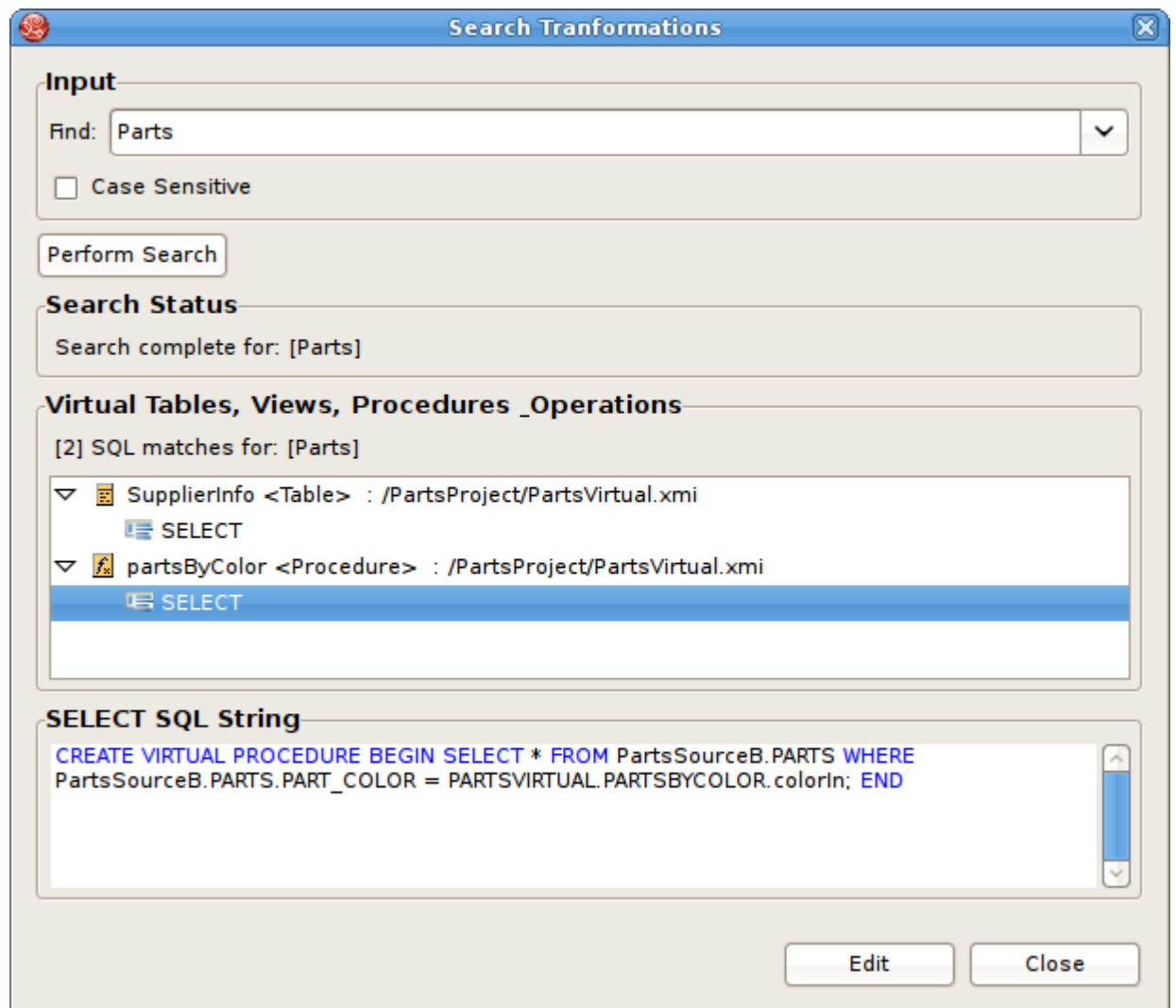


Figure 12.5. Search Transformations Dialog

- Step 2** - Specify a string segment in the **Find:** field and specify/change your case sensitive preference.

- **Step 3** - Select **Perform Search** button. Any transformation object containing SQL text which contains occurrences of your string will be displayed in the results section.

You can select individual objects and view the SQL. If a table or view supports updates and there is insert, update or delete SQL present, you can expand the object and select the individual SQL type as shown below.

