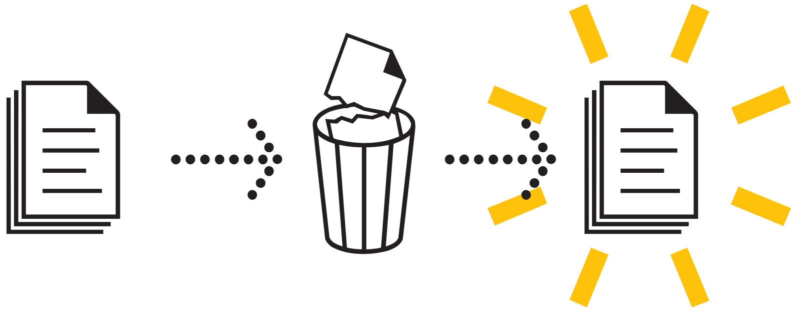
SQL Server

INSIDER

Tips for SQL Server pros March 2007

New backup and recovery *in SQL Server 2005*



Database administrators must have a reliable backup and recovery plan for their SQL Server databases. After five years in development, see how SQL Server 2005 provides greater backup and recovery options. Also, read about SQL Server 2005 performance tuning and upgrade hurdles.

INSIDE

Brought to you by SearchSQLServer.com

New backup and recovery in SQL Server 2005

By Hilary Cotter

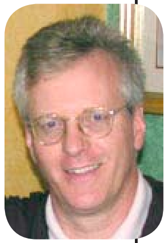
Database administrators are responsible for their companies' data, which is their most precious asset. But research shows that companies are extremely vulnerable when it comes to losing data. DBAs must take an active role in protecting data by implementing and testing reliable backup and recovery plans.

Microsoft has spent more than five years developing SQL Server 2005, and improving the backup and recovery has been one of the many goals of this release. This ezine article will explore the new backup and recovery features of SQL Server 2005.

VISUAL SOURCE SAFE INTEGRATION

SQL Server Management Studio is now completely integrated with Visual Source Safe or other version-control software — as long as it has a plug-in for SQL Server Management Studio. To access these features, go to File, Source Control in SQL Server Management Studio. To configure SQL Server Management Studio with another vendor's version-control software

HILARY COTTER has been involved in IT for more than 20 years as a Web and database consultant. Microsoft first awarded Cotter the Microsoft SQL Server MVP award in 2001. Cotter received his bachelor of applied science degree in mechanical engineering from the University of Toronto and studied economics at the University of Calgary and computer science at UC Berkeley. He is the author of a book on SQL Server transactional replication and is currently working on books on merge replication and Microsoft search technologies.



— such as MKS, PVCS or Subversion — go to Tools, Options and Source Control.

If SQL developers and DBAs are disciplined with their use of this feature, it should never be necessary to restore a large database simply to retrieve a schema-only object such as a stored procedure, function or view. Another feature that helps with source control is SQL Server Management Studio (SSMS), which performs document recovery. Should SSMS crash while T-SQL code is being edited, SSMS will automatically recover the code next time it's launched.

LOG BACKUP BEHAVIOR MODIFICATION

In previous versions of SQL Server, log backups were paused while a backup was being done. This caused two problems:

- In Full and Bulk Logged recovery models, the transaction logs, or tlogs, would continue to grow until the first log backup after the database backup was completed. Backing up a tlog will release the inactive portion of the log for reuse by SQL Server. This log maintenance keeps the tlog size manageable.
- The log can't be shipped until the backup is completed and the next tlog dump is performed. This increases the exposure to data loss and makes standby servers considerably out of sync with the primary or source server.

Although differential backups and third-party backup products like Quest's SQL LiteSpeed, Idera's SQL Safe, and RedGate's SQL Backup decrease the time required for a backup for large databases, the lack of a frequent log backup could be highly problematic.

In SQL 2005, log backups can occur while the full backup is occurring. This allows for more up-to-date log shipping — reducing your exposure to data loss — and better tlog size management.

COPY-ONLY BACKUP

In most environments, there is an occasional need to create a backup of a SQL Server database outside of the normal maintenance window. For example, developers may request a backup of the database at midday to troubleshoot a production issue. This has the potential to break recovery plans using differential backups.

Developers frequently use differential backups when databases become very large. A differential backup is a backup that contains only the database extents that have been modified since the last full backup. As differential backups are a backup of what has changed on the database since the last backup, the backup size will be smaller and the backup time will be less. It can also increase the restore times, as DBAs will have to restore a full backup and then the last differential backup.

