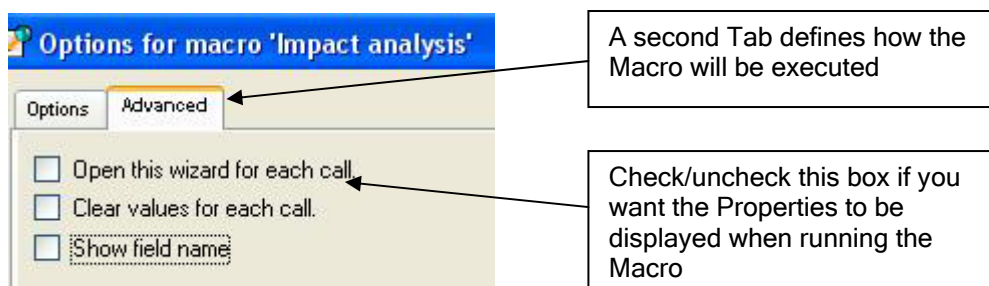
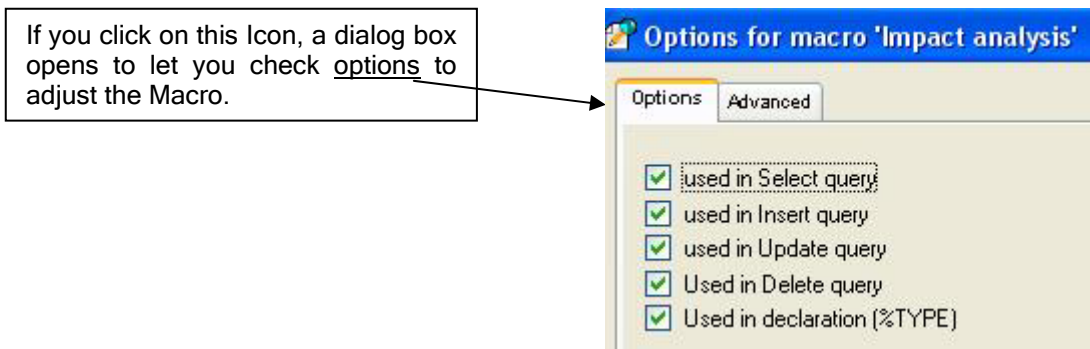
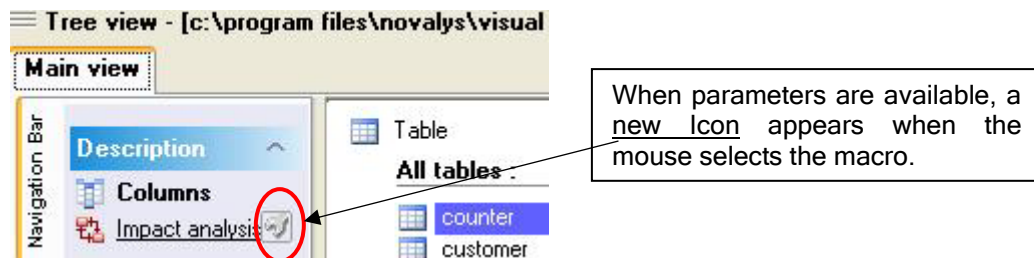


Introduction to Visual Expert 6.0 new features

1.	NEW NAVIGATION BAR	2
2.	NEW MACROS	3
2.1.	LESS MACROS, MORE PARAMETERS	3
2.2.	MACROS SHORTCUTS IN THE TREEVIEW MENU.....	3
2.3.	MACROS EXTENSIONS	4
2.4.	FILTERING OBJECTS IN THE TREEVIEW	5
2.5.	FINDING SPECIAL DATAWINDOWS	6
2.6.	MAPPING DATAWINDOWS / DBMS	7
2.7.	POWERBUILDER ⇔ STORED PROCEDURES DEPENDENCIES.....	7
2.8.	INHERITANCE HIERARCHY (WINDOWS/UO/MENUS).....	8
3.	NEW IMPACT ANALYSIS	9
3.1.	RESULT DISPLAYED AT INSTRUCTION-LEVEL	9
3.2.	IMPACT ANALYSIS EXAMPLES.....	10
3.2.1.	<i>Impact Analysis on a TABLE</i>	10
3.2.2.	<i>Impact Analysis on a DB COLUMN</i>	11
3.2.3.	<i>Impact Analysis on a STORED PROCEDURE</i>	12
4.	EXPLORING REFERENCES WITH VISUAL EXPERT 6.0	13
4.1.	REFERENCES FOUND IN POWERBUILDER COMPONENTS	13
4.1.1.	<i>DataWindows</i>	13
4.1.2.	<i>PowerScripts</i>	14
4.1.3.	<i>RPC FUNC</i>	15
4.2.	REFERENCES FOUND IN PL/SQL OR T-SQL COMPONENTS.....	16
4.2.1.	<i>Oracle Package</i>	16
4.2.2.	<i>Stored Procedures</i>	17
4.2.3.	<i>Other DB Code items</i>	18
4.2.4.	<i>PL/SQL %TYPE references</i>	18
4.2.5.	<i>Parameters & Local Variables</i>	18
5.	SOURCE CODE VIEW	19
5.1.	HYPERLINKS IN THE CODE	19
5.2.	TITLE OF THE SOURCE CODE VIEW	19
5.3.	SEARCH IN THE SOURCE CODE VIEW	20
6.	TECHNICAL DOCUMENTATION	21
6.1.	TECHNICAL DOCUMENTATION FOR ORACLE STORED PROCEDURES	21
7.	MISCELLANEOUS	22
7.1.	OBJECT PREVIEWS	22
7.2.	TOOLTIPS.....	23
7.3.	“SHORT LIST” VIEW	25
7.4.	TECHNICAL REQUIREMENTS	26

1. New Navigation Bar

With Visual Expert 6.0, some treeview macros include parameters. These parameters let you adjust the result expected from the Macro.



2. New Macros

2.1. Less Macros, more parameters

All treeview macros have been redeveloped and reorganized.

A few standard macros are now available for all languages (PB, PL/SQL, and T-SQL):

- Same concept and same Macro name from one language to another
- Each macro uses parameters to cover most needs
- These parameters depend on the type of component selected in the treeview

Component list for PowerBuilder :

Datawindow	54 components
PBL	13 components
Application object	1 component
Function object	8 components
Menu	17 components
Window	73 components
UserObject	410 components
Structure	41 components

Oracle PL/SQL :

PL/SQL File	6 components
Package	2 components
Stored procedure	15 components
Trigger	1 component
Cursor	2 components

Transact SQL :

T-SQL File (ASE)	6 components
Stored procedure	9 components

All stored procedures :

- OrdersbyEmployee
- Productsbycustomer
- as_customer_list

Definition: details about the selected component (methods, variables, ancestor, SQL query, parameters...)

References: lists the items **called by** (referenced by) the selected component

Impact Analysis: lists the items **calling** (referencing) the selected component

