13

Microsoft SQL Server 2008®

In This Chapter

About SQL Server 2008

GoldMine Premium can no longer use the Firebird back end, and it is only capable of utilizing the Microsoft Structured Query Language (**SQL**) back end. A GoldMine Premium Edition license will permit the owner to configure GoldMine for the SQL Server back end. In fact, as of this writing, GoldMine Premium is delivered with Microsoft SQL Server 2008 for Workgroups with per seat Client Access Licenses (**CALs**). Most organizations, having this type of license, will and should install the SQL back end on the Server. I do recommend Microsoft SQL Server 2008 Express with Tools for your Remote installations of GoldMine.

Your organization today faces numerous data challenges. You need your people to make faster and more data-driven decisions, your developers to be more productive and flexible, and your managers to reduce their overall information technology (**IT**) budgets even as they scale your infrastructure to meet ever increasing demands.

SQL Server 2008 is designed to help enterprises address these challenges while enhancing performance in a Server 2008/Windows 7 environment. This next-generation data management and analysis solution delivers increased security, scalability, and availability to enterprise data and analytical applications, while making them easier to build, deploy, and manage.

Extending the strengths of SQL Server 2005, SQL Server 2008 provides an integrated data management and analysis solution that can help your staff do the following:

- Build, deploy, and manage enterprise applications that are more secure, scalable, and reliable.

- Maximize IT productivity by reducing the complexity of developing and supporting database applications.

- Share data across multiple platforms, applications, and devices to make it easier to connect internal and external systems.

- Control costs without sacrificing performance, availability, scalability, or security.

SQL Server 2008 advances your data infrastructure in three key areas: it makes your enterprise data more manageable, your developers more productive, and your business intelligence (**BI**) more comprehensive. It also breaks new ground in affordable pricing and licensing, upgrade paths to SQL Server 2008, and the Microsoft Windows Server System.

It is important to note that all of the above is virtually meaningless for the typical GoldMine user. In fact, most GoldMine Partners only know enough about SQL Server 2008 to get GoldMine Premium functioning, and to possibly create a SQL Maintenance Plan. Of course there is always the exception to the case, but you should be aware that, although supplied by FrontRange, that even FrontRange will not support your SQL Server 2008 needs. If you run into any issues you are expected to contact the Microsoft SQL Server Support Team for assistance.

You should follow the recommendations as put forth by GoldMine Premium, aka FrontRange Solutions. I, on the other hand, will be showing you the new connectivity setup, and use of a new GoldMine Contact Set. You see, as I have said earlier, and throughout this book, GoldMine no longer uses the Borland Database Engine bridge between the GoldMine application, and the database. New with the release of GoldMine 7 Corporate Edition was the removal of the dependency on BDE, and the application of the new bridge, **A**ctiveX **D**ata **O**bjects (**ADO**). The ADO bridge has continued through to the GoldMine Premium Edition.

Server Properties

Many people ask me how I configure SQL Server. Well, I learned by trial and error. So what I'm going to show you here is what I've learned, and do not necessarily represent the best practices. These do, however, work for me in my environment as well as for most of my clients.

Not knowing anything about SQL Server, I installed SQL Server using the default configurations. Save the fact that I installed SQL Server in **Mixed Authentication Mode**, I changed nothing. Let's take a look at the **Properties** for my SQL Server installation, and we do this from the **Microsoft SQL Server Management Studio**. I simply highlight the server name, and then right click on it to select **Properties** from the local menu which produces this dialog form:

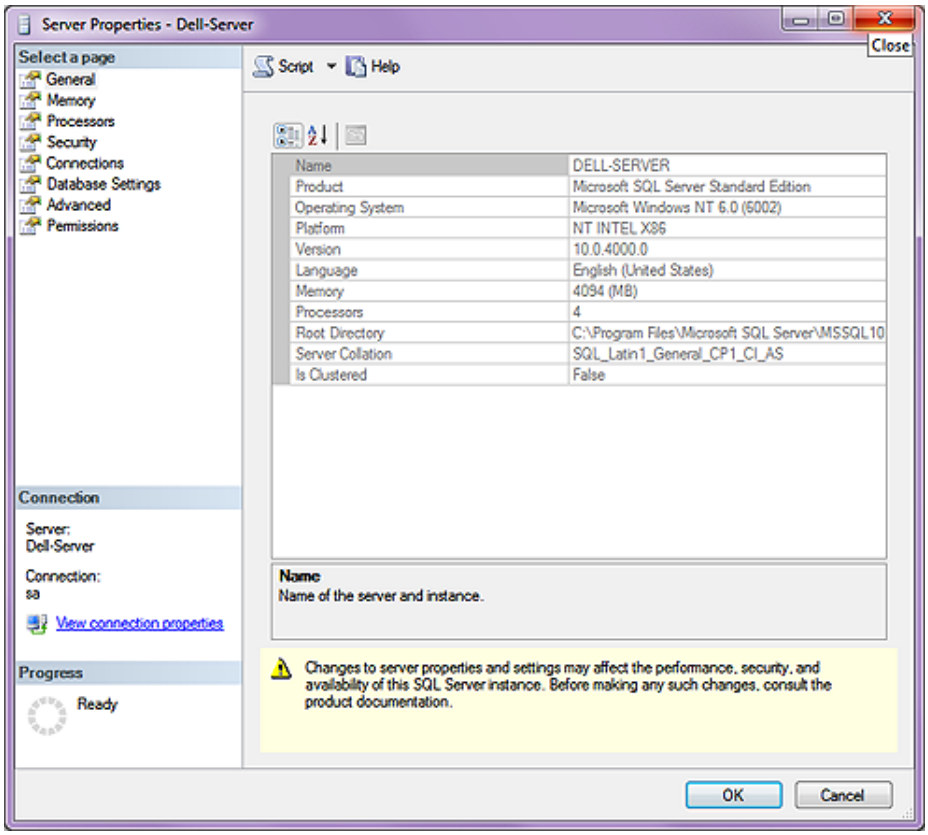

Figure 13-1

You will notice the tree to the left side of figure 13-1, and at the **General** branch is selected when you open this dialog form although, I must say, on my system, the highlighting is barely visible. You may have noticed that on the right half of the dialog form, that this information is grayed out (disabled) as this is only informational, and no changes can be made from this branch.

Once we select the **Memory** branch, however, we can see that there are items that may be modified. I won't try to explain these in any more detail than can be acquired via the Help application, however, suffice it to know that I have made no changes on this screen. Pertaining to Figure 13-2 on the next page, and taken directly from the Server Properties | Help menu item:

Server memory option

☐ **Use AWE to allocate memory**

Specifies that SQL Server will take advantage of Address Windowing Extensions (AWE) in Microsoft Windows 2000 and Windows Server 2003 to support up to 64 gigabytes (GB) of physical memory. AWE only applies to 32-bit operating systems. To use AWE you must configure Windows settings in addition to this SQL Server setting. To set this option, you must configure the lock pages in memory policy.

Minimum server memory (in MB)

Specifies that SQL Server should start with at least the minimum amount of allocated memory and not release memory below this value. Set this value based on the size and activity of your instance of SQL Server. Always set the option to a reasonable value to ensure that the operating system does not request too much memory from SQL Server and inhibit Windows performance.

Note

After all that hemming and hawing about how I have all of the default settings in my screenshots, I have to tell the truth. Since The Hacker's Guide to GoldMine Premium, I have made some tweaks to my SQL Server configuration, and this is the first.

Minimum server memory (in MB)

Default:

0 Megabyte

DJs Tweak:

1073741823 Megabyte

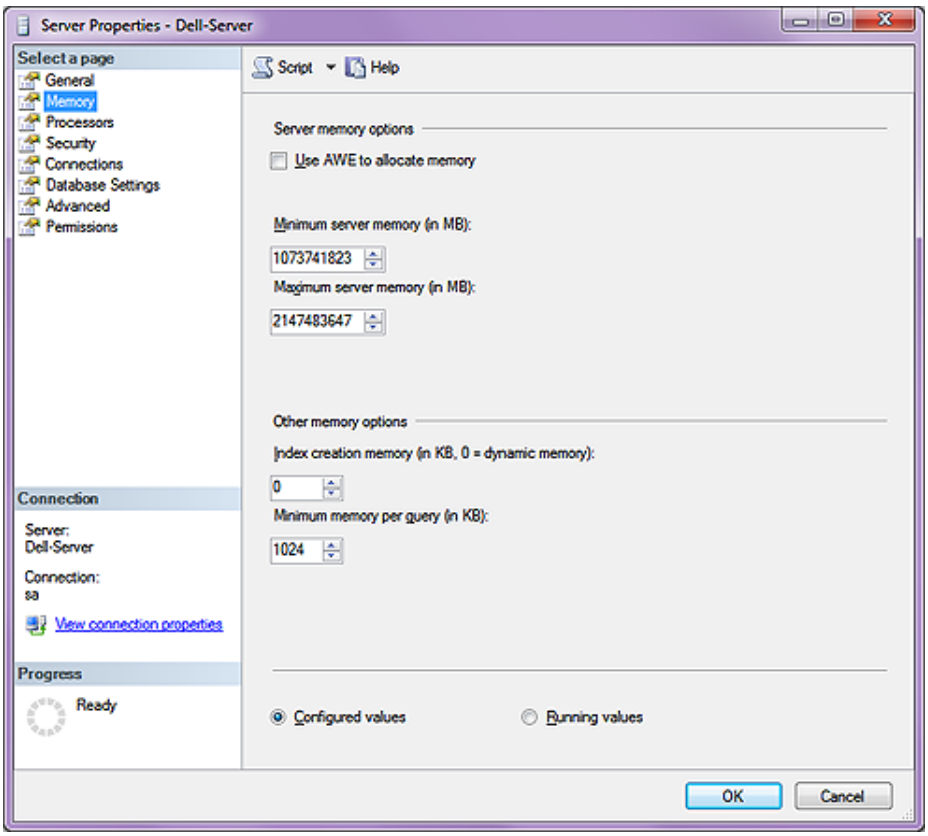

Figure 13-2

Note

Maximum server memory (in MB) *is a setting that many GoldMine Administrators have been known to tweak to their specific needs.*

Maximum server memory (in MB)

Specifies the maximum amount of memory SQL Server can allocate when it starts and while it runs. This configuration option can be set to a specific value if you know there are multiple applications running at the same time as SQL Server and you want to guarantee that these applications have sufficient memory to run. If these other applications, such as Web or e-mail servers, request memory only as needed, then do not set the option, because SQL Server will release memory to them as needed. However, applications often use whatever memory is available when they start and do not request more if needed. If an application that behaves in this manner runs on the same computer at the same time as SQL Server, set the option to a value that guarantees that the memory required by the application is not allocated by SQL Server.

Other memory option

Note

Index creation memory (in KB, 0 = dynamic memory)

Values from 1 to 703 are not allowed. If a value in this range is entered, the field overrides the entered value with 704.

Index creation memory (in KB, 0 = dynamic memory)

Specifies the amount of memory (in kilobytes) to use during index creation sorts. The default value of zero enables dynamic allocation and should work in most cases without additional adjustment; however, the user can enter a different value from 704 to 2147483647.

Note

Minimum memory per query (in KB) *is a setting that many GoldMine Administrators have been known to tweak to their specific needs.*

Please note that this allocation is in Kilobytes, and not in Megabytes.

Minimum memory per query (in KB)

Specifies the amount of memory (in kilobytes) to allocate for the execution of a query. The user can set the value from 512 to 2147483647. The default value is 1024.

◉ **Configured Values**

Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

Running Values

View the currently running values for the options on this pane. These values are read-only.

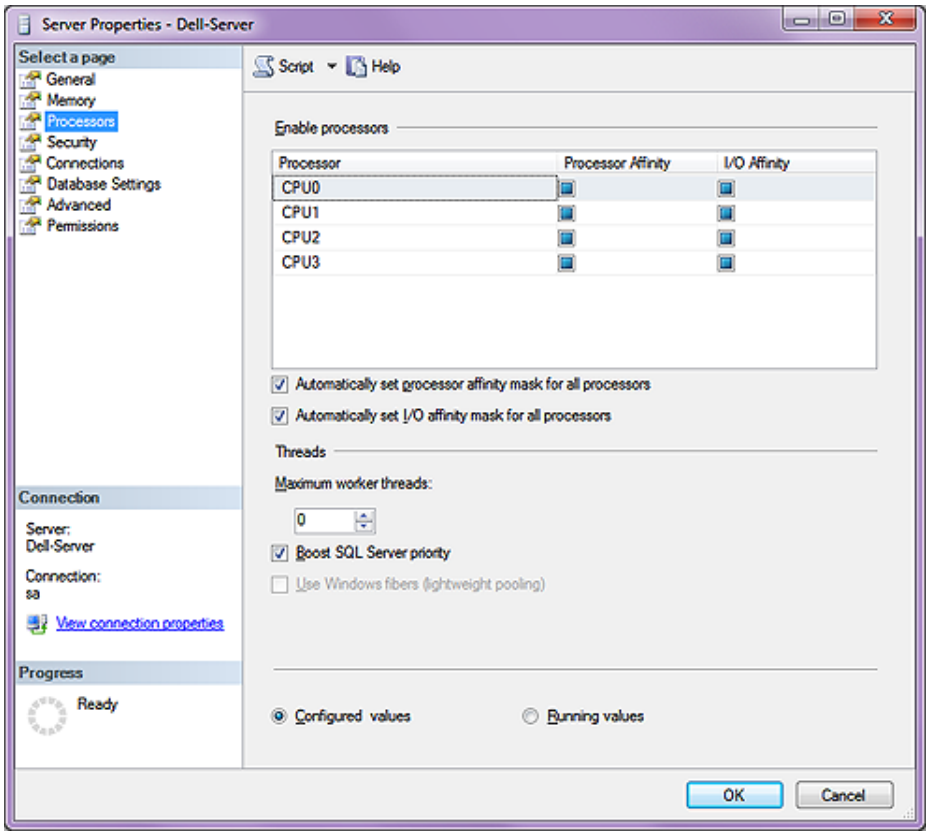

Figure 13-3

Enable processors

Processor Affinity

Assigns processors to specific threads to eliminating processor reloads and reduce thread migration across processors. For more information, see affinity mask Option.

I/O Affinity

Binds Microsoft SQL Server disk I/Os to a specified subset of CPUs. For more information, see affinity I/O mask Option.

☑ **Automatically set processor affinity mask for all processors**

Allows SQL Server to set the processor affinity.

☑ **Automatically set I/O affinity mask for all processors**

Allows SQL Server to set the I/O affinity.

