

Slide 2

How Secure Is It?

Microsoft Access is designed to be the most secure desktop database management system you can buy. However, it has no security rating with the U.S. government or any other certifying body, and it is not *guaranteed* by Microsoft to be secure. Skilled hackers with enough time and computing resources, and a desire to break into your database, could crack Microsoft Access security. If you have applications that require absolute security, you should consider using a server database such as Microsoft SQL Server™ on the Microsoft Windows NT™ operating system, and a compiled application programming language such as Microsoft Visual C/C++®.

Slide 3

User-Level Security vs. Share-Level Security

Microsoft Access 95 now provides both share-level security and user-level security. In a share-level security system, objects are assigned passwords, and anyone who knows the password can access the object. Microsoft Access 95 supports share-level security with a database password.

When Is Security "On?"

It's important to realize that Microsoft Access security is always "on" -- that is, every time a user performs any action, Microsoft Access first checks to make sure the user has permissions to perform that action. However, most Microsoft Access users never realize they are logging on and never see a security-related message. How does this happen? The illusion that security is not "turned on" is created by granting full permissions, by default, to the Users group (all users) on all objects and by automatically logging on each person as a default user (Admin) without displaying a **Log on** dialog box. Until a developer or administrator takes explicit action to expose the security subsystem, most users will never notice that it exists. This allows Microsoft Access to be extremely secure, with no "backdoor" modes of operation, but still keeps security virtually invisible to users who don't need it.

How can I set a single password on my database?

Since Microsoft Access 95, Microsoft Access has supported share-level security with a database password. You can find this feature under **Tools | Security | Set Database Password**. In order to set the database password, you must be a member of the Admins group or the database owner, and have the database open exclusively. The database password is not supported in a replicated database. One danger is that if you set a password and then forget it, you (and everyone else) can be locked out of the database.

Slide 4 / 5

Look at slides.

Slide 6

Unlike most other desktop database management systems, however, Microsoft Access also provides user-level security. In a user-level security system, users are authenticated when they start Microsoft Access by logging on with a password. Administrators grant specific permissions, such as Read Data or Modify Design, to specific users and groups on specific objects. Different users can have different permissions on the same objects.

For example, suppose you have an Employees database containing a Salary table. If you simply want to disallow some users from opening the database, but all users who are authorized to open the database are

allowed to do anything they want once inside the database, assign a password to the database. Users won't have to log on to Microsoft Access, and your administration is quite simple -- just change the password periodically.

However, suppose now that you have two groups of people who need access to your Employees database: managers, who can update the Salary table, and payroll personnel, who can view but not update the table. By implementing user-level security, you can assign Update Data permissions to the managers group and Read Data permissions to the payroll group. Then you make sure that people logging onto your system are enlisted in the appropriate groups. The individuals manage their own passwords, and their passwords are used to verify their identity rather than having any direct effect on their permissions.

When working with user-level security, it's important to remember that there's no such thing as a permission on an object that exists all by itself -- permissions on objects are always granted *to* users and groups. In a Microsoft Access database, it doesn't make sense to say, "the Salary table has Read Data permissions" because no user is indicated. It does make sense to say "the managers group has Read Data permissions on the Salary table."

How Does User-Level Security Work?

In a nutshell, security functions as follows: Administrators assign specific permissions on database objects to users and groups. When a user starts up Microsoft Access in a secure environment, she logs on, usually with a password. The password's function is to authenticate the user -- to prove that she is who she says she is. Microsoft Access checks and remembers all the groups to which that user belongs. Every time the user tries to perform an action on an object, such as open a form, browse a table, or modify a query, Microsoft Access checks to see if the user *or any of the groups to which the user belongs* have the necessary permission. If so, Microsoft Access performs the action. If not, Microsoft Access tells the user that he or she doesn't have permissions to perform the requested operation, and the operation fails.

Slide 7

1. , how does Microsoft Access security work?

The Microsoft Jet database engine, which Microsoft Access uses to store and retrieve its objects and data, employs a workgroup-based security model. Every time the Jet database engine runs, it looks for a workgroup file, which holds information about the users and groups of users who can open databases during that session. Any valid file name can be used, such as Wrkgrp_Sec.mdw.

The workgroup file contains the names and security IDs of all the groups and users in that workgroup, including passwords. There are built-in groups (Admins and Users) and a generic user account (Admin) that every workgroup contains by default. The built-in group Guests and user account Guest, which are included in Microsoft Access 2.0 only, can safely be ignored. You can add new groups and new user accounts using Microsoft Access menus or through code.

The Admins group is always present and its users have Administer rights that cannot be revoked. You can remove rights from the Admins group through the menus or through code, but any member of Admins can assign them right back. There must always be at least one member in the Admins group to administer the database. The default user account, Admin, always starts out as a member of the Admins group and is the account that everyone logs on as by default in an unsecured database. The other built-in group, Users, is a generic group to which *all* users must belong, no matter which other groups they belong to. It is possible to create a user through code, but that user is not automatically added to the Users group. If you do not take the extra step to add the person to the Users group, the person will not be able to start Microsoft Access because many of the tables that Microsoft Access uses internally are mapped to the permissions of the Users group. Neither the Admin user account nor the Users group has any built-in permissions (as the Admins group does).

Securing a database involves adding a new member to the Admins group and removing the Admin user from that group, removing permissions from the Admin user and from the Users group, and assigning permissions to the custom groups that you define.

Permissions to various objects in Microsoft Access can be assigned directly to users (explicit permissions) or to groups. Users inherit permissions from the groups they belong to (implicit permissions). Microsoft Access employs the "least restrictive" rule: users have the sum total of their explicit and implicit permissions. In other words, if a user belongs to a group that has full permissions and you make that user a member of a group that has restricted permissions, the user will still have full permissions because he is still a member of the unrestricted group. Although Microsoft Access allows you to assign permissions directly to users, this is not recommended. Administering your database can become very difficult if you do.

