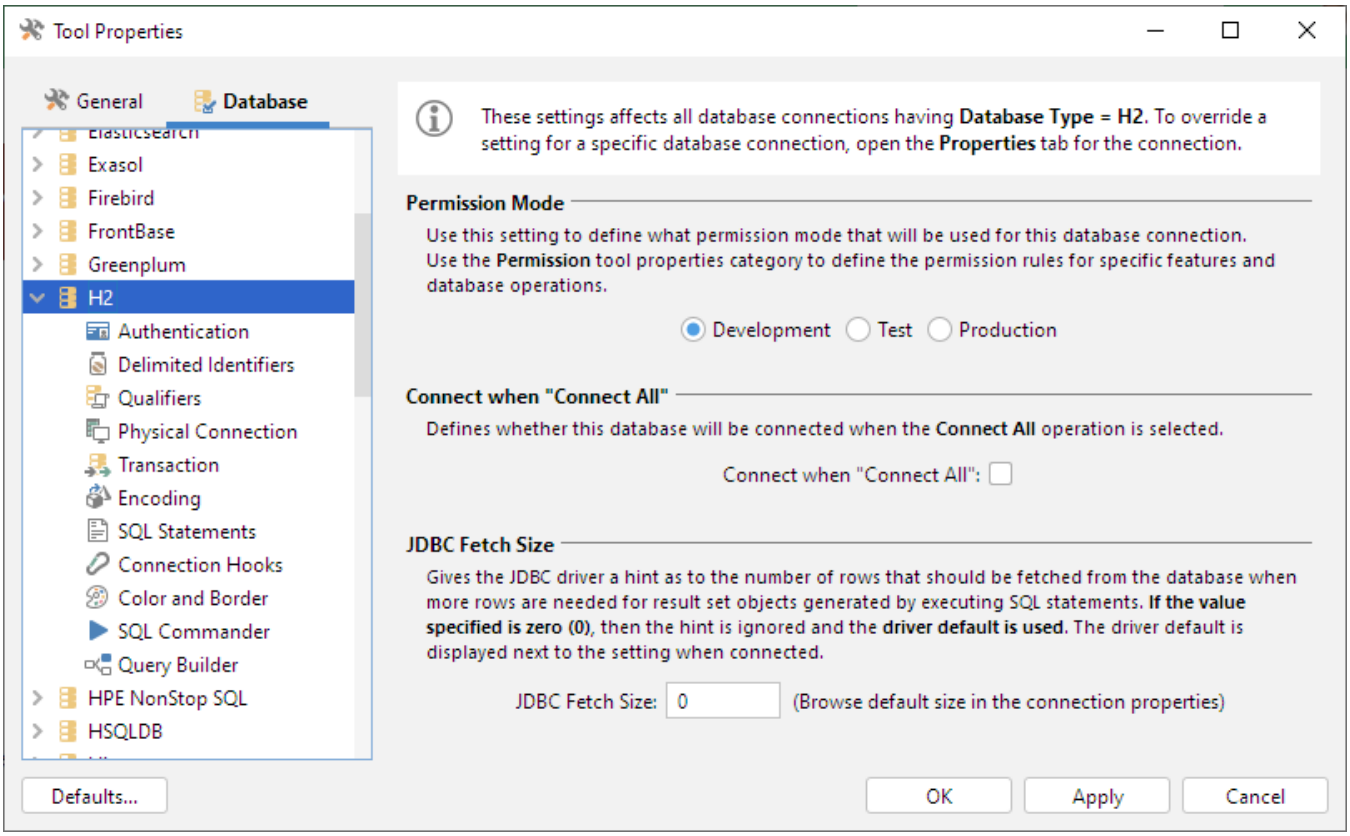
Configuring Connection Properties

In addition to the basic connection information in the **Connection** tab, there is also a collection of connection properties. Which properties are available depends on the **Database Type** selected for the database connection in the Connection tab. Some database types have more properties than others. Which edition of DbVisualizer you use also affects which connection properties are available.

Properties for a connection can be defined at two different levels: **Tool Properties (Database tab)** and **Connection Properties**.

