Installing a JDBC Driver

DbVisualizer bundles JDBC drivers for most common databases, so typically you do not need to install a JDBC driver.

This page describes the way JDBC drivers are managed in DbVisualizer. If a JDBC driver for your database is bundled with DbVisualizer, see Driver Info on the Supported Databases page, you typically do not need to read this chapter.

If, however, any of the these things apply to you, keep on reading:

- want to learn how the Driver Manager in DbVisualizer works,
- need to have several versions of the same JDBC driver loaded simultaneously,
- need to add a Driver that does not exist in the list of default drivers .

What is a JDBC Driver?

DbVisualizer is a generic tool for administration and exploration of databases. DbVisualizer does not deal directly with how to communicate with each database type. That job is done by a JDBC driver, which is a set of Java classes. All JDBC drivers conform to the JDBC specification and its standardized Java programming interfaces. This is what DbVisualizer relies on. A JDBC driver implements all details for how to communicate with a specific database and database version, and there are drivers available from the database vendors themselves as well as from third parties. To establish a connection to a database, DbVisualizer loads the driver and then gets connected to the database through the driver.

It is also possible to obtain a database connection using the Java Naming and Directory Interface (JNDI). This technique is widely used in enterprise infrastructures, such as application server systems. It does not replace JDBC drivers but rather adds an alternative way to get a handle to an already established database connection. To enable database "lookup's" using JNDI, an Initial Context implementation must be loaded into the DbVisualizer Driver Manager. This context is then used to lookup a database connection.

The following sections describe the steps for installing a JDBC Driver, and also how to configure DbVisualizer to use JNDI to obtain a database connection.

Get the JDBC driver file(s)

DbVisualizer comes bundled with all commonly used JDBC drivers that have licenses that allow for distribution with a third party product. Currently, drivers for DB2, H2, JavaDB/Derby, Mimer SQL, MySQL, NuoDB, Oracle, PostgreSQL, SQLite, Vertica as well the jTDS driver for SQL Server and Sybase, are included with DbVisualizer. If you only need to connect to databases of these types, you can skip the rest of this chapter and jump straight to the Creating a Connection page, because by default, DbVisualizer configures all these drivers automatically the first time you start DbVisualizer.

If you need to connect to a database that is not supported by a bundled JDBC driver, you must get a JDBC driver that works with your database type and version. The following web page contains an up-to-date listing of the database/driver combinations we have tested:

http://www.dbvis.com/doc/database-drivers/

To find a JDBC driver for your database, go to the database vendor's website or search for the name of the database plus the word `JDBC`.

Download the driver to an appropriate directory. Make sure to read the installation instructions provided with the driver. Some drivers are delivered in ZIP or JAR format but need to be unpacked to make the driver files visible to the Driver Manager. The Databases and JDBC Drivers web page describes where you can download some drivesr and also what additional steps may be needed to install and load the driver in DbVisualizer.

Drivers are categorized into 4 types. We're not going to explain the differences here, just give you the hint that the "type 4," aka "thin," drivers are the easiest to maintain, since they are pure Java drivers and do not depend on any external DLL's or dynamic libraries. Even though DbVisualizer works with any type of driver, we recommend that you get a type 4 driver if there is one for your database.

