Denodo Model Bridge - User Manual

Revision 20220617

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 OVERVIEW

The Denodo Model Bridge is a graphical tool that enables the integration between Denodo Virtual DataPort (VDP) and third-party modeling tools. Currently, the Model Bridge supports the following modeling tools:

- ER/Studio Data Architect
- ERwin Data Modeler
- IBM InfoSphere Data Architect
- SAP PowerDesigner

and also the following Semantic Web Standards:

- RDF Schema (RDFS) and Web Ontology Language (OWL)

Denodo Model Bridge **transforms data models into VDP models**. It extracts the entities, attributes and relationships from serialized data models and creates the equivalent interface views and associations in VDP.

Given the iterative nature of data models, the Model Bridge can either perform full or incremental synchronizations with VDP models.

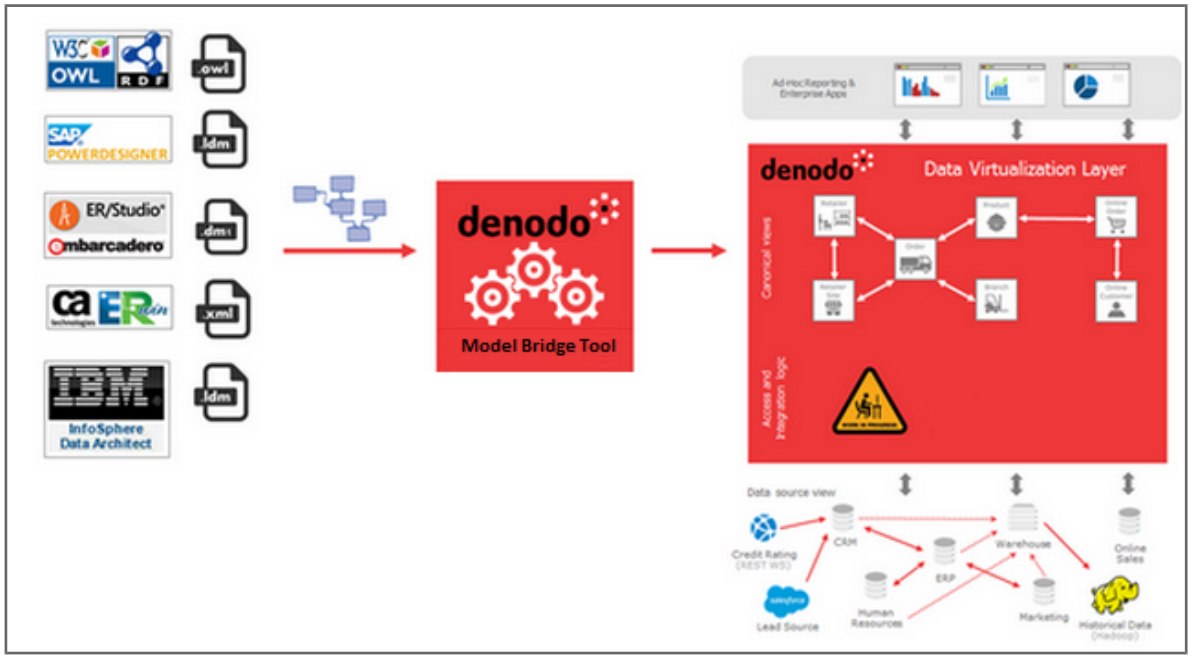

Figure 1 Denodo Model Bridge architecture

2 INSTALLATION

The Denodo Model Bridge distribution consists of:

- A set of command-line executable scripts for Windows and Linux (`/bin` folder)
- Configuration files for both the tool and its logging system (`/conf` folder)
- A series of Java jar binary files (`/lib` folder)

For installing it just download the `.zip` file and extract the tool into the desired folder.

In order to run the Denodo Model Bridge you will need a JRE or JDK version 6 or later, and a `PATH` environment variable correctly configured to run it from the console. You can check your Java installation by running the following command on a console.

```
$ java -version
```

If you get a version number, which needs to be **1.6 or later**, you are ready to start using the Denodo Model Bridge. It is also required that you define the `JAVA_HOME` environment variable.

3 DATA MODELING

Modeling tools are used to design physical and logical data models. These models illustrate the specific **entities**, **attributes** and **relationships** involved in a business function.

