

---

# DbVisualizer 4.2

Users Guide

February 2005



---

<http://www.dbvis.com>

# Table of Contents

<b>Getting Started and General Overview.....</b>	<b>7</b>
Installing.....	7
Installation structure.....	7
Java Properties.....	7
How to install license for DbVisualizer Personal.....	8
Resources.....	8
Starting DbVisualizer.....	9
Command line arguments.....	9
Application overview.....	10
Worth knowing about the GUI.....	11
Grid, Graph and Chart.....	11
Context sensitive controls.....	12
Tool tips.....	12
Grids.....	12
Right click menu.....	13
Calculate Selection.....	14
Column Visibility.....	14
Problem resolution.....	15
How to satisfy the DbVisualizer support team.....	16
<b>Load JDBC Driver and Get Connected.....</b>	<b>17</b>
What is a JDBC Driver?.....	17
Get the JDBC driver file(s).....	18
Driver Manager.....	18
Setup a JDBC driver.....	19
JDBC drivers that requires several JAR or ZIP files.....	21
The JDBC-ODBC bridge.....	21
Loading JNDI Initial Contexts.....	21
Errors (why are some paths red?).....	22
Several versions of the same driver.....	23
Setup a database connection.....	23
Setup using JDBC driver.....	23
Setup using JNDI lookup.....	24
Connection Properties.....	25
Database Profile.....	27
Driver Properties.....	28
Driver Properties for JDBC Driver.....	28
Driver Properties for JNDI Lookup.....	28
Always ask for userid and/or password.....	29
Using variables in the Connection details.....	30
Connect to the Database.....	31
Connections Overview.....	33
<b>Database Objects Explorer.....</b>	<b>35</b>
Introduction.....	35
Database Profiles.....	36
Objects tree for Oracle, DB2, SQL Server, Informix, Sybase ASE, PostgreSQL and MySQL... 36	
Database Objects Tree.....	38
Tab tool bar operations.....	38
Right click menu operations.....	38
Filtering.....	39
Show Table Row Count.....	40
Standard Tree Objects.....	40
Connections object.....	40

Database Connection object.....	41
Connection Alias.....	42
Default databases and schemas.....	42
Remove and copy database connection objects.....	43
Database Connection details tabs.....	43
Search.....	43
Folder object.....	44
Object View Types.....	45
Grid.....	45
Form.....	46
Source.....	47
Table Row Count.....	48
Table Data.....	49
Right click menu.....	50
Where Filter.....	52
Quick Filter.....	52
Monitor row count.....	53
Editing.....	53
References.....	54
Generic Database Profile.....	55
Catalog (aka Database) object.....	55
Schema object.....	56
Table type object.....	57
Table object.....	58
Data tab.....	59
References tab.....	59
Procedure object.....	59
Specialized Database Profiles.....	60
<b>Executing SQL statements in the SQL Commander.....</b>	<b>62</b>
Introduction.....	62
Editor.....	62
Database Connection, Catalog and Schema.....	63
Fonts.....	64
Editor shortcuts.....	64
Load from and save to file.....	65
Comments in the SQL.....	66
Multiple editors.....	66
Tabs style.....	66
Windows style.....	67
Auto Completion.....	68
History.....	70
SQL Bookmarks.....	70
Execution.....	70
Execution control.....	71
Execute statement at cursor position.....	71
Selection executes.....	71
Commit and Rollback.....	71
SQL Scripts.....	72
Anonymous SQL blocks.....	73
Stored Procedures.....	73
Control commands.....	73
Variables.....	73
Variable Syntax.....	74
Output View.....	75
Output View menu.....	76
Result set grids.....	76

Editing.....	77
Multiple result sets produced by a single SQL statement.....	78
Text.....	79
Chart.....	80
Log.....	80
Log controls.....	81
Auto clear log.....	81
<b>Monitor and Charts.....</b>	<b>82</b>
Introduction.....	82
Monitor an SQL statement .....	82
Monitor table row count.....	84
Monitor table row count difference.....	85
Monitor window.....	86
Charts.....	88
Chart Controls.....	88
Data.....	88
Layout.....	89
Chart View.....	91
Zooming.....	91
Rotating.....	91
Export.....	92
<b>Edit Table Data.....</b>	<b>93</b>
Introduction.....	93
Features that support editing.....	93
Edits might be denied.....	93
Commit.....	94
Error Log.....	94
Binary data/BLOB and CLOB.....	95
Inline Editor.....	95
Insert a new row.....	96
Update an existing row.....	97
Delete a row.....	97
Cell pop up menu.....	98
Form Editor.....	98
Form editor controls.....	98
Row Values.....	99
Insert a row.....	100
Edit a row (update, delete or insert copy).....	101
Update the row.....	101
Delete the row.....	102
Insert a copy of a row.....	102
Import from File.....	102
Export from File.....	102
Editing Binary data/BLOB and CLOB.....	102
CLOB.....	103
Binary data/BLOB.....	103
GIF, JPEG and PNG viewer.....	103
Serialized Java objects viewer.....	104
Hex/Ascii viewer.....	104
<b>Create Table and Index Assistants.....</b>	<b>105</b>
Introduction.....	105
Create Table.....	105
Columns.....	106
SQL Preview.....	107
Execute.....	108
Create Index.....	108

<b>SQL Bookmarks.....</b>	<b>110</b>
Introduction.....	110
What's a bookmark in DbVisualizer?.....	110
The Bookmarks Main Menu.....	110
Bookmark Editor .....	111
Bookmark list.....	111
New and History root folders.....	114
SQL Editor.....	114
Monitor information.....	114
The Note field.....	114
Executing an SQL bookmark or folder of SQL bookmarks.....	114
<b>Tool Properties.....</b>	<b>117</b>
Customizing DbVisualizer.....	117
The user preferences (XML) file.....	118
Categories.....	118
Appearance.....	118
Database Connection.....	120
Data Formats.....	120
Date, Time and Timestamp formats.....	120
Table Data.....	121
Data Editors.....	121
SQL.....	122
SQL Statements.....	122
Statement Delimiters.....	123
Pre-processing.....	124
Comments.....	124
New Lines.....	124
Transaction.....	125
Copy.....	125
Bookmarks.....	125
Monitor.....	125
Grid.....	125
Colors.....	126
Binary/BLOB and CLOB Data.....	126
Editor.....	126
Key Bindings.....	126
Auto Completion.....	126
Fonts.....	127
Debug.....	127
<b>Export and Print.....</b>	<b>128</b>
Introduction.....	128
Export.....	128
How to launch.....	128
Grid data.....	128
Destination.....	130
Format.....	130
Columns .....	131
Options.....	131
Format.....	131
Text.....	132
Delimiter.....	132
Extra.....	133
Description.....	133
Generate Test Data.....	133
Test data generator sample.....	134
Preview.....	136

Text data.....	136
Graph data.....	136
Chart data.....	137
Print.....	138
Grid.....	138
Graph.....	138
Print Preview.....	139

# Getting Started and General Overview

This document is worth reading for all new users as it explains the installation process, resources and how the application is organized.

**Note: All documents in the Users Guide are primarily focusing on the DbVisualizer Personal edition.**

The screen shots throughout the users guide are using the [Alloy](#) look and feel.

## Installing

Installing DbVisualizer is a straight forward task. The installation procedure is performed using a graphical application and it is just a matter of answering the questions that are displayed. Follow the [instructions](#) at the DbVisualizer web site how to start the installation specifically for your platform.

## Installation structure

The installer and launcher for DbVisualizer is based on the "install4j" software (<http://www.install4j.com>). The structure of the installation directory contains the following. (The content may differ slightly between platforms):

```
.install4j/  
doc/  
lib/  
resources/  
wrapper/  
dbvis.exe  
README.txt  
uninstall.exe
```

There is basically nothing in this directory that is of general interest except the `dbvis.exe` file which is used to start DbVisualizer.

## Java Properties

DbVisualizer relies on few Java properties that can be used to modify some characteristics of the application. These DbVisualizer specific properties are available in the `resources/dbvis-custom.prefs` file.

**Note:** There's normally no reason to alter these properties as their default values are sufficient for most use.

Properties are expressed as:

```
property=value
```

DbVisualizer reads the following properties:

Property	Description
----------	-------------

<code>dbvis.driver.ignore.dir=</code>	Specify directories from DBVIS-HOME that should not be listed in the Driver Manager "System Classpath" list. Directories are separated with ":" Accepted values: one or several directory names starting from DBVIS-HOME.
<code>dbvis.grid.encode=true</code>	Specifies if encoding of data in result set grids will be performed or not. If set to true then make sure the <code>dbvis.grid.fromEncode</code> and/or <code>dbvis.grid.toEncode</code> is set too.
<code>dbvis.grid.fromEncode=ISO8859_1</code>	Encoding used when translating text data that is fetched from the database
<code>dbvis.grid.toEncode=GBK</code>	Encoding used when translating data that will appear in the result set grid
<code>dbvis.formeditor.unlimitedfields=true</code>	Specify whether the Form Editor should ignore the max column length and allow any number of characters to be entered
<code>dbvis.usegetObject=true</code>	Specifies if the generic <code>ResultSet.getObject()</code> method in JDBC will be used in favor of the data type specific get methods or not. Default is false.
<code>dbvis.savedatacolumns=false</code>	Column layout changes such as reordering and/or visibility is saved for all grids in the Objects Views *except* for the "Data" grid. This property can be used to also include the layout in the "Data" grid. Note: This will result in DbVis saving the layout for each table that is displayed in the Data grid = huge XML file...
<code>dbvis.disabledataedit=false</code>	Specifies if Data editing should be completely disabled. Note: This have only effect when used with a licensed edition.

(DBVIS-HOME is the installation directory for DbVisualizer).

**Note:** All `dbvis.` properties may change in future versions of DbVisualizer. Some are also experimental and may be removed or instead introduced in the DbVisualizer GUI.

## How to install license for DbVisualizer Personal

If you have a license key file for DbVisualizer Personal then start DbVisualizer and open the **Help->License** window. Enter the name of the license file in the **License Key File** field or launch the file chooser by pressing the "..." button to the right of the license file field.

## Resources

Resources related to DbVisualizer that are good to know:

Resources
The home of <a href="#">DbVisualizer</a>
The <a href="#">FAQ</a> which is regularly updated with frequently asked questions and known problems
The <a href="#">User Guide</a>



The Databases and JDBC Drivers online page. This page gives latest information about databases and JDBC drivers

The on line [problem report](#) form. This is the recommended channel for product support.

Support by [email](#). Questions, problems, comments or any feedback related directly to the DbVisualizer product. Please use the support form web page in preference to sending emails to this email address directly.

## Starting DbVisualizer

The way to start DbVisualizer depends on what platform you are using.

- **Windows**

Locate the DbVisualizer sub menu in the **Start** menu. Select the **DbVisualizer** operation in that menu

- **Linux/Unix**

Open a shell and change directory to the DbVisualizer installation directory. Execute the **dbvis** program

- **Mac OS X**

Double click on DbVisualizer application icon.

## Command line arguments

DbVisualizer supports a range of command line arguments. These are listed in the **Help->Usage Information** menu choice in DbVisualizer.

```
Usage: dbvis [-help] [-up <path>] [-sqlfile <path>]
           [-windowtitle <title>]
           [connect options] [remote options]
```

### General Options:

```
-help           Display this help
-up <path>      Use an alternate user preferences file
-sqlfile <path> Load file into the SQL Commander editor
-windowtitle <title> Additional window title
```

### Driver Connect Options:

```
-driver         Setup and connect using Driver:
  -alias <name> Database alias
  -drivername <name> Driver name
  -path <path> Path to driver class
  -class <class> JDBC Driver class
  -url <url> Connection URL
  -userid <user> Userid to connect as
  -password <pw> Connect password
```

### JNDI Connect Options:

```
-jndi          Setup and connect using JNDI:
  -alias <name> Database alias
  -drivername <name> Driver name
  -path <path> Path to initial context class
  -class <class> Initial context class
  -url <url> Provider URL
  -lookup <name> Lookup name
  -userid <user> Userid to connect as
  -password <pw> Connect password
```

### Remote Options:

```
-attachremote  Attach to remote DbVisualizer instance
-enableremote  Enable remote attachment
-host <host>   Remote host name (default: localhost)
-port <port>   Remote port (default: 8787)
```

# Application overview

The main window for DbVisualizer as it looks at startup. The interface is organized around the database objects tree to the left and two main tabs at the right:

- **Database Objects Tree**

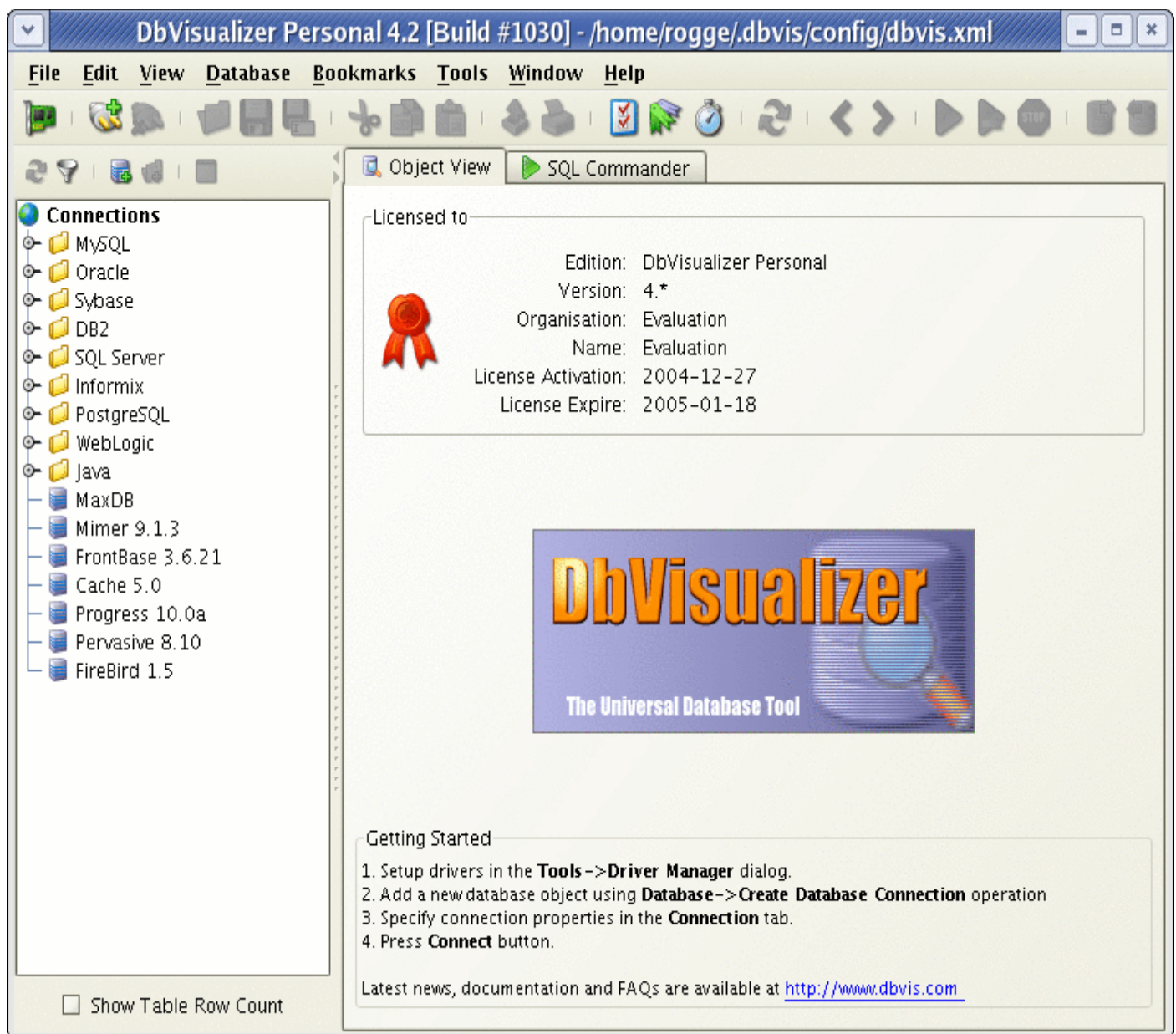
This tree contains at the top level Database Connection objects (or folder objects with their only purpose to group Database Connections). Use this tree to navigate a database. Clicking on an object will change the view in the Object View tab to show details about the selected object.

- **Object View**

This tab shows details about the selected object in the tree. Every object have their own representation in the object view tab.

- **SQL Commander**

Execute arbitrary SQL statements and scripts of statements.



**Figure: The DbVisualizer main window**

The main window is organized around a standard tool layout based on a **menu** and **tool bar** at the top.

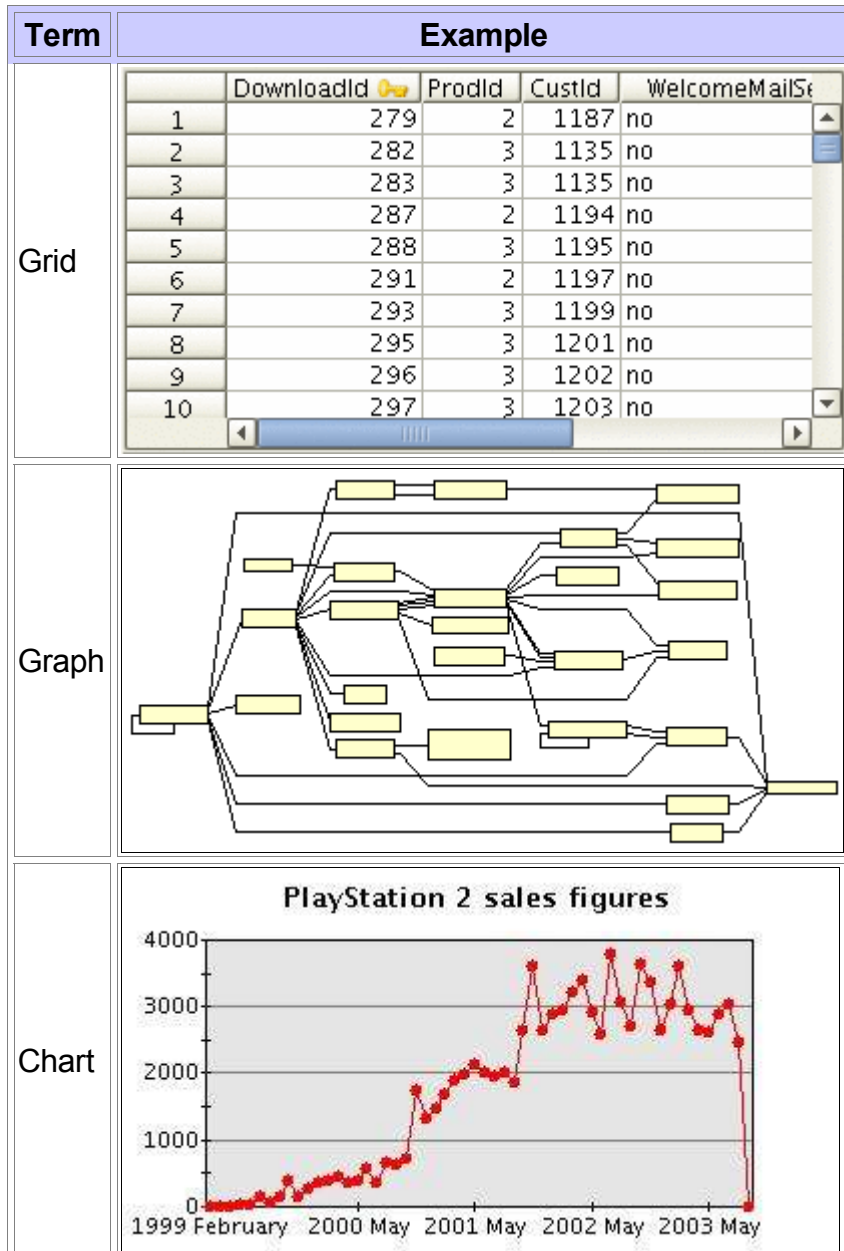
(Visit the master [documentation index](#) for detailed information about each of the features).

## Worth knowing about the GUI

The following section presents some generic topics that are worth knowing about when you using DbVisualizer.

### Grid, Graph and Chart

Grid, graph and chart are three terms that are often used in the application and in the documentation. The following explains what they represent.



**Figure: The grid, graph and chart terms**

**Note:** The reason the documentation refer to "grid" rather than "table" is because otherwise it would in many situations be confused with a database table.

## Context sensitive controls

All controls in DbVisualizer are context sensitive. This simply means that a control is enabled only if it can be used.

## Tool tips

Tooltips are used to explain a graphical control. They are also used to express status information about a control. An example is the grid column header tooltip which lists type data for the column:

Column Name: **Id**  
Display Size: **11**  
Database Column Type: **LONG**  
JDBC Column Type: **INTEGER (code: 4)**  
Java Data Type: **class java.lang.Integer**

Figure: Tooltip example

## Grids

Grids are used heavily in DbVisualizer and probably needs a brief introduction.

The screenshot shows a data grid with the following columns: CustId, CreatedDate, SelfRegistered, and a truncated column. The grid is annotated with several labels and arrows:

- Primary key symbol:** Points to the CustId column header.
- Sort direction indicator:** Points to the sort icons in the CustId header.
- Binary data symbol:** Points to the red star icon in the SelfRegistered column.
- Column Visibility control:** Points to the 'N' icon in the truncated column header.
- Truncated column color:** Points to the red background of the truncated column.
- Null value:** Points to the '(null)' value in the SelfRegistered column for row 6.
- Current row indicator:** Points to the blue arrow icon in the row number column for row 10.

The grid data is as follows:

	CustId	CreatedDate	SelfRegistered	Truncated column
1	1003	1999-02-22	(BINARY - 3 bytes)	kullrolle
2	1004	1999-02-23	(BINARY - 3 bytes)	Nils Pils
3	1005	2002-03-21	(BINARY - 384 bytes)	Ulf Anders
4	1006	1999-03-08	(BINARY - 58,394 bytes)	Matt Perry
5	1007	1999-09-27	(BINARY - 3 bytes)	Rudolf Mul
6	1008	1999-03-09	(null)	Ralph Iden
7	1009	2000-11-01	(BINARY - 3 bytes)	Ima Medres
8	1010	1999-03-09	(BINARY - 3 bytes)	Carl Binde
9	1012	1999-03-10	(BINARY - 3 bytes)	Steve Wils
10	1013	1999-03-11	(BINARY - 3 bytes)	Van Son
11	1014	1999-03-12	(BINARY - 3 bytes)	Roger Wrig
12	1017	1999-03-24	(BINARY - 3 bytes)	Pekka Ucty
13	1018	1999-03-25	(BINARY - 3 bytes)	Mike Swans
14	1019	1999-04-14	(BINARY - 3 bytes)	Larry Crof

At the bottom of the grid, there are status indicators:

- Max Rows: 2500
- Max Chars: 10
- 0.111 sec/0.162 sec
- 2500 / 18
- 1-15

Arrows point from these indicators to labels below the grid:

- Exec/Fetch time
- Rows/Columns (red = truncated)
- Visible row range

Figure: Grid overview

The screen shot shows the grid and controls that are available in the **Database Objects->Data tab** but the differences are minor compared to the standard grid.

## Right click menu

The generic right click menu contains the following operations:

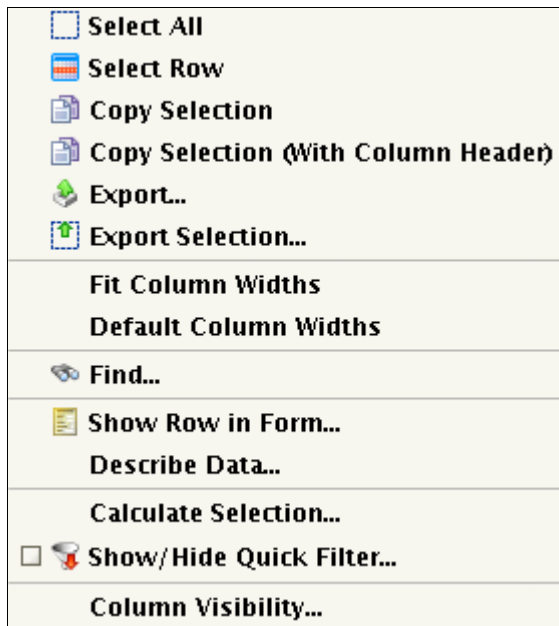


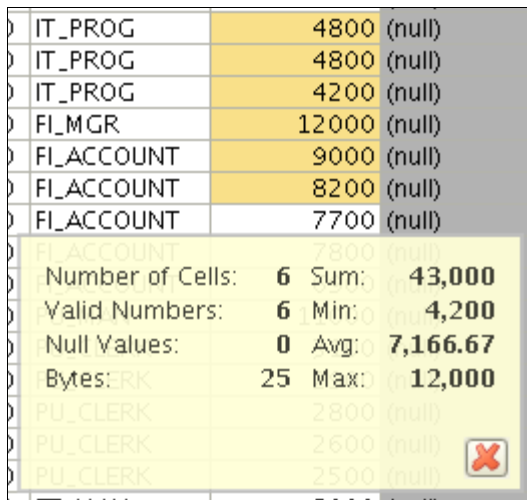
Figure: Grid right click menu

Menu Choice	Description
Select All	Selects all cells (aka rows and columns) in the grid
Select Row	Selects all cells in the row
Copy Selection	Copy all selected cells onto the system clipboard
Copy Selection (With Column Header)	Copy all selected cells including column header onto the system clipboard
Export	Launch the export dialog
Export Selection	Export the selection using the standard export feature
Fit Column Widths	Automatically fit all column widths according to the widest cell value
Default Column Widths	Set the column width equally for all columns
Find	Launch the find dialog
Show Row in Form	Displays all data for the selected row in a form. <b>Note:</b> this is just a read only form as editing is not allowed.
Describe Data	Show detailed information about the columns in the grid
Calculate Selection	Displays some metrics about the current selection. This is especially useful for numeric fields. Read more in <a href="#">Calculate Selection</a> below.
Show/Hide Quick Filter	Displays or hides the quick filter pane. Read more about Quick Filters in the <a href="#">SQL Commander</a> document.
Column Visibility	Displays the column visibility menu. Use this to control what columns should be displayed in the grid. Read more in <a href="#">Column Visibility</a> below.

(The menu in the Data tab adds a few more operations that are used to generate SQL statements based on the current selection).

## Calculate Selection

The **Calculate Selection** feature is a utility that does various calculation on the current selection. It is primarily useful when the selection keeps numbers. The following is an example of what it shows.



The screenshot shows a data grid with columns for a category, a numerical value, and a status. The numerical values are highlighted in yellow. Below the grid, a summary box displays the following statistics:

Number of Cells:	6	Sum:	43,000
Valid Numbers:	6	Min:	4,200
Null Values:	0	Avg:	7,166.67
Bytes:	25	Max:	12,000

A red cross icon is visible in the bottom right corner of the popup.

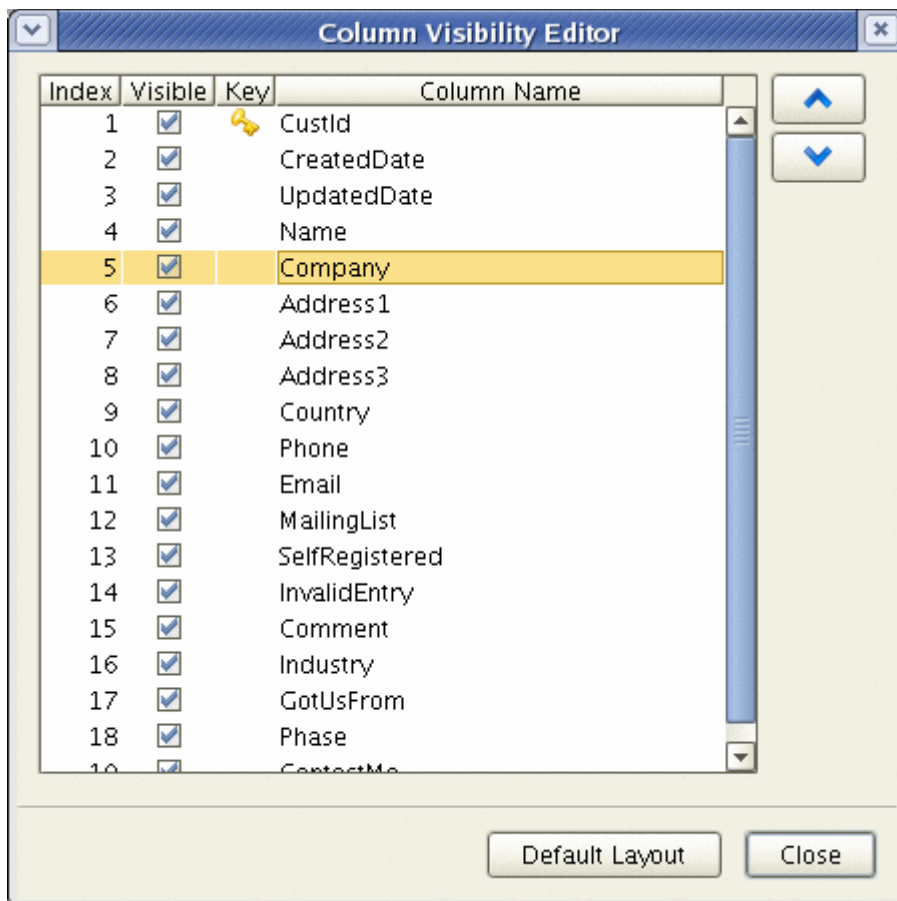
Figure: The calculate selection popup

Property	Description
<b>Number of Cells</b>	shows the number of cells in the selection.
<b>Valid Numbers</b>	lists the number of valid numbers in the selection.
<b>Null Values</b>	shows the number of null values in the selection.
<b>Bytes</b>	shows the total number of bytes in the selection after that the data has been translated to text
<b>Sum</b>	shows the the total summary of the selection
<b>Min</b>	shows the minimum number in the selection
<b>Avg</b>	shows the average value of the selection by doing sum / number of valid numbers
<b>Max</b>	shows the maximum number in the selection

Either click the red cross icon or anywhere in the popup to close it.

## Column Visibility

The Column Visibility feature is used to control what columns that should appear in a grid. The column visibility dialog is displayed either by choosing the **Column Visibility** right menu choice in the grid or by clicking the button above the vertical scrollbar in the grid.



**Figure: The column visibility dialog**

The column visibility dialog shows all columns that are available in the grid. The check mark in front of a column name indicates that the column is visible in the grid while an unchecked box indicates that it is invisible. Columns can be made invisible either by selecting a checked column name in this list or by using the **Remove Column** menu choice in the grid column header menu. The order of the columns can also be adjusted in this dialog. Just select a row and then move it up (left in grid) or down (right in grid).

The **Default Layout** will reset the grid by making all column visible and put them in their default locations.

**Note 1:** Modifying column visibility in conjunction with column resizing and column ordering is saved between invocations of DbVisualizer for all grids in the various **Object Views** except the Data tab.

**Note 2:** If modifying column visibility in the **Data** tab then these changes will persist throughout the session i.e if you for example remove the column **NAME** in the Data tab for the table **EMPLOYEE** then will **NAME** not appear if doing a reload or subsequent shows of the Data tab for that table. You must manually make it visible again or simply select **Default Layout** to bring it back. Another solution is simply to restart the application.

## Problem resolution

There are situations when problems, errors or even bugs appear even though DbVisualizer is extensively tested before every new version. The runtime environment for DbVisualizer is rather complicated when it comes to tracking the source of a potential problem since it's not only DbVisualizer that may cause the problem but also the actual JDBC driver(s).

There are a few things that you can do before reporting problems based on at what stage the problem occurs:

1. Make sure you are using the latest version of [Java 1.4](#)
2. Make sure you are using at least the [version](#) of the JDBC driver that we've tested DbVisualizer with
3. Read the DbVisualizer [FAQ](#).
4. Read the DbVisualizer [Users Guide](#).

If you cannot find a solution to resolve the problem then please do the following and email us the debug output:

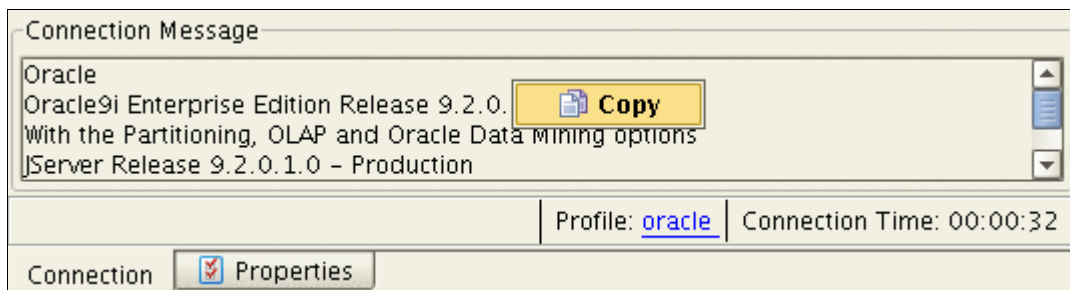
- Problem during installation or when starting DbVisualizer  
[Debugging the installer](#)
- Error or problem while using DbVisualizer  
[Debugging DbVisualizer](#)

Use the DbVisualizer [problem report form](#) or email [support@dbvis.com](mailto:support@dbvis.com). We appreciate detailed reports as well as screen shots when appropriate.

## How to satisfy the DbVisualizer support team

Quite often we get incomplete problem reports and need to follow-up for additional information. If an error or problem happen then you can do the following to let DbVisualizer create system details that you can paste into a support email or in the problem report form:

1. Select the **Connection** tab
2. In the **Connection Message** area select the right click menu
3. In the menu select **Copy**
4. This will copy system details to the clipboard. Then paste the details into an email or in the problem report form mentioned above.
5. A bonus is if you provide screen shots! An image says more then ... you know.



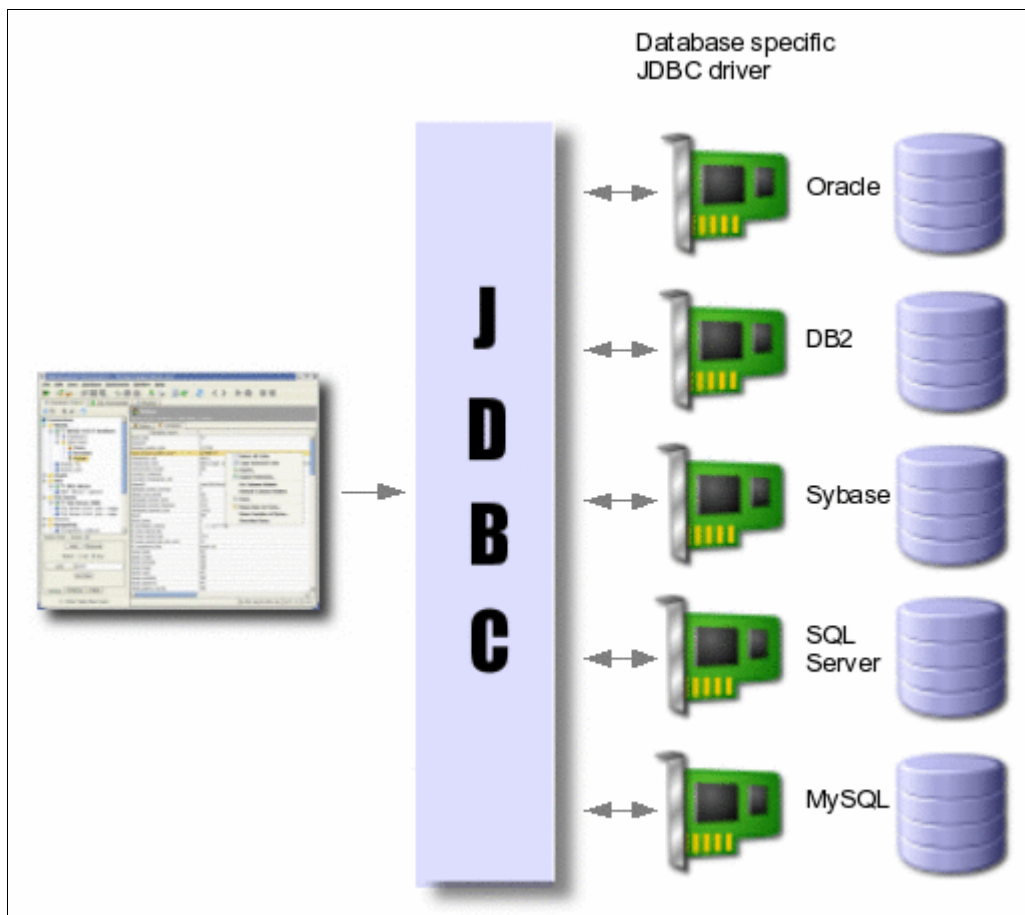
**Figure: The connection message right click menu**



# Load JDBC Driver and Get Connected

## What is a JDBC Driver?

DbVisualizer is as you know a generic tool to administrate and explore databases. DbVisualizer is in fact quite simple since it do not deal with how to communicate with each database. The hard job is done by the JDBC driver which is a set of Java classes that are either organized in a directory structure or collected into a JAR or ZIP file. The magic of these JDBC drivers is that they all match the JDBC specification and the standardized Java interfaces. This is what DbVisualizer relies on. A JDBC driver is a database and database version specific implem/entation and there are a range of drivers from the database vendors themselves and 3:rd party authors. In order to establish a connection with a database using DbVisualizer it needs to load the driver and then get connected to the database through the driver.



**Figure: The runtime environment with the JDBC interface, JDBC driver and sample databases**

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 Driver Manager. These are then used in the connection properties in order to lookup a database connection. The following information explains the steps of how to get connected using a JDBC Driver and also how to use JNDI to obtain a database connection.

A JAR, ZIP or directory that is loaded into the driver manager consists of a number of Java classes that forms the complete implementation of the JDBC driver. DbVisualizer automatically recognize the classes that are used to initiate the connection with the database and presents them in the **Driver Class** list. You must select the correct class in this list to make sure DbVisualizer

successfully can initiate the connection. Consult the driver documentation for information of which class to select or if the number of found classes are low, figure out by trying each of them.

## Get the JDBC driver file(s)

DbVisualizer do not include any JDBC driver so first you must grab a JDBC driver file(s) that works with the actual database and the version of it. The following online web page lists an up to date listing of the tested combinations:

## [Databases and JDBC Drivers](#)

Information about almost all drivers that are available is maintained by Sun Microsystems in this page:

### [JDBC Data Access API - Drivers](#)

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 in order to make the driver files visible to the Driver Manager.

The [Databases and JDBC Drivers](#) web page lists from where to download each driver and also what steps is needed to eventually unpack, install and load the driver in DbVisualizer.

(Drivers are categorized into 4 types. We're not going to explain the differences here but just give a hint that the "type 4" aka "thin" drivers are easiest to maintain since they are pure Java drivers and do not depend on any external DLL's or dynamic libraries i.e try to get a type 4 driver even though DbVisualizer works with any type of driver).

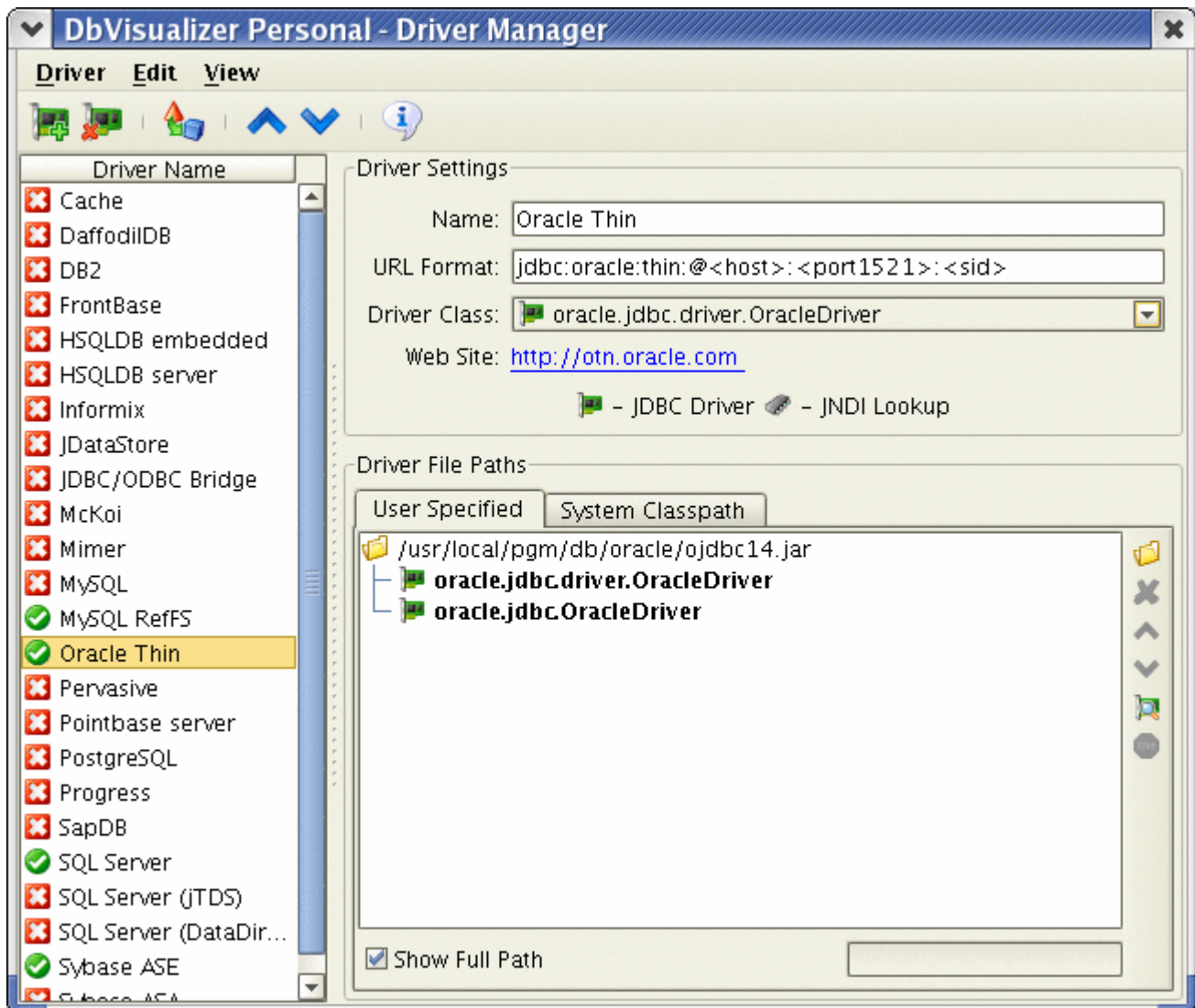
## Driver Manager

The **Driver Manager** in DbVisualizer is used to define the drivers that will be used to communicate with the actual databases. Start the driver manager dialog using the **Tools->Driver Manager** menu choice.

The left part of the driver manager dialog lists a collection of driver names and a symbol indicating whether the driver has been configured or not. The right part displays the definition of 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 name is later listed in the **Connection** tab setup when selecting what driver to use for a **database connection**
- **URL Format**  
The URL format specifies the pattern for the JDBC URL or a JNDI Lookup name. The purpose is to assist the user in the connection tab while entering the URL or lookup name
- **Default Class**  
Defines the default class to use when connecting
- **Web Site**  
Link to the DbVisualizer web site containing up to date information how to download the driver.
- **Driver File Paths**  
Defines all paths to search for JDBC drivers or Initial Contexts during connect with the database. The Driver File Paths is composed of two tabs, the **User Specified** tab is used to locate and identify dynamically loaded JDBC drivers or Initial Context classes. The **System Classpath** tab lists all paths that are part of the Java system classpath.

**Note:** Do not bother about the System Classpath tab unless you are using the [JDBC-ODBC](#) driver.



**Figure: Driver Manager dialog**

The driver list contains initially a collection of default drivers. These are not fully configured as the actual paths used to search for the classes need to be identified. The list can be edited as drivers can be created, copied, removed and renamed. A driver is ready once a default class has been identified and this state is indicated with a green check icon in the list. Not ready drivers are indicated with a red cross icon.

**Note:** Only ready (configured) drivers will appear in the Connection tab driver list.

The figure shows four drivers that are ready, **MySQL RefFS**, **Oracle Thin**, **SQL Server** and **Sybase ASE**.

## Setup a JDBC driver

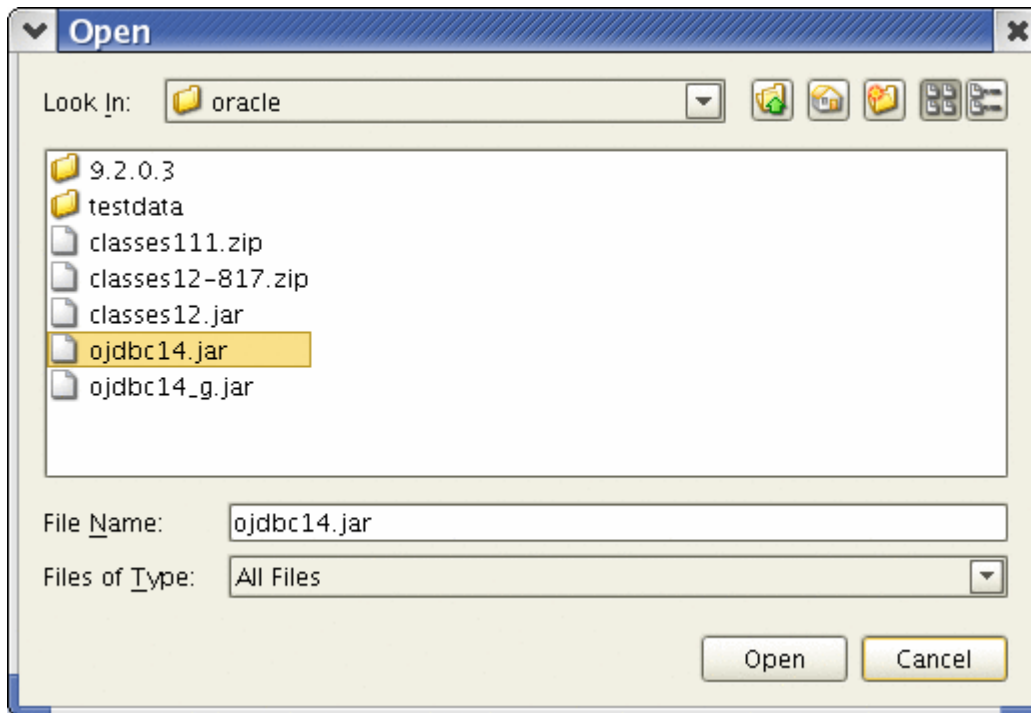
The recommended way to setup a driver is to pick a matching driver name from the list and then simply load the JAR, ZIP or directory that keeps the actual driver class(es) i.e if you are going to load the JDBC driver for **Oracle** then select the Oracle driver in the list. You can also create a new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

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

- It lists what databases and drivers that has been tested
- Download links to JDBC drivers
- Information of what files to load in the driver manager for each JDBC driver
- Information of what **Driver Class** that should be choosen

## Databases and JDBC Drivers

To load jar file(s) then press the **Load** button to the right of the **User Specified** paths tree to show the file chooser and load the driver jar(s).



**Figure: File Chooser dialog**

It is important to load the root of the JDBC Driver i.e a JDBC Driver implementation consists in most cases of several Java classes. These are also in most cases organized using the package mechanism in Java. Example:

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

Each package part in the name above (separated by ".") will be represented by a directory in the file system. These directories are either explicitly visible in the file system or implicitly if the driver is packaged in a ZIP or JAR file. The root of the driver is in this case where the **oracle** directory is located. In the Oracle example this is the **ojdbc14.jar** JAR file so the driver manager must load this path in order to find the driver class. If the driver is packaged in a ZIP file or a directory then point the driver manager to that path in order for the driver manager to locate the driver class.

Once a connection is established in the Connection tab will DbVisualizer search the selected drivers path tree's in the following order:

1. User Specified
2. System Classpath

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

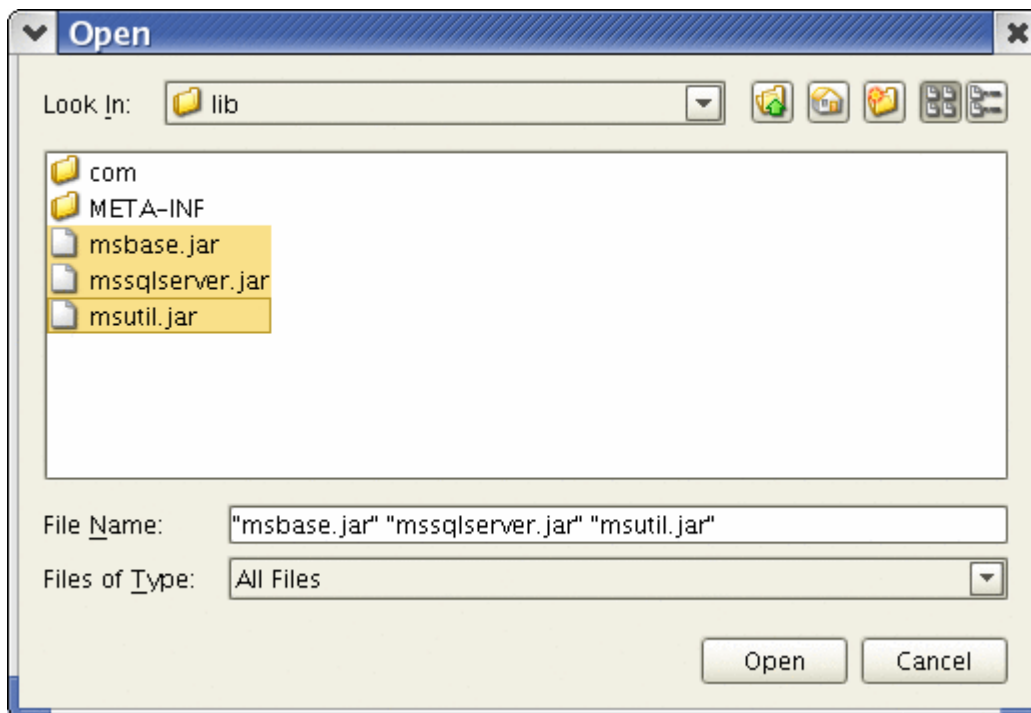
handle multiple versions of a driver is to create several driver definitions.

Once one or several classes has been identified and listed in the **Driver Class** list then make sure you select the correct **Driver Class** from the list. See the table earlier for assistance.

### JDBC drivers that requires several JAR or ZIP files

Some drivers depend on several ZIP, JAR files or directories. An example is the JDBC driver for **Microsoft SQL Server** that requires three different JAR files to be loaded. In this case simply load all JAR files even though the driver manager will only report the driver class (`com.microsoft.jdbc.sqlserver.SQLServerDriver`) in one of them.

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 in the tree.



