

**TOra**

---

---

This manual is for TOra, version 2.1.2+.

# Table of Contents

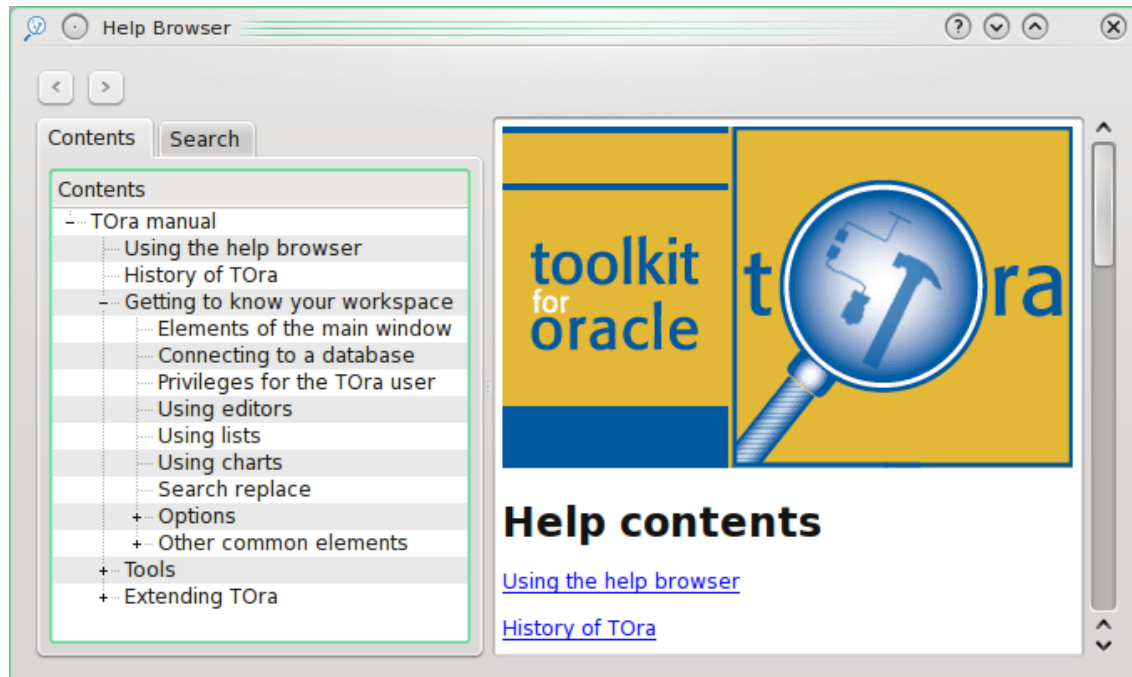
<b>1</b>	<b>Using the help browser</b>	<b>1</b>
<b>2</b>	<b>History of TOra</b>	<b>2</b>
<b>3</b>	<b>Getting to know your workspace</b>	<b>4</b>
3.1	Elements of the main window	4
3.2	Connecting to a database	12
3.3	What privileges do you need to run TOra	13
3.4	Using editors	14
3.5	Using lists	15
3.6	Using charts	17
3.7	Search & Replace	18
3.8	Preferences	20
3.8.1	Global Settings	21
3.8.2	Editor Settings	22
3.8.3	Database Settings	26
3.8.4	Additional Help Settings	29
3.8.5	Tool Settings	30
3.9	Other common elements	31
3.9.1	SGA Statement	31
3.9.2	Explain plan	31
3.9.3	Getting bind parameters	31
3.9.4	Object description	32
3.9.5	Memo editor	32
<b>4</b>	<b>Available tools</b>	<b>34</b>
4.1	SQL Editor	34
4.2	Schema Browser	39
4.3	PL/SQL Editor	46
4.4	PL/SQL Debugger	48
4.5	PL/SQL Unit Tester	53
4.6	Server Tuning	54
4.7	Security Manager	56
4.8	Storage Manager	59
4.9	Session Manager	61
4.10	Rollback Segments	63
4.11	SGA Trace	65
4.12	Current Session	66
4.13	Schema extraction, compare and search	67
4.14	Statistics manager	71
4.15	Alert Tool	72
4.16	Invalid Objects	73

4.17	SQL Output .....	74
4.18	Template Help .....	74
4.19	SQL Editor .....	78
<b>5</b>	<b>Extending TOra .....</b>	<b>81</b>
5.1	TOra Tool Tutorial .....	81
5.2	API Reference .....	85
	<b>Index-list .....</b>	<b>86</b>

# 1 Using the help browser

You can always bring up help on the current dialog etc. by pressing the **F1** key. You can also select it from the application menu bar under Help (dooh).

When you open the help browser which you obviously have done. You are presented with a screen similar to the following.



On the top of the screen is a toolbar with back and forward buttons that you can use to go back and forward in the history of viewed pages that you have previously looked at. This list kept during the duration of running TORA, however when you display help in dialogs that help is not added to the main application help history.

To the right is a large area which displays the documentation. This is essentially a web browser (If you run the KDE version it is actually an embedded konqueror) and works as such. Clicking on blue text will go to the referenced section.

To the left is a pane that can either display the contents of the manuals available for browsing or search result. Note that the TORA help is compatible with Oracle manuals which can be browsed in the help viewer if you add them in the options dialog (see [Section 3.8.4 \[additionalhelp\]](#), page 29).

You can also display a different pane where you can search the manuals. In this version of TORA you can only search external Oracle manuals and not the TORA user manual. This will be fixed in a future version of TORA.

Just enter the words you want to find and the sections containing **all** the words entered will be displayed in the list below it. Selecting a row in either the search result or contents list will bring up that page in the right part of the screen.

## 2 History of TOra

### History of TOra

I started the TOra project in December 2000, the reason being that I had been increasingly jealous of the tools available for interacting with Oracle databases to my colleges using windows. The tools that were available for Linux users were not nearly powerful enough. Execution plans being pretty much the most advanced feature available. During this time I also made a lot of attempts to get VMWare to work without any success fortunately. If I could have made this to work I would never have started TOra but run a VMWare client with Windows.

I started by working on what is now the worksheet in TOra. Already in the beginning I envisaged TOra being plugin based and easily extendable with new functionality without any or much change of the existing code base, the `toTool class` being one of the first written gives this functionality.

In the beginning the development pace was really fast as always and it didn't take long before it had at least in my opinion surpassed any available tool for Linux as an SQL worksheet. Most of the development of these versions were made in airports in Asia since I spent about a week in those on a backpacking holiday there around new years 2000 to 2001. Yes, I know there aren't many other people who would bring a laptop on a backpacking vacation, but I knew I would be spending a lot of time in airports and what better way to waste the time.

By the time I was back I released the first version of TOra with both worksheet and database browser. By this time I started looking at the PL/SQL debugging package available in Oracle, this was the first time I looked at developing functionality in TOra that I didn't really feel a pressing need for myself since I rarely write any large PL/SQL code. This was released in version 0.4 at the end of January, still only about 1 month after the project was started. This was the first functionality in TOra that is not available from any other open source project.

The next big thing to happen to this project was that TrollTech released the first version of Qt Free for windows at which point I started glancing at making TOra available for the windows platform. I've made a few feeble attempts earlier using an evaluation copy of Qt/Windows without much success. But when this release was made I really sat down and went through it finding lots of problems mainly with regards to Visual C++ (Which is really crap, but the one you have to use if you want to use Qt Free. Of course no tool can really replace emacs, gcc, make and gdb if you know how to use them).

After this development slowed down considerably to be able to stabilize the program to get the first stable release which was released on the first of september 2001. At almost the same time as 1.0 was released the first development release for 1.1 was released featuring a few of the most asked for features including improved tuning and better busy feedback. Most parts of TOra have been touched in some way for this release. The 1.2 release was also the first commercial release which is a way for me to ensure that TOra can be supported well forward in the future as it is starting to consume more and more of my time.

This page of the options about fonts and colors used in the user interface.

## The future

I will continue to develop TOra to be the best possible database development and administration tool for Oracle and in the long run probably other databases as well.

By the time of the 1.2 release development of the 1.4 release is already well under way and will mostly feature DBA improvements. The database browser which will start getting more and more functionality to modify the browsed objects. Other changes are made to make DBA:s administrating database which people are accessing using TOra should also be happy with this release since it will warn more often when you are doing things you probably shouldn't be doing (Like running a server tuning session just to get a handle of whats happening). Improved database tuning with a new wait event analysis tool is also already finished and ready for beta testing. Also PostgreSQL support will probably be available for the 1.4 release.

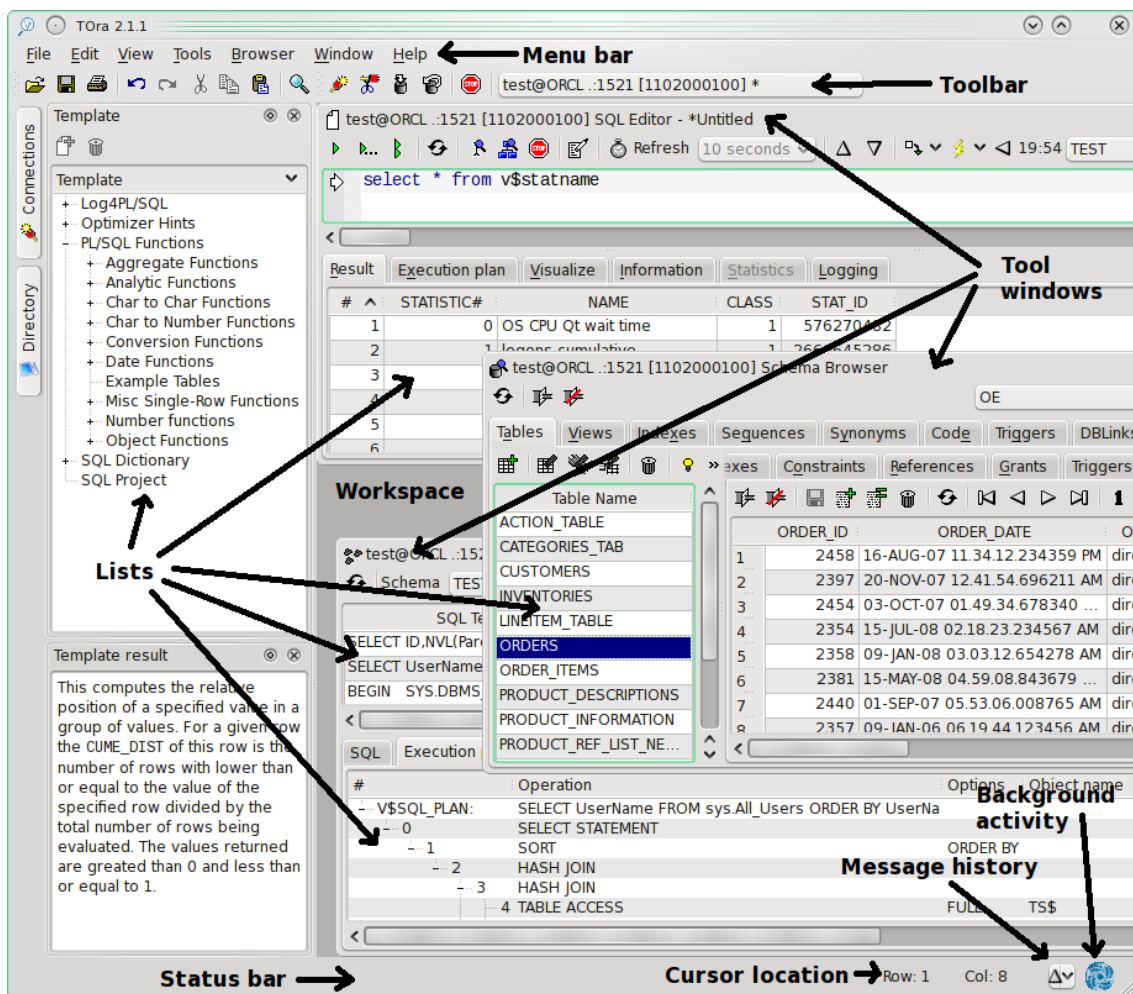
Henrik "Mauritz" Johnson, GlobeCom AB

## 3 Getting to know your workspace

This section will help you familiarize yourself with the framework of the TOra application which is common to all tools in the toolkit.

### 3.1 Elements of the main window

The main window in TOra has the following elements as shown below. It is assumed that you know how to handle a normal windows interface therefore no explanation about how to press buttons and select menus will be presented. If you are administrating Oracle you probably know how to handle a mouse.

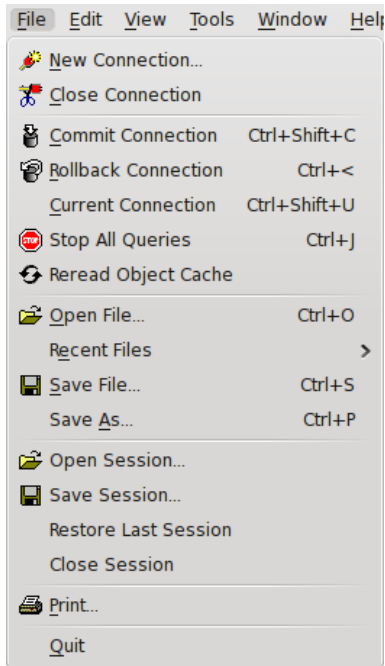


#### Menubar

If you've ever used a windows program you will recognize most of the menus hopefully.



## File menu



### New connection

This will pop up the new connection dialog (see [Section 3.2 \[newconnection\]](#), page 12) which is used to open new connections to the database.

### Close connection

Will close the current connection to Oracle. If uncommitted data exists it will also ask if you want to commit or rollback the changes. Observe that any tools associated with this connection will also be closed when closing a connection.

### Commit connection

Commit the changes currently made to the database.

### Rollback connection

Rollback any changes made in the current connection.

### Current connection

Set focus to change the current connection.

### Stop All Queries

Stops all queries running in current session.

### Reread object cache

Clear the object cache. and read it again from the database.

**Open File** This command will open a dialog asking you for a filename to load into the editor that currently holds the focus.

### Recent Files

This item will have subitems of recently opened files which you can load into the editor that currently holds the focus.

**Save File** This command will open a dialog asking you for a filename to save the contents of the editor that currently holds the focus. If the editor already has a filename this will be used and no dialog displayed.

**Save As** This command will open a dialog asking you for a filename to save the contents of the editor that currently holds the focus.

### Open Session

Open previously saved session.

### Save Session

Save parameters of current session.

### Restore Last Session

Open last saved session.

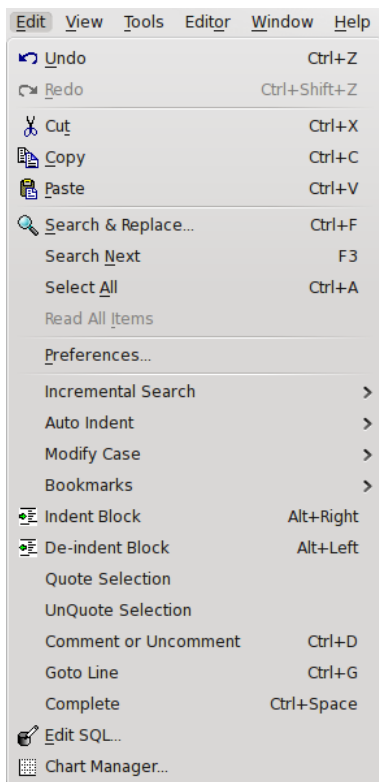
### Close Session

Close current session.

**Print** Will print the contents of the list or editor that currently holds the focus.

**Quit** Close all connections and exit TOra.

## Edit menu



**Undo** Undo the latest change in the editor that currently holds the focus.

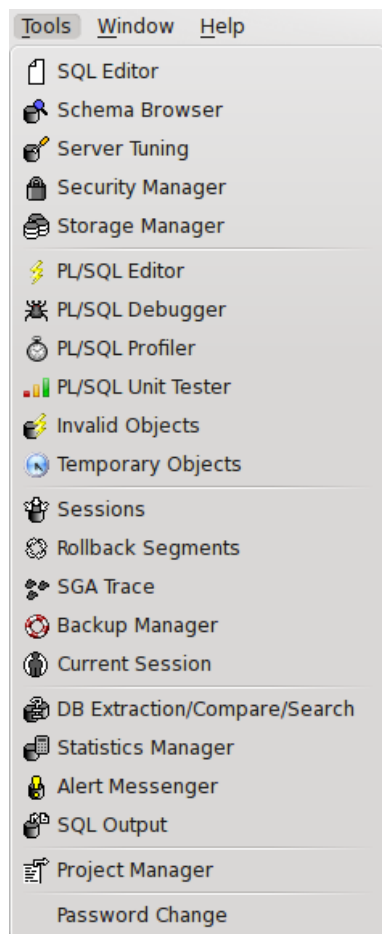
**Redo** Redo a previous undo of the latest change in the editor that currently holds the focus.

- Copy** Copy the current selection in the editor that currently holds the focus.
- Cut** Cut the current selection in the editor that currently holds the focus.
- Paste** Paste the contents of the clipboard into the editor that currently holds the focus.
- Search & Replace**  
Display the search & replace dialog (see [Section 3.7 \[searchreplace\]](#), page 18).
- Search Next**  
Make another search with the current search settings.
- Select All** Select all rows in the current editor.
- Read All Items**  
Read all items available for a list. Normally TOra doesn't read all available rows for a query, just enough to fill the screen. Scrolling down will read more. Selecting this will read all available rows from the query, this could take a very long time and consume a lot of memory if selected for a large query result.
- Preferences**  
Displays the preferences dialog (see [Section 3.8 \[preferences\]](#), page 20) where all aspects of TOra are configured.
- Incremental Search**  
.
- Auto Indent**  
.
- Modify Case**  
Change selected text to uppercase or lowercase.
- Bookmarks**  
Manage bookmarks.
- Indent Block**  
Indent selected block.
- De-indent Block**  
Deindent selected block.
- Quote Selection**  
Quote selection. This also handles quotes inside text being quoted.
- UnQuote Selection**  
Unquote selection.
- Comment or Uncomment**  
Comments/uncomments selected text.
- Goto Line** Goes to specific line number.
- Complete** Pop-up a list of possible completions. For tables and views this would give a list of columns after entering table/view name and a dot. If there is no dot a list of common statements is given. For example if you press ctrl+space after entering "sel" you will get options of "select" and "selection".

**Edit SQL** This may not be available because it is actually a tool plugin. Can be used to customize SQL used by TOra to determine the state of a database(see [Section 4.19 \[sqledit\], page 78](#)).

## Chart Manager

## Tools menu

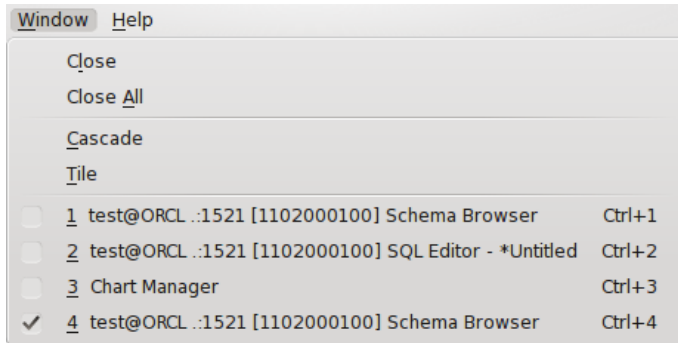


The entries of this menu start up new tools for the current connection (See connection management for more information about current connections). The contents of the menu may vary depending on which plugins are loaded.

### Tool specific menu

There may be a tool specific menu entered between the Tools and Windows menus (In the screenshot at the top of current page it is called Browser). These are described in connection to the tools that display them.

## Windows menu



**Close** Close the current tool window.

**Close All** Close all open windows. This will not close any database connections, only windows.

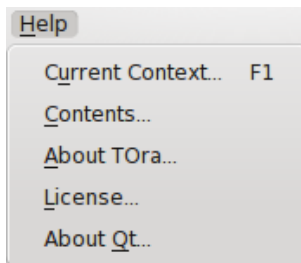
**Cascade** Cascade the open windows over the workspace.

**Tile** Tile the open windows over the workspace.

### Open windows

Here windows will line up in a list in the same order as you open them.

## Help menu



### Current Context

This will display a help page on the context you are currently in. If no specific context is found the table of contents is shown.

**Content** Open the help window displaying the contents of the TOra help.

### About TOra

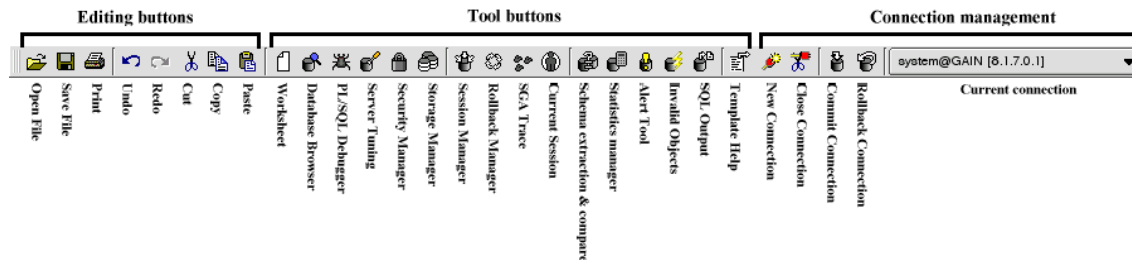
Display information about TOra copyright.

**License** Display the license under which TOra is distributed.

**About Qt** Display information about Qt.

## Application toolbar

Using the toolbar you can perform the most common editing operations, handle your connections and open tool windows.



## Editor buttons

- Load** This command will open a dialog asking you for a filename to load into the editor that currently holds the focus.
- Save** This command will open a dialog asking you for a filename to save the contents of the editor that currently holds the focus. If the editor already has a filename this will be used and no dialog displayed.
- Print** Will print the contents of the list or editor that currently holds the focus
- Undo** Undo the latest change in the editor that currently holds the focus.
- Redo** Redo a previous undo of the latest change in the editor that currently holds the focus.
- Copy** Copy the current selection the in editor that currently holds the focus.
- Cut** Cut the current selection the in editor that currently holds the focus.
- Paste** Paste the contents of the clipboard into the editor that currently holds the focus.

## Tools

The buttons in this section can vary depending on which plugins were loaded at startup, they are described in the tools section (see [Chapter 4 \[tools\], page 34](#)) of the help. What is notable here is that when a button is pressed it is started for the currently selected connection available at the end of the application toolbar.

## Connection management

All the buttons in this section have one thing in common. Except for the new connection they all operate on the currently selected connection. The currently open connections are available in a drop down list at the right of the toolbar. The selected item in this list is the connection that new tools will be working on and also any commit, rollback or close connection command.

The numbers behind the actual connection is the version of the Oracle server that this connection is made to.

**New connection**

This will pop up the new connection (see [Section 3.2 \[newconnection\]](#), page 12) dialog which is used to open new connections to oracle.

**Close connection**

Will close the current connection to Oracle. If uncommitted data exists it will also ask if you want to commit or rollback the changes. Observe that any tools associated with this connection will also be closed when closing a connection.

**Commit connection**

Commit the changes currently made to the database.

**Rollback connection**

Rollback any changes made in the current connection.

**Other items in the workspace**

There are a few other items on the workspace that are worth taking notice of.

**Statusbar**

This will display any messages that TOra need to inform the user of. TOra generally takes the path of not bothering the user with pop ups when something goes wrong, but simply display it in the statusbar. Or at least that was my intention. Due to the fact that many people had problems understanding this the default behaviour of TOra is now to pop up a dialog displaying the error. However this dialog contains a checkbox to revert back to the old productive manner. So if something isn't working, make sure you check the statusbar if something went wrong.

There is an up button available to the right of the toolbar which when pressed can display a history of the contents of the status bar. This is also useful since the statusbar usually clears after certain amount of time but the information is always available here. Both the time until a message disappears and the amount of backlog to keep is configurable from the option dialog (see [Section 3.8 \[preferences\]](#), page 20). If you select an item in the popup menu the contents of the entry will be opened in a memo editor (see [Section 3.9.5 \[memoeditor\]](#), page 32) where you can read them more easily and also copy the text more easily.

Next to the left is the coordinates of the editor that currently has the focus.

Furthest to the right is swirling logo which will move whenever TOra is doing something in the background to indicate that the database is still busy performing some kind of operation in the background. The speed of the animation will reflect the number of background tasks that are currently running.

**Tool windows**

These are the tools that actually let you interact with Oracle. For more information check out the tools section (see [Chapter 4 \[tools\]](#), page 34) of the help. One thing to note about tool windows is that if the caption ends with a < digit > this is the digit that you can use together with the control modifier to select this window.

**Workspace**

The part of the main window that contain the tool windows.

## Editors

For more information about key bindings etc. see [Section 3.4 \[editors\]](#), page 14.

## Result List

These mean every kind of list displayed in TOra. For information about how to interact with these, see [Section 3.5 \[lists\]](#), page 15.

## Charts

Used to visualize data, usually some kind of database statistics. There are no charts in the example window above, see [Section 3.6 \[charts\]](#), page 17.

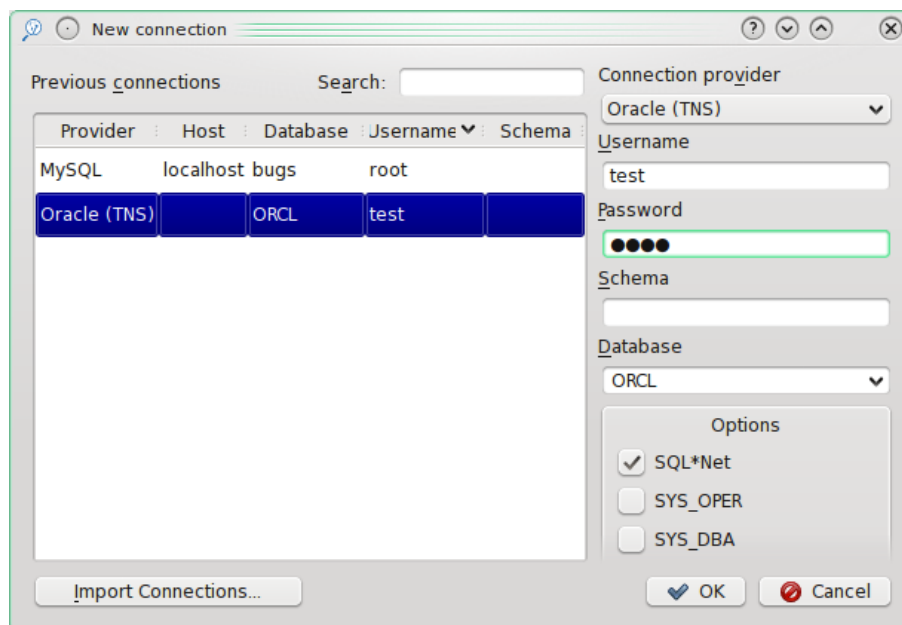
## Object Cache

When TOra opens a new connection to a database the database connected to is queried for all its objects and the synonyms available. This data is then used in most places when lists of objects are displayed. The thing to understand is that this data is cached so that any objects modified during the connection will not be visible unless you clear the cache and reread it from the database. As long as the object cache is read you can see the **background activity** swirling in the right part of the status bar.

