

User-Driven Menus

Keith Bombard



If you've ever worked with Access to develop large, multi-user databases with thousands of objects, then you'll want to think about limiting user access to those objects. In this article, Keith Bombard shows you a simple and proven technique to manage access using user-driven menus.

RECENTLY I had the opportunity to develop a very large Access 97 system for a high-volume electronic printing company on the East Coast. This company builds, prints, and mails high-end brokerage and mutual fund statements for more than 300 client companies. The system's primary mission was to track the flow of print-job work orders as they made their way through the plant. Originally, the system was to be used by a limited number of users and departments. It turned out to be one of those never-ending development projects with an unlimited budget and suffering from chronic scope creep. The system and the user base grew and grew until it had grown to hundreds of forms and reports, more than 50 users, and eight different departments.

It wasn't long before users started to express their dismay at having to wade through a torrent of menu choices that they knew nothing about and cared nothing about. Each department needed specific parts of the system, so the logical solution was to group users into Menu Groups. The plan was to give each user in the department a Menu Group ID that would control which menus a user would be presented with. The project's menu system could be adapted to accommodate this enhancement without affecting the basic menu approach already in place, making the transition transparent to the end user.

This approach helped to solve another problem that was becoming a major issue: enforcing the security scheme. If users could be limited to what they saw on their menus, they would stay on course and be much less likely to run into dreaded security violation messages.

The approach that I used is one that I've found has universal appeal to users: dual list box menus (see [Figure 1](#)). The top-level list box holds "Main Topic" choices, while a subordinate-level list box holds the selected Main Menu's "SubTopic" choices. Selecting a Main Topic displays the related SubTopic. Selecting a SubTopic opens the form, table, or report associated with that SubTopic. The Main Topics displayed to the user form a Menu Group. A user table holds the Menu Group

ID for a given user so that the correct list of Main Topics for the current user is displayed. The whole system is driven by a set of tables that can be updated at any time. In fact, while I've built an extensive system to manage my menus, implementing the menus requires very little code at all.

Tables make it work

I use a set of four tables to implement my menu system. Before describing those tables, I should explain my naming convention. I always number the tables in my databases (a tip from an ex-mainframe DB2 compatriot who never had the flexibility in table naming we've become accustomed to in Access). Numbering tables provides a whole new dimension to working with tables, and I highly recommend it—especially when working with a large number of tables in a database. Numbered tables are sorted in numeric order in the database window, so finding the table you want is easier when you can't remember the exact table name or your table names are similar, provided you group your tables by number.

Most of the systems I work with have upwards of 200-300 tables, and I'd be at a great loss without a numbering scheme. I use the tables numbered in the 900+ range as system/administration tables. This makes it doubly easy to pull these tables into projects to quickly

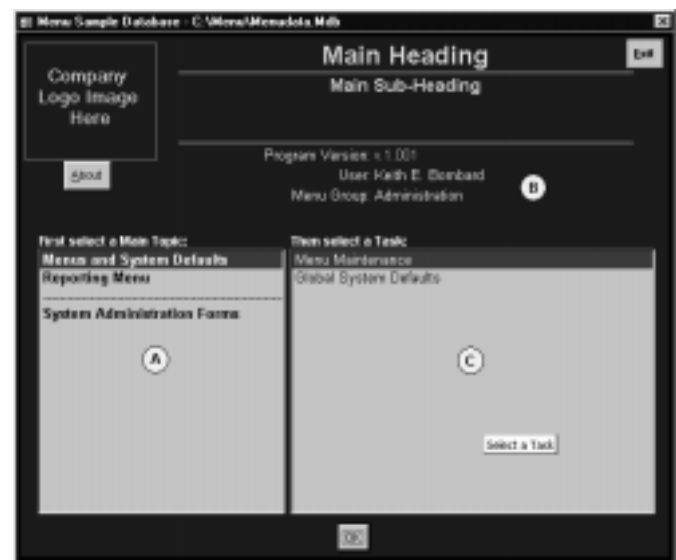


Figure 1. A dynamic menu, keyed to individual users.

establish the same base-level administration services in each new project. The tables that I use when implementing my multi-level menus are:

- tbl907_MenuMaster: The top-level menu table. It holds the MenuGroupID key field and the basic information about the Menu Group, and it establishes each Menu Group in the system. Menu Groups are collections of Main Topics and SubTopics.
- tbl906_MainMenuTopic: The Menu Groups' Main Topic descriptions.
- tbl905_MainMenuSubTopic: Each Main Topics' SubTopic descriptions and other critical object data needed at runtime.
- tbl900_UserMaster: This table is the main user table. It holds basic information specific to each user. A key field in this table is the MenuGroupID field. Each user is assigned a specific MenuGroupID, which keys them into the proper Menu Group at logon.