**Figure: File Chooser dialog**

### The JDBC-ODBC bridge

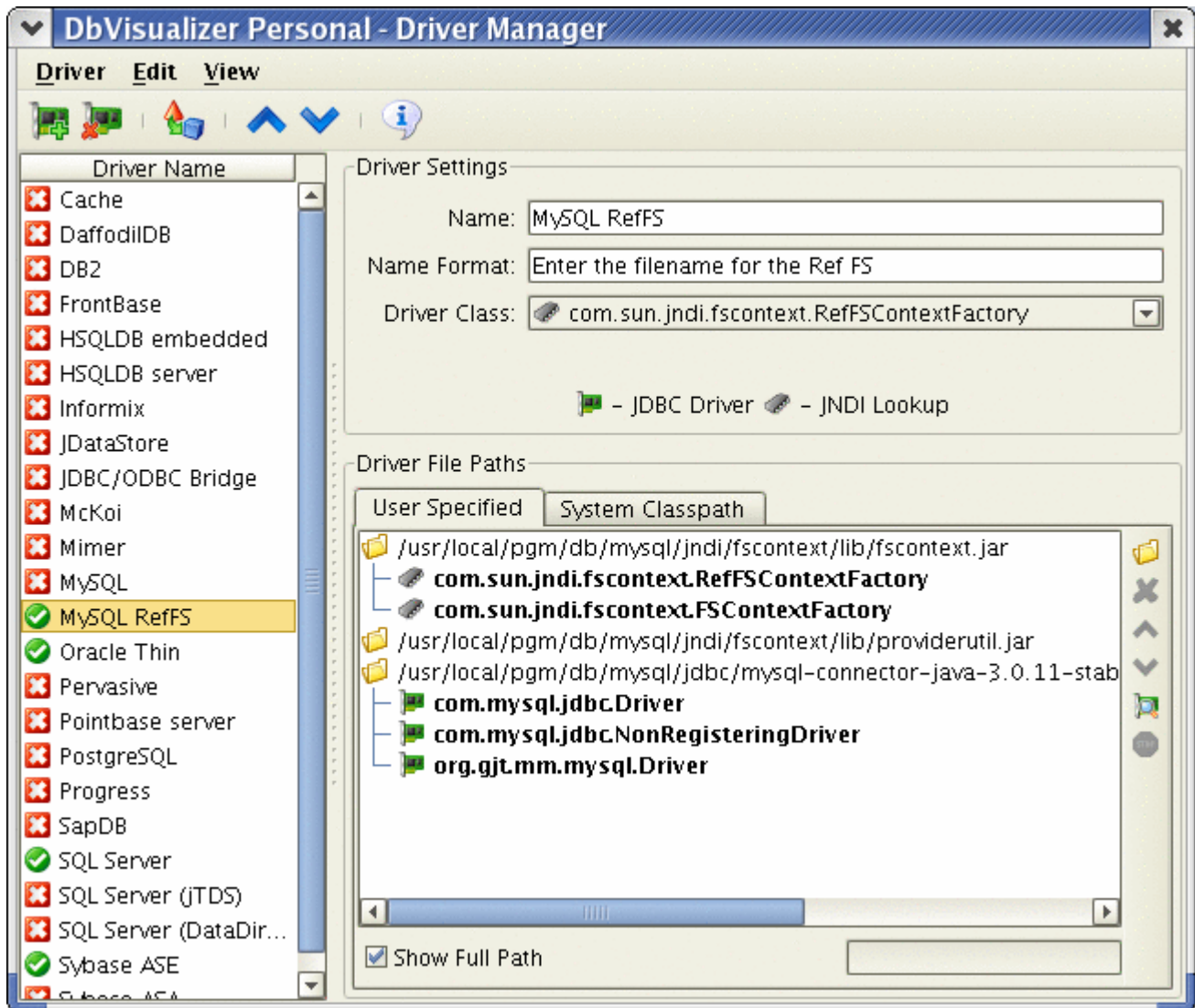
The JDBC-ODBC driver is by default part of most Java installations. The **JdbcOdbcDriver** class is included in a JAR file that is commonly named **rt.jar** and is stored somewhere in the Java directory structure. DbVisualizer automatically identifies this JAR file in the System Classpath tree. To locate the **JdbcOdbcDriver** simply press the **Find Drivers** button to the right of the System Classpath tree. Once found then make sure the **sun.jdbc.odbc.JdbcOdbcDriver** is selected as the **Default Class**.

### Loading JNDI Initial Contexts

Initial Context classes are needed in order to get a handle to a database connection that is registered in a JNDI lookup service. These classes are similar to JDBC driver classes since an Initial Context implementation is required.

**Note:** Remember that the appropriate JDBC driver classes must be loaded into the Driver Manager even if the database connection is obtained using JNDI.

To load Initial Context classes into the Driver Manager simply follow the steps outlined for loading JDBC drivers. The difference is that you will instead load locations that contain Initial Context classes instead of JDBC drivers. Once Initial Context classes have been found the following will appear in the Driver Manager list.



**Figure: Driver Manager List with Initial Context classes**

The visual difference between the identified JDBC drivers and Initial Context classes is the icon in the tree.

The figure shows the required JAR files in order to first obtain the JNDI handle and then also the actual JDBC driver that is needed to interface the database. Check with the application server vendor or similar for more information of what files that need to be loaded to get connected via JNDI.

### **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

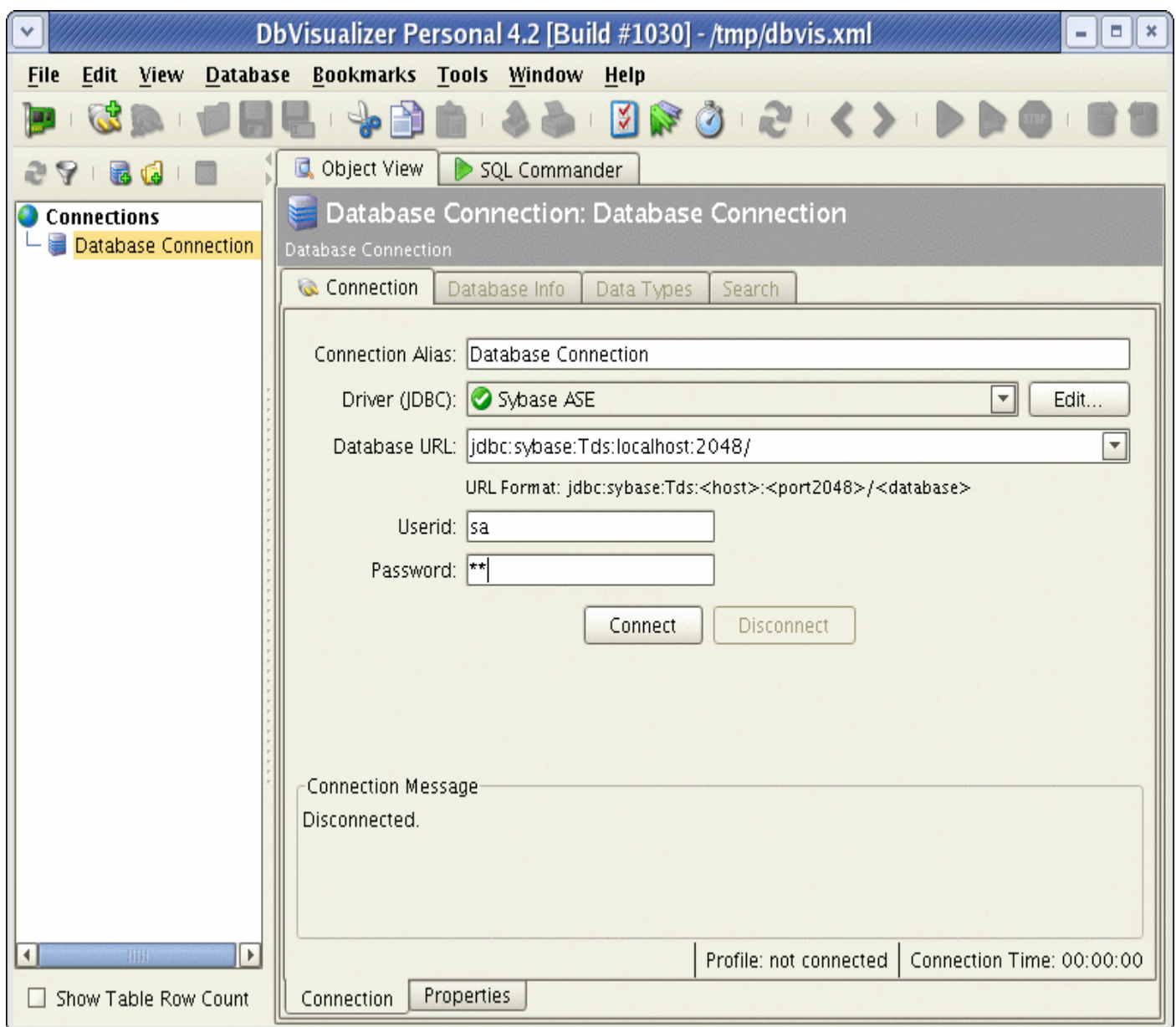
The Driver Manager supports loading and then using several versions of the same driver concurrently. The recommendation is to create a unique driver definition per version of the driver and then name the drivers properly. Ex. **Oracle 9.2.0.1**, **Oracle 10.2.1.0.1**, etc.

## Setup a database connection

This section explains how to setup a Database Connection in the Connection tab.

### Setup using JDBC driver

A Database Connection in DbVisualizer is the root of all communication with a specific database. It requires at a minimum that a driver is selected and a **Database URL** is specified. A new Database Connection is created using the **Database->Add Database Connection** menu choice in the main window:



**Figure: New Database Connection using JDBC driver**

The **Connection** tab is the only enabled tab if you are not already connected to the database.

Database connection objects appear throughout the application and are by default listed by their **URL**. A URL can be, and often is, quite complex and long. The **Database Alias** is used to optionally set a more readable name of the database connection. The **Driver** list when opened shows all defined drivers that have been defined properly in the Driver Manager. Just open the list and select the appropriate driver. The URL Format lists the format that the driver supports.

**Tip:** Put the mouse pointer on the URL Format and click with the mouse to copy the format template into the URL field.

The < and > characters indicate that they are the boundary for a placeholder and that they shall be replaced with appropriate values. Ex.

```
jdbc:oracle:thin:@proddb:1521:bookstore  
jdbc:sybase:Tds:localhost:2638  
jdbc:db2://localhost/crm  
jdbc:microsoft:sqlserver://localhost;DatabaseName=customers
```

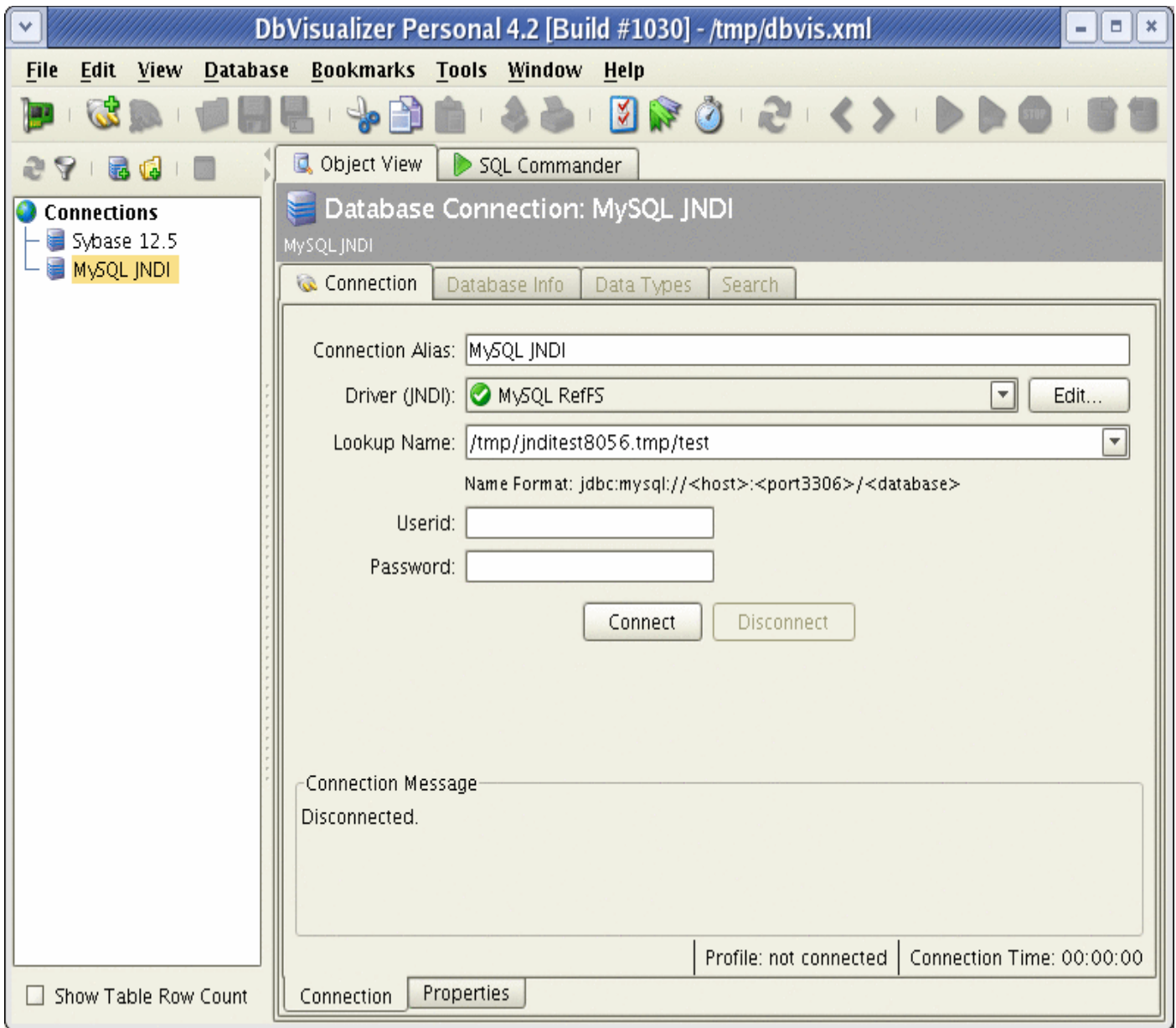
**Userid** and **Password** is optional but most databases require that they are specified.

Some drivers accept additional proprietary parameters described in the [Connection Properties](#) section.

## Setup using JNDI lookup

The information needed in order to obtain a database connection using JNDI lookup is similar to getting connected using a JDBC driver.





**Figure: New Database Connection using JNDI lookup**

The figure above shows parameters to connect with a lookup service via the MySQL RefFS driver. The `/tmp/jnditest8056.tmp/test` lookup name specifies a logical name for the database connection that will be used. This example is in its simplest form since userid and password is not specified, nor where the database connection is finally fetched from. Any errors during the process of getting a handle to the database connection will appear in the **Connection Message** area.

## Connection Properties

Some of the connection properties are used to override the generic properties available in the Tool Properties window. There are two ways to change the property for a database connection:

- **Tool Properties**

These changes will be applied to all database connections that have not overridden the actual properties in its Connection Properties.

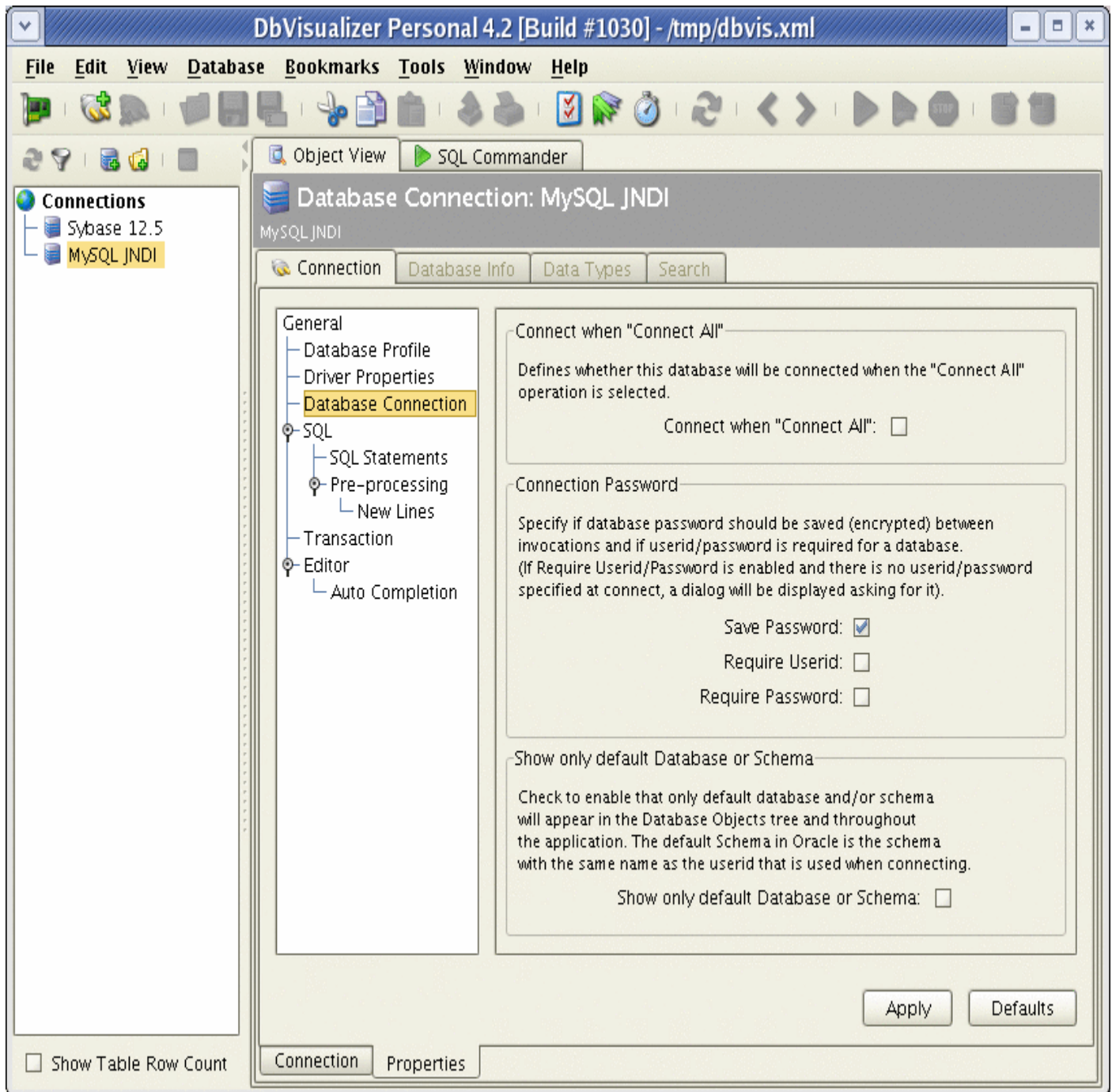
- **Connection Properties**

These changes apply for the actual database connection only.

- "Okay, so there are two places to change the value of a property. Which shall I use?"

This depends on whether the change should be applied to all database connections or not. If the majority of database connections should use the new value then it is recommended to set it in Tool Properties.

Any overridden properties in the Connection Properties tab are indicated with an icon in the **Properties** tab label.



**Figure: Connection Properties**

The connection properties tab is organized in the same way as the tool properties window. The difference is that the list only includes the categories that are applicable for a database connection. The categories are briefly:

- Database Profile
- Driver Properties
- Database Connection
- SQL
  - SQL Statements

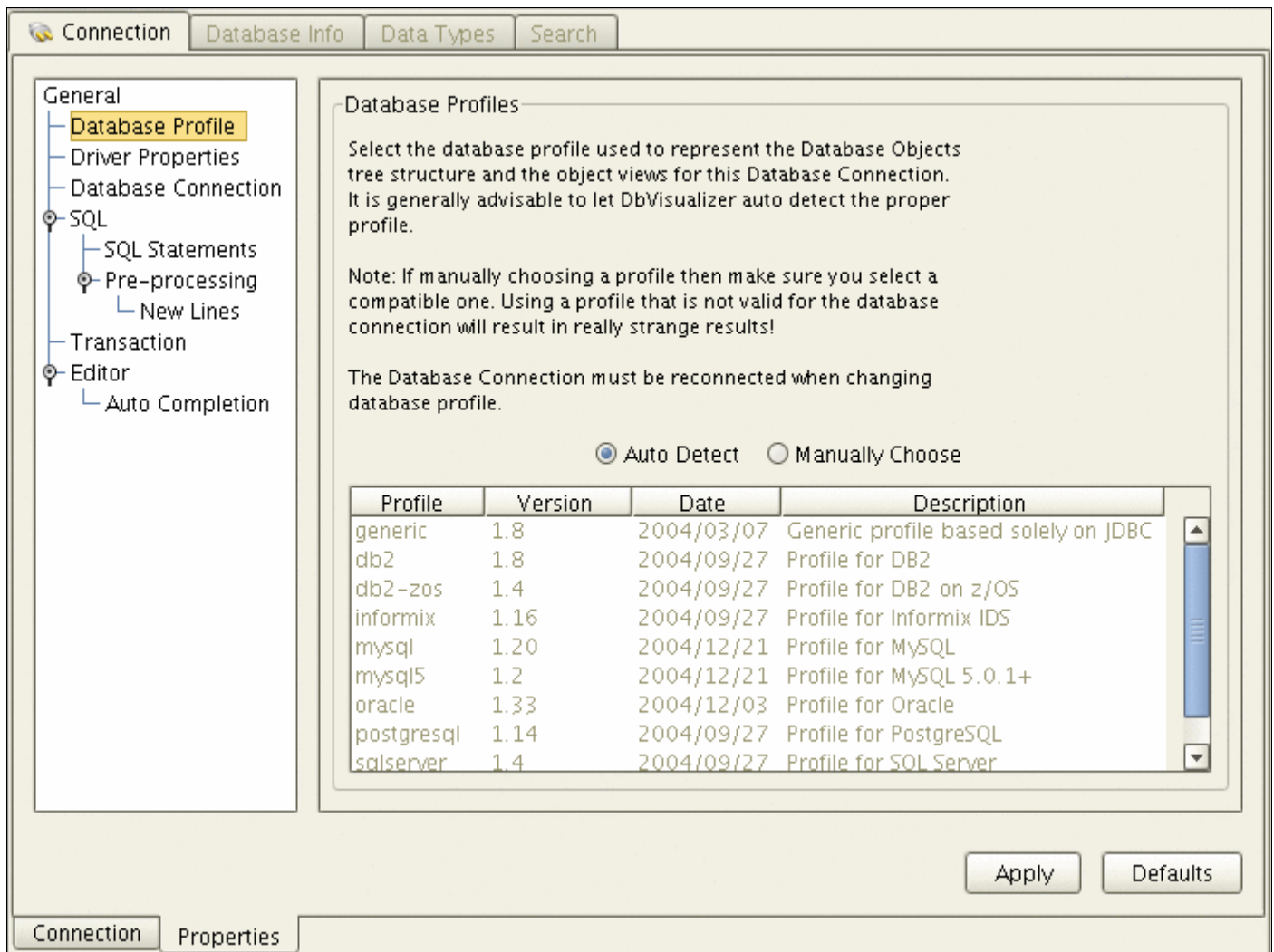
- Pre-processing
  - New Lines
- Transaction
- Editor
  - Auto Completion

The **Database Profile** and **Driver Properties** categories are only available in the Connection Properties tab and not in Tool Properties. The next section explains the Database Profile and Driver Properties categories while the other categories are described in the [Tool Properties](#) document.

## Database Profile

Please read in the [Database Objects Explorer](#) document for detailed information about database profiles.

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.



**Figure: Database Profile category for a database connection**

**Note:** The way DbVisualizer auto detects a profile is based on mappings in the **DBVIS-HOME/resources/database-mappings.xml** file.

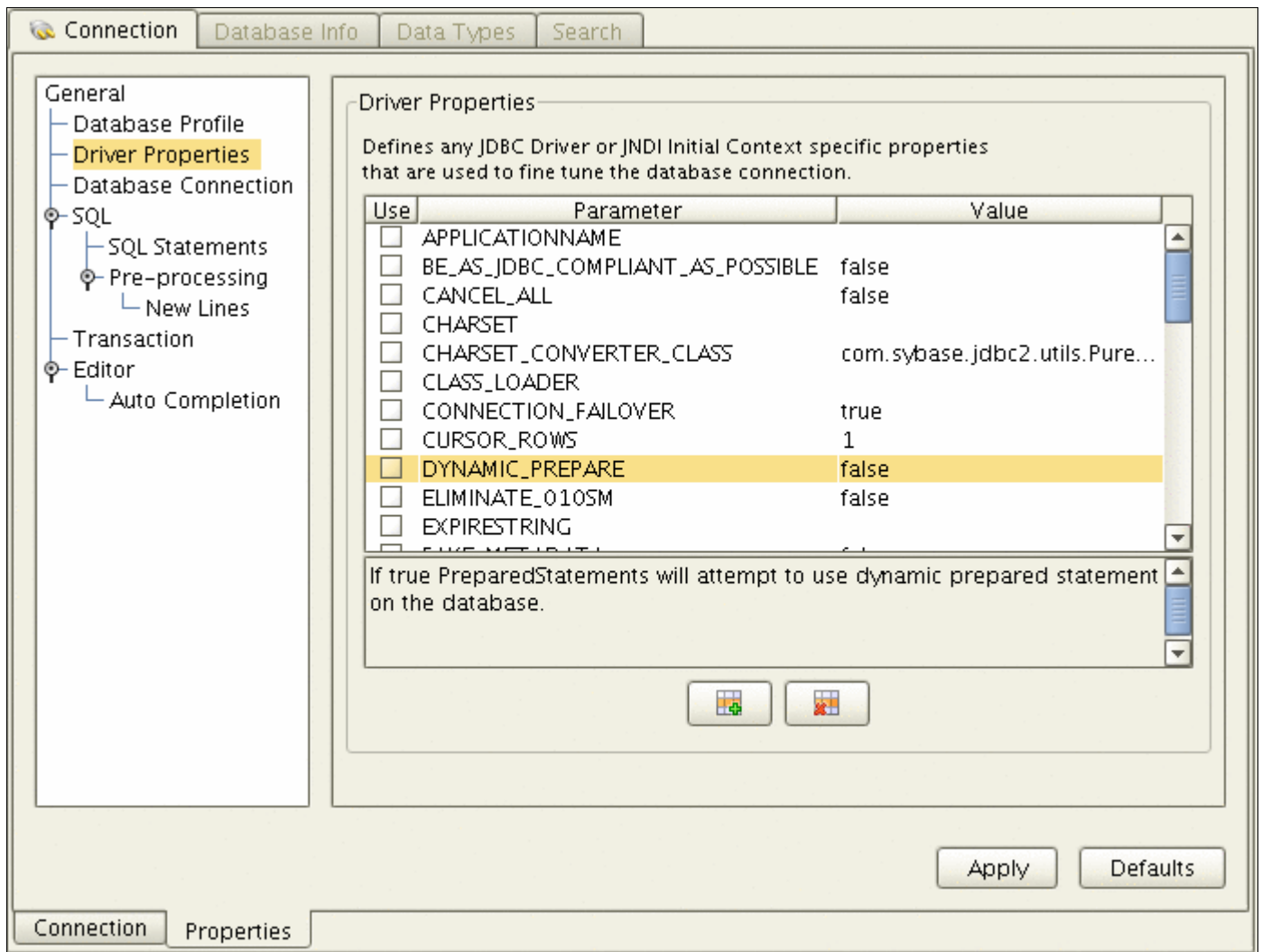
If you manually choose a database profile then this choice will be saved between invocations of DbVisualizer.

## Driver Properties

The Driver Properties category is used to fine tune a driver or Initial Context before the database connection is established.

### Driver Properties for JDBC Driver

Some JDBC drivers support driver specific properties that are not covered in the JDBC specification. New properties can be added for testing purposes (driver tests).



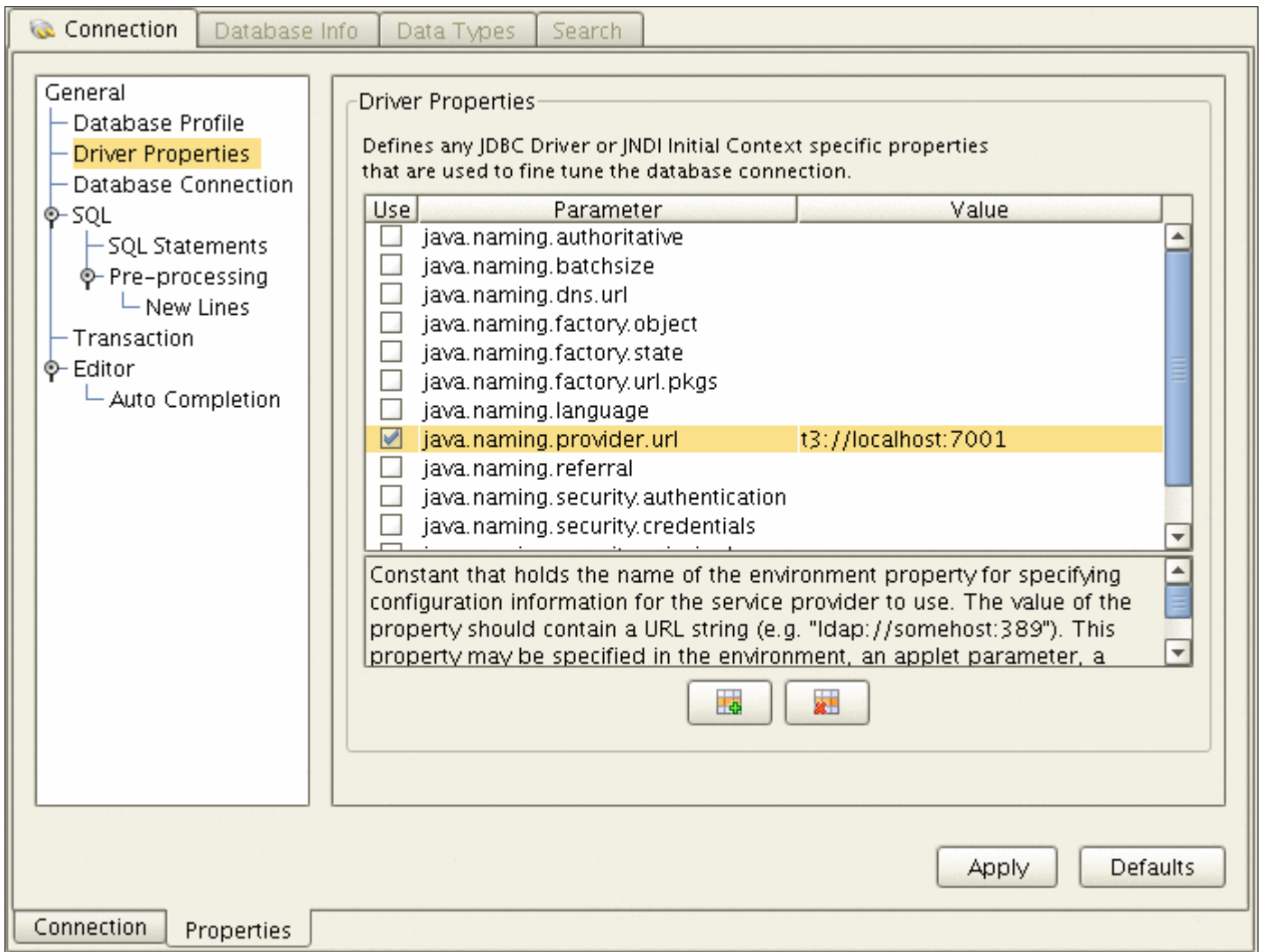
**Figure: Driver Properties for JDBC Driver**

The list of parameters, their default values and parameter descriptions are determined by the actual driver. 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 at the bottom of the dialog. Be aware that additional parameters do not necessarily mean that the driver will do anything with them.

### Driver Properties for JNDI Lookup

The Driver Properties category for a JNDI Lookup connection always contain the same parameters.



**Figure: Driver Properties for JNDI lookup**

The list of options for JNDI lookup is determined by the constants in the **javax.naming.Context** class. To change a value just modify the value of the parameter. 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 at the bottom of the dialog. Be aware that additional parameters do not necessarily mean that the InitialContext class will do anything with them.

### **Always ask for userid and/or password**

Userid and password information is generally information that should be handled with great care. DbVisualizer saves by default both userid and password (encrypted) for each database connection. Userid is always saved while password saving can be disabled in the connection properties.

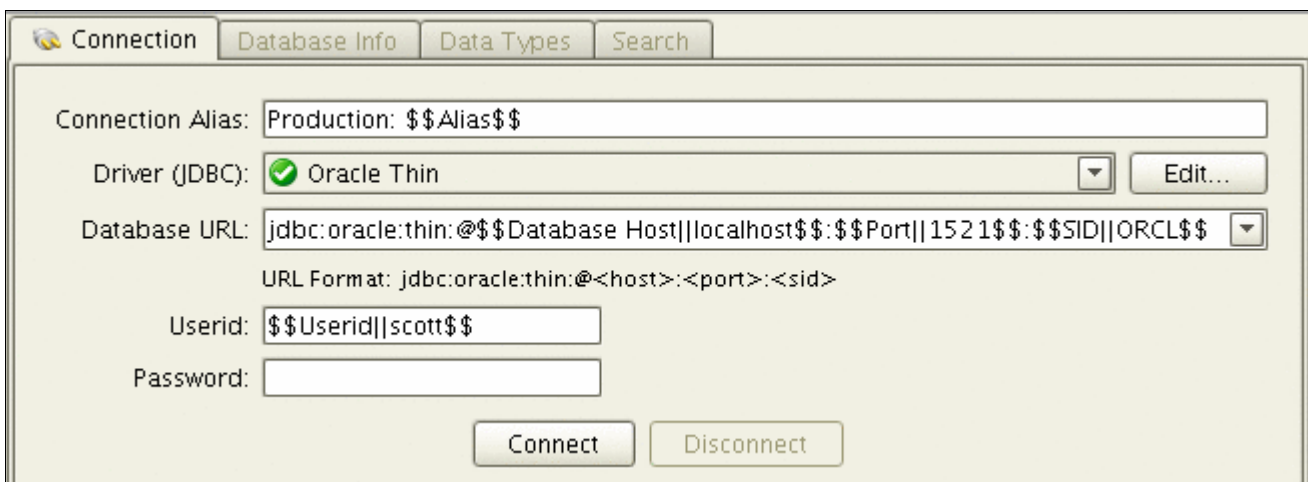
The **Require Userid** and **Require Password** connection properties can be enabled to control that DbVisualizer automatically should prompt for userid and/or password once a connection is established. Enabling either one or both of these and leaving the **Userid** and **Password** fields blank for a database connection ensures that DbVisualizer will not keep this vital information between sessions. The following figure is displayed if requiring both userid and password.



**Figure: Dialog asking for Userid and Password as a result of having Require Userid and Password settings enabled**

## Using variables in the Connection details

Variables can be used in any of the fields in the Connection tab. This can be useful instead of having a lot of similar database connection objects. Several variables can be in a single field and default values can be set for each variable. The following figure shows an example of variables that are identified by the dollar characters, \$\$ . . . \$\$.



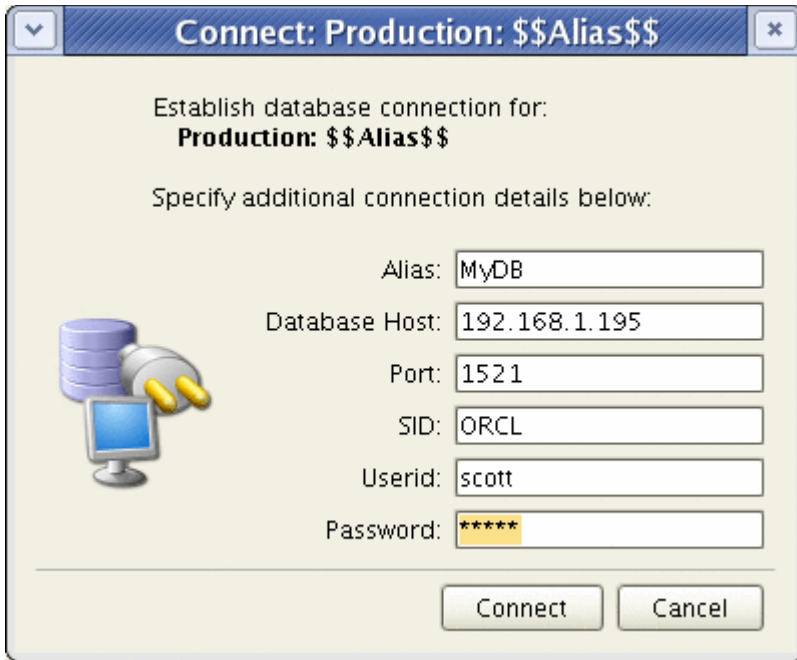
**Figure: Connection tab with variables**

The following variables appear in the figure:

- \$\$Alias\$\$
- \$\$Database Host||localhost\$\$
- \$\$Port||1521\$\$
- \$\$SID||ORCL\$\$
- \$\$Userid||scott\$\$

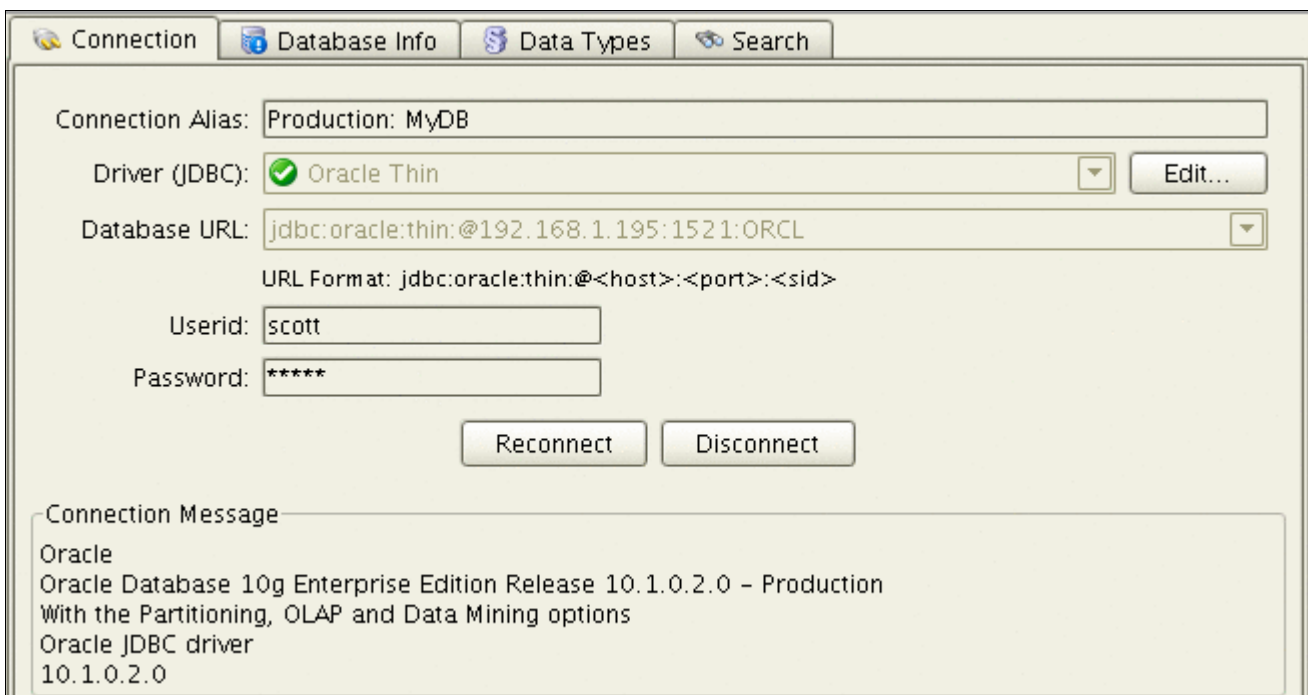
All of these variables defines a default value after the "|" delimiter except \$\$Alias\$\$ that have no default value. These default values will appear in the connect dialog once a connection is requested. The following figure shows the connect dialog based on the information above.

**Note:** Using variables in conjunction with the **Require Userid** and/or **Require Password** settings also works.



**Figure: Connection tab with variables**

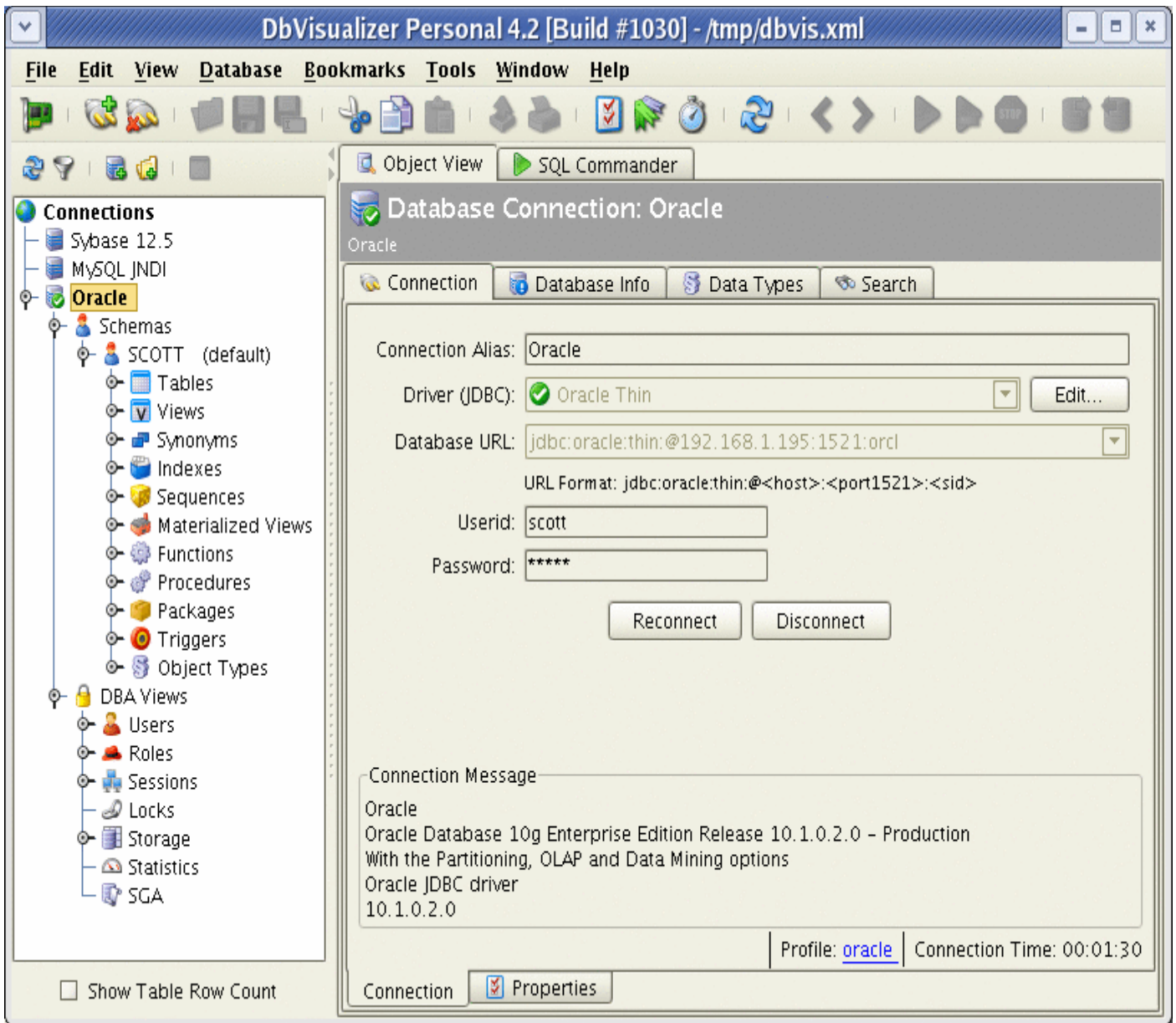
Enter the appropriate information in the fields and then press the **Connect** button to establish the connection. Once connected will DbVisualizer automatically substitute the variables in the Connection tab with the values entered in the connect dialog. These will at disconnect from the database revert back to the original variable definitions.



**Figure: Connection tab as it look once connected using variables**

## Connect to the Database

Press **Connect** when all information has been specified. DbVisualizer will pass all entered information onto the selected driver and if the connection is established the following will appear.



**Figure: A freshly initiated database connection using JDBC driver**

The **Connection Message** now lists the name and version of the database as well as the name and version of the JDBC driver. The database connection node in the tree indicates that it is connected. The connection properties cannot be edited once while a database connection is established. The **Alias** can be edited by selecting the database connection node in the tree and then clicking on the name.

The figure above also shows that the database connection node in the tree has been expanded to show its child objects.

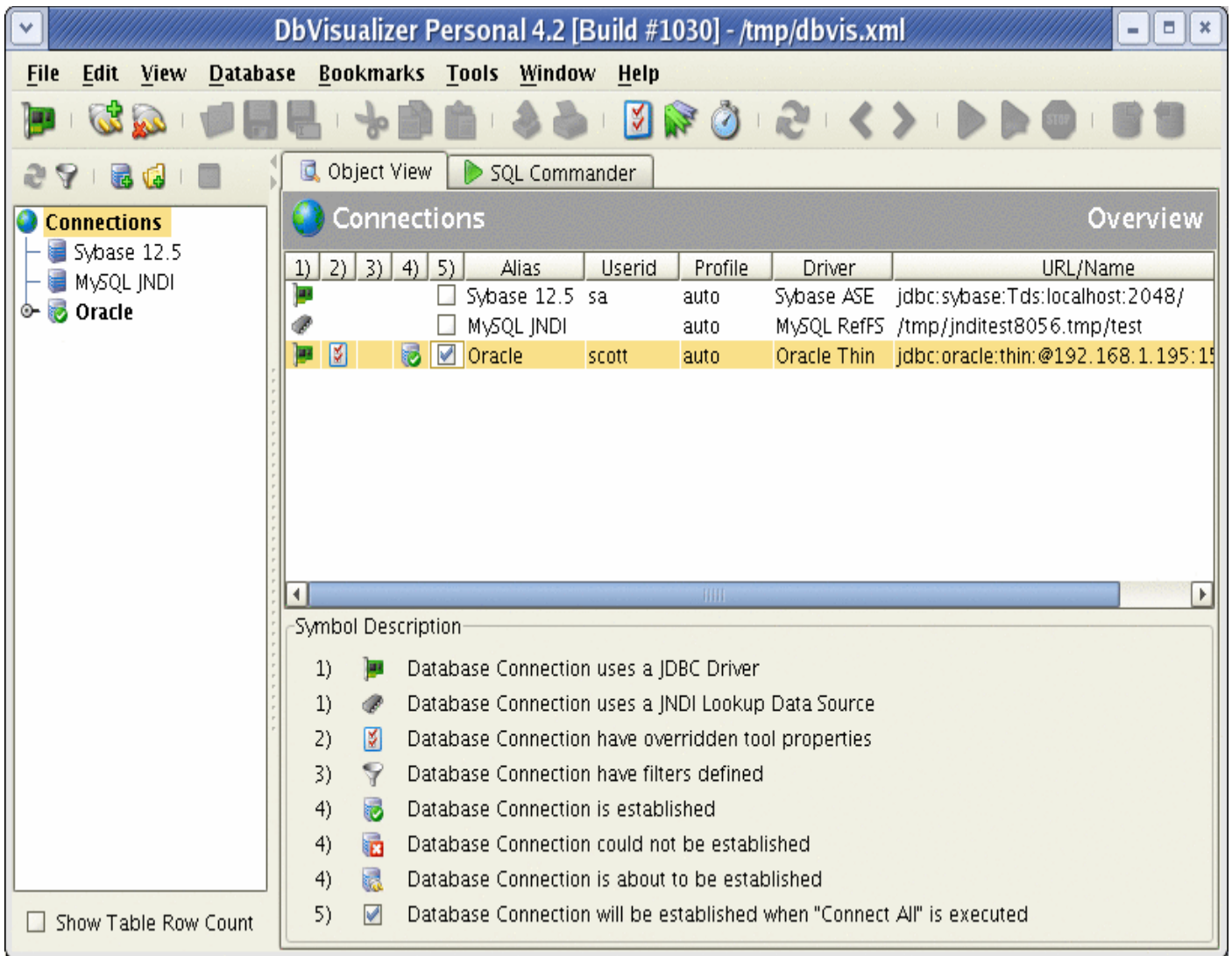
If the connection is unsuccessful it will be indicated by an error icon in the tree. The error message as reported by the database or the driver will appear in the **Connection Message** area. Use this to track the actual problem. Since these conditions are specific for the combination of driver and database it is generally recommended to check the driver and database documentation to find out more. Below are a few common problem situations:



<p>No suitable driver. There is no driver that can handle a connection for the specified URL. The most common reason is that the driver is not loaded in the Driver Manager. Also make sure the URL is correct spelled.</p>	<p>The JDBC support in Java determines what driver to load based on the database URL. If the URL is malformed then there might be no driver that is able to handle the database connection based on that URL. This error is produced when this situation occurs or when the driver is not loaded in the driver manager. The recommendation is to check the JDBC driver documentation for the correct syntax.</p>
<pre>java.sql.SQLException: Io exception: Invalid number format for port number Io exception: Invalid number format for port number</pre>	<p>The URL templates that are available in the Database URL list contains the "&lt;" and "&gt;" place holders. These are there to indicate that the value between them must be replaced with an appropriate value. The "&lt;" and "&gt;" characters must then be removed.</p> <p>This example error message is produced by the Oracle driver when using the following URL:</p> <pre>jdbc:oracle:thin:@&lt;qinda&gt;:&lt;1521&gt;:&lt;fuji&gt;</pre> <p>Simply remove the "&lt;" and "&gt;" characters and try again.</p>

## Connections Overview

The Connections overview is displayed by selecting the **Connections** object in the Database Objects Tree. This overview displays all database connections in a list and is handy to get a quick overview of all connections. In addition to the URL, driver, etc there are a few symbols describing the state of each connection. Double clicking on a connection will change the display to show that specific connection.



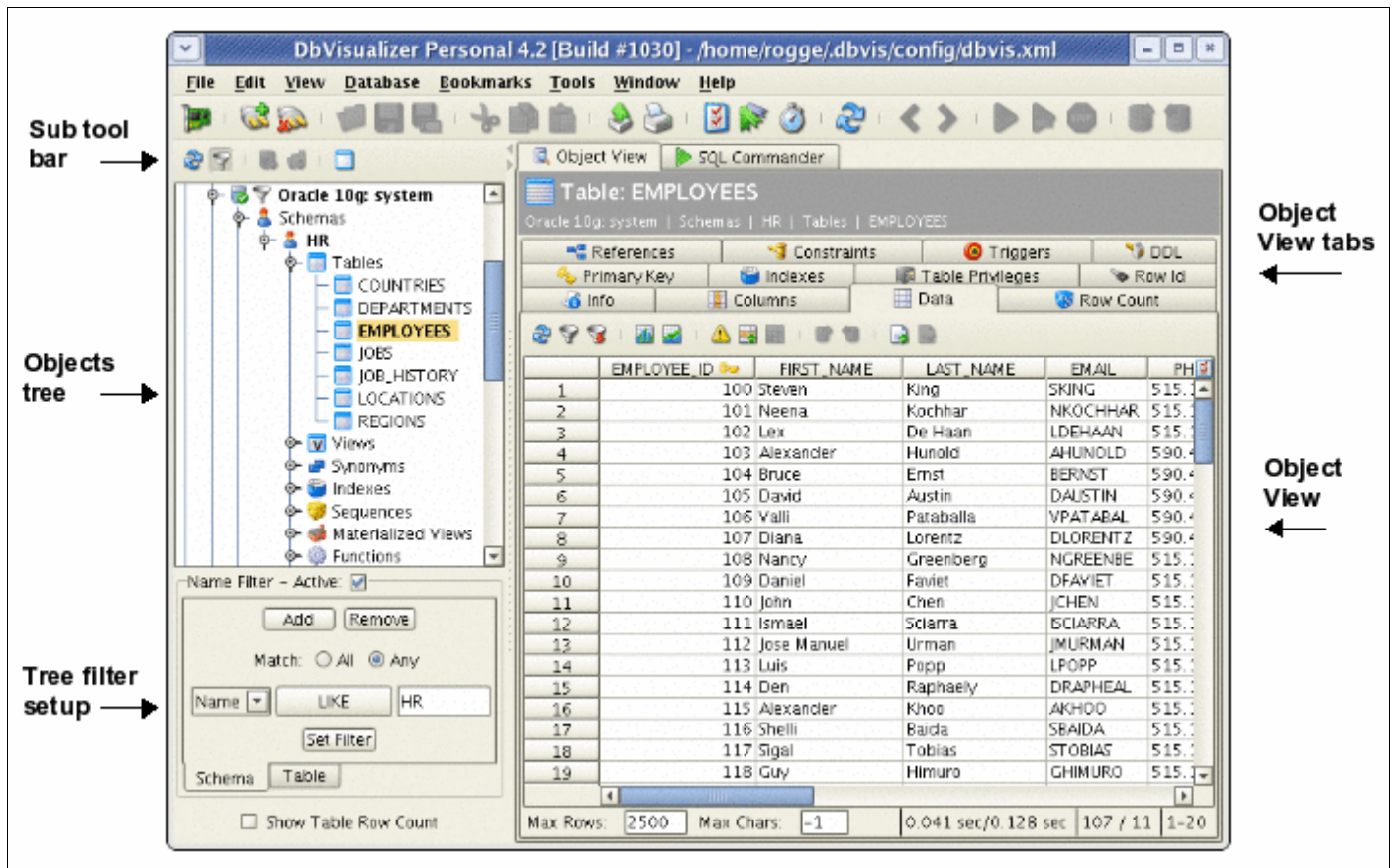
**Figure: The Connections Overview**

Information for each symbol is provided in the description area below the list. The fifth check symbol is the only editable symbol and is used to set the state of the Connect when Connect All property.