Threads

Maximum worker threads

0 allows SQL Server to dynamically set the number of worker threads. This setting is best for most systems. However, depending on your system configuration, setting this option to a specific value sometimes improves performance. For more information, see max worker threads Option.

☑ **Boost SQL Server priority**

Specifies whether SQL Server should run at a higher Microsoft Windows scheduling priority than other processes on the same computer. For more information, see priority boost Option.

☐ **Use Windows fibers (lightweight pooling)**

Use Windows fibers instead of threads for the SQL Server service. Note that this is only available in Windows 2003 Server Edition. For more information, see lightweight pooling Option.

- **Configured Values**

  Displays the configured values for the options on this pane. If you change these values, click Running Values to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

- **Running Values**

  View the currently running values for the options on this pane. These values are read-only.

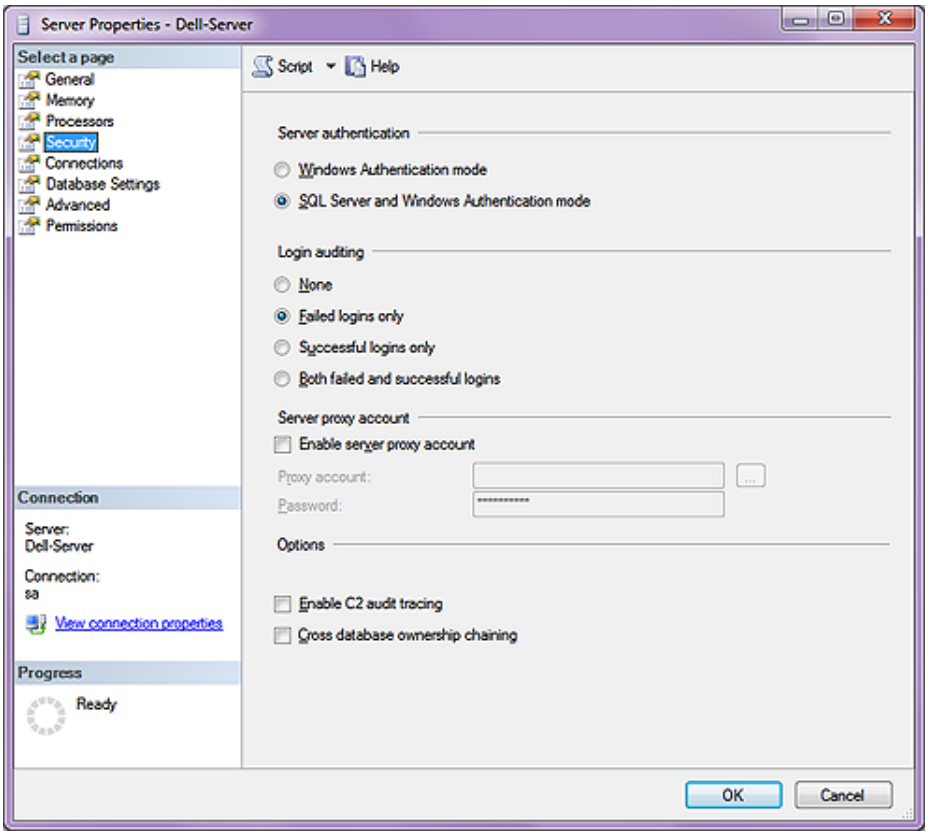

Figure 13-4

Figure 13-4 represents the **Security** branch of the tree, and its options. Most importantly, this is where I had to correct my first mistake made during my initial install of SQL Server 2005. I will again pull out the information from these Server Properties | Help application, however, I will annotate on the sidebar where I corrected my mistake.

Server Authentication

- **Windows Authentication mode**

  Uses Windows Authentication to validate attempted connections. If the sa password is blank when the security mode is being changed, the user is prompted to enter an sa password.

- **SQL Server and Windows Authentication mode**

  Uses mixed mode authentication to verify attempted connections, for backward compatibility with earlier versions of SQL Server. If the sa password is blank when the security mode is being changed, the user is prompted to enter an sa password.

Note

*Had I had chosen **Windows Authentication Mode** when installing SQL Server 2008, when in fact, I should have chosen **Mixed Mode** this, then, is where I would correct my mistake by selecting the radio button option for **SQL Server and Windows Authentication Mode**.*

*It is important to note that the **Windows Authentication Mode** is much more secure than the **SQL Server Authentication Mode**. When possible, you should use the **Windows Authentication Mode**, however, this is not possible with GoldMine Premium.*

*Changing the security configuration requires a restart of the service. When changing the **Server Authentication** to **SQL Server and Windows Authentication mode** the **sa** account is not automatically enabled. To use the **sa** account, execute **ALTER LOGIN** with the **ENABLE** option.*

Note

Changing the audit level requires re-starting the service.

Login Auditing

- ○ **None**

  Turns off login auditing.

- ◉ **Failed logins only**

  Audits unsuccessful logins only.

- ○ **Successful logins only**

  Audits successful logins only.

- ○ **Both failed and successful logins**

  Audits all login attempts.

WARNING

The login used by the server proxy account should have the least privileges required to perform the intended work. Excessive privileges for the proxy account could be used by a malicious user to compromise your system security.

Server proxy account

- ☐ **Enable server proxy account**

  Enables an account for use by **xp_cmdshell**. Proxy accounts allow for the impersonation of logins, server roles, and database roles when an operating system command is being executed.

Proxy account

Specify the proxy account used.

Password

Specify the password for the proxy account.

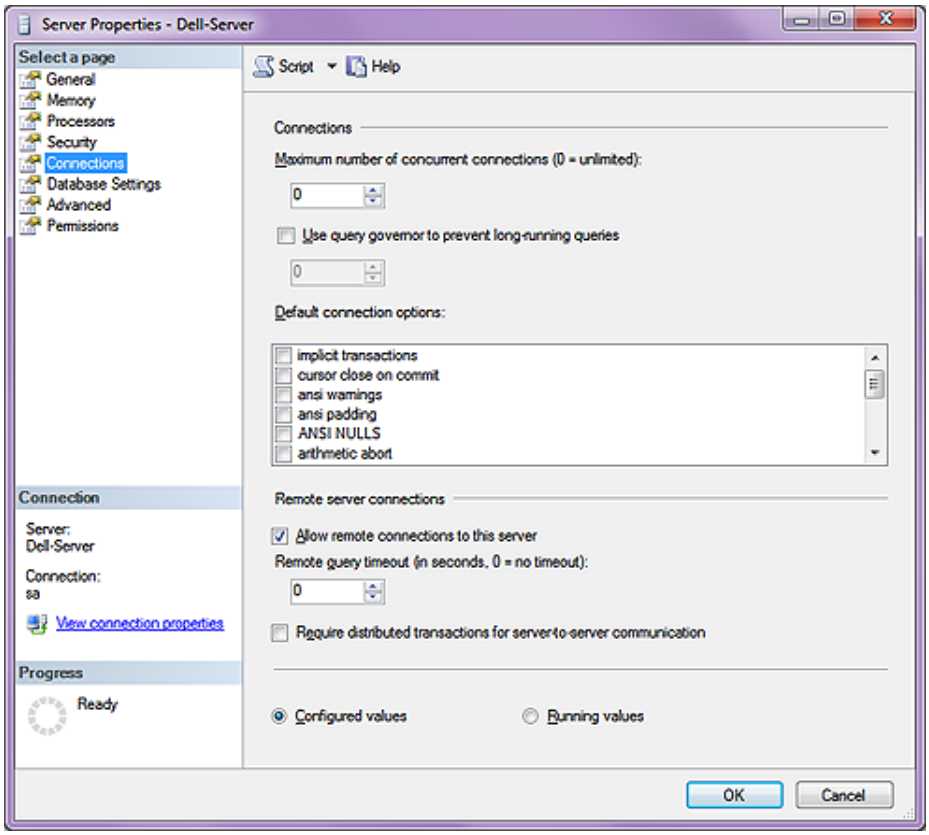

Figure 13-5

Options

☐ **Enable C2 audit tracing**

Audits all attempts to access statements and objects and records them to a file in the \MSSQL\Data directory for default instances of SQL Server, or the \MSSQL$instancename\Data directory for named instances of SQL Server. For more information, see c2 audit mode Option.

☐ **Cross database ownership chaining**

Select to allow the database to be the source or target of a cross-database ownership chain. For more information, see cross db ownership chaining Option.

Climbing on down the tree to the next branch and we come upon the **Connections** branch of the tree in Figure 13-5 on the previous page. You will notice that there are quite a few things that you can do on this dialog form. Looking again at the SQL Server | Help application for support, we find:

Connections

Maximum number of concurrent connections (0 = unlimited)

If set to a value other than zero, limits the number of connections that Microsoft SQL Server will allow.

WARNING

*Setting this to a small value, such as **1** or **2**, can prevent administrators from connecting to administer the server; however, the Dedicated Admin Connection can always connect.*

☐ **Use query governor to prevent long-running queries**

There is nothing in the SQL Help file on this option, however it appears to be self-explanatory. When selected this option defaults to **300**.

Default Connection Options

Specifies the default connection options, as described in the following table.

Configuration Option	Description
disable deferred constraint checking	Controls interim or deferred constraint checking.
implicit transactions	Controls whether a transaction is started implicitly when a statement is run.
cursor close on commit	Controls behavior of cursors after a commit operation has been performed.
ansi warnings	Controls truncation and NULL in aggregate warnings.
ansi padding	Controls padding of fixed-length variables.
ansi nulls	Controls NULL handling when using equality operators.
arithmetic abort	Terminates a query when an overflow or divide-by-zero error occurs during query execution.
arithmetic ignore	Returns NULL when an overflow or divide-by-zero error occurs during a query.
quoted identifier	Differentiates between single and double quotation marks when evaluating an expression.
no count	Turns off the message returned at the end of each statement that states how many rows were affected.
ansi null default on	Alters the session's behavior to use ANSI compatibility for nullability. New columns defined without explicit nullability are defined to allow nulls.
ansi null default off	Alters the session's behavior not to use ANSI compatibility for nullability. New columns defined without explicit nullability are defined not to allow nulls.
concat null yields null	Returns NULL when concatenating a NULL value with a string.
numeric round abort	Generates an error when a loss of precision occurs in an expression.
xact abort	Rolls back a transaction if a Transact-SQL statement raises a runtime error.

For more information on connection options, search Books Online for the specific option.

Remote Server Connections

☐ **Allow remote connections to this server**

Controls the execution of stored procedures from remote servers running instances of SQL Server. Selecting this check box has the same effect as setting the sp_configure remote access option to 1. Clearing it prevents execution of stored procedures from a remote server.

Remote query timeout (in seconds, 0 = no timeout)

Specifies how long (in seconds) a remote operation may take before SQL Server times out. The default is 600 seconds, or a 10-minute wait.

Require distributed transactions for server-to-server communication

Protects the actions of a server-to-server procedure through a Microsoft Distributed Transaction Coordinator (MS DTC) transaction. For more information, see remote proc trans Option.

Configured Values

Displays the configured values for the options on this pane. If you change these values, click Running Values to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

Running Values

View the currently running values for the options on this pane. These values are read-only.

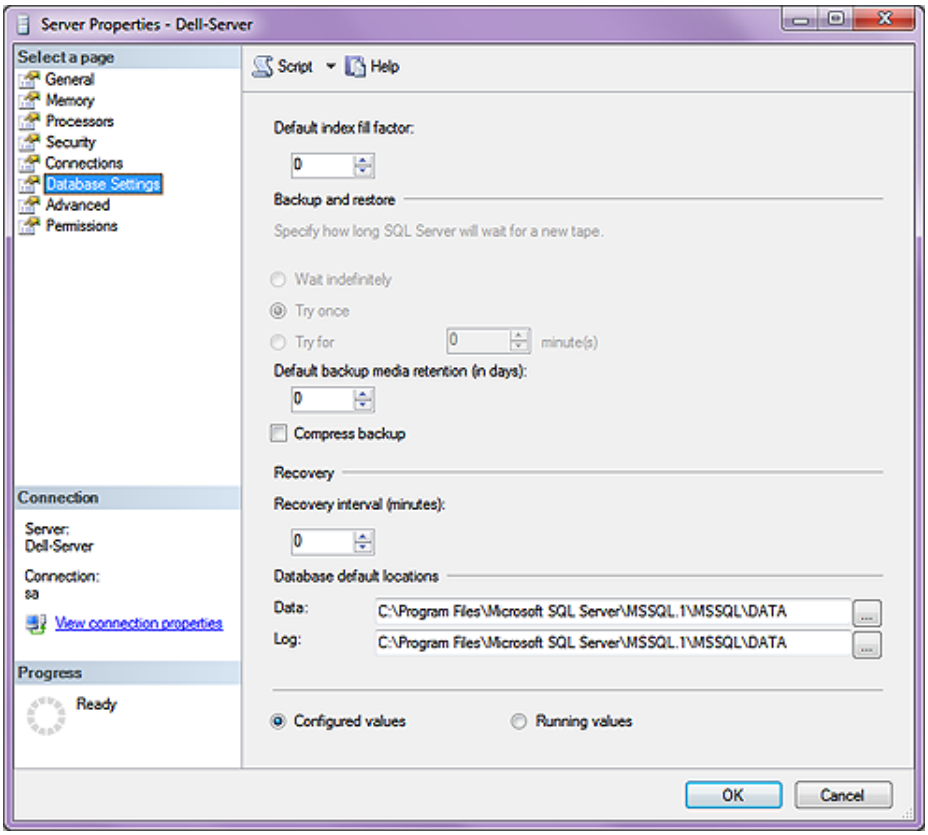

Figure 13-6

As you will see here in Figure 13-6, we have again moved down the tree to the next new branch known as **Database Settings**. Again, I will accept the comments offered by the SQL Server | Help application.

Default index fill factor

Specifies how full SQL Server should make each page when it creates a new index using existing data. The fill factor affects performance because SQL Server must take time to split pages when they fill up.

The default value is 0; valid values range from 0 through 100. A fill factor of 0 or 100 creates clustered indexes with full data pages and nonclustered indexes with full leaf pages, but it leaves some space within the upper level of the index tree. Fill factor values 0 and 100 are identical in all respects.

Small fill factor values cause SQL Server to create indexes with pages that are not full. Each index takes more storage space, but there is more room for subsequent insertions without requiring page splits.

Backup and Restore

○ **Wait indefinitely**

Specifies that SQL Server will never time out while waiting for a new backup tape.

◉ **Try once**

Specifies that SQL Server will time out if a backup tape is not available when needed.