Should an unplanned backup be performed, it will interrupt a backup plan that incorporated differential backups. In other words, the unplanned backup will have to form the base of a recovery effort for restoring subsequent differential backups. Copy-only backups will not interrupt log shipping and differential backup plans. DBAs can use copy-only backups on databases as well as tlogs. The ability to do a copy-only backup on a tlog is beneficial for log analyzer tools like Lumigent's

Watch the numbers

Microsoft reports that a large majority of its support calls are incidents that were preventable if database administrators had been following best practices. Consider the following research:

According to Milford, Mass.-based Enterprise Storage Group, now called the Enterprise Strategy Group, in its January 2004 report, "The Evolution of Enterprise Data Protection"

- **25% of 500 companies surveyed will face a significant data disruption event.**

Gartner Inc. in Stamford, Conn., said in its January 2002 report "Online Server Backup: Niche, Glitch or Killer App?" that

- **50% of companies would close within two years if their data is not restored within 24 hours after a disaster occurs.**
- **As many as 95% would close within two years if their data is not restored within 72 hours after a disaster.**
- **A whopping 77% of backups are unusable when they're needed.**

—Hilary Cotter

Log Explorer and ApexSQL Log and some auditing products.

MIRRORED TAPE BACKUPS

Another new feature in SQL Server 2005 is the ability to perform mirrored tape backups. DBAs can configure SQL Server 2005 to simultaneously back up a maximum of four tape drives without affecting backup performance. This increases the reliability of backing up to tape devices.

To use mirrored tape backups, identical tape units are needed. Backup devices typically allow DBAs to back up to multiple devices simultaneously, spread their backup over each device and achieve higher backup throughput and performance. With mirrored tape backups, however, there is no performance improvement and the backup is not spread over each tape unit.

"Backup devices typically allow DBAs to back up to multiple devices simultaneously, spread their backup over each device and achieve higher backup throughput and performance. Mirrored tape backups are the exception."

CHECKSUM

SQL Server 2000 and SQL Server 7 databases were enabled by default with a feature called "torn page detection." This feature was used to detect whether a page was incompletely written to disk — for example, in the event of a SQL Server error or, more commonly, in the event of a disk or other media failure.

SQL Server 2005 replaced that feature with a checksum feature that calculates the checksum on a page as it writes it to disk and then calculates the checksum on the page as it reads it from disk. If the values are different, there has been some page corruption on disk.

SQL Server 2005 replaces the "torn page detection," found in earlier versions, with the new checksum feature.

Checksums can also be used to verify the extents as they are written to the backup and to write a checksum in the backup file itself. When restoring a database, DBAs can verify that the checksum in the backup is the same as the checksum on the restored pages. Here is an example of how to do this:

```
BACKUP DATABASE 'test'
to disk='c:\test.bak' WITH
CheckSum
```

CONTINUE AFTER ERROR

It is possible to configure SQL Server 2005 to continue to back up and restore when there is a failure. This command is the **Continue_on_Error** parameter.

By default, the backup and restore commands use the **Stop_on_Error** parameter, which means that a backup or a restore will fail if SQL Server encounters an error during a backup or restore. Although this feature is not as useful during the backup commands, it can be really valuable during the restore commands.

Consider a case where the backup has been damaged or corrupted. Using the **Continue_on_Error** parameter will allow the restore to complete even if there has been some corruption.

Here is an example. You back up a database, and an event occurs that requires you to restore the database. Prior to doing that, you do a restore with verify only, and you get the following message:

```
Msg 3189, Level 16, State 1,
Line 1

Damage to the backup set
was detected.

Msg 3013, Level 16, State 1,
Line 1

VERIFY DATABASE is
terminating abnormally
```

An attempt to restore this database will fail with the following message:

```
Msg 3183, Level 16, State 1,
Line 1

Restore detected an error
on page (1:3453) in database
'AllYourBaseAreBelongToUs'
as read from the backup set.

Msg 3013, Level 16, State 1,
Line 1

RESTORE DATABASE is
terminating abnormally.
```

To restore this database, DBAs would incorporate the **Continue_on_error** parameter in the restore command as follows:

```
Restore Database
'AllYourBaseAreBelongToUs'
from disk='c:\test.bak' with
Continue_on_restore
```

In this case, the database will be restored and the error might be fixed using CHECKDB with the repair options. Prior to using the command, try to determine what is going to be repaired. An index can safely be repaired by a drop and rebuild. However, carefully evaluate whether any data loss should be allowed on tables. A page-level restore might also work to resolve the problem, but it may lead to data inconsistencies.

PAGE-LEVEL RESTORES

If, during a restart of SQL Server 2005 Enterprise Edition, a

"Prior to SQL Server 2005, many DBAs might spend their careers without knowing about the tail of the log. With SQL Server 2005, that has changed…"

checksum difference or torn page is detected, the database is marked as recovery pending — instead of the SQL Server 2000 state, which was suspect.

