



e-DBI is a java application that facilitates the navigation and exploration of several multi-format data sources with potential for data integration. **e-DBI** implementation is based on the open source [Squirrel SQL](#) project.

e-DBI is free software, and you are welcome to redistribute it under the terms of the [GNU Lesser General Public License](#).

Acknowledgment:

We acknowledge that a major part of the text included in this document is taken from the open source project [Squirrel SQL version 2.6](#). Parts of the text have been re-phrased or added to better cope the main functionalities and enhancements targeted by e-DBI.

Table of Contents

- I. [e-DBI Project Description](#)
 - A. [e-DBI Architecture](#)
 - B. [e-DBI Implementation](#)
 - C. [Installing e-DBI](#)
 - D. [running e-DBI](#)
 - 1. [Command Line Options](#)
- II. [Connecting to a Database](#)
 - A. [The Driver Registration](#)
 - B. [The Data Source Parameters](#)
- III. [Session Window & Connection Tree](#)
 - A. [Executing SQL](#)
 - 1. [SQL History](#)
 - 1. [Editing the SQL Results](#)
 - B. [The Contents Tab and Editing Data](#)
 - 1. [Viewing Data in the Contents Tab](#)
 - 1. [Enabling Editing](#)
 - 1. [Setting the Format and Editing Mode](#)
 - 1. [How To Edit](#)
 - 1. [Inserting and Deleting Rows](#)
 - 1. [Printing](#)
 - C. [Using the Popup Window](#)
 - 1. [Editing Within the Popup](#)
 - 1. [Import, Export and Editing with an External Command](#)
 - D. [Limitations](#)
 - E. [Summary of How it works \(and why you may care\)](#)
 - F. [Quirks, Oddities and Known Bugs and work-arounds](#)
- IV. [Data Types](#)
- V. [Global Preferences](#)
 - A. [General Tab](#)
 - B. [SQL Tab](#)
 - C. [Proxy Tab](#)
 - D. [Data Type Controls Tab](#)
 - E. [L&F Tab](#)
 - F. [Fonts Tab](#)
- VI. [New Session and Session Properties](#)
 - A. [General Tab](#)
 - B. [Object Tree Tab](#)
 - C. [SQL Tab](#)
- VII. [Plugins](#)
- VIII. [Logs](#)
- IX. [Menus](#)
 - A. [File Menu](#)
 - B. [Drivers Menu](#)
 - C. [Aliases Menu](#)
 - D. [Plugins Menu](#)
 - E. [Session Menu](#)
 - F. [Windows Menu](#)
 - G. [Help Menu](#)

I. e-DBI Project description:

e-DBI is a database application that allows the scientists to seamlessly connect to several of multi-format data sources. It facilitates the navigation and exploration of scientific data sources with potential for data integration. In a typical integration scenario, the scientists need to perform several activities and tasks to gather and collect all the information from the different data sources. With **e-DBI**, however, these tasks are performed in a single-access point, while the integration is carried out by defining a virtual database based on the collected data sources. Furthermore, the **e-DBI** tool uses a relational backend, enabling customizations for the location and format of the virtual database.

A. e-DBI Architecture

The tool e-Science Database Integrator (e-DBI) aims at providing a data access interface more suitable to scientists. As shown in Fig.1, a scientist needs consider the following steps to define a virtual (integrated) database:

1. Define a virtual database (VDB), using any relational database
2. Select the needed information from the different data sources (tables):
 - Filter the data
 - Rename table name and attributes
 - Reformat the data (apply any conversion if required)
3. Transfer the data into the new VDB, by copying the information
4. Enhance the VDB
 - Set new constraints

- n,--native-laf : Use native look and feel
- nop,--no-plugins : Don't load plugins
- nos,--no-splash : Don't display splash screen
- m,--use-default-metal-theme : Use default metal theme
- h,--help : Display Help and exit
 - -userdir: specifies where e-DBI stores its *per-user* settings such as application preferences etc. This defaults to <home>/e-DBI.
 - -home specifies the directory that e-DBI was installed into. It is setup during the installation process. You should never need to change this argument.

II. Connecting to a Database

Connection to a given database is performed in two steps: (1) the driver registration, and (2) the data source parameters (alias). The driver registration specifies the JDBC driver to use for the underlying DBMS, and the alias specifies the database connection parameters.

A. The Driver Registration

A number of default driver definitions ship with e-DBI. These are added to the Drivers List window (*Drivers* option on the *Windows* menu).

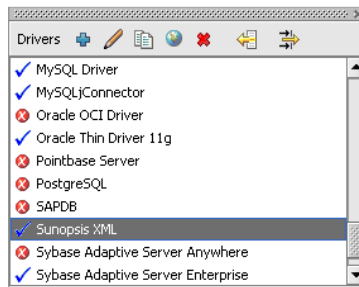


Figure 2. Drivers List Window

The ✓ icon next to a driver definition indicates that it has been successfully loaded, while the ✗ icon indicates that the JDBC driver could not be loaded and so cannot be used to connect to a database.

All of the default driver definitions assume that the JDBC driver classes are in the current class path or in your JRE extensions directory. If this is not the case you will need to modify the driver definition to point to the jar file or the classes directory that contains these classes.

You can create new driver definitions using the *New Driver* option on the *Drivers* menu.

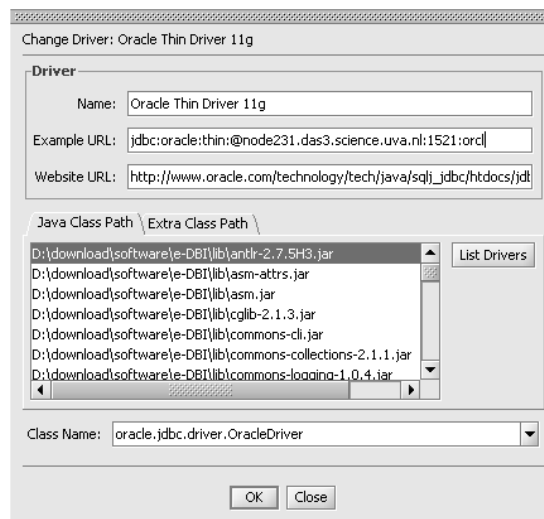


Figure 3. Driver registration (Dialog Box)

This is an example of a driver definition. This dialog can be displayed by selecting a driver definition in the Drivers List window and then selecting the *Modify Driver* option from the *Drivers* menu.

1. The **Name** text field specifies an easy to remember name for the database driver definition and is the text shown in the Drivers List window.
2. The **Example URL** shows an example URL for this driver. This can be found in the documentation that comes with the driver. This is only an example URL, the actual URL is defined in the alias.
3. The entries in the **Java Class path** tab show the directories and jars in the current class path. Pressing the *List Drivers* button will place the class names of all the JDBC drivers found in the class path into the *Class Name* dropdown control.
4. The **Class Name** specifies the class name of the JDBC driver. Either select an existing entry in the dropdown or key in the class name of the JDBC driver (it can normally be found in the documentation that comes with the driver).
5. The **Extra Class Path** tab allows you to select a JDBC driver that is in a jar file or a directory that is not in the current class path.

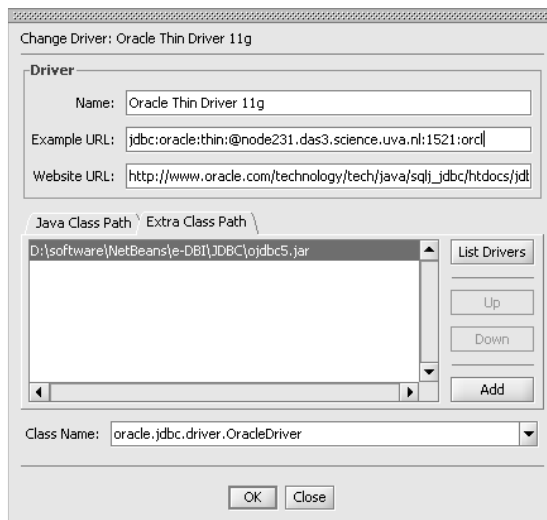


Figure 4. Driver Registration - Extra Class Path

6. The **Add** button will display a File Open dialog allowing you to select one or more jar files or directories. The **Delete** button allows you to remove a jar file or directory from the list. The **Up** and **Down** buttons allow you to change the sequence of the entries. The **List Drivers** button will place the class names of all the JDBC drivers into the **Class Name** dropdown.
7. Press the **OK** button to save the driver definition and close the window or the **Close** button to close the window without saving any changes.

B. The Data Source Parameters Specifications (alias)

Now you need to create an alias to your database.

The data sources List window can be displayed from **View Data Sources** option on the **Windows** menu. When you run e-DBI for the first time this window will be empty.

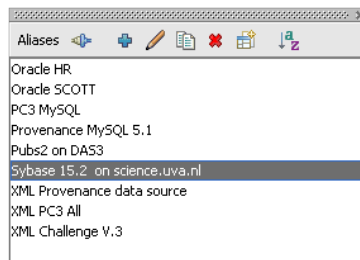


Figure 5. Data Sources List Window

Take the **New Alias** option from the **Aliases** menu and the following dialog will be displayed.

