

## Politicking, or Not

I must warn you against getting mired in office politics. That is a no-win situation for anyone. Aim for good, healthy relationships with coworkers. Healthy relationships are not about conspiring together and beating others down. Healthy relationships are about working together and building others up.

**CAUTION:** Do not get mired into gossip or office politics.

Do not get involved in conjecture on others, gossip, or rumors. Stick to known facts only—not hearsay—and do not knowingly spread false information. Office politics can easily drag down morale, create divisions, and just make life miserable.

## Dealing with Introversions

Not all technical people are introverted, but many are. The nature of our work calls for detail-oriented people comfortable at working alone for long periods of time. Those traits are in stark contrast to the outgoing salesperson who can meet 15 new people in 10 minutes, and be bosom-buddies with all of them. We are the people who can sit in a conference lecture with 100 other people and walk out not having met a single one of them.

Yet career success demands that we make ourselves visible. Sometimes we need to step up and lead, or present on what we know. We need to be comfortable in meeting people and working together. And we need to do all these things while being true to ourselves and respecting the personalities that we were born with.

## Accept That Nothing Is Wrong

If you find that you're the sort of person to want to have lunch alone, do not despair. The very first thing that you should do is

*Realize that you are not put together wrong.*

Yes, it is true. If our entire society were made up of social bubbleheads running around and meeting each other, we'd all be in a world of hurt. We need those outgoing people. Your company needs its sales force, for example. But we need all the other personality types, too. Those salespeople need products and services to sell.

The key is to “know thyself.” Know your own comfort zone. Play to your strengths. Develop skills that you lack. You can actually treat “extroversion” as just another skill to master. Doing so will pay handsomely in both personal and job satisfaction.

**NOTE:** A good book on personality traits is *Please Understand Me*, by David Keirse and Marilyn Bates (Prometheus Nemesis, 1984).

## Take Care of Yourself

The defining trait of an introvert is sometimes said to be that

*An introvert draws energy from being alone, and expends energy to engage in social situations.*

Does that sound like you? If it does, then don't be afraid to lay down some boundaries. For example, when travelling to conferences, my editor generally blocks out breakfast for himself. That's his "me time," when he can charge his batteries for the day. He also takes himself off the clock for lunch. He'll reach out aggressively to engage you if you walk by the Apress booth, but at lunch he switches off the "selling mode" and just enjoys a good meal.

When travelling with colleagues, don't feel obligated to spend every waking moment with them. There is nothing wrong with telling your coworkers that you need some time alone, and that you'd rather not go bar-hopping until 2:00 a.m. Be up front about why you're doing it. Your colleagues will understand. Some may admit to being in the same boat.

Did I just say to eat lunch with coworkers? I'm going to contradict myself now by giving you permission to eat alone if you need some space. Meals are a great venue in which to engage coworkers and get to know them better. But you don't have to spend each and every meal with them. Take care of yourself first. If you need lunch alone to get your head together for the afternoon, then take it.

## Be Open and Straightforward

Don't try to be something you're not. Don't be afraid to admit that you're not a super-outgoing person. Sometimes when I want to meet someone, I just walk up and tell that person I'd like to meet them. It's a straightforward approach lacking in subtlety, but why not? If you take an interest in people, and attempt to befriend them, most people will respond positively regardless of how clumsy your own efforts are.

## Join Some Organizations

Organizations provide structure. And that structure makes it easier to step outside of one's comfort zone. That's why organizations such as the Professional Association for SQL Server (PASS) are so valuable. They provide a socially safe venue in which to make professional contacts. That the other members are there for the same reason only helps.

Organizations also provide opportunities to lead, and to develop speaking and presentation skills. You can start small. You don't need to make your first talk in a 1,000-seat room. Start with a local user group. Work your way up as your comfort and confidence grow.

Don't make it all about work either. The skills and confidence that you develop working in a community or church organization easily translate into the professional space. If you can work with people to make a difference in your community, then you know you can do the same in your job.

## **Accept That Nothing Is Wrong**

Did we just have this heading? We sure did, but the point is so important it bears repeating. It is a sad truth that society in general tends to smile upon bubbly and outgoing people, and to likewise frown upon those who are quiet and withdrawn. That's an unfortunate and wrongheaded attitude. Do not get sucked into it!

Everything I've said about developing people skills is intended with just that one goal in mind: to develop your people skills. Don't try to make yourself something you are not. Your personality is a big part of what makes you good at your job. Don't fight that goodness. Play to your strengths. At the same time, work to develop your skills in other areas. There is no contradiction in doing both of those things.

## **CONVERSATION STARTERS AND SMALL TALK**

**by Jonathan Gennick**

The art of making small talk in social situations is one that did not come easily or naturally to me. Even in elementary school I would notice that some people could sit down to the lunch table and almost instantly be engaged in conversation with everybody around them. What's more, the conversation would drift from topic to topic in ways that I seldom could follow. Just when a topic got interesting and I was ready to dive in deeply with a comment, suddenly the group was talking about something else. I was always a half-step off.

In business, it's helpful to have a few tricks for starting conversations. One of my favorite ways to begin a conversation is with a question. If you ever catch me manning the Apress booth at a conference like PASS, the question you're most likely to get hit with is, do you have any of our books? It's a logical question that usually leads to some back-and-forth conversation about specific books and what's good and bad about them.

My question is a good conversation starter, but it's sincere too. I truly want to know what readers like and don't like about our books. I want to know which topics resonate with our audience and which do not. I'm hungry for help, really, in doing my job better.