# Database Objects Explorer

## Introduction

The **Database Objects Tree** in conjunction with the **Object View** tab is used to explore all of your databases and to show detailed information about selected objects.



**Figure: Database Objects tab**

The **Database Objects Tree** to the left is the place to define new database connection objects and establish connections. Once connected click the database connection object and explore the child objects that are available. The right **Object View** area displays when selected information about the currently selected object in the tree.

The **tab tool bar** buttons are used to perform various operations on the tree and the objects in it. The **Tree Filter** settings are used to control by name what objects are displayed in the tree. It is handy in order to limit the number of objects.

All object views except the table Data editing facility are used to browse the database. This means that DbVisualizer Personal can be used to browse the source for functions, procedures, triggers, etc but it does not allow manipulation of such other than using the **SQL Commander** to perform any changes.

**Tip:** The Database Objects Tree is always visible to the left. If the currently selected main tab is the SQL Commander then you can double click on an object in the tree to automatically switch to the **Object View** tab.

## Database Profiles

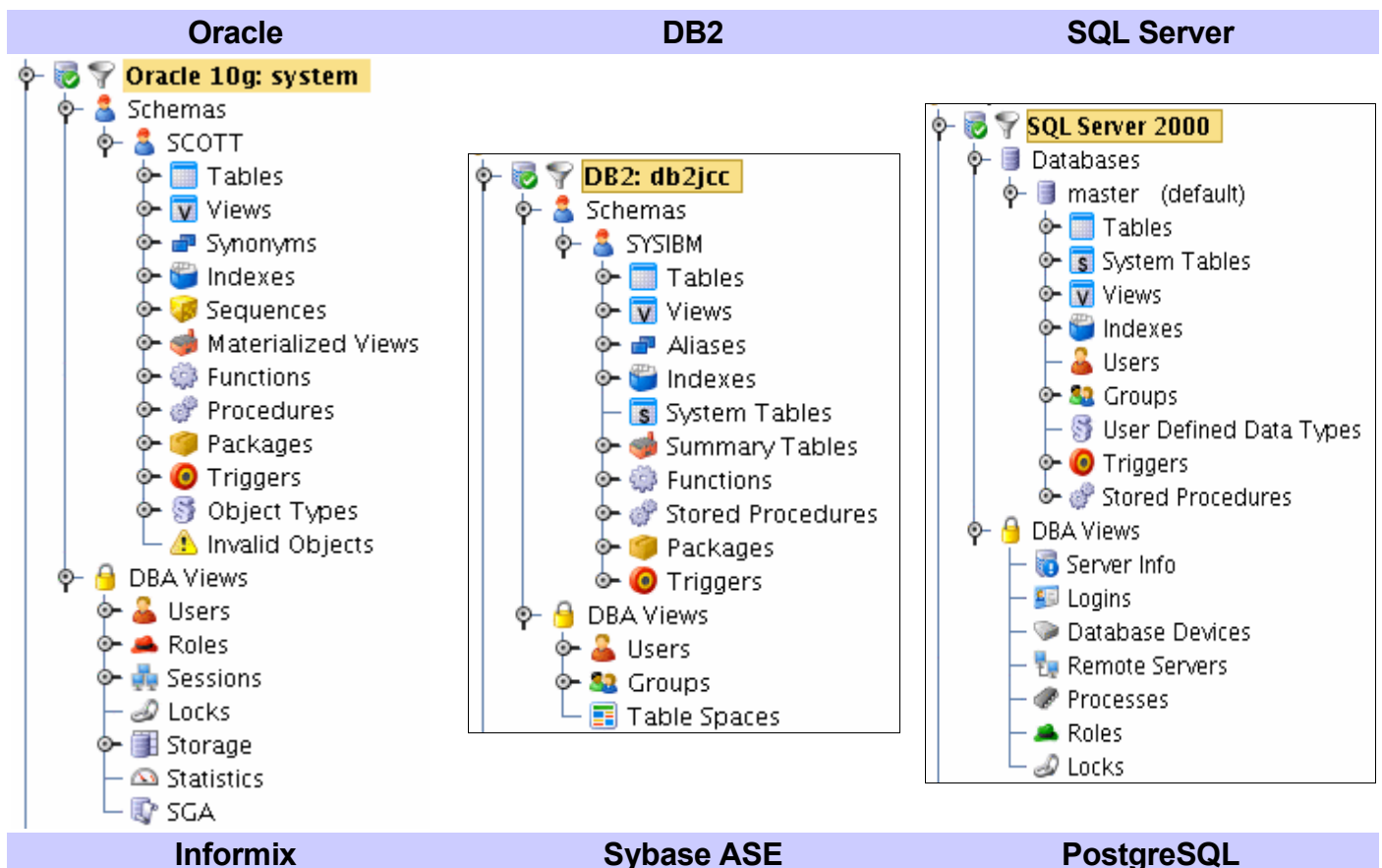
Since DbVisualizer Personal 4.1 there are a collection of new database specific objects represented both in the objects tree and in the various objects views. The definition of what is shown is defined in a **database profile** and DbVisualizer Personal currently supports specialized profiles for the following databases:

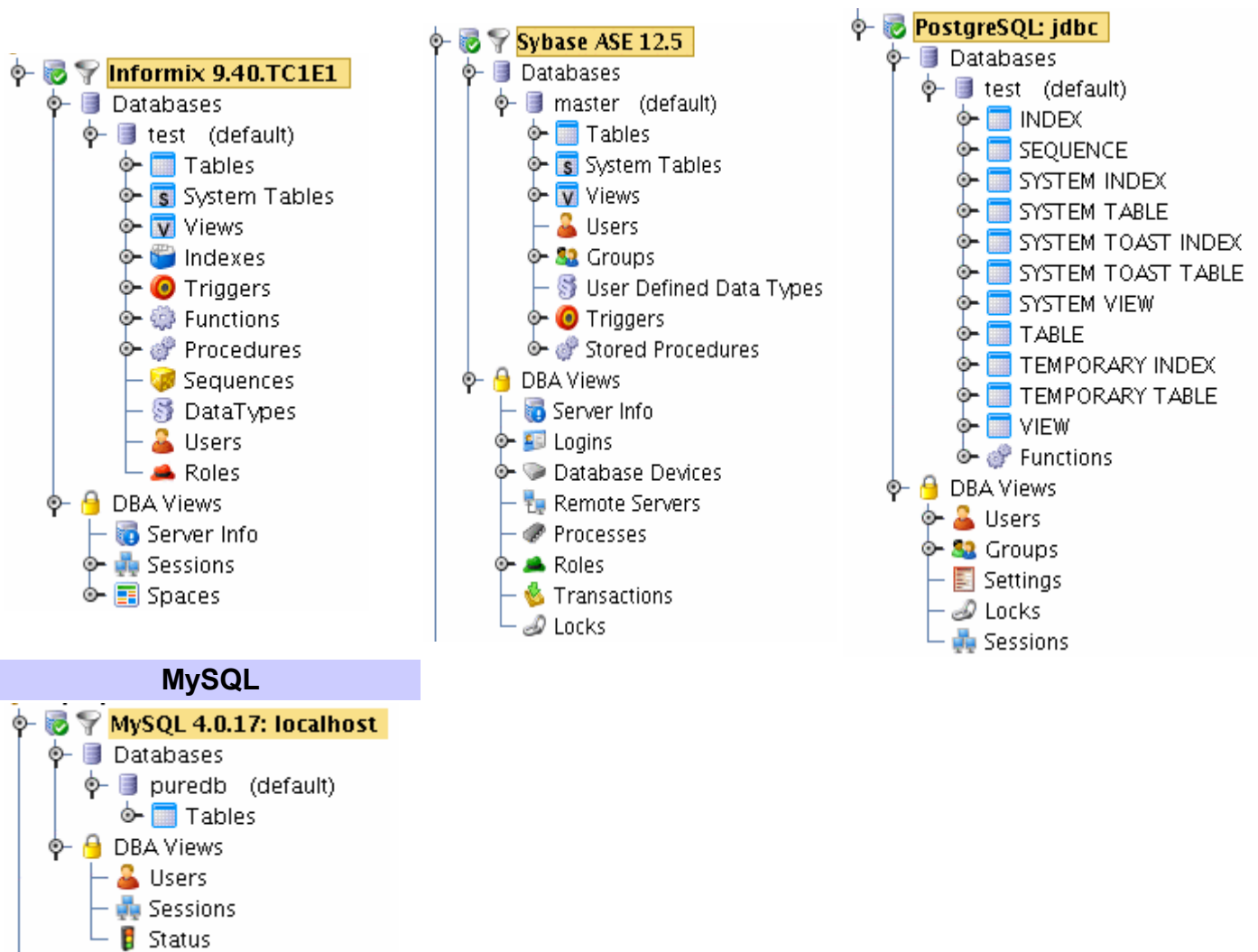
- Oracle
- DB2
- Sybase ASE
- SQL Server
- MySQL
- PostgreSQL
- Informix

The information displayed is specifically per each of these databases and the following document will cover Oracle as an example. For databases where no specialized database profile exist or when using DbVisualizer Free, a generic database profile is used. It displays basic information about catalogs (aka databases), schemas, tables and procedures.

### Objects tree for Oracle, DB2, SQL Server, Informix, Sybase ASE, PostgreSQL and MySQL

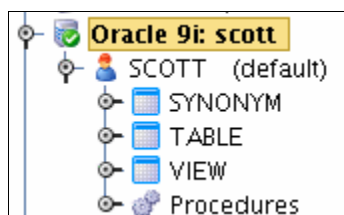
The following shows what each database profile displays in the objects tree.





**Figure: Overview of what objects each supported database profile displays**

The generic database profile when used for an Oracle connection look as follows:



**Figure: The generic database profile when applied to an Oracle database connection**

The appearance of the generic database profile may include **schema** objects and/or **catalog** objects depending on whether the actual database supports these objects or not. The **Procedures** object always appear in the tree independent on whether the database connection supports procedures or not.

**Note:** The generic database profile is always used in DbVisualizer Free.

Read more about the generic tree objects in the [Generic Database Profile](#) section.

# Database Objects Tree

## Tab tool bar operations

The Database Objects tool bar buttons are used to do tree related operations. These are individually enabled or disabled based on what object is currently selected.



Description of the buttons from the left:

Tool bar button	Description
<b>Reload</b>	Reloads the currently selected object by asking the JDBC driver to fetch information for the object from the database. This is useful if new objects have been created or removed.
<b>Toggle display of Filter setup</b>	Is a toggle button that determines whether the Filter management pane will be displayed below the tree or not.
<b>Create Database Connection</b>	Adds a new Database Connection object in the tree. The location of the new object is determined based on the current selection. If no selection then the new object added to the end of the list.
<b>Create Folder</b>	Creates a new folder object.
<b>Show in Window</b>	Request to display the details view in a separate window for the selected object.

## Right click menu operations

The right click menu contains the following operations:

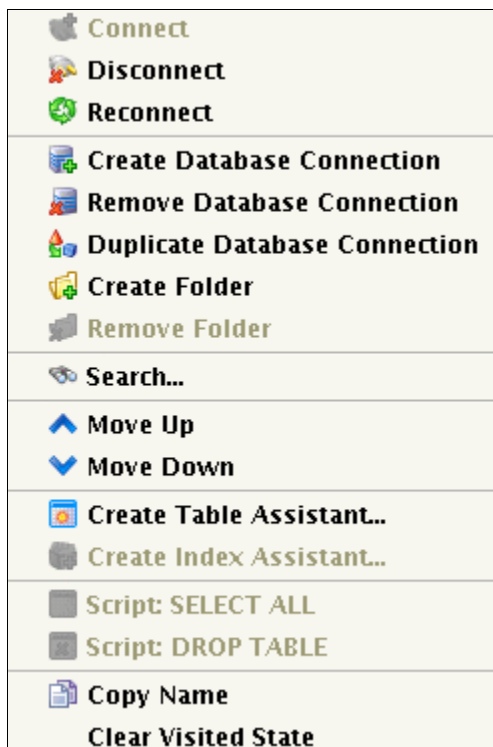


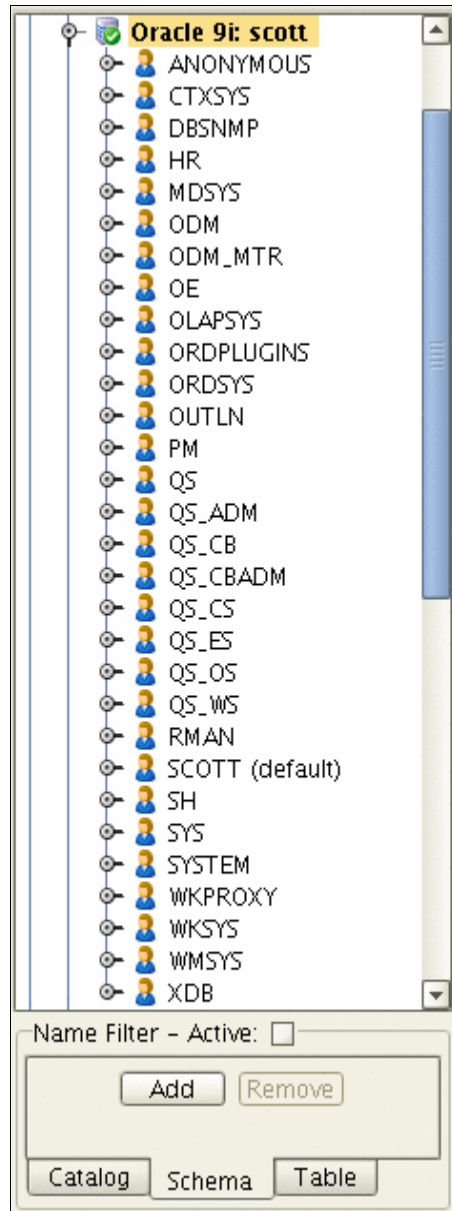
Figure: The tree right click menu

## Filtering

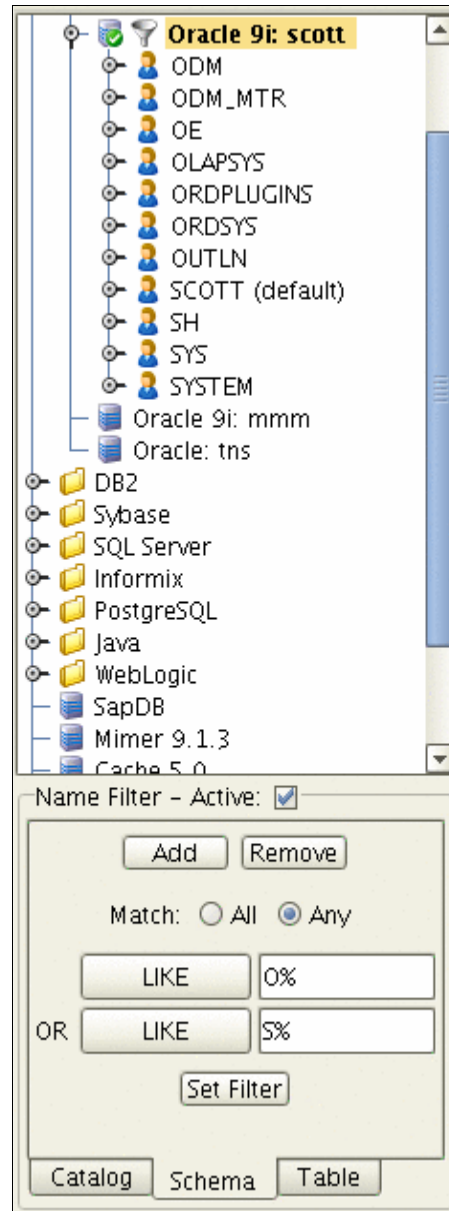
The **Filtering** setup is located below the database objects tree. The visible state of this pane can be controlled using the filter button in the database objects tool bar. Filtering is useful to limit the number of objects that will appear in the tree.

Tree filters are managed per database connection object. What can be filtered is defined per database profile. The generic database profile supports filtering on catalog, schema and table names.

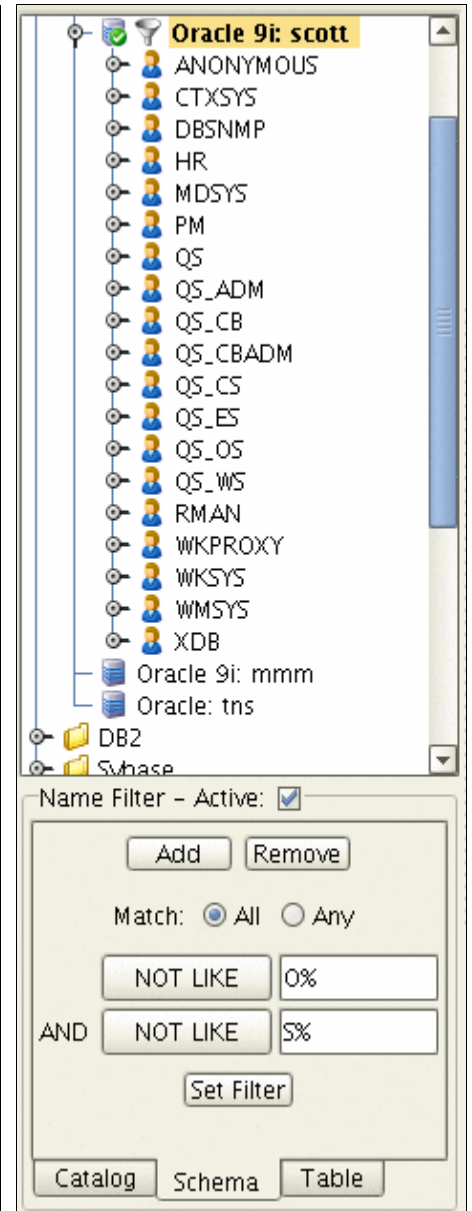
The unfiltered schema objects for an Oracle connection.



The same objects but now filtered based on all schema names starting with "O" or "S".



Filter defined as all names that do not start with "O" and "S".



**Figure: Examples of tree filter settings**

An active filter for a database connection is represented by the funnel icon just before the database connection name. The active state for a filter is defined using the **Active** box in the name filter pane. A filter can only be activated if there are any filters defined.

Up to 5 filters can be defined per catalog, schema or the table objects. Removing a filter definition always removes the last definition in the list.

**Tip:** It is often desired to list only the default schema or catalog (database) in the database objects tree. This can be accomplished using the filtering functionality but the recommended place to do this is in the properties tab for the database connection. Please read more about the **Show only default Database or Schema** in [Tool Properties](#) document.

## Show Table Row Count

The **Show Table Row** Count setting below the database objects tree defines whether the number of rows for table objects will be listed after the name of the table.

**Note:** Enabling this property results in a performance degradation.

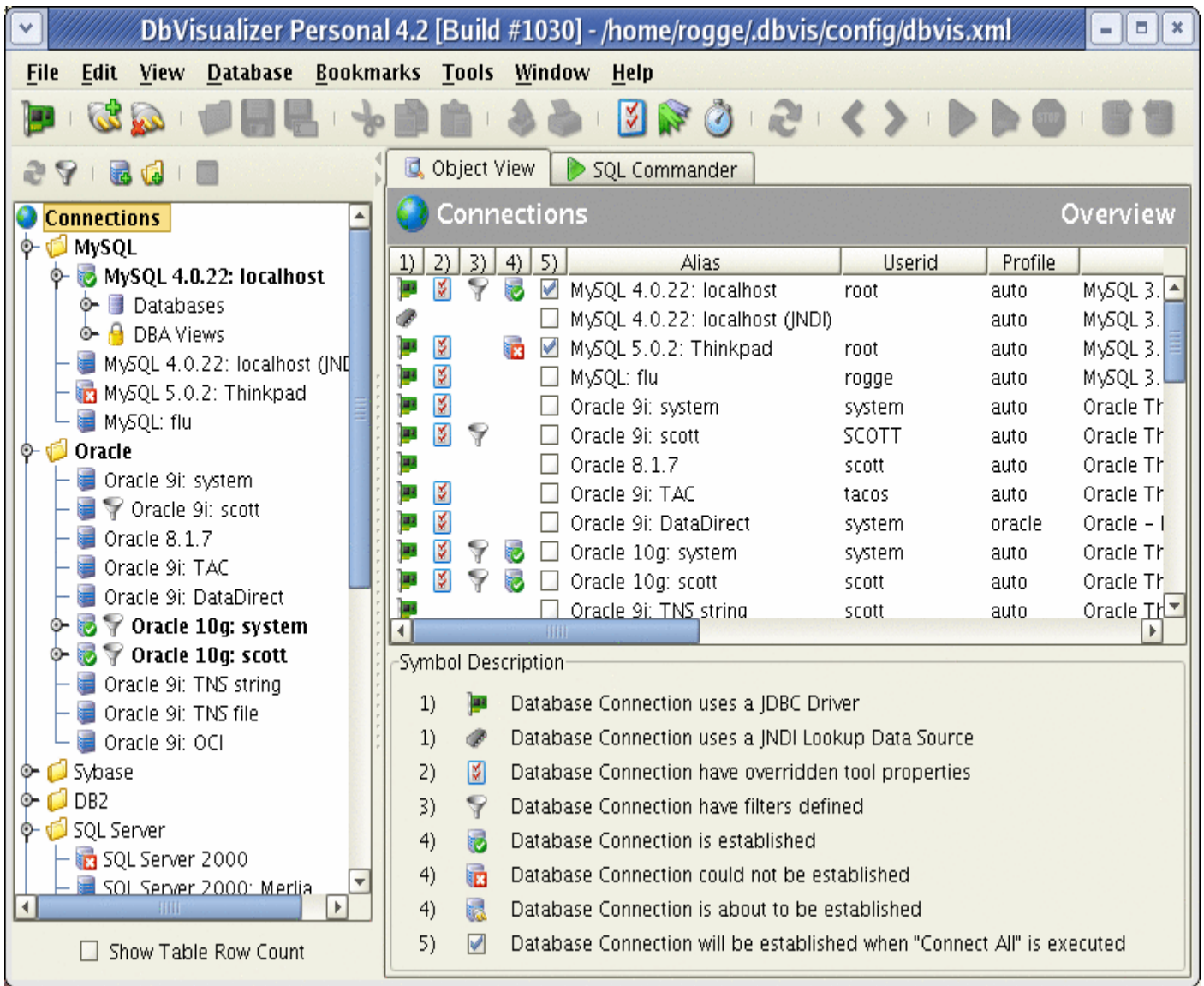
## Standard Tree Objects

There are a few objects that always appear in the tree independent of what edition of DbVisualizer and database profile that is used. The most important object is the **Database Connection**. It is used to setup and establish a database connection. The following sections explain these standard tree objects.

### Connections object

The **Connections** object is the root object in the tree and acts as a holder for all database connections and folders. The purpose of this is that when selected it displays an overview of all database connections in the details view. Here you can see the basic settings and states for your database connections. Read more about it in the [Load JDBC Driver and Get Connected](#) document.

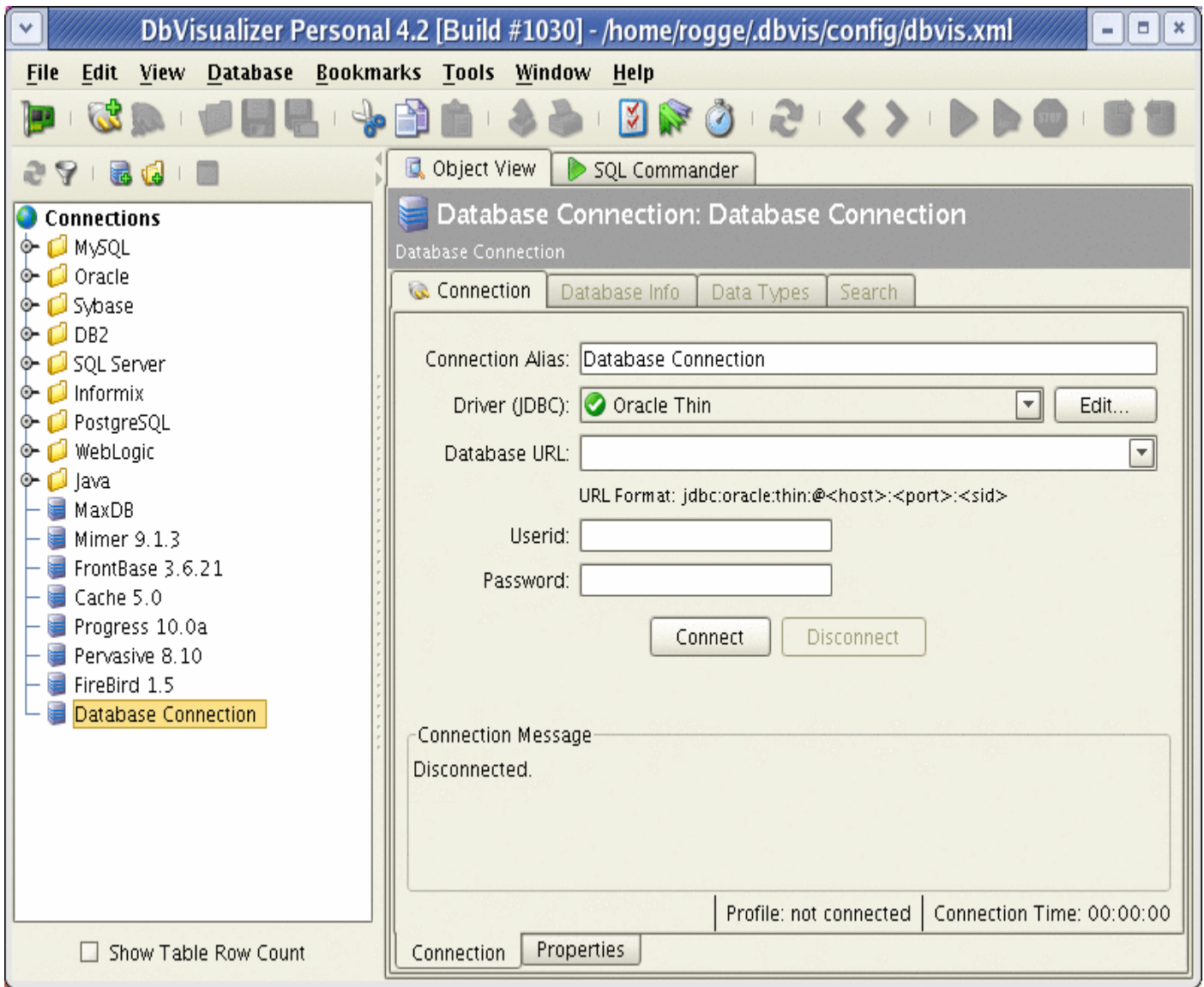




**Figure: Connections object**

### Database Connection object

The **Database Connection** object is the root object for a connection. Before exploring or accessing a database you need to establish the connection. Create a new database connection using the **Database->Add Database Connection** main menu choice and the following appear.



**Figure: The Database Objects Connections object**

(Detailed information on how to establish a connection is provided in the [Load JDBC Driver and Get Connected](#) document).

**Tip 1:** Once a database connection has been setup properly then you just need to double click on the object to establish the connection.

**Tip 2:** The **Database->Connect All** main menu choice is used to connect all enabled database connections with a single click. You make a database connection "Connect All" -aware in the **Properties** for the Database Connection or in the **Connections** overview.

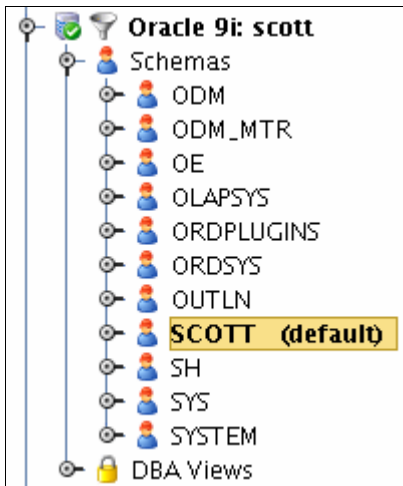
### Connection Alias

The name of the database connection object as it appears in the tree is by default the URL of the connection. The **Connection Alias** can be used to override this name to something more descriptive and shorter. Either enter the new name in the Alias field in the Connection sub tab or click on the name in the tree and start editing the name.

### Default databases and schemas

The **(default)** indicator in the name of a database or schema in the tree indicates that it is the default database or schema. The default is determined by whether the database was supplied in the URL

during connect. A default schema is the same as the schema in Oracle that the user logged in as.



**Figure: The (default) indicator for catalog (aka database) and schema objects**

**Tip:** The Properties sub tab in the connection settings can be used to specify that only default databases or schemas will be visible in the tree.

### Remove and copy database connection objects

To remove a database connection then select the **Database->Remove Database Connection** operation in the main menu. To copy a database connection select **Database->Duplicate Database Connection**.

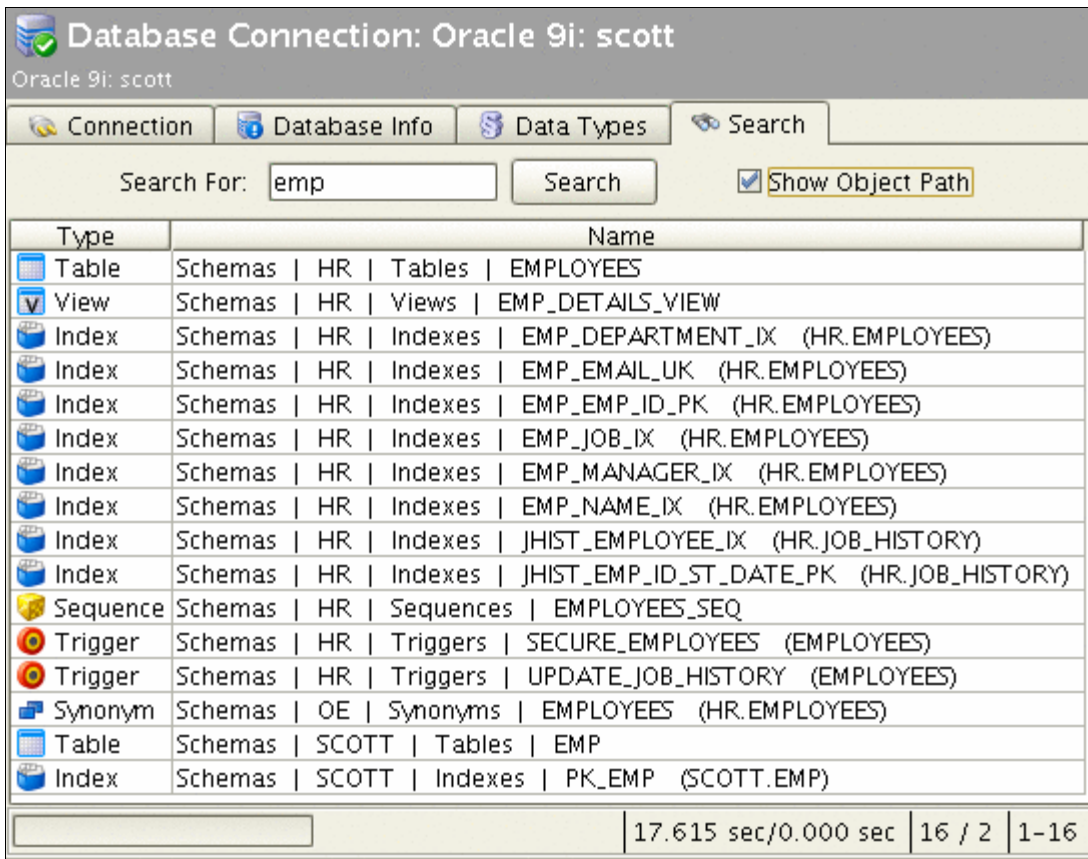
### Database Connection details tabs

The following section briefly explains the tabs in the objects view for a database connection.

Tab	Description
<b>Connection</b>	This tab is always enabled and is used to setup the details for a database connection. This is also the place to control the connection state.
<b>Database Info</b>	When connected, the database info tab shows various information supplied by the driver. Much of this info is low level even though some may be useful.
<b>Data Types</b>	The data types tab lists all data types supported by the database.
<b>Search</b>	The search tab is used to search among the objects in the tree. Search operates on the content in the tree based on if there are any filter defined or if any other setting has been set that effects the content of the tree. See next section for more information about search.

### Search

The Search tab is used to search among the objects in the tree by object name. This result will depend on if there are any tree filter defined or if any other property has been set that affects the content of the tree. The search operation is case insensitive.



**Figure: The Search tab**

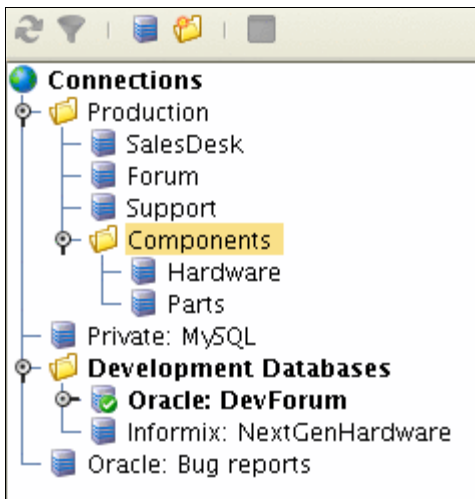
Search by specifying the name of the object or part of the name and press the **Search** button. The search operation can be stopped using the standard **Stop** button in the main tool bar. The **Show Object Path** check box is used to define whether the complete path for each found object should be displayed in the result or not. This path is the same as if navigating to each object manually in the objects tree.

**Note:** The search may take some time to perform the first time since all objects that are defined in the actual database profile are examined.

**Tip:** Detailed information of a specific object can be examined by double clicking on a row. This will display all information about the object in a separate window.

## Folder object

The folder object is used to organize and group database connections. It allows child folder objects in an unlimited hierarchy. You can either use the **View->Move Up/Down** main menu choices to organize the folders (and database connections) in the tree, or you can also use drag and drop to move things around.




**Figure: The database objects tree and the folder object type**


## Object View Types

The object views in the right area of the Database Objects tab shows detailed information about the selected tree object. The object view may contain several object view tabs depending on the current database profile. There are also several representations of a view to better illustrate the information. The following sections explains each of these visual presentation forms.

### Grid

The grid view is the most common one as it displays the data in a standard grid style.

 **Users**  
Oracle 9i: scott | DBA Views | Users

 **Users**

USERNAME	USER_ID	PASSWORD	ACCOUNT_STATUS	
ANONYMOUS	36	anonymous	EXPIRED & LOCKED	20 ▲
CTXSYS	33	71E687F036AD56E5	EXPIRED & LOCKED	20
DBSNMP	19	E066D214D5421CCC	OPEN	(nu
HR	46	6399F3B38EDF3288	EXPIRED & LOCKED	20
MDSYS	32	72979A94BAD2AF80	EXPIRED & LOCKED	20
ODM	42	C252E8FA117AF049	EXPIRED & LOCKED	20
ODM_MTR	43	A7A32CD03D3CE8D5	EXPIRED & LOCKED	20
OE	47	9C30855E7E0CB02D	EXPIRED & LOCKED	20
OLAPSYS	44	3FB8EF9DB538647C	EXPIRED & LOCKED	20
ORDPLUGINS	31	88A2B2C183431F00	EXPIRED & LOCKED	20
ORDSYS	30	7EFA02EC7EA6B86F	EXPIRED & LOCKED	20
OUTLN	11	4A3BA55E08595C81	EXPIRED & LOCKED	20
PM	48	72E382A52E89575A	EXPIRED & LOCKED	20
QS	52	8B09C6075BDF2DC4	EXPIRED & LOCKED	20
QS_ADM	51	991CDDAD5C5C32CA	EXPIRED & LOCKED	20
QS_CB	57	CF9CFACF5AE24964	EXPIRED & LOCKED	20
QS_CBADM	56	7C632AFB71F8D305	EXPIRED & LOCKED	20
QS_CS	58	91A00922D8C0F146	EXPIRED & LOCKED	20
QS_ES	54	E6A6FA4BB042E3C2	EXPIRED & LOCKED	20
QS_OS	55	FF09F3EB14AE5C26	EXPIRED & LOCKED	20
QS_WS	53	24ACF617DD7D8F2F	EXPIRED & LOCKED	20
RMAN	60	E7B5D92911C831E1	EXPIRED & LOCKED	20
SCOTT	59	F894844C34402B67	OPEN	(nu
SH	49	9793B3777CD3BD1A	EXPIRED & LOCKED	20
SYS	0	4DE42795E66117AE	OPEN	(nu ▼

0.036 sec/0.029 sec | 31 / 12 | 1-26

**Figure: The Grid view**

## Form

The form view extends the grid view by a form below the grid. Click on a row in the grid and the information is displayed in the form.

Table Spaces

Oracle 9i: scott | DBA Views | Table Spaces

Table Spaces Usage

TABLESPACE	DATAFILE	STATUS	TO
CWMLITE	C:\ORACLE\ORADATA\FUJI\CWMLITE01.DBF	AVAILABLE	
DRSYS	C:\ORACLE\ORADATA\FUJI\DRSYS01.DBF	AVAILABLE	
EXAMPLE	C:\ORACLE\ORADATA\FUJI\EXAMPLE01.DBF	AVAILABLE	
INDX	C:\ORACLE\ORADATA\FUJI\INDX01.DBF	AVAILABLE	
ODM	C:\ORACLE\ORADATA\FUJI\ODM01.DBF	AVAILABLE	
SYSTEM	C:\ORACLE\ORADATA\FUJI\SYSTEM01.DBF	AVAILABLE	
TOOLS	C:\ORACLE\ORADATA\FUJI\TOOLS01.DBF	AVAILABLE	
UNDOTBS1	C:\ORACLE\ORADATA\FUJI\UNDOTBS01.DBF	AVAILABLE	
USERS	C:\ORACLE\ORADATA\FUJI\USERS01.DBF	AVAILABLE	

TABLESPACE: EXAMPLE

DATAFILE: C:\ORACLE\ORADATA\FUJI\EXAMPLE01.DBF

STATUS: AVAILABLE

TOTAL (MB): 149

LEFT (MB): 0

USED (MB): 149

%-USED: 100

**Figure: The Form view**

If there is only one row in the result will no grid appear but only the form.

## Source

The source view is typically used to show the source for functions, procedures, triggers, etc. It is based on a read only editor with SQL syntax coloring. The sub tool bar buttons from the left:

- Export data to file
- Wrap long lines
- Copy the data to SQL Commander

```
1 PACKAGE sdo_cs AUTHID current_user AS
2
3 -- for TRANSFORM operator: trusted callout interface
4 FUNCTION transform(geom IN mdsys.sdo_geometry,
5     dim IN mdsys.sdo_dim_array,
6     to_srid IN NUMBER)
7     RETURN mdsys.sdo_geometry DETERMINISTIC;
8 PRAGMA restrict_references(transform, wnds, rnps, wnps);
9
10 FUNCTION transform(geom IN mdsys.sdo_geometry,
11     tolerance IN number,
12     to_srid IN NUMBER)
13     RETURN mdsys.sdo_geometry DETERMINISTIC;
14 PRAGMA restrict_references(transform, wnds, rnps, wnps);
15
16 FUNCTION transform(geom IN mdsys.sdo_geometry,
17     to_srid IN NUMBER)
18     RETURN mdsys.sdo_geometry DETERMINISTIC;
19 PRAGMA restrict_references(transform, wnds, rnps, wnps);
20
21 FUNCTION transform(geom IN mdsys.sdo_geometry,
22     dim IN mdsys.sdo_dim_array,
23     to_sname IN VARCHAR2)
24     RETURN mdsys.sdo_geometry DETERMINISTIC;
25 PRAGMA restrict_references(transform, wnds, rnps, wnps);
26
```

Figure: The Source view

## Table Row Count

The row count view is really simple as it only shows the number of rows in the selected object.



Table: NUMB  
Oracle 9i: scott | Schemas | SCOTT | Tables | NUMB

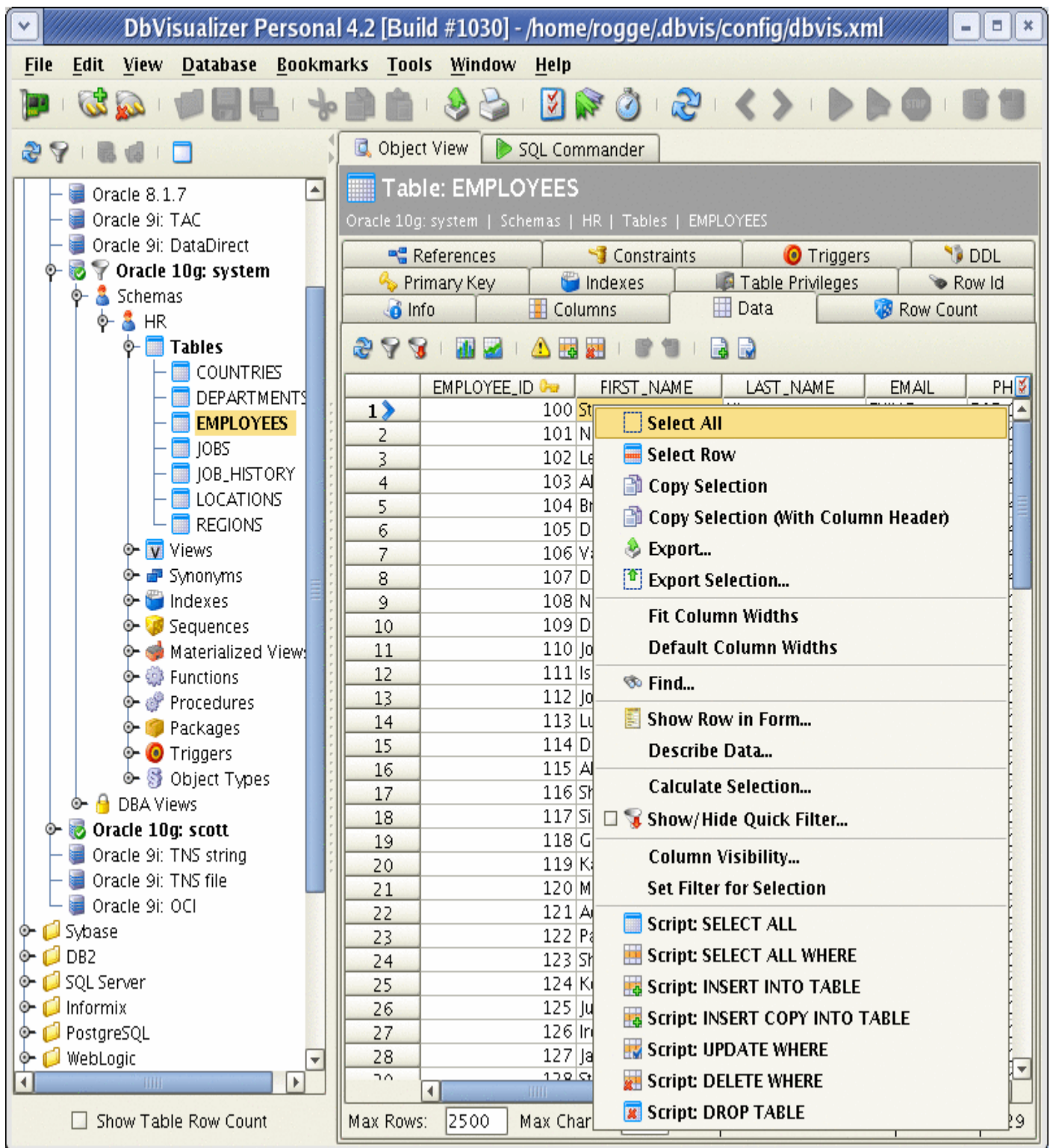
Table Privileges | Row Id | References | Triggers

Info | Columns | Data | Row Count | Primary Key | Indexes

Number of Rows: 18,307

## Table Data

The **Data** tab is used to browse the data in the table and to do various data related operations. This view is based on the [generic grid](#) but adds a few more visual components to limit the max number of rows, the width of text columns and the collection of data tab specific operations in the right click menu. In addition it is also possible to set a filter that will ensure that only the rows that match the filter will be displayed. The data tab is the place to do [edits](#) in DbVisualizer Personal.



**Figure: The Data tab for Table objects**

### Right click menu

The right click menu in the data tab grid menu adds some operations into the standard right click menu. These are primarily used to create SQL statements based on the current selection. Choosing any of these will create the appropriate SQL and then switch the view to the SQL Commander tab. These operations are used to edit table data in the DbVisualizer Free edition since the inline and form based editors are specifically for DbVisualizer Personal. (Information about the standard right click menu operations are available in the [Getting Started and General Overview](#) document).

The generated SQL can contain either static values as they appear in the grid or DbVisualizer **variables**. A variable is essentially used as a place holder for a value in an SQL statement. Once

the statement is executed DbVisualizer will locate all variables and present them in a dialog. The values for the variables can then be entered or modified and DbVisualizer will in the final SQL replace the variable place holders with the new values. Variables can be used in any SQL statement and DbVisualizer relies heavily on them. (Read more about variables in the [Executing SQL statements in the SQL Commander](#) document).

The use of variables in the SQL statements generated by the SQL operations in the right click menu depends on the **Table Data->Include Variables in SQL** setting in **Tool Properties**. This setting is by default true (include variables) and will result in variables being used in the statement. Disabling the property will result in static SQL in the generated statement.

Here follows an example with the **Include Variables in SQL** setting enabled and then disabled. The SQL is generated when the **select \* where** operation is selected based on the selection in the previous figure.

Include Variables in SQL is enabled
<pre>select * from SCOTT.EMP where ENAME = \$\$ENAME (where)  WARD  String  where ds=10 dt=VARCHAR nullable \$\$ and JOB = \$\$JOB (where)  SALESMAN  String  where ds=9 dt=VARCHAR nullable \$\$</pre>
Include Variables in SQL is disabled
<pre>select * from SCOTT.EMP where ENAME = 'WARD' and JOB = 'SALESMAN'</pre>

The following lists the generated SQL for each of the operations based on the selection of ENAME = WARD and JOB = SALESMAN.

Operation	SQL Example
<b>Set Filter for Selection</b>	<pre>ENAME = 'WARD' and JOB = 'SALESMAN'</pre>
<b>Script: SELECT ALL</b>	<pre>select * from SCOTT.EMP</pre>
<b>Script: SELECT WHERE</b>	<pre>select * from SCOTT.EMP where ENAME = 'WARD' and JOB = 'SALESMAN'</pre>
<b>Script: INSERT INTO TABLE</b>	<pre>insert into SCOTT.EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) values (, ' ', ' ', ' ', ' ', ' ', ' ', ' ')</pre>
<b>Script: INSERT COPY INTO TABLE</b>	<pre>insert into SCOTT.EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) values (7521, 'WARD', 'SALESMAN', 7698, '1981-02-22 00:00:00.0', 1250, 500, 30)</pre>
<b>Script: UPDATE WHERE</b>	<pre>update SCOTT.EMP set EMPNO = 7521, ENAME = 'WARD', JOB = 'SALESMAN', MGR = 7698, HIREDATE = '1981-02-22 00:00:00.0', SAL = 1250, COMM = 500, DEPTNO = 30 where ENAME = 'WARD' and JOB = 'SALESMAN'</pre>

<b>Script: DELETE WHERE</b>	delete from SCOTT.EMP where ENAME = 'WARD' and JOB = 'SALESMAN'
<b>Script: DROP TABLE</b>	drop table SCOTT.EMP

## Where Filter

The filter capability in the Data tab is used to form the where clause that will limit the number of rows in the grid.

**Figure: The Data tab filter**

The filter area is composed of two parts. The upper one is used to define the where clause for a single column. The available columns and operators are selected from two lists. The value of the column is specified in a text field. You can use **Ctrl-Enter** while editing the value to force a reload of the grid based on that single filter. The lower part displays the complete filter and the buttons are used to control whether the newly entered filter will be **AND**'ed or **OR**'ed with the complete filter. The buttons change appearance based on whether there is any filter or not. While in the complete filter you can use **Ctrl-Enter** to force a reload based on the complete filter. The right click menu lists the last 20 filters that have been applied to the grid.

**Figure: The filter history right click menu**

To reset the use of the filter select the **Reload** operation in the data tab tool bar.

(The visible state of the filter pane is controlled using the Filter toggle button in the data tab tool bar).

## Quick Filter

The quick filter acts on the data that is already in the grid as opposed of a **where filter** which is used to limit the number of rows fetched from the database. Quick filter is convenient as it is used to quickly list only those rows that match the entered search string.

The following figure shows data that matches the search string "d". Matching cells are highlighted.

Quick Filter

Show Rows Containing:     Instant Filtering

	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	
1	Steven	King	SKING	515.123.4567	1
2	Neena	Kochhar	NKOCHHAR	515.123.4568	1
3	Lex	<b>De Haan</b>	<b>LDEHAAN</b>	515.123.4569	1
4	<b>Alexander</b>	<b>Hunold</b>	<b>AHUNOLD</b>	590.423.4567	1
5	<b>David</b>	Austin	<b>DAUSTIN</b>	590.423.4569	1
6	<b>Diana</b>	Lorentz	<b>DLORENTZ</b>	590.423.5567	1
7	<b>Daniel</b>	Faviet	<b>DFAVIET</b>	515.124.4169	1
8	<b>Den</b>	Raphaely	<b>DRAPHEAL</b>	515.127.4561	1
9	<b>Alexander</b>	Khoo	AKHOO	515.127.4562	1
10	Shelli	<b>Baida</b>	<b>SBAIDA</b>	515.127.4563	1
11	<b>Adam</b>	Fripp	AFRIPP	650.123.2234	1
12	James	<b>Landry</b>	<b>JLANDRY</b>	650.124.1334	1
13	Renske	<b>Ladwig</b>	<b>RLADWIG</b>	650.121.1234	1
14	Curtis	<b>Davies</b>	<b>CDAVIES</b>	650.121.2994	1
15	<b>Randall</b>	Matos	RMATOS	650.121.2874	1
16	<b>Gerald</b>	Cambraut	GCAMBRAU	011.44.1344.619268	1
17	<b>David</b>	Bernstein	<b>DBERNSTE</b>	011.44.1344.345268	1
18	<b>Lindsey</b>	Smith	LSMITH	011.44.1345.729268	1
19	Louise	<b>Doran</b>	<b>LDORAN</b>	011.44.1345.629268	1
20	<b>Danielle</b>	Greene	<b>DGREENE</b>	011.44.1346.229268	1
21	<b>David</b>	Lee	<b>DLEE</b>	011.44.1346.529268	2
22	<b>Sundar</b>	<b>Ande</b>	<b>SANDE</b>	011.44.1346.629268	2
23	Amit	<b>Banda</b>	<b>ABANDA</b>	011.44.1346.729268	2
24	<b>Sundita</b>	Kumar	SKUMAR	011.44.1343.329268	2
25	<b>Girard</b>	Geoni	GGEONI	650.507.9879	2

**Figure: The filter history right click menu**

Entering successive characters will narrow the result even further as in the following figure.

Quick Filter

Show Rows Containing:     Instant Filtering

	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	
1	<b>David</b>	Austin	DAUSTIN	590.423.4569	1
2	<b>David</b>	Bernstein	DBERNSTE	011.44.1344.345268	1
3	<b>David</b>	Lee	DLEE	011.44.1346.529268	2

**Figure: The filter history right click menu**

When the **Instant Filtering** control is enabled then is the grid filtered while entering new characters. Having a lot of rows in the grid may slow down the search if having Instant Filtering enabled. If it is disabled then you must press the **Filter** button in order to apply the filter.