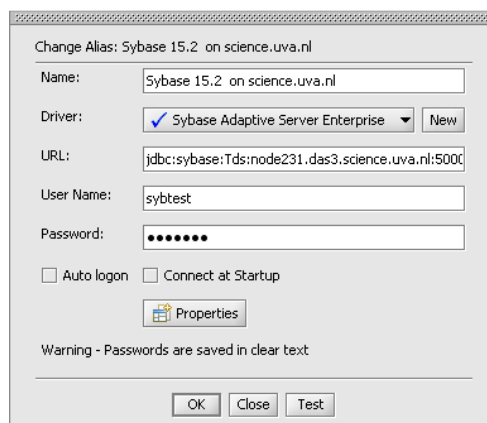



Figure 6. Data Sources - New Alias Dialog Box

1. Enter an easy to remember name in the **Name** field for this database URL. This is the text that will be displayed in the Aliases List Window.
2. Select the driver for this alias from the **Driver** dropdown or click on the **New** button to create a new driver definition.
3. Change the **URL** to point to your database.
4. Optionally enter a **user name** if you want the alias to default to a specific user.
5. The **Test** button will allow you to attempt to connect to your alias to ensure that the parameters you have entered are correct.
6. Press the **OK** button to save the alias definition and close the window or the **Close** button to close the window without saving any changes.
7. The **Properties** button ( Properties) in the alias dialog window allows you to configure alias properties such as which schemas to load into the Object Tree. This will reduce the time that it takes to populate the Object Tree after connecting/refreshing for databases like Oracle. There are three types of objects that can be selected to be loaded, loaded and cached, or not loaded at all for each schema. The Alias Properties dialog is displayed below:

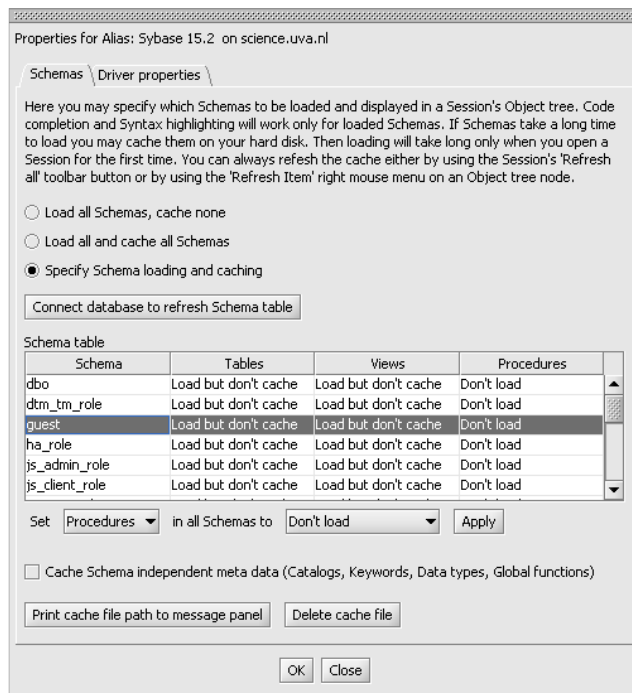


Figure 7. Data Sources - Properties Dialog Box

- The *Connect database to refresh Schema table* button will load all schemas from the database into the table for customization.

III. Session Window & Connection Tree

When you make your first connection to a database, a Session window will be shown. Additional connections to other databases will appear within the same window.

The Connection Tree on the left shows the structure of each connected database. Clicking on various nodes within the nested objects of the tree will show further information in the right hand detail panel. In the example on figure 8, the database metadata is shown. Clicking on the title of a column will sort the display by the data in that column. This is true of all tabular displays in e-DBI.

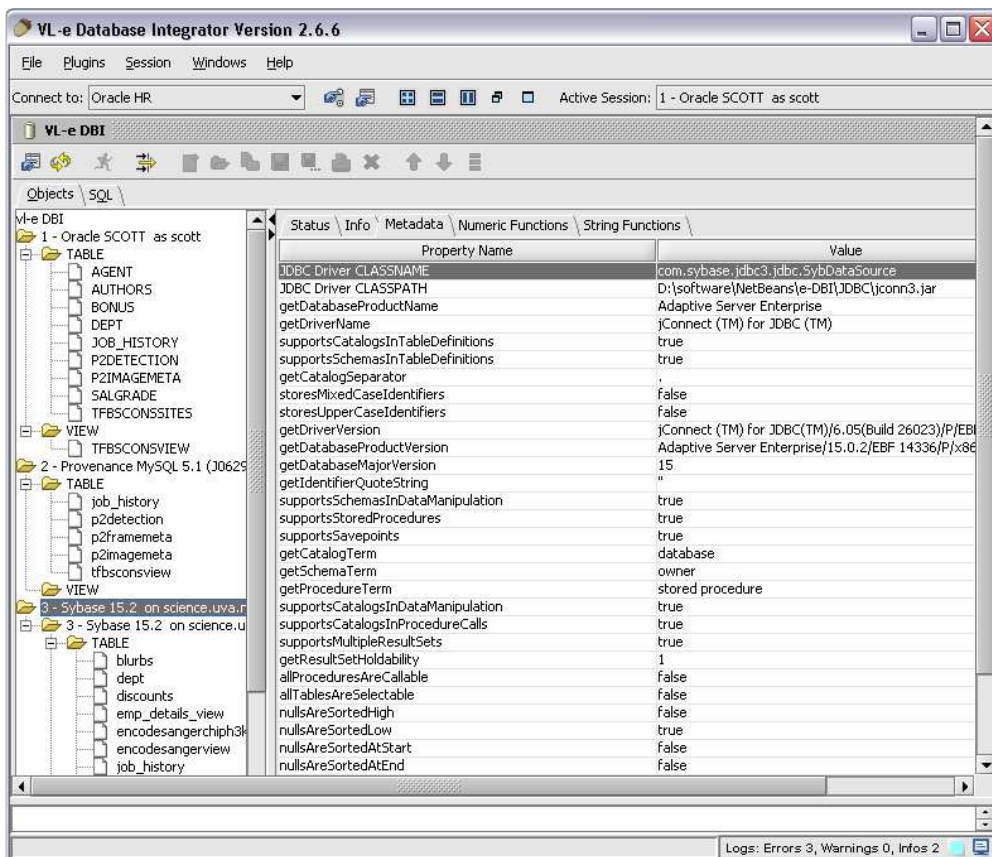


Figure 8. Session Window & Connection Tree - Database metadata

Figure 9, illustrates the data displayed when a table is selected in the Object Tree.

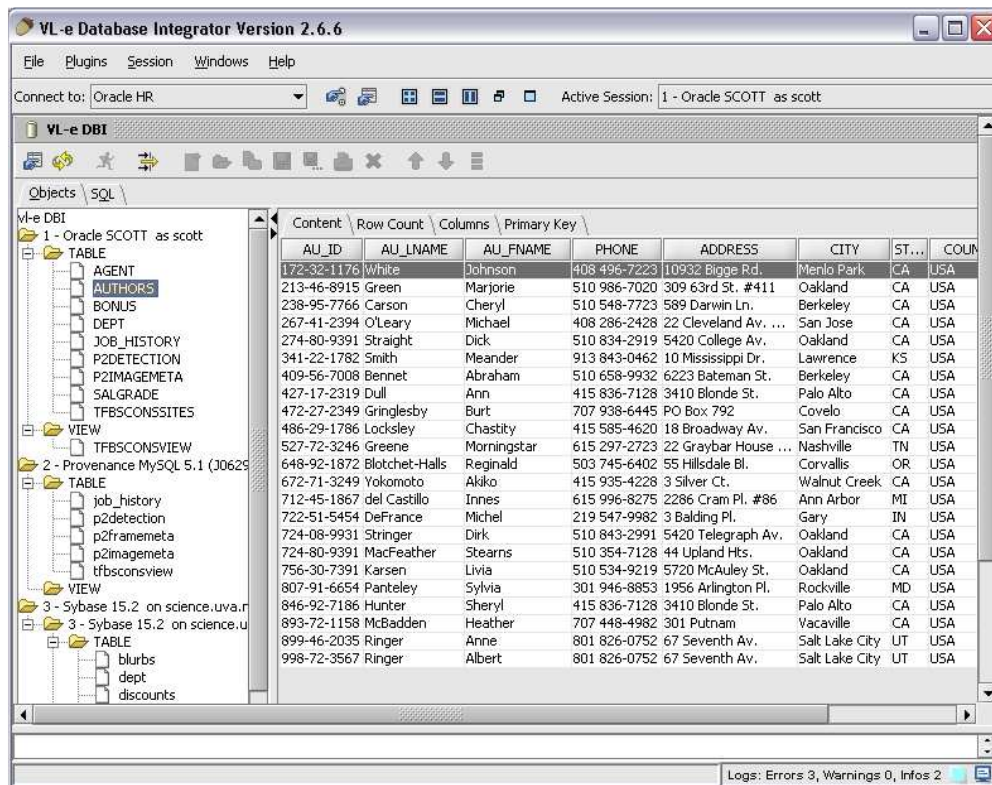


Figure 9. Session Window & Connection Tree - Table/View data

A. Executing SQL

SQL can be executed from the **SQL tab** in the window that opens once you have connected to an alias. Multiple statements can be executed and the results will be displayed in multiple tabs below the SQL entry area. If you are connected to more than one data source, the **SQL tab** will activate a connection to the database corresponding to the selected object on connection tree.

1. When the **SQL tab** is selected pressing **<ctrl><enter>**, taking the Execute SQL option from the Session menu or pressing the Execute SQL button in the session window tool bar will execute the entered SQL.
2. If a single statement is entered then pressing **<ctrl><enter>** will execute just that statement.
3. If you only want to execute part of the SQL entered then highlight the SQL that you want to execute and press **<ctrl><enter>**.
4. To only execute a single statement amongst several statements (separated from the other statements by at least one blank line) within the SQL entry area then click on the line containing the statement that you want to execute and press **<ctrl><enter>**.
5. The characters **--** at the beginning of a line will consider the statement after as a comment.

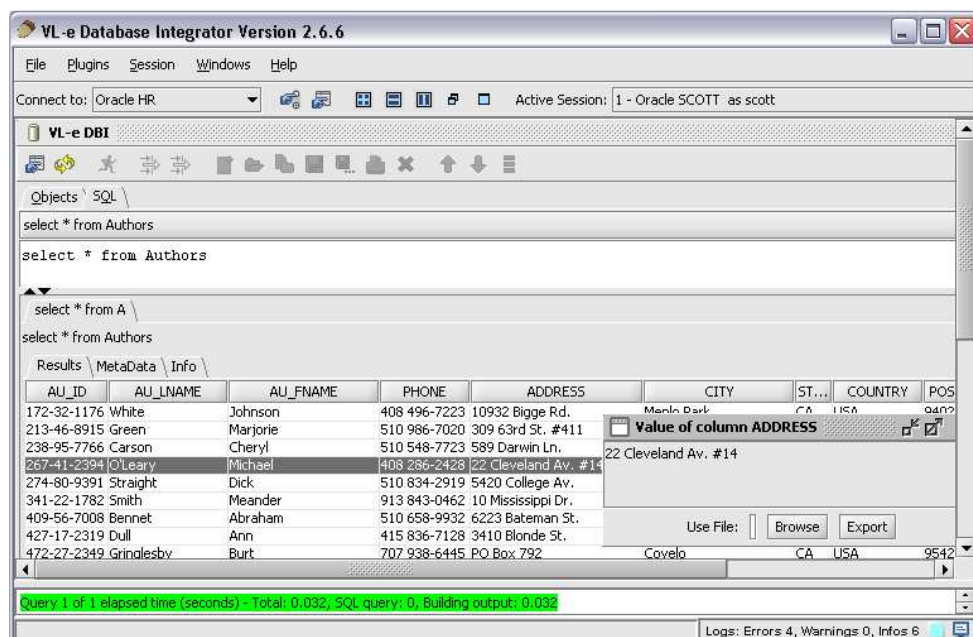


Figure 10. SQL tab - Query results & Cell content

- As can be seen in the above screen shot, double clicking in a cell of a query results will open up a new window showing the entire contents of the cell.
- Plugins can add many functions to the SQL editor. The tools popup shows all of these functions and allows the user to execute them. The functions in the popup may be filtered by typing the name of the filter into the filter text field. The tools popup can be opened by pressing **<Ctrl>-t**. A picture of the tools popup dialog is show in figure 11.

