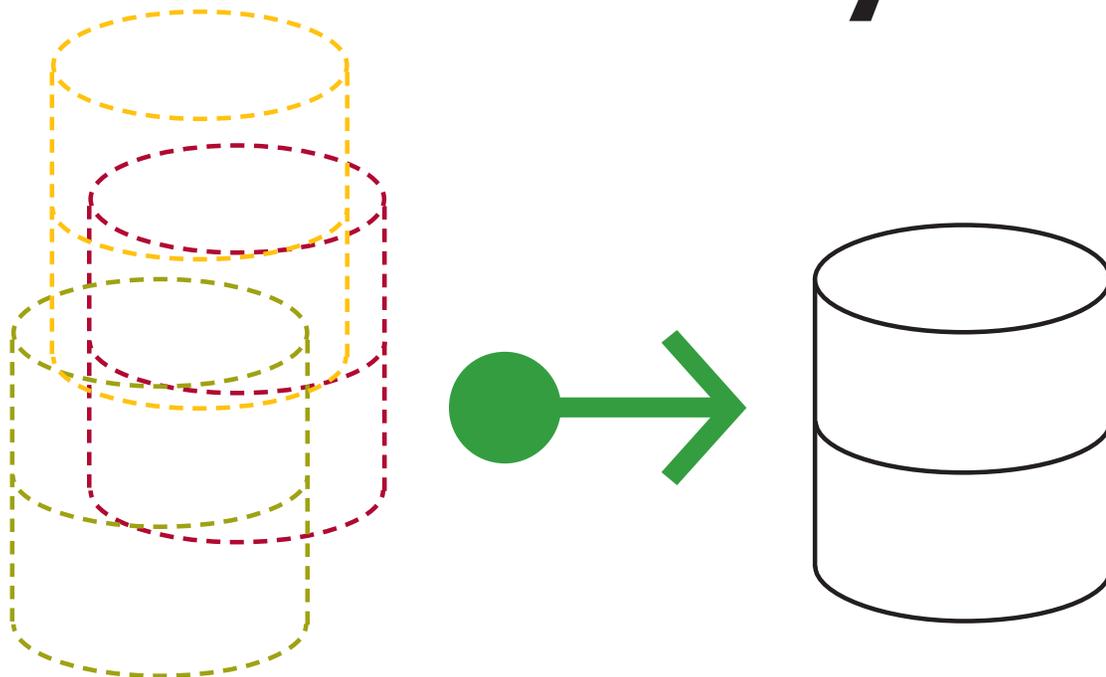


# SQL Server INSIDER

Tips for SQL Server pros September 2007

## Selecting the right SQL Server **recovery model**



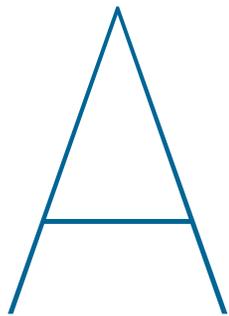
Choose the best backup and recovery model for your SQL Server environment, or be left with a false sense of security.

### INSIDE

- **04** OLTP databases
- **06** Commands vs. maintenance plans
- **07** Testing your backups
- **11** **ARTICLE 1:** SQL Server password management: Six risky assumptions
- **14** **ARTICLE 2:** SQL Server query design: 10 mistakes to avoid

# Selecting the right SQL Server recovery model

BY DENNY CHERRY



As is the case with most duties of a database administrator, there are several strategies to choose from when it comes to a SQL Server backup and recovery model. None of the available options are wrong per se, but if the model you choose isn't the best model for your environment, it may give a false sense of security to DBAs, upper management, business users and business owners. DBAs must have a backup and restore

strategy in place, and they must be certain it's the best plan for their SQL Server environment.

There are six main backup methods in use today:

- 1 File-level backups
- 2 Full backups only
- 3 Full backups with differentials
- 4 Full backups with transaction logs
- 5 Full backups with differentials and transaction logs
- 6 File group backups



File-level backups are not accepted by the database engine as valid because they are performed offline or when the database is detached from SQL Server.