## 3.2 Connecting to a database

You can connect to a database by selecting the **File|New Connection** menu entry or the toolbar button. For more information about the menu and toolbar see [Section 3.1 \[window\]](#), page 4.

After selecting the menu you will be presented with the following dialog.



The filling out of this dialog is pretty straight forward.



**Previous** This list contains the previous connection made, selecting an entry from this list will set all the values to the right to the correct. Doubleclicking will connect straight away (only makes sense if you save passwords).

**Connections**

Values to connect to this database again.

**Connection provider**

The type of connection to create. You can choose from Oracle, MySQL and others.

**Username** The username with which to connect to the database

**Password** The password to use for the connection, this defaults to manager. You can also configure TOra to remember your last used password (see [Section 3.8 \[preferences\], page 20](#))

**Use SQL\*Net**

This is perhaps the most difficult setting. This indicates whether to connect locally to the database or use SQL\*Net. If you check this box the connection is made through the Oracle listener, otherwise a local connection usually using shared memory or named pipes is made. This is only available for Oracle connections.

**Hostname** The host on which the database resides. This is only available for non Oracle connections.

**Database** What database to connect to, the listbox is filled with the values available in your tnsnames.ora file.

**Connection Mode**

The type of connection to make, usually **normal** always suffices here. But sometimes during database administration **SYS\_DBA** or **SYS\_OPER** is needed. If you don't know what these mean you probably shouldn't be using them.

Upon establishing a successful connection all the settings of the dialog will be saved and remembered when the next connection is made with the exception of the password.

### 3.3 What privileges do you need to run TOra

TOra should be able to run as any user, although off course you can never do anything in TOra that aren't allowed from that Oracle account. You will be able to use pretty much any tool in TOra for read only purposes if you are granted the **SELECT\_CATALOG\_ROLE**, also some parts of TOra will have limited functionality if you do not have the **ALTER SESSION** privilege.

Also you need to be able to create the plan table if that is not done for you. The plan table (see see [Section 3.8.3 \[database\], page 26](#) for how to select plan table name) must also be available with **INSERT** and **SELECT** access. TOra will function without the plan table but off course you will not be able to display any execution plans.

### 3.4 Using editors

Editors can be in two modes. The first is read and write in which case the following key bindings are defined.

<b>Left Arrow</b>	Move the cursor one character to the left
<b>Right Arrow</b>	Move the cursor one character to the right
<b>Up Arrow</b>	Move the cursor one line upwards
<b>Down Arrow</b>	Move the cursor one line downwards
<b>Page Up</b>	Move the cursor one page upwards
<b>Page Down</b>	Move the cursor one page downwards
<b>BS</b>	Delete the character to the left of the cursor
<b>HOME</b>	Move the cursor to the beginning of the line
<b>End</b>	Move the cursor to the end of the line
<b>Delete</b>	Delete the character to the right of the cursor
<b>SHIFT - Left Arrow</b>	Mark text one character to the left
<b>SHIFT - Right Arrow</b>	Mark text one character to the right
<b>CTRL-A</b>	Select the entire buffer.
<b>CTRL-B</b>	Move the cursor one character leftwards
<b>CTRL-C</b>	Copy the marked text to the clipboard
<b>CTRL-D</b>	Delete the character to the right of the cursor
<b>CTRL-E</b>	Move the cursor to the end of the line
<b>CTRL-F</b>	Pop up the search & replace dialog
<b>CTRL-H</b>	Delete the character to the left of the cursor
<b>CTRL-K</b>	Delete to end of line
<b>CTRL-N</b>	Move the cursor one line downwards
<b>CTRL-P</b>	Move the cursor one line upwards
<b>CTRL-V</b>	Paste the clipboard text into line edit
<b>CTRL-X</b>	Cut the marked text, copy to clipboard
<b>CTRL-Z</b>	Undo the last operation
<b>CTRL-Y</b>	Redo the last operation
<b>CTRL - Left Arrow</b>	Arrow Move the cursor one word to the left
<b>CTRL - Right Arrow</b>	Arrow Move the cursor one word to the right
<b>CTRL - Up Arrow</b>	Move the cursor one word upwards
<b>CTRL - Down Arrow</b>	Move the cursor one word downwards
<b>CTRL - Home Arrow</b>	Move the cursor to the beginning of the text
<b>CTRL - End Arrow</b>	Move the cursor to the end of the text
<b>CTRL-S</b>	Save the editor with the current filename, if no filename is defined a dialog is displayed for the user to select the filename to use.
<b>CTRL-O</b>	Read a file into the editor. A dialog will pop up to select the name to open.

The second mode is read only in which only the current key binding are available.

<b>Left Arrow</b>	Scrolls the table to the left
<b>Right Arrow</b>	Scrolls the table to the right
<b>Up Arrow</b>	Scrolls the table one line downwards
<b>Down Arrow</b>	Scrolls the table one line upwards
<b>Page Up</b>	Scrolls the table one page downwards

<b>Page Down</b>	Scrolls the table one page upwards
<b>Control-C</b>	Copy the marked text to the clipboard
<b>Control-S</b>	Save the editor with the current filename, if no filename is defined a dialog is displayed for the user to select the filename to use.

Editors support drag and drop which means you can drag a selection to another editor and drop any other text source onto an editable editor to insert the dragged text at the dropped location.

Also as a shortcut pressing the right mouse button will display a context menu containing the entries from the edit menu which pertain to the editors.

## SQL editors

Editors for SQL have another feature which is code completion. If you place the cursor after a '.' character TOra will list the available columns for the table referenced by the name before the '.' character. This will only work for defined views and tables. You can also look up tablename using the **CTRL+T** key which will list all tables starting with the same word as you currently have the cursor in. When the completion list is open you can navigate it using **Up**, **Down** keys and selecting an entry using **Return**. You can also remove the list by pressing the **Escape** key. The completion values are saved in the object cache and can not be used until it is read. If you find this annoying you can disable this in the preferences.

## 3.5 Using lists

A list is a very central element of the TOra user interface and is a standard way to display information in a grid. Here is an example of a list.

Name	Status	Information	Contents	Logging	Size (MB)	Free (MB)	Autoextend (MB)	Use
USERS	ONLINE	LOCAL	PERMANENT	LOGGING	5	.75	32767.98	0.0
/ora/base/oradata/orcl/users01.dbf	ONLINE	READ WRITE			5	.75	32767.98	-
UNDOTBS1	ONLINE	LOCAL	UNDO	LOGGING	60	49.63	32767.98	0.0
/ora/base/oradata/orcl/undotbs01.dbf	ONLINE	READ WRITE			60	49.63	32767.98	-
TEMP	ONLINE	LOCAL	TEMPORARY	NOLOGGING	93	92	32767.98	0.0
/ora/base/oradata/orcl/temp01.dbf	ONLINE	READ WRITE			93	0	32767.98	-
SYSTEM	ONLINE	LOCAL	PERMANENT	LOGGING	680	6.5	32767.98	2.0
/ora/base/oradata/orcl/system01.dbf	SYSTEM	READ WRITE			680	6.5	32767.98	-
SYSAUX	ONLINE	LOCAL	PERMANENT	LOGGING	570	28.81	32767.98	1.0
/ora/base/oradata/orcl/sysaux01.dbf	ONLINE	READ WRITE			570	28.81	32767.98	-
EXAMPLE	ONLINE	LOCAL	PERMANENT	NOLOGGING	100	21.56	32767.98	0.0
/ora/base/oradata/orcl/example01.dbf	ONLINE	READ WRITE			100	21.56	32767.98	-

### Navigating a list

If a list does not fit into its designated area on the screen it will get scrollbars that can be used to navigate the entire contents of the list. There are also the usual key bindings **left**, **right**, **up**, **down**, **page up**, **page down** etc...

Lists can be hierarchical as can be seen in the screenshot above. When there are child items that are not displayed a + is displayed to the left of the row. Clicking the + will expand the row and display the child items. You can also use the **right** key to expand child items and **left** key to collapse them.

## Result lists

Result lists are lists that display the result of a query from the database. Only a configured amount of rows are initially read (see [Section 3.8 \[preferences\]](#), page 20 for more information), although enough items to fill the screen is always read if available. To read more items from the queries simply scroll down. If you want to scroll faster use **page down** which will read an entire screen of data at a time from the query. If you want to read all the data available you can use the **Edit|Read All** menu item.

## Manipulating lists

In a list you can change the sort order by pressing the header of the list to sort on the column whose header you clicked. Clicking the same column again will reverse the sort order. Columns that contain numerical data (Regardless of database format) will be sorted as numbers (2 before 10). Observe that not all lists can be sorted in a different sort order.

You can also rearrange the column order by dragging the heading of a column to another place.

## Accessing the contents of a field

The contents of a field (Specific row and column of a list) can be copied either by double-clicking the field, or selecting copy from the context menu available by right-clicking on the field. You can also drag the contents of a field to an editor.

If the contents is too long to be visible in the list you will get a tooltip with all of the contents if you place the mouse over the field for a short while (As seen in the screenshot). If this is not enough you can display the contents of a field in a separate window by accessing the context menu of the field and selecting **display in editor**.

You can also place a line in the bind parameter(see [Section 3.9.3 \[gettingbindparameters\]](#), page 31) cache by double clicking on a line. The bind name will be the same as the column name in lowercase.

## Other list functionality

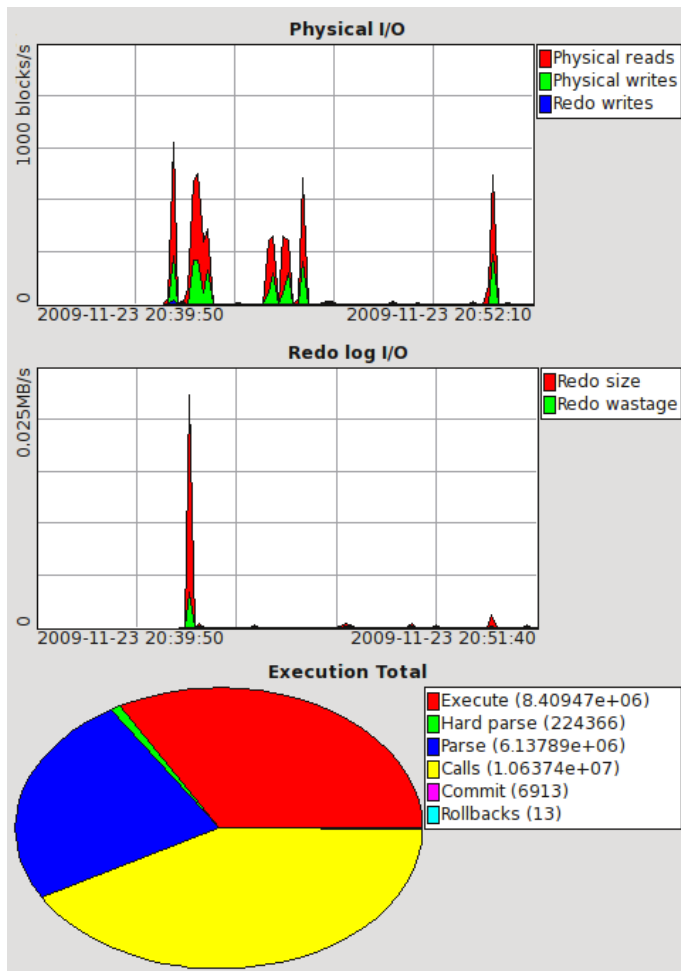
Lists can always be printed by selecting the **File|Print** menu item.

You can also save a list as a text file by selecting **File|Save** menu item. Any graphics in the list will be discarded and all fields will be left justified in the output.

In the context menu for the list you can also select the entry **edit SLQ** in which you can edit the SQL used to generated the list. If there is no single SQL used to generate the list you will not find any SQL, however you will usually be pointed in roughly the right direction in the sql editor (see [Section 4.19 \[sqledit\]](#), page 78, Where you can also find more information about the concept of SQL dictionary TOra uses).

### 3.6 Using charts

There are several different chart types in TOra, you can see them in the screenshot below (taken from Server Tuning tool).



#### Piecharts

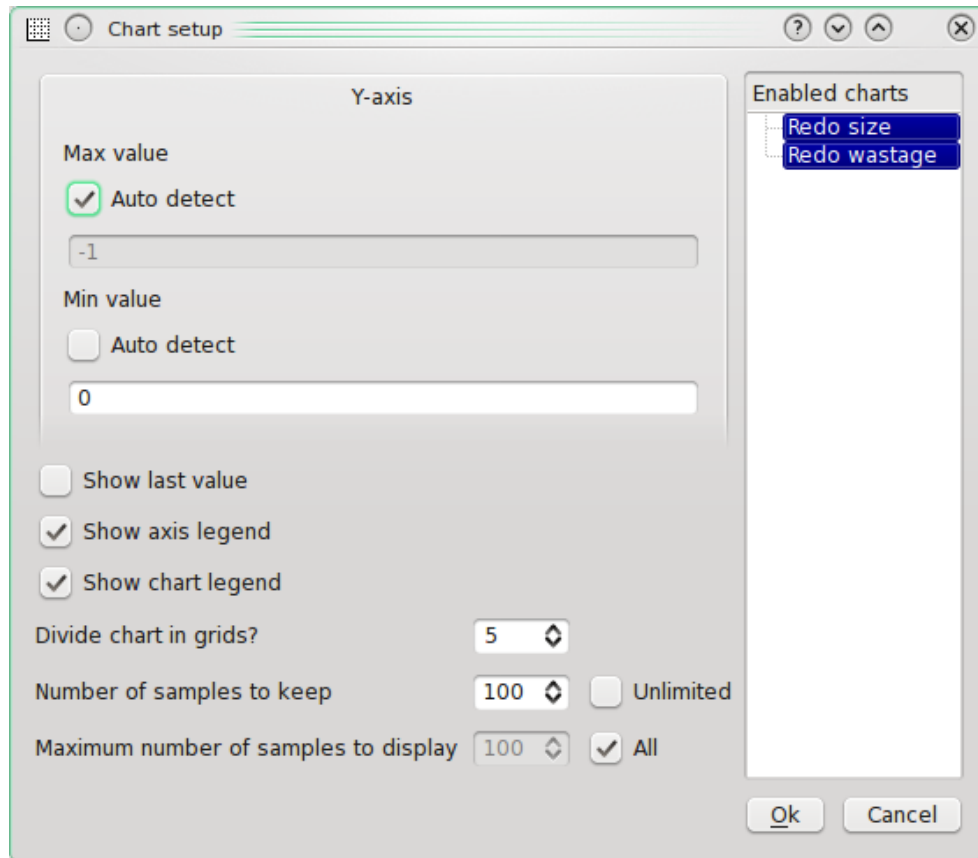
This is the most simple chart. If the chart is too small you can open a snapshot of the chart by doubleclicking it. You can also open a context menu by rightclicking it. In the context menu you can choose to print the chart as well and open a snapshot in a new window.

#### Linecharts & barcharts

These two charts work in the same way. The difference is that in a barchart the values are added on top of each other so you can easily see the sum of all values.

You can zoom a part of a chart by leftclicking it and dragging a selection. If a chart is currently zoomed the word **Zoom** is printed in the upper left corner. You can remove the zooming by right clicking on the chart.

The snapshot feature, context menu and printing works the same for these charts as for the piechart. In the context menu you have an additional item though that is **Properties**



**Y-Axis** This part is pretty self explanatory and is used to set up the range to display on the Y-axis of the chart.

**Show last value**

Display the last sample added to the chart under the title of the chart.

**Show axis legend**

Display text with the range and units of the axes.

**Show chart legend**

Display the colors used in the chart in a legend to the right.

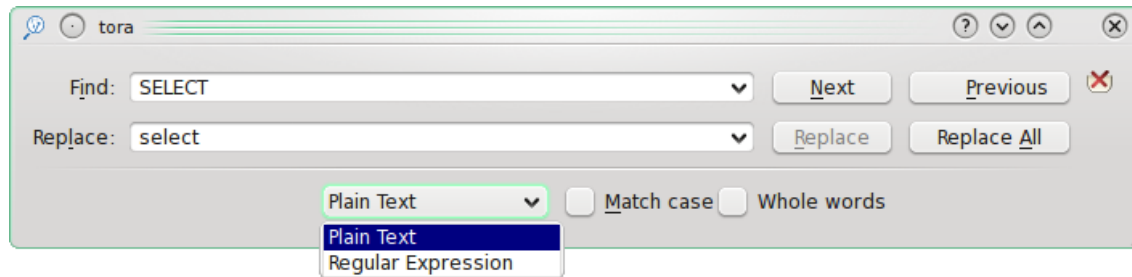
**Divide chart in grids**

How many gridlines to divide a chart into.

### 3.7 Search & Replace

Tora has an advanced search and replace function that allows you to search result list and editors. Replace is only available for editors though. You can invoke the search & replace dialog by selecting the menu **Edit|Search & Replace** or by pressing **CTRL+F**. The search

always operates on the last editor or list that had the focus **when you invoked the dialog**. Not on the current widget of the next search etc. The dialog looks like this.



## Specifying the search criteria

You enter the text to search for, this can be either in the format of plain text or regular expressions as defined by the current version of Qt which TOra is compiled against (For more information check out <http://doc.trolltech.com/qregexp.html>, if you are using Qt 3.0 or later you have a more advanced form of regular expressions).

You can also specify if the search expression must match a whole word or if the match is to be case sensitive.

After specifying what you are looking for you can execute your search by either pressing the **Search Top** button which will start the search from the top of the contents of the target for the search or **Search Next** which will continue searching for the next match from the current location.

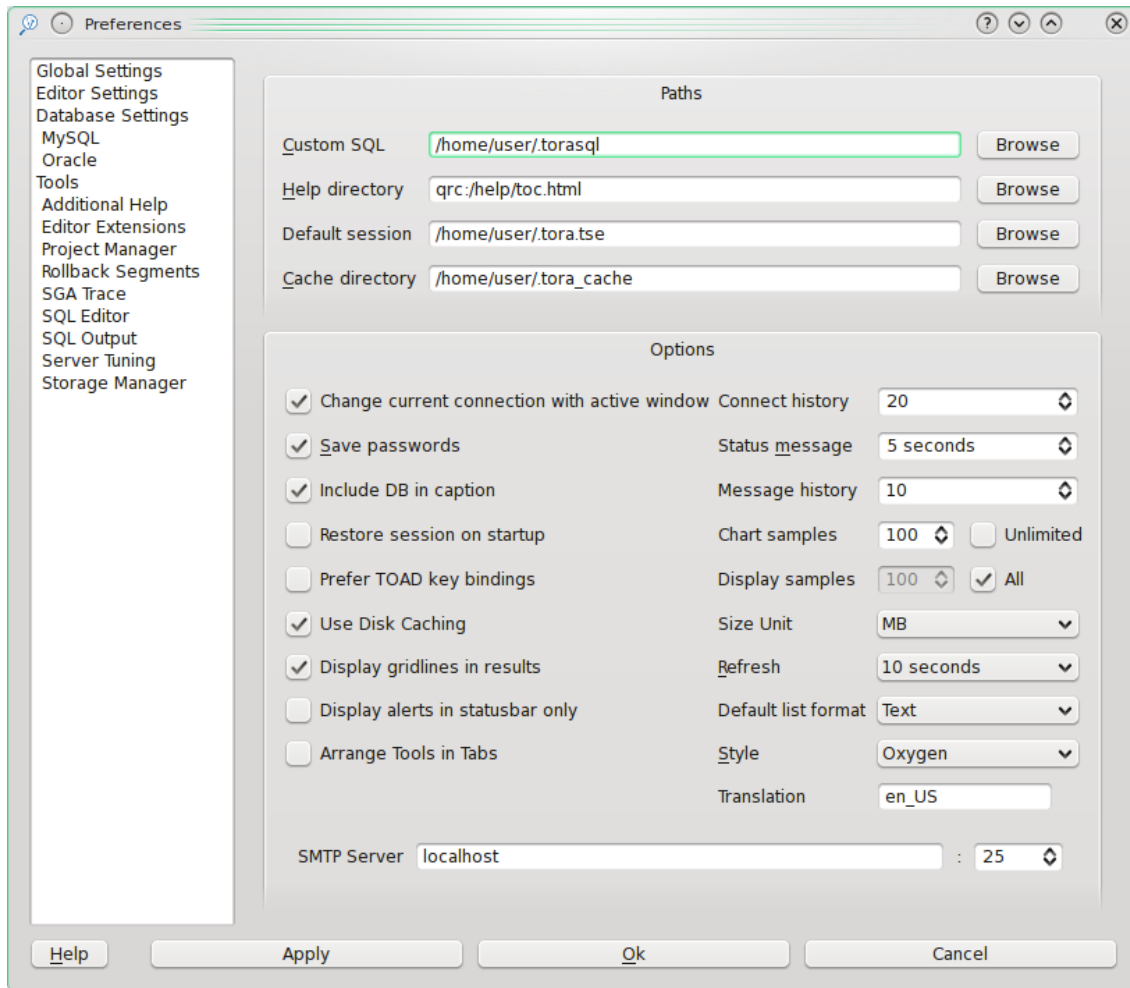
## Replacing

In the replace text field you can specify the text with which to replace a matched find that has been found. Note that regardless of if the search is case sensitive or not the replacement is the same case as specified in the text field.

After finding the first occurrence by pressing **Search Top** or **Search Next** you can either replace it with the current substitution by pressing **Replace**, or skip to the next by pressing **Search Next** button again. At any time the **Replace All** will replace all occurrences after the current location in the editor.

### 3.8 Preferences

To access the preferences of TOra you select the **Edit | Preferences** menu item, you will then be presented with the following dialog.



In windows preferences are stored in the registry, but in Linux it is stored in a file called `‘.torarc’` in the users home directory. If this file doesn't exist a file called `‘/etc/torarc’` is used if it exists.

#### Selecting page

To the left is a list containing the available option pages you can display.

The rest of the settings are tool specific and covered under corresponding tool in the tools section(see [Chapter 4 \[tools\], page 34](#)).

Selecting a page does not discard the changes of another page, and no changes are saved until **OK** is pressed. You can always discard any changes made by pressing the escape key or the **Cancel** button.



### 3.8.1 Global Settings

These may vary some depending on how your system was compiled and configured. Not all options are displayed in the screenshot since that was made on a KDE version of TOra.

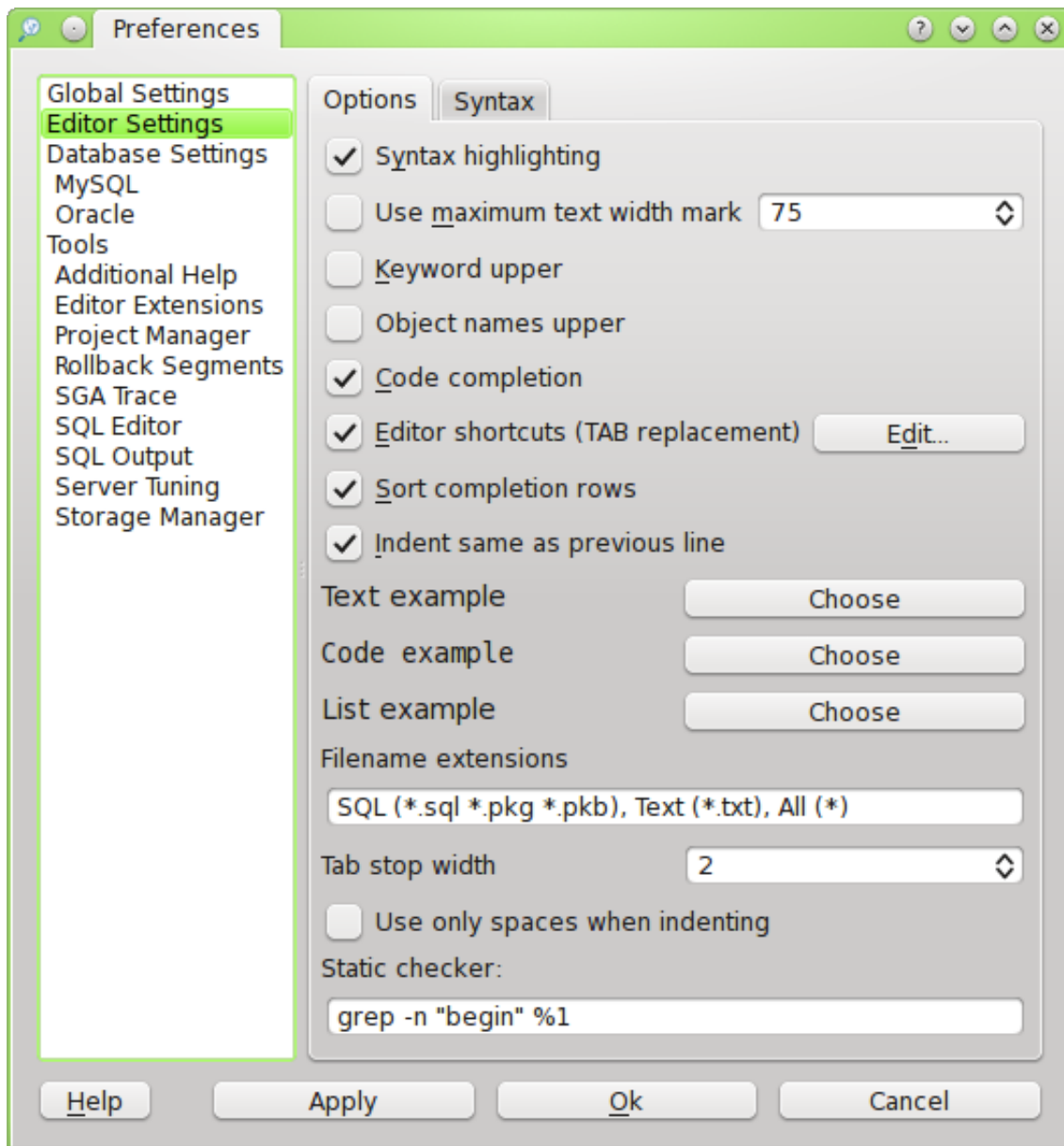
<b>Custom SQL</b>	The file in which to store your custom SQL. For more information about custom SQL and the SQL dictionary (see <a href="#">Section 4.19 [sqledit], page 78</a> ). The string <code>\$HOME</code> will be replaced by your home directory on UNIX or documents directory in Windows.
<b>Help directory</b>	Should point to where the TOra help table of contents file is stored (toc.html). This should be set up correctly if you made a normal install.
<b>Plugin directory</b>	This option is only available in a plugin based TOra installation. It should point to the directory where the plugins for TOra are located. It should be set up correctly on installation if needed.
<b>Cache directory</b>	Should point to where the TOra Disk Cache files are stored. There is one file for each Connection.
<b>Change current connection with active window</b>	If this option is set, changing the current window will also change the current connection to that of the active window. The connection is only changed when you change window so if you want to change the connection to something else you can still do so as long as you don't change the window.
<b>Save last password</b>	Whether or not TOra should store the last password used to connect. <b>Be aware that the password is store in clear text in either the windows registry or a file.</b>
<b>Include DB in caption</b>	If this is set the database name will be included in the caption of all your tool windows after the tool name. Observe that there often is no other way of knowing which database the window works on except for this.
<b>Maximize window on start</b>	If this is checked the TOra main window will open maximized on startup.
<b>Desktop Aware</b>	Set this if Qt should try to figure out and interact nicely with the desktop environment. The reason for this switch is that in some versions of KDE TOra doesn't look nice if this is checked. This switch is only available for Qt only applications.
<b>use Disk Cache</b>	This option tells TOra to Cache the Object List in the Cache Directory. Default is No Caching. When Caching is activated, you have to refresh the object cache with the menu option File->Reread Object Cache, because, the default source for the Object List is the diskfile.
<b>Antialias fonts</b>	Check this if you want TOra to use antialiased fonts in X11 if available. This option may not be available (Regardless of if you have access to antialiased fonts or not).