**Monitor row count**

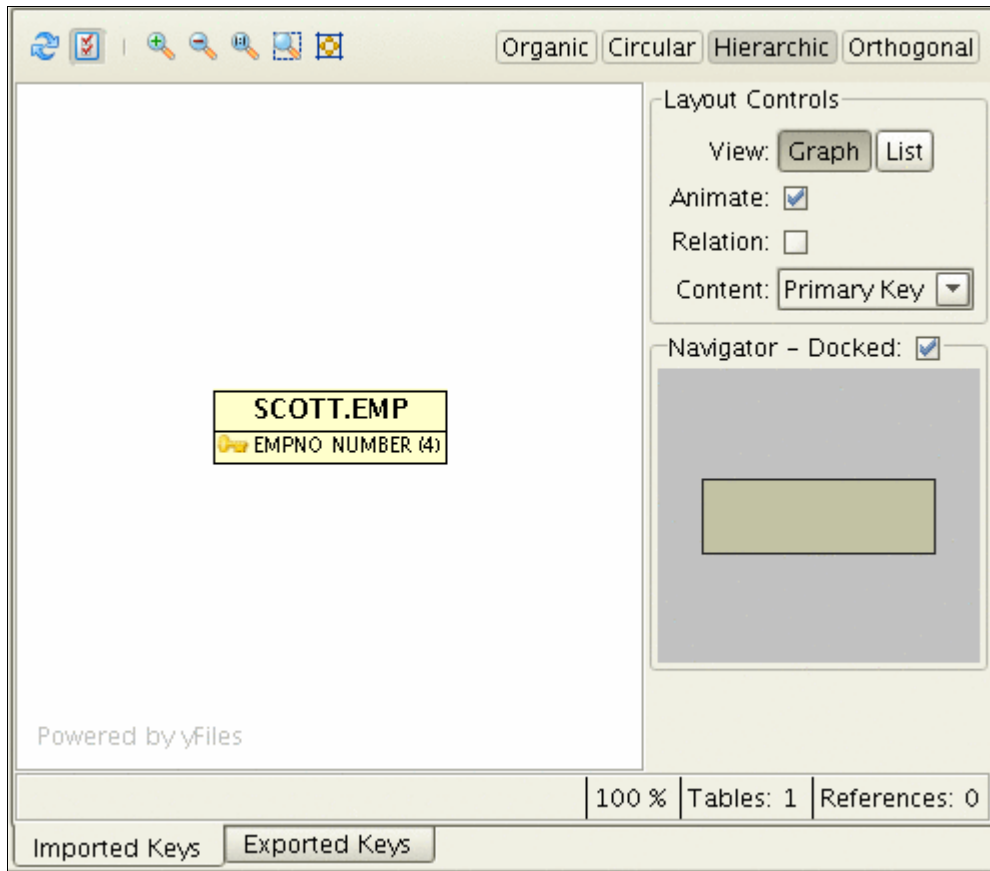
Read more about the **Monitor Row Count** and **Monitor Row Count Difference** in [Monitor and Charts](#).

**Editing**

Read about data editing in [Edit Table Data](#)

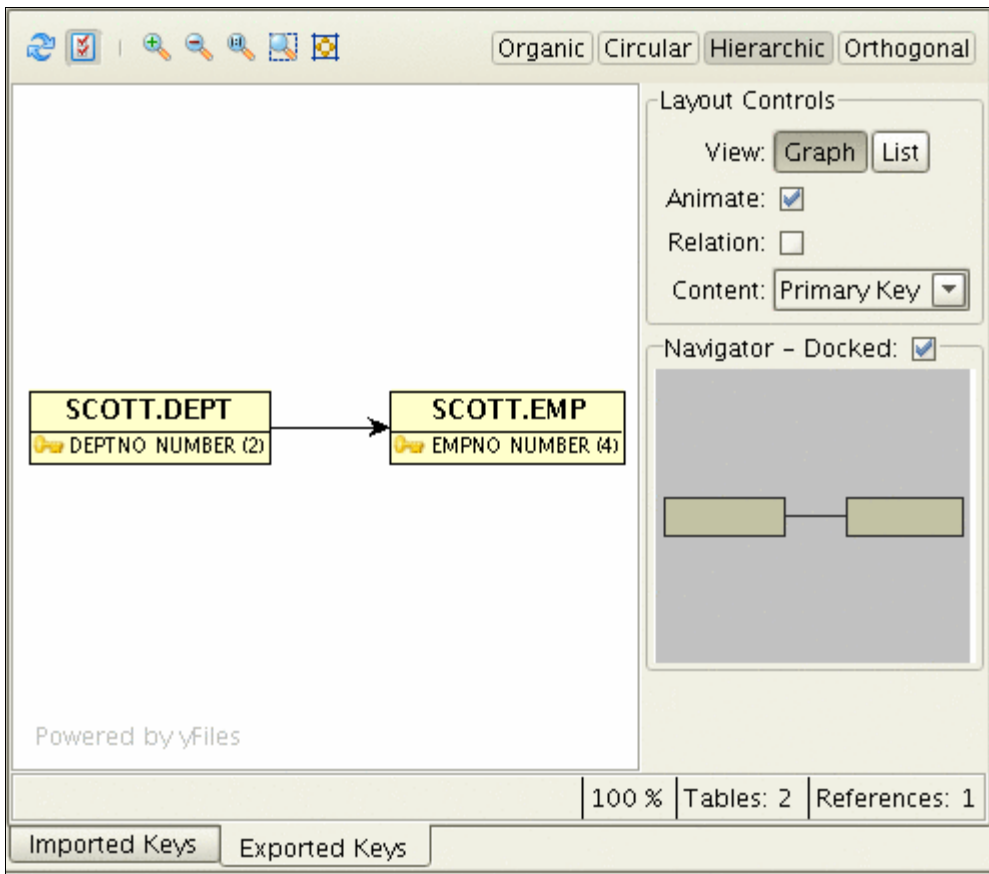
## References

The **references** tab is used to visualize the references from the table and what tables reference it. Use the sub tabs at the bottom of the display to show either view. The following shows the references from the table.



**Figure: The references graph showing imported keys for a table**

The following shows the references to the table.



**Figure: The references graph showing exported keys for a table**

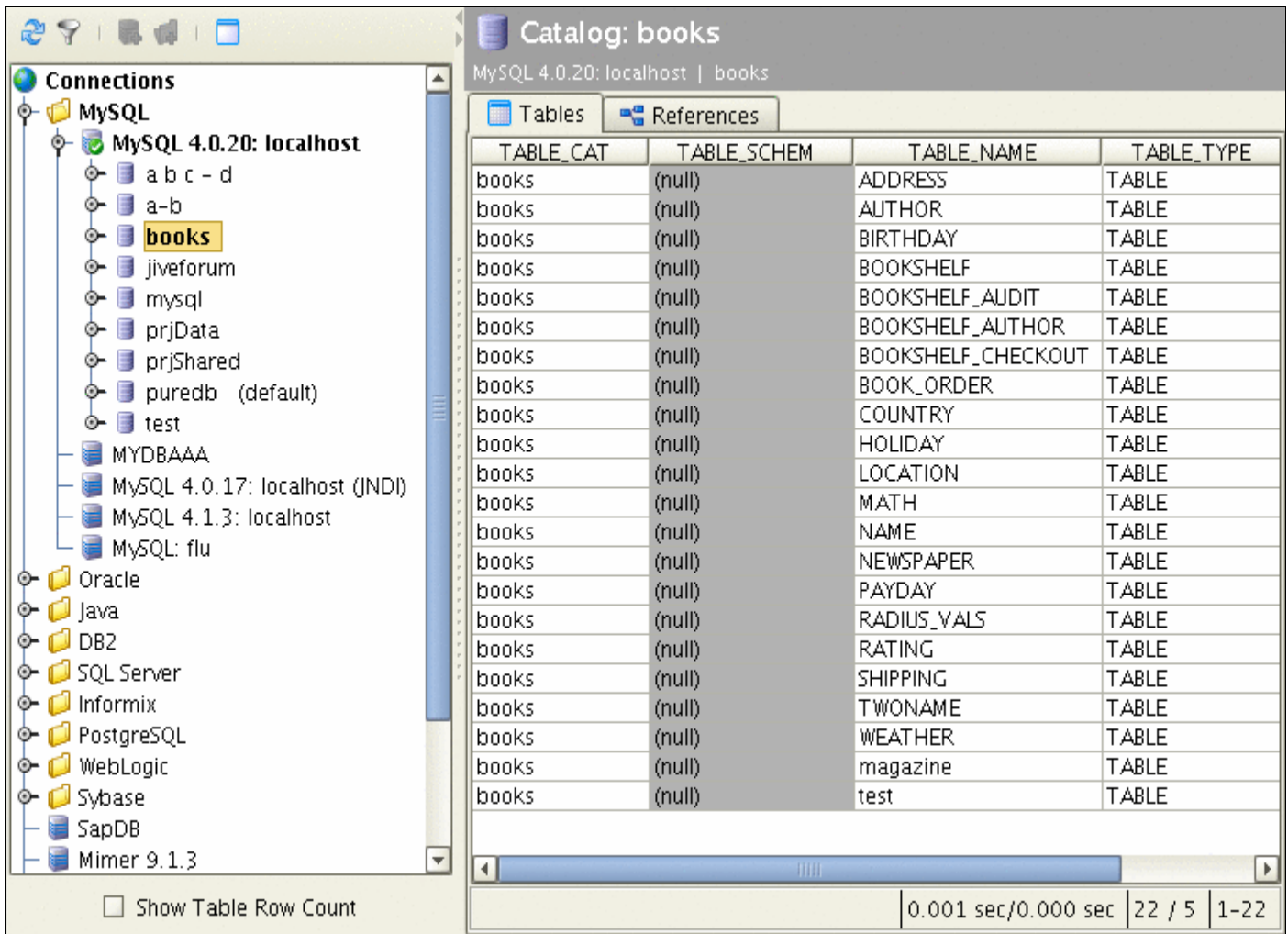
## Generic Database Profile

DbVisualizer supports a wide range of databases and since the nature of these and what they support is different from vendor to vendor so will the appearance and the structure of the tree below the database connection objects look different. The generic database profile and so DbVisualizer Free displays objects based on what JDBC offers in terms of database objects (aka meta data information). DbVisualizer does this by simply asking the actual JDBC driver for all schemas, catalogs, tables and procedures. It then builds the tree based on what it gets.

The advantage of using JDBC to get meta data about the database is that it's the responsibility of the driver to perform the operations in order to get the requested information. The drawback of letting the driver do this is that JDBC does not offer that much support for getting meta data information about the objects in a database i.e. the object types that are presented in the tree are sufficient for most database while there are obvious objects that are missing for some databases. The solution is simply to upgrade to the DbVisualizer Personal edition.

## Catalog (aka Database) object

The **Catalog** object is the generic JDBC term for a **Database** in for example Sybase, PostgreSQL, SQL Server and MySQL. It groups all objects for a logical database. The object view for a catalog is a pane with two tabs, **Tables** and **References**. The tables tab lists all the tables that are located in the catalog while references shows the exact same list of tables but instead as a referential integrity graph.



**Figure: The view for Catalog objects**

The child objects shown for a catalog depends on the capabilities of the JDBC driver. Normally you will see a list of the supported table types that groups the tables of these types. The number within parentheses is the number of tables. The example shows a MySQL database. The driver reports that it can handle the table types, **TABLE** and **LOCAL TEMPORARY**. (These table types are the same as those listed in the **Table Types** tab when selecting a database connection object.)

**Tip 1:** You can double click on a catalog object to display the detail view in a separate window.

**Tip 2:** Select one or several rows (cells) in the tables grid and then choose **Database->Build Select Script** to create a select script for the selected tables.

## Schema object

The **Schema** object is organized in the same way as the Catalog objects. There is in fact no difference except that the schema objects are in another level in the tree and represented by another icon.

The following screen shot shows the information for the selected schema with the Reference tab selected.



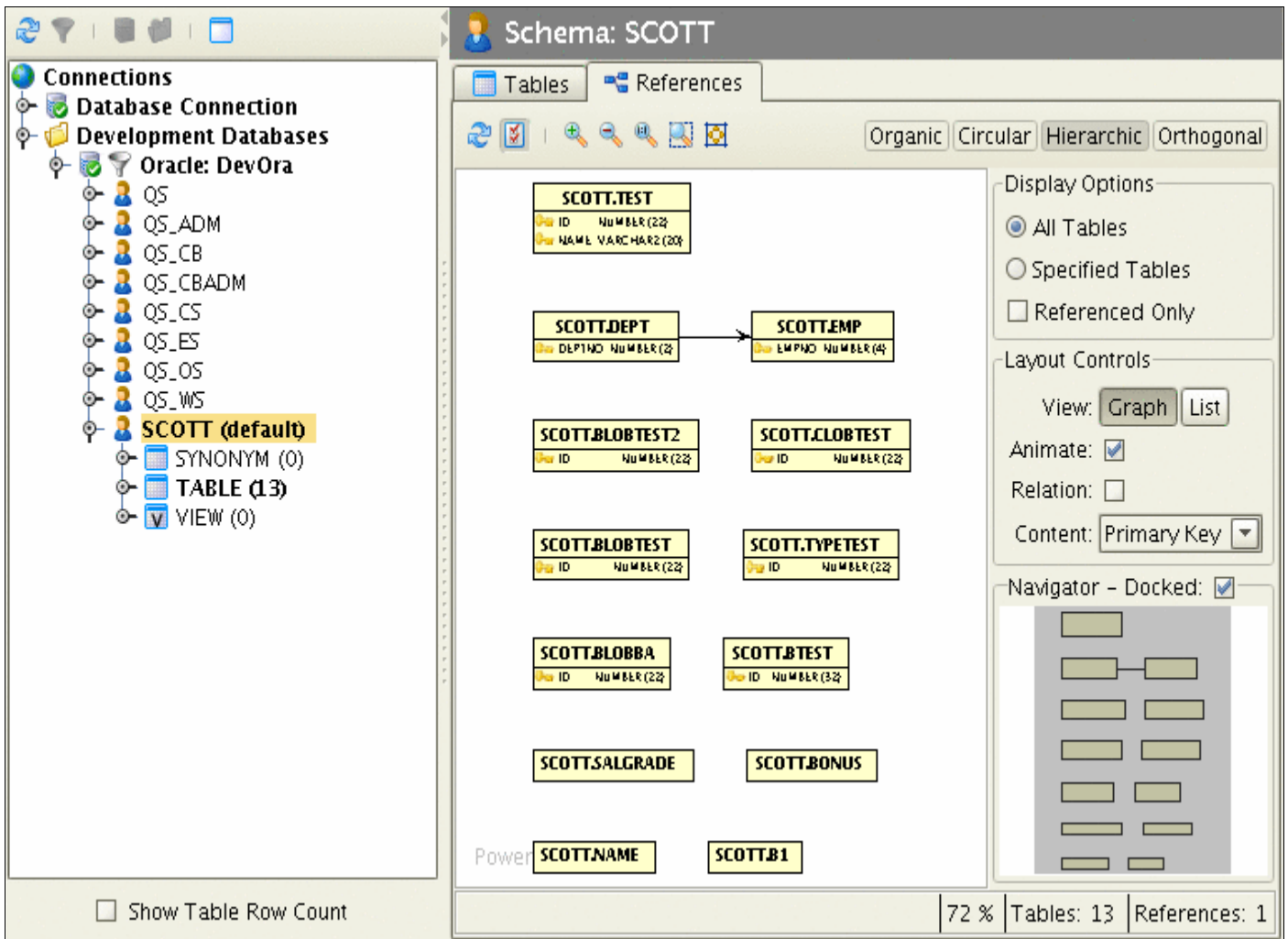


Figure: The view for Schema objects

### Table type object

The **Table Type** object has been briefly explained earlier. The name and the number of table type objects are determined by the driver as DbVisualizer asks for the supported table types. When DbVisualizer retrieves all tables it checks each table's type and puts them into the matching table type object. The reason is simply to make the tree easier to browse.

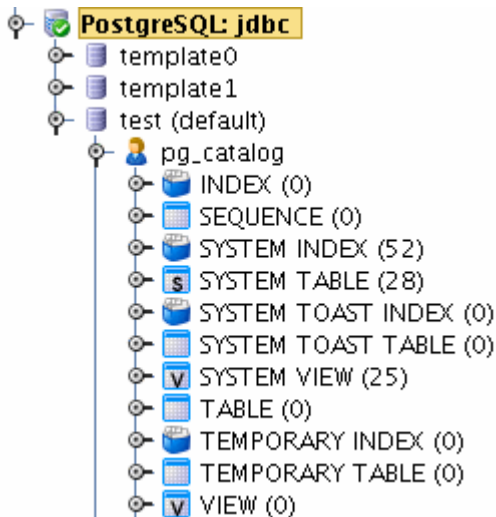


Figure: Example of table type objects for PostgreSQL

Note: Even though the figure above lists objects as INDEX, SEQUENCE, VIEW, etc are all treated as tables by DbVisualizer.

## Table object

The **Table** object is probably the most frequently accessed object in the tree as when selected it shows not only a lot of information about the table but also the data in it. This is also the place where data edits are performed.

The screenshot displays the DbVisualizer Personal 4.2 interface. The left pane shows a tree view of database connections, with 'Oracle 10g: system' selected and expanded to show the 'EMPLOYEES' table. The main pane shows the 'Table: EMPLOYEES' view with a data grid. The data grid contains the following information:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE
1	100	Steven	King	SKING	515.121.2234
2	101	Neena	Kochhar	NKOCHHAR	515.121.2234
3	102	Lex	De Haan	LDEHAAN	515.121.2234
4	103	Alexander	Hunold	AHUNOLD	590.421.2234
5	104	Bruce	Ernst	BERNST	590.421.2234
6	105	David	Austin	DAUSTIN	590.421.2234
7	106	Valli	Pataballa	VPATABAL	590.421.2234
8	107	Diana	Lorentz	DLORENTZ	590.421.2234
9	108	Nancy	Greenberg	NGREENBE	515.121.2234
10	109	Daniel	Faviet	DFAVIET	515.121.2234
11	110	John	Chen	JCHEN	515.121.2234
12	111	Ismael	Sciarra	ISCIARRA	515.121.2234
13	112	Jose Manuel	Urman	JMURMAN	515.121.2234
14	113	Luis	Popp	LPOPP	515.121.2234
15	114	Den	Raphaely	DRAPHEAL	515.121.2234
16	115	Alexander	Khoo	AKHOO	515.121.2234
17	116	Shelli	Baida	SBAIDA	515.121.2234
18	117	Sigal	Tobias	STOBIAS	515.121.2234
19	118	Guy	Himuro	GHIMURO	515.121.2234
20	119	Karen	Colmenares	KCOLMENA	515.121.2234
21	120	Matthew	Weiss	MWEISS	650.121.2234
??	121	Adam	Frinn	ΔFRIPP	650.121.2234

The status bar at the bottom indicates: Max Rows: 2500, Max Chars: -1, 0.019 sec/0.075 sec, 107 / 11, 1-22.

Figure: The view for Table objects

The detailed view for table objects displays

Tab	Description
<b>Info</b>	Brief information about the table object
<b>Columns</b>	This tab lists type information about all columns in the table
<b>Data</b>	Read more in <a href="#">Data tab</a>
<b>Row Count</b>	Lists the table row count
<b>Primary Key</b>	Shows the primary key
Indexes	Lists all indexes for the table
Table Privileges	Displays any privileges for the table
Row Id	Displays the optimal set of columns that uniquely identifies a row
<b>References</b>	Read more in <a href="#">References tab</a>

### Data tab

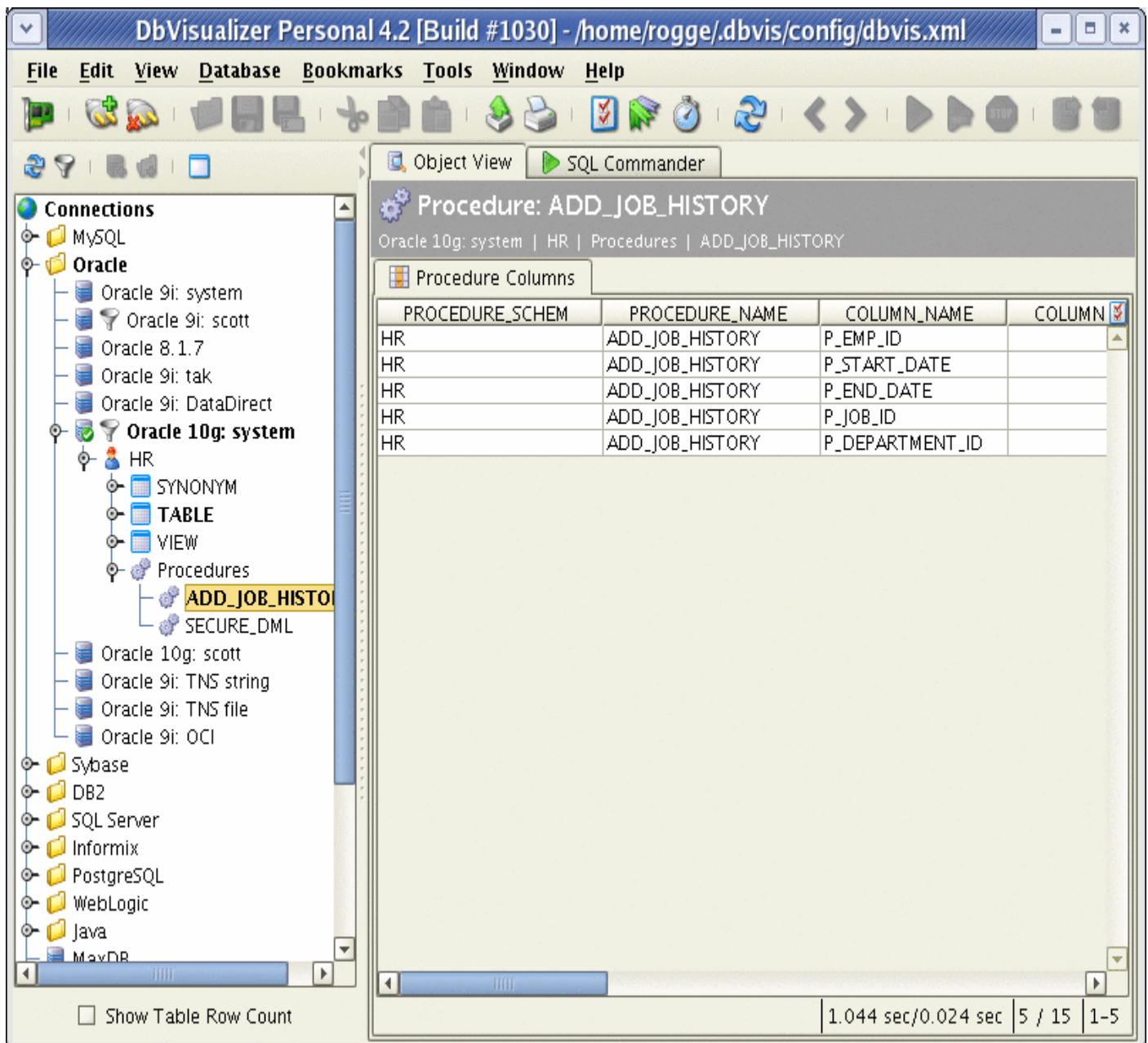
Read more about the Data tab in the [Table Data](#) section.

### References tab

Read more about the References tab in the [References](#) section.

### Procedure object

The procedure object is probably the simplest since it shows the name of the procedure or function in the tree, and in the object view lists the parameters that are used when calling it.



**Figure: The procedure object**

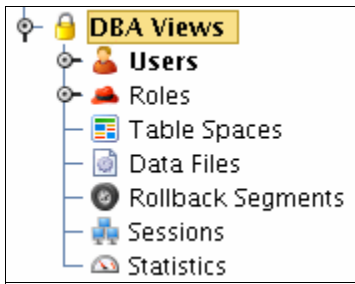
The object view shows a list of column names for the selected procedure.

## Specialized Database Profiles

[Specialized database profiles](#) are available for Oracle, Sybase, SQL Server, Informix, DB2, MySQL and PostgreSQL. Since each of these databases supports different types will the database objects tree look different. The structure and organization of a database profile is also something that may impact the layout of the tree even though the provided ones are similar in their structure. There are two groups of objects in the the tree:

- User objects
- DBA objects

User objects are for example, tables, views, triggers, functions, etc. while DBA objects most probably requires certain privileges in the database in order to access them. DbVisualizer organizes all DBA objects in the **DBA Views** tree object. If privileges are not sufficient to access a DBA object may this result in an error.



**Figure: The DBA Views tree object**

**Note:** Database profiles are defined in XML and it is quite easy to extend and modify them. It is not recommended to modify database profiles currently as the format may change.

# Executing SQL statements in the SQL Commander

## Introduction

The SQL Commander is used to edit and execute SQL statements or SQL scripts (several SQL statements in one batch). The result is either displayed in grids or log entries depending on what result is returned from the execution. There is support for multiple editors, setting of font, key bindings, auto completion, management of multiple result sets and a lot more. The SQL Commander tab is organized as follows:

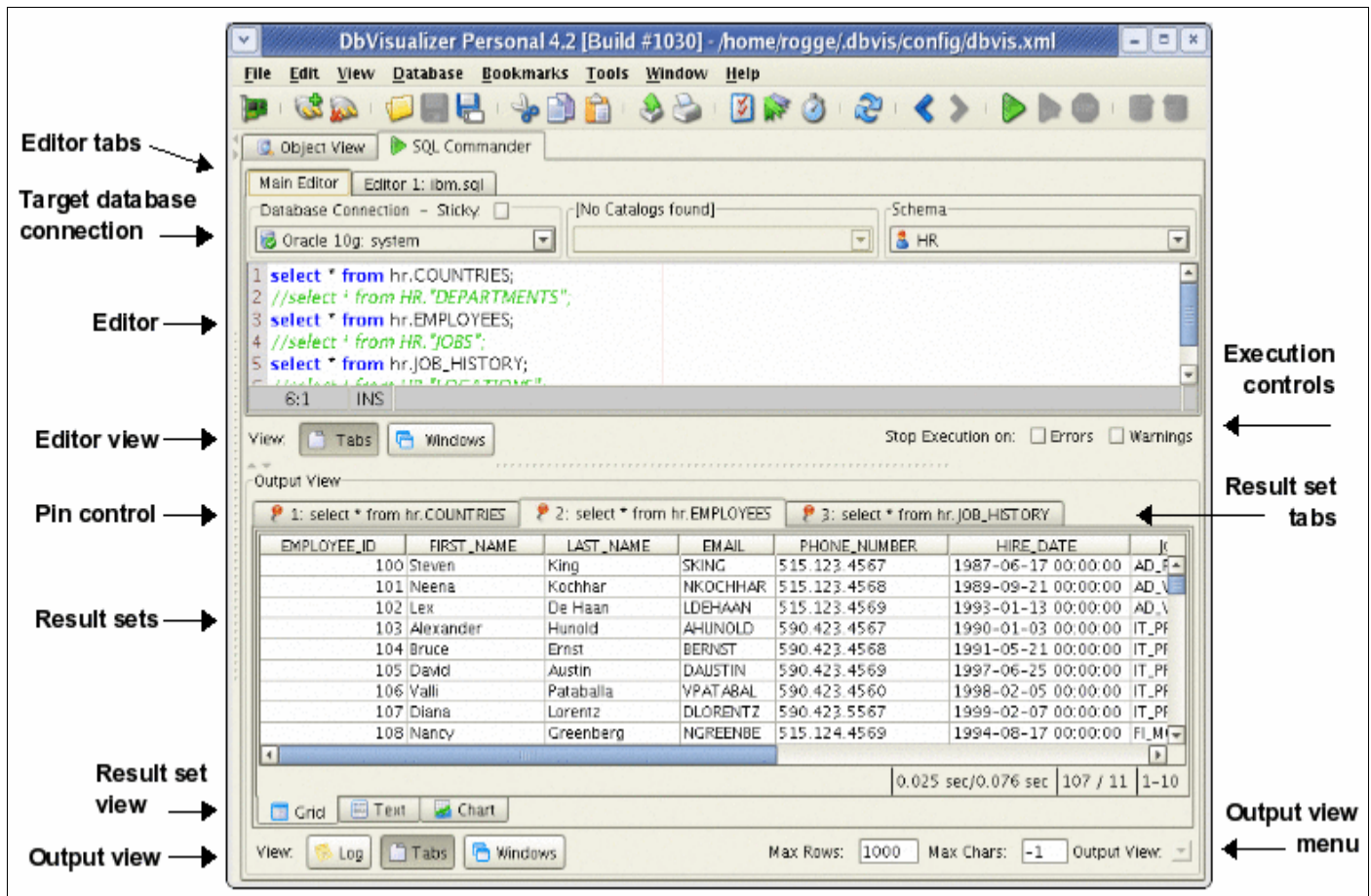


Figure: SQL Commander overview

The figure shows the editing area and controls above and the output view in the lower part of the screen. The following sections give a detailed explanation of all features and controls in the SQL Commander.

## Editor

The SQL editor in DbVisualizer is based on the [NetBeans](#) editor module and supports all standard editing features. The editor supports keyword coloring, auto completion and the key bindings can be customized in Tool Properties.

The right click menu contains the following operations:

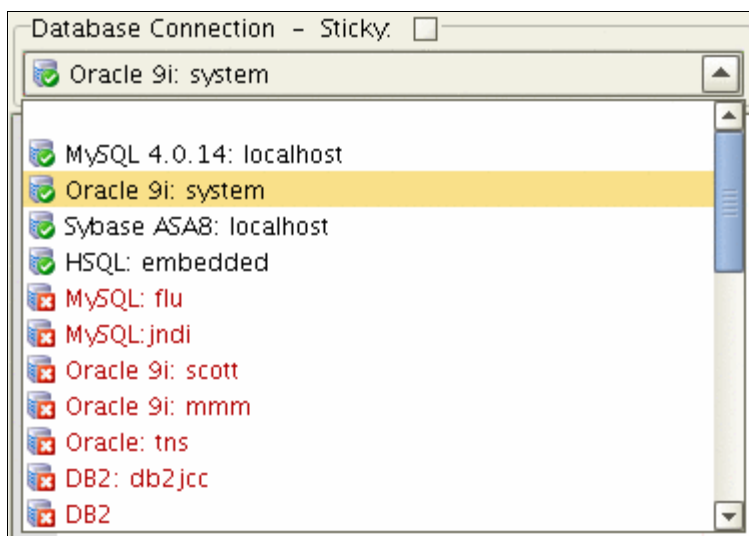
<b>Undo</b>	Undo
<b>Redo</b>	Ctrl-Y
<b>Cut</b>	Ctrl-X
<b>Copy</b>	Ctrl-Insert
<b>Paste</b>	Ctrl-V
<b>Clear</b>	
<b>Find</b>	Find
<b>Find Next Occurrence</b>	F3
<b>Find Previous Occurrence</b>	Shift-F3
<b>Replace</b>	Ctrl-H
<b>Go to Line</b>	Ctrl-G

**Figure: The SQL editor right click menu**

The SQL editor is also used in the Bookmark Editor and when editing CLOB's in the form editor.

## Database Connection, Catalog and Schema

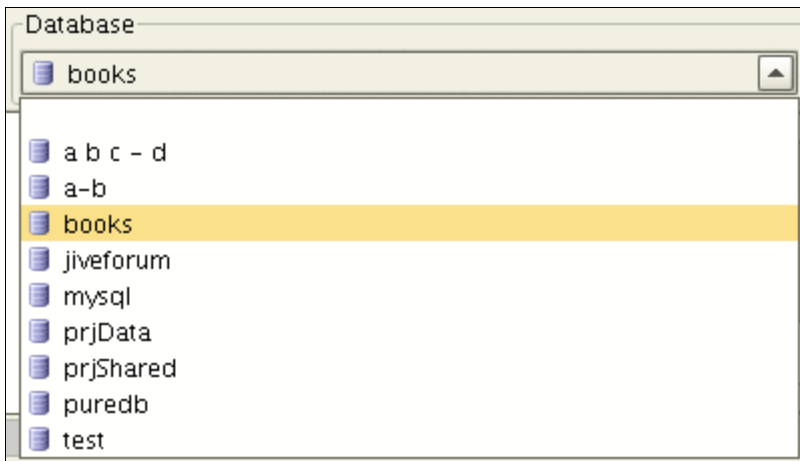
The Database Connection and Catalog (aka Database) boxes above the editor specify which connection and database the SQL in the editor will be executed by. The list of connections shows all connections as they are ordered in the Database Objects tree with the exception that all currently active connections are listed first.



**Figure: The Database Connection box**

The **Sticky** box above the list specifies when enabled that the current connection selection will not change automatically when passing SQL statements from other parts of DbVisualizer. One example is passing an SQL bookmark from the Bookmark Editor. Consider an SQL bookmark defined for database connection "**ProdDB**". If the Sticky setting is disabled the database connection will automatically be changed to ProdDB. If however the Sticky setting is enabled then the current setting of database connection will be unchanged. The Sticky setting is per SQL editor instance.

The Catalog box (more commonly known as Database) is used to set what catalog in the connection will be the target for the execution. In the event of catalogs not being supported by the database connection the header will indicate this with **No Catalogs for the Database Connection**.

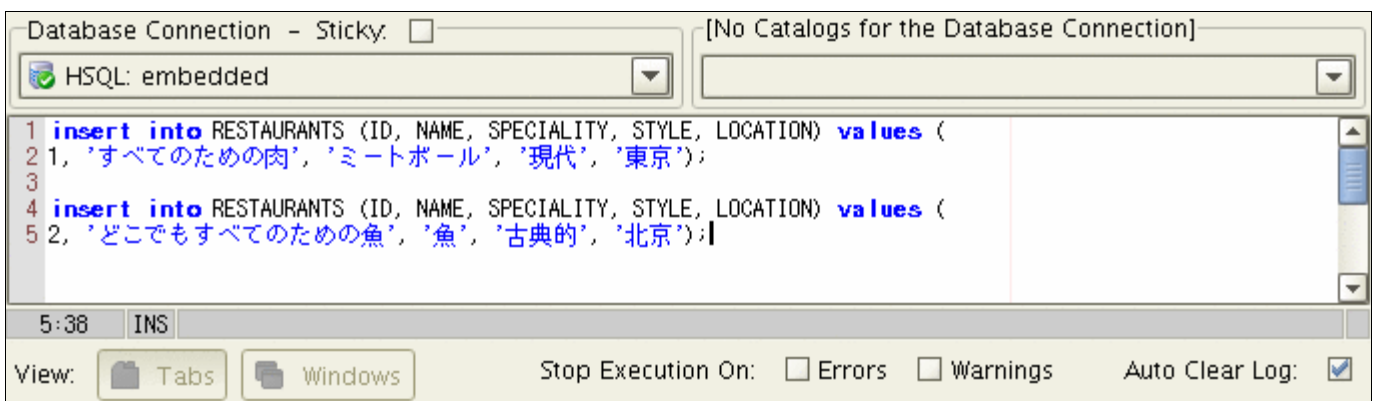


**Figure: The Catalog box**

The Schema box is used only to help the auto completion feature to limit what tables to show in the completion popup. It does not define that the actual SQL should be executed in the selected schema.

## Fonts

The SQL editor supports changing font which is useful and necessary in order to display characters for languages like Chinese, Japanese, etc.



**Figure: SQL Editor with another font**

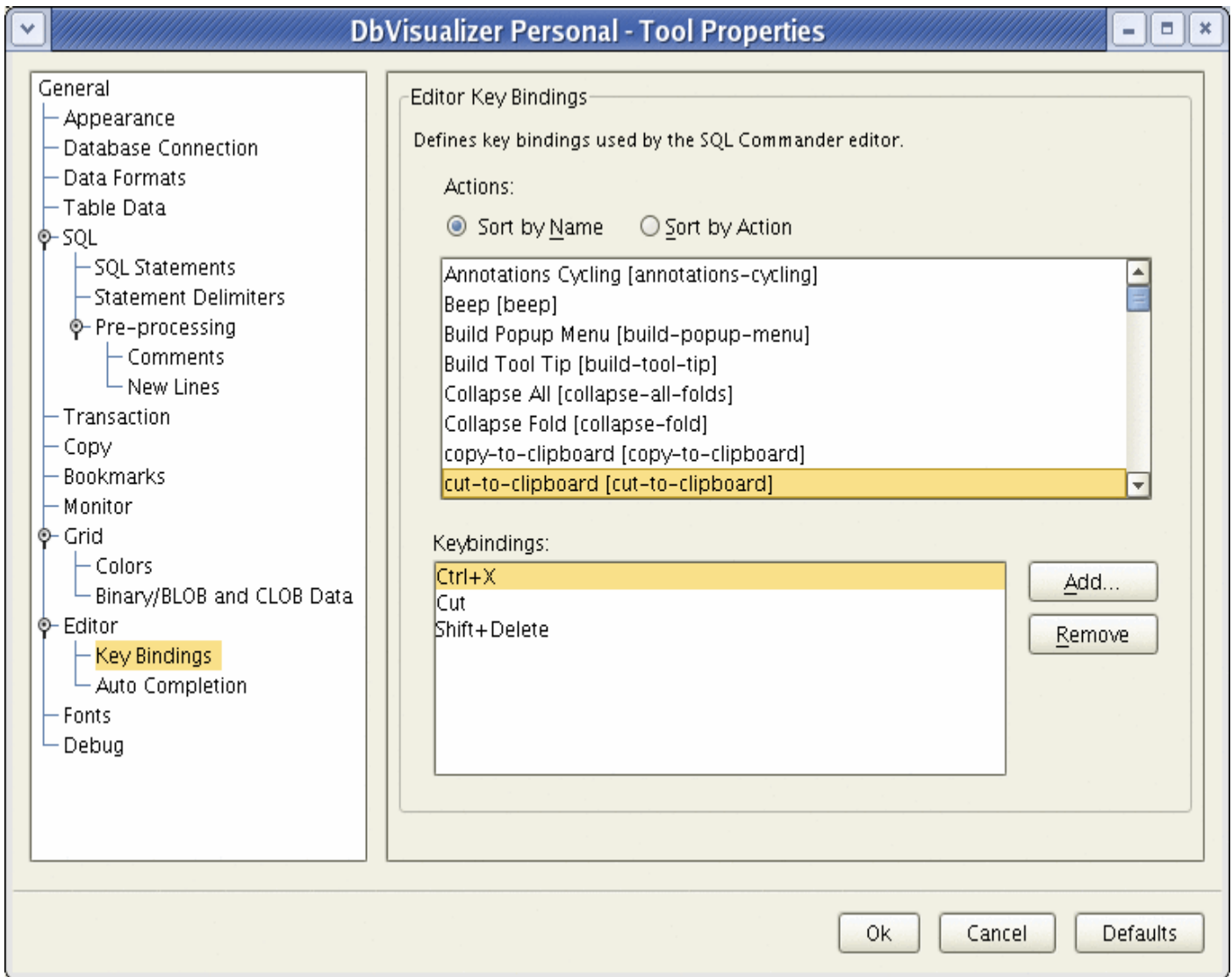
Open [Tool Properties](#) and select the Font category in order to set the font for the SQL Editor. (It is advisable to set the same font for both the SQL editor and the grid components).

**Note:** Displaying data correctly is not just a matter of setting the font. The reason is that the character encoding on the client side (in which DbVisualizer runs) and the database server may not be compatible. There is experimental support to set encodings to accomplish proper conversation between different encodings. Please see the [Getting Started and General Overview](#) document for more information.

## Editor shortcuts

The editor shortcuts or key bindings can be re-defined in the **Tool Properties->Editor** category. These settings are saved between invocations of DbVisualizer.

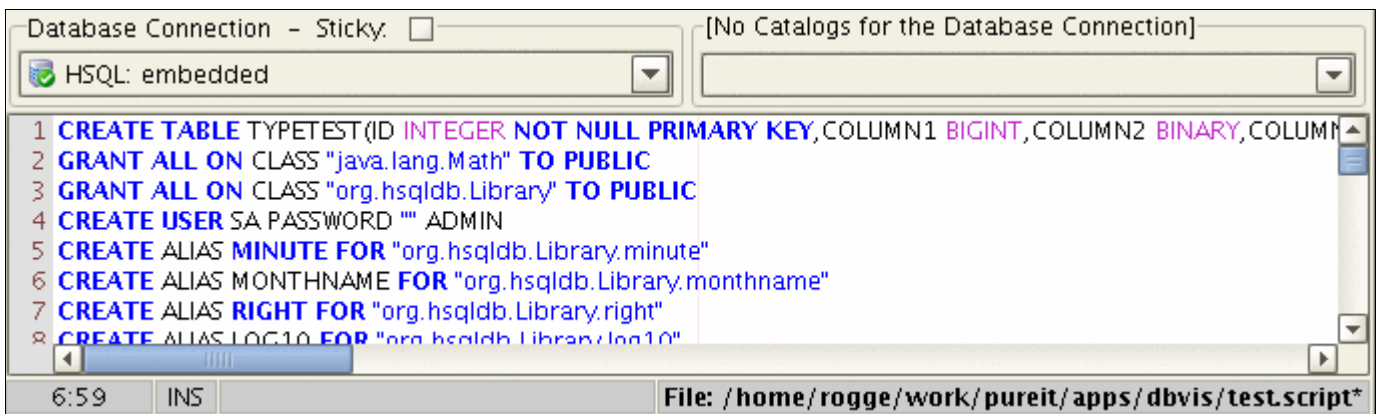




**Figure: The Editor category in Tool Properties used to re-define shortcuts**

### Load from and save to file

The SQL editor supports loading from file and saving to file. Use the standard file operations, **Load**, **Save** and **Save As** in the **File** main menu to accomplish this. Loading a file always loads into the currently selected editor.



**Figure: Loading a file into the SQL Commander**

The name of the loaded file is listed in the status bar of the editor. The editor tracks any modifications and indicates changes with an asterisk (\*) after the filename.

DbVisualizer will ask at exit if there are any pending edits that need to be saved.

## Comments in the SQL

Comments in the SQL editor are identified by the comment identifiers in Tool Properties. These are client side comments and are simply removed before execution.

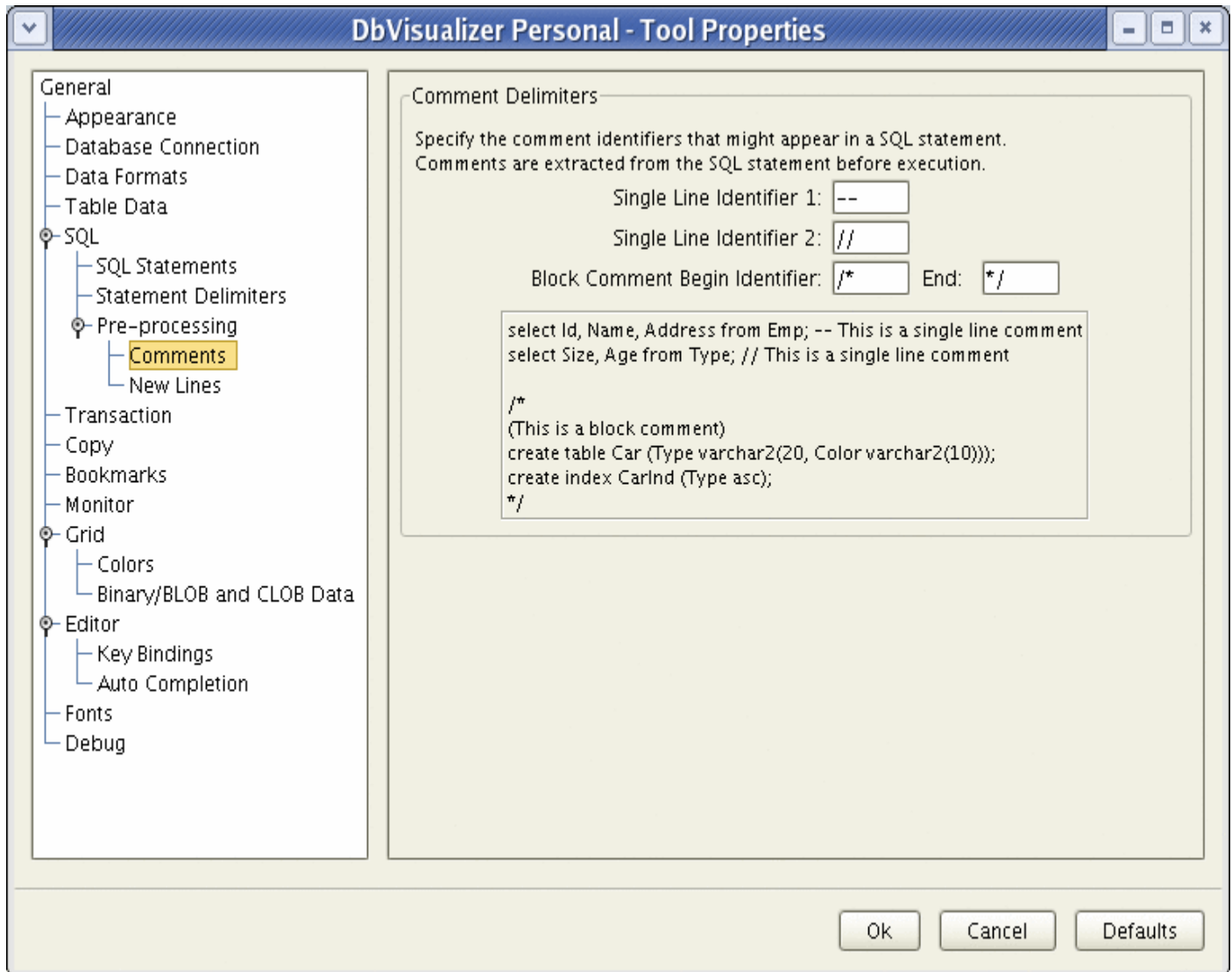


Figure: The Comments category in Tool Properties

## Multiple editors

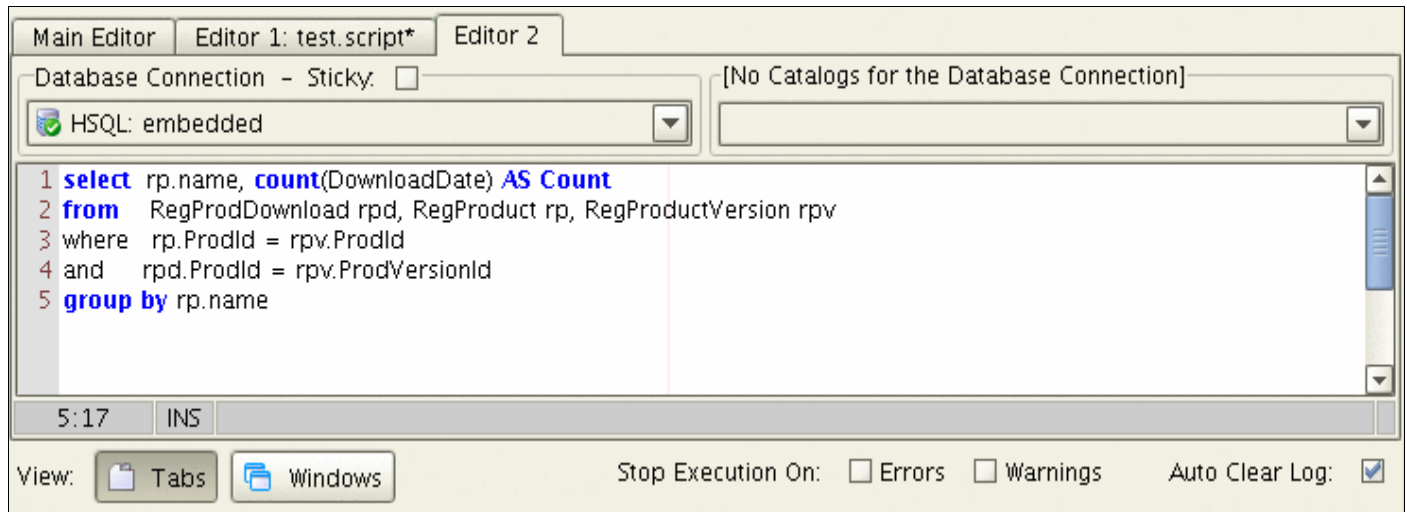
Multiple SQL editors can be created with the **File->Create SQL Editor** main menu operation. Editors can be organized as tabs or internal windows using the **View** buttons. There is always one default editor named **Main Editor**. This editor is used when passing SQL bookmarks from the Bookmarks Editor or when issuing requests from other parts of DbVisualizer that activate the SQL Commander. To remove all but the Main Editor select the **File->Close all SQL Editors** menu operation.

The following figures show 3 editors organized in the tabs style and the windows style

### Tabs style

The SQL editors in the figure below show the Main Editor, Editor 1 and Editor 2. A file has been loaded into Editor 1 and the label shows the file name and indicates with an asterisk if the content in

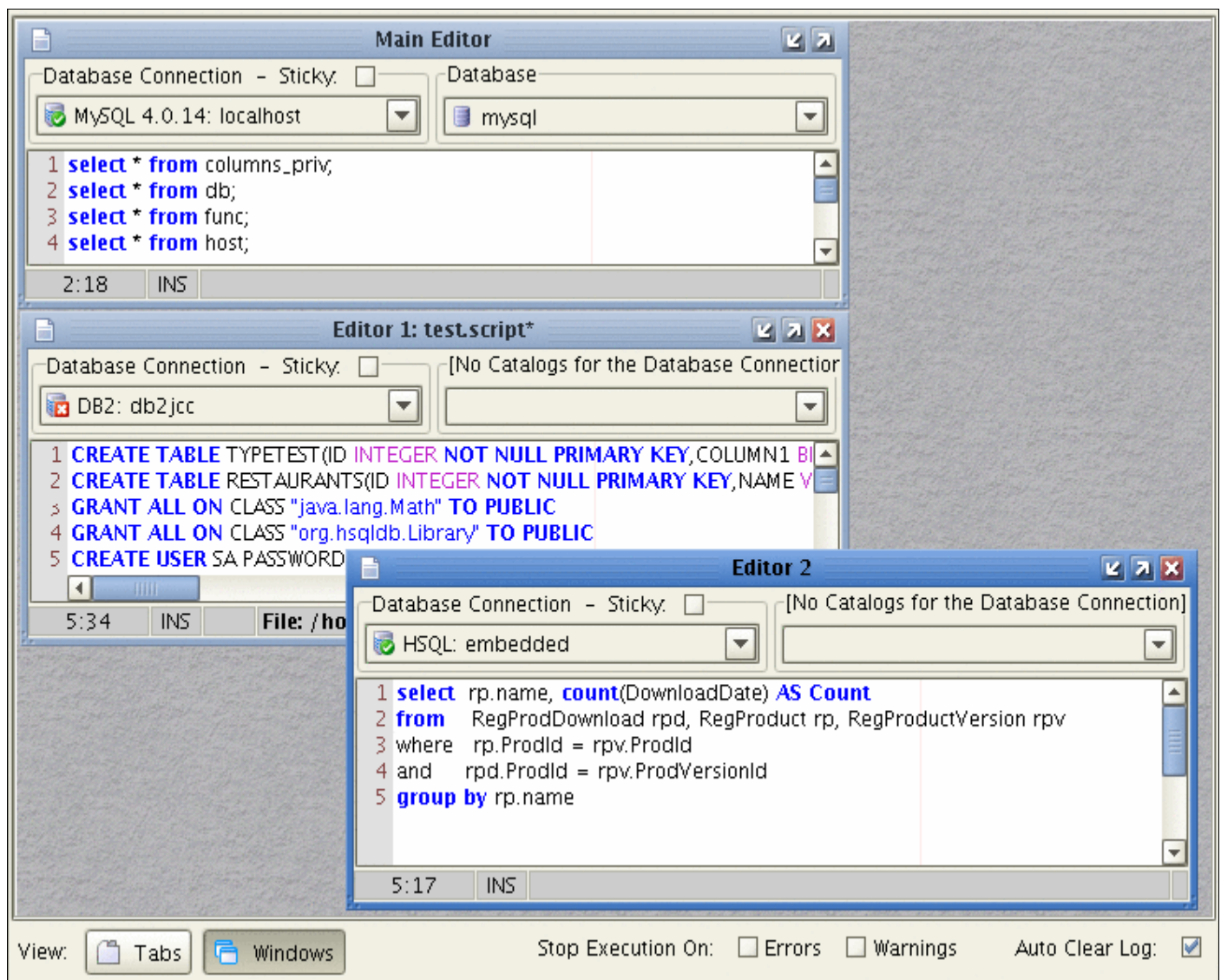
the editor has been modified. Remove an editor by choosing the **Close** operation in the right click menu while over the tab header.



