Red Hat JBoss Data Virtualization 6.4

User Guide Volume 1: Teiid Designer

This guide is for developers.

Last Updated: 2018-09-26

Red Hat JBoss Data Virtualization 6.4 User Guide Volume 1: Teiid Designer

This guide is for developers.

Red Hat Customer Content Services

Legal Notice

Copyright © 2018 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document will guide user through use of the Teiid Designer plug-in for Red Hat JBoss Data Virtualization.

Table of Contents

CHAPTER 1. READ ME **6**
1.1. BACK UP YOUR DATA 6
1.2. VARIABLE NAME: EAP_HOME 6
1.3. VARIABLE NAME: MODE 6
1.4. RED HAT DOCUMENTATION SITE 6

CHAPTER 2. SOME KEY TERMS USED THROUGHOUT THIS DOCUMENT **7**
2.1. WHAT IS TEIID DESIGNER? 7
2.2. WHAT IS METADATA? 7
2.3. METADATA MODELS 7
2.4. BUSINESS AND TECHNICAL METADATA 8
2.5. TECHNICAL METADATA 8
2.6. BUSINESS METADATA 8
2.7. DESIGN-TIME AND RUNTIME METADATA 9
2.8. DESIGN-TIME METADATA 9
2.9. RUNTIME METADATA 9
2.10. WHAT ARE MODELS? 9
2.11. SOURCE AND VIEW METADATA 11
2.12. COMPACT NODE TYPE DEFINITION 11
2.13. JAVA CONTENT REPOSITORY 11
2.14. MODESHAPE TOOLS 12
2.15. THE VIRTUAL DATABASE 12

PART I. TEIID DESIGNER **13**

CHAPTER 3. INTRODUCTION TO TEIID DESIGNER **14**
3.1. WHY USE TEIID DESIGNER? 14
3.2. MODELING YOUR SOURCE METADATA 14
3.3. MODELING YOUR VIEW METADATA 14

CHAPTER 4. : MODELS **16**
4.1. GUIDING THROUGH THE PROCESS 16
4.2. MODEL CLASSES AND TYPES 16
4.3. VDB CONTENT AND STRUCTURE 16
4.4. MODEL VALIDATION 17
4.5. TESTING YOUR MODELS 17

CHAPTER 5. MODEL OBJECT EXTENSIONS **19**
5.1. MODEL OBJECT EXTENSIONS 19
5.2. MODEL EXTENSION DEFINITION (MED) 19
5.3. MODEL EXTENSION DEFINITION REGISTRY (MED REGISTRY) 20

CHAPTER 6. SERVER MANAGEMENT **21**
6.1. TARGETING THE JBOSS DATA VIRTUALIZATION SERVER 21
6.2. SETTING UP A SERVER 23
6.3. CONNECT JBDS TO A REMOTE RED HAT JBOSS DATA VIRTUALIZATION SERVER 26

CHAPTER 7. TEIID DESIGNER EXAMPLES **28**
7.1. TEIID DESIGNER EXAMPLES 28
7.2. GUIDE VIEW 28
7.3. MODEL A JDBC SOURCE 28
7.4. CHEAT SHEETS 42
7.5. CONSUME A SOAP WEB SERVICE 43

CHAPTER 8. NEW MODEL WIZARDS **59**
8.1. LAUNCH NEW MODEL WIZARDS 59
8.2. CREATE NEW RELATIONAL SOURCE MODEL 59
8.3. CREATE A NEW RELATIONAL VIEW MODEL 60
8.4. CREATE A NEW XML DOCUMENT VIEW MODEL 61
8.5. CREATE A NEW XML SCHEMA MODEL 63
8.6. CREATE A NEW WEB SERVICE VIEW MODEL 64

CHAPTER 9. IMPORTERS **66**
9.1. IMPORTERS 66
9.2. IMPORT DDL 67
9.3. IMPORT FROM JDBC DATABASE 69
9.4. IMPORT FROM TEIID DATA SOURCE CONNECTION 74
9.5. IMPORT FROM FLAT FILE SOURCE 75
9.6. IMPORT FROM XML DATA FILE SOURCE 86
9.7. IMPORT FROM SALESFORCE 92
9.8. IMPORT METADATA FROM TEXT FILE 97
9.9. IMPORT WSDL INTO RELATIONAL MODELS 104
9.10. IMPORT WSDL INTO WEB SERVICE 108
9.11. IMPORT DATA FROM REST SERVICES 125
9.12. LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) 126
9.13. IMPORT FROM AN LDAP SERVER 126
9.14. LDAP CONNECTOR UPDATE CAPABILITIES 128
9.15. XSD SCHEMA FILE 129
9.16. RED HAT JBOSS DATA GRID AND COMPLEX DATA TYPES 133

CHAPTER 10. CREATING AND EDITING MODEL OBJECTS **135**
10.1. CREATING NEW MODEL OBJECTS 135
10.2. MODEL OBJECT EDITORS 141
10.3. TRANSFORMATION EDITOR 143
10.4. INPUT SET EDITOR (XML) 156
10.5. CHOICE EDITOR (XML) 158
10.6. RECURSION EDITOR (XML) 160
10.7. OPERATION EDITOR 163
10.8. MANAGING MODEL OBJECT EXTENSIONS 165

CHAPTER 11. USER DEFINED FUNCTIONS **172**
11.1. USER DEFINED FUNCTIONS 172
11.2. MODELING YOUR FUNCTIONS 172
11.3. UTILIZING YOUR FUNCTIONS IN TRANSFORMATIONS 173
11.4. INCLUDING FUNCTIONS IN YOUR VDB 173

CHAPTER 12. METADATA-SPECIFIC MODELING **175**
12.1. RELATIONAL SOURCE MODELING 175
12.2. RELATIONAL VIEW MODELING 182
12.3. XML DOCUMENT MODELING 205
12.4. WEB SERVICES MODELING 207

CHAPTER 13. EDITING MODELS AND PROJECTS **222**
13.1. EDITING MODELS AND PROJECTS 222
13.2. RENAME A MODEL 222
13.3. MOVE MODEL 223
13.4. SAVE COPY OF MODEL 224
13.5. CLONE PROJECT 226

CHAPTER 14. MANAGING YOUR VIRTUAL DATABASES **229**
14.1. CREATE A VIRTUAL DATABASE 229
14.2. EDIT A VIRTUAL DATABASE 229
14.3. MULTI-SOURCE BINDING SUPPORT 229
14.4. USER-DEFINED FUNCTIONS SUPPORT 230
14.5. REUSING VIRTUAL DATABASES 230
14.6. CREATE A VDB SOURCE MODEL 231
14.7. SECURITY AND DATA ACCESS 232

CHAPTER 15. TESTING YOUR MODELS **234**
15.1. MANAGE CONNECTION PROFILES 234
15.2. PREVIEWING DATA FOR A MODEL 235
15.3. TESTING WITH YOUR VDB 241

CHAPTER 16. SEARCHING **248**
16.1. SEARCHING 248
16.2. FINDING MODEL OBJECTS 248
16.3. SEARCH TRANSFORMATION SQL 248
16.4. SEARCH MODELS VIA METADATA PROPERTIES 249

APPENDIX A. SUPPORTED CONFIGURATIONS **250**
A.1. SUPPORTED DATA SOURCES AND TRANSLATORS 250
A.2. DESIGNER METADATA USAGE REQUIREMENTS IN JBOSS DATA VIRTUALIZATION RUNTIME 252

APPENDIX B. USER PREFERENCES **266**
B.1. USER PREFERENCES 266
B.2. TEIID DESIGNER PREFERENCES 266
B.3. DIAGRAM PREFERENCES 267
B.4. EDITOR PREFERENCES 268
B.5. VALIDATION PREFERENCES 271

APPENDIX C. TEIID DESIGNER UI REFERENCE **274**
C.1. TEIID DESIGNER PERSPECTIVES 274
C.2. TEIID DESIGNER PERSPECTIVE 274
C.3. OPENING A PERSPECTIVE 275
C.4. FURTHER INFORMATION 276

APPENDIX D. TEIID DESIGNER VIEWS **278**
D.1. TEIID DESIGNER VIEWS 278
D.2. MODEL EXPLORER VIEW 279
D.3. SELECTION-BASED ACTION MENUS 281
D.4. OUTLINE VIEW 282
D.5. OUTLINE TREE VIEW 282
D.6. OUTLINE THUMBNAIL VIEW 283
D.7. SERVER VIEW 284
D.8. PROPERTIES VIEW 289
D.9. DESCRIPTION VIEW 290
D.10. PROBLEMS VIEW 292
D.11. TOOLBAR MENU 293
D.12. CONTEXT MENU 293
D.13. SEARCH RESULTS VIEW 294
D.14. DATATYPE HIERARCHY VIEW 296
D.15. TEIID MODEL CLASSES VIEW 300
D.16. SYSTEM CATALOG VIEW 301
D.17. SQL RESERVED WORDS VIEW 302

D.18. MODEL EXTENSION DEFINITION REGISTRY VIEW (MED REGISTRY VIEW) 303
D.19. GUIDES VIEW 304
D.20. STATUS VIEW 306
D.21. CHEAT SHEETS VIEW 307
D.22. CONNECTIONS VIEW 307

APPENDIX E. EDITORS 310
E.1. EDITORS 310
E.2. MODEL EDITOR 312
E.3. TABLE EDITOR 316
E.4. SIMPLE DATATYPES EDITOR 321
E.5. SEMANTIC EDITOR 322
E.6. SOURCE EDITOR 322
E.7. MODEL OBJECT EDITORS 322
E.8. VDB EDITOR 323
E.9. MODEL EXTENSION DEFINITION EDITOR 328

APPENDIX F. TEIID DESIGNER MAIN MENU 332
F.1. TEIID DESIGNER MAIN MENU 332
F.2. FILE MENU 332
F.3. EDIT MENU 335
F.4. REFACTOR MENU 337
F.5. NAVIGATE MENU 337
F.6. SEARCH MENU 338
F.7. PROJECT MENU 339
F.8. METADATA MENU 340
F.9. RUN MENU 341
F.10. WINDOW MENU 341
F.11. HELP MENU 342

APPENDIX G. REVISION HISTORY 344

CHAPTER 1. READ ME

1.1. BACK UP YOUR DATA

WARNING

Red Hat recommends that you back up your system settings and data before undertaking any of the configuration tasks mentioned in this book.

1.2. VARIABLE NAME: EAP_HOME

`EAP_HOME` refers to the root directory of the Red Hat JBoss Enterprise Application Platform installation on which JBoss Data Virtualization has been deployed.

1.3. VARIABLE NAME: MODE

`MODE` will either be **`standalone`** or **`domain`** depending on whether JBoss Data Virtualization is running in standalone or domain mode. Substitute one of these whenever you see **`MODE`** in a file path in this documentation. (You need to set this variable yourself, based on where the product has been installed in your directory structure.)

1.4. RED HAT DOCUMENTATION SITE

Red Hat's official documentation site is available at https://access.redhat.com/site/documentation/. There you will find the latest version of every book, including this one.

CHAPTER 2. SOME KEY TERMS USED THROUGHOUT THIS DOCUMENT

2.1. 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.

2.2. 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 us 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 2.4, “Business and Technical Metadata”)
- What data does the metadata represent? (See Section 2.11, “Source and View Metadata”)
- How will my organization use and manage this metadata? (See Section 2.7, “Design-Time and Runtime Metadata”)

2.3. METADATA MODELS

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 represent 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 Source and View Metadata.

2.4. BUSINESS AND TECHNICAL METADATA

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

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

NOTE

The terms technical and business metadata, refer to the content of the metadata, namely what type of information is contained in the metadata. Do not confuse these with the terms physical and view metadata that indicate what the metadata represents.

2.5. 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.

2.6. 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.

