

Starting the Interface with Access Data Projects

Martin Reid



Martin Reid takes you into the world of Access Data Projects and SQL Server. Using a simple application, he shows you how to leverage your Access skills to create views for your application. He then tackles the issues around ADP security that you'll need to know to make your views useful.

In this article, I'll be building a user interface in an Access Data Project (ADP) showing how to create and use ADP views effectively. However, there's more to using views than just creating them, so I'll also be providing some background information on SQL Server Security and a form's ServerFilter property.

Views

Views within SQL Server can prove very useful. Using a view, you can protect your tables by granting users permission to access data only via views as opposed to letting them work with the base tables. If you've worked with Select queries in Access, you already know a great deal about views in SQL Server. However, there are a few limitations with views in SQL Server that make them different from queries. Views can't:

- Contain an ORDER BY clause (though see the sidebar "Using ORDER BY" for a workaround).
- Be parameterized.
- Be used with temporary tables.

Using ORDER BY

If you need to use an ORDER BY in a view, simply use the TOP Percent syntax. Once you've included that Microsoft-specific extension in your statement, you can use an ORDER BY clause. For example, this view will work:

```
SELECT TOP 100 PERCENT StaffID, StaffForeName,
StaffSurname
FROM    dbo.Staff
ORDER BY StaffID
```

In order to retrieve all of the rows that meet your criteria, make sure that you request the top 100 percent.

Fortunately, there are also several compensating advantages to using views. With views you can:

- Hide complex SQL statements from users.
- Protect base tables.
- Improve network performance by restricting the number of records transferred to the client.
- Hide confidential data by not including it in the view definition.

Creating a view is a fairly simple process using the graphical tools in Access 2002. In fact, it's as easy as creating a standard query. For example, in my sample project-management database I have a table of staff members who can be assigned to tasks. Here's how I created a simple view of the Staff assigned to Tasks:

1. From the database window, Select Query | New.
2. Select Design View from the Query dialog. This opens the Query builder.
3. Select the Staff, Staff_To_Task, and Tasks tables to populate the Query builder.
4. Select the fields that you want to display.

Figure 1 (on page 13) shows the query builder at this point. Unlike a query in Access/Jet, you must save a view before you can use it.

In the background, the Query builder is creating the statement required to create the view on SQL Server. Within Access, you can see the SQL statement that's generated to define the view. The full syntax (from SQL Server) looks like this:

```
CREATE VIEW dbo.uvw_StaffTask
AS
SELECT dbo.Staff.StaffForeName,
       dbo.Staff.StaffSurname, dbo.Tasks.TaskTitle
FROM   dbo.Staff
       INNER JOIN dbo.Staff_To_Task
         ON dbo.Staff.StaffID = dbo.Staff_To_Task.StaffID
       INNER JOIN dbo.Tasks
         ON dbo.Staff_To_Task.TaskID = dbo.Tasks.TaskID
```

Nothing hard here, though the use of the table owner prefix (dbo) created by Access might be new to you. By specifying the table owner, Access ensures that you make an unambiguous call to the proper tables (and not someone else's with the same name).

For use later, create a simple view based on the Staff table and then generate an autoform based on this view (Figure 2 shows the form that you should get). There's one more thing to watch out for with views. But before I get to that, I'm going to stop and look at SQL Server security.

ADPs and security

So far, I've been working with the database as the "dbo user" or system admin. Just like Access/Jet, when you enter your project, by default you're logged in as the Admin user. However, you wouldn't permit most users to have this level of access to the database. Just like Access/Jet (but using the built-in security features of SQL Server), you can add users to the application beyond the default Admin user.

Security on SQL Server works on two levels. The first level is access to the server. The second level is access to a specific database object and its objects. In Access/Jet, you can create groups and then assign users to each group. In SQL Server, you create database roles and assign users to a role—not really that much different from what you do in Access with groups. In addition to whatever role you assign a user to, users are, by default, members of the Public role.

Up to this point in these articles, I've been running SQL Server in Windows security mode. For this example, I'll change my security setting to SQL Server and Windows security to demonstrate the full range of security settings. To change the SQL Server model, you must use SQL Server's Enterprise Manager. If you're using the version of SQL Server that ships with Access, you won't have a copy of Enterprise Manager. (See the sidebar "Getting Enterprise Manager.") Start Enterprise Manager and drill down in the tree in the left-hand pane until you see the name of your server computer. Right-click on the computer name and select Properties from the pop-up menu. If you then click on the Security tab, you'll be able to change your SQL Server security settings (see Figure 3, on page 14). You must also set the

Getting Enterprise Manager

You can download the evaluation of SQL Server 2000 Enterprise Edition from www.microsoft.com/sql. This is a time-limited version. However, Enterprise Manager doesn't appear to expire with the rest of the server and can be used to manage MSDE (SQL Server Desktop) databases. How this works in regard to licenses isn't clear.

