# Understanding MS Access Security

*There are two fundamental approaches you can take to securing your database: user-level security or share-level security. We explain how to implement each, and what you should take into account when formulating your strategy.*

**By Pan Pantziarka**
**Technical Writer**

Although Microsoft Access is one of the leading desktop databases in the world, it has a poor reputation when it comes to implementing security. MS Access security is generally perceived as being difficult to understand, overly complex and easy to get wrong. The aim of this article is to shed some light on the subject, to give some guidance on how to implement secure databases and, just as importantly, on how to help your users cope with problems they may have with the Access security system.

To begin with we have to look at what precisely we mean by database security. Firstly it is not about making a database multi-user; issues to do with record locking, concurrency and so on are to do with fundamental database and application design, and are not covered here. In this context we are looking specifically at:

- Making sure unauthorised users cannot log on to the database
- Ensuring users - authorised and unauthorised - cannot use a disk or file editor to look at the database files and so extract data
- Stopping database users from making changes to the database structure or code
- Making sure that only certain users can make changes to data
- Ensuring that different groups have appropriate access to different sets of data (for example, only letting users from an accounts department have access to salary details).

## *The Jet Database Engine*

Any understanding of the security mechanisms available in MS Access must be based on an understanding of how the underlying Jet database engine handles users. Microsoft Access uses the Jet engine to store and retrieve data, to perform queries and so on. In some respects it is helpful to think of Access as a front-end to the Jet engine, with Jet as the database. Naturally, security is an implicit part of Jet; it is central to its structure, rather than something that is particular to the Access front-end.

Security in the Jet database engine is workgroup-based rather than being tied to individual users. When the Jet engine is fired up it looks for a workgroup file - the default name of which is SYSTEM.MDW - which contains information about valid database users and groups. This workgroup file contains nothing more than that; it knows nothing about any other databases, it merely contains the account details of users and groups of users. Jet uses this workgroup information to validate users when they fire up Access or otherwise attempt to log on to a Jet database.

Once a user has successfully logged onto Jet they can be connected to a specific database. This database in turn may contain additional security information relating to specific objects such as database tables, queries, forms and so on. Logging onto a specific database application, therefore, is a two-stage process, with security measures at both stages. In practice, of course, for most users this two-stage process appears as a single seamless logon procedure.

By default every workgroup file contains two user groups: Admins and Users, and one user account called Admin. This Admin account has no password set by default. In a clean install of Access the only account that is defined is that of Admin

- which, as part of the Admins group has full administration rights and no password is required for it. Loading up Access and creating a new database requires no logon procedure in this case, giving the impression that no users or user groups exist. Any database that is created in this state will have Admin as the owner, and full administration rights are granted to that user. It is likely that many occasional users of Access will be extremely naþve about security, as they are presented with a non-secure system from day one.

This situation changes slightly once a password is set for the Admin account. From then on, firing up Access causes the logon dialog to appear. Only defined users can logon, even if they do not have a password set for them. Any attempt to load a database file will fail for users not included in the workgroup, or for users who have forgotten their password. Although this may give the impression that Access is now a secure system, nothing can be further from the truth.

Password-protecting the Admin account gives an extremely false sense of security and is actually to be discouraged. It is extremely simple to create a virgin SYSTEM.MDW file, which can then substitute for the SYSTEM.MDW file which contains the password. Furthermore, the password version of the file does not even need to be physically replaced; it is possible to start Access with a command-line option which points to the "clean" SYSTEM.MDW, thus easily bypassing your users' first line of defence. Before looking at procedures which guarantee a much stronger level of security we will explore what levels of security we can apply to objects within a database.