The relationship diagram shown in [Figure 2](#) illustrates the tables and relationships that support my multi-level menus.

Forms make it work

The tables aren't much use unless you can update them. Three forms support the menuing tables:

- Menu Maintenance form
- User Master form
- Main Menu form

Menu Maintenance form

The Menu Maintenance form (see [Figure 3](#)) implements the concept of Menu Groups and is used to manage all menus in the application. Four data tables and four subforms are shown in this form that display Menu Groups, Menu Main Topics, Menu SubTopics, and the User Master records. This form is used to manipulate the Main Menu display choices for each Menu Group. Users can do the following:

- Add, change, or delete Menu Groups (area A)
- Add, change, or delete Main Topics in a specific Menu

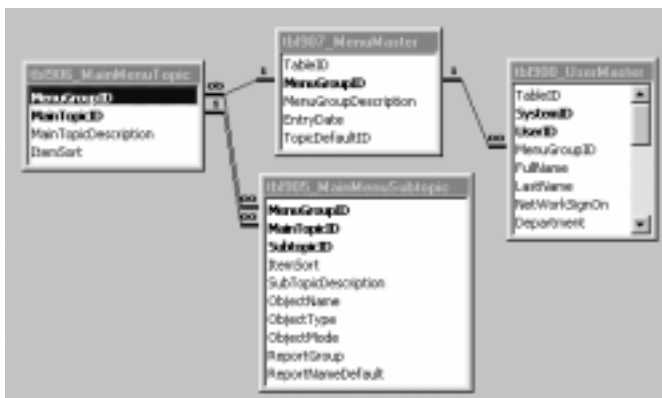


Figure 2. Menu-table relationships.

Group (area B)

- Add, change, or delete SubTopics from a selected Main Topic choice (area C)
- Add, change, or delete a user's MenuGroupID assignment (area D)

Users can also be reassigned into Menu Groups.

Menu Groups define the highest level of menu processing. Menu Groups usually represent a group of personnel performing similar tasks. Examples include separate Menu Groups for the data entry and administration groups. The administration group might even be defined as an unabridged list of menu choices.

As you move your cursor into different Menu Group records, related data dynamically displays in the other three subforms. To add a new Menu Group, enter a new MenuGroupID value (for the sake of consistency, try to stay in the same numeric series as the previous rows) along with a menu description. To delete a Menu Group, select the row for deletion with your mouse and press the Delete key. The form won't let you delete a Menu Group if a user is still linked to the group. When this happens, you'll need to change that user's MenuGroupID in the bottom subform to a different one.

Main Topics are subordinate to the Menu Groups and function as high-level categories for associated SubTopics. These choices display in the left-side list box of the Main Menu form. The sort order column is used to control the display, sorting in both the Main Menu form and this Maintenance form. These values can be edited to change the display position of the listed item. You can even use decimal values to position menu entries between other entries. As you move your cursor into different Main Topic records, related data dynamically displays in the SubTopic subform.

SubTopics are the specific menu choices appearing in the right-hand list box of the Main Menu. They're subordinate to the Main Topics and hold the target object



Figure 3. The Menu Maintenance form.

references that open when the user executes a menu choice from the Main Menu form. As in the Main Topic subform, the Sort Order field controls display sorting. SubTopics can reference a table, form, query, report, or macro. Sort Order values can also reference a custom report-menu form.

Users who belong to the selected Menu Group are listed in the bottom subform. You can remove users from a selected group by changing their MenuGroupID, assigning a user into the new Menu Group.

The MenuGroupID field

The MenuGroupID field in the user table determines which menu the user will see when the Main Menu form loads. I've seen a wide range of uses for user tables in Access systems, from no user table at all up to very elaborate user tables that hold lots of information. I think that too much data in a user table can lead to extra administration costs and even, in extreme cases, user resentment about the amount of data stored about them. On the other hand, systems without a user table can be mired in hard code in order to implement user-specific processing. To implement my menuing system, you'll need a field in a user table that matches the value of Access's own CurrentUser variable. Figure 4 shows the user table that I use for my menuing system.

