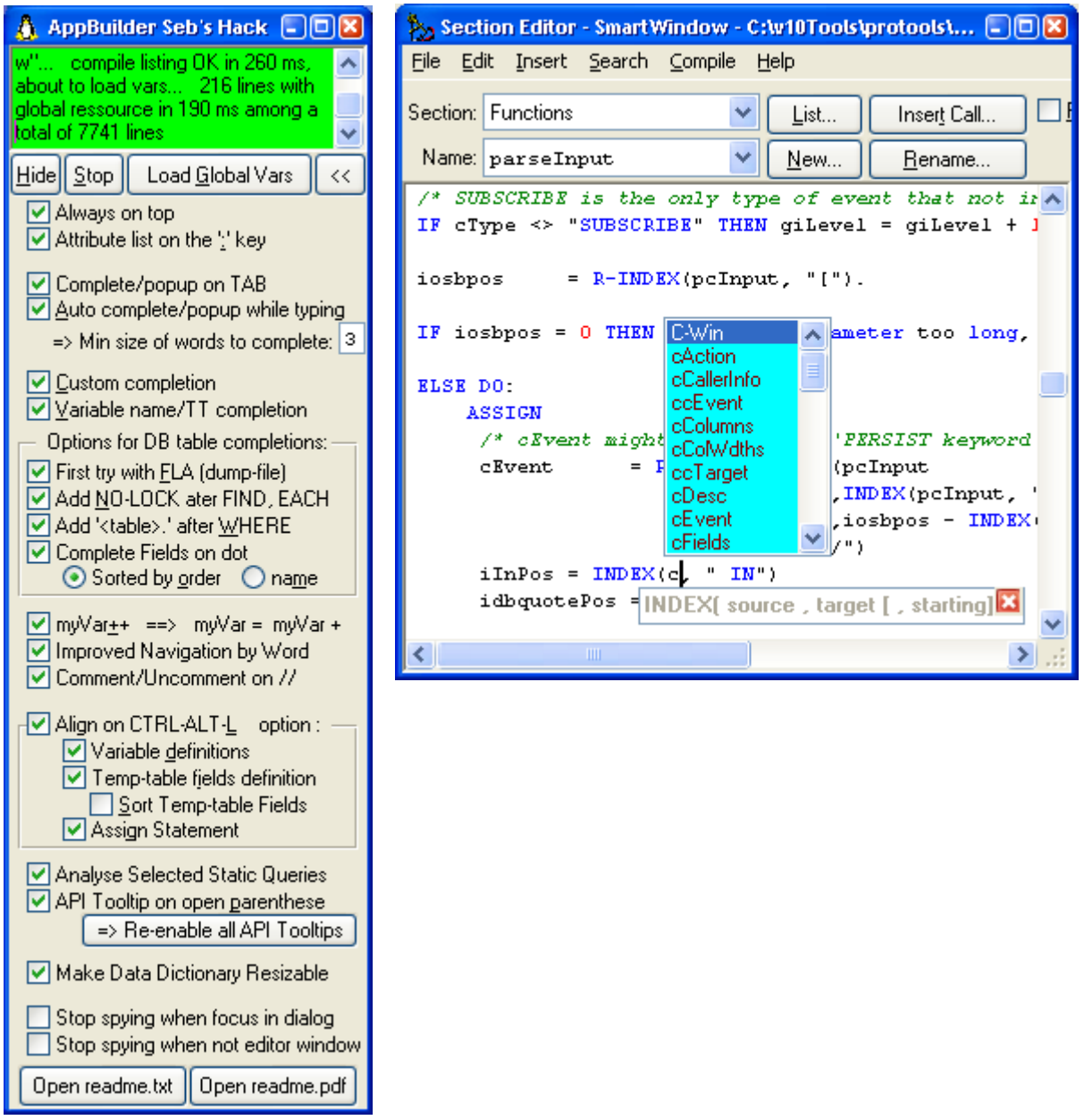
ABHack by Sébastien Lacroix
November 2006, revised on 16-Aug-2007

Nowadays, an Application Development Environment without auto-completion is a kind of torture. So hack it or die...



Note the number of features has been multiplied by about 4 since Nov 2006, so this picture is quite obsolete

ABHack is a kind of demon 4GL procedure to run with the PERSISTENT option in a development session. Its strongest points are:

1. **It does not require any recompilation of the Application Development Environment** (AppBuilder, Procedure Editor...) **nor any OCX setup**. One just need to launch it, ideal for a consultant on site.
2. It is made in pure 4GL (very few simple win32 API's, and a few PSTimers)
3. Source code is given
4. No need to pre-tag a database schema. Everything is found on the fly

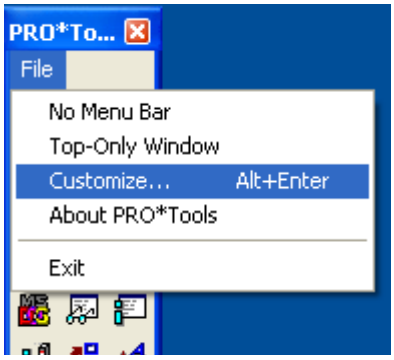
This tool can spy the activity in a source-code procedure editor to achieve auto completion for local or global variables, attributes/methods, buffers, temp-table, table and fields names, block labels.