<b>Docks toolbars</b>	<b>use</b>	This option tells TOra how to emulate docked windows when they are not available. Docked windows are not available if you do not use KDE and have a Qt version less than 3 (If you run in windows you're it). If this is checked docks will be emulated with a toolbar, this has the drawback that you can't resize them. If unchecked docks appear in normal windows.
<b>Style</b>		The GUI style to use for the TOra application. This switch is only available for Qt only applications.
<b>Display alerts in statusbar only</b>		Display errors only in the statusbar. This is really usefull since it doesn't interrupt you as much as a dialog window as long as you know to look for information in the statusbar when things doesn't work.
<b>Connect history</b>		How many connection entries to save in the new connection history list.
<b>Table scale</b>		This is the scale which to use when printing lists. Lists on paper can usually be a little smaller and still be readable. Given as a fraction where 0.5 means half the size on paper and 1.0 the same size.
<b>Status message</b>		The number of seconds to display error messages in the status bar. If you specify 0 they are kept until the next message is displayed.
<b>Message history</b>		The number of status messages to store in the status message history before the oldest are removed when new are added.
<b>Chart samples</b>		Number of samples to save by default in charts.
<b>Size Unit</b>		The unit to display sizes with in TOra.
<b>Refresh</b>		The default refresh time to use in TOra. Tools which refresh automatically use this setting as the default. It can also be changed on a per tool basis.

### 3.8.2 Editor Settings

This page of the options contains settings for TOra editors. Editor settings are organised in two tabs: Options and Syntax.

## Options tab

**Syntax highlighting**

Use this to control whether you want syntax highlighting or not. Observe that without syntax highlighting you will not see the indications for errors and current lines in the PL/SQL editor.

**Use maximum text width mark**

Check this option and specify maximum length of text. A vertical line indicating this margin will be displayed in text editor. Note that this option will **not** prevent text to be written beyond maximum text width margin. this option is purely for visual representation.

**Keyword upper**

Check this box to convert all keywords to uppercase before displaying them. The text itself is not changed so resetting this not changed and the text sent to the database still contain lowercase letters. This setting is disabled if you do not have a monospaced font selected.

**Object names upper**

Extract all names of objects (tables, views, columns etc.) from database as uppercase (default). For example this setting will be used when generating object creation scripts.

**Code completion**

Check this box to enable code completion in SQL editors.

**Editor shortcuts (TAB replacement)**

By checking this option and pressing a button "Edit..." you can add/modify/delete editor shortcuts. Editor shortcut is a short word which can be replaced to a long phrase by pressing TAB button. For example you can set that a short word "selsysd" be replaced with "select sysdate from dual".

**Sort completion rows**

If checked the completion alternatives are sorted in alphabetical order, otherwise they are in the same order as in the source.

**Indent same as previous line**

When inserting a new line indent it to the same level as the previous one.

**Text example**

This is the font to use for all editors in TOra that are not syntax highlighted. Choose the **Choose** button immediately to the right of the label to select a new font.

**Code example**

This is the font to use for all syntax highlighted SQL editors in TOra. Choose the **Choose** button immediately to the right of the label to select a new font.

**List example**

This is the font to use for all lists in TOra. Choose the **Choose** button immediately to the right of the label to select a new font.

**Filename extensions**

TODO.

**Tab stop width**

With of tab character. This value is also used when indenting automatically generated/formated scripts.

**Use only spaces when indenting**

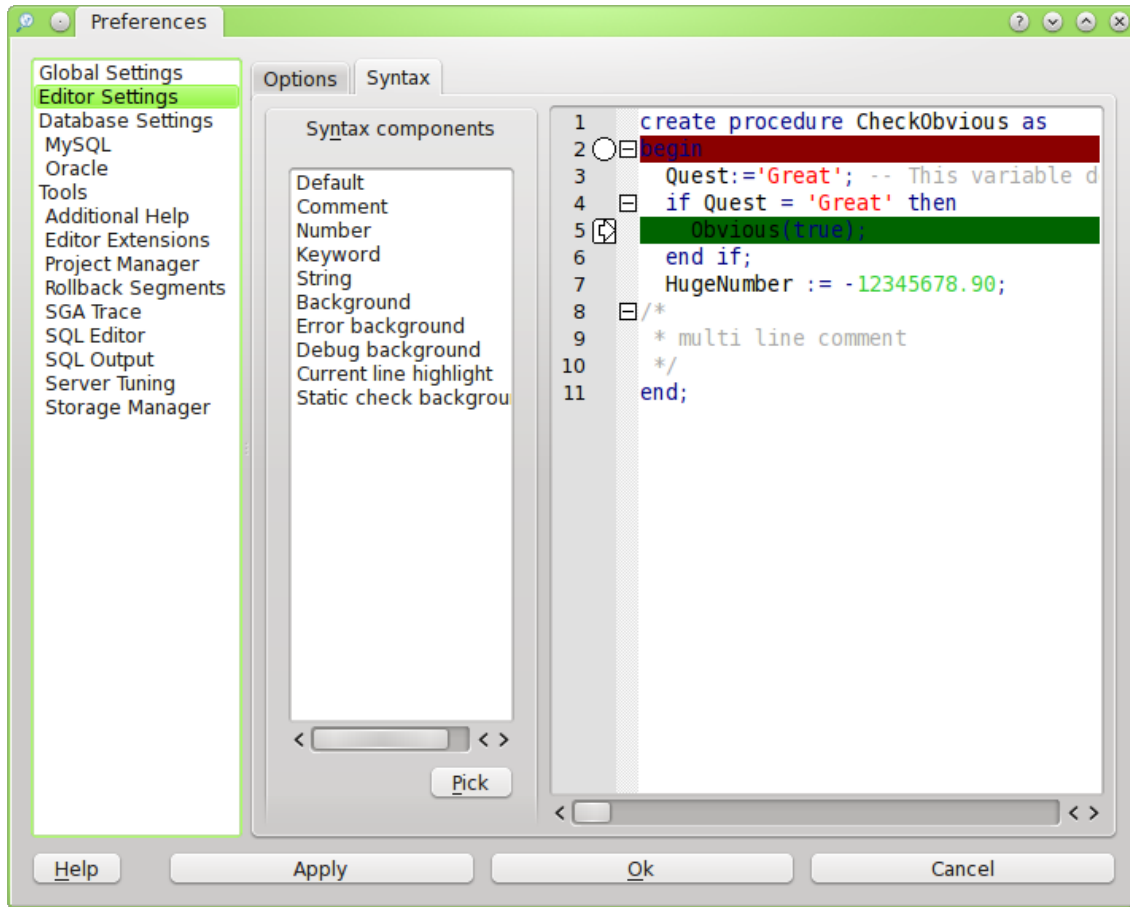
TODO.

**Static checker**

Specifies static checker to be used. This configuration option should contain %1 which will be replaced with a filename (with full path) containing code to be checked. For example:

```
perl /usr/bin/staticChecker.pl --output=TORA --file=%1
```

### Syntax tab



### Syntax components

This list contains different parts of the SQL syntax. You can change the color used for the element by selecting the item in this list and then pressing the Pick button to select a color. The current color of the currently selected item in the list is displayed just to the left of the Pick button.

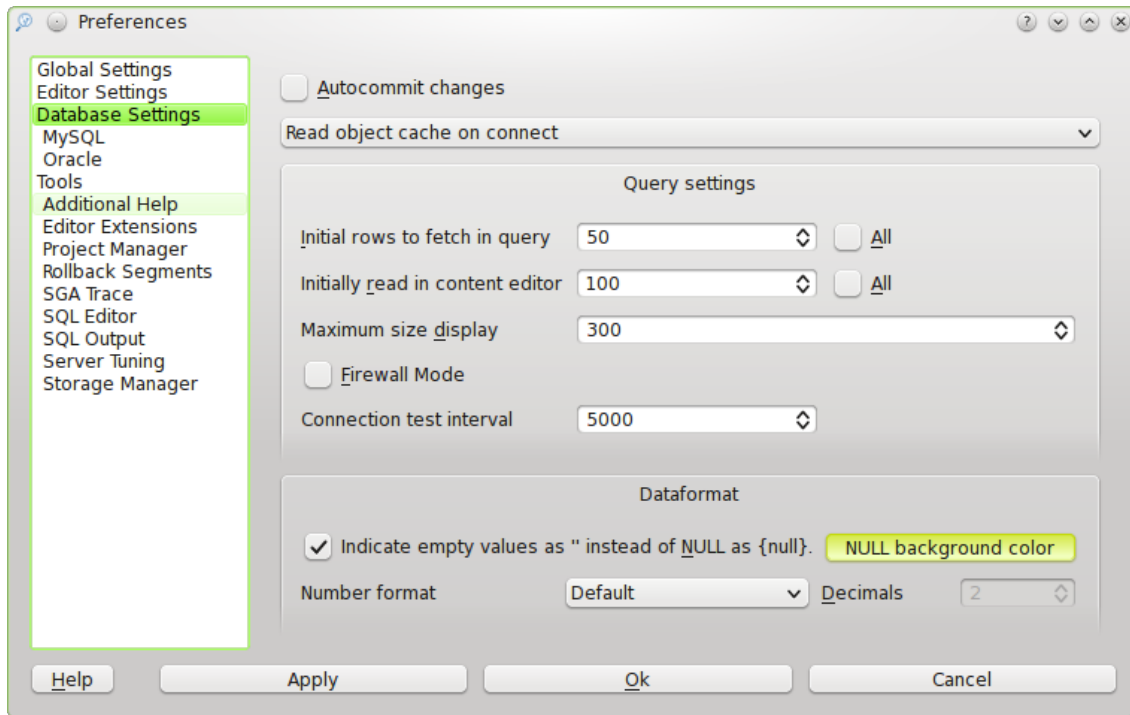
The available elements are as follows.

<b>Background</b>	The normal background color for text.
<b>Comment</b>	An SQL comment. Currently this only includes – comments, not rem comments.
<b>Current background</b>	This is the background to use for the current line of execution in the PL/SQL debugger.
<b>Error background</b>	This is the background to use for lines that have SQL errors in the in the PL/SQL debugger/editor.
<b>Keyword</b>	This is the color to use for the text of an SQL keyword.
<b>Normal</b>	Normal text color.

**String** Color to use for text in strings.  
**Unfinished string** Color to use for strings that miss their terminating ' or ".  
 At the bottom of this dialog is a text field that displays an example of the current setting.

### 3.8.3 Database Settings

This page of the options contains settings on how to interact with the database.



#### Autocommit changes

If this is set any changes made to the database will be automatically committed as soon as you make them. Use this with care, the transaction handling is there for a reason you know.

#### Reading object cache

There are four options of object cache behaviour. **Start read object cache when needed** - will start reading it when it is first used. **Read object cache on connect** - Tora will start reading the object cache when you start a connection to a database. **Never read object cache until mandatory** - TODO. **Never read object cache until mandatory (Will leave some parts of application nonfunctional** - TODO. Note that reading this is quite a costly query so you probably shouldn't have this checked if you are in a production environment, on a slow connection or not using the object cache. The object cache is mostly used for describing tables, the PL/SQL debugger and code completion.

#### Initial rows to fetch in query

This setting indicates how many rows to always fetch from a query when executing it. Observe that you can always read more from the individual query

just by scrolling down. Check the **All** checkbox to always read all available records. Be aware that TOra will interact sluggishly or not at all while reading many lines. It will also keep the entire result in memory so checking the **All** box and reading a really large table could bring your machine to it's knees.

**Initially read in content editor**

If set to something other than **All** a query will be placed only this number of rows initially from Oracle when browsing the content. The reason for this is that Oracle will be under high load if a content list is started on a large table. When this value is reached the query will be reread to fetch the rest of the values so you as a user will not see any difference.

**Maximum size display**

The maximum width a list column will get automatically regardless of how large the content is. This is so you can still see the rest of the columns even if one of the first can contain very large values. You can always resize the column by hand afterwards.

**Firewall mode.**

Makes each connection in connection pool to run extra queries (selecting sysdate from dual) at specified interval (see "Connection test interval" option). This could be useful if there is a firewall between TOra and database and firewall is dropping connections if there is no traffic.

**Note!** If you change this option, you have to reconnect in order to make it effective.

**Connection test interval**

Interval in seconds at which TOra should be sending dummy queries to database (see "Firewall mode").

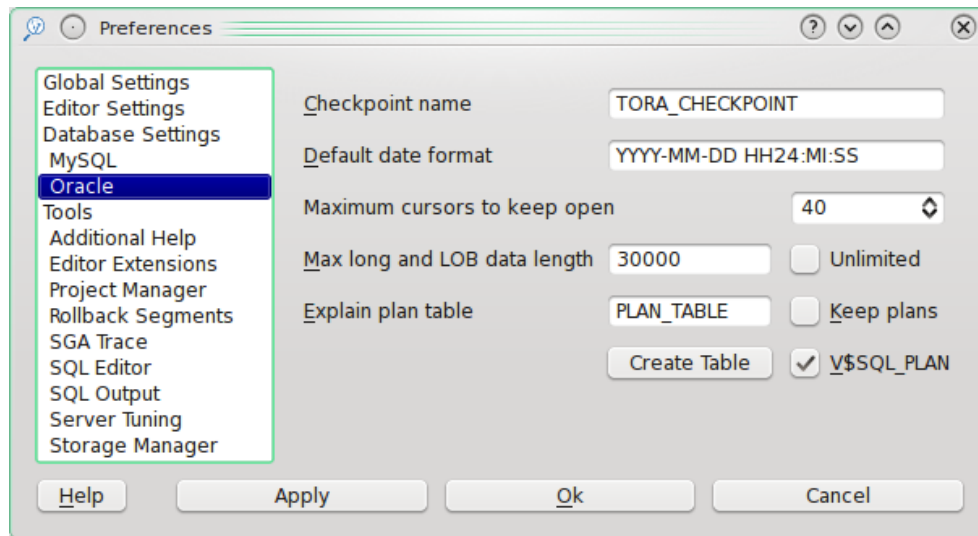
**Indicate empty values as ”.**

TODO.

**Number format.**

TODO.

There may also be pages under this for each individual database provider. Currently only Oracle and MySQL uses this feature. That setup looks like this.



### Checkpoint name

This is a checkpoint TORA should use when it is making changes that need to be rolled back. One place where this is used is to perform explain plans without filling up the plan table. The name doesn't matter, just make sure you don't use this checkpoint name yourself.

### Default date format

The format that you prefer dates to be displayed. Changing this will require a reconnect to the database before it starts to operate. The string should be a standard Oracle date to string conversion specification. For more information see the Oracle SQL Reference.

### Maximum cursors to keep open

TODO.

### Max LONG and LOB data length

The amount of data to read from LONG, BLOB or CLOB data fields. If you always want all the data to be read check the **Unlimited** checkbox to the right. LONG datafields can not be specified as **Unlimited**, if this is specified 33000 characters are read by default.

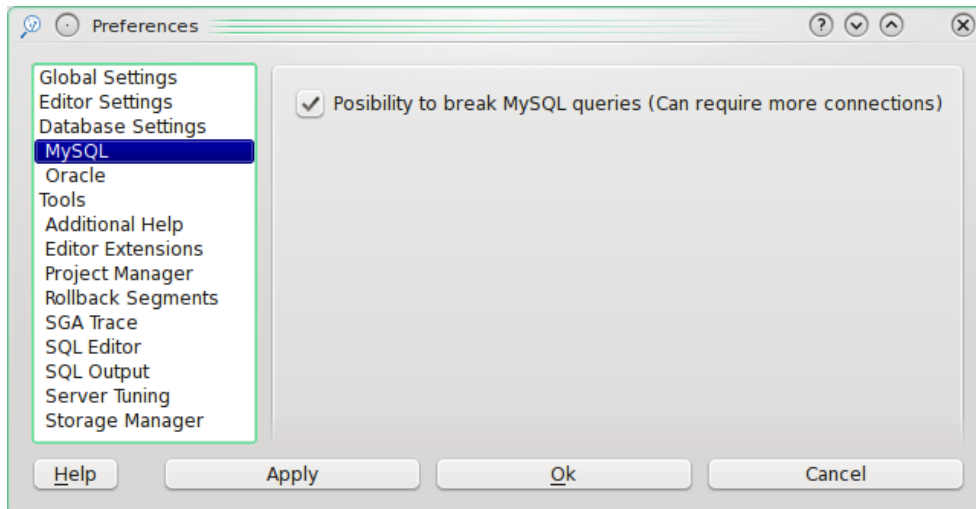
### Explain plan table

The table to use to temporarily store explained execution plans. Use the **Create Table** button to create the table if you don't have it.

### Keep plans

If this is checked explained plans are left in the explain plan table. The normal behaviour is to rollback the explanation after it is initially read to keep the explain plan table from filling up with old plans.

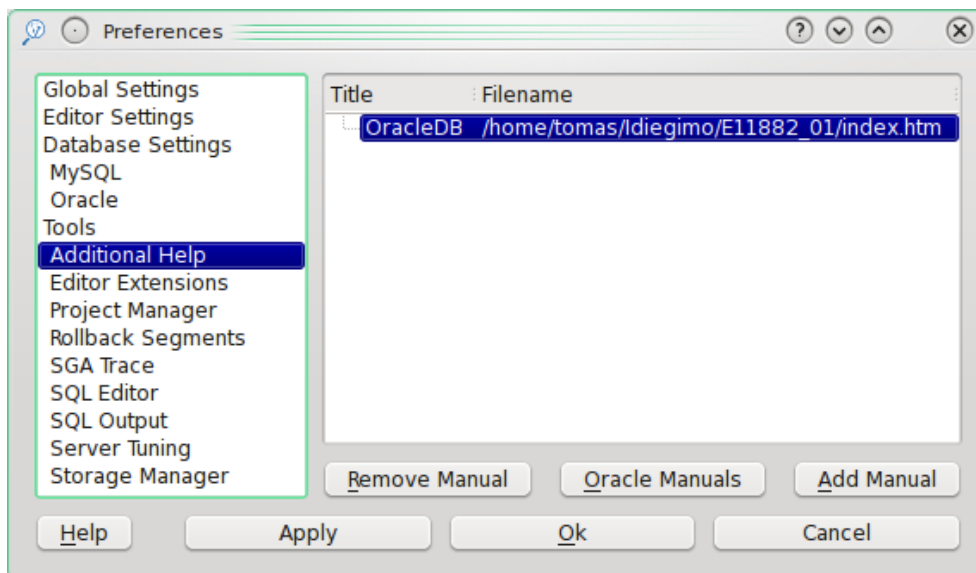




**Possibility to break MySQL queries**  
 TODO.

### 3.8.4 Additional Help Settings

This page is used to configure extra manuals for viewing in the TOra help viewer. Specifically the Oracle HTML based manuals which contents can be parsed and included in the left pane contents tree of the TOra help.



If you have compiled TORA with KDE support (Not Windows or Qt versions) you can point this to any HTTP location. You could even use the help browser to browse the Oracle manuals directly from Technet if you wanted to (Free membership required).

#### Remove path

This will remove the currently selected manual from the list.

#### Oracle Manuals

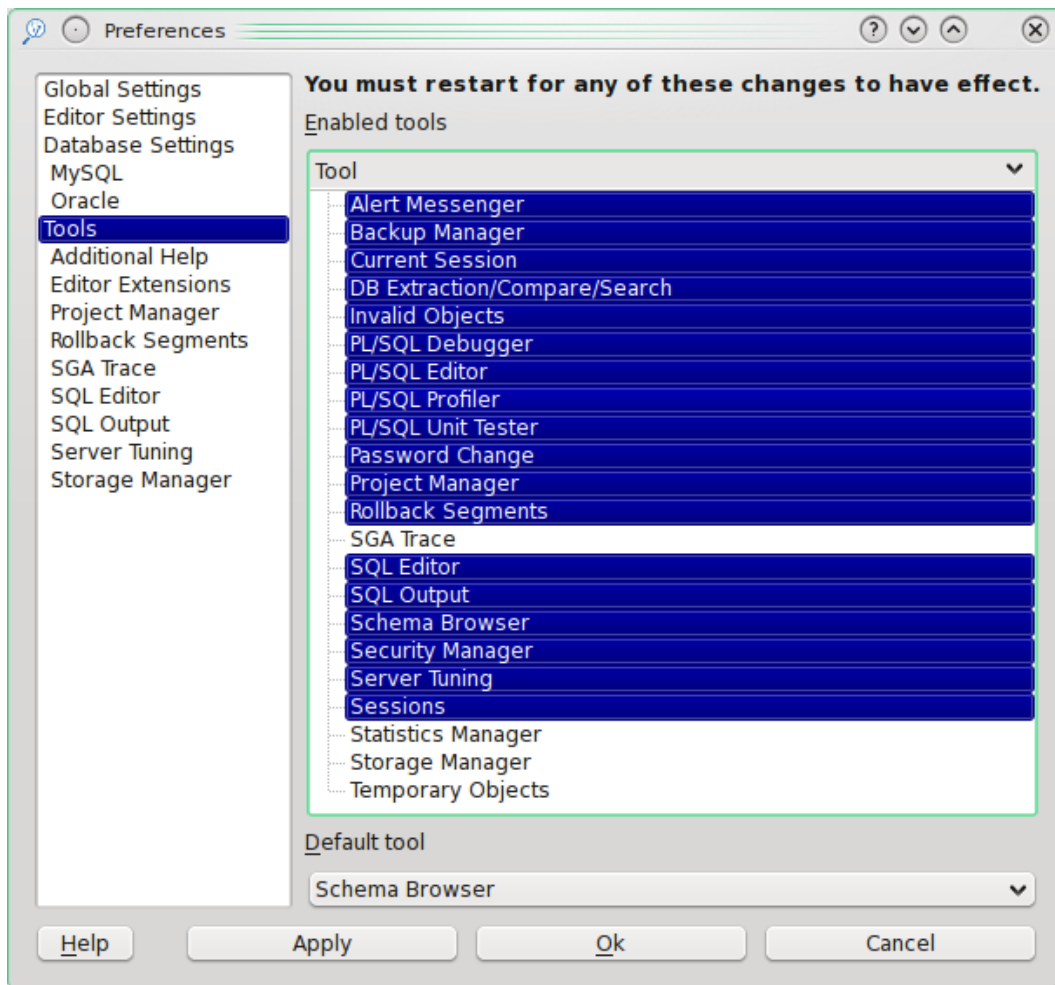
This is a quick way to add a collection of manuals. When you select this button a dialog will pop up and ask you for a filename. In Oracle 8 this should be

the index page that contains the links to the manuals. Observe that this is not the page which links to the different book categories. In Oracle 9i you should point it to the docindex.htm file in the base directory of the documentation. If successful all the manuals available on the page is added to the list of manuals.

**Add path** Add a manual to the list of manuals. This pop up a dialog asking for the name of the manual and the location of the manuals table of contents HTML file (toc.html).

### 3.8.5 Tool Settings

This page of the options contain settings on which tools should be enabled in TOra.



The first list contains all the available tools in TOra. If an item is selected in the list it is enabled. If unselected it is not enabled.

There is also a combobox at the bottom called **Default tool**. Using this setting you can specify which tool to start by default when opening a new connection. Many people are for instance more interested in the Schema browser than a worksheet and could start that instead of a worksheet by default on opening a connection.

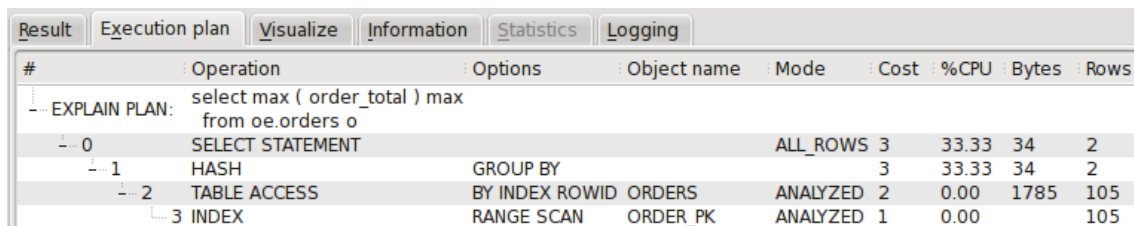
Be aware that you need to restart TOra for changes to these settings take affect.