DENNY CHERRY



is a DBA and database architect managing

one of the largest SQL Server installations in the world, supporting more than 175 million users. Denny's primary areas of expertise are system architecture, performance tuning, replication and troubleshooting.

Of these six methods, only five are recognized by Microsoft SQL Server as valid backups. The one method not recognized is the file-level backup. A file-level backup is not accepted as a valid backup because it is an offline backup. It must be performed while SQL Server is offline or the database is detached from SQL Server.

With file-level backups, the database engine is not aware of the backup; therefore, it cannot ensure that the backup is valid. So far, there has not been a third-party tool that can do a file-level backup while the SQL Server engine is running and accepting transactions. When you perform proper backups using other methods, a file-level backup isn't necessary for SQL Server recovery.

A seventh possible backup method is storage-based backups, called snapshots or snaps. Vendors provide different database backup types — crash-recoverable database as a backup, crash-restartable database as a backup — and some provide a valid backup registered with SQL Server as a true backup. The latter can have transaction log backups applied to it. Check with your storage vendor for the specific capabilities of your storage platform.

The five primary backup methods are all performed within the SQL Server engine, so they are tracked and recorded.

## RECOVERY MODEL OPTIONS IN SQL SERVER

Each database backup method is designed to be used with a specific database recovery level — either simple, bulk-logged or full:

	Full backups	Full backups with differentials	Filegroup backups	Full backups with transaction logs	Full backups with differentials and transaction logs
Simple	✓	✓			
Bulk-logged		✓	✓	✓	✓
Full			✓	✓	✓

When designing your backup and recovery strategy, it is important to think about all facets of the design. You must consider the time and space needed to take the backups of the databases and also the time needed to restore those databases. In addition, weigh the complexity of the restore against the urgency of having the most recent data. When assigning storage for your database backups, be sure to account for having multiple copies of the database. At least two full backups and all the backups between those full backups should be maintained on site.

Databases using a simple recovery mode with low to no data changes can safely use full backups, provided those backups are regularly scheduled. Databases that are probably kept in either simple or bulk-logged recovery modes include databases, database reporting systems and decision support databases. Those databases typically have a high data change rate, which means that there's lots of new data or data being purged. This data change typically happens in a batch process, so the full backups with differentials method will fit the system nicely.

The largest data warehouses back up their databases with the filegroup method. It's typically done on systems that have partitioned one or more tables into multiple filegroups. With this method, you back up a partition frequently while it is the active partition. Once that partition is no longer being updated, you archive the final backup of that filegroup and begin taking backups of the new partitions.

**When selecting the frequency for your transaction log backups, the most important question to ask is how much data loss is acceptable in the event of a hardware failure.**

**OLTP DATABASES ARE** almost always kept in either the bulk-logged mode or full-recovery mode, and you should always back up transaction logs in databases that fit this description. By backing up the transaction logs, you ensure that only the minimum number of transactions are lost in the event of a recovery situation. When selecting the frequency of your transaction log backups, the most important question to ask yourself — and the business unit that owns the data — is how much data loss is acceptable in the event of a hardware failure.

Let's assume you work for Northwind Traders and have a database called Northwind. Management was asked about the backup interval and told you they think 10 minutes worth of data loss is acceptable in the event of a system failure. So, you schedule your transaction log backups to run every 10 minutes, which means you'll be taking 144 transaction log backups per day.

Assume you schedule the full backups to run nightly at midnight, and SQL Server crashes at 10 p.m. You now have 132 backups to restore in the correct order before you can bring the system online. This is where our differential backups come into play. Differential backups contain all the database changes since the prior full backup. You

can take differential backups every four hours at 4:00, 8:00 and 12:00 (both a.m. and p.m.). This leaves just 12 transaction log backups, for a total of 14 backups to restore including your full and most recent differential backup.