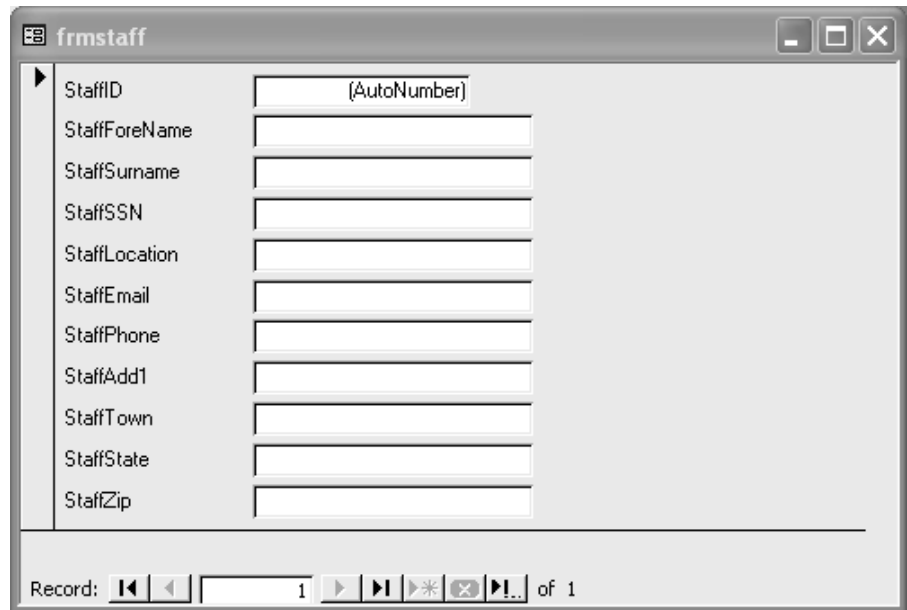


Figure 1. Creating a view.

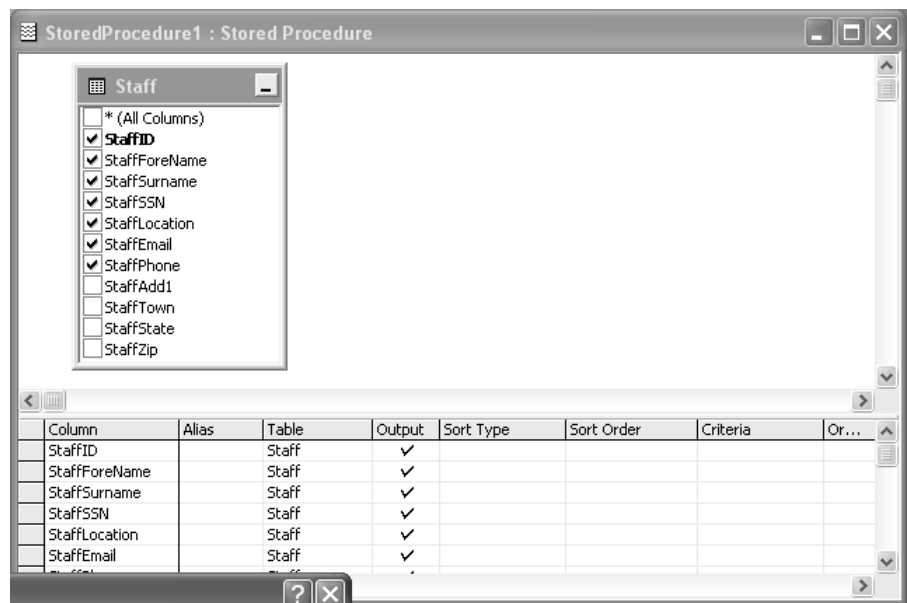


Figure 2. Staff autoform.

connection property of the ADP file from the File | Connection menu choice in Access (see Figure 4, on page 15).

If you want to continue using Windows security, the instructions that I'll provide here on creating roles will still apply. Simply follow the instructions that I provide for creating a new role, and then assign the Windows users (defined in your Windows security system) to the role. You won't need to create the login that I'll also describe.

