

Chapter 1 Getting Started

Merge Replication

The presence of a mobile data store, coupled with an enterprise database that it can synchronize with, stand as two of the most important elements in creating an offline, mobile line of business application. Going all the way back to SQL Server 7, Microsoft's flagship enterprise database has come equipped with a mature, sophisticated data synchronization technology called Merge Replication. Using a hub and spoke architecture, this technology is used to replicate data from a Publication database out to Subscriber databases that make changes and then merge their inserts, updates and deletes back into their Publication database. Changes can also be made to the Publication database directly whether the Subscriber databases are connected or not. SQL Server captures all those incremental data changes and reconciles conflicts according to rules you configure or via a custom resolver you create. This technology has been a perfect fit to push corporate office data out to branch offices where autonomous changes can be made on both sides, reconciled and merged back together over wide-area private circuits or VPN tunnels.

With the introduction of SQL Server CE 1.0 back in 2001, Merge Replication went from being a server-to-server technology, to one that embraced Windows Mobile devices as well. Utilizing Internet standards, the SQL Server team created a firewall-friendly synchronization transport protocol utilizing HTTP, SSL and Internet Information Services (IIS) to not only reach Windows Mobile devices on the wireless LAN, but also those intermittently online and offline devices found roaming wireless WANs. With SQL Server CE supporting Merge Replication right out of the box, Windows Mobile instantly became the mobile line of business platform of choice when combined with development tools like Embedded Visual Basic, Embedded C++ and the .NET Compact Framework. Since 2001, SQL Server has revved from 2000 to 2005 and SQL Server CE has matured to SQL Server Compact (SSC) 3.1. With each version update, the data synchronization technology has become faster, more scalable, more efficient over slow networks

and better able to recover from loss of connectivity. Today, Merge Replication stands as the most advanced data synchronization technology in a crowded field of competitors. No other sync solution offers the breadth and depth of functionality, manageability and performance.

Are there alternatives to this kind of replication in order to keep data synchronized with SSC? Sure, but there's no silver bullet:

- Using the ADO.NET SQL Server driver in the .NET Compact Framework allows Windows Mobile devices to talk directly to the database. This solution would be best suited to corporate WLAN environments that can guarantee complete Wi-Fi coverage campus-wide since there's no automatic recovery from network dropouts. It's not a WWAN solution because the device must talk directly to SQL Server via port 1433. Also, it doesn't support the resolution of conflicts like Merge Replication.
- Remote Data Access (RDA) is the younger brother of Merge Replication. It utilizes a brute-force approach to getting data from SQL Server to the SSC on the device. You have to write code to pull down each table but you don't get referential integrity or the ability to use Identity columns for primary keys. Additionally, it doesn't support the downloading of database change deltas and therefore you must drop and re-download each table to refresh your local database.
- Web Services could be used with custom XML schemas or serialized DataSets and gives you an SOA solution to this problem. This gives you complete control over your server API and is the only way to communicate with other enterprise databases like Oracle or DB2. The problem is that you'll have to write thousands of lines of code to reproduce the built-in functionality of Merge Replication. Additionally, web services don't do a good job of moving large amounts of data over the air and don't offer any recovery from network dropouts.

Scenarios

While it is important to know that Merge Replication can synchronize your enterprise data between SQL Server in the data center and SSC on Windows Mobile, it might be helpful to illustrate scenarios where this kind of technology can add value to your organization:

- **Sales Force Automation**

A Salesperson has leads that are filtered for her particular region replicated out to her device so she knows who to contact in her area. Upon converting a lead to a sale, the Salesperson can have the new client fill out required data on her Windows Mobile device and then immediately replicate that data back to her organization's headquarters.

- **Healthcare**

Upon having a new patient assigned to a doctor at a hospital, the doctor can securely replicate the patient's records down to his Windows Mobile device for review. At the point of diagnosis, the doctor can enter the symptoms, diagnosis, course of treatment and any prescriptions into his Windows Mobile device for immediate synchronization back to the primary patient records database. At this point, the pharmacist would replicate the new prescription for the patient down to her Windows Mobile device and have it filled.

- **Supply Chain Management**

Upon receiving a phone call to place a new order for hardware, the customer service representative enters the order into her Windows Mobile device and then synchronizes the information with the central database. A warehouse "picker" synchronizes his Windows Mobile device and finds out that he needs to pick some hardware products from the various bins located in the warehouse. Upon completion, he synchronizes with the central database thus alerting the forklift driver that there are staged items

waiting to be loaded in a delivery truck. He loads the ordered hardware on the truck, updates his Windows Mobile device and replicates. Early the next morning, the route delivery driver arrives at the distribution center and syncs his Windows Mobile device. His device tells him what items have been loaded on his truck and which customers he will deliver to. Upon receiving a proof of delivery signature on his device, he will replicate the completed order information back to the central database so that accounts receivable can invoice the customer.

- **Retail**

A sales associate at a clothing store notices that the line at the checkout counter has grown extremely long and customers are becoming impatient. She gets her Windows Mobile device with attached credit card swiper and transforms herself into a portable point of sale checkout location to relieve the backed up lines. As she checks out customers, her device replicates the changes in clothing inventory to the inventory management database while simultaneously synchronizing her sales receipts with her company's accounting database.

- **Maintenance**

A plant worker whose job it is to walk around an oil refinery reading gauges, observing the health of equipment and logging that information on his clipboard can now enter that data into his Windows Mobile device. Via Merge Replication, he can instantly notify his superiors or the spare parts department if he finds faulty equipment or gauges that aren't properly calibrated.

- **Emergency Management**

First responders arriving on the scene of a disaster can use their Windows Mobile devices to take pictures and enter data about what they find and then replicate that data back to their agency's headquarters. Furthermore, to prevent the creation of data "silos," SQL Server can replicate with other

agency databases to ensure the information is shared with everyone who needs it. That same data can be replicated back out to first responders on the scene from other government agencies to give them insights into what their inter-agency counterparts have already discovered.

- **Restaurants**

Waiters and waitresses normally take your dinner order by writing it down on paper or memorizing it. Once they walk back to the kitchen, they hang a paper slip in front of the chefs to cook what's been ordered. As everyone knows, this leaves open a reasonable chance for error for each and every order due to errors in writing down the correct order, recalling it from memory, or having it misinterpreted by the chef due to illegible handwriting. Empowering the wait staff with Windows Mobile devices would allow them to take orders by making selection from combo boxes rather than handwriting. Orders would instantly stream in from the customer's table to the kitchen via Merge Replication where it would be accurately displayed for the chefs.

- **Device Management**

Lack of device management is the number one pain point for organizations choosing to roll out mobile devices. When an enterprise needs to push out a thousand copies of a mobile line of business application, ActiveSync and a cradle are not a good option. Mobile application binaries can be converted to byte arrays, inserted into SQL Server image columns, and then replicated out to devices over-the-air where the byte arrays will be rehydrated back into executables and libraries. Devices can report their current health status via Merge by inserting information like current battery life, remaining memory, IP address, the list of installed applications along with other items into a local SSC database and then synchronizing.

Basically, any scenario where data needs to be received, captured or shared is a good candidate for using Windows Mobile devices, SSC and Merge Replication with SQL Server. Instead of waiting for the field worker to return to her office to

enter a day's worth of collected data in "batch" mode, she can make her organization more agile by replicating that data back to her office as it's captured in near real-time.

Running a Magazine

The magazine metaphor might sound like a strange way to describe Merge Replication but you'll soon see that it's right on the money. Below I'm going to run you through the discrete pieces that make up the Merge puzzle so you can better visualize how it all works:

- **Publisher**

The Publisher is the SQL Server that contains the parent database you want to make available to your mobile workforce.

- **Publication**

The Publication is the actual database you are making available. You can choose to make some or this entire database available. Due to this granularity, a Publication is considered a collection of one or more Articles from the parent database that runs on the Publisher.

- **Article**

An Article is a database object that is included in a Publication. These objects can only be tables when synchronizing with SSC since it doesn't support user defined views or stored procedures and triggers. So for our purposes, Articles are the Tables you want to replicate down to the mobile database running on Windows Mobile.

- **Distributor**

The Distributor is a SQL Server that contains the Distribution database. This database stores metadata, snapshots, and status data and is responsible for synchronizing data between the Publisher and Subscribers. The Distributor often resides on the Publisher, but since this book is all about scalability

and performance, the Distributor will run on its own server. Think of the Distributor as essential plumbing.

- **Subscriber**

The Subscriber is the SSC database running on Windows Mobile that is interested in receiving data from the Publisher. The SSC Client Agent resides at the Subscriber and handles all interaction between the local database engine and the SSC Server Agent running on IIS. The local database engine keeps track of all the inserts, updates and deletes that are made on the device so it can send those changes to SQL Server during the next sync.

- **Subscription**

A Subscription is the Subscriber's request for a copy of the Publication. That copy consists of the Articles or Tables that are downloaded and created in SSC. You also get the table indexes, constraints, referential integrity and of course, the data. The initial copy is downloaded from the Distributor and is called the Snapshot. Large databases can be downloaded to Windows Mobile devices with very little free RAM due to the fact that the data is parceled out to the device in very small blocks.

- **IIS**

The one important element that doesn't fit nicely into the magazine metaphor is how IIS and the SSC Server Agent work together to provide HTTP(S) access to all the Subscribers. The SSC Server Agent, which is an ISAPI DLL, the SSC Replication Provider and the SQL Server Reconciler are collectively known as the SSC Server Tools and reside on IIS to provide a secure Internet/Intranet gateway between your devices and SQL Server.

In the logical architecture diagram shown in Figure 1 | 1, you can see how the Publisher, Distributor, IIS and Subscriber all fit together. You'll also notice a number of other moving parts on each tier of this architecture including several Agents designed to do all the heavy lifting. I guess this doesn't look like any magazine you've ever heard of.