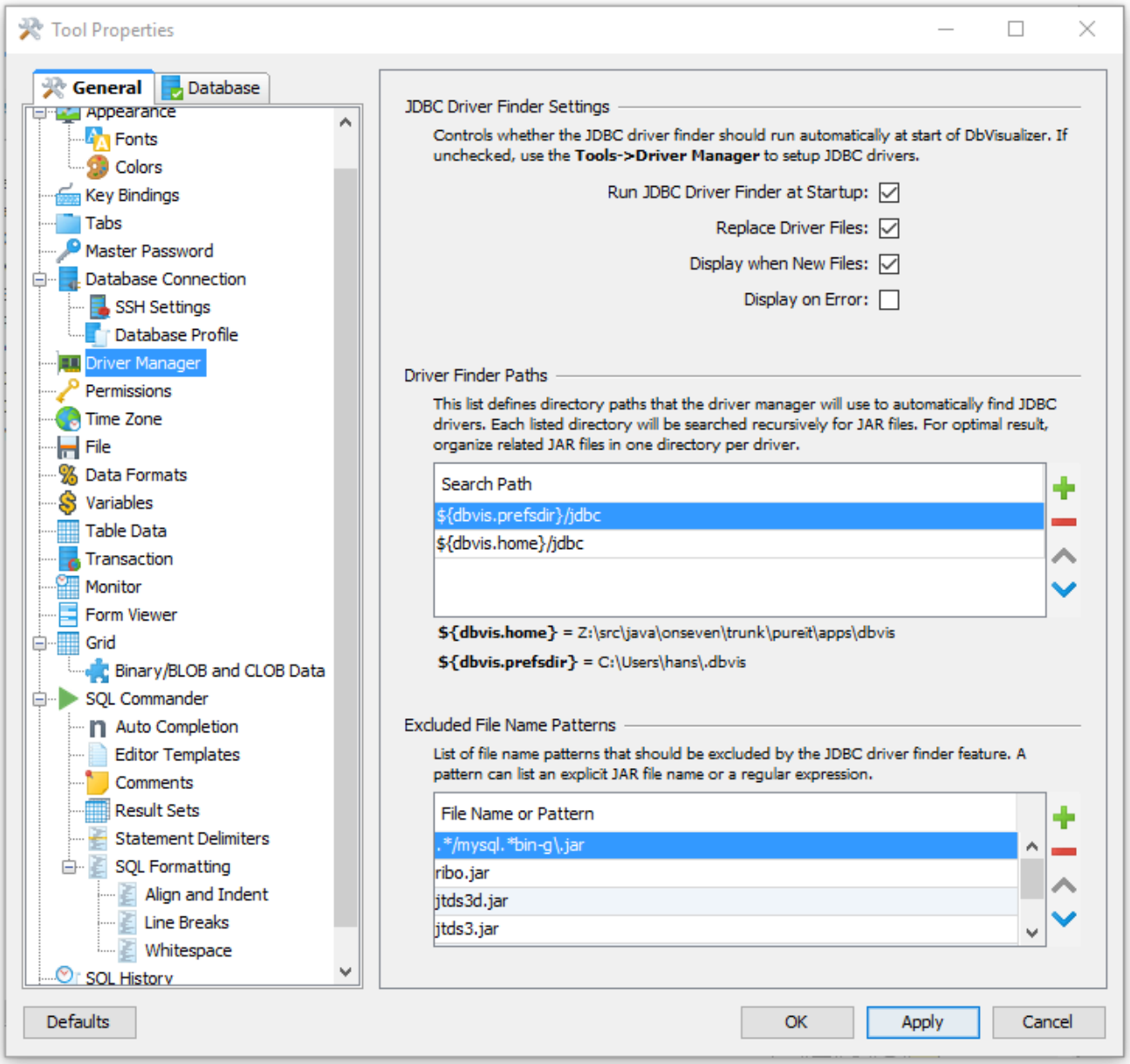
When you have downloaded the JDBC driver into a local folder (and unpacked it, if needed), you can go ahead and create a database connection with the Connection Wizard, as described in the Creating a Connection page. You will then be asked to load the driver files when the wizard needs them. Alternatively, you can move (or copy) the JDBC driver files to the DBVIS_HOME/jdbc folder, where they will be picked up and loaded automatically by the JDBC Driver Finder the next time you start DbVisualizer.

Driver Manager

The **Driver Manager** in DbVisualizer is used to define the drivers that will be used to communicate with the databases. You can manually locate the JDBC driver files and configure the driver, or you can use the JDBC Driver Finder to do most of the work for you, either on demand or automatically.

JDBC Driver Finder

The **JDBC Driver Finder** is a very powerful part of the Driver Manager that automates most of the driver management work. Given the folders where JDBC drivers are located, it loads and configures new drivers (if any) every time you start DbVisualizer. You can configure the JDBC Driver Finder in **Tools ->Tools Properties**, in the **Driver Manager** category under the **General** tab.



Use the following properties to specify the finder behavior:

Property	Description
Run JDBC Driver Finder at Startup	If enabled, the finder will run automatically every time you start DbVisualizer. If it finds any new driver files, it will automatically load and configure them.
Replace Driver Files	If enabled, the driver files are replaced for the matching driver even if the driver already has proper driver files.
Display When New Files	If enabled, the finder window pops-up if it finds any new files when you start DbVisualizer. Otherwise the finder runs invisibly in the background.

Display on Error	If enabled, the finder window pops up if it encounters any errors loading and configuring new drivers. Otherwise it is silent about errors and you have to launch the **Tools->Driver Manager** to see which drivers are not loaded successfully. Enabling this property is only meaningful if you have disabled **Display When New Files**.