Tool Properties

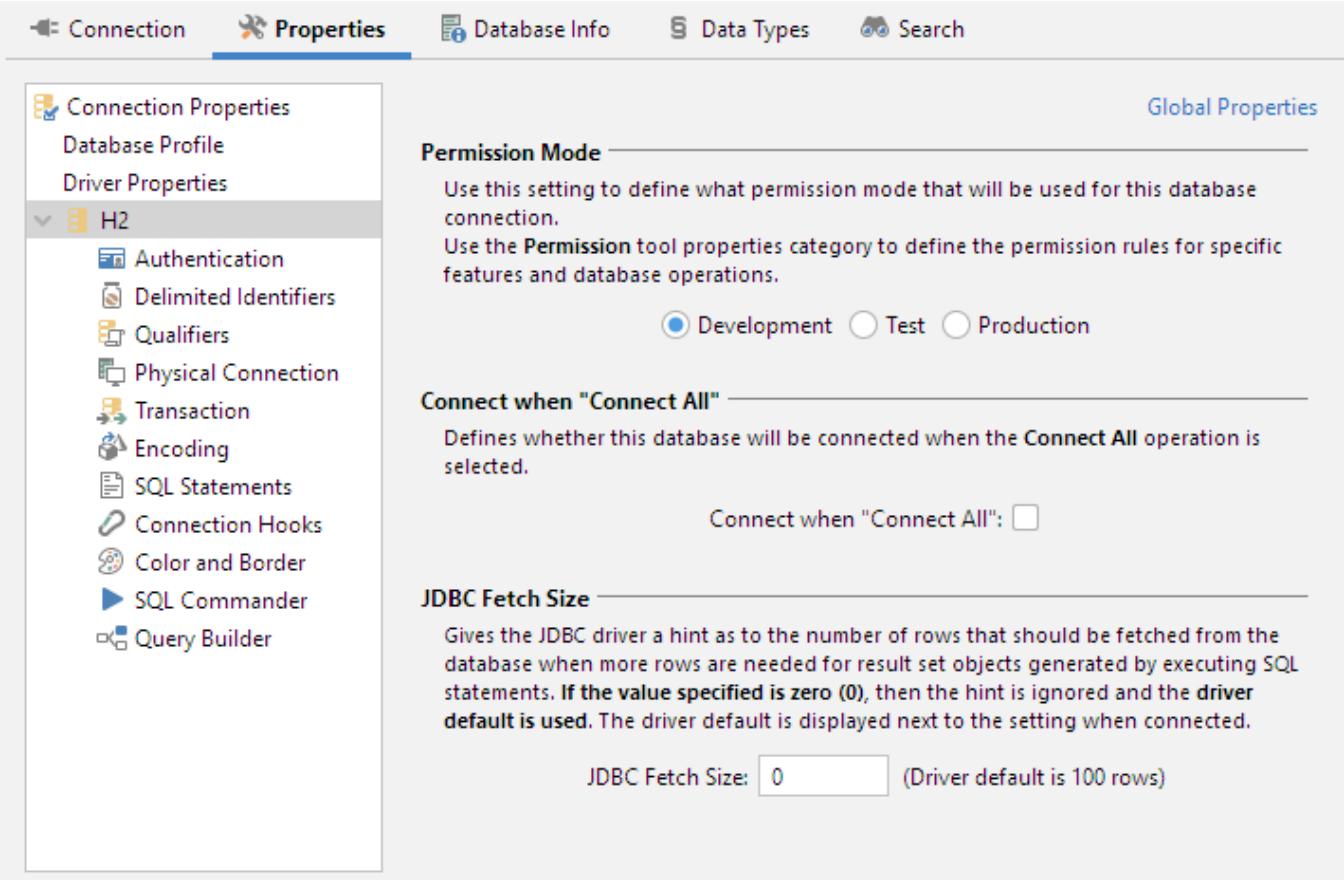
The Database tab in **Tool Properties** defines settings for all connections of the specific database type. All supported database types (Oracle, Informix, SQL Server, Db2, MySQL, etc.) are listed, and for each database type there are a number of properties that are applied to any database connection of that type. This means, for instance, that a database connection defined as being a PostgreSQL database type will use the PostgreSQL properties defined in **Tool Properties**.



Connection Properties

The global properties can be overridden for individual connections. The advantage with this inheritance model is that property changes that apply to all connections can be made in one place, instead of having to apply a common setting for every database connection of a specific database type.

The Connection Properties are available in the **Properties** sub tab in the the database connection's **Object View** tab.