2.7. DESIGN-TIME AND RUNTIME METADATA

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

2.8. DESIGN-TIME METADATA

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

2.9. RUNTIME METADATA

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

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

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

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

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

2.10. 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`** (for 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" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
  <mmcore:ModelAnnotation xmi:uuid="mmuuid:b0355f00-413b-1079-9d18-8acf4a7127d5" description="Northwind_Oracle was cre
    <modelImports xmi:uuid="mmuuid:2d815780-4140-1079-9d18-8acf4a7127d5" name="XMLSchema" modelLocation="http://www.w3
    <modelImports xmi:uuid="mmuuid:2e663940-4140-1079-9d18-8acf4a7127d5" name="SimpleDatatypes-instance" modelLocatio
  </mmcore:ModelAnnotation>
  <relational:BaseTable xmi:uuid="mmuuid:b20e64c0-413b-1079-9d18-8acf4a7127d5" name="CATEGORIES" nameInSource="CATEGOI
    <columns xmi:uuid="mmuuid:bb5ac3c0-413b-1079-9d18-8acf4a7127d5" name="CATEGORYID" nameInSource="CATEGORYID" native
      <type href="http://www.w3.org/2001/XMLSchema#long"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c0-413b-1079-9d18-8acf4a7127d5" name="CATEGORYNAME" nameInSource="CATEGORYNAME" na
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c1-413b-1079-9d18-8acf4a7127d5" name="DESCRIPTION" nameInSource="DESCRIPTION" nat:
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c2-413b-1079-9d18-8acf4a7127d5" name="PICTURE" nameInSource="PICTURE" nativeType='
      <type href="http://www.metamatrix.com/metamodels/SimpleDatatypes-instance#blob"/>
    </columns>
    <primaryKey xmi:uuid="mmuuid:d481f940-413b-1079-9d18-8acf4a7127d5" name="PK_CATEGORIES" nameInSource="PK_CATEGORII
  </relational:BaseTable>
```

Figure 2.1. 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. **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.

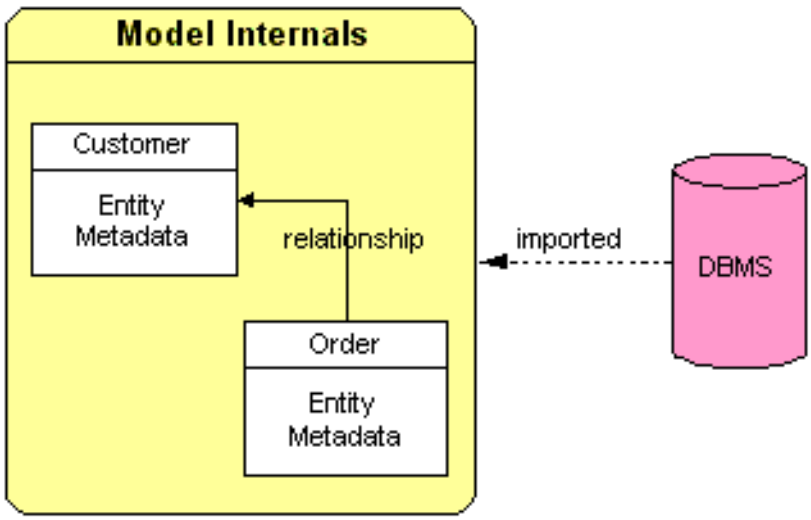


Figure 2.2. Model Internals

In addition to source models, **Teiid Designer** provides the ability to define a variety of view models. 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

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

2.11. SOURCE AND VIEW METADATA

In addition to the distinction between business and technical metadata, it is necessary to know the difference between Source Metadata and View Metadata.

Source and View metadata refers 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 are modeling the data as your applications (and your enterprise) ultimately use it.

2.12. COMPACT NODE TYPE DEFINITION

Compact Node Type Definition (CND) is a file format that defines node type in a compact standardized format. A typical CND file contains series of namespace declarations, node type definitions and comments. CND provides a compact and standard syntax for defining node types and making namespace declarations.

2.13. JAVA CONTENT REPOSITORY

Java Content Repository is a content repository for Java API. JCR provides the following information related services:

- read/write access to information,
- the ability to structure files in hierarchical manner,
- the ability to work with structured/unstructured content,
- the ability to search and query,
- versioning of information, and
- the ability to control access to the content.

2.14. MODESHAPE TOOLS

ModeShape Tools is a set of Eclipse plugins you can use to work with ModeShape and Java Content Repositories (JCRs).

2.15. THE VIRTUAL DATABASE

The critical artifact that **Teiid Designer** is intended to manage is the VDB, or Virtual DataBase. Through the JBoss Data Virtualization server, VDB's behave like standard JDBC 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.

PART I. TEIID DESIGNER

CHAPTER 3. INTRODUCTION TO TEIID DESIGNER

3.1. WHY USE TEIID DESIGNER?

Teiid Designer allows you map data sources to target formats. It also lets you do the following:

- 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 target formats which is compatible and optimized for consumption by your applications

It allows you to abstract the structure of the information you 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 also use Teiid Designer to integrate multiple sources, and access them using these common data access standards:

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

3.2. 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 JBoss Data Virtualization 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.

3.3. 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 JBoss Data Virtualization 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.

CHAPTER 4. : MODELS

4.1. GUIDING THROUGH THE PROCESS

To make the process of using Teiid Designer to build models more as easy as posssible, a guides view (See Section D.19, “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 scenarios:

- Consuming a SOAP Web Service
- Modelling from a Flat File Source (a text file)
- Modelling from a JDBC Data Source
- Modelling from a Local XML File Source
- Modelling from a Remote XML File Source
- Modelling from a Red Hat JBoss Data Virtualization Data Source (deployed on server)
- Connecting to a Red Hat JBoss Data Virtualization Server

4.2. 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.

4.3. VDB CONTENT AND STRUCTURE

In Teiid 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.
- project folder - contains of the models you will be adding in the VDB Editor (that is, **`*.xmi`** and **`*.xsd`** files)

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

The vdb.xml file contains:

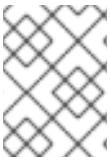
- VDB name, version, properties
- contained model information (name, translator name, connection info)
- translator info
- data role definitions for the referenced models
- import VDB references

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 Section E.8.1, “VDB Editor”)

4.4. MODEL VALIDATION

Models must be in a valid state in order to be used for data access. Validation of a single model means that it must be in a self-consistent and complete state, meaning that there are no missing pieces and no references to non-existent entities. Validation of multiple models checks that all inter-model dependencies are present and resolvable.

Models must always be validated when they are deployed in a VDB for data access purposes. **Teiid Designer** will automatically validate all models whenever they are saved.



NOTE

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

4.5. TESTING YOUR MODELS

Designing and working with data is often much easier when you can see the information you are 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.

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.

CHAPTER 5. MODEL OBJECT EXTENSIONS

5.1. MODEL OBJECT EXTENSIONS

Teiid Designer in conjunction with JBoss Data Virtualization provides an extensible framework to define custom properties for model objects other than 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.

Teiid Designer introduces a new Model Extension Definition (MED) framework that will replace the EMF based Model Extension metamodel.

This 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 10.8.1, “Managing Model Object Extensions” and Section E.9, “Model Extension Definition Editor”.

5.2. 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 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 **Teiid Designer** are stored. MED files can be edited by opening the **`.mxd`** file in the Extension Editor.

5.3. 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 three different types of MEDs stored in the registry:

- Built-In MED - these are registered during Teiid Designer installation. These MEDs cannot be updated or unregistered by the user.
- Local MED - these are created or imported by the user. These MEDs can be updated, registered, and unregistered by the user.
- Imported MED - these are imported automatically when the server starts for the first time. Any functions that depend on these MEDs are available after the server starts for the first time.

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

CHAPTER 6. SERVER MANAGEMENT

6.1. TARGETING THE JBOSS DATA VIRTUALIZATION SERVER

6.1.1. Targeting the JBoss Data Virtualization Server

The **JBoss Data Virtualization Server** is the destination for Teiid Designer's modeling. 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 the **JBoss Data Virtualization Server** in the Servers View.

6.1.2. Server Version Preference

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

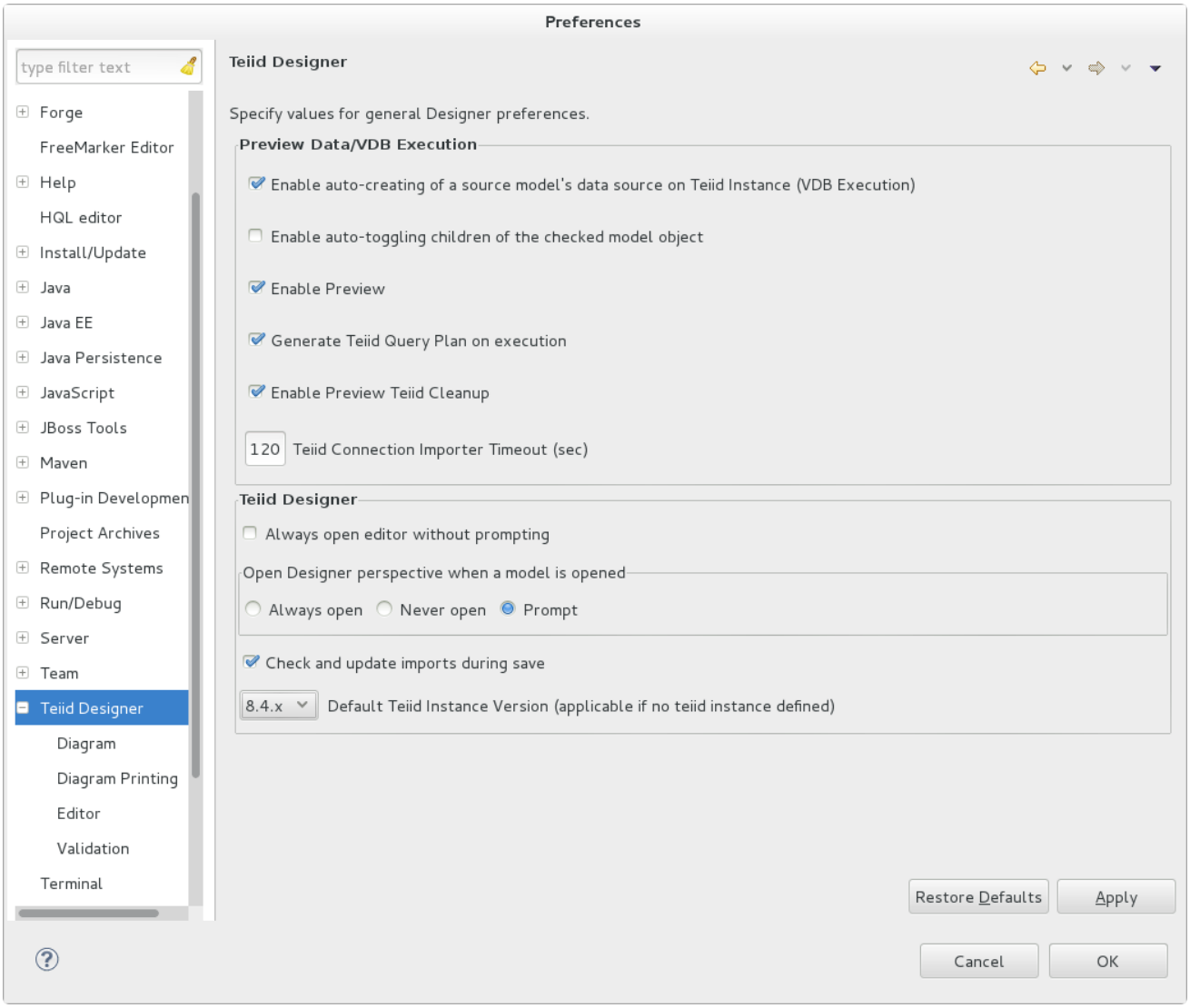


Figure 6.1. Default Server Version Preference

6.1.3. Defining JBoss Data Virtualization Server

You can preview and test the deployment of the models by defining a JBoss Data Virtualization Server. 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 you use the context menu actions in the Server View section.

The Guides View provides the following **JBoss Data Virtualization Server** actions.

- Configure New JBoss Server
- Edit JBoss/Teiid Instance Properties
- Set the Default JBoss/Teiid Instance
- Refresh Teiid Instance

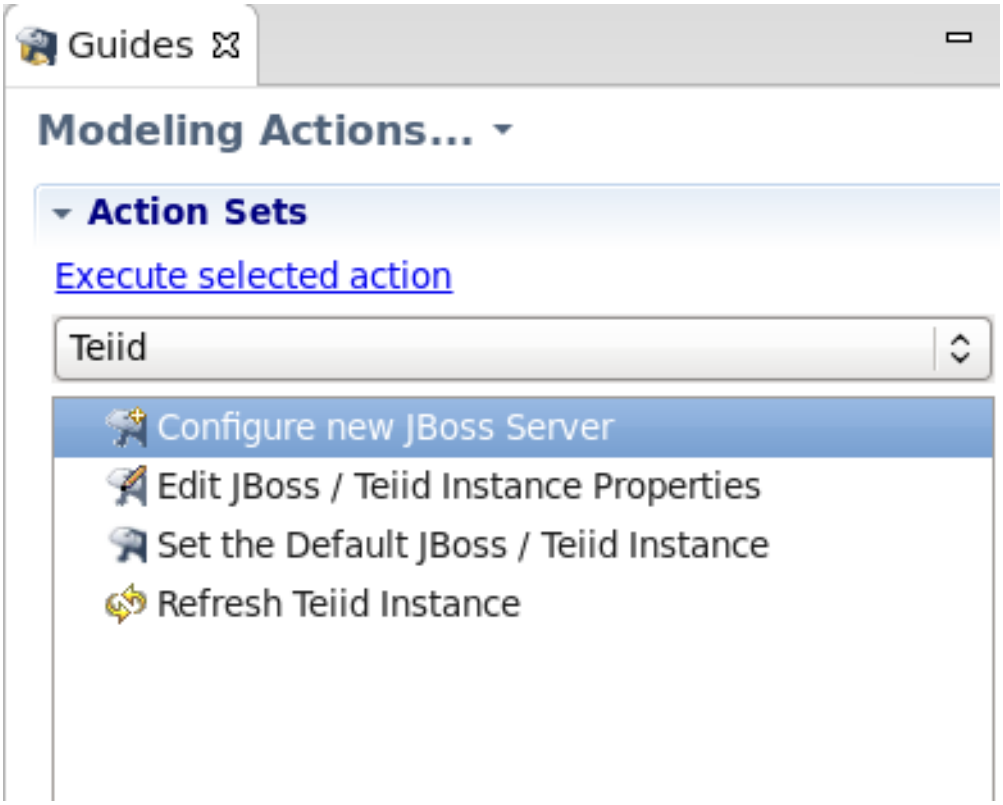


Figure 6.2. Teiid Server Category in the Guides View

The **`Configure New JBoss Server`** action will display the wizard outlined in the Setting Up a Server section and steps through the process of creating both the JBoss Data Virtualization instance and its parent JBoss server in the Server View.

Should more than one JBoss Data Virtualization Server be defined in the Server View then the **`Set the Default JBoss/Teiid Instance`** action allows for the default server to be changed appropriately. The JBoss Data Virtualization server that is currently selected in the Server View is considered as the default server. However, should nothing be selected then a dialog will be displayed inviting you to choose the default server.

NOTE

The version of the defined JBoss Data Virtualization server always takes precedence over the server version selected in the Preference window.

6.1.4. 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 which is located in the **`Connections`** view. This will always reflect the current server version being targeted and the server being used to preview or deploy against.

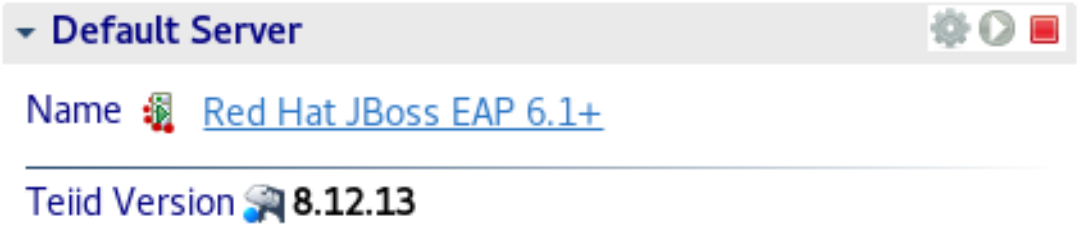


Figure 6.3. Default Server Status Panel

6.2. SETTING UP A SERVER

Teiid Designer is installed as a component of JBoss Developer Studio hence connection to a JBoss Data Virtualization Server requires the setting up and configuration of its parent JBoss Server. This is achieved using the Server View, see Server View section, displayed as part of the Teiid Designer perspective.

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.

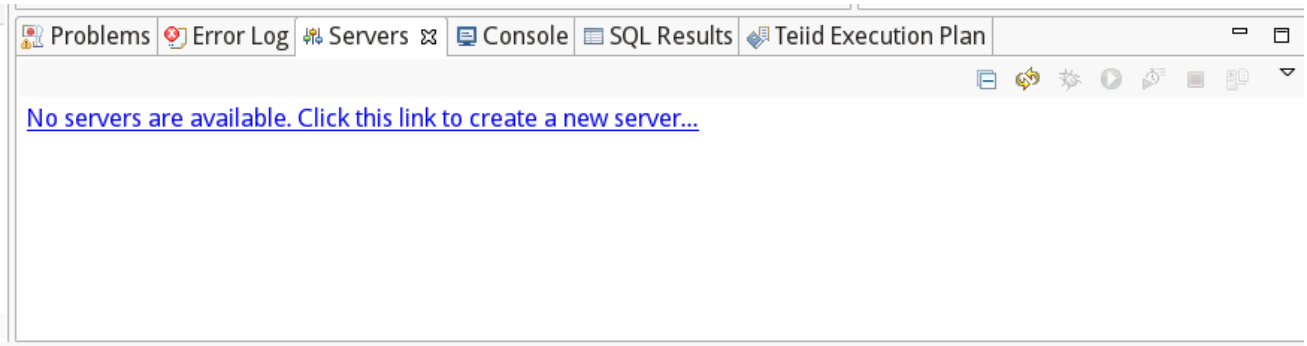


Figure 6.4. Server View with no created servers

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.

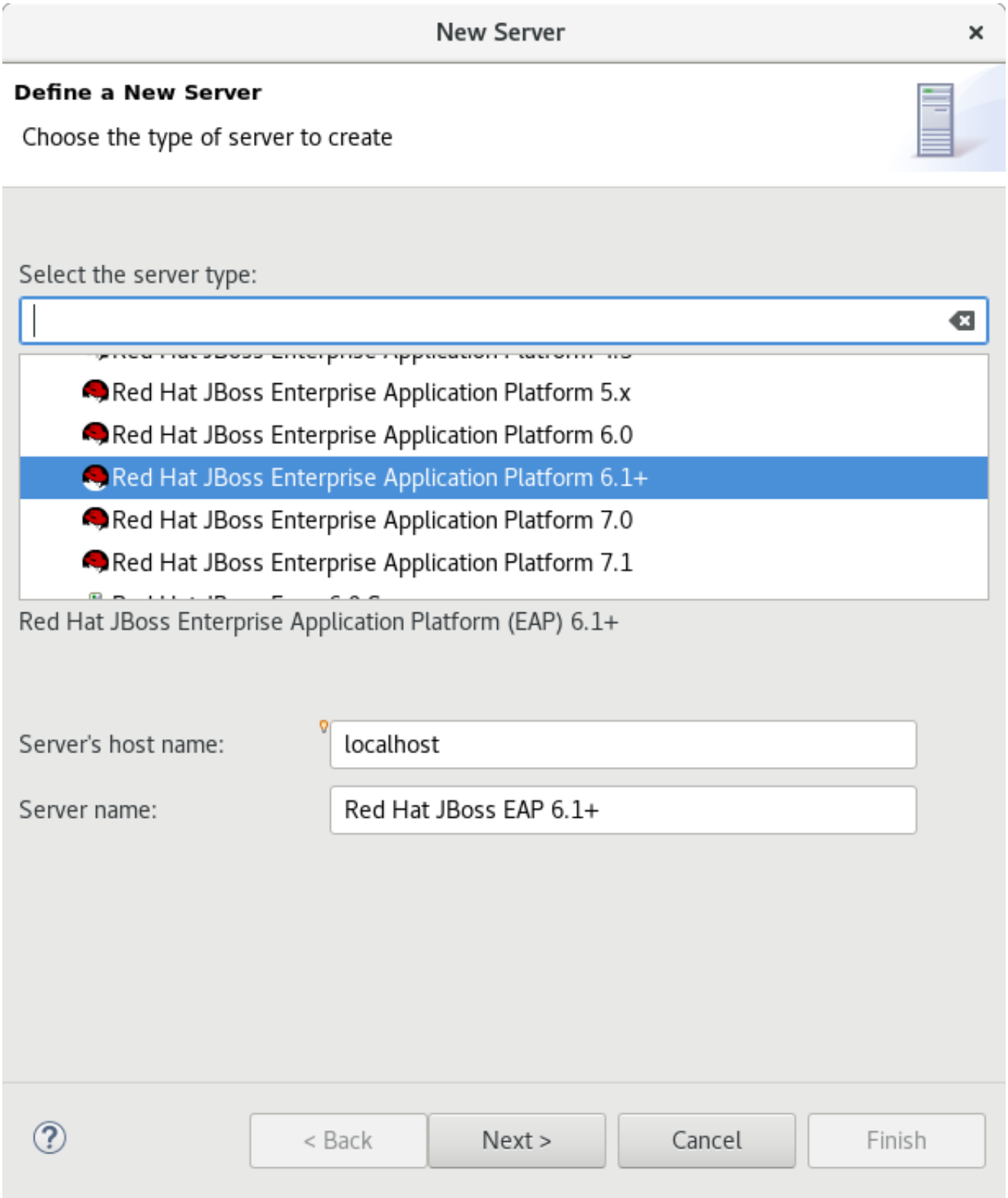


Figure 6.5. New Server Configuration

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

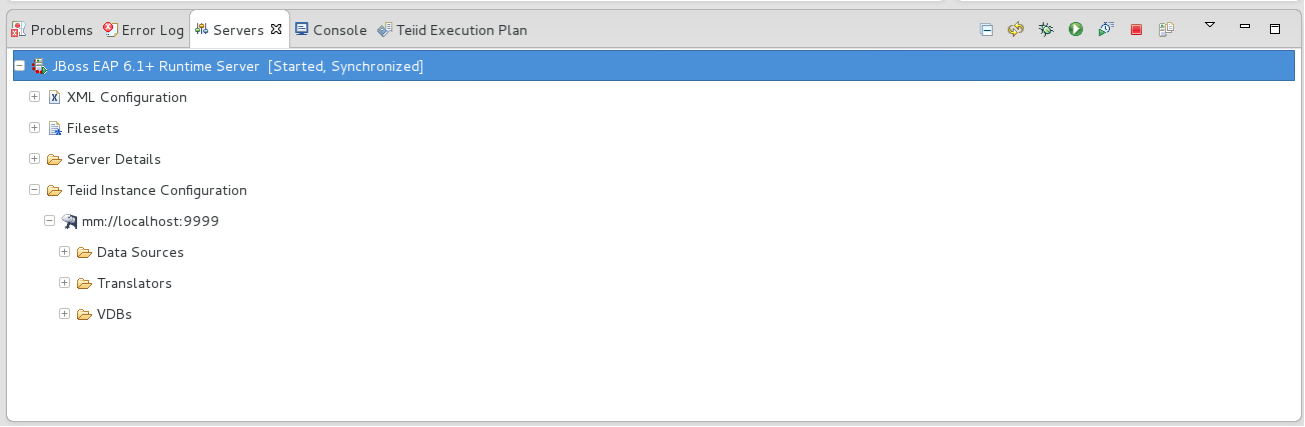


Figure 6.6. Server View with a single server

JBoss Tools provides an editor for the configuration of the JBoss Data Virtualization Server. In addition, Designer provides an extra tab to this editor that displays the configuration of the JBoss Data Virtualization 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.

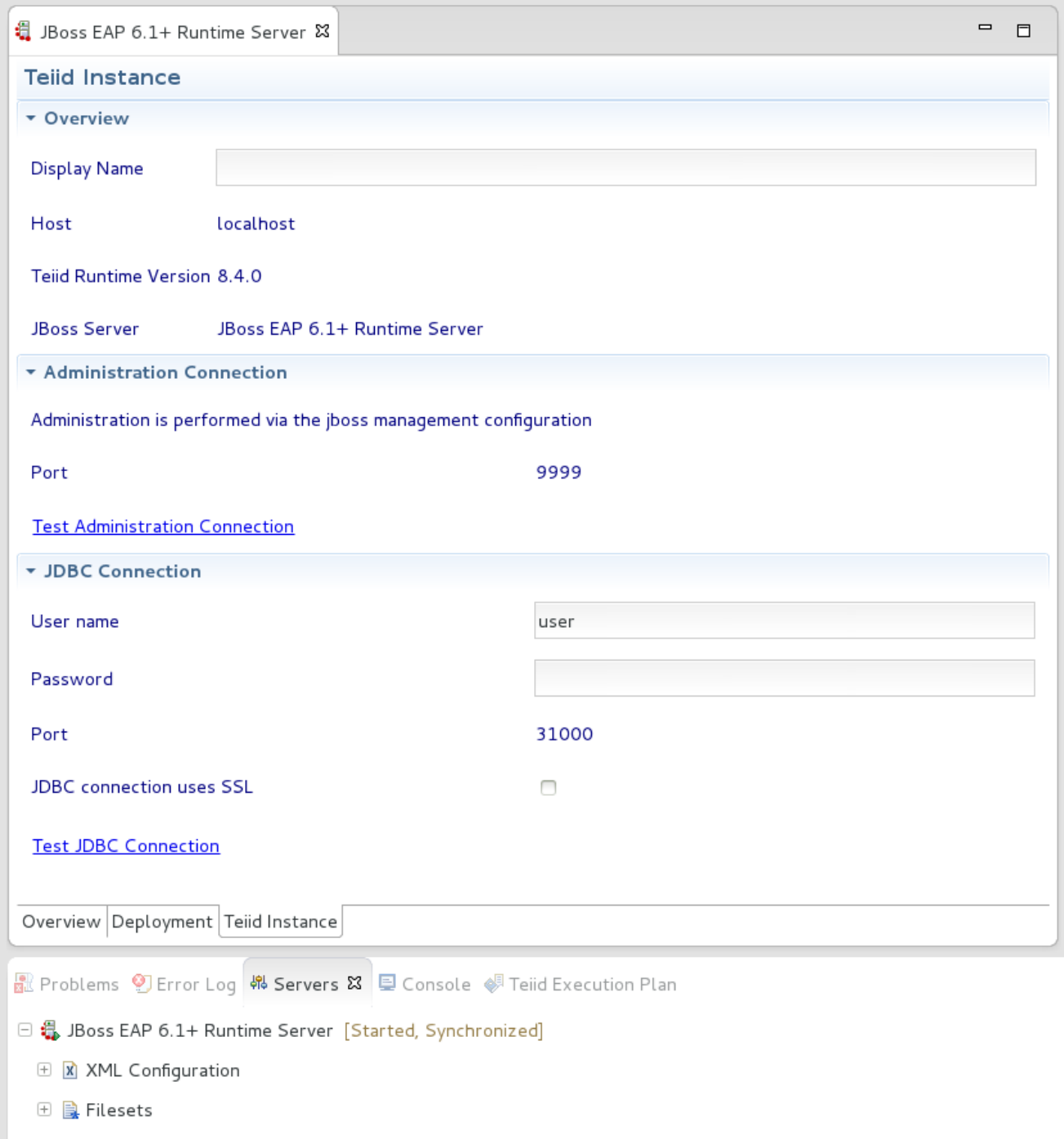


Figure 6.7. Editor for Configuring the Teiid Server

6.3. CONNECT JBDS TO A REMOTE RED HAT JBOSS DATA VIRTUALIZATION SERVER

1. Select File - New - Other - Server
2. Select Red Hat JBoss Middleware -> JBoss Enterprise Application Platform 6.x+
3. Set the host to your remote server host name (such as myserver01.labs.mycompany.com) and click Next.
4. Select Remote, select Management Operations, check Server lifecycle is externally managed, uncheck Assign a runtime to this server and click Next.

5. Click New Host, select SSH Only and then click Next.
6. The Hostname should be pre-filled but otherwise paste in your host name (that is, myserver01.labs.mycompany.com), set the connection name and click Finish.
7. Click Finish again (there is no need to fill in Remote Server Home entry).
8. Double-click the server in Servers View, set the admin username/password and save.

NOTE

You do not need to set up the users on the local machine.

CHAPTER 7. TEIID DESIGNER EXAMPLES

7.1. TEIID DESIGNER EXAMPLES

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 Teiid Designer to assist you with common design tasks. Specifically, we will introduce the following concepts:

- 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 Teiid 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.

7.2. GUIDE VIEW

In this section, we introduce the Guides View 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. Edit VDB
7. 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 **Teiid Designer**. Next we will preview the database contents. Finally we will define a VDB and then deploy it to a running **JBoss Data Virtualization Server** to execute.

7.3. MODEL A JDBC SOURCE

This section shows how to model a JDBC Source, using the **`Guide View`** action set. For this example, we will connect to a MySQL database, 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, on the main menu, click **`Window >Show View > Other...`** and then click **`Teiid Designer > Guides`** view in the dialog.

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

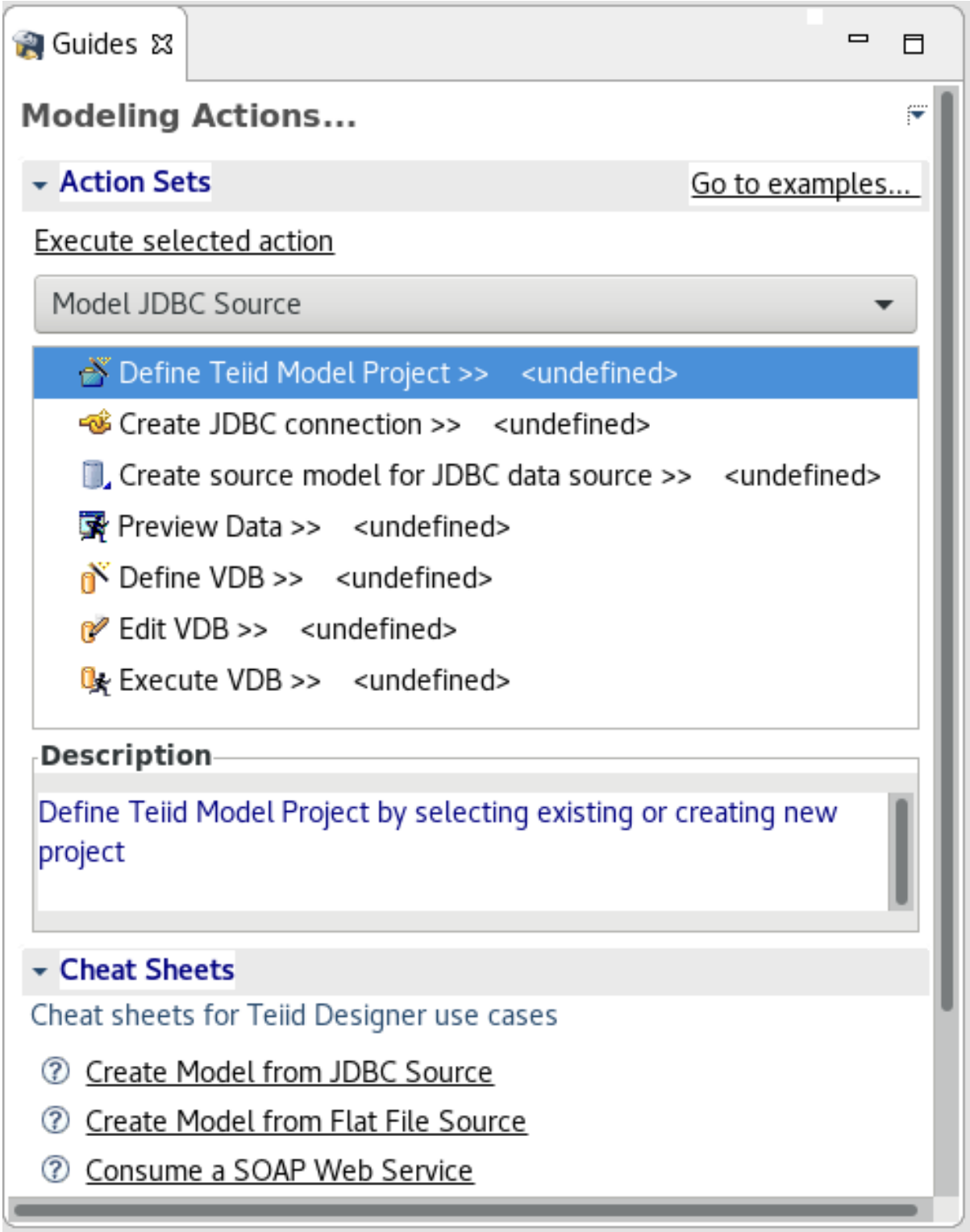


   Figure 7.1. Guides View

2. **Define Teiid Model Project**

NOTE

First, you can see **`Define Model Project`** dialog where you can choose existing project by clicking **`Existing...`** button or create new project by clicking **`New...`** button. Next steps describe the creation of new Teiid Model Project.

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

   Enter a project name, for example, MyProject in the Name field. Then click **`Next`**.

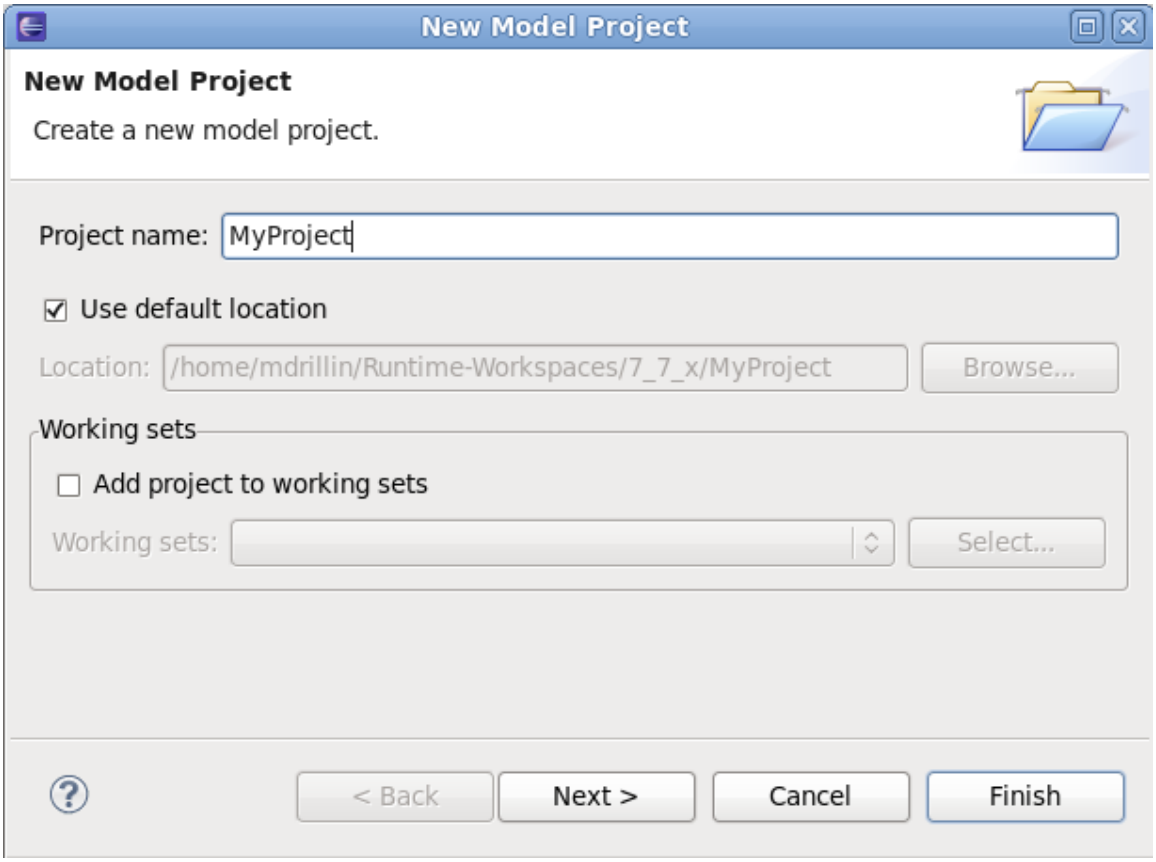


Figure 7.2. New Project Wizard

b. The next page of the wizard is **`Project References`** which allows you to select other projects which this project will reference. Click **`Next`**.

c. The next page of the wizard is shown below:

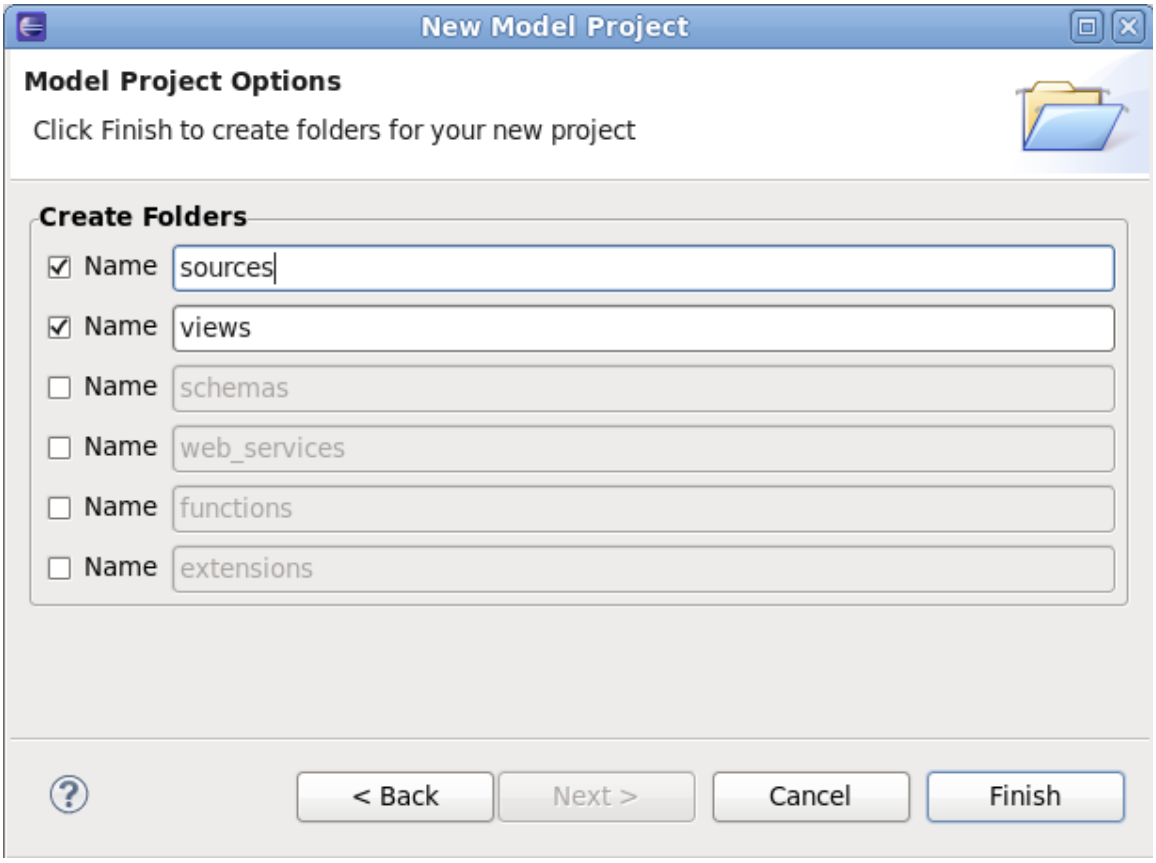


Figure 7.3. New Project Folders

d. Under **`Create Folders`**, clear **`schemas`** and **`web_services`**. We will not need them for this example. Now, click **`Finish`** to exit the wizard. The project has now been created.

Your Model Explorer view will look like this:

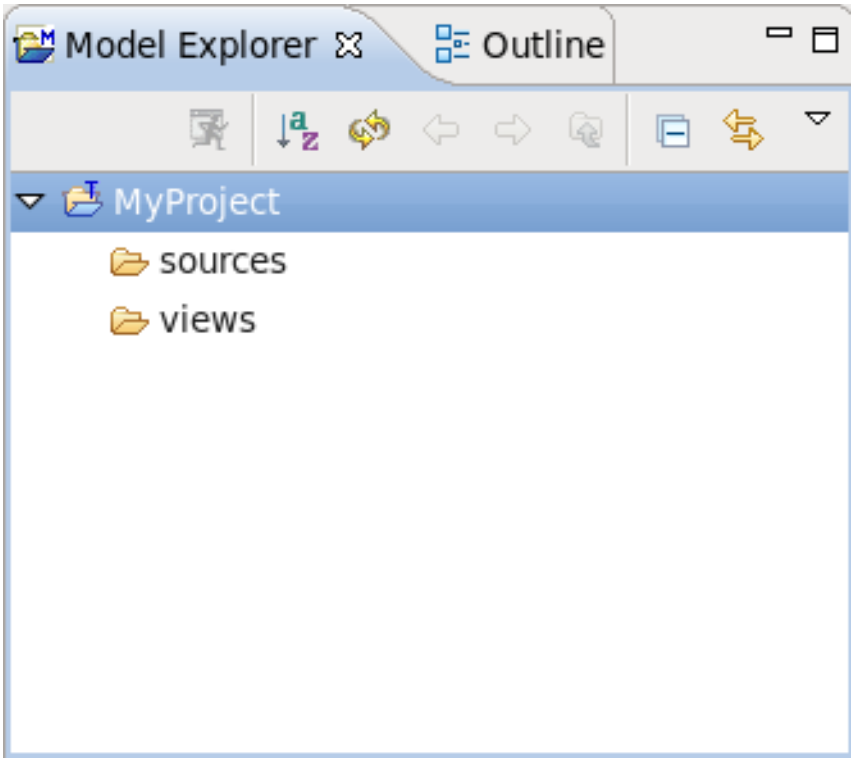


Figure 7.4. Model Explorer

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:

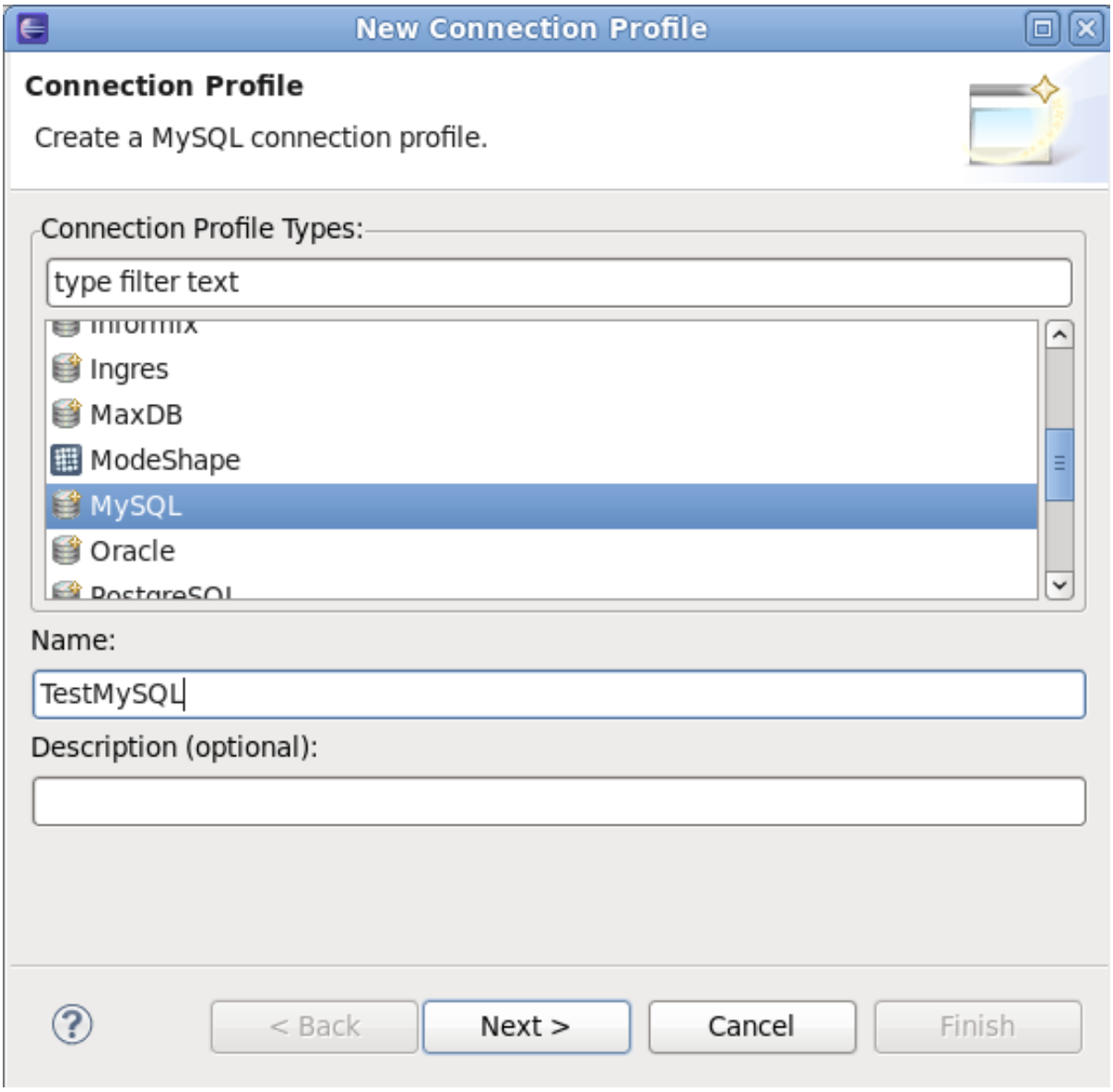


Figure 7.5. Connection Profile Name and Type

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

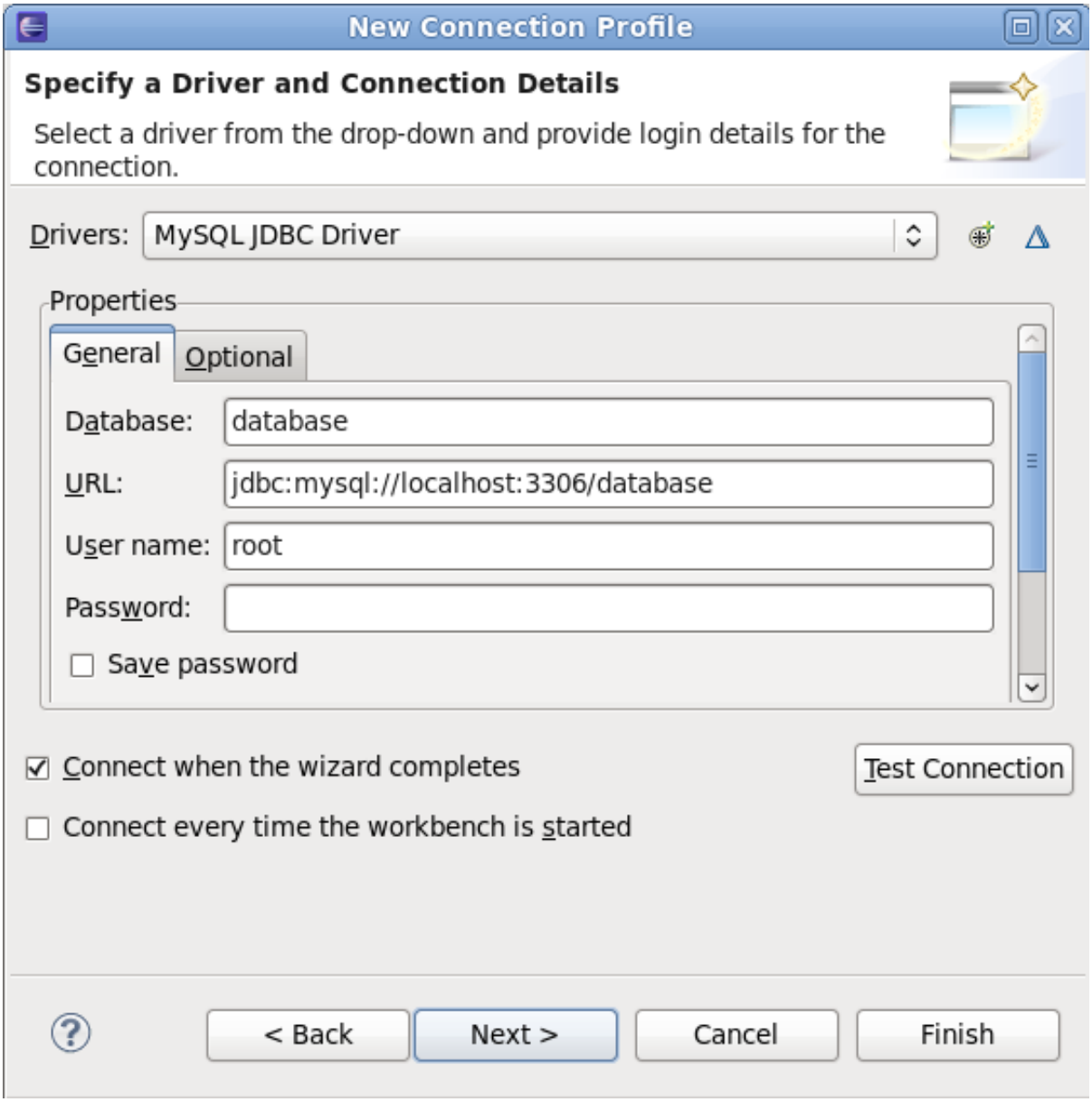


Figure 7.6. Connection Profile properties

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**

   a. The **`Create source model for JDBC data source`** action will now utilize the **`Connection profile`** that you have 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:

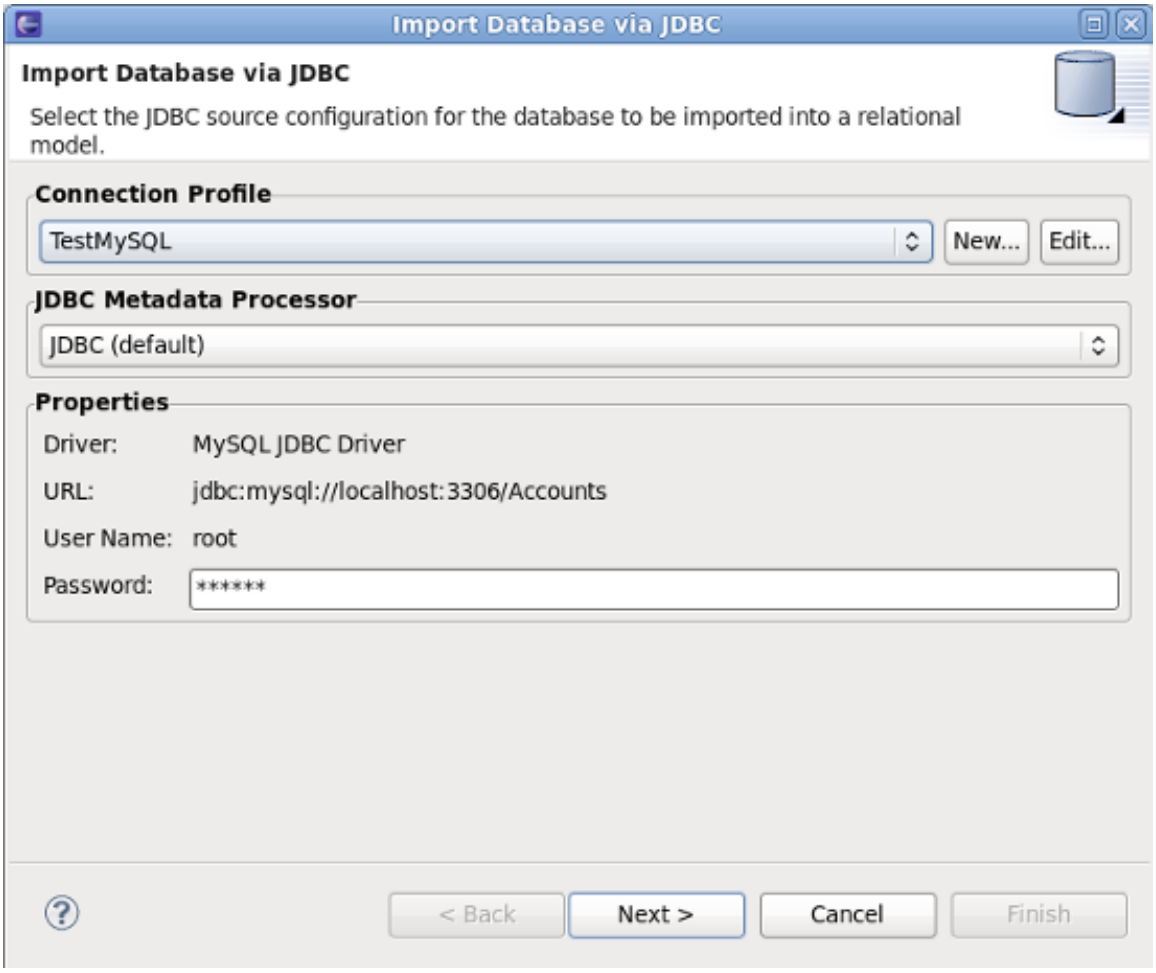


Figure 7.7. Select Connection Profile

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

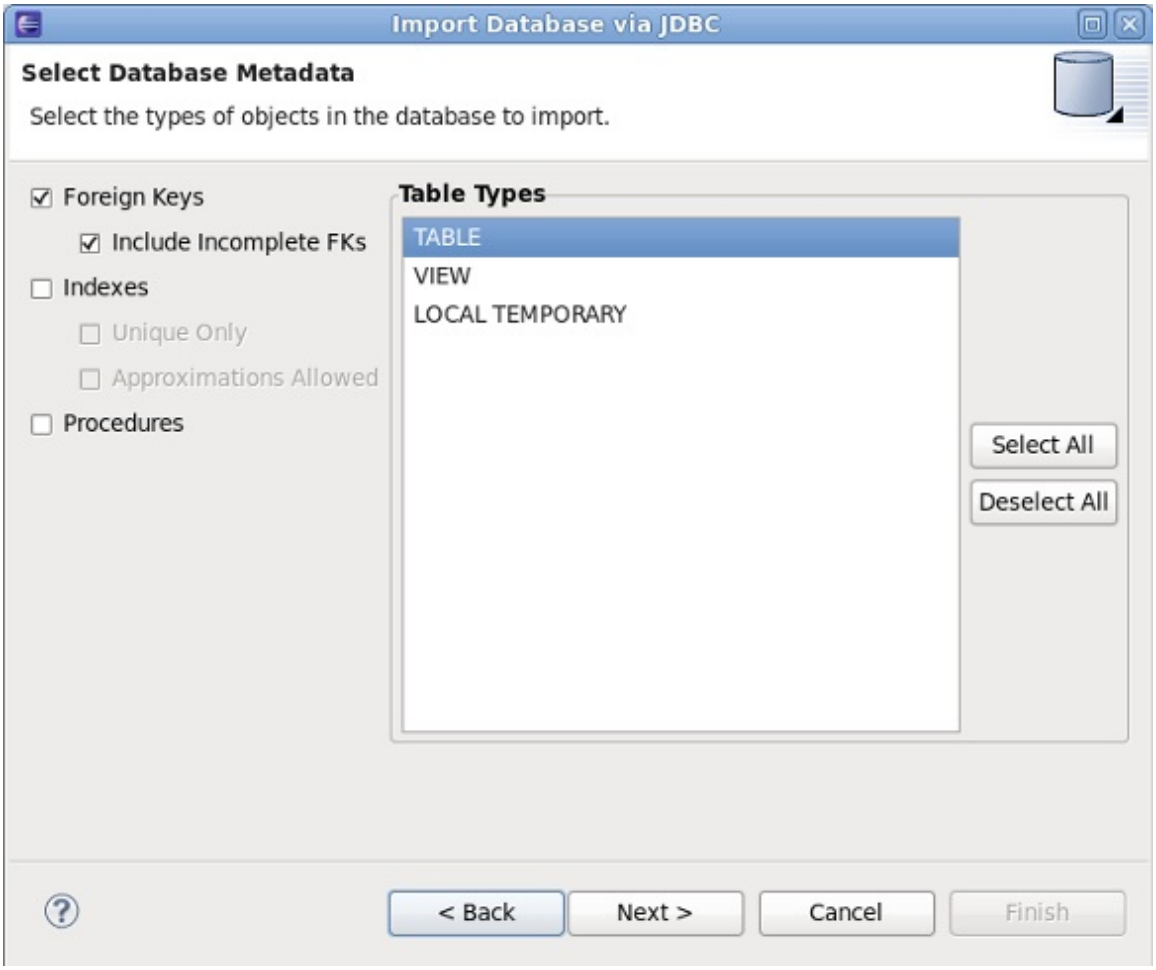


Figure 7.8. Select Database Metadata

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

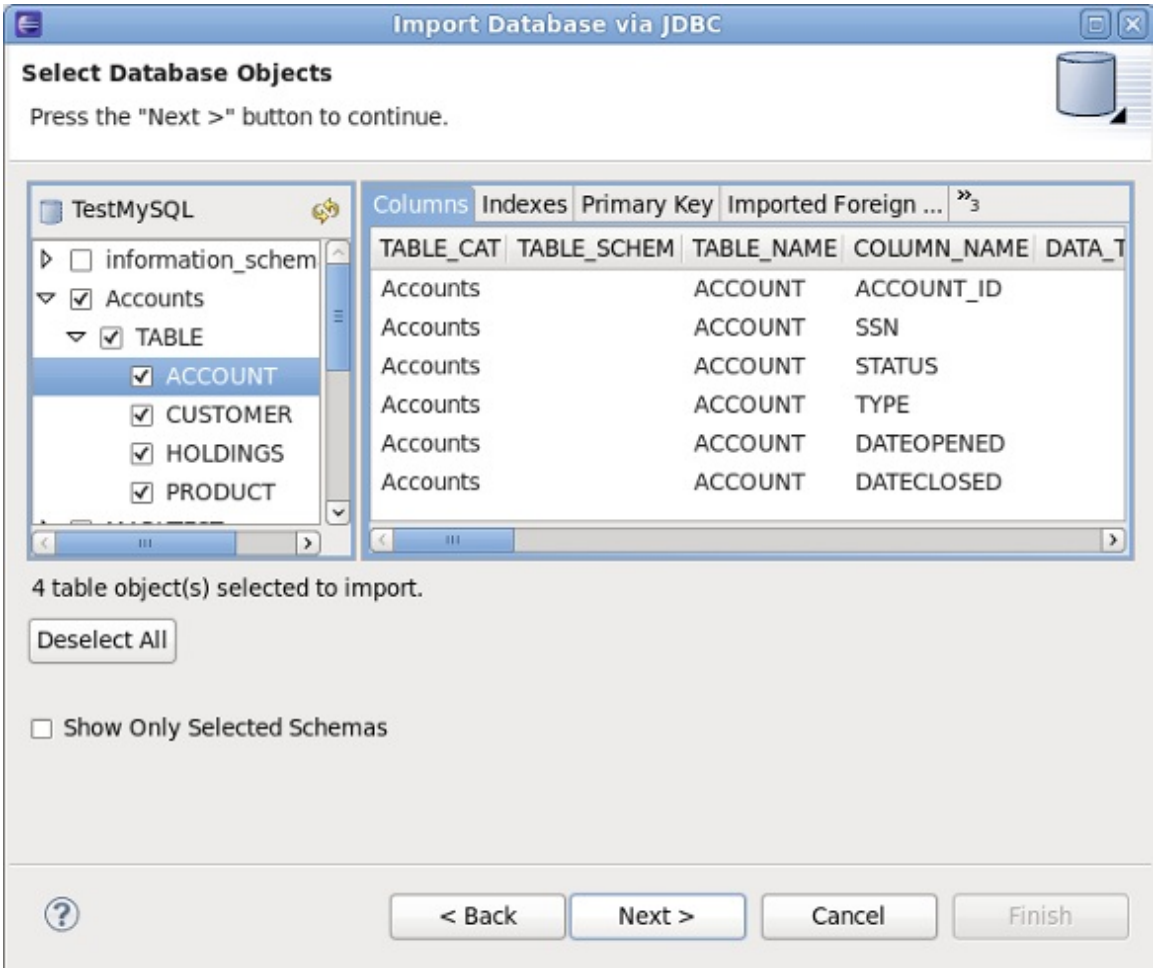


Figure 7.9. Select Database Objects

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

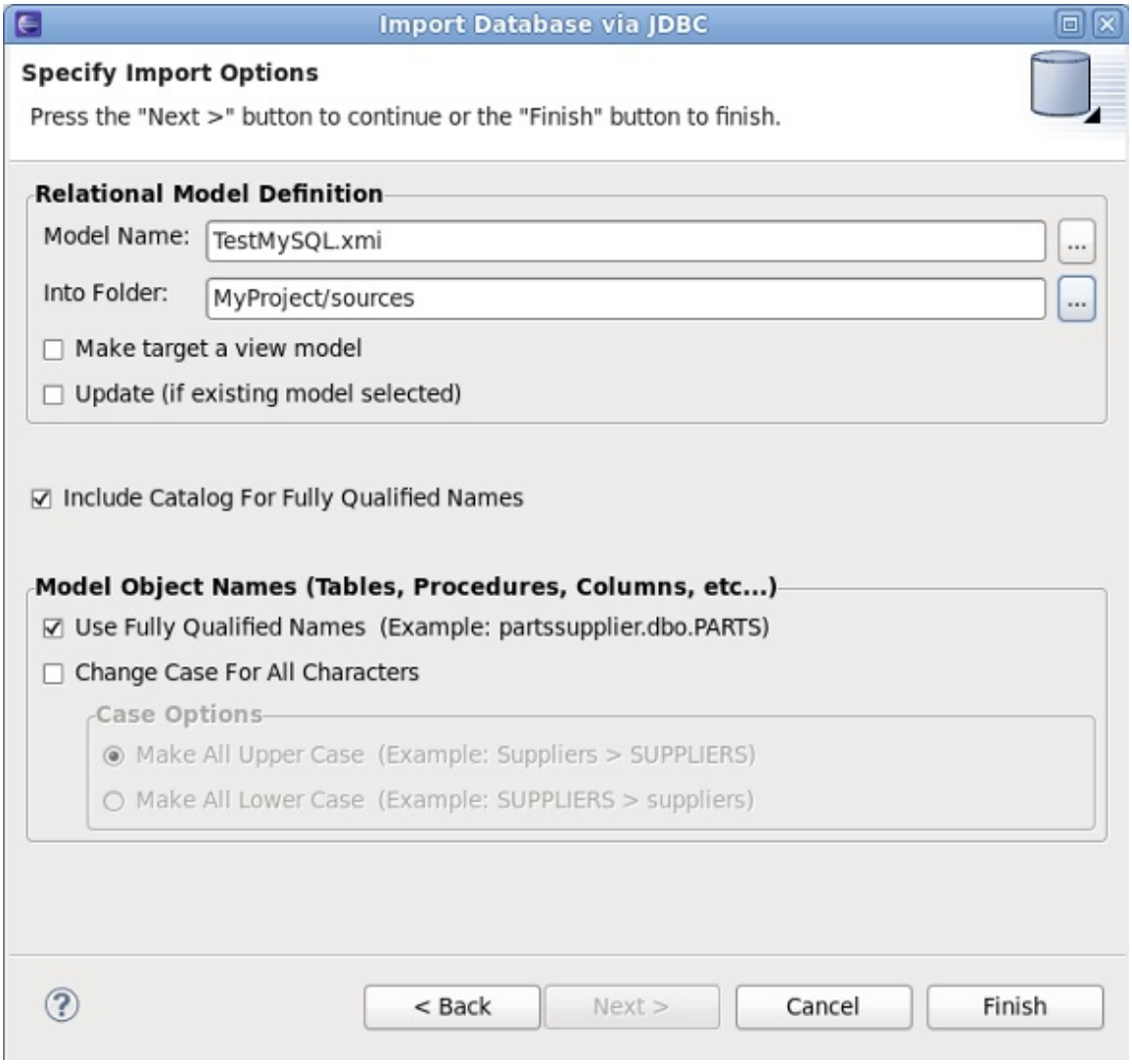


Figure 7.10. Import Options

e. 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 will look like this:

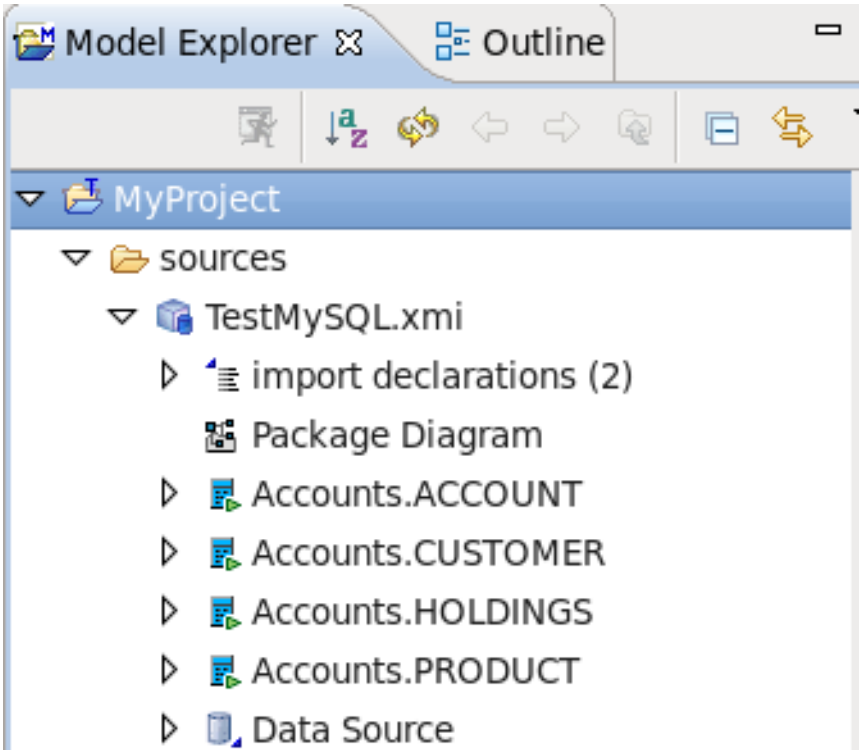


Figure 7.11. Model Explorer

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

   a. 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`**).

   b. In the first dialog, click **`...`** button and then select the source table you want to preview, as shown below:

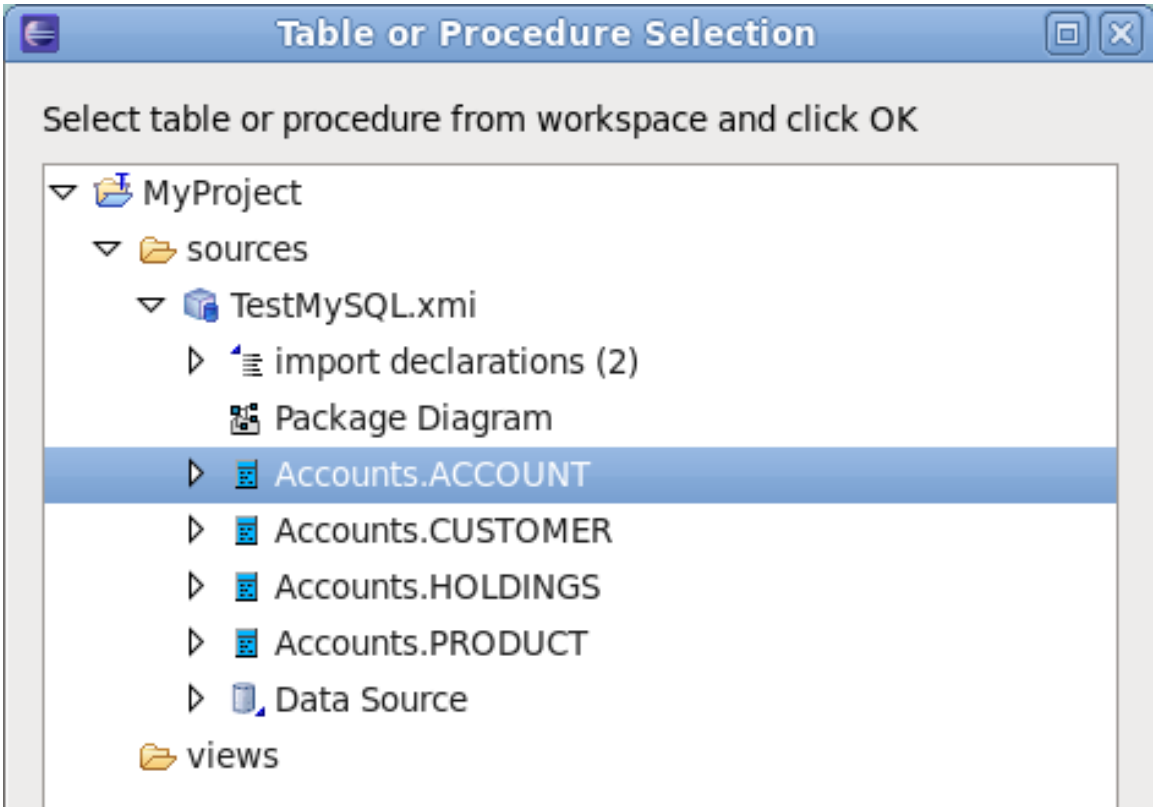


Figure 7.12. Select Preview Table

c. After selecting the table, click **`OK`** button in both dialogs. After that, the **`Custom Preview Data`** dialog is shown when you can see SQL query and VDB XML Content. Click **`OK`** for running query.

NOTE

When the server is not running during creating source model, you can get **`Data Sources Missing`** dialog. Just click to **`Yes`** for creating a new data source.

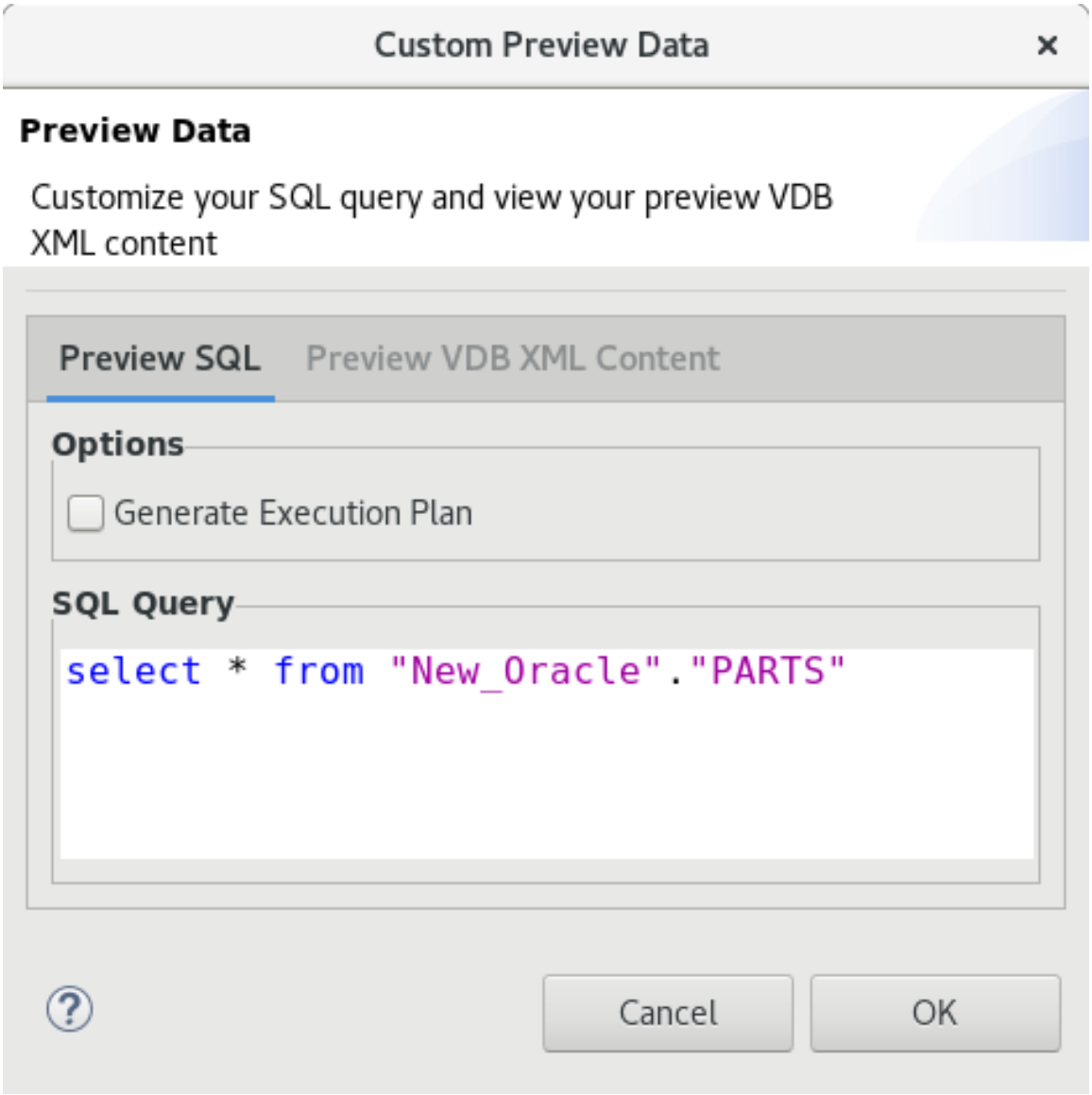


Figure 7.13. Custom Preview Data

d. Now, the preview results will be displayed:

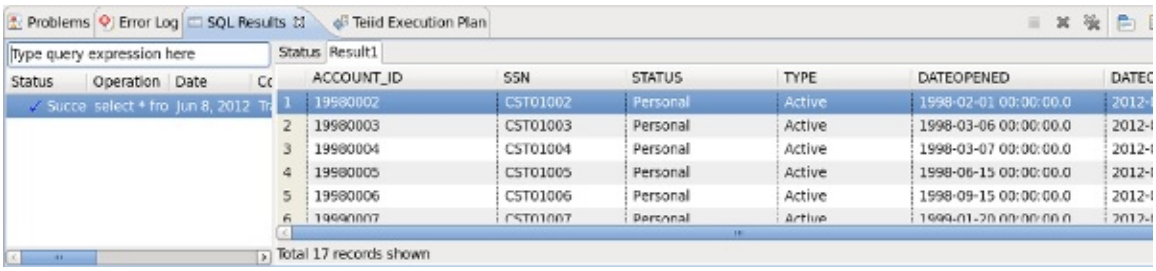


Figure 7.14. Preview Results

6. **Define VDB**

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

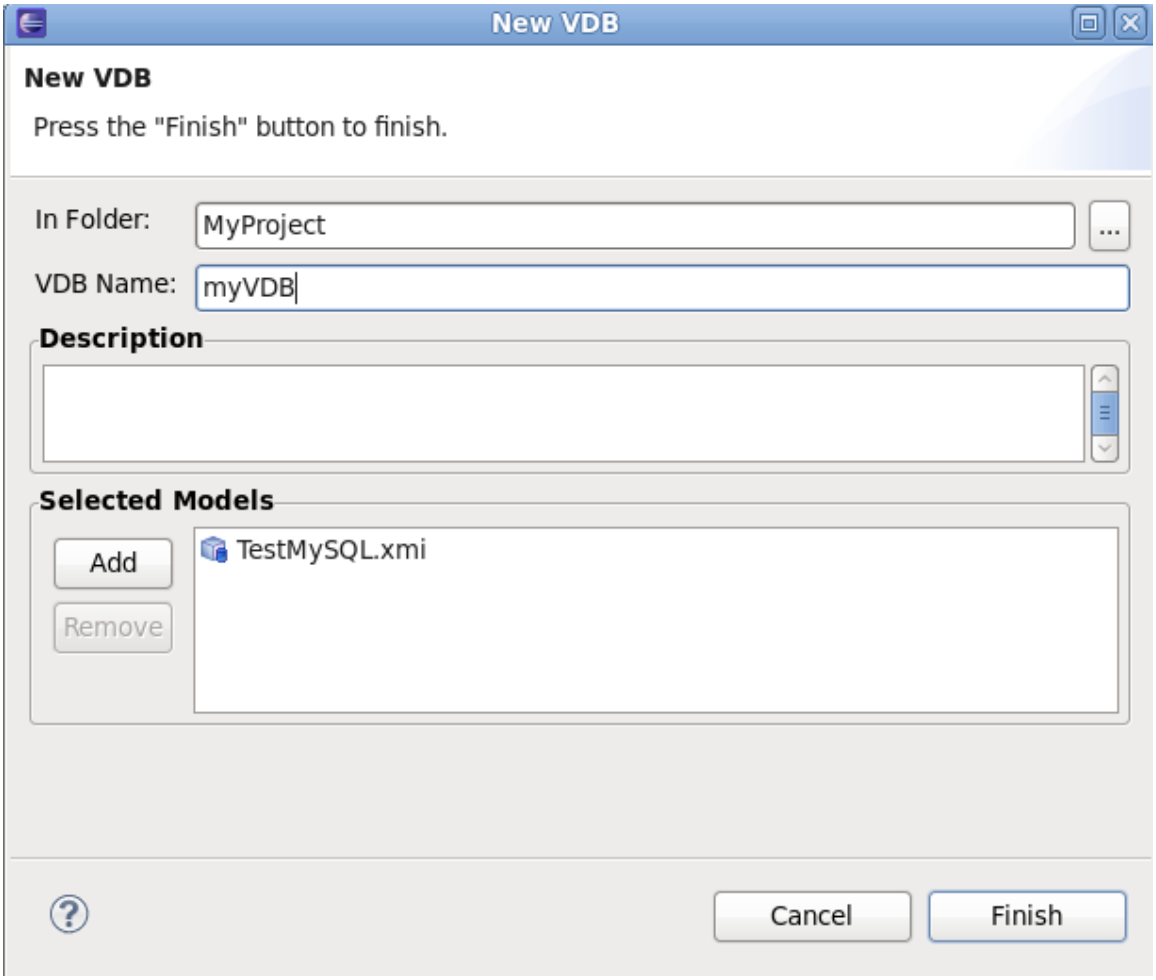


Figure 7.15. New VDB

b. 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.

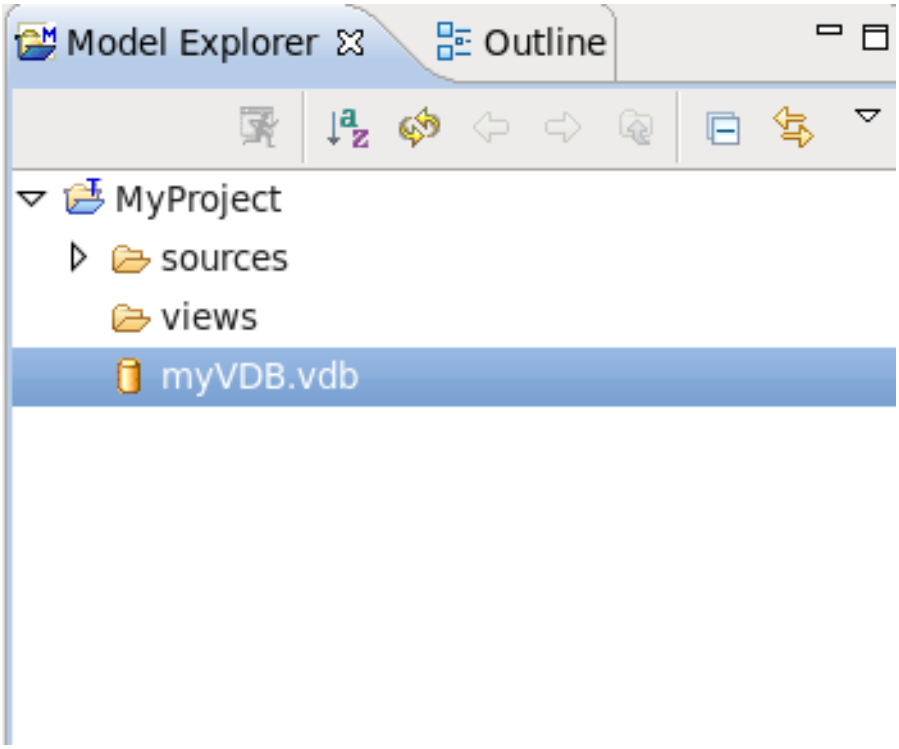


Figure 7.16. Model Explorer

7. **Edit VDB**
   The **`Edit VDB`** action opens **`VDB editor`** for editing VDB.

8. **Execute VDB**
   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 perspective will switch to the **`Database Development`** perspective. You can now run queries against the VDB, as show in the following example:

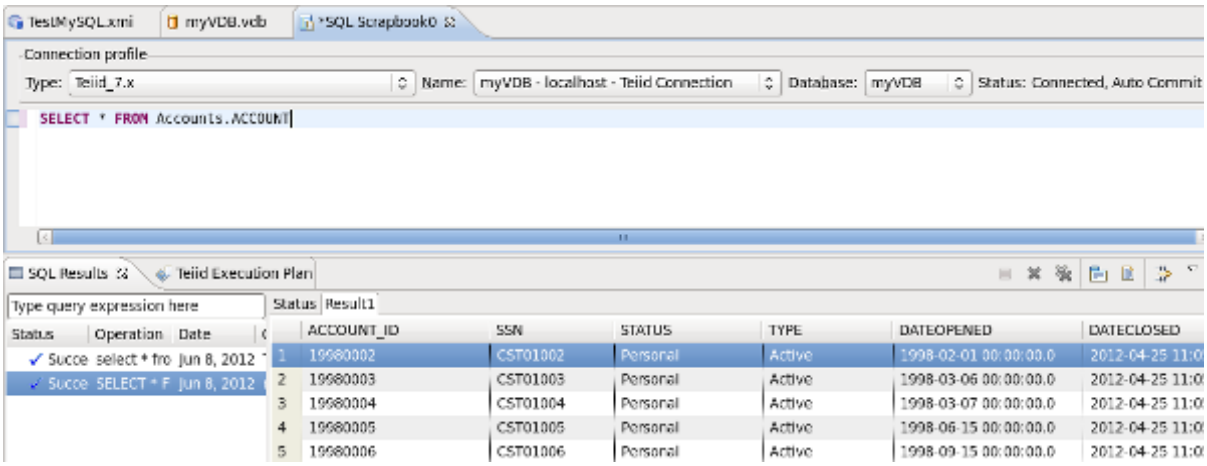


Figure 7.17. Execute VDB Example

7.4. CHEAT SHEETS

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.

7.5. 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:

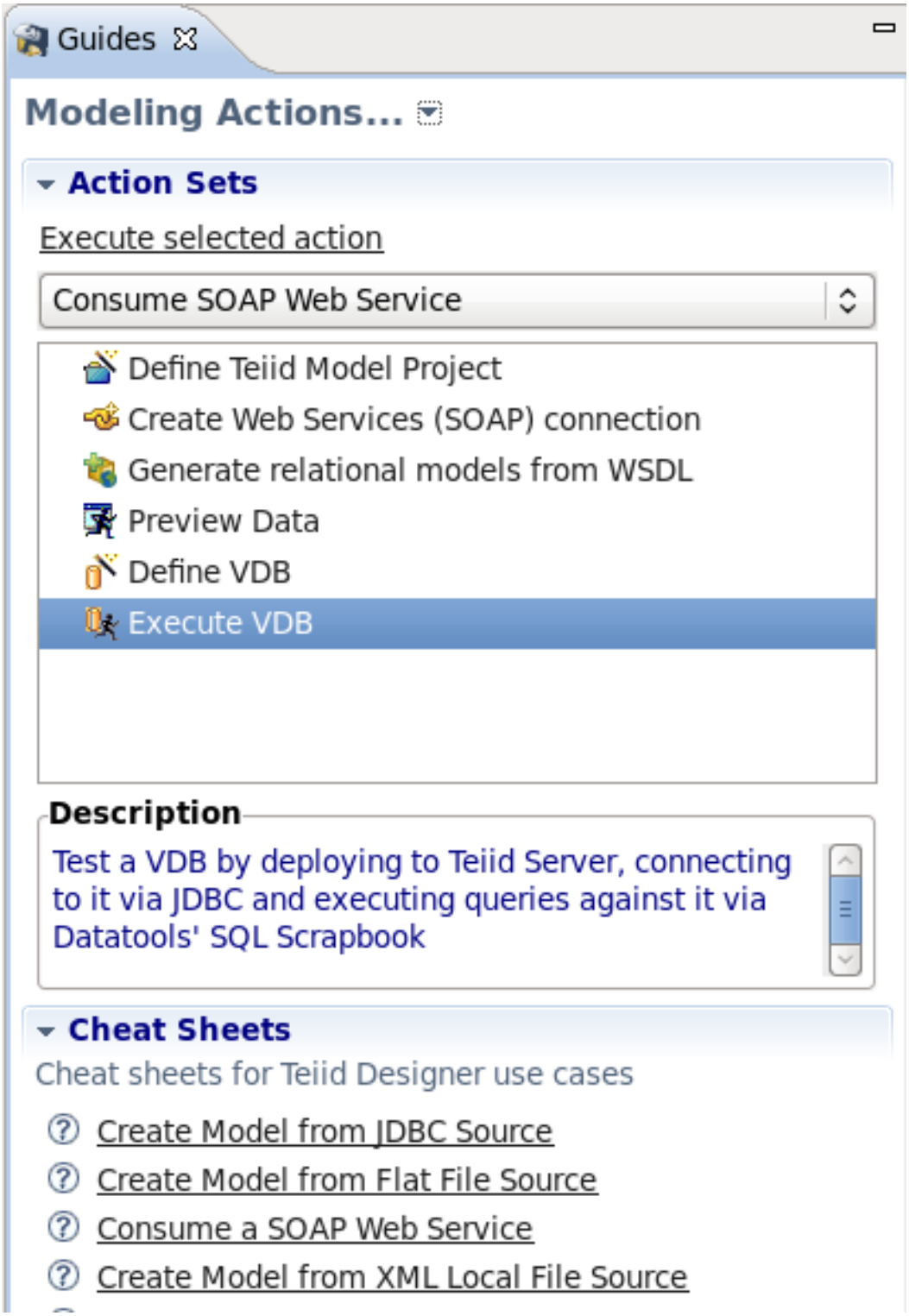


Figure 7.18. Guides View

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:

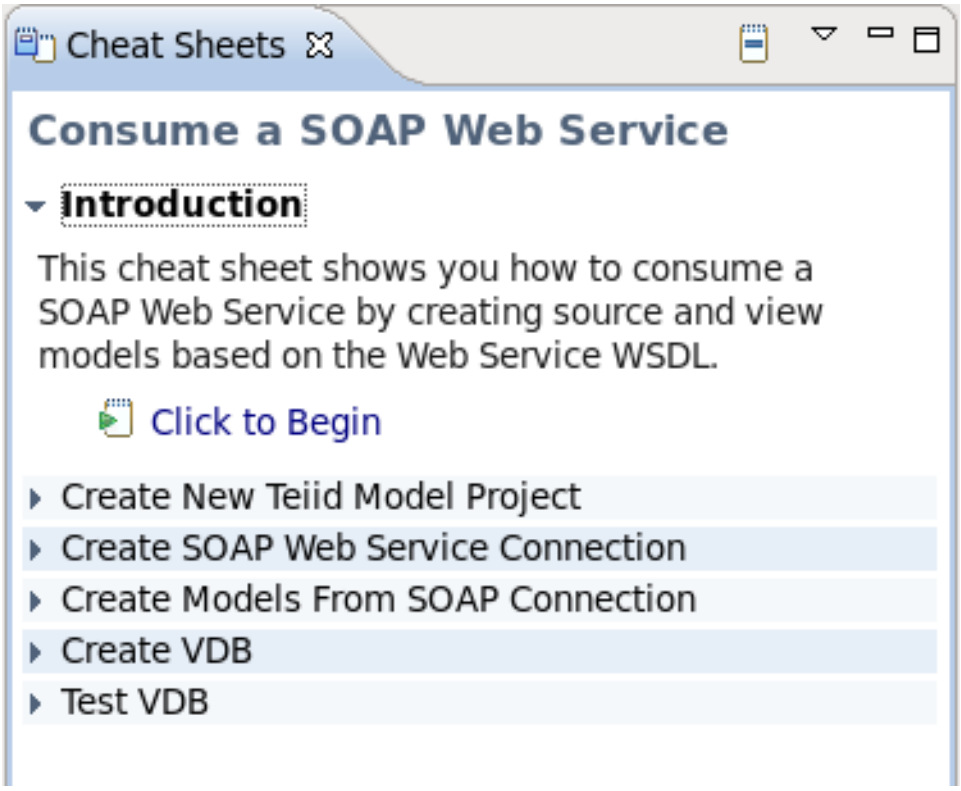


Figure 7.19. Consume SOAP Web Service Cheat Sheet

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 7.20. 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 is not needed any action, just click **`Next`**. On the third 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:

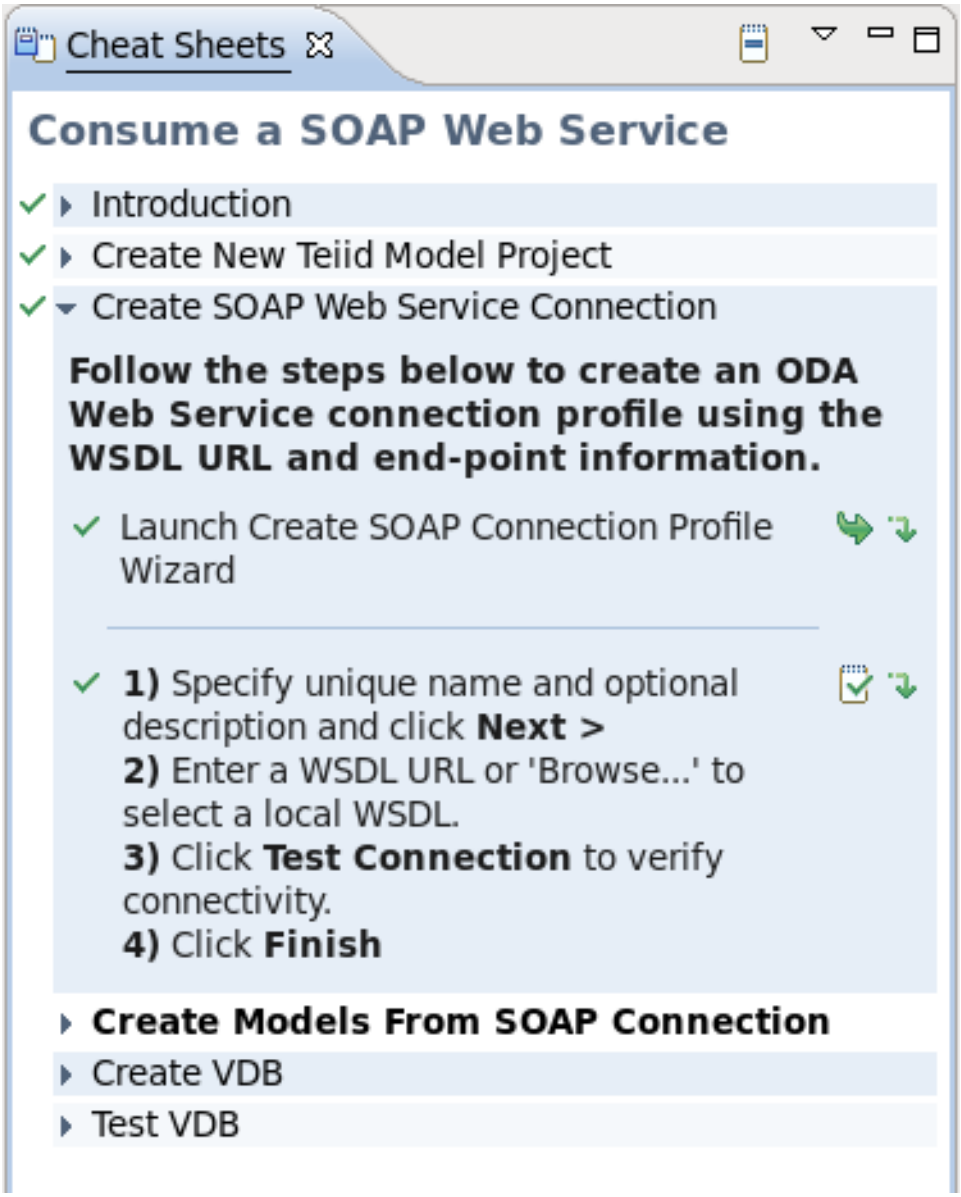


Figure 7.21. Create SOAP Connection Profile

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

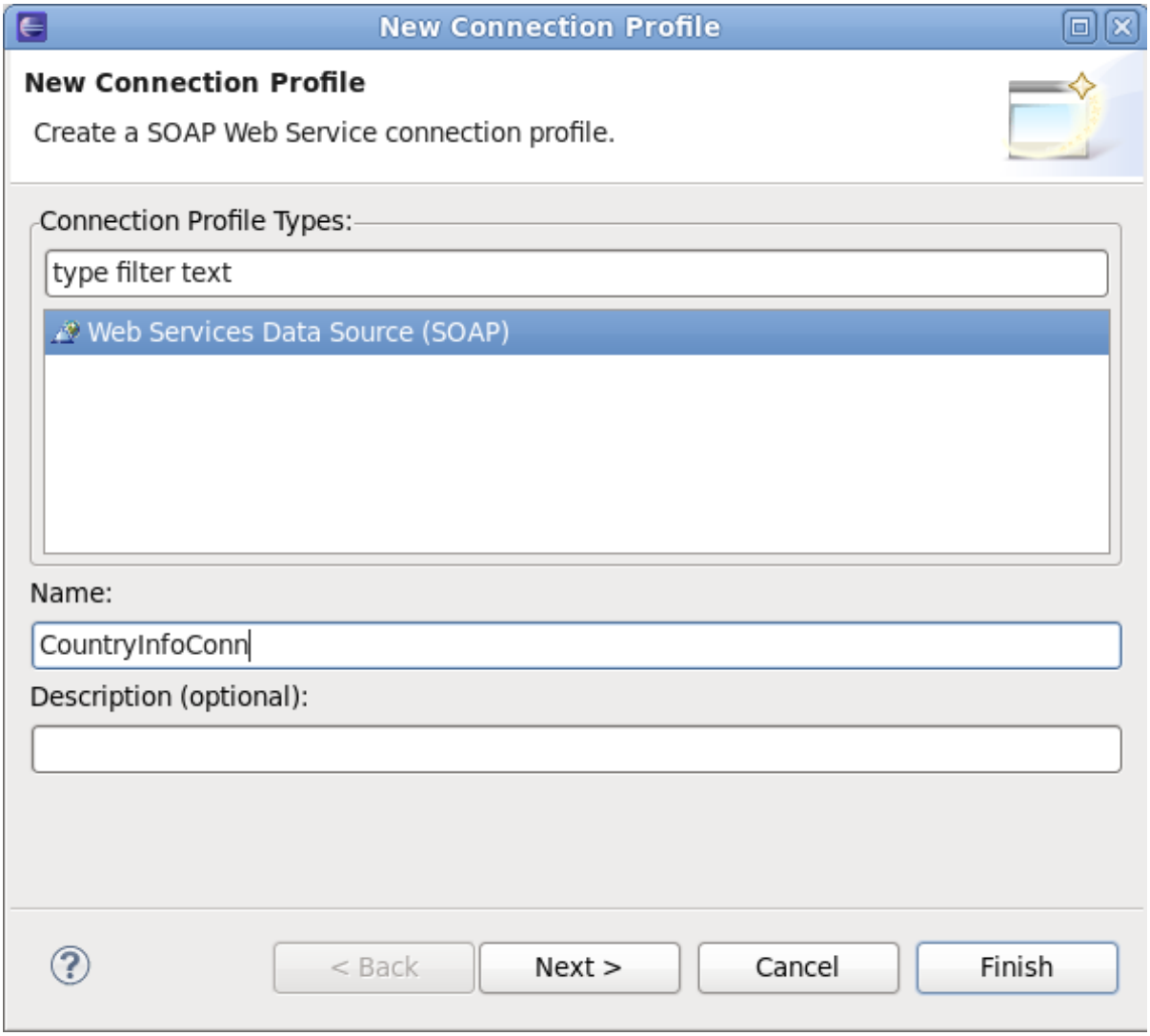


Figure 7.22. Create SOAP Connection Profile

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:

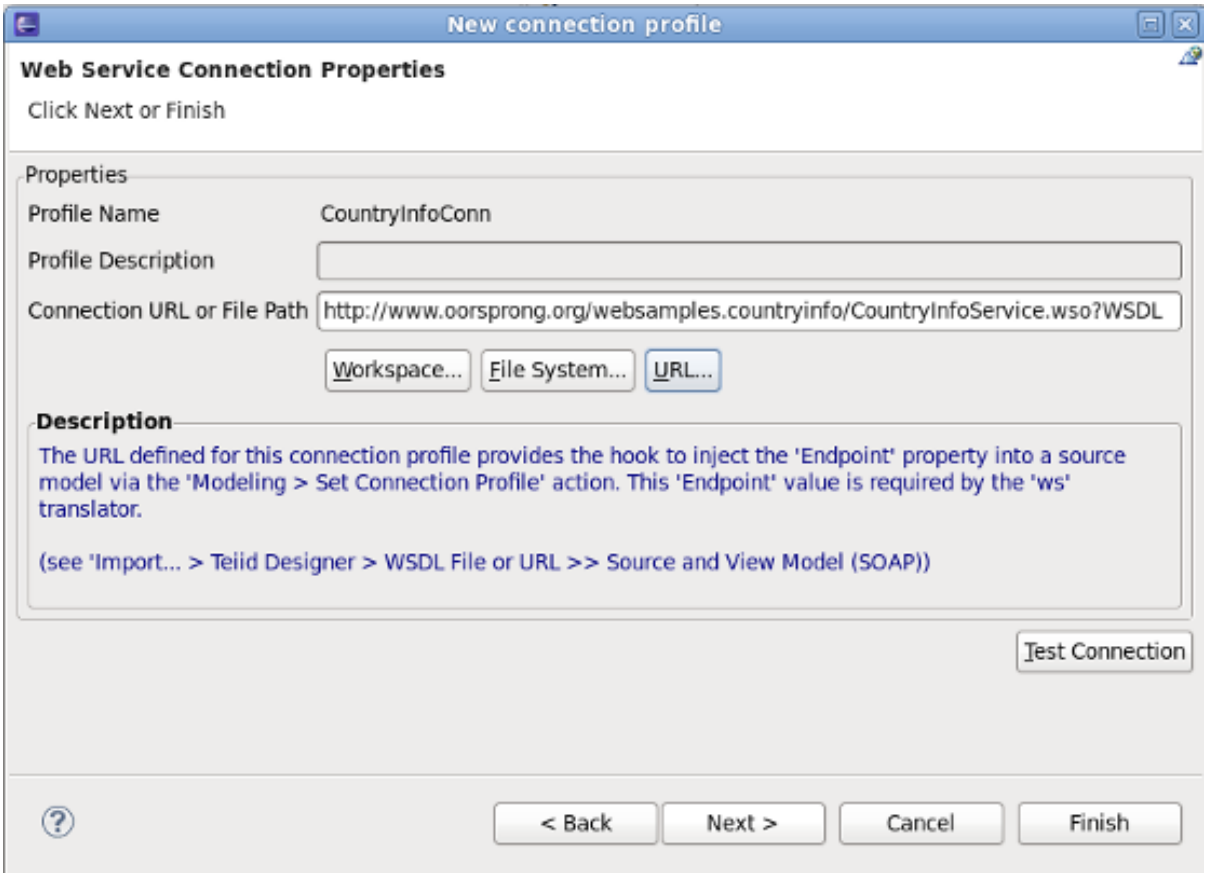


Figure 7.23. SOAP Connection Properties

The connection profile properties are entered on this page. Click 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 dialog. After that, click **`Next`** and choose End Point. For our example choose CountryInfoServiceSoap12 and click **`Finish`** to complete 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:

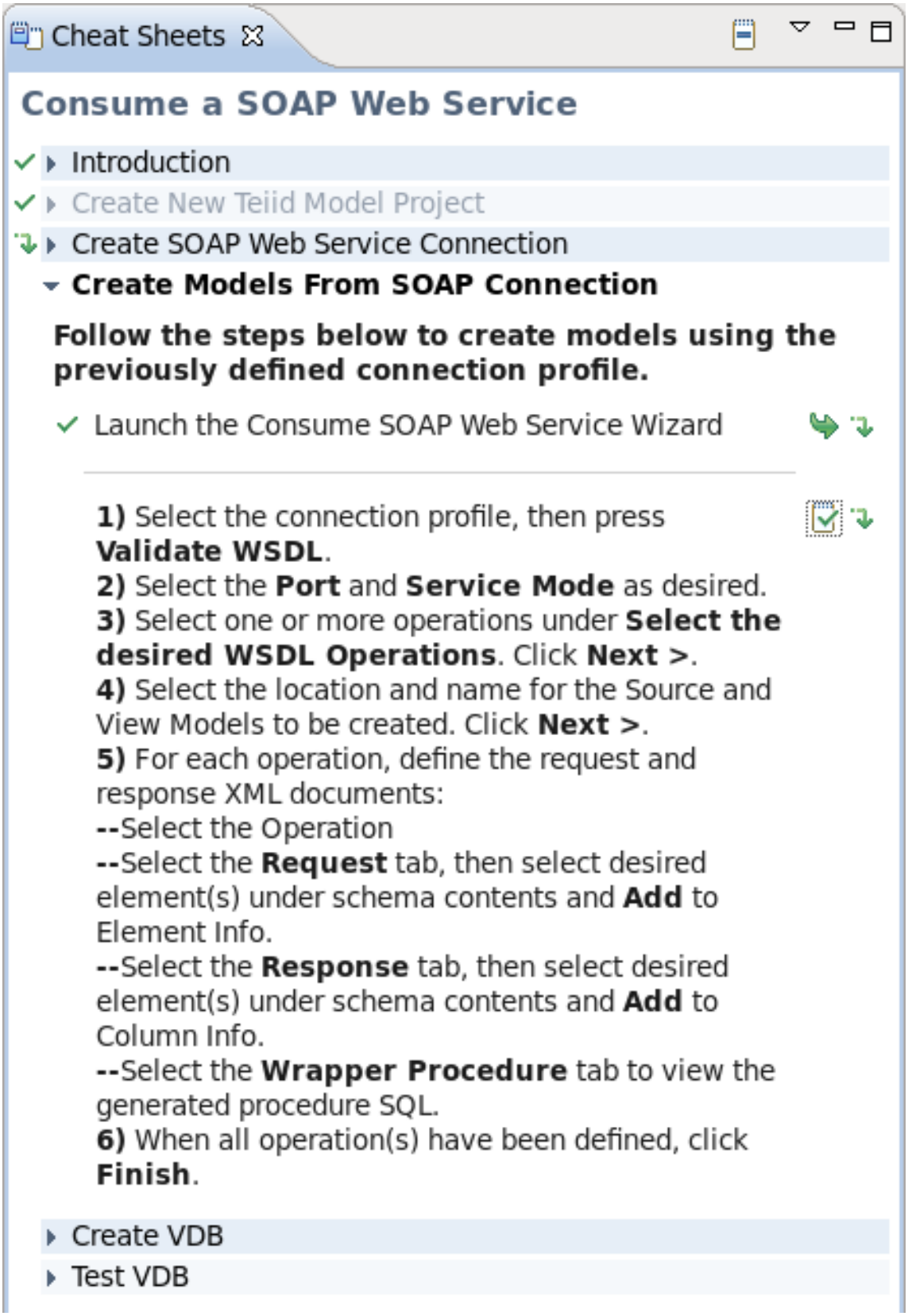


Figure 7.24. Create Models from SOAP Connection

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

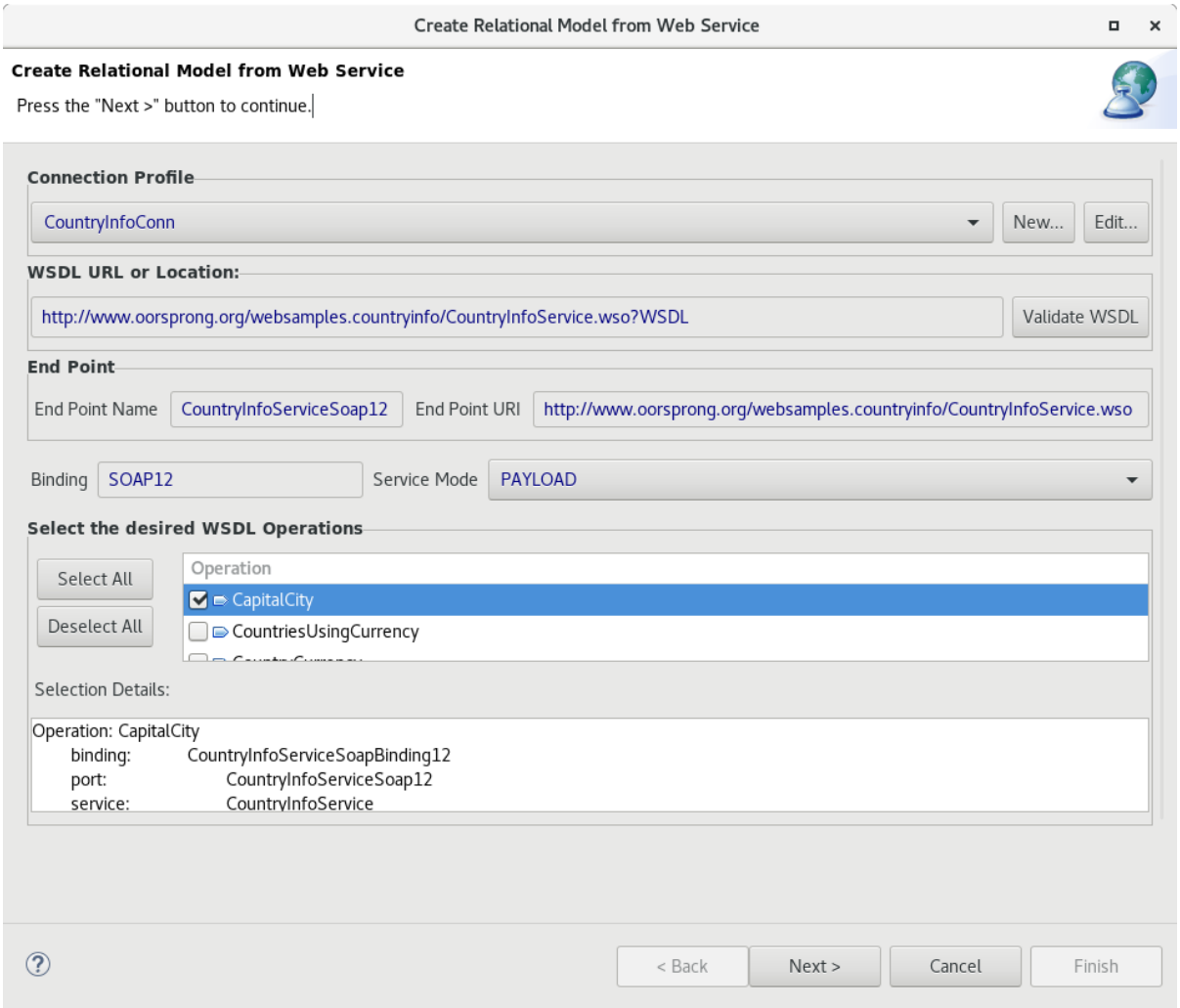


Figure 7.25. Consume SOAP Wizard

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:

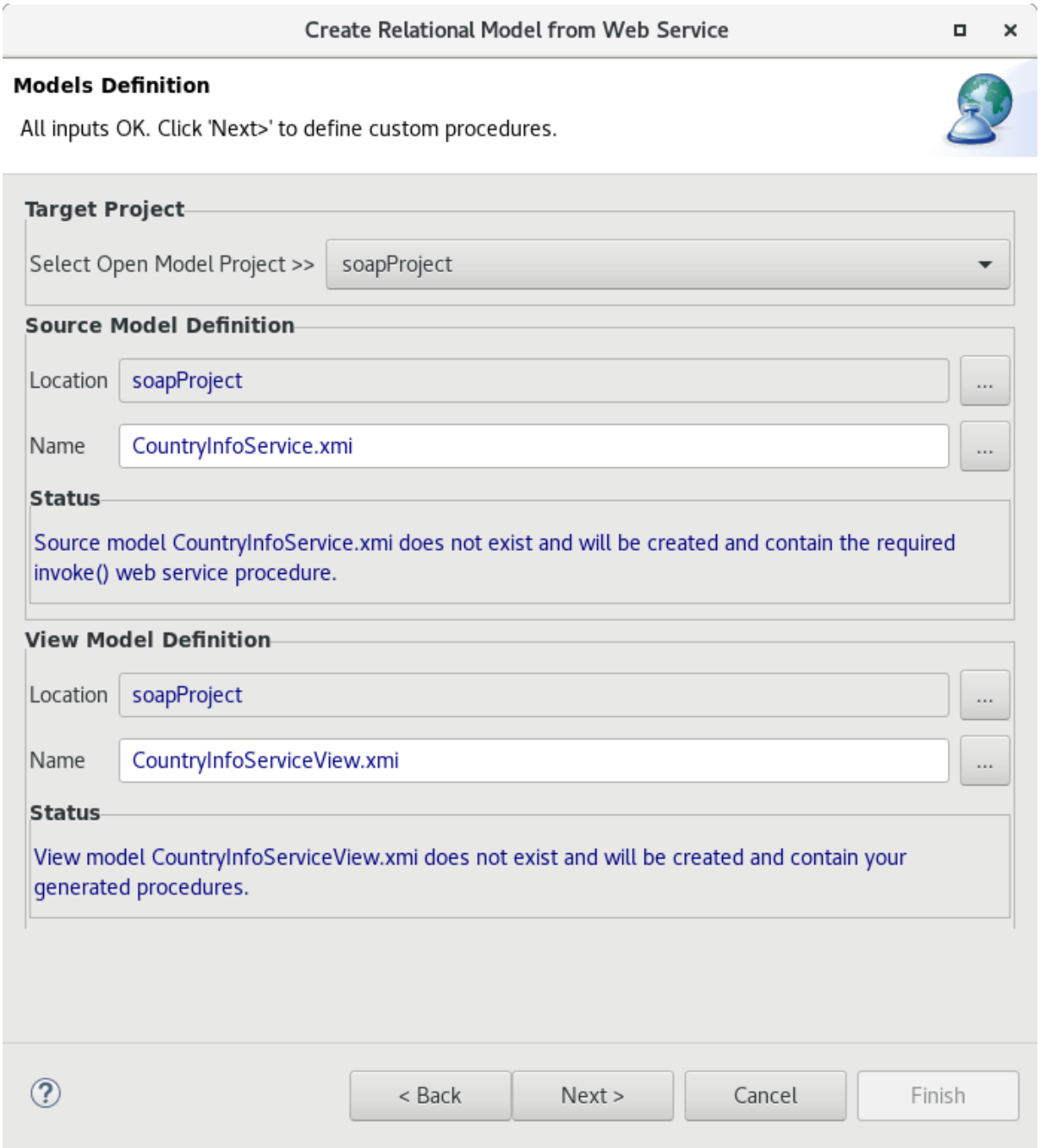


Figure 7.26. Consume SOAP Wizard

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:

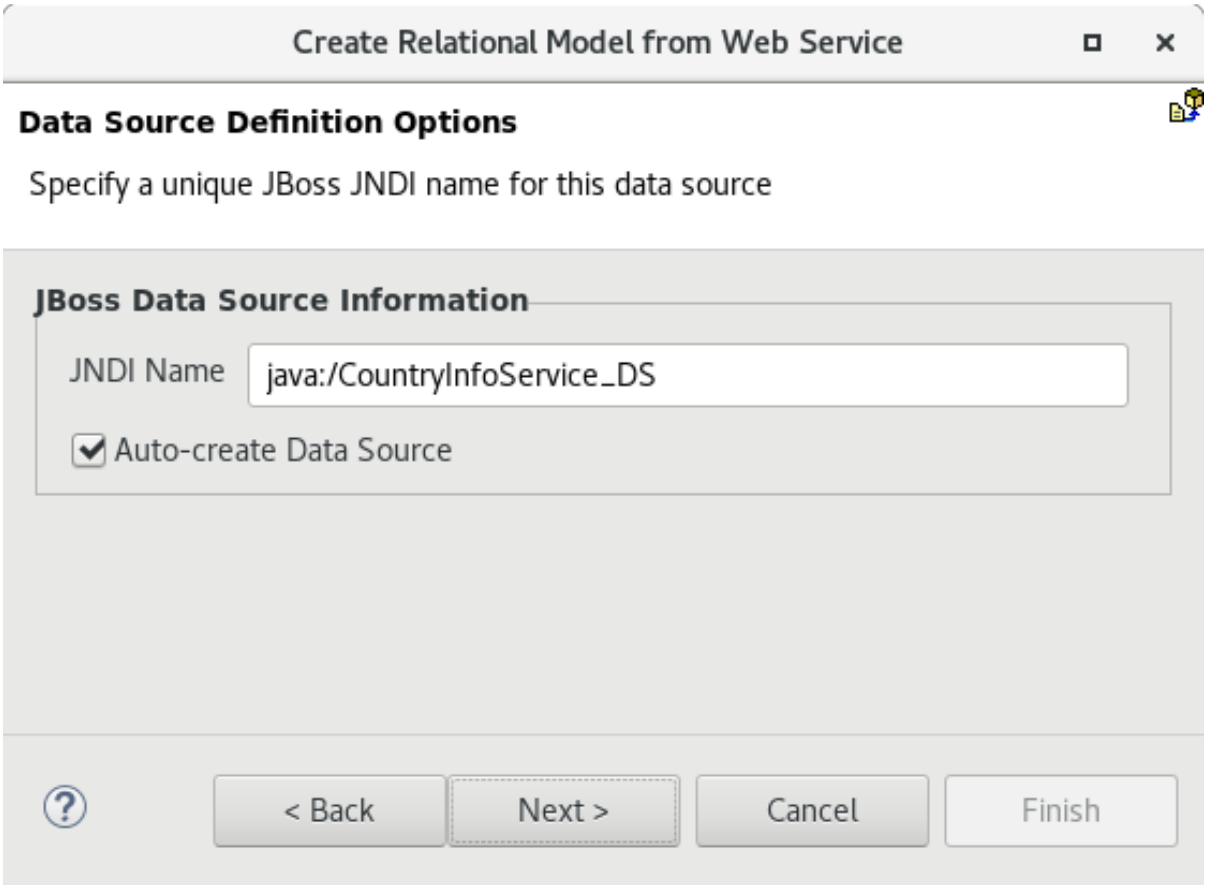


Figure 7.27. Consume SOAP Wizard

On the **`Data Source Definition Options`** page, enter a JNDI name and click**`Next >`** to proceed to the next page, as shown below:

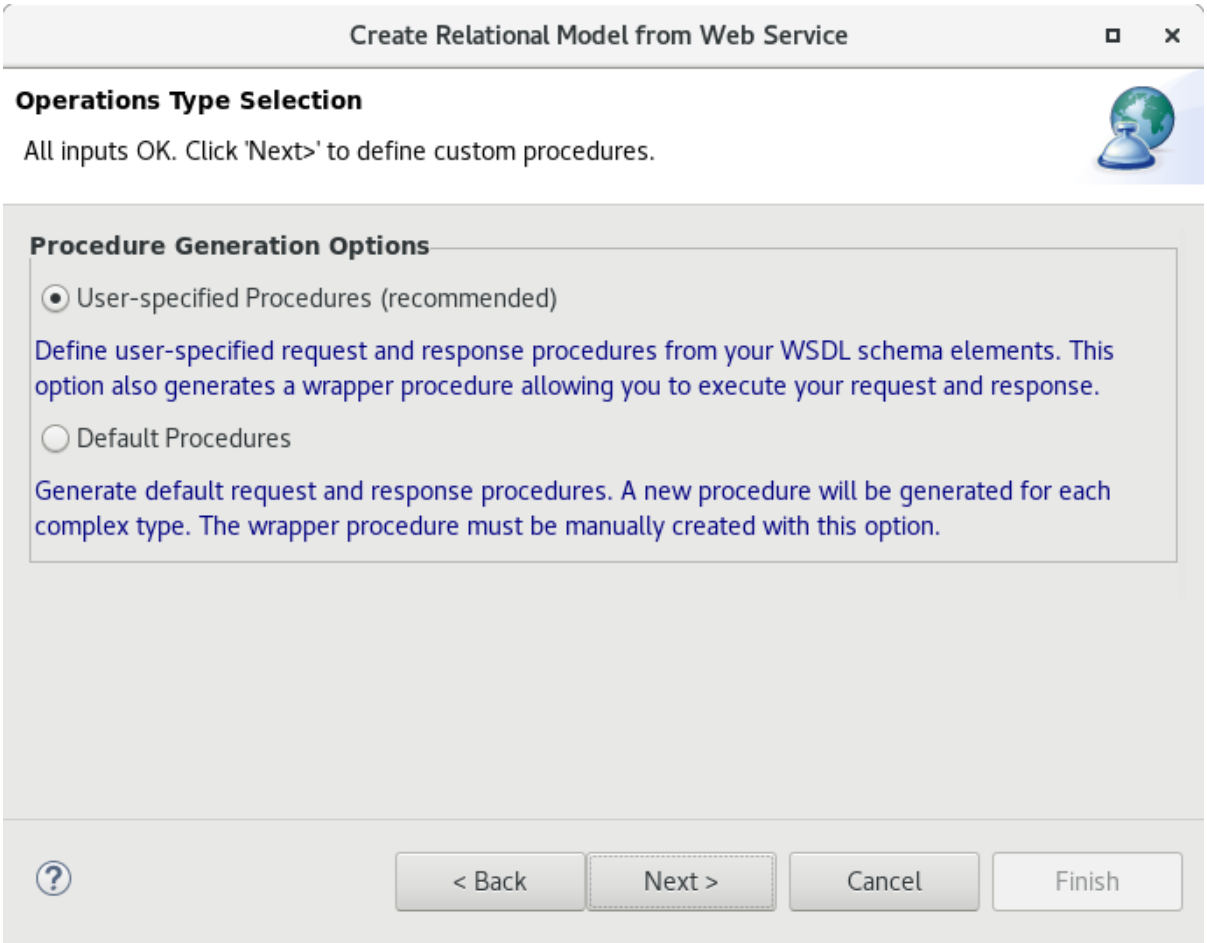


Figure 7.28. Consume SOAP Wizard

On the **`Operations Type Selection`** page, choose Procedure Generation Options and click **`Next >`** to proceed to the next page, as shown below:

NOTE

If the **`Default Procedures`** option is selected, the **`Finish`** button is available instead of **`Next >`** button.

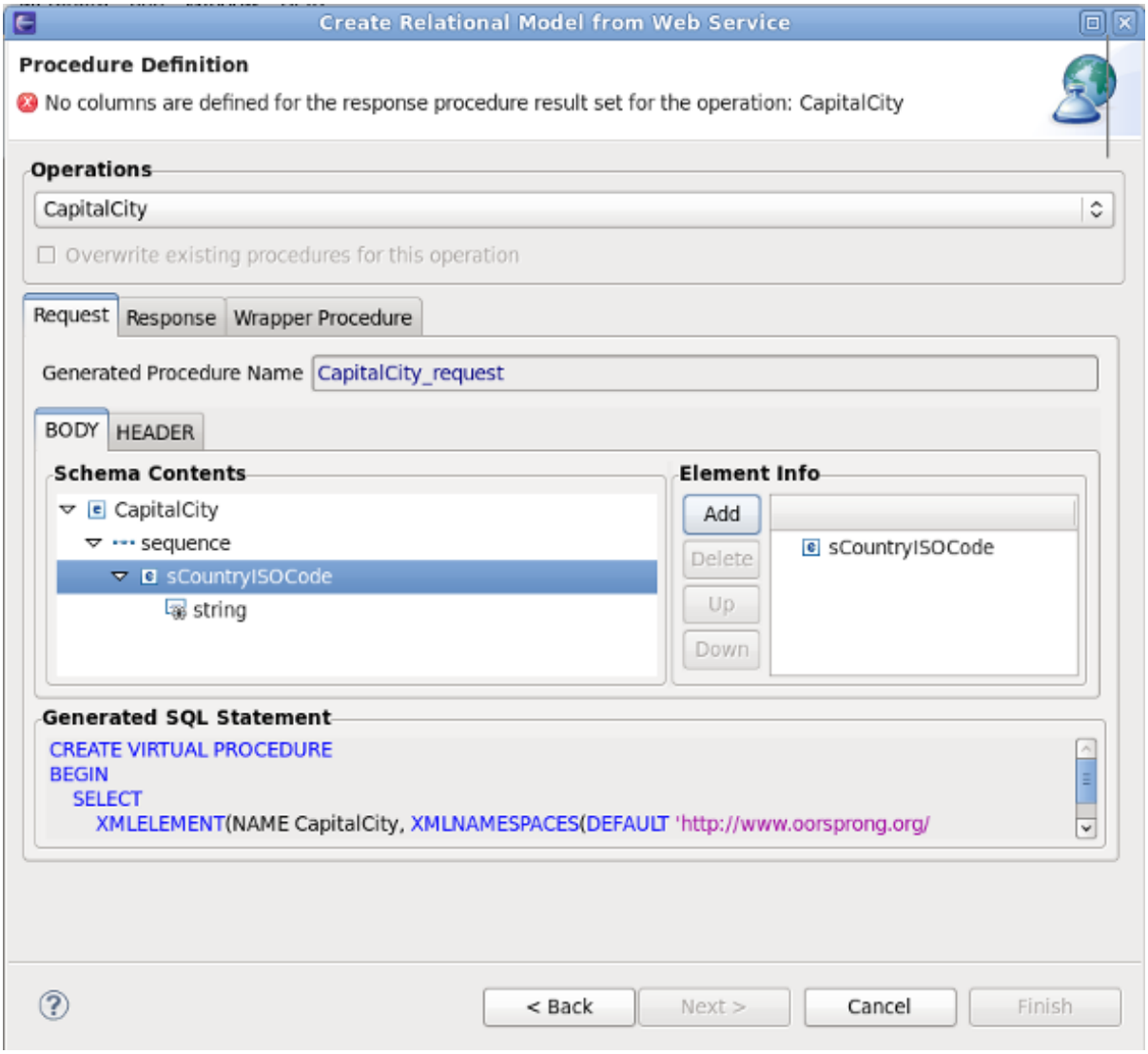


Figure 7.29. Consume SOAP Wizard

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:

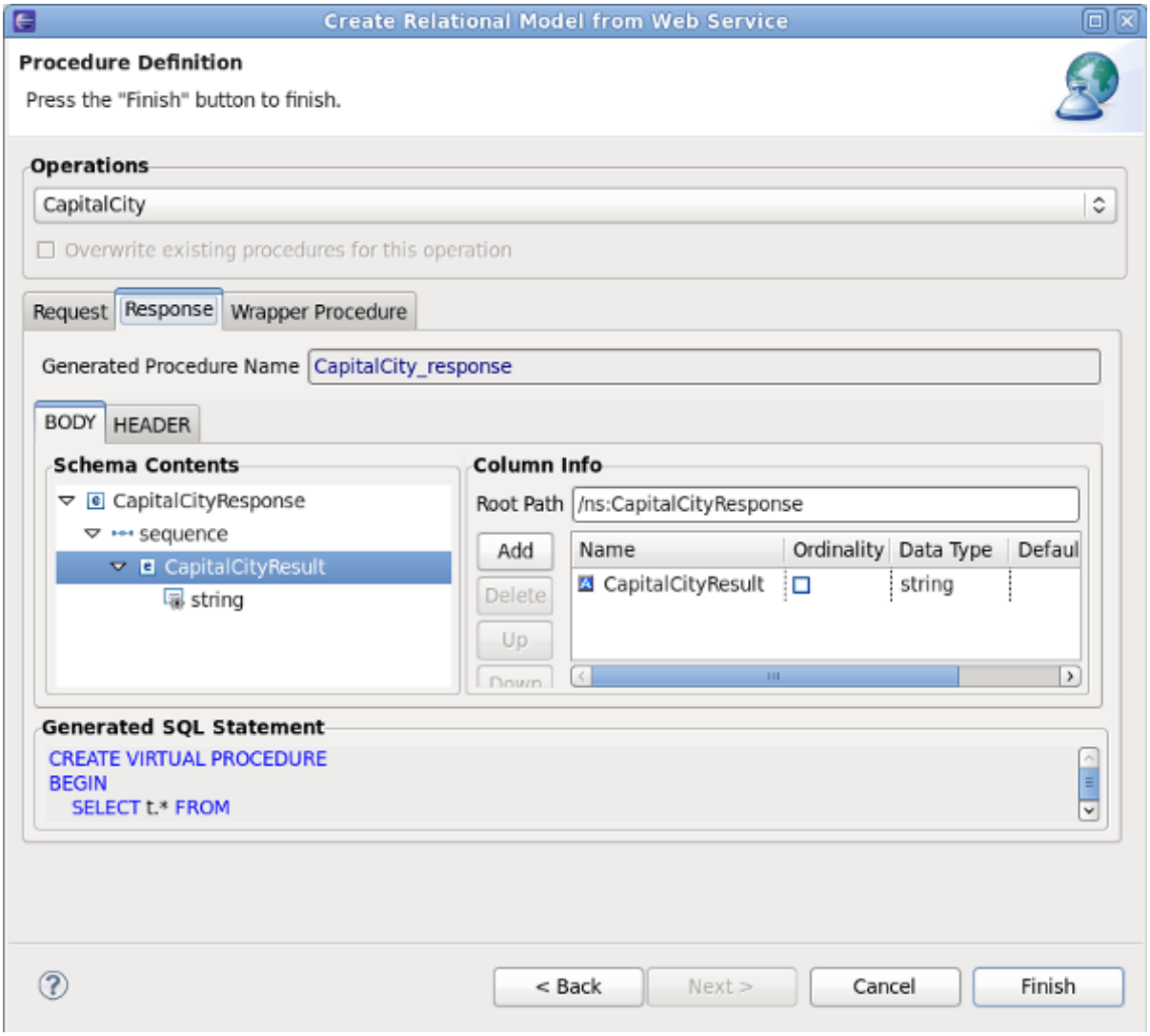


Figure 7.30. Consume SOAP Wizard

On the **`Response`** tab, select the **`Body`** sub-tab. In the **`Schema Contents`**, select the **`CapitalCityResult`**, and 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.

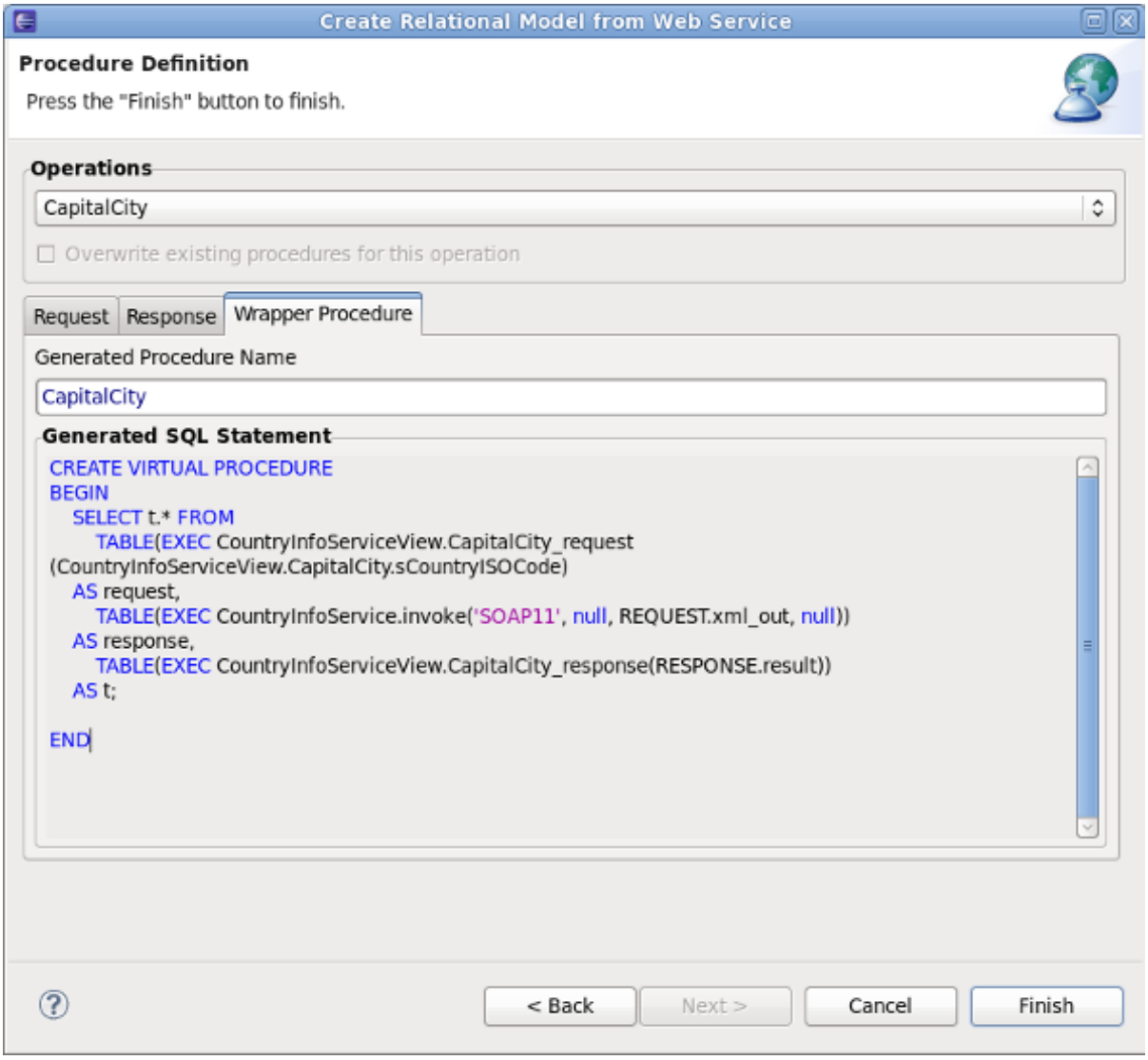


Figure 7.31. Consume SOAP Wizard

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 7.32. 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.

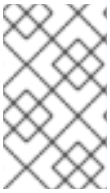
CHAPTER 8. NEW MODEL WIZARDS

8.1. LAUNCH NEW MODEL WIZARDS

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

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

- Click **`File > New > Teiid Metadata Model`** action .
- Select a project or folder in the **`Model Explorer View`** and choose the same action in the right-click menu.
- Select the **`New`** button on the main toolbar and select the Teiid Metadata Model action .



NOTE

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

8.2. CREATE NEW RELATIONAL SOURCE MODEL

8.2.1. Create a New Relational Source Model

To create a new empty relational source model:

1. Launch the **`New Model Wizard`**.
2. Specify a unique model name.
3. Select **`Relational`** option from **`Model Class`** drop-down menu.
4. Select **`Source Model`** from **`Model Type`** drop-down menu.
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, you can also copy from existing model of the same model class.

8.2.2. 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 first 4 steps from the **`Create a New Relational Source Model`** section and continue with these additional steps:

1. Select the model builder labeled **`Generate File Translator Procedures`** and click **`Next >`**. The **`Generate File Translator Procedures`** dialog will be displayed.

2. Select one or more of the **`Available File Translator Procedures`**, then click **`Finish`**.

8.2.3. 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 service translator.

To create a new relational model containing web-service-based translator procedures, complete first 4 steps from the **`Create a New Relational Source Model`** section and continue with these additional steps:

1. Select the model builder labeled **`Generate Web Service Translator Procedures`** and click **`Next >`**. The **`Generate Web Service Translator Procedures`** dialog will be displayed.

2. Select one or more of the **`Available Web Services Translator Procedures`**, then click **`Finish`**.

8.2.4. Copy from an Existing Relational Source 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 first 4 steps from the **`Create a New Relational Source Model`** section and continue with these additional steps:

1. 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.

2. Select an existing relational model from the workspace using the browse button.

   NOTE

   An existing model will be preselected if a relational model in the workspace is selected in the model-explorer-view prior to starting the new model wizard.

3. Select the **`Copy all descriptions`** option if desired. Click **`Finish`**.

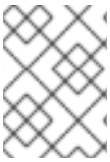
8.3. CREATE A NEW RELATIONAL VIEW MODEL

8.3.1. Create a New Relational View Model

To create a new empty relational view model:

1. Launch the **`New Model Wizard`**.

2. Specify a unique model name.
3. Select **`Relational`** option from **`Model Class`** drop-down menu.
4. Select **`View Model`** from **`Model Type`** drop-down menu.
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:

1. Copy from existing model of the same model class.
2. Transform from existing model.

8.3.2. Copy an Existing Relational View 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 first 4 steps from the **`Create a New Relational View Model`** section and continue with these additional steps:

1. 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.
2. Select an existing relational model from the workspace using the browse button.
3. Select the **`Copy all descriptions`** option if desired. Click **`Finish`**.

8.3.3. Transform from an Existing Relational View 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 Copy from Relational View Model section described above.

There is an additional option in the second dialog window of the wizard which can automatically set the relational table's supports update property to false. If this is not selected, the default value will be true.

8.4. CREATE A NEW XML DOCUMENT VIEW MODEL

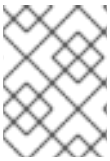
IMPORTANT

Please note that XML Document Modeling has been deprecated.

8.4.1. Create a New XML Document View Model

To create a new empty XML document view model:

1. Launch the **`New Model Wizard`**.
2. Specify a unique model name.
3. Select **`XML`** option from **`Model Class`** drop-down menu.
4. Select **`View Model`** from **`Model Type`** drop-down menu.
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:

1. Copy from existing model of the same model class.
2. Build XML documents from XML schema.

8.4.2. Copy an Existing XML Document View 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 XML document view model by copying contents from another XML document view model, complete first 4 steps from the **`Create a New XML Document View Model`** section and continue with these additional steps:

1. 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.
2. Select an existing XML document from the workspace using the browse button.
3. Select the **`Copy all descriptions`** option if desired. Click **`Finish`**.

8.4.3. Build an XML Document View Model

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 first 4 steps from the **`Create a Xml Document View Model`** section and continue with these additional steps:

1. Select the model builder labeled **`Build XML documents from XML`** schema and click **`Next >`**. The **`Select XML Schema`** dialog will be displayed.
2. Select an existing schema model from the workspace using the browse button.

NOTE

An existing model will be preselected 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.

3. 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 to move all elements.
4. Select the appropriate document options and mapping options. Click **`Finish`**.
5. 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.
6. (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.

8.5. CREATE A NEW XML SCHEMA MODEL

IMPORTANT

Please note that XML Schema Modeling has been deprecated.

8.5.1. Create a New XML Schema Model

To create a new empty XML schema (**`.xsd`**) model:

1. Launch the **`New Model Wizard`**.
2. Specify a unique model name.
3. Select **`XML Schema (XSD)`** option from **`Model Class`** drop-down menu.
4. Select **`Datatype Model`** from **`Model Type`** drop-down menu.
5. Click **`Finish`**.
6. After that, in the dialog choose XML Schema version which will be used and click **`OK`**.

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.

8.5.2. Copy an Existing XML Schema 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 XML schema model by copying contents from another XML schema model, complete first 4 steps from the **`Create a New XML Schema Model`** section and continue with these additional steps:

1. 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.
2. Select an existing XML schema model from the workspace using the browse button.
3. Select the **`Copy all descriptions`** option if desired. Click **`Finish`**.

8.6. CREATE A NEW WEB SERVICE VIEW MODEL

IMPORTANT

Please note that Web Service View Modeling has been deprecated.

8.6.1. Create a New Web Service View Model

To create a new empty web service view model:

1. Launch the **`New Model Wizard`**.
2. Specify a unique model name.
3. Select **`Web Service`** option from **`Model Class`** drop-down menu.
4. Select **`View Model`** from **`Model Type`** drop-down menu.
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:

1. Copy from existing model of the same model class.
2. Build from existing WSDL file(s) or URL.

8.6.2. Copy an Existing Web Service View 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 Web Service view model by copying contents from another web service view model, complete first 4 steps from the **`Create a New Web Service View Model`** section and continue with these additional steps:

1. 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.
2. Select an existing Web Service view model from the workspace using the browse button.
3. Select the **`Copy all descriptions`** option if desired. Click **`Finish`**.

8.6.3. Create a Web Service View Model from a WSDL File 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 Web Service View Model, complete first 4 steps from the **`Create a New Web Service View Model`** section and continue with these additional steps:

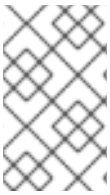
1. Select the model builder labeled **`Build from existing WSDL file(s) or URL`** and click **`Next >`**.
2. Complete the steps in the wizard. The remaining steps are the same as the process covered in *Import WSDL Into Web Service* at Section 9.10.1, “Import WSDL Into Web Service”.

8.6.4. Create a Web Service View Model 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:

1. Select either a single XML Document or single XML Document root in the Model Explorer View.
2. Right-click select **`Modeling > Create Web Service`** action .
3. Fill in missing properties in **`Web Service Generation Wizard`**.
4. Click **`Finish`** to generate model. When model generation is complete, a confirmation dialog is displayed. Click **`OK`**.



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.

CHAPTER 9. IMPORTERS

9.1. IMPORTERS

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

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

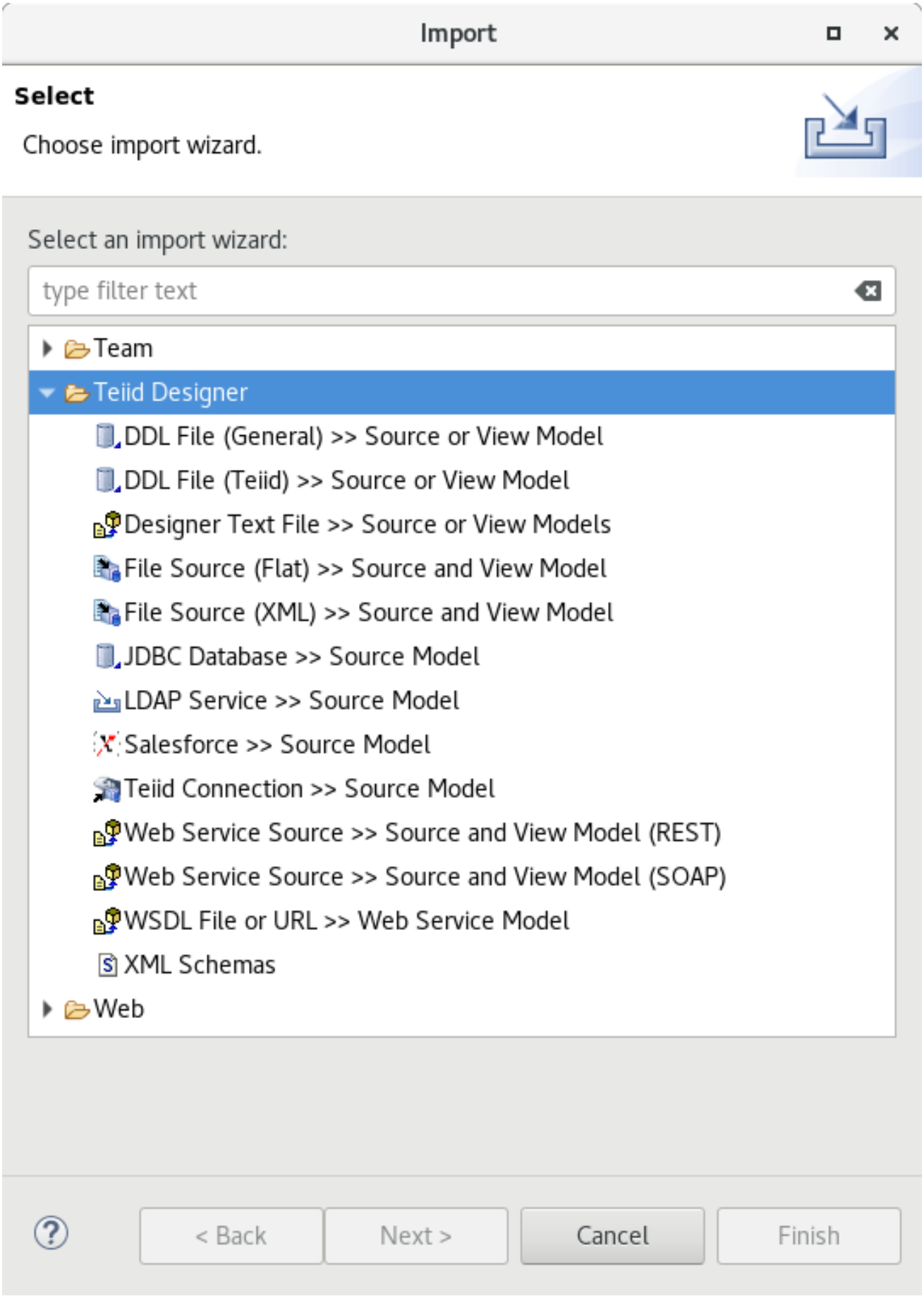


Figure 9.1. Import Wizard

9.2. IMPORT DDL

You can create source relational models by importing DDL using the steps below.

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action.

2. Select the import option **`Teiid Designer > DDL File (General) >> Source or View Model`** and click **`Next>`**.

   NOTE

   You can also choose **`DDL File (Teiid)`** which import metadata from Teiid DDL file.

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)

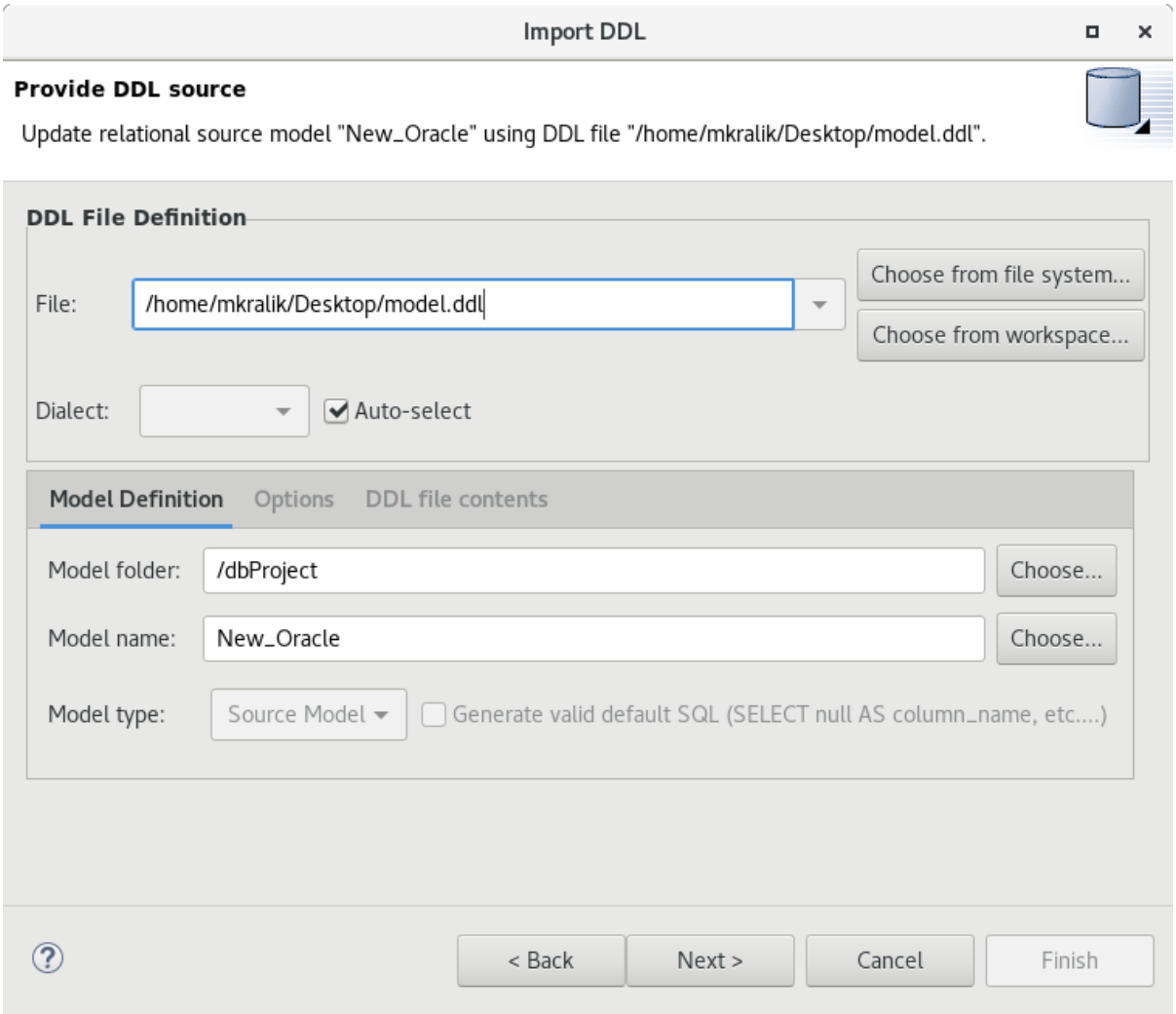


Figure 9.2. DDL Import Options

4. If you click **`NEXT>`**, a difference report is presented for viewing or deselecting individual relational entities. Click **`Finish`** to complete.

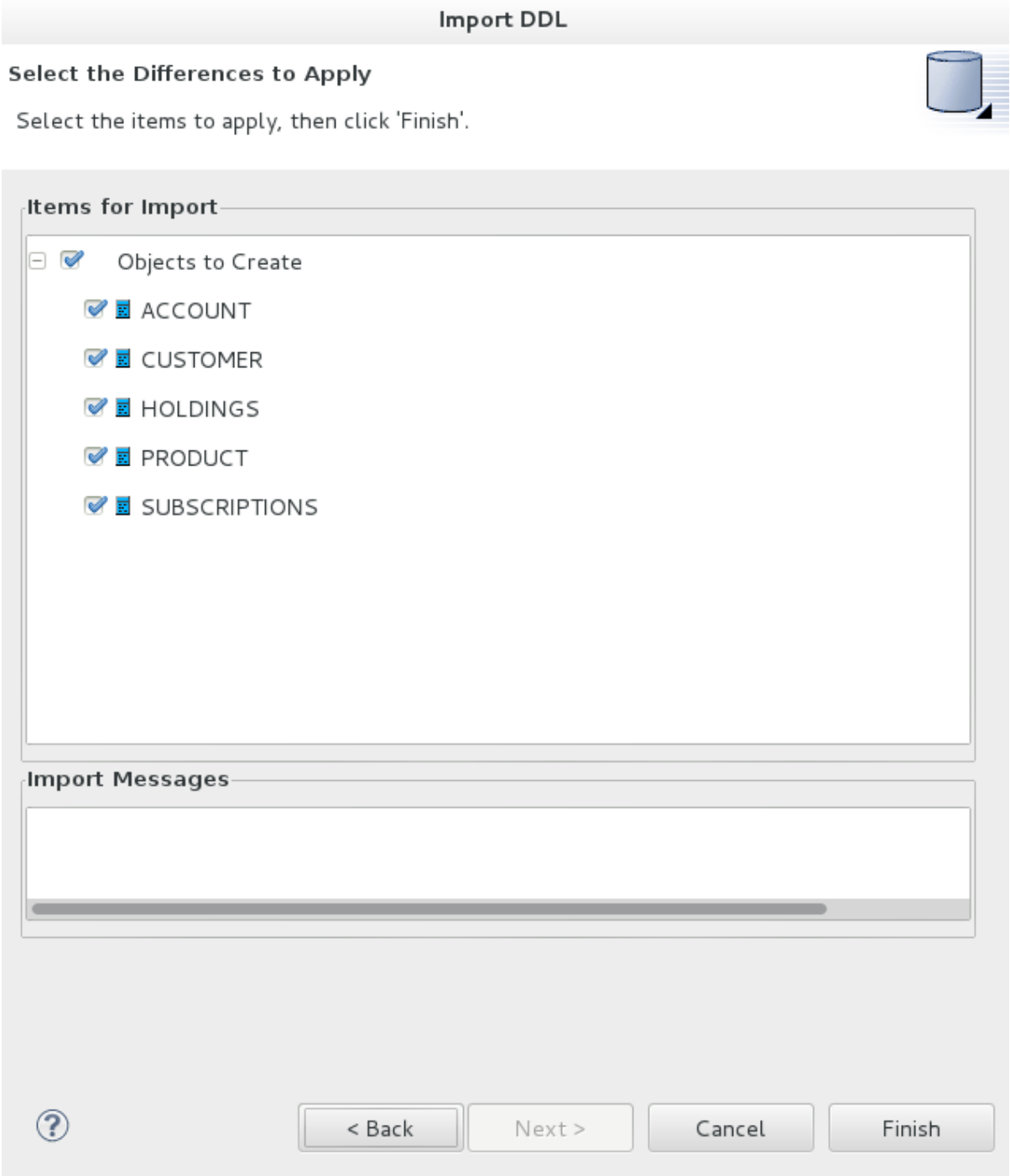


Figure 9.3. Import DDL Dialog

9.3. IMPORT FROM JDBC DATABASE

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

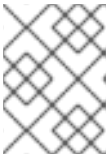
NOTE

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

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**

2. Select the import option **`Teiid Designer > JDBC Database >> Source Model`** and click **`Next>`**.

3. Select existing or previous connection profile from the drop-down selector or click **`New...`** button to launch the **`New Connection Profile`** dialog 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.

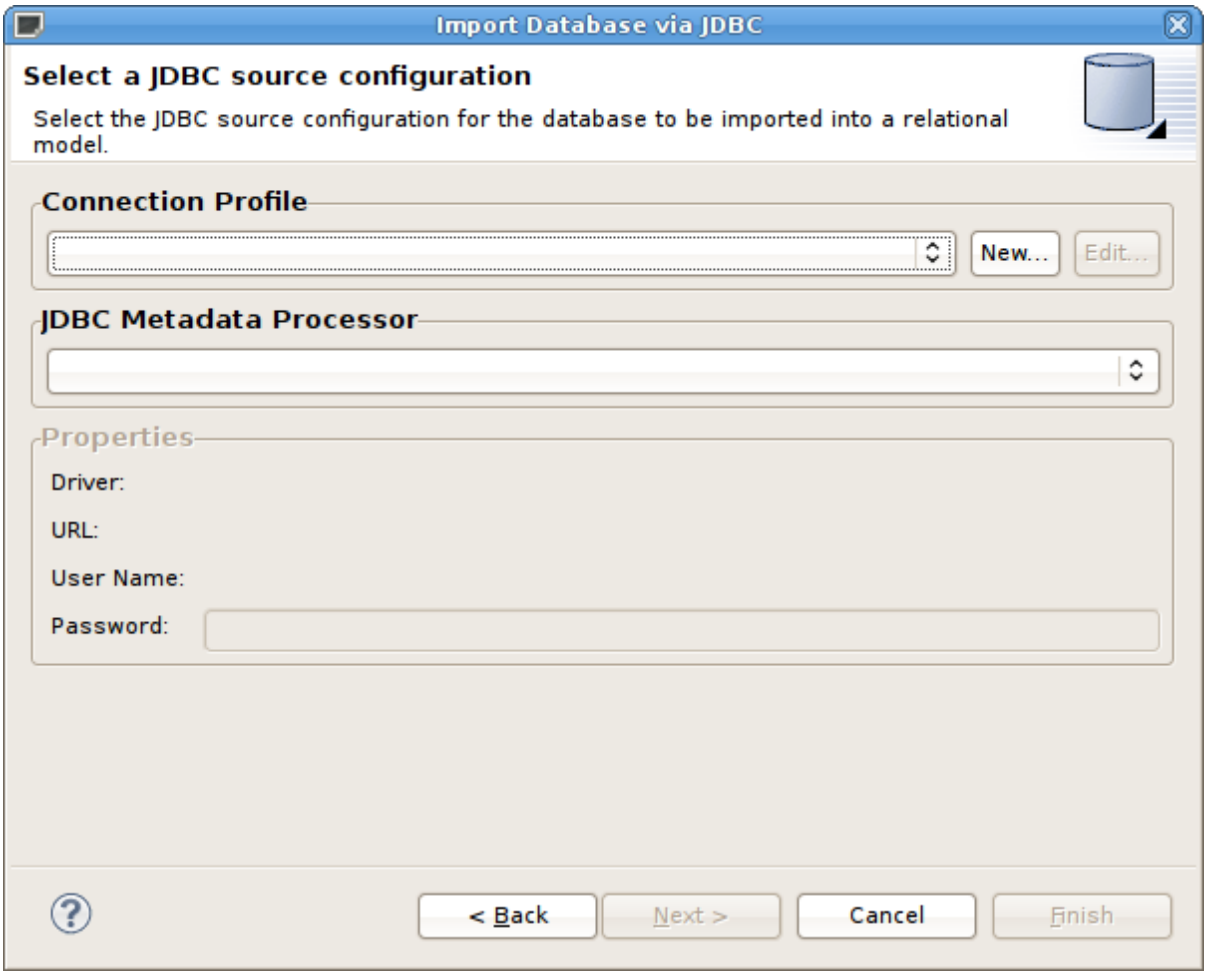


Figure 9.4. Select JDBC Source Configuration Dialog

Because JDBC 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 selected automatically based on your selected connection profile. Special processors are available for DB2, ModeShape, ODBC, Oracle, PostgeSQL, SQL Server and Sybase. For all other DBs a default JDBC processor is available.

4. After selecting a Connection Profile, enter the password (if not provided). Click **`Next>`** (or **`Finish`** if enabled).

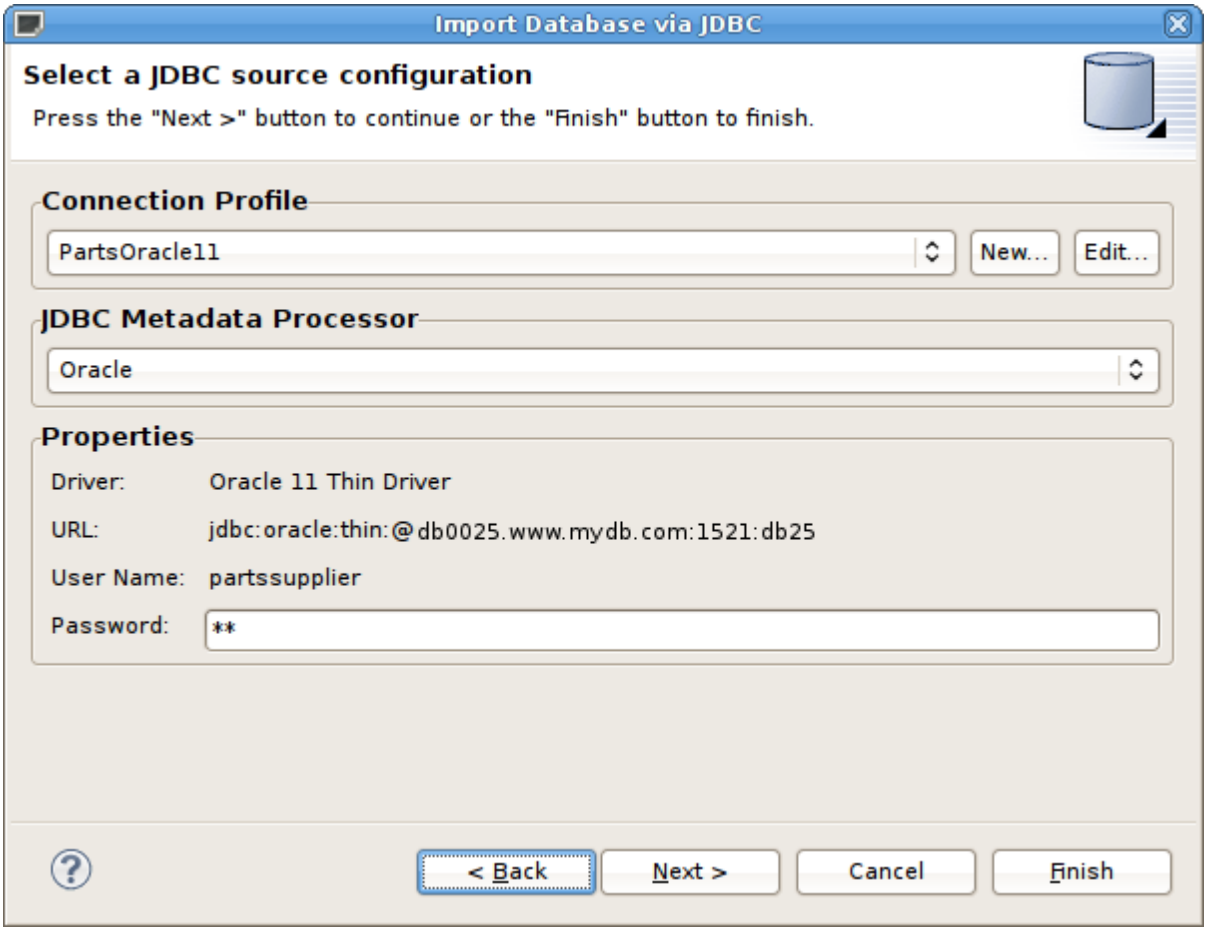


Figure 9.5. Select JDBC Source Configuration Dialog

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

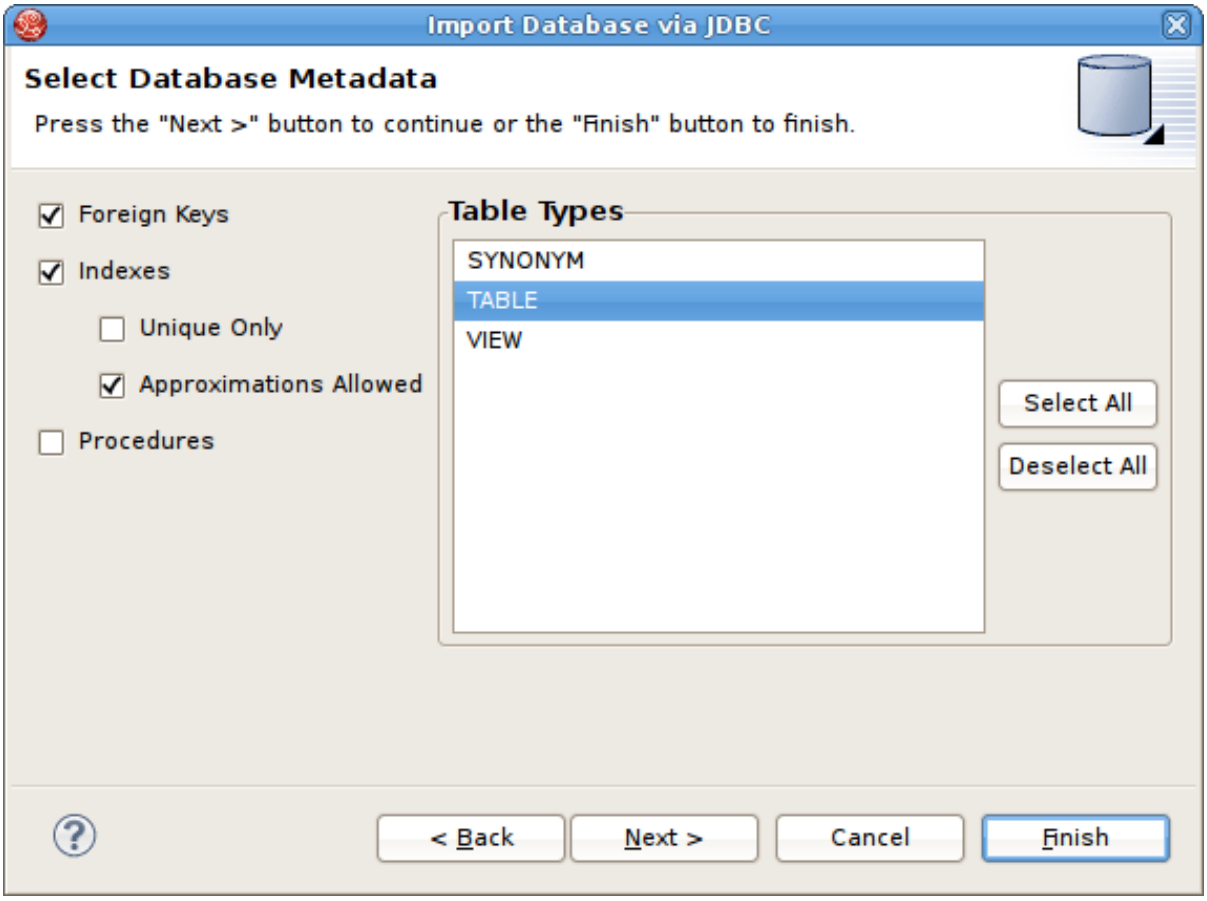


Figure 9.6. Select Database Metadata Dialog

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. Click **`Next>`** (or **`Finish`** if enabled).

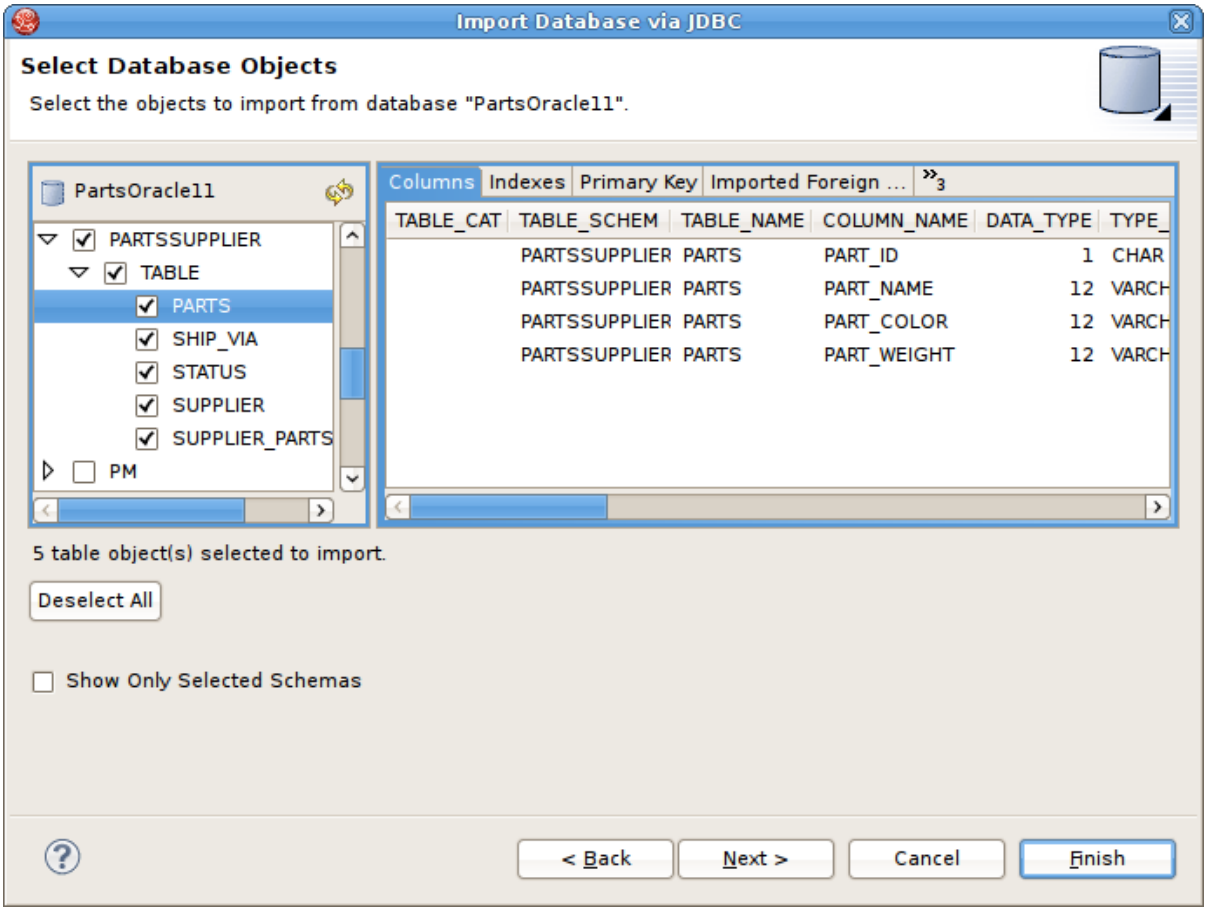


Figure 9.7. Select Database Options Dialog

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. Click **`Finish`** to complete.

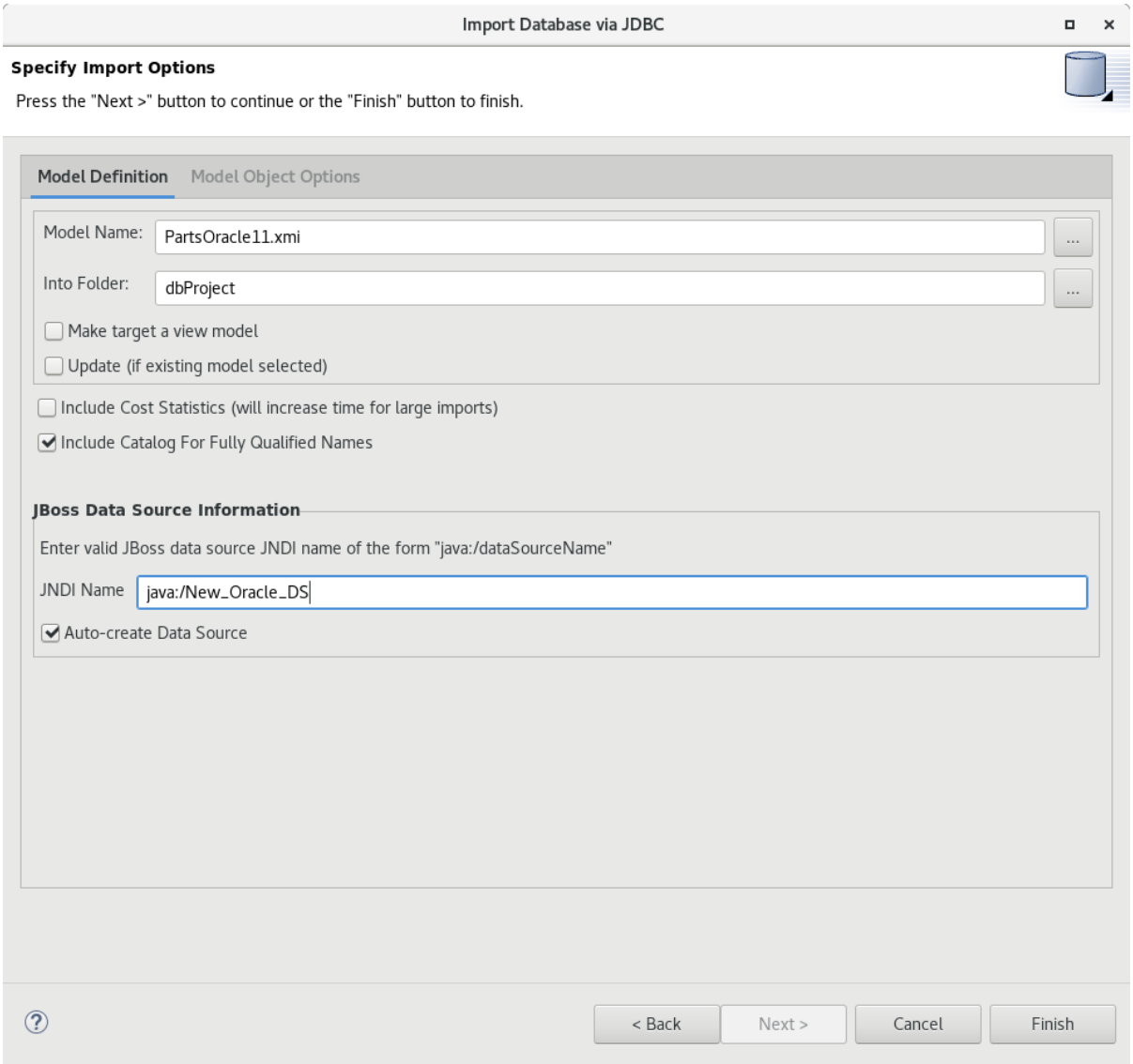


Figure 9.8. Specify Import Options Dialog

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

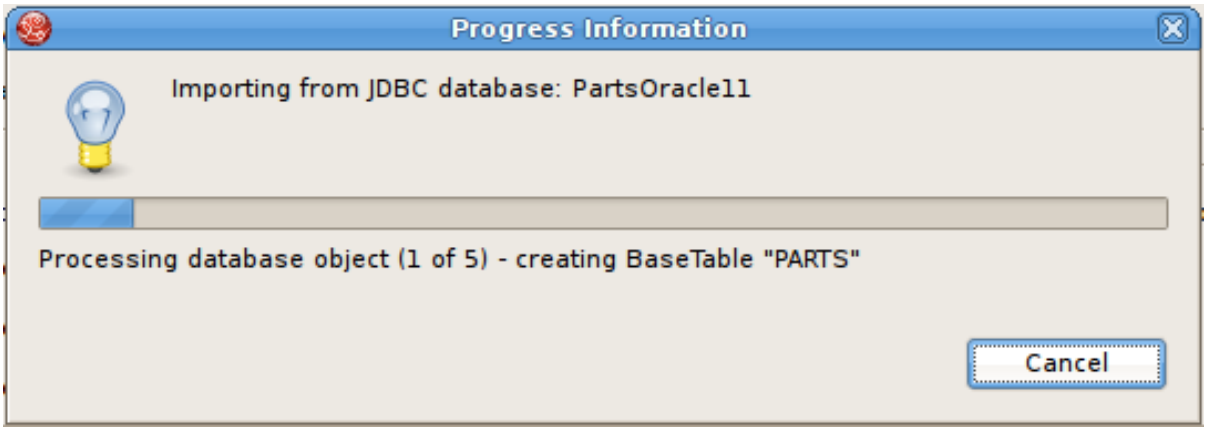
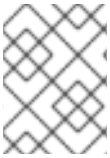


Figure 9.9. JDBC Import Progress Dialog

9.4. IMPORT FROM TEIID DATA SOURCE CONNECTION

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



NOTE

A running Teiid 8.3 or later instance is required. Ensure that the Default Server is started before you begin the import procedure.

Procedure 9.1. Import From Teiid Data Source Connection

1. Choose the **`File > Import...`** action in the toolbar or go to the **`Model Explorer`** and wherever right click and select **`Import...`** .
2. Select **Teiid Designer** > **Teiid Connection** >> **Source Model** and then click **`Next`**.
3. Select the datasource to use for the import.

   You can create a new source if the one you want does not already exist. You can also undertake other source management functions here.
4. Click **`Next`**.
5. Select the appropriate translator for your data source type and then click **`Next`**.
6. Set the location where you plan to import the model, define a name for the model, specify any other options and then click **`Next`**.
7. When you move to next page of the wizard, a temporary dynamic vdb is actually deployed to your server and the schema for your data source is retrieved in DDL form. This DDL is displayed (and you can also be exported if you so desire). Click **`Next`**.
8. On the final page of the wizard, a difference report is presented for viewing or de-selecting individual relational entities. Press **`Click`** to complete.

9.5. 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 parameters 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.

JBoss Data Virtualization 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 **JBoss Data Virtualization**.

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 processes character input to produce tabular output. 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 preceding 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.

1. In **Model Explorer**, click **File > Import** action in the toolbar or select a project, folder or model in the tree and click **Import...**
2. Select the import option **Teiid Designer > File Source (Flat) >> Source and View Model** and click **Next>**.

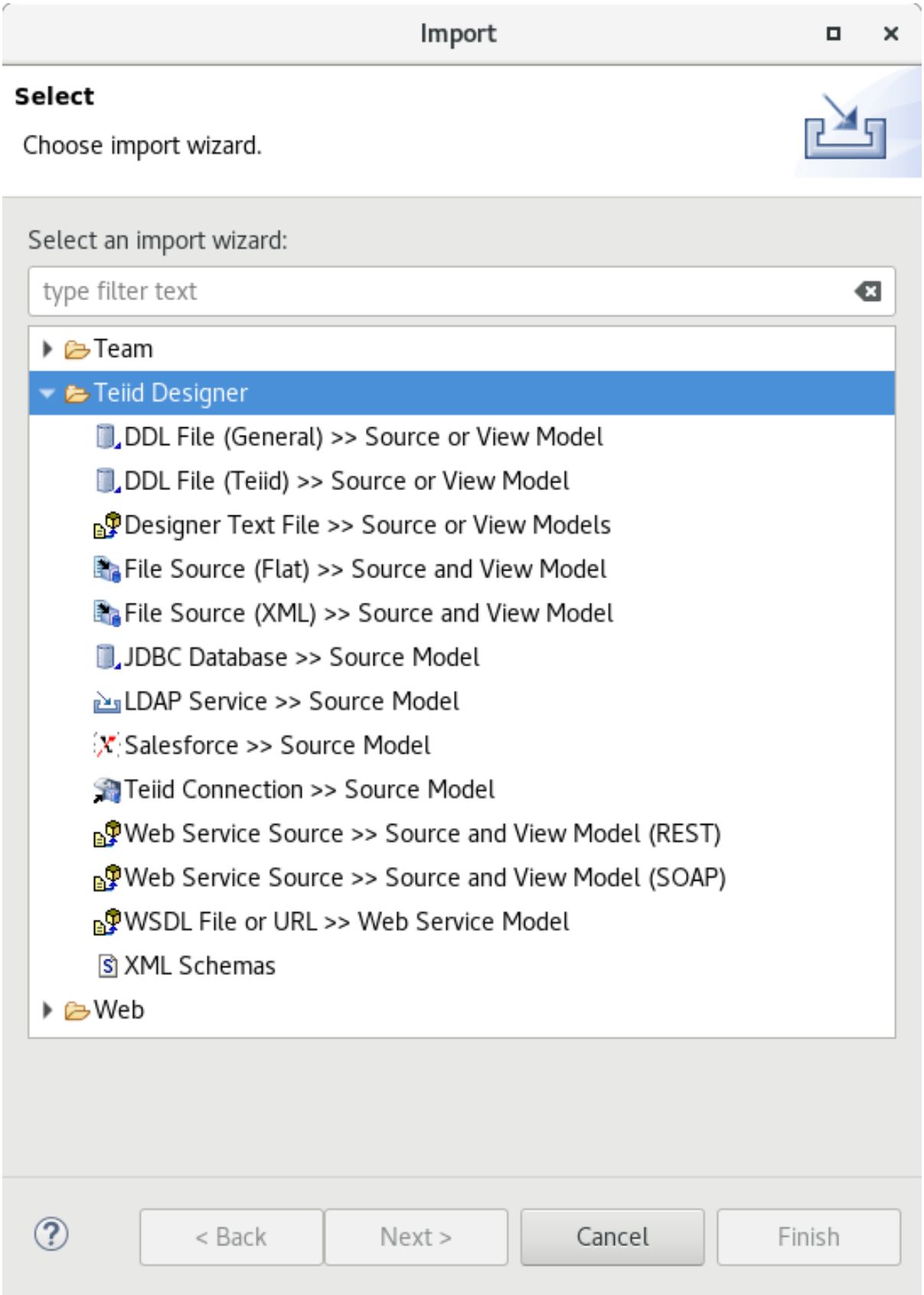


Figure 9.10. Import from Flat File Source

3. Select **`Flat File Import Mode`** and then select either **`Flat file on local file system`** or **`Flat file via remote URL`** and click **`Next>`**.

4. Select existing or previous connection profile from the drop-down selector or click **`New...`** button to launch the **`New Connection Profile`** dialog or **`Edit...`** to modify or change an existing connection profile prior to selection. Note that 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. Select 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.

NOTE

The path to the data file you select must not contain spaces.

Lastly enter a 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>`**.

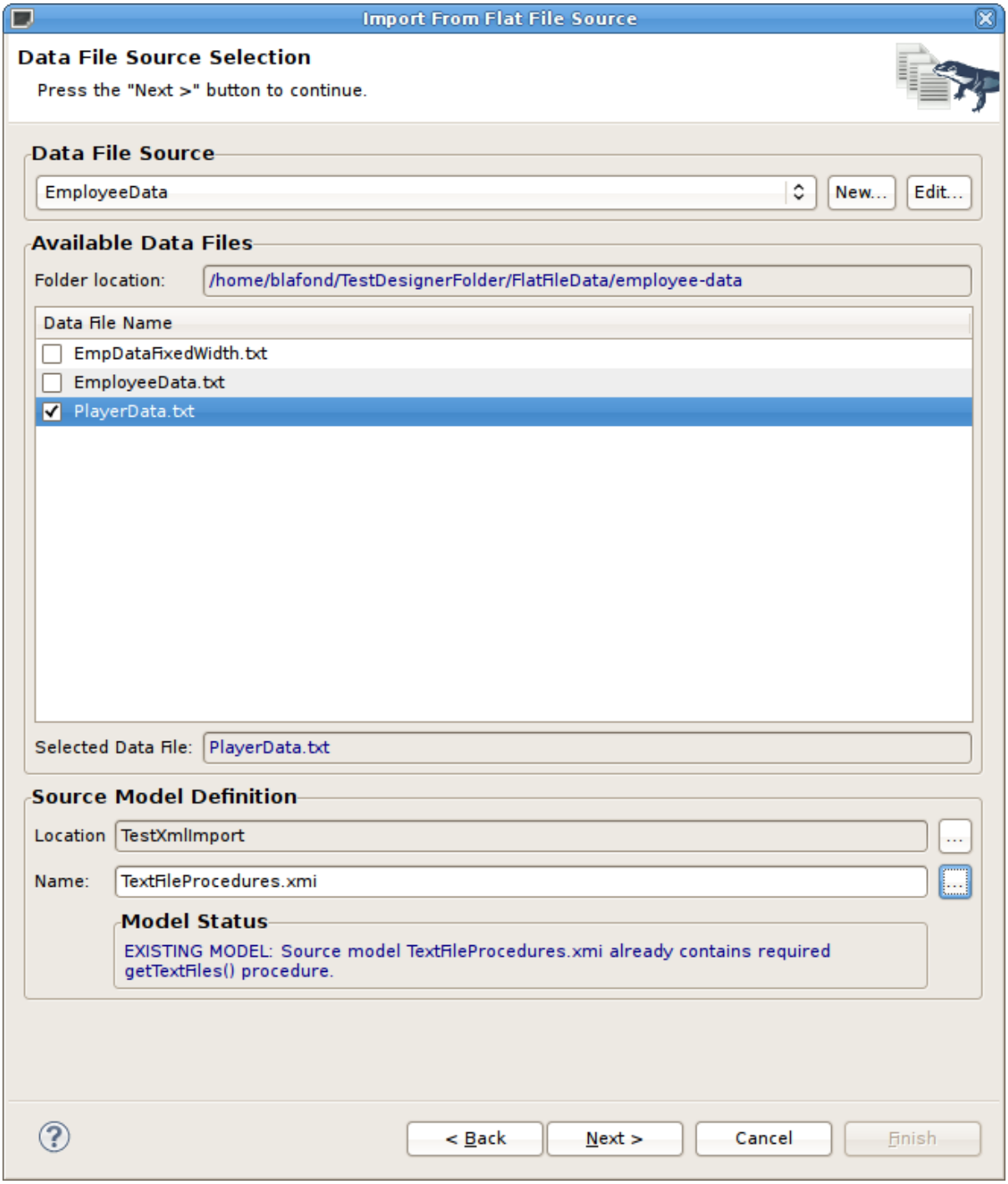


Figure 9.11. Data File Source Selection Page

5. On the next page enter the JNDI name and click **`Next >`**.

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>`**.

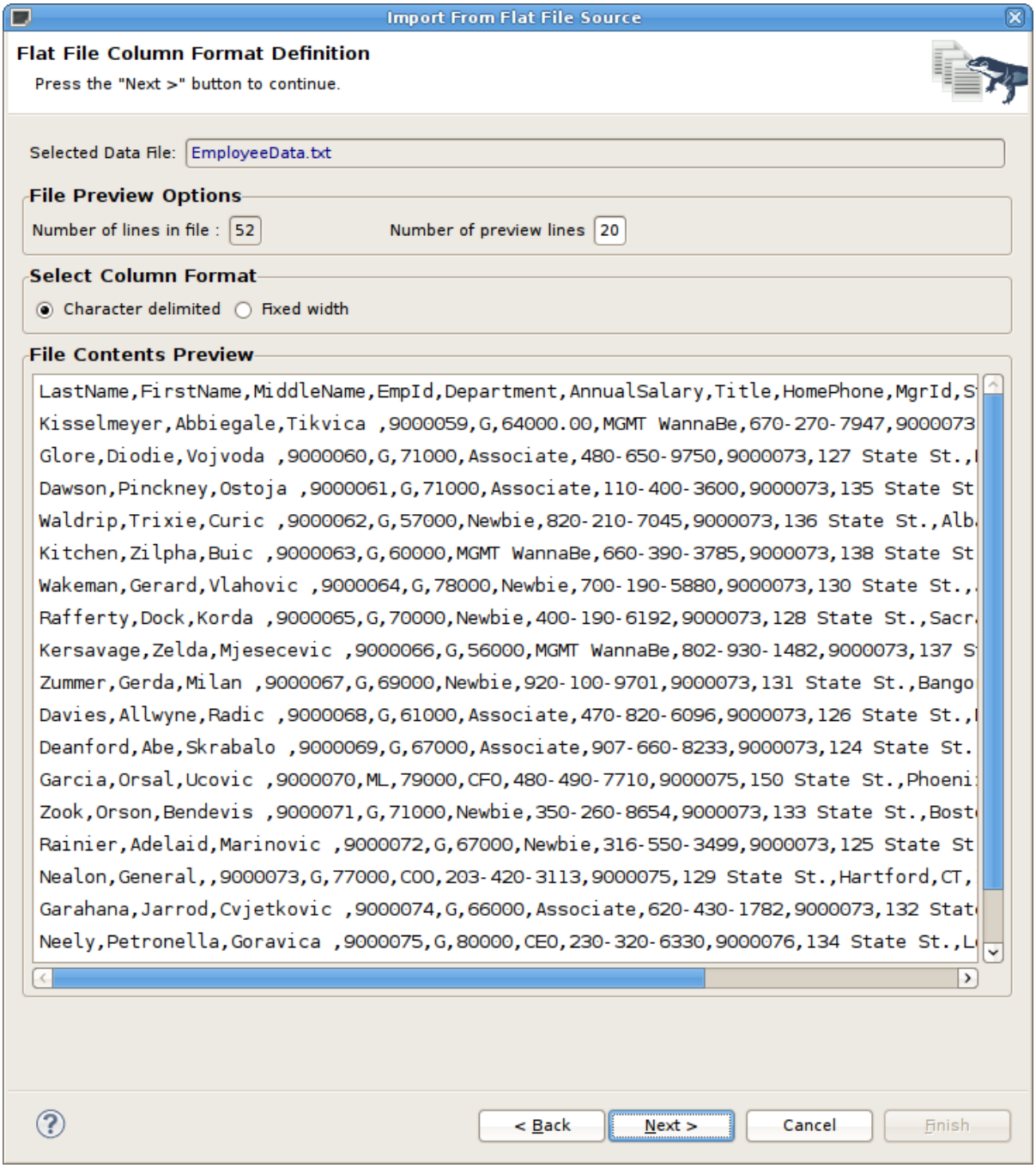


Figure 9.12. Data File Source Selection Page

7. 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 and changed character delimiter.

   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>**.

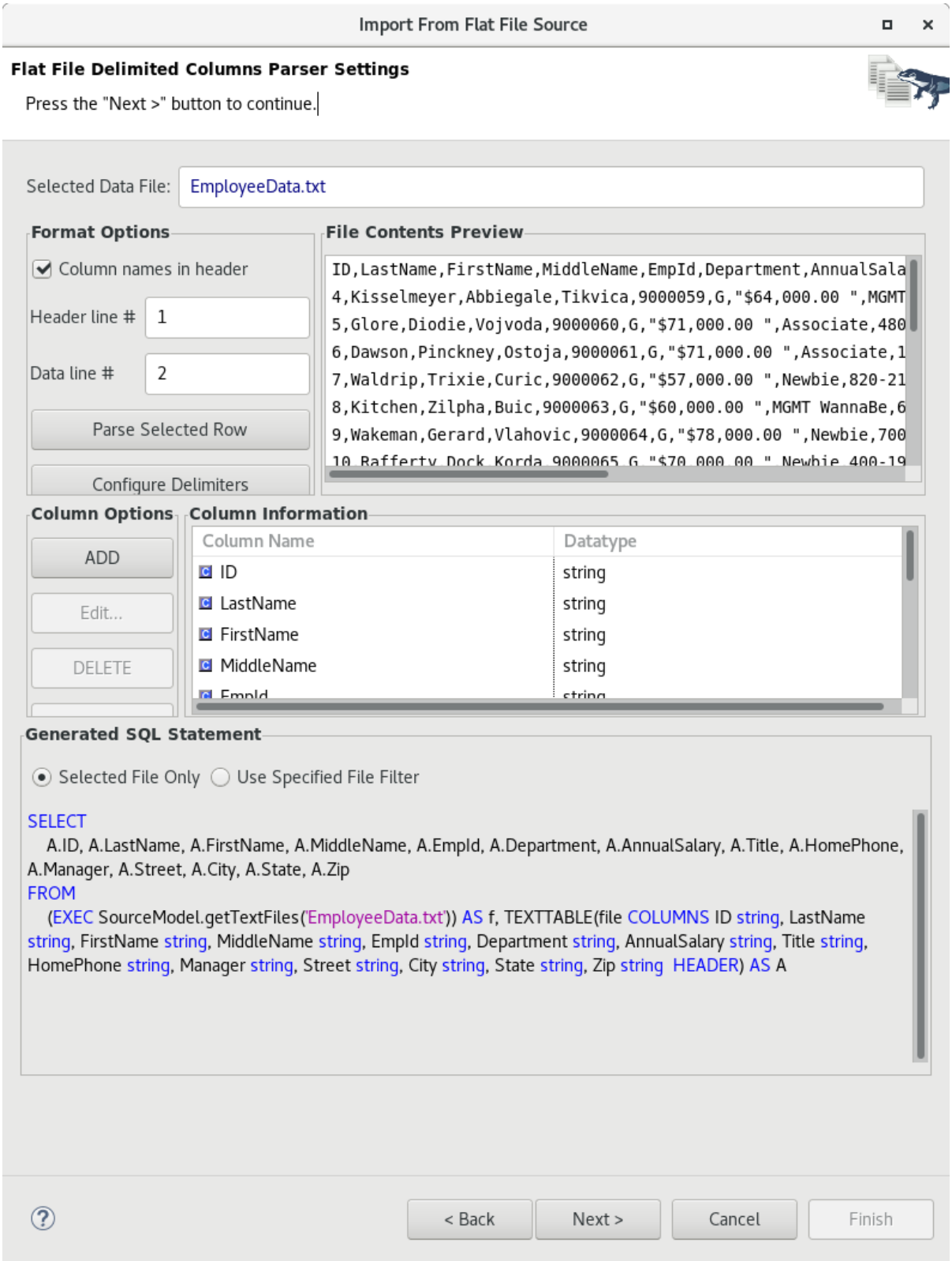


Figure 9.13. Flat File Delimited Columns Options Page

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.

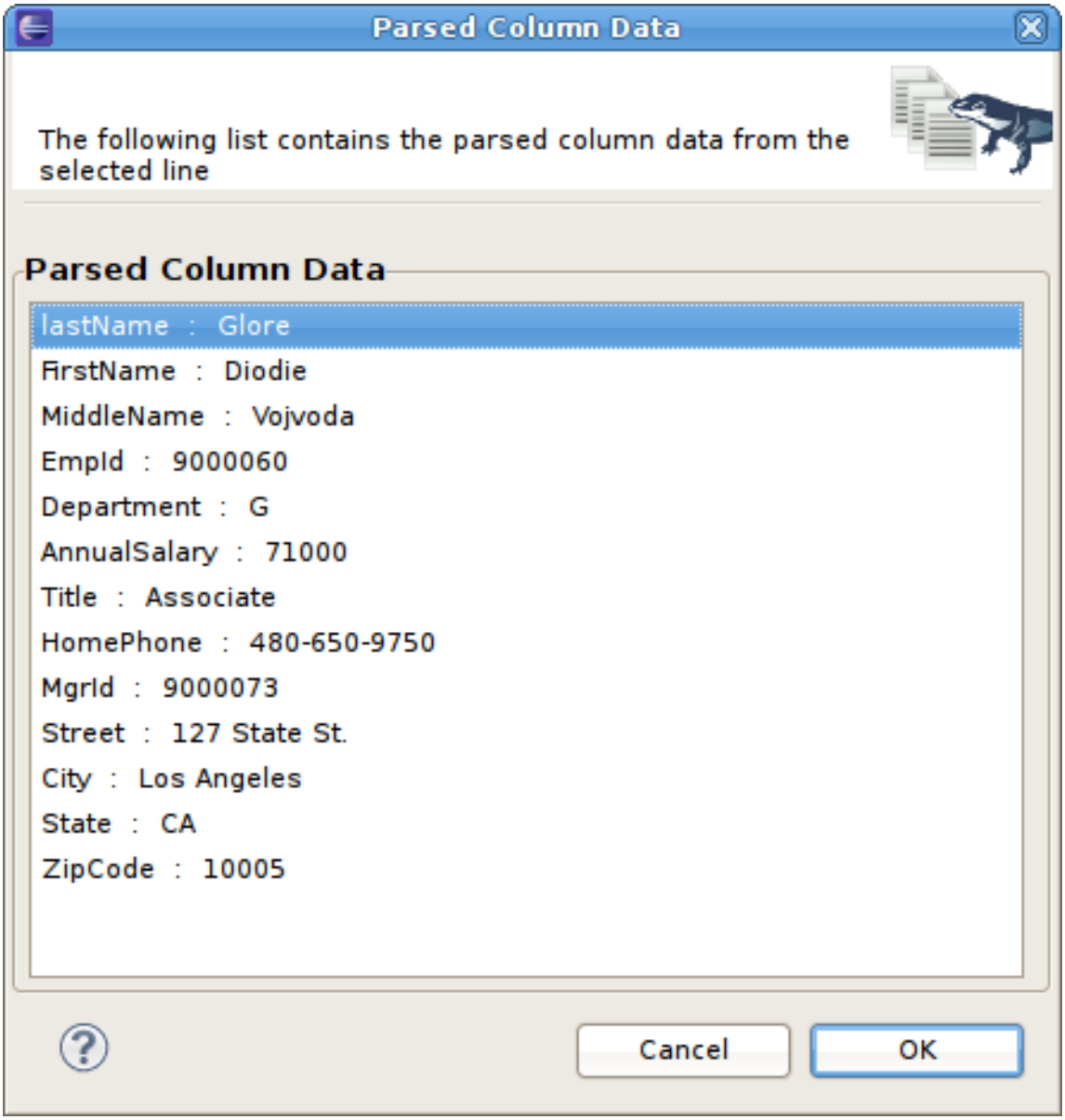


Figure 9.14. Parse Column Data Dialog

8. 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 such as first data line number and changed character delimiter.

   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 position 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>`**.

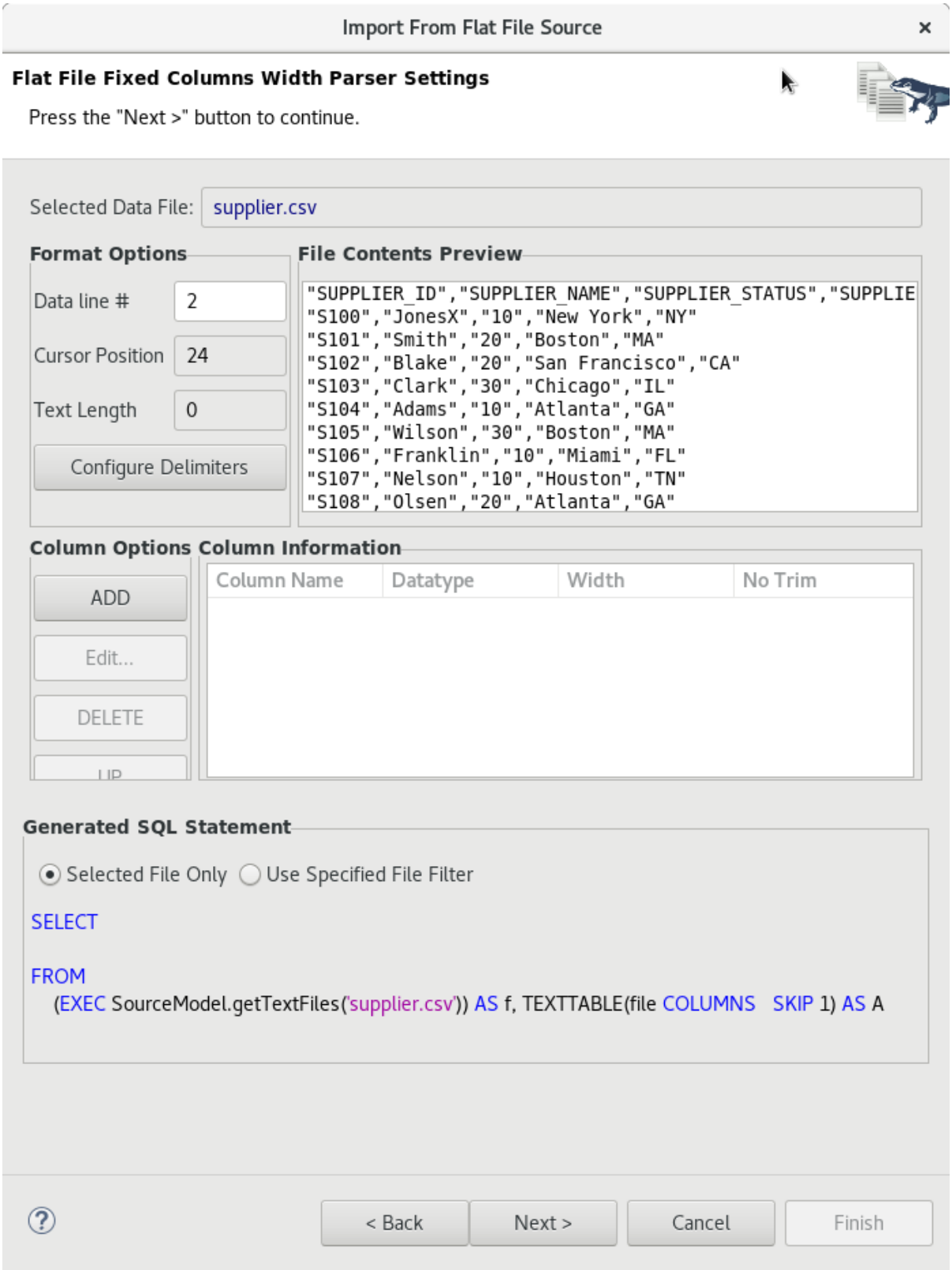


Figure 9.15. Flat File Fixed Columns Width Options Page

9. 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. Click **`Finish`** to generate your models and finish the wizard.

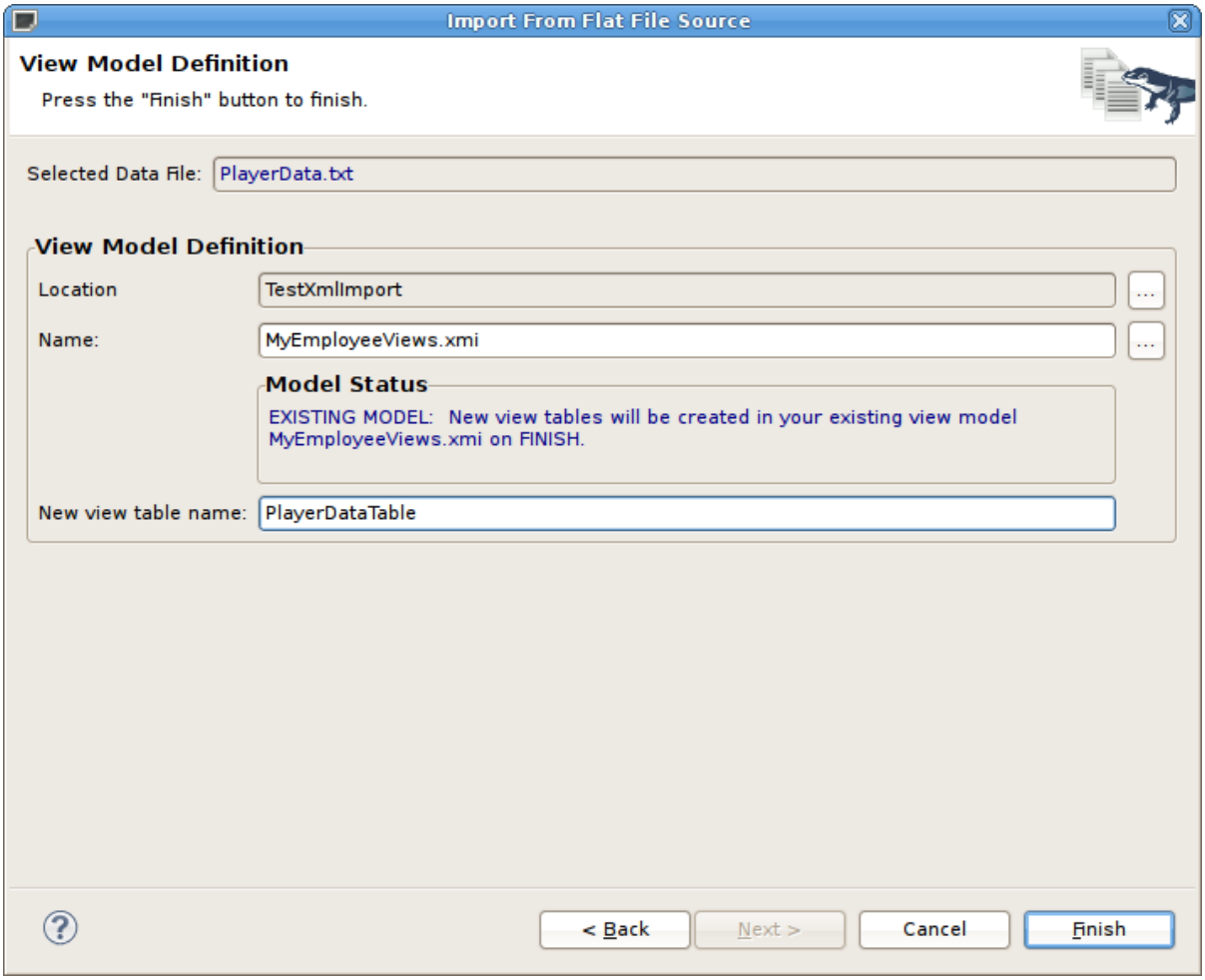


Figure 9.16. View Model Definition Page

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

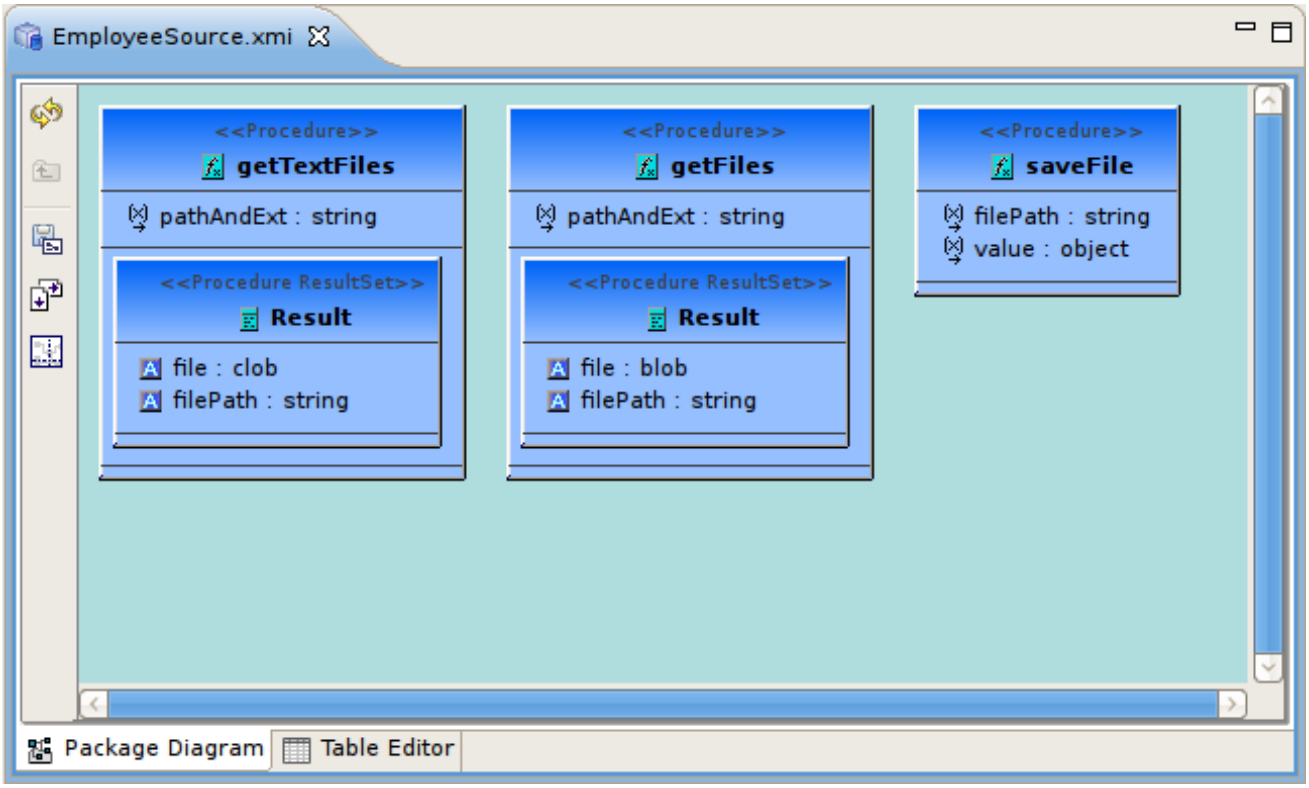


Figure 9.17. Generated Flat File Procedures

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.

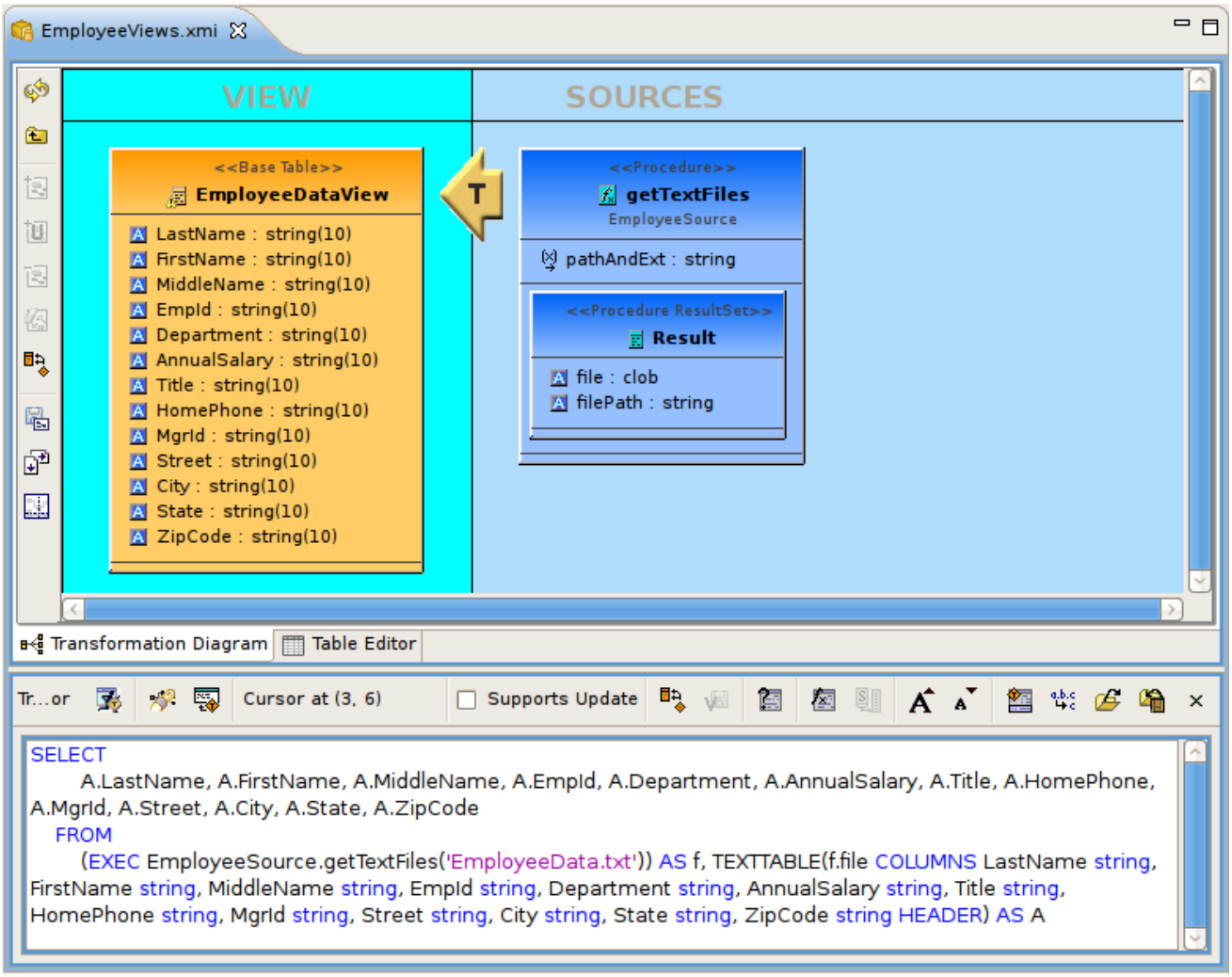


Figure 9.18. Generated Flat File View Table

9.6. IMPORT FROM XML DATA FILE SOURCE

JBoss Data Virtualization 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 XML data source, configure your parsing parameters 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 Teiid 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 JBoss Data Virtualization.

The importer will also create a new view model containing a view table for your selected XML 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 output. The **`XMLTABLE`** function is implicitly a nested table and may be correlated to preceding 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.

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**

2. Select the import option **`Teiid Designer > File Source (XML) >> Source and View Model`** and click **`Next>`**

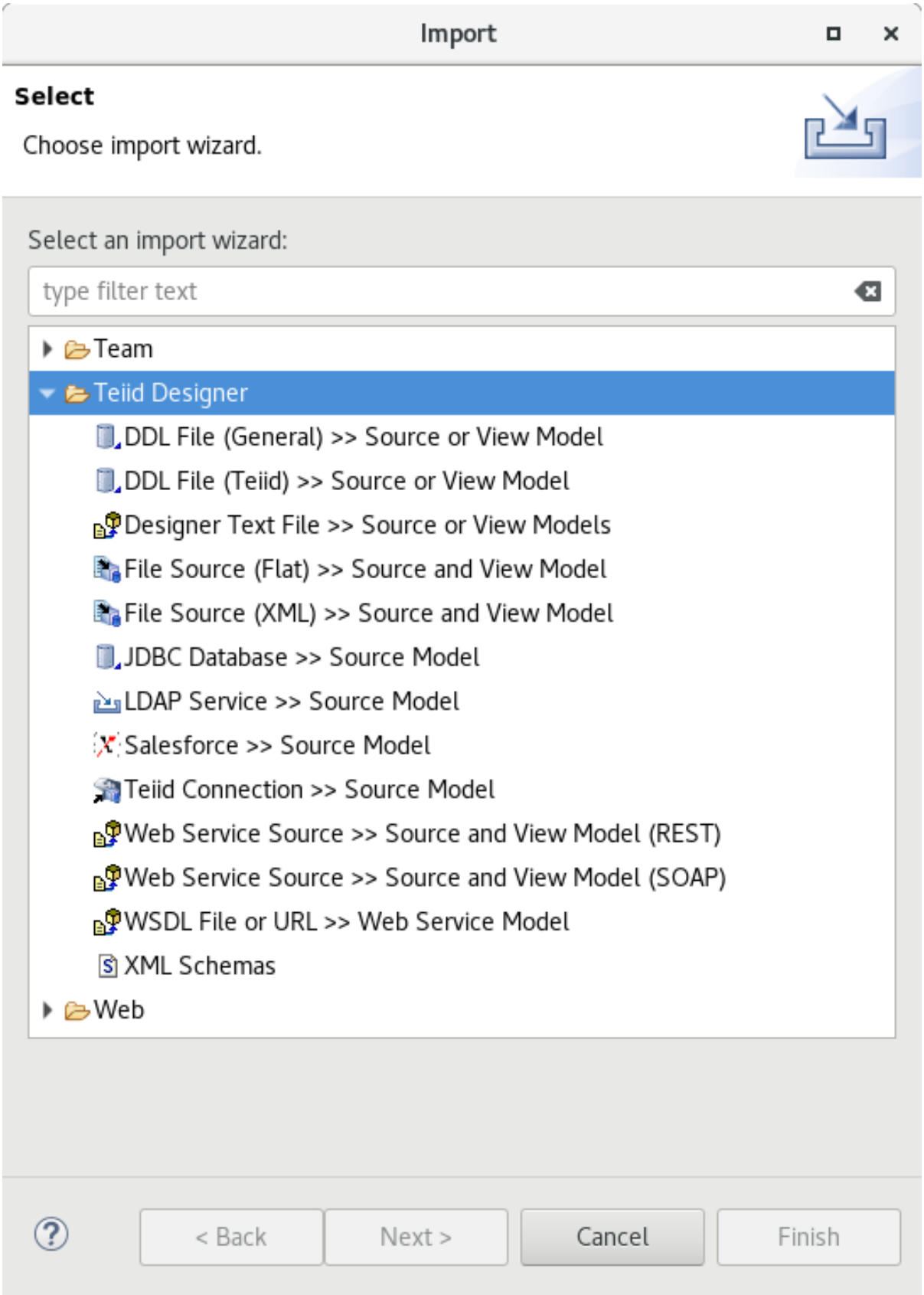


Figure 9.19. Import from XML File Source

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>`**

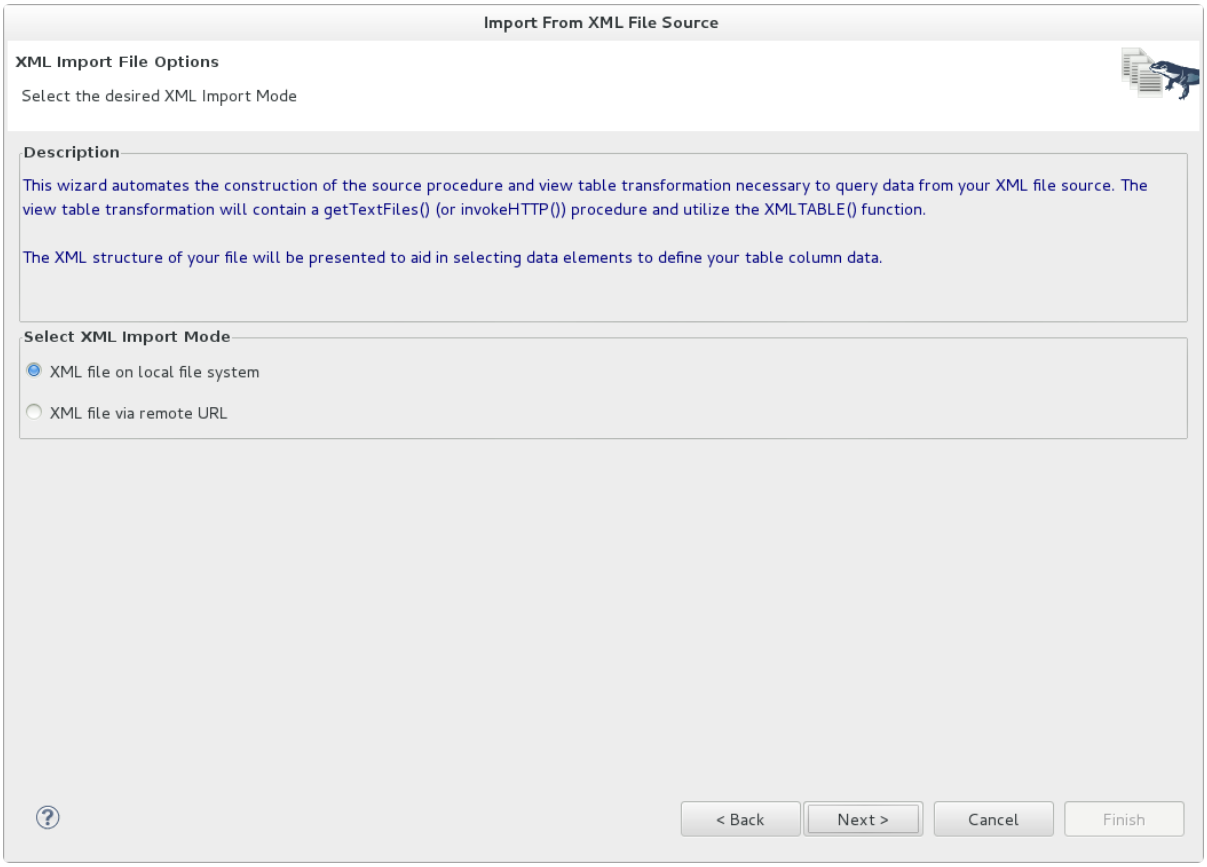


Figure 9.20. XML Import File Options Page

4. Select 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.

   After selecting a Connection Profile, the XML data file from the connection profile will be displayed in the **`Data File Name`** panel. Select 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 the 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>`**.

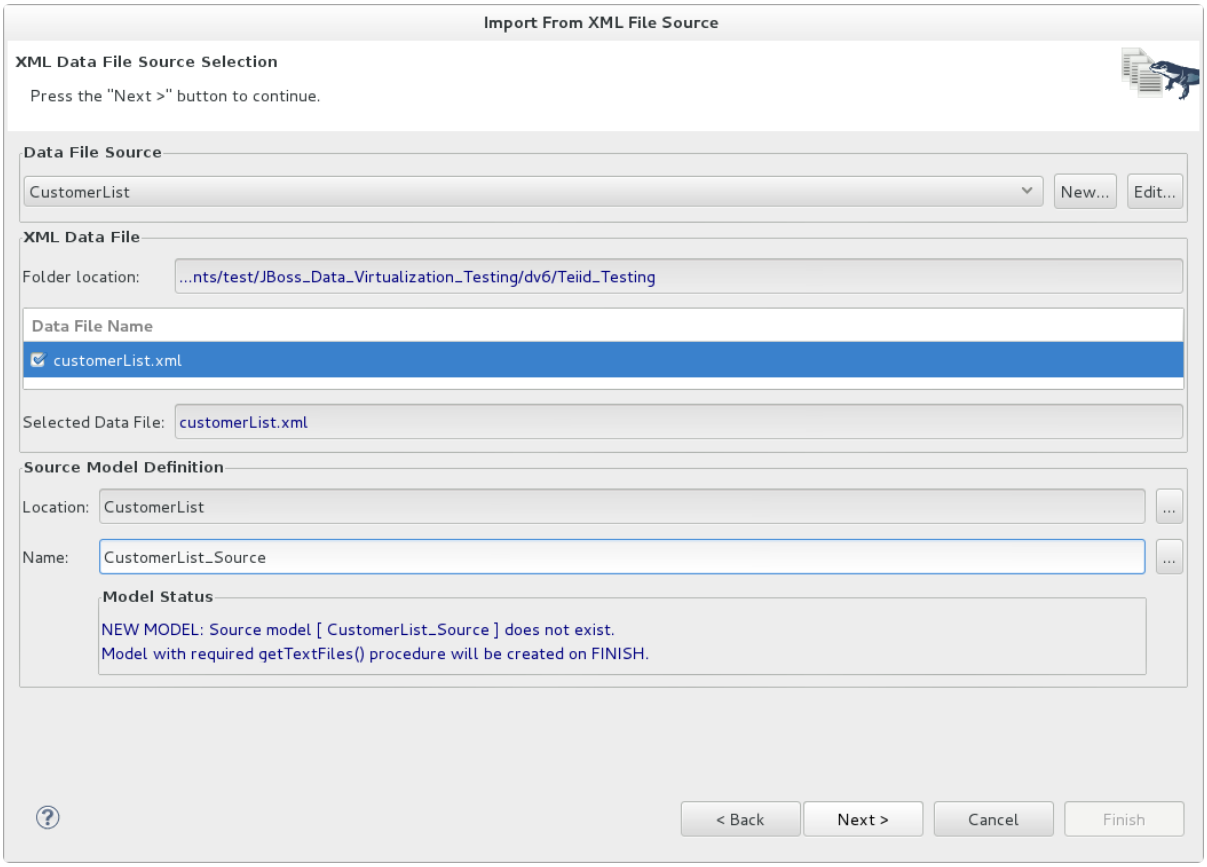


Figure 9.21. XML Data File Source Selection Page

5. On the next page enter the JNDI name and click **`Next >`**.

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. You can modify or create custom columns, by using the **`ADD`**, **`DELETE`**, **`UP`**, **`DOWN`** to manage the column info in your SQL.

   Note that 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 Development Guide Volume 3: Reference Material for details on the **`XMLTABLE()`** function.

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

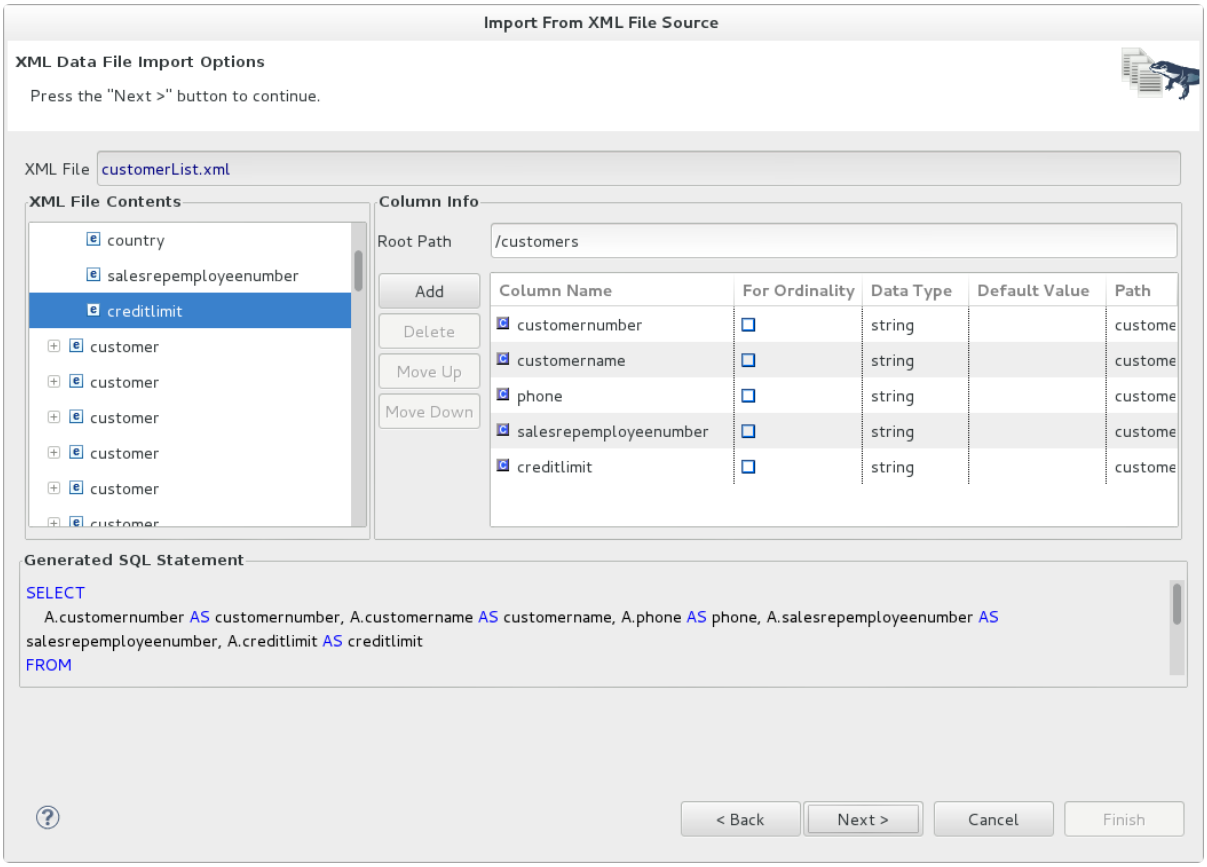


Figure 9.22. XML File Delimited Columns Options Page

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. Click **`Finish`** to generate your models and finish the wizard.

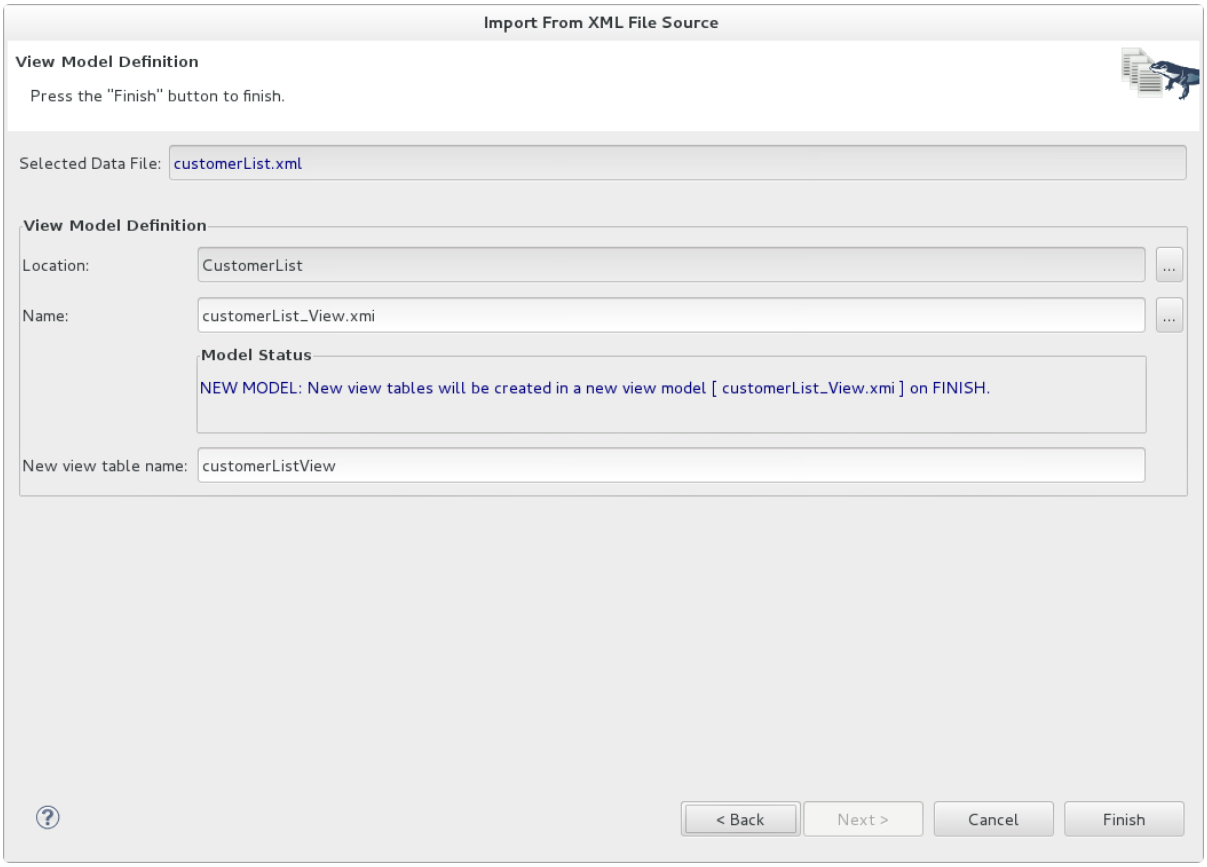


Figure 9.23. View Model Definition Page

9.7. 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.

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**

2. Select the import option **`Teiid Designer > Salesforce >> Source Model`** and click **`Next>`**

3. Select existing or previous connection profile from the drop-down selector or click **`New...`** button to launch the **`New Connection Profile`** dialog or **`Edit...`** to modify or change an existing connection profile prior to selection. Note that the Connection Profile selection list will be populated with only Salesforce connection profiles.

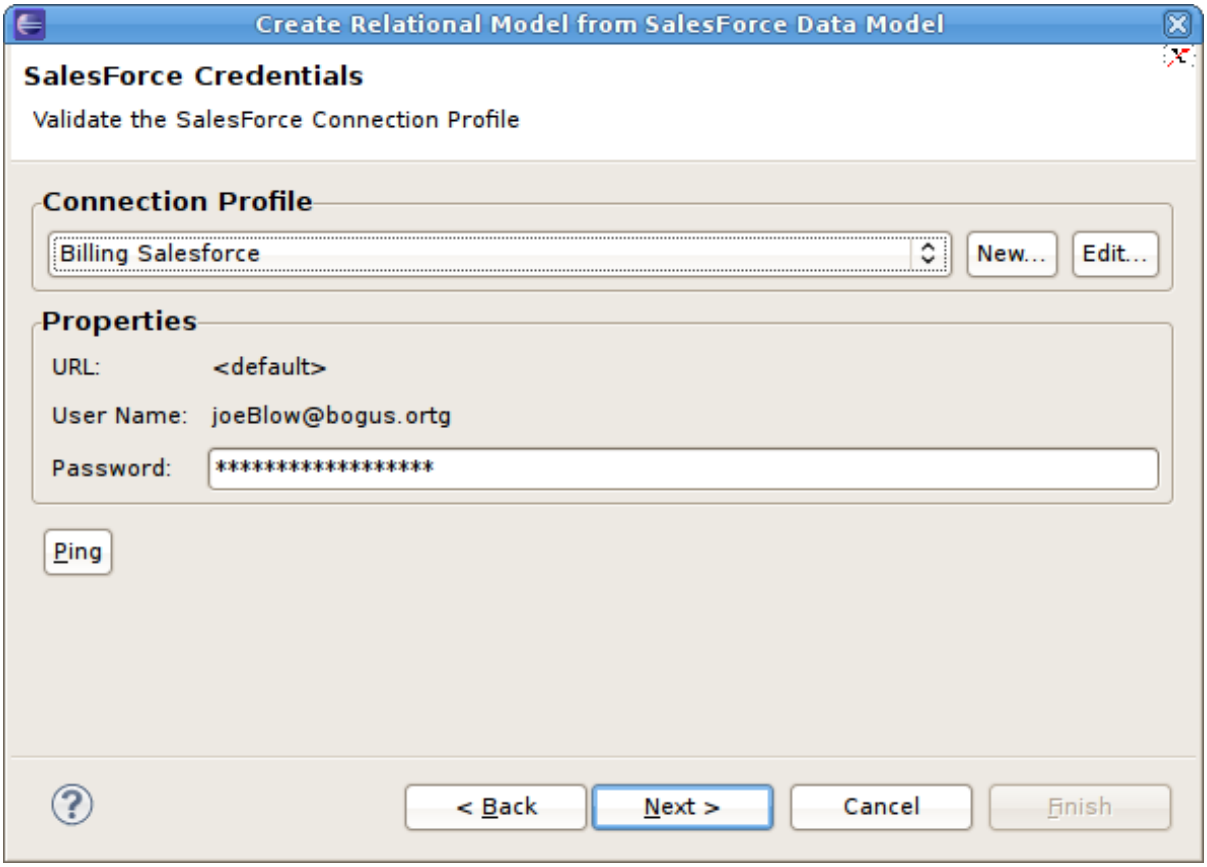


Figure 9.24. Select Salesforce Credentials Dialog

4. After selecting a Connection Profile, enter the password (if not provided). Click **`Next>`** to display the Salesforce Objects selection page.

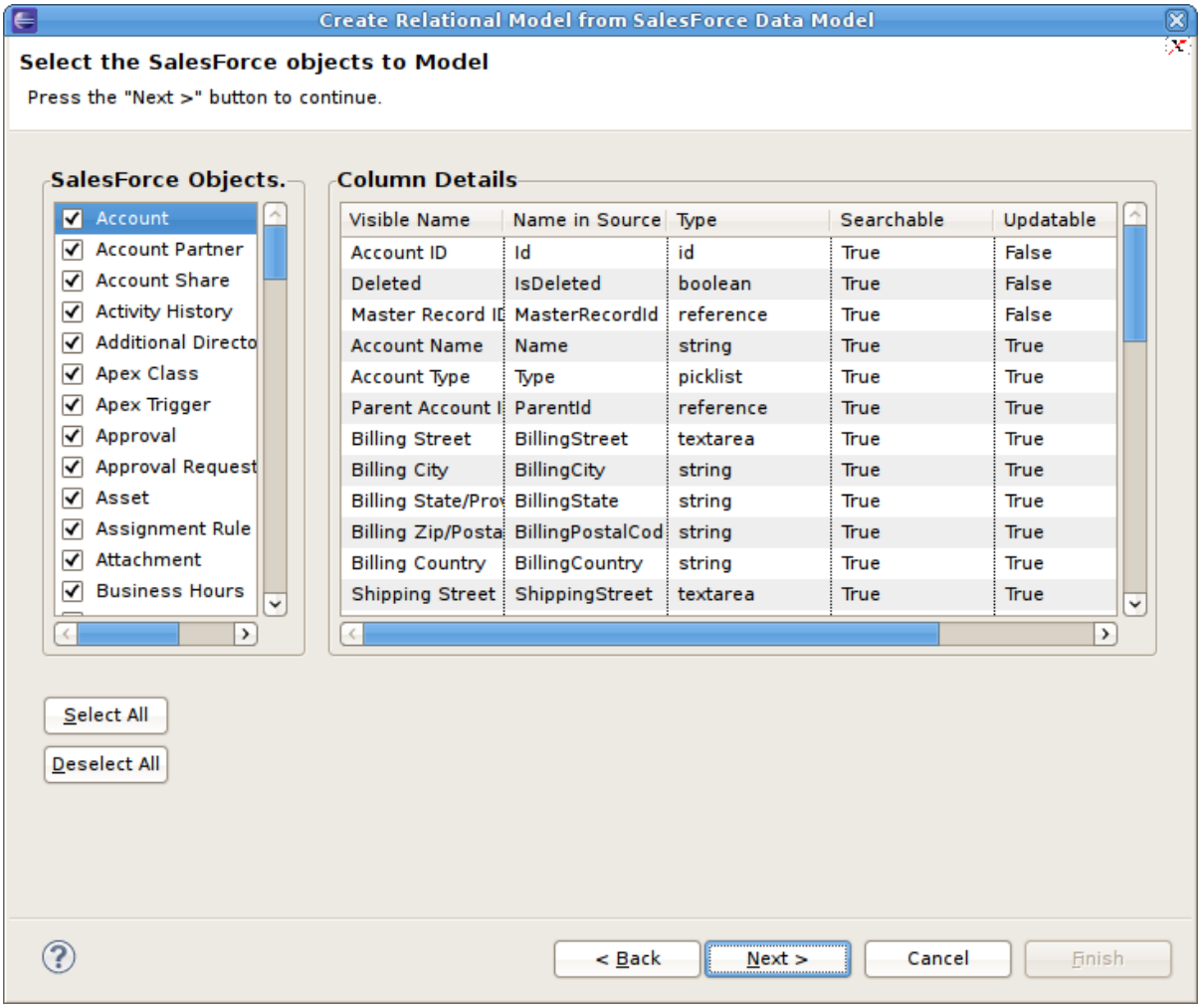


Figure 9.25. Select Salesforce Objects Dialog

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. You have to enter the JNDI name. Click **`Next>`** (or **`Finish`** if enabled).

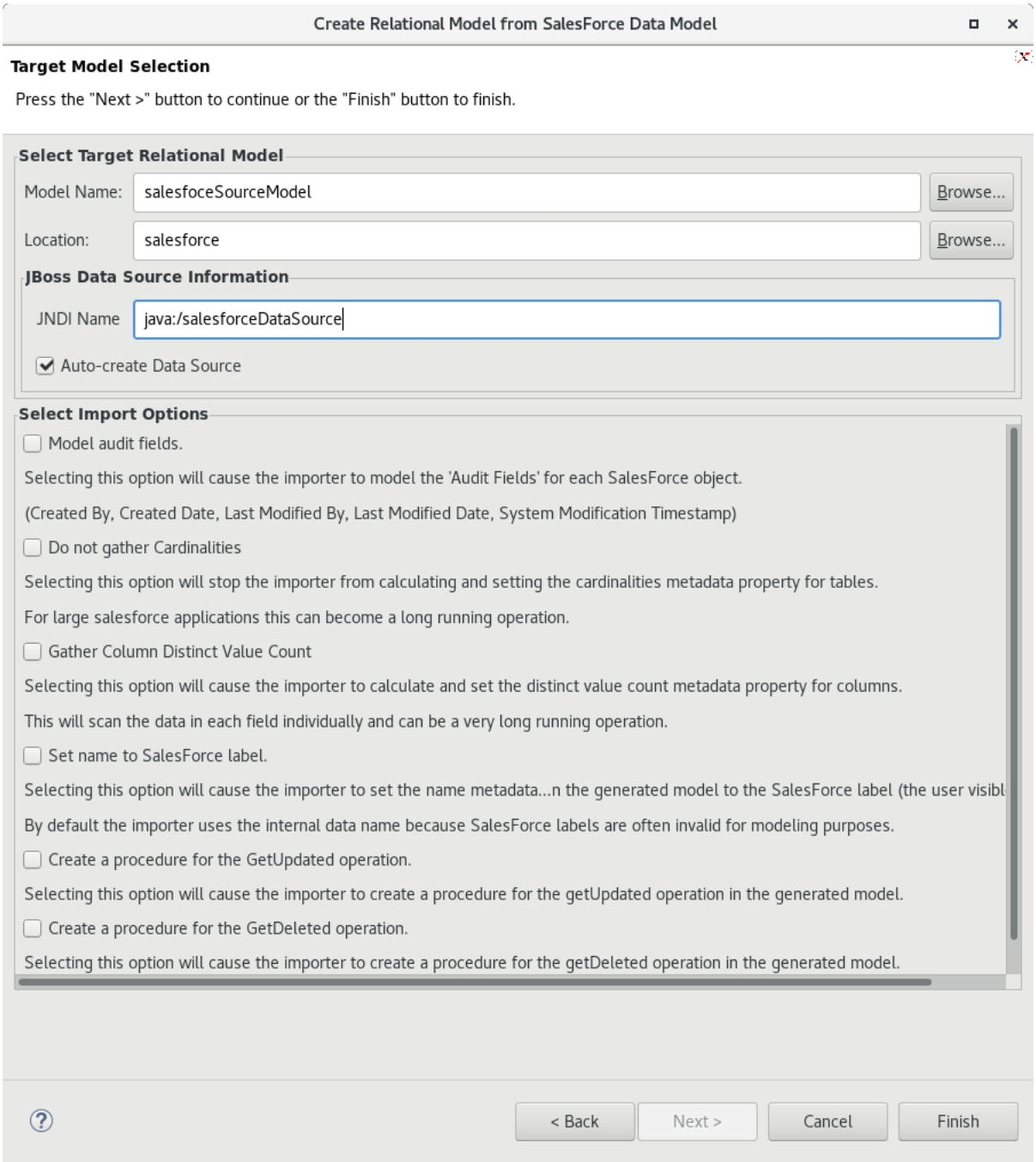


Figure 9.26. Target Model Selection Dialog

6. If you are updating an existing relational model, the next page will be **`Review Model Updates`** page. Any differences. Click **`Finish`** to create your models and tables.

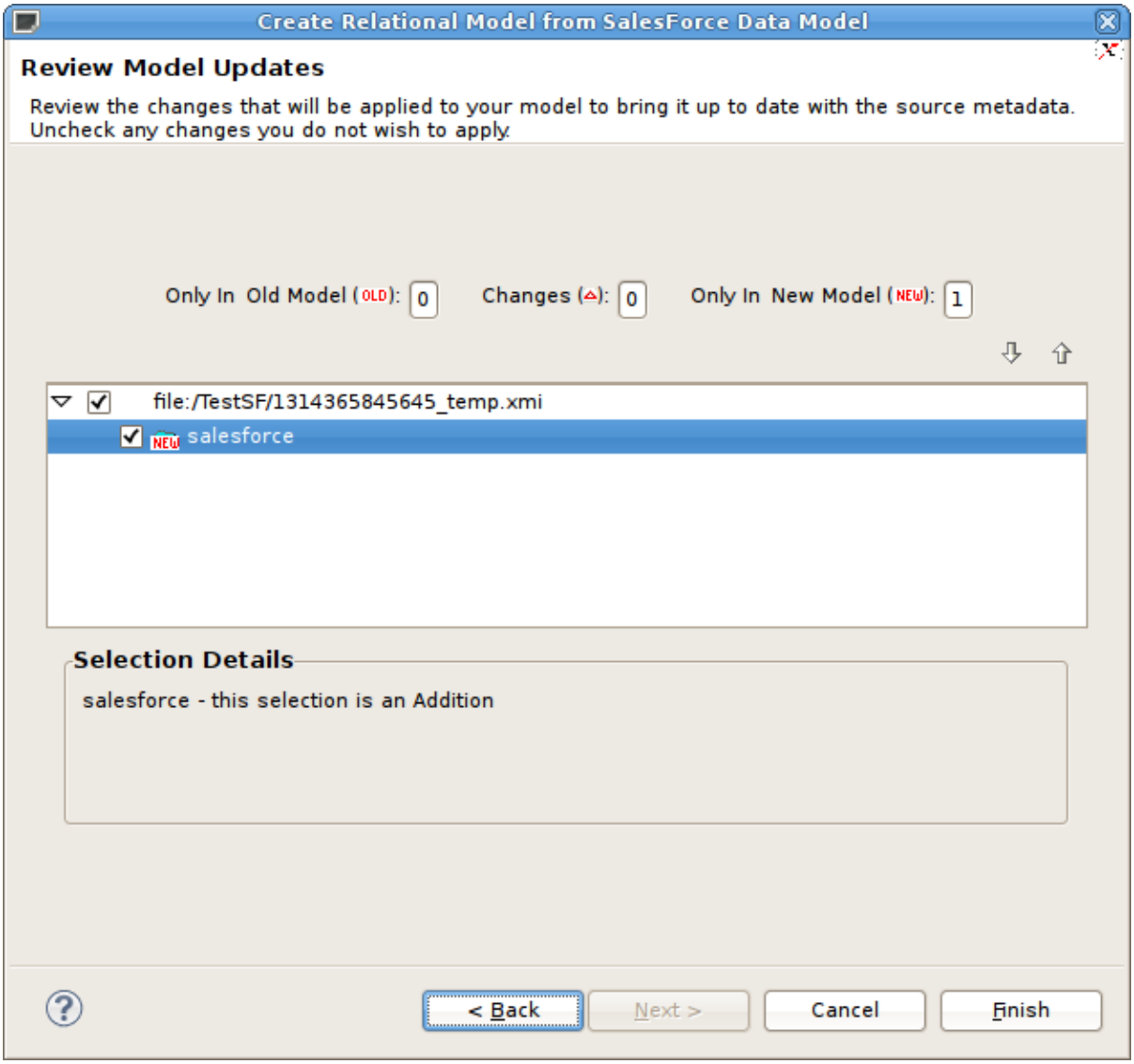


Figure 9.27. Review Model Updates Dialog

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

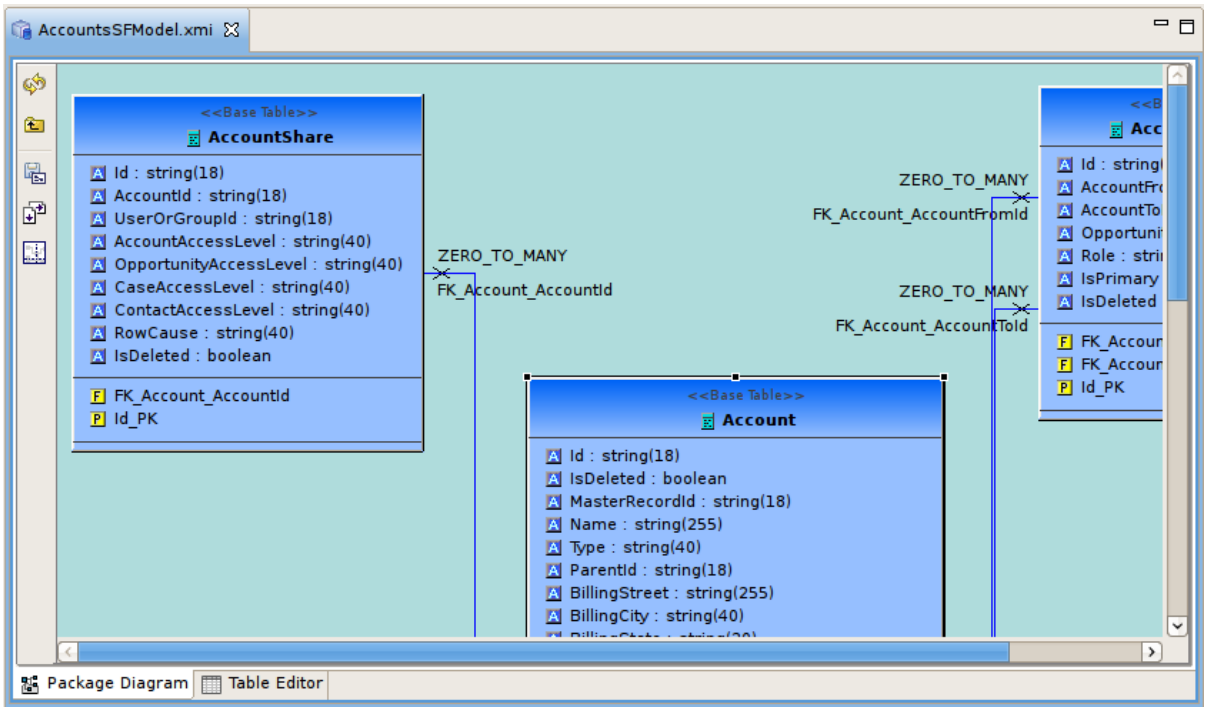


Figure 9.28. New Saleforce Tables Diagram

9.8. IMPORT METADATA FROM TEXT FILE

9.8.1. 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.

1. In **Teiid Designer**, Click **`File > Import...`** action
2. Select the import option **`Teiid Designer > Designer Text File >> Source or View Models`** and click **`Next>`**.
3. Select an import type from the drop-down menu shown below.

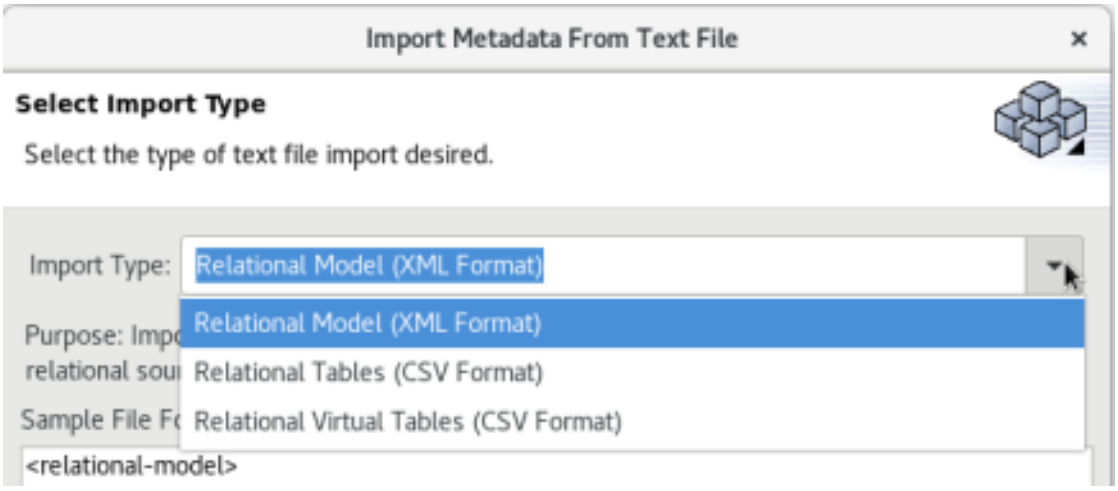


Figure 9.29. Import Wizard

These steps are required for each type are defined below:

- Relation Model Text Import
- Relational Table Text Import
- Virtual Table Text Import

9.8.2. Import Relational Model (XML Format)

To create relational tables from imported XML text file metadata, perform steps 1 to 3 from the Import Metadata from Text File section and then perform following steps:

1. Select the **`Relational Model (XML Format)`** import type, then click **`Next >`**.

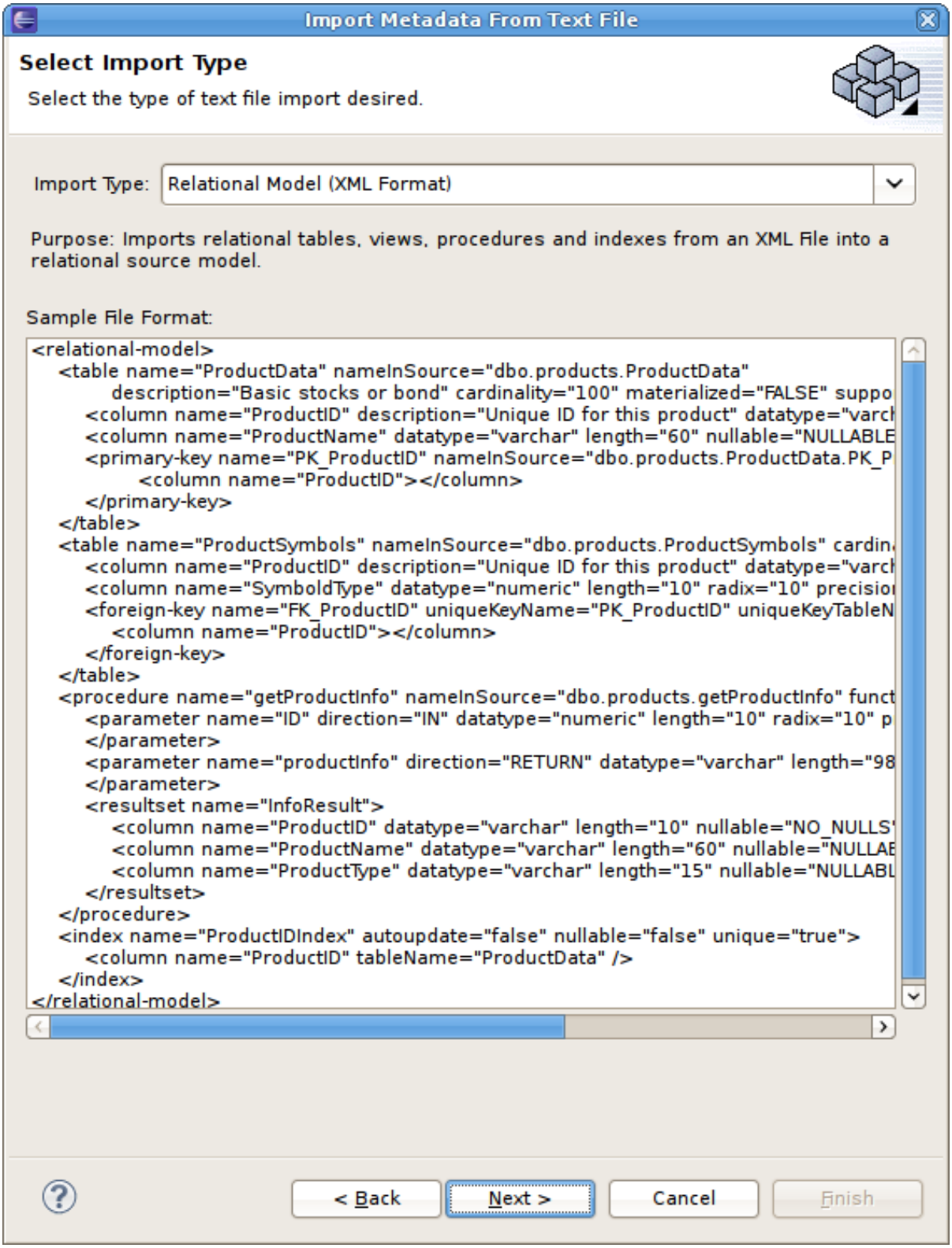


Figure 9.30. Select Import Type - Relational Model (XML Format)

2. 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`**.

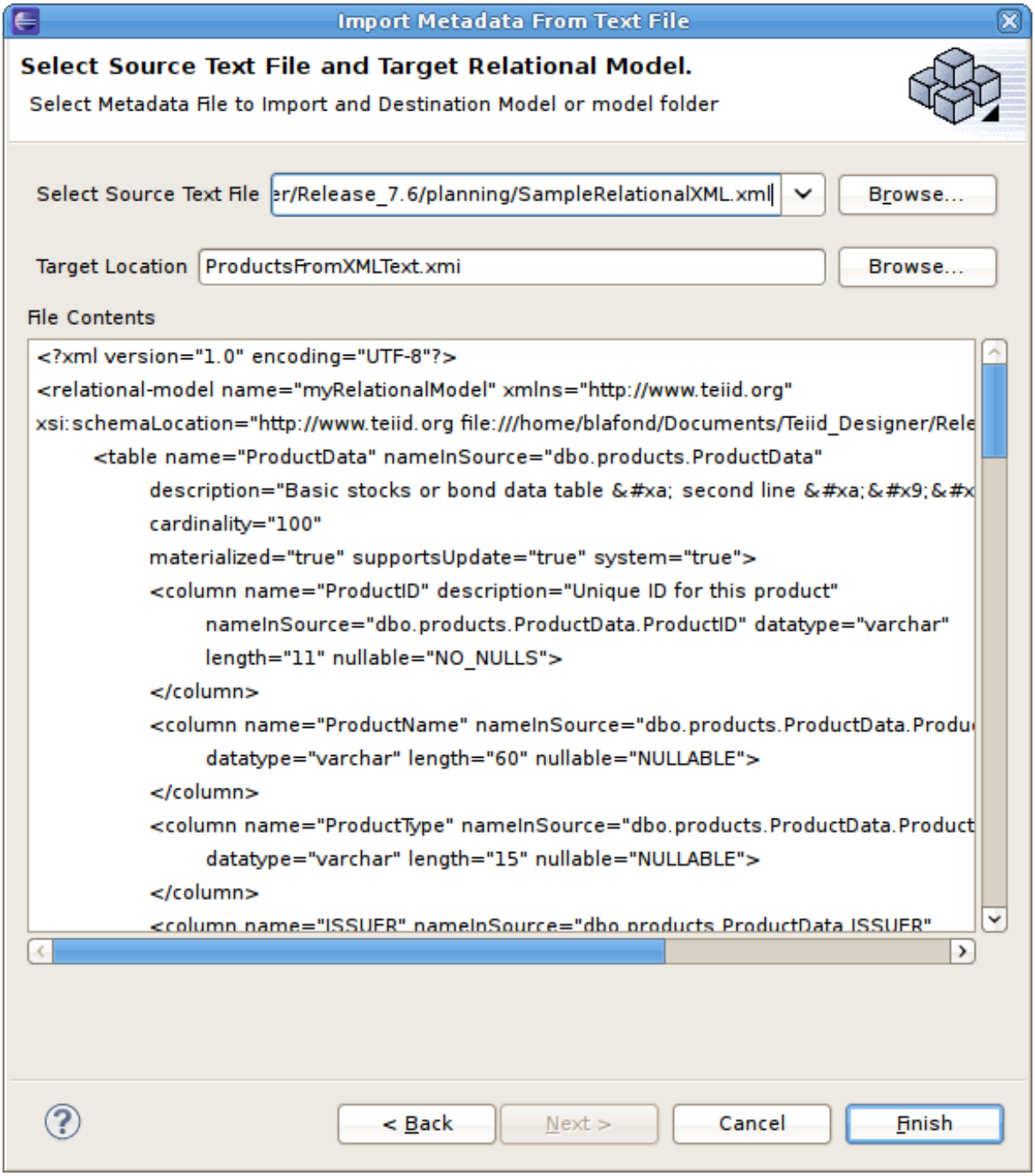


Figure 9.31. Select Source Text File and Target Relational Model Page

3. 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.

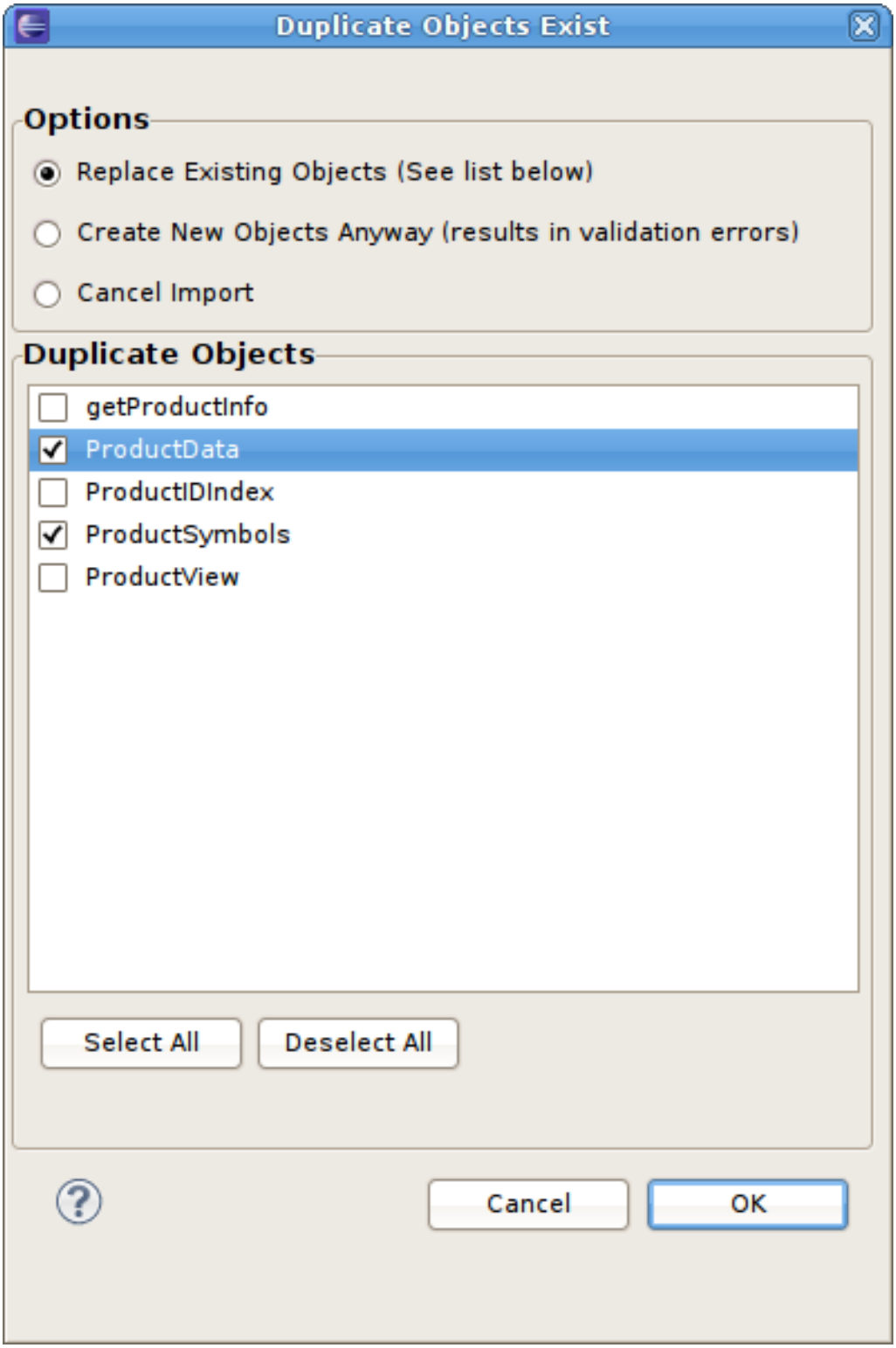


Figure 9.32. Duplicate Objects Dialog

9.8.3. Import Relational Tables (CSV Format)

To create relational tables from imported text file metadata, perform steps 1 to 3 from the Import Metadata from Text File section and then perform following steps:

1. Select the **`Relational Tables (CSV Format)`** import type, then click **`Next >`**.

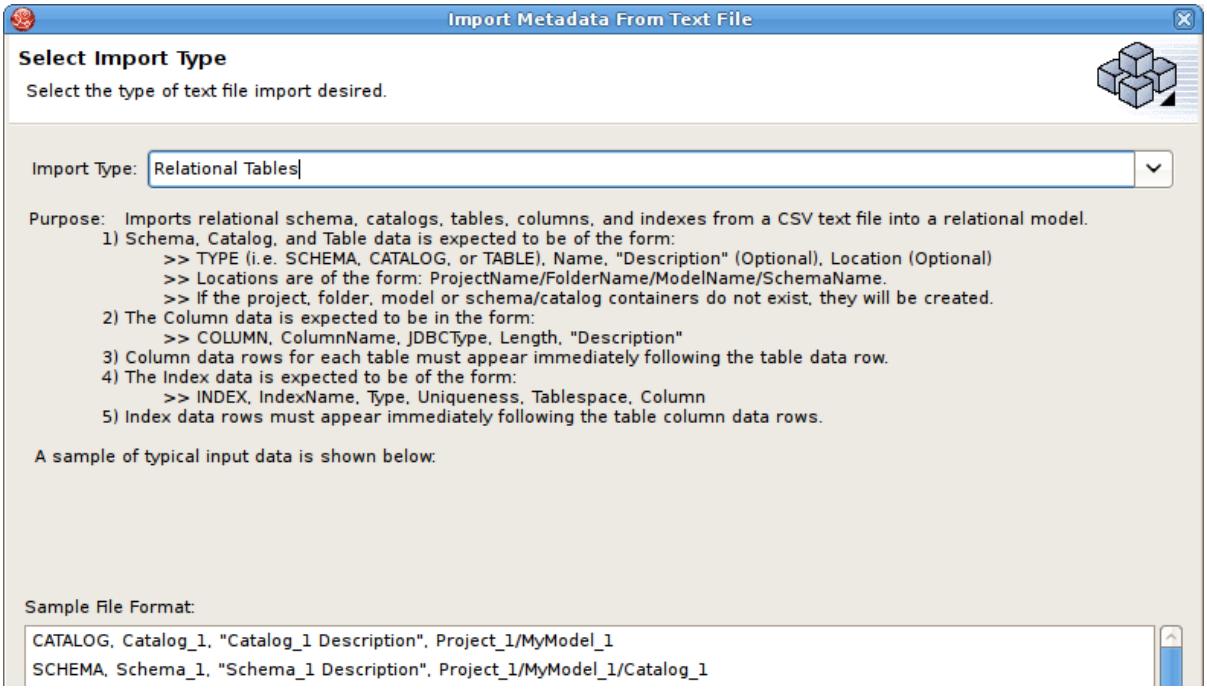


Figure 9.33. Select Import Type - Relational Tables (CSV Format)

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

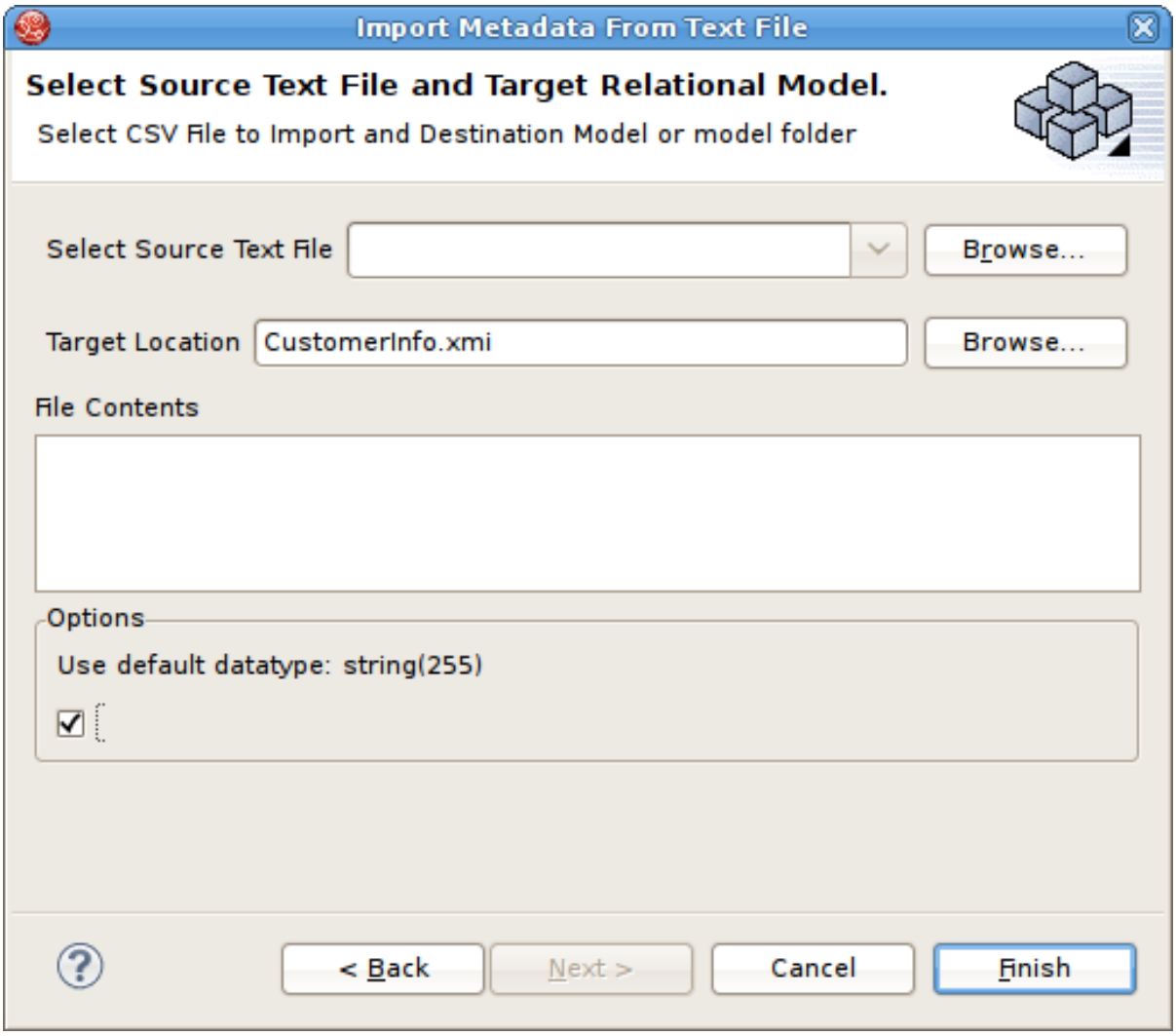


Figure 9.34. Select Source Text File and Target Relational Model

3. 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.

4. Select any additional options and click **`Finish`**.

9.8.4. Import Relational View Tables (CSV Format)

To create relational virtual tables from imported text file metadata, perform steps 1 to 3 from the Import Metadata from Text File section and then perform following steps:

1. Select the **`Relational Virtual Tables (CSV Format)`** import type, then click **`Next >`**.

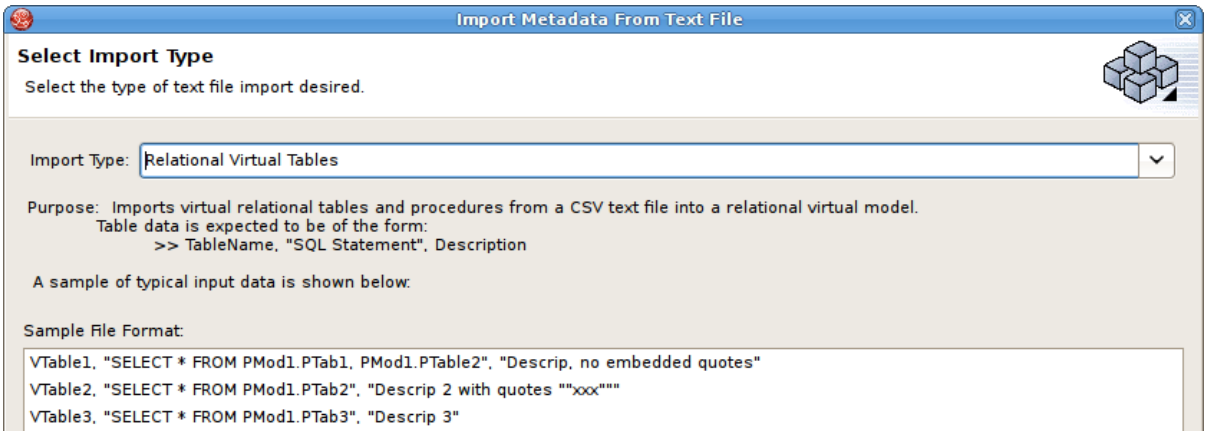


Figure 9.35. Select Import Type - Relational Virtual Tables (CSV Format)

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

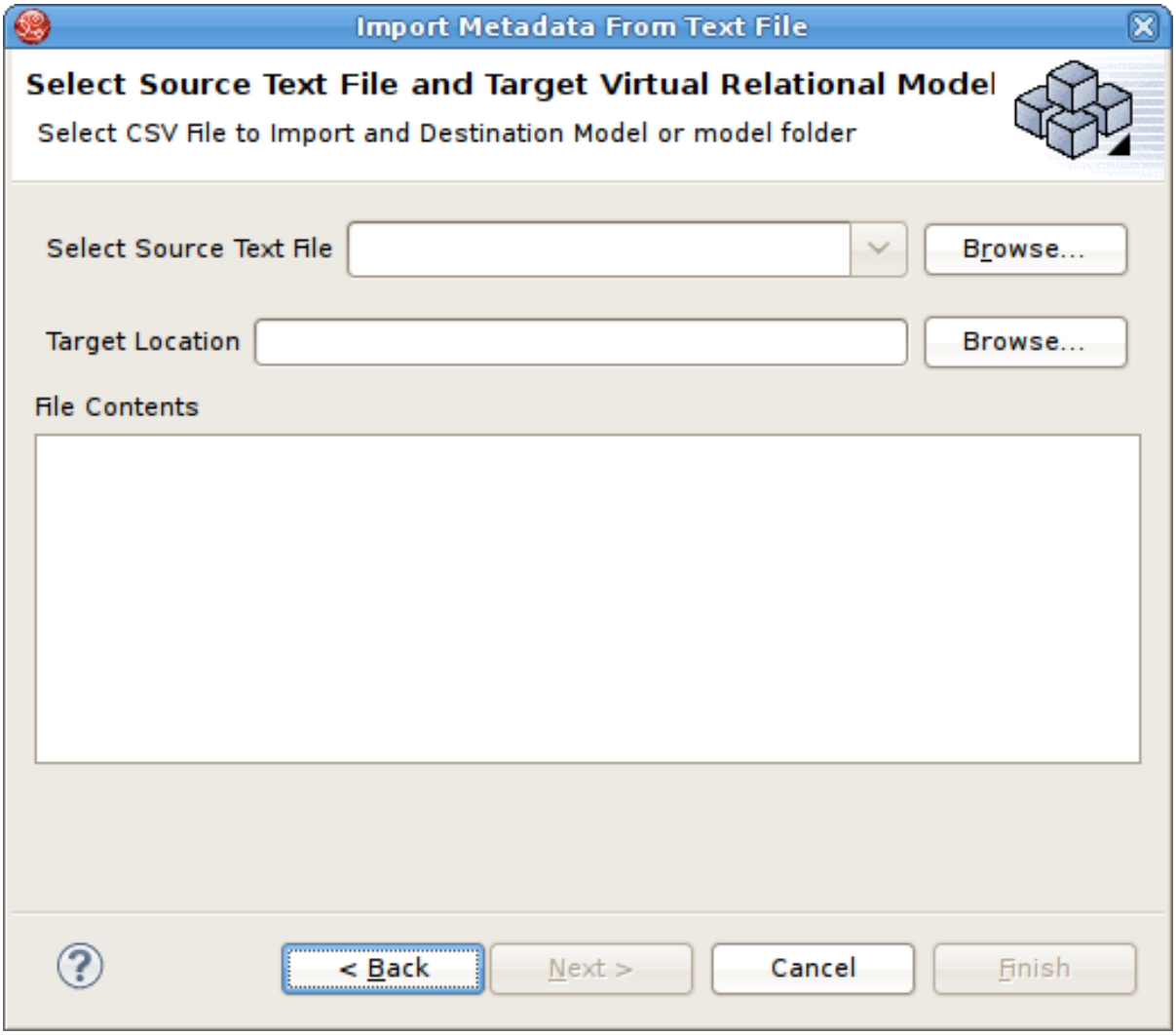


Figure 9.36. Select Source Text File and Target Virtual Relational Model

3. 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.

4. Click **`Finish`**.

9.9. IMPORT WSDL INTO RELATIONAL MODELS

9.9.1. Import WSDL into Relational Models

You can turn your WSDL file (local or URL) into a queryable relational procedures that represent your desired request and response web service structure defined through your WSDL's schema definition. You can access this wizard by clicking **`Import...`** action and then selecting the **`Teiid Designer > Web Service Source >> Source and View Model (SOAP)`** option. Web Services Connection Profile is defined by a WSDL file in your workspace or defined by a URL. Designer will interpret the WSDL, locate any associated or dependent XML schema files, generate a physical model to invoke the service, and generate virtual models containing procedures to build and parse the XML declared as the service messages.

To create relational models from WSDL use the steps below.

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**

2. Select the import option **`Teiid Designer > Web Service Source >> Source and View Model (SOAP)`** and click **`Next>`**.

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.

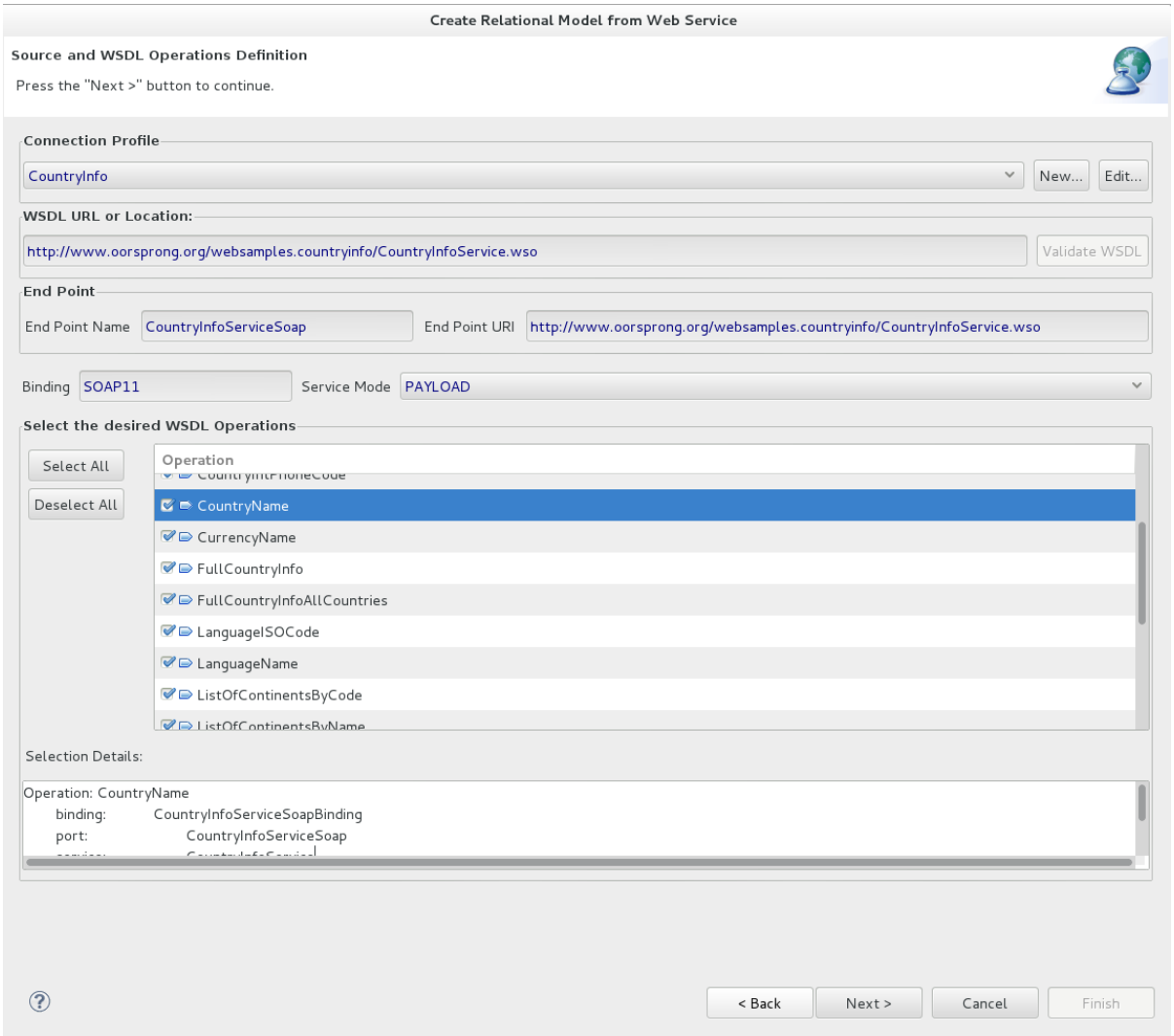


Figure 9.37. WSDL Source Selection

4. Select individual Web Service Operations to model. The default behavior of this page selects all available operations in the tree. Operations can be deselected 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 >`**.

5. The next page entitled **`Model Definition`** 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.

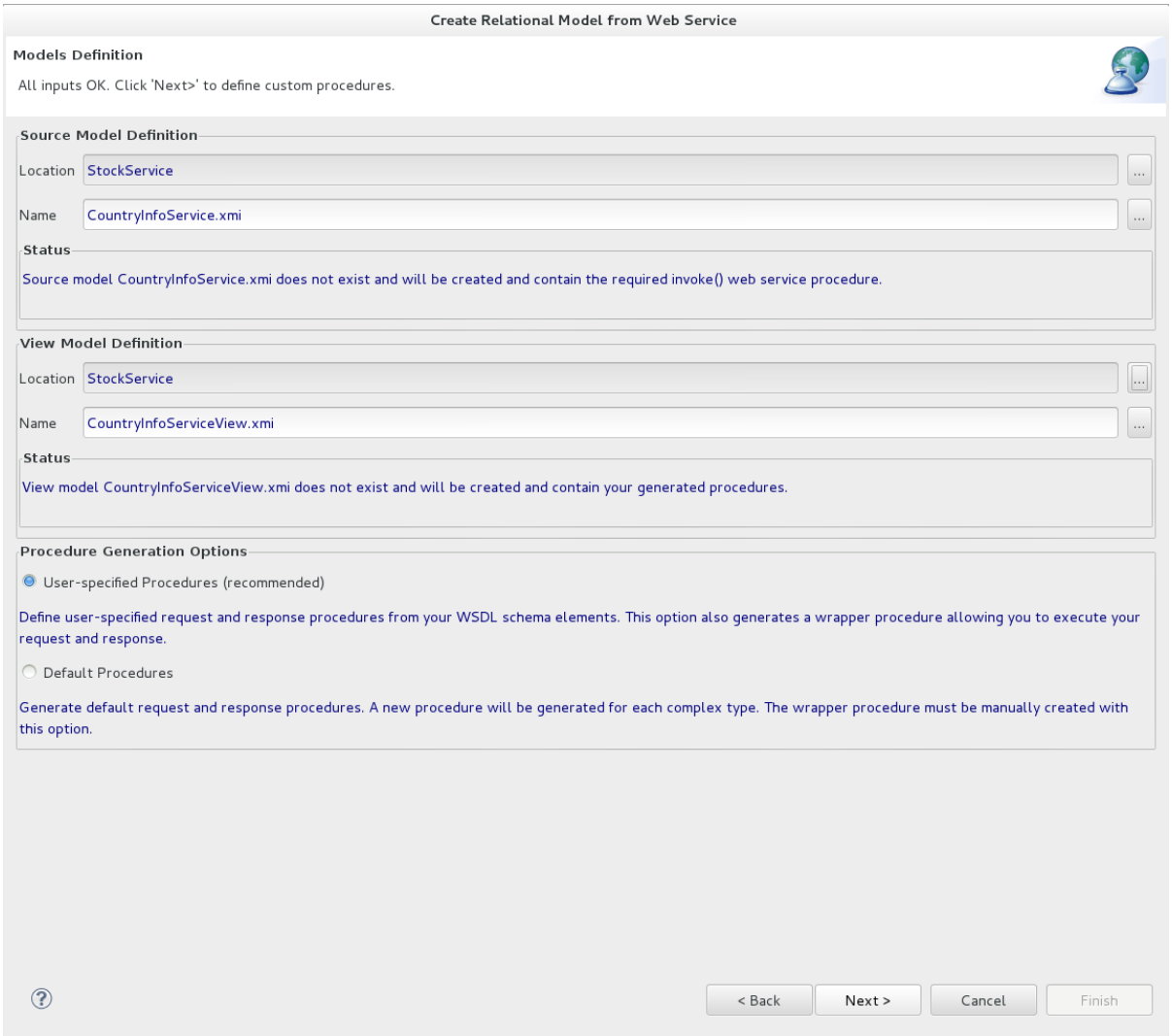


Figure 9.38. WSDL Source Selection

6. On the next page enter the JNDI name and click **`Next >`**.

7. On the next page choose whether the **`User-specified Procedures`** or **`Default Procedures`** should be created and click **`Next >`**.

NOTE

If the **`Default Procedures`** option is selected, the **`Finish`** button is available instead of **`Next >`** button.

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.

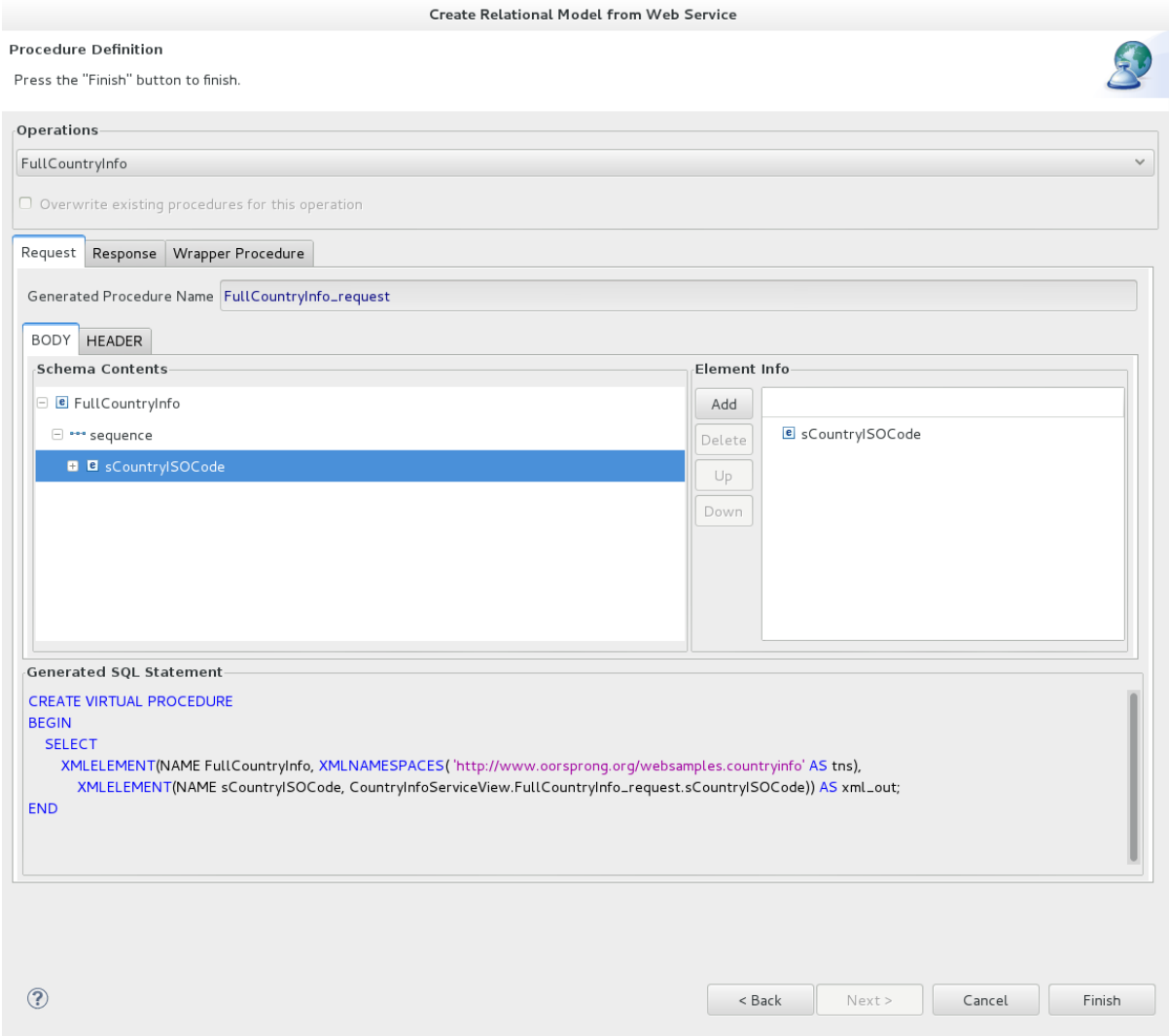


Figure 9.39. Procedure Definition Page

9. Click **`Finish`**. After generation the new models can be found in the specified location in your workspace.

In the Model Explorer you can see the importer created the following a single physical model containing a single procedure called invoke. This model and procedure correspond to the single port declared in the WSDL.

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 Teiid Designer.

9.9.2. 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 **Teiid 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, for example, **`-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.

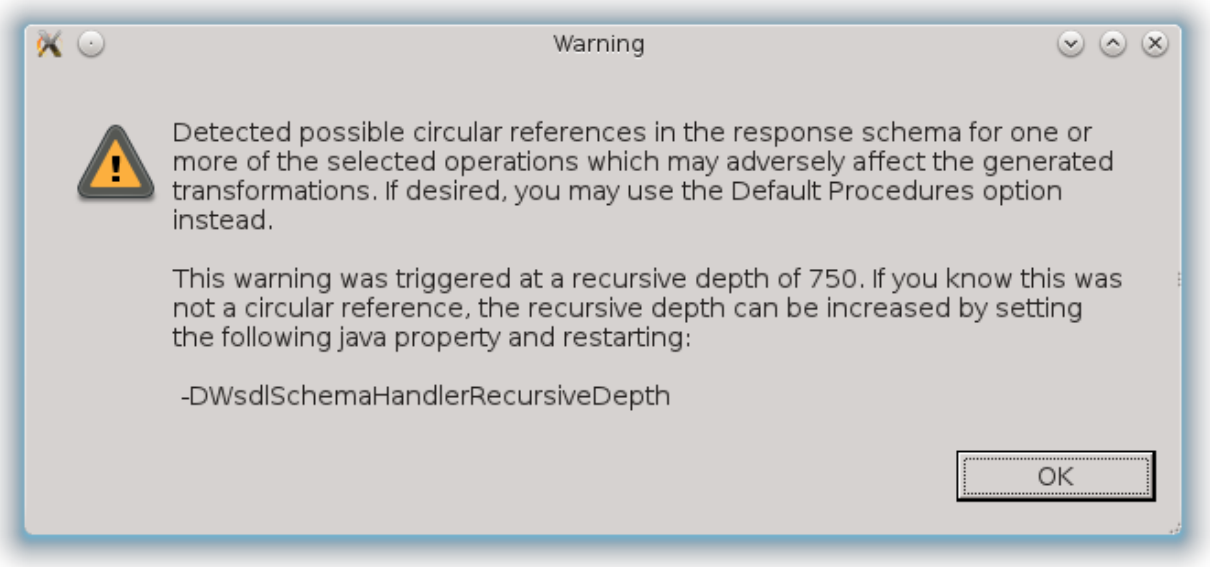


Figure 9.40. Warning message displayed if depth limit has been reached

9.10. IMPORT WSDL INTO WEB SERVICE

9.10.1. 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.

9.10.2. Import WSDL from Workspace Location

You can create a Web Service model by selecting a WSDL file from your workspace.

1. Click the **File > Import...** action in the toolbar or select a project, folder or model in the Model Explorer's tree, right-click and choose **Import...**

2. Select the import option **Teiid Designer > WSDL File or URL >> Web Service Model** option and click **Next>**.

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

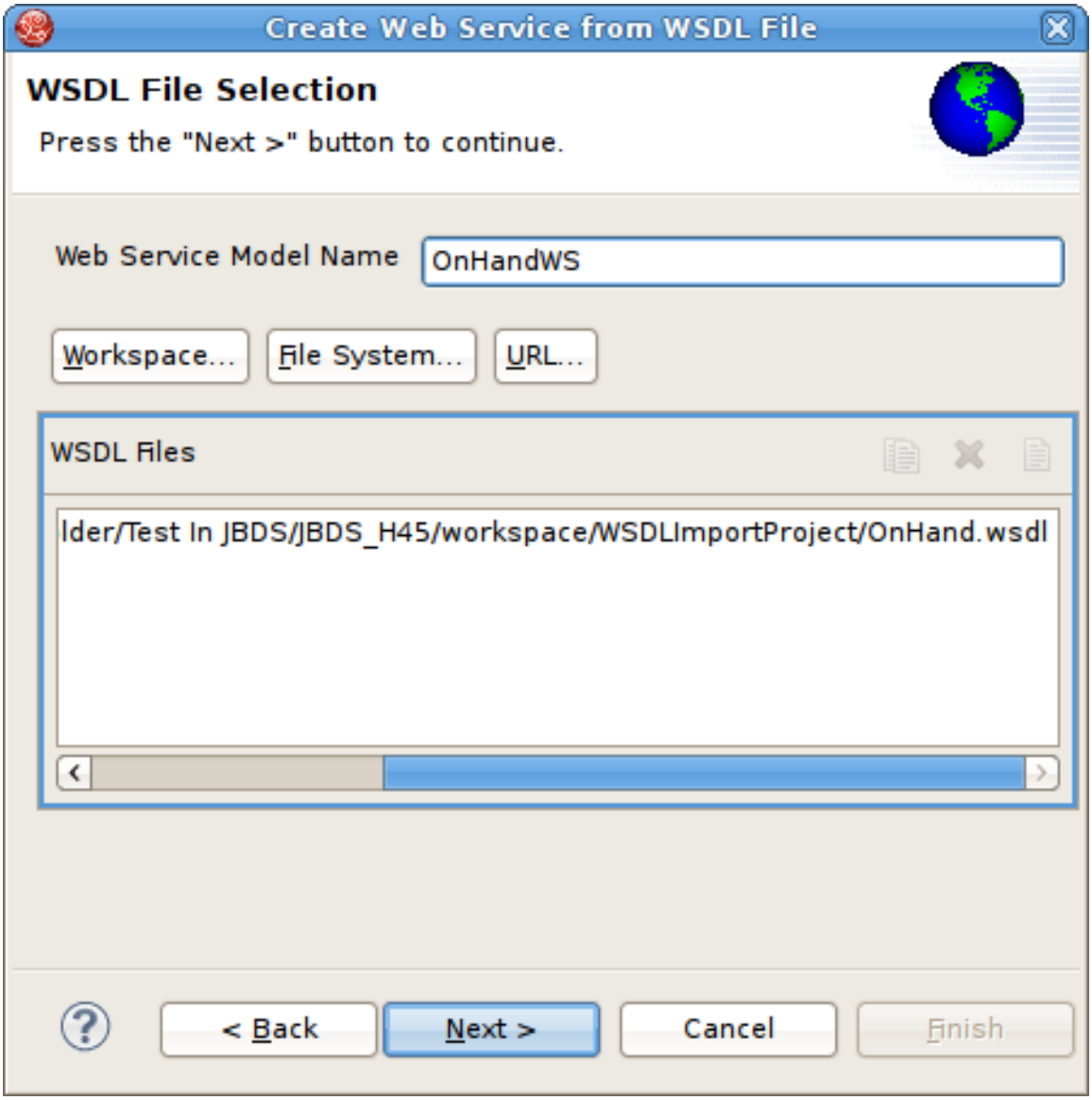


Figure 9.41. WSDL File Selection Dialog

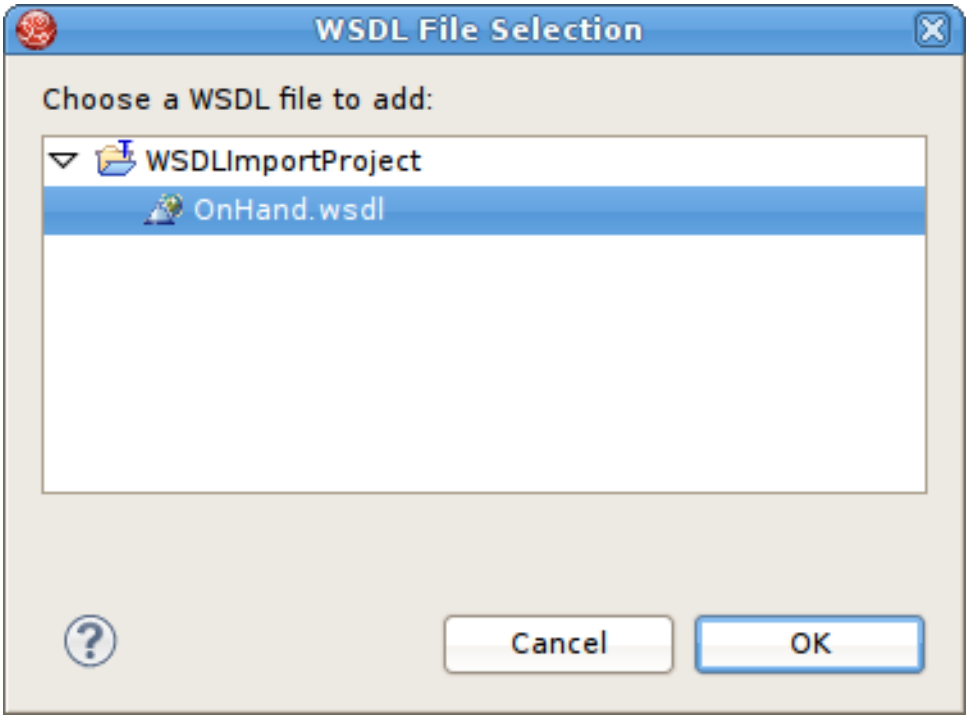


Figure 9.42. WSDL File Workspace Selection Dialog

NOTE

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

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, click **`Next`** to proceed, or **`Finish`** (if enabled) to complete with default options.

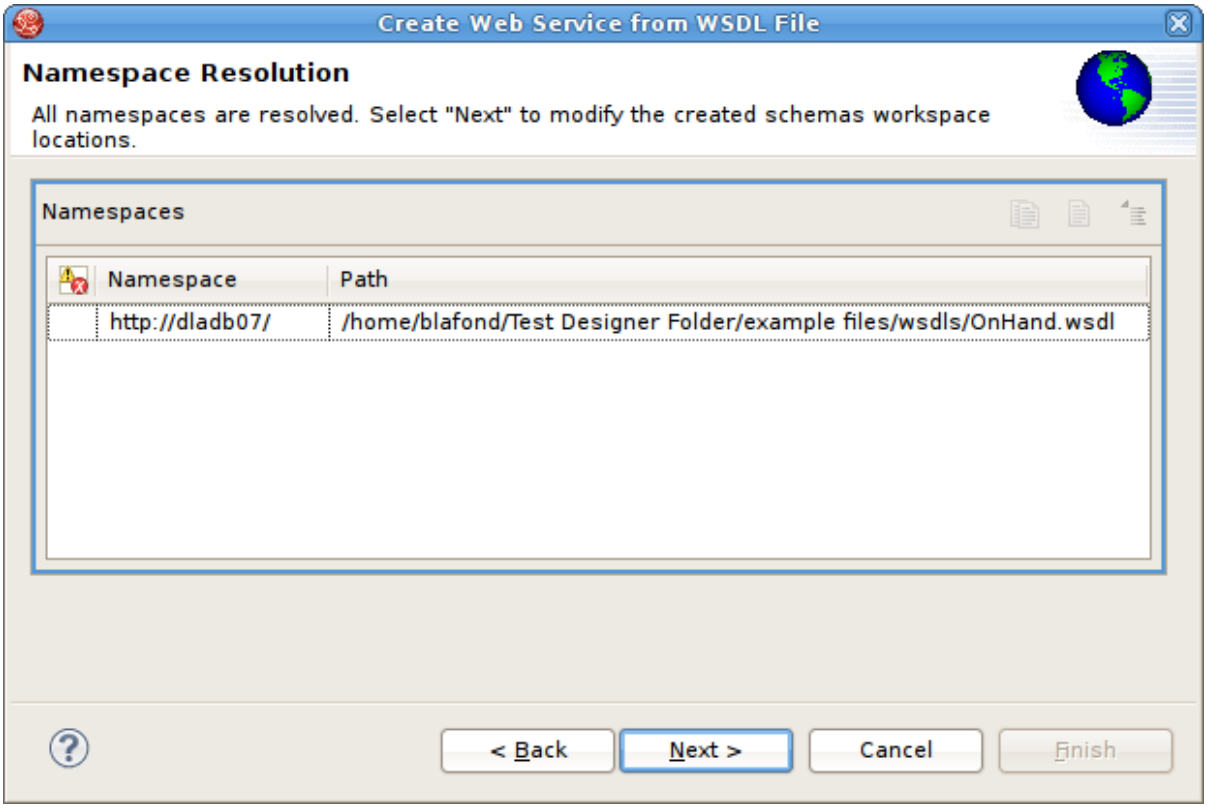


Figure 9.43. Namespace Resolution Dialog

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.

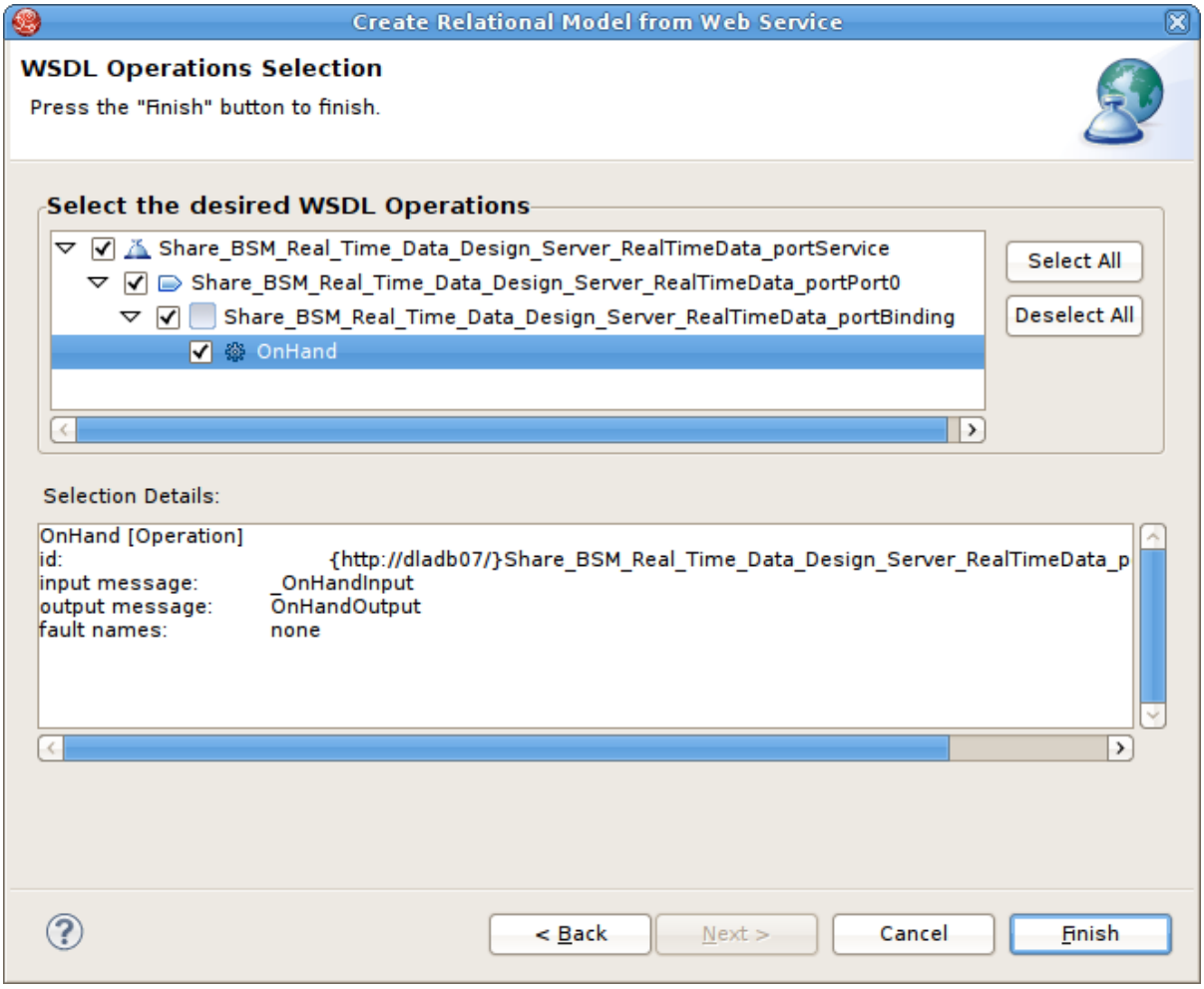


Figure 9.44. Namespace Resolution Dialog

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, click **`Next`** to proceed, or **`Finish`** to complete with default option.

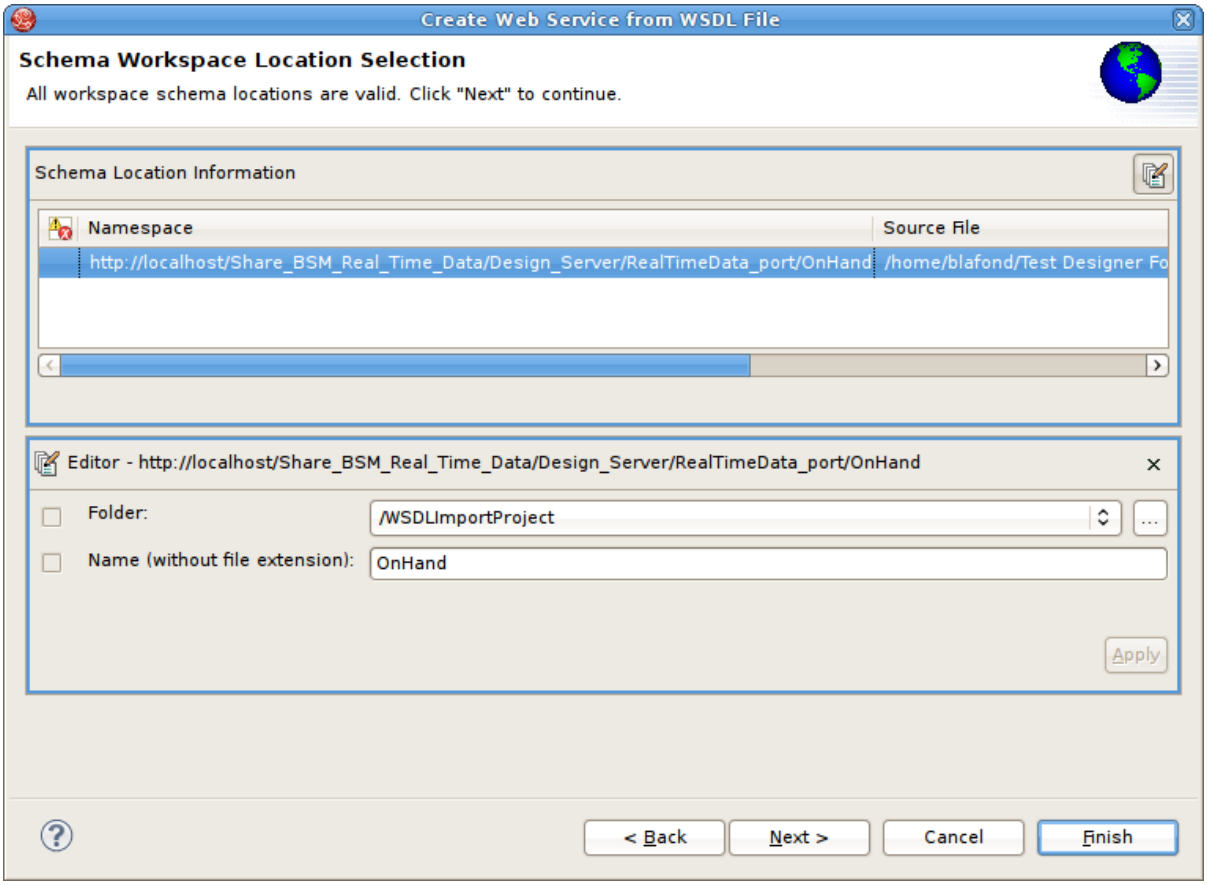


Figure 9.45. Namespace Resolution Dialog

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. Enter desired name or use the default name provide. Click **`Finish`** to complete.

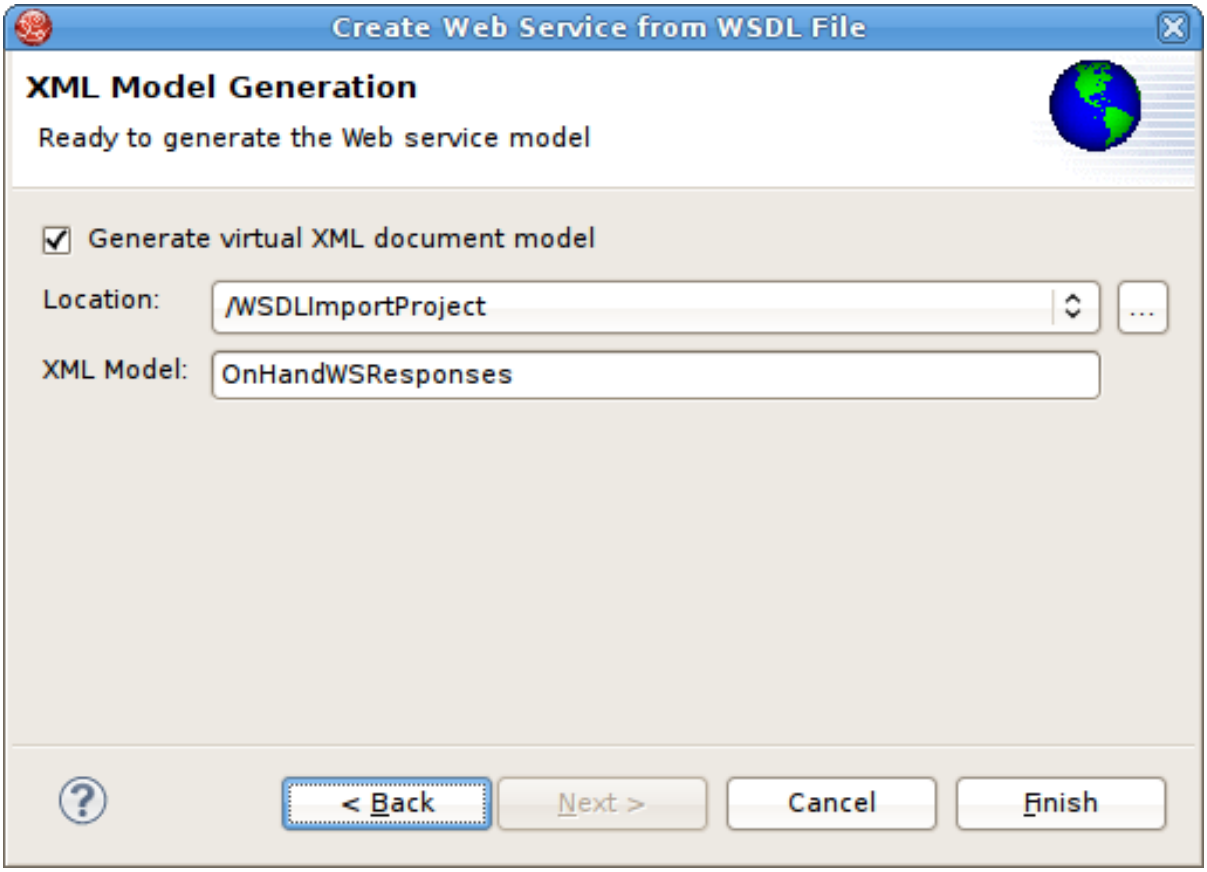


Figure 9.46. Namespace Resolution Dialog

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.

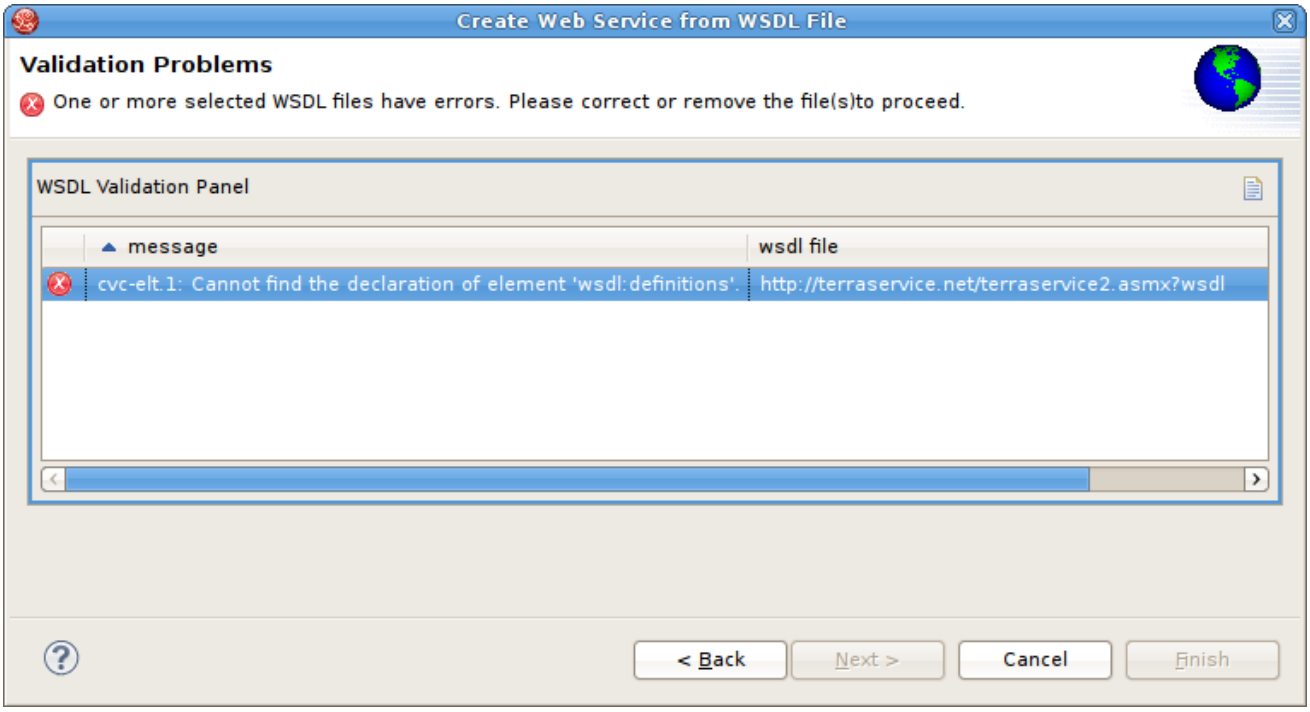


Figure 9.47. WSDL Validation Problems Dialog

9.10.3. Import WSDL from File System Location

You can create a Web Service model by selecting a WSDL file from your local file system.

1. Click the **File > Import...** action in the toolbar or select a project, folder or model in the Model Explorer's tree, right-click and choose **Import...**

2. Select the import option **Teiid Designer > WSDL File or URL >> Web Service Model** and click **Next>**.

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

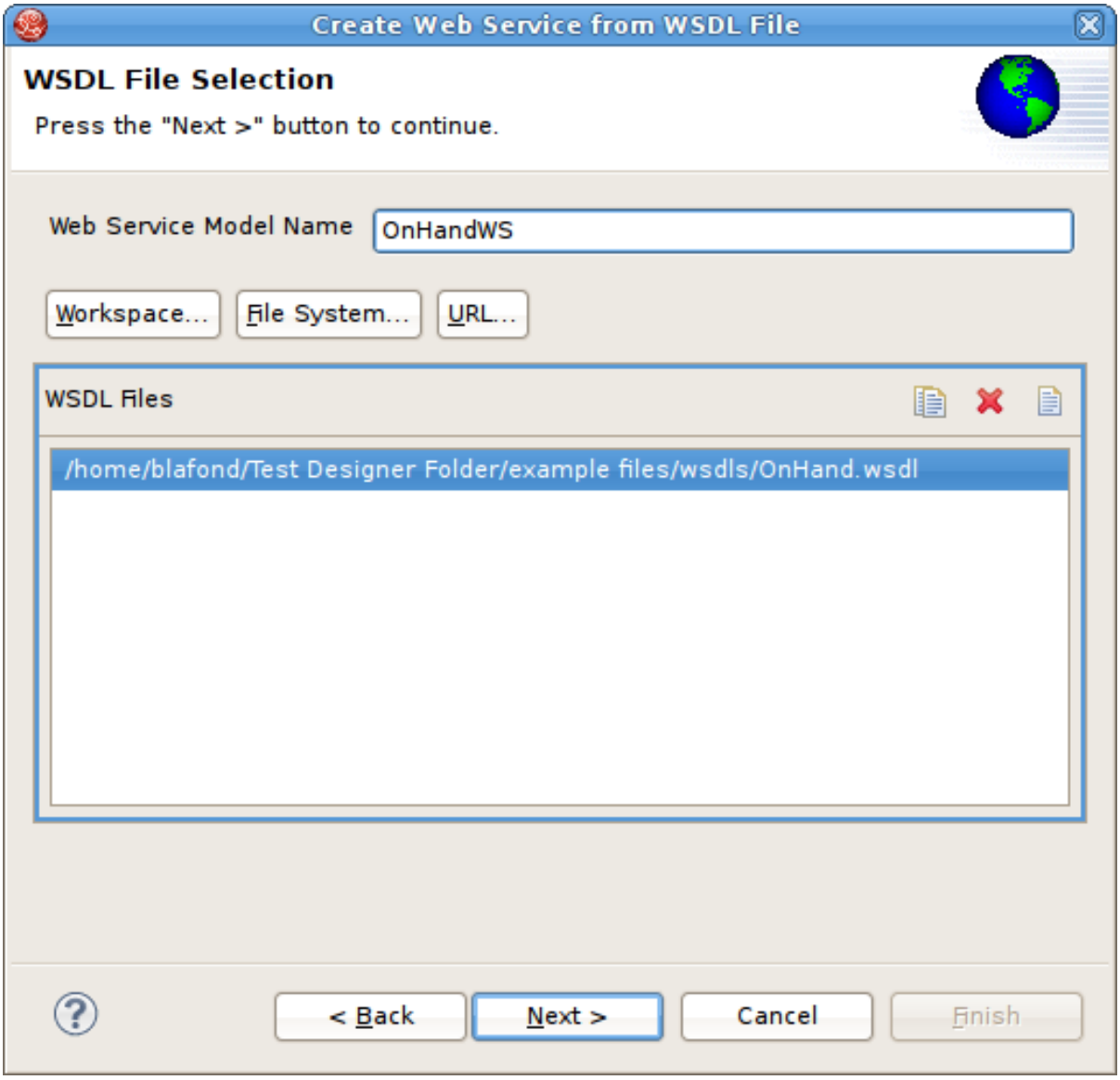


Figure 9.48. WSDL File Selection Dialog

NOTE

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

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, click **`Next`** to proceed, or **`Finish`** (if enabled) to complete with default options.

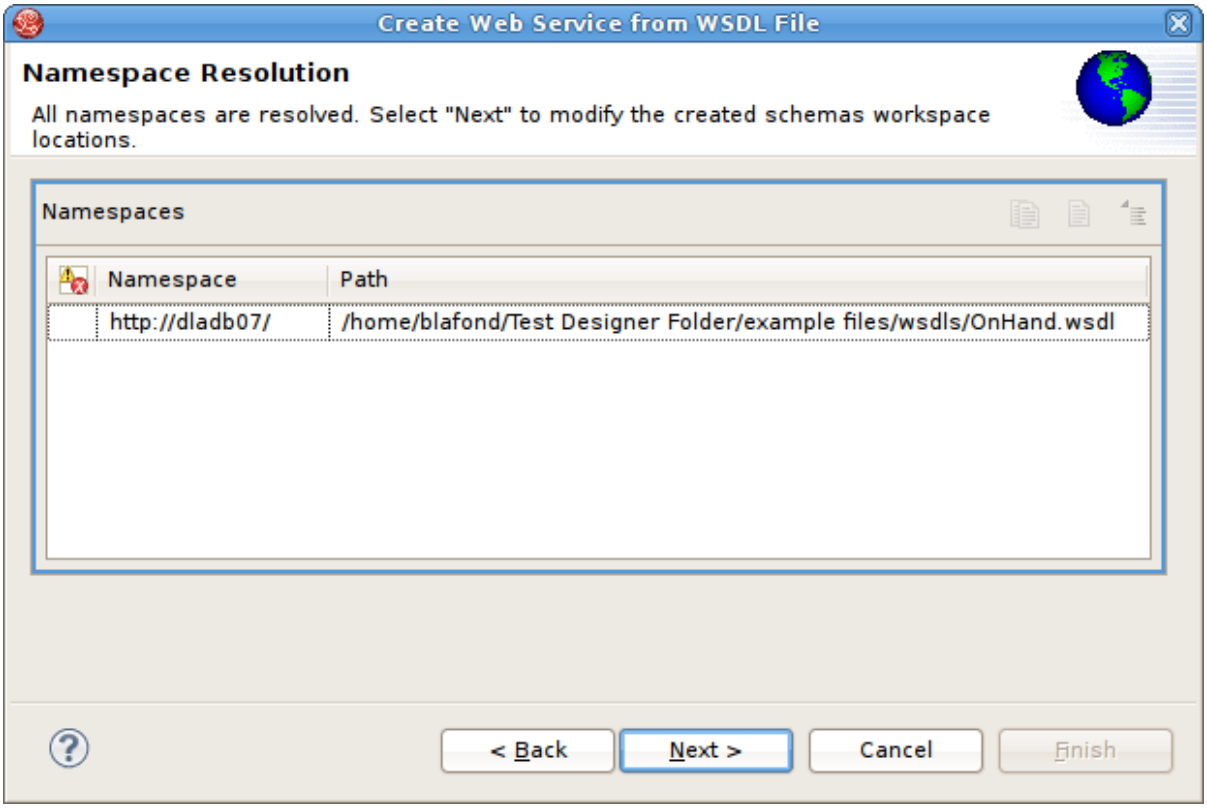


Figure 9.49. Namespace Resolution Dialog

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.

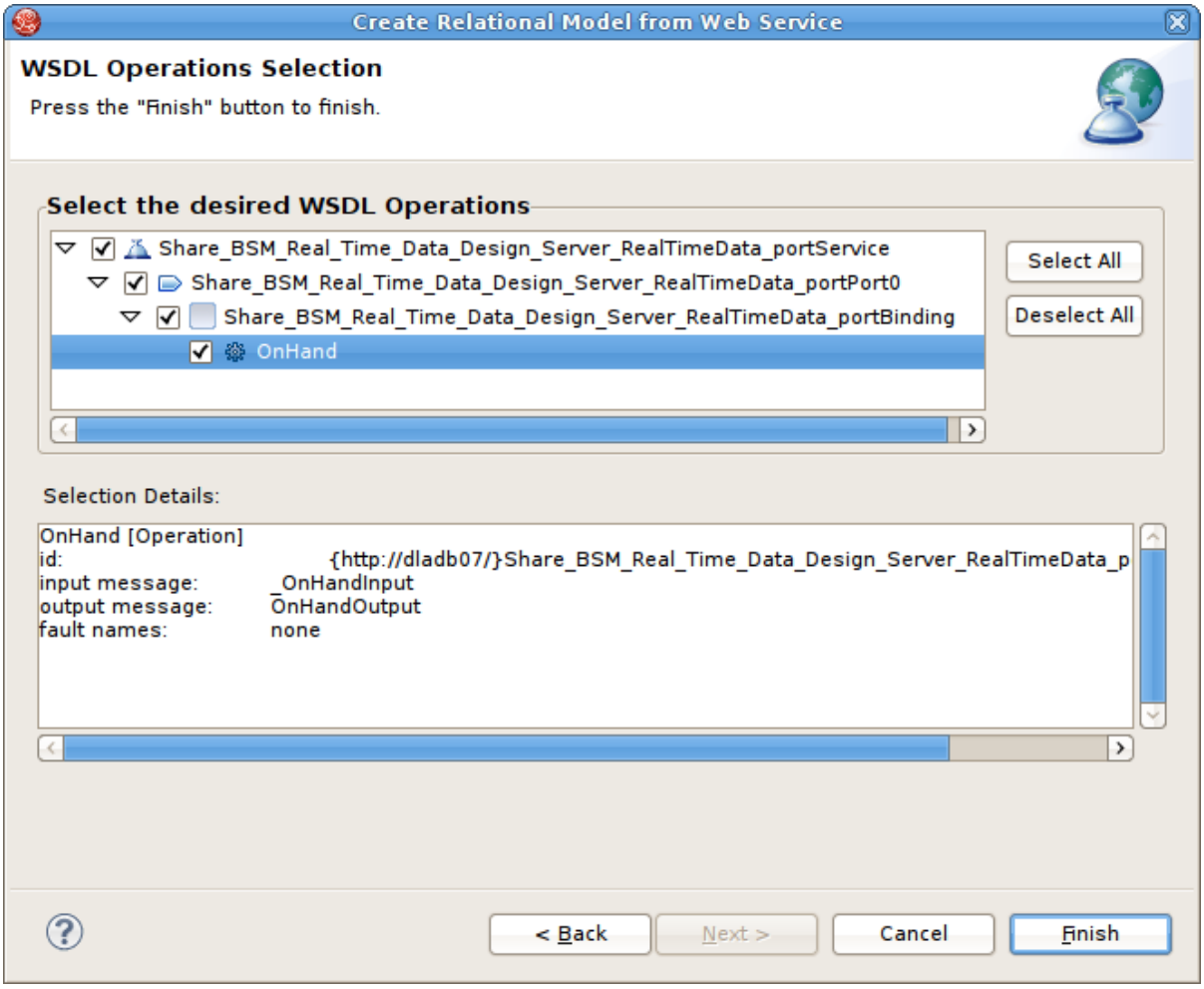


Figure 9.50. Namespace Resolution Dialog

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, click **`Next`** to proceed, or **`Finish`** to complete with default option.

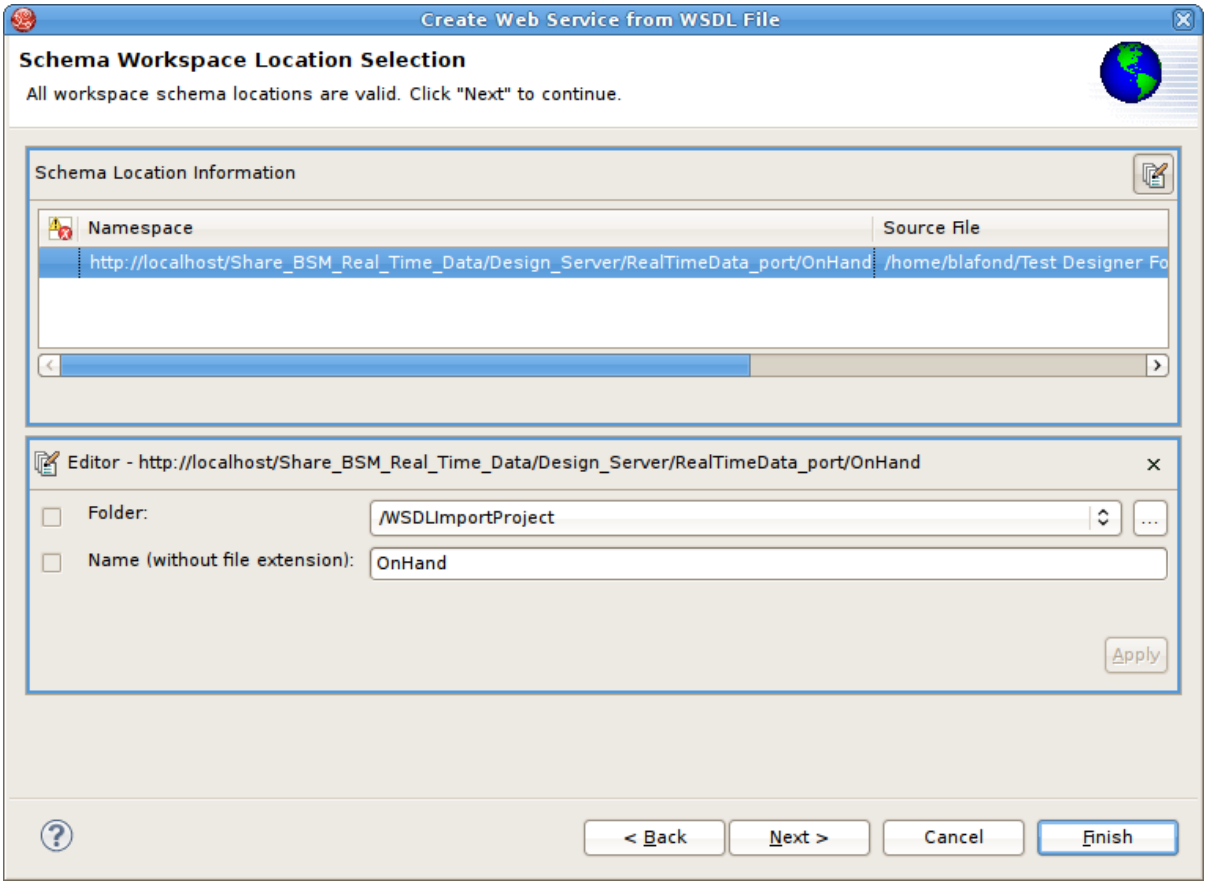


Figure 9.51. Namespace Resolution Dialog

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 selected. Enter the desired name or use the default name provided. Click **`Finish`** to complete.

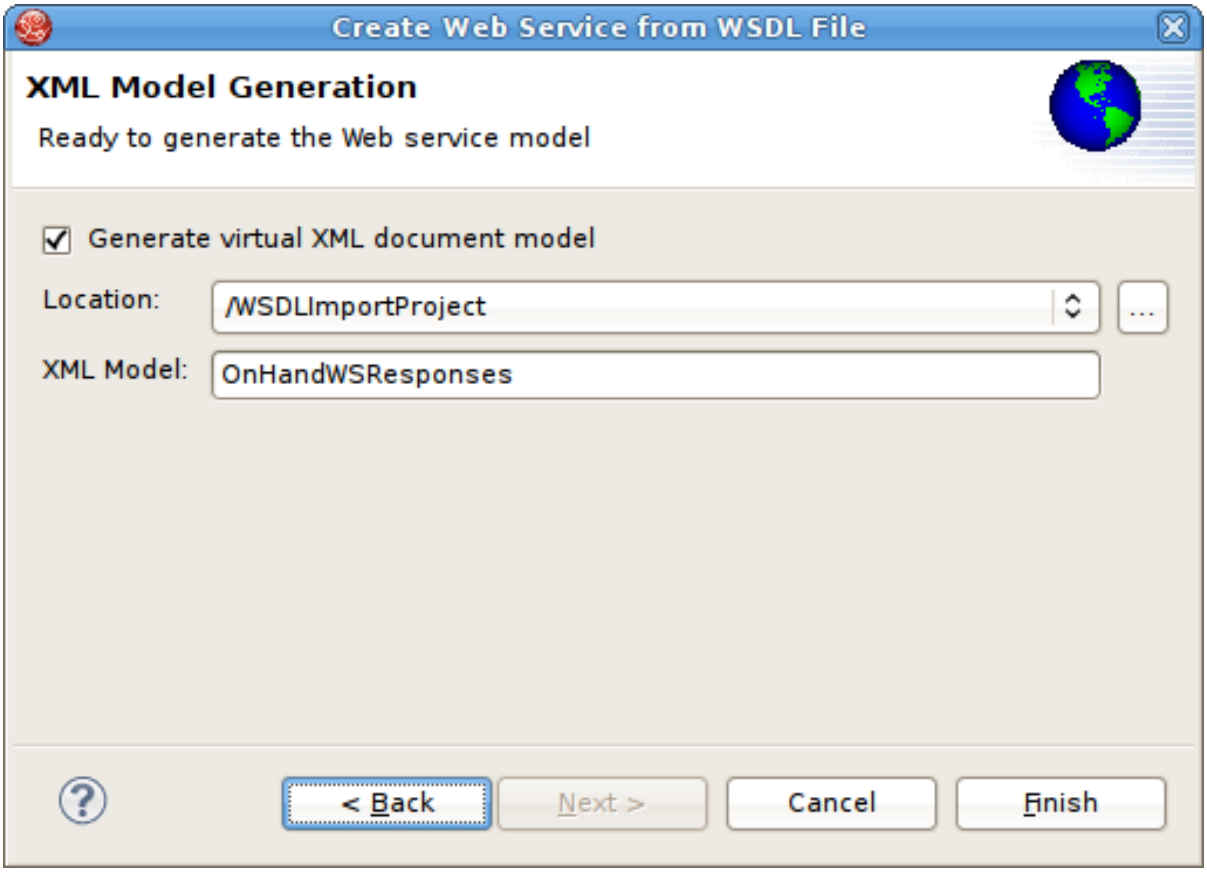


Figure 9.52. Namespace Resolution Dialog

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.

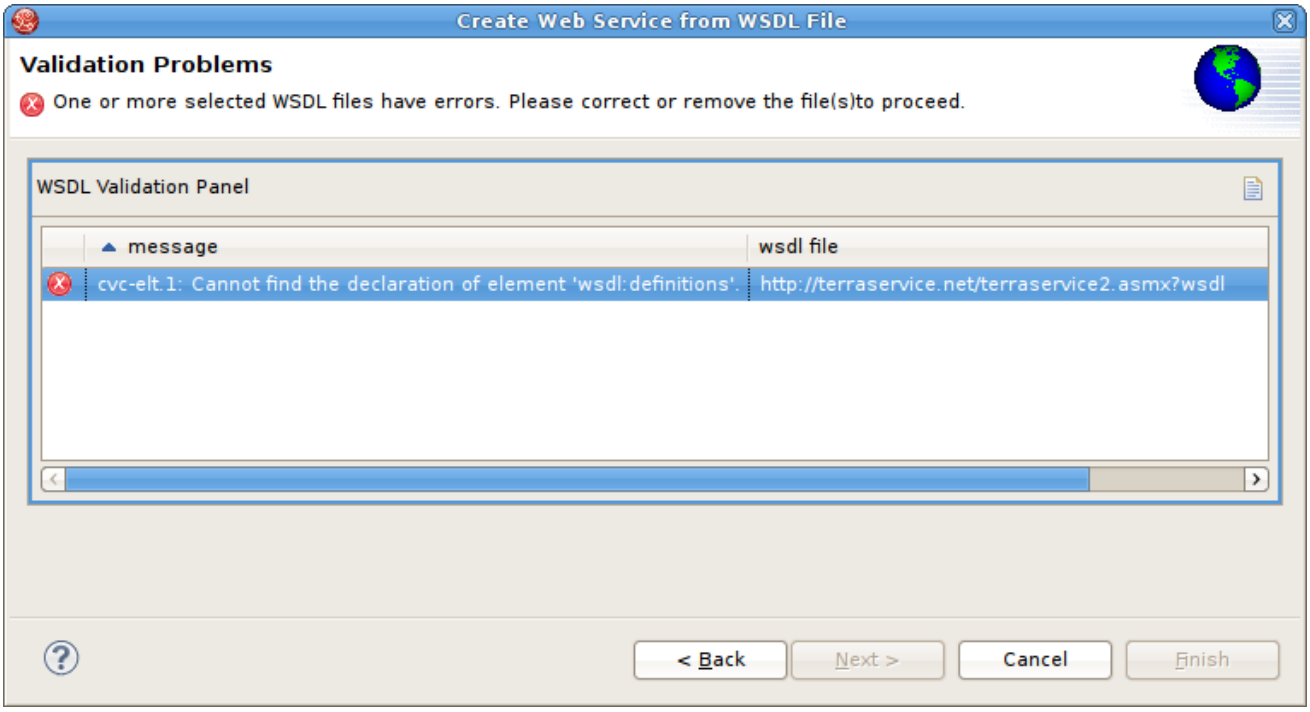


Figure 9.53. WSDL Validation Problems Dialog

9.10.4. Import WSDL from URL

You can create a Web Service model by selecting a WSDL file based on a URL.

1. Click the **File > Import...** action in the toolbar or select a project, folder or model in the Model Explorer's tree, right-click and choose **Import...**

2. Click the import option **Teiid Designer > WSDL File or URL >> Web Service Model** and click **Next>**.

3. Input a valid name for your Web Service model and click the **URL...** button.

   a. Enter a valid WSDL URL. If the URL cannot be validated then an error will be displayed after clicking the **OK>** button.

   b. If the WSDL is protected by basic HTTP authentication then this option should be selected and the appropriate username and password entered.

   c. Click **OK>** to continue.

   Click **Next>** to continue.

Figure 9.54. WSDL URL Dialog

NOTE

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

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, click **Next** to proceed, or **Finish** (if enabled) to complete with default options.

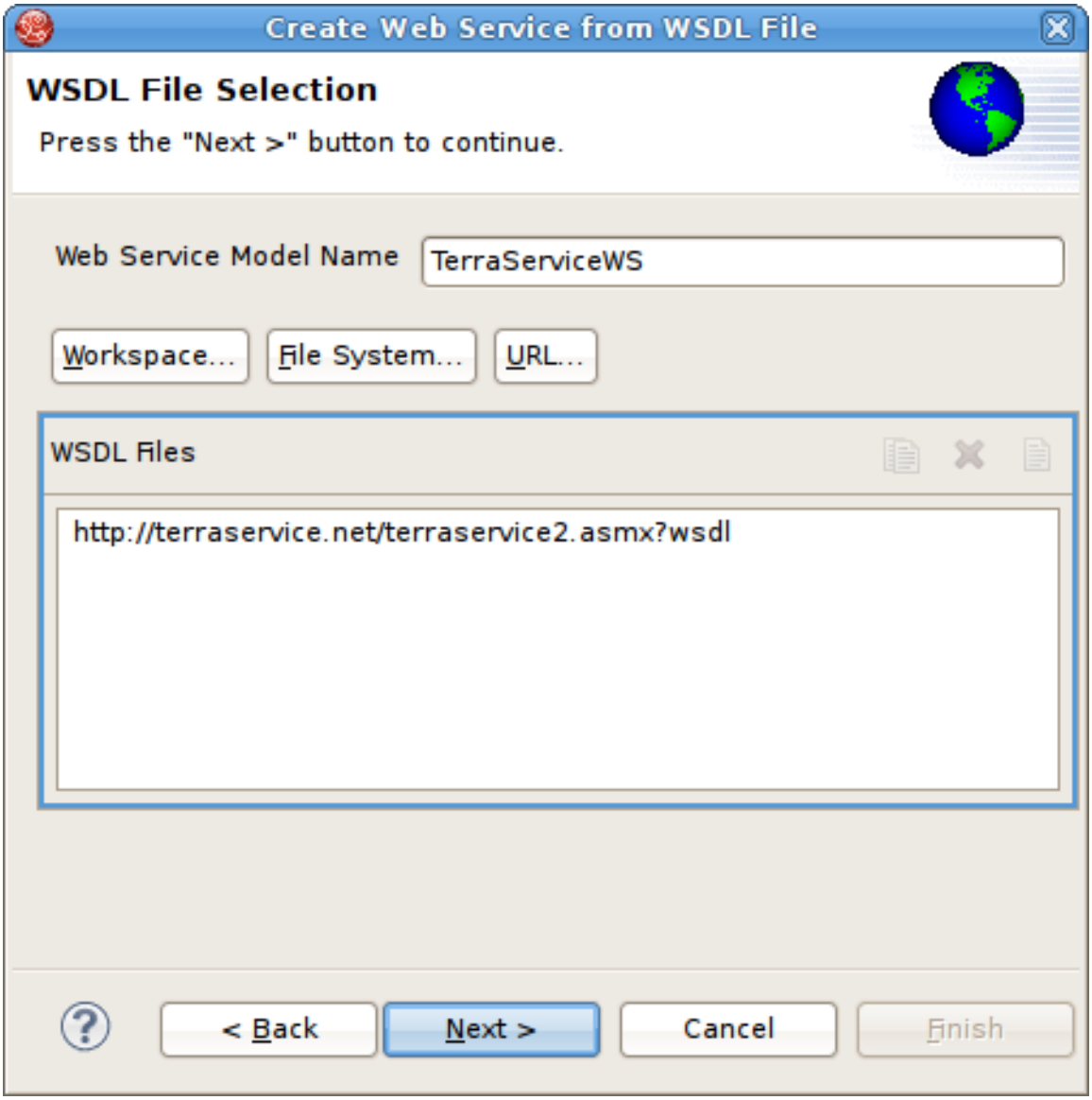


Figure 9.55. Namespace Resolution Dialog

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.

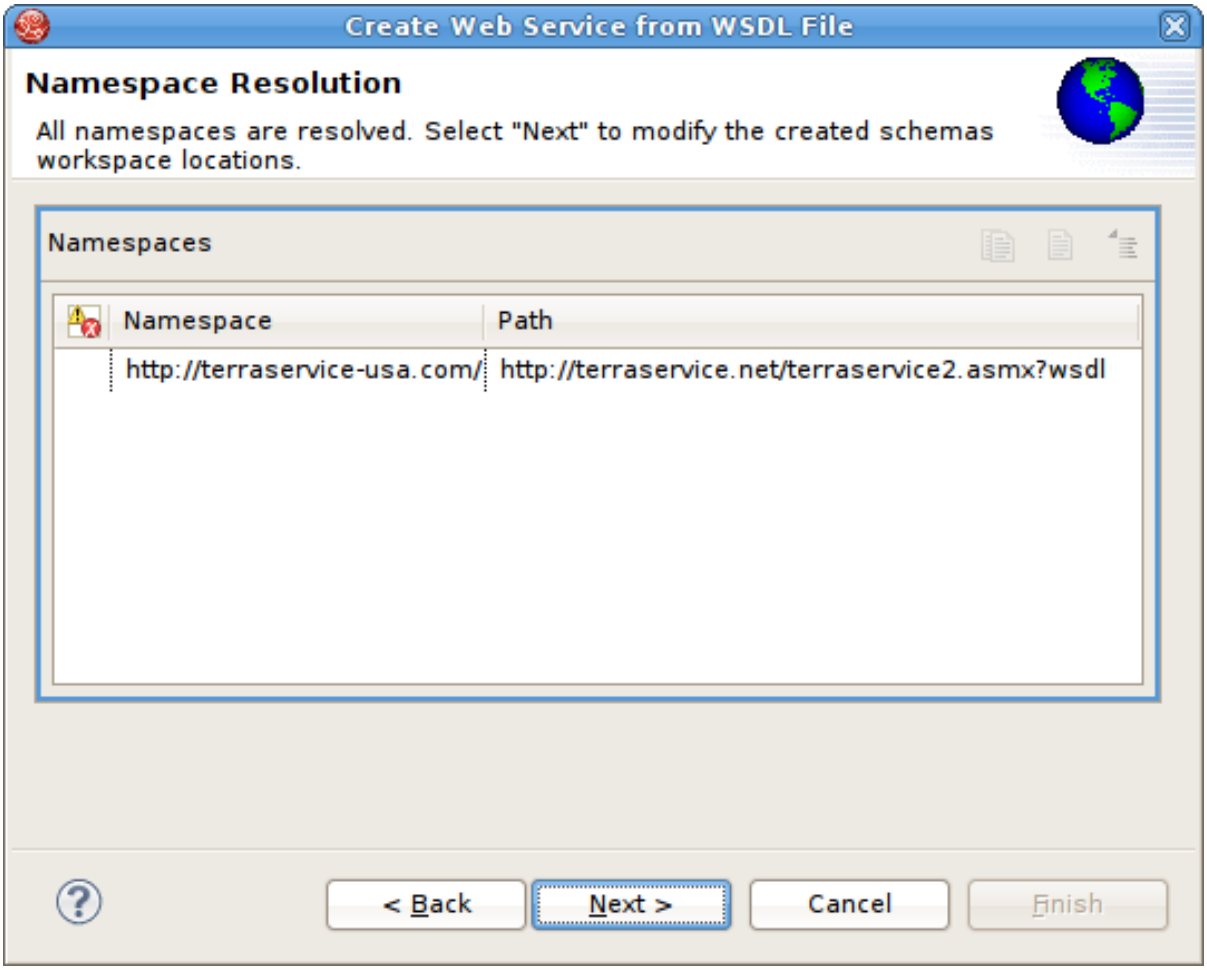


Figure 9.56. Namespace Resolution Dialog

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, click **`Next`** to proceed, or **`Finish`** to complete with default option.

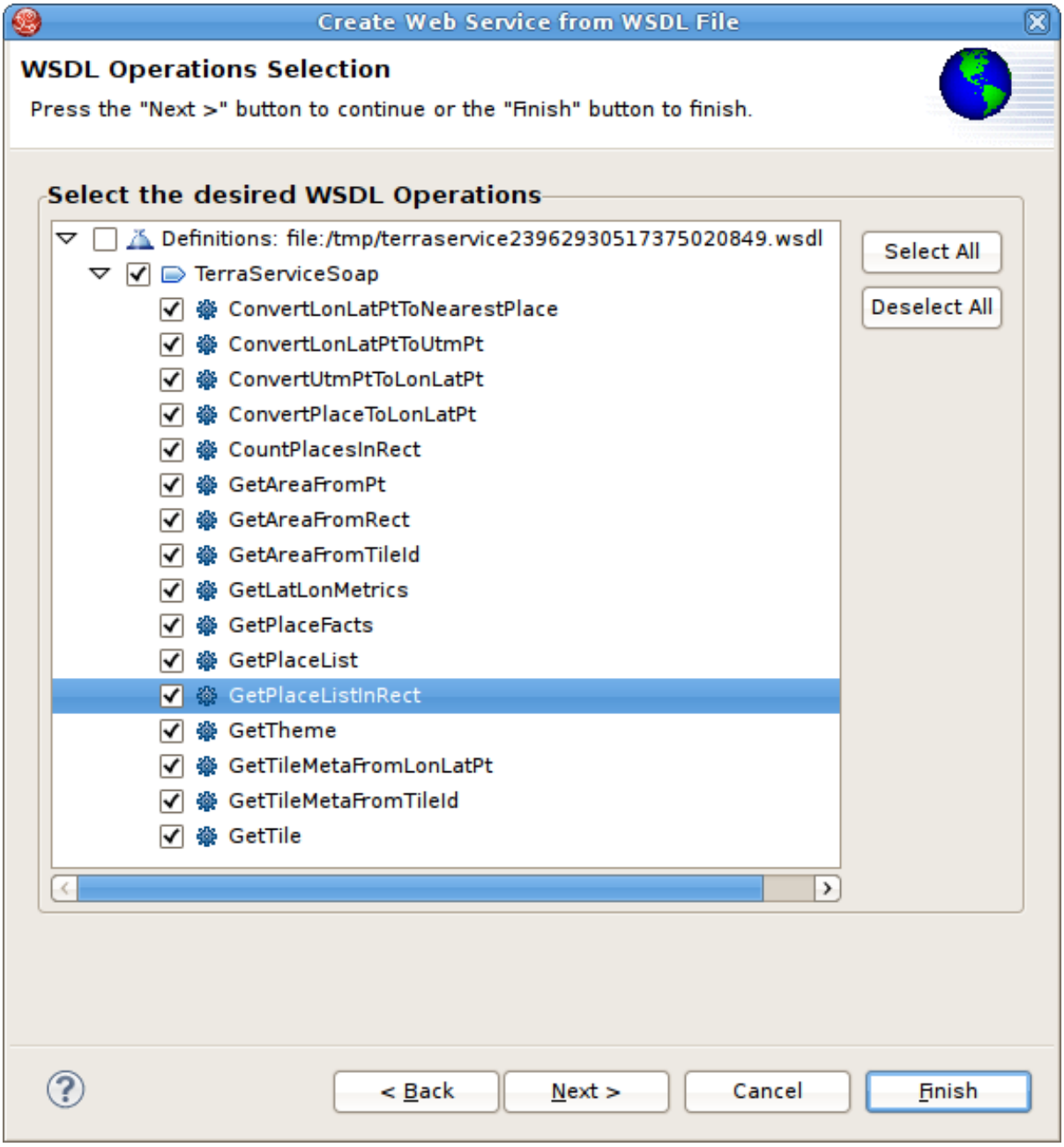


Figure 9.57. Namespace Resolution Dialog

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. Enter desired name or use the default name provide. Click **`Finish`** to complete.

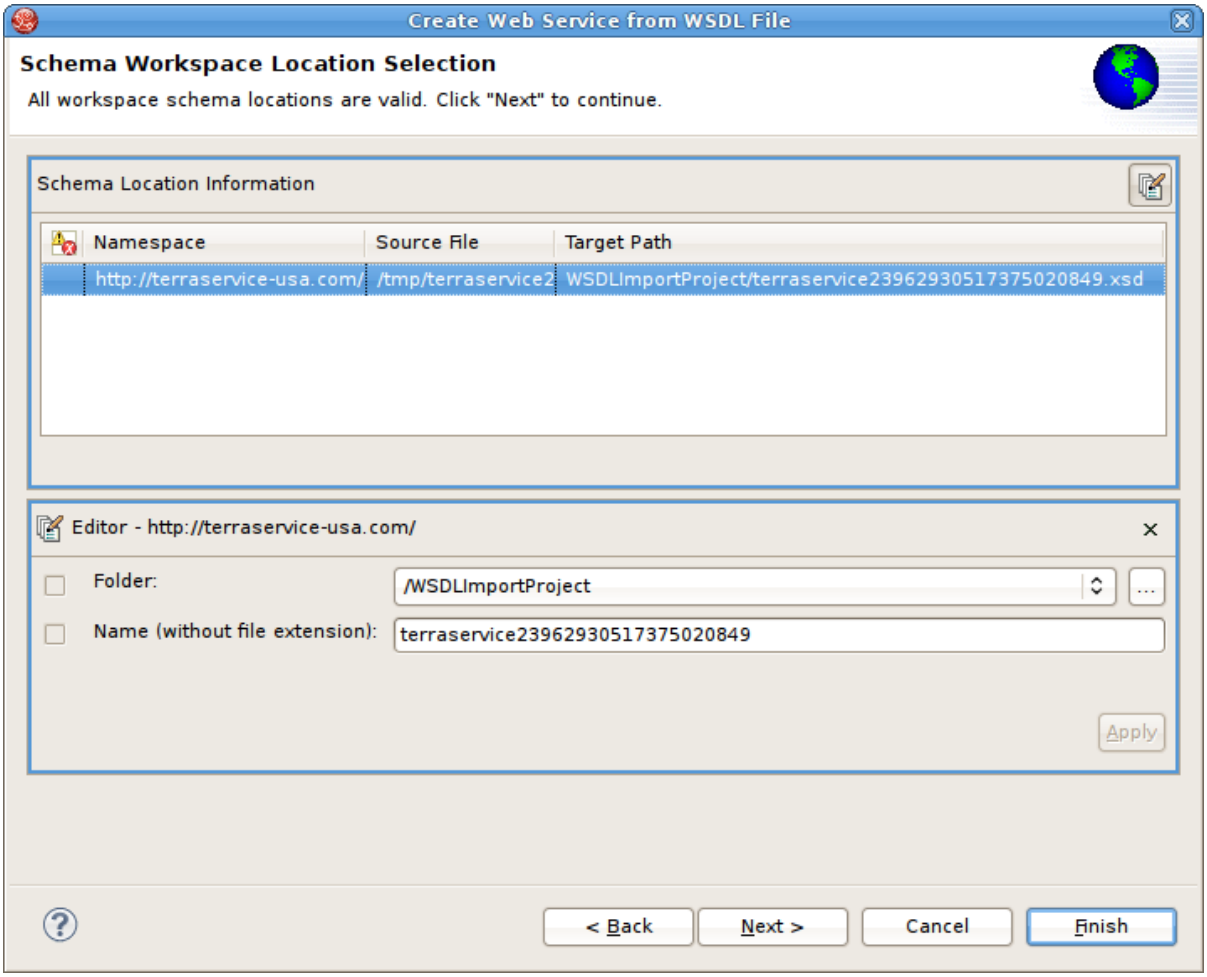


Figure 9.58. Namespace Resolution Dialog

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.

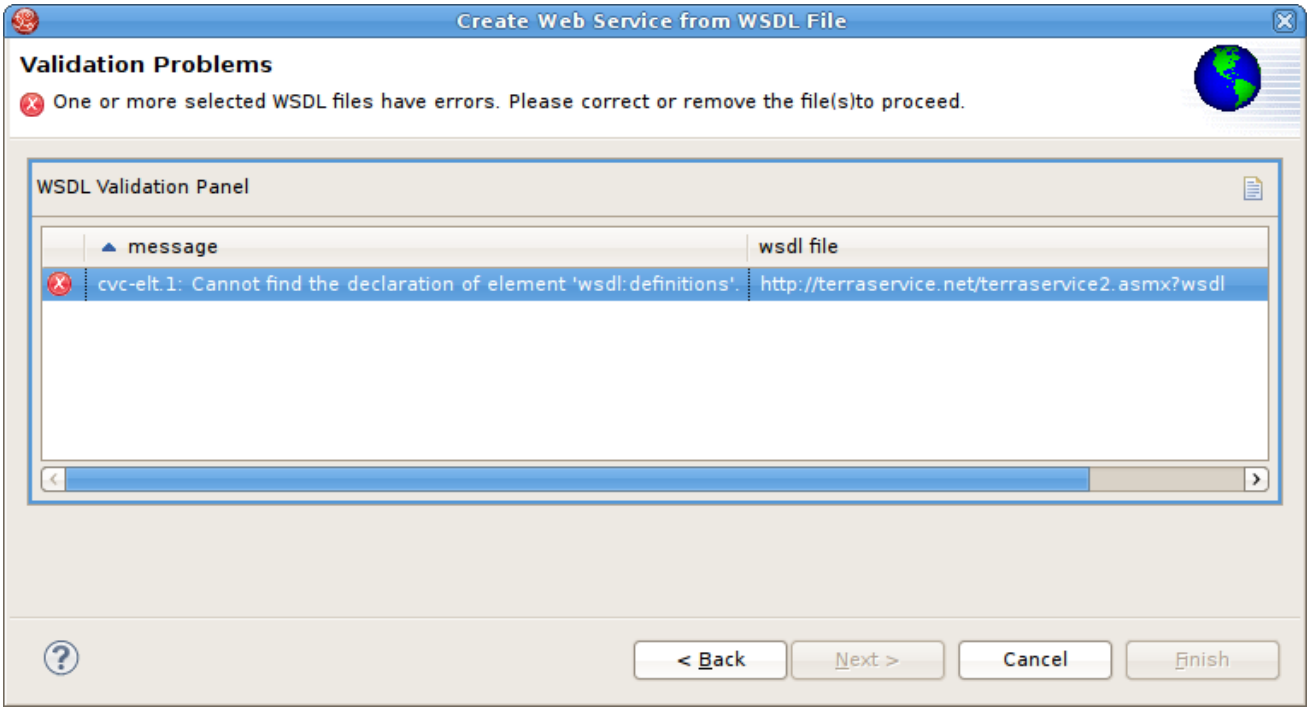


Figure 9.59. WSDL Validation Problems Dialog

9.11. IMPORT DATA FROM REST SERVICES

Procedure 9.2. Import Data from REST Services

1. Go into the **`Model Explorer`**.
2. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**
3. Select the import option Web Service Source >> Source and View Model (REST) and click Next.
4. Select an existing or previous connection profile from the drop-down selector.

   Alternatively, press the **`New...`** button to launch the **`New Connection Profile`** dialog or **`Edit...`** to modify or change an existing connection profile prior to selection.
5. If you are creating a new connection profile, choose the Web Services Data Source (REST)
6. Name your new REST Connection Profile and click **`Next`**.
7. Set your REST URL in the **`Connection URL`** text field and your Security Type and credentials, if applicable. Click **`Next`** to see a summary of your properties or click **`Finish`** to end.

   There is also an option to add Request Header parameters. The importer assumes an Accept header value of application/xml and a Content-Type header value of application/xml. These defaults can be overriden in the Optional Request Header Properties section. You can also add any other header properties required for the service. Click the **`Test Connection`** button to validate your connection properties.
8. After selecting or creating a new Connection Profile, the REST XML result from the connection profile will be displayed in the REST Web Service Response Data File panel. Check the data file you wish to process and click Next.

9. Enter a unique source model name in the Source Model Definition section.

   Alternatively, you can 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 invokeHttp() procedure. In this case, the source model nor the table will be generated.

10. Enter a unique view model name in the View Model Definition section.

    Alternatively, you can select an existing view 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.

11. Enter a new view procedure name in the View Model Definition section. Click Next.

12. Enter a JNDI Name in the Jboss Data Source Information section and click Next.

    NOTE

    The primary purpose of this importer is to help you create a view procedure containing the transformations required to query the user-defined data file. The panel contains an XML tree view of your result contents and actions/buttons you can use to create column entries displayed in the middle, Column Information panel. The root path is used for xpath parsing of the result document. The importer sets a root path for you. You can change the root path, if needed, by selecting an XML element and right-click select Set as root path action. Next, select columns in the tree that you wish to include on your query and select Add button. You can also modify or create custom columns, by using the ADD, DELETE,EDIT, UP, DOWN to manage the column info in your SQL. to generate your models and finish the wizard.

13. Click Finish to generate your models and finish the wizard.

9.12. LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP)

Lightweight Directory Access Protocol (LDAP) is a protocol for accessing distributed directory information over 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.

9.13. IMPORT FROM AN LDAP SERVER

In Teiid Designer, this is how you go about modeling an LDAP Server:

- LDAP subtrees are represented as if they were tables in a relational database.
- Each node in the sub-tree 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.

Create the relational model from LDAP data in this way:

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**
2. Select the import option Teiid Designer > LDAP Service >> Source Model and click Next.
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 or Edit... to modify/change an existing connection profile prior to selection.

   NOTE

   Information required for a new connection:

   - Hostname - eg. falcon (if your URL looks like ldap://falcon:389)
   - Port - eg. 389
   - Bind DN or user - an administrator account to browse the ldap tree, eg. cn=Manager,dc=birds-of-prey,dc=org
   - Bind password - account password

   Selection of the connection profile populates the LDAP Service URL. For LDAP Base DN you have to click on **`Fetch Base DNs`** button. 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.
4. After selecting a Connection Profile, enter JNDI name and click **`Next >`**.
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. Select an entry by ticking its respective checkboxes in the tree. When you highlight an entry, you will see 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.

6. Click Next.

7. 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. Select attributes by ticking their respective checkboxes in the tree. When you highlight an attribute, you will see 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.

8. Click Finish.

9. Once you have completed the wizard, the new source model will be created.

9.14. LDAP CONNECTOR UPDATE CAPABILITIES

The LDAP connector also provides an update capability. However, additional modeling requirements are imposed beyond those required for read-only access. Here is a list of these additional requirements:

- 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';

- You also need these additional 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.

Here are some sample 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'
```