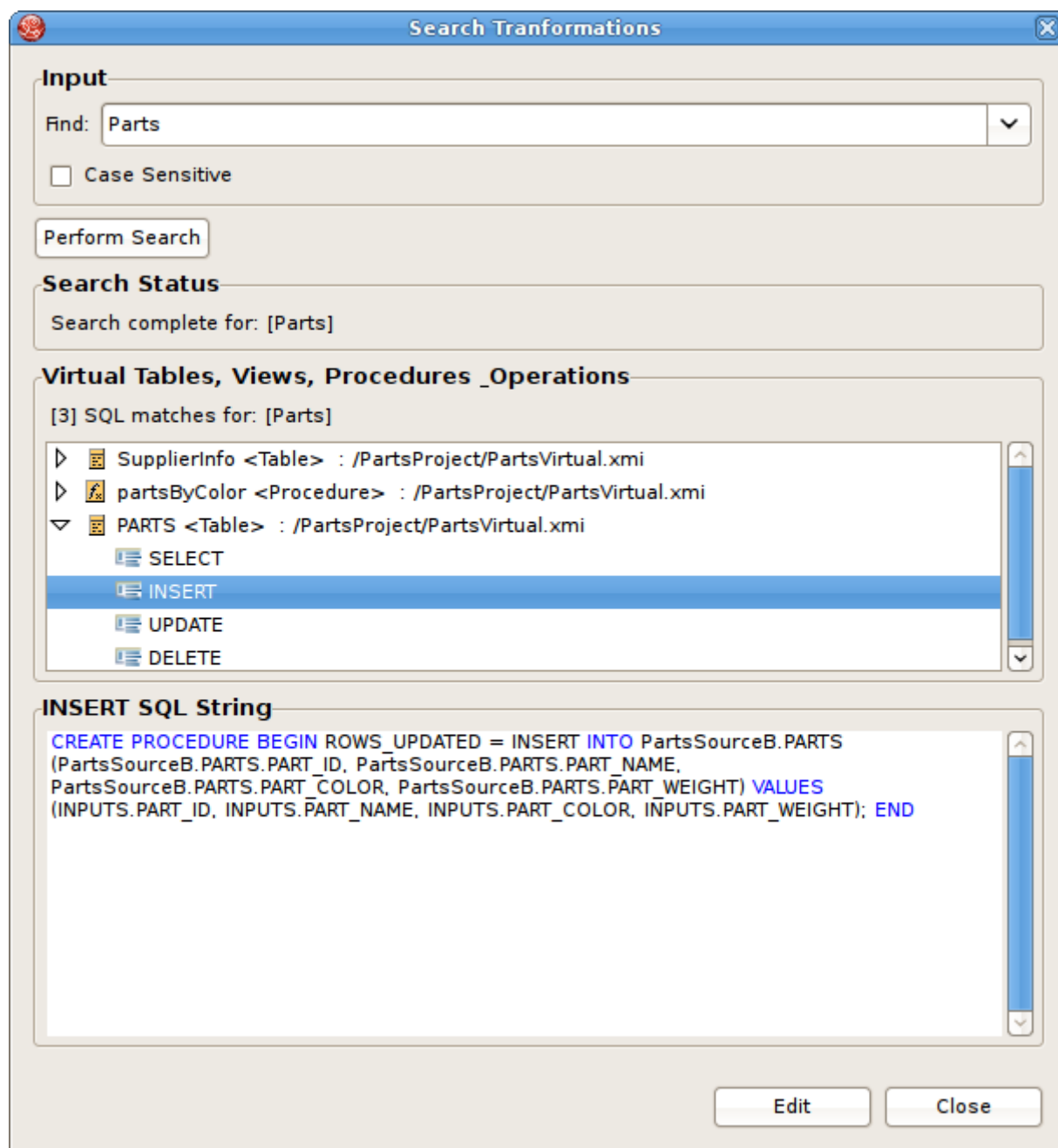



Figure 12.6. Insert SQL Example

If you wish to view the selected object and its SQL in a **Model Editor**, you can click the **Edit** button. An editor will be opened if not already open. If an editor is open its tab will be selected. In

addition, the **Transformation Editor** will be opened and you can perform **Find/Replace** (Ctrl-F) actions to highlight your original searched text string and edit your SQL if you wish.

12.4. Search Models Via Metadata Properties

The Teiid Designer provides a search capability to find model objects that are characterized by one or more metadata property values.

- To search your models using metadata:
 - **Step 1** - Select **Search > Metadata...** action on the main toolbar
 which opens the **Search** dialog.

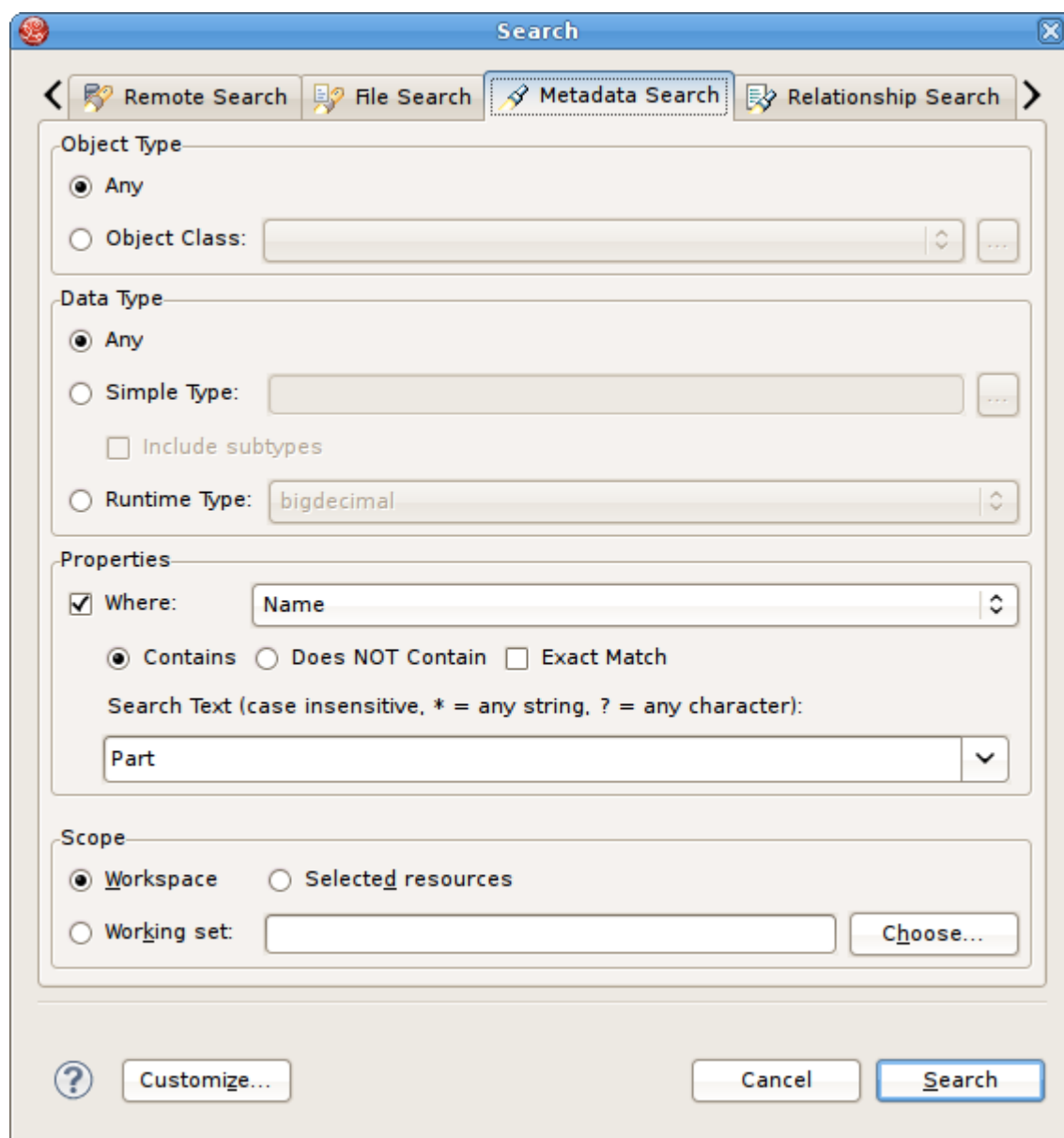


Figure 12.7. Metadata Search Dialog

- **Step 2** - Specify desired search options for **Object Type**, **Data Type** and **Properties**.
- **Step 3** - Click **Search**. The search will be performed and the results will be displayed in the [Section 4.8, “Search Results View”](#). If the view is not yet open, it will be opened automatically.