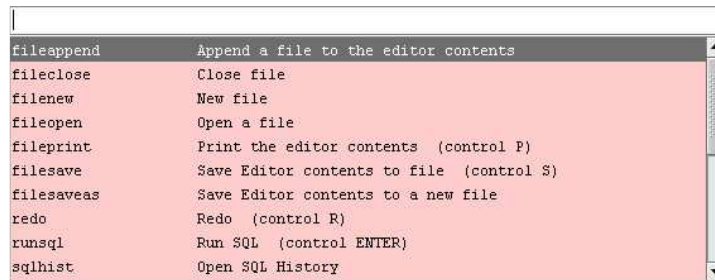



Figure 11. SQL editor- popup functions

- The  icon on the tabbed folder containing the SQL results allow you to "tear off" the query results from the tabbed folder and display it in its own window (see Figure 12).

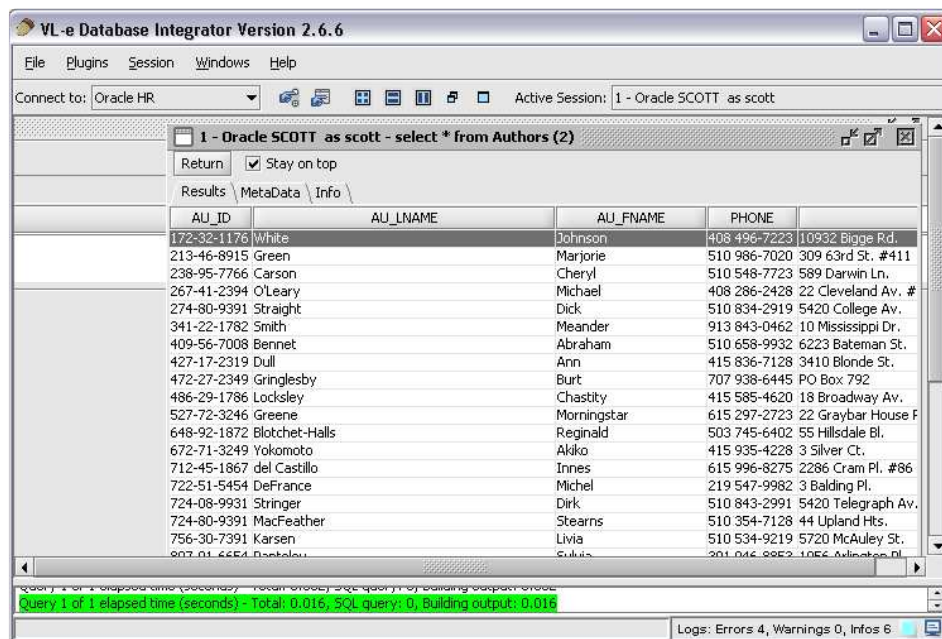


Figure 12. SQL tab Resultset - Proper Window Display

1. SQL History

Every SQL statement that is successfully executed is also added to the drop-down history list, which is shown below.

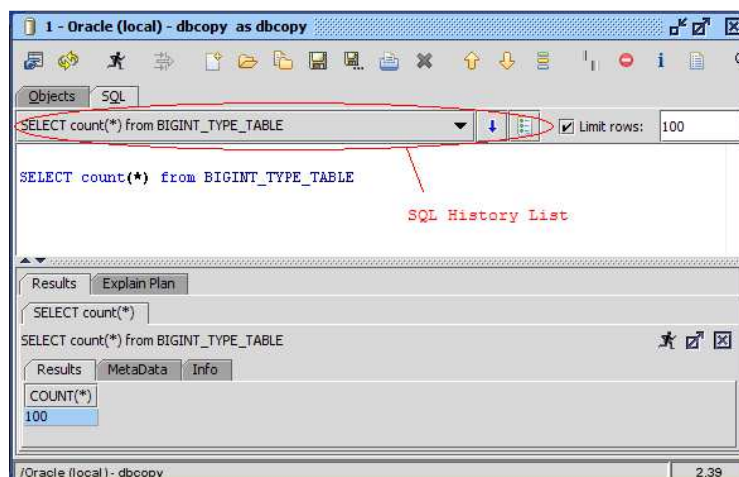




Figure 13. SQL tab - SQL History List

This list can then be used to find and re-execute previously executed statements. The drop-down list allows you to pick the SQL statement, and clicking the blue arrow () will copy the statement into the editor. Additionally, the history list can be searched using the SQL History dialog which is accessed by the icon () which is next to the blue arrow. The SQL History window is shown in Figure 14.

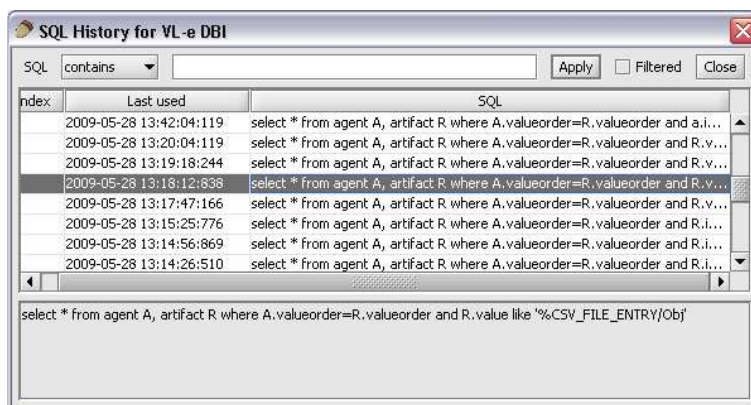


Figure 13. SQL tab - SQL History Window

2. Editing Data in the SQL Results Tables

The data displayed within the SQL Results may be edited in the same manner as the data within the [Contents Tab](#). You should read that section to understand the capabilities and limitations of data editing, including the use of the [Popup window](#) to import and export data. In addition, there are two things that you need to know about editing in the SQL Results:

1. To enable editing in the SQL Results tables, you must set the "SQL Results" entry in the Session Properties to "Editable Table".
2. Editing is only allowed when the SQL that was executed consists of:

"SELECT FROM "
or
"SELECT FROM WHERE ..."

In addition, the name of the table must be unique within the entire DB. This means that if you have multiple catalogs or schemas which have that same table name defined in them, e-DBI cannot determine which to use and thus will not allow data to be updated in the DB. (e-DBI is just a bit stupid on this point: it will let you edit the data, and then tell you that it cannot update the DB when you exit the cell being changed.)

B. The Contents Tab and Editing Data

e-DBI allows you to easily view and change the data in a single table. Data may be changed just by typing the new values when you are viewing the table in the Contents tab under the Objects view. The data may also be viewed and edited in a larger popup window, which has more capabilities, by double-clicking on the cell. There are some small differences between the in-cell editing and editing in the [Popup](#), so be sure to read the [Using the Popup Window](#) information.

1. Viewing Data in the Contents Tab

After connecting to a database, you may view the data in a single table by using the Contents Tab.

To get to the contents tab, first select the Objects view. Next, look under the tree in the window on the left. The exact organization of this window may be different from one DBMS to another. When you find the list of table names or views, click on the name of the table that you wish to view, and then in the window on the right click on the tab labeled "Contents".

The data in the table will be displayed in rows and columns based on the format selected (see [Setting the Format and Editing Mode](#)).

The column ordering is the order of the rows in the table definition. That order may be changed by drag-and-drop, which means clicking on the column header and holding the mouse button down while dragging the column horizontally to another location. Columns are initially displayed with a moderate width so that you can see a reasonable amount of data in as many columns as possible within the limited window space. You may increase or decrease a column's width by click-and-hold on the space between two columns.

The rows are initially unordered. However, after retrieving the data, you may sort the rows based on the values in a column by single-clicking on a column header. That will sort the rows into an order with the values in the selected columns starting with the "smallest" value. Clicking on the same column header again will reverse the order ("largest" first). You can increase or decrease the number of rows retrieved (see the Session Properties, Object Tree).

There are a few special cases for how data is displayed.

1. If a cell contains Newlines (which is possible in the various VARCHAR and CLOB data types), the cell background is colored Cyan and the Newline characters are explicitly shown as "\n" (unless changed in the [Data Type Controls](#)). If the table is editable, the Cyan background means that the cell contents cannot be edited in a the cell, but using the [Popup window](#) (see [Using the Popup Window](#)).
2. BLOB (Binary Large Object) and CLOB (Character Large Object) fields are initially displayed as "<Blob>" and "<Clob>" respectively. This is done for performance reasons, since those data types are often quite large. When you select the Blob or Clob field, the entire data is read in and displayed in the cell. Alternatively, you may use the Session Properties -> Format screen to limit the amount of the Blob/Clob data read and displayed when the Contents Tab is first displayed, in which case the number of characters you have selected will be displayed in each Blob or Clob cell. This may be necessary when the Blob or Clob contains data needed to identify specific rows in the table. When there is more Blob/Clob data for a cell than is displayed in that cell, an ellipsis ("...") is added to the end of the displayed data so you know that the data has been truncated for display. As before, when you select the cell, the whole Blob or Clob will be read and displayed.

Warning: Some DBMSs that claim to have BLOB and CLOB data types actually implement those types using BINARY, VARCHAR, or other types of fields. In those cases, the cells will act as they would for the BINARY, VARCHAR, etc., data types, and not as described above for the BLOB/CLOB types. (There is a functional difference in that "real" BLOB/CLOB types return a "locator" or "reference" object rather than the actual data. Using the locator allows e-DBI to defer reading the actual data until you need it rather than reading it all during the initial display of the Contents Tab.) To see if your DBMS uses real BLOB/CLOB types, go to the Objects view, then select the top element in the tree in the left window and "Data Types" in the right window. In the column labeled "DATA_TYPE", real BLOBs will have the value 2004 and real CLOBs will show 2005. If your DBMS does not have 2004 and 2005 listed under the "DATA_TYPE" field, then it does not support the BLOB/CLOB functionality provided by e-DBI. As an example, the MySQL DBMS claims to support BLOBs, but in the "Data Types" table it shows that the "TYPE_NAME" of "BLOB" is actually using the "DATA_TYPE" of "-4 [LONGVARBINARY]", so a "BLOB" field in MySQL will be treated by e-DBI as a LONGVARBINARY and not as a "BLOB".