I might ask other questions, too. Sometimes a company name will strike me as interesting. Other times I see names of cities that I've visited or lived in. Questions are a great way to get a conversation rolling. But be sincere. Don't just show interest. Be interested.

---

## Track Your Progress

Remember that checklist? Good, because you need it in order to chart your progress over the coming weeks. As a DBA, a lot of your work is done behind the scenes. In fact, people will often wonder what it is you do all day, since much of your work is never actually seen by the end users. Your checklist will serve you well when you try to show people some of the tangible results that you have been delivering.

No matter how many people you meet and greet in the coming weeks, unless you can provide some evidence of tangible results to your manager and others, people will inevitably wonder what it is you do all day long. If your initial checklist shows that you have twenty-five servers, six of which have data and logs on the C: drive, and two others that have no backups at all, it is going to be easy for you to report later that your twenty-five servers now have backups running and all drives configured properly.

In the end, it is not your effort that people will remember. It is the end result. Make certain you keep good track of your progress so that the facts can help provide people a way to understand exactly what you have been delivering.

## Get Proactive

Now that a lot of your research is complete, you should have a very clear picture about what you are up against in your new environment. As time goes on, you will be establishing your own identity in your new role. Most DBAs will fall into one of two categories: *Mr. Right* or *Mr. Right Now*.

Which one of the following would best describe how you handle your work tasks or how you want to be perceived?

*Mr. Right Now:* This is the guy that is always available to solve any problem. He is very visible, everyone knows his name, and everyone knows what he does. He fixes things. It doesn't matter what time of the day or night, he is always there. When there is a problem—any problem—you call Mr. Right Now and he finds a way to solve that problem. He rarely fails to deliver and everyone praises his hard work and effort. However, whenever he solves a problem, he solves it just enough to move on to the next emergency.

*Mr. Right:* This is the guy that you rarely see. When there is a problem, he pokes his head around the corner, but since there are few problems, people never really know that he is around. In fact, people often ask, "So, what is it you do around here?" But when Mr. Right solves a problem, he solves it for good, making sure that he never has to touch that server again to address that issue. Then, he proactively does that same fix against the rest of his servers before they experience a similar issue, thereby solving problems before they happen.

Which one would you rather be? Would you rather be the guy that is very visible and that the end users love because he is always there to fix things? Or would you rather be the guy that people rarely see and have no idea what you do for a living?

How about you answer this question: Which one (Mr. Right or Mr. Right Now) would you consider to be a junior DBA, and which a senior DBA?

**TIP:** Being able to fix things does not make you a senior DBA. You become a senior DBA by making certain that things do not break in the first place.

While Mr. Right Now may be getting slaps on the back from the end users for all of his efforts, the ultimate question must be asked: is he fixing problems that are being caused by his fixing of an earlier problem?

There are two reasons for someone to be a Mr. Right Now. The first is that there are far too many problems at the onset of their tenure and they are going to be fighting a long battle to get all of the fires under control. The second reason is that they are a junior-level administrator who knows just enough to fix the current issue.

Mr. Right, however, is the guy that not only puts out the fires as they happen, but also has the knowledge and experience to put controls into place to make certain that the problems are less likely to happen again. So, they not only know enough to fix the current issue, but *they know enough to solve the root cause of the problem*. The solutions they put in place do not complicate the environment; they help to lessen their workload.

As you start your new role, there is a very good chance that you will need to be Mr. Right Now for an extended period of time, even if you are at a senior level. It could be the case that there is way more work than one person can handle. Of course you have your checklist and are well organized, so you know where to focus your efforts first. But as you go down that checklist, you need to keep thinking about solutions, as opposed to a quick fix.

Over time, you really want to be Mr. Right, and have your environment configured in such a way that it practically heals itself. And do not despair if you feel that you are losing out on an opportunity to show people how valuable you are to the company. Indeed, as you continue to configure your environment and are proactive in heading off potential trouble, you will find you have more time to be visible in different ways.

**TIP:** Work smarter, not harder. Find ways to be proactive in solving issues.

Would you rather be stuck in a server room all weekend or be helping someone learn to write better T-SQL code? Would you rather be rebooting boxes as quickly as possible during the business day as you troubleshoot some issue, or would you rather be helping someone in finance understand how to configure and build their own reports using SSRS? Trust me, there are plenty of ways for you to show your value without having to run around the place as if it were on fire.

Back in college we were always told to study smarter, not harder. The same principal applies here: work smarter, not harder.



## Some Basics

You have your checklist in hand, you are meeting with people every day, and you are slowly bringing your environment under control. As you meet more and more people, you find that they appear to be speaking in code; their use of acronyms is astounding. Even more astounding is that they expect you to understand the acronyms right from the start.

This chapter will cover some of the technical basics you will be expected to be familiar with. After reading this chapter you should be more familiar with the acronyms you might be hearing frequently, such as RAID, SAN, HA, and DR, and even what DBA means to some people.

**TIP:** No one person knows everything. Do not be afraid to tell someone that you have not heard of something that they are talking about. The best DBAs ask questions, even basic questions, to quench their thirst for knowledge.

In this chapter we will discuss the following:

1. RAID
2. Storage area networks (SANs)
3. High availability options
4. Disaster recovery (DR)
5. Why networks are like bathrooms
6. The meaning of “DBA”
7. Why you get all the blame
8. How to politely be right

## Introduction to RAID

Probably the most important hardware concept you need to be familiar with is RAID, which stands for *redundant array of inexpensive disks*. RAID is essentially about different ways of storing data twice, or more than twice, on different disks. I'm oversimplifying just a bit, but a core goal of RAID is to let you survive losing a disk without also losing any of the data that was on that disk.