It includes many other features like a query analyzer for a selected text, an improved navigation by words, ++ insertion, quick comment out, ability to make the data dictionary resizable, a FireFox alike search facility, a Section Outline that also reports buffer usage and their scopes...

ABHack has been tested and used fine with Version 9.1E and with OE 10.1B02. It should work with any version OE 10 and 9.1D+. It is probably possible to adapt few parts of the code to make it work with older versions of 9.1 by refining the parts that use attribute chaining (use temporary variables instead of `hObj1:hObj2:SOME-ATTRIBUTE`)

Steps to launch it:

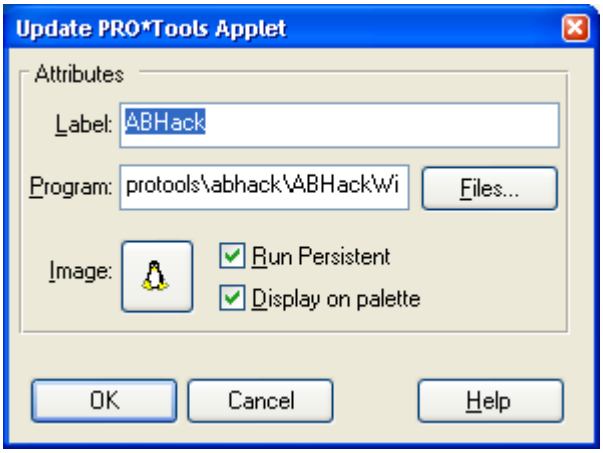
0. Have a 'protools' sub-directory in your working directory (or in a directory of your propath)
   Note : The point is to have the **parent** directory of protools in your propath, do not the protools directory itself
1. Create an 'abhack' sub-directory in this protools directory
2. Unzip the attached zip file into this abhack directory (all run's are **relative** to protools/abhack/)
3. Add a button on the protools palette (right-click on it -> menu-bar -> custom -> add button):



Note : one could just Alt-Enter on the protools palette...

Then add ABHack as defined and illustrated bellow:

- procedure: protools/abhack/ABHackWin.w
- picture: the little penguin protools/abhack/tux.bmp (do not choose the .ico file, but the .bmp)



Just click on the penguin. A little green tick comes on it to show it is running. Clicking another time will result in hiding or showing an already running instance of ABHack.

Quick revision on 16-Aug-2007

Many features have been added to ABHack since the first release in November 2006. At a point, the main window was getting so large that I had to organize it in 3 different pages. Having an outdated documentation is a kind of tradition in IT, which is not an issue for ABHack since it is proactive, self documented with tooltips, and, that all ABHack features are enabled by default. Indeed anyone a bit curious should quickly obtain the best out of ABHack just by using it and reading the few config files that can be open with a dedicated button.

However there is one little thing to be aware of for OO ABL:
ABHack can analyze your source code with a "global load" action fired on request with Alt-G (global load) , or Ctrl-S (save + global load). The global resources include: variable/properties method/func/proc (with parameters), buffers and so on... But in order to catch definitions of ***other*** source files (super classes, or other objects) ABHack relies on global definition description XML files with an file extension of ".abhack", which are managed in a parallel directory tree structure on disk.
Note also that when ABHack deals with a procedure editor for the first time, it automatically tries to find the corresponding xml file and loads it in a very fast way (so no need to fire a global load manually).

For now, the XML files can be managed with 10.1A or greater. Only two settings are necessary to enable the usage (dump and load) of xml files, again in the Misc Page:
1) Define a valid path in the "Dumped resource files root dir" fill-in (a local short path like "C:/ABHackDumps" is fine for performance).
2) Of course, check the "Use Global Resources ABHack XML Dump Files" toggle-box.

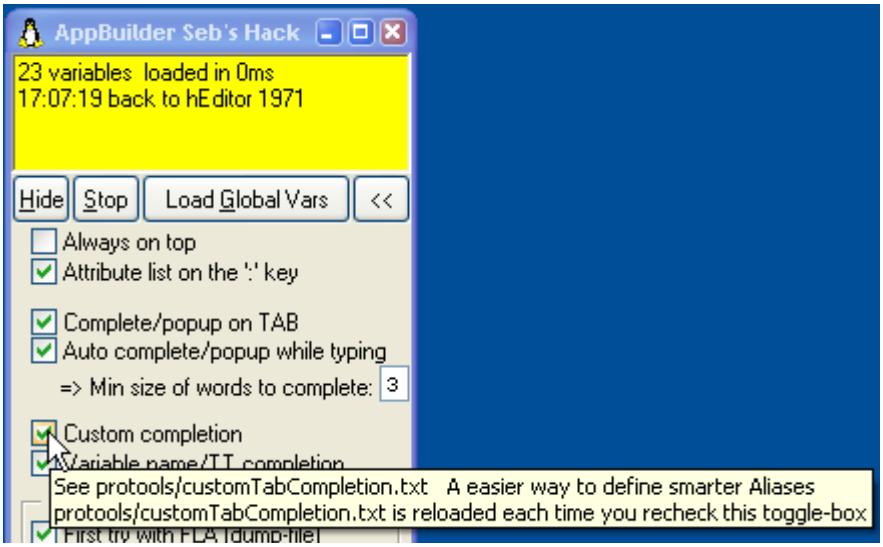
These xml files are generated in two situations:
a) When you fire a global load action (so on Alt-G or Ctrl-S), ABHack parses your code to load the resources then just takes the opportunity to (re)create the xml file to dump them.

b) You run the "Bulk Dump of ABHack Resource Desc Files Utility" (button in the Misc Page) to process all the .p .w .i and .cls files. Just use the root path of your project to generate xml files for all subdirectories. If you are very busy with Progress source files (like Dynamics or adm2), then do

not hesitate to run it with "%DLC%/src" as well (it takes only about 10 minutes to generate the 3100 files).