9.15. XSD SCHEMA FILE

You can import XML Schema file (XSD) files using the steps below.

1. In Model Explorer, right-click and then click **`Import...`** or click the **`File`** > **`Import...`** action in the toolbar or select a project, folder or model in the tree and click **`Import...`**
2. Select the import option **`Teiid Designer > XML Schemas`** and click **`Next>`**.
3. Select either **`Import XSD Schemas from file system`** or **`Import XSD Schemas via URL`** and click **`Next >`**.
4. a. If importing from file system, the Import XSD Files dialog is displayed. Click the **`Browse`** button to find the directory that contains the XSD file(s) you wish to import.
   - To select all of the XSD files in the directory, click the checkbox next to the folder in the left panel.
   - To select individual XSD files, click the check boxes next to the files you want in the right panel.

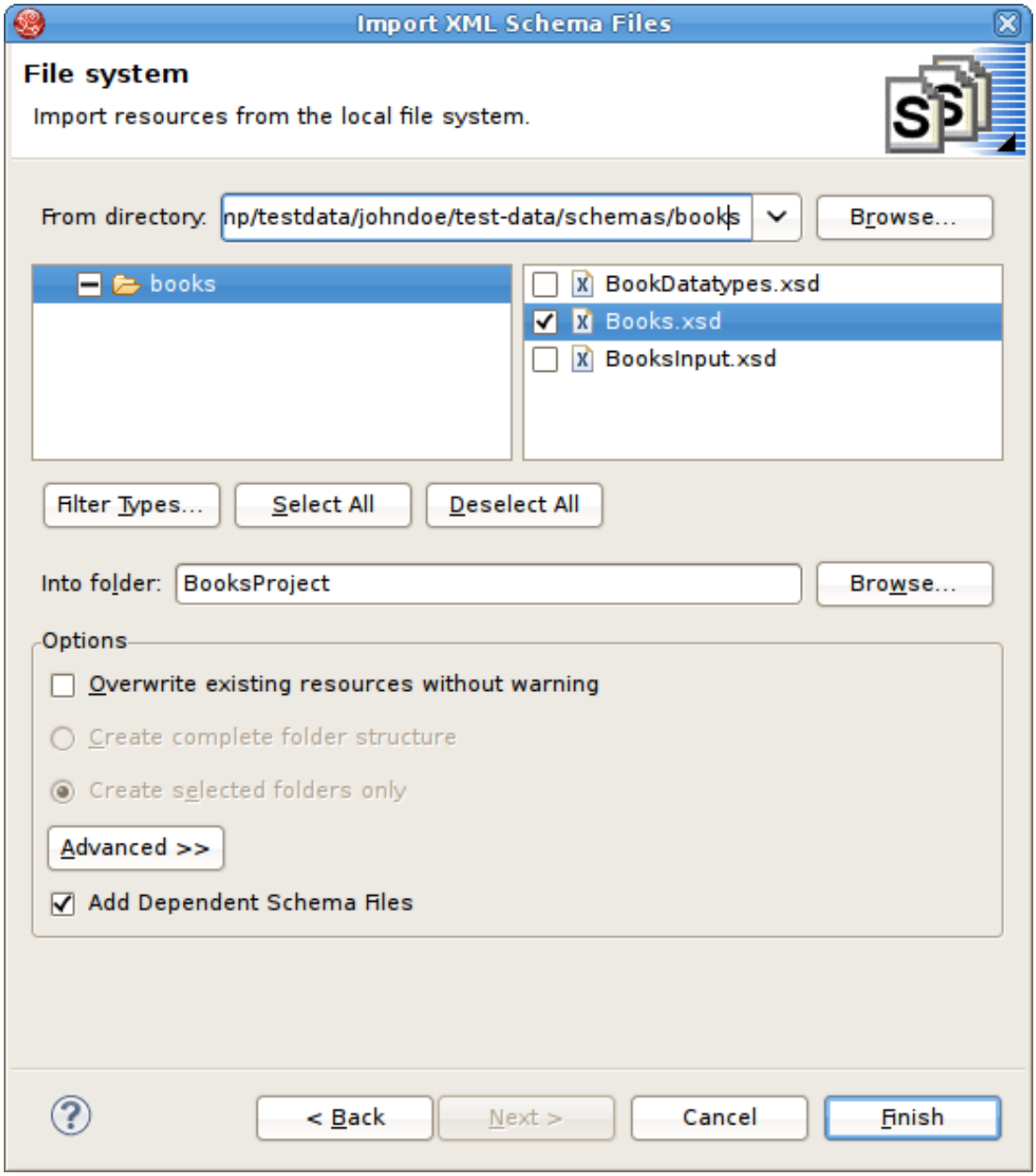


Figure 9.60. Select XSD From File System

b. 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.

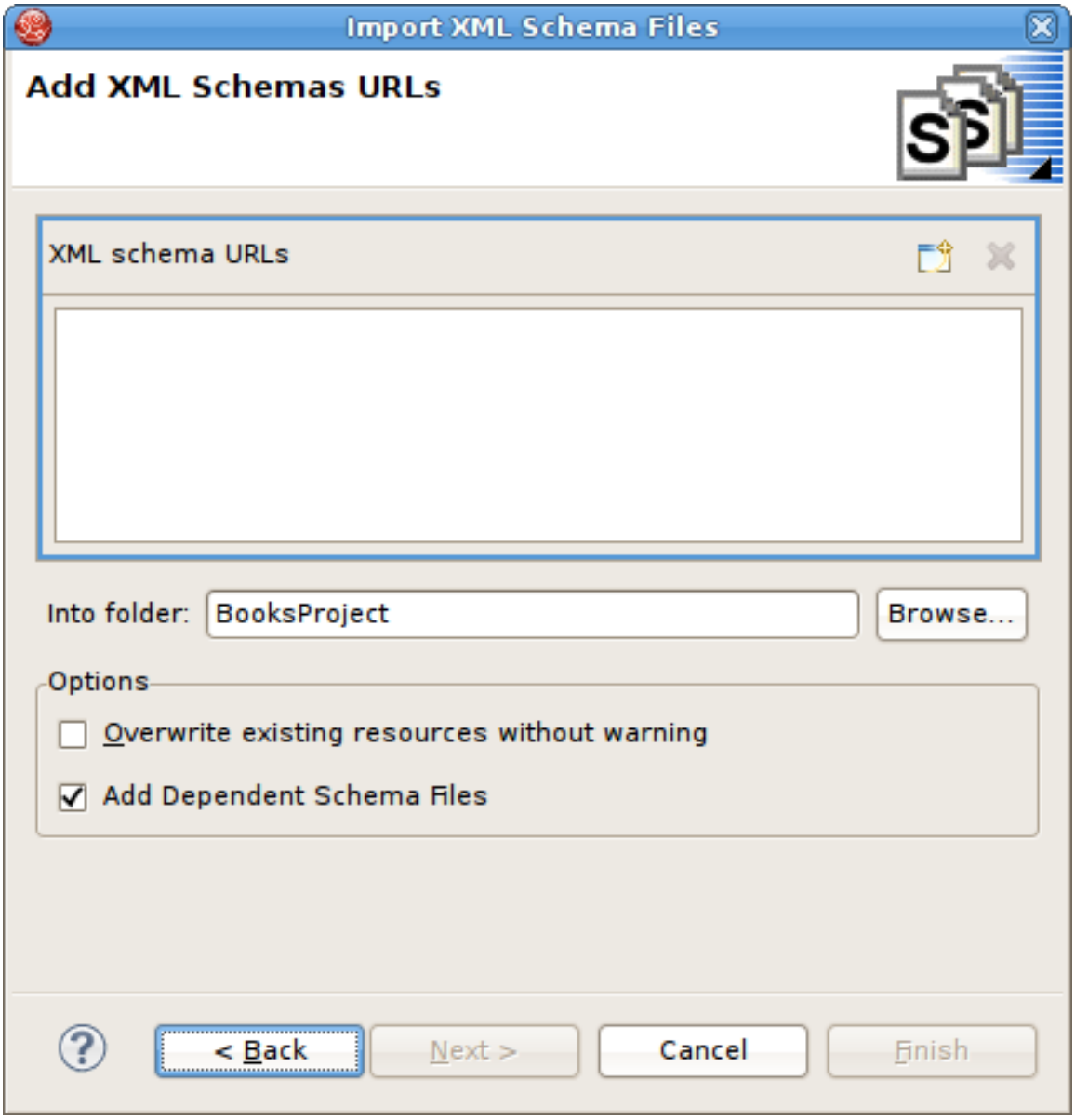


Figure 9.61. Add XML Schema URLs Dialog

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.

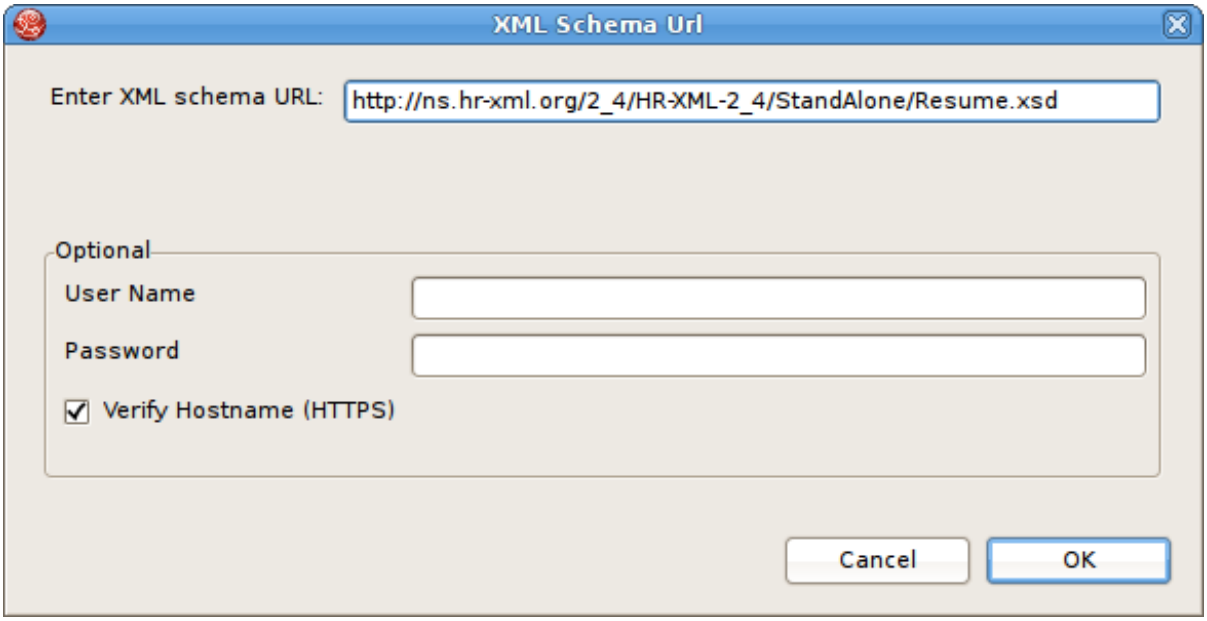


Figure 9.62. Add XSD Schema URLs

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

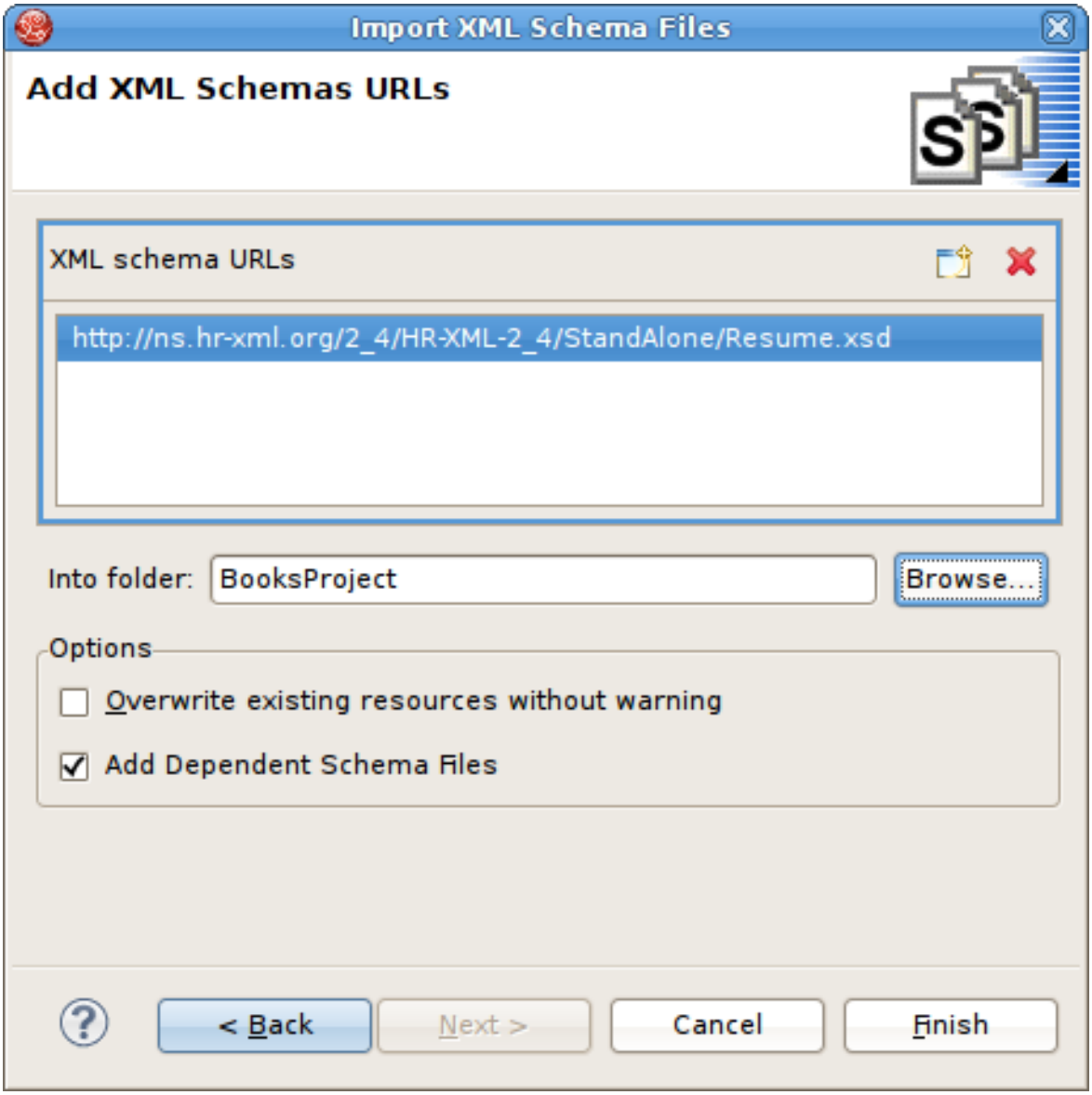


Figure 9.63. Add XSD Schema URLs

6. Click **`Finish`**.

NOTE

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

9.16. RED HAT JBOSS DATA GRID AND COMPLEX DATA TYPES

NOTE

When you use Teiid Designer to reverse-engineer the view into a pojo, a BigDecimal data type is defined in the view. Unfortunately for the Google Protobuf used for serialization, complex data types cannot be converted to either C or C++. It is therefore recommended that you use primitive data types only. (You will come across this situation if you are trying to materialize a view that contains a complex data type or if there is an existing JDG cache that contains a POJO that has complex data types.)

As the protobuffer does not support BigDecimal directly, you have three options:

1. use all primitive data types
2. implement a marshaller that will handle the conversion, which means the .proto file will also need to be created (see Red Hat JBoss Data Grid for the creation of files)
3. create a view that will convert the BigdDecimal to a string, then materialize that view.

CHAPTER 10. CREATING AND EDITING MODEL OBJECTS

10.1. CREATING NEW MODEL OBJECTS

10.1.1. Creating New Model Objects

This section summarizes **Teiid Designer** features for creating and editing existing model objects contained in your models. **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 Model Explorer view, Diagram Editor or Table Editor using the following actions:

- New Child Action
- New Sibling Action
- New Association Action

10.1.2. New Child Action

To create new child model objects in the Model explorer view:

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.
2. Right-click on a container object. From the pop-up menu, click **`New Child`**. You can now select the child object you would like to add.

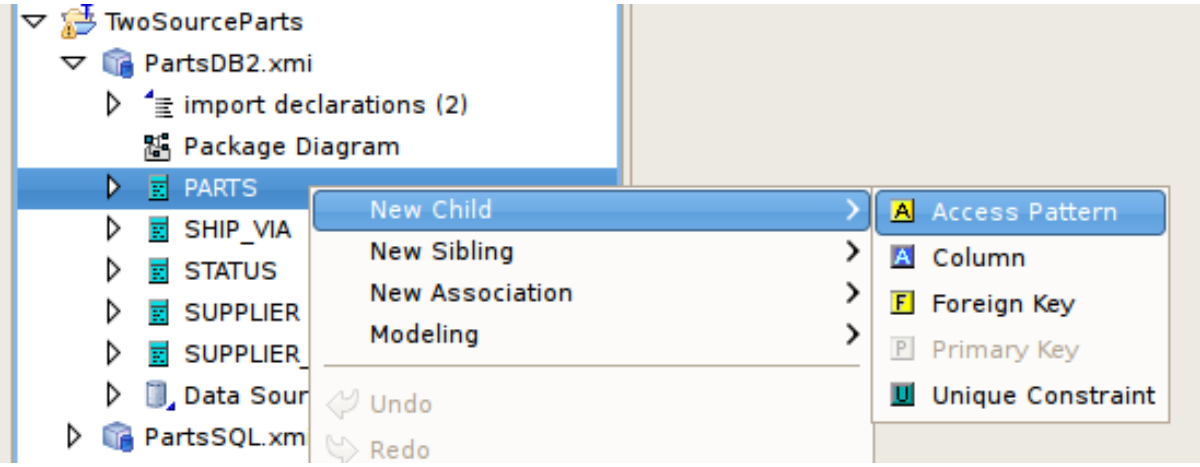


Figure 10.1. New Child Action In Model Explorer

3. The new model object displays on the Model Explorer view and is highlighted for renaming.

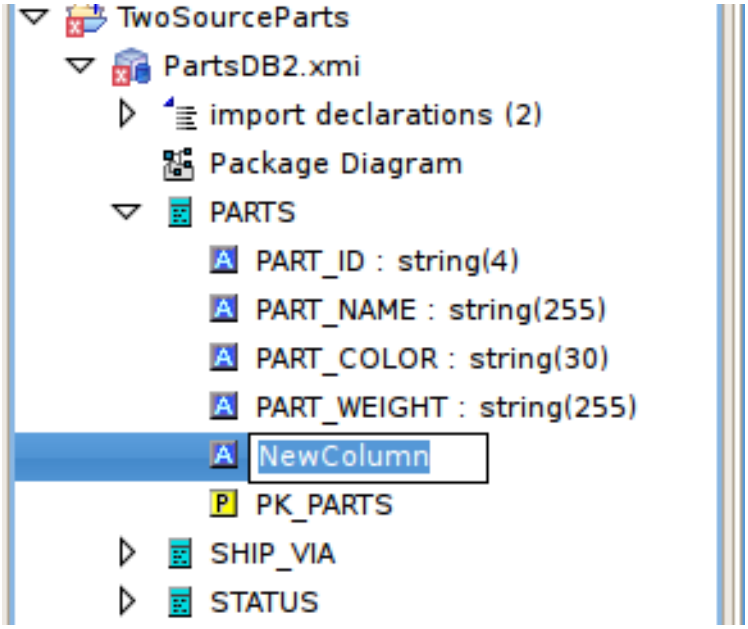


Figure 10.2. New Model Object In Explorer

To create new child model objects in the diagram editor:

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.

2. Right-click on a container object. From the pop-up menu, click **`New Child`**. You can now select the child object you would like to add.

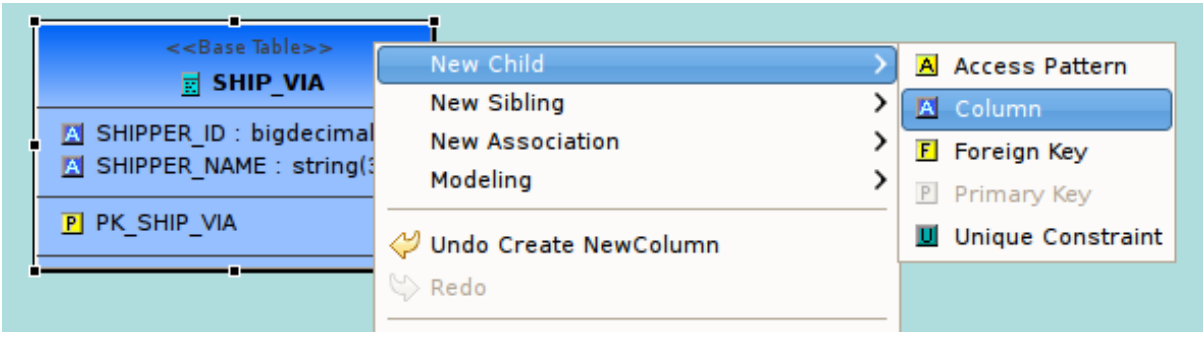


Figure 10.3. New Child Action In Diagram

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

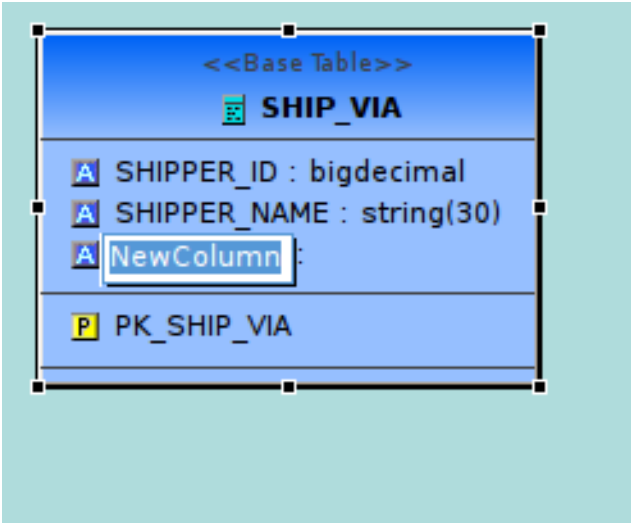


Figure 10.4. New Model Object In Diagram

To create new child model objects in the table editor:

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.

2. Right-click on a table row. From the pop-up menu, click **`New Child`**. You can now select the child object you would like to add.

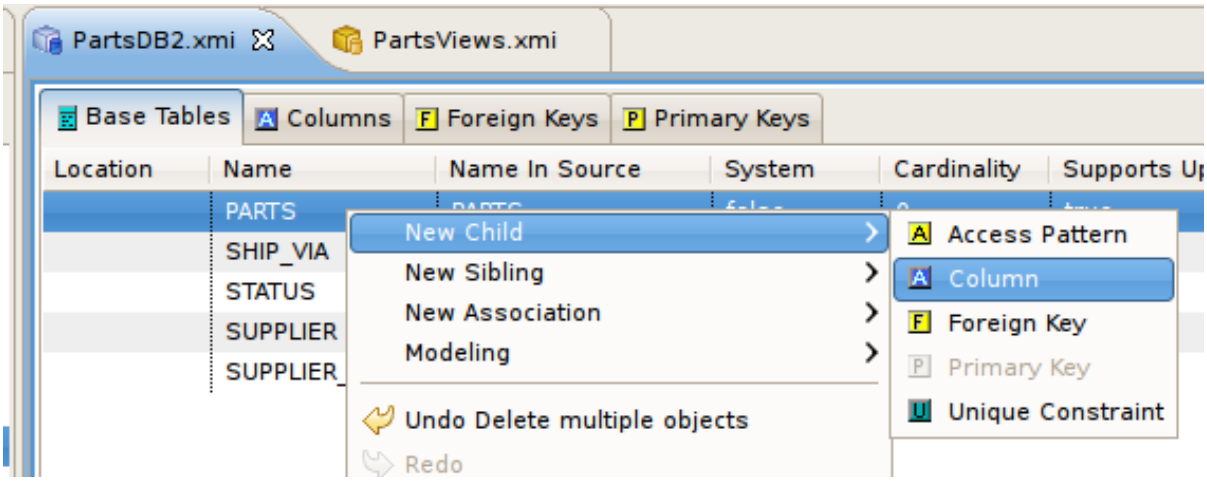


Figure 10.5. New Child Action In Table Editor

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.

10.1.3. New Sibling Action

To create new sibling model objects in the Model Explorer view:

1. Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.

2. Right-click on that object. From the pop-up menu, click **`New Sibling`**. You can now select the sibling object you would like to add.

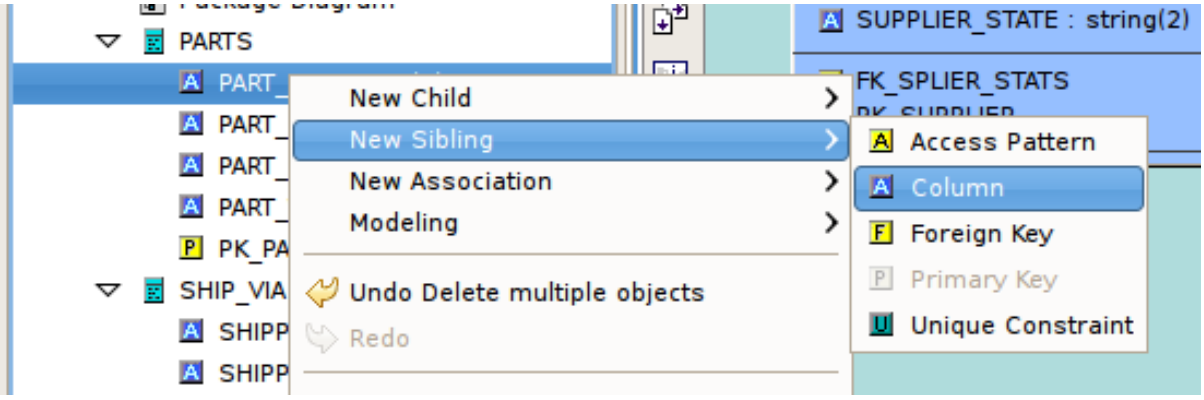


Figure 10.6. New Sibling Action In Model Explorer

3. The new model object displays in the Model Explorer view and is highlighted for renaming.

To create new sibling model objects in the diagram editor:

1. Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.

2. Right-click on that object. From the pop-up menu, click **`New Sibling`**. You can now select the sibling object you would like to add.

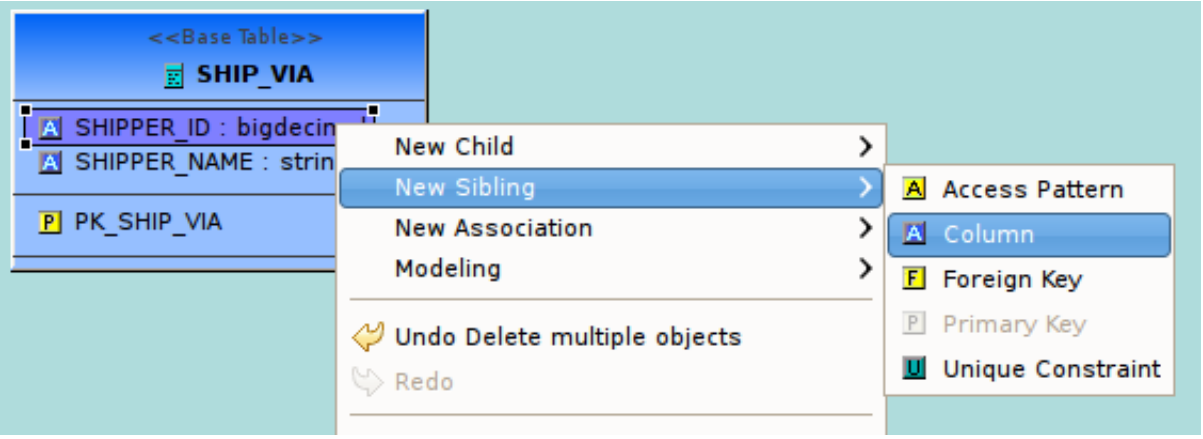


Figure 10.7. New Sibling Action In Diagram

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

To create new sibling model objects in the Table Editor section:

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.

2. Right-click on a row. From the pop-up menu, click **`New Sibling`**. You can now select the sibling object you would like to add.

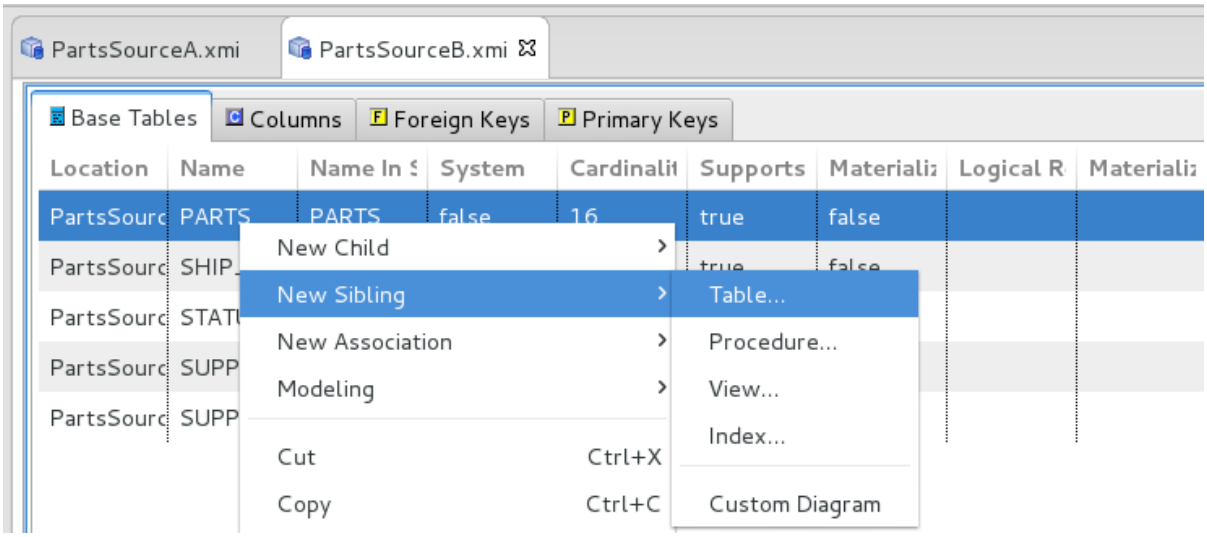


Figure 10.8. New Sibling Action In Table Editor

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.

10.1.4. New Association Action

To create new associations between model objects in the Model Explorer view:

1. Select two objects you wish to associate. For example, select columns in different base tables.

2. Right-click. From the pop-up menu, click **`New Association > Foreign Key Relationship`**.

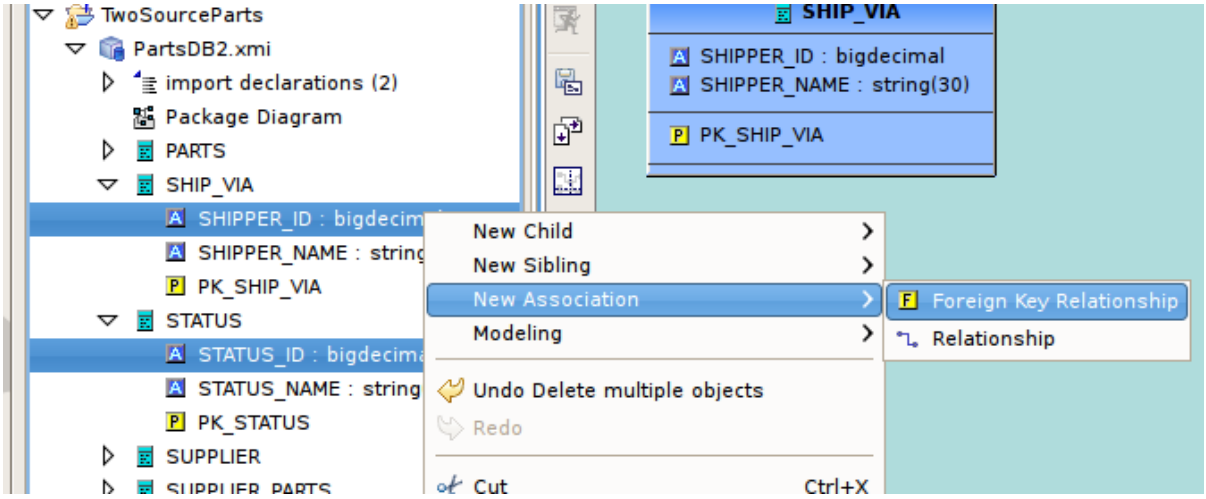


Figure 10.9. New Association Action In Model Explorer

3. The new relationship link is displayed in the diagram.

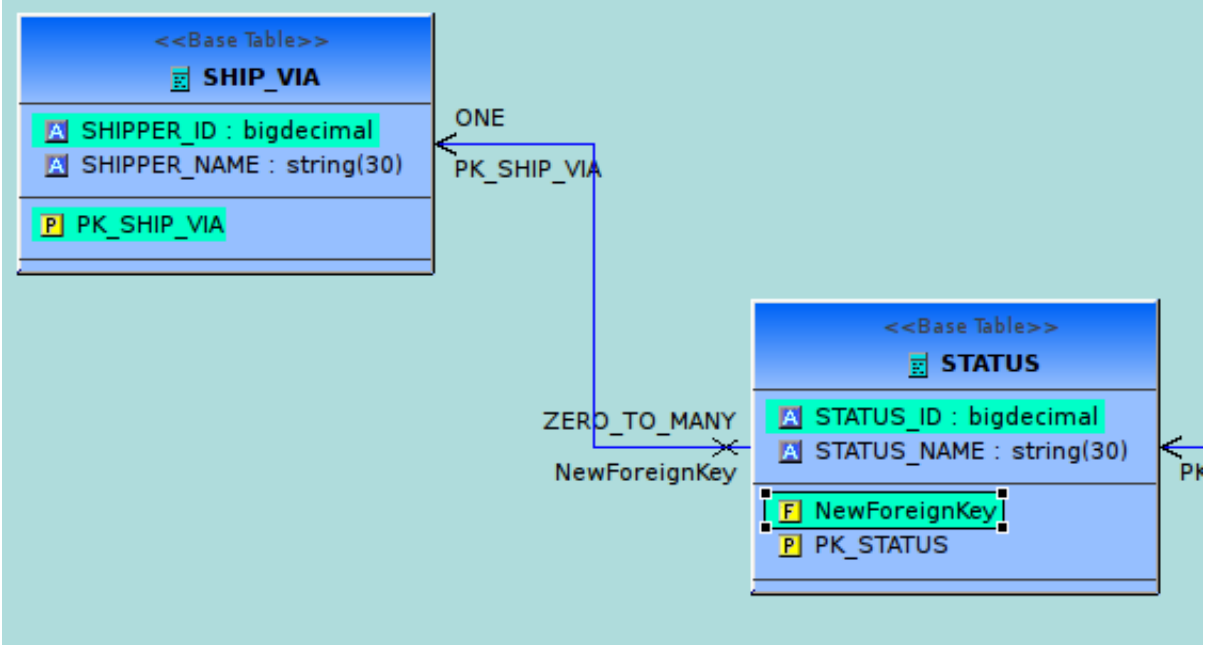


Figure 10.10. New Association In Diagram

To create new associations between model objects in the diagram editor:

- a. Select two objects you wish to associate. For example, select columns in different base tables.

  b. Right-click. From the pop-up menu, click **`New Association > Foreign Key Relationship`**.

  c. 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.

- a. Select a column in table.

  b. Drag the column to another table and drag over a column and drop onto this column. The target column should highlight in Yellow.

  c. 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 Table Editor:

1. Select two objects you wish to associate. For example, select columns in different base tables.
2. Right-click. From the pop-up menu, select **`New Association > Foreign Key Relationship`**.
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.

10.1.5. New Model Objects Wizard

Teiid Designer provides New Child and New Sibling wizards for children of view and source relational models,namely tables, views, procedures and indexes.

- The **`Create Relational Table Wizard`** allows you to create a complete relational table including columns, unique keys, foreign keys definition and indexes.

  NOTE

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

- The **`Create Relational Procedure Wizard`** allows you to create 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 and source function (pushdown function). When the **New Child** - **Procedure...** action is launched, the first dialog gives you the option of selecting the procedure type.

  NOTE

  You can model the user-defined functions in a view model.

  The second dialog in this wizard customizes the Create Relational Procedure dialog based on your selected type.

  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.

- **`Create Relational Index Wizard`**: Indexes can be created at the same time as your relational table object or via **New Child** - **Index...** action and dialog.

- **`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.

- **`Create User Defined Functions Wizard`**: You can define custom functions as view model procedures of the type "function". When you launch the **New Child** - **Procedure...** 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.

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

10.2. MODEL OBJECT EDITORS

The primary actions for editing model objects are:

- Cut - Deletes the selected object(s) and copies it to the clipboard.
- Copy - Copies the selected object(s) to the clipboard.
- Paste - Pastes the contents of the clipboard to the selected context.

- Clone - Duplicates the selected object in the same location with the same name; user is able to rename the new object right in the tree.
- Delete - Deletes the selected object(s).
- Rename - Allows a user to rename an object.

These actions are presented in Teiid Designer's **Edit menu** and also in the right-click context menus for model objects selected in the Model Explorer, Dialog Editor and Table Editor.

Modeling Sub-Menu

In addition to the New Child/Sibling/Association menus available for object creation Teiid 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:

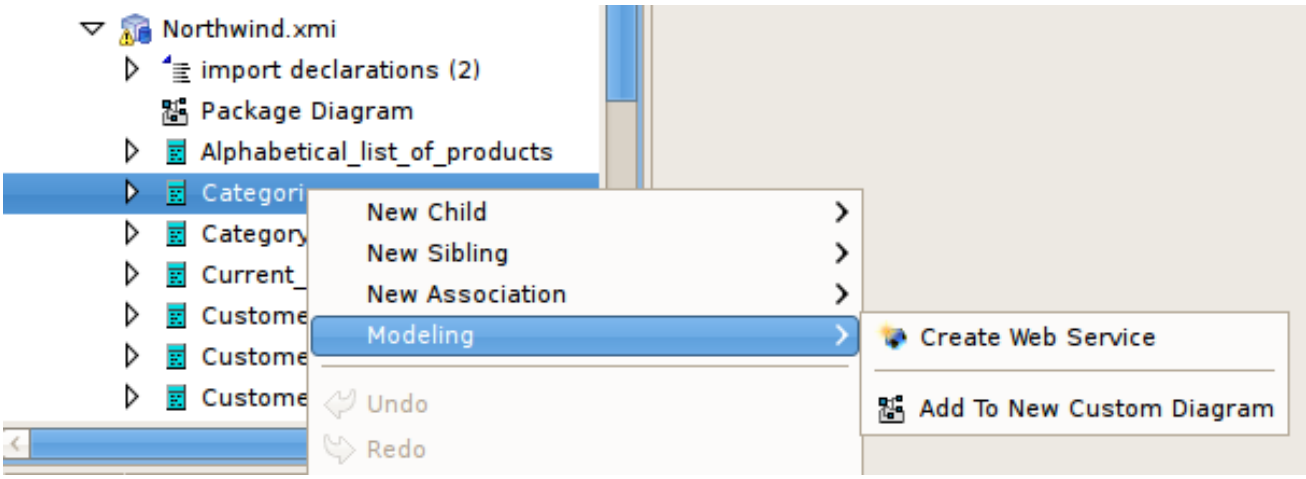


Figure 10.11. Modeling Sub-Menu for Source Table

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

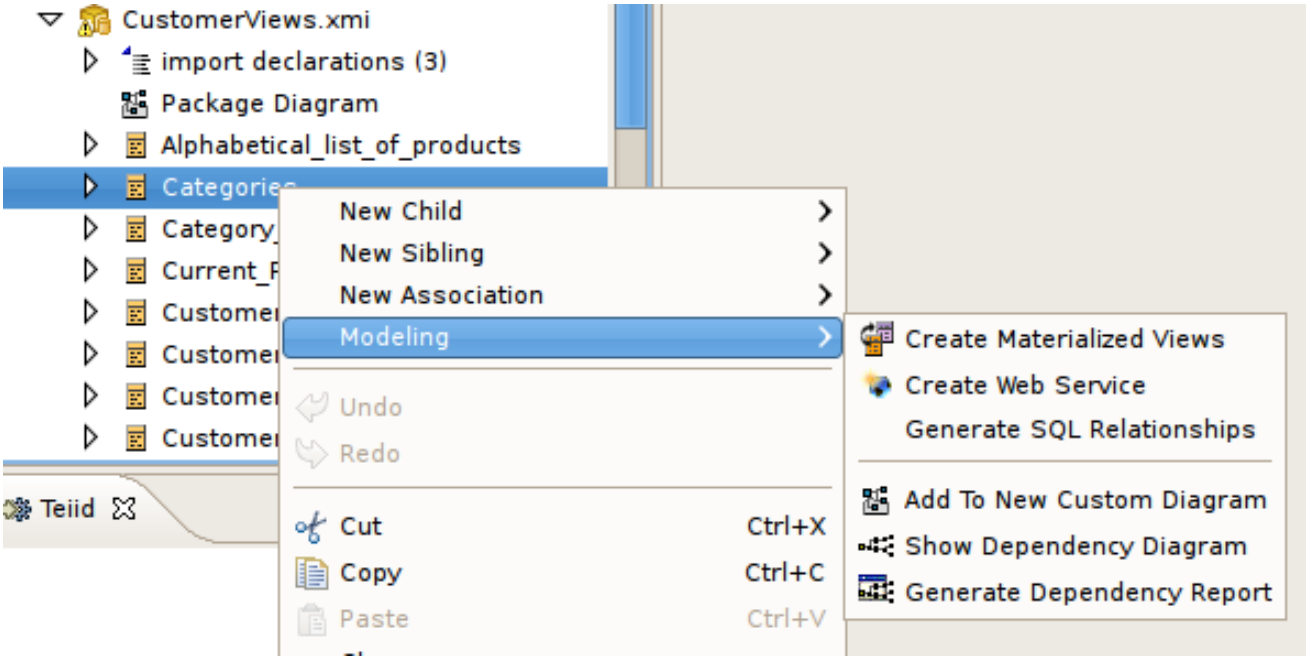


Figure 10.12. Modeling Sub-Menu for Source Table

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

10.3. TRANSFORMATION EDITOR

10.3.1. 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 that you can use to create SQL queries. You can use the tools, or you can type an SQL query into the **`Transformation Editor`**.

To edit a transformation:

- Double-click on:
  - A relational view table or procedure in the **`Model Explorer`** or **`Diagram Editor`**
  - A transformation node in a transformation 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

A **`Model Editor`** is opened if it is not currently open for the selected object's model.

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

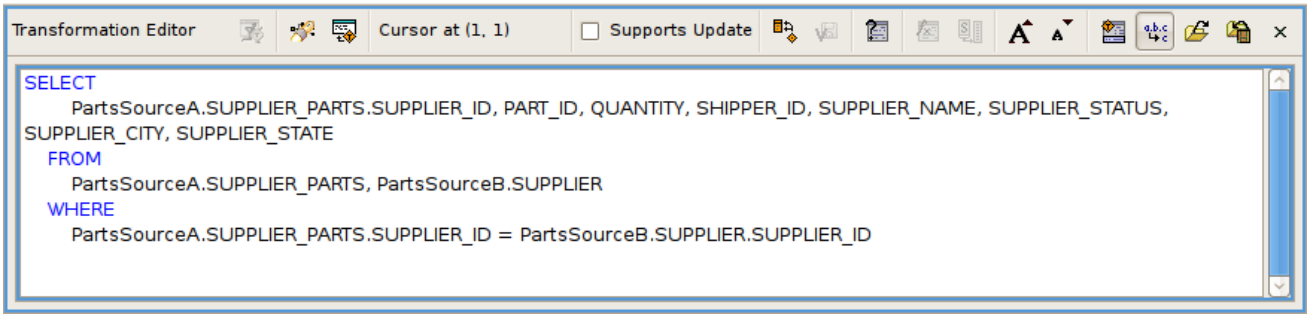


Figure 10.13. Editing String Property

If this virtual class supports updates, the tabs on the bottom of the **`Transformation Editor`** allow you to enter SQL for each type of query that 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`**.

- The **`Transformation Editor`** toolbar actions are summarized below.
  - Preview Virtual Data - executes a simple preview query for the target table or procedure of the transformation being edited.
  - Search Transformations - provides a simple way select and edit another transformation based SQL text search criteria.
  - Edit Transformation - provides a simple way to change which transformation to edit without searching in a diagram or the Model Explorer. Click the action and select from a list of views, tables, procedures or operations from the currently edited model.
  - Cursor at (1, 1) 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 Supports Update - checkbox allows you to enable or disable updates for the current transformation target. If Supports Update is selected, the editor shows four tabs at the bottom for the Select, Update, Insert and Delete transformations. If Supports Update is cleared, 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. Clicking this button will display the Reconcile Virtual Target Attributes dialog box in which you can resolve discrepancies.
  - Save/Validate - saves edits to the current transformation and validates the transformation SQL. Any Warning or Error messages will be displayed at the bottom of the editor in the messages area. If the SQL validates without error, the message area is not displayed.
  - Criteria Builder - allows you to build a criteria clause in your transformation. The button will enable if the cursor position is within a query that allows a criteria. Pressing the button will launch the Criteria Builder dialog. If the Criteria Builder is launched inside an existing criteria, that criteria will be displayed for edit, otherwise the Criteria Builder will be initially empty.
  - Expression Builder - allows you to build an expression within your transformation. The button will enable if the cursor position is at a location that allows an expression. Pressing

the button will launch the Expression Builder dialog. If the Expression Builder is launched inside an existing expression, that expression will be displayed for edit, otherwise the Expression Builder will be initially empty.

- Expand Select * - allows you to expand a SELECT * clause into a SELECT clause which contains all of the SELECT symbols. The button will enable only if the cursor is within a query that contains a SELECT * clause that can be expanded.
- Increase Font Size - increases the font size of all editor text by 1.
- Decrease Font Size - decreases the font size of all editor text by 1.
- Show/Hide Messages - toggles the display of the message area at the bottom of the transformation editor.
- Optimize SQL - when toggled ON, will use the short names of all SQL symbols that can be optimized. Some symbol names may remain fully qualified in the event of a duplicate name or if the optimizer is unable to optimize it. When the action is toggled OFF, all symbol names will be fully qualified.
- Import SQL Text - allows you to import an 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 clicking the right mouse button within the editor's text area. The context menu is show below:

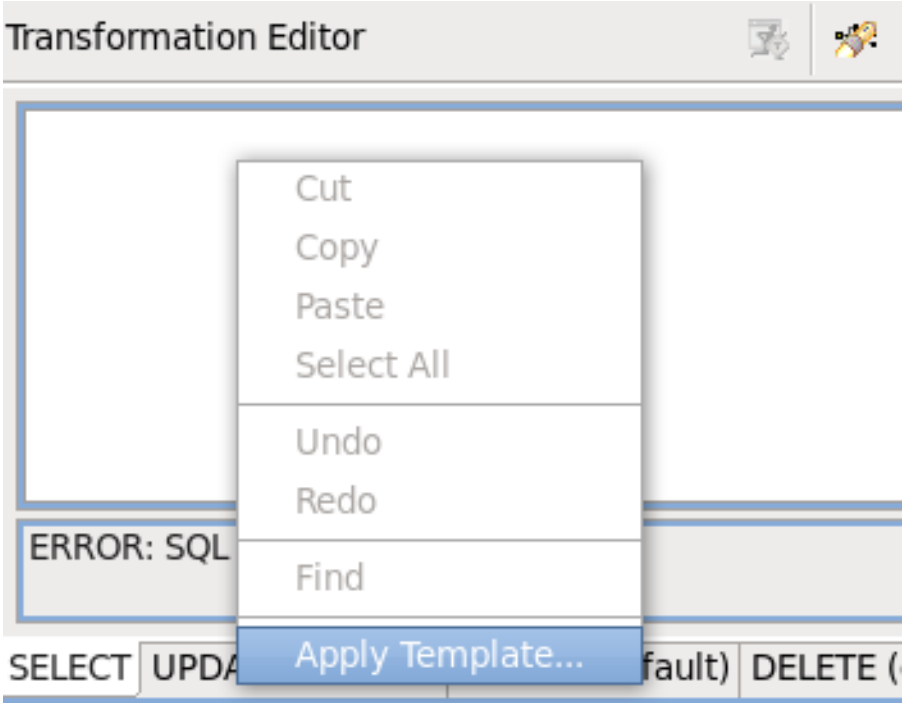


Figure 10.14. Transformation Editor context menu

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 and Replace Dialog which allows you to search and replace text within the transformation.
- Apply Template... - Displays the Choose an SQL Template Dialog, which allows you to choose a starting SQL Template from a list of common SQL patterns. See View Table wizard section for a description of this dialog.
- Create Function... - Opens **`Choose Function Type`** dialog where you can create **`Source Function`** or **`User Defined Function`**.

10.3.2. 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 the **`Reconcile Transformation`** button in the Transformation Editor.

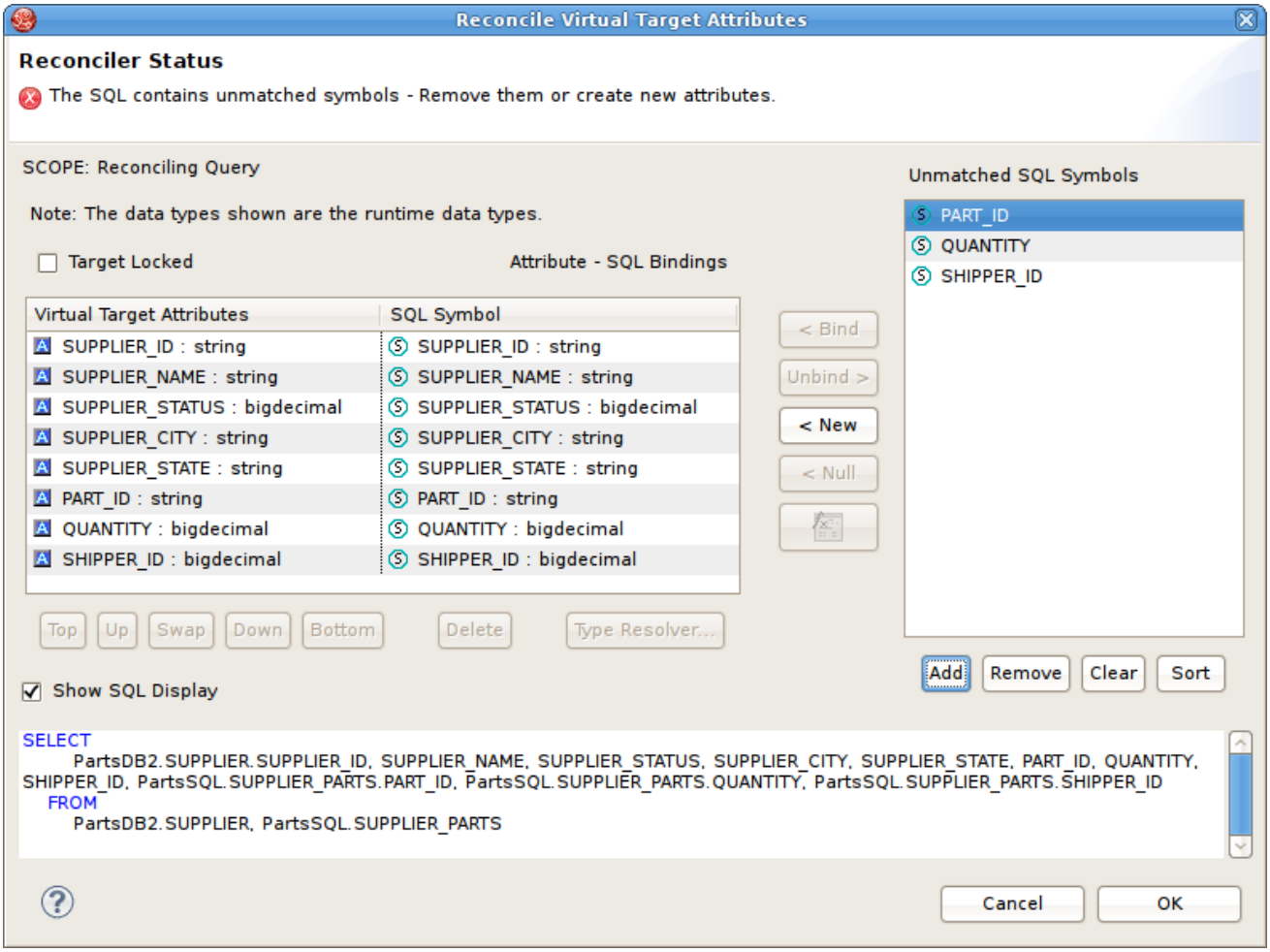


Figure 10.15. Reconciler Dialog

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, select 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 single or multiple table rows, then click the desired action button.
  - Delete Attributes: To delete one or more of the target attributes, select the table row(s) that you want to delete and 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 Using Datatype Resolver section 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 that 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 an 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 an 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 an SQL Symbol, the symbol will be released to the Unmatched Symbol list.

  - Function: This button allows you to define an expression instead of an 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 Using Expression Builder section 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 click OK, this SQL will be your new transformation SQL. If desired, the SQL Display can be hidden by clearing 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.

10.3.3. Using the Datatype Resolver

This dialog is accessible from the **`Reconciler`** dialog (See Using Reconciler section) and offers you a quick way to resolve datatype conflicts between a target attribute and its SQL Symbol. You can resolve the conflicts in the datatype bindings either by converting all source SQL symbol datatypes or by changing all target column datatypes.

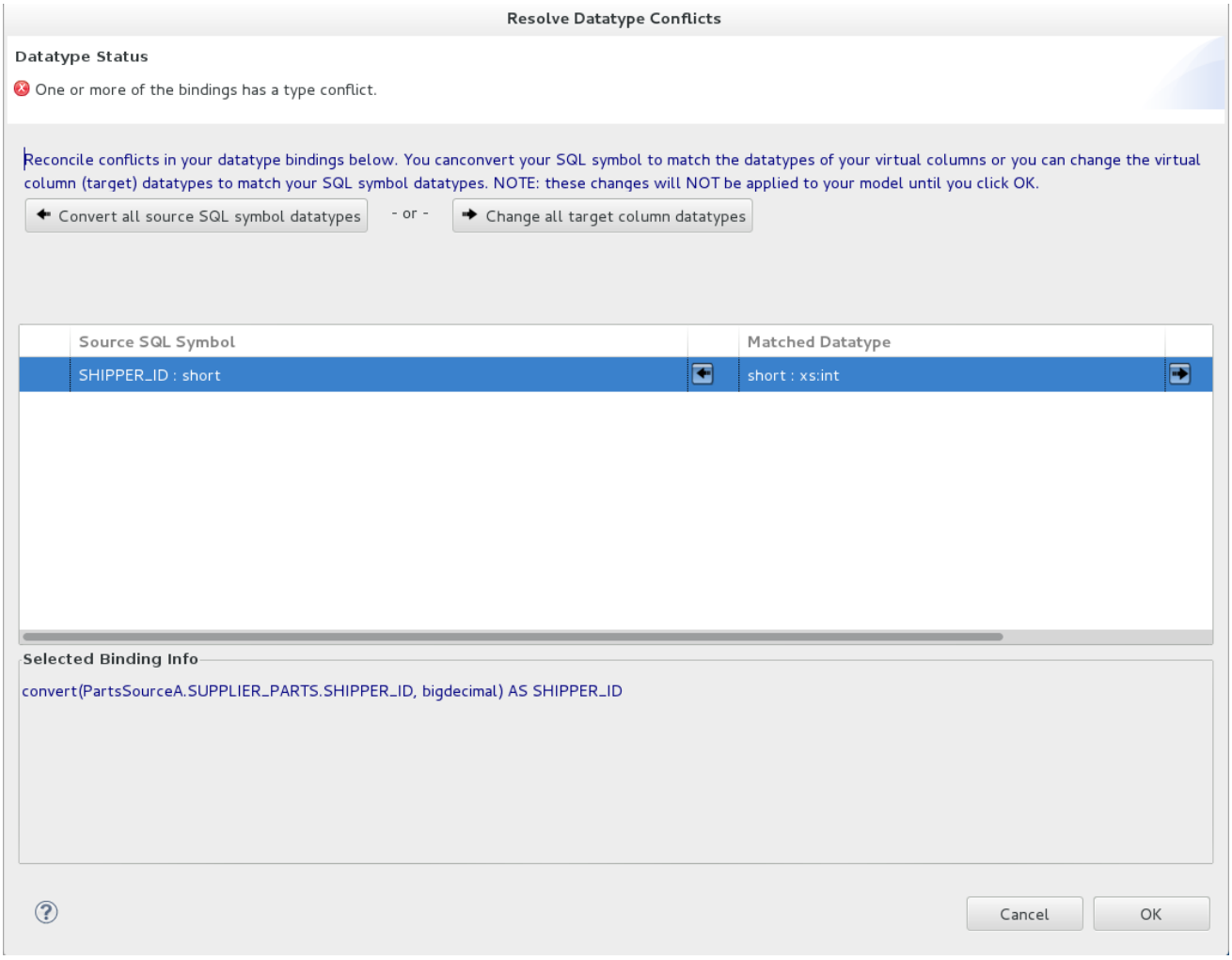


Figure 10.16. Datatype Resolver Dialog

To summarize the different sections of the dialog:

- Convert all source SQL symbol datatypes: Click this button to apply a CONVERT function to all of SQL symbols in the table so that their datatype is compatible with the corresponding attribute datatype.
- Change all target column datatypes: If the suggested datatype is not acceptable, click this button to choose your own datatype from the datatype dialog.
- Source SQL Symbol - Matched Datatype Table: This table shows all SQL Symbol datatype information for the selected binding. Select on a table row to populate the lower panel.
- Selected Binding Info: The lower panel shows the binding information for the selected SQL symbol. hows all SQL Symbol datatype information for the selected binding. Select on a table row to populate the lower panel.

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.

10.3.4. 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:

1. In the **`Transformation Editor`**, click the **`Launch Criteria Builder`** button.
2. The **`Criteria Builder`** displays.

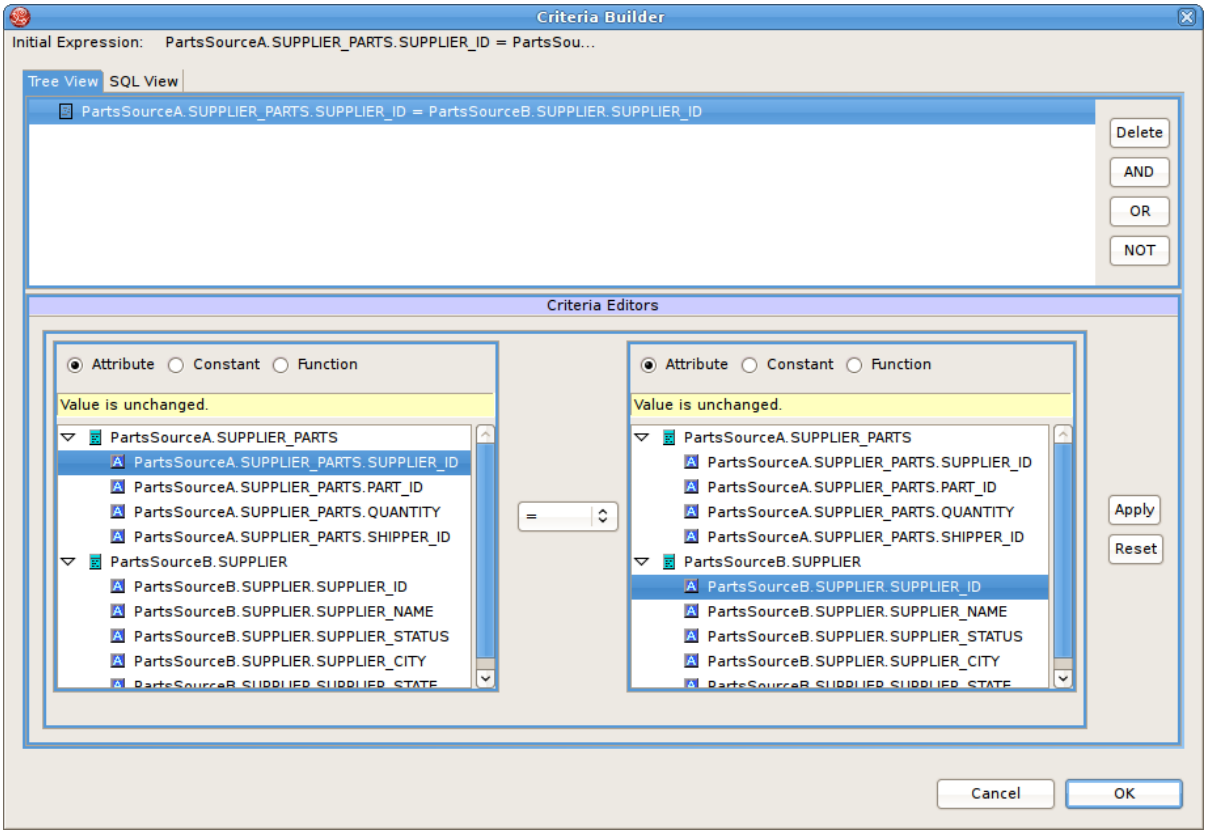


Figure 10.17. Editing String Property

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.

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:

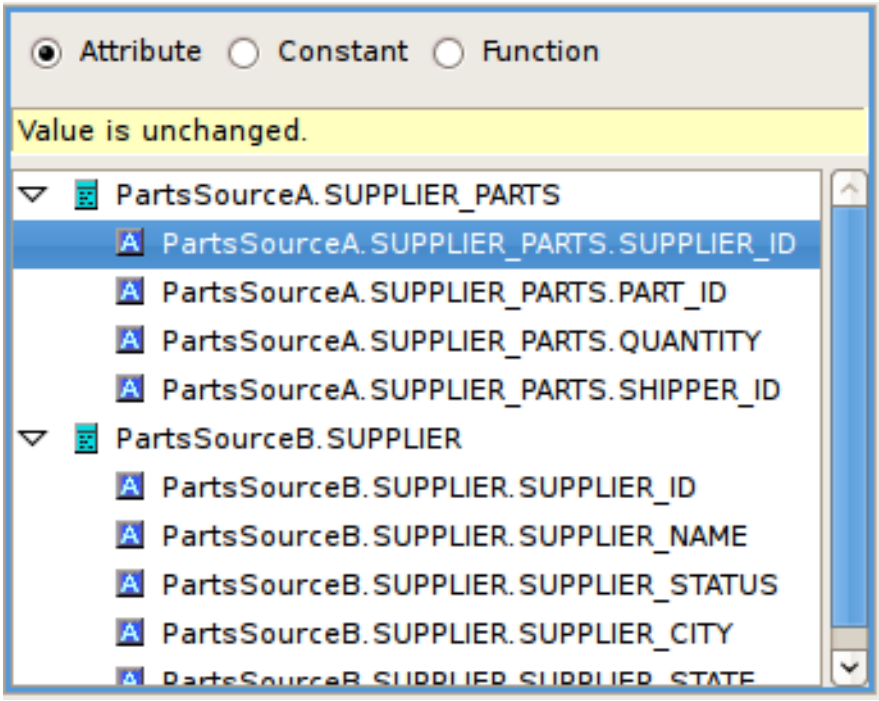


Figure 10.18. Attribute Panel

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 hardwired constant value to the predicate. If you click this radio button, the Predicate Editor looks like this:

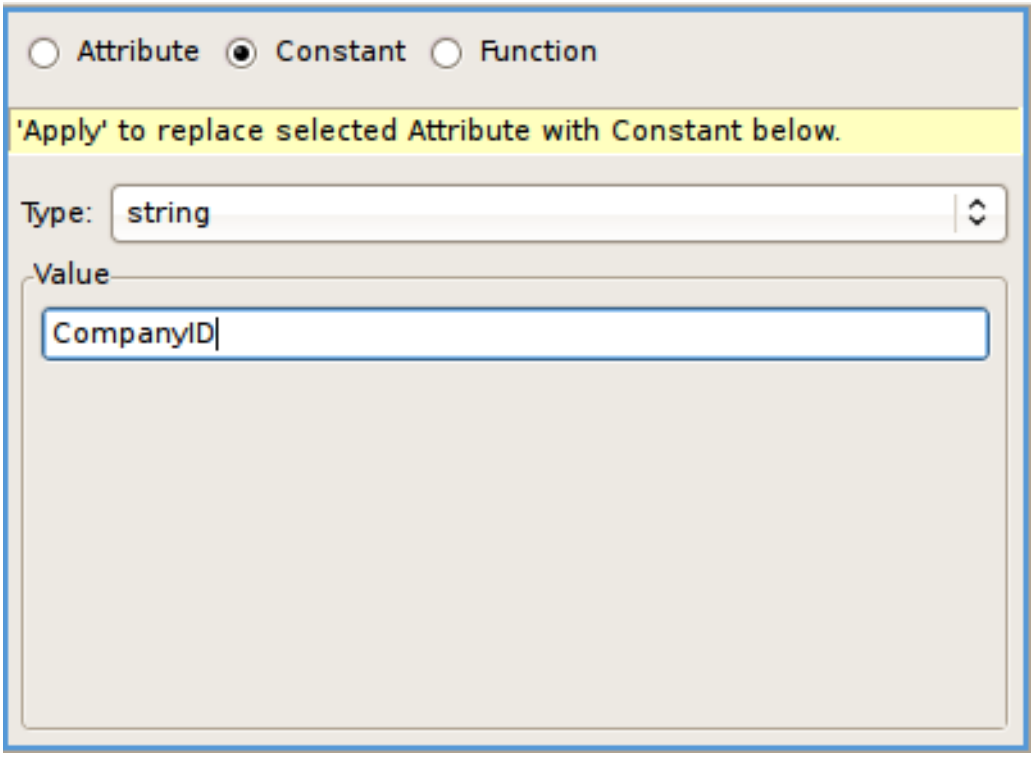


Figure 10.19. Constants Panel

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.

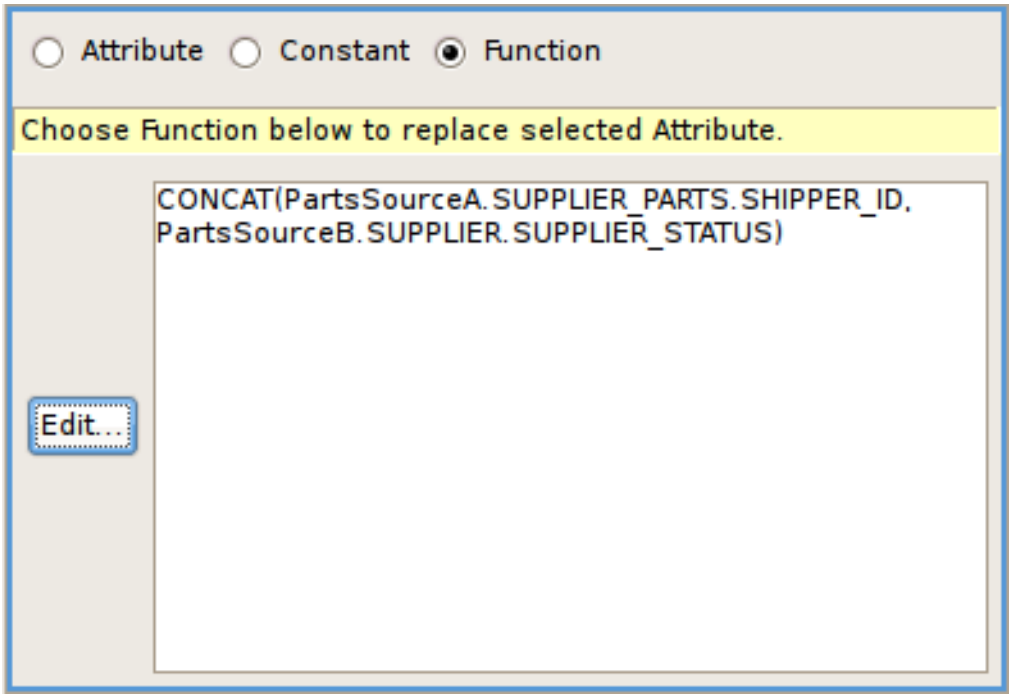


Figure 10.20. Functions

Click the **Edit** button to use the Expression Builder to construct a function to use in the predicate of your SQL Criterion.

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.

5. Click **Apply**.

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.

10.3.5. 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.

NOTE

The functions made available through the expression builder are described in the Teiid Reference Guide.

To use the Expression Builder:

1. In the **`Transformation Editor`**, click the location where you want to insert the function.
2. Click the **`Expression Builder`** button. The **`SQL Expression Builder`** displays.

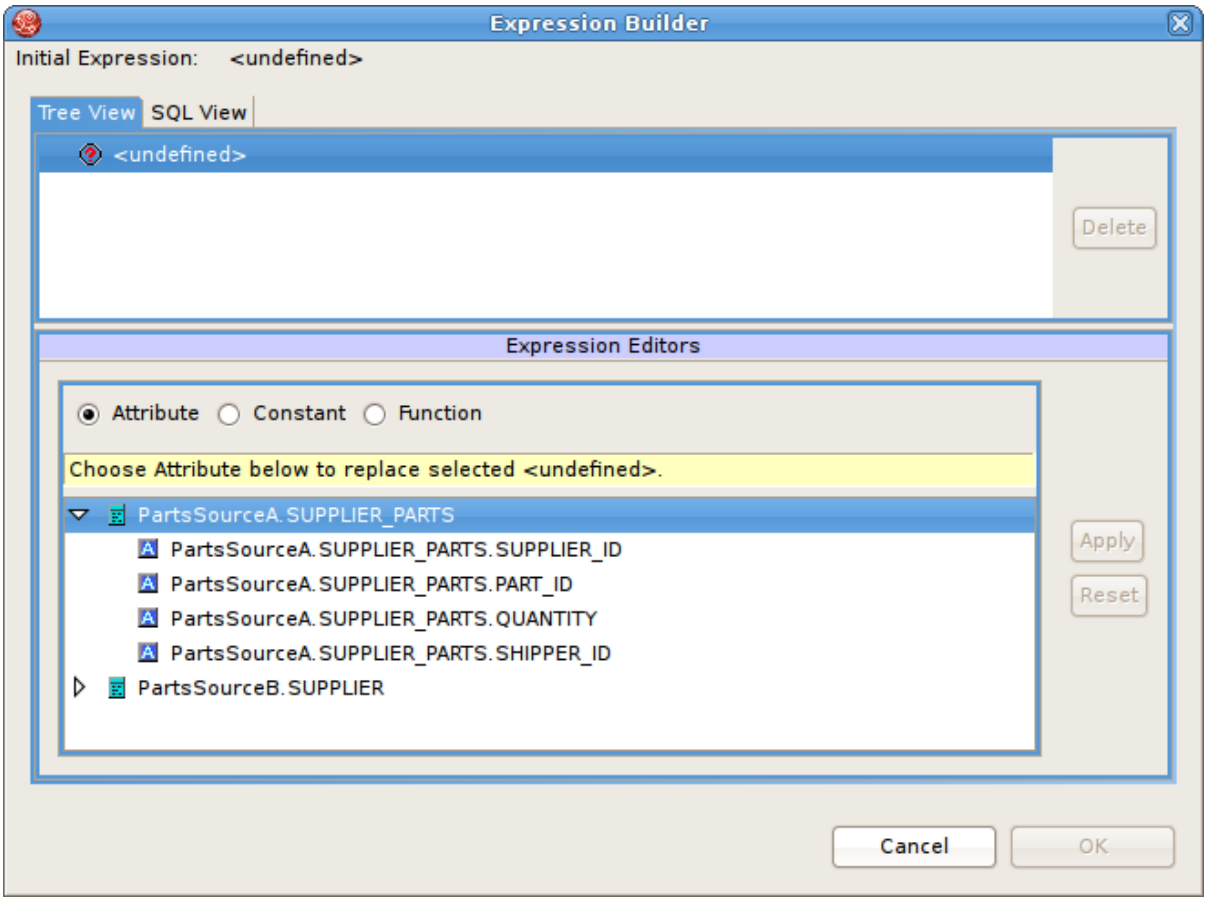
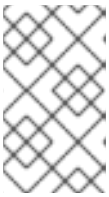


Figure 10.21. Expression Builder

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.

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.

4. The **`Expression Editor`** displays the Function editor.

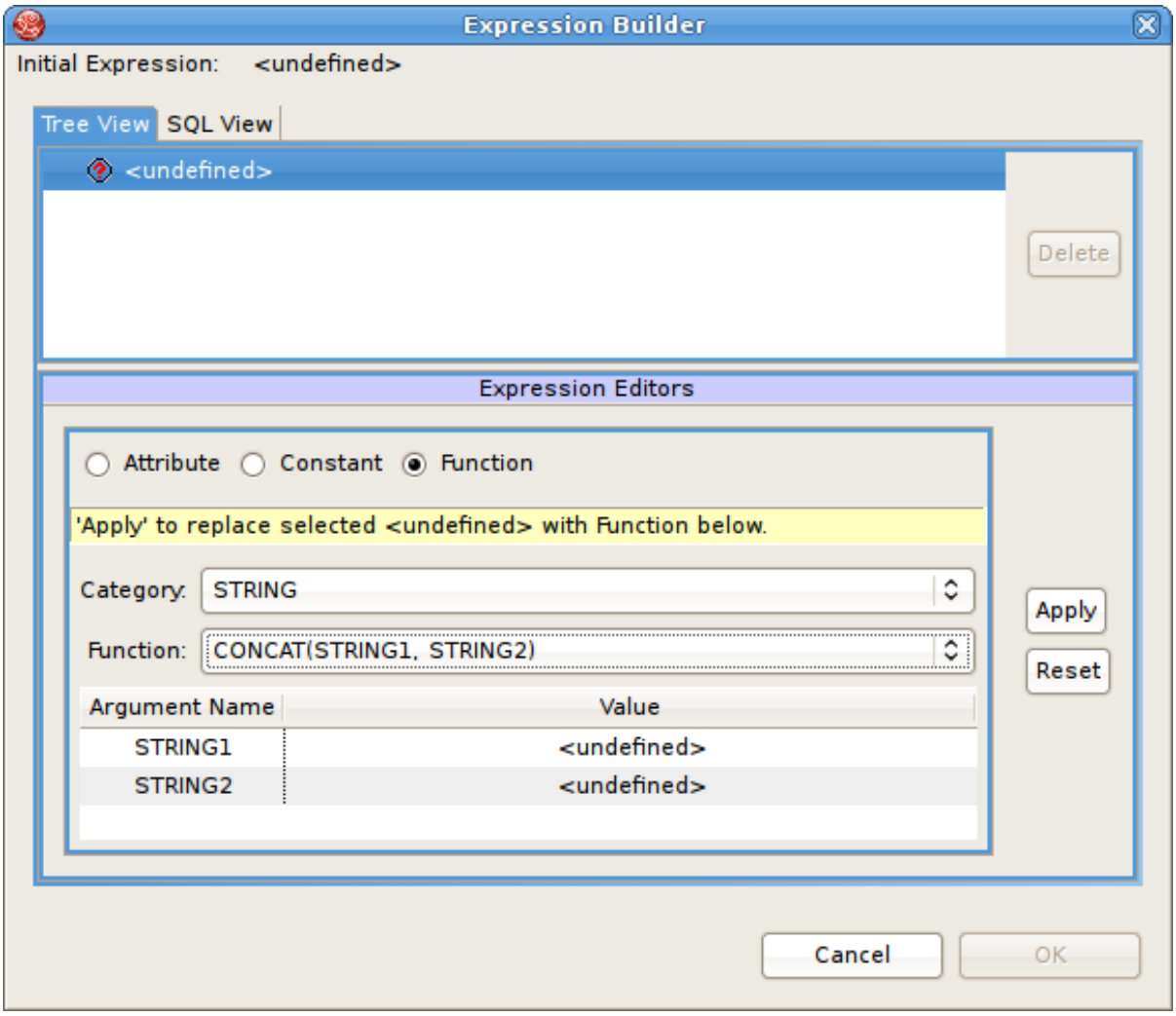


Figure 10.22. Function Panel Selected

From the **`Category`** drop-down list, choose the type of function you want to add. By default, the **Teiid Designer** 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 mathematical and other numeric functions.
- String for string manipulation functions.

NOTE

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

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.

6. Click **Apply**.

7. Your function displays in the tree at the top. Sub nodes display for each argument you need to set for this function.

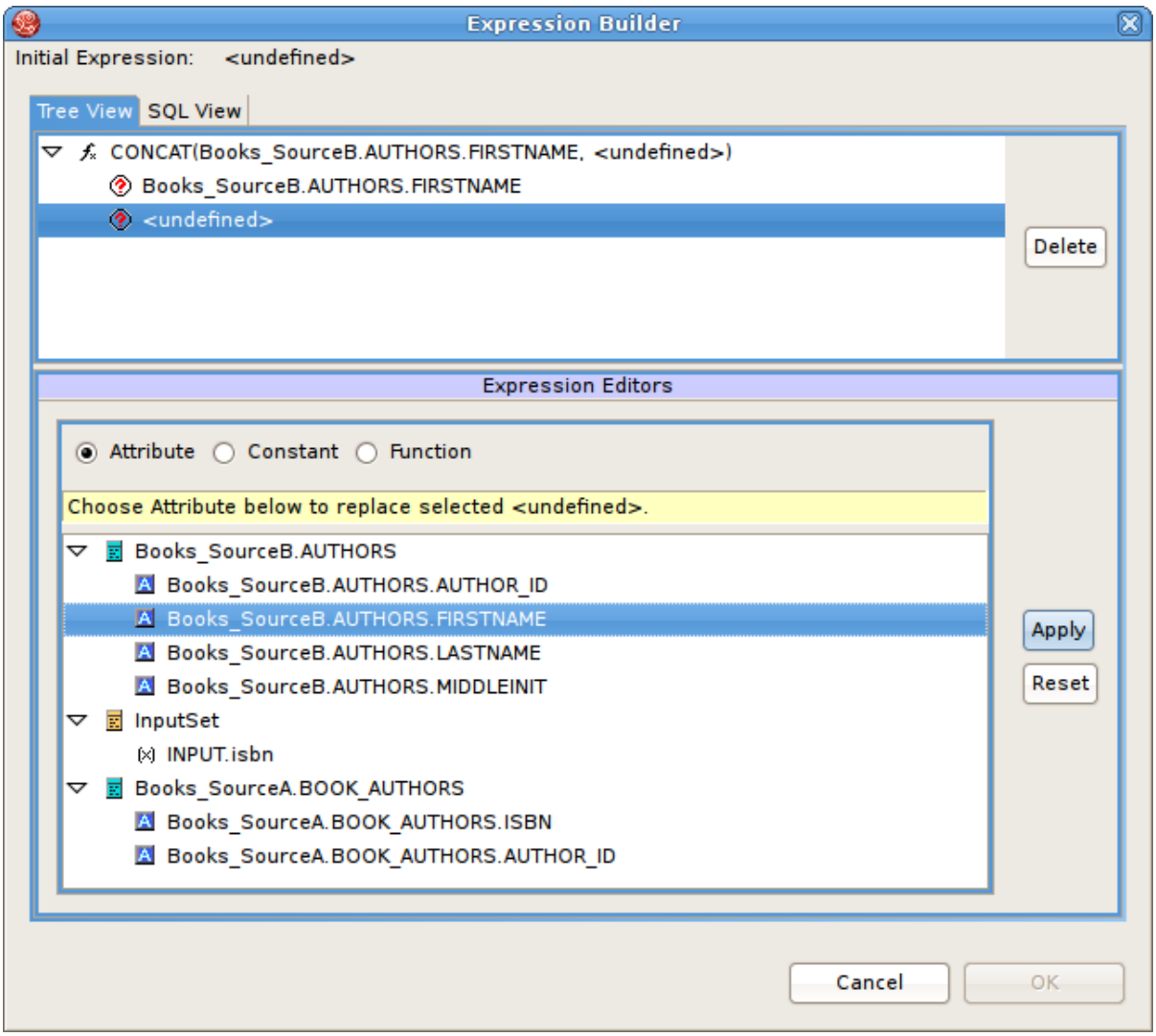


Figure 10.23. New Blank Function Created

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.

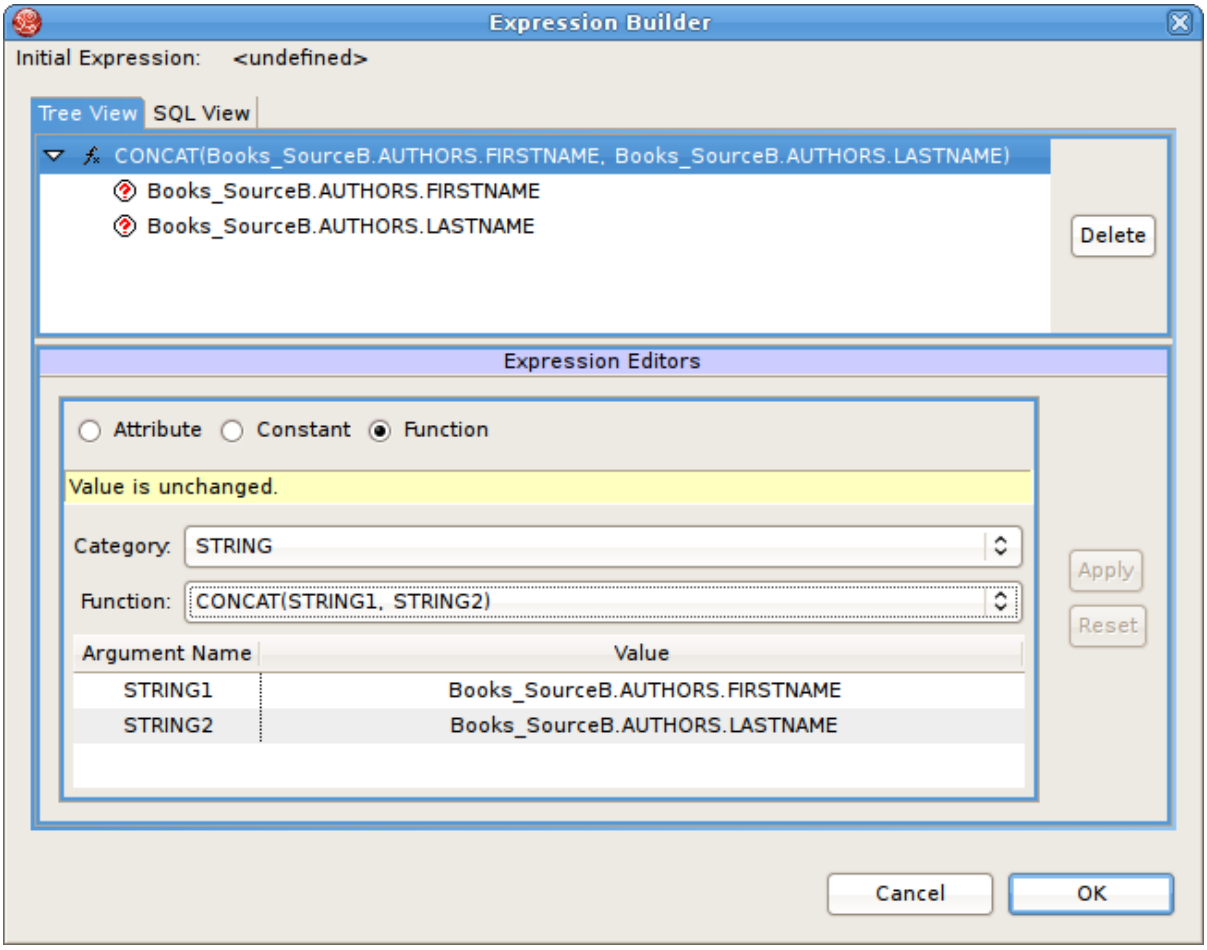


Figure 10.24. Nested Function Example

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.

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.

10.4. INPUT SET EDITOR (XML)

The Input Set represents a special class that contains attributes from a parent mapping class. When you create mapping classes for an XML Document model, the **Teiid Designer** automatically adds an Input Set to all XML transformation diagrams for mapping classes beneath the highest node in the Document meta object.

The Input Set proves especially useful for information integration using the **JBoss Data Virtualization 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 **JBoss Data Virtualization 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 Model Explorer view and you cannot use them as you would a normal class, such as for source classes in other transformations.

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

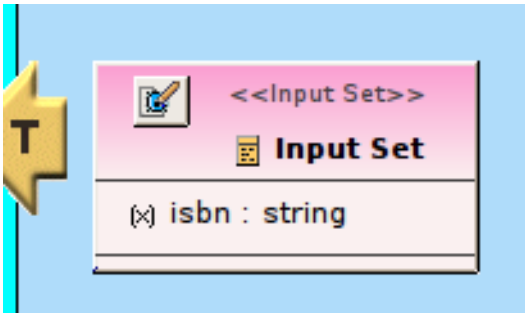


Figure 10.25. Edit Input Set Button

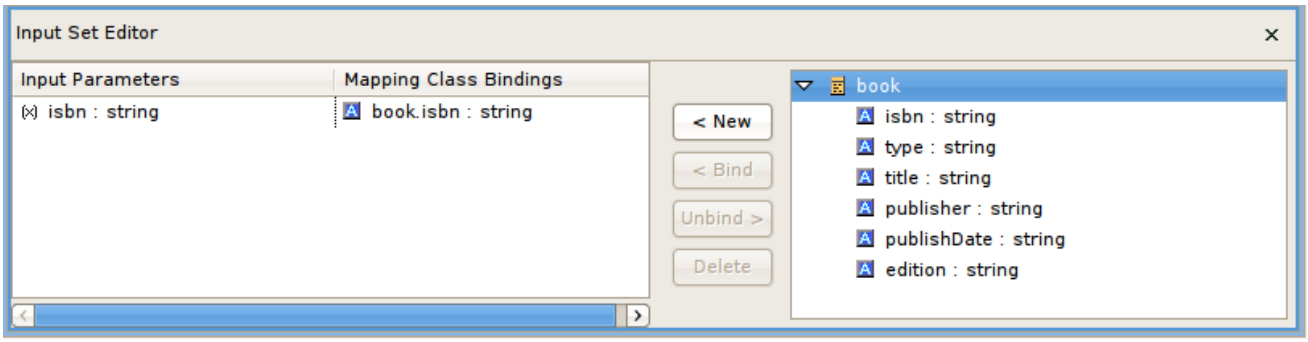


Figure 10.26. Input Set Editor Panel

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.

10.5. CHOICE EDITOR (XML)

10.5.1. 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 **JBoss Data Virtualization 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 click **`Edit`** from the right-click pop up menu.

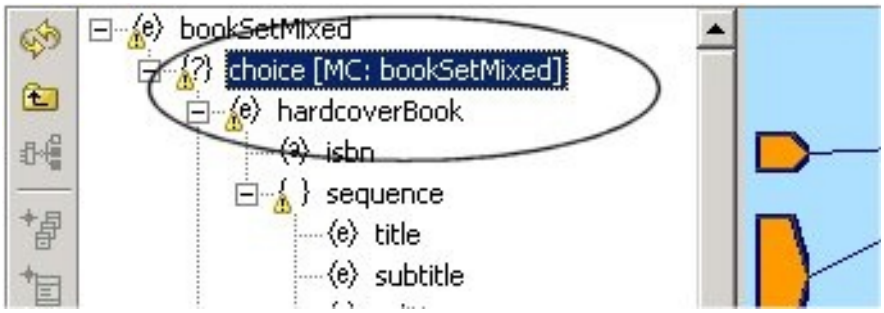


Figure 10.27. Opening The Choice Editor

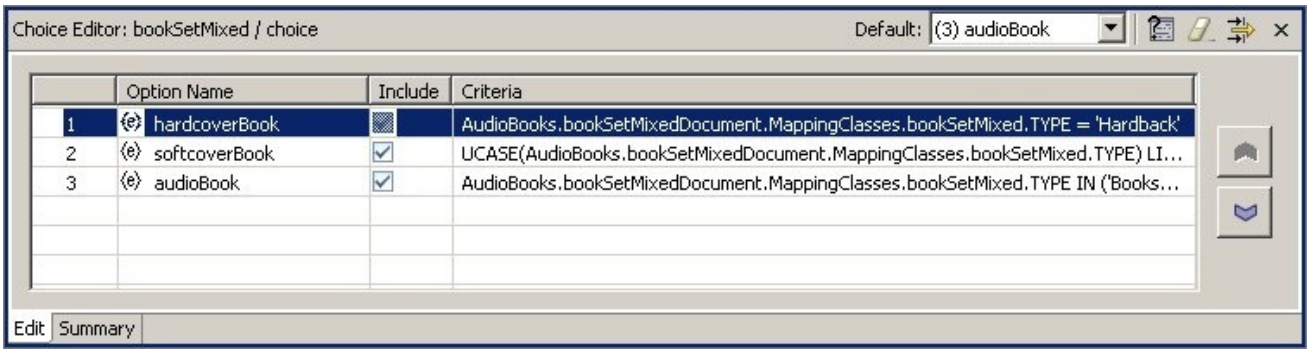


Figure 10.28. The Choice Editor

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 an SQL like version of the current choice criteria.

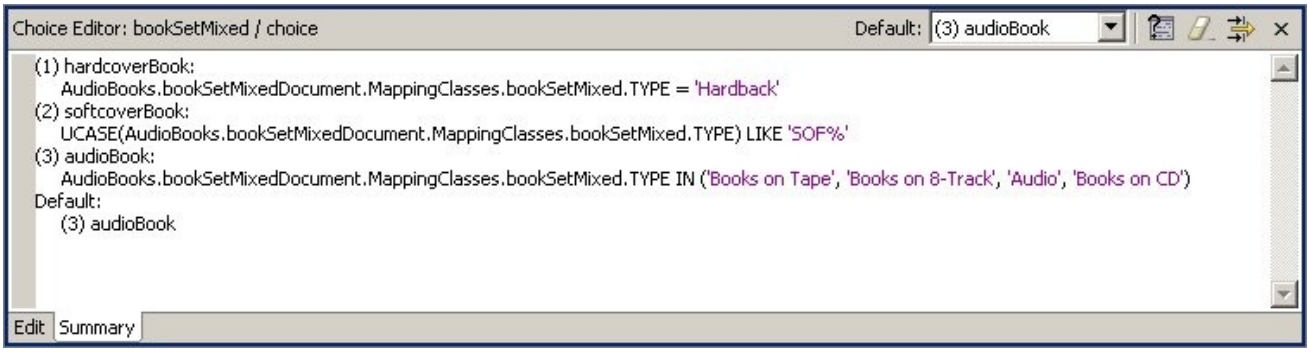


Figure 10.29. Choice Criteria Summary

Using the Choice Editor

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

- Specify a criteria statement for the **JBoss Data Virtualization 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 criteria test order for elements.
- Set a default action that occurs if none of the criteria you set is met.

10.5.2. 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 **JBoss Data Virtualization Server** creates since it will never have credit card information for an XML document.

The table on the **`Choice Editor`** contains the **`Includes`** column. By default, all elements specified by the schema are included. You can click to remove the check mark beside any element you do not want to include within your XML documents generated by this virtual XML document metadata model. By removing the check mark, you are not removing the element from the XML Document model; you are merely telling the **JBoss Data Virtualization 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.

10.5.3. Editing Choice Criteria

To edit the criteria for a choice element:

1. In the table on the **`Choice Editor`** panel, select the element you want to edit..

2. Click **`Edit Criteria`** button to launch the **`Criteria Builder`** dialog.
3. Use the **`Criteria Builder`** to create the conditions for which the **JBoss Data Virtualization Server** will test to determine whether to choose this option in the XML instance document.
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.

10.5.4. Setting Choice Element Order

To edit the criteria for a choice element:

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

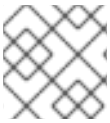
To set this order, select an element in the table and use the or button to move it into a new position in the table. The new order displays both in the table and in the Choice Criteria box and reorders the XML document as well.

10.5.5. Setting a Default Choice Action

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



NOTE

You must set a default action for your choice criteria.

10.6. RECURSION EDITOR (XML)

10.6.1. What is Recursion?

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

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

- The first cycle can be thought of as an entry condition for the recursion. The mapping class located at this node defines a normal mapping transformation like that of any other in the document model.
- The second cycle defines a mapping transformation that will be performed repeatedly until conditions are met that will halt the document instance being generated by the JBoss Data Virtualization 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. Use recursion only when the data access pattern from your data source(s) is also recursive; in other words, when the same query transformation is to be executed over and over to generate and map the nested document's data content.

By default, the Teiid Designer does not mark the recursive fragments in document models to execute recursively in the JBoss Data Virtualization Server. To take advantage of this behavior, you must open the Recursion Editor in the recursive mapping class , mark the transformation query as recursive, and specify the recursion limit properties.

10.6.2. Recursion Editor (XML)

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.
- An SQL based criteria limit condition.
- A combination of both.

To open the Recursion Editor, click the **`Recursion Editor`** button on the displayed mapping class.

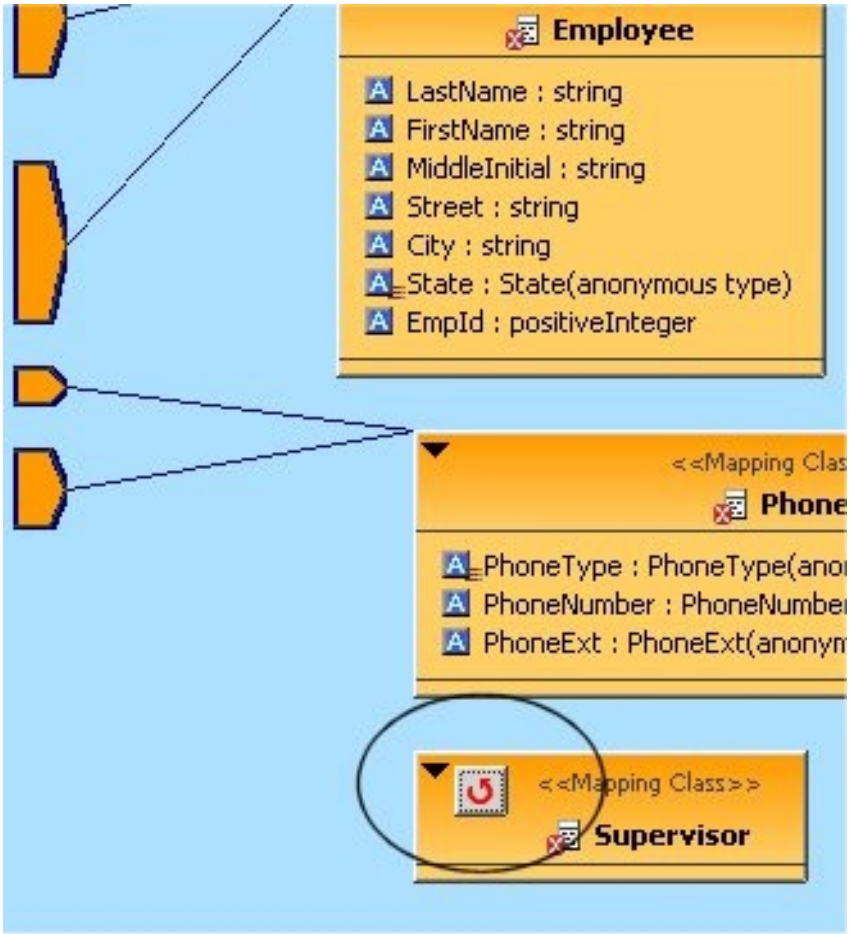


Figure 10.30. Open Recursion Editor Button

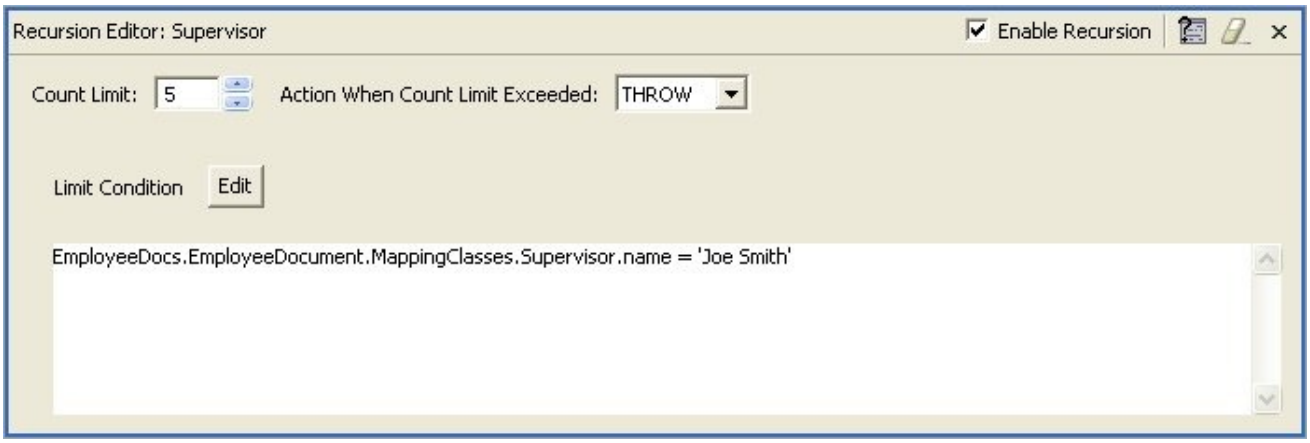


Figure 10.31. Recursion Editor

10.6.3. Edit Recursion Properties

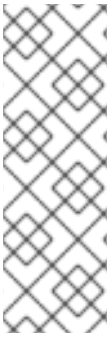
To edit recursion properties:

1. Select the Enable Recursion checkbox if you want the **JBoss Data Virtualization Server** to perform the query you specify to generate the nested tags within the XML document.

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 **JBoss Data Virtualization Server**. For more information about this limit, contact your system administrator.

3. Click the **`Action When Count Limit Exceeded`** drop-down menu to instruct the **JBoss Data Virtualization Server** what to do if it encounters more results for the query than the count limit before it reaches the limit condition.

4. Click the **`Edit`** button to launch the SQL to build a limiting condition for this recursion.



NOTE

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

10.7. 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.

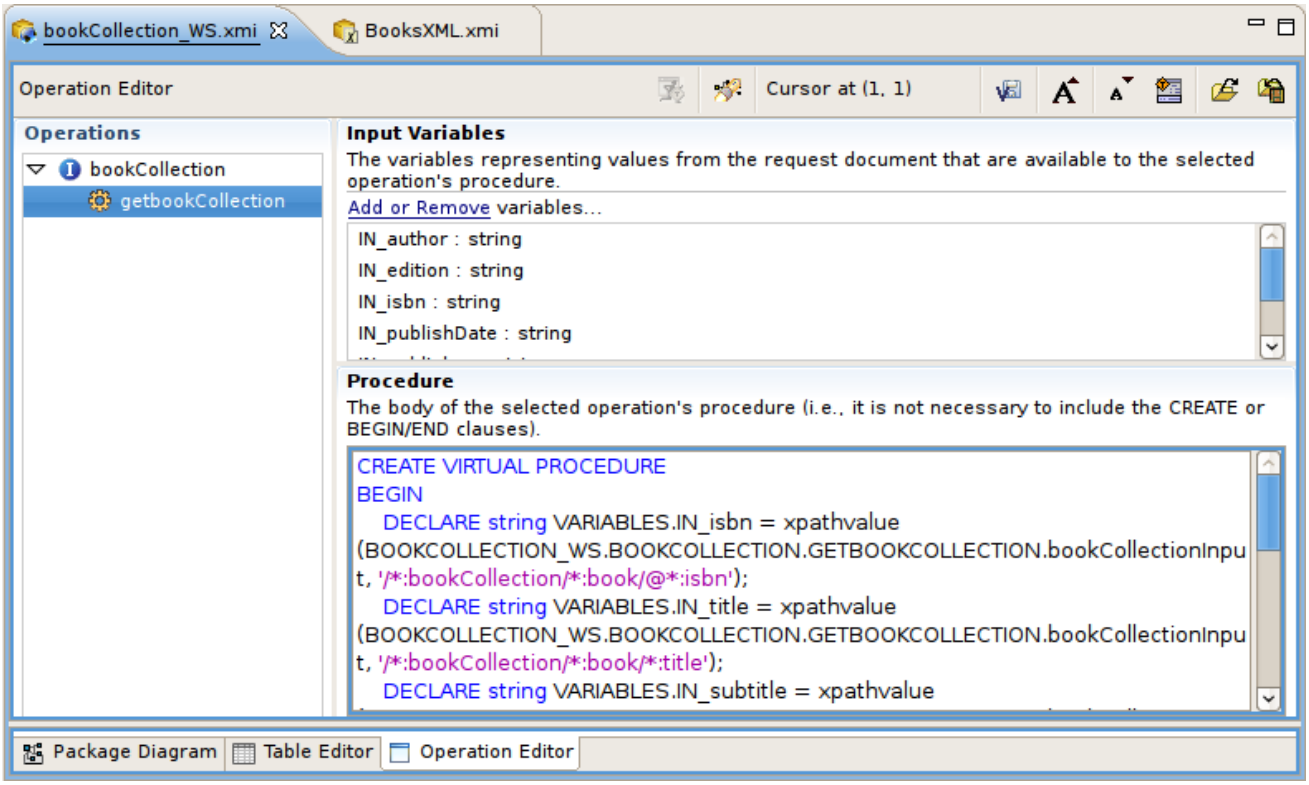


Figure 10.32. Operation Editor

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 **`Add`** or **`Remove`** 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 **`Add`** or **`Remove`** link will display the following dialog:

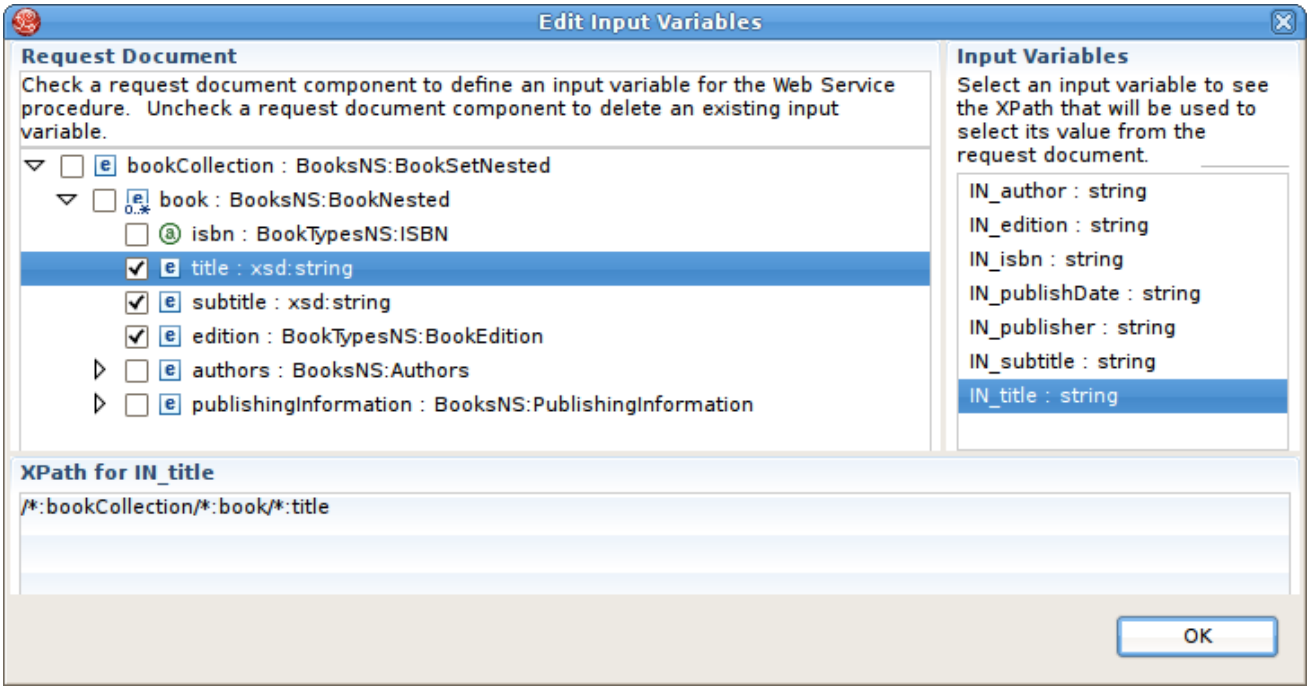


Figure 10.33. Edit Input Variables Dialog

10.8. MANAGING MODEL OBJECT EXTENSIONS

10.8.1. 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 Data Services translator. The **Teiid Designer** model extension framework consists of:

- Model Extension Definitions (MEDs)
- MED Registry - keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model.
- MED Editor

10.8.2. 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.

NOTE

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

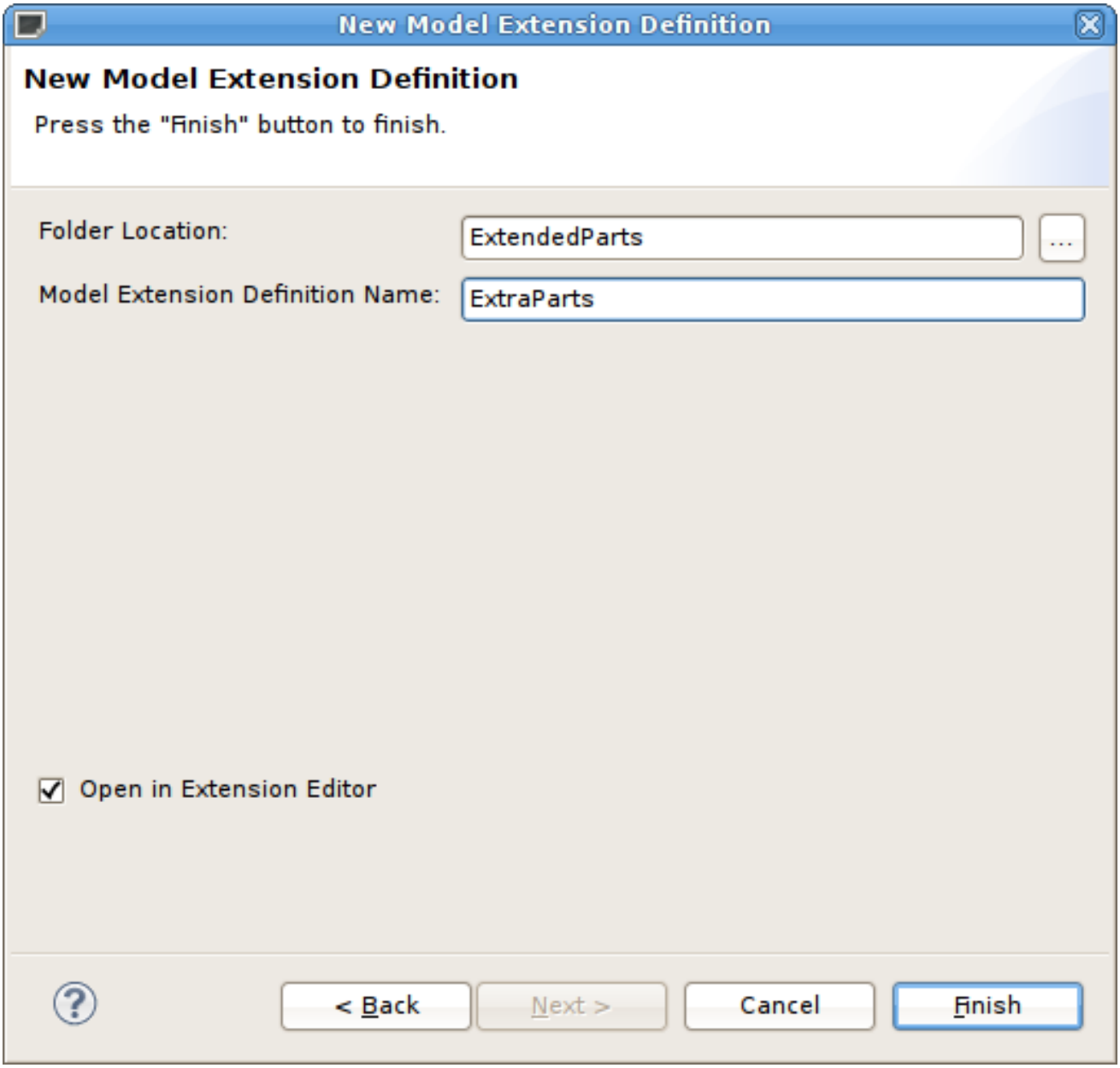


Figure 10.34. MED Editor Overview Tab

Browse and select existing project or project folder location for MED file and specify unique file name. Next you have two options click **`Finish`** to create MED or click **`Next`** and add more properties of MED which shows figure below.

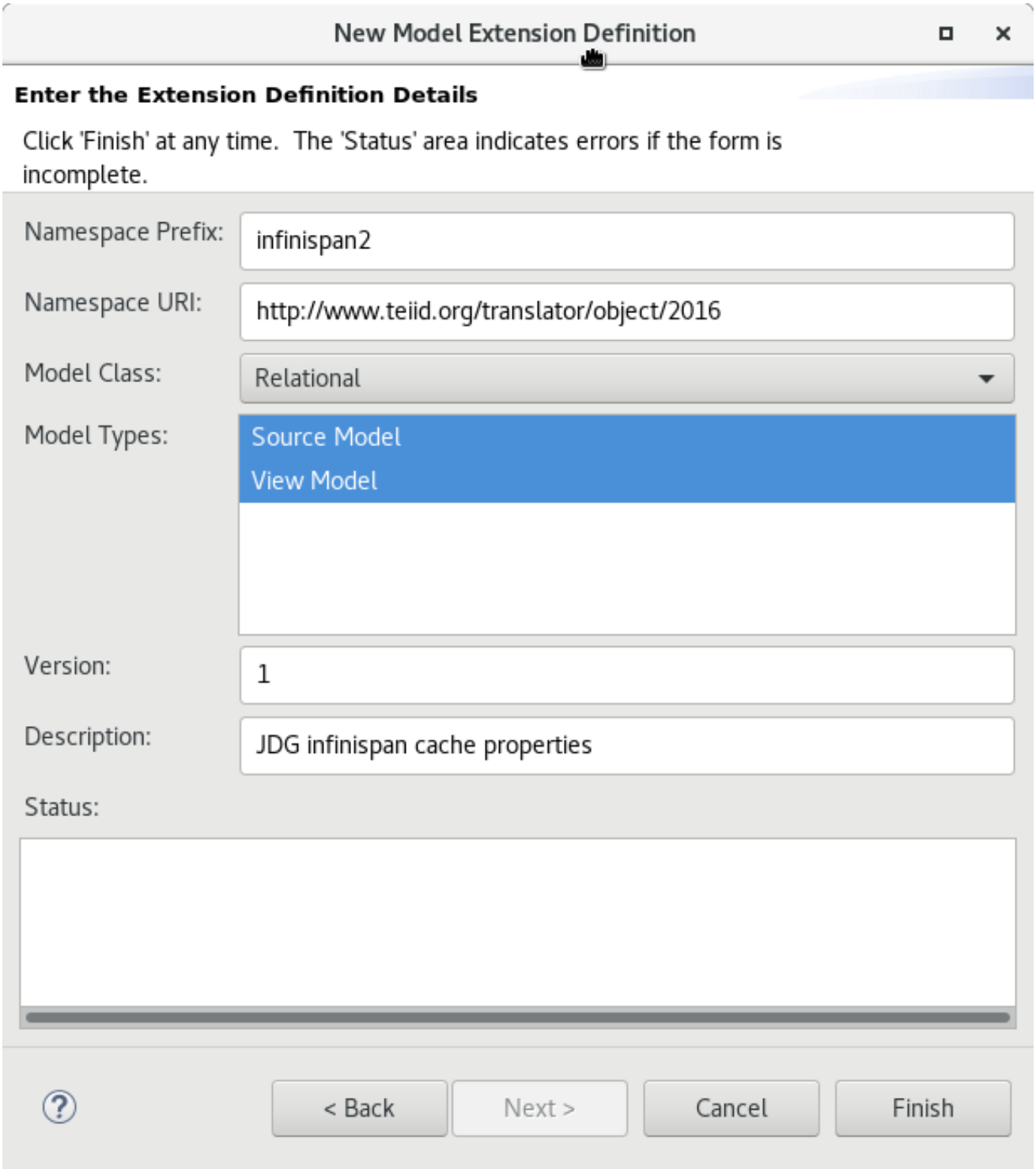


Figure 10.35. Extension Definition Details Tab

10.8.3. 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 open to allow editing .

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.

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.