### *Permissions*

Each object in a Jet database - and here object can refer to a database, table, query, form, report or macro - has an owner. By default the user account which created

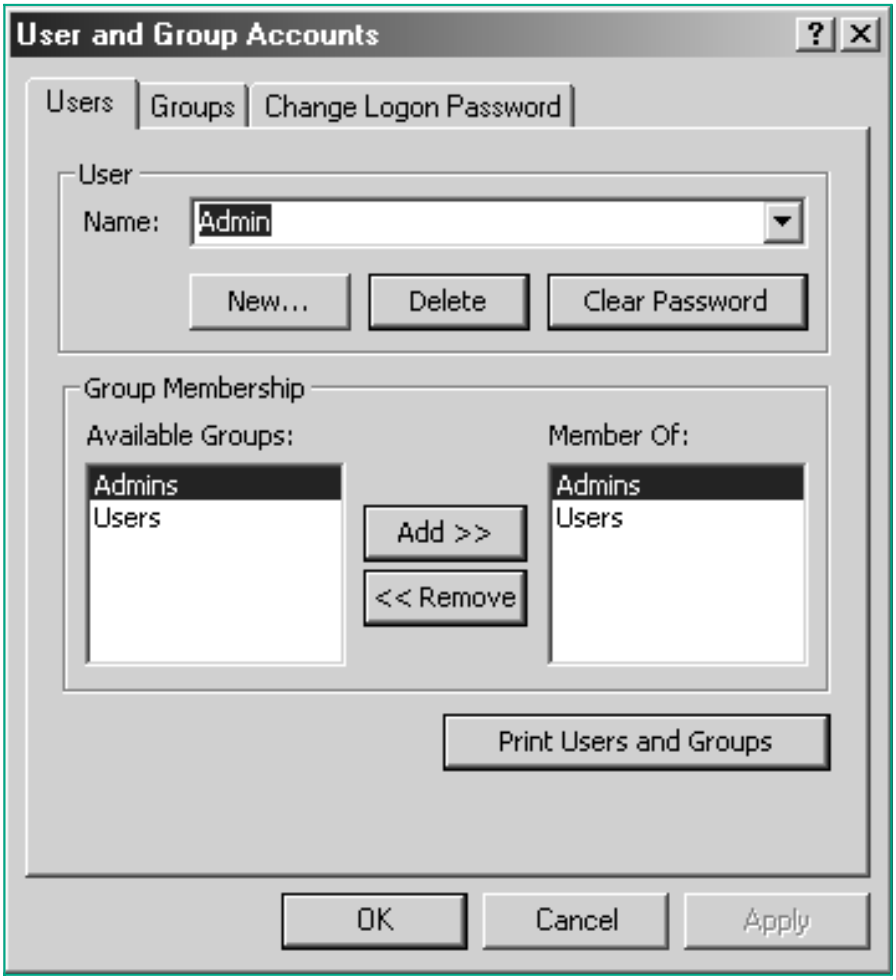


*Figure 1 - The Group Accounts dialog box in Access.*

*"All users have to be members of the Users group, so as soon as you add a new user they are automatically added to this group."*

the object is the owner, though it is possible to change the owner either by using Access menus or via some Visual Basic code. Note that ownership of an object, including a database, can only be assigned to an individual user - it cannot be assigned to a group. The owner of an object will always retain the power to change the permissions for that object - that is, to control who else can have access to that object, and what level of access they can have.

The full set of permissions available on an object can be divided up into those that give users access to data, those which give access to the database design and, most powerfully, Administer rights. The data permissions are: Read, Update, Insert and Delete. These permissions mean that you can fine-tune a user's access to the data in a database. Aside from revoking all rights to data, the most restrictive permission is simply to grant Read rights to it. Update means that changes can be performed on existing data records in the database, but new records cannot be added by that user, nor can existing records be deleted. Insert grants that user permission to add new records to the database, while Delete means that records can be deleted.

In addition to the data permissions, some users can be granted permissions to view the underlying database design and, if required, to change the design using the Read Design and Modify Design permissions. The Administer right means that a user has the right to change the permissions for any other user, and to change the ownership of any object. By default, the Admin account - and any other account in the Admins group - has Administer rights to every database, which is one of the reasons it is a potential security back-door into your users' databases.