In other versions of SQL Server 2005, the database is marked suspect. Locks are held on the page that is corrupt. You can use a page-level restore to restore up to 1,000 corrupt pages. Here is an example:

```
RESTORE DATABASE 'test'
PAGE='1:324243, 1: 324244

 FROM Disk='c:\test.bak'
WITH RECOVERY
```

RETRIEVING THE TAIL OF THE LOG

The tail of the transaction log includes all committed transactions in the log since the last transaction log backup was done. Or, if no transaction log backup was performed, it would include all the committed transactions since the last backup was done. For example, the following syntax can be used to back up the tail of the log:

```
BACKUP LOG MyDatabase TO
disk='c:\thetail.back' WITH
NORECOVERY
```

This leaves the database in a restoring state until the database can be restored or the following command can be issued:

```
RESTORE DATABASE
MyDatabase with Recovery
```

Prior to SQL Server 2005, many DBAs might spend their careers without knowing about the tail of the log. With SQL Server 2005, that has changed because any time a database backup is restored into the database it came from, the following message appears:

```
Server: Msg 3159, Level 16,
State 1, Line 1
```

The tail of the log for the database "testrecovery" has not been backed up. Use BACKUP LOG WITH NORECOVERY to backup the log if it contains work that can't be lost. Use the WITH REPLACE or WITH STOPAT clause of the RESTORE statement to just overwrite the contents of the log.

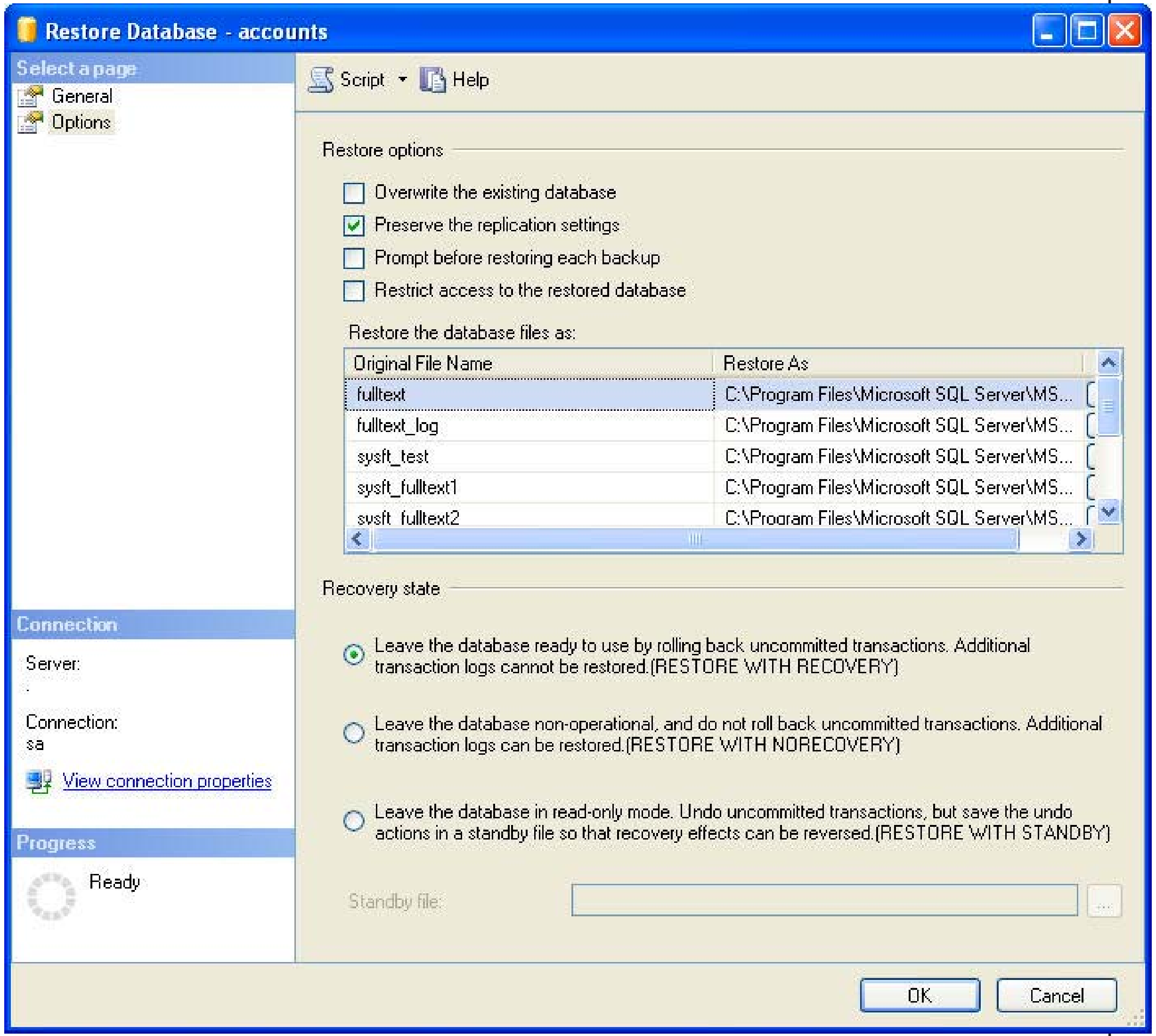
```
Server: Msg 3013, Level 16,
State 1, Line 1

RESTORE DATABASE is
terminating abnormally.
```

Why is this important? The ability to back up the tail was also possible in previous versions of SQL Server and it allowed DBAs to recover all the transactions in the log before a restore or for operations like log shipping.