4-Tier Merge Replication Logical Architecture

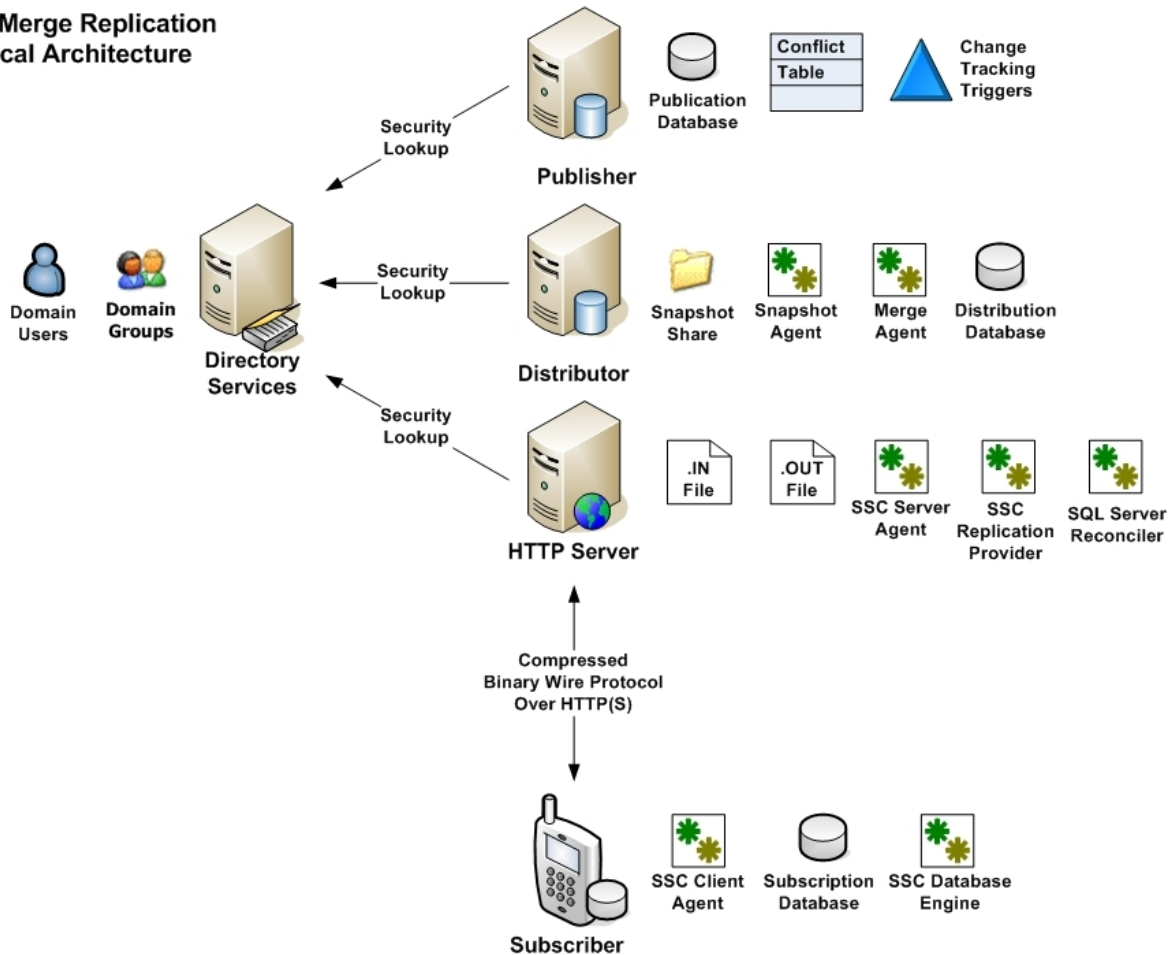


Figure 1 | 1. Logical Data Synchronization Architecture

If I translate Figure 1|1 to a physical architecture, like the one shown in Figure 1|2, you'll finally see my blueprint for this book. From here on out, I am going to teach you how to build a secure, performant and scalable 4-Tier Merge Replication architecture that looks a lot like Figure 1|2.

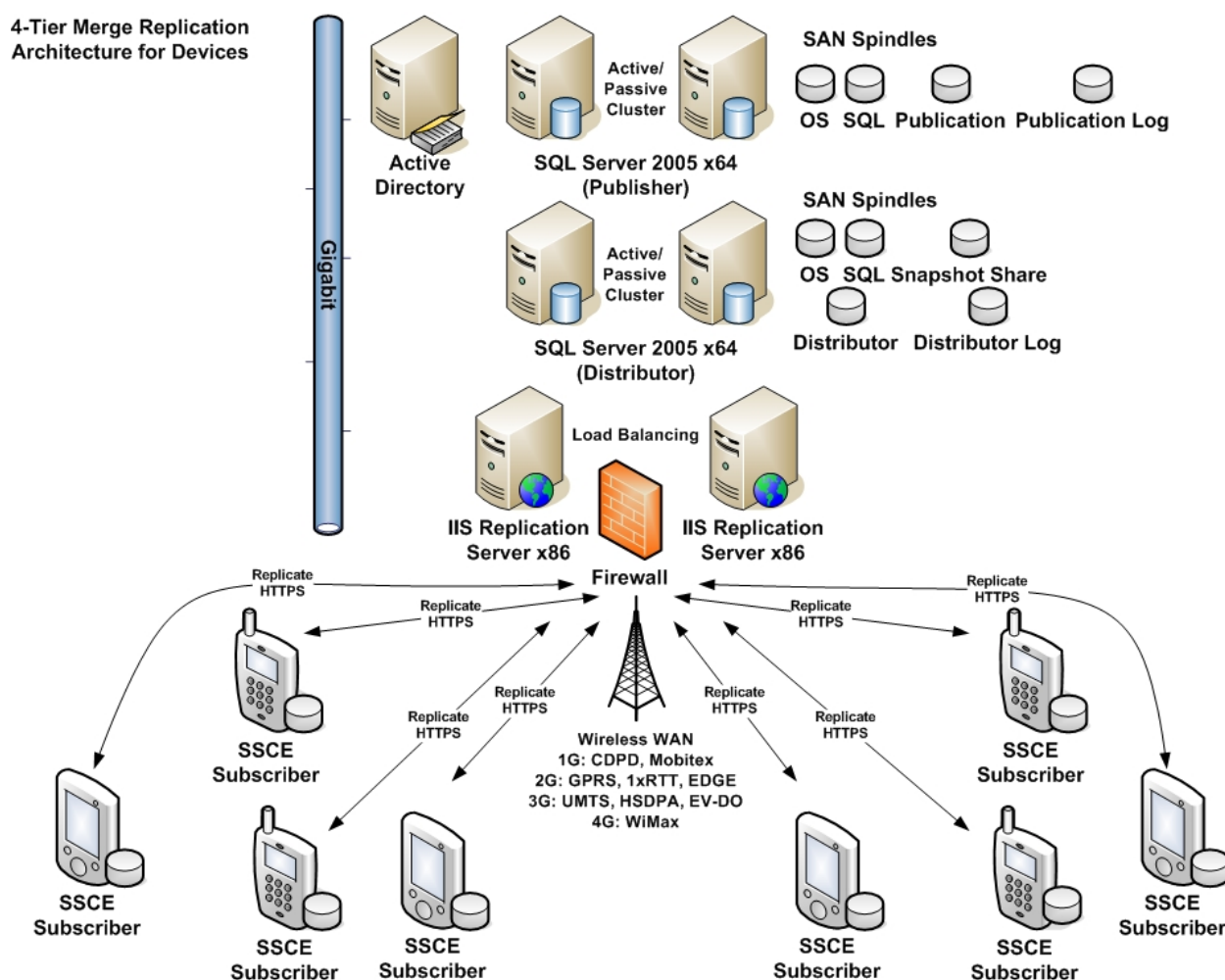


Figure 1|2. Physical Data Synchronization Architecture

Implementing the Blueprint

The combination of the logical and physical architecture diagrams plus the deep dives I'm going to take you on in the upcoming chapters will equip you to build a mobile data synchronization system that will satisfy the needs of the world's largest organizations. Rather than a mere academic exercise, you'll find this book to be a hands-on training manual. I won't waste your time with just theories and concepts so get ready to roll up your sleeves.

In order to create this book and give you a 100% accurate view of how to deploy this system in a large enterprise, I built a private network consisting of a Domain called SYNCDOMAIN, a Domain Controller called SYNCDC, an IIS server called SYNCWEB, a SQL Server Publisher called SYNCPUBLISHER, and a SQL Server Distributor called SYNCDISTRIBUTOR. You will see these names used over and over again throughout the book. When you're ready, I'll expect you to piece together your own network, Domain and servers. If you're lacking in the hardware department, you can recreate everything I'm going to teach on your laptop or desktop through the use of virtual machines running in Virtual PC 2007 or Virtual Server 2005 R2 SP1.

In the remaining pages of this chapter I'm going to have you create a Domain user that you will configure with appropriate Domain and local permissions. You will then restore a SQL Server database I've made available for download. This database will be used to illustrate various examples throughout the book.

In **Chapter 2**, I'll walk you through the configuration of the Distributor. It will start off by having you create and secure the Snapshot share. You will then move on to the **Configure Distribution Wizard** where you'll create and secure the Distribution database and create a connection to the Publisher server. I'll also show you how to disable distribution and publishing all together. I'll wrap up the chapter with some ongoing maintenance tips and commentary on how to improve the performance of your Distributor server.

In **Chapter 3**, I'll have you configure the Publisher which is the very heart of the system. You'll get to follow along through the **New Publication Wizard** where you'll learn about Articles, Column and Static filtering, download-only and upload-only tables, as well as Parameterized filters on the road to creating and securing your Publication. I'll also demonstrate tools that allow you to monitor many aspects of replication in real time so you can quickly identify and troubleshoot problems that might crop up. Just like in **Chapter 2**, I'll finish the chapter with tips on maintaining and improving the performance of your Publication database.

