Teiid Designer User Guide

1

11.1.2

1. Introduction 1
1.1. What is Teiid Designer? 1
1.2. Metadata Overview 2
1.2.1. What is Metadata 2
1.2.2. Business and Technical Metadata 4
1.2.3. Source and View Metadata 5
1.3. It's all in the Modeling... 9
1.3.1. What Are Models? 9
1.3.2. How is a Model Defined? 11
1.3.3. Guiding through the process 11
1.3.4. Targeting Your Teiid Submoduler 11
1.3.5. Model Classes and Types 12
1.3.6. The Virtual Database 12
1.3.7. Model Validation 14
1.3.8. Testing Your Models 14
1.3.9. Model Object Extensions 15
2. Dive Right In! 19
2.1. Targeting the Teiid Server 19
2.1.1. Server Version Preference 19
2.1.2. Defining a Teiid Server 20
2.1.3. Server Version Status Panel 21
2.2. Guide Example 22
2.2.1. Model a Relational (via JDBC) Source 22
2.3. Cheat Sheet Example 37
2.3.1. Consume a SOAP Web Service 37
3. Server Management 51
3.1. Setting up a Server 51
3.1.1. Teiid Version Support 54
4. New Model Wizards 55
4.1. Creating New Relational Source Model 56
4.1.1. Generate File Translator Procedures 57
4.1.2. Generate Web Service Translator Procedures 58
4.1.3. Copy From Existing Model 60
4.2. Creating New Relational View Model 61
4.2.1. Copy From Existing Model 62
4.2.2. Transform From Existing Model 63
4.3. Creating XML Document View Model 63
4.3.1. Copy From Existing Model 63
4.3.2. Build XML Documents From XML Schema 64
4.4. Creating XML Schema Model 68
4.4.1. Copy From Existing Model 69
4.5. Creating Web Service View Model 69
4.5.1. Copy From Existing Model 70
4.5.2. Build From Existing WSDL File(s) or URL 71

4.5.3. Build From Relational Models 71
4.5.4. Build From XML Document View Models 72
5. Importers 75
5.1. Import DDL 75
5.2. Import From Relational Database 77
5.2.1. Relational Model Costing 84
5.3. Import From Teiid Data Source Connection 84
5.4. Import From Flat File Source 89
5.5. Import From XML Data File Source 104
5.6. Import From Salesforce 112
5.7. Import Metadata From Text File 117
5.7.1. Import Relational Model (XML Format) 118
5.7.2. Import Relational Tables (CSV Format) 122
5.7.3. Import Relational View Tables (CSV Format) 124
5.8. Import Data From REST Service 126
5.9. Import Data From SOAP Service 133
5.9.1. Circular References in WSDL Schemas 140
5.10. Import WSDL Into Web Service 141
5.10.1. Import WSDL From Workspace Location 142
5.10.2. Import WSDL From File System Location 148
5.10.3. Import WSDL From URL 154
5.11. Import from an XML Schema File 160
5.12. Import From an LDAP Server 164
6. Creating and Editing Model Objects 171
6.1. Creating New Model Objects 171
6.1.1. New Child Action 171
6.1.2. New Sibling Action 174
6.1.3. New Association Action 176
6.2. New Model Object Wizards 178
6.2.1. Create Relational Table Wizard 178
6.2.2. Create Relational Procedure Wizard 179
6.2.3. Create Relational Index Wizard 186
6.2.4. Create View Model Objects Wizards 187
6.3. Model Object Editors 190
6.3.1. Transformation Editor 192
6.3.2. Input Set Editor (XML) 210
6.3.3. Choice Editor (XML) 212
6.3.4. Recursion Editor (XML) 215
6.3.5. Operation Editor 219
6.4. Managing Model Object Extensions 221
6.4.1. Create New MED 222
6.4.2. Edit MED 223
6.4.3. Extending Models With MEDs 227
6.4.4. Setting Extended Property Values 228

7. Metadata-specific Modeling 231
7.1. Relational Source Modeling 231
7.1.1. Source Function 231
7.2. Relational View Modeling 231
7.2.1. Create Materialized Views 231
7.3. XML Document Modeling 235
7.3.1. Create XML View Documents from schema 235
7.4. Web Services Modeling 239
7.4.1. Create Web Service Action 239
7.4.2. Web Services War Generation 241
8. User Defined Functions 255
8.1. Modeling your functions 255
8.2. Utilizing your UDFs in transformations 256
8.3. Including functions in your VDB 257
8.4. Using legacy UDF models 257
9. Editing Models and Projects 261
9.1. Rename A Model 261
9.2. Move Model 263
9.3. Save Copy of Model 264
9.4. Clone Project 266
10. Managing VDBs 269
10.1. Creating a VDB 269
10.2. Editing a VDB 270
10.3. Test a VDB 270
10.4. Multi-source Binding Support 270
10.5. UDF support 271
10.6. Reusing VDBs 272
10.7. Security and Data Access 277
10.8. Working with Dynamic VDBs 281
10.8.1. Generate VDB Archive 281
10.8.2. Generate Dynamic VDB 283
11. Testing Your Models 287
11.1. Manage Connection Profiles 287
11.1.1. Set Connection Profile for Source Model 287
11.1.2. View Connection Profile for Source Model 287
11.1.3. Remove Connection Profile from Source Model 289
11.2. Previewing Data For a Model 289
11.2.1. Preview Relational Table or View 293
11.2.2. Preview Relational Table With Access Pattern 293
11.2.3. Preview Relational Procedure 294
11.2.4. Preview Web Service Operation 295
11.2.5. Sample SQL Results for Preview Data 296
11.2.6. Execution Plans 297
11.3. Testing With Your VDB 298

11.3.1. Creating Data Sources 298
11.3.2. Execute VDB from Model Explorer 300
11.3.3. Deploy VDB from Model Explorer 301
11.3.4. Executing a Deployed VDB 302
12. Materialization Support 309
12.1. Materialization Extension Properties 309
12.2. Create POJO 311
12.3. Materialize a View 312
12.4. JDG Connection Profile 316
13. Searching 319
13.1. Finding Model Objects 319
13.2. Search Transformation SQL 321
13.3. Search Models Via Metadata Properties 323
A. Supported Data Sources 325
B. Designer Metadata Usage Requirements In Teiid Runtime 327
C. User Preferences 339
C.1. Teiid Designer Preferences 339
C.1.1. Diagram Preferences 341
C.1.2. Editor Preferences 342
C.1.3. Validation Preferences 345
D. Teiid Designer Ui Reference 349
D.1. Teiid Designer Perspectives 349
D.1.1. Teiid Designer Perspective 349
D.1.2. Opening a Perspective 350
D.1.3. Further information 353
D.2. Teiid Designer Views 353
D.2.1. Model Explorer View 353
D.2.2. Outline View 356
D.2.3. Server View 358
D.2.4. Connections View 365
D.2.5. Properties View 367
D.2.6. Description View 368
D.2.7. Problems View 370
D.2.8. Search Results View 372
D.2.9. Datatype Hierarchy View 374
D.2.10. Teiid Model Classes View 375
D.2.11. System Catalog View 376
D.2.12. SQL Reserved Words View 377
D.2.13. Model Extension Definition Registry View (MED Registry View) 378
D.2.14. Guides View 380
D.2.15. Status View 382
D.2.16. Cheat Sheets View 383
D.3. Editors 384
D.3.1. Model Editor 386

D.3.2. VDB Editor 407
D.3.3. Model Extension Definition Editor 412
D.4. Teiid Designer Main Menu 415
D.4.1. File Menu 416
D.4.2. Edit Menu 419
D.4.3. Refactor Menu 420
D.4.4. Navigate Menu 421
D.4.5. Search Menu 421
D.4.6. Project Menu 423
D.4.7. Metadata Menu 424
D.4.8. Run Menu 424
D.4.9. Window Menu 425
D.4.10. Help Menu 426

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.

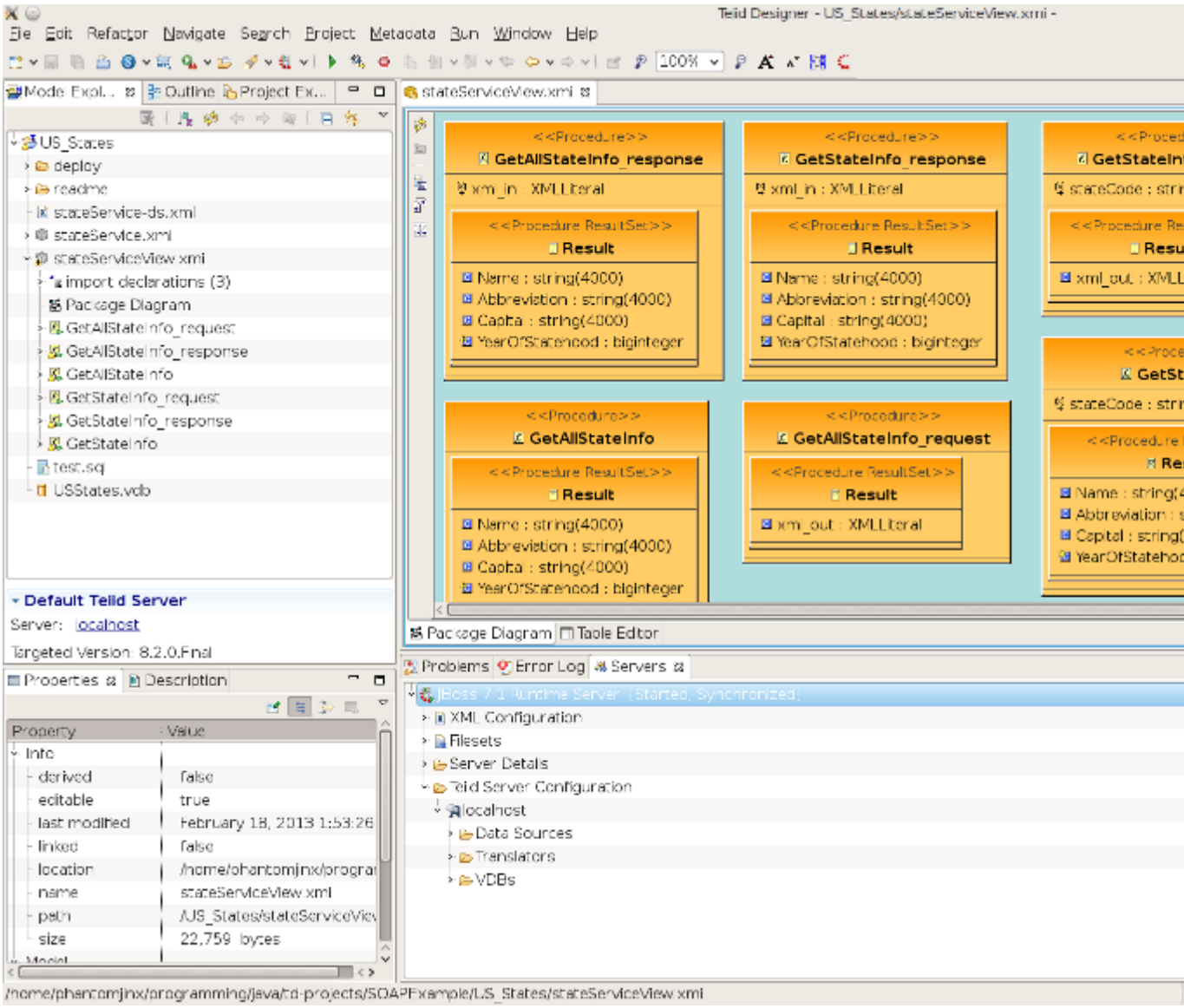


Figure 1.1. Teiid Designer

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

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

1.2. Metadata Overview

1.2.1. 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 1.2.2, "Business and Technical Metadata"*)
- What data does the metadata represent? (see *Section 1.2.3, "Source and View Metadata"*)
- How will my organization use and manage this metadata? (see *???*)

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.

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 1.2.3, "Source and View Metadata"*).

Note

For detailed information about creating models from your metadata, see *Section 1.3, "It's all in the Modeling..."*

1.2.2. 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

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 1.2.3, "Source and View Metadata"*).

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.

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.

1.2.3. 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.

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.

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 6.3.1, "Transformation Editor"*."

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.

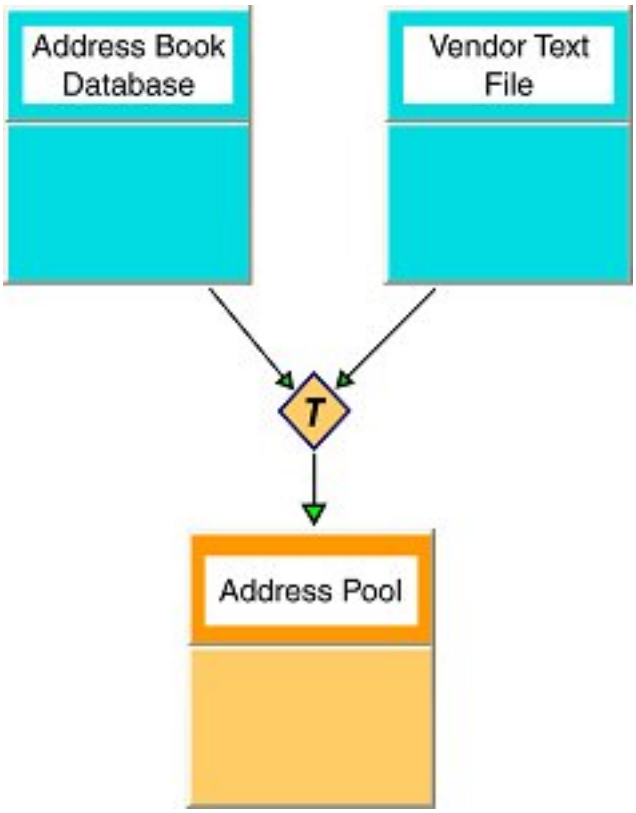


Figure 1.2. Data Flow for View Transformations

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.

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.

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.

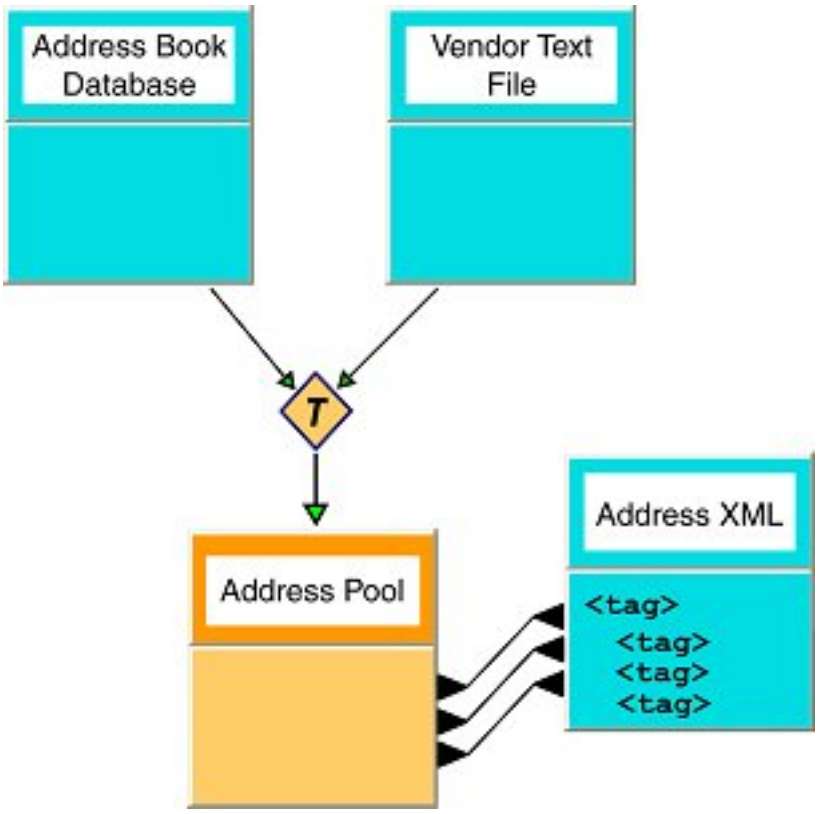


Figure 1.3. Data Flow for XML Transformations

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.

1.3. It's all in the Modeling...

1.3.1. 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-1079-9d
    <modelImports xmi:uuid="mmuuid:2d815780-4140-1079-9d18-8acf4
    <modelImports xmi:uuid="mmuuid:2e663940-4140-1079-9d18-8acf4
  </mmcore:ModelAnnotation>
  <relational:BaseTable xmi:uuid="mmuuid:b20e64c0-413b-1079-9d18
    <columns xmi:uuid="mmuuid:bb5ac3c0-413b-1079-9d18-8acf4a7127
      <type href="http://www.w3.org/2001/XMLSchema#long"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c0-413b-1079-9d18-8acf4a7127
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c1-413b-1079-9d18-8acf4a7127
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c2-413b-1079-9d18-8acf4a7127
      <type href="http://www.metamatrix.com/metamodels/SimpleDat
    </columns>
    <primaryKey xmi:uuid="mmuuid:d481f940-413b-1079-9d18-8acf4a7
  </relational:BaseTable>
```

Figure 1.4. Sample Model File

Note

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 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.

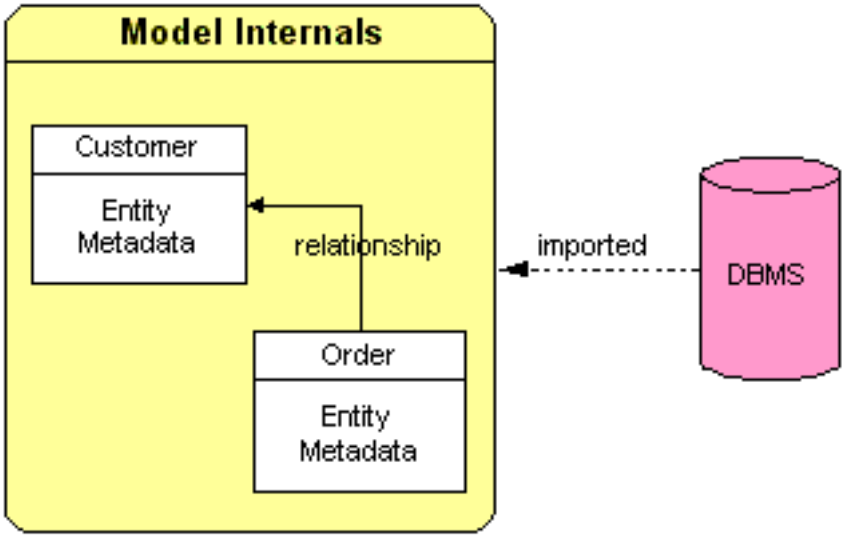


Figure 1.5. Model Internals

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 4, New Model Wizards*

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

1.3.2. How is a Model Defined?

Models are defined using Teiid Designer in various ways:

- Created via importing source data characteristics. (see *Chapter 5, Importers*)
- Manual creation via *Chapter 4, New Model Wizards*
- Transforming or copying from one model into another (see *Chapter 4, New Model Wizards* options)
- Various custom actions

1.3.3. Guiding through the process

To make the process of using Teiid Designer to build models more as easy as posssible, a guides view (*Section D.2.14, "Guides View"*) has been introduced. It provides action sets which bring together the actions necessary to develop models for specific use-cases. Action sets are available for the following scenerios:

- Consume a SOAP Web Service
- Create a REST Web Service
- Model Flat File Source (a text file)
- Model JDBC Data Source
- Model Local XML File Source
- Model Remote XML File Source
- Modell Teiid Data Source (deployed on server)
- Teiid Server Actions

1.3.4. Targeting Your Teiid Submoduler

Like Teiid Designer, the Teiid runtime is under continuous development and as such multiple versions have been and are being released. Due to changes in both its code and the underlying JBoss server, the versions are not always backward compatible. Teiid Designer provides Teiid runtime validation for VDBs based on server versions. New models must be compatible with their targeted server version hence the correct server version must be selected prior to creating them.

To aid with selection of the correct server version, two changes have been made to Teiid Designer:

- A **preference** for the targeted server version that new models will be based on;
- The concept of the **default server** has been extended so that it will determine the targeted server version of new models (superceding the preference).

1.3.5. 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).
- **Function** - The Function metamodel supports the capability to provide user-defined functions, including binary source jars, to use in custom transformation SQL statements.

1.3.6. The Virtual Database

The critical artifact that **Teiid Designer** is intended to manage is the **VDB**, or **Virtual DataBase**. Through the Teiid server, VDB's behave like standard relational database schema which can be connected to, queried and updated based on how the VDB is configured. Since VDB's are just databases once they are deployed, they can be used as sources to other view model transformations. This allows creating and deploying re-usable or common VDB's in multiple layers depending on your business needs.

Starting in Teiid Designer 9.0, support was added for converting between archive (ZIP) VDBs and simple Dynamic VDB XML files. Dynamic VDBs provide users to define their source and view metadata via Teiid DDL statements. This capability allows some users to maintain a simpler version of their VDBs in a source control system. (See the *Section 10.8, “Working with Dynamic VDBs”* section for more details)

1.3.6.1. VDB Content and Structure

In Designer, the VDB file names use a "**.vdb**" file extension. VDBs are structurally just ZIP archive files containing 3 folders:

- META-INF
  - contains "**vdb.xml**" definition file
- runtime-inf
  - contains a binary INDEX file for each model included in your VDB. Note that these INDEX files represent the actual runtime metadata and is an optimized subset of data from your design-time metadata in your models.
- <project folder name>
  - contains of the models you will be adding in the VDB Editor (i.e. *.xmi and *.xsd files)

When deployed, the metadata is consumed by Teiid in order to create the necessary runtime metadata for your model definitions.

The vdb.xml file contains:

- VDB name, version, properties
- contained model information (name, translator name, connection info)
- translator info
- data role definitions for the referenced models
- import VDB references

The **vdb.xml** file example below highlights the basic model information.

Note

The VIRTUAL and PHYSICAL <model> elements containing property references to the INDEX files as well as the <source> element info for the PHYSICAL (aka source) model EU_CustomerAccounts.xmi.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb version="1" name="Financials">
    <model visible="true" type="VIRTUAL" name="US_CustomerAccounts" path="/
Financials/US_CustomerAccounts.xmi">
        <property value="4097408696" name="checksum"/>
        <property value="Relational" name="modelClass"/>
        <property value="false" name="builtIn"/>
        <property value="1592679058.INDEX" name="indexName"/>
        <property value="/Financials/US_CustomerAccounts.xmi"
 name="imports"/>
```

```
    </model>
    <model visible="true" type="PHYSICAL" name="EU_CustomerAccounts" path="/
Financials/EU_CustomerAccounts.xmi">
        <property value="525566235" name="checksum"/>
        <property value="Relational" name="modelClass"/>
        <property value="false" name="builtIn"/>
        <property value="1119071590.INDEX" name="indexName"/>
        <source translator-name="postgresql" connection-jndi-
name="EU_CustomerAccounts" name="EU_CustomerAccounts"/>
    </model>
</vdb>
```

Fortunately, Teiid Designer simplifies the management of your VDBs by providing a dedicated VDB Editor which maintains a consistent, valid vdb.xml file for you and assists in synchronizing your workspace models with any related models in your VDB. (See the *Section D.3.2, "VDB Editor"* section)

1.3.7. 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.

Modle and VDB validation is scoped to a model project.

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

Teiid Designer will automatically validate all models whenever they are saved.

Note

The "Project > Build Automatically" menu option must be checked. When editing models, the editor tabs will display a "*" to indicate that the model has unsaved changes.

1.3.8. 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 Teiid Designer will reliably match when the VDB is deployed to the Server. For more info on server management see *Chapter 3, Server Management*

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 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.

1.3.9. Model Object Extensions

Teiid Designer in conjunction with Teiid provides an extensible framework to define custom properties for model objects over-and-above what is defined in the metamodel. These custom property values are added to your VDB and included in your runtime metadata. This additional metadata is available to use in your custom translators for both source query manipulation as well as adjusting your result set data being returned.

In the 7.6 release, Teiid Designer introduces a new **Model Extension Definition (MED)** framework that will replace the current EMF-based Model Extension metamodel in a later 8.0 release.

This new MED framework provides the following improvements:

- Eliminate need for separate EMF metamodel.
- Simpler approach including reduction of extendable metamodels and metamodel objects (Relational, Web Services, XML Document, User Defined Functions) and replacing EMF terminology with basic object types.

- Allows metamodels to be extended by multiple MEDs
- MEDs are stored in models so no added dependency needed in VDB

Also see: *Section 6.4, "Managing Model Object Extensions"* and *Section D.3.3, "Model Extension Definition Editor"*.

1.3.9.1. Model Extension Definition (MED)

The purpose of a MED is to define one or more sets of extension properties. Each set of extension properties pertains to one model object type (or metaclass). Each MED consists of the following:

- **Namespace Prefix** - a unique identifier. Typically only a small number of letters and can be used as an abbreviation for the namespace URI.
- **Namespace URI** - a unique URI.
- **Extended Metamodel URI (Model Class)** - the metamodel URI that is being extended. Each metamodel URI also has model class and that is typically what is shown in the Designer. The model classes supported for extension are: **Relational, Web Service, XML Document, and Function**.
- **Version** - (currently not being used)
- **Description** - an optional description or purpose.
- **Extended Model Object Types (Metaclasses)** - a set of model object types, or metaclasses, that have extension properties defined.
- **Properties** - the extension property definitions grouped by model object type.

A **MED** file is an XML file with an extension of "mxd." A MED schema file (see attached modelExtension.xsd file) is used to validate a MED file. Here is a sample MED file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<modelExtension xmlns:p="http://org.teiid.modelExtension/2011"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     metamodelUri="http://www.metamatrix.com/metamodels/Relational"
     namespacePrefix="mymodelextension"
 namespaceUri="org.my.extension.mymodelextension"
     version="1"
     xsi:schemaLocation="http://org.teiid.modelExtension/2011
 modelExtension.xsd"
     xmlns="http://org.teiid.modelExtension/2011">
    <p:description>This is my model extension</p:description>
    <p:extendedMetaclass
 name="com.metamatrix.metamodels.relational.impl.BaseTableImpl">
        <p:property advanced="false" index="true" masked="false"
 name="copyable" required="false" type="boolean">
```

```
            <p:description locale="en_US">Indicates if table can be copied</
p:description>
            <p:display locale="en_US">Copyable</p:display>
        </p:property>
    </p:extendedMetaclass>
</modelExtension>
```

The *MED Registry* is where the MEDs used by Designer are stored. MED files can be edited by opening the .mxd file in the *Section D.3.3, "Model Extension Definition Editor"*.

1.3.9.2. Model Extension Definition Registry (MED Registry)

A MED registry keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. There are 2 different types of MEDs stored in the registry:

- **Built-In MED** - these are registered during Designer installation. These MEDs cannot be updated or unregistered by the user.
- **User-Defined MED** - these are created by the user. These MEDs can be updated, registered, and unregistered by the user.

The MED Registry state is persisted and is restored each time a new session is started.

Dive Right In!

We are going to dive right into a couple examples of common tasks in this section. These examples will give you a quick introduction to the capabilities that are built into Designer to assist you with common design tasks. Specifically, we will introduce the following concepts:

- **Targeting the Teiid Server**

  The Teiid Server is the destination for Designer's modelling. It is essential to define the correct server version that models will be deployed to. This is achieved either by setting the server version preference or defining a teiid server in the Servers View.

- **Guides**

  The **Guides View** is a good starting point for many common modeling tasks. The view includes categorized Modeling Actions and also links to Cheat Sheets for common tasks. The categorized Modeling Actions simply group together all of the actions that you'll need to accomplish a task. You can launch the actions directly from the Guides view, rather than hunting through the various Designer menus.

- **Cheat Sheets**

  The **Cheat Sheets** go beyond even the categorized Action Sets, and walk you step-by-step through some common tasks. At each step, the data entered in the previous step is carried through the process when possible.

After seeing the Guides and Cheat Sheets in action, subsequent chapters will offer detailed explanations of the various concepts and actions.

2.1. Targeting the Teiid Server

In this section, the setting of the teiid server version is demonstrated. This can be achieved by either setting a preference or by defining a teiid server.

2.1.1. Server Version Preference

The default server version preference allows the target server version to be changed without actually having to define a teiid server in Designer. The preference's list of possible values is determined by which teiid runtime client plugins have been installed into the application.

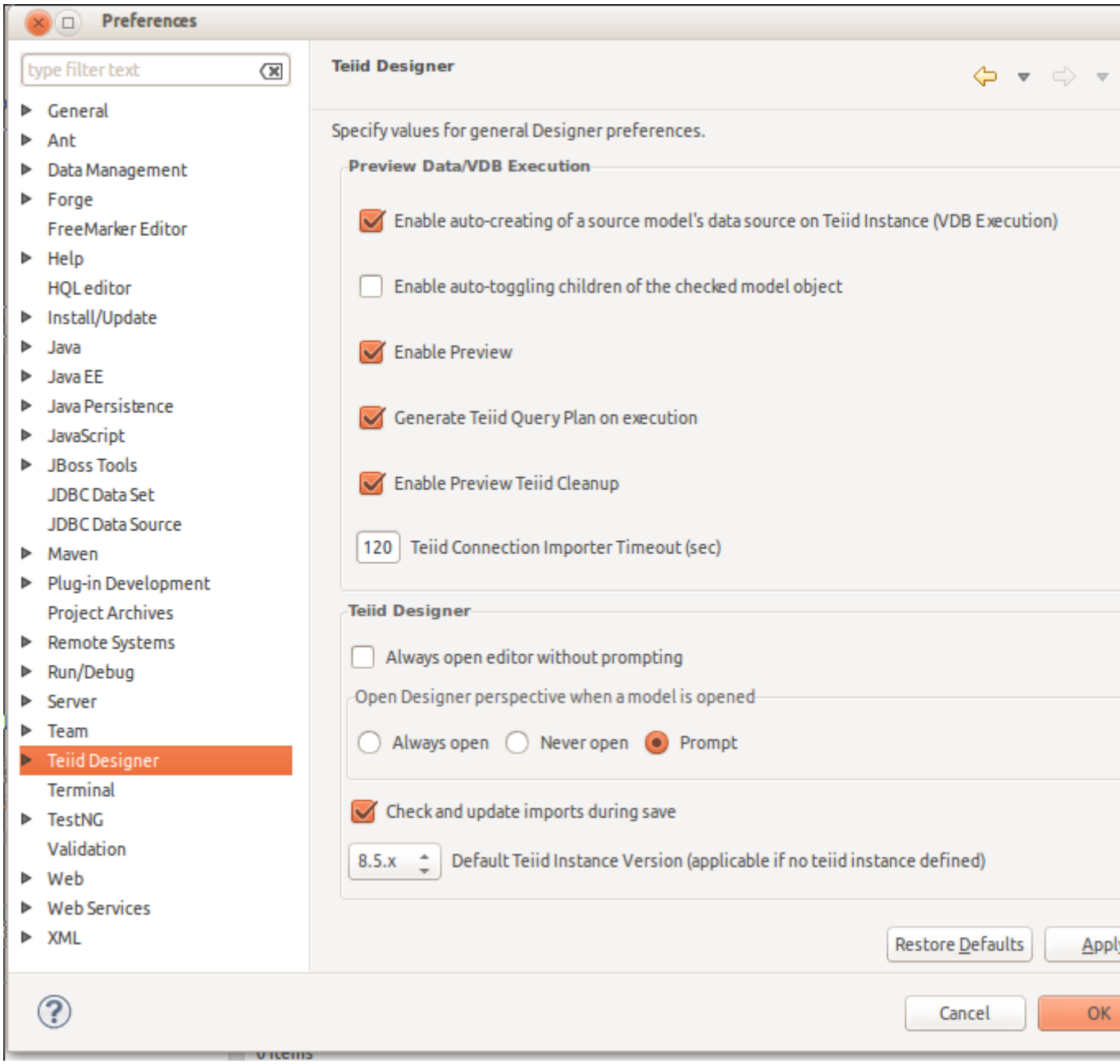


Figure 2.1. Default Server Version Preference

2.1.2. Defining a Teiid Server

The defining of a Teiid Server is encouraged since it allows for models to be previewed and their deployment tested. There is no limit to the number of servers that can be defined. However, the default server will always be used for previewing and deployment, unless using the context menu actions in the *Section D.2.3, "Server View"*.

The Guides View provides the following Teiid Server actions.

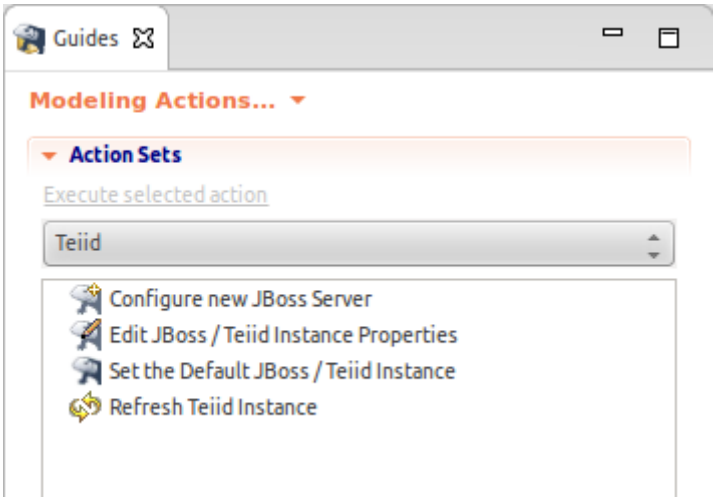


Figure 2.2. Teiid Server Category in the Guides View

The *New Teiid Server* action will display the wizard outlined in *???* and steps through the process of creating both the Teiid Server and its parent JBoss server in the Server View.

Should more than one Teiid Server be defined in the Server View then the *Set the Default Server* action allows for the default server to be changed appropriately. If a Teiid Server is currently selected in the Server View then this will be selected as the default server. However, should nothing be selected then a dialog will be displayed inviting the user to choose which server they wish to select.

Note

The version of the defined Teiid server always takes precedence over the *Section 2.1.1, "Server Version Preference"*

2.1.3. Server Version Status Panel

Whether the server version preference has been modified or a server defined, the server and server target version will be updated in the default server status panel. This will always reflect the current server version being targeted and the server being used to preview or deploy against.

Figure 2.3. Default Server Status Panel

2.2. Guide Example

In this section, the **Guides View** is demonstrated in detail by walking through a simple example. For this example, we will follow the **Model JDBC Source** Action Set. The actions appear in the following order:

1. **Define Teiid Model Project**
2. **Create JDBC connection**
3. **Create source model for JDBC data source**
4. **Preview Data**
5. **Define VDB**
6. **Execute VDB**

The action names are self explanatory. We will create a new "Model Project" in the workspace, then define our connection properties to a MySQL database. We will then connect to the database and import the 'metadata', creating a source model in Designer. Next we will 'preview' the database contents. Finally we will define a 'VDB' and then deploy it to a running Teiid Server to execute.

2.2.1. Model a Relational (via JDBC) Source

This section shows how to Model a Relatioanl Source, using the Guide View action set. We will connect to a MySQL database for this example, but you can use the same process to connect to any supported database.

1. **Open Guides View**

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

   The Guides view is shown below, with the **Model JDBC Source** Action Set selected:

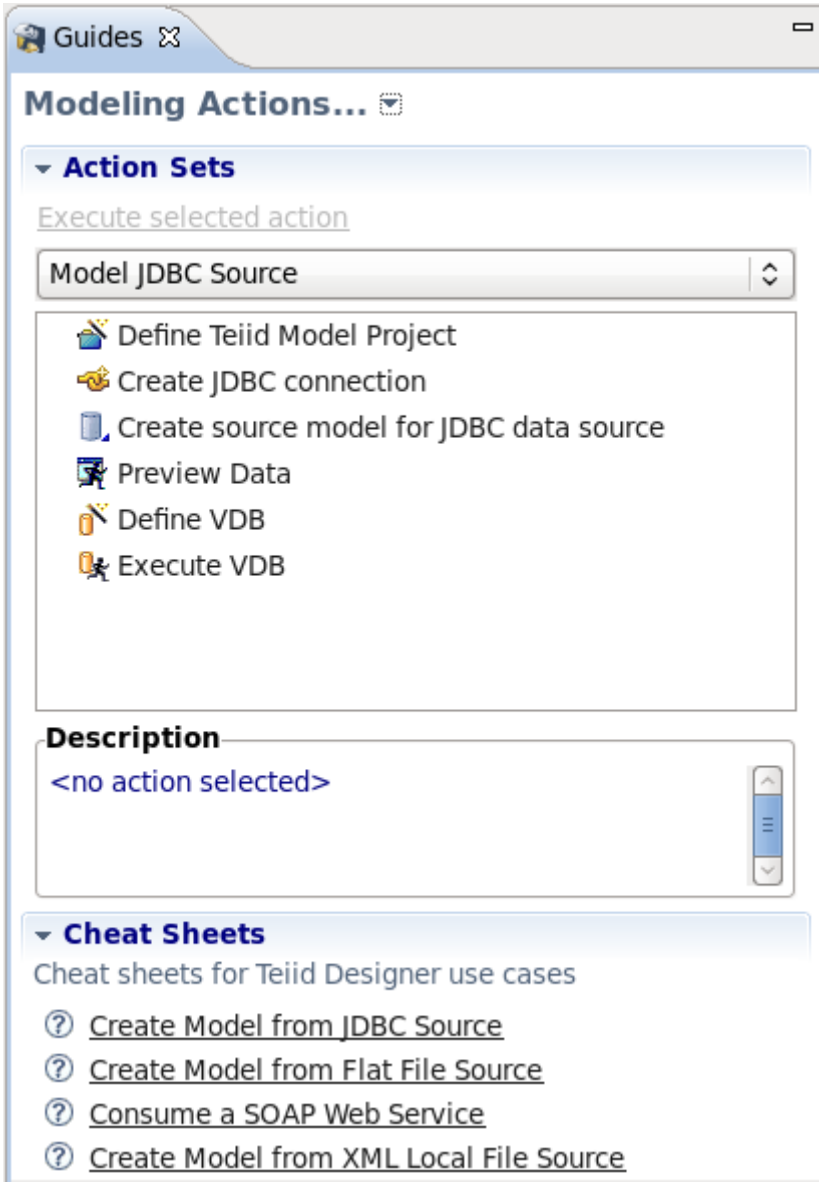


Figure 2.4. Guides View

2. **Define Teiid Model Project**

   The **Define Teiid Model Project** action launches the **New Model Project Wizard**. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

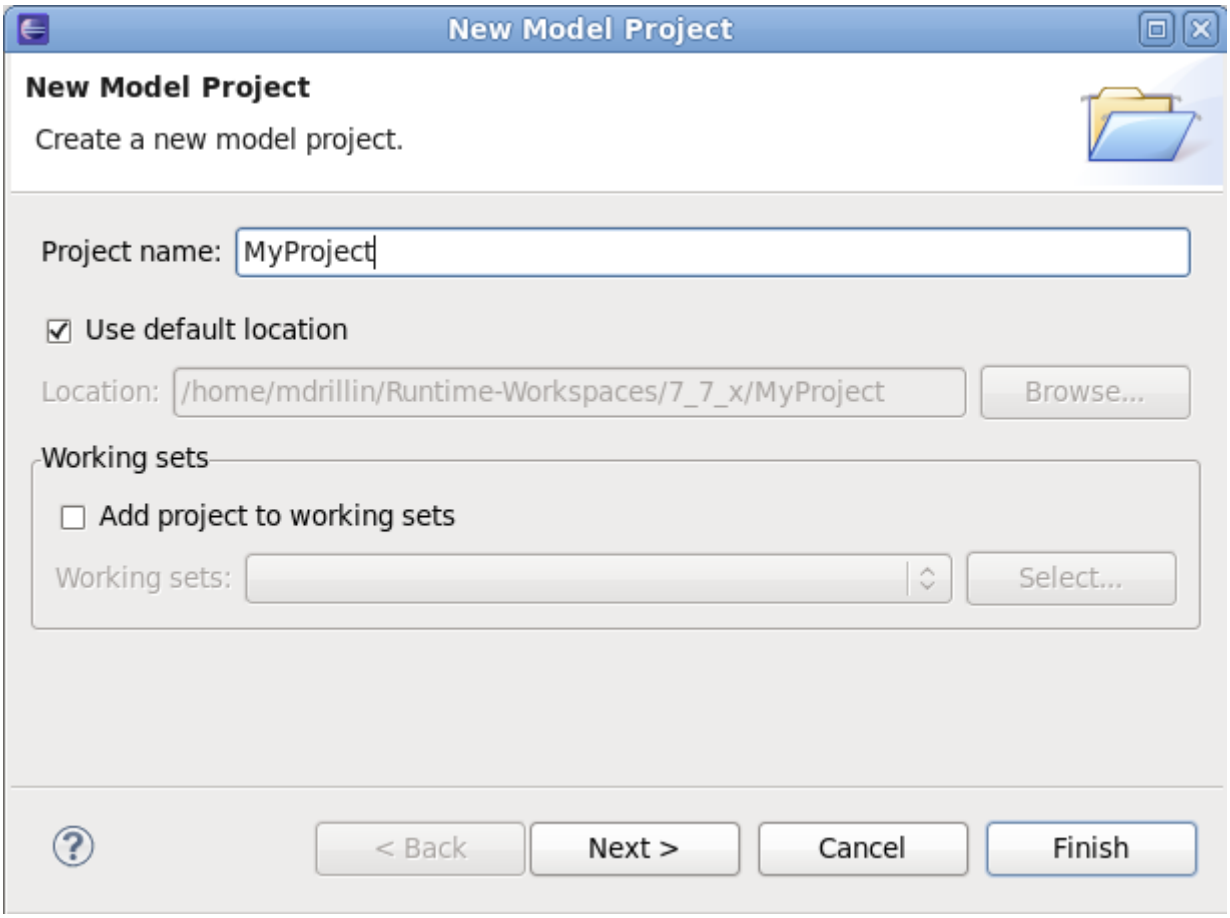


Figure 2.5. New Project Wizard

Enter a project name, e.g. 'MyProject' for the name. Then click **Next**. The next page of the wizard is shown below:

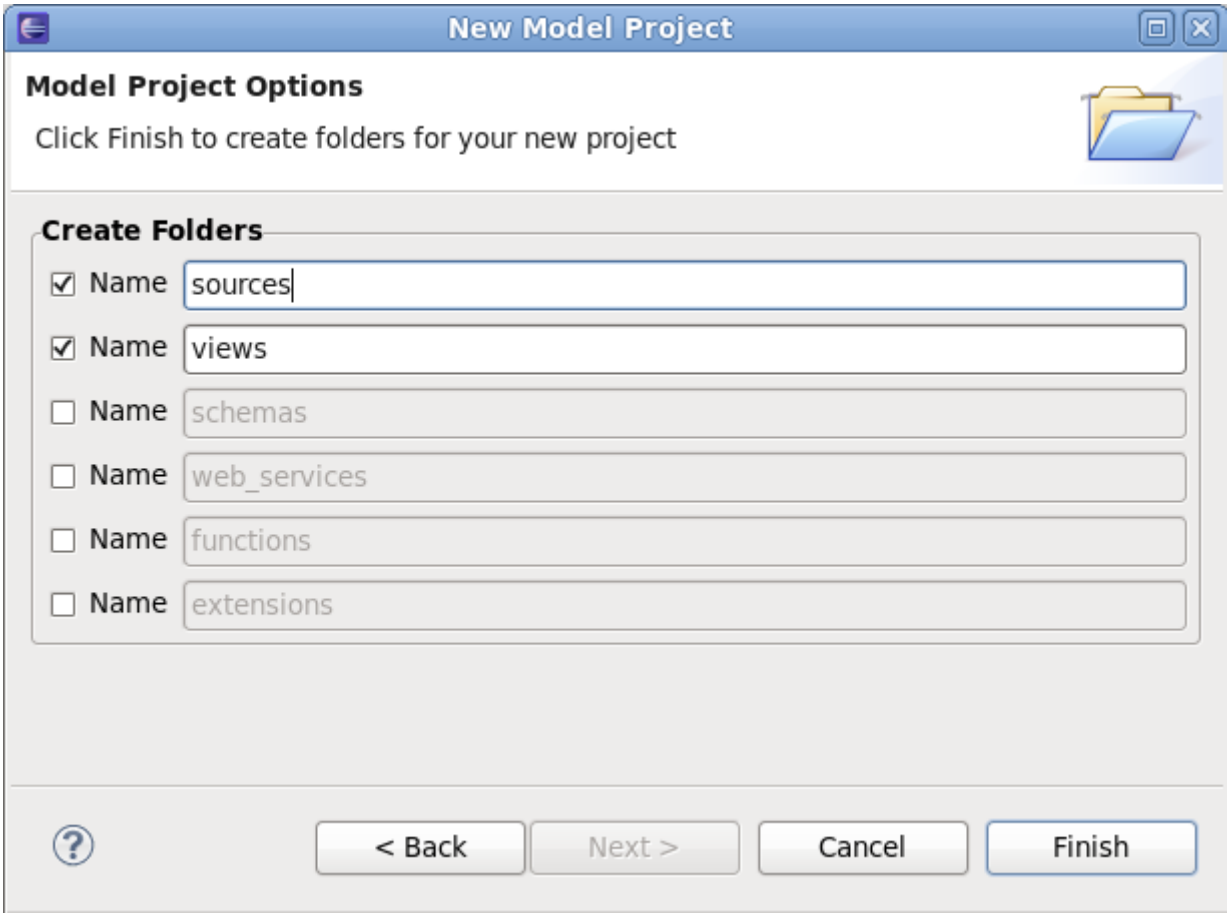


Figure 2.6. New Project Folders

Under 'Create Folders', de-select 'schemas' and 'web_services' - we won't need them for this example. Now, click **Finish** to exit the wizard. The project has now been created - your Model Explorer view should like like this:

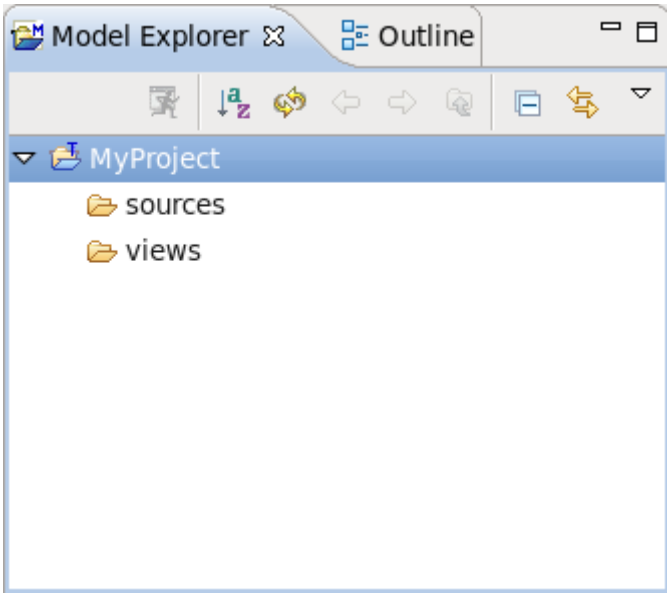


Figure 2.7. Model Explorer

3. **Create JDBC connection**

   The **Create JDBC connection** action will create the 'Connection profile' for your database. The connection profile defines the properties and driver to be used when connecting to the database. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

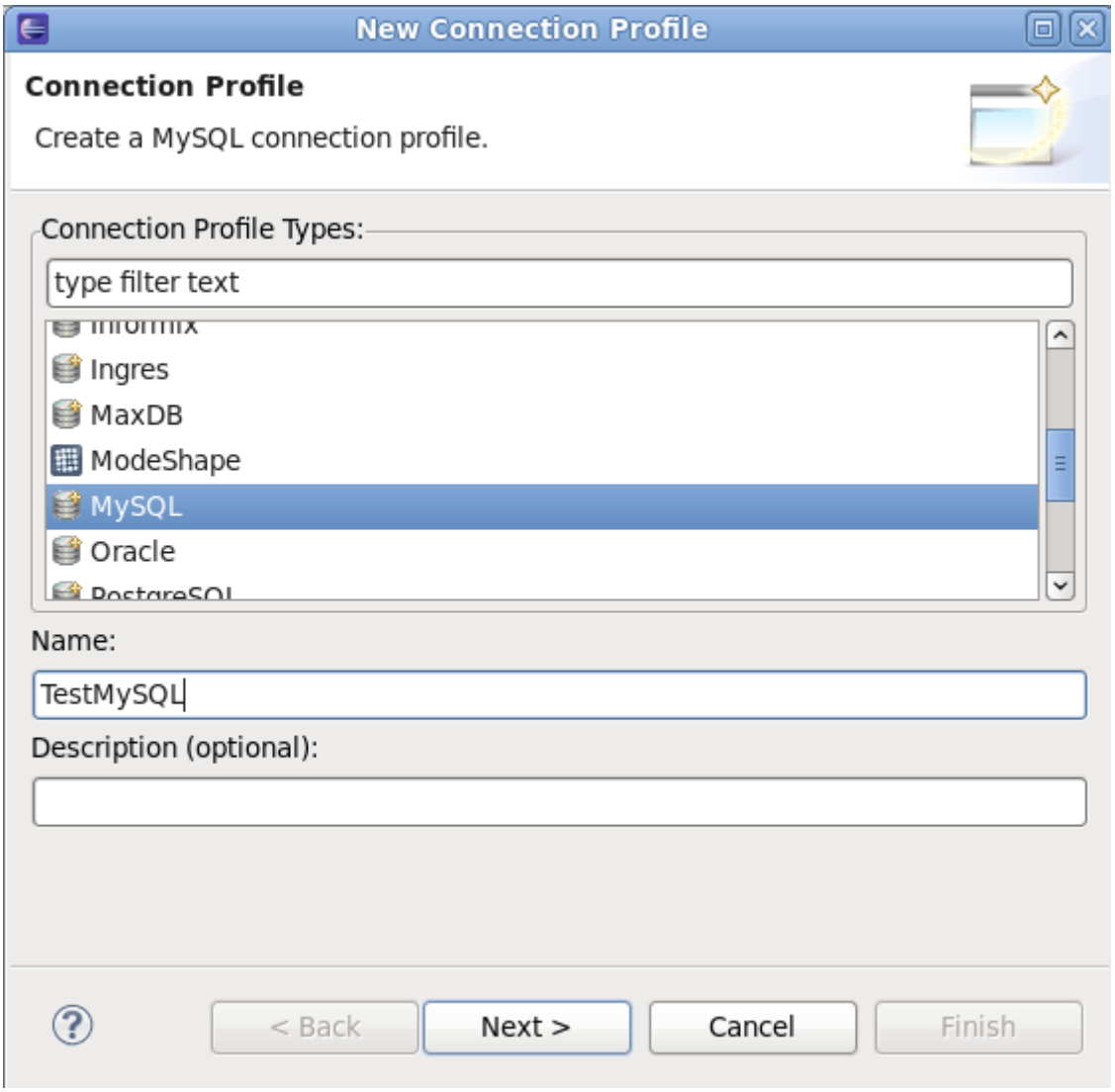


Figure 2.8. Connection Profile Name and Type

Select the type of database that you are connecting to (e.g. MySQL), and enter a name for the connection profile, e.g. 'TestMySQL'. Click **Next**.

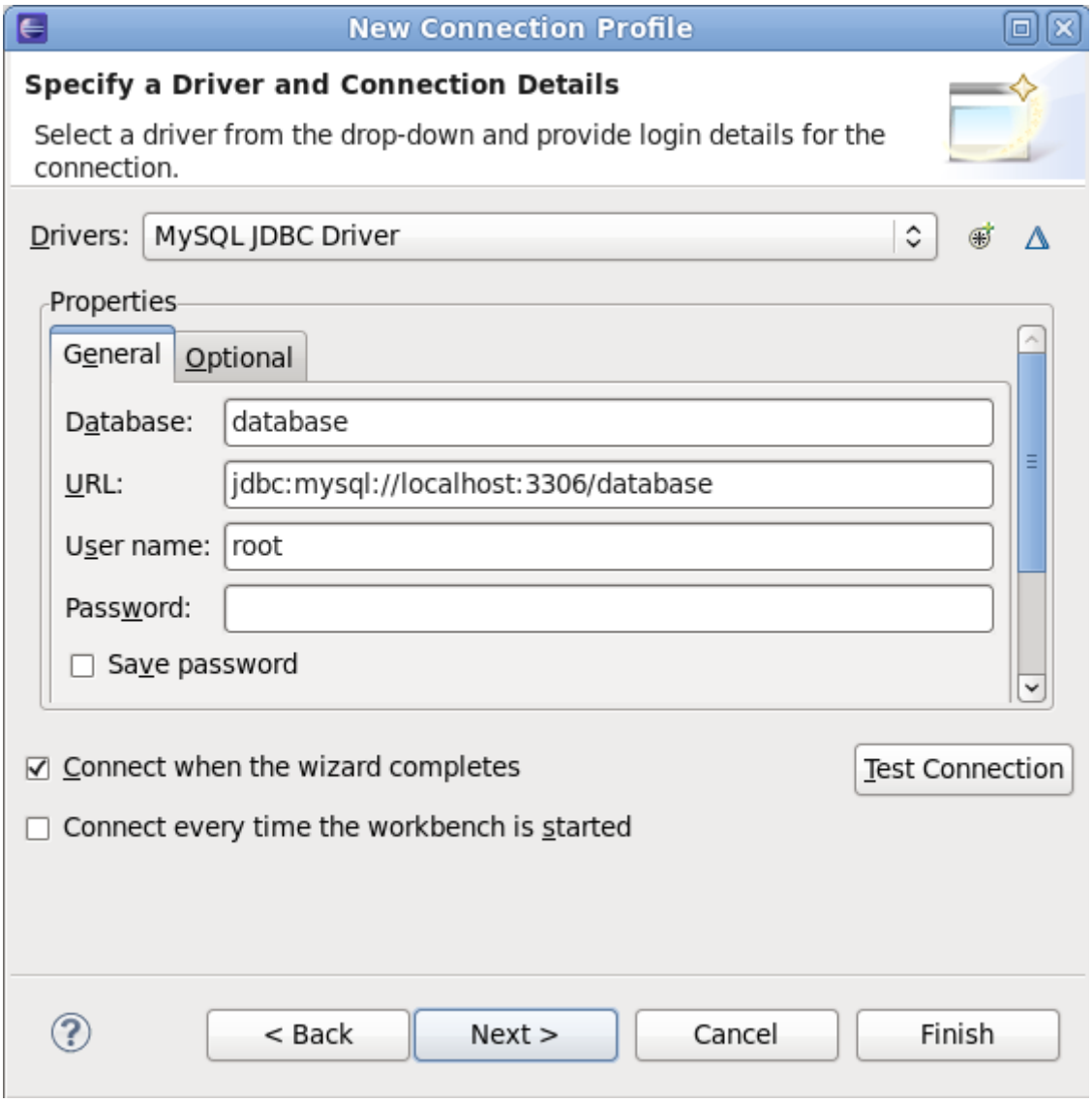


Figure 2.9. Connection Profile properties

Now, select the driver and enter the login properties for your database. Click **Finish** to complete the profile creation.

4. **Create source model for JDBC data source**

   The **Create source model for JDBC data source** action will now utilitze the **Connection profile** that you just created, to import the metadata from the database to create your Teiid Source Model. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

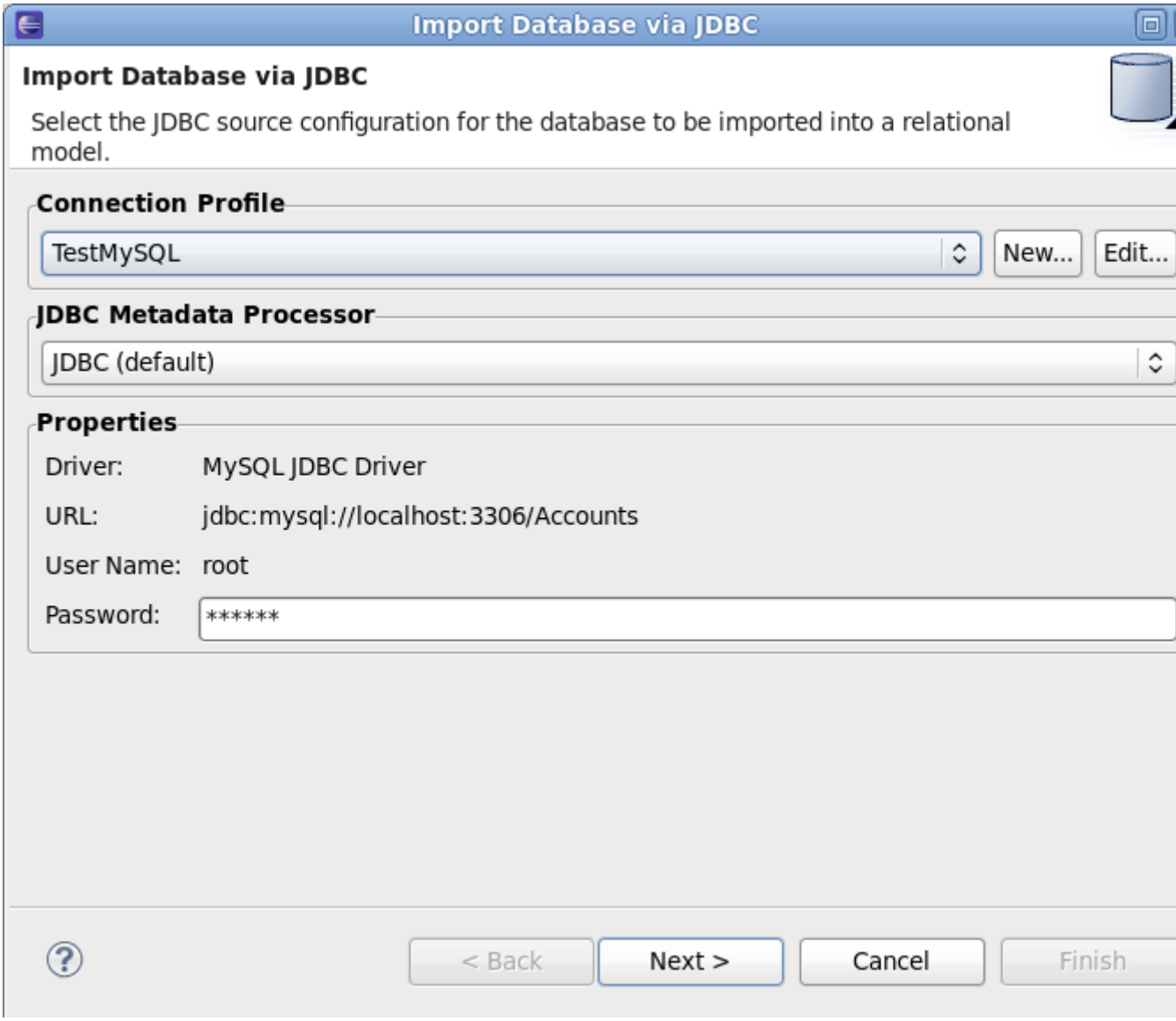


Figure 2.10. Select Connection Profile

On this page, select the 'TestMySQL' Connection profile that you created in the previous step. Click **Next**.

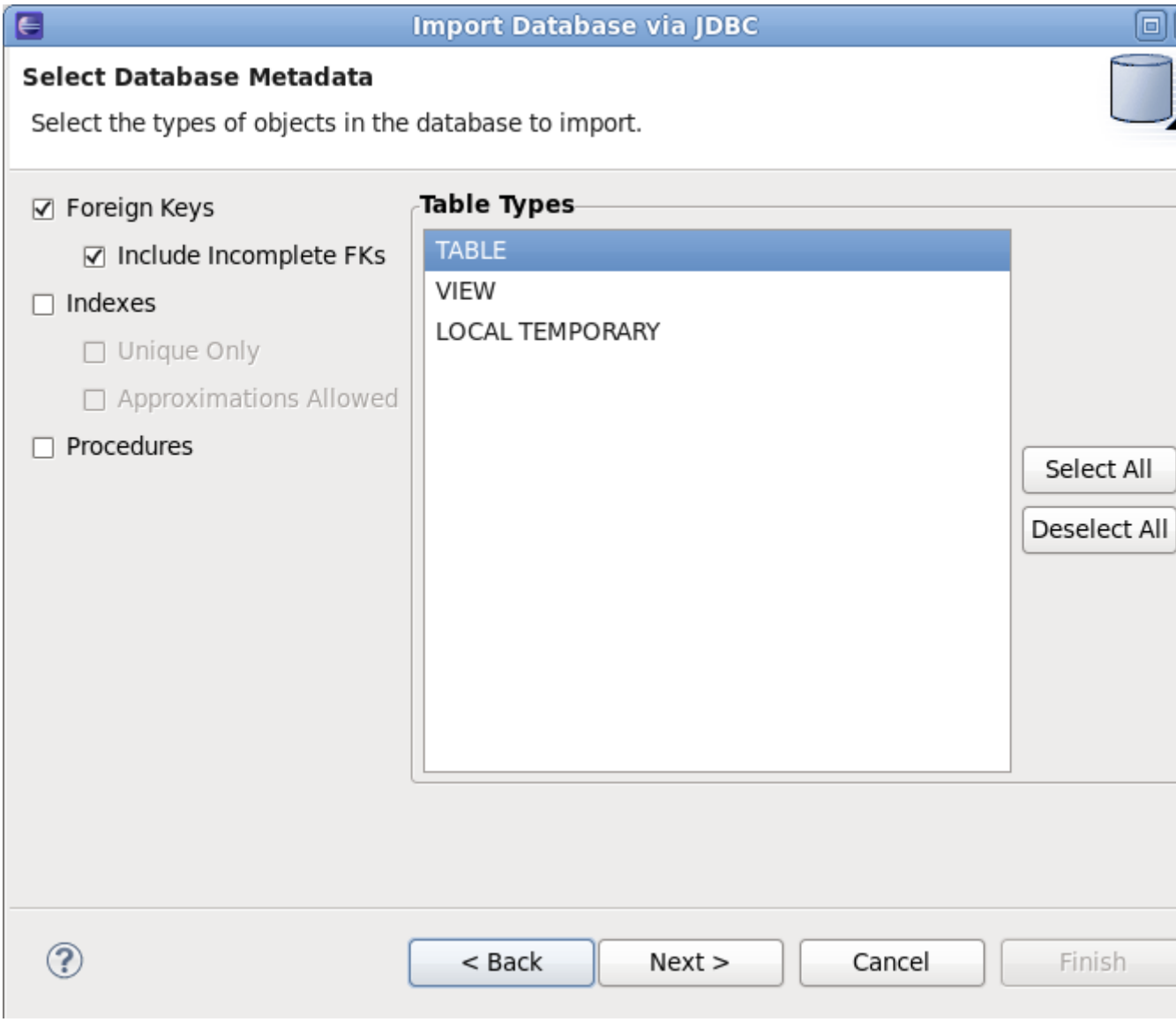


Figure 2.11. Select Database Metadata

On this page, select the database metadata that you want to import. When finished, click **Next**.

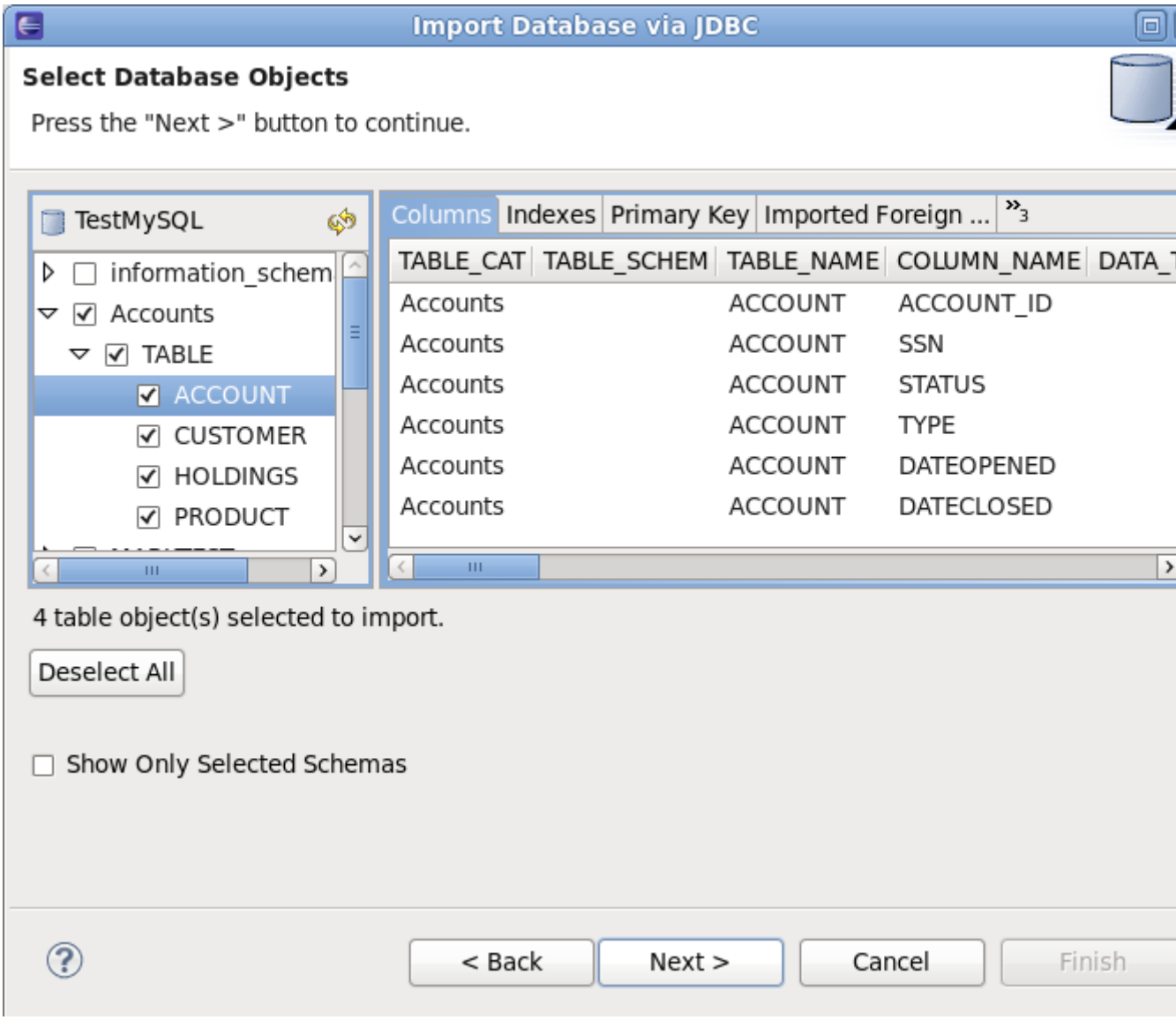


Figure 2.12. Select Database Objects

On this page, select the specific objects from the database that you want to import. When finished, click **Next**.

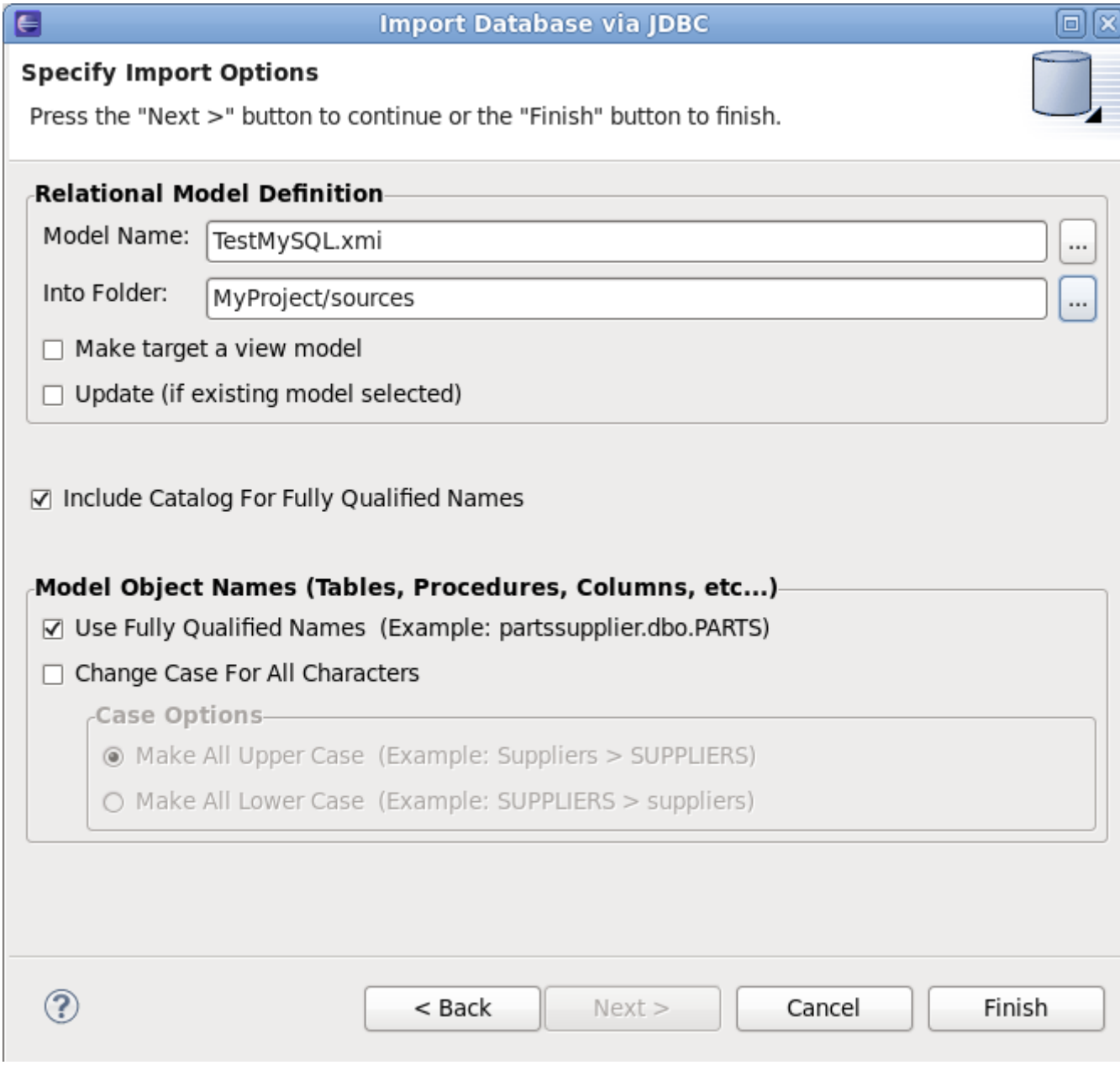


Figure 2.13. Import Options

Finally, choose the name for the model to be created (defaults to 'profileName'.xmi). The 'Into Folder' field defines the target location for your new model. Select the 'MyProject/sources' folder. Now, click **Finish**. The source model has now been created - your Model Explorer view should like like this:

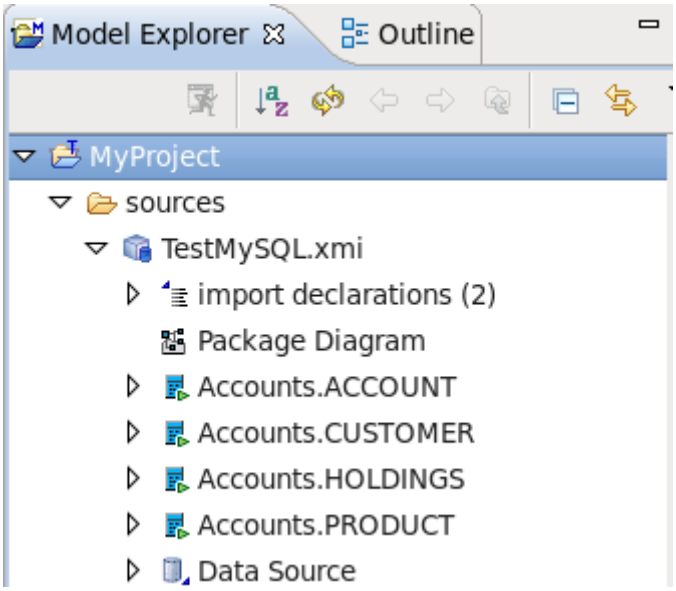


Figure 2.14. Model Explorer

5. **Preview Data**

   All execution capabilities in Designer (Preview Data, VDB execution) require you to connect to a running Teiid Server. See *Chapter 3, Server Management* for instructions on establishing a Teiid Server connection. Once you are connected to a Teiid Server, you can proceed with the following steps.

   The **Preview Data** action allows you to preview a sample of data rows from your source. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). In the dialog, select the source table you want to preview, as shown below:

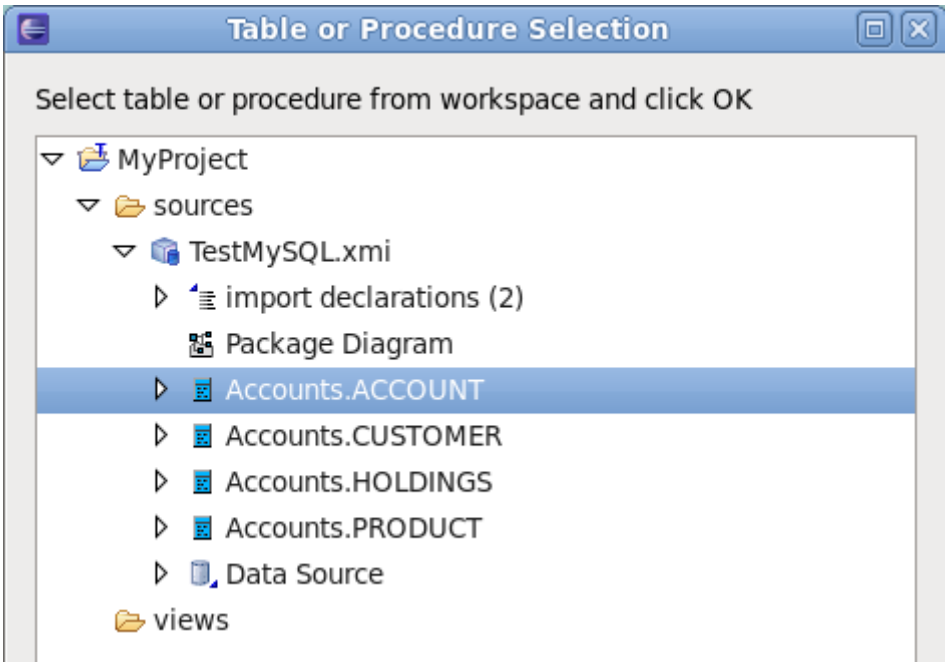


Figure 2.15. Select Preview Table

After selecting the table, click **OK**. Now, the preview results will be displayed:

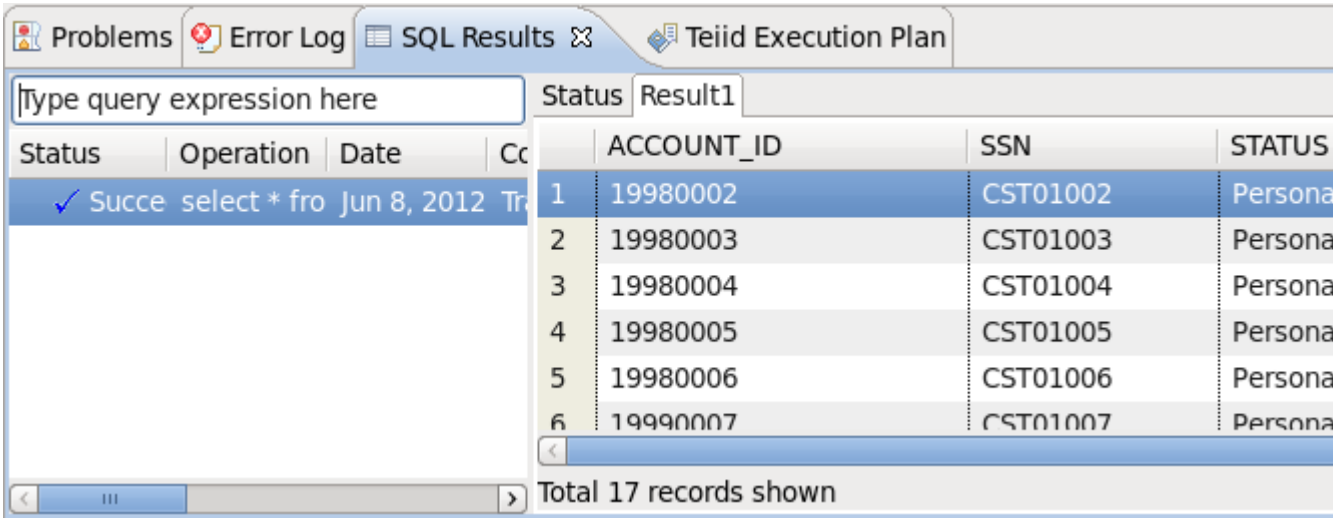


Figure 2.16. Preview Results

6. **Define VDB**

The **Define VDB** action allows you to create a VDB (Virtual Database) artifact for deployment to a Teiid Server. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The following dialog is displayed:

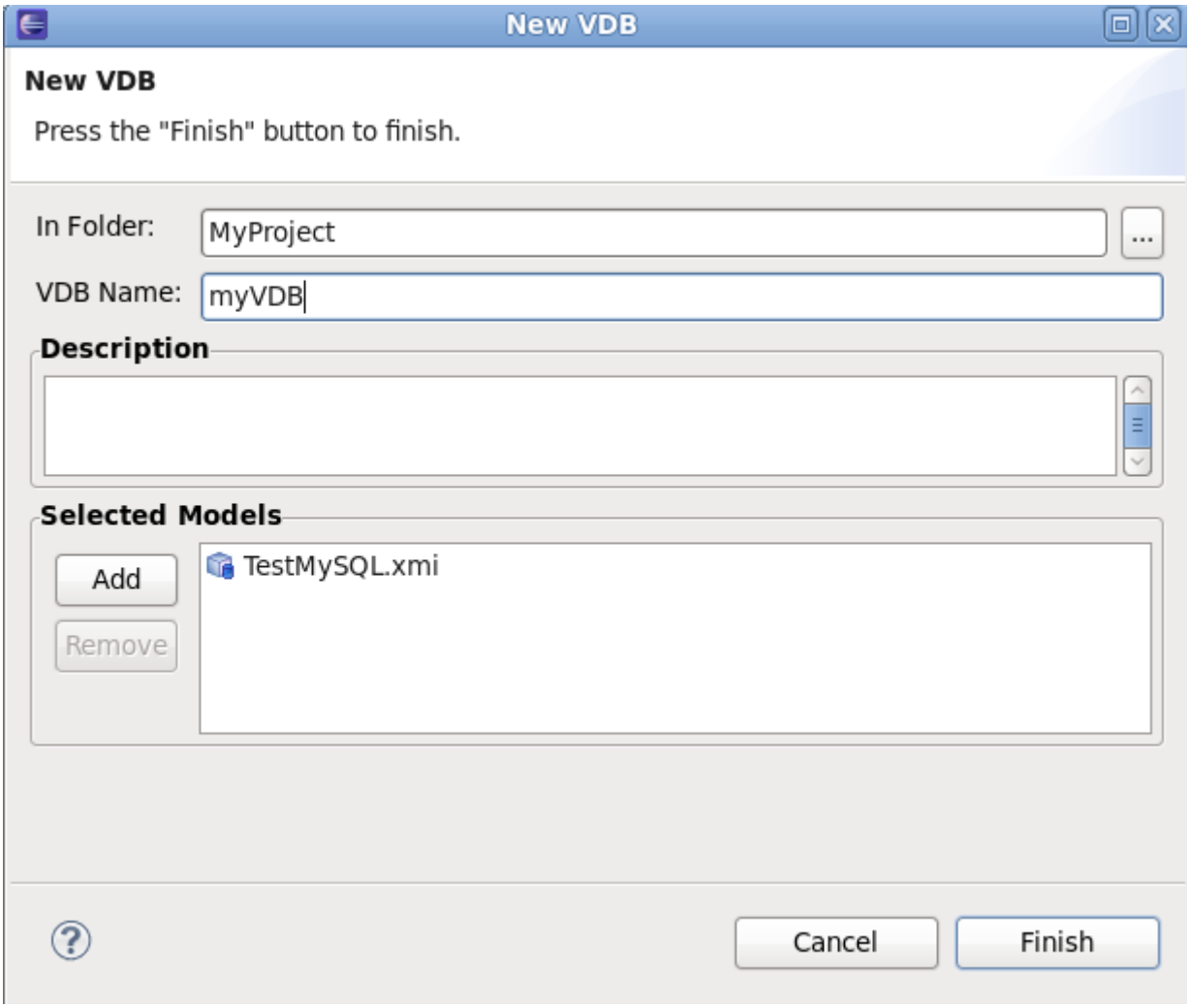


Figure 2.17. New VDB

In the dialog, select the target 'In Folder' location where the VDB will be placed. Enter a Name for the VDB, for example 'myVDB'. Finally, select the models that will be included in the VDB. When finished, click **Finish**. The VDB will be created in your Teiid Model Project - as shown in the following figure.

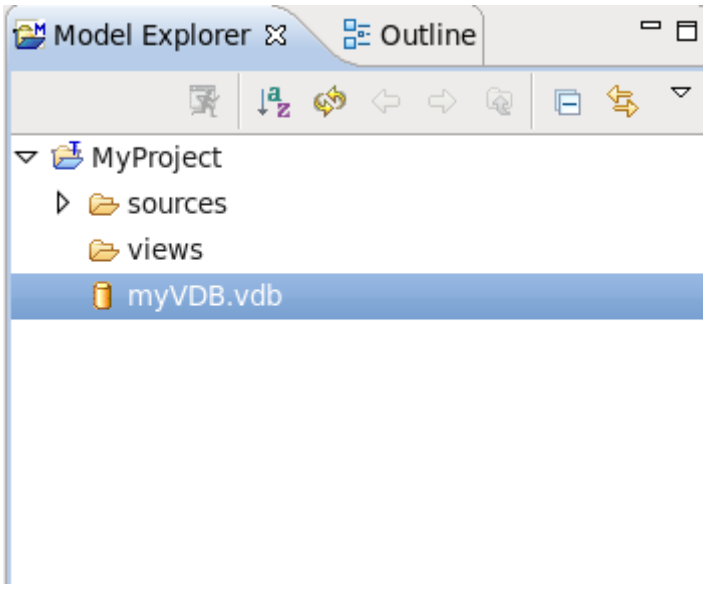


Figure 2.18. Model Explorer

7. **Execute VDB**

   Finally, the **Execute VDB** action allows you to execute your VDB and run sample queries against it. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). In the dialog, select the VDB you want to execute, then click **OK**. The VDB will be deployed and executed, and the perpective will switch to the 'Database Development' perspective. You can now run queries against the VDB, as show in the following example:

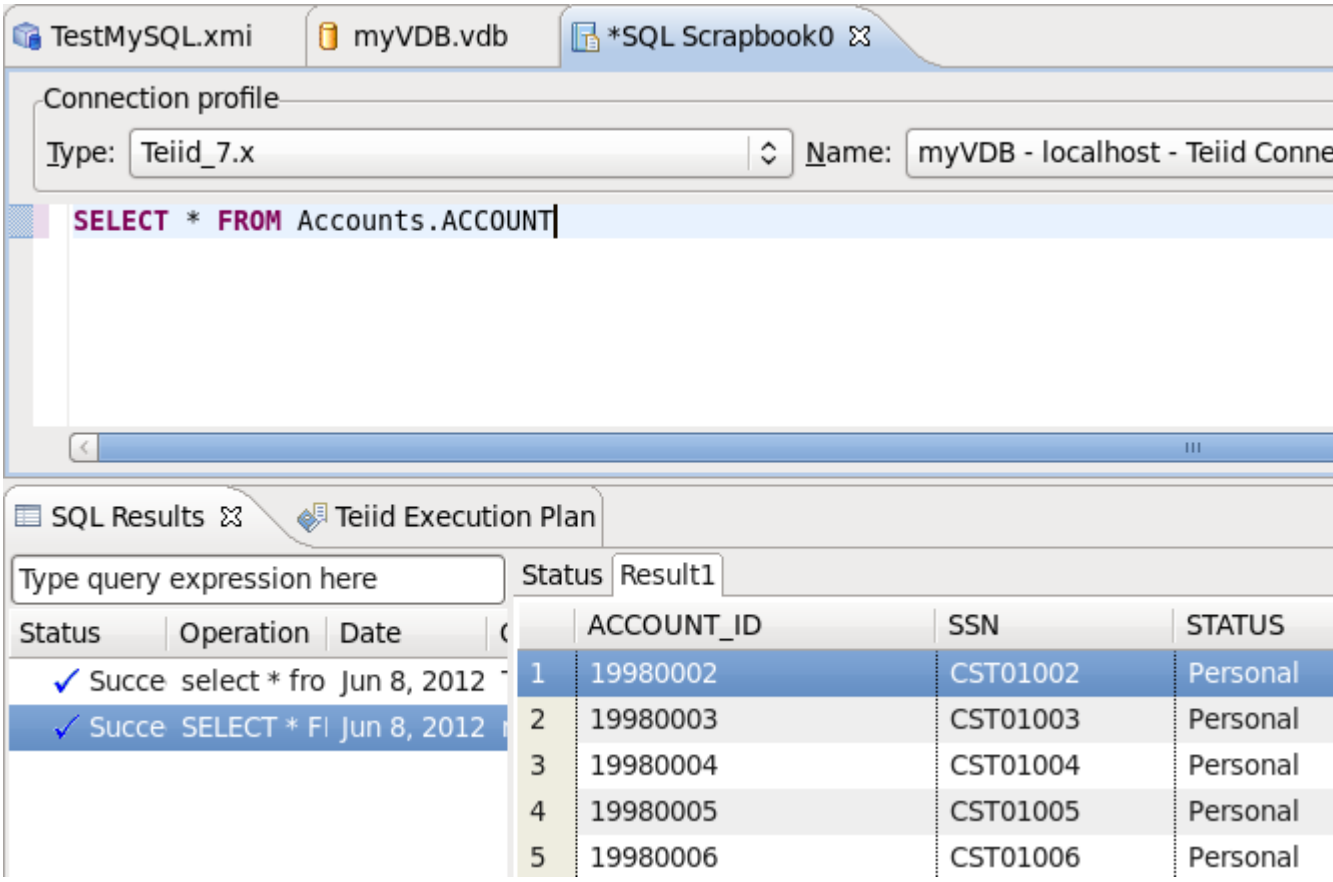


Figure 2.19. Execute VDB Example

2.3. Cheat Sheet Example

In this section, we introduce **Cheat Sheets** by walking through a simple example. For this example, we will follow the **Consume a SOAP Web Service** Cheat Sheet.

2.3.1. Consume a SOAP Web Service

This section shows how to consume a SOAP Web Service, using a Cheat Sheet. We will demonstrate connection to a publicly accessible web service. You can use this process as an example for modeling other web services

1. **Open the Cheat Sheet**

   You can access the **Cheat Sheet** from the Designer Menu. From the Designer main menu, select **Window > Show View > Other...**, then select **Help > Cheat Sheets** in the dialog.

   Alternately, you can access the **Cheat Sheet** from the **Guide View**. A sample Guide view is shown below, with the **Consume a SOAP Web Service** Action Set selected:

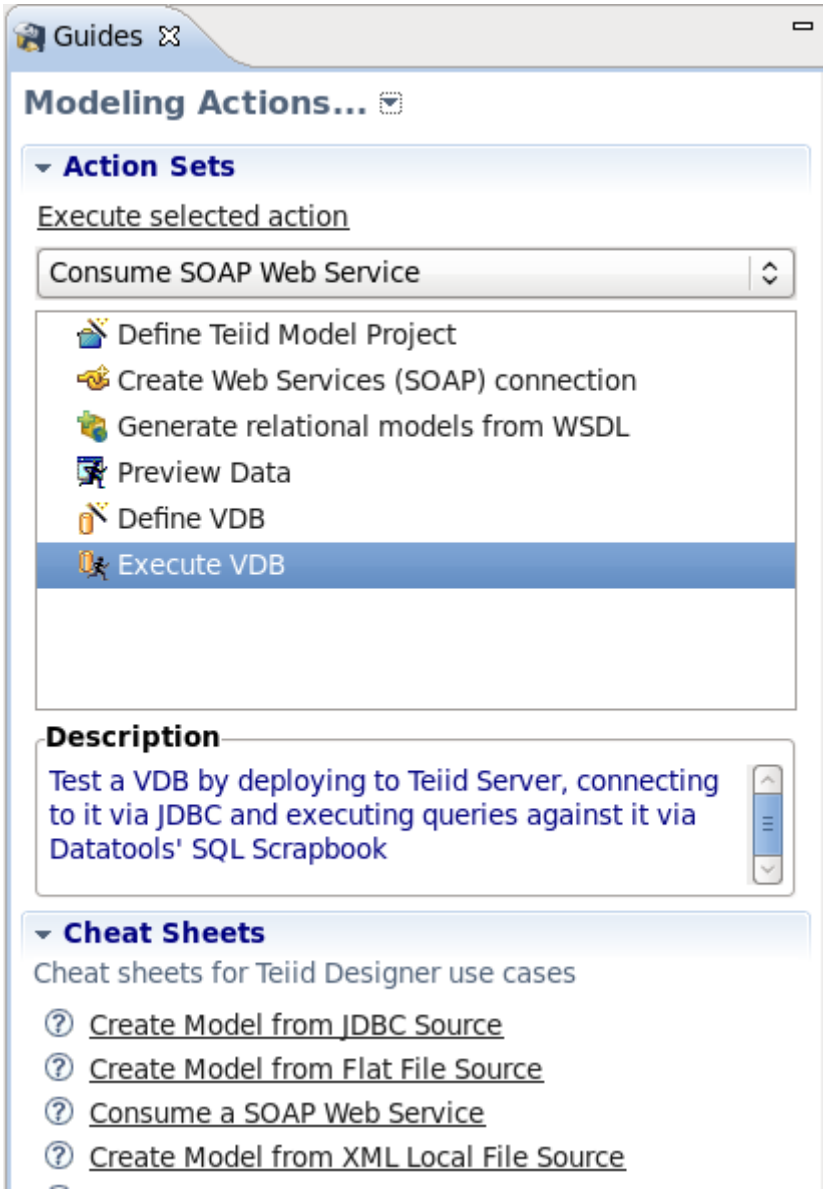


Figure 2.20. Guides View

To open the **Cheat Sheet** from the Guide View, expand the Cheat Sheet section in the lower portion of the Guide View, then select the **Consume a SOAP Web Service** link.

2. **Begin the Cheat Sheet**

The **Consume a SOAP Web Service** Cheat Sheet is shown below:

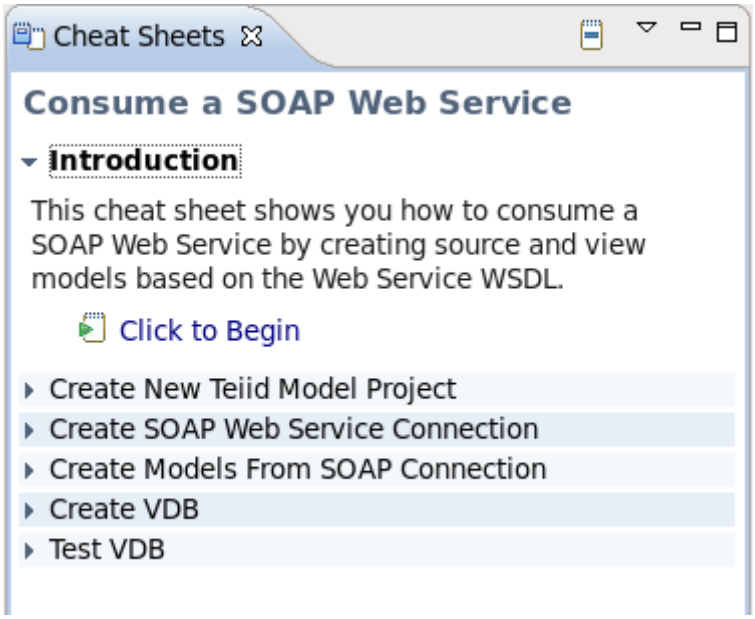


Figure 2.21. Consume SOAP Web Service Cheat Sheet

To start the Cheat Sheet process, expand the **Introduction** section, then select **Click to Begin**. The **Create New Teiid Model Project** section opens, as shown.

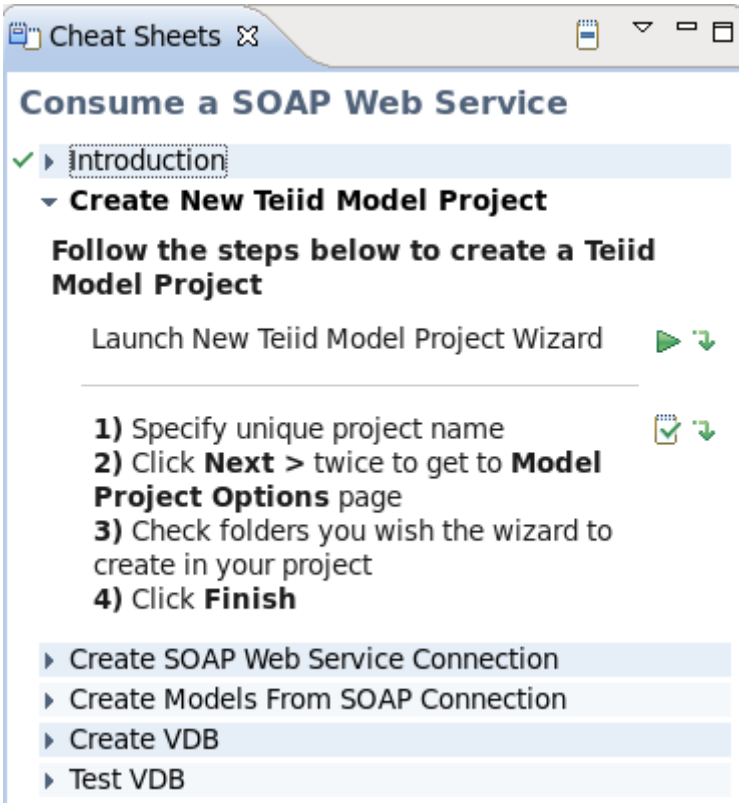


Figure 2.22. Create Model Project

Note

Each section of the sheet has basic instructions outlining what to do at each step.

Click

next to **Launch New Teiid Model Project Wizard** to launch the 'New Project' wizard.

Follow the wizard to create a new Model Project. For this example, we will use **SOAPProj** for our project name. On the second page of the wizard, select the 'sources' and 'views' folders. Click **Finish**. The new project is created.

In the Cheat Sheet, you can advance to the next step - once the wizard has completed. Click

to advance to the next step.

3. **Create SOAP Web Service Connection**

   This section of the Cheat Sheet provides instructions for creating a connection profile for the SOAP Web Service, as shown below:

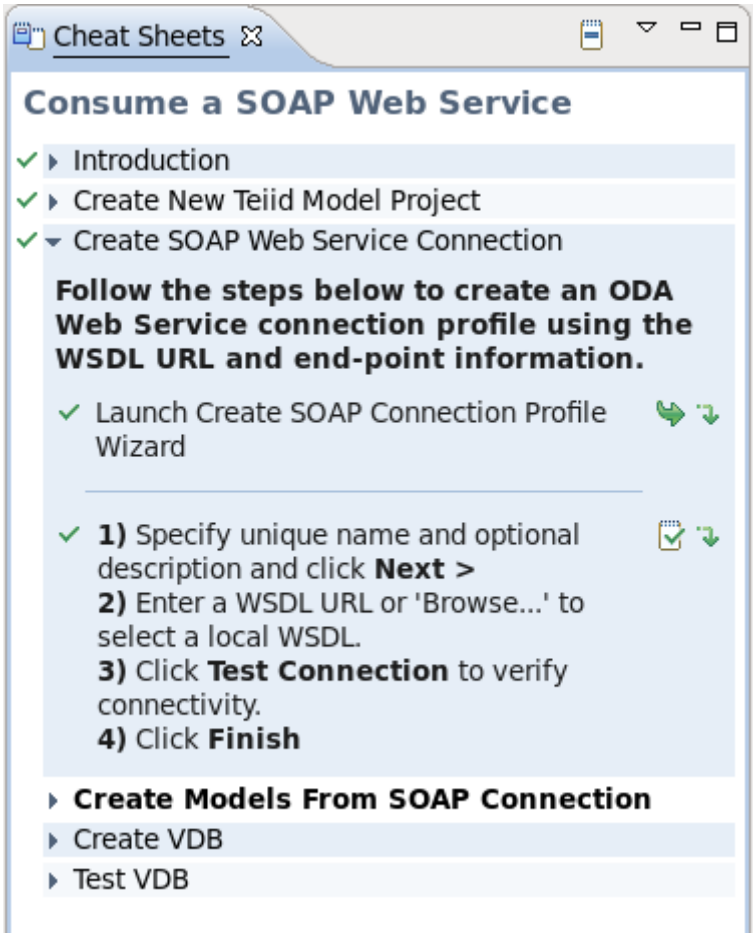


Figure 2.23. Create SOAP Connection Profile

Click

next to **Launch Create SOAP Connection Profile Wizard** to launch the wizard. The first page of the wizard is shown below:

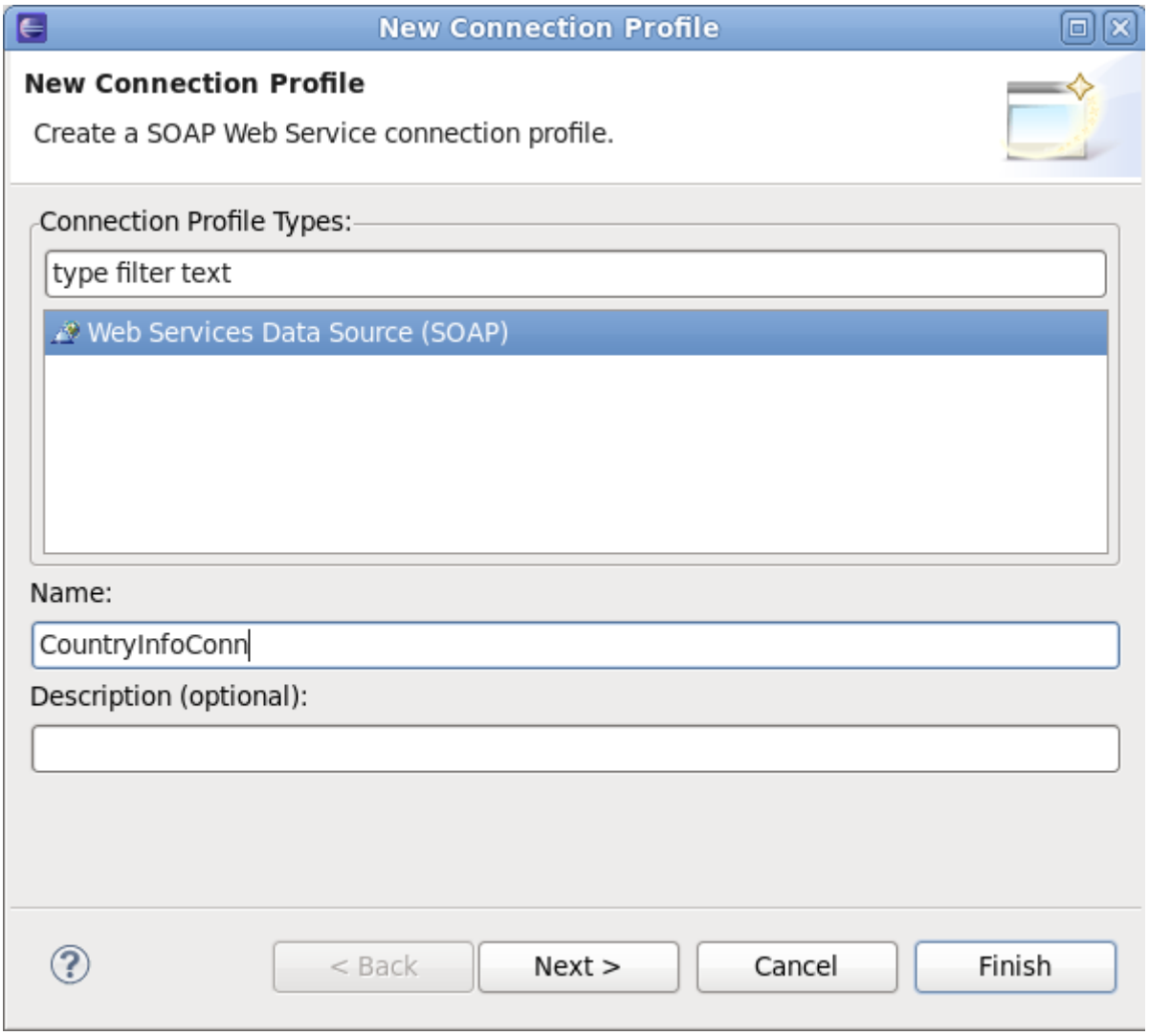


Figure 2.24. Create SOAP Connection Profile

The **Web Services Data Source (SOAP)** profile type will be selected. Enter **CountryInfoConn** for the profile name, then click **Next**. The next page of the wizard is shown below:

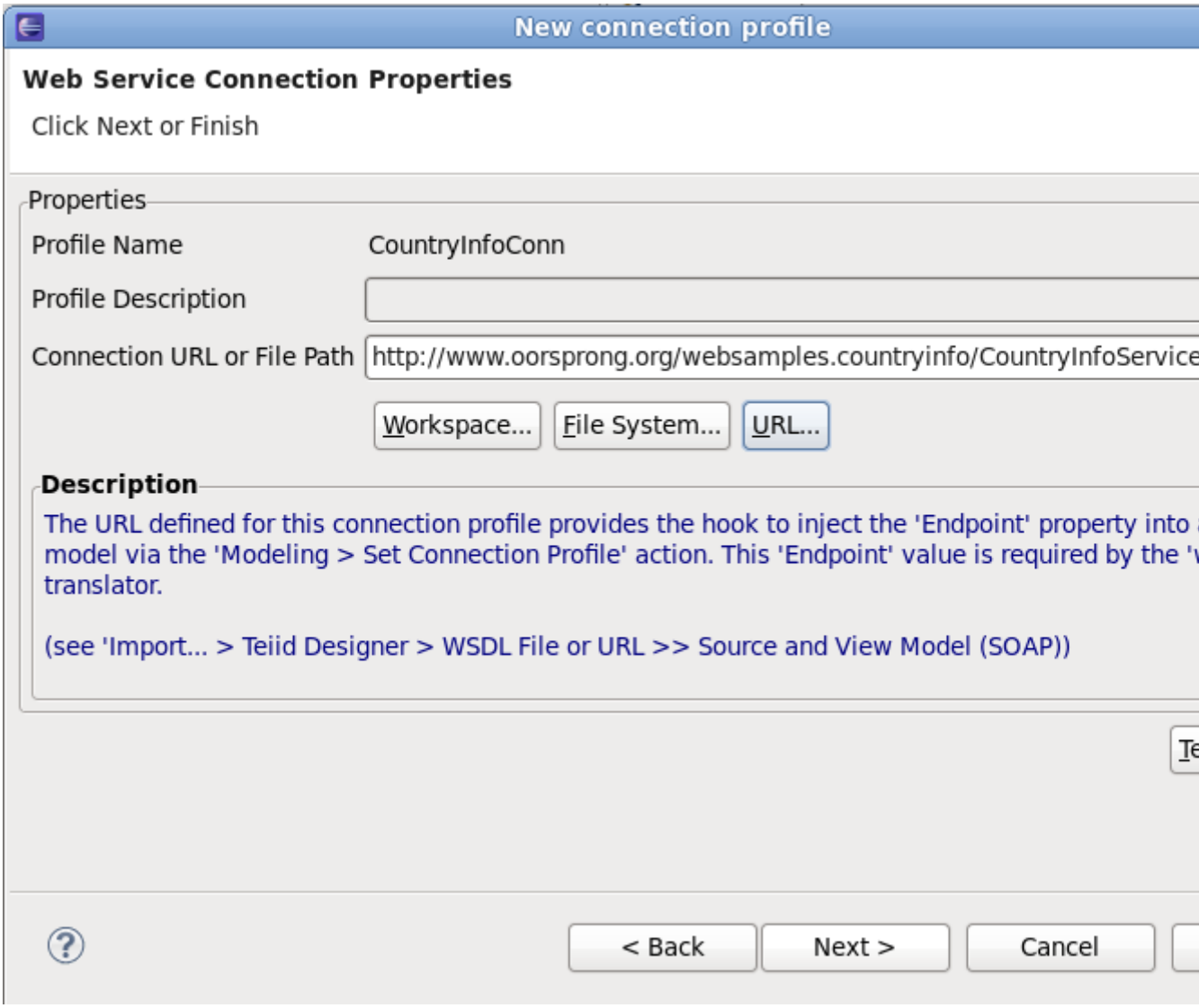


Figure 2.25. SOAP Connection Properties

The connection profile properties are entered on this page. Click on the **URL...** button, then enter the following URL: **http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL**

Select 'None' for SecurityType, then click **OK** to complete the wizard. In the Cheat Sheet, you can now continue - once the wizard has completed. Click

to advance to the next step.

4. **Create Models from SOAP Connection**

   This section of the Cheat Sheet provides instructions for creating relational models using the previously-created connection profile for the SOAP Web Service, as shown below:

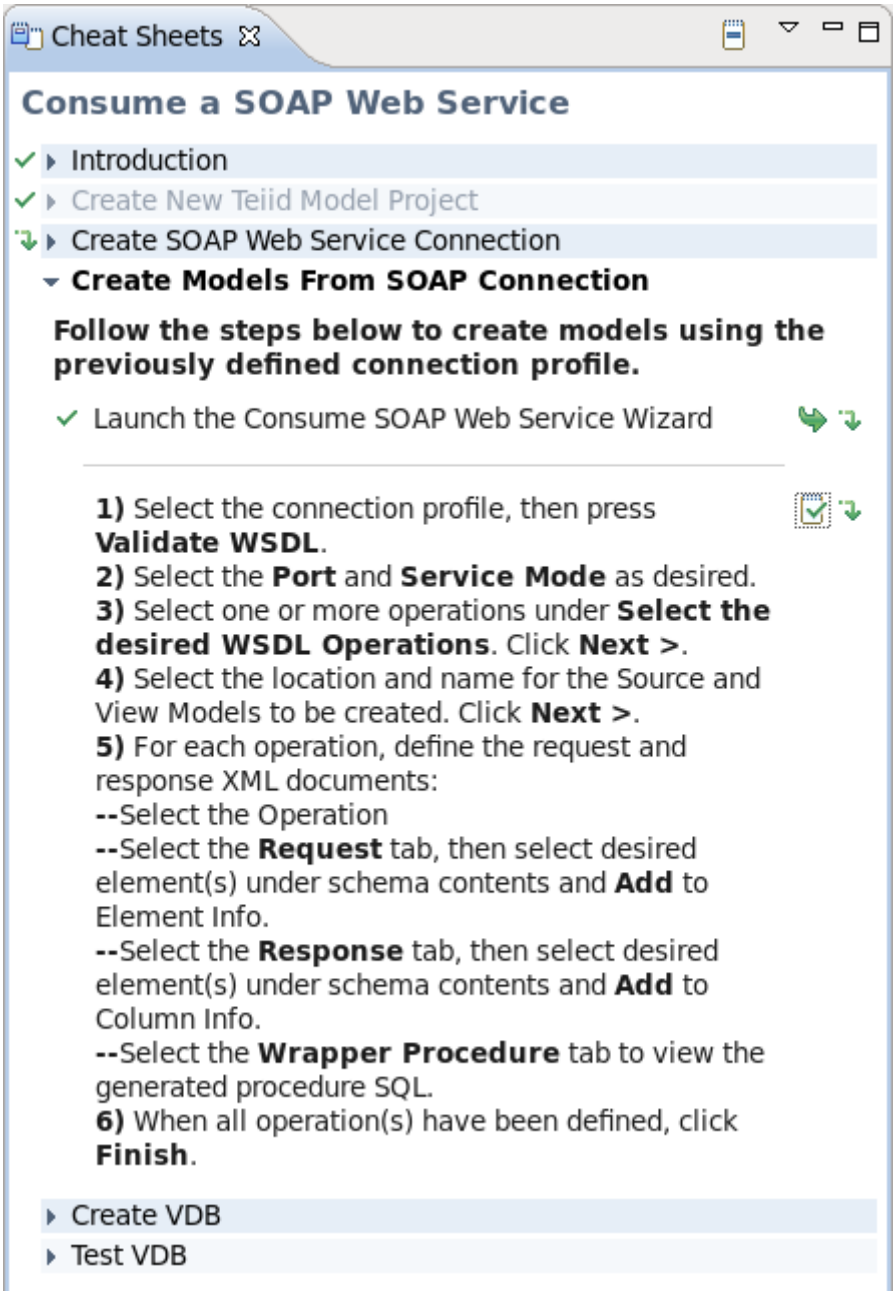


Figure 2.26. Create Models from SOAP Connection

Click

next to **Launch the Consume SOAP Web Service Wizard** to launch the wizard. The first page of the wizard is shown below:

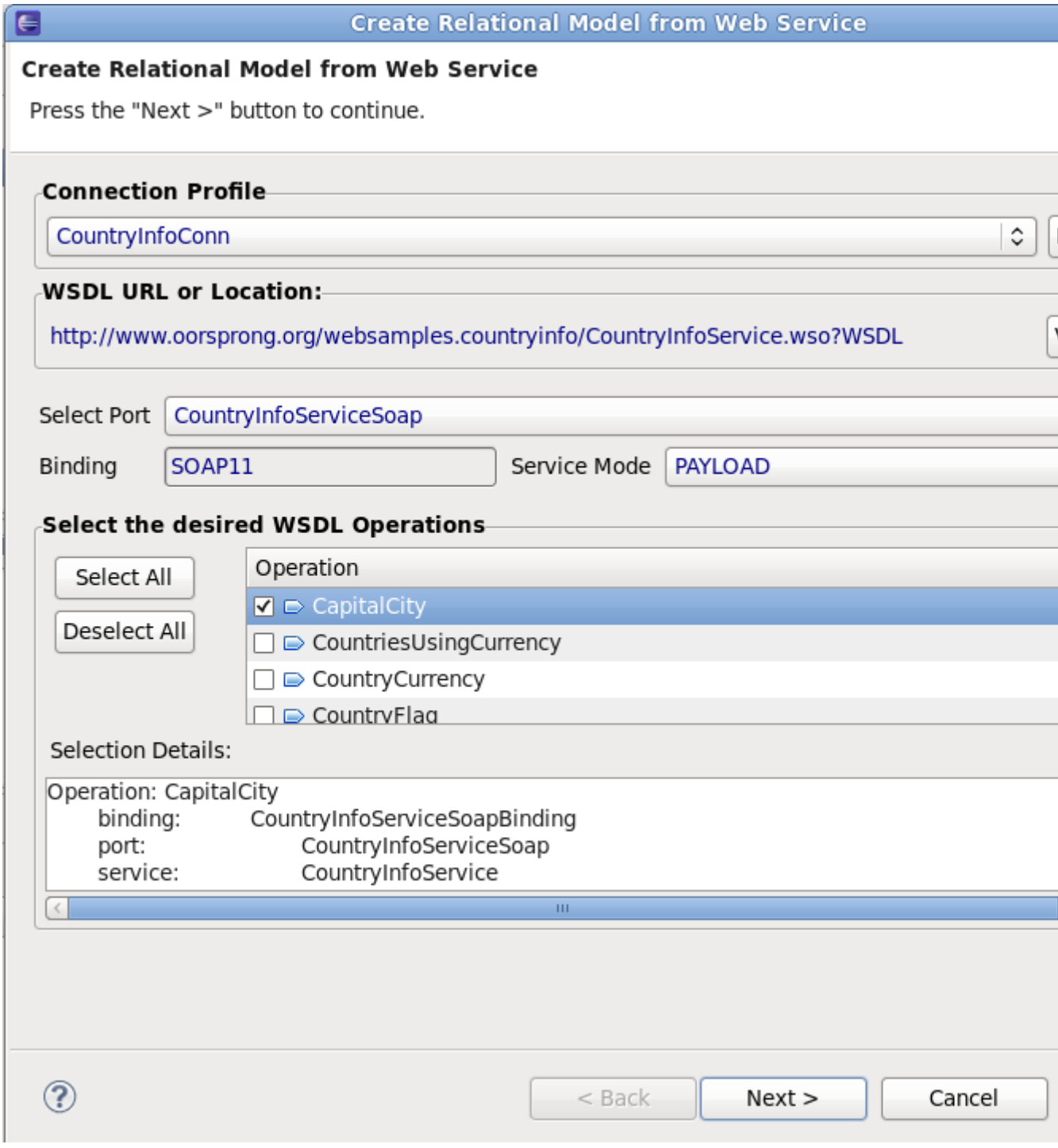


Figure 2.27. Consume SOAP Wizard

For **Connection Profile**, select the previously-created **CountryInfoConn** profile. The available WSDL Operations will then be displayed under **Select the desired WSDL Operations**. Select only the first **CapitalCity** Operation for this example. Click **Next** to proceed to the next page, as shown below:

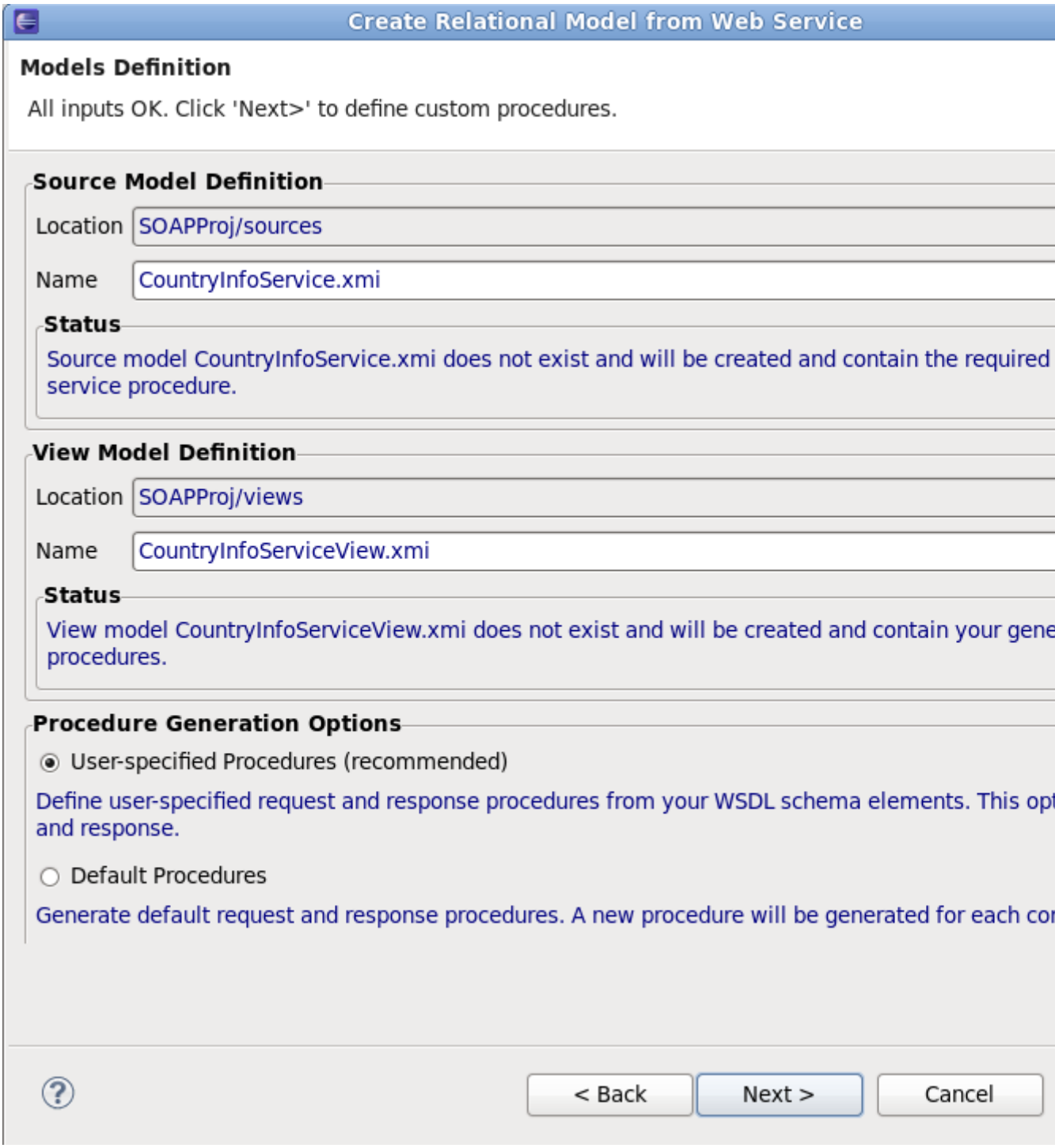


Figure 2.28. Consume SOAP Wizard

On the **Model Definition** page, the source and view model info section will be pre-filled. We will keep the names and location defaults for the source and view models. Click **Next** to proceed to the next page, as shown below:

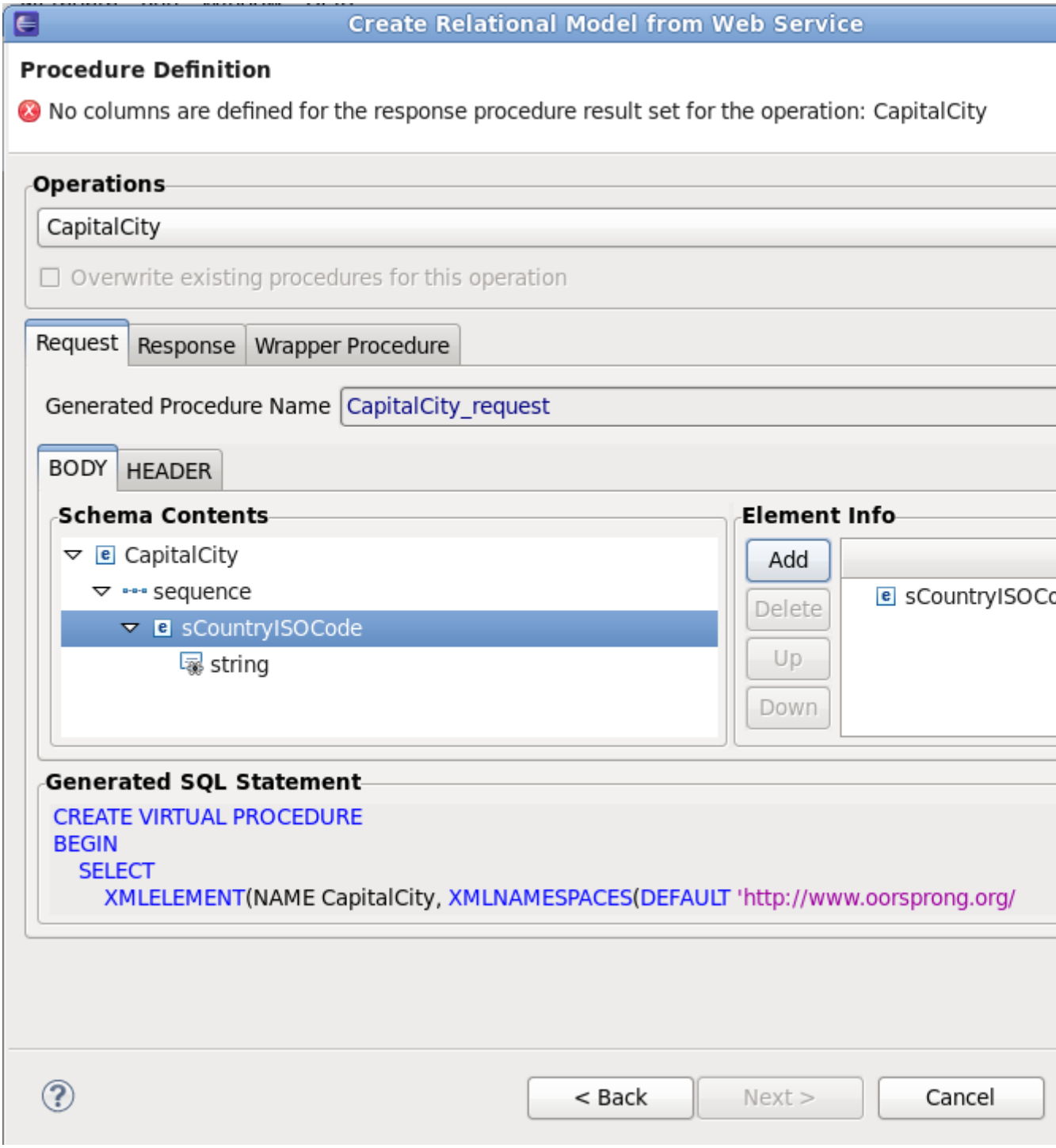


Figure 2.29. Consume SOAP Wizard

On the **Procedure Definition** page, the **CapitalCity** Operation will be selected since it is the only one used for this example. On the **Request** tab, select the **sCountryISOCode** element - then click the **Add** button. This will add the selected element to the request. Now select the **Response** tab, as shown below:

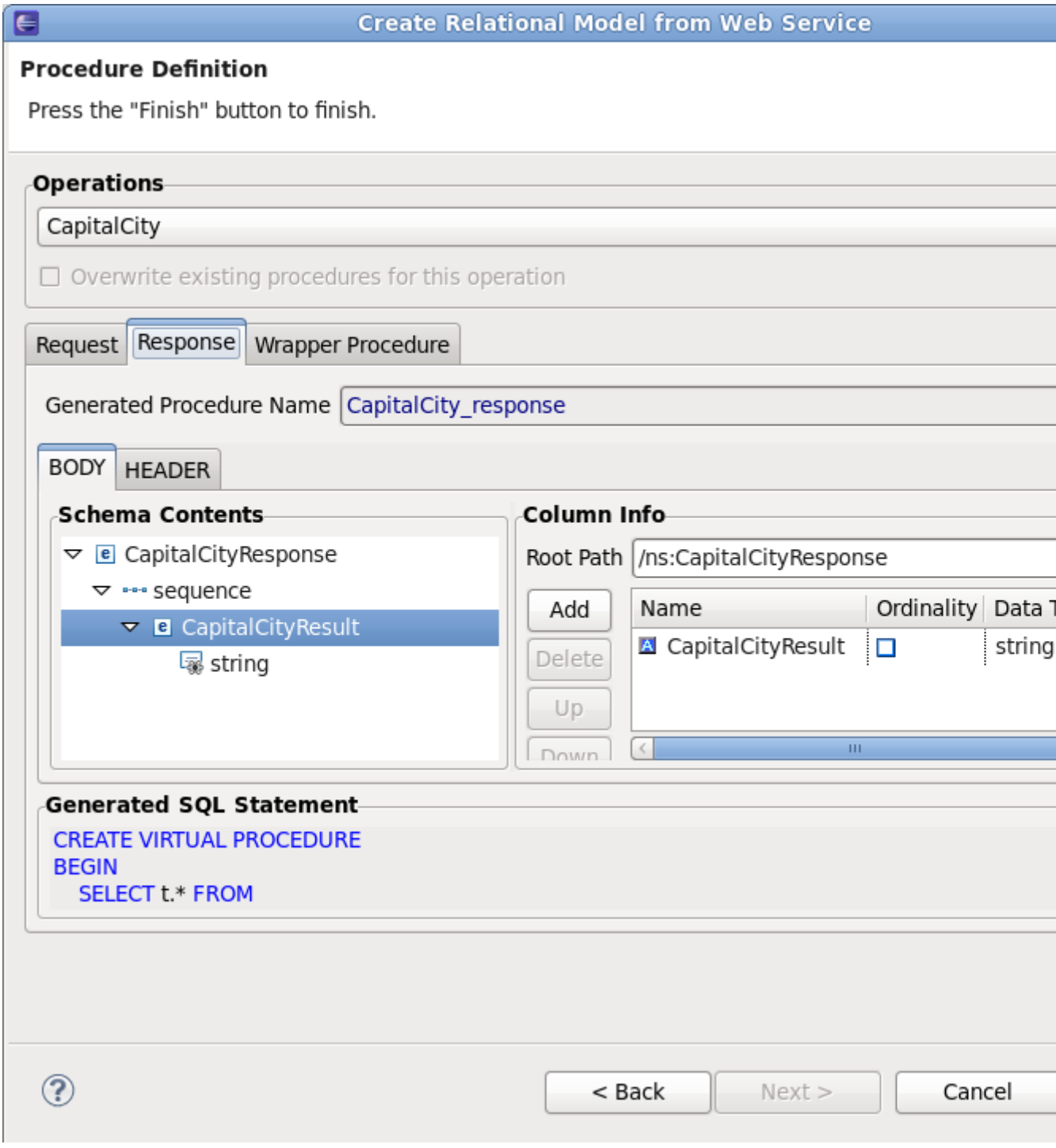


Figure 2.30. Consume SOAP Wizard

On the **Response** tab, select the **Body** sub-tab. In the **Schema Contents**, select the **CapitalCityResult**, then click the **Add** button. This will add the selected element to the response.

Select the **Wrapper Procedure** tab to see the full Generated Procedure SQL, as shown below.

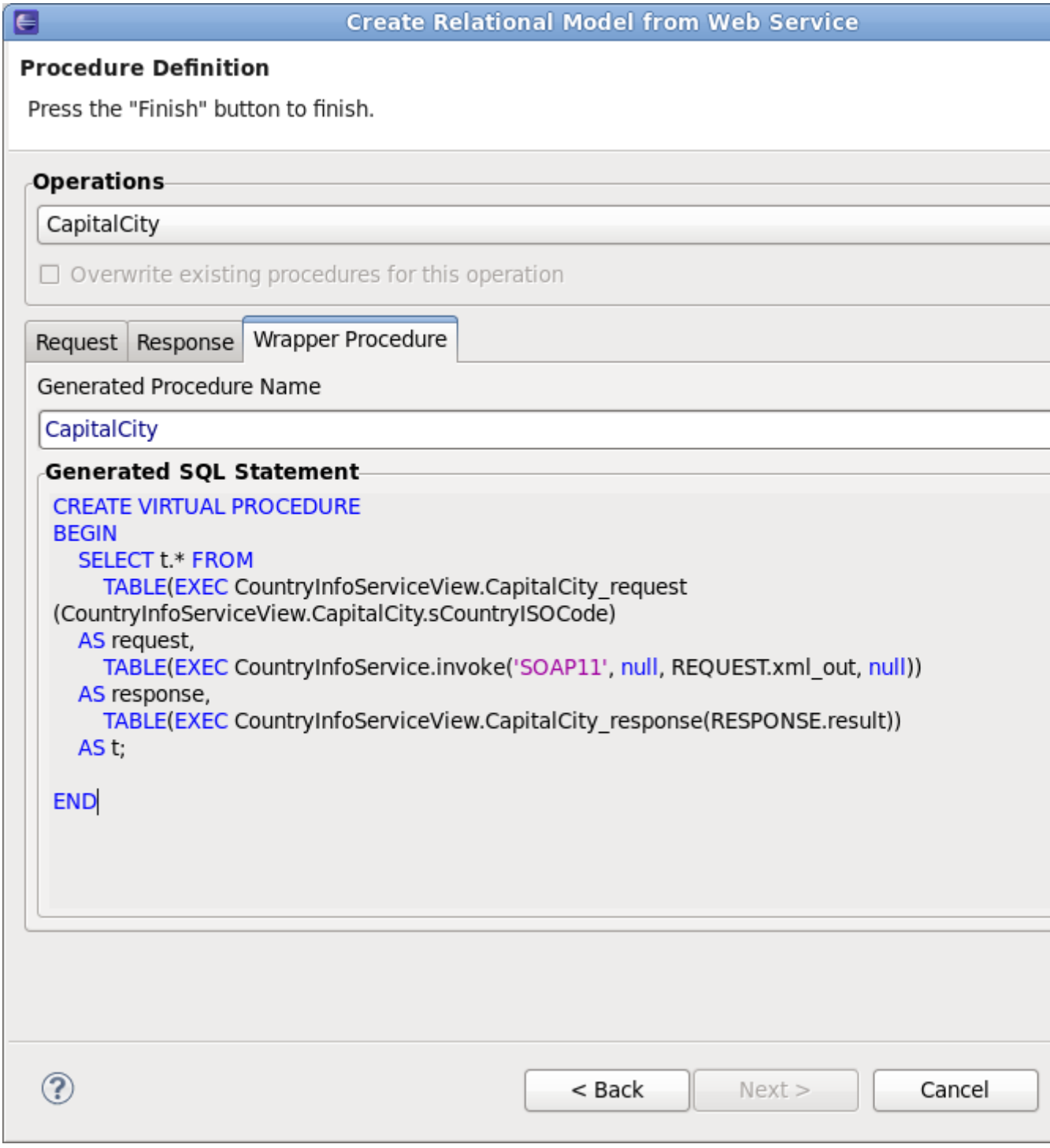


Figure 2.31. Consume SOAP Wizard

Click **Finish** to exit the wizard. In the Cheat Sheet, you can now continue. Click

to advance to the next step.

5. **Create VDB**

This section of the Cheat Sheet provides instructions for creating a VDB using the models that you created in the previous step. The Cheat Sheet section is shown below:

Figure 2.32. Create VDB

Click

next to **Launch New VDB Wizard** to launch the wizard. Follow the steps to create a VDB in your workspace. When complete, exit the wizard. In the Cheat Sheet, you can now continue. Click

to advance to the next step.

6. **Test VDB**

This final section of the Cheat Sheet provides instructions for executing the VDB created in the previous step. Click

next to **Launch Execute VDB Dialog** to launch the wizard. Select the previously-created VDB to execute it.

Server Management

Teiid Designer is a design-time tool that allows setting up and testing deployable VDB artifacts. In order to deploy and test these VDB's a running *JBoss* [http://www.jboss.org] application server is required that contains an installed *Teiid* [https://www.jboss.org/teiid] submodule. This section describes how to set up, connect and maintain your servers and describes the various aspects of what features in Teiid Designer are enabled by this connection and how you can leverage these capabilities.

3.1. Setting up a Server

Teiid is installed as a component of JBoss hence connection to a Teiid Server requires the setting up and configuration of its parent JBoss Server. This is achieved using the Server View, see *Section D.2.3, "Server View"*, displayed as part of the Teiid Designer perspective. The detailed procedures for creating a JBoss Server configuration can be found in the documentation provided at *http://www.jboss.org/tools/docs/reference*. Thus, brief steps are only outlined here.

If no servers have been previously created then the Server View will display a new server hyperlink. To create a new JBoss Server configuration, click the hyperlink.

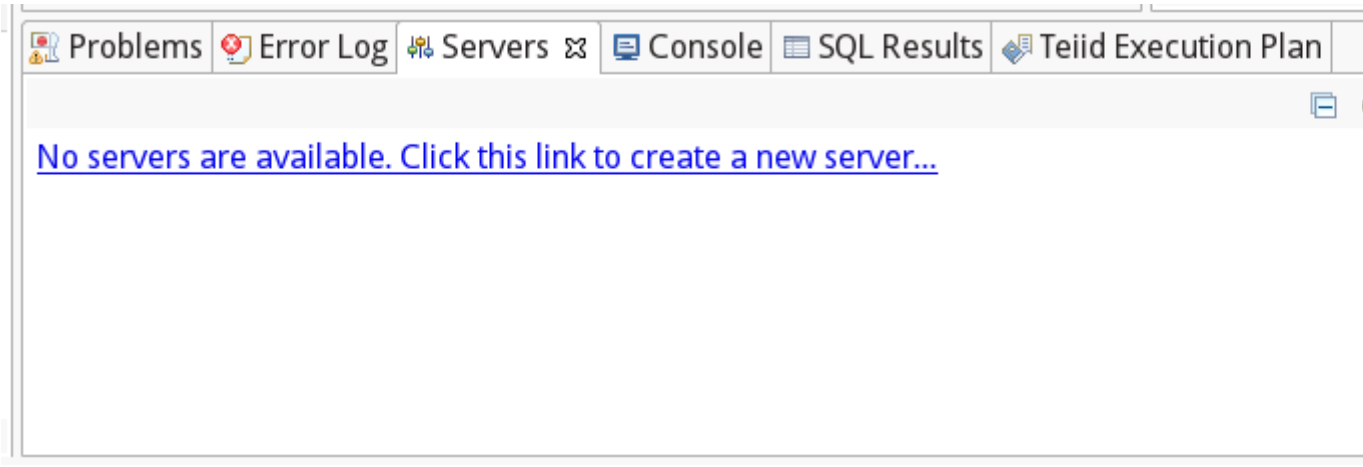


Figure 3.1. Server View with no created servers

Navigate through the wizard, configuring the details of the JBoss Server including its Runtime location, hostname and whether its externally managed. The final property determines whether the server is instantiated within the IDE or whether it is installed and started independently. Should the latter be the case then the Server View merely assumes connection to the independent server.

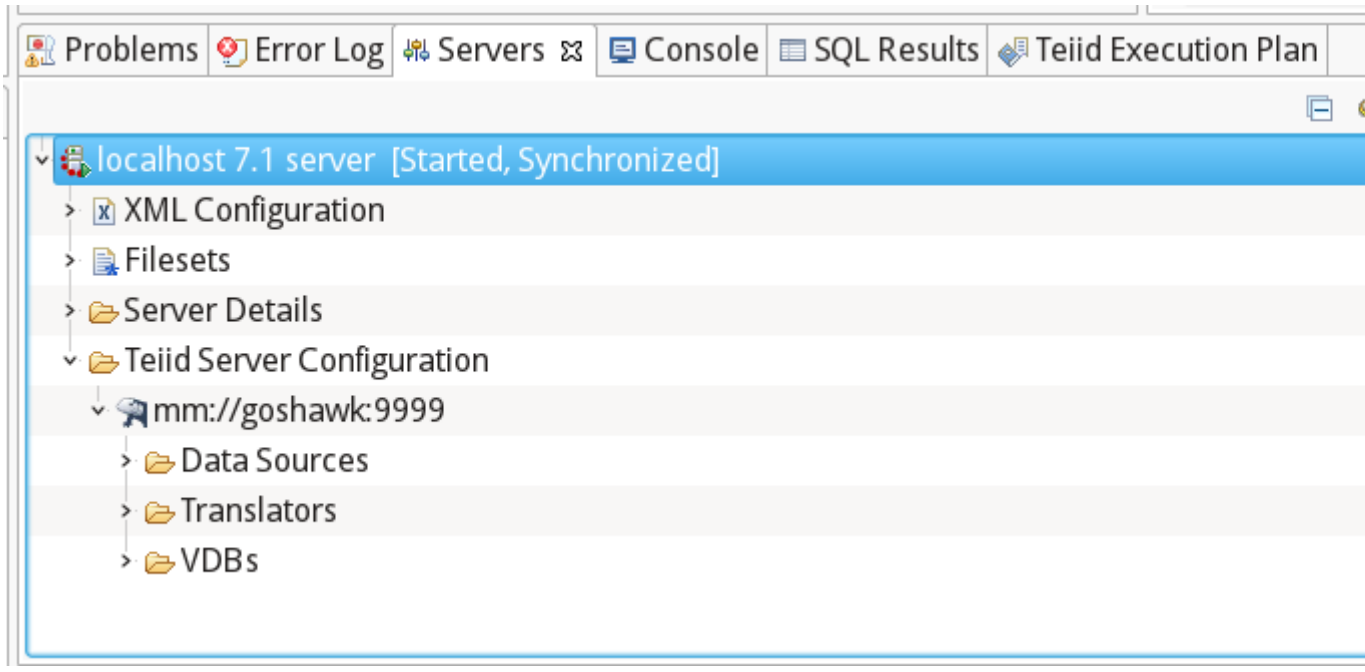


Figure 3.2. Server View with a single server

As illustrated, the server has been installed with a Teiid Server and on clicking the green start button, Designer has successfully connected to the server, resulting in the display of the Teiid Server's configuration.

JBoss Tools provides an editor for the configuration of the JBoss Server. In addition, Designer provides an extra tab to this editor that displays the configuration of the Teiid Server. Only a few options can be modified since most of the configuration is determined by the parent JBoss Server. This editor can be displayed by double-clicking on any node in the JBoss Server tree.

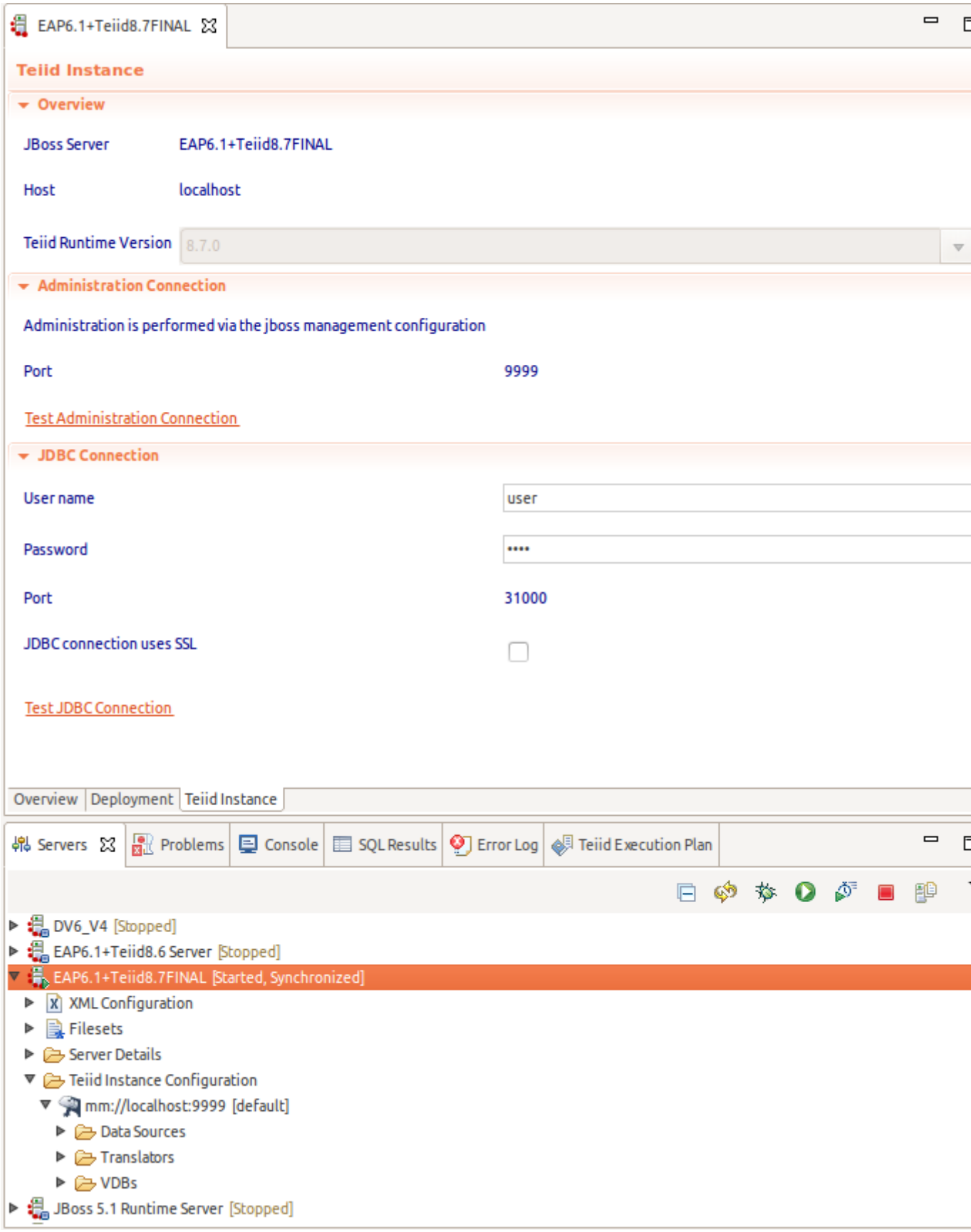


Figure 3.3. Editor for Configuring the Teiid Server

3.1.1. Teiid Version Support

Teiid Designer is bundled with 4 versions of the Teiid Client Runtime, notably:

- **Teiid Server 7.7** version installed on **JBoss Server 5.1**.
- **Teiid Server 8.0**, **8.1, 8.2 and 8.3** versions installed on **JBoss Server 7**.
- **Teiid Server 8.4** thru **8.12** installed on **JBoss Server 7.1.1, 7.2** and **EAP 6.1** or greater.
- **Teiid Server 9.0** installed on **Wildfly 9+**.

This allows multiple client runtime support from within a single installation of Teiid Designer. However, in order to facilitate this support it is necessary for a default server to be chosen by the user. The setting of the default server can be performed from the Teiid Server section of the Guides View or the Servers view, as described in *Section 2.1.2, "Defining a Teiid Server"*. Care should be taken to ensure that any new models are created against the correct version of server to ensure functionality and internal architecture is correct. Note that VDBs are validated against a server version and noted as such in the editor. Note that your VDBs will be validated against the current Teiid runtime client and that version will be persisted in your VDB.

New Model Wizards

Models are the primary resource used by 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 4.1, “Creating New Relational Source Model”
  -
    Section 4.2, “Creating New Relational View Model”
  -
    Section 4.3, “Creating XML Document View Model”
  -
    Section 4.4, “Creating XML Schema Model”
  -
    Section 4.5, “Creating Web Service View 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 *Section D.2.1, “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 prevented from entering an existing name.

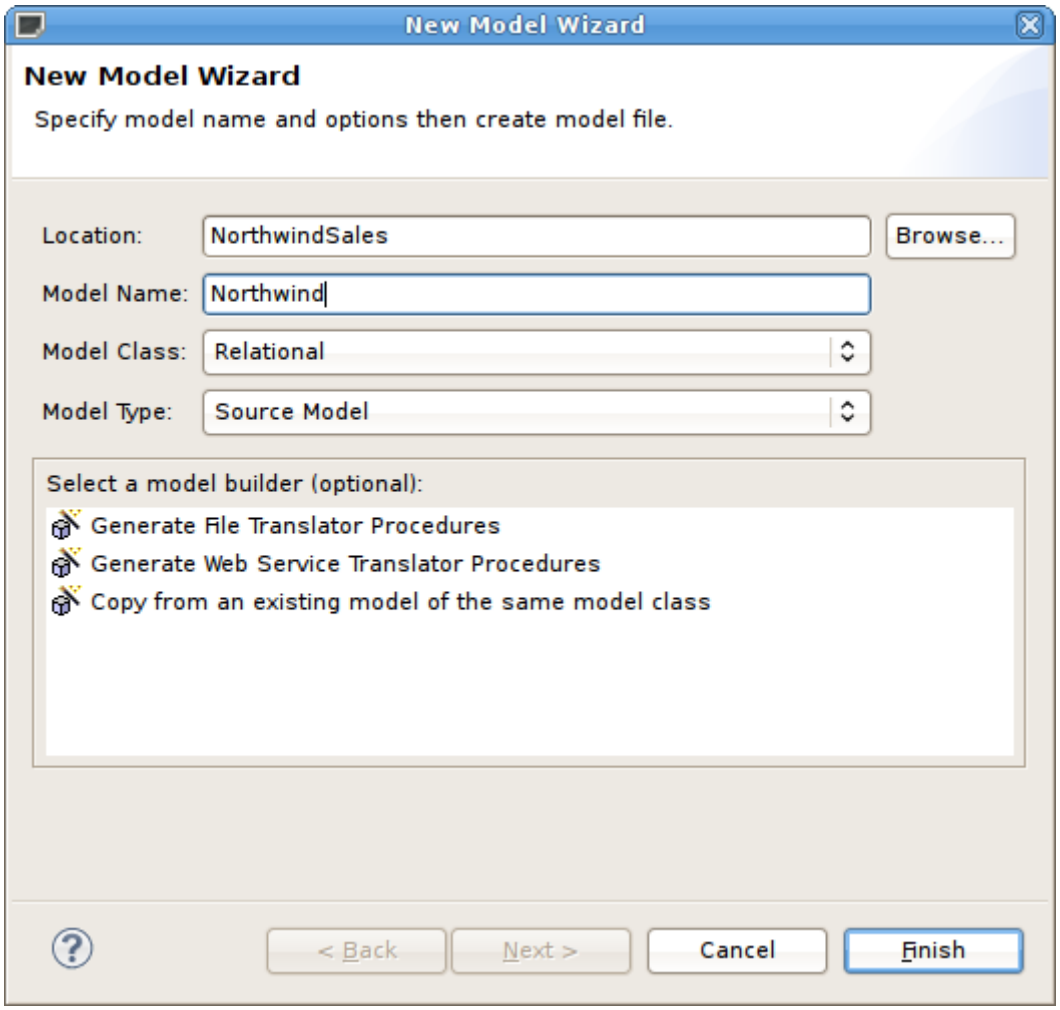


Figure 4.1. Import Wizard Selection Dialog

4.1. Creating New Relational Source Model

Create New Relational Source Model

- To create a new empty relational source model:
  - **Step 1** - Launch the *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.

4.1.1. 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 ore more of the **Available File Translator Procedures**, then Click **Finish**

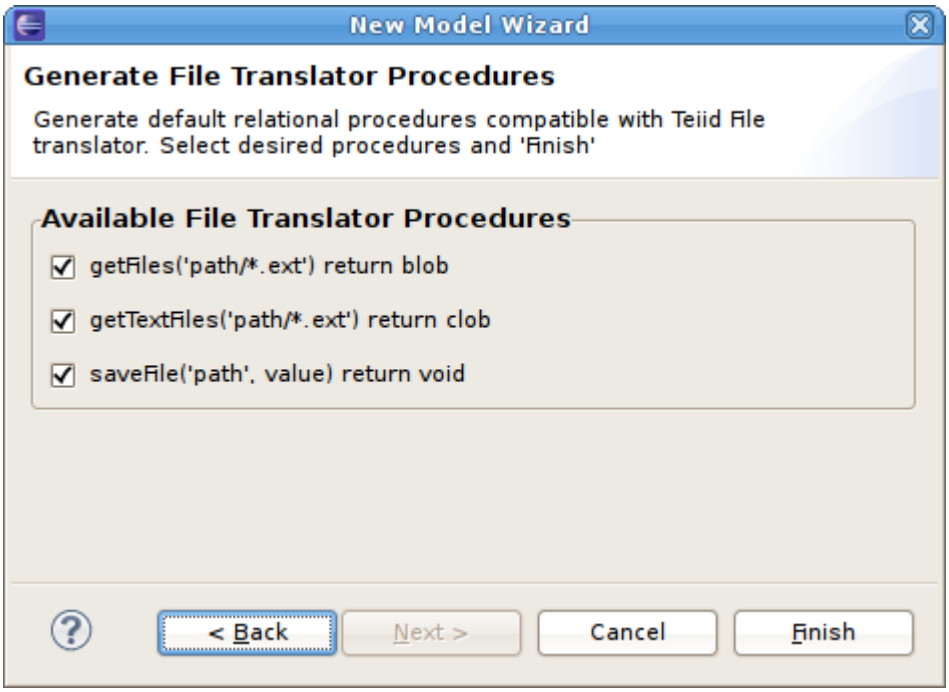


Figure 4.2. Generate File Translator Procedures Dialog

4.1.2. 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 web-service-based 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 Web Service Translator Procedures** dialog will be displayed.
  - **Step 6** - Check one ore more of the **Available Web Services Translator Procedures**, then Click **Finish**

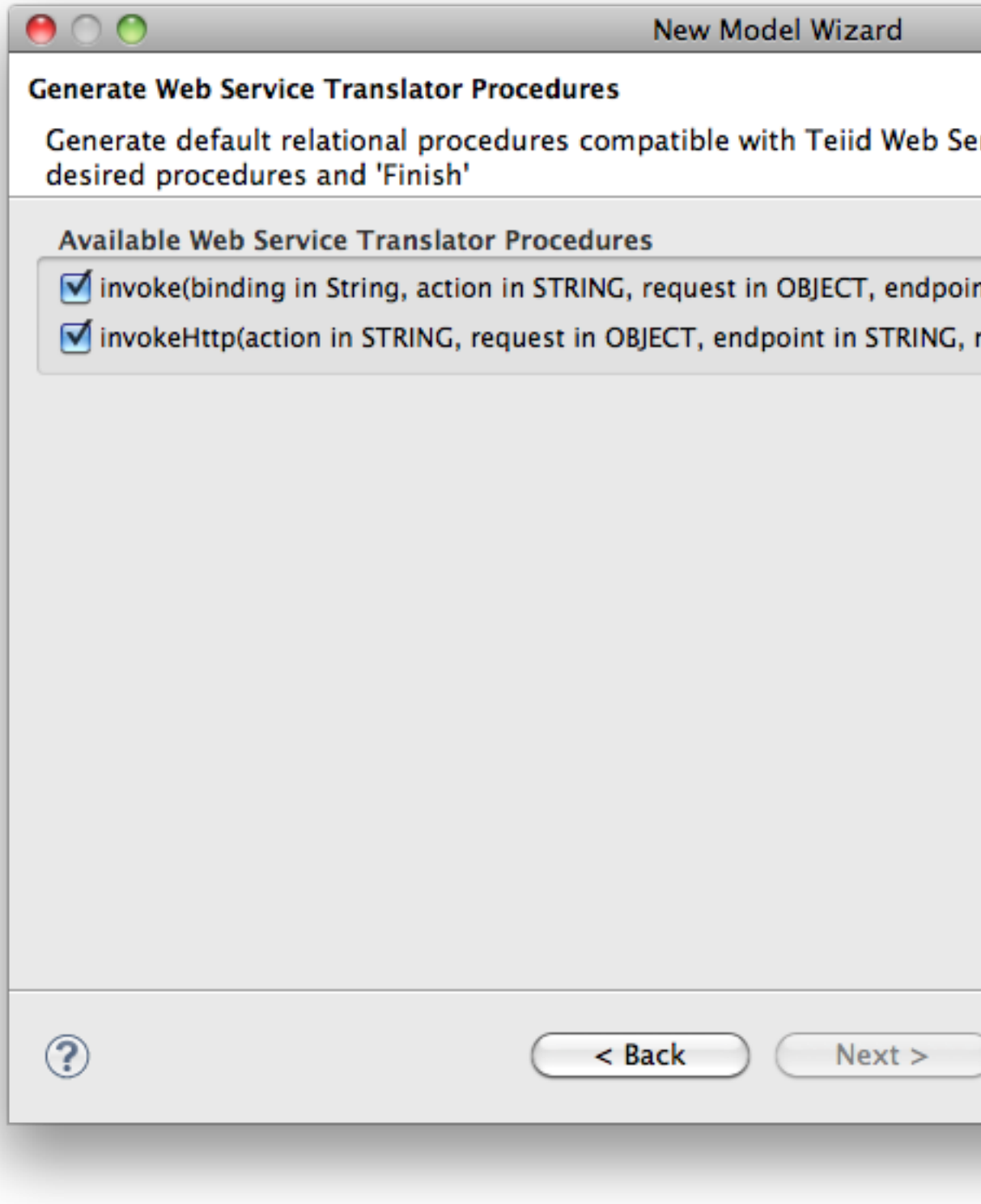


Figure 4.3. Generate Web Service Translator Procedures Dialog

4.1.3. 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 *Section D.2.1, "Model Explorer View"* prior to starting the new model wizard.

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

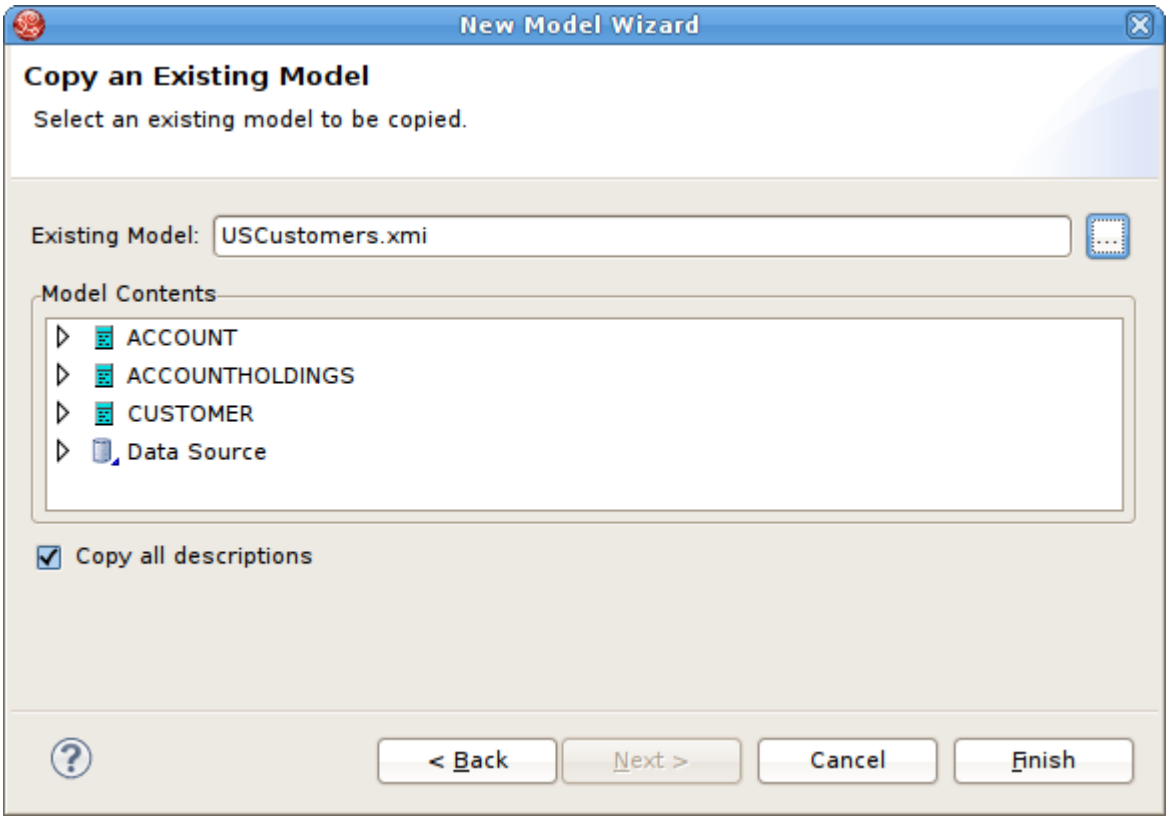


Figure 4.4. Copy An Existing Model Dialog

4.2. Creating New Relational View Model

Create New Relational View Model

- To create a new empty relational view model:
  - **Step 1** - Launch the *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.

4.2.1. 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**

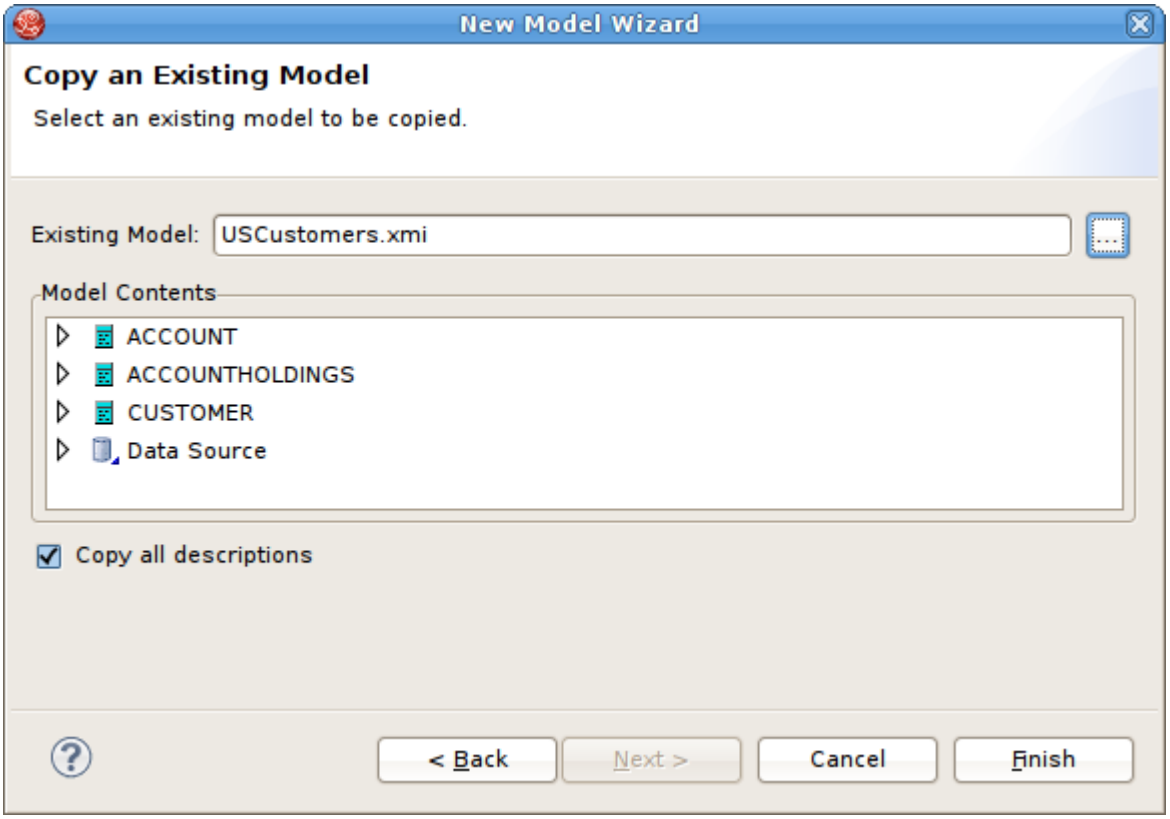


Figure 4.5. Copy An Existing Model Dialog

4.2.2. 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 4.2.1, "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.

4.3. Creating XML Document View Model

Create XML Document View Model

- To create a new empty XML document view model:
  - **Step 1** - Launch the *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.

4.3.1. 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**

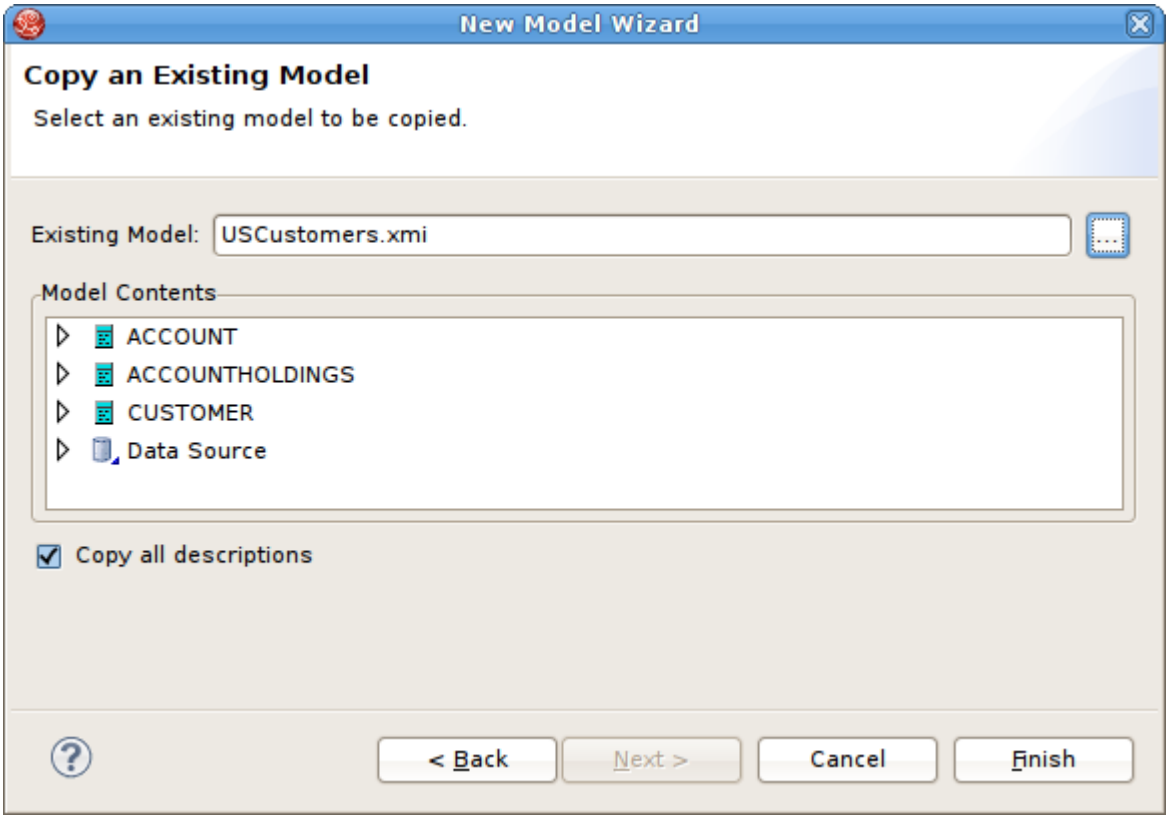


Figure 4.6. Copy An Existing Model Dialog

4.3.2. 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.

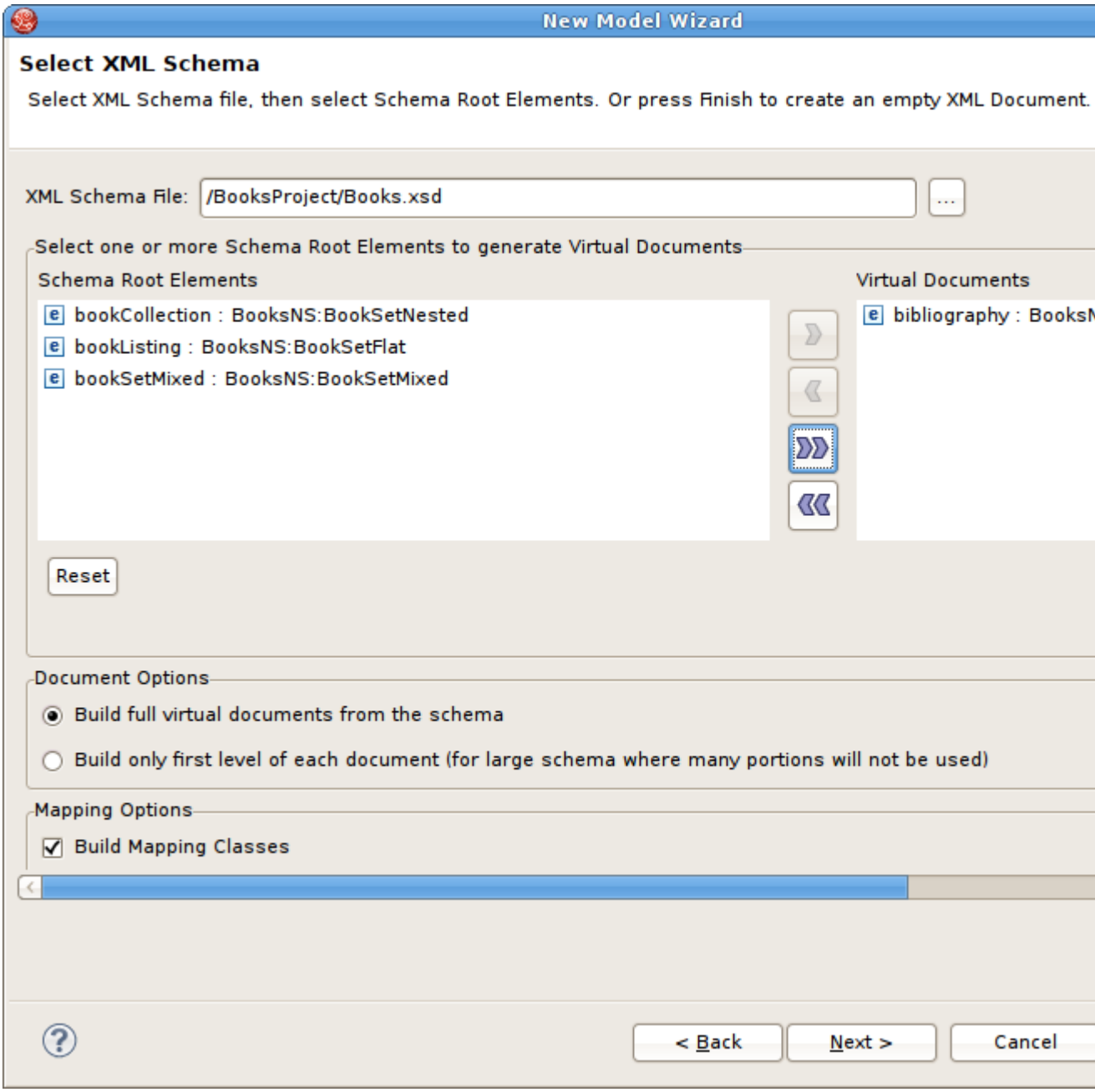


Figure 4.7. Select XML Schema Dialog

- **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.

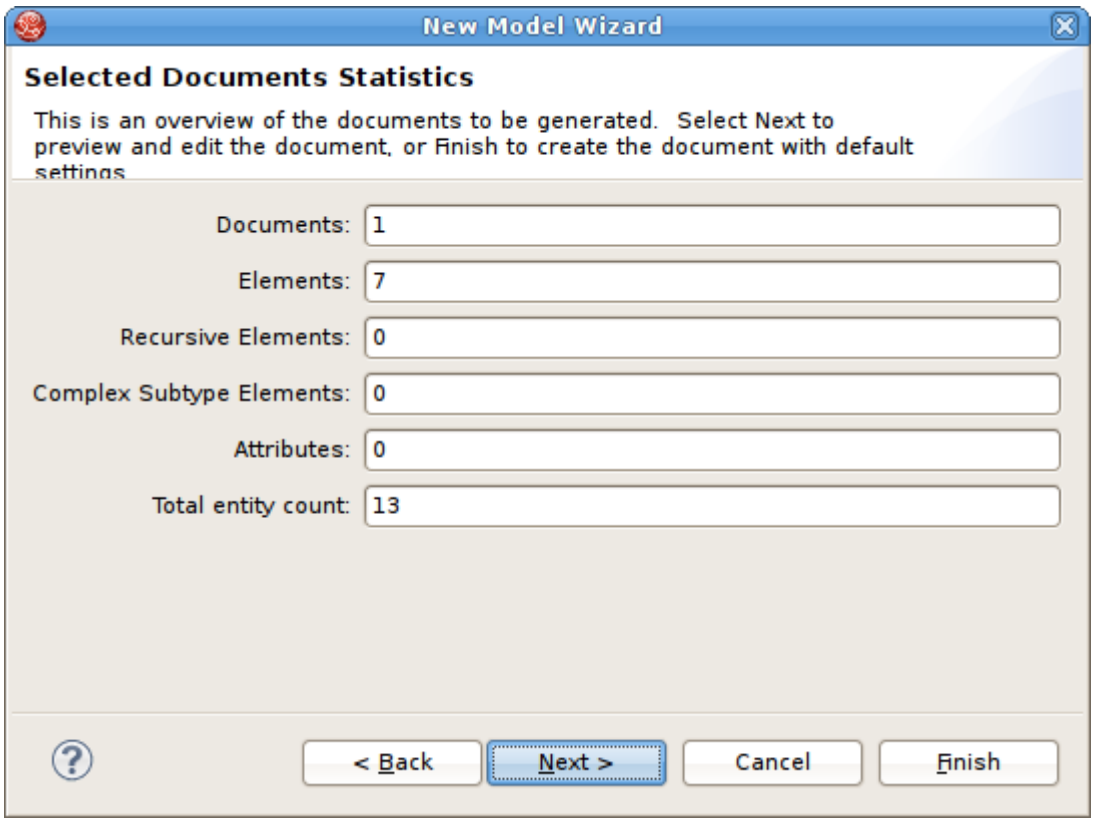


Figure 4.8. Selected Documents Statistics Dialog

- **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.

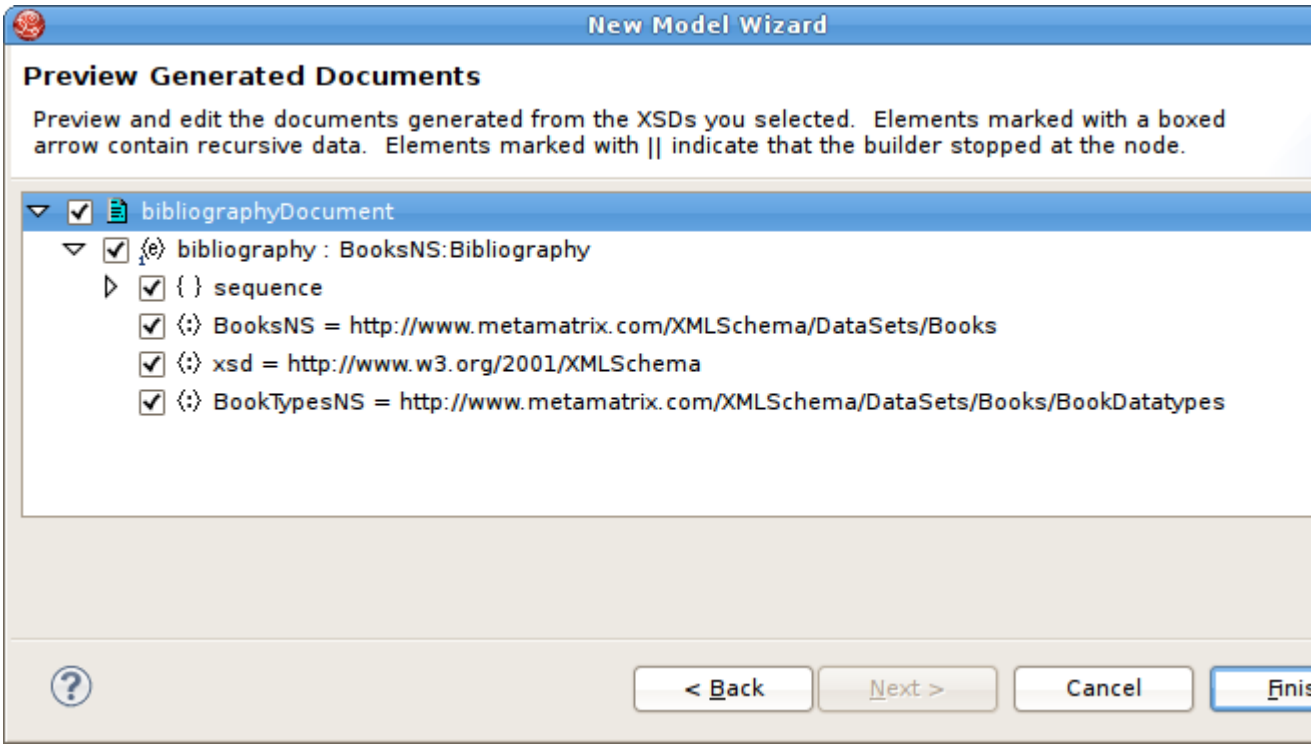


Figure 4.9. Preview Generated Documents Dialog

4.4. Creating XML Schema Model

Create XML Schema Model

- To create a new empty XML schema (.xsd) model:
  - **Step 1** - Launch the *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.

4.4.1. 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**

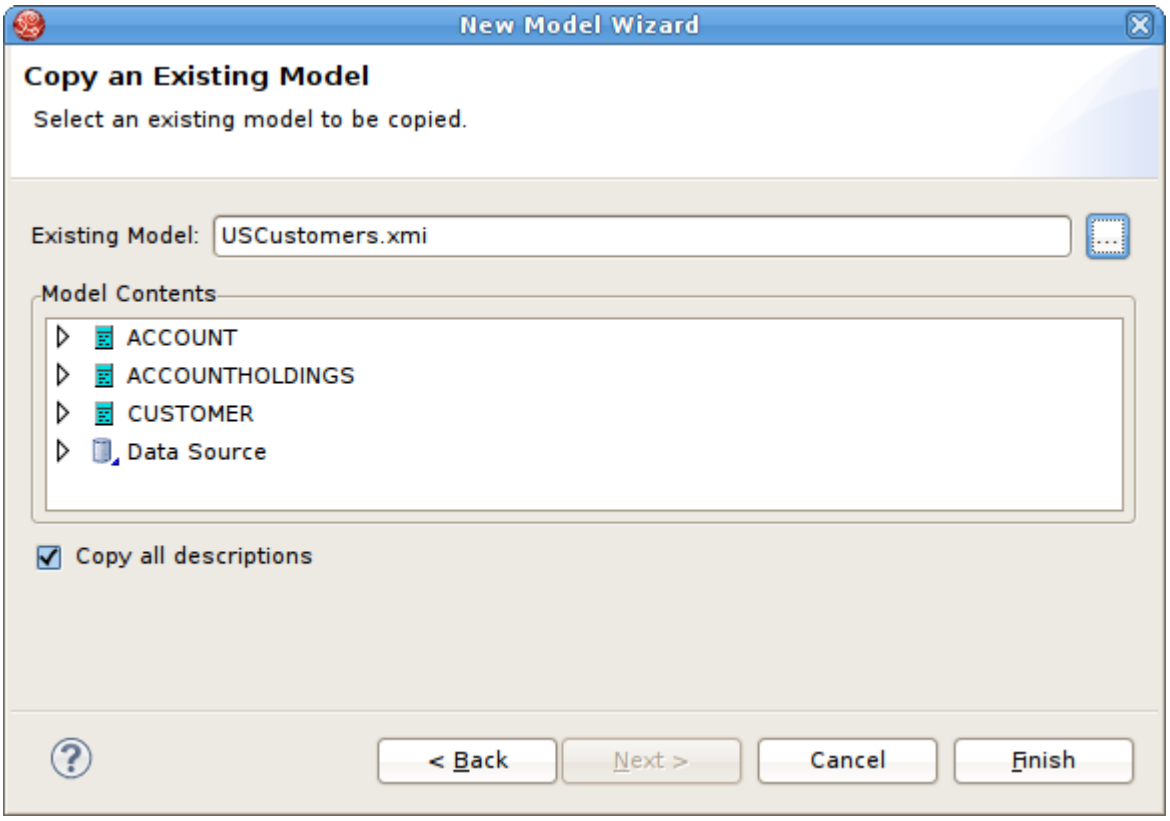


Figure 4.10. Copy An Existing Model Dialog

4.5. Creating Web Service View Model

Create Web Service View Model

- To create a new empty web service view model:
  - **Step 1** - Launch the *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.

4.5.1. 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**

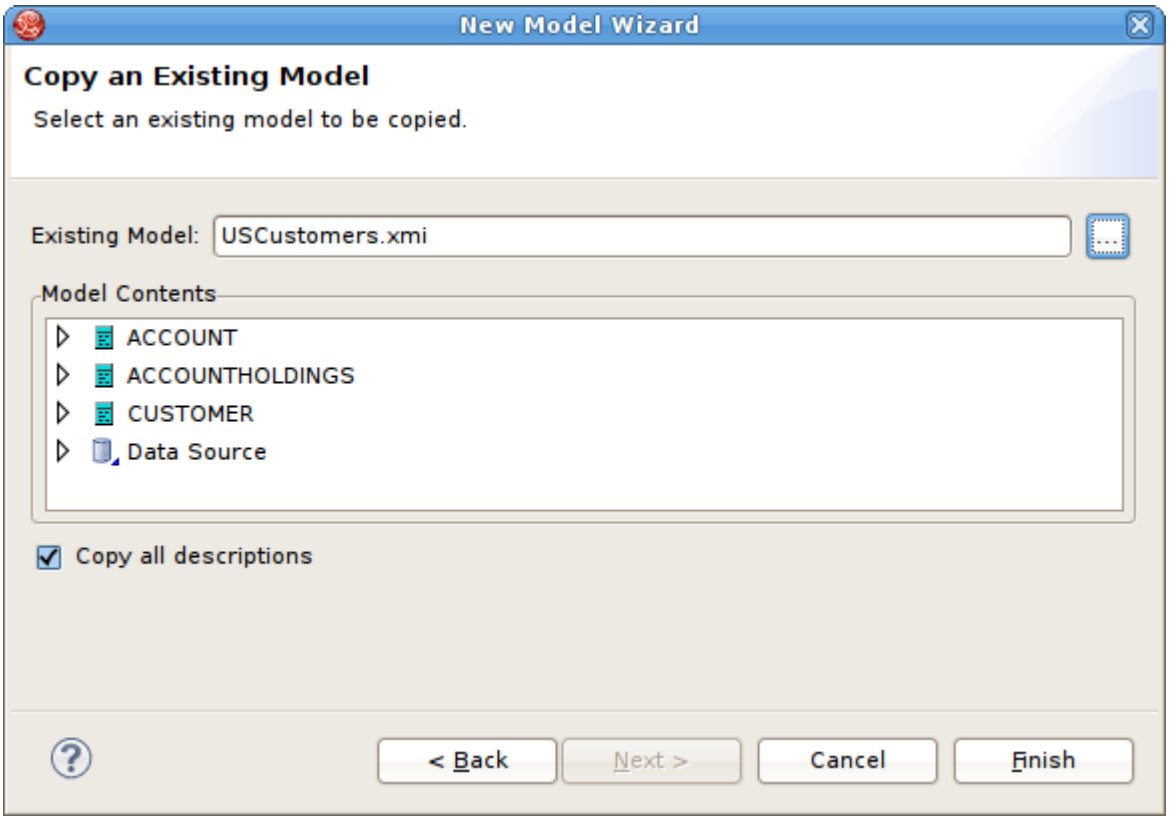


Figure 4.11. Copy An Existing Model Dialog

4.5.2. 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 5.10, "Import WSDL Into Web Service"* action option.

4.5.3. Build From Relational Models

See *Section 7.4.1, "Create Web Service Action"*

4.5.4. 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 *Section D.2.1, "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.

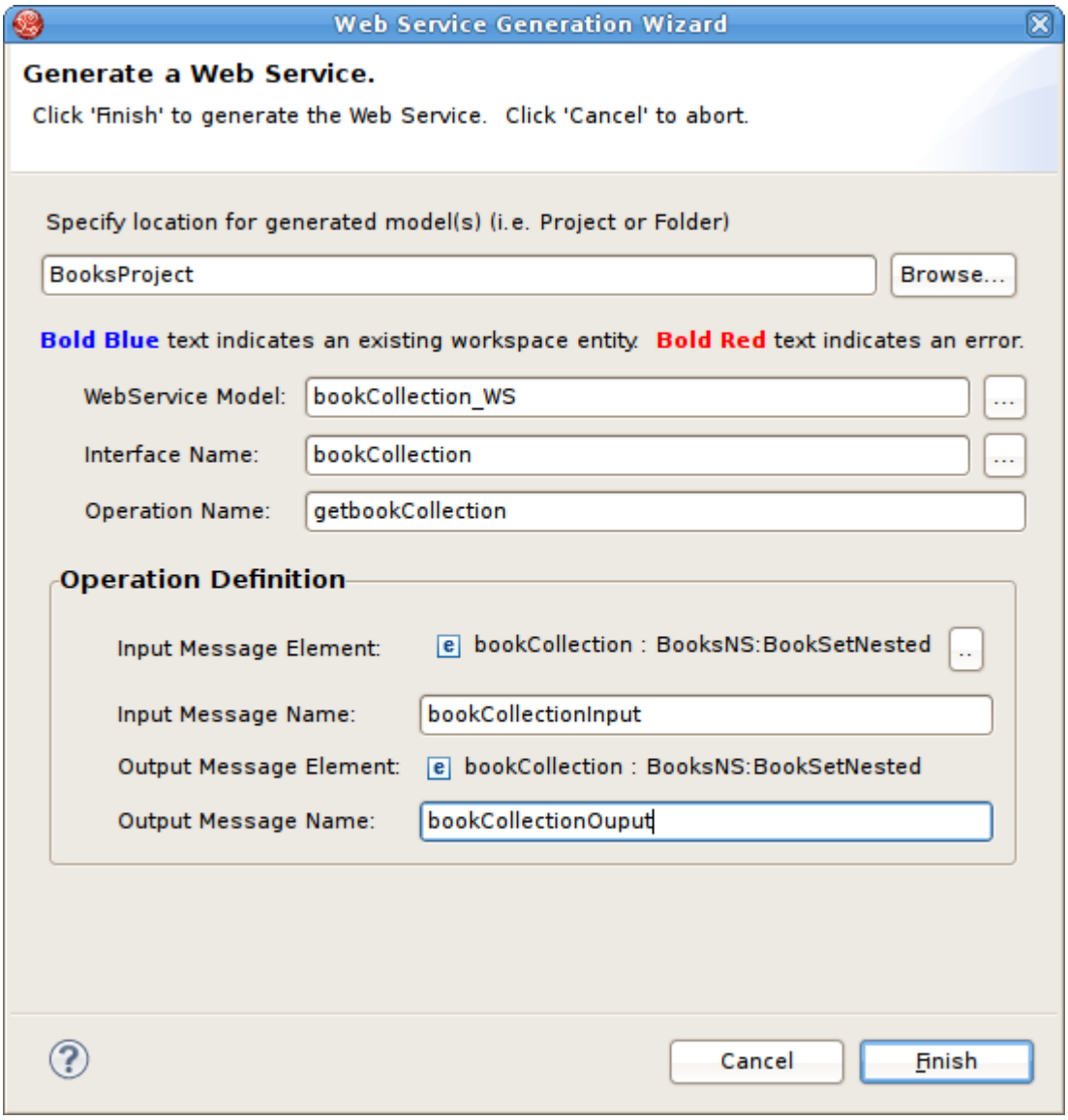


Figure 4.12. Generate A Web Service Dialog

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

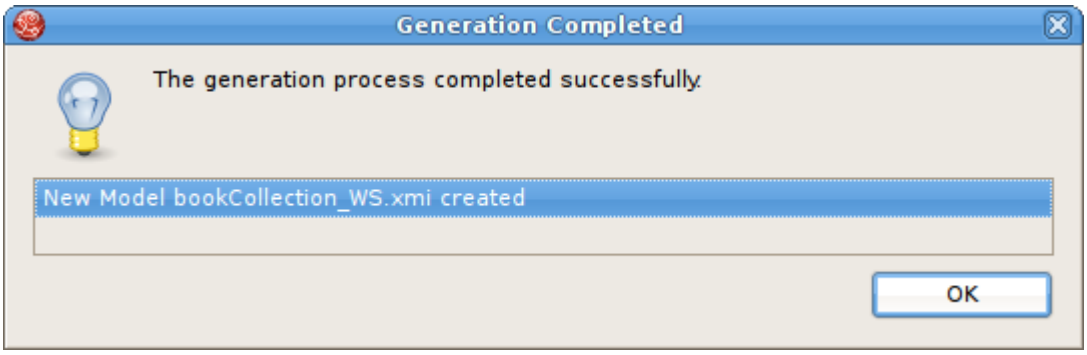


Figure 4.13. Generation Completed Dialog

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.

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...**"

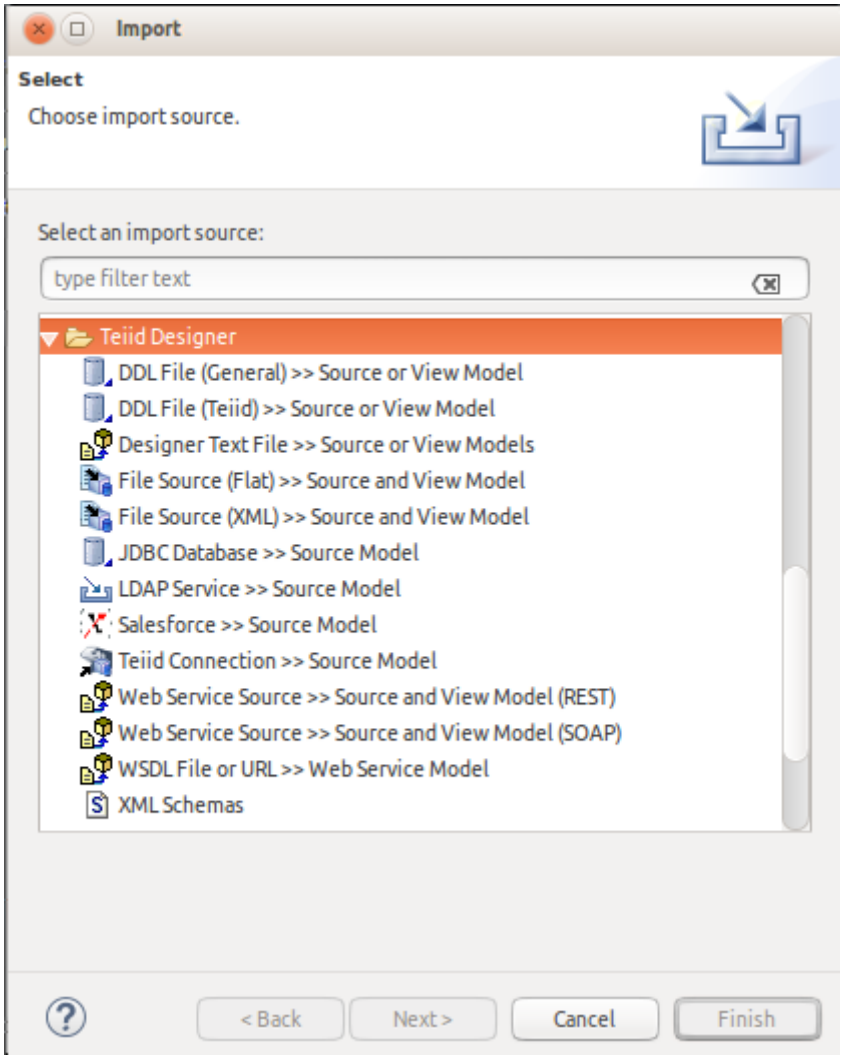


Figure 5.1. Import Wizard

5.1. Import DDL

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 File >> Source or View Model** and click **Next>**
- **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)

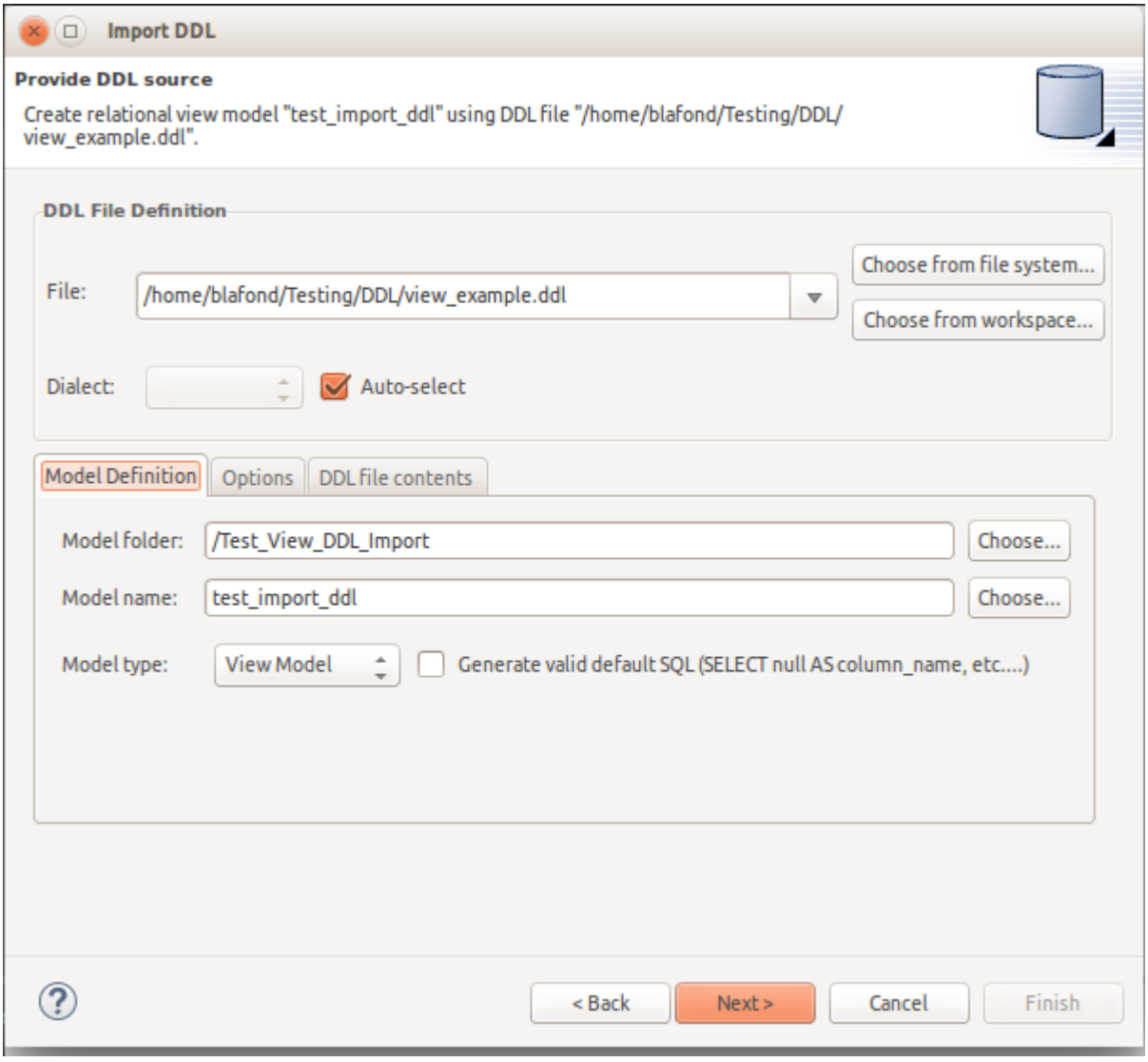


Figure 5.2. DDL Import Options

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

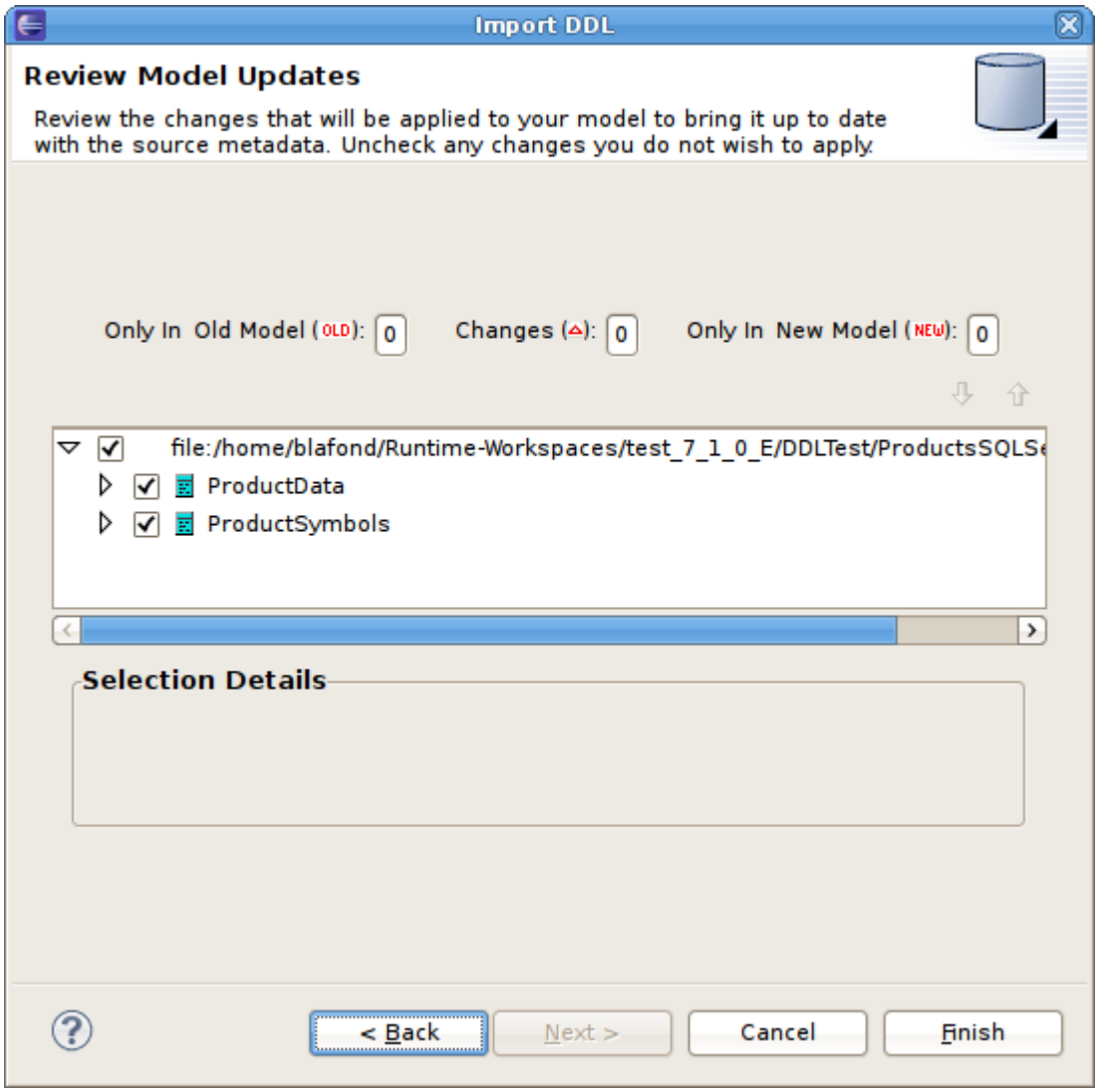


Figure 5.3. Review DDL Updates Dialog

5.2. Import From Relational Database

- You can create source models from your relational 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 **Teiid Designer > JDBC Database >> Source Model** and click **Next>**
- **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.

Note

the Connection Profile selection list will be populated with only JDBC Database connections.

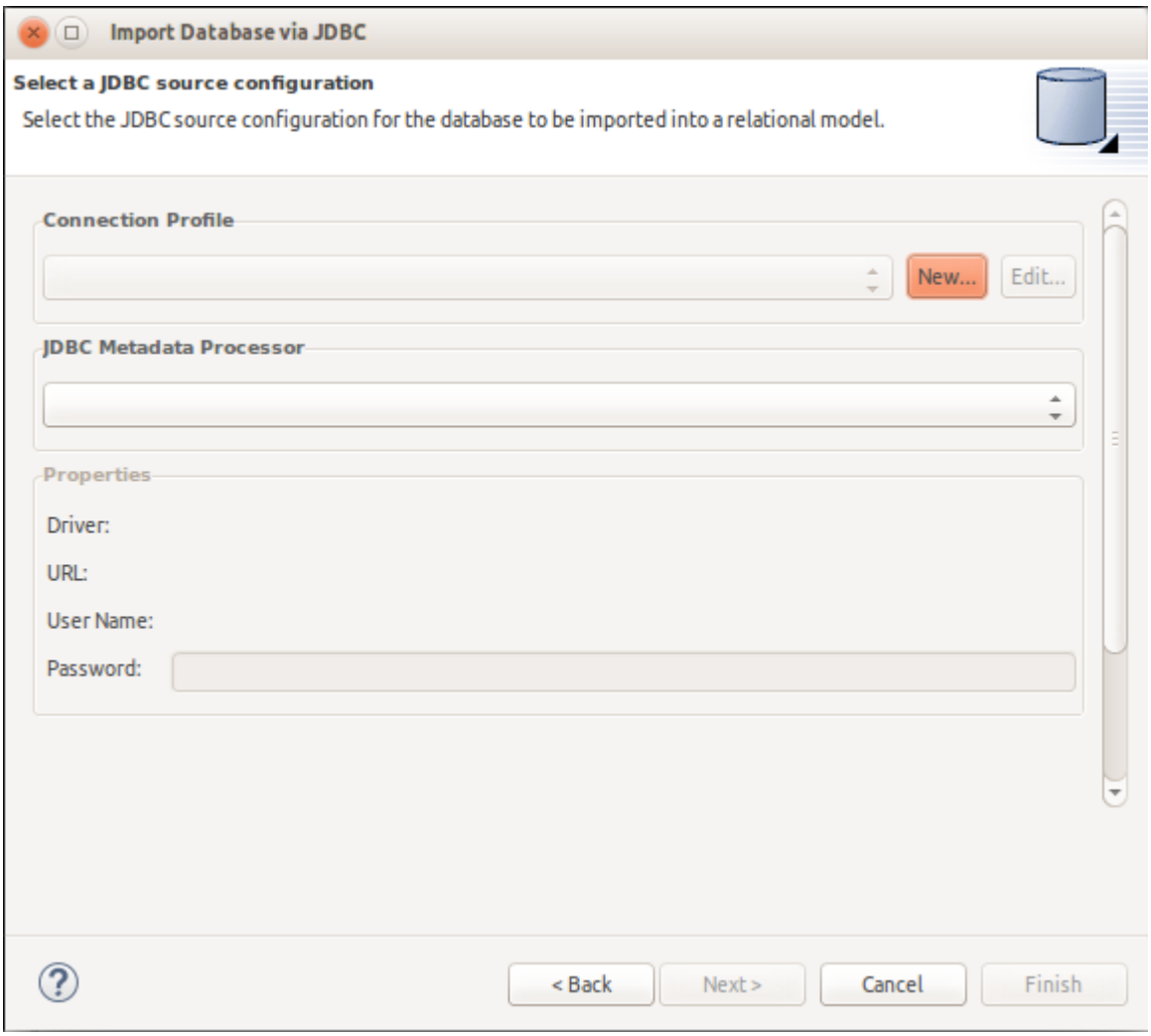


Figure 5.4. Select JDBC Source Configuration Dialog

Because relational databases are different, special processing of your metadata to be required in order to convert datatypes or to interpret your metadata. The **JDBC Metadata**

Processor drop-down selector will be auto-selected based on your selected connection profile. Special processors are available for DB2, Modeshape, ODBC, Oracle, PostgeSQL, SQL Server and Sybase. For all other DB's ajdefault JDBC processor is available.

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

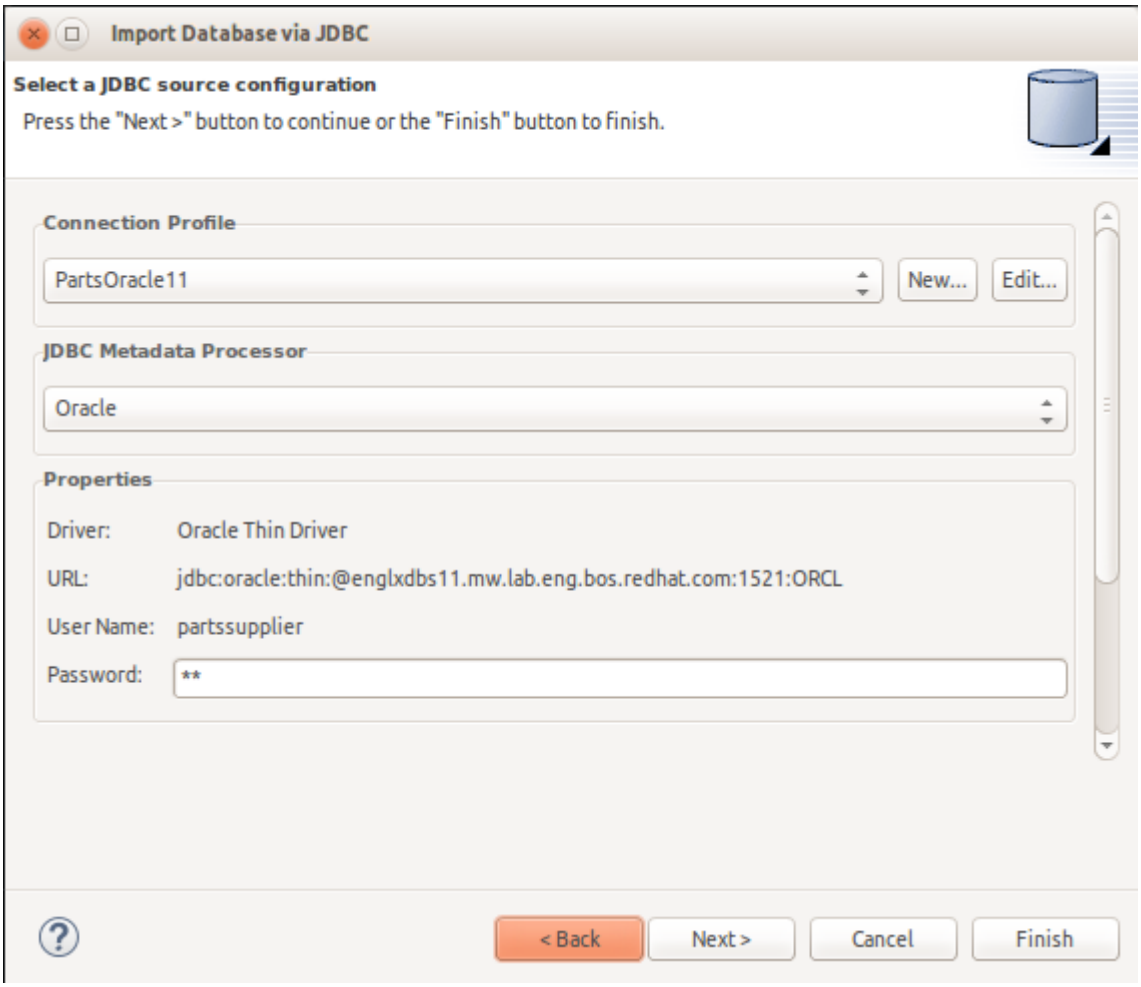


Figure 5.5. Select JDBC Source Configuration Dialog

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

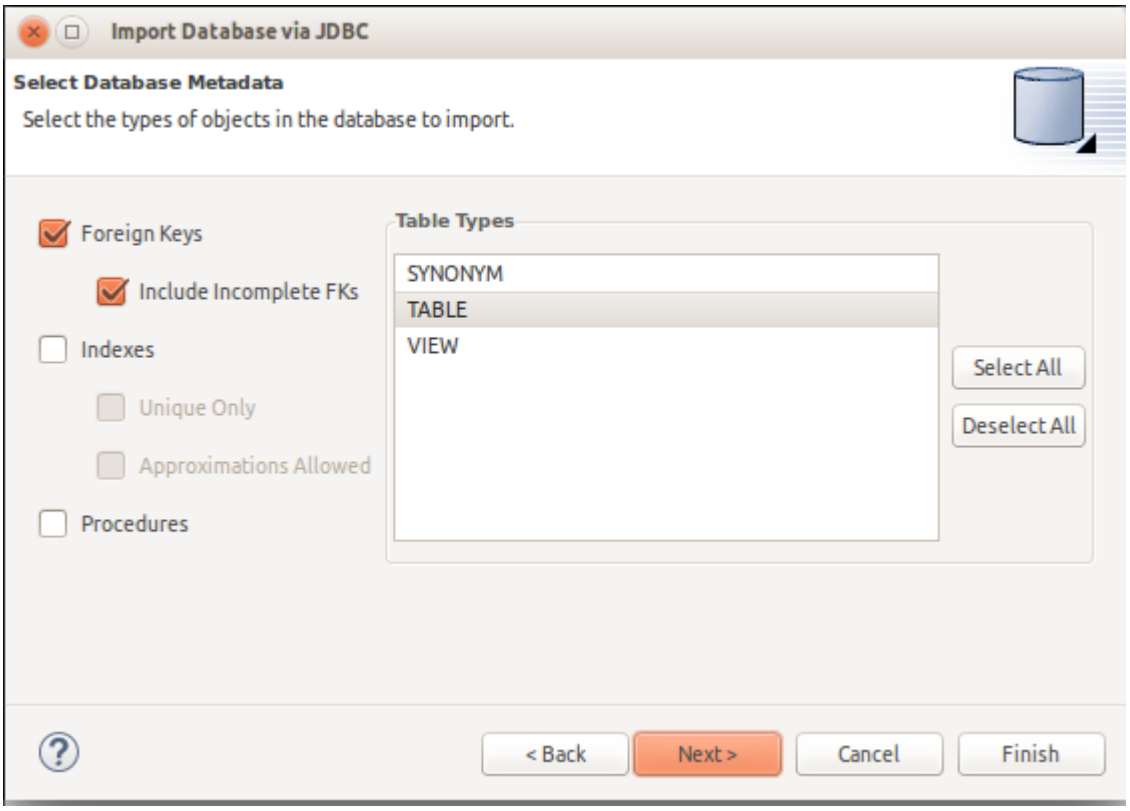


Figure 5.6. Select Database Metadata Dialog

- **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)

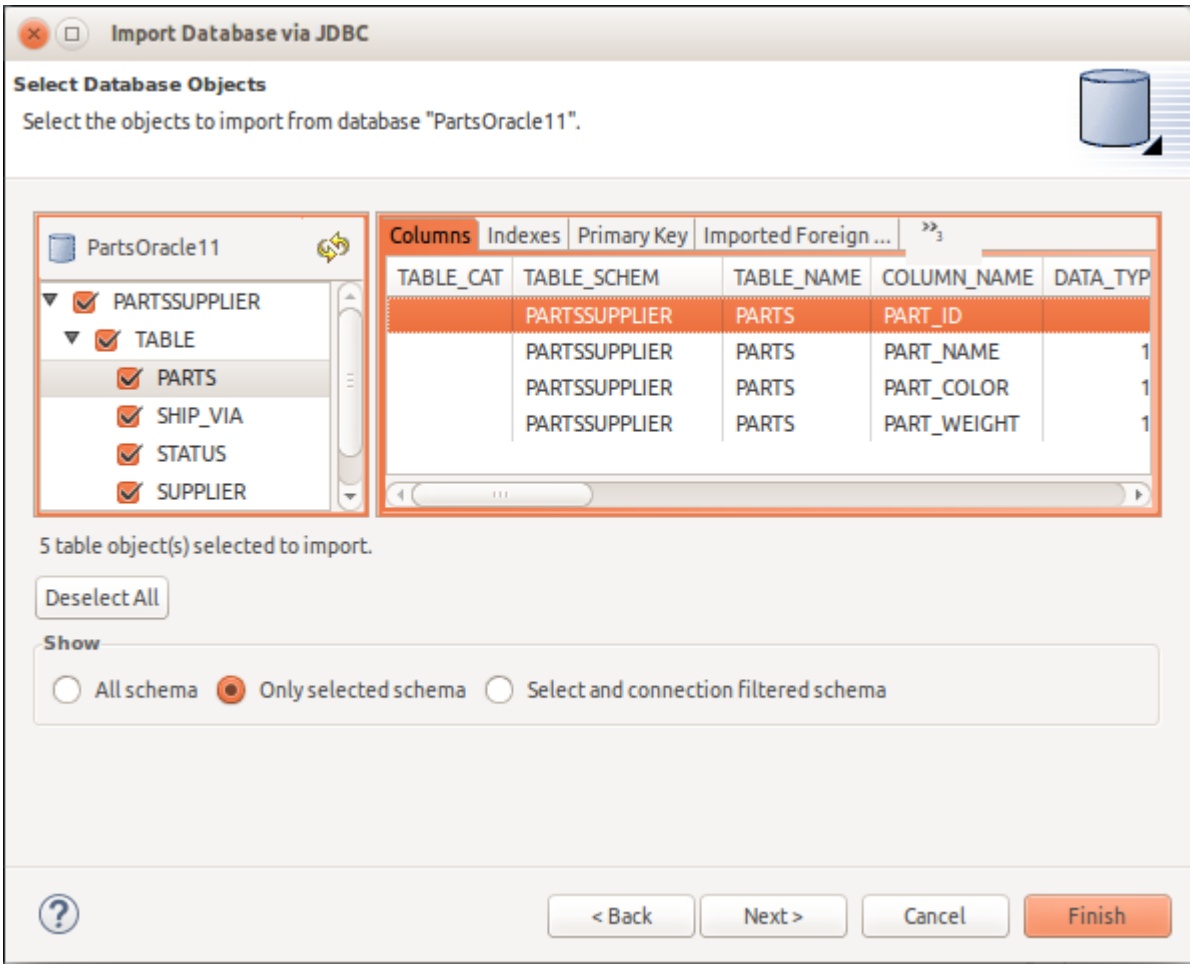


Figure 5.7. Select Database Options Dialog

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

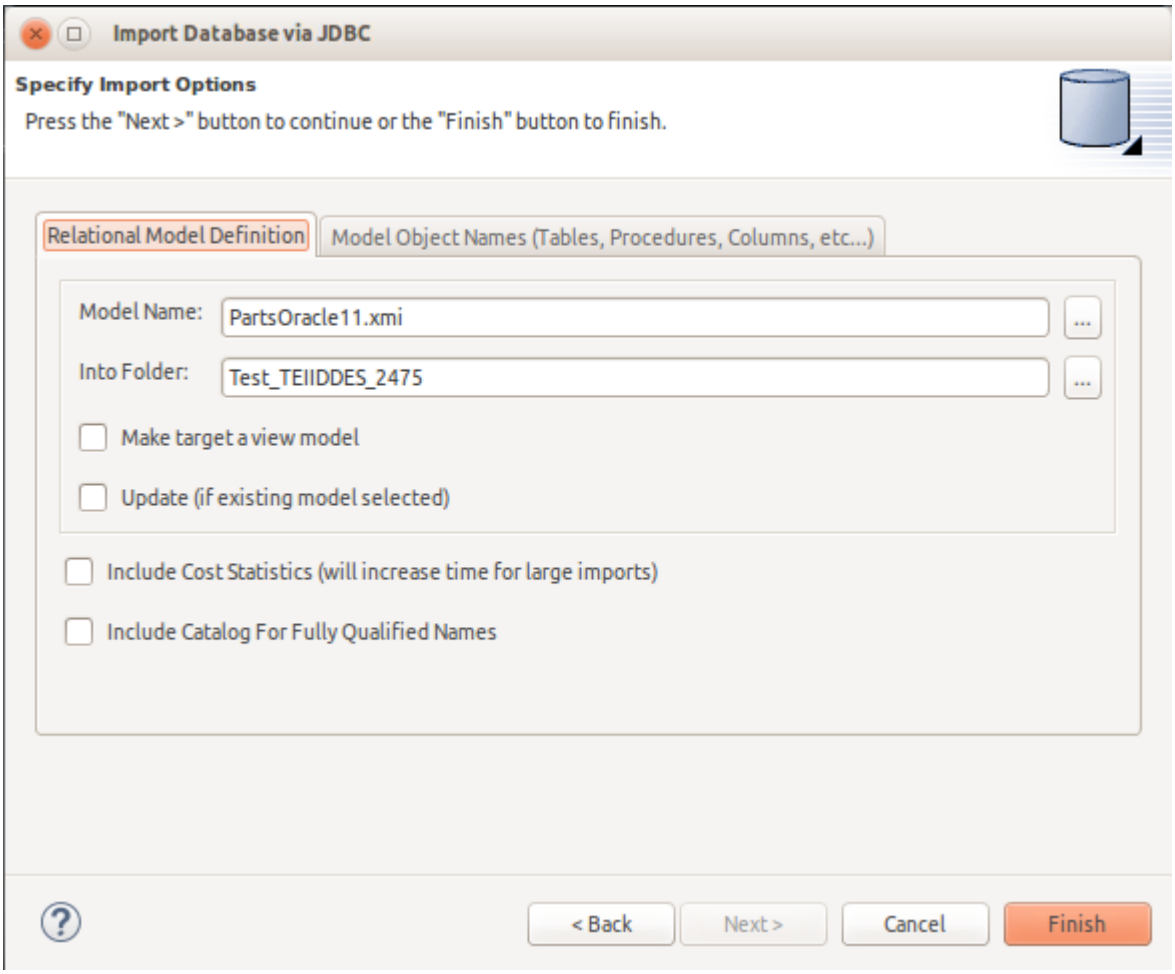


Figure 5.8. Specify Import Options Dialog - Relational Model Definition

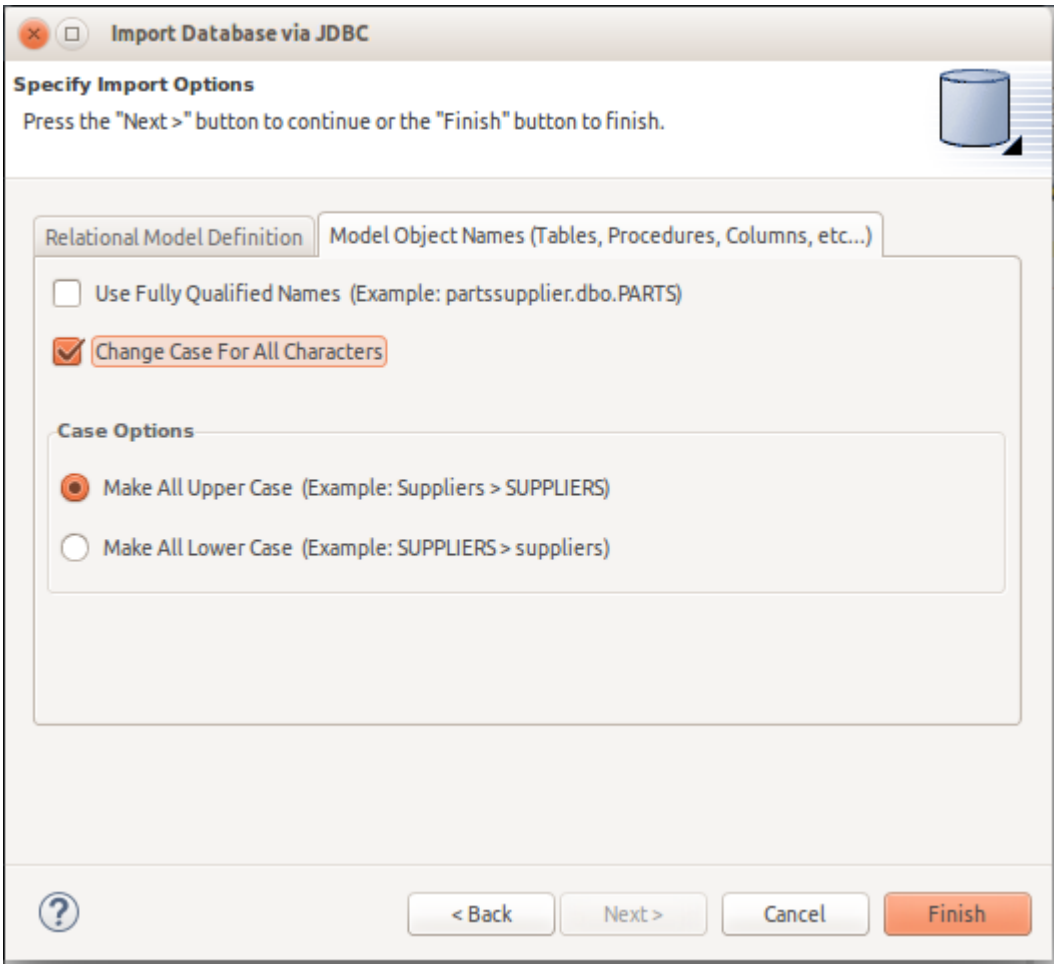


Figure 5.9. Specify Import Options Dialog - Object Naming Options

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

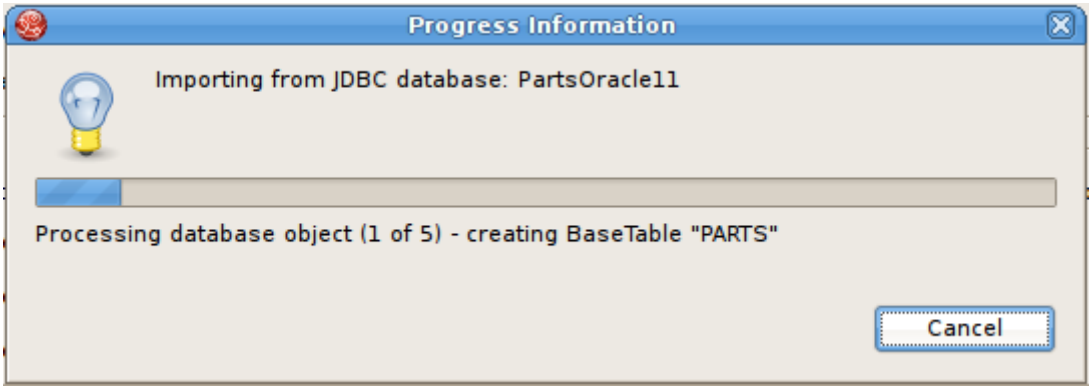


Figure 5.10. JDBC Import Progress Dialog

5.2.1. Relational Model Costing

On the final page of import, you are presented with a checkbox option to **Include Cost Statistics**. If this option is selected, the table cardinalities will be determined and the table cardinality properties will be set. Later, during query execution, if the table cardinalities are available they are used by the Teiid query engine to optimize the query plan.

After import, you can update the cardinalities at any time. If you would like to update the Cost Statistics for the entire model, select the model - then use the Modeling Context menu action **Update Source Data Statistics**. Likewise, you can update the cardinality for a specific table or tables, by selecting the table(s) in ModelExplorer then using the same **Update Source Data Statistics** action.

5.3. Import From Teiid Data Source Connection

The **Teiid Connection >> Source Model** import option provides a means to create relational source models from relational and other deployed data sources that are not supported by other Teiid Designer importers.

NOTE: To launch this importer, you must have at minimum a Teiid 8.x server running in Designer. The Teiid importer deploys a dynamic VDB to Teiid containing the selected source type, then the schema (as determined by Teiid) is retrieved. We expect to move towards this type of import in future versions of Teiid Designer.

- You can create relational source models from your deployed data source connections 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 > Teiid Connection >> Source Model** and click **Next>**
  - **Step 3** - Select the datasource to use for the import. You can create a new source if it doesnt exist, as well as other source management functions. Click **NEXT>** to continue.

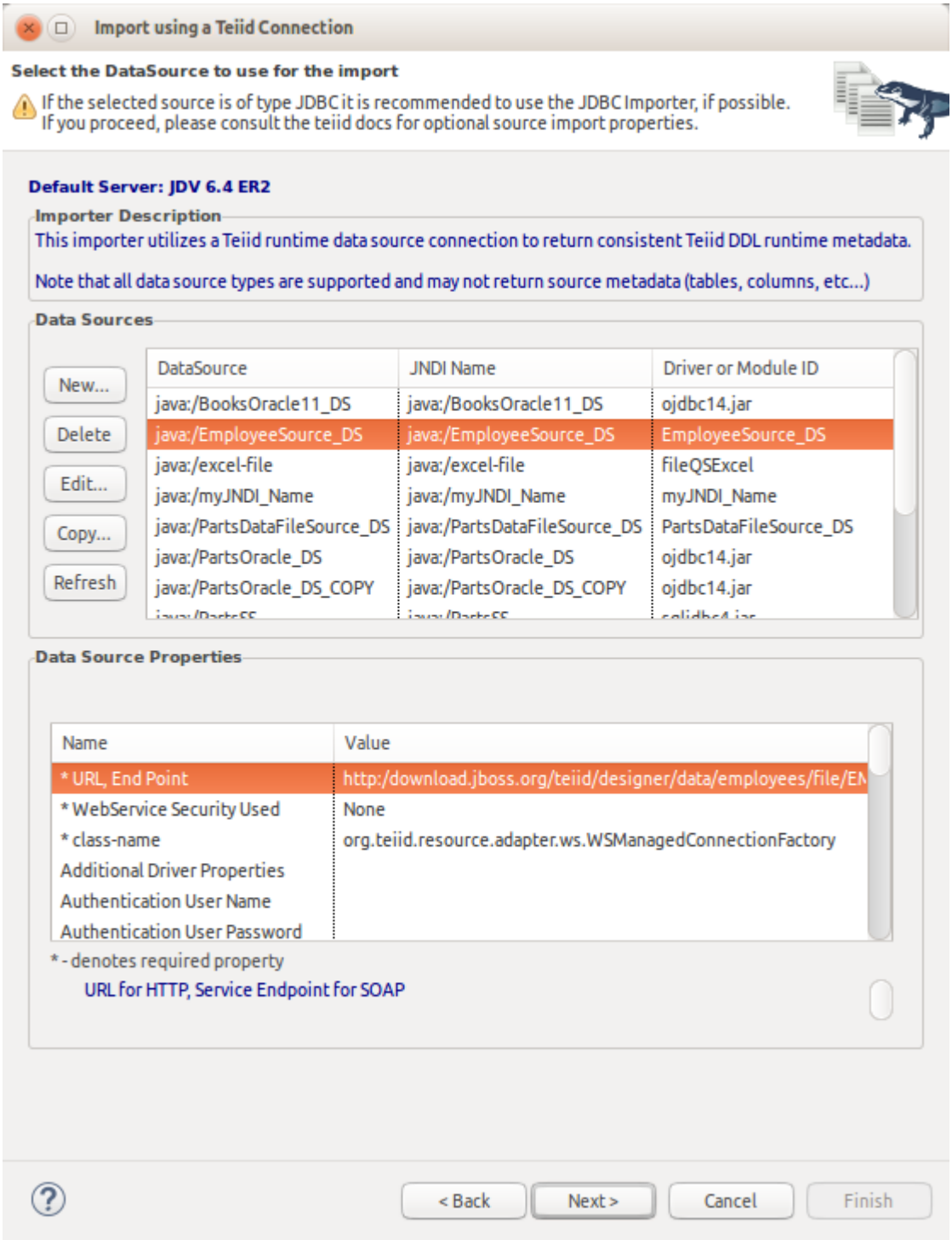


Figure 5.11. Select Deployed Data Source

- **Step 4** - On the next page select the appropriate translator for your data source type as well as defined the target relational model that you wish to create or update. You can also specify value for any built-in import properties in order to fine-tune the schema data you wish to have returned. Click **NEXT>** to continue.

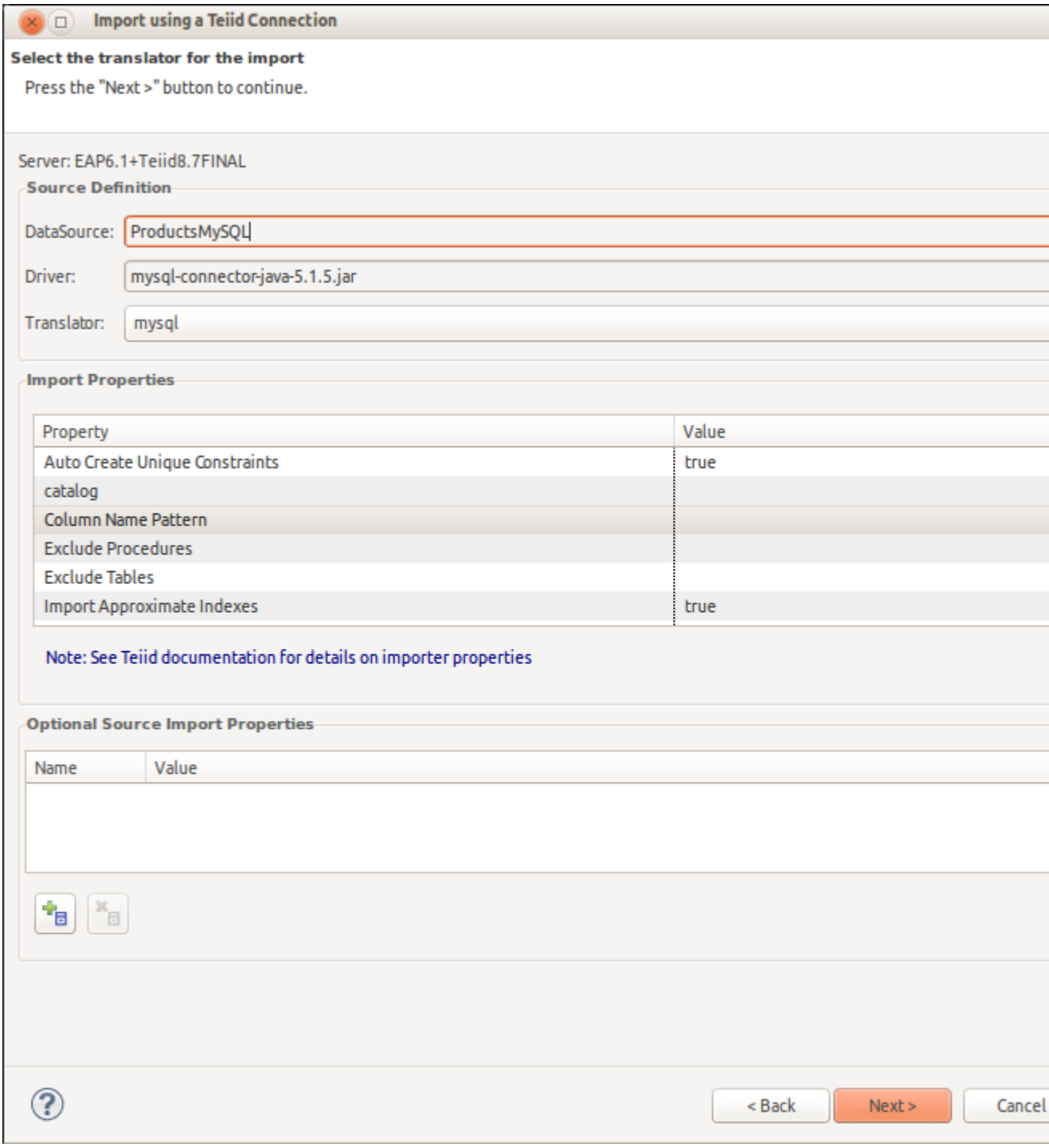


Figure 5.12. Translator and Model Definition

- **Step 5** - On the next page define your target relational model that you wish to create or update. Either enter a new unique name, or select an existing source model. At this point you can click on the *Advanced* tab to view or edit the temporary vdb.xml contents that will be deployed to Teiid in order to return the requested database schema/DDL. Click **NEXT>** to continue.

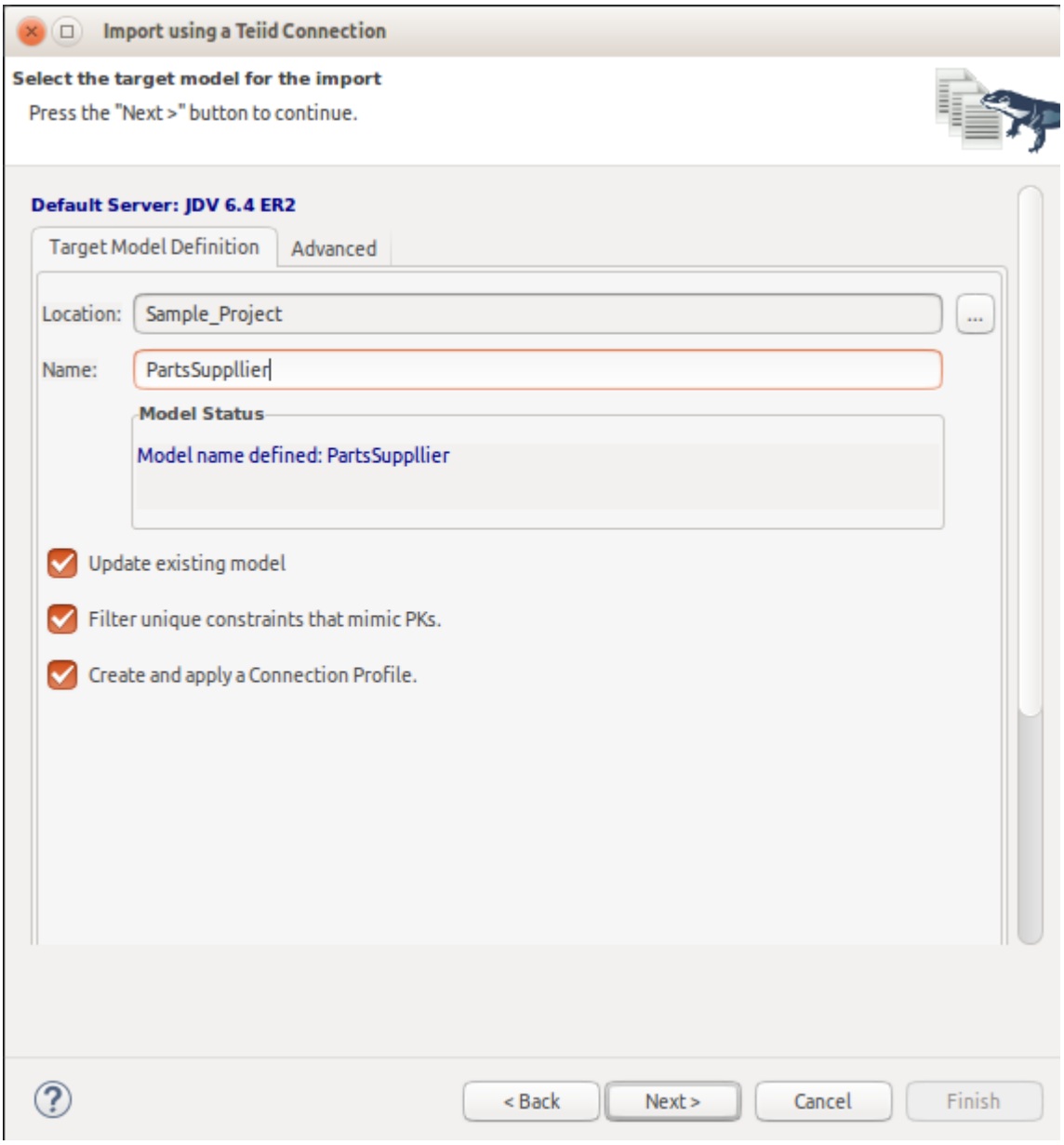


Figure 5.13. Target Model Definition

- **Step 6** - When you move to next page of the wizard, a temporary dynamic vdb is actually deployed to you rserver and the schema your data source is retrieved in DDL form. This DDL is displayed (and can also be exported if desired). Click **NEXT>** to continue.

Figure 5.14. Review DDL Dialog

- **Step 7** - On the final page of the wizard, a difference report is presented for viewing or de-selecting individual relational entities. Press **Finish** to complete.

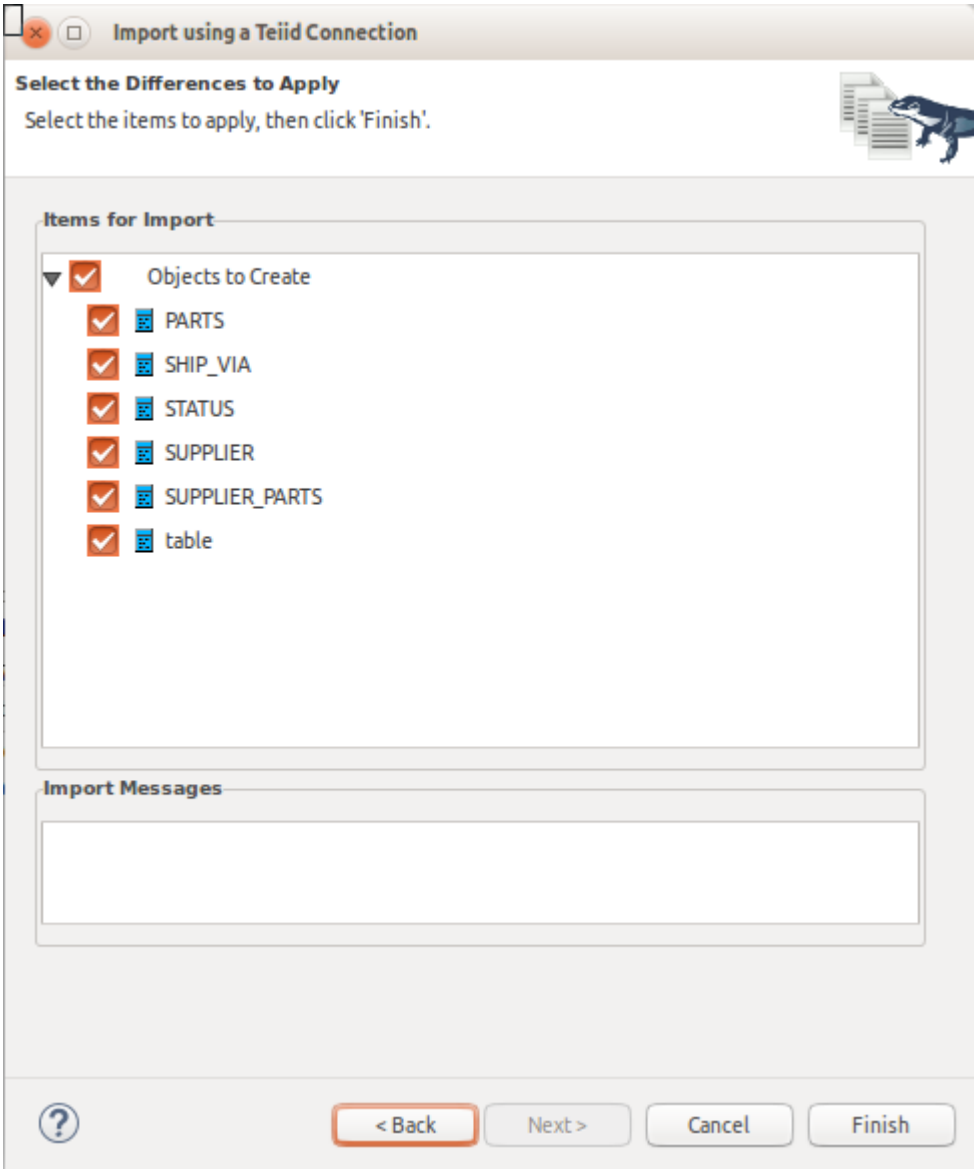


Figure 5.15. Review Model Updates Dialog

5.4. Import From Flat File Source

- You can import metadata from your flat file data sources and create the metamodels required to query your data in minutes. Using the steps below you will define your flat file data source, configure your parsing paramaters for the flat file, generate a source model containing the standard Teiid flat file procedure and create view tables containing the SQL defining the column data in your flat file.

  Teiid supports Flat Files as data sources. Teiid Designer provides an Import wizard designed to assist in creating the metadata models required to access the data in your flat files. As with Designer's JDBC, Salesforce and WSDL importers, the Flat File importer is based on utilizing a specific Data Tools Connection Profile.

The results of the importer will include a source model containing the getTextFiles() procedures supported by Teiid.

The importer will also create a new view model containing a view table for your selected flat file source file. Within the view table will be generated SQL transformation containing the "getTextFiles()" procedure from your source model as well as the column definitions and parameters required for the Teiid TEXTTABLE() function used to query the data file. You can also choose to update an existing view model instead of creating a new view model.

The TEXTTABLE function, as defined in the Teiid documentation, processes character input to produce tabular ouptut. It supports both fixed and delimited file format parsing. The function itself defines what columns it projects. The TEXTTABLE function is implicitly a nested table and may be correlated to preceeding FROM clause entries.