The redundancy part of RAID is what allows us to survive the loss of a drive. There are different ways to provide redundancy, and I'll cover those in this chapter.

Understanding different RAID configurations is going to save your bacon on more than one occasion. Different RAID configurations are called for in different situations. They can be used to improve performance as well as protect against disaster.

### Why Is RAID So Important?

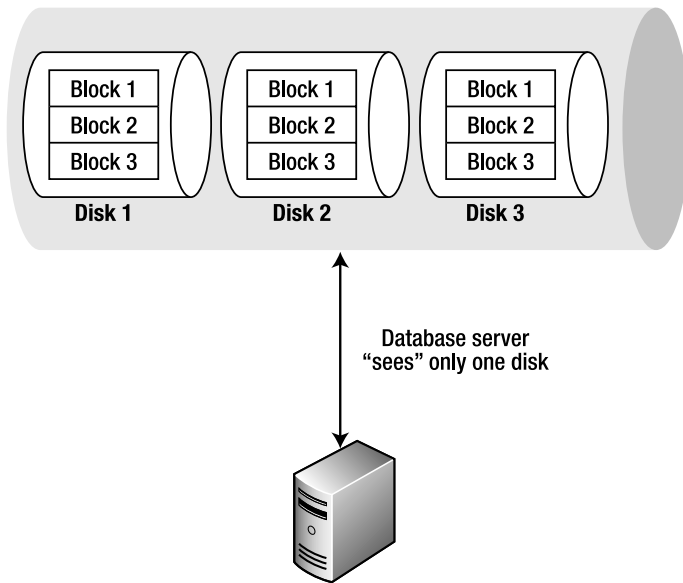
Consider most home computer systems that are bought from a national superstore chain. They come with more than enough memory, have more than enough processing power, and have lots of disk space. The one thing they don't come with is a backup (or external) drive. And why would you want one of those? Users need those to protect their important files, such as e-mails, pictures of the cats, and all their favorite web links.

Many homeowners don't think about backups. They go on merrily for years, adding their financial data, their photographs, and their collections of e-mail and iTunes music to their hard drives. Then one day they learn the hard way about hard drive failure. Their hard drive fails, their computer won't boot, and they lose everything.

What about servers—do they come with extra disks? Usually, yes, servers are ordered with extra disks. Depending on the model of the server, you will have a different number of options and disks available. The primary reason for extra disks, besides the need for disk capacity, is to protect yourself through the utilization of RAID in the event of a hard disk failure.

For example, Figure 3-1 shows one possible RAID configuration. In Figure 3-1, three hard drives are tied together to appear as one. The same data gets written to all three drives. Because the three drives are mirror images of each other, we can trash one of the drives and still have access to our data on the other two. We can even trash two drives, leaving just one good one.

**CAUTION:** Do not rely solely on raid to avert disasters. RAID is not backup. When a user deletes a record, it's instantly deleted on all drives. You still need backups to recover from small problems like these, as well as major disasters like server failures, datacenter outages, and natural disasters.



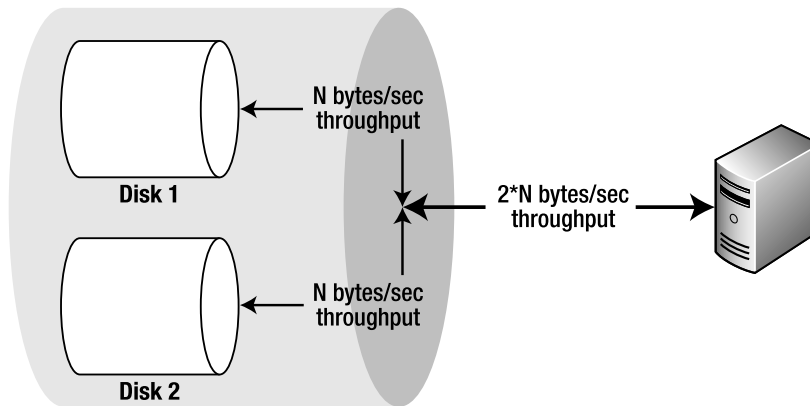
**Figure 3–1.** One path to redundancy is writing the same data to many drives.

## RAID for Performance

You can also utilize RAID to improve performance. Knowing how to do that can be helpful when trying to performance tune queries. It will also serve you well when you go through your initial checklist and start to gather baseline information on the configuration of your servers.

SQL Server is very much an input/output (I/O)–intensive system. That means that the faster you can read and write data to and from your disks, the faster your response times will be. You can use RAID to increase your I/O throughput by taking advantage of *striping* your data across several disks at once.

For example, a traditional hard disk has a little arm inside of it that moves back and forth every time you want to read and write data. Well, let's say you had to do 100 read and 100 write operations, and each one took 1/100th of a second. How long would that take on one disk? Two seconds. Now, let's say you configured RAID as shown in Figure 3–2.



**Figure 3–2.** A RAID configuration to improve performance by striping

Now you have two little arms moving back and forth across two disks at the same time. You would still have your two hundred operations to perform, but they would be done on two drives simultaneously. Each individual drive would get half the work. Your overall time would be reduced from two seconds to one second.

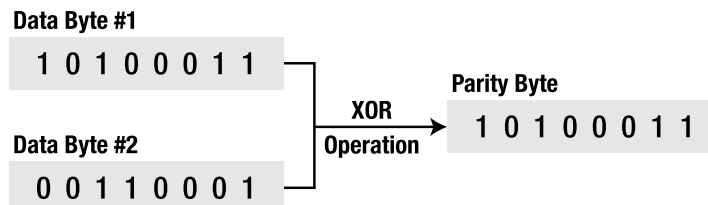
Another method for improving performance would be caching. Most RAID controllers allow for some amount of data to be stored in cache. The idea would be to cache as many nonsequential requests as possible in order to batch them together and send them as a sequential request to the hard drives.