User and group information, including passwords, is saved in the workgroup file, or System.mda/mdw, which validates user logons at startup. Permissions to individual objects are saved in the database itself. You can give the groups and users within a workgroup various levels of permission to view, modify, create, and delete the objects and data in a database. For example, the users of a particular group might be permitted to read only certain tables in a database and not others, or you could permit a group to use certain forms but not to modify the design of those forms.

Setting a password for the default Admin user account activates the logon dialog box so that users will be prompted for a valid user ID and password each time that they start Microsoft Access. If you never set a password, all users will be logged on as the Admin user (with no password) and you will never see the logon dialog box when starting Microsoft Access. So even though it may appear that there is no security present, it is just transparent until you set a password on the Admin user account.

The database password was introduced in Microsoft Access 95. This is a simple password on the database itself that allows only users who know the password to open the file. You cannot assign permissions to users or groups with this feature. In addition, the database password feature is not considered to be very secure.

Slide 8

Where Is Security Information Stored?

Microsoft Access stores its security information in two different places. User and group information is stored in a workgroup database, named by default **System.mdw**. The location of this database is specified in the Windows registry or with the /wrkgrp command-line switch. The workgroup database stores which users and groups exist, which users belong to which group, logon passwords (encrypted, of course), and the internal security identifier (SID) for each user and group. Workgroup databases are created by the Workgroup Administrator. The **User and Group Accounts** command on the **Security** submenu of the **Tools** menu changes data in the workgroup database. A single workgroup database can be used by many application databases.

On the other hand, all the permissions that users and groups have on objects in a database are stored in the database itself. After a user's identity and group membership have been established by checking the logon strings against data in the workgroup database, all permission checking is done against system tables within the user database. The **User and Group Permissions** command on the **Security** submenu of the **Tools** menu affects data in the open database, not the workgroup database. If you move objects from one database to another, their security settings travel with them.

Slides 9-11

Look at slides

Slide 12 –13

2. How do I implement field-level or row-level security on my tables (RWOP or queries with Run Permissions set to Owner's)?

To understand how the Run permissions work you must understand the concept of ownership in Microsoft Access. The user who creates or imports an object in Microsoft Access becomes the owner of that object and has full permissions to administer that object. For example, if I create a query that draws on certain tables, then I must have rights to the data in those tables or I would not be able to create, run, and save the query. The **RunPermissions** property in the query is by default set to **User's**, which means that the query will run with whatever permissions the logged-on user has on the base tables that make up the query. If I change the **RunPermission** property to **Owner's**, I allow users to run the query as though they were logged on as me, the owner of the query. The query will run with the permissions of the query owner, rather than with the permissions of the logged-on user. Such queries are often referred to as RWOP queries (Run With Owner's Permissions).

The **RunPermissions** property applies only to saved queries. You cannot use it with SQL strings defined in your module code by the simple expedient of typing WITH OWNERACCESS OPTION in the string. The reason for this is that security is defined on saved objects only by setting certain bits on them. Your module code has no way of knowing who you want the owner to be in this situation because there is no saved object to read those bits from. You can use RWOP queries to grant partial access to tables by restricting either the columns or the rows returned by the query. Therefore, if you have a Salary column in your payroll table, you can design a query for your users to run that returns all the columns except Salary. If you wanted your users to see all salaries except managers salaries, you could design a query that restricted the rows returned by using a **WHERE** clause that excluded those with a managerial job classification. The best way to implement this is to remove ALL permissions from the underlying table(s) and use only queries to get at the data you want your users to have. You then grant the appropriate level of permissions to the groups or users only on the query itself, allowing them only to view data or to be able to modify it through your query.

Note There are other issues to be considered when using RWOP queries with attached tables. See Section **Error! Reference source not found.**, **"Error! Reference source not found."**, Section **Error! Reference source not found.**, **"Error! Reference source not found."**, and Section 14.1, "Using RefreshLink to relink tables" for more information.

3. What happens when the front-end database permissions on an attached table differ from those in the back-end database?

Microsoft Access uses a combination of security settings in both the current database and remote database when ascertaining rights to attached/linked tables. Permissions set on the back-end tables cannot be over-ridden by granting more liberal permissions on the front-end tables. In this context, it is helpful to think of the attachments in the front-end database as merely the connection strings, or the information needed to locate the actual tables in the back-end database.

Permissions set on connection strings should not, and do not, override permissions set on actual tables.

Microsoft Access does not treat the combination of permissions between the base tables and the connection strings by using the "least restrictive" rule, but rather it determines the most restrictive rights. Microsoft Access first determines the least restrictive rights for the table in both the current and remote databases. Then Microsoft Access compares these permissions and combines them so that, unless a permission is available in both databases, it will restrict rights to the attached table.

The following table highlights this.

Front-End DB	+	Back-End DB	=	Permissions on Attached/Linked Table
None	+	Administer	=	No Permissions
Read Only	+	Administer	=	Read Only
Administer	+	None	=	No Permissions
Administer	+	Read Only	=	Read Only

The simplest thing to do is to grant users full permissions on tables in the front-end and restrict the back-end to the barest minimum needed if users need to relink tables. If they don't need to relink tables, then removing ALL permissions and using RWOP queries will give you the highest level of data security possible.

Slide 14 -23

Each user and group has associated with it a security identifier (SID). The SID is a machine-generated, nonreadable binary string that uniquely identifies the user or group. When a user logs on, Microsoft Access looks in the MSysAccounts¹ table of the workgroup database for a user of the same name (case insensitive). If a user with the same name is found, it then validates the password (case sensitive). If the password matches, the SID of the user is retrieved and saved in an internal structure. The password is only used to validate users when they log on. It has no other effect on security.

By default, Microsoft Access first attempts to log on as the user Admin, with a blank password. If this log on fails, the user is presented with the **Log on** dialog box. If a user name and password were specified on the command line (using the /User and /Pwd flags), Microsoft Access first tries to log on using that user name and password. If this log on fails, the user is presented with the **Log on** dialog box. Once logged on, the user's SID is retrieved. Then the SIDs of all groups to which that user belongs are retrieved and saved in the same internal structure. These SIDs are used for all subsequent security operations within Microsoft Access.