```
TEXTTABLE(expression COLUMNS <COLUMN>, ...
    [DELIMITER char] [(QUOTE|ESCAPE) char] [HEADER [integer]] [SKIP
    integer]) AS name
```

Teiid Designer will construct the full SQL statement for each view table in the form:

```
SELECT A.Name, A.Sport,
    A.Position, A.Team, A.City, A.StateCode, A.AnnualSalary FROM (EXEC
    PlayerDataSource.getTextFiles('PlayerData.txt')) AS f,
    TEXTTABLE(f.file COLUMNS Name string, Sport string, Position
    string, Team string, City string, StateCode string, AnnualSalary
    string HEADER 2 SKIP 3) AS A
```

To import from your flat file source follow 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 > File Source (Flat) >> Source and View Model** and click **Next>**

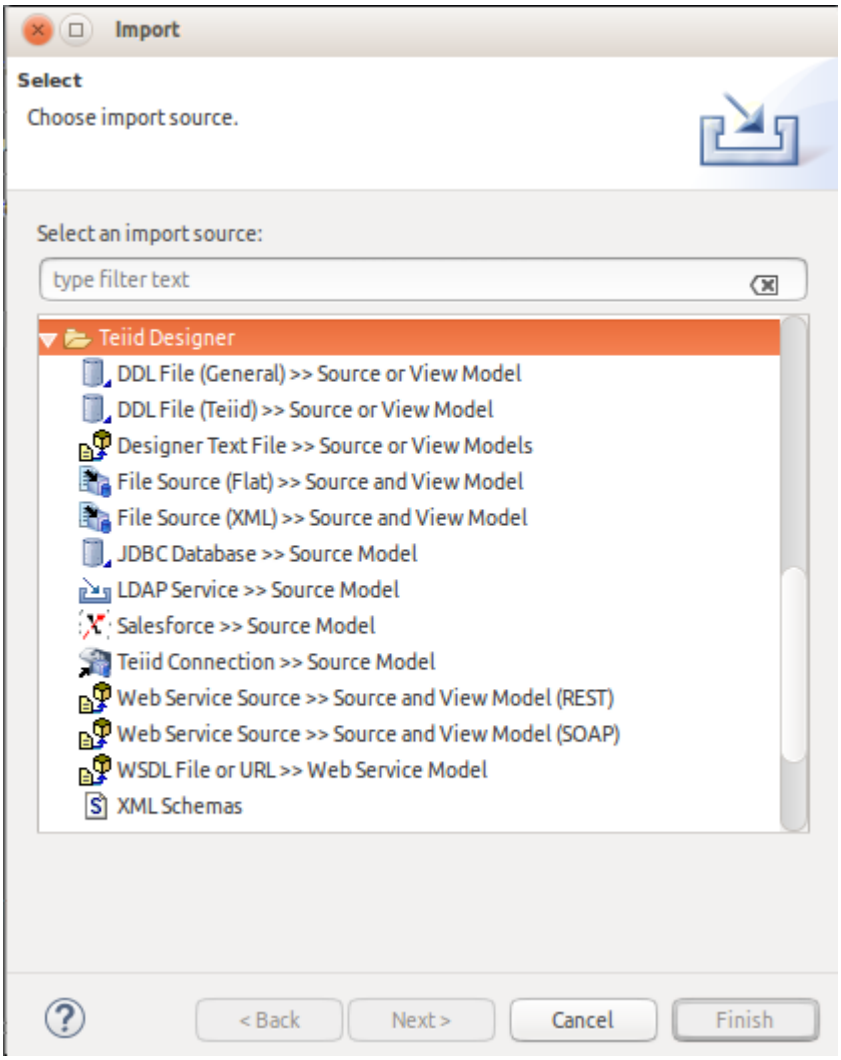


Figure 5.16. Import from Flat File Source

- **Step 3** - On the first wizard page, select the flat file mode you wish to import, either **Flat file on local file system** or **Flat file via remote URL**

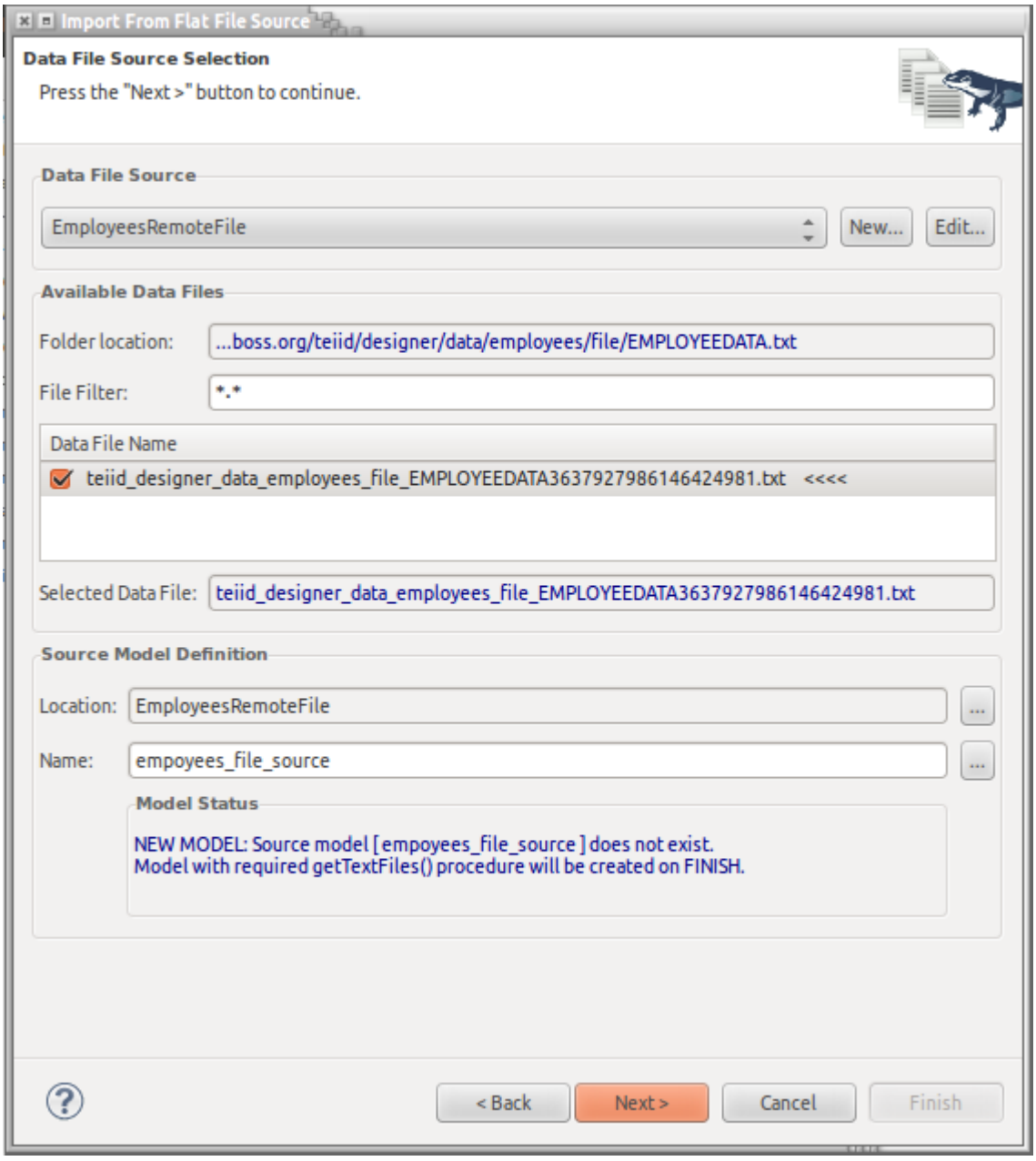


Figure 5.17. Flat File Source Model Options

Note that the local file system connection specifies a folder containing one or more comma separated text data file : **/home/jdoe/employees/**. The remote URL connection will specify a URL to a single data file : **http://download.jboss.org/teiid/designer/data/employees/file/EMPLOYEEDATA.txt**.

- **Step 4** - 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.

Note

The Flat File Source selection list will be populated with only Flat File connection profiles.

After selecting a **Connection Profile**, the file contents of the folder defined in the connection profile will be displayed in the **Available Data Files** panel. Check the the data file you wish to process. The data from this file, along with your custom import options, will be used to construct a view table containing the required SQL transformation for retrieving your data and returning a result set.

Lastly enter or unique source model name in the **Source Model Definition** section at the bottom of the page or select an existing source model using the browse button.

Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not and whether the model already contains the getTextFiles() procedure. In this case, the source model nor the procedure will be generated.

When finished with this page, click **Next>**.

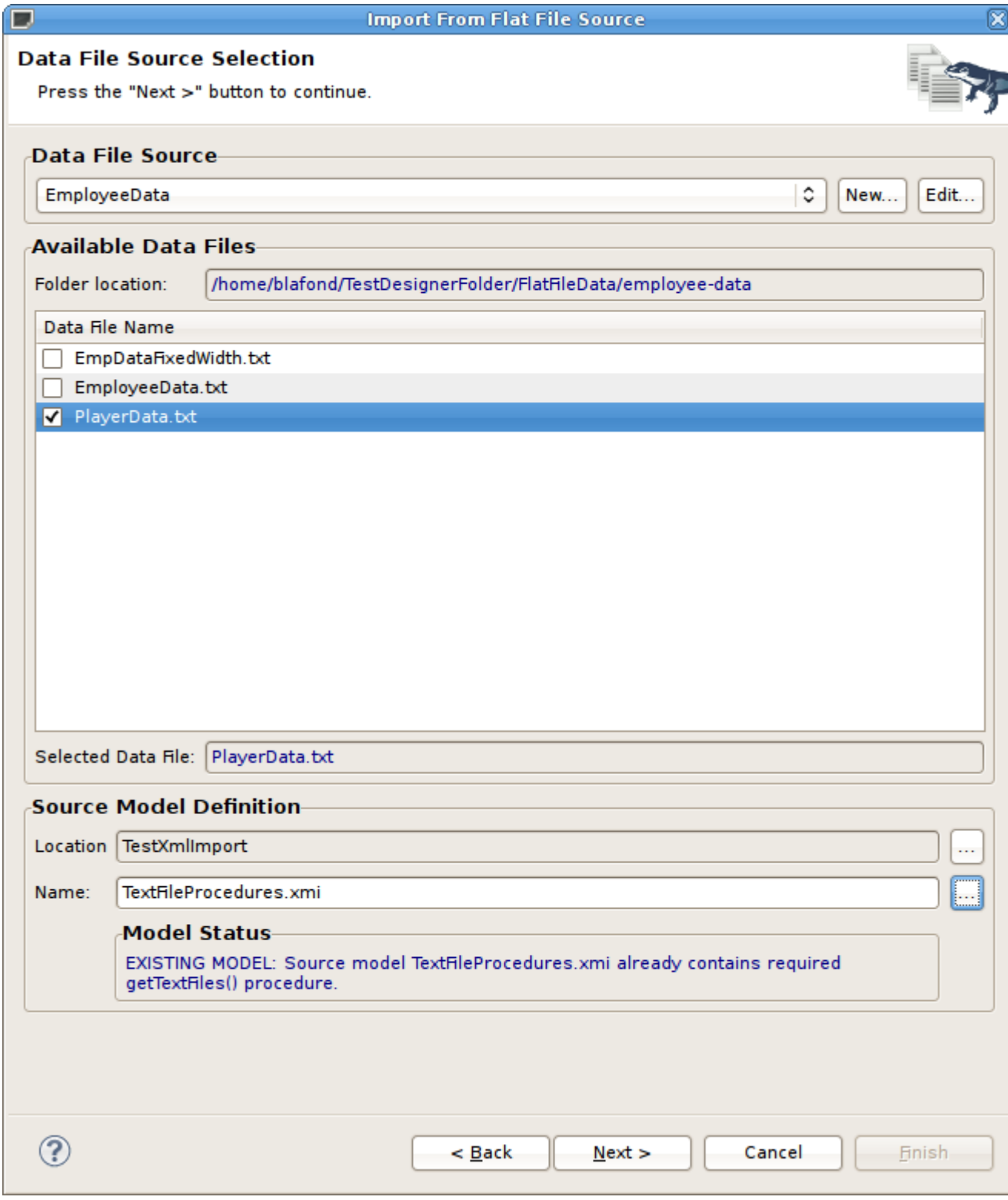


Figure 5.18. Data File Source Selection Page

- **Step 5** - The next page, titled **Data Source Definition Options**, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model

will prepresent. This name will be stored as a property on your model and changeable via the **Modeling > Set JBoss JNDI Data Source Name** action. If there you have a default server defined and running, then **Auto-Create Data Source** check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

When finished with this page, click **Next>**.

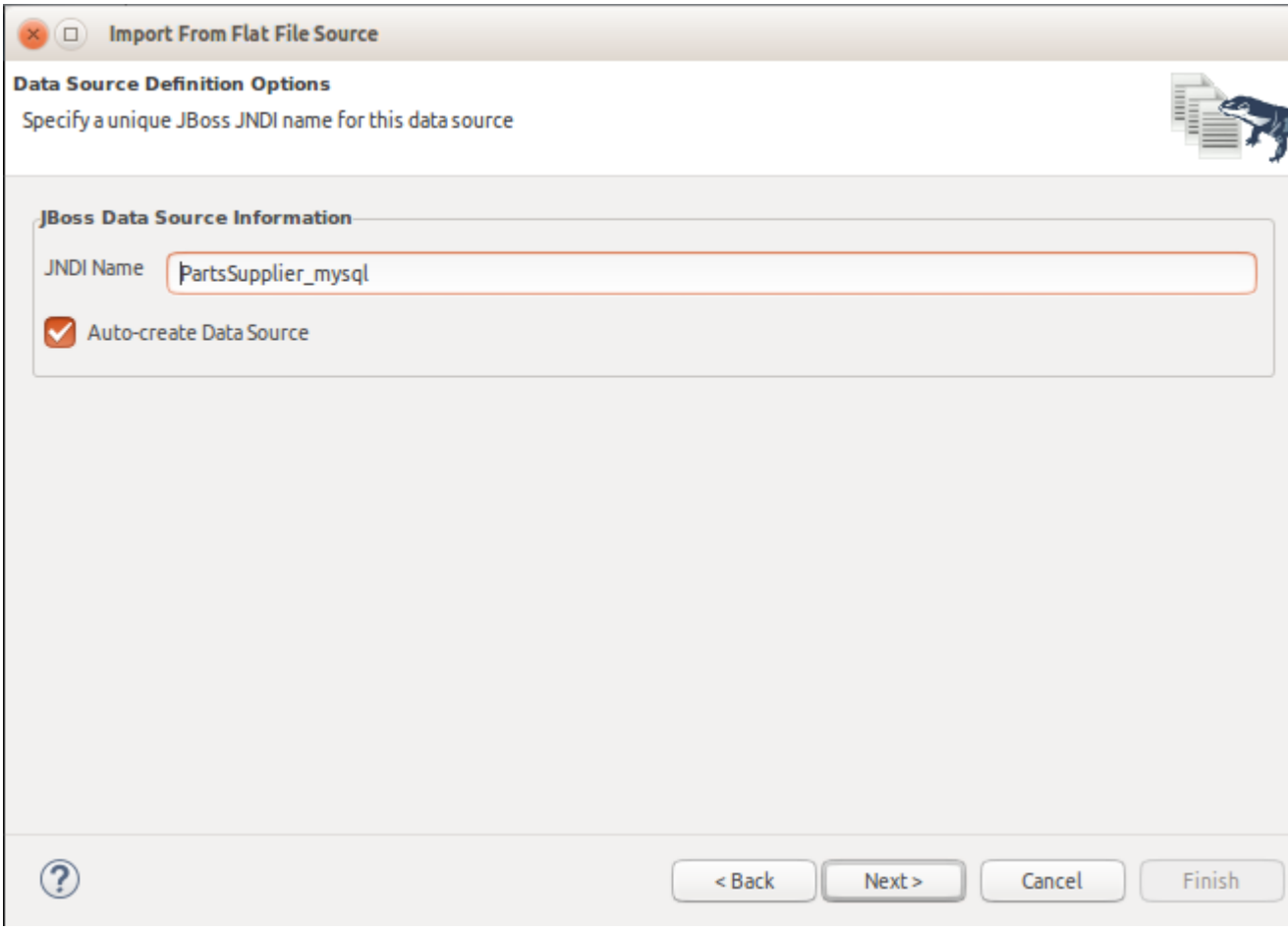


Figure 5.19. Data File Source Selection Page

- **Step 6** - The next page, titled **Flat File Column Format Definition**, requires defining the format of your column data in the file. The options are **Character delimited** and **Fixed width**. This page contains a preview of the contents of your file to aid in determining the format. The wizard defaults to displaying the first 20 lines, but you can change that value if you wish.

  When finished with this page, click **Next>**.

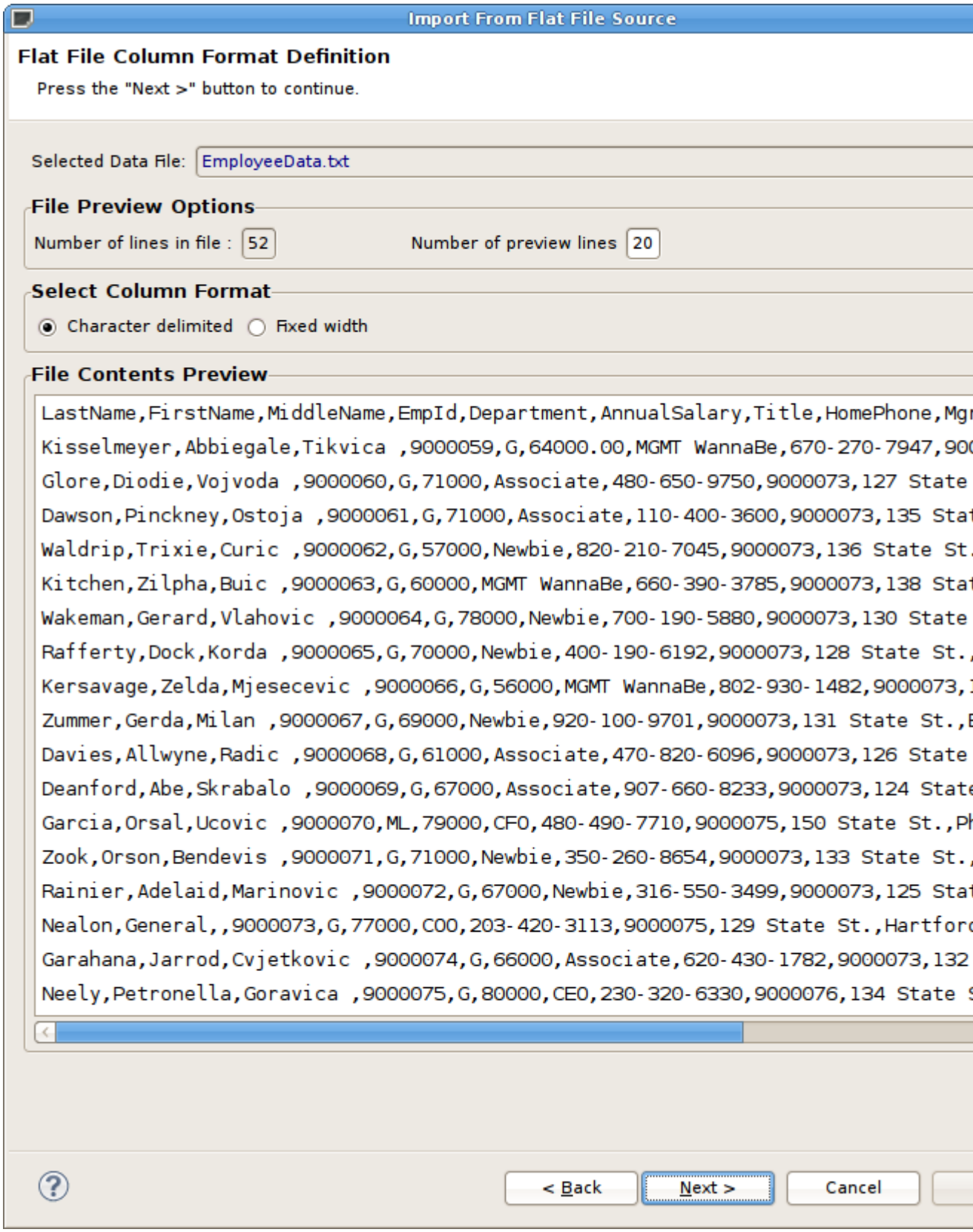


Figure 5.20. Data File Source Selection Page

- **Step 7a : Character Delimited Option** - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the **Generated SQL Statement** , shown in the bottom panel, for the character delimited option. Specify header options (Column names in header, header line number and first data line number), Parse selected row, changed character delimiter and edit the TEXTTABLE() function options. See the Teiid User's Guide for details on the TEXTTABLE() function.

  If columns names are not defined in a file header or if you wish to modify or create custom columns, you can use the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.

  When finished with this page, click **Next>**.

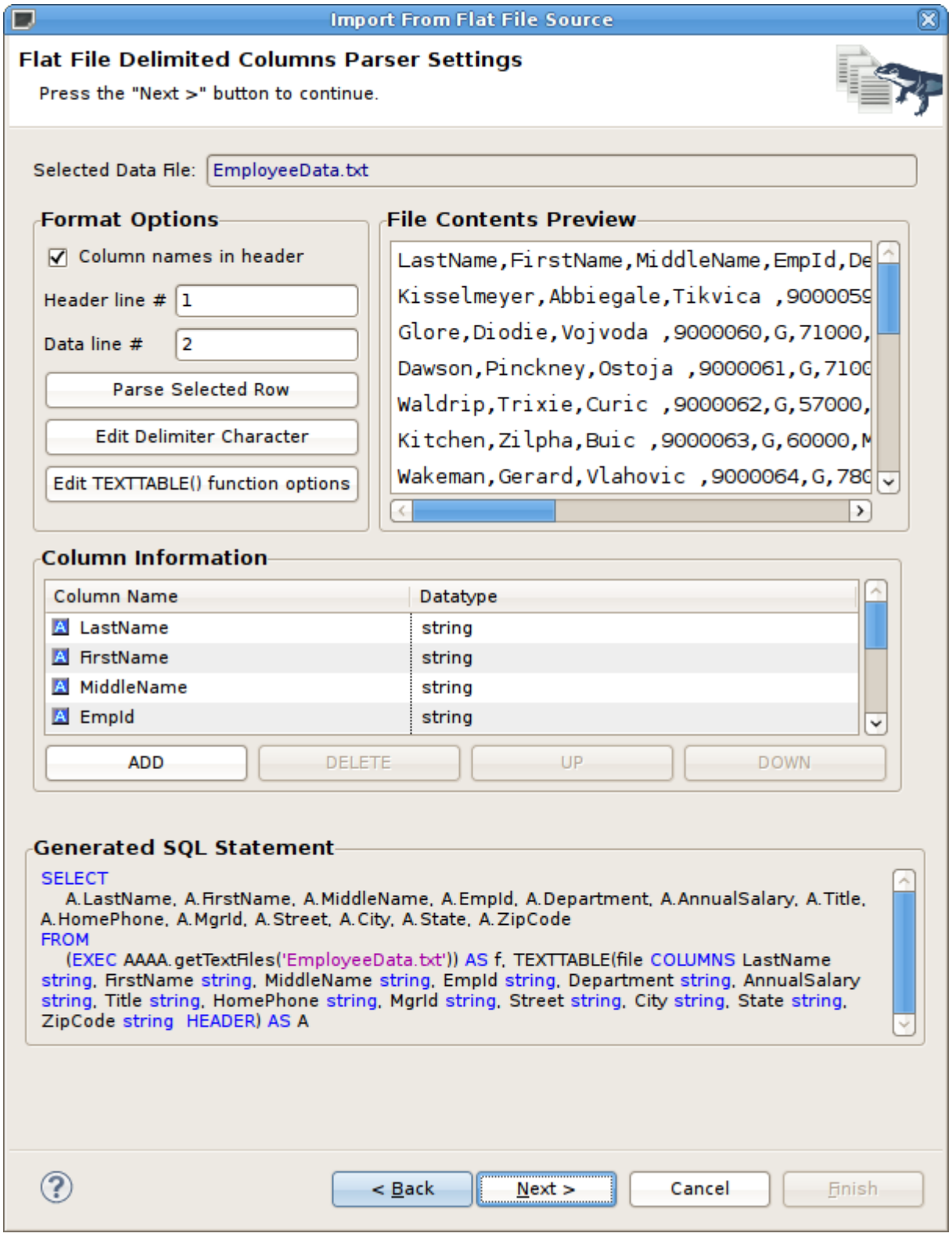


Figure 5.21. Flat File Delimited Columns Options Page

To aid in determining if your parser settings are correct you can select a data row in your **File Contents Preview** section and click the **Parse Selected Row** button. A dialog will be displayed showing the list of columns and the resulting column data. If your column data is not what you expected, you'll need to adjust your settings accordingly.

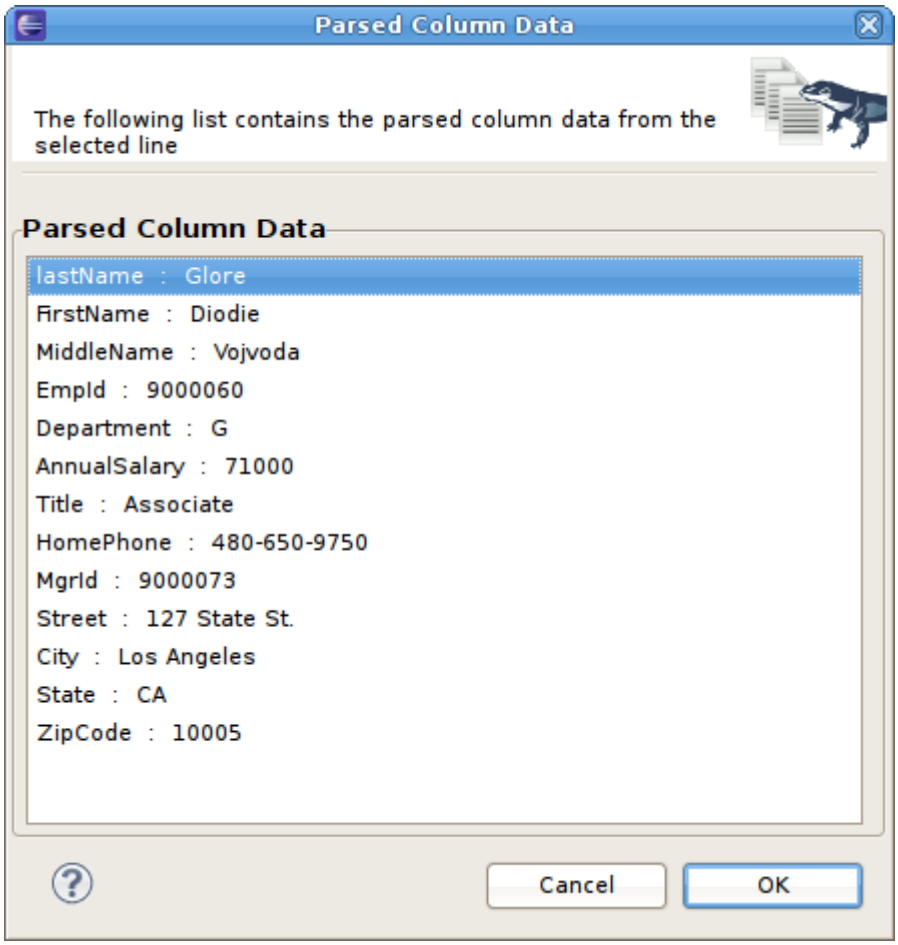


Figure 5.22. Parse Column Data Dialog

- **Step 7b : Fixed Column Width Option** - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the **Generated SQL Statement** , shown in the bottom panel, for the fixed column width option. Specify header options (Column names in header, header line number and first data line number), Parse selected row, changed character delimiter and edit the TEXTTABLE() function options. See the Teiid User's Guide for details on the TEXTTABLE() function.

  If columns names are not defined in a file header or if you wish to modify or create custom columns, you can use the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.

  You can also utilize the cursor postion and text length values in the upper left panel to determine what your column widths are in your data file.

When finished with this page, click **Next>**.

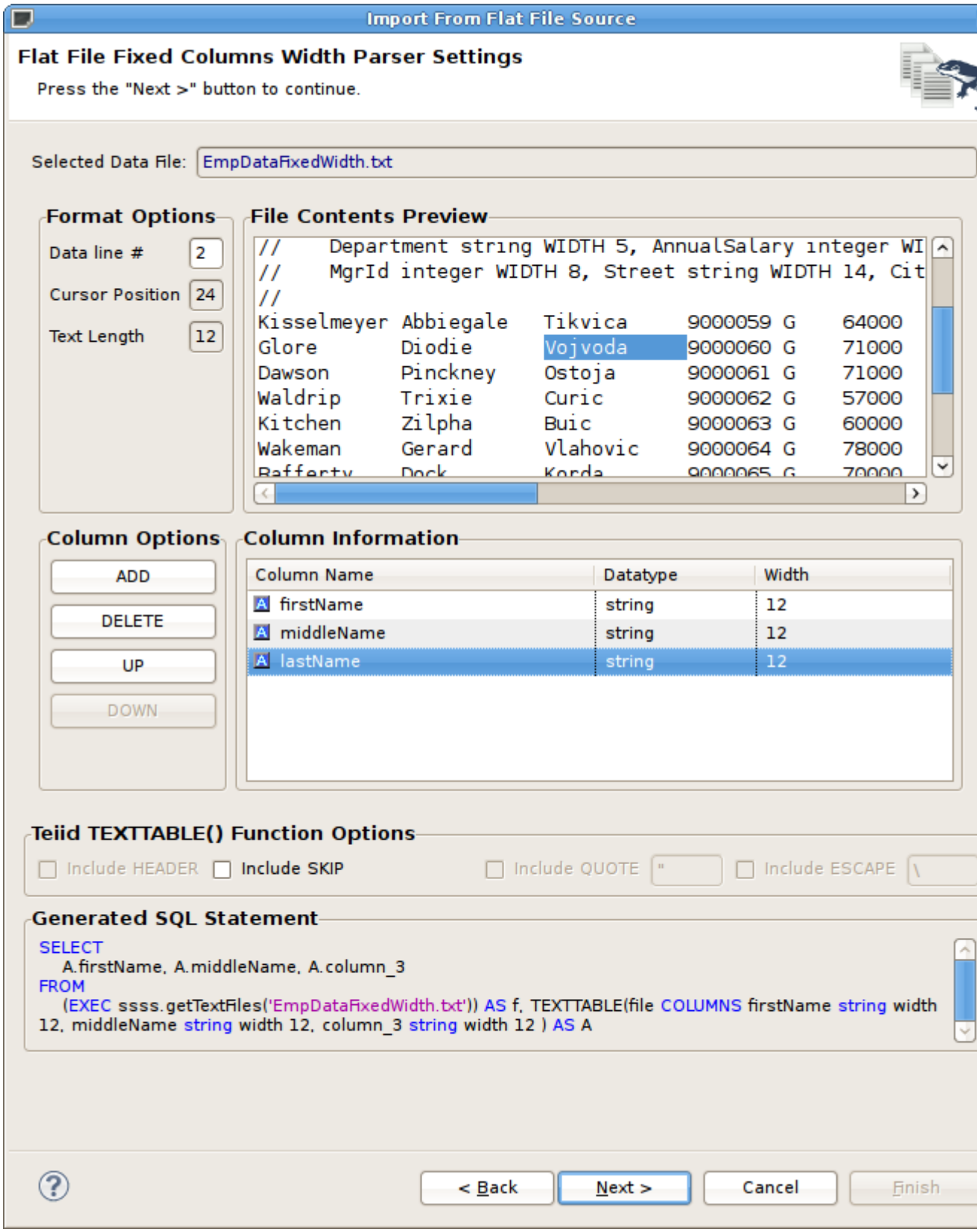


Figure 5.23. Flat File Fixed Columns Width Options Page

- **Step 8** - On the **View Model Definition** page, select the target folder location where your new view model will be created. You can also select an existing model for your new view tables.

> **Note**
>
> The **Model Status** section which will indicate the validity of the model name, whether the model exists or not. Lastly, enter a unique, valid view table name.

Press **Finish** to generate your models and finish the wizard.

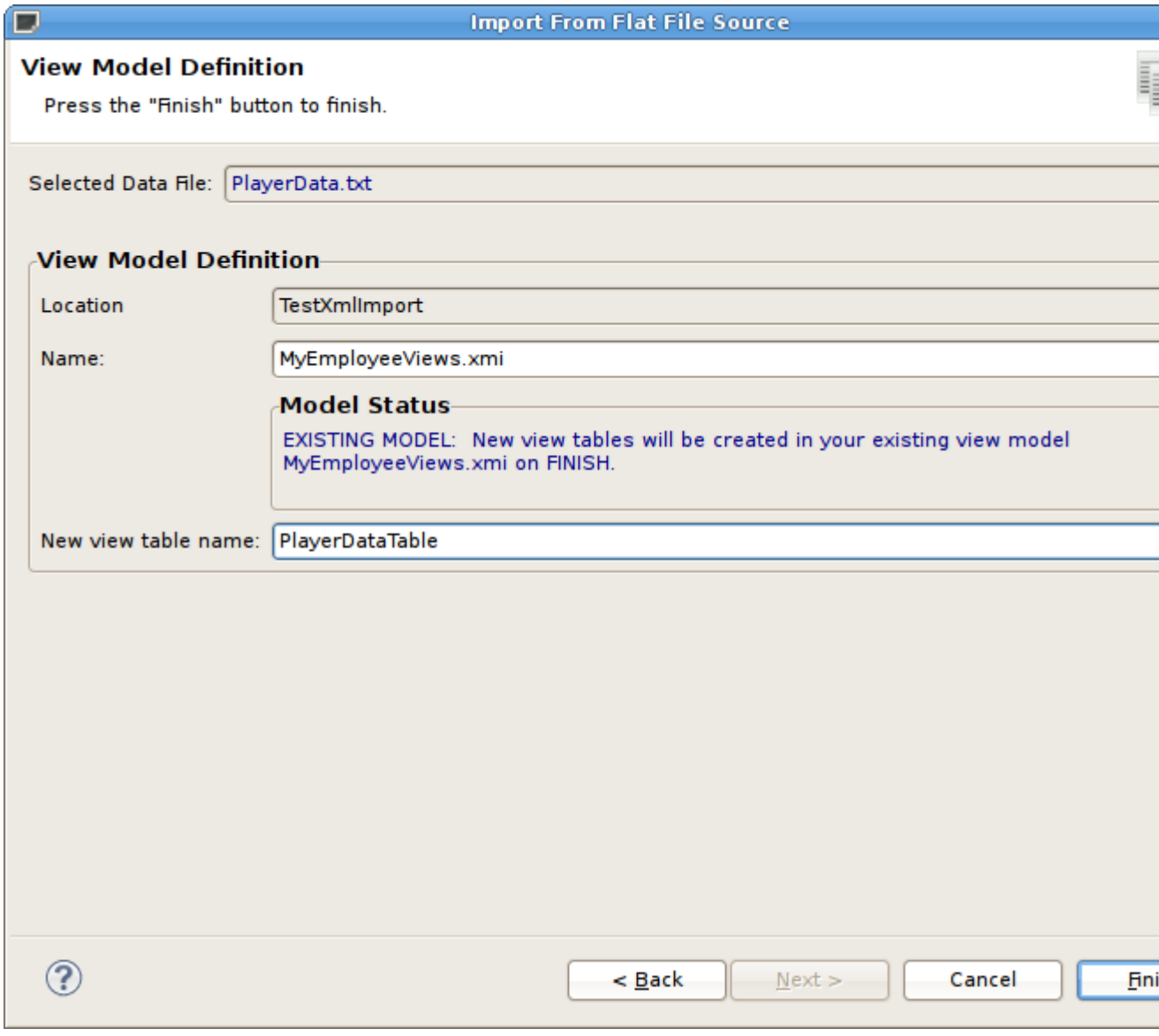


Figure 5.24. View Model Definition Page

When your import is finished your source model will be opened in an editor and show a diagram containing the your getTextFiles() procedure.

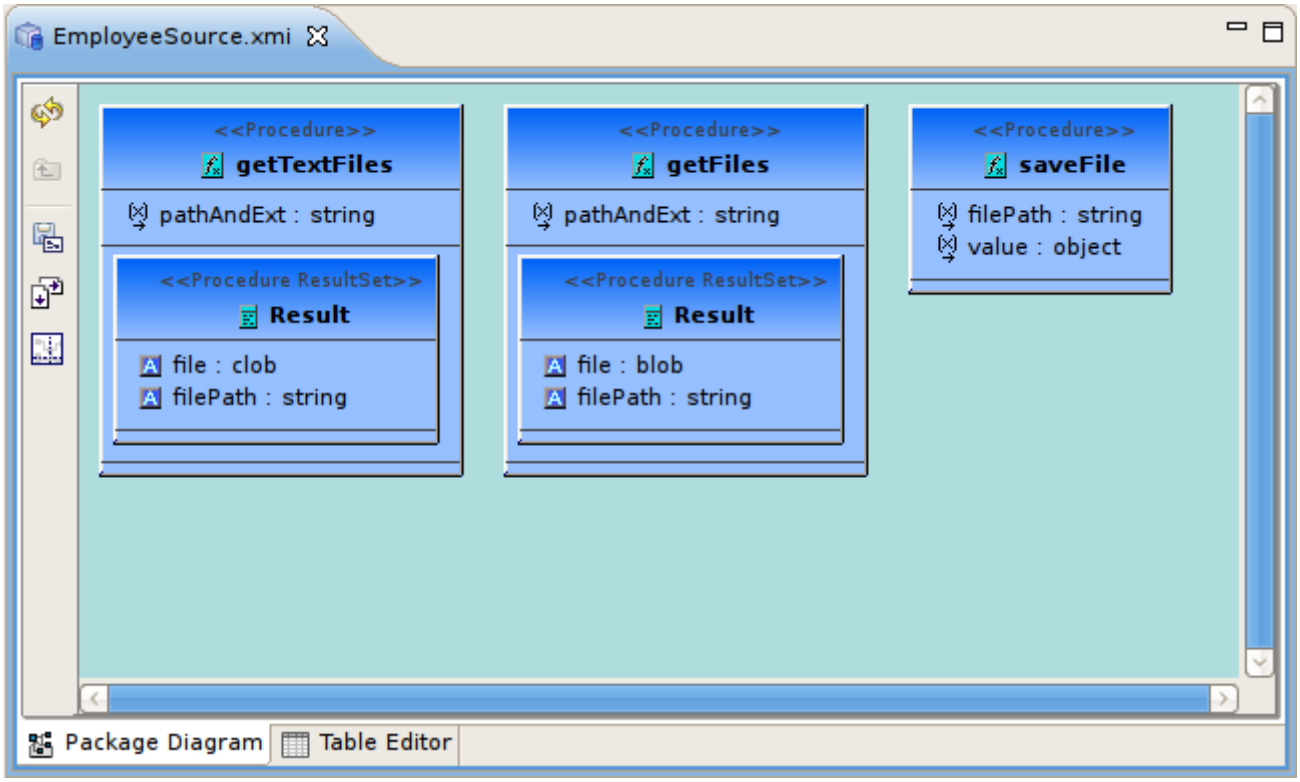


Figure 5.25. Generated Flat File Procedures

In addition, the view model will be opened in an editor and will show the generated view tables containing the completed SQL required to access the data in your flat file using the "getTextFiles" procedure above and the Teiid TEXTTABLE() function. The following figure is an example of a generated view table.

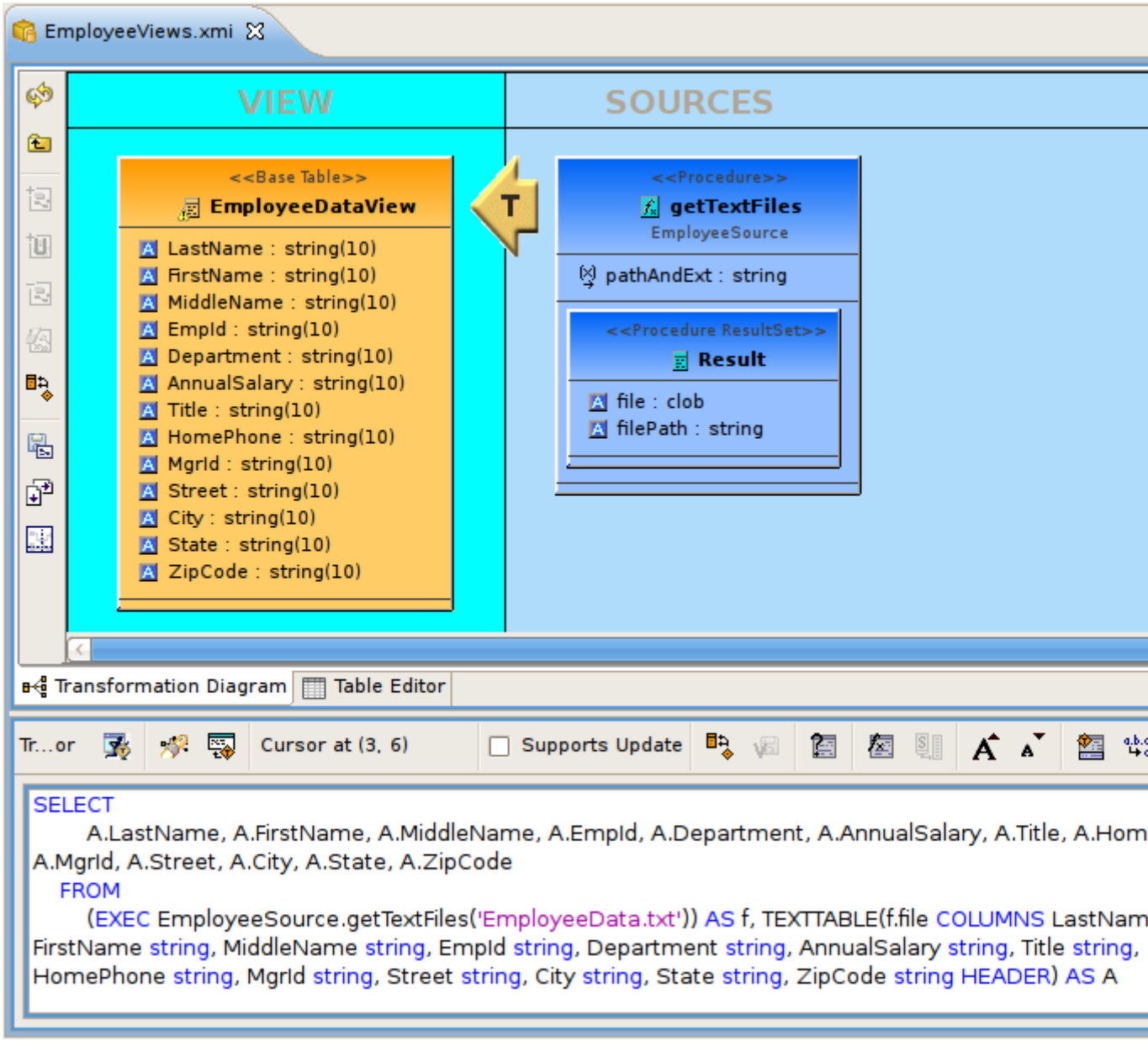


Figure 5.26. Generated Flat File View Table

5.5. Import From XML Data File Source

- Teiid supports XML Files as data sources. You can import from these data sources and create the metamodels required to query your data in minutes. Using the steps below you will define your flat file data source, configure your parsing paramaters for the xml data file, generate a source model containing the required Teiid procedure and create a view table containing the SQL defining the column data in your xml data file.

  As with Designer's JDBC, Salesforce and WSDL importers, the XML File importer is based on utilizing a specific Data Tools Connection Profile.

The results of the importer will include a source model containing the getTextFiles() procedure or invokeHTTP() procedure which are both supported by Teiid.

The importer will also create a new view model containing a view table for your selected flat file source file. Within the view table will be generated SQL transformation containing the "getTextFiles()" procedure from your source model as well as the column definitions and parameters required for the Teiid XMLTABLE() function used to query the data file. You can also choose to update an existing view model instead of creating a new view model.

The XMLTABLE function uses XQuery to produce tabular ouptut. The XMLTABLE function is implicitly a nested table and may be correlated to preceeding FROM clause entries. XMLTABLE is part of the SQL/XML 2006 specification.