For example, in a controlled failover of a log shipping environment, DBAs would backup the tail of the log to get the last committed transactions and put the database into a recovering state so that no applications could use it. Then the log would be applied on the secondary to prevent any data loss. However, SQL Server 2005 allows DBAs to back up the tail of the log even if the data files for their databases are offline or damaged. Here is the command that allows this:

```
BACKUP LOG MyDatabase TO
disk='c:\thetail.back' WITH
CONTINUE_AFTER_ERROR
NO_TRUNCATE
```

The command is useful in the event of a corrupt database file on the primary so that DBAs can extract all commands from


Restore Database - accounts

Select a page: General, Options

Restore options

- [ ] Overwrite the existing database
- [✓] Preserve the replication settings
- [ ] Prompt before restoring each backup
- [ ] Restrict access to the restored database

Restore the database files as:

Original File Name	Restore As
fulltext	C:\Program Files\Microsoft SQL Server\MS...
fulltext_log	C:\Program Files\Microsoft SQL Server\MS...
sysft_test	C:\Program Files\Microsoft SQL Server\MS...
sysft_fulltext1	C:\Program Files\Microsoft SQL Server\MS...
sysft_fulltext2	C:\Program Files\Microsoft SQL Server\MS...

Recovery state

- ● Leave the database ready to use by rolling back uncommitted transactions. Additional transaction logs cannot be restored.(RESTORE WITH RECOVERY)
- ○ Leave the database non-operational, and do not roll back uncommitted transactions. Additional transaction logs can be restored.(RESTORE WITH NORECOVERY)
- ○ Leave the database in read-only mode. Undo uncommitted transactions, but save the undo actions in a standby file so that recovery effects can be reversed.(RESTORE WITH STANDBY)

Standby file:

Connection — Server: . Connection: sa

Progress: Ready

FIGURE 1 The options tab of the restore database dialog illustrates the keep Preserver Replication Settings.

the log and do a complete as possible restore on a new or standby server.

TRANSACTION MARKS

Applications are frequently developed using multiple databases as a single unit. For example, configuration data may be stored in a database, and user data may be stored in another database.

Such topologies present problems to DBAs who must restore both databases to a consistent state. In SQL Server 2005, there is a new feature called "transaction marks." DBAs can give a name to transactions that span both databases and they can restore both databases to this named transaction mark. Script 1 is an example of how to use this feature.

PRESERVE REPLICATION SETTINGS

Notice in Figure 1, in the Options tab of the Restore Database Dialog box, that there is a way to preserve the replication settings, under "Recovery state."

This feature was part of SQL Server 2000, but it was not displayed in the Restore Database dialog.

This feature restores the publications. If — and only if — the distribution databases and msdb databases are restored on the same server, or a server with the same name —will replication run successfully. A replrestart may have to be issued to get it working again.

Restoring a published database into the same database or a published database does not remove the replication metadata. But restoring it into an unpublished database will. This feature is intended for use with the sync_with_backup option, which allows for synchronization of transactional replication with a log-shipped environment.

Script 2 is an example of how to use this feature.

FULL-TEXT CATALOGS

In SQL Server 2005, full-text catalogs are stored inside the backups and are restored with the databases. This may lead to large backup files and will increase the length of time it takes to back up and restore databases.

If backing up full-text catalogs results in database backups that are too large, then place the full-text catalogs in an alternate filegroup and only back up the primary filegroup — of the filegroups that do not contain your full-text catalogs.

While developing SQL Server 2005, Microsoft put a lot of effort into revamping the process of restoring data from damaged database files and backup media.

PIECEMEAL RESTORES

SQL Server 2005 Enterprise Edition allows DBAs to restore different filegroups of their databases in stages. With SQL 2000 a backup of an older filegroup could be restored into a database, but all the logs for that database more recent than the filegroup backup would need to be applied. In SQL 2005 Enterprise Edition DBAs need only to back up the tail, restore the filegroup and then restore the tail into the database. All of these operations are done while the database is online.

This is advantageous if some filegroups that comprise the database are damaged and a DBA wishes to restore the undamaged filegroups in order to access the data. Script 3 is an example of how to do this.

As always, you should run CheckDB before running backups to ensure that you're backing up valid data. Also, remember to verify backups after completing them.

PROTECTING MISSION-CRITICAL DATABASES

Microsoft has spent considerable time and resources to improve the backup and restore options in SQL Server 2005. The company put a lot of effort into revamping the process of restoring data from damaged database files and backup media. As databases get larger and a greater percentage of them become mission critical, it is essential that DBAs provide high availability. Tested disaster recovery plans become essential. The new features in SQL 2005 help with greater reliability and increased availability.

