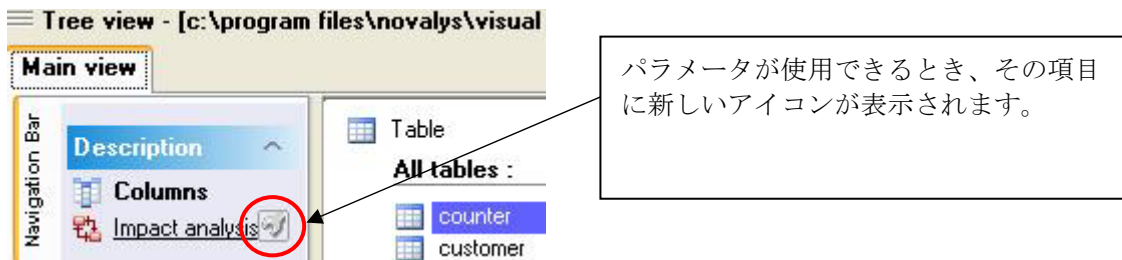


Visual Expert 6.0 新機能のご紹介

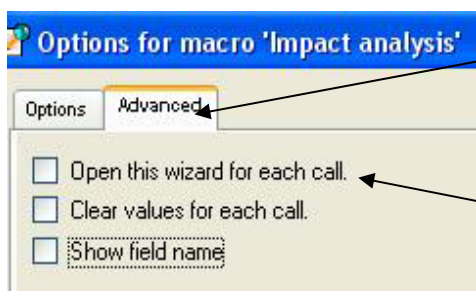
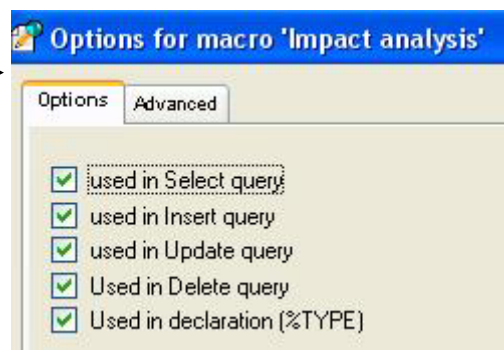
1. 新しいナビゲーションバー	2
2. 新しいマクロ	3
2.1. より少ないマクロで、多彩なパラメータで対応	3
2.2. ツリービューメニュー内のマクロのショートカット	4
2.3. ツリービュー内のオブジェクトのフィルタリング	5
2.4. 特別なデータウィンドウの検出	6
2.5. データウィンドウと DBMS のマッピング	7
2.6. POWERBUILDER ⇄ ストアドプロシージャの従属関係	8
2.7. 継承階層(ウィンドウ/ユーザオブジェクト/メニュー)	9
3. 新しいインパクトアナリシス	10
3.1. インストラクションレベルの結果表示	10
3.2. インパクトアナリシスの例	11
3.2.1. テーブルのインパクトアナリシス	11
3.2.2. DB カラムのインパクトアナリシス	12
3.2.3. ストアドプロシージャのインパクトアナリシス	13
4. VISUAL EXPERT 6.0 による参照の調査	14
4.1. POWERBUILDER コンポーネント内の参照の検出	14
4.1.1. データウィンドウ	14
4.1.2. PowerScripts	15
4.1.3. RPC FUNC	16
4.2. PL/SQL または T-SQL コンポーネント内の参照の検出	17
4.2.1. Oracle パッケージ	17
4.2.2. ストアド プロシージャ	18
4.2.3. 他の DB コード項目	19
4.2.4. PL/SQL %TYPE 参照	19
4.2.5. パラメータ & ローカル変数	19
5. 新しいソースコードビュー	20
5.1. コードのハイパーリンク	20
5.2. ソースコードビューのタイトル	21
5.3. ソースコードビュー内でのサーチ	22
6. その他	23
6.1. オブジェクトのプレビュー	23
6.2. ツールチップ	24
6.3. “ショートリスト”ビュー	26
6.4. 技術的要件	27

1.新しいナビゲーションバー

Visual Expert 6.0 では、ツリービュー マクロが提供されます。これらのいくつかはパラメータを持っています。
これらのパラメータにより、期待する結果をマクロから得ることができます。



このアイコンをクリックするとダイアログボックスが開かれ、マクロで使用するオプションを選択できます。



2番目のタブでは、マクロがどのように実行されるかを定義します。

実行したいマクロのチェックボックスを選択します。

2. 新しいマクロ

2.1. より少ないマクロで、多彩なパラメータで対応

すべてのツリービュー マクロは、再開発され再編成されました。

いくつかの標準マクロは、すべての言語（PB、PL/SQL および T-SQL）で使用可能です。

- マクロ名、マクロのコンセプトはすべての言語で同じです。
- 各マクロは、多くのニーズをカバーするためにパラメータを使用しています。
- これらのパラメータは、ツリービューで選択されるコンポーネントのタイプに依存します。

The screenshot shows the 'Component list for PowerBuilder' window for Transact-SQL. The left sidebar has a tree view with 'Definition' selected. The main area lists components grouped by language: PowerBuilder (PBL), Oracle PL/SQL, and Transact SQL. Callout boxes provide definitions for the tree view options: 'Definition' shows detailed info, 'References' shows items referenced by the selected component, 'Impact analysis' shows items referenced by the selected component, and 'Calling Hierarchy' shows the chain of methods called by other classes.

Component	Count
Component list for PowerBuilder :	
Datwindow	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	
on customer list	

定義: 選択したコンポーネントの詳細情報 (メソッド、変数、先祖、SQL クエリ、パラメータ)