## 3.9 Other common elements

These describe other elements that are used in several parts of the user interface.

### 3.9.1 SGA Statement

This element is used to describe an element in the SGA.



#	Operation	Options	Object name	Mode	Cost	%CPU	Bytes	Rows
-- EXPLAIN PLAN: select max ( order_total ) max from oe.orders o								
0	SELECT STATEMENT			ALL_ROWS	3	33.33	34	2
1	HASH	GROUP BY			3	33.33	34	2
2	TABLE ACCESS	BY INDEX ROWID	ORDERS	ANALYZED	2	0.00	1785	105
3	INDEX	RANGE SCAN	ORDER_PK	ANALYZED	1	0.00	105	105

The SGA statement have three tabs. The first tab called SQL simply display all of the SQL of the statement. The second tab called Execution plan displays the execution plan of the statement. This is described in more detail later. The last tab called Information simply display the information available about the statement in the SGA.

### 3.9.2 Explain plan

This element can be either part of an SGA statement or by itself in some cases (See above for a screenshot). For more information about execution paths check the Oracle manual Designing and Tuning for Performance, chapter 5 in the 8i release.

A few notes here is that the execution plan displayed here is not necessarily the same as when run depending on one of these reasons.

- It is run as a different user which doesn't access the same database objects as the user you are logged in as now.
- The session running the query can have changed some optimizer parameters compared to your session.

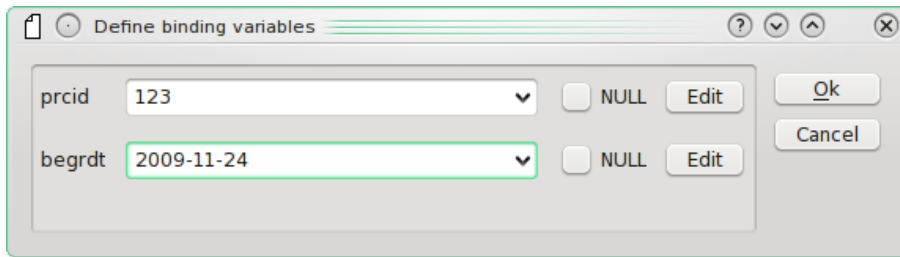
What you see in this window is the execution you would get if you ran the statement like the one you are logged in as now and in your session. One example of this problem is if a user executes SQL which accesses his own objects which doesn't have public synonyms, then another person can simply not execute the SQL without modifying it to add owner specifier to the tables which TOra will not do automatically.

Explain plan requires a table to store the result in. The name of this table can be configured in the options(see [Section 3.8 \[preferences\], page 20](#)). If the table doesn't exist TOra will ask you if it should try to create it. Without this table you can not display execution plans in TOra.

### 3.9.3 Getting bind parameters

This dialog is displayed when input parameters are required to run the SQL. For instance the example dialog is the dialog displayed when running the following SQL in the worksheet.

```
select * from essprc where prcid > :prcid and begrdt > :begrdt;
```



As you can see the field name as specified after the `:` character is presented as a title. If the same field name is used at several parts of the query you only get to specify it once. You specify the value in the editor line just to the right of the field name label. If you want to specify a **NULL** value check the **NULL** checkbox to the right of the editor. You can also edit the value in a memo editor (see [Section 3.9.5 \[memoeditor\], page 32](#)) pressing the **Edit** button to the right on the value.

The parameter editor will cache it's value so it will remember the last parameters you fed to it when you run the query again.

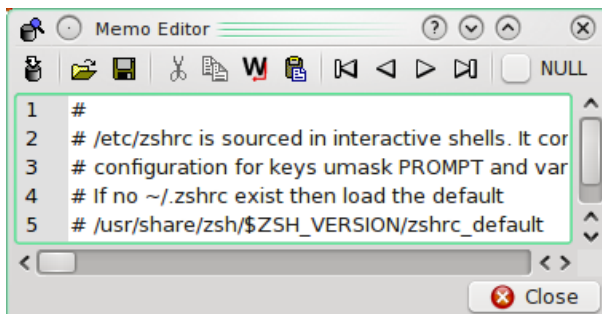
### 3.9.4 Object description

**all\_synonyms** synonym for **sys.all\_synonyms** - All synonyms accessible to the user

#	Column Name	Data Type	NULL	Comments
1	OWNER	VARCHAR2 (30)	NOT NULL	Owner of the synonym
2	SYNONYM_NAME	VARCHAR2 (30)	NOT NULL	Name of the synonym
3	TABLE_OWNER	VARCHAR2 (30)	NULL	Owner of the object referenced by the synonym
4	TABLE_NAME	VARCHAR2 (30)	NOT NULL	Name of the object referenced by the synonym
5	DB_LINK	VARCHAR2 (128)	NULL	Name of the database link referenced in a remote synonym

This is a list that describes the columns of a table or view. There is nothing much special about this view except that you can check the collected analyze statistics for the column by placing the cursor over it and reading the tooltip that will be displayed. If no tooltip is displayed there is no analyzed statistics available. Also notable is that this list is read completely from the object cache if it available. This will mean that if you are modifying the object you are modifying you need to reread the object cache before you can see the changes.

### 3.9.5 Memo editor



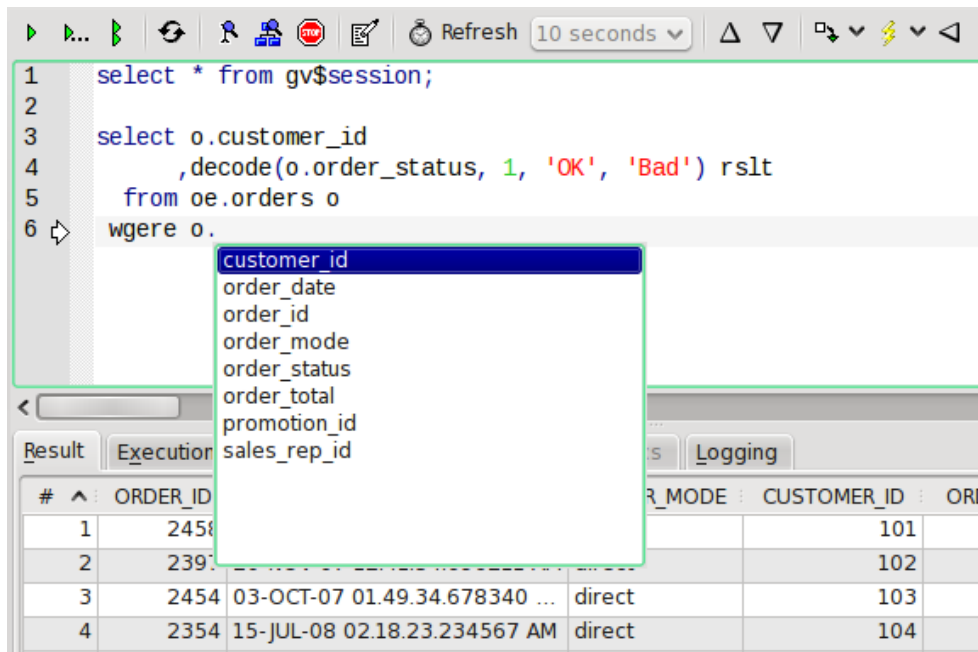
This is used to display or edit larger text than is easily entered in a oneline editor or list. To the top is a toolbar with some editing buttons. Specifically you have the save changes button to the left. You can also do this using the key **CTRL+Return**. You can remove the editor without saving changes using the **Escape** key or closing the window.

## 4 Available tools

These are the available tools in a standard TOra installation. Individual installations may vary.

### 4.1 SQL Editor

This tool provides a way to execute arbitrary SQL or PL/SQL code and also investigate how it is executed and it's resource consumption.



The worksheet is split into two panes. The upper one is the editor where you enter the queries you want to run. In the lower you can investigate the result.

**Tip** You can enter multiple queries/statements by separating them with a semicolon.

#### Toolbar & menu

There is a toolbar and a tool menu available for the worksheet. They both contain the following commands.

##### Execute Current Statement

This executes the statement you current are within or after. When executed the statement executed will be highlighted in the editor. The keyboard shortcut is **CTRL-Return** and **F9**.

##### Step through statements

By pressing this button you can execute all statements entered (and separated by semicolon). The keyboard shortcut is **F9**.

**Execute All Statements**

Execute all the statements in the editor. You get no result from the execution, the statement is simply executed. You can investigate the success of the commands in the **Logging** pane of the result. The keyboard shortcut is **F8**.

**Execute Newline Separated**

Execute a statement that is separated by two newlines (one empty line) instead of the normal ';' character. Could be useful for people who are familiar with some other Oracle tools. The keyboard shortcut is **SHIFT+F9**. Only available in the menu, not in the toolbar.

**Reexecute last statement**

This will re-execute the last executed statement. The keyboard shortcut is **F5**.

**Describe under cursor**

Describe the table currently under the cursor. The keyboard shortcut is **F4**.

**Explain plan of current statement**

Displays explain plan of current statement.

**Stop execution**

Abort execution of the currently running query. This will not work in Windows where the query will keep running in the background until the first row of the query is returned and consume server resources before it is aborted. When you execute a new statement the current execution will be aborted.

**Clear execution log**

Erase the contents of the rows under the **Logging** pane of the result.

**Gather session statistics of execution**

Enable or disable collecting statistics when running the statements. This will cause TOra to execute at least two extra queries for every statement executed when enabled. Next to this button in the toolbar is a combo box in which you can choose the refresh time for the charts in the statistics pane.

**Previous Log Entry**

Display the result of the previous entry in the log. If result caching is on in the settings the previous result is displayed, if it isn't the statement is reexecuted to generate the result. Keyboard shortcut is **ALT+Up**. Only available in toolbar, not in menu.

**Next Log Entry**

Display the result of the next entry in the log. If result caching is on in the settings the next result is displayed, if it isn't the statement is reexecuted to generate the result. Keyboard shortcut is **ALT+Down**. Only available in toolbar, not in menu.

**Insert current Saved SQL**

Popup menu containing saved SQL. Insert chosen SQL under cursor position. Keyboard shortcut is **CTRL+SHIFT+S**.

**Run current Saved SQL**

Execute selected saved SQL. For more information see saved SQL. Keyboard shortcut is **F7**. This is only available in the toolbar.

**Save last SQL**

Save the most recently executed SQL to the saved SQL list. When pressed will ask for the name to save the SQL under. If ':' is entered in this name it will separate into submenus on the colon.

**Current duration**

While a query is currently executing you can see how long the query has been running in the right of the toolbar as a slowly ticking up the seconds. This clock will not stop when the first row is returned but run until all rows are read or the query is aborted. Moving the cursor over this label will also display a tooltip containing the SQL currently executing.

**Change connection**

Change the connection this tool window should operate on.

**Using the worksheet editor**

Tora uses a ; to separate statements in the same way as SQL\*Plus. One difference is that Tora doesn't count whitespaces in any way to separate statements. You could write several statements on the same row and it wouldn't be a problem.

Another difference to SQL\*Plus is that Tora parses the SQL you write to determine if you are within a PL/SQL block. If that is the case the entire block will be executed. This feature can be enabled or disabled in the options.

```
SELECT * FROM dual; SELECT * FROM all_tables;

BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello');
END;
```

In the example above there are three statements that Tora will recognize and execute.

There are a few words that are treated as comments for compatibility with SQL\*Plus scripts. They are -, **REM** and **PROMPT**. Comments will disregard whatever is to the right on the line. Also multi line /\* ... \*/ style comments are recognised and ignored.

You can also use bind variables in the worksheet by using the normal : character. (see [Section 3.9.3 \[gettingbindparameters\], page 31](#) for more information how this works). One nice feature is that the first line retrieved from the result set is stored in the parameter cache with a bind name the same as the column name in lowercase. The example in the getting bind parameters could have been used directly after executing the line select \* from essprc and get the second row in the table by just using the defaults of the bind dialog if essprc contains a column named prcid.

You can also describe objects returned as in PL/SQL by using either **DESC** or **DESCRIBE** and an object name.

**Exploring the result**

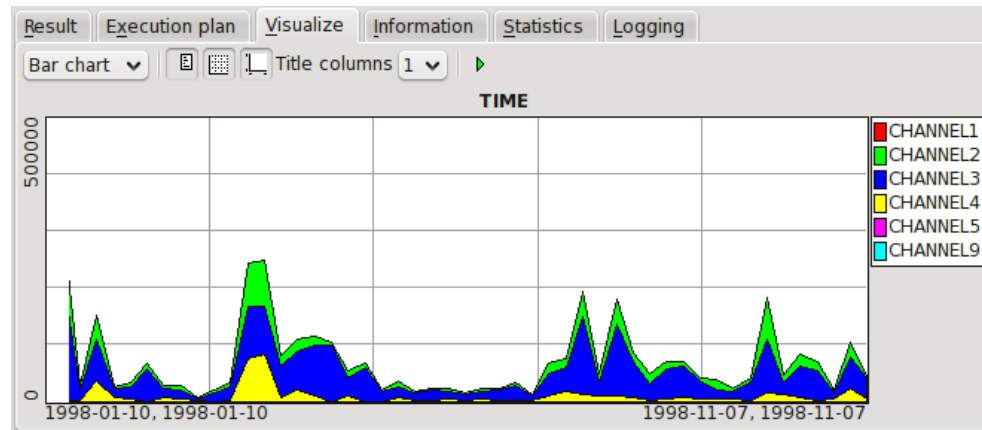
There are six different panes available with information about the executed statements.



**Result** This contains the actual result of the query or describe. For some notes about descriptions see this note.

**Execution plan** The execution plan(see [Section 3.9.2 \[explainplan\]](#), page 31) of the current statement.

**Visualize** Can be used to visualize the current content of the result. The first column is assumed to contain labels and the rest contain the data.



The toolbar available contains the following controls.

#### Chart type

What kind of chart to generate. Available types are Bar Chart, Line Chart and Pie Chart.

#### Display Legend

Display chart legend on the right of the generated chart.

#### Display Grid

Display a grid in the generated chart.

#### Display Axis Legend

Display legends of the axes of the generated chart.

#### Title columns

TODO

#### Update chart

Generate a chart of the current data in the result tab and the current settings.

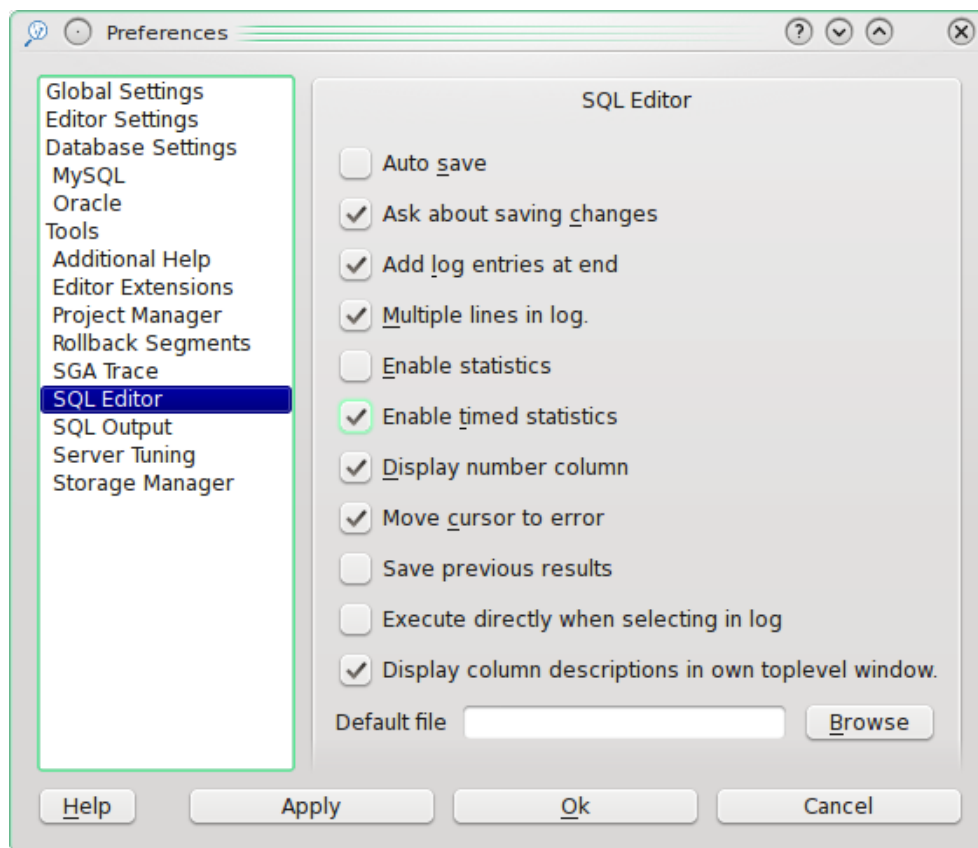
**Information** The information available about the current statement in the SGA.

**Statistics** The statistics collected when this statement was run. You need to enable statistics for this to be enabled. To update the statistic view simply change to another tab and change back to statistics. Charts are updated at the interval specified in the toolbar regardless of if you refresh statistics or not.

**Logging** This pane will display the history of the statements you have executed and their result. It can be sorted up or down depending on options. You can see the statement, it's result and the time (as recorded by the database) when it was executed. You can also see how many seconds until the first row was received. A new statement isn't added to the log until the first row of the query is returned or the statement has executed, whichever comes first.

## Preferences

There are quite a few preferences available for the SQL editor.



**Auto save** If checked the worksheet will always save changes to the editor **without asking** when the window is closed.

### Ask about saving changes

Ask about saving changes to the worksheet when it is modified. Selecting the **Auto save** have precedence over this option.

### Add log entries at end

If checked new entries in the **Logging** pane will be added to the end of list, otherwise they will be added at the top.

**Multiple lines in log**

If checked the lines in the log will display all the lines of the SQL executed. If not checked only the first line is displayed, although you can still see all of it in the tooltip or editor if you want to. For more information (see [Section 3.5 \[lists\], page 15](#)).

**Parse PL/SQL blocks**

If this is checked TOra will parse the text to determine PL/SQL blocks in the editor and execute them as one statement. If it isn't checked statements are strictly separated by ;.

**Enable statistics**

Indicates if statistics should be enabled or disabled as default when a new worksheet is opened.

**Enable timed statistics**

Set this to make TOra always enable timed statistics for the sessions that it is collecting statistics for. Observe that timed statistics will not be disabled after the worksheet is closed. The change is strictly for the current session though.

**Display number column**

Indicate whether or not to display the number column as the first column of the result or suppress it.

**Move cursor to error**

Move the cursor to the location of an error in an execute statement.

**Save previous result**

Save the results of all previous statements. This can consume a lot of memory, but will increase time to navigate history results. You can still clear this cache by erasing the log.

**Execute directly when selecting in log**

TODO.

**Default file**

A filename to open automatically when you start a new worksheet. Choose the **Browse** button to select the file in a file dialog.

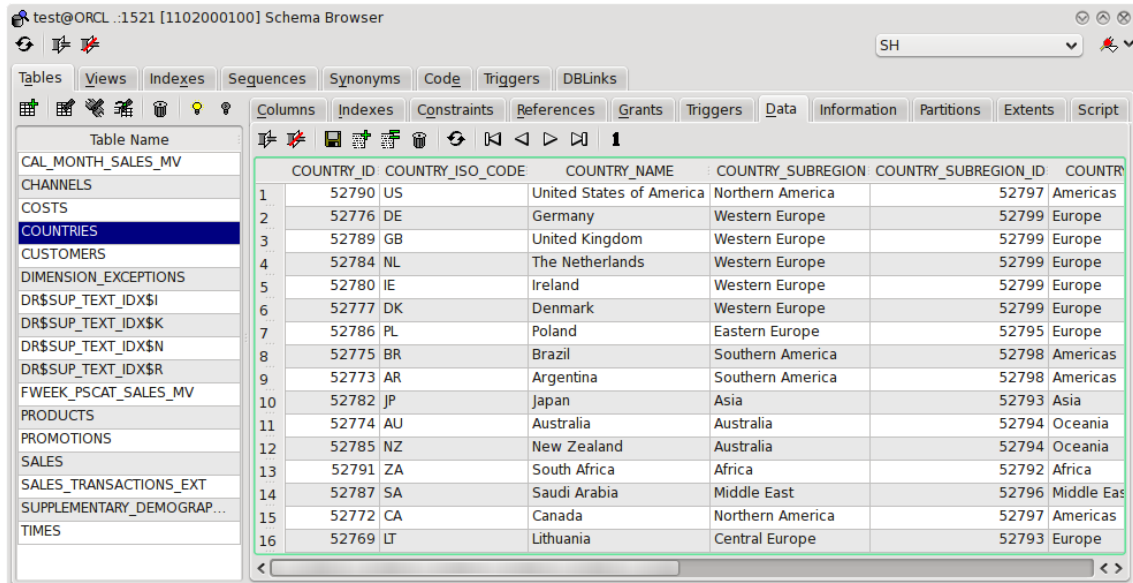
**Saved SQL**

This is a feature you can use to quickly access SQL you often use. This is a popup menu accessible in the toolbar of a worksheet. This popup will display all the SQL under "toWorksheet:" in the SQL dictionary (see [Section 4.19 \[sqledit\], page 78](#)). You can create submenus by adding a ":" character in the SQL name. The name of the SQL dictionary is what is used to generate the menu and not the description. You can still use placebinders in saved SQL. By default this list is empty.

## 4.2 Schema Browser

This tool provides you a simple browser for exploring database schema objects. Initially it can be said that the browser also provides a browser like functionality for the template help (see [Section 4.18 \[template\], page 74](#)) which is covered separately. This browser will

provide you with more information and you will also in future versions of TOra be able to interact with schema objects (Dropping or create tables etc.), this is not fully possible in the current version. The template help on the other hand is tree based and some people consider it easier and faster to work with.



## Using the browser

The schema browser has a very simple interface with a toolbar which contains the following functions. All of these are also available in the Browser menu item except for the schema selection and change object.

**Refresh** Refresh the displayed data from the database. Also available on the **F5** key.

### Define filter

Displays the define filter dialog(.). This is used to specify which objects to show in the browser. The default filter is to show everything. Also available on the **CTRL+SHIFT+G** key.

### Clear filter

Remove the current filter and revert to the default of showing everything. Also available on the **CTRL+SHIFT+H** key.

### Schema

The last item on the toolbar is a combo box where you can select the schema to display objects for. It defaults the schema you are currently logged in as. Only schemas that are visible to the current user are selectable in the list (Really, TOra does not contain the secret DBA password that will always let you access everything regardless of your privileges).

### Change connection

Change the connection this tool window should operate on.

The rest of the interface consists of two tabbed windows, one within the other. The first one you can use to select the object type you want to inspect. When you have selected a

type you can see the visible objects of the specified type available in the selected schema. The second tab indicates what kind of information you want to explore about the selected object and depends on the object type ( . ). When you select an object name in the left list you can inspect information about it to the right.

You can move the keyboard focus to the schema selection using **ALT+S** and to the current object list using **ALT+N**.

## Available information

This is the available information for the different object types.

Object type	Name	Description
<b>Tables</b>	<b>Columns</b>	A description of the columns in the table. For more information about this information also see object description.
	<b>Indexes</b>	A list with the indexes that have been created for this table.
	<b>Constraints</b>	Displays constraints associated with this table.
	<b>References</b>	Displays the foreign key constraints that references this table and the other objects that have dependencies on this table.
	<b>Grants</b>	The grants that have been made on this table.
	<b>Triggers</b>	Lists the triggers that exist for this table.
	<b>Data</b>	Displays the data in the table. This is the only part of the database browser that let you modify any data. For more information about how to use the content editor .
	<b>Information</b>	Contains information about the table available in the SGA including analysed statistics etc.
	<b>Partitions</b>	TODO.
	<b>Extents</b>	TODO.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Views</b>	<b>Columns</b>	A description of the columns in the view. For more information about this information also see object description.
	<b>SQL</b>	The SQL that defines the view.

	<b>Data</b>	Displays the data in the view. This is the only part of the database browser that let you modify any data. For more information about how to use the content editor ( ). You can not modify some views depending on how they are defined. See the Oracle reference for more information about editing views.
	<b>Grants</b>	The grants that have been made on this view.
	<b>Dependencies</b>	The objects that depend on this view.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Indexes</b>		
	<b>Columns Information</b>	The columns that the index operate on. Information in the SGA about an index. For instance analyzed statistics available for this index.
	<b>Extents</b>	TODO.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Sequences</b>		
	<b>Info</b>	Information about the sequence.
	<b>Grants</b>	The grants that have been made on this object.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Synonyms</b>		
	<b>Info</b>	Information about the synonyms
	<b>Grants</b>	The grants that have been made on this object.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Code</b>		
	<b>Declaration</b>	The declaration of a package or type.
	<b>Body</b>	The implementation of a package, body, procedure or function.
	<b>Grants</b>	The grants that have been made on this object.
	<b>Dependencies</b>	Dependencies this object has.

	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>Triggers</b>	<b>Info</b>	Information about the trigger.
	<b>Code</b>	The code that implement the trigger.
	<b>Columns</b>	The columns this trigger operates on.
	<b>Grants</b>	The grants that have been made on this object.
	<b>Dependencies</b>	Dependencies of this object.
	<b>Script</b>	This tab displays an SQL script to recreate the database object.
<b>DBLinks</b>	<b>Info</b>	Information about the trigger.
	<b>Synonyms</b>	TODO.

## Content editor

The content editor lets you edit the contents of a table or view. There are a few things to note about this.

First of all the content editor is designed to behave nicely to the database which means that it will only save the data when it thinks you are finished editing a row. This happens when you save the current row change or you commit the database. When there is unsaved data you can see it in the status bar that there exists unsaved data. Also observe that the content editor will respect the auto commit setting in the database settings.

The content editor has it's own toolbar with the following buttons.

### Define filter

Displays the define filter dialog. This is used to specify which objects to show in the browser. The default filter is to show everything.