What are the steps to secure a database?

The process to secure a database is the same, no matter which version you use. The only differences are: beginning with Microsoft Access 95, the Security Wizard is built into the product and in Access 2000 the Security Wizard can perform all of the steps for you, including creating a new workgroup information file. You may elect not to use the Security Wizard and to secure the database manually by following these steps.

¹Disclaimer: This document refers in several places to the Microsoft Access system tables (those prefixed with MSys*). In spite of this, these tables are officially "undocumented" and are subject to change in future versions. Any applications that read directly from system tables are likely to fail in future versions of Microsoft Access.

1. Use the Workgroup Administrator program (Wrkgadm.exe) to create a new workgroup information file. Write down the **Name**, **Organization**, and **WorkGroup ID** strings that you will be prompted for when you create your new workgroup information file and store them in a safe place. If your workgroup information file ever becomes lost or corrupted, you can reconstruct it by using these identical strings, which are then encrypted to create a unique token. Without a valid workgroup information file, you could conceivably be locked out of your database forever. Another reason to save this information is for upgrading a secured Access database to a newer version of Access. The recommended path for upgrading databases is to re-create the workgroup file in the new version of Access before upgrading the database itself.
2. The Workgroup Administrator automatically switches you to the new workgroup information file. Start Access, and open any database.
3. You will be logged on as a user named Admin. Use the **Security** menu options to add a password for the Admin user. The Admin user is the default account, and setting its password is what causes Access to prompt for a logon Name and Password the next time that you start Access.
4. Create a new user, which is the account you will use to secure the database. Add this new user to the Admins group. Write down the strings that you use for the name and PID in case you ever need to re-create your workgroup information file. The PID is not the password—the string used for PID is encrypted, along with the string used for the Name, to create a unique token (SID, or system identifier) identifying the user.
5. Quit Microsoft Access and log back on as the new user account that you created in step 4. You will not have a password for this account yet, (the PID you typed with the name in step 4 is **not** the password), so now is a good time to set one.
6. Remove the Admin user from the Admins group so that Admin is a member only of the Users group. The Admin user account has no administrative powers built into it; they are derived from membership in the Admins group, which does. Although you cannot delete any of the built-in users or groups (Admin, Admins, and Users), you can move users to and from the Admins group and restrict permissions to the Users group.
7. Open the database that you want to secure and run the Security Wizard. Select the objects that you want to secure (it makes sense to secure them all). The wizard will then create a new database owned by your new user, and will import all of the objects and relationships into it. It will also remove all permissions from the Admin user and the Users group and encrypt the new database. The original database will not be altered. Note that the Access 2000 security wizard does not create a new database—it simply creates a backup copy of the original. One flaw with this arrangement is that not all permissions to open the database are removed from the Admin user and Users group to open the database, even though they appear to have been removed.
8. Open the new database. Because the Security Wizard removed all permissions from the Users group for the secured objects, you need to create your own custom groups and assign the level of permissions needed to these groups. Every user is required to be a member of the Users group (otherwise, a user would not be able to start Microsoft Access), so only grant permissions to Users that you want everyone to have. Members of the Admins group have irrevocable power to administer database objects, so make sure to limit membership in the Admins group to only those users who are administrators.
9. Create your own users and assign them to the groups that reflect the level of permissions that you want them to have. Do not assign permissions directly to users because that is extremely hard to administer. Users inherit permissions

from the groups they are members of, and keeping track of the permissions assigned to a group is much easier than keeping track of the separate permissions of individuals. If a user is a member of multiple groups, then that user will have all of the permissions granted to any of those groups plus any permissions assigned specifically to the user (this is known as the "least restrictive" rule). There is no way to deny permissions to a user if that user is a member of a group that has been granted those permissions. If you need to create specific permissions for only a single user, create a group for that user and assign the permissions to the group; then, add the user to the group. The reason for this becomes clear when you consider that the user may quit, and you may have to set up permissions for the replacement on short notice.

10. Additionally, you may need manually to remove the **Open/Run** permission from the database container for the Users group through the security menus or through code. This will prevent someone from opening the database by using another workgroup information file or the default System.mda/mdw. In Microsoft Access 97, the User Level Security Wizard is supposed to remove the **Open/Run** database permissions for the Users group, but fails to do so. The Access 2000 Security Wizard removes permissions to the point where they are not visible on the security menus, but testing has revealed that in Access 2000 it is possible to open a database by using the default workgroup information file regardless of the menu settings. The cure for both versions of Access is to create a new, empty database while logged on as a member of the Admins group and import all of the objects from the secured database. You should take this step before spending too much time securing objects because Access considers imported objects to be "new" and loses the permission information that was stored in the source database.

The following table lists the default names and locations of the workgroup file and the Workgroup Administrator program.

Version	Workgroup File Name (default)	Workgroup File Location	Wrkgadm.Exe Location
2.0	System.mda	C:\Access	C:\Access
95	System.mdw	C:\MSOffice\Access	C:\MSOffice\Access
97	System.mdw	C:\Windows\System	C:\Windows\System
2000	System.mdw	\Program Files\Common Files\System	\Program Files\Microsoft Office\Office\1033 Note: 1033 is the default folder for the English version of Access

Slide 20

4. The Security Wizard

The Microsoft Access Security Wizard is by far the easiest way to secure databases. It allows you to choose which object types to secure -- tables, queries, forms, reports, macros, or modules. The selected object types are secured by revoking all permissions from all accounts except that of the user running the wizard. Beginning with Microsoft Access 95, the Security Wizard is included in the retail product.

To secure a database, log on as a user other than Admin who has read permissions on all objects and data in a database. Then open the database and invoke the Security Wizard by clicking **User-Level Security Wizard** on the **Security** submenu of the **Tools** menu. The Security Wizard creates a new secured copy of

the database. It will leave the original copy unmodified. The Security Wizard can be used to secure databases in Microsoft Access 95 format.