Chapter 4 takes you over to IIS where you get to install the SQL Server Client Components as well as the SSC Server Tools. I'll then take you on a trip through the **Configure Web Synchronization Wizard** where you'll create and secure Virtual Directories for use by the SSC Server Agent. I'll show you how to protect your IIS worker process from runaway memory usage and then I'll show you how to tweak your web server to get better performance and scalability.

Chapter 5 is for mobile developers who like coding with C# and the .NET Compact Framework 2.0. In this chapter I'll show you how to kick off data synchronization sessions from Windows Mobile by setting properties and calling a method in the Replication object. You'll learn how to perform asynchronous replication and I'll pass on some tips and tricks I've learned to make your synchronization operations more resilient to network dropouts and other failures.

So now that you know what is around the corner in future chapters, let's get started with the tasks I've laid out for you so you can start helping your customers or your own organization sooner rather than later.

Create a Domain user

Data Synchronization with Windows Mobile takes advantage of the security services provided by your network's Active Directory. The .NET Compact Framework code on your device sends the Domain, user name, and password credentials when it synchronizes its data. These credentials are used by IIS and SQL Server and thus both these servers have to verify these credential against a Domain Controller. Therefore, you need to create Domain users and groups that are allowed to participate in the replication of data between SQL Server and SQL Server Compact. For the purposes of this book, you'll create a Domain user that will be used for all the examples and exercises so let's get started. Log on to your Domain Controller and from the Start menu, select **All Programs | Administrative Tools**, and then click on **Active Directory Users and Computers** as shown in Figure 1 | 3.

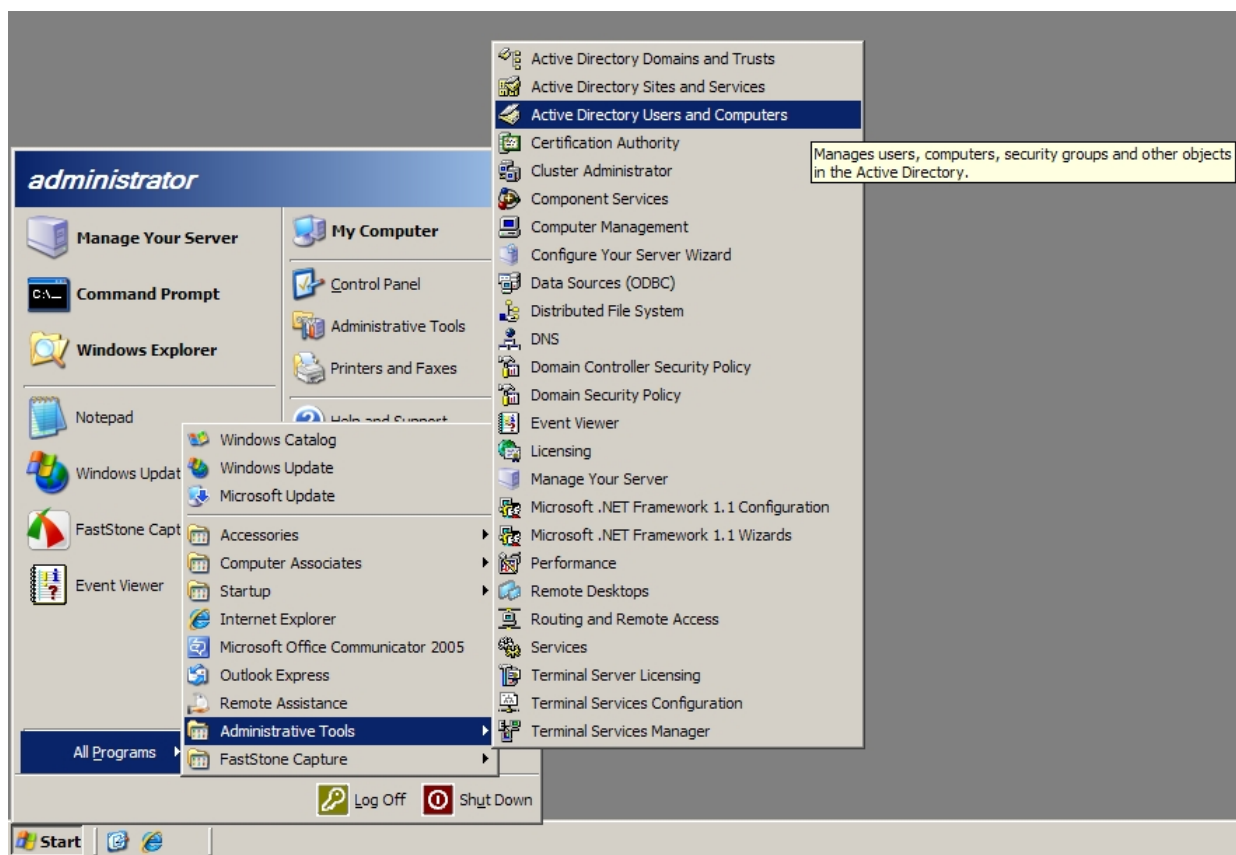


Figure 1 | 3. Active Directory Computers and Users

When the Active Directory Users and Computers dialog opens, expand the **syncdomain.internal** node and select the Computers node to verify that **SYNCDISTRIBUTOR**, **SYNCPUBLISHER**, and **SYNCWEB** are all listed as members of the Domain as shown in Figure 1 | 4.

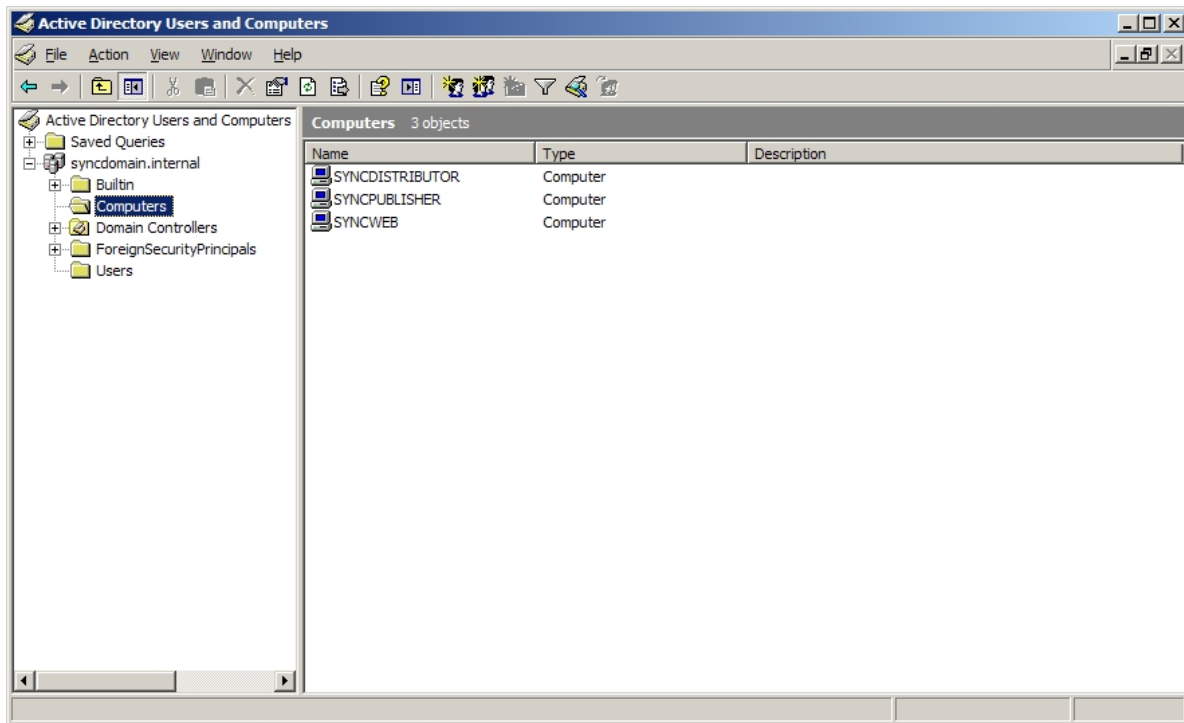


Figure 1 | 4. Computers

Now skip down and highlight the **Users** node. Right click on the **Users** node and select **New | User** as shown in Figure 1 | 5.