User Preferences

The Teiid Designer provides options or preferences which enable customization of various modeling and UI behaviors. Preferences can be accessed via the Edit > Preferences action on the Main toolbar.

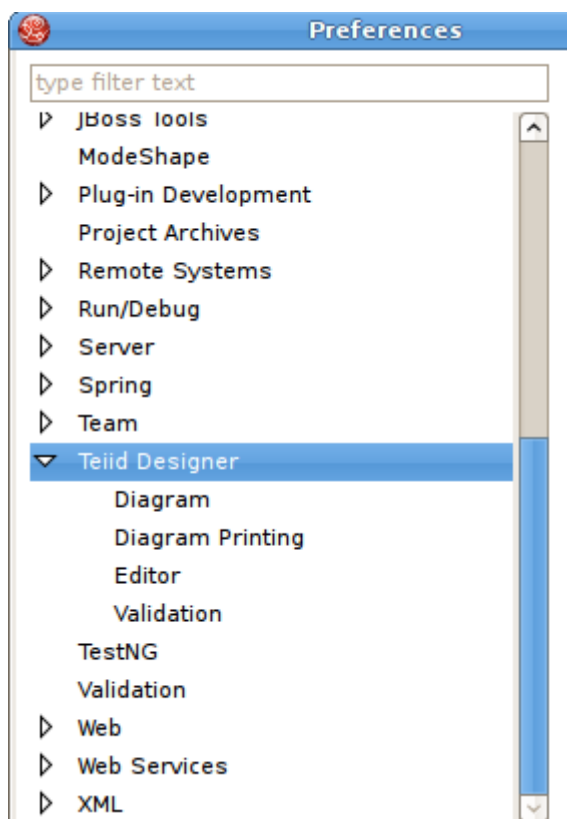


Figure 13.1. Preferences Dialog

13.1. Teiid Designer Preferences

General Teiid Designer preferences include.

- **Always open editor without prompting** To change/edit a model, it must be opened for editing. Checking this box will automatically open the model in an editor if the user attempts to perform a change in a model. If unchecked, the user will be informed that an editor will be opened before the operation is completed.
- *Open Designer perspective when model is opened* - If a model is opened via importing projects, the New > Teiid Metadata Model menu and the Teiid Designer perspective is not open, you may want to automatically open the perspective and begin working on your model. This preference has 3 settings. Always open, which means always open the perspective without prompting; never open, which means do not open the Teiid Designer perspective, or prompt, which will always ask you if you wish to open the Teiid Designer perspective.

- **Check and update imports during save** occasionally editing a model may add or remove objects in one model that reference objects in another model. Model Imports keep track of these dependencies within each model. A validation error or warning may appear during a build. Checking this box will automatically check and update imports during the save process. This will result in any unneeded imports being removed from the model or any required imports added to the model. If unchecked, no updating of imports will be performed.
- **Import Logical ER Models as View Relational Models** - When importing ER models, the default relational model created during the import process is a Source model. This preferences, forces the importer to create a View relational model instead.
- **Enable Preview** - If the Designer Runtime feature is installed and a Teiid Instance is defined, Teiid Designer will automatically keep the preview artifacts (VDBs) in sync with the workspace models. Unchecking this preference will disable preview feature and not create preview artifacts.
- **Enable Preview Teiid Cleanup** - If operating Designer with Enable Preview = TRUE, then this preference will result in automatic clean-up of your preview artifacts from your Teiid servers. Any preview VDBs or preview data sources will be undeployed from your servers as part of Eclipse's shut-down process.

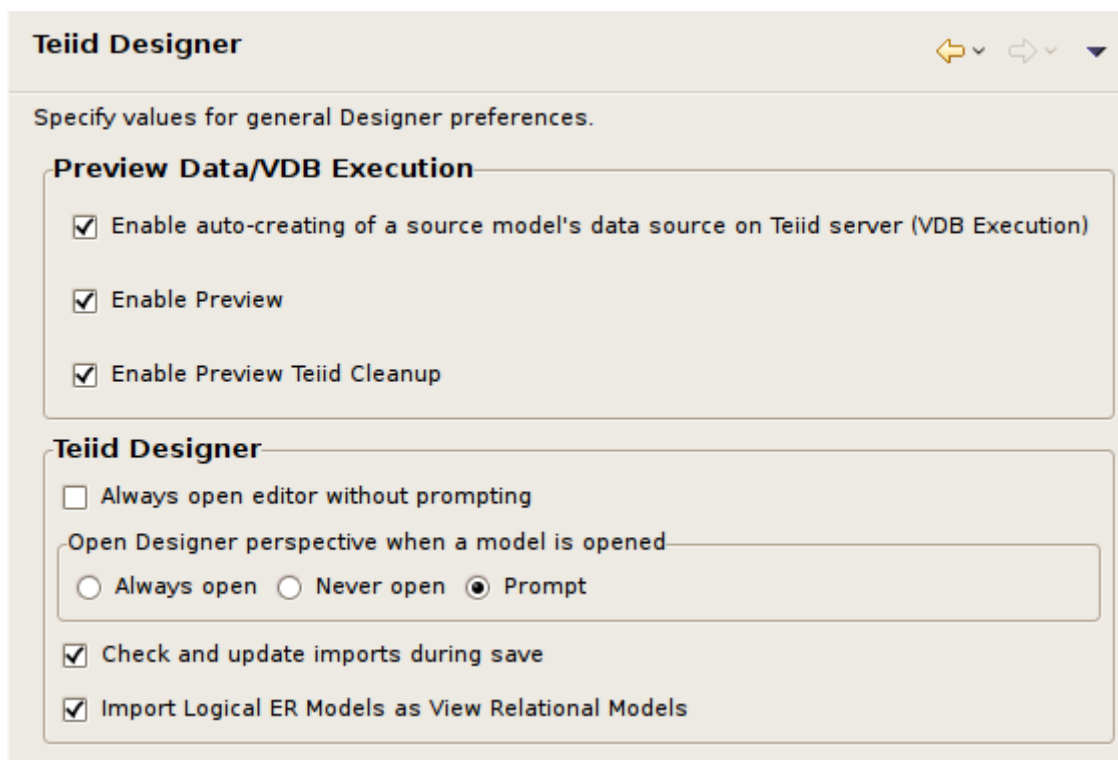


Figure 13.2. General Teiid Designer Preferences Panel

13.1.1. Diagram Preferences

Several diagram preferences are available to customize your diagrams.