```
XMLTABLE([<NSP>,] xquery-expression
    [<PASSING>] [COLUMNS <COLUMN>, ... )] AS name
```

```
COLUMN := name (FOR ORDINALITY | (datatype [DEFAULT
    expression] [PATH string]))
```

Teiid Designer will construct the full SQL statement for each view table in the form:

```
SELECT A.entryDate AS entryDate,
    A.internalAudit AS internalAudit FROM (EXEC
    CCC.getTextFiles('sample.xml')) AS f,
    XMLTABLE(XMLNAMESPACES('http://www.kaptest.com/schema/1.0/party'
    AS pty), '/pty:students/student' PASSING
    XMLPARSE(DOCUMENT f.file) COLUMNS entryDate FOR ORDINALITY,
    internalAudit string PATH '/internalAudit') AS
    A
```

To import from your XML data file source follow 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 > File Source (XML) >> Source and View Model** and click **Next>**

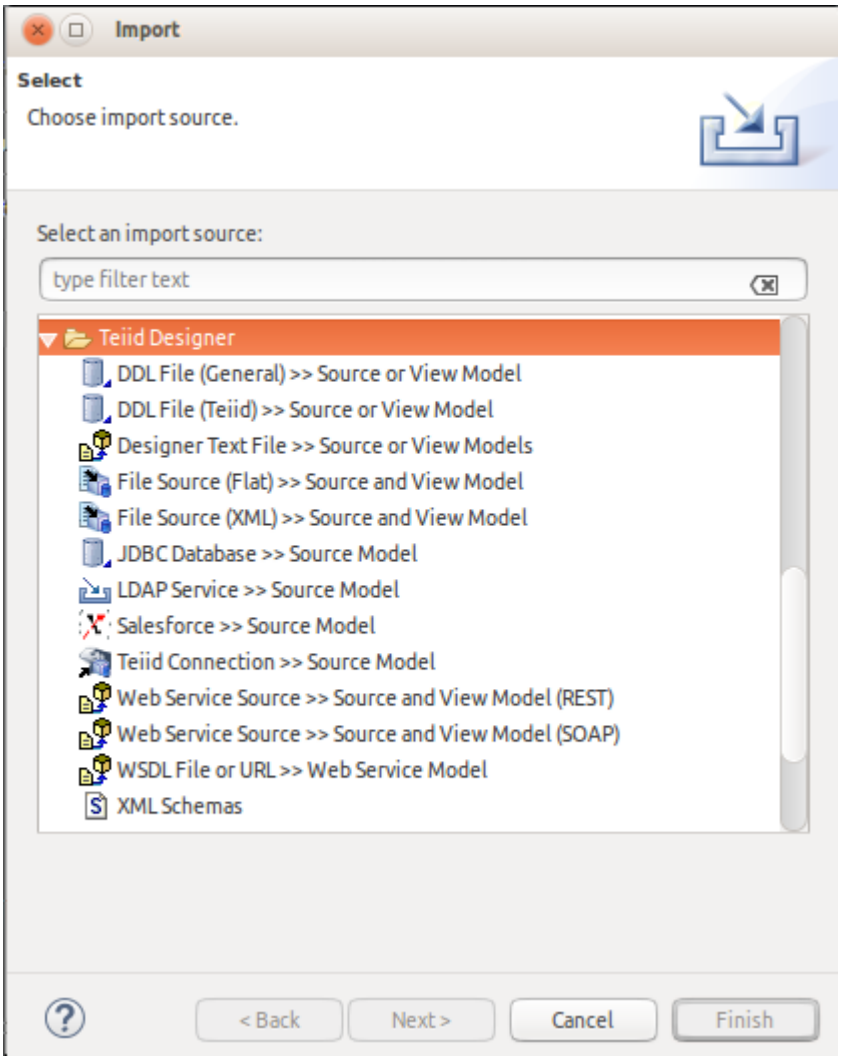


Figure 5.27. Import from XML File Source

- **Step 3** - The next page of the wizard allows selection of the XML Import mode that specifies whether the XML file is local or remote. The description at the top describes what operations this wizard will perform. Select either the **XML file on local file system** or **XML file via remote URL** and click **Next>**

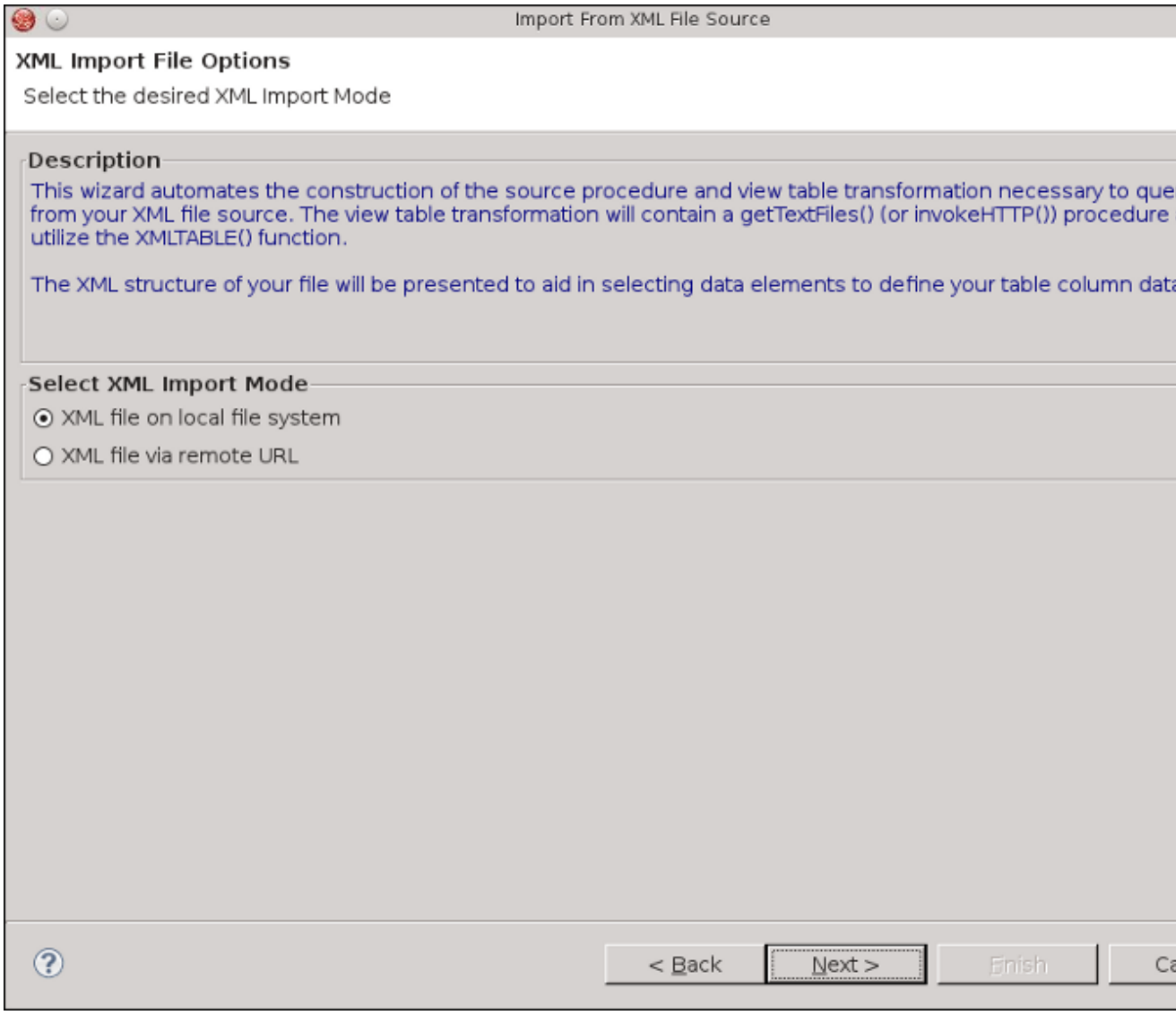


Figure 5.28. XML Import File Options Page

- **Step 4** - 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.

  After selecting a **Connection Profile**, the XML data file from the connection profile will be displayed in the **Available Data Files** panel. Check the the data file you wish to process. The data from this file, along with your custom import options, will be used to construct a view table containing the required SQL transformation for retrieving your data and returning a result set.

  Lastly enter or unique source model name in the **Source Model Definition** section at the bottom of the page or select an existing source model using the browse button.

Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not and whether the model already contains the getTextFiles() procedure. In this case, the source model nor the procedure will be generated.

When finished with this page, click **Next>**.

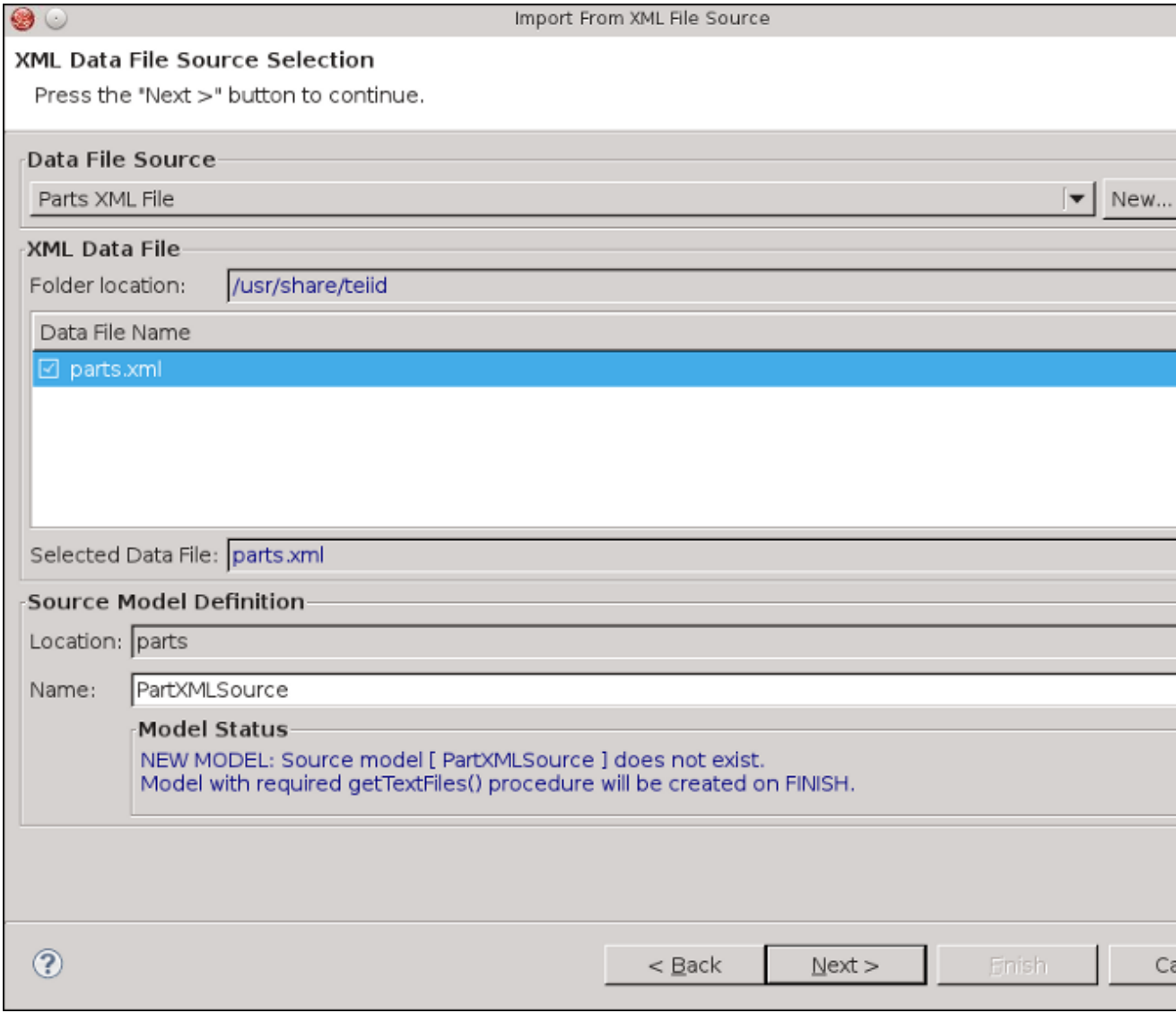


Figure 5.29. XML Data File Source Selection Page

- **Step 5** - The next page, titled **Data Source Definition Options**, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model will prepresent. This name will be stored as a property on your model and changeable via

the **Modeling > Set JBoss JNDI Data Source Name** action. If there you have a default server defined and running, then **Auto-Create Data Source** check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

When finished with this page, click **Next>**.

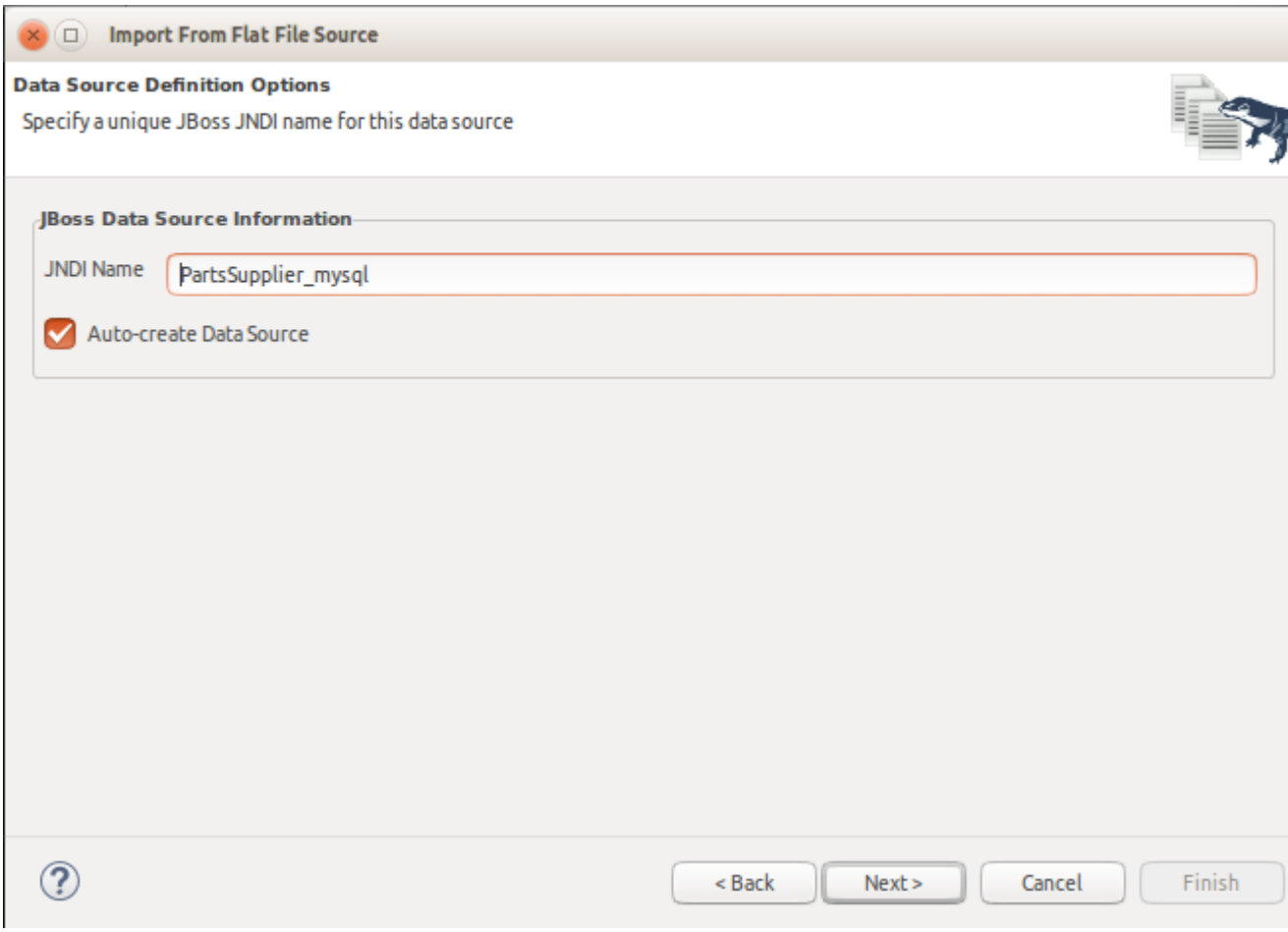


Figure 5.30. Data File Source Selection Page

- **Step 6** - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the **Generated SQL Statement** , shown in the bottom panel. The to panel contains an XML tree view of your file contents and actions/buttons you can use to create column entries displayed in the middle, **Column Information** panel.

  To create columns, select a root XML element and right-click select **Set as root path** action. This populates the root path value. Next, select columns in the tree that you wish to include on your query and select **Add selection as new column** button. You can also modify or create custom columns, by using the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.

Note

The **Path** property value for a column is the selected element's path relative to the defined root path. If no root path is defined all paths are absolute. Each column entry requires a datatype and an optional default value. See the Teiid User's Guide for details on the XMLTABLE() function.

When finished with this page, click **Next>**.

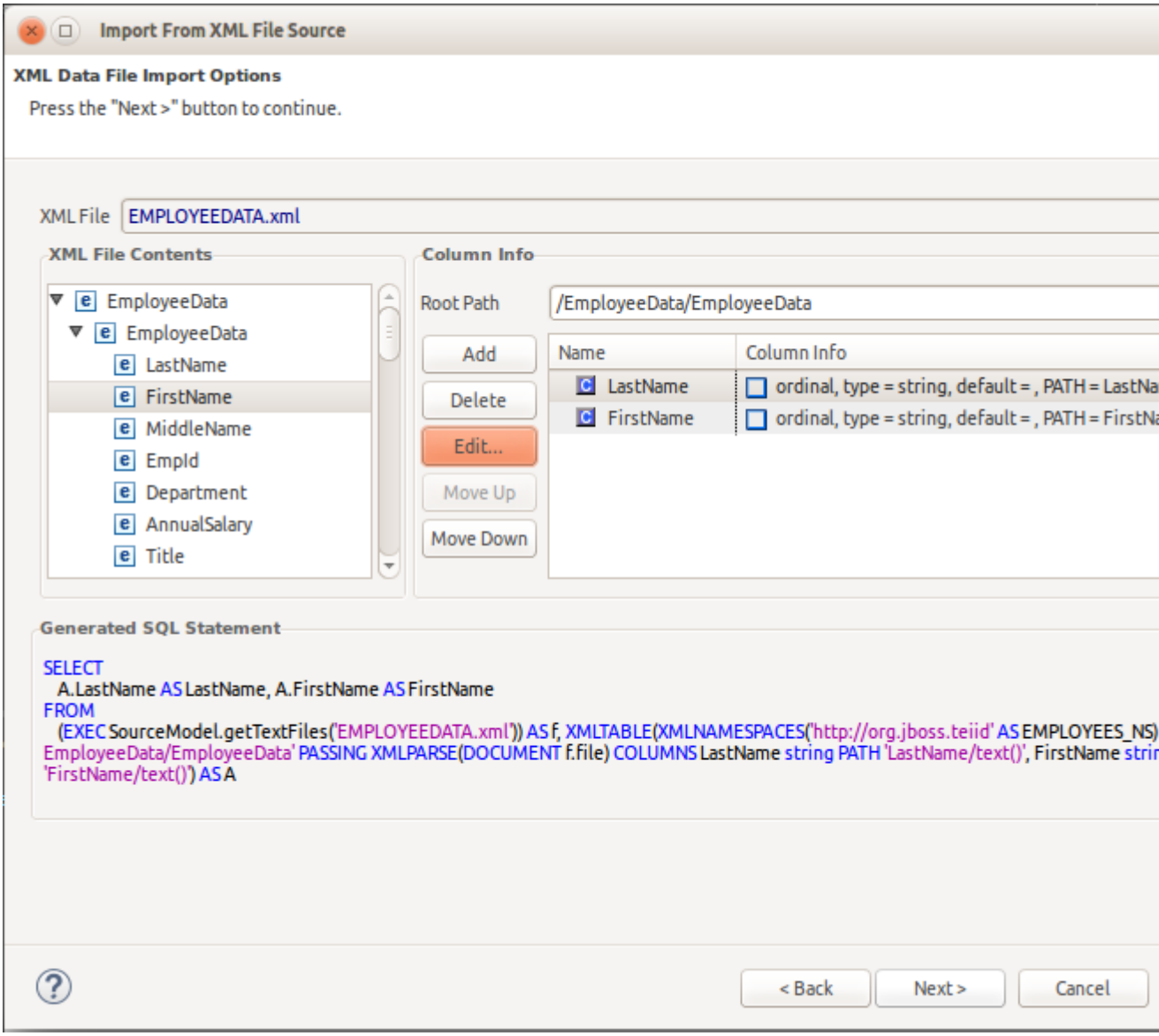


Figure 5.31. XML File Delimited Columns Options Page

- **Step 7** - On the **View Model Definition** page, select the target folder location where your new view model will be created. You can also select an existing model for your new view tables.

Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not. Lastly, enter a unique, valid view table name.

Press **Finish** to generate your models and finish the wizard.

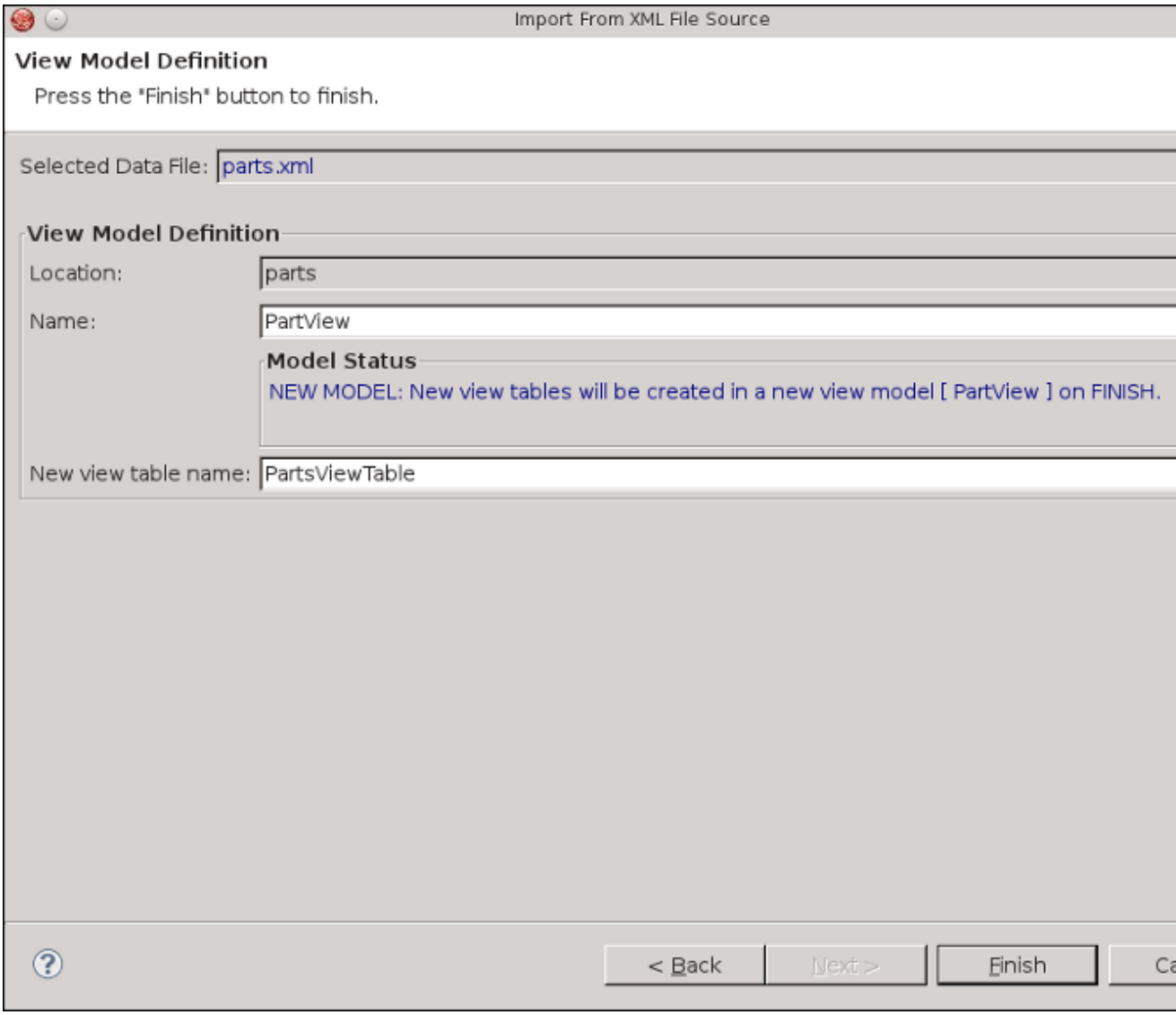


Figure 5.32. View Model Definition Page

5.6. Import From Salesforce

- You can create relational source models from your Salesforce connection 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 **Teiid Designer > Salesforce >> Source Model** and click **Next>**
- **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.

> **Note**
>
> The Connection Profile selection list will be populated with only Salesforce connection profiles.

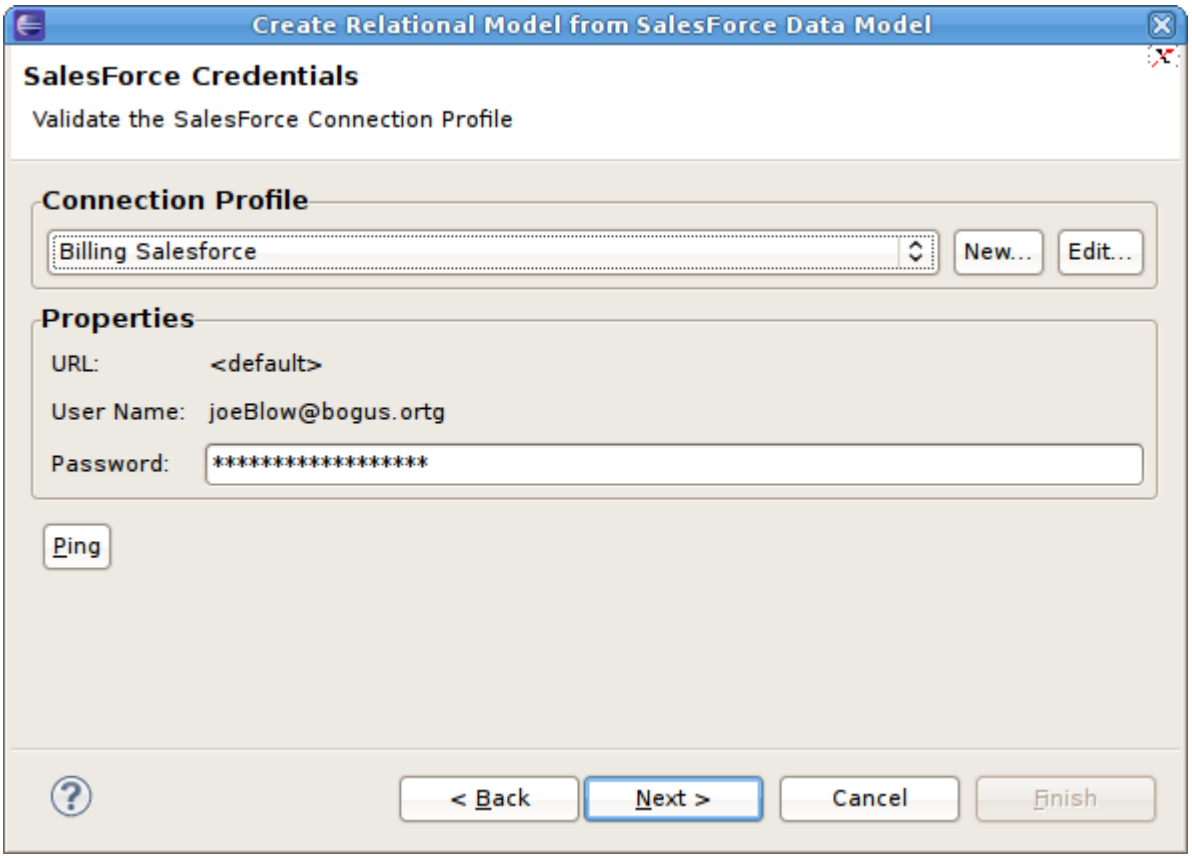


Figure 5.33. Select Salesforce Credentials Dialog

- **Step 4** - After selecting a *Connection Profile*, input password (if not provided). Press **Next>** to display the Salesforce Objects selection page.

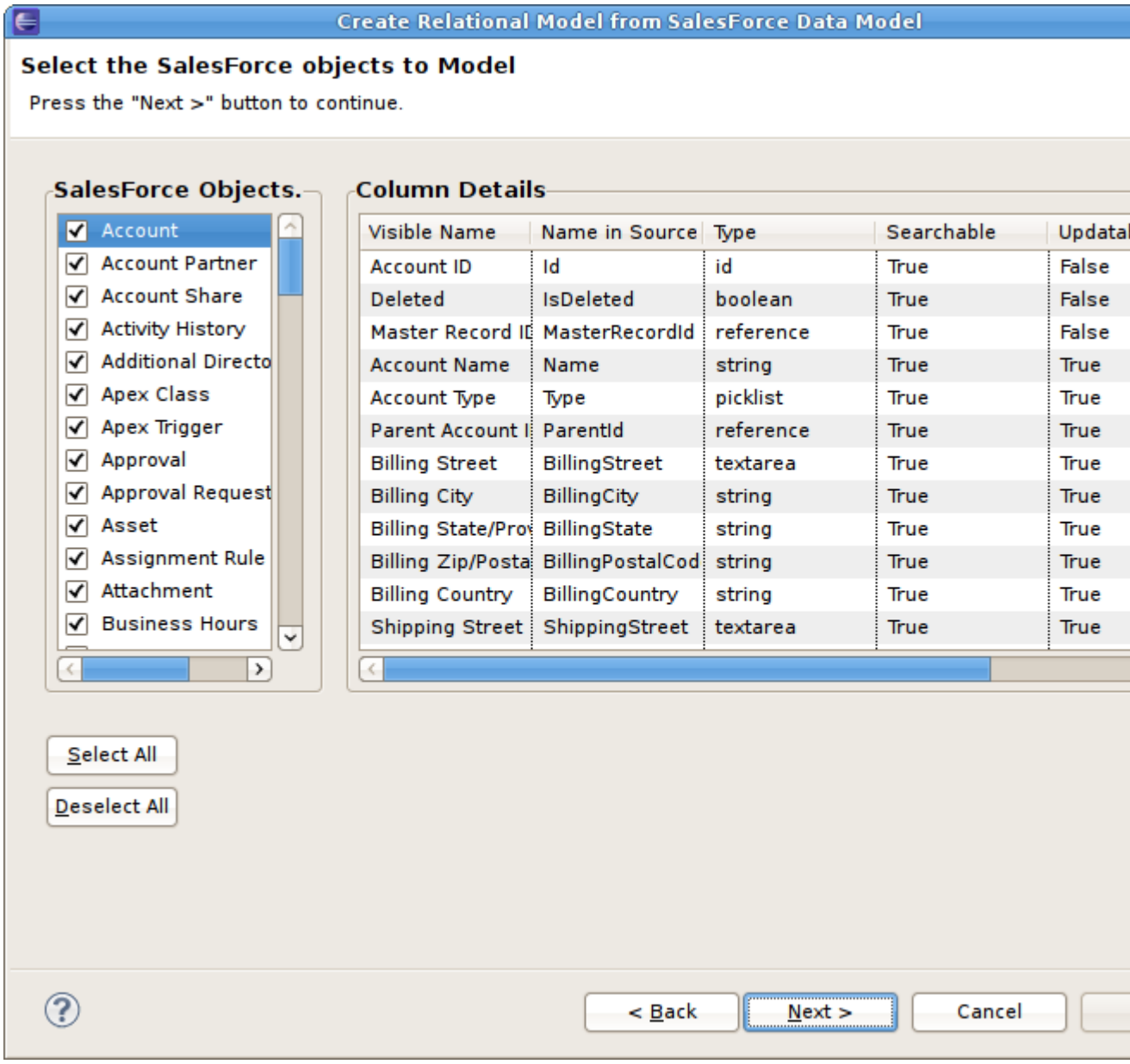


Figure 5.34. Select Salesforce Objects Dialog

- **Step 5** - On the **Target Model Selection** page, specify the target folder location for your generated model, a unique model name and select desired import options. Press **Next>** (or **Finish** if enabled).

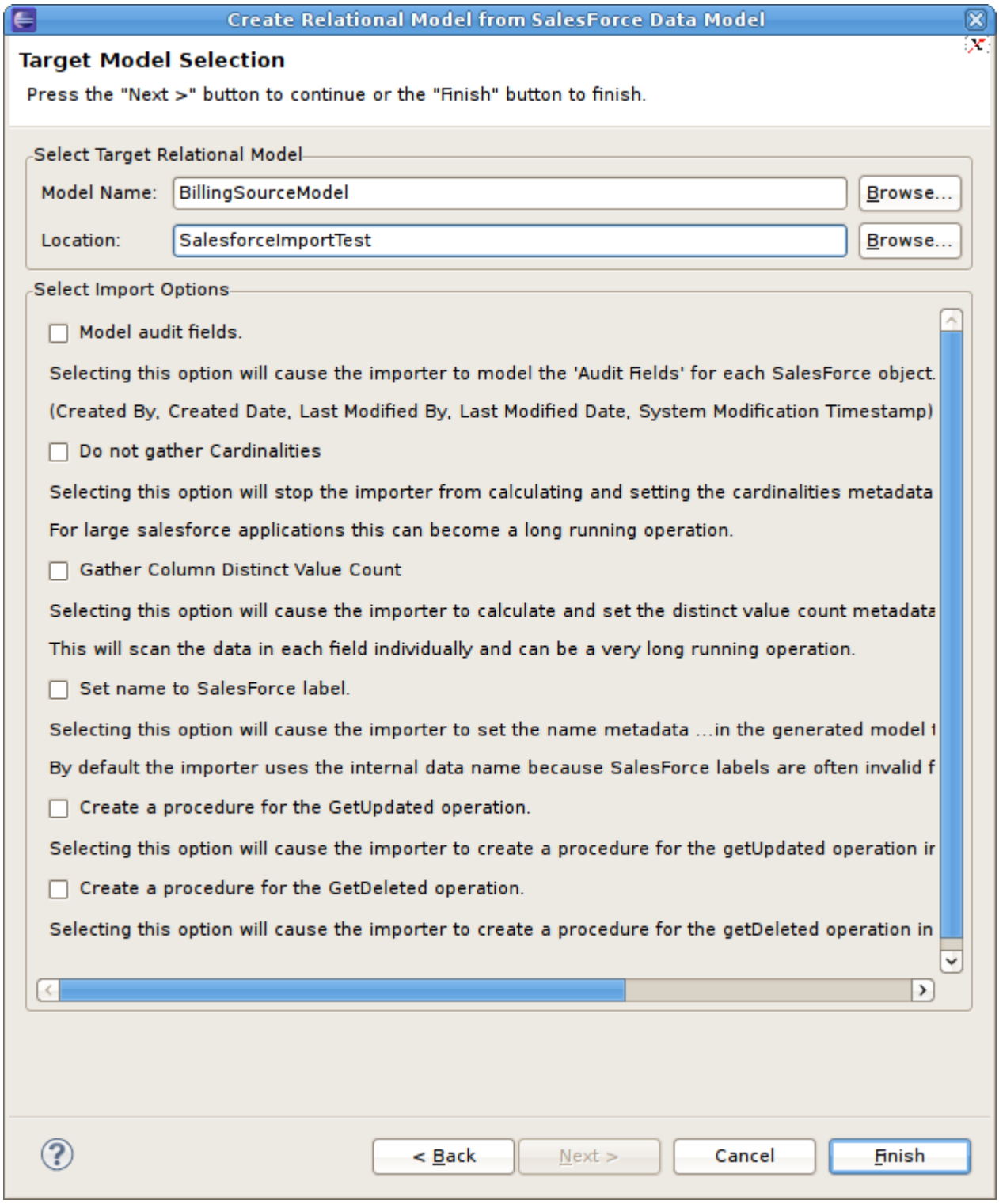


Figure 5.35. Target Model Selection Dialog

- **Step 5a** - If you are updating an existing relational model, the next page will be **Review Model Updates** page. Any differences. Press **Finish** to create your models and tables.

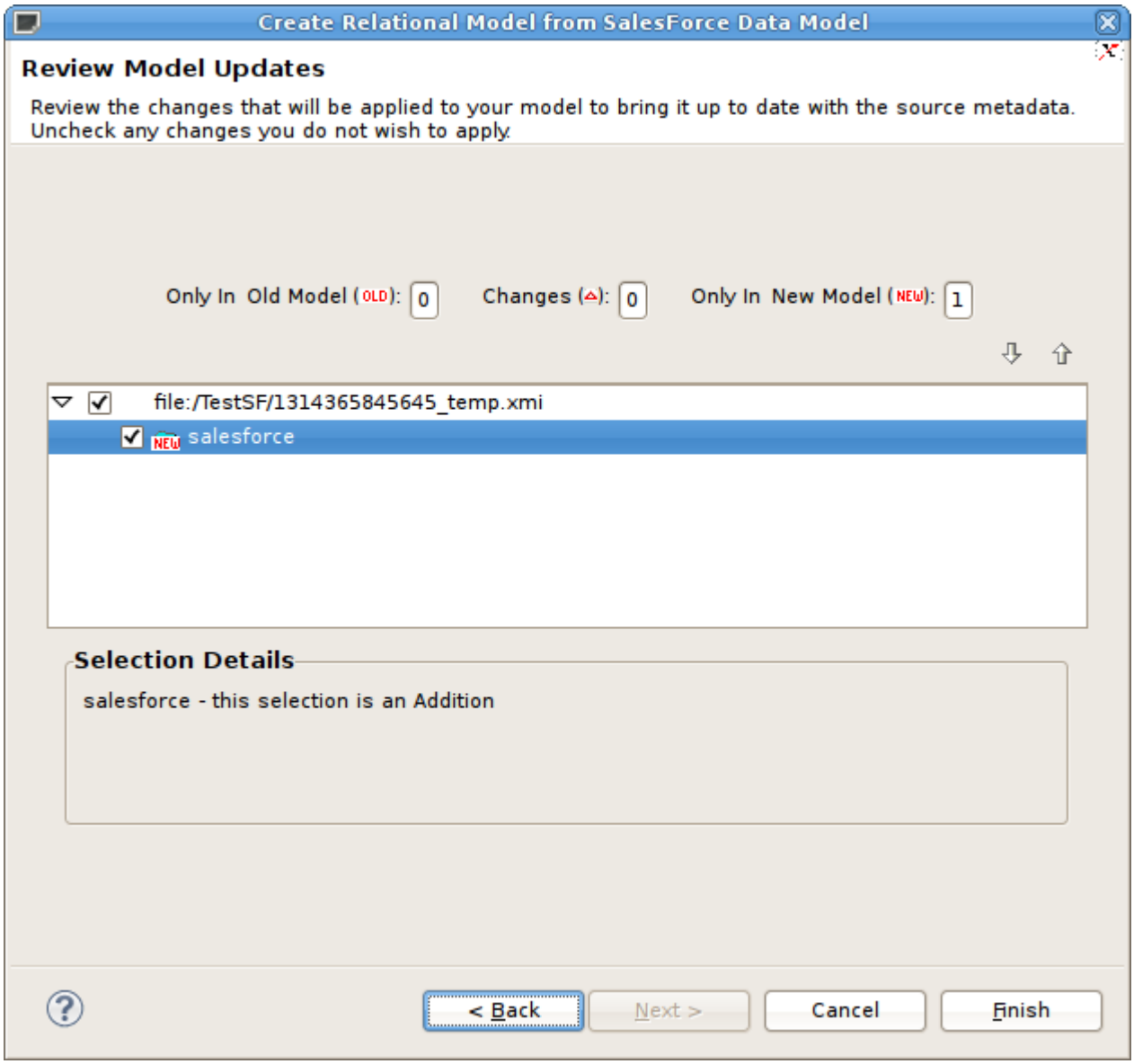


Figure 5.36. Review Model Updates Dialog

When finished, the new or changed relational model's package diagram will be displayed showing your new tables.

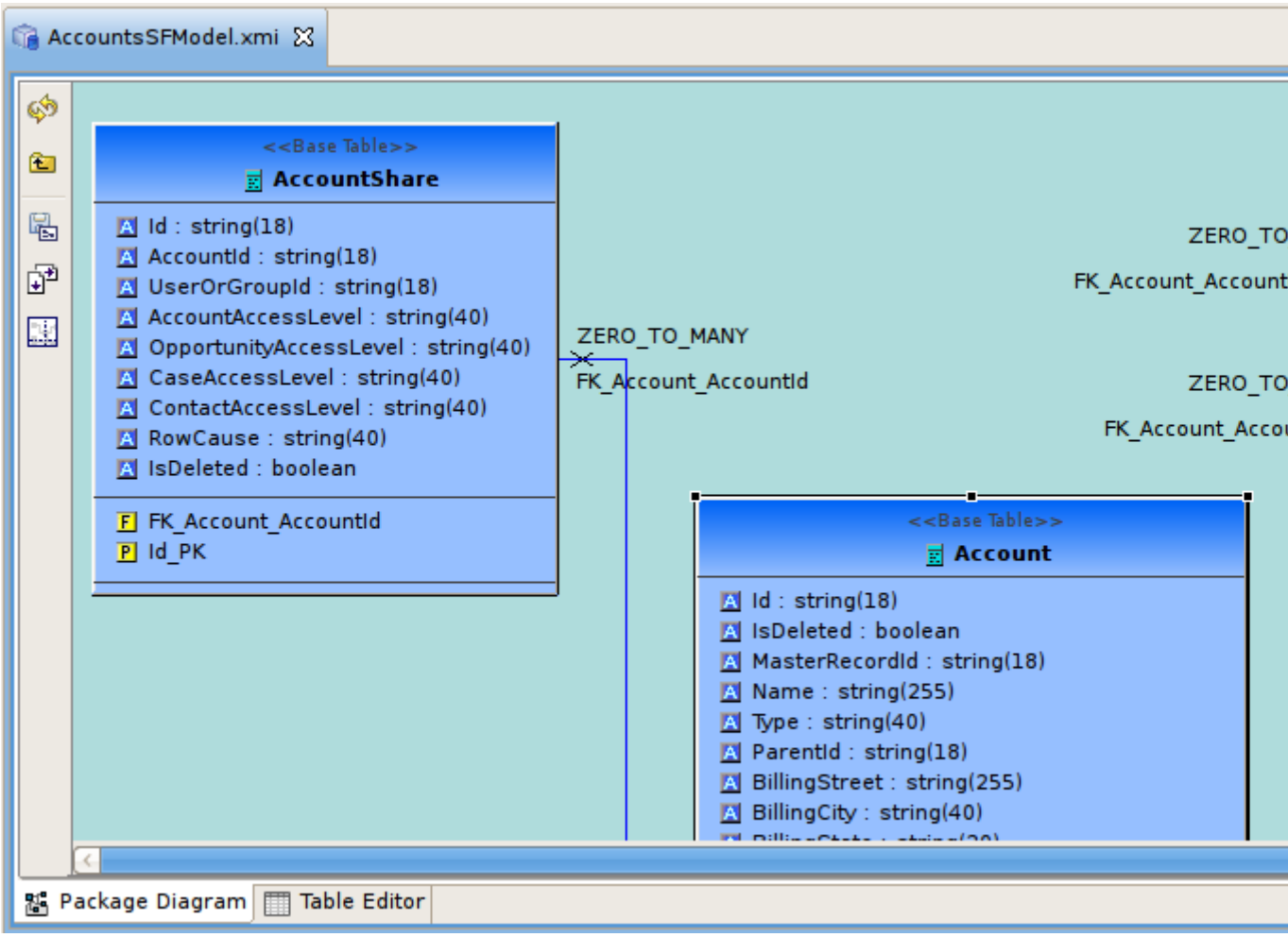


Figure 5.37. New Saleforce Tables Diagram

5.7. Import Metadata From Text File

- The Teiid Designer provides various import options for parsing comma delimited text file metadata into models. This is accomplished via the **Import > Teiid Designer > Designer Text File >> Source or View Models** 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 **Teiid Designer > Designer Text File >> Source or View Models** and click **Next>**
  - **Step 3** - Select an import type via the drop-down menu shown below.

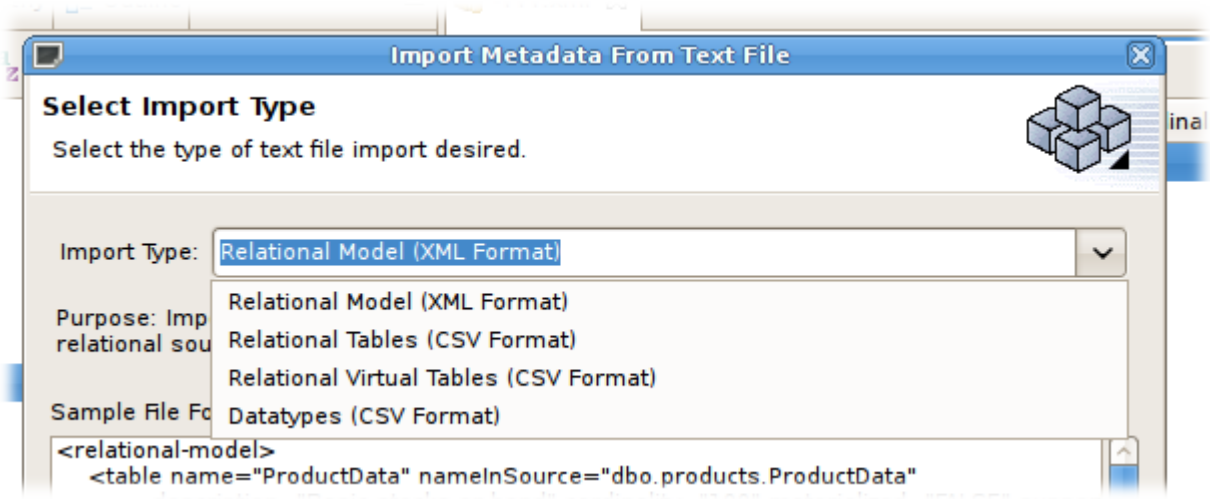


Figure 5.38. Import Wizard

- These steps are required for each type are defined below:
  - *Section 5.7.1, "Import Relational Model (XML Format)"*
  - *Section 5.7.2, "Import Relational Tables (CSV Format)"*
  - *Section 5.7.3, "Import Relational View Tables (CSV Format)"*
  - *???*

5.7.1. Import Relational Model (XML Format)

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

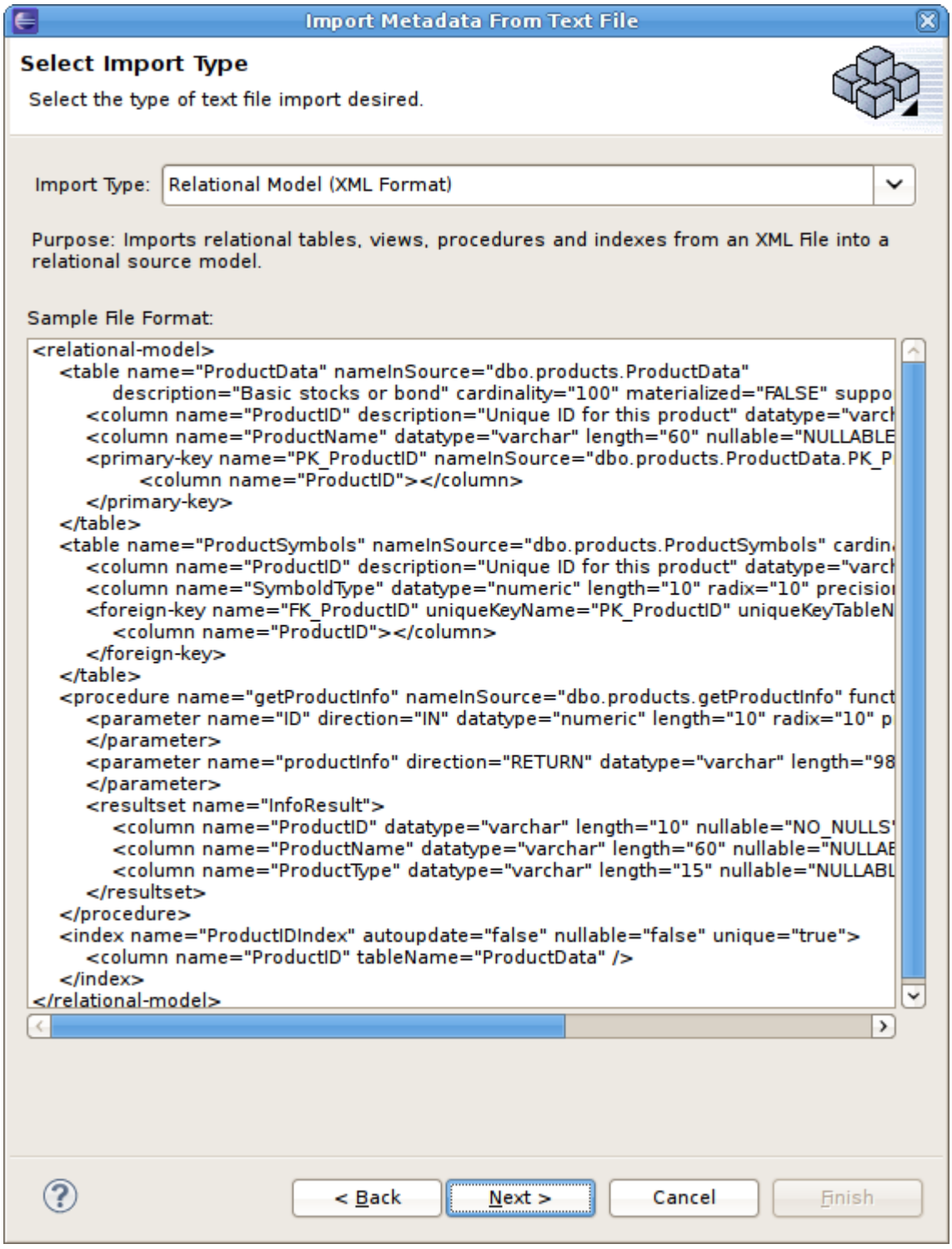


Figure 5.39. Select Import Type - Relational Model (XML Format)

- Perform **Steps 4** - On the next page, select the XML file on your local file system via the **Browse...** button. Select a target model to which the imported relational objects will be added

via the second **Browse...** button. The dialog allows selecting an existing relational model or creating a new model.

Note

The contents of your selected XML file will be display in the File Contents viewer.

Click **Finish** to create your new model.

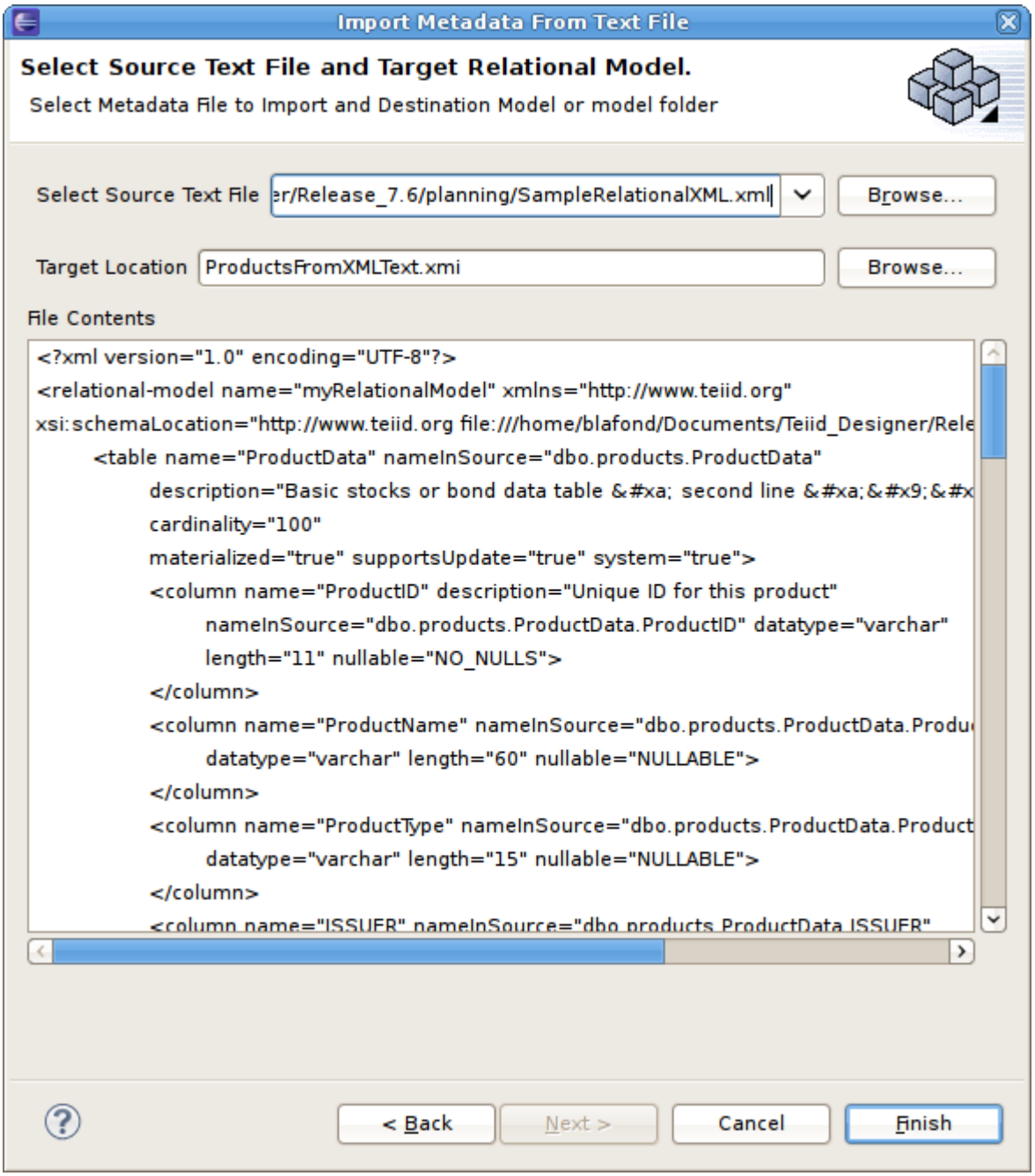


Figure 5.40. Select Source Text File and Target Relational Model Page

If the target model contains named children (tables, views, procedures) that conflict with the objects being imported, a dialog will be displayed giving you options on how to proceed including: replacing specific existing objects, creating new same-named objects or cancel import entirely.

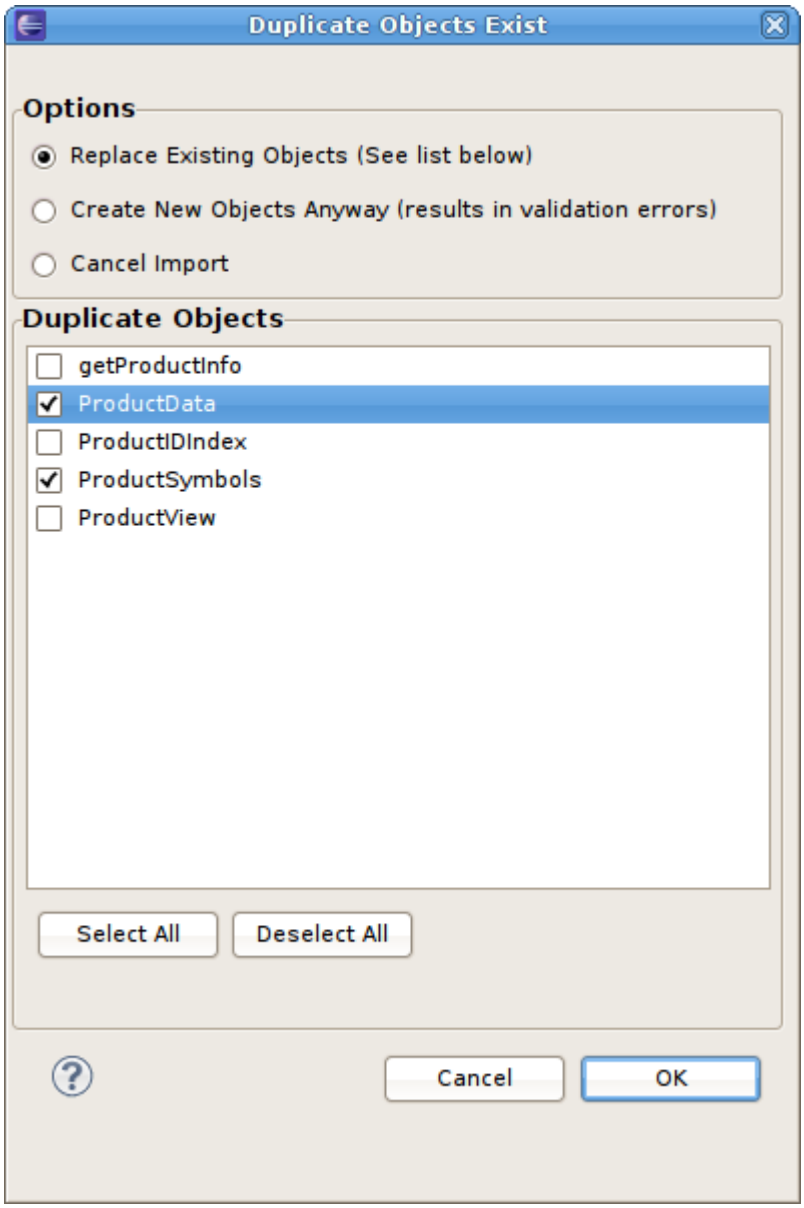


Figure 5.41. Duplicate Objects Dialog

5.7.2. Import Relational Tables (CSV Format)

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

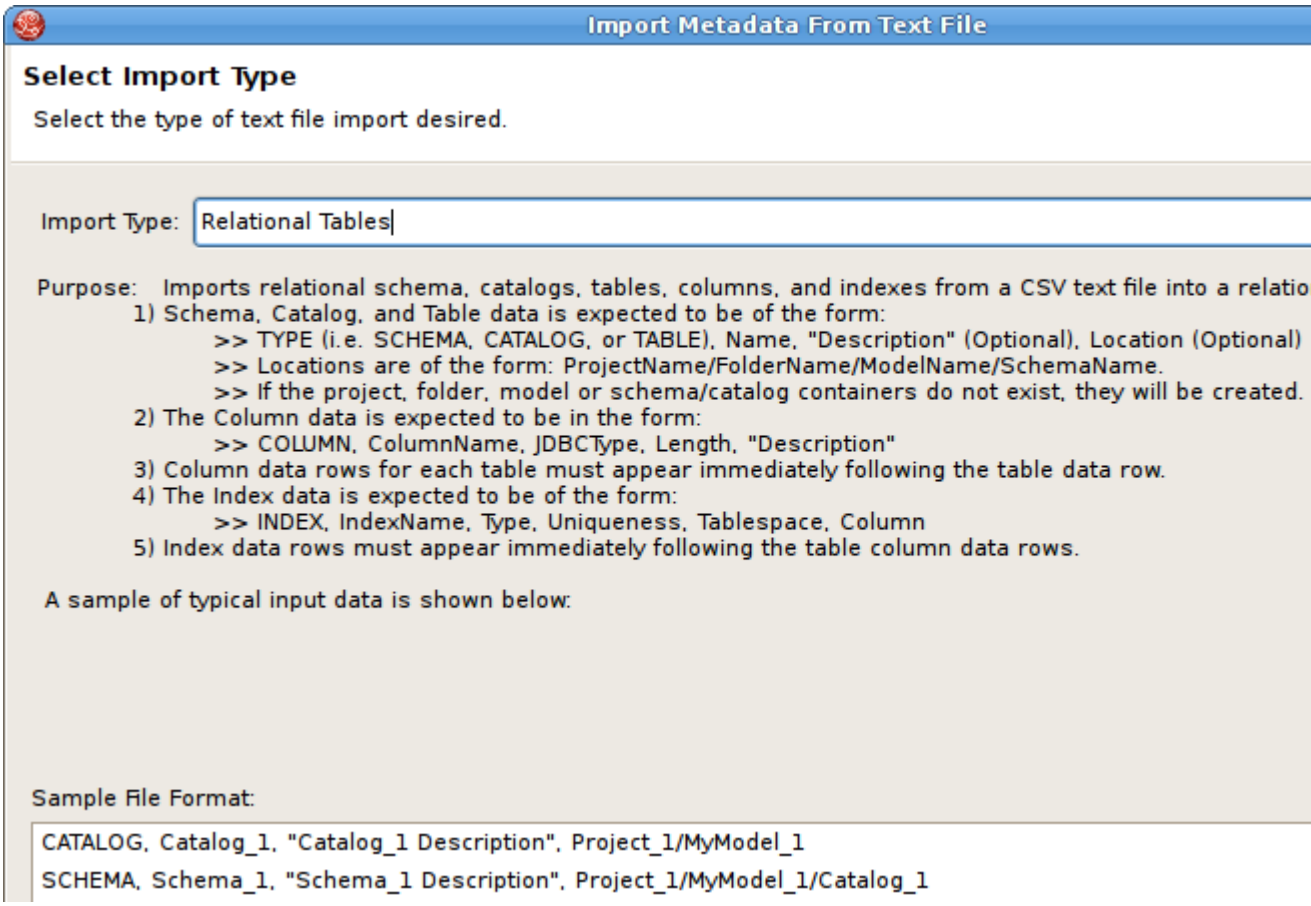


Figure 5.42. Select Import Type - Relational Tables (CSV Format)

- **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.

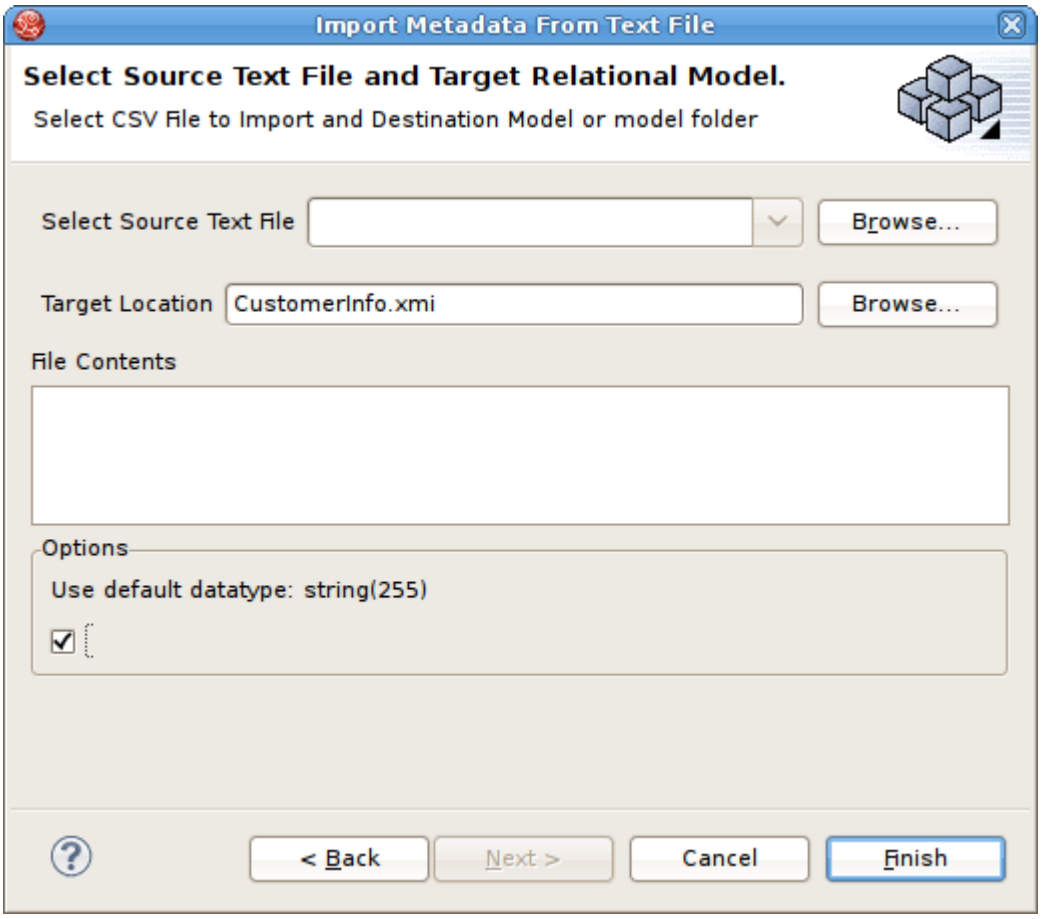


Figure 5.43. Select Source Text File and Target Relational Model

- **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**.

5.7.3. Import Relational View Tables (CSV Format)

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

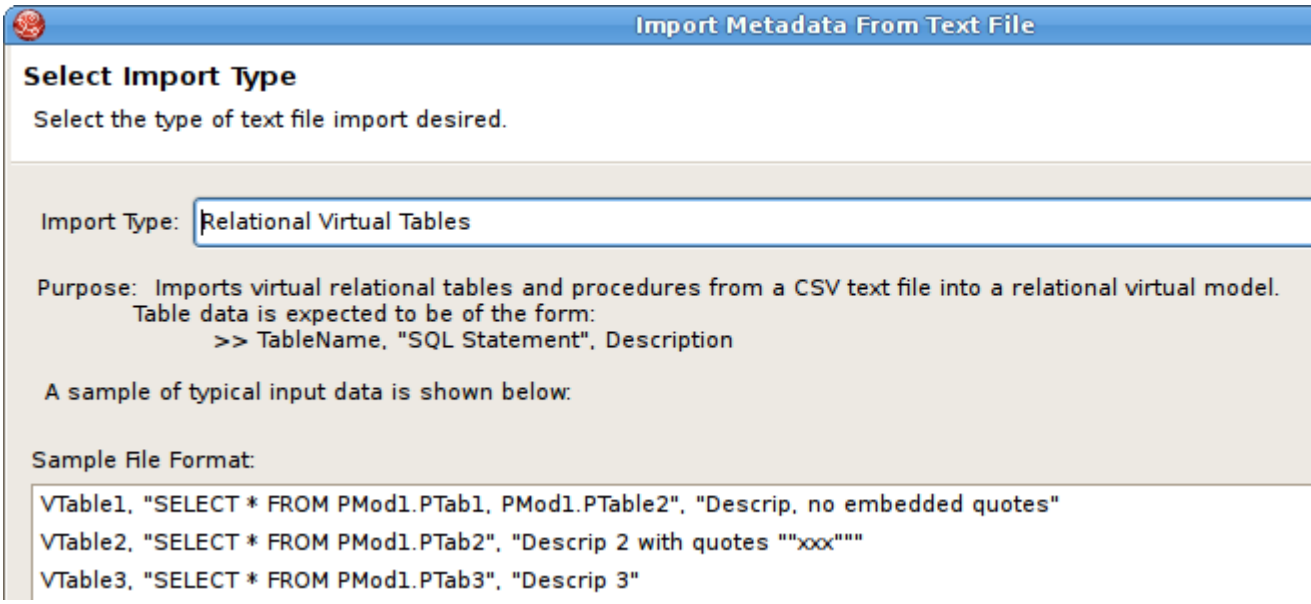


Figure 5.44. Select Import Type - Relational Virtual Tables (CSV Format)

- **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.

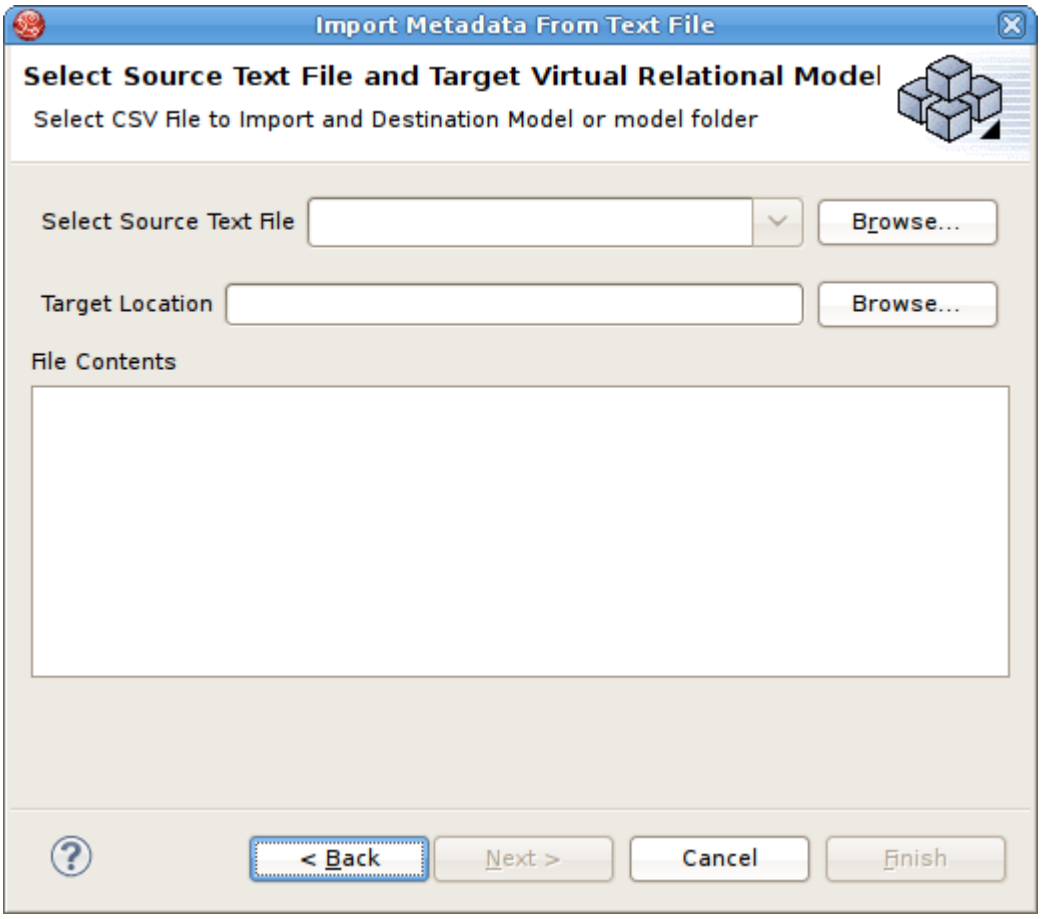


Figure 5.45. Select Source Text File and Target Virtual Relational Model

- **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**.

5.8. Import Data From REST Service

- In addition to SOAP web services, Teiid supports REST web services as data sources. Using the steps below you will define your REST web service data source, select the elements to include in your generated view table and generate a source model containing the required Teiid procedure.

  To import from your target REST web service source:

  - **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 File or URL >> Source and View Model (SOAP)** and click **Next>**
- **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.

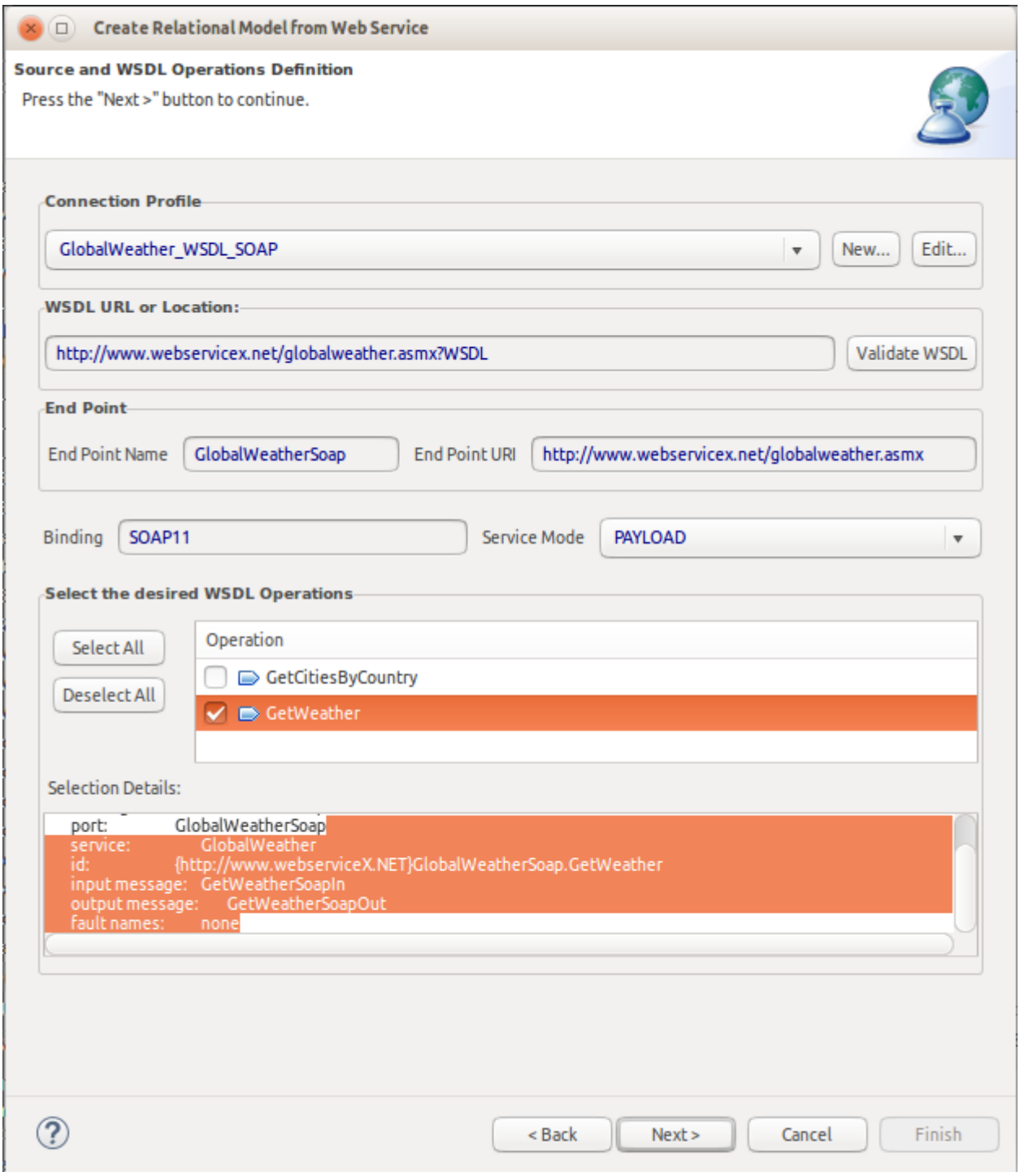


Figure 5.46. WSDL Source Selection

- **Step 4** - Select 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 panel displays static information about the operation such as the names of the input and output messages, and faults thrown by the operation.

  Click **Next >**

- **Step 5** - The next page entitled Model Definiton requires both a model location (i.e. folder or project) and a valid model name for both source and view models. Use the **Browse...** button to select existing folders or models. Click **Next>** when all the information is defined.

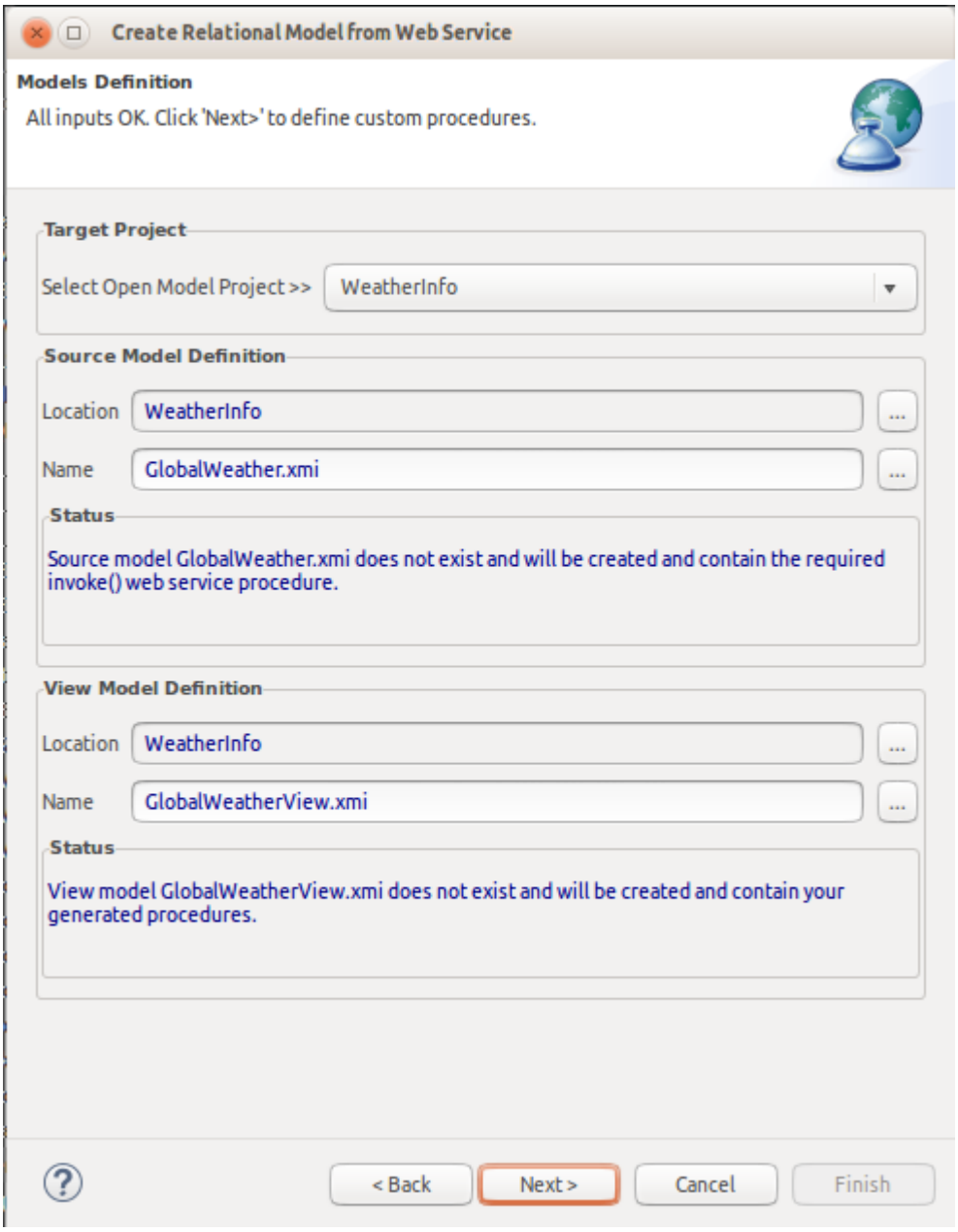


Figure 5.47. WSDL Source Selection

- **Step 6** - The next page, titled **Data Source Definition Options**, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model will prepresent. This name will be stored as a property on your model and changeable via the **Modeling > Set JBoss JNDI Data Source Name** action. If there you have a default server defined and running, then **Auto-Create Data Source** check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

  When finished with this page, click **Next>**.

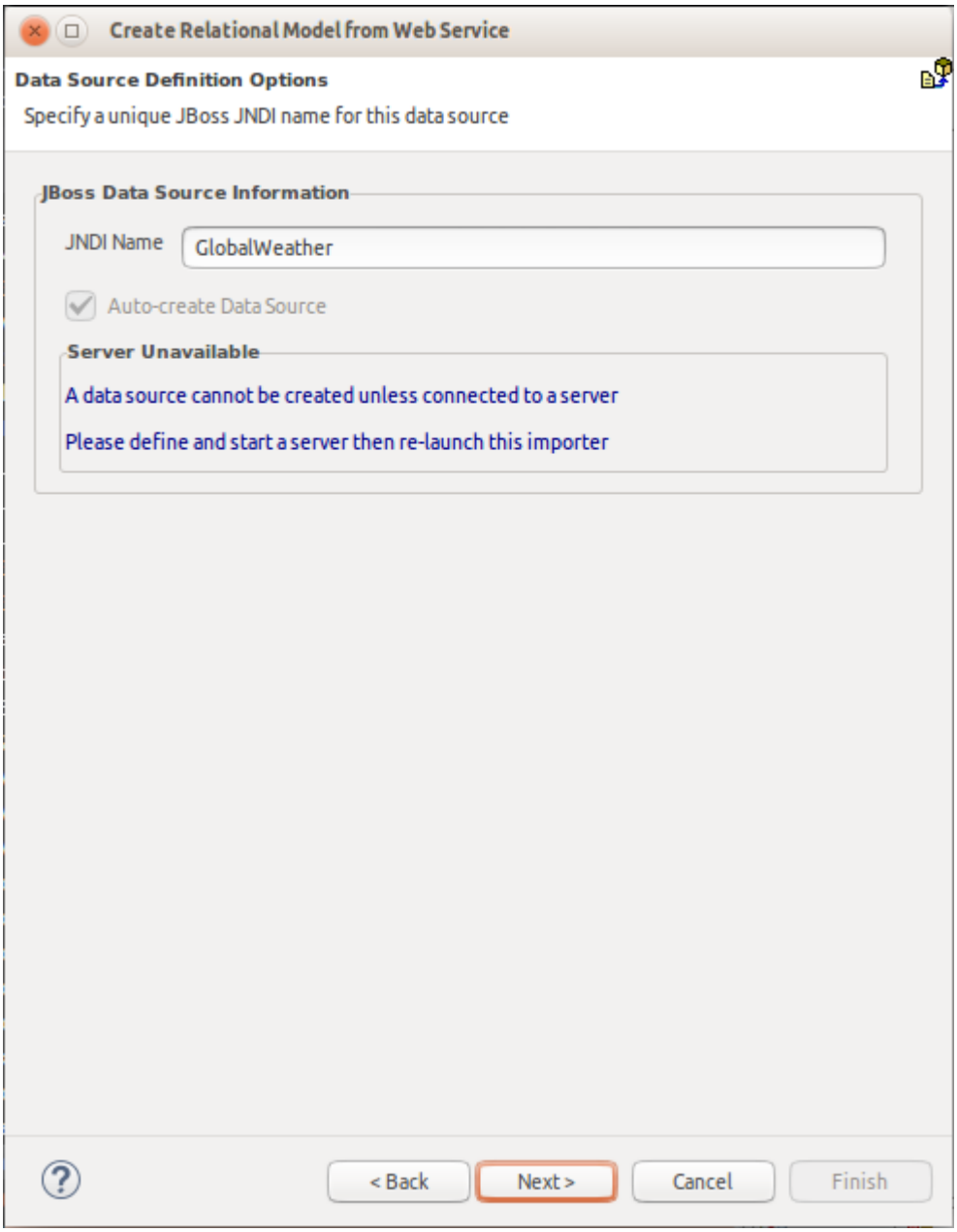


Figure 5.48. Data File Source Selection Page

- **Step 7** - The next page entitled Operations Type Selection gives you the option to define custom procedures (Recommended)shown in **Step 8 below**. or generate default procedures you can edit later. button to select existing folders or models. If customer procedure is selected, click **Next>** to proceed or **Finish** if default option is selected

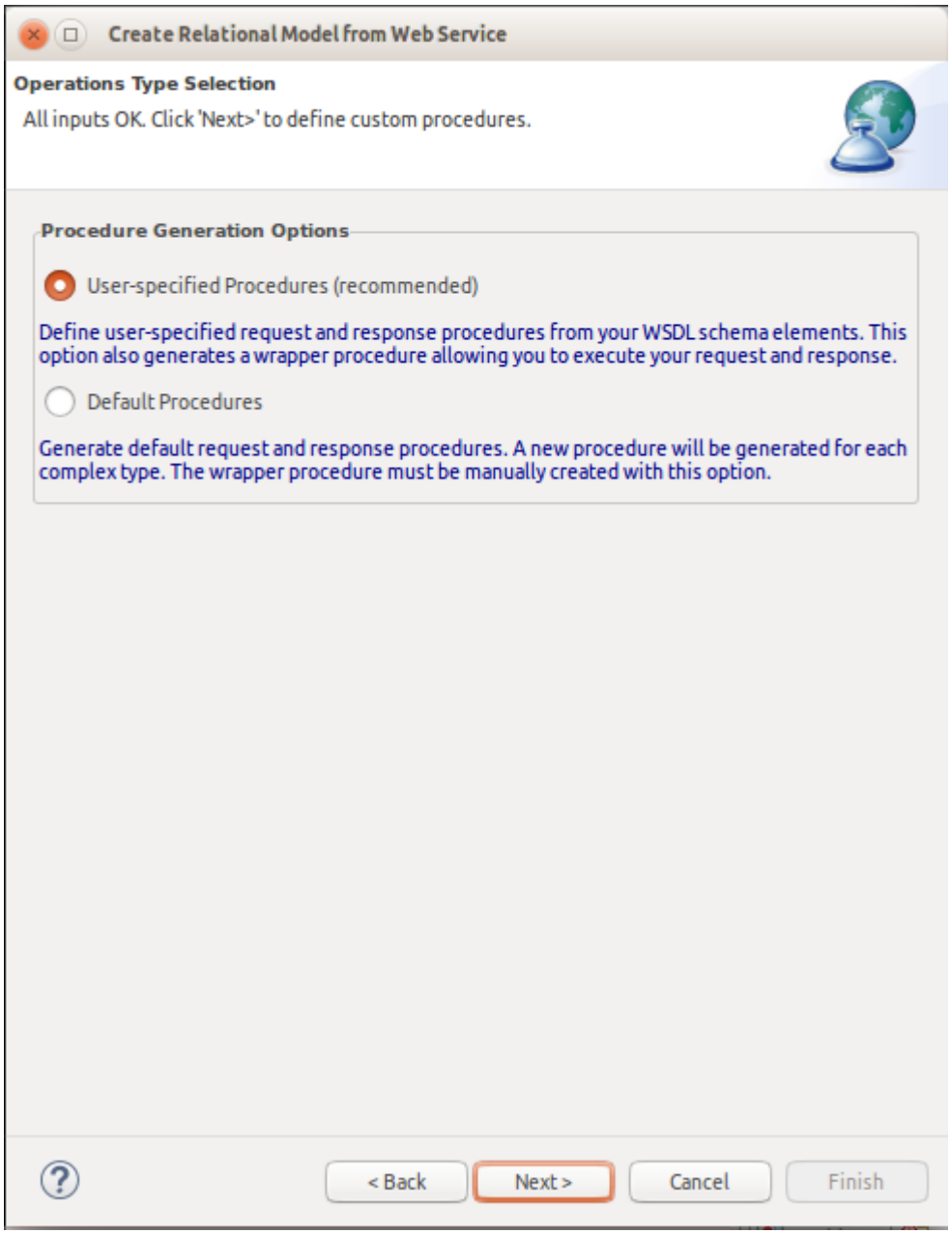


Figure 5.49. WSDL Source Selection

- **Step 8** - This wizard generates both request and response procedures that are used in the queryable wrapped procedure. The next page, Procedure Definition, provides the means to define the details of your request and response structures.

In the Request tab, select and double-click the schema elements you wish to be input parameters for your request. These will be added to the Element Info panel and the resulting generated SQL statement will be updated to reflect the new element.

Note

The BODY and HEADER tabs which exist on both the Request and Response tabs. If the selected service mode for this procedure is set to MESSAGE, the HEADER tab will be enabled and allow you to define the SOAP header variables utilizing the same schema tree.

Select the Response tab and create the response procedures result set columns in the same way.

Repeat this process for all operations by changing the selection target operation via the **Operations** selector at the top.

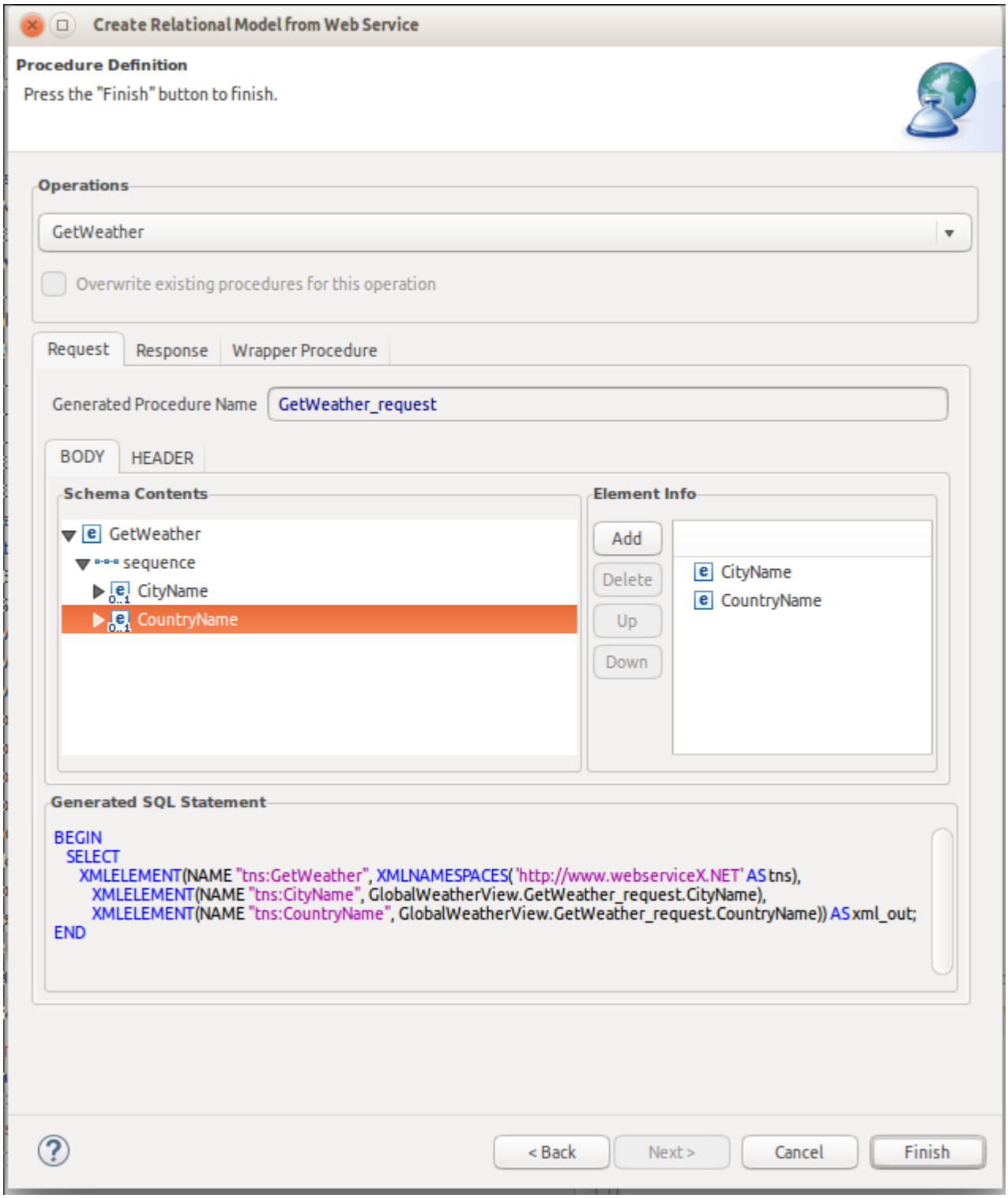


Figure 5.50. Procedure Definition Page

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

5.9. Import Data From SOAP Service

- In addition to REST web services, Teiid supports SOAP web services as data sources. This importer is accessed by launching Eclipse's "Import..." action and selecting the "Teiid Designer > WSDL File or URL >> Source and View Model (SOAP)" option. *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 containg procedures to build and parse the XML declared as the service messages.

  Using the steps below you will define your SOAP web service data source, select the elements to include in your generated view table and generate a source model containing the required Teiid procedure.

  To import from your target SOAP web service source:

  - **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 > Web Service Source >> Source and View Model (SOAP)** and click **Next>**

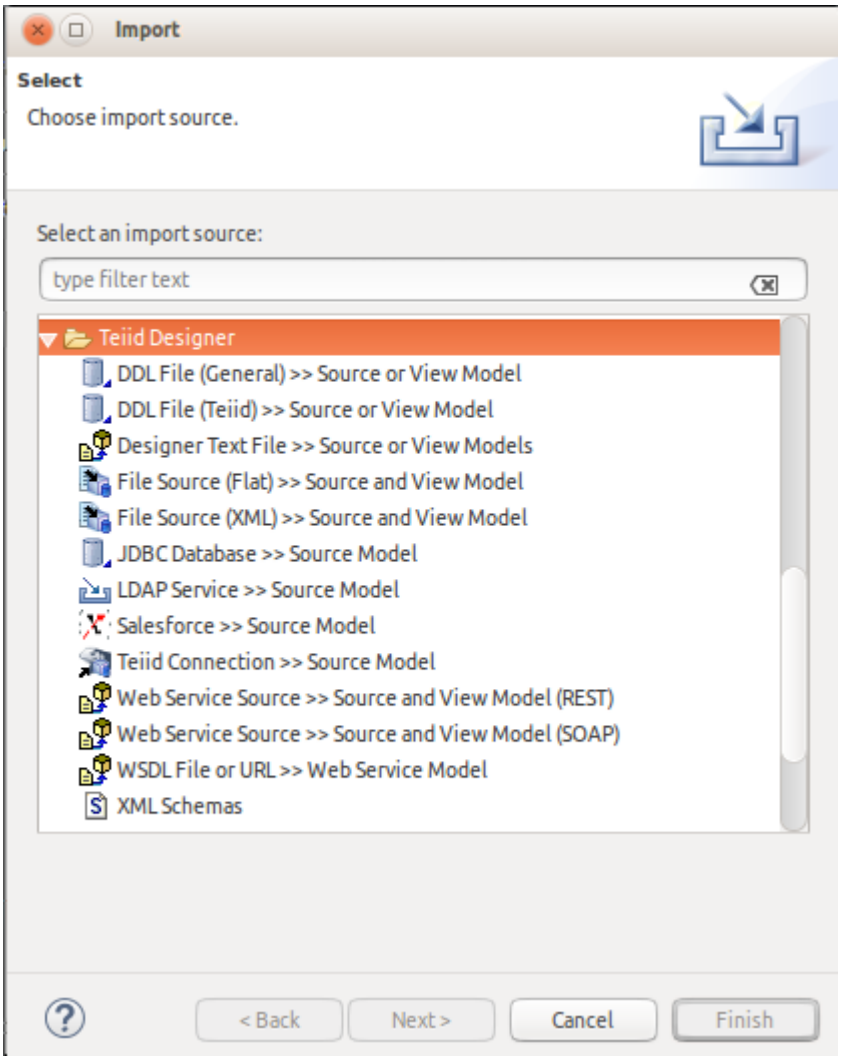


Figure 5.51. Import from SOAP Web Service Source

- **Step 3** - On the first page of the wizard, select an existing or previous connection profile from the drop-down selector or press **New...** button to launch the **New Connection Profile** dialog or **Edit...** to modify/change an existing connection profile prior to selection.

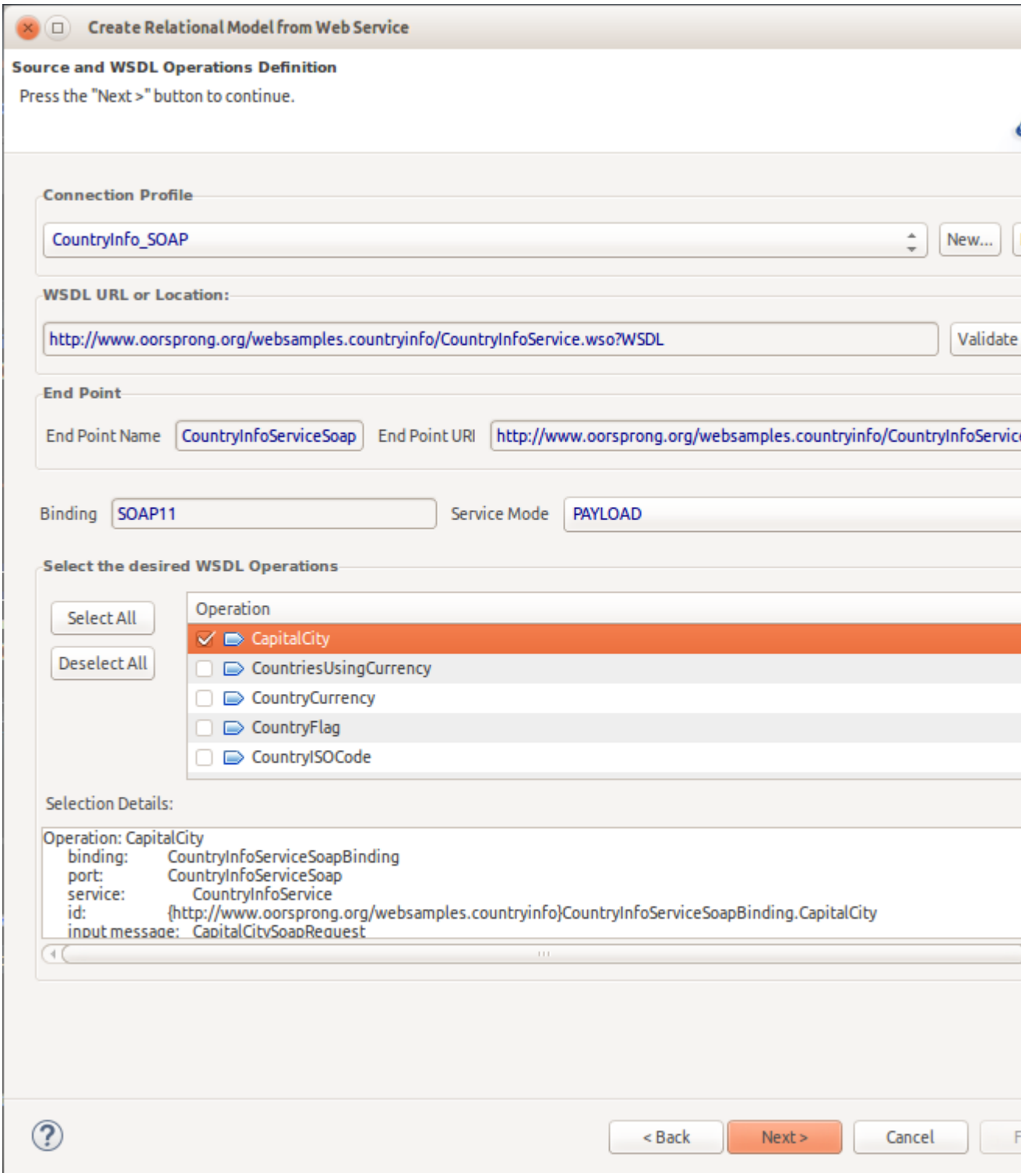


Figure 5.52. Source and WSDL Operations Definition

Name your new SOAP Connection Profile and click the **Next >**

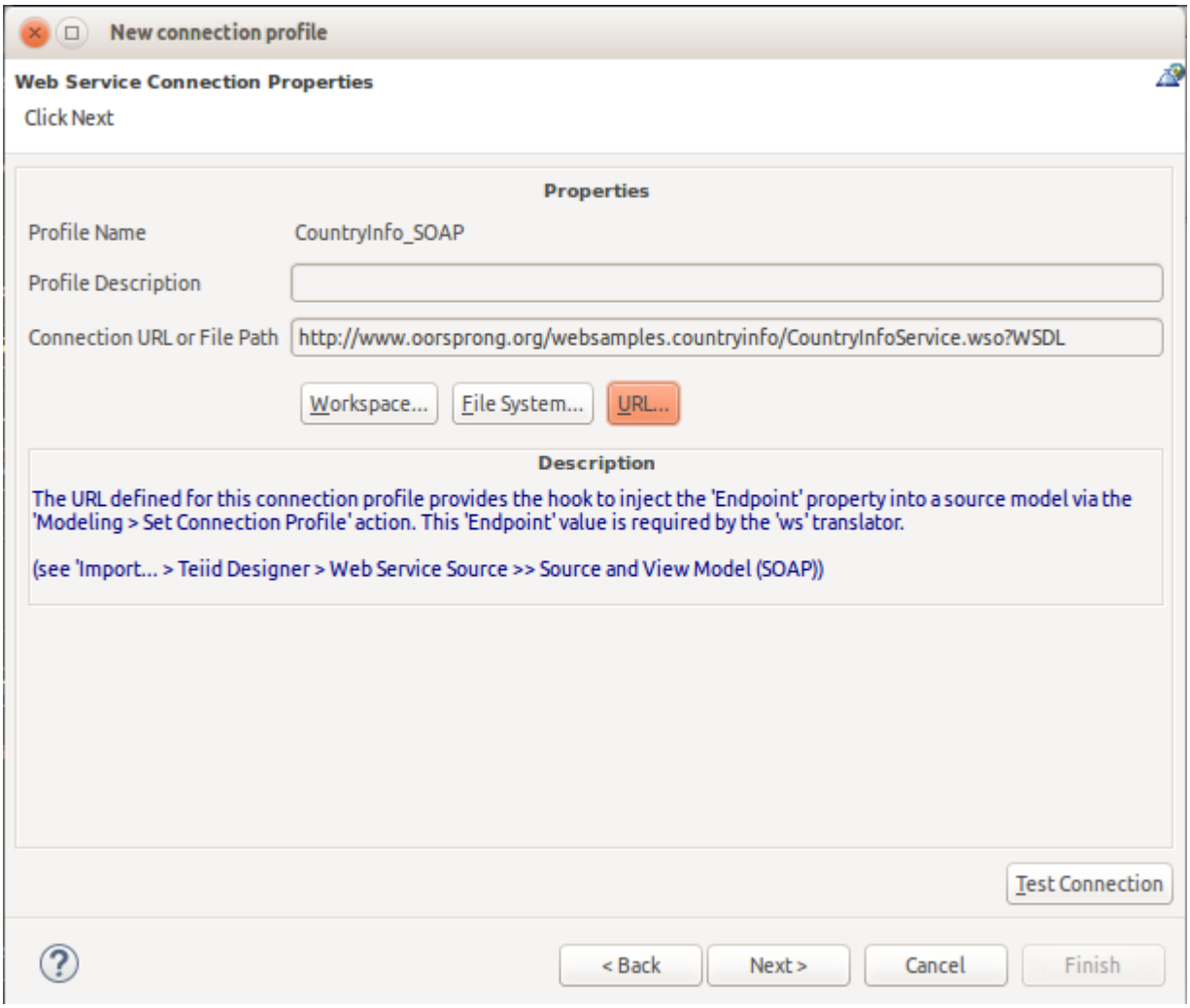


Figure 5.53. New SOAP Connection Profile

Set your Connection URL from either your workspace, your local file system or define a remote URL. Click **Next >** and select an endpoint and binding type, click **Finish**.

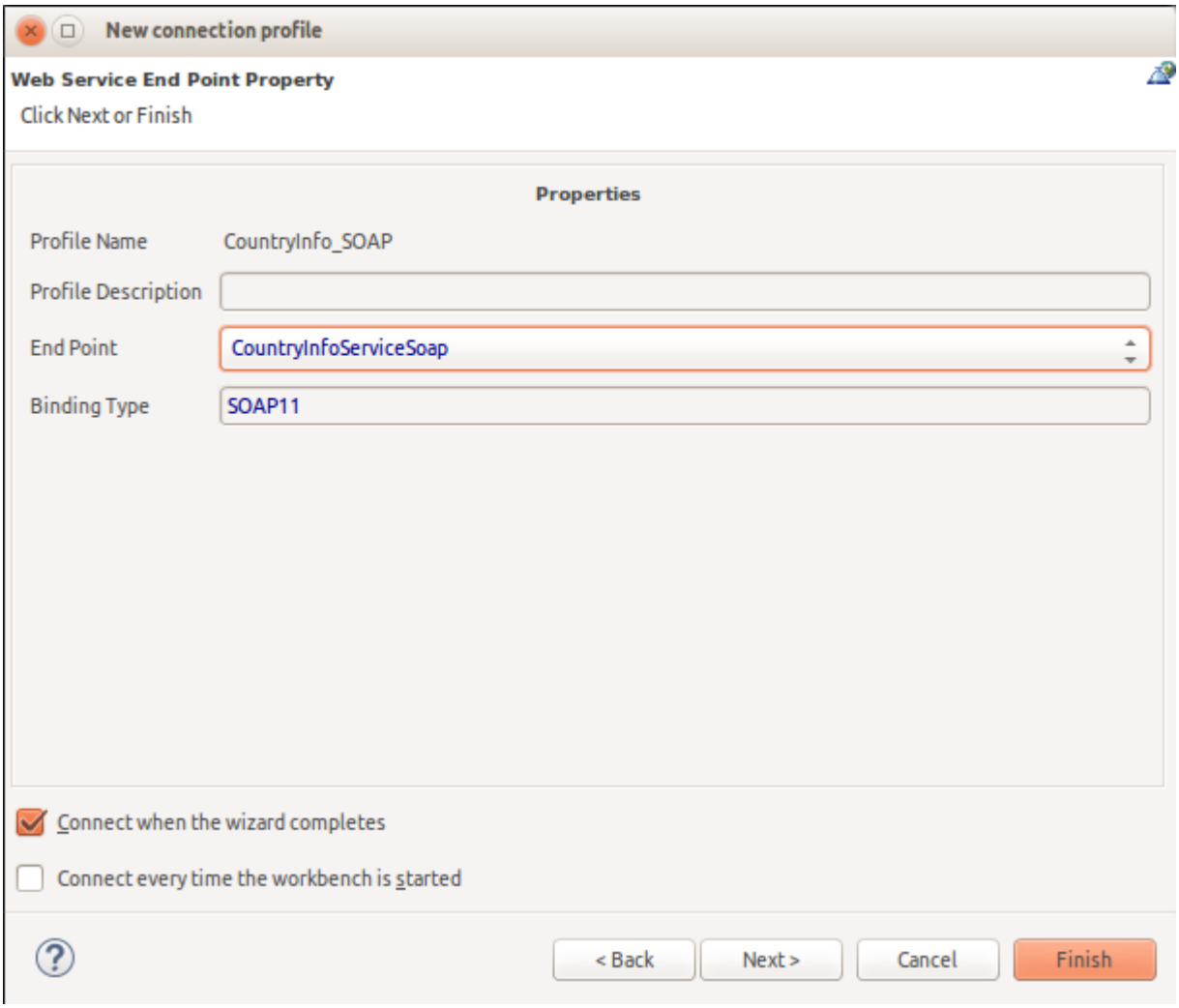


Figure 5.54. New SOAP Connection Profile Endpoint Definition

Once the connection profile is defined, you can view the WSDL URL and validate it via the **Validate WSDL** button. Select Service Model and desired WSDL Operations and click **Next >**

- **Step 4** - On the **Models Definition** page, define your source and view model names and locations and click **Next >**

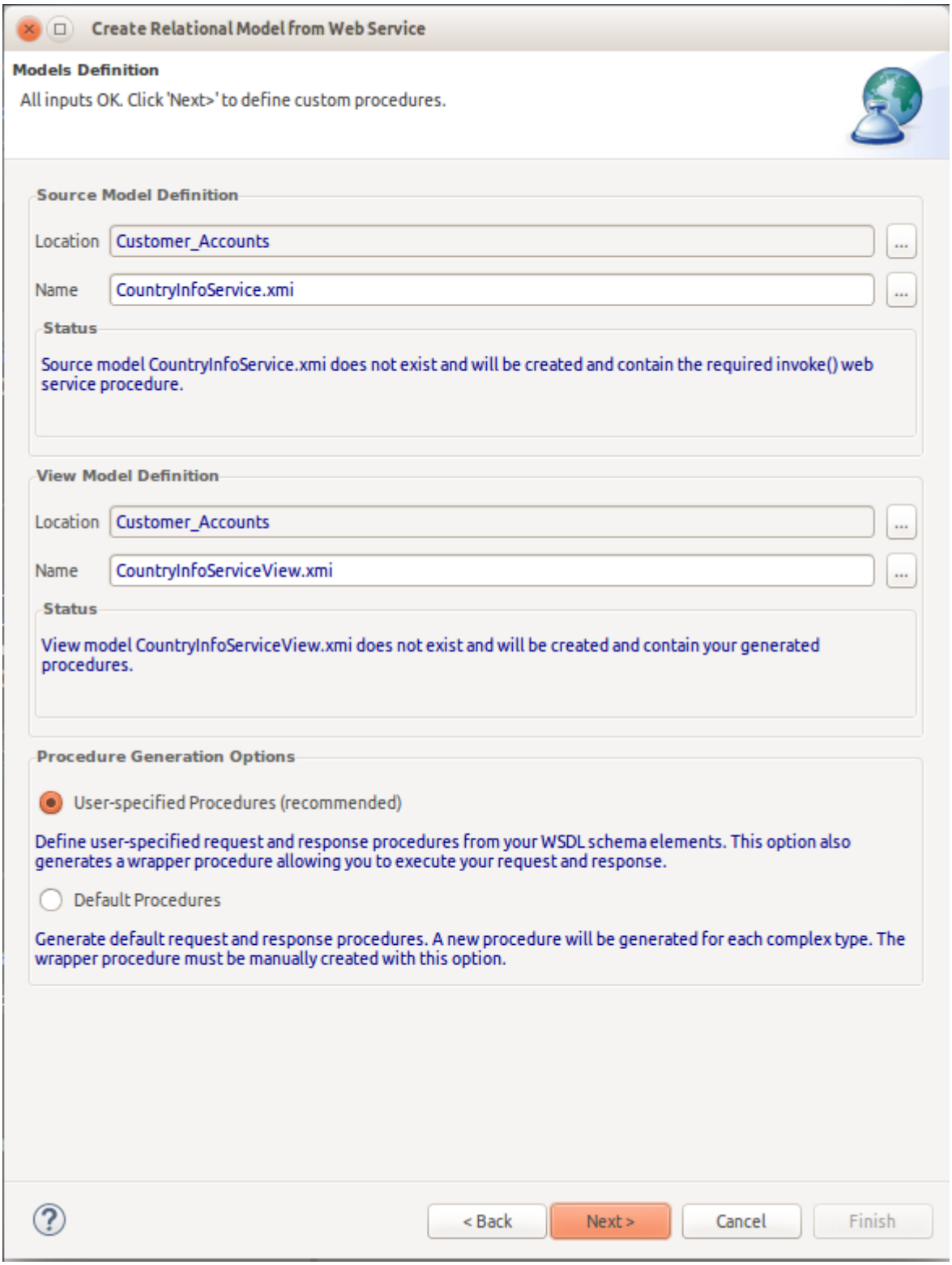


Figure 5.55. Models Definition

- **Step 5** - On the **Procedure Definition** page, defined your request and response elements for each operation selected previously. When all operations are sufficiently defined click **Finish**

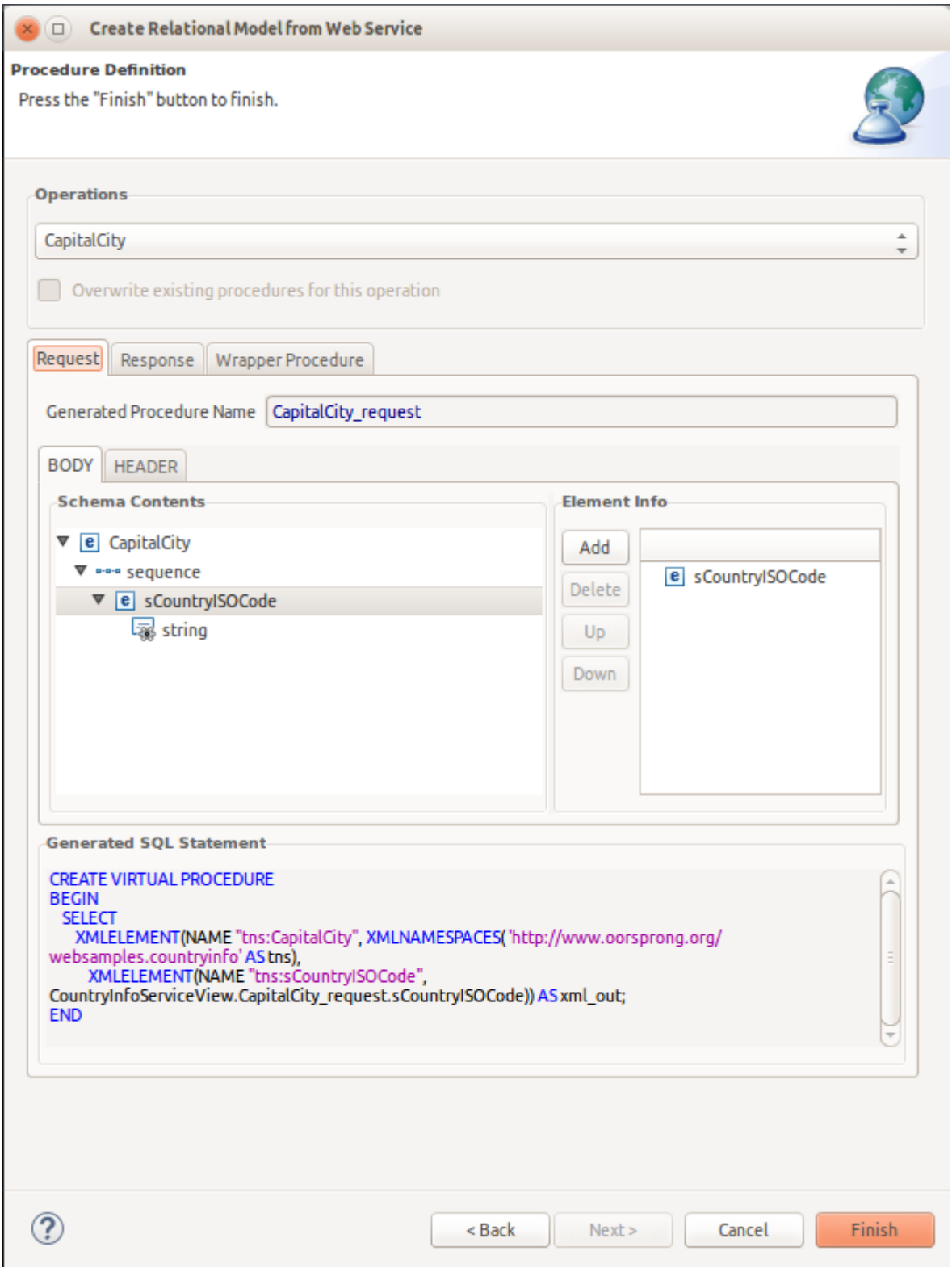


Figure 5.56. Procedure Definition

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.

A single **view model** was also created containing your new procedures named after the operations declared in the WSDL. For each operation a wrapper procedure was created which can be previewed in Designer. Below is an example dependency diagram showing the sources for the wrapper procedure as request, response procedures and the invoke() source procedure.

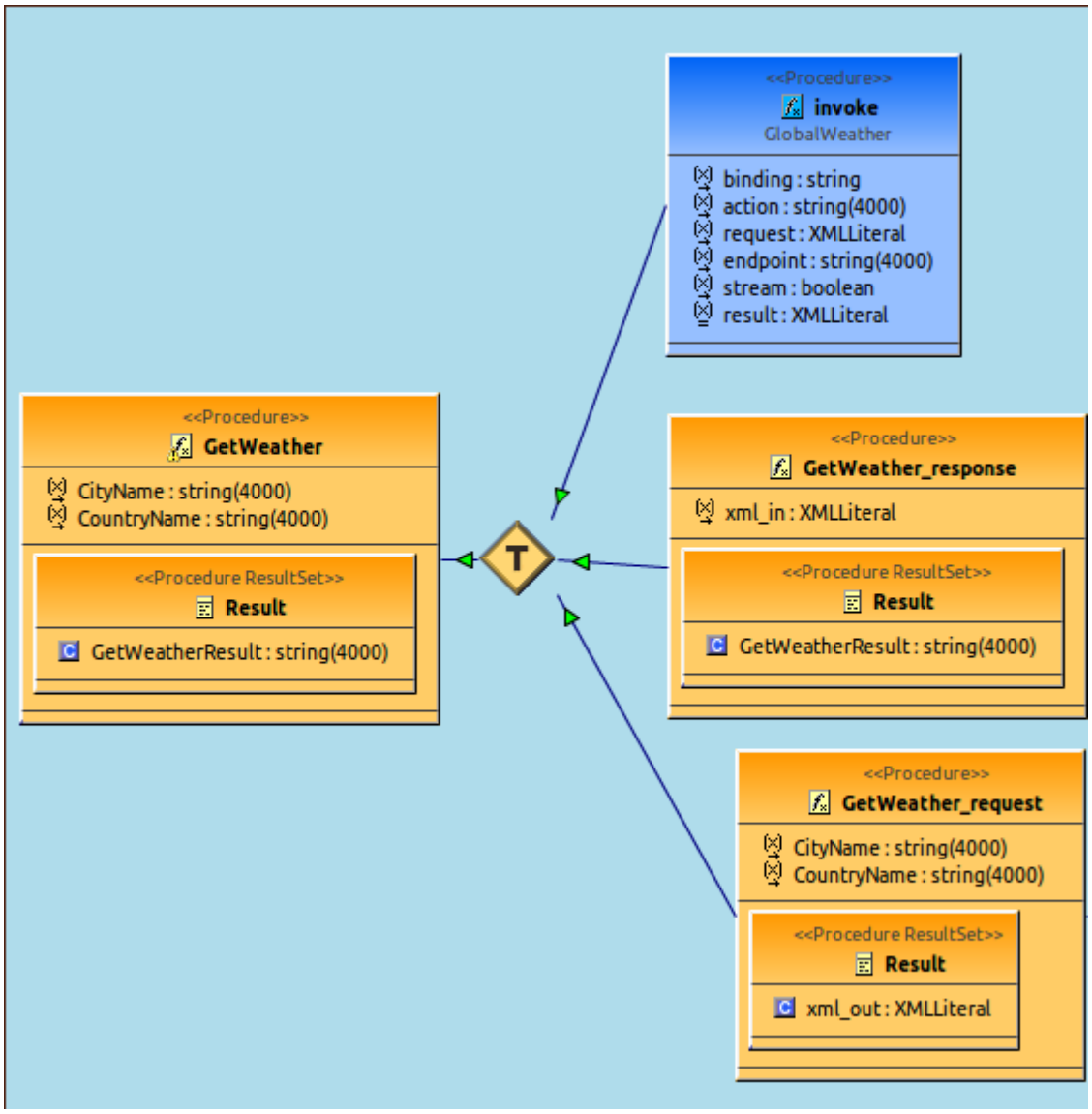


Figure 5.57. Example Web Services Wrapper Procedure

5.9.1. Circular References in WSDL Schemas

It is possible for a WSDL schema to either contain a very deep set of XML type references or indeed for such references to be circular. This is legal in the WSDL schema but can make

processing the schema in Designer difficult. If left unchecked such circular references can result in a JVM **StackOverFlow** exception and exiting of the application.

To mitigate this possibility a depth limit of 750 references has been introduced. Should the depth exceed this limit then a warning is displayed and further processing of that fragment of the schema will end. It may be the case that the reference in question is not circular but just very deep so in such a case it is possible to increase the depth limit by setting the JVM property **WsdlSchemaHandlerRecursiveDepth** to a larger value, eg. -D WsdlSchemaHandlerRecursiveDepth=800. This should only be used with caution as on some systems it is possible the JVM throws a **StackOverFlow** exception before the new depth limit is reached.

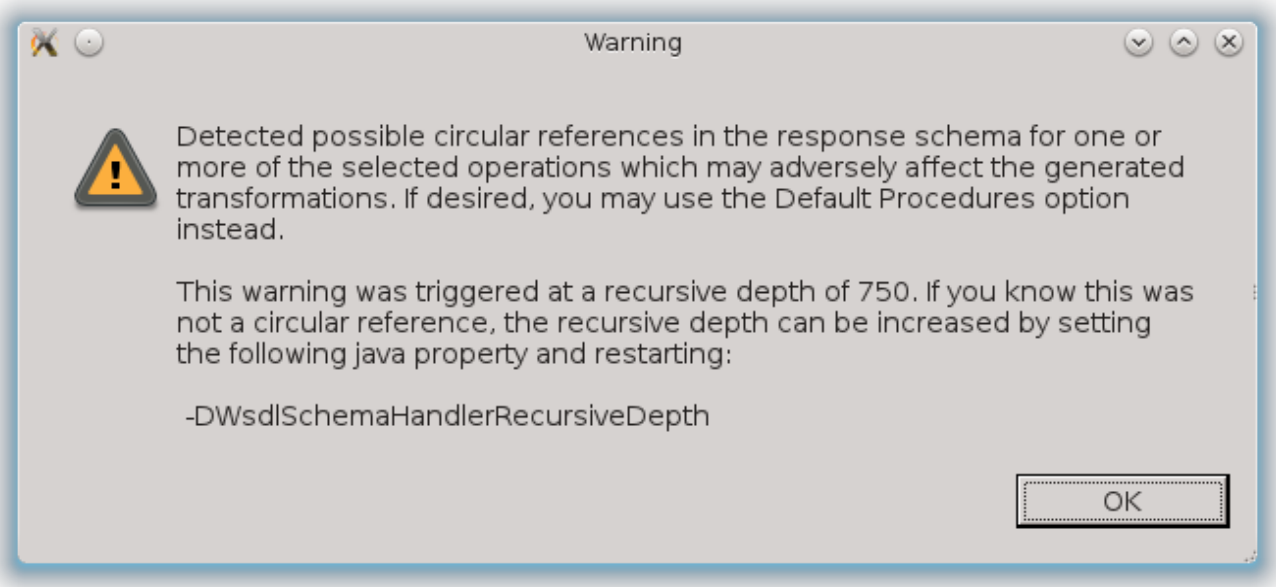


Figure 5.58. Warning message displayed if depth limit has been reached

5.10. 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.

5.10.1. 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 **Teiid Designer > WSDL File or URL >> Web Service Model** option shown below and click **Next>**
  - **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.

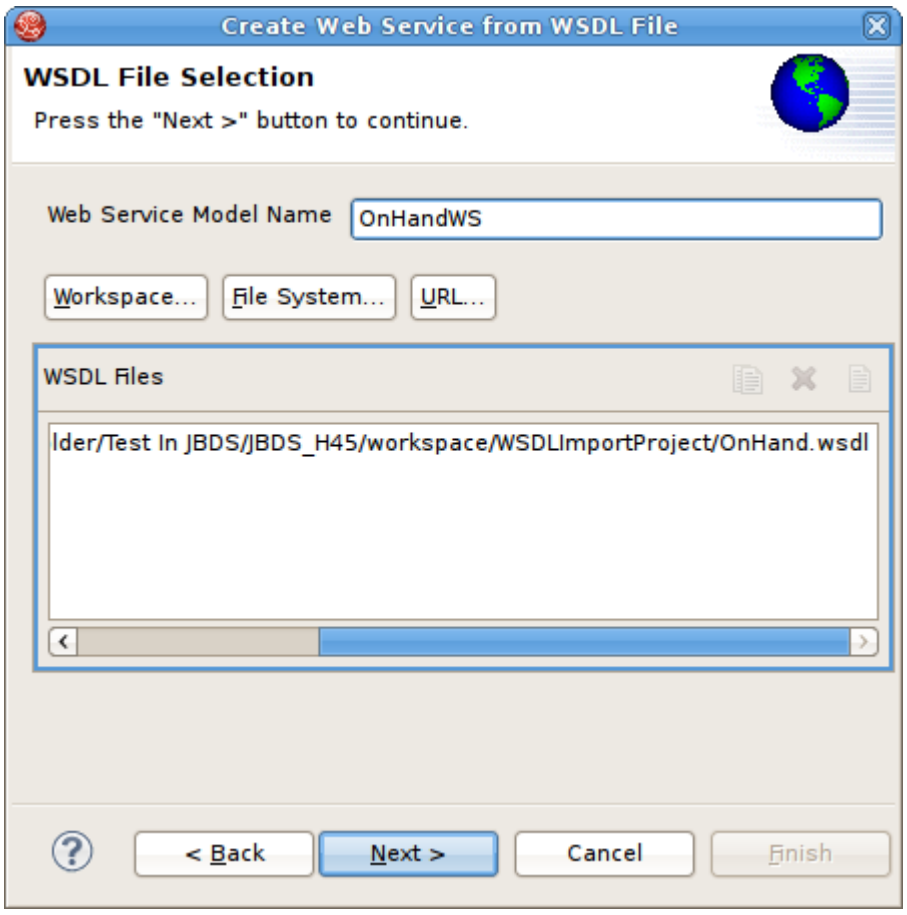


Figure 5.59. WSDL File Selection Dialog

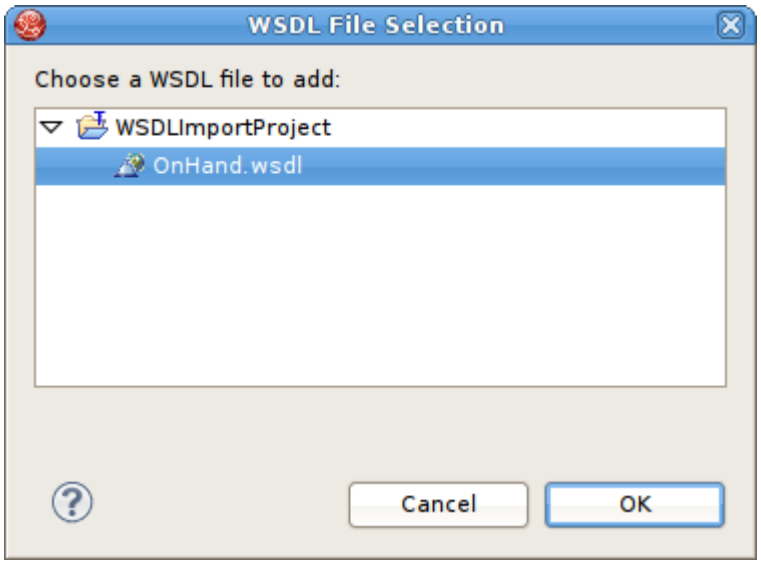


Figure 5.60. WSDL File Workspace Selection Dialog

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 (**WSDL**s 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.

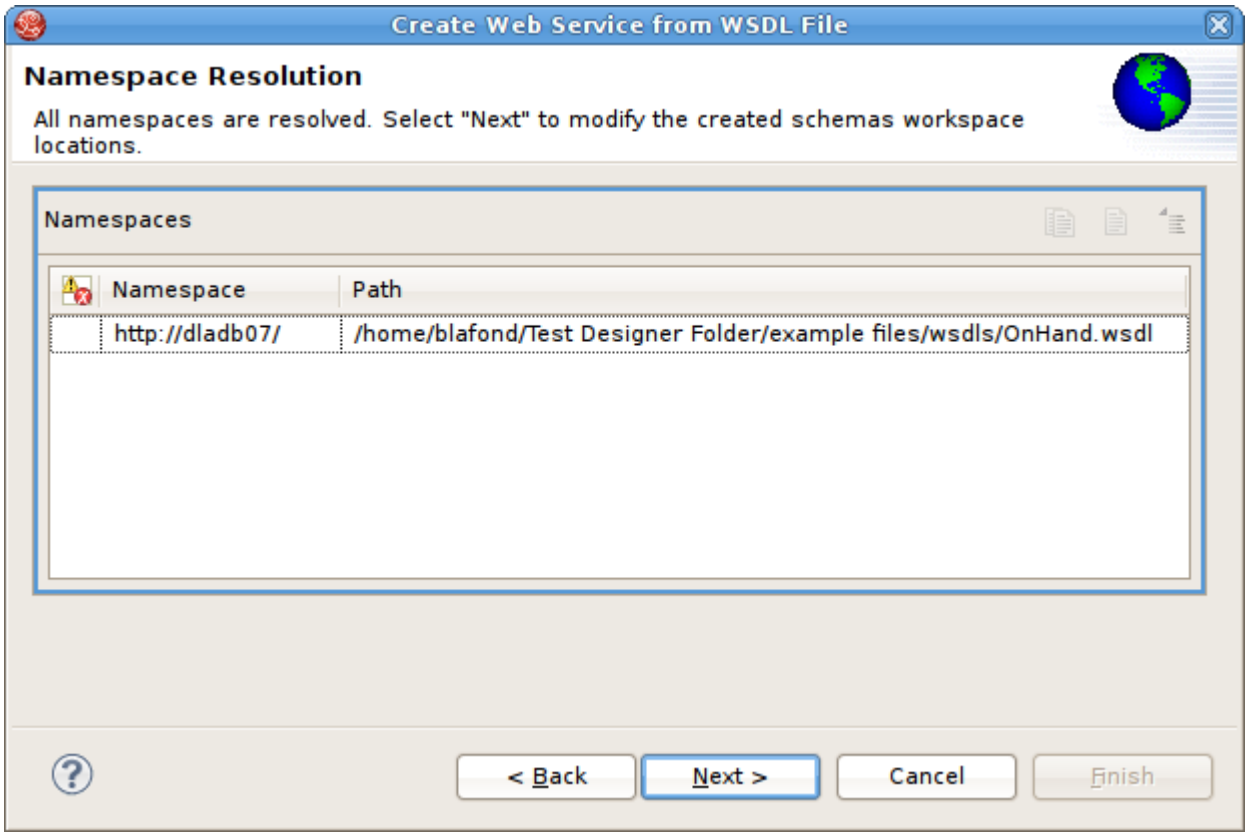


Figure 5.61. Namespace Resolution Dialog

- **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.

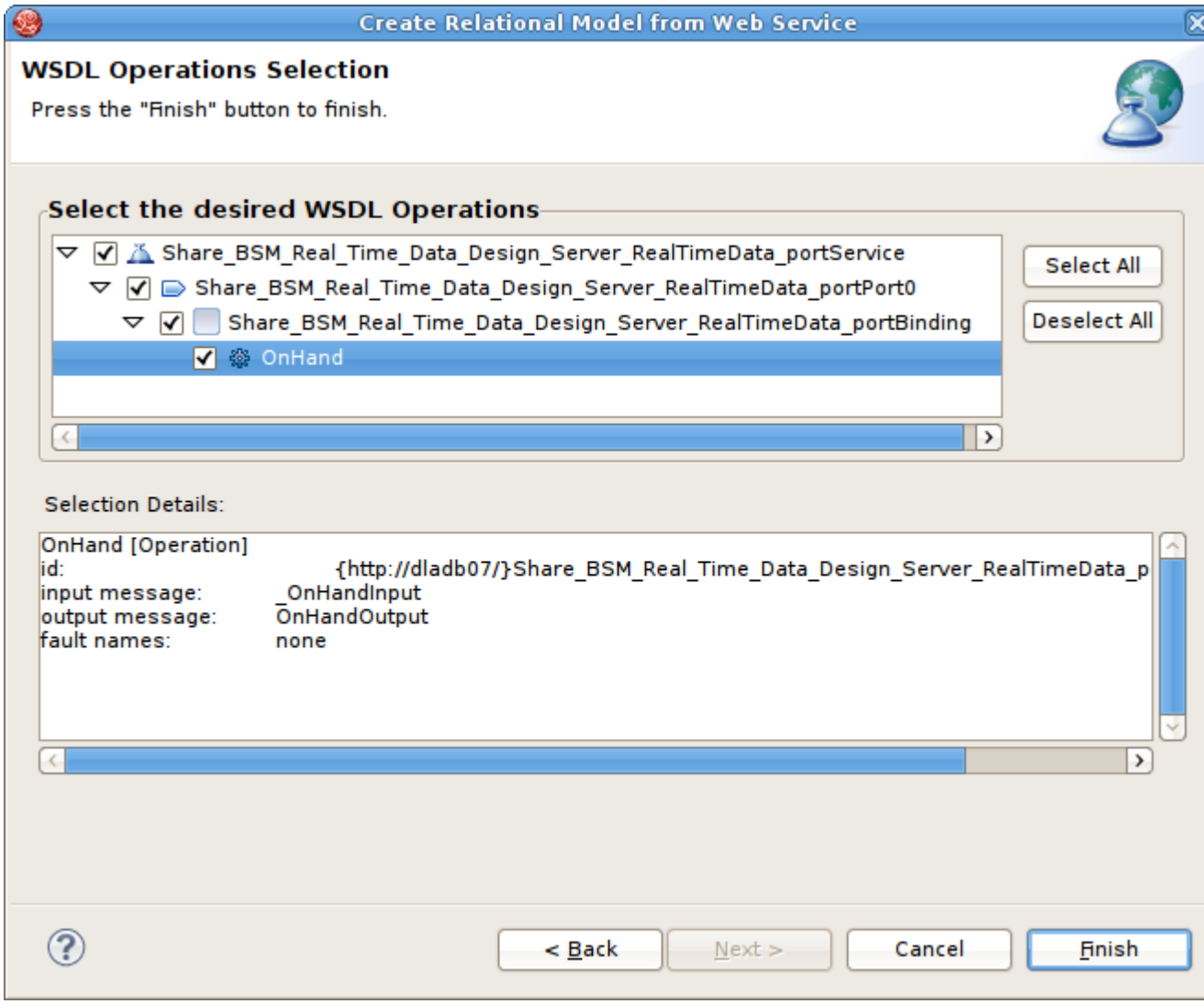


Figure 5.62. Namespace Resolution Dialog

- **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

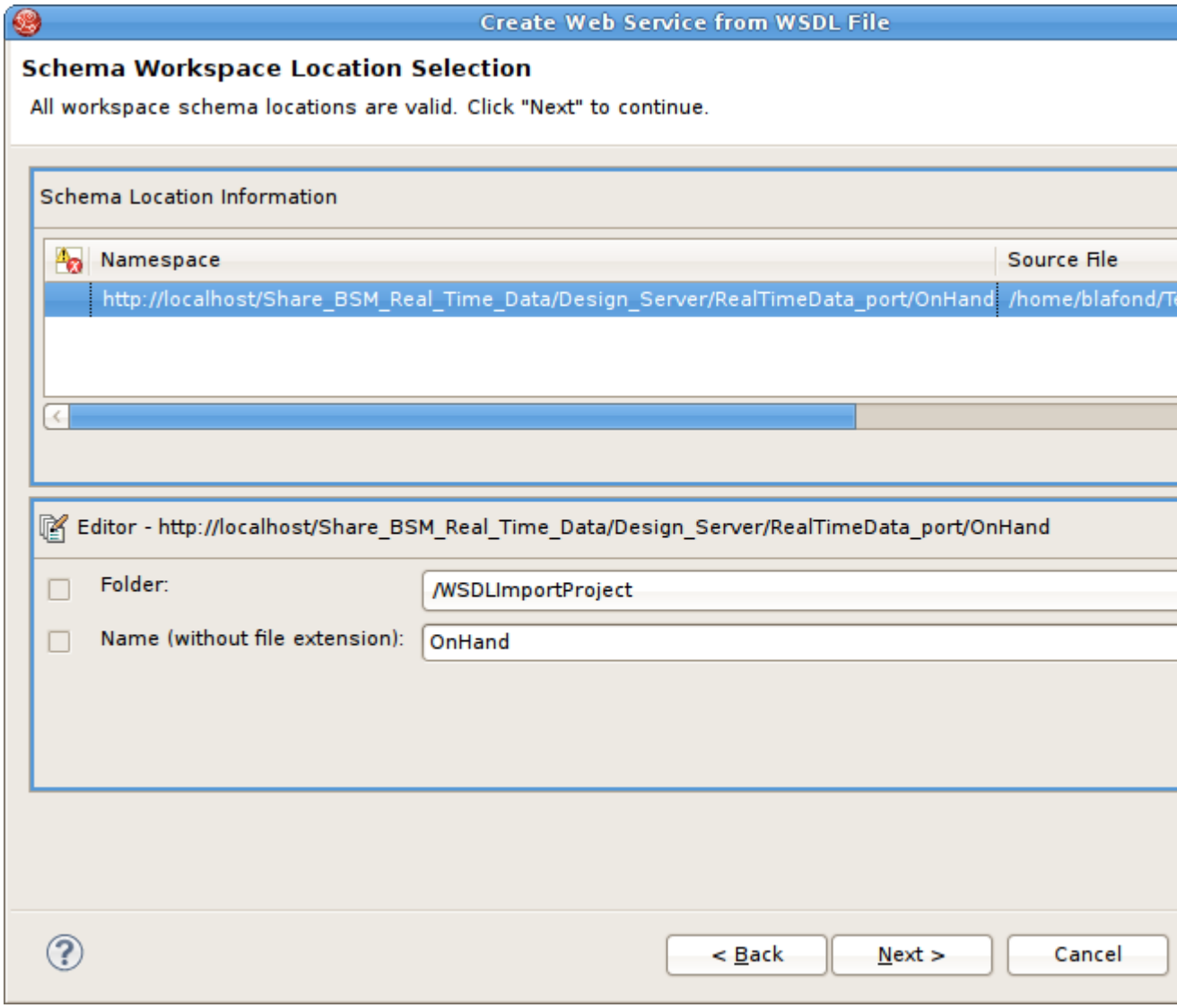


Figure 5.63. Namespace Resolution Dialog

- **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.

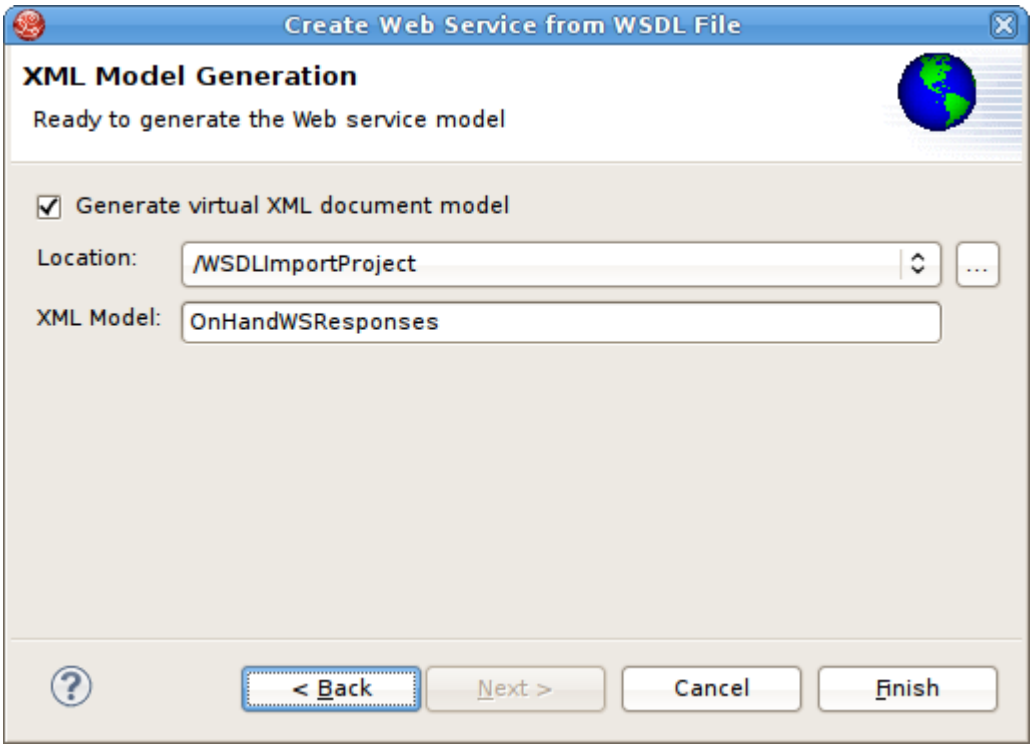


Figure 5.64. Namespace Resolution Dialog

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, a 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.

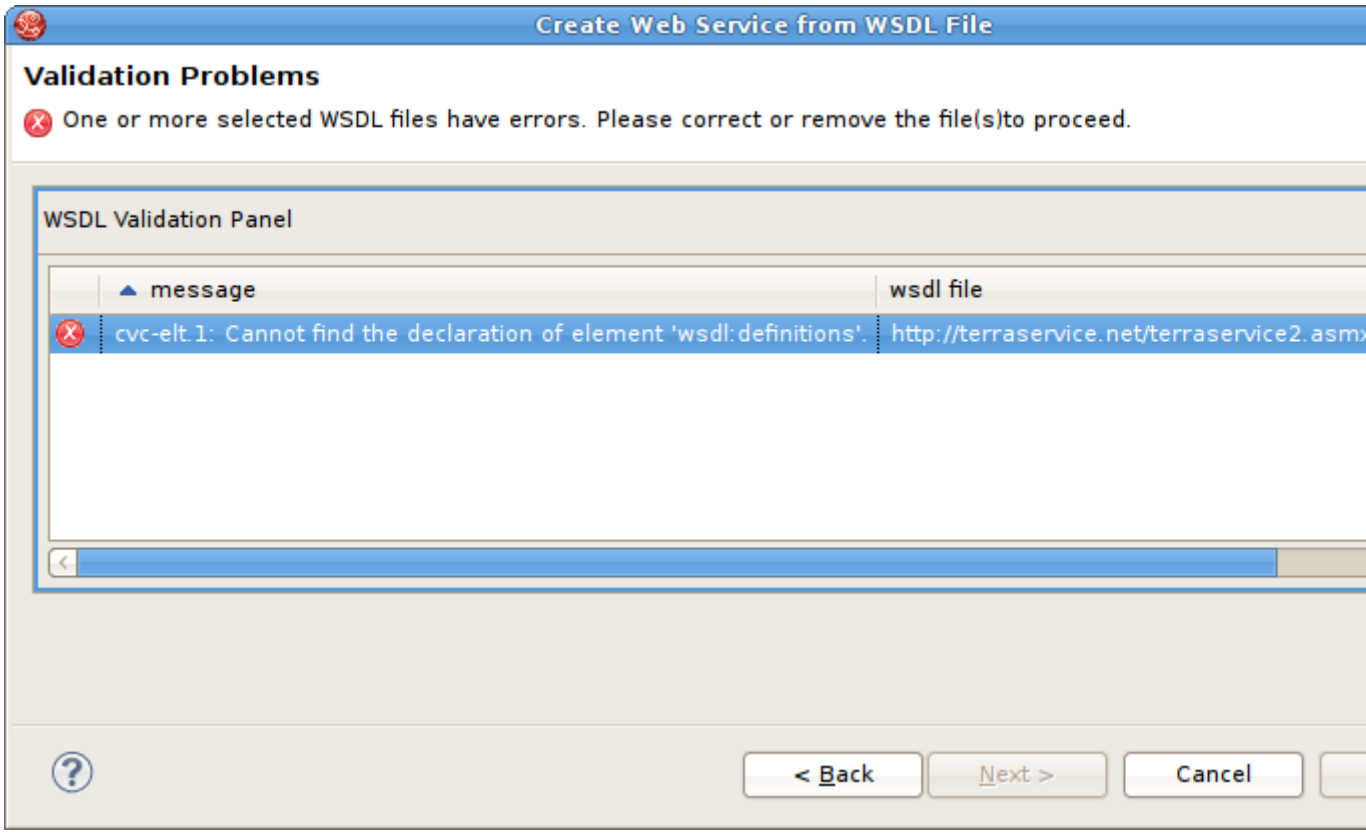


Figure 5.65. WSDL Validation Problems Dialog

5.10.2. 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 **Teiid Designer > WSDL File or URL >> Web Service Model** and click **Next>**
  - **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>**.

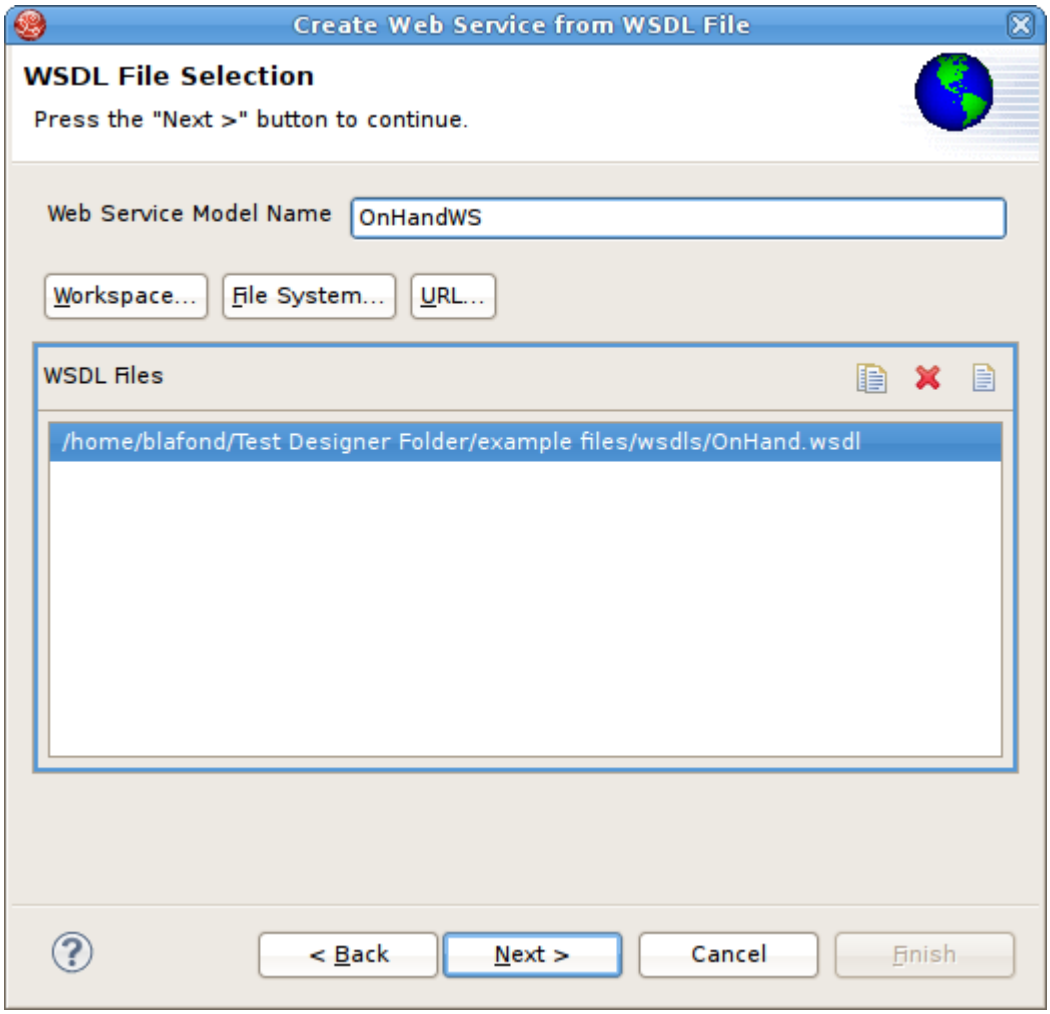


Figure 5.66. WSDL File Selection Dialog

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 (**WSDL**s 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.

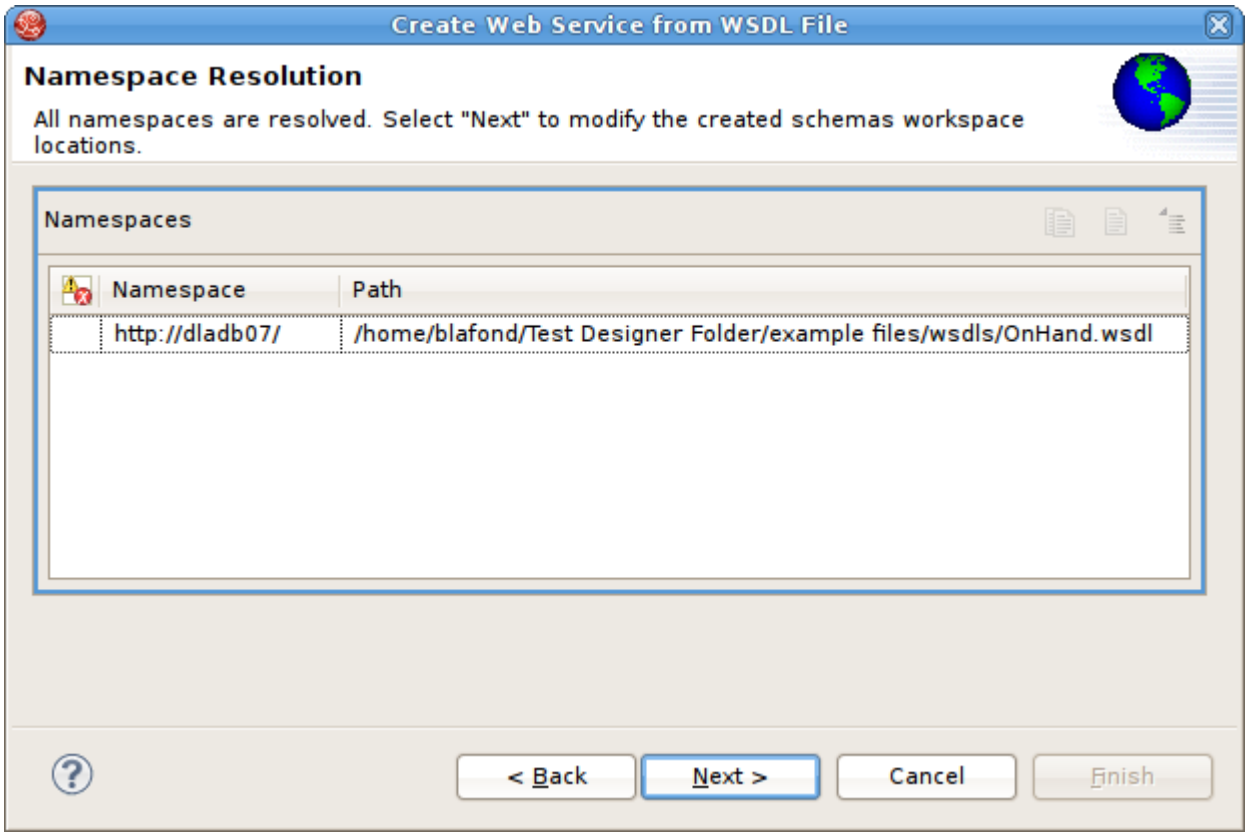


Figure 5.67. Namespace Resolution Dialog

- **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.

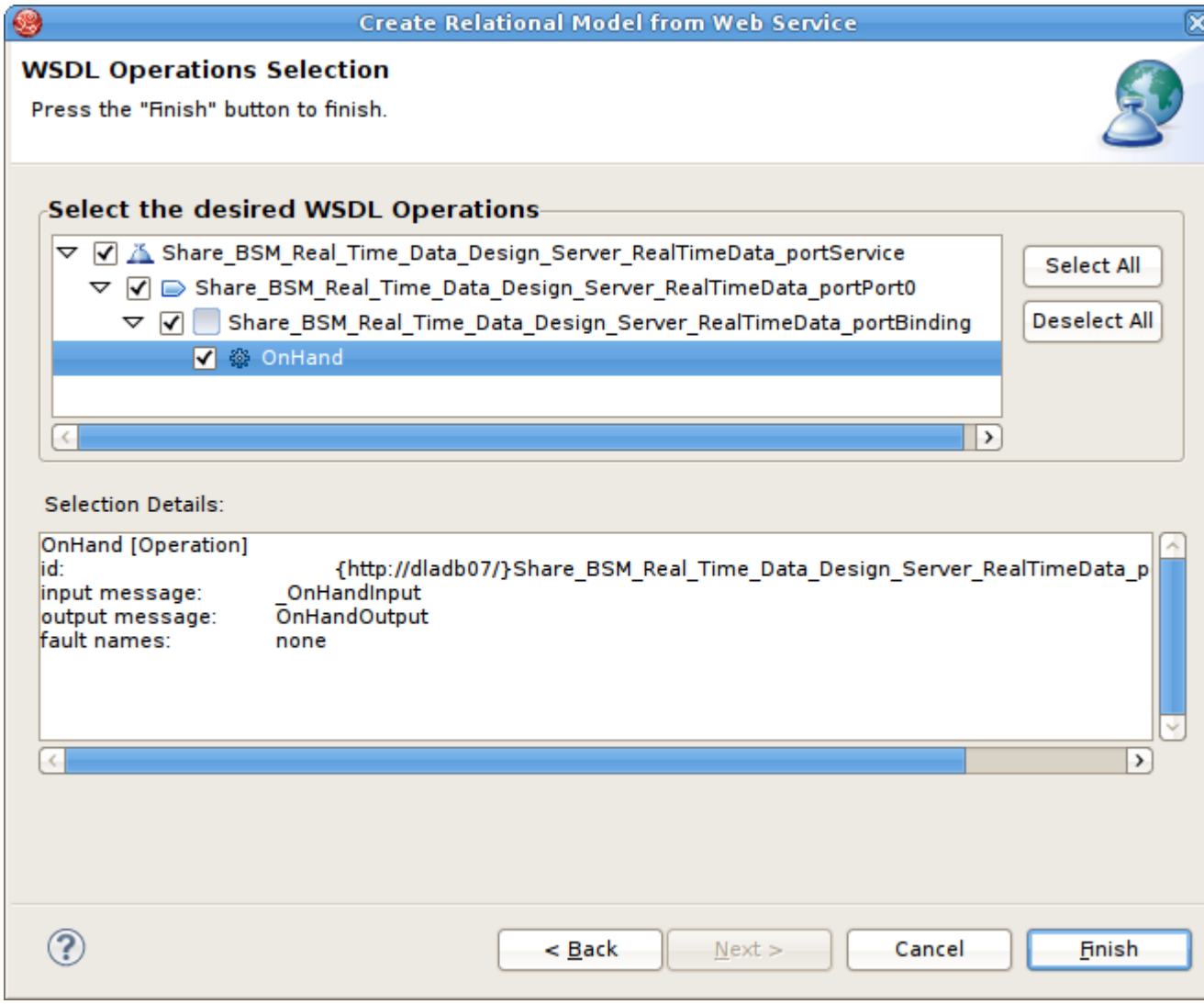


Figure 5.68. Namespace Resolution Dialog

- **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

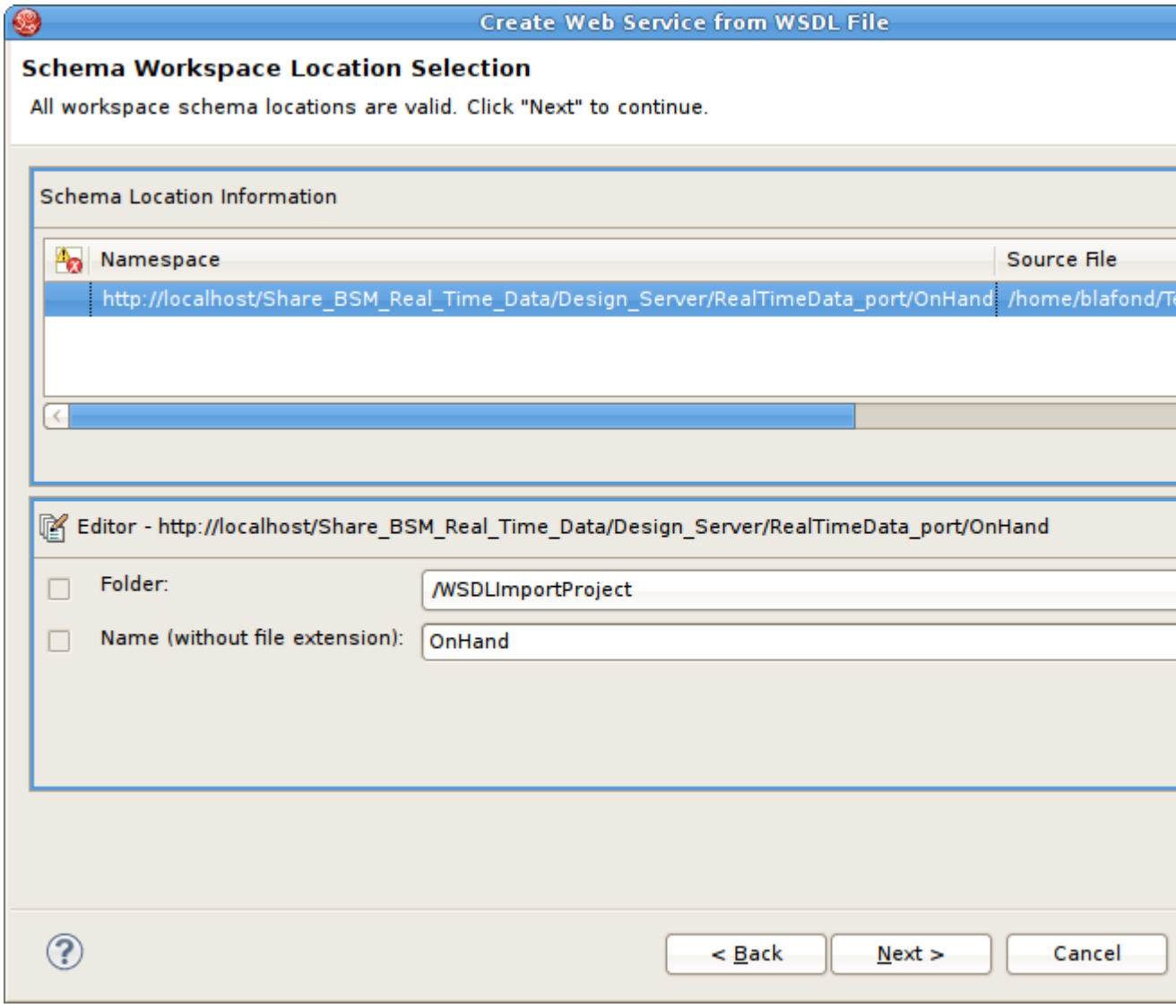


Figure 5.69. Namespace Resolution Dialog

- **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.

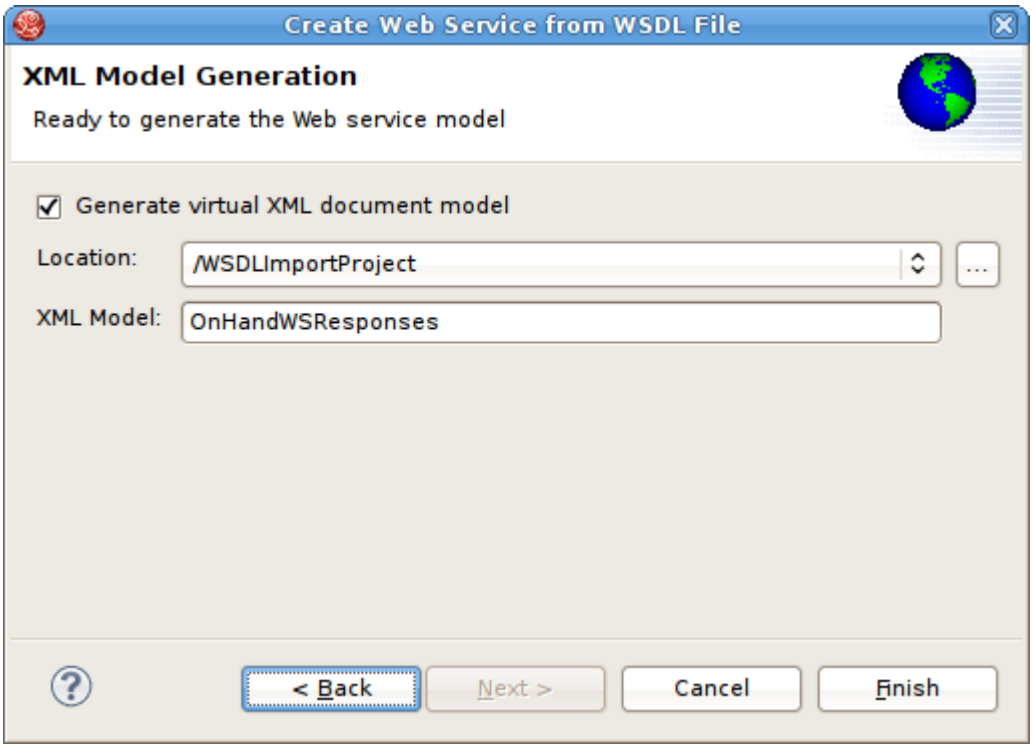


Figure 5.70. Namespace Resolution Dialog

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, a 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.

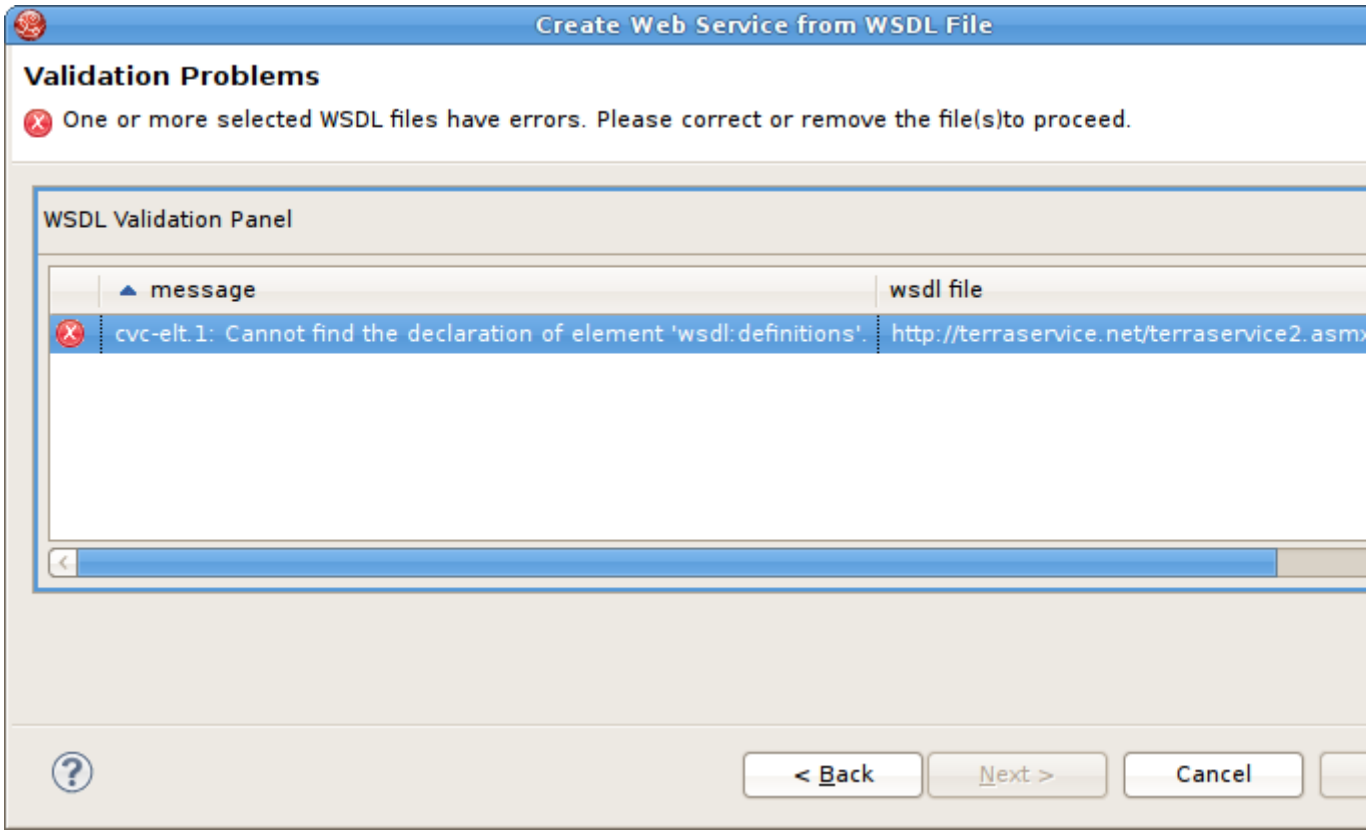


Figure 5.71. WSDL Validation Problems Dialog

5.10.3. 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 **Teiid Designer > WSDL File or URL >> Web Service Model** and click **Next>**
  - **Step 3** - Input a valid name for your **Web Service** model and select the **URL...** button.
    - Enter a valid WSDL URL. If the URL cannot be validated then an error will be displayed and the **OK>** button disabled.
    - If the WSDL is protected by basic HTTP authentication then this option should be selected and the appropriate username and password entered.
    - Click **OK>** to continue.

    Click **Next>** to continue.

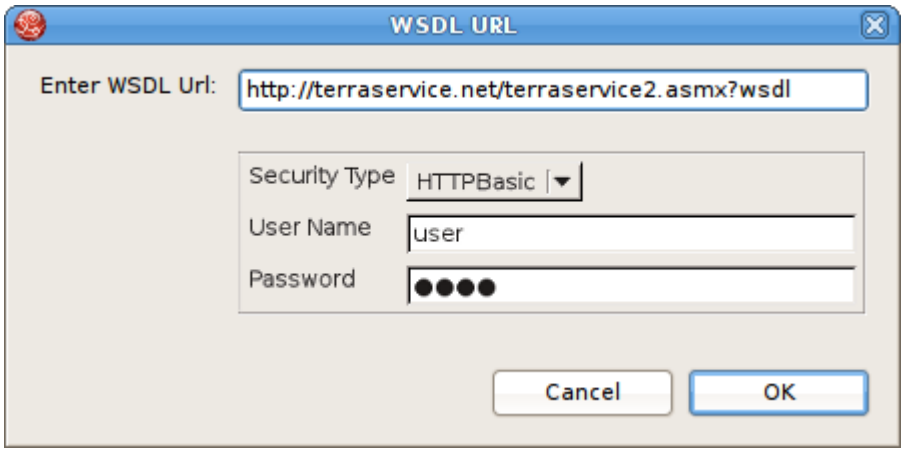


Figure 5.72. WSDL URL Dialog

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 (**WSDL**s 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.

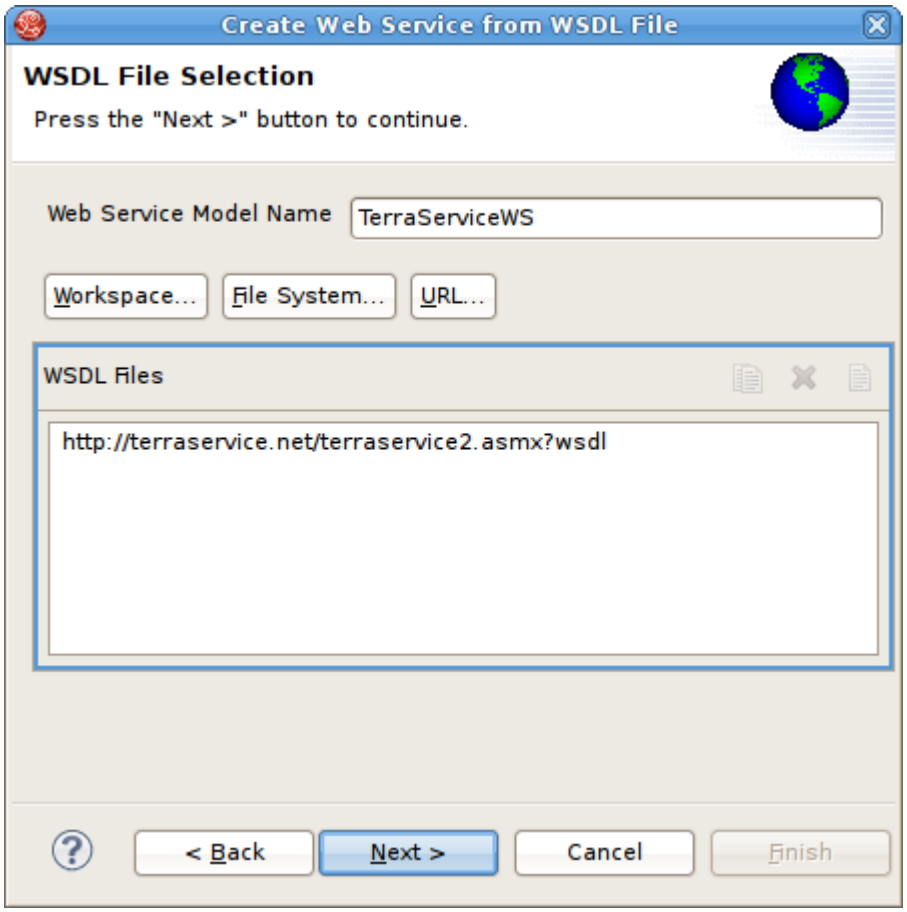


Figure 5.73. Namespace Resolution Dialog

- **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.

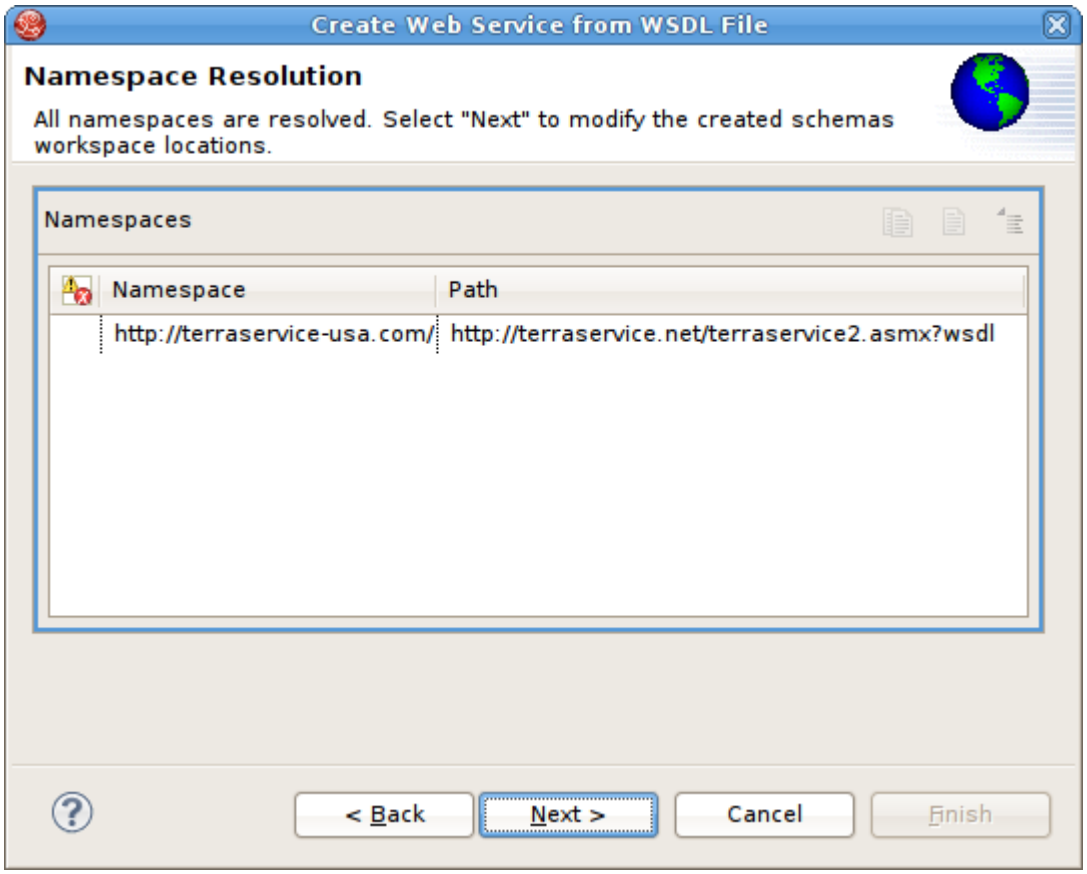


Figure 5.74. Namespace Resolution Dialog

- **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

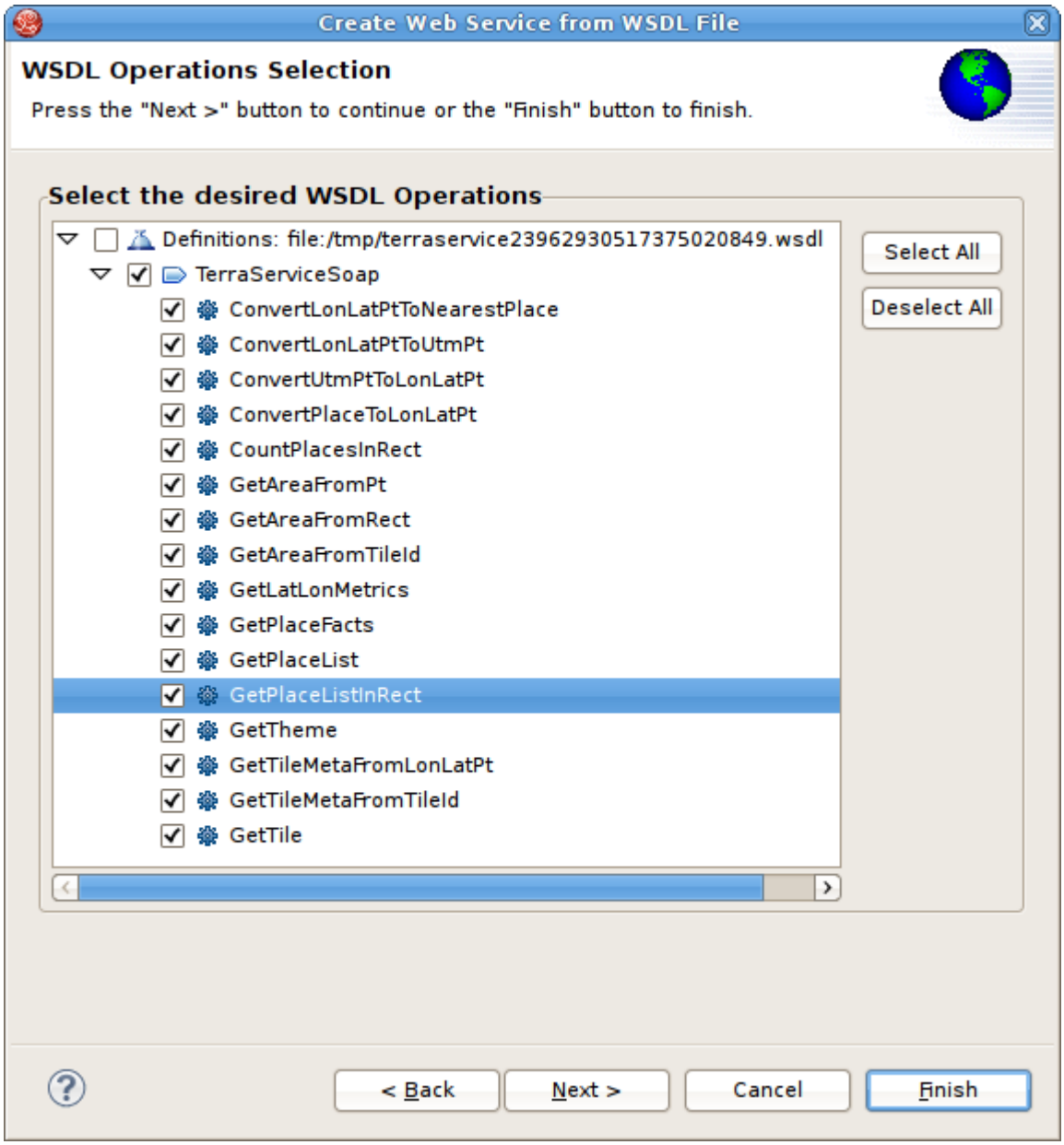


Figure 5.75. Namespace Resolution Dialog

- **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.

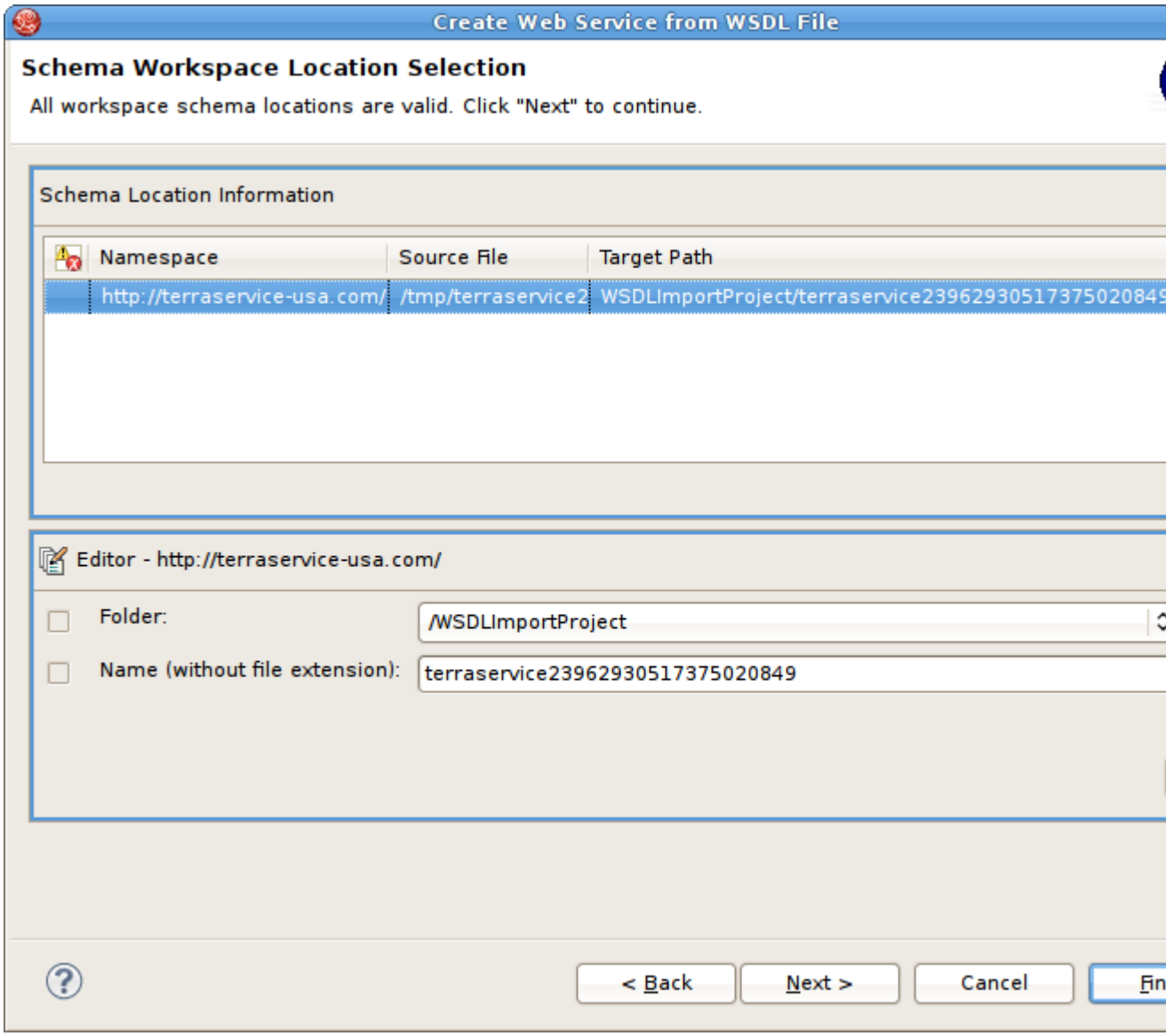


Figure 5.76. Namespace Resolution Dialog

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, a 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.

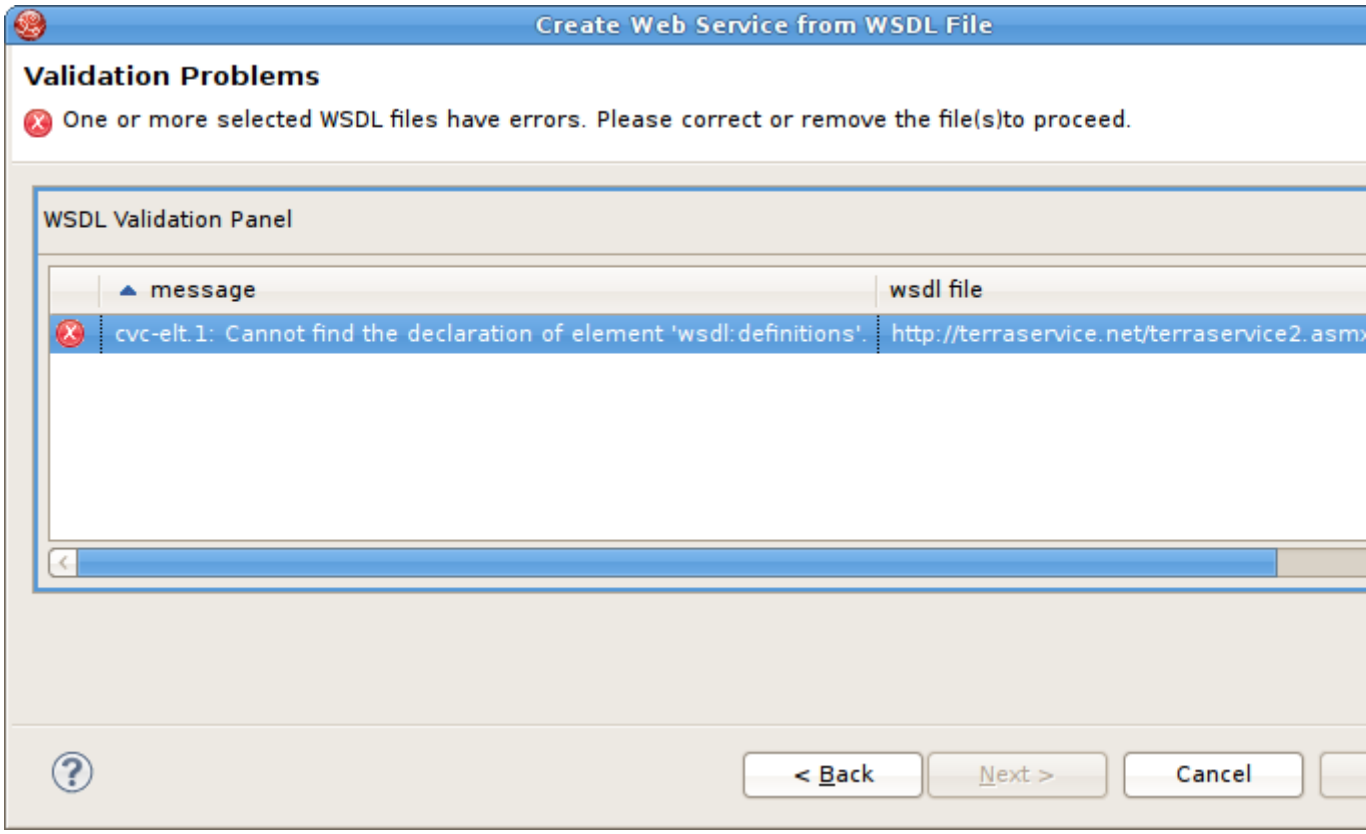


Figure 5.77. WSDL Validation Problems Dialog

5.11. Import from an XML Schema File

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

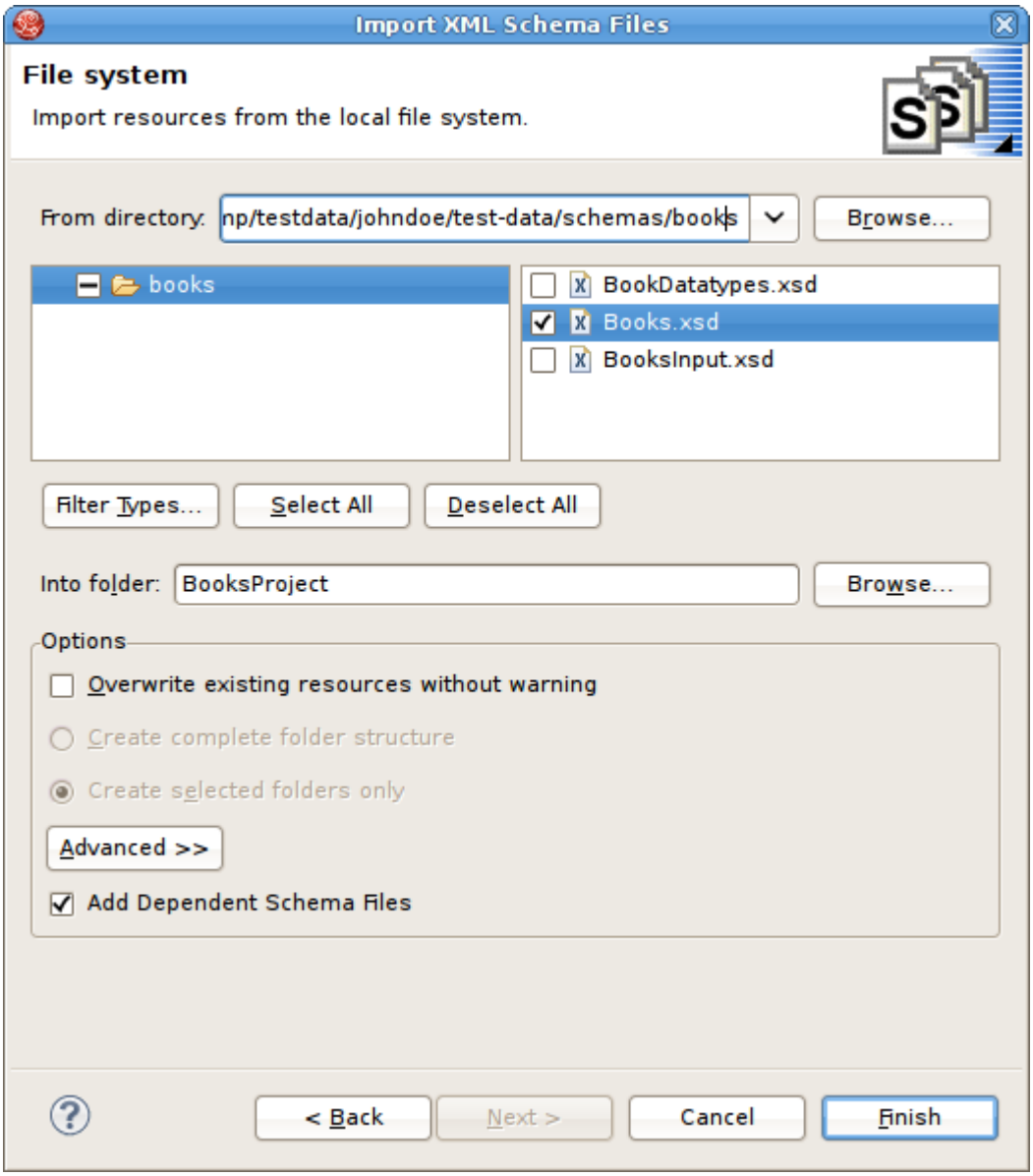


Figure 5.78. Select XML From File System

- **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.

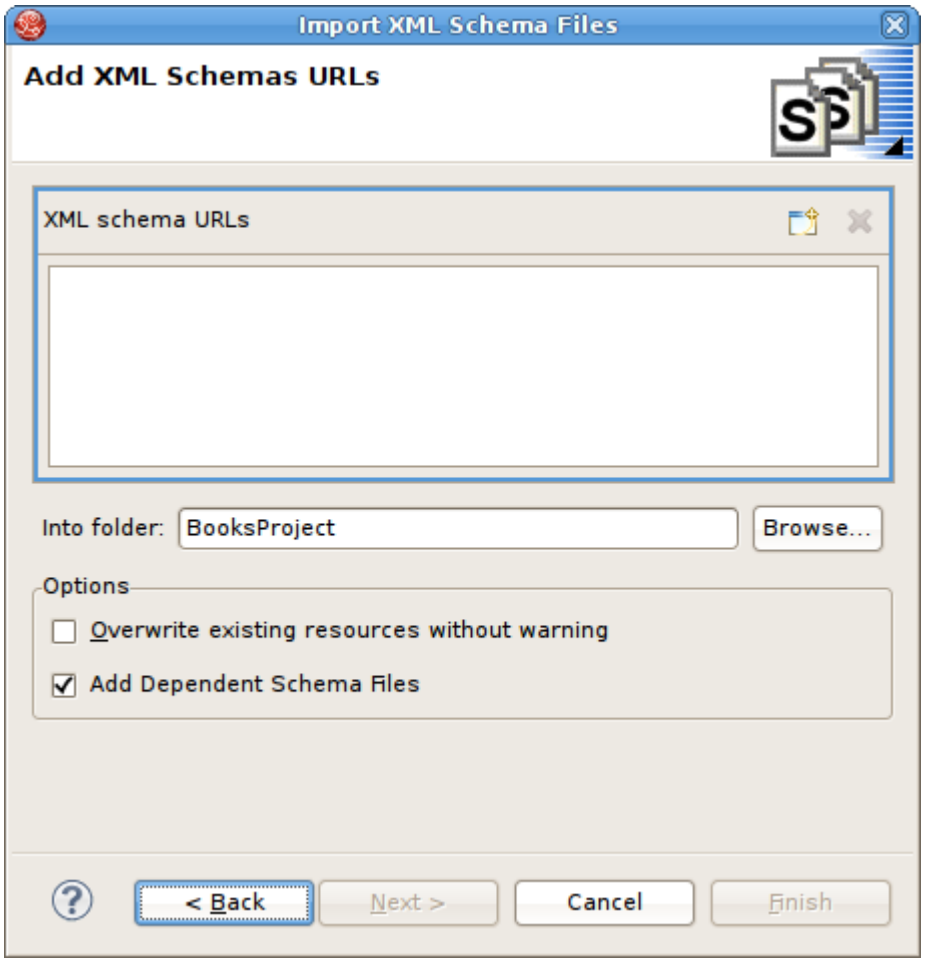


Figure 5.79. Add XML Schema URLs Dialog

- **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.

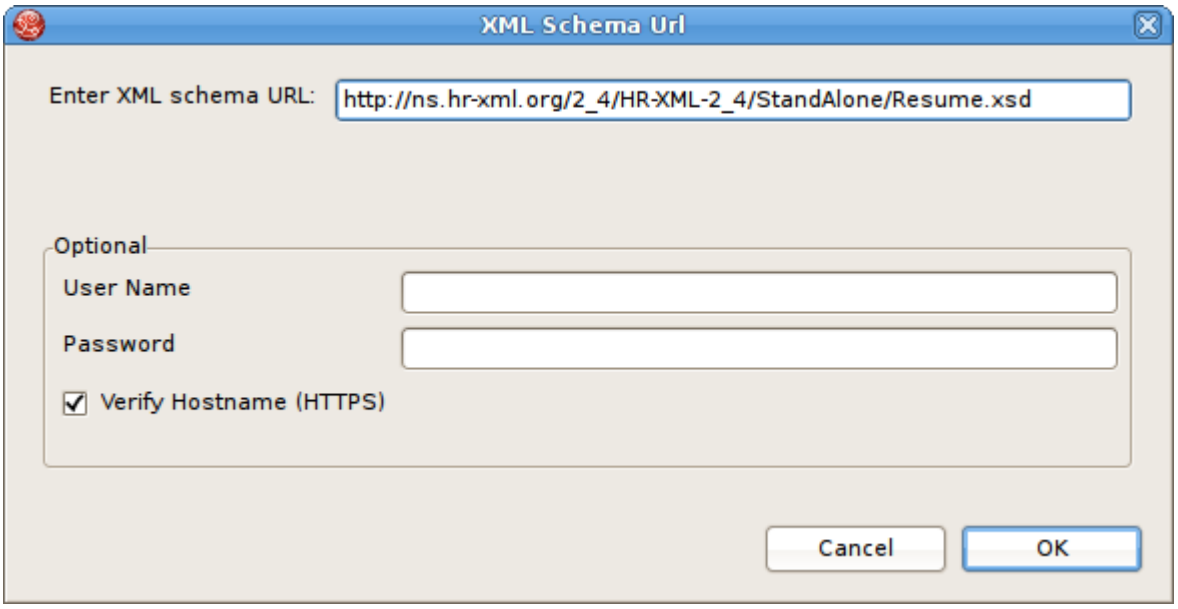


Figure 5.80. Add XML Schema URLs

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

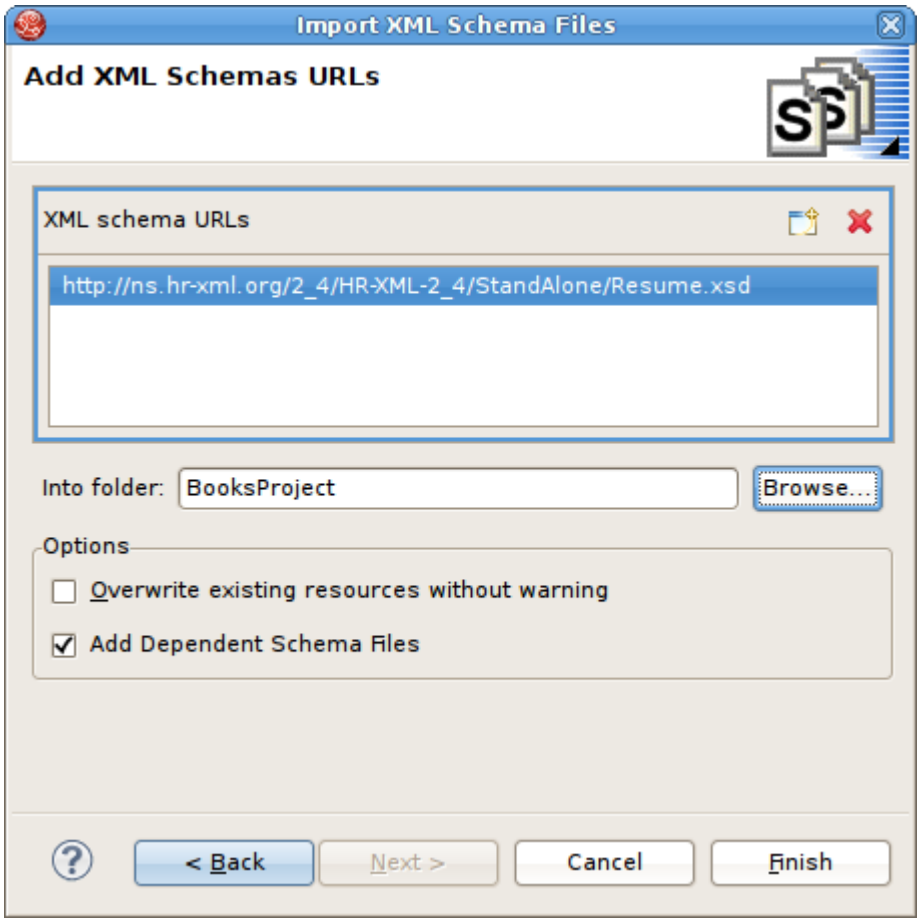


Figure 5.81. Add XML Schema URLs

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

Note

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

5.12. Import From an LDAP Server

The Lightweight Directory Access Protocol, or LDAP, is a network protocol for accessing directory services over TCP/IP. A directory contains a collection of related data, organized hierarchically in a tree format. Each node in the tree is a directory entry, and each entry consists of a set of attribute-value pairs. Each directory entry has a unique identifier, known as its Distinguished Name (DN). The DN consists of a Relative Distinguished Name (RDN), constructed from an attribute from the entry itself, followed by the parent entry's DN.

In Teiid Designer, an LDAP Server can be modelled by doing the following:

- LDAP subtrees are represented as if they were tables in a relational database.
- Each node in the subtree is represented as a row in the table.
- Each attribute of the given node can be represented as a column in the table.
- The RDN (or DN) can be used to represent a primary key.

The LDAP metadata is modeled using the relational metamodel. Each table in the relational model represents a directory entry while each row in the table represents a child entry of the directory entry. Each column of the table represents an attribute of the child entry that may exist. In general, each table and column defines the LDAP-specific information in the property “Name In Source”. This allows the connector to identify the attribute or Base DN name within the data source, ie. within LDAP. The actual name of the table and column can differ from the name in source, allowing for more descriptive labelling in models and queries.

- The relational source model can be created from LDAP data 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 > LDAP Service >> Source Model** and click **Next>**
  - **Step 3** - Select an existing or previous LDAP 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.

    Note

    Information required for a new connection:

    - Connection Username / password - an administrator account to browse the ldap tree, eg. cn=Manager,dc=birds-of-prey,dc=org
    - Connection URL, eg. ldap://falcon:389
    - Principal Distinguished Name (DN) Suffix - the root DN of the ldap tree
    - An LDAP Connection Factory implementation class, eg. com.sun.jndi.ldap.LdapCtxFactory

    Selection of the connection profile populates the LDAP Service URL and DN Suffix fields. The remaining requirements for the wizard page is the choosing of a suitable model file as the destination of the imported tables. If the selection is an existing model then the wizard will merge the new tables with the model's current content.

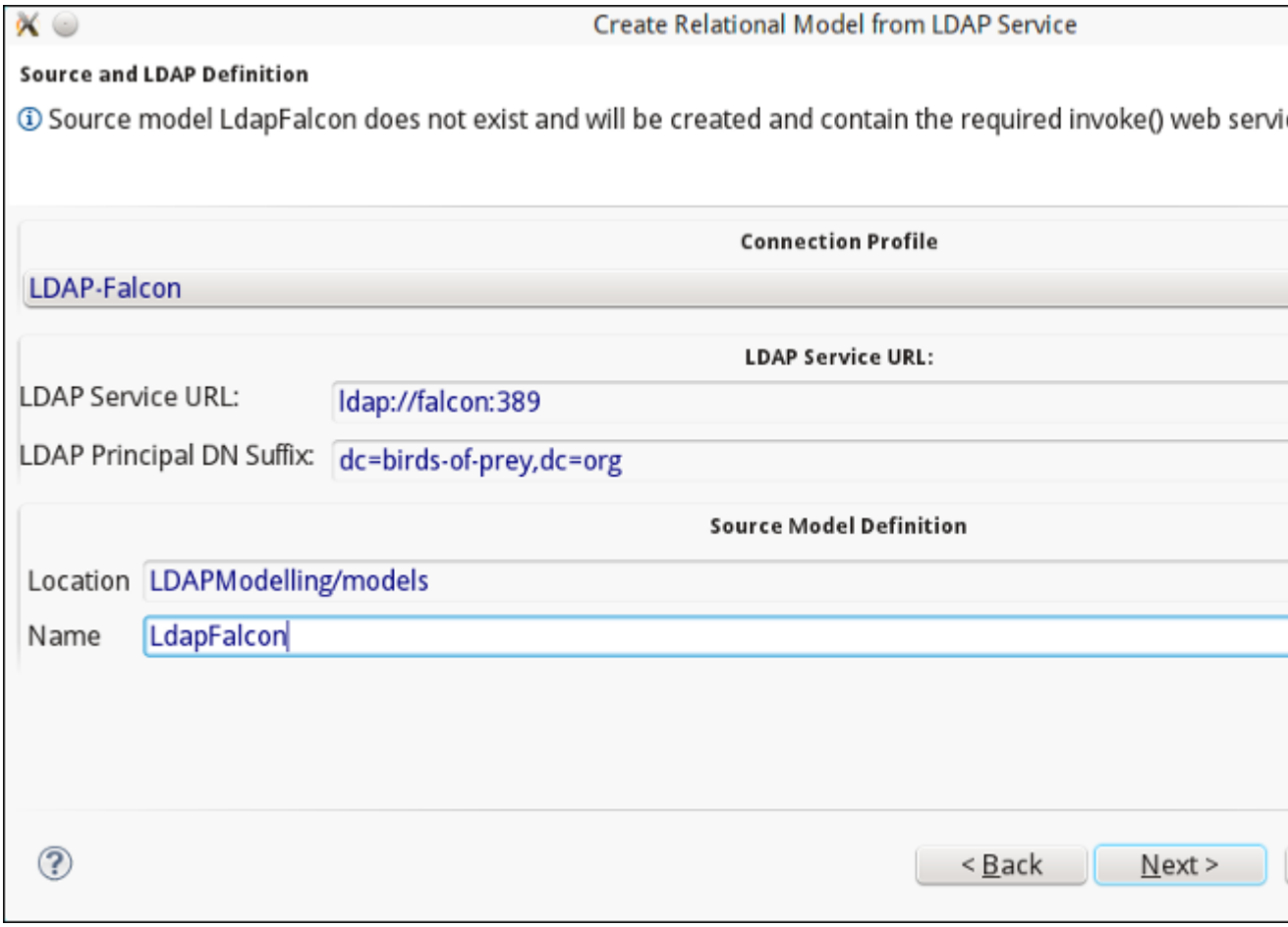


Figure 5.82. LDAP Definition Wizard Page

- **Step 4** - After selecting a *Connection Profile*, click **Next>**
- **Step 5** - On the **Select LDAP Entries to be modelled as tables** page, select the LDAP entries from the tree to be created as tables in the source model. Entries are selected by the ticking of their respective checkboxes in the tree. The highlighting of an entry displays the following attributes:
  - Table Name - this is the table's label and can be modified to a more readable value
  - Table Source Name - the fully qualified entry name. This is not editable in the wizard and should remain unchanged in the subsequently created source model
  - Table Source Name Suffix - an additional suffix can be added that further limits the scope of the table's search criteria. The suffix is in the format of *?search_scope?objectClass_name* where search_scope is one of OBJECT_SCOPE (first and only one entry returned), ONELEVEL_SCOPE (only entries directly below the selected entry are returned) or SUBTREE_SCOPE (recursively return all entries below the selected entry) and objectClass_name is the name of a specific type of objectClass in the LDAP tree, eg.

return only the 'inetOrgPerson' entries. Both criteria are optional (but the '?'s are not) so it is possible to have a suffix such as *? ? inetOrgPerson*.

Click **Next>** .

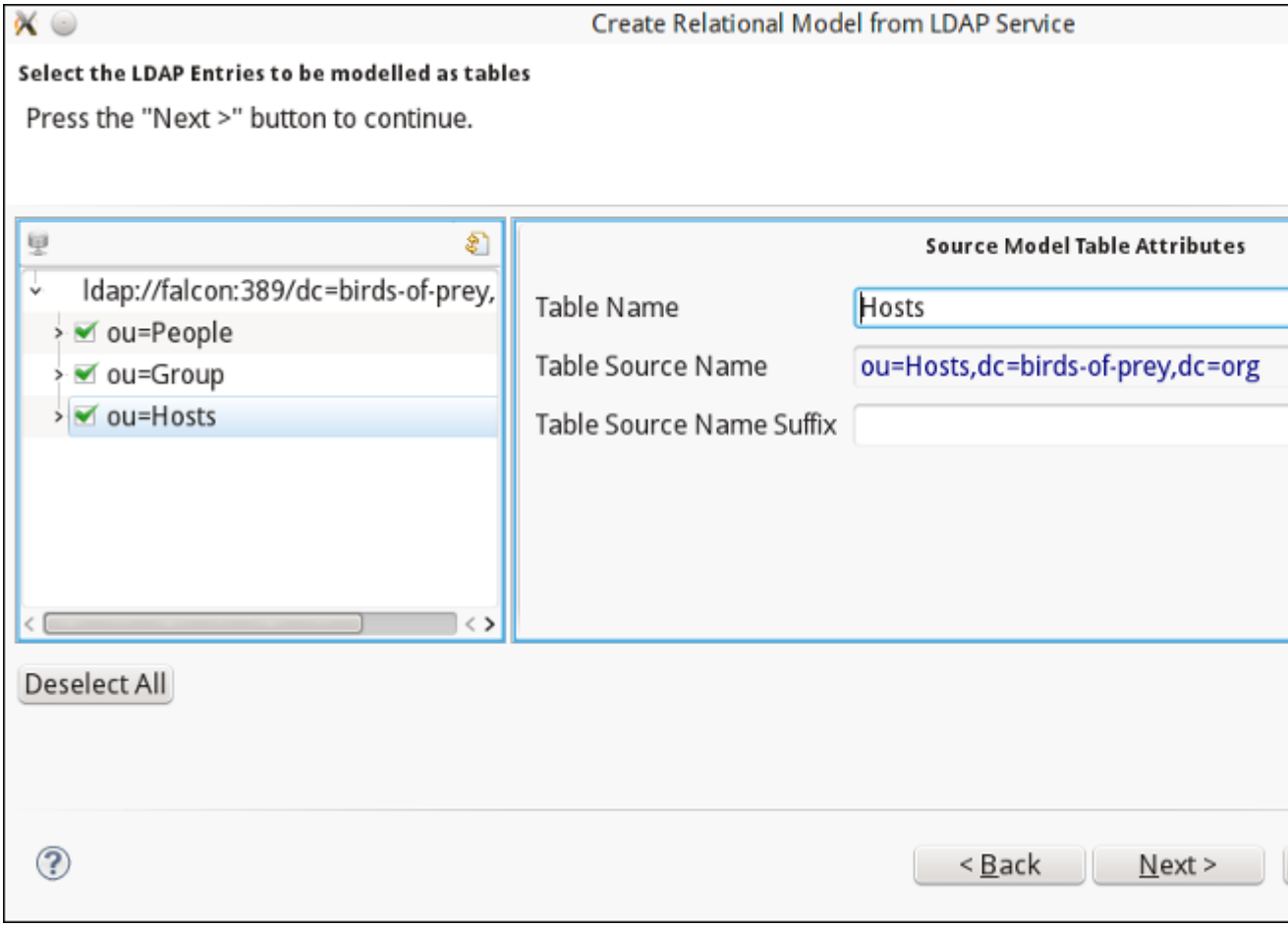


Figure 5.83. Select LDAP entries to be modelled as tables page

- **Step 6** - The **Select the LDAP Attributes to be modelled as columns** page displays the previously selected LDAP entries and the attributes of their *child* entries. The purpose of a selected attribute is to be created as a column in the relevant source model table. Attributes are selected by the ticking of their respective checkboxes in the tree. The highlighting of an attribute displays the following properties:
  - Column Name - this is the column's label and can be modified to a more readable value
  - Column Source Name - the real LDAP attribute name. This is not editable in the wizard and should remain unchanged in the subsequently created source model
  - Column Distinct Value Count - The number of distinct values assigned to the specific attribute in the LDAP service. This value is useful in optimising queries using the source model. This is not editable in the wizard and should remain unchanged in the subsequently created source model.

- Column Null Value Count - The number of entries where the specific attribute has no value assigned in the LDAP service. This value is useful in optimising queries using the source model. This is not editable in the wizard and should remain unchanged in the subsequently created source model.
- Column Length - The maximum length of existing values assigned to the attribute in the LDAP service. This value is assigned as the maximum length of the column. This is not editable in the wizard but can be edited in the source model later should this be required.

Click **Finish>** .

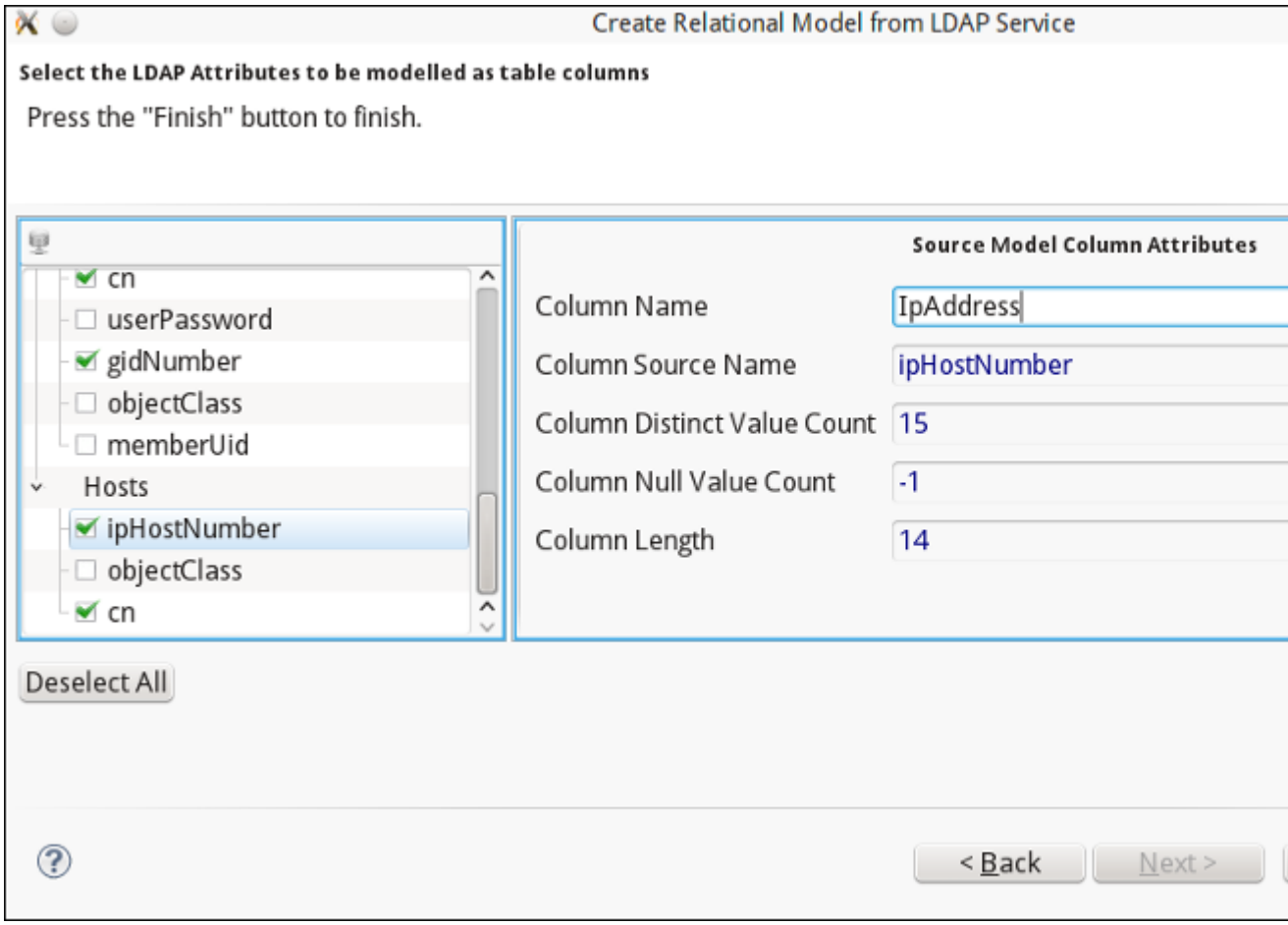


Figure 5.84. Select the LDAP Attributes to be modelled as columns page

Once the wizard has completed, the new source model will be created.

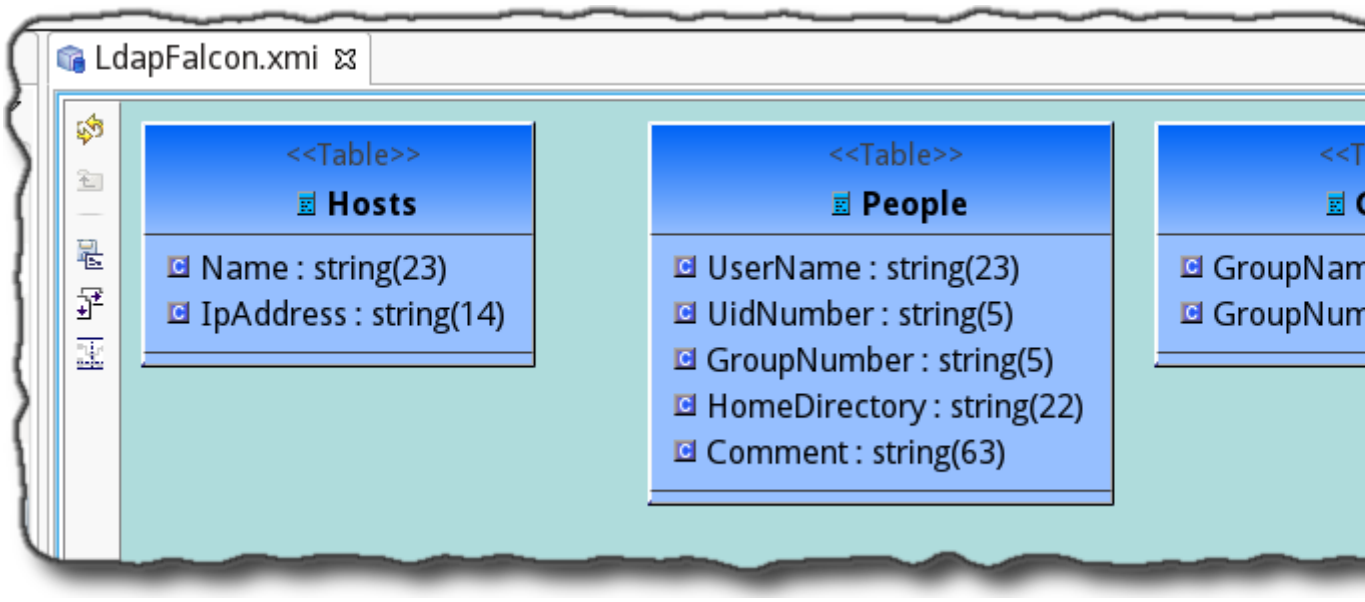


Figure 5.85. Example of an LDAP source model

The LDAP connector also provides an update capability. However, additional modeling requirements are imposed beyond those required for read-only access. The additional requirements are described below.

- *Supports Update* table property - to enable updates, each source model table must have this property set to 'true';
- *Updateable* column property - to enable updates, each column in the source model table must have this property set to 'true';
- Additional required columns:
  - *DN* - for all update types (INSERT, UPDATE, and DELETE), the distinguished name must be modeled as a column, setting the *name in source* to *dn*. For UPDATE and DELETE capability, the DN must be specified in the criteria clause while for INSERT, the DN must be one of the column values to be set.
  - *objectClass* - for INSERT, the objectclass must be modelled as a column, setting the *name in source* to *objectClass*. It must also be one of the column values to be set.
  - *additional* - each entry defined in the LDAP directory's schema may also have one or more additional required columns. This is dependent on the LDAP server implementatation so consult the LDAP documentation accordingly.

A selection of example queries:

- SELECT * FROM LdapModel.People
- ```
  INSERT INTO LdapModel.People (dn, sn,
  ```
  ```

```
    objectclass, Name) VALUES
    ('cn=JoeYoung,ou=people,dc=example,dc=org','Young','person',
    'Joe Young')
```

- ```
  UPDATE LdapModel.People SET
    PhoneNumber='(314) 299-2999' WHERE
    DN='cn=JoeYoung,ou=people,dc=example,dc=org'
  ```

- ```
  DELETE FROM LdapModel.People WHERE
    DN='cn=JoeYoung,ou=people,dc=example,dc=org'
  ```