Slide 24 -26

Security Manager screens

Slides 27-29

How can I secure just my code without users having to log on?

This works by using two separate workgroup information files: one for development and securing your database, and one for distribution. You can even use the default System.mda/mdw for distribution.

1. Secure your database completely by following the steps discussed in Section 0, "Slide 2
2. How Secure Is It?
3. Microsoft Access is designed to be the most secure desktop database management system you can buy. However, it has no security rating with the U.S. government or any other certifying body, and it is not *guaranteed* by Microsoft to be secure. Skilled hackers with enough time and computing resources, and a desire to break into your database, could crack Microsoft Access security. If you have applications that require absolute security, you should consider using a server database such as Microsoft SQL Server™ on the Microsoft Windows NT™ operating system, and a compiled application programming language such as Microsoft Visual C/C++®.

Slide 3

User-Level Security vs. Share-Level Security

Microsoft Access 95 now provides both share-level security and user-level security. In a share-level security system, objects are assigned passwords, and anyone who knows the password can access the object. Microsoft Access 95 supports share-level security with a database password.

When Is Security "On?"

It's important to realize that Microsoft Access security is always "on" -- that is, every time a user performs any action, Microsoft Access first checks to make sure the user has permissions to perform that action. However, most Microsoft Access users never realize they are logging on and never see a security-related message. How does this happen? The illusion that security is not "turned on" is created by granting full permissions, by default, to the Users group (all users) on all objects and by automatically logging on each person as a default user (Admin) without displaying a **Log on** dialog box. Until a developer or administrator takes explicit action to expose the security subsystem, most users will never notice that it exists. This allows Microsoft Access to be extremely secure, with no "backdoor" modes of operation, but still keeps security virtually invisible to users who don't need it.

How can I set a single password on my database?

Since Microsoft Access 95, Microsoft Access has supported share-level security with a database password. You can find this feature under **Tools | Security | Set Database Password**. In order to set the database password, you must be a member of the Admins group or the database owner, and have the database open exclusively. The database password is not supported in a replicated database. One danger is that if you set a password and then forget it, you (and everyone else) can be locked out of the database.

Slide 4 / 5

Look at slides.

Slide 6

Unlike most other desktop database management systems, however, Microsoft Access also provides user-level security. In a user-level security system, users are authenticated when they start Microsoft Access by logging on with a password. Administrators grant specific permissions, such as Read Data or Modify Design, to specific users and groups on specific objects. Different users can have different permissions on the same objects.

For example, suppose you have an Employees database containing a Salary table. If you simply want to disallow some users from opening the database, but all users who are authorized to open the database are allowed to do anything they want once inside the database, assign a password to the database. Users won't have to log on to Microsoft Access, and your administration is quite simple -- just change the password periodically.

However, suppose now that you have two groups of people who need access to your Employees database: managers, who can update the Salary table, and payroll personnel, who can view but not update the table. By implementing user-level security, you can assign Update Data permissions to the managers group and Read Data permissions to the payroll group. Then you make sure that people logging onto your system are enlisted in the appropriate groups. The individuals manage their own passwords, and their passwords are used to verify their identity rather than having any direct effect on their permissions.

When working with user-level security, it's important to remember that there's no such thing as a permission on an object that exists all by itself -- permissions on objects are always granted *to* users and groups. In a Microsoft Access database, it doesn't make sense to say, "the Salary table has Read Data permissions" because no user is indicated. It does make sense to say "the managers group has Read Data permissions on the Salary table."

How Does User-Level Security Work?

In a nutshell, security functions as follows: Administrators assign specific permissions on database objects to users and groups. When a user starts up Microsoft Access in a secure environment, she logs on, usually with a password. The password's function is to authenticate the user -- to prove that she is who she says she is. Microsoft Access checks and remembers all the groups to which that user belongs. Every time the user tries to perform an action on an object, such as open a form, browse a table, or modify a query, Microsoft Access checks to see if the user *or any of the groups to which the user belongs* have the necessary permission. If so, Microsoft Access performs the action. If not, Microsoft Access tells the user that he or she doesn't have permissions to perform the requested operation, and the operation fails.

Slide 7

5. , how does Microsoft Access security work?

The Microsoft Jet database engine, which Microsoft Access uses to store and retrieve its objects and data, employs a workgroup-based security model. Every time the Jet database engine runs, it looks for a workgroup file, which holds information about the users and groups of users who can open databases during that session. Any valid file name can be used, such as Wrkgrp_Sec.mdw.

The workgroup file contains the names and security IDs of all the groups and users in that workgroup, including passwords. There are built-in groups (Admins and Users) and a generic user account (Admin) that every workgroup contains by default. The built-in group Guests and user account Guest, which are included in

Microsoft Access 2.0 only, can safely be ignored. You can add new groups and new user accounts using Microsoft Access menus or through code.

The Admins group is always present and its users have Administrator rights that cannot be revoked. You can remove rights from the Admins group through the menus or through code, but any member of Admins can assign them right back. There must always be at least one member in the Admins group to administer the database. The default user account, Admin, always starts out as a member of the Admins group and is the account that everyone logs on as by default in an unsecured database. The other built-in group, Users, is a generic group to which *all* users must belong, no matter which other groups they belong to. It is possible to create a user through code, but that user is not automatically added to the Users group. If you do not take the extra step to add the person to the Users group, the person will not be able to start Microsoft Access because many of the tables that Microsoft Access uses internally are mapped to the permissions of the Users group. Neither the Admin user account nor the Users group has any built-in permissions (as the Admins group does).

Securing a database involves adding a new member to the Admins group and removing the Admin user from that group, removing permissions from the Admin user and from the Users group, and assigning permissions to the custom groups that you define.

Permissions to various objects in Microsoft Access can be assigned directly to users (explicit permissions) or to groups. Users inherit permissions from the groups they belong to (implicit permissions). Microsoft Access employs the "least restrictive" rule: users have the sum total of their explicit and implicit permissions. In other words, if a user belongs to a group that has full permissions and you make that user a member of a group that has restricted permissions, the user will still have full permissions because he is still a member of the unrestricted group. Although Microsoft Access allows you to assign permissions directly to users, this is not recommended. Administering your database can become very difficult if you do.