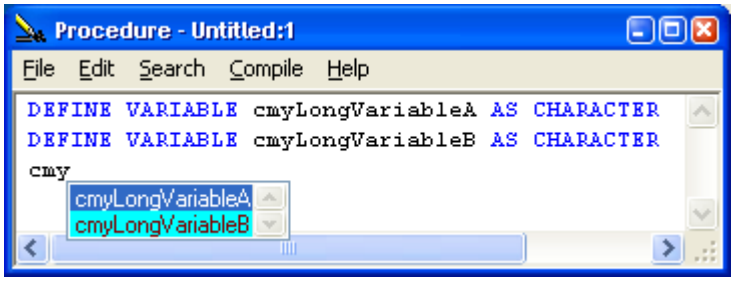
Full list of features (as per November 2006):

All features are described quickly in Tooltips when moving the mouse pointer above the various widgets of ABHack



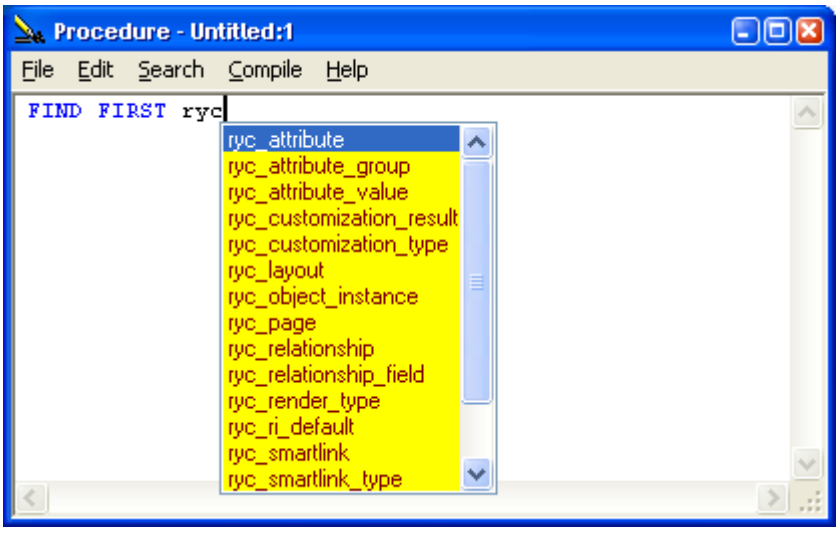
All preferences as well as the position of the window are stored on exit, and restored at start time.

The best feature is probably 'Auto complete/popup while typing'. When the size of the current word reaches 3 characters (can be adjusted), then ABHack tries to suggest a popup list of table names or variables or buffers or local parameters or even custom aliases (see the protools/customTabCompletion.txt flat file).



Note : one can force the popup to come by pressing the Tab key for words shorter than 3 characters.

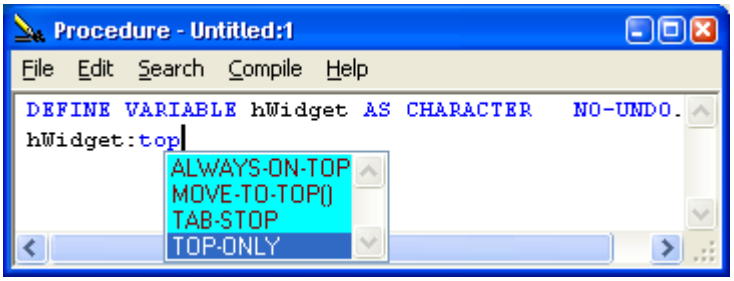
Typing a dot after a valid table/TT/buffer name makes ABHack pop up a selection-list of fields.



Typing colon after anything makes ABHack pop up a selection-list of 4GL attributes/methods (see protools/attrTabCompletion<Version>.txt).

Possible interactions with popup lists:

a) Narrow down the list by typing a part of the item(s) in the editor itself. **It uses a matches operator**, which is much smarter than a usual begins operator.
For example, when the 4GL attribute list comes by typing "myVar:", then typing "top" after the ":" will result in a smaller list with 4 items as shown bellow:



b) The up/down cursor keys automatically put the focus in the list and navigate in it
Note : of the list has one single item, then up/down make ABHack kill the list and move the cursor into the editor appropriately

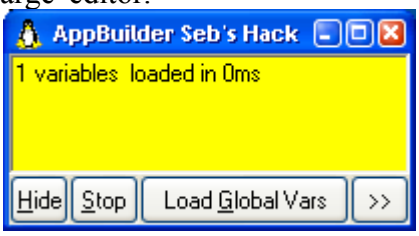
c) Hit RETURN to smart-insert the selected item into the editor
Note :
c.1) This action can be called from the editor itself
c.2) Double-clicking on an item leads to the same result as the RETURN key
c.3) The current word is replaced with the expression to insert

d) Hit ESC to kill the popup list

e) If the focus is in the list, then the TAB key can put the focus back into the editor

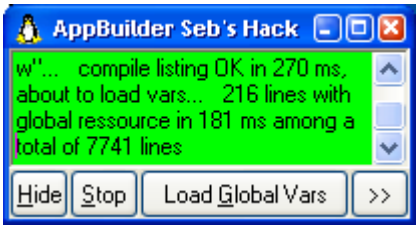
Each time you click in a different editor window, or the current number of lines varies, then ABHack re-loads the local variable and buffers in a very fast way.