Called Hierarchy: chain of methods calling each other in the application

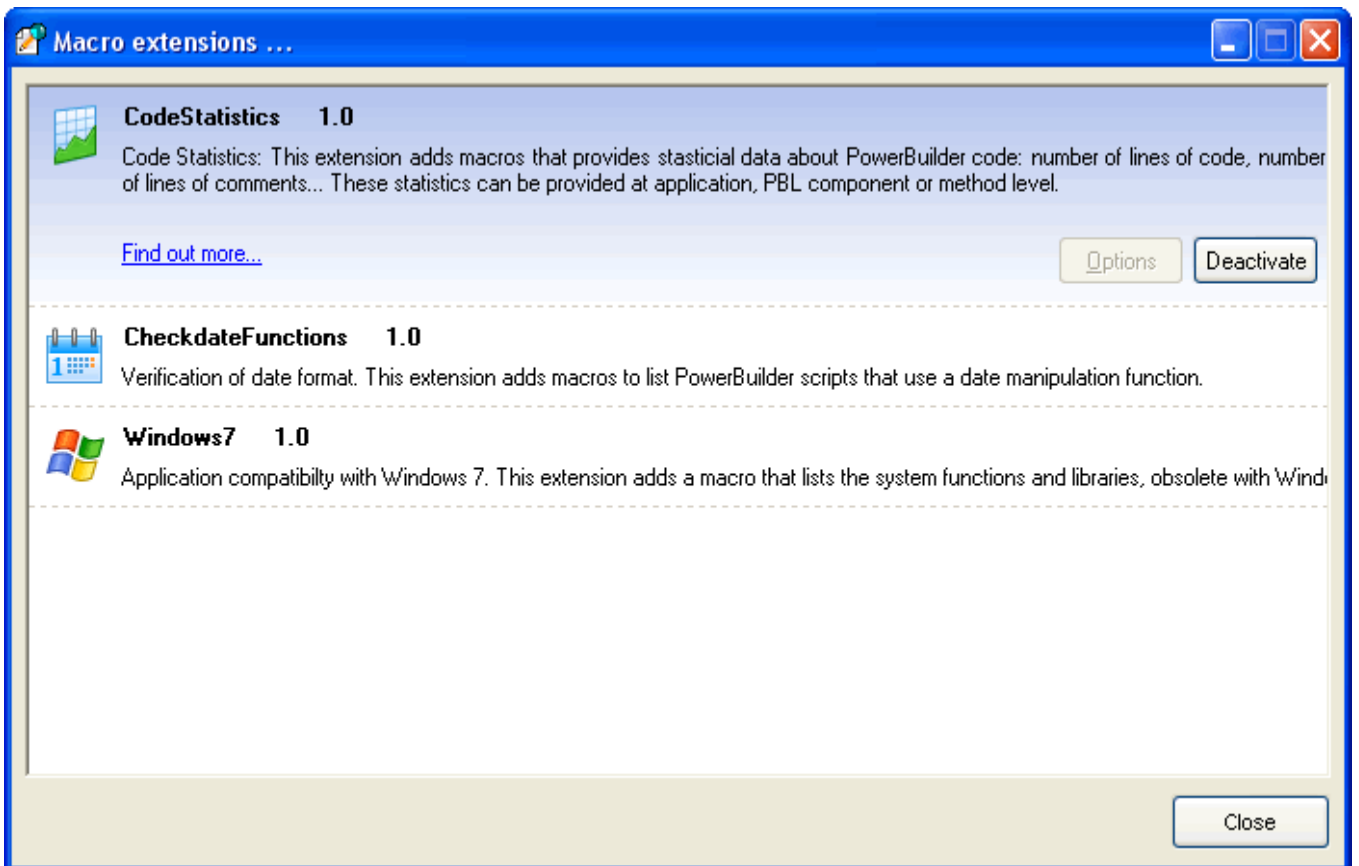
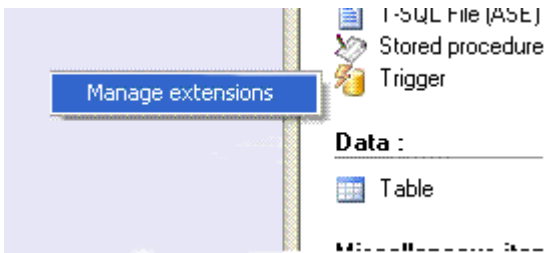
2.2. Macros shortcuts in the Treeview Menu

The pop-menu in the treeview (right-click on an item in the treeview) now includes the main macros available for the selected item – no need to go to the navigation bar.

Main macros available in the treeview Pop-menu

2.3. Macros Extensions

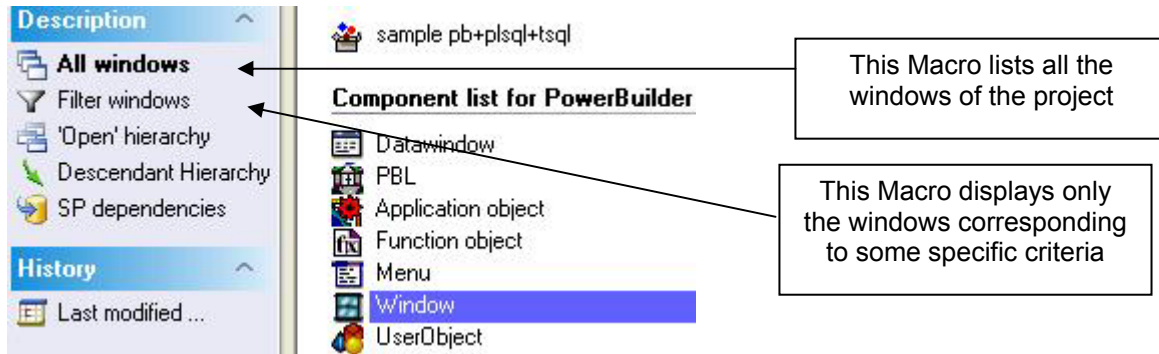
The system allowing the extension of macros is now available with a right-click on the Navigation bar. Some extensions have already been released. The list of macros will evolve.



2.4. Filtering objects in the treeview

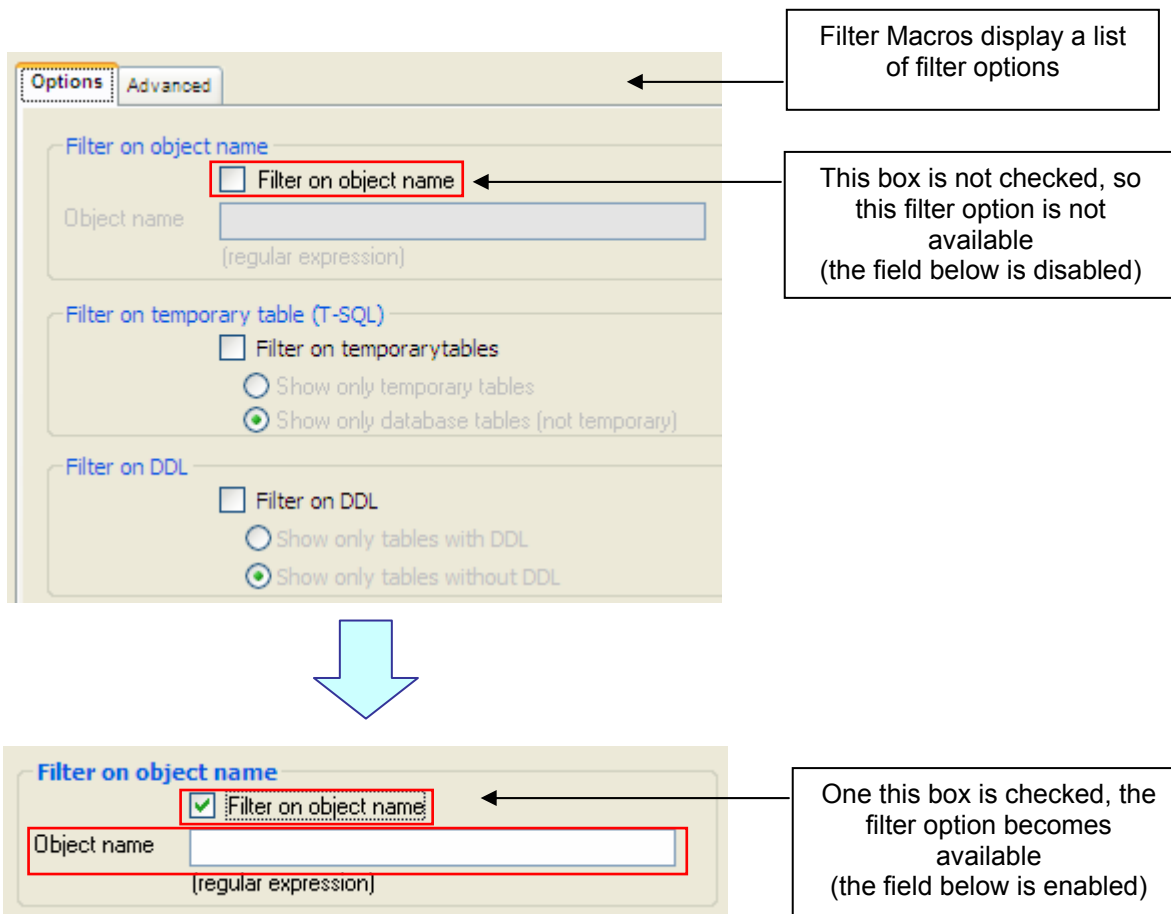
Objects can now be listed in the treeview with 2 macros:

- The macro “**All <Object>**” lists all the objects of this type.
This macro is executed by default with a double-click at the root of the treeview.
- The macro “**Filter <Object>**” offers various parameters to filter each type of object.