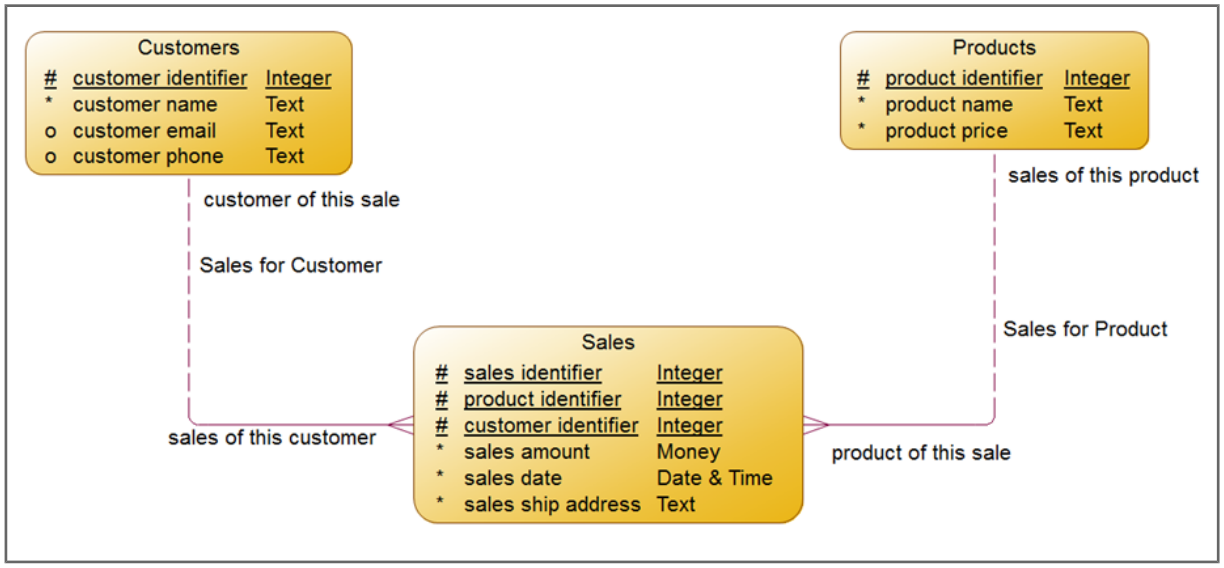

Figure 2 Logical data model

Modeling tools help to enforce modeling and business data policies: mandatory fields, required formats, etc. This way, when these models are implemented by Data Virtualization (DV) developers, everyone will have to follow those conventions.

Integration between a Data Virtualization solution, like VDP, and modeling tools can go in both directions:

1. **From VDP to the modeling tool:**

   VDP exposes metadata through standard interfaces such as JDBC/ODBC that can be used by most modeling tools to import Denodo data models. Metadata exposed by VDP includes views and relationships between those views; associations between views appear as primary key - foreign key relationships.

2. **From the modeling tool to VDP**:

   VDP allows creating views and associations either graphically or via VQL. The **Denodo Model Bridge** automates this process **creating *interface views* and *associations*** from the models generated by modeling tools. Referential Constraints will be registered for those associations that meet the conditions to do so in Denodo VDP.

Interfaces are a special type of views that consist only of a definition of fields and a reference to another view: the *implementation view*. You can use them to do **top-down design** where you first define the fields of the *interface* and at a later stage, associate it with its *implementation view*.

When designing data services in VDP following a top-down design, being able to define and publish canonical models via *interface* views, before implementing the access and integration logic to populate them, decouples the work of Data Virtualization developers from that of Client Application developers, who build informational/operational solutions on top of the data services.

3.1.1 RDFS and OWL ontologies

OWL represents data structure with classes, object properties and datatype properties. The Model Bridge has to transform OWL metadata to an Entity-Relationship model, but it does not support all the (very complex) semantics of OWL. The following is an explanation of which concepts of OWL the Model Bridge is able to transform to VDP and how.

OWL classes are transformed into interfaces in VDP. Classes can be organized into a hierarchy using the `subclassof` tag, but in VDP this concept of inheritance does not exist, so all subclasses are created separately with their own attributes and associations, adding the attributes and associations of their superclasses to each of the resulting interface views.

Datatype Properties are translated into columns of an interface view, the type of these columns is always translated as `text`.

Object Properties will define associations between views in VDP. Cardinality is restricted by upper and lower bounds. In absence of explicit cardinality boundaries, `1:n` cardinality is adopted. If the two sides of the relation express `1:n` cardinality, this association would be unsyncable in VDP, because VDP does not support modeling associations `n:m` directly (i.e without a relationship view).

In OWL there are Object Properties that have their inverse (that's its way to define bidirectional relationships). In VDP there are only bidirectional associations, so even if there is no inverse defined at the OWL side, VDP associations will always be bidirectional. The Classes that take part in an association are extracted from the `range` and `domain` tags of an Object Property or from a restriction of a subclass, if the object property does not have `range` and `domain`. But the inverse side only can be extracted from the Object Property and not from the restrictions. The application supports the union of entities within an Object Property, if this union is in the restriction, creating an association for every entity of the union in VDP.

OWL does not have the concept of primary keys, so in all interfaces an attribute called "`self_uri`" is created, which is used in the mapping of the associations. In addition, another attribute is created with the name of the Object Property at the other side of the `1:n`. This attribute is used at the other side of the association in the condition of the mapping.

At the end of this document, in **Appendix A**, the OWL syntax supported by the Model Bridge is further detailed.