- **Notations** - Standard diagram notation for Teiid Designer is based on UML notation. Future releases may include alternate notations.
- **Routers** - The relationship link type for Package and Custom diagrams (Foreign Key - Primary Key relationships) can be customized. Available options include Orthogonal (default), Direct or Manual (user defined breakpoints).
- **Font Settings** - Select font type, style and size.
- **Background Color Settings** - Select a unique background color for each diagram type to help differentiate between types.
- **Model Size** - Displaying very large diagram may take a considerably long time. This preference allows users to set an upper limit on the number of objects to display in a diagram. If this limit is exceeded, a warning is displayed to the user and the diagram is not constructed.
- **Relationship Options** - UML-type relationships can be customized in a couple of ways. Role Names and Multiplicity labels can be shown or hidden using the check-boxes labeled **Show Role Names** and **Show Multiplicity**.

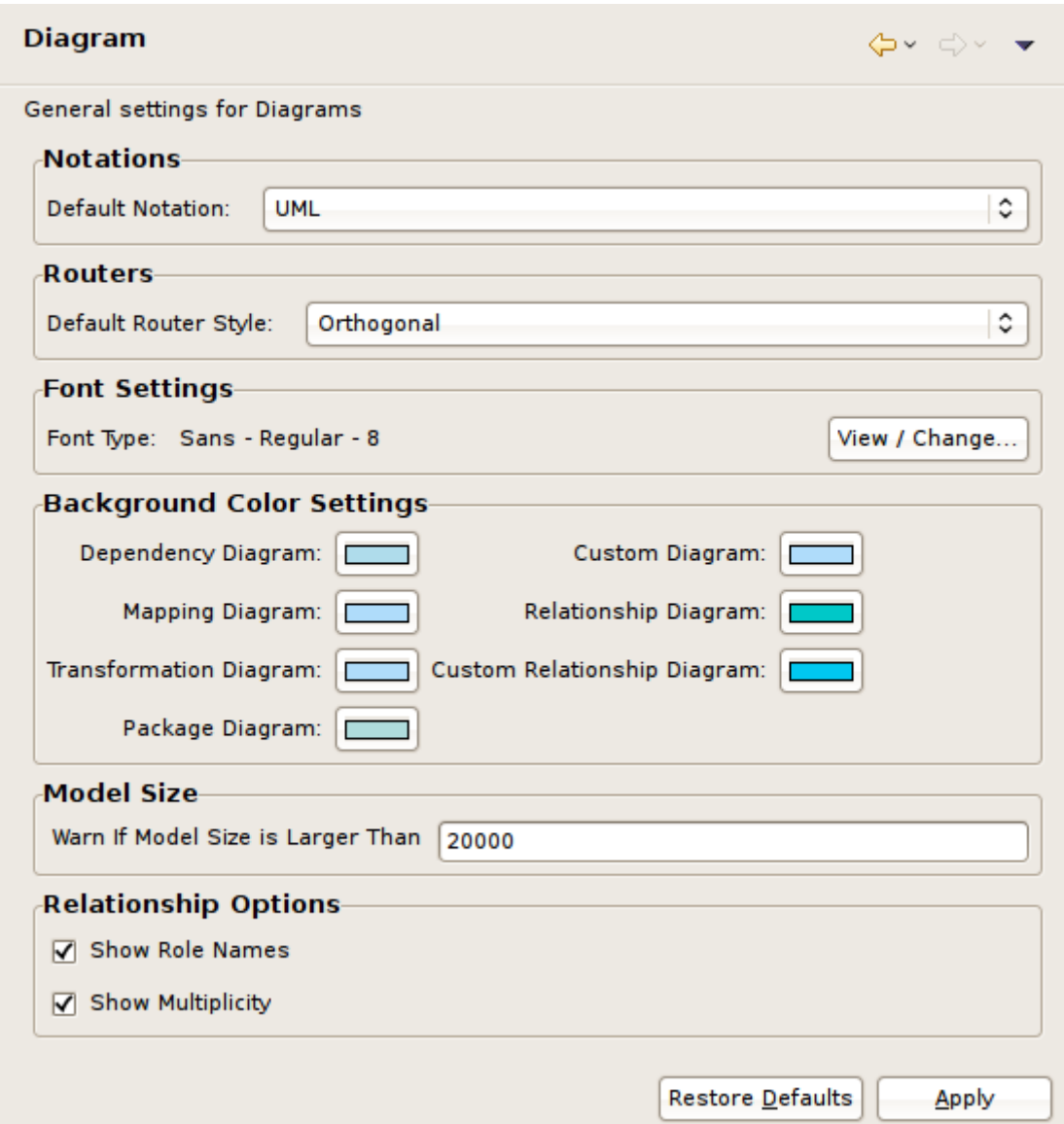


Figure 13.3. Diagram Preferences Panel

13.1.2. Diagram Printing Preferences

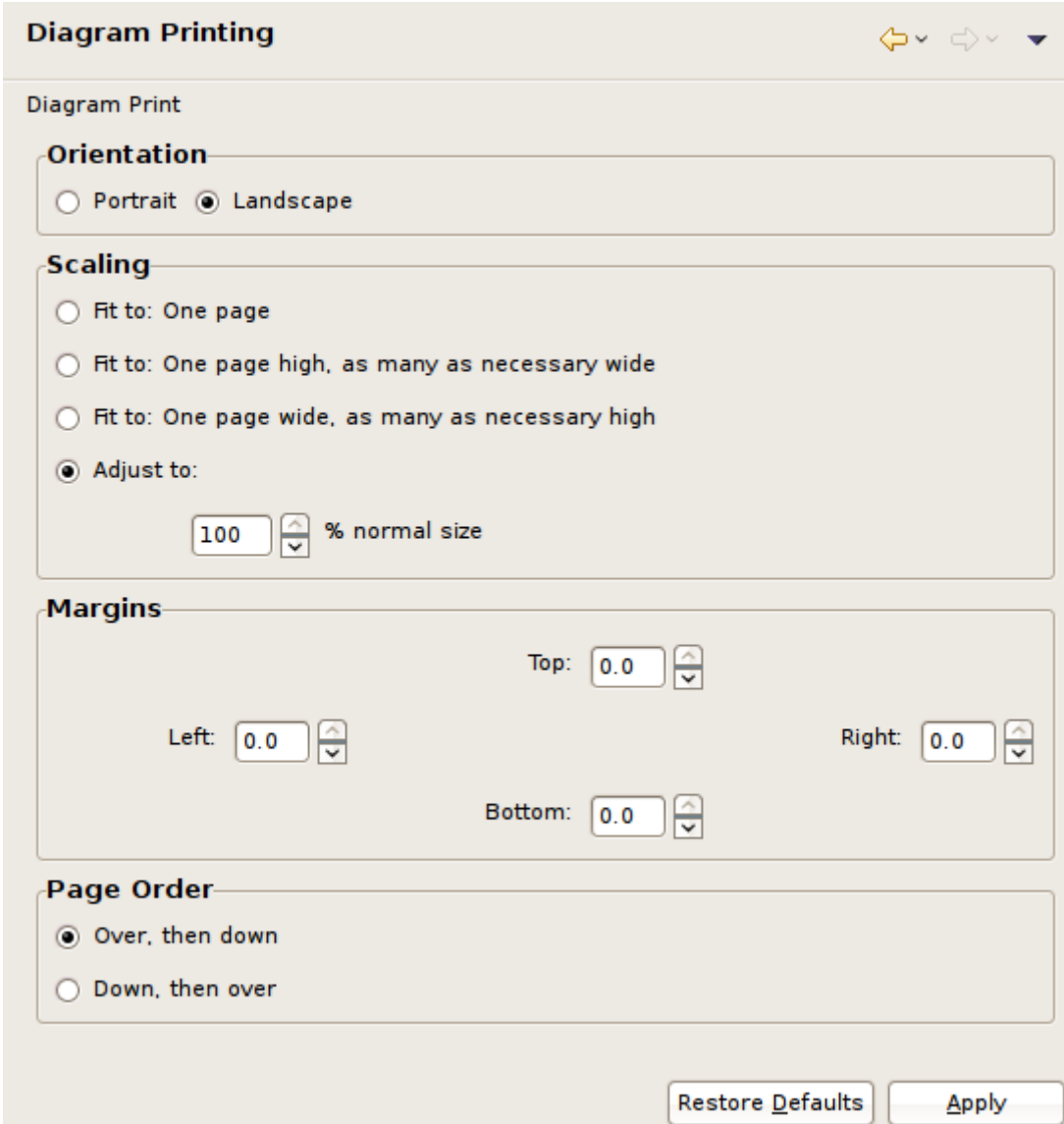
Diagram print options are stored as preferences. These can be accessed through this preference page, by right-click on diagram **Page Setup** action



or via the **Modify Diagram Printing Preferences** action



located on the vertical diagram toolbar .