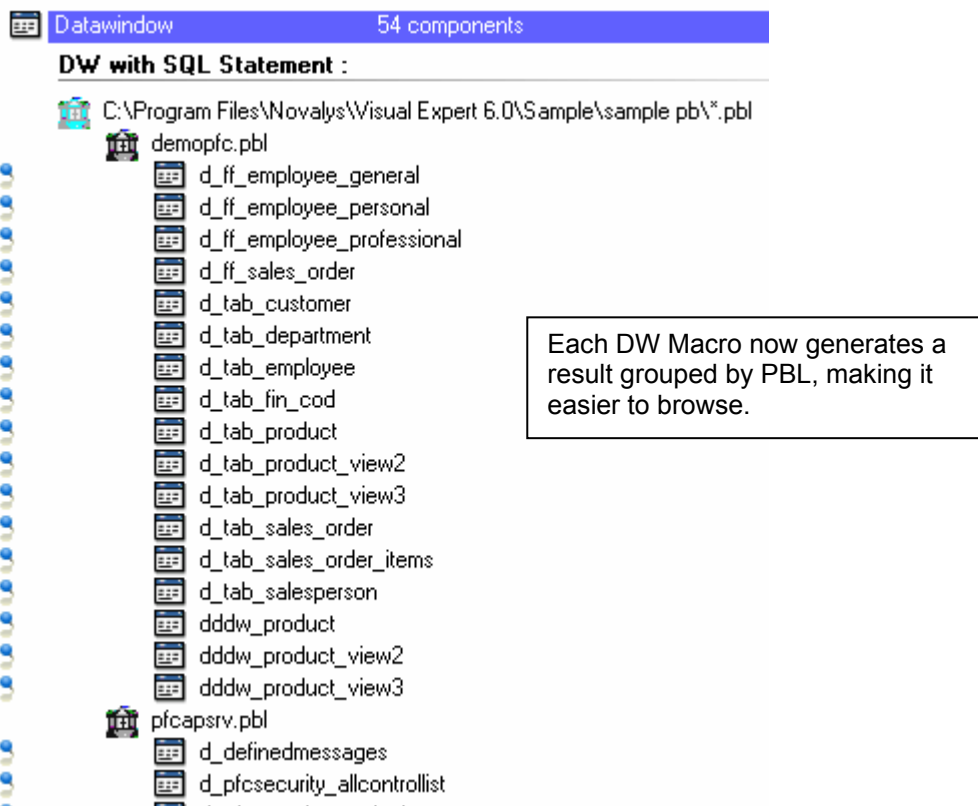
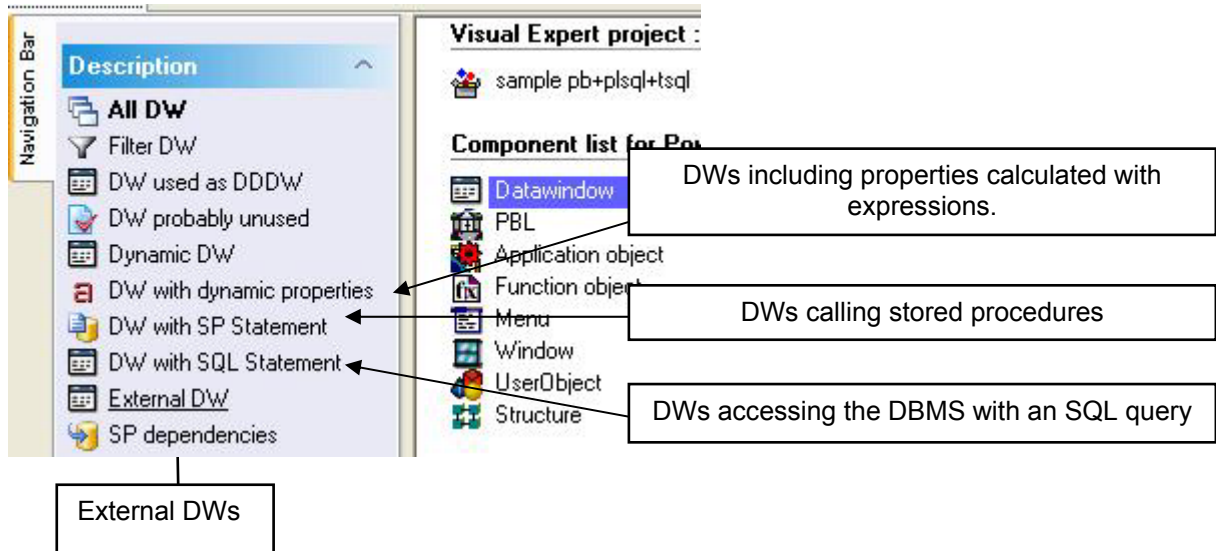
Each filter macro offers a list of filter options.

By default, these options are not activated. You can activate them by checking the corresponding box:



2.5. Finding Special DataWindows

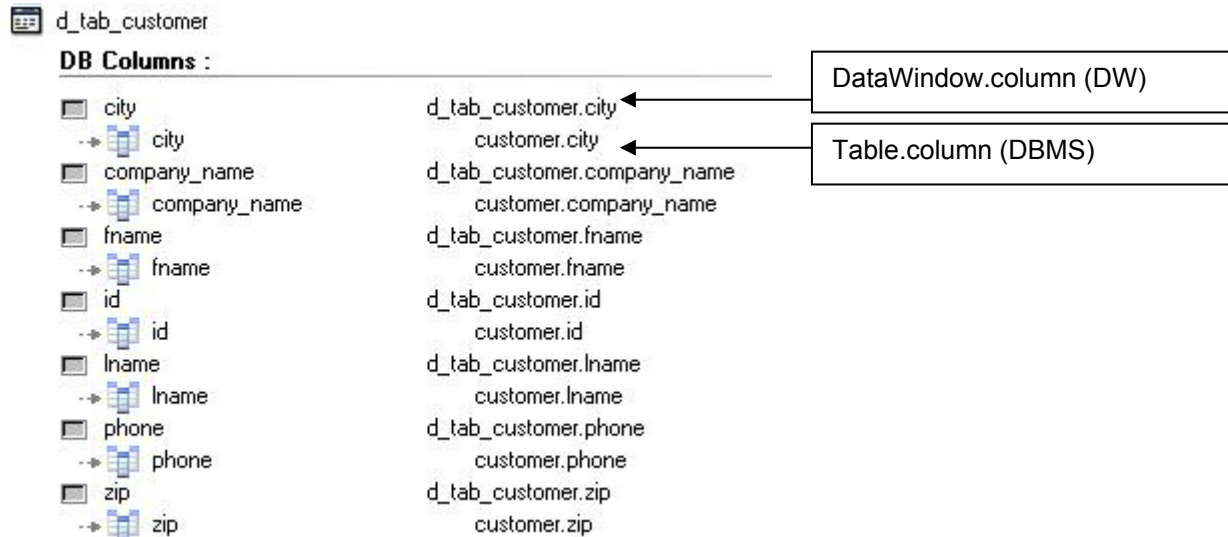
Several new macros have been added with Version 6.0, available at the root of the DWs:



2.6. Mapping DataWindows / DBMS

Visual Expert 6.0 also includes a new “DB columns” macro for DataWindows.

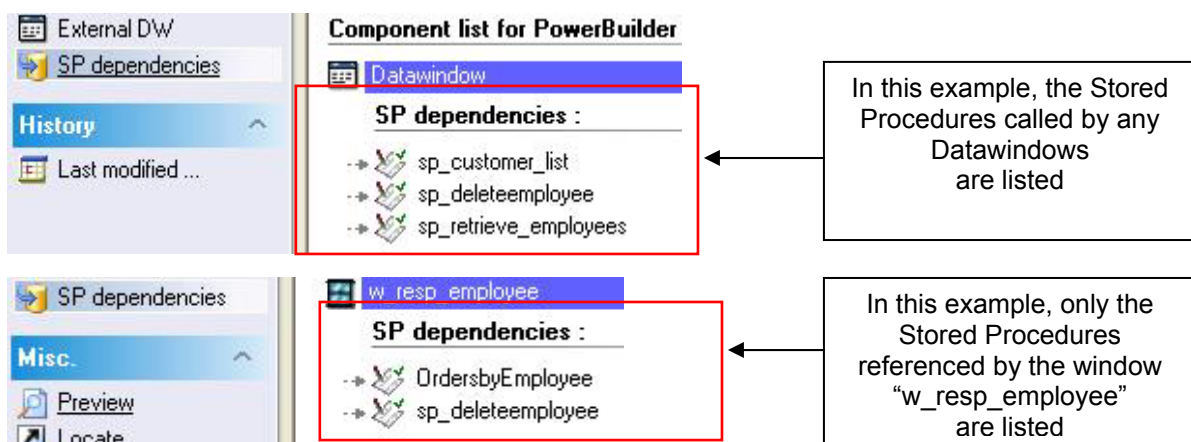
This macro displays the DW- DBMS mapping: it shows which table.column in the DBMS corresponds to each DW column.