Creating and Editing Model Objects

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

6.1. Creating New Model Objects

As discussed in the introduction, *Section 1.1, "What is Teiid Designer?"*, Teiid Designer provides 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 *Section D.2.1, "Model Explorer View"* view, *Section D.3.1.2, "Diagram Editor"* or *Section D.3.1.3, "Table Editor"* using the following actions:
  - **New Child Action**
  - **New Sibling Action**
  - **New Association Action**

6.1.1. New Child Action

- To create new child model objects in the *Section D.2.1, "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.

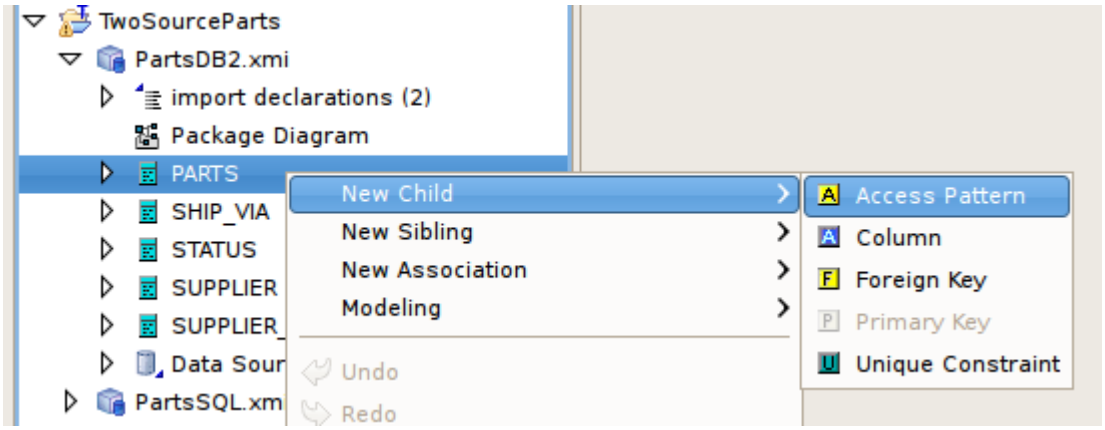


Figure 6.1. New Child Action In Model Explorer

- **Step 3** - The new model object displays on the *Section D.2.1, "Model Explorer View"* and is highlighted for renaming.

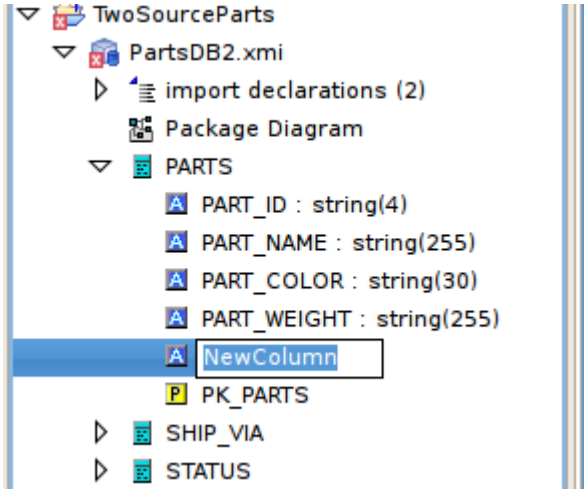


Figure 6.2. New Model Object In Explorer

- To create new child model objects in the *Section D.3.1.2, "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.

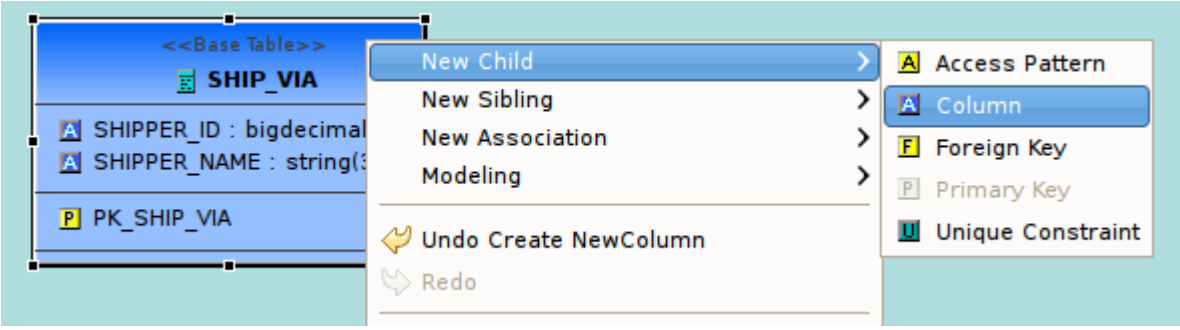


Figure 6.3. New Child Action In Diagram

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

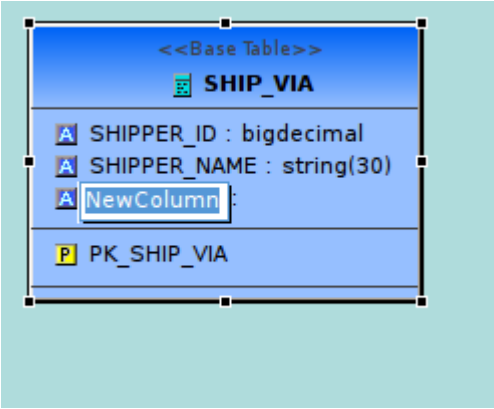


Figure 6.4. New Model Object In Diagram

- To create new child model objects in the *Section D.3.1.3, "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 6.5. 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.

6.1.2. New Sibling Action

- To create new sibling model objects in the *Section D.2.1, "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.

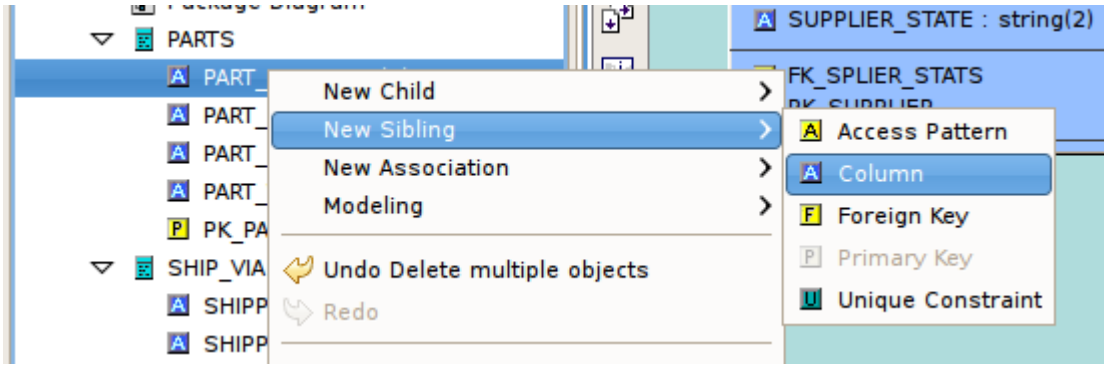


Figure 6.6. New Sibling Action In Model Explorer

- **Step 3** - The new model object displays on the *Section D.2.1, "Model Explorer View"* and is highlighted for renaming.

- To create new sibling model objects in the *Section D.3.1.2, "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.

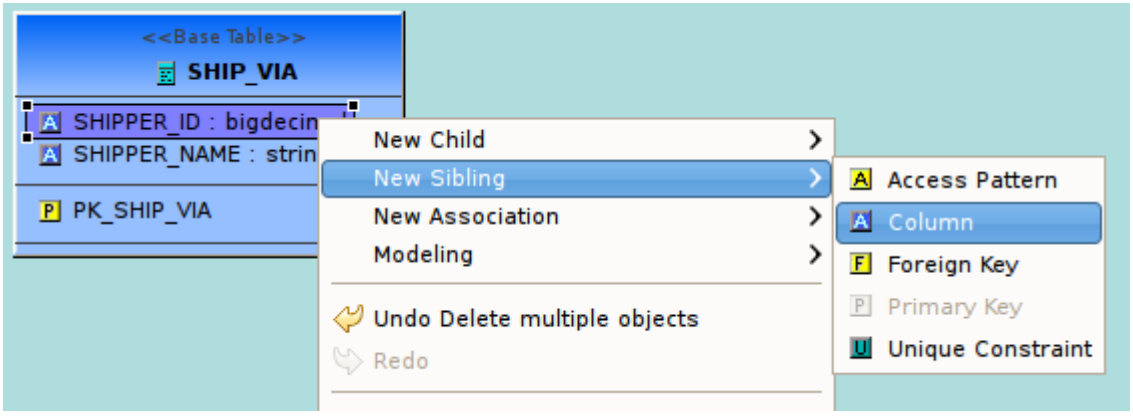


Figure 6.7. New Sibling Action In Diagram

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

- To create new sibling model objects in the *Section D.3.1.3, "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.

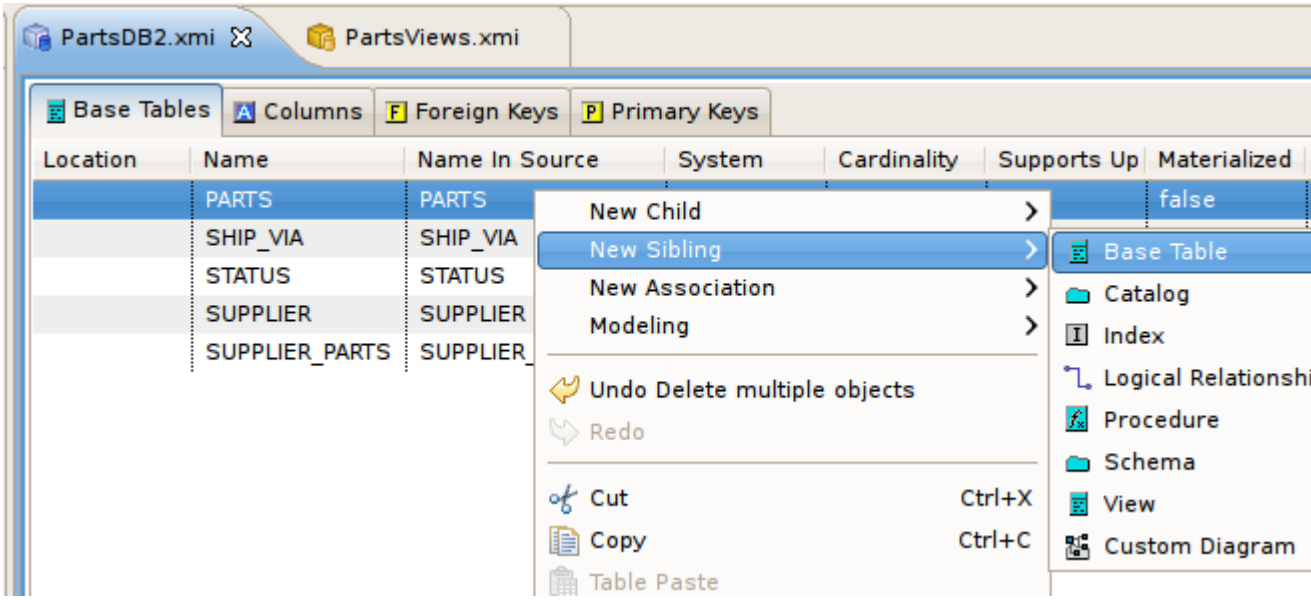


Figure 6.8. New Sibling Action In Table Editor

- **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.

6.1.3. New Association Action

- To create new associations between model objects in the *Section D.2.1, "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.**.

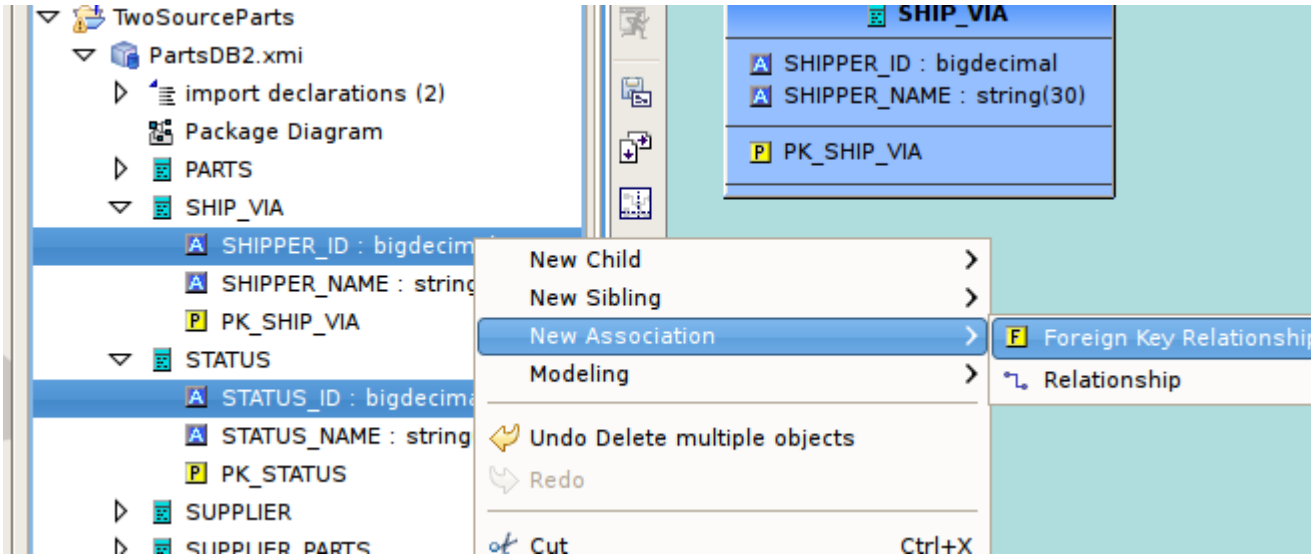


Figure 6.9. New Association Action In Model Explorer

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

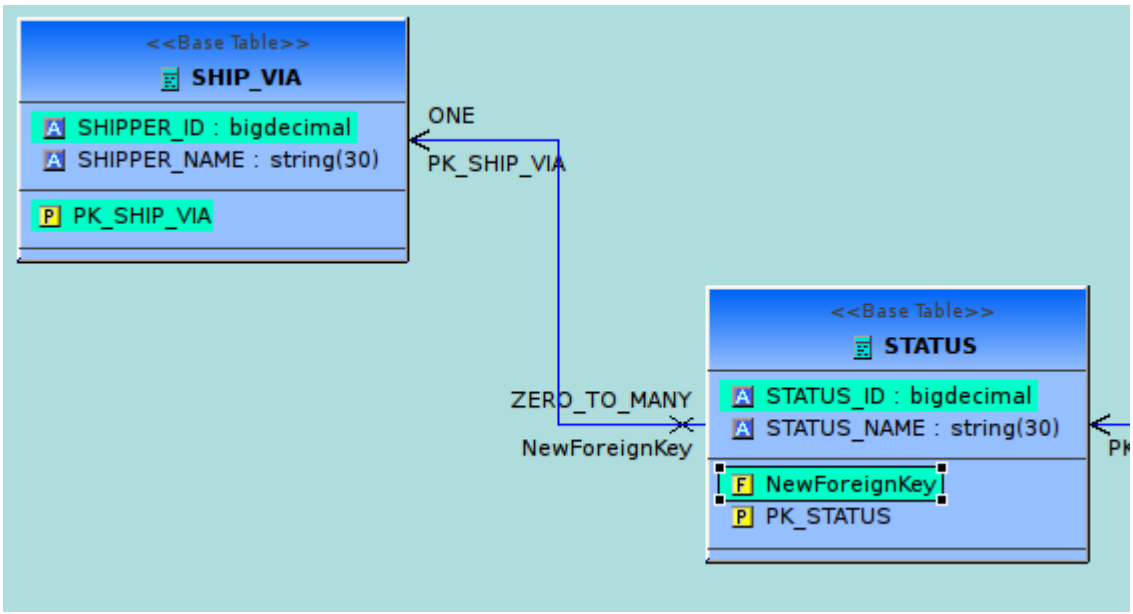


Figure 6.10. New Association In Diagram

- To create new associations between model objects in the *Section D.3.1.2, "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 D.3.1.3, "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.

6.2. New Model Object Wizards

In addition to the simple new object actions, Teiid Designer provides New Child and New Sibling wizards for children of view and source relational models. Namely tables, views, procedures and indexes. The menu now looks like:

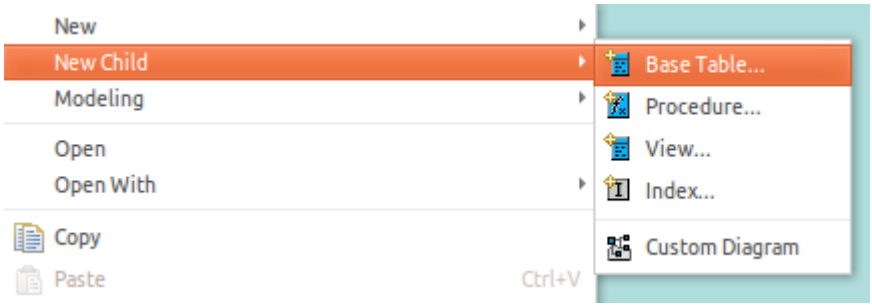


Figure 6.11. Relational Model New Child Menu

6.2.1. Create Relational Table Wizard

The **Create Relational Table Wizard**, shown below, allows creating a complete relational table including columns, unique keys, foreign keys definition and indexes.

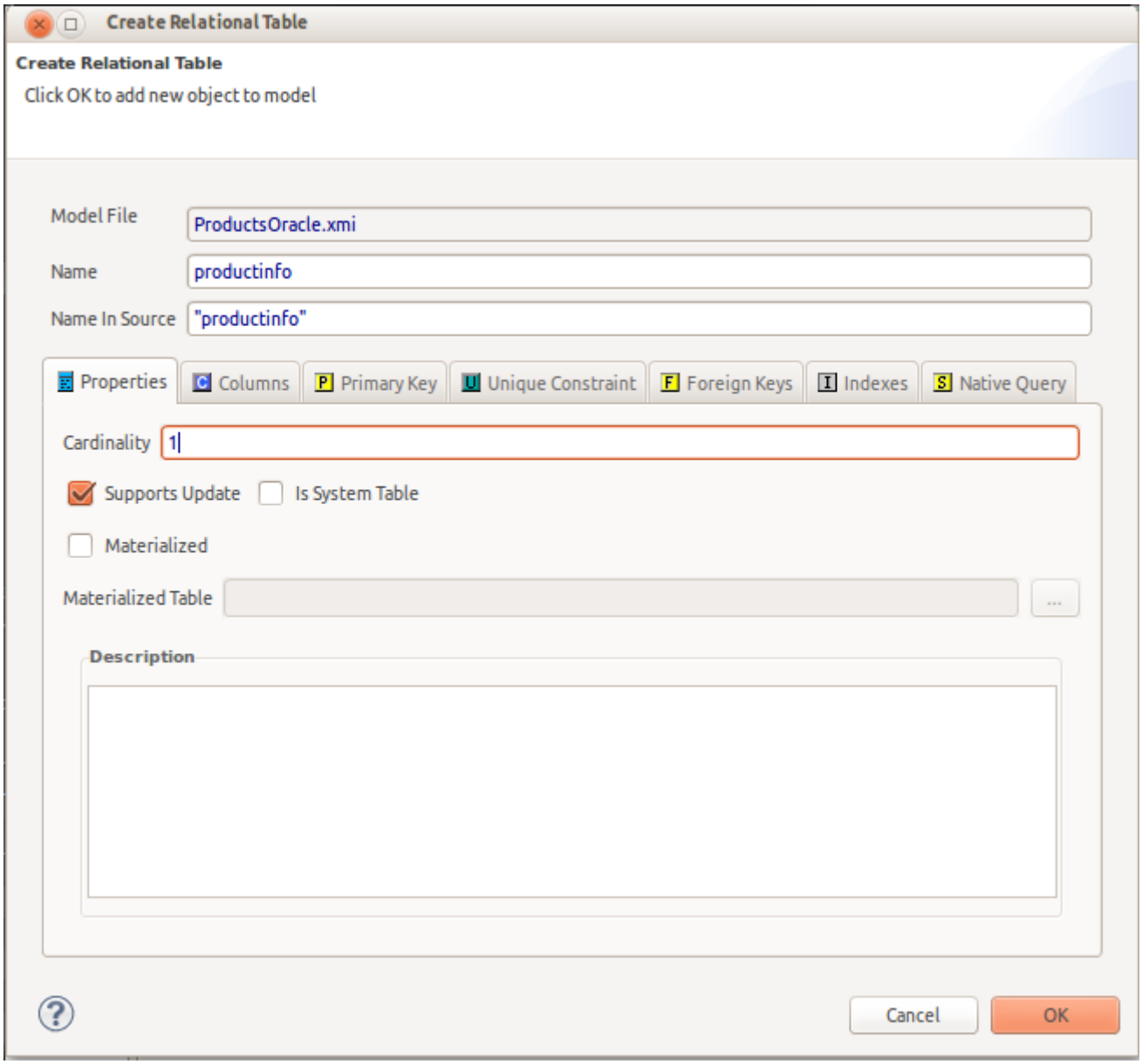


Figure 6.12. Create Relational Table Dialog

Note that the relational view wizard is identical to the relational table wizard.

6.2.2. Create Relational Procedure Wizard

The **Create Relational Procedure Wizard**, shown below, allows creating a complete relational Procedure including columns, unique keys, foreign keys definition and indexes. The relational procedure object can represent different types of procedures, including a standard procedure, source function (pushdown function) and native query procedures. When the **New Child > Procedure...** action is launched the first dialog gives you the option of selecting the procedure type. Note that user defined functions can be modeled in a view model.

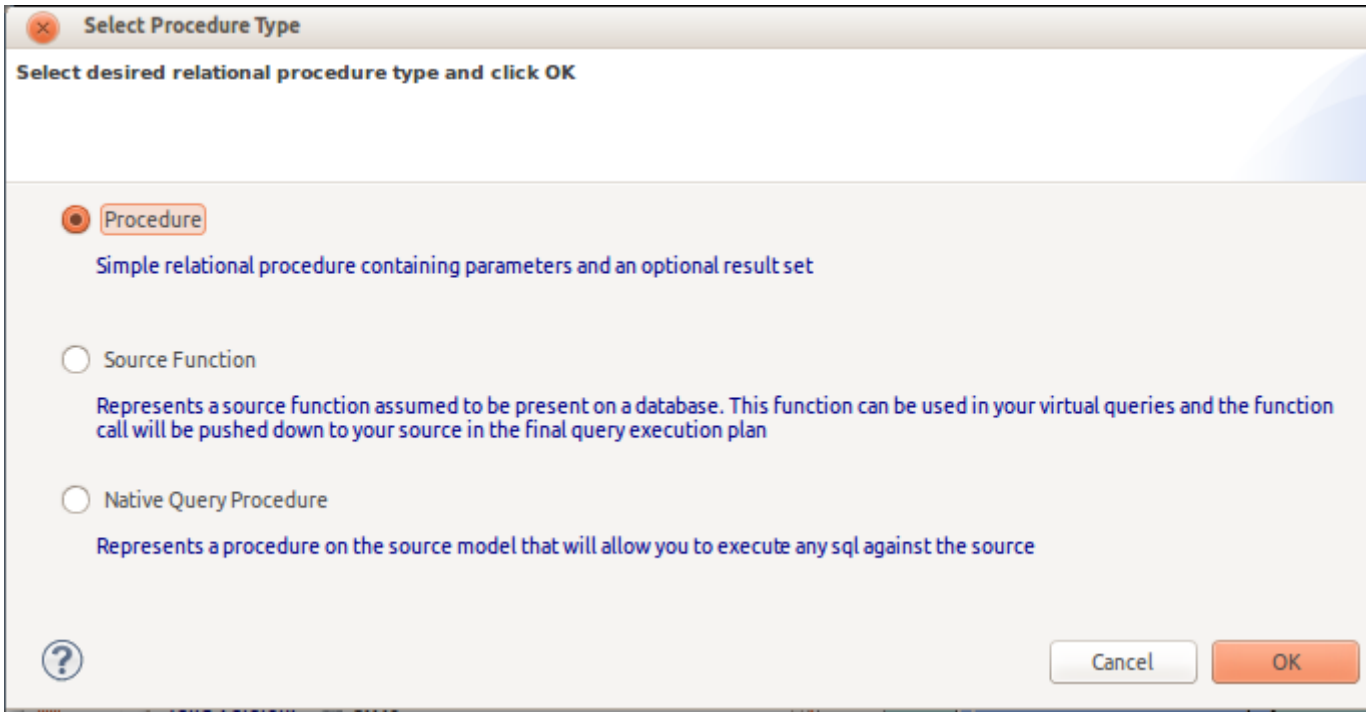


Figure 6.13. Select Procedure Type Dialog

The second dialog customizes the **Create Relational Procedure** dialog based on your selected type. The following dialog provides standard procedure data input.

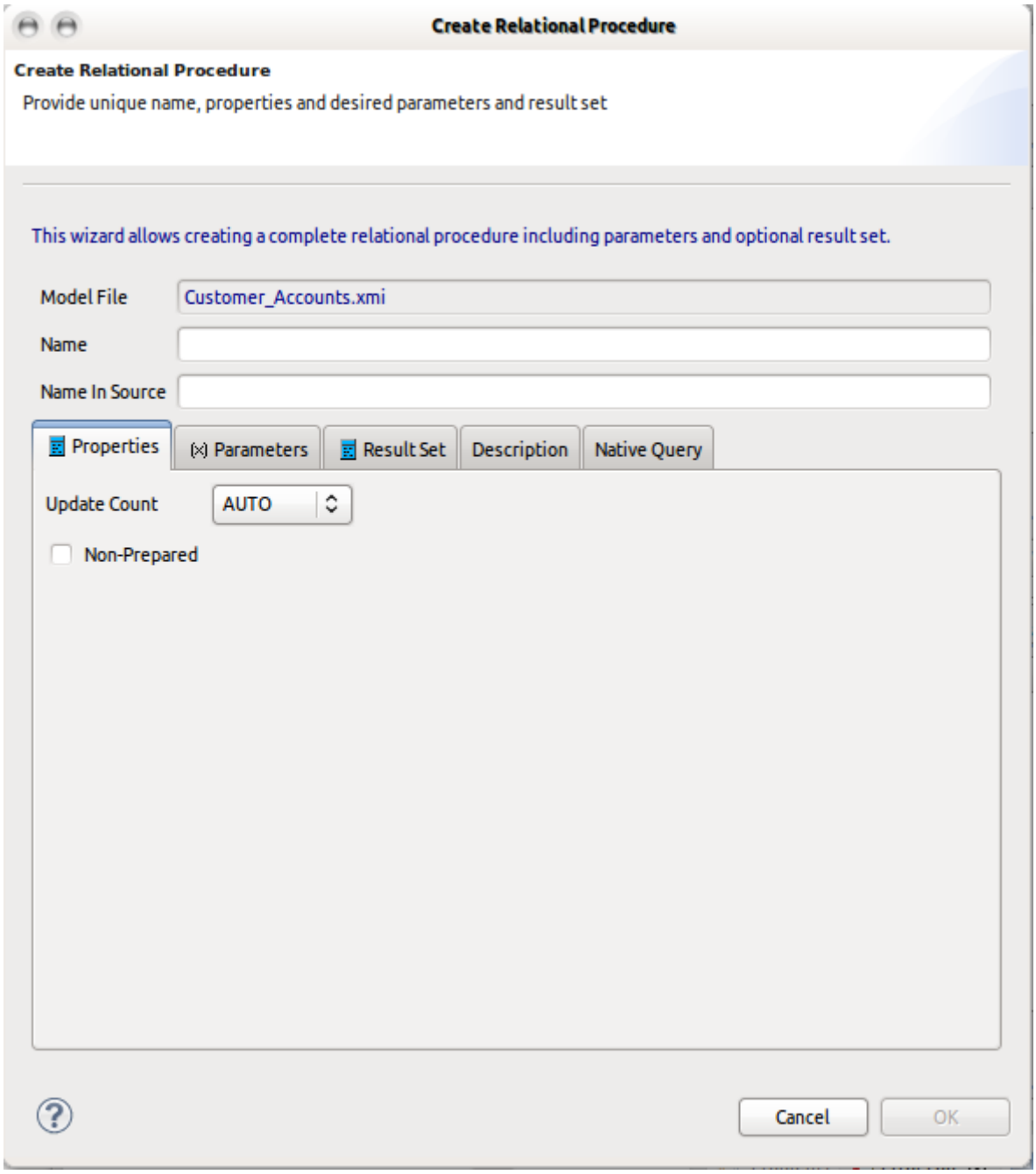


Figure 6.14. Create Relational Procedure Dialog

Source functions are procedures that are already deployed and accessable on your database. By defining source functions in your relational model, you can call these functions from within your transformation SQL and the functions will be pushed down to your database for execution.

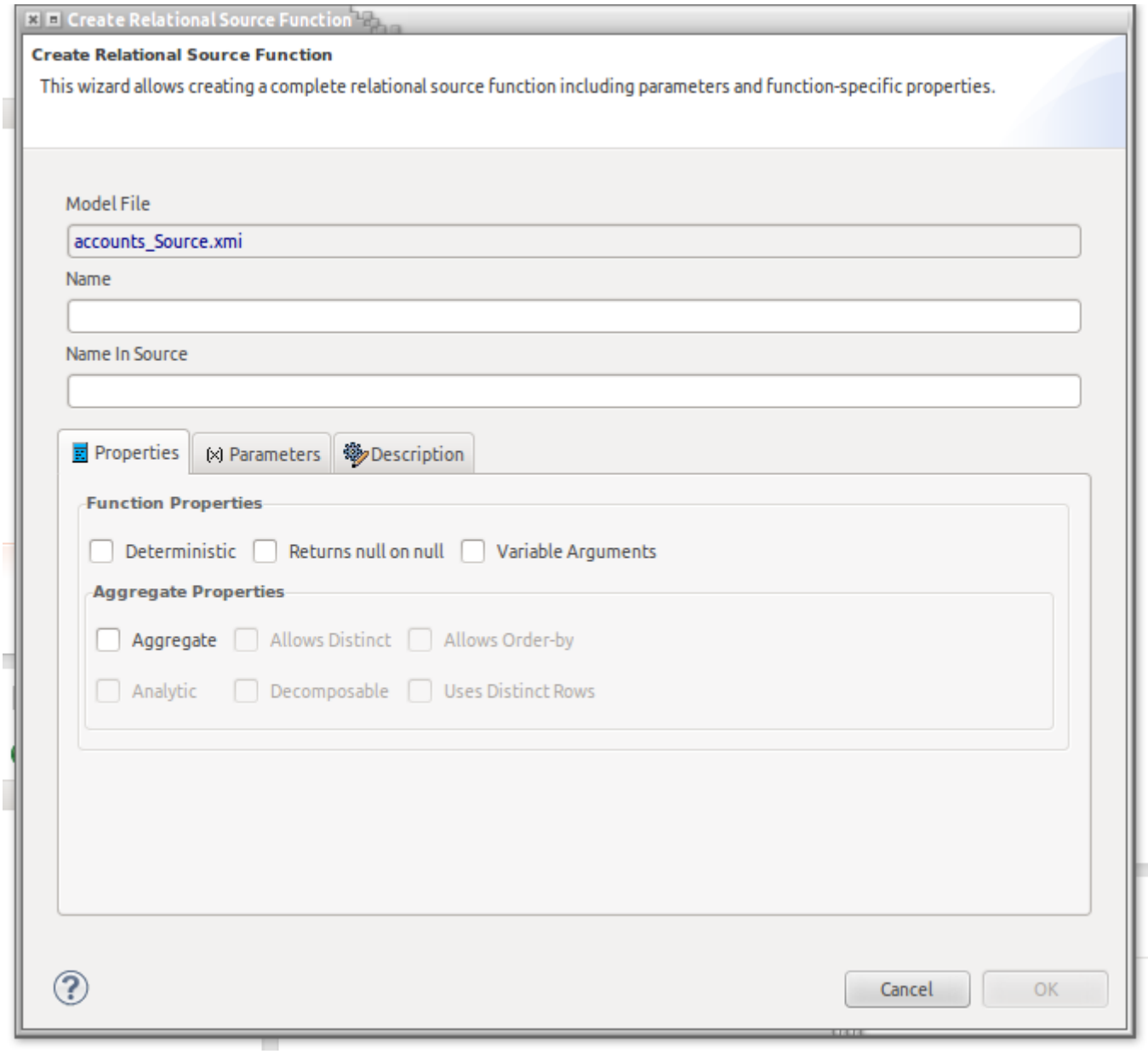


Figure 6.15. Create Relational Source Function Dialog

Native query procedures are procedures that are already deployed and accessable on your database. By defining source functions in your relational model, you can call these functions from within your transformation SQL and the functions will be pushed down to your database for execution.

The resulting Create Relational Native Query Procedure dialog defaults to the following procedure structure including 0 or more input parameters (defined by the user) and a result set with a single output column with object data type (user can edit/change the column name).

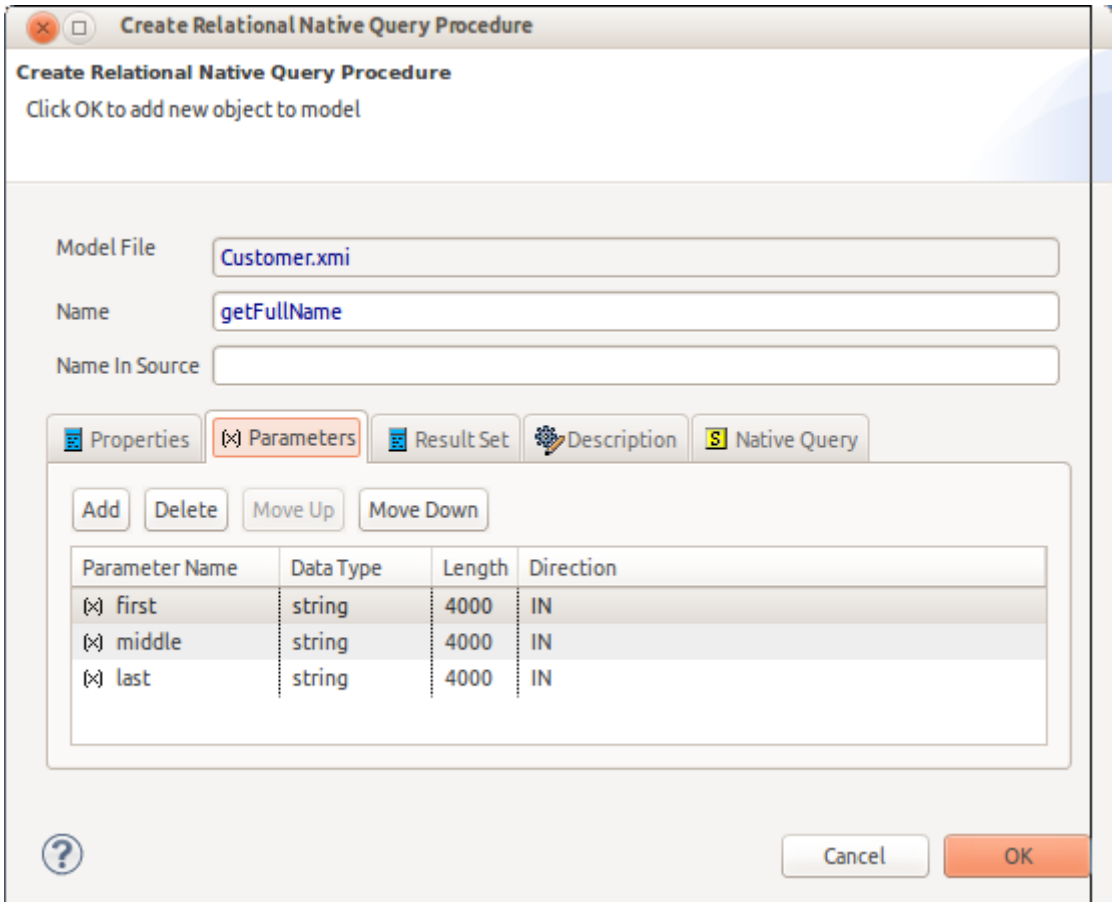


Figure 6.16. Create Relational Native Query Procedure Dialog

6.2.2.1. Create Function from SQL Transformation

Sometimes users will be creating SQL transformations that represent procedures or functions that have yet to be defined. Teiid Designer allows creating Source Functions or User Defined Functions based on the function defined in your SQL.

Below is an example of a query using a getBookInfo() function that does not exist yet. The result is a SQL validation error.

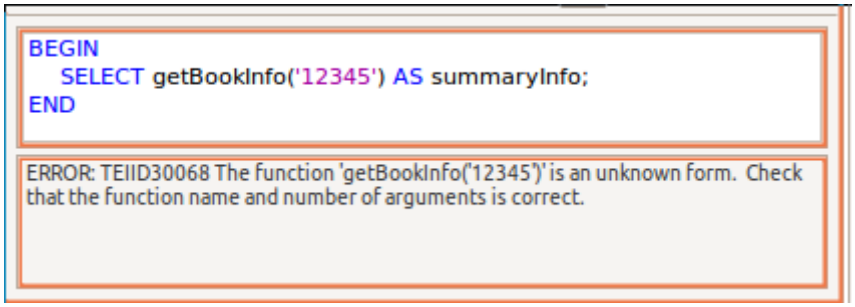


Figure 6.17. Undefined Function in Transformation

By selecting the function name and right-clicking, you can now select the Create Function action.

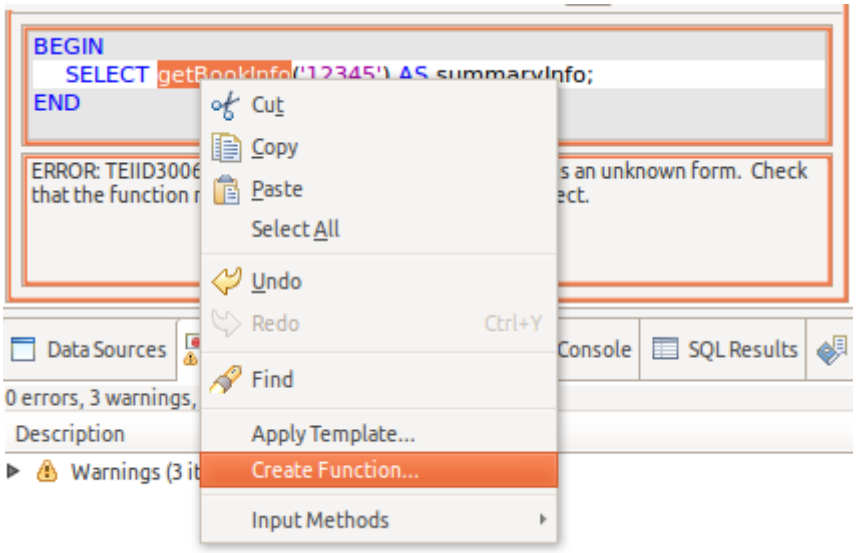


Figure 6.18. Create Function Action

You can then choose to create a Source Function (i.e. pushdown) or a User Defined Function option.

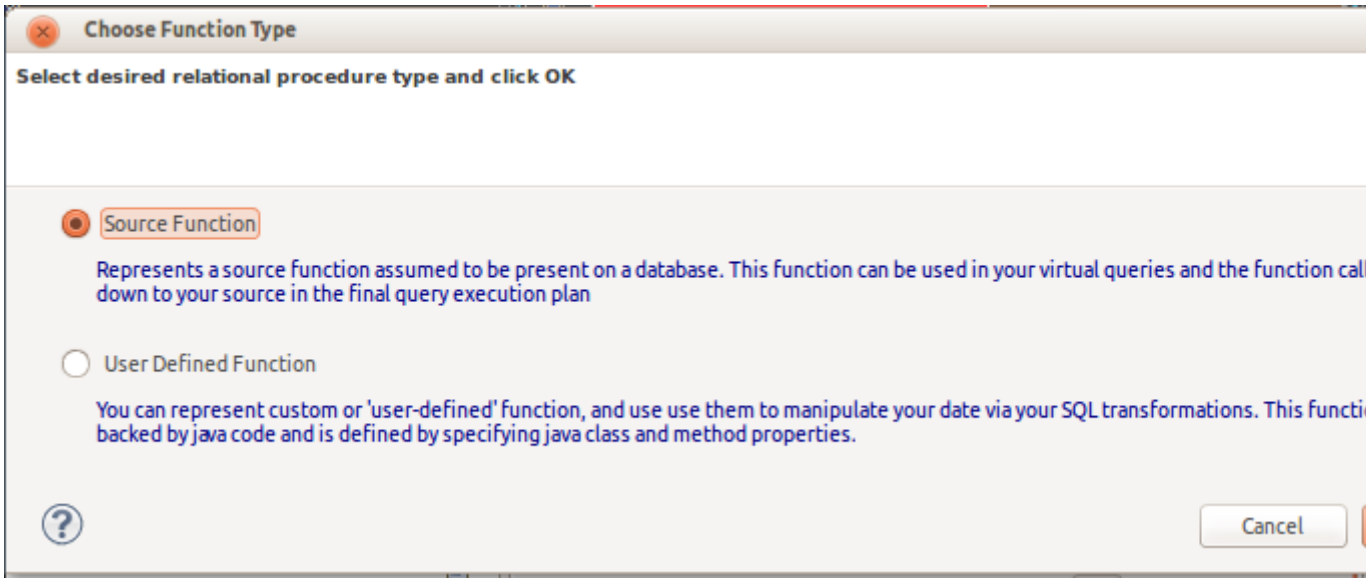


Figure 6.19. Select Function Type

For either option, you'll be presented the appropriate new function/procedure wizard which will seed the parameter definitions based on the procedure definition in the SQL.

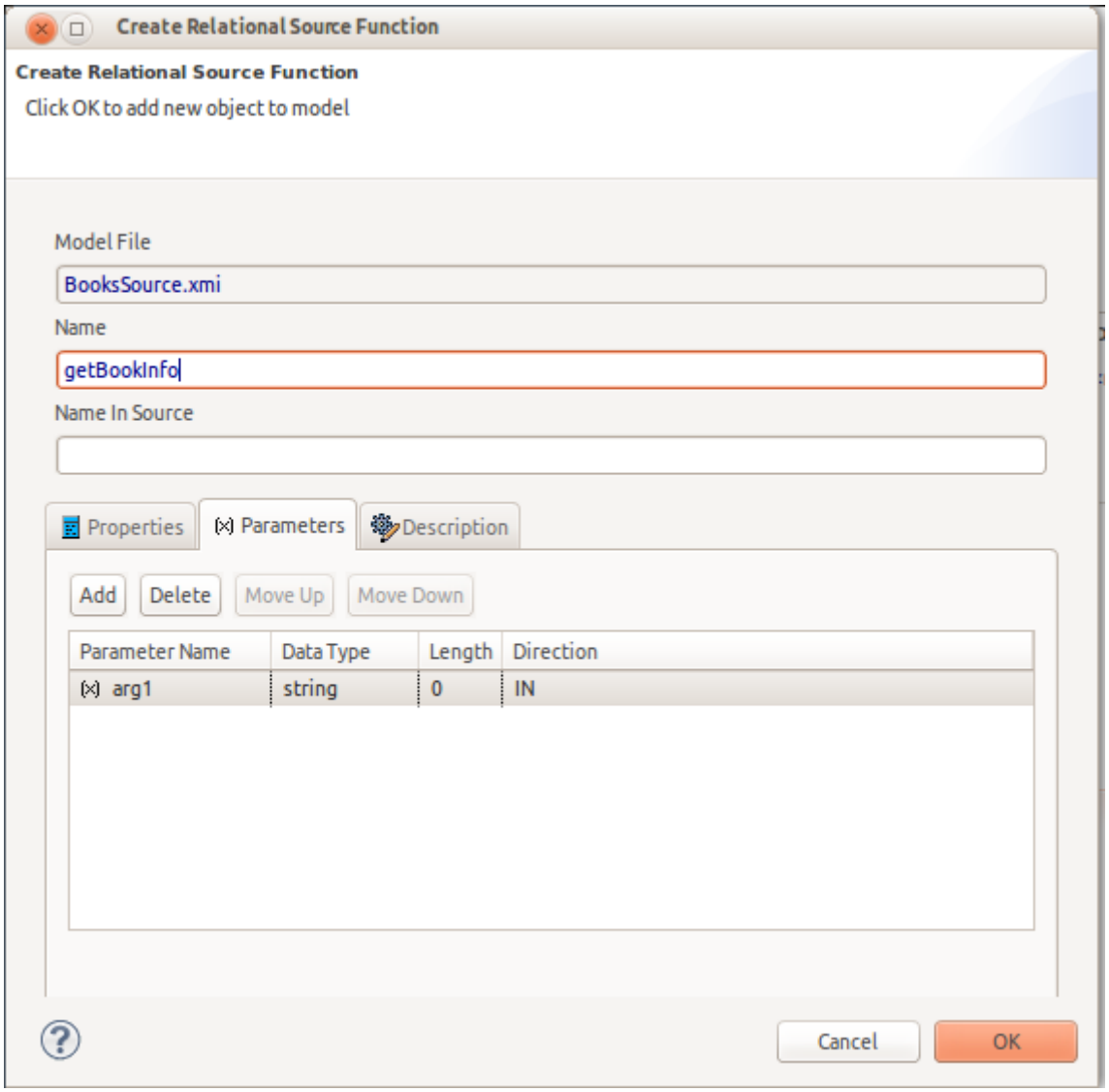


Figure 6.20. Create Source Function

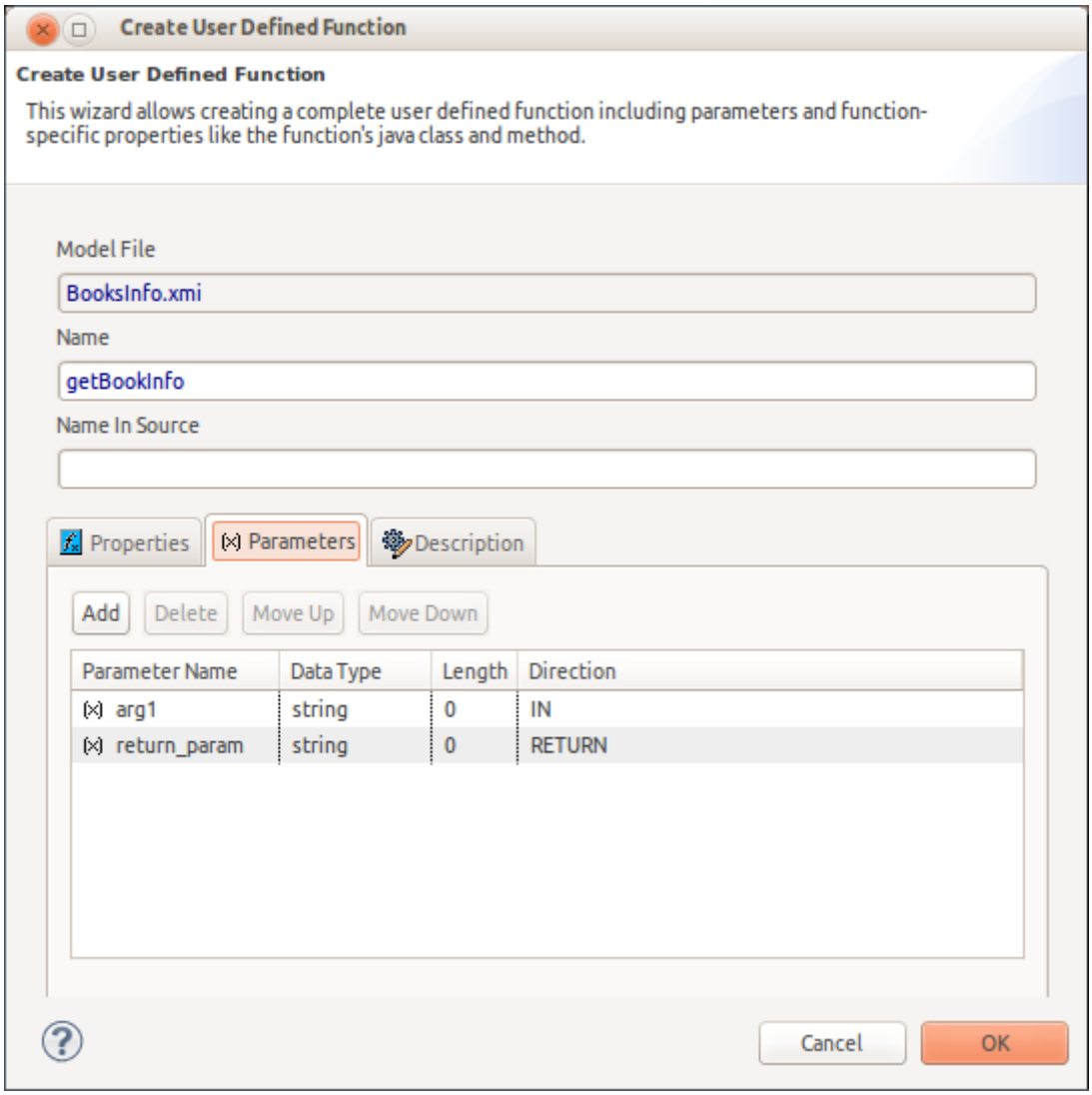


Figure 6.21. Create User Defined Function

In either case, you'll be prompted to select a target Source of View model as the location for your new function.

6.2.3. Create Relational Index Wizard

Indexes can be created at the same time as your relational table object (see Indexes tab in **Create Relational Table** image above) or via **New Child > Index...** action and dialog as shown below.

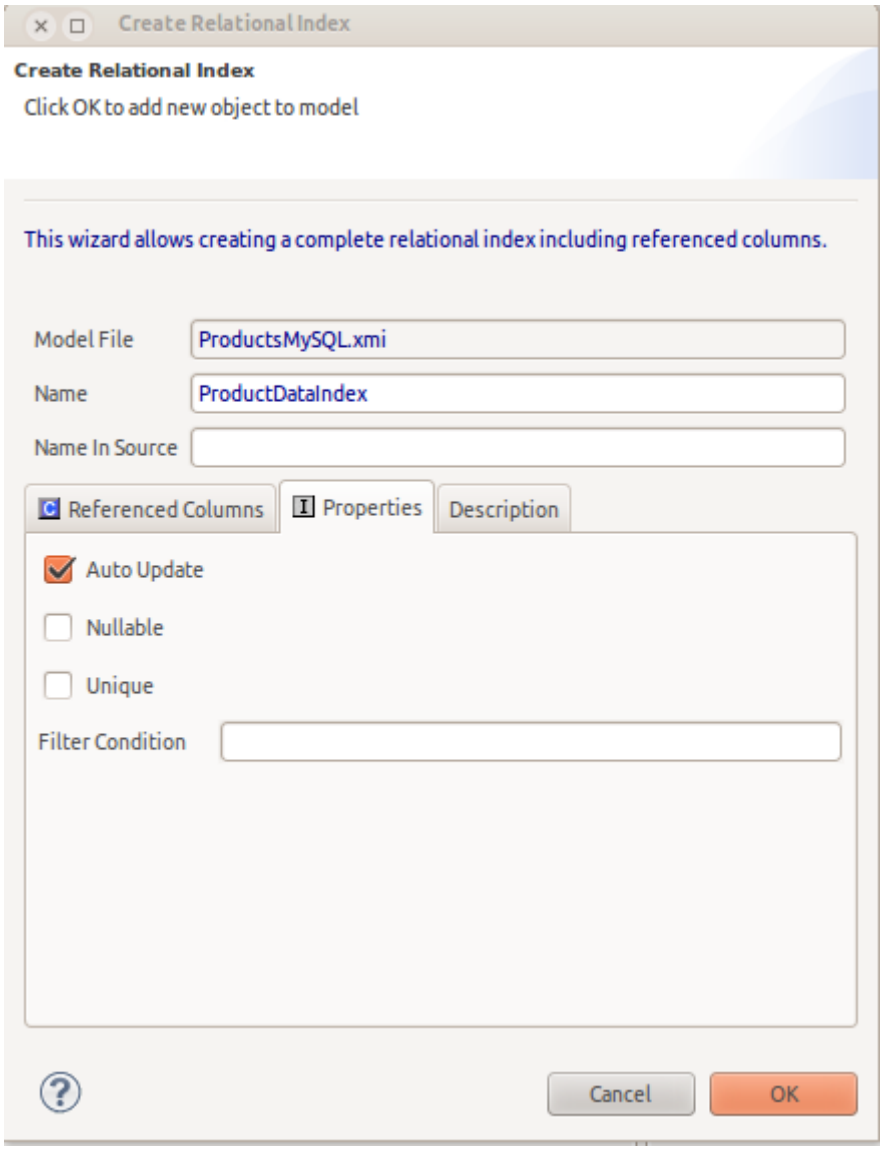


Figure 6.22. Create Relational Index Dialog

6.2.4. Create View Model Objects Wizards

For view models, only tables, procedures (standard procedures and user defined functions) and indexes can be created. For view tables and procedures, the primary difference in the wizards is that they include a SQL Transformation tab.

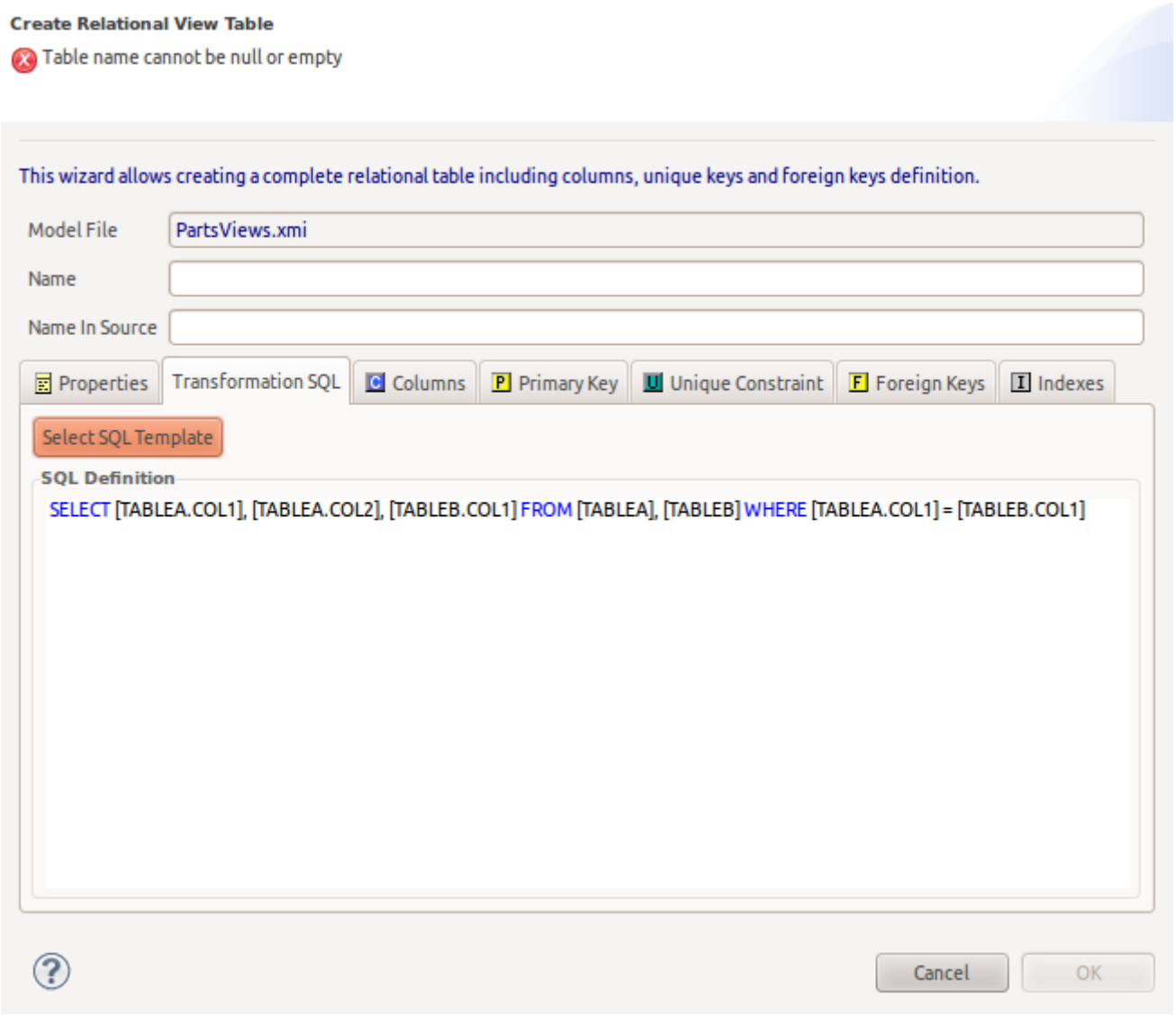


Figure 6.23. Create Relational View Table Dialog

6.2.4.1. Create User Defined Functions Wizard

User-defined functions can be defined as view model procedures that of type "function". When the **New Child > Procedure...** action is launched for a view model, the first dialog gives you the option of selecting the procedure type which includes either a standard view procedure or a user-defined function.

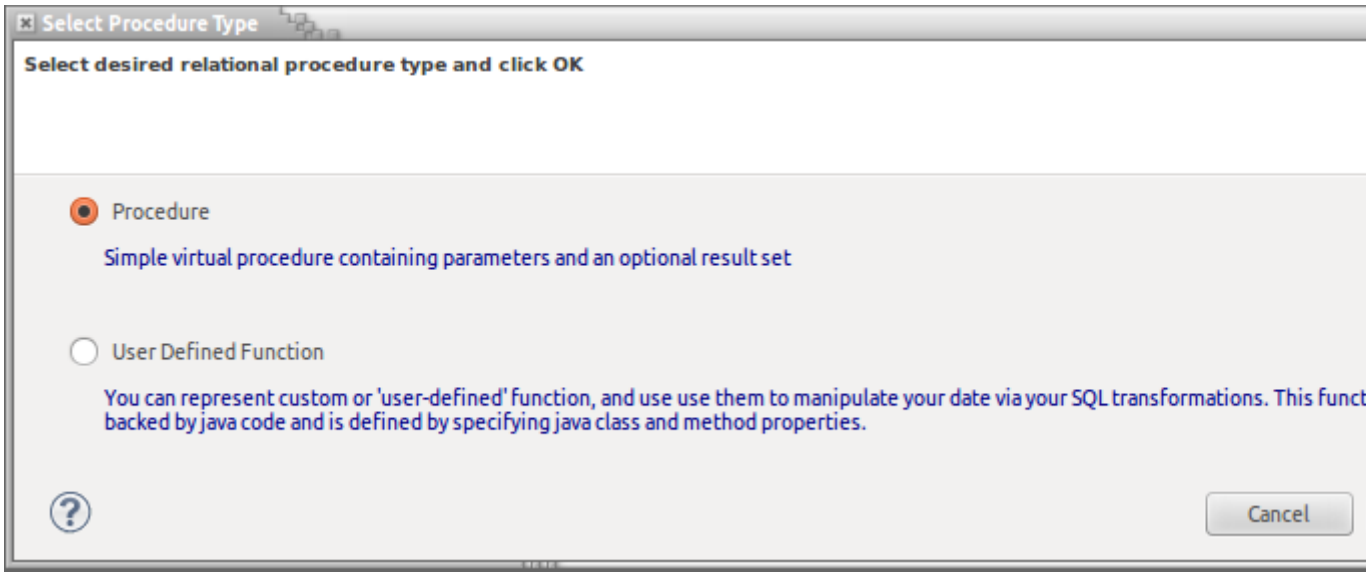


Figure 6.24. Select Procedure Type Dialog

User Defined Functions require additional properties such as **Java Class** and **Method** as well as a path to the jar file containing the code as shown below.

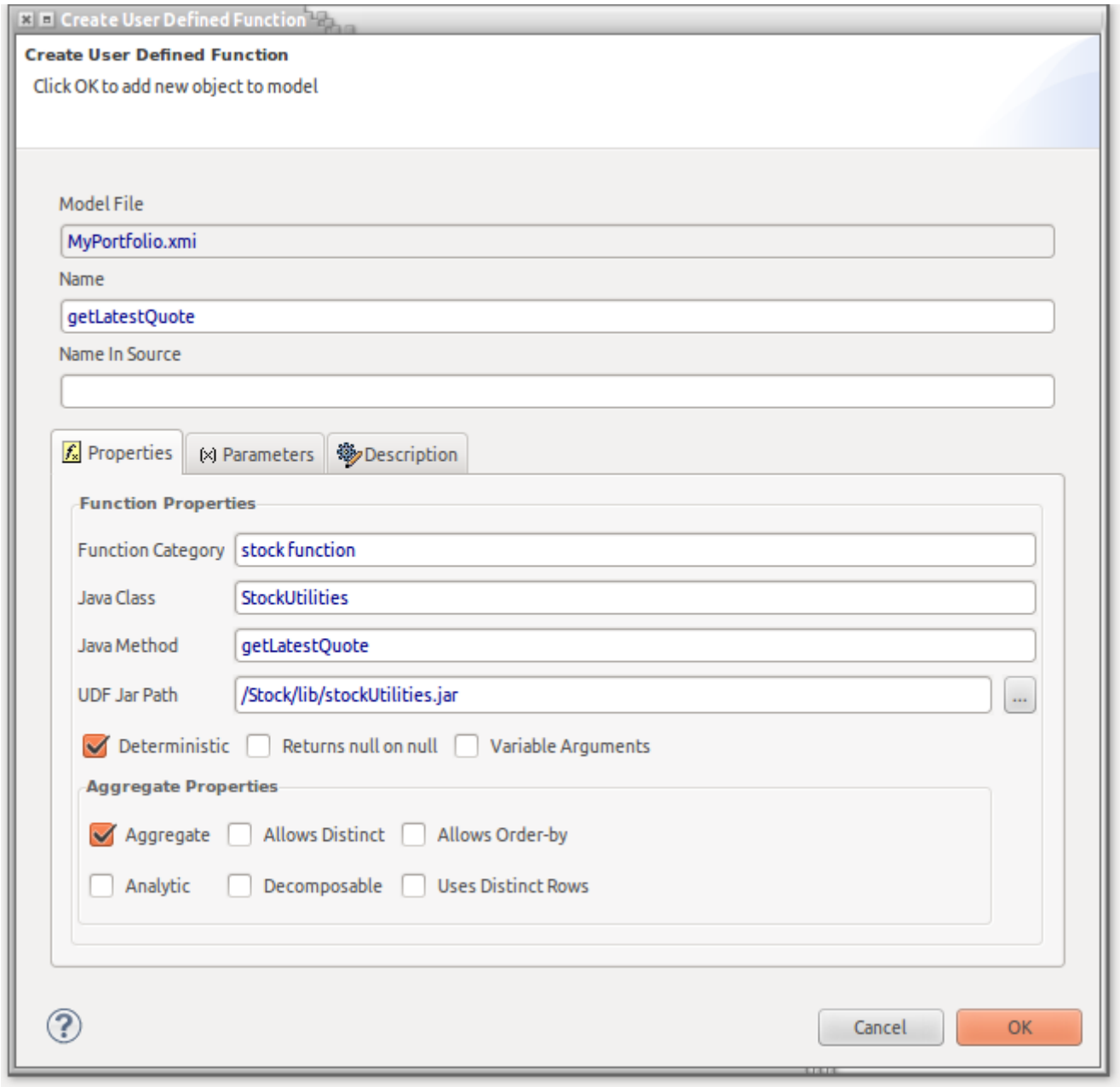


Figure 6.25. Create Relational User Defined Fuction Dialog

Note you can also create a User Defined Function from within a SQL transformation. See *Section 6.2.2.1, "Create Function from SQL Transformation"* for details.

6.3. 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 *Section D.2.1, "Model Explorer View"*, *Section D.3.1.2, "Diagram Editor"* and *Section D.3.1.3, "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:

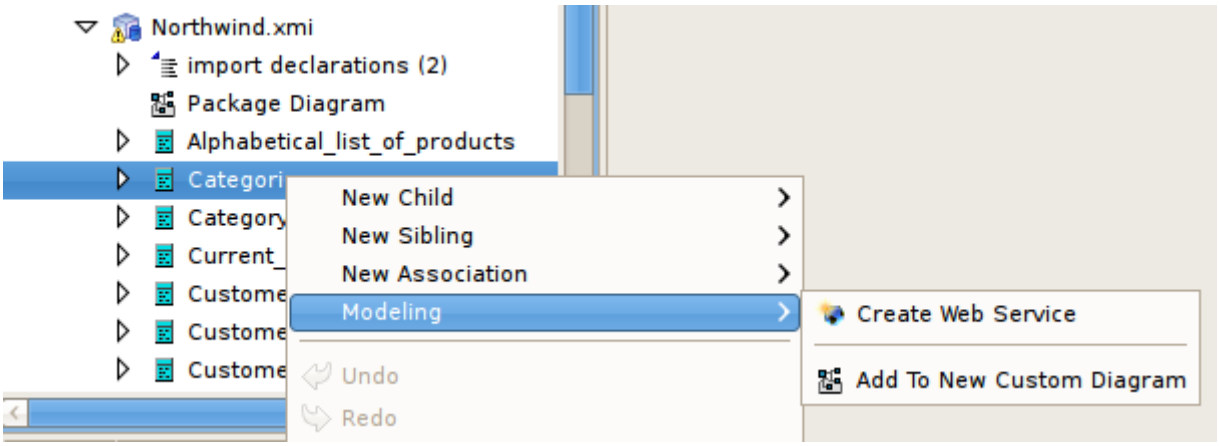


Figure 6.26. Modeling Sub-Menu for Source Table

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

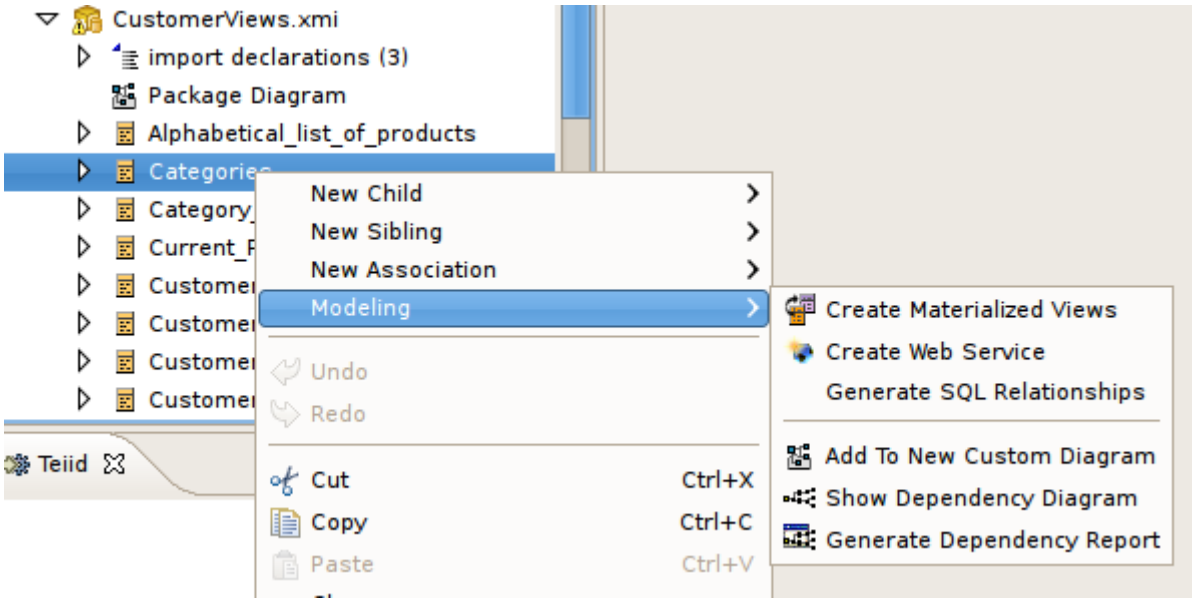


Figure 6.27. Modeling Sub-Menu for Source Table

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

- *Section 6.3.1, "Transformation Editor"*
- *Section 6.3.2, "Input Set Editor (XML)"*
- *Section 6.3.3, "Choice Editor (XML)"*
- *Section 6.3.4, "Recursion Editor (XML)"*
- *Section 6.3.5, "Operation Editor"*

This section describes these editors in detail.

6.3.1. 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.

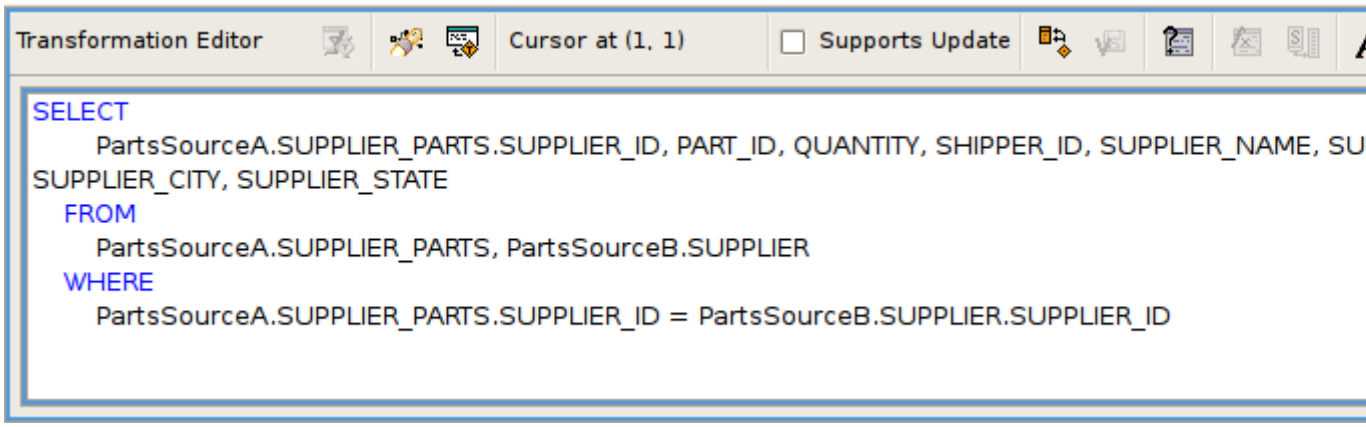


Figure 6.28. Editing String Property

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.
- Start your transformation with a provided SQL Template.
- 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 C.1.2.3, "Transformation Editor Preferences"*

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

  -

    Prevew 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. See *Section 6.3.1.1, "Using the Reconciler"* for more information about the Reconciler Dialog.

  -

    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. See *Section 6.3.1.3, "Using the Criteria Builder"* for further information.

-

  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. See *Section 6.3.1.4, "Using the Expression Builder"* for further information.

-

  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.

- The **Transformation Editor** context menu can be displayed by Rt-Clicking within the editor's text area. The context menu is show below:

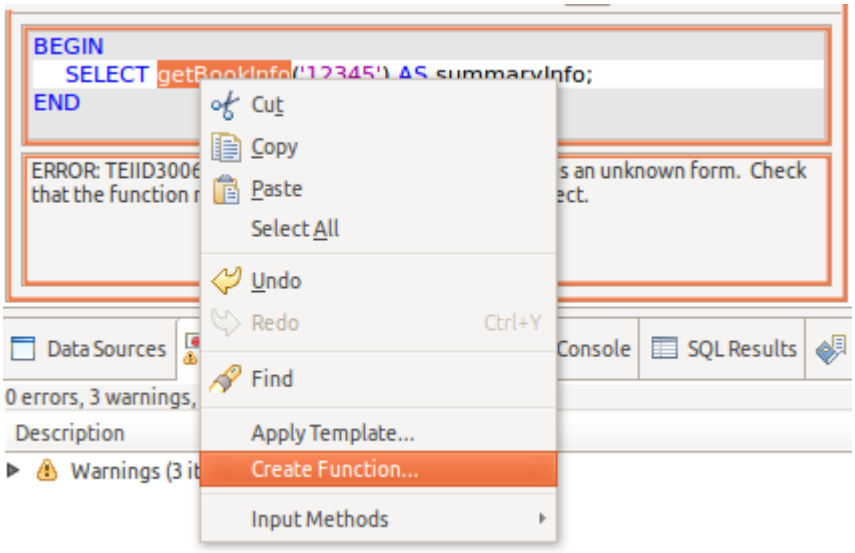


Figure 6.29. Transformation Editor context menu

Following is a summary of the context menu actions:

- **Cut - Copy - Paste** - Typical text editor actions to cut, copy or paste text within the editor.

- **Undo - Redo** - Allows you to Undo or Redo the previous action.

- **Find** - Displays a Find-Replace Dialog which allows you to search and replace text within the transformation.

- **Apply Template...** - Displays the 'Choose a SQL Template' Dialog, which allows you to choose a starting SQL Template from a list of common SQL patterns. See *???* for a description of this dialog.

- **Create Function...** - Allows creating a source function or user-defined fuction based on the selected function call defined in your SQL. See *Section 6.2.2.1, "Create Function from SQL Transformation"* for a description of this action.

6.3.1.1. Using the Reconciler

The Transformation Editor's **Reconciler** offers you a quick, graphical means to reconcile the Target View attributes and the Transformation SQL. As you make changes, the overall status will appear at the top of the dialog to assist you in successfully completing your edits.

To launch the Reconciler, click on the **Reconcile Transformation** button
in the Transformation Editor. The Reconciler Dialog is shown below:

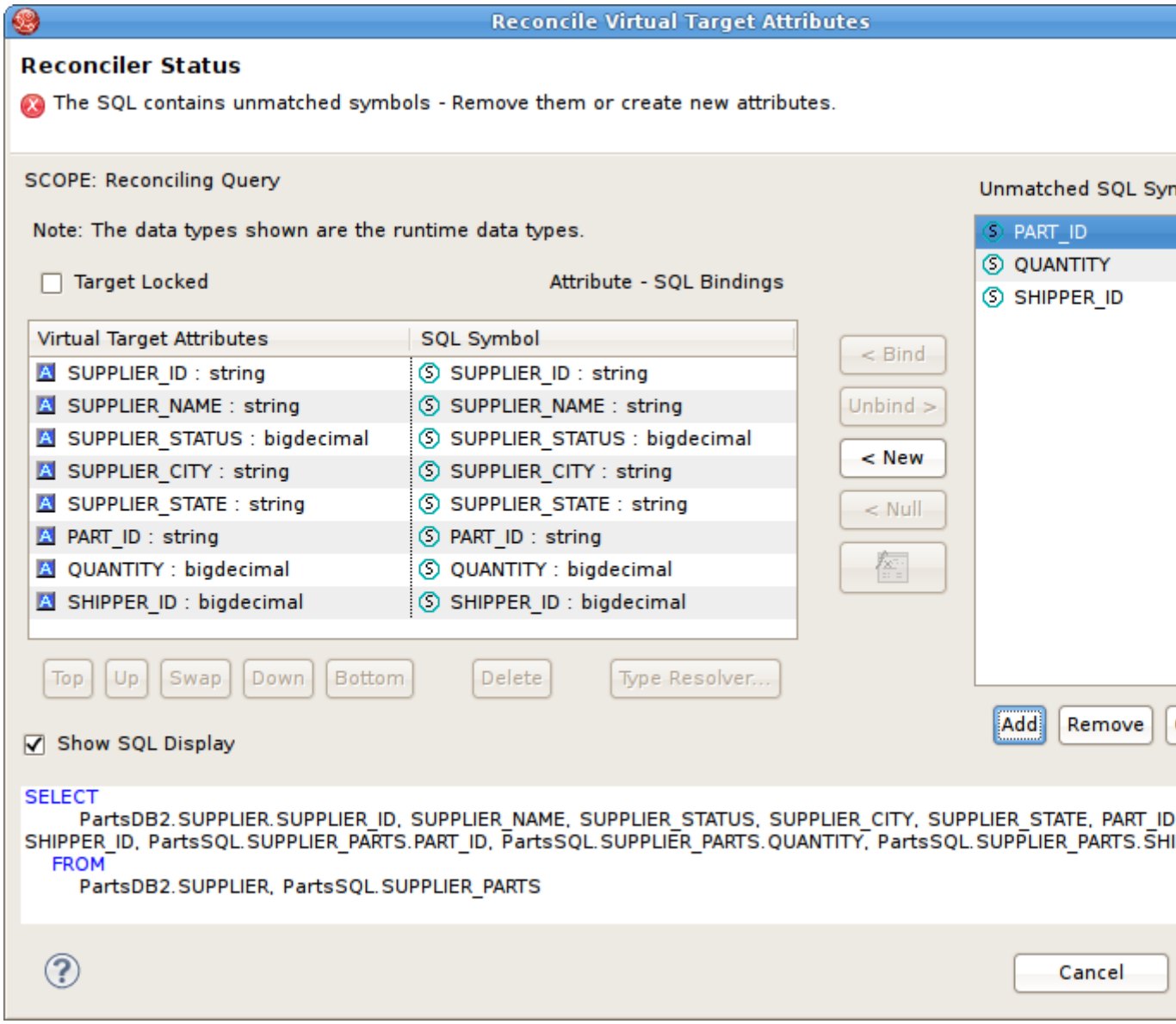


Figure 6.30. Reconciler Dialog

To summarize the different sections of the dialog:

- **Target Attributes - SQL Symbol Table**: This table shows the target attributes in the left column and the SQL Symbols in the right column. The SQL Symbols are the symbols that are 'projected' from the SQL transformation. A symbol is referred to as being 'bound' to a target attribute when it is displayed next to the attribute.

  If a target attribute is 'unbound', its row is highlighted in red. The transformation is not valid until all attributes have a corresponding SQL symbol binding.

  Here are a few things you can do in the table section:

  - **Lock Target Attributes**: To 'lock' the target attribute ordering, check the 'Lock Target Attributes' checkbox. This will lock the attributes in place.

  - **Re-Order Attributes**: To change the ordering of the target attributes, use the 'Top', 'Up', 'Swap', 'Down', and 'Bottom' controls beneath the table. Select or multi-select the table rows, then click the desired action button.

  - **Delete Attributes**: To delete one or more of the target attributes, select the table row(s) you want to delete - then click the 'Delete' button.

  - **Resolve Types**: If an Attribute-SQL Symbol binding has a datatype conflict, a message will be displayed. To assist in resolving the datatype conflict, a 'Datatype Resolver Dialog' is provided. Click on the table row, then click the 'Type Resolver...' button to display the dialog. See *Section 6.3.1.2, "Using the Datatype Resolver"* for further information.

- **Unmatched SQL Symbols list**: This list is to the right of the attribute-symbol binding table, and shows the SQL symbols from the transformation SQL that are not 'bound' to a target table attribute.

  Here are a few things you can do in the list section:

  - **Add SQL Symbols**: To 'Add' SQL Symbols to the list, click the 'Add' button. You will be presented with a dialog showing all available symbols from your transformation source tables. Click on the symbols you want to add, then click 'OK'.

  - **Remove or Clear Symbols**: To remove one or more of the SQL symbols, select the list items then click the 'Remove' button. To clear the entire SQL symbols list, click the 'Clear' button.

  - **Sort Symbols**: By default, the symbols are shown in the order that they appear in the SQL query. To show them alphabetically in the list, click the 'Sort' button.

- **Binding Controls**: The 'Binding Controls' are located between the Attribute-Symbol table and the Unmatched SQL Symbols list. Use these buttons to define the Attribute-Symbol bindings.

  Here are a few things you can do with the binding controls:

  - **Bind**: This button will 'Bind' a SQL Symbol to a target attribute. Select an Unmatched SQL symbol and select a target attribute, then click 'Bind' to establish the binding.

- **Unbind**: This button will 'Unbind' an Attribute-Symbol binding. Select an already-bound attribute in the table, then click 'Unbind'. The SQL Symbol will be released to the Unmatched Symbols list.

- **New**: This button will create a new target attribute, using an Unmatched SQL Symbol. Select an Unmatched Symbol from the list, then click 'New'. A new target attribute will be added to the bottom of the Attribute-Symbol table, bound to the selected SQL symbol.

- **Null**: This button allows you to bind 'null' to a target attribute instead of binding a SQL Symbol to it. Select a row in the Attribute-Symbol table, then click 'Null'. The target attribute will be bound to 'null'. If it was originally bound to a SQL Symbol, the symbol will be released to the Unmatched Symbol list.

- **Function**: This button allows you to define an expression instead of just a SQL Symbol for the binding. To define the expression, select a row in the Attribute-Symbol table, then click the 'Function' button. The Expression Builder Dialog will display, allowing you to define any type of expression. See *Section 6.3.1.4, "Using the Expression Builder"* for further information about the Expression Builder.