**Figure: Multiple SQL editors in the Tabs view**

### Windows style

The following figure shows the same editors but in the Windows view.



## Figure: Multiple SQL editors in the Windows view

Remove an SQL editor window by selecting the close (red cross) button in the window header. Windows can be automatically organized using the **Tile** and **Cascade** operations in the **Window** main menu.

## Auto Completion

Auto completion is a convenient feature used to assist in the editing of SQL statements. The auto completion support in DbVisualizer currently supports completing table and columns names for the following DML commands:

- SELECT
- INSERT
- UPDATE
- DELETE

To display the completion popup then use the key binding Ctrl-SPACE. You can close the popup either by selecting an entry in the list or simply by pressing the ESC button.

**Note 1:** If there are several SQL statements in the editor then make sure to separate them using the statement delimiter character (default to ";").

**Note 2:** In order for the column names completion popup to appear then you must first make sure there are table names in the statement.

**Note 3:** All table names that has been listed in the completion popup are cached by DbVisualizer to make sure subsequent displays of the popup is performed quickly without asking the database. The cache is cleared only when doing a **Refresh** in the datatbase objects tree or reconnecting the database connection.

**Note 4:** The **Schema** list above the editor is used only to assist the auto completion feature to limit what tables to list in the popup.

The following shows the completion popup with table names.

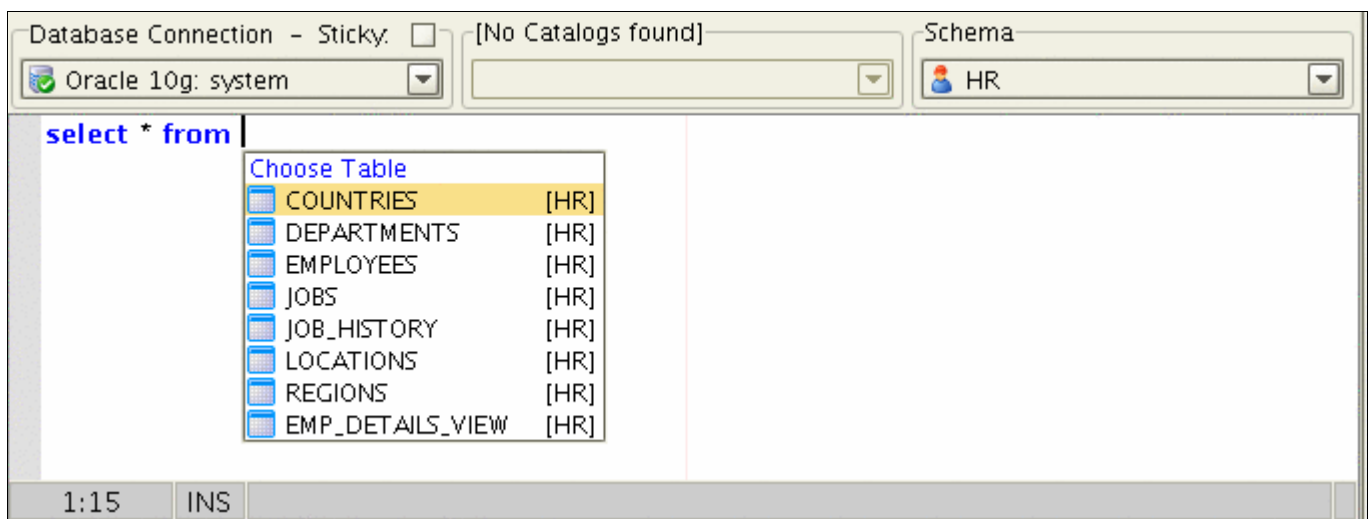
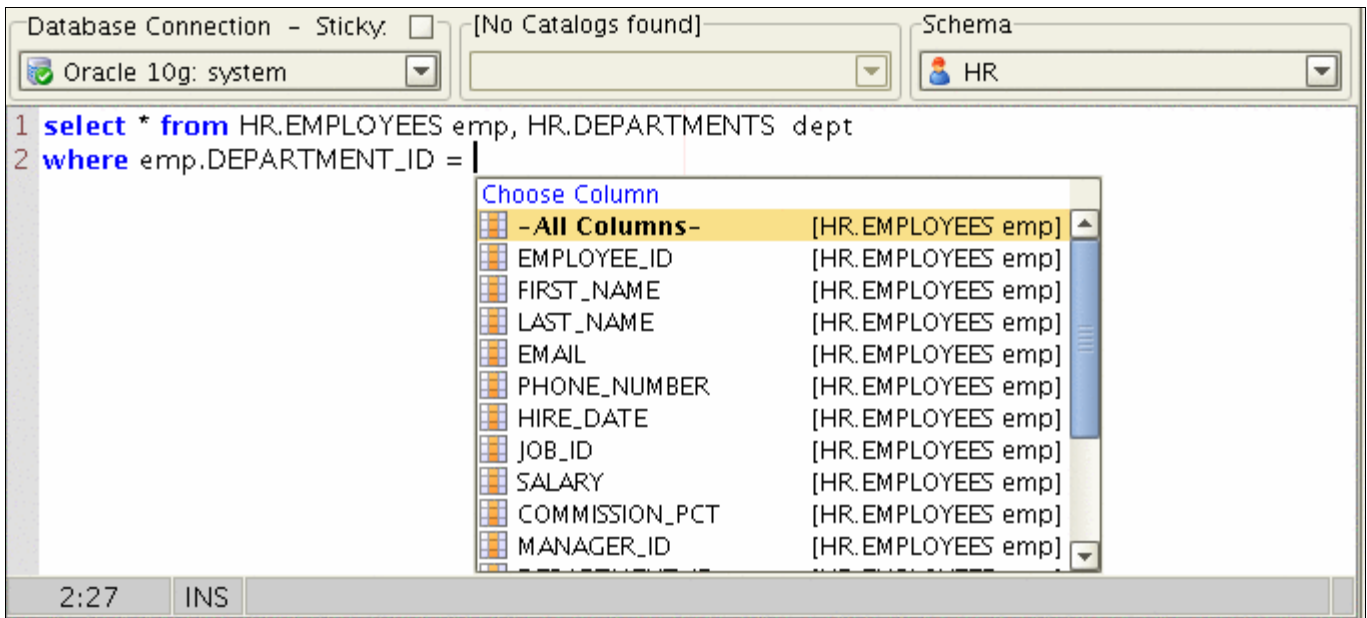


Figure: Auto completion popup showing table names

Here is another completion popup showing column names.

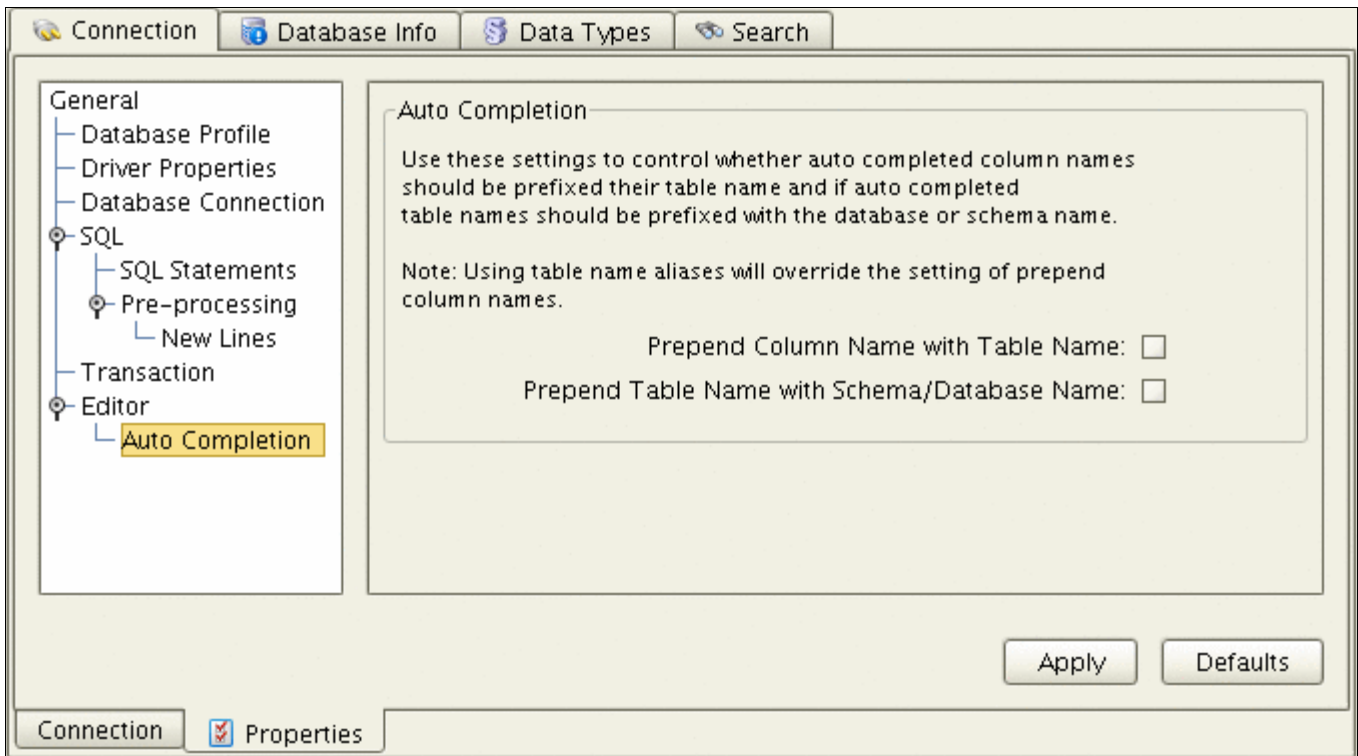


**Figure: Auto completion popup showing column names**

Here follows a couple of examples. The <AC> symbol indicates the position where the auto completion popup is requested. The currently selected catalog is empty and the selected schema is HR. (These examples are when accessing an Oracle database).

- `select * from <AC>` Shows all tables in the **HR** schema (since HR is the selected schema)
- `select * from SYS.<AC>` The popup will display all tables in the **SYS** schema independent of the schema list selection
- `select * from SYS.a<AC>` Lists all tables in the **SYS** schema beginning with the **A** character
- `select <AC> from SYS.all_objects` Lists all column in the **SYS.all\_objects** table
- `select <AC> from SYS.all_objects all, EMPLOYEES` Lists all columns in the **SYS.all\_objects** and **EMPLOYEES** table (in the **HR** schema)
- `select emp.<AC> from EMPLOYEES emp` Lists all columns in the **EMPLOYEES** table here identified by the alias **emp**
- `select emp.N<AC> from EMPLOYEES emp` Lists all columns in the **EMPLOYEES** table identified by alias **emp** starting with the **N** character
- `insert into EMPLOYEES (<AC>` Lists all columns in the **EMPLOYEES** table. Selecting the **-All Columns-** in the popup will result in that all columns will be added. Each table is comma separated.

It is possible to fine tune how auto completion shall work in the connection properties. The following settings can be used to adjust how auto completion should work.



**Figure: Properties controlling auto completion**

**Prepend Column Name with Table Name** specifies if completed column names should be prefixed with the actual table name i.e TABLE.COLUMN. This setting will only have effect if **NOT** using table name aliases.

The **Prepend Table Name with Schema/Database Name** setting is by its name quite self explanatory...

## History

The **History** operations available in the **View** main menu are used to walk forward and backward through the history of executed SQL statements. These operations are performed in the currently selected editor and simply insert the next or previously executed SQL with accompanying settings for **Database Connection** and **Catalog** (if **Sticky** is disabled).

The history entries are in fact SQL Bookmarks and managed by the History root folder in the [Bookmark Editor](#).

## SQL Bookmarks

SQL Bookmarks are used to manage favorite SQL statements between invocations of DbVisualizer. These are handled by the Bookmark Editor but the execution is performed in the SQL Commander. Please refer to the [SQL Bookmarks](#) document for how to use the **Bookmarks** main menu operations in the SQL Commander.

## Execution

The **Database->Execute** main menu operation is used to execute the SQL in the current (selected) SQL editor. The SQL Commander does this by analyzing the content in the editor to determine the SQL statements. It will then execute the statement(s) and indicate the progress. All statements in one editor are executed by the Database Connection that has been selected. The SQL Commander does not support executing SQL's for multiple database connections in one batch.

The result of the execution is displayed in the output view based on what result(s) are returned. If there are several results and an error occurred in one of them the [Log](#) view will automatically be displayed to indicate the error.

## Execution control

The execution of multiple SQL statements can be controlled using the **Stop Execution On** controls. These define whether the execution of the following SQL statements will be stopped based on two states:

- **Errors**  
Stop the execution if the SQL resulted in an error
- **Warnings**  
Stop the execution if the SQL executed successfully but no rows were affected

**Note:** The **Stop Execution On** controls are only effective when executing multiple SQL statements.

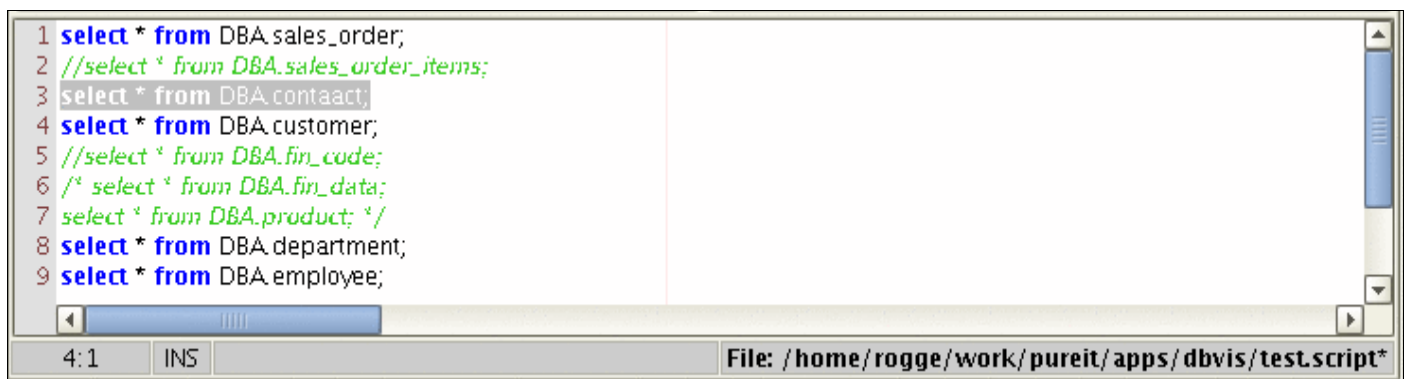
## Execute statement at cursor position

The **Execute Current** operation is useful when having a script of several SQL statements. Use it to execute the statement at the cursor position without first needing to select the SQL statement. The default key binding for execute current is **Ctrl-PERIOD** (Ctrl-.).

**Note:** Execute Current determines the actual statement by parsing the editor buffer using the standard statement delimiters.

## Selection executes

Selection Executes is useful when a batch of SQL statements are in the SQL editor and you just want to execute one or a few of the statement(s).



```
1 select * from DBA.sales_order;
2 //select * from DBA.sales_order_items;
3 select * from DBA.contact;
4 select * from DBA.customer;
5 //select * from DBA.fin_code;
6 /* select * from DBA.fin_data;
7 select * from DBA.product; */
8 select * from DBA.department;
9 select * from DBA.employee;
```

The screenshot shows a SQL editor window with a script of nine SQL statements. The third statement, `select * from DBA.contact;`, is highlighted in blue. The window title bar shows the file path: `File: /home/rogge/work/pureit/apps/dbvis/test.script*`.

**Figure: Selection execute**

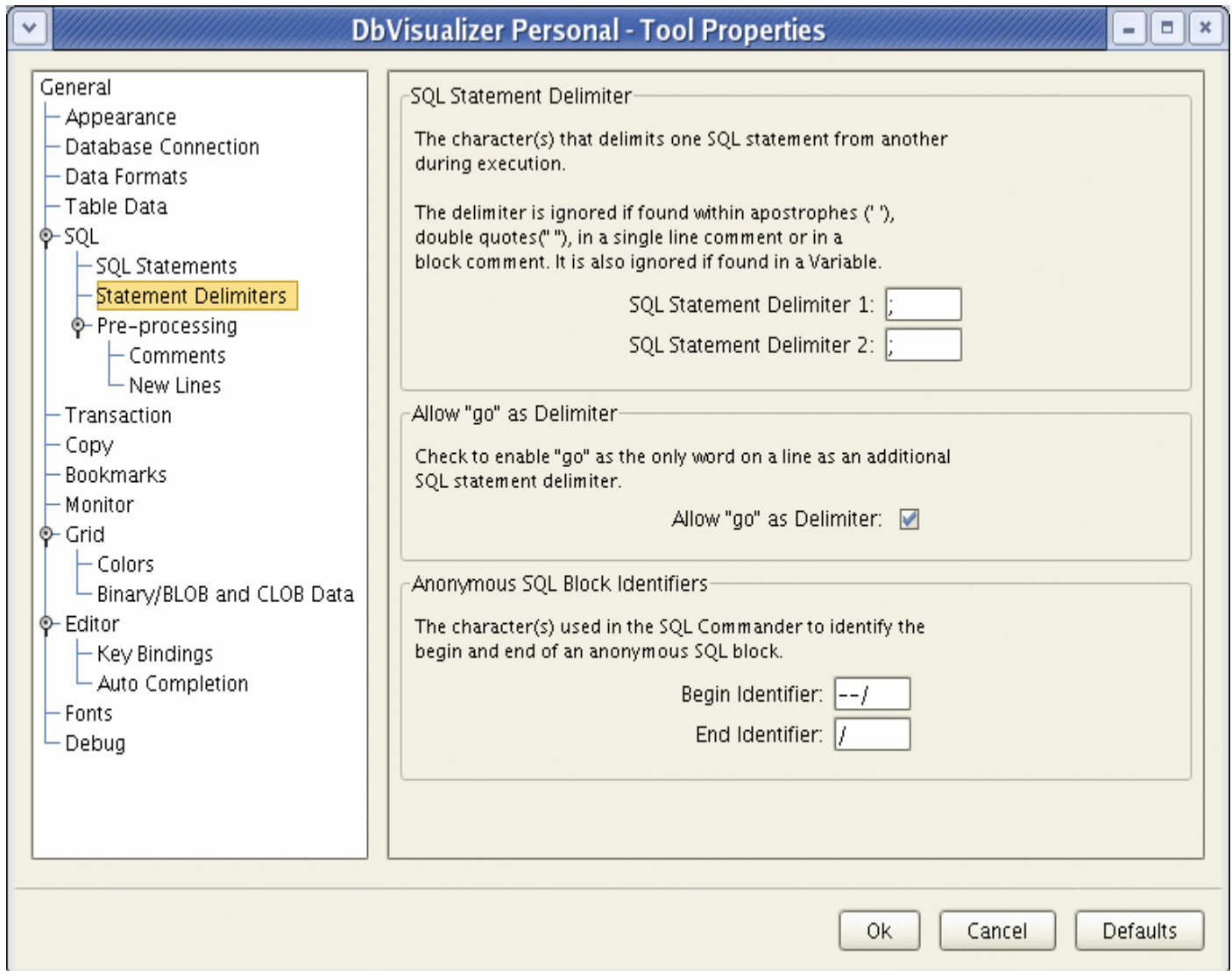
The above figure will result in only the highlighted statement being executed.

## Commit and Rollback

The commit and rollback SQL commands and the accompanying operations in the **Database** main menu are enabled only if the setting of **Auto Commit** is off for the database connection. The default setting for auto commit is on which means that the driver/database automatically commits each SQL that is executed. If auto commit is disabled then it is very important to manually issue the commit or rollback operations when appropriate.

## SQL Scripts

An SQL script is composed of several SQL statements and can be executed in a batch. Each SQL statement is separated by a single character, a sequence of characters or the go word on a single line. The default settings for the separator characters are defined in Tool Properties and can be modified to match your needs.



**Figure: Statement Delimiters**

The following SQL script illustrates some uses of the SQL statement delimiters based on the settings in the previous figure:

```
select * from MyTable; /* Stmt 1
*/

insert into table MyTable /* Stmt 2
(Id, Name) /* This is a comment */ values (1, 'Arnold')
*/
go /* Stmt 3
*/

update MyTable set Name = 'George' where Id = 1; /* Stmt 4
*/

select * from MyTable; // This is a comment
```



## Anonymous SQL blocks

An anonymous SQL block is a block of code which contains not only standard SQL but also proprietary code for a specific database. The anonymous SQL block support in the SQL Commander uses another technique in the JDBC driver to execute these blocks. The way to let the SQL Commander know that a SQL block is to be executed is to insert a begin identifier just before the block and an end identifier after the block. The figure in the previous section shows these settings and the default values:

**Begin Identifier:** `--/`

**End Identifier:** `/`

Here follows an example of an anonymous SQL block for Oracle:

```
--/ script to disable foreign keys
declare cursor tabs is select table_name, constraint_name
from user_constraints where constraint_type = 'R' and owner = user;
begin
  for j in tabs loop
    execute immediate
      ('alter table '||j.table_name||' disable constraint '||j.constraint_name);
  end loop;
end;
/
```

## Stored Procedures

Executing stored procedures is not officially supported by DbVisualizer even though it works for some databases. The best way to figure it out is to test.

Our internal tests show that the Sybase ASE and SQL Server procedure calls work ok in the SQL Commander. DbVisualizer also presents multiple result sets from a single procedure call as of version 4.0 for these databases.

## Control commands

The SQL Commander supports executing SQL script files without first needing to load the file into the SQL editor. This is accomplished by using the `@` control character in the first position of a row. This is useful if you are using an external editor or IDE to edit the SQL and then use DbVisualizer to execute it. The following shows how to use the `@` control character.

```
select * from           -- Selects data from MyTable
MyTable;

-- Load and execute the content in the
-- createDB.sql file. The location
@createDB.sql        -- of this file is the same as the working directory
-- for DbVisualizer.

@cd /home/mupp;    -- Request to change directory to /home/mupp
-- Load and execute the content in the
@loadBackup.sql    -- loadBackup.sql file. This file will now be
-- loaded from the /home/mupp directory.
```

## Variables

Variables can be used to build parameterized SQL statements. The SQL Commander will at

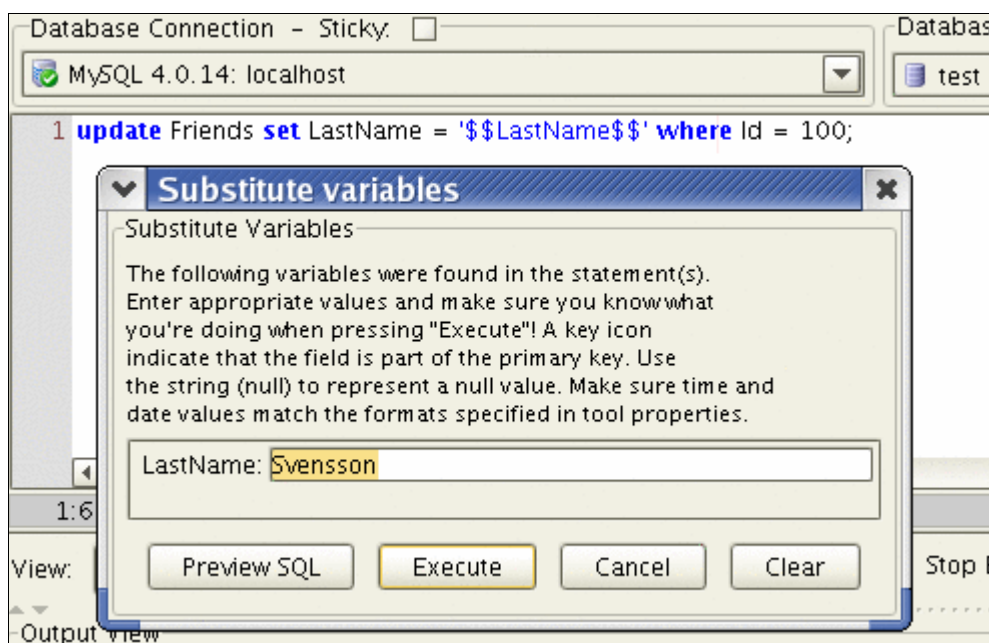
execution check for variables and prompt for replacement values of the variables. Variables are also used internally in DbVisualizer. The SQL templates that are listed in the Tool Properties->SQL->SQL Statements category are used inside DbVisualizer in various situations. The difference with these is that DbVisualizer automatically substitutes the pre-defined variable names with correct values once the templates are used instead of prompting for values as the SQL Commander does.

A variable has the following format in its simplest use:

```
$$FullName$$
```

A variable must begin and end with the character(s) identified by the **Variable Identifier** property in the **Tool Properties->SQL** category (default is **\$\$** as in the example above). During execution the SQL Commander will search for variables and display a window with the name of each variable and an input (value) field. Enter the value for each variable and then press **Execute**. This will replace the original variable with the value and finally let the database execute the statement.

**Tip:** Use the **Ctrl->Enter** key binding as a shortcut for **Execute**.



**Figure: The substitute variables window**

The above example is the simplest case as it only contains the variable name. In this case it is also necessary to place the text value within quotes since the substitution window cannot determine from the variable itself if it is a number or text variable.

The final substituted SQL statement that results from the initial SQL and variable value is:

```
update Friends set LastName = 'Svensson' where Id = 100;
```

## Variable Syntax

The variable format supports setting a default value, data type and a few options as in the following example:

```
$$FullName||Swansons||String||where pk $$
```

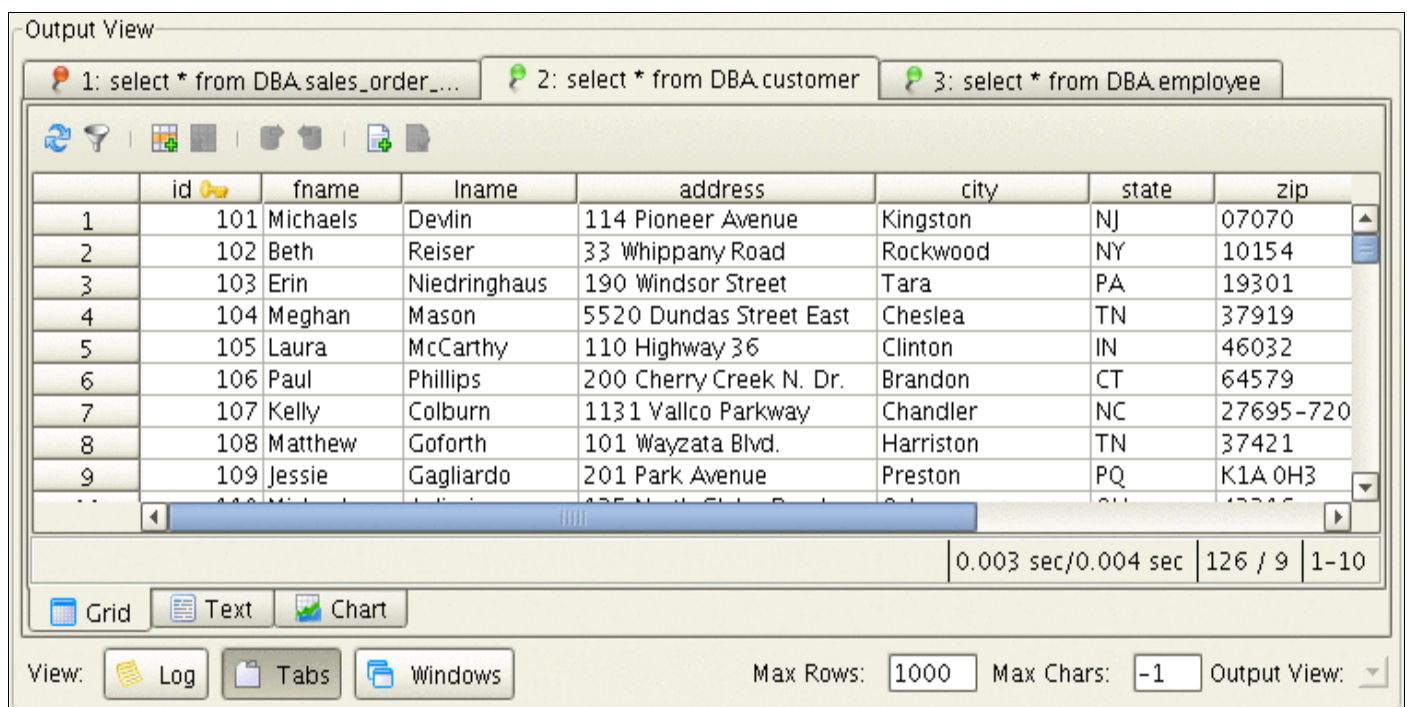
The full format of the variable syntax is:

```
$$variableName [|| defaultValue [|| type [|| options]]]
```

- **variableName**  
Required. This is the name that will appear in the substitution dialog. If several variables have the same name then the substitution dialog will show only one and the entered value will be applied to all variables of that name.
- **defaultValue**  
The default value that will appear in the substitution dialog
- **type**  
The type of variable - String, Integer, BinaryData, etc. This is used to determine if the value will be enclosed by quotes or not. If no type is specified then it is treated as an Integer (no quotes).
- **options**  
The options part is used to express various things. Most interesting are the **pk** and **where** keywords.  
(Note: There must be a whitespace character following a keyword).
  - **pk**  
Defines whether an icon will appear before the variable name in the substitution dialog to indicate that it is a primary key field.
  - **where**  
Defines that the variable is part of the where clause and so will appear last in the list of variables.

## Output View

The **Output View** in the lower area of the SQL Commander is used to display the result of the SQL's being executed. How the results are presented is based on what type of result it is. A log entry is always produced in the **Log** view for each SQL statement that is executed. This entry shows at a minimum the execution time and how many rows were affected by the SQL. There may also be a result set if the SQL returned one. These result sets are presented either as tabs or windows based on your choice.



**Figure: The output view**

If an error occurs during execution the SQL Commander will automatically switch to the Log view so that you can further analyze the problem.

## Output View menu

The Output View menu in the lower right area contains the following choices:



<b>Load SQL into Editor</b>
<b>Insert SQL into Editor</b>
 <b>Pin All</b>
 <b>Unpin All</b>
<b>Remove Pinned</b>
<b>Remove Unpinned</b>
<input type="checkbox"/> <b>Pin New Result Sets</b>
<b>Remove Empty Result Sets</b>
<b>Clear Output View</b>

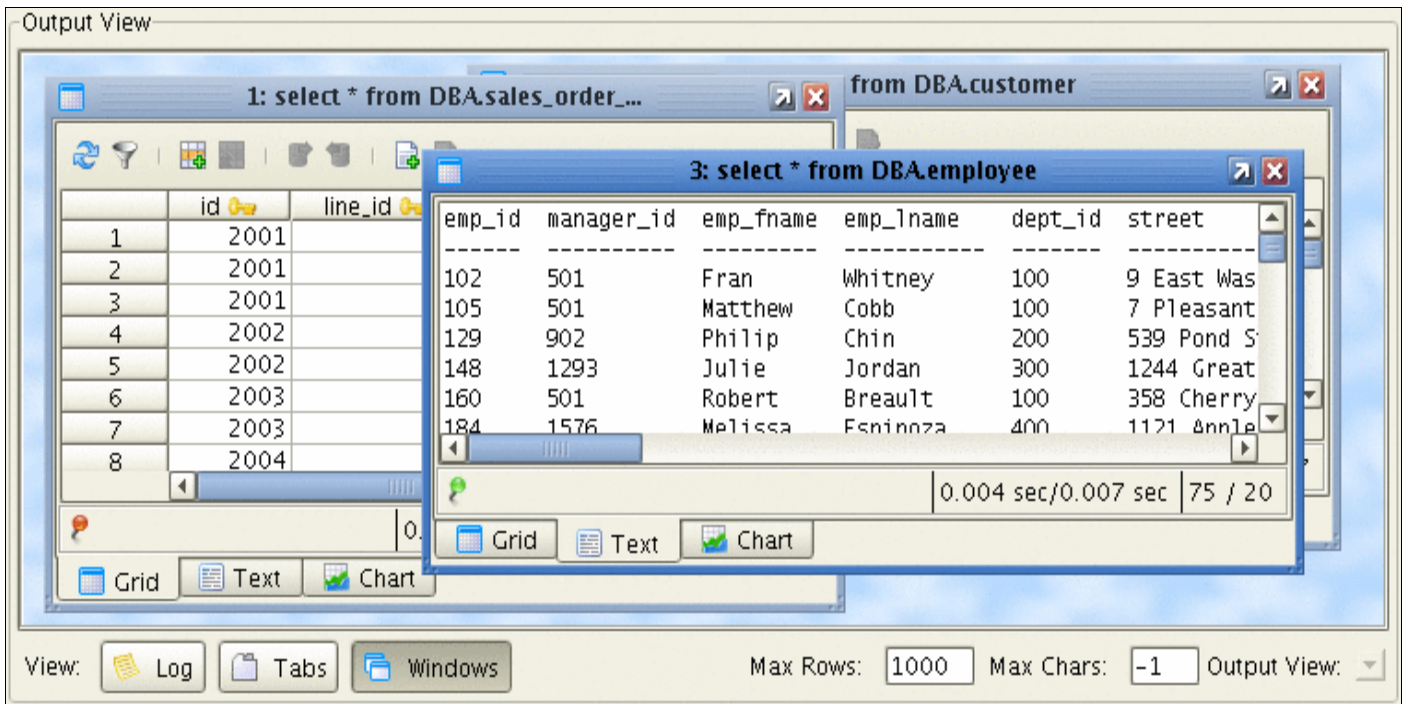
Figure: The output view

Menu Choice	Description
<b>Load SQL into Editor</b>	Loads the SQL for the selected result set tab or window into the current editor.
<b>Insert SQL into Editor</b>	Inserts the SQL for the selected result set tab or window into the current editor at the cursor position.
<b>Pin All</b>	Pins all result sets. Pinning a result set will prevent it from being removed at the next execution.
<b>Unpin All</b>	Unpins any pinned result sets making them candidates for removal during the next execution.
<b>Remove Pinned</b>	Removes all pinned result sets directly.
<b>Remove Unpinned</b>	Removes all unpinned result sets directly.
<b>Pin New Result Sets</b>	Check this to make sure all new result sets are pinned by default.
<b>Remove Empty Result Sets</b>	Removes all empty result sets.
<b>Clear Output View</b>	Clears the current view. This operation is valid to use when the Log is being displayed and will clear all log entries.

## Result set grids

A result set grid is created for every SQL that returns one or more result sets. These grids can be displayed in a tab or window style view similar to how the SQL editors are displayed. Each grid shares the common layout and features as described in the [Getting Started and General Overview](#) document. The format of the result can be one:

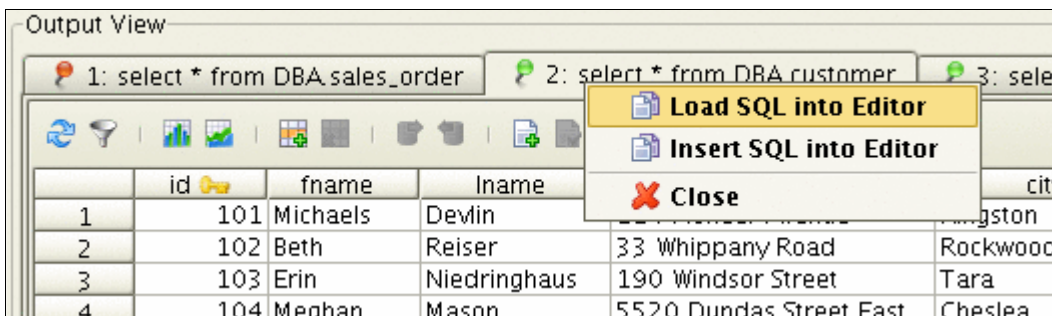
- **Grid**  
The result is presented in a grid.
- **Text**  
The result is presented in a tabular format.
- **Chart**  
Read more in [Monitor and Charts](#).



**Figure: The windows output view**

This figure shows the **Windows** output view with three result set grids. The **Max Rows** and **Max Chars** fields at the bottom of the figure are used to set the maximum number of rows and columns (for text data) that will be fetched and presented in new result set grids. The labels for the number of rows and columns in the grid will be displayed in red if either of these exceed their respective maximum settings. A result set grid can be closed using the red cross in the window frame header.

If the output view is Tabs then use the **Close** right click menu choice when the mouse pointer is in the tab header:



**Figure: The right click menu for tabs**

### Editing

A result set grid may be enabled for editing based on the following criterias:

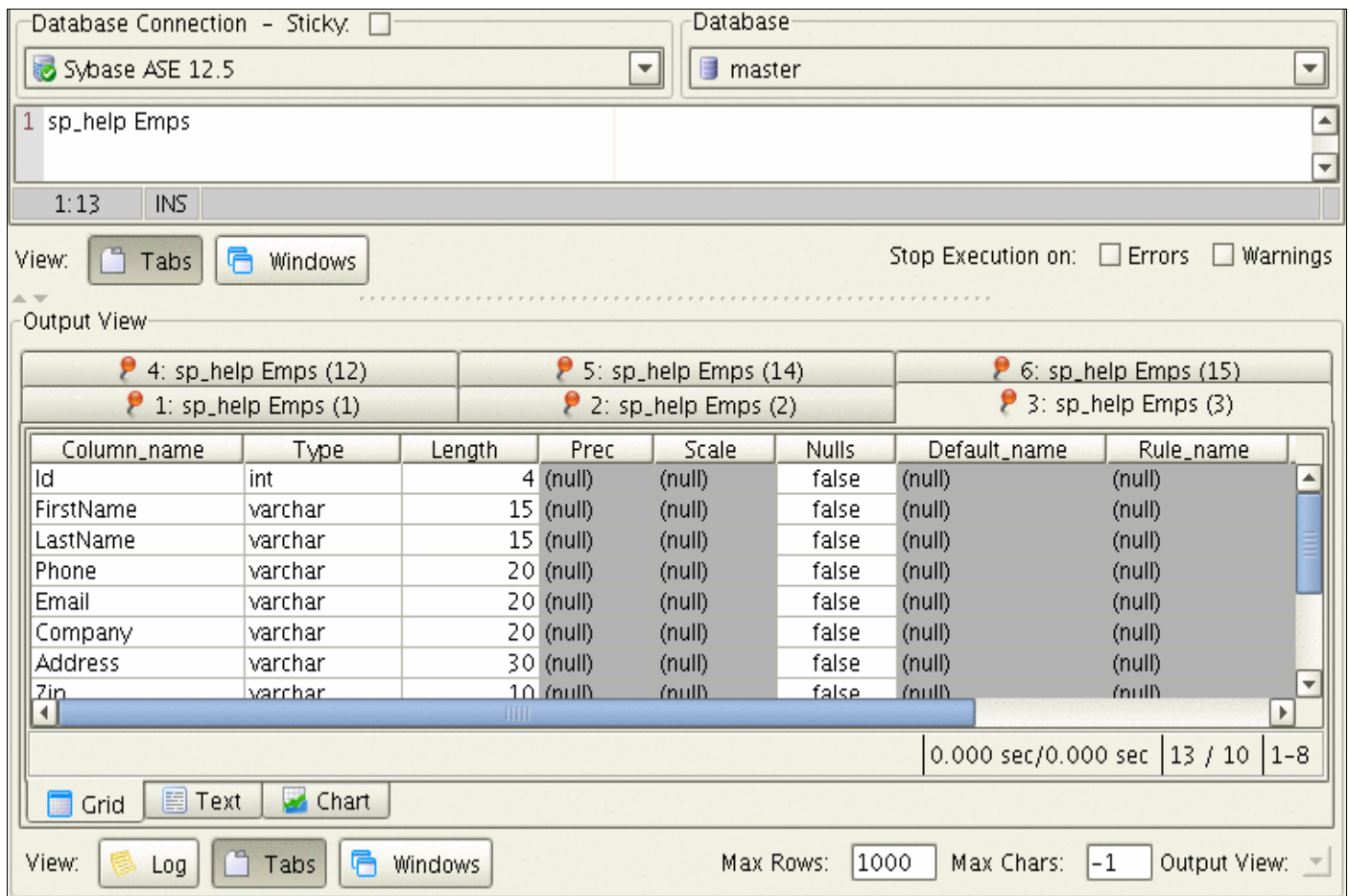
1. The result really is a result set
2. The SQL is a SELECT command
3. Only one table is referenced in the FROM clause
4. All columns in the result set exist in the table with exactly matching names

If all the above is true then the standard editing tool bar will appear just above the grid. Read more about editing in the [Edit Table Data](#) document.

If any of the above fail to comply will the editing tool bar not appear.

## Multiple result sets produced by a single SQL statement

Some SQL statements may produce multiple result sets. Examples of this are stored procedures in Sybase ASE and SQL Server. The SQL Commander will simply check the results as returned by the JDBC driver and add grids to the output view accordingly. The following shows the **sp\_help Emps** command which returns several result sets with various information about the Emps table.



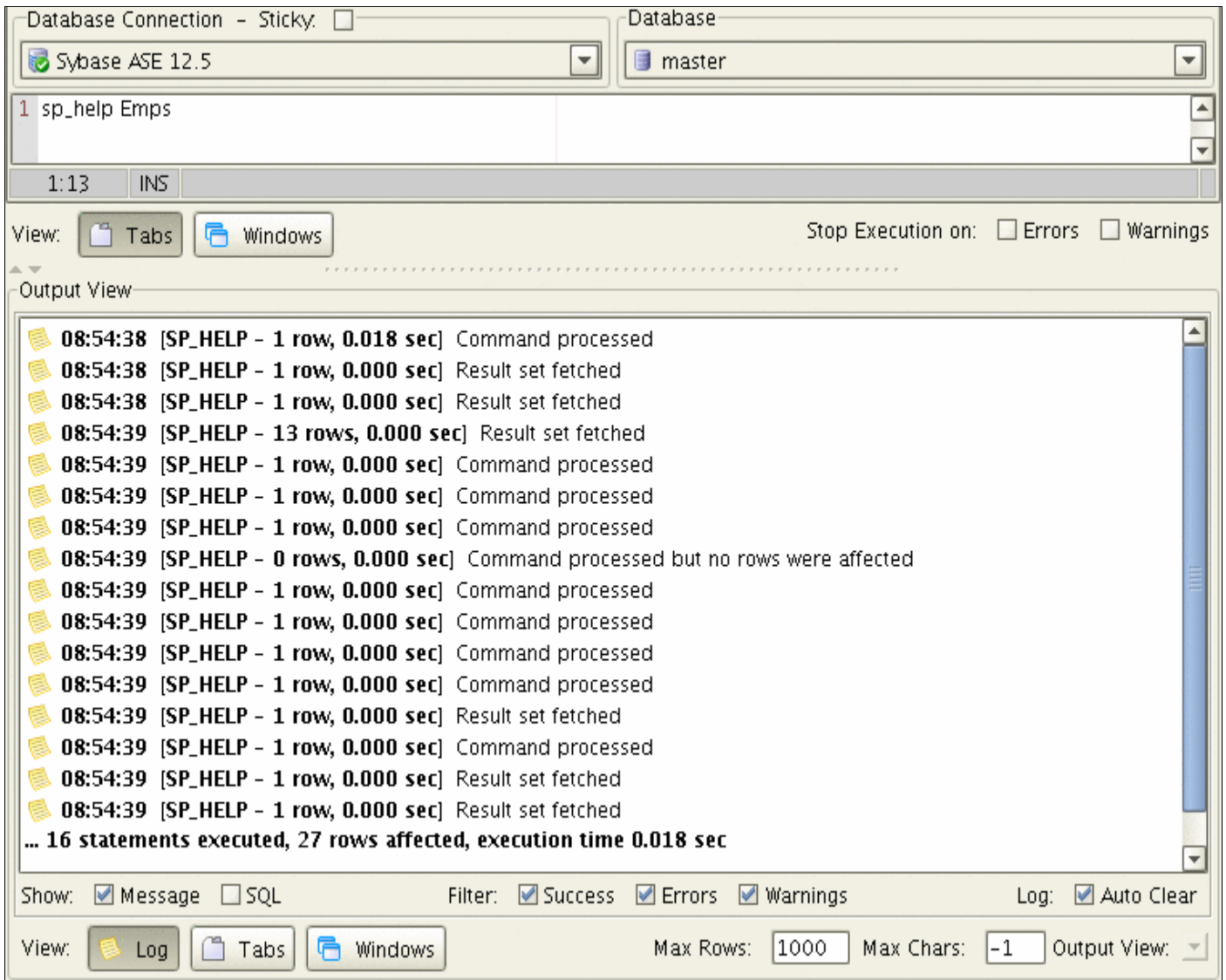
The screenshot shows the SQL Commander interface. At the top, the database connection is set to 'Sybase ASE 12.5' and the database is 'master'. The command editor contains the text '1 sp\_help Emps'. Below the command editor, the 'Output View' displays six result set grids. The first grid, labeled '1: sp\_help Emps (1)', shows the following table metadata:

Column_name	Type	Length	Prec	Scale	Nulls	Default_name	Rule_name
Id	int	4	(null)	(null)	false	(null)	(null)
FirstName	varchar	15	(null)	(null)	false	(null)	(null)
LastName	varchar	15	(null)	(null)	false	(null)	(null)
Phone	varchar	20	(null)	(null)	false	(null)	(null)
Email	varchar	20	(null)	(null)	false	(null)	(null)
Company	varchar	20	(null)	(null)	false	(null)	(null)
Address	varchar	30	(null)	(null)	false	(null)	(null)
Zip	varchar	10	(null)	(null)	false	(null)	(null)

The other five result set grids (labeled 2 through 6) are partially visible but their content is not shown. The interface also includes a 'View' section with 'Grid', 'Text', and 'Chart' options, and a status bar showing '0.000 sec/0.000 sec', '13 / 10', and '1-8'.

**Figure: Multiple result set grids produced by a single SQL statement**

The result set grids above all share the same label, **sp\_help Emps**. The number after the label represents the order number for the actual result. A stored procedure can return different results, not all being result sets. The number helps to identify in the log which entry matches what result set grid. Here is the Log output view for the previous example.



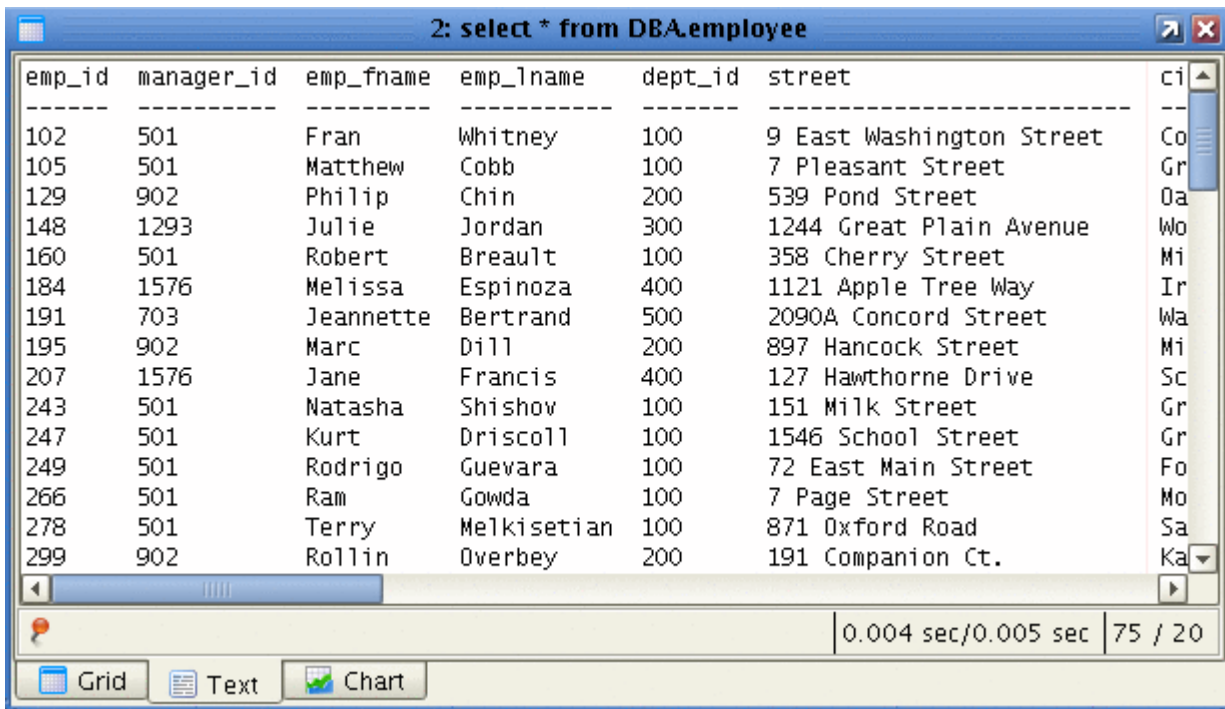
**Figure: The Log after executing an SQL statement that returns multiple results**

All entries with the log message "**Result set fetched**" are represented in the previous figure.

### **Text**

The **Text** format for a result set presents the data in a tabular style. The column widths are calculated based on the length of each value and the length of the column label.

**Note:** The columns widths may vary between executions of the SQL.



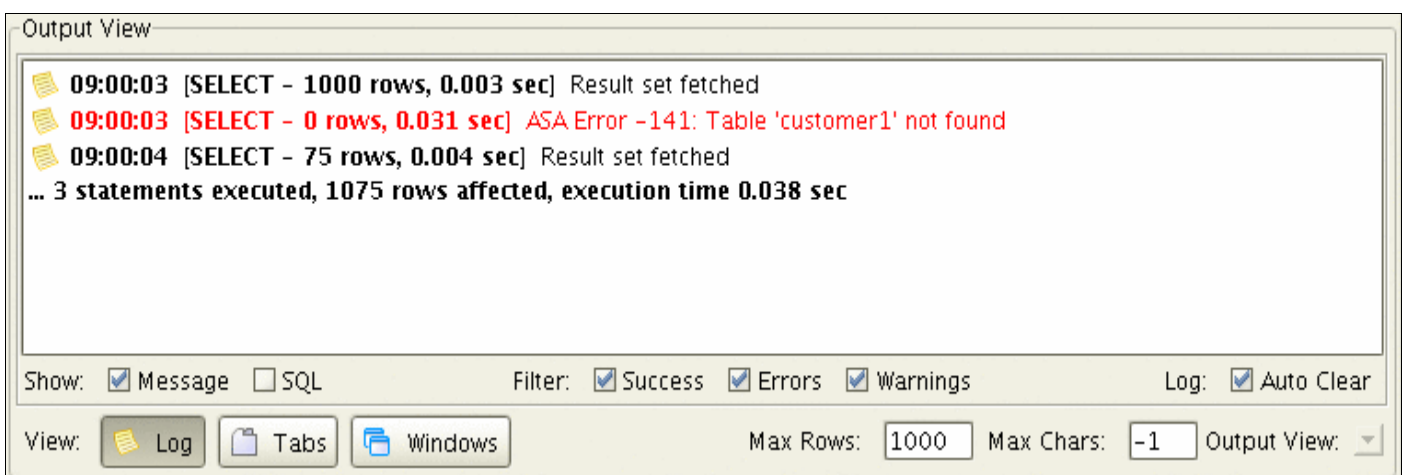
**Figure: The Text result set format**

## Chart

A result set can be charted using the **Chart** view in a grid. Please read more about it in the [Monitor and Charts](#) document.

## Log

The log keeps an entry for each SQL statement that has been executed. It keeps generic information such as how many rows were affected and the execution time. The important piece of information is the execution message which shows how the execution of that specific statement ended. If an error occurred then the complete log entry will be in red indicating that something went wrong.