## RAID for Fault Tolerance

RAID uses two methods to provide for fault tolerance: mirroring and parity. Mirroring is exactly what it sounds like: you have two (or more) copies of your data on two (or more) different disks. the scenario in Figure 3–1 is a mirroring scenario.

**NOTE:** Mirroring is most commonly thought of in terms of two drives mirroring each other. However, there is nothing stopping you from mirroring across more than two drives.

*Parity* occurs when you add extra data called *parity data* to one of your disks, which can then be used to reconstruct your real data should one disk fail. The advantage of the parity approach is that you save on disk space. Figure 3–3 shows how one byte of parity data can protect two bytes of real data.



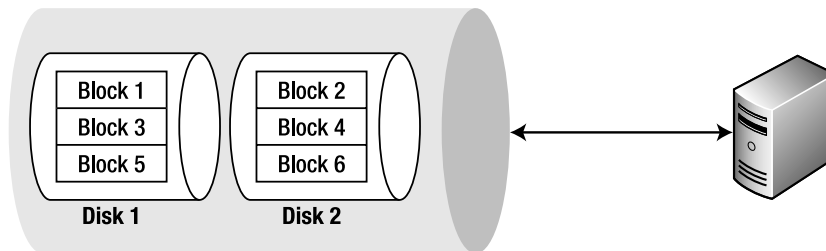
**Figure 3–3.** One byte of parity data can protect two bytes of real data.

The parity approach to RAID is a good thing, unless you are concerned about performance. There's a negative impact on performance, because writing that extra bit of parity information each and every time will add additional overhead.

Let's examine several different RAID levels that use different combinations of mirroring and parity to achieve protection and speed. I don't discuss every possible RAID level, just the ones you are likely to encounter as a DBA.

## RAID 0

This level is commonly called *disk striping*, and the result is often referred to as a *striped set of disks*. Your data gets divided into chunks (or blocks) and distributed across all the disks in the array in a fixed order. Figure 3-4 illustrates the approach.



**Figure 3-4.** An example of a RAID 0 configuration

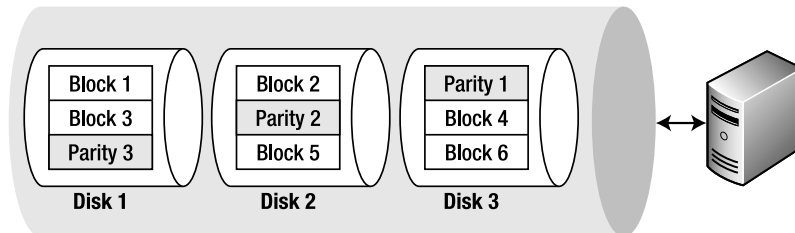
RAID 0 will increase performance as the operations can all be performed at the same time independently from one another, as noted earlier. The drawback is that any piece of data is on only one disk, because the data was divided across all of the disks evenly. In the event of a drive failure, all of the data that was on that disk is permanently lost. In fact, it's worse: lose one drive and you've effectively lost all data in the array. The only way to recover from a drive failure in a RAID 0 array is to restore the entire array from the last backup. Therefore, RAID 0 does not make a good choice for SQL Server storage. It is worth noting that RAID 0 does not provide fault tolerance, nor does it provide redundancy.

## RAID 1

This level is more commonly called mirroring. It's the scenario illustrated earlier in Figure 3-1. Your data is simply written to at least two different places at the same time, creating one or more copies of your data on a separate disk. In the event of a drive failure, the array is still completely available because at least one other copy of the drive is still online and functional.

## RAID 5

This level is also known as striping with parity. It is very similar to RAID 0, with one additional disk needed to hold the parity bit that gets striped across all disks. Figure 3–5 illustrates an example configuration. Without a doubt, RAID 5 is a favorite RAID level, and I have seen it used by many administrators.



**Figure 3–5.** Example RAID 5 configuration

The reason for RAID 5 being so well liked is that if one disk fails, you simply have to add in a new disk (before a second disk fails; otherwise, you lose *everything*) and the array will rebuild itself. Sounds nice, huh? While RAID 5 can offer better performance than RAID 1, the overhead for that parity bit means it will not perform as well as RAID 0. But you do get fault tolerance with RAID 5, and that is why so many administrators love using RAID 5 over anything else.