I use a function called CUserID() to return the three-digit alphabetic UserID field from the user table (I usually use user initials as my UserID). The function is quite simple: It uses the Access CurrentUser function, performs a lookup in the user table by matching CurrentUser with the FullName field, and returns the UserID for that user. Once the UserID is retrieved, it's stored in the global variable glbUserID, and the variable is used in the rest of the application. This technique speeds up repeated processing by limiting the table lookup to the first occurrence:

```
Function CUserID() As String
```

```
On Error GoTo CUserID_ERR

If Len(Trim(glbUserID)) = 0 Then
    glbUserID = DLookup("UserID", _
        "tbl1900_UserMaster", _
        "CurrentUser() = FullName")
    CUserID = glbUserID
Else
    CUserID = glbUserID
End If

CUserID_EXIT:
Exit Function

CUserID_ERR:
Select Case Err.Number
Case Else
Call ScrError(Err, Error, _
    "CUserID", MsgHeader)
Resume CUserID_EXIT
End Select
End Function
```

The user table that's included in the sample database with this month's Source Code files (available at www.smartaccessnewsletter.com) was pulled from a recent project. The preceding user form is an abridged version of the form used in that project. You'll find additional fields in the table that aren't displayed on the form.

Of course, in order for this function to work, the FullName field in the user table should exactly match the Access user name (Access's CurrentUser variable). I recommend that new users be added to the system using the Add New User button on this form. Code behind the Add form (not shown here) will add the new user's name and PID to the system workgroup file at the same time the new user is added to the user table. I also recommend that you use the "/user UserName" command line parameter syntax in the Access launch shortcut on each user's desktop. This will ensure that the correct user is logged in at startup.

Menu selection

The Main Menu form (see Figure 1) implements Main Topic and SubTopic menu selections for each user. This is a dual list box form; the left-hand list box holds the Main Topics, and the right-hand list box holds the Main Topics' subordinate SubTopics.

The Main Topic list box (A) has a SQL statement as its row source that links the current user's MenuGroupID field to the Main Topic MenuGroupID field (see Figure 5).



Figure 4. The User Master form displays essential data for each user.

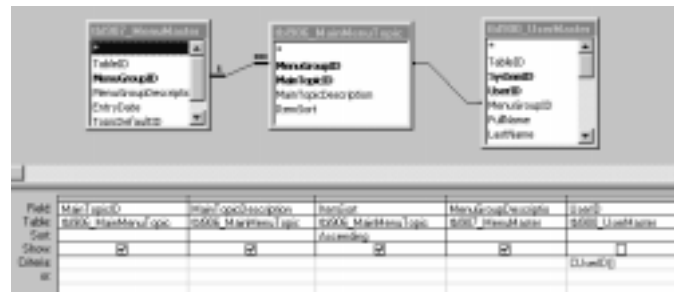


Figure 5. SQL Select row source for the Main Topics list box.

The result is that only Main Topics belonging to the user's Menu Group are listed. Users with the same Menu Group see the same menu choices. When you scroll through the Main Topic choices in the left-hand list box, the AfterUpdate event fires and requeries the right-hand list box, refreshing the corresponding SubTopic listing.

Using the information in the tables, you can load the two list boxes displayed in Figure 2. When a user selects an entry in the SubTopic list box, you can use the information from the tbl905_MainMenuSubTopic to open the Access object associated with the entry. I load the information into columns in a list box. In the following sample code, the list box is called lstST. The code checks to see whether the object associated with the list box entry is a Form, Report, or Table in order to select the right DoCmd method. If the item is a Form, the code then checks to see in what mode to open the Form (Add, Edit, or default) in order to set the right parameters. Finally, the code uses the object name to open the object:

```

If Me![lstST].Column(2) = "Form" Then
  If Forms![frmMainMenu]![lstST].Column(3) _
    = 2 Then
    DoCmd.OpenForm _
      Forms![frmMainMenu]![lstST].Column(1)
  Else
    DoCmd.OpenForm _
      Forms![frmMainMenu]![lstST].Column(1), _

```

```

      ' ' ' -
      Forms![frmMainMenu]![lstST].Column(3)
  End If
End If
If Me![lstST].Column(2) = "Report" Then
  DoCmd.OpenReport _
    Forms![frmMainMenu]![lstST].Column(1), _
    A_PREVIEW
ElseIf Me![lstST].Column(2) = "Table" Then
  DoCmd.OpenTable _
    Forms![frmMainMenu]![lstST].Column(1), _
    acViewNormal, acReadOnly
End If

```

The real benefit of this system is that the menus are completely table-driven. Adding a new form to the Main Menu just consists of adding the appropriate entries to the underlying tables. You'll find a complete implementation of the system with this month's Source Code files, ready to add to your application. ▲

 [TBLMENU.ZIP at www.smartaccessnewsletter.com](http://www.smartaccessnewsletter.com)

Keith Bombard is a contract programmer who's employed by Howard Systems International and specializes in large Access implementations. He's been developing MS Office-based solutions for multiple clients since 1993. Prior to that, Keith was a systems manager in the financial industry. He's currently on an extended assignment building three large Access databases for the Connecticut Department of Environmental Protection. Keith.Bombard@PO.STATE.CT.US, KeithBombard@Home.com.