Home x Install x Full Index x Tutorial x EnumerateControls x FindOutTableExists x UseVariablesInSQL x CreateRecordsetFrom x AddRecordToRecordset x CountRecordsRecordset x LimitsRecordset x MixAccess2baseAndUNO ! User's Guide x AllForms x DatabaseWindow x ShortcutNotationMore x DLookupSamples x CalculatedField x MultiSelectListboxSelectForm x FillAutoControlValue x CarryToNewRecord x BrowseThruControls x TipTextForLongValues x AskBeforeSaving x Sync2Combos x ZoomOnImage x AddAllToBox x KeepFormsSynchro x SelectListboxOnFirstLetters x MoveItemsBetweenListboxes x SimulateTabbed x SearchStandalone x CalculatorDialog x ExploreTables x ExtractDataTable x FindPositionRecordset x DMedian function x DPercentile x ImportImages x ExportImages x CrossTabQuery x DbaccessFromCalc x

DbaccessFromCalc

tags:
HowTo

(Q) How could I get data stored in a database from within a Calc cell formula ?

(R) Calc offers as a standard feature to insert data from database tables and queries via a specific browser that is invoked by using the `View` + `Data Sources` menu command or the `F4` function key.
The Help files describe in detail how to proceed.
However the question here is, as an example : how is it possible to enter a **product code** in a cell and get in another calculated cell its **description**, knowing that the correspondence between both fields is somewhere in a database table and not in the spreadsheet ?

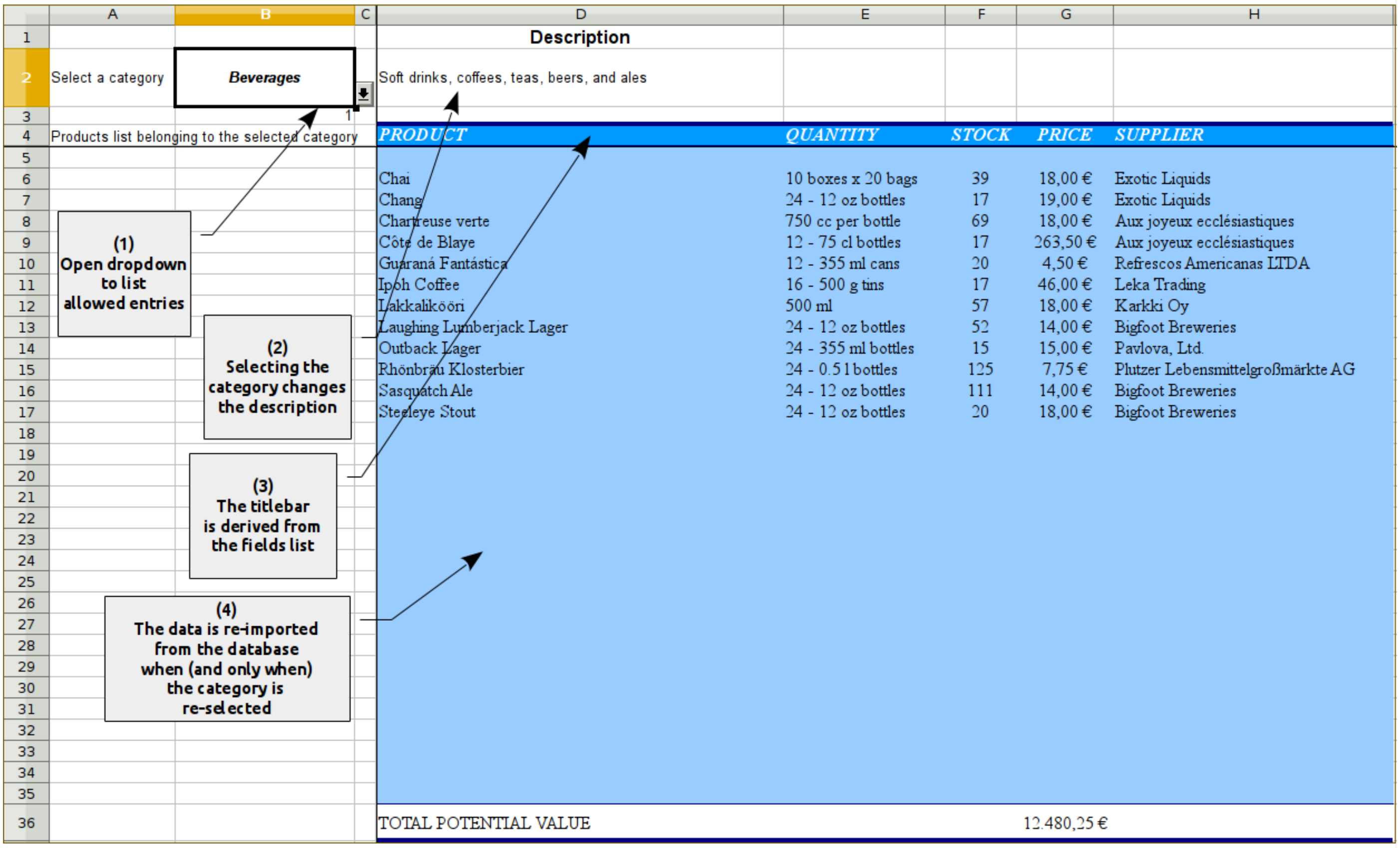
Obviously, to not simplify the problem, we would appreciate to have things happen automatically, i.e. also when the sheet is configured to have its

> `Tools + Cell Contents`

set to **`AutoCalculate`**, and without useless database accesses.