2. Enabling Editing

When e-DBI is initially set up, the Contents Tab will be set in a read-only mode. This is a safety precaution. Once a table is made editable, changing the contents of a cell causes the database to be immediately updated. While extremely convenient, this is dangerous. With this capability enabled, it becomes very easy to accidentally destroy data, violate some consistency checks, mis-direct references between tables, etc.

You can set e-DBI so that tables are editable all the time, or you can enable editing on a specific table as you are working on it. Either option may be set as your default mode of operation, and either option may be selected for use on a specific table.

Editing may only be done in Contents tab on the single table view, not on the results of manually entered SQL queries involving join operations or aggregate functions.

3. Setting the Format and Editing Mode

To set the default format and editing mode for the Contents Tab, select the *Session Properties* option from the *File* Menu and click on the *General* tab. The **Table Contents** dropdown list gives the following options:

1. Table
2. The data in the Contents tab is a read-only table. To enable editing on this table, from the Content Tab, click on the right mouse button and select the **Make Editable** option. This makes that table *and only that table* editable.
3. Text
4. The data in the Contents tab is shown as simple formatted text and is read-only. To enable editing on this table, from the Content Tab, click on the right mouse button and select the **Make Editable** option. This makes the table *and only that table* editable.
5. Editable Table
6. The data in the Contents tab is an editable table. Changes made to the data in this view are made to the database.

After connecting to a database, you may change the format or editing mode for the data in the Contents tab by the *Session Properties* option from the *Session* and clicking on the *General* tab and changing the selection for *Table Contents*.

Note that the individual data items may be displayed in different formats as described in "[Data Type Controls](#)".

4. How To Edit

The following steps briefly describe how to edit data in the table:

1. Connect to the database.
2. Select a table in the object tree and select the *Contents* tab to view the data.
3. Make sure the table is editable. See [Setting the Format and Editing Mode](#) above.
4. Select the cell that you want to edit. This may be done either by using the keyboard (tab and enter keys) to move around the table to reach the cell, or you may click on the cell with the mouse. Note that if a cell cannot be edited (see [Limitations](#)) you may select the cell, but you will not be allowed to edit the data. When a cell is being edited, its background will change to Yellow to let you know that you may be changing the contents of the database. Cells with a Cyan background may be edited only by using the Popup window (see [Using the Popup Window](#)).
5. If you select the cell using the mouse, the background immediately turns Yellow and the cursor is set to the place where you clicked in the cell. If you select the cell by using the tab or enter keys, the background remains white until you type the first character to be added (or use the delete key). That character is added to (or deleted from) the end of the text displayed in the cell, and the cell background changes to Yellow.
6. Change the data in the cell to look the way that you want it to. Editing is done in the usual way by typing characters, using delete to erase, and using the mouse to select a point to insert or characters to delete. There are a couple of specialized features that are described later in this section.
7. When the data in the cell looks the way you want, leave the cell by pressing the tab key, the enter key, or clicking on another cell with the mouse. This causes the data in that cell to be updated in the database.

Hints: Data edition in SQL tab

1. If the data in the cell gets really messed up and you are confused about what is going on, press the <delete> key until the original data appears (see below). If that doesn't work then immediately close the database connection without leaving the cell (<ctrl>W or click the close button). This will abort the operation and avoid submitting the updates to the database.
2. If the cell contains a NULL value, that is shown as the string "<null>". To change that to a non-null value, just type the new data. When you enter the first character of the data, the "<null>" will be replaced by that character.
NOTE: This means that e-DBI will not allow you to enter a string with the value "<null>" into a nullable column, since that string will be interpreted by e-DBI to be the null value. Also, if you have a field in the database containing the string "<null>", do not edit that field using e-DBI since the value will be replaced by the null value.
3. To change the value of a nullable cell to NULL, just delete all of the data in that cell. When you delete all of the data in a cell, the contents of the cell will be set to "<null>". Since the VARCHAR data types allow a non-null empty string, for those cells you must delete all of the data so that the cell is blank, and then press delete. The VARCHAR cell will then be set to <null>.
4. If a cell is not nullable, deleting the last character in the cell will leave the cell blank. You may then enter new data for the cell.
5. To restore the cell to its original value, delete all of the data in the cell, then press the delete key one more time. This

will cause the cell to be reset to its original value. For cells that allow nulls, continually pressing the delete key causes a cycle:

original data -> delete data -> <null> -> original data

and for cells that do not allow nulls:

original data -> delete data -> blank cell -> original data

Thus, if you mess up the contents of a cell, repeatedly press the delete key until you see the original data reappear.

When leaving the cell, if the cell data is identical to the original contents of the cell, the database will not be updated.

5. Inserting and Deleting Rows

In order to be able to insert or delete rows in the table using the *Contents Tab*, the table must be set to editable (see [Enabling Editing](#))

To insert a row into the table, click the right mouse button, and choose the "Insert row" option from the menu This brings up a separate window showing the columns of the table with two rows The first row contains the data to be filled with default values as defined for the table's column already filled in. The second row shows information about the field including the type of data, the field length (if known), and so forth.

You should edit the values in the first row in the same way that you would if that row already existed in the table. When finished with the data entry, click the "OK" button to have the changes submitted to the table.

If there are no rows in the table when you open the Contents Tab clicking the right mouse button brings up a menu with only the "Insert Row option You may use this in the normal manner to enter the first row in the table

To delete one or more rows from the table select the rows to be deleted, then click the right mouse button and choose the "Delete Rows" option from the menu

6. Printing

Any data that is displayed in a Table may be printed. This includes all of the meta-data tables displayed in the Object View as well as the Contents Tab and SQL Results tables. To print, first make sure that the Session Properties for the type of information (Meta Data, Table Contents, or SQL Results) has been set to either "Table" or "Editable Table". Then, with the cursor over the table to be printed, click on the right mouse button and click on "Print". This will bring up a dialog box, in which, you can adjust the parameters of the print job and print the selected table.

Tables that are too wide or too long to fit onto one page will be printed across multiple pages that can be pieced together to show the entire table.

If the Session Properties for "Table Contents" or "SQL Results" have been set to "Text", then the data within the Contents Tab or SQL Results tables (respectively) may be copied and pasted into a separate window for printing.

C. Using the Popup Window

To use the Popup window, double-click in the cell to start the Popup, then edit the text in that window. Editing uses the same mechanism as the in-cell editing where repeatedly typing the backspace key eventually restores the original data in the window. One difference from the in-cell editing is that, depending on the data type being displayed, you may enter newline and tab characters into strings. once you you are done with editing, click on the "Update DB" button to commit the changed data into the Database. The "Cancel" button closes the Popup window without applying the changes to the Database.

1. Editing within the Popup

For simple data (e.g. integers, floats, or simple strings) you will be able to edit the data directly in the cell, but there are cases where you will need to use a separate window, in order to perform the data edition in a convenient manner. A Popup window is more convenient for text edition, when:

1. the data is too long to comfortably work with in the space provided by the cell
2. the data contains newlines or tabs (which are not displayed correctly in the cell)
3. the data must be represented in a complex way, such as multiple lines (e.g. an Array type)
4. the data is potentially so large or complex. In this case, the cell may display just the name of the data type, something like "<Blob>", while the Popup window would show the actual contents of that field.

For these reasons the data may not be editable in the cell but still be editable in the Popup. If the data is not editable in the cell, you will need to bring up the Popup window to know whether the data can be edited there.

For Binary data types, the data may be changed from the default Hex representation into Decimal, Octal or Binary formats. You may also choose to view bytes of data that correspond to printable ASCII characters rather than in their numeric form. This may be useful if the file contains strings, in which case you could see "ContentsTab.java" instead of "43 6f 6e 74 65 6e 74 73 54 61 62 2e 6a 61 76 61".

The Popup window provides line-wrap and word-wrap functions. These are accessed in the menu brought up by right-clicking in the text area. Both of these options are toggle-switches that flip from/to on-off each time you click on them. The word-wrap function operates only when the line-wrap function is on.

2. Import, Export and Editing with an External Command

The Popup window also allows you to import data into a cell, export data from the cell to a file, and operate on the data using an external command.

1. **Importing data into a cell:** Enter the name of the file containing the data in the "File:" field, then click on "Import".
2. **Exporting data from the cell into a file:** Enter the name of the target file in the "File:" field, then click on "Export".
3. **Operating on the data with an External Command:** This function provides a single-click operation to export the

data to a file, operate on that file with an external process (e.g. MS Word, VI, various graphical editors, etc.), then read the result back from the file when editing is completed. To do this you must:

- n Enter a file name in the "File:" field or leave it set to "<Temp File>". If you don't specify the file name, a temporary file will be created for you. Be warned that if you specify a file, it will be deleted after the editing is completed.
- n Enter a command in the "Command:" field. In the command you should typically specify the name of the file to edit, which could be done by using "%f" as a parameter for the name of the file to go in that command.

An example, could be:

```
xterm -e vi myfilename
or
xterm -e vi %f
```

- n Click on "Execute". This causes the cell data to be exported to the selected file and the command to be executed. When the external process exits, e-DBI imports the data from that same file back into the popup and deletes the file. IMPORTANT: e-DBI does NOT automatically update the database with this data. To put the edited data into the database, click on the "Update DB" button.

4. Notes:

- n e-DBI does not provide a window for the command to operate in, so you may need to provide for that as part of the command. For example, to run VI on Linux you may need to enter the command:

```
xterm -e vi %f
```

If you simply enter "vi %f" as the command, you will not get a window to operate in. Other commands (e.g. gimp) provide their own windows, so the extra step will not be necessary for them.

- n In all of the above cases, the entries you make in the "File:" and "Command" fields will be associated with that column so that you do not need to re-type them the next time that you edit that field.
- n If you just want to set the default values for a particular column, you can enter the "File:" and "Command:" field information and click on "Apply".

D. Limitations

Some DBMSs implement "standard" data types in non-standard ways. For example, we know that many DBMSs claim to implement the BLOB data type but they actually store that data using a BINARY or LONGVARBINARY field. e-DBI uses the *actual* data type as defined in the DATA_TYPE field in the Columns tab for the table, not the TYPE_NAME. Thus it is possible that you may think you are working with one type of data but actually using something else so controls and limitations on specific data type (e.g. limiting the data read when loading the Contents tab may not work as expected).

e-DBI allows editing only on cells that are defined using the basic standard SQL data type (see [Data Types](#)). If your DBMS provides other data types, you can create a plug-in that supports it. (see the package fw/datasetviewer/cellcomponent/CellComponentFactory for more information).