### Clear filter

Remove the current filter and revert to the default of showing everything.

### Save changes

Save the changes made to the current row to the database. Observe that this will not commit the changes.

### Add new record

Add a new record to the current table.

### Duplicate an existing record

Create a new record copying all data from currently selected one.

### Delete current record from table

Delete the current record from the database.

### Refresh data

Requery data from the database.

**Go to first row**

Go to the first record in the editor.

**Go to previous row**

Go to the previous record in the editor.

**Go to next row**

Go to the next record in the editor.

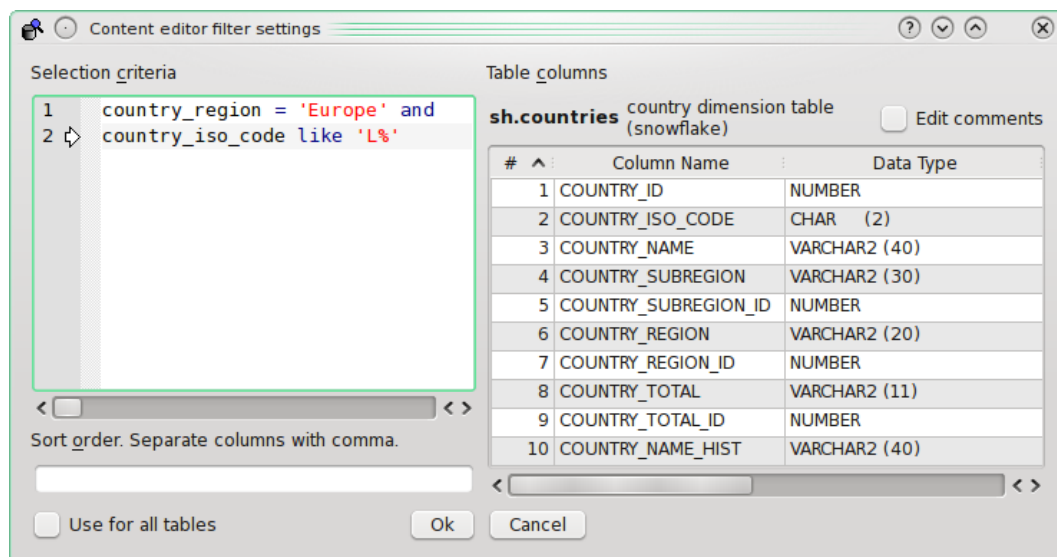
**Go to last row**

Go to the last record in the editor. Observe that this will read in all available records in the table into memory.

**Toggle between table or single record editing**

Switch between editing data as a table or single record (the later one could be useful for table with a lot of columns).

The content editor can define a filter to specify which part of a table to edit. This dialog looks like this.



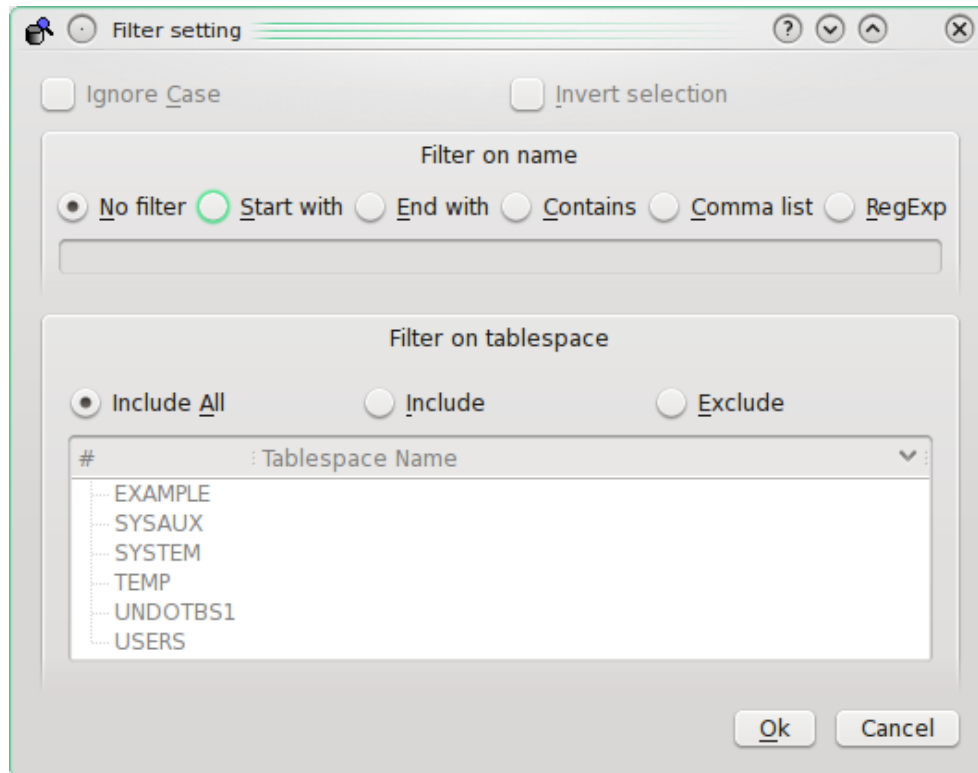
In the **selection criteria** part you can add whatever you want to go after a **where** in a select statement. In the **sort order** you can list the columns you want to sort on. The sort order only affect the order the records are read from the database, after they are read you can still sort on any column by clicking it. To the right is a list of available columns in the table.

Finally at the bottom is a checkbox called **use for all tables**. If this box is checked the index is used for all tables, if not checked you can define specific filters for each tables and the content editor will remember the different filters as you jump between tables. The reason for this is that it's quite common that not all tables have all the columns you want to filter on for a table, in this case you will get an error trying to access the table with a faulty filter setting.



## Defining a filter

Filtering is based on the object name and can be specified in a variety of ways. This is the dialog used to specify the filter.



The most important part of this dialog is the editor which lets you specify the string to apply the criteria to. This is not available for No filter setting. The different available types and options are.

**No filter** No filter specified, display all objects.

**Start with** Only include objects which name starts with the specified string.

**End with** Only include objects which name ends with the specified string.

**Contains** Only include objects which name contain the specified string.

**Comma list**  
TODO.

**RegExp** Only include objects which name matches the regular expression specified in the editor. The dialect of RegExp:s is the ones in the version of Qt that TOra is compiled against. For more information on regular expressions check out <http://doc.trolltech.com/qregexp.html>, if you are using Qt 3.0 or later you have a more advanced form of regular expressions.

**Ignore case** Ignore the case when matches. Since object names are pretty much always uppercase it is probably safest to leave this on.

**Invert selection**

If set, include all the objects that does not meet the specified criteria instead of the ones which does.

Some objects you can also search on what tablespace they are located on. This can be specified by the lower part of the dialog.

**Include all**

Include all regardless of tablespace

**Include** Include the objects located on the selected tablespaces in the list below.

**Exclude** Include the objects not located on the selected tablespaces in the list below.

### 4.3 PL/SQL Editor

This tool provides you with an advanced editor for developing PL/SQL objects.

#### Elements of the editor window

On the top of the screen is the toolbar for the editor.

Below that are two panes. The left one "Objects" contains the available objects in the selected schema. The schema is selectable through the toolbar. Select the code object you want to edit in this list or start a new object before starting to edit the code.

The right pane contains more widgets: Contents, Message Log, and the editor itself. In the Contents list you can find declared methods of packages and also variables or even anonymous blocks in your code. Selecting a line in this pane will move the cursor to where the element is in your code, for instance where a variable or method is declared.

The Message Log contains all errors and optional warnings for given PL/SQL code. Clicking the log items will take you to the exact place in the editor.

#### Toolbar & menu

The toolbar and menu contain pretty much the same commands with a few exceptions.

**Refresh object list**

Update the **Object** list to the left of the screen. This is first on the toolbar and close to the end in the menu. The keyboard shortcut for this is **F5**. Observe that this list is read from the object cache and will not be updated unless you reread the object cache as well (Available from the File menu).

**Select schema**

In the toolbar this is a combobox listing the available schemas. Changing the schema will change the objects displayed in the **Object** list. Selecting it in the menu (Close to the end) or selecting the **ALT+S** keyboard shortcut will move the keyboard focus to the toolbar combobox.

**New sheet** Start editing a new object.

**Compile** Compile the current editor. This will compile the procedure with the debug option, if you don't have the debug options you need to recompile it in order to be able to see watches. The keyboard shortcut for this is **F9**.

**Compile with Warnings**

Compile the current editor with suggested Oracle's PL/SQL Warnings (PLW). Fixing these warnings can speed up your code.

**Static Check**

Runs external static checker and displays result along with oracle errors and warnings. This button is only visible if static checker has been set up in preferences.

**Next error**

Move the cursor to the next syntax error in the current editor. For more information about syntax errors in editors see the editor section. The keyboard shortcut for this is **CTRL+N**.

**Previous error**

Move the cursor to the previous syntax error in the current editor. For more information about syntax errors in editors see the editor section. The keyboard shortcut for this is **CTRL+P**.

**Static check**

Running static check will create a temporary file containing currently open code tab. Then external program will be called with name of temporary file (with full path) given as parameter. Result from that external program should be in specific format in order to be interpreted by TOra.

TOra expects one line of output (written to stdout) to contain one message from static checker. This line should start with number of line followed by semicolon and then text of message.

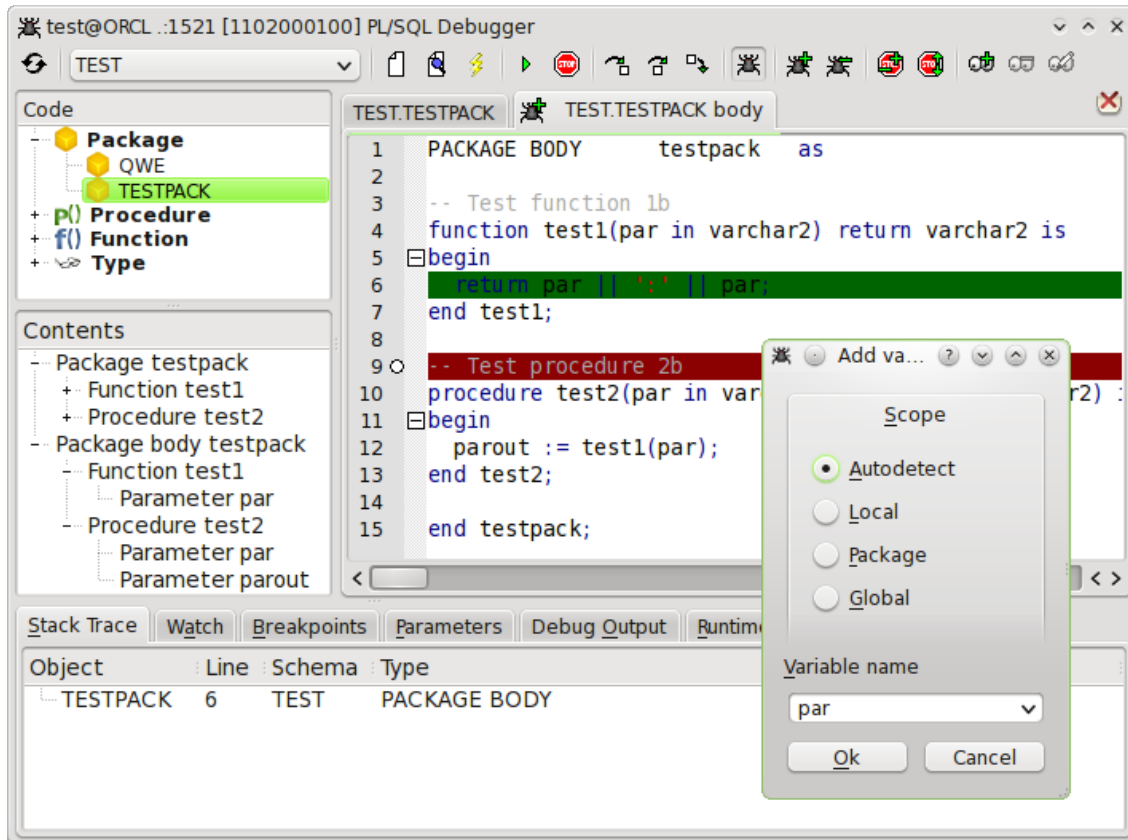
**Note:** in future TOra should be able to interpret linenummer, code of static message and then message itself. Code of static message will be used to open automatically generated URL descibing static checker message.

Example of output from static checker:

```
72:STC-0100:Comment describing function/procedure and it's parameters is missing█  
107:STC-0200:Name of procedure "my_procedure" does not comply with naming standard█
```

## 4.4 PL/SQL Debugger

This tool provides you with an advanced editor and debugger for developing PL/SQL objects.



### Elements of the debugger window

On the top of the screen is the toolbar for the debugger.

Below that to the left are two panes. The top one Objects contains the available objects in the selected schema. The schema is selectable through the toolbar. Select the code object you want to debug or edit in this list or start a new object before starting to edit the code.

The lower pane Contents contains the structure of the selected object. Here you can find declared methods of packages and also variables or even anonymous blocks in your code. Selecting a line in this pane will move the cursor to where the element is in your code, for instance where a variable or method is declared.

To the right is a large editor area where you can edit the current object. The editor is described in more detail later.

The last part of the screen is not always visible. It contains information about the currently running debugging session.

### Toolbar & menu

The toolbar and menu contain pretty much the same commands with a few exceptions.

**Refresh object list**

Update the **Object** list to the left of the screen. This is first on the toolbar and close to the end in the menu. The keyboard shortcut for this is **F5**. Observe that this list is read from the object cache and will not be updated unless you reread the object cache as well (available from the File menu).

**Select schema**

In the toolbar this is a combobox listing the available schemas. Changing the schema will change the objects displayed in the **Object** list. Selecting it in the menu (Close to the end) or selecting the **ALT+S** keyboard shortcut will move the keyboard focus to the toolbar combobox.

**New sheet** Start editing a new object.

**Scan source**

Update the **Contents** list to the left on the screen according to the source currently in the editor. TOra will try to keep up with this as you write, but new blocks and declarations will only show up if you rescan the source. The keyboard shortcut for this is **CTRL+F9**.

**Compile** Compile the current editor. This will compile the procedure with the debug option, if you don't have the debug options you need to recompile it in order to be able to see watches. The keyboard shortcut for this is **F9**.

**Execute** Call the method, procedure or function closest to the cursor. It's off course only when you edit packages that there are several methods to choose from in the current editor. If you have changed the source TOra will ask you if you want to recompile and start over or simply continue without recompiling. The keyboard shortcut for this is **CTRL+Return**.

**Stop** Halt the currently running execution. The keyboard shortcut for this is **F12**.

**Step into** Step into any functions or procedures called on the current line of execution. The keyboard shortcut for this is **F7**.

**Step over** Step to the next line of the current editor. The keyboard shortcut for this is **F8**

**Return from.**

Continue running until returned from the current function or procedure. The keyboard shortcut for this is **F6**.

**Debug pane**

Display or hide the debugging pane at the bottom of the debugging window. The keyboard shortcut for this is **F11**.

**Next error**

Move the cursor to the next syntax error in the current editor. For more information about syntax errors in editors see the editor section. The keyboard shortcut for this is **CTRL+N**.

**Previous error**

Move the cursor to the previous syntax error in the current editor. For more information about syntax errors in editors see the editor section. The keyboard shortcut for this is **CTRL+P**.

**Toggle breakpoint**

Add or remove a breakpoint on the current line of the editor. Breakpoints are indicated with a small stop sign to the right of the line in the editor. The keyboard shortcut for this is **CTRL+F5**.

**Disable breakpoint**

Will disable or enable a breakpoint on the current editor. A disabled breakpoint will be seen is grayed out in the margin of the editor of the line it is on. The keyboard shortcut for this is **CTRL+F6**.

**Add watch**

Add a watched variable. See here for more information about watches. The keyboard shortcut for this is **F4**.

**Delete watch**

Remove a watched variable. The keyboard shortcut for this is **CTRL+Del**.

**Change watch**

Change the value of a watch. See here for more information about watches. The keyboard shortcut for this is **CTRL+F4**.

**Erase runtime log**

Remove the content of the runtime log which is in the rightmost pane of the debugger info.

**Debugger editor**

When you run a program and make a change, but still decide to keep stepping through your program without restarting you can have one other changed file plus the current file, if you try to step to another object TOra will ask if you want to continue until you are back in the edited files or recompile.

Another thing special in this editor is that the PL/SQL editor will show you errors in the package. The errors are highlighted with a special background color (The default is red, but this is configurable in the options(see [Section 3.8 \[preferences\], page 20](#)). You can also use the **next error** or **previous error** toolbar button/menu entries to step between the errors.

Next thing to know about the debugger is breakpoints. You can set a breakpoint either by selecting the **toggle breakpoint** toolbar button or menu entry. This will add a breakpoint to the line you are currently on in the editor. To remove the breakpoint simply select **toggle breakpoint** again. Another way to add or remove breakpoints is to double-click in the left margin of the editor where you also see where the breakpoints are. They are indicated by a small stop sign in the margin. If you want to temporarily disable a breakpoint that you might need to enable again later you can also choose **disable breakpoint**, this is also a toggle so selecting the command again will re-enable the breakpoint. For those of you who don't know breakpoints means that when execution of the object comes to the line containing the breakpoint the debugger will stop and allow you to inspect the data.

Which brings us to the last special thing in this editor which is the current line of execution, this is an indication of where the execution currently is at. This line is indicated by a special background (Default is green, but that is configurable in the options(see [Section 3.8 \[preferences\], page 20](#))).

## Debugger information

Information about the current state of execution is shown in the pane at the bottom of the debugger window which you can be hidden. You hide or show this by selecting the command **debug pane** command or by pressing the **F11** key. The debug pane consist of several tabs containing the following information.

### Stack trace

Contains the current stack trace. This means the list of called objects that have lead to the current line of execution. Selecting a line will bring up the object and line where the the call to the next object was made except for the last line which is the current line of execution.

**Watches** Watches is where you can inspect variables in your PL/SQL code while running. They are explained in more detail later.

### Breakpoints

A list of where you current have your breakpoints and their status. The status can be **enabled** or **disabled** which are pretty self explanatory. It can also be **deferred** which means that some error have occurred setting the breakpoint. Every time execution is restarted TOra tries to set any breakpoints that are **deferred**.

### Parameters

This displays the input and output parameters that you passed to the original call that started the debugging. The output parameters are off course not available until the execution is finished.

### Debug output

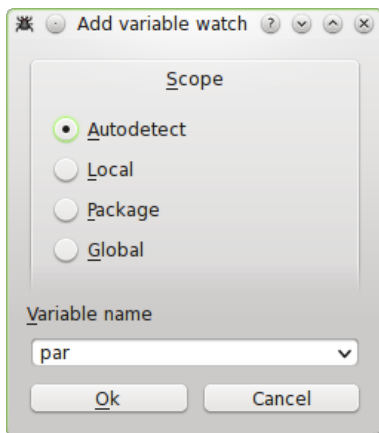
Any DBMS\_OUTPUT output from your debugging session will end up in this window. For more information about using this see the output tool.

### Runtime log

This is simply a log displaying what is happening in the target debugging thread and probably not of much interest to anybody except TOra developers. One exception is that if you somehow get an SQL error when you function or procedure of choice is executed those errors will end up here.

## Watches

Watches are a way to inspect the contents of variables in a running program. When you add a watch using the **add watch** command you are presented with the following dialog.



First of all you need to select the scope the variable is declared in. The **local** scope is variables only declared in the current context, like for instance this function or procedure. The **head** and **body** are variables declared in packages. Variables in packages are usually also available in the global scope. When you declare a watch in any scope except for **local** you should also indicate the object in which the variable is declared in the format `schema.object.variable`. The current object is prepended to the variable name when you select anything but the **local** scope.

Finally you have to fill out the variable name. If you have the cursor on a variable in the editor that name is the default when you add the watch.

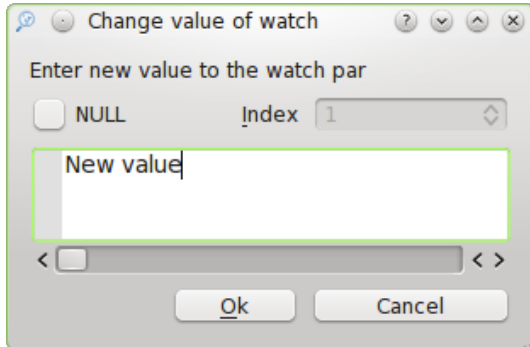
You can then see the contents of this variable in the **watch** pane of the debugger info. The value will be updated every time execution stops while you are debugging. If the variable can not be found it will be **{Unavailable}**. If all variables are unavailable even though they shouldn't you have probably forgotten to compile the object with the debug option. You can remedy this by simply using the **compile** command. TOra will always compile programs using the debug option when in the PL/SQL debugger.

TOra can also inspect list and table object which will show up as child items in the list to the variable. When this is a case you can also see how many items an array is having in the parent of the actual data items.

You can remove a watch by selecting it in the watch list and selecting the **remove watch** command.



You can also change the content of a watch. First you select the watch you want to change in the watch list, then you select the **change watch** command. You will be shown this dialog.



If you have selected the parent of an array watch you will be able to select the index you want to assign a new value with the **index** indicator to right. Then you enter the new value in the text field or check the **NULL** indicator to set the value to NULL.

## 4.5 PL/SQL Unit Tester

This tool allows you to run procedures, functions, and package members independently. Selecting proper PL/SQL unit will result in anonymous PL/SQL block containing given unit and an environment with setting variables and required outputs with unit results (via DBMS\_OUTPUT).

Automatically created code blocks can be modified by user.

What is the Unit Test Script? The anonymous PL/SQL block with the following structure:

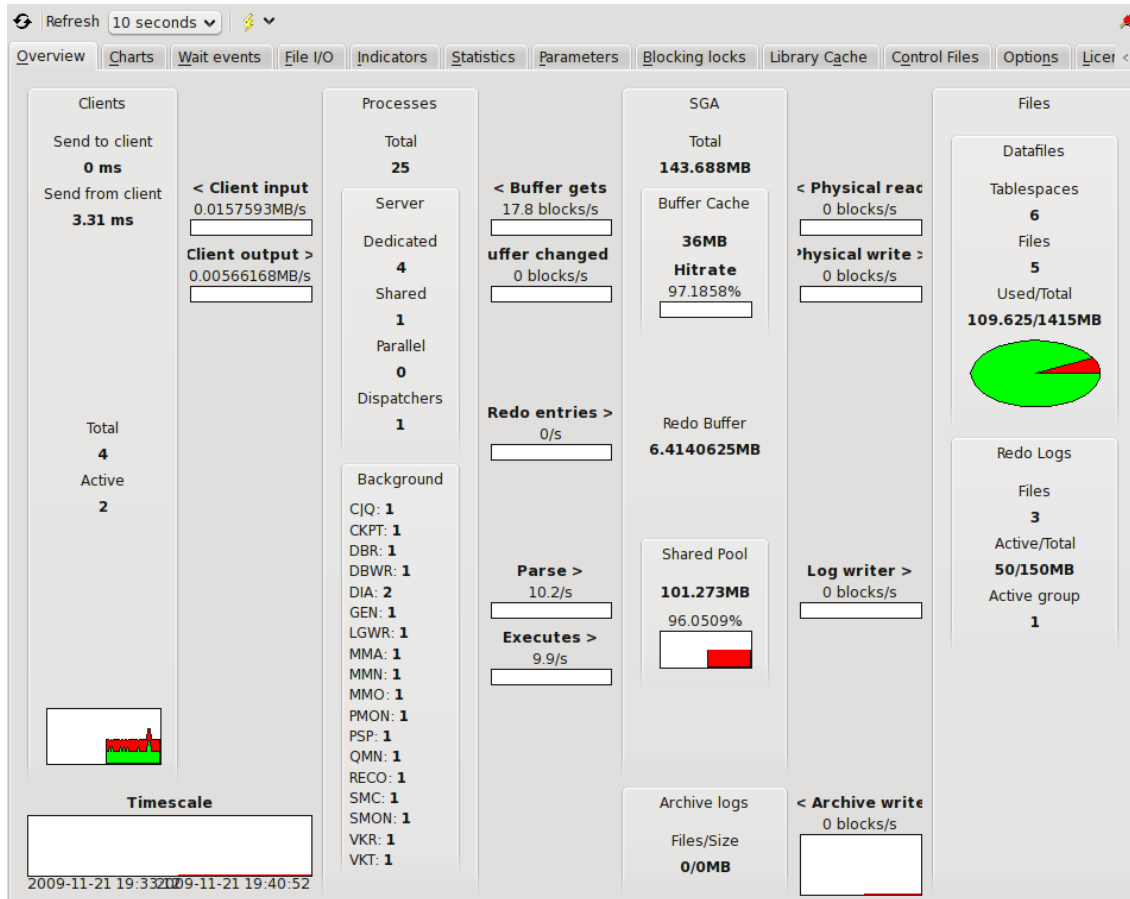
```

DECLARE
    -- declarations of IN/OUT parameters
    foo number;
    bar varchar2(22);
BEGIN
    -- definitions of parameters
    foo := 1;
    bar := 2;
    -- calling requested unit
    MYOWN.COOLPACKAGE.FOobar(
        foo => foo,
        bar => bar
    );
    -- printing of the results
    sys.dbms_output.put_line('foo: ' || foo);
    sys.dbms_output.put_line('bar: ' || bar);
END;
```

It can handle only basic data types currently. No PLSQLdata (record of etc.) supported yet.

## 4.6 Server Tuning

This tool provides easy access to important server indicators as well as server statistics and an editor for database parameters.



The interface for this tool is very simple. First there is a toolbar with the following controls.

**Refresh** Refresh data from the database, will not read another value for charts.

### Refresh rate

The interval between which to auto update data, observe that setting this too low could put a high load on a database server. This is also the sample interval for charts.

### Change connection

Change the connection to monitor in the tuning tool.

There are several tabs in the window.

**Overview** Displays an overview of database statistics, shows the flow of data through the server in an intuitive way.

**Charts** Charts containing important performance information from the database.

**Wait events**

Information and charts about wait events.

**File I/O**

Displays information about I/O separated by tablespace or datafile. You can select if you want to see average wait time, blocks for tablespace or datafile by using the combobox at the top of the charts.