Note : This is more appropriate with Section Editor Windows for structured .p, .i and .w procedures, than with the 'Large' editor.



It is up to you to press Ctrl-Alt-G or choose the 'Load global vars' button to make ABHack (re)load the global definitions of your procedure (in other words, the definitions in the definition block).

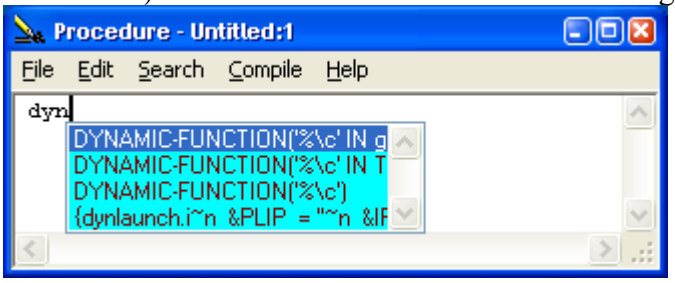
Note : the little monitoring editor changes its background color (green/yellow) to show whether global resources have been made available or not for the current procedure editor.



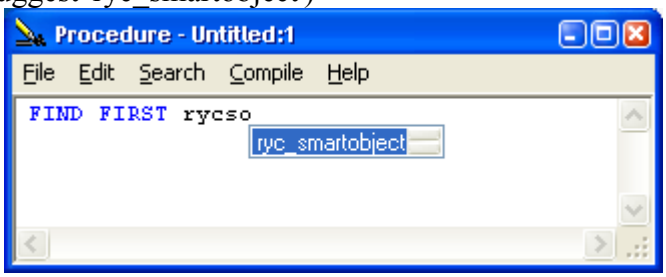
A few important points regarding the handling global resource:

1. It is required to (re)save the source file when new global definitions have been added into the definition block or in an include file
2. In a .w file, the variable names corresponding to the visual widgets are caught with the Load lobal Var action (indeed, they are just global variables with a VIEW-AS option)
3. ABHack relies on a COMPILE LISTING NO-ERROR. Never mind if it fails to compile the code, it can exploit the temporary listing file up to the error point. (just try to avoid errors in the definition block)
4. The temporary listing is generated in the SESSION:TEMP-DIRECTORY, and deleted at the end of the analyses
5. It is required to let the AppBuilder open multiple editor windows if multiple objects/procedures are open (see in AppBuilder->Option->Preferences). Indeed the global resources context is attached to source code Editor handles

ABHack offers custom aliases in the protools/customTabCompletion.txt file, that is much easier to maintain than the native alias feature, and that offers a few parsing directives. Type 'sla' and popup comment will come with my own short name with date in international format (no american/european confusion issue) and a directive to move the cursor at a given place.

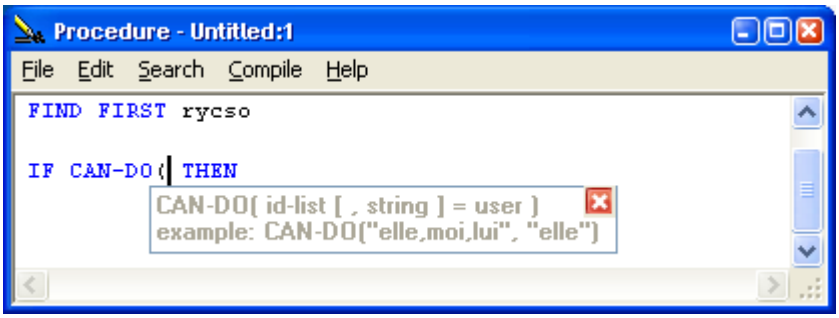


ABHack can also resolve table names from short Five Letter Acronym (type 'rycso' in a Dynamics session, ABHack then suggest 'ryc_smartobject')



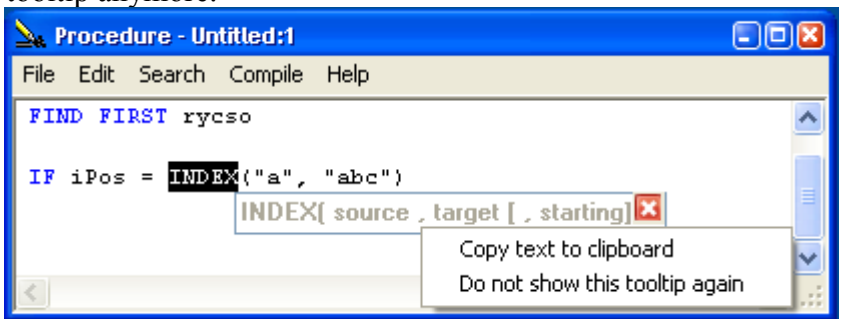
But this was just the beginning...