Because you planned your backup strategy with recovery in mind, you have saved yourself a major restore project in the middle of the night and brought the system back online faster. It is often faster to restore a single differential than many smaller transaction log backups. While it is not *always* the case, it is a good rule of thumb.

### **DATA WAREHOUSE BACKUP AND RECOVERY METHODS**

Separate from the OLTP database is your data warehouse. The data warehouse should be using a completely different backup and recovery strategy. Data warehouses typically get a single batch of changes once per day. Because of this, the databases are typically kept in simple or bulk-logged recovery mode. Keep that in mind, too, when you're planning your backup strategy.

Performing transaction log backups may or may not be necessary. If they are, you only need to do them after the data load is complete. The differential backup also needs to be taken only once per day — again, once the data load has completed.



Removing backup history from the msdb database reduces the size of that database and improves restore performance. This strategy should be part of your regular backup procedure.

Any other backups throughout the day would simply be a duplicate of those done prior and would waste space on the tape or disk.

Full backups should be taken every few days or weekly depending on the amount of data changes seen in the differential backups. If your data change rate for your data warehouse is close to or above 50% daily, then daily full backups will probably be the best solution for your system.

### **BACKUP MAINTENANCE**

Part of your regular backup procedures should include removing old database backup history from the msdb database. That reduces the size of the msdb database and improves performance of the Enterprise Manager or Management Studio applications when you use them to restore.

This maintenance should be done via the `sp_delete_backuphistory` procedure in the msdb database. Microsoft provides it through the current

CTP of Microsoft SQL Server 2008. The procedure accepts a single parameter called `@oldest_date`, which is the oldest date's backup information to keep. Differences between full backups, differential backups and transaction log backups are not shown. So, by running the command written below, you could end up with entries in the `msdb` database for transaction log backups and no corresponding full backup. That's fine because the backups don't have to be listed in the `msdb` tables in order to be restored to the SQL Server.

The maintenance command is as follows:

```
exec sp_delete_backuphistory '1/1/2005'
```

Another handy procedure is `sp_delete_database_backuphistory`, shown below. This removes all backup history for the database specified and should be used when databases leave production and are dropped from the server. For example:

```
exec sp_delete_database_backuphistory  
'Northwind'
```

### **BACKUP DATABASE COMMANDS VS. MAINTENANCE PLANS**

Although many database administrators have a preference, there is no actual difference with regard to backups between writing jobs that call the `BACKUP DATABASE` and `BACKUP LOG` com-

mands and SQL Server maintenance plans. Both provide backups that you can restore using the `RESTORE DATABASE` command. Either method can be easily set up. Although maintenance plans provide a nice wizard to walk you through them, they do not allow for all the available options that `BACKUP DATABASE` and `BACKUP LOG` commands provide.

If you have worked on SQL Server 6.5, you'll remember having the ability to mirror the backup across multiple destinations to increase redundancy of the backup job. Microsoft restored this functionally in Microsoft SQL Server 2005 where it's done with the `MIRROR TO` clause. In using the `MIRROR TO` clause, the `FORMAT` switch must be included when creating the initial mirrored backup set. Like the regular destination, you can specify UNC paths to a network share on another server.

```
BACKUP DATABASE Northwind TO DISK='C:\  
northwind.bak' MIRROR TO DISK='\\  
SecondaryServer\c$\ northwind.bak'
```

When using the `MIRROR TO` clause, either `DISK` or a backup device can be specified. If any of the paths listed in the backup command are not available, the `BACKUP DATABASE` command will fail with an error message.