**Indicators**

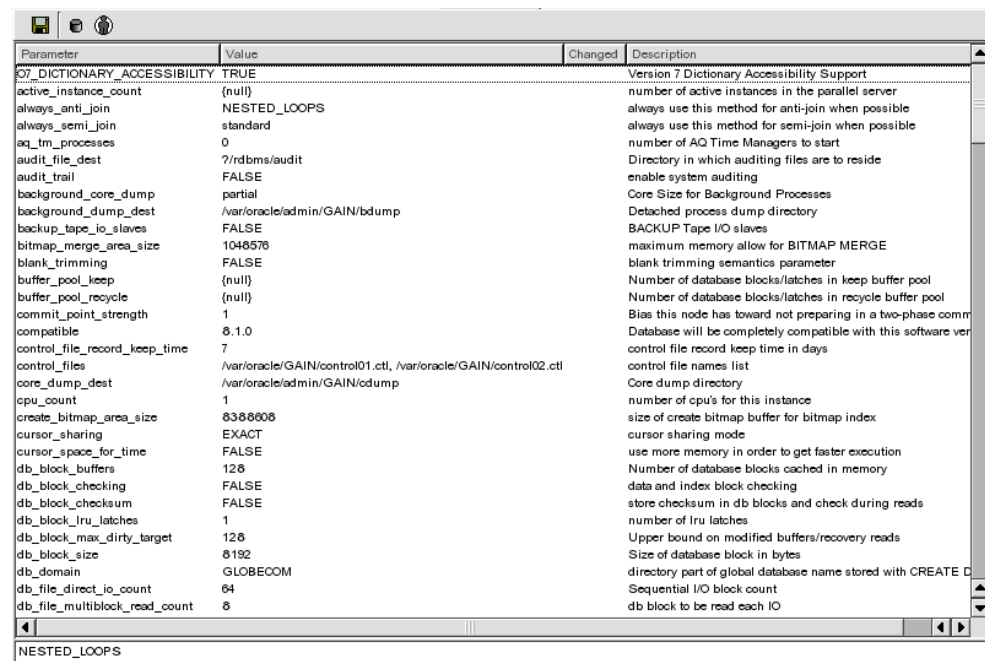
Important performance indicators.

**Statistics**

Raw statistic information from the server. The statistics is shown in two columns, the first one contain the actual value, the second show the change in the value since the last update.

**Parameters**

This tab displays the values of the parameters of the database. You can also use it to change the values by selecting a row and entering a new value at the bottom of the tab. After changes has been made you can apply or export these changes using the buttons in the toolbar at the top of the parameter editor.



Parameter	Value	Changed	Description
07_DICTIONARY_ACCESSIBILITY	TRUE		Version 7 Dictionary Accessibility Support
active_instance_count	{null}		number of active instances in the parallel server
always_anti_join	NESTED_LOOPS		always use this method for anti-join when possible
always_semi_join	standard		always use this method for semi-join when possible
aq_tm_processes	0		number of AQ Time Managers to start
audit_file_dest	?/rdbsms/audit		Directory in which auditing files are to reside
audit_trail	FALSE		enable system auditing
background_core_dump	partial		Core Size for Background Processes
background_dump_dest	/var/oracle/admin/GAIN/bdump		Detached process dump directory
backup_tape_io_slaves	FALSE		BACKUP Tape I/O slaves
bitmap_merge_area_size	1048576		maximum memory allow for BITMAP MERGE
blank_trimming	FALSE		blank trimming semantics parameter
buffer_pool_keep	{null}		Number of database blocks/latches in keep buffer pool
buffer_pool_recycle	{null}		Number of database blocks/latches in recycle buffer pool
commit_point_strength	1		Bias this node has toward not preparing in a two-phase commit
compatible	8.1.0		Database will be completely compatible with this software version
control_file_record_keep_time	7		control file record keep time in days
control_files	/var/oracle/GAIN/control01.ctl, /var/oracle/GAIN/control02.ctl		control file names list
core_dump_dest	/var/oracle/admin/GAIN/cdump		Core dump directory
cpu_count	1		number of cpus for this instance
create_bitmap_area_size	8386608		size of create bitmap buffer for bitmap index
cursor_sharing	EXACT		cursor sharing mode
cursor_space_for_time	FALSE		use more memory in order to get faster execution
db_block_buffers	128		Number of database blocks cached in memory
db_block_checking	FALSE		data and index block checking
db_block_checksum	FALSE		store checksum in db blocks and check during reads
db_block_lru_latches	1		number of lru latches
db_block_max_dirty_target	128		Upper bound on modified buffers/recovery reads
db_block_size	8192		Size of database block in bytes
db_domain	GLOBECOM		directory part of global database name stored with CREATE DATABASE
db_file_direct_io_count	64		Sequential I/O block count
db_file_multiblock_read_count	8		db block to be read each IO

**Display hidden parameters**

TODO

**Generate configuration file**

Generate a p-file with the current changes (And original changes as well) into a memo editor(see [Section 3.9.5 \[memoeditor\]](#), page 32).

**Apply changes to system**

Apply changes made to the system.

**Apply changes to session**

Apply changes to the current session. Will only apply changes to the main connection and not any child sessions TOra might open for instance when statements are executed in the worksheet.

**Drop changes**

Drop the changes currently made to the parameters and revert to the current session settings.

**Blocking locks**

TODO.

**Library cache**

TODO.

**Control files**

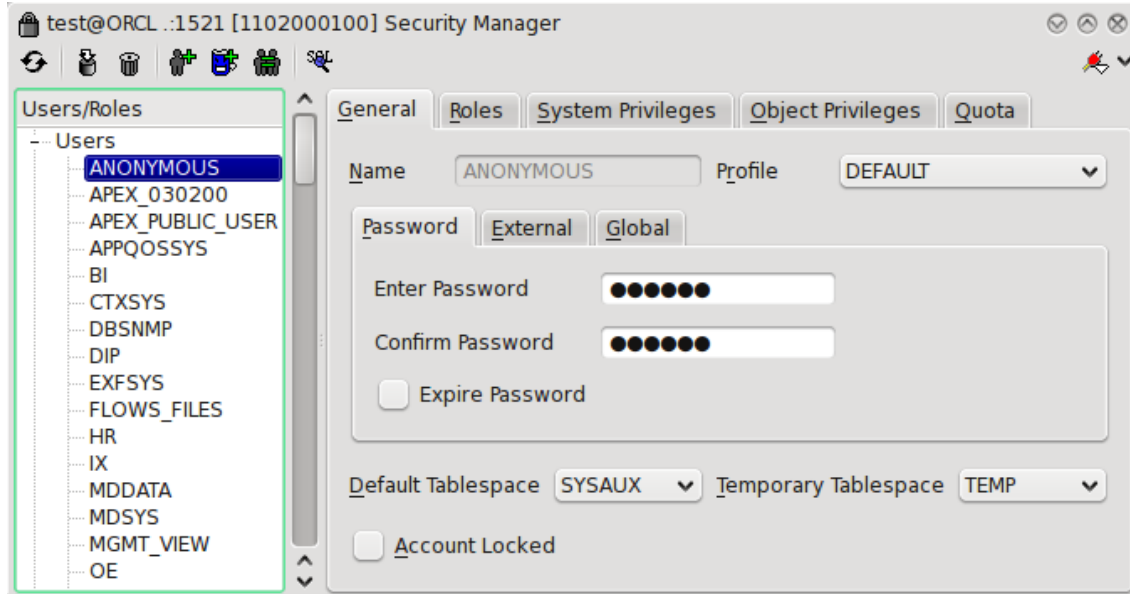
TODO.

**Options** List the options available in the database server.

**Licenses** Display license information to the current database.

## 4.7 Security Manager

This tool provides an easy interface to manipulate users, roles and their privileges.



The window consists of three parts. First the toolbar on top, to the left is a list of all the users and roles. Finally to the left is a tabbed window containing the different settings and privileges you can manipulate for the chosen user.

### Toolbar

The commands in the toolbar are from left to right.

**Refresh** Update the user and role list to the left of the screen.

**Save changes**

Perform any changes made to the current user or role.

**Remove** Drop the user or role currently selected. If the user still owns objects a question will be asked about also removing these objects or abort the drop.

**New user** Start defining a new user.

**New role** Start defining a new role.

**Copy** Copy the current user or role to a new name.

**SQL** Display the SQL that will be used to apply the current changes in a memo editor(see [Section 3.9.5 \[memoeditor\]](#), page 32).

**Change connection**

Change the connection to administrate security for.

You select the role you want to work on simply by pressing it in the left pane or any of the new user, new role or copy buttons to start defining a new one.

## Defining a user

There are five tabs available when defining a user or four when doing the same for a role.

**General** Used for setting the name, authentication, default tablespaces etc. for the new user. For more information about this see the Oracle SQL Reference.

**Roles** Which roles are granted to the user or role. For more information about defining privileges see the section below.

**System privileges**

Which system privileges this user or role should have. For more information about defining privileges see the section below.

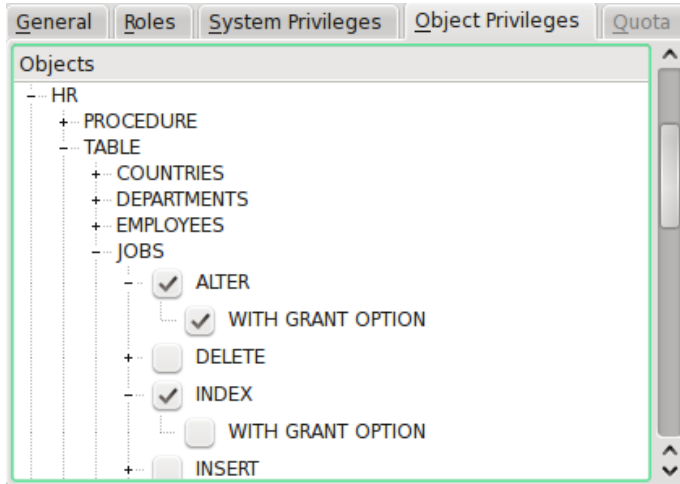
**Object privileges**

Which privileges this user or role should have on specific schema objects. For more information about defining privileges see the section below.

**Quota** Define the quota for the user, this tab is not available for roles. For more information about defining quotas see the section below.

## Defining privileges

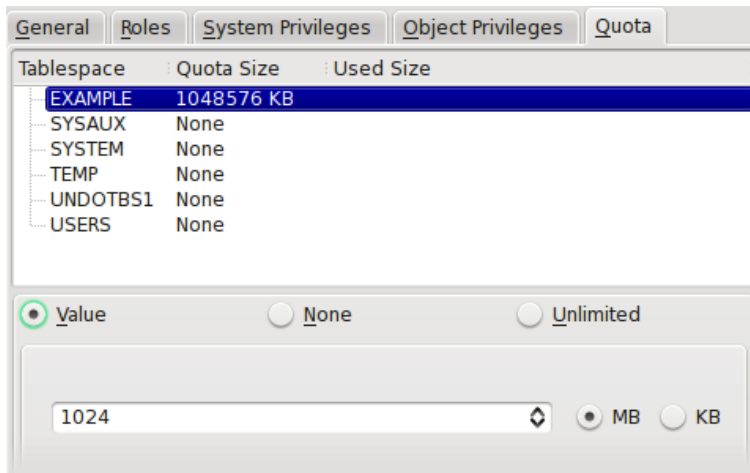
All roles, system or object privileges are defined using basically the same interface.



This list contains a tree view with first the schema, then the type, the object and finally the actual privilege to define. In the system or role tabs, the privileges are in the root of the list. If you open the privilege you will find an additional **with grant option** item which if checked will give the user or role the ability to grant the privilege on to other users or roles. In the role case you also have another child item to the privilege which is **default**. If **default** is checked the role will be enabled by default when the user logs in. When you start a new user all checked items will be made visible by expanding the parent items.

## Defining quota

You define quotas using the following interface.



To change a quota simply select the tablespace and specify the new quota using the radio buttons at the end of the screen. If **value** is selected you can specify a value using the size control at the bottom.

## 4.8 Storage Manager

This tool provides an easy interface to administrate tablespaces and datafiles.

Name	Status	Information	Contents	Logging	Size (MB)	Free (MB)	Autoexten
USERS	ONLINE	LOCAL	PERMANENT	LOGGING	5	.69	32767.98
/ora/base/oradata/orcl/users01.dbf	ONLINE	READ WRITE			5	.69	32767.98
UNDOTBS1	ONLINE	LOCAL	UNDO	LOGGING	60	50.38	32767.98
/ora/base/oradata/orcl/undotbs01.dbf	ONLINE	READ WRITE			60	50.38	32767.98
TEMP	ONLINE	LOCAL	TEMPORARY	NOLOGGING	93	92	32767.98
SYSTEM	ONLINE	LOCAL	PERMANENT	LOGGING	680	6.38	32767.98
/ora/base/oradata/orcl/system01.dbf	SYSTEM	READ WRITE			680	6.38	32767.98
SYSAUX	ONLINE	LOCAL	PERMANENT	LOGGING	580	36.19	32767.98
EXAMPLE	ONLINE	LOCAL	PERMANENT	NOLOGGING	100	21.56	32767.98

Owner	Segment	Partition	Extents	Blocks
1 SYS	_SYSSMU3_1204390606\$		5	40
2 SYS	_SYSSMU10_4131489474\$		3	144
3 SYS	_SYSSMU1_592353410\$		6	48
4 SYS	_SYSSMU6_2897970769\$		6	48
5 SYS	_SYSSMU5_538557934\$		6	48
6 SYS	_SYSSMU2_967517682\$		5	40
7 SYS	_SYSSMU8_3901294357\$		6	48
8 SYS	_SYSSMU4_1003442803\$		3	144

As you can see in the screenshot above the tool is a list of all the tablespaces available in the database. The tablespaces can be expanded to display the datafiles they consist of. There is also one option available to this tool, whether to collect information about how coalesced the tablespaces are, the coalesced fetching can be very slow on fragmented databases and if TOra seems to lock itself when you try to start this tool try resetting that option to not show coalesced info. Using the toolbar you can perform these commands.

**Refresh** Update the view.

**Show extent view**  
TODO.

**Show tablespaces or just datafiles**  
TODO.

**Take tablespace online**  
Take the selected tablespace online.

**Take tablespace offline**  
Take the selected tablespace off-line.

**Set tablespace default to logging**  
Enable logging for the selected tablespace.

**Set tablespace default to no logging**  
Disable logging for the selected tablespace.

**Allow read write access to tablespace**  
Enable read and write access to the selected tablespace.

**Set tablespace read only**

Only allow read only access to the selected tablespace.

**Modify tablespace**

Modify the selected tablespace settings. If you don't understand the settings in this dialog please refer to the Oracle Server Concepts manual.

**Drop tablespace**

TODO.

**Modify file**

Modify the current datafiles settings. If you don't understand the settings in this dialog please refer to the Oracle Server Concepts manual.

**Create new tablespace**

Create a new tablespace. If you don't understand the settings in this dialog please refer to the Oracle Server Concepts manual.

**Add datafile to tablespace**

Create a new datafile for the currently selected tablespace. If you don't understand the settings in this dialog please refer to the Oracle Server Concepts manual.

**Coalesce tablespace**

Coalesced free data in the current tablespace.

**Move datafile**

Move a datafile to another location.

**Change connection**

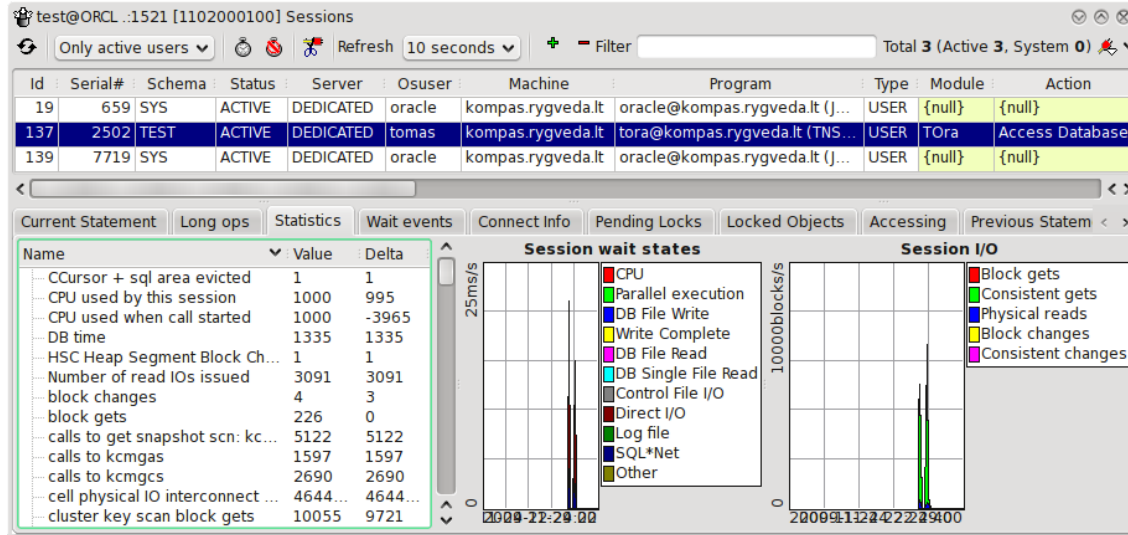
Change the connection to administrate storage for.

When you select a tablespace or datafile the lower part of this tool displays the objects (to the left) and extents (to the right) in that tablespace or file. A used extent is displayed as green in the extent view if the object it belongs to is not selected in the list to the right. Selecting an item in the list to the right will paint it's extents red in the right extent view. Unallocated space is colored whie. Some information about the number of files and extents are displayed at the top of the extents display. Several datafiles are separated by one line of black in the extent view if a tablespace containing several files are viewed.



## 4.9 Session Manager

This tool is used to manage the connections made to the database.



The session tool consists of a toolbar, a list displaying the connections currently open to the database and at the bottom a tabbed pane where you can investigate a specific connection to the database.

### Toolbar

The toolbar contains the following commands.

- Refresh** Refresh the view from the database.
- Filter** Here you can define which sessions to display. You can choose to see only sessions of specific users, only active sessions, all sessions, all sessions except background or all sessions except system ones.

### Enable timed statistics

Enable timed statistics for the currently selected session.

### Disable timed statistics

Disable timed statistics for the currently selected session.

### Disconnect session

Disconnect the currently selected session. You will be asked if you allow the current transaction to finish before kicking the session out or not.

### Refresh time

A list where you can select the interval between automatic updates.

### Select all sessions

TODO

### Deselect all sessions

TODO

**Filter**      TODO

**Change connection**

Change the connection to administrate storage for.

## Inspecting a session

When you select a session the following information is available in the tabbed pane at the bottom of the window.

**Current statement**

The statement that is currently being executed (see [Section 3.9.1 \[sgastatement\]](#), [page 31](#) for information about the statement).

**Long ops**    TODO

**Statistics**    The session statistics available for the session. You will get more statistics if you enable timed statistics through the toolbar. There are also two charts displaying waitstates and I/O for the selected session.

**Wait events**

TODO

**Connection info**

Information about the connection and versions of the components used.

**Pending locks**

What locks this session is currently waiting for.

**Locked objects**

The objects currently locked by this session.

**Accessing**    TODO

**Previous statement**

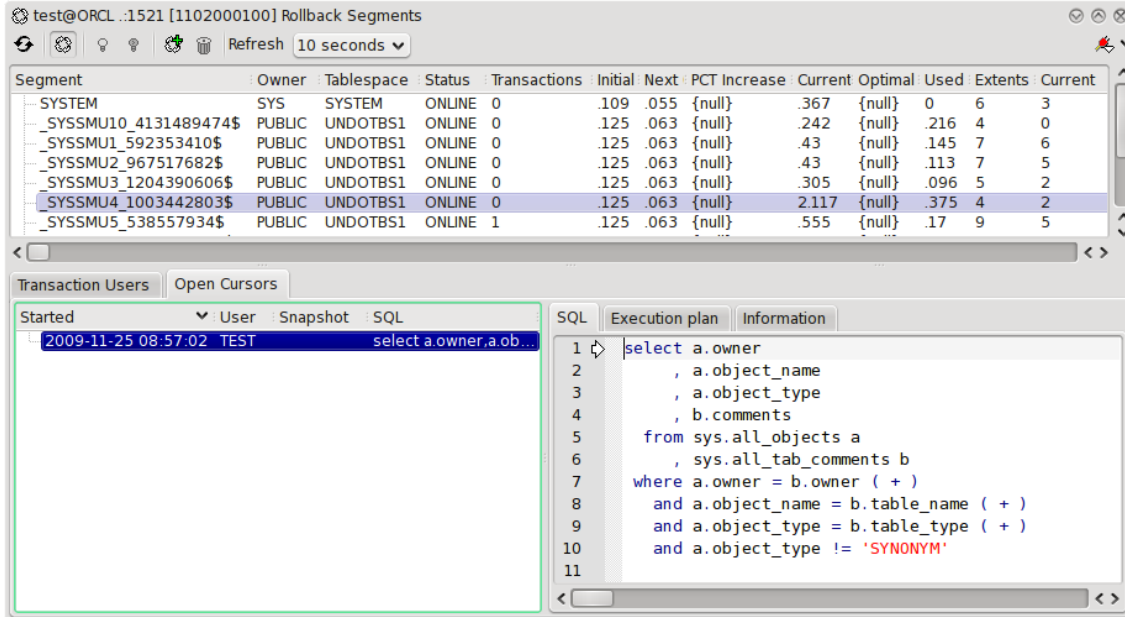
The next to last statement being executed by the session (see [Section 3.9.1 \[sgastatement\]](#), [page 31](#) for information about the statement).

**Open cursors**

This contains a list of all the cursors currently held open by the session. When a statement is selected in the list you can investigate it using the normal SGA statement view.

## 4.10 Rollback Segments

This tool is used to manage rollback segments in the database.



The window consist of three part. The top one displays the available rollback segments and you can also see running transaction that use segment in the transactions column of the view, the length of the **transactions** show up as bars in this column. One bar for each running transaction using the segment.

The last part is a unique tool to detect snapshot too old problems in your database. This is discussed more later.

### Toolbar

The toolbar contains the following commands.

#### Update segment list

Update the view from the database.

#### Enable snapshot too old detection

TODO

#### Take segment online

Take currently selected segment online.

#### Take segment offline

Take currently selected segment off-line.

#### Create new rollback segment

Create new rollback segment. If you are confused about what the items in the dialog mean refer to the Oracle Server Concepts manual as well as the Oracle SQL Reference.

#### Drop segment

Drop the currently selected segment.

**Refresh time**

A list where you can select the interval between automatic updates.

**Change connection**

Change the connection to administrate storage for.

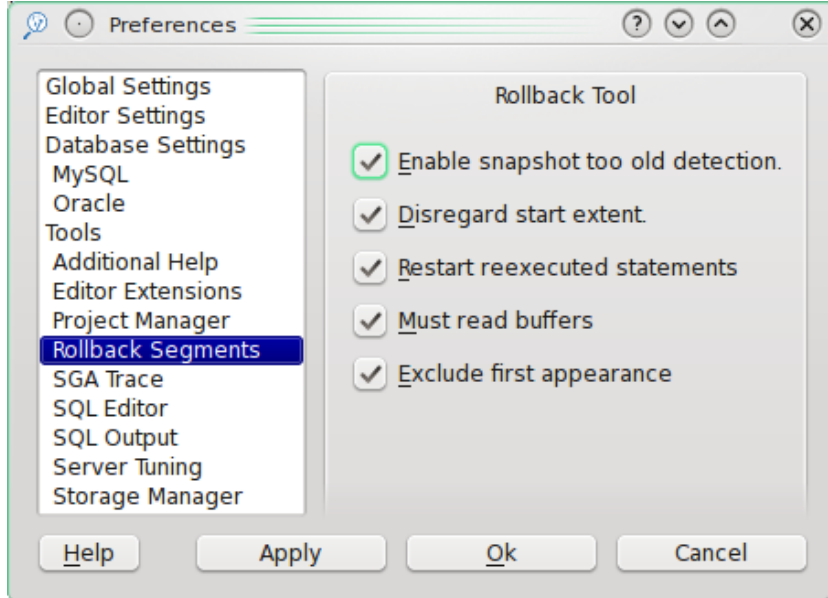
**Snapshot too old detection**

The lower part of this view is used to heuristically detect snapshot too old problems. The operation of this is that whenever the view is updated all SGA statements are cataloged and the corresponding current position in all the rollback segments are also stored the first time a new statement is detected. Since the information about the current location of the rollback when a statement was executed is not available in the SGA you have to keep the tool window open during the entire duration of the execution for this to work.

In the view you can see when the statement was first detected and who is running it. You also see the snapshot info which will display how far each of the available rollback segments have moved since this statement was executed. This is represented by one bar for each rollback segment. If any of these bars goes all the way through this column you are likely to encounter a snapshot too old statement.

Finally you can see the SQL for the row. Selecting a line will also let you investigate the statement using the normal SGA statement(see [Section 3.9.1 \[sgastatement\], page 31](#)) view.

There are a few options to configure how the snapshot too old detection is to work.

**Enable snapshot too old detection**

TODO

**Disregard start extent**

If you check this the bars will always start from the left of the column. Otherwise the bars will start corresponding to the current extent of the tablespace and then wrap around as they grow.

**Restart reexecuted statements**

If the execution count is changed for a statement reset the location of the rollback segments to the current location.