The image shows a 'Diagram Printing' preferences panel. It has a title bar with the text 'Diagram Printing' and three small icons (a yellow arrow pointing left, a grey arrow pointing right, and a downward-pointing triangle). The panel is divided into four main sections: 'Diagram Print', 'Orientation', 'Scaling', and 'Margins'. The 'Diagram Print' section is at the top and contains the 'Orientation' section. The 'Orientation' section has two radio buttons: 'Portrait' and 'Landscape', with 'Landscape' being selected. The 'Scaling' section has four radio buttons: 'Fit to: One page', 'Fit to: One page high, as many as necessary wide', 'Fit to: One page wide, as many as necessary high', and 'Adjust to:'. The 'Adjust to:' option is selected. Below the 'Adjust to:' radio button is a text input field containing '100' and a '% normal size' label. The 'Margins' section has four spinners for 'Top', 'Left', 'Right', and 'Bottom', all set to '0.0'. The 'Page Order' section has two radio buttons: 'Over, then down' (selected) and 'Down, then over'. At the bottom right of the panel are two buttons: 'Restore Defaults' and 'Apply'.

Diagram Printing

Diagram Print

Orientation

☐ Portrait ☒ Landscape

Scaling

☐ Fit to: One page

☐ Fit to: One page high, as many as necessary wide

☐ Fit to: One page wide, as many as necessary high

☒ Adjust to:

100 % normal size

Margins

Top: 0.0

Left: 0.0

Right: 0.0

Bottom: 0.0

Page Order

☒ Over, then down

☐ Down, then over

Restore Defaults Apply

Figure 13.4. Diagram Preferences Panel

13.1.3. Editor Preferences

13.1.3.1. XML Document Preferences

XML Document Mapping Preferences provide ways to customize [Section 5.1.1.4, “Mapping Diagram”](#) and [Section 8.2.4, “Recursion Editor \(XML\)”](#) behavior.