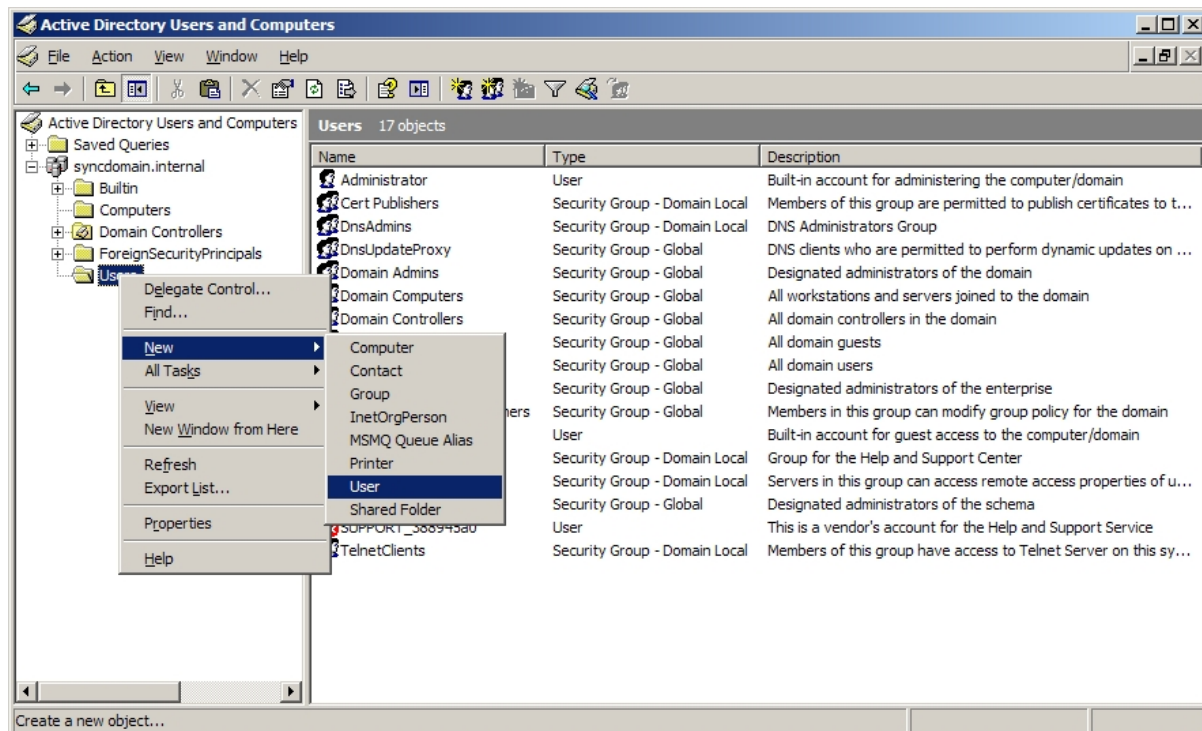
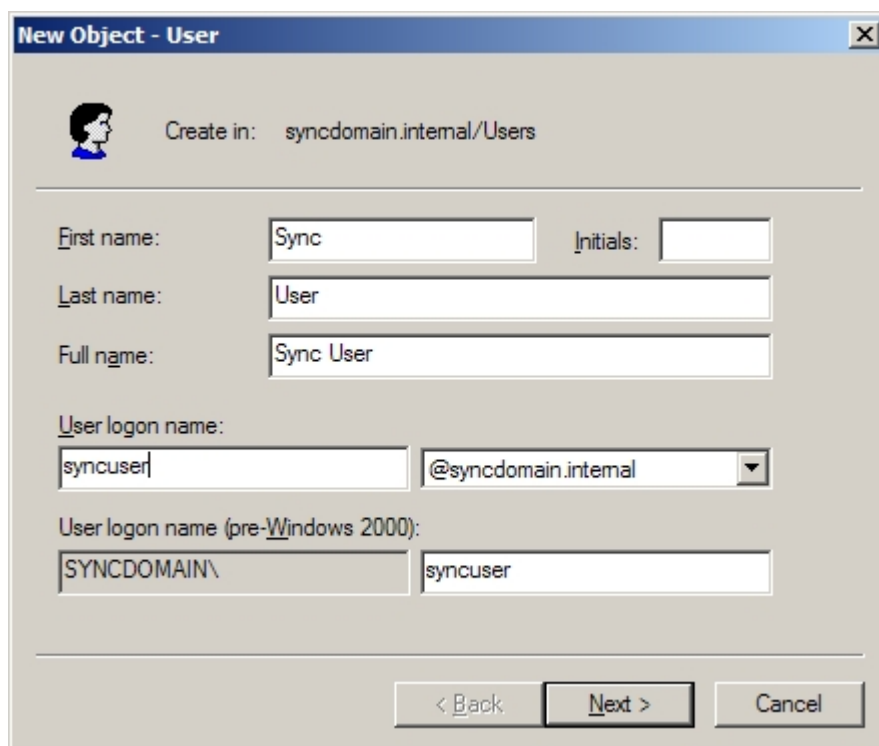


Figure 1 | 5. Select New User

This will bring up the **New Object – User** dialog. Enter **Sync** in the **First name** text box, **User** in the **Last name** text box, and **syncuser** in the **User logon name** text box as shown in Figure 1 | 6 and click **Next**.



New Object - User

Create in: syncdomain.internal/Users

First name: Sync Initials:

Last name: User

Full name: Sync User

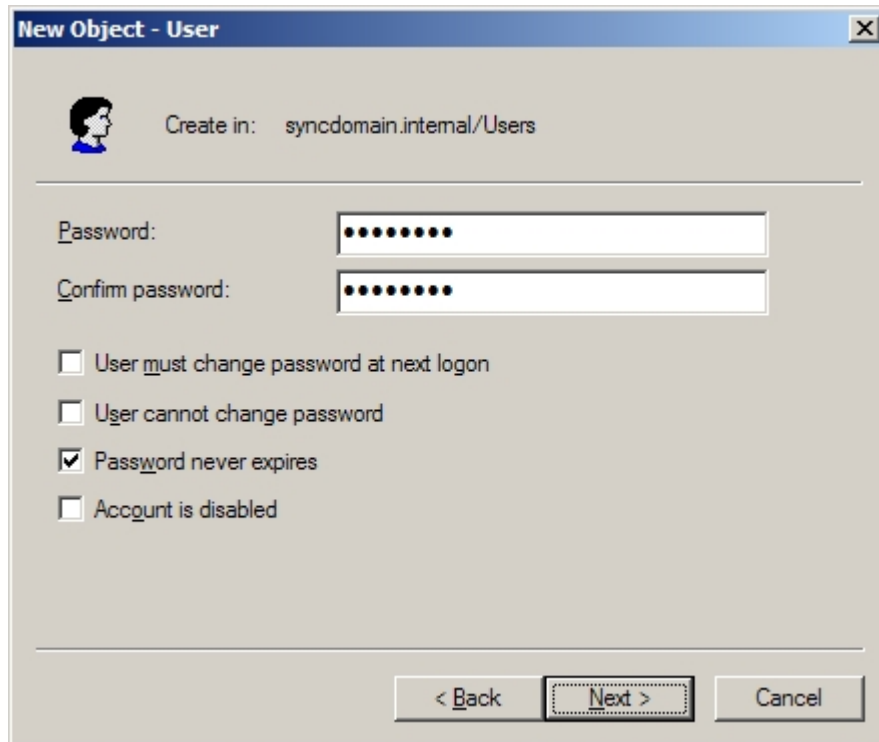
User logon name: syncuser @syncdomain.internal

User logon name (pre-Windows 2000): SYNCDOMAIN\ syncuser

< Back Next > Cancel

Figure 1 | 6. New Object - User

In the next **New Object – User** dialog that appears, enter **P@ssw0rd** in both the **Password** and **Confirm password** text boxes as shown in Figure 1 | 7. Ensure that only the **Password never expires** check box is checked and click **Next**.



The screenshot shows a Windows-style dialog box titled "New Object - User". At the top, it says "Create in: syncdomain.internal/Users" next to a user icon. Below this are two text boxes for "Password:" and "Confirm password:", both filled with masked characters (dots). Underneath the text boxes are four checkboxes with the following labels: "User must change password at next logon", "User cannot change password", "Password never expires", and "Account is disabled". The "Password never expires" checkbox is checked. At the bottom of the dialog are three buttons: "< Back", "Next >", and "Cancel".

Figure 1 | 7. New Object – User Password

The last **New Object – User** dialog appears and displays a summary of your choices as shown in Figure 1 | 8. Ensure that they are correct and click **Finish**. You can now close the **Active Directory Users and Computers** dialog.

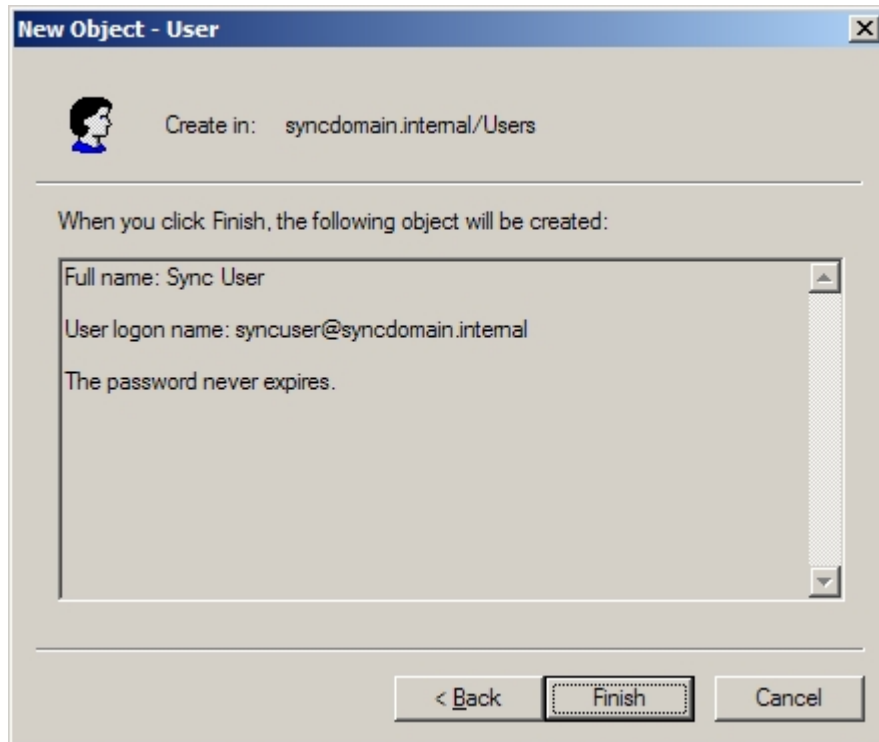


Figure 1 | 8. New Object – User Create

The **syncuser** you created is just a Domain user when it comes to the level of power it has on the network. While this is appropriate network-wide, I need you to elevate **syncuser's** privileges on the servers involved in the Merge Replication activities. Logon to SYNC PUBLISHER, SYNC DISTRIBUTOR, AND SYNC WEB and add syncuser to the local Administrators group of each server. This will ensure that SQL Server and IIS will have the rights they need to perform all their tasks.