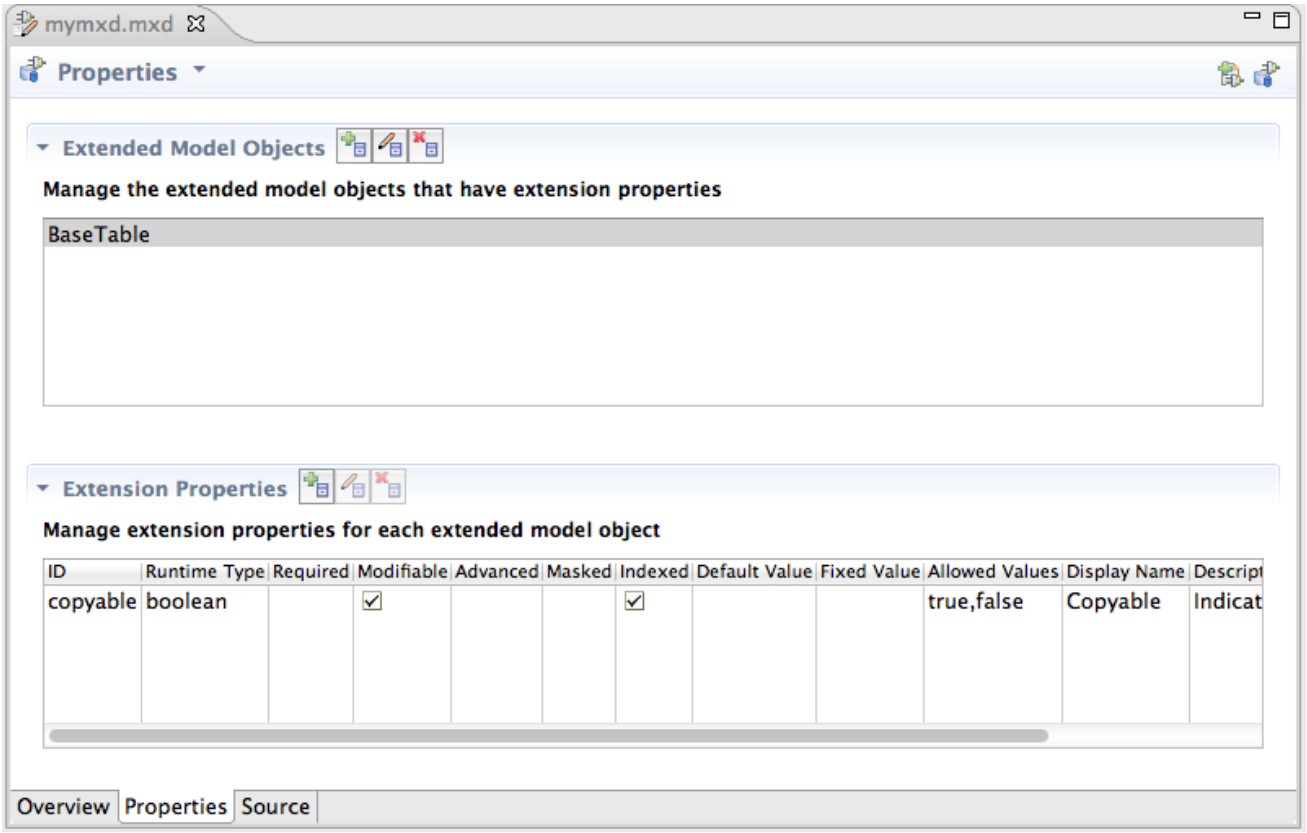


Figure 10.36. MED Editor Properties Tab

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

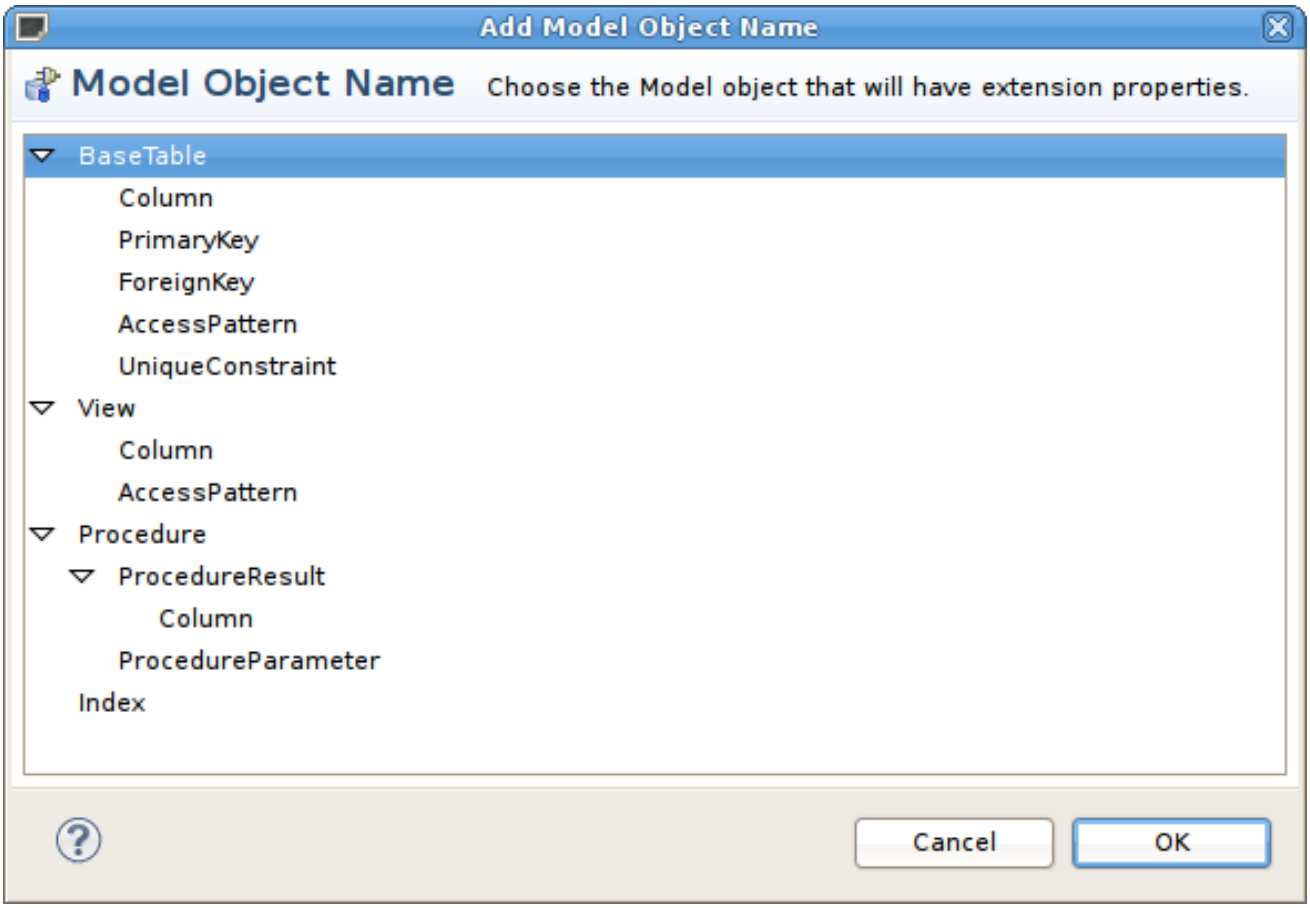


Figure 10.37. Select Model Object Name Dialog

Next, select the model object in the **`Extended Model Objects`** section and use the actions and properties 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.

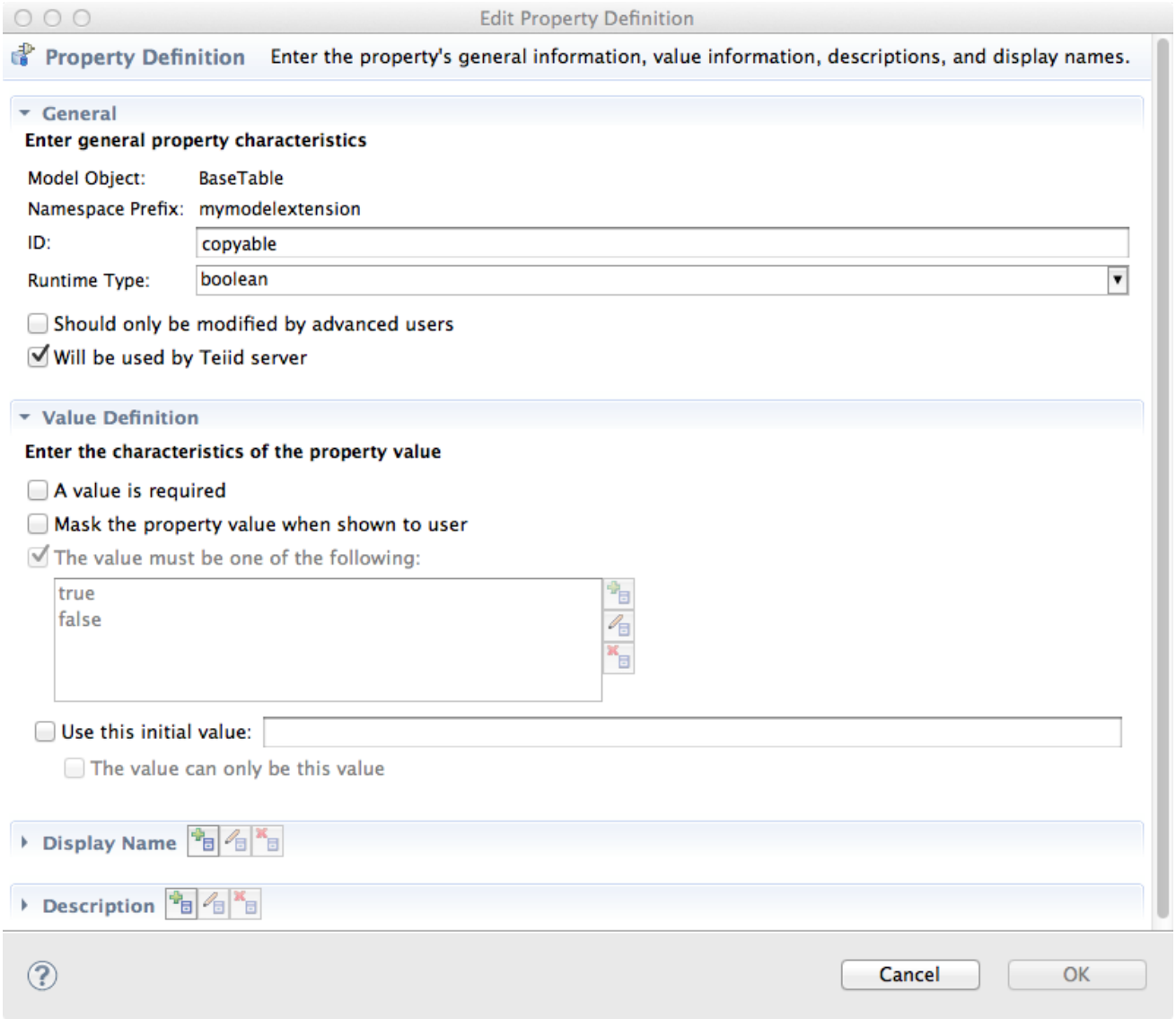


Figure 10.38. Edit Property Definition Dialog

10.8.4. 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.

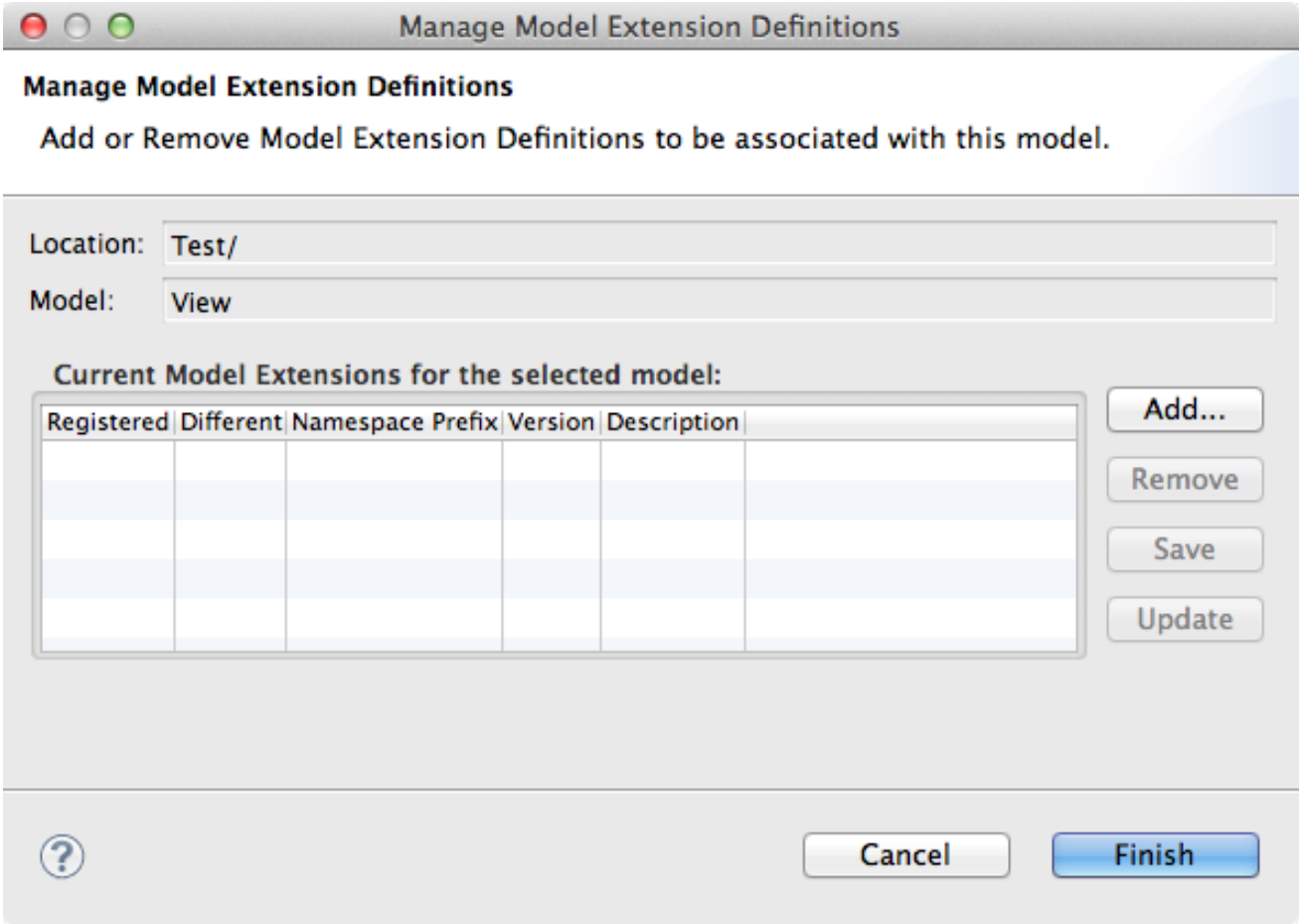


Figure 10.39. Manage Model Extension Definitions Dialog

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

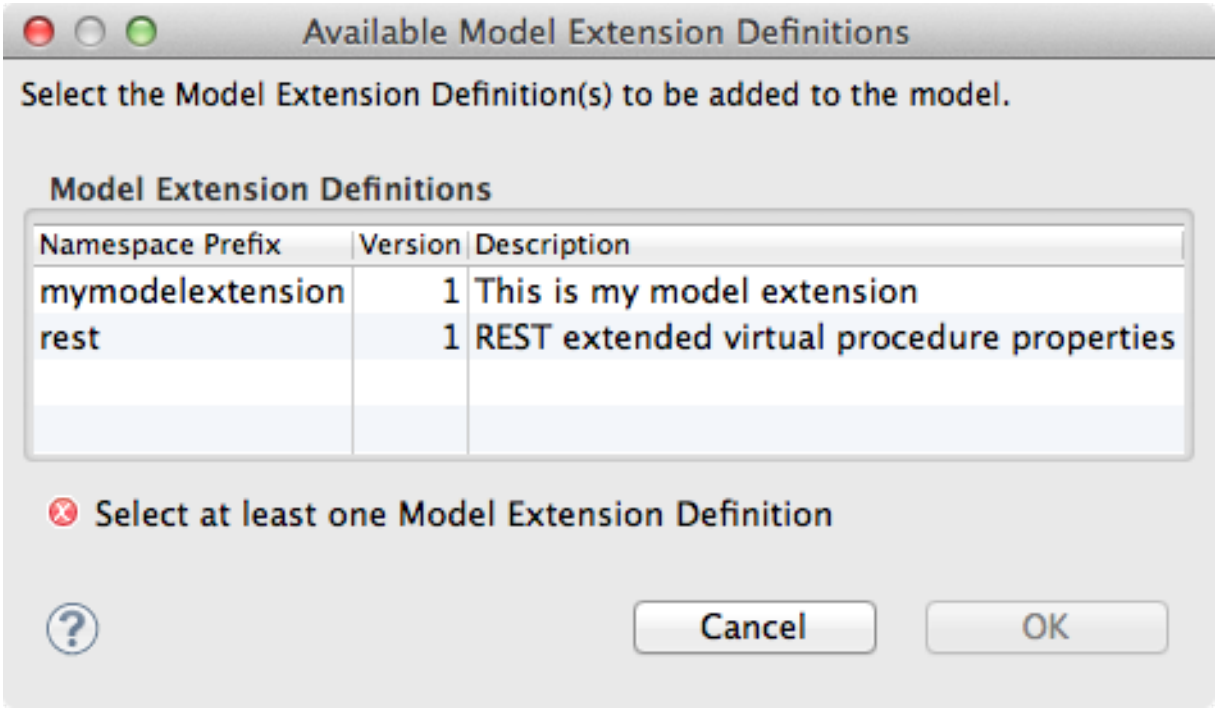


Figure 10.40. Add Model Extension Definitions Dialog

NOTE

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

10.8.5. Setting Extended Property Values

Extension properties are user defined properties available to any extended model object via the **`Properties View`**. All extension properties are available 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.

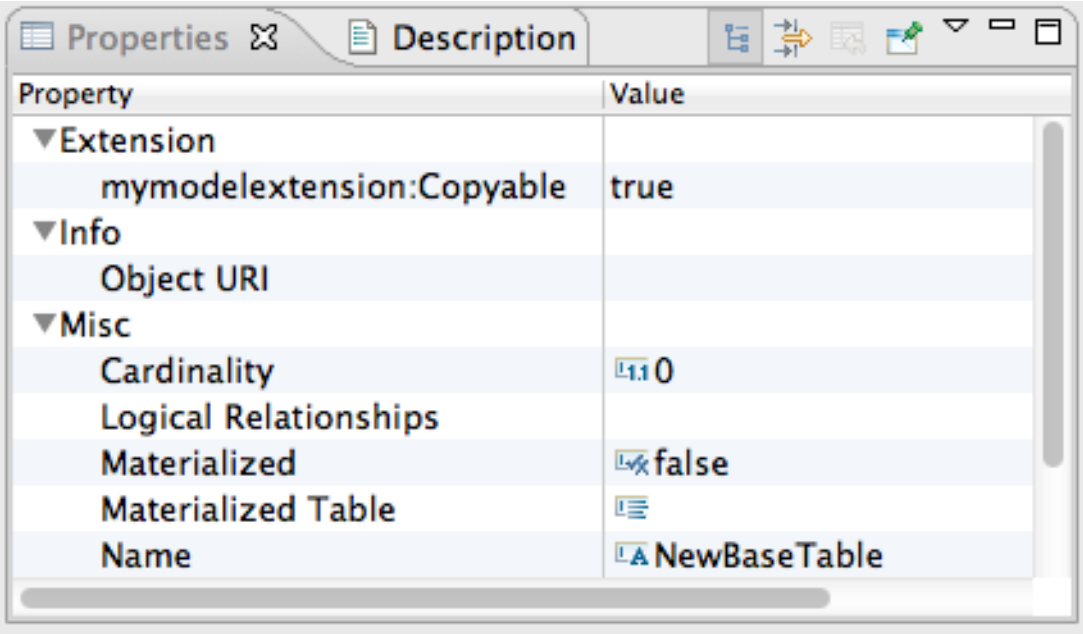


Figure 10.41. Properties View For Extended Model Object

CHAPTER 11. USER DEFINED FUNCTIONS

11.1. USER DEFINED FUNCTIONS

Teiid Designer enables you to extend Teiid's function library by using custom or User Defined Functions(UDFs). You can define a UDF with follwing properties.

- Function Name - When you create the function name, remember that:
  - 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- Input Parameter defines a type specific signature list. All arguments are considered required.
- Return Type- the expected type of the returned scalar value.
- Pushdown - Indicates the expected pushdown behavior. It can be one of REQUIRED, NEVER, ALLOWED. If NEVER or ALLOWED are specified then a Java implementation of the function must 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 always returns 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 leads to better performance.

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 is raised if a pushdown required function cannot be evaluated by the appropriate source.

11.2. MODELING YOUR FUNCTIONS

Teiid Designer provides an object wizard to assist you in modeling your User Defined Functions. Right-click the existing Relational View Model and click New Child - Procedure and then click on the radio button for "User Defined Function". In the wizard, you can see an option to specify the function's jar location on your file system, as well as the Java class and the Java method.

If you create a function using alternative methods, you must select the procedure and edit the different properties for the procedure in the **`Properties`** view.

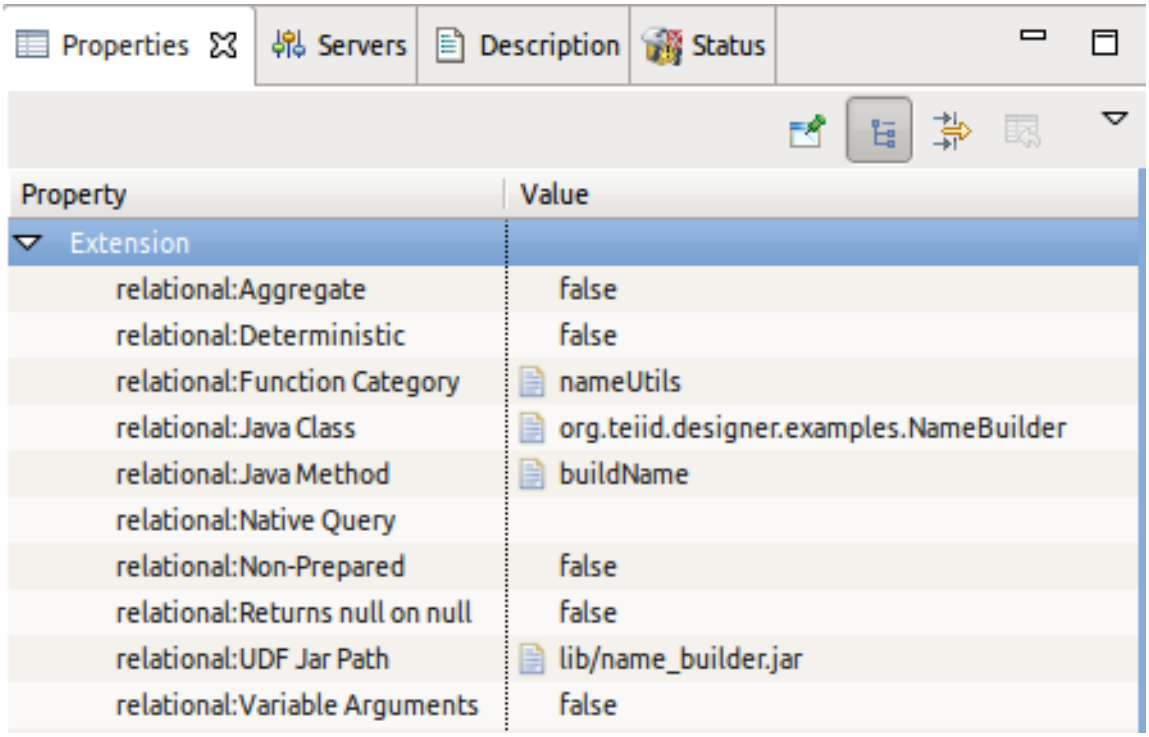


Figure 11.1. VDB UDF Jar Files

A User Defined Function represents a defined method in a java class. In order for the Teiid runtime to register the function and allow its use in the transformations, you must specify the following as properties:

- Function Category - unique name for a grouping of user defined functions. This category is displayed 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.

11.3. UTILIZING YOUR FUNCTIONS IN TRANSFORMATIONS

Once you have finished modeling your functions, you can use your function calls in your transformation SQL. These functions are accessible through the Transformation Editor's expression builder wizard.

11.4. INCLUDING FUNCTIONS IN YOUR VDB

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

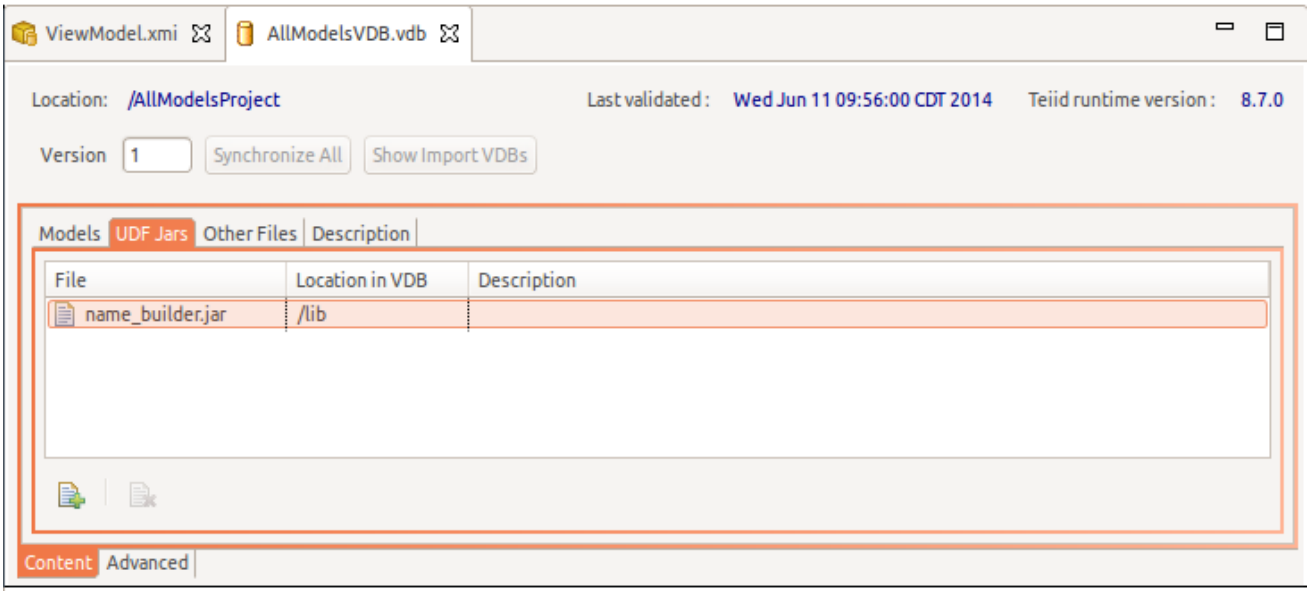


Figure 11.2. VDB UDF Jar Files

CHAPTER 12. METADATA-SPECIFIC MODELING

12.1. RELATIONAL SOURCE MODELING

12.1.1. Source Function

To improve ability to utilize database functions within View transformations, Source Function action and wizard is added to Teiid Designer. Source Function assists you in building a source procedure that conforms to a function structure, including input and output parameters. Right click on source model and select **`New Child > Procedure...`** to open the **`Procedure Type`** dialog.

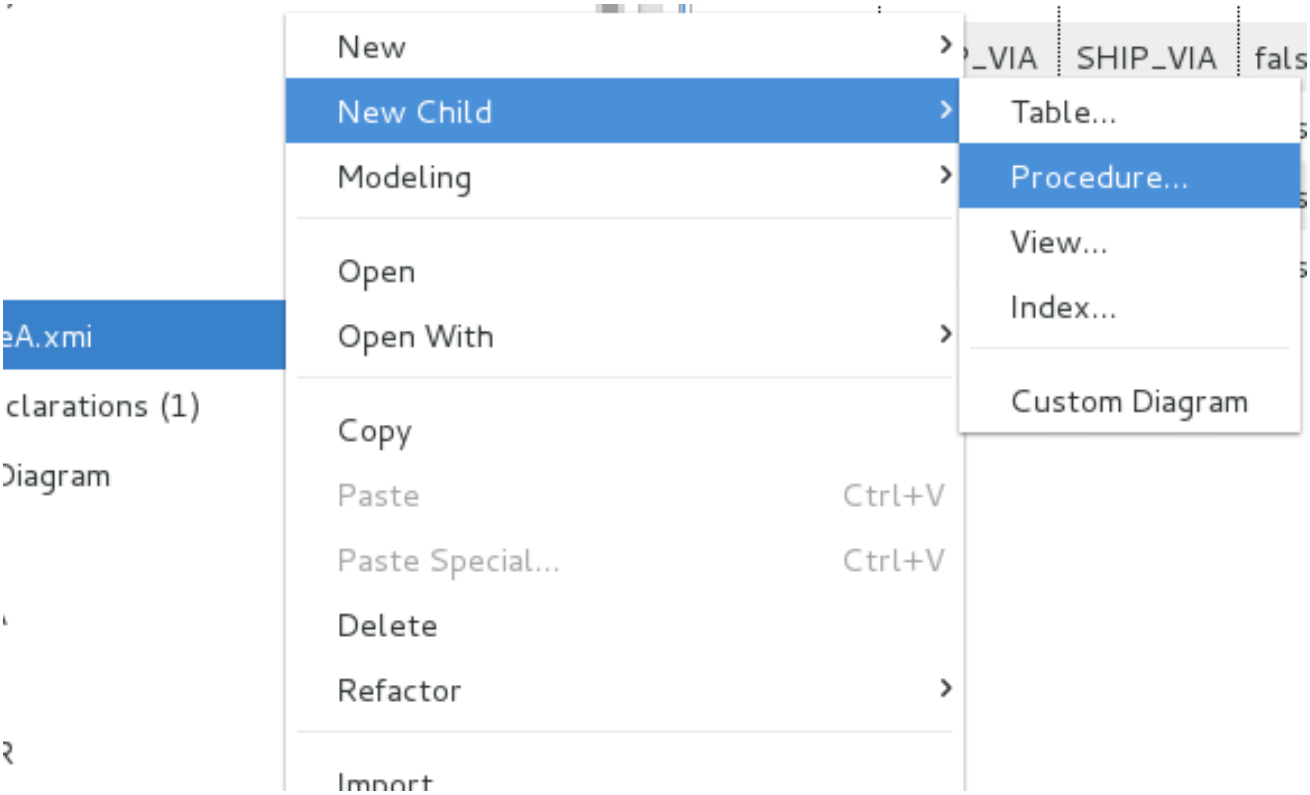


Figure 12.1. New Source Function Action

Select **`Source Function`** action to open **`Create Relational Source Function`** dialog. Enter your database function name, define input parameters including datatype and length, specify output parameter info, set options and click **`OK`**.

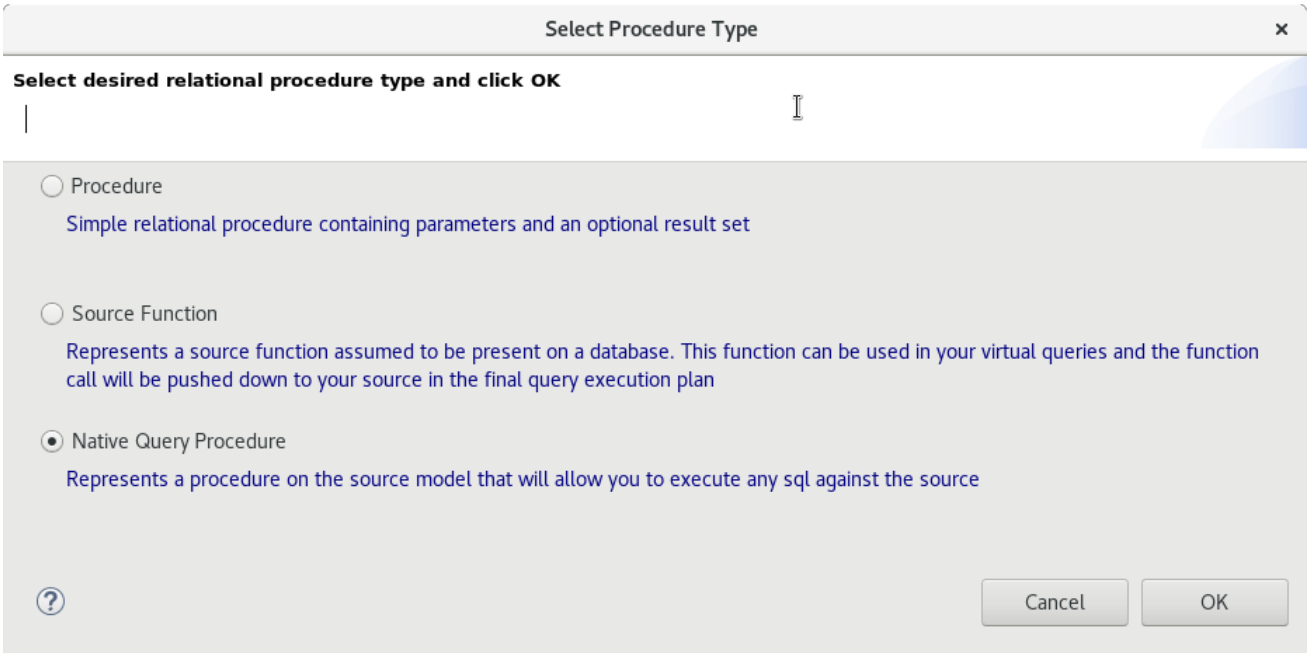


Figure 12.2. New Source Function Action

The resulting source function will be added to your model and will be represented by the icon.

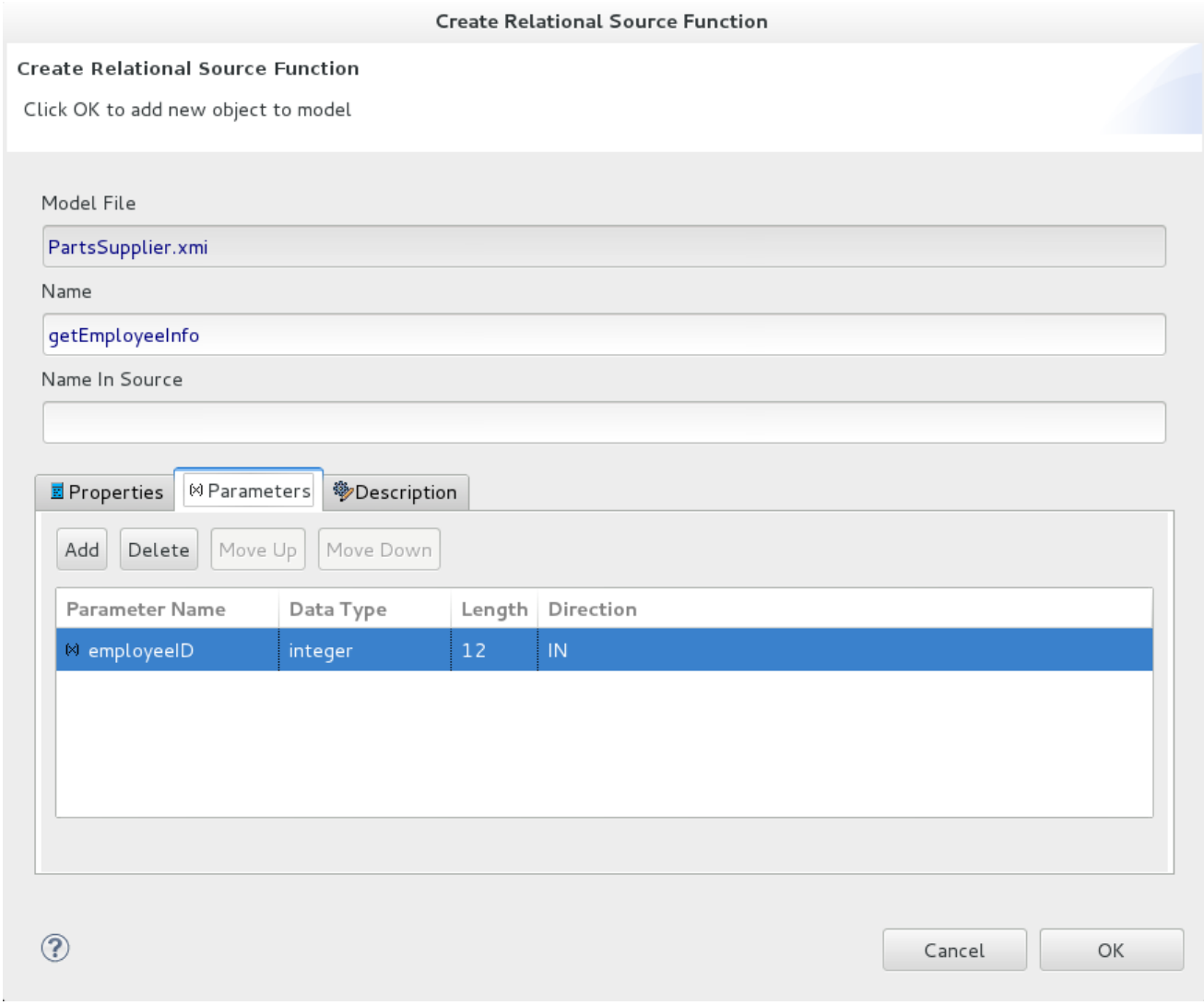


Figure 12.3. Create New Source Function Dialog

When finished, the new source function will be displayed in your model's package diagram.

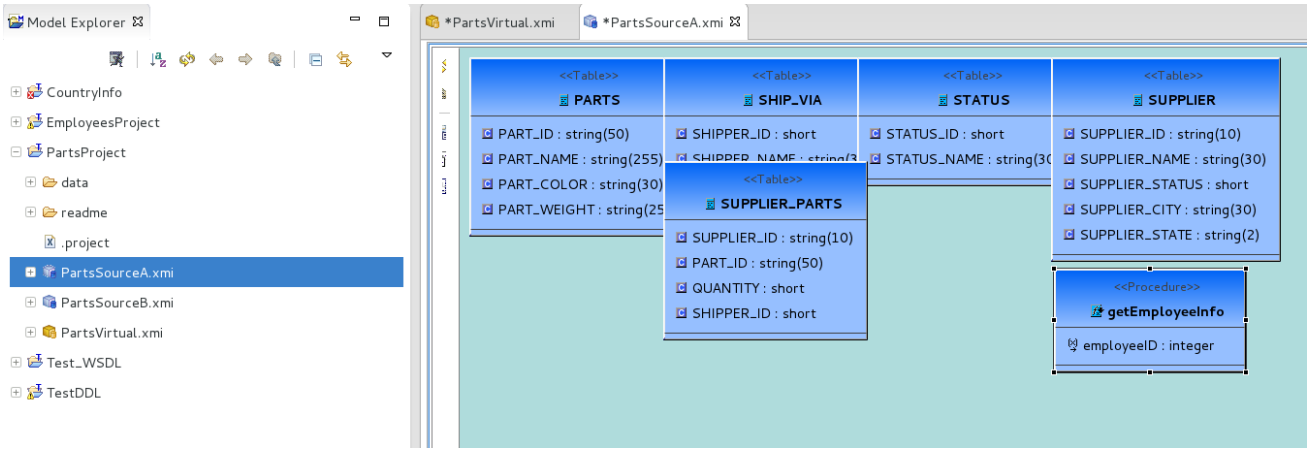


Figure 12.4. New Source Function In Package Diagram

After saving your model, your new source function will be available for use in your transformations. If you open the Expression builder, your source functions will be selectable in the Function drop-down selector for a Category named for the model.

12.1.2. Create Relational Table Wizard

Right click on source model and select action **`New Child > Table...`** to create a table.

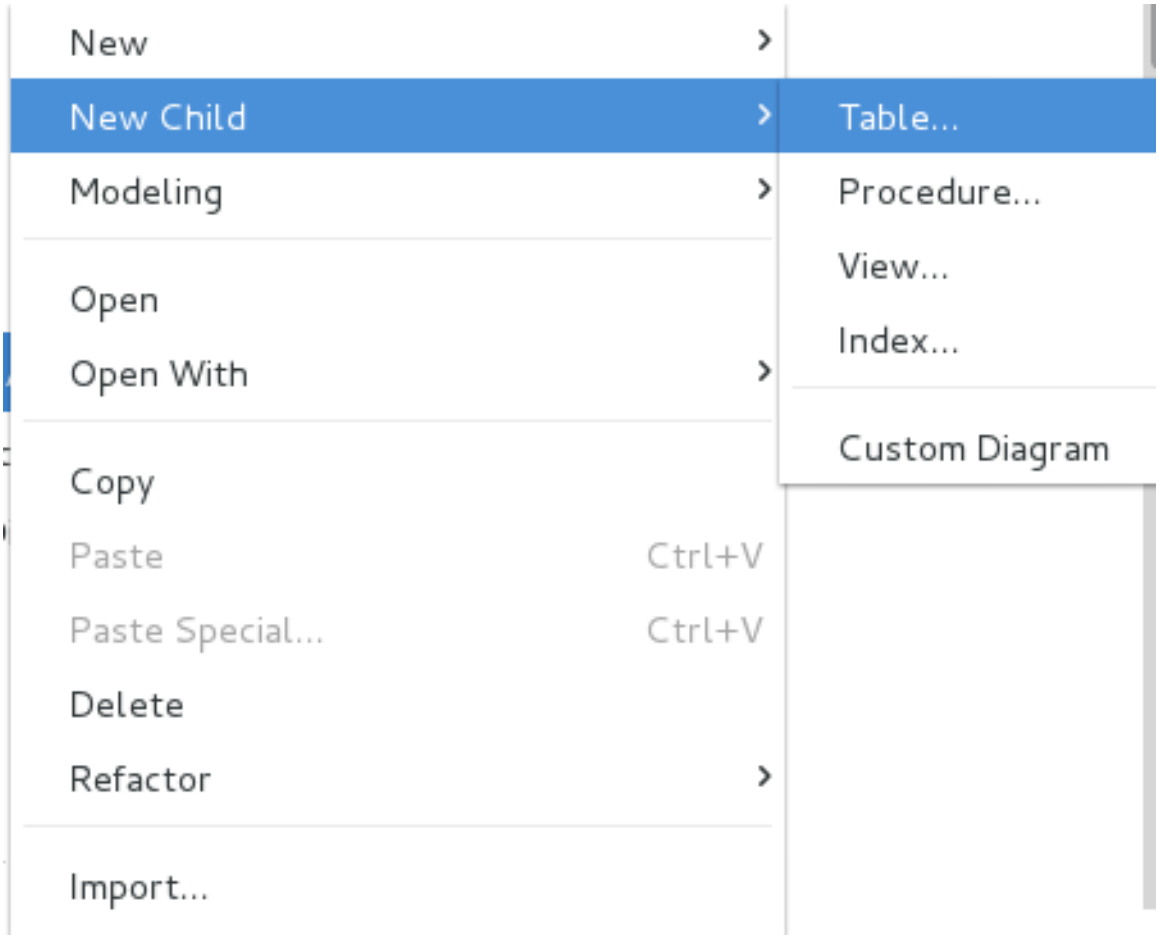


Figure 12.5. New Relational Table Wizard Action

Running the action will display the **`Create Relational Table`** wizard. The wizard page contains

tabbed panels representing the various properties and components that make up the possible definition of a relational table. Enter your table name, define columns, keys, constraints and other options, then click **OK**.

This wizard is designed to provide feedback as to the completeness of the relational table information as well as the validation state of the table and its components. Note that although errors may be displayed during editing, the wizard is designed to allow finishing with the construction of an incomplete table containing errors.

The first tab labeled **Properties** contains the input for the simple table properties including name, name in source, cardinality, supports update and is system table properties.

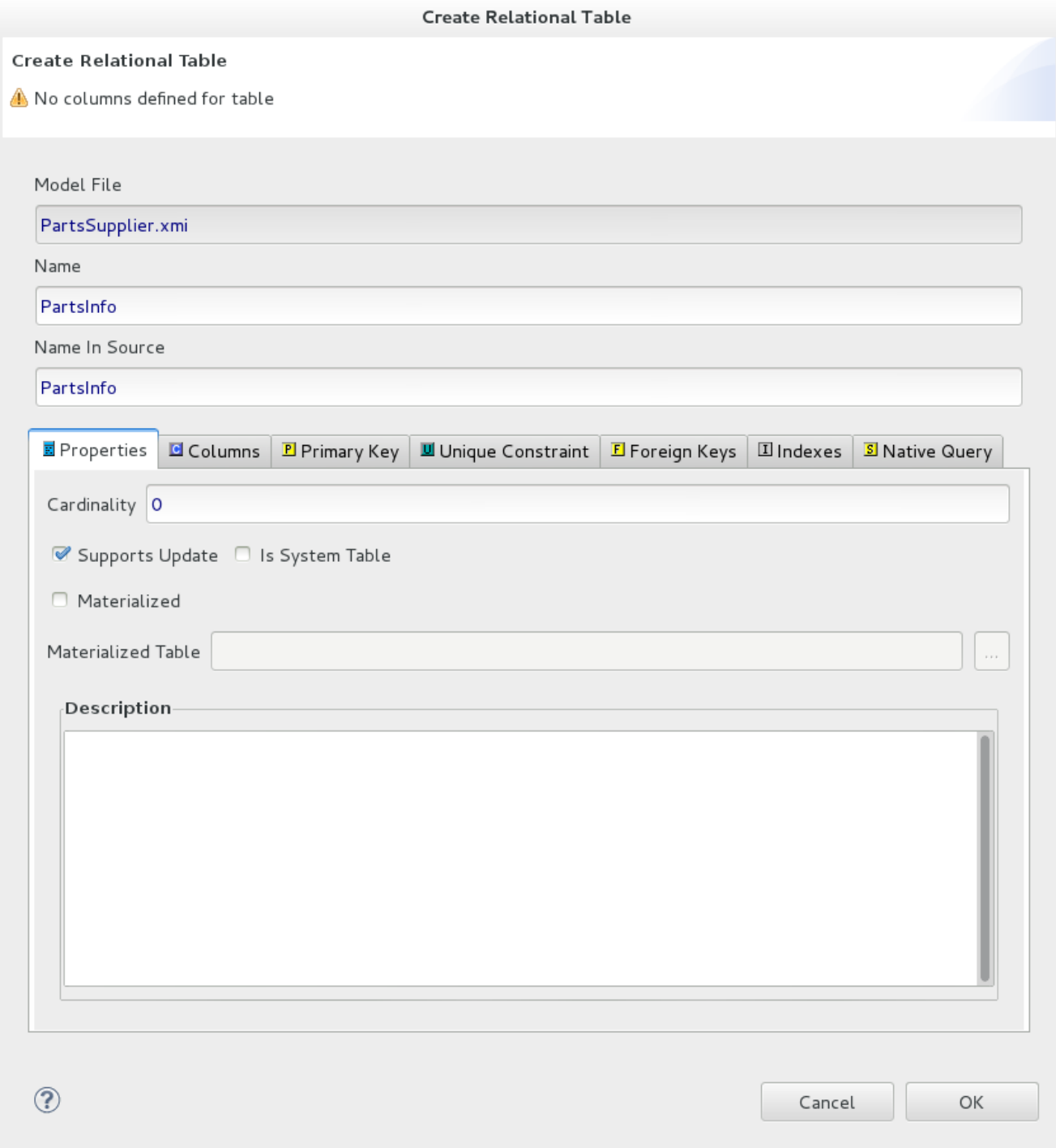


Figure 12.6. Properties Tab

The **Columns** tab allows creation and editing of basic relational columns. This includes adding, deleting or moving columns as well as changing the name, datatype and length properties.

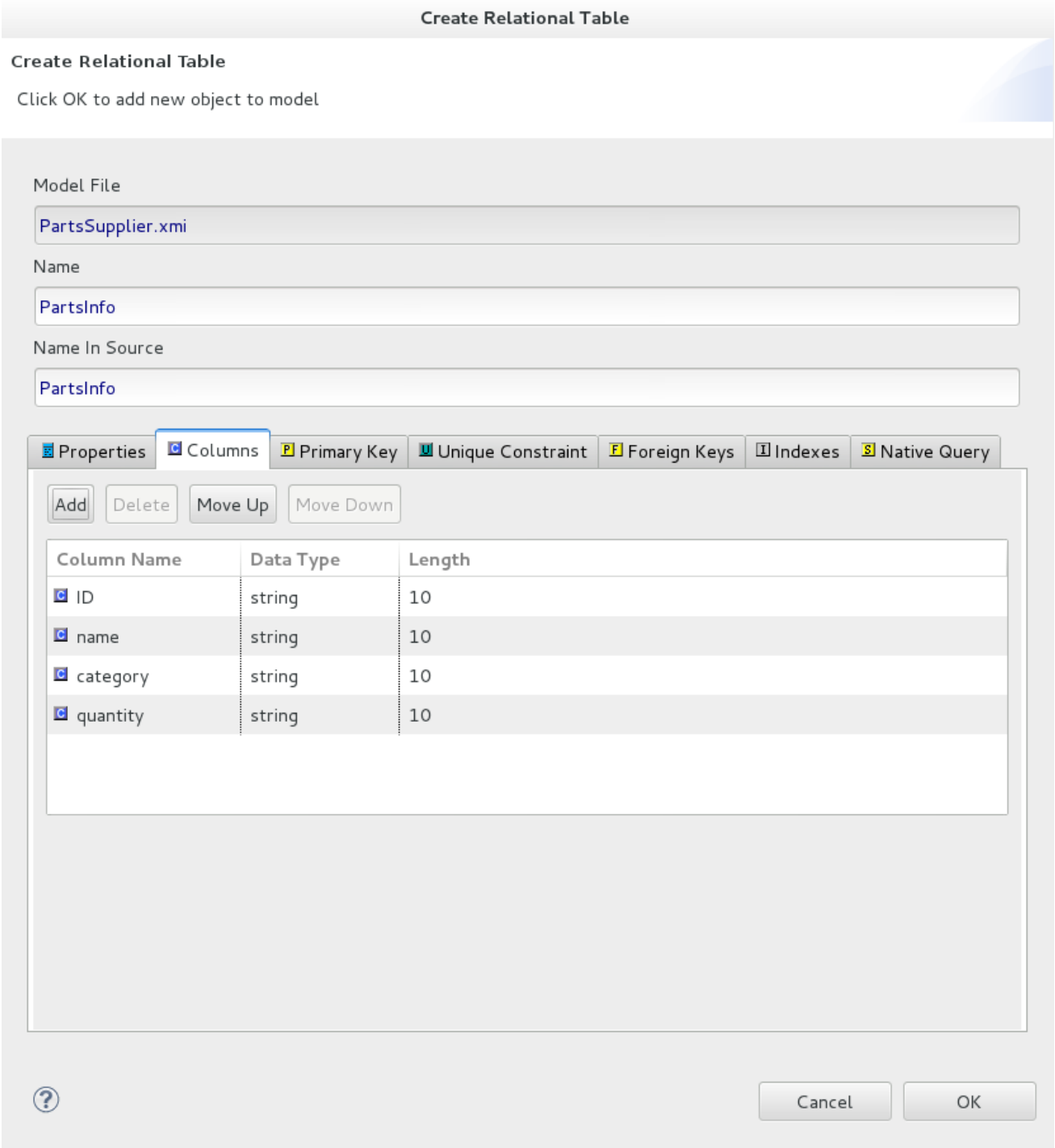


Figure 12.7. Columns Tab

The **`Primary Key`** tab allows editing of the name, name in source and column definitions.Note that clearing the box will clear the data. The **`Unique Constraint`** tab contains the identical information.

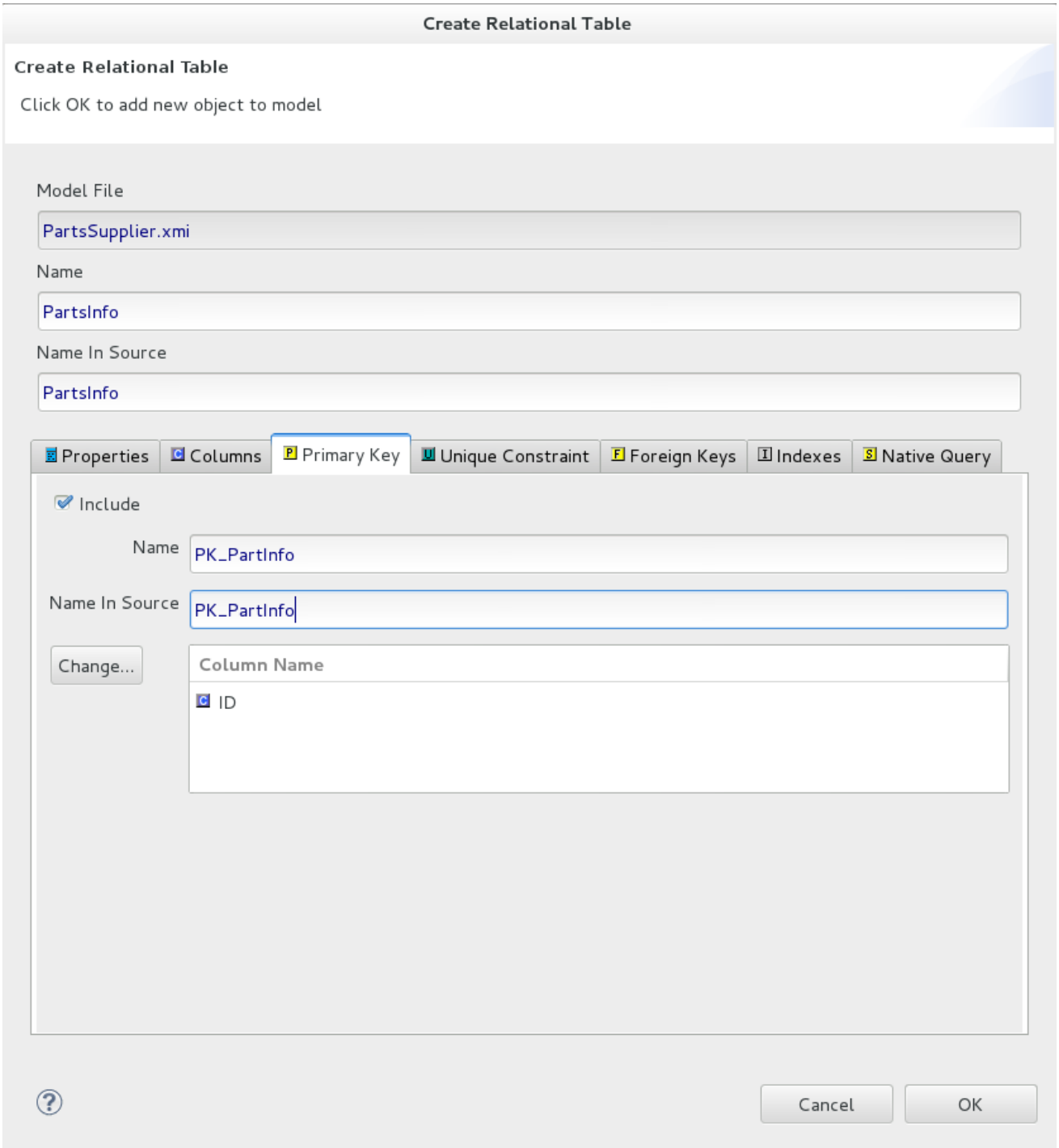


Figure 12.8. Primary Key Tab

The **`Foreign Keys`** tab allows creating, editing and deleting multiple foreign keys.

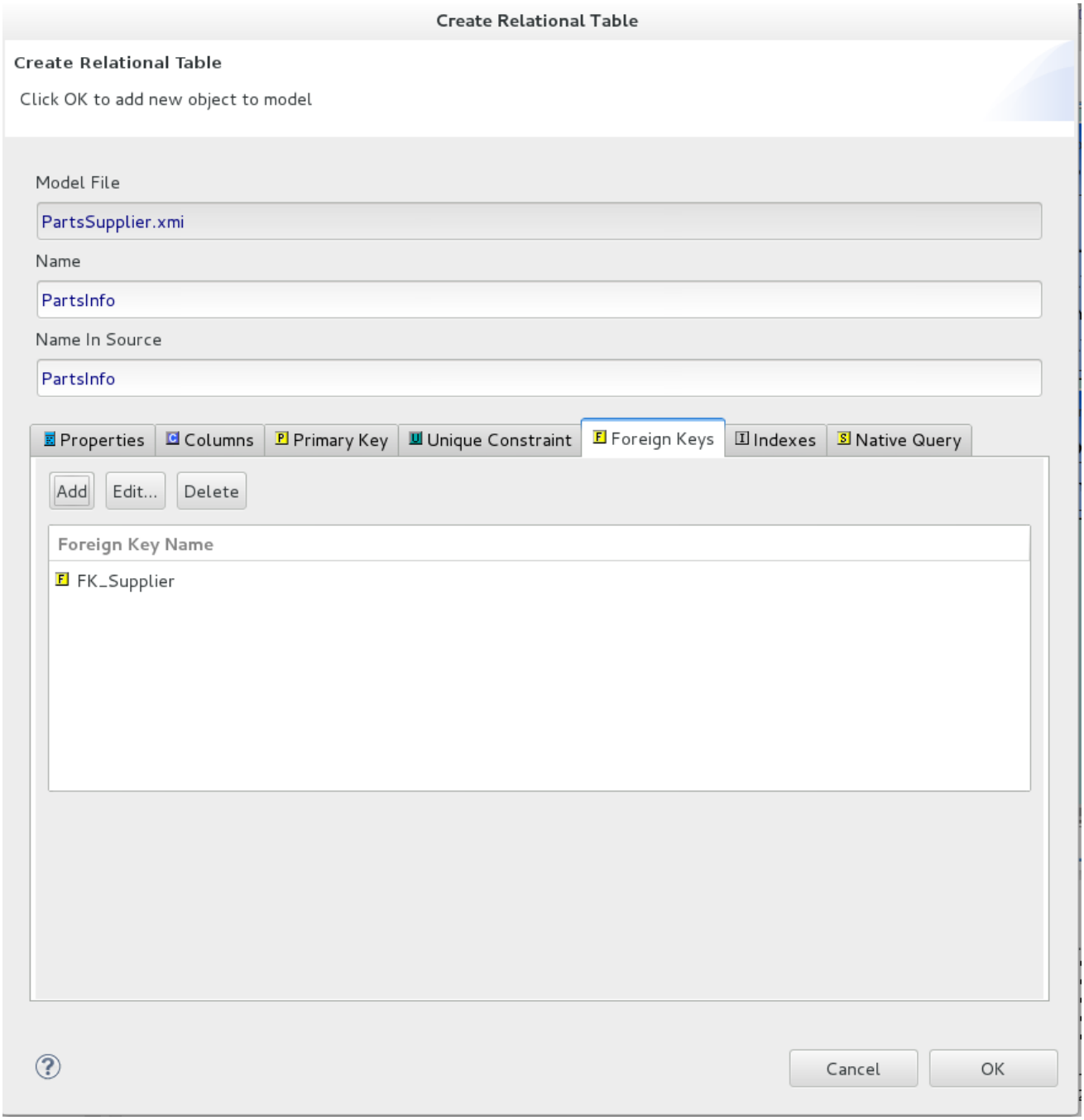


Figure 12.9. Foreign Keys Tab

To create a new **`Foreign Key`**, select the **Add** button and enter/select the properties, key references in the tables shown below. Note that the Select Primary Key or Unique Constraint table will display any PK/UC existing in the selected relational model. If no tables in that model contain a PK or UC, then the table will be empty.

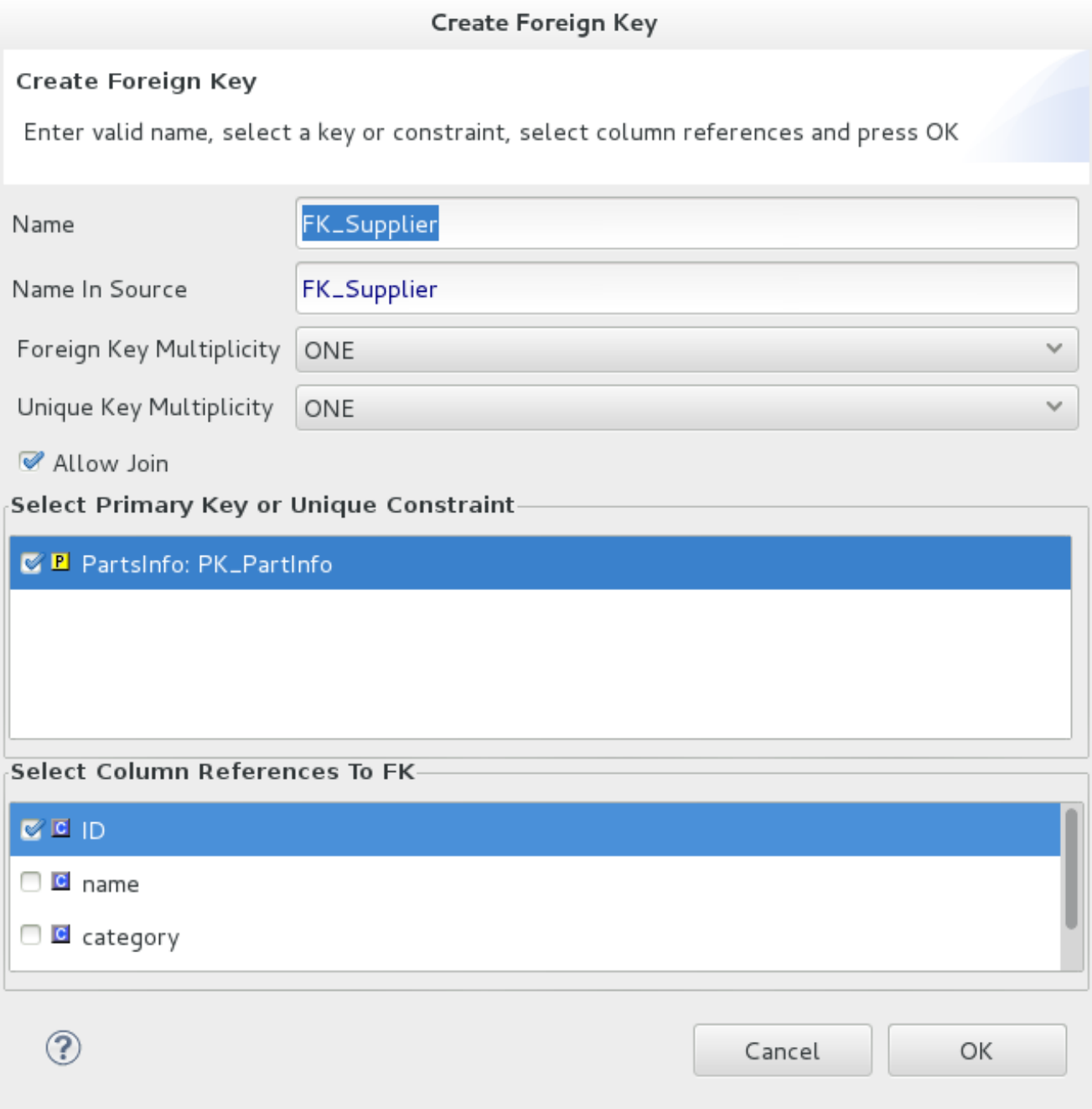


Figure 12.10. Create Foreign Key Dialog

12.2. RELATIONAL VIEW MODELING

12.2.1. Create Materialized Views

For any relational view table you can enable its materialized view option by setting the Materialized property to TRUE and setting the Materialized Table reference, as shown in the figure below. Note that you are required to have already created your relational tables.

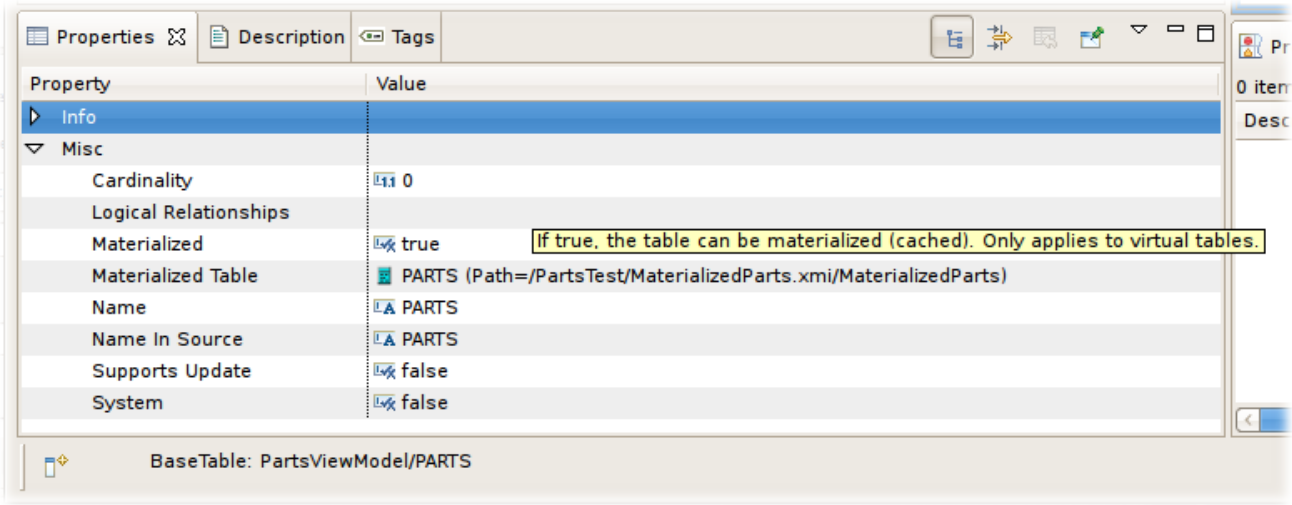


Figure 12.11. Materialized Table Properties

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

To create materialized views:

1. The materialization process is undertaken one view at a time. Select the view to be materialized then right-click the view tables in the **`Model Explorer View`** and select the **`Modeling > Create Materialized Views`** action. (Note that you should select two or more tables otherwise you will see a different context menu and wizard.)

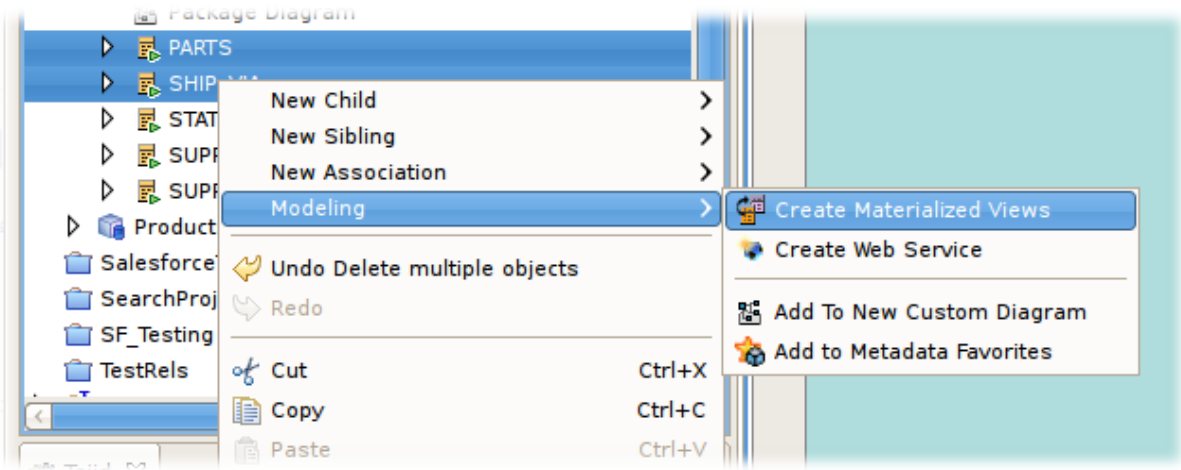


Figure 12.12.

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

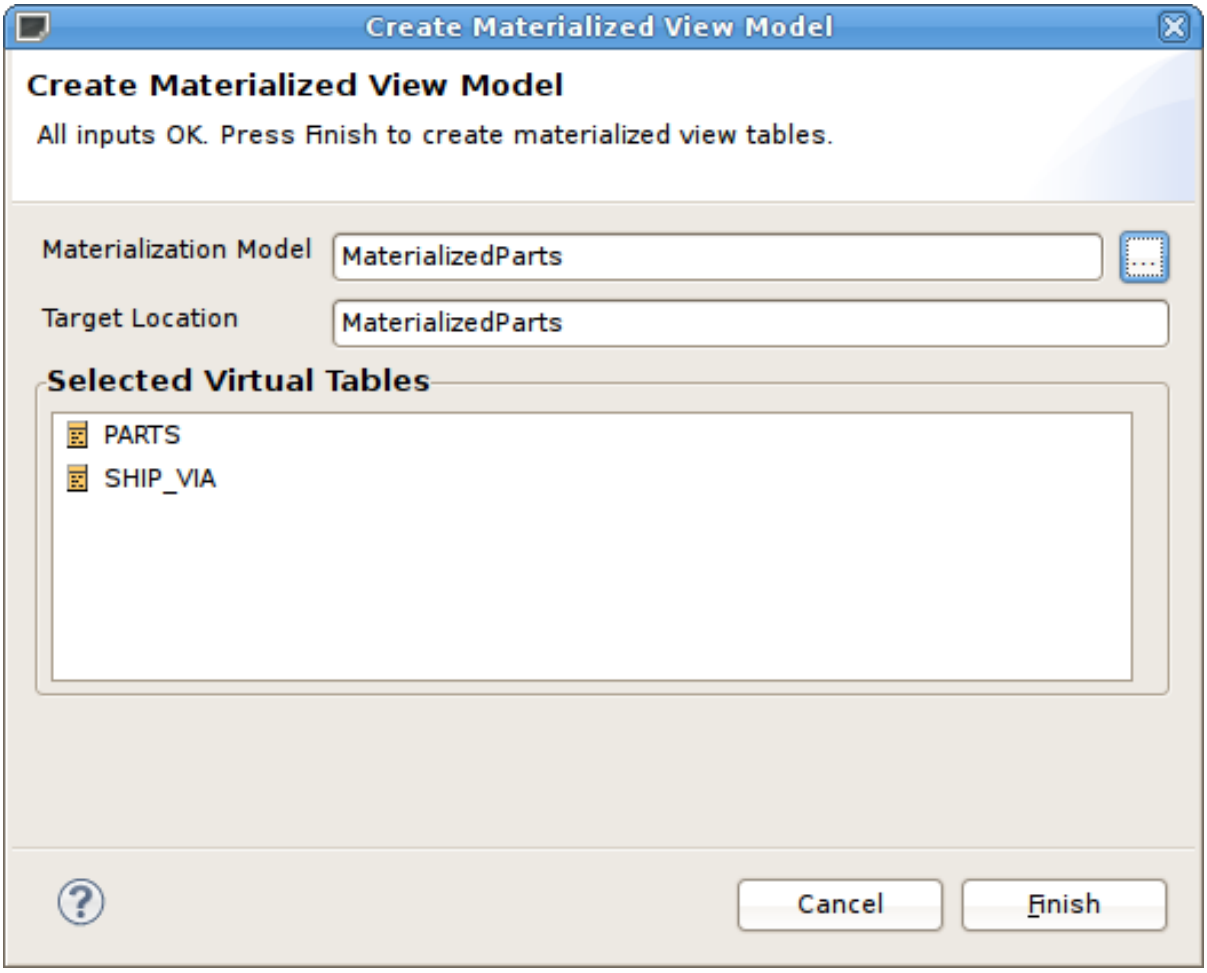


Figure 12.13.

3. 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.

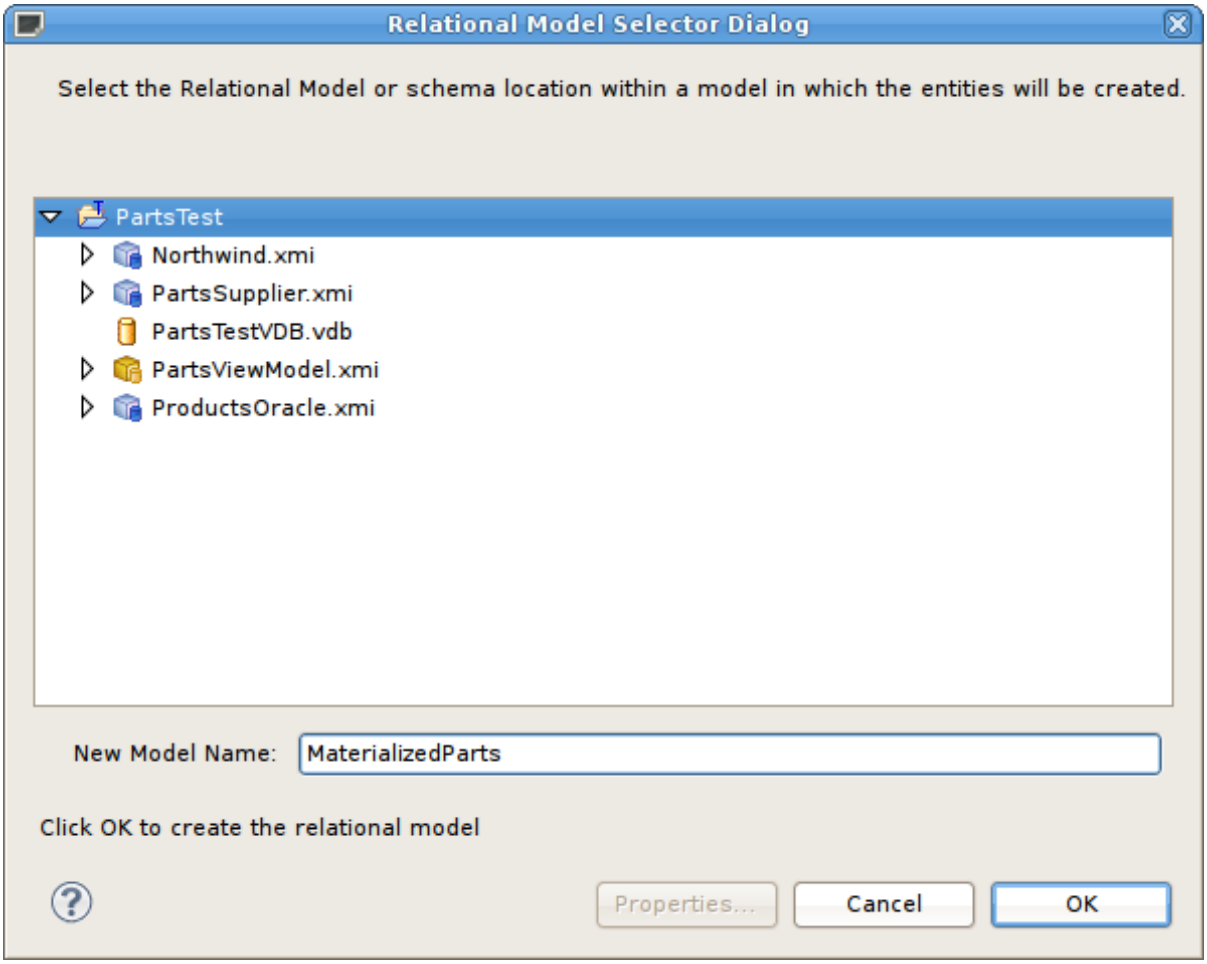


Figure 12.14.

4. 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.

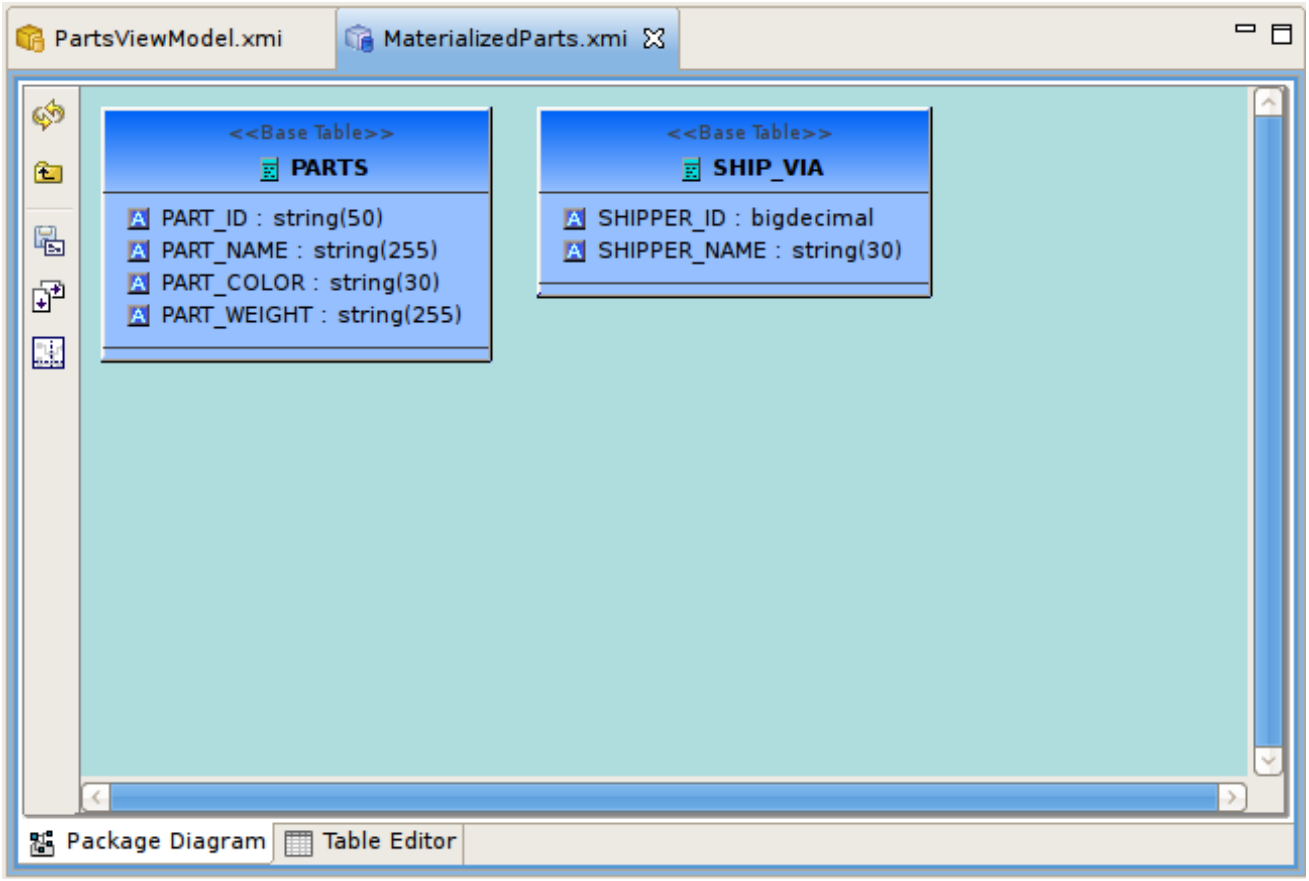


Figure 12.15. Materialized Table Properties

12.2.2. Red Hat JBoss Data Grid Materialization

JBoss Data Grid caches running in client/server mode can use a Red Hat JBoss Data Virtualization-specific JCA connector for accessing as a data source, which is deployed into WildFly 10.0.0 during installation. This connector can be configured to support the accessing of a remote JBoss Data Grid cache using the the Hot Rod client.

12.2.2.1. JBoss Data Grid 6 Materialization

Each JBoss Data Grid cache that has a referenced pojo object by doing a "get(key)" on the cache, for which you want to access, will require a different configured resource-adapter.

There are two options for how the JDG schema can be configured in the connector; protobuf annotations or protobuf (.proto) file with marshaller(s). These are the requirements:

- Minimum, JDG 6.2 - this requires you provide a protobuf definition file and pojo marshaller(s) for the pojo to configure the JDG schema
- Minimum, JDG 6.6 - this can be used when the pojo has defined protobuf annotations which are used to configure the JDG schema.

The following property is required as it provides the mapping to the JDG cache and pojo that will be accessed.

Table 12.1. Mapping

Property Name	Property Template	Description
CacheTypeMap	cacheName:className[;pkFieldName[:cacheKeyJavaType]]	For the indicated cacheName, map the root Java Object (pojo) class name. Optionally, but required for updates, identify which class attribute is the primary key to the cache. Optionally, identify primary key java type when different than class attribute type.

The following properties are required when the protobuf definition file (.proto) and the pojo marshaller(s) are being used to configure the JDG schema:

Table 12.2. Base Execution Properties

Property Name	Required?	Property Template	Description
ProtobufDefinitionFile	Yes	Path to the Google Protobuf file that's packaged in a jar (ex: /quickstart/addressbook.proto)	NA
MessageMarshallers	Yes	marshaller \ [,marshaller,..\]	Contains Class names mapped its respective message marshaller, (class:marshaller,\ [class:marshaller,..\]), that are to be registered for serialization
MessageDescriptor	Yes	NA	Message descriptor class name for the root object in cache

The pojo class is the object that is used to store the data in the cache.

If the pojo is to be used to define the schema, then should use the protobuf annotations.

If the protobuf definition and mashaller(s) are to be used, then these should also be packaged in the jar (or a separate jar that is included in the classpath).

The class should be packaged into a jar so that it can be deployed as a module.

```
public class Person {

   @ProtoField(number = 2, required = true)
   public String name;
   @ProtoField(number = 1, required = true)
   public int id;
   @ProtoField(number = 3)
   public String email;
   private List<PhoneNumber> phones;

   public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public List<PhoneNumber> getPhones() {
        return phones;
    }

    public void setPhones(List<PhoneNumber> phones) {
        this.phones = phones;
    }
    }
```

To configure the resource adapter to use the pojo, deploy the pojo jar as a module in the JBOSS EAP server.

One of the following properties is required for defining how the RemoteCacheManager will be created/accessed:

Table 12.3. Configuration

Property Name	Required?	Property Template	Description
CacheTypeMap	Yes	cacheName:className[;pkFieldName[:cacheKeyJavaType]]	For the indicated cache, map the root Java Object class name. Optionally, but required for updates, identify which class attribute is the primary key to the cache. Identify primary key java type when different than class attribute type

Property Name	Required?	Property Template	Description
ProtobinFile	Yes	NA	Path to the Google Protobin file that's packaged in a jar (ex: /quickstart/addressbook.protobin)
MessageMarshallers	Yes	marshaller [,marshaller,..]	Contains Class names mapped its respective message marshaller, (class:marshaller, [class:marshaller,..]), that are to be registered for serialization
MessageDescriptor	Yes	NA	Message descriptor class name for the root object in cache
module	No	NA	Specify the JBoss EAP module that contains the cache classes that need to be loaded
CacheJndiName	No	NA	JNDI name to find the CacheContainer
RemoteServerList	No	host:port[;host:port….]	Specify the host and ports that will be clustered together to access the caches
HotRodClientPropertiesFile	No	NA	The HotRod properties file for configuring a connection to a remote cache

The following property should be defined when using protobuf definition file and marshallers:

Table 12.4. Properties

Property Name	Required?	Property Template	Description
module	No	NA	Specify the WildFly module that contains the cache classes that need to be loaded

The following are the additional properties that need to be configured if using the Remote Cache for external materialization:

Table 12.5. Mapping

Property Name	Required?	Description
StagingCacheName	Yes	Cache name for the staging cache used in materialization .
AliasCacheName	Yes	Cache name for the alias cache used in tracking aliasing of the caches used in materialization. This cache can be shared with other configured materializations. .

There are many ways to create the data source, using the CLI, AdminShell, admin-console and so forth. The first example is an XML snippet of a resource-adapter that is used to connect to the JDG remote-query quick start:me Sample Resource Adapter defining Protobuf Definition and Marshaller:

```
<resource-adapter id="infinispanRemQS">
  <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.hotrod"/>
  <connection-definitions>
      <connection-definition class-
name="org.teiid.resource.adapter.infinispan.hotrod.InfinispanManagedConnec
tionFactory" jndi-name="java:/infinispanRemote" enabled="true" use-java-
context="true" pool-name="infinispanDS">
        <config-property name="CacheTypeMap">

addressbook:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person;i
d
        </config-property>
        <config-property name="ProtobufDefinitionFile">
          /quickstart/addressbook.proto
        </config-property>
        <config-property name="MessageDescriptor">
          quickstart.Person
        </config-property>
        <config-property name="Module">
          com.client.quickstart.pojos
        </config-property>
        <config-property name="MessageMarshallers">

org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person:org.jboss.as.
quickstarts.datagrid.hotrod.query.marshallers.PersonMarshaller,org.jboss.a
s.quickstarts.datagrid.hotrod.query.domain.PhoneNumber:org.jboss.as.quicks
tarts.datagrid.hotrod.query.marshallers.PhoneNumberMarshaller,org.jboss.as
.quickstarts.datagrid.hotrod.query.domain.PhoneType:org.jboss.as.quickstar
ts.datagrid.hotrod.query.marshallers.PhoneTypeMarshaller
        </config-property>
        <config-property name="RemoteServerList">
          127.0.0.1:11322
        </config-property>
      </connection-definition>
  </connection-definitions>
</resource-adapter>
```

Here is a sample Resource Adapter using Pojo with annotations:

```
                <resource-adapter id="infinispanRemQSDSL">
                     <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.dsl"/>
                     <connection-definitions>
                         <connection-definition class-
name="org.teiid.resource.adapter.infinispan.dsl.InfinispanManagedConnectio
nFactory" jndi-name="java:/infinispanRemoteDSL" enabled="true" use-java-
context="true" pool-name="infinispanRemoteDSL">
                             <config-property name="RemoteServerList">
                                 127.0.0.1:11322
                             </config-property>
                             <config-property name="CacheTypeMap">

addressbook_indexed:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.
Person;id
                             </config-property>
                         </connection-definition>
                     </connection-definitions>
                 </resource-adapter>
```

Here is a sample Resource Adapter for external materialization:

```
                 <resource-adapter id="infinispanRemQSDSL">
                     <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.hotrod"/>
                     <connection-definitions>
                         <connection-definition class-
name="org.teiid.resource.adapter.infinispan.hotrod.InfinispanManagedConnec
tionFactory" jndi-name="java:/infinispanRemoteDSL" enabled="true" use-
java-context="true" pool-name="infinispanRemoteDSL">
                             <config-property name="CacheTypeMap">

addressbook_indexed:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.
Person;id
                             </config-property>
                             <config-property name="StagingCacheName">
                                 addressbook_indexed_mat
                             </config-property>
                             <config-property name="AliasCacheName">
                                 aliasCache
                             </config-property>
                             <config-property name="Module">
                                 com.client.quickstart.addressbook.pojos
                             </config-property>
                             <config-property name="RemoteServerList">
                                 127.0.0.1:11322
                             </config-property>
                         </connection-definition>
                     </connection-definitions>
                 </resource-adapter>
```

The HotRod Translator, known by the type ispn-hotrod, can read the java objects from a remote JDG Cache via the Hot Rod client using the Google Protobuf for serialization. This will enable Teiid to query

the remote cache using JDG DSL. This translator extends the Object Translator and uses it for the core processing of reading and writing objects. It can do the following:

- Retrieve objects from a cache and transform into rows and columns.
- Supports performing writes to the cache
- Use for external materialization to improve query performance

The following are the connector capabilities:

- Compare Criteria - EQ
- Compare Criteria Ordered - LT, GT, LE, GE - support for SupportsComapareCriteriaOrdered will be controlled by the version of JDG being accessed. Any JDG version 6.5 and prior will have this set to false due to an issue with JDG.
- And/Or Criteria
- In Criteria
- Like Criteria
- Order By
- INSERT, UPDATE, DELETE (non-transactional)

The following will not be pushed down to JDG for processing, but will be done within Red Hat JBoss Data Virtualization:

- Not (NE)
- IsNull

These are its limitations:

- support for 'Not' has been disabled
- boolean data type: JDG will throw an exception if no value is specified on the insert or when no default value is defined in the protobuf definition file.
- char data type: is not a supported type in theProtobuf data types (https://developers.google.com/protocol-buffers/docs/proto#scalar). Would either have to handle conversion in the protobuf marshaller or create a Teiid view with the data type as char.
- 1-to-Many, currently only supports Collection or Array, not Maps
- Write transactions not supported by JDG when using Hot Rod client

There are several options to defining the metadata representing your object in the cache. Red Hat recommmends that you use the Teiid Connection Importer in Teiid Designer to create the physical source model based on your object cache. The table columns will be created from the google protobuf definition, that corresponds to a registered class.

Use Teiid Designer to manually create the physical source model based on your object cache using the below Definition Requirements:

```
<model name="People" type="Physical">
```

```
    <property name="importer.useFullSchemaName" value="false"/>

    <source name="infinispan-hotrod-connector" translator-name="ispn-
hotrod" connection-jndi-name="java:/infinispanRemoteDSL" />
  </model>
```

The metadata will be resolved by reverse engineering the defined object in the cache. This can be useful when using the Teiid Designer Teiid Connection Importer for building the physical source model(s).

You can also define the metadata using DDL. See the Object Translator for an example.

See the Object Translator Metadata section for base definition requirements.

Columns will be identified as SEARCHABLE if either the protobuf definition for a column indicates its indexed or the pojo class has the attribute/method annotated.

A 1-to-* relationship class must have a foreign key to map to the root class/table, where the name in source for the foreign key is the name of the root class method to access those child objects. Note, this is the class method, not a reference in the google protobuf definition.

A container/child class will have attributes where the NIS contain a period. Example: phone.number. This is because this maps to to google protobuf definition and what is expected to be used in the DSL query.

This translator supports using the cache for external materialization. However, there are specific configuration changes that are required at the [Infinispan-HotRod resource-adapter] and at the translator.

External materialization is enabled by the use of native queries in the BEFORE_LOAD_SCRIPT and AFTER_LOAD_SCRIPT. A translator override will need to be set to enable native queries: SupportsNativeQueries=true

The following materialization properties must be defined:

Table 12.6. Mapping

Script	Native query	Description
teiid_rel:MATVIEW_BEFORE_LOAD_SCRIPT	truncate cache	To truncate the cache identified as the staging cache.
teiid_rel:MATVIEW_AFTER_LOAD_SCRIPT	swap cache names	To swap the aliases for the caches, so that the primary cache points to the recently loaded cache.

This is how you define the load scripts in DDL:

```
..
"teiid_rel:MATVIEW_BEFORE_LOAD_SCRIPT" 'execute
StockMatCache.native(''truncate cache'');',
"teiid_rel:MATVIEW_LOAD_SCRIPT" 'insert into StockMatCache.Stock
(productId, symbol, price, companyName) SELECT  A.ID, S.symbol, S.price,
A.COMPANY_NAME FROM Stocks.StockPrices AS S, Accounts.PRODUCT AS A WHERE
S.symbol = A.SYMBOL',
"teiid_rel:MATVIEW_AFTER_LOAD_SCRIPT"  'execute
StockMatCache.native(''swap cache names'');',
```

Native queries are used to simulate how its done using RDBMS and renaming tables, because JDG doesn't currently support renaming a cache. So the native queries will trigger the clearing of the "staging" cache, and the swapping of the cache aliases.

Additionally, the execution of native queries is done through the support of direct query procedures. The procedure to be executed is called native.

WARNING

This feature is turned off by default because of the security risk this exposes to execute any command against the source. To enable this feature, override the execution property [Override Execution Properties] called SupportsDirectQueryProcedure to true.

If you manually model the cache table in Teiid Designer, then you will need to add the property extension for defining the property "primary_table":

```
SET NAMESPACE 'http://www.teiid.org/translator/object/2016' AS n0;

CREATE FOREIGN TABLE Trade (
         ....
 CONSTRAINT PK_TRADEID PRIMARY KEY(tradeId)
) OPTIONS (UPDATABLE TRUE);

CREATE FOREIGN TABLE ST_Trade (
        ....
) OPTIONS (NAMEINSOURCE 'Trade', UPDATABLE TRUE, "n0:primary_table"
'ObjectSchema.Trade');
```

NOTE

If you do manual modeling, your column name must match that of the POJO.

See the JDG HotRod Data Sources resource adapter for this translator. It can be configured to look up the cache container via JNDI, server list, or hot rod properties.

IMPORTANT

It is highly recommended that you use the Teiid Designer tooling wizards (such as the Teiid Connection Importer and/or Materialize) to create the table and column names. If, instead, you use a VDB XML form which points to a JDG resource adatper, do not define the DDL metadata but, rather, let the translator expose the metadata.

The reason for this is that the configuration of the JDG schema is case sensitive. The 'name' used in the query to JDG must match the name of the property (methodName) that was used.

1. Using annotations, the pojo method names are used to configure the JDG schema, by passing the Class to the configuration builder.
2. Using protobuf/mashallers, the JDG schema is configured by using the protobuf file.

Both configuration methods enable the JDG cache to provide a descriptor, for which the names of the columns (and nameInSources when using complex objects) are derived. It is these names that must be used in the model.

12.2.2.2. JBoss Data Grid 7 Materialization

You can use Red Hat JBoss Data Grid (JDG) 7 as a target data source for external materialization to improve query result performance.

Prerequisites

- JBoss Data Virtualization 6.4 and above
- JBoss Data Grid 7.1.2 and above
- Configure the primary and staging cache and the **`teiid-alias-naming-cache`** cache in JDG. For details, see the *External Materialization and Red Hat JBoss Data Grid* section in the *Development Guide Volume 5: Caching Guide* for JBoss Data Virtualization 6.
- Configure the translator and the **`jdg`** module. For details, see the *Configure Red Hat JBoss Data Grid Connectors* section in the *Installation Guide* for JBoss Data Virtualization 6.

IMPORTANT

Due to a bug, if the server does not contain the **`infinispan-hotrod-translator`** extension or Model Extension Registry is not synchronized with the server, an attempt to create the JDG source model fails and the **`NullPointerException`** message appears in the log. To work around this issue, make sure that the **`Model Extension`** view includes **`infinispan-hotrod-translator`**. If the view does not include **`infinispan-hotrod-translator`**, ensure that the server contains it and refresh the server in the **`Teiid Designer Connection`** view.

Procedure 12.1. Configuring JBoss Data Grid 7 Materialization

1. Create a JDG source model. In the Teiid Designer, right-click on the view to materialize and select **`Modeling`** > **`Materialize`**.

Make sure, that the view has a primary key defined because the Red Hat JBoss Data Grid source table requires the key for updates. Otherwise, manually create a primary key on each of the new JDG source tables.

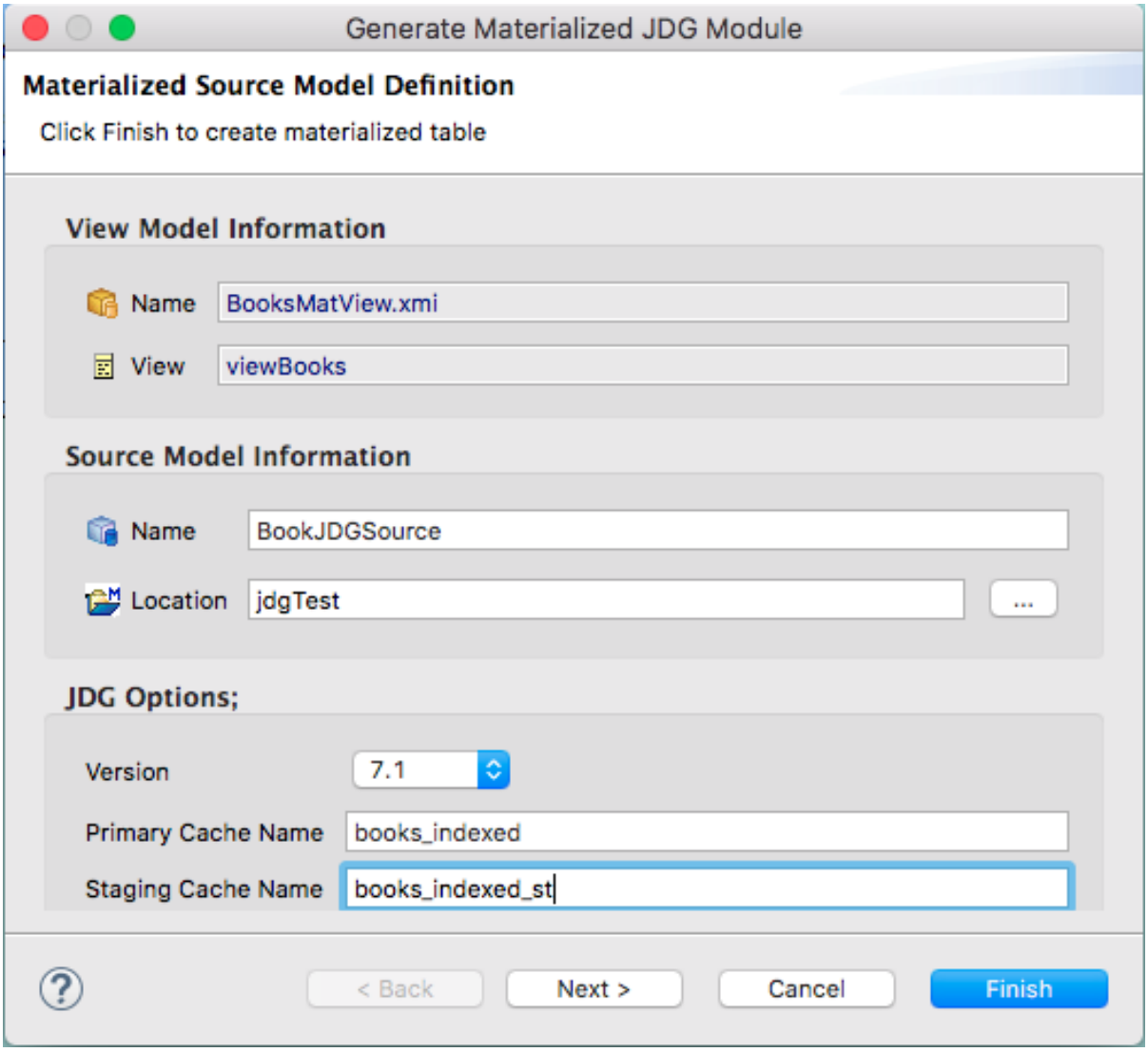


Figure 12.16. Generate Materialized JDG Module

a. Enter the primary and staging cache names.

b. Optionally, change the **JDG Source** model name and the directory in which the model is saved. Note that Red Hat JBoss Data Grid restricts the name of the source model because the **`protobuf`** code is based on the Java package naming constraints. The model name becomes the package name in the **`.proto`** file. This is due to a limitation in the way that **`protobuf`** is defined. Because Red Hat JBoss Data Grid uses Java, the package name must follow the Java package naming standards. Dashes, for instance, are not allowed.

c. Click **`Finish`**.

2. Create a new connection profile.

a. In the **Data Source Explorer** view click **New connection profile**.

b. Choose JDG 7.x, set name, and click **`next`**.

c. Set properties. Specify the Java Naming and Directory Interface (JNDI) Name, which is the name of the JDG Source model.

For details about other properties to set, see the *Red Hat JBoss Data Grid 7.x Configuration* section in the *Administration and Configuration Guide* for JBoss Data Virtualization 6.

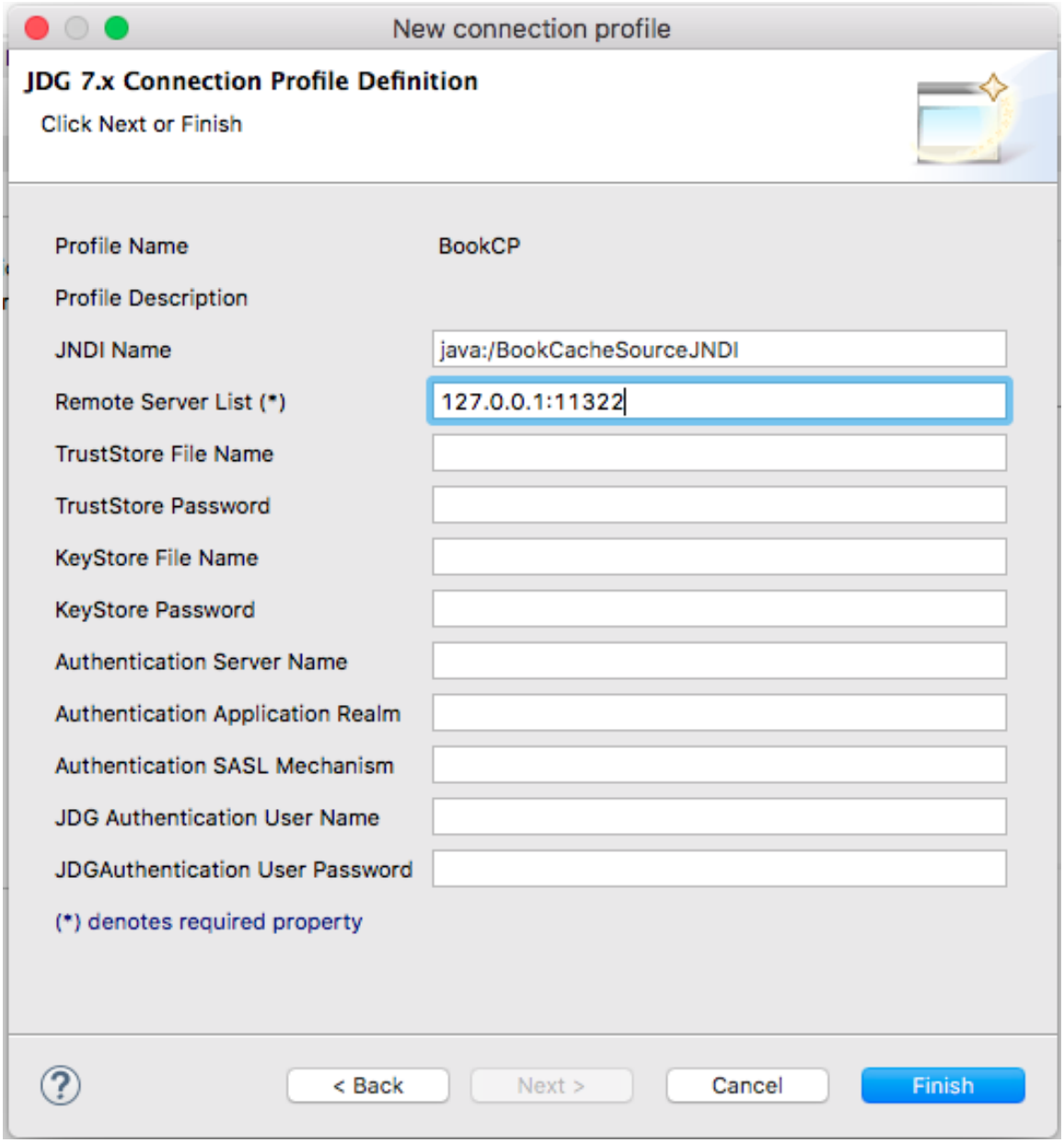


Figure 12.17. New connection profile

d. Click **`Finish`**.

3. Add the connection profile to the created JDG Source model. Right-click the **`JDG Source Model`** and select **`Modeling`** > **`Set connection profile`**.

4. Create Data Source for the JDG Source model. Right-click **`JDG Source Model`** and select **`Modeling`** > **`Create Data Source`**.

5. To control the materialization process, update the materialized view extension properties on the above selected view:

   - MATVIEW_TTL - to set the refresh rate, in milliseconds

6. If you use the materialization management status table, set the following extension properties:

   - ALLOW_MATVIEW_MANAGEMENT = true

- MATVIEW_STATUS_TABLE = {status table name}

7. Create the Virtual Database (VDB), using the models needed for materialization.
   a. For the JDG Source model, be sure the JNDI is mapped to the JDG data source.
   b. Enable native queries. To do this, create a translator override for the **`infinispan-hotrod`** translator. Click the **supportsDirectQueryProcedure** property and set the value to **true**.
8. Create a VDB XML file. Right-click the VDB and select **`Modeling`** > **`Generate VDB XML`**.

IMPORTANT

Due to a bug, the VDB must be deployed to the server as a VDB XML file.

9. Deploy the VDB XML file. Right-click the VDB XML file and select **`Modeling`** > **`Deploy`**

12.2.3. Create Relational View Table Wizard

You can create relational view tables by right-clicking on view model and select **`New Child > Table...`** action. This action also includes creating view tables with its target columns and SQL transformation.

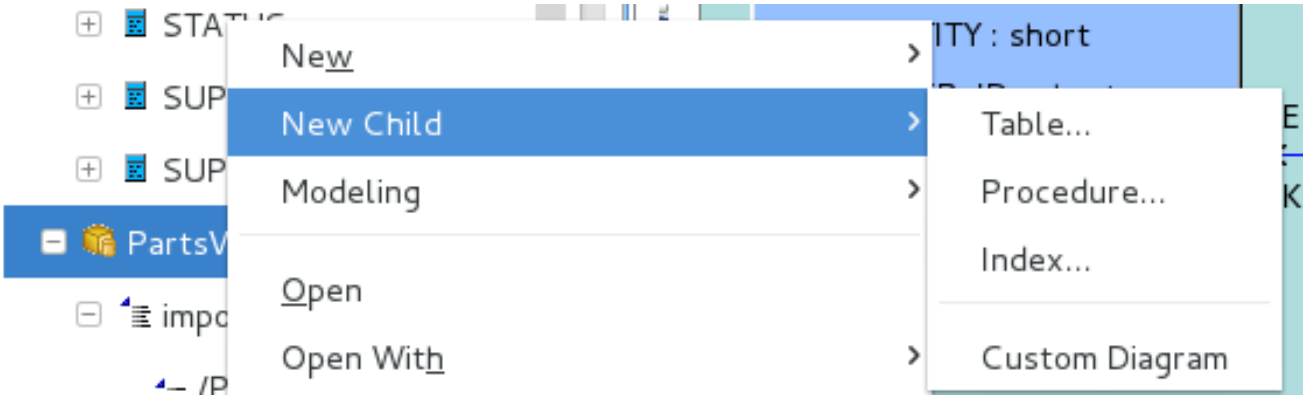


Figure 12.18. New View Table Wizard Action

Running the action will display the **`Create Virtual Table Options`** wizard. You have two options how to create a table:

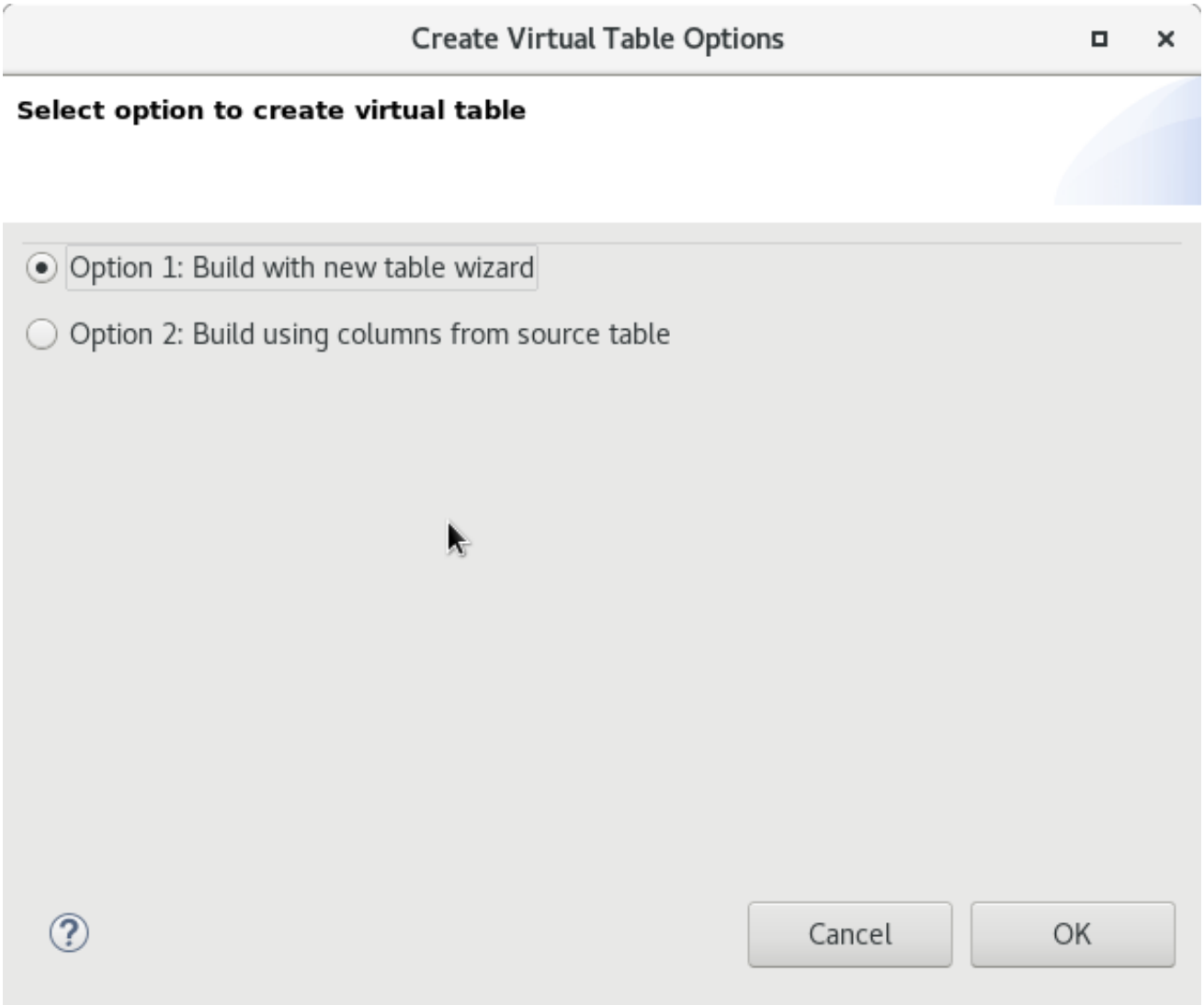


Figure 12.19. Create Virtual Table Options wizard

12.2.3.1. Build with new table wizard

Select the option 1 and click OK. Next will be display **`Create Relational View Table`** wizard. The wizard page contains tabbed panels representing the various properties and components that make up the possible definition of a relational view table. Enter your table name, define the desired columns and specify the SQL transformation, then click **`OK`**.

This wizard is designed to provide feedback as to the completeness of the relational view table information, as well as the validation state of the table and its components. Note that although errors or warnings may be displayed during editing, the wizard is designed to allow finishing even if the table definition is incomplete.

The first tab labeled **`Properties`** contains the input for the simple table properties including name and supports update.

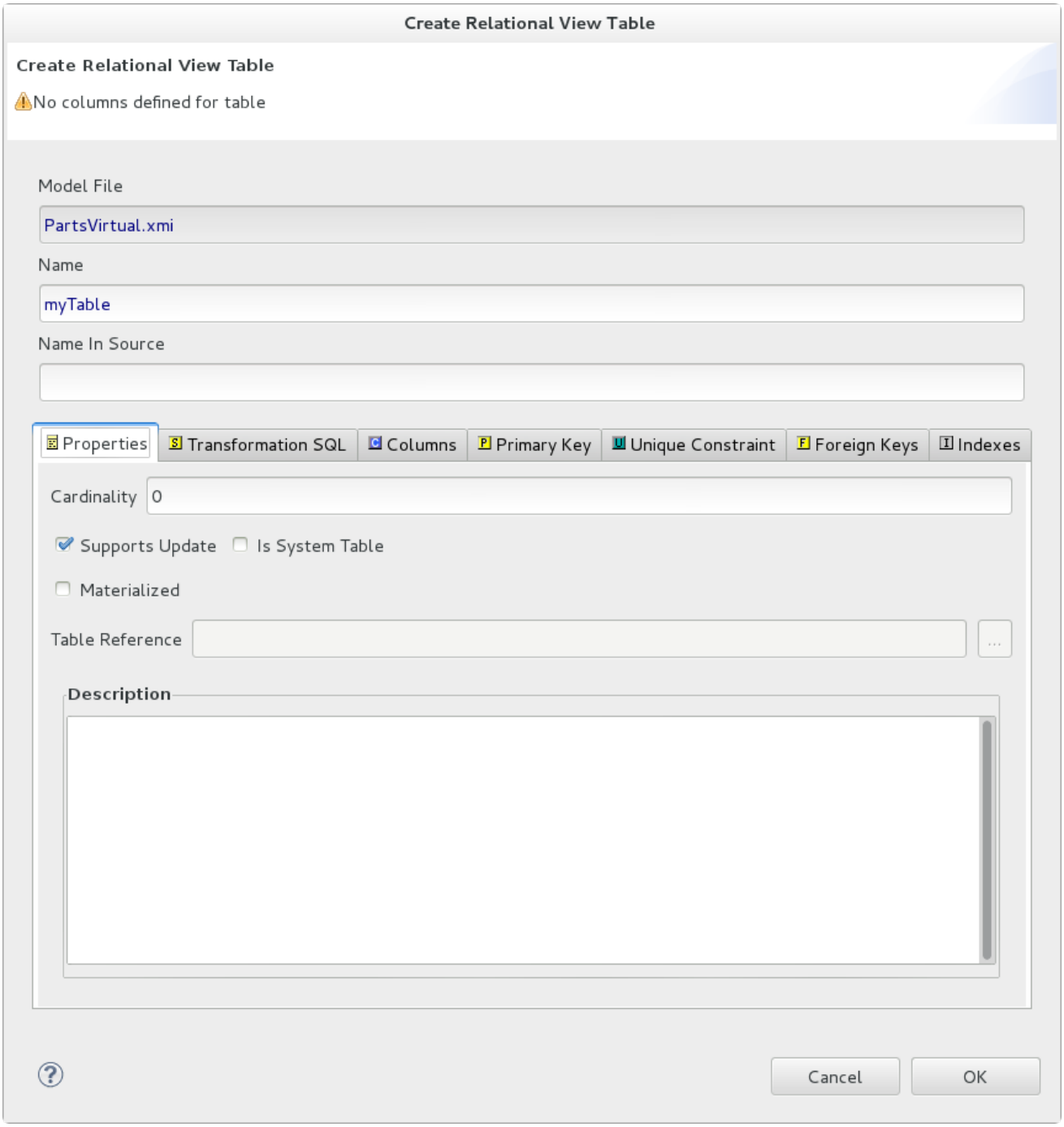


Figure 12.20. Properties Tab

The **`Columns`** tab allows creation and editing of basic relational columns. This includes adding, deleting or moving columns as well as changing the name, datatype and length properties.

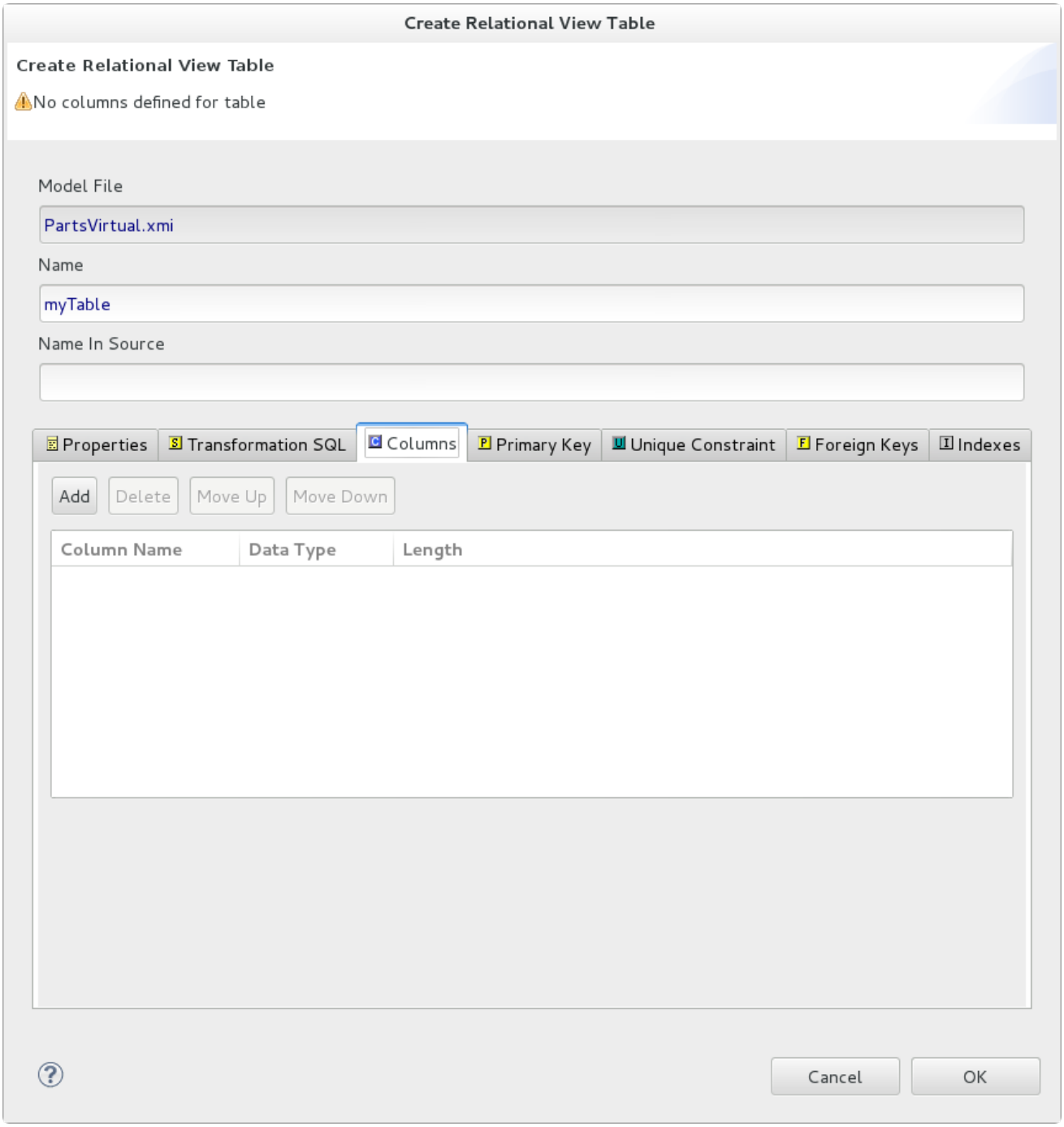


Figure 12.21. Columns Tab

The **`Transformation SQL`** tab allows editing of the SQL Transformation for the relational view. The desired SQL can be entered directly into the text area or a SQL Template may be selected by clicking the **`Select SQL Template`** button.