○ **Try for minute(s)**

Specifies that SQL Server will time out if a backup tape is not available within the period specified.

Default backup media retention (in days)

Provides a system-wide default for the length of time to retain each backup medium after it has been used for a database or transaction log backup. This option helps protect backups from being overwritten until the specified number of days has elapsed.

☐ **Compress backup**

In SQL Server 2008 Enterprise (or later versions), indicates the current setting of the **backup compression default** option. This option determines the server-level default for compressing backups, as follows:

- If the ☐ **Compress backup** box is blank, new backups are uncompressed by default.
- If the ☑ **Compress backup** box is checked, new backups are compressed by default.

If you are a member of the **sysadmin** or **serveradmin** fixed server role, you can change the setting by clicking the Compress backup box

Recovery

Recovery interval (minutes)

Sets the maximum number of minutes per database to recover databases. The default is 0, indicating automatic configuration by SQL Server. In practice, this means a recovery time of less than one minute and a checkpoint approximately every one minute for active databases. For more information, see recovery interval Option.

Database default locations

Data

Specifies the default location for data files. Click the browse button to navigate to a new default location.

Log

Specifies the default location for log files. Click the browse button to navigate to a new default location.

Note

If you backup to another hard drive, as I do, you may want to consider setting this option to:

○ ***Try for x minute(s)***

WARNING

By default, compression significantly increases CPU usage, and the additional CPU consumed by the compression process might adversely affect concurrent operations. Therefore, you might want to create low-priority compressed backups in a session whose CPU usage is limited by Resource Governor. For more information, see How to: Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL).

Configured Values

Displays the configured values for the options on this pane. If you change these values, click Running Values to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restated first.

Running Values

View the currently running values for the options on this pane. These values are read-only.

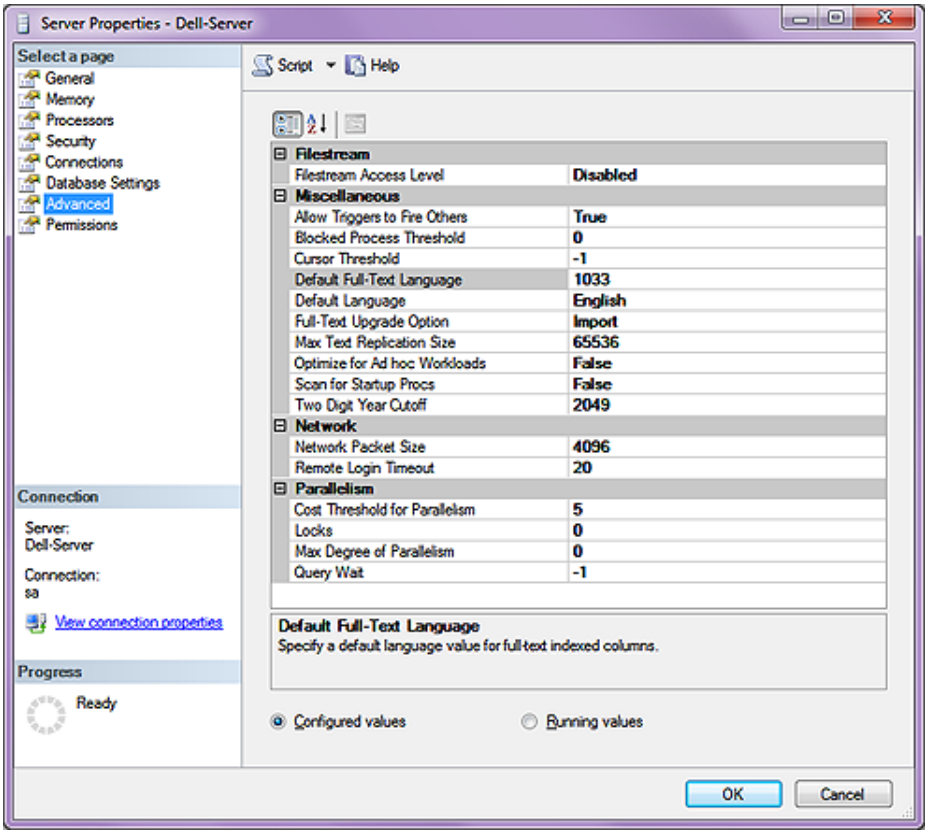

Figure 13-7

Let's climb on down to the **Advanced** branch to see options we have available to us out on this branch. Here is what the SQL Server | Help application has to say about this branch of the dialog form:

Note

When you enable FILESTREAM for the first time, you might have to restart the computer to configure drivers.

Filestream

Filestream Access Level

Shows the current level of FILESTREAM support on the instance of SQL Server. To change the access level, select one of the following values:

Disabled

Binary large object (BLOB) data cannot be stored on the file system. This is the default value.

Transact-SQL Only

FILESTREAM data is accessible by using Transact-SQL, but not through the file system.

Transact-SQL and file system (local client access only)

FILESTREAM data is accessible by using Transact-SQL and through the file system.

Miscellaneous

Allow Triggers to Fire Others

Allows triggers to fire other triggers. Triggers can be nested to a maximum of 32 levels. For more information, see the "Nested Triggers" section in CREATE TRIGGER (Transact-SQL).

Block Process Threshold

The threshold, in seconds, at which blocked process reports are generated. The threshold can be set from 0 to 86,400. By default, no blocked process reports are produced. For more information, see blocked process threshold Option.

Cursor Threshold

Specifies the number of rows in the cursor set at which cursor keysets are generated asynchronously. When cursors generate a keyset for a result set, the query optimizer estimates the number of rows that will be returned for that result set. If the query optimizer estimates that the number of returned rows is greater than this threshold, the cursor is generated asynchronously, allowing the user to fetch rows from the cursor while the cursor continues to be populated. Otherwise, the cursor is generated synchronously, and the query waits until all rows are returned.

If set to -1, all keysets are generated synchronously; this benefits small cursor sets. If set to 0, all cursor keysets are generated asynchronously. With other values, the query optimizer compares the number of expected rows in the cursor set and builds the keyset asynchronously if it exceeds the number set. For more information, see cursor threshold Option.

Default Full Text Language

Specifies a default language for full-text indexed columns. Linguistic analysis of full-text indexed data is dependent on the language of the data. The default value of this option is the language of the server. For the language that corresponds to the displayed setting, see sys.fulltext_languages (Transact-SQL).

Default Language

The default language for all new logins, unless otherwise specified.

Note

The full-text upgrade option can also be set by using the sp_fulltext_service upgrade_option action.

Full-Text Upgrade Option

Controls how full-text indexes are migrated when upgrading a database from SQL Server 2000 or SQL Server 2005 to SQL Server 2008 or later version. This property applies to upgrading by attaching a database, restoring a database backup, restoring a file backup, or copying the database by using the Copy Database Wizard.

The alternatives are as follows:

Import

Full-text catalogs are imported. This operation is significantly faster than **Rebuild**. However, an imported full-text catalog does not use the new and enhanced word breakers that are introduced in SQL Server 2008. Therefore, you might want to rebuild your full-text catalogs eventually.

If a full-text catalog is not available, the associated full-text indexes are rebuilt. This option is available for only SQL Server 2005 databases.

Rebuild

Full-text catalogs are rebuilt using the new and enhanced word breakers. Rebuilding indexes can take awhile, and a significant amount of CPU and memory might be required after the upgrade.

Reset

Full-text catalogs are reset. SQL Server 2005 full-text catalog files are removed, but the metadata for full-text catalogs and full-text indexes is retained. After being

upgraded, all full-text indexes are disabled for change tracking and crawls are not started automatically. The catalog will remain empty until you manually issue a full population, after the upgrade completes.

For information about how to choose the full-text upgrade option, see Full-Text Search Upgrade.

After you attach, restore, or copy a SQL Server 2005 or SQL Server 2000 database to SQL Server 2008, the database becomes available immediately and is then automatically upgraded. If the database has full-text indexes, the upgrade process either imports, resets, or rebuilds them, depending on the setting of the **Full-Text Upgrade Option** server property. If the upgrade option is set to **Import** or **Rebuild**, the full-text indexes will be unavailable during the upgrade. Depending on the amount of data being indexed, importing can take several hours, and rebuilding can take up to ten times longer. Note also that when the upgrade option is set to **Import**, if a full-text catalog is not available, the associated full-text indexes are rebuilt. For information about viewing or changing the setting of the **Full-Text Upgrade Option** property, see How to: View or Change Server Properties for Full-Text Search (SQL Server Management Studio).

Max Text Replication Size

Specifies the maximum size (in bytes) of text, ntext, varchar(max), nvarchar(max), xml, and image data that can be added to a replicated column or captured column in a single INSERT, UPDATE, WRITETEXT, or UPDATETEXT statement. Changing the setting takes effect immediately. For more information, see max text repl size Option.

Open Objects

Specifies the maximum number of database objects that can be open at one time on an instance of Microsoft SQL Server. Only available for SQL Server 2000.

Scan For Startup Procs

Specifies that SQL Server will scan for automatic execution of stored procedures at startup. If set to True, SQL Server scans for and runs all automatically run stored procedures defined on the server. If set to False (the default), no scan is performed. For more information, see scan for startup procs Option.

Two Digit Year Cutoff

Indicates the highest year number that can be entered as a two-digit year. The year listed and the previous 99 years can be entered as a two-digit year. All other years must be entered as a four-digit year.

For example, the default setting of 2049 indicates that a date entered as '3/14/49' will be interpreted as March 14, 2049, and a date entered as '3/14/50' will be interpreted as March 14, 1950.

Network

Network Packet Size

Sets the packet size (in bytes) used across the whole network. The default packet size is 4096 bytes. If an application does bulk-copy operations or sends or receives large amounts of **text** or **image** data, a packet size larger than the default may improve efficiency, because it results in fewer network reads and writes. If an application sends and receives small amounts of information, you can set the packet size to 512 bytes, which is sufficient for most data transfers. For more information, see network packet size Option.

Note

Do not change the packet size unless you are certain that it will improve performance. For most applications, the default packet size is best.

Remote Login Timeout

Specifies the number of seconds SQL Server waits before returning from a failed remote login attempt. This setting affects connections to OLE DB providers made for heterogeneous queries. The default value is 20 seconds. A value of 0 allows for an infinite wait. For more information, see remote login timeout Option.

Changing the setting takes effect immediately.

Parallelism

Cost Threshold for Parallelism

Specifies the threshold above which SQL Server creates and runs parallel plans for queries. The cost refers to an estimated elapsed time in seconds required to run the serial plan on a specific hardware configuration. Only set this option on symmetric multiprocessors. For more information, see cost threshold for parallelism Option.

Locks

Sets the maximum number of available locks, thereby limiting the amount of memory SQL Server uses for them. The default setting is 0, which allows SQL Server to allocate and deallocate locks dynamically based on changing system requirements.

Allowing SQL Server to use locks dynamically is the recommended configuration. For more information, see locks Option.

Max Degree of Parallelism

Limits the number of processors (up to a maximum of 64) to use in parallel plan execution. The default value of 0 uses all available processors. A value of 1 suppresses parallel plan generation. A number greater than 1 restricts the maximum number of processors used by a single query execution. If a value greater than the number of available processors is specified, the actual number of available processors is used. For more information, see max degree of parallelism Option.

Query Wait

Specifies the time in seconds (from 0 through 2147483647) that a query waits for resources before timing out. If the default value of -1 is used, the time-out is calculated as 25 times of the estimated query cost. For more information, see query wait Option.

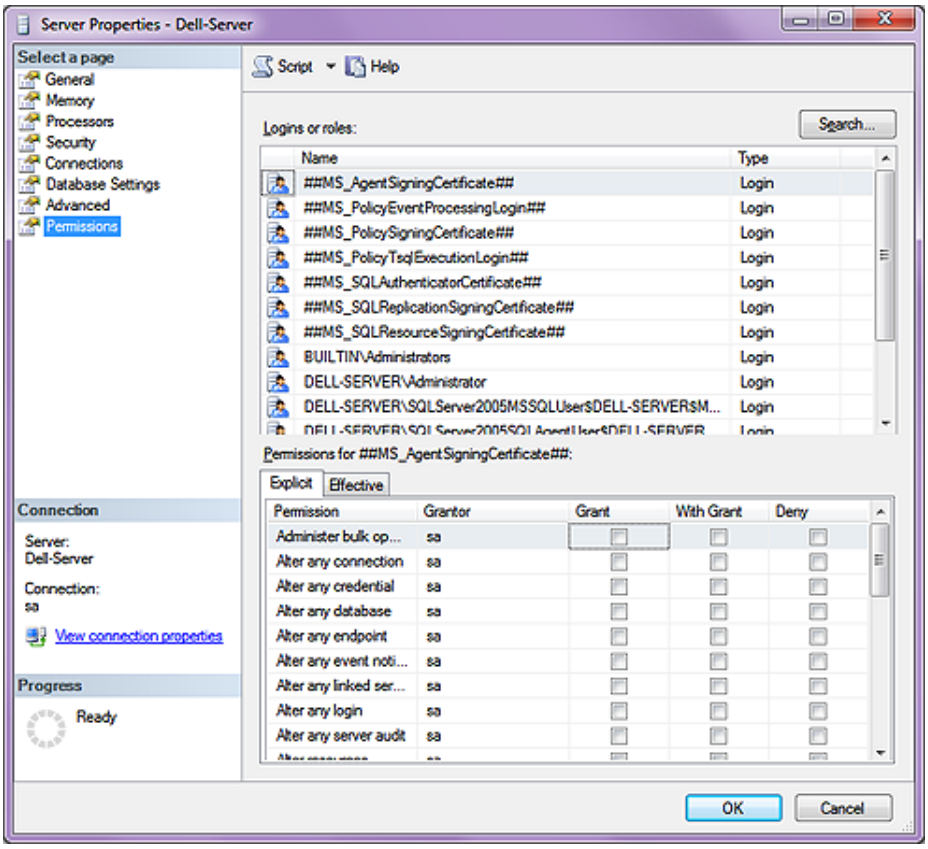

Figure 13-8

Finally we move on to the last branch of the tree, the **Permissions** branch. There is very little that I can say about this branch, but again I will give you the Help application information as displayed in SQL Server.

Use the **Permissions** page or the **Securables** page to view or set the permissions for securables. This page can be opened from many locations. The contents of the page can change slightly, depending on how the page is opened and what it contains. The top grid of the page might be populated when the page opens, or it might be empty. To add items to the upper grid, click **Search**. In the upper grid, select an item, and then set the appropriate permissions on the **Explicit** tab. To view aggregated permissions, use the **Effective** tab.