4 USAGE

The Denodo Model Bridge is started using a command-line script:

```
bin> denodo-modelbridge.bat
```

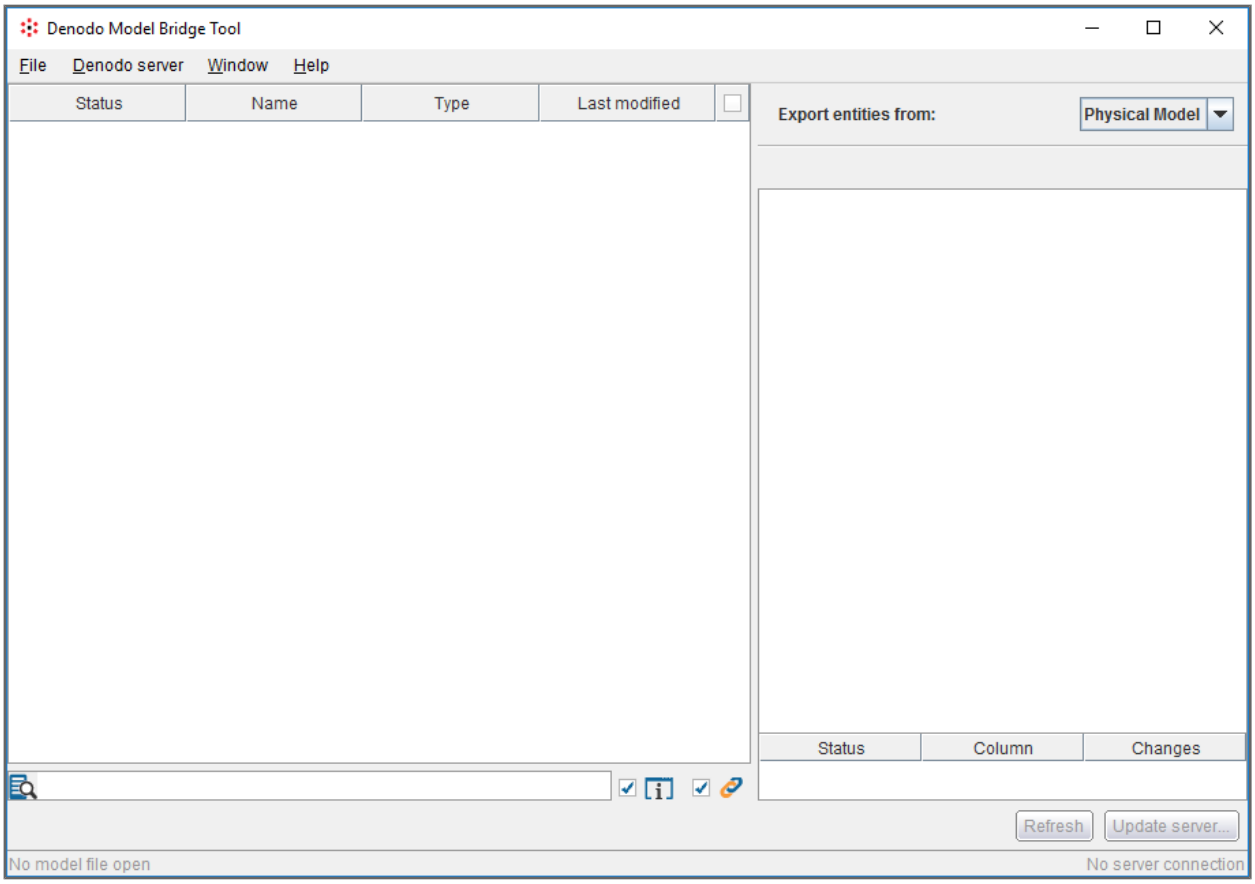

Figure 3 Denodo Model Bridge GUI

Then, the user has to select the file that contains the model created with the modeling tool:

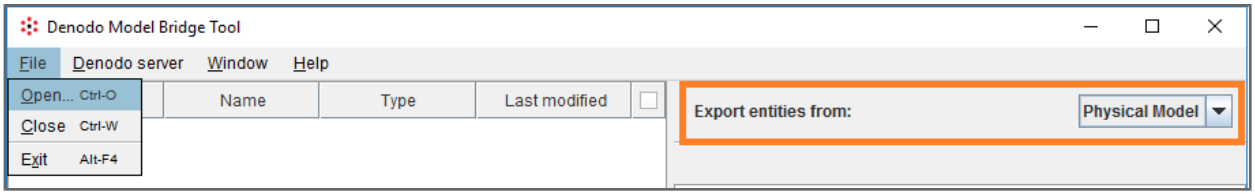

Figure 4 Starting point of the Tool

The Denodo Model Bridge currently supports the following file types:

- IDERA ER/Studio Data Architect: `.dm1` files with **logical or physical** models(only relational models).
- ERwin Data Modeler: `.xml` files with **logical or physical** models.
- IBM InfoSphere Data Architect: `.ldm` files with **logical or physical** models.

- SAP PowerDesigner: `.ldm` files with **logical or physical** models.
- RDFS and OWL ontologies: `.owl` and `.rdf` / `.rdfs`.

Once the user chooses the file to open and the kind of model to export from it (from the drop-down list in orange in the image above) the Model Bridge parses the entities, attributes and relationships and shows information about them. In any case, there could be repeated names among the entities, this is not allowed in VDP, and for this reason Denodo Model Bridge will rename the entity by adding "_x" to the name where x is a number. For every element the tool displays:

- Status of the element with respect to VDP: `Not_connected` in the image below because no connection to VDP has been established yet.
- Name
- Type: `Interface` or `Association`.
- VQL to create the element in VDP.