- **SQL Display**: The current transformation SQL is shown at the bottom of the reconciler dialog. As you add / remove SQL symbols and make other changes, you can see the SQL display change to reflect those changes. When you 'OK' the dialog, this SQL will be your new transformation SQL. If desired, the SQL Display can be hidden by un-checking the 'Show SQL Display' checkbox.

Once you are finished defining the bindings and resolving datatypes, click 'OK' to accept the changes. The transformation SQL will change to reflect your edits.

6.3.1.2. Using the Datatype Resolver

This dialog is accessible from the **Reconciler** dialog (See *Section 6.3.1.1, "Using the Reconciler"*) and offers you a quick way to resolve datatype conflicts between a target attribute and its SQL Symbol. The Datatype Resolver Dialog is shown below:

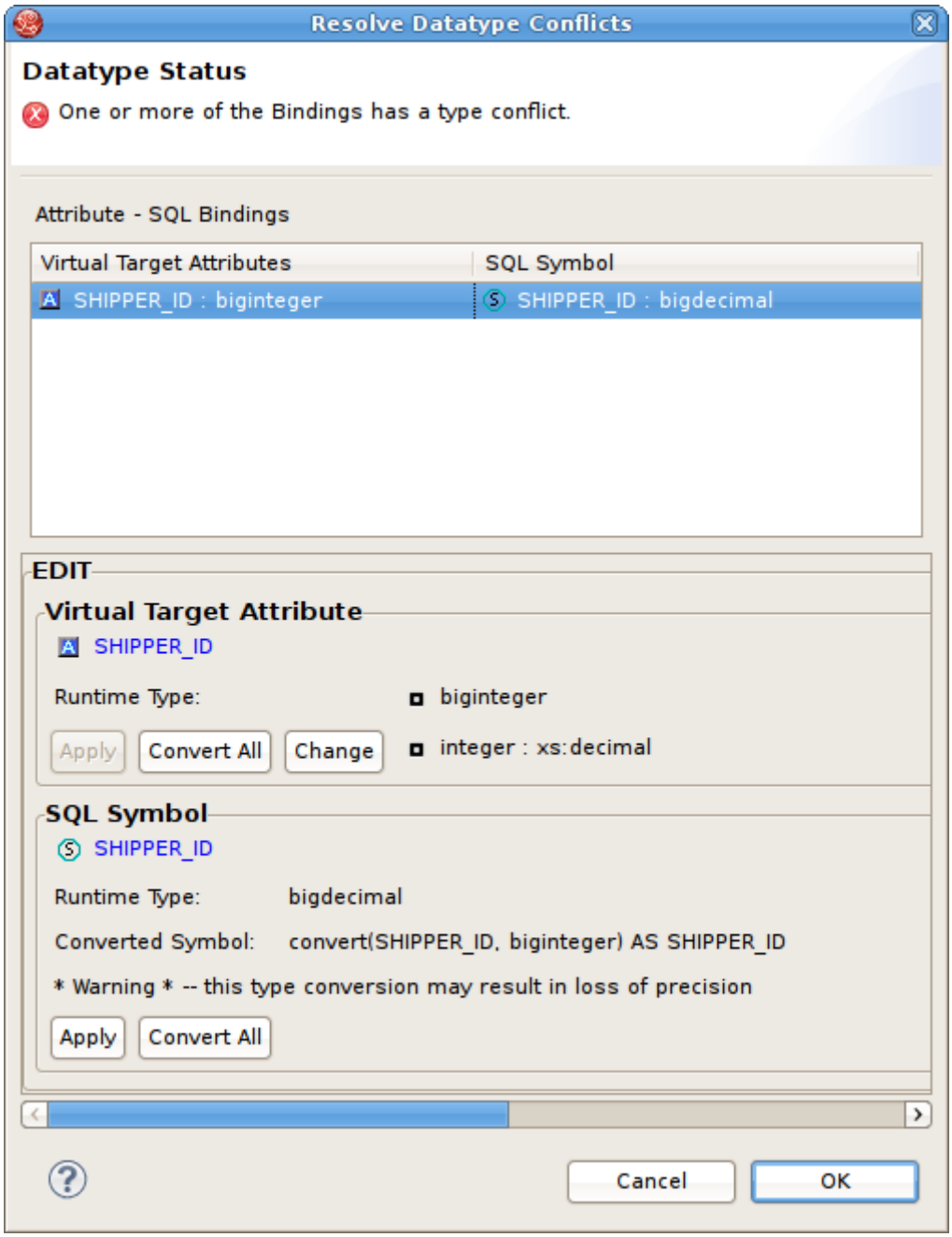


Figure 6.31. Datatype Resolver Dialog

To summarize the different sections of the dialog:

- **Target Attribute - SQL Symbol Table**: This table shows all target attribute - SQL Symbol bindings from the Reconciler Dialog which have a type conflict. Select on a table row to populate the lower Edit Panel
- **Edit Panel**: The lower panel shows the Target Attribute and SQL Symbol datatype information for the selected binding. You can resolve the conflict in one of the following ways:

- **Virtual Target Attribute**: Resolve the type conflict by changing the target attribute type to be compatible with the SQL Symbol type. The attribute's current runtime type is shown, along with a potential new datatype - and some button controls:
  - **Apply Button**: If the suggested datatype is acceptable, click 'Apply' to allow the attribute type to be changed.
  - **Convert All Button**: If you wish to change all of the attribute types in the table to be compatible with its corresponding SQL Symbol datatype, click the 'Convert All' button.
  - **Change Button**: If the suggested datatype is not acceptable, click 'Change' to choose your own datatype from a datatype dialog.
- **SQL Symbol**: Resolve the type conflict by applying a CONVERT function to the SQL Symbol, so that its type is compatible with the target attribute type. The SQL Symbol's current type is shown, along with a suggested CONVERT function - and two button controls:
  - **Apply Button**: If the suggested CONVERT function is acceptable, click 'Apply' to apply the CONVERT function to the SQL Symbol.
  - **Convert All Button**: If you wish to apply a CONVERT function to all of the SQL Symbols in the table so that their datatype is compatible with the corresponding attribute datateyp, click the 'Convert All' button.

Once you are finished resolving datatypes, click 'OK' to accept the changes. You are directed back to the Reconciler Dialog, which will be updated to reflect your edits.

6.3.1.3. 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.

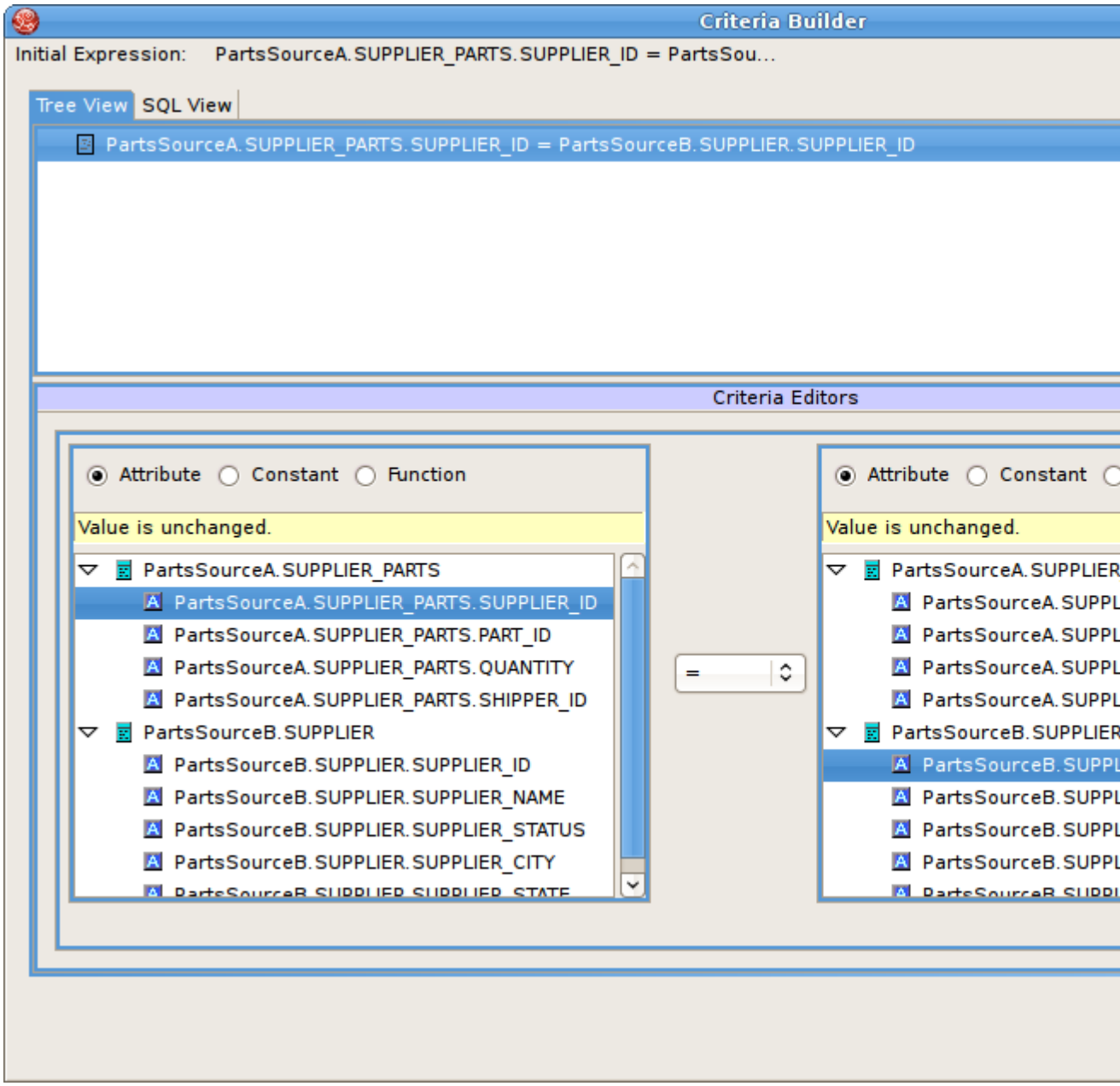


Figure 6.32. Editing String Property

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:

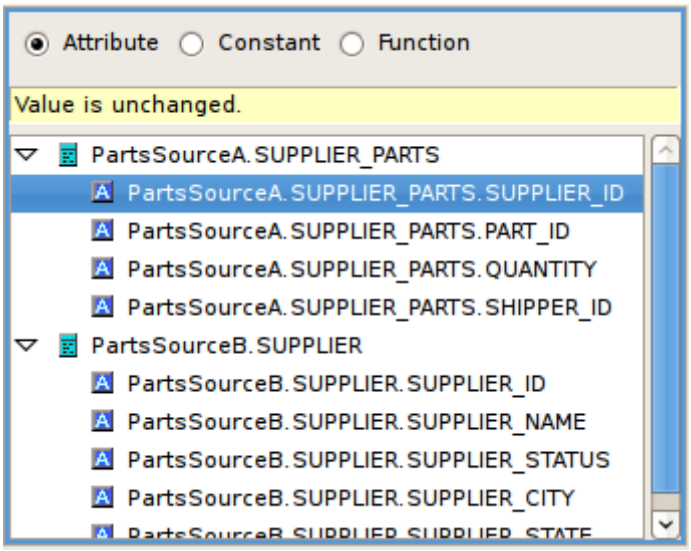


Figure 6.33. Attribute Panel

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:

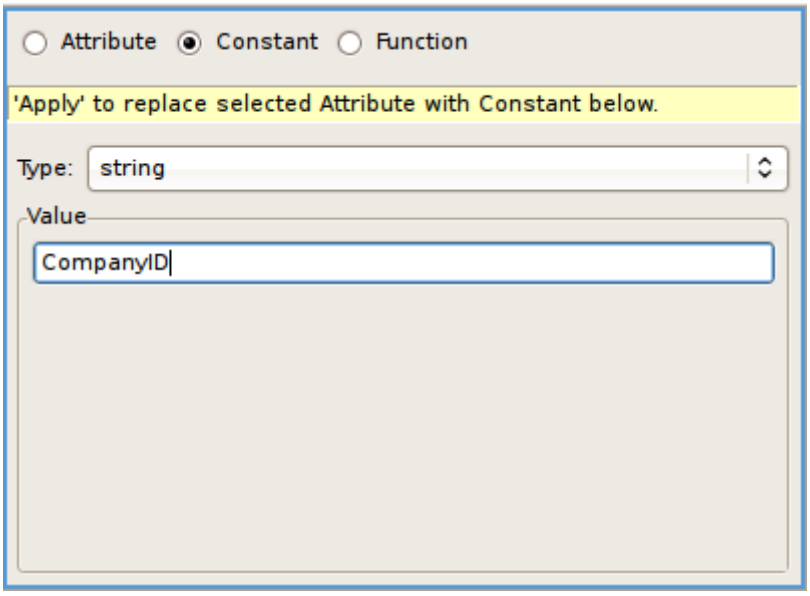


Figure 6.34. Constants Panel

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:

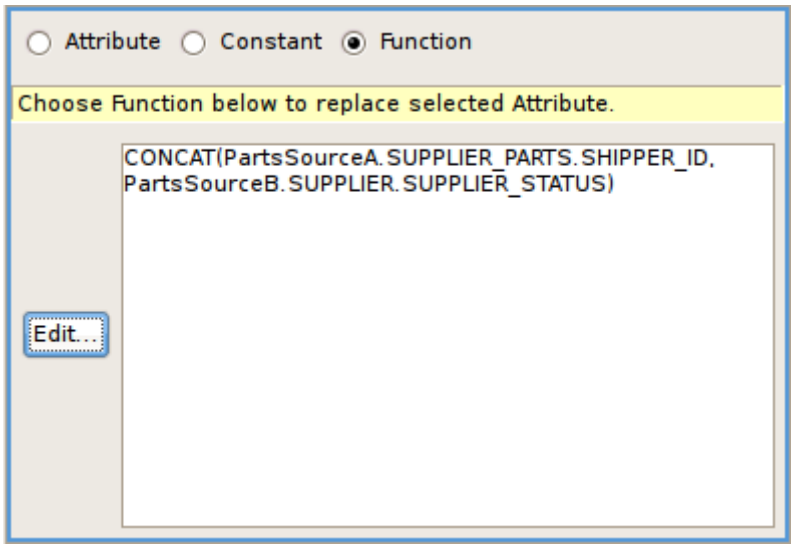


Figure 6.35. Functions

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 6.3.1.4, "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.

6.3.1.4. 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.

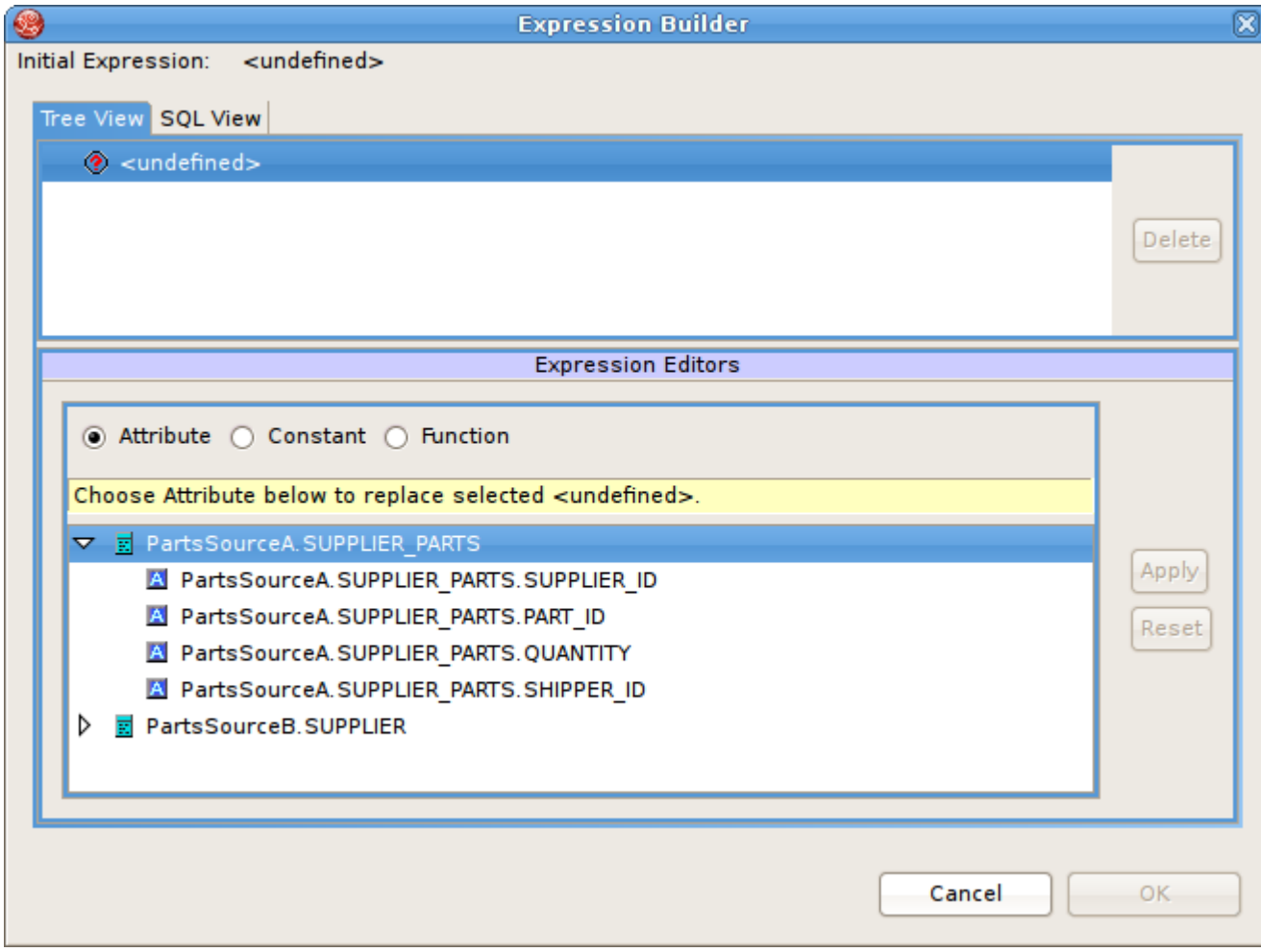


Figure 6.36. Expression Builder

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.

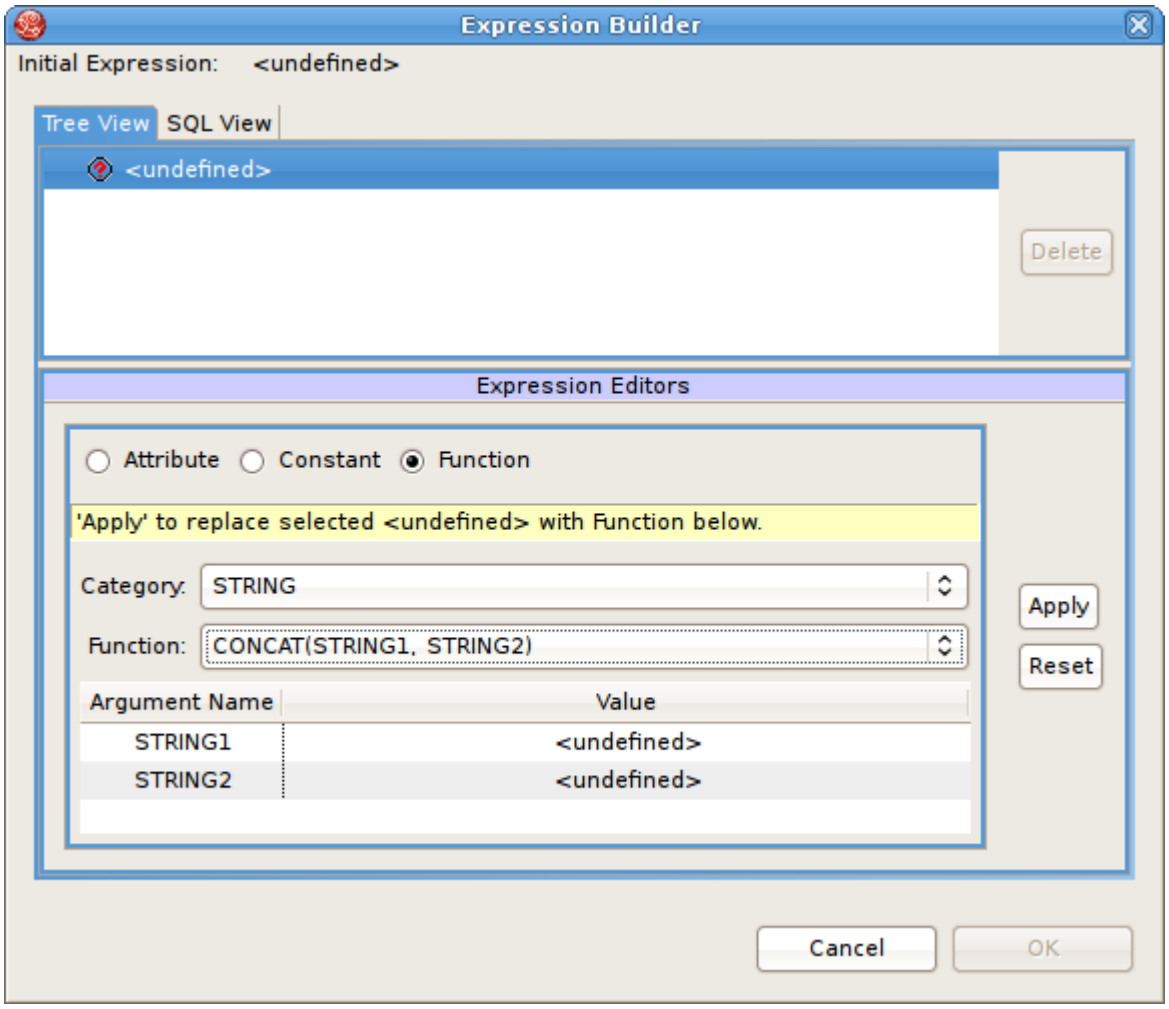


Figure 6.37. Function Panel Selected

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.

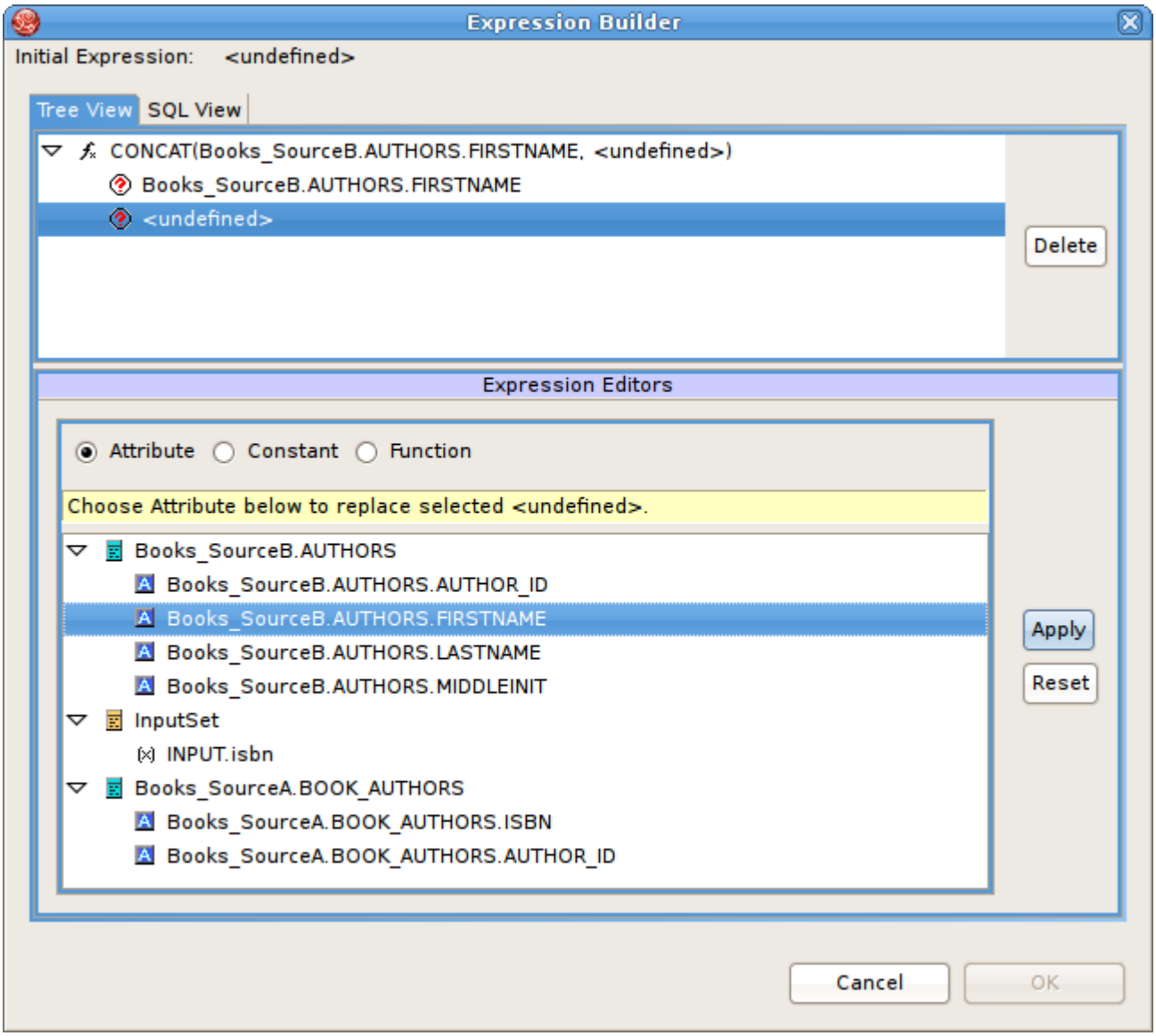


Figure 6.38. New Blank Function Created

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.

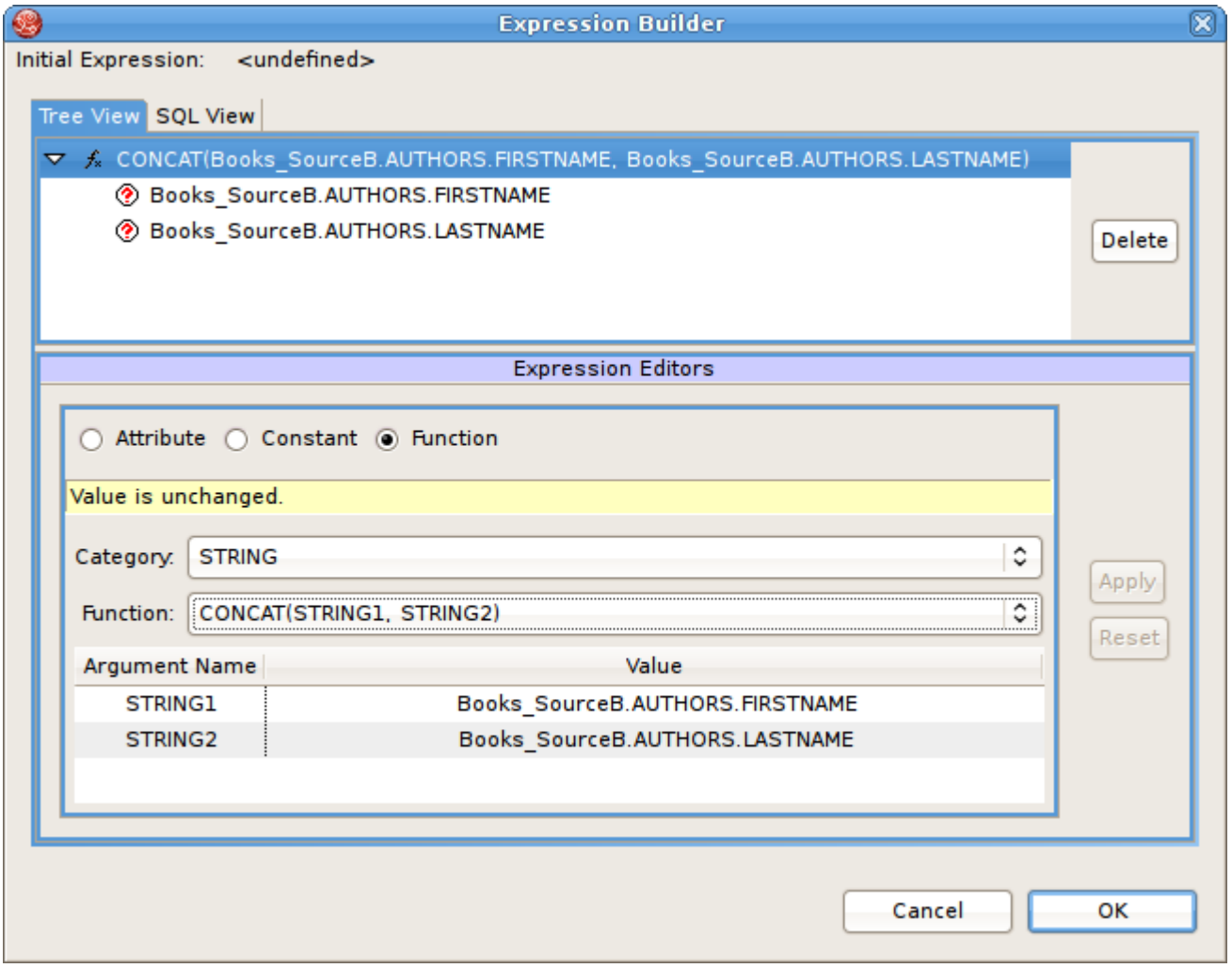


Figure 6.39. Nested Function Example

- **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.

6.3.2. 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, 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 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 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 *Section D.2.1, "Model Explorer View"* 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)

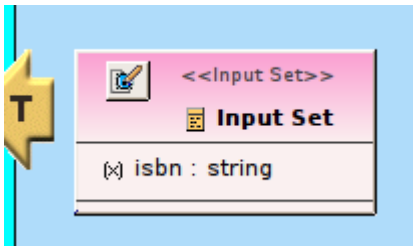


Figure 6.40. Edit Input Set Button

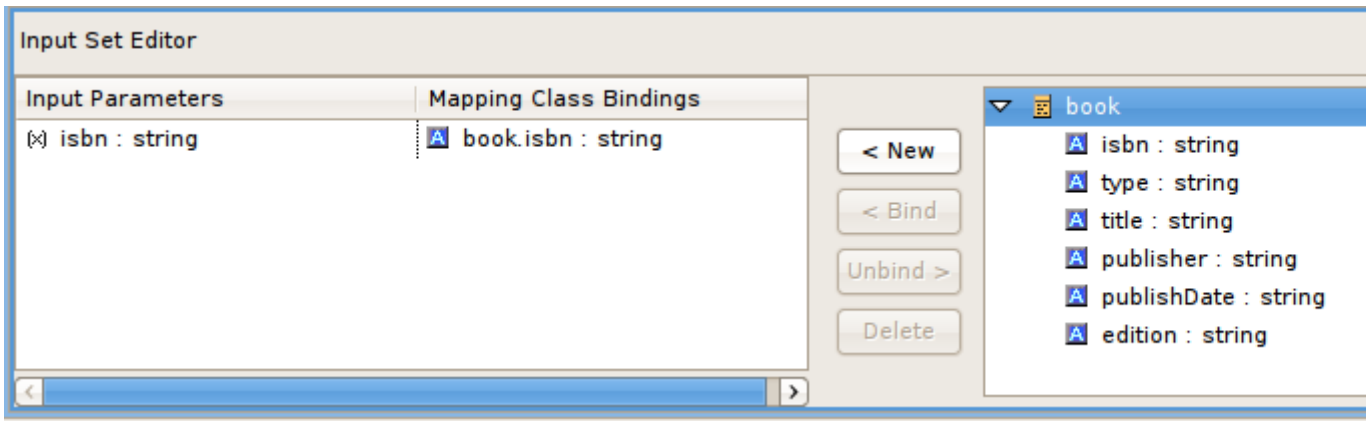


Figure 6.41. Input Set Editor Panel

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.

6.3.3. 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 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.

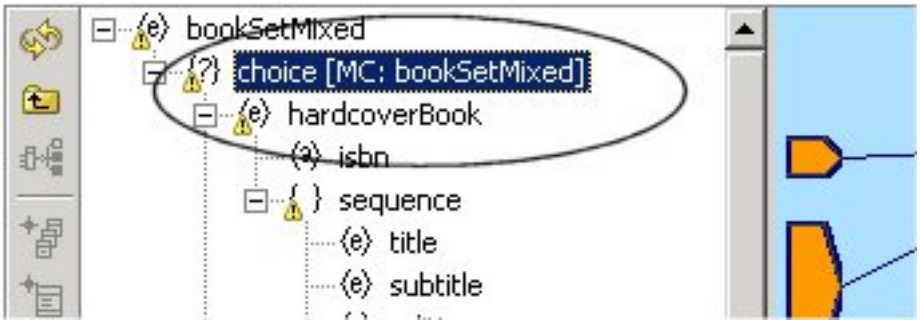


Figure 6.42. Opening The Choice Editor

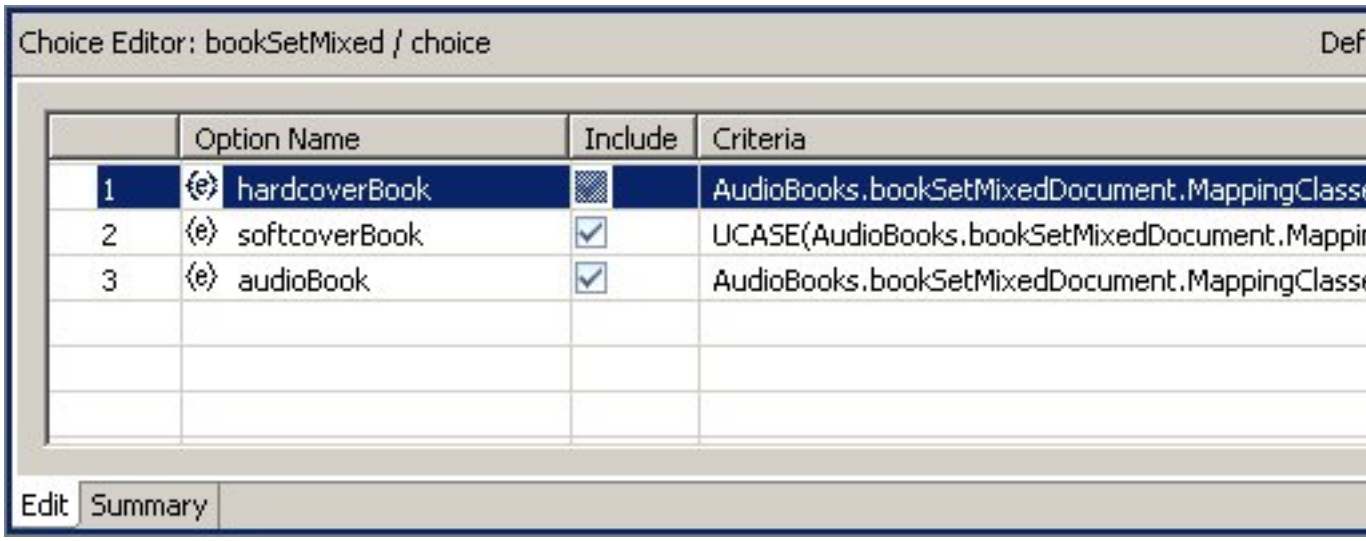


Figure 6.43. The Choice Editor

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.

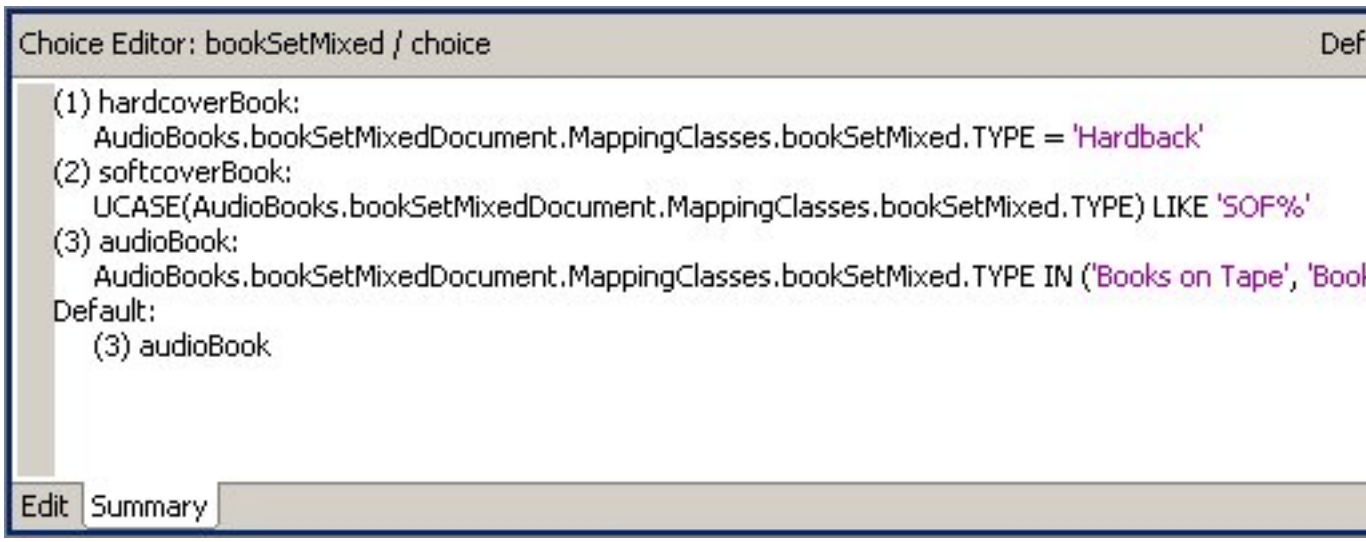


Figure 6.44. Choice Criteria Summary

6.3.3.1. Using the Choice Editor

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

- **Specify a criteria** statement for the Teiid 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.

6.3.3.2. 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 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.

6.3.3.3. 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 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.

6.3.3.4. Setting Choice Element Order

To edit the criteria for a choice element:

The Teiid Server evaluates the choice criteria in the order in which they appear, and when one choice criteria is met, the Teiid Server populates the XML instance document with that option. The Teiid 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 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.

6.3.3.5. Setting a Default Choice Action

The default action represents the course the Teiid 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 Server exception.
- **RECORD** to record the Teiid Server exception.
- **DISCARD** to place no element within the XML instance document.

Note

A default action for the choice criteria must be set.

6.3.4. 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**, 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 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, Teiid Designer does not mark the recursive fragments in document models to execute recursively in the Teiid Server. To take advantage of this behavior, you must open the **Recursion Editor** in the recursive mapping class *Section D.3.1.2.5, "Mapping Transformation Diagram"*, mark the transformation query as recursive, and specify the recursion limit properties.

6.3.4.1. The Recursion 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.

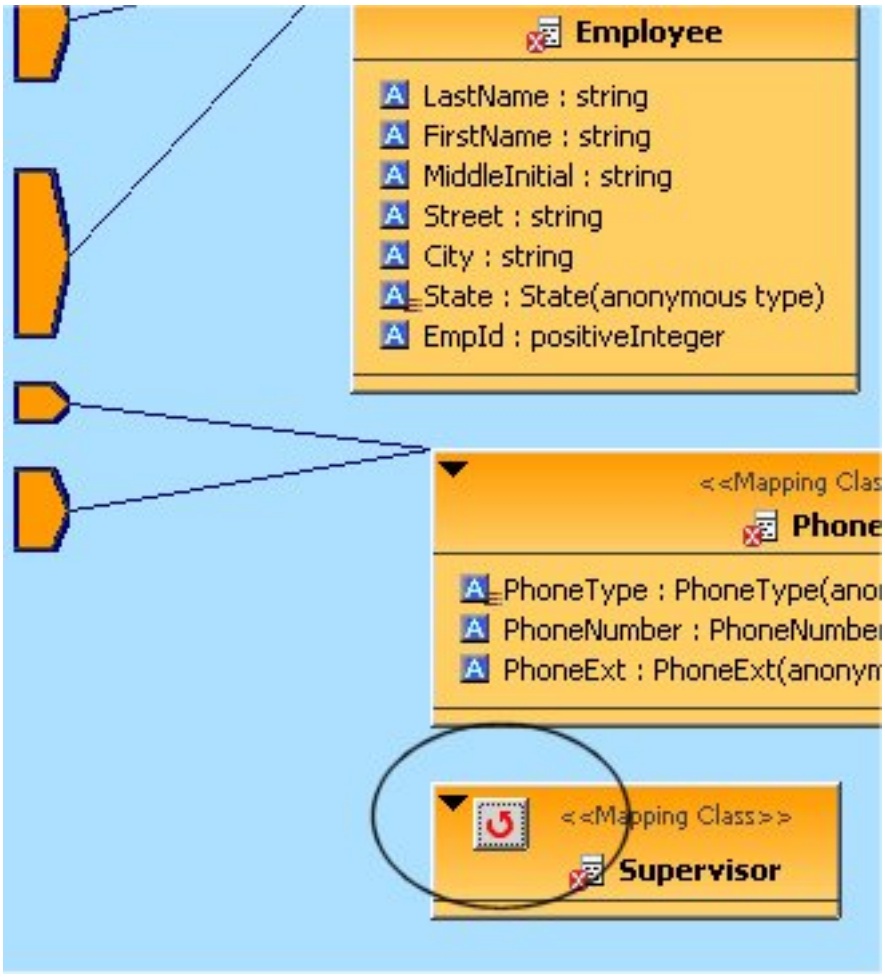


Figure 6.45. Open Recursion Editor Button

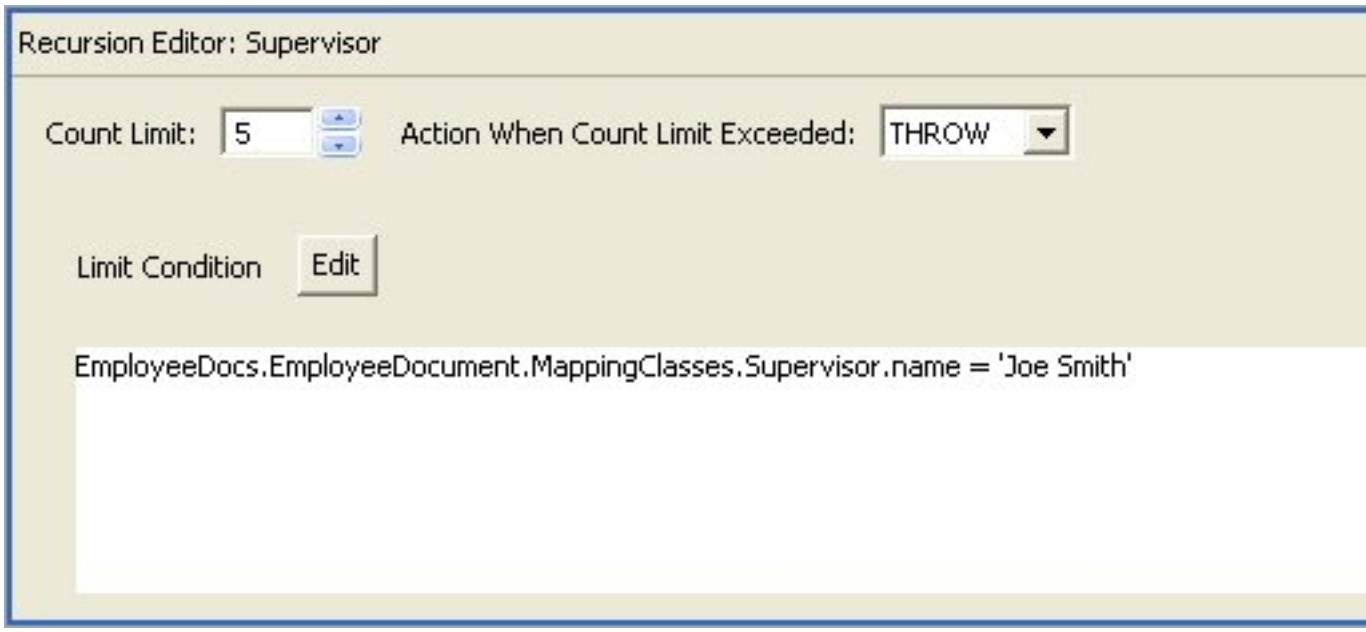


Figure 6.46. Recursion Editor

- To edit recursion properties:
  - **Step 1** - Click the **Enable Recursion** check box if you want the Teiid 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 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 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 6.3.1.3, "Using the Criteria Builder"*** to build a limiting condition for this recursion.

Note

The Teiid Server will evaluate this condition each time it recursively performs this query. If this criteria clause evaluates false, the Teiid Server performs the query recursively again unless it has reached the **Count Limit**. If the criteria evaluates true, the Teiid 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 Server dynamically populates your XML documents at runtime, it will use the recursion specifications you entered here.

6.3.5. 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.

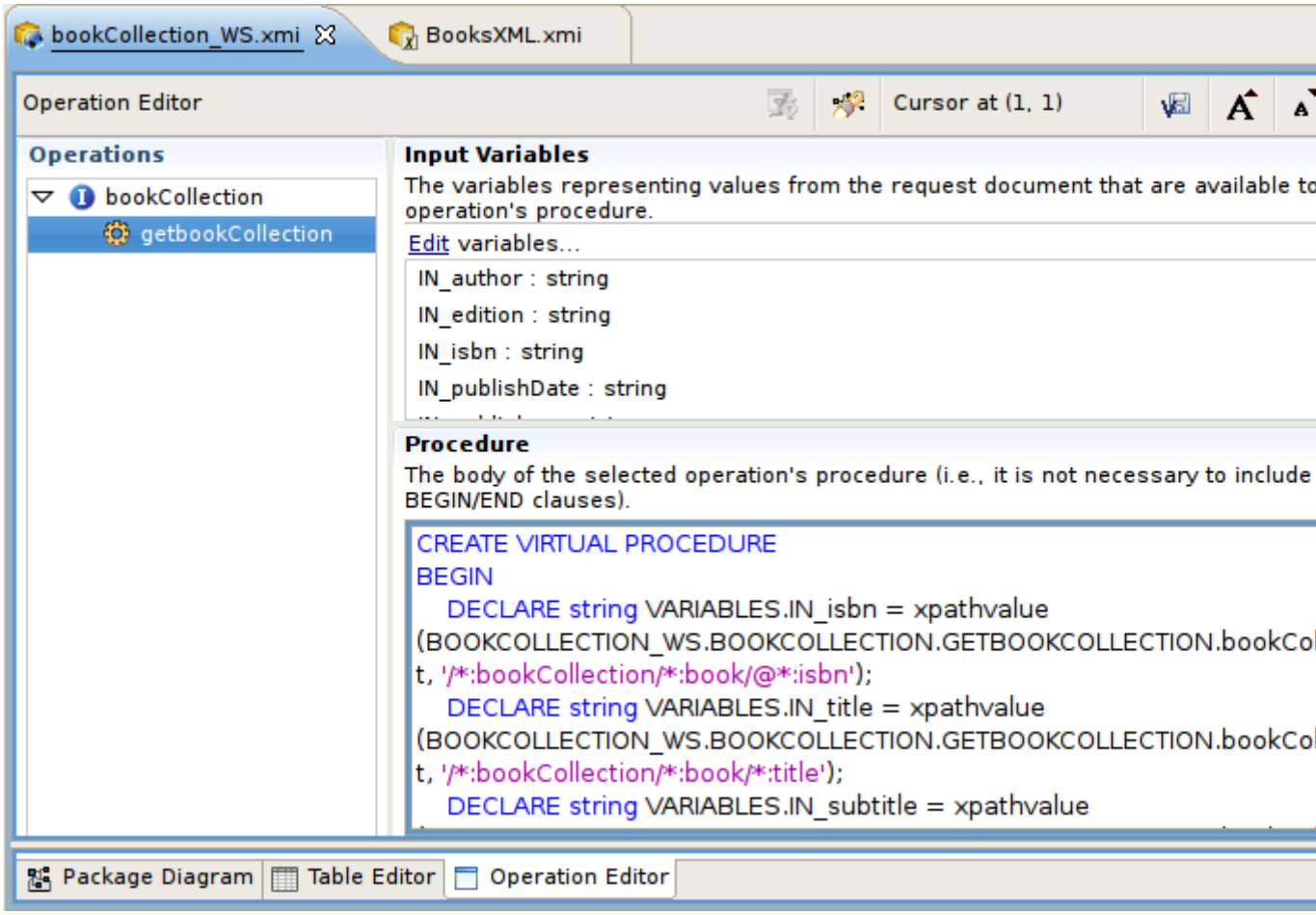


Figure 6.47. Operation Editor

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:

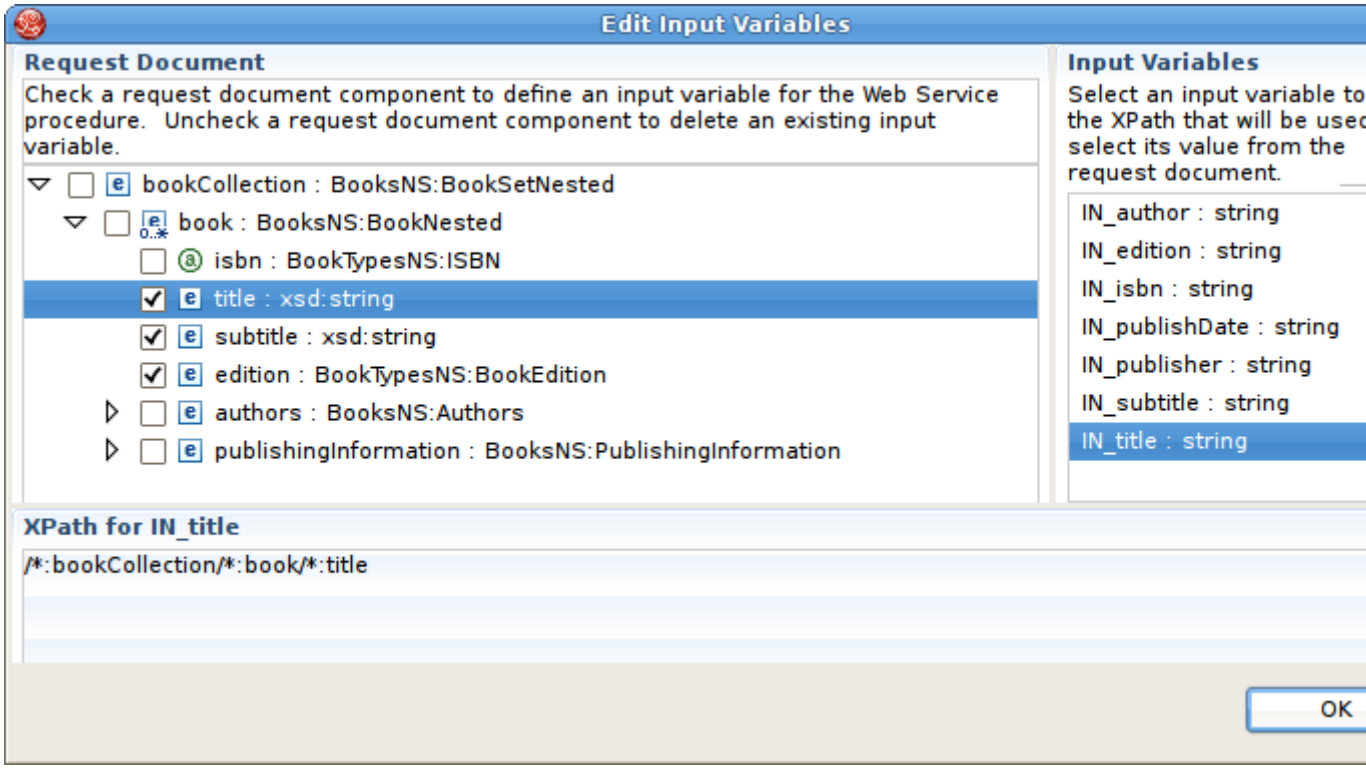


Figure 6.48. Edit Input Variables Dialog

You can create new operations in the *Operation Editor* by right-clicking an interface or operation and selecting the *Create Operation* action. This will display a dialog for entering the definition for your new operation.

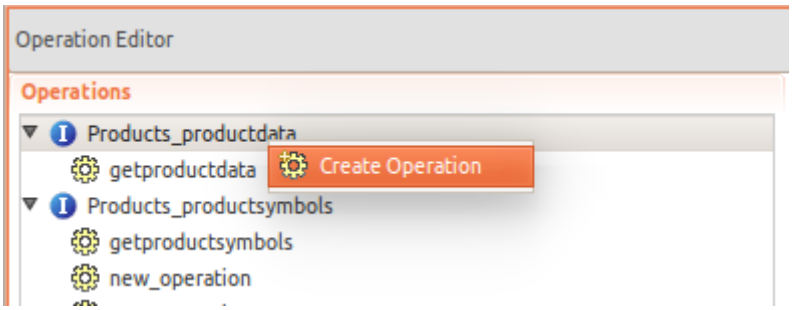


Figure 6.49. Create Operation Action

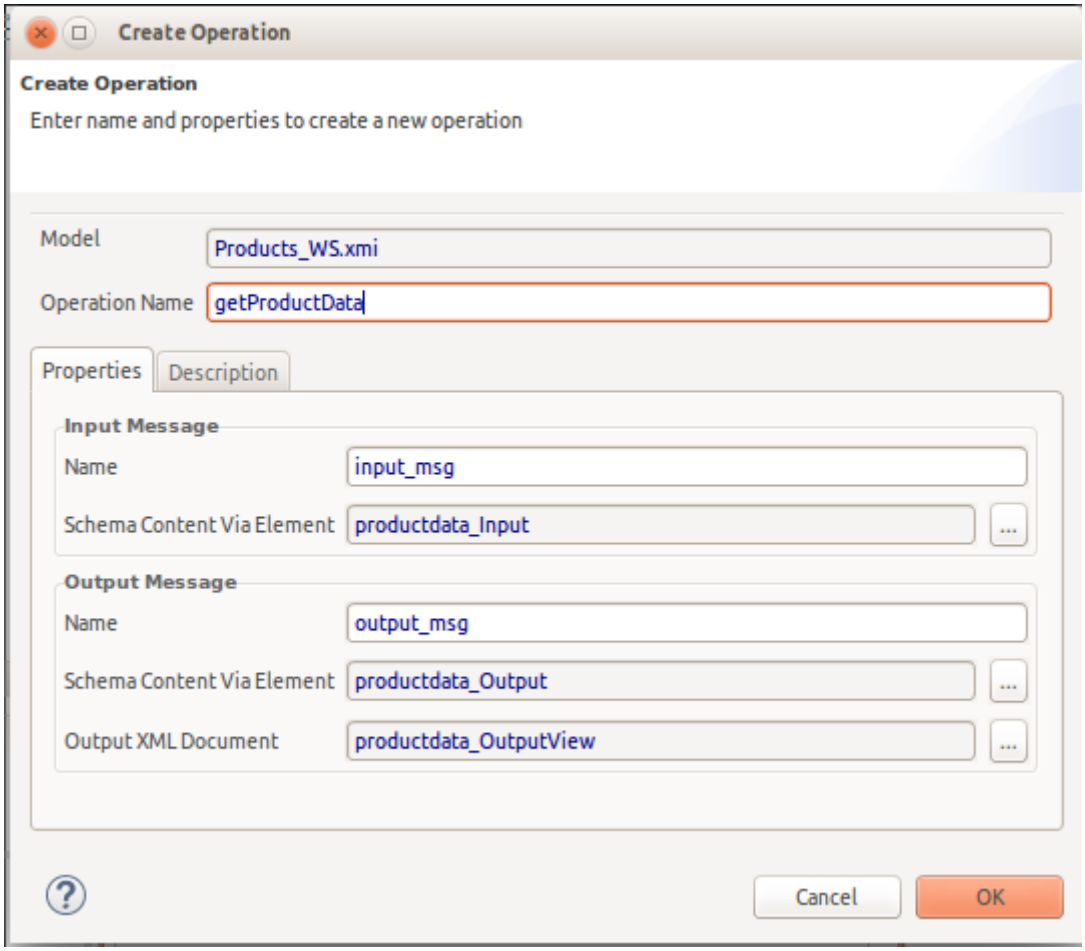


Figure 6.50. Create Operation Dialog

6.4. Managing Model Object Extensions

Extending a model adds extra properties to its model objects. One good use of these extension properties is for passing data to a customized Teiid translator. The Designer model extension framework consists of:

- **Model Extension Definitions (MEDs)** (See *Section 1.3.9, "Model Object Extensions"*)

- **MED Registry** -keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. (See *Section D.2.13, "Model Extension Definition Registry View (MED Registry View)"*)
- **MED Editor** (See *Section D.3.3, "Model Extension Definition Editor"*)

6.4.1. Create New MED

To create a new MED select the **File > New > Other...** action to display the New wizard dialog. Select the **Teiid Designer > Teiid Model Extension Defn** option which displays the **New Model Extension Definition** dialog. Browse and select existing project or project folder location for MED file and specify unique file name and press **Finish**.

Note

If a project is already selected when wizard is launched, the location field will be pre-populated).

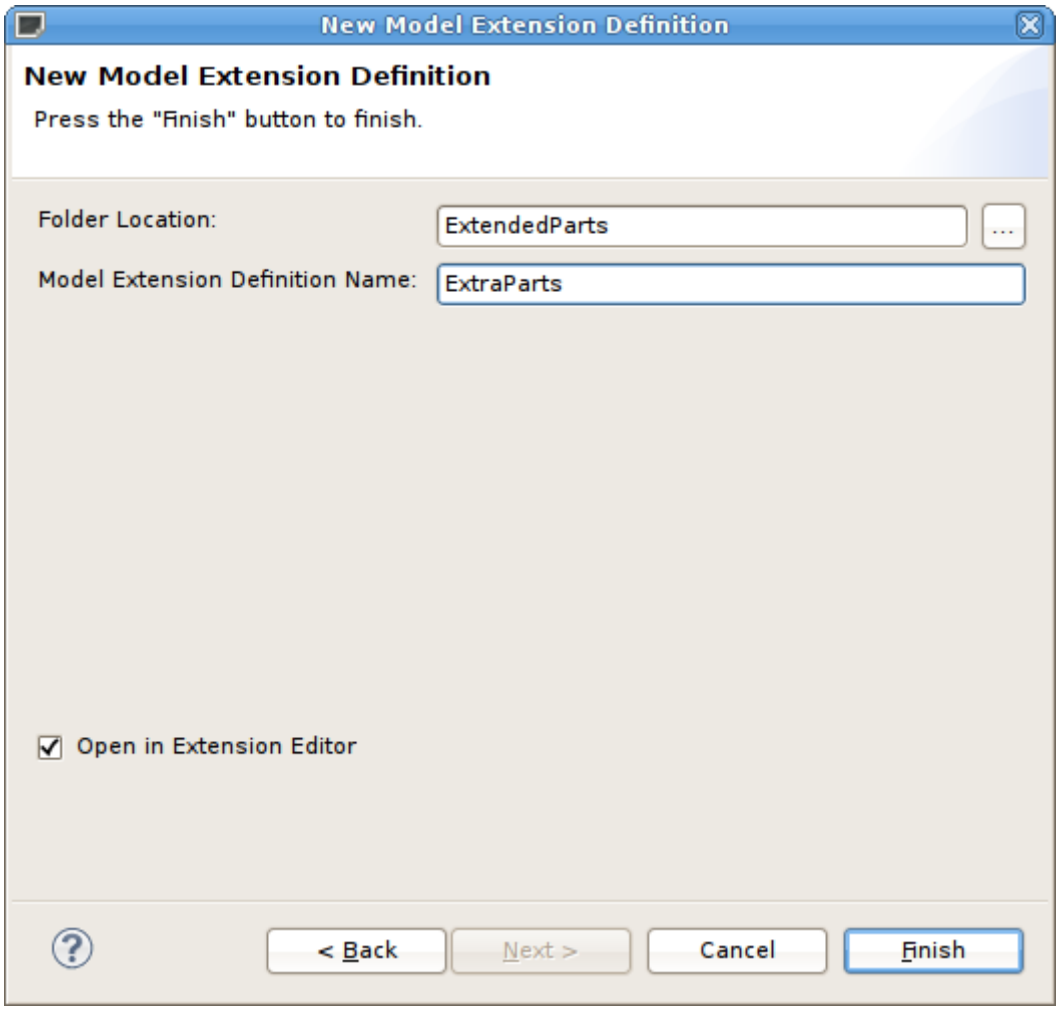


Figure 6.51. MED Editor Overview Tab

6.4.2. Edit MED

To edit an MED file select an existing '.mxd' file in your workspace and right-click select the **Open** action. The MED Editor will be opened to allow editing (See *Section D.3.3, "Model Extension Definition Editor"*).

On the Overview tab, you can specify or change the **Namespace Prefix**, **Namespace URI**, the **Model Class** you wish to extend (Relational, Web Service, XML Document, and Function) and a description. Note that version number is available, but is not currently being used.

After entering the basic MED info, you can now switch to the **Properties** tab and begin creating your extended property definitions for specific model objects supported by selected model class.

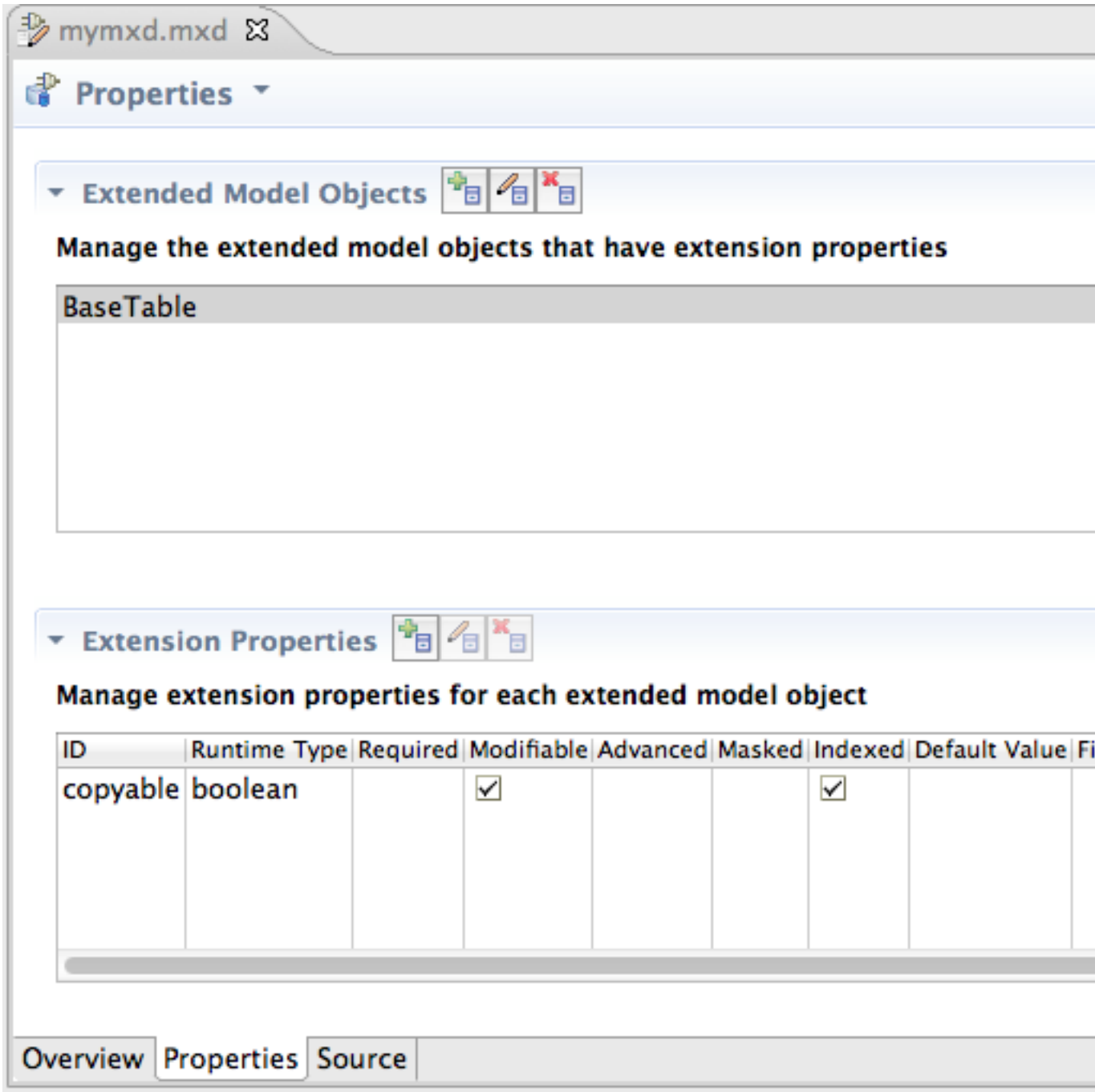


Figure 6.52. MED Editor Properties Tab

Start by selecting the **Add Extended Model Object** toolbar button to display the **Model Object Name** selection dialog. Select an object and press **OK**.

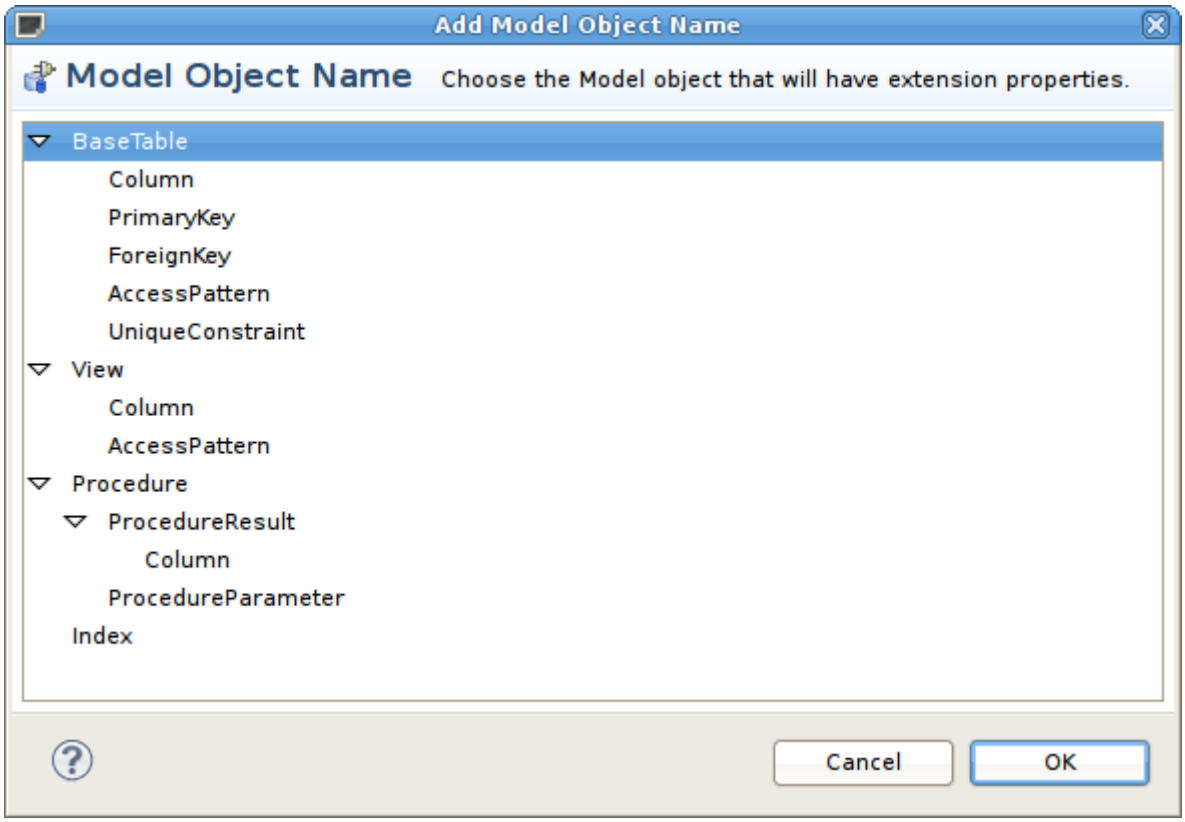


Figure 6.53. Select Model Object Name Dialog

Next, select the model object in the **Extended Model Objects** section and use the actions and propertes table in the lower **Extension Properties** section to add/remove or edit your actual extended properties. Selecting the add or edit extension properties actions displays a dialog containing sections to edit general properties, value definition (required, masked, allowed values) as well as display name and description values which can be internationalized.

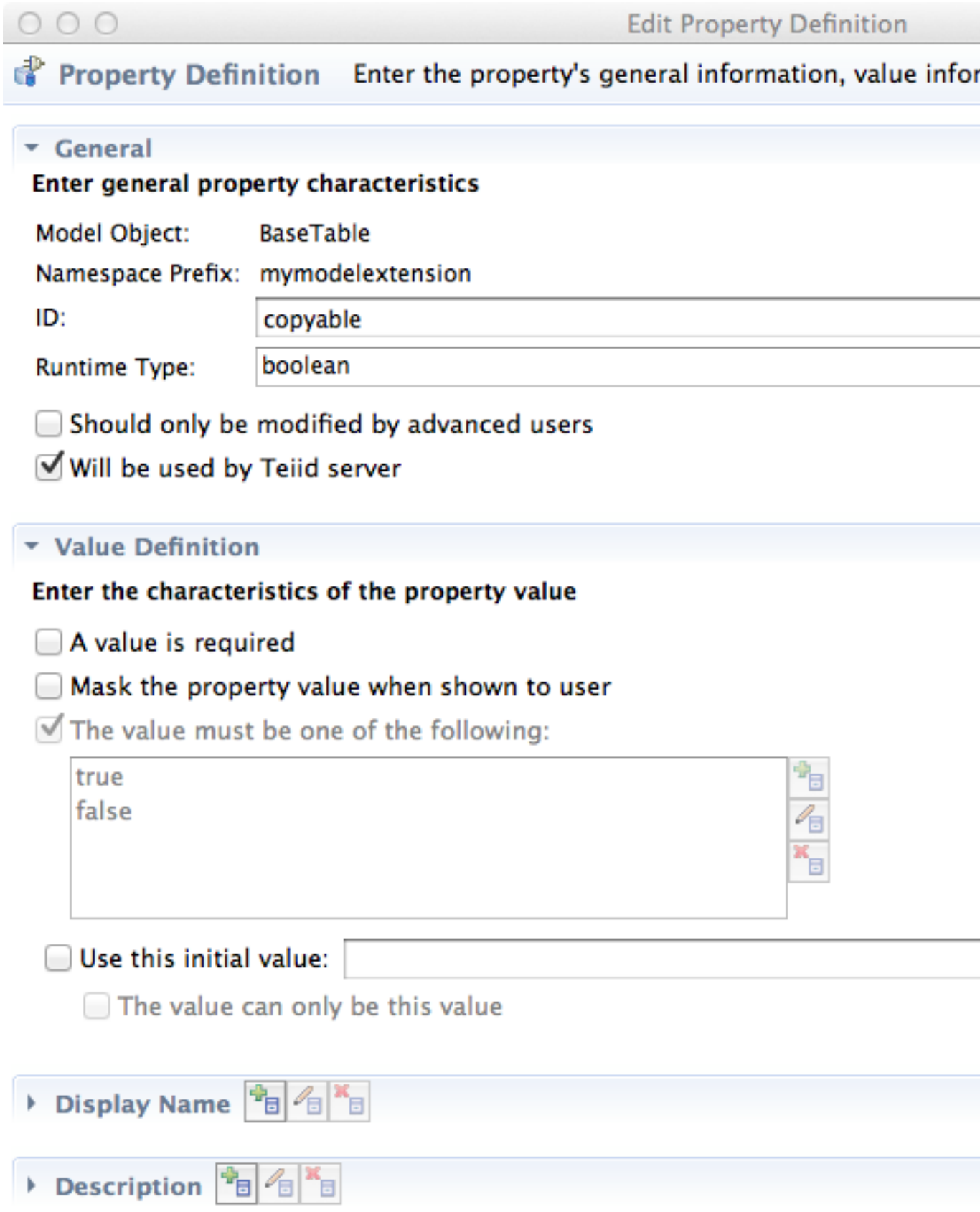


Figure 6.54. Edit Property Definition Dialog

6.4.3. Extending Models With MEDs

MEDs must be applied to a model in order for their extension properties to be available to that model's model objects. To manage the applied MEDs for a specific model select the model and right-click select the **Modeling > Manage Model Extension Definitions** action. This will display a dialog listing the current applied MEDS and actions and buttons to add or remove MEDs from a model, extract a MED from a model and save a copy of it locally as a '.mxd' file and lastly, update the version of MED in a model if it differs from a version in your MED registry.

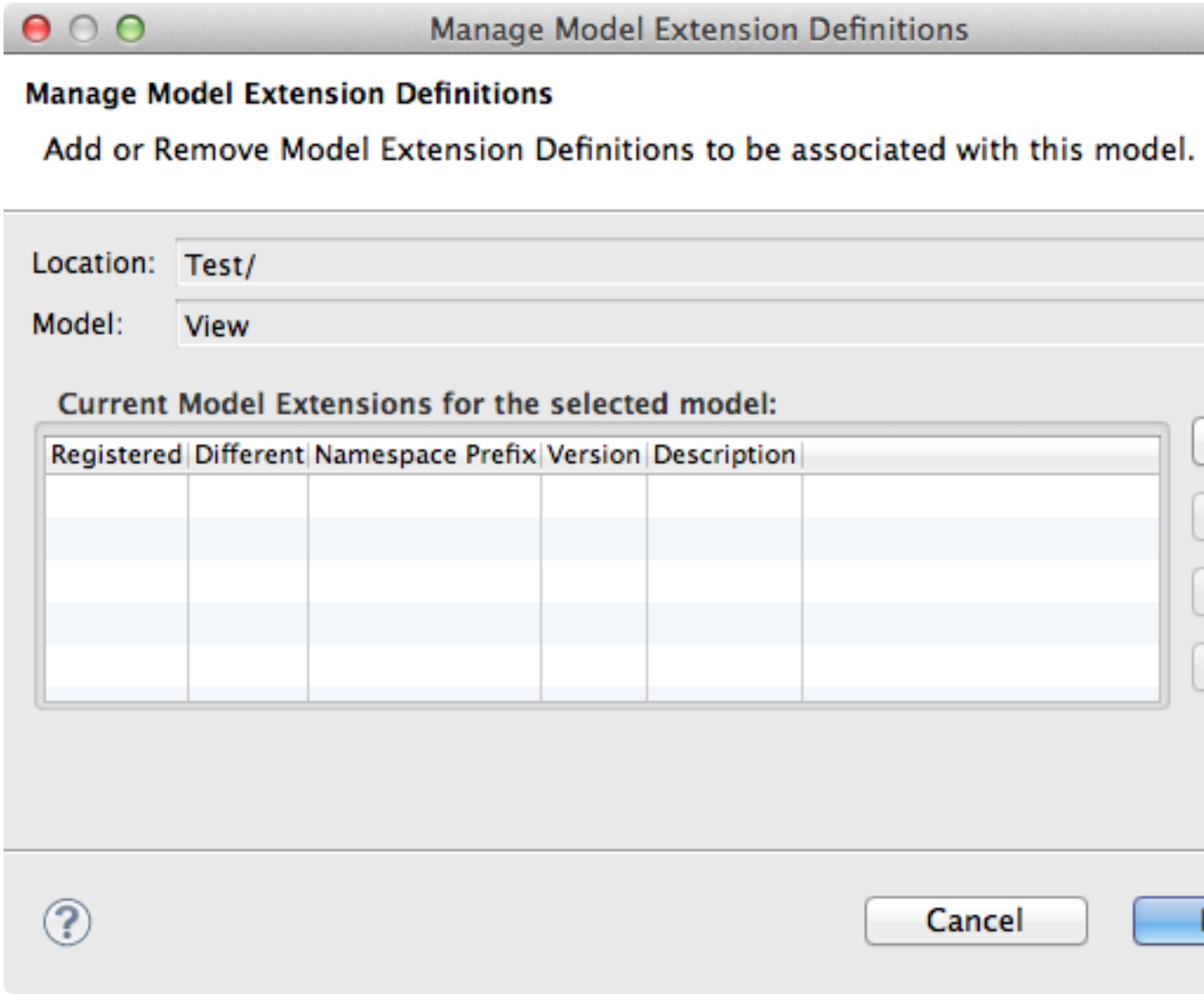


Figure 6.55. Manage Model Extension Definitions Dialog

Selecting the **Add** button displays a list of applicable MEDS based on model class.

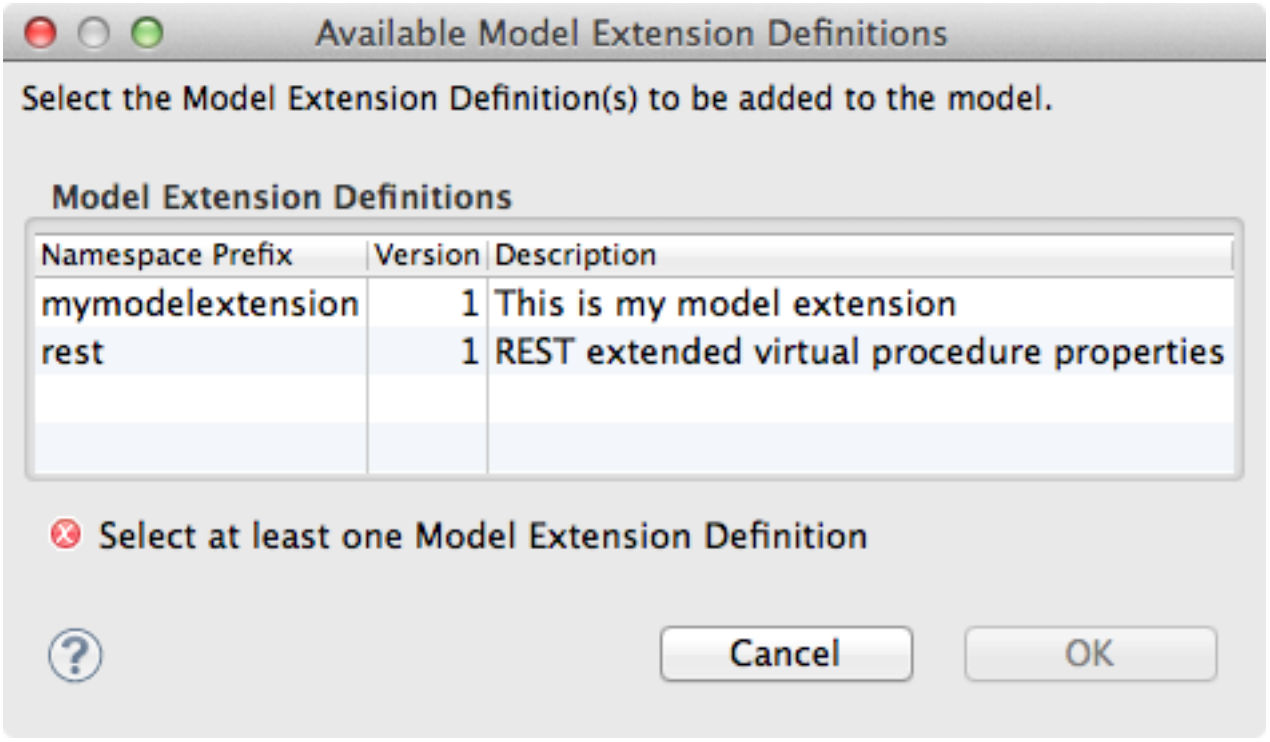


Figure 6.56. Add Model Extension Definitions Dialog

> **Note**
>
> After adding/removing MEDs from the model, click 'Finish' to accept all of the changes. Cancelling the dialog will discard all changes and revert to the original model state.

6.4.4. Setting Extended Property Values

Extension properties are user-defined properties available to any extended model object via the Properties View. As shown below, all extension properties are available up under the "Extension" category and are prefixed with a MED's namespace prefix. If there is an initial value for an extension property it will be set to the default value using the property definition found in the MED.

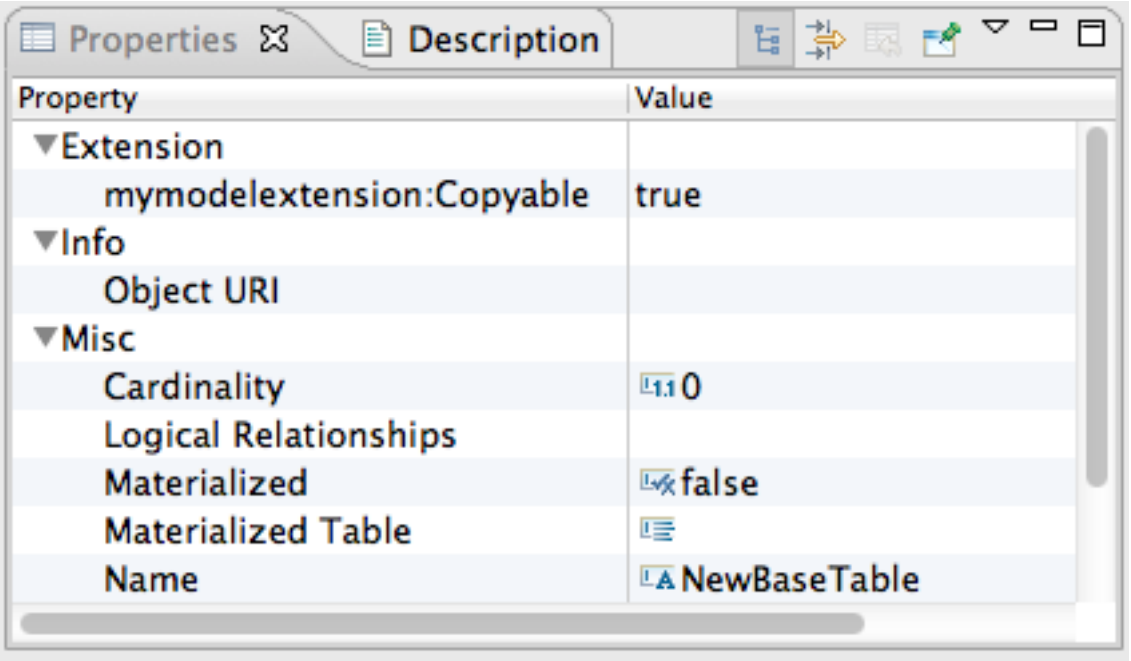


Figure 6.57. Properties View For Extended Model Object

Metadata-specific Modeling

This chapter discusses various features targeted at defining and managing metamodel-specific objects.

7.1. Relational Source Modeling

7.1.1. Source Function

The ability to utilize functions within your transformations can be a powerful tool for manipulating and managing the structure and content of your virtual data. Teiid Designer exposes a wealth of system functions through the Expression Builder wizard to select and define function calls within your SQL.

Teiid Designer also provides features for defining source functions to represent functions for your target relational data base, as well as user-defined functions coded in java.

If you define a source function in a source model or user-defined function in a view model, that new function will be available to use in your transformations.

See *Section 6.2.2, "Create Relational Procedure Wizard"* for details.

7.2. Relational View Modeling

This section contains descriptions of various features related to creating and managing relational view model objects.

7.2.1. Create Materialized Views

For any relational view table you can enable it's materialized view option by setting the **Materialized** property to TRUE and setting the **Materialized Table** reference, as shown in the figure below.

Note

You are required to have already created your relational tables.

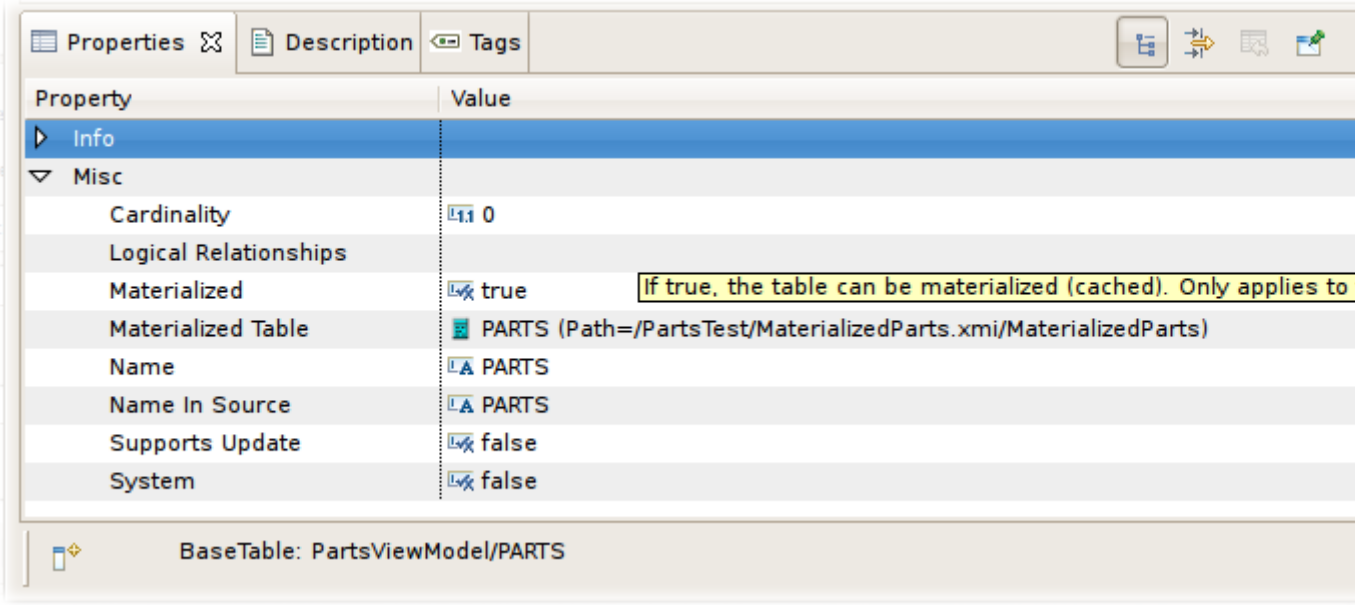
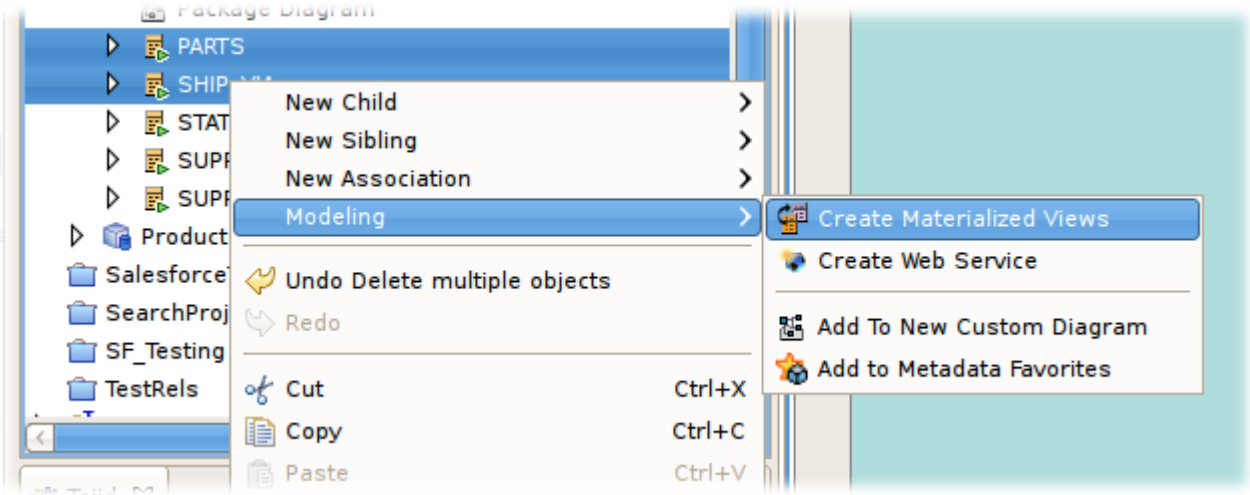


Figure 7.1. Materialized Table Properties

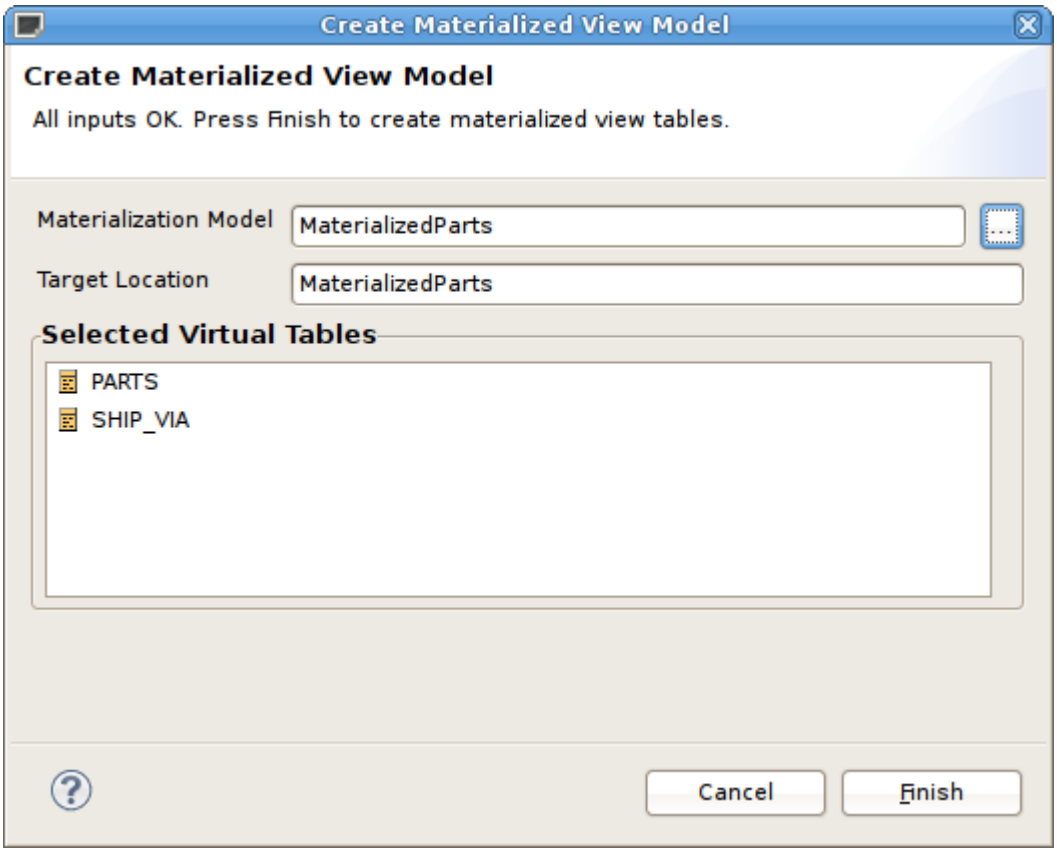
Designer includes a feature to assist in quickly creating materialized relational tables based on your existing view tables.

To create materialized views:

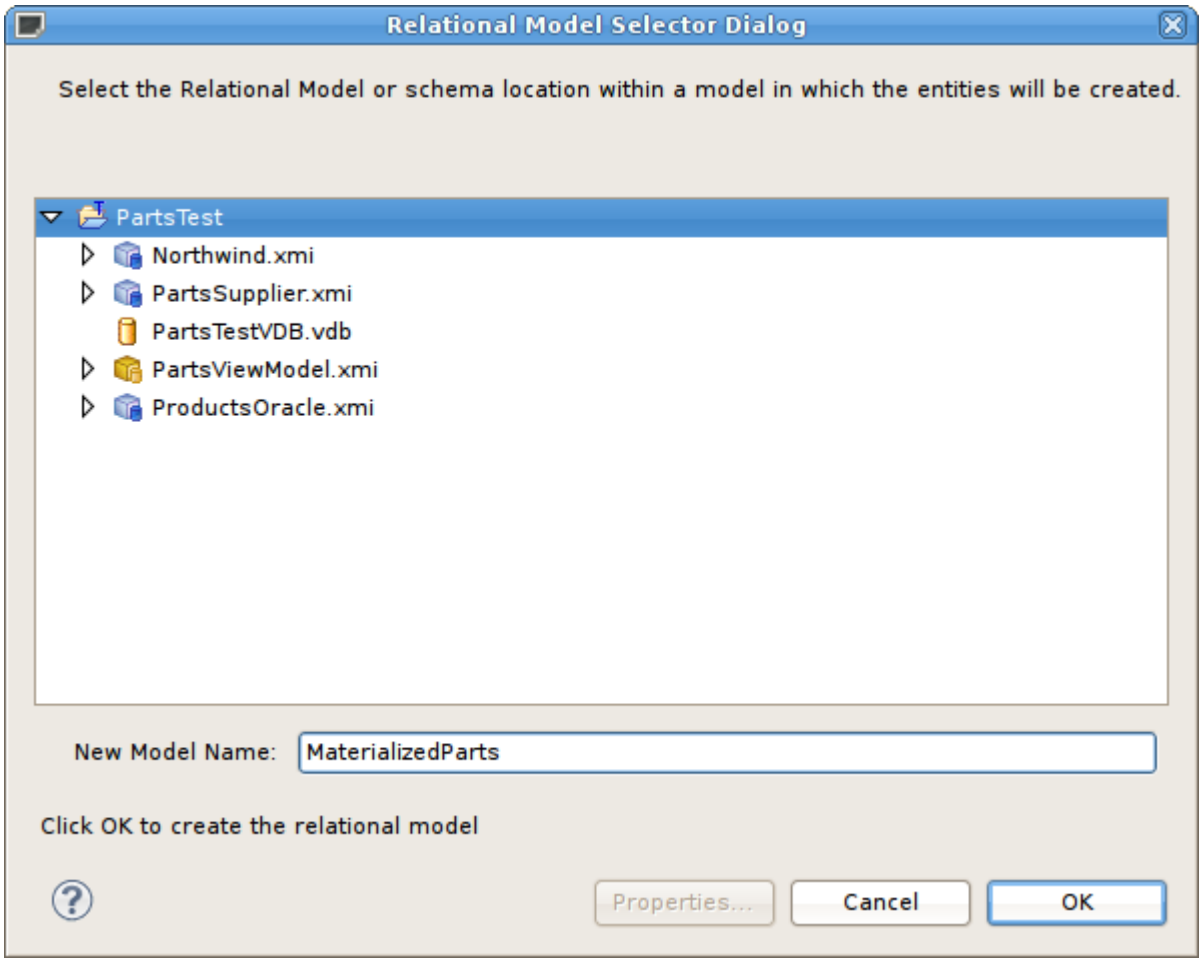
- **Step 1** - Right-click on one or more view tables in the *Section D.2.1, "Model Explorer View"* and select the **Modeling > Create Materialized Views** action.



- **Step 2** - In the **Create Materialized View Model** dialog specify or select a target relational model for your generated relational tables.



- **Step 2a** - Selecting the browse '...' button displays the **Relational Model Selector dialog** where you select an existing relational model or specify a unique name for a new model.



- **Step 3** - Click **OK** to create relational tables corresponding to your selected view tables and automatically set the **Materialized** property to TRUE and the **Materialized Table** reference value to your newly generated table.

When finished your view tables will be configured with their new materialized properties and the corresponding relational tables will be shown in their package diagram.

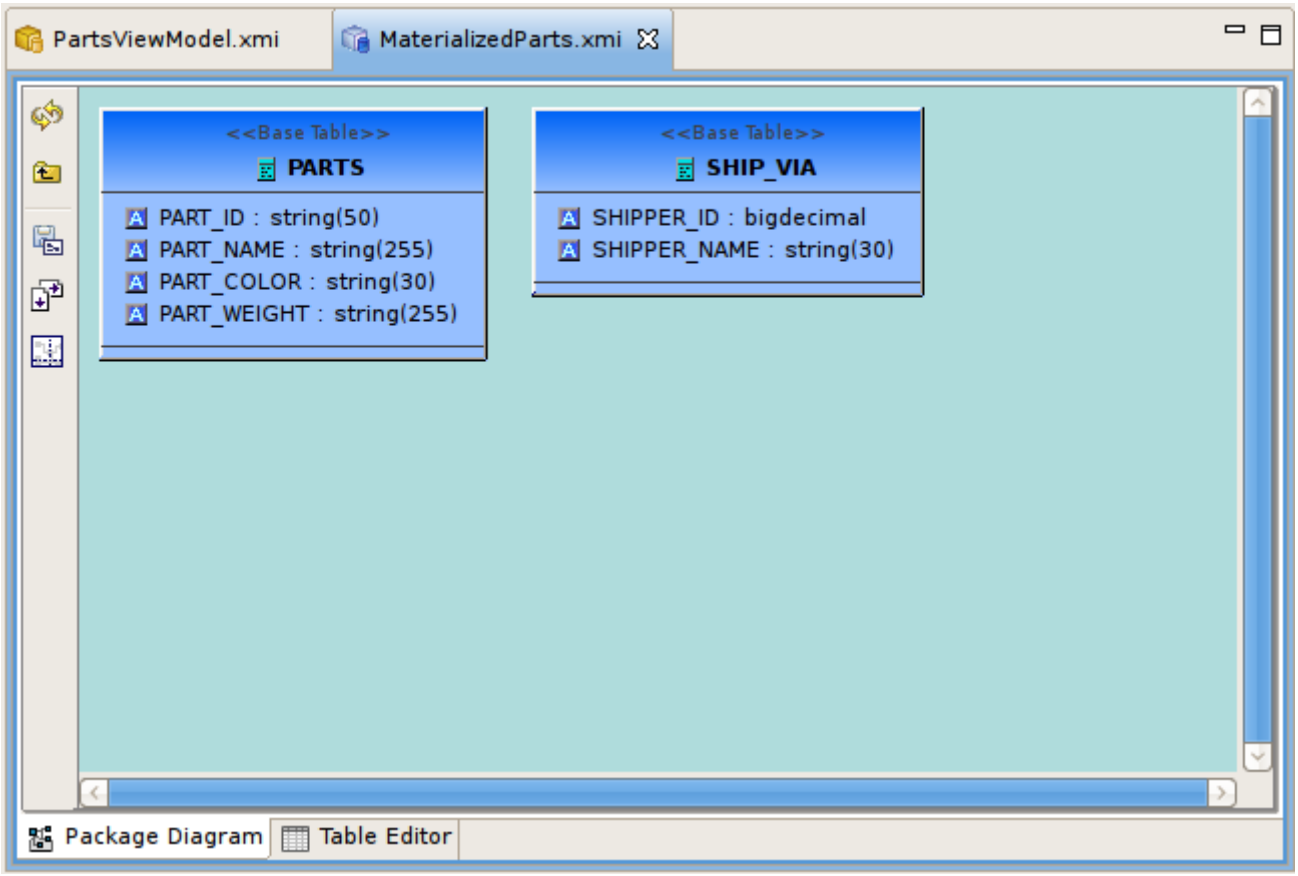


Figure 7.2. Materialized Table Properties

7.3. XML Document Modeling

7.3.1. Create XML View Documents from schema

You can create XML View Documents by selecting an element in the Model Explorer and selecting the **Modeling > Create XML View Documents** action.

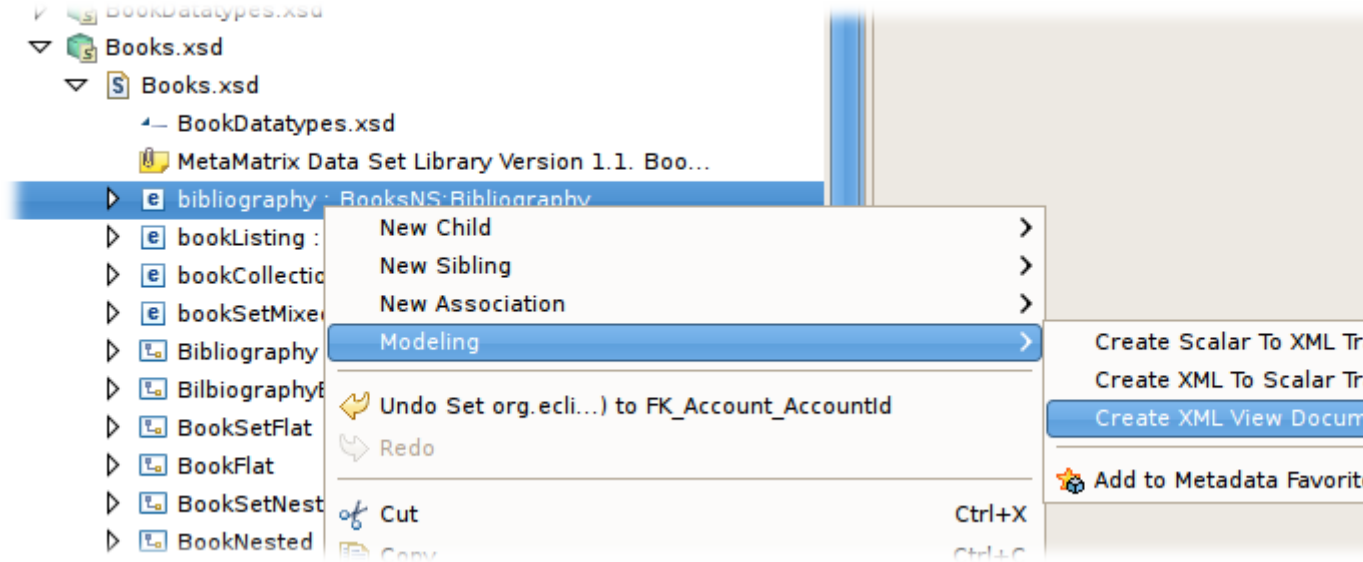


Figure 7.3. Create XML View Documents Action

The action will query you for a target XML Document model. You can either select an existing XML Document model from your workspace, or enter a unique model name and the wizard will create a new model for you.

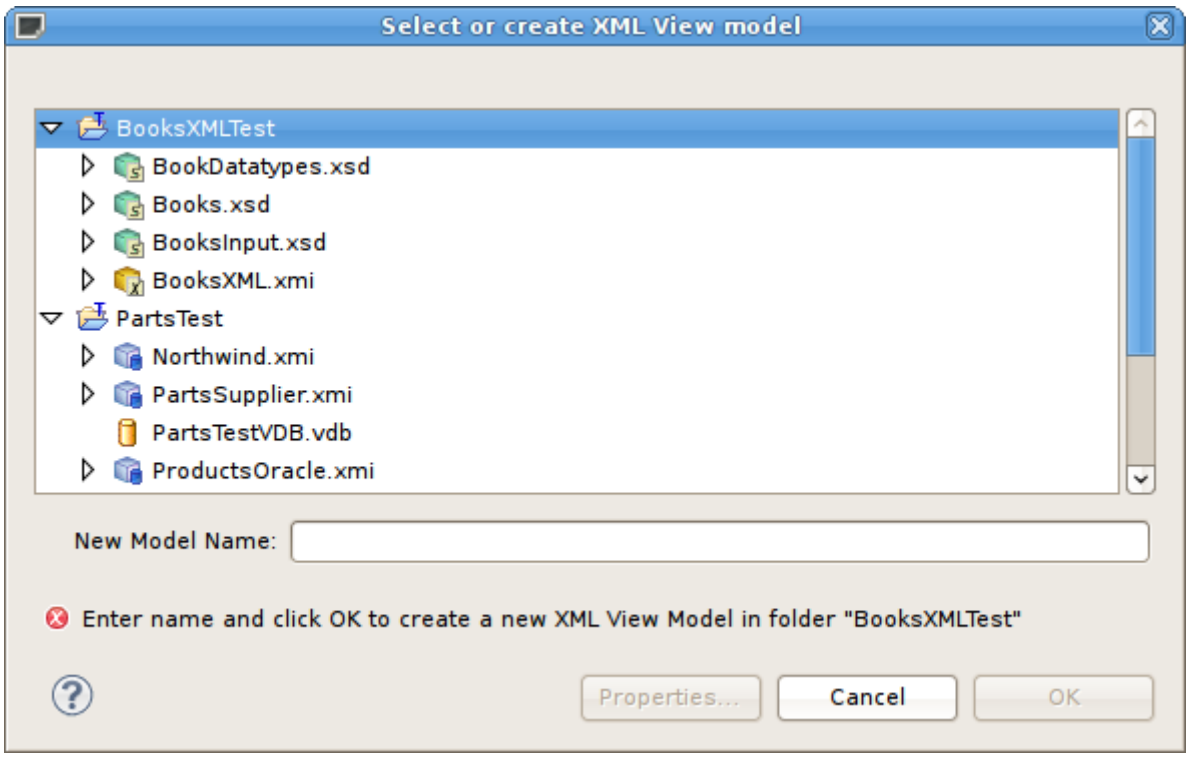


Figure 7.4. Select or Create XML View Model Dialog

After selecting or creating your new XML Document model, the XML Document builder page will be displayed. This page is explained in greater detail in *Section 4.3.2, “Build XML Documents From XML Schema ”* section.

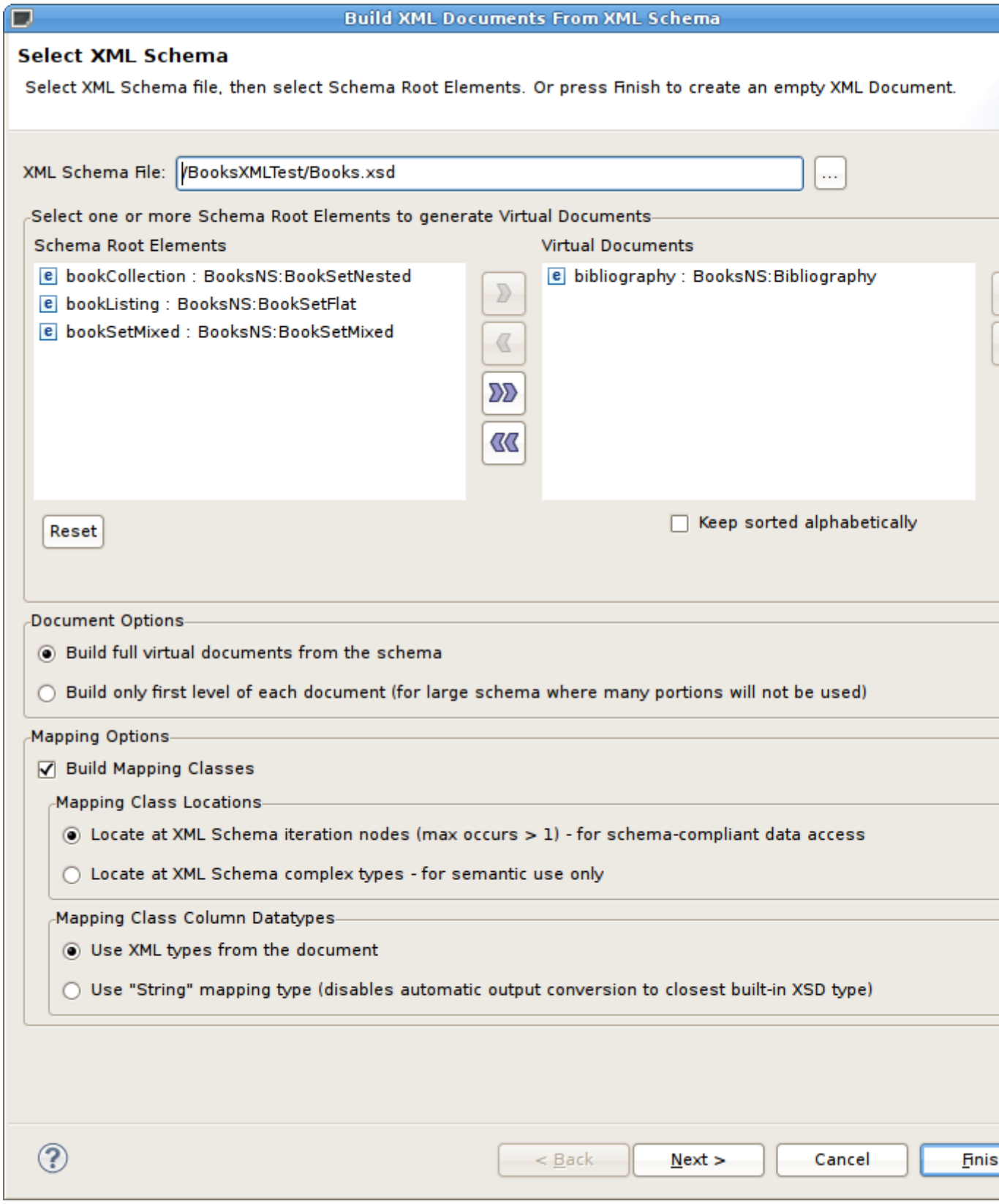


Figure 7.5. Build XML Documents From XML Schema Dialog

7.4. Web Services Modeling

7.4.1. Create Web Service Action

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 *Section D.2.1, "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 7.6. 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.

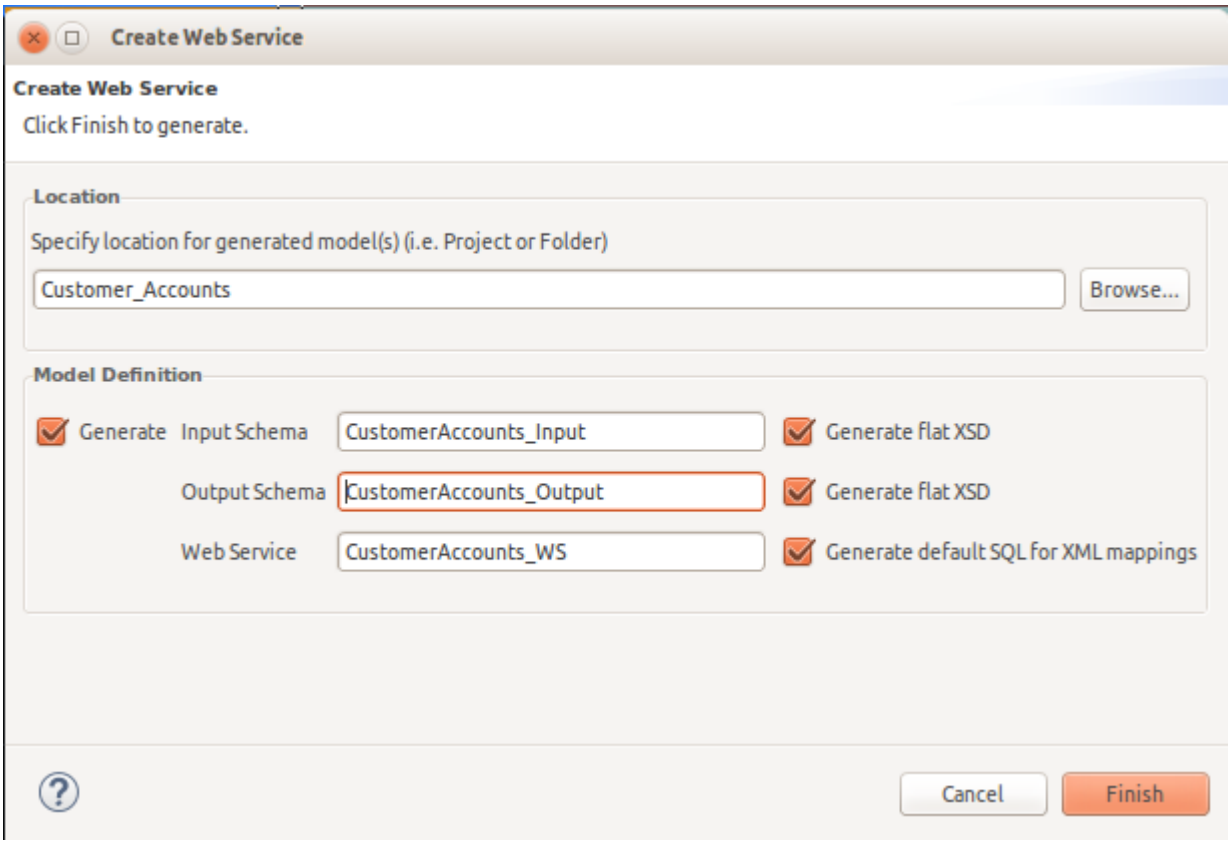


Figure 7.7. Create Web Service Dialog

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

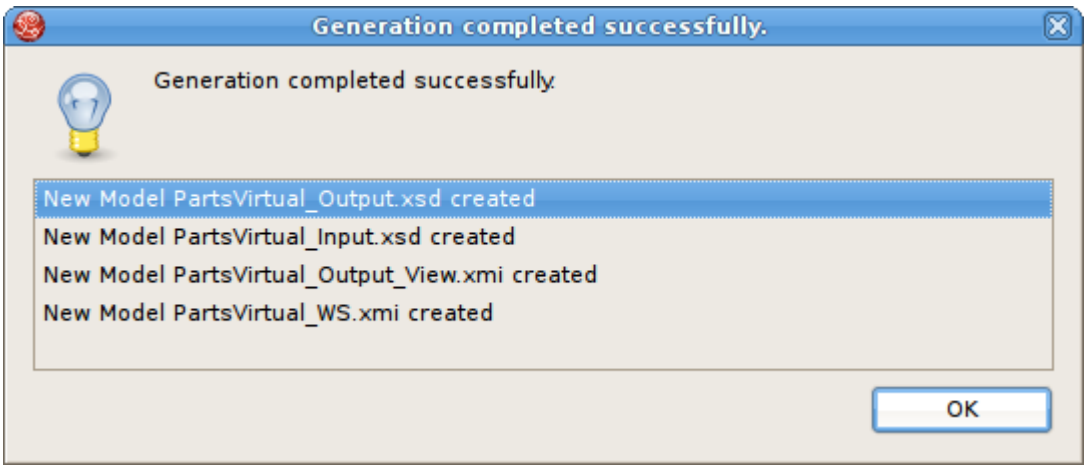


Figure 7.8. Generation Completed Dialog

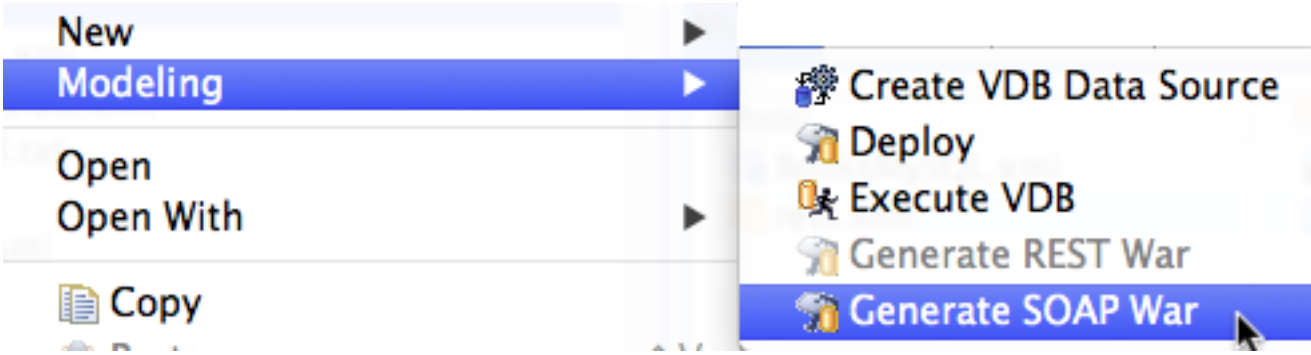
7.4.2. Web Services War Generation

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.

7.4.2.1. Generating a SOAP War

The Teiid Designer provides SOAP web service generation capabilities in the form of a JBossWS-CXF war. Once you have added your **Web Service Models** as described in *Section 4.5, "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 SOAP 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.

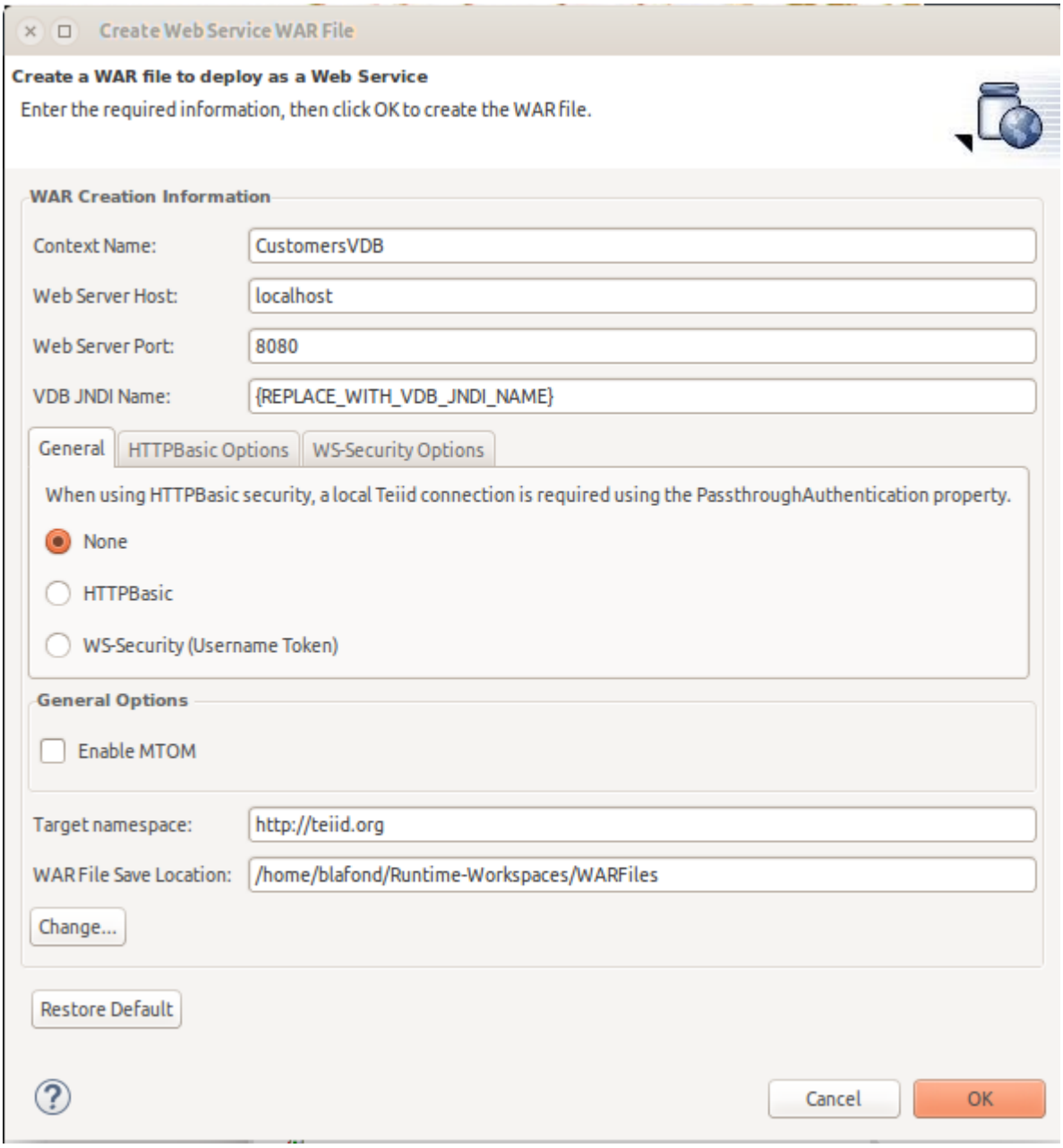


Figure 7.9. Generate a SOAP War Web Service Dialog

Table 7.1. Field Descriptions

Field Name	Description
Name	The name of the generated war file.
Host	The server host name (or IP).
Port	The server port.

Field Name	Description
VDB JNDI Name	The JNDI connection name to the deployed Teiid source VDB.
Security options	• **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.
MTOM (Message Transmission Optimization Mechanism)	If selected, MTOM will be enabled for the web service endpoint(s). You will also need to update your output schema accordingly by adding the xmlns:xmime="http://www.w3.org/2005/05/xmlmim" schema and adding type="xs:base64Binary" xmime:expectedContentTypes="application/octet-stream" to the output element you wish to optimize.

Field Name	Description
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**.

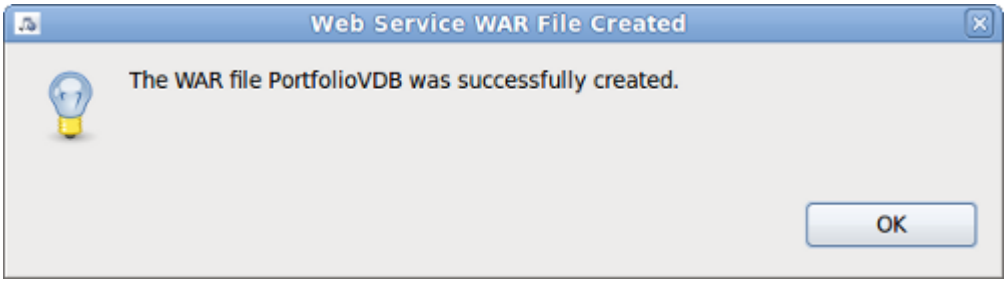
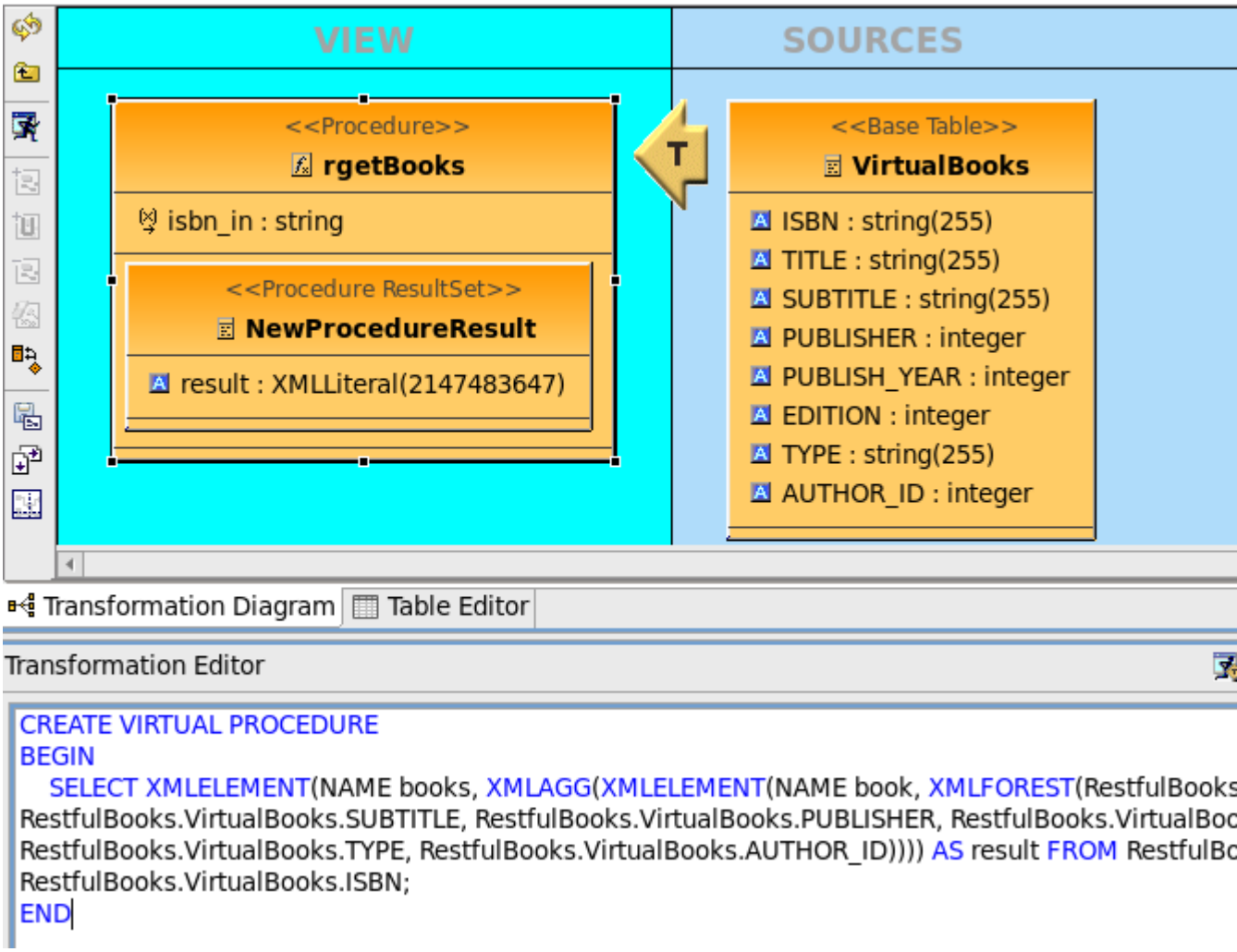


Figure 7.10. Generation Completed Dialog

7.4.2.2. Generating a REST War

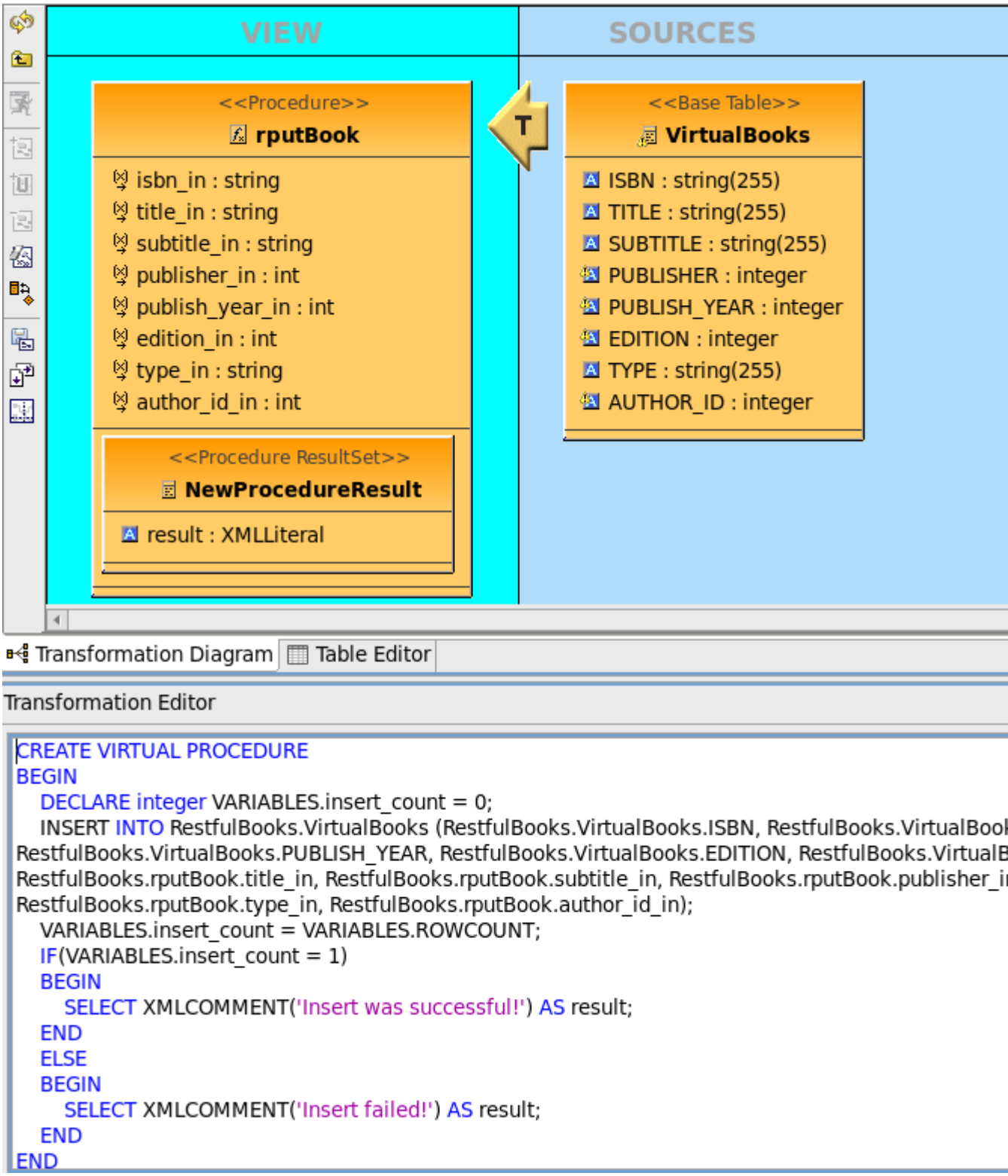
In Teiid Designer, it is also possible to expose your VDBs over REST using a generated RESTEasy war. Also, if your target virtual model has update, insert and delete SQL defined, you can easily provide CRUD capabilities via REST. Accepted inputs into the generated REST operations are URI path parameters and/or XML/JSON. JSON is exposed over a URI that includes "json". For example, "http://{host}:{port}/{war_context}/{model_name}/resource" will accept URI path parameters and/or XML while "http://{host}:{port}/{war_context}/{model_name}/json/resource" will accept URI path parameters and/or JSON.

- **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 input format for the REST procedure could be URI parameters, an XML/JSON 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 7.11. Sample XML Input

When using a JSON document, your values should match the order of your procedure input parameters as well. Here is the input for the above insert procedure:

```
{
"ISBN": "1-55615-484-4",
"TITLE": "Code Complete",
"SUBTITLE": "A Practical Handbook of Software Construction",
"PUBLISHER": 5,
"PUBLISH_YEAR": 1993,
"EDITION": 1,
"TYPE": "Hardback",
"AUTHOR_ID": 31
}
```

Figure 7.12. Sample JSON Input

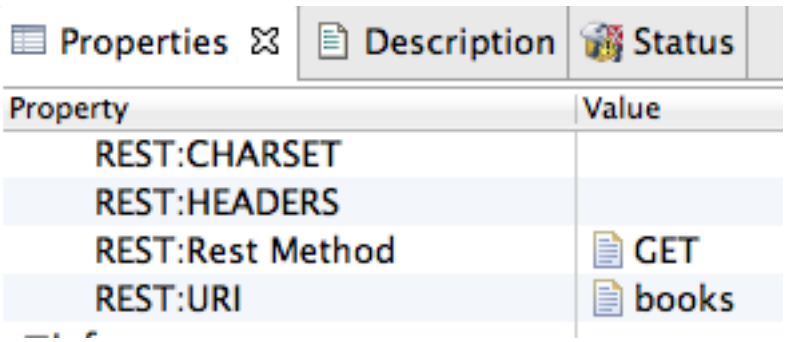
- **Step 2** - Now we need to identify our procedure as REST eligible. To do this, select the procedure and note the REST properties in the property panel. The REST Method and URI are the only required properties to make the procedure REST eligible. All of the properties are defined in the table below:

Table 7.2. Properties for RESTful Procedures

Property Name	Description	Required
Rest Method	The HTTP method that will determine the REST mapping of this procedure. Supported	Yes

Property Name	Description	Required
	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.	Yes
Description	Optional property that describes the procedure. This value is used by Swagger when documenting the REST API.	No
CHARSET	Optional property for procedures that return a Blob with content type that is text-based. This character set to used to convert the data.	No
HEADERS	Semi-colon delimited list of HTTP Header parameters to pass into the REST service operation. Ex. header1;header2;etc. These value will be passed into the procedure first.	No

Here's what the above example would look like in the Property tab:



Note

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**.

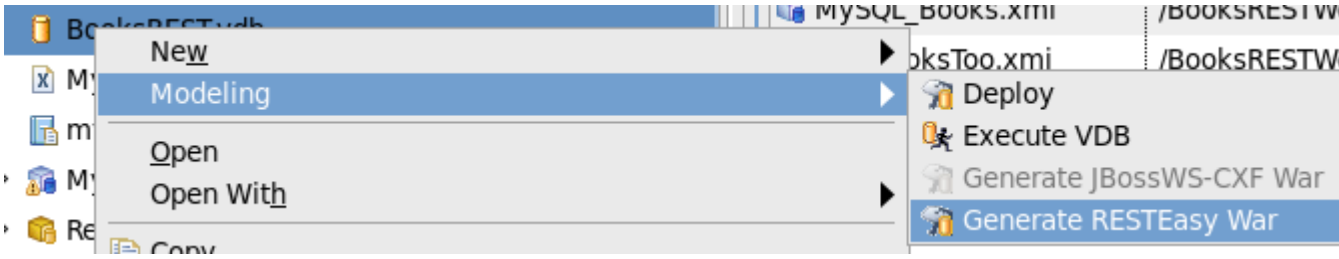
Once you have added all of your procedures along with the required extended properties, be sure and add the model to your VDB or synchronize if it's already included in the VDB. You will then need to re-deploy the VDB.

Important

*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.

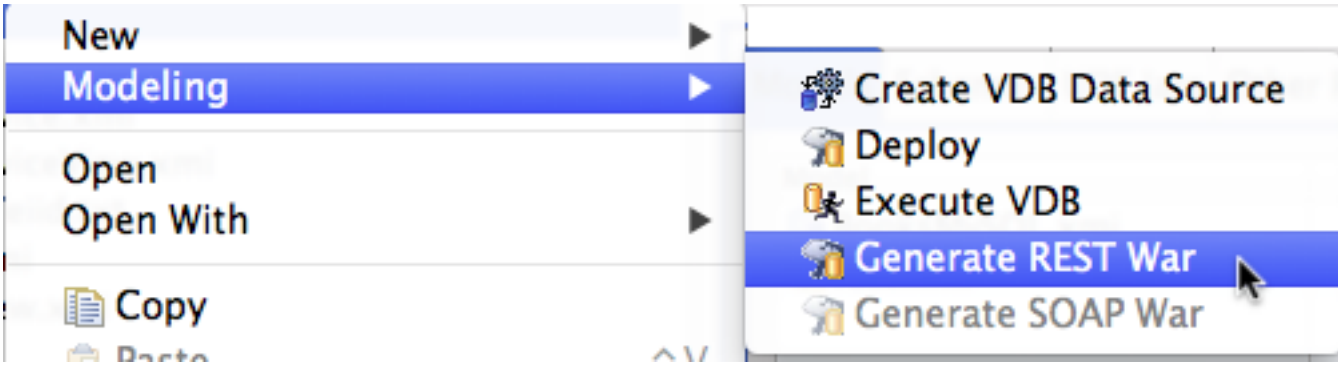


Figure 7.13. Generate a REST WAR War File Dialog

Table 7.3. 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 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**.

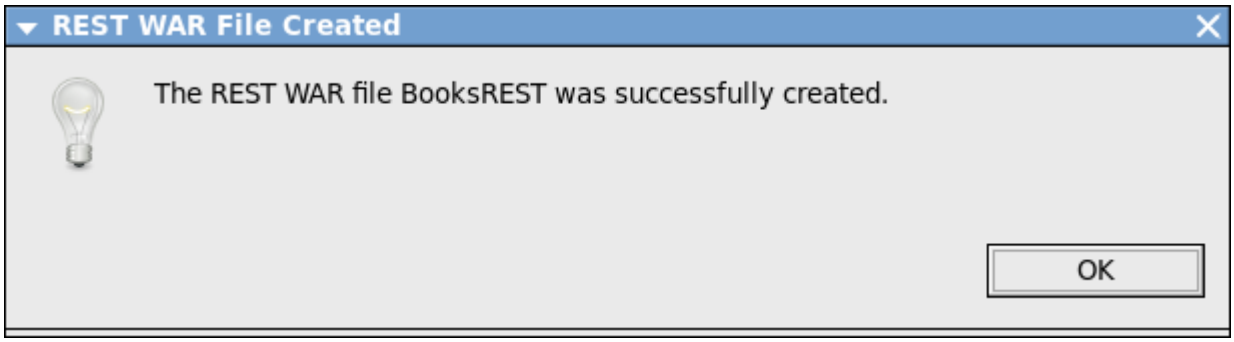


Figure 7.14. Generation Completed Dialog

7.4.2.3. Deploying Your Generated WAR File

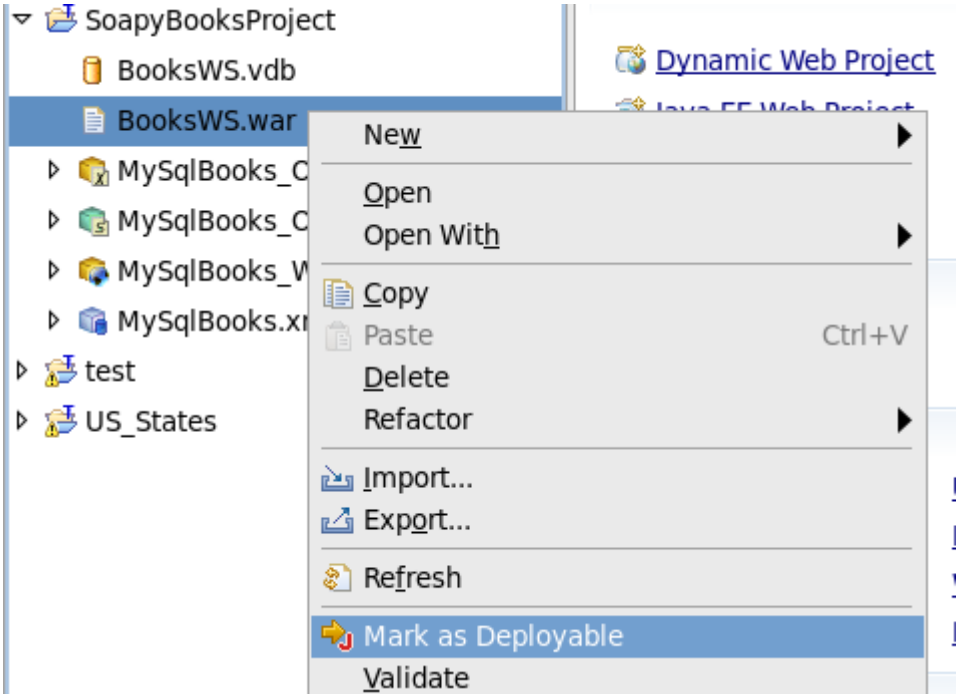
Once you have generated your war file, you will need to deploy it to your JBoss AS instance. There are a few ways to accomplish this.

- **From JBDS or JBoss Tools**

  1. Insure target JBossAS is configured and running.

  2. Select your WAR file in the Model Explorer view. If you didn't generate your war to that location, you can copy and paste it there.

  3. Right-click on the WAR file and select 'Mark as Deployable'. This will cause you WAR file to be automatically deployed the JBoss AS instance you have defined.



- **Using the JBoss AS Administration Console**

  Using the administration console that comes with JBoss AS, you can deploy WAR files. The administration console is available at http://{host:port}/admin-console. Once logged on, simply use the "Add a New Resource" button of the "Web Application (WAR)" resource folder.

- **Manual Deployment to JBossAS**

  It is possible to deploy the generated WAR by manually copying the file to the "deploy" folder of the target JBoss AS. If the server is running, the WAR will deploy automatically via "hot" deploy. Otherwise, the WAR will deploy at the next start of the server.

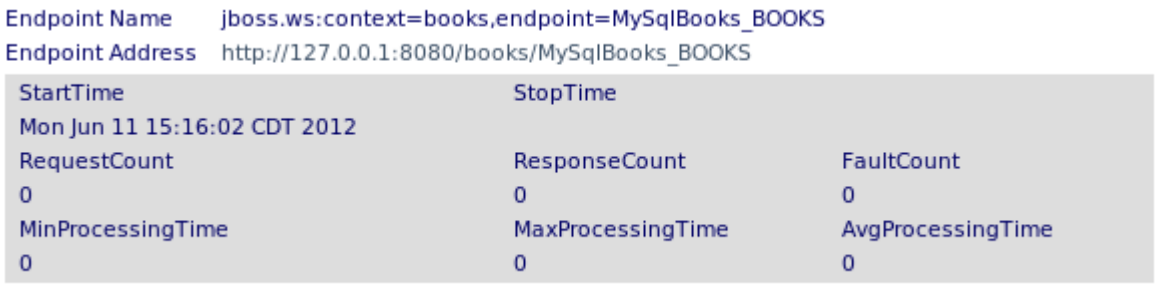
7.4.2.4. Testing Your Generated WAR Files

Once you have deployed your war file, you are ready to test it out. There are a few ways to accomplish this.

SOAP WAR Testing

- **Determining Your WSDL URL**

  You can get your WSDL URL at http://{server:port}/jbossws/services. This is where all the deployed web services for the target JBossAS server will be listed. Find your service and click the Endpoint Address link. This will retrieve your web service's WSDL and the WSDL URL address will appear in the browser's address bar.



  Now that you have your WSDL URL, you can use any SOAP testing tool such as the Web Service Tester that comes with JBDS and JBoss Tools or an external tool like soapUI.

- **Using the JBoss AS Administration Console**

  Using the administration console that comes with JBoss AS, you can deploy WAR files. The administration console is available at http://{host:port}/admin-console. Once logged on, s simply use the "Add a New Resource" button of the "Web Application (WAR)" resource folder.

REST WAR Testing

- **What is my URI?**

  When you modeled your REST procedures, you assigned a URI for each HTTP Operation you defined along with the corresponding operation (GET, PUT, POST or DELETE). The full path of each URI is defined as /{war_context}/{model_name}/{resource} for XML input/output and /{war_context}/{model_name}/json/{resource} for JSON input/output.

  Swagger is a framework that exposes REST APIs and it is embedded in the generated REST WAR. This allows you to execute your methods directly from the deployed WAR. It is available at the root of the WAR (http://{host:port}/{context}).

VirtualHana

GET /VirtualHana/BooksVirtual

Response Class (Status 200)

Model | Model Schema

Response Content Type application/xml; charset=UTF-8

Try it out! Hide Response

Curl

```
curl -X GET --header "Accept: application/xml; charset=UTF-
```

Request URL

```
http://localhost:8080/hana/VirtualHana/BooksVirtual
```

Response Body

```
<elems>
  <elem>
```

Using your REST URL, you can also use any testing tool with REST support such as the Web Service Tester included with JBDS and JBoss Tools or an external tool like soapUI or cURL.

7.4.2.5. Auto-generate a REST WAR

Instead of generating the REST WAR in Designer, you can also set a property on your VDB containing a REST eligible procedure to auto-generate a REST war at VDB deployment time. In the VDB Editor, the properties tab has a checkbox to tell the Teiid engine to generate a REST WAR upon deployment. In addition to checking the **Generate REST WAR** checkbox, a role of **rest** is required for the user accessing the REST URL. Please see the Teiid documentation for more information about the auto-generated REST WAR.