These different levels of Permission can be assigned to either individual users or to groups. Permissions assigned directly to a user account are called "explicit"

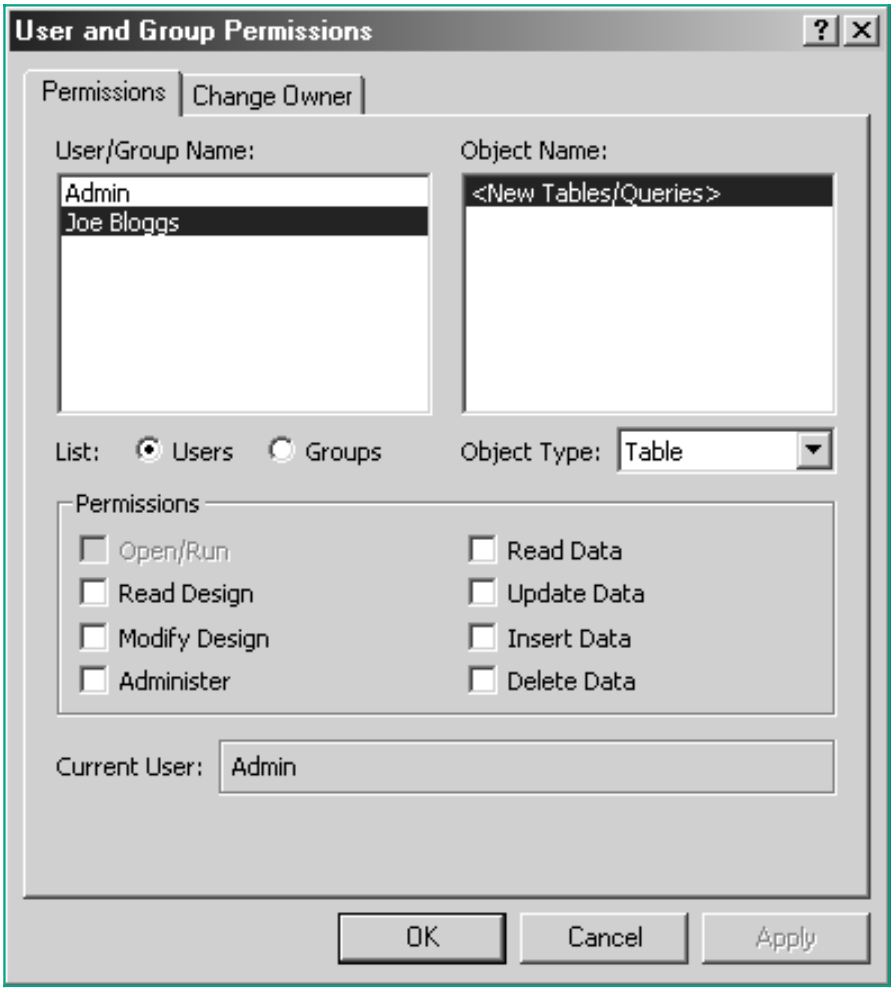


*Figure 2 - Group Permissions for Joe Bloggs.*

*"Once a file has been converted from an MDB file to an MDE there is no way of going back, therefore it is important that the original file is safely backed up."*

permissions, whereas if a user inherits them because he or she is a member of a group then they are called “implicit” permissions. Explicit permissions can be useful, but in situations where a database is going to have multiple users then the administration required to manage permissions for many individuals may become too much of an overhead. From an administration point of view access to a database should be seen in the same terms as access to the network - managing groups is to be preferred to managing everything in terms of individual rights. Where a user is a member of more than one group, the rights that he or she inherits are those which give the highest level of permission. For example, if a user is a member of a group which has Update and Insert Data permissions, and a member of a group which only has Read permission, then the rights which will apply are the Update and Insert rights.

It should be borne in mind that these permissions to database objects are stored with individual databases, they are not part of the workgroup file. As such they cannot be set to apply to every database that a user can log on to. It is important to understand that the workgroup file contains only the workgroup, user group and individual user definitions and passwords. Furthermore, although you create new users or groups when logged on to a specific database, their details are only stored in the SYSTEM.MDW workgroup file.