To understand the possible combinations of securables and principals, see the securable-specific syntax links in the topic GRANT (Transact-SQL). Other topics that you can refer to help you understand permissions are: Permissions Hierarchy (Database Engine), Securables, and Permissions (Database Engine).

Page Header

The header of the **Permissions** or **Securables** page varies depending on the securable or principal. It displays information relevant to the item, such as its name

Upper Grid

The upper grid contains one or more items for which permissions can be set. This dialog box provides the **Search** button for selecting objects or principals to add to the upper grid. The name of the grid might display Securables or one or more types of securables or principals. The columns that are displayed in the upper grid vary depending on the principal or securable.

Name

The name of each principal or securable that is added to the grid.

Type

Describes the type of each item.

Explicit Tab

The Explicit tab lists the possible permissions for the securable that are selected in the upper grid. To configure the permissions, select or clear the **Grant** (or **Allow**), **With Grant**, and **Deny** check boxes. All options are not available for all explicit permissions.

Permissions

The name of the permission.

Grantor

The principal that granted the permission.

Grant

Select to grant this permission to the login. Clear to revoke this permission.

With Grant

Reflects the state of the WITH GRANT option for the listed permission. This box is read-only. To apply this permission, use the GRANT statement.

Deny

Select to deny this permission to the login. Clear to revoke this permission.

Column Permissions

For objects that contain columns (such as tables, views, or table-valued functions), the **Column Permissions** button opens the **Column Permissions** dialog box. In this dialog box, you can set **Grant**, **Allow**, or **Deny** permissions on individual columns of a table or view. This option is not available for all object types or permissions.

Effective Tab

The permissions that a principal has related to a securable may come from permissions that are set for several different principals. For example, a login might be granted permissions individually and

also as a member of a group. The **Effective** tab shows the result of combining explicit permissions and the permissions that are received from group or role memberships. Grant permissions are aggregated. A deny permission overrides all grant permissions.

Permissions

The name of the permission.

Column

The names of columns that are affected by the permission.

Database Properties

And that pretty much concludes the Server Properties for SQL Server 2008. I would next like to continue on with the GoldMine SQL Database Properties. For most of the settings in here we will accept the default properties, however, there are a couple which we have chosen to tweak. Please right-click on a GoldMine database, and select **Properties** from the local menu to bring up the dialog form displayed here in Figure 13-9.

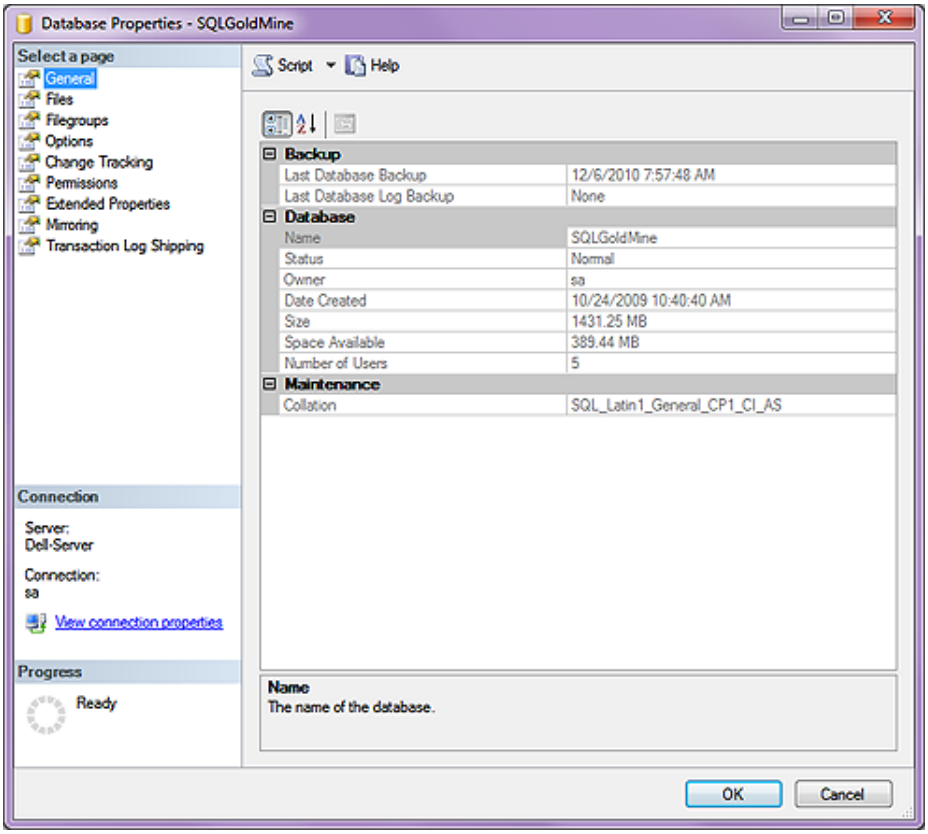

Figure 13-9

I will again work through the various tree branches in combination with the SQL Server | Help application information. Where appropriate, I will make my annotations in the sidebar. Please remember to read all of the notes contained in the sidebar as they may be very important to your SQL Server configuration. All of the information contained in the following screenshots is contained in the same live database that I have been using over the last 10 years, **SQLGoldMine**. If it works for me, it will probably work for you as well.

Beginning at the top of the tree, and working our way down the first branch that we encounter is again the **General** branch. As you can see, the dialog form is disabled (greyed out), and none of the information can be edited on this dialog page. However, here is an explanation of what these options represent:

Backup

Last Database Backup

Displays the date that the database was last backed up.

Last Database Log Backup

Displays the date that the database transaction log was last backed up.

Database

Name

Displays the name of the database.

Status

Displays the database state. For more information, see Database States.

Owner

Displays the name of the database owner. The owner can be changed on the Files page.

Date Created

Displays the date and time that the database was created.

Size

Displays the size of the database in megabytes.

Space Available

Displays the amount of available space in the database in megabytes.

Number of Users

Displays the number of users connected to the database.

Maintenance

Collation Name

Displays the collation used for the database. The collation can be changed on the Options page.

Climbing on down the tree, we next come to the **Files** branch of this tree, see Figure 13-10 on the next page. Although this dialog form does not contain that much information, it is saying much about the database.

Database name

Add or display the name of the database, and in it's default state is disabled (greyed out).

Owner

Specify the owner of the database by selecting from the list.

☑ **Use full-text indexing**

This check box is checked and disabled because full-text indexing is always enabled in SQL Server 2008. For more information, see Full-Text Search (SQL Server).

Database files

Add, view, modify, or remove database files for the associated database. Database files have the following properties:

Logical Name

Enter or modify the name of the file.

File Type

Select the file type from the list. The file type can be **Data**, **Log**, or **Filestream Data**. You cannot modify the file type of an existing file.

Note

Even though my personal database shows the **Owner:** as **sa**, many GoldMine Administrators will opt to change this to a generic like **GMUser** that has database ownership rights.

In fact, Computerese Inc has a corporate standard where we always create a **System Administrator** login of **GMUser** while assigning that user a corporate standard password as we are virtually assured that the client will forget their sa login.

Note

You'll notice that ☑ **Use full-text indexing** is selected in my configuration as we utilize the GoldMine Universal Search capabilities.

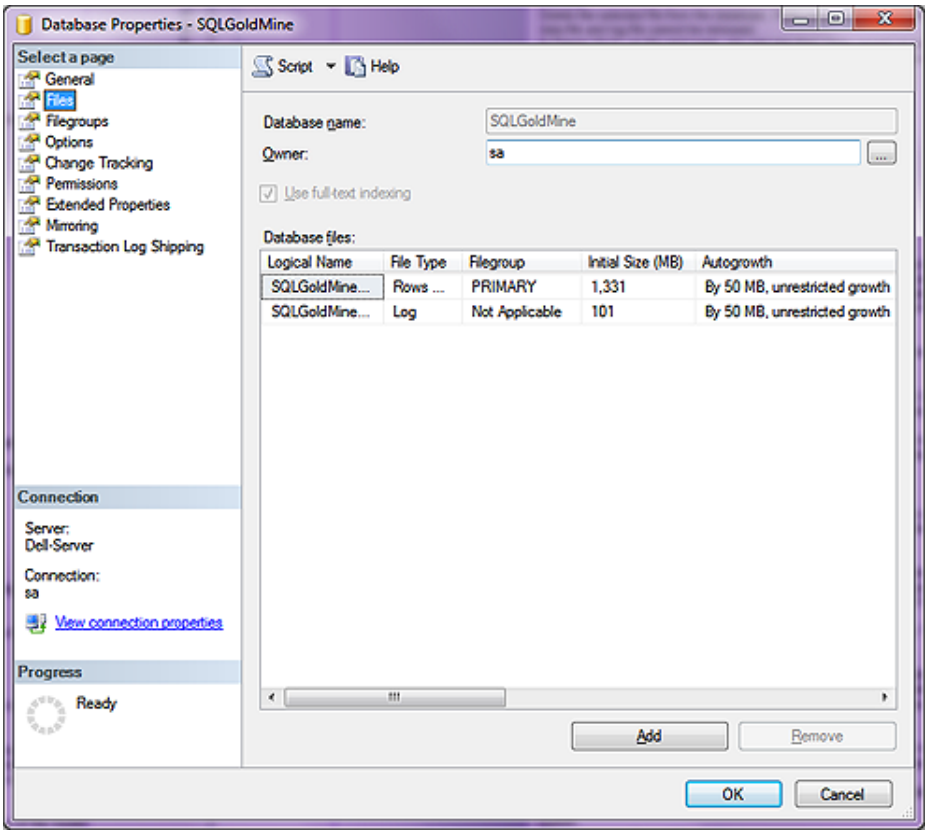

Figure 13-10

The FILESTREAM option will not appear if FILESTREAM is not enabled. You can enable FILESTREAM by using the Server Properties (Advanced Page) dialog box.

Filegroup

Select the filegroup for the file from the list. By default, the filegroup is PRIMARY. You can create a new filegroup by selecting **<new filegroup>** and entering information about the filegroup in the **New Filegroup** dialog box. A new filegroup can also be created on the **Filegroup** page. You cannot modify the filegroup of an existing file.

Initial Size

Enter or modify the initial size for the file in megabytes. By default, this is the value of the **model** database.

This field is not valid for FILESTREAM files.

Note

You'll notice that **Autogrowth** is set to **By 50 MB, unrestricted growth** on my setup.

Autogrowth

Select or display the autogrowth properties for the file. These properties control how the file expands when its maximum file size is reached. To edit autogrowth values, click the edit button next to the autogrowth properties for the file that you want, and change the values in the **Change Autogrowth** dialog box. By default, these are the values of the **model** database.

This field is not valid for FILESTREAM files.

Note

You cannot modify the path of an existing file.

Path

Displays the path of the selected file. To specify a path for a new file, click the edit button next to the path for the file, and navigate to the destination folder. You cannot modify the path of an existing file.

For FILESTREAM files, the path is a folder. The SQL Server Database Engine will create the underlying files in this folder.

File Name

Displays the name of the file.

This field is not valid for FILESTREAM files.

Add

Add a new file to the database.

Remove

Delete the selected file from the database. A file cannot be removed unless it is empty. The primary data file and log file cannot be removed.

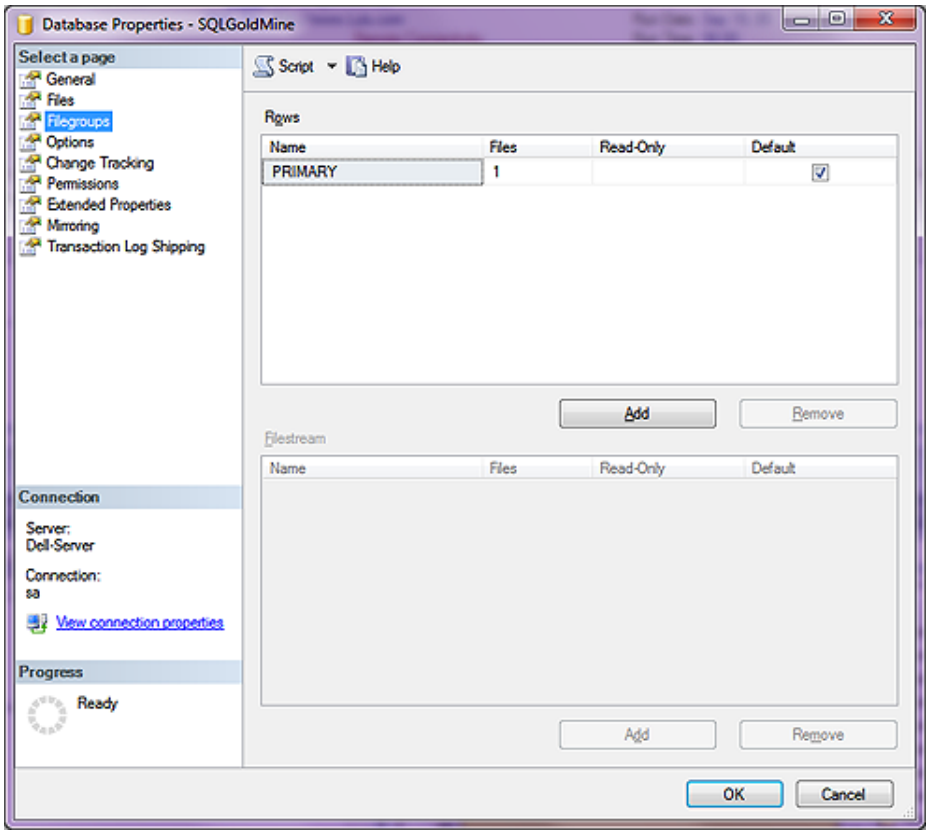

Figure 13-11

Here in Figure 13-11, you can see that I have moved on to the **Filegroups** branch of the tree, and that there is very little that can be said about this page of the dialog form. Here is what the Help application has to say about this page:

Rows

Name

Enter the name of the filegroup.

Files

Displays the count of files in the filegroup.

Read-only

Select to set the filegroup to a read-only status.

Default

Select to make this filegroup the default filegroup. You can have one default filegroup for rows and one default filegroup for FILESTREAM data.

Add

Adds a new blank row to the grid listing filegroups for the database.

Remove

Removes the selected filegroup row from the grid.

And that is all that can really be said about the **Filegroups** branch of the tree. So, let's move on to the **Options** branch of the tree, referred to here in Figure 13-12.