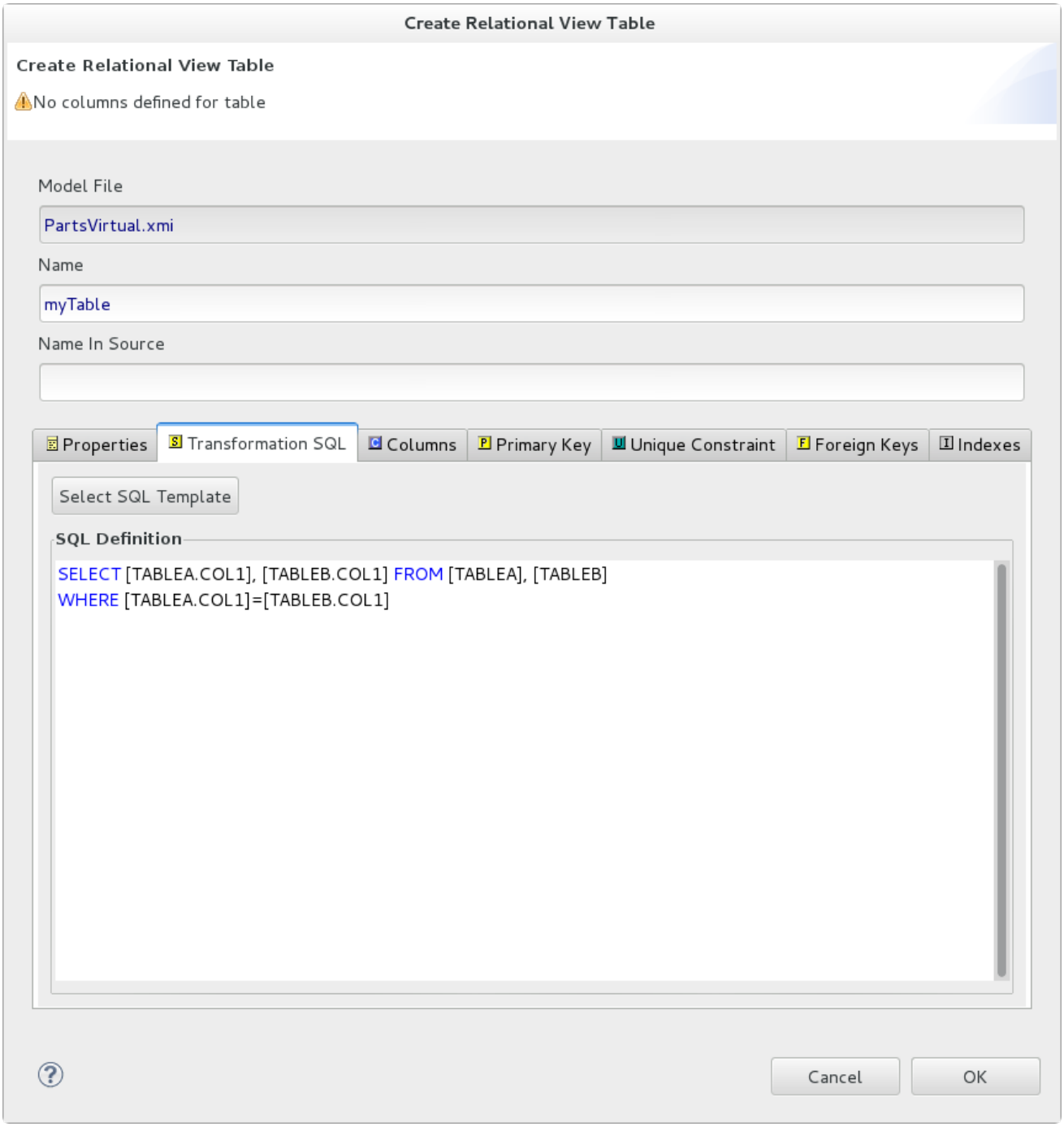


Figure 12.22. Transformation SQL Tab

If the **`Select SQL Template`** button is selected on the **`Transformation SQL`** tab, the **`Choose a SQL Template`** dialog is displayed.

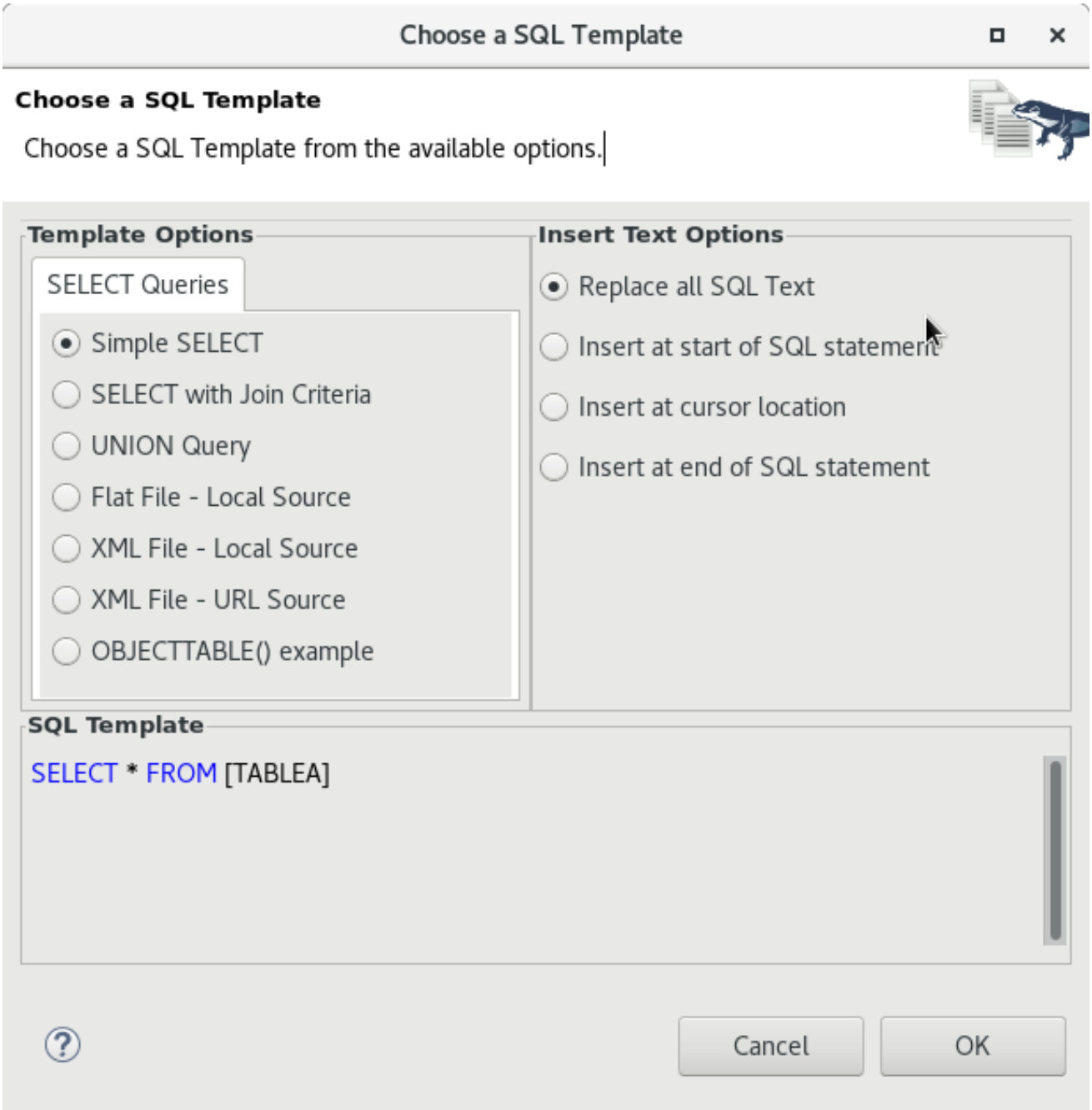


Figure 12.23. SQL Templates Dialog

A number of common SQL templates may be chosen using the dialog on the left side of wizard. Also you can choose Insert Text Options on the right side. You can create and update SQL Definition query for table. The **`SQL Template`** dialog can also be accessed in the **`Transformation Editor`**, from the right-click context menu and click to **`Apply Template...`**.

12.2.3.2. Build using columns from source model

Select the option 2 and click OK. Next will be display **`Select Source Table`** wizard. Select table from what you want to create virtual table and click OK. After that **`Select Columns for View`** dialog shows.

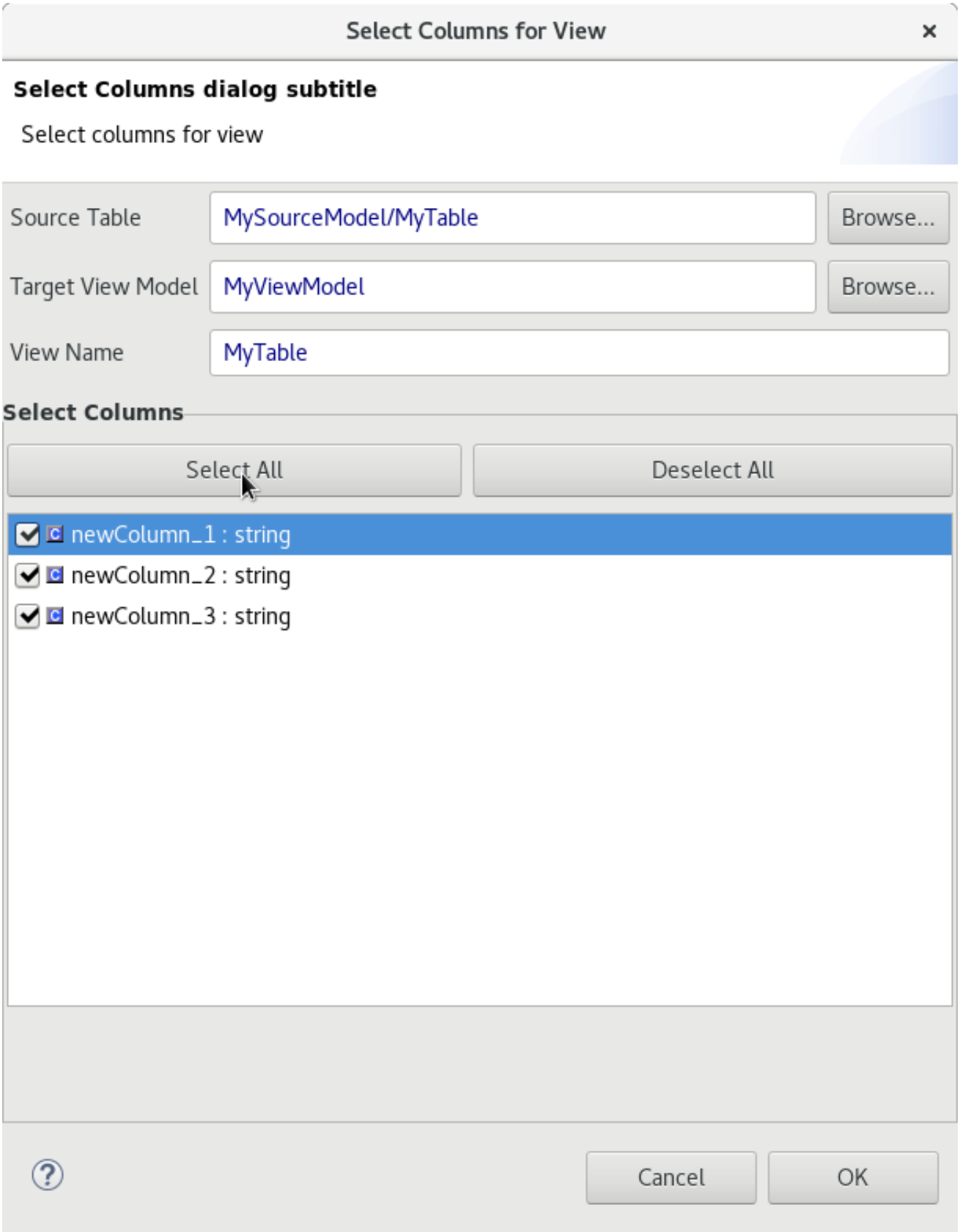


Figure 12.24. Select Columns for View dialog

This dialog has prefilled labels but you can change it.

- **`Source table`** is table from what you want to build your virtual table.
- **`Target View Model`** is your view model to where you want to build your virtual table.

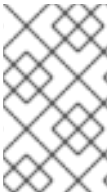
- **`View Name`** is name of your new virtual table.

Also you can pick which columns from source table will be in your virtual table. After that click OK and your virtual table will be created.

12.3. XML DOCUMENT MODELING

12.3.1. 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.



NOTE

You can create XML View Documents also by right-clicking on the XML model and select **`Child > XML document`**. Then XML Document builder page will be displayed which is described at the end of the section 12.3.1 (figure 12.23).

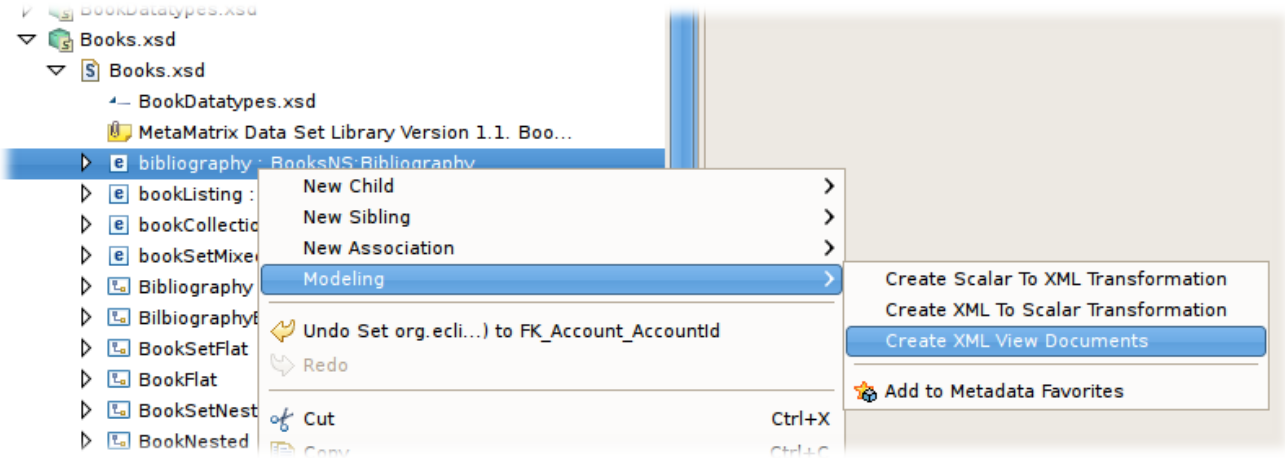


Figure 12.25. Create XML View Documents Action

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.

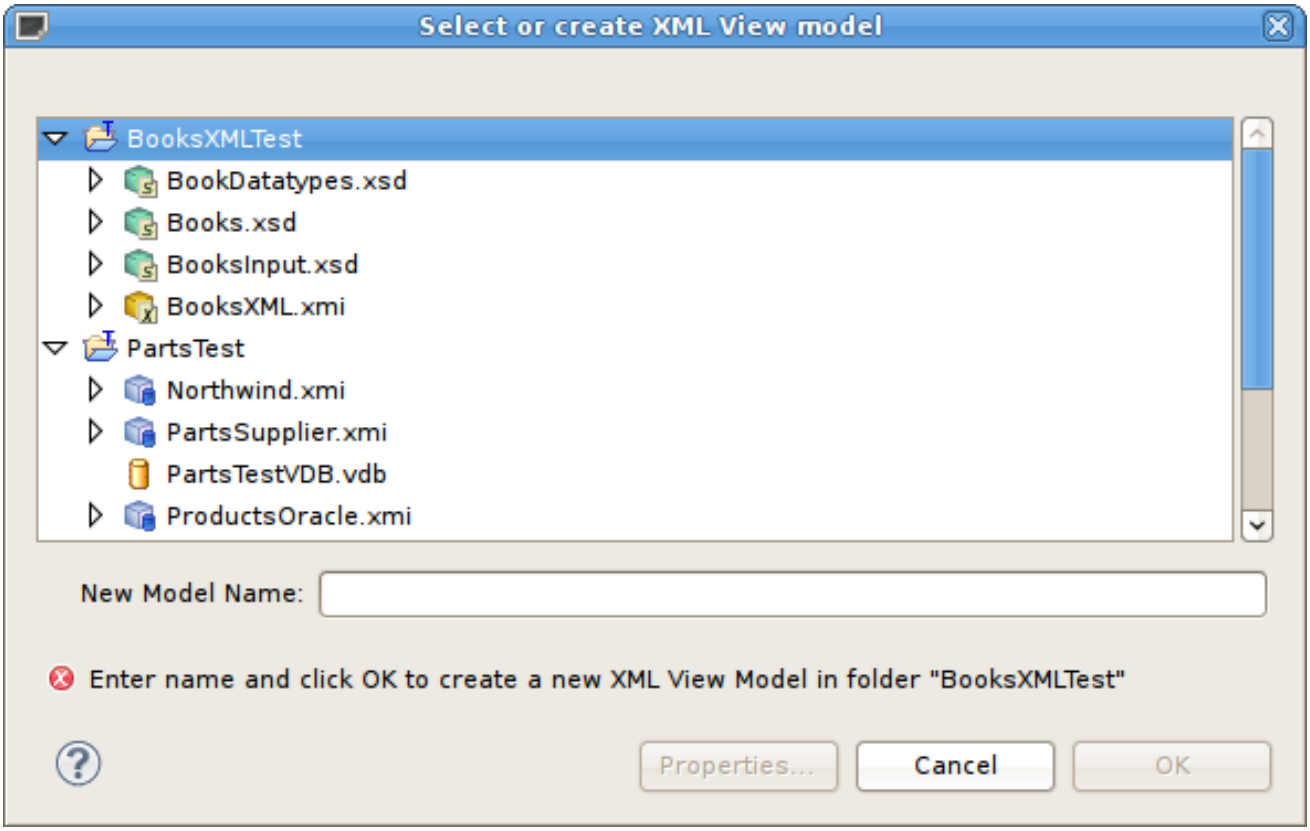


Figure 12.26. Select or Create XML View Model Dialog

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 the document view from schema section.

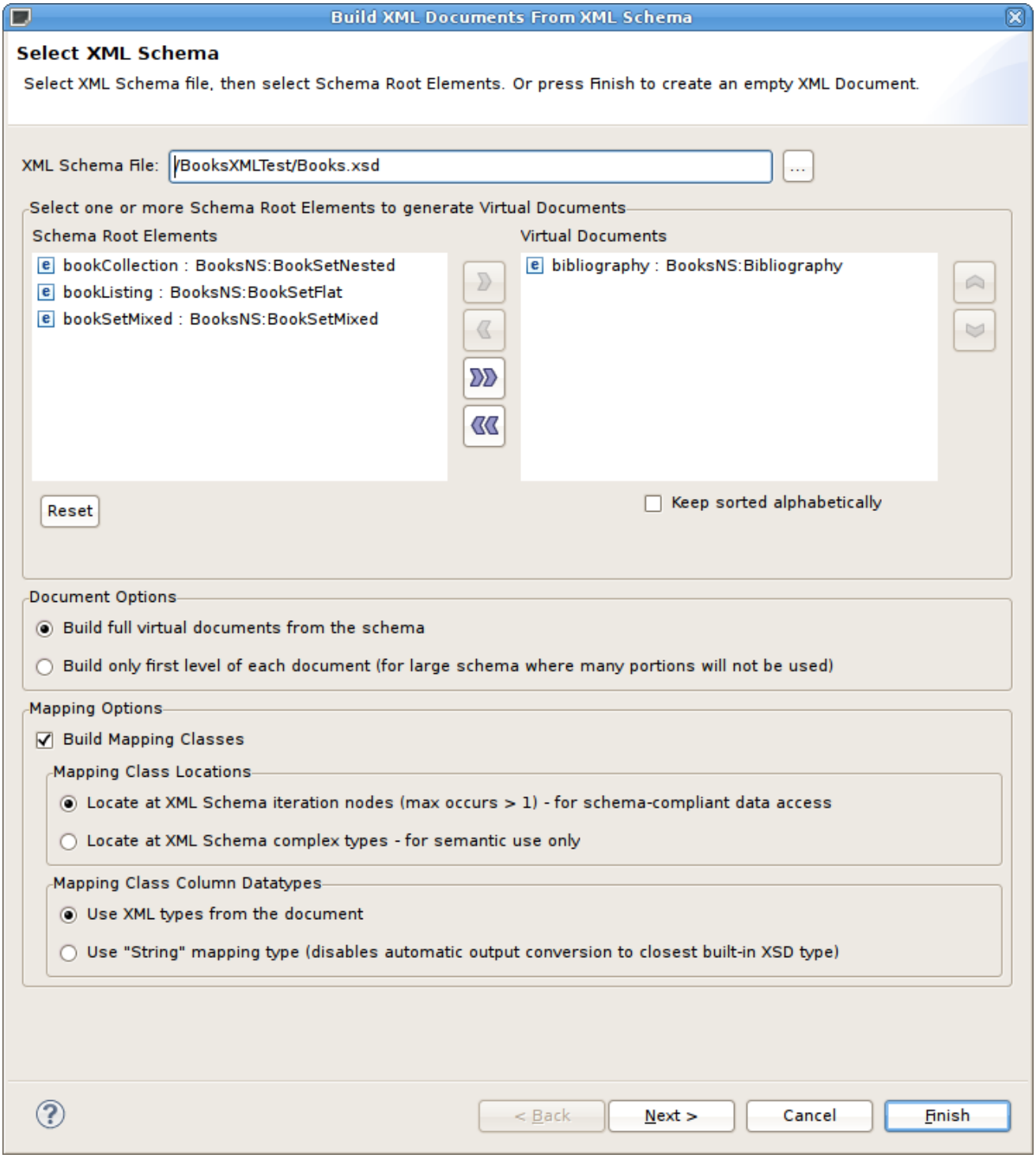


Figure 12.27. Build XML Documents From XML Schema Dialog

12.4. WEB SERVICES MODELING

12.4.1. 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:

1. Select any combination of relational models, tables and/or procedures in the 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.

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

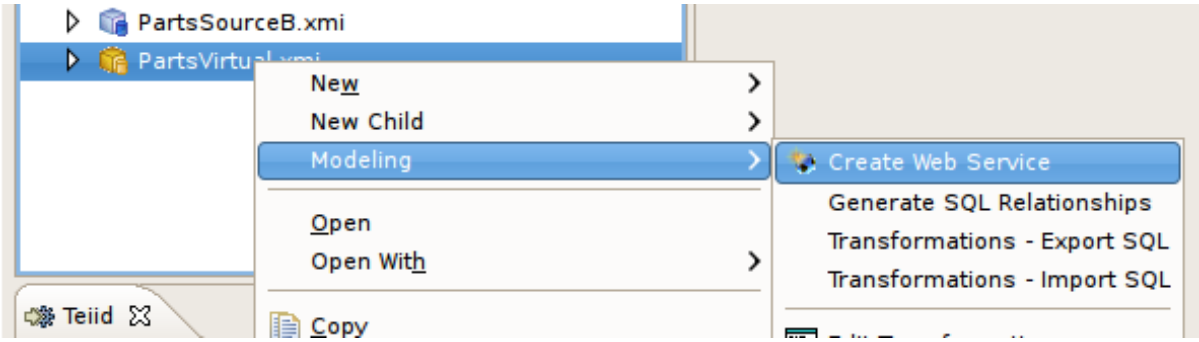


Figure 12.28. Create Web Service Action

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.

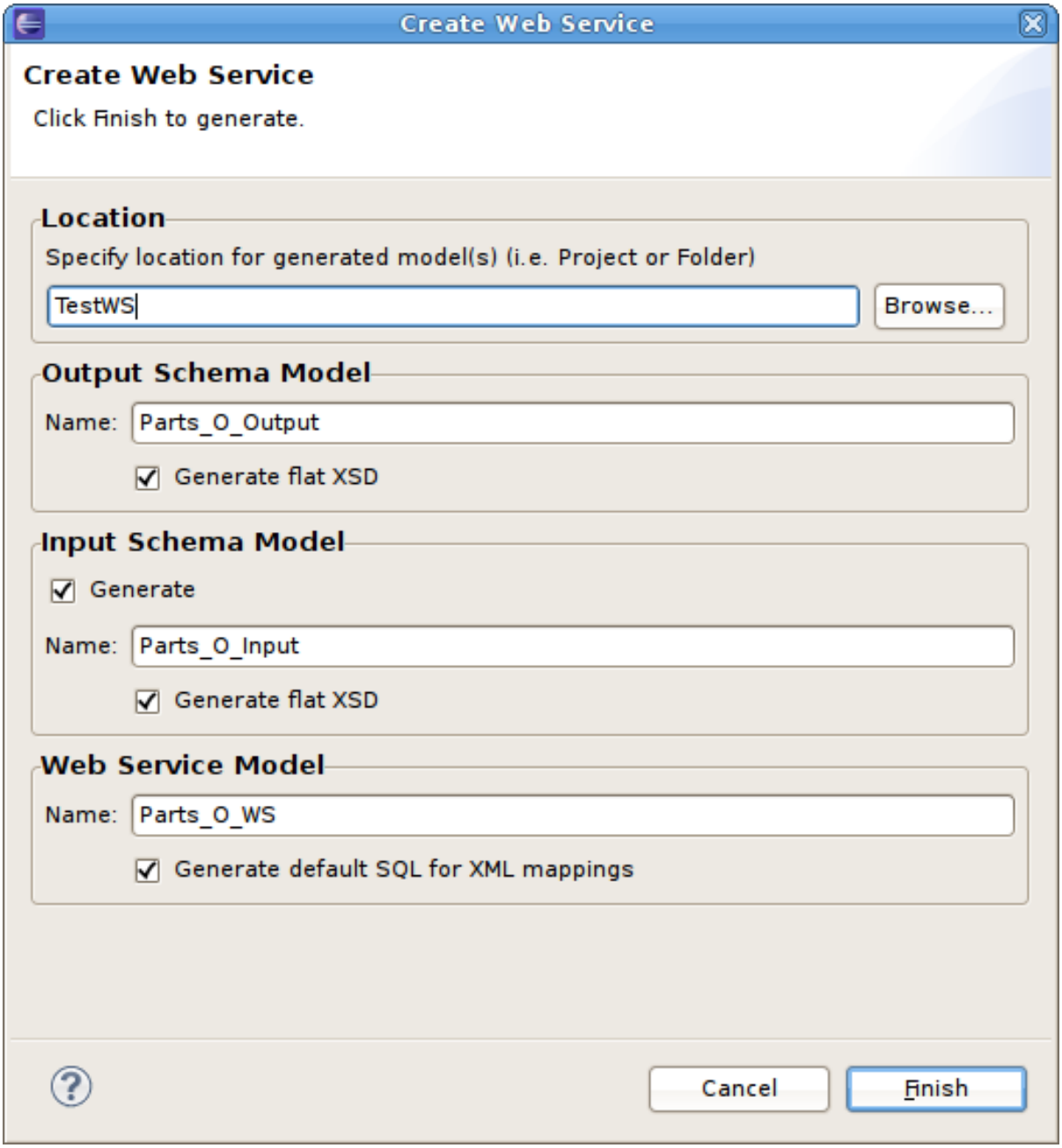


Figure 12.29. Create Web Service Dialog

4. When model generation is complete, a confirmation dialog appears. Click **OK**.

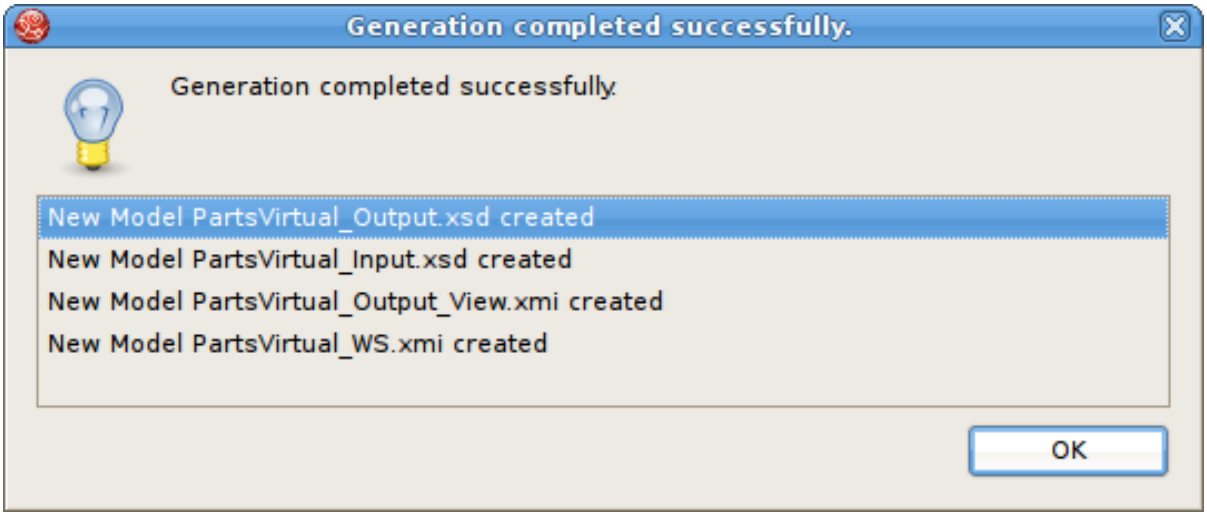


Figure 12.30. Generation Completed Dialog

12.4.2. Web Services War Generation

12.4.2.1. 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.

12.4.2.2. Generating a SOAP War

The **Teiid Designer** provides web service generation capabilities in the form of a JBossWS-CXF war. Once you have added your Web Service Models, as described in New Web Service View Model section, to your VDB, deployed the VDB to a running JBoss Data Virtualization 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:

1. Right-click on the VDB containing your web service model(s) and select the **`Modeling > Generate SOAP War`** action.

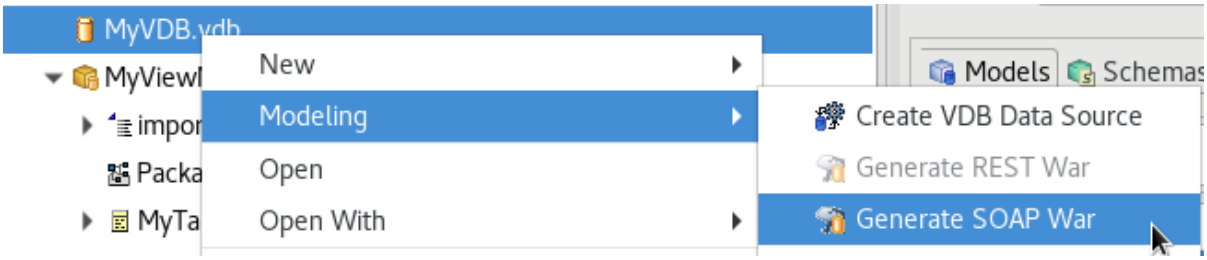


Figure 12.31.

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

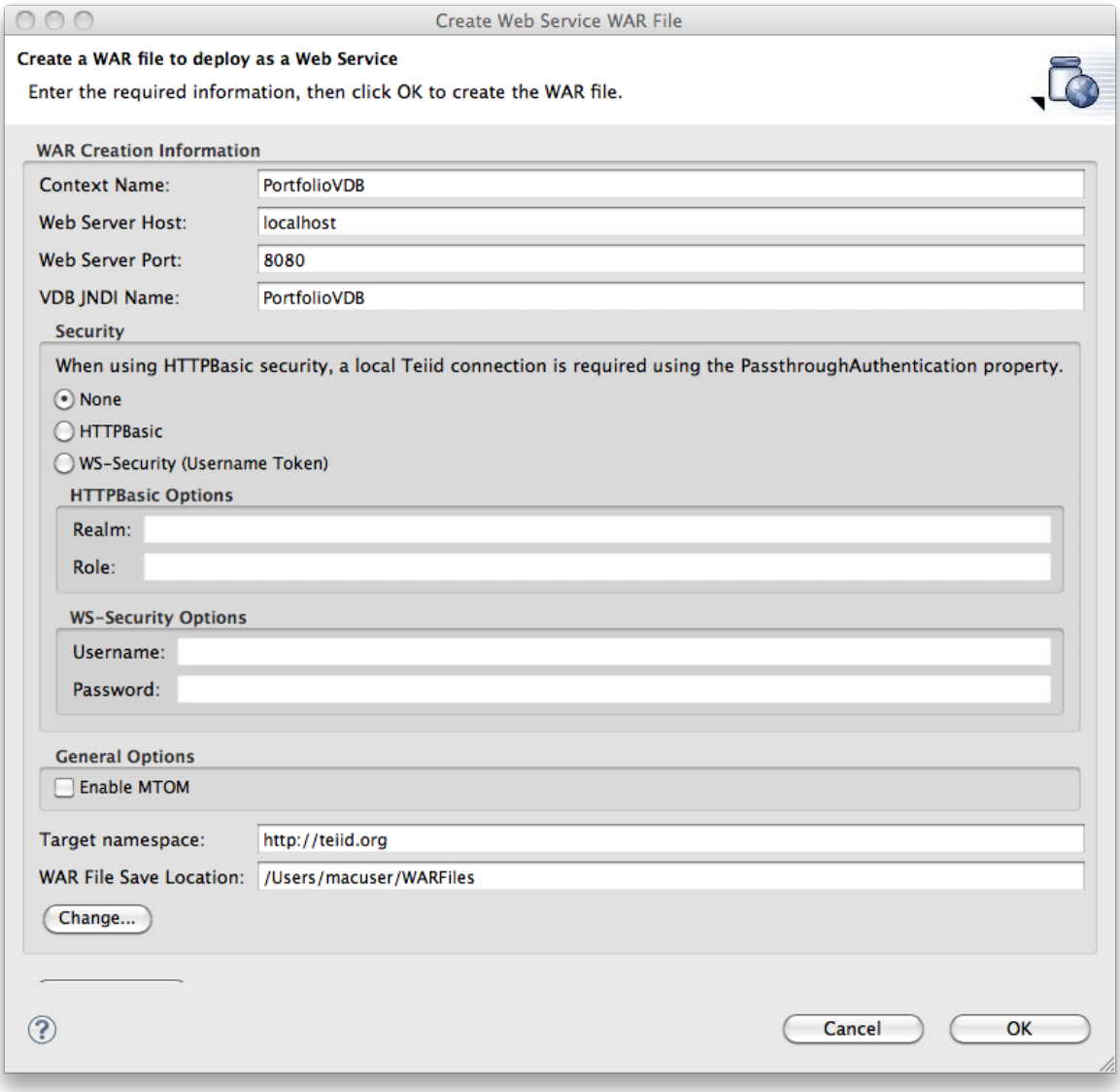


Figure 12.32. Generate a JBossWS-CXF War Web Service Dialog

Table 12.7. Field Descriptions

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

Field Name	Description
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 JBoss Data Virtualization (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. ◦ 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.
War File Save Location	The folder where the generated WAR file should be saved.

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

Figure 12.33. Generation Completed Dialog

4. Click **OK**.

12.4.2.3. 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, query 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. You can specify query parameters in the target REST procedure's URI property using & as a delimiter. For example, **`REST:URI = authors&parm1&parm2`**.

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:

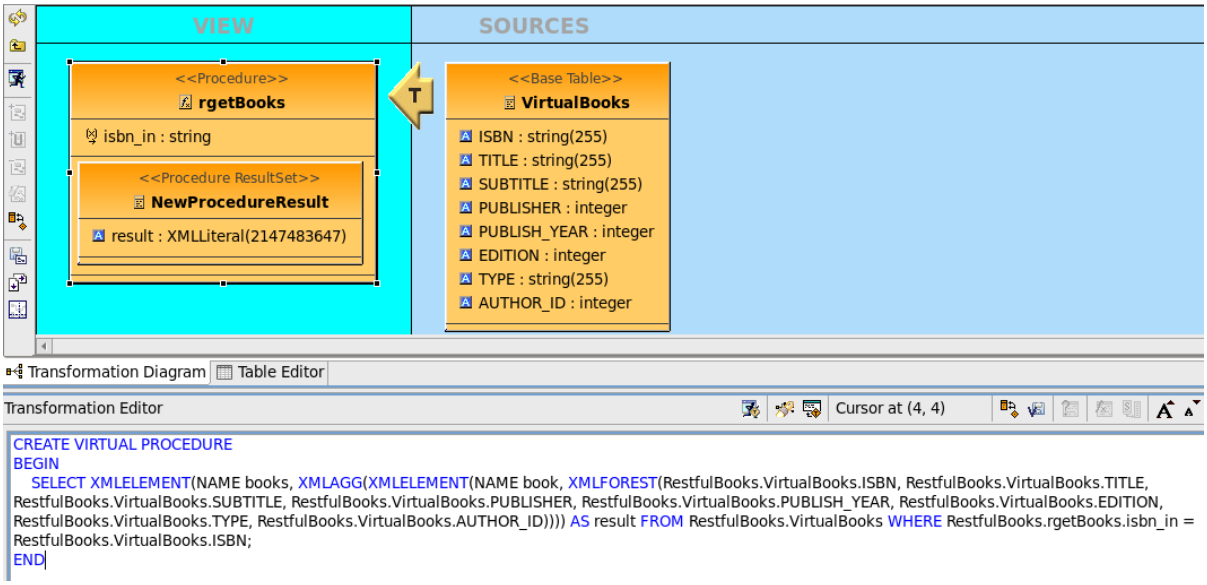


Figure 12.34.

Notice the syntax used to convert the relation table result of the select from VirtualBooks, to an XMLLiteral.

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

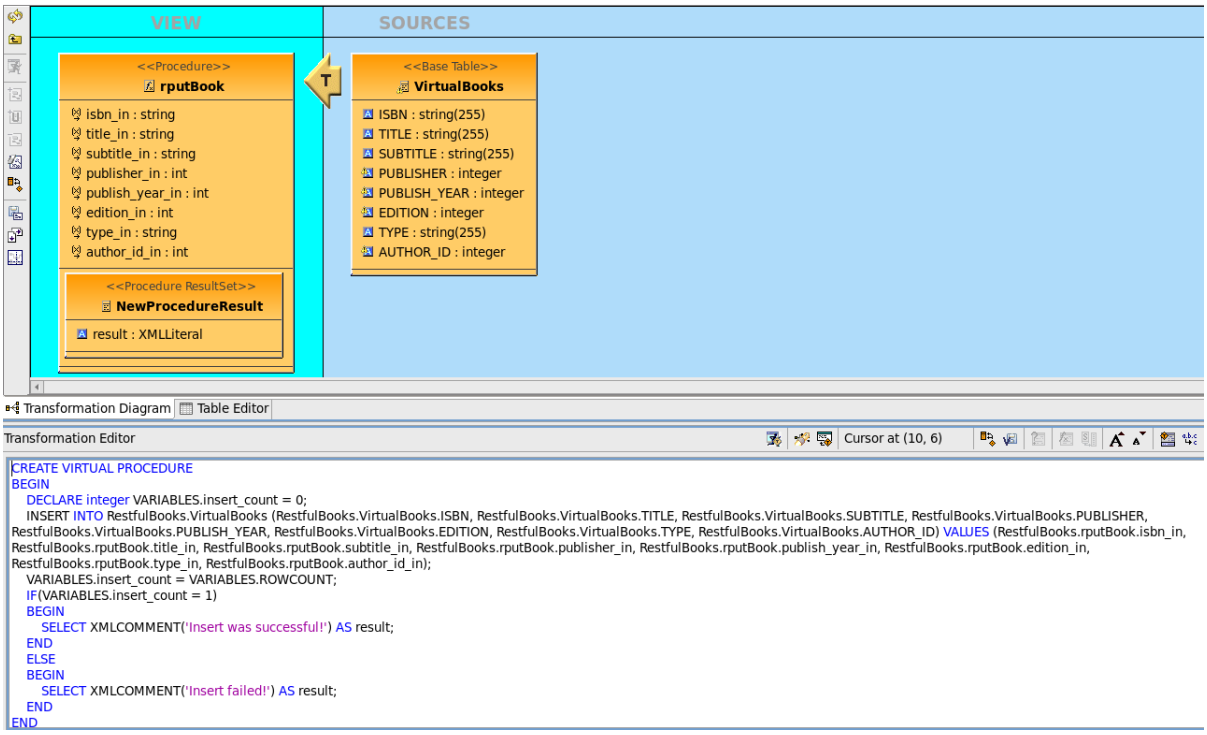


Figure 12.35.

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 must be <input> and the XML nodes must 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 12.36. Sample XML Input

When using a JSON document, ensure your values 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 12.37. Sample JSON Input

2. To enable REST for specific procedure we need to check this option in **`Create Relational View Procedure`** dialog. (When we are creating procedure)

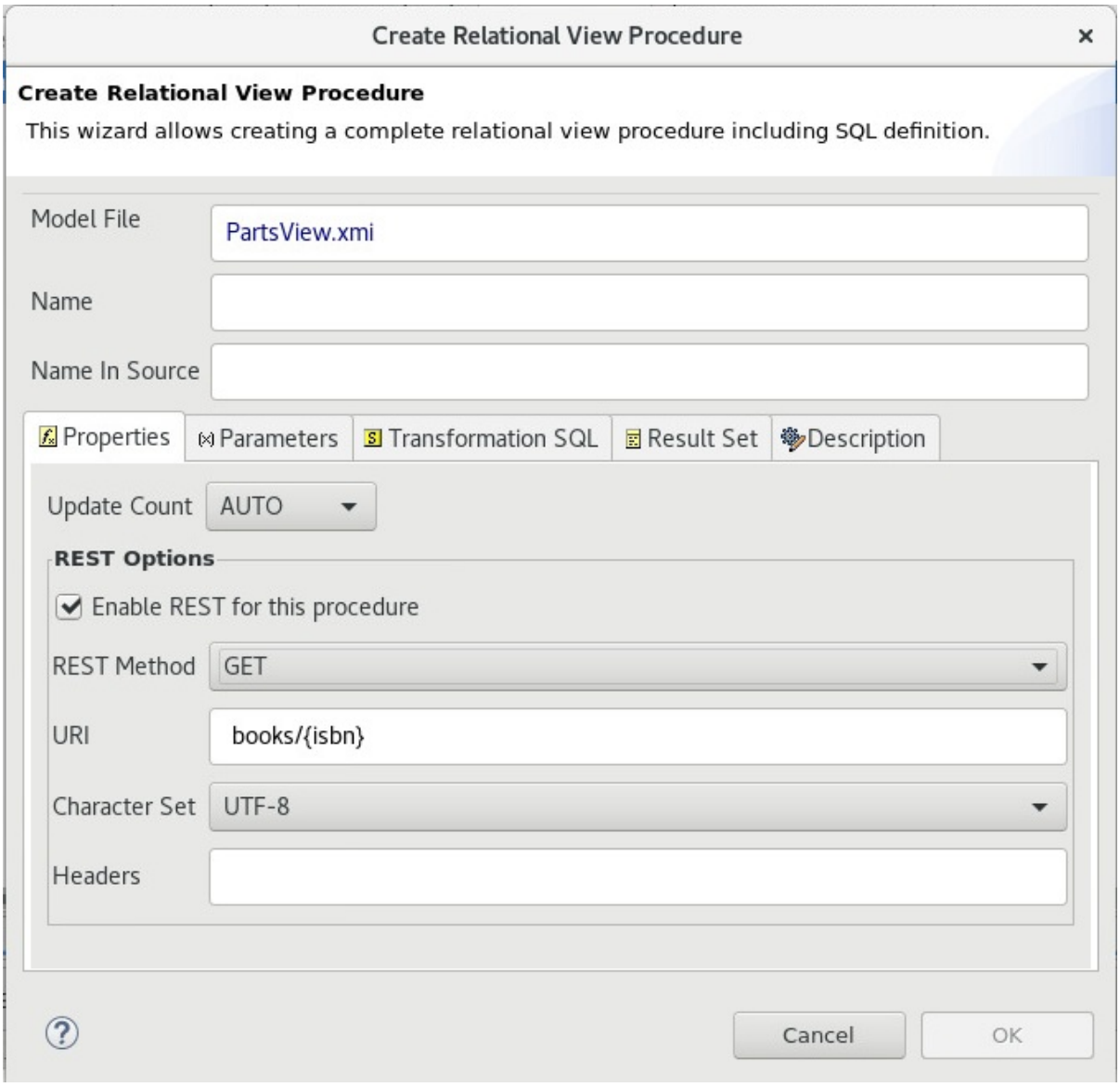


Figure 12.38.

This will enable six new properties in the property tab for this procedure defined in the model. These properties are defined in the table below:

Table 12.8. Extended Properties for RESTful Procedures

Property Name	Description
Rest Method	The HTTP method that will determine the REST mapping of this procedure. Supported methods are: GET, PUT, POST and DELETE
URI	The resource path to the procedure. For example, if you use books/{isbn} as your URI value for a procedure, http://{host}:{port}/{war_context}/{model_name}/books/123 would execute this procedure and pass 123 in as a parameter.
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. There are two supported options: UTF-8 and US-ASCII
Content Type	Type of content produced by the service. There are four supported types: any text, xml, plain and json.
Description	The description of this procedure.
HEADERS	Semi-colon delimited list of HTTP Header parameters to pass into the REST service operation. Example: header1;header2;etc. These values will be passed into the procedure first.

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

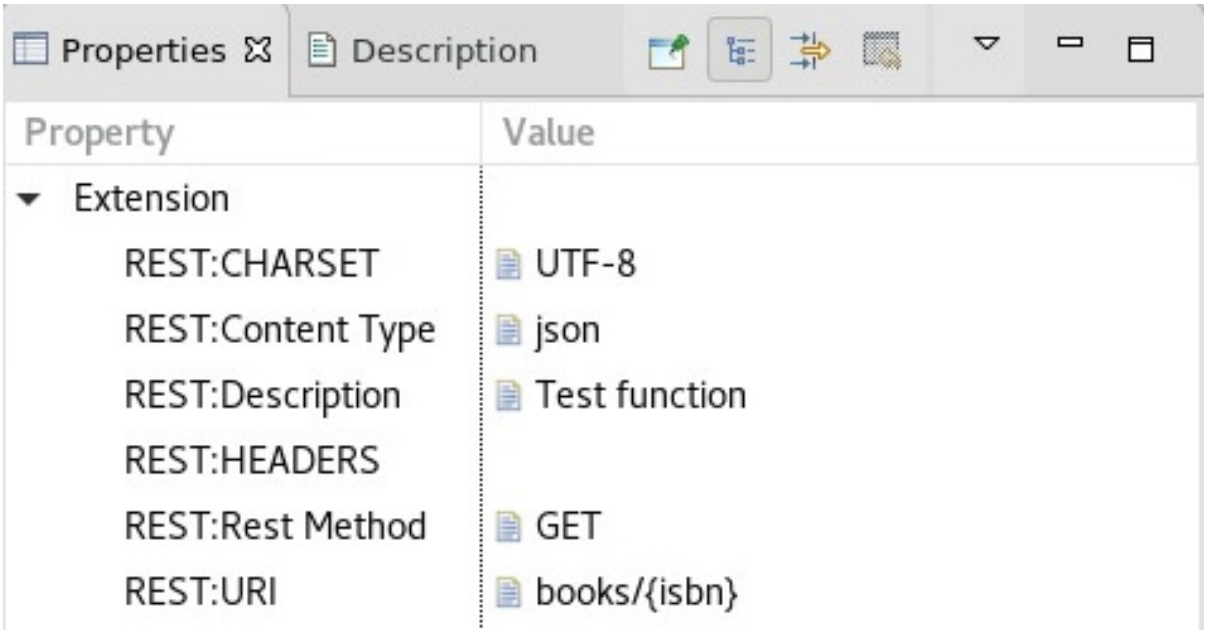
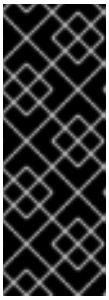


Figure 12.39.

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

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

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 **`Server 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.

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

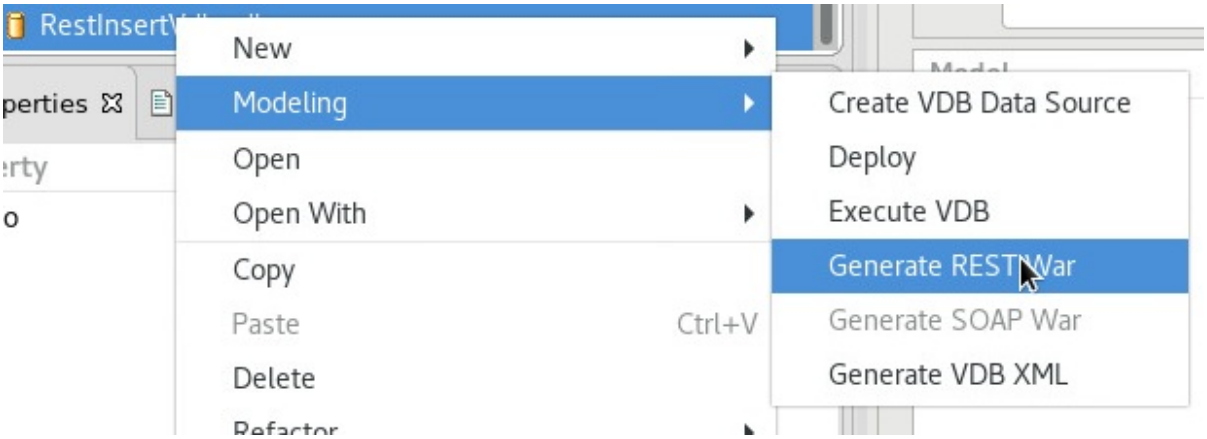


Figure 12.40.

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

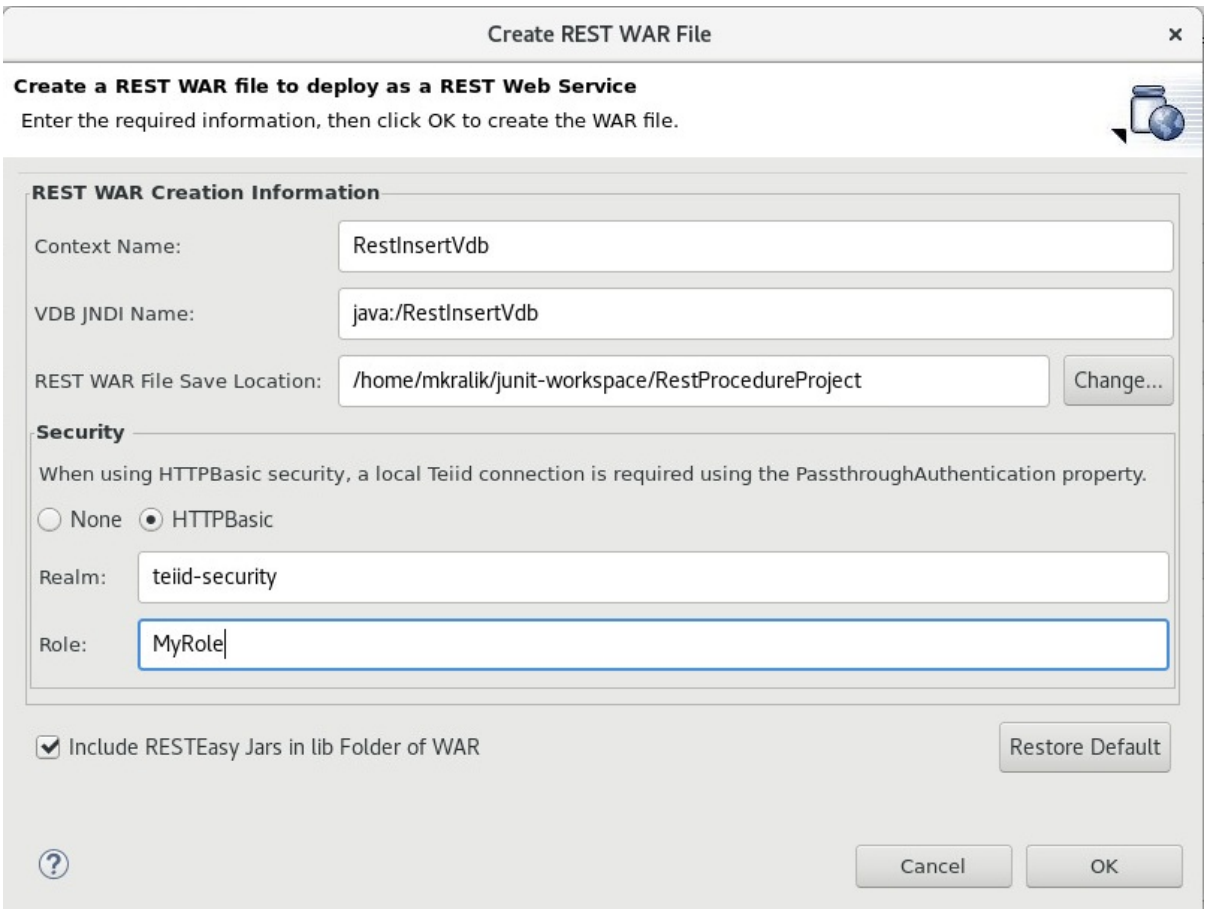


Figure 12.41. Generate a REST WAR War File Dialog

Table 12.9. 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.

Field Name	Description
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 folder of the generated WAR. If not selected, the jars will not be included. This should be cleared in environments where RESTEasy is installed in the classpath of the server installation to avoid conflicts.
None or HTTPBasic	Type of security. Two options: None or HTTPBasic security.
Realm	Defines a protection space.
Role	Defines role in your secure HTTPBasic system.

6. Click **OK** to generate the REST war. When war generation is complete, a confirmation dialog appears. Click **OK**.

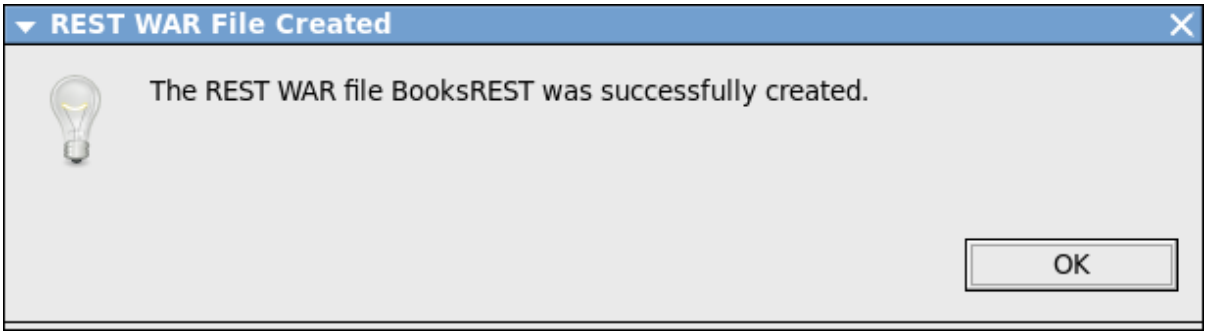


Figure 12.42. Generation Completed Dialog

12.4.2.4. Deploying Your Generated WAR File

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

From JBDS or JBoss Tools

1. Insure target JBoss Data Virtualization is configured and running.
2. Select your WAR file in the Model Explorer view. If you did not 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 Data Virtualization instance you have defined.

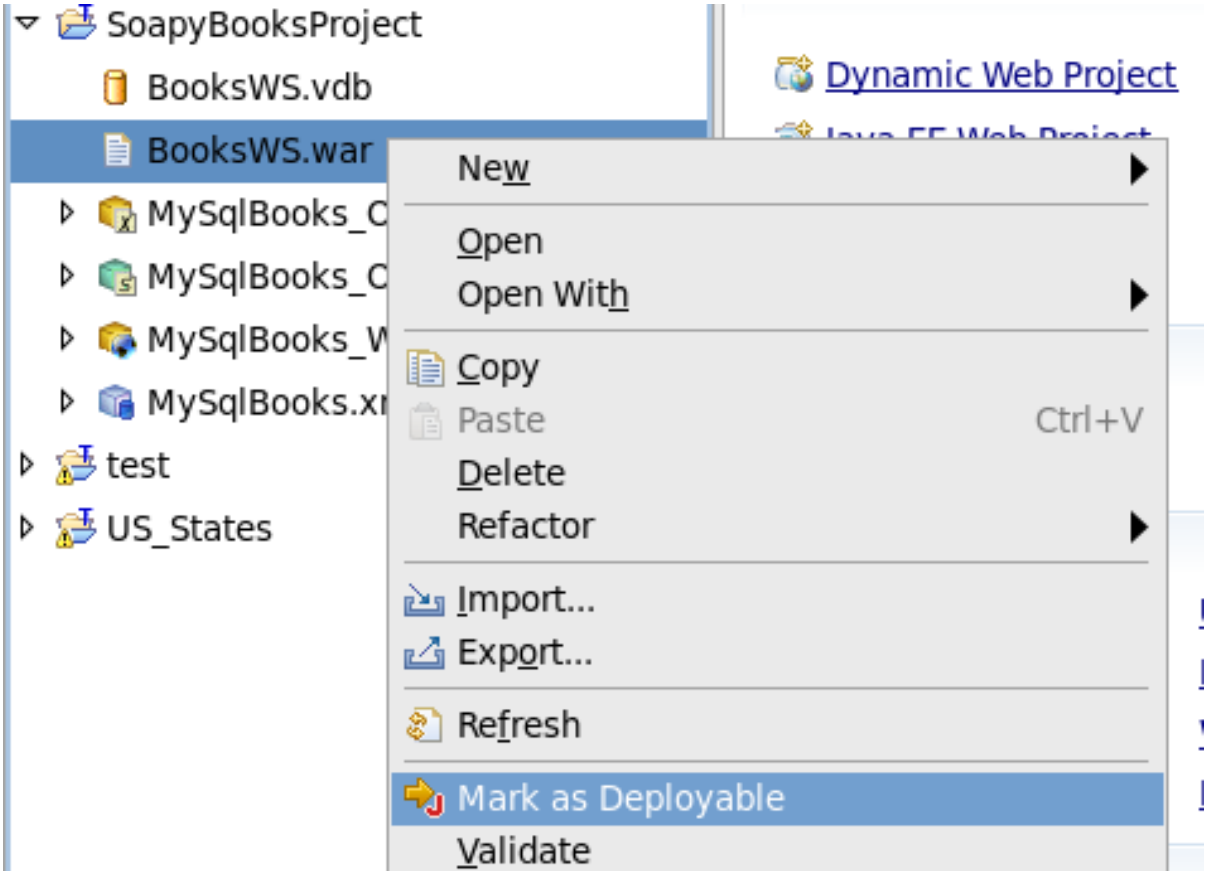


Figure 12.43.

Using the JBoss Data Virtualization Administration Console

1. Using the administration console that comes with JBoss Data Virtualization, you can deploy WAR files. The administration console is available at http://{hostname}:9990. Once logged on, use the **`Add`** button in the **`Deployments`** tab.

   Note that the default port for the administration console is **`9990`**. Depending on the **`jboss.socket.binding.port-offset`** property in the server configuration, the port number might be different.

Manual Deployment to JBoss Data Virtualization

1. It is possible to deploy the generated WAR by manually copying the file to the deploy folder of the target JBoss Data Virtualization. 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.

12.4.2.5. 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}/{warName}/{interfaceName}?wsdl`**. Also, you can show all web services in the Administration console. Once logged on, click on Runtime tab and choose Webservices from the side menu.

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.

Using your REST URL, you can 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.

CHAPTER 13. EDITING MODELS AND PROJECTS

13.1. EDITING MODELS AND PROJECTS

Teiid Designer offers three basic model edit actions: Rename, Move and Save As... and one project related action, Clone Project.

13.2. RENAME A MODEL

To rename a model in your workspace:

1. Select a model in the **`Model Explorer`** view.
2. Right-click select the **`Refactor > Rename`** action.

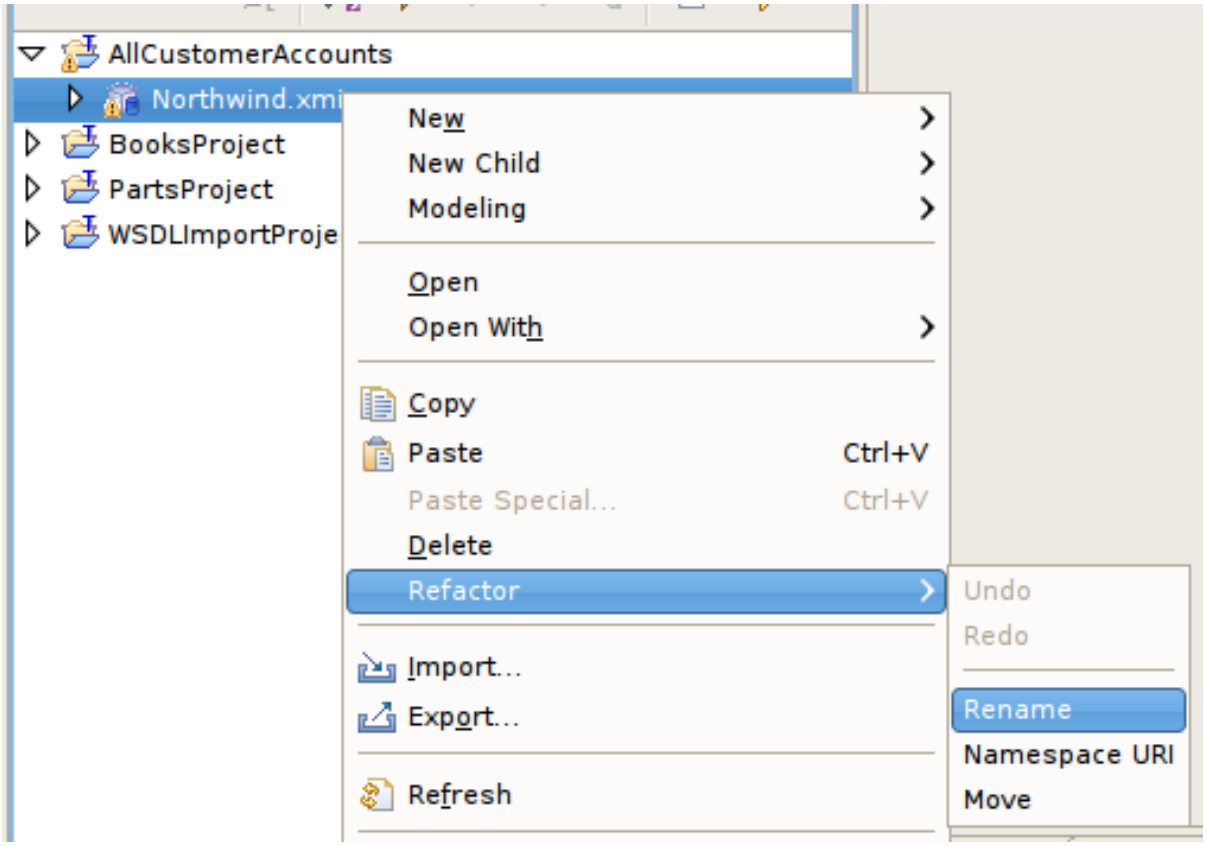


Figure 13.1. Refactor Rename Action In Model Explorer

3. Specify unique model name in the **`Rename Resource`** dialog. Click **`OK`**.

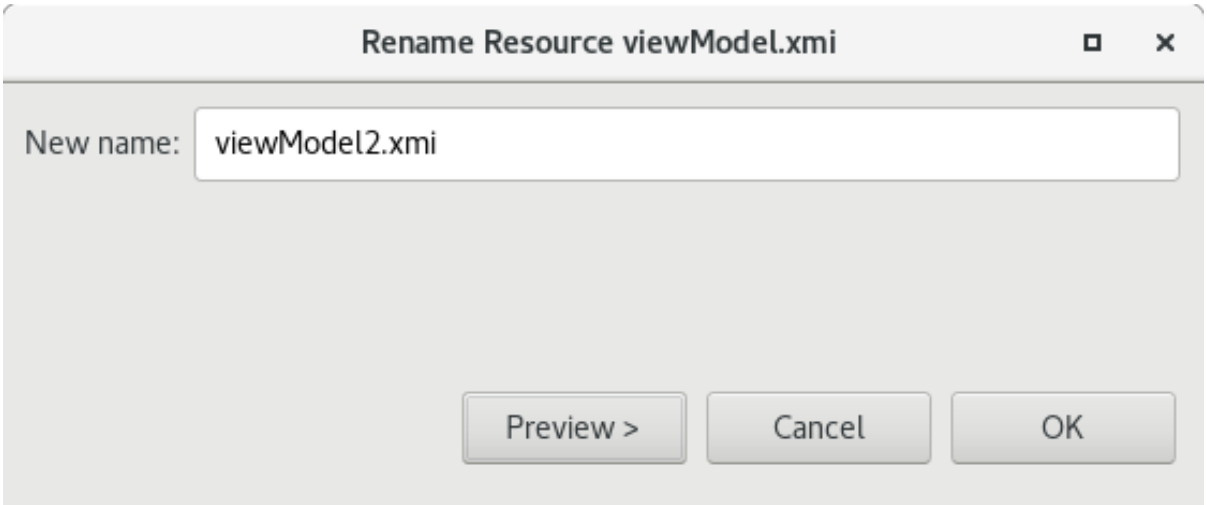


Figure 13.2. Rename Resource Dialog

NOTE

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

13.3. MOVE MODEL

To move a model in your project:

1. Select a model in the **`Model Explorer`** view.
2. Right-click select the **`Refactor > Move`** action.

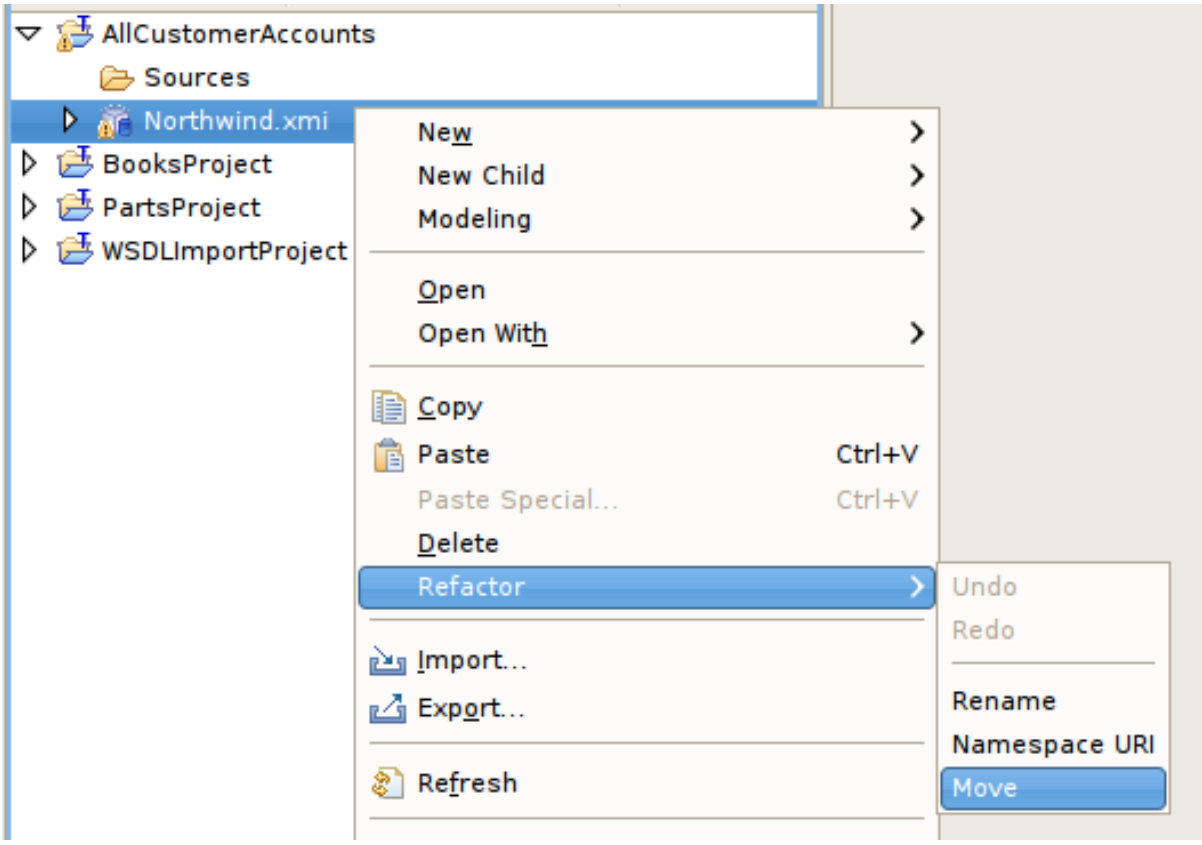


Figure 13.3. Refactor Move Action In Model Explorer

3. Select the new folder location and click **OK**.

13.4. SAVE COPY OF MODEL

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

NOTE

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

To create a duplicate model using Save As...:

1. Open the model you wish to copy in a Model Editor by double-clicking the model in Model Explorer view or right-click and click **Open** action.

2. Select the editor tab for the model you opened.

Figure 13.4. Select Editor Tab

3. Click **`File > Save As...`** action to open the **`Save Model As`** dialog.

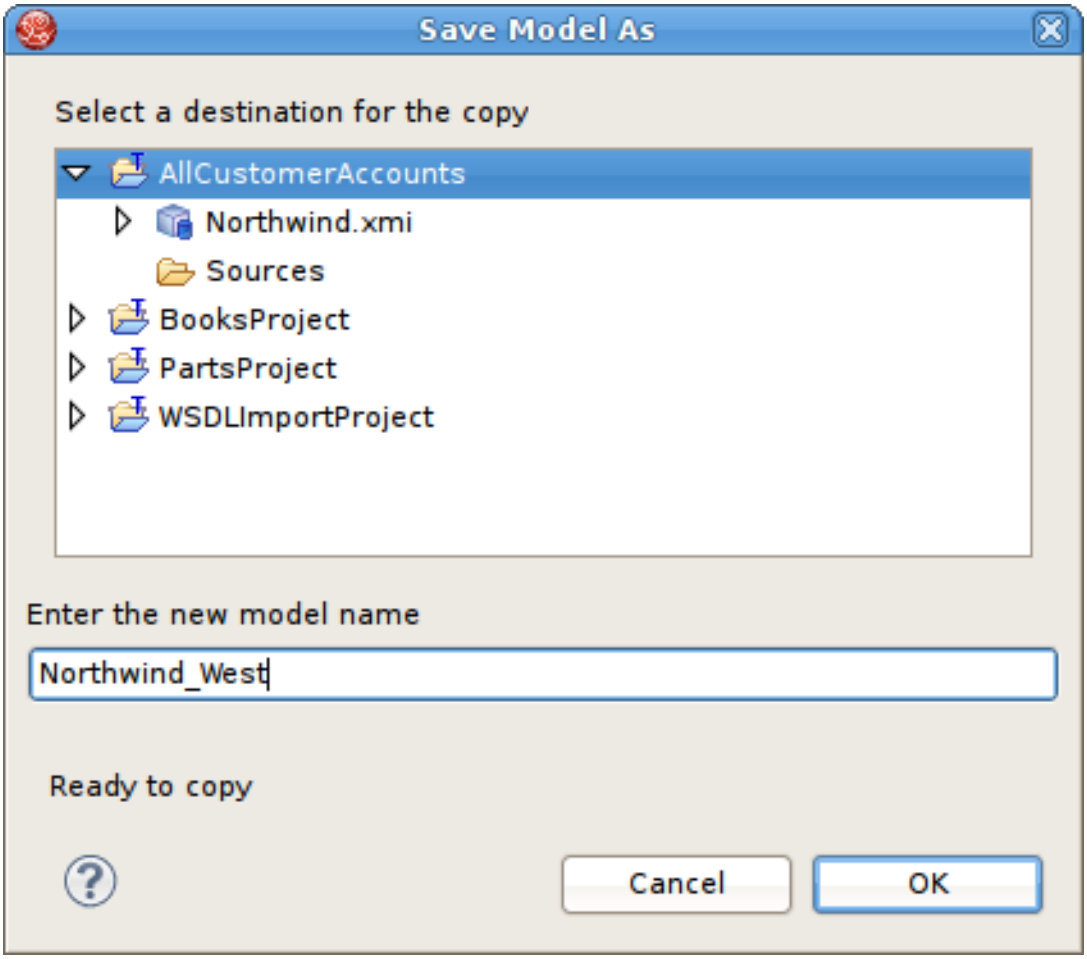


Figure 13.5. Save Model As Dialog

4. Enter a unique model name in the new model name text field and click **`OK`**.

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.

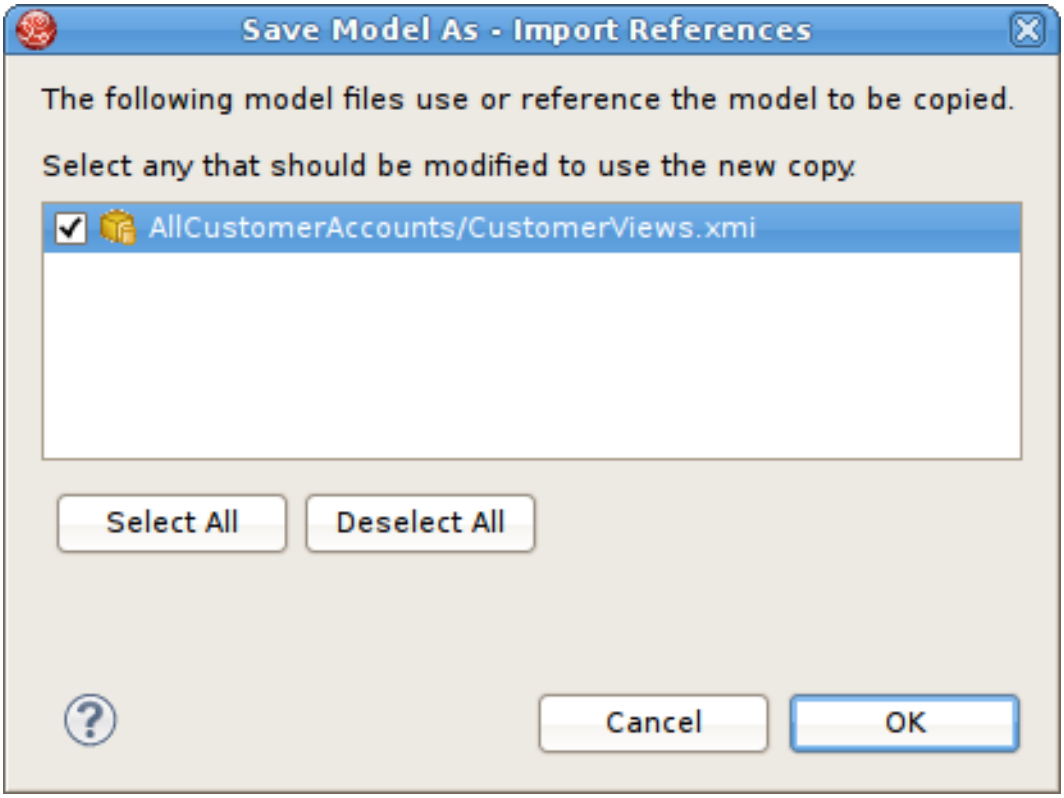


Figure 13.6. Save Model As Dialog

13.5. CLONE PROJECT

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

The following lists specific rules and limitations for this action.

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

To clone a model project:

1. Select an existing model project in the **`Model Explorer`** view.
2. Right-click, then select **`Model Project > Clone`** in the context menu. Otherwise you can select the **`Project > Clone Project`** action located in the **Teiid Designer**'s main menu bar.

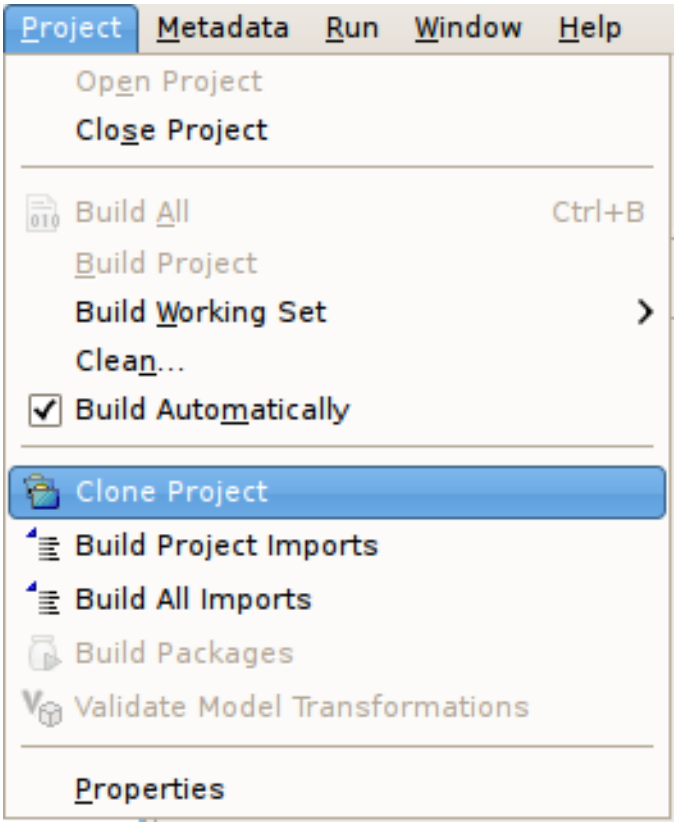


Figure 13.7. Clone Project In Project Menu

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

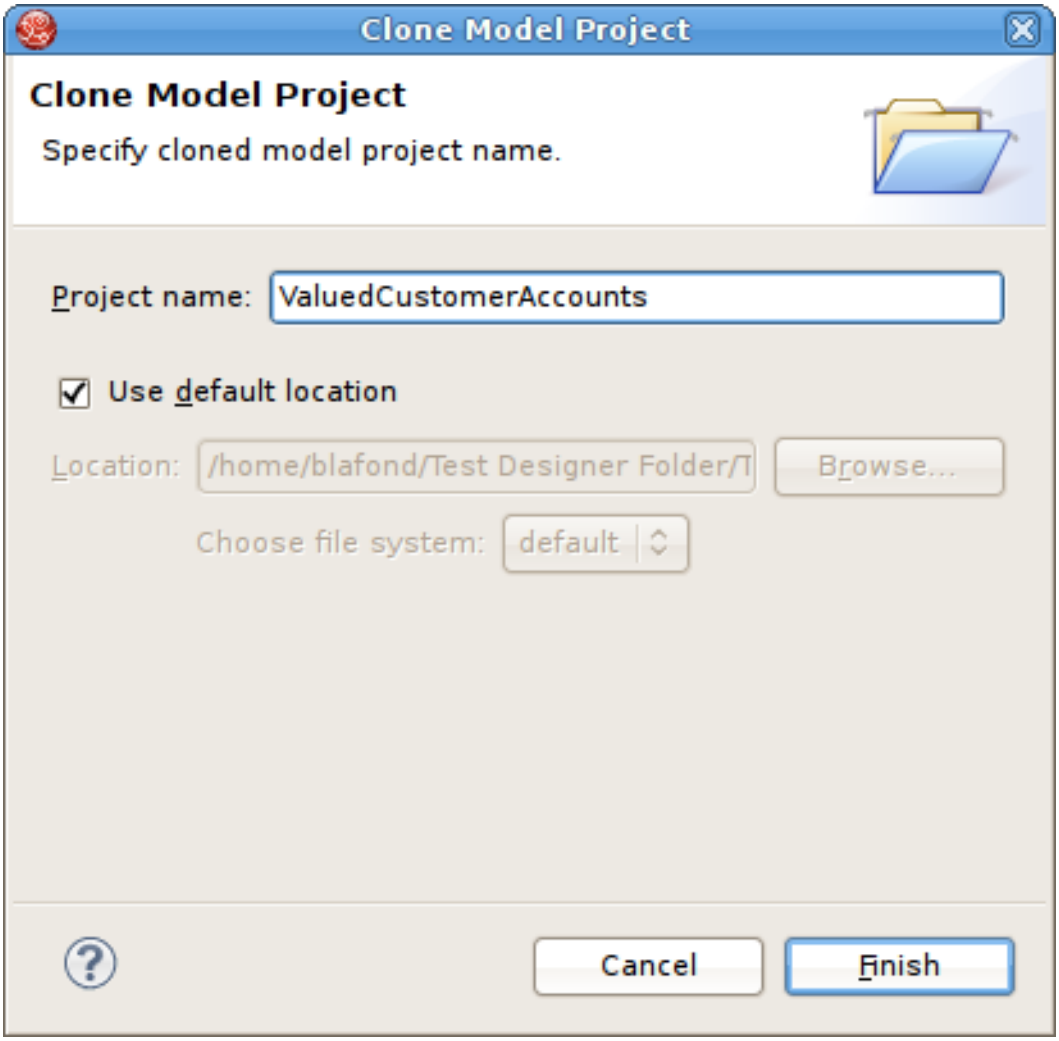


Figure 13.8. Clone Project In Project Menu

4. (Optional) If you wish to create your cloned project in a location other than your default workspace location, clear the **`Use default location`** checkbox and specify (type in or browse to) a new directory location on your local file system.

5. Click **`Finish`** to generate your new project.

CHAPTER 14. MANAGING YOUR VIRTUAL DATABASES

14.1. CREATE A VIRTUAL DATABASE

Procedure 14.1. Create a Virtual Database

1. Launch Red Hat JBoss Developer Studio's **`New`** wizard.
2. Open the **`Teiid Designer`** category directory and select **`Teiid VDB`**.

   NOTE

   You can also select one or more models in a model project. To do so 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.

   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.

3. Fill in VDB Name and then click **Finish**.

14.2. EDIT A VIRTUAL DATABASE

Procedure 14.2. Edit an Existing Virtual Database

1. Select the VDB you want to edit in the **`Explorer`**.
2. Right-click and select the **Open** action. (You could also double-click it.)

   The VDB will then open in the Editor.

14.3. MULTI-SOURCE BINDING SUPPORT

Teiid Designer allows you to define relational source models and binding them to multiple data sources.

Use Multi-source models to quickly access data in multiple sources with homogeneous metadata. When you have multiple instances using identical schema, Red Hat JBoss Data Virtualization 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 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

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.

14.4. USER-DEFINED FUNCTIONS 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.

When a UDF model is added to a VDB, each scalar function is interrogated and its referenced UDF jar (if available) is added to the VDB as well as shown in the UDF **Jars** tab in the editor

14.5. REUSING VIRTUAL DATABASES

You can 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 import-vdb XML element in the **vdb.xml** definition. 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.

This sample **vdb.xml** file 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

14.6. CREATE A VDB SOURCE MODEL

Procedure 14.3. Create a VDB Source Model

1. Deploy your VDB.
2. Launch the **`JDBC Import Wizard`** by clicking **Import** - **Teiid Designer** - **JDBC Database >> Source Model**.
3. On the first page of the wizard 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`**.

   NOTE

   If **`Import as VDB source model`** is NOT checked, then the wizard will continue importing as a normal JDBC import

4. On the **`Select Database Objects`** page, select a single schema to use in order to create as VDB source model.
5. The final page shows the name of the resulting VDB source model. The name is NOT editable.

   NOTE

   The target Into Folder must NOT contain a model with the same name or the Finish button will be disabled.

   NOTE

   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 reference is added in its 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.

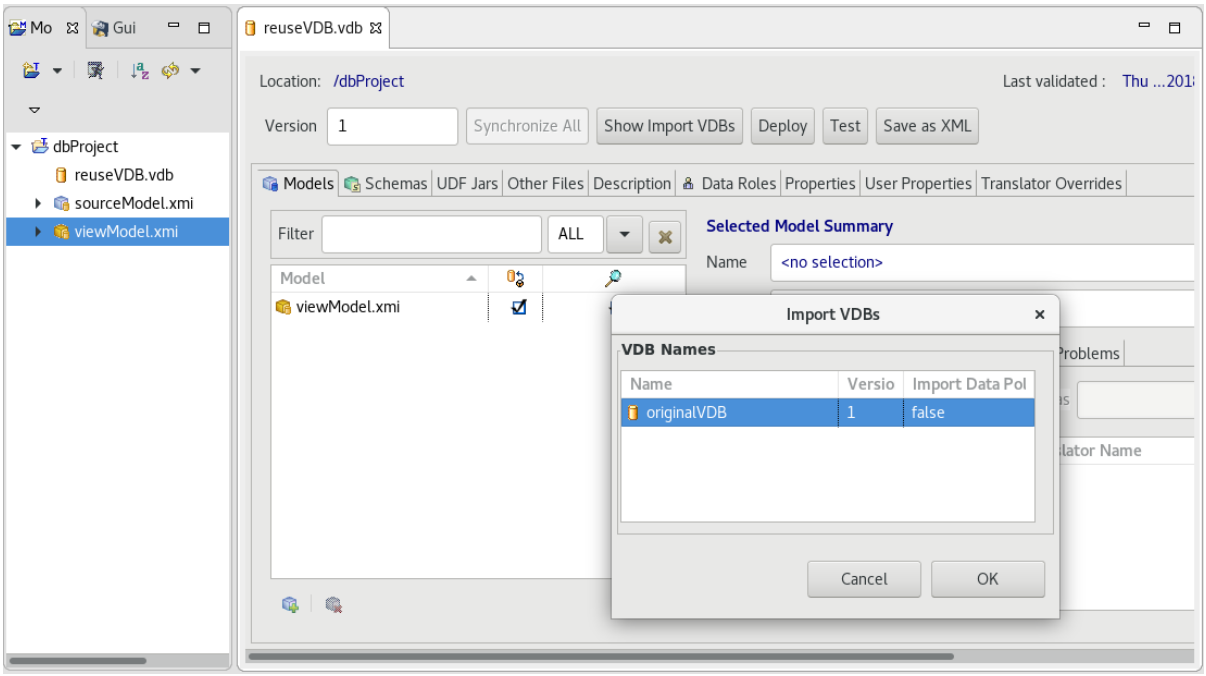


Figure 14.1. VDB Editor

14.7. SECURITY AND DATA ACCESS

You have some options on defining your data access security for your VDB via the VDB Editor.

The first level is provided by the model visibility check-box in the Models section (Spyglass column). If unchecked, that model and its 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 Data Roles tab 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 the following:

- Security (Row-based condition and column masking)
- Create
- Read
- Update
- Delete
- Execute
- Alter

Double-clicking the **`Security`** box for a table will launch the **`Row Filter Definition`** dialog where you can define condition. Double-clicking the **`Security`** box for a column will launch the **`Column Masking Definition`** dialog where you can define condition and column masking. In order to edit or remove security, select the **`Row Filter`** or **`Column Masking`** tabs and use the **`Edit`** or **`Remove`** buttons.

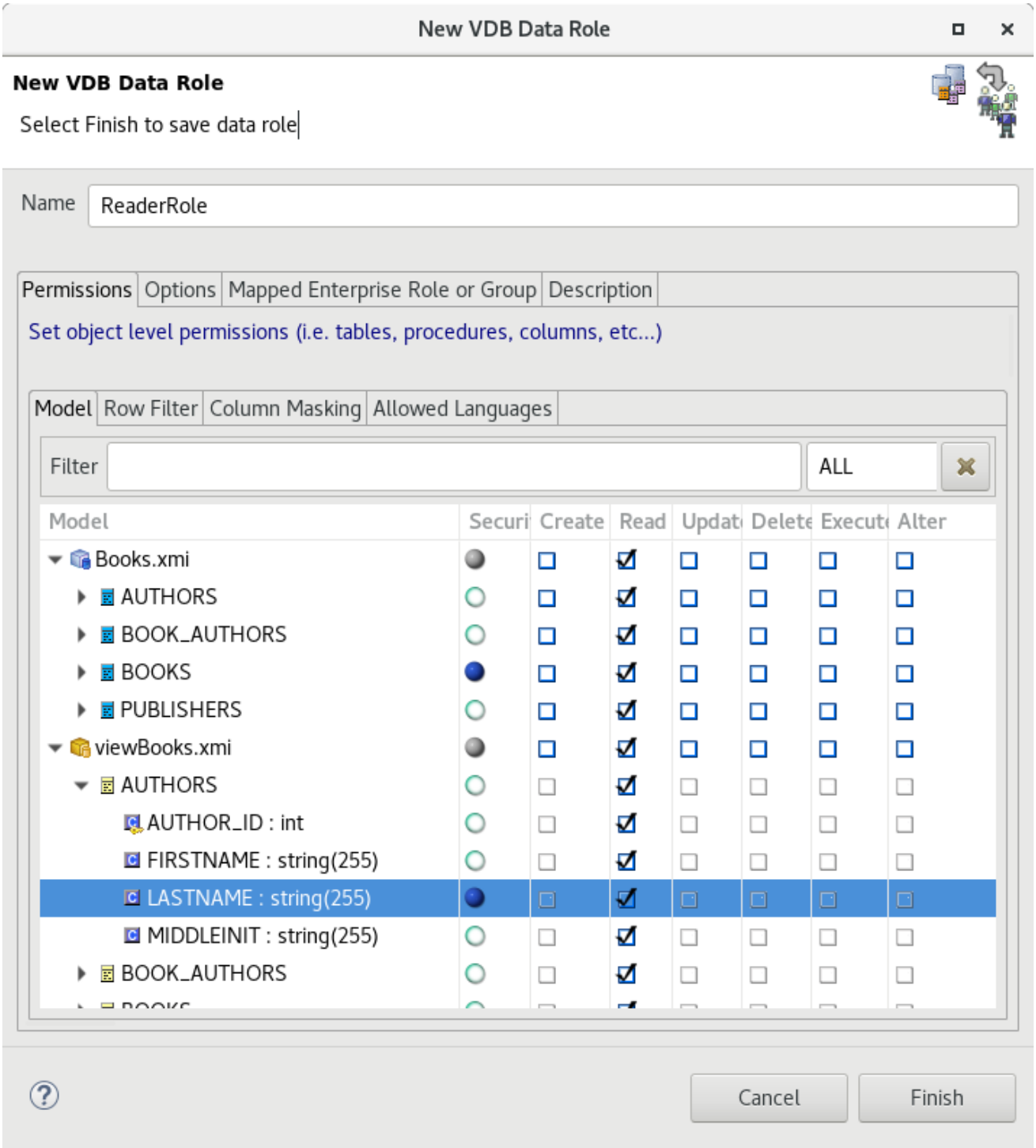


Figure 14.2. Data Role dialog

CHAPTER 15. TESTING YOUR MODELS

15.1. MANAGE CONNECTION PROFILES

15.1.1. Manage Connection Profiles

As described earlier, you can test your models in **Teiid Designer** by using the **`Preview Data`** action or test your models via your deployable VDB.

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

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

The Server view provides access to running Teiid instances and shows data source and VDB artifacts deployed there. The Create Data Source action available on this view utilizes the available and applicable connection profiles.

15.1.2. 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.

15.1.3. 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 Profile Info`** action which displays the detailed properties of the connection.

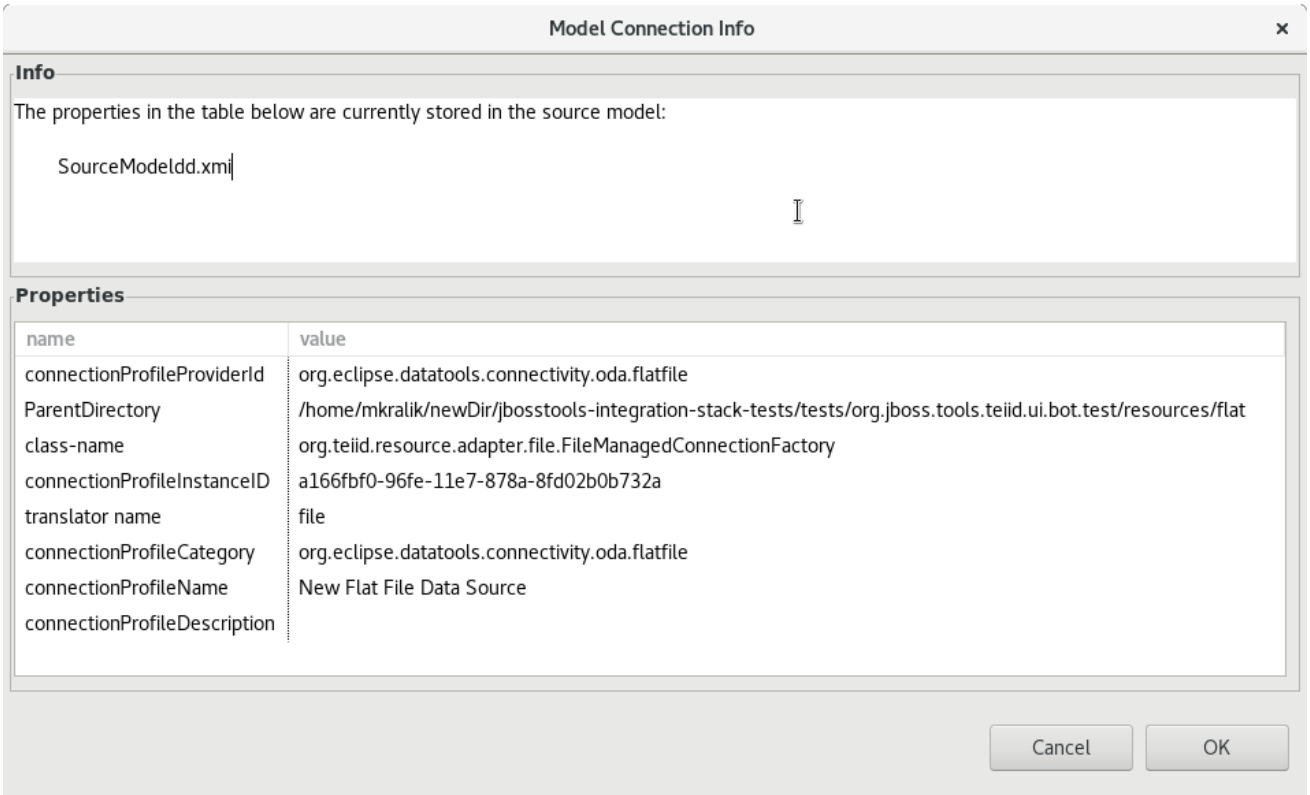


Figure 15.1. Connection Profile Information Dialog (in this image is used flat file connection profile)

NOTE

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

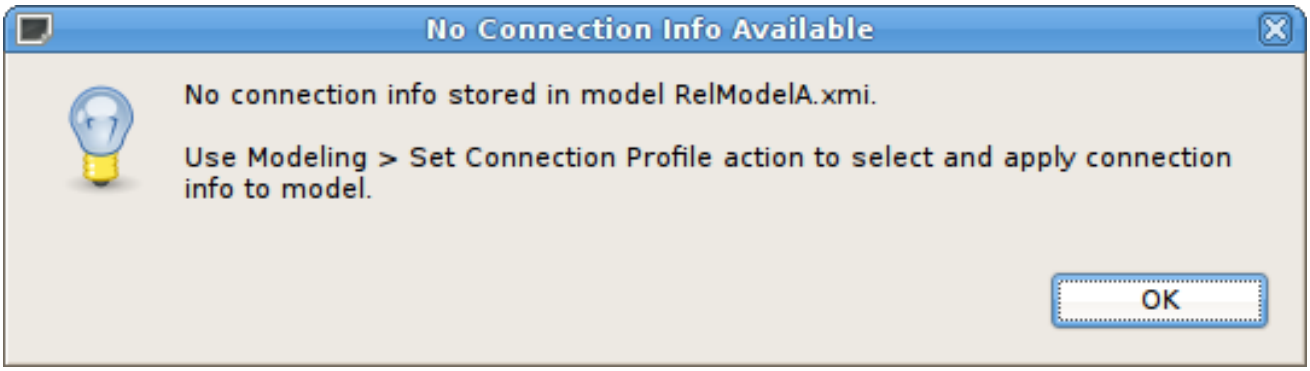


Figure 15.2. No Connection Info Dialog

15.1.4. 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.

15.2. PREVIEWING DATA FOR A MODEL

15.2.1. 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 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. 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 **`SQL Results`** 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.

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

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

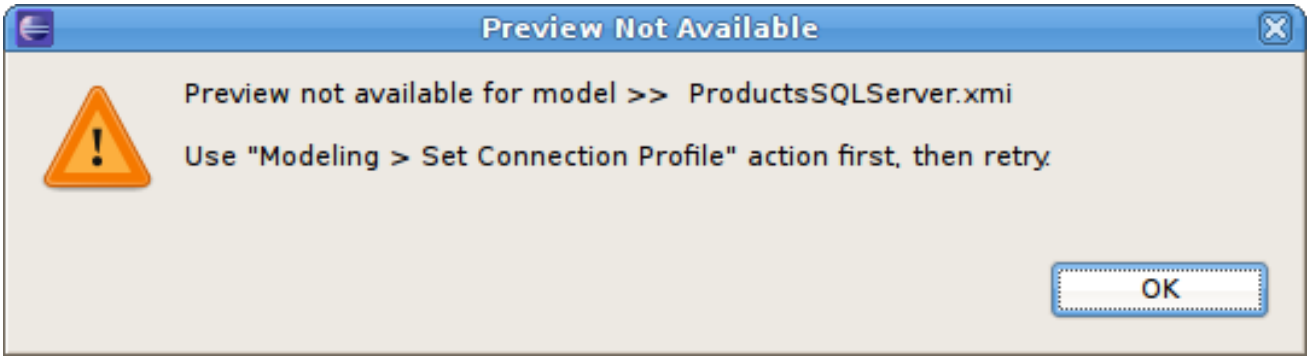


Figure 15.3. Preview Not Available

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

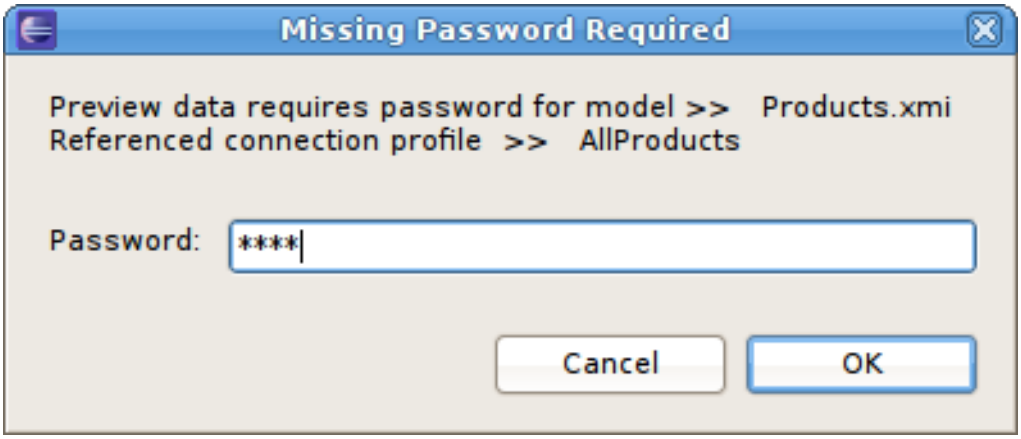


Figure 15.4. Missing Password

Testing Your Transformations

When editing transformation SQL in the Transformation Editor, a special **`SQL Results`** data action is provided in the editor toolbar . You can change your transformation SQL, re-validate and preview your the data for your modified SQL. The following sections provide steps for previewing your data. Note that all steps assume that all source models referenced by your models, either directly or through dependencies, are bound to connector bindings.

15.2.2. Preview Relational Table or View

To preview a relational table, relational view or staging table:

1. Select a relational table or view in the 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 Model Explorer view, so you need to open the mapping diagram and select it there.

2. Right-click select the **`Preview Data`** action . You can also select the same action in the toolbar of either the Model Explorer view or diagram.

3. Your query results will be displayed in the **`SQL Results`** view. The view will automatically open or get focus if not visible in your perspective.

15.2.3. Preview Relational Table With Access Pattern

To preview a relational table or view with access pattern:

1. Select a relational table or view in the 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.

2. Right-click select the **`Preview Data`** action . You can also select the same action in the toolbar of either the Model Explorer view or diagram.

3. A column input dialog is presented. Select each access pattern and enter a value for each required column.

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.

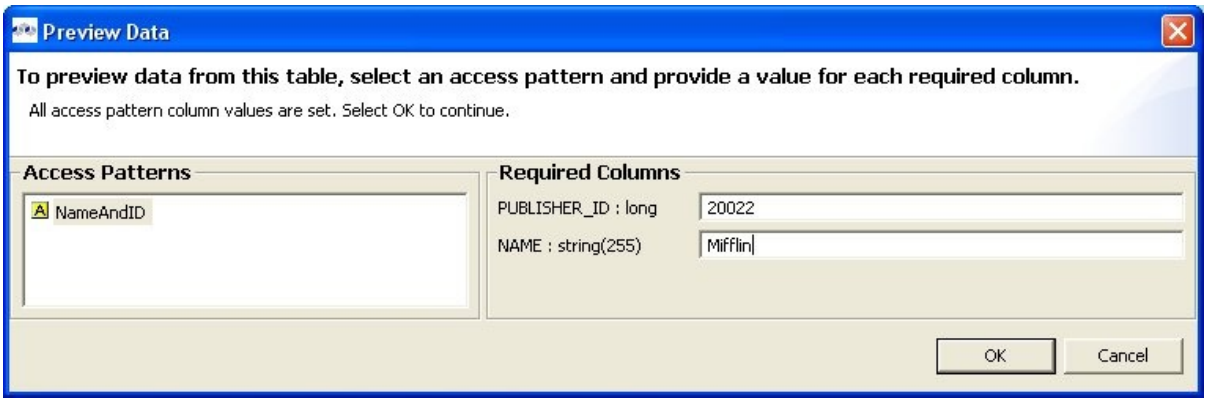


Figure 15.5. Access Pattern Column Input Dialog

4. Your query results will be displayed in the **`SQL Results`** view. The view will automatically open or get focus if not visible in your perspective.

15.2.4. Preview Relational Procedure

To preview a relational procedure:

1. Select a relational procedure in the Model Explorer view or diagram. The procedure can be in a view model as well as a source model.

2. Right-click select the **`Preview Data`** action . You can also select the same action in the toolbar of either the Model Explorer view or diagram.

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.

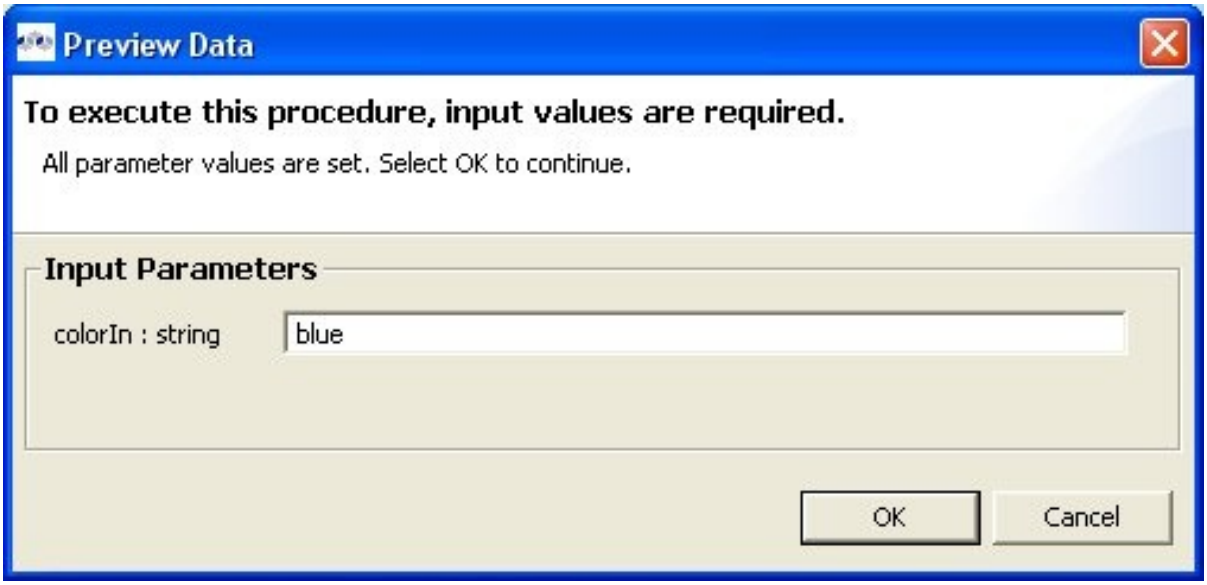


Figure 15.6. Procedure Parameter Input Dialog

4. Your query results will be displayed in the **`SQL Results`** view. The view will automatically open or get focus if not visible in your perspective.

15.2.5. Preview Web Service Operation

To preview a Web service operation:

1. Select a Web service operation in the Model Explorer view or diagram. The operation can be in a view model as well as a source model.

2. Right-click select the **`Preview Data`** action . You can also select the same action in the toolbar of either the Model Explorer view or diagram.

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.

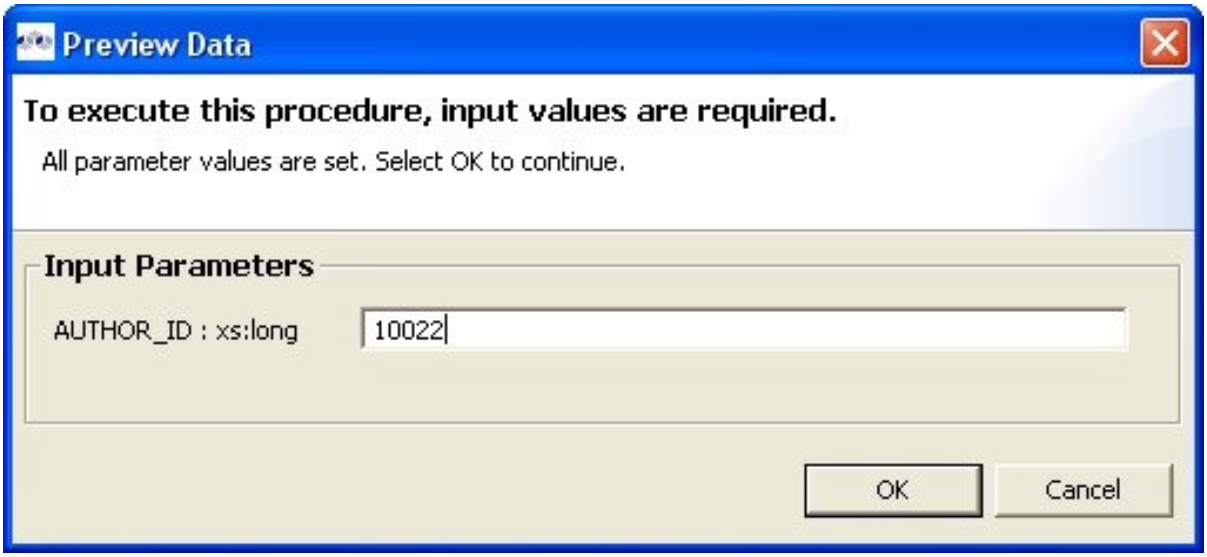


Figure 15.7. Procedure Parameter Input Dialog

4. Your query results will be displayed in the **`SQL Results`** view. The view will automatically open or get focus if not visible in your perspective.

15.2.6. 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 drop-down menu.

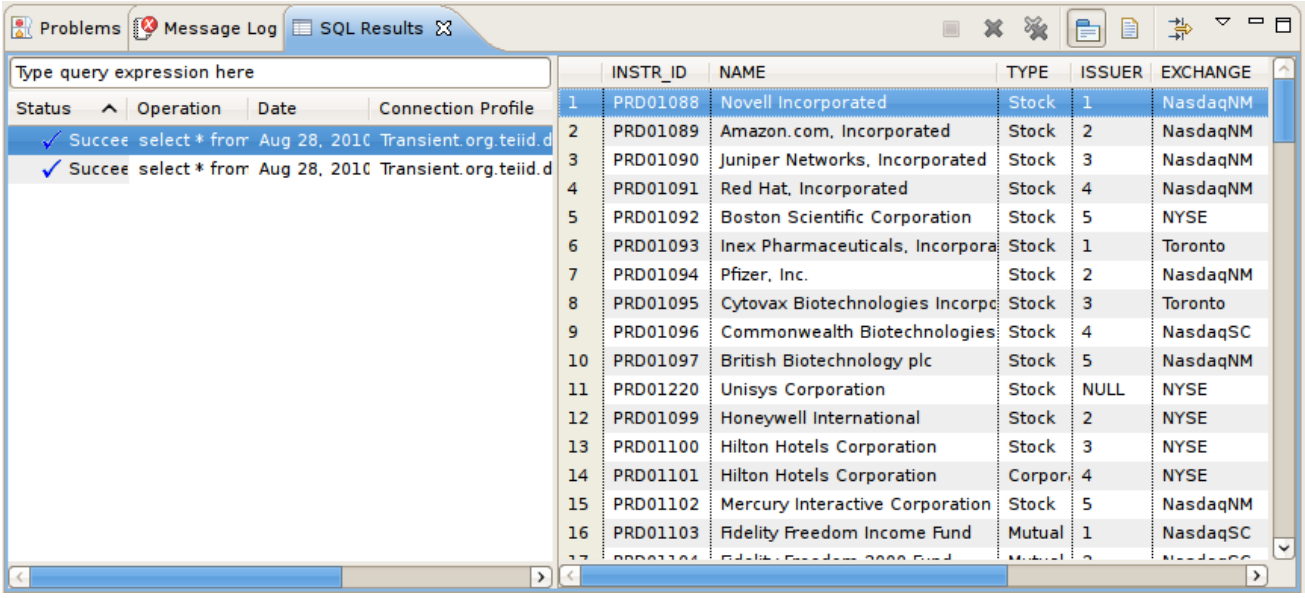


Figure 15.8. SQL Results View

15.2.7. Execution Plans

When Preview Data is executed and you checked **`Generate Execution Plan`** option in the **`Custom Preview Data`** dialog, you can also show the **`Teiid Execution Plan`** view as shown below. The

Execution Plan may also be obtained by right-clicking on a previewable object, then selecting **`Modeling > Show Execution Plan`** in the context menu.

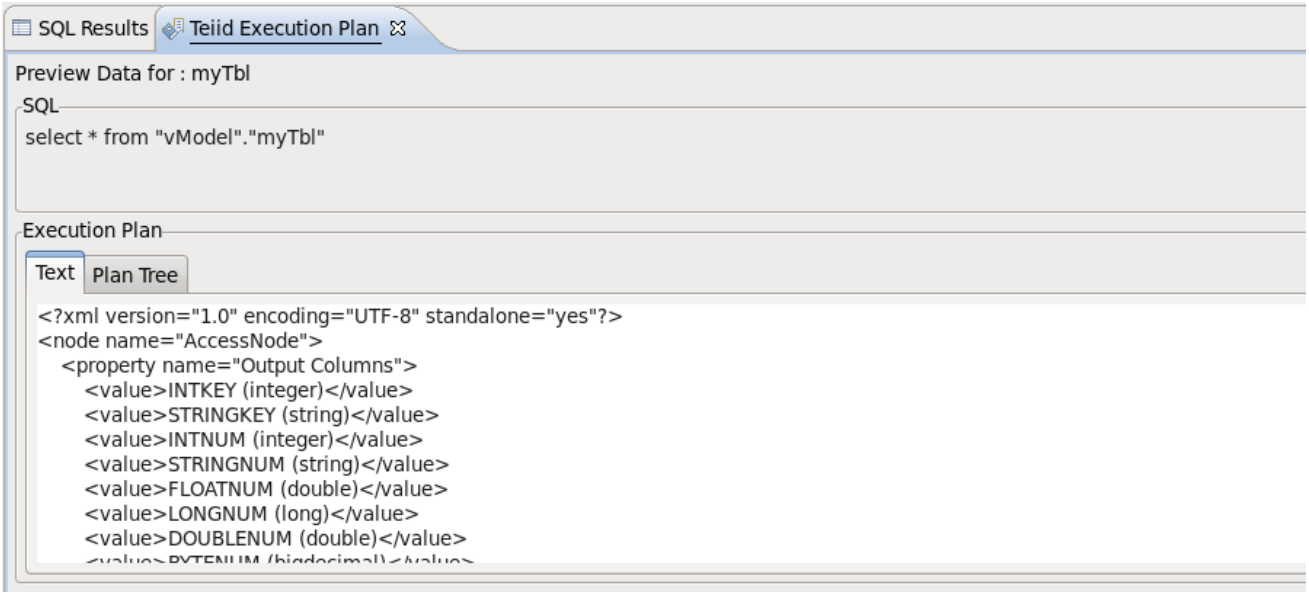


Figure 15.9. Teiid Execution Plan View

15.3. TESTING WITH YOUR VDB

15.3.1. 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.

15.3.2. Creating Data Sources

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

When creating VDBs you do not need to have deployed data sources on your **JBoss Data Virtualization** 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 JBoss Data Virtualization server.

You can also create data sources from the Server view. Expand a JBoss Data Virtualization server instance in the Server view, select a particular data source and right-click the **`Create Data Source`** action. This will launch the **`Create Data Source`** dialog.

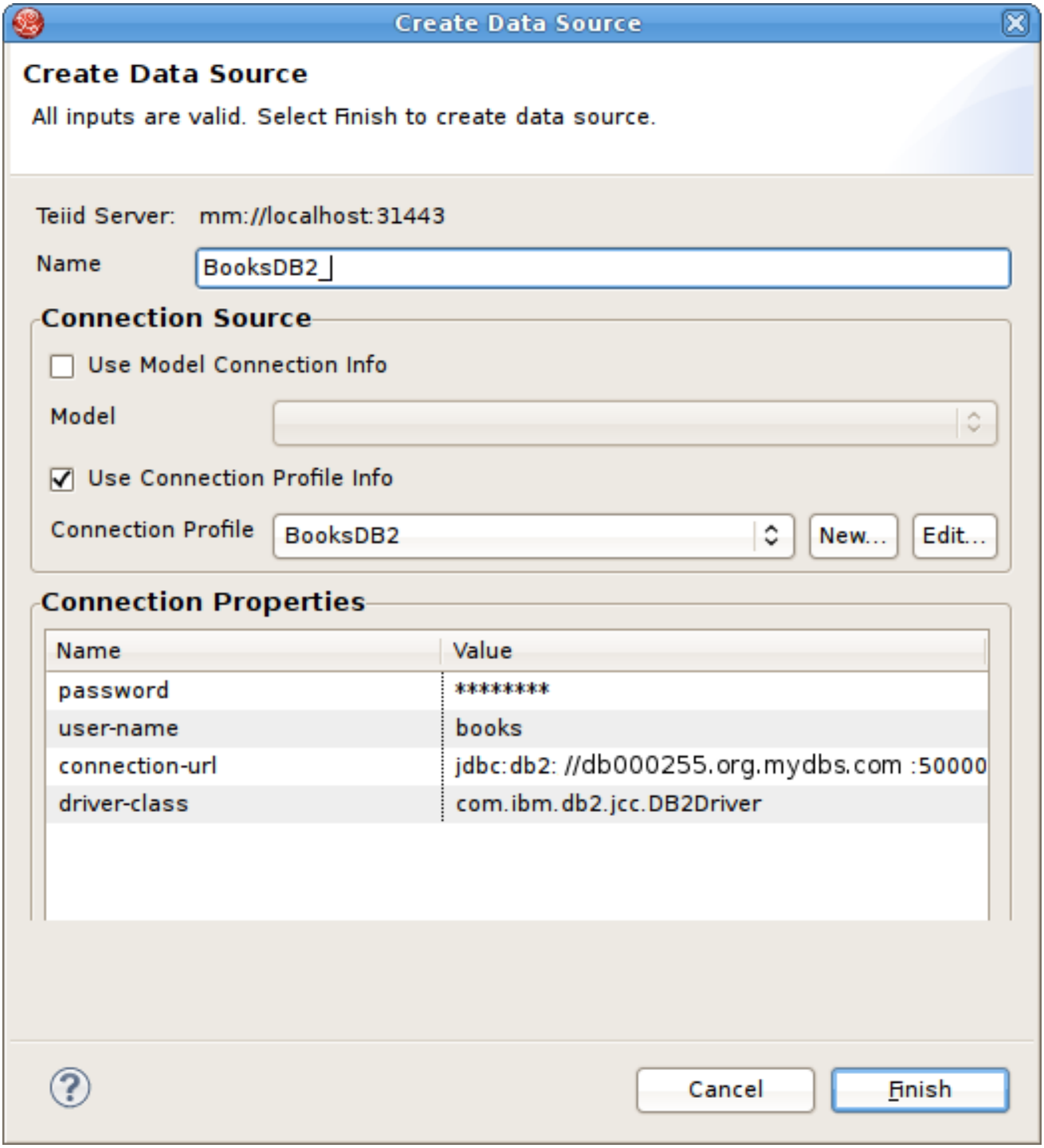


Figure 15.10. Create Data Source Dialog

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 JBoss Data Virtualization server.

15.3.3. Execute VDB from Model Explorer

If you have a JBoss Data Virtualization instance defined and connected in your Server view you can:

1. Right-click a VDB in your Model Explorer select **`Modeling > Execute VDB`** action. This action will insure your selected VDB is deployed to JBoss Data Virtualization, create a Teiid Connection Profile specific for that VDB, open the Database Development perspective and create a connection to your VDB.

Figure 15.11. Execute VDB Action

2. In the **`SQL Scrapbook`**, enter your designer SQL (i.e. **`SELECT * FROM TableXXXX`**), select all text and right-click select **`Execute Selected Text`**.

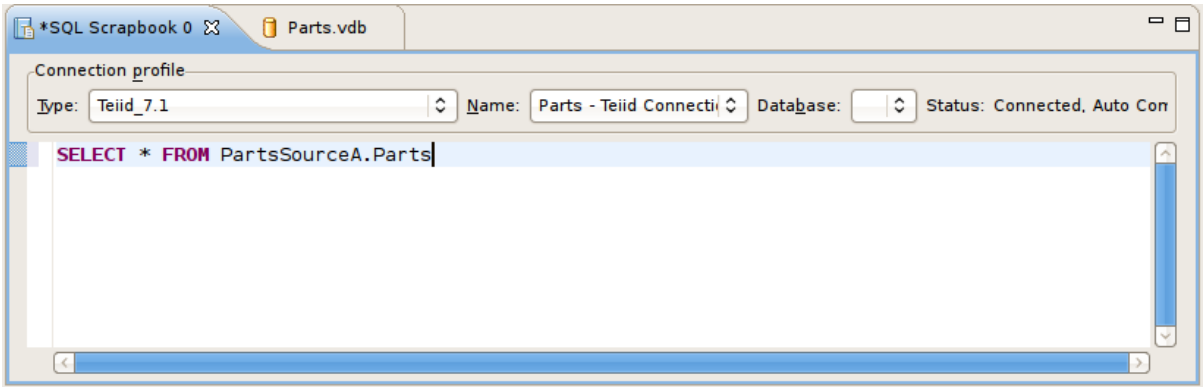


Figure 15.12. SQL Scrapbook Editor

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

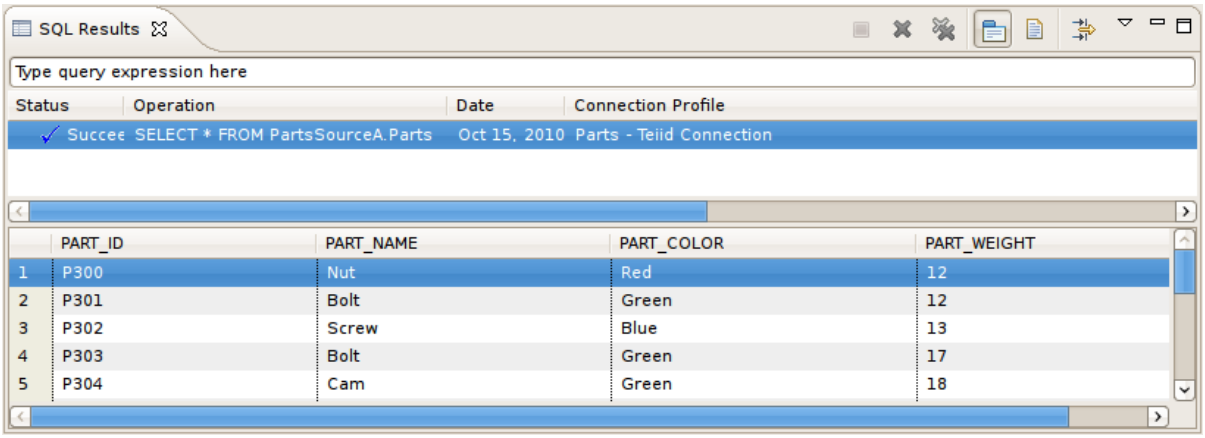


Figure 15.13. SQL Results View

15.3.4. 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 JBoss Data Virtualization instance in the Server view, or right-click select **`Modeling > Deploy`** action.