Roles and users

There are two types of roles on the server: Fixed-Server

roles and Fixed-Database roles. As indicated by the name, one set of roles applies to the server, and the other set applies to individual databases. Table 1 shows the main server roles available.

There's also a set of Fixed-Database roles. Unlike the Fixed-Server roles, the Fixed-Database roles are specific to each database. I've listed them in Table 2.

The real power of roles lies in the fact that you can create your own, as I'll show you now. For this example, you'll need Enterprise Manager.

Creating a role

Using the Enterprise Manager, I'm going to create a new

Table 1. Fixed-Server roles.

Role	Description
Sysadmin	Can perform any role on the server.
Serveradmin	Has the ability to configure the server.
Setupadmin	Can work with and manage linked servers.
Security Admin	Can manage server security.
Unique Table	When working with multiple tables, views, and stored procedures, users can select one object as unique (that is, updateable). Access Help states that this property must be set, or the recordset will be read-only. This is not the case, as I'll show.
Input parameters	One of the more interesting properties. Can be used to pass values from forms to stored procedures.

Table 2. Fixed-Database roles.

Role	Description
db_owner	Can carry out all owner actions on all databases. Combines the privileges of all database roles.
db_accessadmin	Manage groups, users, and logins within a specific database.
db_datareader	Read data from all database tables.
db_datawriter	Insert, update, and delete from any table within the database.
db_backupoperator	Can back up the database.
db_ddladmin	Change objects using DDL statements.
db_securityadmin	Work with database permissions.

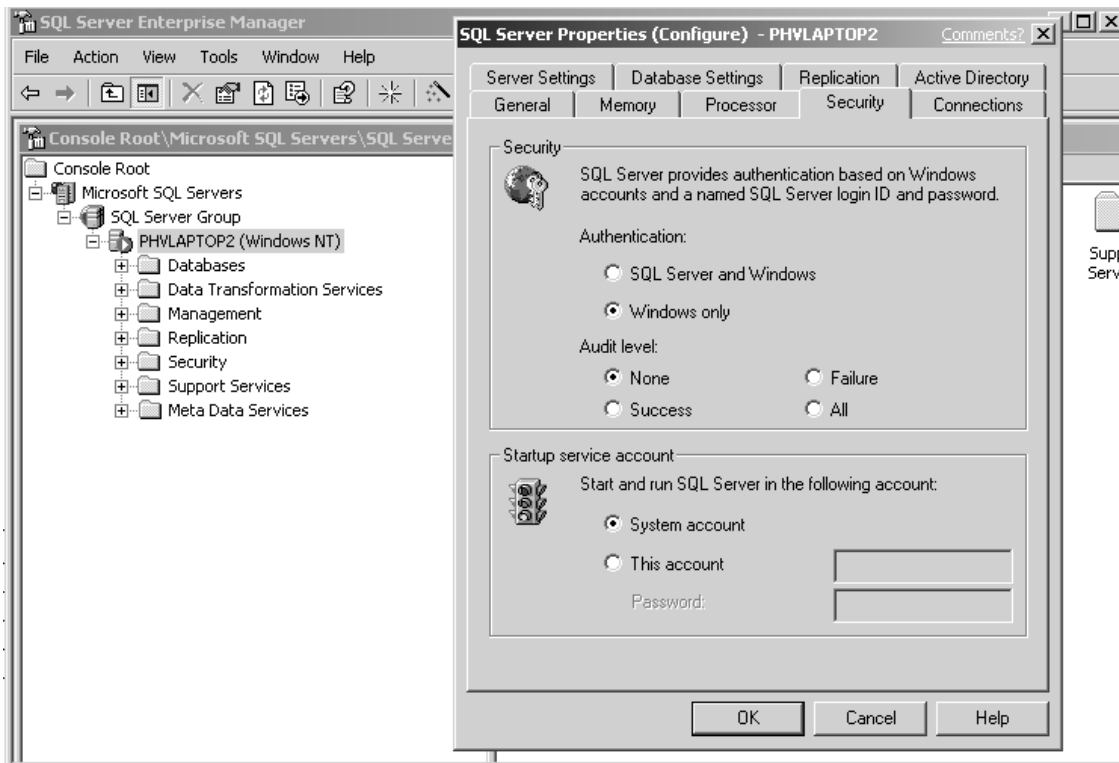


Figure 3. Changing security settings in SQL Server Enterprise Manager.

role called StaffUsers and assign a single user to the role, AineReid. This user will be given full permissions on the view based on the Staff table that I created earlier (this is where you'll see the problem with views that I mentioned before). According to Microsoft, I can give permissions to users to access data via a view instead of giving permission to access the base table. While this is the case, as you'll see there are one or two problems to bear in mind.