### BATTLE AGAINST ANY RAID 5?

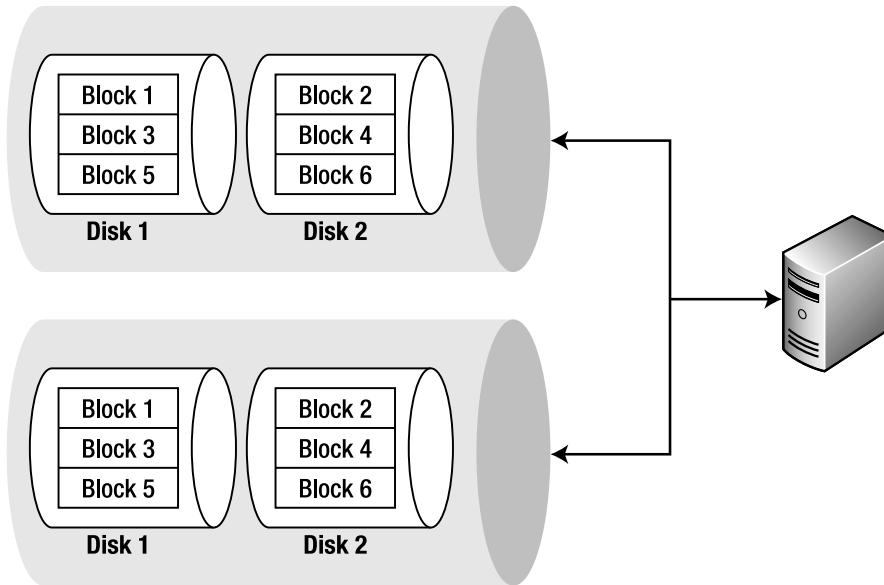
**by Jonathan Gennick**

Love for RAID 5 isn't universal. Frustrated DBA James Morle created a tongue-in-cheek group known as the BAARF Party in 2003. The goal of The BAARF Party is to battle against any and all RAID 5. You can visit the web site and read the arguments at the following URL:  
<http://miracleas.com/BAARF/BAARF2.html>.

You may or may not agree with the BAARF Party—so far, I've chosen to keep an open mind—but some of the papers hosted on the BAARF Party's site are well thought out, and are worth the time to read.

## RAID 1+0

This level is commonly known as mirroring with striping, and is sometimes referred to as RAID 10 or even RAID 1/0. You would configure a striped set (RAID 0) and then mirror them (RAID 1) onto a similar striped set. Figure 3–6 shows how that might look.



**Figure 3–6.** RAID 1+0 mirroring a striped set of drives

The RAID 1+0 level offers the highest performance of any RAID level. It provides fault tolerance, too. And it costs more money, because you need twice as many disks.

## Which RAID Level Do You Want?

It depends, of course. Do you want your data to stay online when one drive fails? Most people do, but perhaps you do not. In that case, RAID 0 is for you! No need to worry about parity or mirroring, just configure your disks as one big striped set and be done with it.

What about performance and costs? Most people care about them as well, and chances are you care, too. But when it comes to SQL Server, you will find that no one RAID level is going to suit your every need. Wait a minute . . . do you know what those needs are yet? Well, let's discuss them in their simplest view.

## Your Needs

You really only have three basic needs:

- Placement of your database data files
- Placement of your transaction log files
- Placement of your tempdb database files
- Placement of your database backups

OK, that's four needs, but that's it, really. Your data files should be placed on one drive or set of disks (array), your transaction log files should be placed on a separate drive or

set of disks, your tempdb files should be placed onto a third set, and your backups should be stored either on a fourth set or on a different server altogether (which is often recommended).

Each item in the preceding list leads to a different set of needs. In a perfect world with bottomless budgets and hardware designed specifically for database servers, you would be able to configure everything with RAID 10. Your server's operating system files would go on one array, and your pagefile would go on a different array, as well as your data, your transaction logs, and your tempdb files. That would be five different arrays, configured as RAID 10, meaning you would need (at a minimum) twenty hard drives. In case you are wondering, that is a lot of drives. Most servers do not come with twenty hard drive bays.

But do you really need all of that? I have thousands of databases under my care right now, and 99 percent of them would perform just fine with RAID 5 for data, transaction logs, and tempdb. But there are a handful of systems that do need some extra care and attention. And when the time arises for you to help performance tune a system, it is going to benefit you if you just stick to the basics.

**NOTE:** Tom's comment here about many systems performing just fine on RAID 5 arrays fits my own experience. It's why I haven't been able to quite bring myself to drink the BAARF Kool-Aid. – Jonathan Gennick

The biggest thing for you to recognize when troubleshooting is where you are feeling the pressure. Is it in your tempdb? Then put your tempdb on a RAID 10, if possible. Is it your transaction logs? Put your transaction logs onto a different RAID 10 set, if possible. If not, then go with a RAID 1 for your transaction logs. Is it your data files? Your data should also go onto yet a different set of RAID 10, but if that is not possible, then go with RAID 5 for your data.

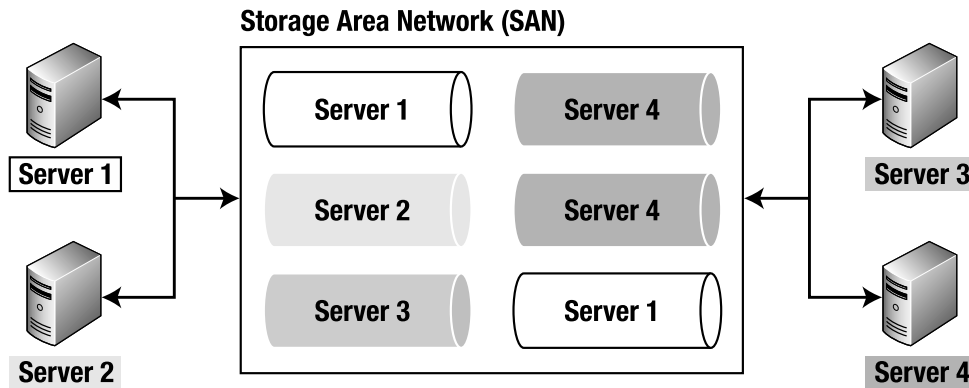
The bottom line is this: know the difference between these RAID levels as well as which ones you want to use. That knowledge will be a benefit when you are helping to architect the design of any system.

## The ABCs of SANs for DBAs

By now you should have the concept of RAID safely under your belt to the point that you could sit down with a member of your server team and talk a bit of shop. If you did, you will probably find that most of your servers will not have enough drive bays for all the different flavors of RAID that you might desire. You may even find yourself asking for different RAID configurations that result in conversations around external storage . . . and then someone, somewhere, will mention the following: SAN.