In the proposed solution we will go even further. The example shown will illustrate next 4 functionalities:

- Use database data to populate a dropdown listbox.
- Derive automatically the *description* from the selected item in the listbox.
- Prepare a report with the heading derived from the database table field names.
- Populate an array of data in the sheet extracted from the database and filtered by the choice in the dropdown box..



Description

Select a category **Beverages**

Soft drinks, coffees, teas, beers, and ales

Products list belonging to the selected category

PRODUCT	QUANTITY	STOCK	PRICE	SUPPLIER
Chai	10 boxes x 20 bags	39	18,00 €	Exotic Liquids
Chang	24 - 12 oz bottles	17	19,00 €	Exotic Liquids
Chartreuse verte	750 cc per bottle	69	18,00 €	Aux joyeux ecclésiastiques
Côte de Blaye	12 - 75 cl bottles	17	263,50 €	Aux joyeux ecclésiastiques
Guaraná Fantástica	12 - 355 ml cans	20	4,50 €	Refrescos Americanas LTDA
Ipoh Coffee	16 - 500 g tins	17	46,00 €	Leka Trading
Lakkalikööri	500 ml	57	18,00 €	Karkki Oy
Laughing Lumberjack Lager	24 - 12 oz bottles	52	14,00 €	Bigfoot Breweries
Outback Lager	24 - 355 ml bottles	15	15,00 €	Pavlova, Ltd.
Rhönbräu Klosterbier	24 - 0.5 l bottles	125	7,75 €	Plutzer Lebensmittelgroßmärkte AG
Sasquatch Ale	24 - 12 oz bottles	111	14,00 €	Bigfoot Breweries
Steeleye Stout	24 - 12 oz bottles	20	18,00 €	Bigfoot Breweries

TOTAL POTENTIAL VALUE 12.480,25 €

(1) Open dropdown to list allowed entries

(2) Selecting the category changes the description

(3) The titlebar is derived from the fields list

(4) The data is re-imported from the database when (and only when) the category is re-selected

A solution

Let's consider next tables:

- Categories table

Fields	Field Type	Primary
CategoryName	Text	
Description	Text	
Picture	Binary	
CategoryID	BigInt	Y

- Products table

Fields	Field Type	Primary
CategoryID	BigInt	
ProductName	Text	
QuantityPerUnit	Text	
ReorderLevel	Integer	
SupplierID	BigInt	
UnitPrice	Number	
UnitsInStock	Integer	
UnitsOnOrder	Integer	
Discontinued	Boolean	
ProductID	BigInt	Y
Picture	Image	

The final purpose of the spreadsheet is to list all products belonging to the category selected by the user.

Preamble

The challenge will be to do next things apparently simultaneously without the user becoming aware of the underlying complexity:

- to load the Access2Base macro library, only once
- to open the database file (".odb") referring to the effective database, also only once
- to extract the needed data only when relevant and every time it is, i.e. when some input parameter has been modified.

When Calc recomputes a worksheet, it sequences its computations in function of the respective arguments present in each cell formula.
For example, if

  - cell A1 is a number
  - C1 contains the formula `"=A1*2"`
  - and B1 contains the formula `"=C1*C1"`

the worksheet will not be recomputed from left to right, but in the sequence A1, C1, B1. In addition, C1 and B1 will be both recomputed automatically every time A1 receives another value.
Choosing in this matter either cells or "names" (defined by `Insert` + `Names`) does not make any difference. Names will be preferred if they do not need to be visible in the sheet.
The sequencing of computations done by Calc is the mean we will use to reach our challenge.

Global

To access the database we have to make use of a database object. Let's define it as a `Global` variable.
Such variables remain in life as long as

- the AOO/LibO session is lasting
- the module where it is declared is not edited.

Usually I declare `Global` variable in a separate module as such a module is unlikely to be modified often.

```
Global oMyDatabase              As Variant
```

Define Name

Define a `Name` called `IsConnected`. Store in it the formula

```
=DBCONNECTED()
```

DBCONNECTED is a user-defined function and behaves exactly like any builtin Calc function or expression. It has no argument.
The code of the function is here:

```
Sub _Init()
Dim oLib As Object
        Set oLib = GlobalScope.BasicLibraries
        If oLib.hasByName("Access2Base") Then
                oLib.loadLibrary("Access2Base")
        End If
End Sub

Function DBConnected() As Variant
Dim sCalc As String
        DBConnected = 0
        If IsEmpty(oMyDatabase) Then
                Call _Init()            '       Load Basic libraries
                Set oMyDatabase = OpenDatabase(".../TT%20NorthWind.odb")        '       Put here the URL of the targeted database
        End If
        DBConnected = 1
End Function
```

The result is that the loading of the library is put on the computation path. There is no reason why Calc would require recomputation several times of `IsConnected` except while loading the spreadsheet.
Now we can build other formulas, like:

```
=IF(IsConnected;USERDEFINED(...);False)
```

being sure that such formulas will be evaluated by Calc only when the evaluation of `IsConnected` has been achieved.

