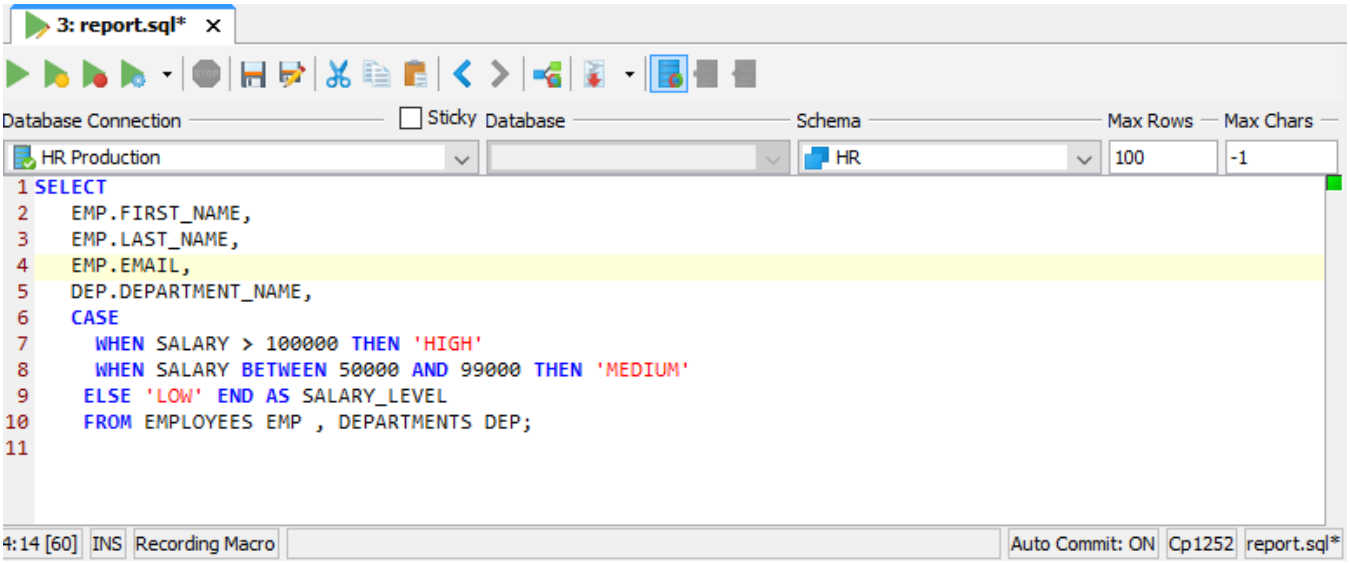
Editing SQL Scripts

The SQL Commander contains an SQL editor, used to edit SQL scripts.

The editor area looks like this:



Above the editor is a toolbar with buttons related both to execution of scripts and to editing. The editing related buttons are covered below.

The left margin shows the line numbers.

Below the editor, you see a Status Bar. The first field shows the current caret position in the format:

<line>:<column> [<position from top>]

The last figure, within square brackets, is the caret position from the top. This can be useful when you get an error message executing a script that contains this information rather than a line/column location.

The next field in the Status Bar shows INS if characters you type will be inserted at the caret position or OVR if they will overwrite the current text at the caret position. You can toggle this mode using the **Toggle Typing Mode** keyboard shortcut, by default bound to the **Insert** key.

The next field shown in the screenshot is only visible when working with macros, described in the Recording and Playing Edit Macros section.

Next comes the Auto Commit Status field, showing whether Auto Commit is enabled.