2.7. PowerBuilder ↔ Stored Procedures dependencies

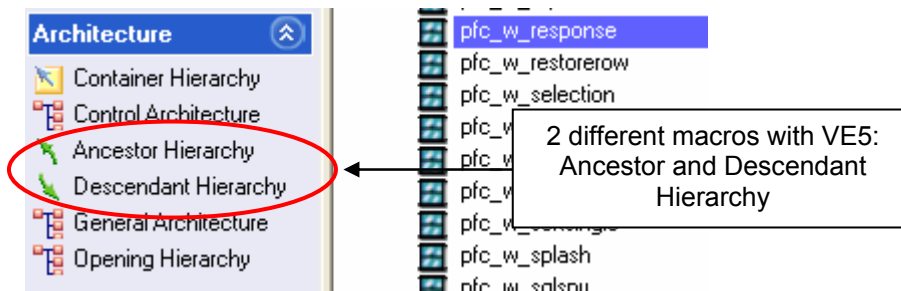
A new Macro called « SP dependencies » lists all stored procedures referenced by PB objects:

- This macro is available at the root of the treeview for each type of PB components
- It is also available for PB objects displayed in the treeview

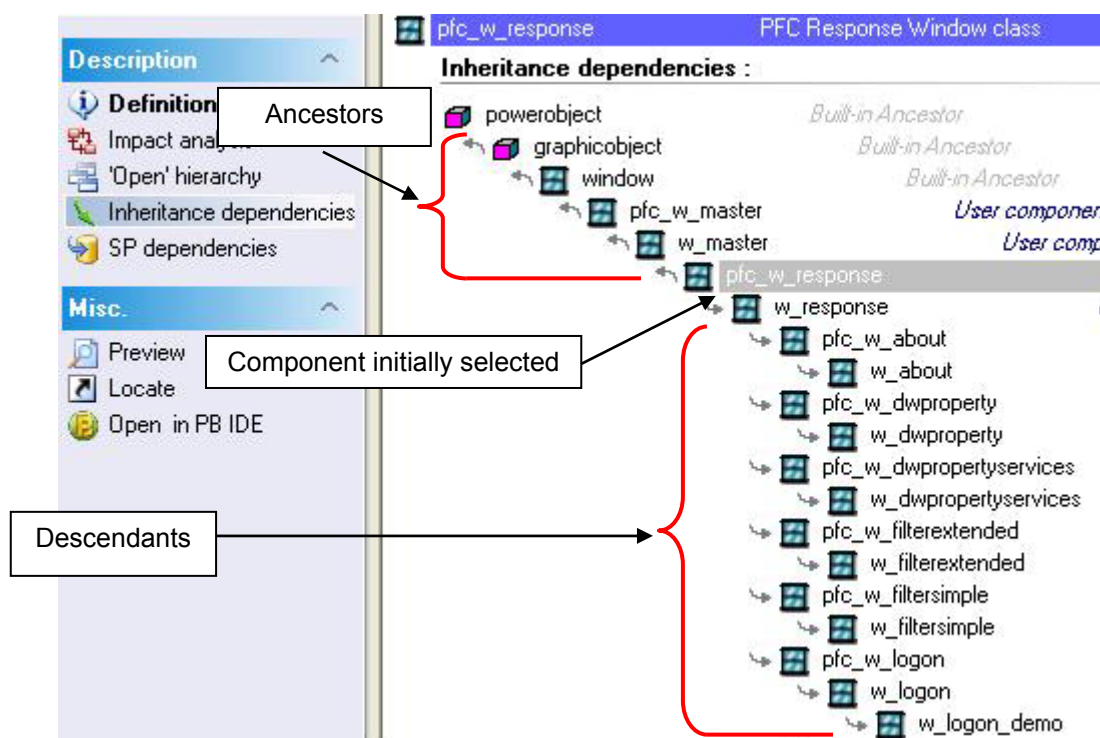


2.8. Inheritance hierarchy (Windows/UO/Menus)

Visual Expert 5.x offered 2 different macros: Descendants + Ancestors



With Visual Expert 6.0, the same Macro displays **both** Ancestors and Descendants:



3. New Impact Analysis

3.1. Result displayed at instruction-level

Visual Expert used to perform impact Analysis across the entire application and provide a component-level result in the treewiew. Visual Expert 6.0 now provides a fine-grain result at instruction level!

This icon means that this item belongs to the Impact Analysis result

This symbol means that this event is referencing the table "Customer"

This indicates how many references were found in this code, and the number the reference currently highlighted

customer
Impact analysis :
dempfc
d_ff_sales_order
Data Source
d_tab_customer
Data Source
d_tab_sales_order
Data Source
Pack_DataManager.plsql
DataManager
OrdersbyEmployee
Productsbycustome
sp_customer_list.sql
sp_customer_list
C:\Program Files\Novalys\Vis
DataManager.tsq
{Create procedure
OrdersbyEmpli
Productsbycu

```
39 return uf_GetUpperCase(lname) ||  
40 end uf_GetFullname;  
41  
42  
43  
44 procedure OrdersbyEmployee( Employee_I  
45 is  
46 Emp_fname varchar2;  
47 Emp_lname varchar2;  
48  
49 cursor c1 is  
50 SELECT sales_order.id Oid,  
51 sales_order.order_date Od  
52 DECODE(sales_order.fin_co  
53 'r1','Revenue','e1',  
54 sales_order.region Oregio  
55 customer.fname Cfname,  
56 customer.lname Clname,  
57 sum (sales_order_items.qu  
58  
59 FROM customer,  
60 employee,  
61 product,  
62 sales_order_items,  
63 sales_order  
64 WHERE ( employee.emp_id = Employee  
65 ( employee.emp_id = sales  
66 ( customer.id = sales ord  
67 ( sales_order_items.id =  
68 (product.id = sales_order  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

Find | references: 1 of 12

54 55 57 64 70 129 133 134 143 147 151 152

Each line including a reference are marked with this symbol.

This index lists all the lines of code with a reference. If you click on a number, the source code will scroll down to display this line.

You can navigate from one reference to the next by pressing [Enter]. Pressing [Shift Enter] will take you back to the previous reference. The line with the current reference is highlighted in Yellow.

3.2. Impact Analysis Examples

3.2.1. Impact Analysis on a TABLE

Example 1: Table « bonus » used in a **PowerBuilder Script**

bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test**
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source

```
1 string ls1, ls2
2
3 ▶ select avg(:ls1), bonus.bonus_id, sysdate, sp_plsql1(), col3
4 into :ls1, :ls2
5 ▶ from myTable2, bonus;
6
7 wf_retrieve_data()
8
9 ▶ Select col1, col2, bonus.bonus_date, sp_plsql1(bonus.date)
10 into :ls1, :ls2
11 ▶ from myTable2, bonus
12 where col2 = col3
13 and f(x) = 100;
14
```

Example 2: Table « bonus » used in a **PowerBuilder Datawindow**

bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source**

```
1 SELECT
2 ▶ bonus.bonus_amount,
3 ▶ bonus.bonus_date,
4 ▶ bonus.emp_id
5 FROM
6 ▶ bonus
```

Example 3: Table « bonus » used in a **Stored Procedure**

bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source
 - S:\WELOG\2008-11-07 15-57-44 [V]
 - Test_Package
 - proc_SQL**

```
40
41 BEGIN
42
43 SELECT a,b,c,d
44 FROM ttt;
45
46 proc2( proc3( proc1() ) );
47
48 Select
49 bonus_date,
50 sp_plsql2()
51 from
52 ▶ bonus;
53
54 END ;
55
56 Select bonus_date
57 from bonus;
58
```

3.2.2. Impact Analysis on a DB COLUMN

Exemple 1: Column « bonus.bonus_date » used in a **PowerBuilder Script**

Columns :

- bonus_amount
- bonus_date

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test**
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source

```

1  string ls1, ls2
2
3  select avg(:ls1), bonus.bonus_id, sysdat
4  into :ls1, :ls2
5  from myTable2, bonus;
6
7  wf_retrieve_data()
8
9  ▶ Select col1, col2, bonus.bonus_date, sp_
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 =col3
13 and f(x) = 100;
14
15 ls2 = ls1
16
17 ▶ Select bonus_date,sp_plsql2()
18 into :ls1, :ls2
19 from bonus;
```

Exemple 2: Column « bonus.bonus_date » used in a **PowerBuilder Datawindow**

Columns :

- bonus_amount
- bonus_date

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source**

```

1  SELECT
2     bonus.bonus_amount,
3     ▶ bonus.bonus_date,
4     bonus.emp_id
5  FROM
6     bonus
```

Exemple 3: Column « bonus.bonus_date » used in a **Stored Procedure**

Columns :

- bonus_amount
- bonus_date

Impact analysis :

- validation_pb
 - w_sql
 - d_proc3
 - data source
 - S:\VELOG\2008-11-07 15-57-44 [v]
 - Test_Package
 - proc_SQL**
 - {BEGIN SELECT a,...