You can also specify the folders the JDBC Driver Finder will search. By default, it will search folders named jdbc in the DbVisualizer installation directory (`${dbvis.home}`) and the DbVisualizer preferences folder (`${dbvis.prefsdir}`). These folder paths are shown under the list of **Driver Finder Paths**.

Finally, you can specify regular expression patterns for filenames that the finder should ignore. This can be useful if you need to store other files besides driver files in the designated folders.

If you let the JDBC Driver Finder load all drivers for you, all you need to do to install a new driver is to put the driver files in one of the folders specified for the finder in Tool Properties and then restart DbVisualizer.

The Driver Finder is always activated when upgrading from an older DbVisualizer version.

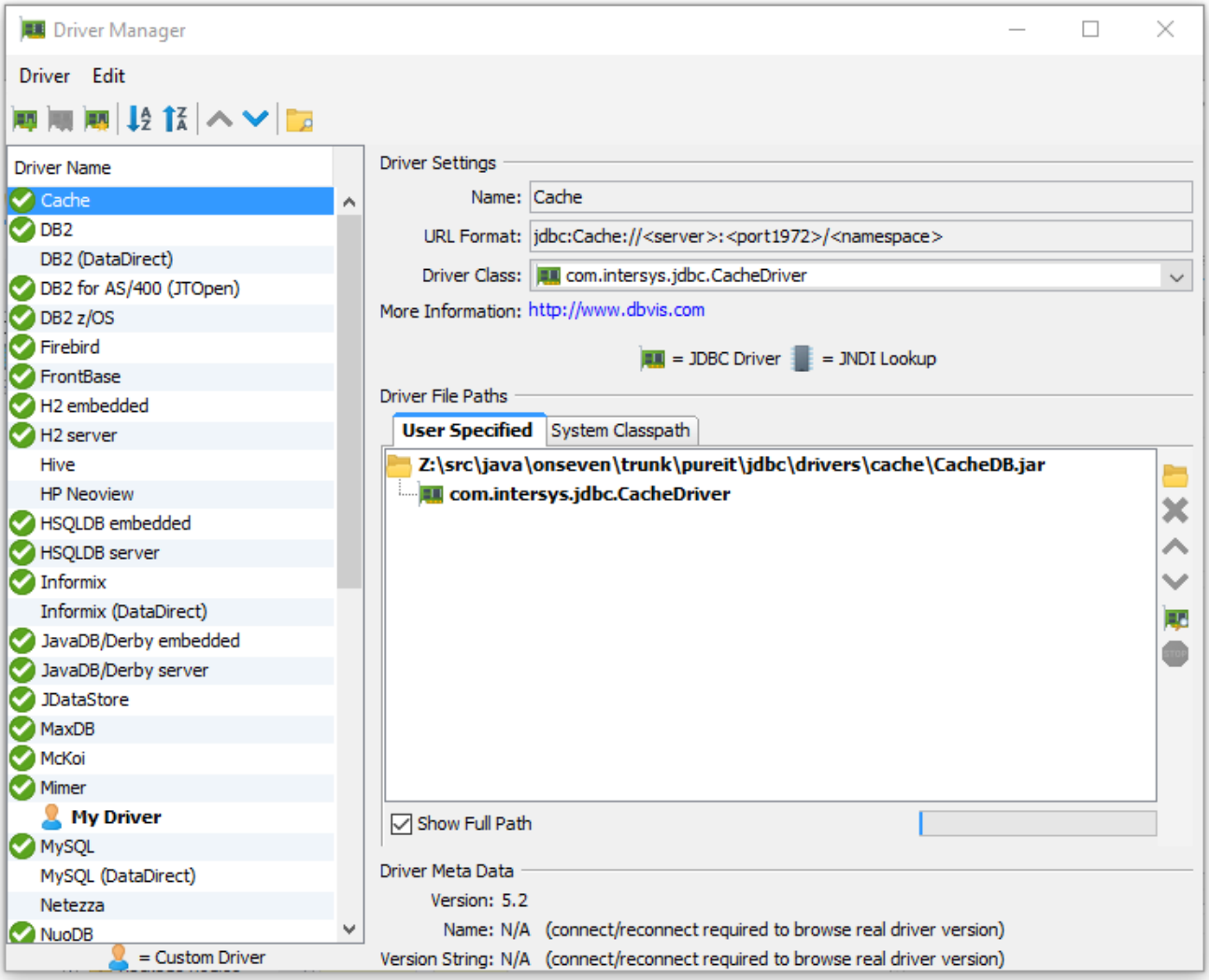
Loading and Configuring Drivers Manually

You can also load and configure JDBC drivers manually using the Driver Manager. If you use JNDI to provide access to the database, you must use this option, since the JDBC Driver Finder does not handle JNDI. Start the Driver Manager dialog using the **Tools->Driver Manager** menu choice.