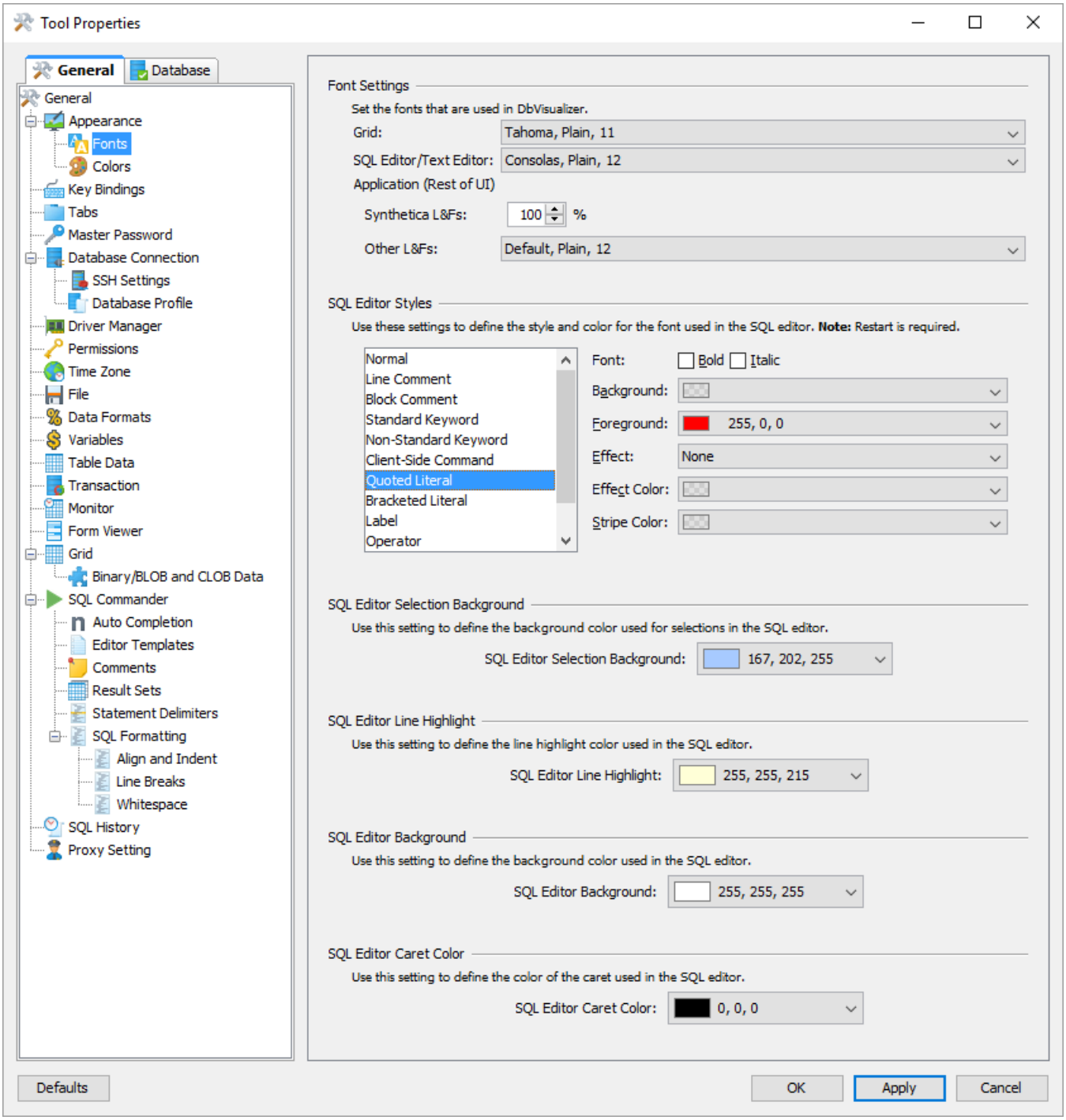
The last two fields show information about the file loaded in the editor, if any. First the character encoding and then the filename. If you just type into the editor without loading a file, the filename "Untitled" is shown instead. An asterisk after the filename indicates that there are unsaved edits.

The SQL Editor is like any editor you're used to when it comes to typing, scrolling etc. But it also offers additional features to help you specifically with editing SQL scripts. These are described in the following sections.

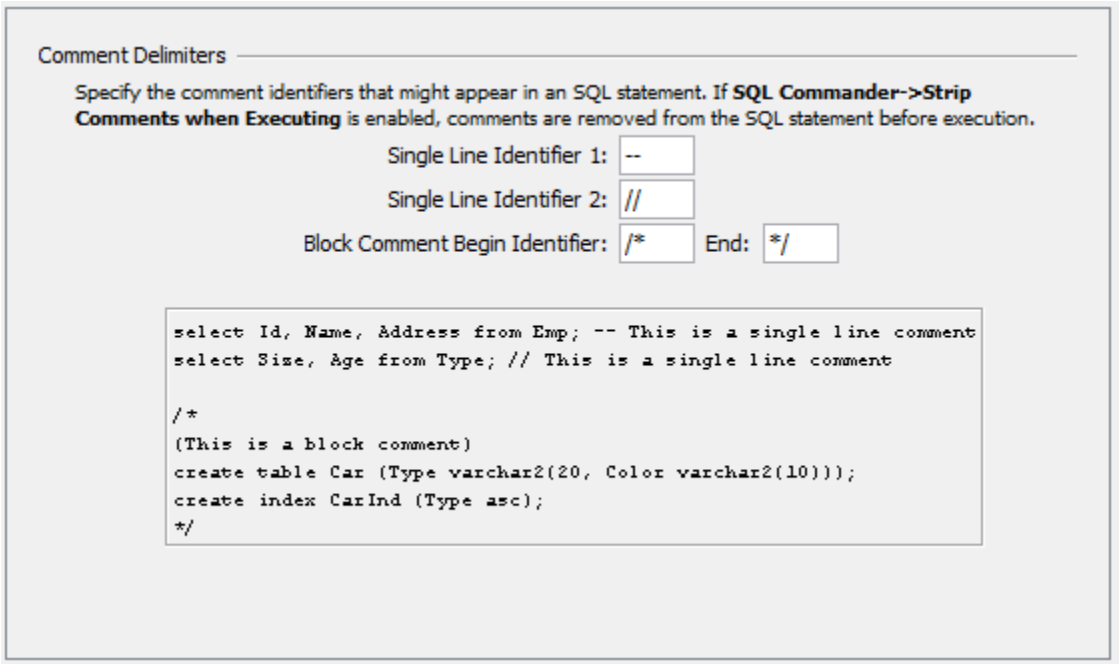
Syntax Color Coding

An SQL script consists of keywords, operators, object identifiers, quoted text, etc. It may also contain comments. To make it easier to see at a glance what is what, the SQL Editor displays words using different font styles depending on their classification. For instance, keywords are displayed with a bold blue font, while quoted text is displayed with a regular type red font.