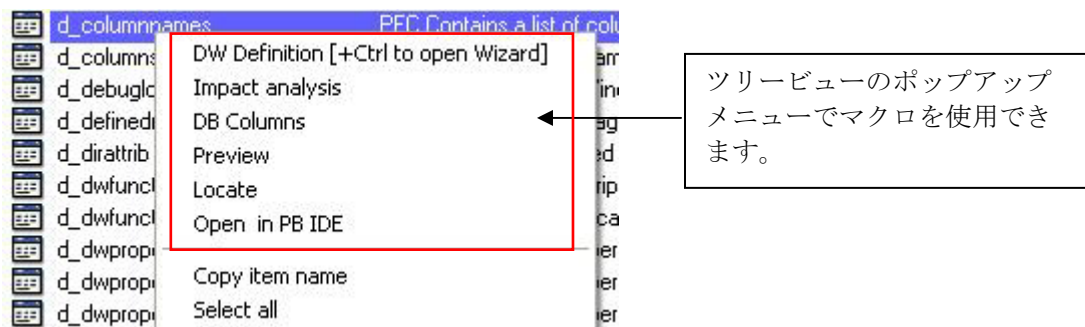
参照: 選択したコンポーネントから参照される項目がリストされます

インパクト アナリシス: 選択したコンポーネントを参照する項目がリストされます。

呼出し階層: 他のクラスを呼出しているメソッドのチェーン

2.2. ツリービューメニュー内のマクロのショートカット

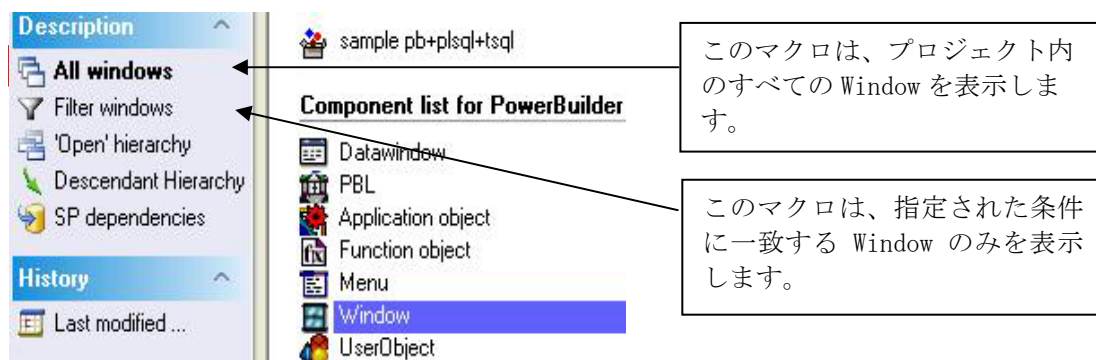
ツリービューのポップアップメニュー（項目を右クリック）には、選択された項目に適用可能なマクロが含まれています。ナビゲーションバーを使用する必要はありません。



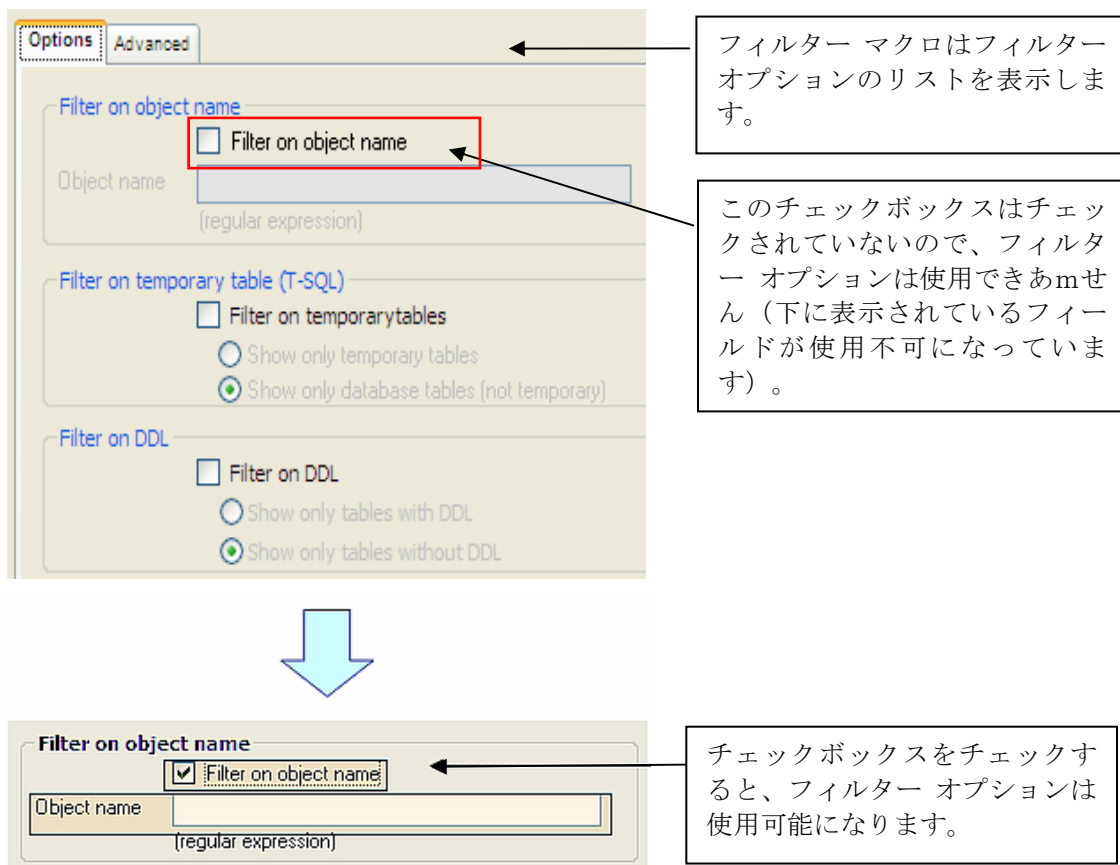
2.3. ツリービュー内のオブジェクトのフィルタリング

下図のツリービューでは、現在2つのマクロがリストされています：

- “すべて <オブジェクト>” マクロは、このタイプのすべてのオブジェクトをリストします。
このマクロはデフォルトでは、ツリービューのルート項目をダブルクリックすることにより実行されます。
- “フィルター <オブジェクト>” マクロは、各オブジェクト タイプでフィルターする多彩なパラメータを提供します。