**Must read buffers**

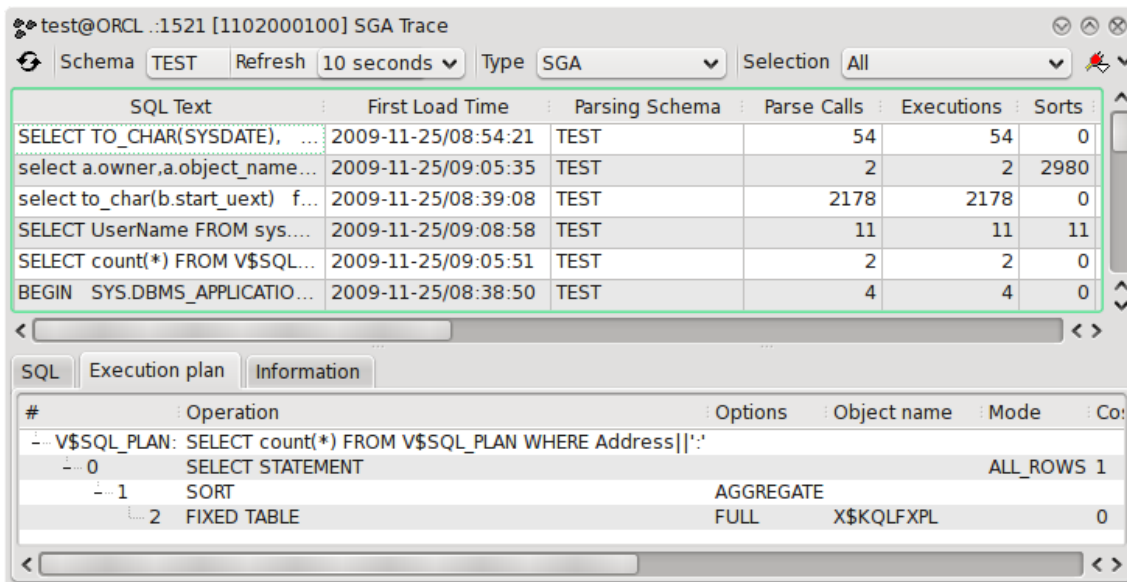
To display a statement it must be reading buffers, it is still kept in memory if no buffers are read, but not displayed in the list until buffers are read again. Check this to remove any statements that you execute and simply kept open even though you will not be reading from them any more.

**Exclude first appearance**

Exclude the first update that would show the statement. Without this checked any statement with execution changed and buffers read will show up.

## 4.11 SGA Trace

This tool is used investigate the statements available in the SGA.



The screenshot shows the SGA Trace tool interface. At the top, there is a toolbar with controls for Schema (TEST), Refresh (10 seconds), Type (SGA), and Selection (All). Below the toolbar is a table listing SQL statements with columns for SQL Text, First Load Time, Parsing Schema, Parse Calls, Executions, and Sorts.

SQL Text	First Load Time	Parsing Schema	Parse Calls	Executions	Sorts
SELECT TO_CHAR(SYSDATE), ...	2009-11-25/08:54:21	TEST	54	54	0
select a.owner,a.object_name...	2009-11-25/09:05:35	TEST	2	2	2980
select to_char(b.start_uext) f...	2009-11-25/08:39:08	TEST	2178	2178	0
SELECT UserName FROM sys....	2009-11-25/09:08:58	TEST	11	11	11
SELECT count(*) FROM V\$SQL...	2009-11-25/09:05:51	TEST	2	2	0
BEGIN SYS.DBMS_APPLICATION...	2009-11-25/08:38:50	TEST	4	4	0

Below the table, there are tabs for SQL, Execution plan, and Information. The Information tab is selected, showing a detailed execution plan for the statement 'SELECT count(\*) FROM V\$SQL\_PLAN WHERE Address||':'

#	Operation	Options	Object name	Mode	Co:
0	SELECT STATEMENT			ALL_ROWS	1
1	SORT	AGGREGATE			
2	FIXED TABLE	FULL	X\$KQLFXPL		0

The session tool consists of a toolbar, a list of SGA statements ( see [Section 3.9.1 \[sgastatement\]](#), page 31 ) and finally at the bottom a SGA statement view.

### Toolbar

The toolbar contain the following controls.

**Update** Update the view from the database.

**Schema** The user schema to extract statements for.

**Refresh time**

A list where you can select the interval between automatic updates.

**Type** What type of statements you want to examine. Can either be SGA to investigate all entries of the SGA or Long operations to check long operations in the

database (Databases must have timed statistics enabled to generate entries for long operations).

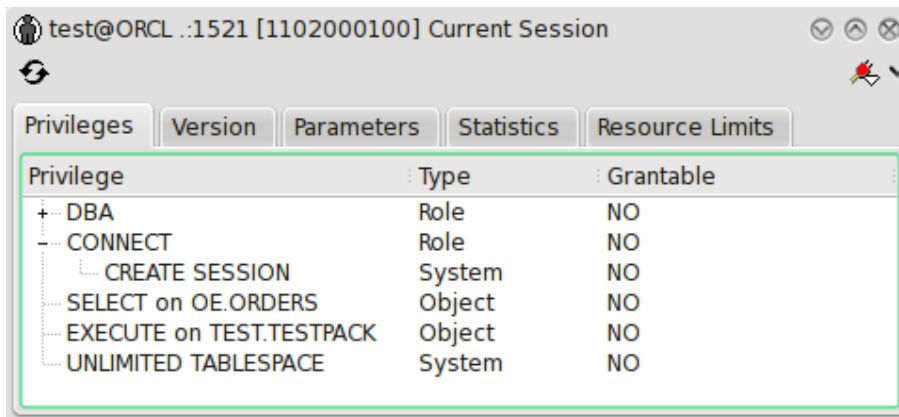
### Change connection

Change the connection to administrate storage for.

Don't forget you can sort the view by selecting a column header in the SGA list.

## 4.12 Current Session

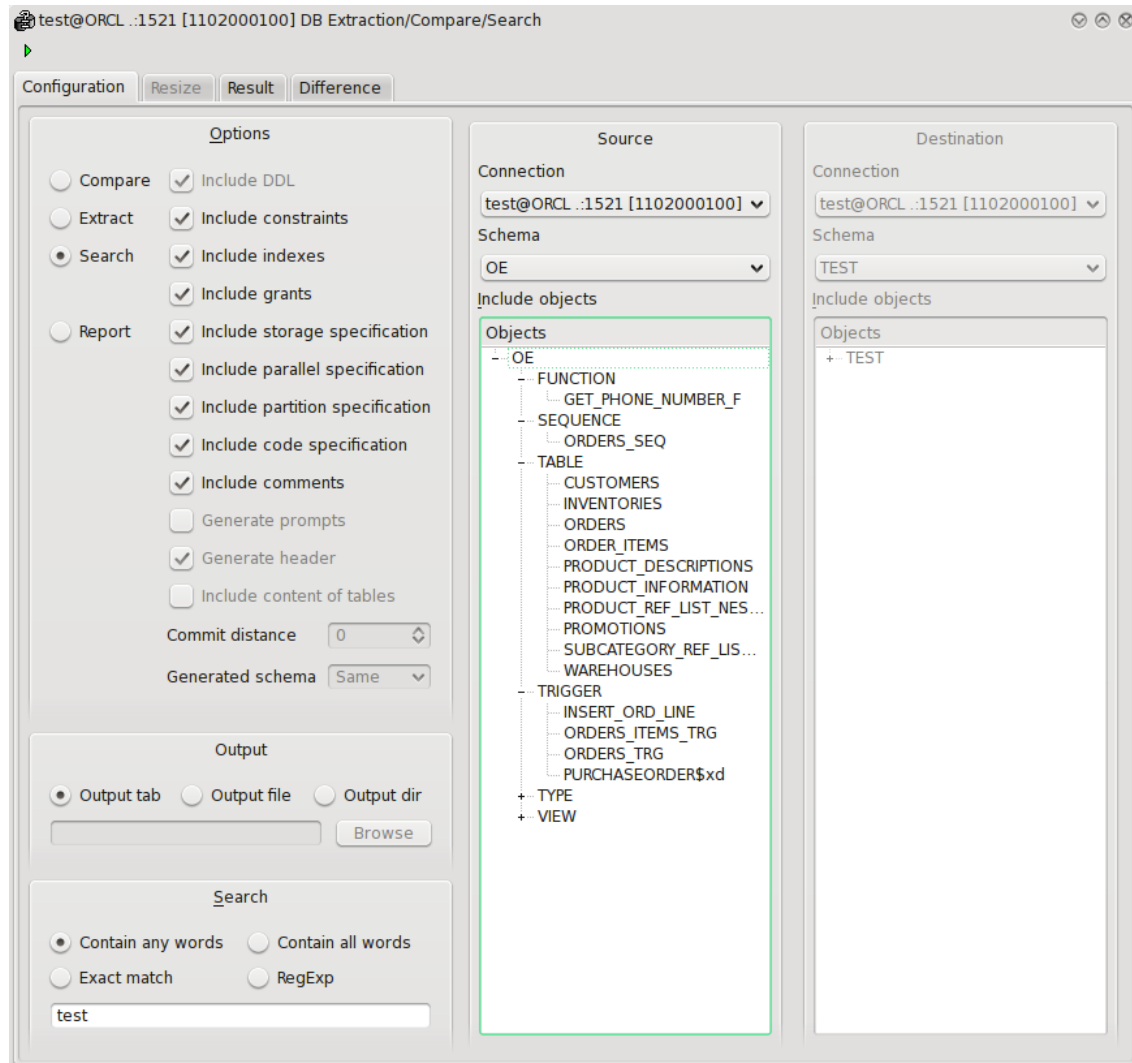
This tool provides information about the current session.



The interface for this tool is very simple. There are four tabs in the window. The first one present the **privileges** currently granted to the your session. Roles can be expanded to display what privileges are available through those groups. The second one present the **version** of the different components in the connection. The **parameters** tab displays the parameters of the current session (for more information of parameter editors see [Section 4.6 \[tuning\], page 54](#)). The fourth and last tab display the **statistics** of the session. The statistics is shown in two columns, the first one contain the actual value, the second show the change in the value since the last update. To update press the **refresh** button in the toolbar. You can also change the connection for the tool using the change connection button to the right on the toolbar.

### 4.13 Schema extraction, compare and search

This tool is used to either extract a script to recreate database objects, compare database objects or search database objects.



#### Defining the operation

The first thing to select is if you want to **extract** script to recreate an object, **compare** or **search**. You choose this by selecting one of the radio buttons to the left of the screen.

Next you should indicate what kind of information you want included in the extract or compare. This is selected with the checkboxes just to the right of the mode selection. Depending on the mode some options may be disabled.

#### Include DDL

Include any database definition.

#### Include constraints

Extract constraint definitions from tables.

**Include indexes**

Extract indexes for tables.

**Include grants**

Extract grants for database objects.

**Include storage specification**

Extract storage specifications for database objects.

**Include parallel specification**

Extract parallel specification for tables.

**Include partition specification**

Extract partition specification for tables.

**Include code specification**

Extract code for packages, procedures and functions.

**Include comments**

Extract comments for database objects.

**Generate prompts**

Generate prompts in the extracted result indicating what the script is doing.

**Generate header**

Generate header of the script.

**Include content of tables**

Extract the content of tables as well.

**Commit distance**

TODO.

**Generated schema**

This specifies the schema to be used when extracting. If you select Same, the same schema as the source database is used. If you select None, no schema is generated. Specifying anything else will replace the source schema with the one entered here.

If you are searching you also need to indicate what you are searching for in the bottom left of the front page.

**Containing any words**

The object should contain at least one of the words in the line editor.

**Containing all words**

The object should contain all of the words in the line editor.

**Exact match**

The object should contain the exact string as in the line editor include whitespaces.

**RegExp**

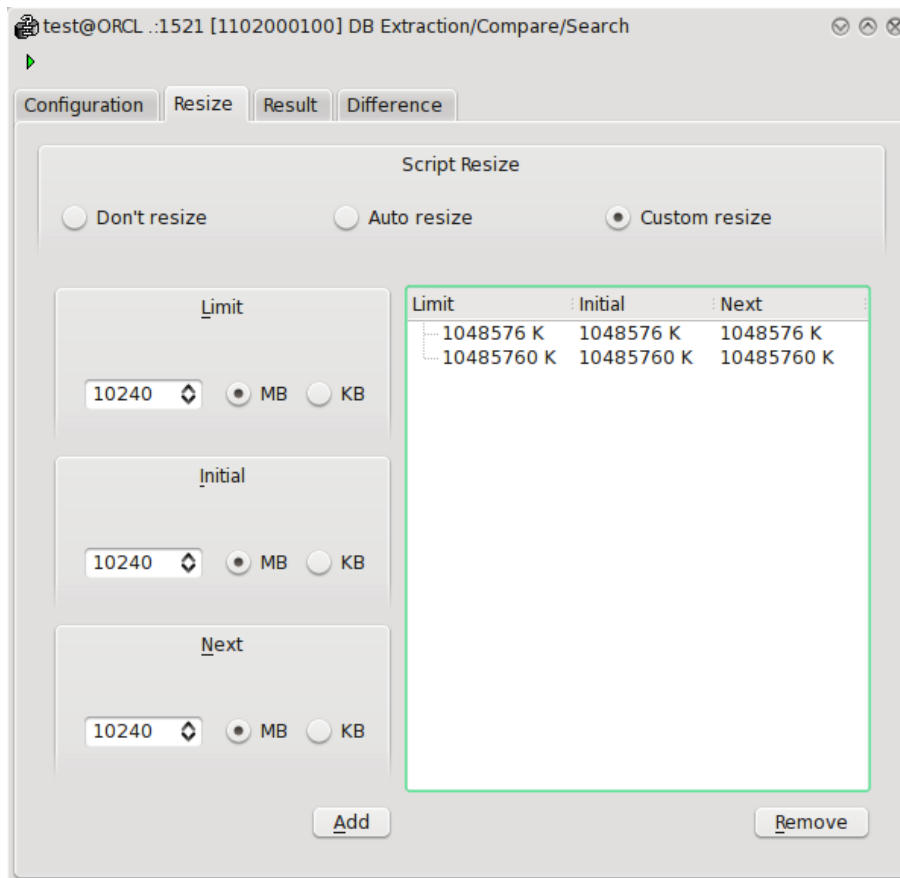
The object should contain at least one of the words in the line editor. The dialect of RegExp:s is the ones in the version of Qt that TOra is compiled against. For more information on regular expressions check out <http://doc.trolltech.com/qregexp.html>, if you are using Qt 3.0 or later you have a more advanced form of regular expressions.



The next thing you need to do is to specify which objects you want to extract or compare. If you extract you only need to specify the source, if comparing destination objects are also needed.

Selecting or deselecting an item in this list will select all it's child items.

If you specify the **extract** mode you can also indicate a way to resize generated storage specifications. This could be useful to make a database have only a few standard storage specifications making the database less prone to fragmentation.

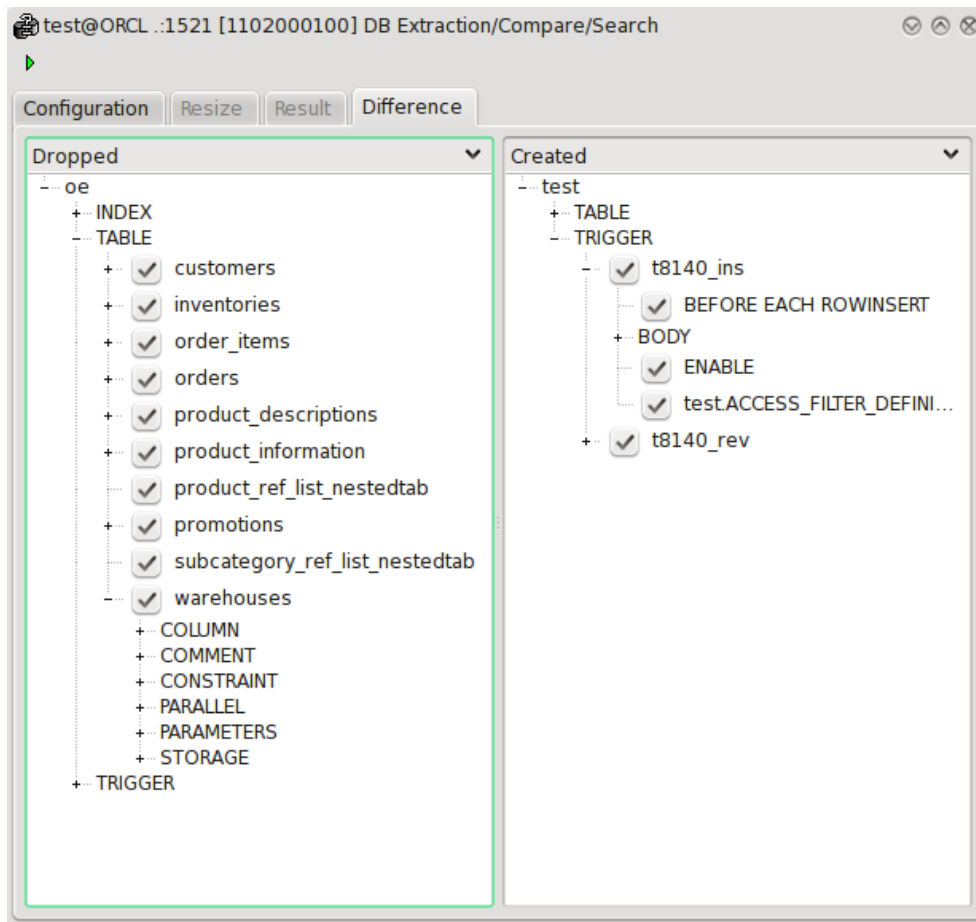


You have three modes to specify how resizing should work. First there is **Don't resize** which means leave the same as the original. Then there is **Auto resize** which should work for pretty much everyone. In this a resize specification for each order of magnitude of size is given a specific storage specification. And last is the **Custom resize** mode in which you specify how to transform the storage specifications to new values.

When you specify a custom resize you add items to the list with a **limit** size, **initial** size and **next** size. The sizes are used in the following way. When TOra is to determine the storage clause of an object it will take the **initial** and **next** sizes of the row in this list with the lowest **limit** size still higher than the current size of the object. If no row is found the one with the highest **limit** is used.

## Reading the result

If you **extract** the result is available under the **Result** tag of the dialog in the form of a worksheet tool where you can either start executing it or save it to a file. If you are comparing objects you will get the result under the **Difference** tab. The tab will look something like this.

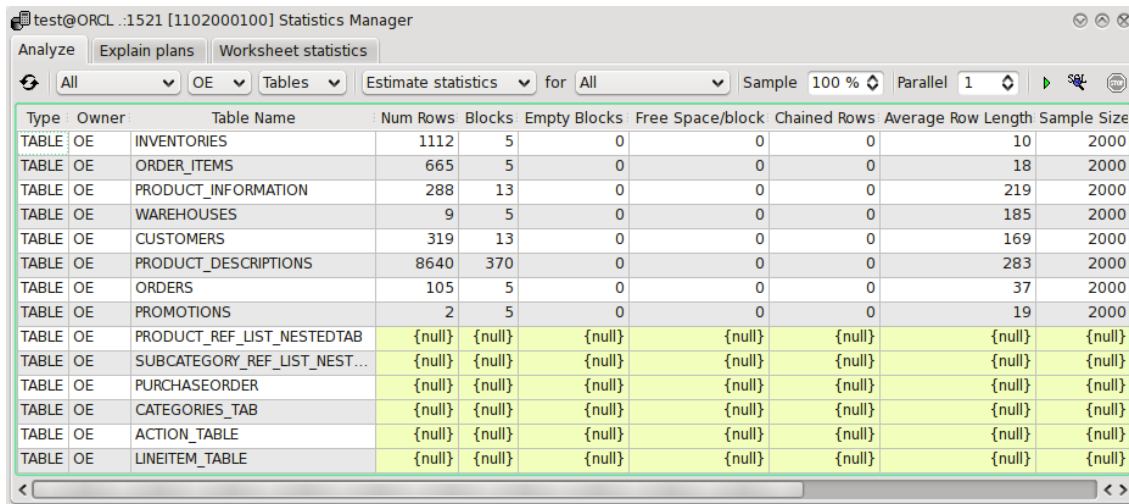


On the left is a pane containing objects only available in the source database. On the right is a pane containing the objects only available in the destination database.

Only the items in the lists that have checks on them are actually dropped or created, the other items only need to be added to lead the list tree to the items that are actually dropped or created.

## 4.14 Statistics manager

This tool is useful for collecting statistics and analysing tables in parallel. You can also use it to browse collected statistics.



The screenshot shows the Oracle Statistics Manager interface. The title bar reads "test@ORCL .:1521 [1102000100] Statistics Manager". The interface includes a toolbar with buttons for "Refresh", "Start analyzing", and "Stop". Below the toolbar is a table list with the following columns: Type, Owner, Table Name, Num Rows, Blocks, Empty Blocks, Free Space/block, Chained Rows, Average Row Length, and Sample Size. The table lists several tables in the OE schema, including INVENTORIES, ORDER\_ITEMS, PRODUCT\_INFORMATION, WAREHOUSES, CUSTOMERS, PRODUCT\_DESCRIPTIONS, ORDERS, and PROMOTIONS. Some rows are highlighted in yellow, indicating they are selected for analysis.

Type	Owner	Table Name	Num Rows	Blocks	Empty Blocks	Free Space/block	Chained Rows	Average Row Length	Sample Size
TABLE	OE	INVENTORIES	1112	5	0	0	0	10	2000
TABLE	OE	ORDER_ITEMS	665	5	0	0	0	18	2000
TABLE	OE	PRODUCT_INFORMATION	288	13	0	0	0	219	2000
TABLE	OE	WAREHOUSES	9	5	0	0	0	185	2000
TABLE	OE	CUSTOMERS	319	13	0	0	0	169	2000
TABLE	OE	PRODUCT_DESCRIPTIONS	8640	370	0	0	0	283	2000
TABLE	OE	ORDERS	105	5	0	0	0	37	2000
TABLE	OE	PROMOTIONS	2	5	0	0	0	19	2000
TABLE	OE	PRODUCT_REF_LIST_NESTEDTAB	{null}	{null}	{null}	{null}	{null}	{null}	{null}
TABLE	OE	SUBCATEGORY_REF_LIST_NEST...	{null}	{null}	{null}	{null}	{null}	{null}	{null}
TABLE	OE	PURCHASEORDER	{null}	{null}	{null}	{null}	{null}	{null}	{null}
TABLE	OE	CATEGORIES_TAB	{null}	{null}	{null}	{null}	{null}	{null}	{null}
TABLE	OE	ACTION_TABLE	{null}	{null}	{null}	{null}	{null}	{null}	{null}
TABLE	OE	LINEITEM_TABLE	{null}	{null}	{null}	{null}	{null}	{null}	{null}

You control the interface through the controls in the toolbar. The rest of the interface simply displays currently available tables in the current schema. When collecting statistics only the selected items in the tablelist are analysed.

**Refresh** Refresh the table list from the database.

**Schema** The next control is which schema to investigate tables for.

**Operation** What operation to perform. Select one of **compute statistics**, **estimate statistics**, **delete statistics** and **validate references**.

**Target** Depending on the **operation** you can sometimes select what kind of statistic to collect. Choose between **all**, **table**, **indexed columns** and **local indexes**.

**Sample** If the **operation** is **estimate statistics** you can choose how large sample should be used to estimate the statistic information in percent of total amount of data.

**Parallel** Indicate the number of separate connections to open to the database and run collecting statistics concurrently when collecting.

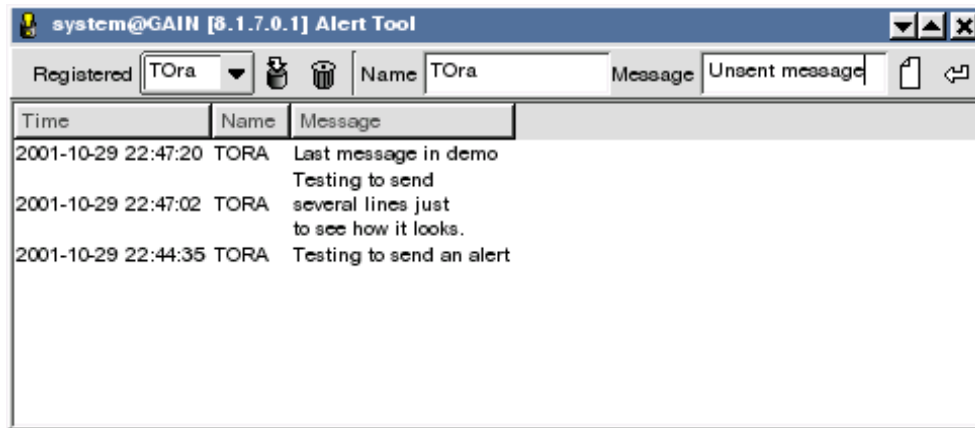
### Start analyzing

Press this button to perform the indicated operation on the tables selected in the list below the toolbar.

**Stop** Stop current run. Before the stop button you can also see information about the currently running statistics gathering run if any is running.

## 4.15 Alert Tool

This tool is useful for debugging database packages that send or depend on receiving dbms alerts.



This tool is controlled solely from its toolbar which is followed by a list containing the alerts received so far on the watched alerts. The toolbar contains the following controls.

**Registered** This list contains the alerts that are registered for listening. To add a new alert simply write the name in this combo box and press return or the register current button immediately to the right. To remove an alert select it in the list and press the remove registered button. 'Registered'.

### Register current

Register the name currently entered in the registered combo box.

### Remove registered

Stop listening to the registered alert currently selected in the registered combo box.

**Name** Name of the alert to send.

**Message** Message to send in an alert. Pressing return in this line edit will send the alert.

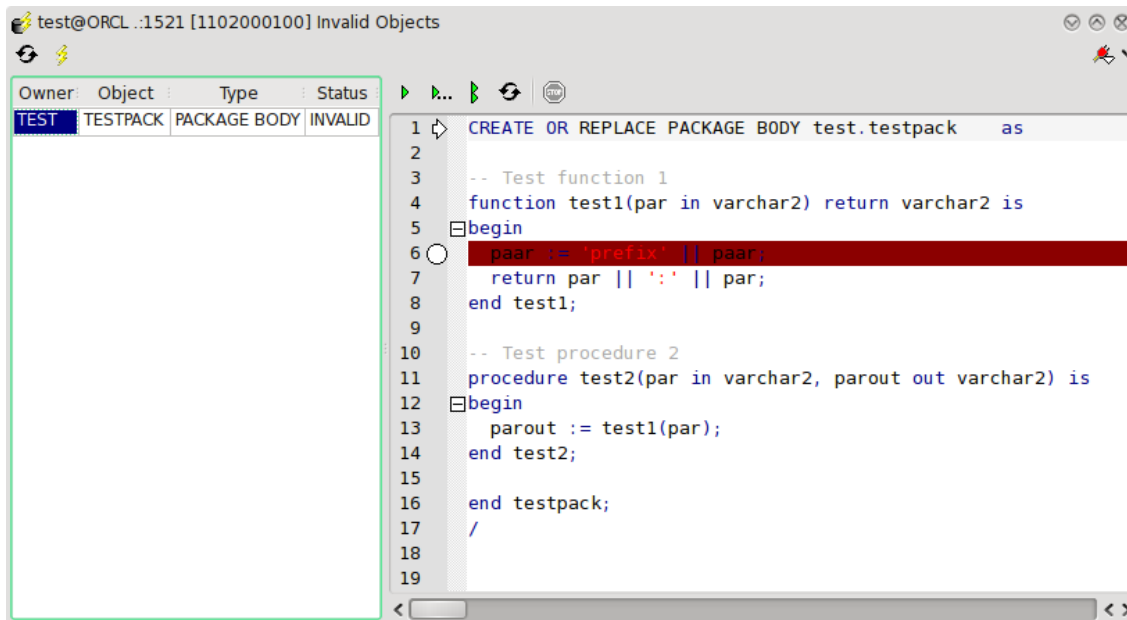
### Edit message in memo

Edit the message in a memo editor, useful for sending newlines and longer messages.