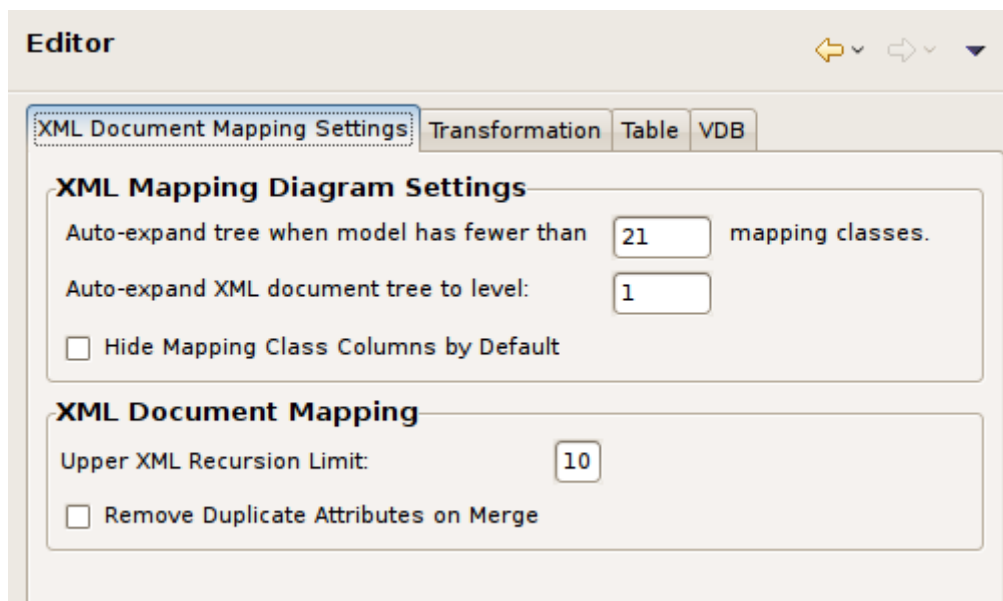


Figure 13.5. XML Document Preferences Panel

13.1.3.2. Table Editor Preferences

[Section 5.1.2, “Table Editor” Preferences](#) provide a way to customize the order and the information content for each model object type.

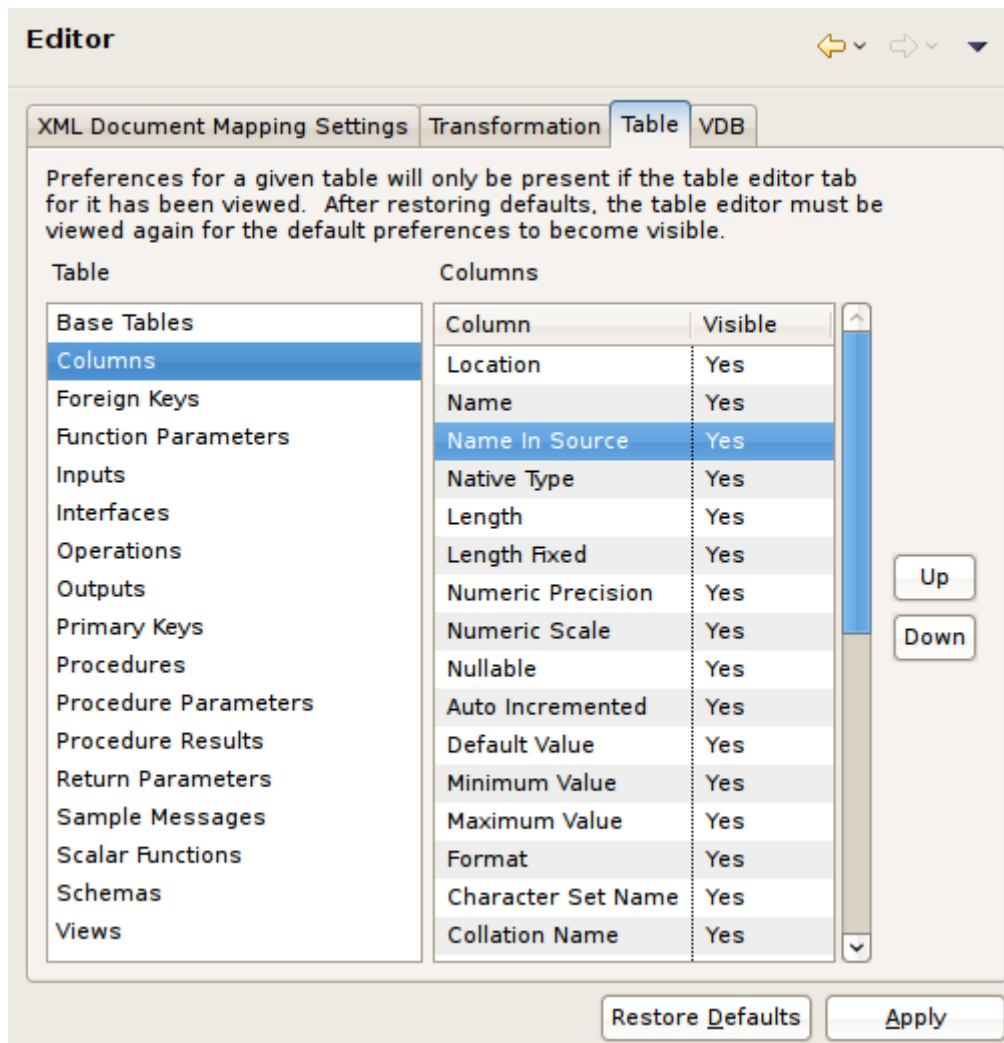


Figure 13.6. Table Editor Preferences Panel

13.1.3.3. Transformation Editor Preferences

[Section 8.2.1, “Transformation Editor”](#) Preferences provide a way to customize SQL formatting, diagram layout, and default view entity properties.