## IMPROVING BACKUP PERFORMANCE

If you are not getting the backup performance you need when taking your large backups, span them across several backup devices, called a media family. A media family includes disk, tape and backup devices as targets. Note: If any of the backup files are lost or damaged, all the backup files from that media family are lost.

```
BACKUP DATABASE Northwind TO DISK='C:\northwind_file1.bak',  
DISK='C:\northwind_file2.bak'  
WITH FORMAT
```

When restoring from a media family, you must specify each member of the family.

```
RESTORE DATABASE Northwind FROM DISK='C:\northwind_file1.bak',  
DISK='C:\northwind_file2.bak'
```

## TEST YOUR BACKUPS

Once you've backed up your data, you have to be ready to restore it at a moment's notice. The only way to ensure that you can restore the databases when needed is to test your backups. Testing backups means more than simply verifying them. A proper test involves a test restore.

A test restore simply means restoring the database to a test or development server. Ideally, it

**The only way to ensure that you can restore the databases when necessary is to test your backups and that means performing a test restore.**

should be done monthly, at a minimum. There are countless stories of companies that performed backups for years only to find out that the tape drive didn't work or the tape wasn't formatted properly. Most of the time these issues are not discovered until the worst has happened, and the emergency restore must be implemented. To put it nicely, this leaves the company in a very bad position: no database, no backups and no way to get the information back.

Schedule your test restores to run on a regular basis. The restores should come from your regular backups, not your special backups. Doing test restores from backups taken for the purpose of testing the restores invalidates the test, as those backups would not be available in typical restore circumstances.

Database restores can be scheduled as a SQL Agent Job and have the job run the RESTORE DATABASE command. The hardest part of scheduling a restore is identifying the filename of the

database. It can be done in a number of ways, but the easiest way is to use the msdb database and query for the filename of filenames the database was backed up to. Here is the basic query to find the file names:

```
SELECT physical_device_name
FROM msdb.dbo.backupmediafamily
backupmediafamily
JOIN msdb.dbo.backupset backupset ON
backupmediafamily.media_set_id = backupset.
media_set_id
    and backupset.backup_start_date = (SELECT
max(backup_start_date)
    FROM msdb.dbo.backupset child
    WHERE child.database_name =
backupset.database_name)
    and database_name = 'Northwind'
```

In order to schedule a dynamic restore, the following code can be used:

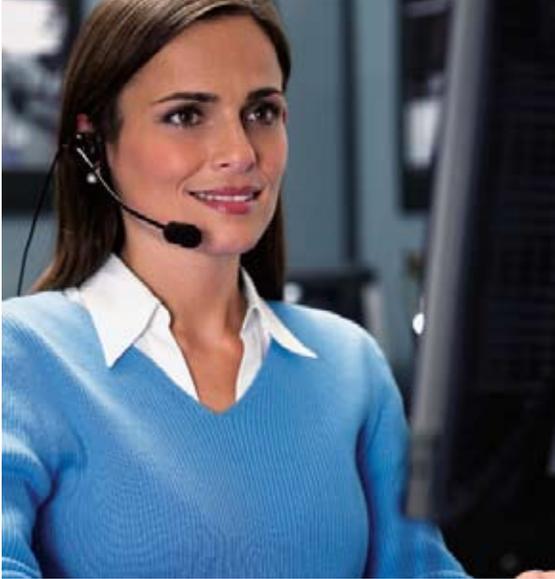
```
DECLARE @physical_device_name as
nvarchar(4000)
SELECT @physical_device_name = physical_
device_name
FROM msdb.dbo.backupmediafamily
backupmediafamily
JOIN msdb.dbo.backupset backupset ON
backupmediafamily.media_set_id = backupset.
```

```
media_set_id
    and backupset.backup_start_date = (SELECT
max(backup_start_date)
    FROM msdb.dbo.backupset child
    WHERE child.database_name =
backupset.database_name)
    and database_name = 'Northwind'
RESTORE DATABASE Northwind FROM DISK = @
physical_device_name
```

---

So, YOU'VE SELECTED the right recovery model, taken backups and tested restoring those backups, but there's one equally important component that remains. After taking your backups, be sure to secure them. Not securing the folder where the backups reside can, in turn, give anyone on the network access to your entire database. Someone with malicious intent would simply need to restore the database to another server or workstation running MSDE. The best method of securing a backup would be to use the MEDIANAME, NAME, MEDIAPASSWORD and secure the folder via NTFS permissions and network share permissions. Don't be one of the many DBAs who forget this important step of backup management.