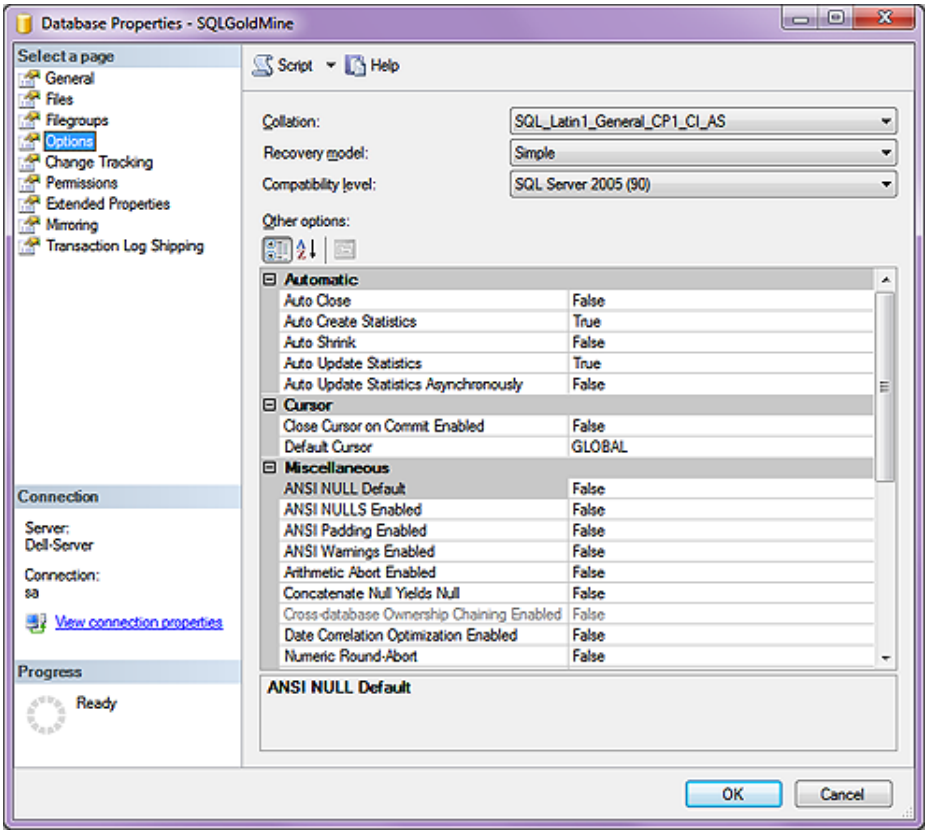

Figure 13-12

I guarantee you, that there is a lot more to be said for **Options**. Please pay attention to the sidebar notes as I have made changes to the SQL Server defaults in this area. Again, from the SQL Server | Help application we find the following information:

Collation

Specify the collation of the database by selecting from the list. For more information, see Working with Collations.

Recovery model

Specify one of the following models for recovering the database: **Full**, **Bulk-Logged**, or **Simple**. For more information about recovery models, see Recovery Model Overview.

Compatibility level

Specify the latest version of SQL Server that the database supports. Possible values are **SQL Server 2008**, **SQL Server 2005**, and **SQL Server 2000**. For more information, see ALTER DATABASE Compatibility Level (Transact-SQL).

Note

As GoldMine Premium maintains its own TLogs, we have no need for the SQL version of these logs, and we always opt for the **Simple** recovery model.

Note

If you are utilizing any 3rd party applications with GoldMine, you will probably want to leave the **Auto Close** option at **False**.

Note

You will notice that I have chosen to have the **Auto Create Statistics** to be **True**. The default would have been False, and in BDE days would have been required to have been False.

Note

Setting the **Auto Update Statistics Asynchronously** option to **True** has no effect unless **Auto Update Statistics** is also set to **True**.

Automatic

Auto Close

Specify whether the database shuts down cleanly and frees resources after the last user exits. Possible values are **True** and **False**. When **True**, the database is shut down cleanly and its resources are freed after the last user logs off.

Auto Create Statistics

Specify whether the database automatically creates missing optimization statistics. Possible values are **True** and **False**. When **True**, any missing statistics needed by a query for optimization are automatically built during optimization. For more information, see CREATE STATISTICS (Transact-SQL).

Auto Shrink

Specify whether the database files are available for periodic shrinking. Possible values are **True** and **False**. For more information, see Shrinking a Database.

Auto Update Statistics

Specify whether the database automatically updates out-of-date optimization statistics. Possible values are **True** and **False**. When **True**, any out-of-date statistics needed by a query for optimization are automatically built during optimization. For more information, see CREATE STATISTICS (Transact-SQL).

Auto Update Statistics Asynchronously

When **True**, queries that initiate an automatic update of out-of-date statistics will not wait for the statistics to be updated before compiling. Subsequent queries will use the updated statistics when they are available.

When **False**, queries that initiate an automatic update of out-of-date statistics, wait until the updated statistics can be used in the query optimization plan.

Cursor

Close Cursor on Commit Enabled

Specify whether cursors close after the transaction opening the cursor has committed. Possible values are **True** and **False**. When **True**, any cursors that are open when a transaction is committed or rolled back are closed. When **False**, such cursors remain open when a transaction is committed. When **False**, rolling back a transaction closes any cursors except those defined as INSENSITIVE or STATIC. For more information, see SET CURSOR_CLOSE_ON_COMMIT (Transact-SQL).

Default Cursor

Specify default cursor behavior. When **True**, cursor declarations default to LOCAL. When **False**, Transact-SQL cursors default to GLOBAL. For more information, see Scope of Transact-SQL Cursor Names.

Miscellaneous

ANSI NULL Default

Specify the default behavior of the Equals (=) and Not Equal To (<>) comparison operators when used with null values. Possible values are **True** (on) and **False** (off). For more information, see SET ANSI_NULL_DFLT_ON (Transact-SQL) and SET ANSI_NULL_DFLT_OFF (Transact-SQL).

ANSI NULLS Enabled

Specify the behavior of the Equals (=) and Not Equal To (<>) comparison operators when used with null values. Possible values are **True** (on) and **False** (off). When **True**, all comparisons to a null value evaluate to UNKNOWN. When **False**, comparisons of non-UNICODE values to a null value evaluate to **True** if both values are NULL. For more information, see SET ANSI_NULLS (Transact-SQL).

ANSI Padding Enabled

Specify whether ANSI padding is on or off. Permissible values are **True** (on) and **False** (off). For more information, see SET ANSI_PADDING (Transact-SQL).

ANSI Warnings Enabled

Specify ISO standard behavior for several error conditions. When **True**, a warning message is generated if null values appear in aggregate functions (such as SUM, AVG, MAX, MIN, STDEV, STDEVP, VAR, VARP, or COUNT). When **False**, no warning is issued. For more information, see SET ANSI_WARNINGS (Transact-SQL).

Arithmetic Abort Enabled

Specify whether the database option for arithmetic abort is enabled or not. Possible values are **True** and **False**. When **True**, an overflow or divide-by-zero error causes the query or batch to terminate. If the error occurs in a transaction, the transaction is rolled back. When **False**, a warning message is displayed, but the query, batch, or transaction continues as if no error occurred. For more information, see SET ARITHABORT (Transact-SQL).

Concatenate Null Yields Null

Specify the behavior when null values are concatenated. When the property value is **True**, **string** + NULL returns NULL. When **False**, the result is **string**. For more information, see SET CONCAT_NULL_YIELDS_NULL (Transact-SQL).

Cross-database Ownership Chaining Enabled

This read-only value indicates if cross-database ownership chaining has been enabled. When **True**, the database can be the source or target of a cross-database ownership chain. Use the ALTER DATABASE statement to set this property.

Date Correlation Optimization Enabled

When **True**, SQL Server maintains correlation statistics between any two tables in the database that are linked by a FOREIGN KEY constraint and have **datetime** columns.

When **False**, correlation statistics are not maintained. For more information, see Optimizing Queries That Access Correlated datetime Columns.

Numeric Round-Abort

Specify how the database handles rounding errors. Possible values are **True** and **False**. When **True**, an error is generated when loss of precision occurs in an expression. When **False**, losses of precision do not generate error messages, and the result is rounded to the precision of the column or variable storing the result. For more information, see SET NUMERIC_ROUNDABORT (Transact-SQL).

Parameterization

When **SIMPLE**, queries are parameterized based on the default behavior of the database. When **FORCED**, SQL Server parameterizes all queries in the database. For more information, see Simple Parameterization and Forced Parameterization.

Quoted Identifiers Enabled

Specify whether SQL Server keywords can be used as identifiers (an object or variable name) if enclosed in quotation marks. Possible values are **True** and **False**. For more information, see SET QUOTED_IDENTIFIER (Transact-SQL).

Recursive Triggers Enabled

Specify whether triggers can be fired by other triggers. Possible values are **True** and **False**. When set to **True**, this enables recursive firing of triggers. When set to **False**, only direct recursion is prevented. To disable indirect recursion, set the **nested triggers** server option to 0 using **sp_configure**. For more information, see Using Nested Triggers.

Trustworthy

When displaying **True**, this read-only option indicates that SQL Server allows access to resources outside the database under an impersonation context established within the database. Impersonation contexts can be established within the database using the EXECUTE AS user statement or the EXECUTE AS clause on database modules.

To have access, the owner of the database also needs to have the AUTHENTICATE SERVER permission at the server level.

This property also allows the creation and execution of unsafe and external access assemblies within the database. In addition to setting this property to **True**, the owner of the database must have the EXTERNAL ACCESS ASSEMBLY or UNSAFE ASSEMBLY permission at the server level.

By default, all user databases and all system databases (with the exception of **MSDB**) have this property set to **False**. The value cannot be changed for the **model** and **tempdb** databases.

TRUSTWORTHY is set to **False** whenever a database is attached to the server.

The recommended approach for accessing resources outside the database under an impersonation context is to use certificates and signatures as apposed to the **Trustworthy** option.

To set this property, use the ALTER DATABASE statement.

VarDecimal Storage Format Enabled

This option is read-only starting with SQL Server 2008. When **True**, this database is enabled for the vardecimal storage format. Vardecimal storage format cannot be disabled while any tables in the database are using it. In SQL Server 2008, all databases are enabled for the vardecimal storage format. For information about the vardecimal storage format, see Storing Decimal Data As Variable Length. This option uses sp_db_vardecimal_storage_format.

Recovery

Page Verify

Specify the option used to discover and report incomplete I/O transactions caused by disk I/O errors. Possible values are **None**, **TornPageDetection**, and **Checksum**. For more information, see Understanding and Managing the suspect_pages Table.

State

Database Read Only

Specify whether the database is read only. Possible values are **True** and **False**. When **True**, users can only read data in the database. Users cannot modify the data or database objects; however, the database itself can be deleted using the DROP DATABASE statement. The database cannot be in use when a new value for the **Database Read Only** option is specified. The **master** database is the exception, and only the system administrator can use **master** while the option is being set.

Restrict Access

Specify which users may access the database. Possible values are:

- **Multiple**

  The normal state for a production database, allows multiple users to access the database at once.

- **Single**

  Used for maintenance actions, only one user is allowed to access the database at once.

- **Restricted**

  Only members of the **db_owner**, **dbcreator**, or **sysadmin** roles can use the database.

Encryption Enabled

When **True**, this database is enabled for database encryption. A Database Encryption Key is required for encryption. For more information, see Understanding Transparent Data Encryption (TDE).

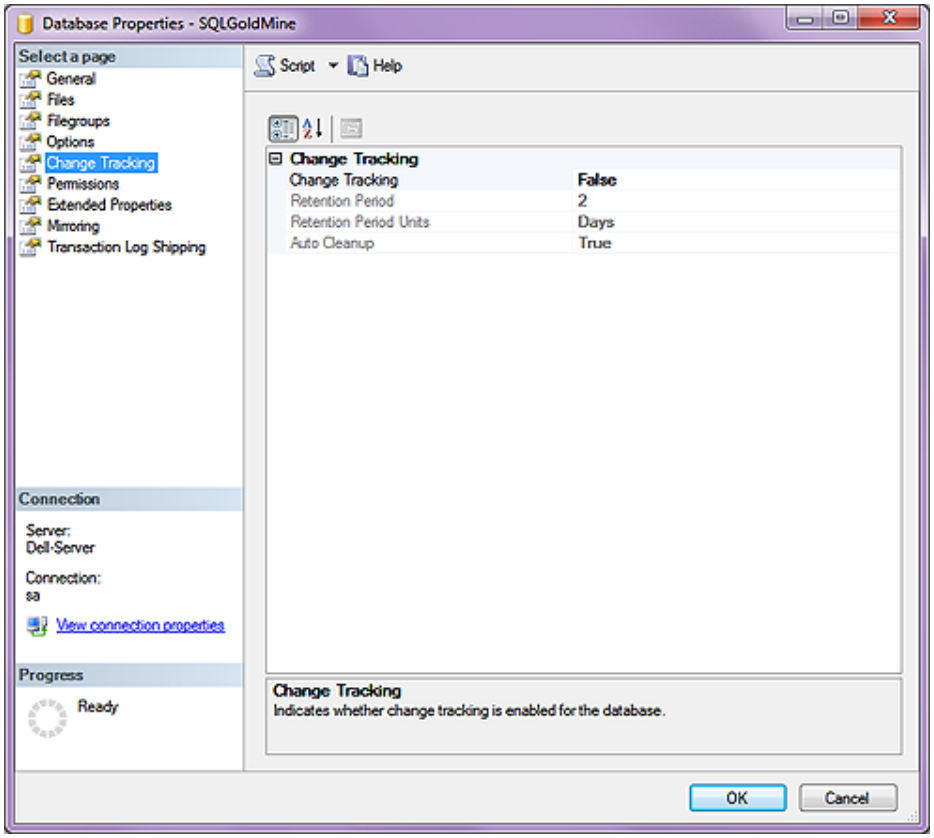

Figure 13-13

New in SQL Server 2008 is the **Change Tracking** branch with its four options.

Change Tracking

Change Tracking

Use to enable or disable change tracking for the database.

To enable change tracking, you must have permission to modify the database.

Setting the value to **True** sets a database option that allows change tracking to be enabled on individual tables.

You can also configure change tracking by using ALTER DATABASE.

Retention Period

Specifies the minimum period for keeping change track information in the database. Data is removed only if the **Auto Clean-Up** value is **True**.

The default value is 2.

Retention Period Units

Specifies the units for the Retention Period value. You can select **Days**, **Hours**, or **Minutes**. The default value is **Days**.

The minimum retention period is 1 minute. There is no maximum retention period.

Auto Cleanup

Indicates whether change tracking information is automatically removed after the specified retention period.