User and group information, including passwords, is saved in the workgroup file, or System.mda/mdw, which validates user logons at startup. Permissions to individual objects are saved in the database itself. You can give the groups and users within a workgroup various levels of permission to view, modify, create, and delete the objects and data in a database. For example, the users of a particular group might be permitted to read only certain tables in a database and not others, or you could permit a group to use certain forms but not to modify the design of those forms.

Setting a password for the default Admin user account activates the logon dialog box so that users will be prompted for a valid user ID and password each time that they start Microsoft Access. If you never set a password, all users will be logged on as the Admin user (with no password) and you will never see the logon dialog box when starting Microsoft Access. So even though it may appear that there is no security present, it is just transparent until you set a password on the Admin user account.

The database password was introduced in Microsoft Access 95. This is a simple password on the database itself that allows only users who know the password to open the file. You cannot assign permissions to users or groups with this feature. In addition, the database password feature is not considered to be very secure.

Where Is Security Information Stored?

Microsoft Access stores its security information in two different places. User and group information is stored in a workgroup database, named by default **System.mdw**. The location of this database is specified in the Windows registry or with the /wrkgrp command-line switch. The workgroup database stores which users and groups exist, which users belong to which group, logon passwords (encrypted, of course), and the internal security identifier (SID) for each user and group. Workgroup databases are created by the Workgroup Administrator. The **User and Group Accounts** command on the **Security** submenu of the **Tools** menu changes data in the workgroup database. A single workgroup database can be used by many application databases.

On the other hand, all the permissions that users and groups have on objects in a database are stored in the database itself. After a user's identity and group membership have been established by checking the logon strings against data in the workgroup database, all permission checking is done against system tables within the user database. The **User and Group Permissions** command on the **Security** submenu of the **Tools** menu affects data in the open database, not the workgroup database. If you move objects from one database to another, their security settings travel with them.

Slides 9-11

Look at slides

Slide 12 –13

6. How do I implement field-level or row-level security on my tables (RWOP or queries with Run Permissions set to Owner's)?

To understand how the Run permissions work you must understand the concept of ownership in Microsoft Access. The user who creates or imports an object in Microsoft Access becomes the owner of that object and has full permissions to administer that object. For example, if I create a query that draws on certain tables, then I must have rights to the data in those tables or I would not be able to create, run, and save the query. The **RunPermissions** property in the query is by default set to **User's**, which means that the query will run with whatever permissions the logged-on user has on the base tables that make up the query. If I change the **RunPermission** property to **Owner's**, I allow users to run the query as though they were logged on as me, the owner of the query. The query will run with the permissions of the query owner, rather than with the permissions of the logged-on user. Such queries are often referred to as RWOP queries (Run With Owner's Permissions).

The **RunPermissions** property applies only to saved queries. You cannot use it with SQL strings defined in your module code by the simple expedient of typing WITH OWNERACCESS OPTION in the string. The reason for this is that security is defined on saved objects only by setting certain bits on them. Your module code has no way of knowing who you want the owner to be in this situation because there is no saved object to read those bits from. You can use RWOP queries to grant partial access to tables by restricting either the columns or the rows returned by the query. Therefore, if you have a Salary column in your payroll table, you can design a query for your users to run that returns all the columns except Salary. If you wanted your users to see all salaries except managers salaries, you could design a query that restricted the rows returned by using a **WHERE** clause that excluded those with a managerial job classification. The best way to implement this is to remove ALL permissions from the underlying table(s) and use only queries to get at the data you

want your users to have. You then grant the appropriate level of permissions to the groups or users only on the query itself, allowing them only to view data or to be able to modify it through your query.

Note There are other issues to be considered when using RWOP queries with attached tables. See Section **Error! Reference source not found.**, “Error! Reference source not found.”, Section **Error! Reference source not found.**, “Error! Reference source not found.”, and Section 14.1, "Using RefreshLink to relink tables" for more information.

7. What happens when the front-end database permissions on an attached table differ from those in the back-end database?

Microsoft Access uses a combination of security settings in both the current database and remote database when ascertaining rights to attached/linked tables. Permissions set on the back-end tables cannot be over-ridden by granting more liberal permissions on the front-end tables. In this context, it is helpful to think of the attachments in the front-end database as merely the connection strings, or the information needed to locate the actual tables in the back-end database. Permissions set on connection strings should not, and do not, override permissions set on actual tables.

Microsoft Access does not treat the combination of permissions between the base tables and the connection strings by using the "least restrictive" rule, but rather it determines the most restrictive rights. Microsoft Access first determines the least restrictive rights for the table in both the current and remote databases. Then Microsoft Access compares these permissions and combines them so that, unless a permission is available in both databases, it will restrict rights to the attached table.

The following table highlights this.

Front-End DB	+	Back-End DB	=	Permissions on Attached/Linked Table
None	+	Administer	=	No Permissions
Read Only	+	Administer	=	Read Only
Administer	+	None	=	No Permissions
Administer	+	Read Only	=	Read Only

The simplest thing to do is to grant users full permissions on tables in the front-end and restrict the back-end to the barest minimum needed if users need to relink tables. If they don't need to relink tables, then removing ALL permissions and using RWOP queries will give you the highest level of data security possible.

Slide 14 -23

Each user and group has associated with it a security identifier (SID). The SID is a machine-generated, nonreadable binary string that uniquely identifies the user or group. When a user logs on, Microsoft Access looks in the MSysAccounts table of the workgroup database for a user of the same name (case insensitive). If a user with the same name is found, it then validates the password (case sensitive). If the password matches, the SID of the user is retrieved and saved in an internal structure. The password is only used to validate users when they log on. It has no other effect on security.