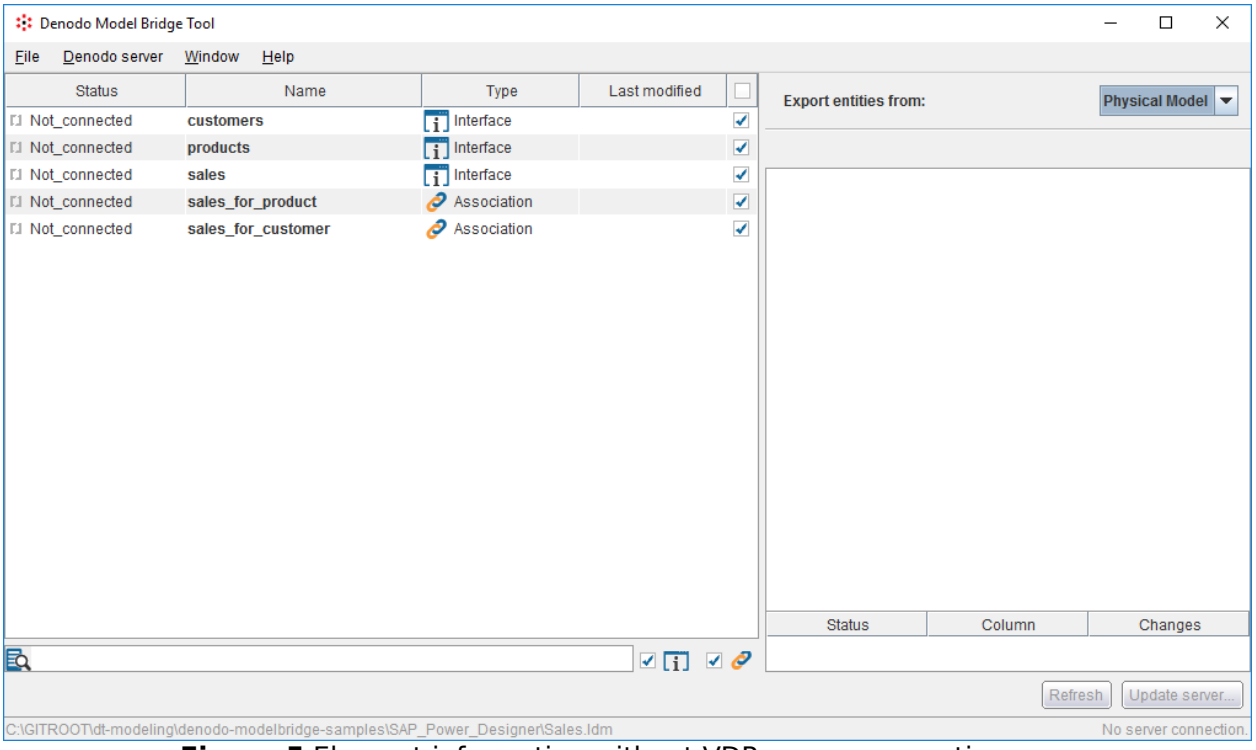

Figure 5 Element information without VDP server connection

4.1.1 Identifiers charset configuration

Model Bridge supports Unicode characters in the name of its elements: views and their fields and associations.The user can select one of the following charsets:

- Restricted (default one):
  Element identifiers follow these rules:

  1. The first character has to be one of these:
     - a to z. If you enter uppercase letters, they are **converted to lowercase**
     - Unicode characters with the code point from 200 to 377
  2. The next characters have to be one of these:
     - a to z
     - Numbers
     - Underscore: _
     - Unicode characters with the code point from 200 to 377

- Unicode:
  Identifiers of elements can contain any case and any character, except the underscore, _, as the first character.

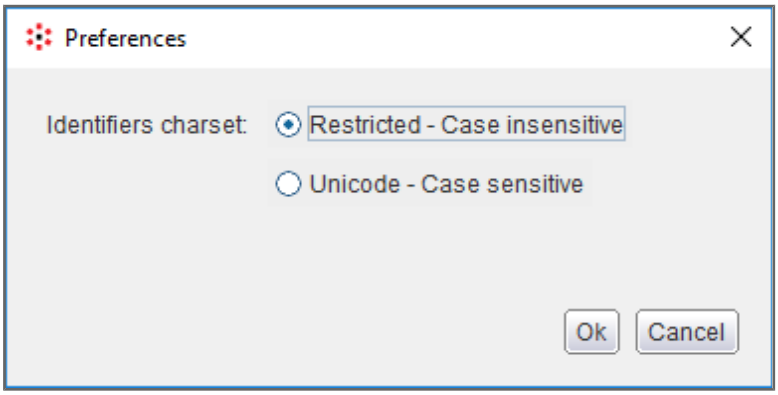

Figure 6 Identifiers charset configuration

Before synchronizing the model with VDP, the Model Bridge has to establish a connection with a VDP server:

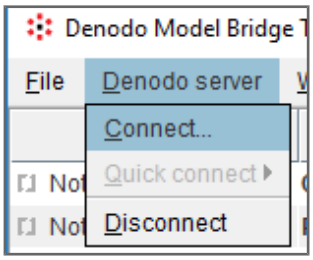

Figure 7 Connecting to VDP server

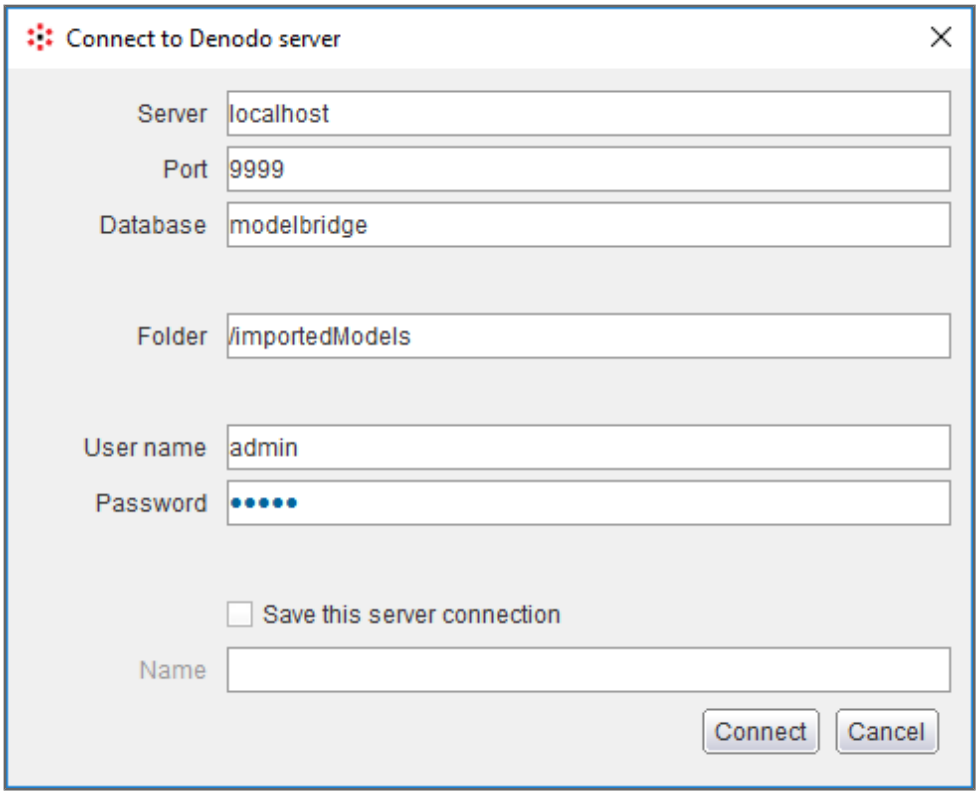

Figure 8 VDP server connection

The following parameters must be supplied:

- `Server`: host where VDP is running.
- `Port`: port where VDP is listening to connections Default port is 9999.
- `Database`: VDP database.
- `Folder`: target folder where the elements will be created. It is an optional parameter.
- `Login`: login to access VDP. The user name should have **admin privileges** on the specified database as operations in VDP are performed in single user mode.
- `Password`: password to access VDP.

4.1.2 How to store VDP connections with encrypted passwords

In the `Connect to Denodo server` dialog there is an option for saving the connection parameters for its later use. Passwords are stored in plain text by default.

But users can decide to store their passwords encrypted by simply supply an encryption password to the Model Bridge Tool in the `DENODO_MODELBRIDGE_ENCRYPTION_PASSWORD` environment variable or JVM system

property.

4.1.3 Connecting to an SSL-enabled server

If the VDP server has SSL enabled the Model Bridge might need some configuration to be able to establish a connection with VDP.

If you use the trust store by default and if the server's public key is signed by a well known international Certificate Authority (CA) no further configuration will be required.

If not, you should configure the trust store you are using by opening the file `<DENODO_MODEL_BRIDGEHOME>/conf/configuration.properties` defining the following property and make sure a valid path for the trust store is specified for it:

```
com.denodo.security.ssl.trustStore=path to the TrustStore
```

Once the Model Bridge establishes a successful connection with VDP the entities and relationships loaded from the file are searched in VDP by querying its catalog. For every element the tool displays:

- Status of the element with respect to the VDP server: `New`, `Updated` or `Ok`.
- Name
- Type: `Interface` or `Association`.
- Date of the last modification in VDP, if the element already exists in the VDP server.
- VQL to create the element in VDP.
- Changes between the element in the Model Bridge and the one in VDP, in case the status of the element is `Updated`. Possible changes are: `Added`, `Updated` or `Deleted`.