Some attempt has been made to validate user input for each of the types. For example, INTEGER fields allow only digits and '-' as input. However, not all of the necessary information is supplied by all DBMSs, so we are not able to perform a complete validation on the user input. For example, we cannot check that a value entered for a SMALLINT field will fit into a SMALLINT column in a specific DBMS. Also, the drivers for each DBMS provide different amounts of information for the same field type in different situations, and e-DBI cannot validate the input when such data is missing.

The "validation of last resort" is done when we try to commit the changes to the database; at this stage updates either succeed or fail, and eventual errors are displayed to the user.

- n VARCHAR fields containing tabs or newlines may only be edited in the Popup window. Fields that can only be edited in the Popup are displayed with a cyan background.
- n VARCHAR fields that may be set to null cannot be set to the string value "<null>" (without the quotes). VARCHAR fields in the database may include that value as long as you do not try to edit them.
- n Tables with a large number of columns or with a column containing a very long value (not including BLOB/CLOB data types) may get an error when trying to update the DB because the WHERE clause is too long. In this case you should open the table in a session, go to the **Session** menu in the main window, and select the entry "Limit cell edit WHERE clause size". That tool lets you specify which columns e-DBI should use in the WHERE clause when doing the requested update on the cell contents. See [Session Menu](#).

E. Summary of How it works (and why you may care)

The database updates are based on the assumption that the list of column names and their values for the current row are unique, or if they are not unique, that all of the rows with those values are to be updated. For example, if you have a database with columns a, b, and c, and in a particular row you have the values

```
a=34 b='your name' c=55
```

and you change column c to be 66, then e-DBI does the following:

1. Query the database for how many rows contain a=34, b='your name', and c=55. If there are more than one, then the update will affect all of those rows, so **e-DBI** asks if you want to do that before continuing.
2. Query the database for how many rows contain a=34, b='your name' and c=66. If there are any existing rows with those values, changing the current row to c=66 will make the current row identical to the existing row(s), so **e-DBI** asks if you want to do that before continuing.
3. Create and execute the following:

```
update <table> set c=66 where
a=34 and b='your name' and c=55
```

To help distinguish rows that are similar to each other, any DBMS that provides a pseudo-column to uniquely identify each row (such as Oracle's ROWID) has that pseudo-column included in all queries.

The problem with this is that some columns cannot be differentiated in this way, most notably BLOBs and CLOBs. Thus, if your table contains BLOB or CLOB columns, **e-DBI** may think that rows are identical because all of the non-BLOB/CLOB fields are identical, but the rows are actually not the same because the contents of the BLOB/CLOB fields differ. This could lead to "erroneous" warning messages.

F. Quirks, Oddities and Known Bugs and work-arounds

- After error or warning messages, the focus is lost. You will need to click on a cell in order to continue editing.
- When using the enter key to leave a cell, you must press it two times. The first time causes the data to be updated in the database, and the second moves to the cell below the current one.

IV. Data Types

This is a quick description of how **e-DBI** handles various data types when displaying the results of SQL statements. **e-DBI** uses the column type (`ResultSetMetaData.getColumnType(...)`) from the metadata for the result set to determine how to display the column.

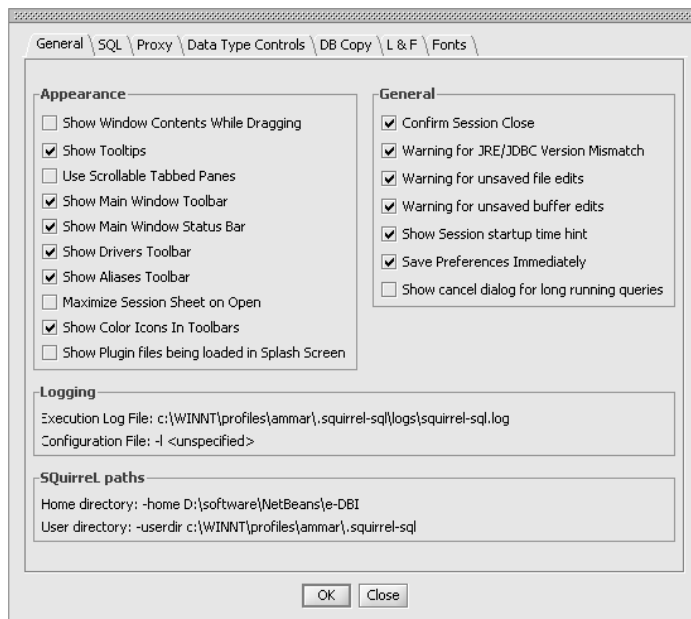
SQL Data Types java.sql.Types	e-DBI Action
NULL	<Null> is displayed.
BOOLEAN and BIT	If the retrieved column data is a <i>java.lang.Boolean</i> object then the appropriate <i>true/false</i> is displayed. If the retrieved column is a <i>java.lang.Number</i> then a non-zero value will display <i>true</i> while a zero value will display <i>false</i> . Otherwise the column data is converted to a string and if this string equals (ignoring case) "true" then <i>true</i> is displayed else <i>false</i> .
TIME	A <i>java.sql.Time</i> object is retrieved from the result set and displayed in <i>hh:mm:ss</i> format.
DATE	A <i>java.sql.Date</i> object is retrieved from the result set and displayed in <i>yyyy-mm-dd</i> format.
TIMESTAMP	A <i>java.sql.Timestamp</i> object is retrieved from the result set and displayed in <i>yyyy-mm-dd hh:mm:ss.ffffff</i> format. WARNING: When updating any column in a row containing TIMESTAMP fields, the MySQL DBMS will automatically set the first TIMESTAMP field in the row to the current time. This is a "feature" of MySQL, and not a bug in e-DBI.
BIGINT	If the retrieved column type is a <i>java.lang.Long</i> or a <i>java.lang.Number</i> then it is stored as a <i>java.lang.Long</i> and displayed as a base ten signed long. Otherwise the column data is converted to a string which is then parsed as a base ten signed long.
DOUBLE, FLOAT, REAL	If the retrieved column type is a <i>java.lang.Double</i> or a <i>java.lang.Number</i> then it is stored as a <i>java.lang.Double</i> and displayed as a string representation of the double. Otherwise the column data is converted to a string which is then parsed as a double.
DECIMAL, NUMERIC	If the retrieved column type is a <i>java.math.BigDecimal</i> or a <i>java.lang.Number</i> then it is stored as a <i>java.math.BigDecimal</i> and displayed as a string representation of the <i>BigDecimal</i> . Otherwise the column data is converted to a string which is then parsed as a <i>BigDecimal</i> .
INTEGER, SMALLINT, TINYINT	If the retrieved column type is a <i>java.lang.Integer</i> then it is stored as a <i>java.lang.Integer</i> and displayed as a string representation of the <i>Integer</i> . Otherwise the column data is converted to a string which is then parsed as a <i>Integer</i> .
CHAR, VARCHAR, LONGVARCHAR	The column data is displayed as a string. Note that if the data includes a newline character, that particular data item cannot be edited in the cell within the table, but if you double-click on the cell to get the Popup window, you will be able to edit the data there.
BINARY, VARBINARY, LONGVARBINARY	In the SQL tab the column data is displayed as a string. The ContentsTab in the Object view displays each byte using Hex format. In the Popup editing window, you may choose to display these in Hex, Decimal, Octal, or Binary, and you may choose to display values representing printable ASCII characters as those characters (e.g. to see strings embedded in the binary object).
BLOB	If the Global Preferences, Data Type Controls says to retrieve blobs then the column data is displayed as bytes. Note that the Global Preferences, Data Type Controls can specify the number of bytes of the blob to read in during the initial setup of the ContentsTab table. If you do not specify a limit, the default operation is to not read in any data and to display "" in the column. Clicking on an individual cell causes the entire BLOB data to be read and displayed at that time. Note: BLOB fields may be very large, and this can cause the Popup window to take some time to start up (e.g. 30 sec. for a BLOB containing a relatively small sized photo).
CLOB	If the Global Preferences, Data Type Controls says to retrieve clobs then the column data is displayed as a string. Note that the Global Preferences, Data Type Controls can specify the number of characters of the clob to read in during the initial setup of the ContentsTab table. If you do not specify a limit, the default operation is to not read in any data and to display "" in the column. Clicking on an individual cell causes the entire CLOB data to be read and displayed at that time.
OTHER	If the Global Preferences, Data Type Controls says to retrieve columns of this type then the column data is displayed as a string.
Any other type	If the Global Preferences, Data Type Controls says to retrieve all other data types then the column data is displayed as a string.

The JDBC specification (in file `java.sql.Types`) defines several codes whose semantics are not understood by **e-DBI**: `JAVA_OBJECT`, `DISTINCT`, `STRUCT`, `ARRAY`, `REF`, and `DATALINK`. These are all handled as "Any other type" as described above.

V. Global Preferences

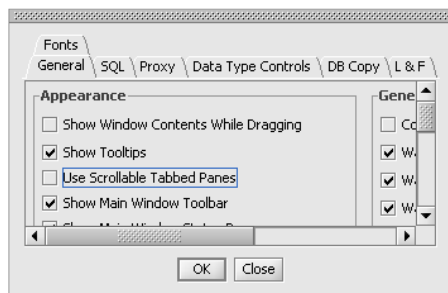
Global Preferences specify configuration settings for the application.

A. General Tab

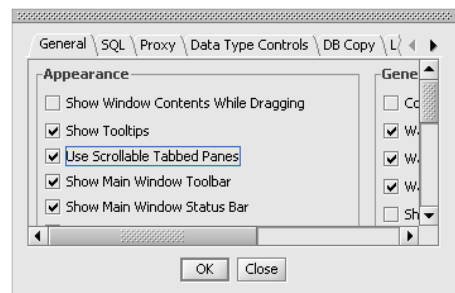


1. General Tab (Appearance)

- **Show Window Contents While Dragging** - If checked this shows the contents of windows as they are being dragged. If unchecked only the outline of the window will be shown. Uncheck for a speed improvement on slow machines.
- **Show Tooltips** - If checked then tooltips (or hints) will be shown when the mouse "hovers" over some controls.
- **Use Scrollable Tabbed Panes** - If checked will display the tabs in tabbed folders in a scrollable region rather than wrapping them when all tabs will not fit within a single run.