You can change how to display the different kinds of words, as well as the editor selection background color, the current line highlight color and the editor background color, in the **Tools->Tool Properties** dialog, in the **Appearance/Fonts** category.

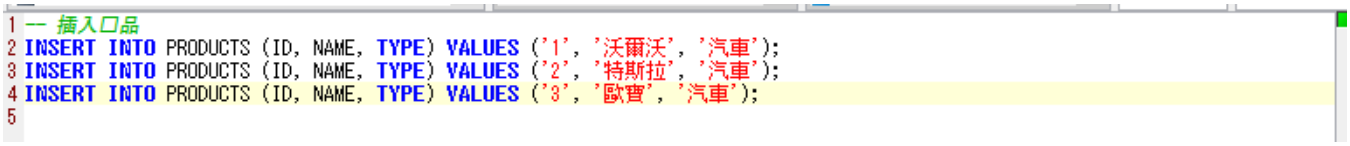


The editor uses the Tool Properties settings from the **SQL Commander/Comments** category under the General tab to detect comments.



Charsets and Fonts

You can also change the SQL Editor font family, which is useful and necessary in order to display characters for languages like Chinese, Japanese, etc., in **Tool Properties** in the **Appearance/Fonts** category to set the font for the SQL Editor.



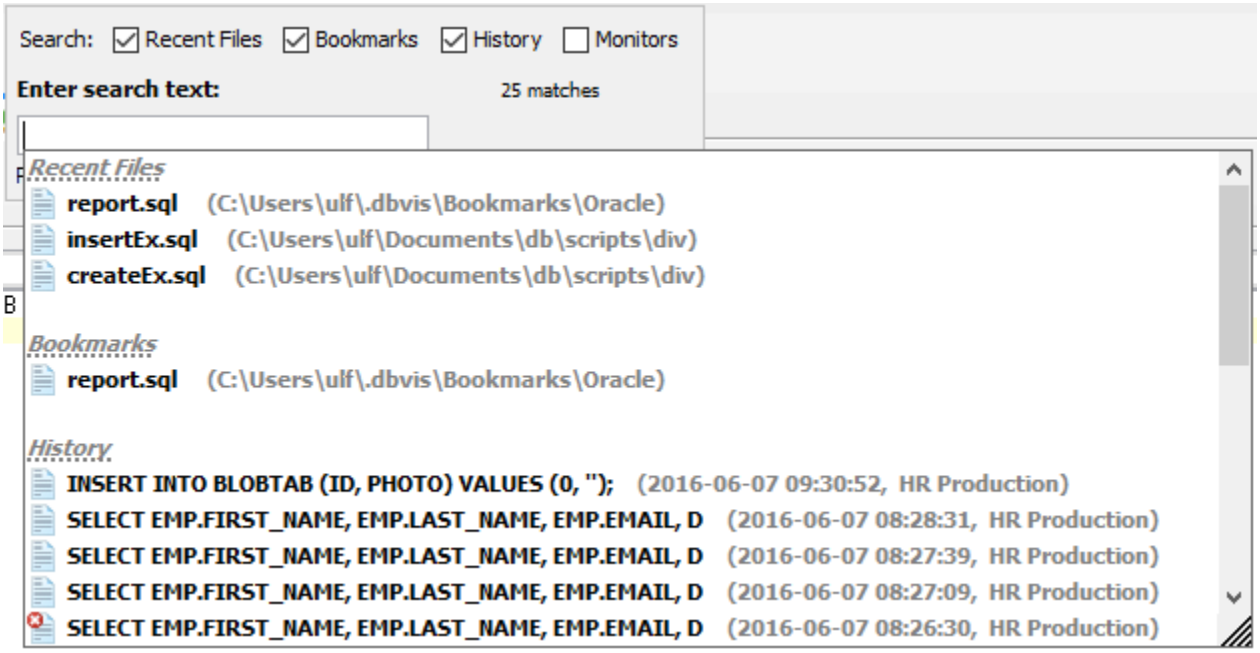
Loading and Saving Scripts

The SQL editor supports loading statements from a file and saving the content of the editor to a file. Use the standard file operations, **Open**, **Save** and **Save As** in the **File** main menu or the main toolbar to accomplish this. Loading a file loads it into a new **SQL Commander** tab or activates the tab that already holds it.

The name of the loaded file is listed in the status bar of the editor, with the full file path shown in the window title. The editor tracks any modifications and indicates changes with an asterisk (*) after the filename. When you close the SQL Commander tab or exit DbVisualizer, you are asked what to do if there are any pending edits that need to be saved.

The **File->Open Recent** submenu lists the recently loaded files. How many recent files to keep track of can be specified in the Tool Properties dialog, in the **SQL Commander** category under then General tab.

You can also use the **Quick File Open** feature to open recent files as well as Bookmarks and History entries. By default, it is bound to the **Ctrl+Alt+O** key combination, and is also available via a main toolbar button as well as in the main **File->Quick File Open** menu.