A cute tooltip comes up when pressing the open parentheses after a function like INDEX or LOOKUP or CAN-DO to remind you the syntax (see protools/customAPITooltip.txt)



Notes :

- One can implement his own API's or even amazing messages into this file.
- Just selecting a text like 'LOOKUP' in the source editor will make the LOOKUP tooltip show at any time.
- These tooltips remain visible for 5 sec, but one can click on them to keep them or move them around. They also have a popup menu to say the developer is not interested in a particular tooltip anymore.



Type: `MyTable.LongFieldName++`
ABHack will then transform it into: `MyTable.LongFieldName = MyTable.LongFieldName +`
Other similar actions:

`myVar-+`	`myVar = myVar -`
`myVar,+`	`myVar = myVar + "," +` (the same with ';' or '|')
`myVar1+`	`myVar = myVar + 1`
`myVarCHR(x)+`	`myVar = myVar + CHR(x) +`

Note : This works with `myTable.myField` or `myHandle:SOME-ATTRIBUTE`

Type double slash ('//') anywhere in a line (even in a middle of a word) and it will be commented out or uncommented, then the cursor moves to the next line so it can be processed it the same way. If double slash is type at the end of a line, then a new empty comment is inserted at the end of the line and the cursor stays there

ABHack can achieve a smarter navigation by words on Ctrl-cursor-left and ctrl-cursor-right:

1. it does not ignore words made of only digits
2. goes to end of line before going to next word at next line
3. it considers '_' as part of words

A Ctrl-Alt-L feature can align, refine and sort a selected block of variable definitions, or align the '=' signs in a multiple assign block.

For example, select the following definition code:

```
DEF VARIABLE cLine         AS CHAR  NO-UNDO.
DEF VARIABLE cOldString    AS CHAR  NO-UNDO.
DEF VAR iStartTime  AS INT    NO-UNDO.
DEF VARIABLE iCount        AS INT    NO-UNDO.
DEFINE VAR iLine     AS INT    NO-UNDO.
DEF VARIABLE cTxt          AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cFunctionCase AS CHAR  INITIAL "notDoneYet" NO-UNDO.
DEFINE VARIABLE cPrevVar      AS CHAR  NO-UNDO.
DEFINE VAR iLineMax      AS INTEGER    NO-UNDO.
DEFINE VARIABLE cVar  AS CHAR  NO-UNDO.
DEF VARIABLE cBuffer       AS CHARACTER  NO-UNDO.
DEFINE VAR cFor          AS CHARACTER  NO-UNDO.
```

Then press CTRL-ALT-L and it is transformed to:

```
DEFINE VARIABLE cBuffer       AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cFor          AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cFunctionCase AS CHARACTER  INITIAL "notDoneYet" NO-UNDO.
DEFINE VARIABLE cLine         AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cOldString    AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cPrevVar      AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cTxt          AS CHARACTER  NO-UNDO.
DEFINE VARIABLE cVar          AS CHARACTER  NO-UNDO.
DEFINE VARIABLE iCount        AS INTEGER    NO-UNDO.
DEFINE VARIABLE iLine         AS INTEGER    NO-UNDO.
DEFINE VARIABLE iLineMax      AS INTEGER    NO-UNDO.
DEFINE VARIABLE iStartTime    AS INTEGER    NO-UNDO.
```

Or select the lines *bellow* the ASSIGN Statement (so 3 lines)

```
ASSIGN
 cMyVar = "this is an assign alignment test"
iCount = iCount + 1
    myTable.myField = myVar.
```

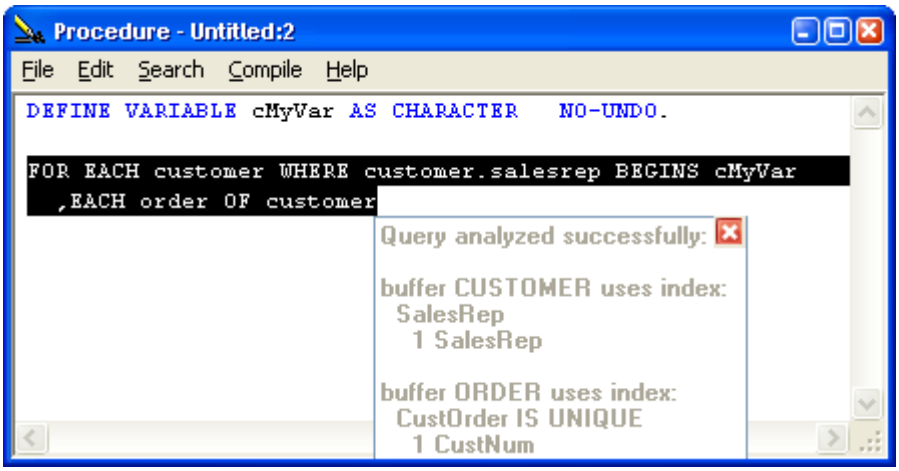
Then press CTRL-ALT-L:

```
ASSIGN
 cMyVar          = "this is an assign alignment test"
 iCount          = iCount + 1
 myTable.myField = myVar.
```