Enabling **Auto Clean-Up** resets any previous custom retention period to the default retention period of 2 days.

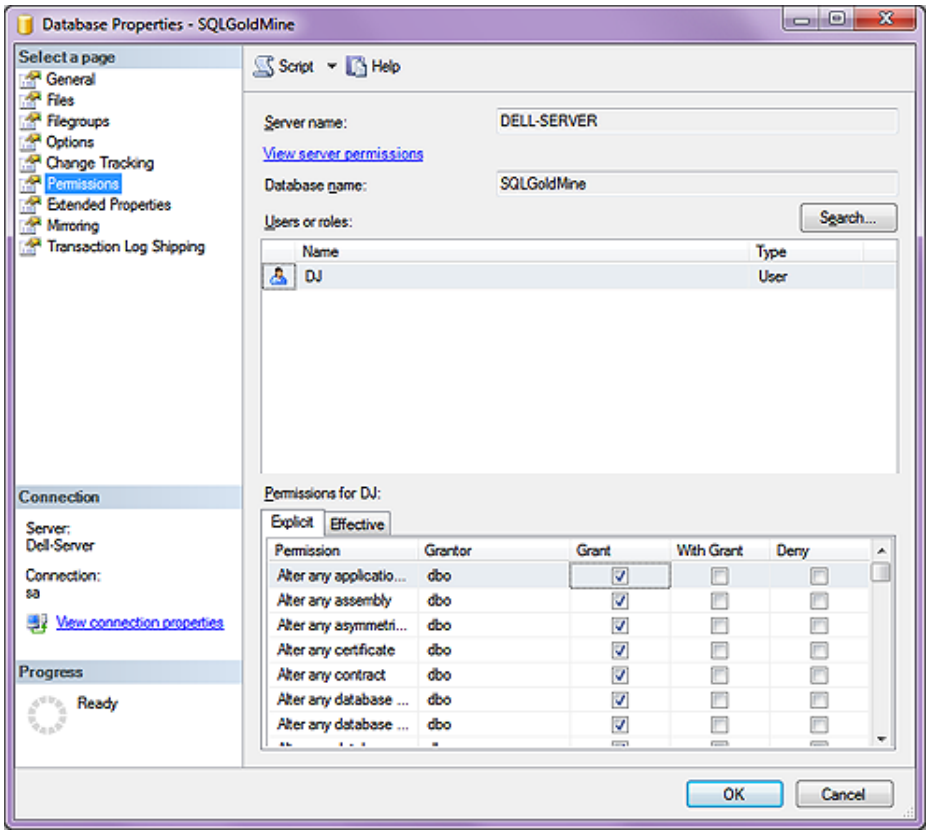

Figure 13-14

The **Permissions** branch of the tree is pretty much the same as we had discussed previously, and the screenshot of this dialog form can be seen above in Figure 13-14. There is very little that I can add to this section, so I will just take from the SQL Server | Help application:

Server name

Non modifiable field displaying the database Server name.

Database name

Non modifiable field displaying the Database name.

Users or roles

The users or roles grid contains one or more items for which permissions can be set. This dialog box provides the **Search** button for selecting objects or principals to add to the upper grid. The name of the grid might display **Securables** or one or more types of securables or principals. The columns that are displayed in the users or roles grid vary depending on the principal or securable.

Explicit Permissions

The **Explicit** tab lists the possible permissions for the securable that are selected in the upper grid. To configure the permissions, select or clear the **Grant** (or **Allow**), **With Grant**, and **Deny** check boxes. All options are not available for all explicit permissions.

Permissions

The name of the permission.

Grantor

The principal that granted the permission.

Grant

Select to grant this permission to the login. Clear to revoke this permission.

With Grant

Reflects the state of the WITH GRANT option for the listed permission. This box is read-only. To apply this permission, use the GRANT statement.

Deny

Select to deny this permission to the login. Clear to revoke this permission.

Column Permissions

For objects that contain columns (such as tables, views, or table-valued functions), the **Column Permissions** button opens the **Column Permissions** dialog box. In this dialog box, you can set **Grant**, **Allow**, or **Deny** permissions on individual columns of a table or view. This option is not available for all object types or permissions.

Effective Permissions

The permissions that a principal has related to a securable may come from permissions that are set for several different principals. For example, a login might be granted permissions individually and also as a member of a group. The **Effective** tab shows the result of combining explicit permissions and the permissions that are received from group or role memberships. Grant permissions are aggregated. A deny permission overrides all grant permissions.

Permissions

The name of the permission.

Column

The names of columns that are affected by the permission.

I won't be covering the lower three branches of the tree, **Extending Properties**, **Mirroring** & **Transaction Log Shipping** as these are pretty much read-only, however, you are welcome to delve into these on your own using the SQL Server | Help application to guide you.

SQL Server Maintenance Plan for GoldMine

Over the years people have asked me for documentation on an accepted SQL Server Maintenance Plan. To date, I have been unable to find one that was written for the GoldMine application/SQL Database. FrontRange utilizes SQL Server 2008 for Workgroups in that it distributes this with GoldMine Premium v9.0.0.110, yet to date, FrontRange will not support the SQL Server application/installation/configuration.

Computerese Incorporated has implemented many SQL Server Maintenance Plans over the years, and although we know how we do it, we really don't know if this is the correct ***FrontRange Approved*** way of doing this. I am writing this as a way of documenting the Computerese Incorporated SQL Server Maintenance Plan implementation process. As always, if you utilize these steps to implement your own SQL Server Maintenance Plan for your GoldMine Database(s), you do so at you own risk knowing full well that these steps have not received the stamp of approval from FrontRange nor any organization for that matter. Although, since it's original printing in **The GoldMine Advisor** - **August** issue, I have heard from many GoldMine Partners who have approved this process with a few exceptions which I will discuss at the appropriate time.

Step 01

Open SQL Server Management Studio, and **Connect** to your SQL Server.

Step 02

Expand the tree **+Management** by clicking upon the **+** sign.

Step 03

Right-click on the **+Maintenance Plans** and select **Maintenance Plan Wizard** from the local menu. Even though we know the steps involved, we always use the Wizard to assure a consistent Maintenance Plan execution.

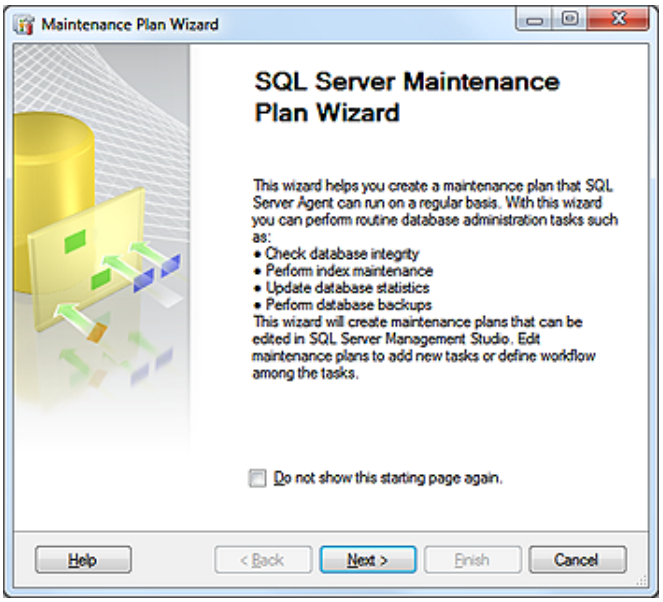

Figure 13-15

Doing so will produce the dialog form shown here in Figure 13-15. I have chosen to not check the ☐ **Do not show this starting page again.** option as I wanted you to see the default screens, however, you could select this option as this splash screen is really superfluous.

Step 04

Clicking on the **Next >** button will bring you to the **Select Plan Properties** dialog form shown here.

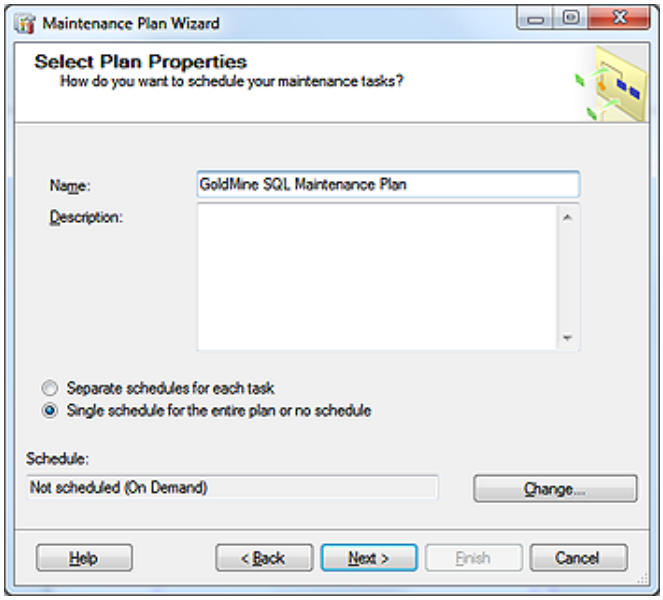

Figure 13-16

It is here that one should supply a unique **Name** for their plan. For this example I have chosen **GoldMine SQL Maintenance Plan.** Now is also the time to decide on the frequency of execution of this plan once created. One does this by clicking in the **Schedule:** area of the dialog form on the **Change** button to produce this dialog form to bring up the dialog form shown on the next page in Figure 13-17.

Now, again, this is the default screenshot, however, you must decide how to configure this based on your office backup procedures. The SQL Maintenance Plan, which can include its own Backup Plan, should not be confused with your Office Backup Plan. Redundant, yes, but how important is your GoldMine Database to your office? So important, I would imagine, that one should welcome redundancy.

Regardless, the Computerese Inc **Frequency** is **Daily**, in fact, we set up two of these plans with the first running at 12:15 pm Eastern Time, and the second running at 7:00 pm Eastern Time. We don't mind redundancy at all, and we don't plan on losing any more than 7 hours worth of work. We have clients that have requested a plan run hourly. The decision should be strictly between you and your IT department. If you plan on just nightly, make sure it executes before your Office Backup Plan so the files that you will be creating will get backed up there as well.

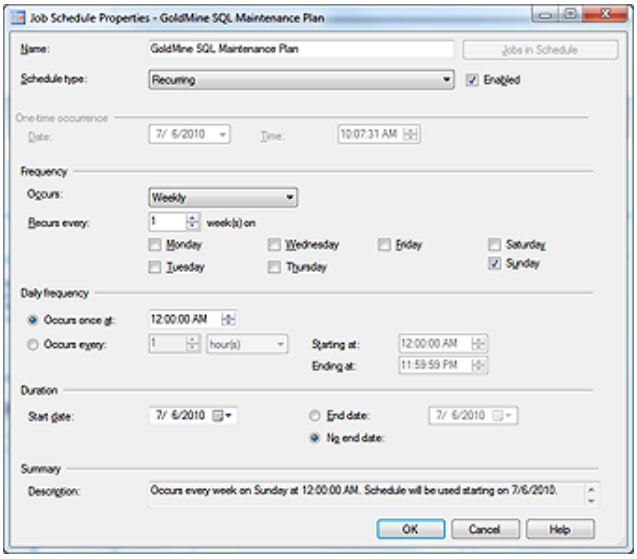

Figure 13-17

Everything else on this dialog form remains in its default state, and we would now click upon the **OK** button.

Step 05

Clicking on the **Next >** button will bring you to the **Select Maintenance Task** dialog form:

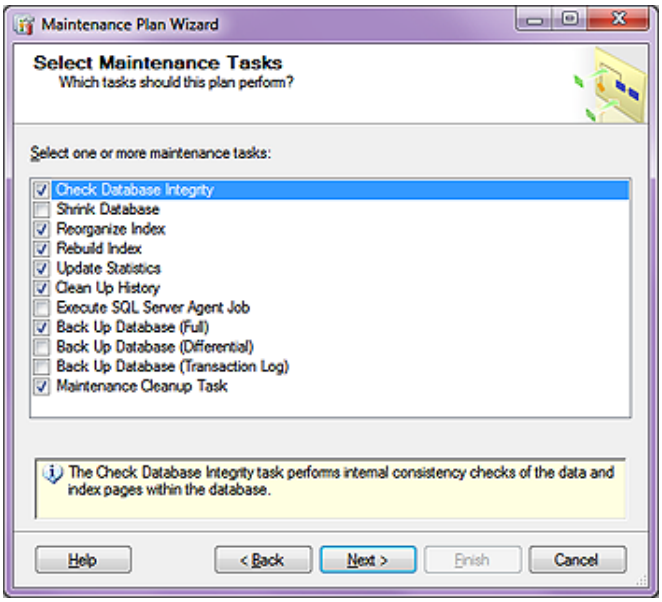

Figure 13-18

True, this is not the default state of this form, however, I did want to show you the selections that we make on this screen. I won't try to explain what each of these options represents, however, if that is of interest to you then you will find the SQL Server Help files to be most useful in this matter.

In the original article, and in the past for that matter, I had selected to ☐ **Shrink Database** as and option. Since then Jane Oliverio sent me a link that discusses the pros and the cons, actually more cons than pros, for doing this, and I have decided against shrinking my database. Here is the article reprinted from Paul S Randals' Blog in case this site goes dormant:

Note

*Readers of the original article in The GoldMine Advisor may remember that the **Shrink Database** option was selected in that article. Since that printing I have received many e-mails pointing me to sites which described why the **Shrink Database** was a bad idea. Once such article was written by one of the developers of the SQL Server application, hence, I am no longer recommending it be utilized. Said article is reprinted in this book on the next page for your review.*

Why you should not shrink your data files

by

Paul B. Randal

One of my biggest hot-buttons is around shrinking data files. Although I used to own the shrink code while I was at Microsoft, I never had a chance to rewrite it so that data file shrink is a more palatable operation. I really don't like shrink.

Now, don't confuse shrinking the transaction log with shrinking data files. Shrinking the log is necessary if your log has grown out of control, or as part of a process to remove excessive VLF fragmentation. However, shrinking the log should be a rare operation and should not be part of any regular maintenance you perform.