Create the Database

Throughout the book, you'll be utilizing a simple database called ParkSurvey designed to help illustrate different aspects of merge replication. From your SQL Server Publisher server, copy the ParkSurvey.bak file that accompanies this book to the hard drive where you can find it later in the chapter. Launch the **SQL Server Management Studio** and connect to the local **Database Engine**. In the Object Explorer, right click on the **Databases** node and select **New Database...** as shown in Figure 1 | 9.

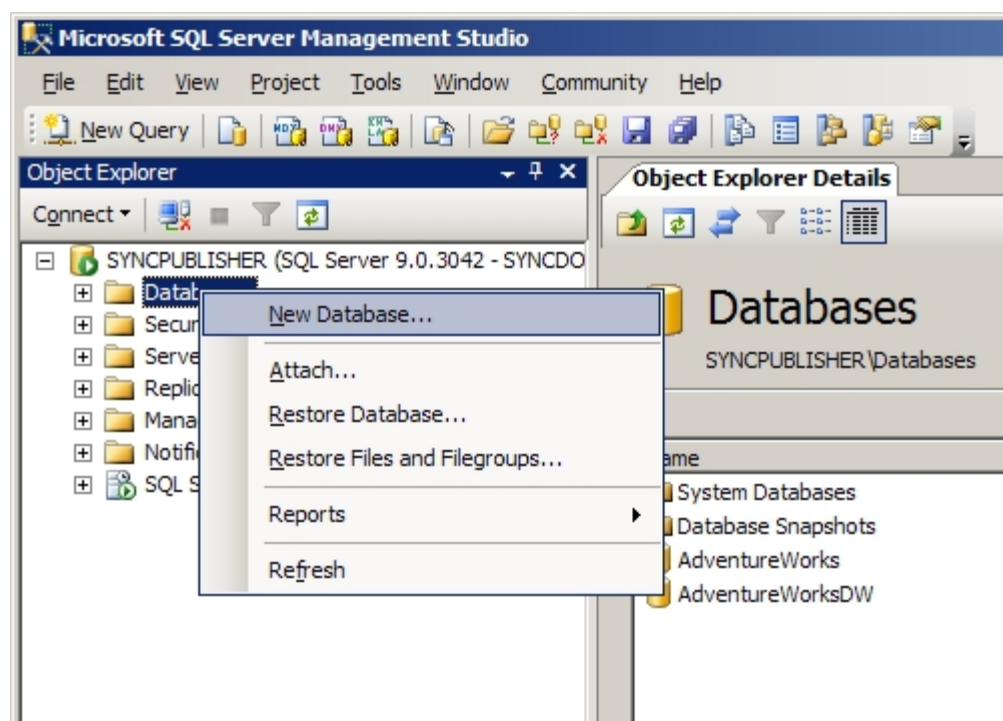


Figure 1 | 9. New Database from Object Explorer

The **New Database** dialog will appear as shown in Figure 1 | 10. In the **Database name** text box, type **ParkSurvey** and then skip down to the **Database files** grid below. Once in the grid, click on the first row and then tab over until you reach the Path column to define where the database file will reside. If you're running your Publisher on a server with only one hard disk, don't make any changes. Otherwise, I recommend you edit the path so it points to a disk drive that is separate from the drives where Windows Server and SQL Server reside to ensure best I/O performance. With that done, skip down to the second row and repeat the same procedure with the database transaction log file. Again, try to put the log file on a drive other than the one where the database file, Windows Server or SQL Server resides. Click **OK** to create the new database and log. Use Windows Explorer to verify that the .mdf and .ldf files were created on the proper drives.

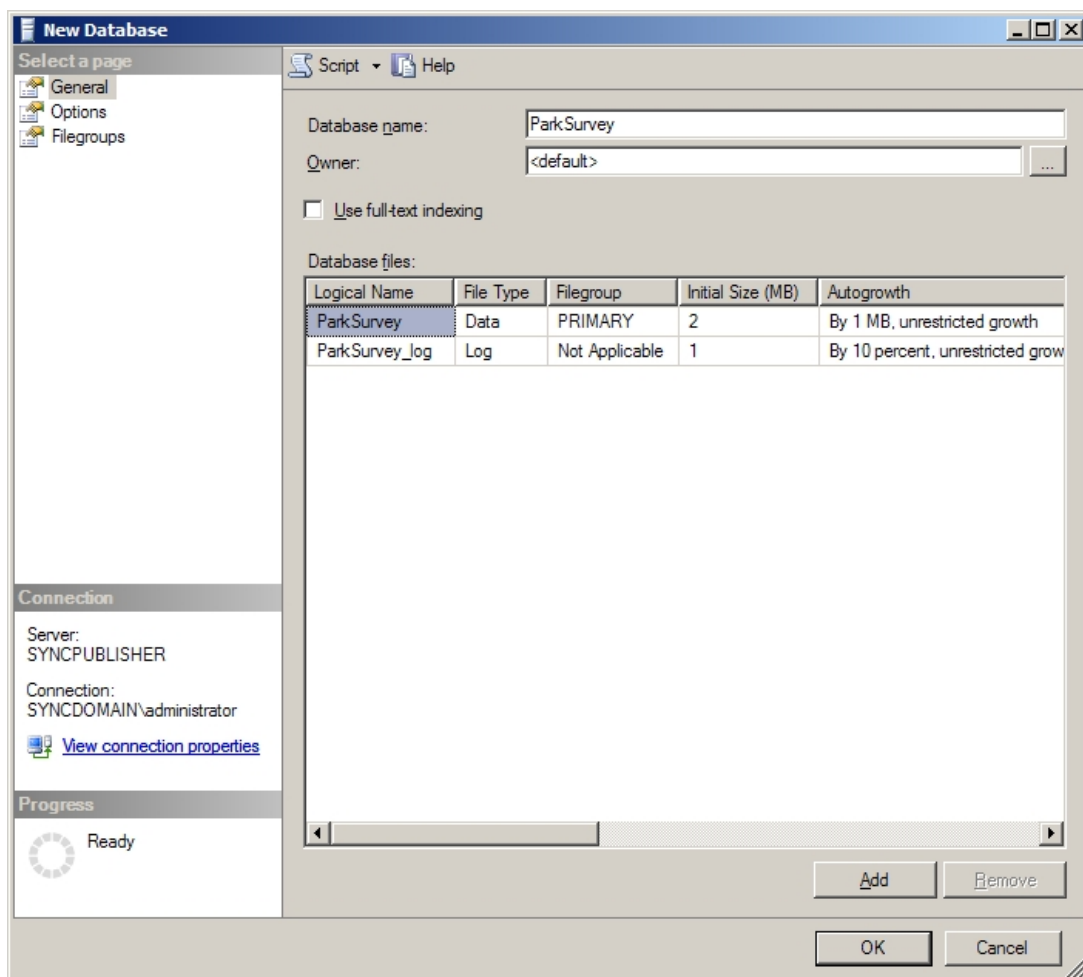


Figure 1 | 10. New Database dialog

Expand the **Databases** node, right click on your new **ParkSurvey** node and select **Tasks**, **Restore**, and **Database...** as shown in Figure 1 | 11.

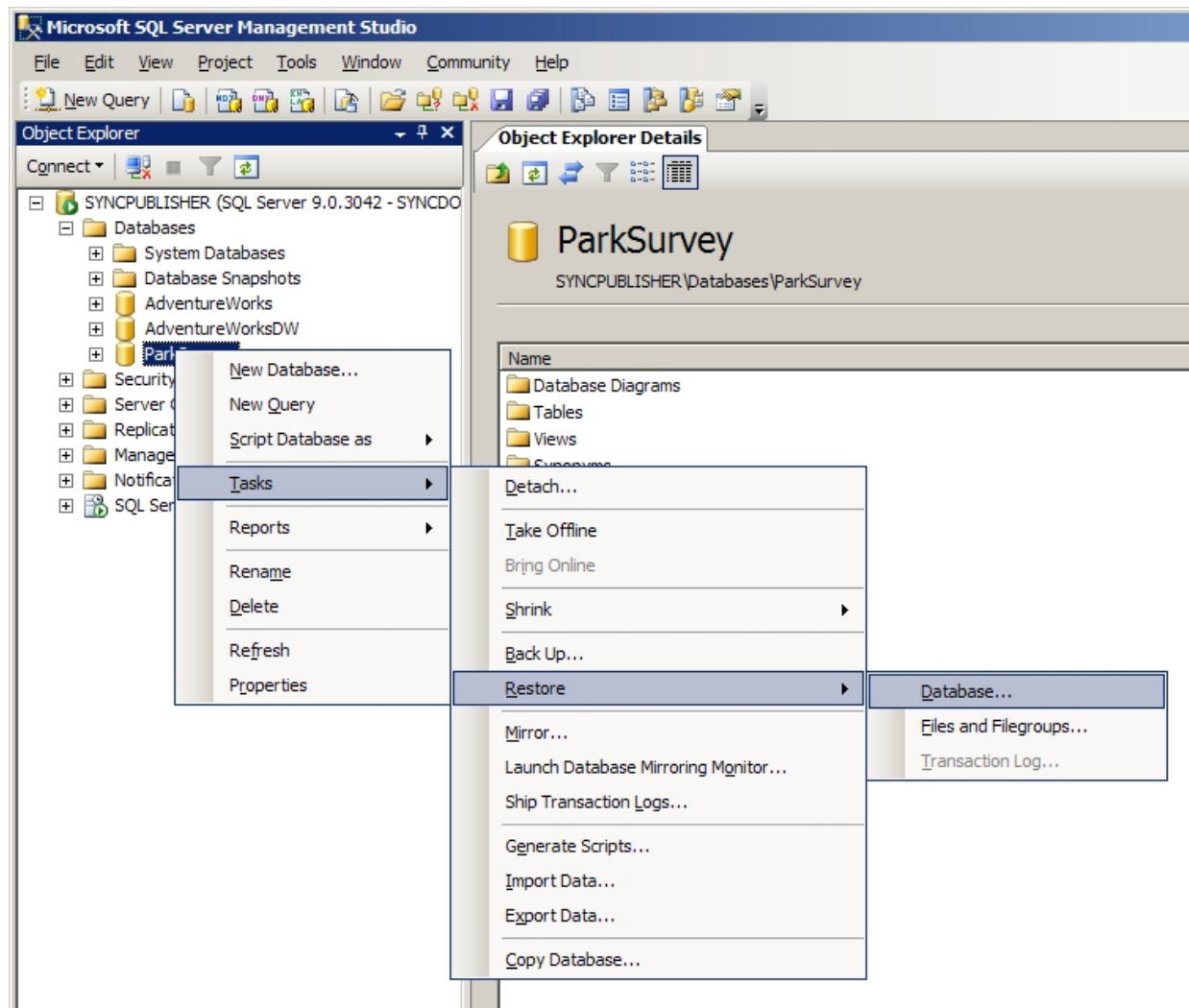


Figure 1 | 11. Restore database from Object Explorer

When the **Restore Database – ParkSurvey** dialog appears as shown in Figure 1|12, select the **From device** radio button and then click the ... (ellipsis) button on the right.