Stacked Tabs



Scrollable Tabs

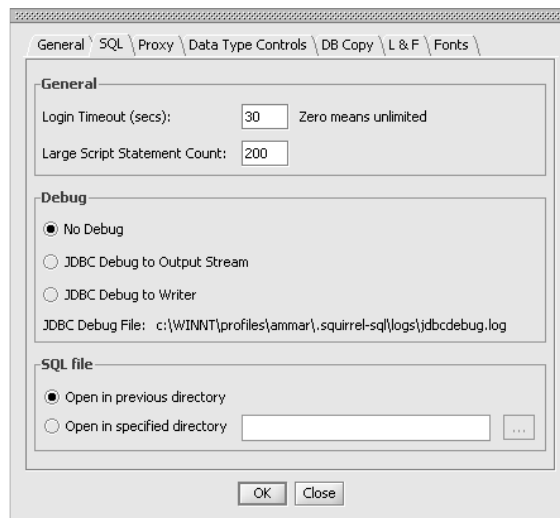
Show Main Window Tool Bar - If checked then the tool bar will be displayed in the main window.

- **Show Main Window Status Bar** - If checked then the status bar will be displayed in the main window.
- **Show Drivers Tool Bar** - If checked then the tool bar will be displayed in the Aliases Window.
- **Show Aliases Tool Bar** - If checked then the tool bar will be displayed in the Aliases Window.
- **Maximize Session Sheet on Open** - If checked then when a session is opened it will automatically maximize to the whole space available within the e-DBI window. If not checked, it is opened using a smaller part of the space.
- **Show Color Icons in Toolbars** - If checked then the icons in the toolbars are shown in color. If not checked, the icons are in black-and-white.

2. General Tab (Logging)

- **Execution Log File** - This (display only) setting tells you the name (and location) of the logging file for e-DBI. If the file name is too long for the label then the full name will be displayed in a tooltip. For more information see [Logs](#).
- **Configuration File** - This (display only) setting tells you the name (and location) of the logging configuration file for e-DBI. If the file name is too long for the label then the full name will be displayed in a tooltip. For more information see [Logs](#).

B. SQL Tab



1. SQL Tab (General)

- ⌋ **Login Timeout** - Sets the maximum time in seconds that a JDBC driver will wait while attempting to connect to the database.
- ⌋ **Large Scrip Statement Count** - By default, a script is considered large if it has more than 200 SQL statements selected for execution in the script. When this is set to zero, large script execution support is disabled. Large script support consists of the following behavior during execution:
 - ⌋ Only one message is written in the message panel at the bottom of the session window at the end of running many statements summarizing the details of execution (timing, statement type and count, etc).
 - ⌋ The message "SQL Statement x of y <some sql...>" is still displayed in the results panel to indicate progress.
 - ⌋ The history bar isn't updated during large script execution.
 This feature can dramatically improve UI responsiveness when executing many statements in large scripts (especially those with statement counts in the thousands).

2. SQL Tab (Debug)

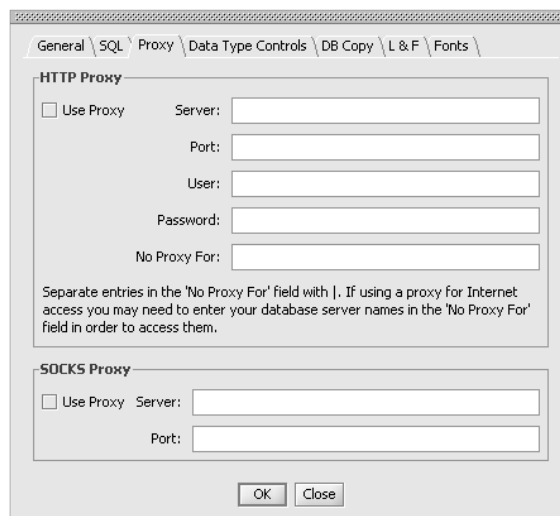
- ⌋ **No Debug** - If checked then no debugging information is output from e-DBI.
- ⌋ **JDBC Debug to Output Stream** - If checked tells the JDBC drivers to write debugging information to the JDBC Debug File. Depending on the JDBC driver you are using this may generate a lot of output and significantly slow down e-DBI.
- ⌋ **JDBC Debug to Writer** - If checked tells the JDBC drivers to write debugging information to the JDBC Debug File. Depending on the JDBC driver you are using this may generate a lot of output and significantly slow down e-DBI.
- ⌋ **JDBC Debug File** - If JDBC Debug is enabled then this (display only) setting is the file that the debug information will be written out to. If the file name is too long for the label then the full name will be displayed in a tooltip.

The **JDBC Debug to Output Stream** has been deprecated by JavaSoft in favour of the **JDBC Debug To Writer** option. However some older JDBC drivers may use the **Output Stream** rather than the **Writer**. Other drivers (such as the oracle 8 JDBC driver) may write different information to the **Output Stream** than they do to the **Writer**.

3. SQL Tab (SQL File)

- ⌋ **Open in previous direcorey** When opening files in the SQL editor, sets the location in the file chooser to be the last directory that a script file was opened from.
- ⌋ **Open in specified direcorey** When opening files in the SQL editor, sets the location in the file chooser to be the directory that is specified in the text field.

C. Proxy Tab



1. Proxy Tab (HTTP Proxy)

- ⌋ **Use Proxy** - If checked then a HTTP proxy server will be used for connecting to servers.

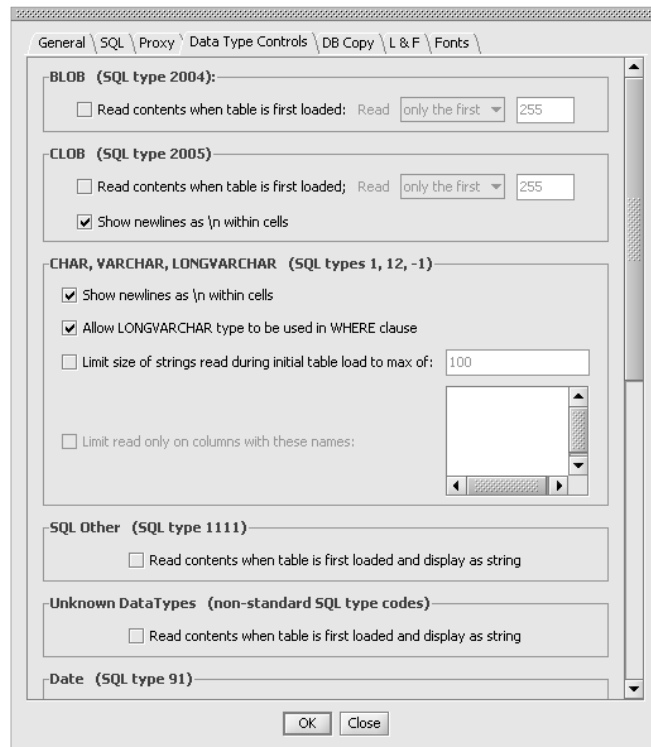
- ⌋ **Server** - The name (or IP address) of your proxy server.
 - ⌋ **Port** - The port number to use on the proxy server.
 - ⌋ **User** - The user name to use to log onto the proxy server.
 - ⌋ **Password** - The password to use to log onto the proxy server.
 - ⌋ **No Proxy For** - A list of servers (separated by |) that do not need to be accessed through your proxy server.
2. **Proxy Tab (SOCKS Proxy)**

- ⌋ **Use Proxy** - If checked then a SOCKS proxy server will be used for connecting to servers.
- ⌋ **Server** - The name (or IP address) of your proxy server.
- ⌋ **Port** - The port number to use on the proxy server.

D. **Data Type Controls Tab**

The contents of this tab may change depending on what Plugins you have loaded. Plugins are allowed to define new Data Types or override the existing Data Types, and that may add or remove controls from this window. The following describes the standard Data Type controls with no changes made by any Plugins.

Even with only the default Data Type Controls defined, there are still two screens of them to describe.



1. **BLOB (SQL Type 2004)**

- ⌋ **Read contents when table is first loaded** - If checked then the actual contents of the BLOB is read when the table is first loaded. This is usually not recommended because it usually degrades performance. If this is not checked, then the data will initially display as ">Blob<" in the table, and the entire contents will be read in and displayed when you click on that cell.
- ⌋ **Read** - If reading the contents during initial table load, select whether to read the entire contents or just part of the data. If only part of the data is read, then the entire contents will be read when you click on that cell.
- ⌋ **(number)** - When reading part of the data during initial table load, how many bytes to read.

2. **CLOB (SQL Type 2005)**

- ⌋ **Read contents when table is first loaded** - If checked then the actual contents of the CLOB is read when the table is first loaded. This is usually not recommended because it usually degrades performance. If this is not checked, then the data will initially display as ">Clob<" in the table, and the entire contents will be read in and displayed
- ⌋ **Read** - If reading the contents during initial table load, select whether to read the entire contents or just part of the data. If only part of the data is read, then the entire contents will be read when you click on that cell.
- ⌋ **(number)** - When reading part of the data during initial table load, how many characters to read.
- ⌋ **Show newlines as \n within cells** - If checked, when the data contains newlines, the newlines will be displayed as "\n" within the cell. When unchecked, the newlines are ignored within the cell. The difference is between seeing:

```

first line\nsecond line\nthird line
and
first linessecond linethird line

```

Note that this option does not affect display in the Popup window, where newlines actually become new lines.

3. **CHAR, VARCHAR, LONGVARCHAR (SQL Types 1, 12, -1)**

- ⌋ **Show newlines as \n within cells** - If checked, when the data contains newlines, the newlines will be displayed as "\n" within the cell. When unchecked, the newlines are ignored within the cell. The difference is between seeing:

```

first line\nsecond line\nthird line
and
first linessecond linethird line

```

Note that this option does not affect display in the Popup window, where newlines actually become new lines.

Allow LONGVARCHAR type to be used in WHERE clause - When checked, the LONGVARCHAR type is used in WHERE clauses just as other types. When unchecked, columns of type LONGVARCHAR are not used in the WHERE clause when performing edits in the cell data. This is necessary because some databases (e.g. Oracle, DB2) do not allow LONGVARCHAR data types to be used in WHERE clauses and generate odd error messages when they are seen.

Limit size of strings read during initial table load ... - When checked, the amount of string data loaded during the initial table read is limited to a maximum size (see next field). When unchecked, the amount of data is not limited. (Note: This was originally built as a performance improvement, but the implementation actually reads all of the data from the DB in either case and just displays or does not display all of it, and thus may not make an impact on performance.) If the initial data read is limited, the entire data contents will be displayed when the user clicks on the cell.