```

40
41
42  BEGIN
43
44      SELECT a,b,c,d
45      FROM ttt;
46
47      proc2( proc3( p
48
49      ▶ Select
50         bonus_date,
51         sp_plsql2()
52      from
53         bonus;
54
55  END ;
56
57  ▶ Select bonus_date
58  from bonus;
```

3.2.3. Impact Analysis on a STORED PROCEDURE

Example: Impact Analysis for the stored procedure « sp_deleteemployee »:

Visual Expert will find all references to stored procedures and display a result at instruction-level:

The screenshot displays the Visual Expert interface for impact analysis. On the left, a project tree shows the following structure:

- sp_deleteemployee
- Impact analysis :
- demopfc
 - d_sp_customer_list (Data Source)
 - w_resp_employee
 - cb_delete (clicked) → SP sp_delete_employee
- C:\Program Files\Novalys\Visual Expert 6.0\Sample\Samp
 - DataManager.tsq
 - {Create procedure OrdersbyEmployee(@Employ
 - OrdersbyEmployee

The central pane shows the impact analysis results for the stored procedure `sp_deleteemployee`. On the right, the SQL code editor shows the following code:

```
111
112 --If @Emp_lname is null Or @Emp_lname = "
113 exec sp_deleteemployee @Employee_Id
114 --End
115
116 end
117 go
118
119 Create procedure Productsbycustomer( @Custon
120 as
121 Declare
122 @Cust_fname varchar(256),
123 @Cust_lname varchar(256),
124 @Prod_num NUMBER,
125
126 c1 cursor for
127 SELECT product.id Pid,
128 product.name prod name.
```

Callout boxes provide additional context:

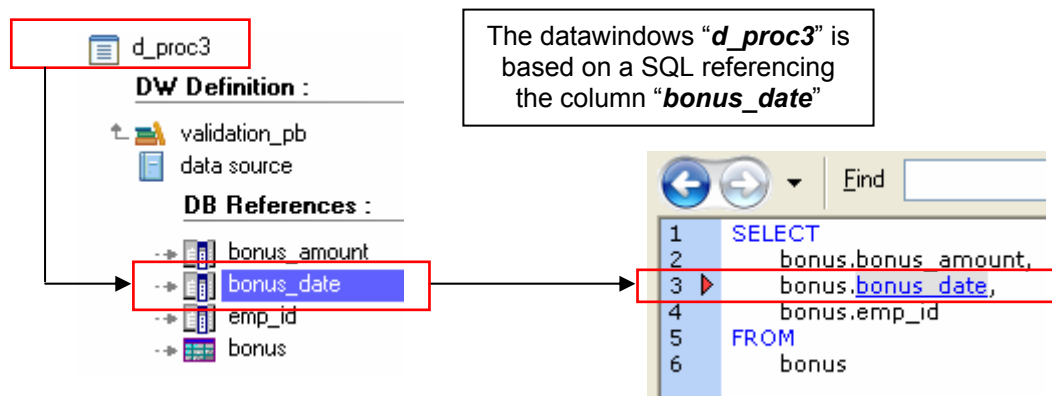
- The Stored Procedures "sp_deleteemployee" is called by a Datawindow...
- ... by a PowerBuilder event...
- ...and by another Stored Procedure.
- The procedure "OrderByEmployee" is calling "sp_deleteemployee" at line 113

4. Exploring references with Visual Expert 6.0

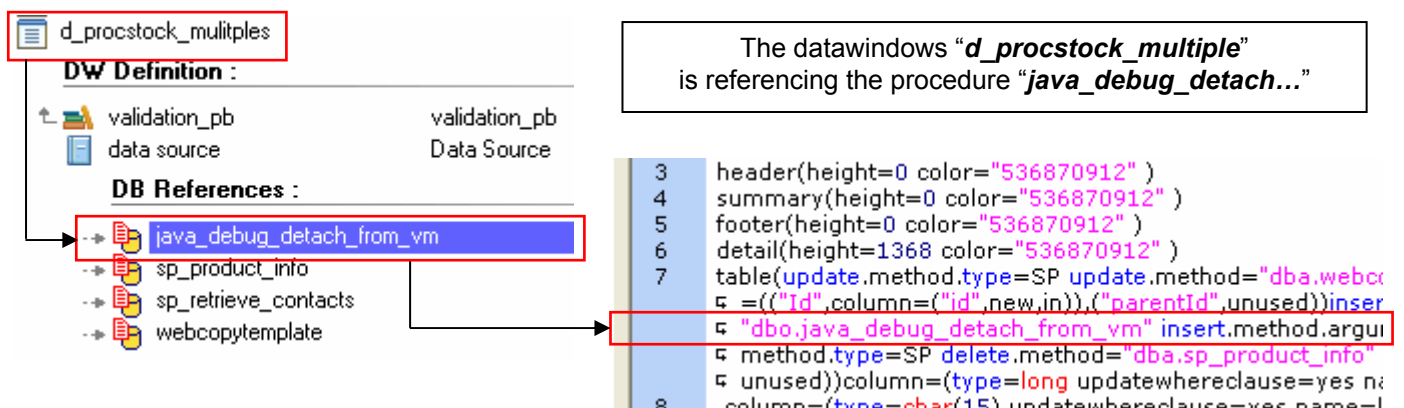
4.1. References found in PowerBuilder components

4.1.1. DataWindows

When a DataWindow is based on a SQL DataSource, the table and columns referenced are listed in the treeview and highlighted in the source:



Both « update » and « select » procedures are displayed as referenced by the DataWindow:



4.1.2. PowerScripts

All items referenced by a Script are listed in the treeview and highlighted in the code.

The event "ue_sql_test" is referencing the column "bonus_date"...

References:

- bonus_date
- bonus_id
- col1
- col2
- col3
- date
- x
- f
- sp_plsql1
- sp_plsql2
- sql sql_1
- sql sql_2
- sql sql_3
- bonus
- myTable2

```

9  ▶ Select col1, col2, bonus.bonus_date, sp_plsq
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 =col3
13 and f(x) = 100;
14
15 ls2 = ls1
16
17 ▶ Select bonus_date,sp_plsql2()
18 into :ls1, :ls2
19 from bonus;

```

...as well as the Stored Procedure "sp_plsql1"...

References:

- bonus_date
- bonus_id
- col1
- col2
- col3
- date
- x
- f
- sp_plsql1
- sp_plsql2

```

2
3  ▶ select avg(:ls1), bonus.bonus_id, sysdate,
   □ sp_plsql1(), col3
4  into :ls1, :ls2
5  from myTable2, bonus;
6
7  wf_retrieve_data()
8
9  ▶ Select col1, col2, bonus.bonus_date,
   □ sp_plsql1(bonus.date)
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 =col3
13 and f(x) = 100;

```

...and the local variable "ls1"

References:

- ls1
- ls2

```

3  ▶ select avg(:ls1), bonus.bonus_id, sysdate, sp_plsql:
4  into :ls1, :ls2
5  from myTable2, bonus;
6
7  wf_retrieve_data()
8
9  Select col1, col2, bonus.bonus_date, sp_plsql1(bonu
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 =col3
13 and f(x) = 100;
14
15 ▶ ls2 = ls1
16
17 Select bonus_date,sp_plsql2()
18 into :ls1, :ls2
19 from bonus;

```

4.1.3. RPC FUNC

An RPCFUNC declaration is another solution for PowerBuilder to use a stored procedure.

This declaration is similar to the declaration of an external dll function. The RPCFUNC keyword indicates that a stored procedure is called (and not a dll function).

Visual Expert considers RPCFUNC methods like any other method of the component:

When an RPCFUNC method or dll is selected, Visual Expert displays its declaration:

<ul style="list-style-type: none"> ☐ w_rpcfunc_test Definition : ↑ validation_pb activate clicked close open f(x) my_dll_func1 f(x) my_dll_func2 f(x) my_plsql_proc1 f(x) my_plsql_proc2 f(x) uf_method1 f(x) uf_method2 	<pre> 25 function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1 26 function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2 27 28 // déclarations de fonctions externes de DLL 29 // 30 ► function string my_dll_func1 (string toto1, string toto2) library "myDLL.d 31 function string my_dll_func2 (string toto1, string toto2) library "myDLL.d 32 33 34 35 end prototypes 36 forward prototypes 37 public function integer uf_method1 (integer p1) </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Visual Expert also displays the references to RPCFUNC or dll functions when they are selected:

<ul style="list-style-type: none"> open f(x) my_dll_func1 f(x) my_dll_func2 f(x) my_plsql_proc1 References : → sp_plsql1 f(x) my_plsql_proc2 f(x) uf_method1 f(x) uf_method2 	<pre> 20 21 type prototypes 22 23 // déclarations de procédures stockées sous forme de méthode 24 ☐ (RPCFUNC) 25 ► function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1 26 function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2 27 28 // déclarations de fonctions externes de DLL 29 // 30 function string my_dll_func1 (string toto1, string toto2) library </pre>
<ul style="list-style-type: none"> open f(x) my_dll_func1 References : → f(x) external_func1 f(x) my_dll_func2 f(x) my_plsql_proc1 f(x) my_plsql_proc2 f(x) uf_method1 f(x) uf_method2 	<pre> 24 // 25 function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1 26 function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2 27 28 // déclarations de fonctions externes de DLL 29 // 30 ► function string my_dll_func1 (string toto1, string toto2) library 31 ☐ "myDLL.dll" alias for external_func1 32 function string my_dll_func2 (string toto1, string toto2) library 33 ☐ "myDLL.dll" alias for external_func2 34 </pre>

4.2. References found in PL/SQL or T-SQL components

4.2.1. Oracle Package

A Package is related to any item referenced by the package itself or its procedures, types, etc... As a result, a Package may reference lots of items. You can list all referenced items in the treeview and select one of them: the source code will automatically highlight the references to this item.

The Package "Package_Reference" is referencing the column "column1"...

Package_Reference
References :

- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
94 PROCEDURE proc3(  
95 param1 IN table1.column1%  
96 param2 IN table1.column2%  
97 )  
98 AS  
99  
100 l_varlocal_1 table1.column1%type;  
101  
102 BEGIN  
103  
104 SELECT column1, proc1()  
105 FROM table1;
```