SAN stands for *storage area network*, and will either be the most wonderful thing you can possibly imagine (high probability) or a nightmare (not likely, but possible). The concept behind a SAN is quite simple: bundle together hundreds of physical disks,

partition them out in logical units (called LUNs), and connect your servers to the LUNs for their storage. Figure 3–7 shows how one SAN can serve many servers.



**Figure 3–7.** Drives in a SAN can be assigned to many servers as needed.

This way, if you needed to add additional storage to a server, it would be as easy as assigning some additional LUNs. In fact, most of the time, all your SAN administrator is going to ask you is, “How much space do you need now, Bob?” No need to swap out smaller drives for larger ones and let your RAID controller rebuild your array. Sounds wonderful, right?

Maybe.

Let’s imagine for a moment that you are the SAN administrator. You have hundreds of disks bundled together into LUNs. Which RAID level would you use for all these disks? If you answered RAID 5 for its cost-effectiveness and large storage volume, then you may have a future in SAN administration when your DBA days are over.

Chances are your SAN is indeed configured with RAID 5. If you recall, RAID 5 is not always ideal for a database server. Don’t panic: when used correctly in a SAN, RAID 5 arrays may not be ideal, but they can still be good enough.

## Why RAID 5 Might Work for You

The first reason not to fear is that SAN controllers have built-in cache: memory that serves as a buffer between your server and its storage arrays. Whenever SQL Server issues writes to your drives, they’re considered committed as soon as the data enters the SAN controller’s cache. When configured correctly, that cache is faster than any drive arrays, whether they’re RAID 5 or RAID 10. While your SAN may be at RAID 5, your data is not going to disk directly; it gets dumped into cache first and then goes to disk. Your queries do not wait for it to get to disk; they think it is already there when in fact it is actually still in cache. You see no performance issues until such time as you would actually fill your cache, in which case you have performance issues *everywhere*, not just on one server.

The second reason you should not panic is because you may have not seen any actual performance problem yet. Just because your SAN may be at RAID 5 is not enough reason to raise the hairs on your back. No sir, you had better be able to provide some proof that you are seeing performance issues that could be related to SAN performance.

## The Lost Art of Benchmarking

The only way to find out for certain if storage is a bottleneck before a server goes into production is to do some benchmarking. One thing I find common in most shops these days is that fact that benchmarking is far from a common practice. People tend to slap things together, throw them into production, and wait to see if anyone screams. If you want to make the leap to being Mr. Right, then you will be wise to benchmark some of the more important processes in your shop. That way, when someone says, “This is slow,” you can quickly reference the latest benchmark to determine if it really is slow or if it is just perception.

The first question you need to ask your SAN administrator when troubleshooting possible SAN performance issues is, “What is the expected I/O throughput for our SAN?” Get whatever details you can, and then go back to your cube and start gathering your own metrics. Then compare to see if the expected throughput is close enough to the observed. If not, now you can go back to the SAN administrator with some actual facts.

Believe me, that will get you a lot more traction than making a blanket statement such as, “We would like to have some space on the SAN carved out in RAID 10 for our tempdb.” Unless you have proof that there is a performance issue, your SAN administrator is not going to be compelled to start carving out different RAID levels, even if you know what works best for SQL Server.

**TIP:** The more benchmarking and testing you perform, the more control you have over your environment.

Your SAN could indeed be a performance bottleneck, and it may be very difficult for you to troubleshoot. Much in the same way that people will blame a database server because it is a big black box they do not understand, I have seen some DBAs be frustrated because their SAN is also a big box that they do not understand. The best advice I could give to you is to think of your SAN as being a constant. It should serve up a standard I/O throughput. It should allow for you to dynamically add space to a server when needed. It should be the same for each server. And that means you can perform some benchmarks on different servers, assume that the SAN performance should be equal, and take your results to the server team to discuss the discrepancies. It may indeed be the SAN, but it could also be something else.

## NOT A CONSTANT, BUT A VARIABLE!

**by Brent Ozar**

Hmm—I tell people the opposite. Assume the SAN is a variable. The SAN team can change it without warning. You can get new neighbors that share your drives, you can have switch failures, and so forth. All these things can change your performance without warning. Perhaps the point is to test to ensure that your SAN remains a constant like it is supposed to, and so that you are alerted when it ceases to be a constant due to some unknown change by an unthinking SAN administrator.

---

### **It's All About the Spindles, Baby**

If the SAN creates a performance bottleneck, then one discussion you will want to have is about the spindles. A *spindle* is a SAN administrator's slang for an individual drive that is combined together with other drives to make arrays. For example, you could have ten disks in an array, build five logical volumes on those ten disks, and each volume would have ten spindles. But if two or more of those volumes are put under stress, then all ten spindles are under stress. In an ideal configuration, your SAN would have as many spindles as possible and a minimal chance for resource contention.

Want to have some fun? Go and ask your SAN administrator this question: "If my current LUNs were converted into physical disks, how many spindles would I have allocated?" If the number is low, then you may have cause for concern. Again, that is why I stress the importance of benchmarking.

While SANs may be great, it is not all unicorns and rainbows. One of the drawbacks to SAN storage is that it can be unpredictable. For example, what if you are doing all of your database backups at the same time of day and week on your SAN? Chances are you will flood the cache and cause issues everywhere. Now imagine that it is not your backup process but an end user that decides to bulk load 20GB of data that causes performance issues on other servers that are sharing disks within the same LUNs.