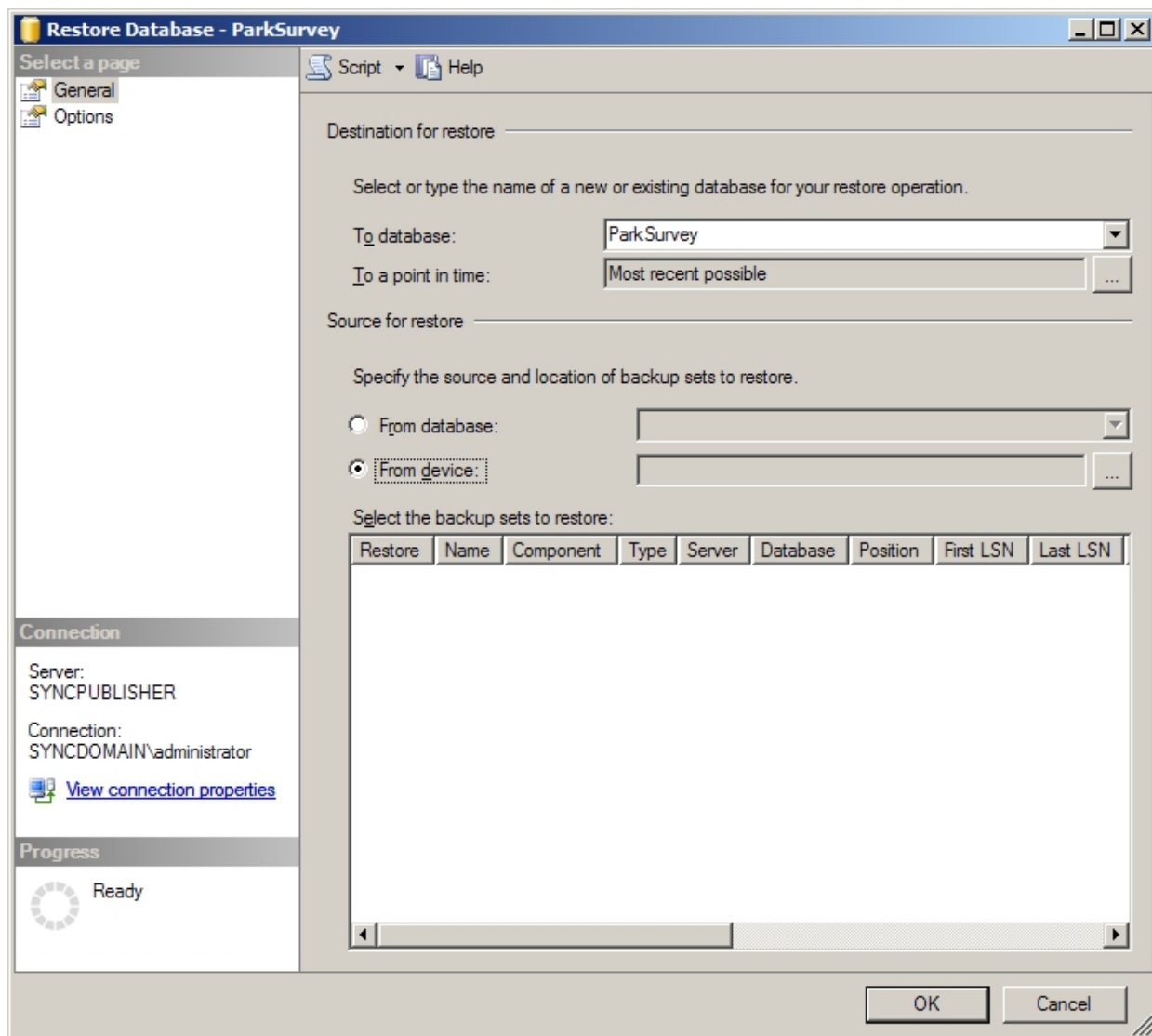


Figure 1|12. Restore Database - ParkSurvey

The **Specify Backup** dialog will appear as shown in Figure 1|13. Select **File** from the **Backup media** combo box and then click the **Add** button.

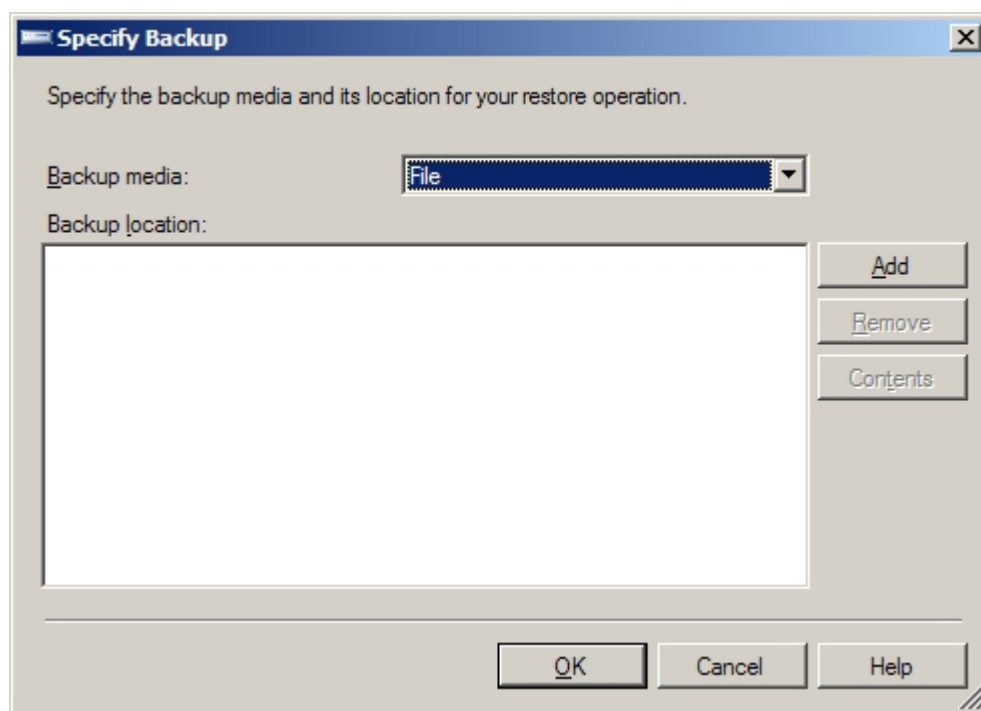


Figure 1|13. Specify Backup

The **Locate Backup File – SYNCPUBLISHER** dialog will appear defaulted to SQL Server's Backup directory as shown in Figure 1 | 14. Navigate to, and highlight the **ParkSurvey.bak** file wherever it may be found on your server's hard drive and click OK.