...as well as the Procedure "Proc2"...

Package_Reference
References :

- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
84 BEGIN  
85  
86 SELECT column1, column2 , proc1()  
87 FROM table1;  
88 proc2( proc3( proc1() ) );  
89  
90 END;
```

...and the variable "local2"...

Package_Reference
References :

- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
59 local1 := local2;  
60  
61 DECLARE  
62  
63 local3 table1.column2%TYPE;  
64  
65  
66 BEGIN  
67  
68 local1 := local2;  
69 local1 := local1 + local2;  
70 local3 := local1 + local3;  
71  
72 END;
```

4.2.2. Stored Procedures

Visual Expert can list the items referenced by a stored procedure.

This is the same concept as the Oracle Packages, with a scope limited to a stored procedure:

The screenshot displays the Visual Expert interface. On the left, a 'Content' pane shows a tree view under 'Package_Reference' with a 'Definition' section containing 'proc1'. Below it, a 'References' section lists 'column1', 'column2', 'proc1', 'proc2', 'table1', 'local3', 'local1', and 'local2'. On the right, a code editor shows the SQL code for 'PROCEDURE proc1' and 'PROCEDURE proc2'. The code for 'proc1' includes local variable declarations and a 'SELECT' statement. The code for 'proc2' includes local variable declarations and assignment statements. A status bar at the bottom indicates '48 54'.

```
42 IS
43
44 -----
45 PROCEDURE proc1
46 AS
47
48     local1 table1.column1%TYPE;
49     local2 table1.column2%TYPE;
50
51
52 BEGIN
53
54     SELECT column1, column2, proc1()
55     FROM table1;
56
57     proc2();
58
59     local1 := local2;
60
61 DECLARE
62
63     local3 table1.column2%TYPE;
64
65
66 BEGIN
67
68     local1 := local2;
69     local1 := local1 + local2;
70     local3 := local1 + local3;
71
72 END;
73
74 END;
75
76 -----
77 PROCEDURE proc2(param1 IN table1.column2
78 AS
79
80
```

4.2.3. Other DB Code items

Visual Expert also analyses Views, Types and Cursors.

It supports references to tables and columns with on the instruction %Type

4.2.4. PL/SQL %TYPE references

Visual Expert supports all %TYPE references. Parent items (Views, Types, and Cursors) and the table/column referenced. For instance:

Content : Package_Reference Definition : S:\VELOG\2008-11-07 proc1 References : -> column1 -> column2 -> proc1 -> proc2 -> table1 -> local3 -> local1 -> local2 proc2 proc3	44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	<pre>PROCEDURE proc1 AS local1 table1.column1%TYPE; local2 table1.column2%TYPE; BEGIN SELECT column1, column2, proc1() FROM table1; proc2(); local1 := local2; DECLARE local3 table1.column2%TYPE; BEGIN</pre>
Content : Package_Reference Definition : S:\VELOG\2008-11-07 proc1 proc2 proc3 References : -> column1 -> column2 -> proc1 -> proc2 -> proc3 -> table1 -> l_varlocal1_inBloc -> l_varlocal2_inBloc -> l_varlocal_1	93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114	<pre>PROCEDURE proc3(param1 IN table1.column1%TYPE, param2 IN table1.column2%TYPE) AS l_varlocal_1 table1.column1%type; BEGIN SELECT column1, proc1() FROM table1; BEGIN SELECT column1, column2, proc1() FROM table1; proc2(proc3(proc1())); END ;</pre>

4.2.5. Parameters & Local Variables

Variables & Parameters are referenced as any other DB Item. Each reference will be highlighted in the source code.

5. Source code view

5.1. Hyperlinks in the code

Hyperlinks are now available in the source code, both for methods and variables:

A click on a referenced method will display its source code (Go to definition):

```
21 tab ltab
22 integer li_nb
23
24 this.of_getparentwindow(iw_parentwindow)
25
26 ltab = getparent ()
27
28 itab_parent= ltab
29
```

```
1 ////////////////////////////////////////////////////////////////////
2 //
3 // Function: of_GetParentWindow
4 // Access: public
5 // Arguments:
6 // aw_parent The Parent window for this object
7 // Returns: Integer
8 //           1 if it succeeds and -1 if an error occu
9 //
10 // Description: Calculates the parent window of a w
11 //
12 ////////////////////////////////////////////////////////////////////
13 powerobject lpo_parent
14
15 lpo_parent = this.GetParent()
```

A click on a referenced variable will display its declaration (Go to declaration):

```
24 this.of_getparentwindow( iw_parentwindow)
25
26 ltab = getparent ()
27
28 itab_parent= ltab
29
30 li_nb = upperbound (ltab.control)
31
32 li_nb ++
33 ltab.control [li_nb] = this
34
```

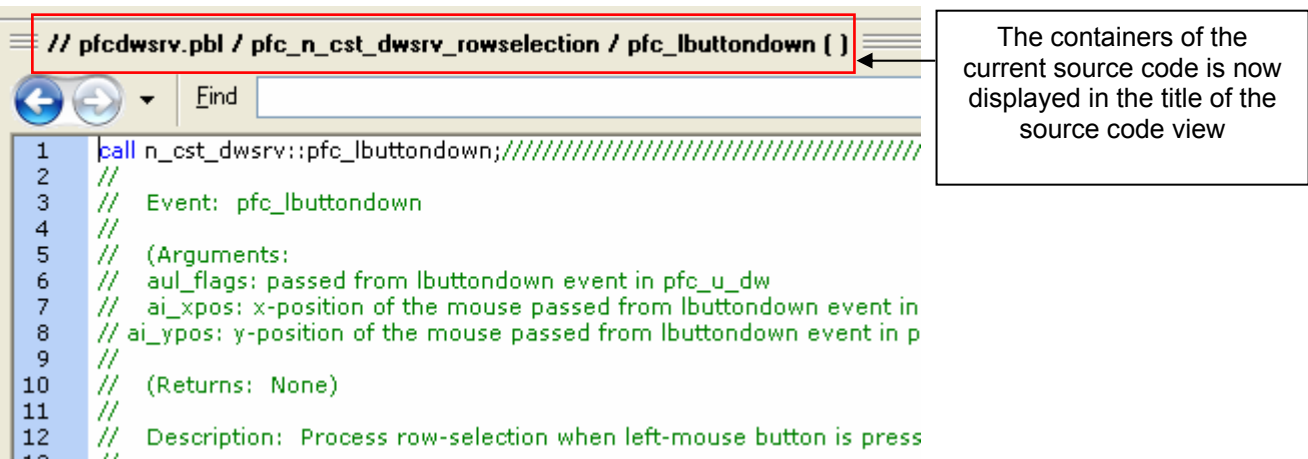
```
13
14 type variables
15 w sheet employee iw_parentwindow
16 u_tab itab_parent
17 end variables
18
```

5.2. Title of the source code view

The title of the source code view now displays the containers of the current code. It helps to understand where is located the current code.

For instance:

- //PBL/composant/contrôle/méthode
- //text file/Package/Procedure



5.3. Search in the source code view

The search feature in the source code view has been redeveloped:

- When entering a string in the “find” field, the code automatically scrolls down to display the first occurrence found.
- The number of occurrences found in this code is automatically displayed.
- Each line containing a reference is marked with a bullet
- You can type [enter]/[Shift Enter] to navigate up/down in the list of occurrences, or click on the up/down button.