The left part of the driver manager dialog contains a list of driver names with a symbol indicating whether the driver has been configured or not. The right part displays the driver configuration for the selected driver in terms of the following:

- **Name**
  A driver name in the scope of DbVisualizer is a logical name for either a JDBC driver or an Initial Context in JNDI. This is the name shown in the Connectiontab when selecting which driver to use for a Database Connection
- **URL Format**
  The URL format specifies the pattern for the JDBC URL or a JNDI Lookup name. Its purpose is to assist the user in the Connection tab when entering URL information or a lookup name. See Using Variables in Connection Fields for more about how you can make it really easy to create Database Connections for this driver later on.
- **Driver Class**
  Defines the main class for the JDBC driver, used for connecting to the database.
- **Driver Version**
  Shows the version for a loaded driver.
- **Web Site**
  Link to the DbVisualizer web site, where you can get up-to-date information about how to download the drivers for many databases.
- **Driver File Paths**
  Defines all paths to search for JDBC drivers or Initial Contexts when connecting to the database. The Driver File Paths area is composed of two tabs: the paths in the **User Specified** tab are used for dynamically loaded JDBC drivers or Initial Context classes, and the **System Classpath** tab lists all paths that are part of the Java system class path.

The System Classpath tab is only of interest for the JDBC-ODBC driver.



Initially, the driver list contains a collection of default drivers. They are not fully configured, as the paths to search for the classes need to be identified. You can edit the list, i.e., create, copy, remove and rename drivers. A driver is ready to use once a driver class has been identified, which is indicated with a green check icon in the list. Drivers that are not ready for use are shown without an icon, or with a red cross icon if an error has been detected (such as a missing file) .

Setup a JDBC driver

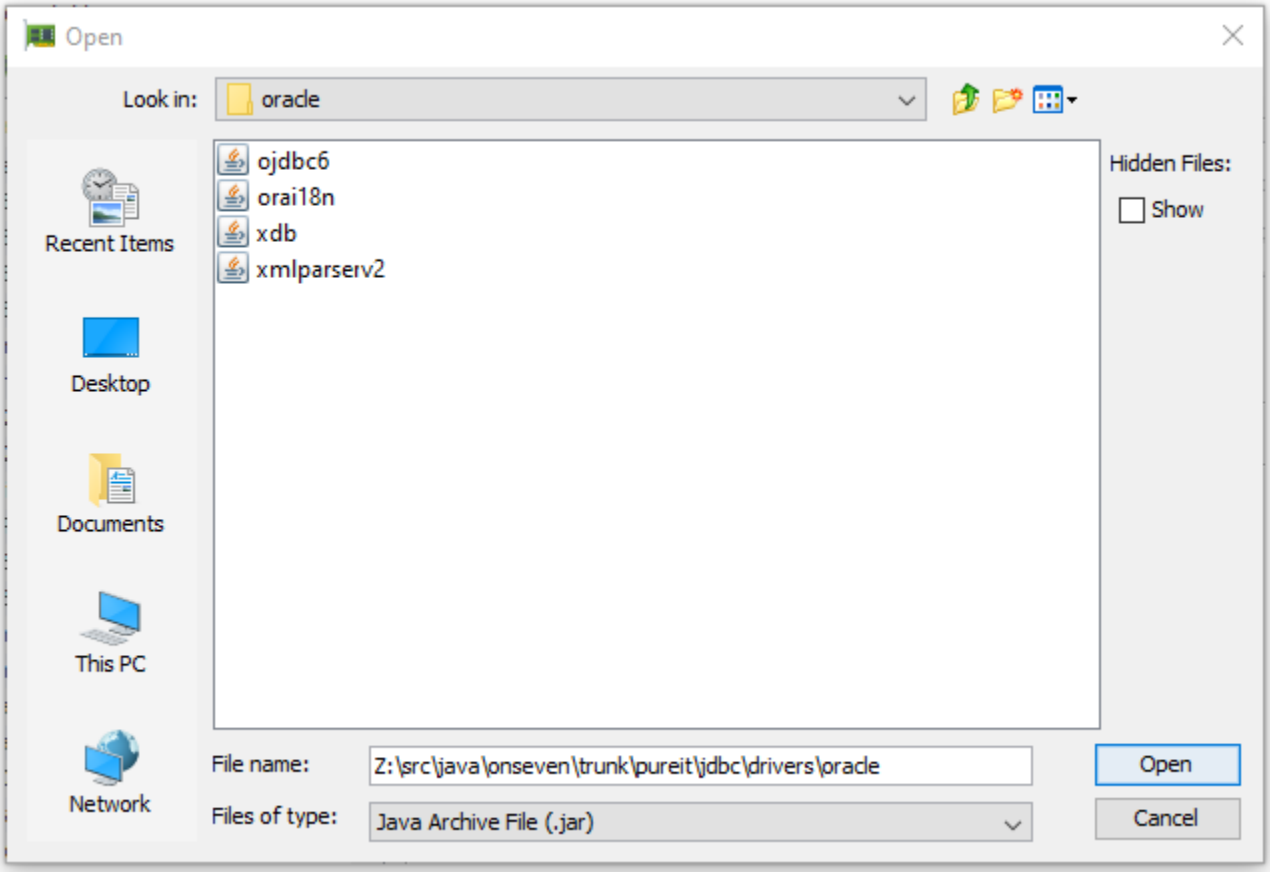
The recommended way to setup a driver is to pick a matching driver name from the list and then simply load the JAR, ZIP or directory that keeps the driver class(es). For instances, if you are going to load the JDBC driver for **Oracle**, select the Oracle driver in the list . You can also create a new driver or copy an existing one.