By default, Microsoft Access first attempts to log on as the user Admin, with a blank password. If this log on fails, the user is presented with the **Log on** dialog box. If a user name and password were specified on the command line (using the /User and /Pwd flags), Microsoft Access first tries to log on using that user name and password. If this log on fails, the user is presented with the **Log on** dialog box. Once logged on,

the user's SID is retrieved. Then the SIDs of all groups to which that user belongs are retrieved and saved in the same internal structure. These SIDs are used for all subsequent security operations within Microsoft Access.

What are the steps to secure a database?

The process to secure a database is the same, no matter which version you use. The only differences are: beginning with Microsoft Access 95, the Security Wizard is built into the product and in Access 2000 the Security Wizard can perform all of the steps for you, including creating a new workgroup information file. You may elect not to use the Security Wizard and to secure the database manually by following these steps.

11. Use the Workgroup Administrator program (Wrkgadm.exe) to create a new workgroup information file. Write down the **Name, Organization, and WorkGroup ID** strings that you will be prompted for when you create your new workgroup information file and store them in a safe place. If your workgroup information file ever becomes lost or corrupted, you can reconstruct it by using these identical strings, which are then encrypted to create a unique token. Without a valid workgroup information file, you could conceivably be locked out of your database forever. Another reason to save this information is for upgrading a secured Access database to a newer version of Access. The recommended path for upgrading databases is to re-create the workgroup file in the new version of Access before upgrading the database itself.
12. The Workgroup Administrator automatically switches you to the new workgroup information file. Start Access, and open any database.
13. You will be logged on as a user named Admin. Use the **Security** menu options to add a password for the Admin user. The Admin user is the default account, and setting its password is what causes Access to prompt for a logon Name and Password the next time that you start Access.
14. Create a new user, which is the account you will use to secure the database. Add this new user to the Admins group. Write down the strings that you use for the name and PID in case you ever need to re-create your workgroup information file. The PID is not the password—the string used for PID is encrypted, along with the string used for the Name, to create a unique token (SID, or system identifier) identifying the user.
15. Quit Microsoft Access and log back on as the new user account that you created in step 4. You will not have a password for this account yet, (the PID you typed with the name in step 4 is **not** the password), so now is a good time to set one.
16. Remove the Admin user from the Admins group so that Admin is a member only of the Users group. The Admin user account has no administrative powers built into it; they are derived from membership in the Admins group, which does. Although you cannot delete any of the built-in users or groups (Admin, Admins, and Users), you can move users to and from the Admins group and restrict permissions to the Users group.
17. Open the database that you want to secure and run the Security Wizard. Select the objects that you want to secure (it makes sense to secure them all). The wizard will then create a new database owned by your new user, and will import all of the objects and relationships into it. It will also remove all permissions from the Admin user and the Users group and encrypt the new database. The original database will not be altered. Note that the Access 2000 security wizard does not create a new database—it simply creates a backup copy of the original. One flaw with this arrangement is that not all permissions to open the database

are removed from the Admin user and Users group to open the database, even though they appear to have been removed.

18. Open the new database. Because the Security Wizard removed all permissions from the Users group for the secured objects, you need to create your own custom groups and assign the level of permissions needed to these groups. Every user is required to be a member of the Users group (otherwise, a user would not be able to start Microsoft Access), so only grant permissions to Users that you want everyone to have. Members of the Admins group have irrevocable power to administer database objects, so make sure to limit membership in the Admins group to only those users who are administrators.
19. Create your own users and assign them to the groups that reflect the level of permissions that you want them to have. Do not assign permissions directly to users because that is extremely hard to administer. Users inherit permissions from the groups they are members of, and keeping track of the permissions assigned to a group is much easier than keeping track of the separate permissions of individuals. If a user is a member of multiple groups, then that user will have all of the permissions granted to any of those groups plus any permissions assigned specifically to the user (this is known as the "least restrictive" rule). There is no way to deny permissions to a user if that user is a member of a group that has been granted those permissions. If you need to create specific permissions for only a single user, create a group for that user and assign the permissions to the group; then, add the user to the group. The reason for this becomes clear when you consider that the user may quit, and you may have to set up permissions for the replacement on short notice.
20. Additionally, you may need manually to remove the **Open/Run** permission from the database container for the Users group through the security menus or through code. This will prevent someone from opening the database by using another workgroup information file or the default System.mda/mdw. In Microsoft Access 97, the User Level Security Wizard is supposed to remove the **Open/Run** database permissions for the Users group, but fails to do so. The Access 2000 Security Wizard removes permissions to the point where they are not visible on the security menus, but testing has revealed that in Access 2000 it is possible to open a database by using the default workgroup information file regardless of the menu settings. The cure for both versions of Access is to create a new, empty database while logged on as a member of the Admins group and import all of the objects from the secured database. You should take this step before spending too much time securing objects because Access considers imported objects to be "new" and loses the permission information that was stored in the source database.

The following table lists the default names and locations of the workgroup file and the Workgroup Administrator program.

Version	Workgroup File Name (default)	Workgroup File Location	Wrkgadm.Exe Location
2.0	System.mda	C:\Access	C:\Access
95	System.mdw	C:\MSOffice\Access	C:\MSOffice\Access
97	System.mdw	C:\Windows\System	C:\Windows\System
2000	System.mdw	\Program Files\Common Files\System	\Program Files\Microsoft Office\Office\1033 Note: 1033 is the default folder

			for the English version of Access
--	--	--	-----------------------------------

Slide 20

8. The Security Wizard

The Microsoft Access Security Wizard is by far the easiest way to secure databases. It allows you to choose which object types to secure -- tables, queries, forms, reports, macros, or modules. The selected object types are secured by revoking all permissions from all accounts except that of the user running the wizard. Beginning with Microsoft Access 95, the Security Wizard is included in the retail product.

To secure a database, log on as a user other than Admin who has read permissions on all objects and data in a database. Then open the database and invoke the Security Wizard by clicking **User-Level Security Wizard** on the **Security** submenu of the **Tools** menu. The Security Wizard creates a new secured copy of the database. It will leave the original copy unmodified. The Security Wizard can be used to secure databases in Microsoft Access 95 format.