# Ticket to Growth



## ServiceU gains 99.999% uptime, more than \$900,000 in benefits, and 595% return on investment thanks to Dell PowerEdge Servers and Microsoft SQL Server 2005

Memphis-based ServiceU, in business since 1997, is an on-demand service provider which delivers Web-based software for event management. It serves more than 1,000 organizations worldwide, ranging from Fortune 500 companies to public universities and small, nonprofit institutions. The company's software has been used to handle scheduling for more than 12 million events.

The always-on availability of ServiceU's databases and Web-based software is the key to the company's success. "The entirety of our business is done online," explains ServiceU Chief Technology Officer, David P. Smith. "It accounts for all of our revenue, so uptime is crucial to us."

“The Dell and SQL Server 2005 database mirroring solution allows us to eliminate risks from disasters and assure our customers that they will always have the same level of service that they rely on.”

— David P. Smith,  
Chief Technology Officer,  
ServiceU



# Ticket to Growth

ServiceU



ServiceU faces some unique challenges in maintaining such high levels of availability. Payment Card Industry (PCI) standards mandate that level-one PCI service providers like ServiceU meet a rigorous set of disaster recovery (DR) requirements. In addition, the company's Memphis offices sit atop an active fault. Says Smith, "according to seismologists, we are within 40 years of another major earthquake, so we have to be prepared."

Faced with the requirement of always-on availability, quick disaster recovery (DR), and compliance with PCI standards, ServiceU realized that it needed to improve on a DR plan that called for physical tapes to be sent by helicopter between facilities. After a detailed analysis, ServiceU turned to Dell and Microsoft SQL Server 2005 with Database Mirroring, to mirror the databases from its main facility in Memphis to its disaster recovery facility in Atlanta.

"Database mirroring allows us to have the Atlanta facility functional, compliant, and ready to use at a moment's notice," says Smith. "Additionally, the Dell and SQL Server 2005 database mirroring solution allows us to eliminate risks from natural disasters, such as earthquakes, and assure our customers that they will always have the same level of service that they rely on."

Thanks to the mirroring solution, ServiceU can guarantee high availability that will help it expand into new markets. ServiceU expects to realize a cumulative, projected, three-year net benefit of US\$908,985 which will result in an ROI of 595 percent and a payback period of five months.

To view the entire story, [click here](#).

## SQL Server Insider **MANAGEMENT**

# SIX risky assumptions

SQL Server  
password  
management:

BY KEVIN BEAVER

WHEN MANAGING INTERNAL SQL Server accounts and passwords, it's easy to assume that everything is reasonably secure. After all, your SQL Server systems are behind the firewall, you're forcing Windows authentication and all users are required to have a strong password. Sounds pretty secure — especially since you're thinking everyone else is doing it, right? Well, not so much.

There are several risky assumptions about SQL Server passwords that can get you into a bind quite easily. Here's what not to assume:

# 1

### **BASIC PASSWORD TESTING REQUIRES NO PLANNING.**

It's a big mistake to just start cracking away at SQL Server passwords when testing. Whether you're doing it locally or across the network, which is much slower, I highly recommend obtaining permission and having a fall-back plan when and if accounts start become locked. The last thing you want is to have

KEVIN BEAVER is



an independent security consultant

and speaker with

Atlanta-based

[Principle Logic](#).

He has nearly two decades experience

and specializes in performing security assessments. Beaver

has written seven books on security,

including [Hacking For Dummies](#)

(Wiley). He's also

the creator/producer of the [Security On Wheels](#)

audiobook series. He can be

reached at [kbeaver@](mailto:kbeaver@principlelogic.com)

[principlelogic.com](http://principlelogic.com).

database users unable to do their jobs or applications not working properly because accounts are locked.