Check the following online web page with the most current information about the tested databases and drivers:

http://www.dbvis.com/doc/database-drivers/

- It lists which databases and drivers we have tested
- Download links to JDBC drivers
- Information about which files to load in the driver manager for each JDBC driver
- Information about which Driver Class to choose

When you have selected the driver to configure, you need to load the driver files. Click the **Load** button to the right of the **User Specified** paths tree to show the file chooser and load the driver JAR, ZIP or individual files.



A JDBC Driver implementation typically consists of several Java classes. If they are packaged in a JAR or a ZIP file, you don't have to worry about the details; just select and load the JAR or ZIP file. For instance, in the example above, use the ojdbc6.jar file.

If the driver classes are not packaged, it is important to select and load the root folder for the JDBC Driver. Java classes are typically organized using a package name structure. Example:

```
oracle.jdbc.driver.OracleDriver
```

Each package part in the name above (separated by ".") is represented by a folder in the file system. The root folder for the driver is the folder named by the first part, i.e., the *oracle* directory in this example. The class files are stored in the *oracle/jdbc/driver* sub folder. When the driver classes are located in a folder structure like this, you must select and load the root folder, so that the Driver Manager gets the complete package structure.

When a connection is established in the Connection tab, DbVisualizer searches the selected drivers path tree's in the following order:

1. User Specified
2. System Classpath

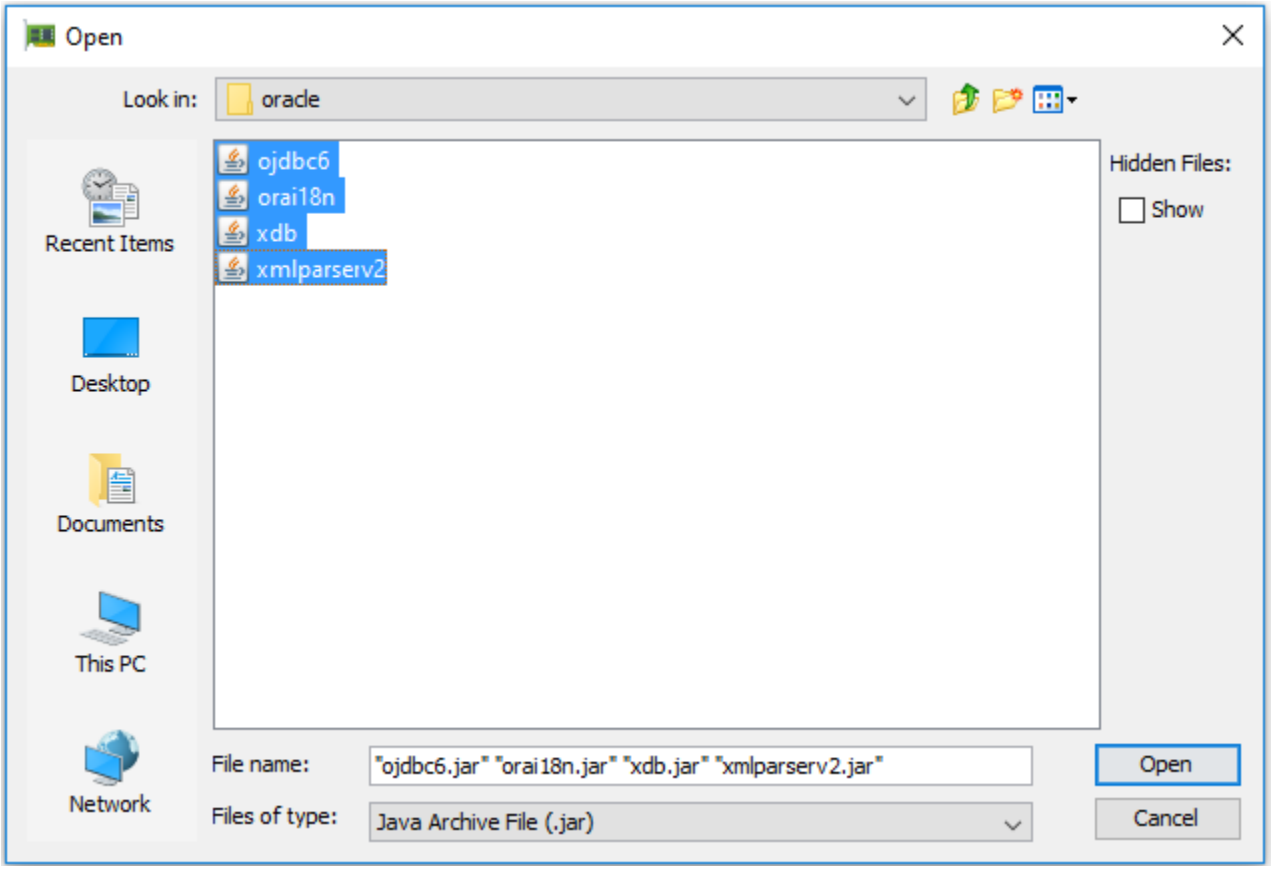
The paths are searched from the top of the tree, i.e., if there are several identical classes in, for example, the dynamic tree, the topmost class will be used. Loading several paths containing different versions of the same driver in one driver definition is not recommended, even though it works (if you do this, you must move the driver you are going to use to the top of the tree). The preferred method for handling multiple versions of a driver is to create several driver definitions.

When you load files in the User Specified paths list, DbVisualizer analyzes each file to find the classes that represent main driver classes. Each such class is listed under the path where it was found in the User Specified paths lists, and it is also added to the **Driver Class** list in the Driver Settings area above. If there is more than one class in the list, make sure you select the correct Driver Class from the list. Consult the driver documentation (or the Databases and JDBC Drivers page) for information about which class to select.

JDBC drivers that requires several JAR or ZIP files

Some drivers depend on several ZIP or JAR files, or directories. One example is if you want XML support for an Oracle database. In addition to the standard JAR file for the driver, you then also need to load two additional JAR files. These are not JDBC driver files but adds functionality the driver needs to fully support XML.

Simply select all JARs at once and press **Open** in the file chooser dialog. The Driver Manager will then automatically analyze each of the loaded files and present any JDBC driver classes or JNDI initial context classes it finds.



The JDBC-ODBC bridge

The JDBC-ODBC driver used to be bundled with most Java installations, but starting with Java 8 it is no longer included. For older Java versions, the JdbcOdbcDriver class is included in a JAR file that is commonly named *rt.jar*, stored somewhere in the Java directory structure. DbVisualizer automatically identifies this JAR file in the System Classpath tree when an older Java version is used. To locate the JdbcOdbcDriver, simply press the **Find Drivers** button to the right of the **System Classpath** tree. When it is found, make sure the `sun.jdbc.odbc.JdbcOdbcDriver` is selected as the Driver Class in the Driver Settings area.

The JDBC-ODBC bridge driver is not intended for production use and is known to be limited and unreliable. Use it only if there is no pure JDBC driver for your database. Please see JDBC-ODBC Bridge Driver Alternatives for more information about alternatives.

Errors (why are some paths red?)

A path in red color indicates that the path is invalid. This may happen if the path has been removed or moved after it was loaded into the driver manager. Simply remove the erroneous path and locate the correct one.

Several versions of the same driver

The Driver Manager supports loading and using several versions of the same driver concurrently. We recommend that you create a unique driver definition per version of the driver and name the driver definitions properly, e.g., **Oracle 9.2.0.1**, **Oracle 10.2.1.0.1**, etc.