In SQL 2005 Enterprise Edition DBAs need only to back up the tail, restore the filegroup and then restore the tail into the database. All of these operations are done while the database is online.

Electronic Publishing Performs

“We spoke to a Microsoft SQL Server™ consultant and explained our problems. He was extremely helpful.”

— **Dave Martin**, Group Technical and Development Manager, Electronic Publishing

Electronic Publishing improves business continuity and performance using Dell™ PowerEdge™ blade servers and Dell | EMC storage technology

Electronic Publishing is a division of Johnston Press, the second largest local newspaper publisher in the UK. It is responsible for the company’s Web presence and hosts 290 websites for a range of newspapers, employment, property and trading organizations. Electronic Publishing has experienced enormous growth and popularity online in a very short time. Just five years ago the website business was worth £300,000, but by last year this figure had grown to a massive £6.9 million, with 3.2 million unique users per month.

With user numbers continuously expanding, the organization’s existing IT infrastructure faced capacity issues. Dave Martin, Group Technical and Development Manager, Electronic Publishing says: “Although a good problem to have, it became clear that we were going to hit capacity with our current system, and it needed replacing.”

As it stood, the database server was struggling to cope with the growth in user numbers, and server processing power was becoming stretched, threatening downtime.

Electronic Publishing Performs

Johnston Press

Electronic Publishing undertook a thorough review of the marketplace and chose Dell. Dell not only highlighted a greater understanding of Electronic Publishing's business needs, it showed a greater commitment to achieving its business goals.

Dell provided an independent consultant to advise on the best database solution to meet continued business growth. Martin says: "We spoke to a Microsoft® SQL Server™ consultant and explained our problems. He was extremely helpful. Not being bound to Dell or competitors, he provided an independent opinion and delivered the best business case for Microsoft® SQL Server™ 2005."

As part of the solution, Dell offered a proof of concept (POC) environment. This meant Dell pre-built a replacement infrastructure at its enterprise solution centre based in Ireland. The POC highlighted the advantages of Dell implementing Microsoft® SQL Server™. With the vendor's standards-based solutions and expertise, the organization simplifies the deployment, optimization and management of Microsoft® SQL Server™ 2005. The Microsoft® SQL Server™ 2005, married with Dell, has increased the overall performance of the business. There is now greater throughput and improved query notification. Additionally, business intelligence and enhanced management features are being employed for click stream analysis.

To view the entire story, click **here**

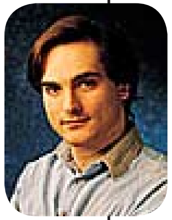
Partition indexes for improved SQL Server 2005 performance

BY SERDAR YEGULALP, CONTRIBUTOR

Index partitioning is one of a number of new features introduced in SQL Server 2005 as a way to distribute the load for a given index across multiple files, which can enhance parallelism or improve index performance in other ways.

Earlier editions of SQL Server accomplished index partitioning with **partitioned views**. Queries to and changes against tables could be constrained in certain ways by a view, so only needed physical files would be queried or modified. For instance (this is an arbitrary example but it suits our needs), if you had 26 tables for a customer database, one for each letter of the alphabet, you could use a partitioned view to aggregate the results from all of the tables and use WITH CHECK constraints to update only the needed tables. You could run

SERDAR YEGULALP *has been writing about Windows and related technologies for over 10 years, and is a regular contributor to various sections of TechTarget, as well as other publications. He hosts the website WindowsInsider.com, where he posts regularly about Windows and has an ongoing feature guide to Vista for emigrants from Windows XP.*



a query against all customers whose names begin with "B" and the partitioned view would know only to poll the "B" table.

The downside of partitioned views is that they must be created and managed manually. In SQL Server 2005, there's greater abstraction between partitions, tables and data, so they can be manipulated independently.

Also new is index partitioning, where the index (or indexes) for a given table are partitioned or constrained across multiple files and filegroups. Here I've assembled some basic guidelines for how to set up and use index partitioning; the exact details for how to do this are described in SQL Server 2005 Books Online.

CREATE AN INDEX WITH PARTITIONED DATA

There are two ways to create an index with partitioned data: Partition the index as the data is partitioned or partition the index separately. Which partition scheme to use should depend on how your data is accessed and updated.

In the first setup your index is *"partition aligned."* By default any newly-created index on a partitioned table will have the same partitioning as the table itself. This is best if you:

- know that a great deal of data will be inserted into the table over time;
- anticipate adding partitions as you go;
- and believe that these are the most important aspects of your data setup for this table.

Take the example where a partition arrangement covers a year of data split up according to months, where the date is used as a primary key. Partitioning the index this way speeds up date lookups since SQL Server can quickly determine where a given key may be in the index.