**Figure: The Log with one failed statement**

The detail level in an error message is dependent on the driver and database that is being used. Some databases are very good at telling what went wrong and why while others are very quiet. The icon to the left of each log entry is used to pass the SQL for the entry into the current SQL editor when clicked.



## Log controls

The **Show** controls below the log are used to define what information will appear in the log. The **Filter** controls are used to specify what entries will be displayed.

## Auto clear log

The **Auto Clear Log** control can be enabled to let the SQL Commander automatically clear the log between executions.

# Monitor and Charts

## Introduction

The monitor feature in both editions of DbVisualizer is used to show the results of one or many SQL statements in the Monitor window. These monitors can then be re-loaded (executed) manually or by the auto reloader which will automatically re-load all monitors at a given interval. A monitored SQL statement is an SQL Bookmark and the definition and management of which SQL Bookmarks are monitored is controlled in the Bookmark Editor. Any SQL Bookmark that produces a result set (data in a grid) can be monitored. The monitor feature supports monitoring SQL Bookmarks for different database connections concurrently.

The monitoring feature in conjunction with the charting capability in DbVisualizer Personal is really powerful since it delivers real time charts of one or several monitored SQL Bookmarks simultaneously. Typical scenarios when this is useful are to see live trends in a production database, surveillance, statistics, etc. It is just a matter of imagination and the level of SQL expertise that sets the limit! At Minq Software there is a dedicated work station that automatically presents various charts from our database.

Charts can also be exported to JPEG and PNG files.

**Note:** Charts cannot be printed directly. You must first export and then use another tool to print.

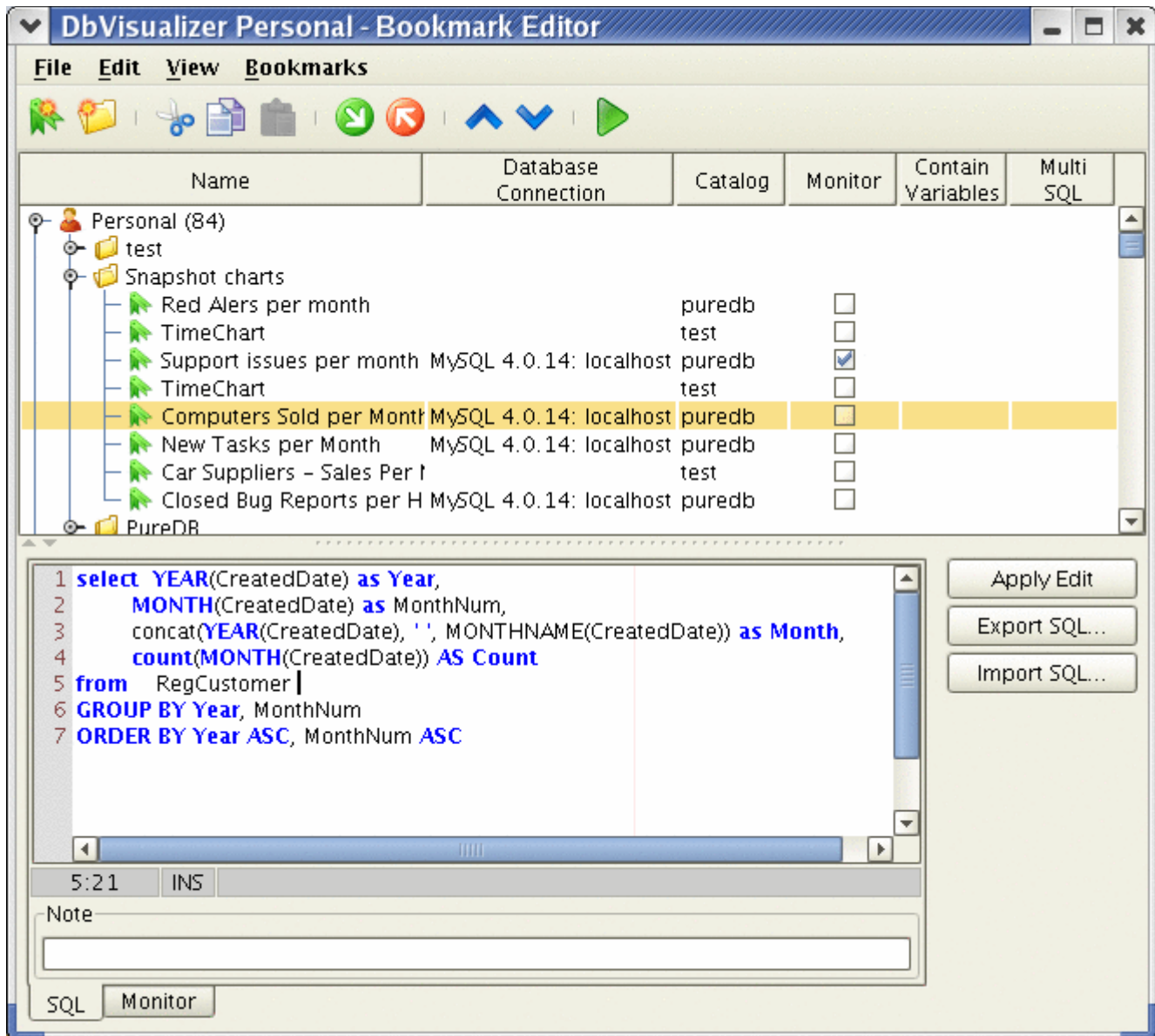
**Note:** The chart customization covered in this document is also applicable in the SQL Commander (DbVisualizer Personal).

**The Monitor window with four monitored SQL Bookmarks. The results can be viewed as windows or tabs. This example shows the grid data as returned from each SQL statement.**

**The same monitored SQL Bookmarks as in the left figure but here presented as charts. (DbVisualizer Personal)**

## Monitor an SQL statement

An SQL statement to be monitored must be defined as an SQL Bookmark in DbVisualizer. A bookmark is briefly an SQL statement with associated information about its database connection and an optional catalog (generic JDBC denomination which translates to a *database* in for example Sybase, MySQL, SQL Server, etc). The Bookmark Editor supports organizing SQL bookmarks in a tree structured folder view and the complete structure and all SQL Bookmarks are saved in the XML file between invocations of DbVisualizer. It is the Bookmark Editor that is used to enable a SQL Bookmark to become a monitored SQL Statement.



**Figure: Bookmark Editor**

The figure shows the **Computers Sold per Month** bookmark and the SQL that is associated with it. The **Monitor** field in the tree is used to determine whether the SQL Bookmark is a monitor or not. Just click on the check mark and the SQL Bookmark will appear in the Monitor window. Uncheck it to remove the monitor.

The following is an example of what the above SQL Bookmark produces:

Year	MonthNum	Month	Count
1999	2	1999 February	3
1999	3	1999 March	10
1999	4	1999 April	10
1999	5	1999 May	25
1999	6	1999 June	31
1999	7	1999 July	138
1999	8	1999 August	72
1999	9	1999 September	159
1999	10	1999 October	406
1999	11	1999 November	151
1999	12	1999 December	259
2000	1	2000 January	367
2000	2	2000 February	397

**Figure: Monitor showing the result in Grid format**

The interesting columns in the result are the **Month** and **Count**. The **Year** and **MonthNum** are there just to get the correct ascending order of the result.

There are alternative operations to simplify creation of monitored SQL Bookmarks. Read the following sections to find out how this is done.

### Monitor table row count

The Monitor Table Row Count operation is activated in the Data tab for a table (left button below):

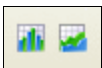


Figure: Data tab tool bar buttons that are used to create monitors

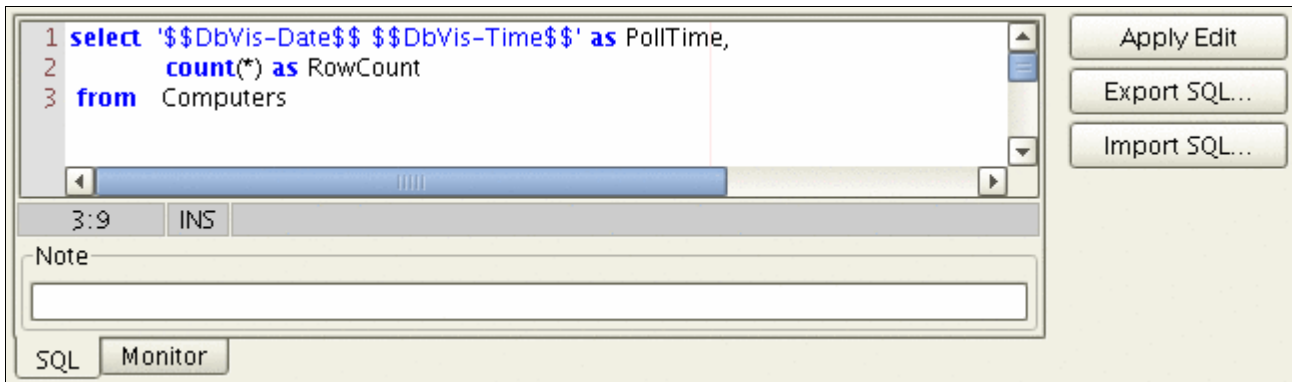
It is used to create a monitor that displays in a single row the current time stamp for when the monitor is executed and the total number of rows (count(\*)) in the table. Each execution of the monitor will result in one row being added to the grid. The monitoring feature in this example keeps a pre defined number of rows until the oldest rows are removed. Example:

Computers: Row Count	
PollTime	RowCount
2003-01-23 12:19:10	43123
2003-01-23 12:11:40	43139
2003-01-23 12:21:10	43143
2003-01-23 12:22:40	43184
...	...

**Figure: Example of the result from a Table Row Count monitor**

The SQL for this monitor introduces two variables, **DbVis-Date** and **DbVis-Time**. These variables are substituted with the current date and time formatted according to the formats in Tool Properties. The reason these variables are used instead of using appropriate SQL functions to retrieve them is simply because it is almost impossible to get the values of these in a database independent way. Another reason is that we want to set the time of the client machine rather than the database. The

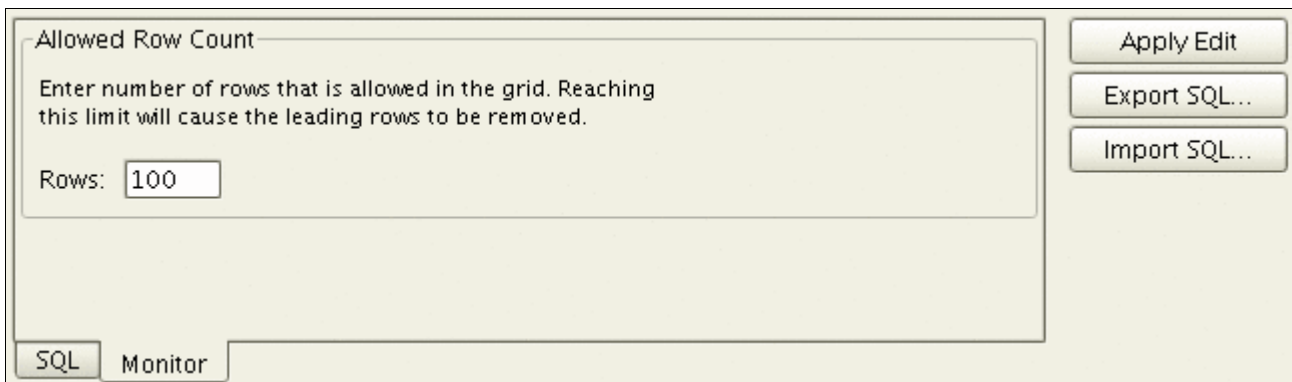
SQL can of course be modified to contain whatever SQL that is appropriate as long as the **PollTime** and **RowCount** labels are not changed.



**Figure: Sample of the SQL for the Table Row Count monitor**

The above does not introduce any big news for an SQL hacker.

The magic of this monitor is that it keeps a pre defined number of rows in the grid. This is managed by the **Allowed Row Count** property in the Bookmark Editor. This property is automatically set when creating a row count monitor. The default value is to keep the 100 latest rows added to the grid (one per execution).

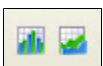


**Figure: Allowed Row Count property pane**

The above setting can be modified to limit or extend the number of rows that the monitored grid will keep. Setting it to 0 or a negative number tells DbVisualizer to always clear the grid between executions of monitors.

## Monitor table row count difference

The Monitor Table Row Count Difference operation is activated in the Data tab for a table buttons (right button below):



**Figure: Data tab tool bar buttons that are used to create monitors**

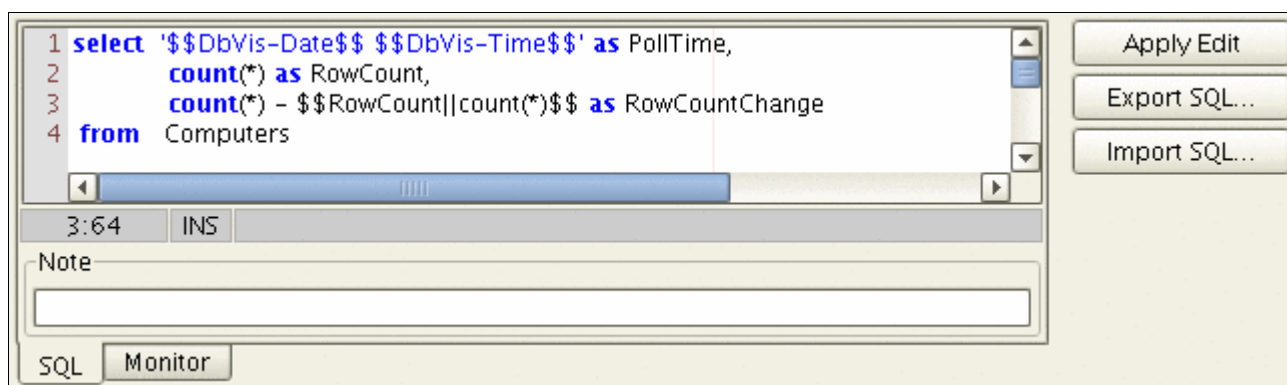
Its purpose is similar to Monitor Table Row Count except that this monitor reports the difference between the two latest executions in the result grid:



PolITime	RowCount	RowCountChange
2003-01-23 12:19:10	43123	0
2003-01-23 12:11:40	43139	16
2003-01-23 12:21:10	43143	4
2003-01-23 12:22:40	43184	41
...	...	...

**Figure: Example of the result from a Table Row Count Difference monitor**

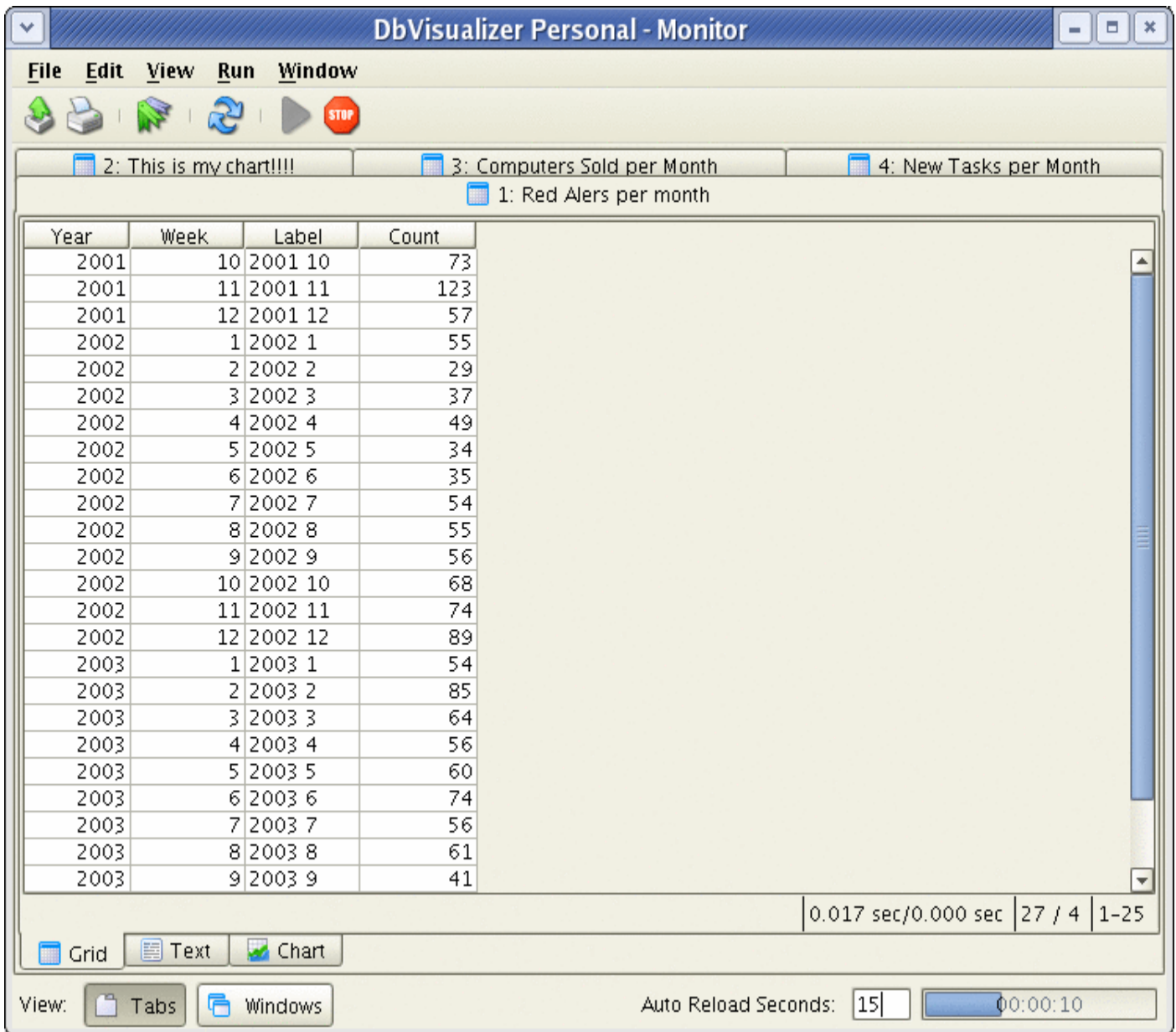
The SQL for this monitor adds a third field which is the **RowCountChange**. It is rather simple since the current count(\*) in the table is used when subtracting the RowCount in the previous execution round or count(\*) if there are no previous rows in the grid. This gives the difference. The trick here is that DbVisualizer always keeps all values of the last row that was added in the grid. Any of its fields can be referenced in the succeeding execution of the monitor.



**Figure: Sample of the SQL for the Table Row Count Difference monitor**

## Monitor window

The Monitor feature launched via the **Tools->Monitor** menu option is used to browse the active monitors. The monitors can be organized either as tabs or internal windows. In DbVisualizer Free the monitor results can be viewed as grids while DbVisualizer Personal adds the capability to view them as charts. The following figure is a screen shot of the Monitor window:



**Figure: The Monitor window with all monitors organized as tabs**

The screen shot is from DbVisualizer Personal as the selected monitor has the **Text** and **Chart** sub tabs which are not there in DbVisualizer Free. The **Auto Reload** feature at the bottom of the main window is used to control whether auto update of all monitors is enabled or not. The **Seconds** field specifies how many seconds the Monitor feature should wait before doing an auto reload. If auto reloading is enabled then the monitor toolbar icon in the main window is displayed to indicate its state. The **Edit Bookmark** button is used to open the Bookmark Editor for the currently selected monitor. The Bookmark Editor will automatically locate the actual SQL Bookmark for the monitor.

**Note:** The specified number of seconds may be increased automatically by DbVisualizer if the total execution time for all monitors is longer.

The Window menu contains choices to control the appearance in the Monitor:

	<b>Show Grids</b>	Ctrl+4
	<b>Show Texts</b>	Ctrl+5
	<b>Show Charts</b>	Ctrl+6
	<b>Cascade Windows</b>	
	<b>Tile Windows</b>	

## Figure: Window menu operations

The **Show Grids**, **Show Texts** and **Show Charts** toggles the monitors to display the monitors in the selected view. **Cascade** and **Tile** are used to automatically arrange the windows in the Windows view.

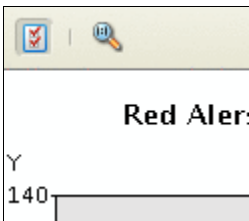
## Charts

**This section is only applicable for DbVisualizer Personal.**

Charts in conjunction with the Monitor feature is really powerful since monitored data is very often a good candidate to be charted. The charting capability in DbVisualizer Personal is also available in the SQL Commander feature even though this document does not cover it.

The basic setup of a chart is really easy since it is just a matter of selecting one or more columns that should appear as series in the chart. The basic requirement is that the monitor has been executed so that there are columns to choose the series from. The appearance of the charts can be thoroughly customized using the advanced customization editor.

The chart view is controlled by sub toolbar for each monitor:



## Figure: Chart control buttons

The controls are from the left:

1. Show/Hide chart controls pane
2. Reset any zoom

The following sections explain the features and how to setup the chart.

## Chart Controls

The chart controls are used to customize the **Data** that shall be displayed in the chart, optional axis labels, titles, etc. It is also used to control the **Layout** of the chart in terms of chart type, legend type, etc.

## Data

Specify in the Data customization which data shall appear in the chart.



Data **Layout**

X-Axis Label:

Series	Column	Label	Type
<input type="checkbox"/>	Year		Long
<input checked="" type="checkbox"/>	Week	Vecka	Long
<input type="checkbox"/>	Label		String
<input checked="" type="checkbox"/>	Count		Long

Series:

Chart Title: Red Alerts per month

X-Axis Title: X Rotation: None

Y-Axis Title: Y Rotation: None

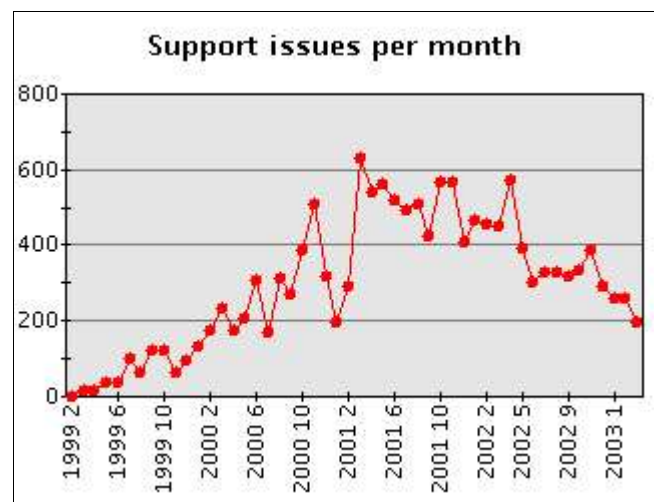
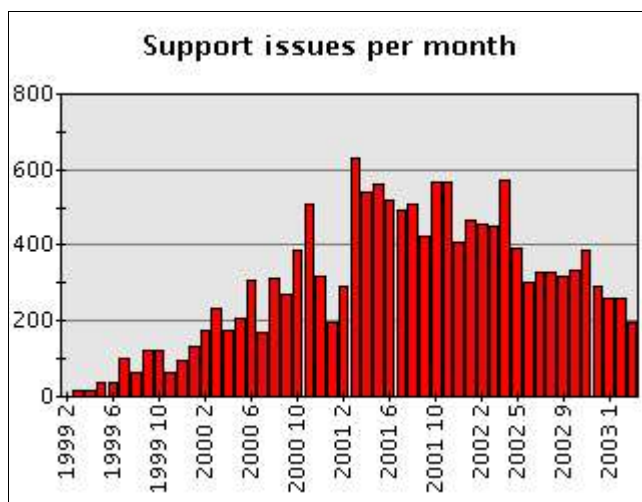
**Figure: Data customizer**

Select at least one **Series** from the list of columns and the chart is ready! Selecting several series will show them accordingly in the chart. The **Label** field can be used to specify an optional label for the serie as it will appear in a legend. The name of the column is used if no label is specified.

The **X-Axis Label** box is used to specify the column in the result that should be used to render the labels of the X-axis. **Chart Title** specifies the main title of the chart. This is the same title as the SQL Bookmark in the Bookmark Editor. **X-Axis Title** and **Y-Axis Title** specifies the titles for the X and Y axis. The **Rotation** settings are used to set the rotation of the X and Y axis.

## Layout

The layout tab is used to configure the appearance of the chart and primarily what type of chart that will be displayed. Note that all settings are per monitor. The following screen shots show some of the most commonly used chart types.



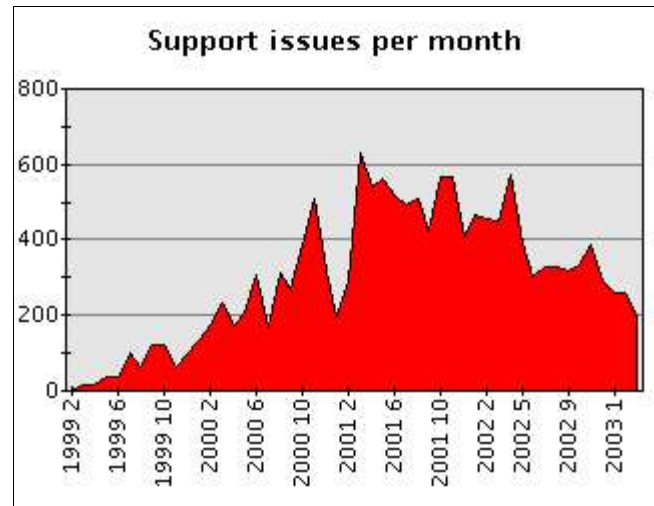
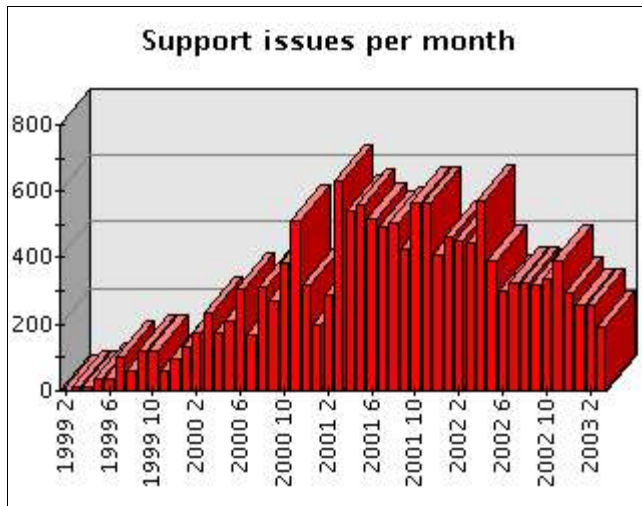


Figure: Chart type examples

The advanced layout editor can be used to customize every aspect of the layout. The basic layout settings however are the following:

Data
Layout

Show Symbols:

Show Inverted:

3D:

Chart Type: Area

Fill Pattern: Solid

Legend Type: Regular

Advanced Settings: Start Editor...

Figure: Layout customizer

**Show symbols** specifies whether each value in a line chart will be represented by a symbol. **Show Inverted** defines whether the X and Y axis will be switched. **3D** specifies if a bar chart will be displayed in 3D. The **Chart Type** lists all the available chart types. Fill Pattern defines how a bar, area and pie chart shall be filled. **Legend Type** specifies whether a legend will be displayed or not.

The **Advanced Settings** editor is used to customize all the bits and pieces of the chart. This document does not explain all the configurations that can be done using this editor since that would result in a 100 page book.

**Note:** Settings that are made in the Advanced Editor are not saved between invocations of DbVisualizer.

## Chart View

### Zooming

Charts support zooming by selecting a rectangle in the chart area. Selecting another rectangle in that zoomed area will zoom the chart even further and so on. To reset the zoom then just press the Reset Zoom button or the "r" keyboard button while the mouse pointer is in the chart area.

### Rotating

All 3D chart types support rotating and changing the depth of the chart. Use the following to change the appearance:

- **Shift+Left Mouse button**  
Changes the depth of the chart
- **Ctrl+Left Mouse button**  
Changes the rotation of the chart

Examples:

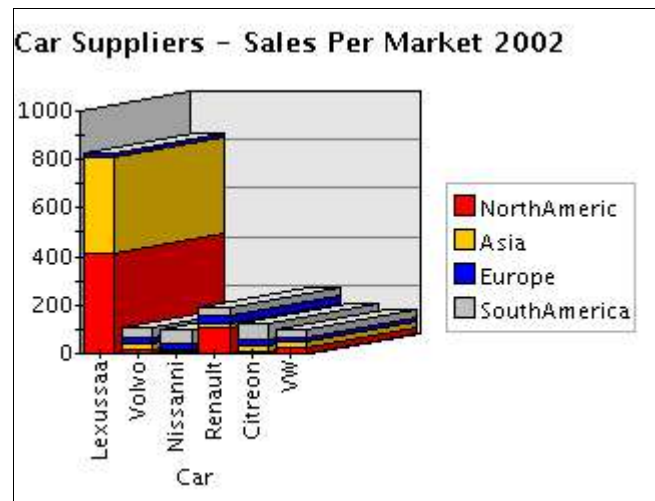
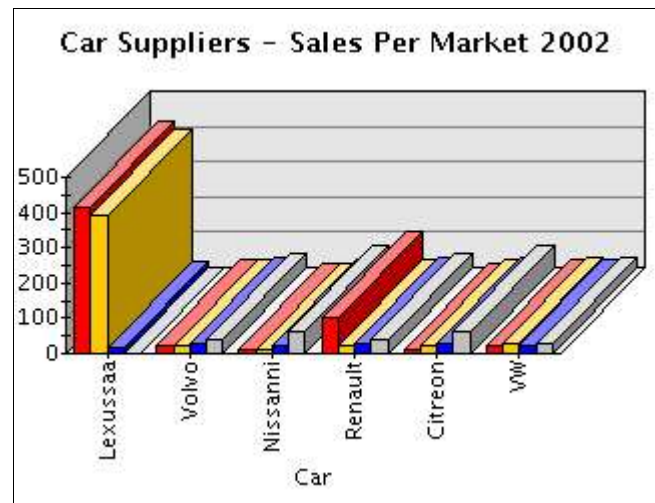
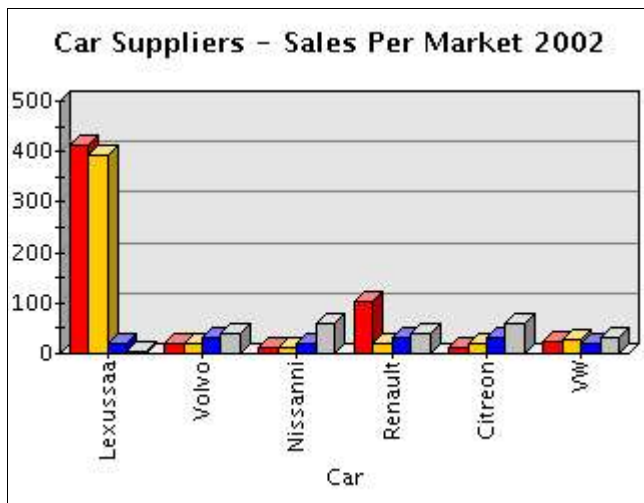


Figure: Example of 3D charts

The above screen shots are just a few examples of the 3D chart types and how depth and rotation settings are used to change the appearance.

## Export

The export operation is context sensitive and works on the currently selected chart, graph or grid. The controls in the export dialog also adapt to the currently selected object. If a chart is the current object the following export dialog will appear:



**Figure: Export dialog for charts**

The default size of the image that is about to be exported is the same as it appears on the screen. To change the size then either select a pre-defined paper size in the **Size** list or enter a size in pixels.

# Edit Table Data

## Introduction

The editing support in DbVisualizer Personal is used to insert, update or remove single rows in a database table. Editing is performed using two different editors:

- Inline Editor
- Form Editor

The Inline Editor is convenient in situations when fast edits of single columns need to be made. The Form Editor presents all columns in a form and some users prefer it since it is easier to get an overview of the data. The final SQL and the rules to determine the row that is affected are the same irrespective of which editor is used.

The following figure shows what buttons in the **Data** tab and in a result set grid that are used specifically for editing.

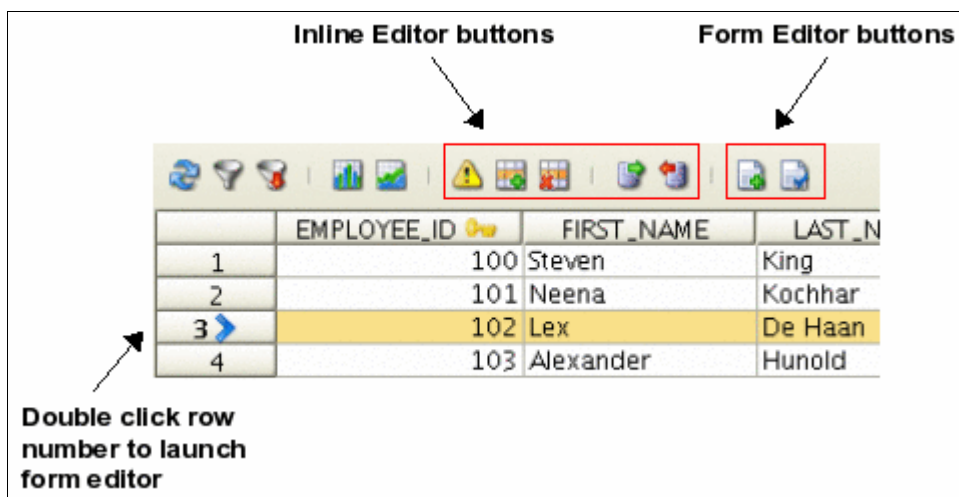


Figure: The buttons in the Data tab used to control the inline and form editors

## Features that support editing

Editing of table data can be performed in the **Database Objects->Data** tab or in the results from an SQL statement in the **SQL Commander**.

There are a few rules that must be fulfilled in order to enable editing in the SQL Commander:

1. It is a result set
2. The SQL is a SELECT command
3. Only one table is referenced in the FROM clause
4. All current columns exist by name (case sensitive) in the identified table

The editing tool bar is hidden if these rules are not met.

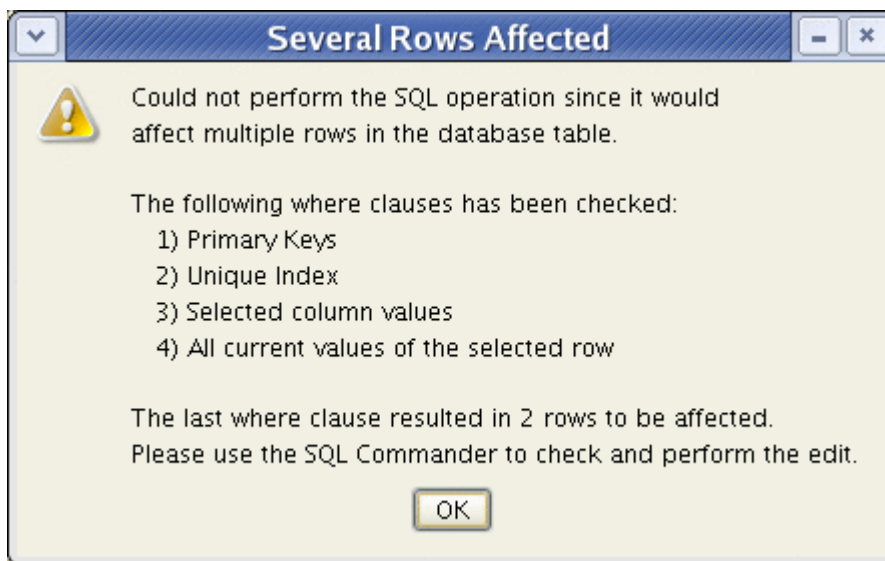
## Edits might be denied

The editing features in DbVisualizer ensure that only one row in the table will be affected by update and delete requests. This prevents the user from doing changes in one row that might also silently affect the data in other rows. DbVisualizer uses the following strategies to determine the uniqueness of the edited row:

1. Primary Key
2. Unique Index
3. Selected Columns
4. All Previous Values

The **Primary Key** concept is widely used in databases to uniquely identify the key columns in tables. If the table has a primary key then DbVisualizer will use it. There are situations when primary keys are not supported by a database or when primary keys are supported but not used. If no primary key is defined then DbVisualizer will check if there is a unique index. If there are several unique indices then DbVisualizer will pick one of them. So why not only try to identify the row by all its previous values? The reason **Selected Columns** is searched before checking **All Previous Values** is because not all data types are allowed to be used in a where clause. Typical examples are BLOB and CLOB data types. The **Selected Columns** check simply uses the selected columns in the inline editor or checked columns in the form editor to find out if the values identify a unique row. If that check fails then **All Previous Values** is searched.

The following dialog appears when none of these strategies can uniquely identify the row.



**Figure: Warning dialog when a unique row could not be identified**

## Commit

DbVisualizer issues an implicit database commit after each successful edit operation (the value of the **Auto Commit** property in Tool Properties is bypassed during editing). The commit request is per database connection and it commits all pending updates even those that have been executed in the SQL Commander.

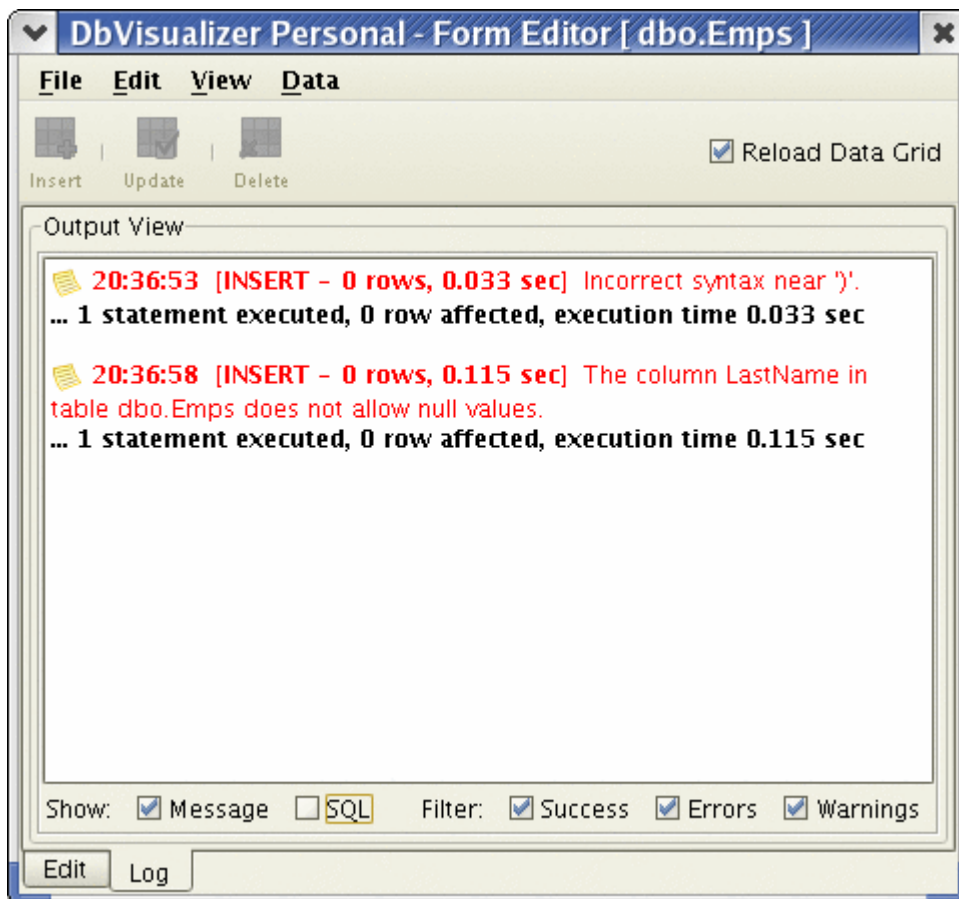
The following example explains the effect of the implicit commit:

1. The **Auto Commit** property is disabled in Tool Properties.
2. An SQL statement that deletes some rows is executed in the SQL Commander but it is not explicitly committed.
3. We now go to the Data tab for a table and edit the value of a column.
4. Once the update in the grid is performed the update is committed. This commit will also commit the delete statement that was executed earlier in the SQL Commander. The reason is that all commits are performed per Database Connection.

## Error Log

If the insert, update or delete request is successfully executed by the database then the row in the

grid will be updated accordingly. There are however situations when errors occur during the database execution phase. These messages are presented in the execution log.



**Figure: Error Log view**

The log keeps information of all edits that have been executed in both the inline and form editors. It presents not just failed operations (red color) but also all successful edits. All error messages that appear in the log are produced by either the JDBC driver or the database. Refer to the driver or database documentation for explanation of the error messages.

## Binary data/BLOB and CLOB

Binary data/BLOB and CLOB data can only be edited in the form editor. Read more about it in [Editing Binary/BLOB data and CLOB](#).

## Inline Editor

The inline editor is handy when fast edits need to be made. The editor is simple to use since it is activated by typing characters in the cell (column in a row) that is to be modified. The inline editor is data type aware and checks that the entered values are valid for each column's data type. Any errors are reported in the error log. Only those cells that have been modified (indicated by a yellow color) during the editing session will be propagated to the database.

The inline editor keeps all edits that have been made in a single row until the edit is implicitly committed by selecting another row in the grid or by using the update tool bar button (see below). All other operations such as selecting another table in the **Database Objects** tree will silently revert any edit.

The following tool bar buttons are used to control the inline editor:



**Figure: Tool bar buttons to control the inline editor**

Description of the buttons (from left):

- Confirm edits (on/off)  
(Check this button to make sure DbVisualizer always asks before performing any edits in the database table)
- Insert row
- Delete row
- Perform the current edit
- Revert the currently edited row

A row is edited by typing characters in a cell. All cells that have been edited are indicated by a yellow background color. Only these columns will be updated once the final SQL is executed to perform the change. All cells irrespective of whether they have been edited or not are highlighted with a yellow border to indicate which row is being edited.

*It is not possible to edit binary data, BLOB and CLOB values in the inline editor. Use the form editor to manipulate values for these data types.*

## Insert a new row

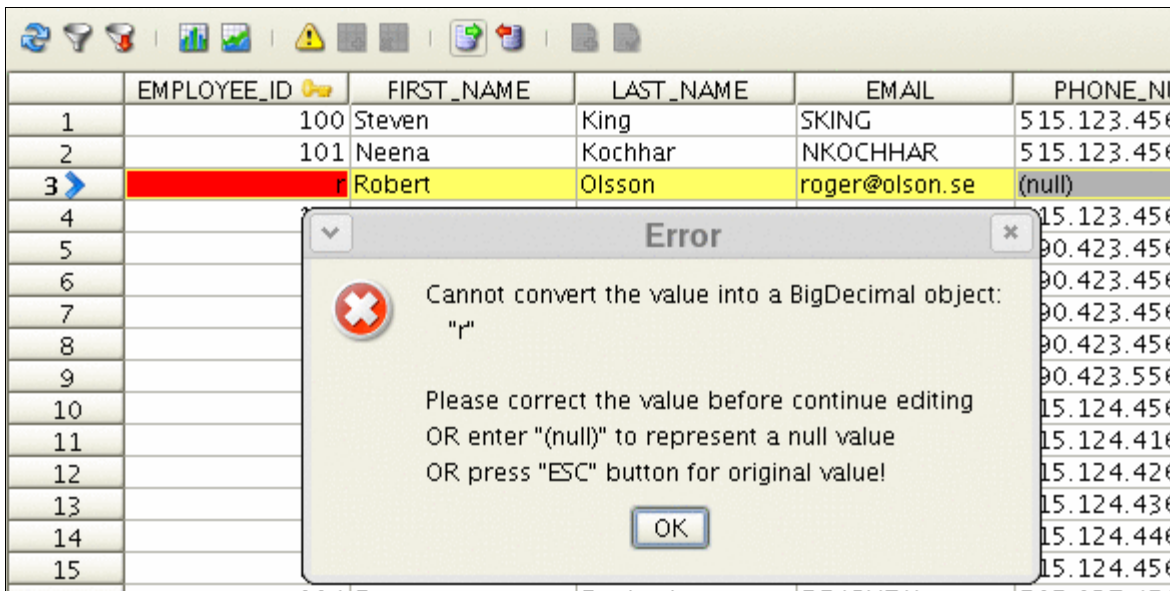
To insert a new row just press the **Insert row** tool bar button. All columns in the table that do not allow nulls are by default switched to the yellow background color. The default value for all columns is the character representation for null which is **(null)** by default. Now just start typing in the cells to set the values of the new row.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER
1	100	Steven	King	SKING	515.123.4567
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568
3	(null)	(null)	(null)	(null)	(null)
4	102	Lex	De Haan	LDEHAAN	515.123.4569
5	103	Alexander	Hunold	AHUNOLD	590.423.4567
6	104	Bruce	Ernst	BERNST	590.423.4568
7	105	David	Austin	DAUSTIN	590.423.4569

**Figure: Initial view of the inline editor when a new row is about to be edited**

If a value is entered that is not valid for a cell then an error dialog is displayed. The color of the invalid cell is at the same time switched to red.





**Figure: Error dialog when the data type is invalid for a column**

Follow the instructions in the dialog to either correct the value to match the data type of the column or revert to its original value.

If the newly added row is the only row in the table then there is no other row to select in order to perform the insertion. You must in this situation explicitly press the **Perform the current edit** button in the tool bar.

### Update an existing row

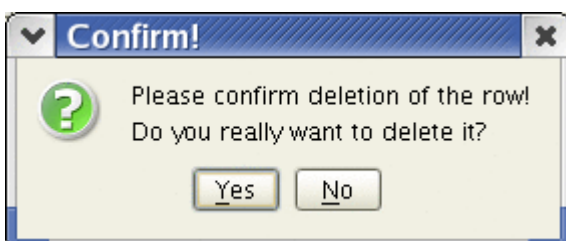
To update the value of a cell then just select the cell and start typing. The same checks as when inserting a new row are done here as well. To perform the edit just press the **Perform the current edit** button in the tool bar or select a cell in another row.

DbVisualizer use three different [strategies](#) to determine the row that will be updated. In order to let DbVisualizer use **Selected Columns** then first edit the cells that should be updated. Now select each cell that should be part of the selected columns list. Press the **Perform the current edit** button in the tool bar to let DbVisualizer use the selected columns in the final **where** clause.

### Delete a row

To delete a row then select at least one cell in the row to be removed. DbVisualizer will use one of the [strategies](#) in order to determine the row that will be deleted. Selecting one or several cells in the row will form the **where** clause that optionally will be used if the **Selected Columns** strategy is used to identify the row.

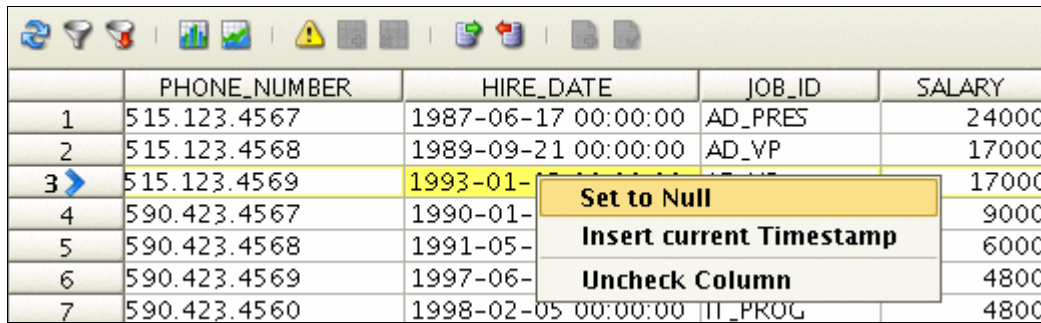
A confirmation dialog is displayed (the appearance of this dialog can be controlled in Tool Properties) in which the deletion must be confirmed.



**Figure: Confirmation dialog when deleting a row**

## Cell pop up menu

The cell pop up menu is active while editing a cell value and it is displayed using the right mouse button.



The screenshot shows a table with five columns: an index column, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, and SALARY. Row 3 is selected, and a context menu is open over the HIRE\_DATE cell. The menu items are 'Set to Null', 'Insert current Timestamp', and 'Uncheck Column'. The table data is as follows:

	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	515.123.4567	1987-06-17 00:00:00	AD_PRES	24000
2	515.123.4568	1989-09-21 00:00:00	AD_VP	17000
3	515.123.4569	1993-01-		17000
4	590.423.4567	1990-01-		9000
5	590.423.4568	1991-05-		6000
6	590.423.4569	1997-06-		4800
7	590.423.4560	1998-02-05 00:00:00	IT_PROG	4800

Figure: Cell pop up menu

The list of operations in the cell pop up menu is different depending on what data type the column is. Common for all data types are the **Set to Null** and **Uncheck Column** operations. The **Uncheck Column** is used to remove the editing state of the cell which means that it will not be included in the update of the row (the yellow background color is removed).

Remember that the format of time stamp, date or time values must match the format settings in Tool Properties.

**Insert current Timestamp** (can also be **Insert current Time** and **Insert current Date**) menu choice is valid only for the appropriate data types and simply inserts the current time stamp. The format of these values matches the format settings in Tool Properties.

## Form Editor

The Form Editor enables editing of data in a form based dialog. This editor is useful when inserting new rows and when it is important to get a more balanced overview of all the data. The form editor and the inline editor are based on the same set of rules and the visual appearance is similar.

The following tool bar buttons are used to start the form editor:



Figure: Tool bar buttons to control the form editor

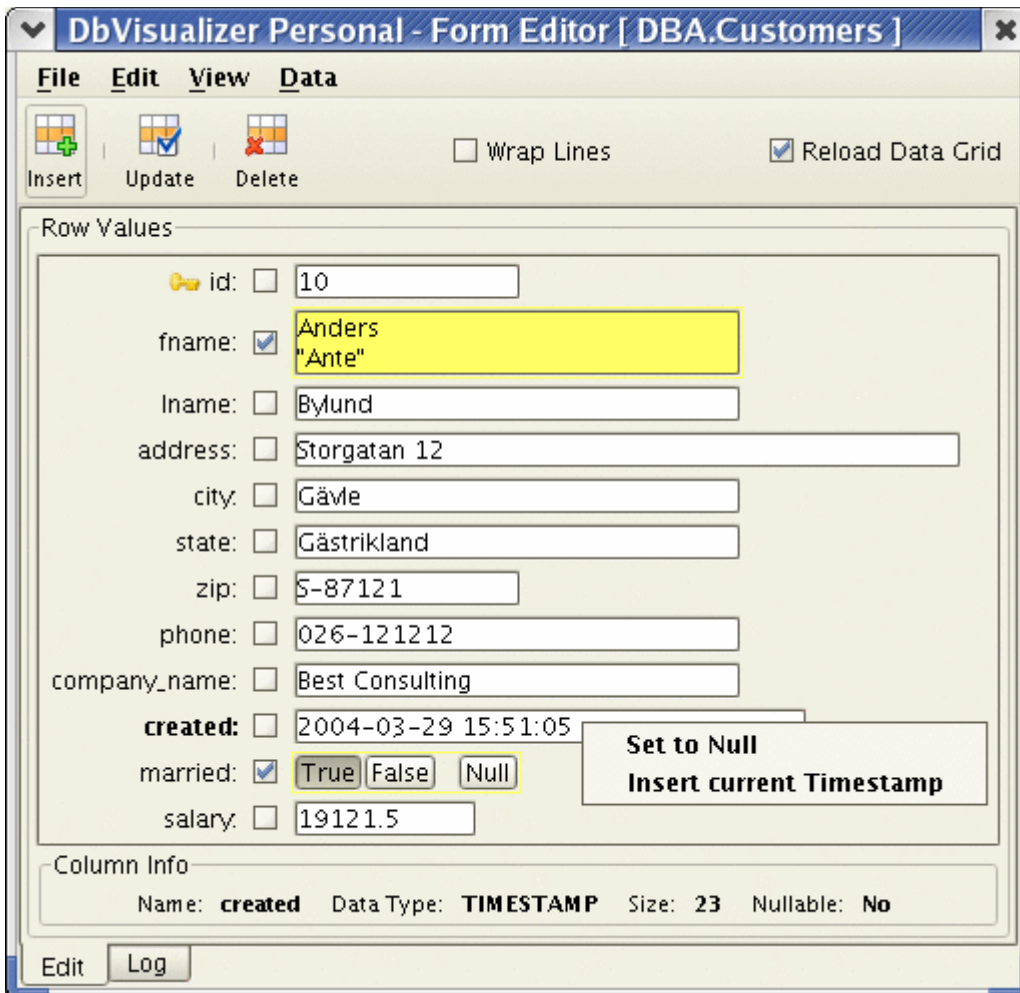
Description of the buttons (from left):

- Insert row
- Edit row (update, delete or insert copy)

An alternative way to start the form editor in edit mode is to double click on the row header.