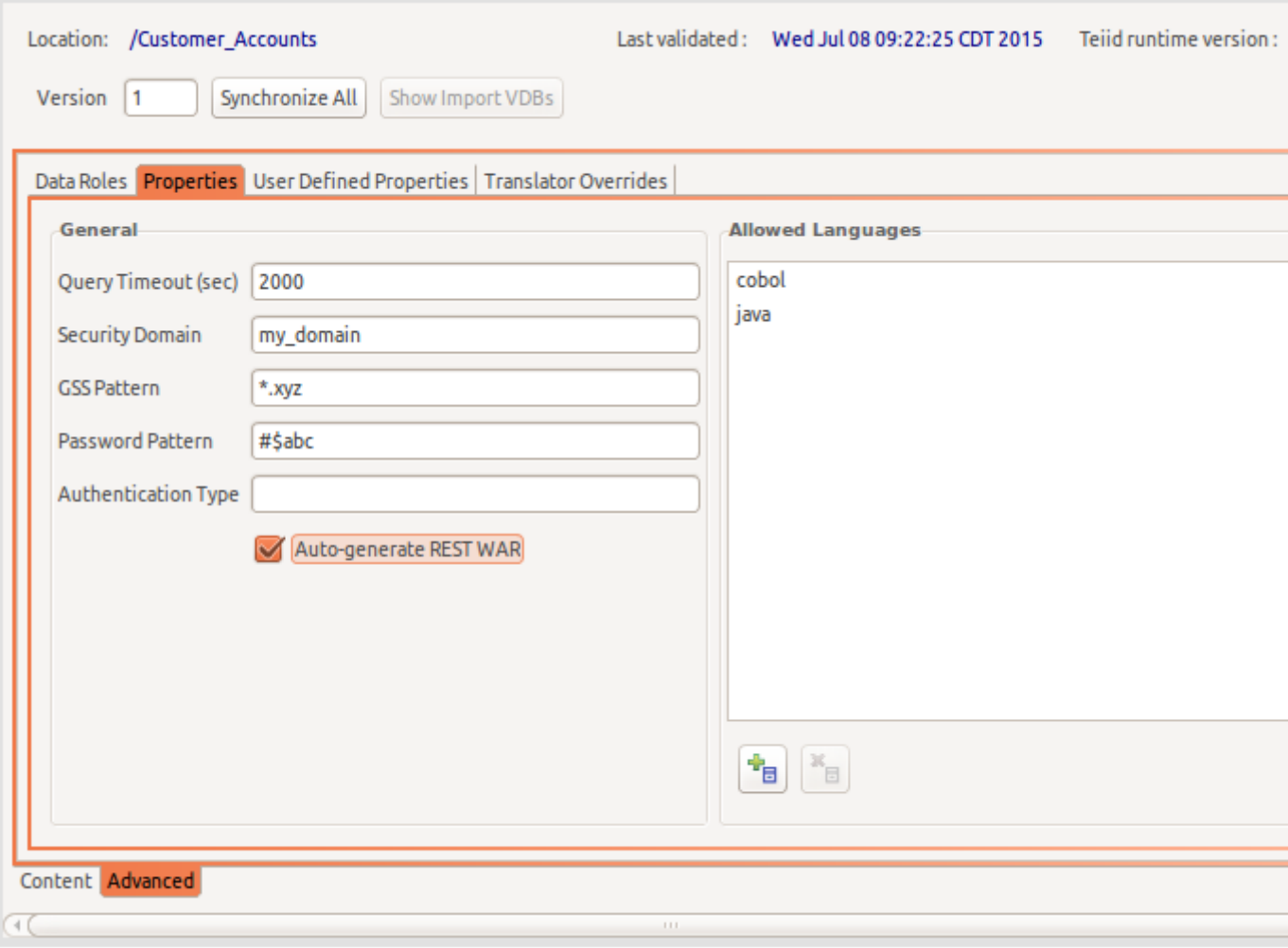


Figure 7.16. REST WAR Auto-generation Option

User Defined Functions

Teiid Designer enables users to extend Teiid's scalar or aggregate function library via modeling and definition of custom or User Defined Functions(UDFs).

The following are used to define a UDF.

- **Function Name** - When you create the function name, keep these requirements in mind:
  - You cannot overload existing Teiid System functions.
  - The function name must be unique among user-defined functions in its model for the number of arguments. You can use the same function name for different numbers of types of arguments. Hence, you can overload your user-defined functions.
  - The function name cannot contain the '.' character.
  - The function name cannot exceed 255 characters.
- **Input Parameters**- defines a type specific signature list. All arguments are considered required.
- **Return Type**- the expected type of the returned scalar value.
- **Pushdown** - can be one of REQUIRED, NEVER, ALLOWED. Indicates the expected pushdown behavior. If NEVER or ALLOWED are specified then a Java implementation of the function should be supplied. If REQUIRED is used, then user must extend the Translator for the source and add this function to its pushdown function library.
- **invocationClass/invocationMethod**- optional properties indicating the method to invoke when the UDF is not pushed down.
- **Deterministic** - if the method will always return the same result for the same input parameters. Defaults to false. It is important to mark the function as deterministic if it returns the same value for the same inputs as this will lead to better performance. See also the Relational extension boolean metadata property "deterministic" and the DDL OPTION property "determinism". Defaults to false. It is important to mark the function as deterministic if it returns the same value for the same inputs as this will lead to better performance. See also the Relational extension boolean metadata property "deterministic" and the DDL OPTION property "determinism".

Even Pushdown required functions need to be added as a UDF to allow Teiid to properly parse and resolve the function. Pushdown scalar functions differ from normal user-defined functions in that no code is provided for evaluation in the engine. An exception will be raised if a pushdown required function cannot be evaluated by the appropriate source.

8.1. Modeling your functions

Teiid Designer provides a new object wizard to assist in modeling your UDF. Simply right-click select a model and right-click select **New > Procedure** action. In the wizard there will be an option

to specify the functions's jar location on your file system, as well as the java class and java method. (See *Section 6.2.2, "Create Relational Procedure Wizard"* for further information)

If creating a function by other means, you'll need to select the procedure and edit the different properties for the procedure in the Properties view.

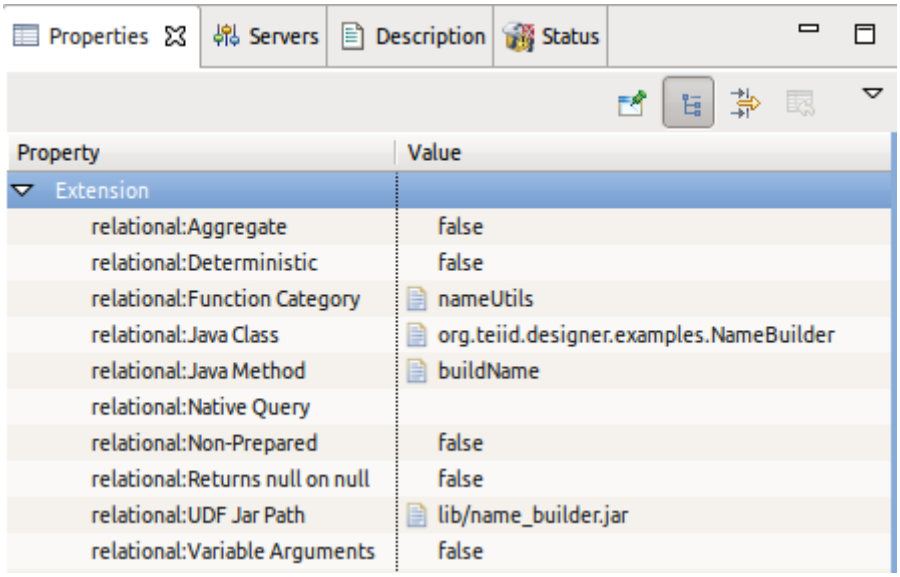


Figure 8.1. VDB UDF Jar Files

A User Defined Function represents a defined method in a java class. In order for the Teiid runtime to register the function and allow it's use in your transformations, you'll need to specify the following as properties in the figure above:

- **Function Category** - unique name for a grouping of user-defined functions. This category will be selectable in the *Expression Builder* wizard so you can locate and select your function.
- **Java Class** - fully qualified name of the java class containing your function method.
- **Java Method** - the name of the function method in your java class.
- **UDF Jar Path** - the relative path in your project for the jar containing your UDF which will be in a *lib* folder in your project. If you are defining your jar for the first time and there is no jars in your lib folder, the property editor will allow you to select a jar from your file system. If one or more jars already exist in a *lib* folder, the editor will allow you to choose between selecting from your workspace or your file system.

8.2. Utilizing your UDFs in transformations

Once you've modeled your functions you can use your function calls in your transformation SQL. These functions will be accessible through the Transformation Editor's expression builder wizard,

just like any other built-in function. (See *Section 6.3.1.4, "Using the Expression Builder"* for further information)

8.3. Including functions in your VDB

In order for Teiid to become aware of your functions, the actual code must be deployed on your server and available to your Teiid submodule. Your Teiid Designer workspace is aware of any models containing functions and their referenced jars and class info. When a source model containing UDFs is added to a VDB, the jar containing the defined function will be added to your VDB and visible in the VDB Editor's UDF Jars tab.

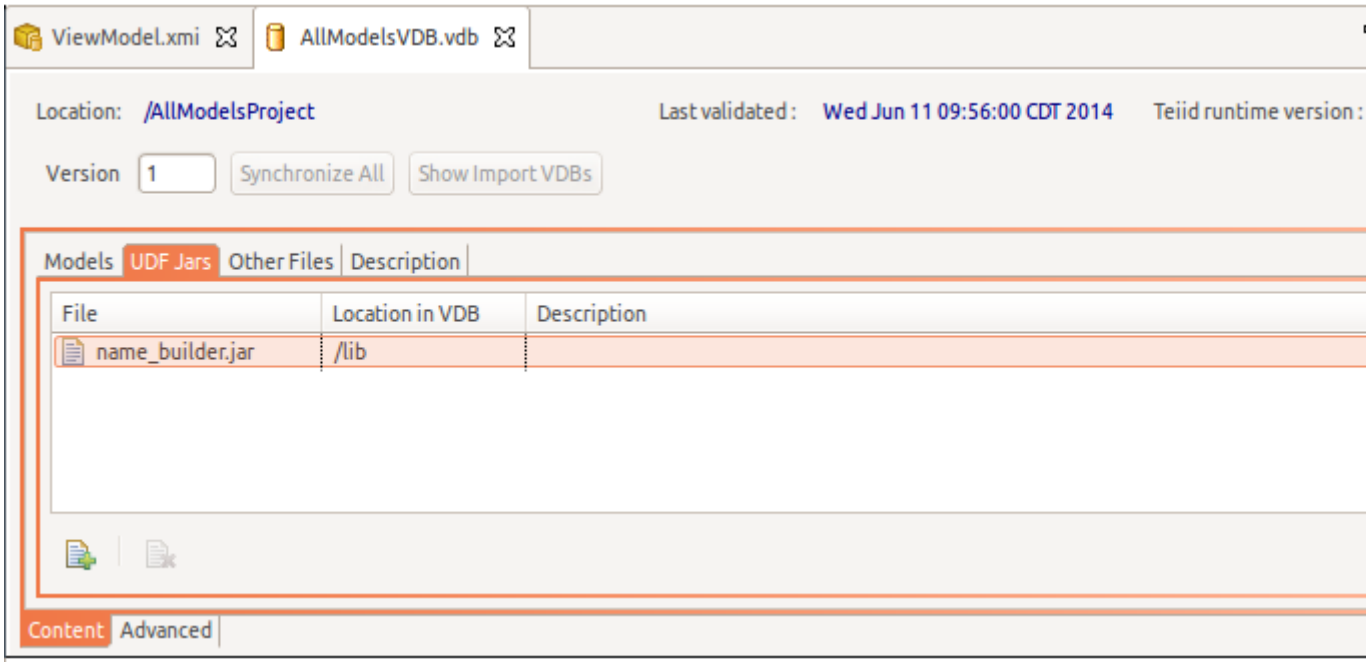


Figure 8.2. VDB UDF Jar Files

8.4. Using legacy UDF models

Function models have been deprecated and tooling support to create or edit them will be removed in Teiid Designer 11.0. In 10.0, we've added a feature to convert the functions in your legacy function models into User Defined Functions (virtual procedures) in a target view model.

To convert your function model:

- **Step 1** - In the **Model Explorer** select your function model and select the **Modeling > Convert Legacy Functions to Virtual Procedures** action.

Figure 8.3. Convert Legacy Functions to Virtual Procedures Action

- **Step 2** - In the dialog, check the functions you wish to convert and select the target virtual model and click **OK>**.

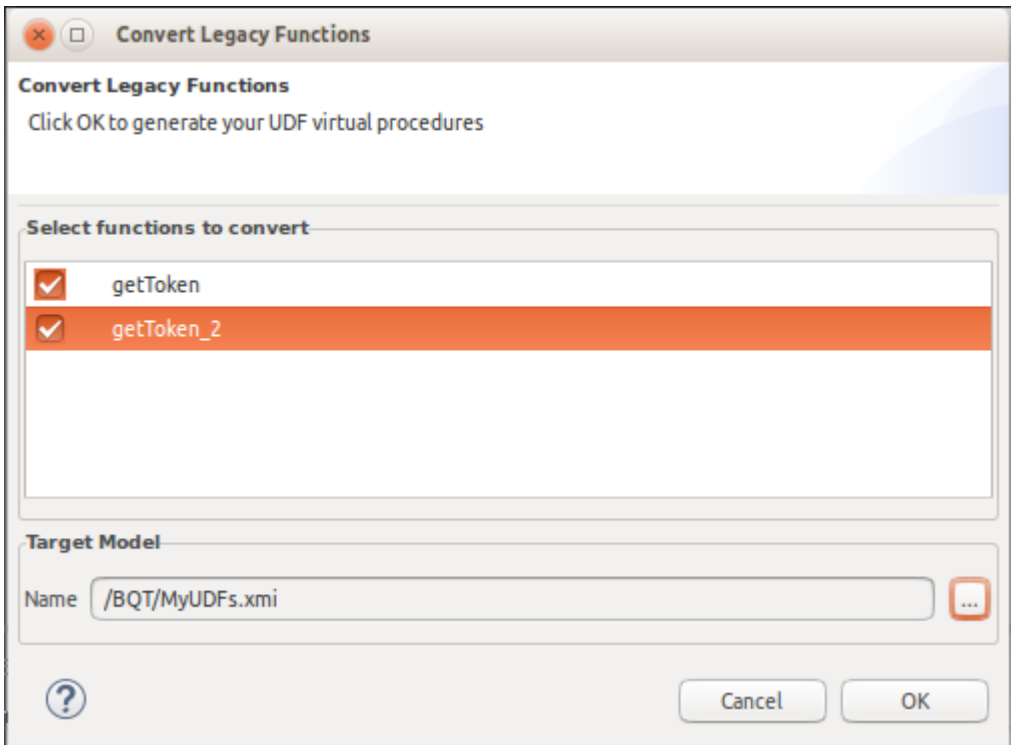


Figure 8.4. Convert Legacy Functions Dialog

The selected scalar functions are now defined as virtual model User Defined Functions and available to use in your transformations

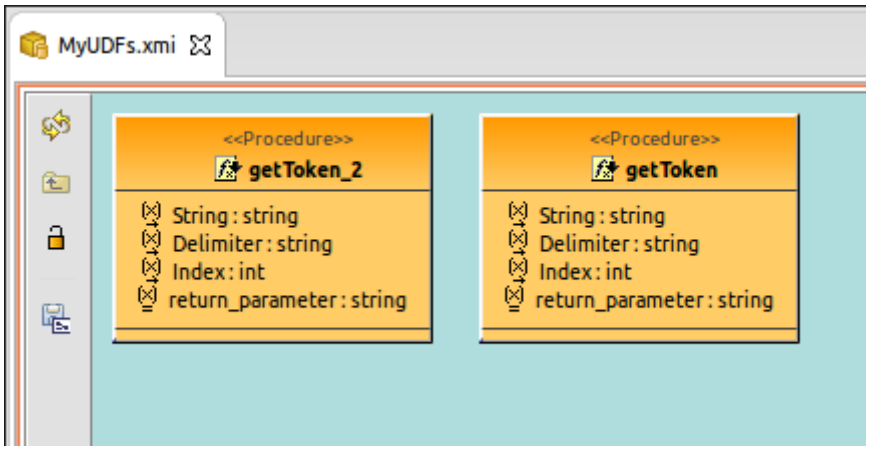


Figure 8.5. Converted Functions Diagram

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 *Section D.2.1, "Model Explorer View"*.
  - **Step 2** - Right-click select the **Refactor > Rename** action.

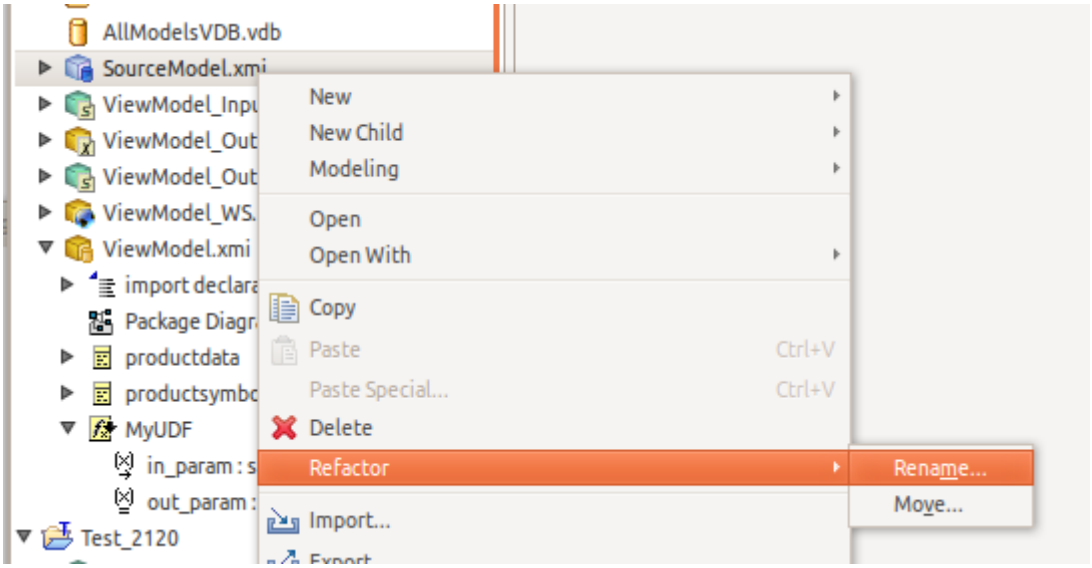


Figure 9.1. Refactor Rename Action In Model Explorer

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

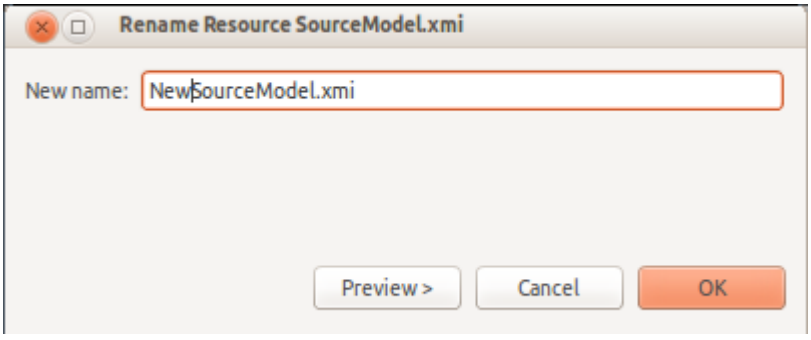


Figure 9.2. Rename Model File Dialog

- **Step 4 (Optional)** - The next **Rename Resource File** dialog will preview the changes that will be applied as part of renaming your model. Click **OK** to execute the changes or **Cancel**.

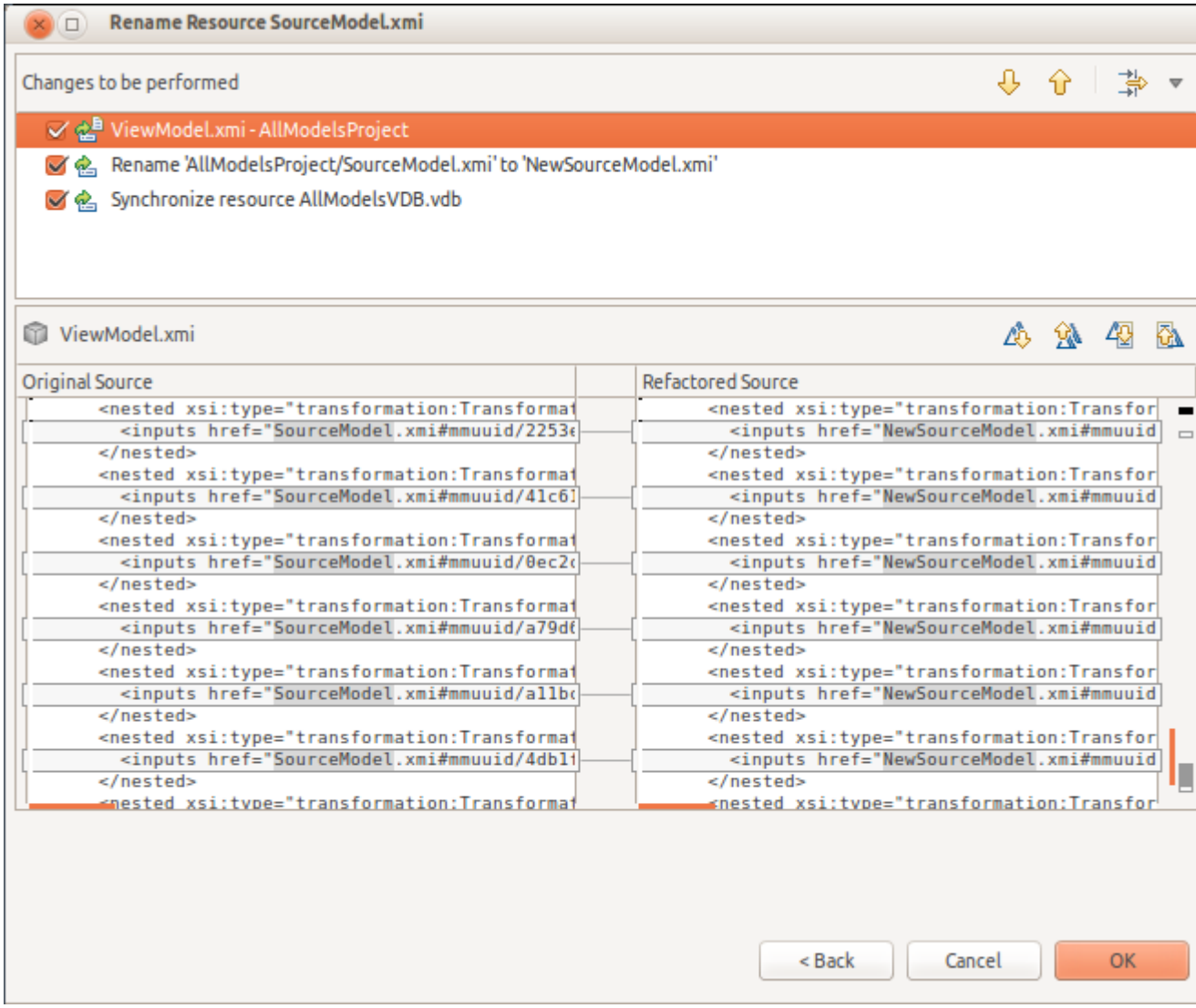


Figure 9.3. Preview Changes For Renamed Model

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 *Section D.2.1, "Model Explorer View"*.
  - **Step 2** - Right-click select the **Refactor > Move** action.

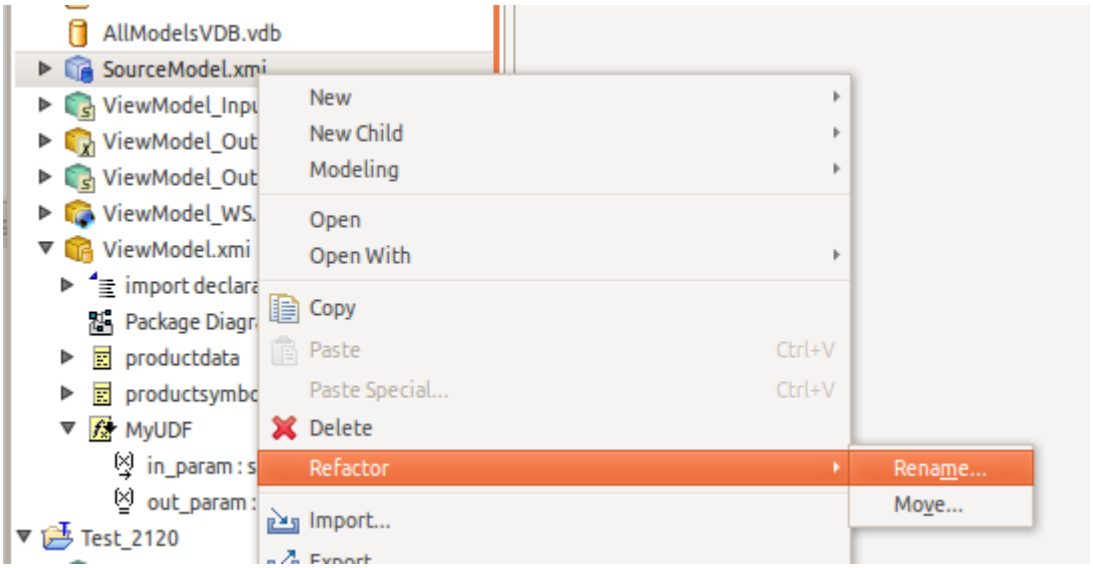


Figure 9.4. Refactor Move Action In Model Explorer

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

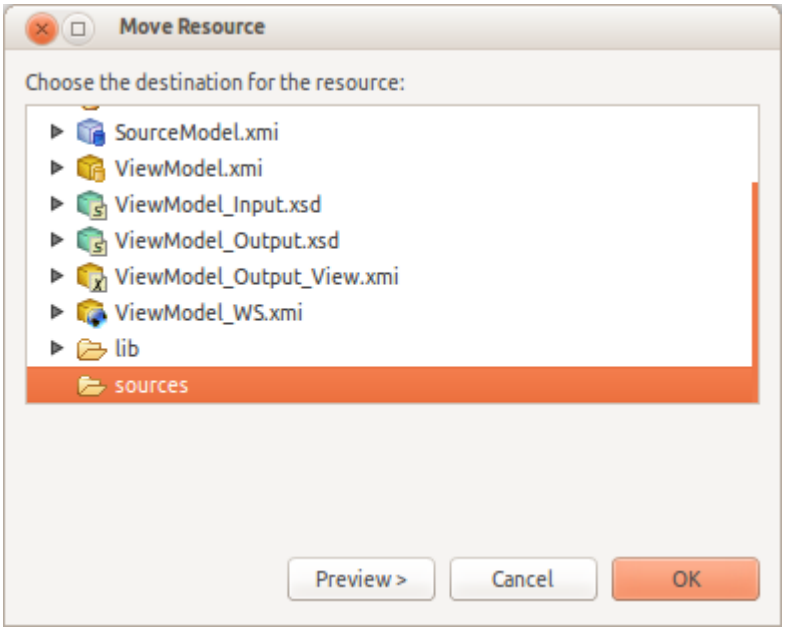


Figure 9.5. Move Model Dialog

- **Step 4 (Optional)** - The next **Move Resource File** dialog will preview the changes that will be applied as part of mvving your model. Click **OK** to execute the changes or **Cancel**.

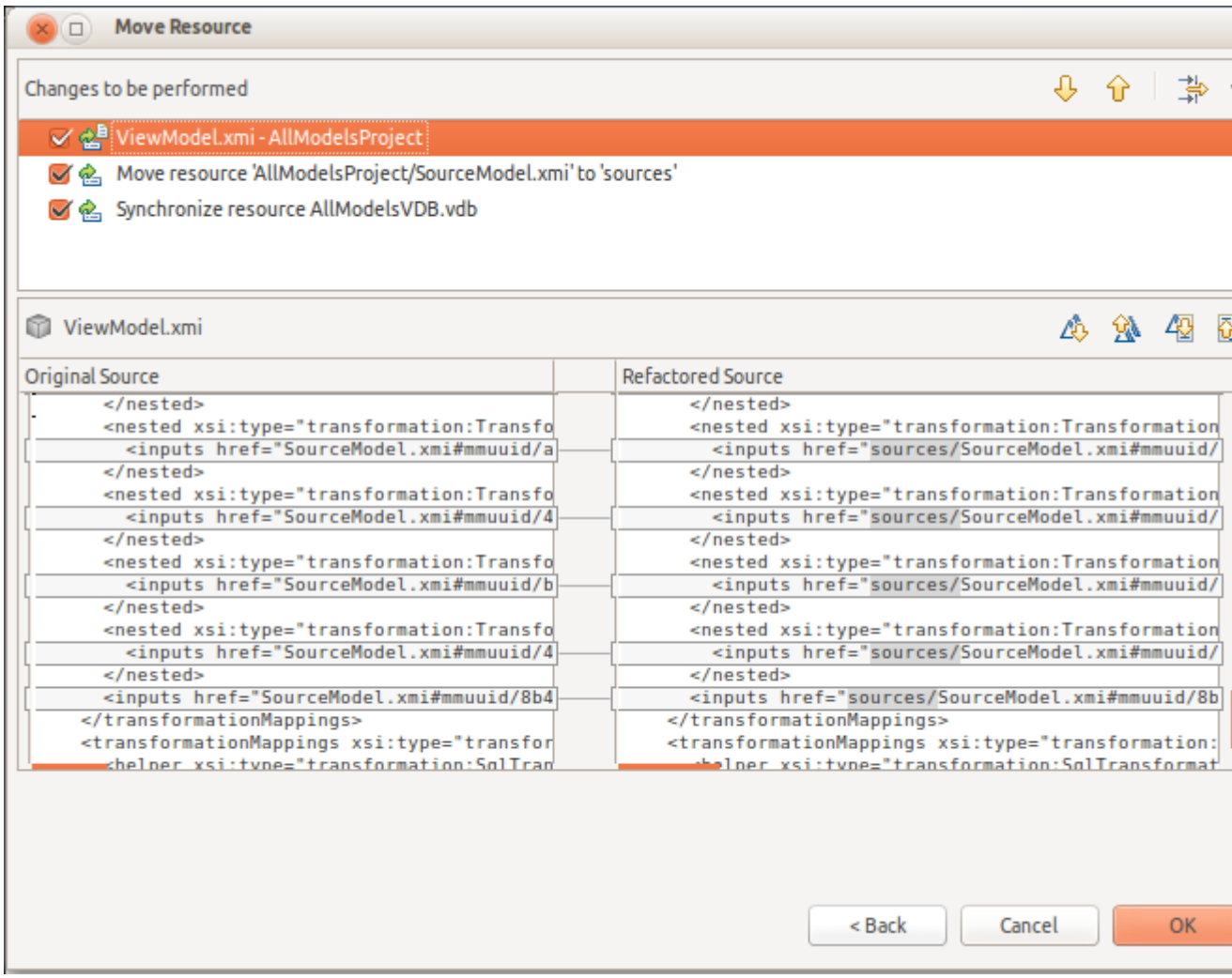


Figure 9.6. Preview Changes For Renamed Model

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 it's own unique ID, so copying a model will result in a 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 *Section D.2.1, "Model Explorer View"* or right-click select **Open** action.
  - **Step 2** - Select the editor tab for the model you opened.

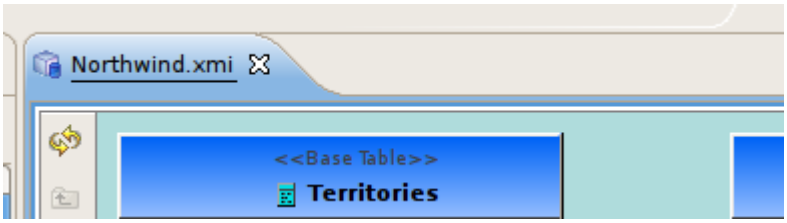


Figure 9.7. Select Editor Tab

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

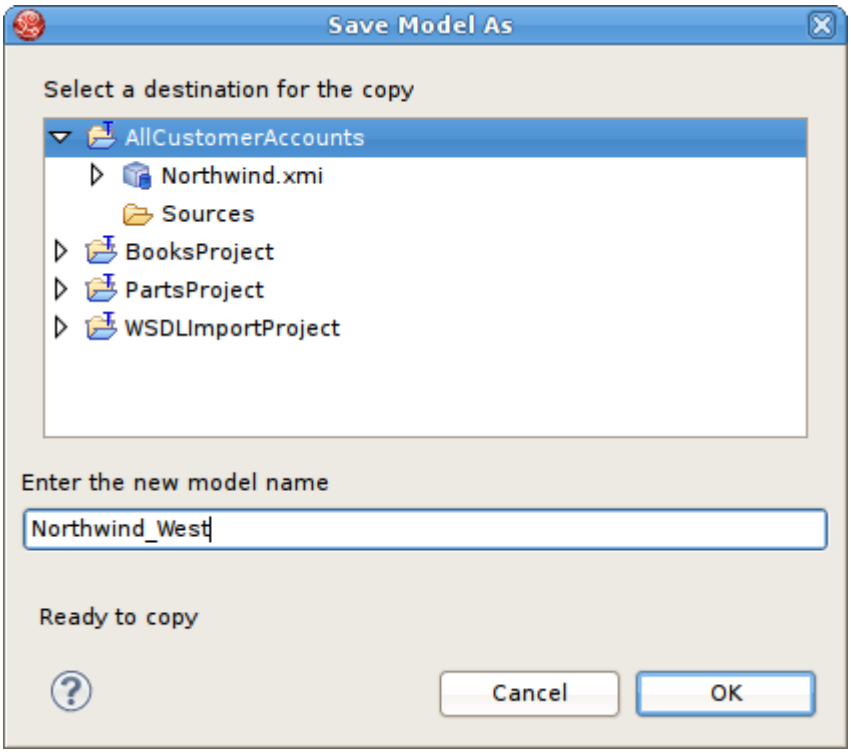


Figure 9.8. Save Model As Dialog

  - **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.

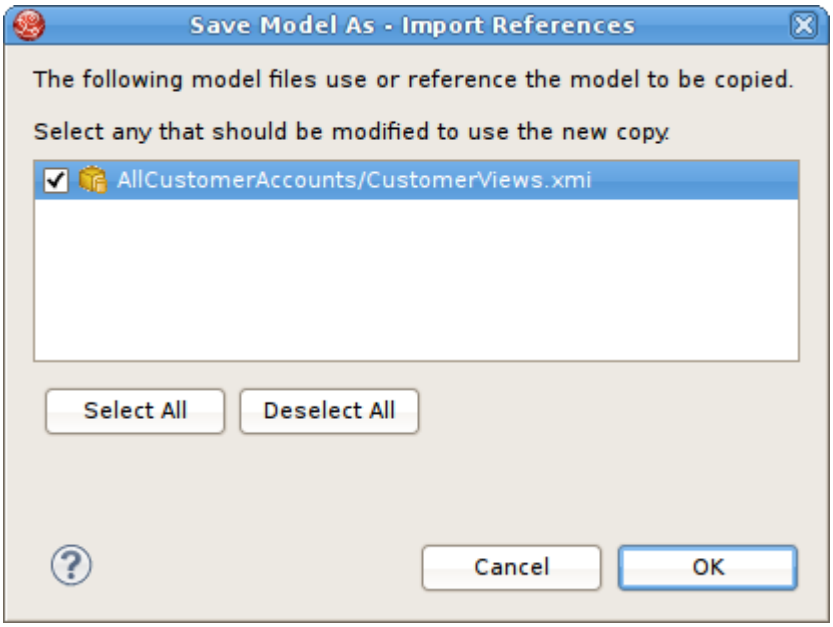


Figure 9.9. Save Model As Dialog

9.4. Clone Project

Because each instance of 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 *Section D.2.1, "Model Explorer View"*.
  - **Step 2** - Right-click, then select **Model Project > Clone** in the context menu. Otherwise you can select the **Project > Clone Project** action, located in Teiid Designer's main menu bar.

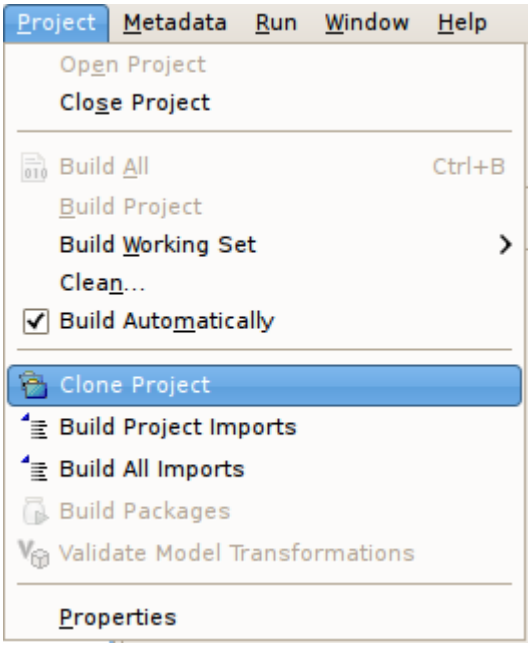


Figure 9.10. Clone Project In Project Menu

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

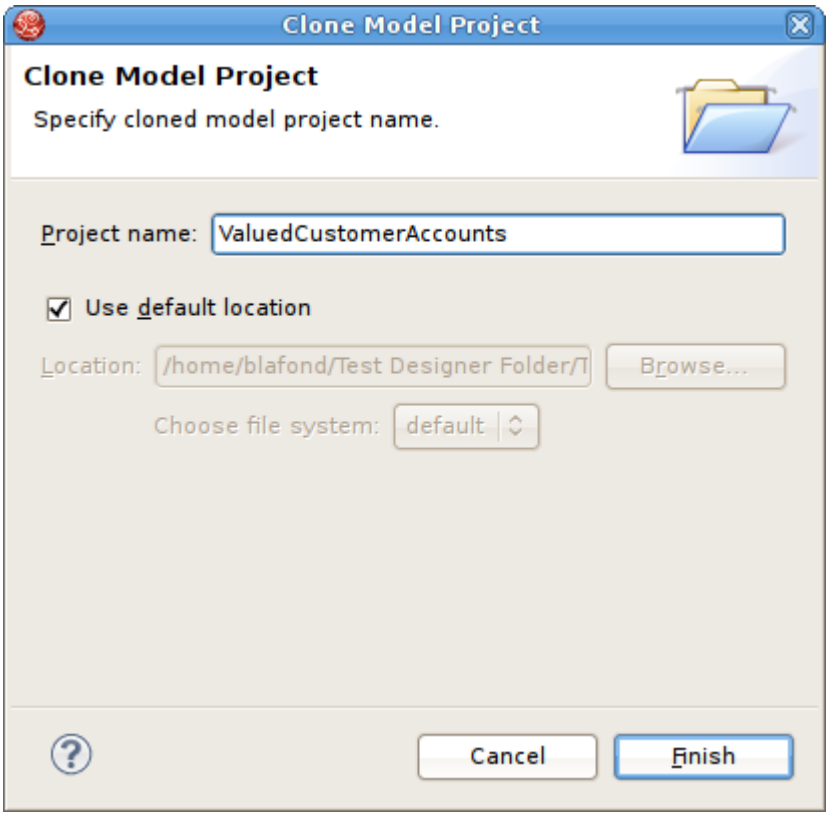


Figure 9.11. Clone Project In Project Menu

- **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.

Managing VDBs

As stated in the introduction, the critical artifact for Teiid Designer the VDB, or Virtual DataBase. This section describes the details of how to create, edit, deploy and test your VDBs.

10.1. Creating a VDB

To create an empty VDB launch Eclipse's **New** wizard, open the **Teiid Designer** category folder and select **Teiid VDB**. You can also select one or more models in a model project, right-click and select **New > Teiid VDB** action.>

Launching this wizard will open the New VDB dialog. If you launched with one or more models selected the dialog will contain the pre-selected models for inclusion in the new VDB.

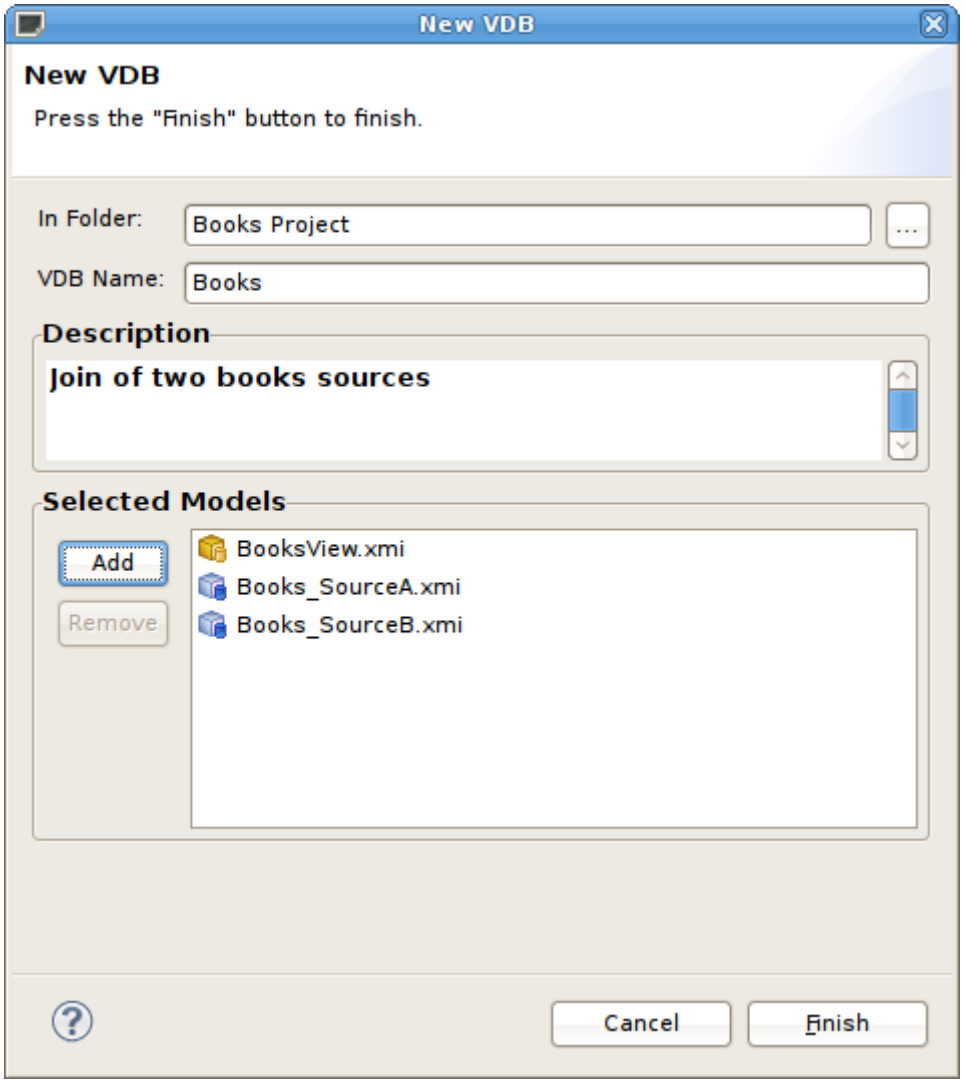


Figure 10.1. New VDB Dialog

Note

A VDB is scoped to be aware of models and files within the same model project as the VDB. You will not be allowed to add models to a VDB that exist in a different project.

10.2. Editing a VDB

To Edit an existing VDB, select the VDB in the explorer and right-click select Open action or simply double-click the VDB. The VDB will be opened in a VDB Editor. (See the *Section D.3.2, "VDB Editor"* section)

10.3. Test a VDB

For details on how to test your VDB, see *Section 11.3, "Testing With Your VDB"* section

10.4. Multi-source Binding Support

Teiid Designer now supports the Teiid feature of defining relational source models and binding them to multiple data sources.

Multi-source models can be used to quickly access data in multiple sources with homogeneous metadata. When you have multiple instances using identical schema, Teiid can help you gather data across all the instances, using "multi-source" models. In this scenario, instead of creating/importing a model for every data source, one source model is defined to represents the schema and is configured with multiple data "sources" underneath it. During runtime when a query issued against this model, the query engine analyzes the information and gathers the required data from all sources configured and gathers the results and provides in a single result. Since all sources utilize the same physical metadata, this feature is most appropriate for accessing the same source type with multiple instances

The VDB editor's **Models** tab now contains a simplified model table on the left and a new tabbed panel on the right containing **Model Details** and **Source Binding Definition** tabs. Click the **Multi-source** check box if you wish to add additional source bindings. Note that each binding must be defined with a unique **Source Name** as well as unique **JNDI Name** representing a deployed data source you your server.

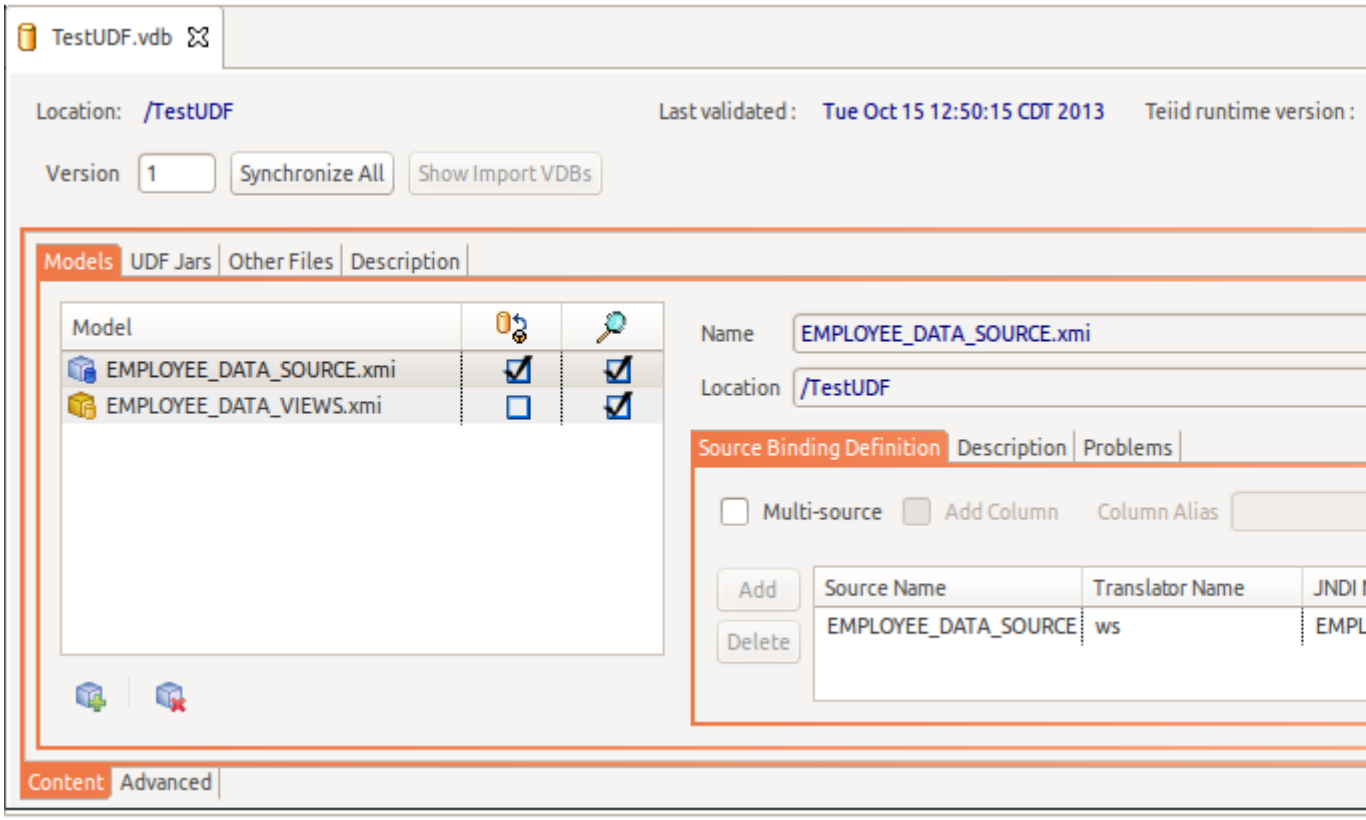


Figure 10.2. VDB Editor's Source Binding Definition

10.5. UDF support

In Teiid Designer you can create, manage and use User Defined Functions (UDFs). These functions allow you to perform simple or complex java operations on your data during runtime. This is accomplished by deploying your custom UDF jars on your server and creating a scalar function representation of your function method to use in your view transformation. In the VDB Editor, you have the option of including your UDF jars as part of the VDB artifact. If included in the VDB, the jars will automatically be deployed to the server for you when the VDB is deployed.

The figure below illustrates a sample project setup which includes a UDF jar in a lib folder under a project. When a model defining a UDF is added to a VDB, each function is interrogated and it's referenced UDF jar (if available) is added to the VDB as well as shown in the UDF Jars tab in the editor.

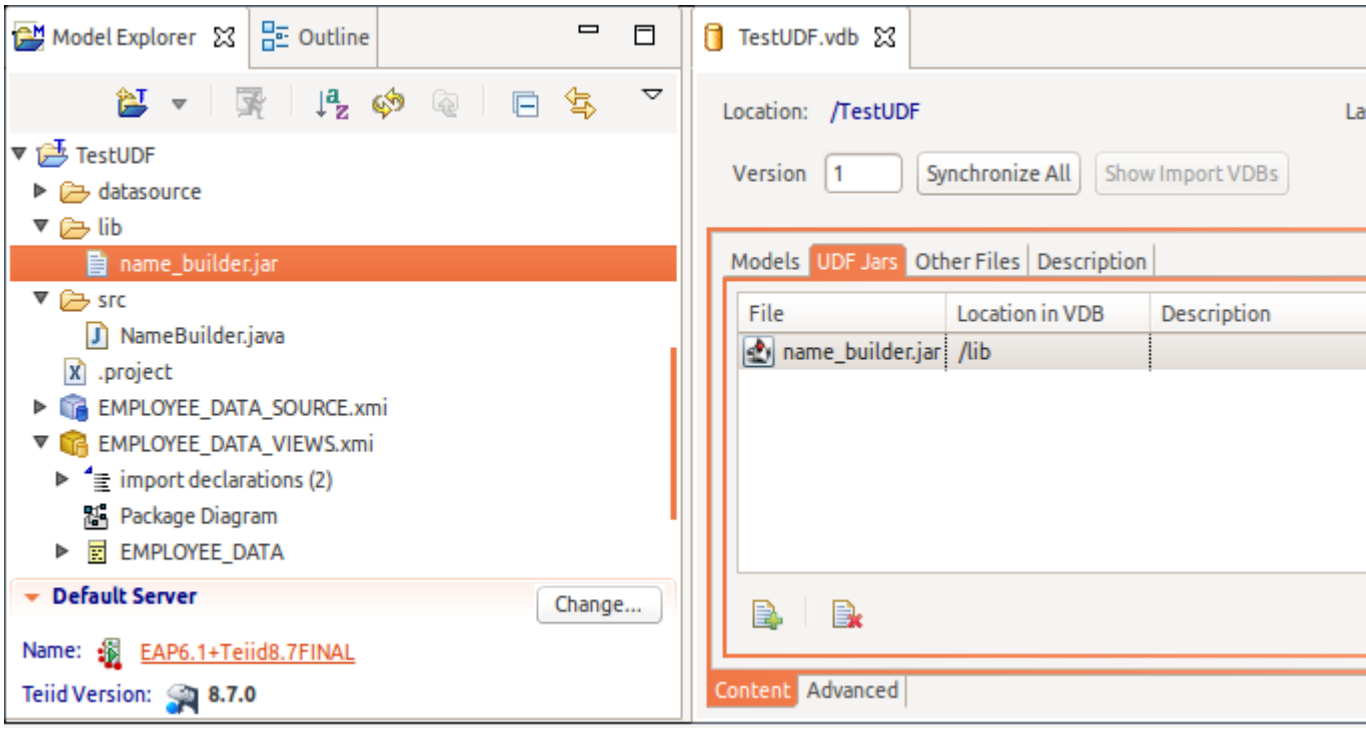


Figure 10.3. UDF Jar In Project and VDB

10.6. Reusing VDBs

Teiid 8.1 introduced the ability to treat your deployed VDB as just another database where the database category is your VDB name and each visible model in your VDB is treated as a schema. This is accomplished via a new <import-vdb> element in the vdb.xml definition. (see *Teiid VDB Reuse section* [https://docs.jboss.org/author/display/TEIID/VDB+Reuse]). By allowing VDB's to referenced other VDBs, users can create reusable database components and reduce the amount of modeling required to create complex transformations.

The sample vdb.xml file below highlights the <import-vdb> element and the corresponding import-vdb-reference within the view model's <model> element.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb version="1" name="PartssupplierViewsVDB">
    <property value="false" name="preview"/>
    <import-vdb import-data-policies="false" version="1"
 name="PartssupplierSourcesVDB"/>
    <model visible="true" type="VIRTUAL" name="PartsViewModel" path="/
PartssupplierProject/PartsViewModel.xmi">
        <property value="1623826484" name="checksum"/>
        <property value="Relational" name="modelClass"/>
        <property value="false" name="builtIn"/>
        <property value="655076658.INDEX" name="indexName"/>
```