Once deployed, you can select the VDB in the Server 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

If you do not have a JBoss Data Virtualization instance defined or your default JBoss Data Virtualization instance is disconnected, the following dialog will be displayed if the **`Modeling > Deploy`** action is launched.

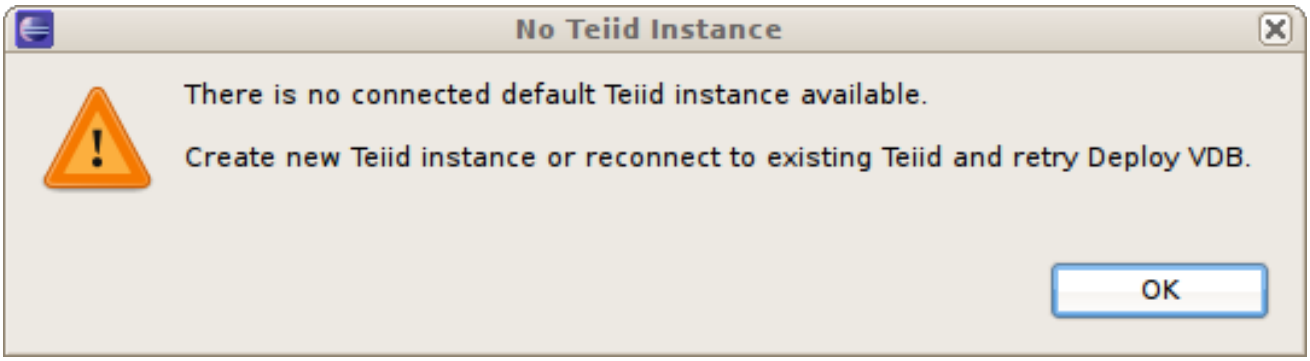


Figure 15.14. No Teiid Instance Defined

15.3.5. Executing a Deployed VDB

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

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

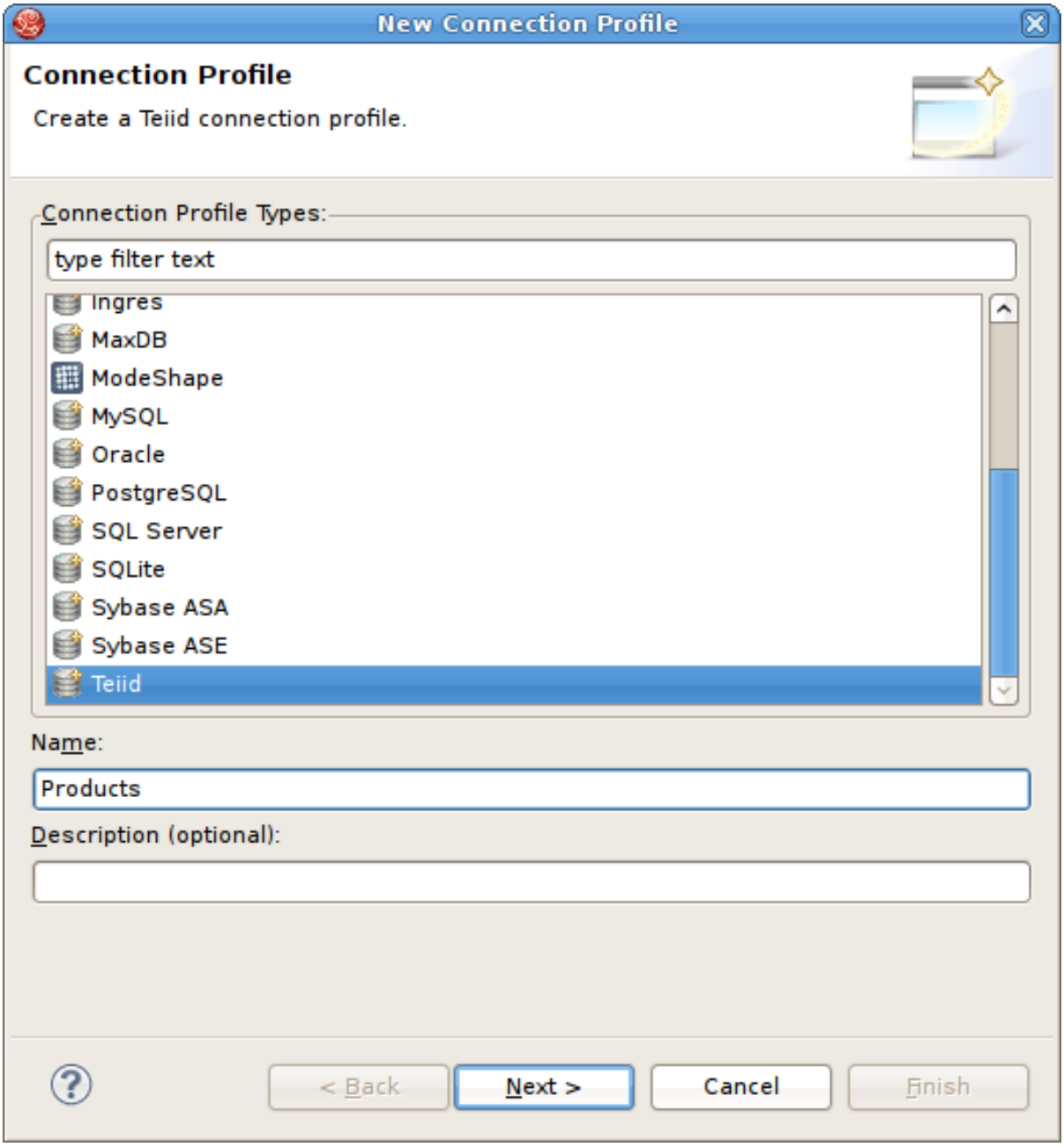


Figure 15.15. New Connection Profile Dialog

3. Enter unique name for your profile, select an existing connection profile type and hit **`Next`**.

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).

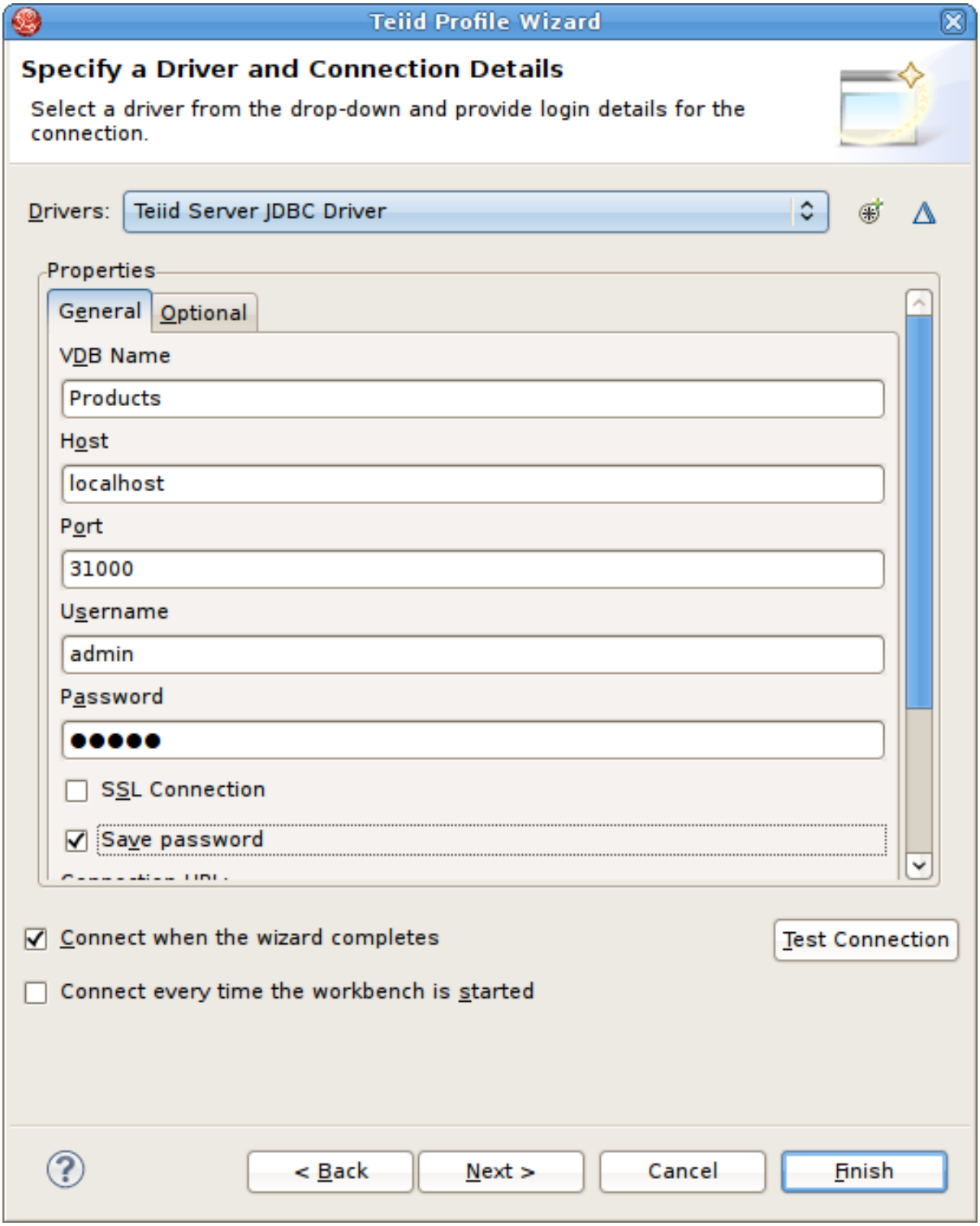


Figure 15.16. Teiid Connection Profile Dialog

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

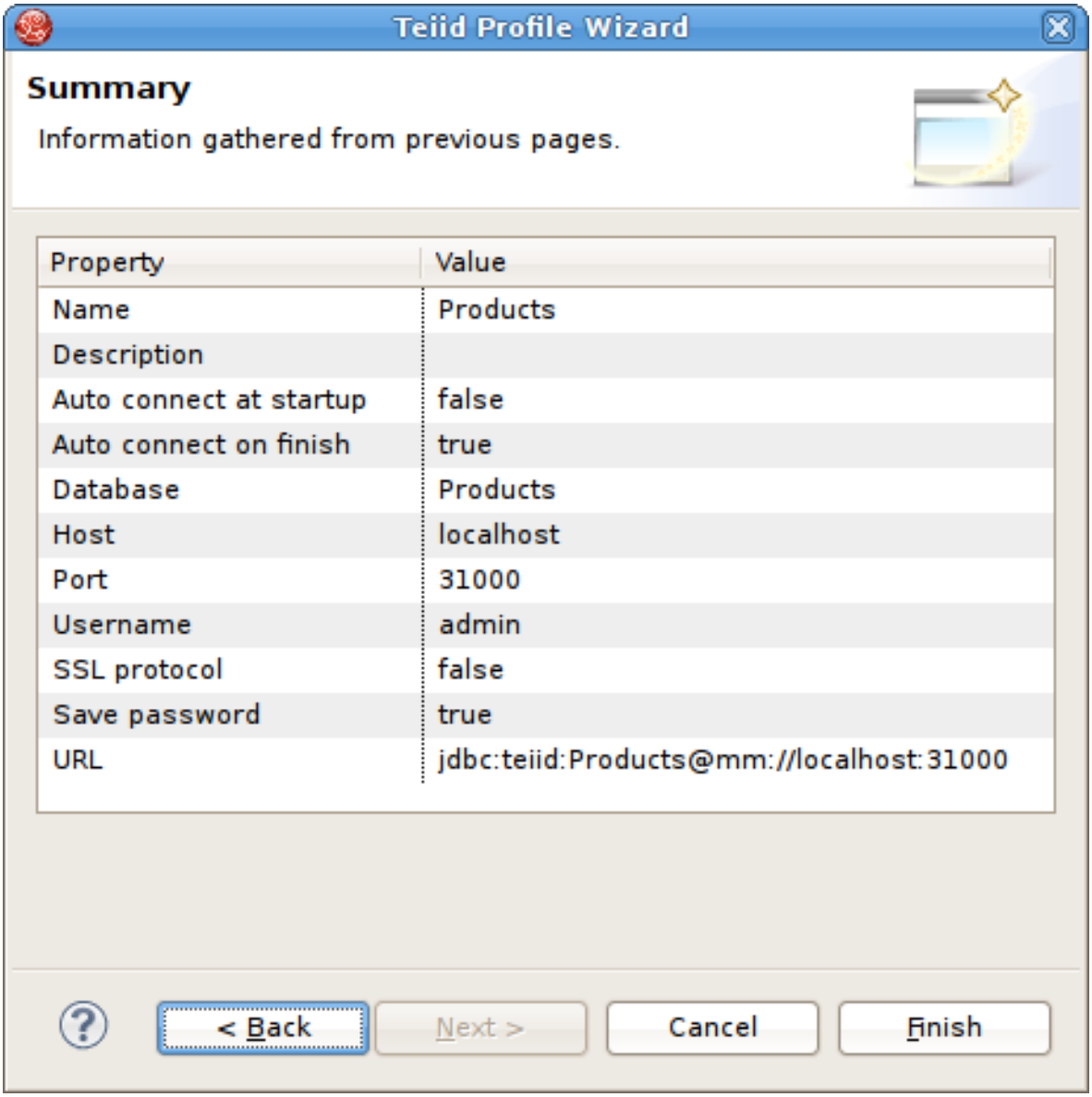


Figure 15.17. Teiid Connection Profile Summary

6. Select **`Finish`**.

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`**.

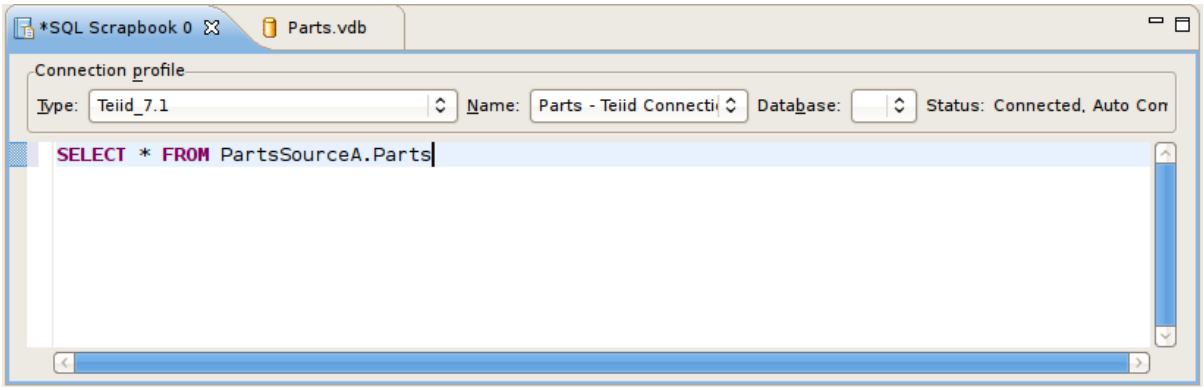


Figure 15.18. SQL Scrapbook Editor

CHAPTER 16. SEARCHING

16.1. SEARCHING

Designer provides multiple search actions located via Teiid Designer submenu in Eclipses Search menu.

The individual actions in the Teiid Designer submenu 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 Search Results 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.

16.2. FINDING MODEL OBJECTS

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

To find a model object:

1. Open the **`Find Model Object`** dialog by selecting the action through the main menu's **`Search > Teiid Designer > Find Model Object`** action.
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.
3. If more than one object exists with the desired name, select the one of the locations.
4. Click **`OK`**. If editor is not open for the object's model, an editor will open. The desired object results in a diagram (if applicable) and selected.

16.3. SEARCH TRANSFORMATION SQL

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

To search for string values in your transformations SQL:

1. Click **`Search > Teiid Designer > Transformations...`** action on the main toolbar which which opens the **`Search Transformations`** dialog.
2. Specify a string segment in the **`Find:`** field and specify/change your case sensitive preference.
3. Select **`Perform Search`** button. Any transformation object containing SQL text which contains occurrences of your string will be displayed in the results section.

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

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.

16.4. SEARCH MODELS VIA METADATA PROPERTIES

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

To search your models using metadata:

1. Click **`Search > Teiid Designer > Transformations...`** action on the main toolbar which opens the **`Search`** dialog.

2. Specify desired search options for Criteria, Object Type, Data Type and Scope.

3. Click **`Search`**. The search will be performed and the results will be displayed in the**`Search Results View`**. If the view is not yet open, it will be opened automatically.

APPENDIX A. SUPPORTED CONFIGURATIONS

A.1. SUPPORTED DATA SOURCES AND TRANSLATORS

The following table provides a list of data sources and translators that are supported by Red Hat.

Table A.1. Supported Data Sources and Translators

Data Source	Translator	Driver
Apache Accumulo 1.6.5	accumulo	-
Apache Cassandra 2.2.4	cassandra	-
Apache Hive	hive	v2.0 Hive JDBC driver
Apache Solr 4.9.0	solr	-
Cloudera Hadoop	-	-
EDS 5.5.1	teiid	current Teiid Driver
Files – delimited, fixed length	file	-
Generic Datasource-JDBC ansi	jdbc-ansi	-
Generic Datasource-JDBC simple (postgresql84)	jdbc-simple	postgresql 8.4
Google Spreadsheet	google-spreadsheet	-
Greenplum 4.1	greenplum	postgresql 9.0
Hortonworks Hadoop	-	-
Hortonworks Data Platform 2.4	hive	1.2.1
HPE Vertica 7.2.1	vertica	7.2.1-0
HSQL (for test/examples only)	-	-
IBM DB2 9.7	db2	Universal Driver v4.x
IBM DB2 10	db2	Universal Driver v4.x
IBM DB2 11.1	db2	Universal Driver v4.x
IBM Informix 12.10	informix	4.10.JC5DE
Ingres 10	ingres	4.1.4

Data Source	Translator	Driver
Intel Hadoop	-	-
JBoss Data Grid 6.4 (remote client - hotrod)	infinispan-cache-dsl	Hot Rod Client
JBoss Data Grid 6.4 (library mode)	infinispan-cache	-
JBoss Data Grid 7.1.0 (hotrod)	infinispan-hotrod	Hot Rod Client
LDAP/ActiveDirectory v3	ldap	-
Mainframe (CICS,IMS,VSAM)	-	-
MariaDB	mysql5	mysql 5.1.22
ModeShape/JCR 3.1	modeshape	3.8.4
MongoDB 2.2	mongodb	-
MongoDB 3.4	mongodb	-
MS Access 2010	-	-
MS Access 2013	-	-
MS Excel 2010	excel	-
MS Excel 2013	excel	-
MS SQL Server 2008	sqlserver	Microsoft SQL Server JDBC Driver 4
MS SQL Server 2012	sqlserver	Microsoft SQL Server JDBC Driver 4
MS SQL Server 2014	sqlserver	Microsoft SQL Server JDBC Driver 4
MySQL 5.1	mysql5	v5.1
MySQL 5.5	mysql5	v5.5
Netezza 6.0.2	netezza	-
Oracle 10g R2	oracle	Oracle JDBC Driver v10
Oracle 11g R2	oracle	Oracle JDBC Driver v12
Oracle 12c	oracle	Oracle JDBC Driver v12

Data Source	Translator	Driver
PostgreSQL 8.4	postgresql	postgresql 8.4
PostgreSQL 9.2	postgresql	postgresql 9.2
PostgreSQL 9.6	postgresql	JDBC4 PostgreSQL Driver, Version 42.2.2.jre6
PostgreSQL 10.1	postgresql	JDBC4 PostgreSQL Driver, Version 42.2.2.jre6
REST/JSON over HTTP	ws	-
Red Hat Enterprise Linux 5.5/6 PostgreSQL config	-	-
Red Hat Directory Server (RHDS) 9.0	ldap	-
Red Hat JBoss Data Virtualization	teiid	current Teiid Driver
Salesforce.com API 22	salesforce	-
SAP Netweaver Gateway (OData)	sap-nw-gateway	-
Support SAP Service Registry as a Data Source	-	-
SAP HANA v1.0 SP 10+	hana	ngdbc.jar v1.00 in-memory JDBC driver
SAP IQ 16	sybaseiq	jconn4 (JDBC 4.0)
Sybase ASE 15	sybase	jconn4 (JDBC 4.0)
Teradata Express 12	teradata	-
Webservices	ws	-
XML Files	FILE	-

NOTE

MS Excel is supported in so much as there is a write procedure.

A.2. DESIGNER METADATA USAGE REQUIREMENTS IN JBOSS DATA VIRTUALIZATION RUNTIME

Based on the metadata exposed by the Teiid Designer the below table shows which fields are required and how that information is being used in JBoss Data Virtualization runtime.

Table A.2. 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,XmlMappingClass,XmlStagingTable, MaterializedTable
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
SupportsUpdate	Boolean	Yes	Yes	Yes	To allow updates on the table

TABLE	Type	In Designer	In Metadata API	Required	Description
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
deleteEnabled	Boolean	**	Yes	Yes	Flag for checking delete procedure is enabled for view

TABLE	Type	In Designer	In Metadata API	Required	Description
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 A.3. 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
IsSelectable	Boolean	Yes	Yes	Yes	Column is allowed in select

COLUMN	Type	In Designer	In Metadata API	Required	Description
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	??	??	

COLUMN	Type	In Designer	In Metadata API	Required	Description
DefaultValue	String	Yes	Yes	Yes	Used for Insert and procedure execute operations when the values are not supplied
Position	Integer	Yes	Yes	Yes	Used in the index calculations

Table A.4. 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 A.5. 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 A.6. 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 A.7. 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 A.8. Data Usage for Result Sets

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

Table A.9. 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				

FOREIGN KEY	Type	In Designer	In Metadata API	Required	Description
UniqueKeyID	String				

Table A.10. 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
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

DATA TYPE	Type	In Designer	In Metadata API	Required	Description
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
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 A.11. Data Usage for Procedures

PROCEDURE	Type	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the column

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

Table A.12. 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

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

Table A.13. Data Usage for SQL Transformations

SQL TRANSFORMATION(**)	Type	In Designer	In Metadata API	Required	Description
VirtualGroupName	String	Yes	No	Yes	See Table, the properties defined on Table

SQL TRANSFORMATION(**)	Type	In Designer	In Metadata API	Required	Description
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 A.14. 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
Identifier	String	Yes	No	No	Not required
Description	String	Yes	vdb.xml	No	Used by System metadata
ProducerName	String	Yes	No	No	Not required

VDB	Type	In Designer	In Metadata API	Required	Description
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 A.15. 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 B. USER PREFERENCES

B.1. USER PREFERENCES

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

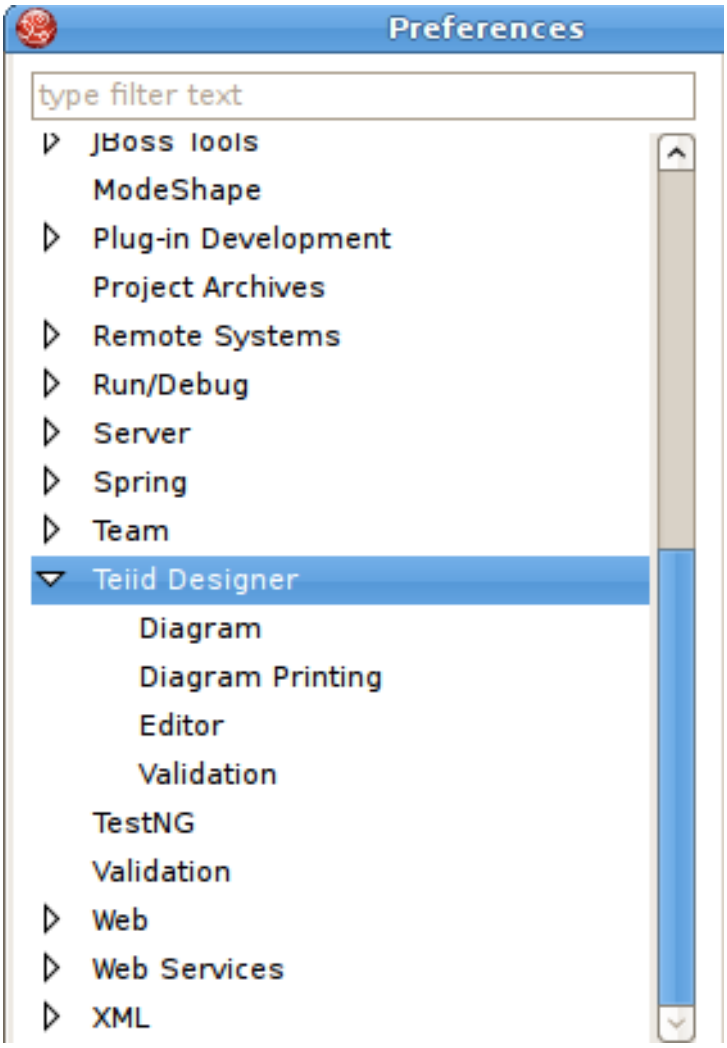


Figure B.1. Preferences Dialog

B.2. TEIID DESIGNER PREFERENCES

General **`Teiid Designer`** preferences include:

- **`Enable auto-creating of a source model's data source on Teiid Instance`** - indicates if data sources that match the default name should be auto-created if they do not exist on Teiid server.
- **`Default Teiid Server Version`** - determines what version of server modelling will be targeted at, if no 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.
- **`Always open editor without prompting`** - To change/edit a model, it must be opened

for editing. Selecting this box will automatically open the model in an editor if the user attempts to perform a change in a model. If unchecked, the user will be informed that an editor will be opened before the operation is completed.

- **`Open Designer perspective when model is opened`** - If a model is opened via importing projects, the **`New > Teiid Metadata Model`** menu and the Teiid Designer perspective is not open, you may want to automatically open the perspective and begin working on your model. This preference has 3 settings. Always open, which means always open the perspective without prompting; never open, which means do not open the Teiid Designer perspective, or prompt, which will always ask you if you wish to open the Teiid Designer perspective.

- **`Check and update imports during save`** - Occasionally editing a model may add or remove objects in one model that reference objects in another model. Model Imports keep track of these dependencies within each model. A validation error or warning may appear during a build. Checking this box will automatically check and update imports during the save process. This will result in any unneeded imports being removed from the model or any required imports added to the model. If unchecked, no updating of imports will be performed.

B.3. DIAGRAM PREFERENCES

Several diagram preferences are available to customize your diagrams.

- **`Routers`** - The relationship link type for Package and Custom diagrams (Foreign Key - Primary Key relationships) can be customized. Available options include Orthogonal (default), Directed 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 checkboxes labeled **`Show Role Names`** and **`Show Multiplicity`**.

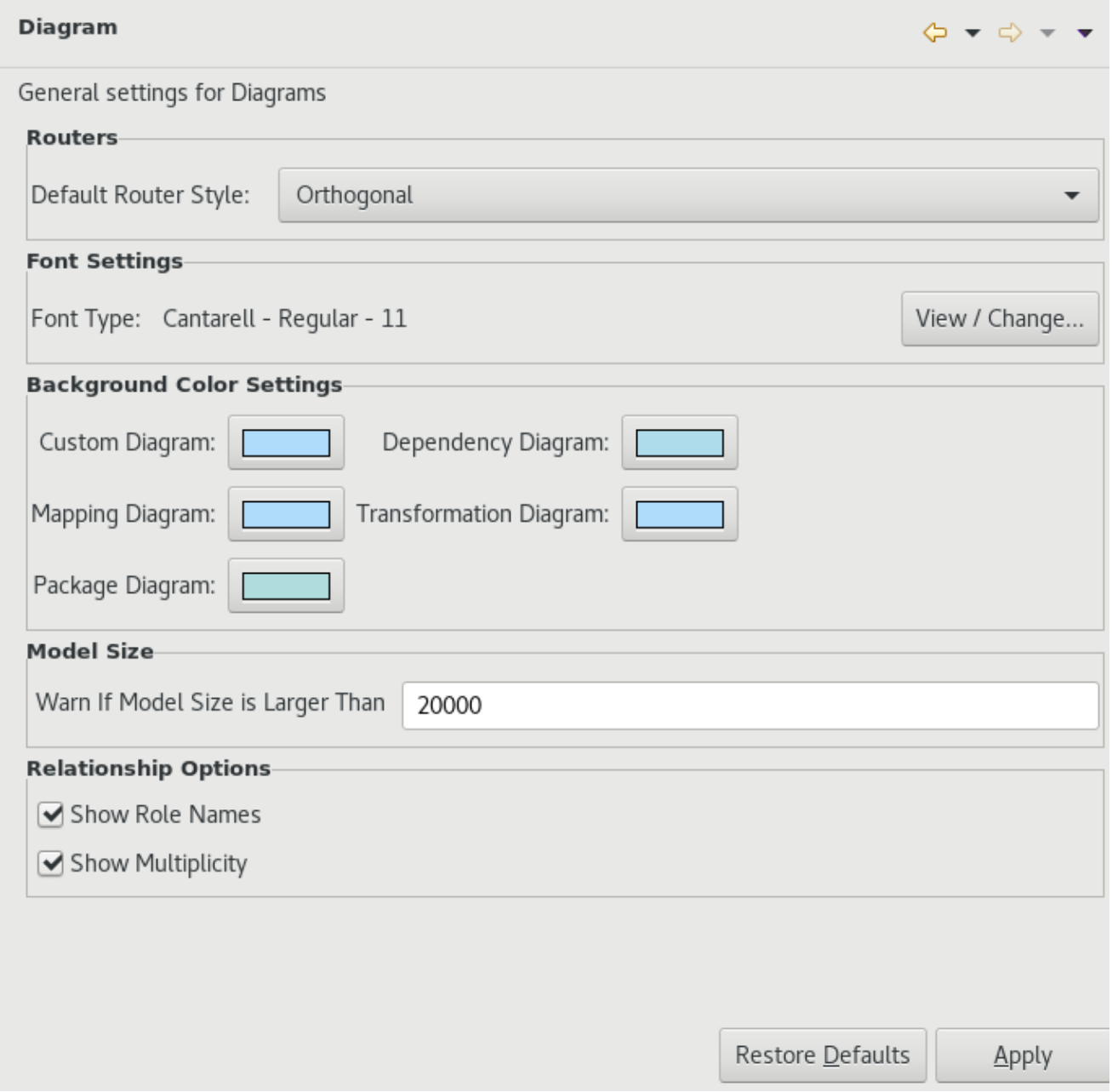
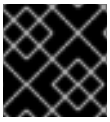


Figure B.2. Diagram Preferences Panel

B.4. EDITOR PREFERENCES

B.4.1. XML Document Preferences



IMPORTANT

Please note that XML Document Modeling has been deprecated.

`XML Document Mapping Preferences` provide ways to customize Mapping Diagram and Recursion Editor (XML) behavior.

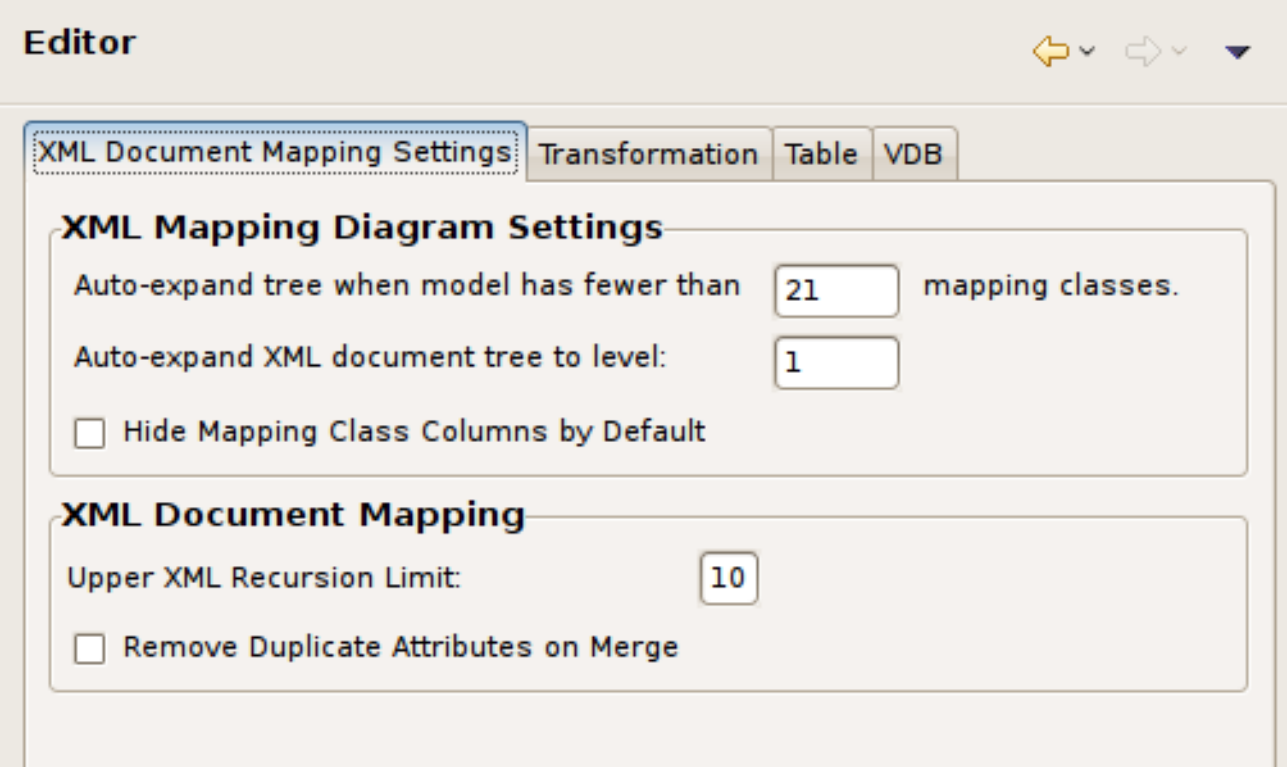


Figure B.3. XML Document Preferences Panel

B.4.2. Table Editor Preferences

`Table Editor Preferences` provide a way to customize the order and the information content for each model object type.

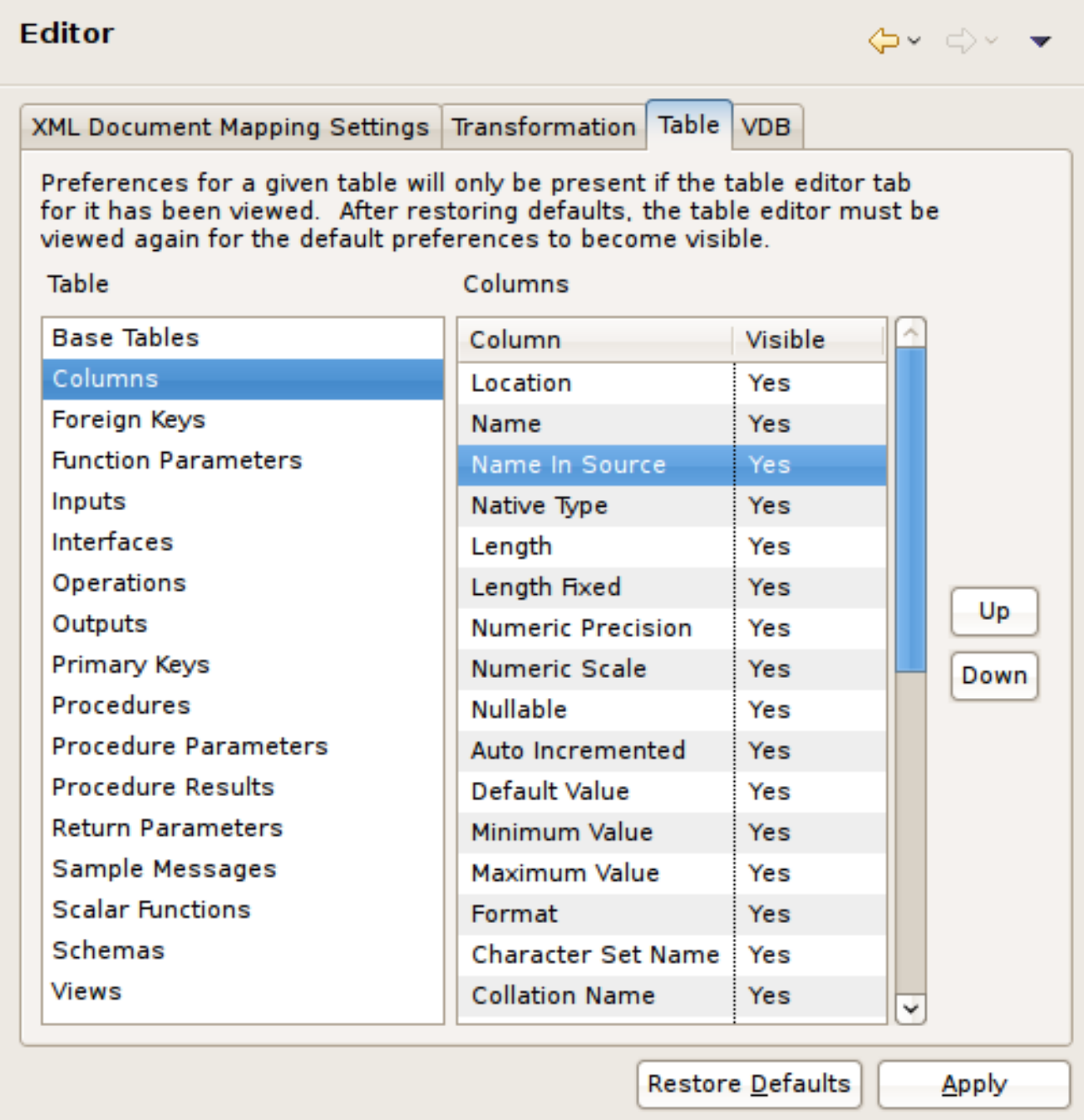


Figure B.4. Table Editor Preferences Panel

B.4.3. Transformation Editor Preferences

`Transformation Editor Preferences` provide a way to customize SQL formatting, diagram layout, and default view entity properties.

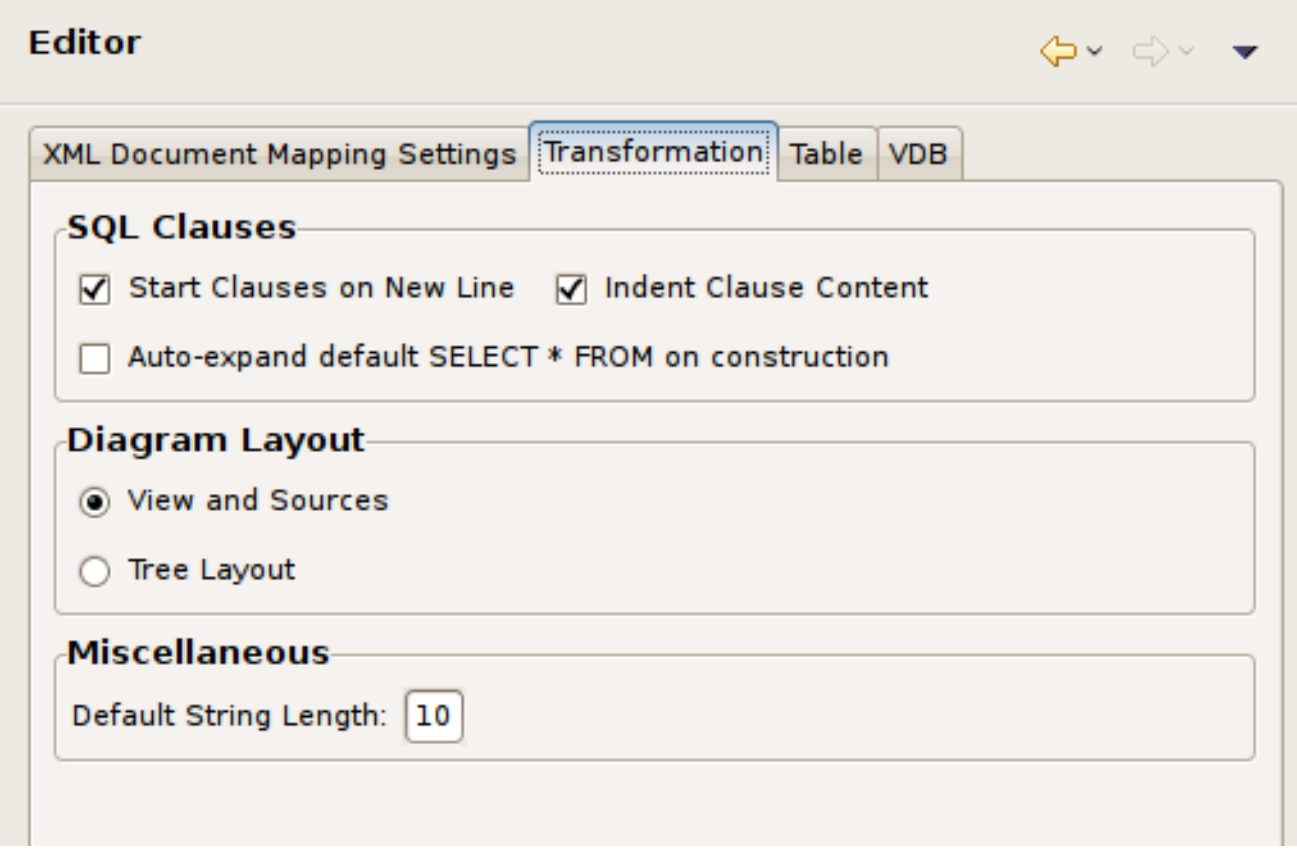


Figure B.5. Transformation Editor Preferences Panel

B.4.4. VDB Editor Preferences

`VDB Editor Preferences` provide a way to customize VDB editor behavior.

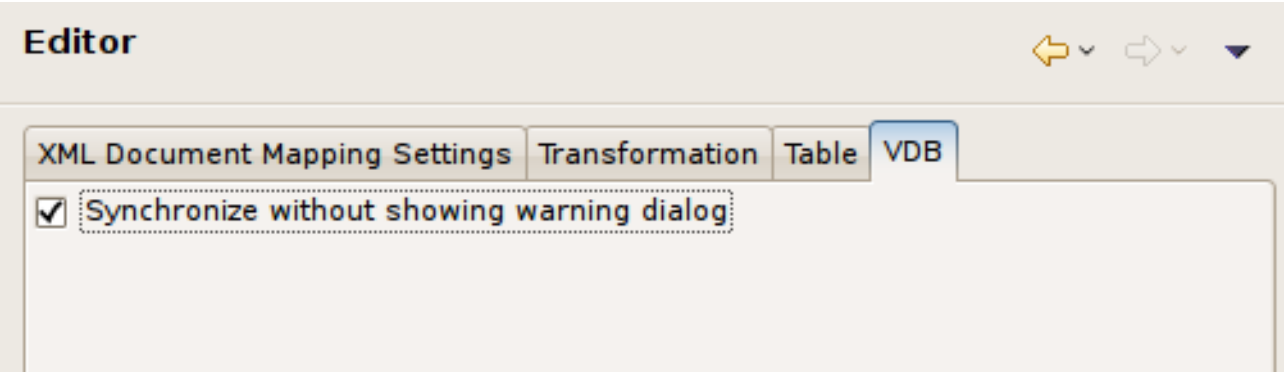


Figure B.6. VDB Editor Preferences Panel

B.5. 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.

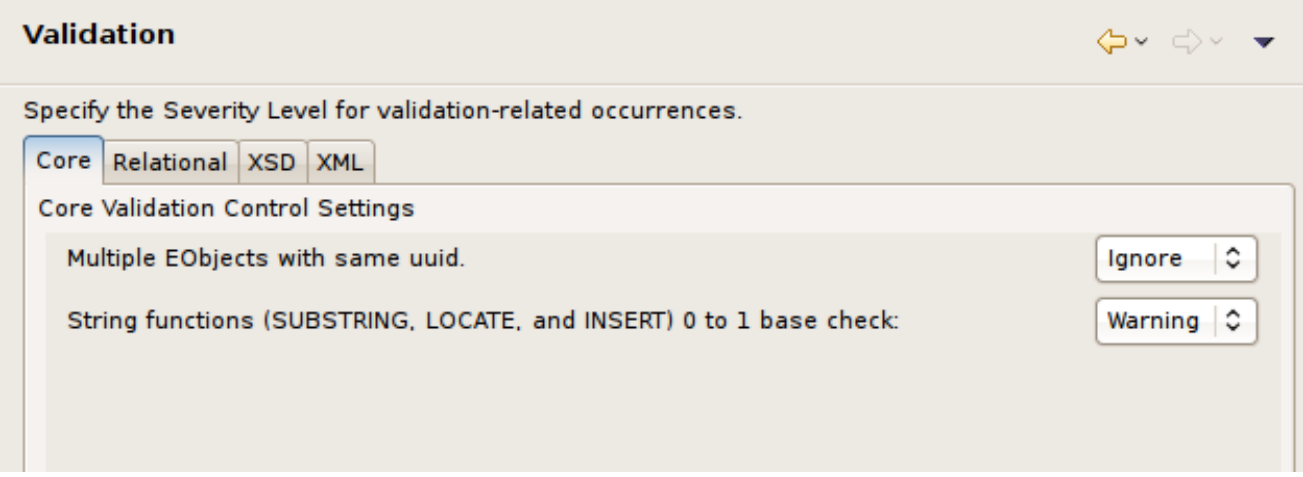


Figure B.7. Core Model Validation Preferences Panel

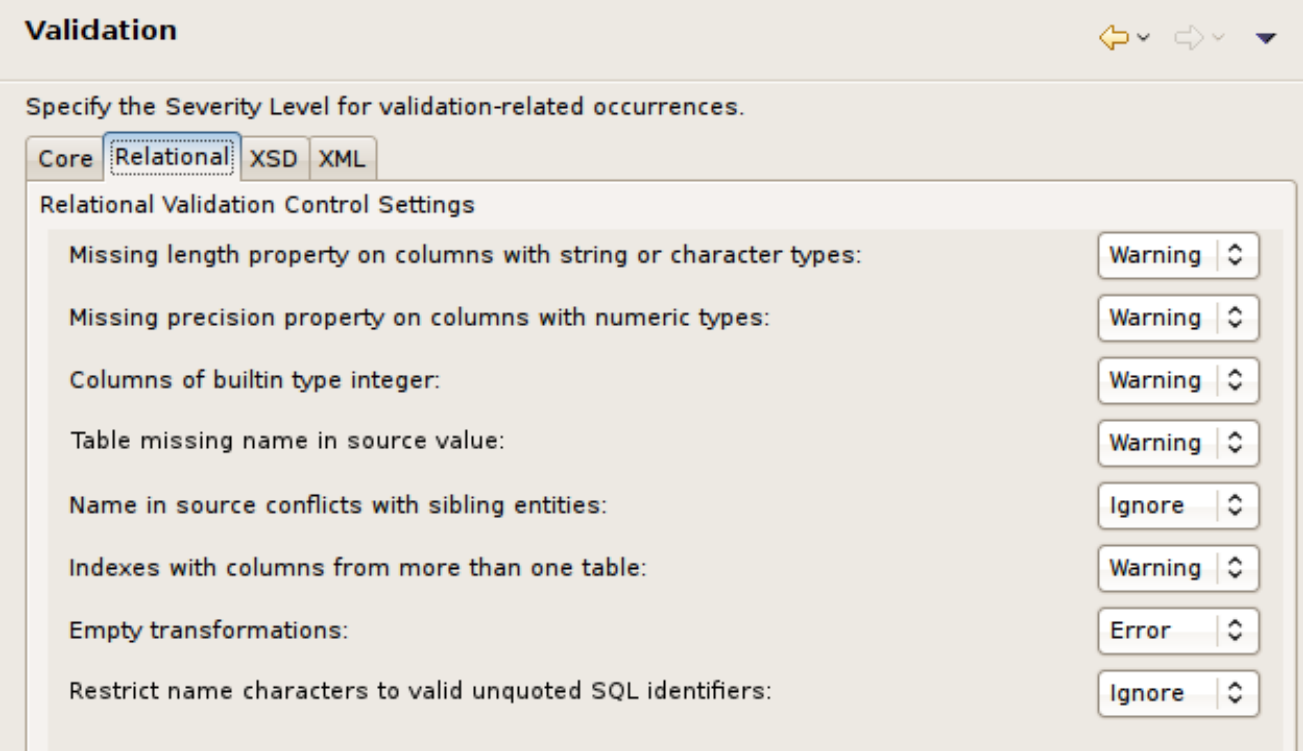


Figure B.8. Relational Model Validation Preferences Panel

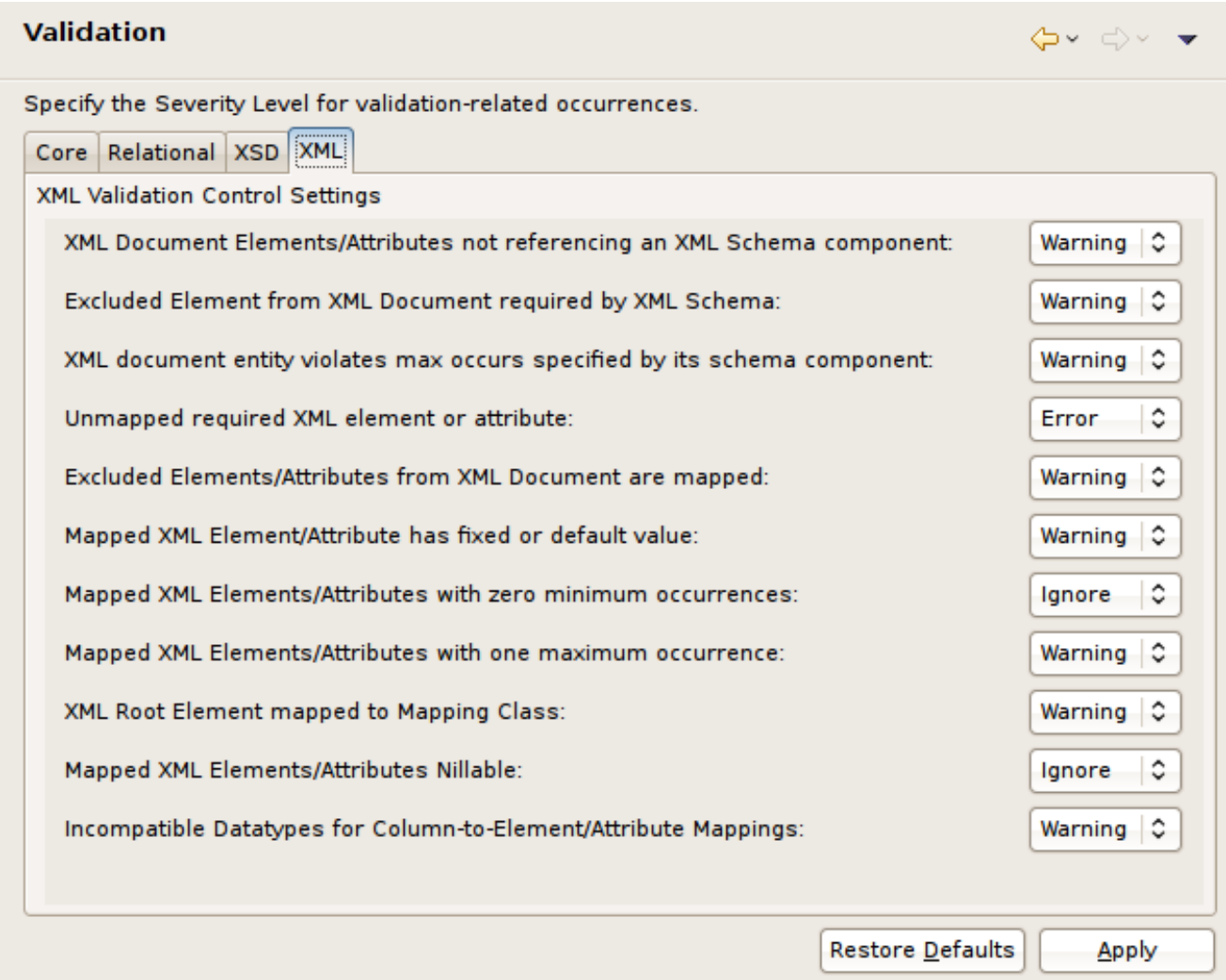


Figure B.9. XML Document Model Validation Preferences Panel

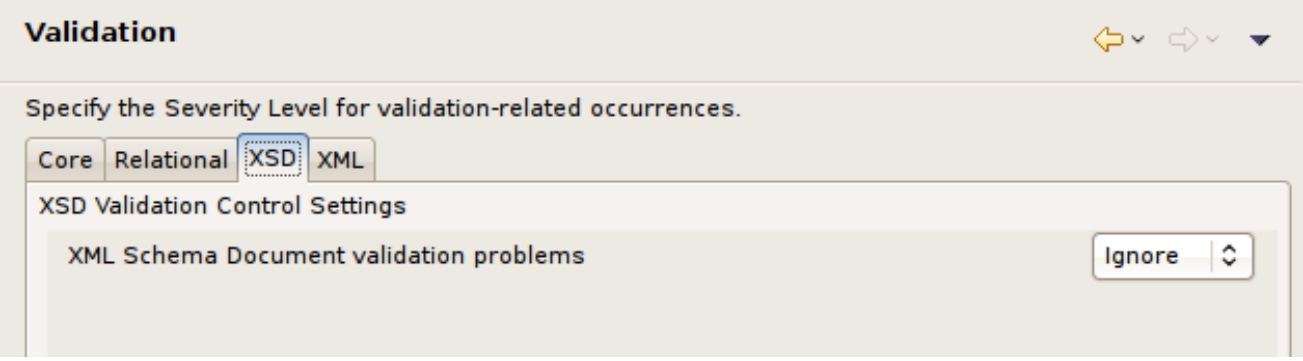
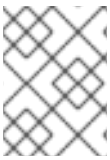


Figure B.10. XSD Schema Model Validation Preferences Panel



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 C. TEIID DESIGNER UI REFERENCE

C.1. TEIID DESIGNER PERSPECTIVES

Teiid Designer utilizes the Eclipse 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.

C.2. TEIID DESIGNER PERSPECTIVE

The **Teiid Designer** perspective provides access to fundamental model editing and management capabilities. This perspective includes the following main UI components or groups of components:

- Model Explorer View - Teiid tree view of Model Objects.
- Server View - JBoss Data Virtualization Server instance view. Provides view of contents for connected instances of installed Teiid runtime.
- Model Editors - Custom editors targeted for `.xmi` metadata model files.
- Properties View - Standard property values for selected workbench objects.
- Guides View - Guides and Status Views which enhance usability.
- Miscellaneous Views - Includes the Problems view, Error Log view and the SQL Results view (opened if Preview Data action is performed).
- Connections Views - Includes information about Connection Profiles, Data Sources and Default Server.

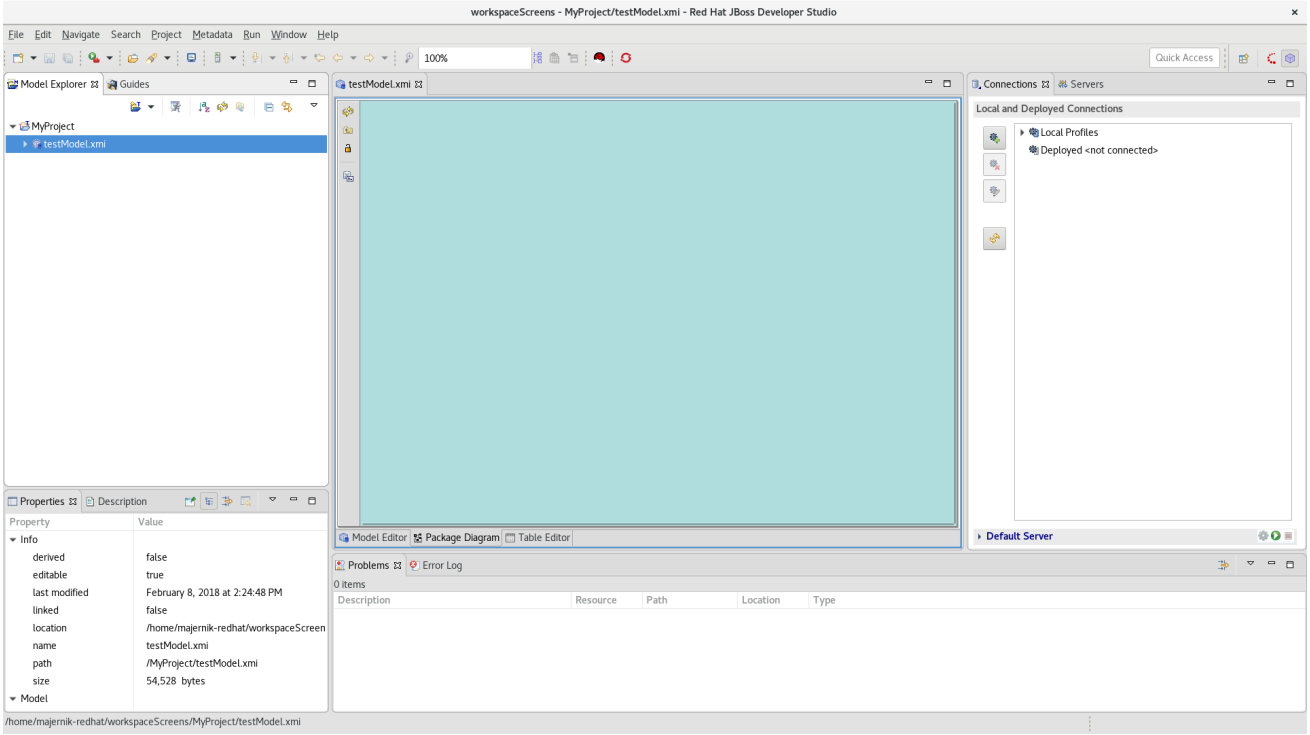


Figure C.1. Teiid Designer Perspective Layout

C.3. OPENING A PERSPECTIVE

There are two ways to open a perspective:

- Using the **`Open Perspective`** button on the shortcut bar.
- Selecting a perspective from the **`Window > Perspective > Open Perspective`** menu.

To open a perspective by using the shortcut bar button:

1. Click the **`Open Perspective`** button .
2. In the **`Select Perspective`** dialog, select **`Teiid Designer`** and click **`OK`**.

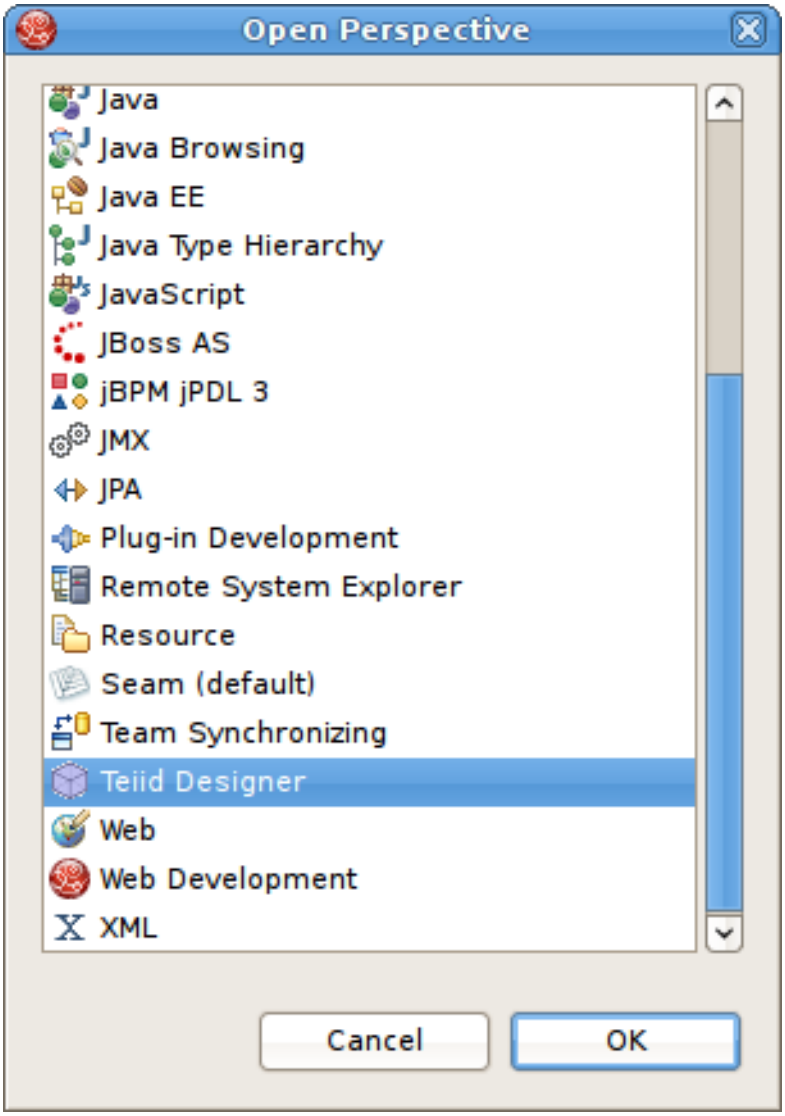


Figure C.2. Select Perspective Dialog

The **`Teiid Designer`** perspective is now displayed.

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

- 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 click **`Show Text`** and conversely click **`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.

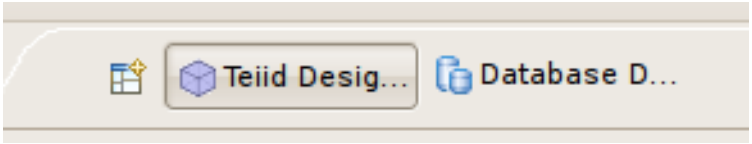


Figure C.3. Workbench Window Title Bar

C.4. FURTHER INFORMATION

For more details on perspectives, views and other Eclipse workbench details, see formal Eclipse Documentation.

APPENDIX D. TEIID DESIGNER VIEWS

D.1. TEIID DESIGNER VIEWS

Views are dockable windows which present data from your models or your modeling session in various forms. Some views support particular Model Editors 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.

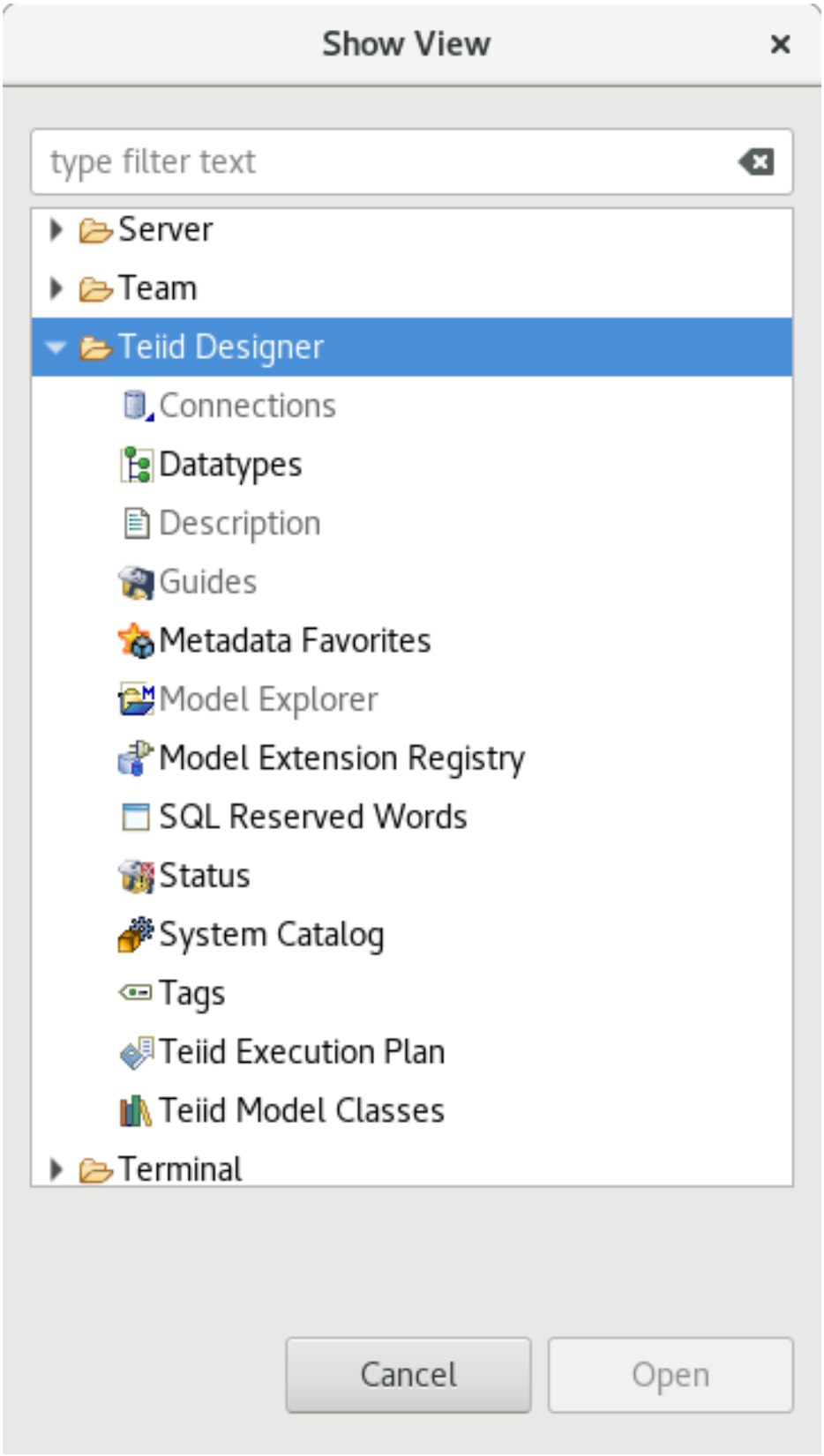


Figure D.1. Show View Dialog

D.2. 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.

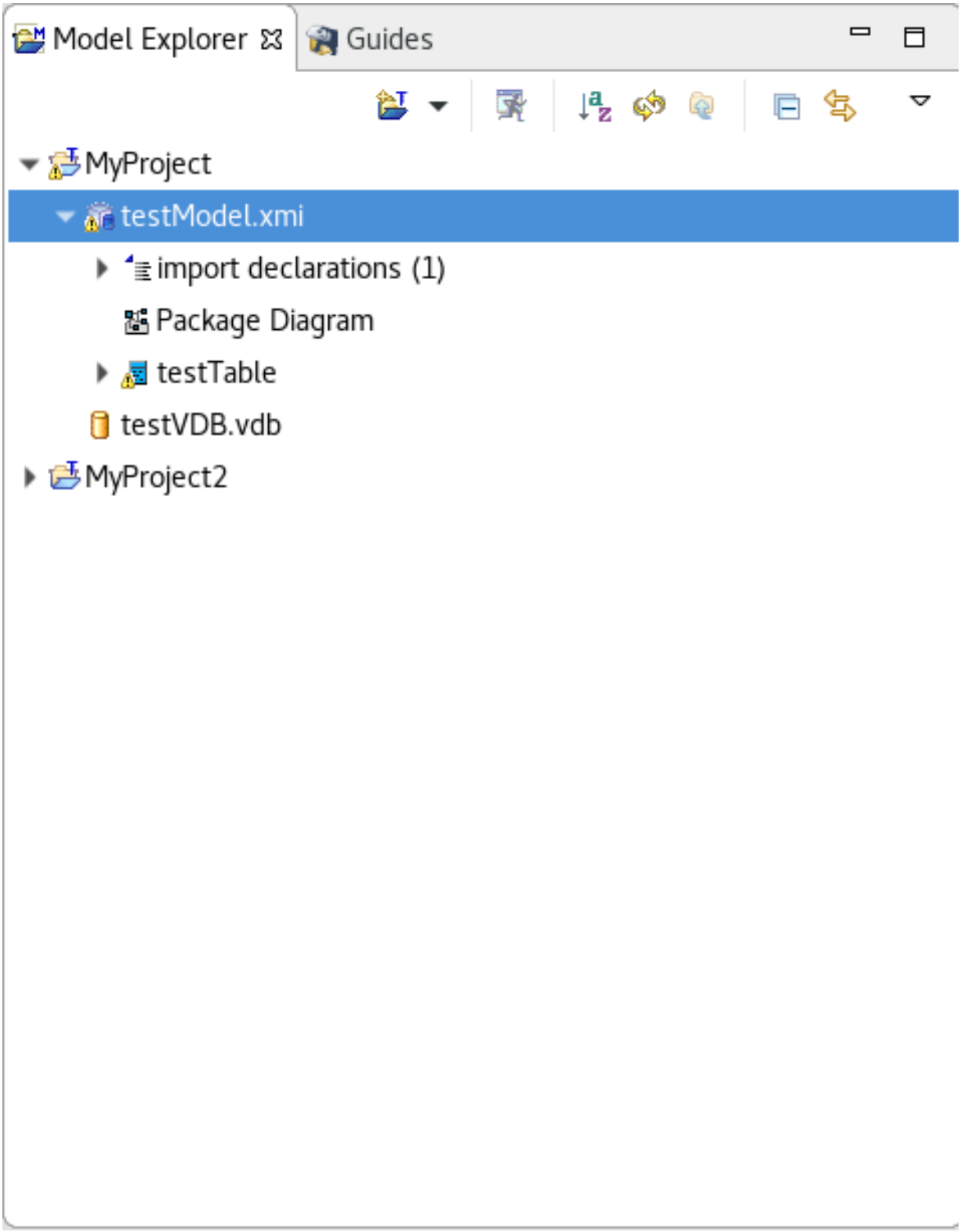


Figure D.2. Model Explorer View

The toolbar consists of nine common actions:

- Preview Data - Executes a simple preview query `(SELECT * FROM )`.
- Sort Model Contents - Sorts the contents of the models based on object type and alphabetizing.
- Refresh Markers - Refreshes error and warning markers for objects in tree.

- 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:

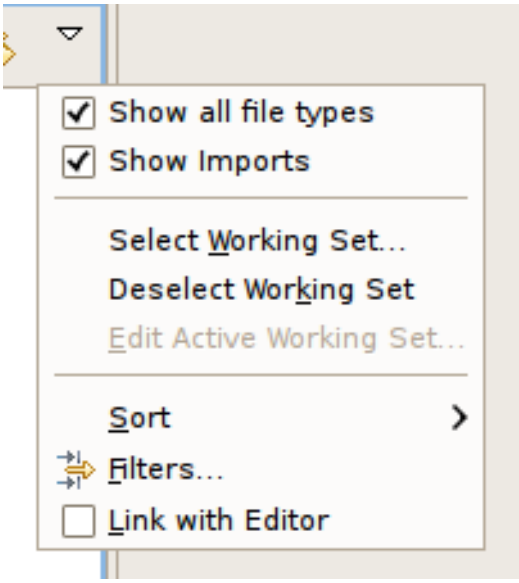


Figure D.3. Additional Actions

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

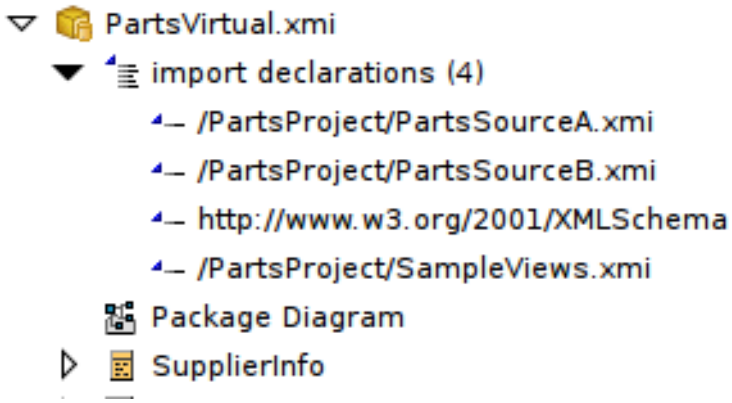


Figure D.4. Show Model Imports Action

D.3. SELECTION-BASED ACTION MENUS

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

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

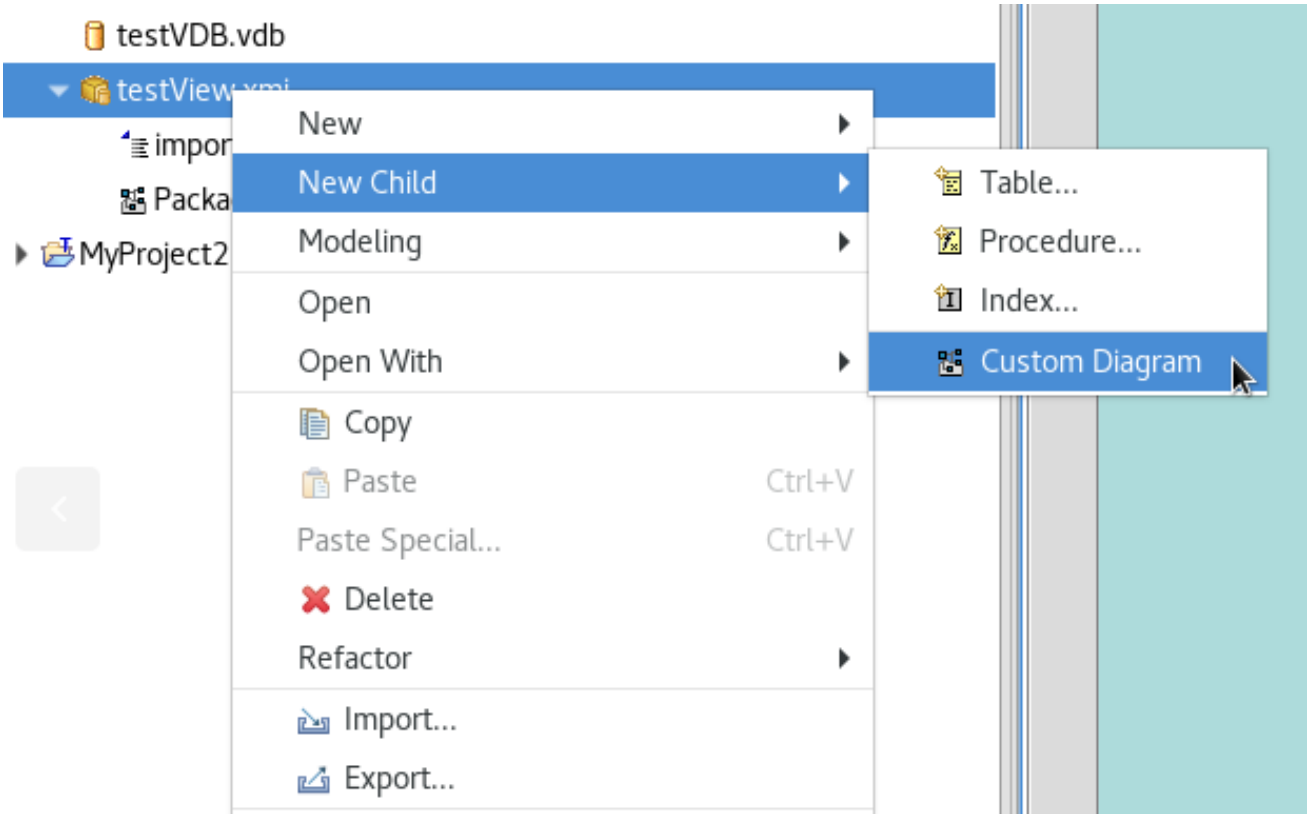


Figure D.5. Sample Context Menu

D.4. 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 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.5. 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.

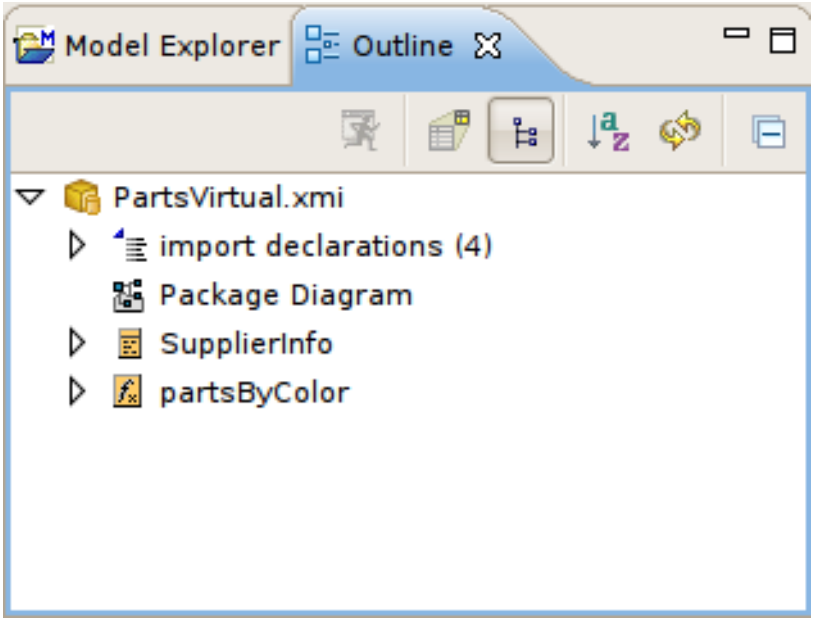


Figure D.6. Outline View

D.6. 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`**.

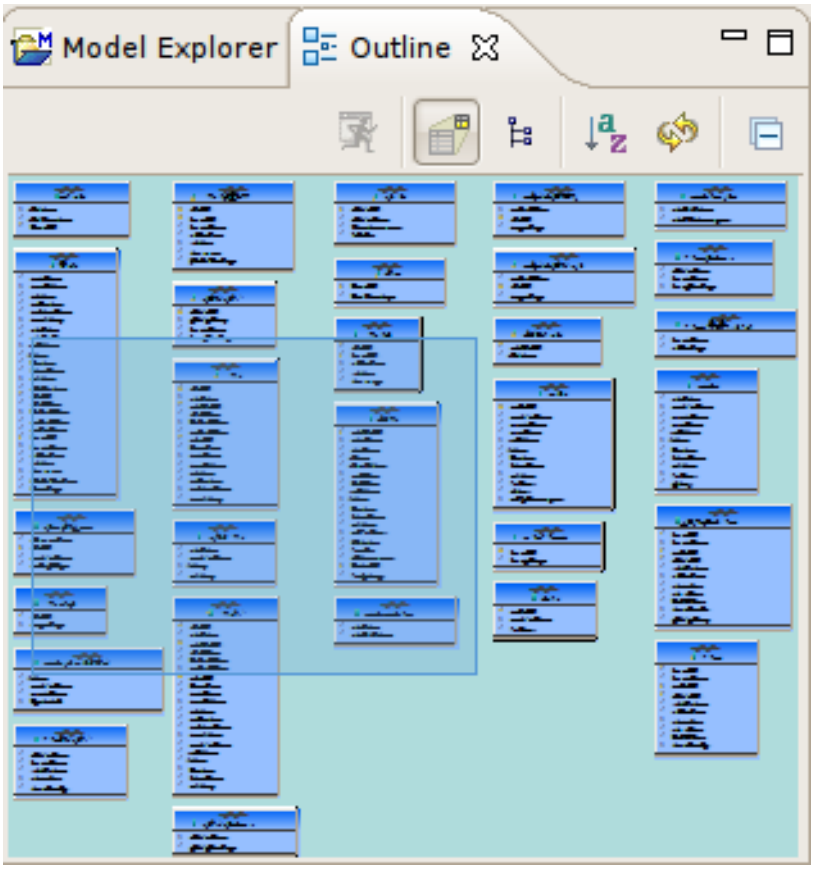


Figure D.7. Outline View

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

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

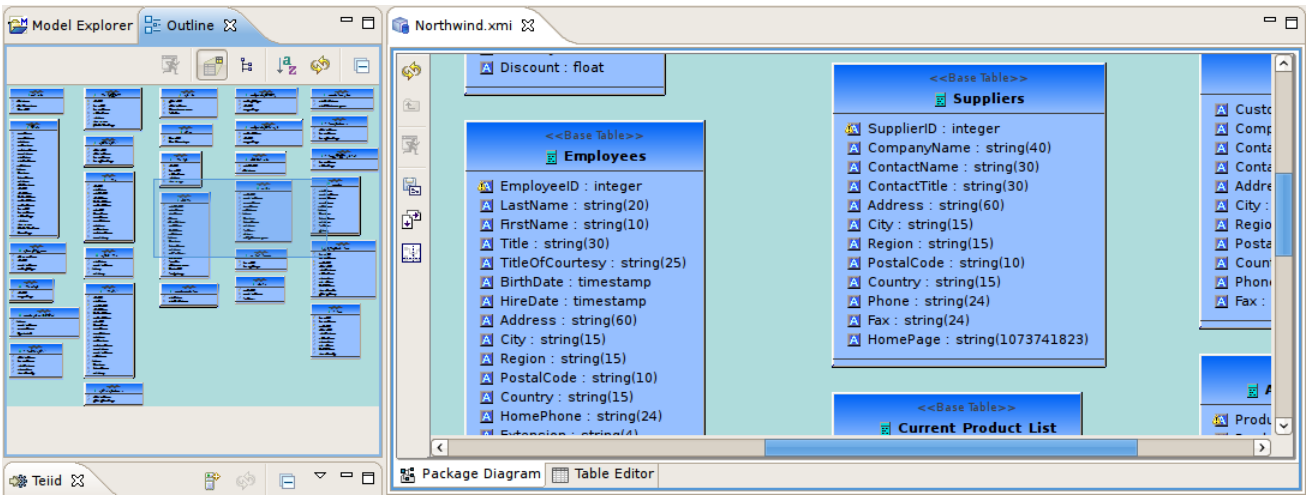


Figure D.8.

D.7. SERVER VIEW

The Server View provides a means to display and manage server instances and their contents within the JBoss Developer Studio environment. Since Teiid Designer is installed as part of a JBoss Data Virtualization 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. Click **Server > Servers** view and click **OK**.

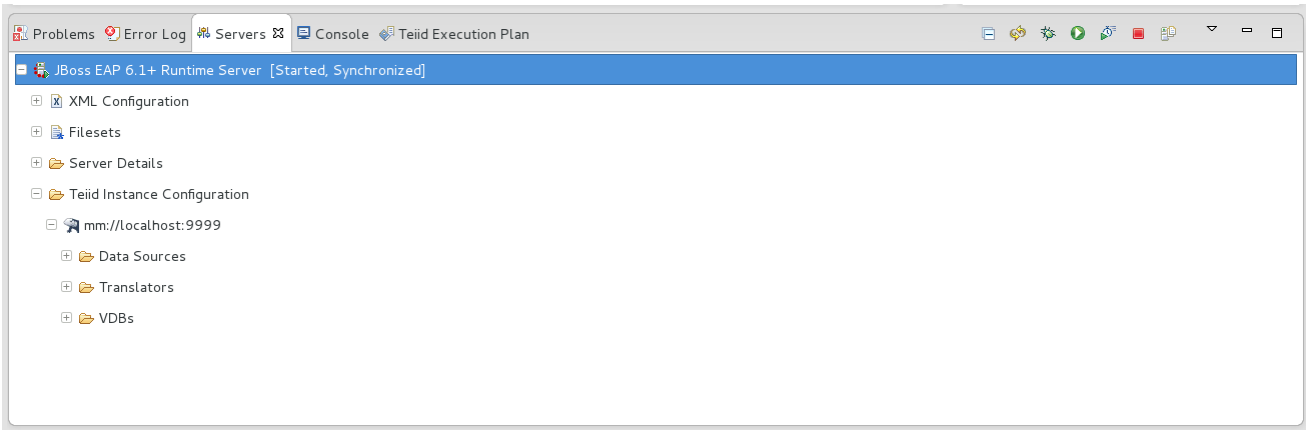


Figure D.9. Server View

To create your Teiid instance:

- Right-click and then select **New** > **Server** to launch the JBoss Data Virtualization Server wizard. Configuring the JBoss Data Virtualization Server instance (with Teiid installed) will enable connection to the Teiid Server.
- In the **New Server** wizard, select the Red Hat JBoss Enterprise Application Platform server type under the Red Hat JBoss Middleware category and click **Next >**.

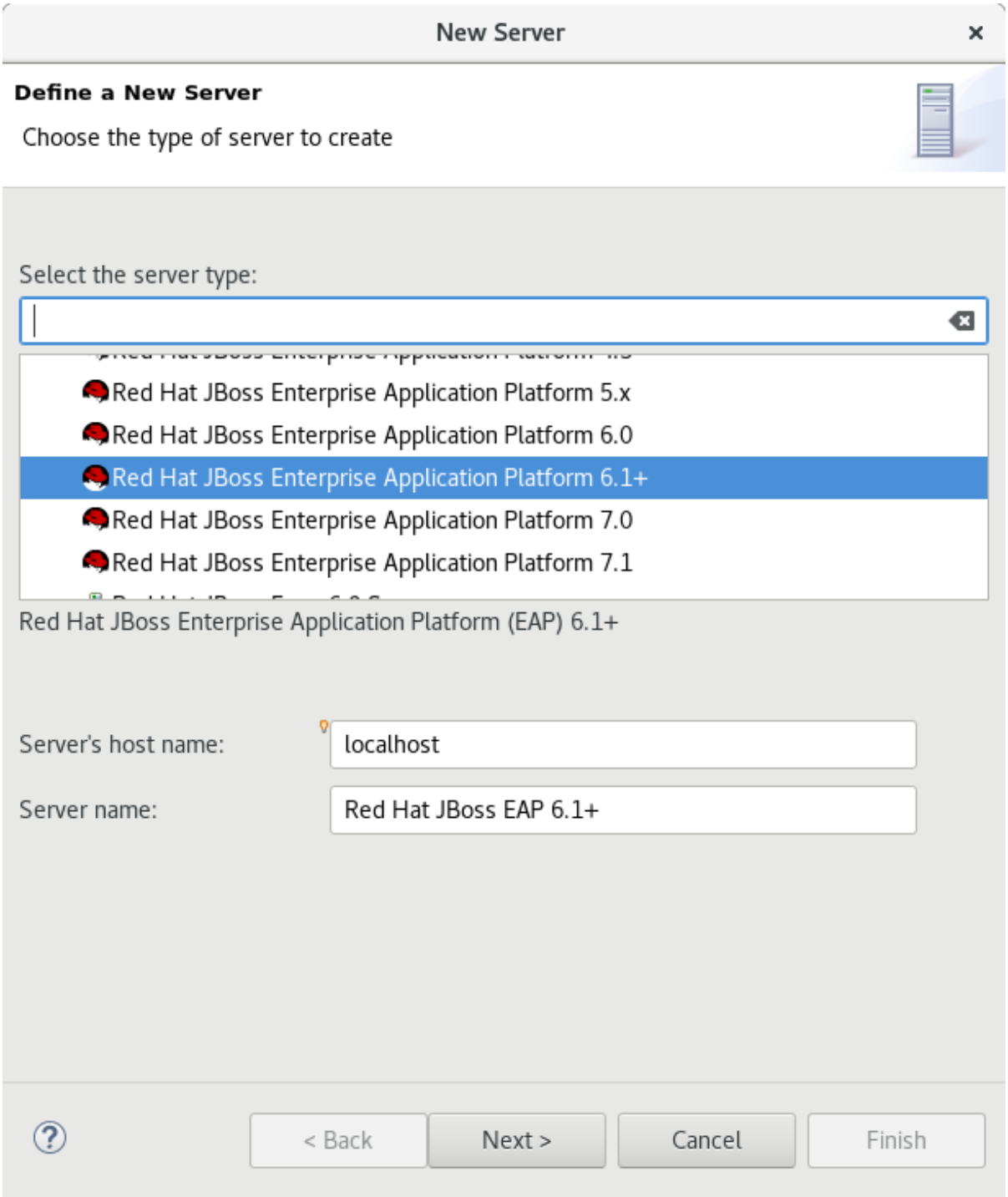


Figure D.10. New Server Dialog

- On the JBoss Runtime page, click the top **`Browse...`** button to select the installation folder of your JBoss Enterprise Application Platform server.

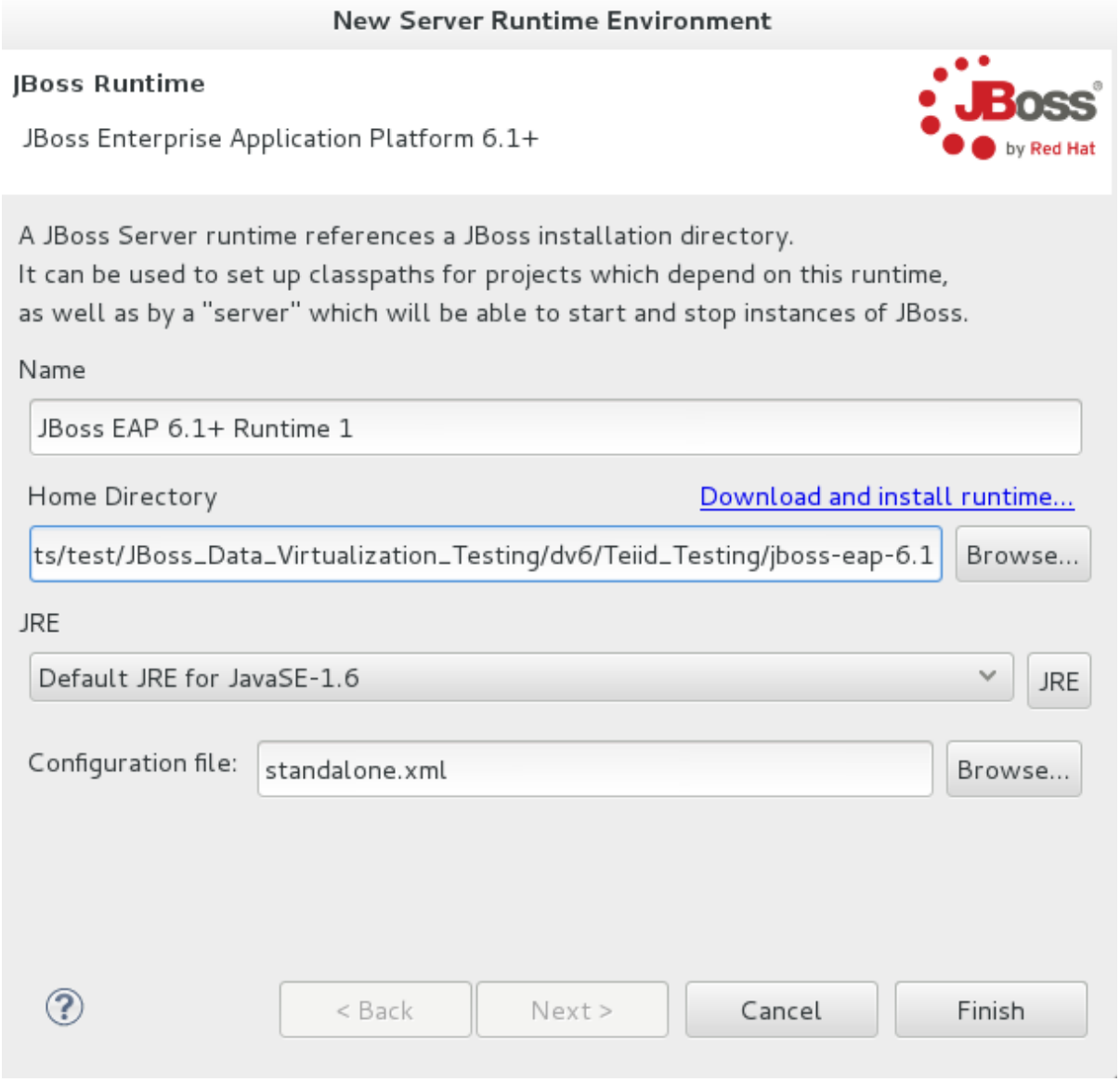


Figure D.11. JBoss Runtime Definition

- Then click the bottom **`Browse...`** button to select the **`standalone.xml`** configuration file located under the *EAP_HOME*/standalone/configuration/ folder on your file system. Then click **`Finish`**.

Name	Size	Modified
standalone_xml_history		Yesterday at 12:20
teiid-security-users.properties~	97 bytes	12/18/2013
teiid-security-users.properties	96 bytes	12/27/2013
teiid-security-roles.properties~	195 bytes	12/27/2013
teiid-security-roles.properties	189 bytes	12/27/2013
standalone-osgi.xml	28.7 kB	12/24/2013
standalone-ha.xml	31.6 kB	12/24/2013
standalone-full-ha.xml	38.0 kB	12/24/2013
standalone-full.xml	32.3 kB	12/24/2013
standalone.xml	27.3 kB	Monday
modeshape-users.properties	85 bytes	12/19/2013
modeshape-roles.properties	207 bytes	12/19/2013
mgmt-users.properties	876 bytes	12/24/2013
logging.properties	1.9 kB	14:58
application-users.properties	812 bytes	08/21/2013
application-roles.properties	634 bytes	08/21/2013

Figure D.12. Teiid Configuration File Selection

- Your new JBoss Data Virtualization server configuration will be opened in the JBoss / Teiid Editor for viewing. In this editor you can test both Teiid admin and JDBC connections.

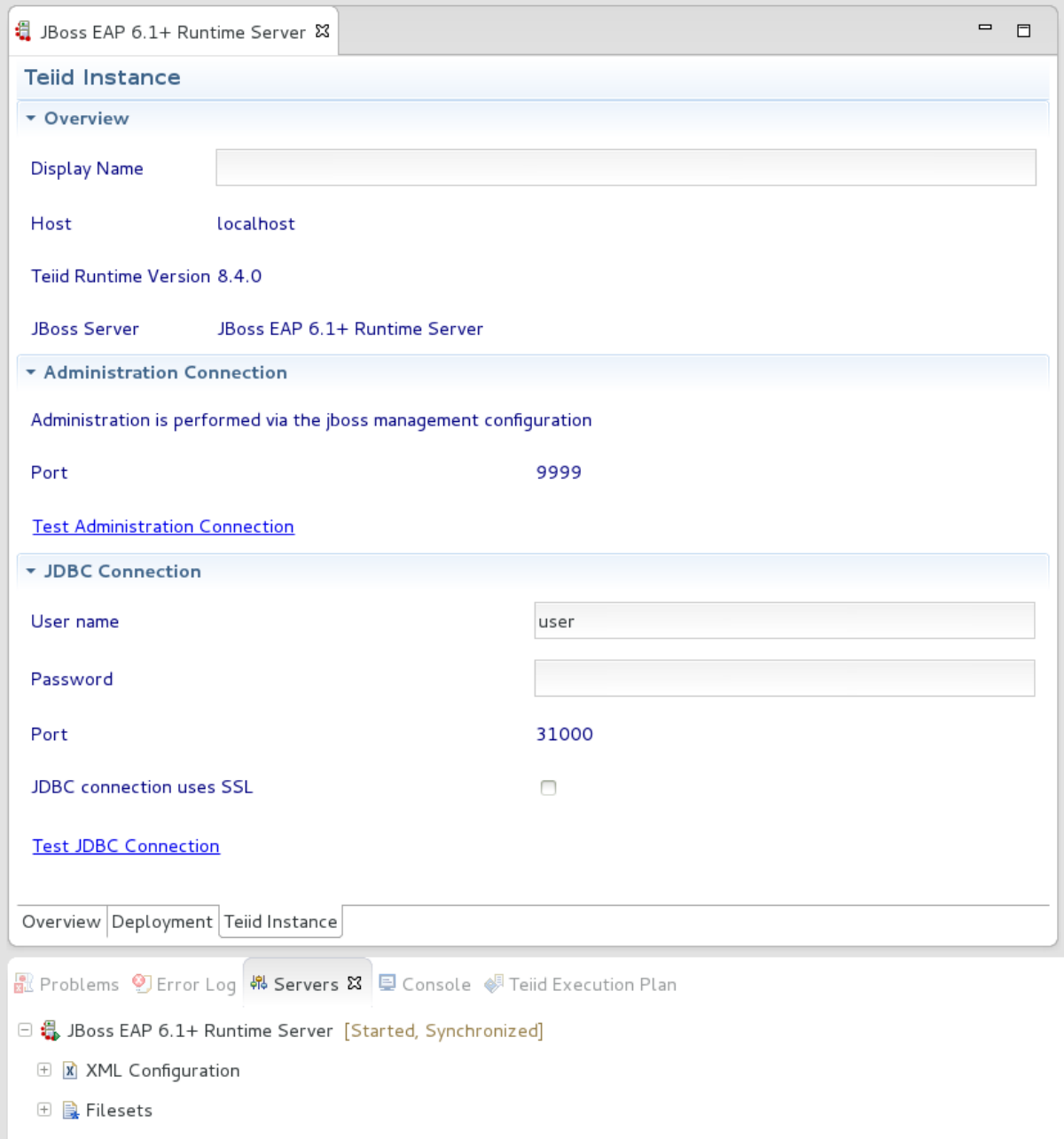


Figure D.13. Teiid Editor

Actions available in this view include:

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

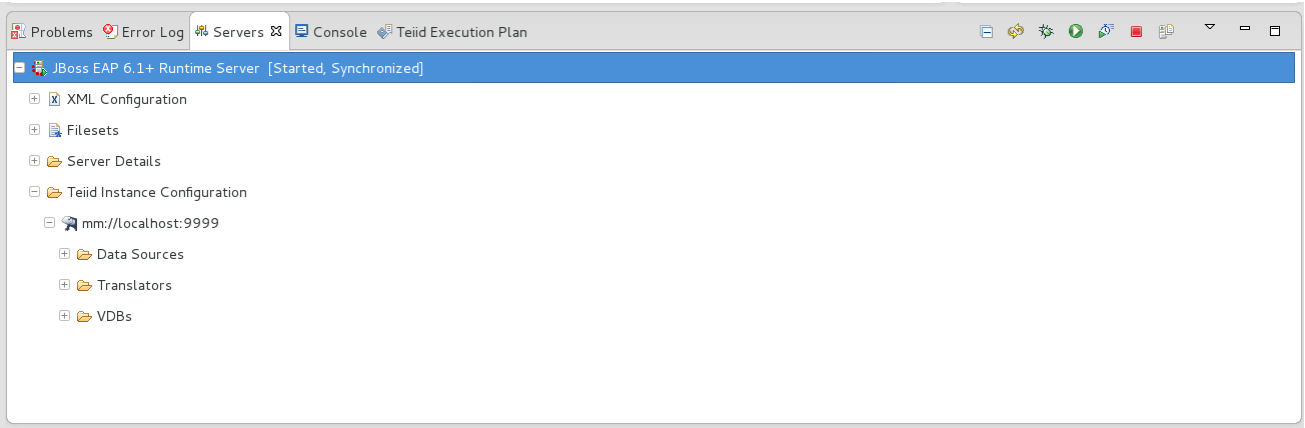


Figure D.14. Teiid Contents in Server View

D.8. 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.

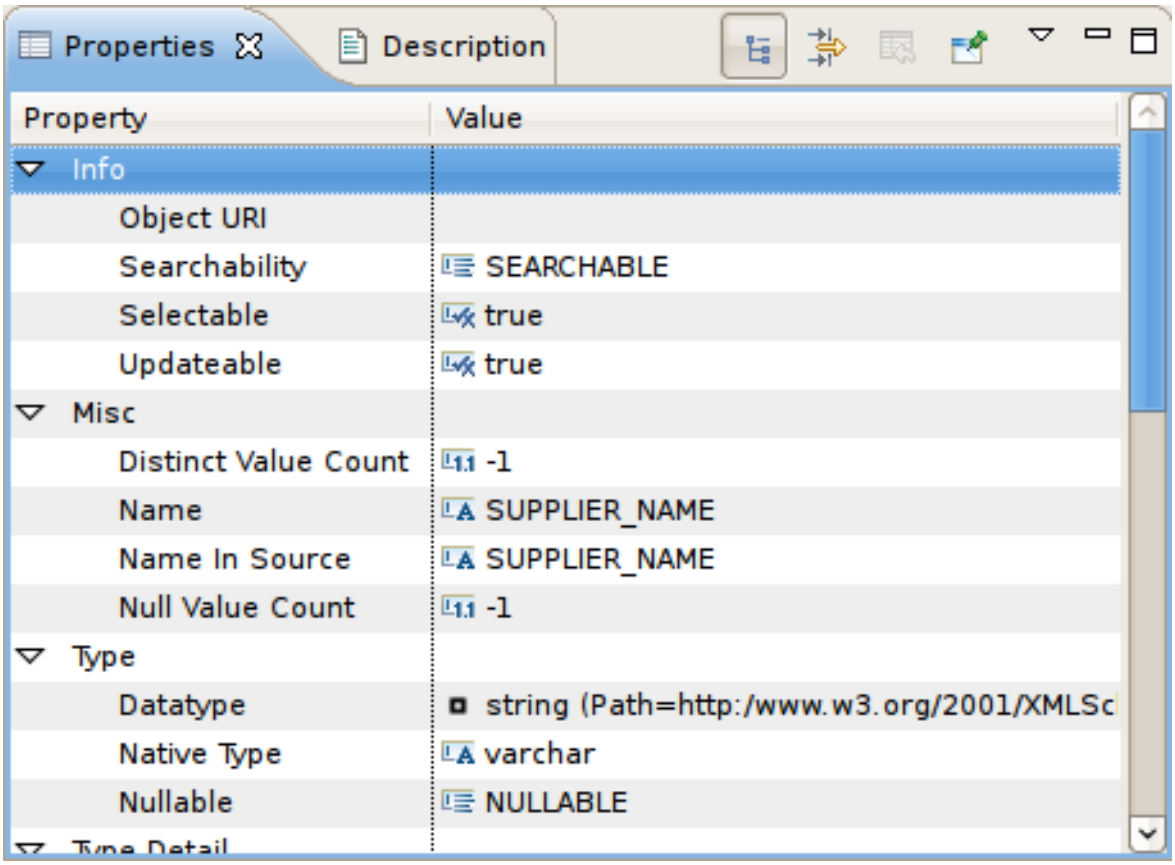


Figure D.15. Properties View

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 selecting the preference **`Always open editor without prompting`**. You can also select/clear this property via the Teiid Designer's main preference page.

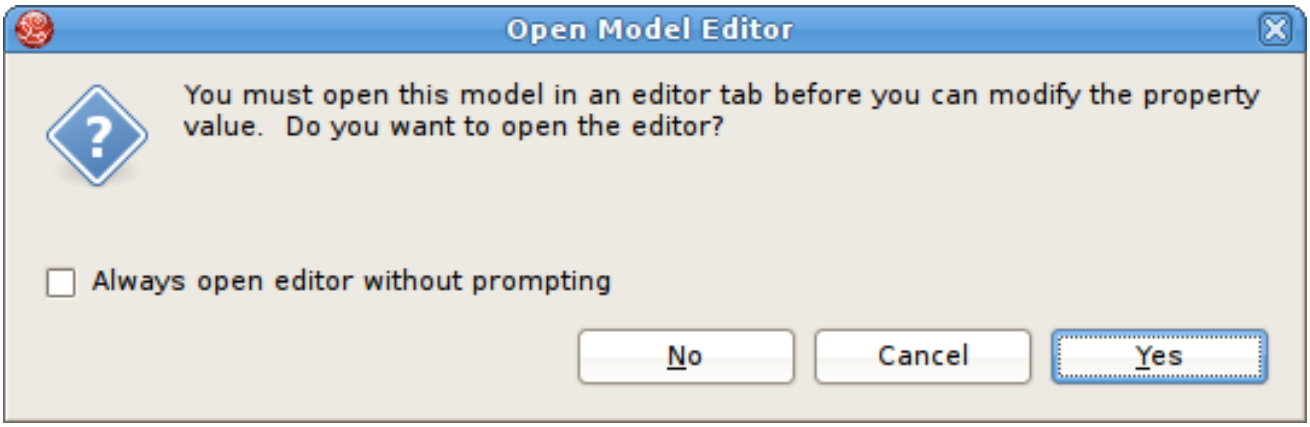


Figure D.16. Open Model Editor Dialog

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

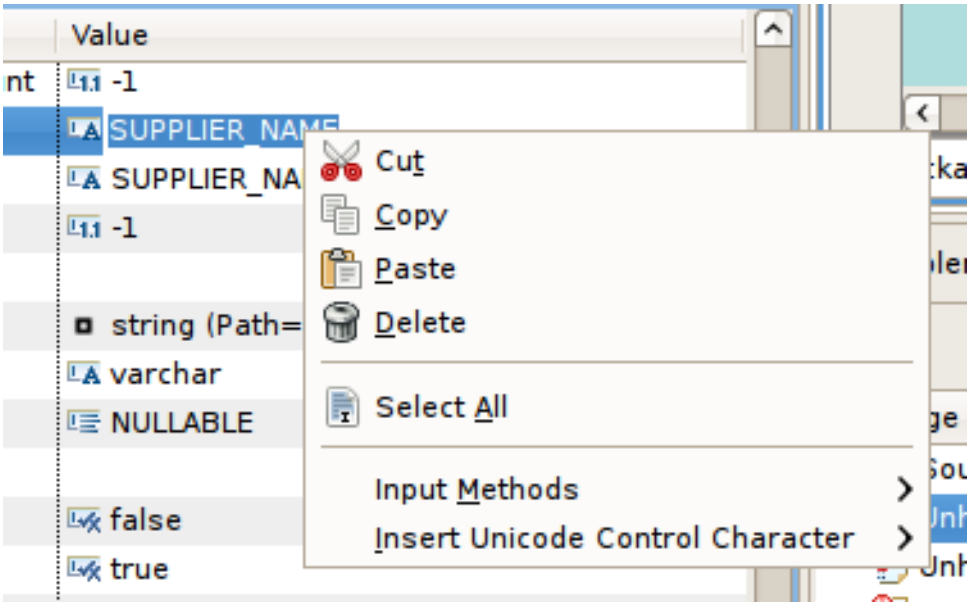


Figure D.17. Open Model Editor Dialog

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.9. 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 **Eclipse Show View** dialog. Click **Teiid Designer > Description** view and then click **OK**.

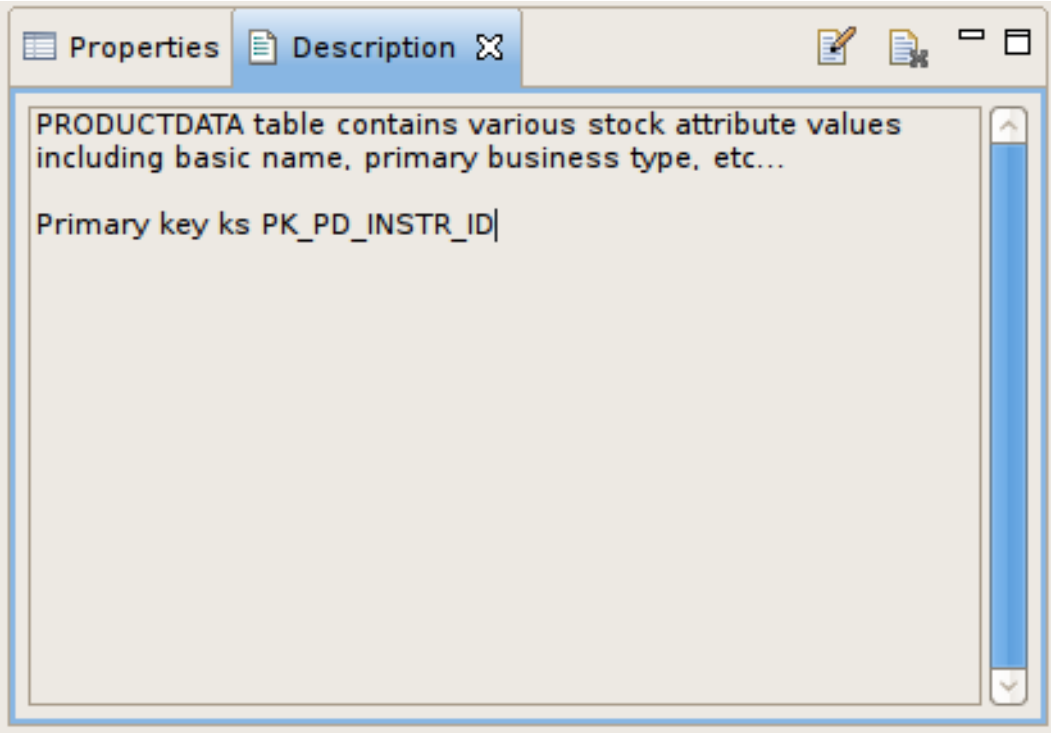


Figure D.18. Description View

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.

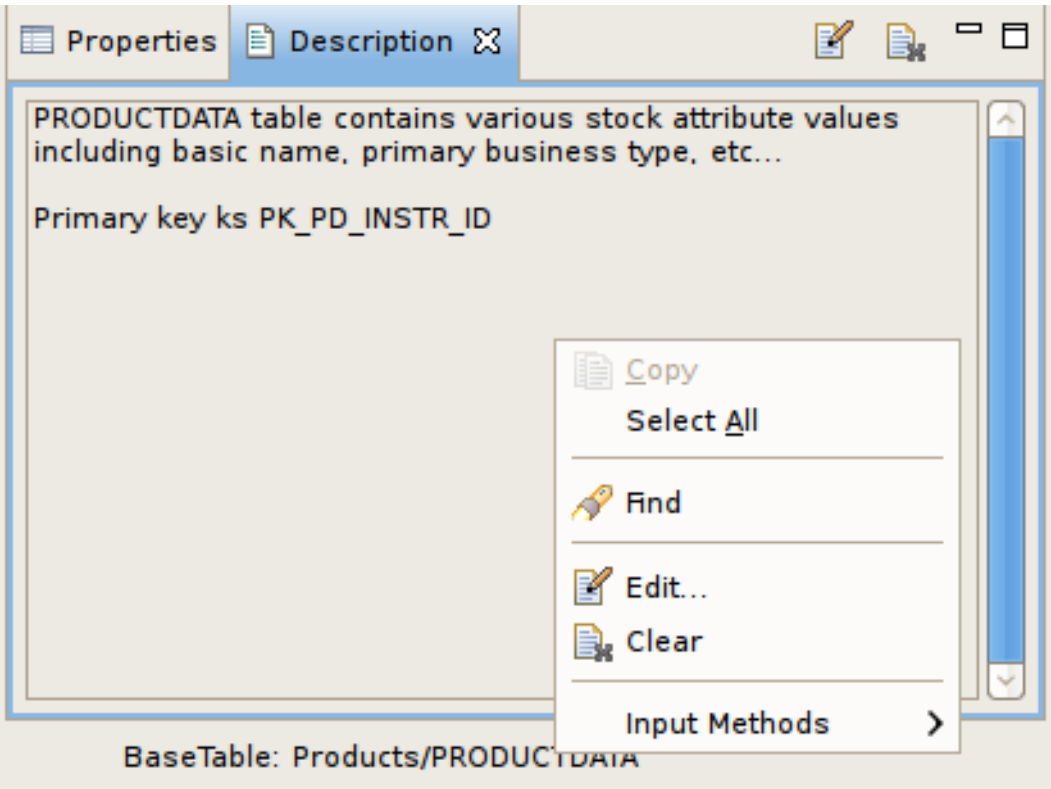


Figure D.19. Description View Context Menu

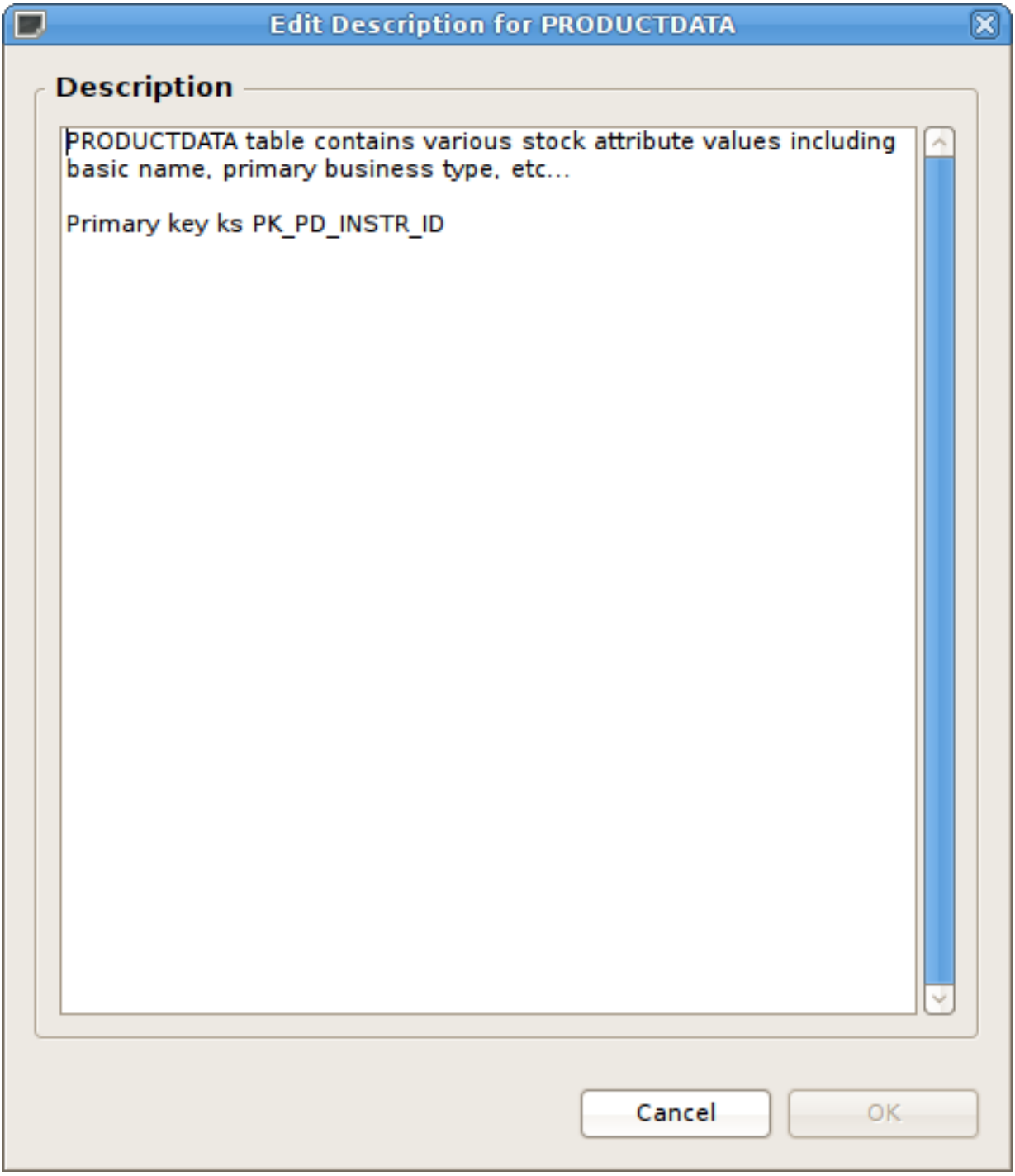


Figure D.20. Edit Description Dialog

D.10. PROBLEMS VIEW

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

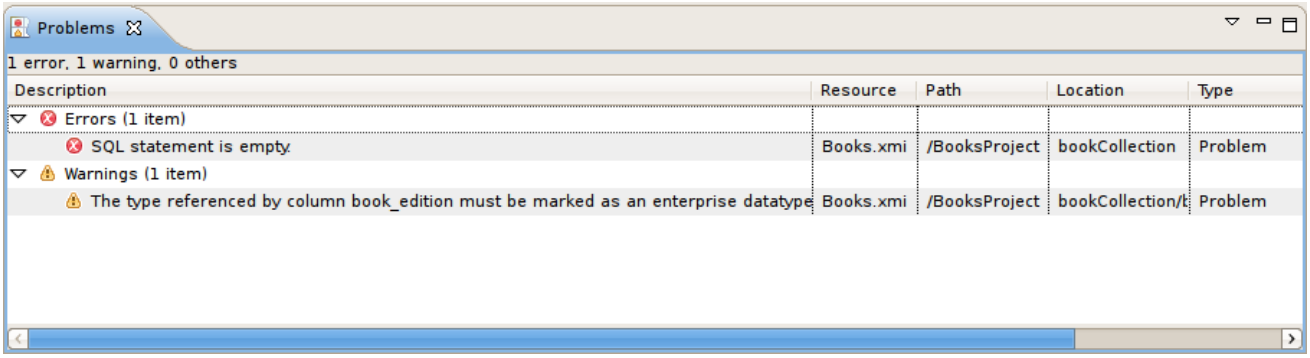


Figure D.21. Problems View

By default, the **`Problems View`** is included in the Teiid Designer perspective. If the **`Problems View`** is not showing in the current perspective, click **`Window > Show View > Other > General > 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.11. 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.12. CONTEXT MENU

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

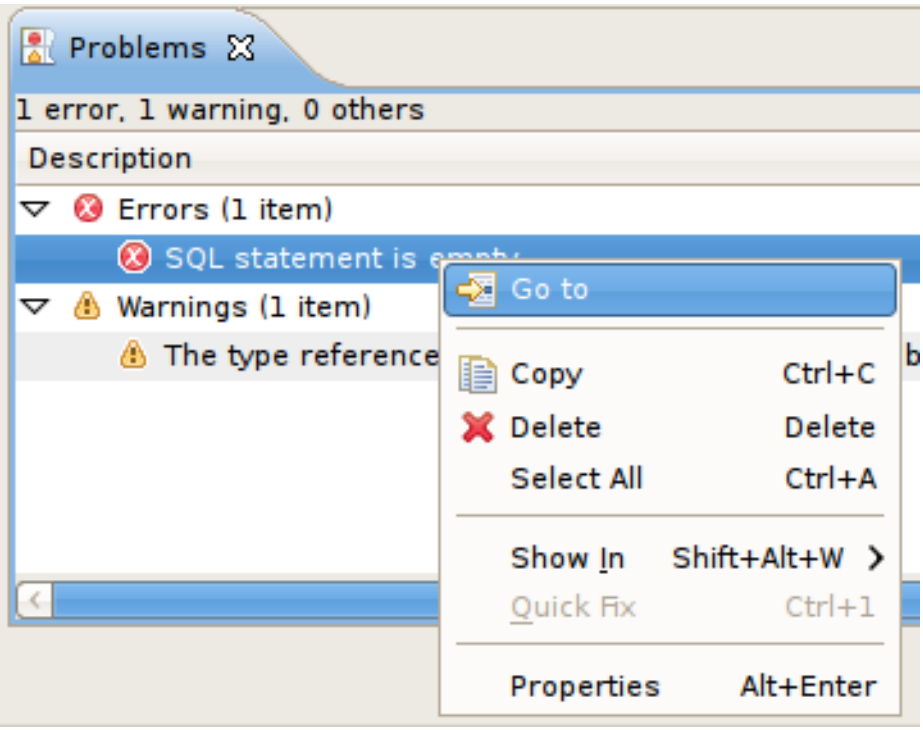


Figure D.22. Problems View Context Menu

- 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.
- 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.13. 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.