## *Creating New Users And Groups*

Having established the split between what exists in the workgroup file and the permissions which can exist in different database files, we can now move on to look at how we can create new workgroups, new user groups and new users. As mentioned previously, there is nothing to stop somebody from substituting your SYSTEM.MDW file with a “clean” one. How can a new workgroup file be created? Access comes with a workgroup administration program called WRKGADM.EXE, which can be used to create a new workgroup file. It is advisable at this stage to take a backup of any MDW file that you have installed on your system if you are going to try any of the following steps to create a new workgroup file.

Let’s assume that we wish to create a special workgroup file to be used exclusively for a database used by an accounts department. For the moment we will ignore the pros and cons of having different .MDW files rather than one central file, and go ahead and create a new workgroup file. The first step is to run the WRKGADM utility, which will prompt us to either join an existing workgroup file or to create a new one.

If we select the Create option then we are prompted for the following information: Name, Organisation and Workgroup ID. The name you enter will become the owner of the workgroup, and the workgroup ID will be used as a unique identifier for that workgroup. In this example we will call the workgroup ACCOUNTS01. Once you have entered this information you will be prompted to enter a path and file name for the workgroup file. The default is SYSTEM.MDW, but for this example we want to use this file for an accounts database, so let’s call the file ACCOUNTS.MDW. After confirming that the information entered is correct, the file is created. This file now becomes the default, and the next time that an Access database is loaded, Jet will use this file for the workgroup information. The information is stored in the system registry. To change the default workgroup file back to a previous version it is a simple matter to run the WRKGADM utility again, but this time hit the Join button and then point to the previous .MDW file.

Having created our new workgroup file we can now set up a new group and some new users. First we need to load Access, and while it is possible to load up any database, it is better that we load Access and start with an empty file. User and group administration is performed via the Tools|Security|User and Group Accounts menu item. If we select this option we are presented with a tabbed dialog box (see Figure 1). To add a new group click on the Groups tab and press the New button. Enter AccUsers as the name of our new group, and of course we must give it a unique ID. Using the same ID as the group name is not a good idea, nor is it a good idea when creating the workgroup or individual users. In every case the name is combined with the ID to create a unique token which acts as a key to that object. If somebody attempts to recreate the group (or workgroup or user) with the same name, but uses a different ID, then they will fail as Jet will generate a different token for that object.

Adding a new user is pretty much the same procedure, but you'll notice that there is a "group membership" pane to the Users dialog box. All users have to be members of the Users group, so as soon as you add a new user they are automatically added to this group. To add your new user to the AccUsers account you have to explicitly select the group.

To set the different permissions for your group of users you select the Tools | Security | User and Group Permissions option. For our new user - Joe Bloggs - you can see that by default he has no permissions granted (see Figure 2). Below the User / Group Name is an option button to switch views between Users and Groups, and it is here that we can set the permissions for a group as a whole. Select the type of object you wish to set permissions for, select the different objects of that type which are listed, and then use the checkboxes to assign the appropriate permissions. Doing this for many users and many objects can become a nightmare, but as was mentioned previously; administering groups is easier than administering individual user accounts.

Note that if Access cannot find your new MDW file it will look for a SYSTEM.MDW file instead; if one isn't available then Access will refuse to load as the Jet engine cannot validate users. For this reason it is good practice to make sure that rogue copies of SYSTEM.MDW do not exist on users' hard disks or, even worse, on a LAN.

## *Types Of Security*

Having looked at the information contained in the workgroup file and the permissions available in a database, and examined how we can apply permissions to different objects, we can now start to discuss the steps necessary to create a more secure environment. The first question we need to address, however, is what it is we wish to achieve. If we are only interested in stopping unauthorised users from accessing a database, and we are happy to let authorised users have free reign once they are logged on, then we can apply what is called "share-level security". Simply put, this means password-protecting the database itself. This is a separate option to having passwords set for individuals in the workgroup file. Here we are talking about adding a password to the Access database file itself.