## 2 YOUR PASSWORDS ARE SAFE GOING ACROSS THE WIRE.

For mixed-mode implementations of SQL Server, you could easily use a network analyzer such as [OmniPeek](#) or [Ethereal](#) to grab passwords right off the wire or out of thin air in the case of wireless. [Cain and Abel](#) can also be used to glean TDS-based passwords. Running switched Ethernet that's immune to network sniffing? Cain's ARP poison routing "feature" can take care of that. Within a minute or so, anyone using this free tool can make your Ethernet switches behave like hubs and see all internal traffic on the local network

**Windows authentication in SQL Server does not equal better security. Tools like Cain and Abel can be used to glean Windows, Web, email and related passwords right off the network.**

segment facilitating the capture of passwords.

The issues don't stop here. There's a misnomer that using Windows authentication in SQL Server equals better security. Not true. These same tools can be used to glean Windows, Web, email and related passwords right off the network as well — all of which very likely tie right back to SQL Server access.

## 3 THERE'S NO NEED TO TEST YOUR PASSWORDS — YOU HAVE A PASSWORD POLICY.

Regardless of how stringent your password policy may be, there will *always* be a gap that provides some way around it. Be it a misconfigured server, a host outside your Windows domain, an unknown SQL Server installation or some fancy tool that can crack even the strongest passwords — something is bound to introduce password weaknesses and effectively nullify your password policy.

Just as important, never rely on the results of a checklist audit that says your database is secure because strong passwords are being used. Always take your testing to the next level and actually verify whether or not weaknesses exist. Even if you think all is well, you'll likely uncover something predictable.

## 4 SQL SERVER PASSWORDS AREN'T RECOVERABLE, SO WHY TRY TO CRACK THEM IF I KNOW THEY'RE STRONG AND SECURE?

Actually, you *can* recover SQL Server passwords! In SQL Server versions 7 and 2000, you can recover the password hashes using a tool such as [Cain and Abel](#) or the commercial product [NGSSQLCrack](#) and then run a dictionary or brute-force attack against them. These tools allow you to dump and then reverse engineer the SQL Server password SHA hashes. Your cracking results aren't guaranteed, but it's a weakness nonetheless.

## 5 YOU'VE USED MBSA TO CHECK FOR SQL SERVER PASSWORD FLAWS AND NOTHING SERIOUS TURNED UP.

Microsoft Baseline Security Analyzer is a great starter tool for rooting out SQL Server vulnerabilities, but it's by no means comprehensive — especially in the password cracking department. For more in-depth SQL Server and Windows password cracking, turn to third-party tools, such as the free [SQLat](#) and [SQLninja](#) tools available in the [BackTrack](#) as well as Windows-centric password cracking tools such as [ElcomSoft's Proactive Password Auditor](#) and [Ophcrack](#).

Again, even though you're using Windows

authentication in SQL Server, you still could have password weaknesses. People with a little bit of time and know-how can crack Windows passwords and own the entire network — especially if they're using [Ophcrack's LiveCD](#) against a physically unsecured Windows host like a laptop or easy-to-access server.

## 6 YOU ONLY NEED TO WORRY ABOUT YOUR MAIN DATABASE SERVERS.

It's easy to focus on your bread and butter SQL Server systems, but don't forget about MSDE, SQL Server Express and other random SQL Server installations you likely have across your network. Some or all of them may be using unsecured defaults or have no password requirements at all. Use a tool such as [SQLPing 3](#) to track down these "rogue" database servers on your network. You'll likely be surprised at what you uncover.

As with anything else in IT, it's always the little stuff that gets you. Ditch these dangerous SQL Server password assumptions and oversights and you'll undoubtedly improve your SQL Server security.