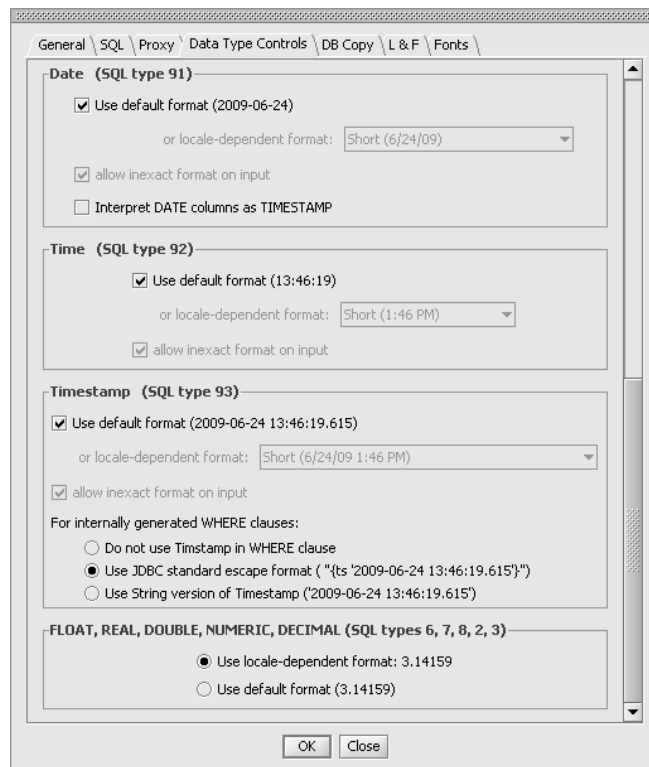
... to max of - When limiting the size of the data, this defines how many characters to display.

Limit read only on columns with these names: - When limiting the size of data during the initial table load, only apply the limit to columns with the names listed in this box. All other columns load all of the data. (This was built as a technology demo to show how Data Type Controls, which are global to all columns of the given SQL type(s), can be applied selectively to a limited set of columns.)

4. SQL Other (SQL Type 1111)

Read contents when table is first loaded and display as string - If checked then columns of this SQL Type will be read by e-DBI and displayed using the toString() function.

Warning: Since fields of this data type are not known to e-DBI and have been explicitly defined as NOT being of a normal, known type, attempting to display them as Strings may result in unintelligible displays, and may cause exceptions during processing.



5. Unknown DataTypes (non-standard SQL type codes)

Read contents when table is first loaded and display as string - If checked then columns of this SQL Type will be read by e-DBI and displayed using the toString() function.

Warning: Since fields of this data type are not known to e-DBI and have been explicitly defined as NOT being of a normal, known type, attempting to display them as Strings may result in unintelligible displays, and may cause exceptions during processing.

6. Date (SQL Type 91)

Use default format(yyyy-mm-dd) - If checked then dates will be displayed in the JDBC default format as shown. If unchecked, then the "locale-dependent" formats are made available.

or locale-dependent format: - When not using the default format, this allows you to select from one of the other available formats.

allow inexact format on input - If checked then e-DBI allows some flexibility when you type a new value for the field.

7. Time (SQL Type 92)

Use default format(hh:mm:ss) - If checked then times will be displayed in the JDBC default format as shown. If unchecked, then the "locale-dependent" formats are made available.

or locale-dependent format: - When not using the default format, this allows you to select from one of the other available formats.

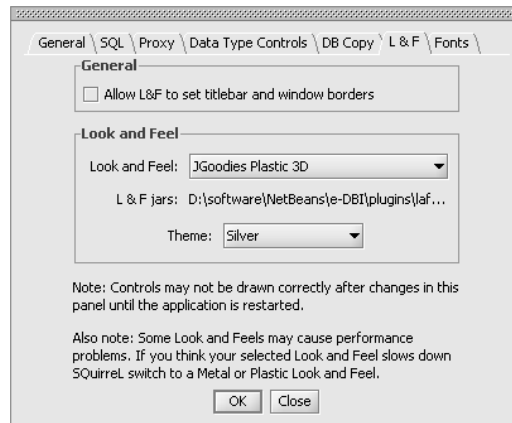
allow inexact format on input - If checked then e-DBI allows some flexibility when you type a new value for the field.

8. Timestamp (SQL Type 93)

Use default format(yyyy-mm-dd hh:mm:ss.msec) - If checked then timestamps will be displayed in the JDBC default format as shown. If unchecked, then the "locale-dependent" formats are made available.

- | **or locale-dependent format:** - When not using the default format, this allows you to select from one of the other available formats.
- | **allow inexact format on input** - If checked then e-DBI allows some flexibility when you type a new value for the field.

E. L&F Tab



This tab is installed by the Look and Feel (L&F) plugin which is part of the "standard" plugin set. If you didn't choose to install the "standard" plugins then this tab will not appear.

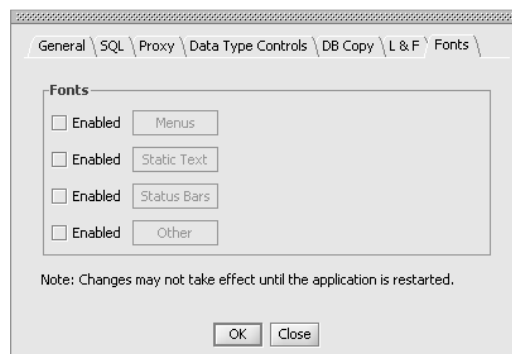
1. L&F Tab (General)

- | **Allow L&F to set titlebar and window borders** - allows you to specify whether the Look and Feel can set these, otherwise the Window Manager will set them. This setting only takes effect for newly created windows.

2. L&F Tab (Look and Feel)

- | **Look And Feel** - allows you to specify which L&F family to use. This setting only takes effect for newly created windows.
- | **Theme** - If the selected L&F supports themes, this allows you to specify which one to use

F. Fonts Tab



1. Fonts Tab (Fonts)

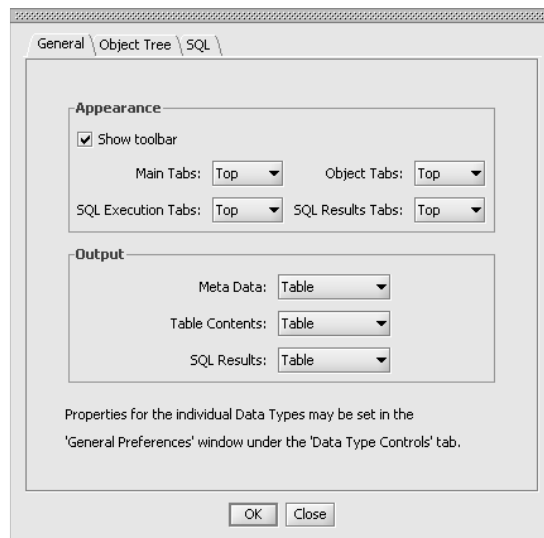
- | **Menus** - If enabled this will override the default font with the selected font for menus/menu items in the toolbar and the popup menu. Pictured below is the main frame menubar with Ravie 12-point Bold font.
- | **Static Text** - If enabled this will override the default font with the selected font for labels on Tabs and Object Tree Nodes. The picture below shows a session window with Ravie 12-point Bold font.
- | **Status Bars** - If enabled this will override the default font with the selected font for labels on status bars. The picture below shows a session window status bar and the main application status bar with Ravie 12-point Bold font.
- | **Other** - If enabled this will override the default font with the selected font for labels in many places not already covered by the preceding settings. The picture below shows "other" labels with Ravie 12-point Bold font.

The font tab is also installed by the Look and Feel (L&F) plugin which is part of the "standard" plugin set. If you didn't choose to install the "standard" plugins then this tab will not appear. There may be other tabs present in the Global Preferences dialog (plugin-specific) which are not listed here. See the plugin documentation for details. For preference tabs installed by plugins, the tab name should match the plugin name.

VI. New Session and Session Properties

The "New Session Properties" dialog allows you to specify settings for future session. To change the settings for existing sessions use the "Session Properties" menu option. The descriptions included here are for the tabs created by the core product. Plugins may add other tabs to this window. See the Plugin-specific help for descriptions of the tabs added by that plugin.

A. General Tab



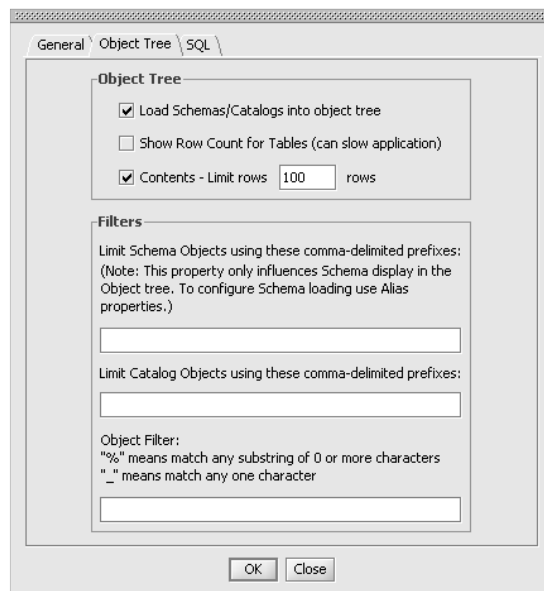
1. General Tab (Appearance)

- ▮ **Show Tool Bar** - If checked then display a toolbar on the session window.
- ▮ **Main Tabs** - Defines where the Main tabs (ie: "Objects" and "SQL") in the session window go relative to the data being displayed. This may be set to top, bottom, left or right.
- ▮ **Object Tabs** - Defines where the tabs in the Object window (eg: Metadata, Contents, Status, etc.) go relative to the data being displayed. This may be set to top, bottom, left or right.
- ▮ **SQL Execution Tabs** - Defines where the tabs for the sets of results from SQL execution in the SQL window go relative to the data being displayed. This may be set to top, bottom, left or right.
- ▮ **SQL Results Tabs** - Defines where the tabs within a single SQL result window ("Results", "Metadata" and "Info") go relative to the data being displayed. This may be set to top, bottom, left or right.

2. General Tab (Output)

- ▮ **Meta Data** - Specify the output type (text, read-only table, or editable table) for meta data displays.
- ▮ **Table Contents** - Specify the output type (text, read-only table, or editable table) for the Contents tab in the Object display.
- ▮ **SQL Results** - Specify the output type (text, read-only table, or editable table) for SQL result displays. (Note that there are additional limits on what can or cannot be edited in the results of SQL queries, so setting this to "Editable table" may not result in an editable output for any given query. See [Editing the SQL Results](#) for details.)