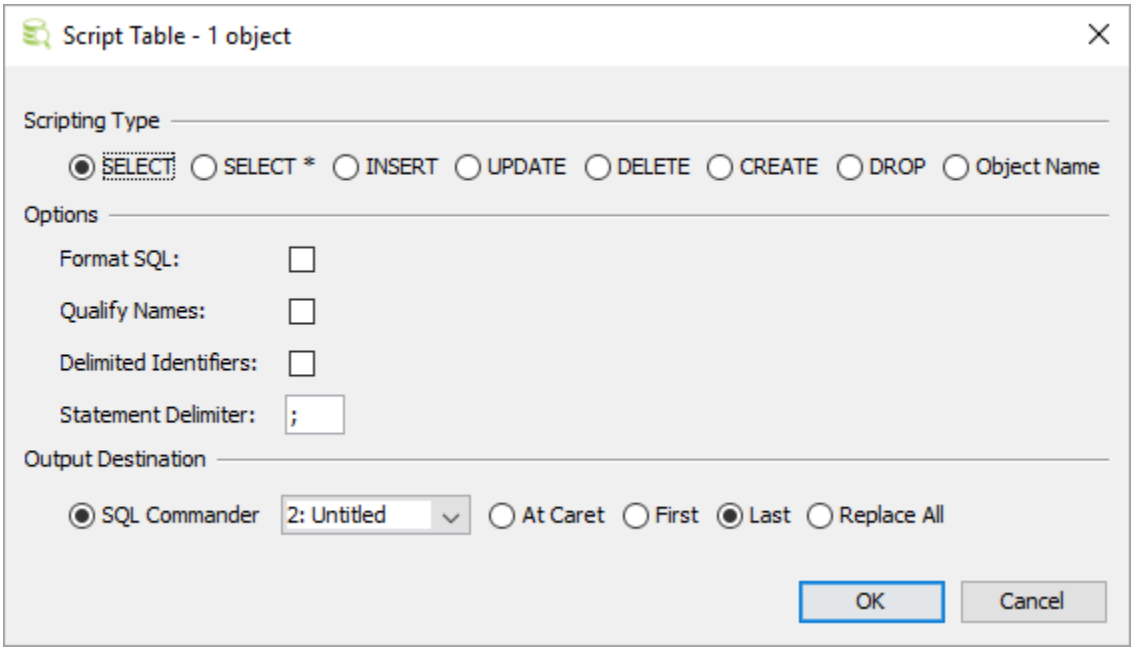


Drag and Drop a File

You can also select a file in the platform's file browser and drop it somewhere in the DbVisualizer window. If you drop it in an editor, the file content is inserted at the caret position in the editor. If you instead drop it in the toolbar area, the file is opened in a new **SQL Commander** tab.

Drag and Drop Database Objects

If you want to include an object shown in the database objects tree, you can select the node and drop it in the editor where you want it inserted. The **Script Object** dialog is shown where you can select exactly what you want to insert in the editor.



First of all, you can select to insert an SQL statement based on the dropped object, e.g. a SELECT statement or a CREATE statement. You can also choose to just insert the object name. The choices available depends on the type of object you drop.

In the **Options** area, you can opt to format the SQL before it is inserted and use qualifiers and quoted identifiers, and even change which statement delimiter to use.

The **Output Destination** is set to the SQL Commander tab you dropped the object on by default, but you can change your mind and pick another destination. If you stick with an **SQL Commander** as the destination, you can tell where in the editor to insert the text.

You can also open this dialog from the **Databases** tab, from the object's right-click menu.

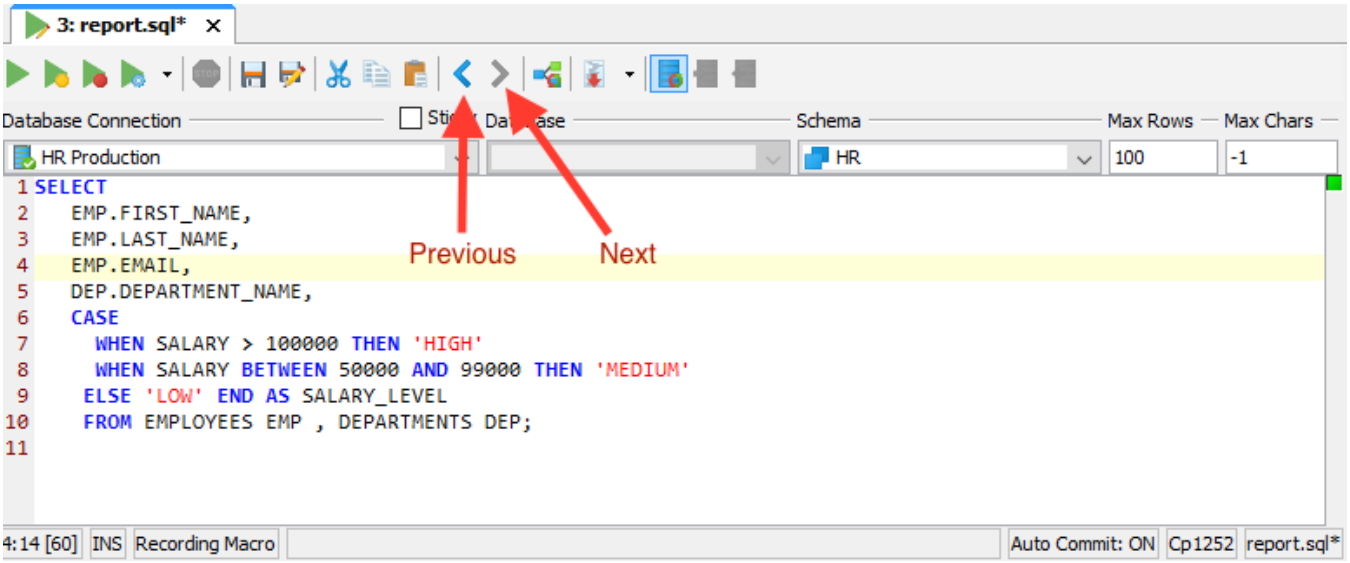
If you just want to insert the object names in the editor, hold down the **Ctrl** key (or the **Alt** key on macOS) while dragging and dropping. This behavior can be reversed in **Tool Properties**, in the **SQL Commander** category, so that dropping without pressing a key inserts the names and pressing the key launches the dialog.