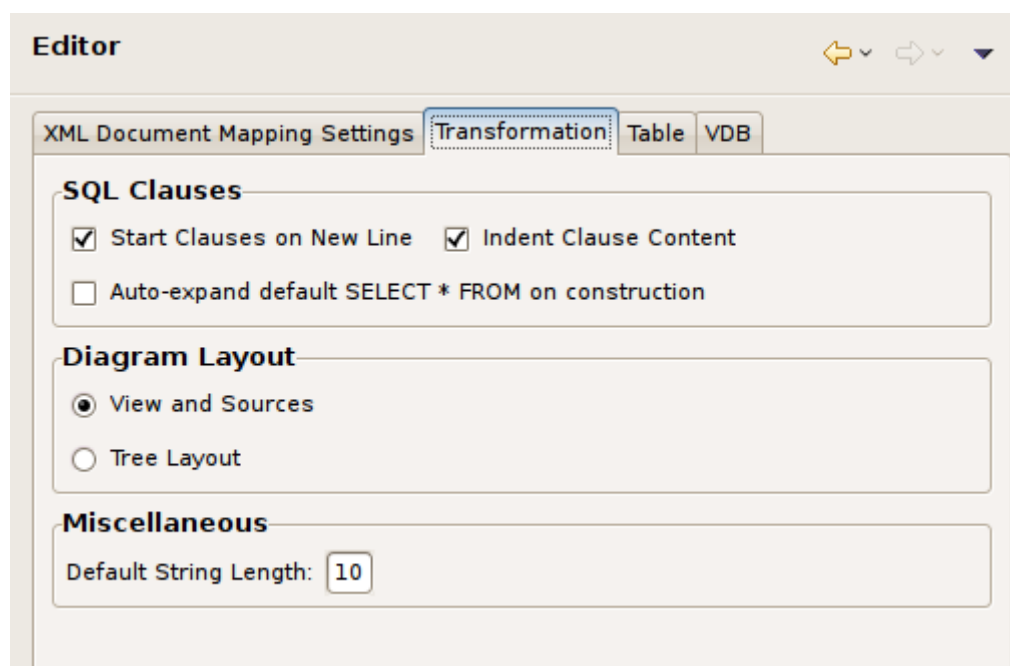


Figure 13.7. Transformation Editor Preferences Panel

13.1.3.4. VDB Editor Preferences

[Section 5.2, “VDB Editor”](#) Preferences provide a way to customize VDB editor behavior.

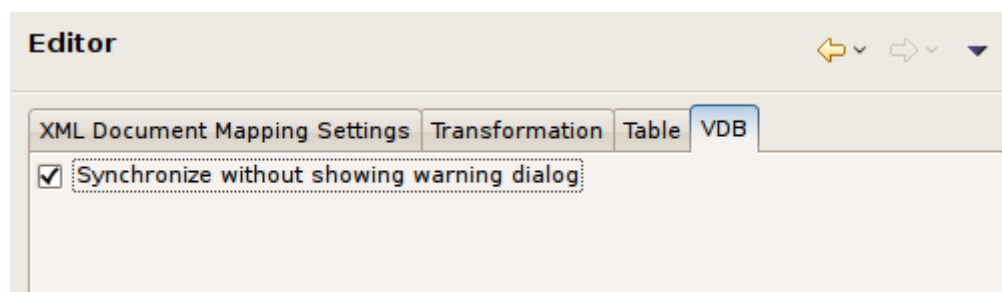


Figure 13.8. VDB Editor Preferences Panel

13.1.4. Validation Preferences

Validation Preferences provide a way to customize the severity of some of the rules checked during model validation.

Validation preference pages, shown below, include Core, Relational Model, XML-related and XML Schema (XSD) models.