There are times you won't always want to use a partition-aligned arrangement, possibly in cases where you have a unique index key that does not contain the table's partitioning column. For instance, if we were using the above A to Z partitioning scheme but the index key for the tables was a GUID or auto-increment number rather than the customer name, you could keep the index in its own partition so it's not aligned with the table. In any case if you explicitly put the index on a dif-

There are times you won't want to use a partition-aligned arrangement, possibly in cases where you have a unique index key that does not contain the table's partitioning column.

ferent filegroup, partitioning will not match the table.

If you're partitioning data that has a unique index, the column used for partitioning be the same used for the unique index key. If your unique partition index is a client ID number, for instance, that will be the same column used to partition the index key as well.

UNDERSTAND PARTITION FUNCTION AND PARTITION SCHEME

Partitions are made up of two things: *partition function* and *partition scheme*. The first represents how data itself is split across different partitions. For instance, in the A to Z example, the data is partitioned according to each letter of the alphabet as 26 separate partition functions.

A *scheme* represents how each partition in the partition function is mapped to a filegroup. If our A to Z table has "A" data stored in a physical file in one filegroup and the "A" index stored in another physical file in the same filegroup, a partition-aligned index can help speed up and parallelize access to both the data and indexes. That way multiple CPUs can work on different partitions or physical files. (You can parallelize things further by placing indexes and data on separate physical spindles, if you have them.)

ACCOUNT FOR TEMPDB SPACE

Building partition-aligned indexes takes memory and uses space in **tempdb**. Many database administrators don't set tempdb's space allotments to anything beyond the default when they install SQL Server, and the time and effort it takes to auto-expand tempdb can put a crimp in performance. Also, indexes that are partitioned differently are built using different memory allocation schemes: a partition-aligned index is built with one sort table at a time, while a nonaligned index is built with all its sort tables at the same time.

Microsoft states in Books Online that the minimum size for each sort table per partition is 40 pages of 8 KB per page, so a nonaligned partitioned index with 26 partitions (using the previous A to Z example) would require 1,040 pages -- approximately 4.25 MB of memory. A nonaligned index would only need 163,840 bytes. For the most robust SQL Server installations this shouldn't be a problem, but be mindful if you're dealing with extremely large partitioning schemes *and* working with multiple partitioning schemes at one time.

SQLServer Insider MANAGEMENT

SQL Server 2005 upgrade hurdles

BY SERDAR YEGULALP, CONTRIBUTOR

Few people planning to upgrade to SQL Server 2005 will start from absolute scratch. Most will migrate to 2005 either from another database product entirely or from an earlier version of SQL Server. Here are some key issues to consider.

The first thing that needs to be touched on is whether the upgrade is going to be performed by taking an existing SQL Server installation and upgrading it or by installing SQL Server 2005 on a new machine and migrating databases to it. A clean install can avoid some issues that would go with upgrading an existing installation, and you can install it in parallel with an existing version; you don't have to take the existing SQL Server offline. But if you don't have the funds or the provisioning for a new server, then upgrading the existing installation may be your only choice.

In some ways Microsoft has done a good deal of the heavy lifting when it comes to figuring out what you're in for when upgrading. The Microsoft SQL Server Upgrade Advisor (last updated April 2006) examines installed instances of SQL Server 7.0 and 2000 to determine if blatant issues exist that need to be dealt with immediately. The utility is non-destructive and doesn't require you to take anything offline when you use it, so it can be run at any time.

During the installation itself, a component called the System Configuration Checker will scan your current SQL Server version to determine if there are any problems that would prevent the installation. Microsoft has documented parameters that may cause blocking issues, so you can pre-emptively check the system against any such problems if you want to (and it's probably a good idea).

When upgrading an existing installation, consider if the default settings in an earlier version of SQL Server will successfully convert to SQL Server 2005. Some of those modified defaults -- which may have been changed to solve specific issues now properly addressed in 2005 -- may have adverse side effects.

For instance, SQL Server 2005 will have memory buffer pool problems if the max server memory setting is not set to its default 2147483647 (i.e., all available memory). You can always fine-tune this value later if you need to. Another changed default, which can cause things to behave unexpectedly, is the query governor cost limit. SQL Server 2005 uses a different cost-modeling algorithm for queries. Set this to 0 before upgrading whenever possible.

As a side note, if the AUTO_UPDATE_STATISTICS option is turned off in any database to be migrated, re-enable it. Without it SQL Server 2005 can't generate optimal query plans.

On the other hand, before an upgrade you should disable the trace flags feature; some SQL Server 2000 trace flags simply do not exist in 2005. You should also disable duplicate security identifiers (SIDs), as they are unsupported in 2005.

"Before an upgrade you should disable the trace flags feature; some SQL Server 2000 trace flags simply do not exist in 2005. You should also disable duplicate security identifiers (SIDs), as they are unsupported in 2005."