Loading and Saving Bookmarks and Monitors

Bookmarks and Monitors are also files, but with special meaning. See the Managing Frequently Used SQL for how to create and edit them in the SQL Editor.

Navigating Between History Entries

When you execute a script, DbVisualizer saves it as a history entry, see the Re-Executing SQL Statements section for details. You can use the **Previous** a nd **Next** buttons in the editor toolbar to navigate between (load) these entries.



Confirming Overwriting Unsaved Changes

By default, you have to confirm overwriting unsaved changes in an editor, e.g. when navigating between history entries, and when closing an SQL Commander tab with unsaved edits. You can disable these confirmation popups in the Tool Properties dialog, under the **SQL Commander** category under the General tab.

SQL Formatting

Only in DbVisualizer Pro

This feature is only available in the DbVisualizer Pro edition.

The **SQL->Format SQL** menu contains four operations for formatting SQL statements.

Format Buffer formats the complete editor content and **Format Current** formats the current SQL (at cursor position).

The formatting is done according to the settings defined in the Tool Properties dialog, in the **SQL Editor/SQL Formatting** category under the General tab. There are many things you can configure. After making some changes, press **Apply** and format again to see the result.

Here is an example of an SQL statement before formatting:

```
select
CompanyName, ContactName, Address,
City, Country, PostalCode from
Northwind.dbo.Customers OuterC
where CustomerID in (select top 2 InnerC.CustomerId
from Northwind.dbo.[Order Details] OD
join Northwind.dbo.Orders O on OD.OrderId = O.OrderID
join Northwind.dbo.Customers InnerC
on O.CustomerID = InnerC.CustomerId
Where Region = OuterC.Region
group by Region, InnerC.CustomerId
order by sum(UnitPrice * Quantity * (1-Discount)) desc)
order by Region
```

And here is the same statement after formatting has been applied with default settings:

```
SELECT
    CompanyName,
    ContactName,
    Address,
    City,
    Country,
    PostalCode
FROM
    Northwind.dbo.Customers OuterC
WHERE
    CustomerID in
    (
    SELECT
        top 2 InnerC.CustomerId
    FROM
        Northwind.dbo.[ORDER Details] OD
    JOIN
        Northwind.dbo.Orders O
        ON
        OD.OrderId = O.OrderID
    JOIN
        Northwind.dbo.Customers InnerC
        ON
        O.CustomerID = InnerC.CustomerId
    WHERE
        Region = OuterC.Region
    GROUP BY
        Region,
        InnerC.CustomerId
    ORDER BY
        SUM(UnitPrice * Quantity * (1-Discount)) DESC)
ORDER BY
    Region
```

Copy Formatted and **Paste Formatted** are powerful tools for copying SQL statements between programs written in languages like Java, C#, PHP, VB, etc. and the SQL Editor. Both operations display a dialog where you can adjust some of the formatting options, most importantly the **Target SQL** option and the **SQL is Between** option. **Target SQL** can be set to a number of common programming language formats.

For instance, to copy an SQL statement and paste it as Java code for adding it to a Java StringBuffer:

1. Select the statement,
2. Choose **SQL->Format SQL->Copy Formatted**,
3. Set **Target SQL** to Java StringBuffer,
4. Click **Format** to place the formatted statement on the system clipboard,
5. Paste it into your Java code.

To copy a statement wrapped in code from a program:

1. Select the code containing an SQL statement in your program,
2. Copy it to the system clipboard,
3. Choose **SQL->Format SQL->Paste Formatted**,
4. Check **SQL is Between** and enter the character enclosing the SQL statement in the code,
5. Click **Format** to extract the SQL statement and paste the formatted SQL in the editor.

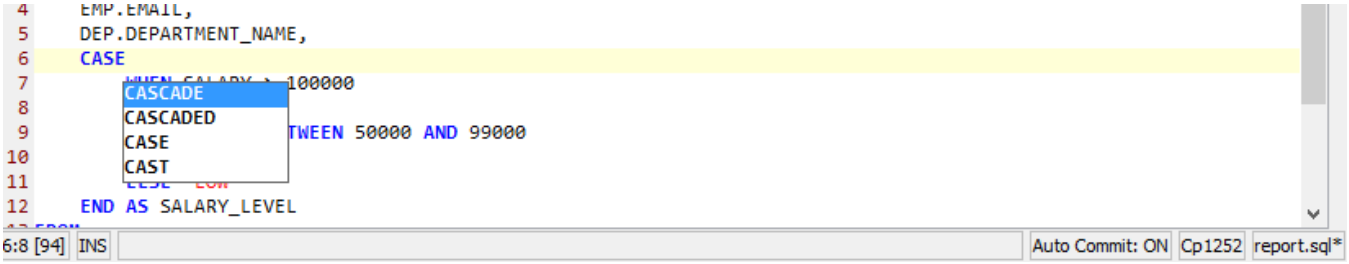
Auto Completion

Only in DbVisualizer Pro

This feature is only available in the DbVisualizer Pro edition.