Slide 24 -26

Security Manager screens

Slides 27-29

4. "
5. Make sure that all permissions to modules are revoked for the Users group and the Admin user. (If you have used the Security Wizard, this is already taken care of.)
6. Grant full permissions to the Admin user and the Users group for all the objects that you want everyone to be able to use.
7. Distribute your application using the default workgroup information file. Because there is no password assigned to the Admin user in the default System.mda/mdw, everyone logs on as Admin and everyone has only those permissions you have assigned the Admin user and the Users group.

When you need to make modifications to your application, you need to switch to your development workgroup database and log on as the owner of the database.

Beginning with Microsoft Access 97, a database can be converted to an .mde file. This removes all editable code from the database and makes the design of forms, reports, and modules inaccessible to users. It has no effect whatsoever on data or on linked tables. The account creating the .mde file must own or have Administer permission to the source .mdb file.

How do I work with a secured application and an unsecured application at the same time?

Create separate shortcuts on your desktop for your secured application and the default, unsecured installation of Microsoft Access. You can only log on to one workgroup information file (System.mda/mdw) at a time, so your users must understand that they have to quit and restart Microsoft Access when they want to switch back and forth between secured and unsecured applications. A second option is to train your users to use the Workgroup Administrator, but this is usually confusing to end-users. Creating separate shortcuts is not that difficult, and you

only have to do it once. The techniques are different for each version of Microsoft Access.

Microsoft Access 95, Microsoft Access 97, and Microsoft Access 2000: You don't need .INI files any more; you can use the `/WRKGRP` switch directly on the command line of the icon:

```
c:\msoffice\access\msaccess.exe /wrkgrp c:\myapp\secacc.mdw
```

How do I keep users from viewing Code Behind Forms?

Microsoft Access 97 and Microsoft Access 2000: You can compile your database as an MDE file. This removes all of the code so that it cannot be viewed. In addition, users will be unable to create or modify any forms, reports, and module code. Compiling as an MDE has no effect on the data.

Extra info below

8.1 Using Owner-Permission Queries to Prevent Users From Viewing Table Designs

To display a table or query, Microsoft Access needs to know field names and other field properties such as **Format**, **InputMask**, and so on. Thus, for a user to read the data in a table or query and display it in the UI, she must also have permissions to read the design of the table or query. That's why checking the **Read Data** permission check box in the **User and Group Permissions** dialog box automatically checks the **Read Design** check box as well. If you don't want your users to see your table or query definitions, you can use owner-permission queries to restrict their access to this information. Here are the steps:

1. Remove *all* permissions on the sensitive tables or queries from the users or groups that you want to restrict. For this example we'll use Secret Table, but it can be a query as well.
2. Build a new query that includes all the fields from your sensitive table or query. It can take the form `SELECT * FROM SECRET TABLE`. For this example, we'll call this query My Owner Query.
3. Make sure you or a secure group owns these queries. In multideveloper environments, use the **Change Owner** tab to change the owner of My Owner Query to a developers' group. This will allow any developer to modify and save My Owner Query.
4. Set the **RunPermissions** property of My Owner Query to **Owner's**.
5. Grant the users and groups that you want to be able to update data but not view the table's design appropriate data permissions on My Owner Query. This usually means Read Design, Read Data, Update Data, Delete Data, and Insert Data.
6. Base your forms and reports on My Owner Query.

Your users will be able to update data in Secret Table through forms based on My Owner Query. However, they won't be able to view the design of the table. If they try to view the design of Secret Table, they will receive the message "You don't have permissions to view Secret Table."

Encryption

Encryption is separate from Microsoft Access security. It is used to prevent someone using a file or disk editor (such as Norton Utilities®) from reading or writing data directly to an **.mdb** file, bypassing the Microsoft Jet database engine. Encryption by itself is meaningless without implementing security, because

by default anyone can open an unsecured database because they are transparently logging on as the Admin user and thus have access to the data.

Microsoft Jet reads and writes all data one page at a time. Pages are 2K in size. Encryption is done at the page level, not at the data level. That is, at the level that encryption is being done, there is no notion of what is on the page, only that there's 2K of data that needs to be encrypted and written, or read and decrypted. This implies that everything in a Microsoft Access **.mdb** file is encrypted, including tables, queries, forms, indexes, and so on. Microsoft Access uses RSA Data Security Incorporated's RC4 algorithm for database encryption.

If a database isn't encrypted, it would be possible for a (very) knowledgeable person to use a disk editor to read the SID of the database owner or the SID of the Admins group *of the workgroup database in use when the .mdb was created*. Given that SID, this person could assume ownership of the database using the methods outlined above and do anything he wanted with the database. For this reason, it is recommended that if you want to protect against such intrusions, you should encrypt your database. The Security Wizard encrypts databases as part of the process of securing them.

Only the creator (owner) of a database or a member of the Admins group *of the workgroup database in use when the database was created* can encrypt or decrypt it.

Due to the overhead of encrypting and decrypting, there is a performance degradation of approximately 10–15% in encrypted databases. Encrypted files are also essentially incompressible (using utilities such as PKZIP®, Stacker®, or DriveSpace™).

*****8

Extra

How can I tell who is logged on to my shared, networked application?

Microsoft Access has a function, **CurrentUser()**, that returns the name of the logged-on user of the current database, but that won't help you if you're trying to determine all of the users of a shared database centrally located on a network. Unfortunately, there is no built-in way to do this. In order to implement this functionality, you would have to include your own special table that adds a user's name whenever someone starts your application, and then deletes it when that person logs off.

The problem to this method is that if a user does not exit gracefully (for instance, if that user's computer has a fatal error (that is a GPF or IPF) or if that user turns the computer off without first quitting Microsoft Access) then that user will still be listed as being "logged on" even though that user is not. To guard against this, you should delete all names from the table at a time when you are certain no one is in the database (such as when you are doing a backup/repair/compact). You may also want to provide a way for your administrative users to browse and adjust the table to remove such a phantom logon.