Extended stored procedures that were previously registered *without the full path for the DLL name* may not work after you upgrade to SQL Server 2005. Run the sp_dropextendedproc and sp_addextendedproc stored procedures to drop and add back extended stored procedures if needed.

Another possible upgrade issue that is more related to databases in general than SQL Server itself deserves mention here: SQL Server 2005 uses slightly more data per column for some data types; text, ntext and image data types require 40 more bytes per column. For that reason, any migrated databases that use these data types should be allowed to grow automatically if they aren't already allowed to do so. The same goes for the tempdb database: Set it up to grow automatically during the course of the upgrade. Its own settings may be preserved during the upgrade, which is why it's worth looking into before starting.

ABOUT THE AUTHOR: Serdar Yegulalp is editor of the newsletter, Windows Insight.

You may encounter an upgrade issue related more to databases than SQL Server itself; Tempdb databases and migrated databases that use text, ntext and image data types should be allowed to grow automatically.

SQL Server INSIDER

is brought to you by SearchSQLServer.com. The stories "Partition indexes for improved SQL Server 2005 performance" and "SQL Server 2005 upgrade hurdles" originally appeared on SearchSQLServer.com

EDITORS
Chris Casatelli
Heidi Sweeney

COPY EDITOR
Martha Moore

DESIGN DIRECTOR
Ronn Campisi
www.ronncampisi.com

Additional Resources from Dell

- **System of Excellence: IT Infrastructure Keeps the Kenton School System at the Head of the Class**
  www.dell.com/sql

- **SQL Server 2005: Preparing for a Smooth Upgrade**
  http://www.dell.com/downloads/global/power/ps1q06-20060126-Microsoft.pdf

- **Maximizing SQL Server Performance**
  http://www.dell.com/downloads/global/power/ps4q05-20050272-Symantec.pdf

- **The Scalable Enterprise Technology Center**
  http://www.dell.com/content/topics/global.aspx/power/en/setc?c=us&cs=555&l=en&s=biz

- **Microsoft SQL Server 2005 Virtualization**
  http://www.dell.com/downloads/global/power/ps4q06-20060405-Muirhead.pdf

- **The Definitive Guide to Scaling Out SQL Server 2005**
  http://www.dell.com/content/topics/global.aspx/alliances/en/ebook_landing?c=us&cs=555&l=en&s=biz

SCRIPT 1

GO BACK TO PAGE 9

```
create database MarkedTransaction
GO
create database MarkedTransaction1
GO
use MarkedTransaction
go
create table test1(pk int)
GO
use MarkedTransaction1
GO
create table test1(pk int)
GO
backup database MarkedTransaction to disk='c:\
MarkedTransaction.bak' with init
GO
backup database MarkedTransaction1 to
disk='c:\MarkedTransaction1.bak'with init
GO
begin transaction
insert into MarkedTransaction.dbo.test1(pk)
values(1)
insert into MarkedTransaction1.dbo.test1(pk)
values(1)
commit tran
GO
begin transaction mark6 with mark 'marked
transaction'
insert into MarkedTransaction.dbo.test1(pk)
values(2)
insert into MarkedTransaction1.dbo.test1(pk)
values(2)
commit transaction mark6
GO
begin transaction mark7 with mark 'marked
transaction - 1'
insert into MarkedTransaction.dbo.test1(pk)
values(3)
insert into MarkedTransaction1.dbo.test1(pk)
values(3)
commit tran mark7
GO
```

SCRIPT 1 (continued)

```
backup log MarkedTransaction to disk='c:\
MarkedTransactionlog.bak' with init

backup log MarkedTransaction1 to disk='c:\
MarkedTransactionlog1.bak' with init

GO

use master

GO

--backing up the tail

backup log MarkedTransaction to disk='c:\
MarkedTransactionlogTail.bak' with init,
norecovery

backup log MarkedTransaction1 to disk='c:\
MarkedTransactionlog1Tail.bak' with init,
norecovery

GO

--restoring the database

restore database markedtransaction from
disk='c:\markedtransaction.bak' with
norecovery

restore database markedtransaction1 from
disk='c:\markedtransaction1.bak' with
norecovery

GO

--restoring the log

RESTORE LOG MarkedTransaction FROM

DISK = 'c:\MarkedTransactionlog.bak'

WITH  STOPATMARK = 'mark6', recovery

GO

RESTORE LOG MarkedTransaction1 FROM

DISK = 'c:\MarkedTransactionlog1.bak'

WITH  STOPATMARK = 'mark6', recovery

GO

select * from MarkedTransaction.dbo.test1

select * from MarkedTransaction1.dbo.test1

--both 1, 2 are there, 3 is not

--repeating,  but this time we will stop before

backup database MarkedTransaction to disk='c:\
MarkedTransaction.bak' with init

GO

backup database MarkedTransaction1 to
disk='c:\MarkedTransaction1.bak'with init

GO

begin transaction

insert into MarkedTransaction.dbo.test1(pk)
values(6)
```