One last thing to note is that your SAN may use replication, automatically sending all data from one SAN to another SAN in a different city, for high availability and/or DR purposes. Even if that is the case, you still need to be taking regular database backups. Do not let anyone try to convince you that SAN replication eliminates the need for regular database backups. About the only thing sillier would be if they said that RAID 5 eliminated the need for a database backup. It is not true, and it will only take one disaster (which will be your last disaster, by the way) for everyone to understand why the database backups were necessary. Furthermore, tools like SAN replication and RAID should still only be one part of your high availability strategy.

## High Availability Options

You know enough basics on storage, but you have to worry about more than just storage to make certain your systems are always available. Keeping a database online despite problems like server crashes, OS freezes, and hardware failures is called *high availability (HA)*. As the DBA, you are going to be held responsible for making certain your database servers are always available, *even if you have nothing to do with the apparent outage*. You read that correctly. Your end users see you as the face of availability. Should the server be down because someone decided to unrack it by mistake, it will be your phone that rings, not the guy in the server room using your box as a new paperweight.

There are four major groups of HA options that you should be familiar with when having group discussions. They are clustering, log shipping, replication (both SQL and SAN), and database mirroring. Each one has advantages and disadvantages when compared to the others. The best advice I could give you is this: *there is nothing permanent except change*. What I mean by that is that you can spend a lot of time and money designing the perfect HA solution for your shop only to find that in 18 months your design is no longer adequate. Therefore it is best to keep in mind all available options and how to be flexible enough to move things around, if necessary.

**TIP:** HA is HA and DR is DR. They are very different things. Remember that.

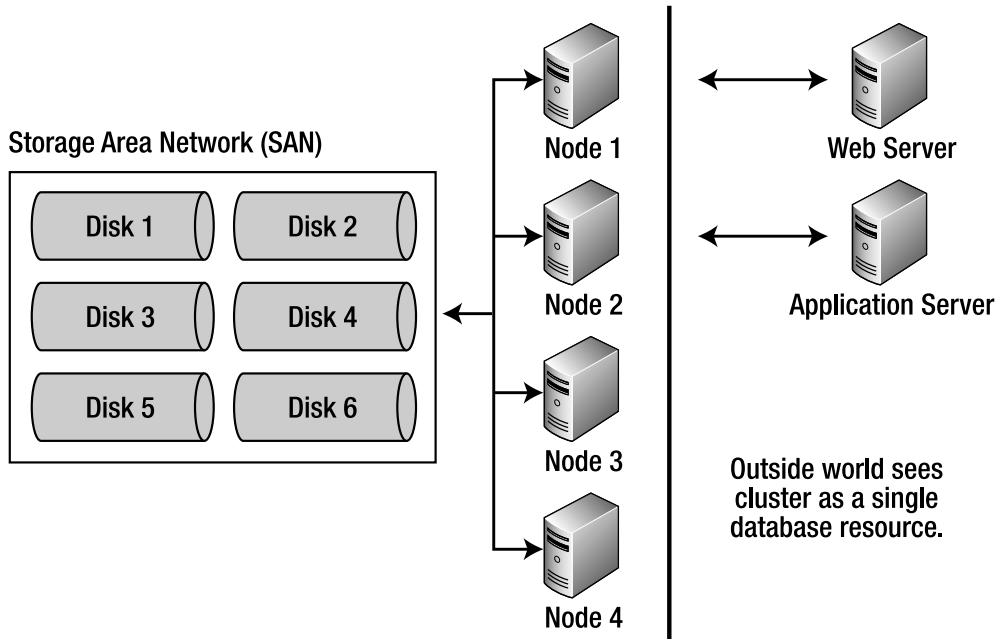
One last note before we continue: *HA does not mean the same thing as DR*. You systems can be highly available, but have no ability to recover from a disaster. Likewise, your system can be configured to recover from a disaster, but during recovery your systems may not be available. Please keep this in mind during your conversations. At some point you will run into a person who feels that HA is more important than DR. They will be wrong, and you will do your best to help them understand that your jobs depend more on recovering from a disaster than availability.

The topics we discuss next are mostly with an eye toward HA options, but some of them can double as both HA and DR solutions. You will need to decide which ones work best for your shop.

## Clustering

One of the more expensive options with physical servers is the notion of *clusters*. In order to build a cluster, you first build your primary server (or *node*), which has a defined set of shared disks (called a *resource group*). Sharing disks between multiple nodes does not require a SAN, but having a SAN can make your life a little easier. You then add nodes to the cluster, with each node being defined to that same resource group. The entire cluster appears on your network as a single server instance, but it is really made up of multiple server instances. You will hear talk of primary nodes, secondary nodes, or failover nodes; all that talk amounts to fancy words for a server instance.

Figure 3–8 provides an example of a four-node cluster. The arrangement may at first glance appear similar to that in Figure 3–7. The difference is that all four nodes in the cluster share access to the same set of database disks. Furthermore, the outside world sees the cluster as a single database instance.



**Figure 3–8.** A four-node cluster built around shared disks

The reason clusters can be so expensive is because they typically require a lot of additional hardware like a SAN, and each node in the cluster needs to have very similar hardware. However, with virtualization you can look to use clusters with virtual servers and reduce your costs. You can even find some third-party software applications that help you to failover your virtual instances during times of peak usage, which will (in theory) help to increase your availability (often called *uptime*).

If this all sounds too good to be true, that's because it is too good to be true. There simply must be a downside to clustering, even on virtual servers, right? Absolutely.

Many times I have had the opportunity to load software onto a cluster. And despite the vendor assuring me that I only need to load their software onto the primary node, I will often find their software not working should the cluster fail over to a different node. Historically you will find horror stories relating to the application of SQL hotfixes and service packs to the primary node, and their failure to work on the other nodes. And while each version of SQL and server OS helps to reduce those headaches, you should be aware that they can still happen.