```
        <property value="PartssupplierSourcesVDB" name="import-vdb-
reference"/>
    </model>
</vdb>
```

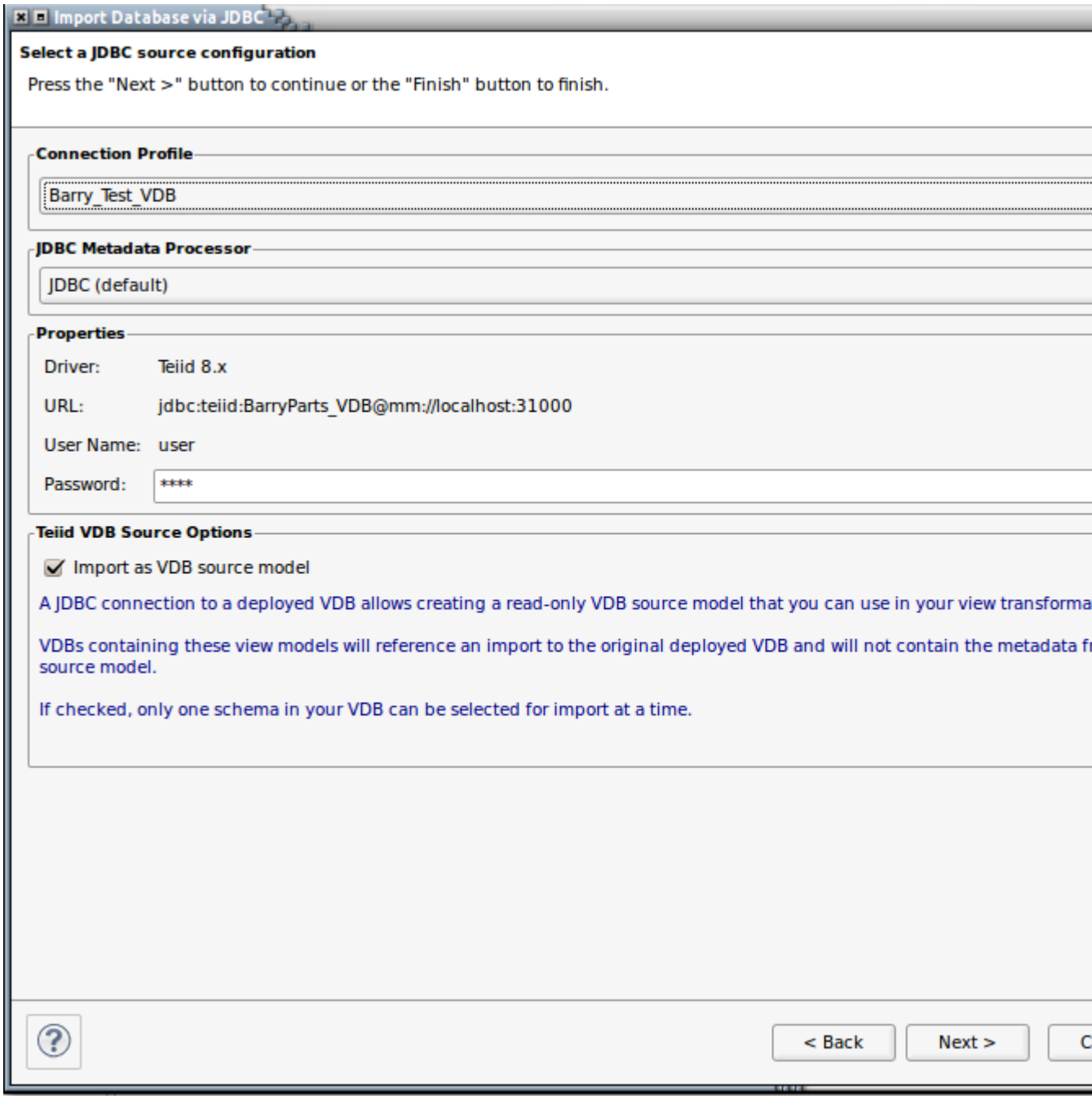
Teiid Designer exposes this capability by allowing users to import metadata from deployed VDBs via the JDBC Import option. Through this import, relational VDB source models are created which structurally represent the Catalog (VDB), Schema (Model) and Tables in Virtual DataBase.

When dealing with the these VDB source models there are some limitations or rules, namely:

- VDB source models are read-only
- VDB source model name is determined by the deployed model name (schema) from the VDB it was imported from
- Model names have to be unique within a model project
  - VDB source models have to be imported/created in a project different than the project used to create and deploy the Reuse VDB
- The JDBC Import Wizard will restrict your options to comply with these rules

To create a VDB source model:

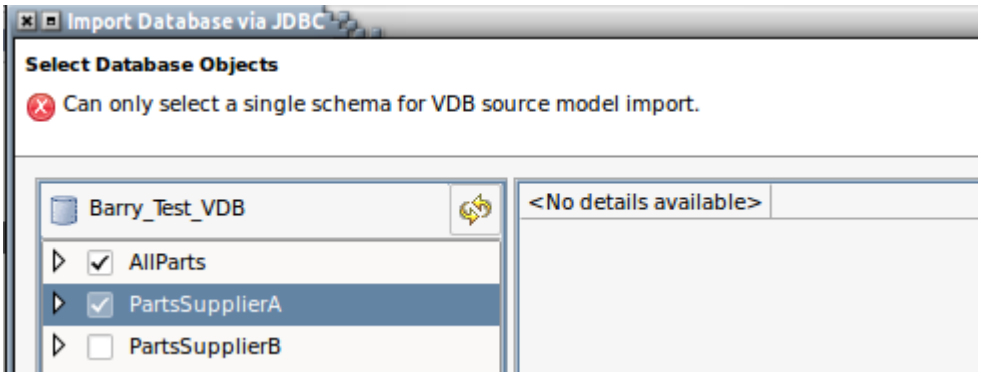
- **Step 1** - Deploy your VDB
- **Step 2** - Launch the JDBC Import Wizard via the "**Import > Teiid Designer > JDBC Database >> Source Model**" action
- **Step 3** - On the first page of the wizard create/select a valid connection profile for your deployed VDB.
  - The wizard will detect that the connection profile is a Teiid VDB connection and a section will be displayed on the wizard page titled **Teiid VDB Source Options**
  - If **Import as VDB source model** is NOT checked, then the wizard will continue importing as a normal JDBC import



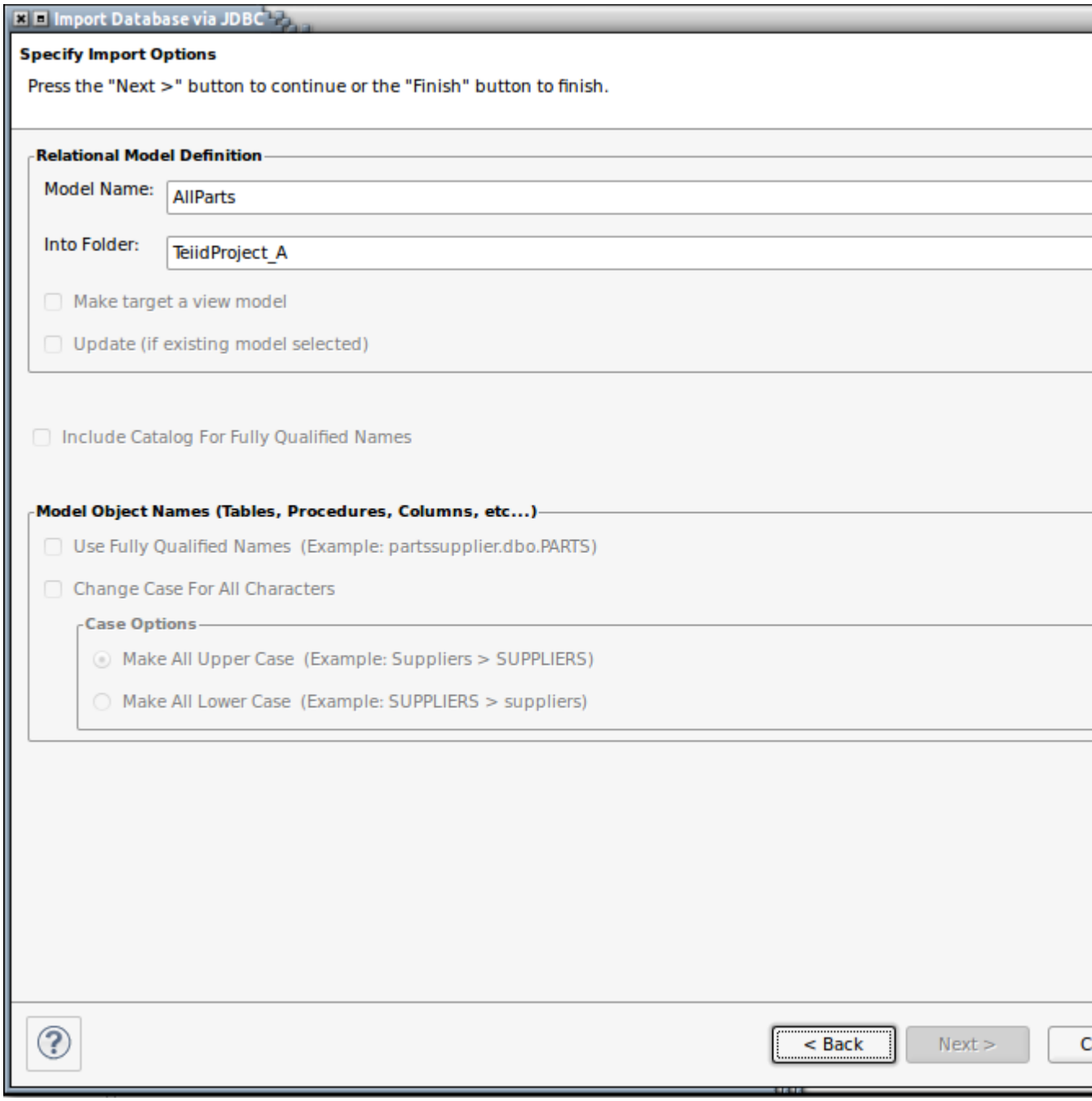
- **Step 4** - On the 3rd page, titled **Select Database Objects**, select a single schema to use to create as VDB source model.

Note

The schema names are the names of the visible models in your deployed VDB.

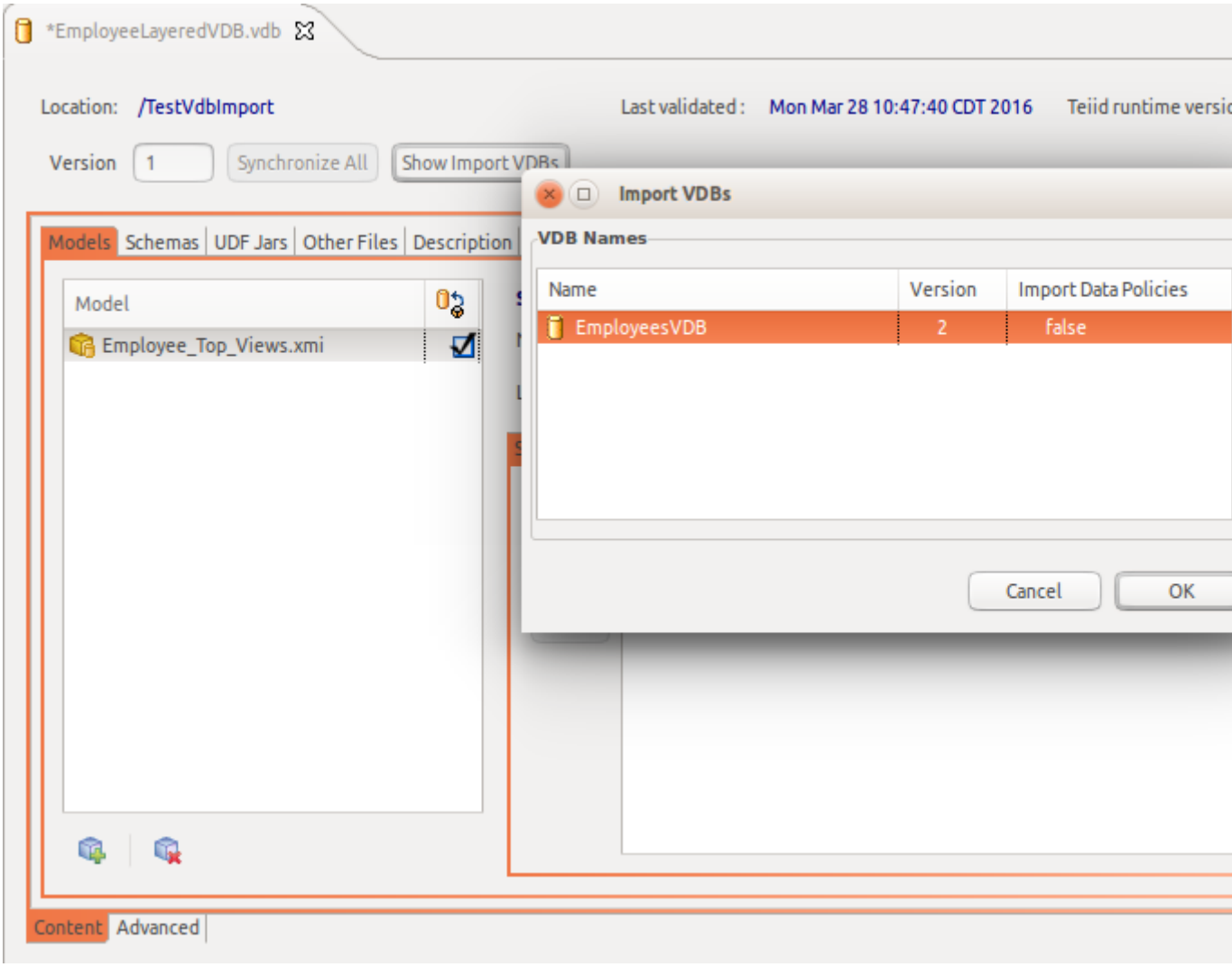


- **Step 5** - The final page shows the name of the resulting VDB source model and the name is NOT editable.
  - All other options are disabled
  - The target **Into Folder** must NOT contain a model with the same name or the **Finish** button will be disabled



You can use your VDB source model like any other source model in your project. VDB source model tables can be used in your transformation queries and the view models will contain model imports to your VDB source models. However, when your view model is added to a VDB, any referenced VDB source models do NOT get added to your VDB. Instead, an <import-vdb> element (described above) reference is added in it's place.

If VDB imports exist for a VDB, the **Show Import VDBs** button will be enabled and allow viewing the names of the imported VDBs as shown below. Note that the dialog now supports editing of the version number for each referenced VDB.



10.7. Security and Data Access

You have some options on defining your data access security for your VDB via the VDB Editor (See the *Section D.3.2, "VDB Editor"* section).

The first level is provided by the model visibility check-box in the Models section (Spyglass column). If unchecked, that model and it's contents will not be returned by the Teiid runtime with the standard JDBC metadata.

The next level of security is provided defining permissions for your data roles which can be managed via the lower panel in the VDB Editor. For a unique data role, each model and most objects within that model can have specific values of data access including:

- Security (Row-based condition and column masking)
- Create
- Read

- Update
- Delete
- Execute
- Alter

The image below is an example of the **Permissions** defined for a data role.

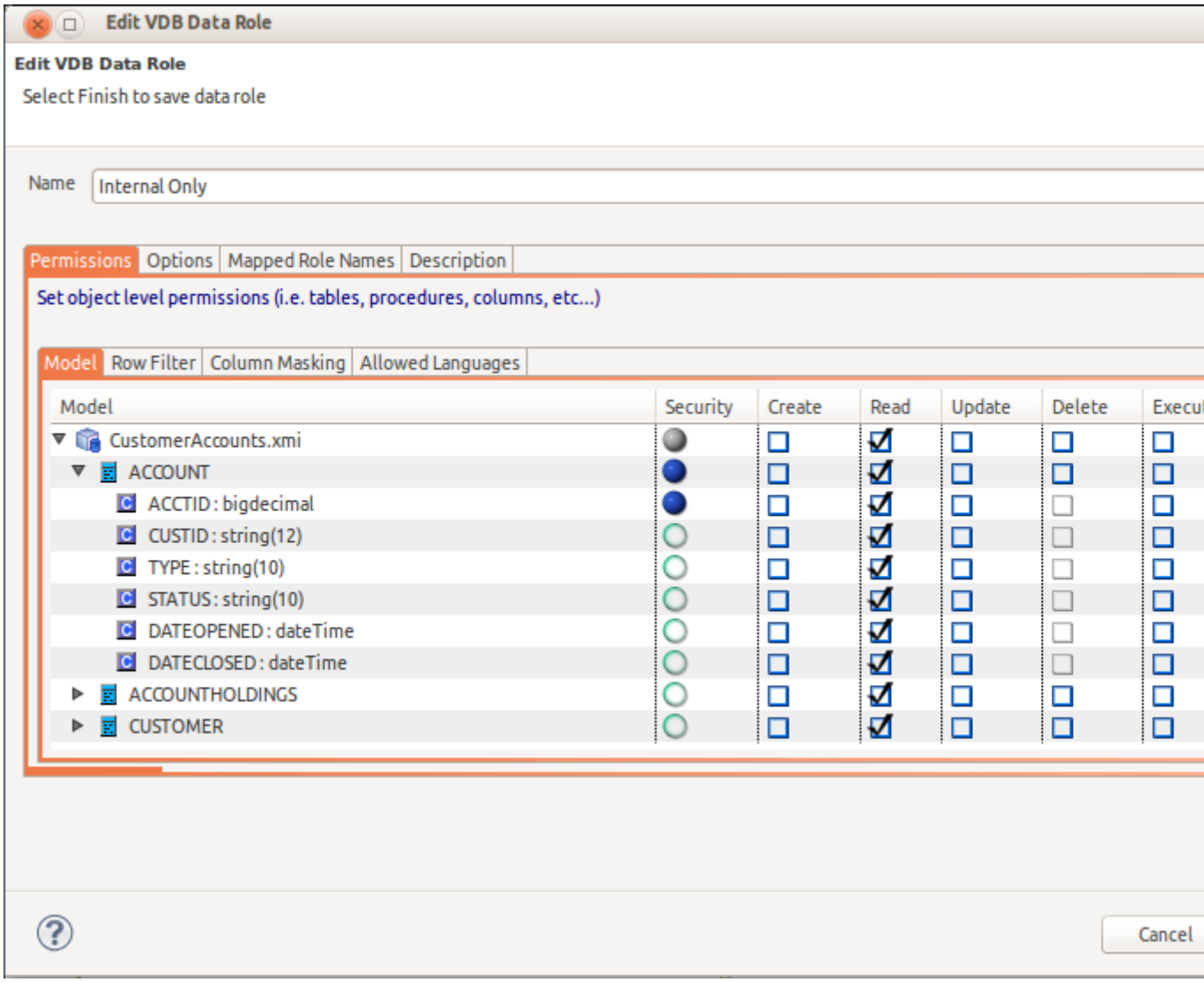


Figure 10.4. Permissions Section

Teiid provides 2 security concepts for object level permissions: (see *Permissions* [https://docs.jboss.org/author/display/teiid88final/Permissions])

- Row-Based Security is a permission against a fully qualified table/view/procedure may also specify a condition.

- Unlike the allow CRUD actions defined above, a condition is always applied - not just at the user query level. The condition can be any valid SQL referencing the columns of the table/view/procedure. Procedure result set columns may be referenced as proc.col. The condition will act as a row-based filter and as a checked constraint for insert/update operations.
- An example of a condition might be: **column1=user()**

- Column Masking is "a permission against a fully qualified table/view/procedure column may also specify a mask and optionally a condition."
  - When the query is submitted the roles are consulted and the relevant mask/condition information are combined to form a searched case expression to mask the values that would have been returned by the access. Unlike the CRUD allow actions defined above, the resulting masking effect is always applied - not just at the user query level. The condition and expression can be any valid SQL referencing the columns of the table/view/procedure. Procedure result set columns may be referenced as proc.col.
  - An example of a mask might be: **CASE WHEN column1=user() THEN column1 END**

You can define Row and Column Based security in Designer's Data Row Wizard. By double-clicking a target table, view, procedure or column in the Models table, the appropriate editor dialog will be displayed. Note those objects that already have existing security defined are highlighted in blue.

If a table, view or procedure is double-clicked, the Row Filter Definition dialog is displayed. Enter a valid SQL condition and specify whether or not this filter should be treated as a constraint or not.

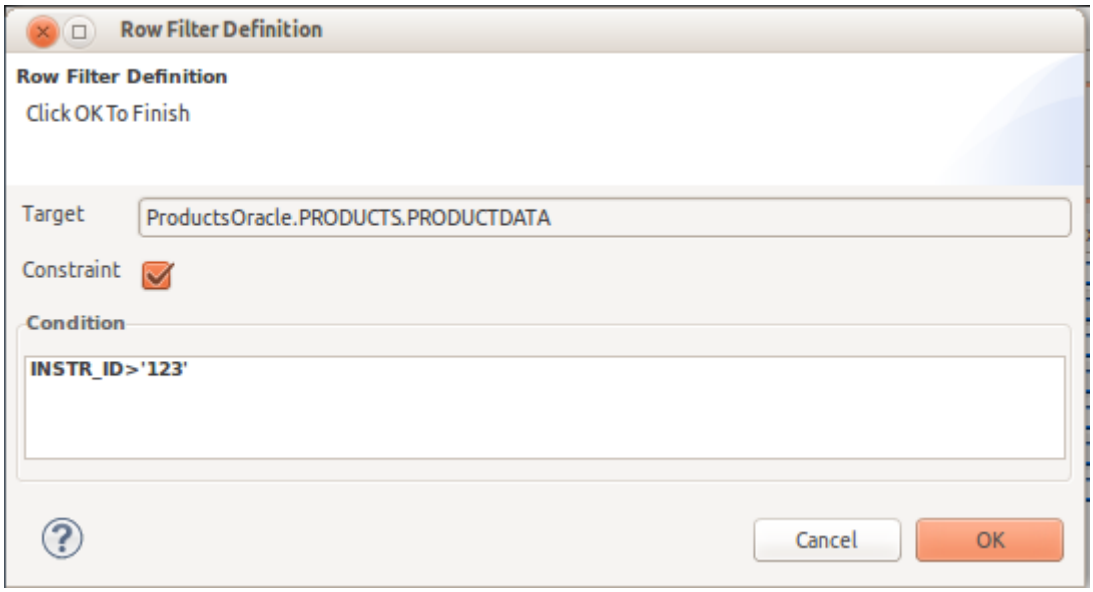


Figure 10.5. Row Filter Definition Dialog

If a column is double-clicked, the Row Filter Definition dialog is displayed. Enter valid column masking SQL expression, an optional order (see Teiid documentation) and an option condition expression.

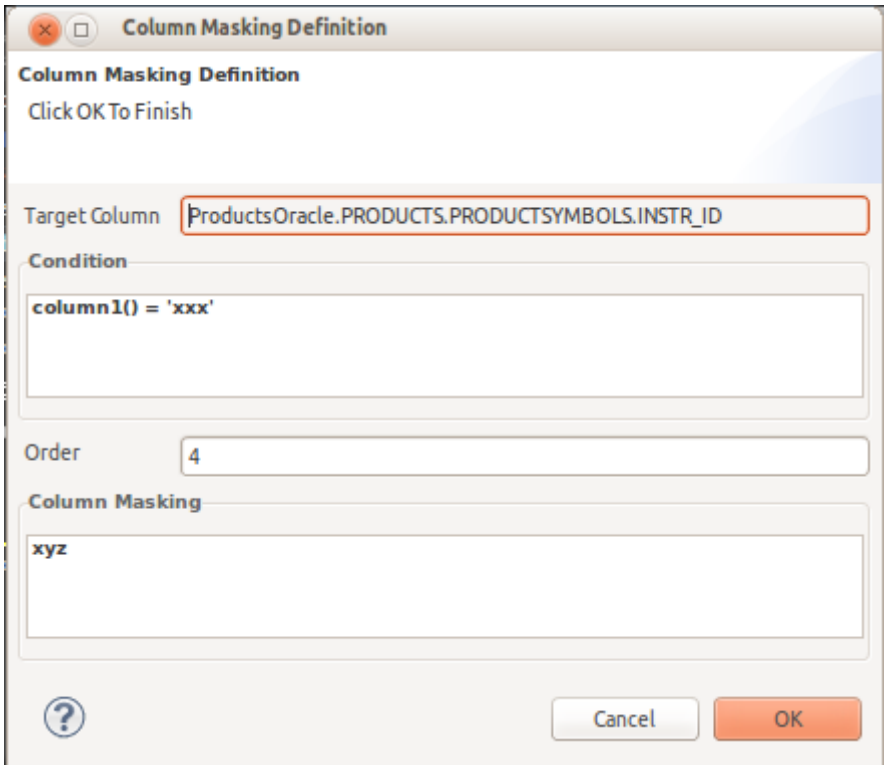


Figure 10.6. Column Masking Definition Dialog

You can also edit these these values via the **Add, Edit and Remove** buttons on the respective Row Filter and Column Masking tabs

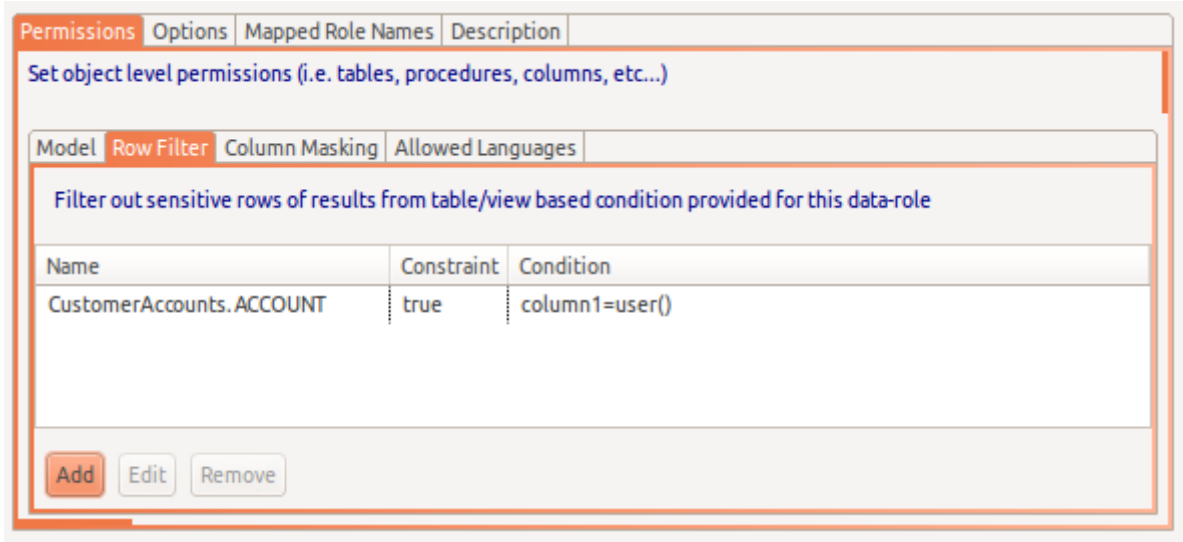


Figure 10.7. Row Filter Tab

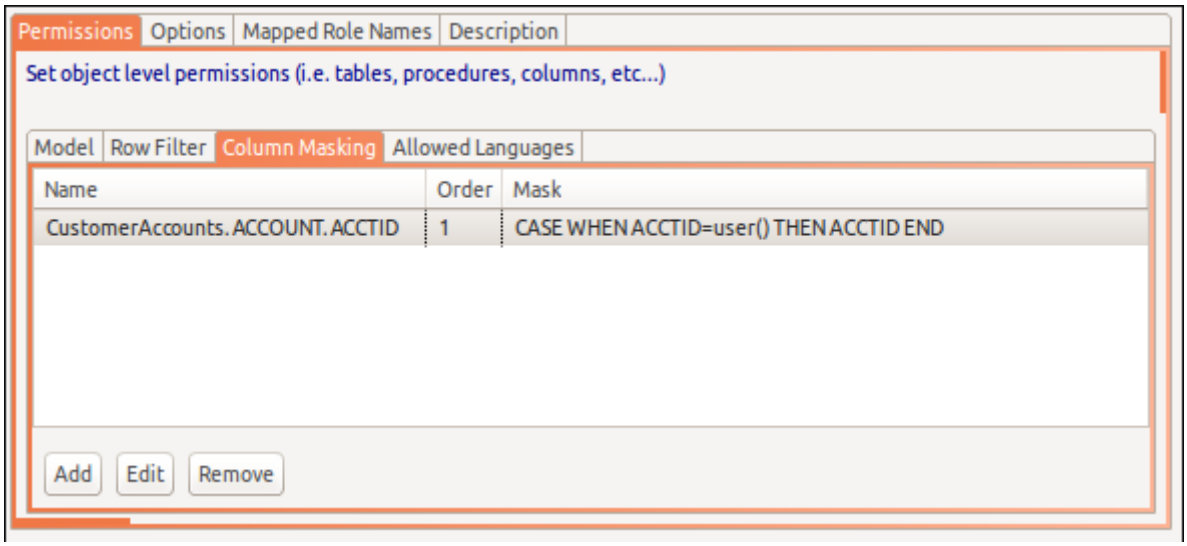


Figure 10.8. Column Masking Tab

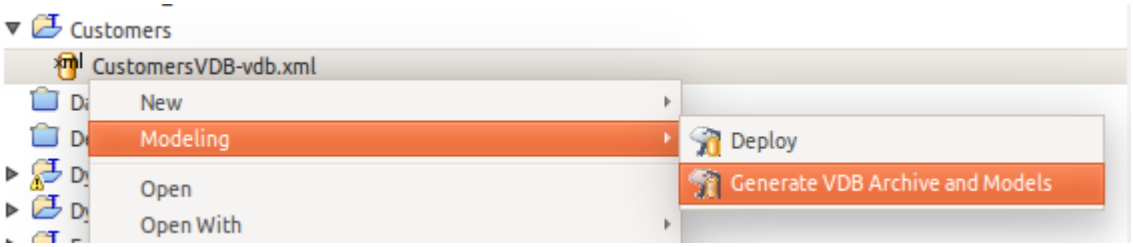
10.8. Working with Dynamic VDBs

This section desribes the Designer features which allow generating a VDB archive with XMI models from existing dynamic VDBs (*-vdb.xml) files and generating dynamic VDBs from existing VDB archives.

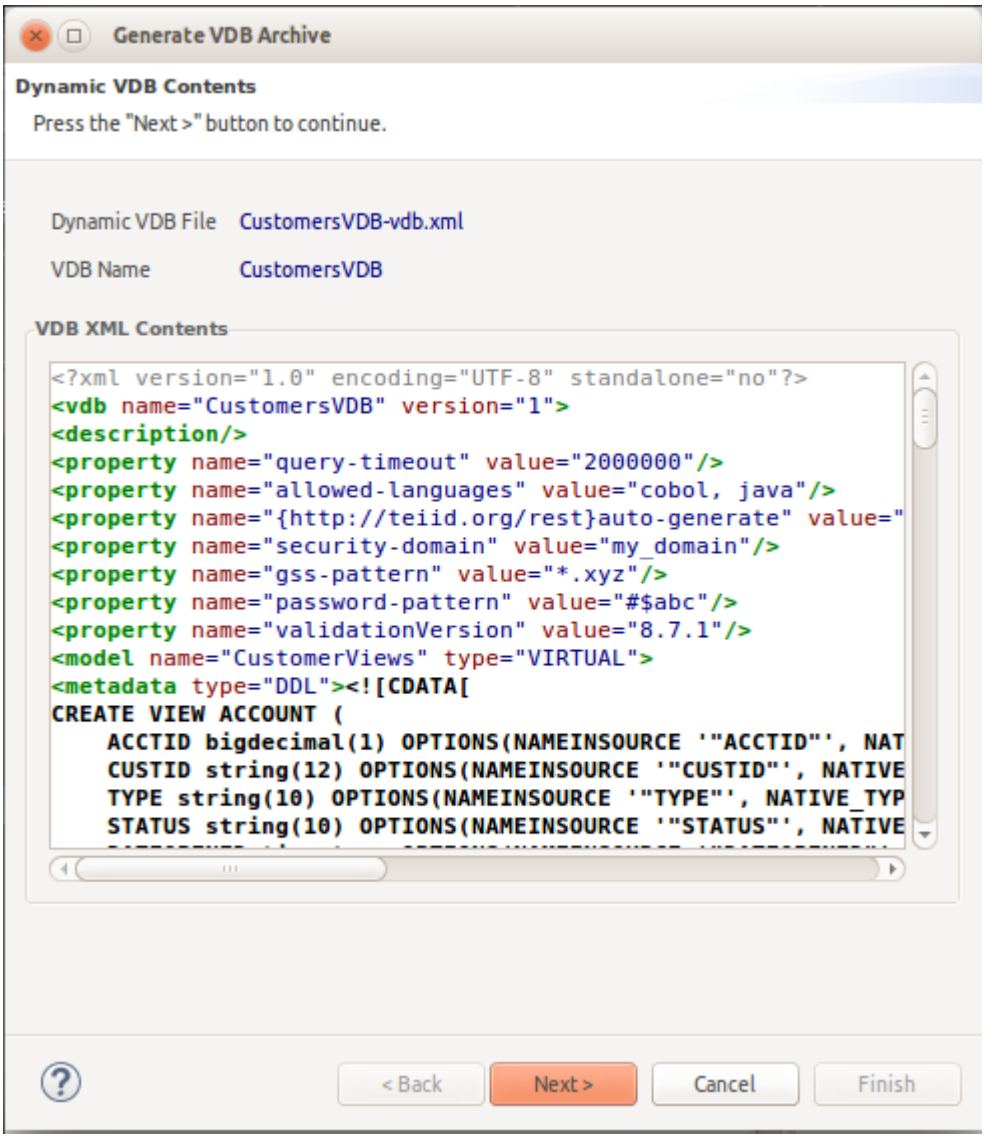
10.8.1. Generate VDB Archive

To generate a VDB archive and models from a dynamic vdb (*-vdb.xml) file in your workspace:

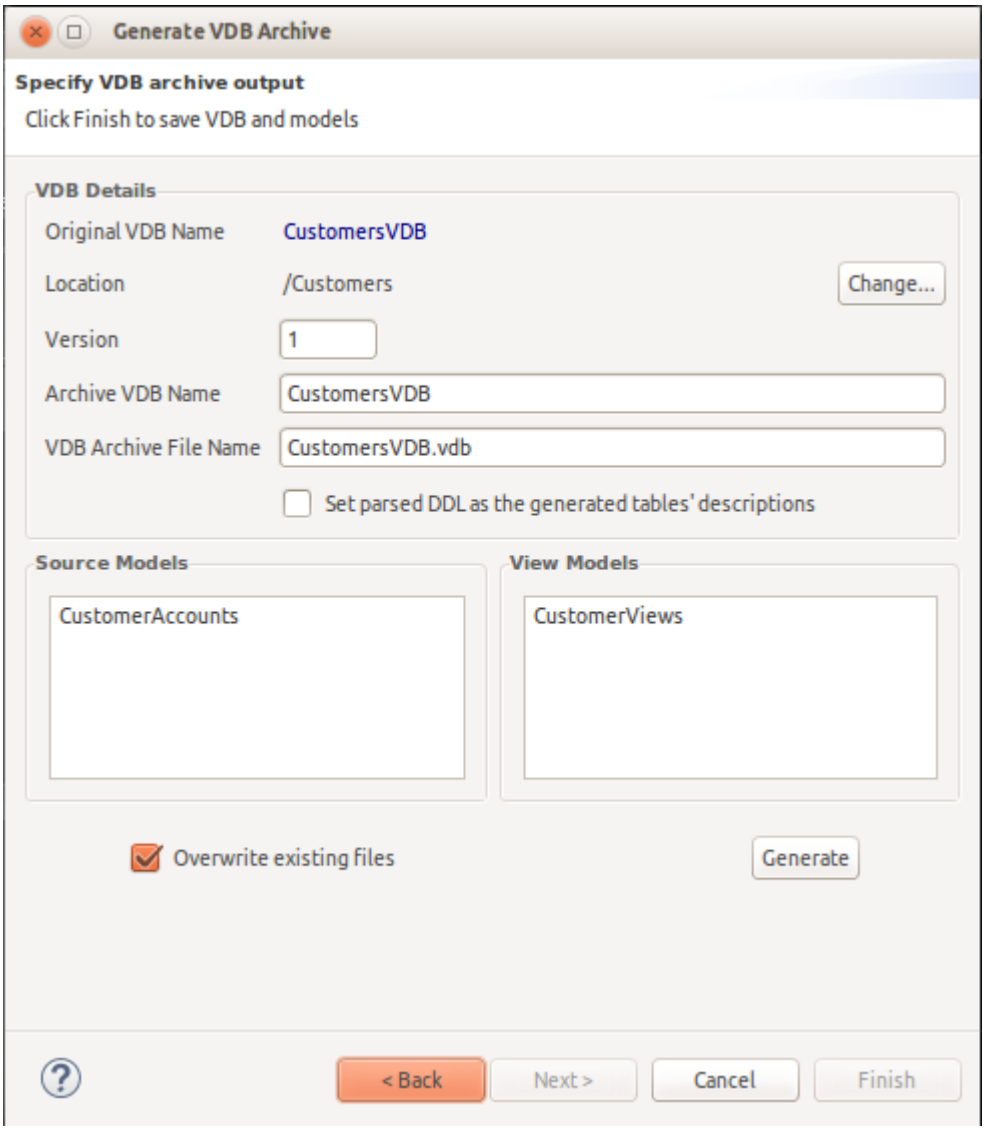
- **Step 1** - Import (or drag and drop) a dynamic vdb file into a model project or folder
- **Step 2** - Select the dynamic vdb file in the **Model Explorer**, then right-click and select the **Modeling > Generate Archive VDB and Models** action.



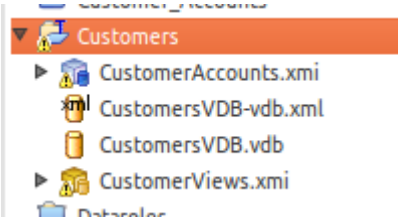
- **Step 3** - The first page of thw wizard will show the **Dynamic VDB Contents** Verify the contents and click **Next >**



- **Step 4** - On the second page, specify details and options for yor VDB archive including actual VDB name, VDB archive file name and target location (project or folder). Click **Generate** to create the VDB archive and associated models in your target location. Then click **Finish**



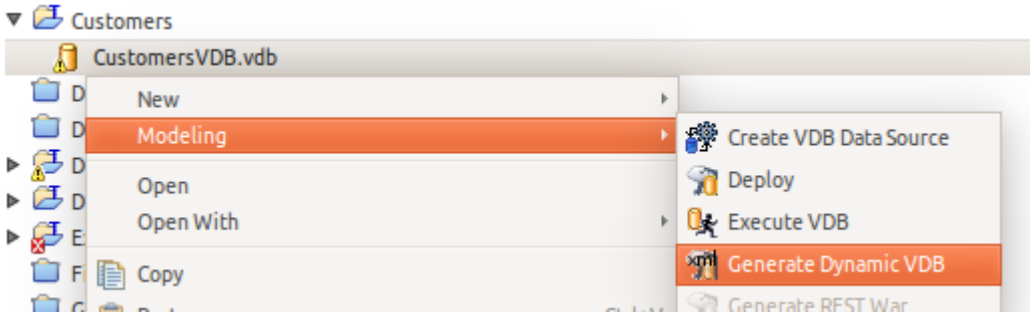
The generated VDB and XMI models will now be visible in the **Model Explorer** view.



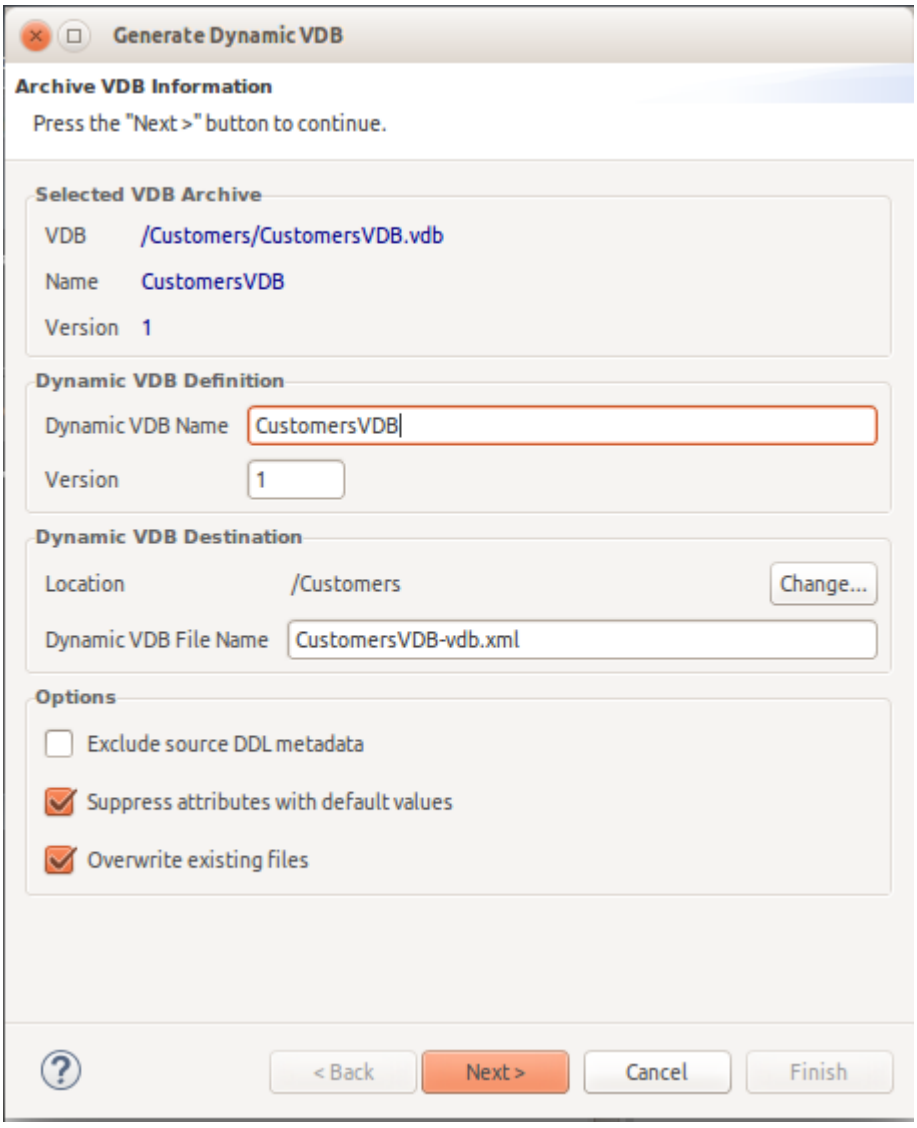
10.8.2. Generate Dynamic VDB

To generate a dynamic VDB from a VDB archive (*.vdb) in your workspace:

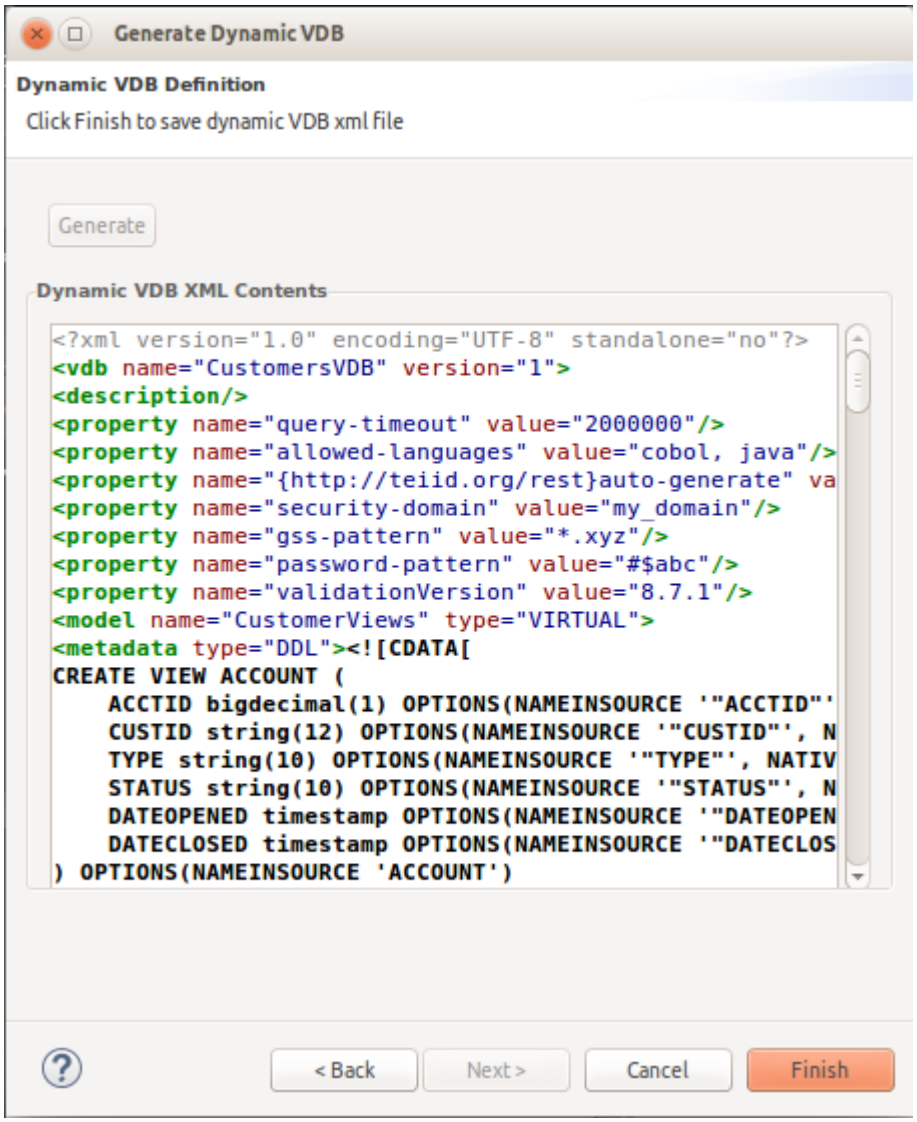
- **Step 1** - Select a VDB archive file in the **Model Explorer**, then right-click and select the **Modeling > Generate Dynamic VDB** action.



- **Step 2** - The first page of thw wizard will show the selected VDB archive file information. You can also set the VDB name adn version as well as specify the target location and actual file name (*-vdb.xml). Click **Next >**



- **Step 3** - On the second page, click the **Generate** button. The XML contents will be displayed in the viewer which represents the same basic metadata from the origional VDB. Click **Generate** to create the VDB archive and associated models in your target location. Then click **Finish** to save the XML string to the file specified in Step 2.


Generate Dynamic VDB
Dynamic VDB Definition
Click Finish to save dynamic VDB xml file
Generate
Dynamic VDB XML Contents
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<vdb name="CustomersVDB" version="1">
<description/>
<property name="query-timeout" value="2000000"/>
<property name="allowed-languages" value="cobol, java"/>
<property name="{http://teiid.org/rest}auto-generate" va
<property name="security-domain" value="my_domain"/>
<property name="gss-pattern" value="*.xyz"/>
<property name="password-pattern" value="#$abc"/>
<property name="validationVersion" value="8.7.1"/>
<model name="CustomerViews" type="VIRTUAL">
<metadata type="DDL"><![CDATA[
CREATE VIEW ACCOUNT (
ACCTID bigdecimal(1) OPTIONS(NAMEINSOURCE '"ACCTID"'
CUSTID string(12) OPTIONS(NAMEINSOURCE '"CUSTID"', N
TYPE string(10) OPTIONS(NAMEINSOURCE '"TYPE"', NATIV
STATUS string(10) OPTIONS(NAMEINSOURCE '"STATUS"', N
DATEOPENED timestamp OPTIONS(NAMEINSOURCE '"DATEOPEN
DATECLOSED timestamp OPTIONS(NAMEINSOURCE '"DATECLOS
) OPTIONS(NAMEINSOURCE 'ACCOUNT')
< Back
Next >
Cancel
Finish

Testing Your Models

As described briefly in *Section 1.3.8, "Testing Your Models"*, 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.

11.1. Manage Connection Profiles

Teiid Designer utilizes the Eclipse Data Tools Platform (DTP) Connection Profile framework for connection management. Connection Profiles provide a mechanism to connect to relational and non-relational 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 Teiid server contents in the *Section 3.1, "Setting up a Server"* 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.

11.1.1. 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.

11.1.2. 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.

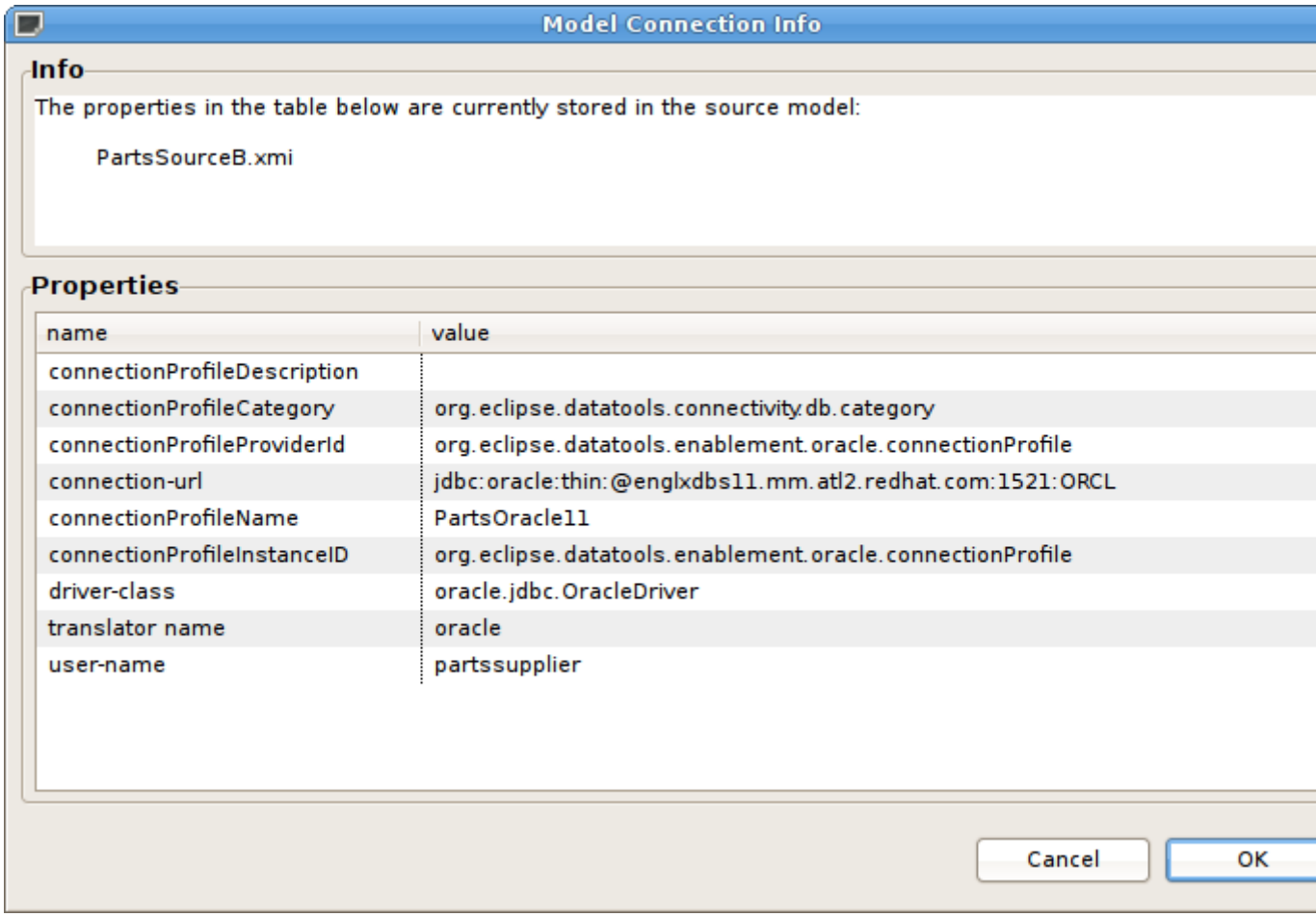
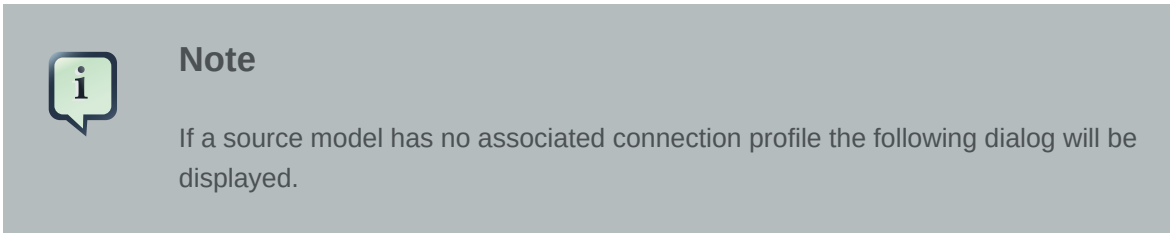


Figure 11.1. Connection Profile Information Dialog



Note

If a source model has no associated connection profile the following dialog will be displayed.

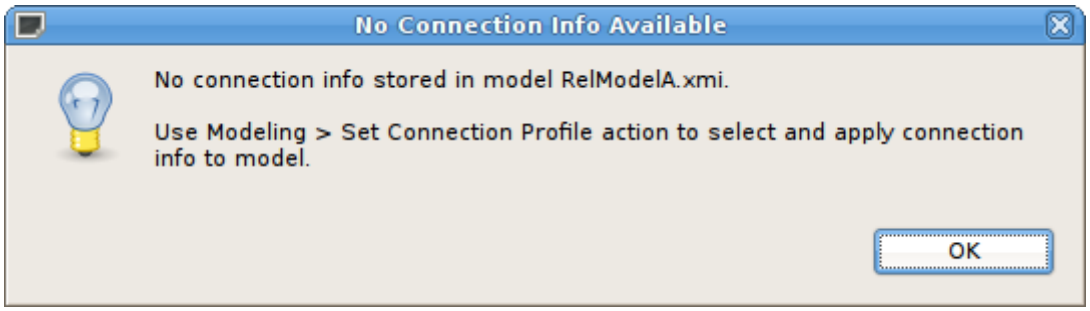


Figure 11.2. No Connection Info Dialog

11.1.3. 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.

11.2. 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 *Section 11.2.5, "Sample SQL Results for Preview Data"* view. This action is accessible throughout the Teiid Designer in various view toolbars and context menus.

There are two requirements for previewing your data:

1. The selected object must be one of several previewable model object types

2. 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

Any virtual table, view or procedure is previewable as long as all "physical" source models reference sufficient connection info. (See ??? view)

After selecting the Preview Data action, Designer will insure that all source models have corresponding deployed data sources .

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.

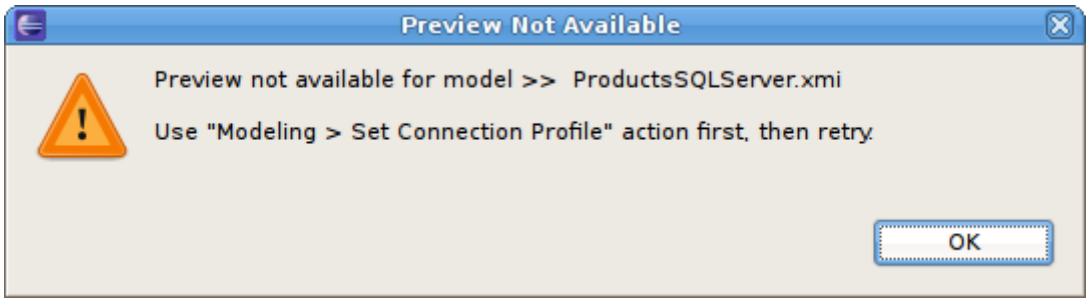


Figure 11.3. Preview Not Available

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

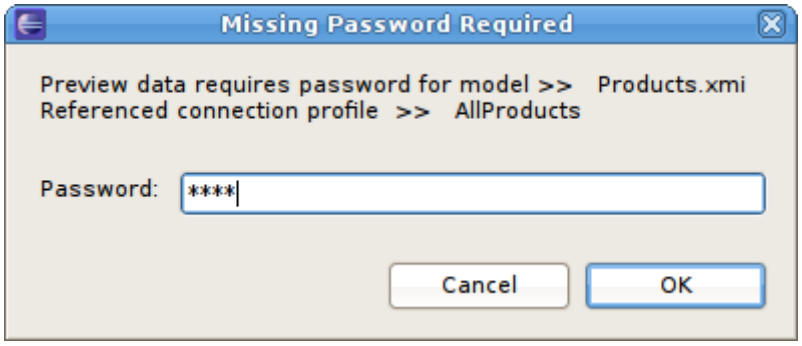


Figure 11.4. Missing Password

In order to preview, the Teiid Runtime requires deployed metadata representing your source and view metadata. If all required data sources are deployed the Preview Data dialog will be displayed containing 2 tabs to view the simple SQL Query to be executed as well as the VDB Xml Content that will be temporarily deployed in order to execute the query. Clicking OK will execute the SQL query.

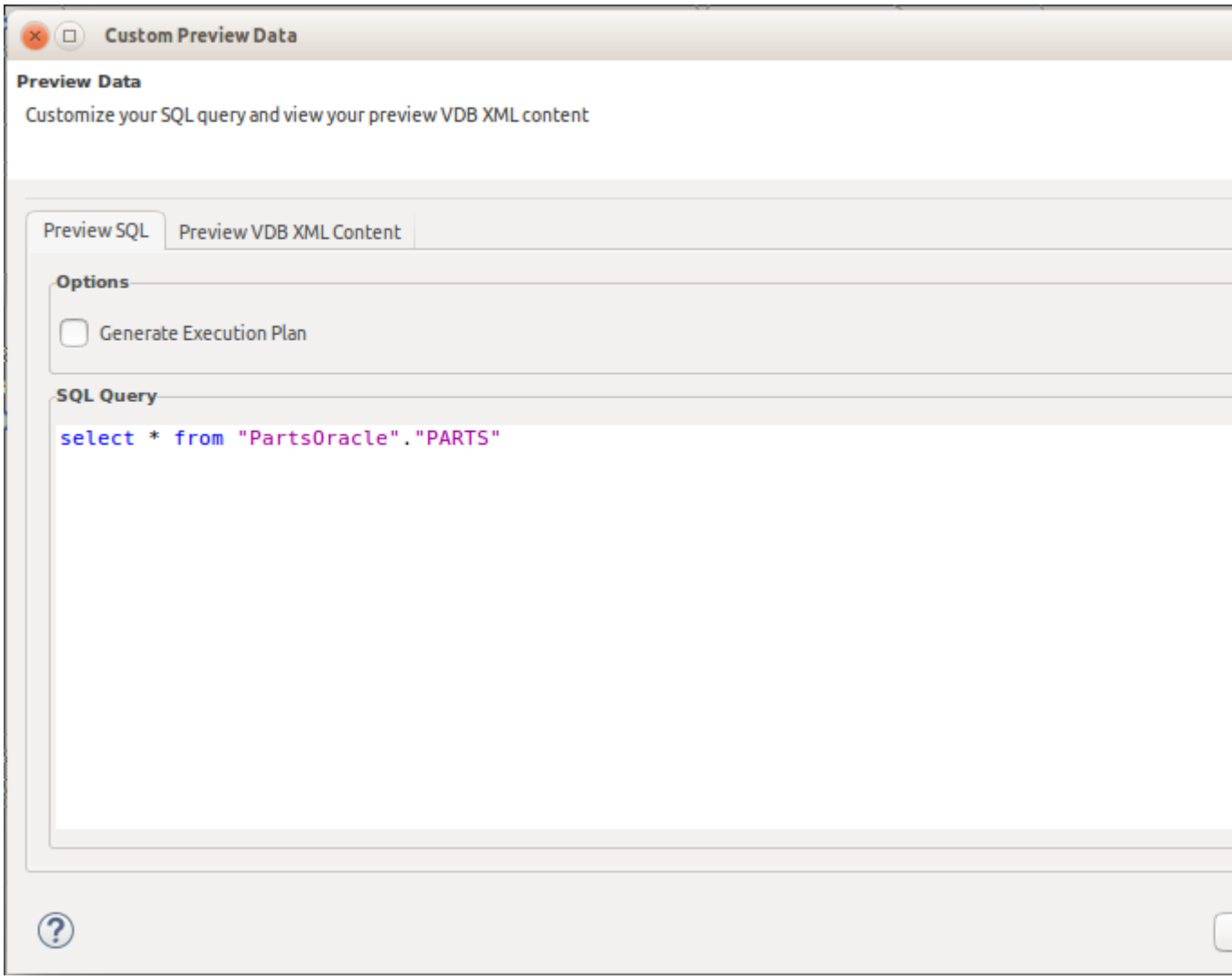


Figure 11.5. Preview Data: SQL Query Tab

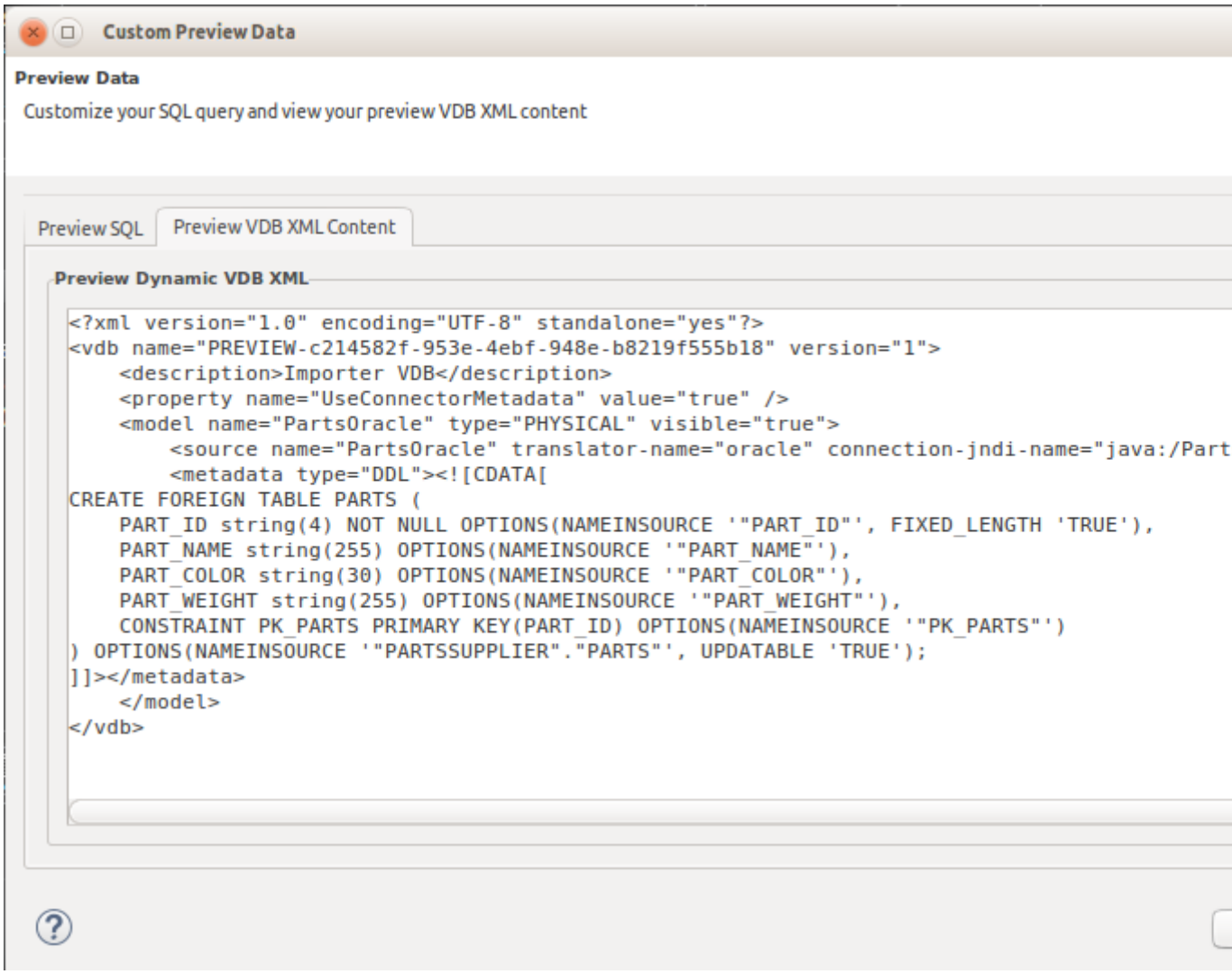


Figure 11.6. Preview Data: VDB XML Content Tabb

Testing Your Transformations

When editing transformation SQL in the Transformation Editor, a special **SQL Results** 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

All steps assume that all source models referenced by your models, either directly or through dependencies, are bound to connector bindings.

11.2.1. 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 D.2.1, "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 D.2.1, "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 D.2.1, "Model Explorer View"* or diagram.
  - **Step 3** - Your query results will be displayed in the *Section 11.2.5, "Sample SQL Results for Preview Data"* view. The view will automatically open or get focus if not visible in your perspective.

11.2.2. 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 D.2.1, "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 D.2.1, "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.
    I

Note

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.

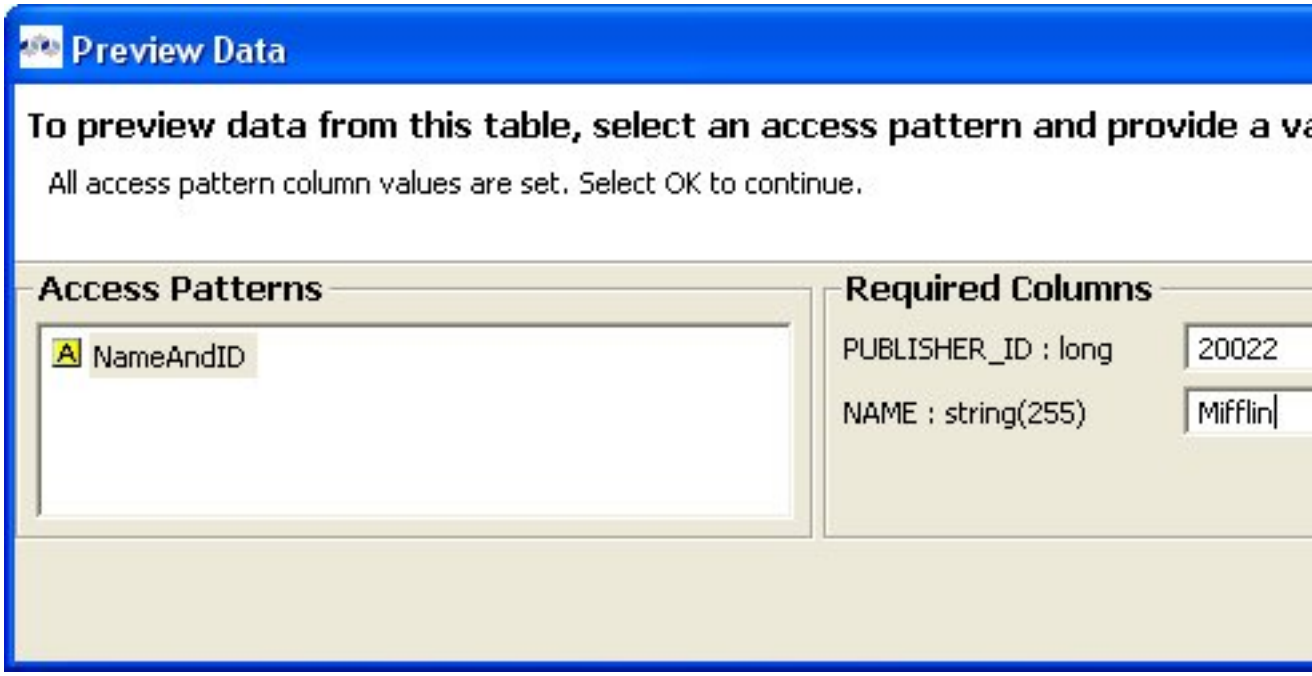


Figure 11.7. Access Pattern Column Input Dialog

- **Step 4** - Your query results will be displayed in the *Section 11.2.5, "Sample SQL Results for Preview Data"* view. The view will automatically open or get focus if not visible in your perspective.

11.2.3. Preview Relational Procedure

- To preview a relational procedure:
  - **Step 1** - Select a relational procedure in the *Section D.2.1, "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 D.2.1, "Model Explorer View"* or diagram.
  - **Step 3** - An input parameter input dialog is presented. Enter a valid value for each parameter.

Note

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.

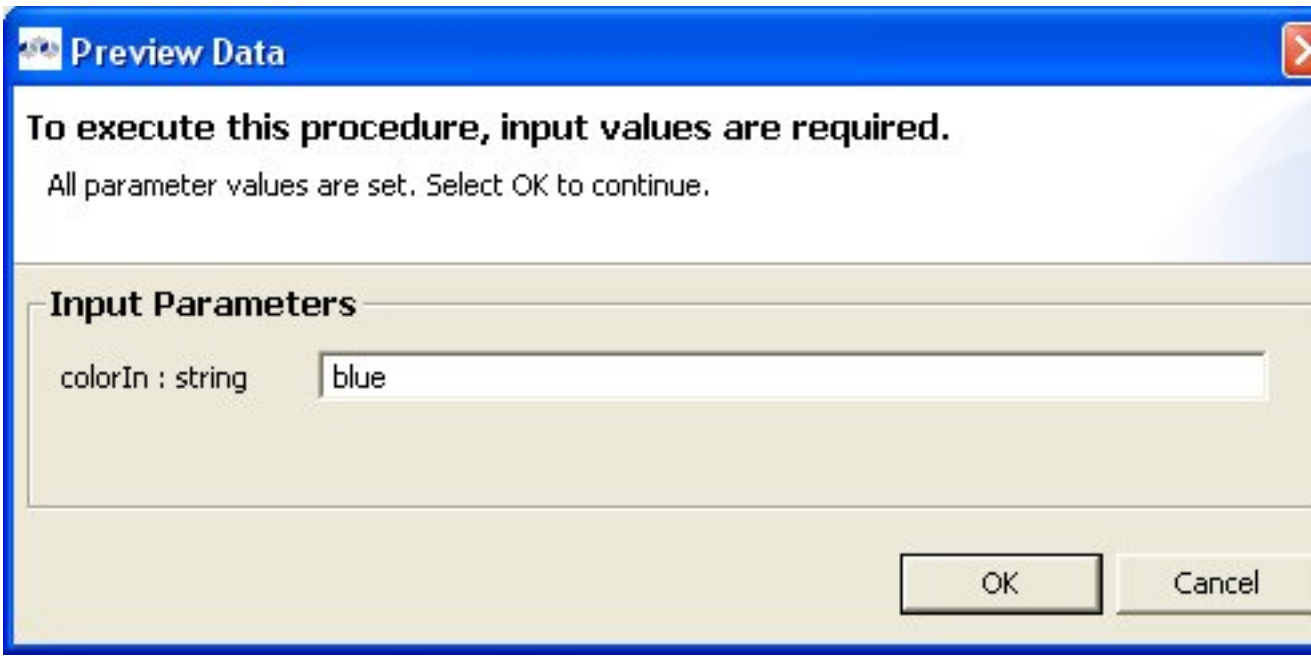


Figure 11.8. Procedure Parameter Input Dialog

- **Step 4** - Your query results will be displayed in the *Section 11.2.5, "Sample SQL Results for Preview Data"* view. The view will automatically open or get focus if not visible in your perspective.

11.2.4. Preview Web Service Operation

- To preview a Web service operation:
  - **Step 1** : Select a Web service operation in the *Section D.2.1, "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 D.2.1, "Model Explorer View"* or diagram.
  - **Step 3** : An input parameter input dialog is presented. Enter a valid value for each parameter.

Note

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.

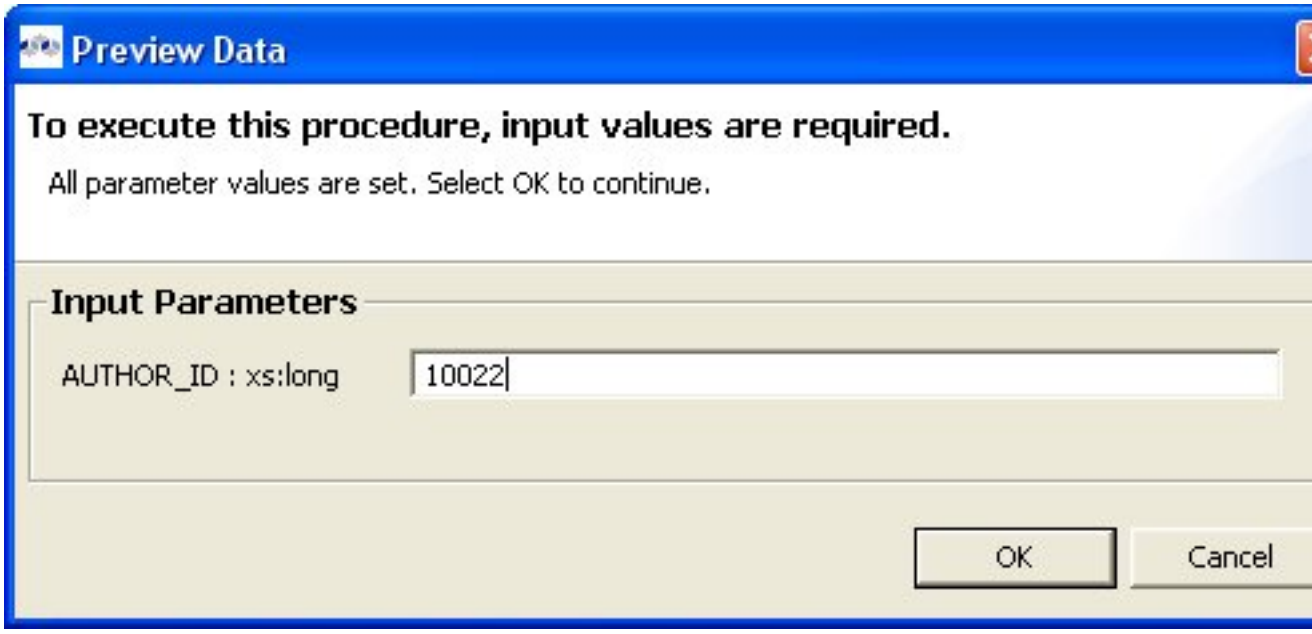


Figure 11.9. Procedure Parameter Input Dialog

- **Step 4** - Your query results will be displayed in the *Section 11.2.5, "Sample SQL Results for Preview Data"* view. The view will automatically open or get focus if not visible in your perspective.

11.2.5. 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.

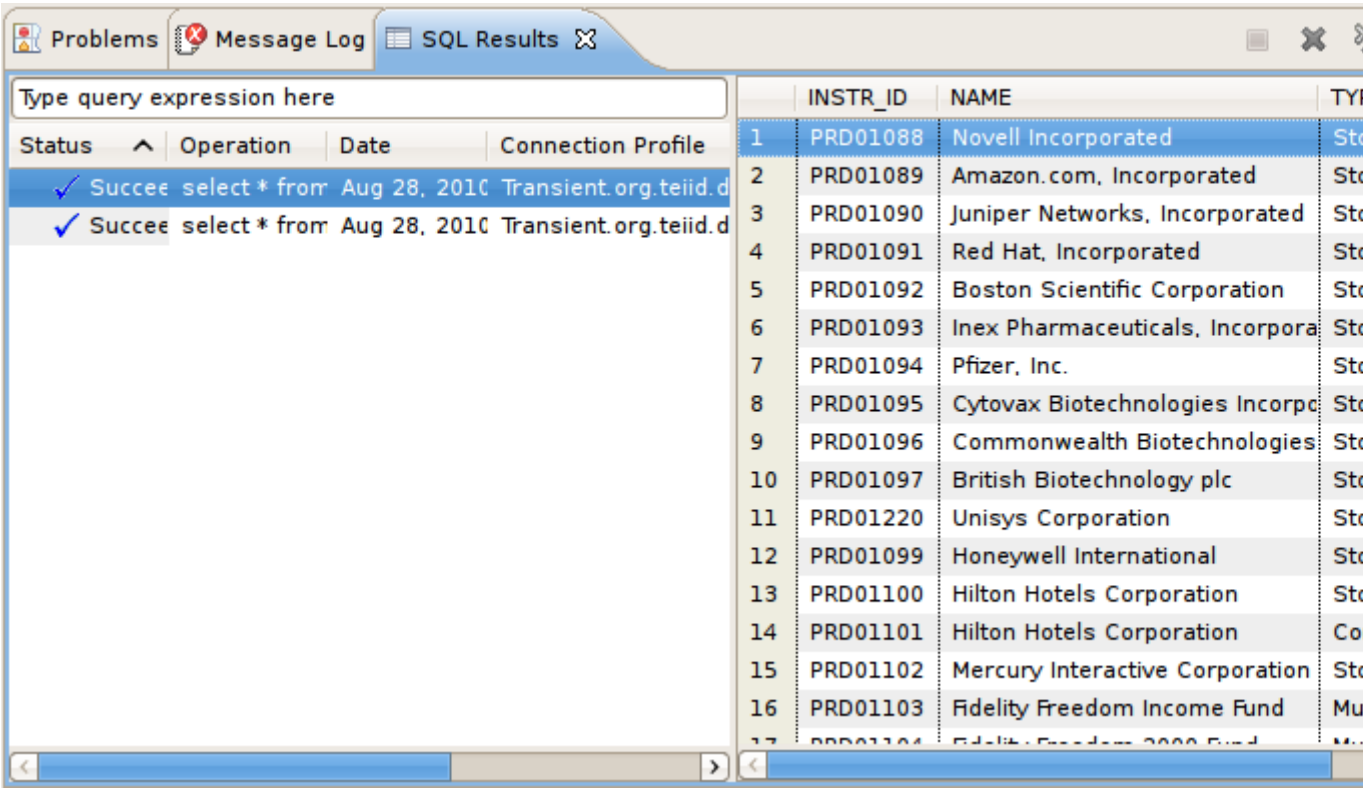


Figure 11.10. SQL Results View

11.2.6. Execution Plans

When Preview Data is executed, the **Teiid Execution Plan** is also displayed as shown below. The Execution Plan may also be obtained by rt-clicking on a previewable object, then selecting **Modeling > Show Execution Plan** in the context menu.

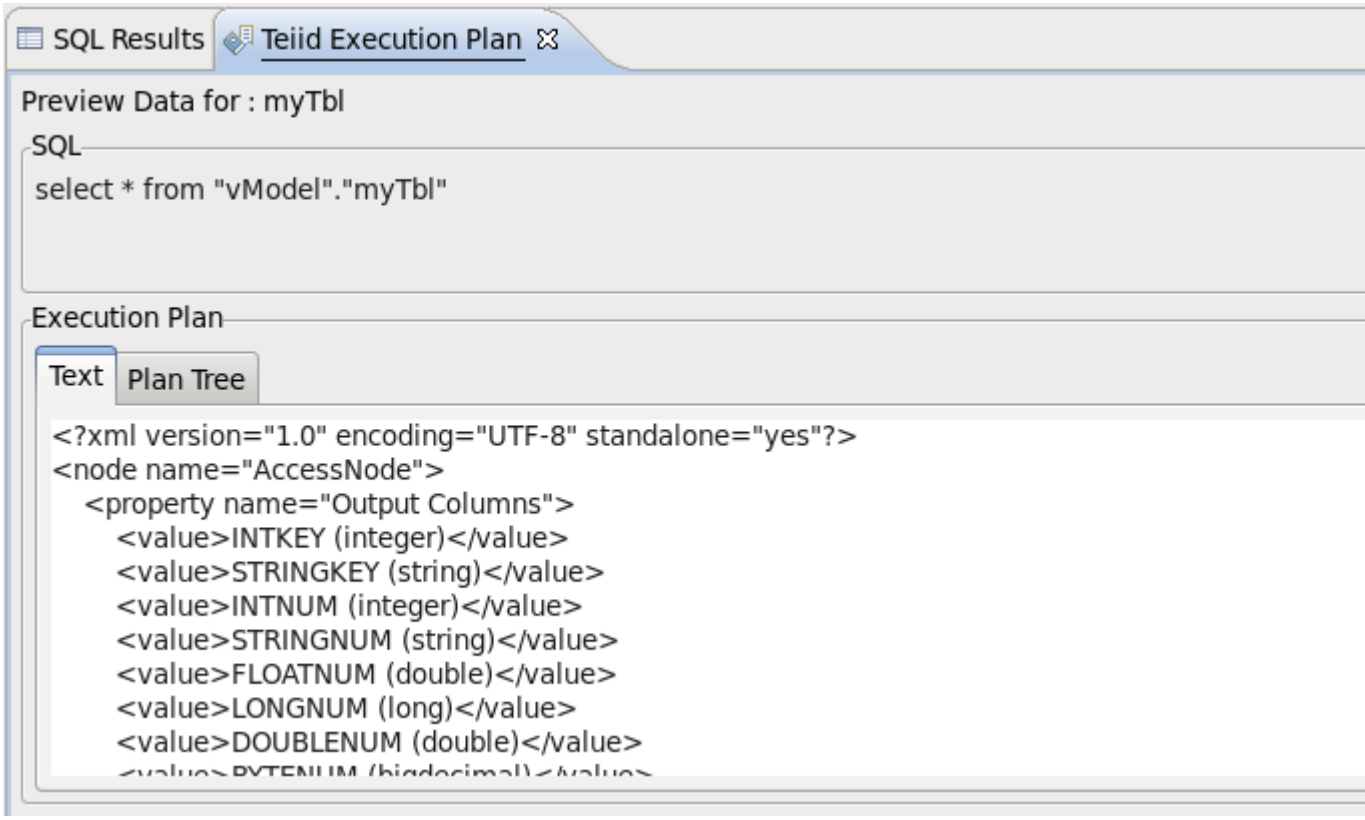


Figure 11.11. Teiid Execution Plan View

11.3. 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.

11.3.1. 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 requiries 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 Servers view. Select a Teiid server instance in the Servers view and right-click select the **Create Data Source** action. This will launch the Create Data Source Dialog shown below.

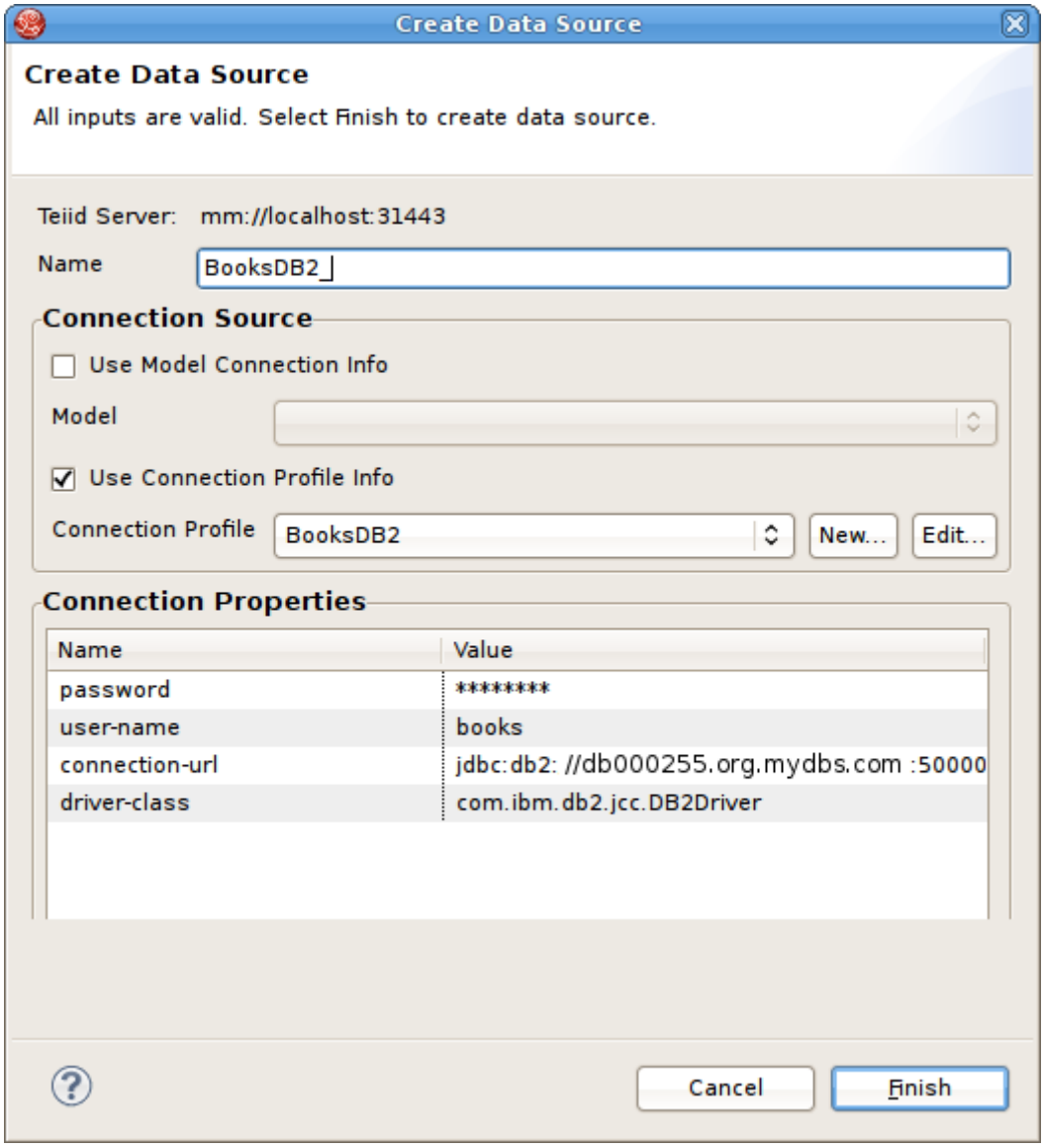


Figure 11.12. Create Data Source Dialog

You can either select and 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.

11.3.2. Execute VDB from Model Explorer

- If you have a Teiid instance defined and connected in your Servers 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 11.13. 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**

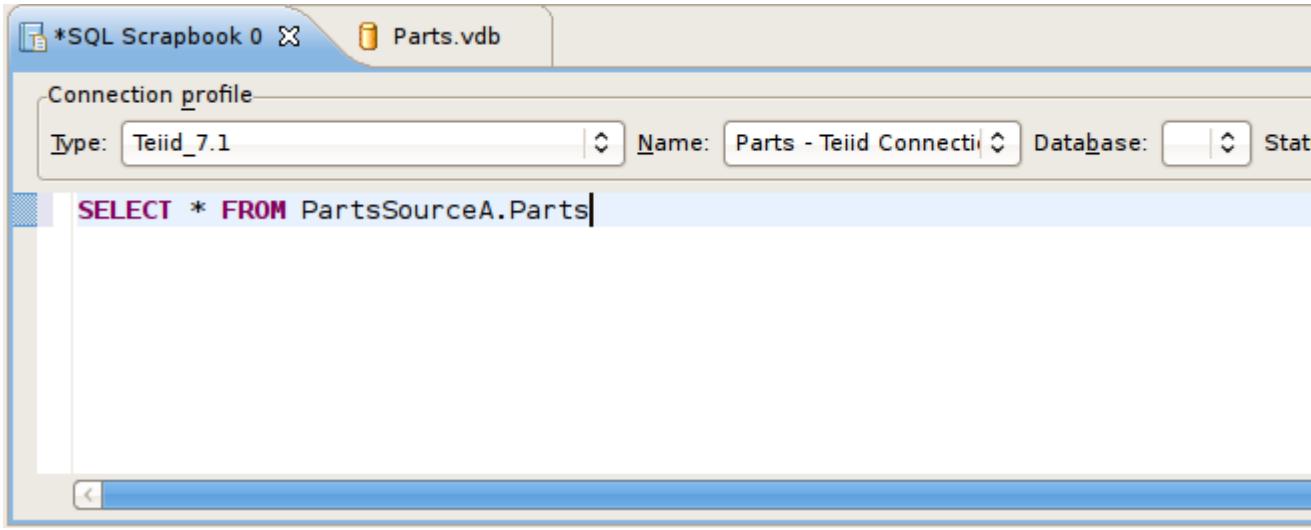


Figure 11.14. SQL Scrapbook Editor

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

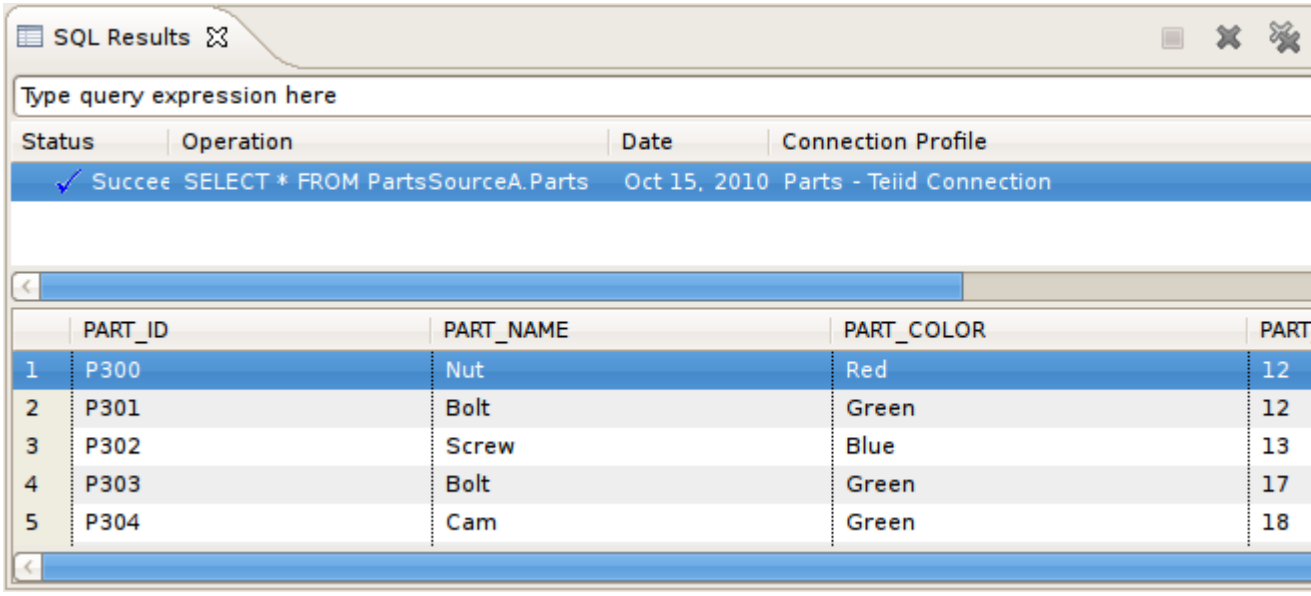


Figure 11.15. SQL Results View

11.3.3. 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 Servers 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

You can deploy a dynamic VDB to the server, but the VDB may end up being INACTIVE if any referenced data source (JNDI name) is not already be deployed. Dynamic VDBs do not contain enough connection or data source information to automatically find and/or deploy a data source.

Note

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.

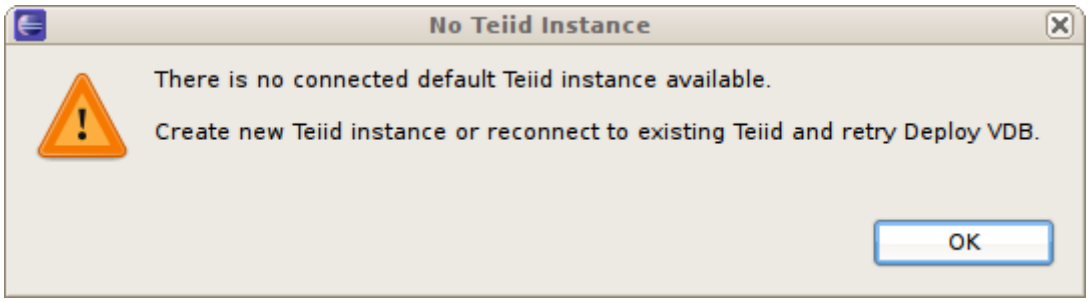


Figure 11.16. No Teiid Instance Defined

11.3.4. 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.

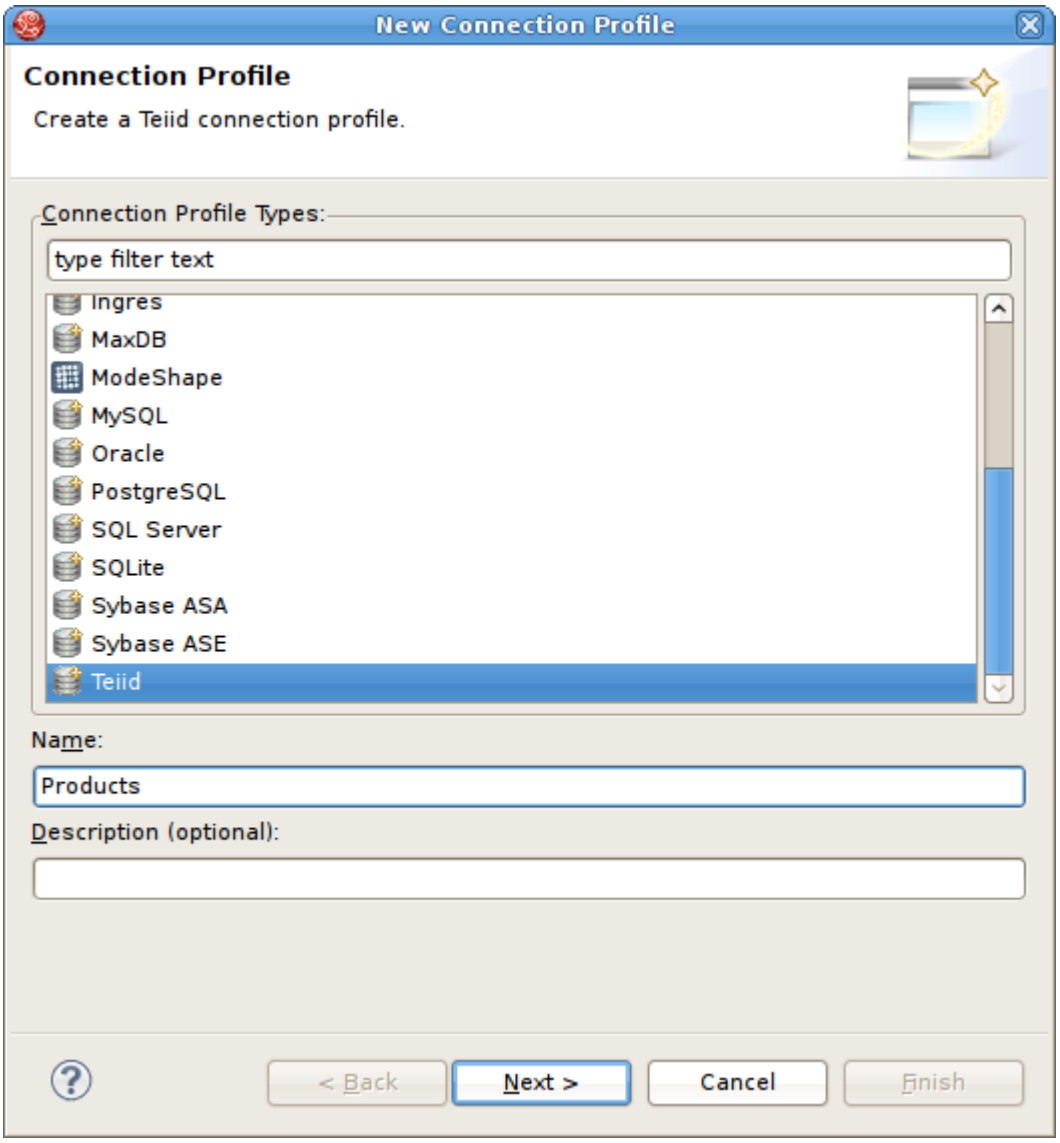


Figure 11.17. New Connection Profile Dialog

- **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").

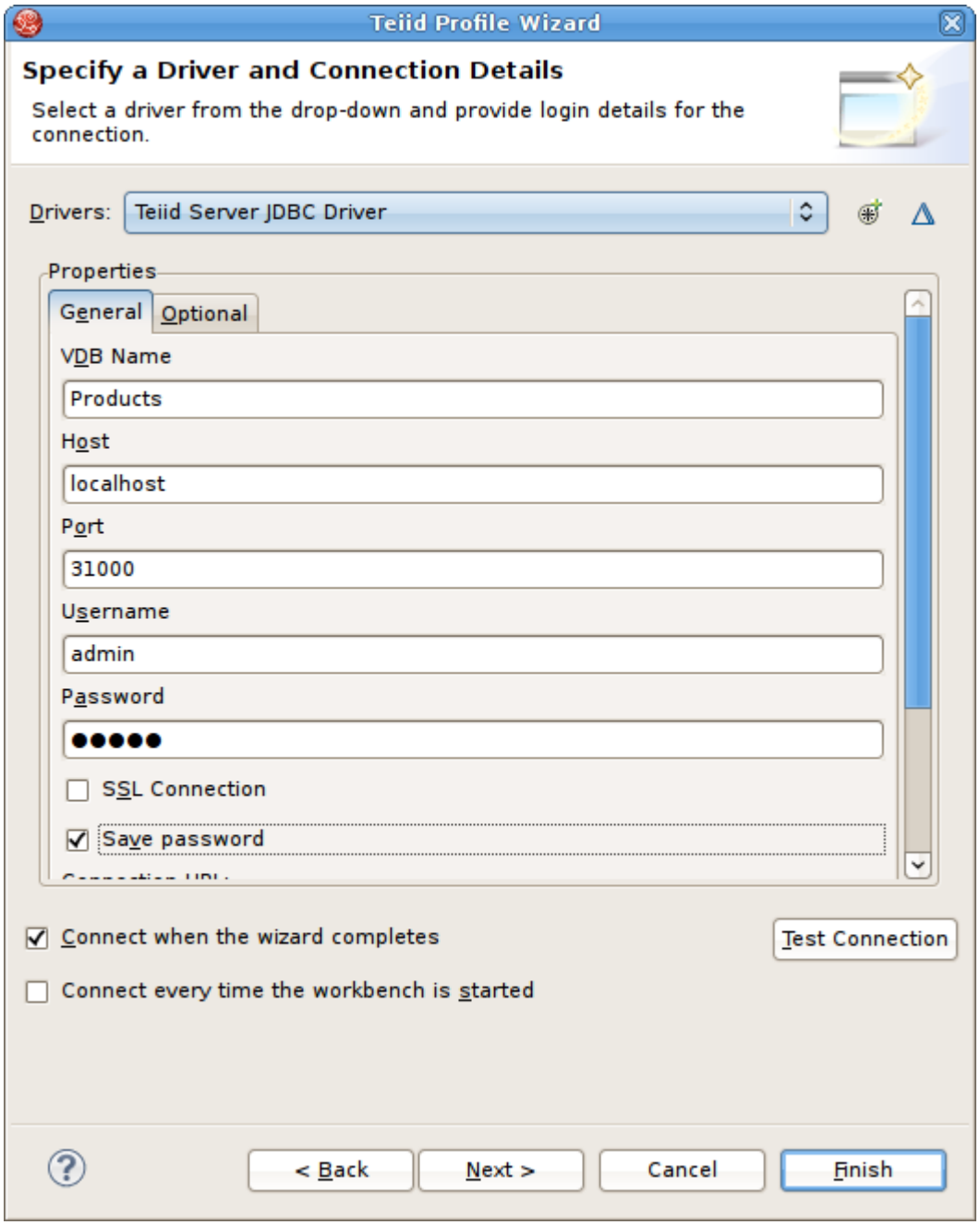


Figure 11.18. Teiid Connection Profile Dialog

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

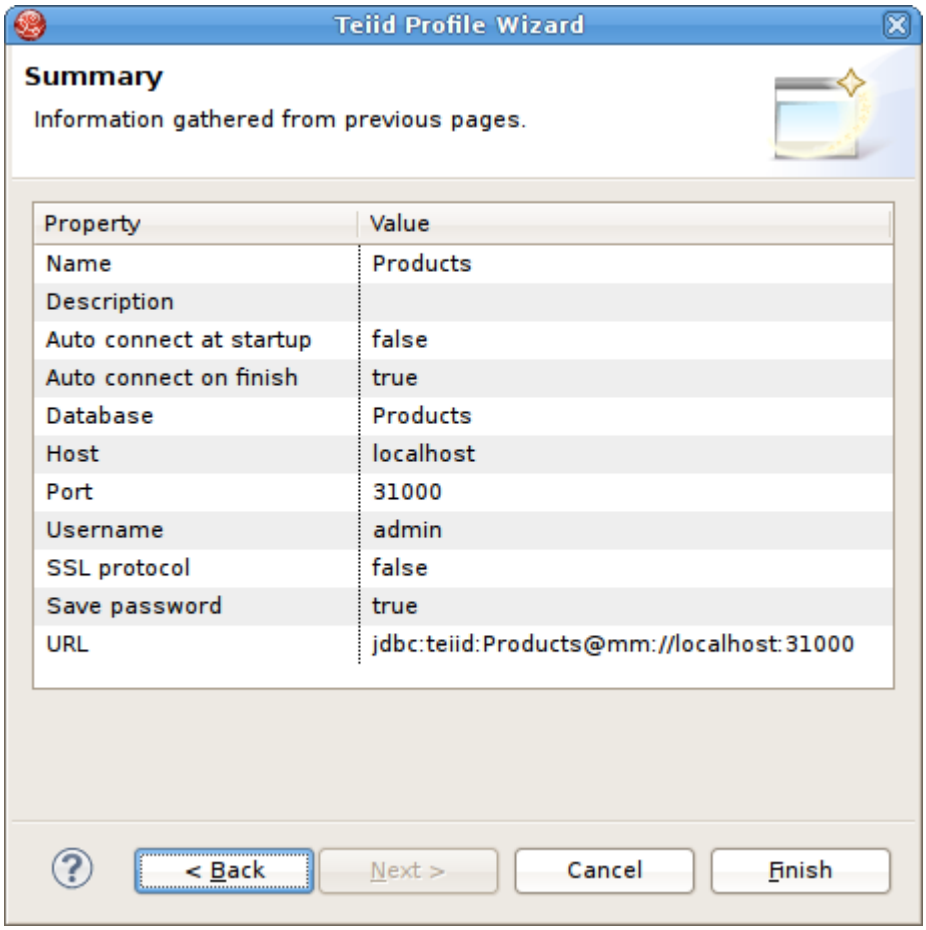


Figure 11.19. Teiid Connection Profile Summary

- **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**.

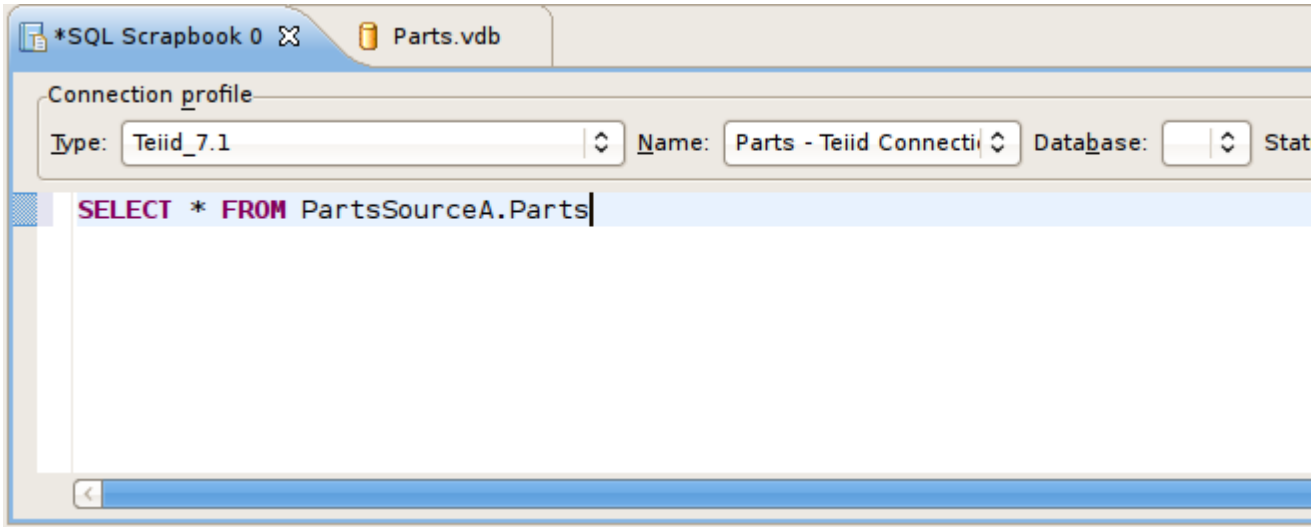


Figure 11.20. SQL Scrapbook Editor

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

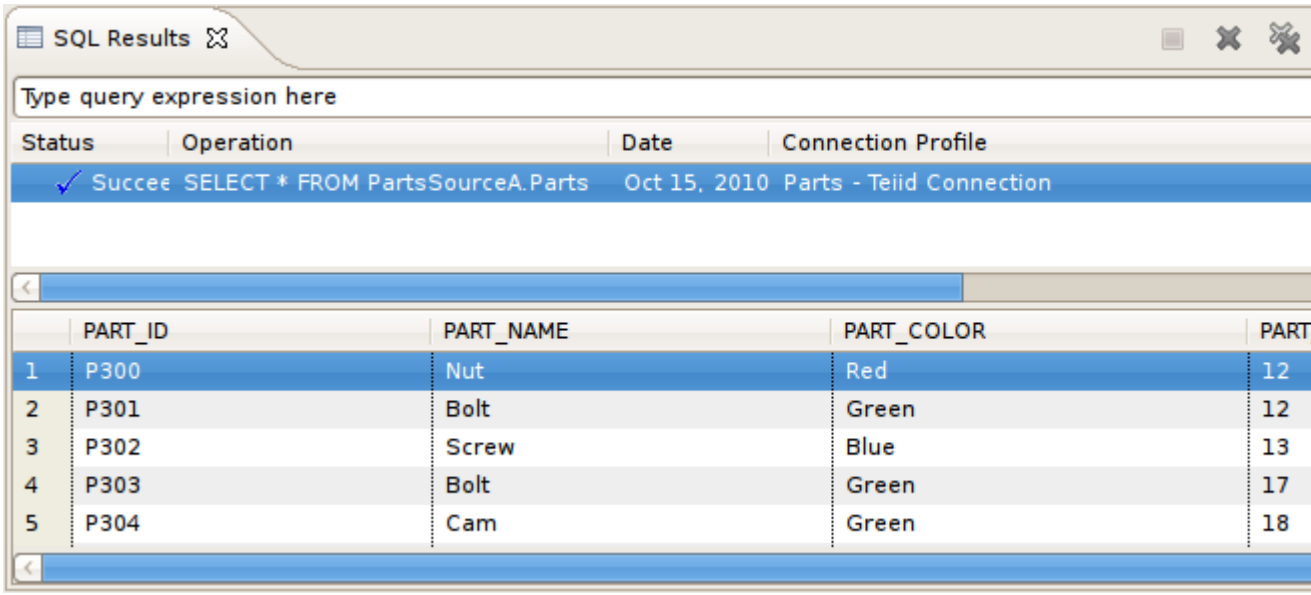


Figure 11.21. SQL Results View

- **Step 9** : The query Execution Plan should also be displayed on the **Teiid Execution Plan** view tab. The Execution Plan can also be generated without running the query. In the SQL scrapbook, rt-click then select **Teiid_7.x > Get Execution Plan**.

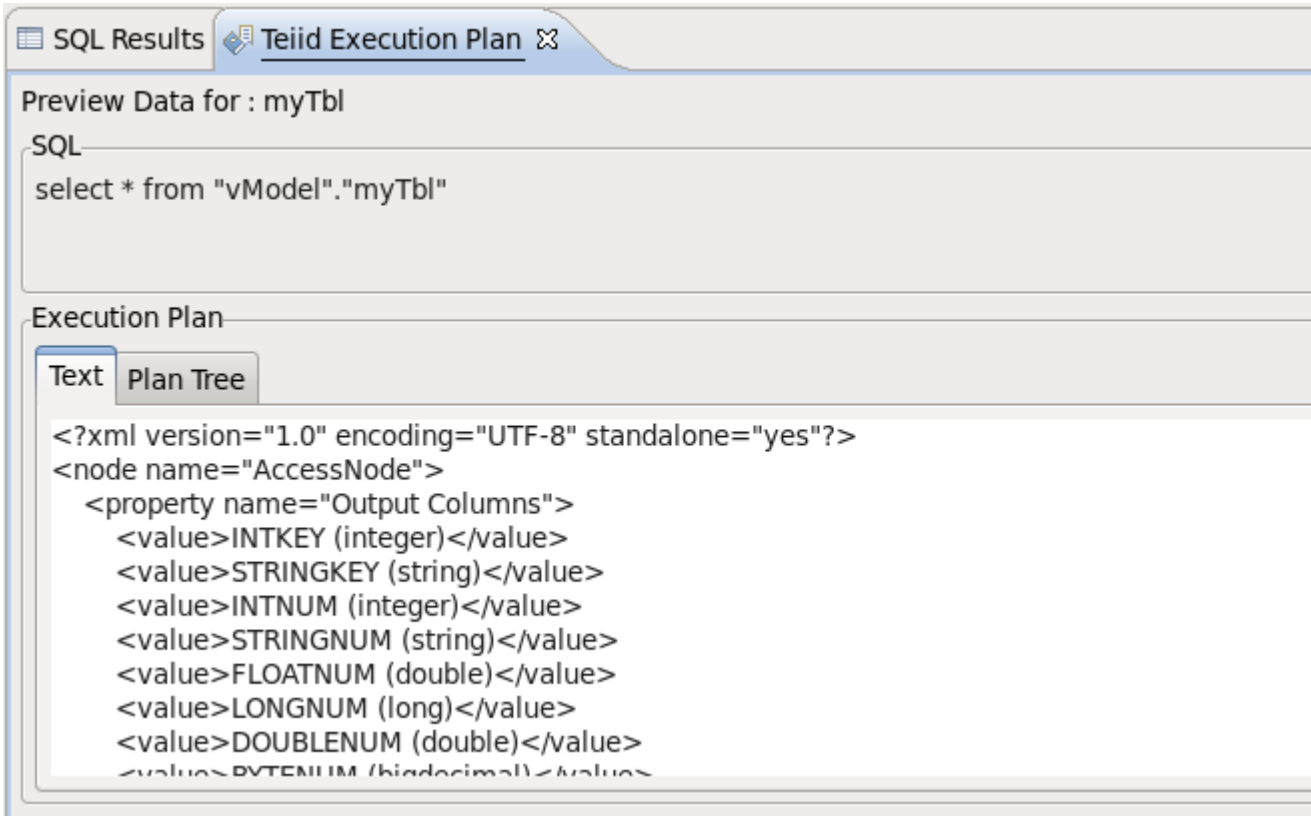


Figure 11.22. Teiid Execution Plan View

Materialization Support

Designer provides modeling support for server-side materialization including:

- Extension properties for materialized view and referenced materialized tables
- General Create POJO feature for selected table or view
- Materialize action and wizard to set required properties and generate optional JDG module corresponding to the generated materialized table
- JDG connection profile definition and wizard

12.1. Materialization Extension Properties

The Teiid runtime supports materialization which is identified at VDB deployment time through metadata properties. A couple base properties are provided by Designer's relational table properties. These include a **Materialized** boolean property and a **Materialized Table** reference property as shown below. In addition, there are extension properties for a materialized table to help set-up your materialized table behavior. (See Teiid Runtime documentation for details)

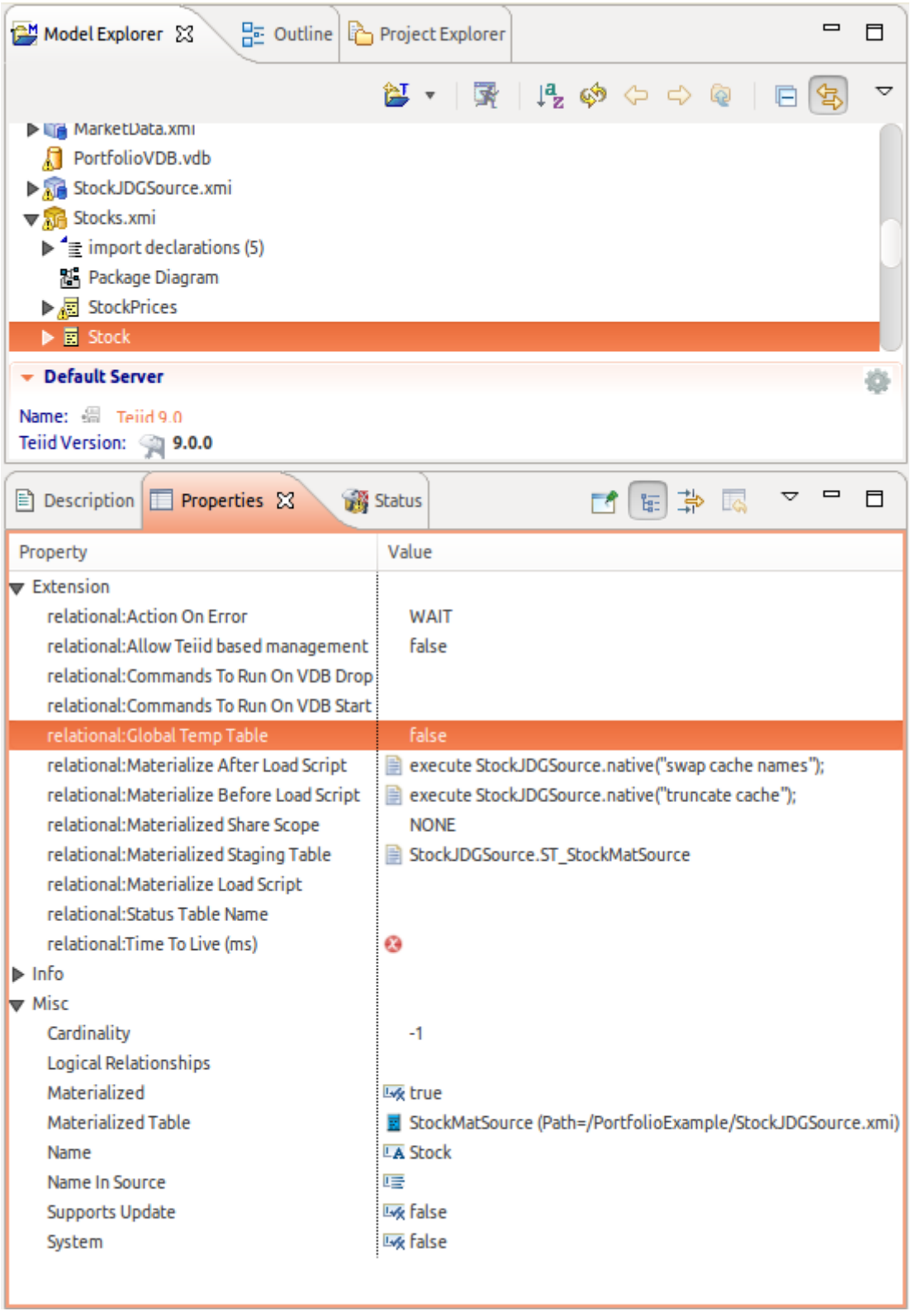


Figure 12.1. Materialized View Properties

12.2. Create POJO

As part of the new 11.0 Designer feature for supporting JDG and materialization, we added a separate **Create POJO** action and wizard. This feature will generate a simple java class that results in getter and setter methods for each column. There is also an option to generate an JDG module containing the compiled class.

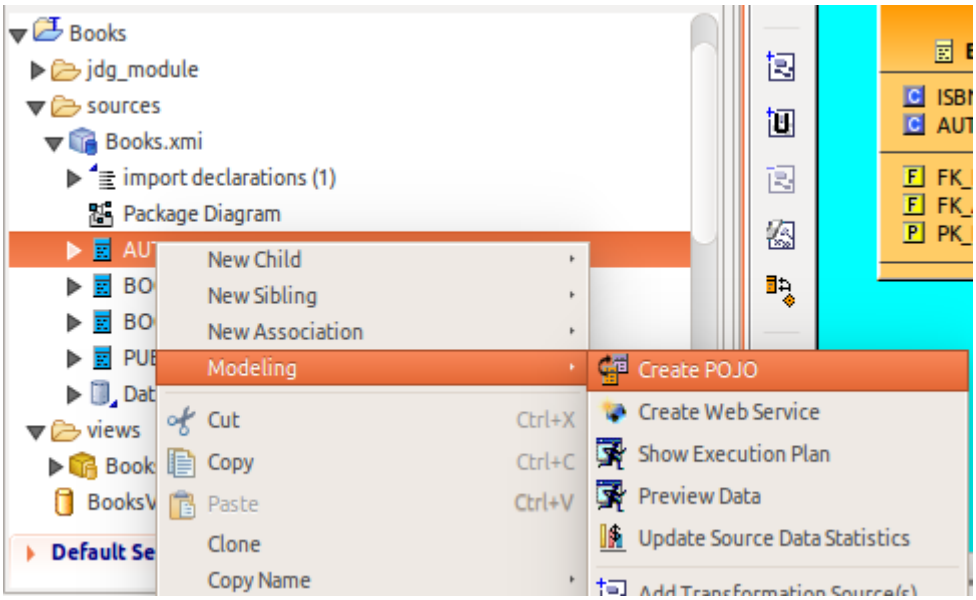


Figure 12.2. Create POJO Action

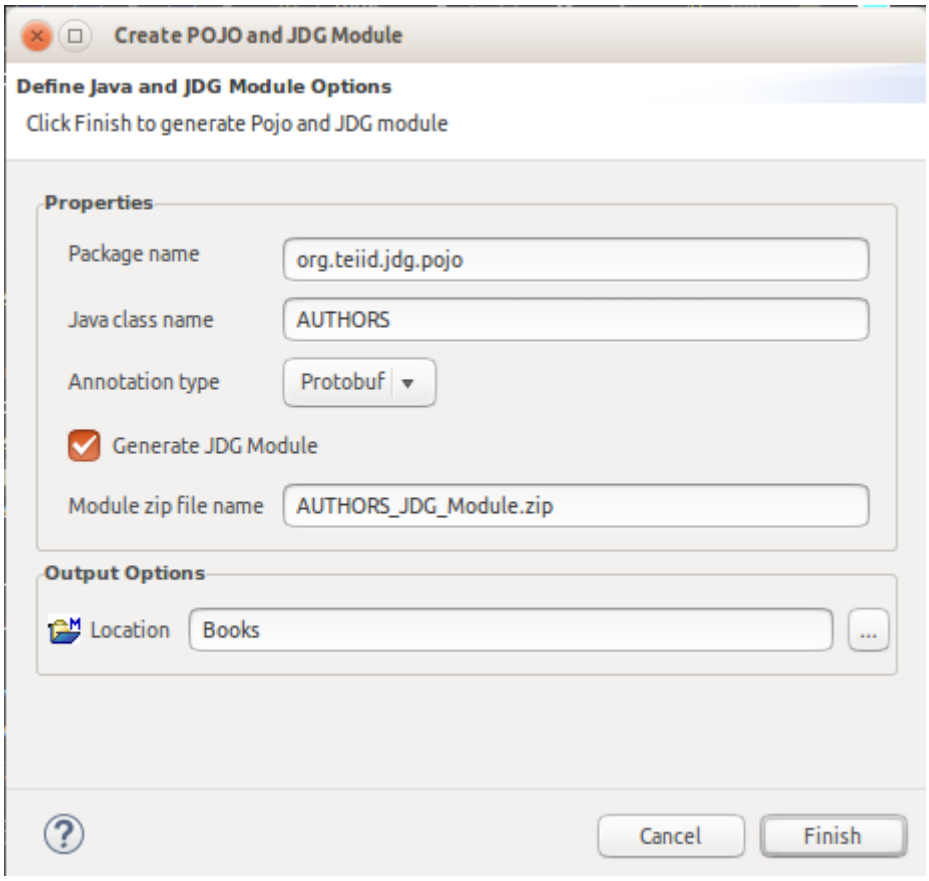


Figure 12.3. Create POJO Dialog

12.3. Materialize a View

You can now materialize a view and generate the necessary related artifacts to support external materialization through your JDG cache. The generated artifacts include:

- New source model containing both the materialized table and companion staging table.
- POJO java file and compiled java class
- JDG zip module containing packaged jar file and module.xml file for deployment/extraction into your JBoss server

Follow the steps below to materialize your view....

- **Step 1**- In the **Model Explorer**select the view you wish to materialize and right-click select the **Modeling > Materialize** action

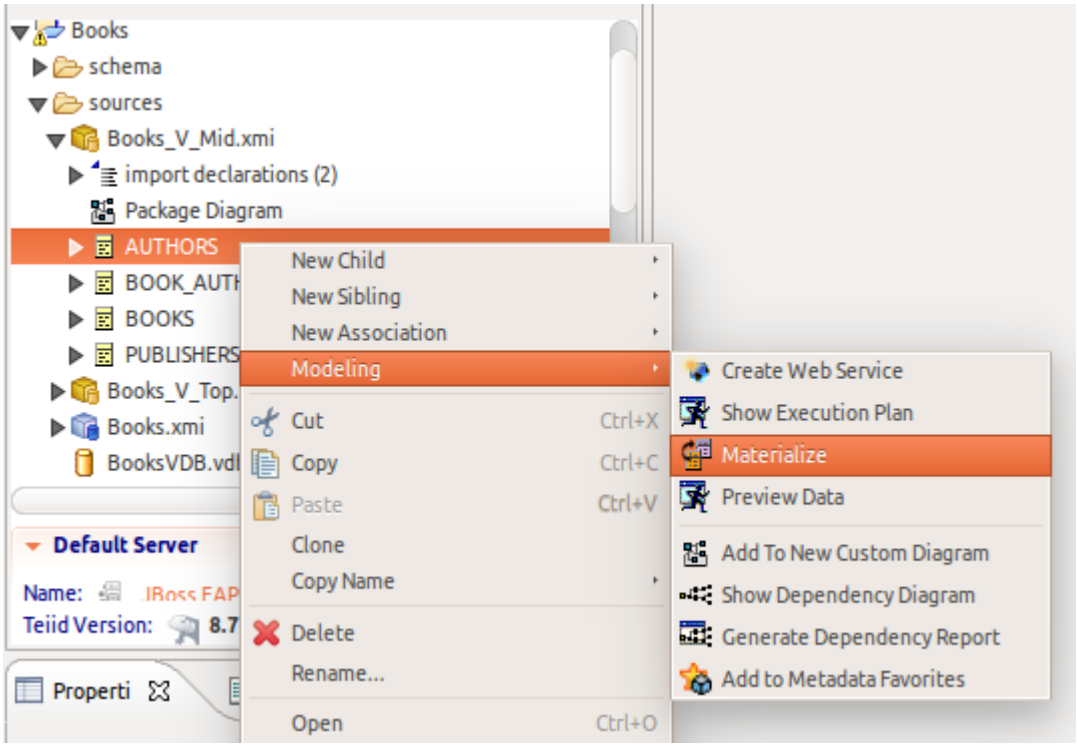


Figure 12.4. Materialize Action

- **Step 2**- On the first wizard page, define your generated source model name and location within the same model project. Click **NEXT>** to continue.

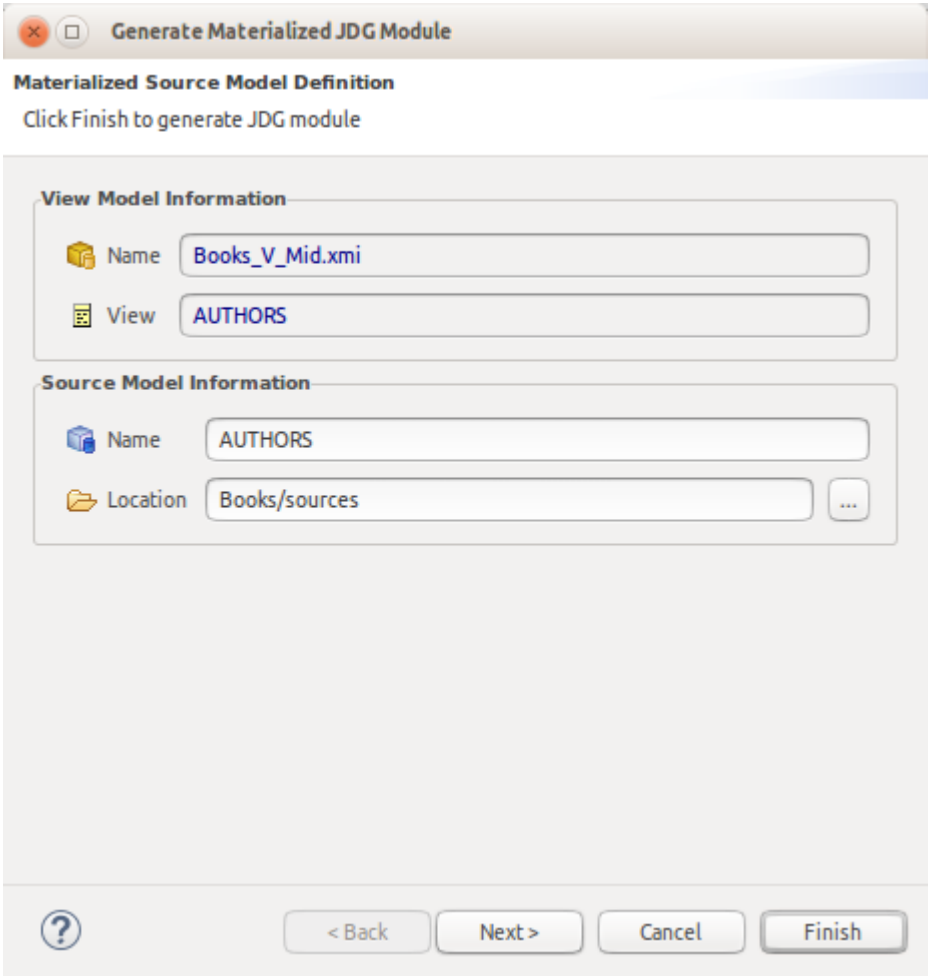


Figure 12.5. Materialized Source Model Definition

- **Step 3**- On the second wizard page, select your options for generating a POJO and a JDG module as well as the output folder location in your model project. Click **Finish** perform to complete the operations.

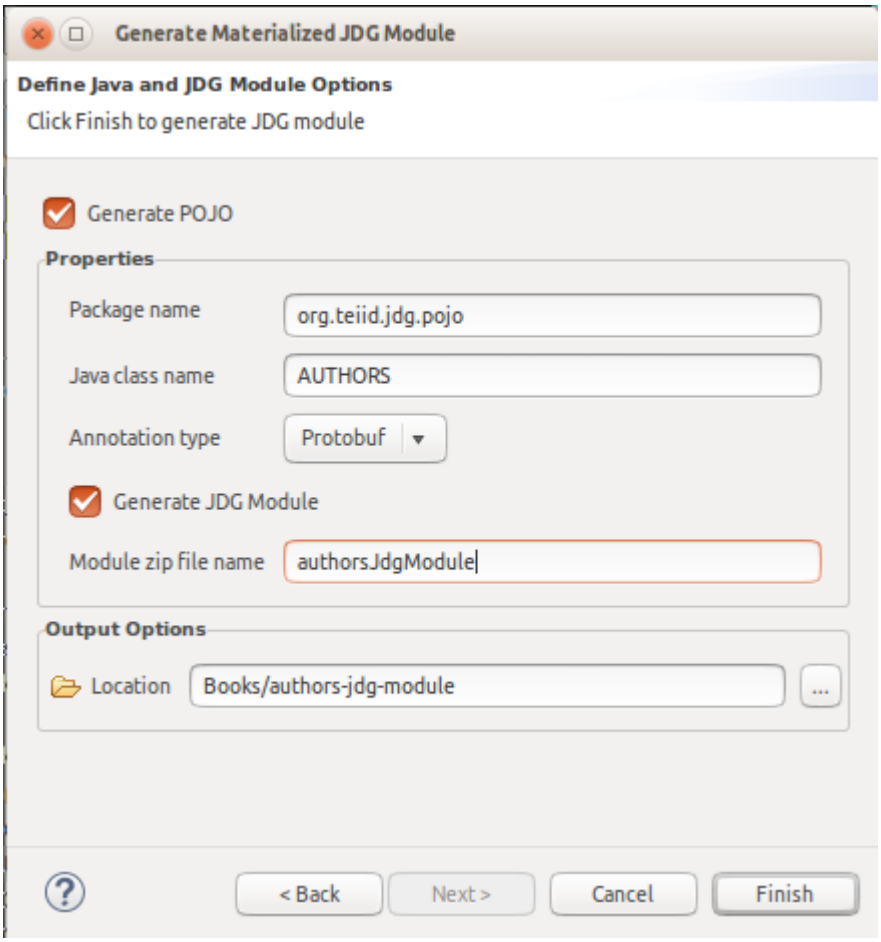


Figure 12.6. Define Java and JDG Module Options

The figure below shows the file structure of the generated POJO, class file, jar and jdg module zip file

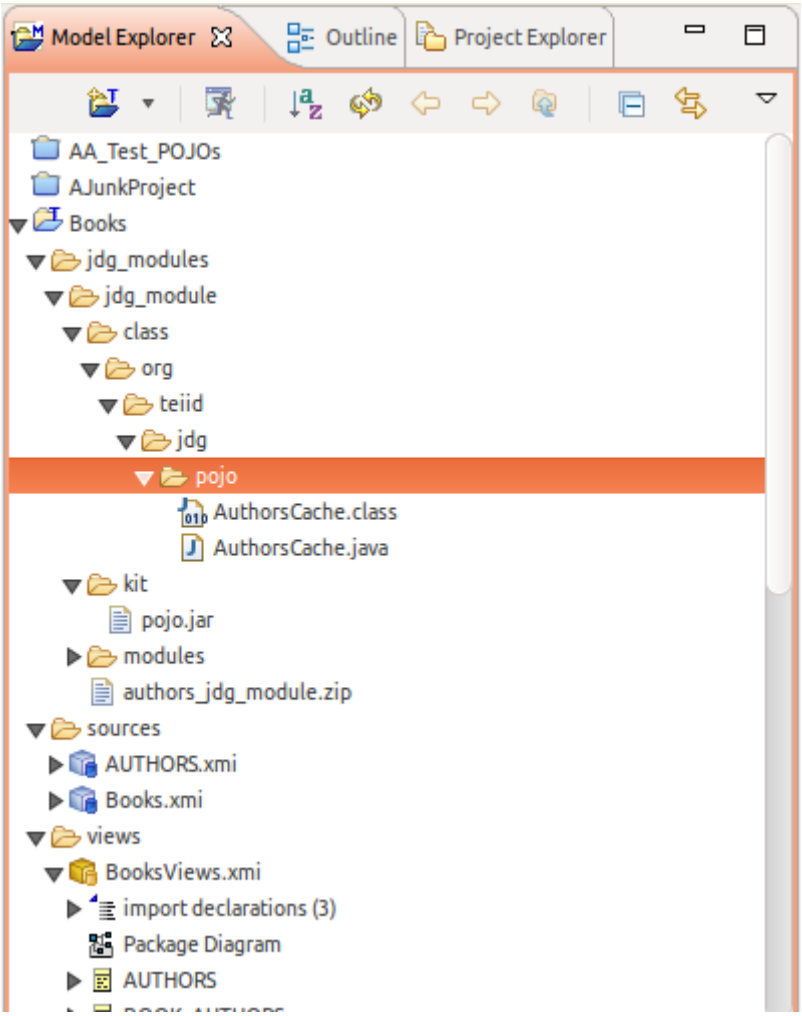


Figure 12.7. Generated JDG module files

12.4. JDG Connection Profile

JDG connection profile wizard can be used to define JDG-specific connection information required to create a server-side resource adapter. By setting the JDG connection profile on your generated materialized source model you can use the **Create Data Source** feature to create and deploy a resource adapter that supports your materialized view.

To create a JDG connection profile, open the **Database Development** perspective and select the **New Connection Profile** action and **JDG** profile option as shown below.

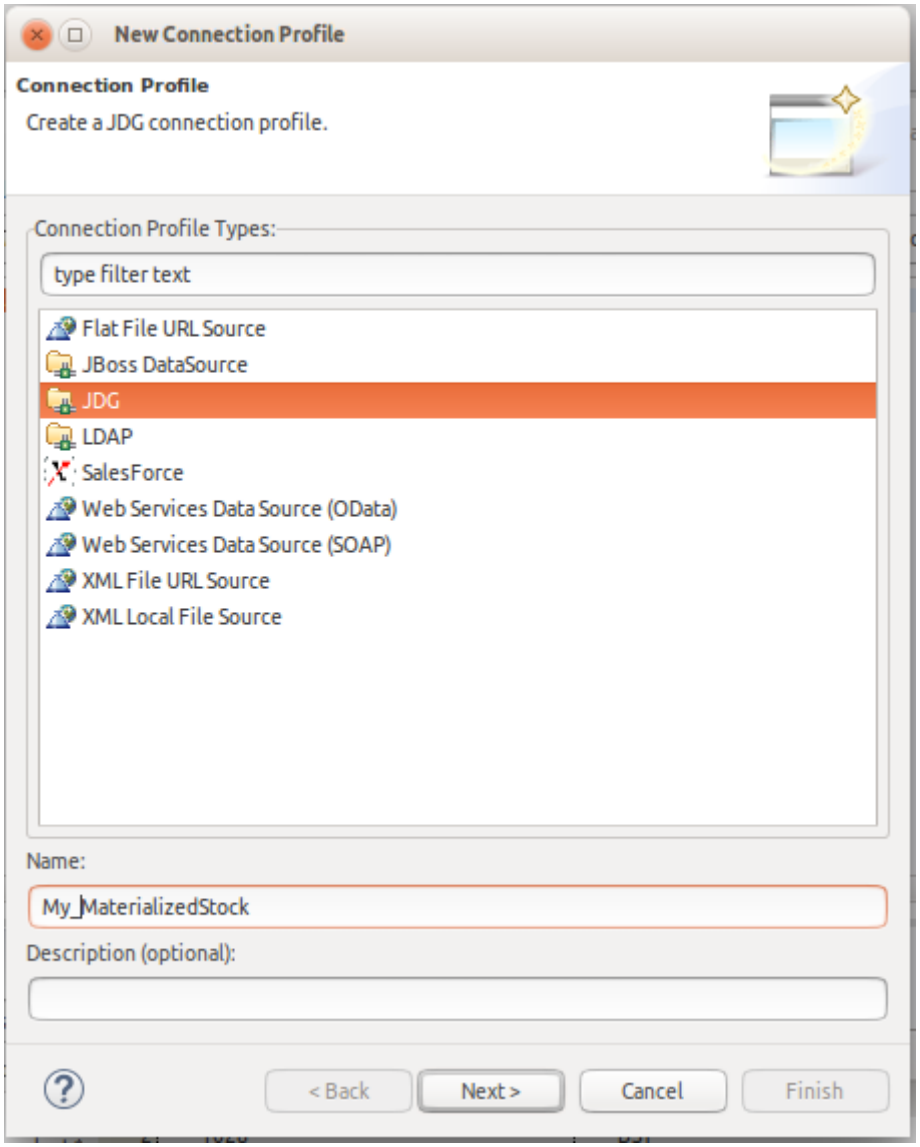


Figure 12.8. JDG Connection Profile Page 1

Enter your unique JBoss JNDI name and various JDG-specific resource adapter properties and click **Finish**

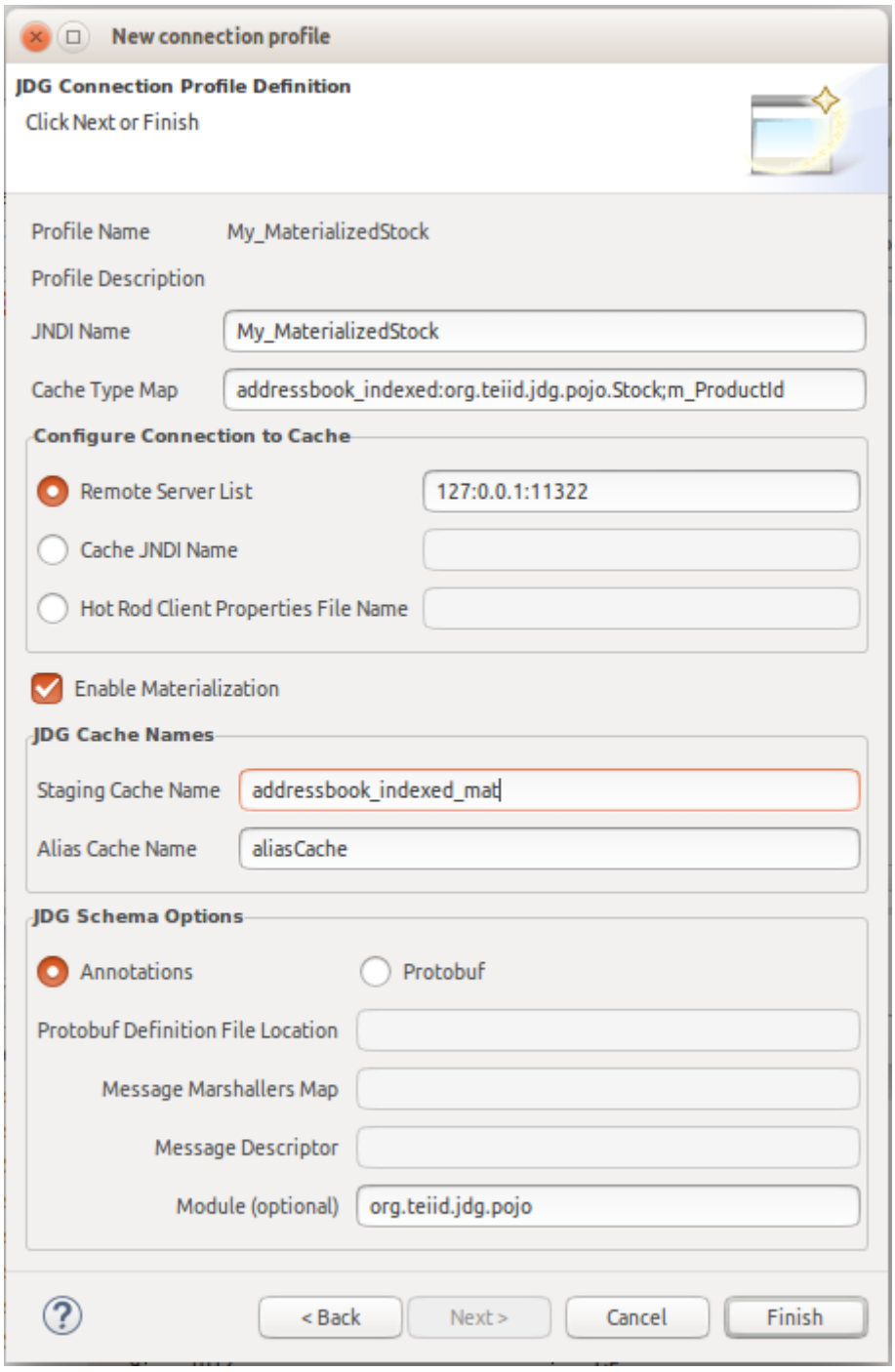


Figure 12.9. JDG Connection Profile Page 2

Searching

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

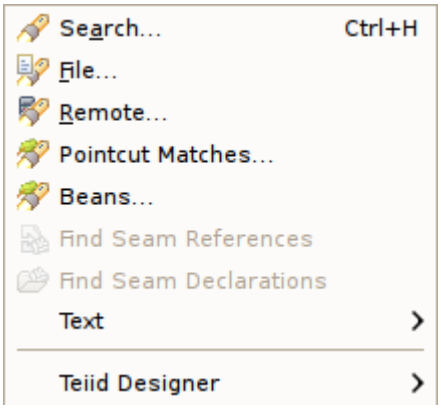


Figure 13.1. Search Options

- The individual actions in the Teiid Designer sub-menu are described below:
  - **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 D.2.8, "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.

13.1. 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.

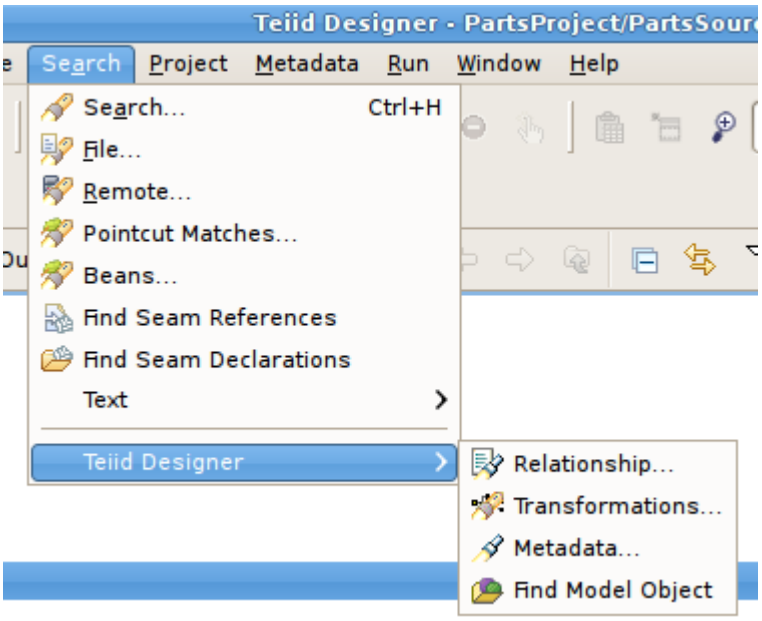


Figure 13.2. Find Model Object Action In Toolbar

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

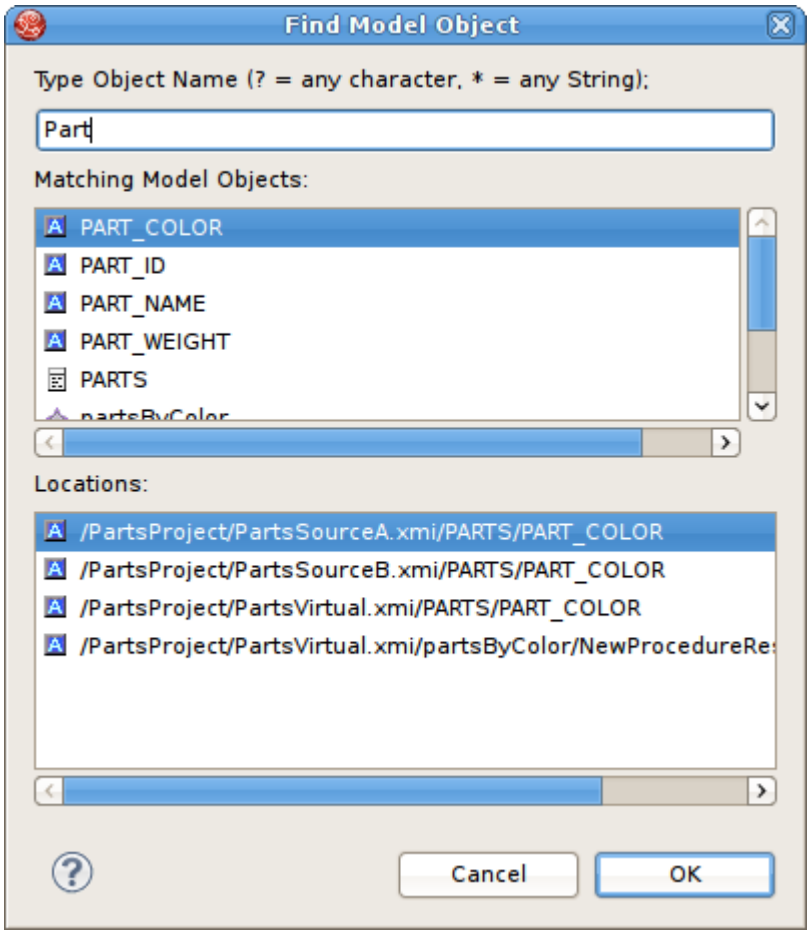


Figure 13.3. Find Model Object Dialog

- **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.

13.2. 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 Teiid Designer main menu

  which opens the **Search Transformations** dialog.

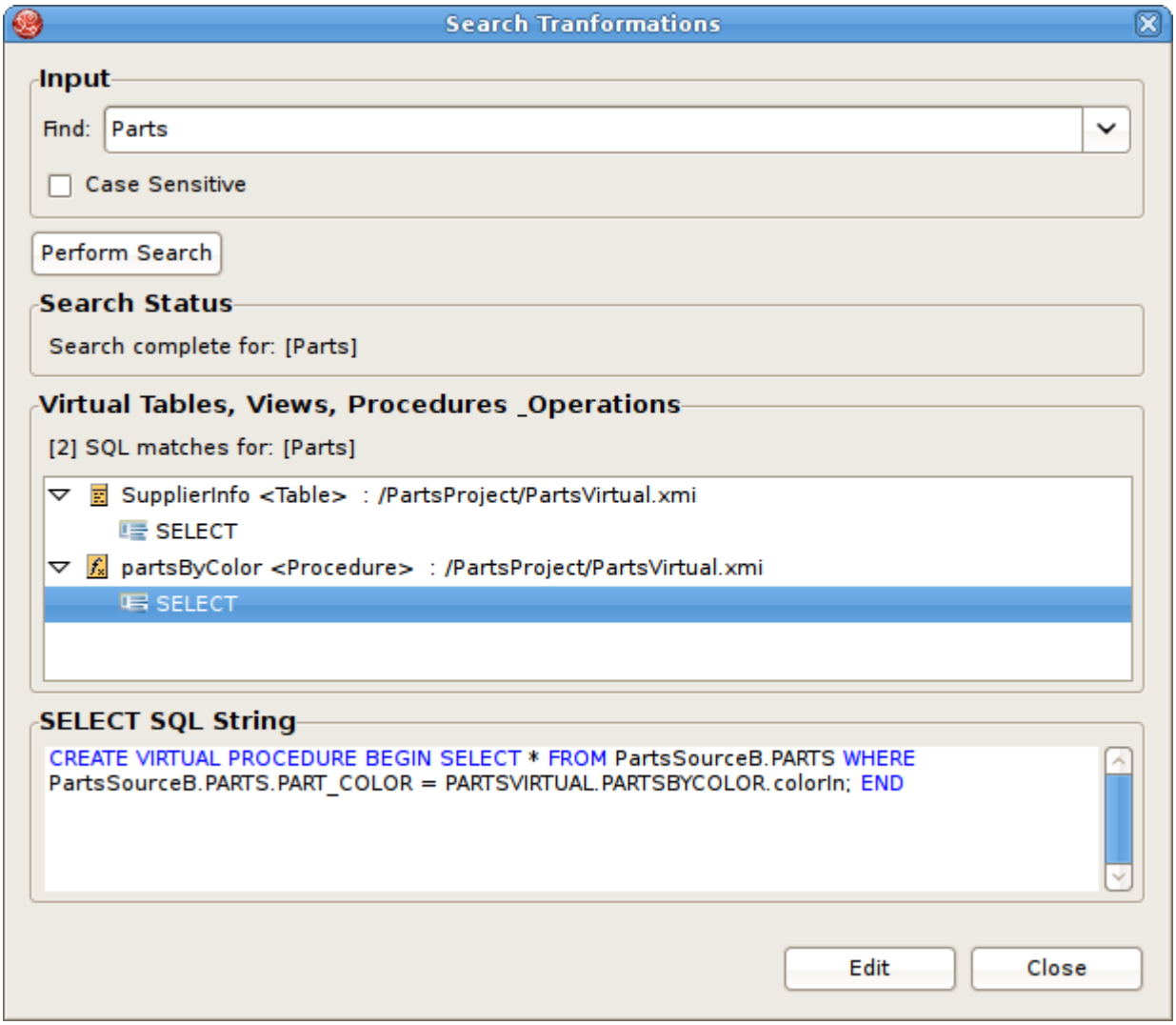


Figure 13.4. Search Transformations Dialog

- **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 occurances 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.

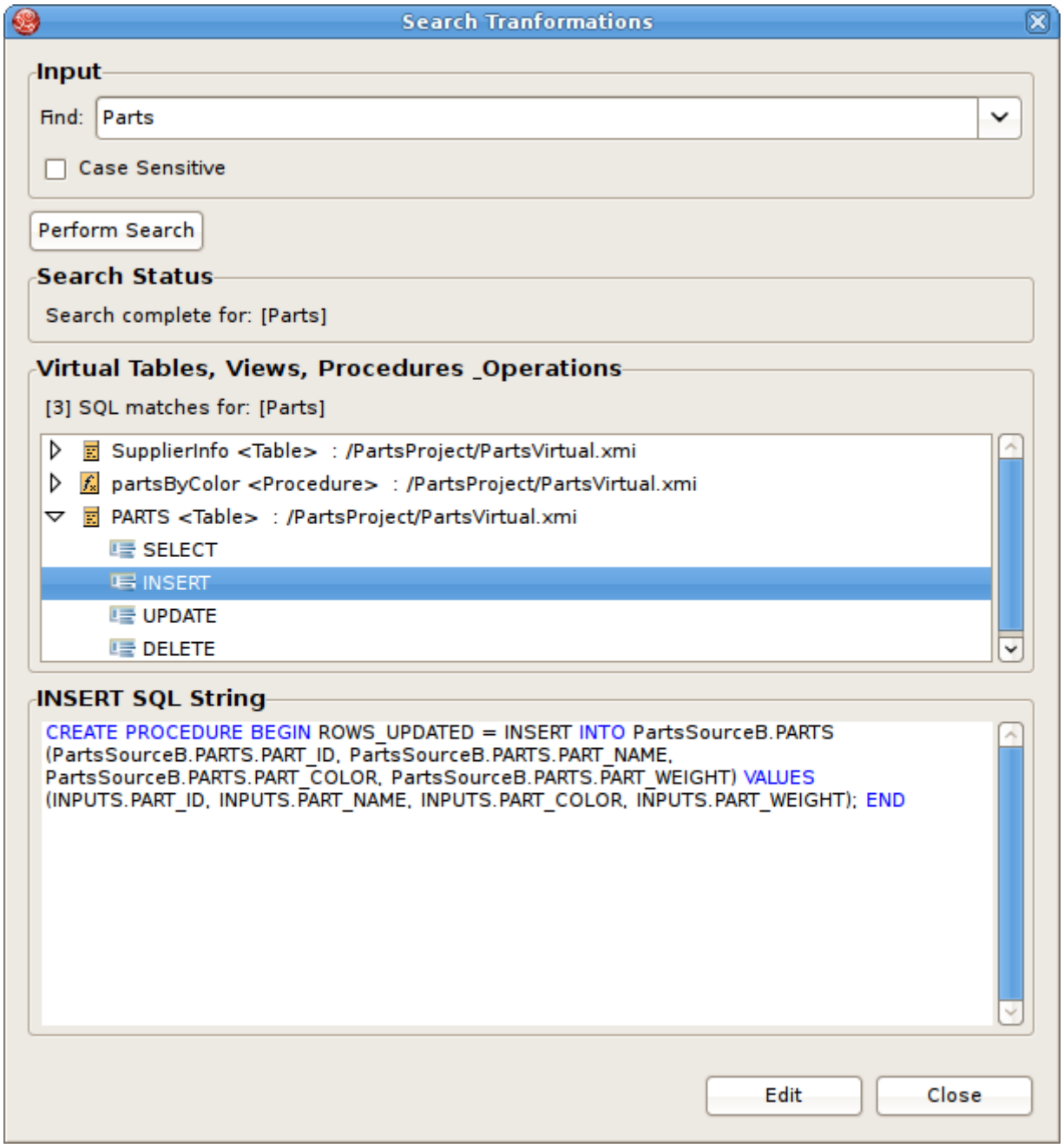


Figure 13.5. Insert SQL Example

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.

13.3. 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 Teiid Designer toolbar
    which opens the **Search** dialog.

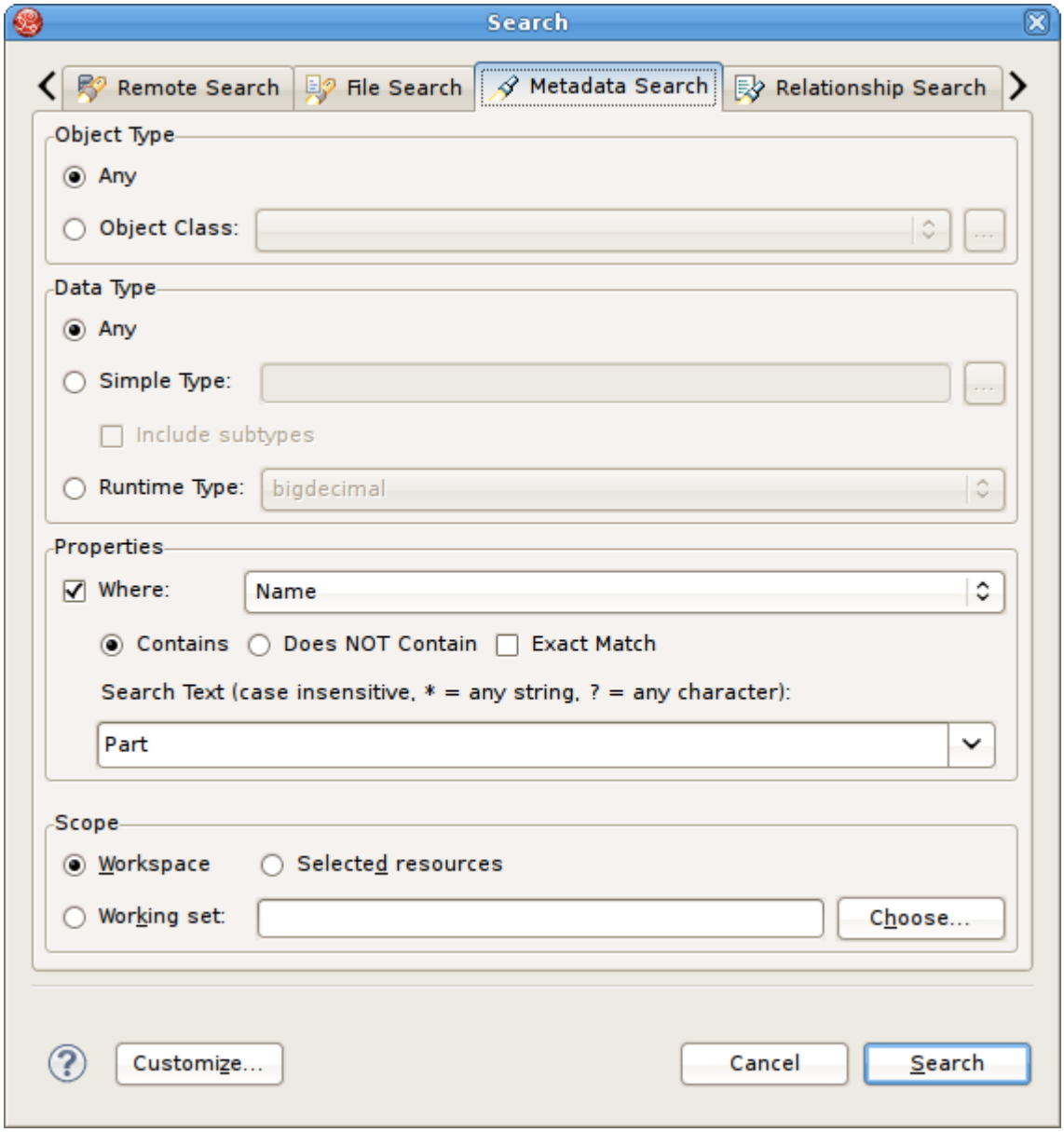


Figure 13.6. Metadata Search Dialog

  - **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 D.2.8, "Search Results View"*. If the view is not yet open, it will be opened automatically.

Appendix A. Supported Data Sources

The matrix indicates for a given data source how a model can be created (Desginer Import Option) and how the data source is integrated (Translator) for data access.

Note

The DDL Import option is an available option to build a source model for any data source. Its only indicated below when there's no specific importer created for that specific data source type.

Table A.1. Teiid Designer Supported Data Sources

Data Source	Translator Type	Designer Import Option
Apache Derby	JDBC - derby	JDBC Importer
Files	file	File Importer
General JDBC	JDBC - jdbc-simple	JDBC Importer
HDFS (Hadoop)	hive	use DDL importer or perform modeling manually
HSQL	JDBC - hsql	JDBC Importer
H2	JDBC - h2	JDBC Importer
Ingres	JDBC - ingres (Ingres 2006 or later) JDBC - ingres93 (Ingres 9.3 or later)	JDBC Importer
IBM DB2	JDBC - db2	JDBC Importer
Informix	JDBC - informix	JDBC Importer
LDAP/ActiveDirectory	ldap	LDAP Importer
LoopBack	JDBC - loopback	use DDL importer or perform modeling manually
MetaMatrix	JDBC - metamatrix	JDBC Importer
ModeShape/JCR	JDBC - modeshape	JDBC Importer
Mondrian	olap	use DDL importer or perform modeling manually
MS Access	JDBC - access	JDBC Importer

Data Source	Translator Type	Designer Import Option
MS Excel	JDBC - excel-odbc	JDBC Importer
MS SQL Server	JDBC - sqlserver	JDBC Importer
MySQL	JDBC - mysql5 (mysql)	JDBC Importer
Netezza	JDBC – netezza	JDBC Importer
Oracle	JDBC - oracle	JDBC Importer
PostgreSQL	JDBC - postgresql	JDBC Importer
Salesforce.com	salesforce	SalesForce Importer
SAP Gateway	ws	File Source (XML) Importer
SAP R/3	ws	
SAP Services Registry	ws	WSDL Importer
Sybase ASE	JDBC - sybase	JDBC Importer
Teradata	JDBC - teradata	JDBC Importer
Teiid	JDBC - teiid	JDBC Importer
Web Services (SOAP/WSDL)	ws	WSDL or URL Importer
Web Services (Rest/OData)	ws	File Source (XML) Importer

Appendix B. Designer Metadata Usage Requirements In Teiid Runtime

Based on the metadata exposed by the Teiid Desinger the below table shows which fields are required and how that information is being used in Teiid runtime currently as of Teiid version 7.5. A ODS file attached if you like to modify.

Table B.1. Data Usage for Tables

TABLE	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the Table
NameInSource	String	Yes	Yes	Yes	Name of Table in the source system, for view this can be empty, also used on variety of use cases
Cardinality	Integer	Yes	Yes	Yes	Cardinality is used to calculate the cost of source node access
TableType	Integer	Yes	Yes	Yes	Table,View,Document,XmlMappin
IsVirtual	Boolean	Yes	Yes	Yes	Used to find if this is source table Vs view
IsSystem	Boolean	Yes	Yes	No	Only used for System metadata
IsMaterialized	Boolean	Yes	Yes	Yes	To identify that the table is materialized

TABLE	Type	In Designer	In Metadata API	Required	Description
SupportsUpdate	Boolean	Yes	Yes	Yes	To allow updates on the table
PrimaryKeyID	String	Yes	KeyRecord	Yes	Used for creating indexes on temp tables and to create default update/delete procedures
ForeignKeyIDs	Collection	Yes	List<ForeignKey>	Yes	Used in Planning of query (rule raise access)
IndexIDs	Collection	Yes	List<KeyRecord>	Yes	Used for creating indexes on temp tables and in planning (estimate predicate cost)
UniqueKeyIDs	Collection	Yes	List<KeyRecord>	Yes	Used for query planning
AccessPatternIDs	Collection	Yes	List<KeyRecord>	Yes	Used for enforcing the criteria on query
MaterializedTableID	String	Yes	Table	Yes	Reference to Materialization table
insertEnabled	Boolean	**	Yes	Yes	Flag for checking insert procedure is enabled for view

TABLE	Type	In Designer	In Metadata API	Required	Description
deleteEnabled	Boolean	**	Yes	Yes	Flag for checking delete procedure is enabled for view
updateEnabled	Boolean	**	Yes	Yes	Flag for checking update procedure is enabled for view
Select Transformation	String	**	Yes	Yes	Transformation for Select in case of View
Insert Plan	String	**	Yes	Yes	Transformation for Insert in case of View
Update Plan	String	**	Yes	Yes	Transformation for Update in case of View
Delete Plan	String	**	Yes	Yes	Transformation for Delete in case of View
Bindings	Collection	**	Yes	Yes	XML Document
SchemaPaths	Collection	**	Yes	Yes	XML Document

Table B.2. Data Usage for Columns

COLUMN	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the column
NameInSource	String	Yes	Yes	Yes	Name of the column in source system

COLUMN	Type	In Designer	In Metadata API	Required	Description
IsSelectable	Boolean	Yes	Yes	Yes	Column is allowed in select
IsUpdatable	Boolean	Yes	Yes	Yes	Column is allowed in Update/ Insert/Delete
NullType	Integer	Yes	Yes	Yes	Used for validation if null value allowed
IsAutoIncrementable	Boolean	Yes	Yes	Yes	During insert used to validate if a value is required or not
IsCaseSensitive	Boolean	Yes	Yes	??	??
IsSigned	Boolean	Yes	Yes	??	Used in System Metadata
IsCurrency	Boolean	Yes	Yes	No	Only used for System metadata
IsFixedLength	Boolean	Yes	Yes	No	Only used for System metadata
IsTranformationInputParameter	Boolean	Yes	??	??	??
SearchType	Integer	Yes	Yes	Yes	Used for defining the capability of the source
Length	Integer	Yes	Yes	??	Used in System Metadata
Scale	Integer	Yes	Yes	??	Used in System Metadata

COLUMN	Type	In Designer	In Metadata API	Required	Description
Precision	Integer	Yes	Yes	??	Used in System Metadata
CharOctetLength	Integer	Yes	Yes	No	only used for System metadata
Radix	Integer	Yes	Yes	??	Used in System Metadata
DistinctValues	Integer	Yes	Yes	Yes	Used for cost calculations, System metadata
NullValues	Integer	Yes	Yes	Yes	Used for cost calculations, System metadata
MinValue	String	Yes	Yes	Yes	Used for cost calculations, System metadata
MaxValue	String	Yes	Yes	Yes	Used for cost calculations, System metadata
Format	String	Yes	Yes	No	Only used for System metadata
RuntimeType	String	Yes	DataType	Yes	Data Type
NativeType	String	Yes	Yes	Yes	Translators can use this field to further plan
DatatypeObjectID	String	Yes	??	??	
DefaultValue	String	Yes	Yes	Yes	Used for Insert and procedure execute operations

COLUMN	Type	In Designer	In Metadata API	Required	Description
					when the values are not supplied
Position	Integer	Yes	Yes	Yes	Used in the index calculations

Table B.3. Data Usage for Primary Keys

PRIMARY KEY	Type	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
ForeignKeyIDs	Collection				Extends KeyRecord

Table B.4. Data Usage for Unique Keys

UNIQUE KEY	Type	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
ForeignKeyIDs	Collection				

Table B.5. Data Usage for Indexes

INDEX	Type	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				

Table B.6. Data Usage for Access Patterns

ACCESS PATTERNS	Type	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				

Table B.7. Data Usage for Result Sets

RESULT SET	Type	In Designer	In Metadata API	Required	Description
FullName	String				See DataType
NameInSource	String				
ColumnIDs	Collection				

Table B.8. Data Usage for Foreign Keys

FOREIGN KEY	Type	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
UniqueKeyID	String				

Table B.9. Data Usage for Data Types

DATA TYPE	Type	In Designer	In Metadata API	Required	Description
FullName	String			No	Only used for System metadata
NameInSource	String			No	Only used for System metadata
Length	Integer			No	Only used for System metadata

DATA TYPE	Type	In Designer	In Metadata API	Required	Description
PrecisionLength	Integer			No	Only used for System metadata
Scale	Integer			No	Only used for System metadata
Radix	Integer			No	Only used for System metadata
IsSigned	Boolean			No	Only used for System metadata
IsAutoIncrement	Boolean			No	Only used for System metadata
IsCaseSensitive	Boolean			No	Only used for System metadata
Type	Integer			No	Only used for System metadata
SearchType	Integer			No	Only used for System metadata
NullType	Integer			No	Only used for System metadata
JavaClassName	String			Yes	Maps to runtime type based on java class name
RuntimeTypeName	String			No	Only used for System metadata
DatatypeID	String			No	Only used for System metadata

DATA TYPE	Type	In Designer	In Metadata API	Required	Description
BaseTypeID	String			No	Only used for System metadata
PrimitiveTypeID	String			No	Only used for System metadata
VarietyType	Integer			No	Only used for System metadata
VarietyProps	Collection			No	Only used for System metadata

Table B.10. Data Usage for Procedures

PROCEDURE	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the column
NameInSource	String	Yes	Yes	Yes	Name of the column in source system
IsFunction	Boolean		Yes	Yes	Determines if this function
IsVirtual	Boolean		Yes	Yes	If Function then UDF else stored procedure
ParametersIDs	Collection		Yes	Yes	Parameter List
ResultSetID	String		Yes	Yes	Result set columns
UpdateCount	Integer		Yes	Yes	Update count defines the number of sources being updated, only applicable

PROCEDURE	Type	In Designer	In Metadata API	Required	Description
					for virtual procedures

Table B.11. Data Usage for Procedure Parameters

PROCEDURE PARAMETER	Type	In Designer	In Metadata API	Required	Description
ObjectID	String				Same as Column
FullName	String				Same as Column
nameInSource	String				Same as Column
defaultValue	String				Same as Column
RuntimeType	String				Same as Column
DatatypeObjectID	String				Same as Column
Length	Integer				Same as Column
Radix	Integer				Same as Column
Scale	Integer				Same as Column
NullType	Integer				Same as Column
Precision	Integer				Same as Column
Position	Integer				Same as Column
Type	String			Yes	Defines parameter is IN/OUT/RETURN
Optional	Boolean			No	Defines if the parameter is optional or not, only

PROCEDURE PARAMETER	Type	In Designer	In Metadata API	Required	Description
					used system metadata

Table B.12. Data Usage for SQL Transformations

SQL NSFORMATIOI	Type	In Designer	In Metadata API	Required	Description
VirtualGroupName	String	Yes	No	Yes	See Table, the properties defined on Table
TransformedObjectID	String	Yes	No	Yes	See Table, the properties defined on Table
TransformationObjectID	String	Yes	No	Yes	See Table, the properties defined on Table
TransformationSql	String	Yes	No	Yes	See Table, the properties defined on Table
Bindings	Collection	Yes	No	Yes	See Table, the properties defined on Table
SchemaPaths	Collection	Yes	No	Yes	See Table, the properties defined on Table

Table B.13. Data Usage for VDBs

VDB	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	vdb.xml	Yes	Name of the VDB
NameInSource	String	??	No	No	Not required
Version	String	Yes	vdb.xml	Yes	VDB version

VDB	Type	In Designer	In Metadata API	Required	Description
Identifier	String	Yes	No	No	Not required
Description	String	Yes	vdb.xml	No	Used by System metadata
ProducerName	String	Yes	No	No	Not required
ProducerVersion	String	Yes	No	No	Not required
Provider	String	Yes	No	No	Not required