There are many situations, however, where this relatively simplistic form of security will not suffice. In such situations a properly secured database is required, which means using a combination of workgroup settings and permissions. This form of security is referred to as "user-level security", and although there are tools within Access to help implement it, a full understanding of it can only be gleaned by going through all of the steps required manually.

*"An alternative approach, which can be taken if protection of the application rather than the data is the primary requirement, is to look at porting the code, forms and so on to a standalone Visual Basic application rather than Access itself."*

## *Share-Level Security*

Share-level security - password-protecting an individual database file - can be implemented with and without using the workgroup file. To add a password to a database it needs to be opened for "Exclusive Use" first. Once this is done, the Tools | Security | Set Database Password menu option allows the setting of a password. Similarly, there is a Tools | Security | Unset Database Password menu option in password-protected databases. In an environment where user accounts have been defined in the workgroup file and users need to log on to Access, they will have to enter two passwords in order to gain access to the database. The first will prompt for user name and password for the workgroup and then they will be prompted for the specific database password. Where the Access default applies and no Admin password has been set in the SYSTEM.MDW file, users will simply have to enter the database password.

Although this form of security is extremely simple to implement, two things need to be borne in mind. The first is simply that anyone who knows the password can load the database and then have access both to its data, its design and any code that sits inside it. Accidental or malicious damage to the database is thus a serious possibility. If a database with confidential data is secured using share-level security, then at the very least the password should be changed at regular intervals to maintain some degree of security. A more serious concern is that the password is encrypted and stored within the database file. This makes the file open to attack by hackers or other hostile parties. Programs exist - such as

AccessPassword, AccessPasswordPro and so on - which can recover passwords from protected files. This means that confidential information cannot be considered secure in a password-protected Access database. By the same token this means that, should any of your users forget their passwords, all is not lost and steps can be taken to recover the password and hence the data in a database. Finally, password-protecting a database means that it cannot be replicated, as synchronisation cannot take place if passwords are defined.

### *User-Level Security*

The aim of this form of security is to utilise both the workgroups and the database permissions to produce a system that is as secure as possible given the facilities available within Access. When undertaking to implement the security procedures that follow it is important to fully document all passwords, user and group IDs and so on. It is also a good idea to take a backup of the database you wish to secure in its unsecured state. If the worst comes to the worst then at least you have an unsecured version to return to.

The first step is to produce a new SYSTEM.MDW file. Run the WRKGADM program and make a note of the Name, Organisation and WorkGroup ID that you use when creating the new file. These variables are used by the program to create a unique identifier for the workgroup. If the workgroup file ever becomes corrupted it will be possible to recreate a new copy by running WRKGADM again and entering the same information. Once you have created the new workgroup file load any database or create a new one. You will be logged on as the default user Admin, and no password will be set for you. The first step then is to activate security by using Tools | Security | User and Group Accounts to set a password for Admin. Because the Admin account is inherently risky due to the defaults it inherits (and the fact that every new workgroup file contains an Admin account), we need to replace it with a new database administrator account.

After adding a password to the default Admin account, create a new user account - taking care to document the user name and the personal ID - and add it to the Admins group. As you are logged in as Admin you cannot assign a password for this new account, so exit Access and then log in to the database as the new admin user and set a password. Again, document this somewhere safe. The next step is to remove the Admin account from the Admins group. Ideally completely deleting the Admin account would make a system even more secure, but Access does not allow the deletion of the built-in Admin account (or the built-in Admins and Users groups, come to that). At this point the Admin account still has the permissions that it inherited for the database. For example, if the database had been created by an Admin account then that account is still the owner of the database and has free reign to make changes willy-nilly, as well as to grant or revoke permissions for other users. It has lost the power to add or delete users and to change group memberships, however.