*SQL Server Insider* **PERFORMANCE**

# 10

## SQL Server query design: mistakes to avoid

BY JEREMY KADLEC

WITH THE SUBSTANTIAL and sustained data growth of SQL Server databases, coupled with the sub-second response times expected by users, it is critical to avoid poorly written queries. Take a look to make sure you are not falling victim to these mistakes and consider the recommendations as a means of correcting your queries.

JEREMY KADLEC



is the principal database engineer at

Edgewood Solutions, a technology services company delivering professional services and product solutions for Microsoft SQL Server. He has authored numerous articles and delivers frequent presentations regionally and nationally. He authored the “Rational Guide to IT Project Management” and is the SearchSQLServer.com performance and tuning expert.

## TOP 10 LIST OF QUERY DESIGN CONSIDERATIONS

### 1 DATA MODEL AND SUBSEQUENT QUERIES

Not thinking about how to access the data as you are building the data model can result in unwieldy queries. You may have unnecessary JOINS that overcomplicate the code and hurt performance.

To correct this problem, think about the queries needed to access the data. If the query is not clear at this stage of the process, then it will be even more difficult to code. Odds are that the database design may be overcomplicated and can be simplified to improve query performance.

On a related note, if you are a visual person, be sure to print out the data model or review the online model in your [data modeling tool of choice](#). This should improve your coding time and accuracy.

### 2 WHAT'S THE BEST TECHNIQUE?

This is the notorious cursor-logic versus set-based logic debate. Conventional wisdom says to use set-based logic for all database access. In general, I would agree that is the best rule of thumb. Using cursors when set-based logic is the right choice can also impose significant performance penal-

ties. SQL Server is designed for set-based logic and should be used in most processing.

On the other side of the coin is this [cursor example](#). In this situation, cursor logic outperformed set-based logic. The takeaway from this information would be to determine the type of processing you are performing and select the technique that best suits the need.

### 3 DOING IT THE OLD WAY...

SQL Server 2005 brought a whole new set of opportunities for your queries. So doing things the old way may still work, but it may be time to consider the latest options. The [TRY...CATCH](#) error-handling method is one of the first techniques you should embrace in your code. Additional considerations are [common table expressions](#) for working with hierarchies; and a final consideration is to extend the capabilities of the relational database engine: [common language runtime](#) (CLR). These three technologies are significantly changing how you can work with SQL Server, and they're just the tip of the iceberg.

### 4 WOULD YOU TAKE A GANDER OVER HERE?

Reviewing your code and scheduling a peer review is a must-do before you deploy code. Review

## There is no better preparation for your SQL Server queries than testing against millions of records in fragmented tables in the test environment.

your code — and specifically the query plans — to ensure that proper indexes are being used and that the query will perform as expected.

On a related note, one of the simplest means to ensure the code is accurate is to have another set of eyes review it. This can be a learning experience for both the developer and his or her peer to see how other developers and DBAs approach problems. It's also a means to trade ideas to improve each other's skills.

**5 IT'S THE CLASSIC MISTAKE.**  
Issuing a `SELECT *` statement, thinking that the table will never change, is a classic query design mistake. Even in the simplest situations, inevitably the table changes and that leaves you to review the code to make sure the additional column is not included. Or, worse yet, you must wait for the application to break and then fix those issues. The

best practice is to only include columns that are needed in your queries and change them as necessary. Don't disrupt your day to search through code in firefighting mode.

**6 I HAVE NO COMMENTS!**  
Unfortunately, most code that I see has few or no comments. So making a change is a daunting task even for the developer and/or DBA who originally developed the application. Commenting your code is truly a quick and painless process, and it is critical for future developers to understand and change the code in a safe and timely manner.

**7 SURE, I WILL TEST IT...**  
Few developers and DBAs enjoy simple testing, nor do they enjoy rigorous testing prior to releasing the code to the production environment. Furthermore, development environments typically are not to the scale of the production environment in terms of the hardware and volume of data. That said, simple queries will perform well with a couple hundred or even a few thousand records; but, in production, that is not the case. There is no better preparation for your queries than testing against millions of records in fragmented tables in the test environment to make sure the queries will perform as expected.