## Form editor controls

The following explains how the form editor is organized and what controls are available.



**Figure: The form editor**

The form editor is composed of two main views (at the bottom of the dialog), they are the **Edit** and **Log** views. The edit view is as its name implies the place where the actual edits take place. The log view maintains a list of log messages for all edits that have been performed. The log view is displayed automatically if an error occurs during the database operation.

The tool bar buttons are used to **Insert**, **Update** and **Delete** the current row. (Only the insert button is enabled if the form editor has been launched to insert a new row).

The **Wrap Lines** check box is used to control whether long values should be wrapped automatically.

The **Reload Data Grid** check box determines whether the Data tab grid should be automatically reloaded once the form editor is closed. If this box is disabled then make sure to reload the data tab grid manually to reflect the changes that have been made in the form editor.

## Row Values

The **Row Values** section in the form editor lists all columns and values (fields) as they appeared in the data tab grid when the form editor was launched (i.e if a column was manually removed from the grid then it wont show in the form editor). Each field is composed of a label which is the column name and the value. Any primary key columns are indicated with a key-like icon to the left of the label.

The check box between the label and the value indicates whether the column will be part of the SQL that is finally executed to perform the edit. The checked state is automatically enabled if the field of the value is being edited and the color of the value field is automatically switched to yellow to indicate what fields will be part of the database request.

Additional information about a column is listed below the list of fields. This information is updated when the pointer is in a value field.

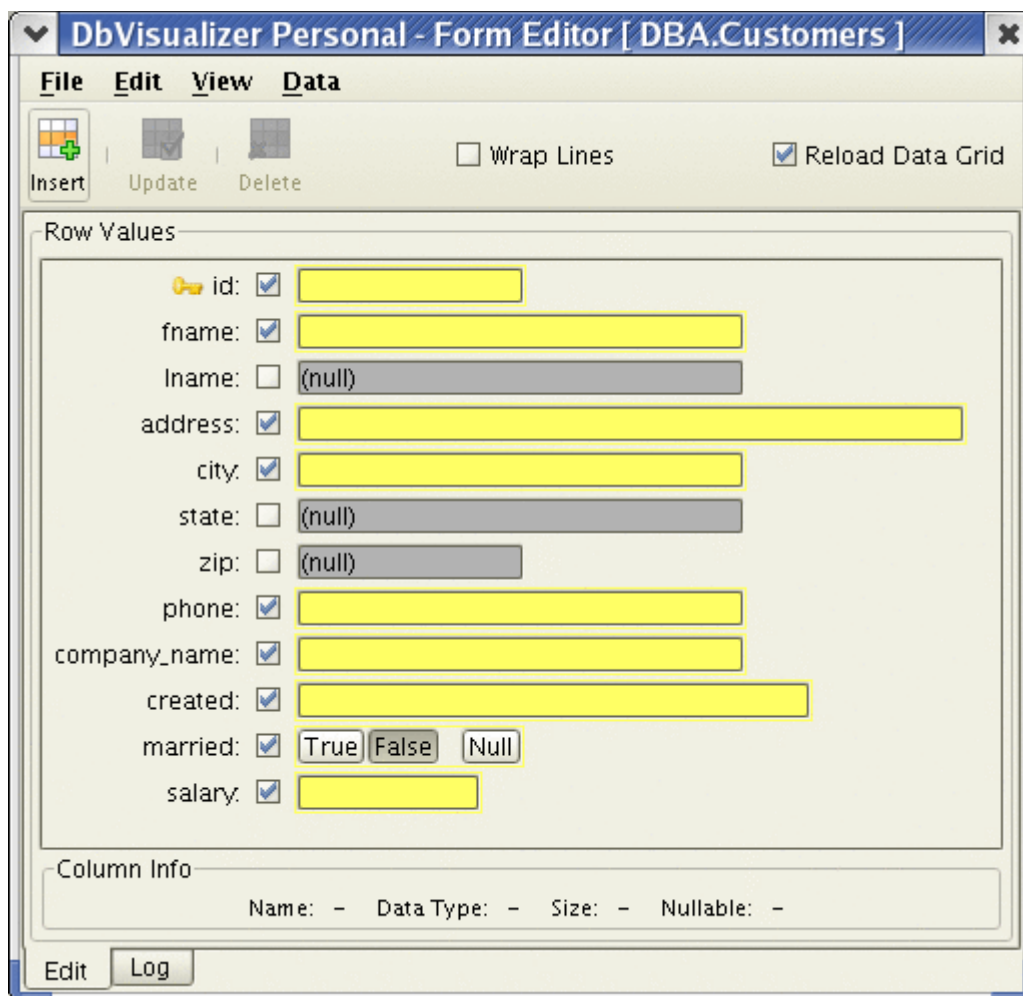
All text fields allow data to be entered in multiple lines (press the enter keyboard button). Boolean and Bit fields are managed by toggle buttons. The state of these fields can be either true, false or null. All fields except boolean fields have a pop up menu associated with them similar to the one in the inline editor. For all fields it contains the **Set to Null** operation. For time stamp, date and time columns it also contains **Insert current Timestamp**, **Insert current Time** or **Insert current Date**. The formats of time stamp, date and time are the same as specified in Tool Properties.

Remember to use the same format for time stamp, date and time fields as specified in the formats section in Tool Properties.

Each of the fields limits the amount of data that can be entered. Entering more characters than allowed will be denied. Some JDBC drivers report invalid column lengths such as 0 or a negative size. The form editor tries to figure out if the length is invalid and adjust the width accordingly.

## Insert a row

The following dialog is displayed when choosing to insert a new row using the Form Editor.



The screenshot shows the 'DbVisualizer Personal - Form Editor [ DBA.Customers ]' window. It features a menu bar with 'File', 'Edit', 'View', and 'Data'. Below the menu are three buttons: 'Insert' (with a grid icon), 'Update' (with a refresh icon), and 'Delete' (with a trash icon). To the right of these buttons are two checkboxes: 'Wrap Lines' (unchecked) and 'Reload Data Grid' (checked). The main area is titled 'Row Values' and contains a list of fields with checkboxes and input fields:

- id:  [Yellow input field]
- fname:  [Yellow input field]
- lname:  (null) [Grey input field]
- address:  [Yellow input field]
- city:  [Yellow input field]
- state:  (null) [Grey input field]
- zip:  (null) [Grey input field]
- phone:  [Yellow input field]
- company\_name:  [Yellow input field]
- created:  [Yellow input field]
- married:  [True] [False] [Null]
- salary:  [Yellow input field]

At the bottom, there is a 'Column Info' section with the following fields: Name: -, Data Type: -, Size: -, Nullable: -. At the very bottom are two buttons: 'Edit' and 'Log'.

**Figure: The form editor as it appears when a new row is about to be edited**

The form editor automatically enables the checked state for all columns that do not accept nulls. All other fields are unchecked with the value of (null) (or whatever the text representation of null has been set to in Tool Properties). Now specify the values accordingly and press the **Insert** button to perform the insertion. The form editor is closed if the execution was successful.

The **Data->Insert (Keep Window)** is an alternate insert operation that is used to insert the new row. The difference is that the form editor dialog is kept. This is useful if successive inserts need to be made.

## Edit a row (update, delete or insert copy)

Do the following in the Data tab grid to edit a row using the form editor:

1. Double click on the row header for the row that is to be edited
2. Select one or more cells in the row and press the **Edit row in a form** tool bar button

The first choice is useful if the table has primary keys or if the database table accepts an update request based on all current values for the row. The second choice is useful when the table has primary keys or when one must be able to select the columns (cells) that will form the **Selected Columns** where clause. See [Edits might be denied](#) for more information.

The screenshot shows the 'DbVisualizer Personal - Form Editor [ DBA.Customers ]' window. The title bar includes a dropdown arrow, the text 'DbVisualizer Personal - Form Editor [ DBA.Customers ]', and a close button. The menu bar contains 'File', 'Edit', 'View', and 'Data'. The toolbar has three icons: 'Insert' (a grid with a plus sign), 'Update' (a grid with a checkmark), and 'Delete' (a grid with an 'X'). To the right of the toolbar are two checkboxes: 'Wrap Lines' (unchecked) and 'Reload Data Grid' (checked). The main area is titled 'Row Values' and contains a form with the following fields: 'id' (with a key icon and a checkbox, value '10'), 'fname' (checkbox, value 'Anders "Ante"'), 'lname' (checkbox, value 'Bylund'), 'address' (checkbox, value 'Storgatan 12'), 'city' (checkbox, value 'Gävle'), 'state' (checkbox, value 'Gästrikland'), 'zip' (checkbox, value 'S-87121'), 'phone' (checkbox, value '026-121212'), 'company\_name' (checkbox, value 'Best Consulting'), 'created' (checkbox, value '2004-03-29 15:51:57'), 'married' (checkbox, with radio buttons for 'True', 'False', and 'Null'), and 'salary' (checkbox, value '19121.5'). At the bottom of the form area is a 'Column Info' section with fields for 'Name:', 'Data Type:', 'Size:', and 'Nullable:', all containing dashes. At the very bottom of the dialog are 'Edit' and 'Log' buttons.

**Figure: The form editor as it appears when edit of an existing row has been requested**

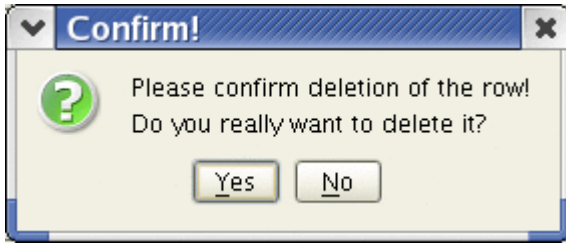
### Update the row

All fields are automatically unchecked when the form editor appears in edit mode. Change the desired values and press the **Update** button to perform the update request.

The form editor is closed if the update operation is successful.

## Delete the row

To delete the current row in the form editor just press the **Delete** tool bar button. A confirmation dialog might be displayed (the appearance of this dialog can be specified in Tool Properties) in which the deletion must be confirmed.



**Figure: Confirmation dialog when deleting a row**

The form editor will be closed if the delete operation is successful.

## Insert a copy of a row

The form editor allows a new row to be inserted based on the values that are currently in the editor. Make sure to set each fields check state or use the **Edit->Check All Values** menu choice to enable the checked state for all fields. Use the **Insert** tool bar button or **Data->Insert (Keep Window)** menu operation to perform the insertion.

## Import from File

The **File->Import From File** operation can be used to load any file into the currently selected field. Importing from file is exactly the same as manually entering the data i.e the max field sizes are considered and it might not be possible to load a file for which content does not fit into the actual field.

Imported binary data with a recognized binary viewer will be displayed accordingly.

## Export from File

The **File->Export To File** is used to export the content of any field including binary data to a file.

## Editing Binary data/BLOB and CLOB

There are a few constraints specifically for editing of BLOB and CLOB data types in DbVisualizer:

- They can only be edited in the form editor
- A primary key is recommended to successfully update these data types
- **Note:** A primary key is **required** for update of BLOB and CLOB in **Oracle**

**Binary data** in DbVisualizer is the generic term for several common binary database types:

- LONGVARBINARY
- BINARY
- VARBINARY
- BLOB

Read the following sections about CLOB and Binary/BLOB data for specific information

## CLOB

CLOB data appears (apart from other data types) in a multi line text field in the form editor. Data can be entered manually or imported using the **File->Import From File** operation.

```
1 The editing support in DbVisualizer Personal is used to insert, update or remove single row
2
3 * Inline Editor
4 * Form Editor
5
6 The Inline Editor is convenient in situations when fast edits of single columns shall be ma
7 Features that supports editing
8 Editing of table data can be performed in the Database Objects->Data tab or in any results
9
10 There are a few rules that must be fulfilled in order to enable editing in the SQL Commande
11
12 1. It is a result set
13 2. The SQL is a SELECT command
14 3. Only one table is referenced in the FROM clause
15 4. That all current columns exist by name (case sensitive) in the identified table
16
17 The editing toolbar is hidden if these rules are not matched.
```

CV:

17:62 INS

Figure: The CLOB text editor

## Binary data/BLOB

Binary data data can by its nature not be manually edited in DbVisualizer, it can only be imported from a file.

The form editor recognizes some common formats and presents them in an appropriate viewer.

### GIF, JPEG and PNG viewer



Figure: Binary data viewer for common image formats

## Serialized Java objects viewer

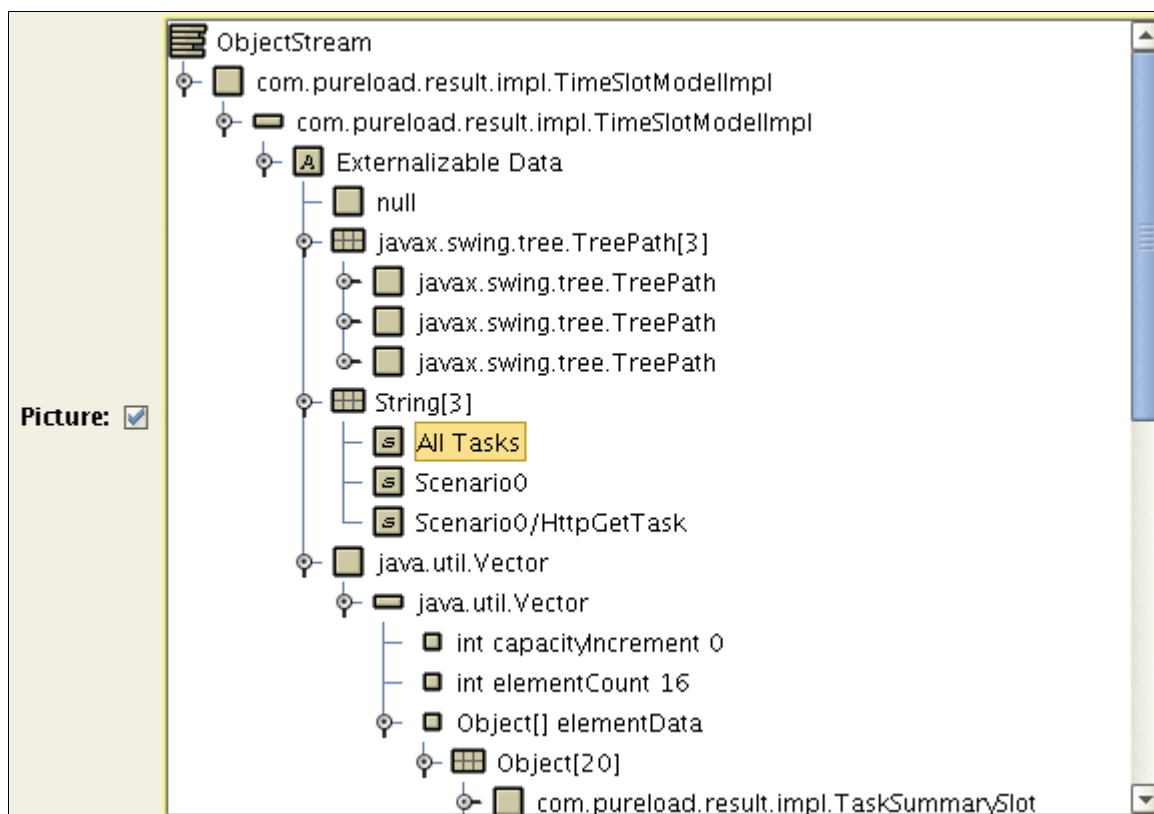


Figure: Binary data viewer for serialized Java objects

## Hex/Ascii viewer

The generic hex/ascii viewer is used if the data format is not recognized.

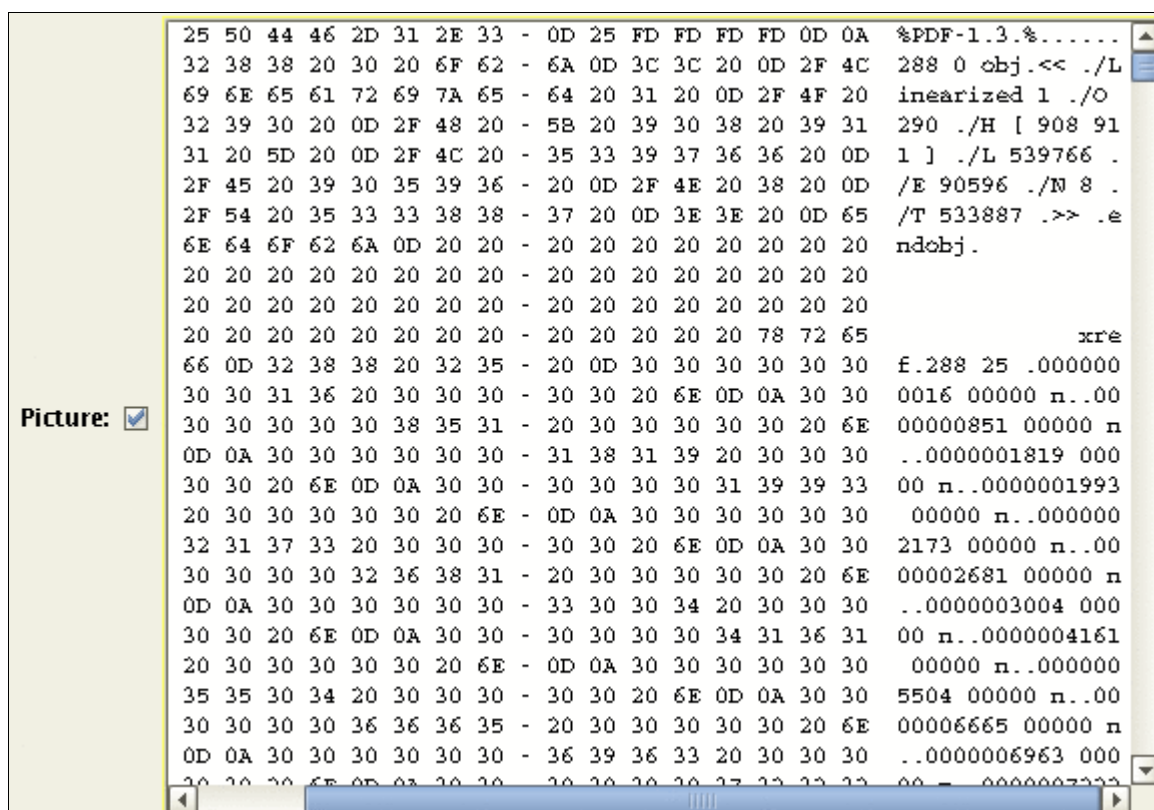


Figure: Hex/Ascii viewer for unrecognized binary data



# Create Table and Index Assistants

## Introduction

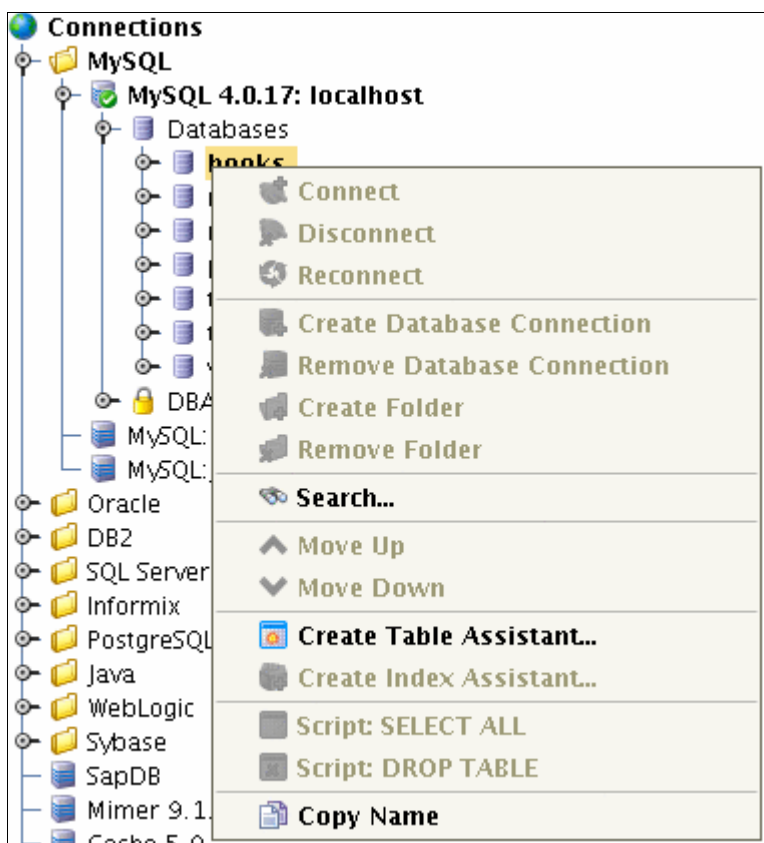
The Create Table and Index Assistants are used to create new tables and indexes. The assistants are quite simple to use since they examine various meta data in the database (depending on what assistant is used) and then let the user point and click to define the actual table or index. Both assistants finally generate the appropriate SQL and pass it over to the SQL Commander which is used to execute it.

These utilities are launched from the **Database** main menu or in the **Database Objects** tree right click menu. The menu choices are enabled only if a table or index can be created for the selected node in the Database Objects tree.

**Note:** Remember to manually select the **Reload** right click menu choice in the Database Objects tree once a new table has been created.

## Create Table

Start the table creation assistant by choosing the **Database->Create Table** menu choice or the **Create Table** menu choice in the Database Objects tree right click menu. Make sure you have located and selected the appropriate object in the **Database Objects** tree as this selection is used to define which database, schema, etc. the table will be created in.



**Figure:** The right click menu in the Database Objects tree

The Create Table assistant is organized in three areas from the top:

- **Table Info**

Specifies the owning database connection, database and/or schema. These are picked up

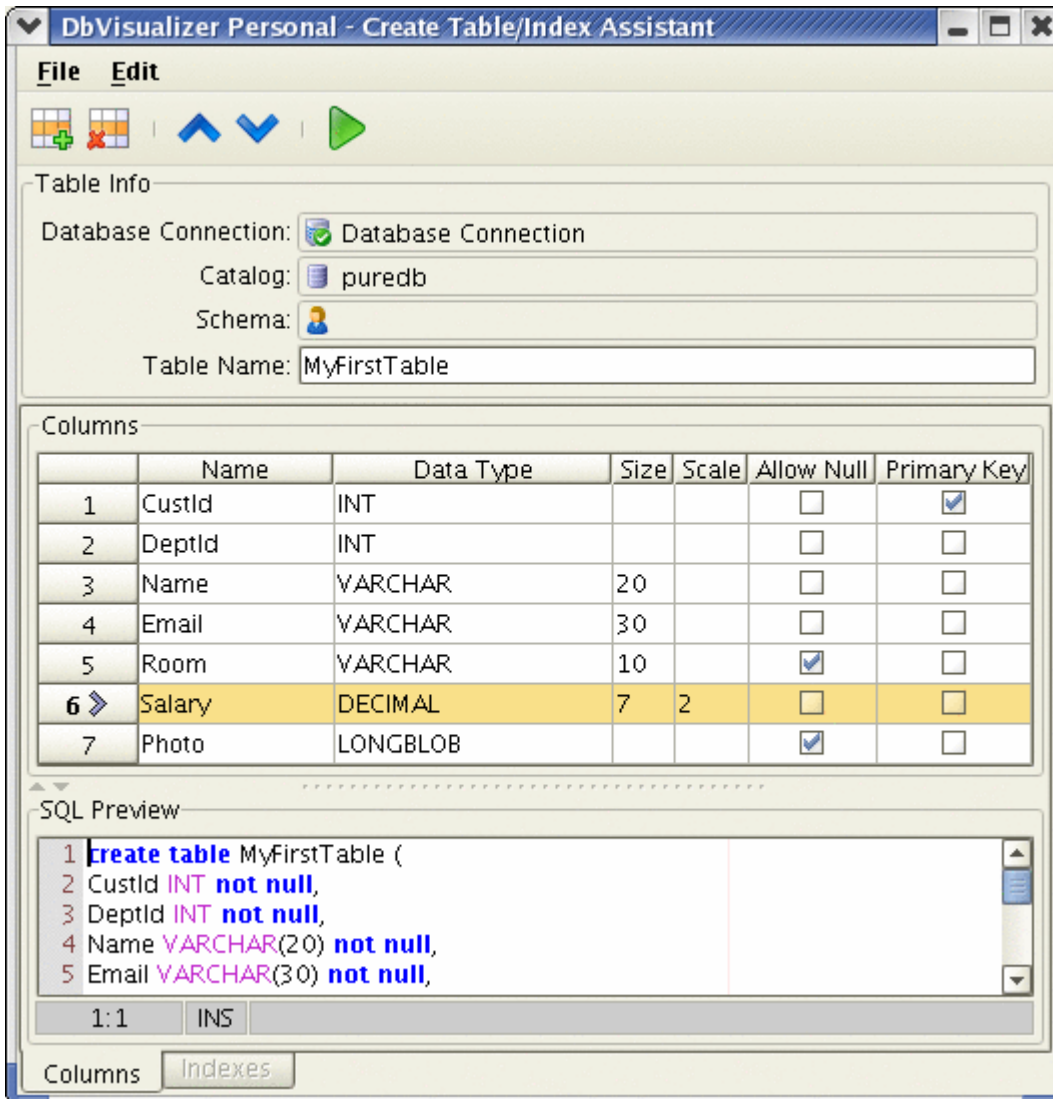
from the selection in the tree when the assistant is started. Table name is empty and must be specified.

- **Columns**

The list is used to organize the columns that will be in the final create statement. The Data Type column is a list of supported data types for the actual database. Columns can be re-ordered using the main controls.

- **SQL Preview**

The SQL previewer instantly shows the SQL that is used to create the table.



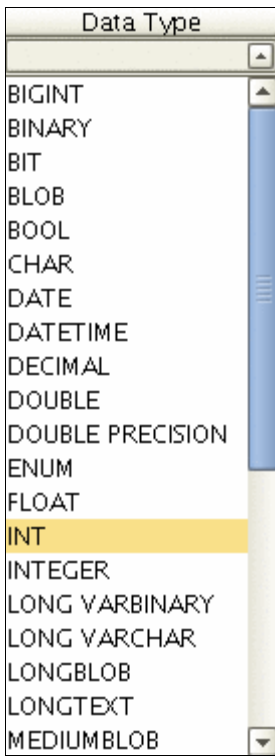
**Figure: The table assistant**

## Columns

Add columns by using the **Edit->Insert** menu choice and **Edit->Delete** to remove the currently selected row. They can be re-organized using the **Edit->Move Up** and **Edit->Move Down** menu operations.

The assistant is based on generic JDBC and it is the responsibility of the user to enter the required fields i.e. specifying size for text data types, ignore size for some BLOB types, enter scale for decimal types, etc.

The **Data Type** field for a column when selected contains a list of valid data types for the actual database connection.



**Figure: Data Type list (for MySQL)**

The **Size** and **Scale** fields are used either in conjunction or else the size field on its own. Size is often used to set the max length of text columns. It is also used in combination with the Scale field when defining decimal boundaries.

5	Room	VARCHAR	10		<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	Salary	DECIMAL	7	2	<input type="checkbox"/>	<input type="checkbox"/>
7	Photo	LONGBLOB			<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Figure: Size and scale for a DECIMAL data type**

The above example will allow a total length (including the decimal places) of 7. Examples:

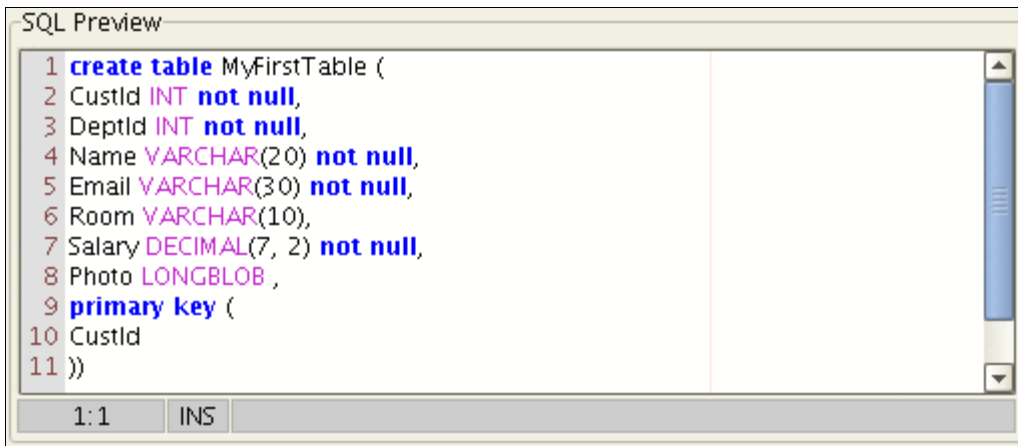
```

1.02
9871.1
8172.0
18291.22
12.112 <- Error
1921211.11 <- Error

```

## SQL Preview

The SQL Preview area is updated automatically to match the edits made in the assistant. The preview is read only.



```
1 create table MyFirstTable (  
2 CustId INT not null,  
3 DeptId INT not null,  
4 Name VARCHAR(20) not null,  
5 Email VARCHAR(30) not null,  
6 Room VARCHAR(10),  
7 Salary DECIMAL(7, 2) not null,  
8 Photo LONGBLOB ,  
9 primary key (  
10 CustId  
11 ))
```

**Figure: The SQL Preview for a table**

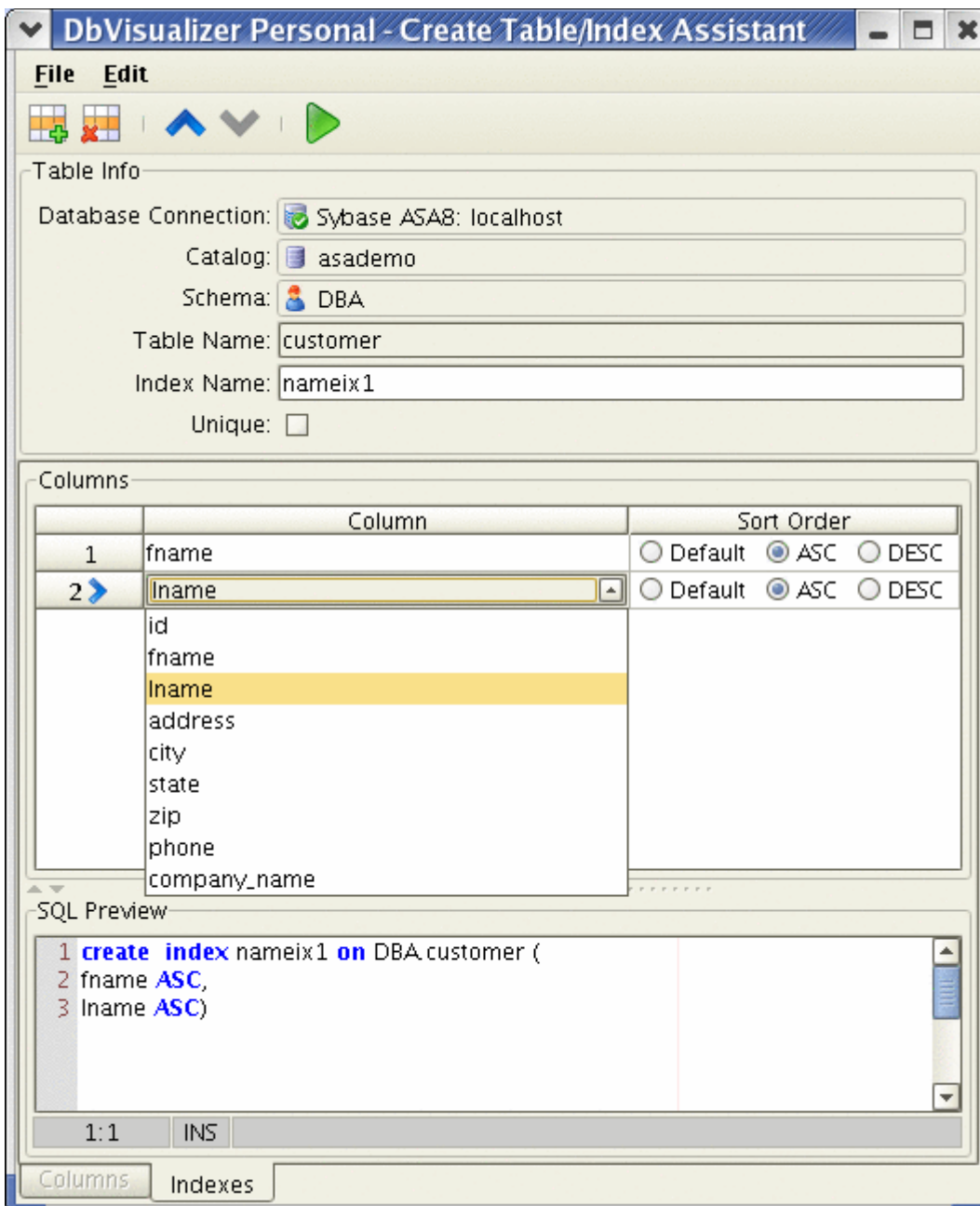
## Execute

When you are satisfied with the table then choose **File->Pass SQL to SQL Commander** menu choice. The SQL is then inserted into the SQL Commander. Now select **Database->Execute** main menu choice to execute it as per any regular SQL. If any errors appear during the execution then go back to the Table Assistant and make the necessary changes.

**Note:** If you want to refine the setup of a table then just select the already visible Table Assistant window. Do not choose the Create Table menu choice again since it will then clear the assistant from its current setup.

## Create Index

The Create Index Assistant is much the same as the Create Table Assistant. It is launched from the same menus and the overall layout and controls are the same. The only difference is the **Columns** list which now lists the columns that will be part of the index. The **Column** field when selected lists the current columns in the table.



**Figure: The Index creation assistant**

The example shows that a new **unique** index, **UniqueName** will be created for the **MyFirstTable**. It will index the **Name** column in **Ascending** order.

Please read the previous sections on how to use the Table assistant to edit and create the actual index.

# SQL Bookmarks

## Introduction

The SQL Bookmark feature was one of the original ideas when DbVisualizer was first designed. The background is that most SQL tools offer quite good support to edit and test SQL statements. But when the final optimized and really awesome SQL is complete then you're all alone. (We used to use a standard text editor and the famous copy/paste capability to transfer SQL's from and to SQL\*Plus ;-)

The key concept behind the bookmark management is to offer a way to save SQL statements between invocations of DbVisualizer and make it easy to execute them. Another important requirement is to organize SQL statements in folders for structural and grouping purposes. The core of the bookmark management is the Bookmark Editor. It is here the bookmarks are organized.

The bookmark editor depends heavily on the SQL Commander since when requesting to execute an SQL Bookmark the bookmark editor will pass the actual SQL along with the connection data to the SQL Commander. It is then the SQL commander that is used to edit and test the SQL until it is complete.

## What's a bookmark in DbVisualizer?

An SQL Bookmark is generally an SQL statement that is saved between invocations of DbVisualizer. In addition it also keeps related information needed to execute the SQL and present the result accordingly once it is requested.

- SQL statement
- Bookmark name
- Database Connection
- Catalog (aka Database)
- Chart settings (optional)

The bookmark management is primarily used to save SQL statements that are used often or for whatever reason there might be. There are different types of bookmarks and DbVisualizer automatically creates bookmarks in the following function areas:

- Each SQL that is executed in the SQL Commander is saved as an SQL bookmark in the History folder
- Each monitored SQL statement in the Monitor feature is an SQL bookmark and is added to the New folder

(Read more about this in the following sections).

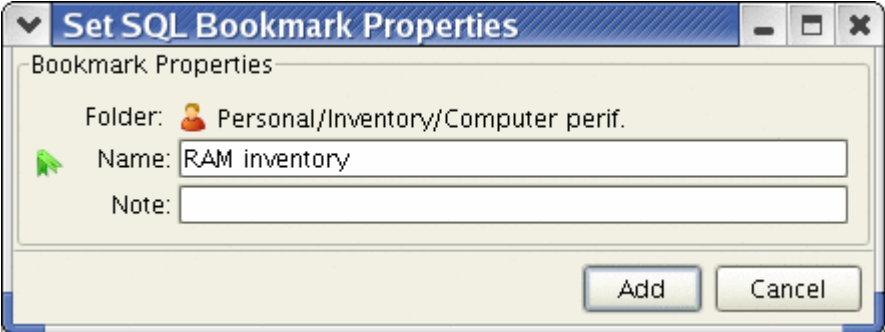

## The Bookmarks Main Menu

The bookmarks main menu in the DbVisualizer window contains the following choices:



Figure: The Bookmark main menu

All except the Bookmark Editor choice are disabled if you are not in the SQL Commander tab.

Menu Choice	Description
<b>Bookmark Editor</b>	Requests to display the <b>Bookmark Editor</b> .
<b>Add Bookmark to Folder</b>	<p>This is composed of a sub menu in which all folders are displayed. This list displays the paths for all folders (i.e the folder hierarchy from the root). The root folders are <b>Personal, New</b> or <b>History</b> (read more about these in the sections below). Once a folder has been selected the following dialog is displayed. Here you can change the default name and add an optional note.</p> 
<b>Replace Bookmark</b>	<p>This option is used to replace the chosen SQL bookmark with the SQL and connection data that is in the current SQL Commander editor. The replace bookmark menu consists of the root folders and last in the menu possibly the name of the last SQL bookmark that was passed from the bookmarks editor. If you want to replace the data for that SQL Bookmark just select its name in the menu.</p> 
<b>Get Bookmark</b>	Get Bookmark shows the same menu hierarchy as Replace Bookmark except that it is used to fetch the chosen SQL Bookmark and insert it into the current SQL Commander editor.

## Bookmark Editor

The bookmark editor is the core of the bookmark management and is used to organize SQL bookmarks in folders and to do various adjustments.

### Bookmark list

The editor is based on a tree list with the same structure as the tree that appears in the **Bookmarks** main menu options. The tree has three root folders that cannot be changed, moved or removed. There is basically no difference between these root folders except that they are used in different contexts in DbVisualizer.

- **Personal**

This root folder is supposed to hold the structure of favorite SQL bookmarks. By putting SQL bookmarks in folders you get a better organisation and overview of your bookmarks. All nodes in the root folder are manually maintained.

- **New**

Creating **Row Count Monitors** in the **Database Objects->Data** tab will add these monitors (as SQL Bookmarks) to the **New** root folder.

- **History**

All SQL statements or scripts that are executed in the SQL Commander are automatically added in the **History** root folder. The latest executed statement appear first in the list.

(The number after the root folder names indicates the number of SQL bookmarks that are in that root folder).



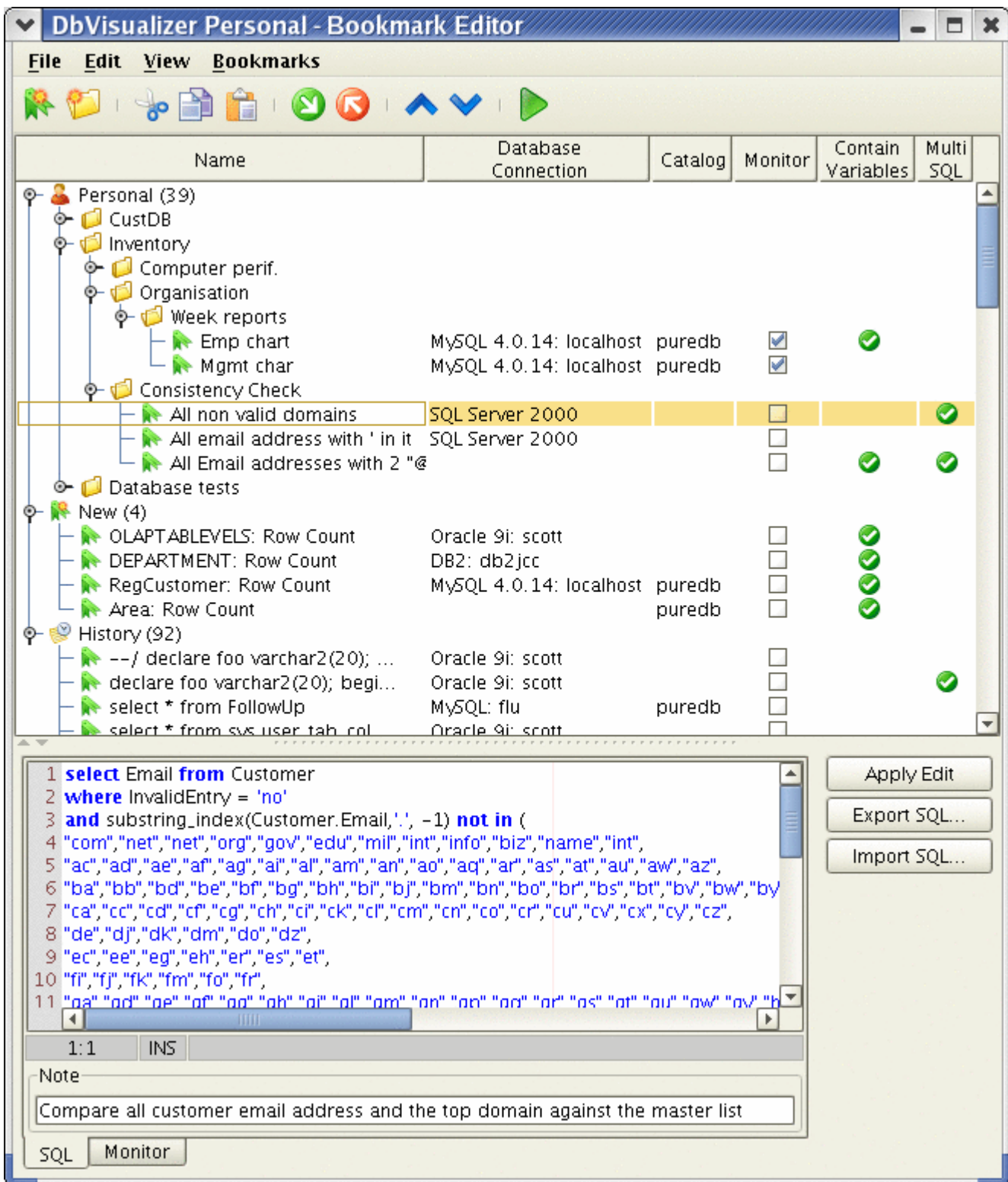


Figure: The Bookmark Editor

You cannot create folders or SQL bookmarks in the **New** or **History** root folders. The way to work with these folders is to copy the SQL bookmarks you want from them into the appropriate location in the Personal root folder.

The tree of folders and SQL bookmarks is contains the following information:

Column in list	Description
----------------	-------------

<b>Name</b>	The name of the node (folder or SQL bookmark). Modify the name by selecting the column and click once to get into editor mode. The <b>Edit-&gt;Change Name</b> menu choice can be used for the same purpose. If a SQL bookmark was created by some other function in DbVisualizer then the name will be the first 40 characters of the SQL statement.
<b>Database Connection</b>	The database connection column when clicked displays a list of all defined database connections. The list indicates whether a connection is established or closed. It is here you specify using another database connection for an SQL bookmark.
<b>Catalog</b>	This column lists the Catalog (aka Database) that was current when the bookmark was created. You can change the Catalog by clicking in it. A list of accessible catalogs is then displayed.  <b>Note:</b> The list of catalogs is empty if the Database Connection is not established.
<b>Monitor</b>	Check this box to enable the SQL bookmark to become a monitor and thereby appear in the Monitor main tab. SQL's that returns results are the most obvious candidates for being monitored.
<b>Contain Variables</b>	This column is read only and indicates whether the SQL statement includes any DbVisualizer variables. (I.e <code>\$\$variable name\$\$</code> )
<b>Multi SQL</b>	This column is also read only and indicates whether the SQL statement is composed of several SQL statements (aka script). This is determined by looking for statement delimiters in the SQL.

## New and History root folders

The number of SQL bookmarks that may be added by DbVisualizer to the **New** and **History** root folders are specified in the **Tool Properties->Bookmarks** category. Bookmarks in these folders can be removed one by one or each folder can be cleared using the **File->Clear all New entries** or **File->Clear all History** entries.

## SQL Editor

### Monitor information

The monitor sub tab controls the total number of rows that will be kept in the result grid until rows are automatically removed. This feature is specific to the monitor feature. Please see [Charts and Monitors](#) for more information.

### The Note field

The note field can be used to write a short note about the SQL Bookmark. This note will appear as a tool tip in the Bookmarks main menu.

## Executing an SQL bookmark or folder of SQL bookmarks

The SQL editor in the bookmark editor can be used to modify the SQL but it is not the place to execute SQL statements. Instead it is the SQL Commander that is used to execute SQL's. Select the **Edit->Execute** menu operation to copy the selected SQL bookmark into the SQL Commander. The SQL Commander is then used to execute and edit the SQL. Once you are satisfied with it then select the last entry in the Bookmarks menu:



Figure: The Bookmark->Replace sub menu

If the last entry displays "No current bookmark" then it indicates that the currently edited SQL was not passed from the Bookmark Editor. You can use the other menu choices to locate the actual bookmark that will be replaced.

The **Execute** operation can also operate on a folder. Doing this will result in a script of all direct child SQL bookmarks that are located in that folder. Each of the SQL statements will be delimited by the delimiter as specified in Tool Properties.

Name	Database Connection	Catalog	Monitor	Contain Variables	Multi SQL
Personal (44)					
CustDB					
Inventory					
Add users					
Drop table	DB2: db2jcc		<input type="checkbox"/>		
Create table	DB2: db2jcc		<input type="checkbox"/>		
Insert master	DB2: db2jcc		<input type="checkbox"/>		
Insert master	DB2: db2jcc		<input type="checkbox"/>		
Insert master	DB2: db2jcc		<input type="checkbox"/>		
Computer perif.					
Organisation					
Consistency Check					
Database tests					
New (4)					
History (92)					

Figure: Selecting a folder for execution

Once the **Execute** is selected the following will appear in the SQL Commander editor:

```

1 // Drop the table
2 drop table Employee;
3
4 -- Create the Employee table
5 create table Employee (id int, name varchar(20), primary key (id));
6
7 // Insert master record
8 insert into Employee (id, name) (1, "ROOT");
9
10 // Insert simple record
11 insert into Employee (id, name) (2, "Nisse");
12
13 // Insert master another
14 insert into Employee (id, name) (3, "Pekka");
15

```

Figure: The SQL Commander editor

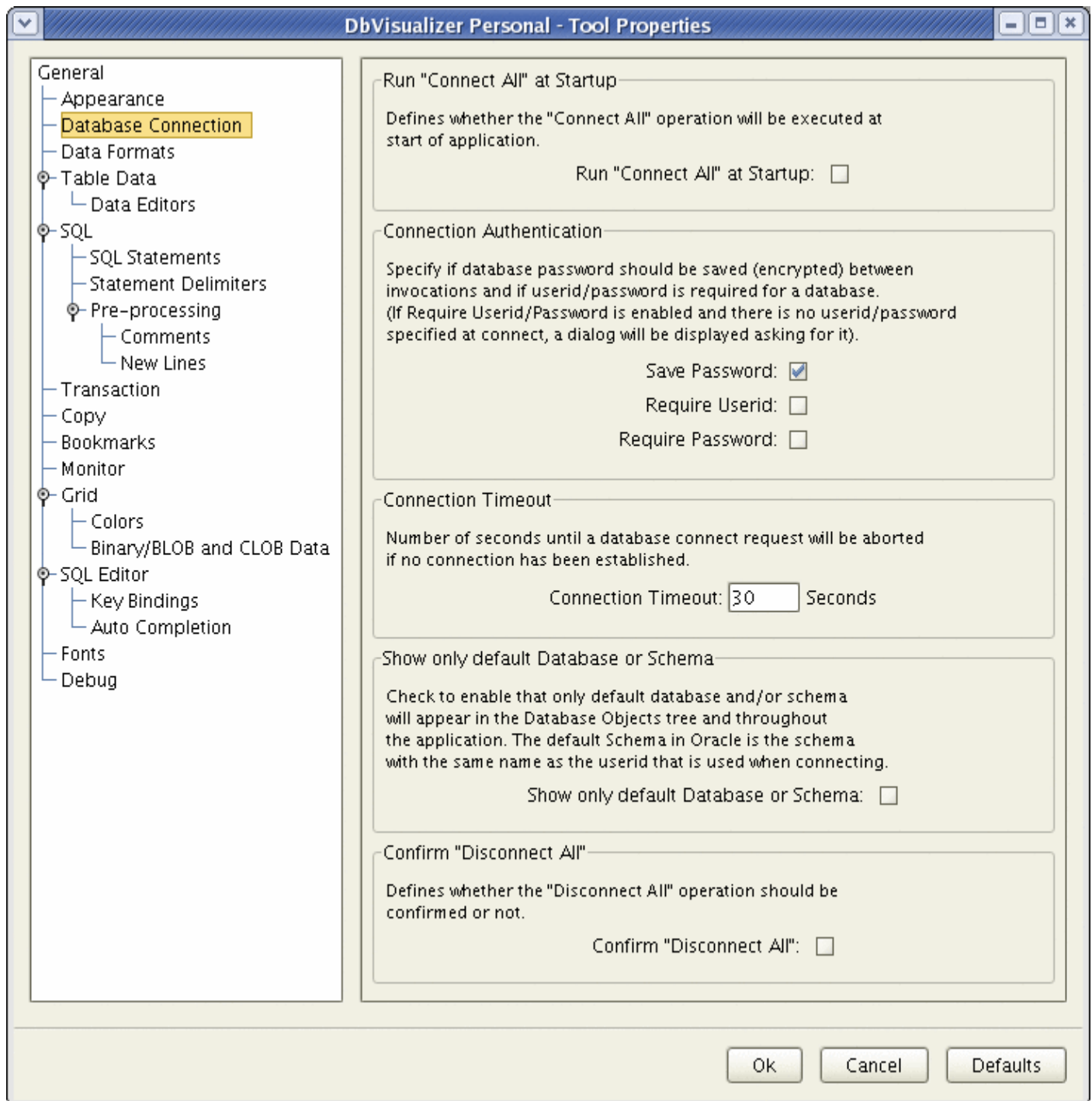
Note that the **Database Connection** and **Catalog** lists are empty. You need to select these from the lists when a script of SQL bookmarks is passed from the Bookmark Editor.

# Tool Properties

## Customizing DbVisualizer

DbVisualizer is highly customizable, you can control formatting, layout and the way DbVisualizer interacts with databases. The default settings are good enough for the average user but sometimes it is necessary to modify properties. This chapter guides you through all the properties.

**Note:** Not all properties that are saved between invocations are configurable through the tool properties dialog since these are instead adjusted in the appropriate places in the application.



**Figure: The Tool Properties window**

The Tool Properties window is organized as in many other applications. The **category** list in the left area and the **properties** for the selected category in the right area. A category may group one or

several properties.

The buttons at the bottom of the window control whether the changed properties should be applied using the **Ok** button, if changes should be reverted using the **Cancel** button or if the factory defaults should be applied using the **Defaults** button.

Note that the **Ok** and **Defaults** buttons are used to apply values for all categories and not only the currently selected one.

## The user preferences (XML) file

All properties are saved in an XML file. The exact location of this file is platform dependent. The location on your system is listed in the frame header of the main window. The XML file contains, in addition to all properties, also the information about drivers, database connections, bookmarks, etc. The general recommendation is to **not edit** this file manually even though it is quite easy to do so.

DbVisualizer automatically creates a backup copy of the XML file when the application is started. The location of this file is the same as for the standard XML file except that the **.bak** suffix is appended to the file name. The standard XML file might get broken for various reasons. If a warning message that the XML file could not be read is displayed during launch of DbVisualizer then simply copy the backup file to the standard location and restart the application. If the XML file is moved from its standard location or if it is removed then DbVisualizer will automatically create a new one.