*"If we are only interested in stopping unauthorised users from accessing a database, and we are happy to let authorised users have free reign once they are logged on, then we can apply what is called share-level security. This means password-protecting the database itself."*

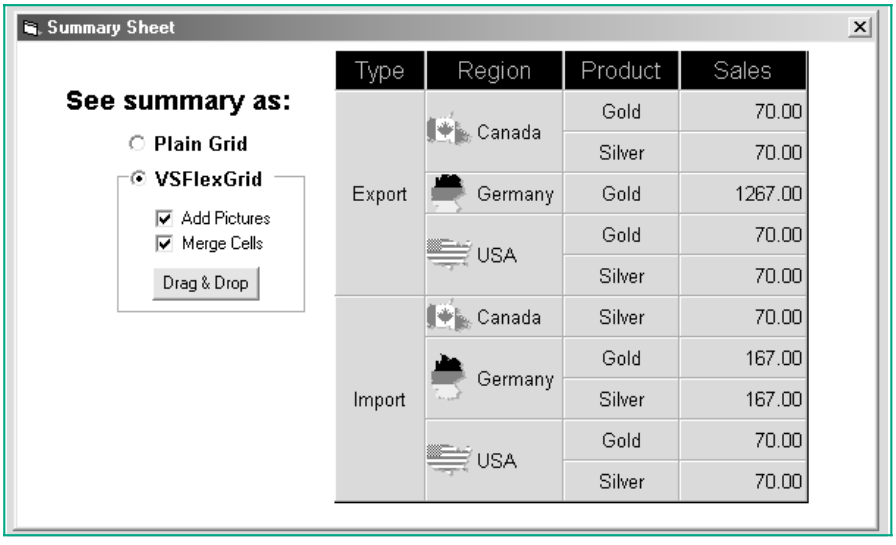


*Figure 3 - VideoSoft's VSFlexGrid Pro.*

Having created a new administrator account, we can now turn our attention to the database itself, rather than the accounts stored in the workgroup file. Remembering that the owner of a database always retains full Administer rights, we need to change ownership of the entire database to that of our new administrator account. The easiest way to do this is to use the new account to create an empty, blank database. Once this has been done the target database we wish to secure can be imported into it. Use the File | Get External Data | Import menu option and point to the database to be secured. Select all the objects within that database and import them into the new database. The new database, now functionally equivalent to the one you wish to secure, has the new administrator account as the owner of every item. Before proceeding any further save this database, taking care not to overwrite or delete the original, unsecured version.

As an additional security measure at this stage it is a good idea to encrypt the database. Close the database and then use Tools | Security | Encrypt / Decrypt Database to encrypt the database file. An encrypted database means that a casual hacker cannot simply view the database file with a disk editor and extract data by that means.

The task now is to create a set of groups which will encompass all the types of user we expect to use the database. Typically there will be some users who have read-only rights to the data, others who can make changes to data but not to the design, developers who can modify the design, and perhaps even a group who have full access to the data and the design but who cannot, unlike the members of the Admins group, make changes to other user accounts. At this stage it is not necessary to actually populate these groups; it is enough to create them for now. It is important to decide which permissions these groups will have for each of the database objects. These can be set for each group / object using the Tools | Security | User and Group Permissions menu option. Given even a moderately complex database this task can be both time-consuming and prone to error, but it does need to be done. Once the groups have been created, and the permissions set, then the users can be added as and when required.

At this point we can finally say that we have secured an Access database. It is safe from people loading it up using a dummy workgroup file, safe from unauthorised users making changes to data and design, safe from people using file editors to extract data and safe from users not registered in the workgroup. Having secured the database it is a good idea now to store the unsecured version on diskette or CD, along with a note of the user names, IDs and passwords used in the security procedure, and to place these in a safe location in case they are needed in the future.

*"In addition to the data permissions, some users can be granted permissions to view the underlying database design and, if required, to change the design using the Read Design and Modify Design permissions."*

### *User-Level Security Wizard*

The procedure as outlined above is both complex and prone to error, but it does illustrate the steps needed to secure an Access database. However, in practice Access provides a Wizard to guide you through the process; it covers every step in the procedure, from generating a new workgroup file to saving an unsecured copy of the database to encrypting the finished secure version. The Wizard is accessed via the Tools | Security | User Level Security Wizard menu option.