Setup the dropdown box

Use the `Data` + `Validity` menu commands to define *Criteria* as being a *Cell range*, select the *Show selection list* checkbox and enter as *Source* next formula

```
=IF(IsConnected;CATEGORIESLIST();"")
```

The CATEGORIESLIST() function:

```
Function CategoriesList() As Variant
'       Return the list of available product categories as a vector
Dim oRs As Object, sCatsRC() As Variant, sCats() As Variant, i As Integer
        If Not IsEmpty(oMyDatabase) Then
                Set oRs = oMyDatabase.OpenRecordset("SELECT [CategoryName] FROM [Categories] ORDER BY [CategoryName]")
                sCatsRC = oRs.GetRows(1000)             '       matrix (row, column)
                sCats() = Array()                               '       Reduce to column only
                ReDim sCats(0 To UBound(sCatsRC, 1))
                For i = 0 To UBound(sCats)
                        sCats(i) = sCatsRC(i, 0)
                Next i
                CategoriesList = sCats()
                oRs.mClose()
        End If
End Function
```

Find description

The dropdown box is in cell `B2`. We put in cell `D2` next formula:

```
=IF(IsConnected;CATLOOKUP(B2);"")
```

that will be recomputed by Calc each time the cell `B2` is modified by the user.
The CATLOOKUP function:

```
Function CatLookup(ByVal pvArg As Variant) As Variant
        If Not IsEmpty(oMyDatabase) Then CatLookup = oMyDatabase.DLookup("[Description]", "[Categories]", "[CategoryName]='" & pvArg & "'")
End Function
```

Fill the data

The data matrix will be inserted as an **array formula** (`Ctrl` + `Shift` + `Enter`) in cells `D6:H35`. Look at the AOO/LibO help to know more about them.

```
{=IF(IsConnected,PRODUCTSLOAD($B$2),"")}
```

The PRODUCTSLOAD function:

```
Function ProductsLoad(ByVal pvCat As Variant) As Variant
```

```
Dim oRS As Object, sSQL As String, vResult() As Variant, i As Integer
Const cstSize = 30
        If Not IsEmpty(oMyDatabase) Then
                sSQL = "SELECT [ProductName] AS [Product], [QuantityPerUnit] AS [Quantity], [UnitsInStock] AS [Stock], [UnitPrice] AS [Price]" _
                        & ", [CompanyName] AS [Company]" _
                        & " FROM [Products], [Categories], [Suppliers]" _
                        & " WHERE [Products].[CategoryID] = [Categories].[CategoryID]" _
                        & " AND [Suppliers].[SupplierID] = [Products].[SupplierID]" _
                        & " AND [Categories].[CategoryName] = '" & pvCat & "'" _
                        & " ORDER BY [Product] ASC"
                Set oRS = oMydatabase.Openrecordset(sSQL)
                '       Enumerate field names
                vFields() = Array()                     '       Mandatory before resizing a variant
                ReDim vFields(0 To oRS.Fields.Count - 1)
                For i = 0 To UBound(vFields)
                        vFields(i) = oRS.Fields(i).Name
                Next i
                '       fetch recordset data
                vResult() = oRS.GetRows(cstSize)
                oRS.mClose()
                '       To avoid #N/A
                If UBound(vResult, 1) < cstSize Then
                        ReDim Preserve vResult(0 To cstSize, 0 To UBound(vResult, 2))
                End If
                ProductsLoad = vResult()
        End If
End Function
```

A second `Global` variable is used here to store the field names in a more generic way:

```
Global vFields()                As Variant
```

Headings

The title of the data matrix is again an array formula, put in cells `D4:H4`.
The `G36>0` condition below makes that the heading cells are evaluated after the data cells. Indeed titles are extracted from the database in the same `Sub` as the data.

```
{=IF(AND(IsConnected,G36>0),DBTITLE(),"")}
```

associated with next code:

```
Function DbTitle() As Variant
        If Not IsEmpty(oMyDatabase) Then
                DbTitle() = vFields()
        End If
End Function
```

Close the connection

Finally it is always recommended to clean the connection to the database.
Associate next code

```
Function DbClose()
Dim vEMPTY As Variant
        If Not IsEmpty(oMyDatabase) Then oMyDatabase.mClose()
        oMyDatabase = vEMPTY            '       Reinitialize oMyDatabase to empty
End Function
```

with the `Document closed` event (`Tools` + `Customize` - `Events` tab).

See also

Close
DLookup
GetRows
OpenDatabase
OpenRecordset

Refer to ...

File	Basic module
TT NorthWind Calc.ods	Connect Globals

Bookmark this page » » DbaccessFromCalc