To create a role, follow these steps:

1. Open Enterprise Manger.
2. Expand the ProjectSQL database.
3. Select Roles.
4. Right-click and select New Role from the shortcut menu.
5. Enter the role name into the dialog.
6. Click OK.

That's all there is to it. As you can see, at this stage, a role is really just a name. Having created that name, you can now assign users to it. While I created the role first in this example, you can create the users first and then add them to a role.

Once you've created a role, you can the assign permissions to it. To do that, follow these steps in Enterprise Manager:

1. Right-click the newly created StaffUsers role.
2. Select Properties.
3. In the dialog that appears, click on the Permissions button to open the Permissions dialog. This dialog allows you to assign specific permissions for the role on any of the database objects.
4. Assign the permissions by checking the box in the Insert, Update, and Delete columns for the staff table.

In effect, you have granted the StaffUsers role permission to Insert, Update, and Delete data within the Staff table via the view.

Creating the server login

In SQL Server, logins are created independently of roles, so I now need to create a login. Once I've created a login (a user id and a password), I can assign my role to it. Effectively, anyone who logs on with this user id and password will be assigned to my role. Here are the steps for that, again using the Enterprise Manager:

1. Expand the server security tree.
2. Right-click on Login.
3. Select New Login to create the server login.
4. In the dialog box that appears, enter a name for the login (in this case, AineReid). In this dialog, you can choose to use both Windows and SQL Server Authentication.
5. Because I'm not on a Windows Domain, I select SQL Server Security and enter a password for the login.
6. Click on the Database Access tab, and check the ProjectSQL database to display a list of roles for this database.
7. Select the StaffUsers role and click OK.
8. When prompted, re-enter the password.

That's it. You've created a server login and added the role created earlier to that login.

If you're working along with me, before continuing, make sure that you've done the following:

- Changed the security model of SQL Server to Windows and SQL Server.
- Changed the connection in the ADP to use Windows and SQL Server Security.
- Created the server login.
- Created the Staff role.

View problems

If you now log in to the ADP using the newly created user AineReid, you should notice several changes to the database:

- Only the Staff table is available.
- Only the view created earlier is available.
- All forms are available in the database window but not all functions.

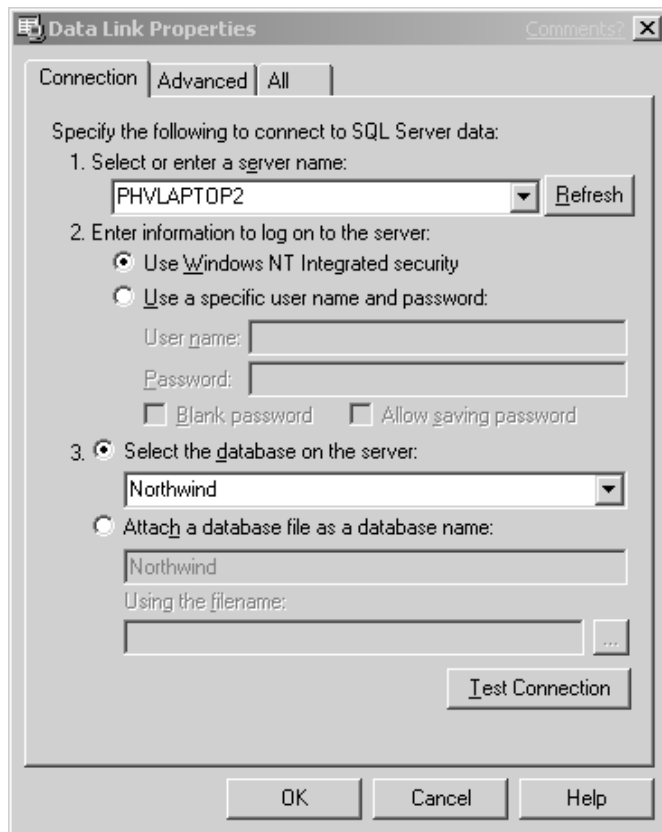


Figure 4. Changing connection settings in Access.