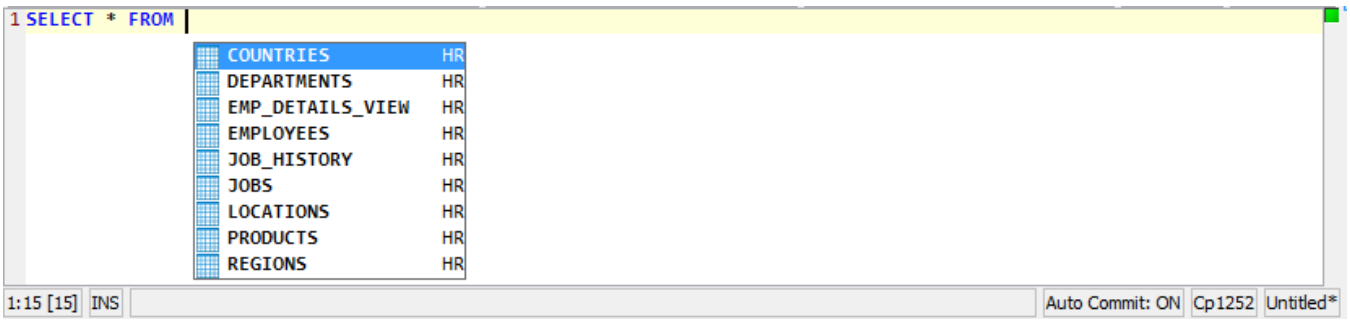
Auto completion is a convenient feature used to assist you when editing SQL statements and DbVisualizer commands. By default, you activate auto completion with the key binding **Ctrl-SPACE**, but you can also configure it to activate as you type (in the Tool Properties dialog, in the **SQL Editor/Auto Completion** category under the General tab).

With the caret in any place in a statement where you can type something other than a table name or a column name, and at least one character just before the caret, activating auto completion displays a list of keywords that starts with the letters you have typed so far. As you continue to type, the list narrows.

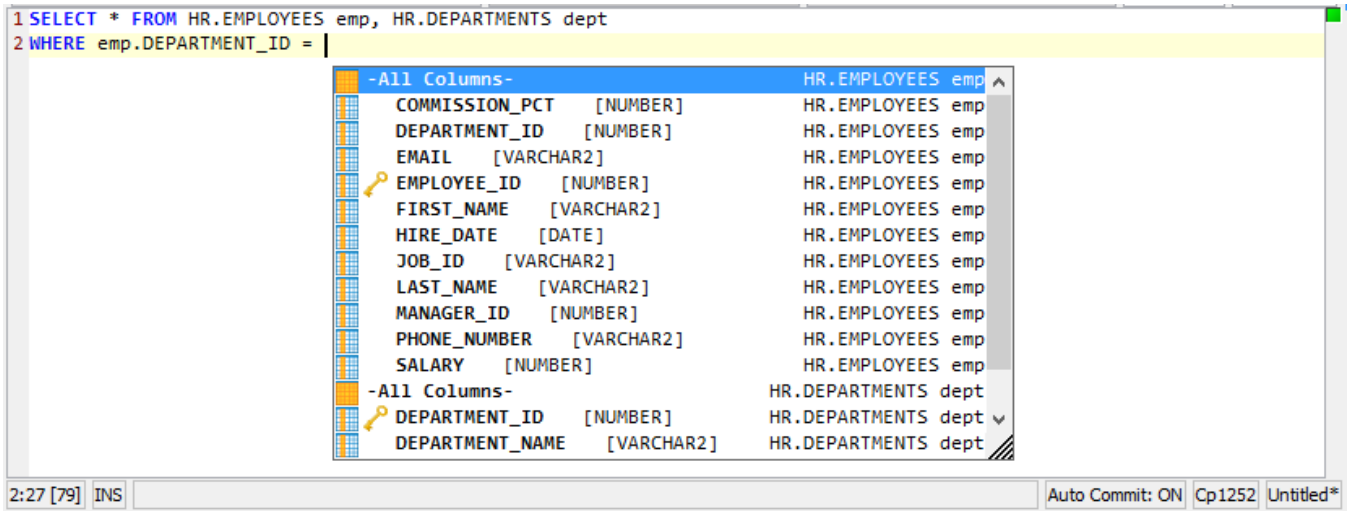


The list of keywords is database specific, selected based on the database type for the connection currently selected in the **Database Connection** list above the editor.

With the caret placed where a table or view name may be typed in a supported SQL statement type, the auto completion list shows a list of tables and views from the currently selected database connection, assuming you are actually connected to the database. The following figure shows the completion pop up with table names.