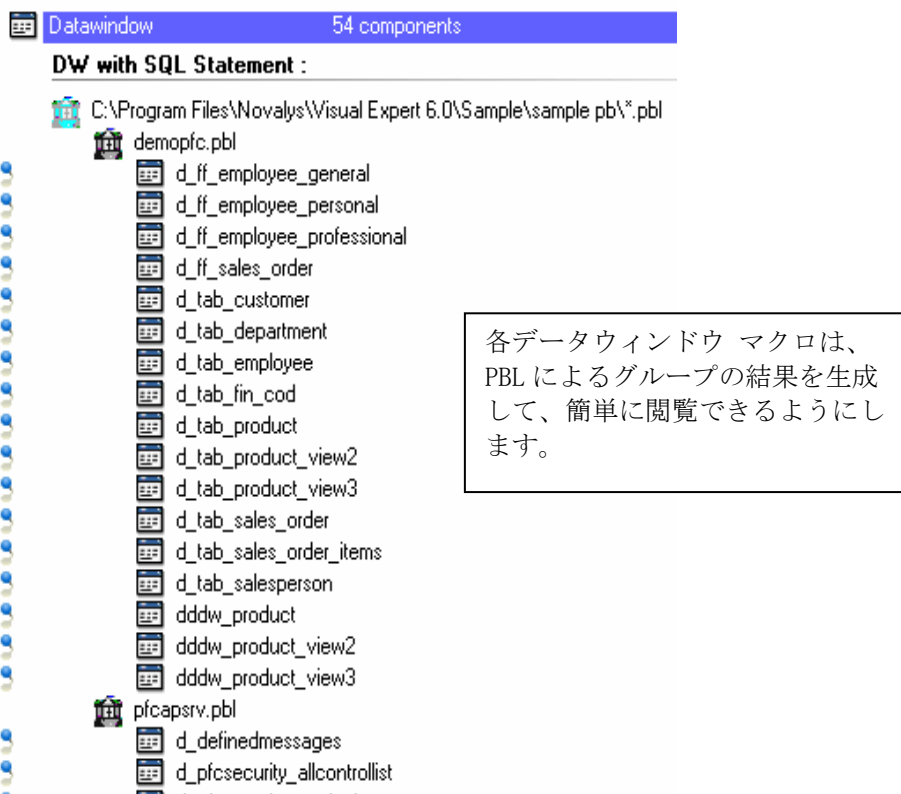
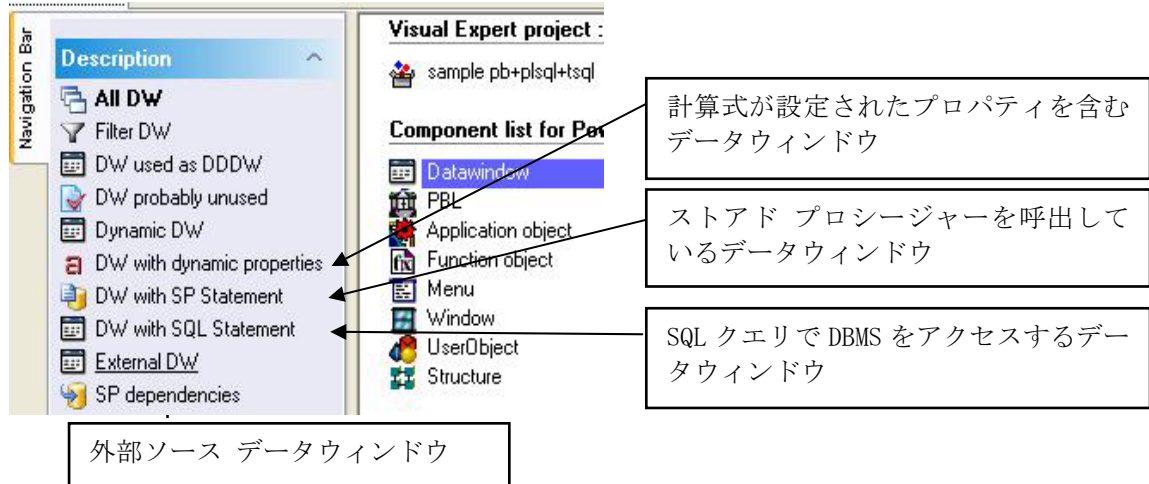


各フィルター マクロはフィルター オプションのリストを提供します。デフォルトでは、これらのオプションは有効化されていません。対応するチェックボックスをチェックして有効化します。



2.4. 特別なデータウィンドウの検出

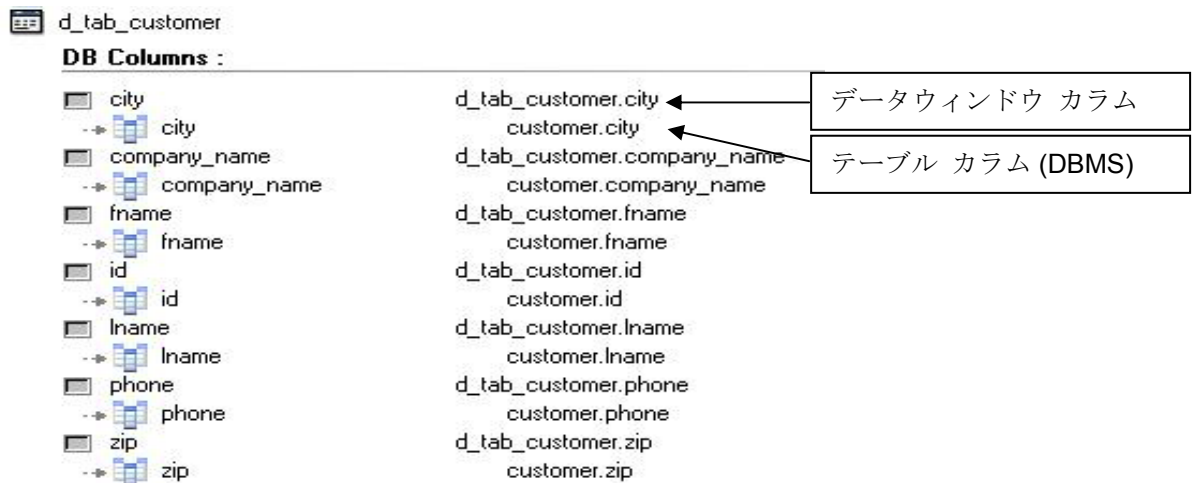
Visual Expert 6.0 では、データウィンドウのルート項目にいくつかの新しいマクロが追加されています。