The Wizard even includes a number of predefined user groups which may make the task even simpler. Any groups that you have already created for your database will automatically be picked up by the Wizard. It will also suggest that the Users group, to which every user account is automatically joined, is granted no permissions at all. However, it may be that you wish the Users group to have minimal permissions as this may simplify the administration process later on. Additionally the Wizard helps in the process of adding individual users to the workgroup. Not only does it allow you to create new user accounts and assign them to the groups which you have created, it also allows you to assign a password to each of these users.

### *Securing Forms And Code*

If securing the design of the forms, reports or VBA code used in an Access database application is of concern then there are a number of steps which can be taken - with or without the user-level security measures discussed previously.

The obvious question to ask is: why not use the permissions to secure forms and code? Although it is possible to use the permissions to secure forms and reports, Visual Basic modules are not covered by any of the permissions. Secondly, unless you expressly remove the Read Design and Modify Design permissions, there may be some users who can look at the design of your forms and the VBA code which sits behind them.

To remedy this situation and to provide a higher level of security for applications developers, Access is able to compile all source code and remove it from the file, leaving behind only the executable code. This process converts the database file from an MDB file to an MDE file. This new database file contains the compiled code for VBA modules and code behind forms and reports, and removes the ability to add new code, forms and reports. Users are still able to design and modify tables, queries and macros, however.

Once a file has been converted from an MDB file to an MDE there is no way of going back, therefore it is important that the original file is safely backed up. It also makes sense to split the database application into a front-end database which contains the forms, reports, code and so on, and a back-end which contains the data tables. Access provides a function to perform this split between front-end application design and back-end data via Tools | Database Utilities | Database Splitter. The front-end database is connected to the data via linked tables, ensuring that it performs as before. To convert a database to an MDE file, load it in exclusive mode and then select Tools | DatabaseUtilities | Make MDE File. The original file is copied, the code compiled and the permissions and menu options changed so that the reports and forms are protected.

An alternative method of protecting VBA code is to load the Visual Basic Editor (alt-F11 in Access 97 and Access 2000), select Tools | <database> Properties (where <database> is the name of the database you are securing), and select Protection from the tabbed dialog box. Here you can click on the Lock Project For Viewing checkbox to stop users viewing VBA modules, and there is also an option to add a password to protect the code.

There is an alternative approach which can be taken if protection of the application rather than the data is the primary requirement, and that is to look at porting the code, forms and so on to a standalone Visual Basic application rather than Access itself. The move towards fully implementing VBA in the latest releases of MS Access has made this task even easier. By taking the code and the forms out of the database completely, not only do these objects become more secure, it also facilitates the use of additional tools and techniques. For example, there are additional data grid controls, including VideoSoft's excellent VSFlexGrid Pro (see Figure 3), which offer functionality far in advance of those supplied as part of Microsoft Access. VSFlexGrid also supports ADO and OLEDB support as well as DAO, and can therefore be used for all types of application development. Drag and drop, the ability to easily merge cells, automatic totalling and so on mean that the application is freed from the constraints of the controls supplied by Microsoft.

Where the front-end application is built on a pure Visual Basic platform, the data still resides in an MDB database file. Similarly, for an Access database using a front-end MDE file, the data sits in a back-end MDB file. In both situations steps need to be taken to secure the MDB file using the methods detailed earlier.

*"Microsoft Access uses the Jet engine to store and retrieve data, to perform queries and so on. In some respects it is helpful to think of Access as a front-end to the Jet engine, with Jet as the database."*

## *Conclusion*

As can be seen, Access supports a number of different security schemes, providing a flexible security approach. The key element which drives the security process has to be the set of user requirements. An overly restrictive security system, using a database password, user-level security and an MDE file might be overkill when all that is required is a method of ensuring that only a few users can access the database. Similarly, adding a password to secure a database which contains highly confidential data cannot substitute for a user-level secured database. In every case it is important to draw up the requirement first and then to look at the best method for implementation.

**PCNA**