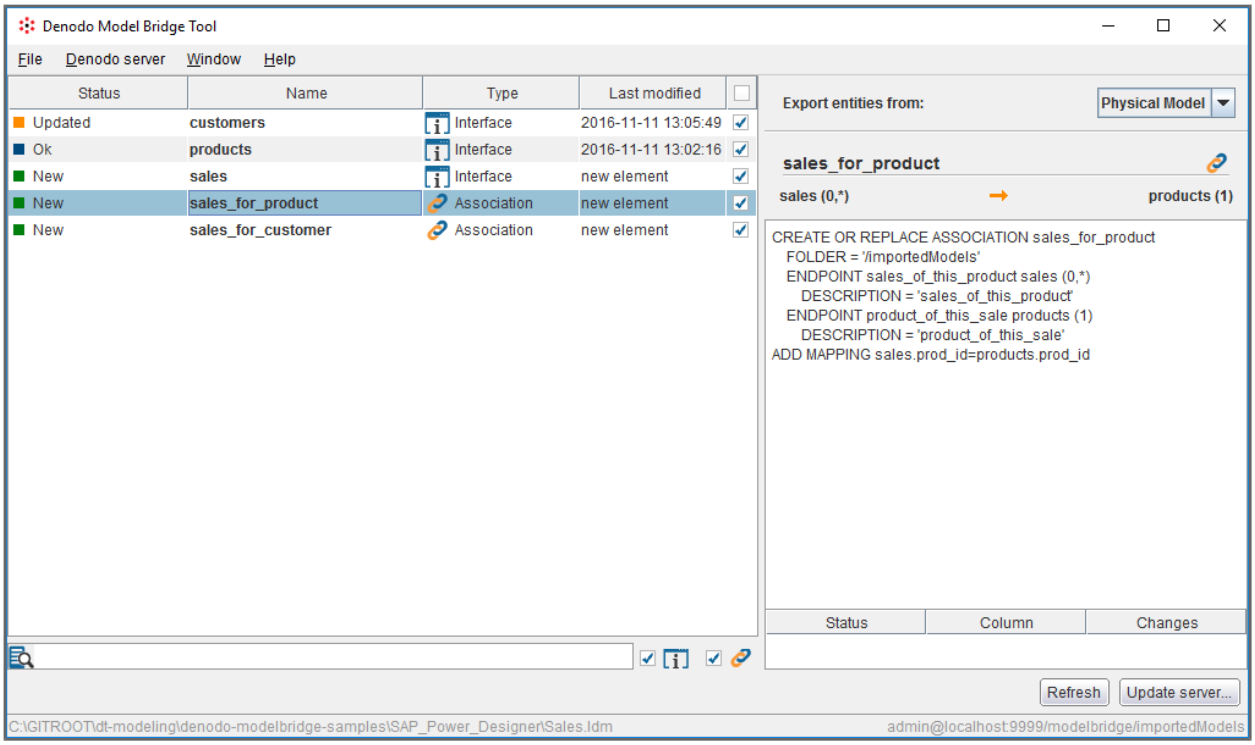

Figure 9 Elements information with VDP server connection

Whenever the user clicks the `Refresh` button the Model Bridge will search for the displayed elements in the VDP catalog and it will show the information mentioned above.

By clicking the `Update Server…` button the new elements will be created in the VDP server and the updated ones will be modified accordingly.