## 8 LET ME HAVE IT — I MEAN IT!

Issuing SELECT statements without a WHERE clause and expecting the middle tier or front end to process the data in a more efficient manner than SQL Server is a tough sell. SQL Server is designed for query processing and does so very efficiently. Moving large sets of data is only going to bog down systems and networks to a point where they are flooded. Be sure to filter your data set as much as possible to avoid the performance implications.

## 9 I WOULD LIKE A QUERY WITH A VIEW, PLEASE.

Views fulfill the need to simplify coding for complex queries. They are used often to help power users query the database. Unfortunately, too much of a good thing can severely impact performance. The view is simply a SELECT statement and the view's SELECT statement must be issued each time your SELECT statement is issued. Limit the use of views and prevent them from querying other views. Or, build a stored procedure to query the data and pass in the needed parameters to fulfill application or user needs.

## 10 NO, IT'S NOT MY CODE...

We all make mistakes, and the last system we worked on would benefit from the knowledge we

are gaining on our current project. So, record the items you have learned and share them with your team so that the organization will benefit. When you have an opportunity, go back to those previous systems and improve them with the knowledge you have gained since that project.

### SUMMARY

If you are making these mistakes or others with your queries, recognize the mistakes and make the effort to correct them. That may be easier said than done, but correcting the issue will reap benefits for the organization and for the application's reputation. Begin to build a personal coding guide to use for your current and future projects.



SQL Server is designed for query processing and does so very efficiently. These query design dos and don'ts will lead to improved performance and the capability for that sub-second response time expected by users.

## Additional Resources from Dell

- **Dell's SQL Server 2005 Tested & Validated Configurations**  
[http://www.dell.com/content/topics/global.aspx/sitelets/solutions/software/db/microsoft\\_sql\\_2005\\_se?c=us&cs=555&l=en&s=biz](http://www.dell.com/content/topics/global.aspx/sitelets/solutions/software/db/microsoft_sql_2005_se?c=us&cs=555&l=en&s=biz)
- **Dell's SQL Server 2005 Reference Architecture**  
[http://www.dell.com/downloads/global/solutions/sql\\_server\\_2005\\_reference\\_architecture\\_w2k3\\_std.pdf?c=us&cs=555&l=en&s=biz](http://www.dell.com/downloads/global/solutions/sql_server_2005_reference_architecture_w2k3_std.pdf?c=us&cs=555&l=en&s=biz)
- **Dell SQL Server 2005 Advisor Tool**  
[http://www.dell.com/content/topics/global.aspx/tools/advisors/sql\\_advisor?c=us&cs=555&l=en&s=biz](http://www.dell.com/content/topics/global.aspx/tools/advisors/sql_advisor?c=us&cs=555&l=en&s=biz)
- **Dell Tech Center Wiki**  
<http://www.delltechcenter.com/>
- **Case Study: University of Mary Hardin - Baylor**  
[http://www.dell.com/downloads/global/casestudies/456\\_UMHB\\_9.pdf](http://www.dell.com/downloads/global/casestudies/456_UMHB_9.pdf)
- **Optimizing Microsoft SQL Server 2005 Environments with EMC Assessments and Quest Software**  
<http://www.dell.com/downloads/global/power/ps4q06-20070103-EMC-Quest.pdf>

## SQL Server INSIDER

is brought to you by  
[SearchSQLServer.com](http://SearchSQLServer.com).  
The articles “SQL Server password management: Six risky assumptions” and “SQL Server query design: 10 mistakes to avoid” originally appeared on SearchSQLServer.com

### EDITORS

Heidi Sweeney  
Christine Casatelli

### COPY EDITOR

Martha Moore

### DESIGN

Ronn Campisi  
[www.ronncampisi.com](http://www.ronncampisi.com)