B. Object Tree Tab



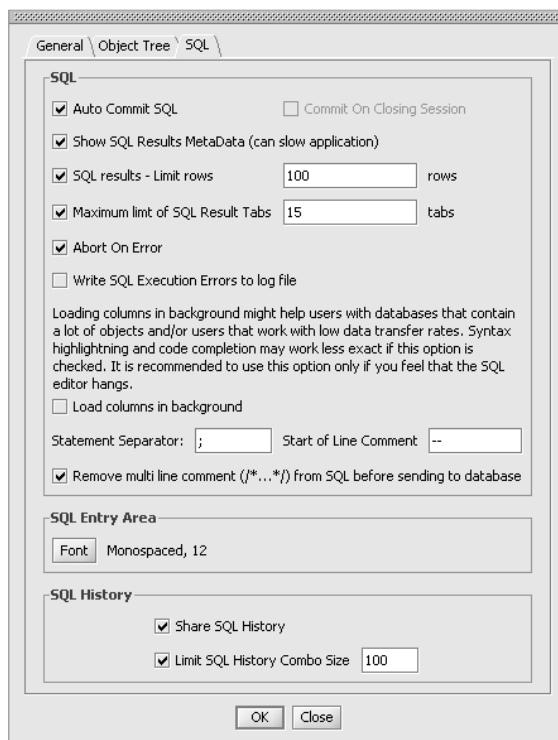
1. Object Tree Tab (Object Tree)

- ▮ **Load Schemas/Catalogs into object tree** - If checked then the global DB information is automatically loaded into the Object Tree view of the Database. This is useful for navigating the entire Database contents, but it may cause an unacceptable performance penalty, e.g. when using e-DBI for remote access through a low-bandwidth connection.
- ▮ **Show Row Count for Tables (can slow application)** - If checked then when a table is opened the number of rows in the table is included in parentheses after the name of the table in the message bar at the bottom of the e-DBI windows. If not checked, then the number is not shown. Showing the number may slow the application. The number is still available by using the "Row Count" tab in the Object view.
- ▮ **Contents - Limit rows [number] rows** - If checked then the number of rows read from the table is limited to the number in the text box, which may be adjusted as needed. If not checked, then e-DBI will attempt to read the entire table.

2. Object Tree Tab (Filters)

- | **Limit Schema Objects using these comma-delimited prefixes:** - Limit the list of objects shown in the Object Tree to only those with Schema names that start with the entries in this box. If the box is empty, then no limiting is done and all objects are shown.
- | **Limit Catalog Objects using these comma-delimited prefixes:** - Limit the list of objects shown in the Object Tree to only those with Catalog names that start with the entries in this box. If the box is empty, then no limiting is done and all objects are shown.
- | **Object Filter** - Limit the list of objects shown in the Object Tree to only those with names that match the pattern specified in this box.

C. SQL Tab



1. SQL Tab (SQL)

- | **Auto Commit** - If checked then all SQL will be automatically committed after it is executed. If unchecked then you will be able to use the "Commit" and "Rollback" options on the "Session" menu.
- | **Show SQL Results MetaData (can slow application)** - If checked then the MetaData associated with the results of an SQL query will be read and made available to the user. This may slow down e-DBI operation, especially when being used over low-speed connections. If unchecked, then the MetaData will not be available for the SQL query results.
- | **SQL Results - Limit Rows [number] rows** - If *SQL - Limit rows* is checked then only the number of rows specified in the *rows* will be displayed for an SQL query.
- | **Abort on Error** - If checked then when executing multiple commands in a single sequence, if one of the commands returns an error, then the sequence is aborted at that point. If not checked, then the SQL commands following the one with the error are executed anyway.
- | **Statement Separator** - The character used to separate SQL statements in the SQL entry area.
- | **Start of Line Comment** - The character that specifies that the line in the SQL entry area is a comment and should not be passed to the database for execution.

2. SQL Tab (SQL Entry Area)

- | **Font** - The *Font* button allows you to specify the font to be used in the SQL entry area.

3. SQL Tab (SQL History)

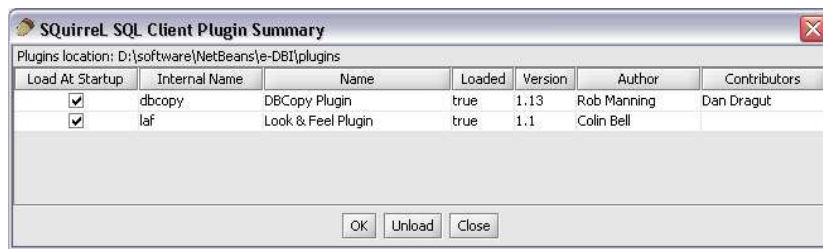
- | **Share SQL History** - if checked then the SQL history is shared across all Sessions. If not, then the history is kept for this Session only.
- | **Limit SQL History Combo Size [number]** - if checked then limit the number of lines of SQL available in the SQL History Combo box (pull down list above the SQL command entry area) to the (adjustable) number given.

VII. Plugins

A plugin is an application written in Java that runs within e-DBI. This allows developers to enhance the functionality of e-DBI without having to rebuild the e-DBI application itself. It also allows you to "pick and choose" the functionality that you want in e-DBI. For instance if you don't use Oracle then you don't need the Oracle specific functionality supplied by the Oracle plugin.

All supported plugins are available in the install jar and can be installed at the same time when installing e-DBI, or at any time afterward. Third-party plugins can be installed by unpacking the archive file (*.zip, *.gz) into the plugins directory within the e-DBI application directory keeping the directory structure. To use the plugin you will need to restart e-DBI. The e-DBI project team no longer makes individual plugins available for download.

The "Summary" option on the "Plugins" menu will display a dialog box showing the plugins currently installed.



VIII. Logs

IX. Menus

A. File Menu

- **Global Preferences:** Displays the [Global Preferences](#) dialog.
- **New Session Properties:** Displays the [New Session Properties](#) dialog.
- **Dump Application:** This option will dump some internal structures of the application to a text file along with a [dump](#) of all the currently open sessions. This may be useful for debugging problems with e-DBI.
- **Exit:** Exit the application after closing all sessions.

B. Drivers Menu

- **New Driver:** Displays a dialog allowing you to specify a new JDBC [driver](#).
- **Modify Driver:** Displays a dialog allowing you to modify an existing JDBC [driver](#).
- **Delete Driver:** Allows you to delete an existing JDBC [driver](#).
- **Copy Driver:** Copies the JDBC [driver](#) currently selected in the Drivers list and displays it as a new driver in the driver maintenance dialog
- **Install Default Driver Definitions:** Copies entries from the default JDBC driver definitions that ships with e-DBI to the Drivers List. If the driver definition is already in the list then the default one will not be copied.

C. Aliases Menu

- **Connect:** Display a connection dialog allowing you to [connect](#) to the specified [alias](#).
- **New Alias:** Displays a dialog allowing you to specify a new [alias](#).
- **Modify Alias:** Displays a dialog allowing you to modify an existing [alias](#).
- **Delete Alias:** Allows you to delete an existing [alias](#).
- **Copy Alias:** Copies the [alias](#) currently selected in the ALiases list and displays it as a new alias in the alias maintenance dialog

D. Plugins Menu

- **Summary:** Displays a dialog showing the installed [plugins](#).

E. Session Menu

- **Session Properties:** This option will display the Session Properties dialog allowing you configure your current session. [More](#).
- **Dump Session:** This option will dump some internal structures of the current session to a text file. This may be useful for debugging problems with e-DBI.
- **Refresh Tree:** This option will refresh the object tree.
- **Run SQL:** This option will run the current SQL in the SQL entry area. [More](#).
- **Commit:** This option will commit the current SQL transaction. This is only applicable if "Auto Commit" has been turned off in the [Session Properties](#).
- **Rollback:** This option will rollback the current SQL transaction. This is only applicable if "Auto Commit" has been turned off in the [Session Properties](#).
- **Goto Previous Results Tab:** This option will select the Results Set tab to the left of the currently selected one. If the leftmost tab is currently selected then the rightmost tab will be selected.
- **Goto Next Results Tab:** This option will select the Results Set tab to the right of the currently selected one. If the rightmost tab is currently selected then the leftmost tab will be selected.
- **Show Native SQL:**

This option will append the native SQL for the current JDBC SQL into the SQL entry area. E.G. If you enter the following in an Oracle session

```
select * from table1 where entered_date = {d'2002-12-01'}
```

and take this option then the following will be appended to the SQL entry area.

```
select * from table1 where entered_date = TO_DATE ('2002-12-01', 'YYYY-MM-DD')
```

- **Reconnect:** Close the current connection to the database and reopen it using the same user name and password.
- **Close Session:** Close the current connection to the database and close the session window.
- **Close All SQL Result Tabs:** Close all SQL results displayed in the tabbed folder for the current session.
- **Close All SQL Result Windows:** Close all SQL results windows "torn off" from the tabbed folder for the current session
- **Limit cell edit WHERE clause size:** This allows you to select specific columns to use as the key field when updating the DB after you edit a cell in a table. The purpose of thi is to reduce the size of the WHERE clause, which normally includes all field in the table. For tables with many columns, or with columns containing lon data entries (not including BLOB/CLOB fields, which are handled differently) the automatically generated WHERE clause may exceed the DBMS length limitation You may move column names between the "use columns" and "not use columns" boxe by selecting the names and using the arrow buttons to move them to the other box You must leave at least one column in the "use columns" box, and the column in the "use columns" box must be sufficient to uniquely identify each row in the table, e.g. the primary key fields for the table The "Reset" button restores the column name lists to the way they were when you opened the window Remember to click "OK" when you are done
- If you define a set of columns to use in the WHERE clause and then edit some other column, e-DBI will warn you that "1 duplicate row" is about to be created. Since you have told e-DBI to not look at the edited column it does no recognize that the new data is different than the old data so it gives you a warning. In this case there is no problem and you should just tell e-DBI to go ahead with the update.

F. Windows Menu

- **View Aliases:** Display the list of [aliases](#) that define a connection to a database.
- **View Drivers:** Display the list of [drivers](#).
- **View e-DBI Logs:** Display the execution [logs](#) for e-DBI.
- **Tile:** Tile the open session windows.
- **Cascade:** Cascade the open session windows.
- **Maximize:** Maximize the open session windows.
- **Close All Sessions:** Close all existing sessions.

G. Help Menu

- **Help:** Displays this Help File.
- **FAQ:** Displays the Frequently Asked Questions file.
- **Change Log:** Displays the development history of e-DBI.
- **Licence:** Displays the licence for e-DBI.
- **About:** Displays the About Box for e-DBI.

Contact Person: [Ammar Benabdelkader](#)