2.5. データウィンドウと DBMS のマッピング

Visual Expert 6.0 では、データウィンドウに対し新たに“DB カラム”マクロが組み込まれております。

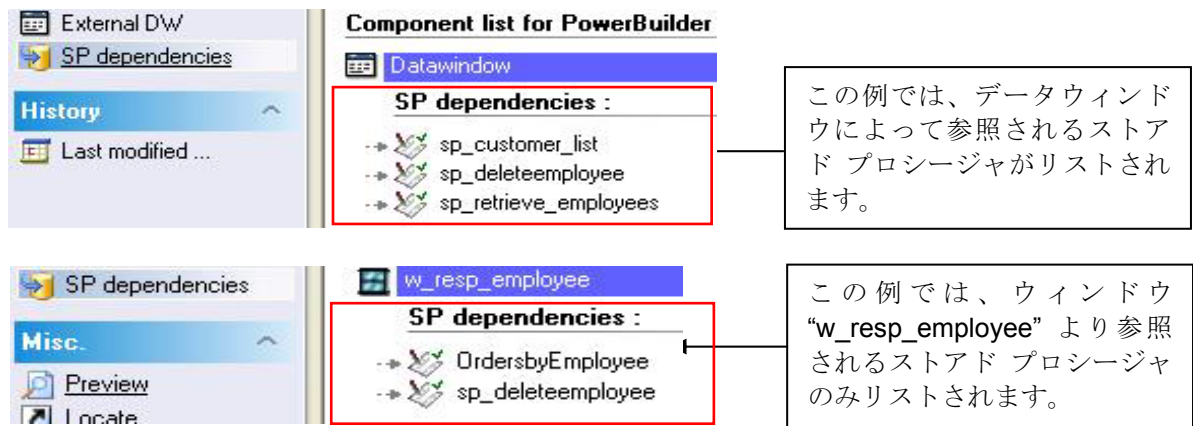
このマクロは、データウィンドウ-DBMS 間のマッピングを表示し、DBMS のどのテーブル カラムがデータウィンドウ カラムに相当するかを示します。



2.6. PowerBuilder ⇄ ストアド プロシージャの従属関係

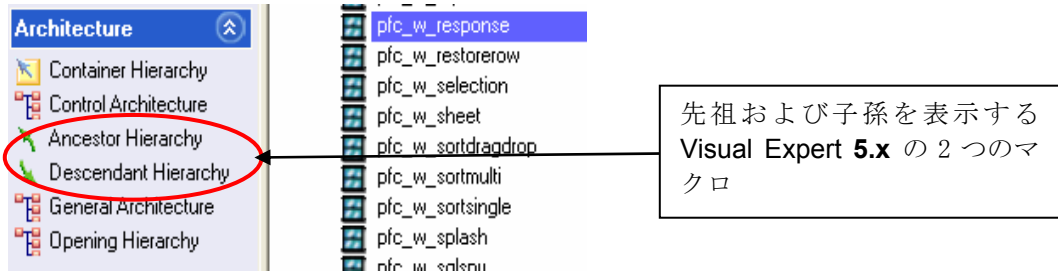
新しいマクロ « SP の従属関係 » は、PB オブジェクトにより参照されるすべてのストアド プロシージャをリストします：

- このマクロは、ツリービューに表示される各 PB コンポーネントのルート項目で使用できます。
- これはまた、ツリービュー内に表示される PB オブジェクトでも使用できます。

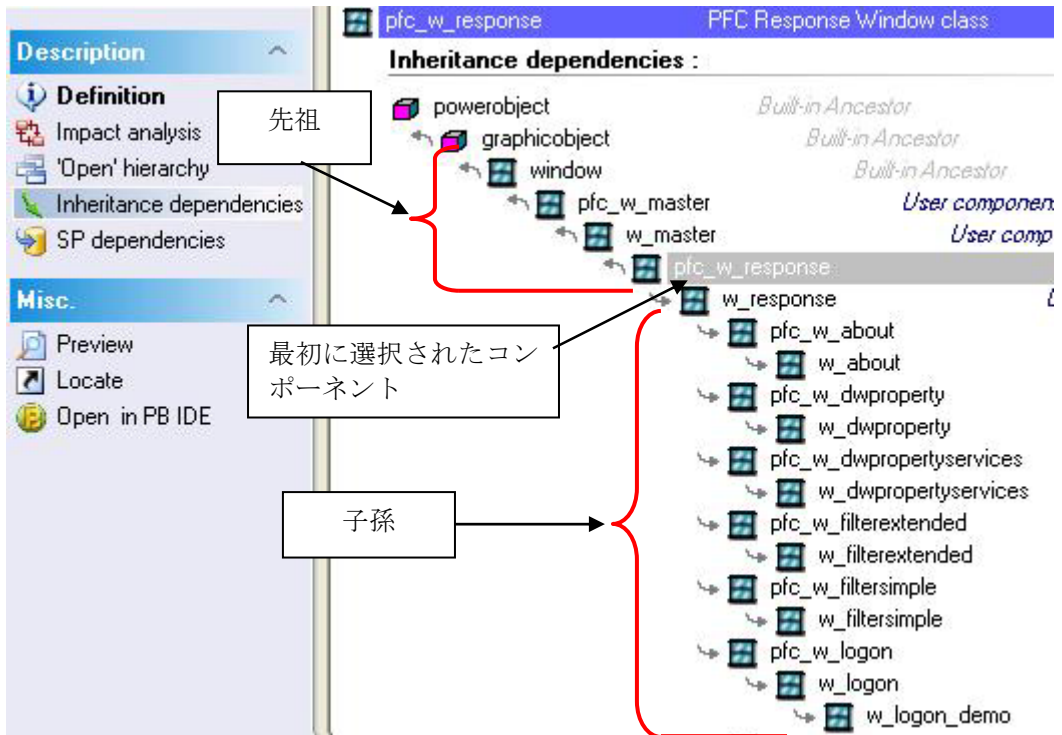


2.7. 継承階層 (ウィンドウ/ユーザオブジェクト/メニュー)

Visual Expert 5.x では、各々子孫オブジェクトと先祖オブジェクトを表示する2つのマクロを提供していました。



Visual Expert 6.0 では、1つのマクロで先祖と子孫の階層の両方を表示します：



3.新しいインパクト アナリシス

3.1. インストラクション レベルの結果表示

Visual Expert は、アプリケーション全体およびツリービューで提供されるコンポーネント レベルでのインパクト アナリシスを行うために使用されます。

Visual Expert 6.0 では、インストラクション レベルのきめの細かい結果を提供します！

このアイコンは、この項目がインパクトアナリシスの結果に属することを示します。

この記号は、このイベントが“Customer”テーブルから参照されることを示します。

これは、このコード内で参照されるオブジェクトを示し、参照オブジェクトがハイライト表示されます。

このインデックスは、参照を持つすべてのコードをリストします。この数値をクリックすると、その行が表示されるようにソースコードがスクロールダウンされます。

[Enter]を押すことにより、1つの参照から次の参照にナビゲートできます。[Shift + Enter]を押すと、前の参照に戻ります。現在、参照されている行は黄色にハイライト表示されます。