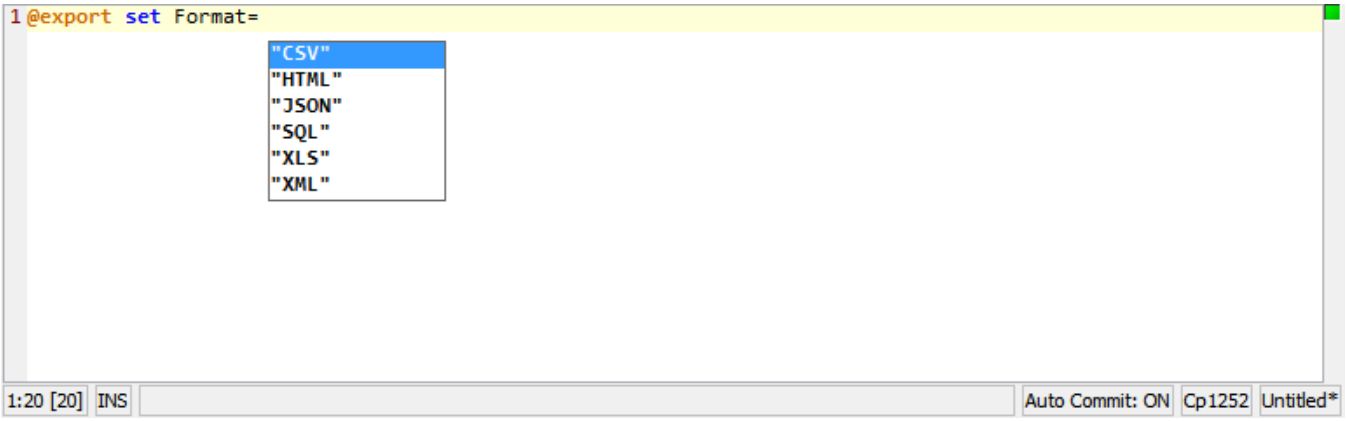
A completion pop-up showing column names is shown when the caret is placed where a column name may be typed.



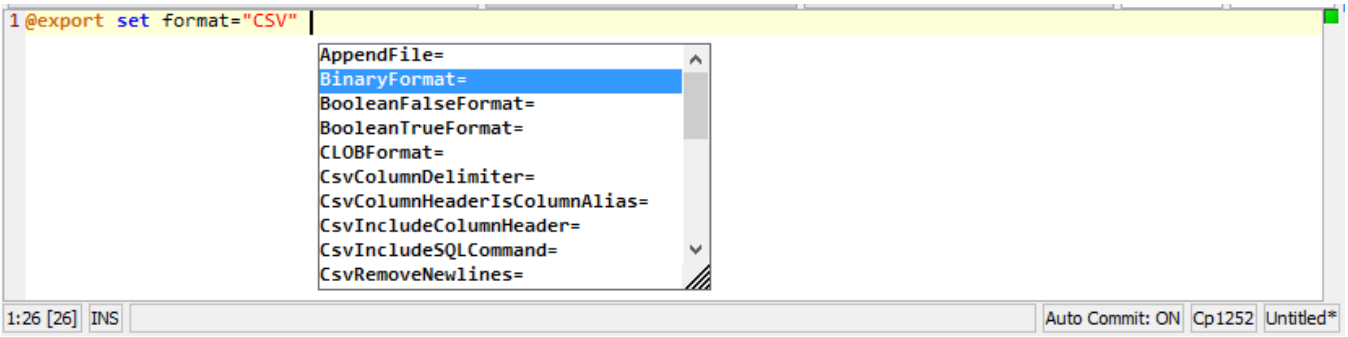
DbVisualizer provides auto completion for table and columns names for the following DML commands:

- SELECT
- INSERT
- UPDATE
- DELETE

Auto completion for DbVisualizer commands is very similar. Activating it after a partial command name lists all matching commands. If you activate it after a complete command name, you get a list of all valid parameters for the command. After a parameter name, you can select from a list of valid values.



For the `@export set` command, the parameter list is adapted for the specified output format after you have entered the `Format` parameter setting, for instance only showing parameters that are valid for the CSV format.



To display the completion pop-up, use the key binding **Ctrl-SPACE** (by default). You select an entry in the pop-up menu with a mouse double-click, the **ENTER** key, or the **TAB** key. To cancel the pop-up, press the **ESC** key.

If there are several SQL statements in the editor, make sure to separate them using the statement delimiter character (the default is ";").

In order for the column name completion pop-up to appear, you must first make sure there are table names in the statement.

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

The **Database** and **Schema** lists above the editor are used to limit the list of tables in the auto complete pop-up to those in the selected database and/or schema. To include all tables, select the blank entries in these lists. The default selections for the lists can be set as connection properties, in the **SQL Commander** category.

It is possible to fine-tune how auto completion works in the connection properties.

- Enable or disable the use of identifier qualifiers (i.e. qualifying table names with the schema name) in the **[Database Type]/Qualifiers** category,
- Enable or disable the use of delimited identifiers (e.g. quotes around a table name) in the **[Database Type]/Delimited Identifiers** category.

Sorting, when to show the popup, etc. can be configured in the Tool Properties dialog, in the **SQL Editor/Auto Completion** category under the General tab.