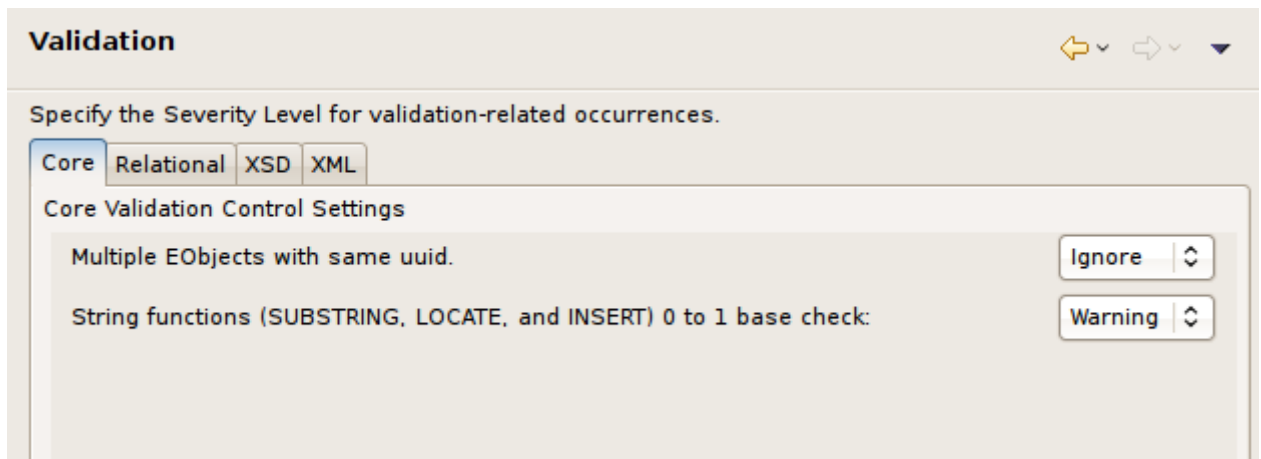


Figure 13.9. Core Model Validation Preferences Panel

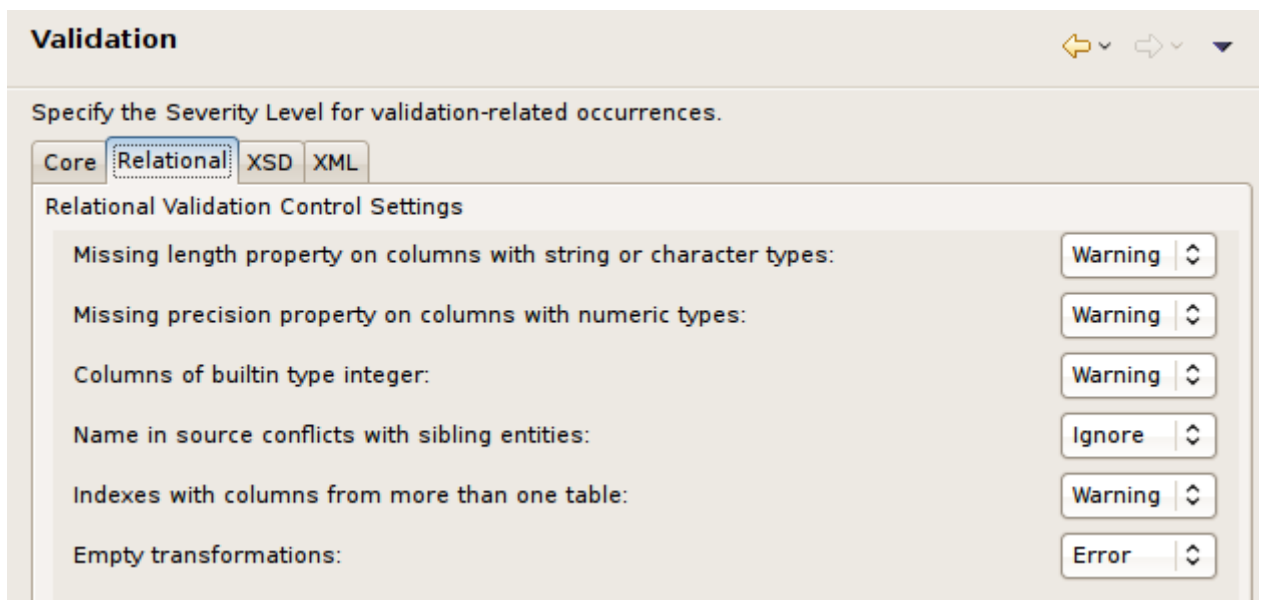


Figure 13.10. Relational Model Validation Preferences Panel

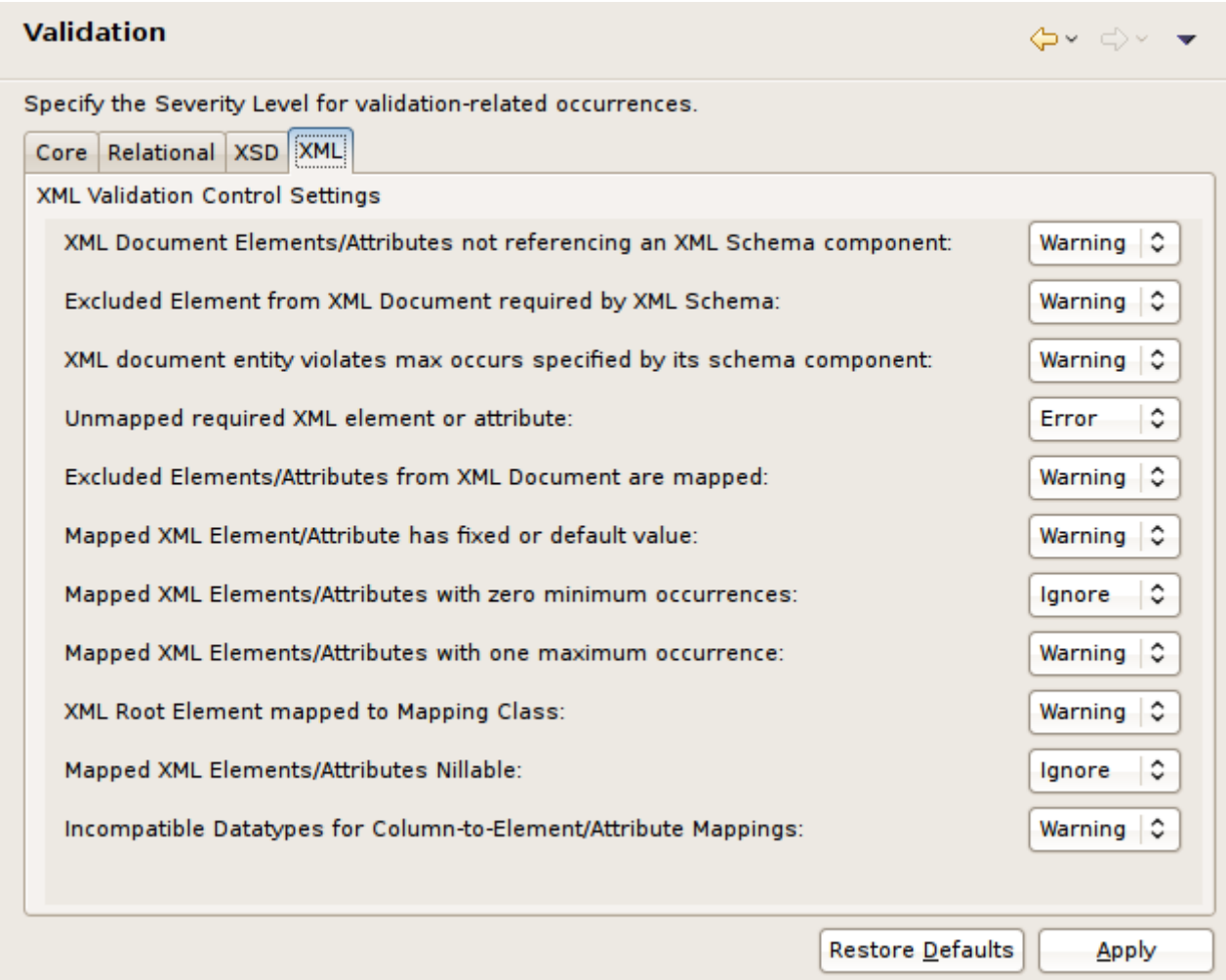


Figure 13.11. XML Document Model Validation Preferences Panel

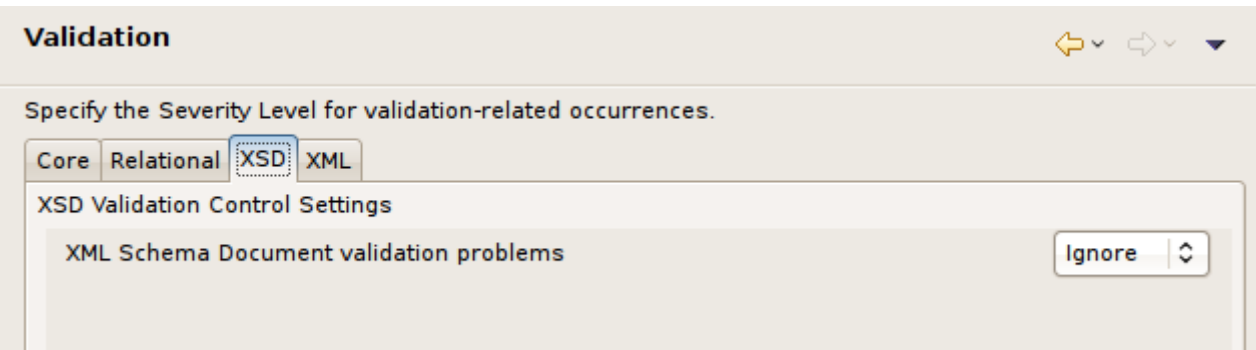


Figure 13.12. XSD Schema Model Validation Preferences Panel



Note

Increasing the severity level to error will prevent you from testing your VDB or deploying a web service if violations of that preference are found during validation.