Find | references: 1 of 12

```
39      return uf_GetUpperCase(lname) ||
40      end uf_GetFullname;
41
42
43      procedure OrdersbyEmployee( Employee_I
44      is
45          Emp_fname  varchar2;
46          Emp_lname  varchar2;
47
48          cursor c1 is
49              SELECT sales_order.id Oid,
50                     sales_order.order_date Od
51                     DECODE(sales_order.fin_co
52                             'r1', 'Revenue', 'e1',
53                             sales_order.region Oregio
54                             customer.fname Cfname,
55                             customer.lname Clname,
56                             sum (sales_order_items.qu
57
58          FROM customer,
59          employee,
60          product,
61          sales_order_items,
62          sales_order
63          WHERE ( employee.emp_id = Employee
64                ( employee.emp_id = sales
65                ( customer.id = sales_ord
66                ( sales_order_items.id =
67                (product.id = sales_order
68          GROUP BY Oid,Odate,
69                  sales_order.fin_code_id,
```

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

3.2. インパクト アナリシスの例

3.2.1. テーブルのインパクト アナリシス

例 1 : PowerBuilder Script 内で使用されている « bonus » テーブル

The screenshot shows the Impact analysis tool interface. On the left, a tree view under 'validation_pb' shows 'w_sql' expanded, with 'ue_sql_test' selected. On the right, a code editor displays the following SQL script:

```
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
10 Select col1, col2, bonus.bonus_date, sp_plsql1(bonus.date)
11 into :ls1, :ls2
12 from myTable2, bonus
13 where col2 =col3
14 and f(x) = 100;
```

例 2 : データウィンドウで使用されている « bonus » テーブル

The screenshot shows the Impact analysis tool interface. On the left, a tree view under 'validation_pb' shows 'w_sql' expanded, with 'data source' selected. On the right, a code editor displays the following SQL query:

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

例 3 : ストアド プロシージャで使用されている « bonus » テーブル

The screenshot shows the Impact analysis tool interface. On the left, a tree view under 'validation_pb' shows 'w_sql' expanded, with 'proc_SQL' selected. On the right, a code editor displays the following stored procedure code:

```
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. DB カラムのインパクト アナリシス

例 1 : PowerBuilder Script で使用されている « bonus.bonus_date »カラム

The screenshot shows the 'bonus' object's 'Impact analysis' window. Under 'Columns', 'bonus_date' is listed as impacted. The 'ue_sql_test' script is selected in the 'Impact analysis' tree. The script editor shows the following code:

```

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;

```

例 2 : データウィンドウで使用されている « bonus.bonus_date » カラム

The screenshot shows the 'bonus' object's 'Impact analysis' window. Under 'Columns', 'bonus_date' is listed as impacted. The 'data source' is selected in the 'Impact analysis' tree. The script editor shows the following code:

```

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

```

例 3 : ストアド プロシージャで使用されている « bonus.bonus_date » カラム

The screenshot shows the 'bonus' object's 'Impact analysis' window. Under 'Columns', 'bonus_date' is listed as impacted. The 'proc_SQL' script is selected in the 'Impact analysis' tree. The script editor shows the following code:

```

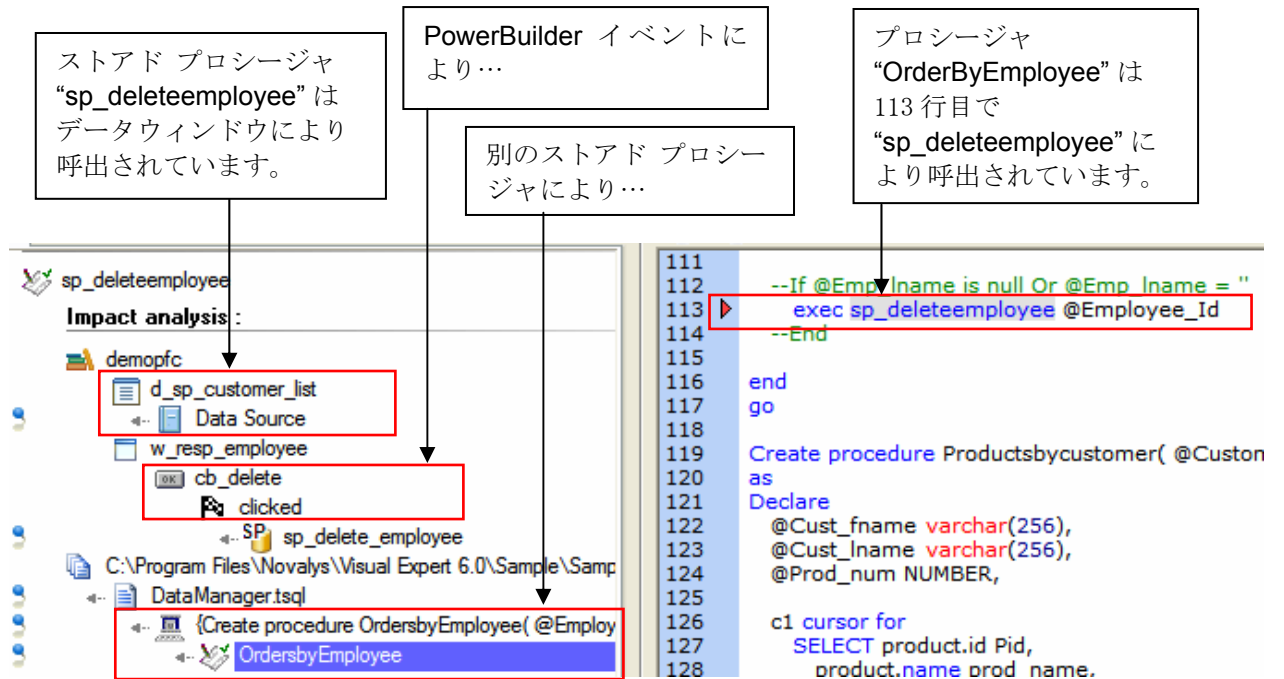
40
41 BEGIN
42
43 SELECT a,b,c,d
44 FROM ttt;
45
46 proc2( proc3( p
47
48 Select
49 bonus_date,
50 sp_plsql2()
51 from
52 bonus;
53
54 END ;
55
56 Select bonus_date
57 from bonus;
58

```

3.2.3. ストアド プロシージャのインパクト アナリシス

例：ストアド プロシージャ « `sp_deleteemployee` » のインパクト アナリシス

Visual Expert はすべてのストアド プロシージャの参照を検出しインストラクション レベルの結果を表示します。



4. Visual Expert 6.0 による参照の調査

4.1. PowerBuilder コンポーネント内の参照の検出

4.1.1. データウィンドウ

SQL データソースに基づくデータウィンドウでは、参照されるテーブルとカラムはツリービュー内にリストされ、ソース ビューではハイライト表示されます。

データウィンドウ **"d_proc3"** は **"bonus_date"**カラムを参照しています。

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

« update » と « select » の両方のプロシージャはデータウィンドウによる参照として表示されます：

データウィンドウ **"d_procstock_multiple"** は、プロシージャ **"java_debug_detach..."** から参照されます。