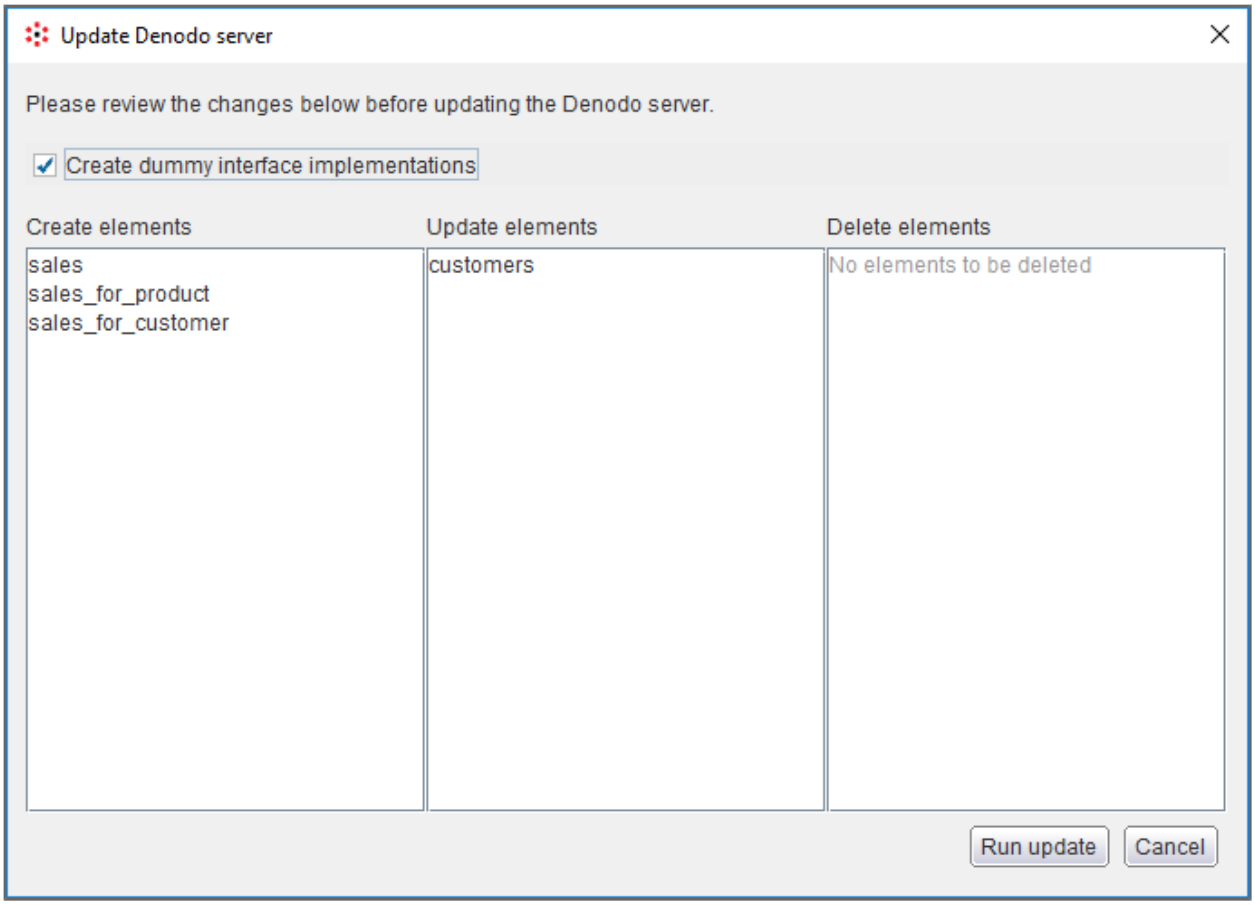

Figure 10 Update VDP server

The update process offers us to `Create dummy interface implementations`. When selected, the Model Bridge will create a view with a single row of null values that will be used as the implementation for each interface in the model.

The image below shows the result of the update process for the model used in this manual as an example. In the `importedModels` folder there are three interfaces and two associations, with the corresponding dummy implementation for the interfaces.