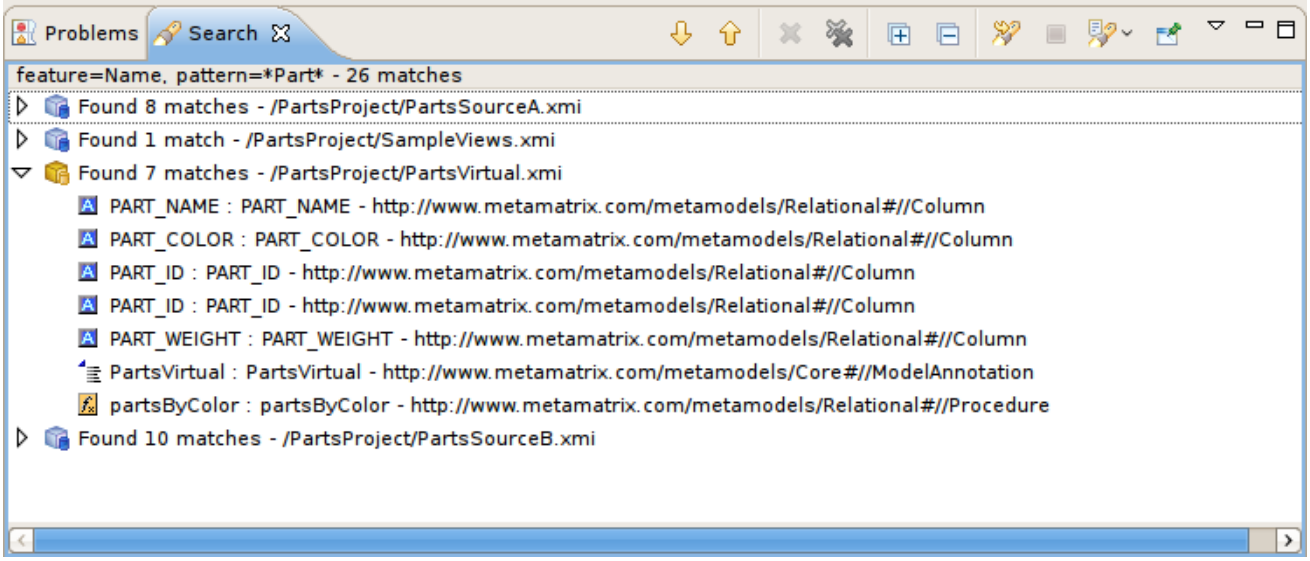


Figure D.23. Search Results View

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.
- Expand All - Expand all nodes in the search.
- Collapse All - Collapse all nodes in the search.
- Search - Launches the Teiid Designer Dialog.
- Previous Search Results - Select previous search results from history.

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

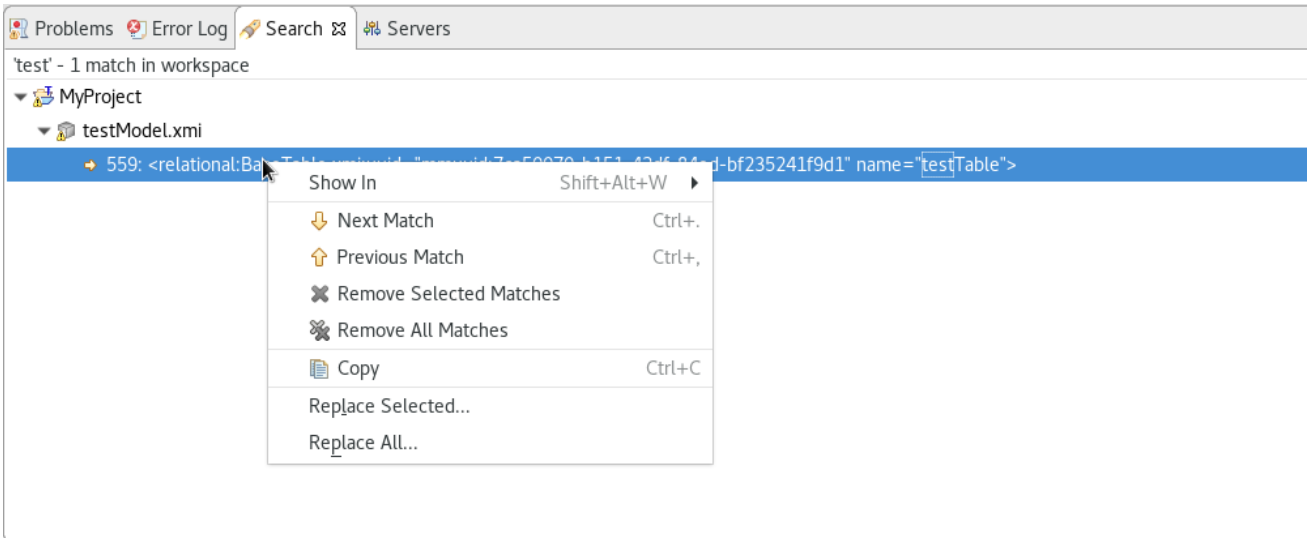


Figure D.24. Search Results Context Menu

D.14. DATATYPE HIERARCHY VIEW

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

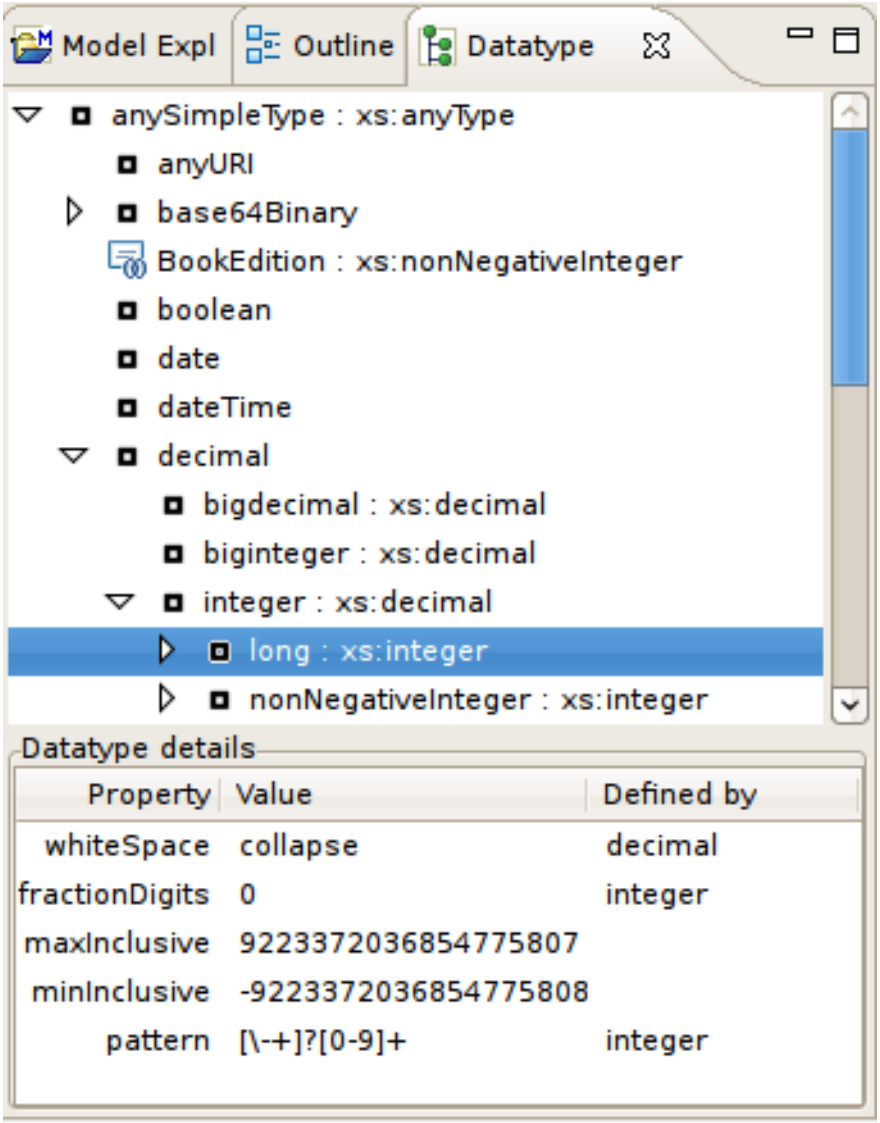


Figure D.25. Datatype Hierarchy View

The following table shows the mapping between Teiid Designer Types and JBoss Data Virtualization Runtime Types.

Table D.1. Corresponding Runtime Types

Teiid Designer Type	Java Runtime Type
anyURI	java.lang.String
base64Binary	java.lang.String
bigdecimal	java.math.BigDecimal
biginteger	java.math.BigInteger
blob	java.sql.Blob [a]
boolean	java.lang.Boolean

Teiid Designer Type	Java Runtime Type
byte	java.lang.Byte
char	java.lang.Character
clob	java.sql.Clob [b]
date	java.sql.Date
dateTime	java.sql.Timestamp
decimal	java.math.BigDecimal
double	java.lang.Double
duration	java.lang.String
ENTITIES	java.lang.String
ENTITY	java.lang.String
float	java.lang.Float
gDay	java.math.BigInteger
gMonth	java.math.BigInteger
gMonthDay	java.sql.Timestamp
gYear	java.math.BigInteger
gYearMonth	java.sql.Timestamp
hexBinary	java.lang.String
ID	java.lang.String
IDREF	java.lang.String
IDREFS	java.lang.String
int	java.lang.Integer
integer	java.math.BigInteger

Teiid Designer Type	Java Runtime Type
language	java.lang.String
long	java.lang.Long
Name	java.lang.String
NCName	java.lang.String
negativeInteger	java.math.BigInteger
NMTOKENS	java.lang.String
NMTOKENS	java.lang.String
nonNegativeInteger	java.math.BigInteger
nonPositiveInteger	java.math.BigInteger
normalizedString	java.lang.String
NOTATION	java.lang.String
object	java.lang.Object
positiveInteger	java.math.BigInteger
QName	java.lang.String
short	java.lang.Short
string	java.lang.String
time	java.sql.Time
timestamp	java.sql.Timestamp
token	java.lang.String
unsignedByte	java.lang.Short
unsignedInt	java.lang.Long
unsignedLong	java.lang.BigInteger
unsignedShort	java.lang.Integer

Teiid Designer Type	Java Runtime Type
XMLLiteral	java.sql.SQLXML [c]

[a] The concrete type is expected to be org.teiid.core.types.BlobType.

[b] The concrete type is expected to be org.teiid.core.types.ClobType.

[c] The concrete type is expected to be org.teiid.core.types.XMLType.

D.15. TEIID MODEL CLASSES VIEW

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

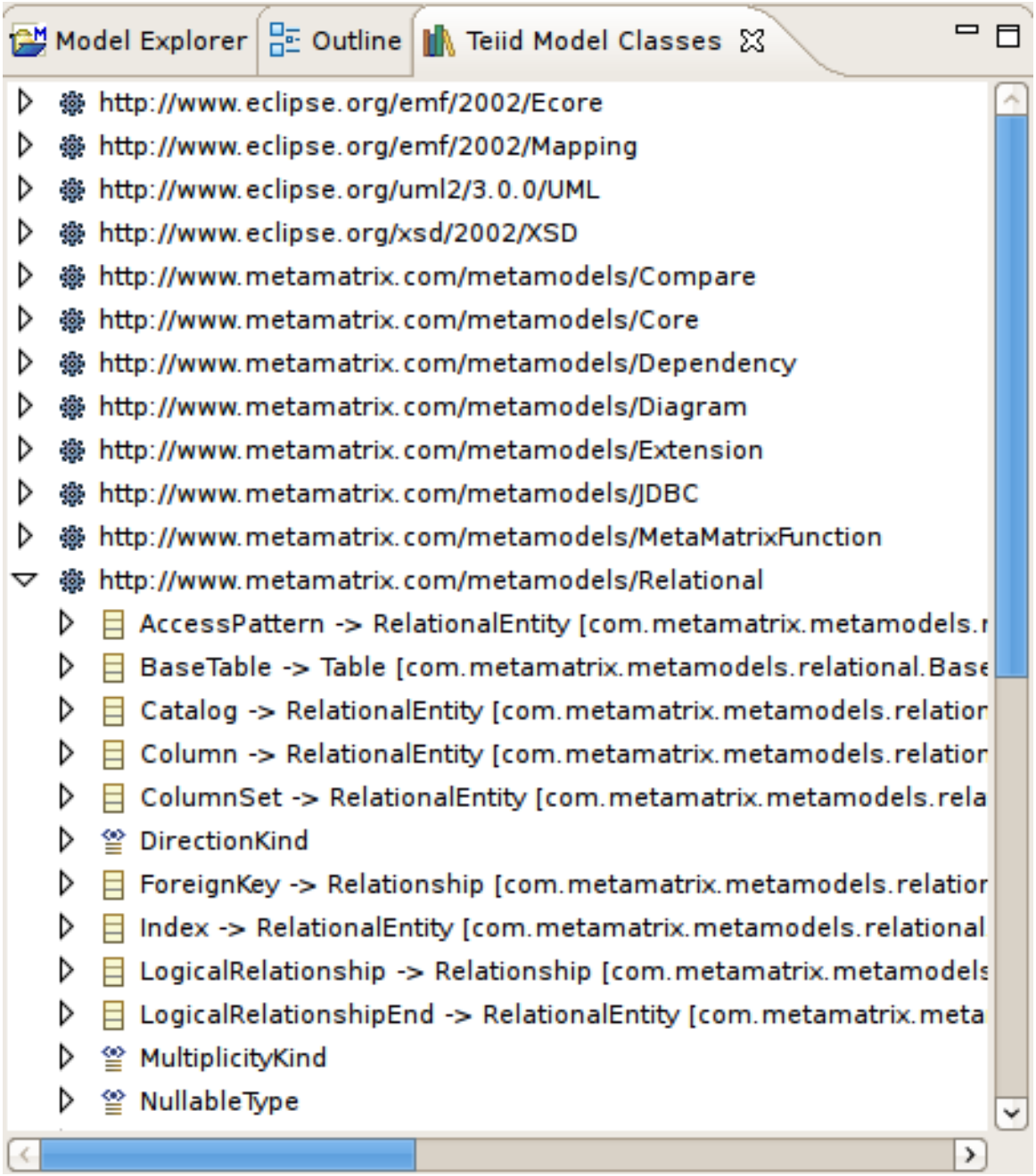


Figure D.26. Datatype Hierarchy View

D.16. SYSTEM CATALOG VIEW

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

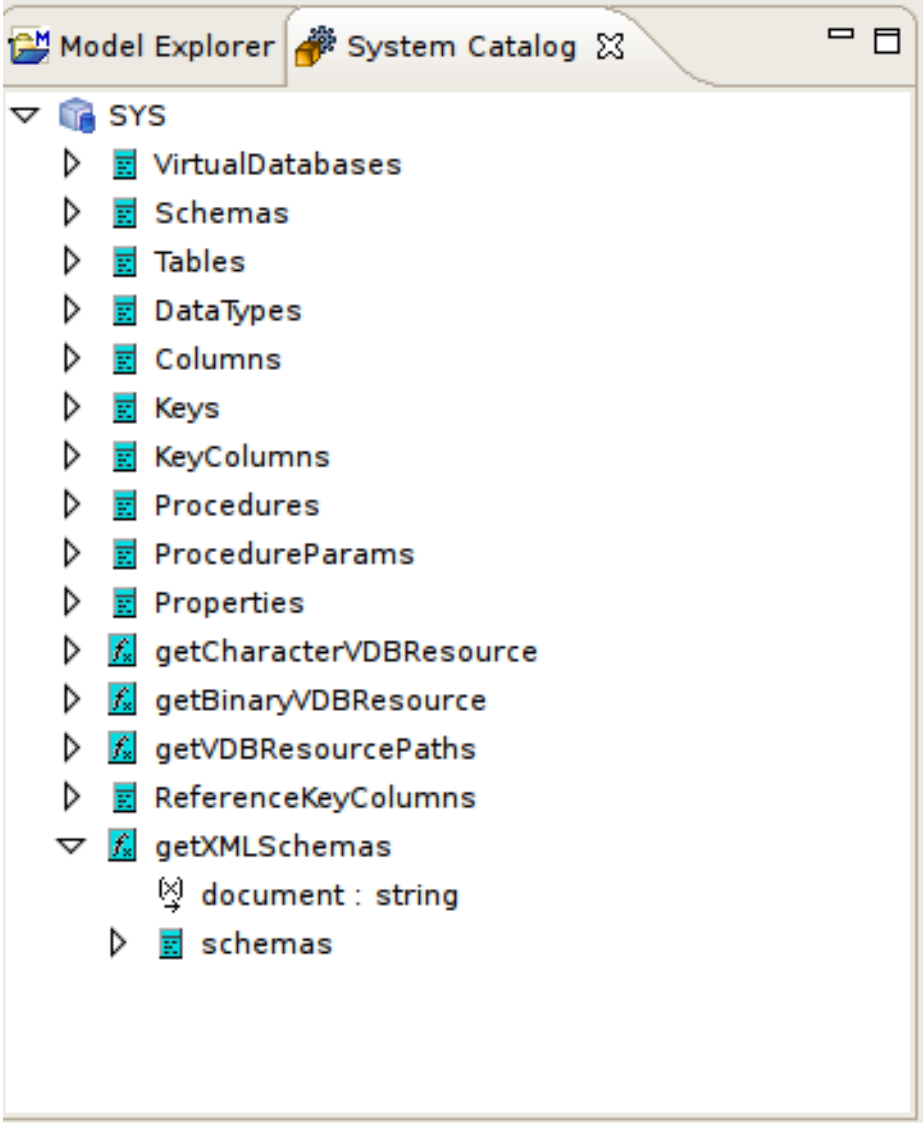


Figure D.27. System Catalog View

D.17. SQL RESERVED WORDS VIEW

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

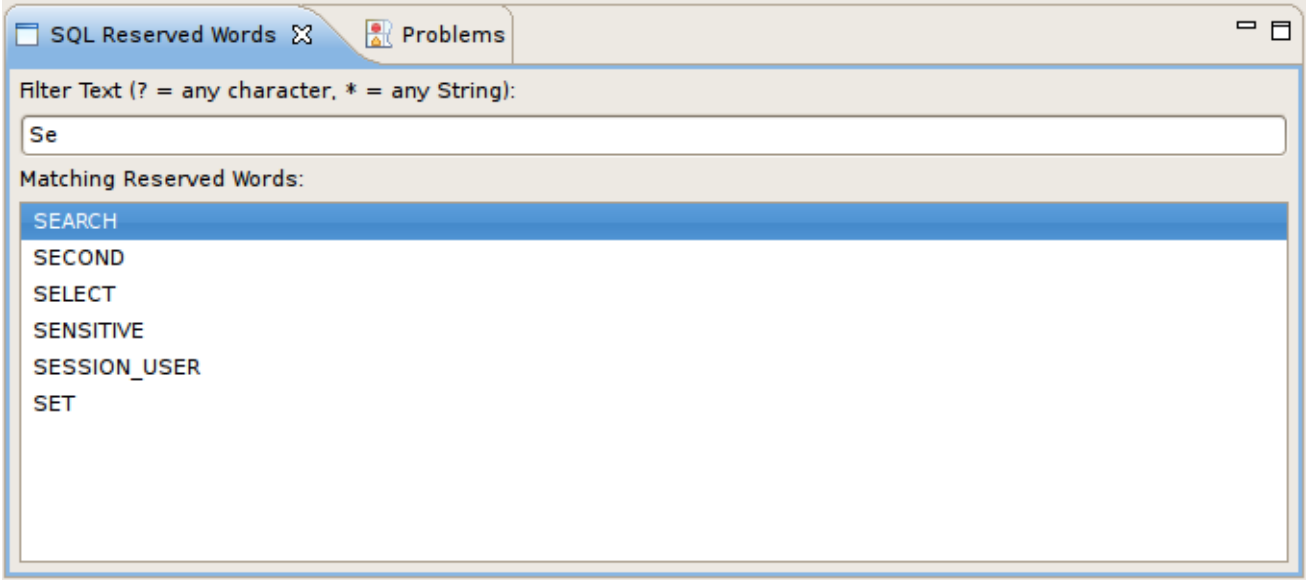


Figure D.28. System Catalog View

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

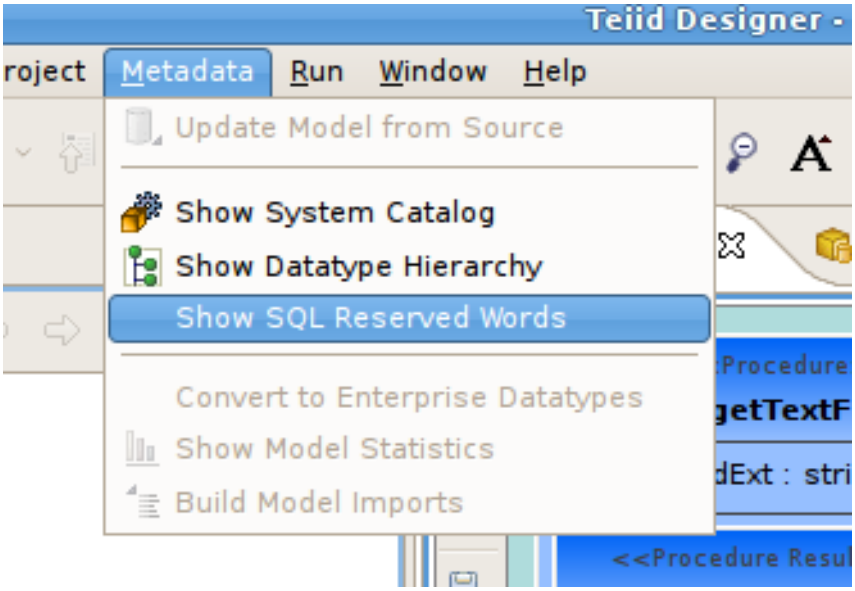


Figure D.29. SQL Reserved Words Action

D.18. MODEL EXTENSION DEFINITION REGISTRY VIEW (MED REGISTRY VIEW)

To open Teiid Designer's **`MED Registry View`**, click the main menu's **`Window > Show View > Other...`** and then click the **`Teiid Designer > Model Extension Registry`** view in the dialog.

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

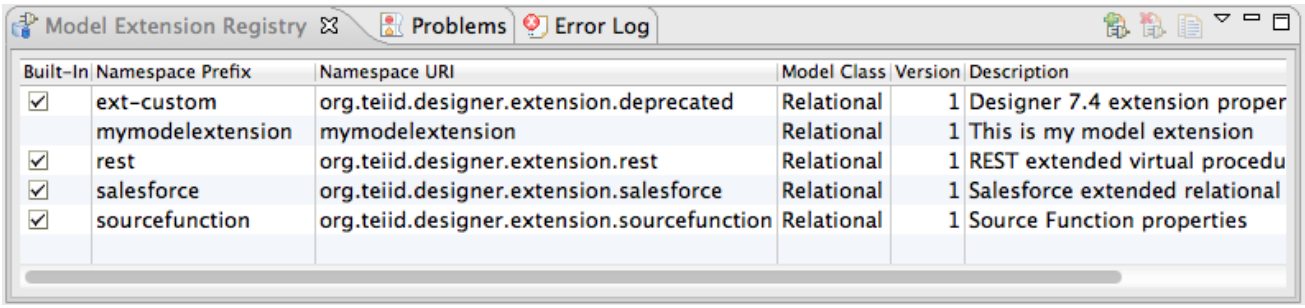


Figure D.30. System Catalog View

You can also open the view by clicking the **`MED Editor`** toolbar action in the right corner of shared sub-editor header section.

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, and copy a registered MED to the workspace. All 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 Teiid 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.19. GUIDES VIEW

To open Teiid Designer's **`Guides View`**, click the main menu's **`Window > Show View > Other...`** and then click 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 Cheat Sheets view). The Guides view is shown below:

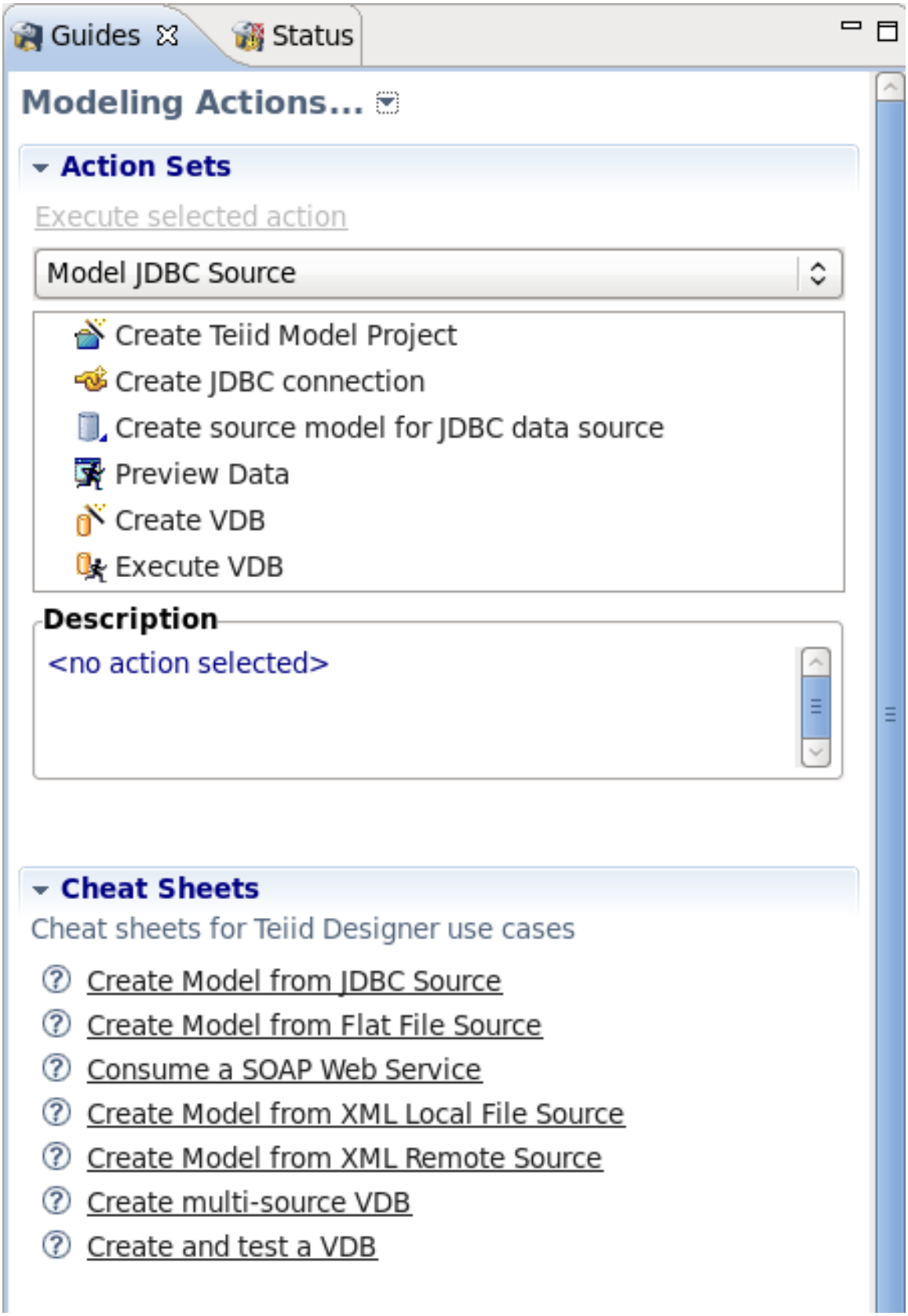


Figure D.31. Guides View

The upper **`Action Sets`** section provides categorized sets of actions. Select the desired category in the drop-down, 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 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.20. STATUS VIEW

To open Teiid Designer's **`Status View`**, click the main menu's **`Window > Show View > Other...`** and then click 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:

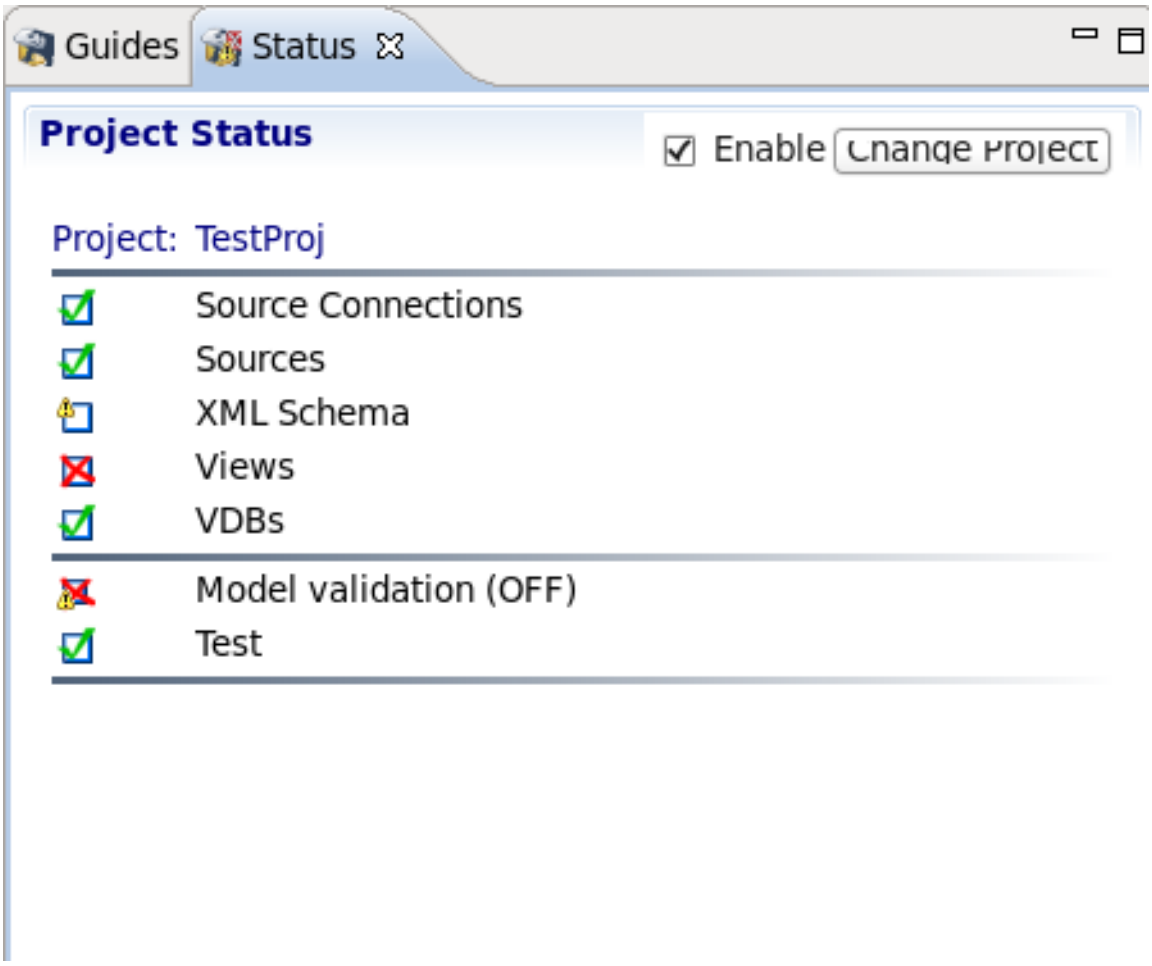


Figure D.32. Status View

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.21. CHEAT SHEETS VIEW

To open **`Cheat Sheets View`**, click the main menu's **`Window > Show View > Other...`** and then click 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 Guides View) provides links to these Cheat Sheets, as previously described. A sample Cheat Sheet is shown below:

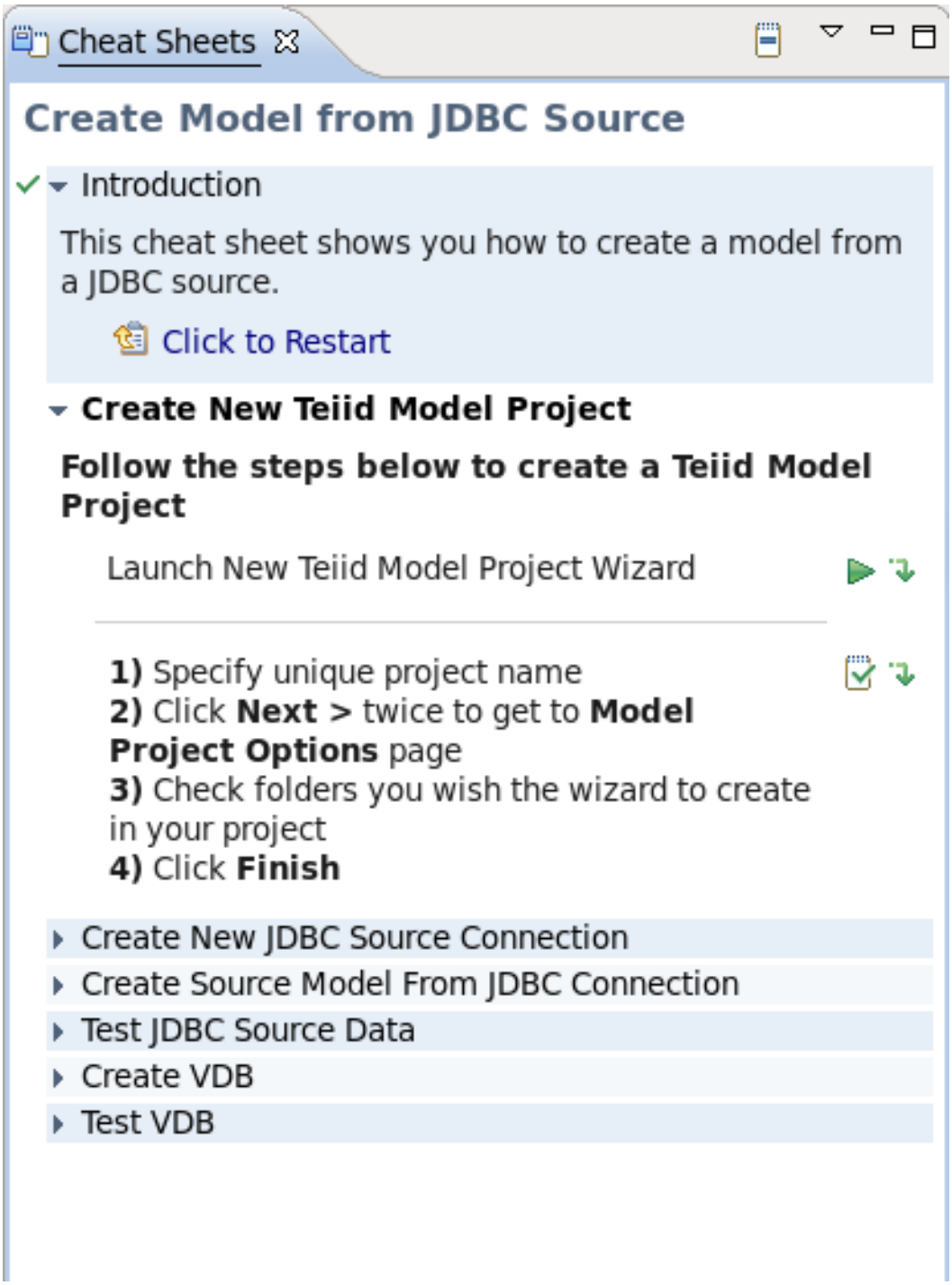


Figure D.33. Cheat Sheet Sample

D.22. CONNECTIONS VIEW

The **`Connections View`** provides a quick overview and management of Connection Profiles and

Deployed Data Sources. When the **`Local Profiles`** section is marked, add button opens the wizard for creating a new Connection Profile. When the **`Deployed`** section is marked, add button opens the wizard for creating a new Data Source.

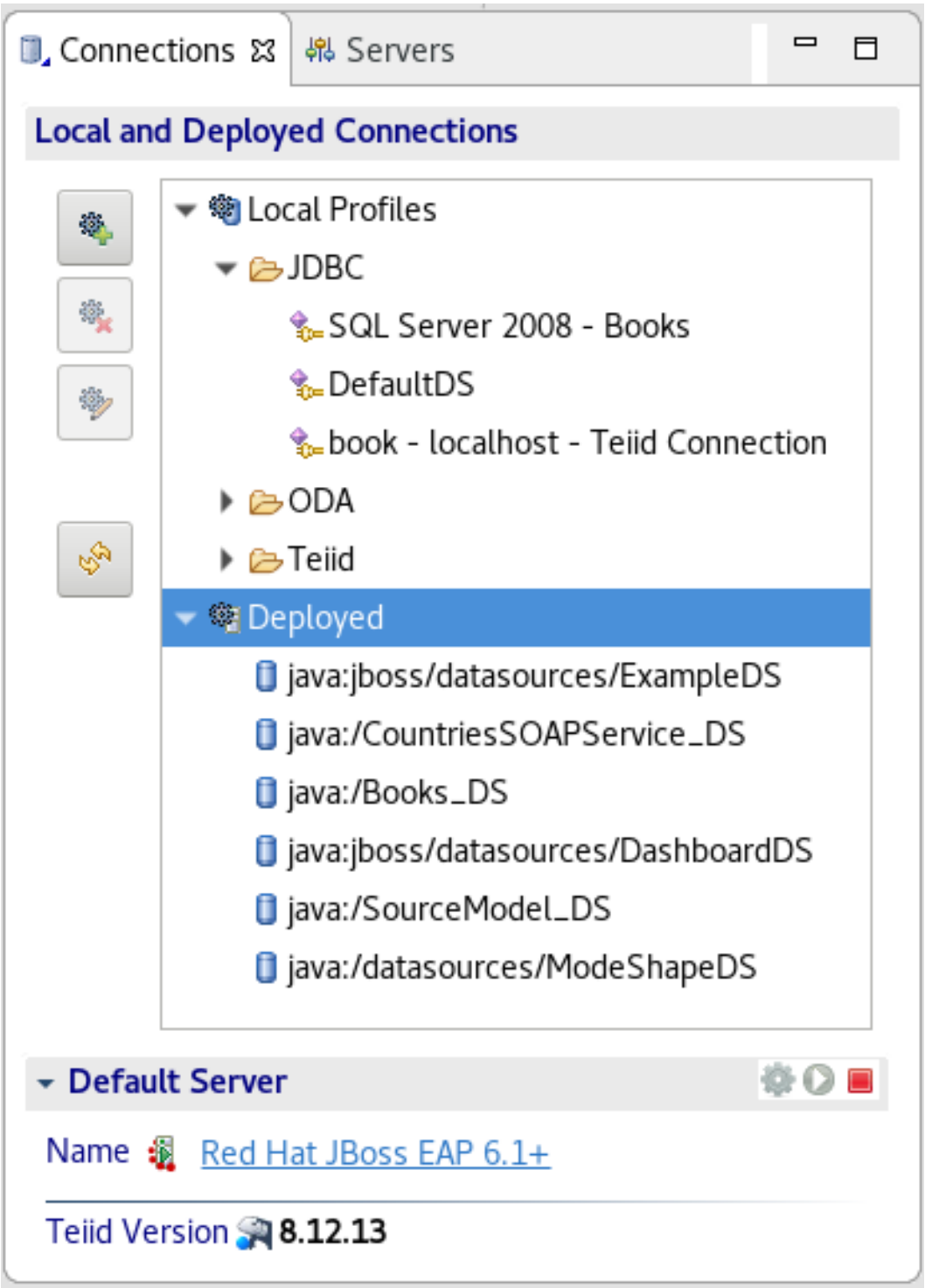


Figure D.34. Connections View

Connections View contains these buttons:

- Create a new Data Source or Connection Profile
- Delete marked Data Source or Connection Profile

- Edit marked Data Source or Connection Profile
- Refresh profiles and deployed data sources

The **`Connections View`** also contains Default Server panel which is described in the **`Server Version Status Panel`** section.

APPENDIX E. EDITORS

E.1. 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.

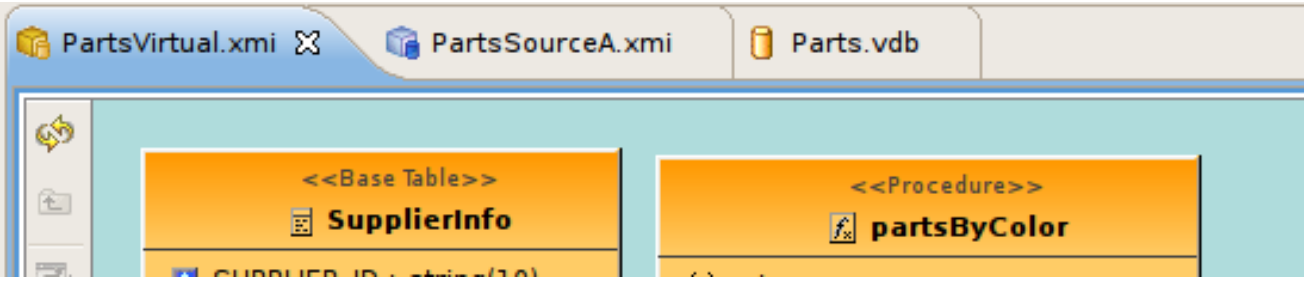


Figure E.1. Editor Tabs

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

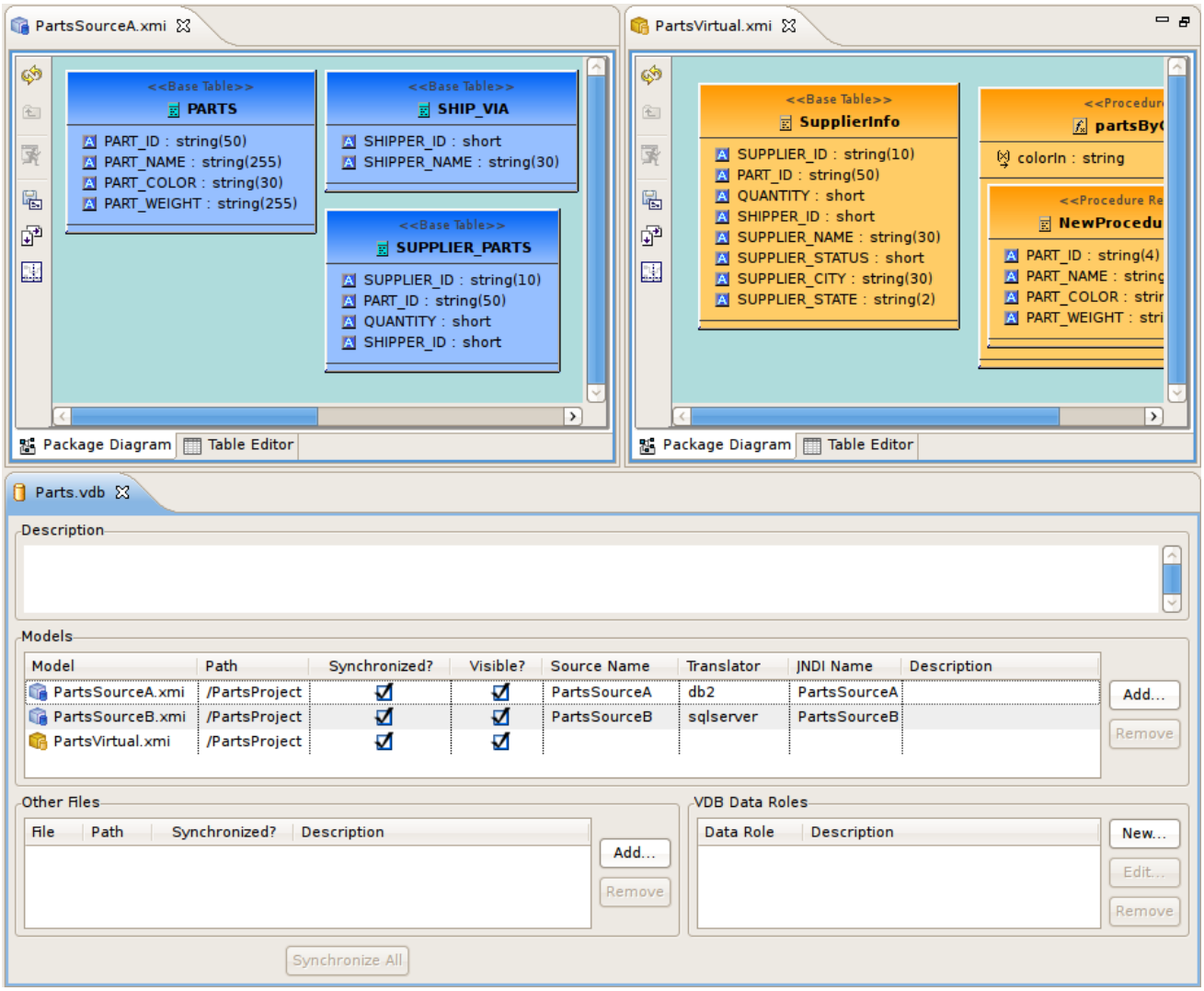


Figure E.2. Viewing Multiple Editors

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

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

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

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

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

- Transformation Editor - Manages Transformation SQL for Relational View Base Tables, Procedures and XML Web Service Operations.

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

E.2. MODEL EDITOR

E.2.1. Model Editor

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

E.2.2. 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 E.3. Main Toolbar Diagram Actions

The actions include:

- Zoom In

- 125% Zoom to Level
- Zoom Out
- Increase Font Size
- Decrease Font Size
- Perform Diagram Layout

E.2.3. 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.

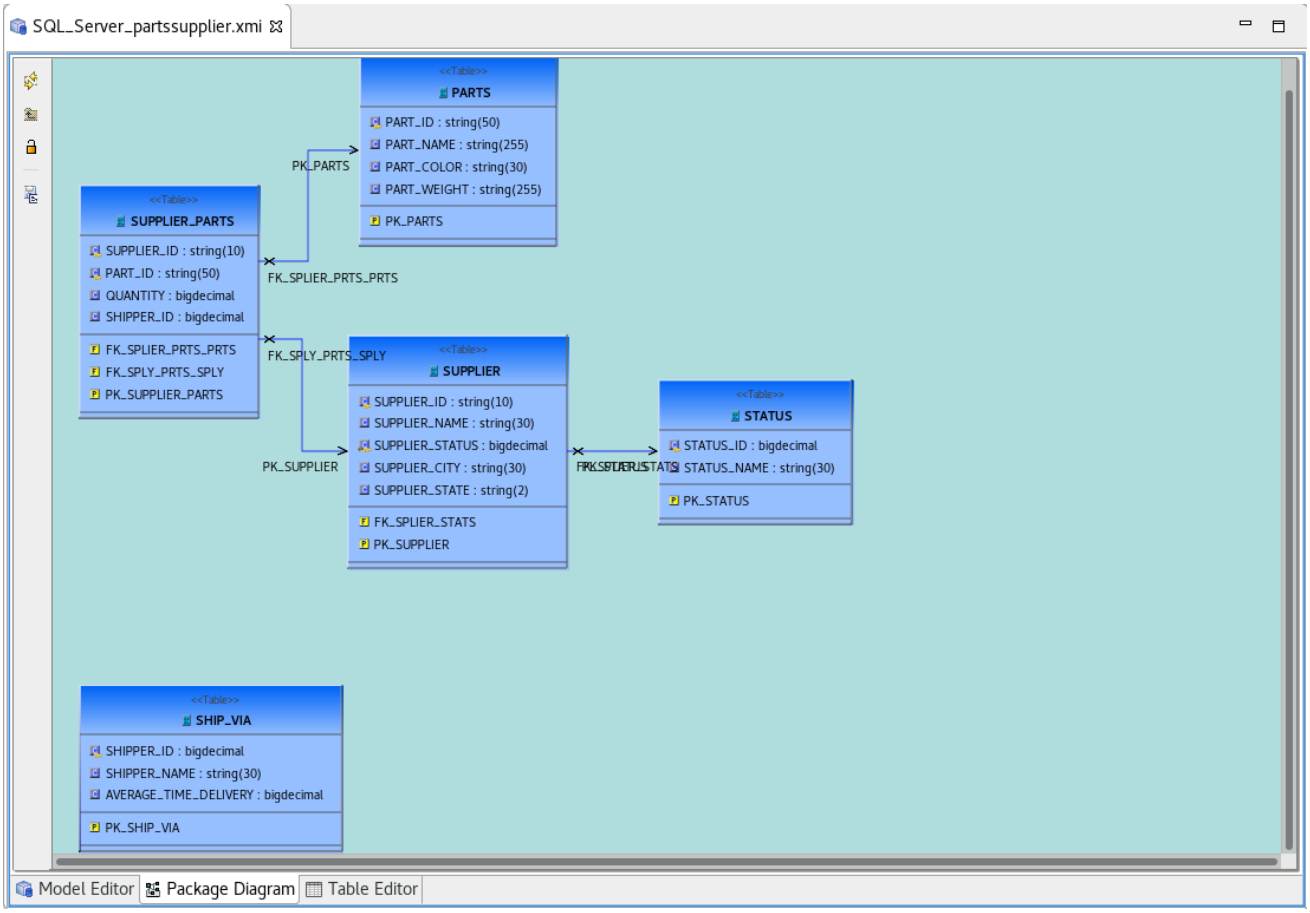


Figure E.4. Package Diagram Example

Package Diagram toolbar actions include:

- Refresh Diagram - Redraws diagram.
- Show Parent Diagram - Navigates to diagram for parent object (if available).
- Save Diagram as File - Save the diagram image to file in JPG or BMP format.

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).

E.2.4. 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 E.5. Custom Diagram Example

Custom Diagram toolbar actions include:

- Refresh Diagram - Redraws diagram.
- Show Parent Diagram - Navigates to diagram for parent object (if available).
- Preview Data - Executes a simple preview query **`(SELECT * FROM )`**.
- Add To Diagram - Add objects selected in Model Explorer to diagram.
- Remove From Diagram - Removed objects selected in diagram from diagram.
- Clear Diagram - Remove all objects from diagram.
- Save Diagram as Image - Save the diagram image to file in JPG or BMP format.
- Modify Diagram Printing Preferences - Modify page layout information for printing diagrams. Includes margins, orientation, etc...
- Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Since Custom Diagrams do not represent 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.

E.2.5. 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 E.6. Transformation Diagram Example

Transformation Diagram toolbar actions include:

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

- 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.

E.2.6. 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 E.7. Mapping Diagram Example

Mapping Diagram toolbar actions include:

- Refresh Diagram - Redraws diagram.
- Show Parent Diagram - Navigates to diagram for parent object (if available).
- Show detailed mapping diagram - Show detailed mapping transformation diagram for selected mapping class.
- 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
- Hide all columns in Mapping Classes
- Populate diagram with Mapping Class derived from tree 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..)

E.2.7. 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.

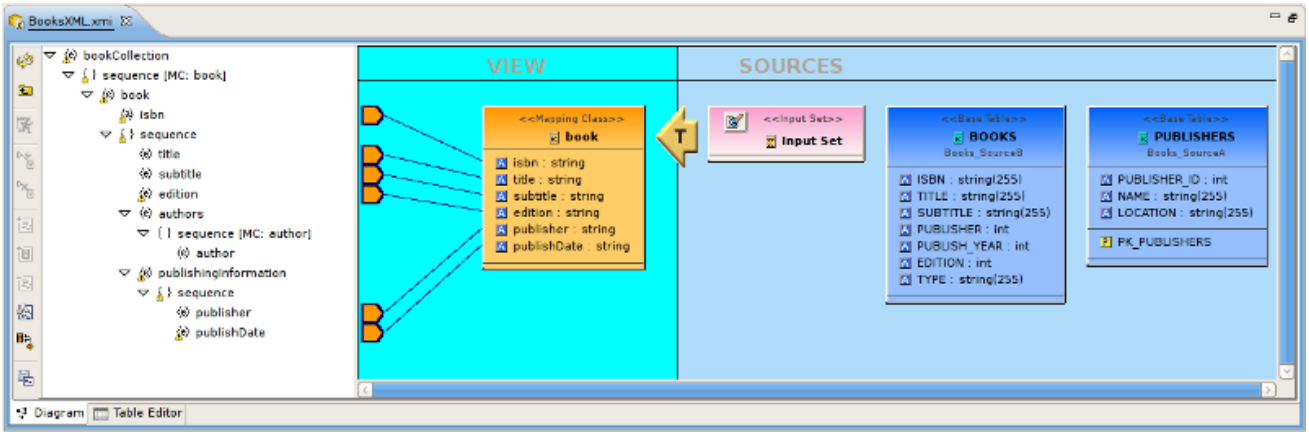


Figure E.8. Mapping Transformation Diagram Example

Mapping Transformation Diagram toolbar actions include:

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

E.3. TABLE EDITOR

E.3.1. 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.

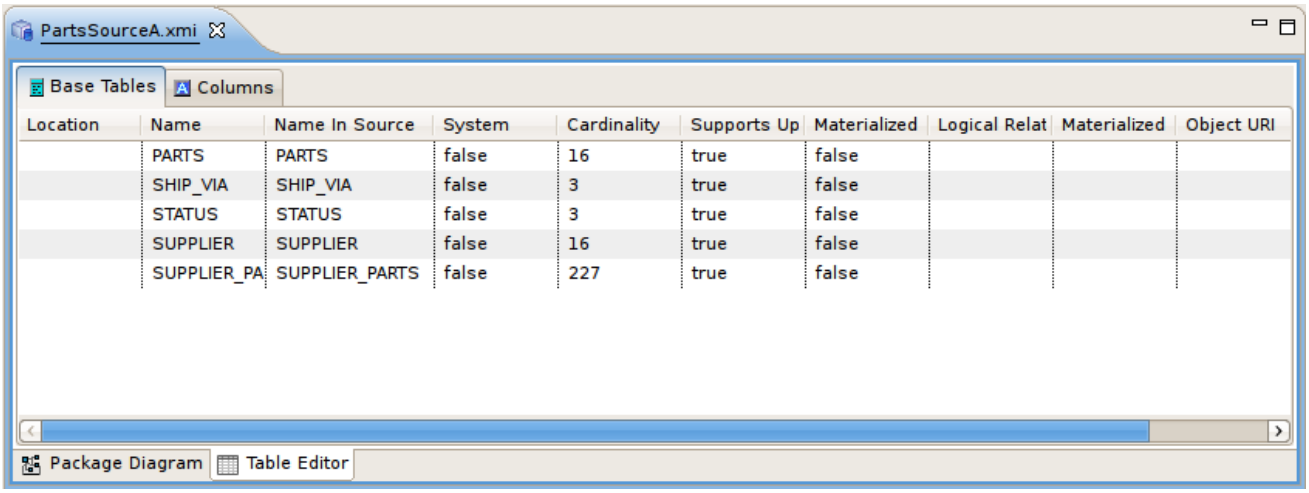


Figure E.9. Table Editor Example

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:

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

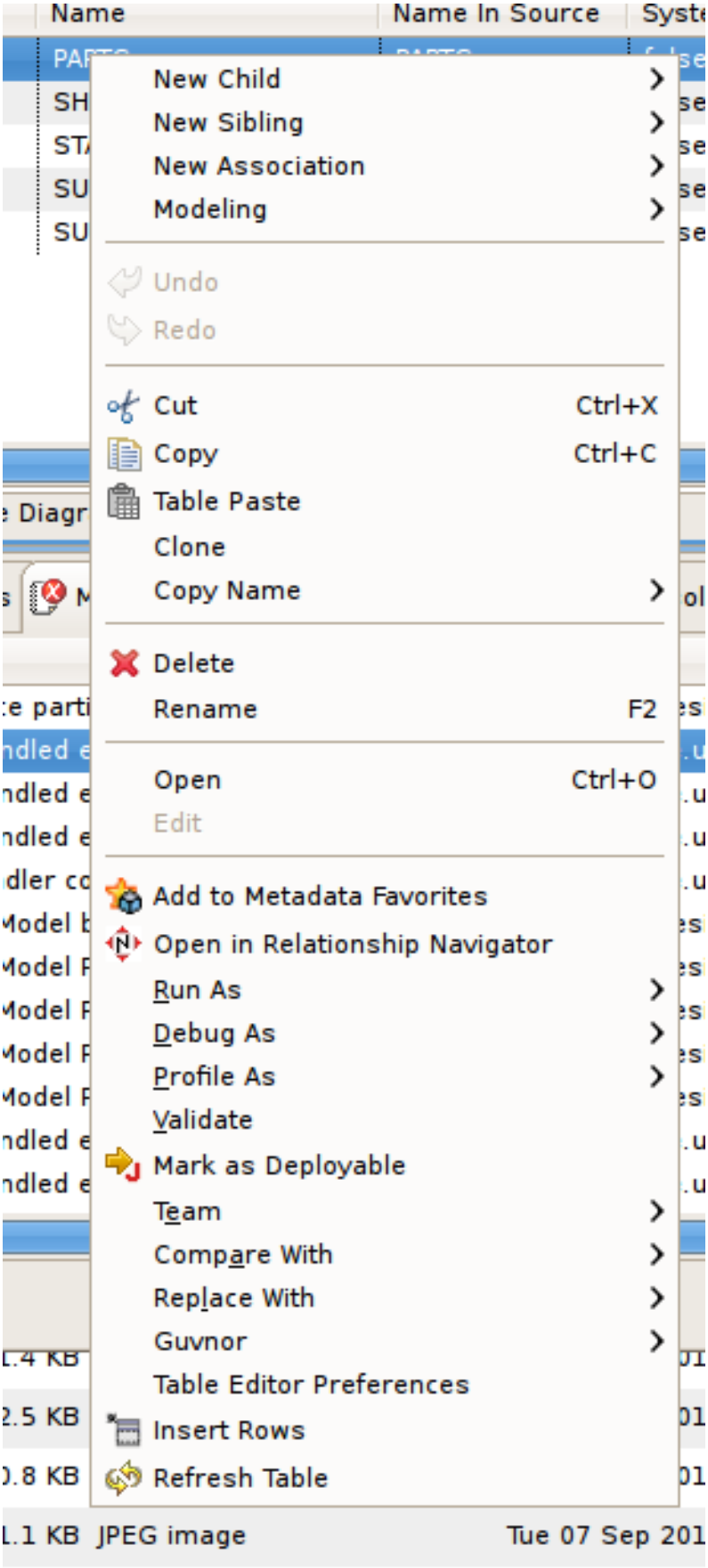


Figure E.10. Table Editor Example

E.3.2. 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.

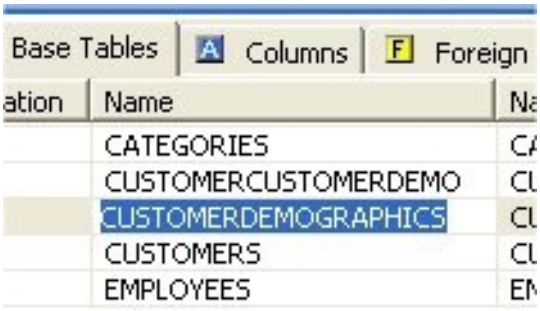


Figure E.11. Editing String Property

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.

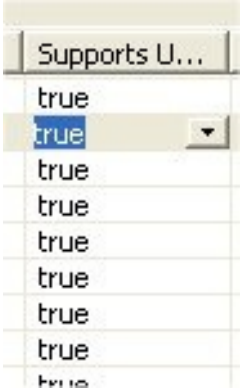


Figure E.12. Editing Boolean Value

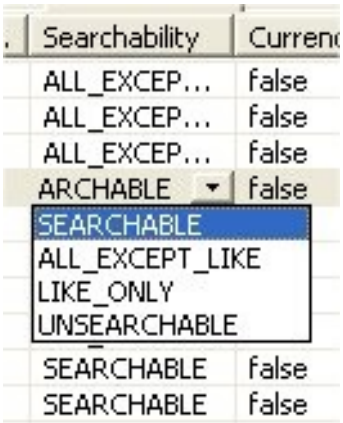


Figure E.13. Editing Multi-Value Property

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.

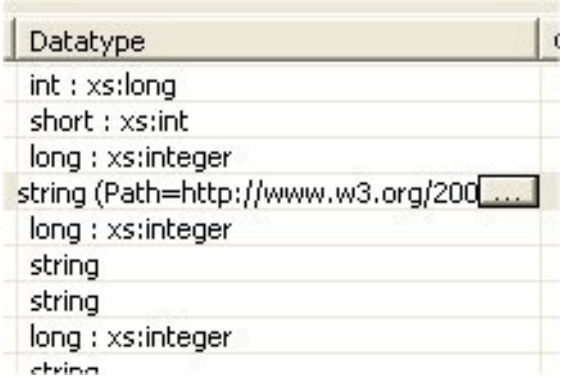


Figure E.14. Editing Multi-Value With Picker

An example 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 E.15. Editing Datatype Values

E.3.3. 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:

1. Select a table row to insert rows after.
2. Right-click select **`Insert Rows`** action or click the **`Insert Rows`** action on the main toolbar. The following dialog will be displayed.

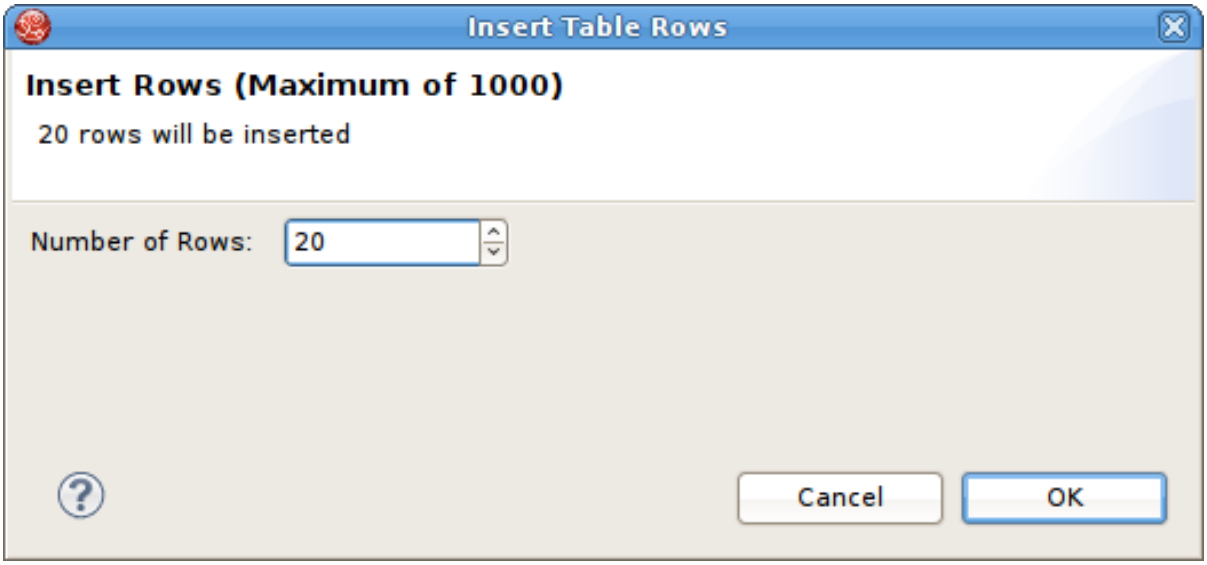


Figure E.16. Editing String Property

3. Edit the Number of Rows value in the dialog, or use the up/down buttons to change the value.
4. Click **OK** in dialog.

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

E.4. SIMPLE DATATYPES EDITOR

The **`Simple Datatypes Editor`** provides a form based properties view of XML Schema data.

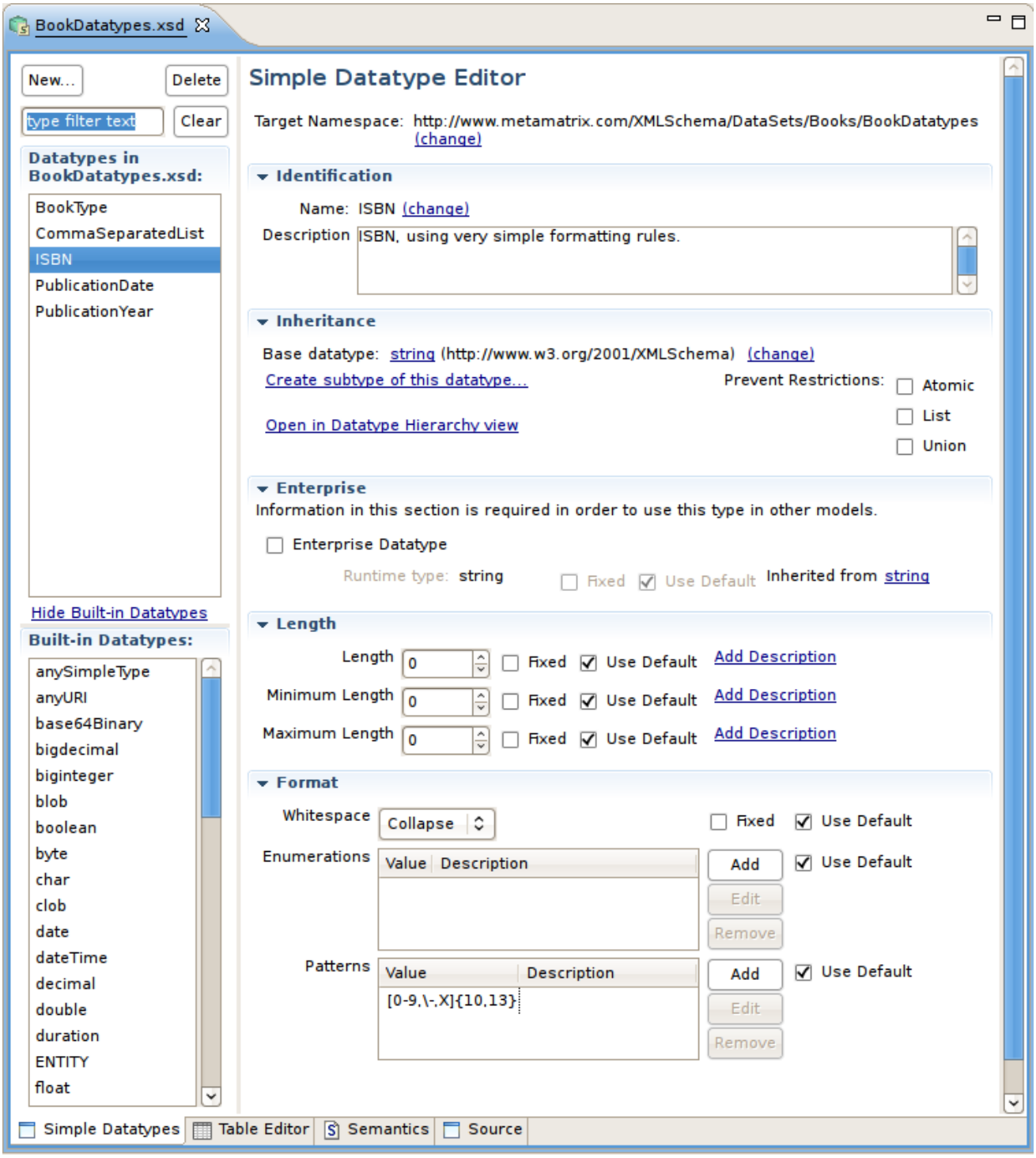


Figure E.17. Editing String Property

E.5. SEMANTIC EDITOR

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

E.6. SOURCE EDITOR

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

E.7. MODEL OBJECT EDITORS

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

E.8. VDB EDITOR

E.8.1. 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 following image shows the **`VDB Editor`**:

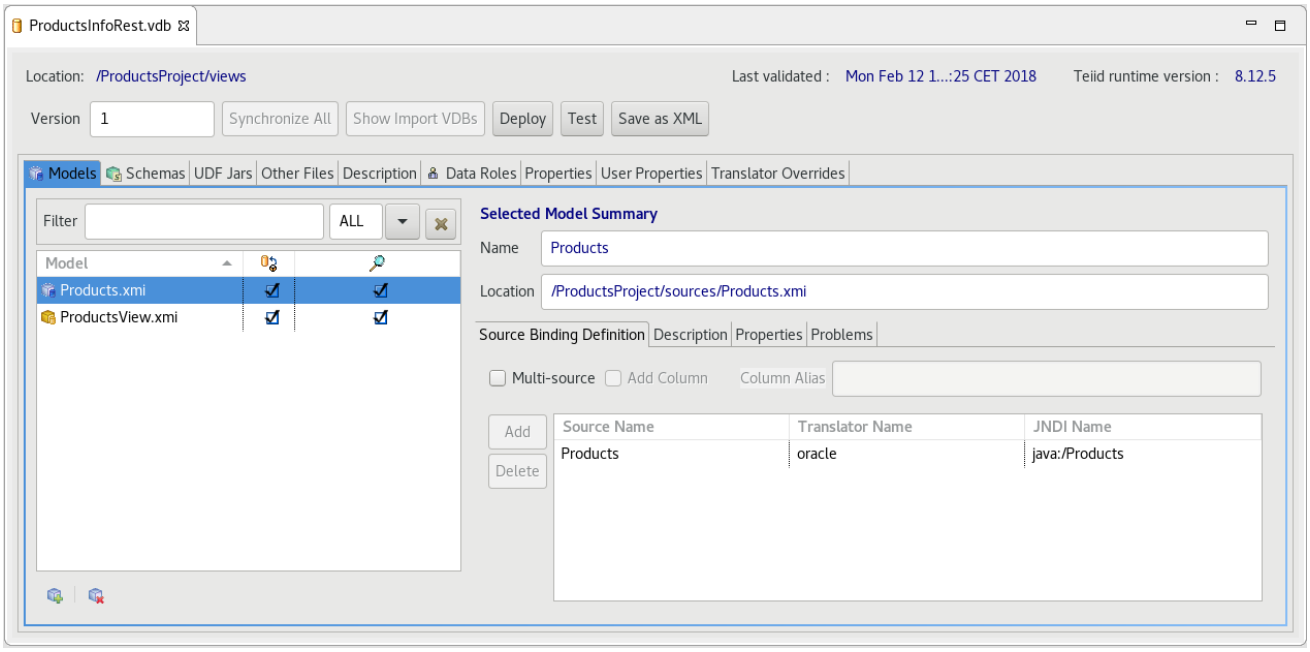


Figure E.18. VDB Editor

The **`VDB Editor`** contains the following tabs and buttons:

Table E.1. Tabs and Buttons on the VDB Editor

Name	Description
`Schemas`	This tab manages the schema files included in the VDB.
`UDF Jars`	This tab manages the User-Defined Function jars included in the VDB.
`Other Files`	This tab manages the non-model files included in the VDB.
`Description`	In this tab, you can set descriptions of the VDB.

Name	Description
`Properties`	This tab manages VDB properties.
`User Properties`	This tab manages user-defined VDB properties.
`Translator Overrides`	This tab manages the overridden translators and their properties.
`Synchronize All`	This button synchronizes all VDB entries with their corresponding workspace files.
`Show Import VDBs`	This button is enabled if VDB imports exist for a VDB and allows viewing the names and versions of the imported VDBs.
`Deploy`	This button deploys the selected VDB to JBoss Data Virtualization.
`Test`	This button deploys the selected VDB to JBoss Data Virtualization, creates a Teiid Connection Profile specific for that VDB, opens the Database Development perspective, and creates a connection to your VDB.
`Save as XML`	This button generates an XML version of the **`*.vdb`** archive and saves it to the workspace or to local file system.

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

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

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

The **`Filter`** bar provides full text search capability in the VDB editor. You can also choose a filter by type. The supported types are: ALL, Source, View, Web, and XML Doc. The **`X`** button clears the search bar and returns it to its default state.

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

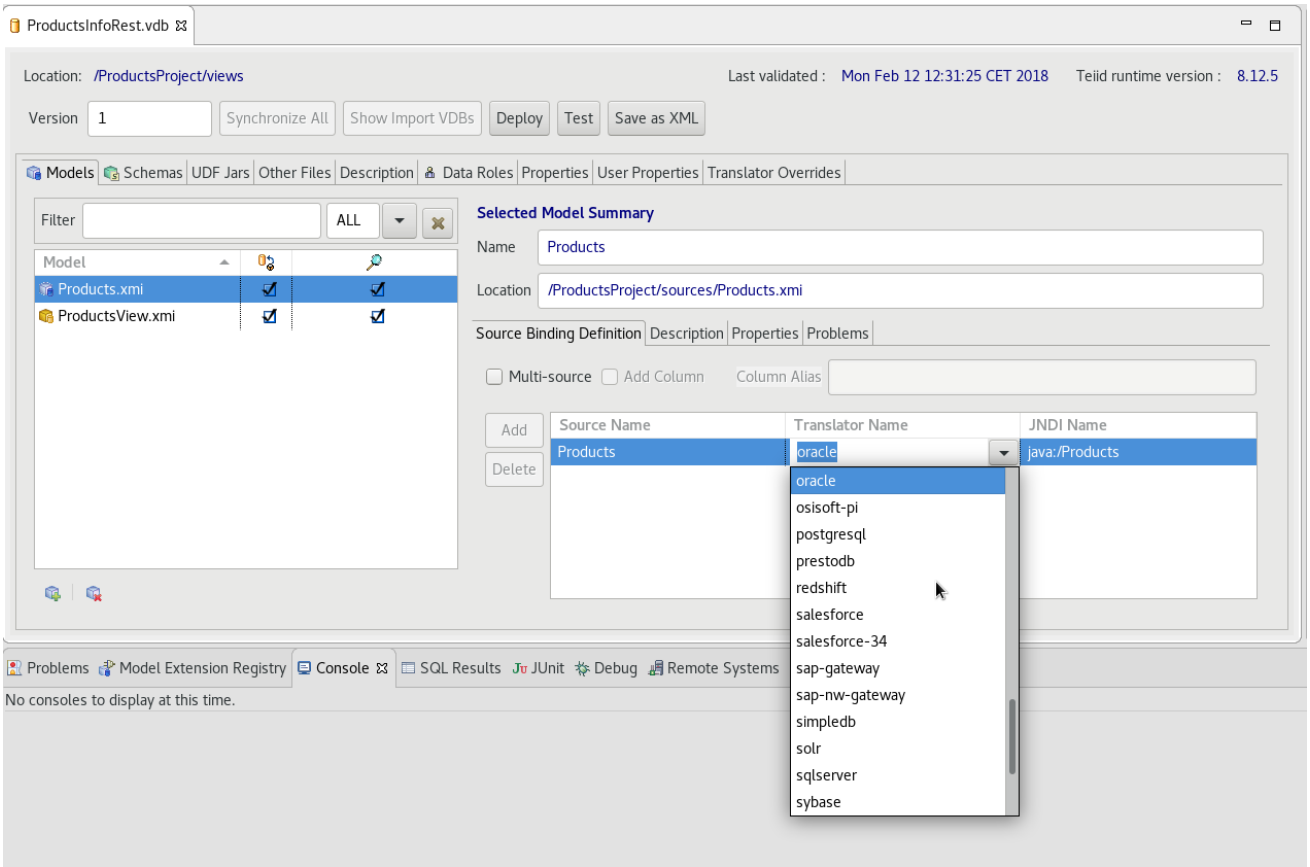


Figure E.19. Change Translator or Data Source Actions

If you have a default JBoss Data Virtualization 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.

E.8.2. Editing Data Roles

Teiid Designer provides a means to create, edit and manage data roles specific to a VDB. Once deployed within a JBoss Data Virtualization 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.

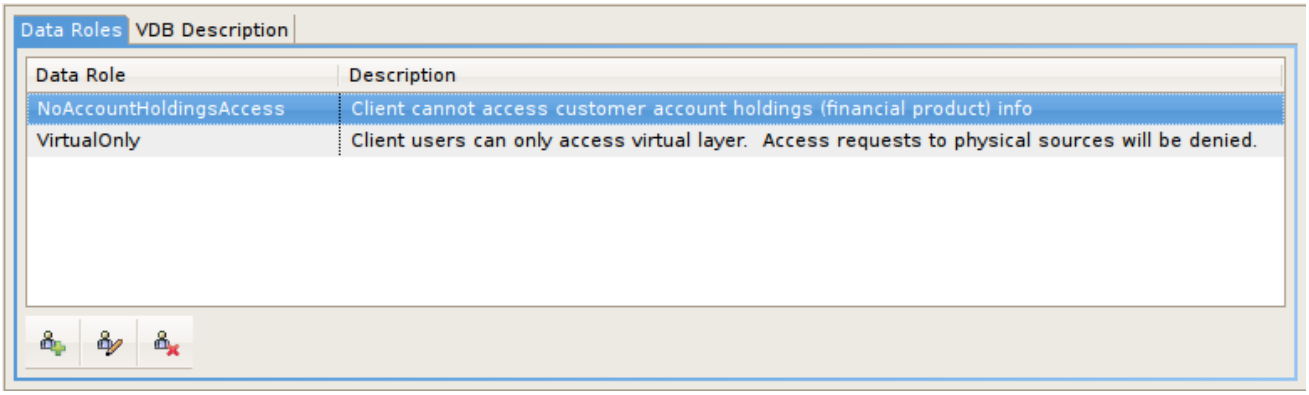


Figure E.20. VDB Data Roles Panel

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

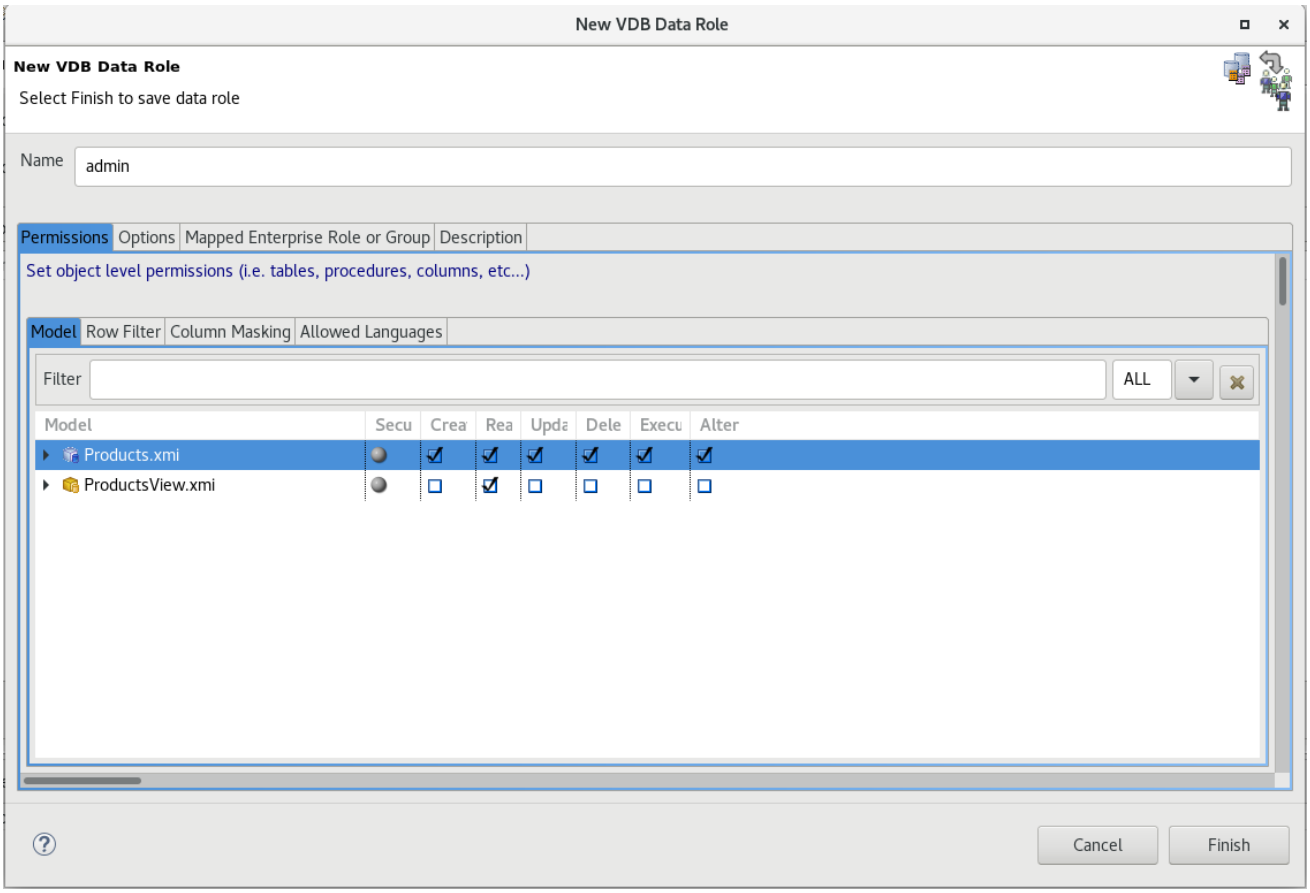


Figure E.21. VDB Data Roles Tab

The **Filter** bar provides full text search capability in the VDB editor. You can also choose a filter by type. The supported types are: ALL, Source, View, Web, and XML Doc. The **X** button clears the search bar and returns it to its default state.

E.8.3. Editing Translator Overrides

Teiid Designer provides a means to create, edit and manage translator override properties specific to a VDB via the **Translator Overrides** tab. A translator override is a set of non default properties targeted for a specific source model's data source. So each translator override requires a target translator name like oracle, db2, mysql, etc. and a set of non-default key-value property sets.

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

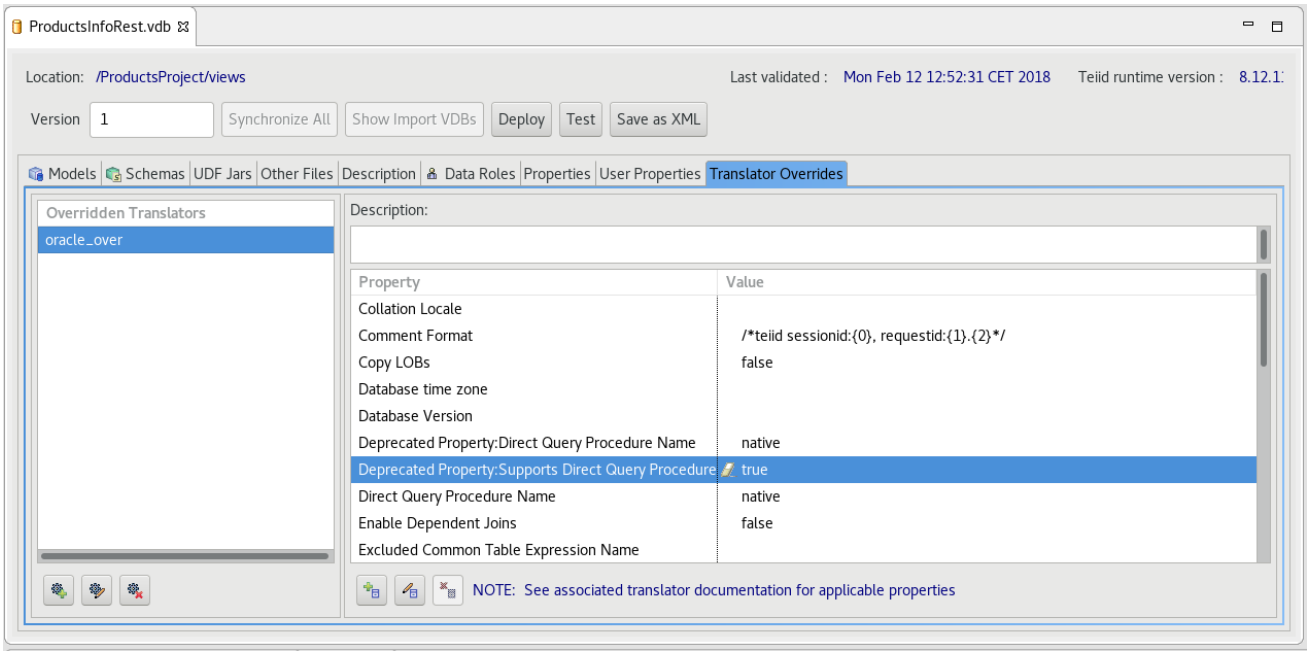


Figure E.22. VDB Translator Overrides Tab

To override a specific translator type, click the add translator action **`(+)`**. If a default JBoss Data Virtualization server instance is connected and available the **`Add Translator Override`** dialog (below) is displayed, select an existing translator type and click **`OK`**. Note that the override is only applicable to sources within the VDB, so be sure and select a translator type that corresponds to one of the VDB's source models. The properties panel on the right side of the panel will contain editable cells for each property type based on the datatype of the property. (i.e. boolean, integer, string, etc.).

Figure E.23. Add Translator Override Dialog

If no default Teiid server instance is available, the **`Add New Translator Override`** dialog is

displayed. Enter a unique name for the translator override (i.e. oracle_override), a valid translator type name (i.e. oracle) and click **OK**. The properties panel on the right side of the panel will allow adding, editing and removing key-value string-based property sets. When editing these properties all values will be treated as type string.

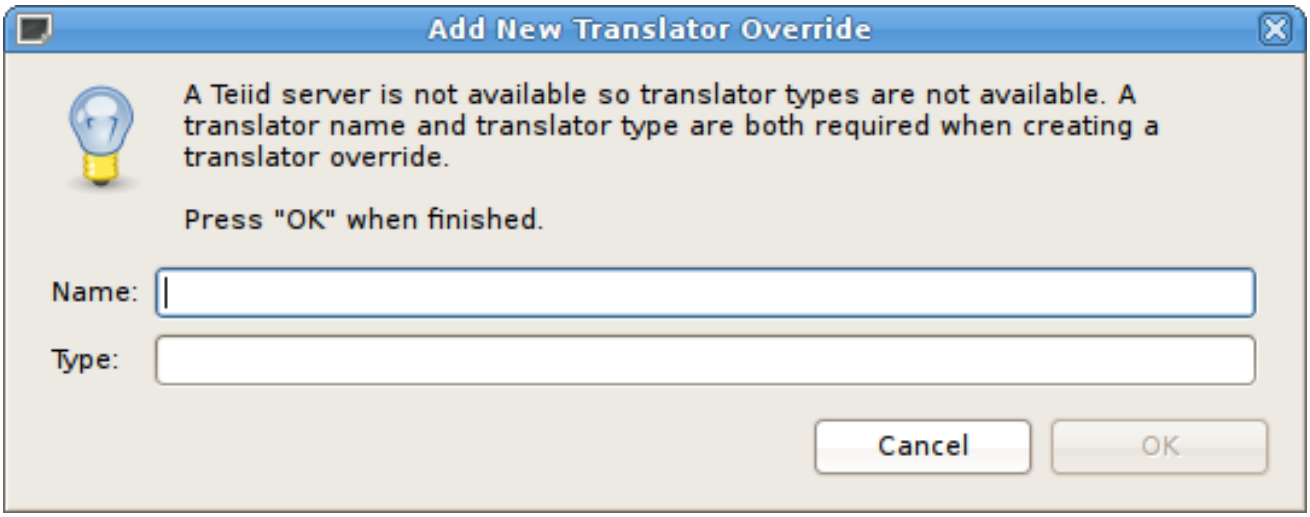


Figure E.24. Add New Translator Override Dialog

E.9. MODEL EXTENSION DEFINITION EDITOR

The MED Editor is a editor with multiple tabs 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:

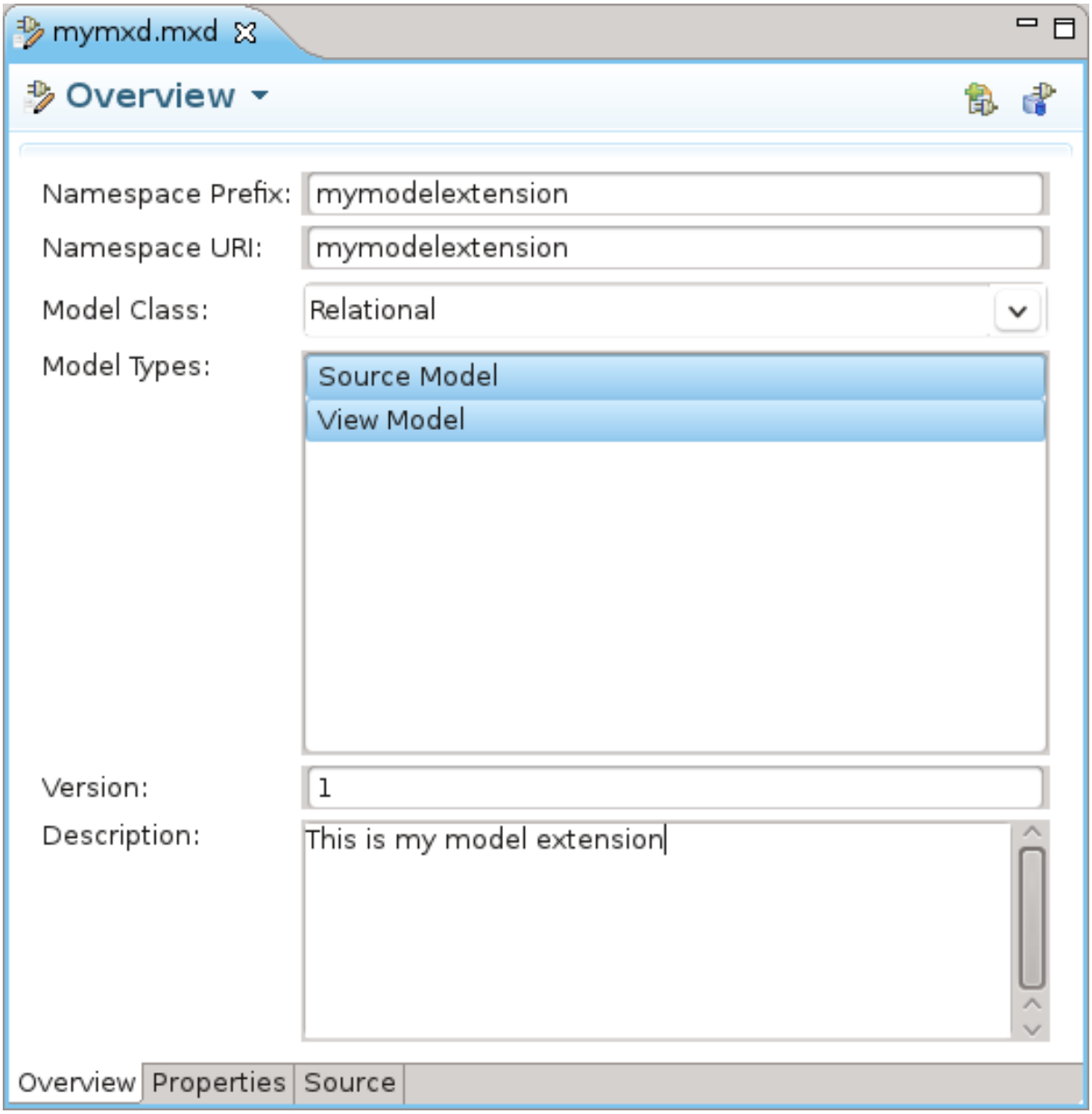


Figure E.25. Overview Tab

- 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:

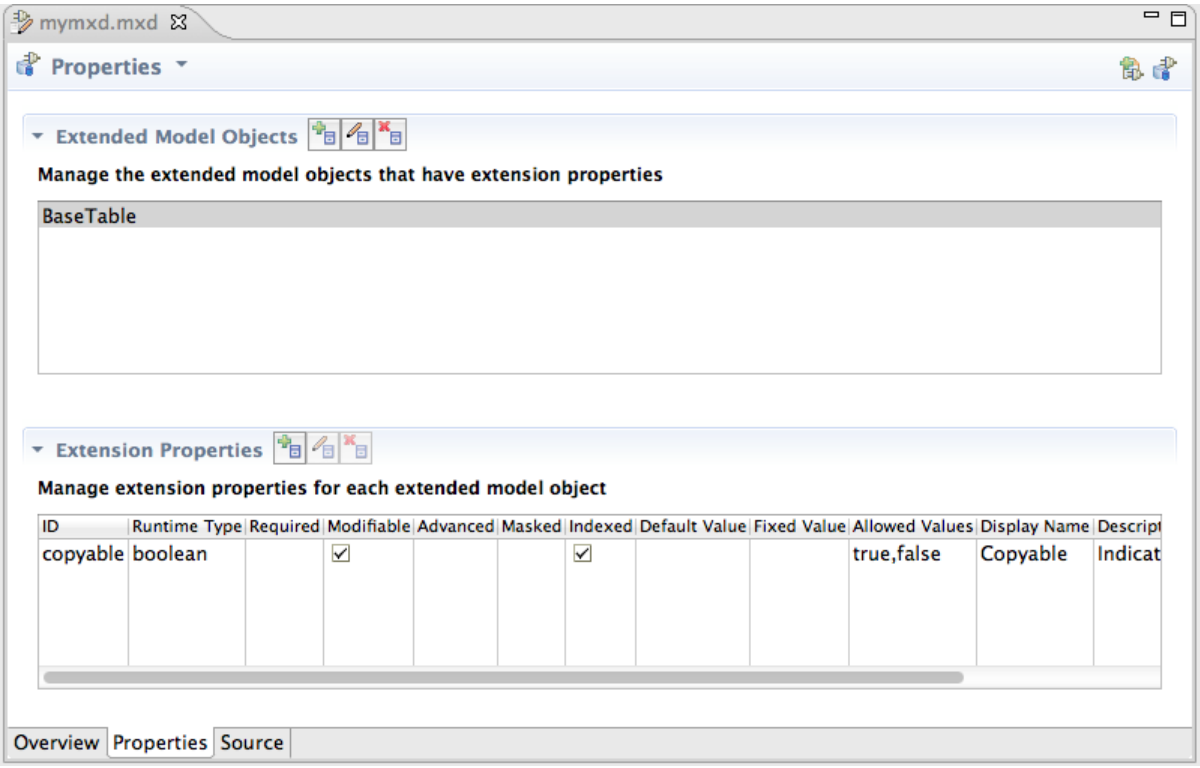


Figure E.26. Properties Tab

- Source - this tab is a read-only XML source viewer 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 E.27. 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.

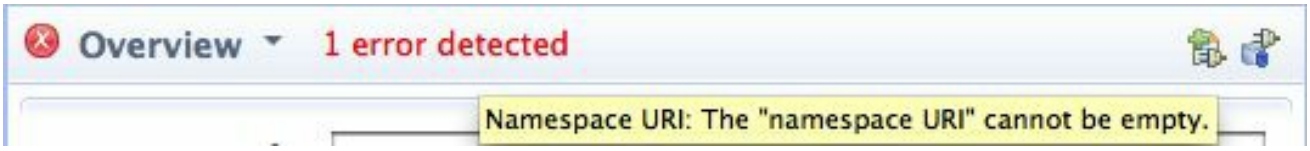


Figure E.28. Shared Header Example

APPENDIX F. TEIID DESIGNER MAIN MENU

F.1. TEIID DESIGNER MAIN MENU

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

These categories include:

- Resource management actions
- Standard edit actions including undo/redo
- Resource actions (i.e. Rename, Move, etc...)
- Find data within your workspace.
- Model level actions
- Custom metadata-related actions
- Change perspectives or add/remove views to your perspective.
- Access available Teiid Designer help documents, Teiid Designer SQL Support Guide and Eclipse Overview information.

Figure F.1. Application Main Menu

F.2. FILE MENU

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

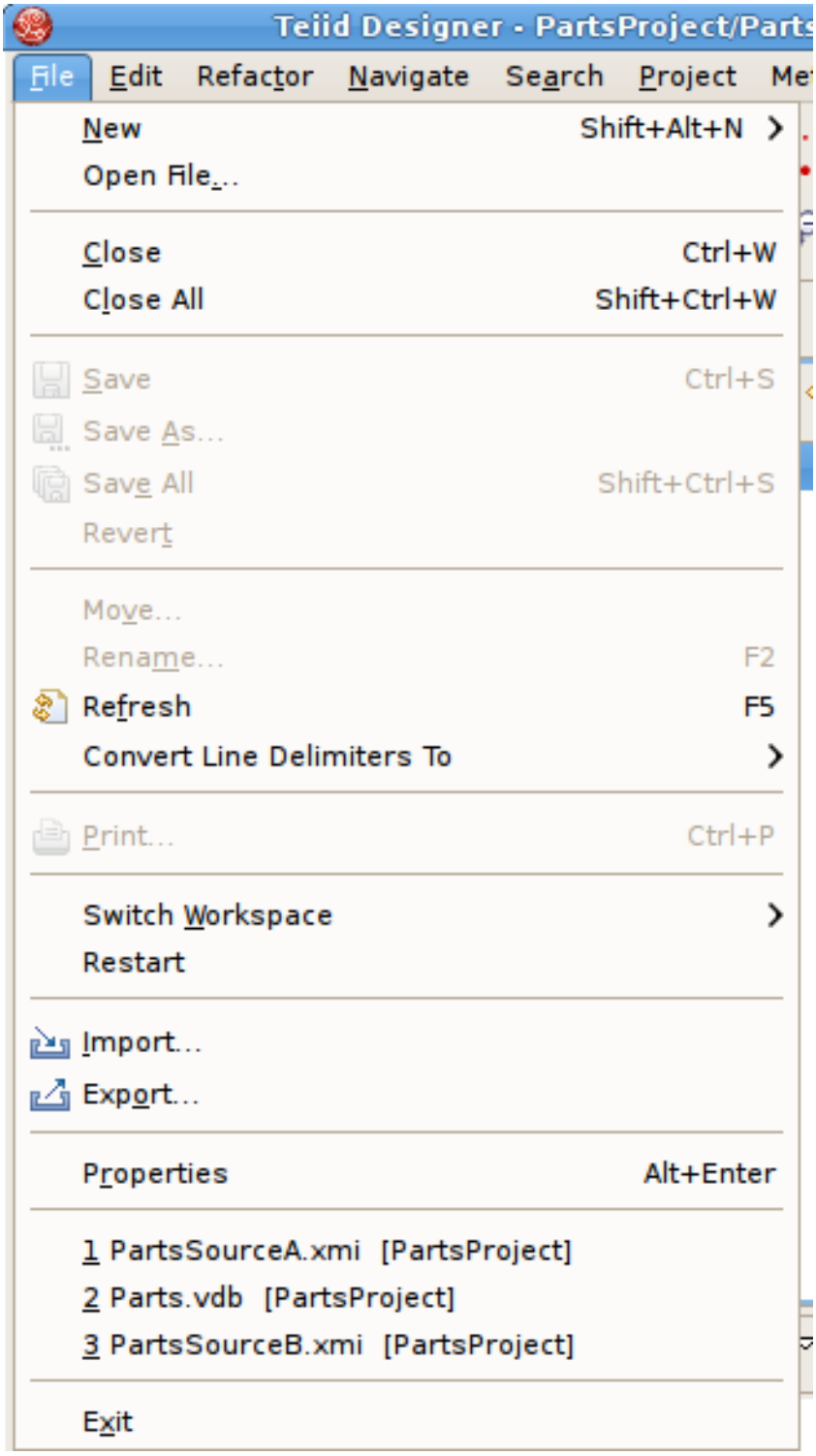


Figure F.2. File Menu

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

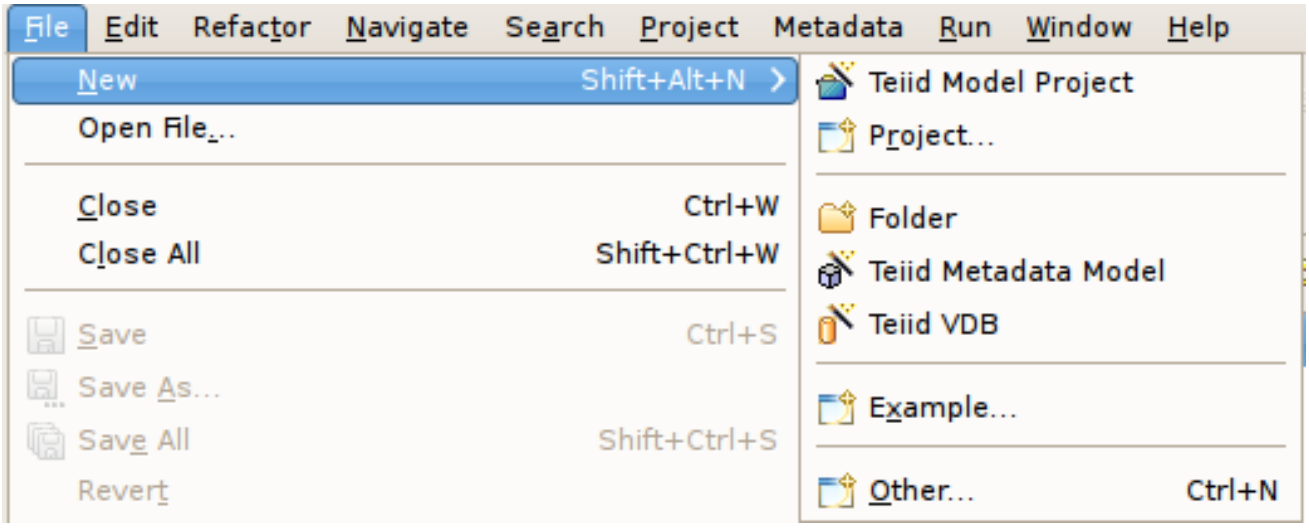


Figure F.3. File Menu

The **`File`** menu contains the following actions:

- New > Model Project - Create user a new model project.
- New > Folder - Create new folder within an existing project or folder.
- New > Model - Create a new model of a specified model type and class using the 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 inline rename is preformed.
- Refresh - Refreshes the resource with the contents in the file system.
- Convert Line Delimiters To - Alters the line delimiters for the selected files. Changes are immediate and persist until you change the delimiter again - you do not need to save the file.

- Print **(Ctrl+P)** - Prints the contents of the active editor. In the Teiid Designer, this action prints the diagram in the selected editor. Allows control over orientation (portrait or landscape), scaling, margins and page order. User can also specify a subset of the pages to print (i.e., 2 through 8).
- Switch Workspace - Opens the Workspace Launcher, from which you can switch to a different workspace. This restarts the Workbench.
- Restart - Exits and restarts the Workbench.
- --> Import - Launches the Import Wizard which provides several ways to construct or import models..
- Export - Launches the Export Wizard which provides options for exporting model data.
- Properties **(Alt+Enter)** - Opens the Properties dialog for the currently selected resource. These will include path to the resource on the file system, date of last modification and its writable or executable state.
- Most Recent Files List - Contains a list of the most recently accessed files in the Workbench. You can open any of these files from the File menu by selecting the file name.
- Exit - Closes and exits the Workbench.

F.3. 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.

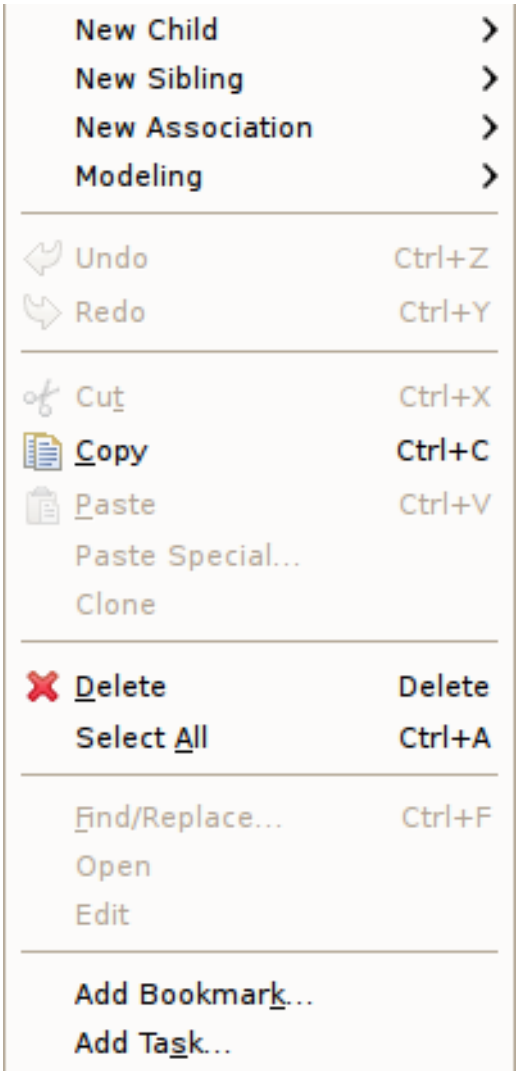


Figure F.4. Edit Menu

The **`Edit`** menu contains the following actions:

- New > Child - This menu is created dynamically to support the creation of whatever types of child objects can be created under the selected object.
- New > Sibling - This menu is created dynamically to support the creation of whatever types of sibling objects can be created under the same parent as the selected object
- New > Association - This menu is created dynamically to support the creation of whatever types of associations can be created with the selected object.
- Modeling > - This menu is created dynamically. Various modeling operations are presented based on selected model object type.
- Undo - Reverses the effect of the most recent command.
- Redo - Reapplies the most recently undone command.
- Cut - Deletes the selected object(s) and copies it to the clipboard.
- Copy - Copies the selected object(s) to the clipboard.

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

F.4. REFACTOR MENU

The **`Refactor`** sub-menu provides Teiid Designer specific actions for file level changes to the models. This sub-menu is in the edit context menu and it is available on the models in the **`Model Explorer`**.

Rename...

Move...

Figure F.5. Refactor Menu

The **`Refactor`** menu contains the following actions:

- Move - Move a model from one folder to another.
- Rename - Rename a model.

F.5. NAVIGATE MENU

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

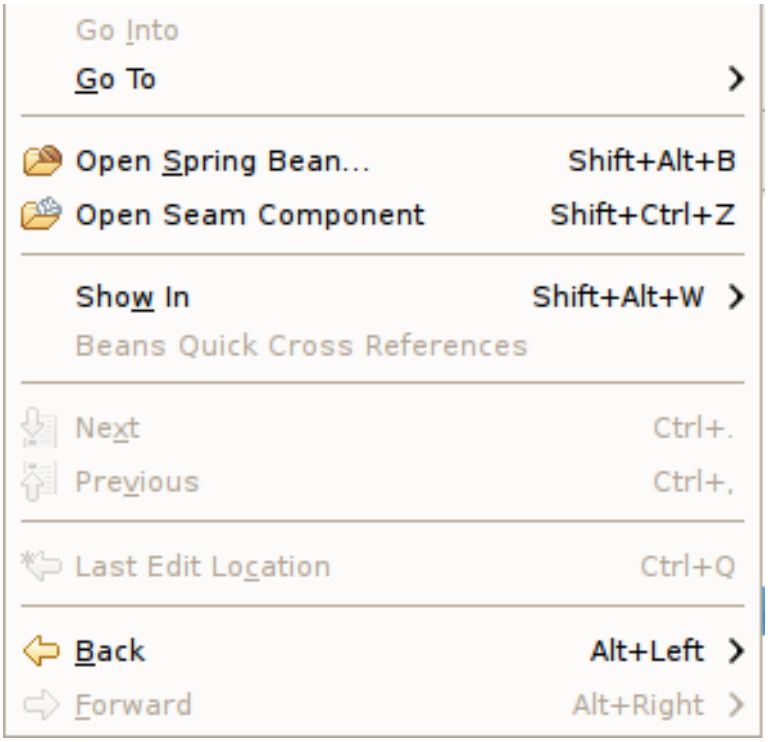


Figure F.6. Navigate Menu

F.6. SEARCH MENU

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

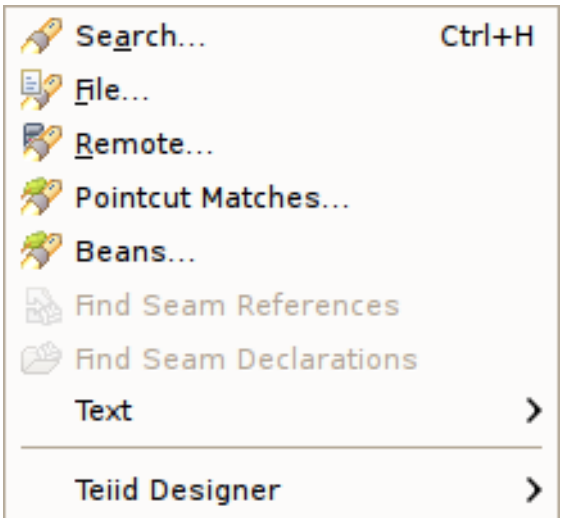


Figure F.7. Search Menu

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

Figure F.8. Search Menu

The individual actions in the Teiid Designer submenu 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 Search Results 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.

F.7. PROJECT MENU

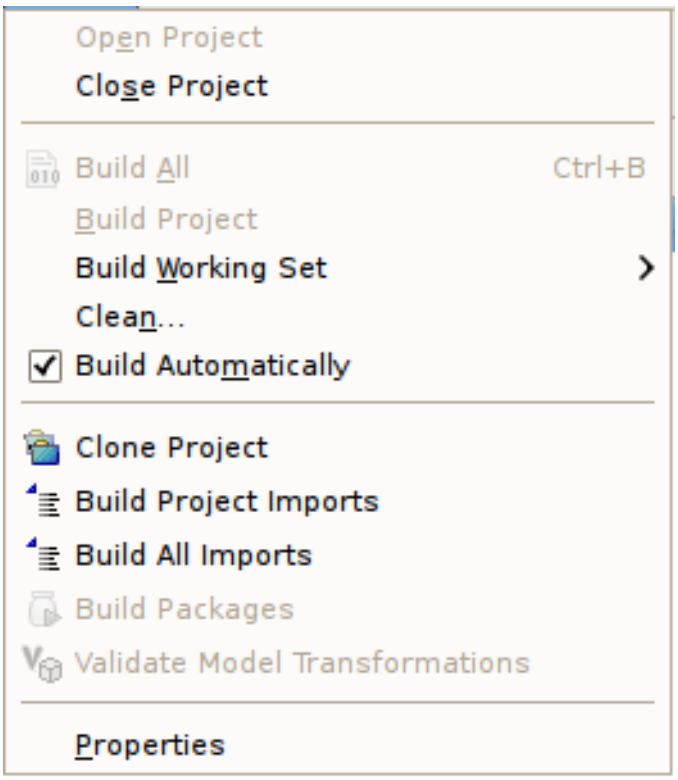


Figure F.9. Project Menu

The individual actions in the **`Project`** menu are described below:

- Open Project - Launches the Open Project dialog.
- Close Project - Closes the currently selected project(s).
- Build All - Validates the contents of the entire workspace. Any errors or warnings will appear in the Problems View.
- Build Project - Validates the contents of the selected project(s). Any errors or warnings will appear in the Problems View.
- Build Working Set - Validates the contents of the selected working set. Any errors or warnings will appear in the Problems View.
- Clean.. - Launches the Clean dialog.
- Build Automatically - Sets the Build Automatically flag on or off. When on, a checkmark 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.

F.8. METADATA MENU

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

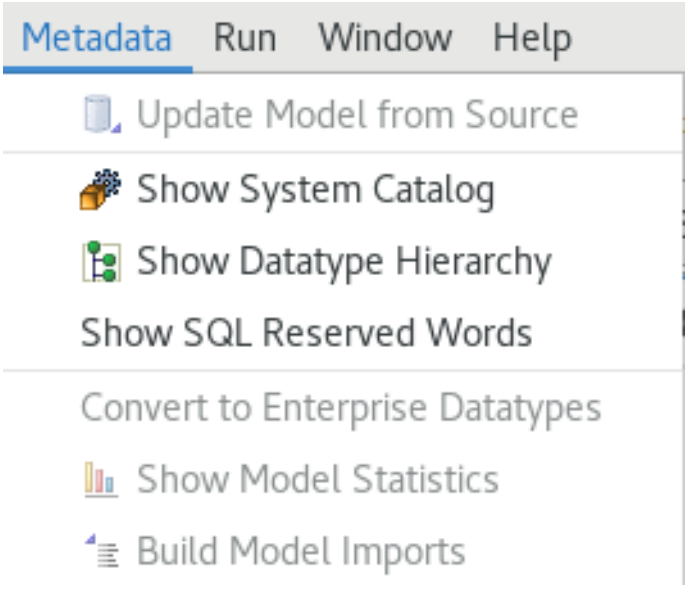


Figure F.10. Metadata Menu

The **Metadata** menu contains the following actions:

- Update Model from Source - If the selected model is a relational source model that was originally created via JDBC Import, then the model will be updated based on changes in the database schema.
- Show System Catalog - Opens the System Catalog view.
- Show Datatype Hierarchy - Opens the Datatype Hierarchy view.
- Show SQL Reserved Words - Opens the SQL Reserved Words view.
- 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.

F.9. RUN MENU

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

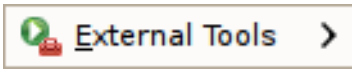


Figure F.11. Window Menu

F.10. WINDOW MENU

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

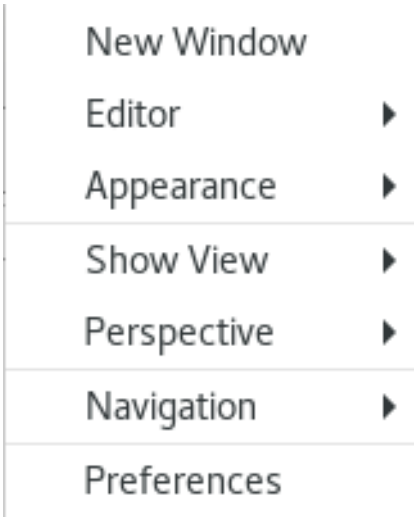


Figure F.12. Window Menu

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

NOTE

These menu items may vary depending on your set of installed Eclipse features and plugins.

To customize a perspective to include one or more Teiid Designer views, click the **`Show View > Other...`** action and expand the **`Teiid Designer`** category to show the available views.

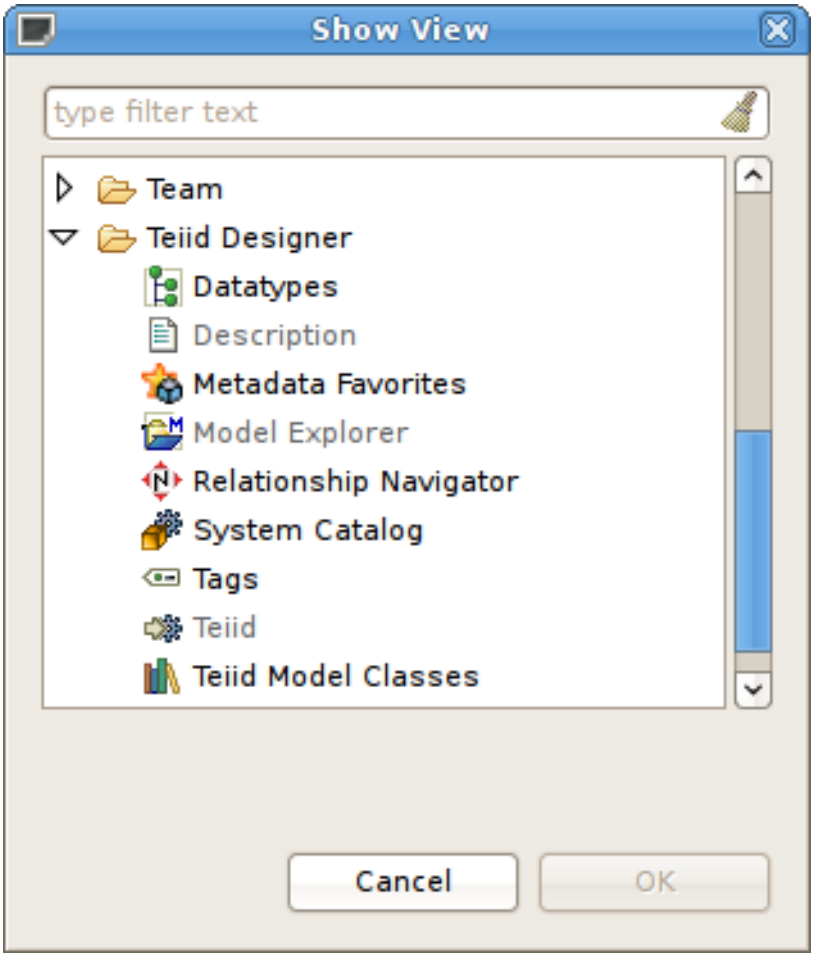


Figure F.13. Show View Dialog

F.11. HELP MENU

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

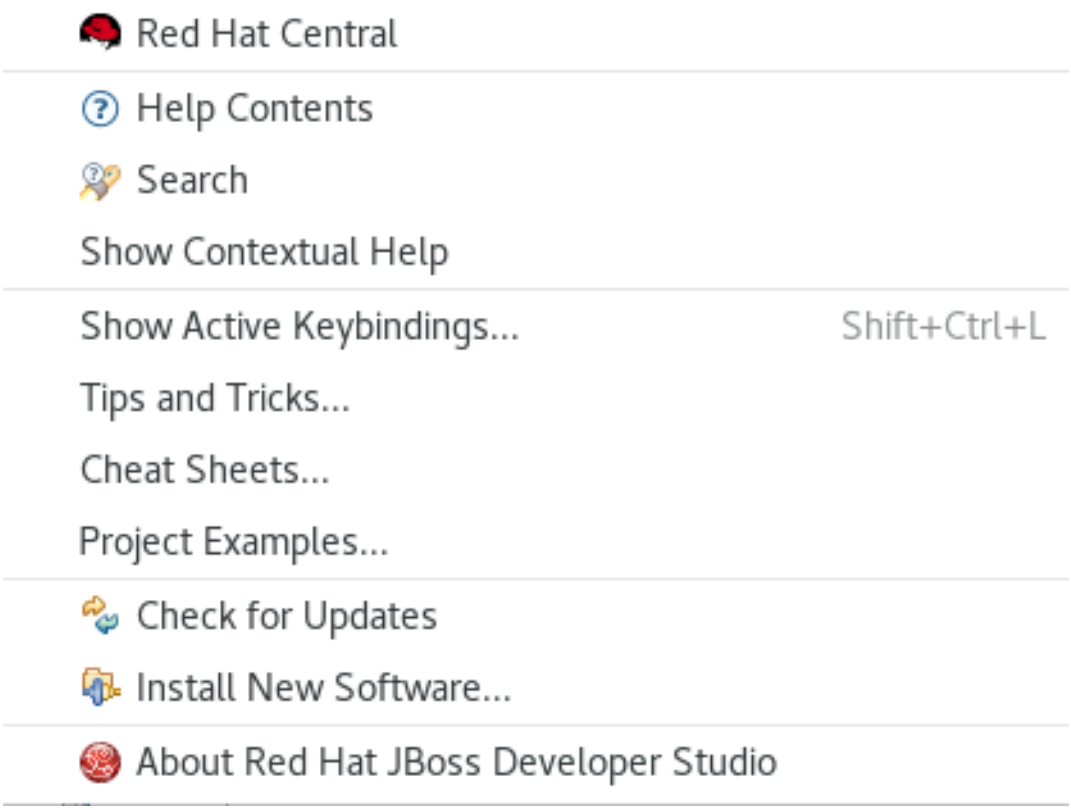


Figure F.14. Help Menu

The individual actions are described below:

- Red Hat Central - Launches the Red Hat Central view.
- 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.
- Show Contextual Help - Launches the Help Search view.
- Show Active Keybindings... - 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.
- 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.

APPENDIX G. REVISION HISTORY

Revision 6.4.0-06 Updates for 6.4.	**Wed Sep 26 2018**	**Barbora Ančincová**
Revision 6.4.0-05 Updates for 6.4.	**Tue Jul 11 2017**	**David Le Sage**