```
3 header(height=0 color="536870912" )
4 summary(height=0 color="536870912" )
5 footer(height=0 color="536870912" )
6 detail(height=1368 color="536870912" )
7 table(update.method.type=SP update.method="dba.webco
8 =(("Id",column=("id",new,in)),("parentId",unused))insert
9 "dbo.java_debug_detach_from_vm" insert.method.argu
10 method.type=SP delete.method="dba.sp_product_info"
11 unused))column=(type=long updatewhereclause=yes n
12 column=(type=char(15) updatewhereclause=yes name=
```

4.1.2. PowerScripts

スクリプトによって参照されるすべての項目は、ツリービュー内にリストされ、コードビューでハイライト表示されます。

ue_sql_test

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

“ue_sql_test” イベントは “bonus_date” カラムを参照しています。 ...

```

8
9  Select col1, col2, bonus.bonus_date, sp_pls
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;

```

ue_sql_test

References :

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

...同様にストアード プロシージャ “sp_plsql1” を参照しています。 ...

```

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

```

さらにローカル変数 “ls1” を参照しています。

ue_sql

ue_sql_test

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(bon
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

RPCFUNC の宣言は PowerBuilder がストアード プロシージャを使用するソリューションの 1 つです。

この宣言は、外部 DLL 関数の宣言に似ています。キーワード RPCFUNC は DLL 関数では無くストアード プロシージャが呼出されることを示します。

Visual Expert は、他のコンポーネントのメソッドと同様、RPCFUNC メソッドも考慮しております。

RPCFUNC メソッドや DLL が選択されると、Visual Expert はその宣言を表示します：

```
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.dll"
31 function string my_dll_func2 ( string toto1, string toto2) library "myDLL.dll"
32
33
34
35 end prototypes
36 forward prototypes
37 public function integer uf_method1 (integer p1)
```

Visual Expert はまた RPCFUNC または DLL が選択されたとき、それらの参照を表示します：

```
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
```



```
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
```

4.2. PL/SQL または T-SQL コンポーネント内の参照の検出

4.2.1. Oracle パッケージ

パッケージでは関連事項として、そのパッケージによって参照される項目またはそのプロシージャ、タイプなどがあります。その結果として、パッケージでは多くの参照項目があります。

Visual Expert ではツリービュー内のすべての参照項目をリストし、その中から1つを選択できます。それによりソースコードはその項目への参照を自動的にハイライト表示します。

Package_Reference

References :

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

パッケージ“**Package_Reference**”はカラム“**column1**”を参照しています。

```
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;
```

Package_Reference

References :

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

パッケージ“**Package_Reference**”は同様にプロシージャ“**Proc2**”を参照しています。

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

Package_Reference

References :

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

パッケージ“**Package_Reference**”は同様に変数“**local2**”を参照しています。

```
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. ストアド プロシージャ

Visual Expert は、ストアド プロシージャによって参照される項目をリストできます。これは Oracle Package と同じコンセプトで、スコープはストアド プロシージャに制限されます：

The screenshot displays the Visual Expert interface. On the left, a 'Content' pane shows a tree view under 'Package_Reference'. The 'Definition' section lists 'proc1'. The 'References' section lists 'column1', 'column2', 'proc1', 'proc2', 'table1', 'local3', 'local1', and 'local2'. Below these are 'proc2' and 'proc3'. The main editor shows the SQL code for 'PROCEDURE proc1' and 'PROCEDURE proc2'. The code for 'proc1' includes declarations for 'local1' and 'local2', a 'BEGIN' block with a 'SELECT' statement, and an 'END;' statement. The code for 'proc2' includes a declaration for 'local3', a 'BEGIN' block with three assignment statements, and an 'END;' statement. The status bar at the bottom shows '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. 他の DB コード項目

Visual Expert はまた、ビュー、タイプ、およびカーソルを解析します。
%Type 命令を使用して、テーブルとカラムへの参照をサポートします。

4.2.4. PL/SQL %TYPE 参照

Visual Expert は、すべての %TYPE 参照をサポートします。親項目（ビュー、タイプ、およびカーソル）とテーブル/カラムを参照します。

例：

The screenshot displays two examples of PL/SQL code with %TYPE references. In the first example, 'proc1', the references 'table1.column1%TYPE' and 'table1.column2%TYPE' are circled in red. In the second example, 'proc3', the references 'table1.column1%TYPE' and 'table1.column2%TYPE' in the parameter list are circled in red. The interface includes a left-hand pane with a tree view of database objects and a main pane showing the code with line numbers.

```
Content : 44  
45  
46 Package_Reference  
47  
48 Definition :  
49 S:\WELOG\2008-11-07  
50 proc1  
51  
52 References :  
53  
54 column1  
55 column2  
56 proc1  
57 proc2  
58 table1  
59 local3  
60 local1  
61 local2  
62  
63 proc2  
64 proc3  
65  
66  
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
```

```
Content : 93  
94  
95 Package_Reference  
96  
97 Definition :  
98 S:\WELOG\2008-11-07  
99 proc1  
100 proc2  
101 proc3  
102  
103 References :  
104  
105 column1  
106 column2  
107 proc1  
108 proc2  
109 proc3  
110 table1  
111 l_varlocal1_inBloc  
112 l_varlocal2_inBloc  
113 l_varlocal_1  
114  
115  
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 ;
```

4.2.5. パラメータ & ローカル変数

パラメータと変数は、他の DB 項目として参照されます。各参照はコード ビューでハイライト表示されます。

5. 新しいソースコード ビュー

5.1. コードのハイパーリンク

ソースコード ビューでは、メソッドと変数の両方にハイパーリンクが使用されています：

参照されるメソッドをクリックすると、そのソースコードが表示されます：

```
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()
```

参照される変数をクリックすると、その宣言が表示されます：

```
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
```