As you remember, I've only granted the AineReid role access to the Staff view that I created. SQL Server will disallow access to all other objects. If you open the autofrom created earlier (or go back and look at Figure 2), you can see that it's not possible to add a new record with the form. In the background, ADO and OLEDB are using the base table, not the view. Since our user doesn't have permission for the table, he can't update it. In order to enable adding new records, you must change the properties of the view.

To change your view settings, you'll have to either reset your connection to Windows or log in as a system admin user because AineReid doesn't have sufficient permissions.

System admin

This raises an important point. When working with an ADP, it's essential that you create a system admin user with full access to all database and server objects. The best advice that I can give you, from a security point of view, is to create a new user with a new login password and assign that user to the System Administrators role—don't use the built-in system admin user.

After you've created a new user and assigned system admin permissions to that user, ensure that you properly password-protect the system admin login (this is particularly true when you make your database available

on the Web). SQL Server still permits you to access the server with "system admin" and a blank password. Do not do this. *Always* assign a password to the "system admin" login.

View properties

To change the properties of the Staff view, in the database window, I open the view in design view and select View Properties. Figure 5 shows the property sheet for the view.

In the property sheet, check "Update using view rules" to enable updating via the view. It's also useful to check the Bind to Schema box. With this option checked, changes to the base table that would invalidate the view are forbidden. As I said earlier, every change to the view just changes the underlying SQL statement that's sent to SQL Server. Here's the view's SQL after these options are checked:

```
CREATE VIEW dbo.vw_Staff_Form
WITH SCHEMABINDING,VIEW_METADATA
AS
SELECT
StaffID, StaffForeName, StaffSurname,
StaffSSN,StaffLocation,StaffEmail, StaffPhone,
StaffAdd1,StaffTown,StaffState,StaffZip
FROM dbo.Staff
```

The magic of those options is performed by the VIEW_METADATA statement. This statement passes information about the view back to the client instead of passing information about the base tables on which the view is based.

Server filter

I said in a previous article that one of the things you must consider with SQL Server is how you can reduce the number of records that are transported from the server to the client. There are several ways to do this. In my next article, for instance, I'll look at using stored procedures to restrict the records. Here, I'll show you how the server filters provided by Access 2002 can allow you to control the records returned by a view.

By using server filters, I have the filtering process take place on the server as opposed to filtering the records on the client. This reduces the records sent to the client, improving network performance at the cost of doing some extra work on the server.

There are two filters that you can

Continues on page 22

objtype	objname	name	value
COLUMN	ProjectDesc	MS_Description	Description of the project
COLUMN	ProjectDesc	MS_DisplayControl	109
COLUMN	ProjectDesc	MS_Format	
COLUMN	ProjectDesc	MS_IMEMode	0
COLUMN	ProjectEnd	MS_DisplayControl	
COLUMN	ProjectEnd	MS_Format	
COLUMN	ProjectEnd	MS_IMEMode	0
COLUMN	ProjectID	MS_DisplayControl	109
COLUMN	ProjectID	MS_Format	
COLUMN	ProjectID	MS_IMEMode	0
COLUMN	ProjectManager	MS_DisplayControl	109
COLUMN	ProjectManager	MS_Format	
COLUMN	ProjectManager	MS_IMEMode	0
COLUMN	ProjectStart	MS_DisplayControl	
COLUMN	ProjectStart	MS_Format	
COLUMN	ProjectStart	MS_IMEMode	0
COLUMN	ProjectTitle	MS_Caption	Project Title
COLUMN	ProjectTitle	MS_Description	Title of the current Project
COLUMN	ProjectTitle	MS_DisplayControl	109
COLUMN	ProjectTitle	MS_Format	
COLUMN	ProjectTitle	MS_IMEMode	0

Figure 5. View property sheet.

Starting the Interface...

Continued from page 16

set on the server side: Server Filter and Server Filter By Form. However, if you're using a stored procedure in the record source for a form, the Server Filter By Form isn't available. To set the Server Filter property, simply put a WHERE clause in the property (without the word "WHERE"). For example, to restrict my auto Staff form to those staff members whose forenames begin with A, I'd set the Server Filter property to this:

```
StaffForeName Like 'A%'
```

The property can also be set via code. For example, by adding the following code to a command button, I can

change the server filter currently being used:

```
Me.ServerFilter = "StaffSurname Like 'R%'"  
Me.Refresh
```

In my next article, I'll be looking at building several forms for the project and looking at stored procedures as a means to filter records returned to the client. I'll also look at how common Access form objects behave in the world of SQL Server. ▲