The screenshot shows a source code editor with a search bar at the top containing 'sales' and '4 of 27'. The code below is a PL/SQL procedure. Several lines are highlighted with blue squares, and a line is highlighted with a yellow background. Callout boxes provide the following information:

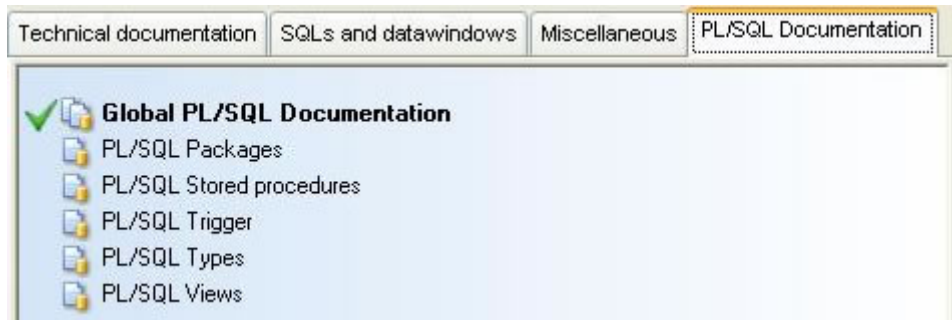
- While you type a string, the code automatically scrolls down to the first occurrence.** (Points to the search bar)
- ...and the number of occurrences found is** (Points to the search bar)
- The up/down buttons will jump to the next/previous occurrence. Same result when typing [enter/Shift enter]** (Points to the search bar)
- This Index shows all line number with an occurrence. A click on a number scroll to the** (Points to the index at the bottom)
- The occurrences currently visible in this view are selected with a blue** (Points to the blue squares on the code lines)
- Each line with an occurrence of the string is marked with this** (Points to a bullet on line 56)
- Each occurrence found is selected with a square** (Points to a blue square on line 56)
- When navigating up/down in the occurrences, the current** (Points to the search bar)

- An Index shows the number of all lines containing an occurrence of the string.

6. Technical Documentation

6.1. Technical documentation for Oracle Stored procedures

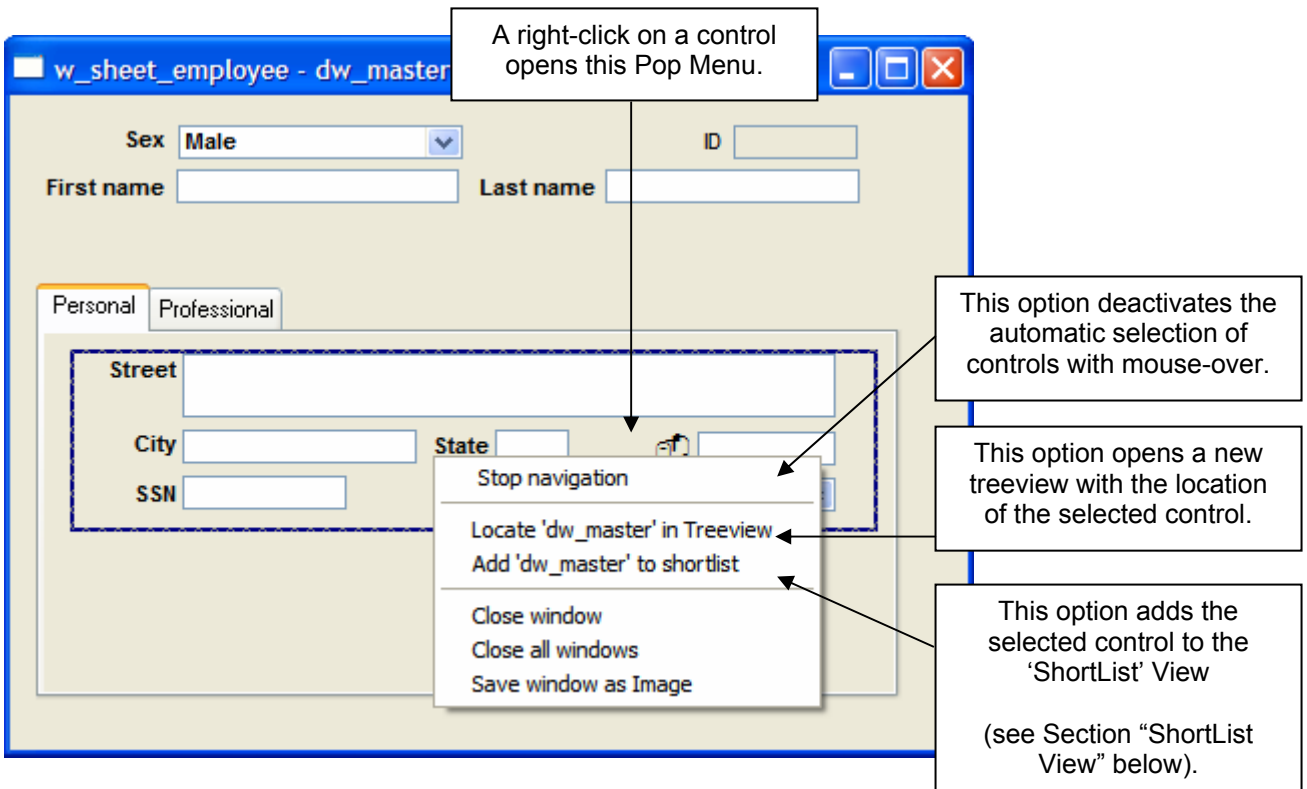
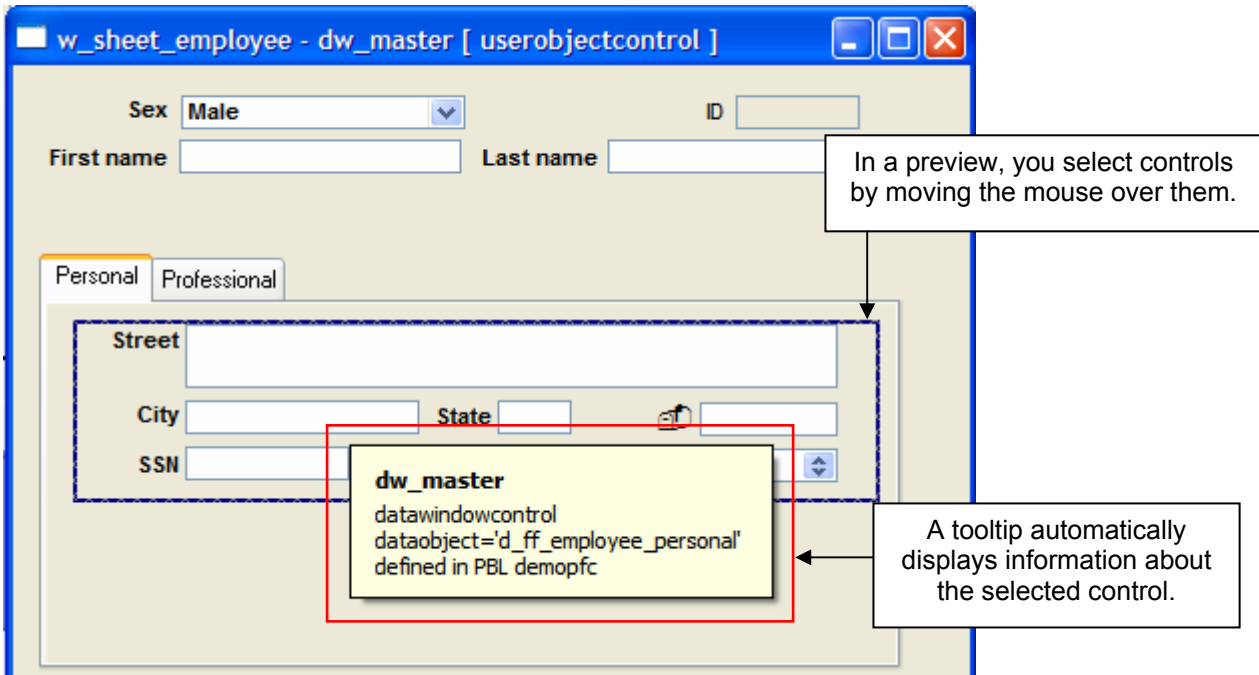
The technical documentation is now available for PL/SQL code: Packages, Stored procedures, Triggers, Types, Views.



7. Miscellaneous

7.1. Object Previews

While exploring your application, you can use the Macro "Preview" to display a graphical preview of PowerBuilder Windows, Datawindows or Visual UserObjects:



7.2. Tooltips

When you move the mouse over code item, Visual Expert displays ToolTips: They provide additional information about this item (object, method, variable, table...).