Recording and Playing Edit Macros

If you repeatedly need to run a sequence of edit operation, you can record them as a macro and play it as many times as needed during an editing session. The editor status bar indicates when a recording is in progress and when a macro is available to play.

As an example, suppose you have some plain text that you need to convert into INSERT statements:

```
12345 123456
89012 890123
45678 456789
```

Place the caret at the beginning of the first line and start the macro recording, using the right-click menu or the corresponding key binding, and then type text and use key bindings (or menu items) to perform the following operations:

1. Type `insert into mytable values('`
2. **Insertion Point to End of Word**
3. Type `',`
4. **Insertion Point to Next Word**
5. Type `'`
6. **Insertion Point to End of Word**
7. Type `');`
8. **Insertion Point Down**
9. **Insertion Point to Beginning of Line**

Then stop the recording. You now have a macro for converting a single line to an INSERT statement. To convert the remaining lines, just use Play Macro for each line. The result will look like this:

```
insert into mytable values('12345', '123456');
insert into mytable values('89012', '890123');
insert into mytable values('45678', '456789');
```

The Find operation, by default mapped to the **Find** key and **Ctrl-F** key stroke, can not be recorded. You must instead use **Find Selection**, **Find with Dialog**, **Find Next** and **Find Previous**. Mouse gestures are also not recorded, only key strokes and menu selections.

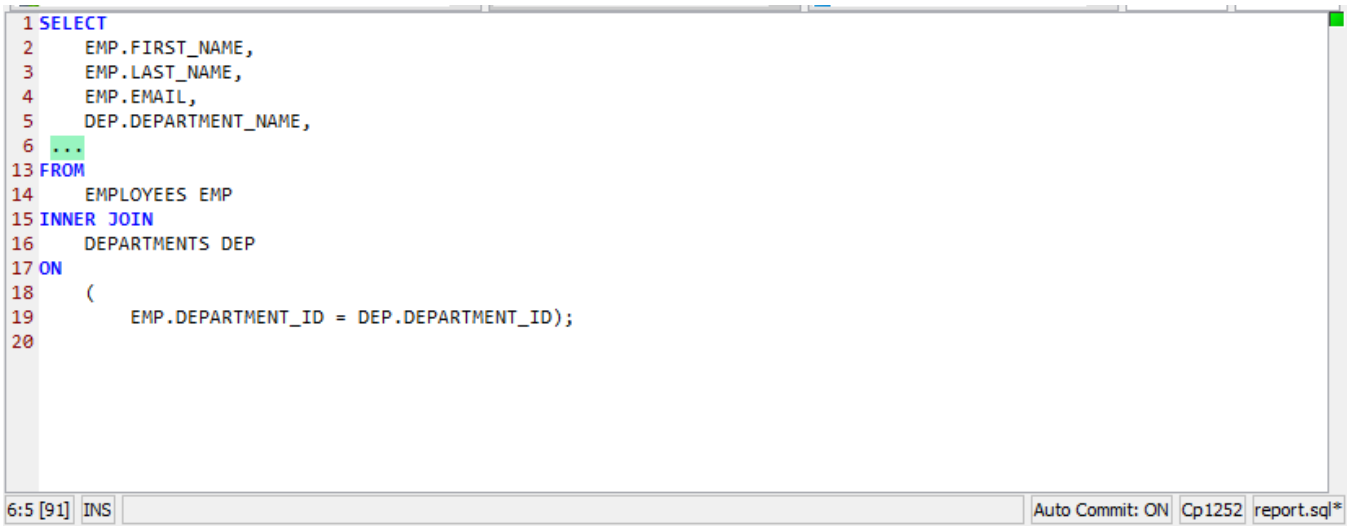
Folding Selected Text

If you work with a large script, it can sometimes be helpful to hide parts of it. You can do so using the Code Folding feature.

Select the text you want to hide and then choose **Toggle Fold Selection** in the right-click menu. The selected text is then replaced (visually only) with a folding marker.

Here's an unfolded script:

And here is the same script with the CASE expression folded:



You can fold more than one part of a script using the same procedure.

To unfold just one part, select the folding marker (be careful to select all of it) and then choose **Toggle Fold Selection** from the menu again. To unfold all folded parts, use **Expand All Foldings**.

Selecting a Rectangular Area

In some cases, it is handy to be able to select a rectangular area in the middle of a script. Say, for instance, that you need to copy just the first part of a few lines and paste it at the beginning of some other lines.

To do this in the SQL editor, click the mouse where you want to start the selection and then press the **Alt** key (by default) while you extend the selection by dragging the mouse. If you prefer to use the **Ctrl** key as the modifier, you can change the default in Tool Properties in the SQL Commander category under the General tab.

Tab Key Treatment

Pressing the TAB key in the editor inserts four space characters by default. If you instead want a TAB character to be inserted, or want to insert another number of space characters, you can specify this in the Tool Properties dialog, in the **SQL Commander** category under the General tab.

Key Bindings

The editor shortcuts, or key bindings, can be redefined in the **Tool Properties** dialog, in the **Key Bindings** category under the General tab. Expand the **Editor Commands** node to manage all editor actions and the **Main Menu/Edit** node to manage the key bindings for the edit operations in the right-click editor menu and the main window **Edit** menu.