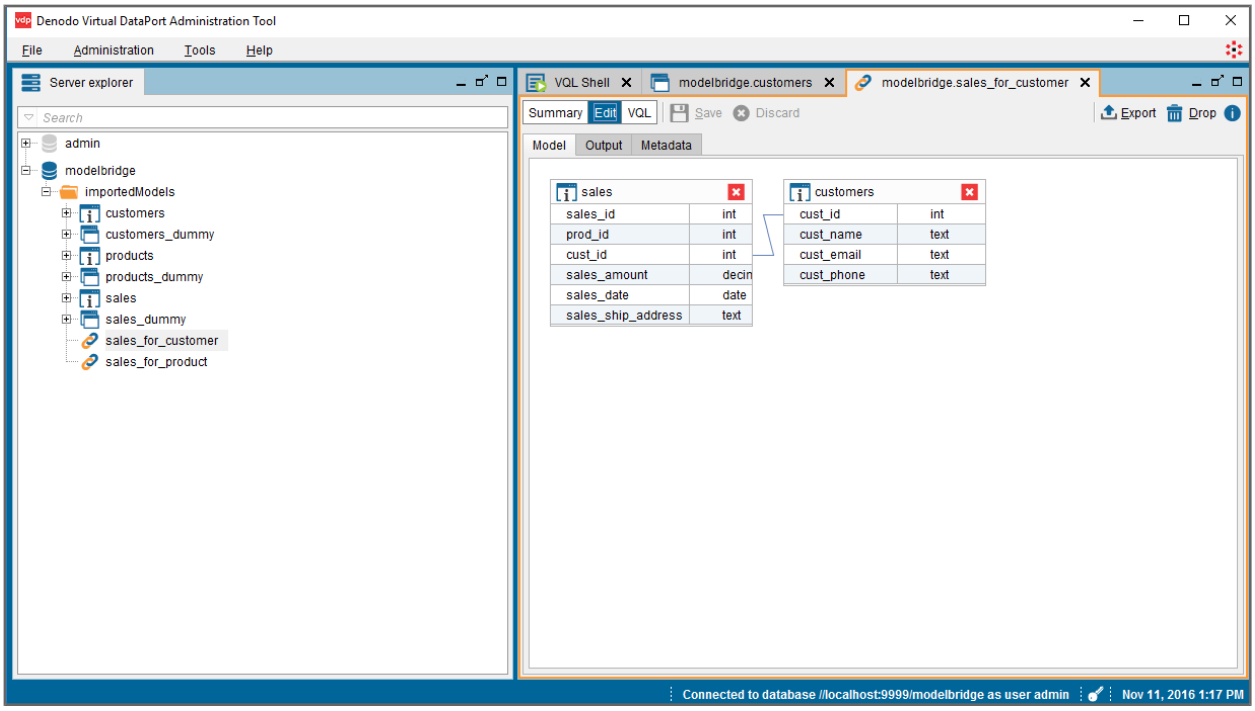

Figure 11 Model created in VDP

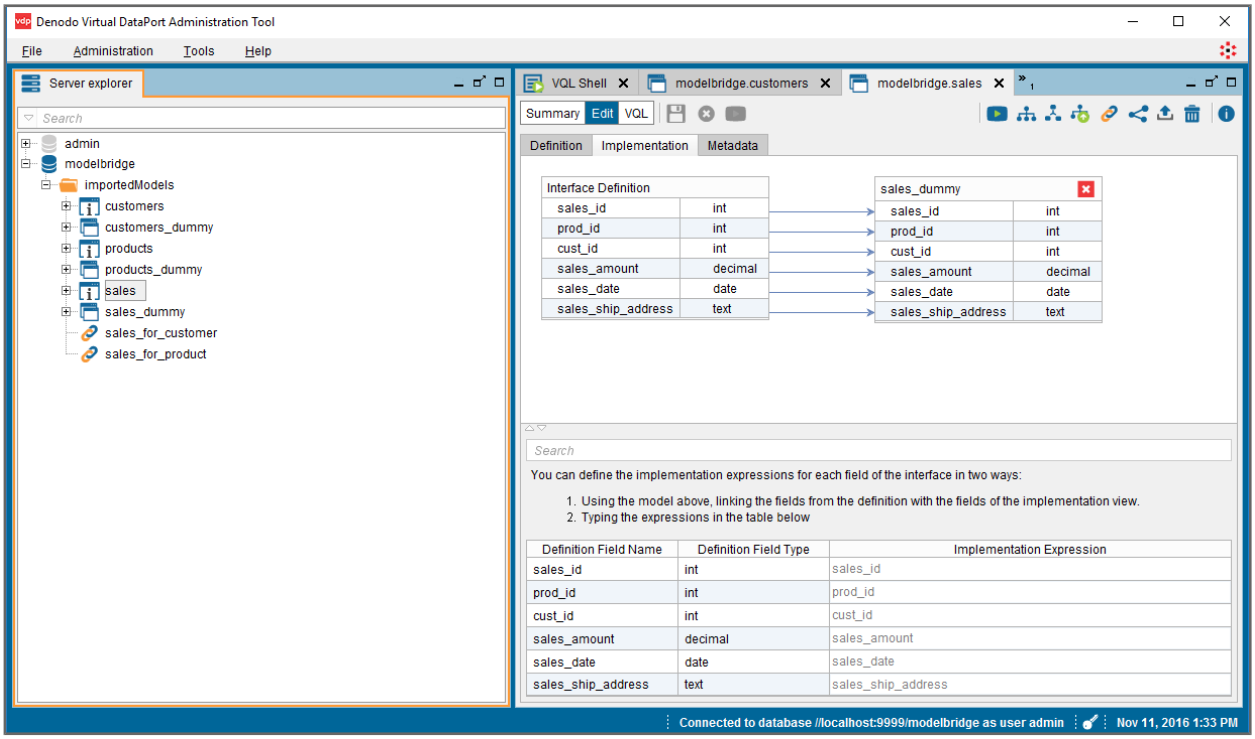

Figure 12 Definition of an interface implementation

5 LIMITATIONS

Primary keys

Primary keys of the model will not be reflected in VDP models as in VDP you cannot define primary keys for interface views.

Not-null constraints

As with primary keys, VDP interface views don’t allow the definition of other types of constraints on specific columns, such as not-null constraints. So these are ignored when synchronizing to the VDP database.

No *forced synchronization* operation

The current version of the tool does not allow *forcing* the synchronization of elements that the tool already considers in “OK” state.

STRUCT and ARRAY types

The types of BIG SQL, STRUCT and ARRAY, are converted to TEXT when you import the model from IBM InfoSphere Data Architect.

Logical Models with associations many to many

It is not possible to translate an association `many to many` from a logical model to VDP, it does not support this kind of association. For instance a logical model from IBM InfoSphere Data Architect could contain an association `many to many` and it wouldn't be syncable with VDP, and in the status of the interface would appear as `UNSYNCABLE`.

Associations without mappings

If there is an association that hasn't got a mapping, this association is not syncable with a VDP server, and the status of the interface would appear as `UNSYNCABLE`.

Synchronization Referential Constraints

The current version cannot detect if in an association of VDP the property Referential constraint is checked. So, this property is ignored to obtain the status of the associations with regard to VDP.

Data Types in RDFS/OWL

All attribute data types in OWL classes are translated to VDP as type *text*.

Dimensional models in ER/Studio

Dimensional models are not supported by Denodo Model Bridge, only relational ones are supported.

6 TROUBLESHOOTING

Unable to find valid certification path to requested target

Symptom
The Model Bridge tries to connect to a VDP server and shows the following error: "`sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target`"

Resolution
VDP Server has SSL enabled, so it is necessary to specify the location of the trust store in the configuration of the Model Bridge.

In order to enable SSL connections between VDP Server and the Model Bridge see section **Connecting to an SSL-enabled server**.

7 APPENDIX A

This appendix is a table with the tags supported by The Model Bridge and their contexts. Every tag has to be within its context to be transformed to VDP structures.

tags	context
<owl:DatatypeProperty>	
<owl:ObjectProperty>	
<rdfs:domain>	<owl:ObjectProperty> *<owl:DatatypeProperty>*
<rdfs:range>	<owl:ObjectProperty> *<owl:DatatypeProperty>*
<owl:inverseOf>	<owl:ObjectProperty>

<owl:Class>	
<rdfs:Class>	
<rdfs:subClassOf>	<owl:Class> <rdfs:Class>
<owl:Restriction>	<rdfs:subClassOf>
<owl:onProperty />	<owl:Restriction>
<owl:allValuesFrom>	<owl:Restriction>
<owl:someValuesFrom>	<owl:Restriction>
<owl:hasValue>	<owl:Restriction>
<owl:onClass>	<owl:Restriction>
<owl:maxCardinality>	<owl:Restriction>
<owl:maxQualifiedCardinality>	<owl:Restriction>
<owl:minCardinality>	<owl:Restriction>
<owl:minQualifiedCardinality>	<owl:Restriction>
<owl:cardinality>	<owl:Restriction>
<owl:qualifiedCardinality>	<owl:Restriction>
<owl:unionOf>	<owl:allValuesFrom><owl:Class>
<rdf:rest>	<owl:unionOf>
<rdf:first>	<owl:unionOf>