Shrinking of data files should be performed even more rarely, if at all. Here's why - data file shrink causes *massive* index fragmentation. Let me demonstrate with a simple script you can run. The script below will create a data file, create a 10MB 'filler' table at the start of the data file, create a 10MB 'production' clustered index, drop the 'filler' table and then run a shrink to reclaim the space.

```
USE MASTER;
GO

IF DATABASEPROPERTYEX ('DBMaint2008', 'Version') > 0
 DROP DATABASE DBMaint2008;

CREATE DATABASE DBMaint2008;
GO
USE DBMaint2008;
GO

SET NOCOUNT ON;
GO

-- Create the 10MB filler table at the 'front' of the data file
CREATE TABLE FillerTable (c1 INT IDENTITY,  c2 CHAR (8000) DEFAULT 'filler');
GO

-- Fill up the filler table
INSERT INTO FillerTable DEFAULT VALUES;
GO 1280

-- Create the production table, which will be 'after' the filler table in the data file
CREATE TABLE ProdTable (c1 INT IDENTITY,  c2 CHAR (8000) DEFAULT 'production');
CREATE CLUSTERED INDEX prod_cl ON ProdTable (c1);
GO

INSERT INTO ProdTable DEFAULT VALUES;
GO 1280

-- check the fragmentation of the production table
SELECT [avg_fragmentation_in_percent] FROM sys.dm_db_index_physical_stats (
   DB_ID ('DBMaint2008'), OBJECT_ID ('ProdTable'), 1, NULL, 'LIMITED');
GO

-- drop the filler table, creating 10MB of free space at the 'front' of the data file
DROP TABLE FillerTable;
GO

-- shrink the database
DBCC SHRINKDATABASE (DBMaint2008);
GO

-- check the index fragmentation again
SELECT [avg_fragmentation_in_percent] FROM sys.dm_db_index_physical_stats (
   DB_ID ('DBMaint2008'), OBJECT_ID ('ProdTable'), 1, NULL, 'LIMITED');
GO

avg_fragmentation_in_percent
----------------------------
0.390625
```

```
DbId   FileId      CurrentSize MinimumSize UsedPages   EstimatedPages
------ ----------- ----------- ----------- ----------- --------------
6      1           1456        152         1448        1440
6      2           63          63          56          56

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

avg_fragmentation_in_percent
-----------------------------
99.296875
```

Look at the output from the script! The logical fragmentation of the clustered index before the shrink is a near-perfect 0.4%. After the shrink, it's almost 100%. The shrink operation *completely* fragmented the index, removing any chance of efficient range scans on it by ensuring the all range-scan readahead I/Os will be single-page I/Os.

Why does this happen? A data file shrink operation works on a single file at a time, and uses the GAM bitmaps to find the highest page allocated in the file. It then moves it as far towards the front of the file as it can, and so on, and so on. In the case above, it completely reversed the order of the clustered index, taking it from perfectly defragmented to perfectly fragmented.

The same code is used for DBCC SHRINKFILE, DBCC SHRINKDATABASE, and auto-shrink - they're equally as bad. As well as introducing index fragmentation, data file shrink also generates a lot of I/O, uses a lot of CPU, and generates *loads* of transaction log - as everything it does is fully logged.

Data file shrink should never be part of regular maintenance, and you should NEVER, NEVER have auto-shrink enabled. I tried to have it removed from the product for SQL 2005 and SQL 2008 when I was in a position to do so - the only reason it's still there is for backwards compatibility. Don't fall into the trap of having a maintenance plan that rebuilds all indexes and then tries to reclaim the space required to rebuild the indexes by running a shrink - that's a zero-sum game where all you do is generate a log of transaction log for no actual gain in performance.

So what if you *do* need to run a shrink? For instance, if you've deleted a large proportion of a very large database and the database isn't likely to grow, or you need to empty a file before removing it?

The method I like to recommend is as follows:

- Create a new filegroup
- Move all affected tables and indexes into the new filegroup using the CREATE INDEX ... WITH (DROP_EXISTING) ON <filegroup> syntax, to move the tables and remove fragmentation from them at the same time
- Drop the old filegroup that you were going to shrink anyway (or shrink it way down if its the primary filegroup)

Basically you need to provision some more space before you can shrink the old files, but it's a much cleaner mechanism.

If you absolutely have no choice and have to run a data file shrink operation, be aware that you're going to cause index fragmentation and you should take steps to remove it afterwards if it's going to cause performance problems. The only way to remove index fragmentation without causing data file growth again is to use DBCC INDEXDEFRAG or ALTER INDEX ... REORGANIZE. These commands only require a single 8KB page of extra space, instead of needing to build a whole new index in the case of an index rebuild operation.

Bottom line - try to avoid running data file shrink at all costs!

Note

Normally, I would not touch the arrangement on this dialog form, however, Alberto Diaz of 180° Solutions has told me that he prefers to move the **Maintenance Cleanup Task** to run prior to the **Back Up Database (Full)** task. He found, in his tests, that the plan tends to fail less frequently when executed in that order. I must say that I have had a few instances where having the **Maintenance Cleanup Task** as the last task in the sequence order has caused a failure of the Maintenance Plan to produce a **Successful** log entry.

Your mileage may vary!

Step 06

Clicking on the **Next >** button will bring you to the **Select Maintenance Task Order** dialog form.

This dialog form permits you to change the order of execution of the steps which you had selected on the previous dialog form. Review the Notes before proceeding to Step 07.

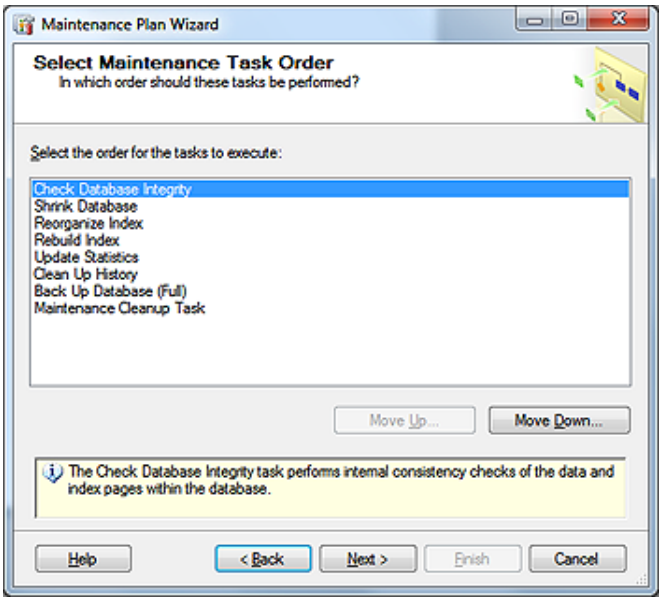

Figure 13-19

Step 07

Clicking on the **Next >** button will bring you to the default screen of the **Define Database Check Integrity Task** dialog form shown here in Figure 13-20.

However, in the default state, the **Database:** input field would have stated: **<Select one or more>**, and it would have been up to you to select one or more databases to include in this plan. As this is the same dialog form that is displayed on each of the successive dialog forms, I will show it to you but only this once in this article.

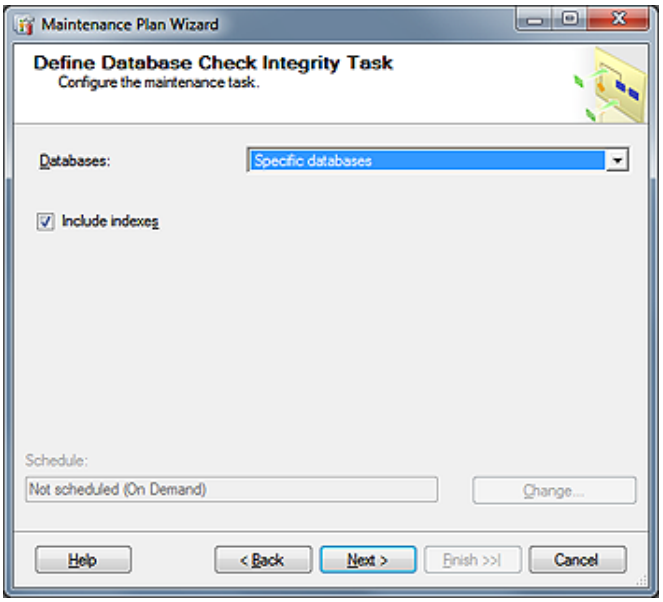

Figure 13-20

As you can see in Figure 13-21 which was instantiated by clicking on the down arrow at the end of the **<Select one or more>** option, the default setting is: ◉ **These databases:** where one would actually check the database(s) to include in this plan. As I mentioned previously, this setting **is not** sticky, and you will need to reselect the database(s) on each of the successive dialog forms in this Wizard where you come across the verbiage **Database: <Select one or more>**.

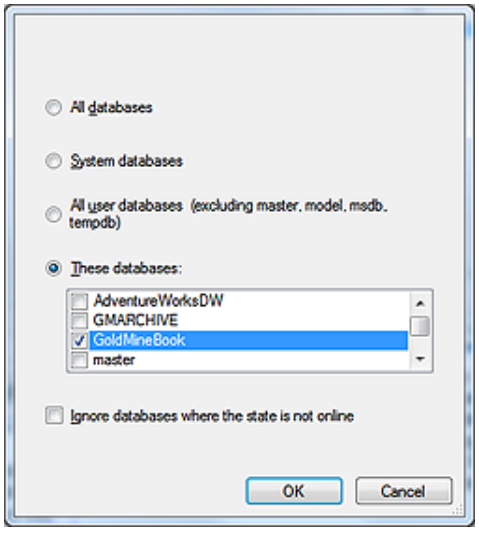

Figure 13-21

I have only selected a single **GoldMineBook** database, however, you could have selected all of your GoldMine databases if you utilize more than one (not recommended). One would next click on the **OK** button to return to the **Define Database Integrity Check Task** dialog form.

Step 08

Clicking on the **Next >** button will bring you to the **Define Reorganize Index Task** dialog form displayed in the next column.

Again, this is not the default dialog form as I have already selected **Specific databases**, although everything else on this dialog form remains in its default state.

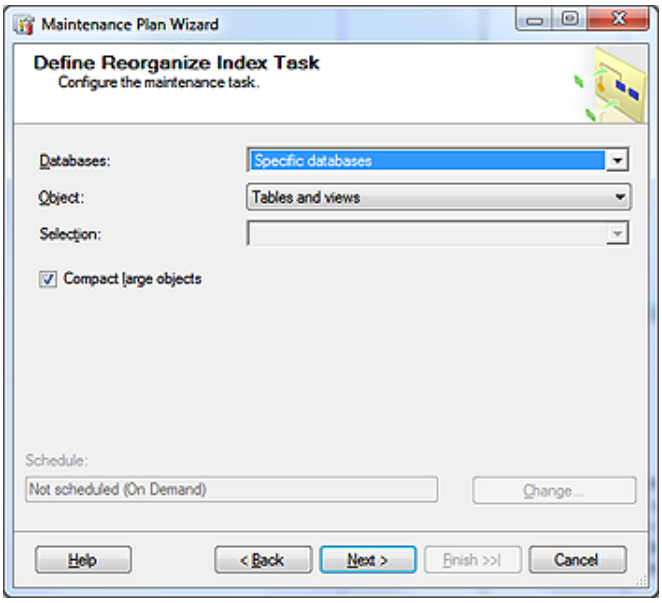

Figure 13-22

Step 09

Clicking on the **Next >** button will bring you to the **Define Rebuild Index Task** dialog form displayed here in Figure 13-23.

Again, this is not the default dialog form as I have already selected **Specific databases**, although everything else on this dialog form remains in its default state.

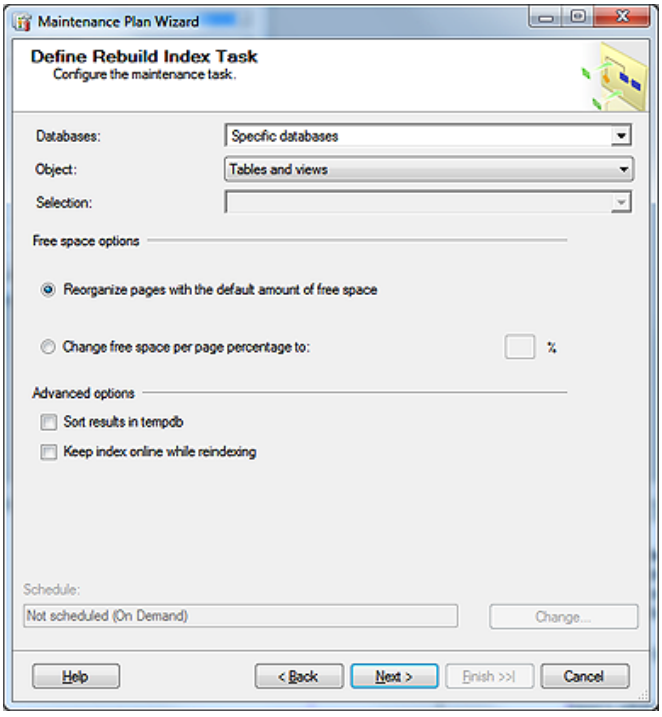

Figure 13-23

Step 10

Clicking on the **Next >** button will bring you to the **Define Update Statistic Task** dialog form.

Again, this is not the default dialog form as I have already selected **Specific databases**, although everything else on this dialog form remains in its default state.

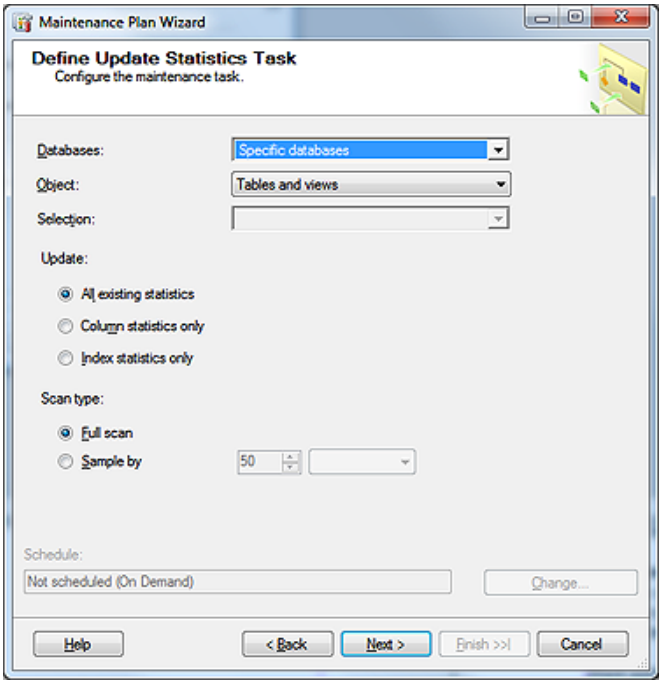

Figure 13-24

Step 11

Clicking on the **Next >** button will bring you to the **Define History Cleanup Task** dialog form show at the top of page 9. Everything on this dialog form remains in its default state.