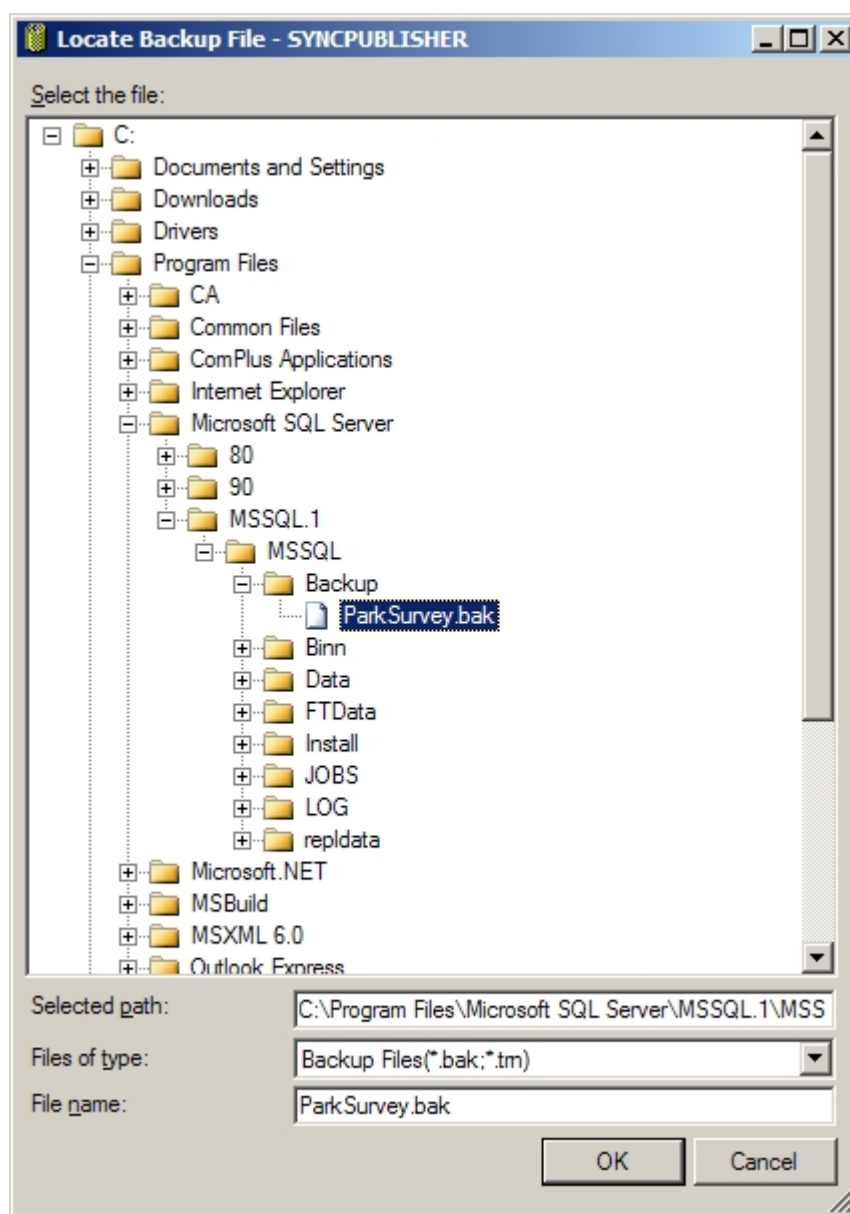


Figure 1 | 14. Locate Backup file

You will be returned to the **Specify Backup** dialog. This time, the path to the **ParkSurvey.bak** file will be listed in the **Backup location** list box as shown in Figure 1 | 15. Click the **OK** button.

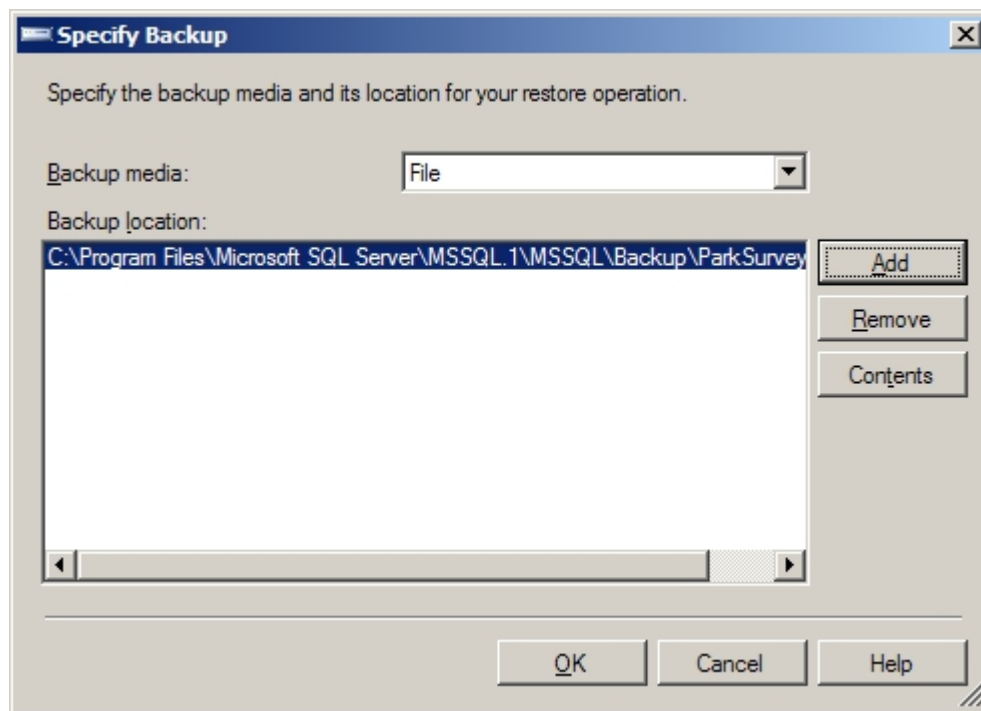


Figure 1 | 15. Specify Backup

The **Restore Database – ParkSurvey** dialog new displays your **ParkSurvey** backup in the **Select the backup sets to restore** list view as shown in Figure 1|16. Select the **Restore** check box.

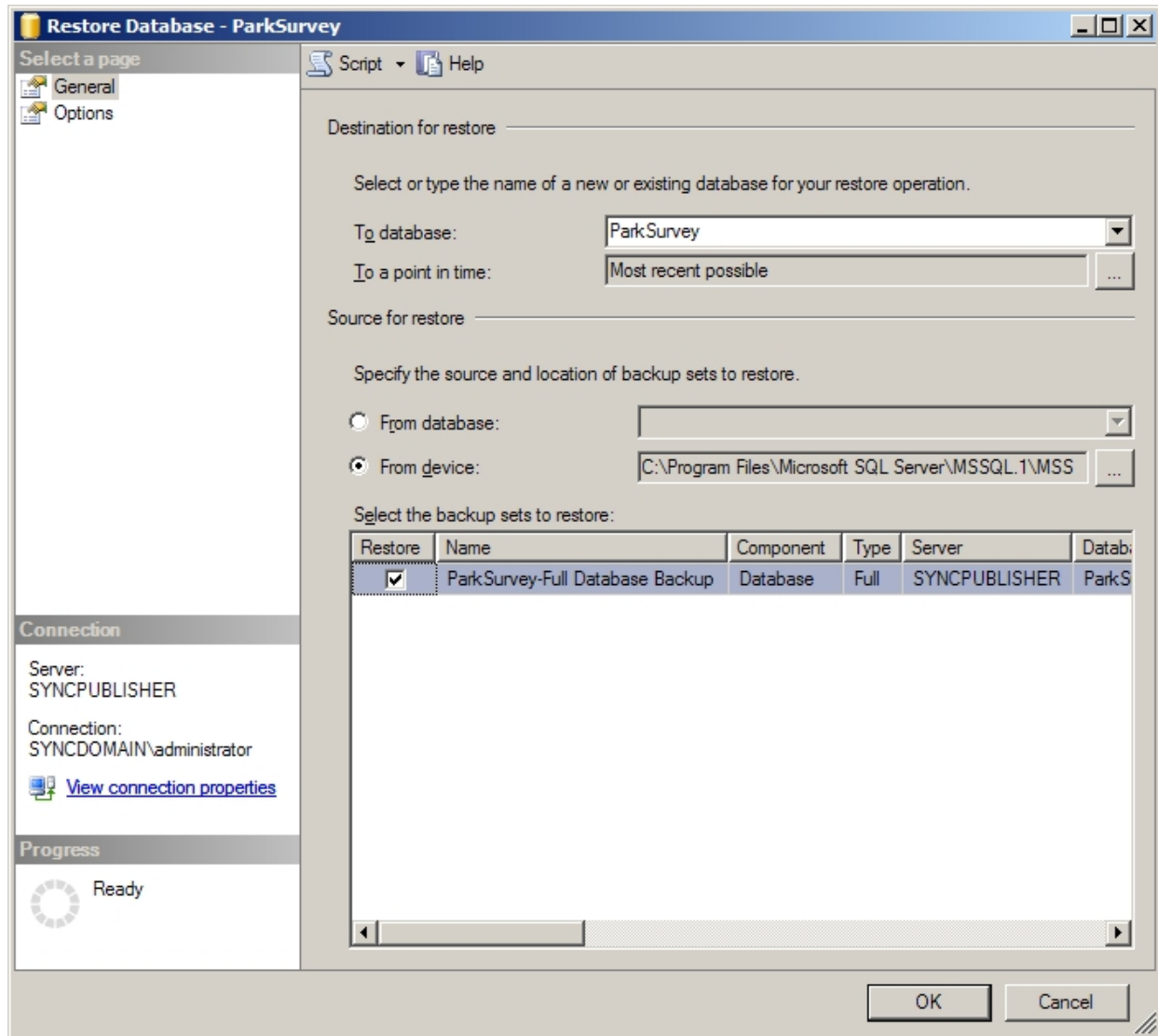


Figure 1|16. Restore Database - ParkSurvey

In the **Select a page** section on the left side of the dialog, select **Options**. Check the **Overwrite the existing database** check box as shown in Figure 1 | 17 and then navigate to the Restore **the database files** as grid. To ensure that the backup is being restored as the same name and to the same location as the ParkSurvey database you created, click the ... (ellipsis) button on the first row.