There is a DLL available from Microsoft that will let you see which people are using a specific database (it will list all the names their computers used on the network), but although it can check Jet database engine 2.0/2.5/3.5/4.0 applications, it cannot be run from a 16-bit application such as Microsoft Access 2.0. Check the Microsoft Web site for the most recent version, which should be compatible with Jet database engine 2.5 through 4.0. The JETLOCK.EXE file contains the "Understanding Jet Locking" white paper, as well as the LDBView utility and the MSLDBUsr.dll.

In Access 2000/Jet 4.0, the capabilities of MSLDBUsr.dll have been merged with Jet and can be accessed via the Jet 4.0 OLE DB provider. To use this "user roster," simply run the following code:

```

Const dbSecUserRosterGuid = "{947bb102-5d43-11d1-bdbf-
00c04fb92675}"
Dim rs As ADODB.Recordset
Dim con As ADODB.Connection

Set con = CurrentProject.Connection
Set rs = con.OpenSchema(adSchemaProviderSpecific, ,
dbSecUserRosterGuid)

```

This GUID is a value defined in a Jet OLE DB header file and is simply a provider-specific schema view. It will return a recordset with the following information.

Column Name	Column Type	Meaning
COMPUTER_NAME	Text	The name of the computer
LOGIN_NAME	Text	The Jet user name, matching the value that would be returned by the CurrentUser function (not the operating system user name!)
CONNECTED	Boolean	Whether or not the user is currently connected
SUSPECT_STATE	Boolean	Whether or not the user has left the database in a suspect state (note: this column will never have a False value; it will always be either True or Null).

This information is the same as that provided by MSLDBUsr.dll, with the addition of the Jet user name. Obviously, this will only be useful in a secured workgroup that has different users defined.

In tandem with the User Roster is another new feature in Jet 4.0 known as passive connection control. This feature gives you the ability to put the database in a state where, although users who are already in the database can continue their work, no new users can be admitted. If anyone tries to open the database, they will receive errors as if the database were opened exclusively. Passive connection control can only be used from the Jet OLE DB provider. To use passive connection control, run code such as the following:

```

Dim con As ADODB.Connection
Set con = CurrentProject.Connection
con.Properties("Jet OLEDB:Connection Control") = 1

```

Once you have set this property, you can use the User Roster's list to help determine who is still connected and then have them disconnect, so that operations such as Compact and Backup can be performed.

Note There is no method to actively disconnect users from a database. You **must** have Open Exclusive permission on the database in order to change the Connection Control setting of the database.

How can I prevent users from creating new objects in my database?

The following function will remove permissions to create new tables or queries in the current database from the specified user. The argument, *strUser*, can be either a User or a Group. Remember that you will have to run the **faq_NoNew()** function not only for the specified user, but also for the Users group. This is because the user

has both explicit permission to create new tables or queries, and implicit permission derived from membership in the Users group.

There is no way to remove permission to create forms, reports, macros, or modules in Microsoft Access unless you compile your database as an MDE in Access 97 or Access 2000. You need to have Administer permissions in order to run this code (shown in Microsoft Access 2.0 format).

```
Function faq_NoNew (strUser as String)

    Dim db As Database
    Dim con As Container

    Set db = DBEngine(0)(0)
    Set con = db.Containers("Tables")
    con.UserName = strUser
    con.Permissions = con.Permissions And Not DB_SEC_CREATE
End Function
```

If you want to set the permission to create new tables and queries back again, use:

```
Function faq_OKNew (strUser as String)

    Dim db As Database
    Dim con As Container

    Set db = DBEngine(0)(0)
    Set con = db.Containers("Tables")
    con.UserName = strUser
    con.Permissions = con.Permissions Or DB_SEC_CREATE
End Function
```

Microsoft Access 95, Microsoft Access 97, and Access 2000: The intrinsic constants are the difference here. The correct spelling can be found in the Object Browser or listed in the table in Section **Error! Reference source not found.**, **"Error! Reference source not found."**.

```
...
con.Permissions = con.Permissions And Not dbSecCreate
...
```

How do I prevent users from holding down the SHIFT key to bypass the AutoExec macro?

Microsoft Access 95, Microsoft Access 97, and Microsoft Access 2000: You can set the **AllowBypassKey** property to **False** to prevent a user from bypassing the startup properties and the AutoExec macro by holding down the SHIFT key. This option is only available through code.

This function only needs to be run once for each database to set the **AllowByPassKey** property. The documented syntax for creating and setting properties in the Access Help file is CreateProperty(PropertyName, PropertyType, PropertyValue, DDL) syntax. However, any knowledgeable user can run code to reverse the **AllowByPassKey** setting. In order to prevent users from reversing the property setting, you must set the fourth (DDL) parameter to True: CreateProperty(PropertyName, PropertyType, PropertyValue, True). The following function can be run from another database or from the Debug/Immediate window.

```

Function faq_DisableShiftKeyBypass(strDBName as String, fAllow as
    Boolean) As Boolean

    On Error GoTo errDisableShift

    Dim ws As Workspace
    Dim db As DATABASE
    Dim prop As Property
    Const conPropNotFound = 3270

    Set ws = DBEngine.Workspaces(0)
    Set db = ws.OpenDatabase(strDBName)

    db.Properties("AllowByPassKey") = Not fAllow
    faq_DisableShiftKeyBypass = fAllow
exitDisableShift:
Exit Function

errDisableShift:
    'The AllowBypassKey property is a user-defined
    ' property of the database that must be created
    ' before it can be set. This error code will execute
    ' the first time this function is run in a database.

    If Err = conPropNotFound Then
        ' You must set the fourth DDL parameter to True
        ' to ensure that only administrators
        ' can modify it later. If it was created wrongly, then
        ' delete it and re-create it correctly.
        Set prop = db.CreateProperty("AllowByPassKey", _
            dbBoolean, False, True)
            db.Properties.Append prop
            Resume
    Else
        MsgBox "Function DisableShiftKeyBypass did not complete
            successfully."
        Faq_DisableShiftKeyBypass = False
        GoTo exitDisableShift
    End If
End Function

```