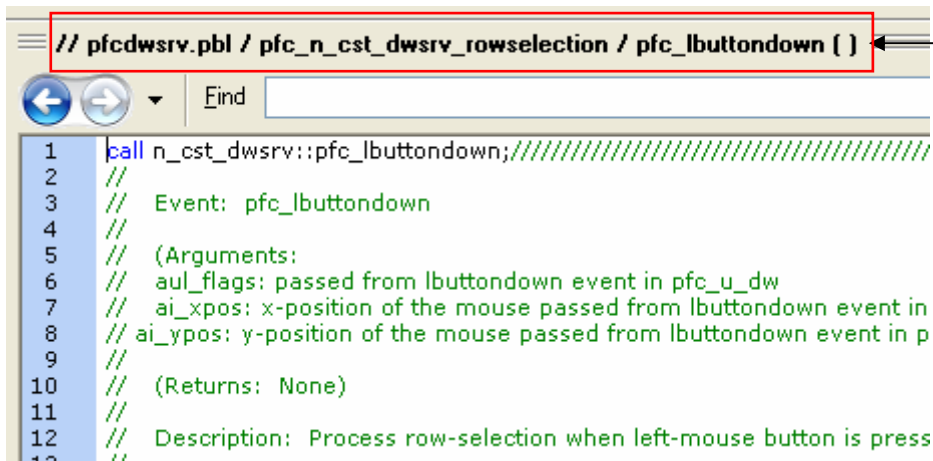
```
13
14 type variables
15 w_sheet_employee iw_parentwindow
16 u_tab itab_parent
17 end variables
18
```

5.2. ソースコード ビューのタイトル

ソースコード ビューのタイトルには、現行コードのコンテナが表示されます。それにより現行コードがどこに配置されているかを理解できます。

例：

- `/{PBL}/コンポーネント/{コントロール}/メソッド`
- `/{テキスト ファイル}/パッケージ/{プロシージャ}`



```
// pfc_dwsrv.pbl / pfc_n_cst_dwsrv_rowselection / pfc_lbuttondown ( )
1  call n_cst_dwsrv::pfc_lbuttondown;////////////////////////////////////
2  //
3  //  Event: pfc_lbuttondown
4  //
5  //  (Arguments:
6  //  aul_flags: passed from lbuttondown event in pfc_u_dw
7  //  ai_xpos: x-position of the mouse passed from lbuttondown event in
8  //  ai_ypos: y-position of the mouse passed from lbuttondown event in p
9  //
10 //  (Returns: None)
11 //
12 //  Description: Process row-selection when left-mouse button is press
13 //
```

現行ソースコードのコンテナはソースコード ビューのタイトルに表示されます。

5.3. ソースコード ビュー内でのサーチ

ソースコード ビュー内でのサーチ機能は再開発されました：

- サーチする文字列を入力すると、コードは自動的にスクロール ダウンされ最初に検出した項目を表示します。
- このコードでの検出数が自動的に表示されます。
- 参照を含む各行は、緑色の○印でマークされます。
- **[enter]/[Shift Enter]** を入力するか、またはアップ/ダウン ボタンをクリックして、検出された項目のリストを上下にナビゲートできます。
- インデックスは、検出された項目のすべての行数を示します。

文字列を入力している間、最初の項目が検出されるまでコードはスクロール ダウンされます。

サーチの結果、検出数が表示されます。

アップ/ダウン ボタンで検出項目の前/後にジャンプできます。
[enter/Shift enter]の入力も同様です。

```
43 procedure OrdersbyEmployee( Employee_Id in NUMBER )
44 is
45     Emp_fname varchar2;
46     Emp_lname varchar2;
47
48     cursor c1 is
49     SELECT sales_order.id Oid,
50            sales_order.order_date Odate,
51            DECODE(sales_order.fin_code_id,
52                  'r1','Revenue','e1','Expense'),
53            sales_order.region Oregion,
54            customer.fname Cfname,
55            customer.lname Clname,
56            sum (sales_order_items.quantity*unit_price) Oa
57     FROM customer,
58            employee,
59            product,
60            sales_order_items,
61            sales_order
62     WHERE ( employee.emp_id = Employee_Id ) and
```

ビューの中に表示されている検出項目は、青色の枠で囲まれます。

このインデックスは検出された項目の行数を示します。この数値をクリックすると、該当する行にスクロールできます。

現行行はハイライト表示されます。

各検出項目は四角で囲まれます。

検出された項目にはこのマークが付けられます。

6. その他

6.1. オブジェクトのプレビュー

アプリケーションを調べている間、PowerBuilder ウィンドウ、データウィンドウまたはビジュアル ユーザオブジェクトのプレビューを表示するために“プレビュー”マクロを使用できます：

The first screenshot shows a window titled "w_sheet_employee - dw_master [userobjectcontrol]". It contains a form with fields for Sex (Male), ID, First name, Last name, and tabs for Personal and Professional. A red box highlights a control named "dw_master" with a tooltip that reads: "dw_master", "datawindowcontrol", "dataobject='d_ff_employee_personal'", "defined in PBL demopfc".

The second screenshot shows the same window with a context menu open over the "dw_master" control. The menu items are: "Stop navigation", "Locate 'dw_master' in Treeview", "Add 'dw_master' to shortlist", "Close window", "Close all windows", and "Save window as Image".

Annotations for the first screenshot:

- プレビューでは、マウスによってコントロールを選択します。
- ツールチップは選択したコントロールの情報を自動的に表示します。

Annotations for the second screenshot:

- コントロールを右クリックすると、ポップアップメニューが表示されます。
- このオプションは、マウス上のコントロールの自動選択を無効にします。
- このオプションは新しいツリービューを開き、選択されたコントロールを配置します。
- このオプションは選択されたコントロールを 'ShortList' ビューに追加します。 "ShortList View" については、次のセクションを参照してください。

6.2. ツールチップ

マウスがコード項目上にあるとき、Visual Expert はツールチップを表示します：

ツールチップはその項目（オブジェクト、メソッド、変数、テーブルなど）についての付加情報を提供します。

ツールチップでは次のことが可能です：

- ツリービュー内にリストされる項目に適用されます。
- ソースコード ビュー内で参照される項目に適用されます。
- プレビュー内に表示されるコントロールに適用されます（「オブジェクトのプレビュー」セクションを参照してください）。

ツールチップは付加情報を表示します。

ツリービュー内にリストされた項目のツールチップ

<input type="checkbox"/>	w_resp_sales_order	Select a sales order
<input type="checkbox"/>	w_resp_stock	Edit product informatio
<input type="checkbox"/>	w_sheet_employee	Edit employee informat
<input type="checkbox"/>	w_sheet_sales_order	Edit sales order inform
<input type="checkbox"/>	n_cst_demopfc	Application manager
<input type="checkbox"/>	u_persotab	personal inform
<input type="checkbox"/>	u_proftab	professional in
<input type="checkbox"/>	u_tabemployee	Tab object displaying

ソースコード ビュー内で参照される項目のツールチップ

```

18
19 ///////////////////////////////////////////////////////////////////
20
21 Message.of_setstringparm ( "create" )
22
23 OpenSheet(w_sheet_sales_order, Parentwindi
24

```

プレビューに表示されたコントロールのツールチップ

dw_master
datawindowcontrol
dataobject='d_ff_sales_order' defined in PBL demopfc

The screenshot shows a data entry form with fields for ID, Date (11/06/2009), Region, Financial code, Customer, and Salesperson. Below these is a table with columns ID, Product, and Description. A dropdown menu is open for the Product field, and a tooltip is displayed over it, showing the object name 'dw_master' and its definition: 'datawindowcontrol dataobject='d_ff_sales_order' defined in PBL demopfc'.