SCRIPT 1 (continued)

```
insert into MarkedTransaction1.dbo.test1(pk)
values(6)

commit transaction

GO

begin transaction mark8 with mark ‘marked
transaction -2’

insert into MarkedTransaction.dbo.test1(pk)
values(7)

insert into MarkedTransaction1.dbo.test1(pk)
values(7)

commit transaction mark8

GO

begin transaction

insert into MarkedTransaction.dbo.test1(pk)
values(8)

insert into MarkedTransaction1.dbo.test1(pk)
values(8)

commit transaction

GO

backup log MarkedTransaction to disk=’c:\
MarkedTransactionlog.bak’ with init

backup log MarkedTransaction1 to disk=’c:\
MarkedTransactionlog1.bak’ with init

GO

use master

GO

backup log MarkedTransaction to disk=’c:\
MarkedTransactionlogTail.bak’ with init,
norecovery

backup log MarkedTransaction1 to disk=’c:\
MarkedTransactionlog1Tail.bak’ with init,
norecovery

GO

RESTORE Database MarkedTransaction
from disk=’c:\markedtransaction.bak’ with
norecovery

RESTORE Database MarkedTransaction1
from disk=’c:\markedtransaction1.bak’ with
norecovery

GO

RESTORE LOG MarkedTransaction FROM

DISK = N’c:\MarkedTransactionlog.bak’

WITH  STOPBEFOREMARK = N’mark8’ , recovery

GO

RESTORE LOG MarkedTransaction1 FROM
```

SCRIPT 1 (continued)

```
DISK = N'c:\MarkedTransactionlog1.bak'
WITH  STOPBEFOREMARK = N'mark8' ,
recovery
GO
select * from MarkedTransaction.dbo.test1
select * from MarkedTransaction1.dbo.test1
--both 1, 2, and 6 are there, 7 is not
```

In SQL Server 2005, the new "transaction mark" feature allows DBAs to name transactions that span more than one database, and then they can restore the databases to this named transaction mark.

SCRIPT 2

GO BACK TO PAGE 9

```
create database script2
GO
use script2
go
create table test1(pk
int not null primary
key, charcol char)
GO
sp_replicationdboption
'script2', 'publish','true'
GO
sp_addpublication
'test',@status='active'
GO
sp_addpublication_
snapshot 'test'
GO
sp_addarticle
'test','test1','test1'
GO
backup database
script2 to disk='c:\
script2.bak'
GO
create database
script2a
GO
restore database
script2a] from
disk = 'c:\script2.bak'
with  move 'script2' to
'c:\script2a.mdf',  move
'script2_log' to
'c:\script2a_log.ldf',
keep_replication,
replace
GO
use script2a
GO
sp_helppublication
'test'
—notice how there is a
publication here
```

SCRIPT 3

GO BACK TO PAGE 10

```
create database script3
GO
use script3
go
alter database script3 add filegroup ReadOnly
GO
alter database script3 add file (name =
'script3readonly',
filename = 'c:\script3readonly.ndf') to filegroup
readonly
GO
create table test(pk int not null identity, intcol
int)
create table test1(pk int not null identity, intcol
int) on readonly
GO
--lets put some data in our tables
declare @counter int
set @counter=1
while @counter<100
begin
insert into test(intcol) values(@counter)
insert into test1(intcol) values(@counter)
select @counter=@counter+1
end
GO
alter database script3 modify filegroup readonly
readonly
GO
backup database script3 filegroup ='readonly'
to disk='c:\readonly.bak'
GO
declare @counter int
set @counter=1
while @counter<100
begin
insert into test(intcol) values(@counter)
select @counter=@counter+1
end
GO
```

Copy -only backup | Checksum | Full-text catalogs | Partition indexes | SQL Server 2005 upgrade

SCRIPT 3 (continued)

```
use master
GO
--backup the tail, restore the filegroup and
then restore the tail
backup log script3 to disk='c:\script3tail.bak'
with norecovery
GO
restore database script3 file = 'script3readonly'
from  disk = 'c:\readonly.bak' with  file = 2,
norecovery
go
restore log script3 from  disk = 'c:\script3tail.
bak'
go
```

Got a question on SQL Server 2005 backup? We've got answers!

For three years, SearchSQLServer.com's panel of experts has fielded a wide range of questions on database backup and restore as well as other essential SQL Server topics, such as management, development, security and performance.

Check out our knowledgebase of SQL Server questions from your peers and the answers from our SQL Server gurus. If you don't find a solution to your specific problem, you can send your own question directly to our experts:

GREG ROBIDOUX Backup and Recovery. Greg is founder and Principal Database Engineer at Edgewood Solutions.

ADAM MACHANIC SQL Server 2005. Adam is co-author of Pro SQL Server 2005, published by Apress.

JEREMY KADLEC Performance Tuning. Jeremy Kadlec is Principal Database Engineer at Edgewood Solutions.

—Heidi Sweeney