**Tip:** the **-up** command line argument is used to identify the file name (and path) to an alternate XML file.

## Categories

The following sections describes each property.

### Appearance

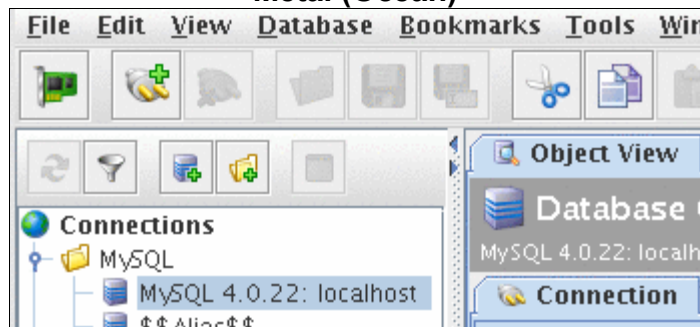
Property	Description
----------	-------------

Controls which look and feel will be used.

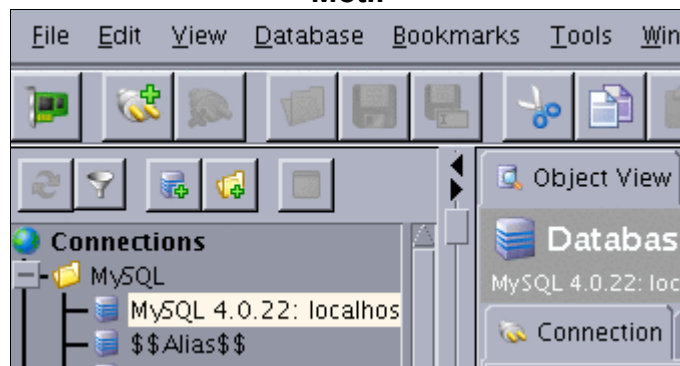
**Note 1:** You must restart DbVisualizer in order to use a new look and feel.

**Note 2:** Some look and feels are platform specific and do not appear on all OS'es

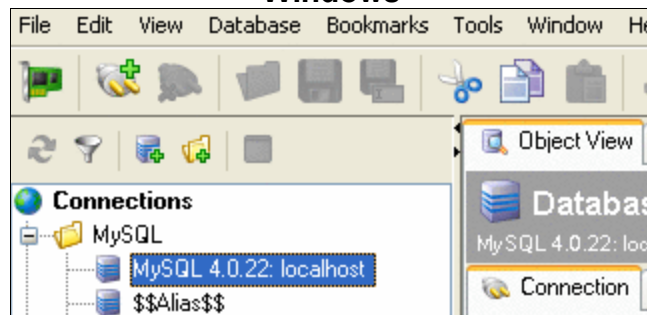
### Metal (Ocean)



### Motif

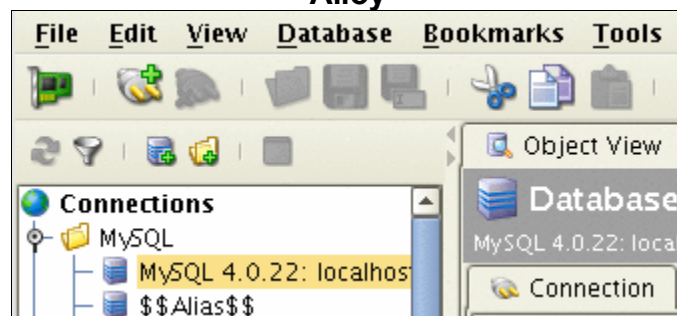


### Windows



Look and Feel

### Alloy



### GTK+



<b>Icon Sizes</b>	The <b>Menus, Main Tool Bars, Sub Tool Bars</b> settings are used to control the size of the icons.
<b>Show Tab Icons</b>	Specifies whether an icon will appear in the header of all tabs.

## Database Connection

Property	Description
<b>Run "Connect All" at Startup</b>	Defines whether a database connection will be connected when the <b>Connect All</b> operation is selected in the main window menu bar.
<b>Connection Authentication</b>	<b>Save Password</b> specifies if connection passwords should be saved between invocations of the application. (Passwords are encrypted). <b>Require Userid</b> specifies if DbVisualizer should ask for userid when connecting a database connection. (DbVisualizer will only ask if there is no userid entered already). <b>Require Password</b> specifies if DbVisualizer should ask for password when connecting a database connection. (DbVisualizer will only ask if there is no password entered already).
<b>Connection Timeout</b>	Specify number of seconds that the driver will wait until terminating ongoing connection request. <b>Note:</b> This property is handled by JDBC drivers and might not be supported.
<b>Show only default Database or Schema</b>	Check to enable that only default database and/or schemas will appear in the Database Objects tree and throughout the application. The default Schema in Oracle is the schema with the same name as the userid that is used when connecting.
<b>Confirm "Disconnect All"</b>	Checking this property will force a dialog to be displayed before disconnecting all current database connections using the Disconnect All operation.

## Data Formats

Property	Description
<b>Date Format</b>	Select the date format that will be used throughout the application (i.e grids, forms and during editing). <a href="#">More information below.</a>
<b>Time Format</b>	Select the time format that will be used throughout the application (i.e grids, forms and during editing). <a href="#">More information below.</a>
<b>Timestamp Format</b>	Select the timestamp format that will be used throughout the application (i.e grids, forms and during editing). <a href="#">More information below.</a>
<b>Format Numbers</b>	Specifies how numbers will be formatted.
<b>Null String</b>	This is the string representation of the null value. This string is the readable form of null and appears in grids, forms, exports and during editing.

### Date, Time and Timestamp formats

The lists for date, time and timestamp format contains a collection of standard formats. If these formats are not suitable then you can enter your own format in the appropriate field. The tokens used to define the format is listed in the right click menu while the field has focus.



**G - Era Designator**  
**y - Year**  
**M - Month in year**  
**w - Week in year**  
**W - Week in month**  
**D - Day in year**  
**d - Day in month**  
**F - Day of week in month**  
**E - Day in week**  
**H - AM/PM marker**  
**H - Hour in day (0-23)**  
**k - Hour in day (1-24)**  
**K - Hour in AM/PM (0-11)**  
**h - Hour in AM/PM (1-12)**  
**m - Minute in hour**  
**s - Second in minute**  
**S - Millisecond**  
**z - Time zone**  
**Z - Time zone**

**Figure: The date and time right click menu**

The complete documentation for these tokens are listed in the following web page [SimpleDateFormat](#).

## Table Data

Property	Description
<b>Show Table Row Count</b>	Specifies if the number of rows in a table will be displayed in the header of the table when in the Database Objects->Data tab. Enabling this property will cause an extra round trip to the database (i.e minor performance penalty)
<b>Highlight Primary Key Columns</b>	Specifies if Primary Key columns will be indicated in the Database Objects->Data tab, Variable Substitution dialog, SQL Commander Result grids and in the References Graph.
<b>Confirm Data Deletion</b>	Check to enable the Confirm Delete dialog when removing rows in the Inline and Form based editors
<b>Include Variables in SQL</b>	Specifies if the right click menu operations in the Data tab will create appropriate SQL statements that include DbVisualizer variables or if the generated statements are plain SQL. Letting DbVisualizer generating statements with variables results in the variable substitution dialog being displayed when these statements are executed in the SQL Commander.
<b>Max Rows at First Display</b>	Set the number of rows that will be fetched for a table in the Data tab when a table is first displayed.

## Data Editors

Property	Description
----------	-------------

<b>Confirm Data Deletion</b>	Check to enable the Confirm Delete dialog when removing rows in the Inline and Form based editors
<b>Reload Grid after Edit</b>	Check this to enable auto reloading of the grid after a successful edit in the inline editor.

## SQL

The SQL category groups properties that are related to execution of SQL statements and the processing associated with it.

### SQL Statements

This category controls the SQL templates that DbVisualizer uses internally throughout the application. Each SQL template is composed of the standard SQL and the DbVisualizer **Variable** notation. A variable is identified by the **Variable Identifier** and can be composed of several parts each delimited by the **Variable Delimiter**. DbVisualizer relies on a list of pre-defined variable names that are accessed in the **SQL Statement Template** right click menu:

catalog
catalogseparator
schema
schemaseparator
table
table-name
where-columns
quoted-where-columns
columns
values
column-values
create-columns
index-type
index
unique
index-columns
create-primary-key
DbVis-Date
DbVis-Time

**Figure: All pre-defined variables**

A specific pre-defined variable can be used in on or more of the SQL templates. Using a variable in a SQL statement that is not valid will result in the variable appearing as is once the statement is executed.

There is normally no reason to modify the SQL templates nor the variable identifier or delimiter settings. There might however be circumstances when edits are needed:

- To put quotes or brackets around table names
- To change the variable identifier or variable delimiter since the default settings may interfere with object names in the database
- To modify the appearance of the where clause or the list of columns

**Note:** change of SQL templates should be considered in the Connection Properties tab instead of in Tool Properties. The reason is that SQL templates and the associated settings are most often

related to a specific database connection.

Detailed information about variables and their syntax is provided in the [Executing SQL statements in the SQL Commander](#).

Property	Name	Description
SQL Templates	<b>SELECT ALL</b>	Command used when selecting all rows for a table
	<b>SELECT ALL WHERE</b>	Command used when selecting some rows for a table
	<b>SELECT COUNT</b>	Command used to get the number of rows in a table
	<b>INSERT INTO</b>	Command used to insert a new row into a table
	<b>UPDATE WHERE</b>	Command used to update an existing row in a table
	<b>DELETE WHERE</b>	Command used to delete a specific row in a table
	<b>DROP TABLE</b>	Command used to drop a specific table
	<b>CREATE TABLE</b>	Command used to create a new table with an optional primary key
	<b>CREATE INDEX</b>	Command used to create an index for a specific table
	<b>Monitor Row Count</b>	Command used to get the number of rows in a table and the current time stamp
	<b>Monitor Row Count Change</b>	Command used to get the row count difference in a table compared to the previous execution. The calculated row count and the current time stamp is returned
Enclosing Characters	<b>Table Name Prefix/Suffix</b>	Here you can enter the character(s) that DbVisualizer will prefix/suffix all table names with. This is needed when dealing with table names that for example contains a blank character, etc.
	<b>Column Name Prefix/Suffix</b>	See above.
<b>Variable Identifier</b>		The identifier for a variable. A variable starts and ends with this identifier.
<b>Variable Delimiter</b>		The delimiter used to identify the parts of a variable.

### Statement Delimiters

Statement delimiters define how a script should be divided into specific SQL statements in the pre-processing phase.

Property	Description
<b>SQL Statement Delimiter 1</b>	Defines the character(s) used to delimit one SQL statement from another in a SQL script

<b>SQL Statement Delimiter 2</b>	Defines the additional character(s) used to delimit one SQL statement from another in a SQL script. If there is no need for more than one SQL statement delimiter then set this one to the same as delimiter 1.
<b>Allow "go" as Delimiter</b>	Specifies whether <b>go</b> as the first word on a single line will be interpreted as a statement delimiter.
<b>Begin Identifier</b>	Defines the character(s) that identifies the start of an anonymous SQL block
<b>End Identifier</b>	Defines the character(s) that identifies the end of an anonymous SQL block

## Pre-processing

Pre-processing properties define as the name implies the settings that are used when DbVisualizer pre-processes any SQL statement or script of statements before executing it with the JDBC driver.

### Comments

Property	Description
<b>Single Line Identifier 1</b>	Defines the character(s) that identifies the beginning of a one line comment
<b>Single Line Identifier 2</b>	Defines the additional character(s) that identifies the beginning of a one line comment
<b>Block Comment Begin Identifier</b>	Specifies the character(s) that identifies the start of a multi line comment block
<b>End</b>	Specifies the character(s) that identifies the end of a multi line comment block

### New Lines

Property	Description
<b>Remove New Line Characters</b>	Specifies whether any new line characters should be removed from any SQL statement executed in the SQL Commander and in the implicit SQL execution functionality in DbVisualizer. Some drivers/databases such as DB2 requires that no new line characters are part of any executed SQL.

## Transaction

Property	Description
<b>Auto Commit</b>	Defines if each executed SQL statement will be auto committed or not. This setting applies for all SQL's that are executed in the SQL Commander. The inline and form editors in DbVisualizer Personal handles the commit and rollback management independently of the setting of Auto Commit.
<b>Pending Transactions at Disconnect</b>	Defines what DbVisualizer will do on exit from the application when the auto commit setting is disabled.
<b>Transaction Isolation</b>	Attempts to change the transaction isolation level for all database connections. Note: If this property is changed during a transaction, the result is JDBC driver specific.

## Copy

The copy category groups properties that are used to control the result of using **Copy Selected Cells** or when using the **Ctrl-C** shortcut in grids.

Property	Description
<b>Column Delimiter</b>	Specifies the delimiter between columns in a multi column copy
<b>End of Line Delimiter</b>	Specifies the new line control characters for multi row copy requests

## Bookmarks

Property	Description
<b>Number of Bookmarks Limit</b>	Specifies the number of SQL bookmarks that the <b>New</b> and <b>History</b> bookmark object may keep until the lists are truncated.

## Monitor

Property	Description
<b>Start Monitors Automatically</b>	Check to enable start of monitors automatically when database connections are established.

## Grid

Property	Description
<b>Fit Grid Column Widths</b>	Enable to let DbVisualizer automatically fit the content in each grid column based on the widest cell value.

### Meaning of setting Max Chars

The Max Chars property in the Database Objects Data tab and in the SQL Commander is used to control the max number of characters that text values can hold. If the number of characters for a text column is wider than this setting then the column is colored in a light red color.

The meaning of setting this property can be one of the following:

- **Truncate Values**

Will truncate the original value to be less than the setting of Max Chars.

**Note:** this will affect any subsequent edits and SQL operations that use the value since it's truncated. This setting is only useful to save memory if viewing very large text columns.

- **Truncate Values Visually**

Will truncate the visible value only and leave the original value intact.

This is the preferred setting since it will not harm the original value. The disadvantage is that more memory is needed if dealing with large text columns.

## Colors

The Colors category defines how odd and even numbered rows in grids should be presented.

## Binary/BLOB and CLOB Data

Property	Description
<b>BLOB</b>	Specifies how BLOB and binary data values will be represented in grids. Setting this property to <b>By Value</b> will result in performance penalties and the memory consumption will increase dramatically.
<b>CLOB</b>	Specifies how BLOB and binary data values will be represented in grids. Setting this property to <b>By Value</b> will result in performance penalties and the memory consumption will increase dramatically.

## Editor

The editor category controls various settings specific for the SQL Commander editor.

## Key Bindings

The editor category is used to modify the key bindings that are used in the editor.

## Auto Completion

Property	Description
<b>Prepend Column Name with Table Name</b>	specifies if completed column names should be prefixed with the actual table name i.e TABLE.COLUMN. This setting will only have effect if <b>NOT</b> using table name aliases.
<b>Prepend Table Name with Schema/Database Name</b>	This property is self explanatory.

## Fonts

Individual fonts can be defined for SQL editors and grids.

## Debug

The debug category is used to control the amount of output that is produced when setting various debug modes. Normally only error messages are displayed in the default debug destination which is the **Tools->Debug Window**. The support team often refer to the debug properties when we want more information in a problem situation.

Property	Description
<b>Debug Output Destination</b>	Specifies the destination to which all debug messages will be written to. It is not advisable to set this to <b>Off</b> since then also error messages will then also be ignored. <b>Standard Out</b> is only useful if the debug mode of the DbVisualizer launcher is enabled.
<b>Debug DbVisualizer</b>	Defines the amount of logging that will be produced. <b>Full</b> output is when <b>Log Level</b> is set to Debug and lowest output is <b>Error</b> . Setting <b>Detail Level</b> to <b>Full</b> produces the most detail and also consume more resources.

Read more about [Problem Resolution](#).

# Export and Print

## Introduction

The export and printing capabilities are used to transfer information from DbVisualizer primarily to files and printers. Both functions are based on what's in the current context and adapt the configuration possibilities accordingly. Data in the following contexts (components) can be exported:

- Grid
- Graph
- Chart

These contexts can be printed:

- Grid
- Graph

Please read more in the [Getting Started and General Overview](#) document for information about these components.

Printing of charts is currently not supported. Instead we recommend that you first export the chart and then use your favorite printing tool to get it on paper (a standard web browser is sufficient). The **Print Preview** operation can only be used for grid data.

There are major differences between DbVisualizer Free and Personal when exporting grid data. This document explains the complete functionality in the Personal edition even though it implicitly covers the export functionality in DbVisualizer Free.

The functionality for Export of Graphs and Charts and Printing of Graphs and Grids is the same in both editions.

## Export

The export functionality is used to export data that has been fetched and presented in DbVisualizer to another source. The export dialog looks different depending on whether a grid, graph or chart data is going to be exported. The following sections describe the settings that can be made in each of these contexts.

### How to launch

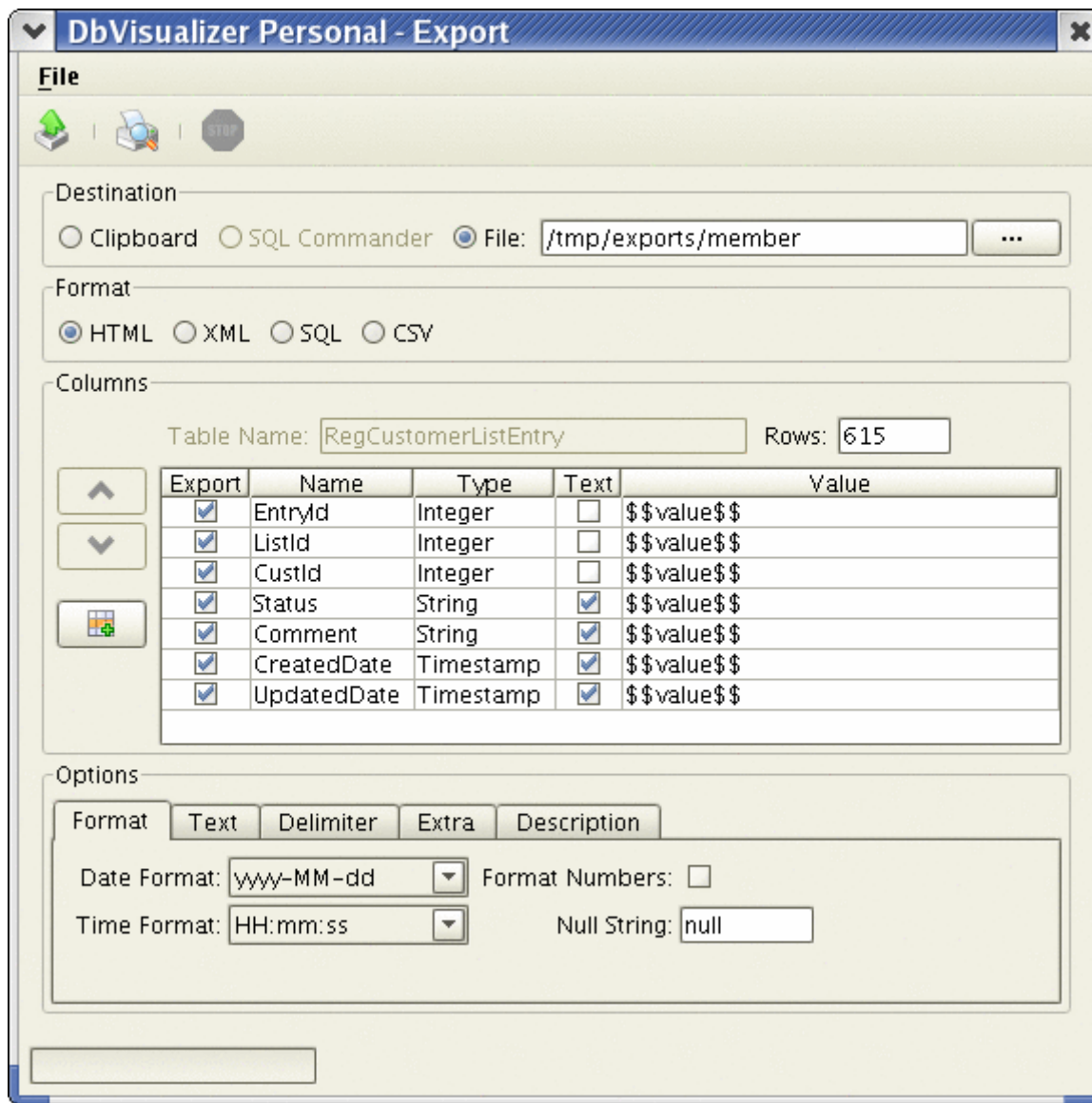
Export is primarily initiated using the **File->Export** main menu choice. This operation examines the current context and displays the appropriate dialog. The **File->Export Selection** main menu choice is specifically for **Grid** contexts and is used to export the current selection instead of all data in the grid. It is only enabled if the current context is a grid and if there are any selected cells in it. In addition all grids throughout DbVisualizer offer the right click menu choice for **Export Selection**. It is simply a shortcut for the **File->Export Selection** main menu operation.

**Note:** Export Selection will only include the selected columns in the Export window (i.e in the Columns list).

### Grid data

There are a range of options to configure the exported data for grids. This is how the window is organized:





**Figure: The Export window for grid data**

Please see the sections below for detailed information on each field and what settings that can be made.

## Destination

The destination field specifies the target destination for the exported data.

Destination	Description
<b>Clipboard</b>	Export to the (system) clipboard is convenient if you want to use the exported data in another application without the extra step of exporting to file first. Data can even be pasted into a spreadsheet application such as Excel or StarOffice and the cells in the grid will appear as cells in the spreadsheet. Read more about the CSV format in the <a href="#">Format</a> section.
<b>SQL Commander</b>	This option is only enabled if exporting in the SQL format. It simply transfers the export data (as SQL INSERT statements) to the SQL Commander editor. The SQL Commander can then be used to execute the script of statements.
<b>File</b>	This option outputs the data to a named file.

## Format

Grid data can be exported in the following formats.

Format	Description
<b>HTML</b>	The data is exported in HTML format using the <TABLE> and associated tags.
<b>XML</b>	<p>The XML format is handy when importing or using the exported data in an XML enabled application. The structure of the XML format is:</p> <pre>&lt;ROWSET&gt;   &lt;ROW&gt;     &lt;Column1&gt;5&lt;/Column1&gt;     &lt;Column2&gt;Hepp&lt;/Column2&gt;     &lt;Column3&gt;59248&lt;/Column3&gt;   &lt;/ROW&gt;   &lt;ROW&gt;     &lt;Column1&gt;15&lt;/Column1&gt;     &lt;Column2&gt;Hopp&lt;/Column2&gt;     &lt;Column3&gt;41993&lt;/Column3&gt;   &lt;/ROW&gt;   &lt;ROW&gt;     &lt;Column1&gt;15&lt;/Column1&gt;     &lt;Column2&gt;Hupp&lt;/Column2&gt;     &lt;Column3&gt;44115&lt;/Column3&gt;   &lt;/ROW&gt; &lt;/ROWSET&gt;</pre>
<b>SQL</b>	<p>The SQL format simply creates an SQL INSERT statement for each row in the grid. It also uses the column names from the grid to define the column clause in the SQL statement.</p> <pre>insert into table1 (Column1, Column2, Column3) values (5, 'Hepp', 59248); insert into table1 (Column1, Column2, Column3) values (15, 'Hopp', 41993); insert into table1 (Column1, Column2, Column3) values (16, 'Hupp', 44115);</pre>

<b>CSV</b>	<p>The CSV format (Character Separated Values) is used to export the grid of data to a file in which each column is separated with a character or several. It is even possible to specify the row delimiter (aka newline sequence of characters).</p> <pre>5,Hepp,59248 15,Hopp,41993 16,Hupp,44115</pre> <p>The above example use a "," as the column delimiter and a "\n" sequence as the row delimiter (invisible above).</p>
------------	--

## Columns

The columns list is used to control what columns will be exported and the format of their data. The list is exactly the same as the column headers in the original grid i.e. if a column was manually removed from the grid before launching export then it will not appear in this list. The **Table Name** field may list the name of the actual database table if it is known at the time when the export dialog is displayed. The **Rows** field lists the number of rows that are going to be exported.

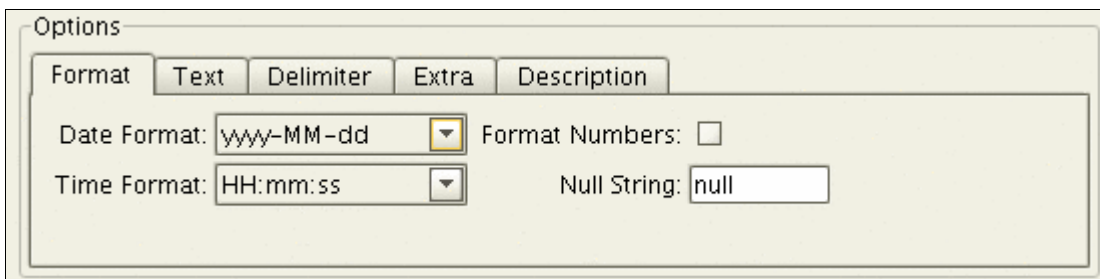
Field	Description
<b>Export</b>	Defines whether the column will be exported or not. Uncheck it to ignore the column in the exported output.
<b>Name</b>	The name of the column. This is only used if exporting in HTML, XML or SQL format. Column headers are optional in the CSV output format.
<b>Type</b>	The internal DbVisualizer type for the column. This type is used to determine if the column is a text column (i.e if the data will be surrounded in quotes or not).
<b>Text</b>	Specifies if the column is considered to be a text column (this is determined based on the type) and so if the value will be enclosed in quotes.
<b>Value</b>	The default "\$value\$" variable will simply be substituted by the actual value in the exported output. You can enter additional static text in the value field. This is also the place where any <a href="#">test data generators</a> are defined.

## Options

The options tabs are used to manipulate a range of settings that are used to control the output.

### Format

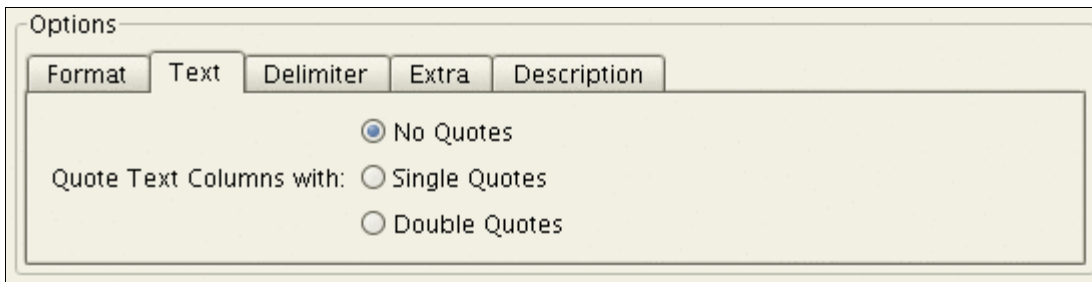
The format settings control the appearance of date, time, the textual representation of null and whether numbers will include a thousands separator in the exported output.



**Figure: The Format settings**

## Text

The text settings control whether text data will be enclosed by single, double or no quotes. A text column is defined by checking the **Text** column in the **Columns** list. All other columns will be exported without any quotes.



Options

Format Text Delimiter Extra Description

No Quotes

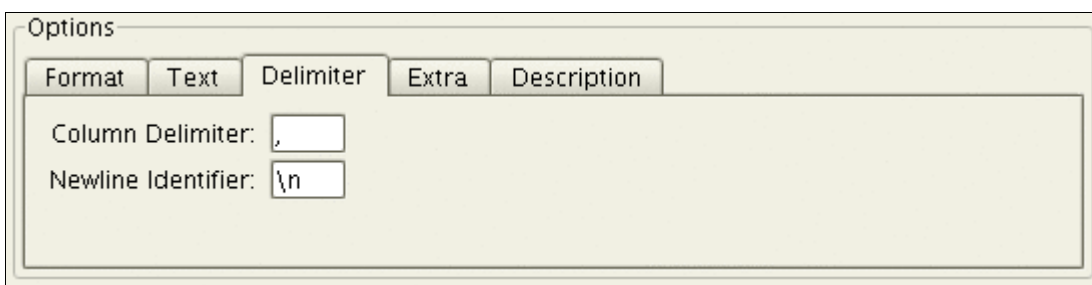
Quote Text Columns with:  Single Quotes

Double Quotes

**Figure: The Text settings**

## Delimiter

The delimiter settings specify the column delimiter character(s) and the newline character sequence for CSV output format. These settings are used independently if exporting to file or any of the other destinations.



Options

Format Text Delimiter Extra Description

Column Delimiter: ,

Newline Identifier: \n

**Figure: The Delimiter settings**

The \ character indicates that an encoded character will follow. The following escaped characters can be used in the column delimiter and the newline identifier:

Sequence	Description
\n	New line
\r	Carriage Return
\t	Tab

Exporting to file in different OS'es and exporting to clipboard for later inclusion in other applications requires different settings of both the column delimiter and the newline identifier. The following lists the most common situations:

Export Destination	Target OS/Application	Column Delimiter	Newline Identifier
File	Windows	any	\r\n
	Unix/Linux	any	\n

<b>Clipboard</b>	Paste into Microsoft Excel	\t	\n
	Paste into StarOffice/OpenOffice SpreadSheet	\t	\n or \r\n

### Extra

The extra settings are used to specify if the original SQL command and the column names will appear in the exported output.

The **SQL Command** setting is only enabled if the actual SQL command that produced the data in the grid is known. The **Column Names** setting is only valid for use in CSV formats since the column names are required in the other formats.

Options

Format Text Delimiter **Extra** Description

Include:  SQL Command  Column Names

**Figure: The Extra settings**

### Description

The description setting consists of a text field in which informative text about the data can be entered. This will be in the exported output for HTML and XML formats only (within a comment block for XML).

Options

Format Text Delimiter Extra **Description**

Sales figures for the third quarter.

**Figure: The Description settings**

### Generate Test Data


The test data generator is useful when you need to add random column data to the exported output. The actual value of the data that is going to be in the exported output is referenced by the `$$value$$` variable in the Columns list. This variable is simply replaced by the real value during the export process. Additional static data can be added before and after the `$$value$$` field and will be exported as entered. The value field is also the place to setup any test data generators. While in the editing mode of the value field there is a right click menu with the supported generator functions:

Function Name	Function Call	Example
---------------	---------------	---------

<b>Generate random number</b>	<code>\$\$var  randomnumber(1, 2147483647)\$\$</code>	Generates a random number between 1 and 2147483647
<b>Generate random string of random size</b>	<code>\$\$var  randomtext(1, 10)\$\$</code>	Generates random text with a length between 1 and 10 characters
<b>Generate random value from a list of values</b>	<code>\$\$var  randomenum(v1, v2, v3, v4, v5)\$\$</code>	Picks one of the listed values in random order
<b>Generate sequential number</b>	<code>\$\$var  number(1, 2147483647, 1)\$\$</code>	Generates a sequential number starting from 1. The generator re-starts at 1 when 2147483647 is reached. The number is increased with 1 every time a new value is generated.

### Test data generator sample

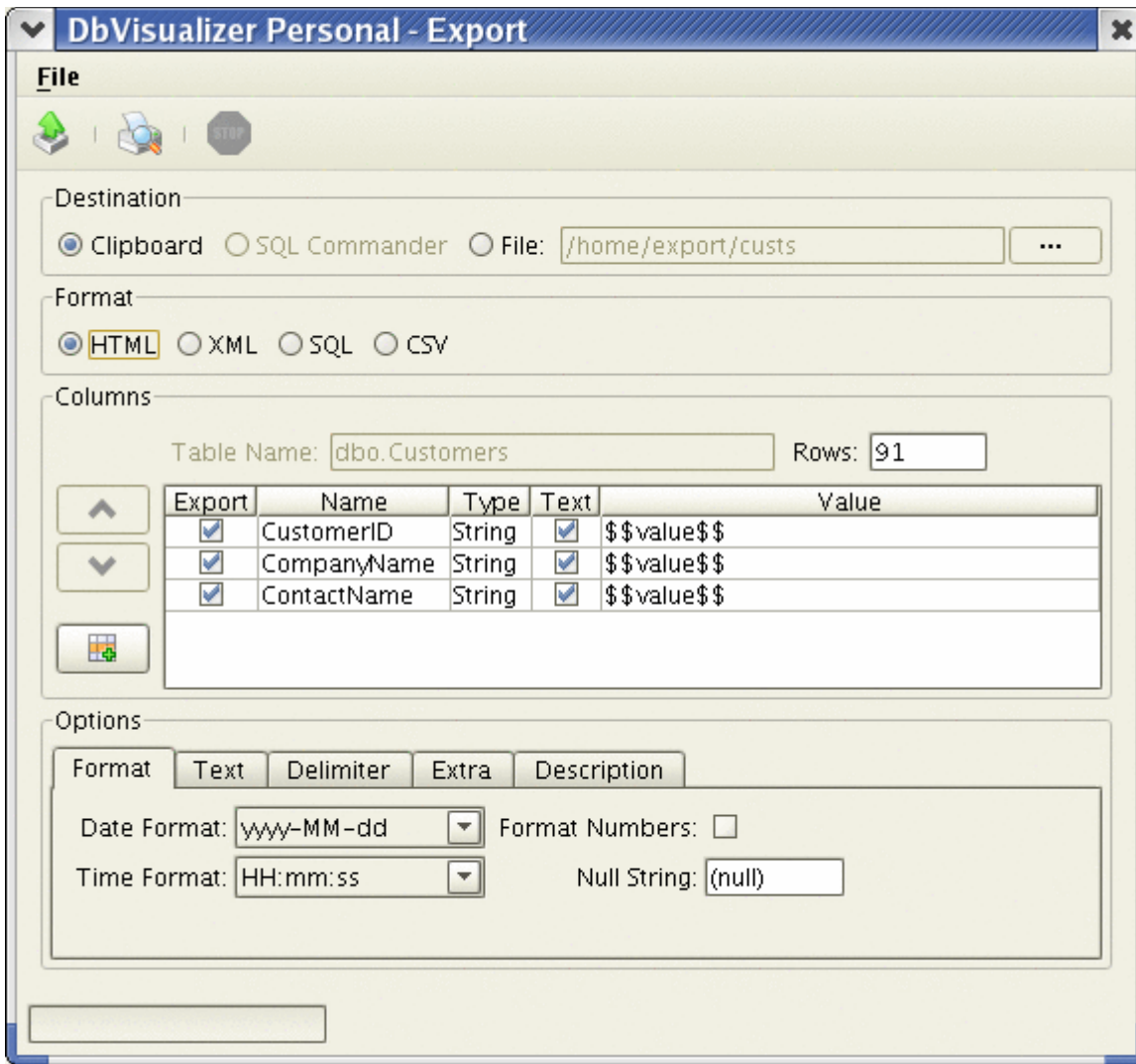
Here follows an example that utilizes the test data generators. Consider this data:

	CustomerID 	CompanyName	ContactName
1	ALFKI	Alfreds Futterkiste	Maria Anders
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
3	ANTON	Antonio Moreno Taquería	Antonio Moreno
4	AROUT	Around the Horn	Thomas Hardy
5	BERGS	Berglunds snabbköp	Christina Berglund
6	BLAUS	Blauer See Delikatessen	Hanna Moos
7	BLONP	Blondesddsl père et fils	Frédérique Citeaux
8	BOLID	Bólido Comidas preparadas	Martín Sommer
9	BONAP	Bon app'	Laurence Lebihan
10	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln
11	BSBEV	B's Beverages	Victoria Ashworth
12	CACTU	Cactus Comidas para llevar	Patricio Simpson
13	CENTC	Centro comercial Moctezuma	Francisco Chang
14	CHOPS	Chop-suey Chinese	Yang Wang
15	COMMI	Comércio Mineiro	Pedro Afonso
16	CONSH	Consolidated Holdings	Elizabeth Brown
17	DRACD	Drachenblut Delikatessen	Sven Ottlieb
18	DUMON	Du monde entier	Janine Labrune
19	EASTC	Eastern Connection	Ann Devon
20	ERNSH	Ernst Handel	Roland Mendel
21	FAMIA	Familia Arquibaldo	Aria Cruz

Max Rows:  Max Chars:  0.113 sec/0.003 sec | 91 / 3 | 1-22

**Figure: Sample of grid data**

The export window will display the following:



**Figure: The export window**

In addition to the original columns we want to remove the **CustomerID** from the exported data and add a new column named **Id** as the first column. This column will be the new primary key and will be a sequential number starting at 1000. This is done by adding the **Generate sequential number** function. The list will now look like this:

Export	Name	Type	Text	Value
<input checked="" type="checkbox"/>	Id	String	<input type="checkbox"/>	\$\$var number(1000, 2147483647, 1)\$\$
<input type="checkbox"/>	CustomerID	String	<input checked="" type="checkbox"/>	\$\$value\$\$
<input checked="" type="checkbox"/>	CompanyName	String	<input checked="" type="checkbox"/>	\$\$value\$\$
<input checked="" type="checkbox"/>	ContactName	String	<input checked="" type="checkbox"/>	\$\$value\$\$

**Figure: Customized columns list with a generator function**

Previewing the data (or exporting it) results in this:

```
Id,CompanyName,ContactName
1000,Alfreds Futterkiste,Maria Anders
1001,Ana Trujillo Emparedados y helados,Ana Trujillo
1002,Antonio Moreno Taquería,Antonio Moreno
1003,Around the Horn,Thomas Hardy
1004,Berglunds snabbköp,Christina Berglund
1005,Blauer See Delikatessen,Hanna Moos
```

1006,Blondesdds1 père et fils,Frédérique Citeaux  
1007,Bólido Comidas preparadas,Martín Sommer  
1008,Bon app',Laurence Lebihan  
1009,Bottom-Dollar Markets,Elizabeth Lincoln  
1010,B's Beverages,Victoria Ashworth  
1011,Cactus Comidas para llevar,Patricio Simpson  
1012,Centro comercial Moctezuma,Francisco Chang  
1013,Chop-suey Chinese,Yang Wang  
1014,Comércio Mineiro,Pedro Afonso  
1015,Consolidated Holdings,Elizabeth Brown  
1016,Drachenblut Delikatessen,Sven Ottlieb  
1017,Du monde entier,Janine Labrune  
1018,Eastern Connection,Ann Devon  
1019,Ernst Handel,Roland Mendel  
1020,Familia Arquibaldo,Aria Cruz  
...

## Preview

The preview operation is used to check the data before it is exported. The general recommendation is to use this feature before exporting the data. Having to re-launch the export window (to do more edits) will reset the columns list and any manual changes will be lost.

## Text data

The dialog when exporting result sets in **Text** format is very simple.

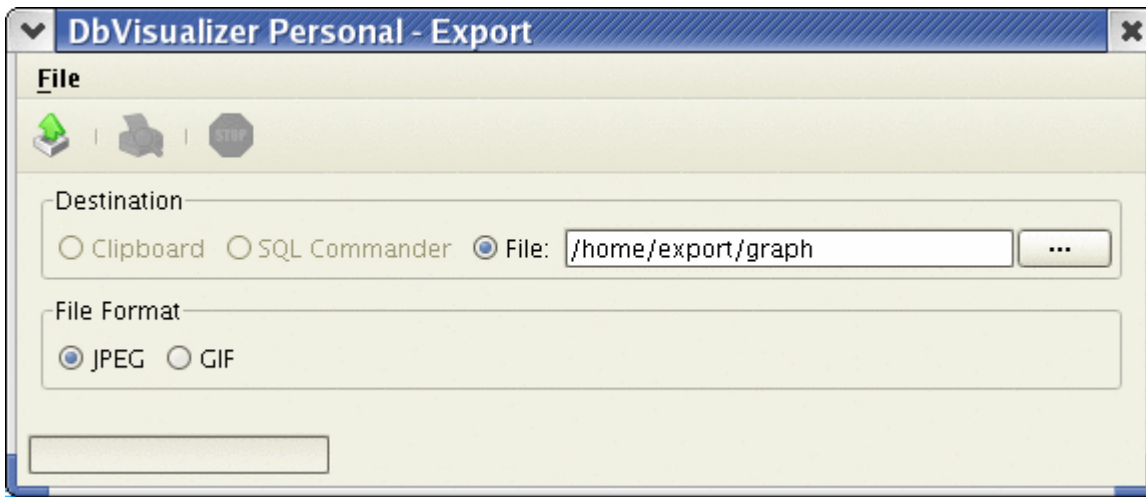


Figure: Export window for text format result sets

## Graph data

Exporting references graphs will export the graph in the same zoom level as it appears on the screen. The export window when exporting graphs looks like this:





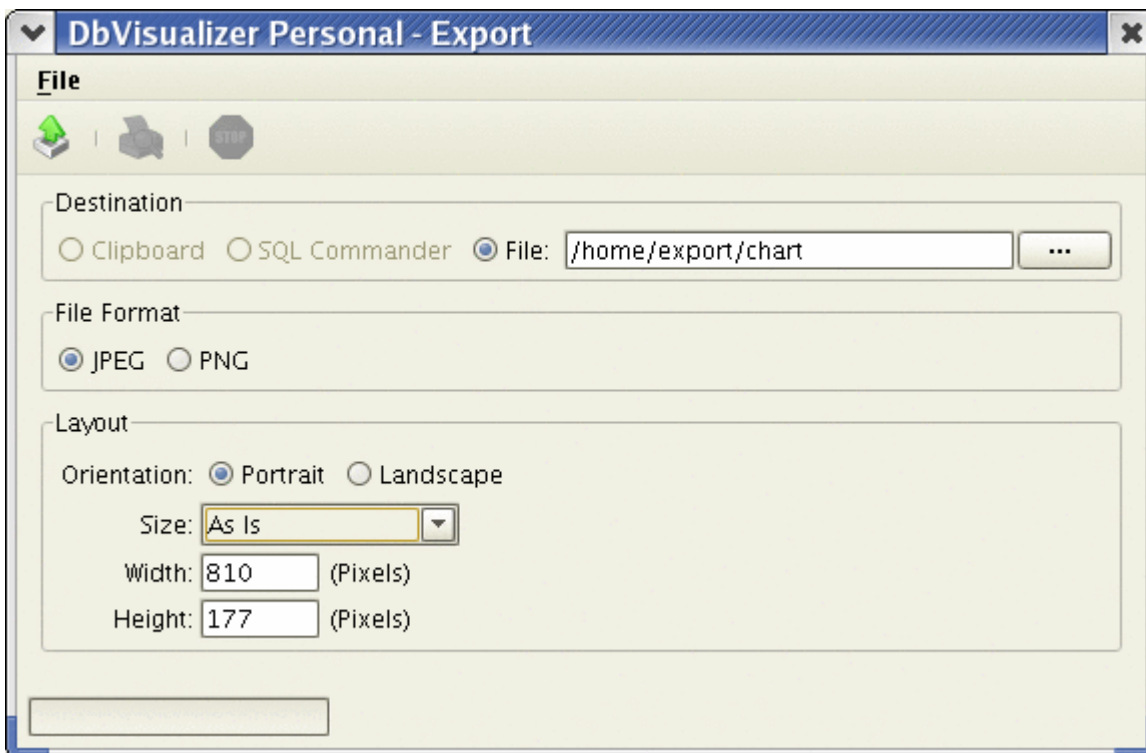
**Figure: Export window for graphs**

The export window is quite limited as compared to when exporting grids. The graph can only be exported to a **File** in the **JPEG** or **GIF** formats.

Export of graphs cannot be previewed.

## Chart data

Exporting charts adds the capabilities to set the size and orientation of the exported file.



**Figure: Export window for charts**

A chart can only be exported to a **File** in the **JPEG** and **PNG** formats. The optional **Layout** settings are used to control the size of the image. The initial width and height are the same as the size of the chart as it appear on the screen. The **Size** list when clicked shows a list of well known paper formats. The **Width** and **Height** will be changed to match the selected size. Setting the width and height or selecting a pre-defined size will scale the exported image accordingly.

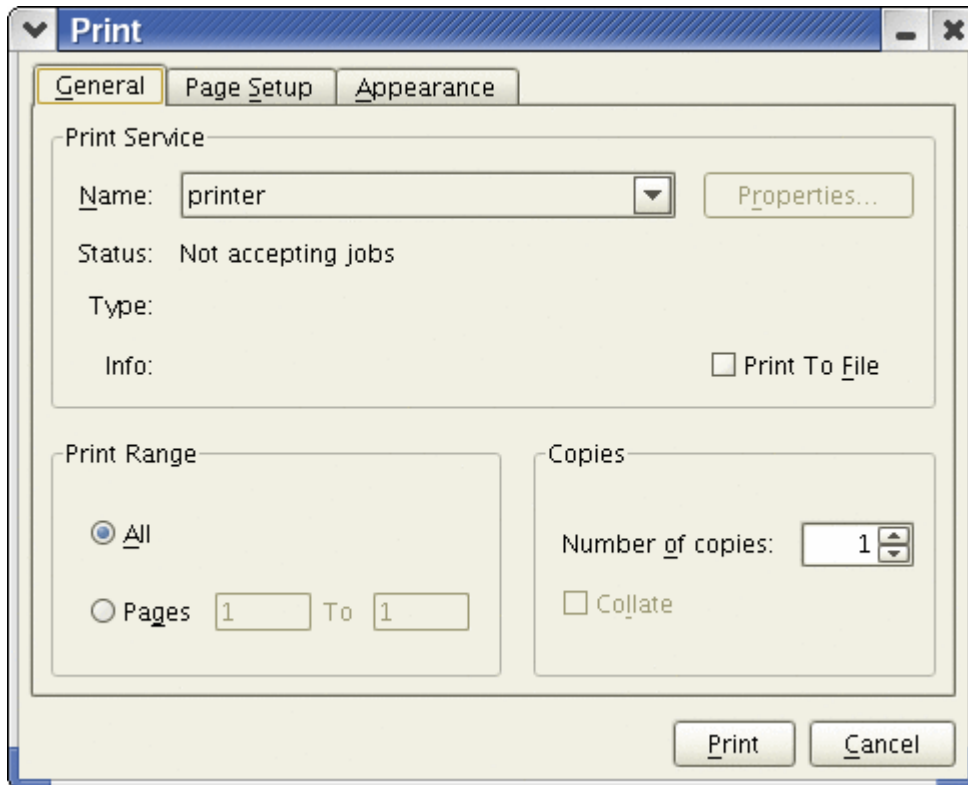
Export of charts cannot be previewed.

# Print

The printing support in DbVisualizer supports printing of Grids and Graphs. The print dialog looks somewhat different depending on what is printed.

## Grid

Printing a grid in DbVisualizer causes the visual grid to be output on paper. This includes the table headers, sort and primary key indicator, etc. It can be output as a screen shot that spans several pages depending on the number of rows and columns that are printed. The other solution to printing grids is to export to HTML and then use a web browser to print it. The choice of which is more attractive than the other is up to you to decide.

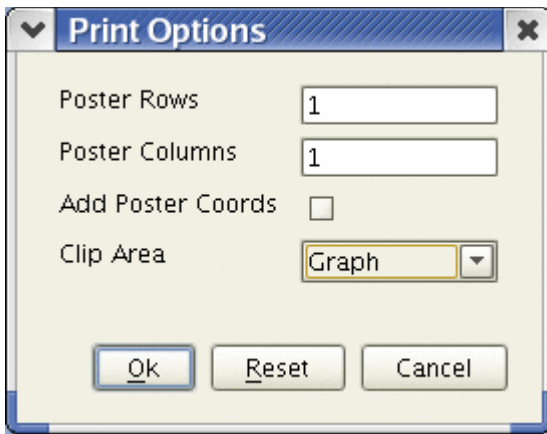


**Figure: Standard print dialog**

The content and layout of the print dialog is platform specific. The above screen shot is from Linux/RedHat.

## Graph

The graph printing setup dialog adds a step before the standard printing dialog is displayed.



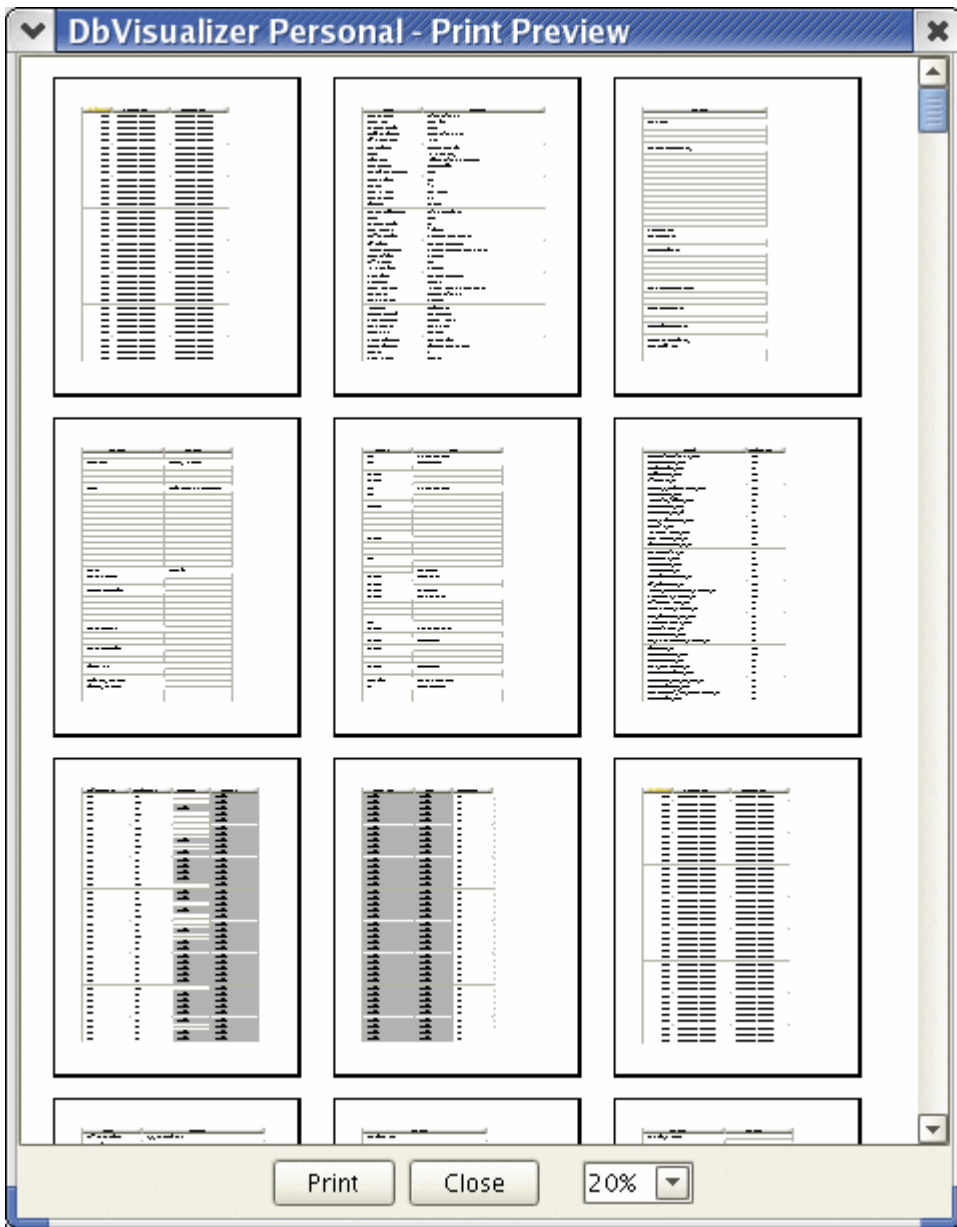
**Figure: Print options when printing graphs**

It is possible to specify the number of rows (pages) and columns (pages) that the complete image will be divided into. It is also possible to select whether the view as it appears on the screen will be printed or the complete graph.

## Print Preview

The **File->Print Preview** operation is used to launch a viewer in which a preview of the current grid is presented.

**Note:** The Print Preview can only be used for grid printouts.



**Figure: Print preview window**

The print preview supports setting the zoom level.