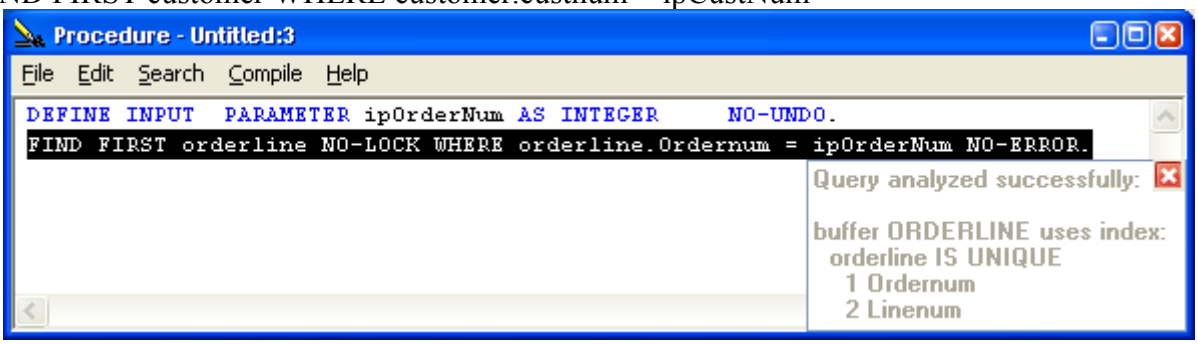
Note : the left side of the lines is aligned to the one of the first selected line.

Now, cherry on the cake (Selected Query text analyzer):

Select some text like:
"FOR EACH customer WHERE customer.salesrep BEGINS cMyVar
,EACH order OF customer":



Or select something like
"FIND FIRST customer WHERE customer.custnum = ipCustNum"



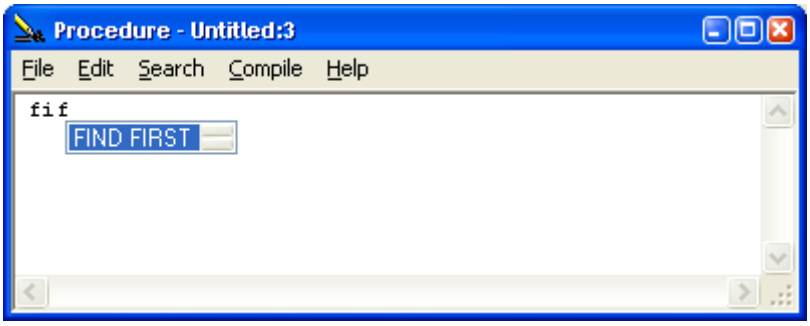
ABHack will analyze the query and show index usage and details about them in a tooltip.
The analyzer can cope with any Static Query or FOR EACH or FIND as long as they are against tables of a connected Database. It can manage variable name and temp-table field names used in the where clause of the query (they are replaced by '?').

The analyzer will show potential syntax errors if it fails to analyze the query. In this case, the tooltip shows for a very short time. If one is really interested in it, then he shall just click on it before it disappears.

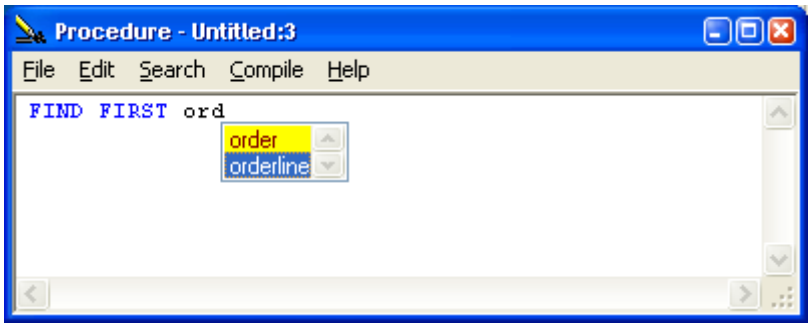
> Note : To analyse a selected FIND Statement,ABHack transforms into a Dyn Query Prepare String with a FOR EACH, then it takes care of the first entry of INDEX-INFORMATION()

Quick Example of auto-completion (sports or sports2k DB):

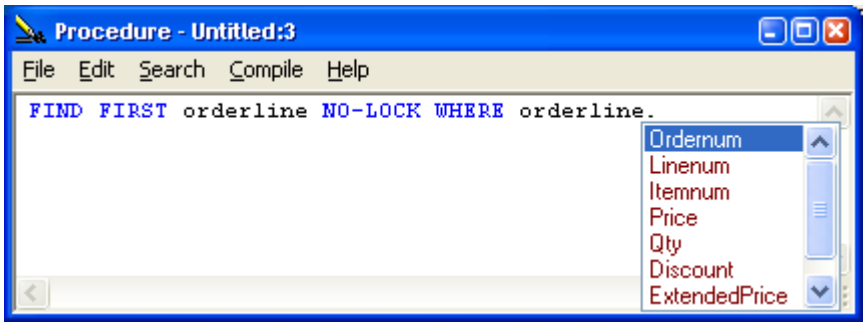
Type 'fif', "FIND FIRST " pops up, then hit the RETURN key



Then type 'ord', a table list pop up to offer the order and orderline table names. Just hit cursor-down and the focus goes into the list then 'orderline' is selected. Then hit the RETURN key.