ツールチップの例：

```

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 // Defined in PBL demopfc.pbl
25 // Inherits from w_sheet
26 IF IsNull(ai_row) or NOT IsValid(gnv_app.inv_mru) THEN
27     Return -1
28 END IF
29

```

PowerBuilder オブジェクト：
タイプ、先祖および 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"

```

ネストされたコントロール：
コンテナ、先祖、および 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!"

```

データウィンドウ コントロール：
使用されているすべてのデータオブジェクトリストされます（PB ペインタで定義されたものおよびコードで動的に関連付けが行われたもの）。

```

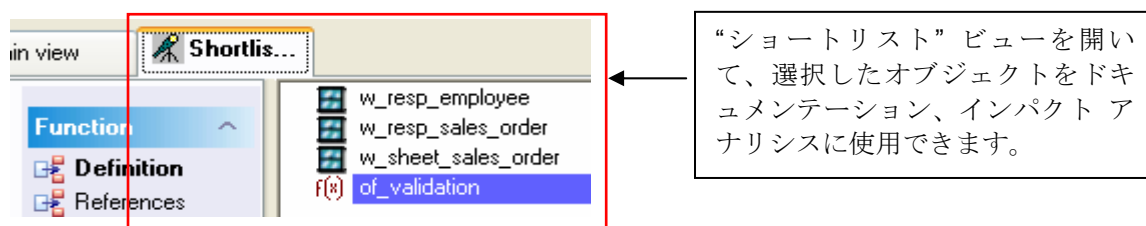
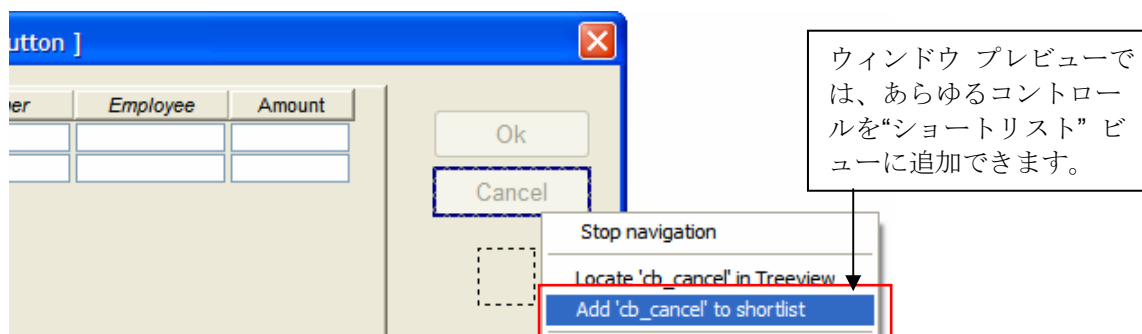
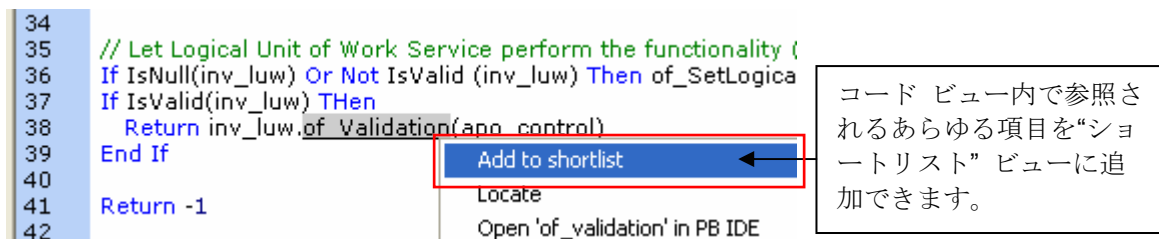
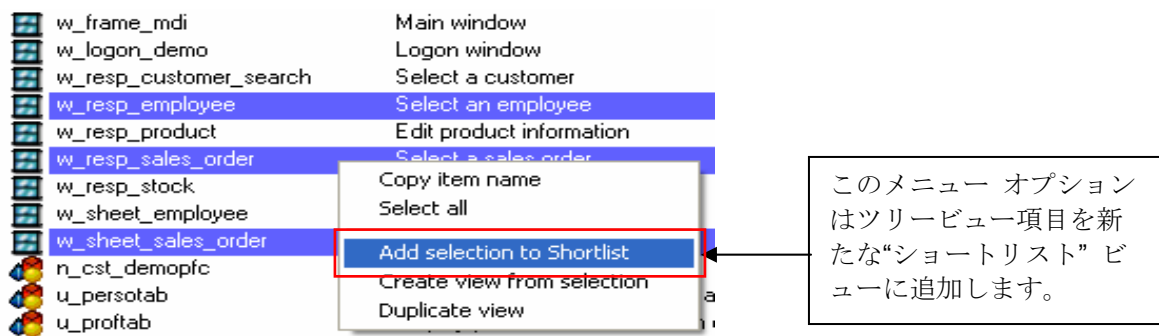
2 m_mymenu ll_menu
3 cb_1     ll_var_cb
4 dw_1     ll_var_dw
5 userobject ll_var_uo
6 w_menu   ll_win1
7 window   ll_win2
8
9 ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"
13
14 uo_1.cb_1.text = "ok"

```

変数：
スコープ（ローカル）、タイプ（'dw_1'）、およびタイプに関する情報

6.3. “ショートリスト” ビュー

Visual Expert 6.0 では、コード項目を1つずつ選択して、それを特定のビューにグルーピングできます。そのビューを“ショートリスト”ビューといいます：



6.4. 技術的要件

- Visual Expert6.0 では多相性をサポートします。これは、Visual Expert の例外として宣言される必要があります。
- Visual Expert の GUI は、将来において.NET コントロールを VE に統合するため、PB11 に移行されています。
- PB11 プリープロセッシングがサポートされます：C# code は識別されますが現在は無視されます。
- PB 11.5 がサポートされます。(構文解析、PB 11.5 IDE との統合)