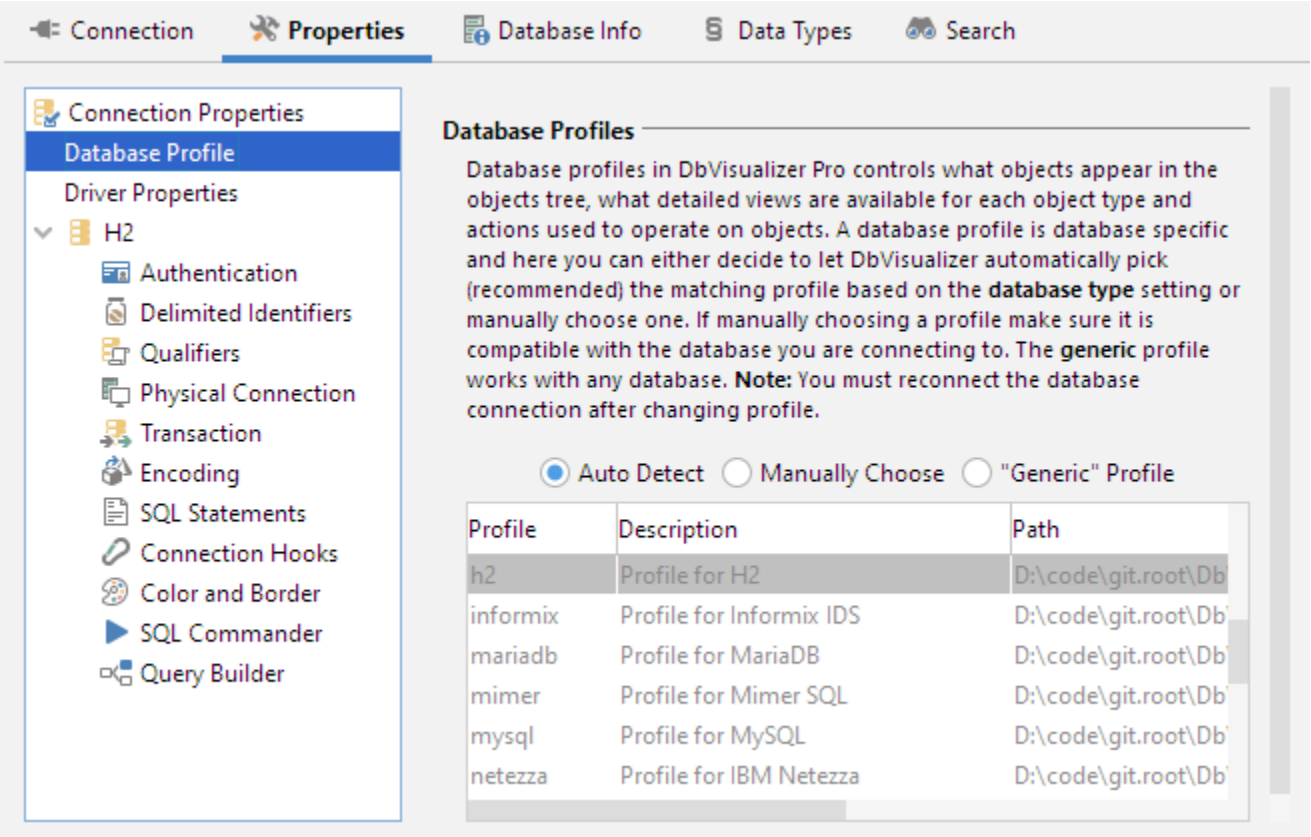
The **Properties** tab is organized basically the same way as the **Tool Properties** window. The main difference is that the list contains only the categories that are applicable to this database connection. The **Database Profile** and **Driver Properties** categories are available only in the **Properties** tab and not in **Tool Properties**. The page explains the **Database Profile** and **Driver Properties** categories, while the other categories are described in pages that describe feature the property applies to.

Additional categories may appear in the connection properties depending on the type of database. An example is the category for Explain Plan for the databases where this feature is supported.

At the top right corner, you will also find a link to set the **Global Properties** for all connections of this database type.

Database Profile

The Database Profile category is used to select whether a profile should be automatically detected and loaded by DbVisualizer, or if a specific one should be used for the database connection. The default strategy is to **Auto Detect** a database profile.