ABHack then automatically completes it to *FIND FIRST orderline NO-LOCK WHERE orderline.* then and a list pops up with orderline fields to complete the query:

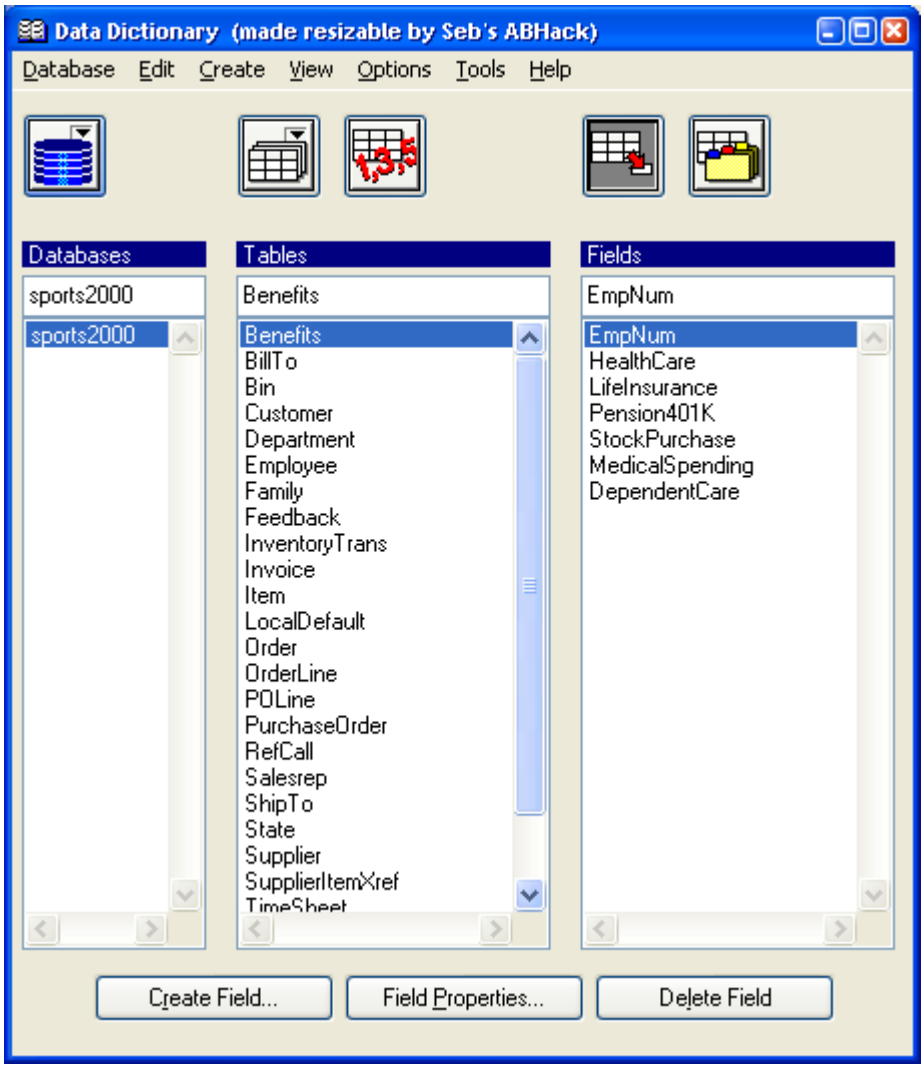


So, to resume 9 key stokes with : *fif* RETURN *cus* CURSOR-DOWN RETURN

> Note : that 'fif' is a custom alias that can be removed from the protools/customTabCompletion.txt file if wanted

Make the Data Dictionary resizable:

Nowadays, the original size of the data ditionary looks a bit ridiculous on a modern display. ABHack can solve this problem by grabbing its widgets and defining a WINDOW-RESIZED trigger. It shows it can do it by enlarging it a first time