**Send alert** Send the currently defined alert. You can also send an alert by pressing return in the message line editor.

## 4.16 Invalid Objects

This tool is useful to recompile invalid objects in the database.



To the left you see a list of invalid objects. Select an object and the editor on right hand side will be filled with code to recreate the object, you can then fix whatever problem may be with it. Rows with errors will have red background and you can see the error in the statusbar by putting the cursor on the row. You can also refresh the list or change the active connection by the buttons in the toolbar.

### Recompile all invalid

This button will try recompiling all displayed invalid objects. Different types of objects are recompiled by calling different statements: Index:

```
alter index <schema.index_name> rebuild
```

Package body:

```
alter package <schema.package_name> compile body
```

Public synonym:

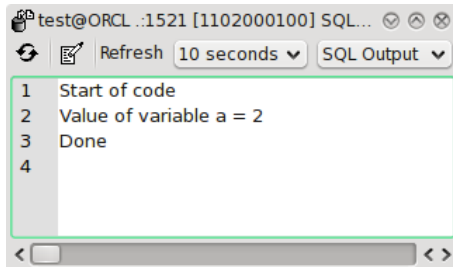
```
create or replace public synonym <synonym_name> for <synonym_target>
```

All other objects are recompiled using:

```
alter <object_type> <schema.object_name> compile
```

## 4.17 SQL Output

This tool displays the output from the DBMS\_OUTPUT package (For more information see the Oracle PL/SQL Supplied Packages manual).



Use this if you are running PL/SQL from the worksheet (see [Section 4.1 \[worksheet\]](#), [page 34](#)) and want to check the output. It is also used in the debugger (see [Section 4.4 \[debugger\]](#), [page 48](#)) but as a pane in the debugger information.

The toolbar contain the following controls.

**Update** Poll for more output immediately.

**Clear output**  
Clear the currently collected output.

**Refresh** The time-out to update automatically. The default value of this is configurable though the options.

**Type of output**  
You can choose if TOra should fetch output from standard oracle output or from Log4plsql module.

## 4.18 Template Help

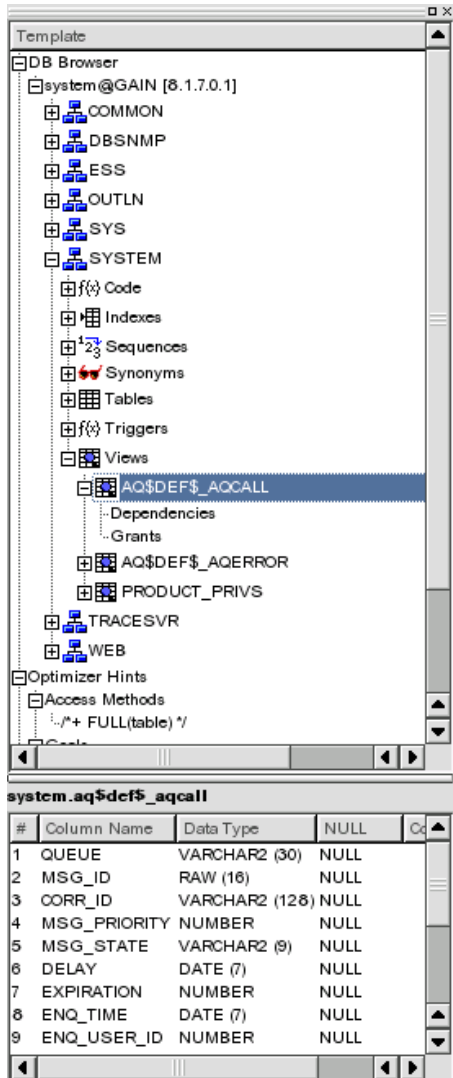
This tool provides help for writing PL/SQL and SQL queries by allowing a non obtrusive database browser and help lookup.

The template help is implemented as a docked window if these are available. They are available in Qt 3.0 (Released only as beta as of 2001-07-11) and KDE.

The help browser are divided into two panes. The top pane displays a tree view of the available documentation. The lower is used to display more information about the selected item in the top pane and it's content may differ depending on what is selected.

The top pane is in the default distribution divided into two parts as well. The DB Browser (see [Section 4.2 \[browser\]](#), [page 39](#)) which lets you explore your connected databases through a tree structure. The second part is a plain tree list of available text documentation. By default only one manual is supplied which contain PL/SQL functions, but an editor is supplied to write more help.

Items in the second part is simply items with a descriptive text associated with them. Selecting the item will bring up the description of it in the information.



## DB Browser

The database browser allows you to browse the schema objects in the open databases. All the open databases will have an item under which the following information is available. Under each table is a list of the visible schemas for the database. Under the schema the following items are available.

**Code** Under this item all the code objects are displayed. The information in the child objects will bring up the source of the object. There are also child items to this that contain information about **grants** and **dependencies**.

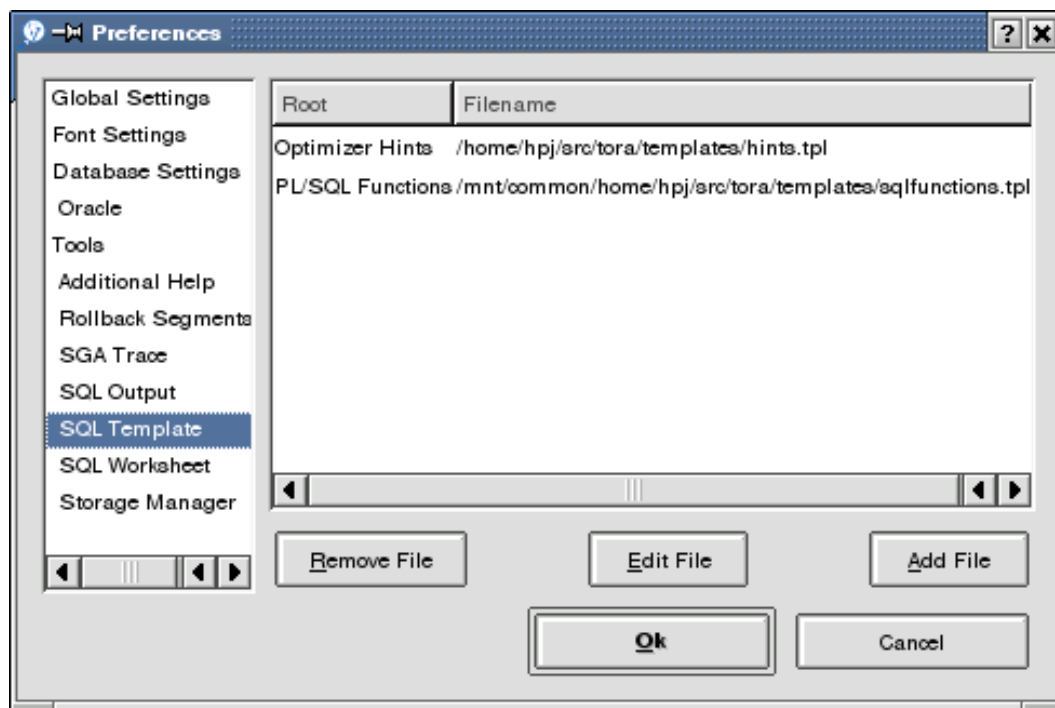
**Indexes** Contains all the indexes of the schema. The information in the child objects will bring up the columns on which the index is defined.

- Sequences** The sequences of the schema. The information in the child objects will bring up information about the sequence. There are also child item to this that contain information about **grants**.
- Synonyms** The synonyms of the schema. The information in the child objects will bring up information about the sequence. There are also child item to this that contain information about **grants**
- Tables** The tables available in the schema. Selecting the table will bring up the columns of the table. There are also three child items to this that contain information about **constraints**, **grants** and **references**.
- Triggers** Triggers in the schema. Selecting the item will bring up the source for the trigger. There are also child items to this that contain information about **grants** and **dependencies**.
- Views** The views available in the schema. Selecting the view name will bring up the columns of the table. There are also child items to this that contain information about **grants** and **dependencies**.

Don't forget that more database browsing functionality is available in the database browser tool (see [Section 4.2 \[browser\]](#), page 39).

## Setup help templates

You set up additional help files by selecting the SQL Template page in the options dialog (see [Section 3.8 \[preferences\]](#), page 20).





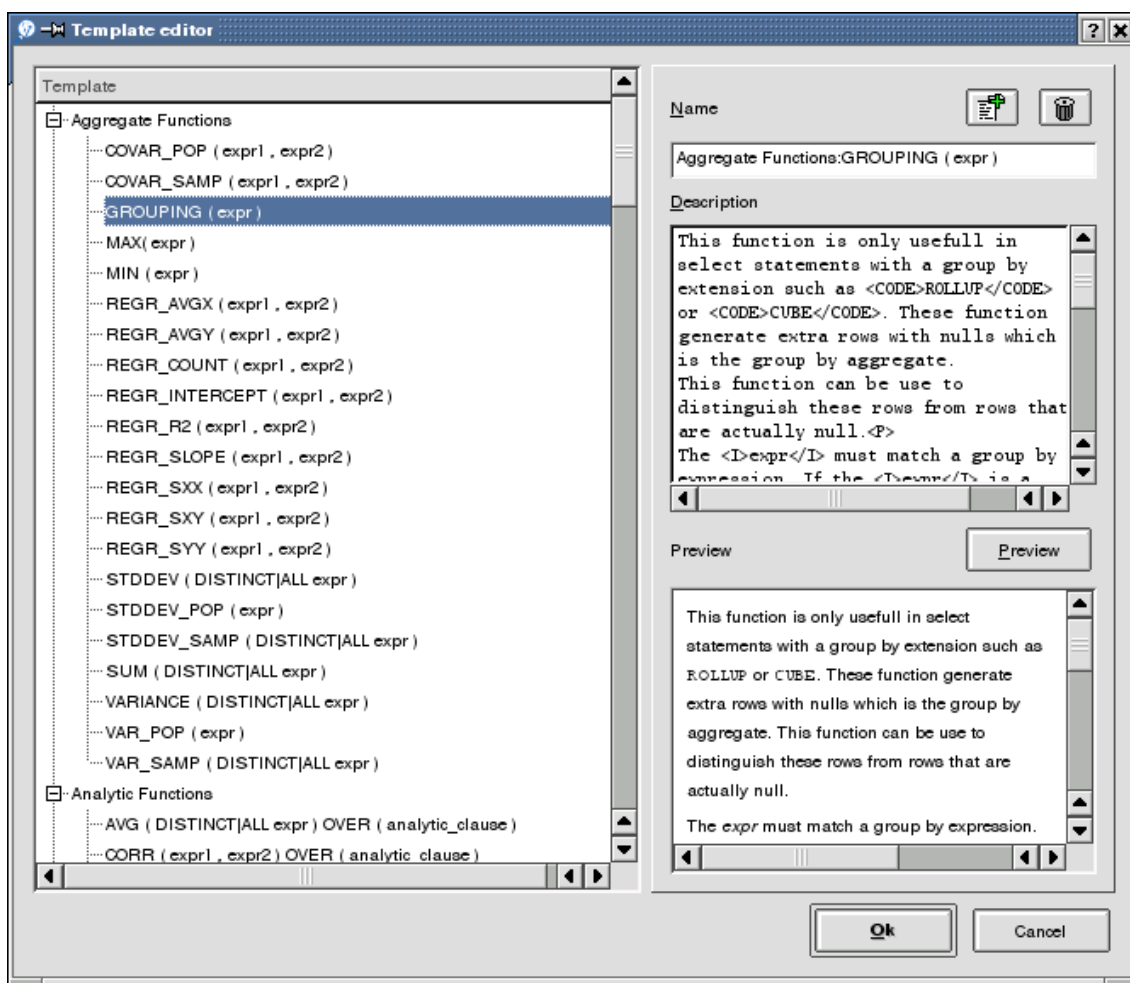
To add an existing or start a new template help collection select the **add file** button and enter the filename under which to store the help collection and the name of the root item under which the collection should be displayed in the template help tree.

To edit a collection select the collection in the list and press the **edit file** button to bring up the template editor. If the name doesn't exist you are asked to start a new file or cancel.

To remove a template press the **remove file**, this will not remove the actual file.

## Template editor

The template editor is used to edit and create new text template help collections.



The editor are separated into a few different parts. On the left is a list of available items. To the top right are two buttons **add new template** and **remove current template**. Then comes a line editor where the name of template is. If a ':' character is in the name it will split into a tree in the template collection tree. Then comes a text editor where the actual description of the title is written. You can specify simple HTML in this editor. Finally there is a button called **preview** which will give you a preview of the editor result of HTML.

The result is not saved until you press OK in the dialog. Pressing cancel will discard any changes made.

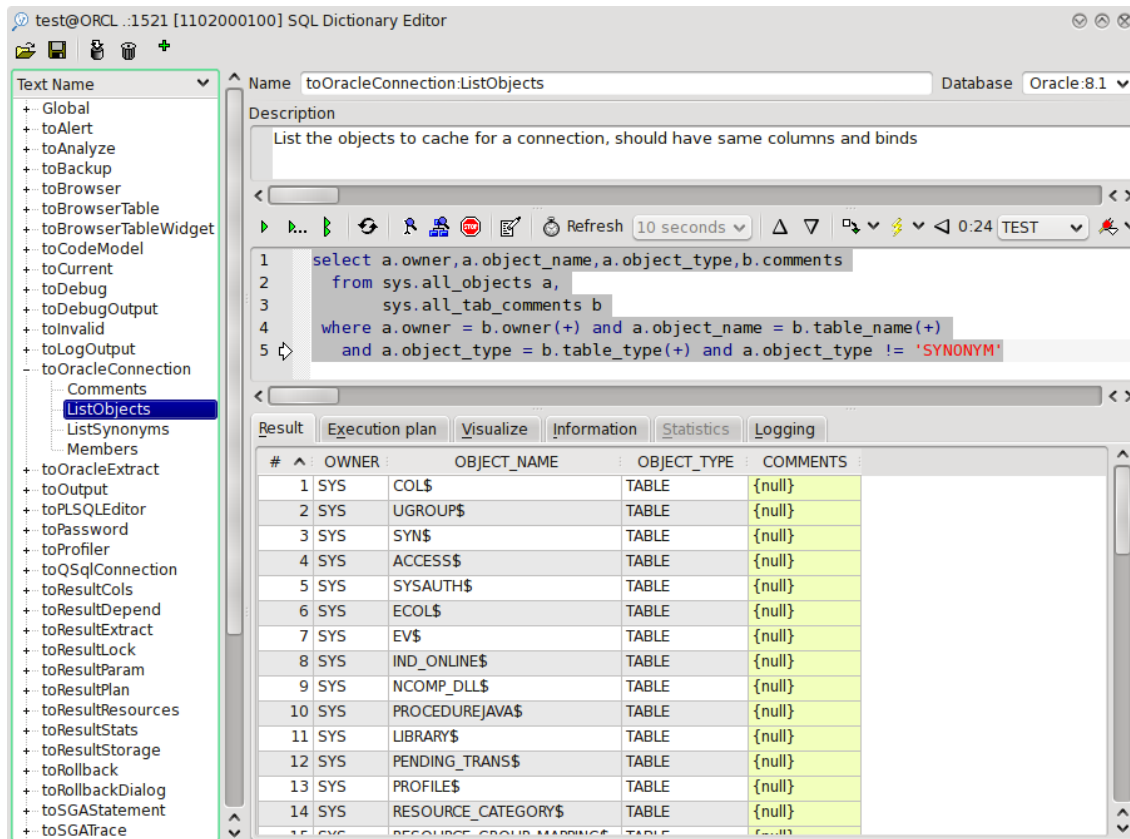
## 4.19 SQL Editor

When TOra needs to run an SQL statement it will not actually use the SQL needed directly. It will fetch an SQL statement by addressing it with a string. The string is built up by a module name, followed by a colon and a unique name for the SQL (Example: toBrowser:ListIndex which will contain the SQL statement for getting all the indexes for a schema).

For each SQL name there may be a number of actual SQL strings that are available, each are associated with a version number. When TOra requests an SQL statement by a name it will receive the string with the statement with the highest version number lower than the version of Oracle for the current connection that will be used to run it. Each SQL name also has a description associated with it so you can understand what the SQL is supposed to do.

TOra has a default SQL name to statement map which can be redefined by the user to either add support for different versions than TOra originally supports, or change the default behavior for TOra in some cases. The changes from the default map is saved in the file defined in the options dialog (see [Section 3.8 \[preferences\], page 20](#)). If you make changes to improve support for other versions of Oracle please send this file to [tora-develop@lists.sourceforge.net](mailto:tora-develop@lists.sourceforge.net) and we will incorporate them in the future releases.

You start the SQL editor either by selecting **Edit | Edit SQL** or by selecting **Edit SQL** in the result list context editor. You are then presented with the following dialog.



The window is divided into several parts. To the left is a list of SQL statements available divided into modules. When an item is selected in this list you can use the right part of the screen to modify the SQL associated by the selected name.

The right part of the window is divided into the name and version at the top. The version is also preceded with the name of the provider (Usually **Oracle**) separated by a colon. Each statement name can have several versions of an SQL. You can add a new version by simply entering a new text into the version combobox. You can also select any of the existing versions by selecting them from the list.

Below the name and version is the description which is independent from the version. Below that is a worksheet (see [Section 4.1 \[worksheet\], page 34](#)) which you can use to edit and test the SQL statement that should be associated with the given name and version.

There is also a toolbar with the current available commands.

**Open** Read in a file containing SQL descriptions saved before.

**Save** Save the changes made from the default to the SQL statements to a file. This file can be used to send updates and improvements of the internal SQL to incorporate in future versions.

**Save this entry**

Save the changes made. If changes have been made and you select a new item you will be asked to save or discard the changes if you forget to save entry before.

**Drop this entry**

Drop the current entry. If the last version of a statement is deleted the entire statement will be deleted. Observe that you can not delete default statements permanently, they will reappear when you restart TOra so you can't really screw up with this.

**New SQL** Start new SQL statement definitions.

## 5 Extending TOra

### 5.1 TOra Tool Tutorial

This tutorial will create a simple tool that can execute an SQL statement and display it's content in a list. This tutorial assumes you have knowledge of C++ and Qt programming.

First of all we create an include file which defines the tool widget. This is the window that will be displayed when the a tool window is created. How that happens comes later.

```

#ifndef TOSIMPLEQUERY_H
#define TOSIMPLEQUERY_H

#include "toutil.h"

class toResultView;
class QLineEdit;
class toConnection;

class toSimpleQuery : public toToolWidget {
    Q_OBJECT

    toResultView *Result;
    QLineEdit *Statement;
private slots:
    void execute(void);
public:
    toSimpleQuery(QWidget *parent,toConnection &connection);
};

#endif

```

If this file is put in the main TOra source directory the configure script will generate the appropriate moc file which will be needed later. If you don't know what moc is, please go back to the Qt manual to read up on slots and signals.

The rest is pretty straight forward and will be much clearer after we start working our way through the implementation of this tool.

The first thing we need to do is create a tool description which is done by subclassing the toTool class. This is how we implement this class for this function.

```

#include "toutil.h"
#include "tosimplequery.h"

static char * tosimplequery_xpm[] = {
    "16 16 3 1",
    " c None",

```



`'pictureXPM'`

This function should if defined return a pointer to a xpm definition. This will then be used for toolbar icon, menu icon and tool window icon. `'pictureXPM'`

`'menuItem'`

Should return a string containing the name of the menu item to add to the tools menu. This is also used for a tip for the toolbar icon by default. `'menuItem'`

`'toolWindow'`

This function will create a new tool widget and return a pointer to it. It doesn't need to create a widget, then it should then return NULL. Some tools there might only be one per connection for instance. `'toolWindow'`

The number in the constructor is a priority indicator that denote where in the list of tools this tool should be inserted, between each 100 step a separator is inserted into the toolbar and menubar.

And last a not so obvious line where the an instance of the tool descriptor is instantiated. This is a feature that is used a lot in TOra. This will ensure that all tools are registered when the application is started. It also works when using modules, if this is compiled as a module the tool will be instantiated on loading without any hassle with functions with predefined names and such stuff.

Lastly comes the implementation of the tool widget which is not much longer. I have divided this into several parts to simplify explaining them.

```
#include <list>

#include <qtoolbar.h>
#include <qlabel.h>
#include <qtoolbutton.h>
#include <qlineedit.h>

#include "tosimplequery.h"
#include "toresultview.h"
#include "toparamget.h"
#include "tochangeconnection.h"

#include "tosimplequery.moc"

static char * execute_xpm[] = {
    "16 16 3 1",
    " c None",
    ".c #000000",
    "+c #OFFE14",
    "      ",
    "      ",
    "      ",
    "      ",
    "      ",
    "      "
}
```

```

"      .+.      ",
"      .++.      ",
"      .+++      ",
"      .+++      ",
"      .++.      ",
"      .+.      ",
"      ..       ",
"      .        ",
"              ",
"              ",
"              ",
"              "};

toSimpleQuery::toSimpleQuery(QWidget *main,toConnection &connection)
    : toToolWidget(SimpleQueryTool,"simplequery.html",main,connection)
{
    QToolBar *toolbar=toAllocBar(this,"Simple Query",connection.description());
    QPixmap executePixmap((const char **)execute_xpm);
    new QPushButton(executePixmap,
                    "Execute current statement",
                    "Execute current statement",
                    this,SLOT(execute()),
                    toolbar);
    toolbar->setStretchableWidget(new QLabel("",toolbar));
    new toChangeConnection(toolbar);
}

```

In this part the parent constructor is called and the toolbar is set up. Also note the inclusion of the moc file which by convention is called `tosimplequery.moc`. One thing worth noticing here is the `toAllocBar` which is used to be able to transparently support either using `KToolBar` or `QToolBar` depending on whether this is a Qt or KDE application. This is very important since TOra also supports windows to which KDE is not available.

The second part is the `setStretchableWidget` call which is used to indicate that an empty label should be stretch instead of the tool button which just looks really weird.

Also worth noting is that the `toToolWidget` class is derived from `QVBox` so any widgets constructed in this widget will be lined up vertically in the order of creation.

Next up is creating our widgets and connecting them.

```

Statement=new QLineEdit(this);
Result=new toResultView(this);
connect(Statement,SIGNAL(returnPressed()),this,SLOT(execute()));
}

```

This just adds two additional widgets and connect the `returnPressed` signal to the `execute` slot. One thing to realise here is that all the `toResult` children will use the connection of the closest parent of type `toToolWidget` in the widget hierarchy. And now the last thing to do is implement the `execute` method.



```

void toSimpleQuery::execute(void)
{
    try {
        QString sql=Statement->text();
        toQList params=toParamGet::getParam(this,sql);
        Result->query(sql,params);
    } TOCATCH
}

```

The toParamGet::getParam function is used to ask for bind values in the query string. To understand what I mean try executing the query "select :hello from dual" when you try the result. The toQList is simple a list of toQValue which can hold different datatypes and converting between them transparently.

Now finally to compile this module you need to add the tosimplequery.cpp file to SOURCES define in the file Makefile. To build a plugin you also need to add the following line.

```

plugins/tosimplequery.tso:objs/tosimplequery.o

```

This should go among the other plugin definitions and you also need to add the plugins/tosimplequery.tso to the dependencies of tora-plugin.

You must rerun configure for the tosimplequery.moc file to be generated the first time. Any subsequent changes should update the moc file automatically from the Makefile.

Here are the example files in their entire.

- `tosimplequery.h`
- `tosimplequery.cpp`

Hopefully this is a starting point to help you read the rest of the documentation and start cranking out those plugins.

## 5.2 API Reference

API documentation can be generated on your own directly from source code files.

You need to have Doxygen (doxygen.org) installed.

Base configuration for the documentation are located in TORA\_SRC\_DIR/doc/help/api directory.

Unix/Linux: just run make in this directory. Make clean and more various targets are predefined. See the Makefile.

MS Windows: Try make too. Or run doxygen manually.

## Index-list

### A

Additional Help Settings .....	29
Alert Tool .....	72
Available tools .....	34

### C

Connecting to a database .....	12
Current Session .....	66

### D

Database Settings .....	26
-------------------------	----

### E

Editor Settings .....	22
Elements of the main window .....	4
Extending TOra .....	81

### G

Getting to know your workspace .....	4
--------------------------------------	---

### H

History and future of TOra .....	2
----------------------------------	---

### I

Invalid Objects .....	73
-----------------------	----

### L

lists .....	15
-------------	----

### O

Other common elements .....	31
-----------------------------	----

### P

PL/SQL Debugger .....	48
PL/SQL Editor .....	46
PL/SQL Unit Tester .....	53
Preferences .....	20

### R

Rollback Segments .....	63
-------------------------	----

### S

Schema Browser .....	39
Schema extraction, compare and search .....	67
Search & Replace .....	18
Security Manager .....	56
Server Tuning .....	54
Session Manager .....	61
SGA Statement .....	31
SGA Trace .....	65
SQL Editor .....	34, 78
SQL Output .....	74
Statistics manager .....	71
Storage Manager .....	59

### T

Template Help .....	74
Tool Settings .....	30
TOra Tool Tutorial .....	81

### U

Using charts .....	17
Using editors .....	14
Using the help browser .....	1

### W

What privileges do you need to run TOra .....	13
---	----