Such tooltips are available:

- For the items listed in the treeview
- For the items referenced in the source code view
- For the controls displayed in a preview (see the section *Object Preview*)

Tooltips display additional information for...

The diagram illustrates tooltip functionality in three different views of Visual Expert:

- Treeview:** A box labeled "...Items listed in the treeview" points to a treeview containing several items. The item `w_sheet_sales_order` is selected, and a tooltip is displayed over it. The tooltip text is:


```

      window
      Defined in PBL demopfc
      Inherits from w_sheet
      
```
- Source Code:** A box labeled "Items referenced in the code" points to a code editor showing a line of code: `OpenSheet(w_sheet_sales_order, Parentwindi`. A tooltip is displayed over the `w_sheet_sales_order` reference, with the same text as the treeview tooltip.


```

      window
      Defined in PBL demopfc
      Inherits from w_sheet
      
```
- Preview:** A box labeled "Controls displayed in a preview" points to a preview window showing a form with fields for ID, Date, Region, Financial code, Customer, and Salesperson. A tooltip is displayed over a control, with the text:


```

      dw_master
      datawindowcontrol
      dataobject='d_ff_sales_order' defined in PBL
      demopfc
      
```

Tooltips examples:

```

14 //
15 // Date      Version
16 //
17 // 06/07/2000  1.0  Initial version
18 //
19 ///////////////////////////////////////////////////////////////////
20
21 w_sheet_sales_order lw_window
22 n_cst
23 // Che
24 // Inherits from w_sheet
25
26 IF IsNull(ai_row) or NOT IsValid(gnv_app.inv_mru) THEN
27     Return -1
28 END IF
29

```

PowerBuilder Object:
type, ancestor and PBL

```

2  m_mymenu ll_menu
3  cb_1     ll_var_cb
4  dw_1     ll_var_dw
5  userobject ll_var_uo
6  w_menu   ll_win1
7  window   ll_win2
8
9  ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"
13
14 uo_1.cb_1.text = "ok"
15 dw_1.dataobject = "hello!"
16
17 m_menu.text="ok"

```

Nested controls:
container, ancestor and PBL

```

2  m_mymenu ll_menu
3  cb_1     ll_var_cb
4  dw_1     ll_var_dw
5  userobject ll_var_uo
6  w_menu   ll_win1
7  window   ll_win2
8
9  ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"

```

DataWindowControl:
All dataobjects used are listed (both define in PB painter and dynamically associated by code)

```

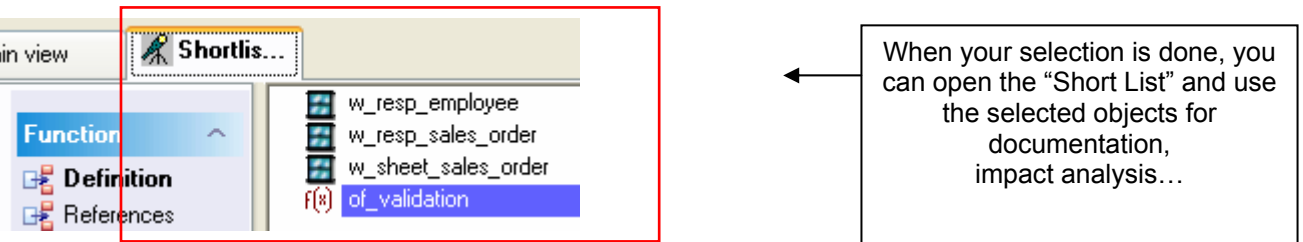
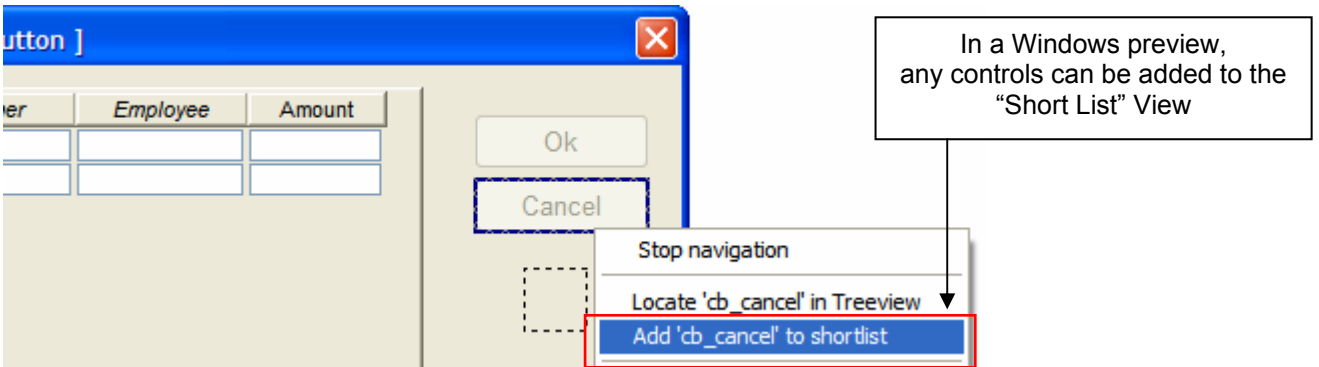
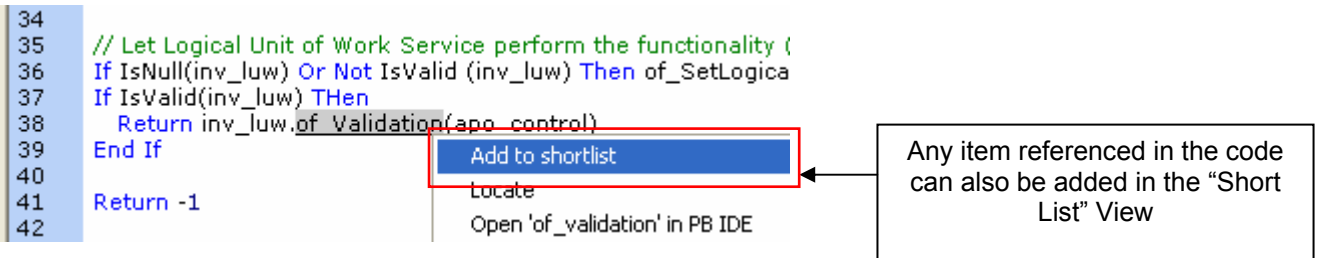
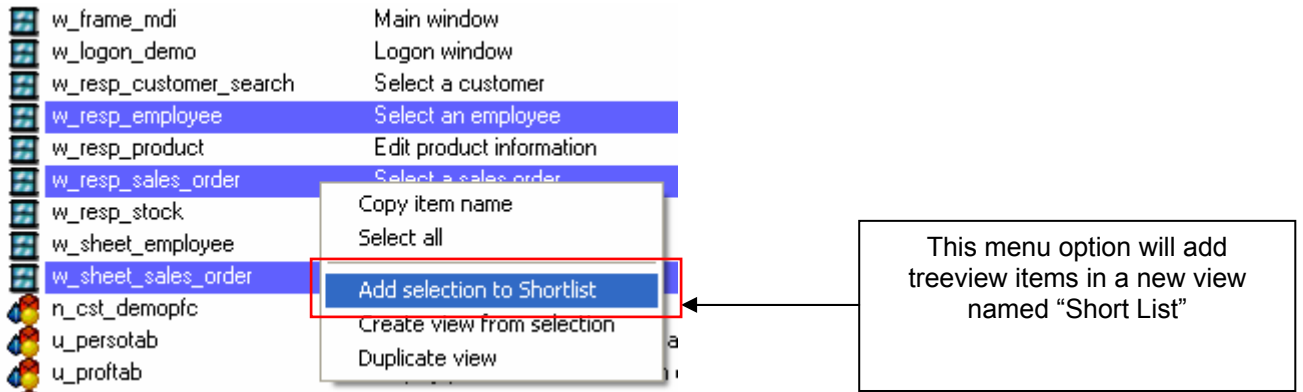
2  m_mymenu ll_menu
3  cb_1     ll_var_cb
4  dw_1     ll_var_dw
5  userobject ll_var_uo
6  w_menu   ll_win1
7  window   ll_win2
8
9  ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"
13
14 uo_1.cb_1.text = "ok"

```

Variable:
scope (local), type ('dw_1'), and information about the type.

7.3. “Short List” View

You can now select one by one code items and group them in a specific view named “ShortList”:



7.4. Technical requirements

- Polymorphism is now supported: it required Visual Expert exceptions to be declared
- Visual Expert GUI has been migrated to PB11 for future integration of .NET controls in VE
- PB11 Pre-processing supported: C# code is identified and ignored for now.
- PB 11.5 is now supported (syntax analysis, integration with PB 11.5 IDE...)