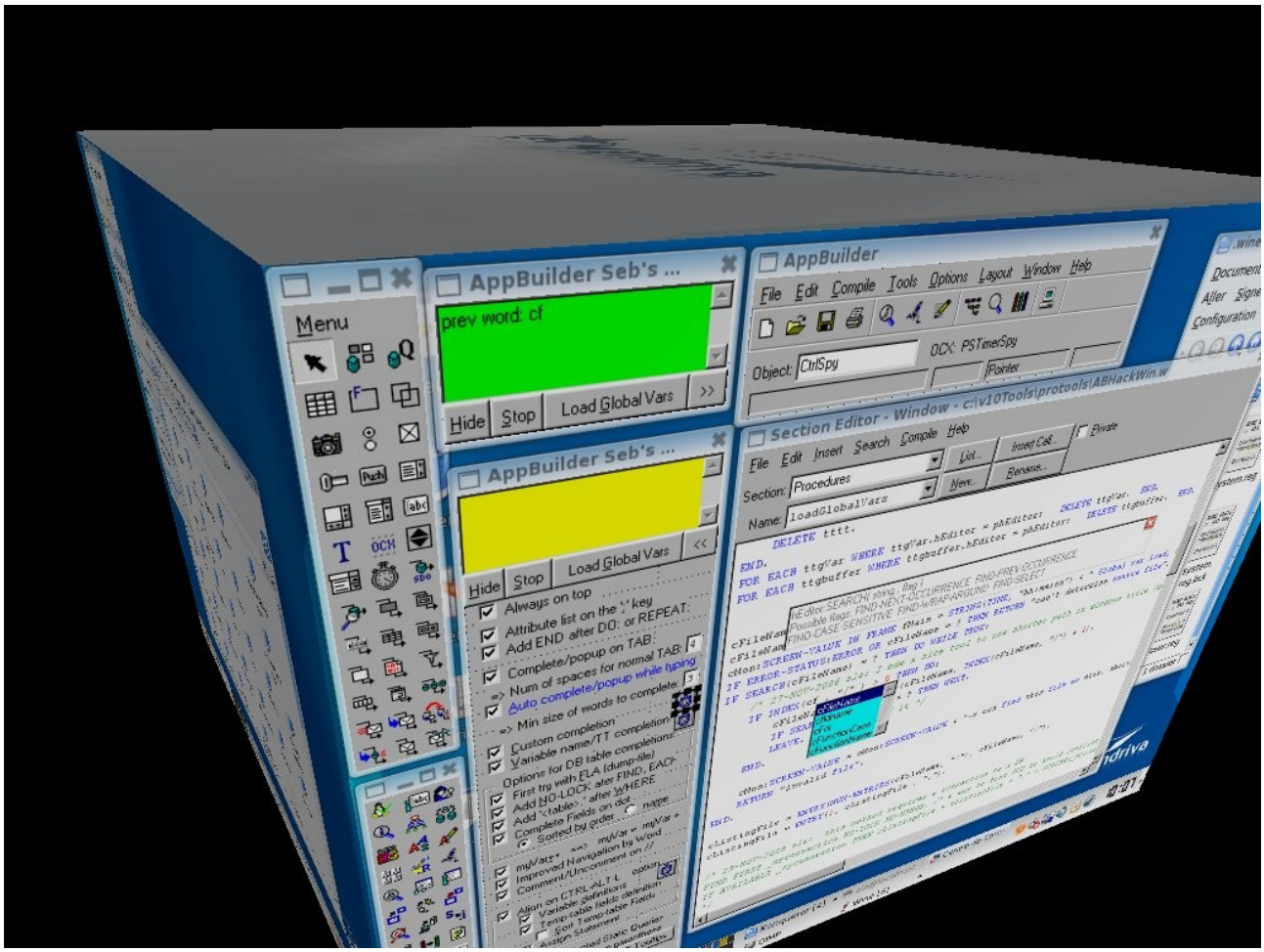
Why "ABHack" ?

ABHack is not limited to the procedure editor in itself. For instance it increases the number of visible inner-lines of the Event/ProcName Combo-box of the Section Editor (this is enough to change your all life isn't it?), or can make the Data Dictionary resizable.

Indeed, ABHack is a good place holder to hack all these tools dynamically, without modifying their source code, let alone recompiling those guys. I may also hack a few property dialog-boxes in the future. How about the tiny editor to define temp-tables in a .w file? Or a procedure/function pickup list for a selected .p/.w?

By the way, why a penguin icon? Well, it's a cute and positive animal isn't it?

ABHack in a Compiz 3D Desktop on Linux with wine?



Hard to get used to a few features?

I got some feedback about a few features that can be a little bit irritating until one can really get used to them. Indeed, one might find the automatic adding of `NO-LOCK`and `WHERE` a little bit annoying in a few cases, espacially when one wants to remove them with backspace.

Example say you want to type a simple `FIND FIRST customer NO-LOCK.` (full stop after no lock). Then you might end up with `FIND FIRST customer NO-LOCK WHERE customer.`*`[field popup list]`* before typing the full stop without asking...

First, Note that if you do not like these ***optional*** features, then you can just disable them for goods with their corresponding toggle-boxes. They are indeed switched on by default, but they can be switched off for goods as well.

Next, notice these added words are more often wanted than unwanted. One good way to remove them is to select them (`WHERE customer.` you can even do that with a few shift-ctrl-left key strokes) and type '.' to ***replace*** the selected text by the full stop, an action that will **not** re-fire the automatic addition of the `WHERE` word.

Note also that `NO-LOCK` is not added after a temp-table name. A killing little details that shows that a developper knows what he is doing (unlike a stupid `FIND FIRST myTT EXCLUSIVE-LOCK NO-WAIT NO-ERROR`)

The same applies with 'Comment/Uncomment with '//''. One can very well to keep this feature on and manage to type '//' occasionaly by first typing '/ /' then remove the space between the slashes. Just hack the hack.

Actually, despite I am the author of ABHack, I am considerably changing my own way of developping with it, and this is not a very fast process. One first aspect is I tend to use copy/paste far less often (a really good point with this single item clipboard of MS Winbloze).

At last, as a popular Wizzard of Technology likes to quote on the peg *"A language that does not affect the way you think about programming is not worth knowing"*.

Thanks:

Many thanks to my current project leader Jean-Christophe Cardot, for his strong interest and support in the achievement of this tool. Thanks to my developer pears at PSA for their feedback when using it.
At last, many special thanks to my wife Amandine, who had to bear with yet another 'programming fever' and that laptop that even came to bed a few times...

Now, I am wondering if this tool could not be taken as a good competitor to Eclipse itself.
If you like it, then I would appreciate a little email to sebastien@slacroix.fr, or a little bottle of whisky ;)

A special mailing list has been opened on the peg for ABHack:

http://www.peg.com/lists/abhack/web/threads.html
=> please have a look there if you have any concern about it

Release Notes are now handled in a dedicated .txt file