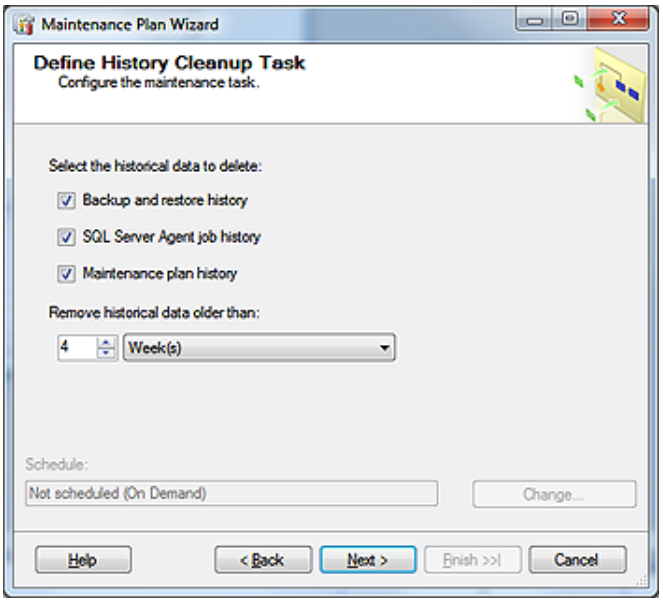

Figure 13-25

Step 12

Clicking on the **Next >** button will bring you to the **Define Back Up Database (Full) Task** dialog form displayed in the next column.

Again, this is not the default dialog form as I have already selected **Specific databases**. Unlike the other screenshots, I have made changes to the default settings on this dialog form.

For instance, I did select to ☑ **Create a sub-directory for each database**, and, as well, I did select to ☑ **Verify backup integrity**.

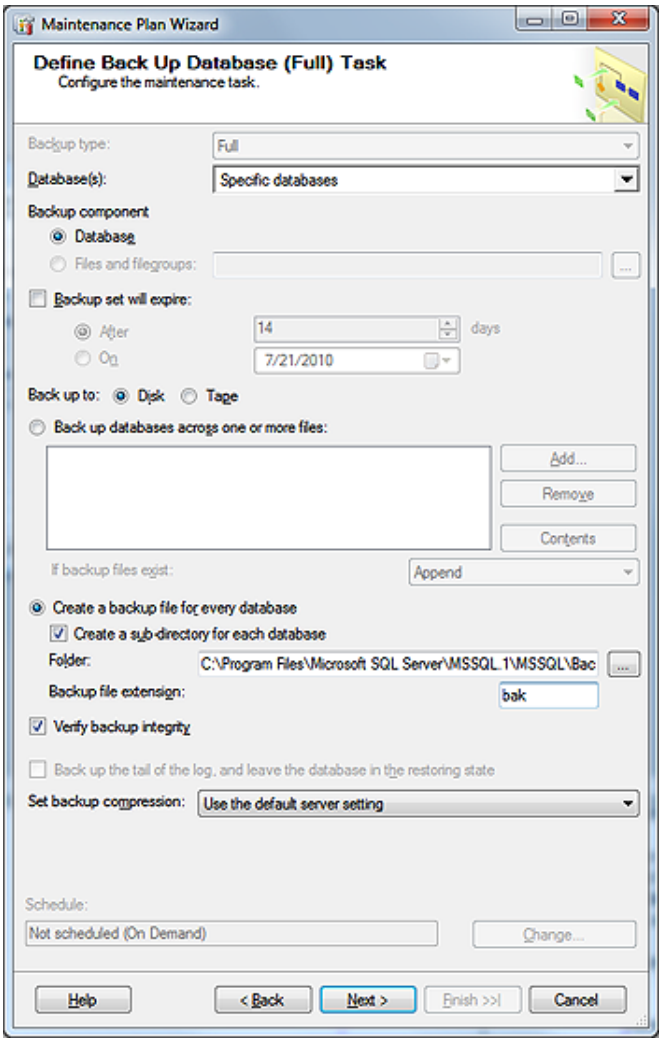

Figure 13-26

Step 13

Clicking on the **Next >** button will bring you to the **Define Maintenance Cleanup Task** dialog form shown here in Figure 13-27.

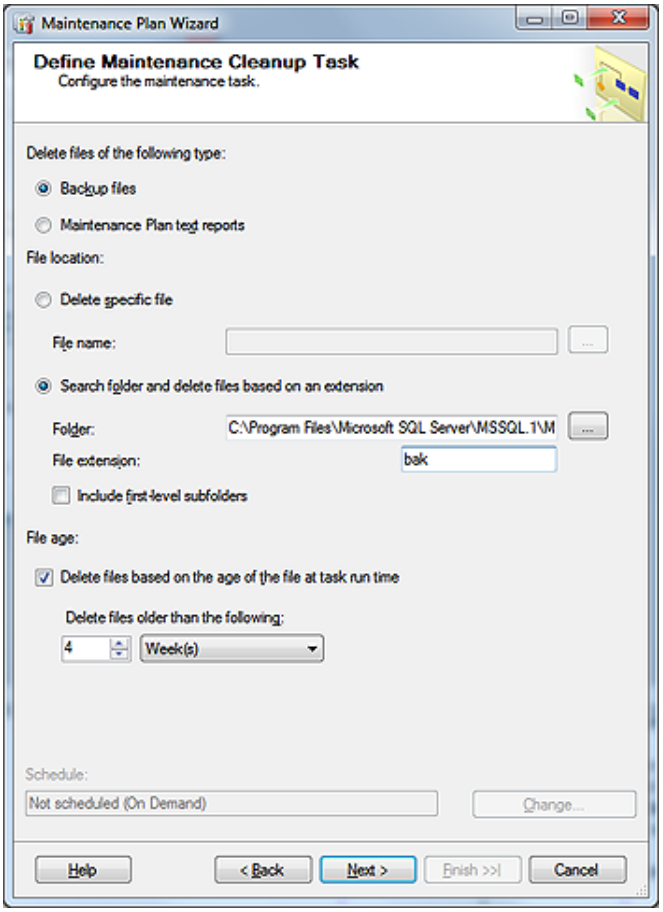

Figure 13-27

In the default state for this dialog form the ⊙ **Search folder and delete files based on an extension** is selected, however, it is up to you to supply the actual **Folder:** in which to search and the **File extension:** for which you wish to search.

In my particular case I used the ellipsis (**...**) button to browse to my designated backup folder and the selection entered:

C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup\SQLGoldMine

I also entered the file extension of **bak** which is the default extension as accepted on the above **Define Back Up Database (Full) Task**.

Step 14

Clicking on the **Next >** button will bring you to the **Select Report Options** dialog form as displayed in Figure 13-28.

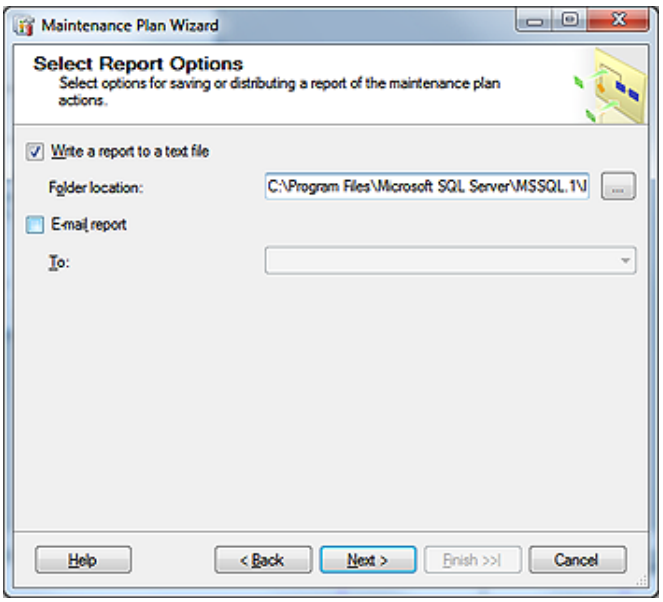

Figure 13-28

I usually just accept the defaults here, however, I have had instances where the clients had wanted to also have the report e-mailed to them. In those cases, I would have to select the ☐ **Email report** option while including an e-mail address in the **To:** field.

Step 15

Clicking on the **Next >** button will bring you to the **Complete the Wizard** dialog form

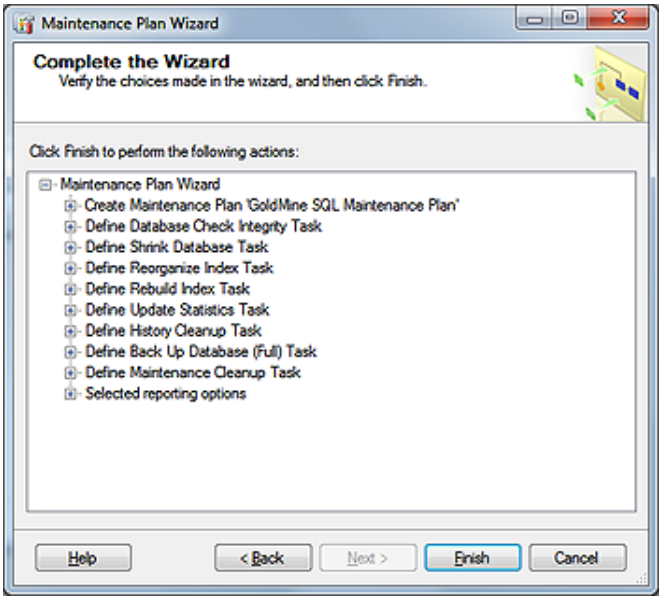

Figure 13-29

Finally, we have completed the Wizard, and we have little more to do than to click on the **Finish** button which will generate the **Maintenance Plan Wizard Progress** dialog form, refer to Figure 13-30. Hopefully, yours will also display the **Success** check mark. Clicking on the **Close** button, and you will have created your GoldMine SQL Maintenance Plan.

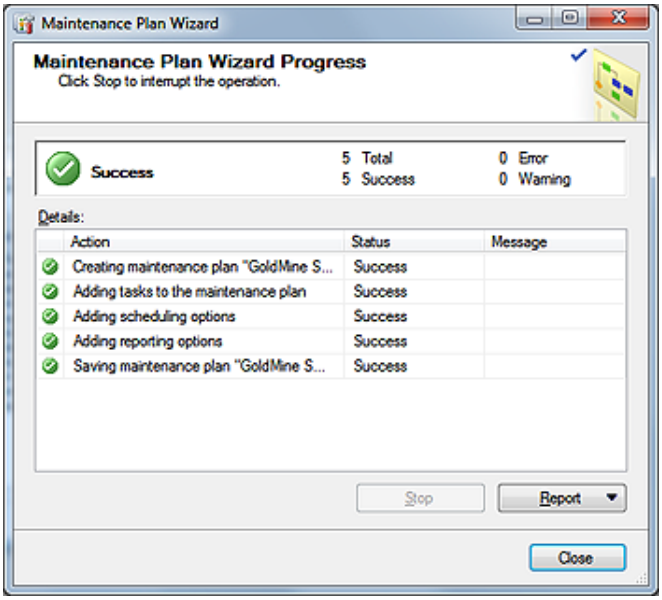

Figure 13-30

Step 16

3 days later you decide to check your **Job Activity Monitor**, and you notice that your GoldMine SQL Maintenance Plan has never run. You look in your SQL Backup folder and there are no backups to be found.

Why do I even bring this up, because just yesterday I had a client call me to tell me that the plan that I had created for them wasn't working, and hadn't worked since June 30th, 2010.

My first thought was to open SQL Server Management Studio to see if there was anything obviously wrong. Sure enough, there it was staring me right in the face. The **SQL Server Agent** had been **Stopped** as often happens when one stops the **SQL Service** itself. People remember to turn the **SQL Service** back on, but often forget that the **SQL Server Agent**, which is automatically stopped when the **SQL Service** is stopped, **must be Started**.

The dialog form shown in Figure 13-31 on the next page has the **SQL Server Agent** highlighted, and the icon to the left displays a green right facing arrow indicating that the service is **Started**. It is imperative that this service be started if you want any of your SQL Maintenance Plans to function at all.

Once I restarted the service for my client, the GoldMine Maintenance Plan ran Successfully again.

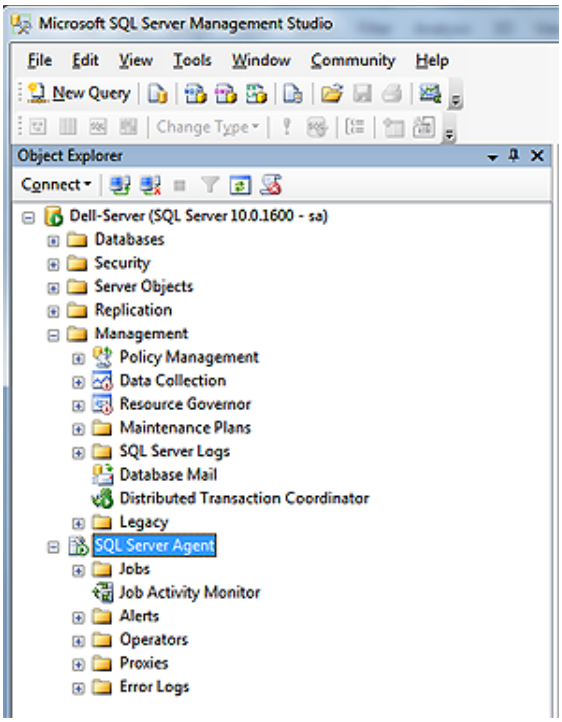

Figure 13-31

Conclusion

Caveat

Please remember that anything that you do within the **Microsoft SQL Management Studio** is **not** sync aware, hence, the transactions **will not** be propagated out to your Remote-side Undocked/Site License Users.

This means that any changes that you make through the Microsoft SQL Management Studio Server-side, will have to be reproduced by hand through the Microsoft SQL Management Studio on any of your Remote-side installations.

This pretty much wraps up this chapter for us, however, I would like to mention some of the powerful features that are accessible to you via the **Microsoft SQL Server Management Studio**.

You may **Open** any of your the tables within your database, and manipulate the data any way that you wish. You may even **Delete** records in this table once it has been opened.

You may script any table within your database using the **Script Table as** feature. This will allow you to build: **CREATE to**; **DROP to**; **SELECT to**; **INSERT to**; **UPDATE to** & **DELETE to** scripts which can be saved and utilized anywhere.

You can also **Add Fields**, **Add Indexes**, and do just about anything at all that you want within the Microsoft SQL Server Management Studio.