TimeLastChanged	String	Yes	No	No	Not required
TimeLastProduced	String	Yes	No	No	Not required
ModelIDs	Collection	Yes	vdb.xml	Yes	Defines the model list in a VDB

Table B.14. Data Usage for Annotations

ANNOTATION	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	No	System metadata, as description on procedure parameter
NameInSource	String	Yes	No	No	Not required
Description	String	Yes	No	No	Not required

Appendix C. 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.

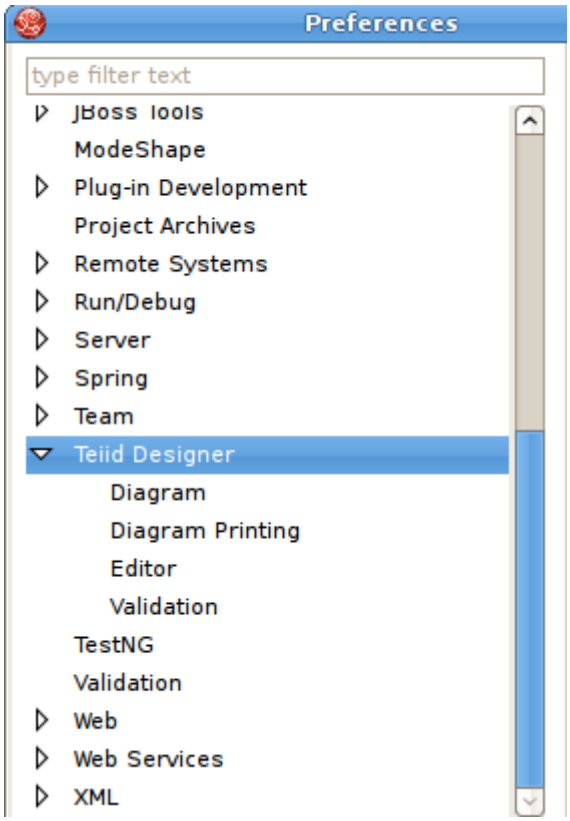


Figure C.1. Preferences Dialog

C.1. Teiid Designer Preferences

General Teiid Designer preferences include.

- **Enable auto-creating of a source model's data source on Teiid Server** - indicates if data sources that match the default name should be auto-created if they do not exist on Teiid server.
- **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.

- **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 beging 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.

- **Default Teiid Server Version** - determines what version of server modelling will be targeted at, **if no teiid server has been defined in the Servers View**. Thus, it is possible to still design models without the need to define and connect to a server. However, possible values are confined to the teiid runtime clients installed.

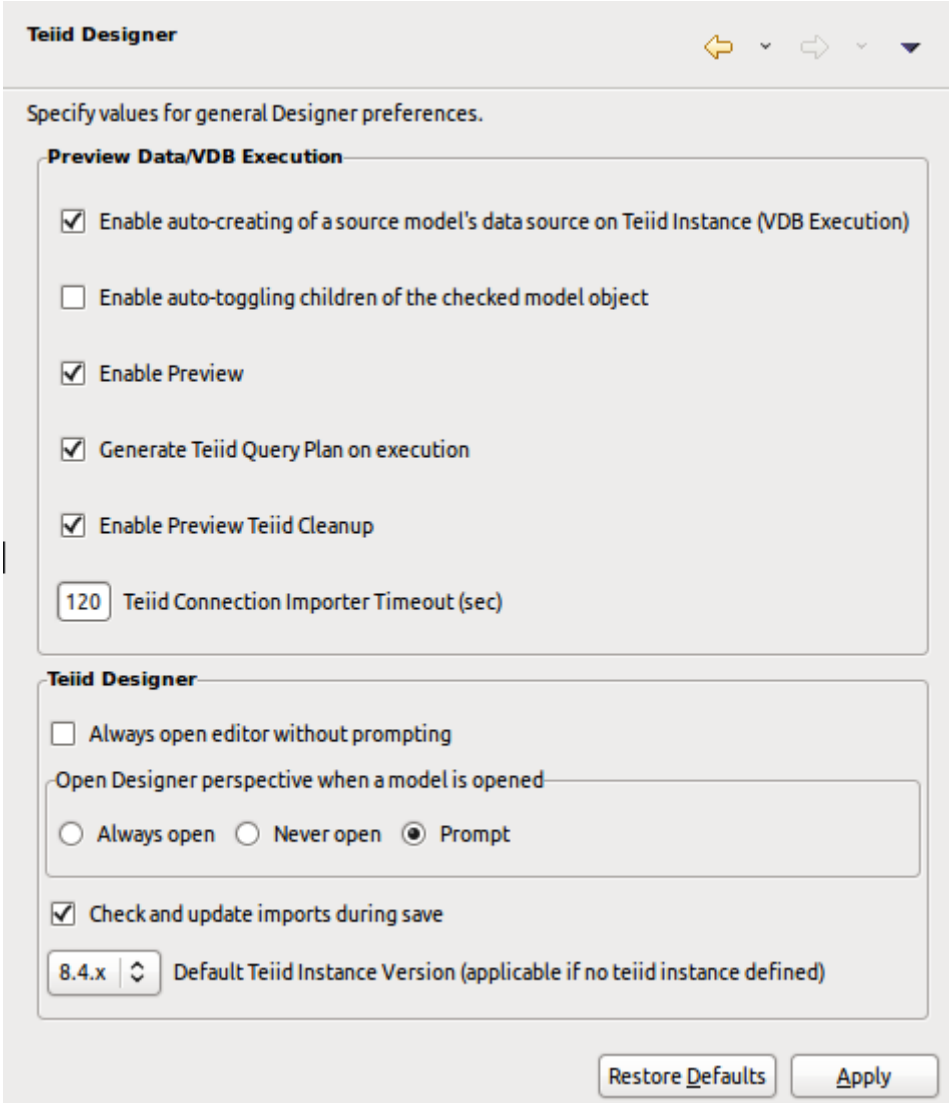


Figure C.2. General Teiid Designer Preferences Panel

C.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**.

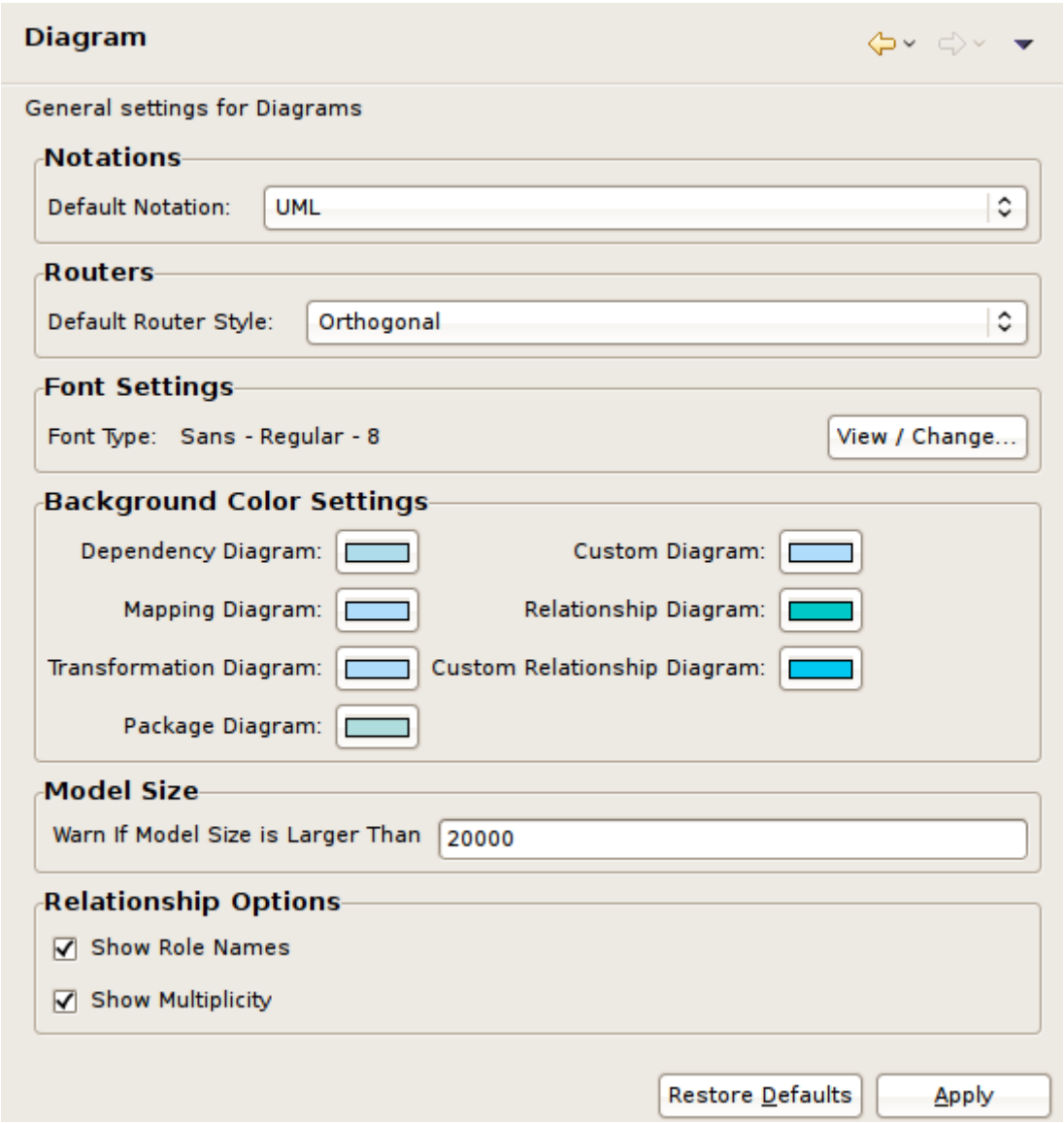


Figure C.3. Diagram Preferences Panel

C.1.2. Editor Preferences

C.1.2.1. XML Document Preferences

XML Document Mapping Preferences provide ways to customize *Section D.3.1.2.4, "Mapping Diagram"* and *Section 6.3.4, "Recursion Editor (XML)"* behavior.

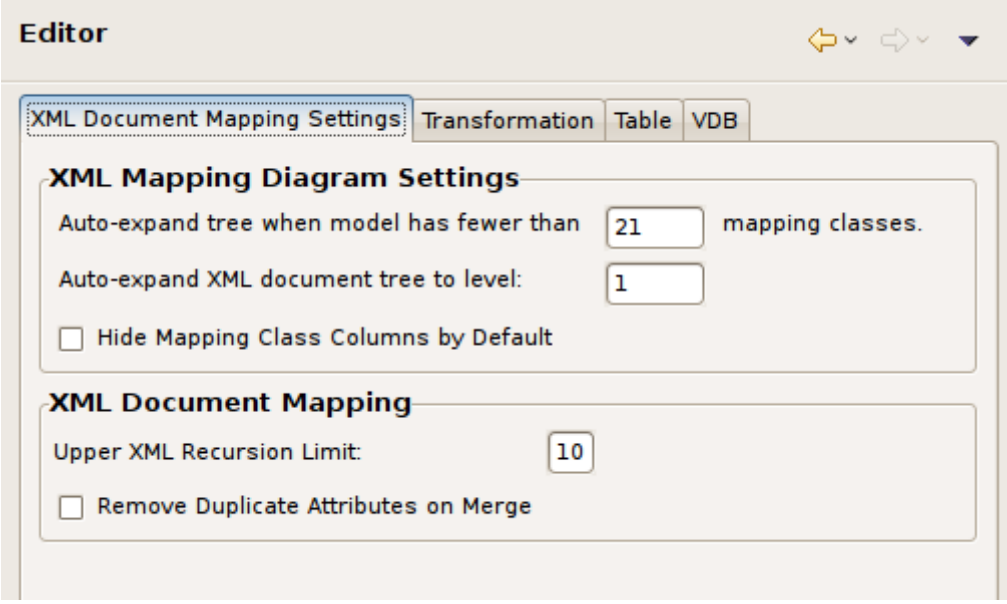


Figure C.4. XML Document Preferences Panel

C.1.2.2. Table Editor Preferences

Section D.3.1.3, "Table Editor" **Preferences** provide a way to customize the order and the information content for each model object type.

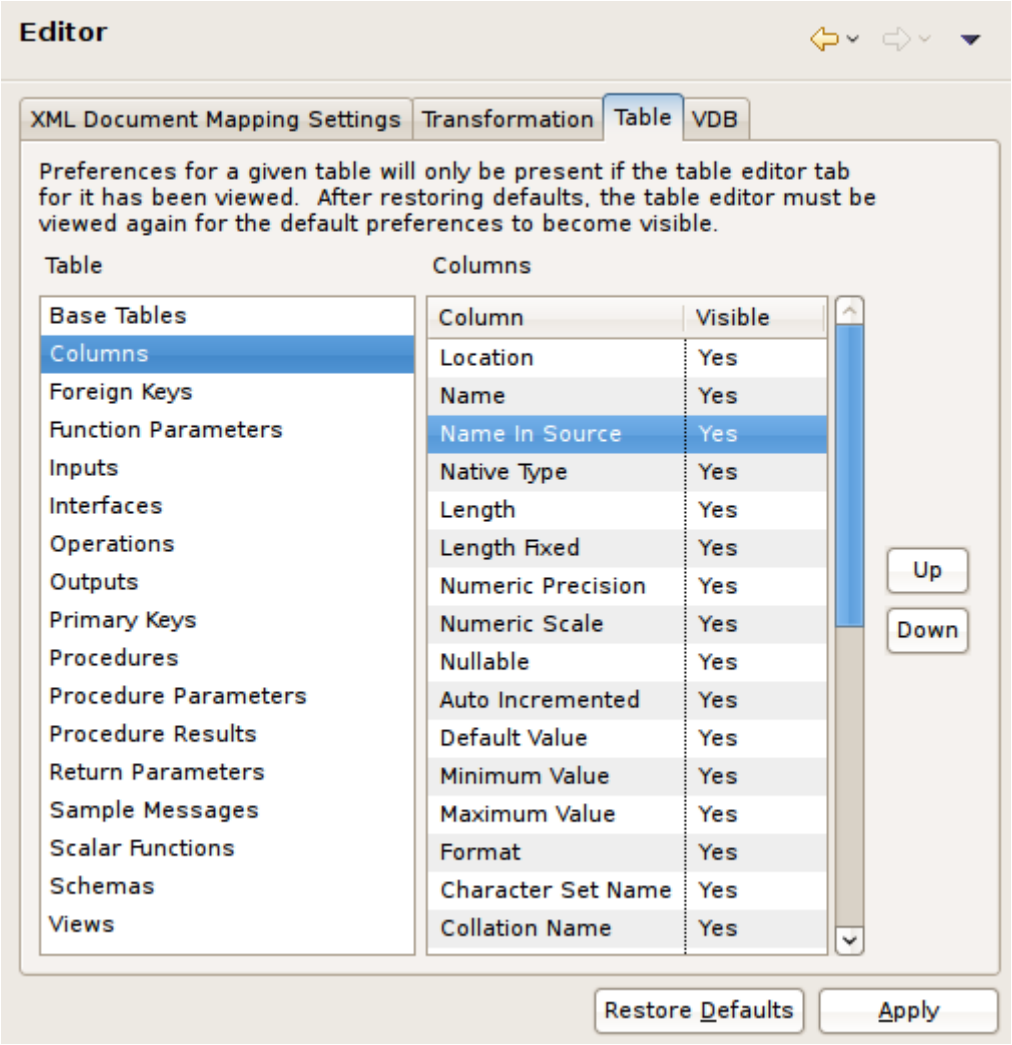


Figure C.5. Table Editor Preferences Panel

C.1.2.3. Transformation Editor Preferences

Section 6.3.1, "Transformation Editor" **Preferences** provide a way to customize SQL formatting, diagram layout, and default view entity properties.

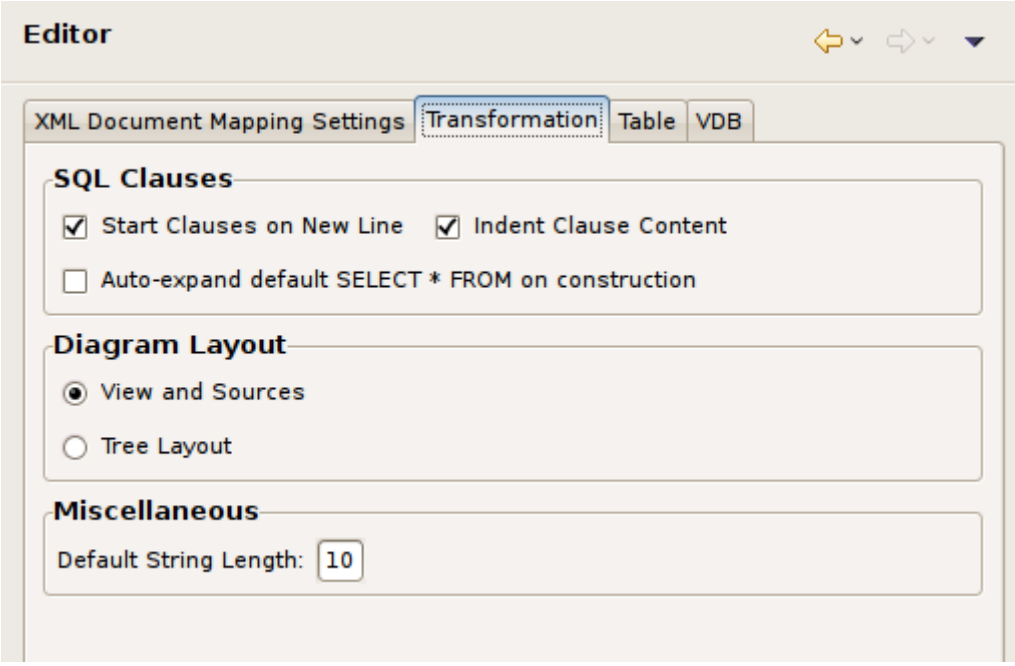


Figure C.6. Transformation Editor Preferences Panel

C.1.2.4. VDB Editor Preferences

Section D.3.2, "VDB Editor" **Preferences** provide a way to customize VDB editor behavior.

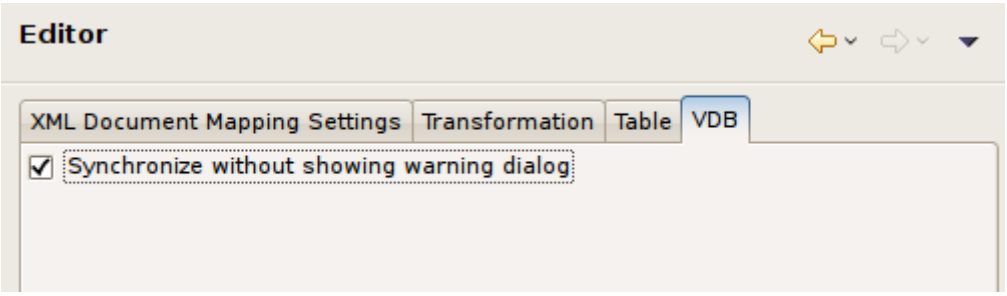


Figure C.7. VDB Editor Preferences Panel

C.1.3. Validation Preferences

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

The Validation preference pages, shown below, include the validation preferences for **Core, Relational, XML and XSD (XML Schema)** models.

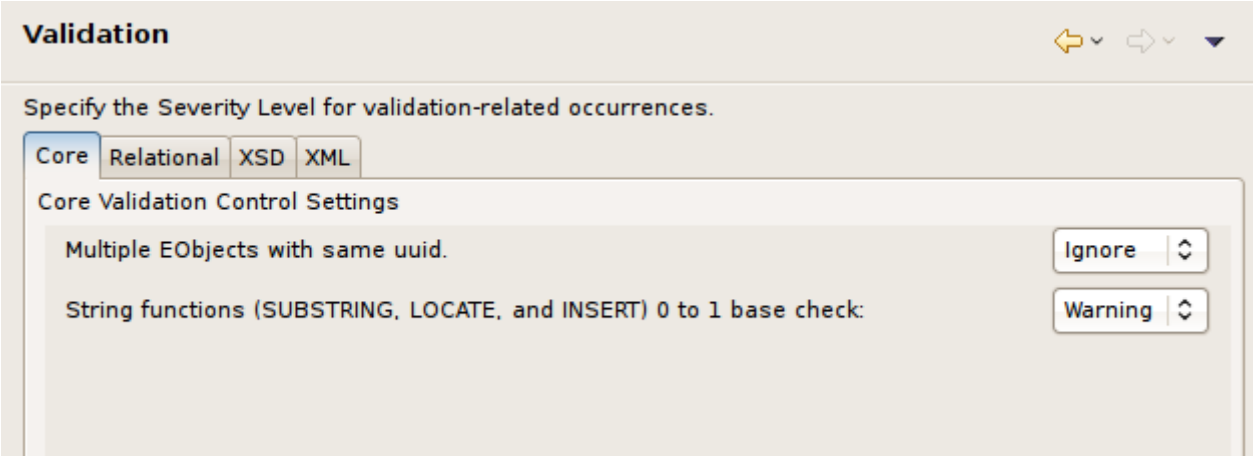


Figure C.8. Core Model Validation Preferences Panel

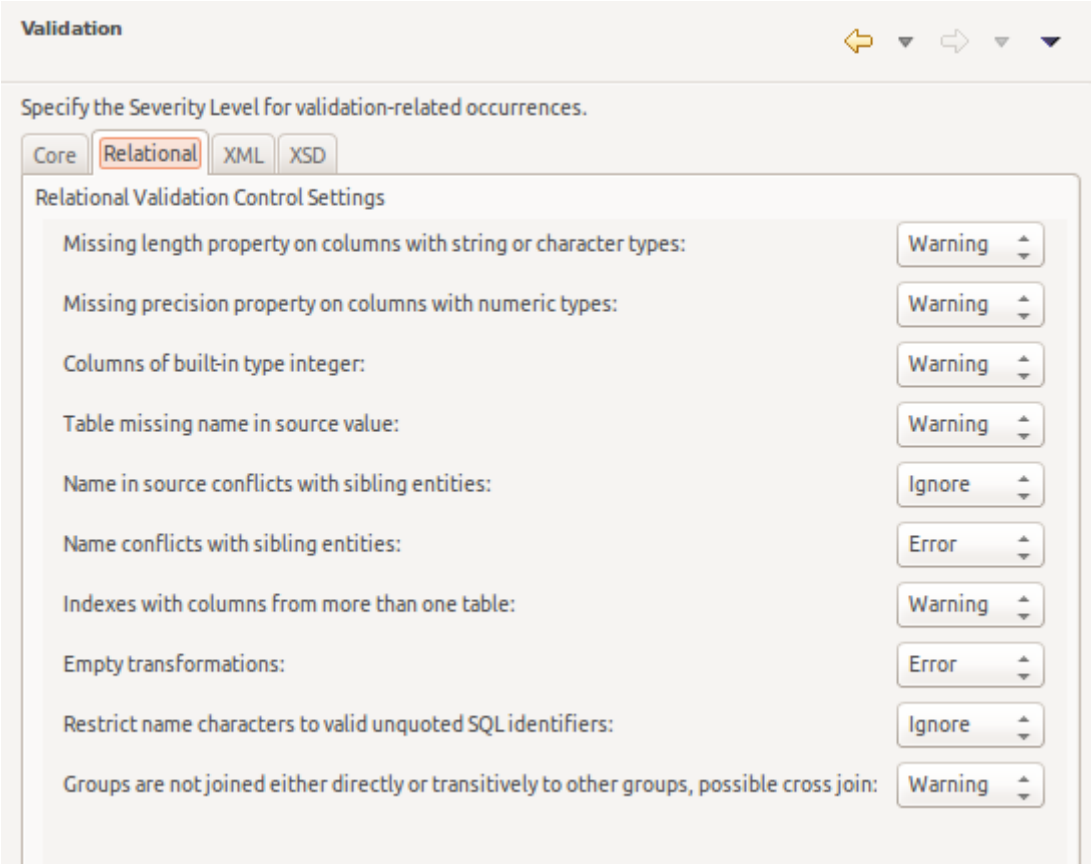


Figure C.9. Relational Model Validation Preferences Panel

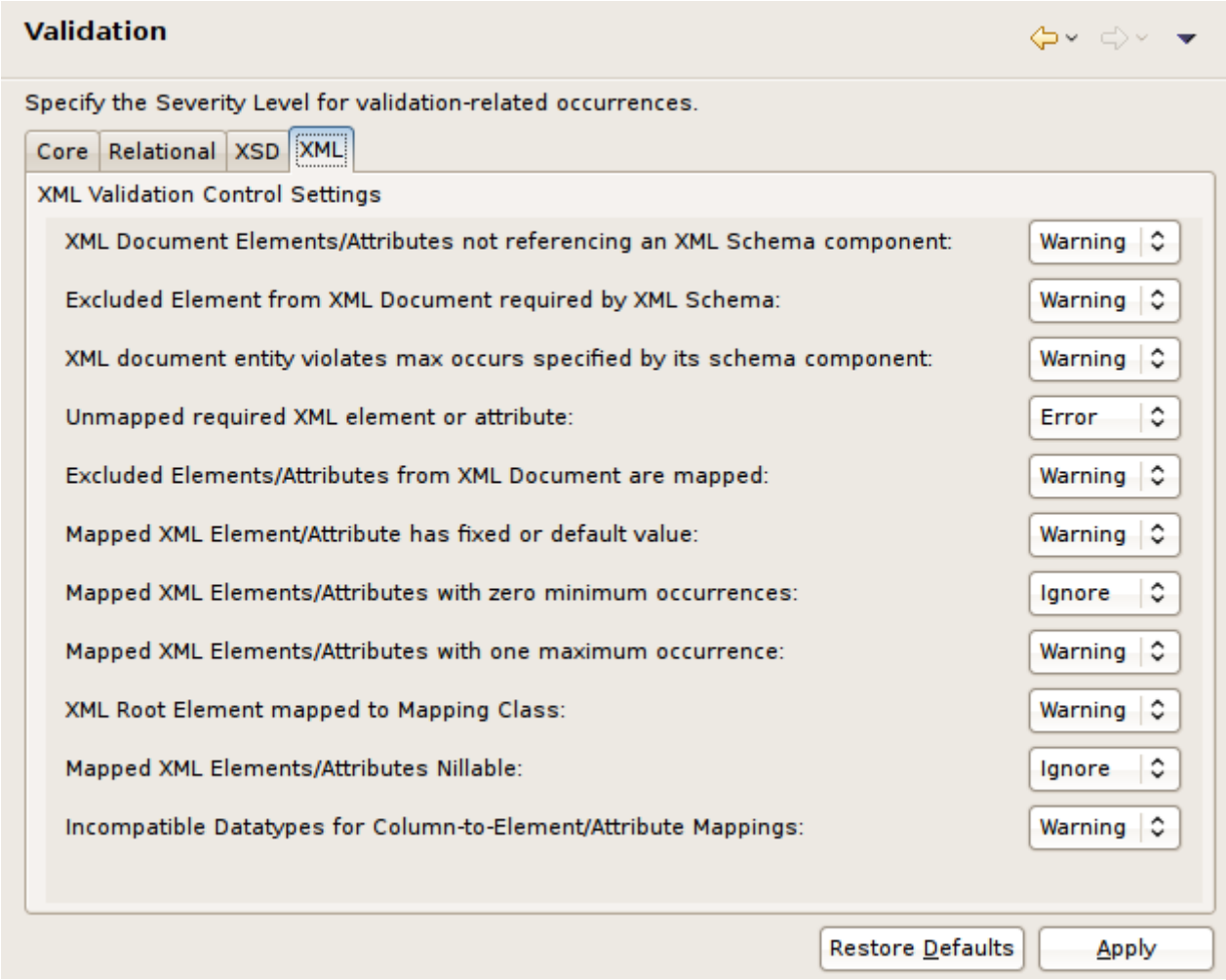


Figure C.10. XML Document Model Validation Preferences Panel

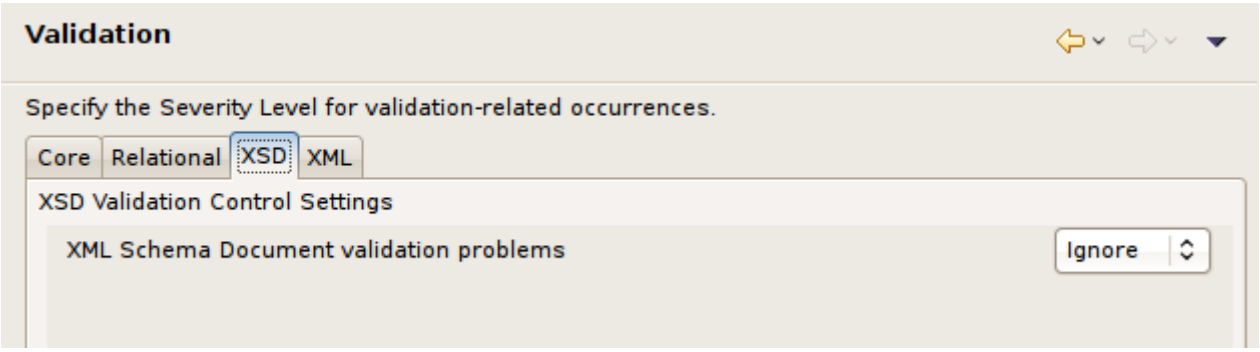


Figure C.11. XSD Schema Model Validation Preferences Panel

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.

Appendix D. Teiid Designer Ui Reference

D.1. Teiid Designer Perspectives

Teiid Designer utilizes the *Eclipse* [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.

D.1.1. Teiid Designer Perspective

The Teiid Designer perspective provides access to fundamental model editing and management capabilities. This perspective includes 6 main UI components (or groups of components) as shown below. They include:

- *Section D.2.1, "Model Explorer View"* - Teiid 'tree' view of Model Objects.
- *???* - Teiid Server instance view. Provides view of contents for connected instances of installed Teiid runtime.
- *Section D.3.1, "Model Editor"* - Custom editors targeted for ".xmi" metadata model files.
- *Section D.2.5, "Properties View"* - Standard property values for selected workbench objects.
- *Section D.2.14, "Guides View"* - Guides and Status Views which enhance usability.
- **Miscellaneous Views** - Includes the Problems view, Message Log view and the SQL Results view (opened if Preview Data action is performed)

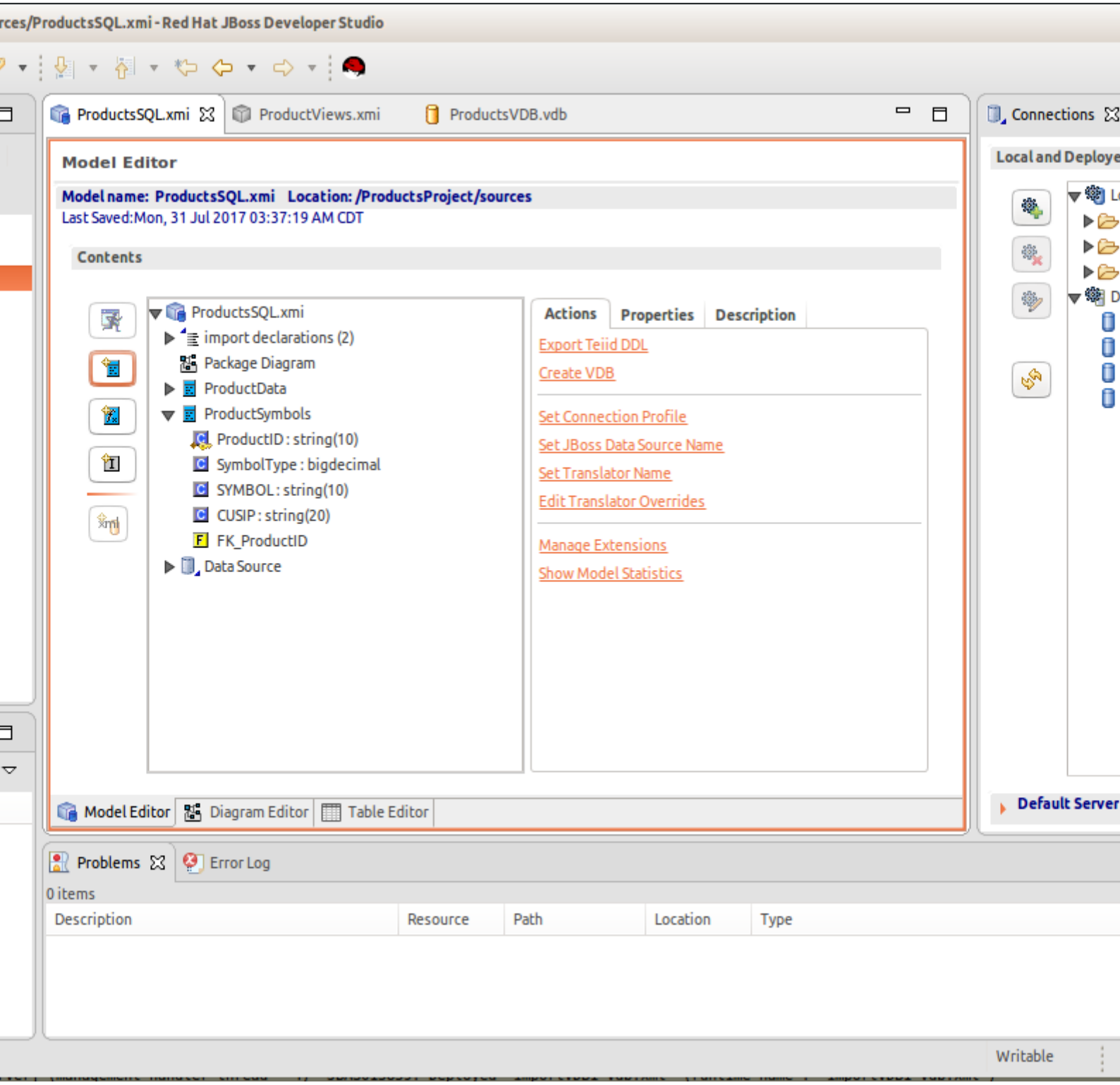


Figure D.1. Teiid Designer Perspective Layout

D.1.2. 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 D.2. Perspectives Menu

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

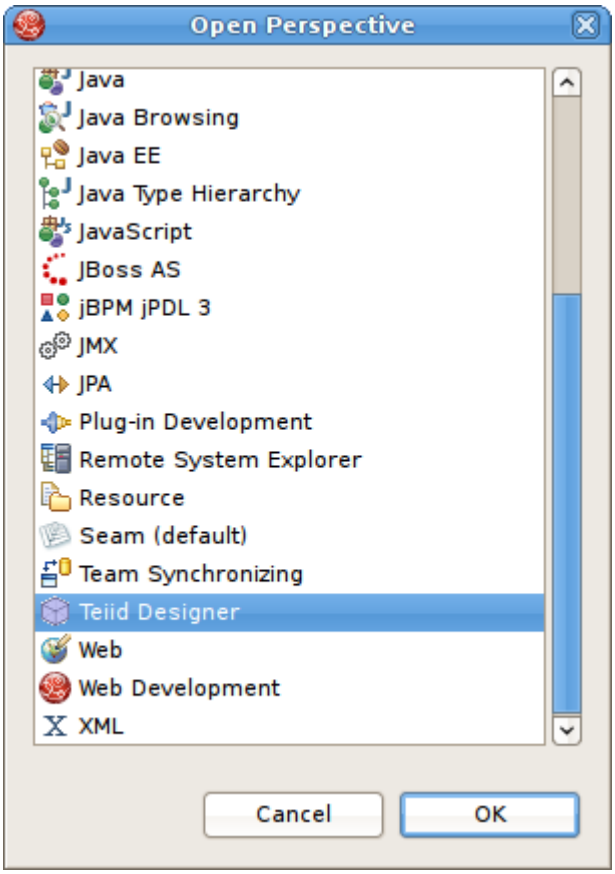


Figure D.3. Select Perspective Dialog

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.

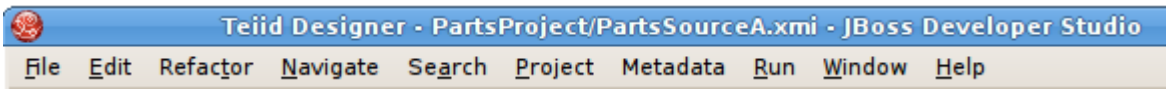


Figure D.4. Workbench Window Title Bar

- 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.

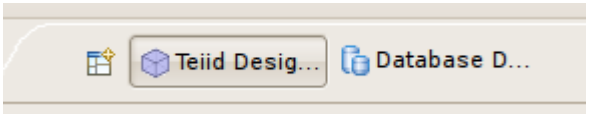


Figure D.5. Workbench Window Title Bar

D.1.3. Further information

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

D.2. Teiid Designer Views

Views are dockable windows which present data from your models or your modeling session in various forms. Some views support particular *Section D.3.1, "Model Editor"* and their content is dependent on workspace selection. This section summarizes most of the views used and available in Teiid Designer. The full list is presented in the main menu's *Window* > *Show View* > *Other...* dialog under the **Teiid Designer** category.

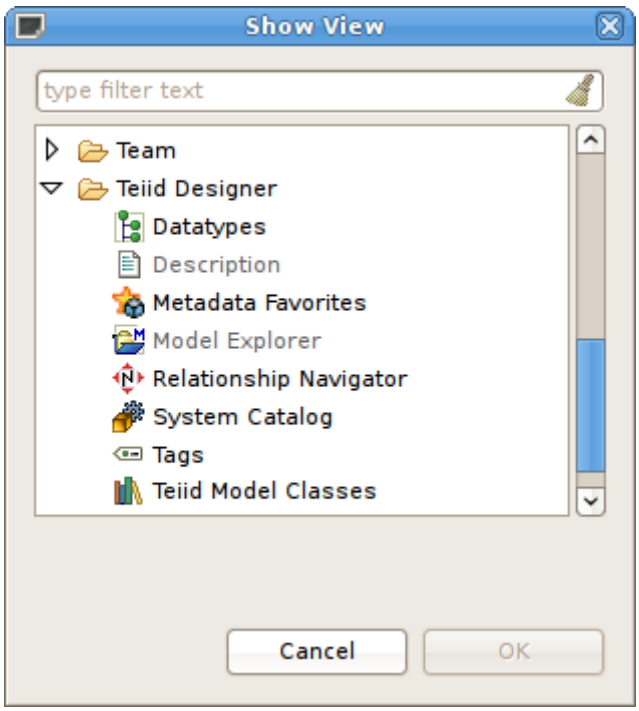


Figure D.6. Eclipse Show View Dialog

D.2.1. 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.

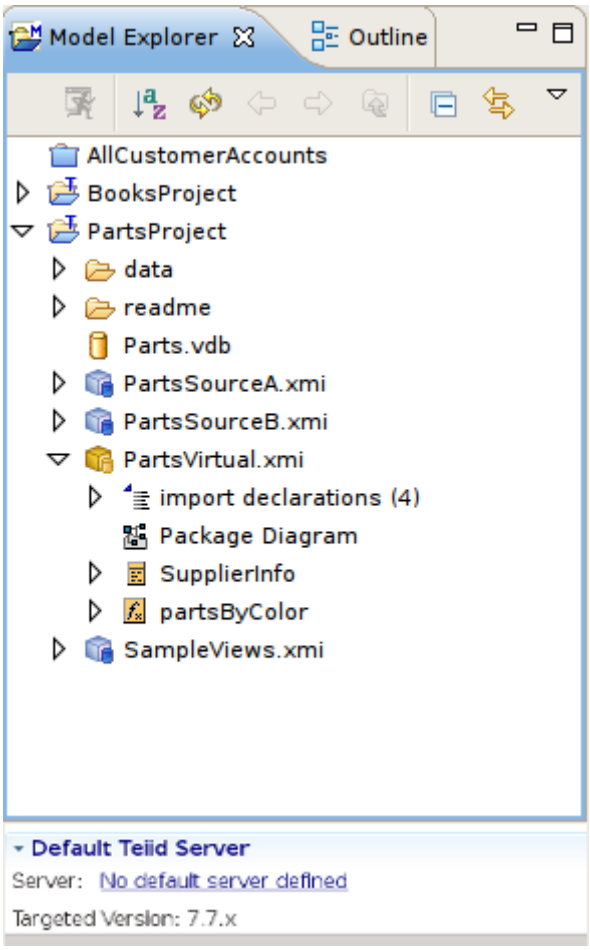


Figure D.7. Model Explorer View

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:

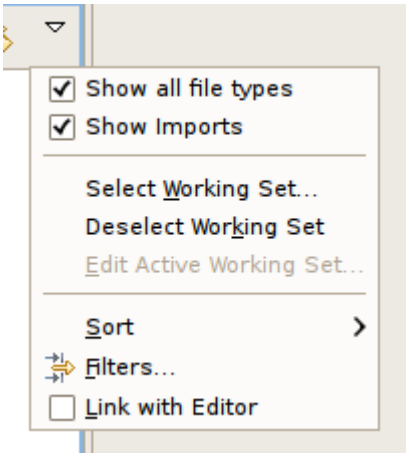


Figure D.8. Additional Actions

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

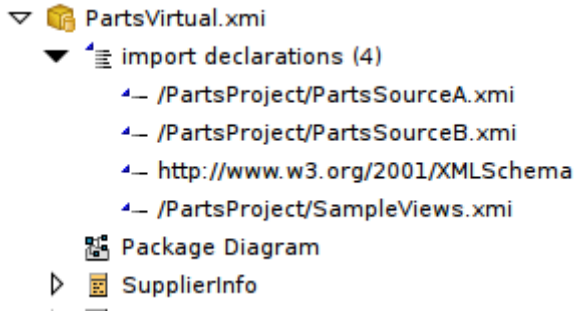


Figure D.9. Show Model Imports Action

D.2.1.1. Selection-Based Action Menus

Selecting specific objects in the **Model Explorer** provides a context from which 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.

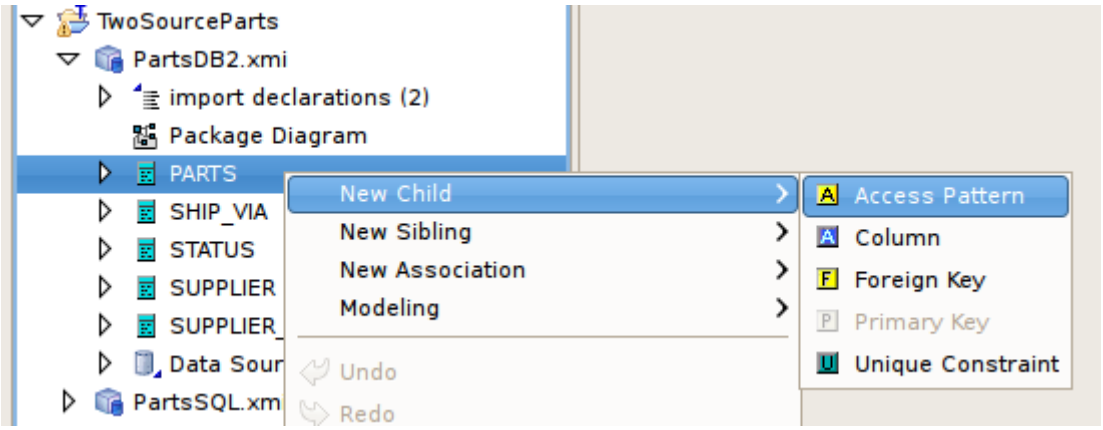


Figure D.10. Sample Context Menu

D.2.2. Outline View

The Outline View is a utility view which provides both at 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).

D.2.2.1. 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.

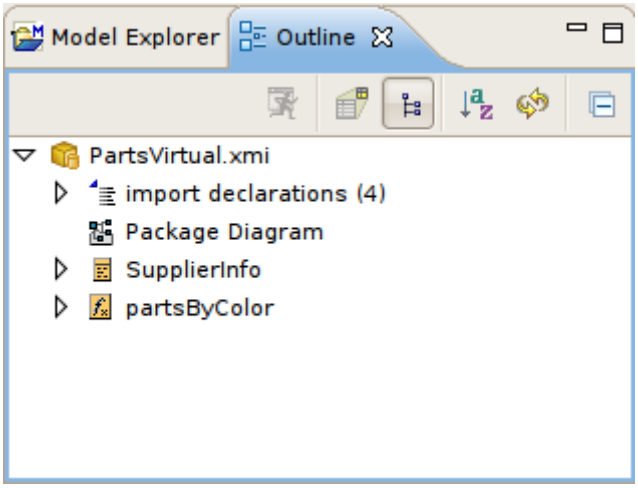


Figure D.11. Outline View

D.2.2.2. 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.

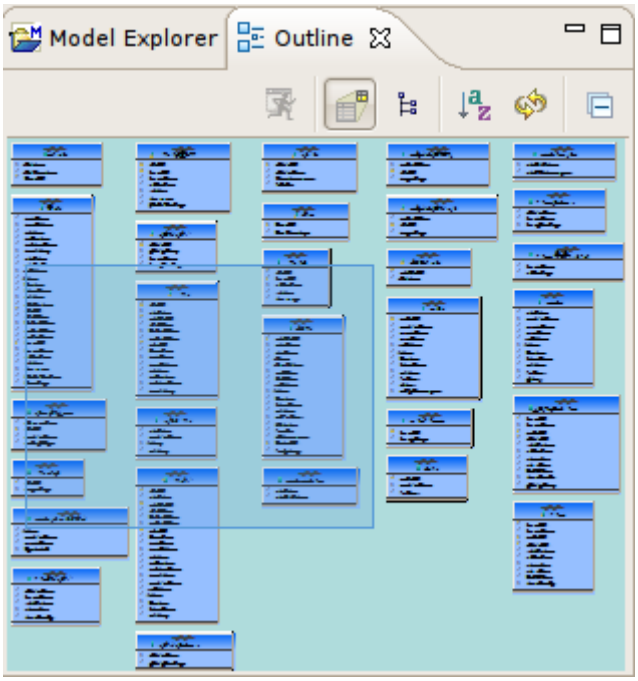
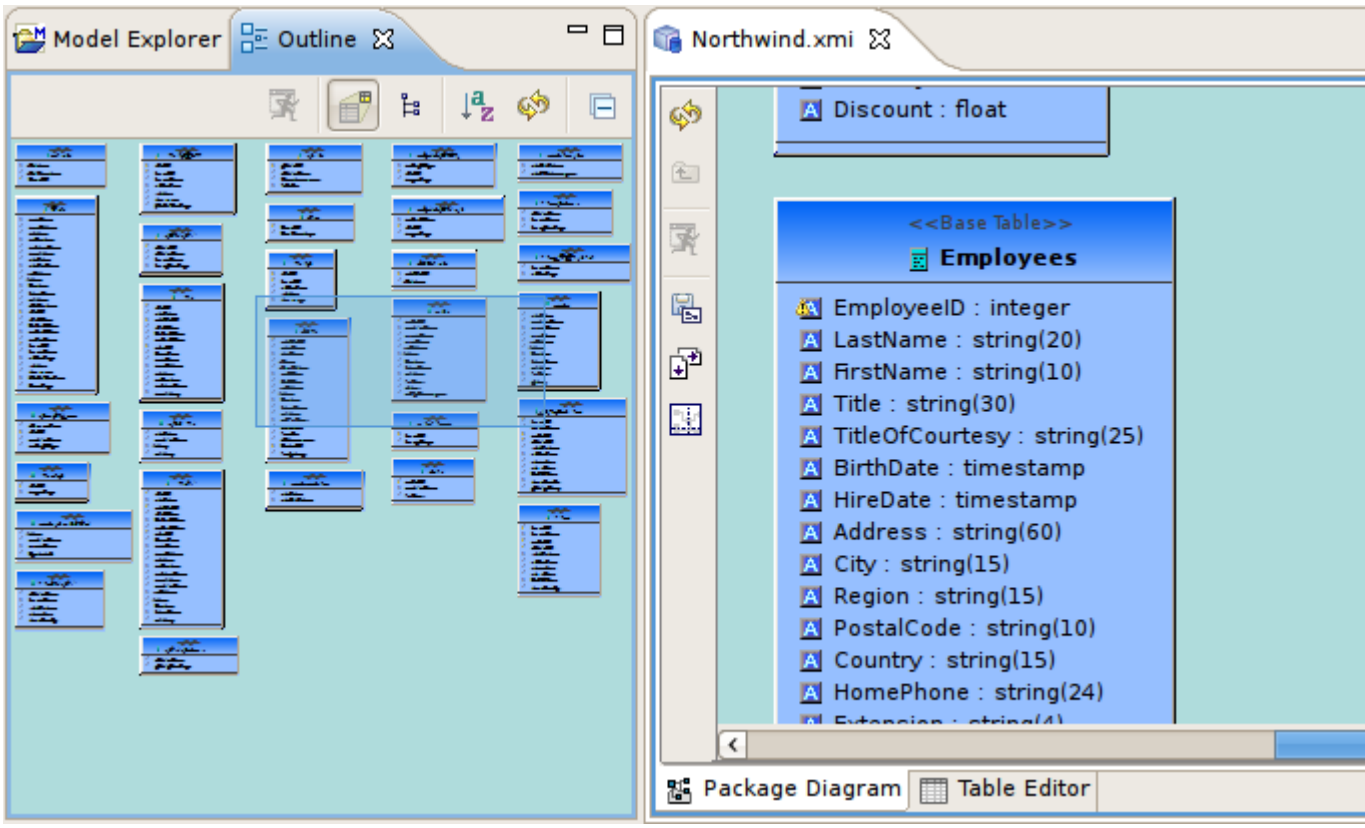


Figure D.12. Outline View

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.



D.2.3. Server View

The Server View provides a means to display and manage server instances and their contents within the Eclipse environment. Since Teiid is installed as part of a JBoss server, its contents is displayed as part of its JBoss parent.

To show the Server View click "**Window > Show View > Other...**" to display the Show View dialog. Choose "**Servers > Server**" view and click **OK**.

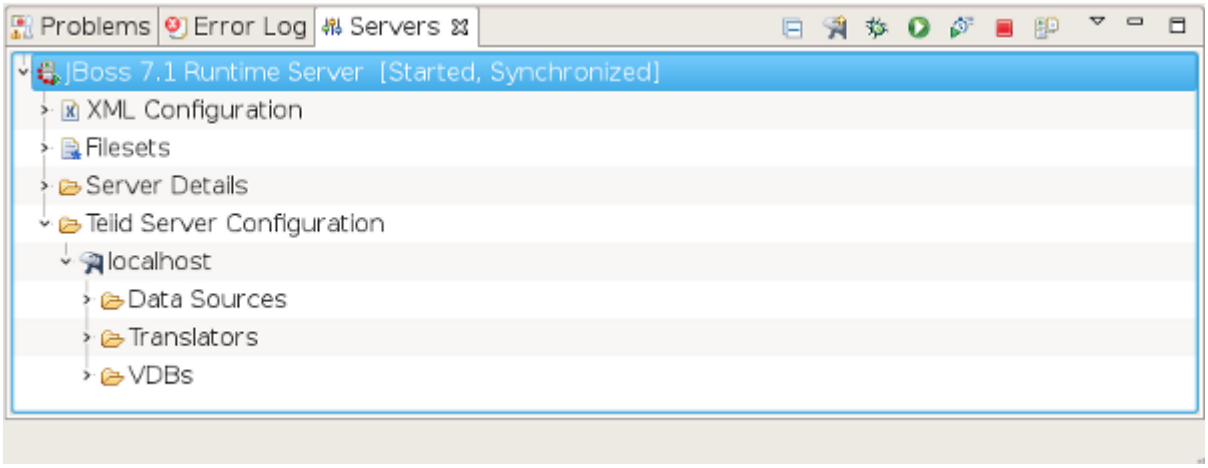


Figure D.13. Server View

To create your Teiid instance:

- Select the **New...** action in the Server view and this will launch the JBoss Server wizard. Configuring the JBoss Server instance (with Teiid installed) will enable connection to the Teiid Server.
- In the **New Server** wizard. Select the **JBoss AS 7.1.1** server type under the **JBoss Community** category and click **Next>**.

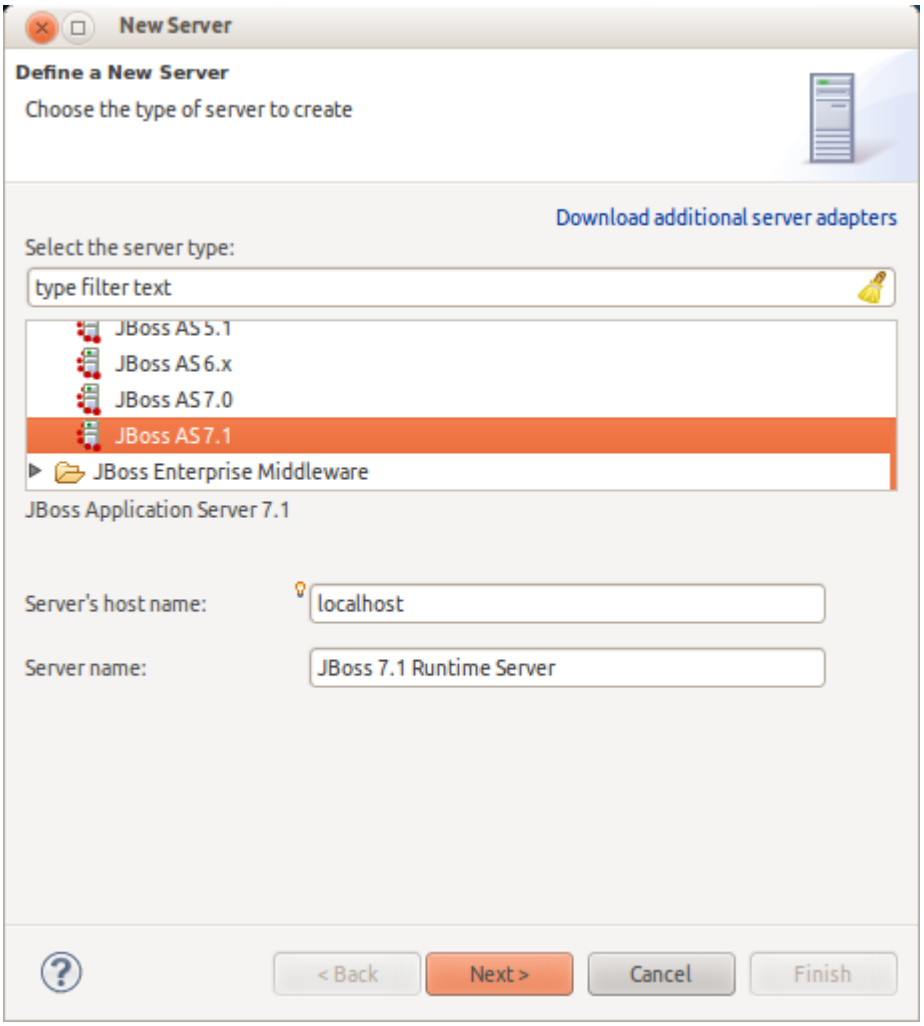


Figure D.14. New Server Dialog

- On the **JBoss Runtime** page, click the top **Browse...** button to select the installation folder of your **JBoss AS 7.1** server.

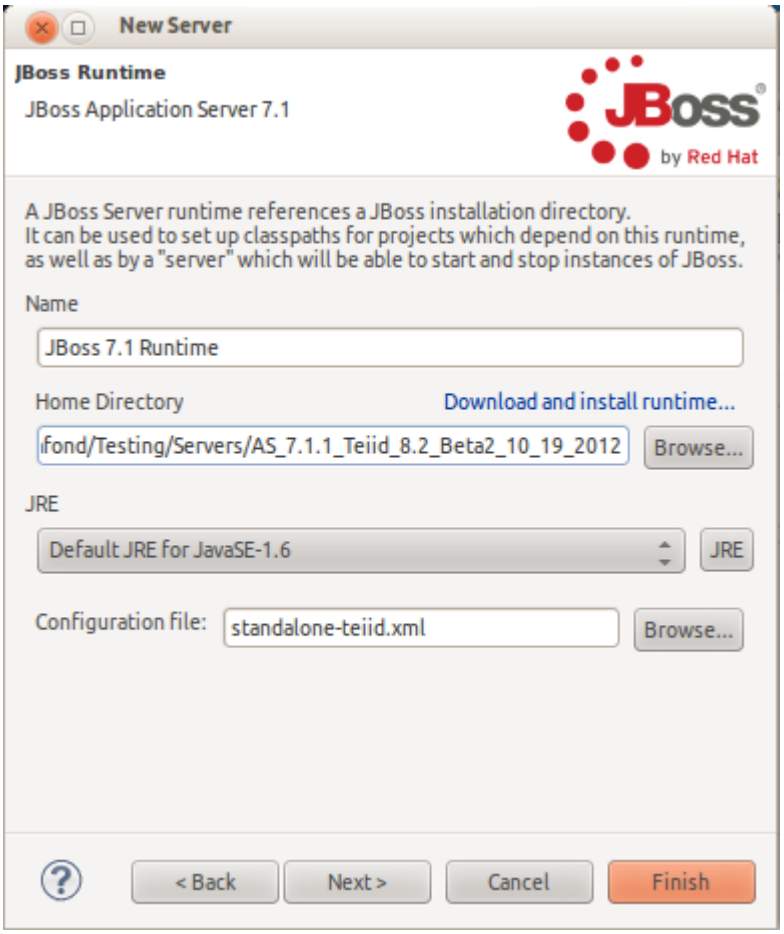


Figure D.15. JBoss Runtime Definition

- Then click the bottom **Browse...** button to select the **standalone-teiid.xml** configuration file located under the **standalone/configuration/** folder on your file system. Then click **Finish** to return the the **New Teiid Instance** dialog.

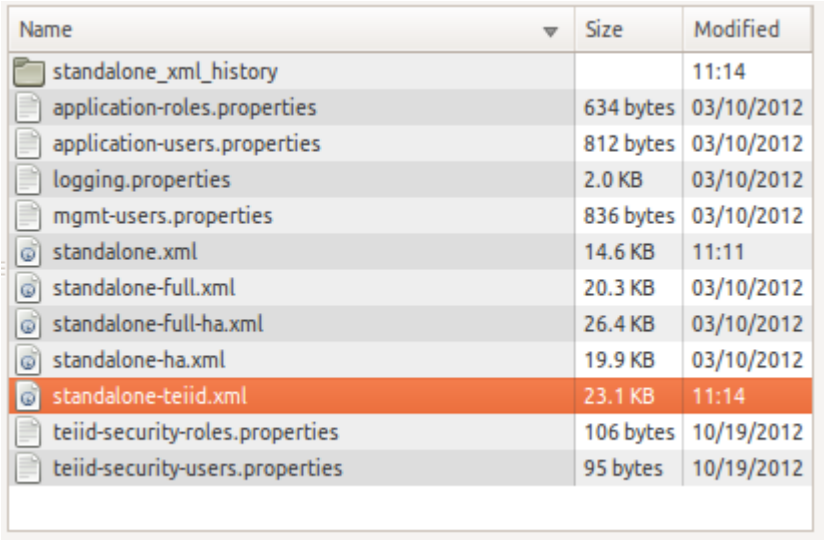
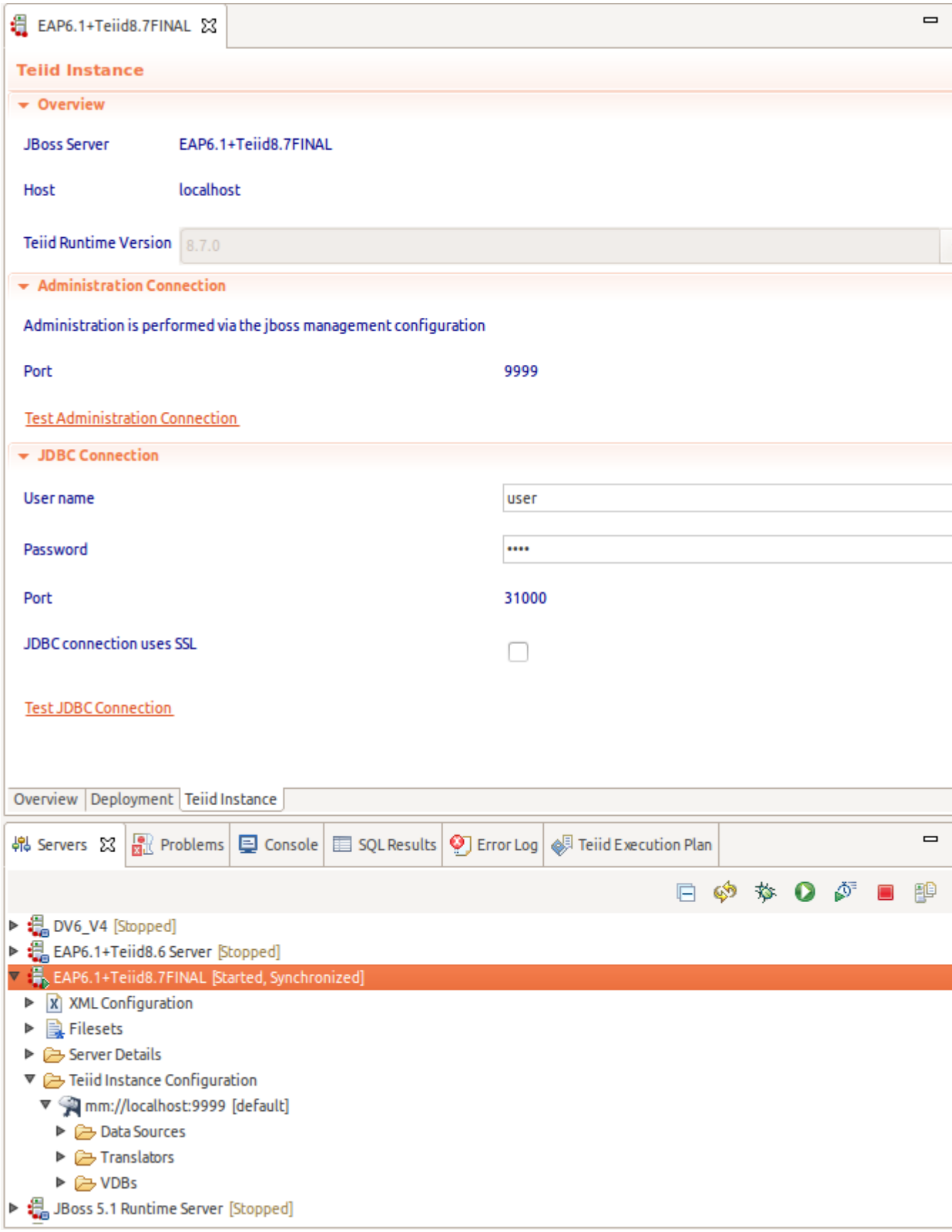


Figure D.16. Teiid Configuration File Selection

- Click **Finish** and your new Teiid server configuration will be opened in the **JBoss / Teiid Editor** for viewing. In this editor you can test both Teiid admin and JDBC connections.


EAP6.1+Teiid8.7FINAL
Teiid Instance
Overview
JBoss Server
EAP6.1+Teiid8.7FINAL
Host
localhost
Teiid Runtime Version
8.7.0
Administration Connection
Administration is performed via the jboss management configuration
Port
9999
Test Administration Connection
JDBC Connection
User name
user
Password
Port
31000
JDBC connection uses SSL
Test JDBC Connection
Overview
Deployment
Teiid Instance
Servers
Problems
Console
SQL Results
Error Log
Teiid Execution Plan
DV6_V4 [Stopped]
EAP6.1+Teiid8.6 Server [Stopped]
EAP6.1+Teiid8.7FINAL [Started, Synchronized]
XML Configuration
Filesets
Server Details
Teiid Instance Configuration
mm://localhost:9999 [default]
Data Sources
Translators
VDBs
JBoss 5.1 Runtime Server [Stopped]

Actions available in this view include:

-
  Teiid Server Properties - View and edit properties of an existing Teiid instance
-
  - Reconnect and refresh contents of 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

Note

Once the server is started, the **Server** view will now display the data source, translator and VDB content for your running Teiid server as shown below.

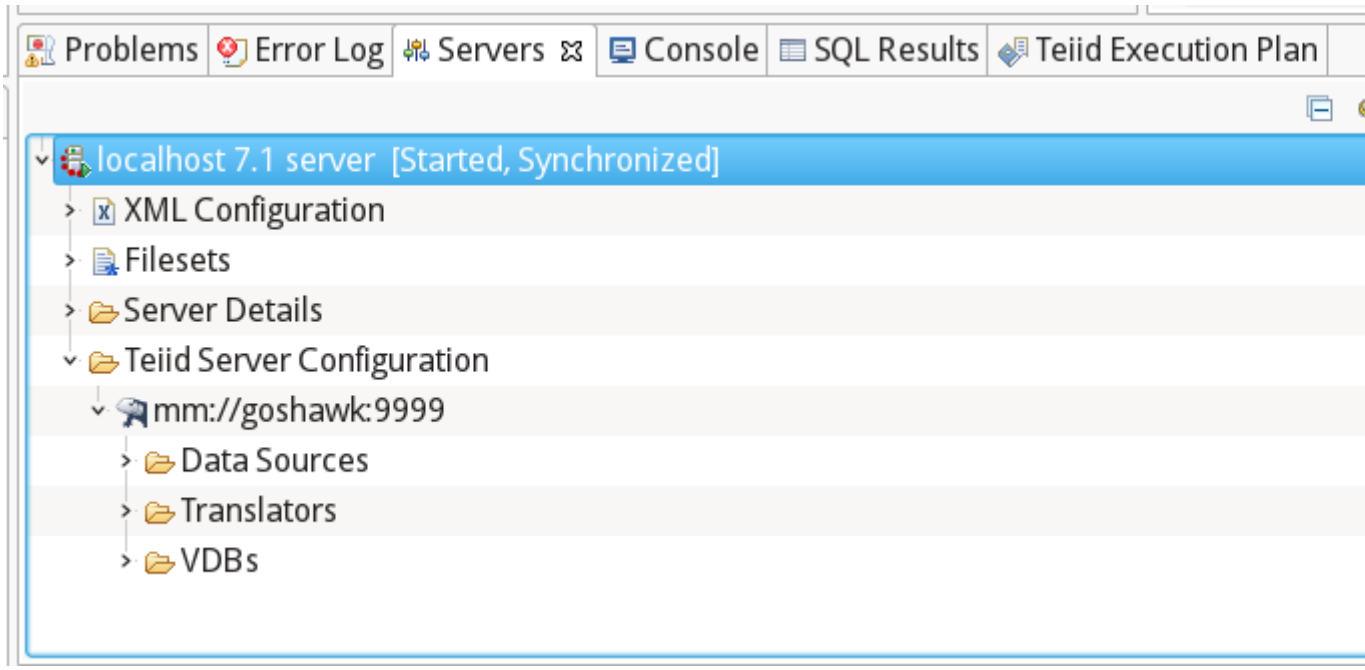


Figure D.18. Teiid Contents in Server View

D.2.4. Connections View

The **Connections View** allows editing and management of your local connection profiles as well as deployed data source configurations in one view.

The figure below shows 2 panels labeled **Local and Deployed Connections** and **Default Server**. The Connections panel contains a tree view containing locally defined connection profile definitions and a list of data sources configurations that are deployed on your default server.

The toolbar on the left contains create, delete and edit actions as well as a refresh button.

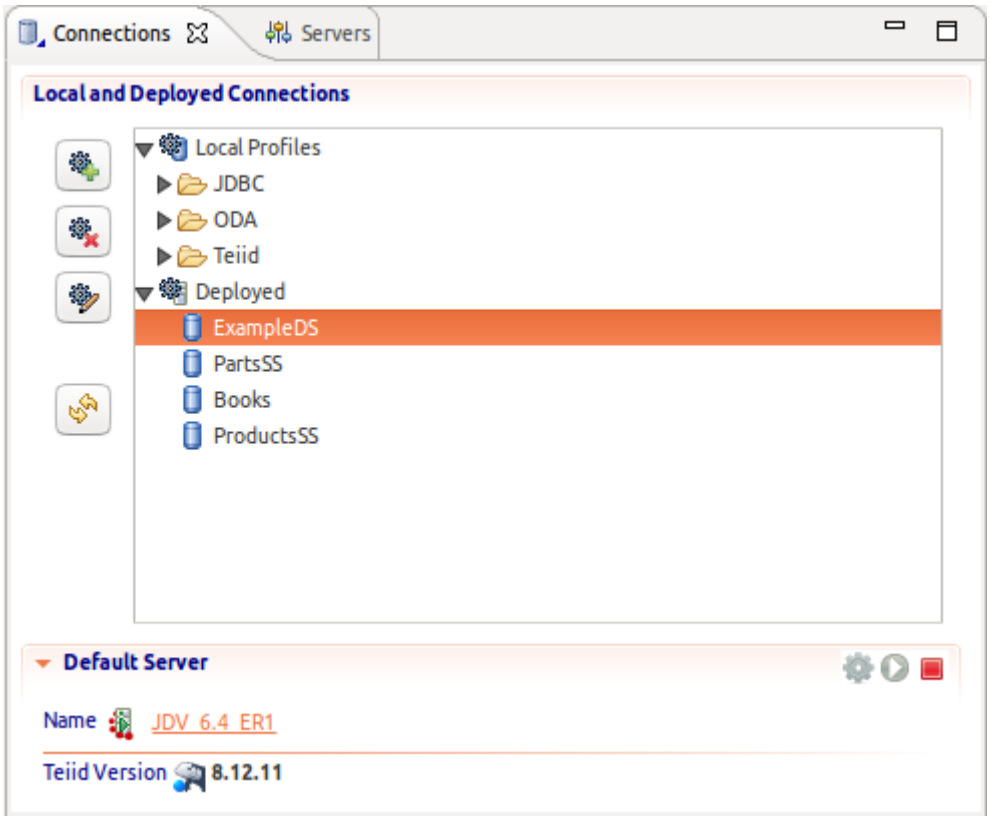


Figure D.19. Connections View

In addition to managing your connections, you can also create new source models (i.e. import) from selecting an existing profile or deployed data source and launching the corresponding import wizard via a **Generate Source Model** action (see below).

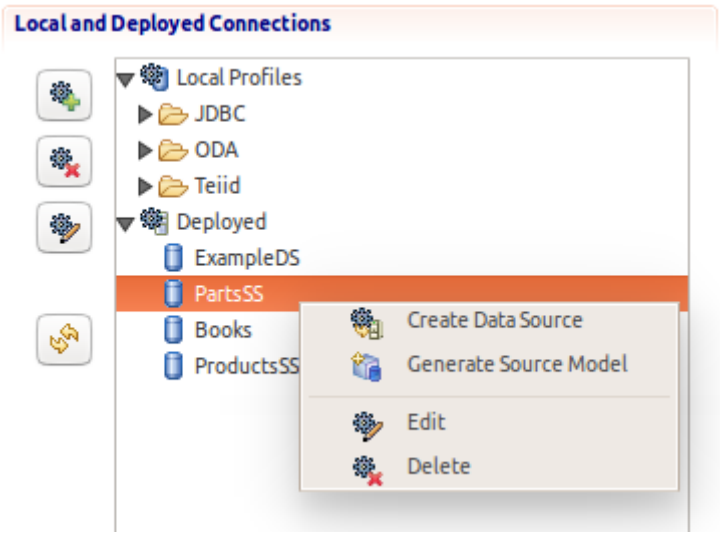


Figure D.20. Available Actions

The **Default Server** section includes a useful toolbar that includes start, and stop actions. Note that the default Designer perspective includes the standard **Servers** view as a tab behind the **Connections View**.

D.2.5. 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.

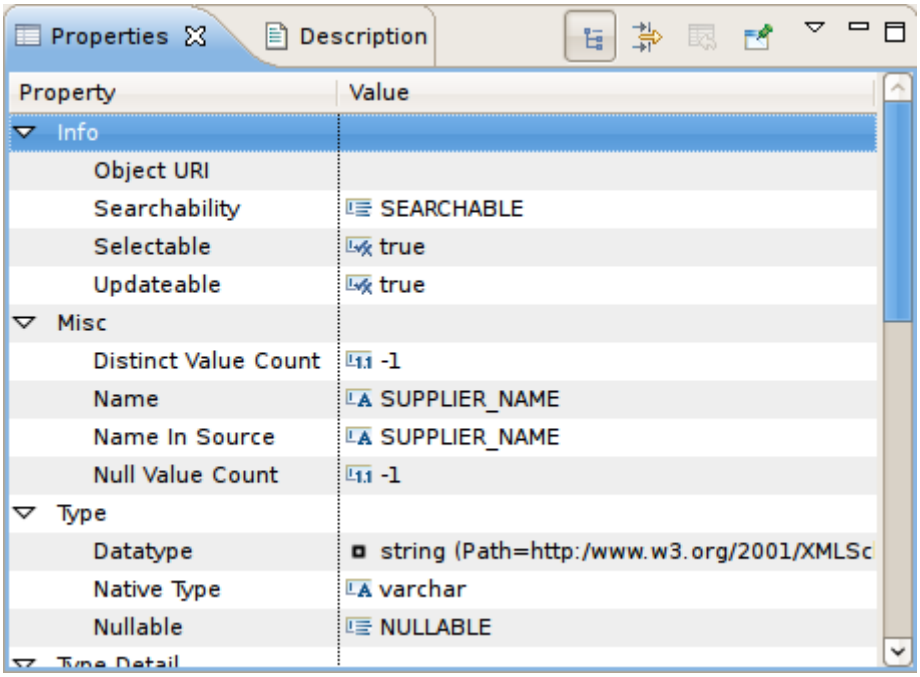


Figure D.21. Properties View

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 Teiid Designer's main preference page.

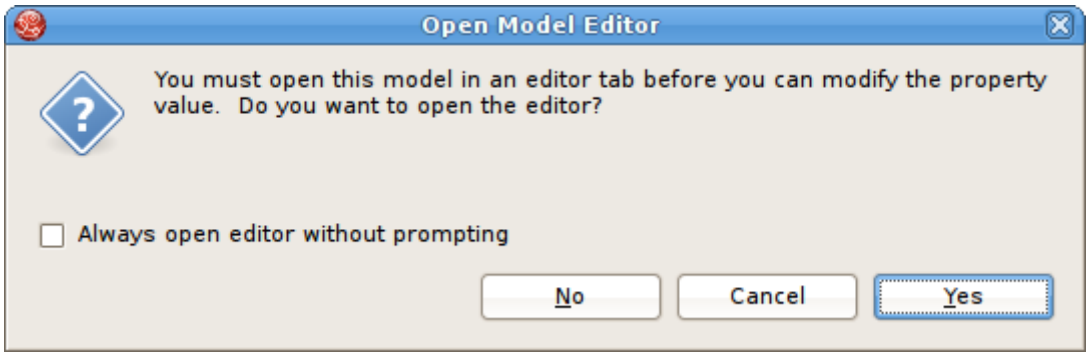


Figure D.22. Open Model Editor Dialog

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

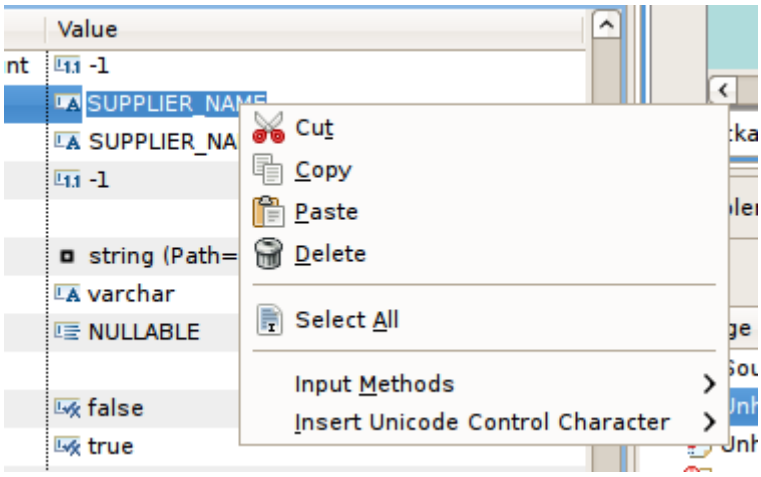


Figure D.23. Open Model Editor Dialog

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).

D.2.6. 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 **Window > Show View >**

Other... to display the ***Figure D.6, "Eclipse Show View Dialog"*** dialog. Choose **Teiid Designer > Description** view and hit **OK** .

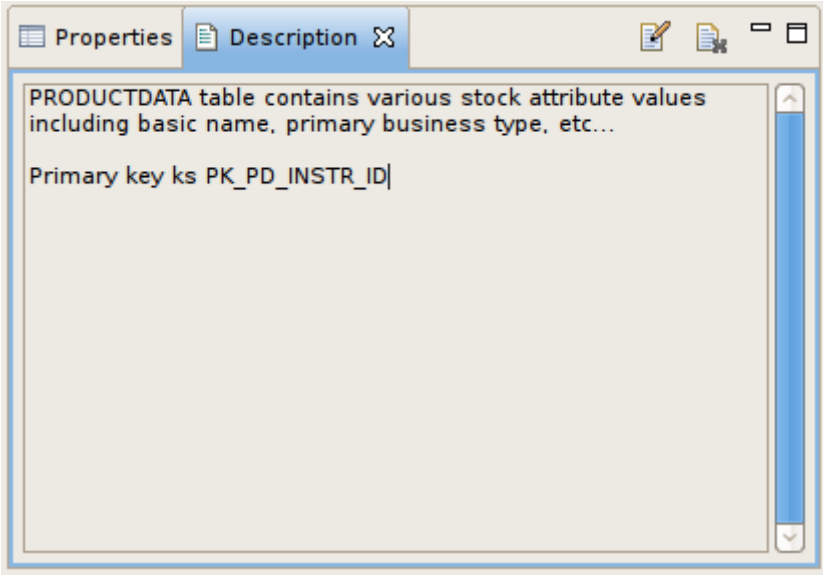


Figure D.24. Description View

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.

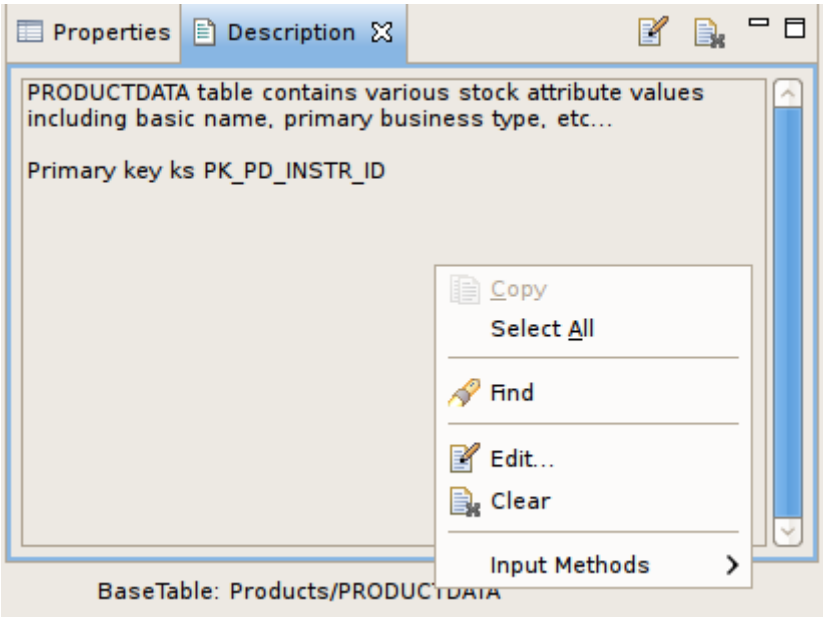


Figure D.25. Description View Context Menu

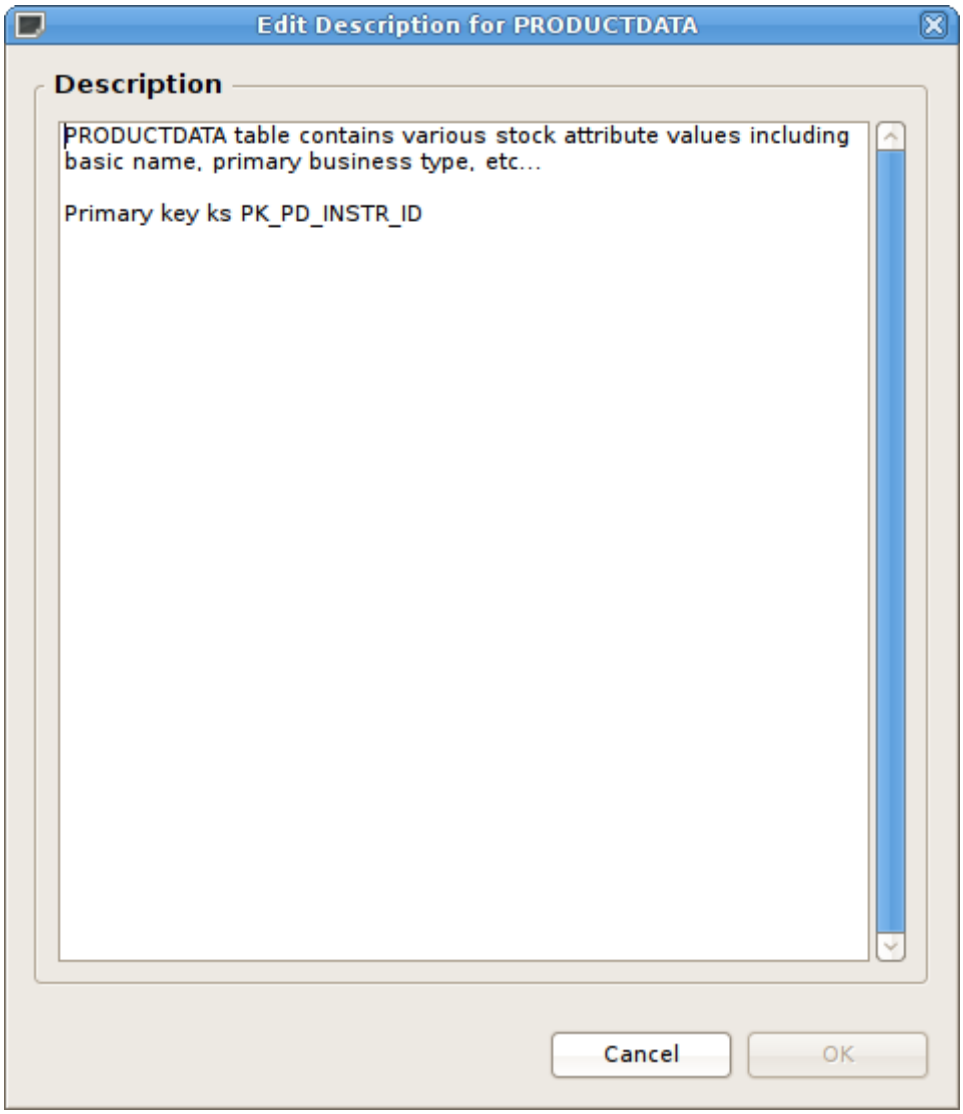


Figure D.26. Edit Description Dialog

D.2.7. Problems View

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

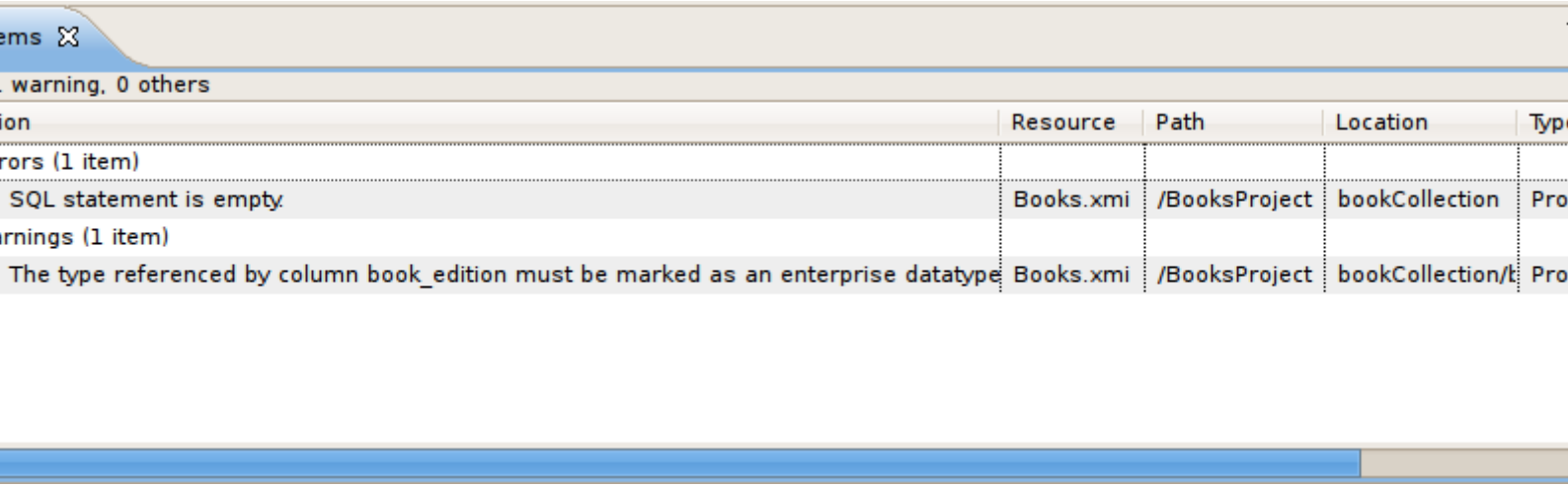


Figure D.27. Problems View

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

D.2.7.1. 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.

D.2.7.2. Context Menu

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

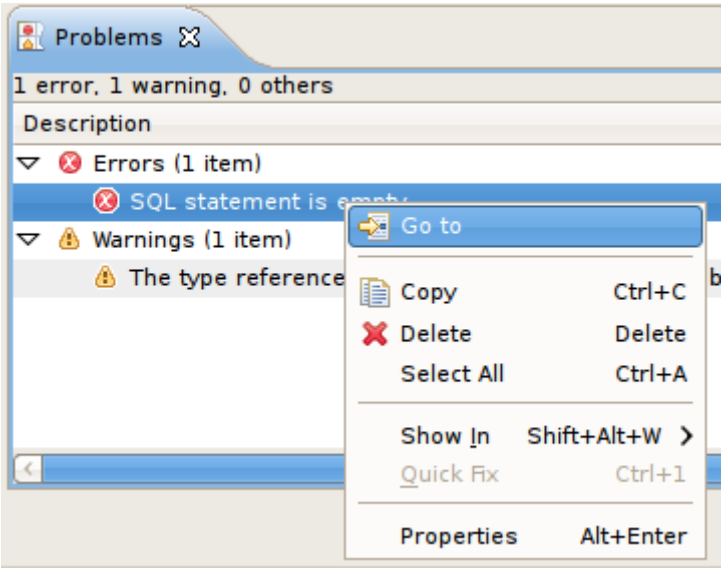


Figure D.28. Problems View Context Menu

- **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 curor 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.

D.2.8. 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.

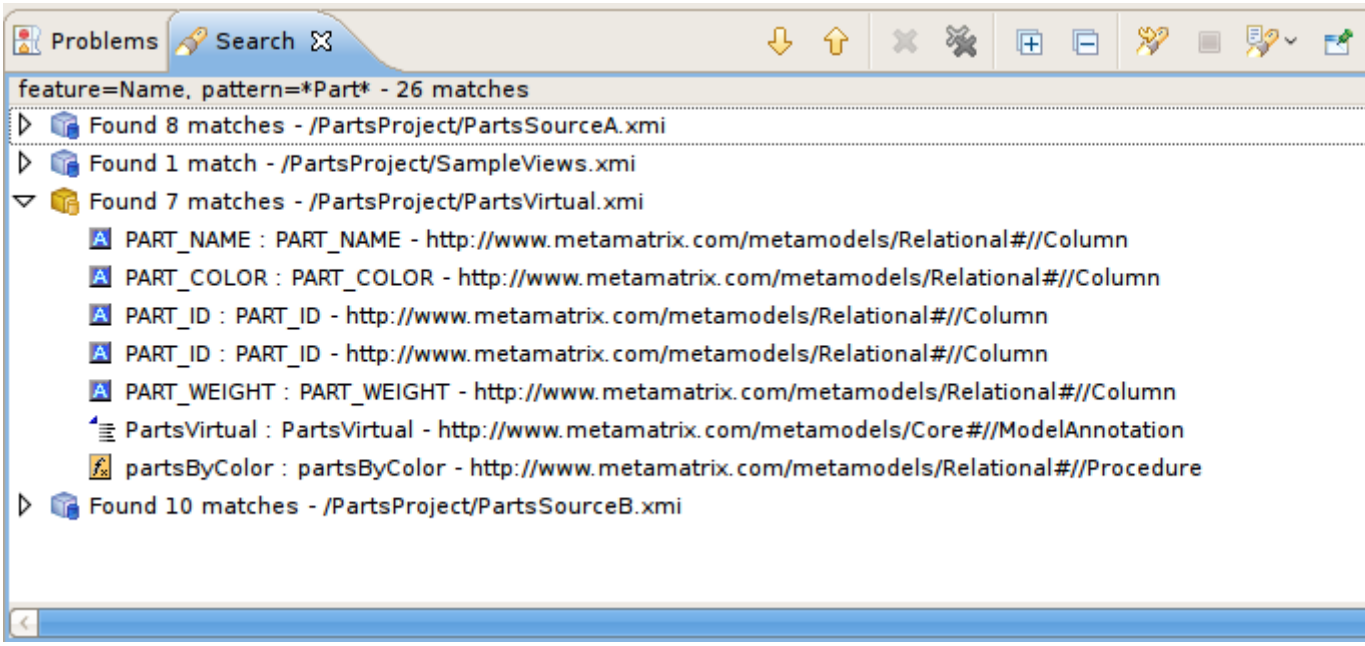


Figure D.29. Search Results View

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:

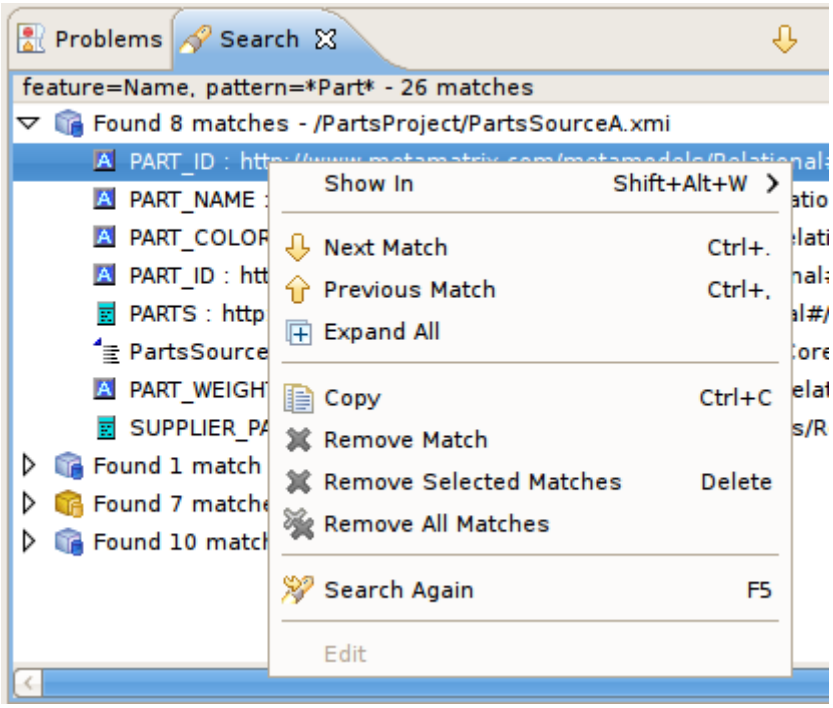


Figure D.30. Search Results Context Menu

D.2.9. Datatype Hierarchy View

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

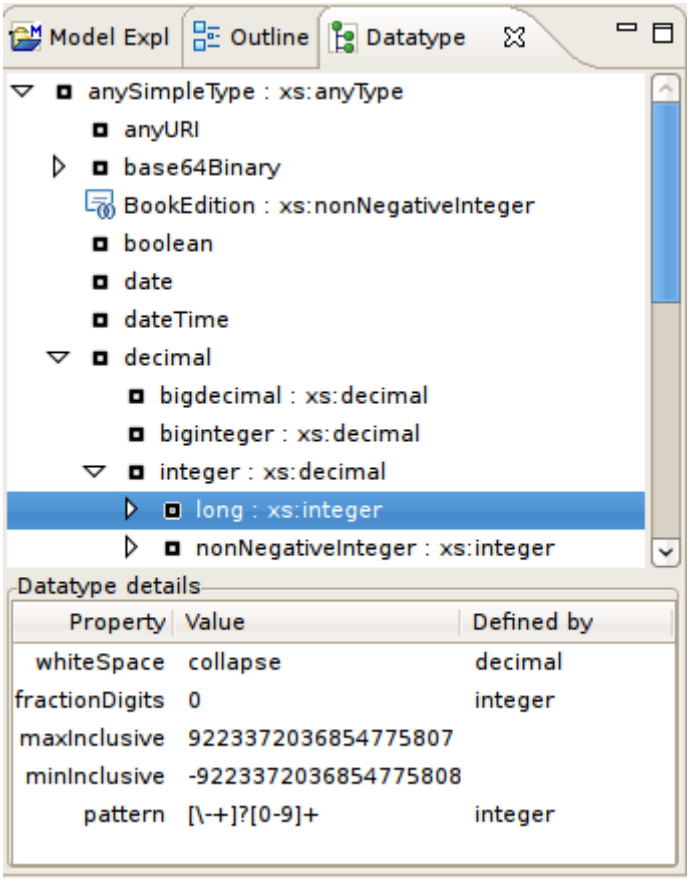


Figure D.31. Datatype Hierarchy View

D.2.10. Teiid Model Classes View

The Model Classes View provides a hiearchical 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.

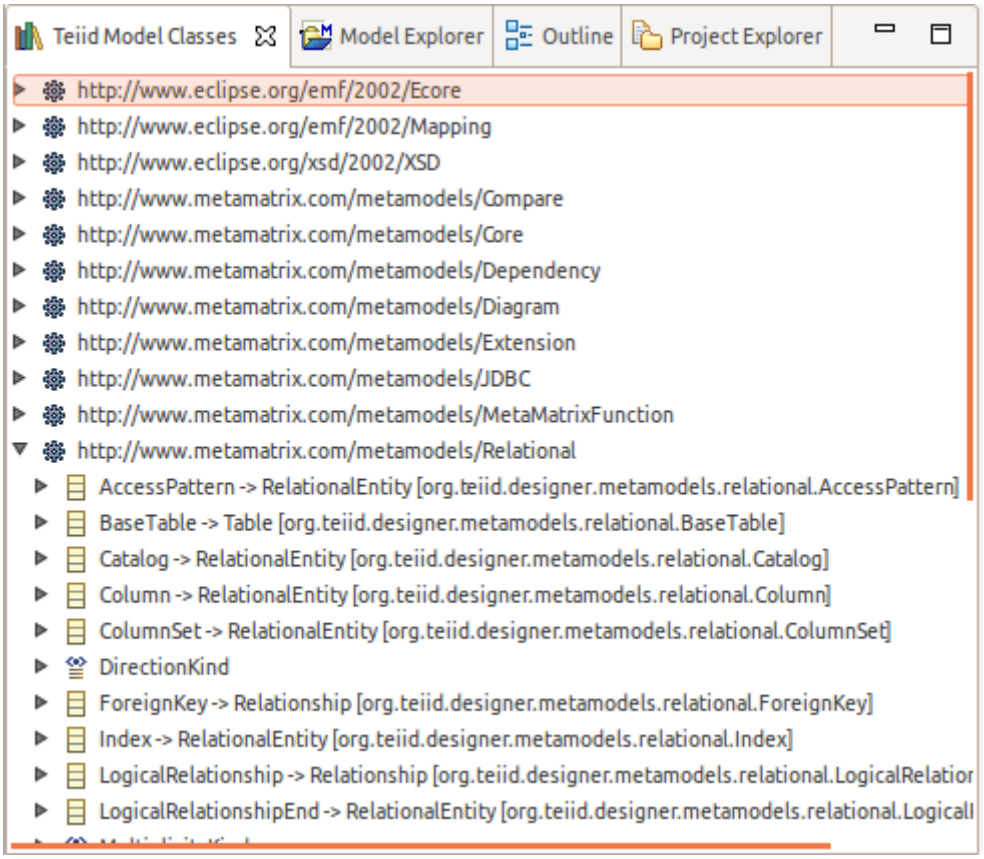


Figure D.32. Datatype Hierarchy View

D.2.11. System Catalog View

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

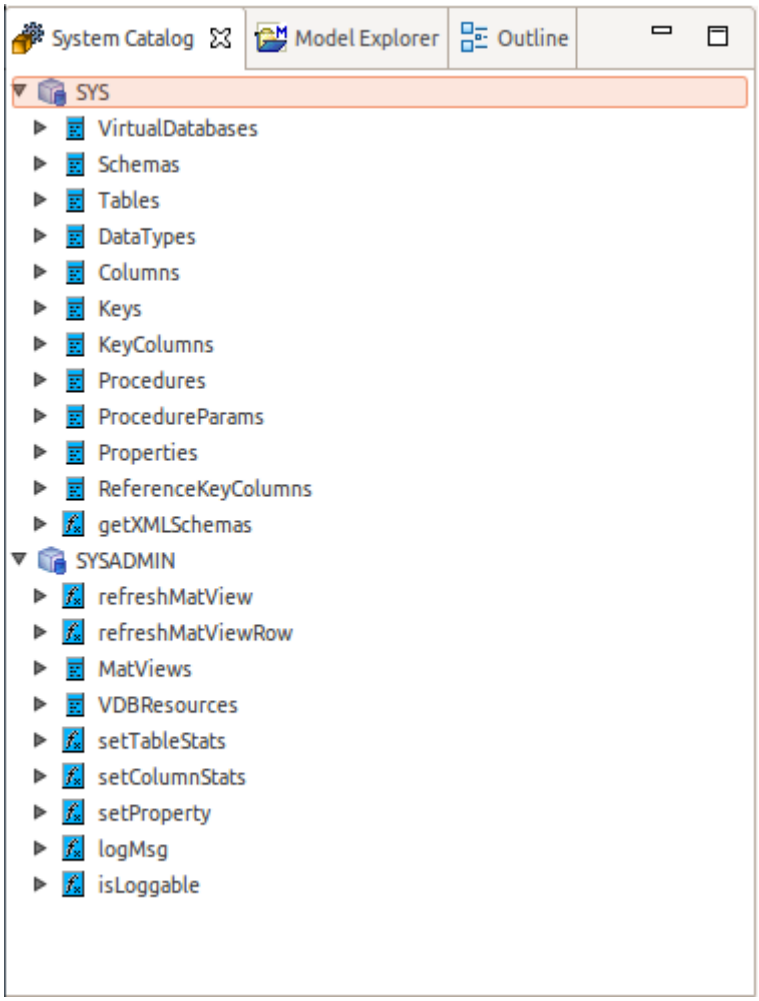


Figure D.33. System Catalog View

D.2.12. SQL Reserved Words View

To open **Teiid Designer's SQL Reserved Words** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > SQL Reserved Words** view in the dialog.

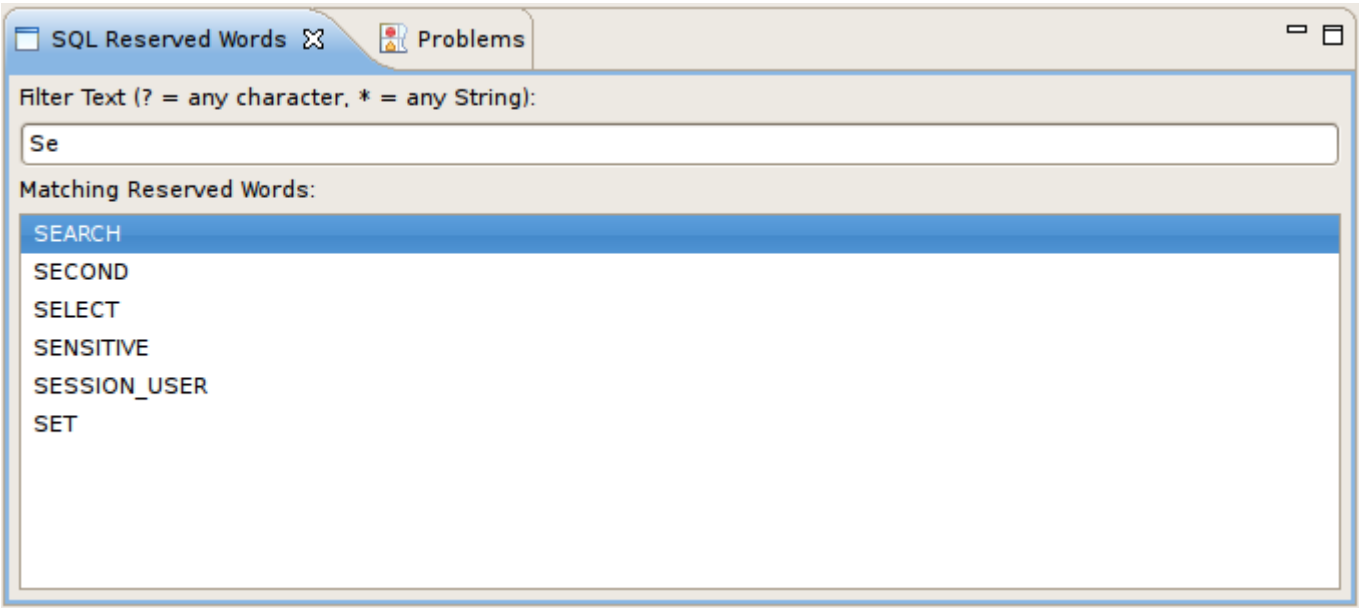


Figure D.34. SQL Reserved Words View

You can also display the view by selecting the the main menu's **Metadata > Show SQL Reserved Words** action as shown below.

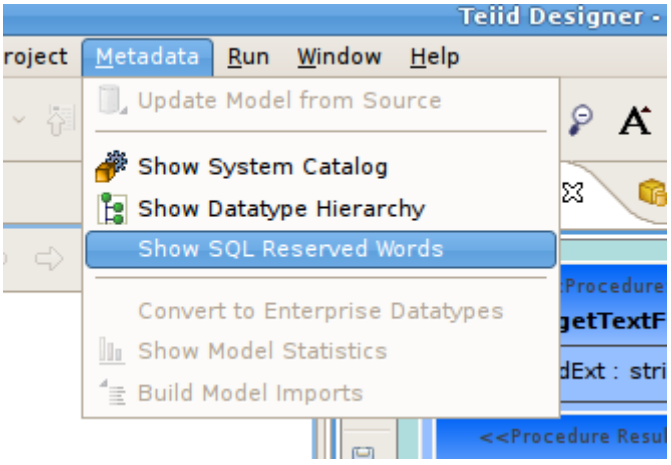
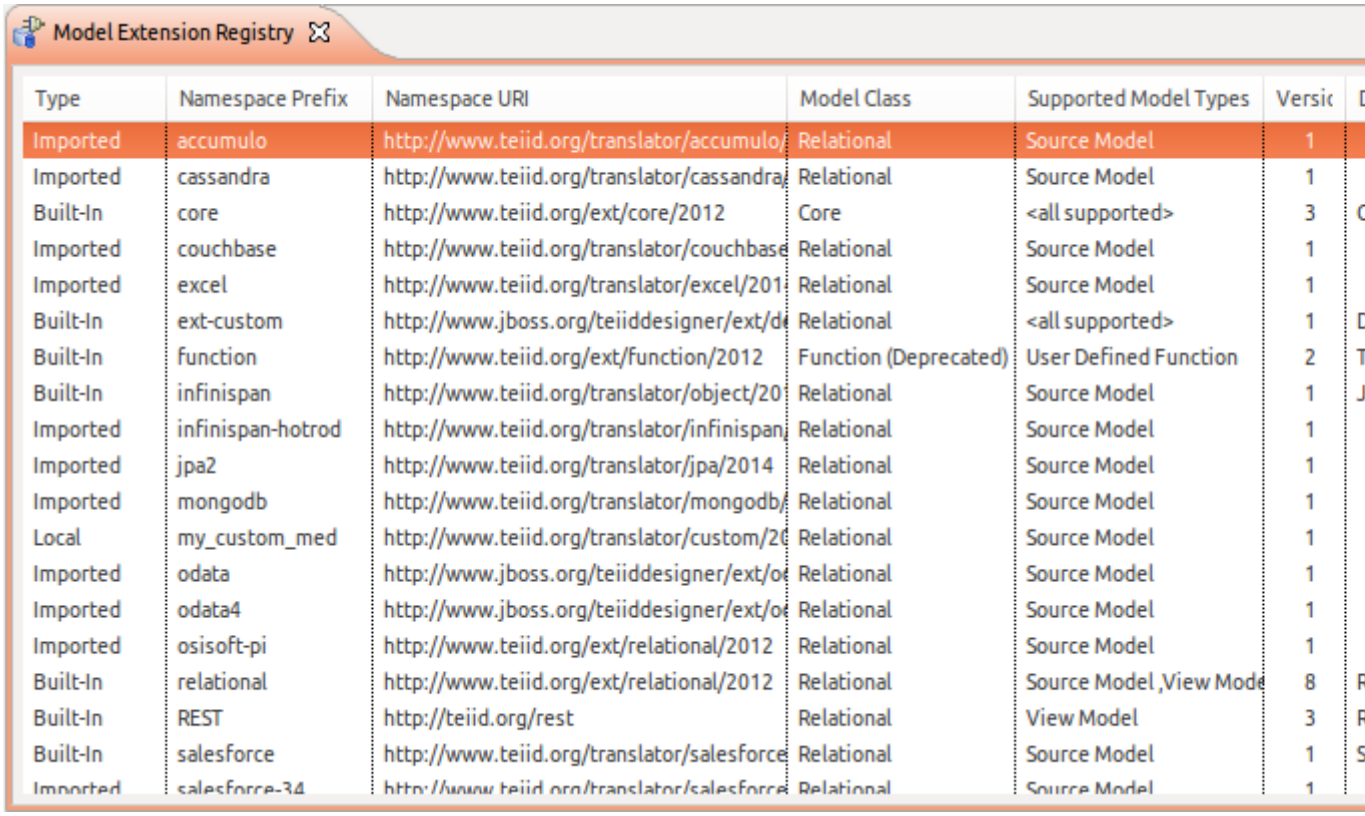


Figure D.35. SQL Reserved Words Action

D.2.13. Model Extension Definition Registry View (MED Registry View)

To open **Teiid Designer's MED Registry** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > Model Extension Registry** view in the dialog. You can also open the MED Registry view from the MED Editor - by selecting the MED Editor toolbar action in the right corner of the MED Editor header section.

The **Model Extension Registry** view shows the currently registered MEDs. Registered MEDs can be applied to models in the workspace (see *Section 6.4, "Managing Model Object Extensions"*). The Model Extension Registry view looks like this:



Type	Namespace Prefix	Namespace URI	Model Class	Supported Model Types	Versic
Imported	accumulo	http://www.teiid.org/translator/accumulo/	Relational	Source Model	1
Imported	cassandra	http://www.teiid.org/translator/cassandra/	Relational	Source Model	1
Built-In	core	http://www.teiid.org/ext/core/2012	Core	<all supported>	3
Imported	couchbase	http://www.teiid.org/translator/couchbase	Relational	Source Model	1
Imported	excel	http://www.teiid.org/translator/excel/201	Relational	Source Model	1
Built-In	ext-custom	http://www.jboss.org/teiiddesigner/ext/de	Relational	<all supported>	1
Built-In	function	http://www.teiid.org/ext/function/2012	Function (Deprecated)	User Defined Function	2
Built-In	infinispan	http://www.teiid.org/translator/object/20	Relational	Source Model	1
Imported	infinispan-hotrod	http://www.teiid.org/translator/infinispan/	Relational	Source Model	1
Imported	jpa2	http://www.teiid.org/translator/jpa/2014	Relational	Source Model	1
Imported	mongodb	http://www.teiid.org/translator/mongodb/	Relational	Source Model	1
Local	my_custom_med	http://www.teiid.org/translator/custom/2(	Relational	Source Model	1
Imported	odata	http://www.jboss.org/teiiddesigner/ext/od	Relational	Source Model	1
Imported	odata4	http://www.jboss.org/teiiddesigner/ext/od	Relational	Source Model	1
Imported	osisoft-pi	http://www.teiid.org/ext/relational/2012	Relational	Source Model	1
Built-In	relational	http://www.teiid.org/ext/relational/2012	Relational	Source Model ,View Mode	8
Built-In	REST	http://teiid.org/rest	Relational	View Model	3
Built-In	salesforce	http://www.teiid.org/translator/salesforce	Relational	Source Model	1
Imported	salesforce-34	http://www.teiid.org/translator/salesforce	Relational	Source Model	1

Figure D.36. MED Registry View

For each registered MED, the namespace prefix, namespace URI, extended model class, version, and description is shown. In addition, a flag indicating if the MED is built-in is shown. The Model Extension Registry view has toolbar actions that register a workspace MED file, unregister a user-defined MED, copy a registered MED to the workspace, or view the MED. All of these actions are also available via a context menu.

A MED registry keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. There are 2 different types of MEDs stored in the registry:

- **Built-In MED** - these are registered during Designer installation. These MEDs cannot be updated or unregistered by the user.
- **User-Defined MED** - these are created by the user. These MEDs can be updated, registered, and unregistered by the user.

Note

When a workspace MED is registered it can be deleted from the workspace if desired. The registry keeps its own copy. And a registered MED can always be

copied back to the workspace by using the appropriate toolbar or context menu action.

D.2.14. Guides View

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

The **Guides** view provides assistance for many common modeling tasks. The view includes categorized Modeling Actions and also links to 'Cheat Sheets' for common processes. 'Cheat Sheets' are an eclipse concept for which Teiid Designer has provided contributions (see *Section D.2.16, "Cheat Sheets View"*). The Guides view is shown below:

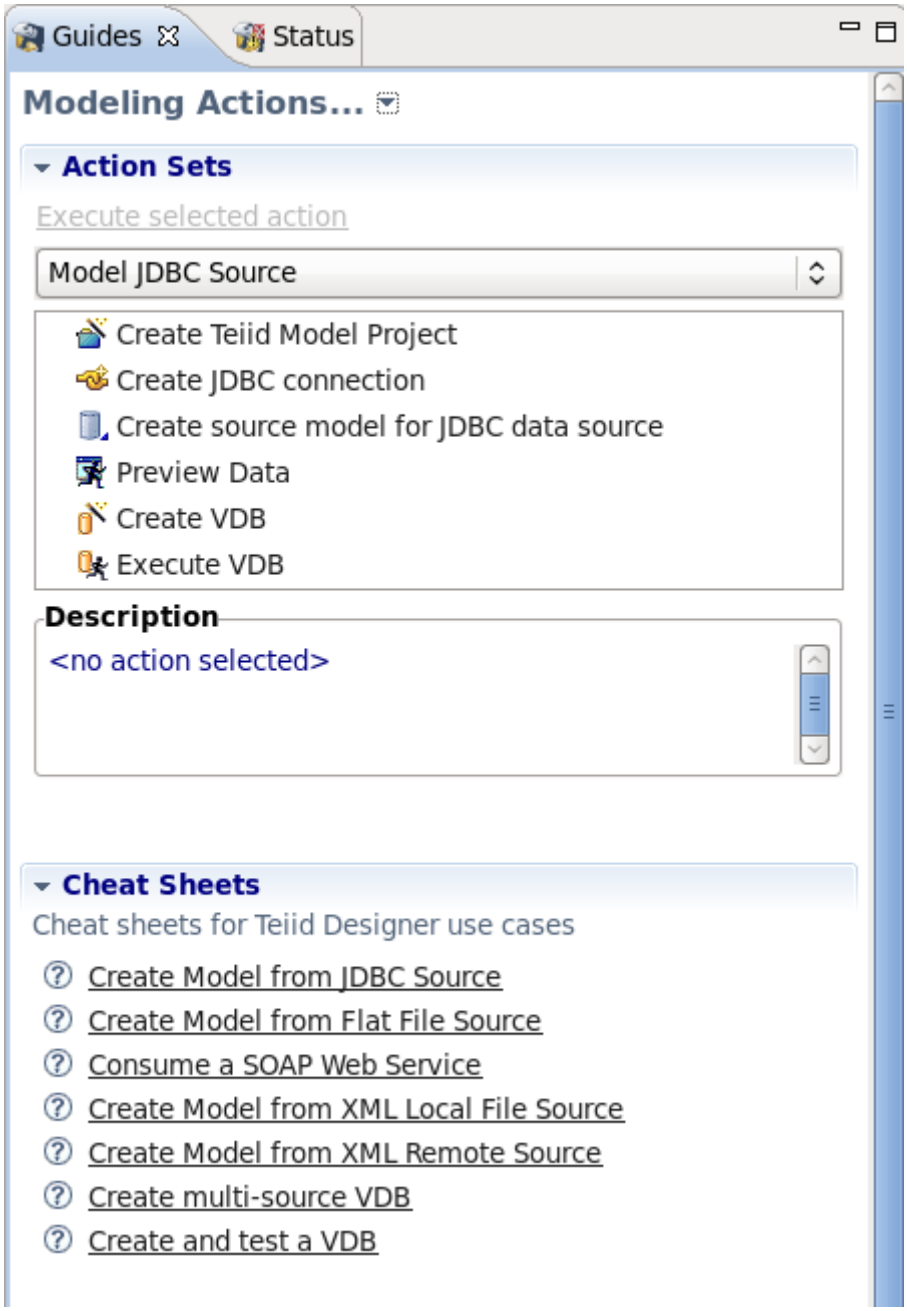


Figure D.37. Guides View

The upper 'Action Sets' section provides categorized sets of actions. Select the desired category in the dropdown, then the related actions for the selected category are displayed in the list below it. Execute an action by clicking the 'Execute selected action' link or double-clicking on the action.

The lower 'Cheat Sheets' section provides a list of available 'Cheat Sheet' links, which will launch the appropriate Cheat Sheet to guide you step-by-step through the selected process.

D.2.15. Status View

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

The **Status** view provides a quick overview status of the selected project. A sample Status view for a project is shown below:

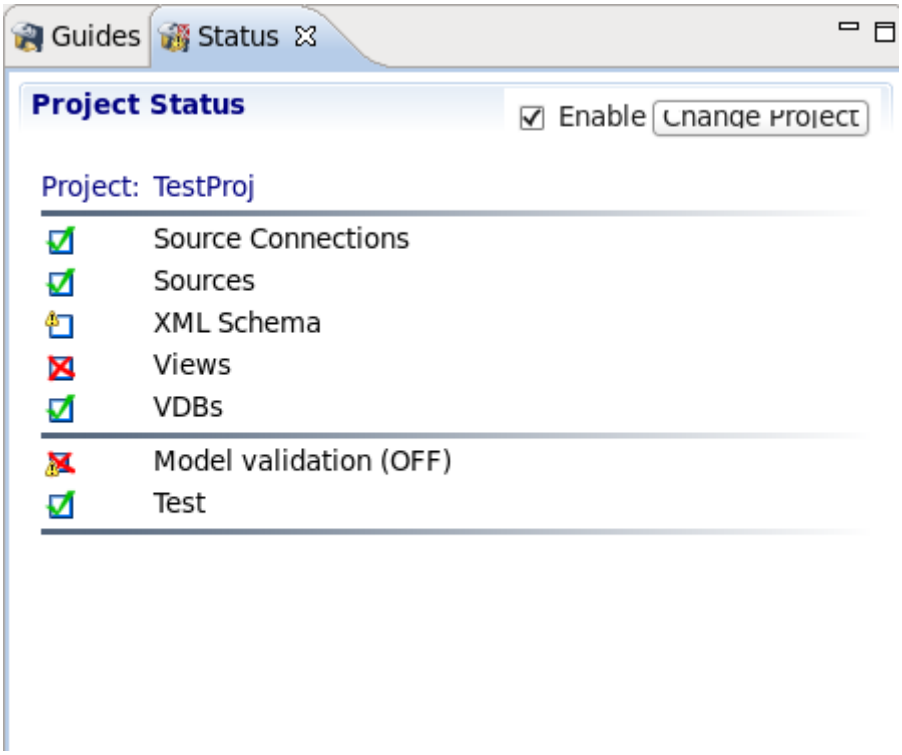


Figure D.38. Status View

The status view is broken down into common project areas:

- **Source Connections** - all Source Connections are fully defined.
- **Sources** - Source Models exist.
- **XML Schema** - XML Schemas exist.
- **Views** - View Models exist.
- **VDBs** - VDBs exist and are deployable.
- **Model Validation (Status)** - all Models pass validation.
- **Test** - all defined VDBs pass validation.

The status of each area is denoted by an icon: A green check indicates OK, a red 'x' indicates errors and a 'warning' icon indicates potential problems. The project can be changed by selecting the 'Change Project' button.

D.2.16. Cheat Sheets View

To open **Cheat Sheets** view, select the main menu's **Window > Show View > Other...** and select the **Help > Cheat Sheets** view in the dialog.

The **Cheat Sheets** view is a standard Eclipse Help concept. Cheat Sheets provide step-by-step assistance for common process workflows. Teiid Designer has contributed to the Eclipse help framework to provide assistance for many common modeling tasks. The Guides View (see *Section D.2.14, "Guides View"*) provides links to these Cheat Sheets, as previously described. A sample Cheat Sheet is shown below:

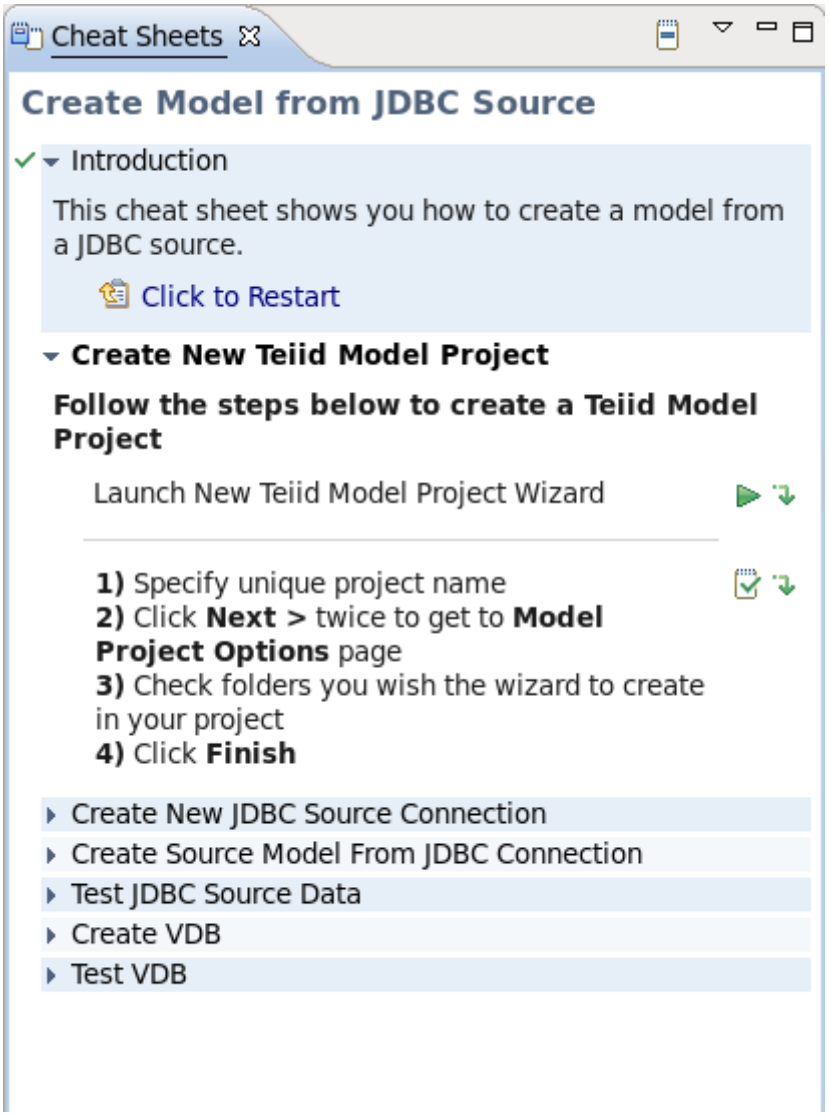


Figure D.39. Cheat Sheet Sample

D.3. 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.

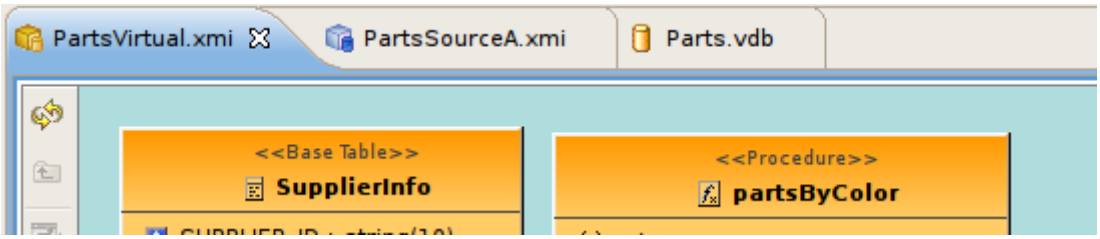


Figure D.40. Editor Tabs

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.

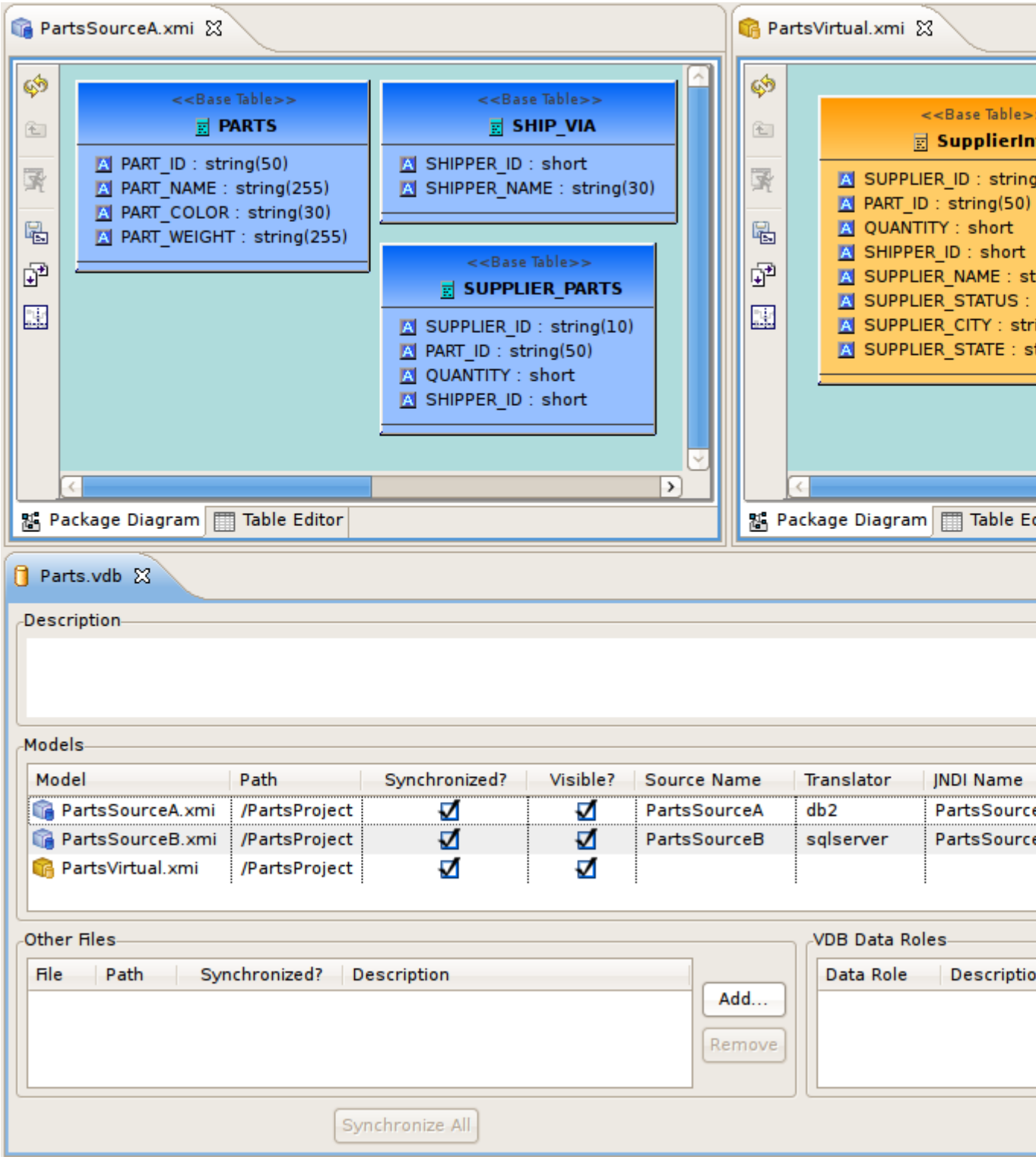


Figure D.41. Viewing Multiple Editors

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.

D.3.1. Model Editor

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

D.3.1.1. Model Editor

The Model Editor includes a detailed summary of the contents of a model and provides a centralized panel for editing the model contents and properties.

The editor page provides for editing the model contents via:

- vertical toolbar containing the following operations
  -

    Preview Data (i.e. Test)

-

  New Table

-

  New View

-

  New Procedure or Function

-

  New View Procedure or Function

-

  New Index

-

  Generate Data Service - Generate and export a deployable *-vdb.xml as a simple data service for the selected table or view. (Also available when view or table is selected via the **Modeling.... > Generate Data Service** action)

- Model **Contents** tree view (same information shown in **Model Explorer** tree)
  - Simplified action context menus compared to **Model Explorer** and limited to applicable model object actions.
- **Actions panel**
  - Contains applicable operations you can perform for the reference model
    - Also available in **Model Explorer** tree view via model selection and context menu **Modeling...** action category
- **Properties panel**
  - Provides the same information that can still be edited in Eclipse's **Properties View**
    - Properties are driven based on selection in tree view in this editor
- **Description panel**
  - Provides the same description that can still be edited in Designer's **Description View**
    - Description editing is driven by selection in tree view in this editor

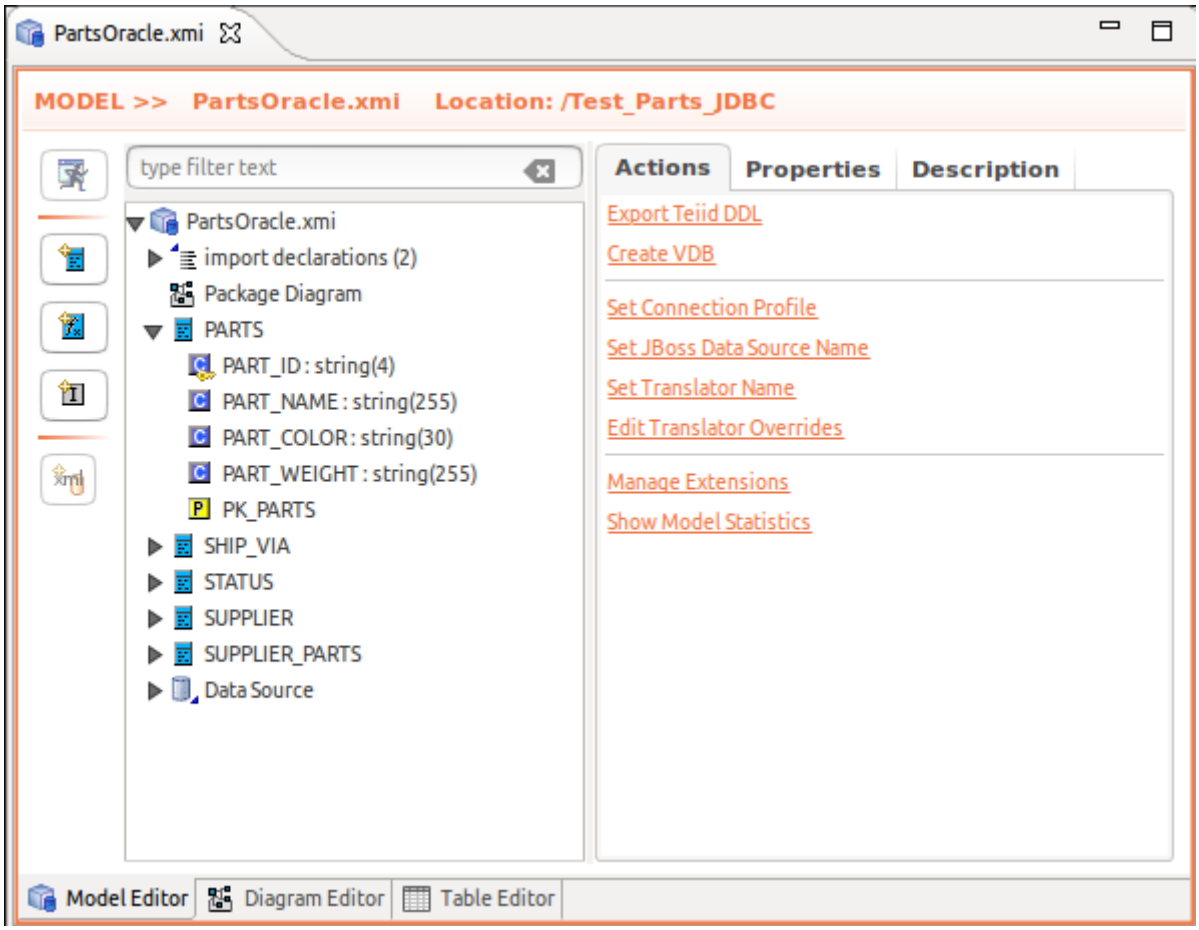


Figure D.42. Model Editor Tab

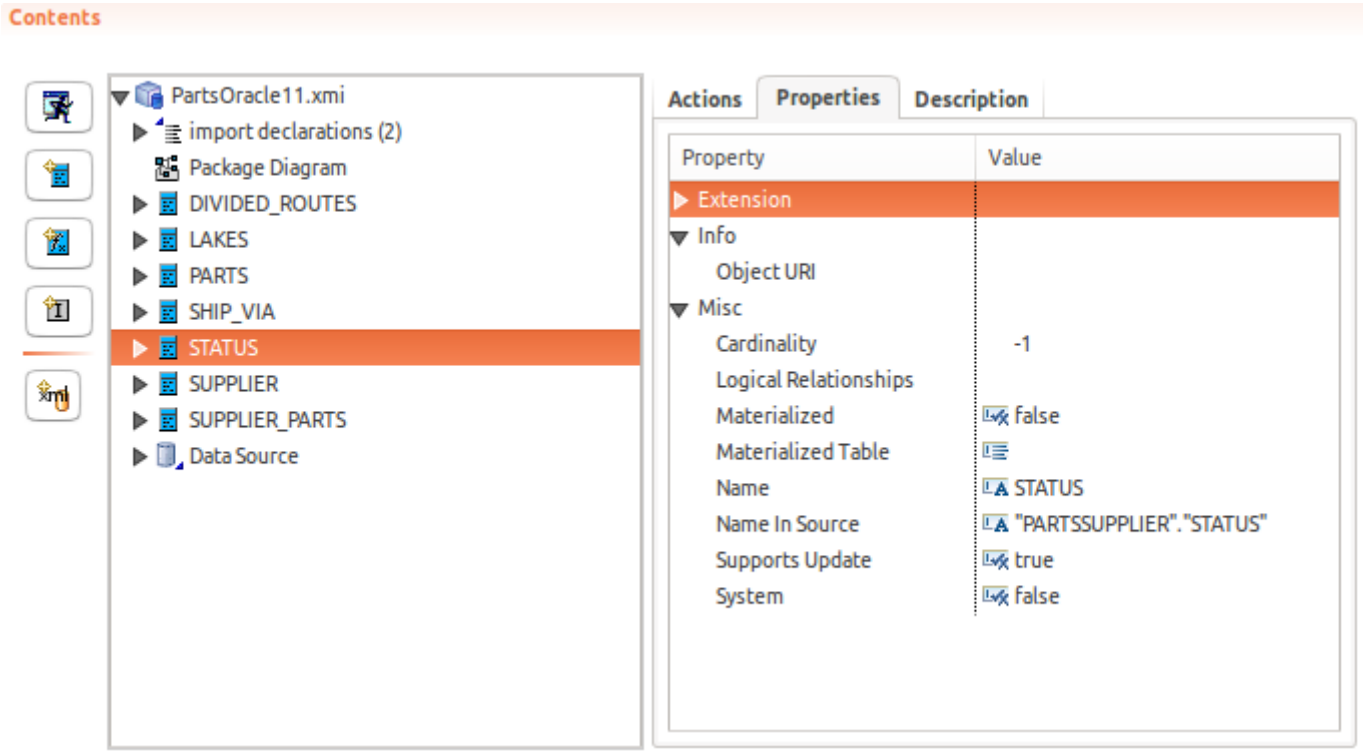


Figure D.43. Model Editor Tab

D.3.1.2. 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 D.44. Main Toolbar Diagram Actions

- The actions include:
  -
    Zoom In
  -

    Zoom to Level
  -
    Zoom Out
  -
    Increase Font Size
  -
    Decrease Font Size
  -
    Perform Diagram Layout

D.3.1.2.1. 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.

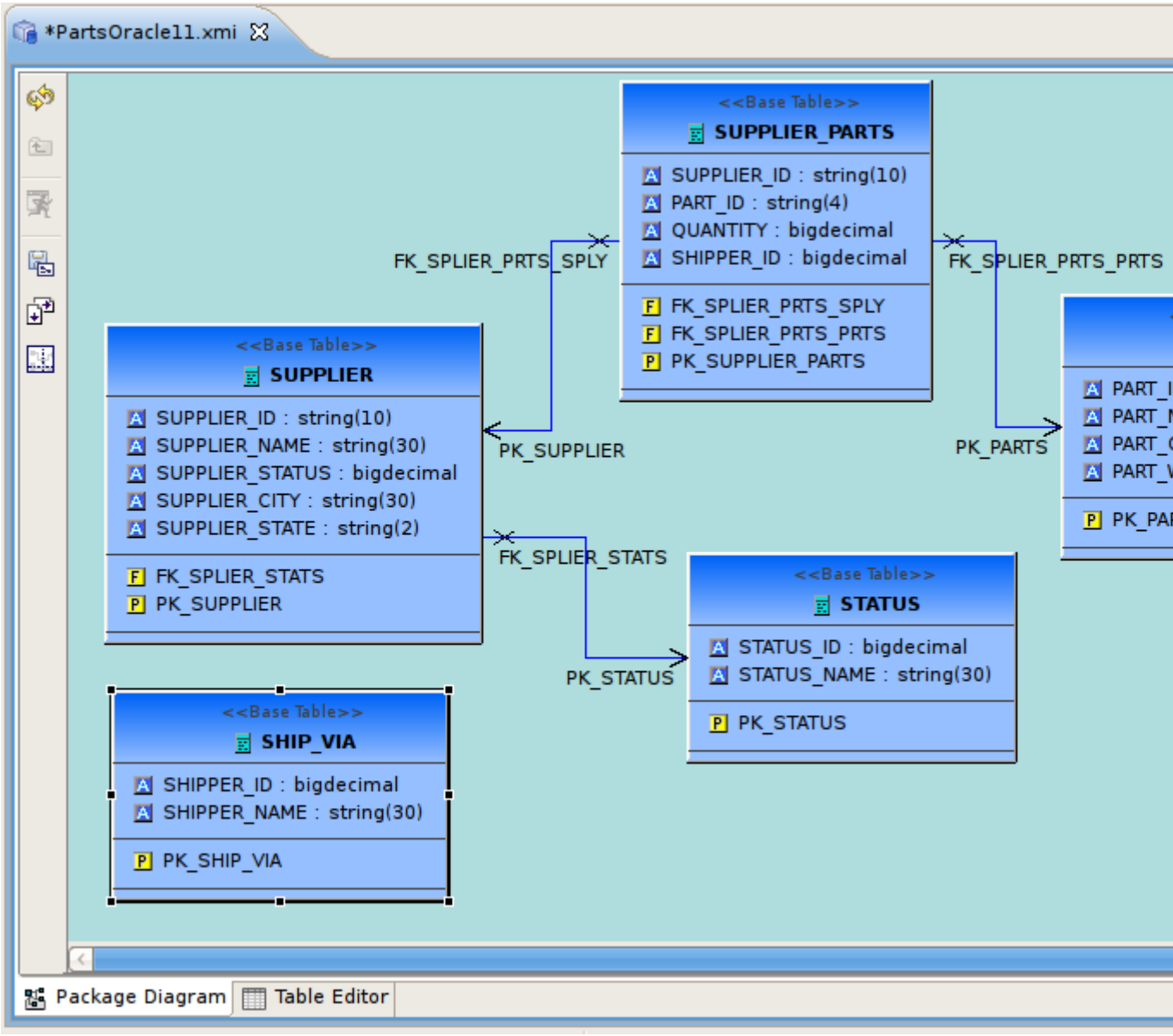


Figure D.45. Package Diagram Example

- 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.

-

  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.

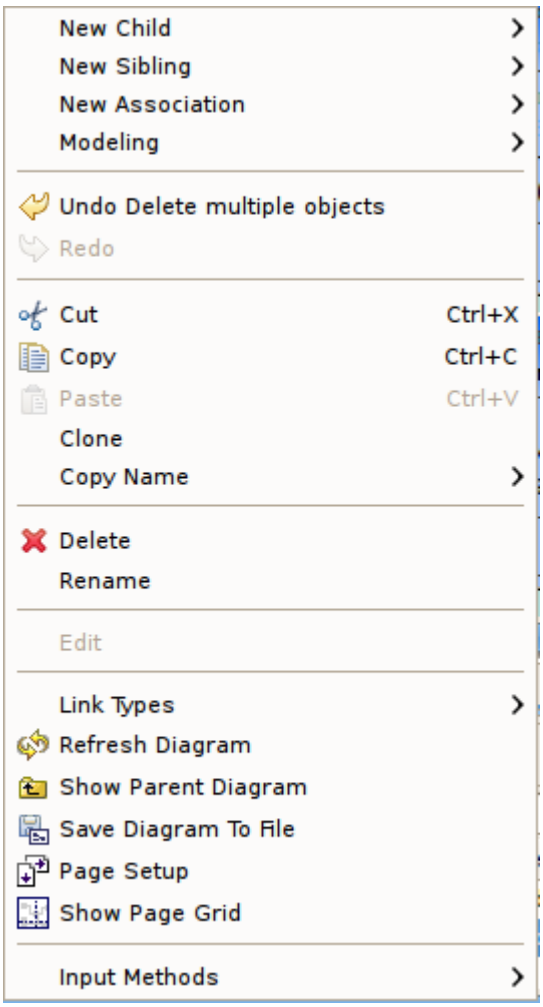


Figure D.46. Package Diagram Context Menu

D.3.1.2.2. 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.

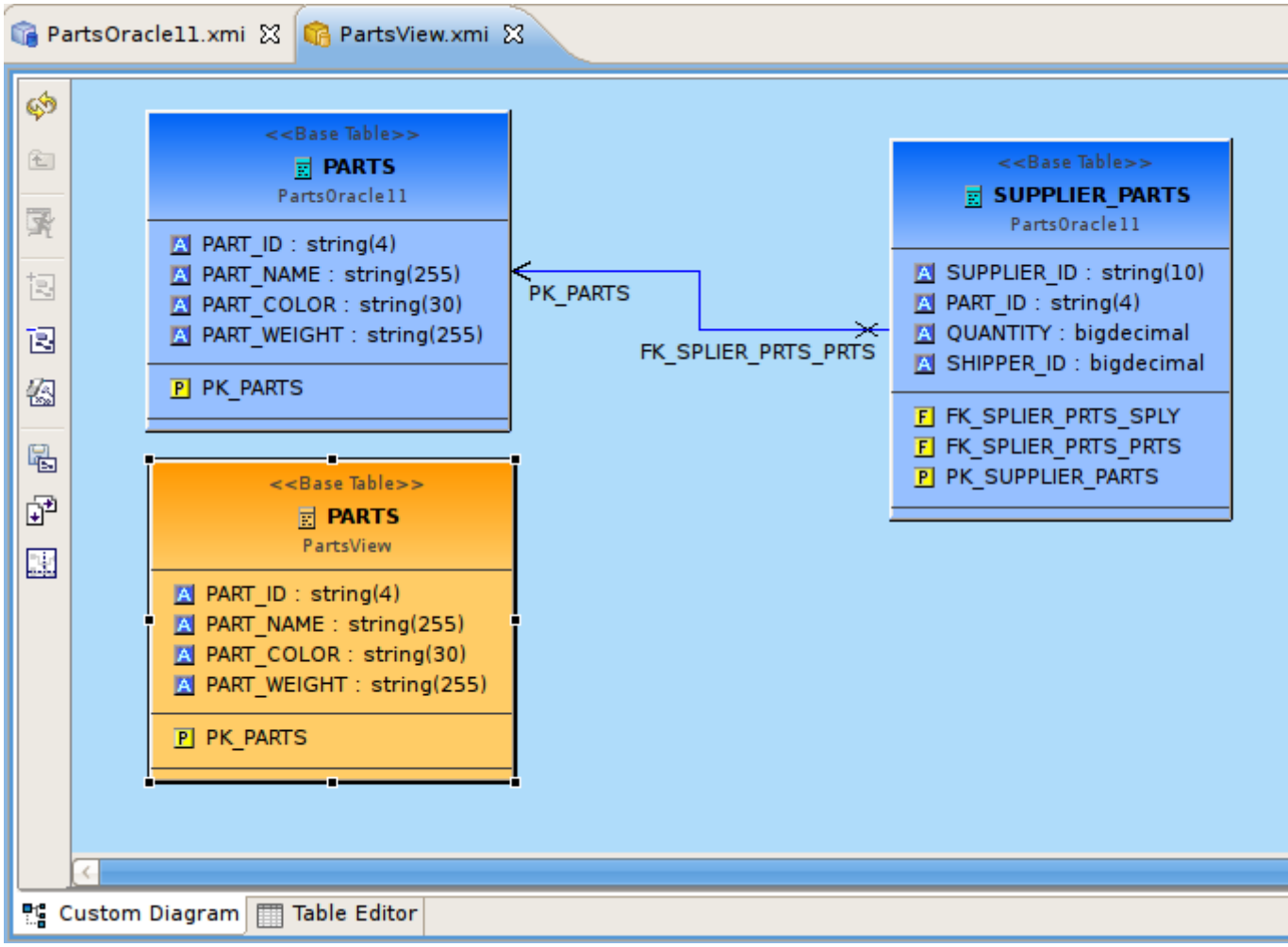


Figure D.47. Package Diagram Example

- 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.

-

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

Since **Custom Diagrams** do not represent represents 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.

D.3.1.2.3. Transformation Diagram

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

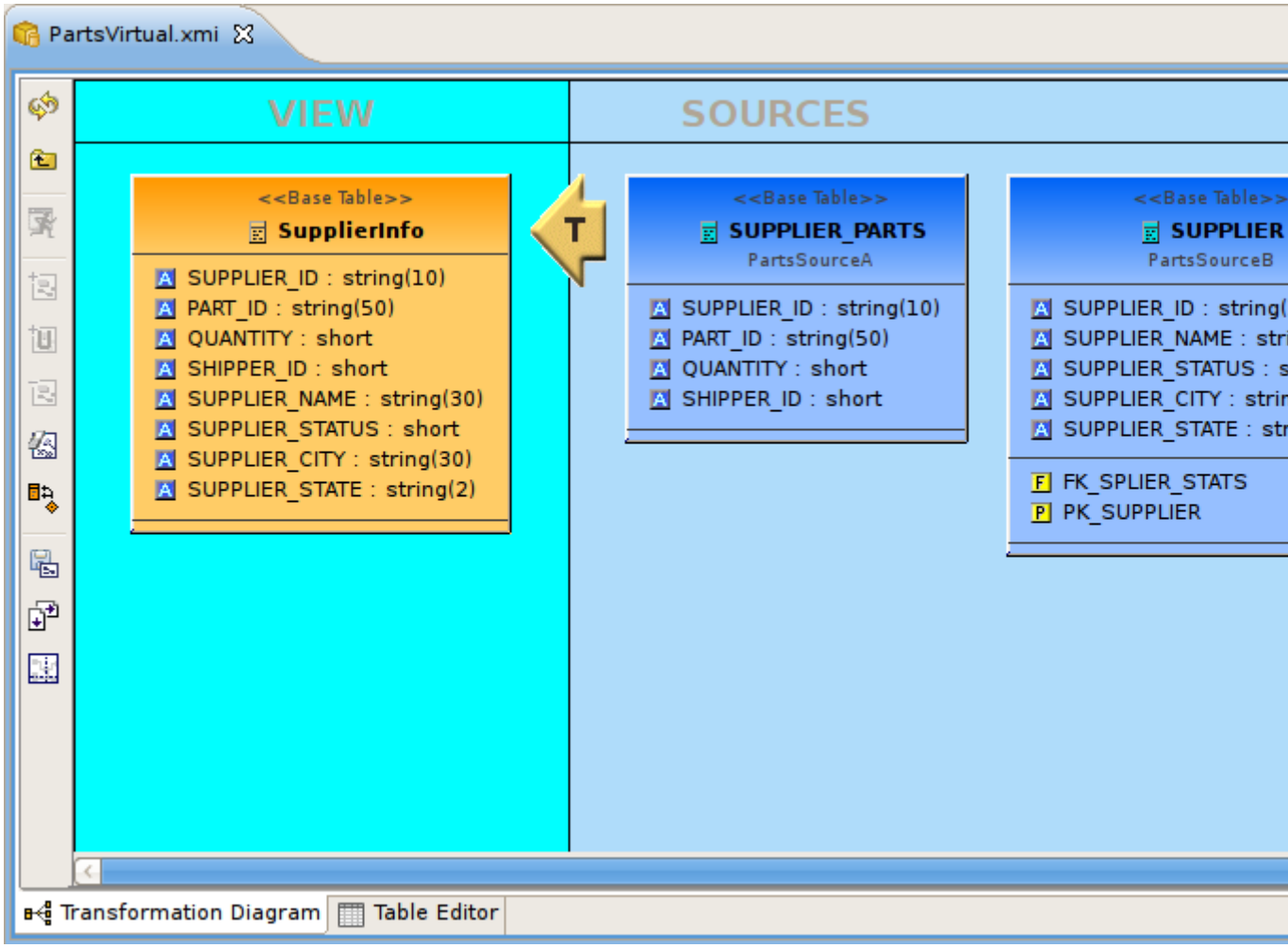


Figure D.48. Transformation Diagram Example

- 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.

-

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

Context menus for the

D.3.1.2.4. 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.

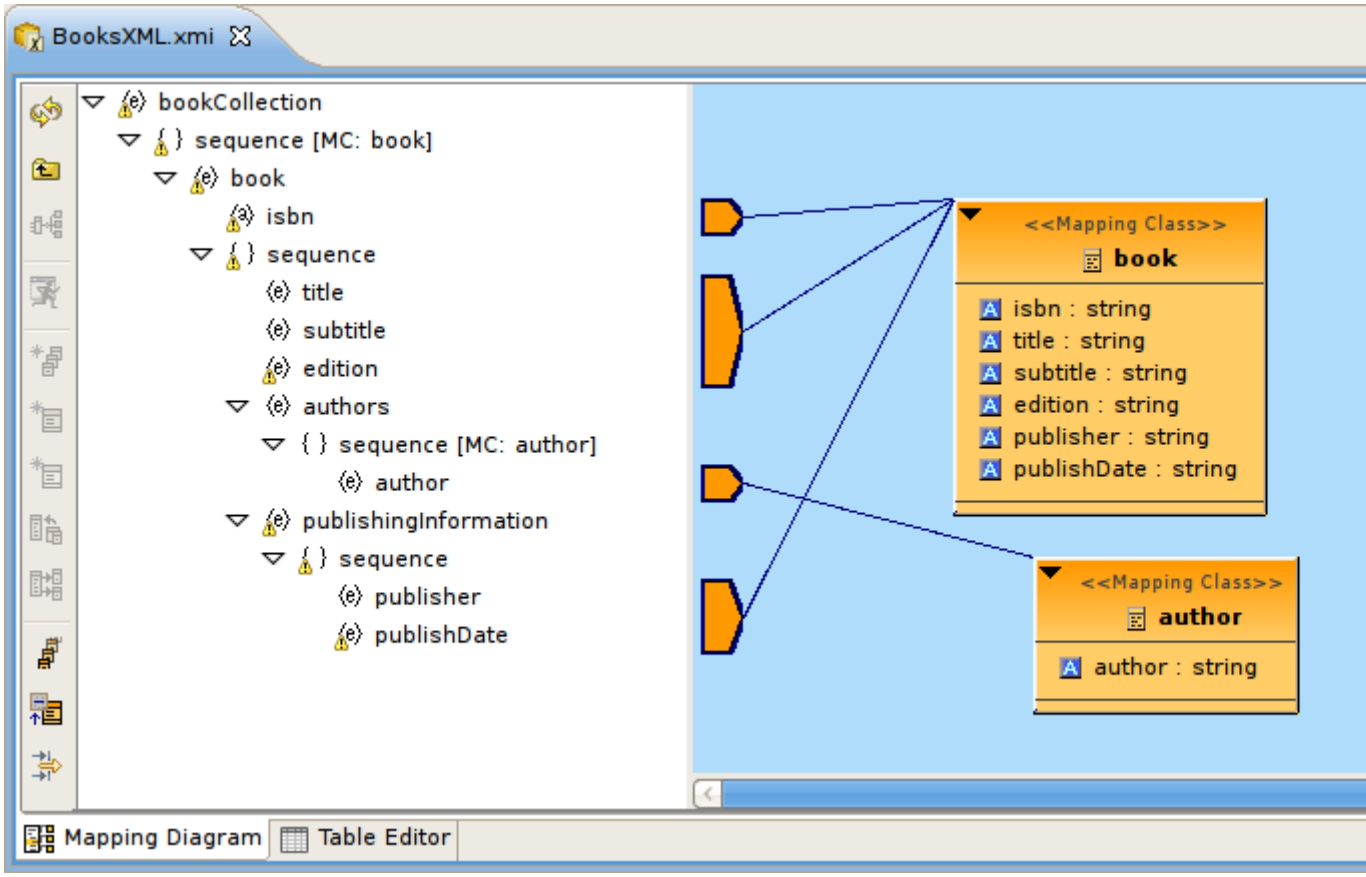


Figure D.49. Mapping Diagram Example

- 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..)

D.3.1.2.5. 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.

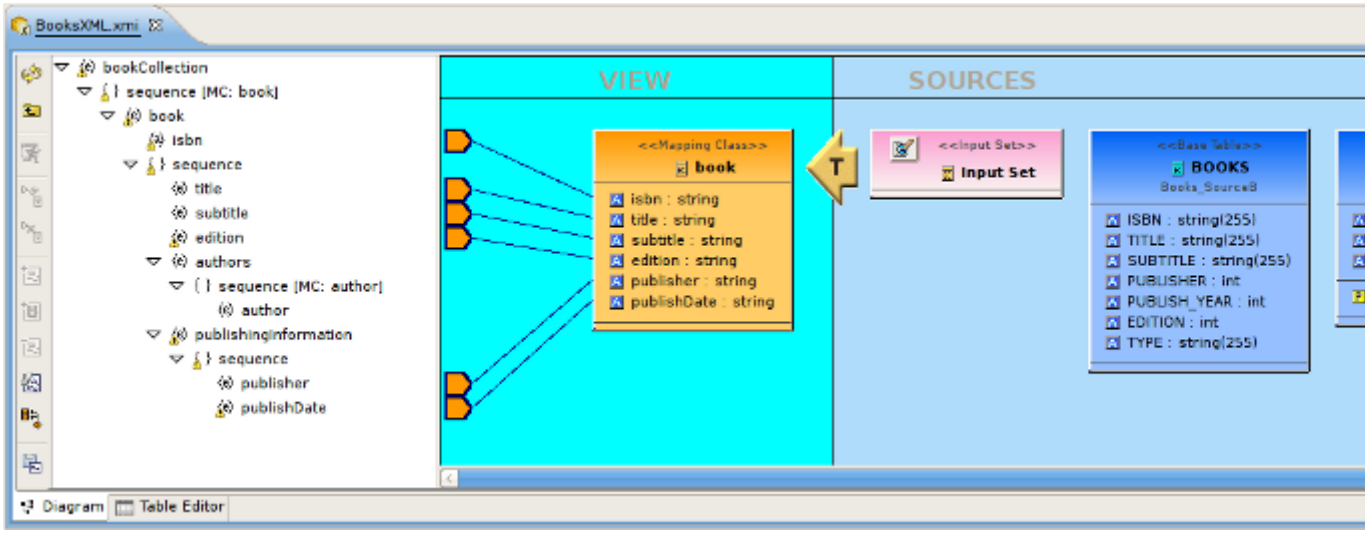


Figure D.50. Mapping Transformation Diagram Example

- 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.

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

D.3.1.3. 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.

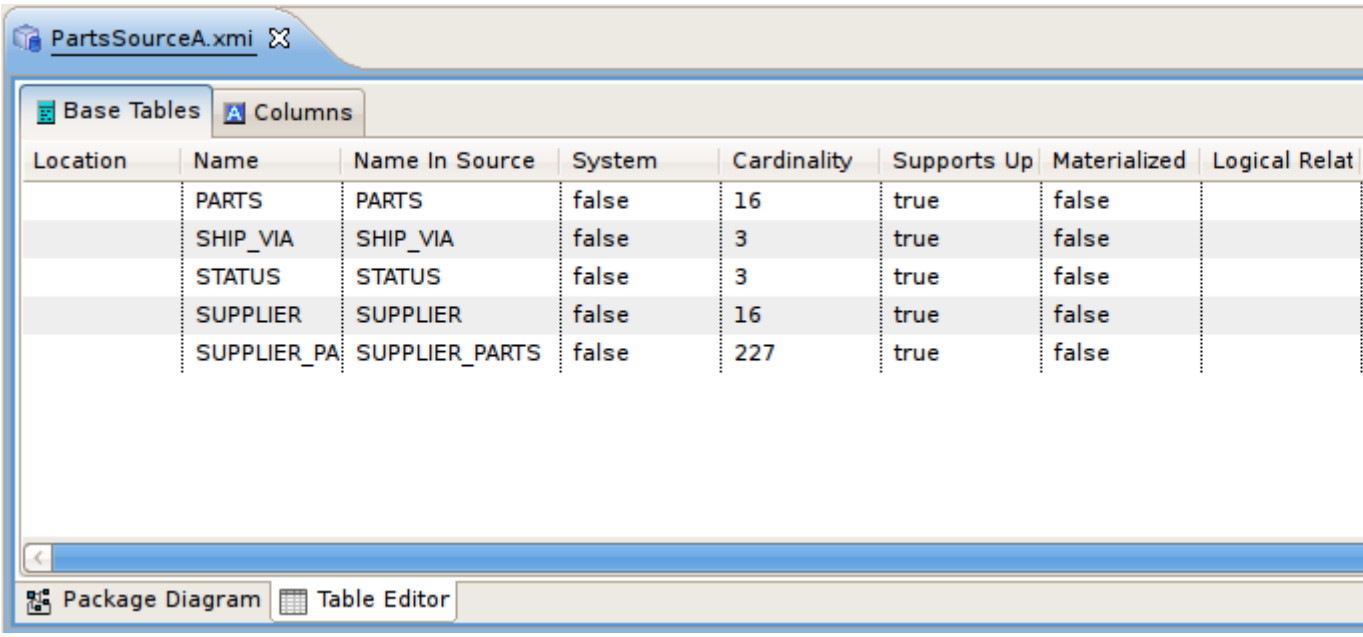


Figure D.51. Table Editor Example

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.

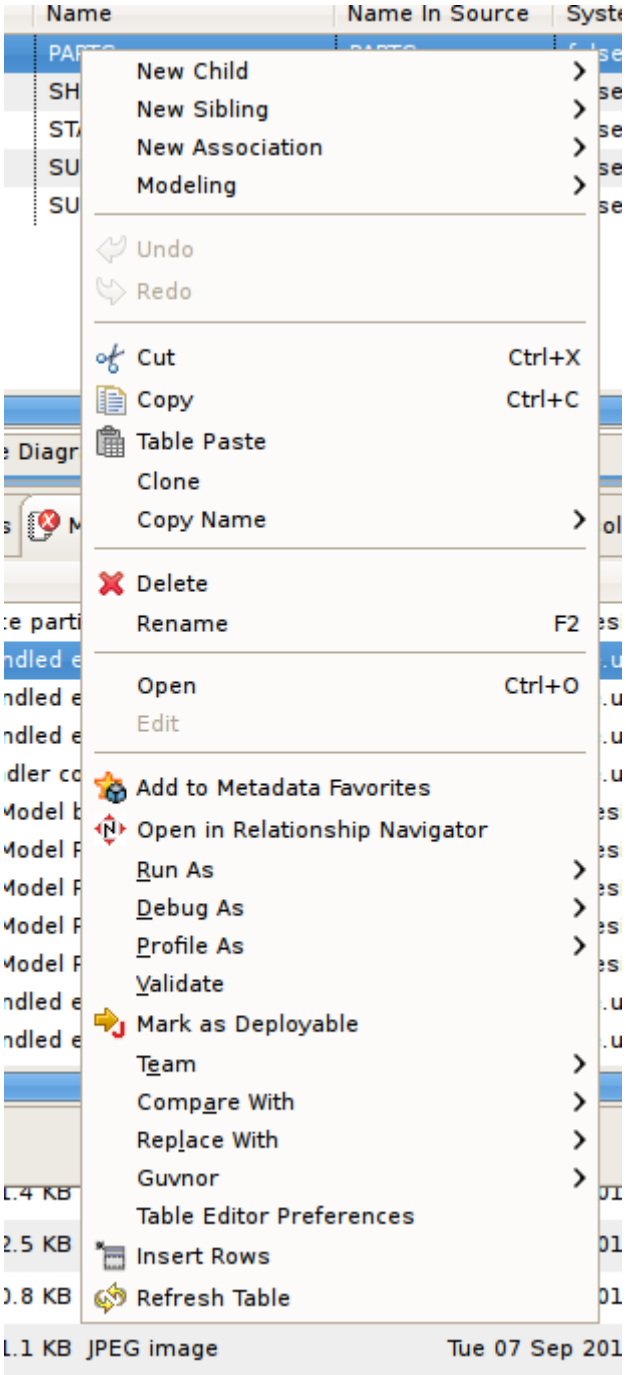


Figure D.52. Table Editor Example

D.3.1.3.1. 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.

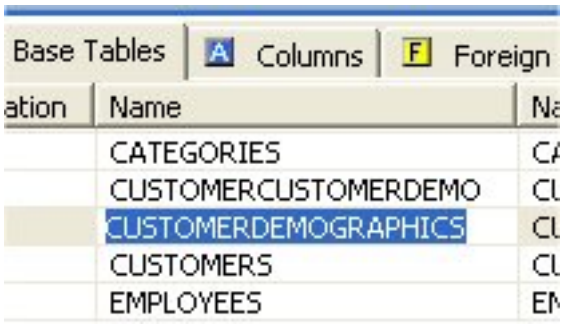


Figure D.53. Editing String Property

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.

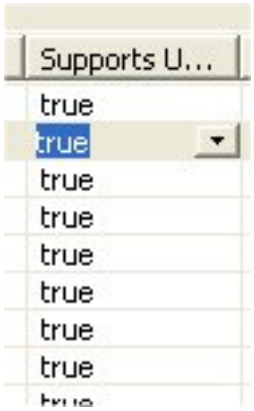


Figure D.54. Editing Boolean Value

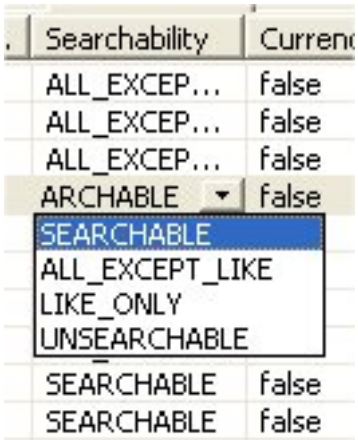


Figure D.55. Editing Multi-Value Property

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.

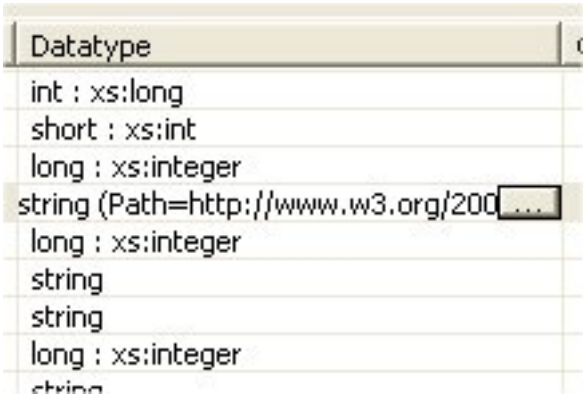


Figure D.56. Editing Multi-Value With Picker

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.

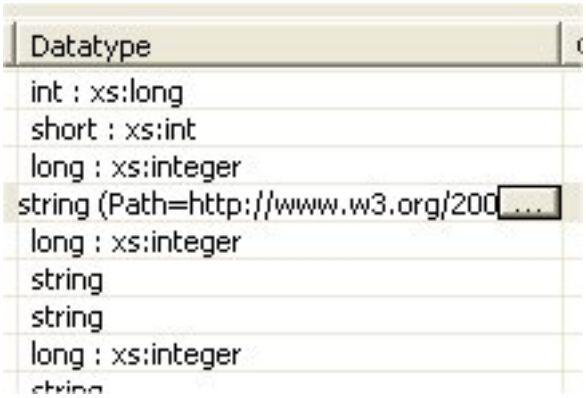


Figure D.57. Editing Datatype Values

D.3.1.3.2. 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.

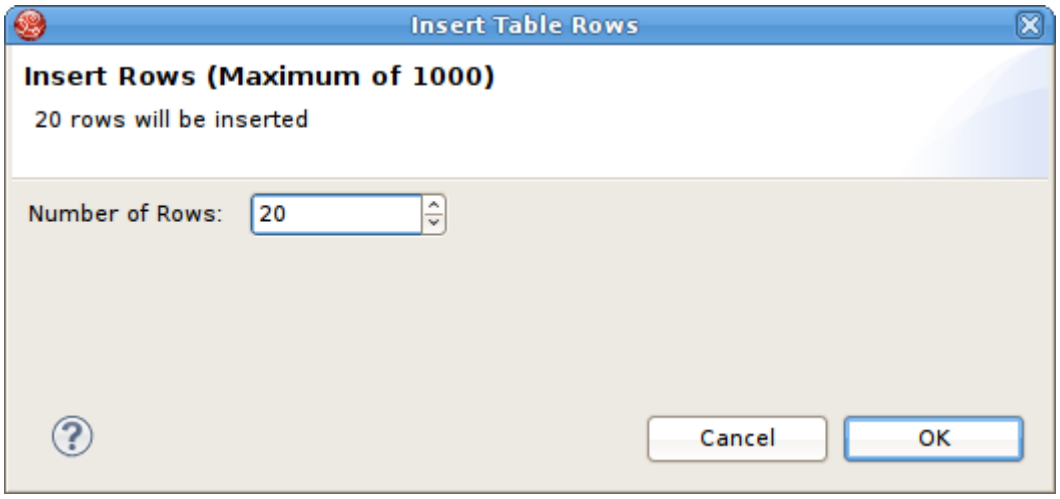


Figure D.58. Editing String Property

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.

D.3.1.4. Simple Datatypes Editor

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

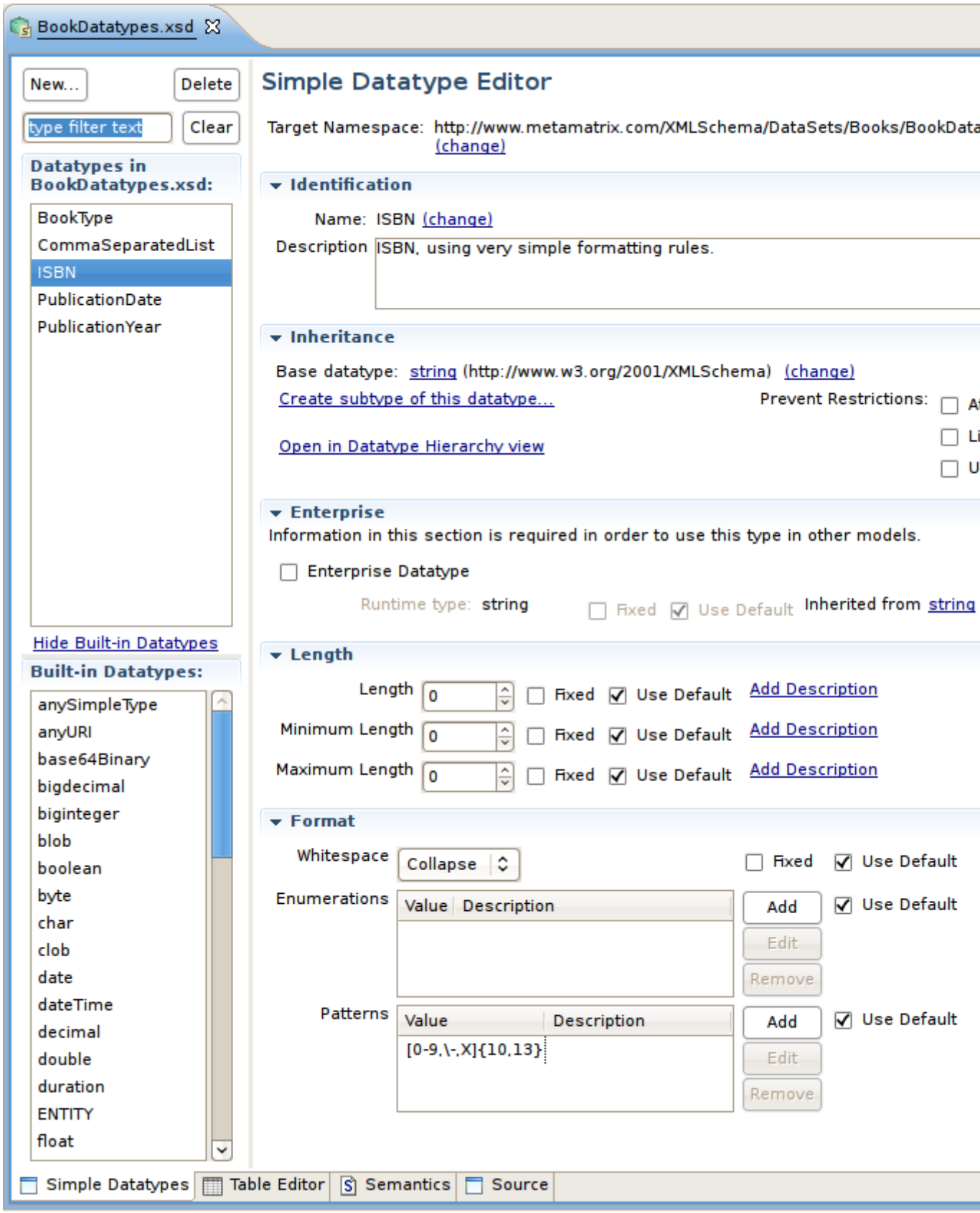


Figure D.59. Editing String Property

D.3.1.5. Semantic Editor

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

D.3.1.6. Source Editor

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

D.3.1.7. 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 6.3.1, “Transformation Editor”*

    Section 6.3.2, “Input Set Editor (XML)”

    Section 6.3.3, “Choice Editor (XML)”

    Section 6.3.4, “Recursion Editor (XML)”

    Section 6.3.5, “Operation Editor”

D.3.2. 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 number of tabs for setting and editing the various allowable properties and contents of your VDB. These tabs enable management of Models, UDF (User Defined Functions) jars and Other Files you wish to include in your VDB, tabs for managing Data Roles, VDB Properties, Description and Translator Overrides.

In our 11.1 release, we've improved usability by adding a couple more options to the toolbar.

- **Deploy** - Performs the same operation as **Modeling.... > Deploy** action.
- **Test** - Performs the same operation as **Modeling.... > Execute VDB** action.
- **Save as XML** - Performs the same function as the previous **Modeling... > Generate Dynamic VDB** action by generating an XML version of the *.vdb archive and saving it to a workspace location or to your local file system.

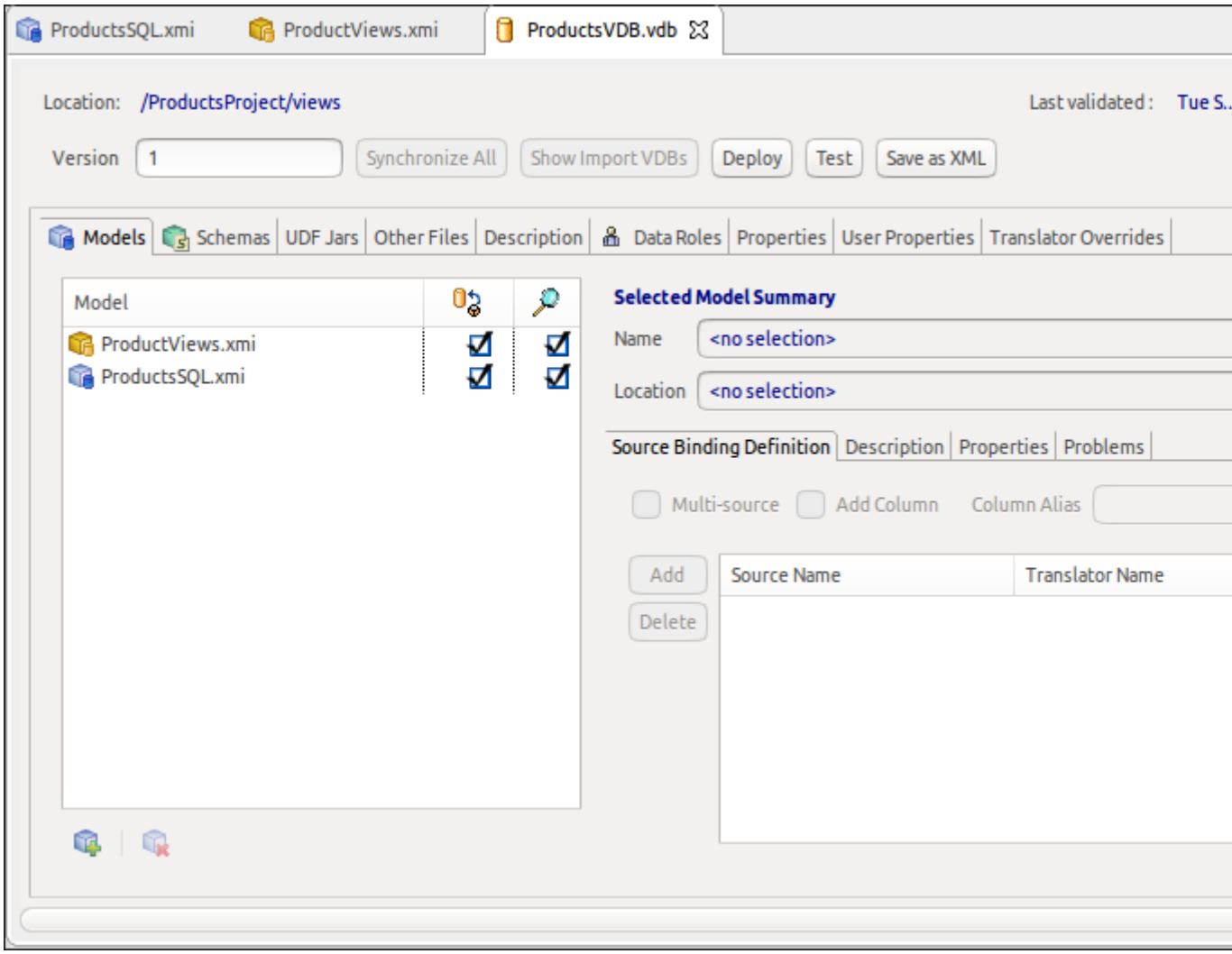


Figure D.60. VDB Editor

Note that Deploy and Test actions require you've defined and connected to a default server.

You can manage your VDB contents by using the *Add* or *Remove* models via the buttons below the Models panel.

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, 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.

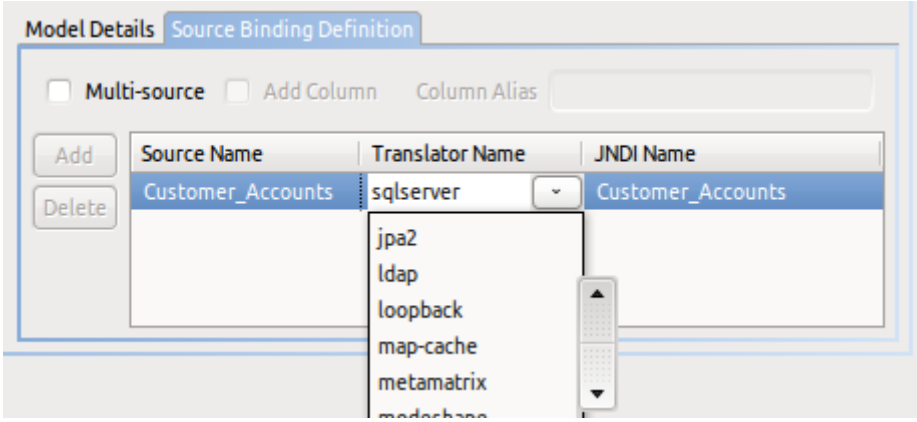


Figure D.61. Change Translator or Data Source Actions

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.

D.3.2.1. 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.

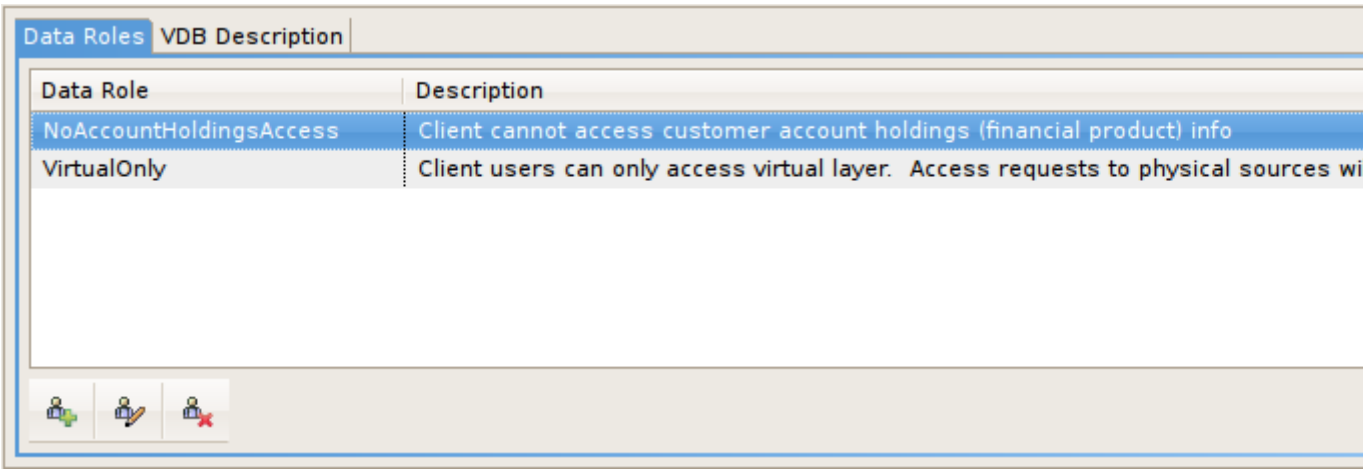


Figure D.62. VDB Data Roles Panel

Clicking *New...* or *Edit...* will launch the *New VDB Data Role* editor dialog. Speicify 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.

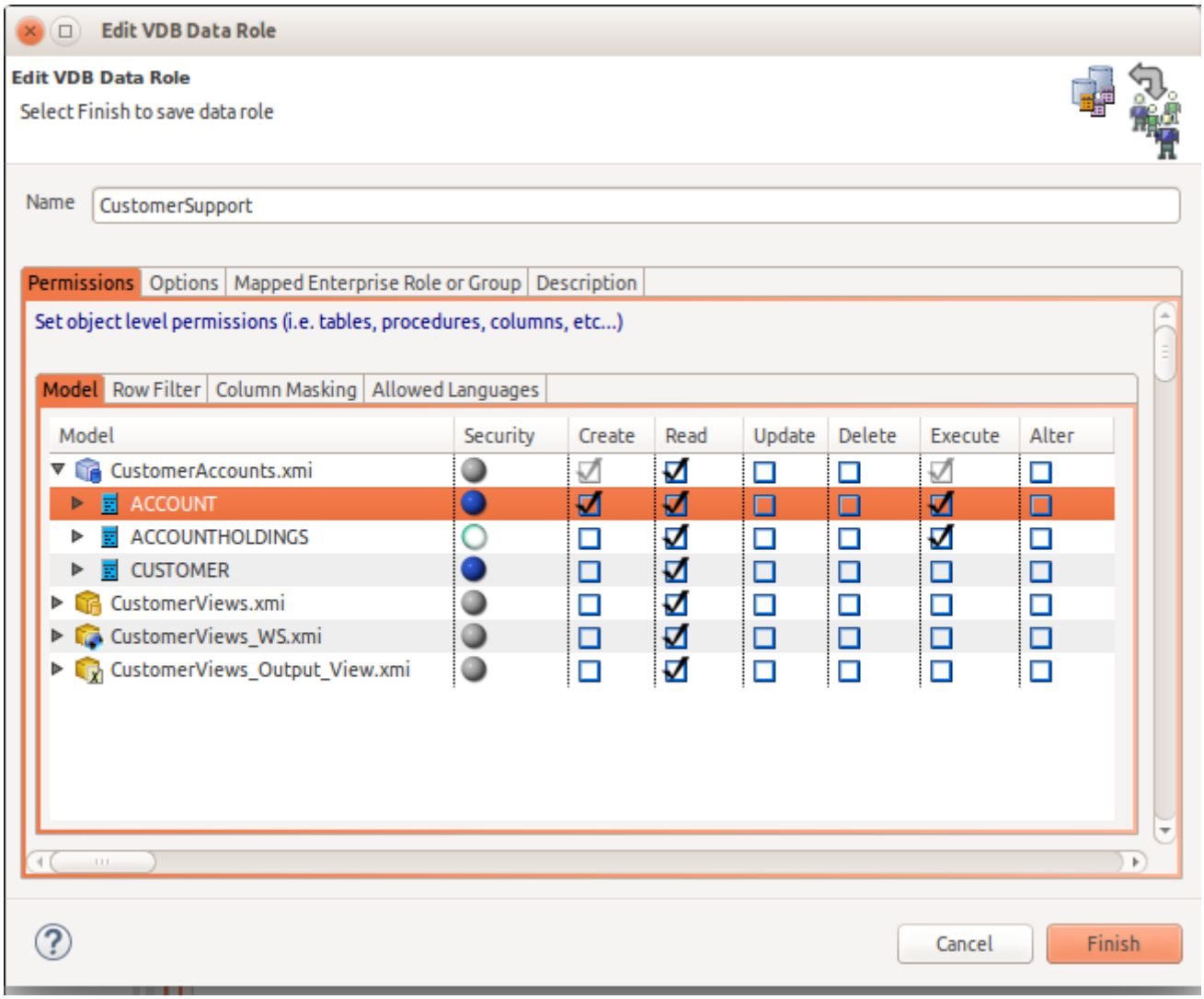


Figure D.63. VDB Data Roles Tab

D.3.2.2. Editing Translator Overrides

Teiid Designer provides a means to create, edit and manage translator override properties specific to a VDB via the Tranlator 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 *Tranlator Overrides* section consisting of a List of current tranlator overrides on the left, a properties editor panel on the right and *Add* (+) and *Remove* (-) action buttons on the lower part of the panel.

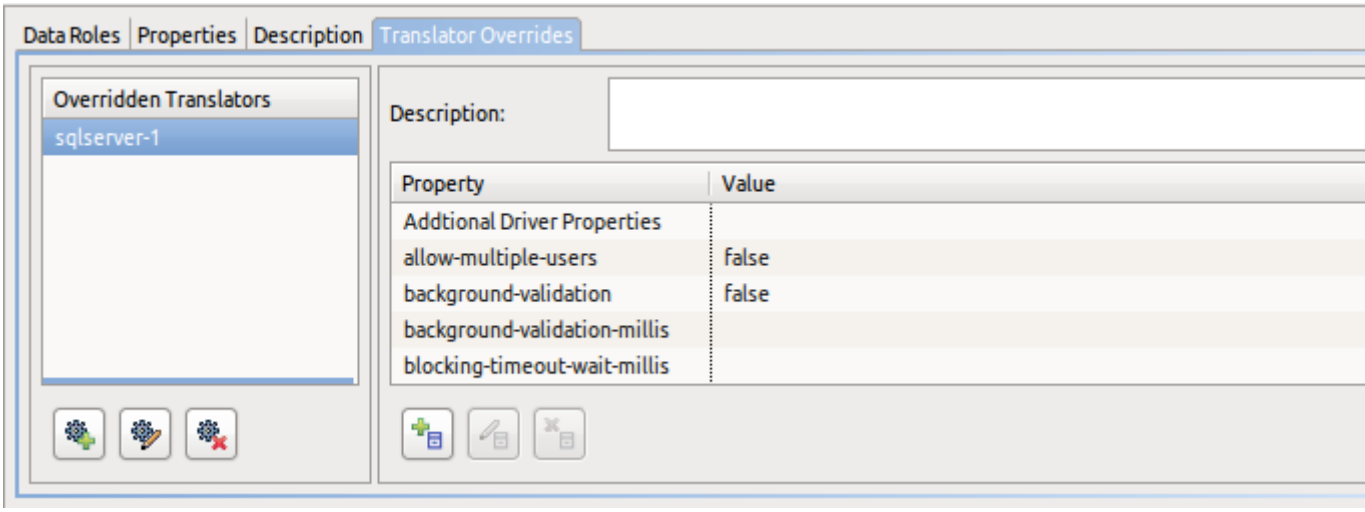


Figure D.64. VDB Translator Overrides Tab

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 tranlator type and clicks OK.

Note

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 editiable cells for each property type based on the data-type of the property. (i.e. boolean, integer, string, etc.).

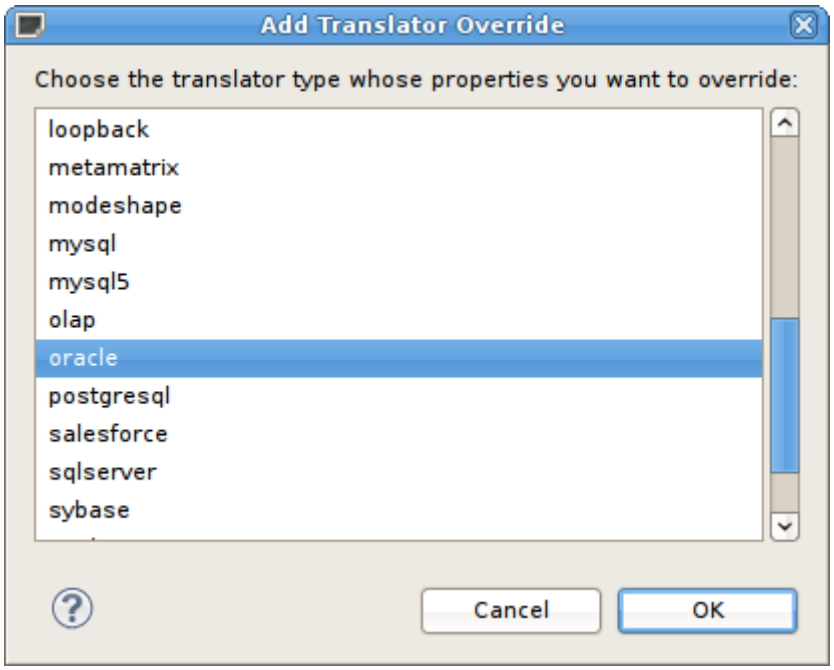


Figure D.65. Add Translator Override Dialog

If no default Teiid server instance is available, the "Add New Translator Override" dialog is presented. Enter a unique name for the tranlator 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.

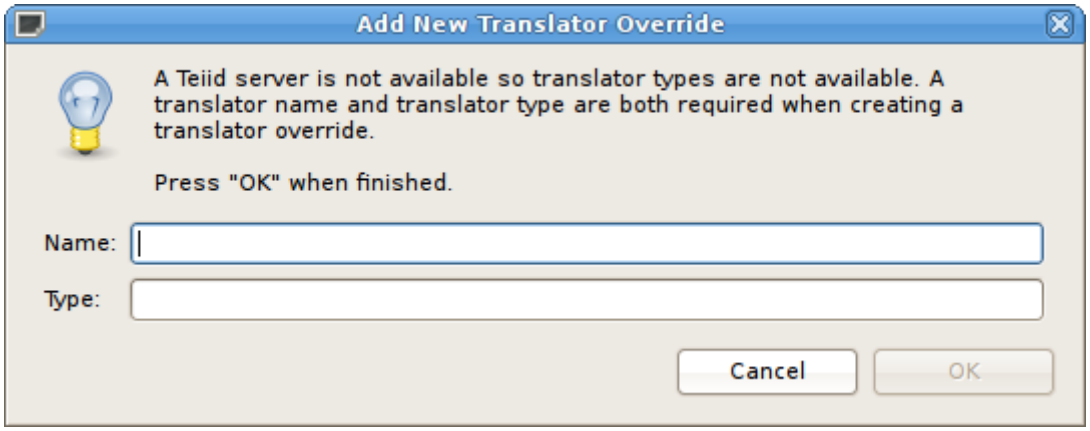


Figure D.66. Add New Translator Override Dialog

D.3.3. Model Extension Definition Editor

The **MED Editor** is a multi-tabbed editor and is used to create and edit user-defined MEDs (*.mxd files) in the workspace. The MED Editor has 3 sub-editors (**Overview, Properties, and Source**) which share a common header section. Here are the MED sub-editor tabs:

- **Overview Sub-Editor** - this editor is where the general MED information is managed. This information includes the namespace prefix, namespace URI, extended model class, and the description. The Overview sub-editor looks like this:

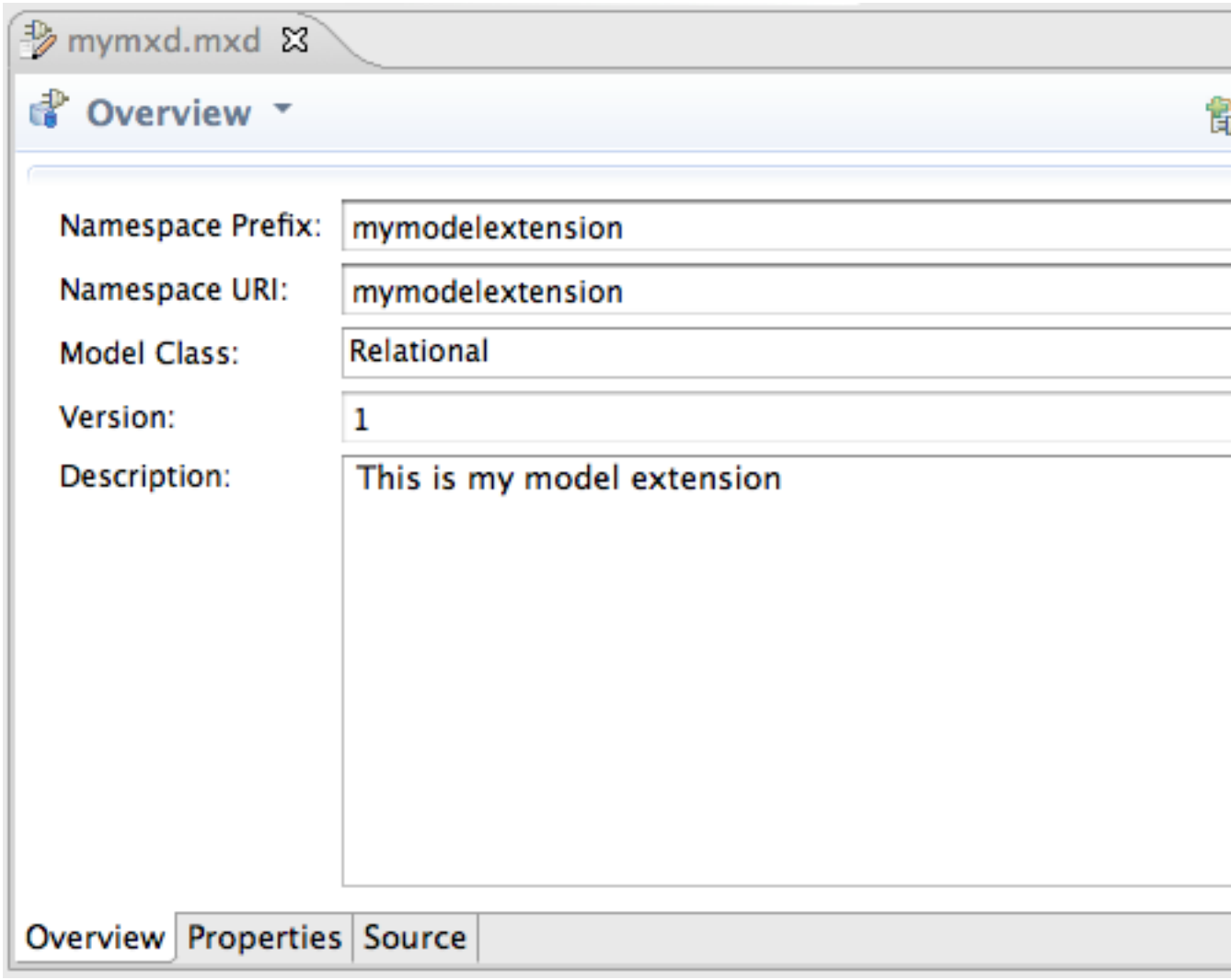


Figure D.67. Overview Tab

- **Properties Sub-Editor** - this editor is where the MED extension properties are managed. Each extension property must be associated with a model object type. The Properties sub-editor is divided into 2 sections (**Extended Model Objects** and **Extension Properties**) and looks like this:

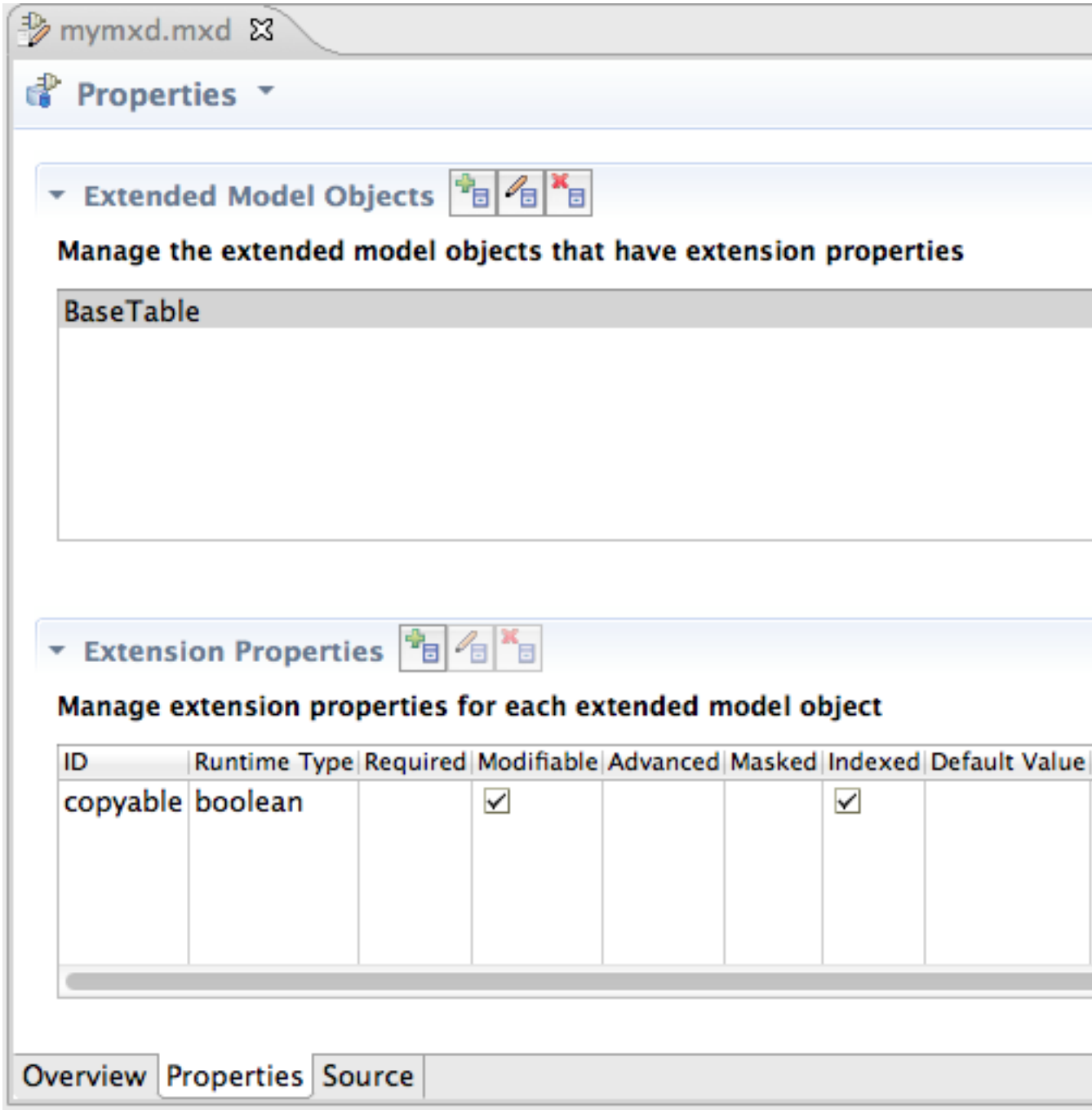


Figure D.68. Properties Tab

- **Source** - this tab is a read-only XML source viewer if you wish to view the details of your MED. This source viewer is NOT editable.

The GUI components on the Overview and Properties sub-editors will be decorated with an error icon when the data in that GUI component has a validation error. Hovering over an error decoration displays a tooltip with the specific error message. Those error message relate to the error messages shown in the common header section. Here is an example of the error decoration:

Figure D.69. Text Field With Error

The MED sub-editors share a header section. The header is composed of the following:

- **Status Image** - an image indicating the most severe validation message (error, warning, or info). If there are no validation messages the model extension image is shown.
- **Title** - the title of the sub-editor being shown.
- **Menu** - a drop-down menu containing actions for (1) adding to and updating the MED in the registry, and (2) for showing the Model Extension Registry View.
- **Validation Message** - this area will display an OK message or an error summary message. When a summary message is shown, the tooltip for that message will enumerate all the messages.
- **Toolbar** - contains the same actions as the drop-down menu.

Below is an example of the shared header section which includes an error message tooltip.

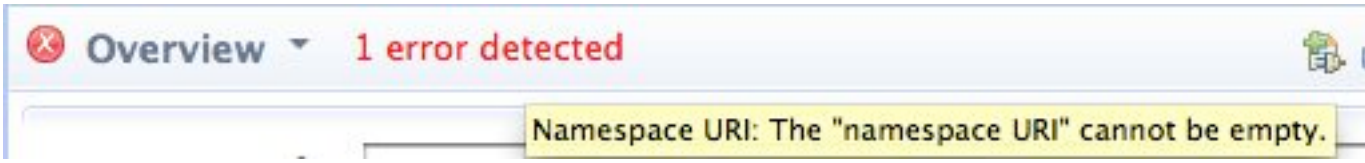


Figure D.70. Shared Header Example

D.4. Teiid Designer Main Menu

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

- These categories include:
  - *Section D.4.1, "File Menu"* - Resource management actions.
  - *Section D.4.2, "Edit Menu"* - Standard edit actions including undo/redo.
  - *Section D.4.3, "Refactor Menu"* - Resource actions (i.e. Rename, Move, etc...).
  - *Section D.4.5, "Search Menu"* - Find data within your workspace.
  - *Section D.4.6, "Project Menu"* - Model level actions.
  - *Section D.4.7, "Metadata Menu"* - Custom metadata-related actions.
  - *Section D.4.9, "Window Menu"* - Change perspectives or add/remove views to your perspective.

- *Section D.4.10, "Help Menu"* - Access available Teiid Designer help documents, Teiid Designer SQL Support Guide and Eclipse Overview information.

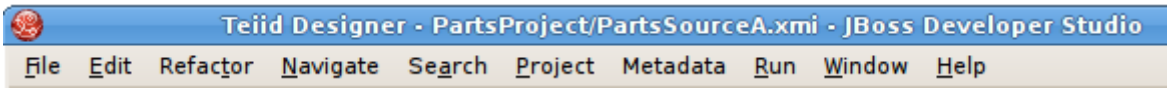


Figure D.71. Application Main Menu

D.4.1. File Menu

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

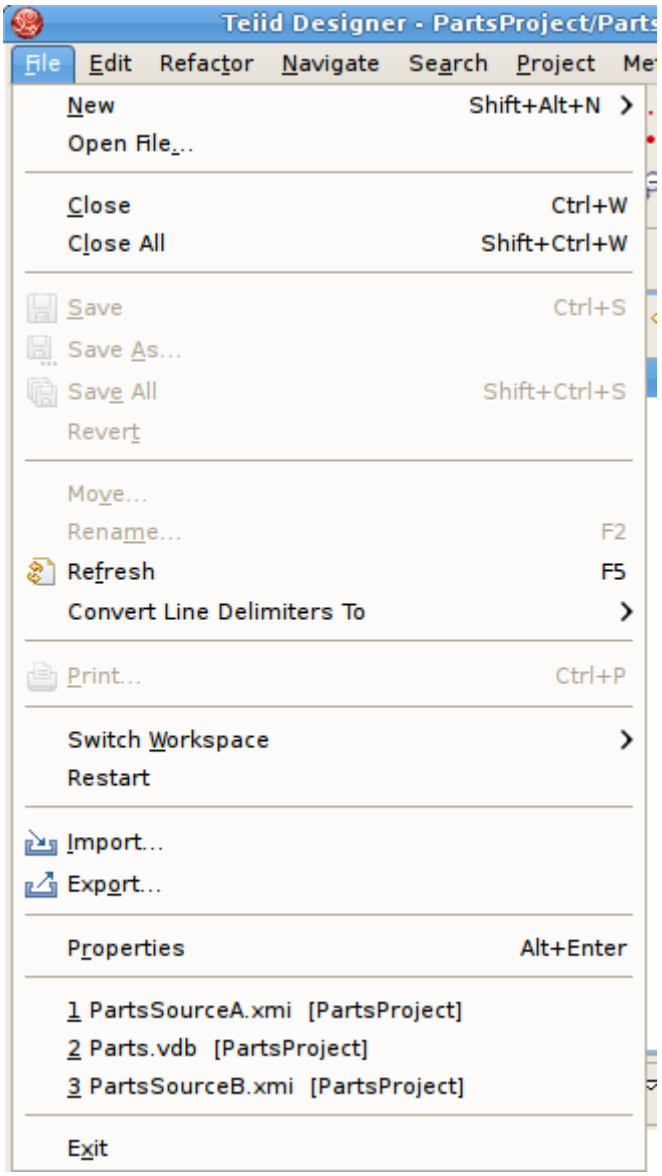


Figure D.72. File Menu

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

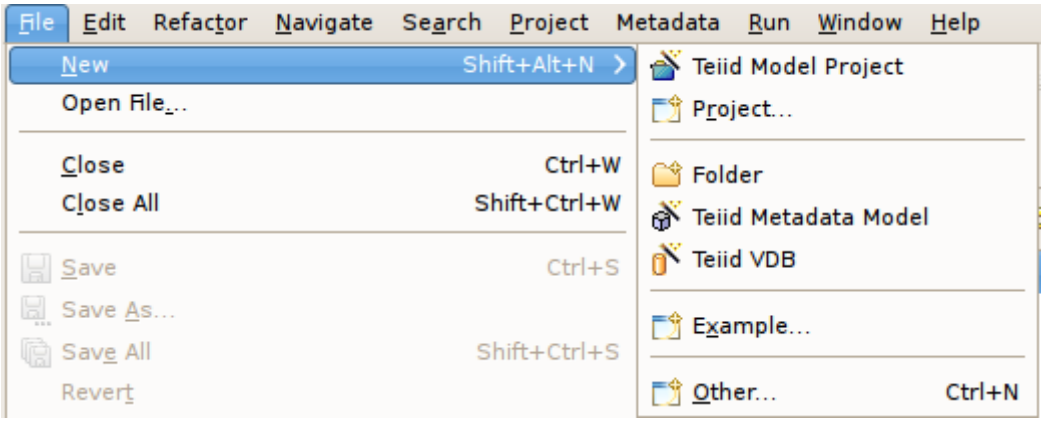


Figure D.73. File Menu

- 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 4, 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.

- **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**.

D.4.2. 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.

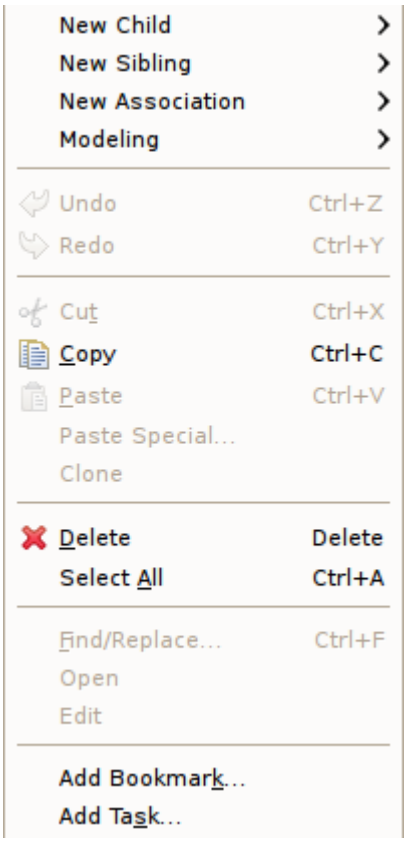


Figure D.74. Edit Menu

- 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.

D.4.3. Refactor Menu

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

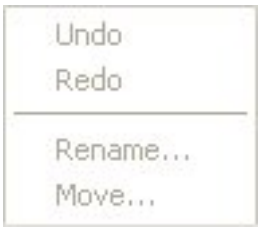


Figure D.75. Refactor Menu

- 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.

D.4.4. Navigate Menu

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

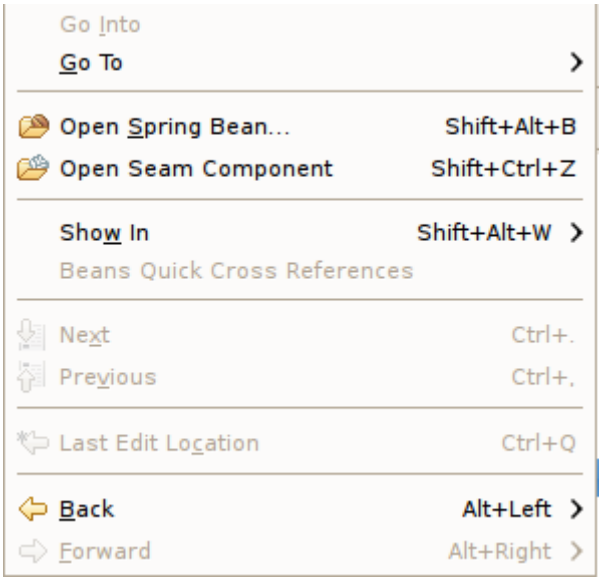


Figure D.76. Navigate Menu

D.4.5. Search Menu

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

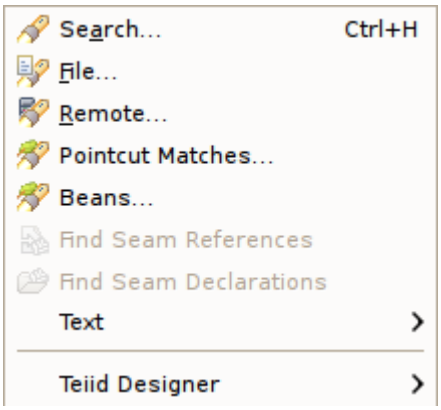


Figure D.77. Search Menu

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

Figure D.78. Search Menu

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

    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 D.2.8, "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.

D.4.6. Project Menu

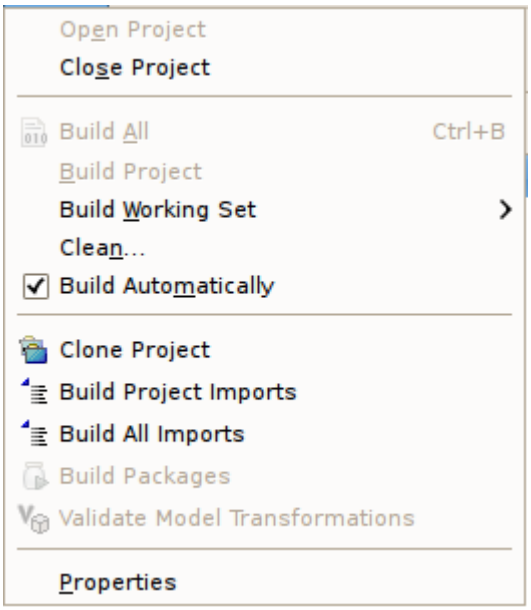


Figure D.79. Project Menu

- 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.

D.4.7. Metadata Menu

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

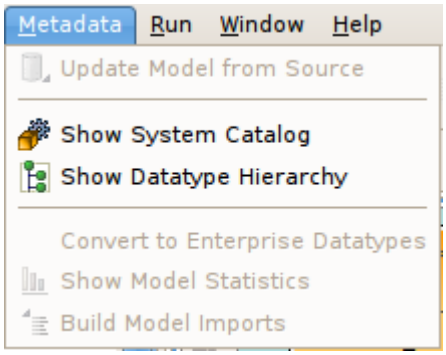


Figure D.80. Metadata Menu

- 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 D.2.11, "System Catalog View"*.
  - **Show Datatype Hierarchy** - Opens the *Section D.2.11, "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.

D.4.8. Run Menu

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

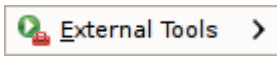


Figure D.81. Window Menu

D.4.9. Window Menu

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

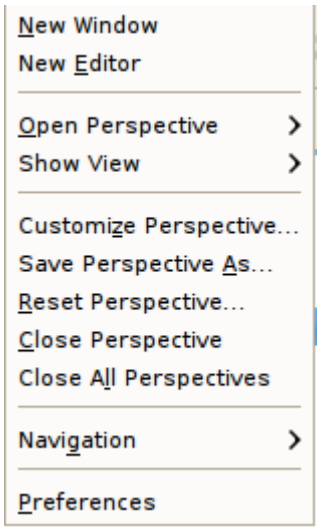


Figure D.82. Window Menu

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

Note

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.

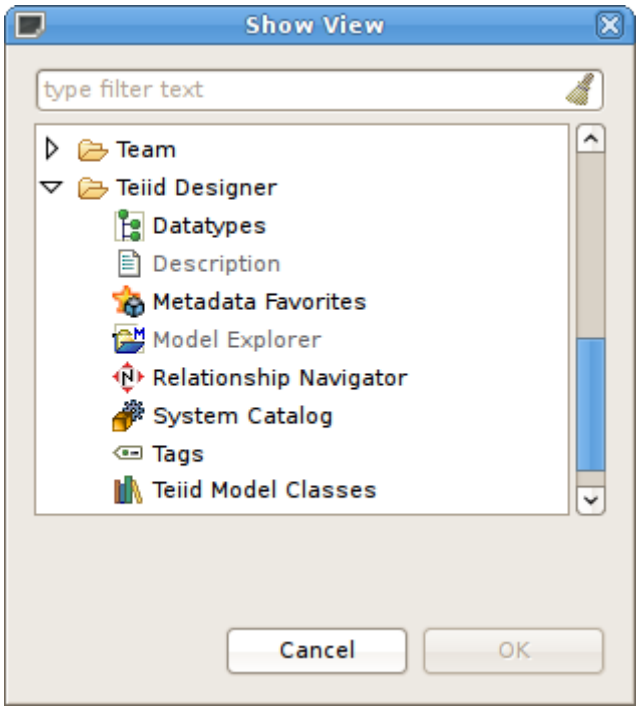


Figure D.83. Show View Dialog

D.4.10. Help Menu

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

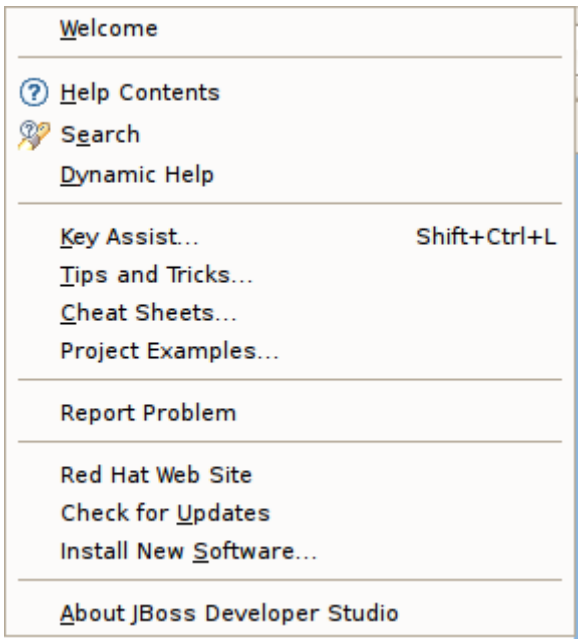


Figure D.84. Help Menu

- 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 JBoss Developer Studio** - Launches the About dialog.