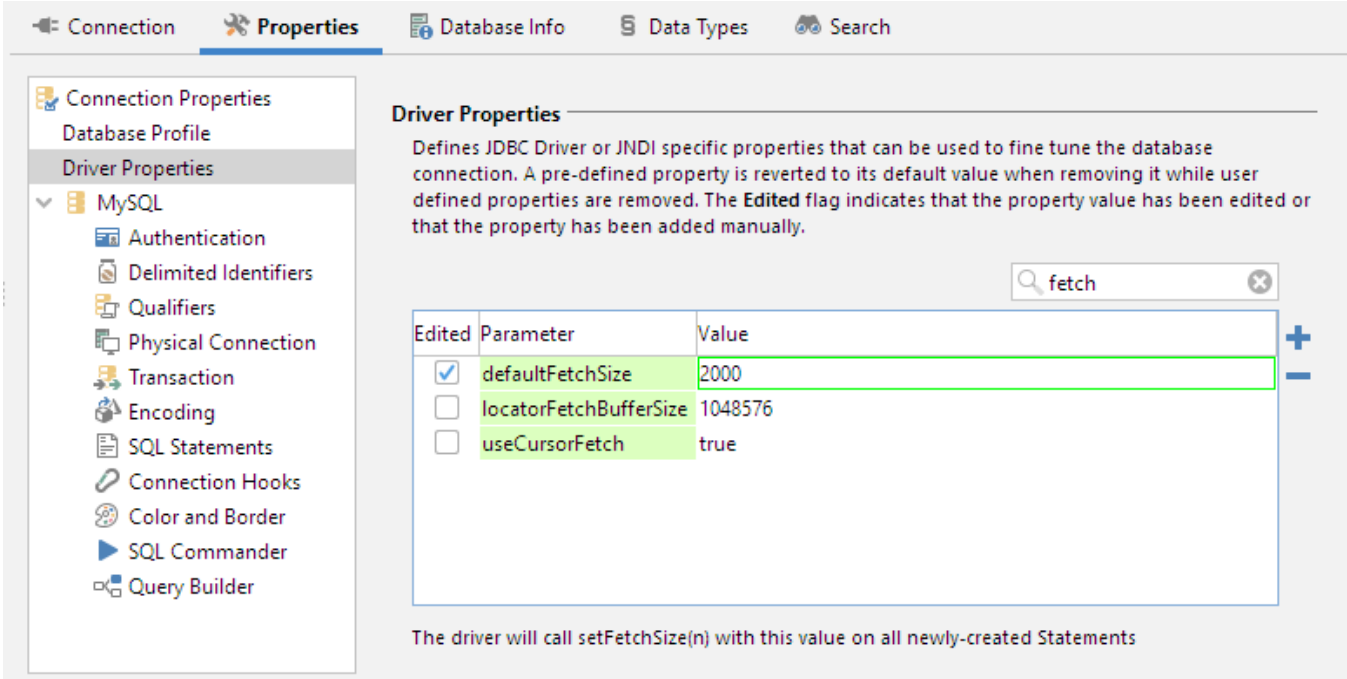


The way DbVisualizer auto detects a profile is based on the setting of **Database Type** in the **Connection** tab. If the **Database Type** is also set to **Auto Detect**, DbVisualizer first detects the database type based on the JDBC information, and then detects the profile based on the database type.

There is rarely a reason to use another setting than **Auto Detect**, but if you manually choose a database profile, this choice will be saved between invocations of DbVisualizer.

Driver Properties

The **Driver Properties** category is used to fine tune a JDBC driver before the database connection is established. You can use the filter to quickly find a specific property.



The list of parameters, their default values and parameter descriptions are determined by the JDBC driver used for the connection. Not all drivers supports additional driver properties. To change a value, just modify it in the list. The first column in the list indicates whether the property has been modified or not, and so, whether DbVisualizer will pass that parameter and value onto the driver at connect time.

New parameters can be added using the buttons to the right of the list.

Invoke Java methods in the JDBC driver

In addition to driver properties it is also possible to invoke low level Java methods in the JDBC driver classes, java.sql.Connection and java.sql.Statement. These are edited in the driver properties list.

You may run single argument methods taking one of **String.class**, **Integer.class**, or **Boolean.class** Java types as argument. The method name should be specified as **Parameter** which must be fully qualified with the all lowercase class name. The argument is specified in the **Value** field in the driver properties list.

Here are a few examples:

Parameter	Value
`java.sql.connection.setReadOnly`	`true`
`java.sql.statement.setFetchSize`	`1000`

java.sql.connection methods are invoked just after a physical connection with the database has been established. **java.sql.statement** methods are invoked just after a statement has been created and ready for execution.