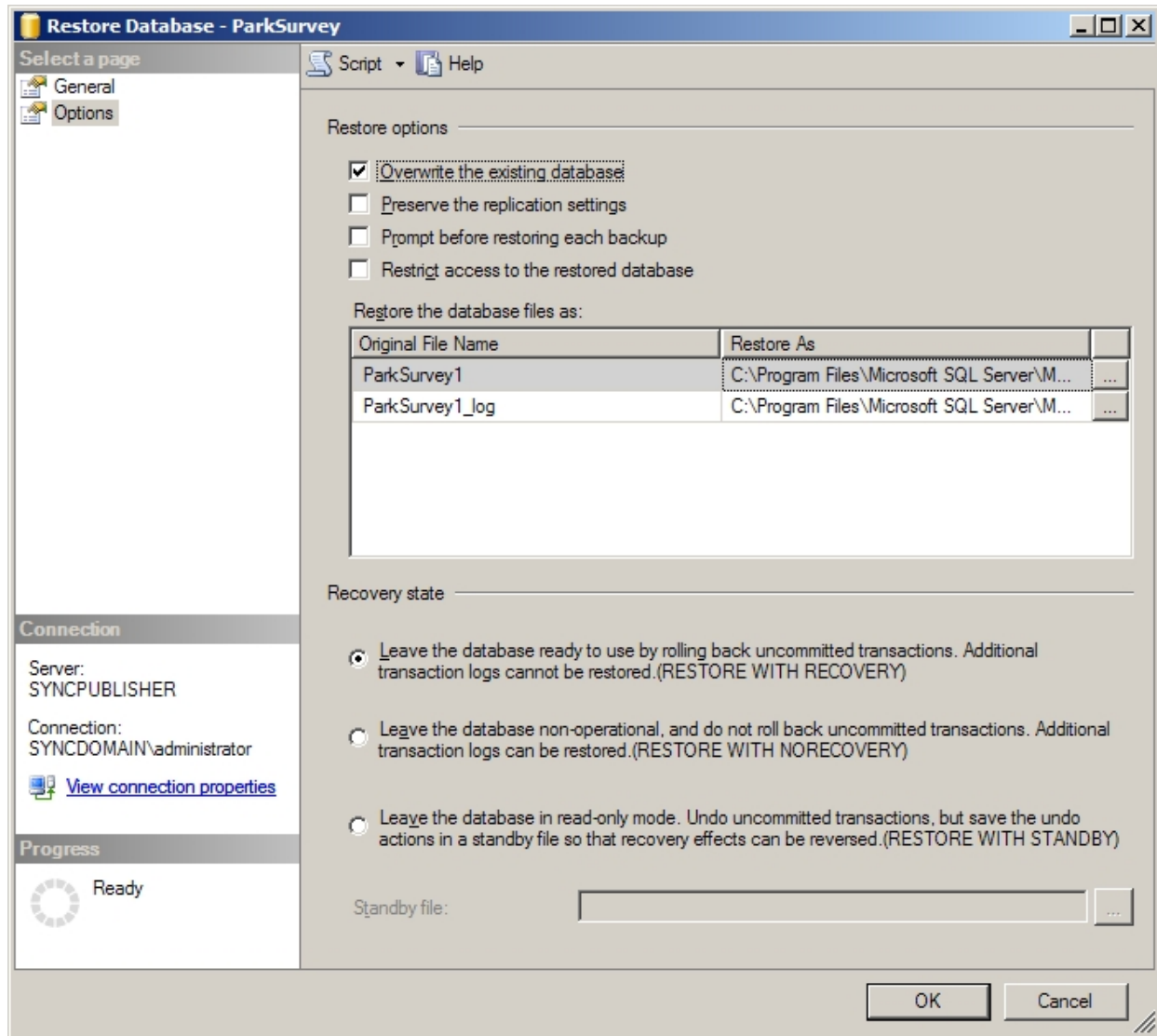


Figure 1 | 17. Restore Database – ParkSurvey Options

This will bring up the **Locate Database Files – SYNC PUBLISHER** dialog as shown in Figure 1 | 18. Navigate to and highlight the **ParkSurvey.mdf** file that was created when you created the ParkSurvey database and click **OK**. Skip down to the next row in the grid and repeat the process, except this time navigate to and highlight the **ParkSurvey_log.ldf** file and click **OK**. Verify that the paths in both the grid rows point to the proper database file and log and then click **OK**.

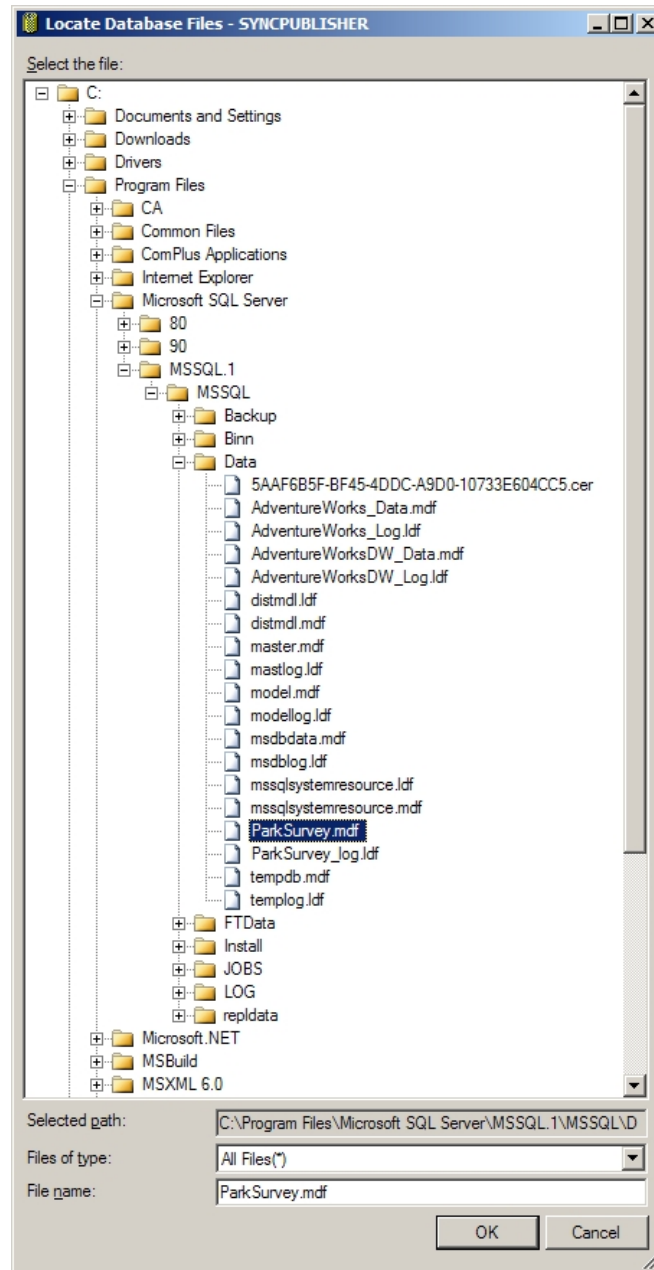


Figure 1 | 18. Locate Database Files - SYNC PUBLISHER

If everything was configured properly, a dialog saying **The restore of database 'ParkSurvey' completed successfully** will appear. From the **Object Explorer**, expand the **ParkSurvey** node and its **Tables** node to verify that the **Cities**, **Parks**, and **Survey** tables were created and contain data as shown in Figure 1 | 19.

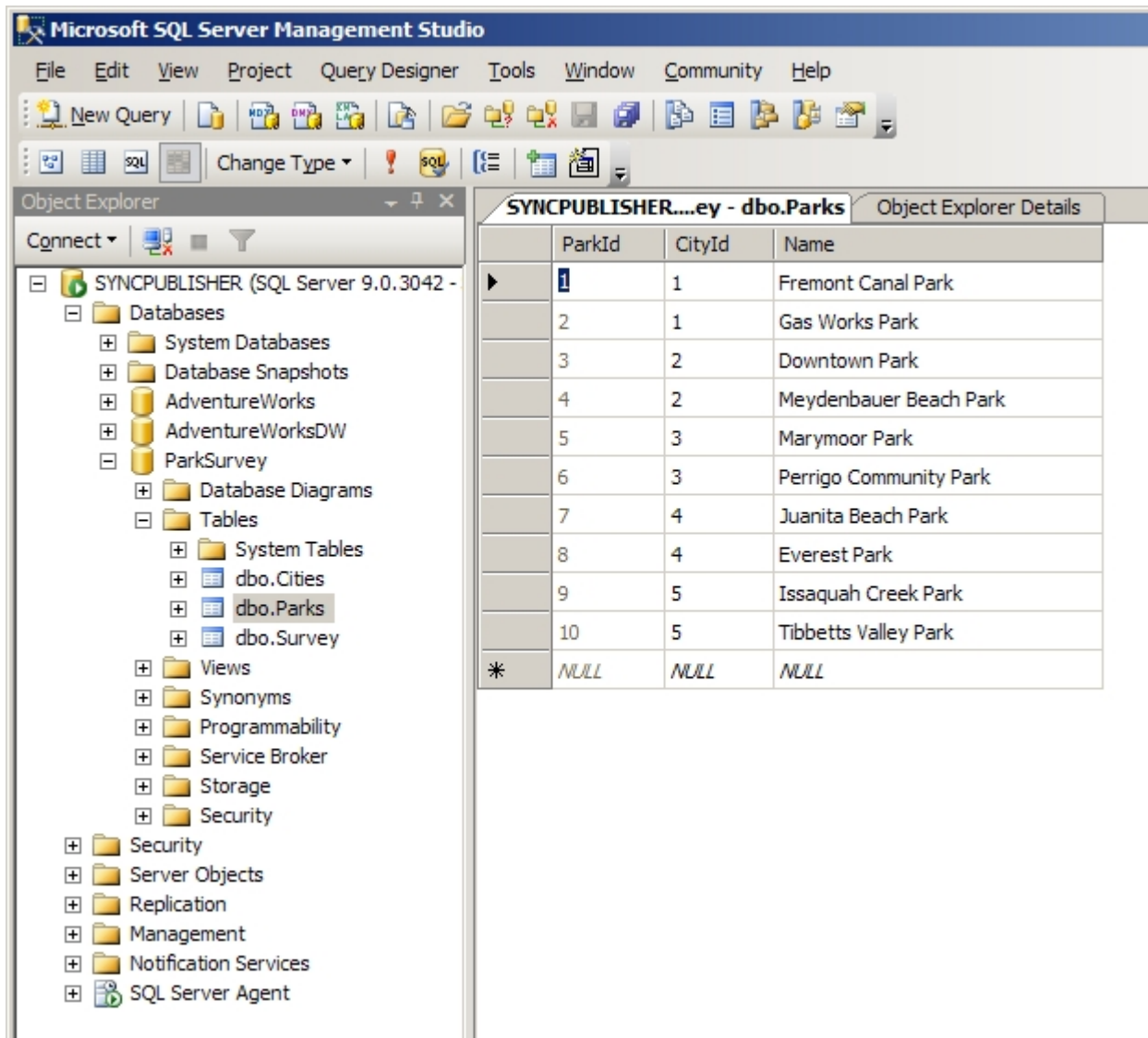


Figure 1 | 19. Expanded ParkSurvey

With your ParkSurvey database restored, you're now ready to move on to Chapter 2 where you'll get to create the Distributor.