The other downside to the use of clusters is the increase in the number of servers that your shop needs to administer. While having three nodes sounds like a great way to achieve HA, it may not sound so great to the people that need to administer those

instances and apply things like service packs. Believe me, administrators enjoy having a weekend off every now and then, same as everyone else.

Need more downsides to clusters? How about the amount of time it takes for the failover to happen? Depending on your configuration it can take less than a minute to several minutes. And what about those transactions and connections in flight during the failover? Will they see an interruption in service? Probably, so your application had better be able to handle such an event; otherwise, your phone will ring off the hook the minute people perceive there is an issue.

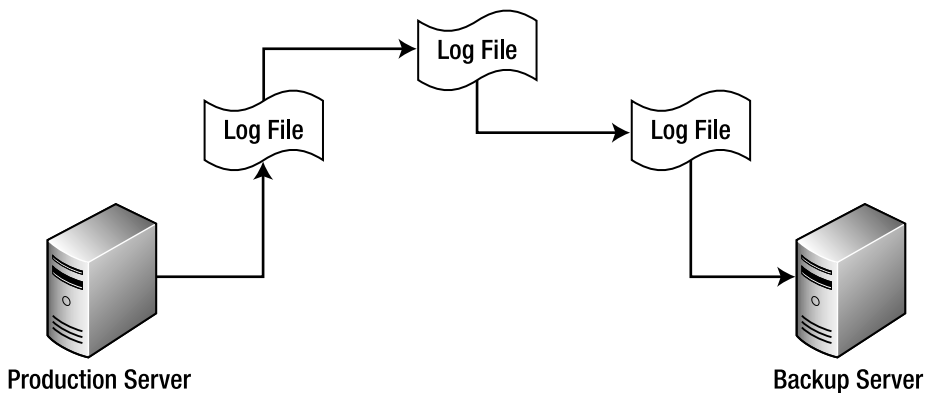
Need more downside? Well, how about that clusters do not protect you from a disk failure? If you are going about setting up clusters and think that they are a good DR strategy, then you are looking Mr. Murphy right in the face and laughing at his laws.

## Log Shipping

Log shipping is a traditional method of HA, allowing for you to maintain your data in two (or more) places at the same time. The idea is quite simple:

1. Do a backup of your transaction log on your production server.
2. Copy that backup file to another server.
3. Restore that backup file to another server.

And there you go—you now have a database (see Figure 3–9) that should look just like the database that went offline for whatever reason. Well, close enough to the database that went offline. I mean, how often are you doing those transaction log backups? Every minute? Five minutes? Ten? Fifteen?



**Figure 3–9.** A log-shipping scenario

Just how long would you be able to go without data? Would it be acceptable to tell your end users that they would be without their data for the past ten or fifteen minutes? And if you did transaction log backups every minute, are you certain they would complete in that minute? Depending on the usage and activity, you could find your transaction logs taking longer than one minute to complete their backup, meaning it would take some

time to do the file copy, and then longer than a minute to do the restore. What does that mean for the jobs that keep trying to do the backups and restores every minute?

They fall behind, that's what it means. And if they fall too far behind, you will find yourself having to reinitialize everything from scratch. And depending upon the size of the files and databases involved, that could make for a long afternoon and evening.

But for a lot of people, log shipping is the way to go. Log shipping doesn't require a SAN or identical hardware on each server involved. A lot of vendors will recommend log shipping as a way to achieve HA with a fairly low amount of overhead. You do not necessarily save on costs when compared to clusters, but you do save on overall administration costs, because things like service packs and hotfixes are easier to implement, as well as installation of vendor software.

Be mindful that log shipping requires manual intervention. Unlike clustering, which can provide you with an automated failover, log shipping requires a human to get involved and to manually get the backup server up and running as a production server.

## Replication

You will often be involved in conversations about replication. My first word of caution is to make certain that everyone understand what type of replication is being discussed. You have two main types of replication to be concerned with: SQL replication and SAN replication. They are not the same thing—in fact they are very different, and can exist together or separately depending on your needs.

SAN replication is simply a way your shop has configured the SAN to accommodate for moving bits of data from point A to point B. This is done primarily for HA purposes, but you will find someone who thinks it is also being done for DR purposes. Depending on how the SAN replication is configured, it most certainly is an HA option and is possibly a DR option. For example, if you have two sites and are doing SAN replication from site A to site B, then you certainly have an HA scenario configured. However, when an event happens that affects both site A and site B, both SANs will go down. Your HA is clearly going to be out of the water, and so will your DR unless you are also replicating to a third site.

I can hear people now saying that they can recover from tape, so technically they have a DR solution in place. And I will tell them the same thing I am telling you right now: *then your tape backups are your DR solution, not your SAN replication*. I hope that helps clear up the confusion.

As for SQL replication, you get your choice of flavors. Each one is unique from the rest, and each appropriate according to your particular needs. You should have a cursory knowledge of the different types and when it might make sense to deploy. Your environment and the nature of the requirements you are given will be your primary decision factors when selecting a mode of SQL replication.

Before we even get started, a quick overview of replication terminology is in order. Many of the terms used (publisher, subscriber, distributor) are used in the publishing industry,

and thus the publishing industry is often used to describe replication concepts. So the term *publisher* means exactly what you would expect: someone who publishes information, like a newspaper. A *distributor* would be someone who distributes the newspapers to the people who are *subscribers*. Simple enough? Great, now let us have a look at the different types of SQL replication.

## Transactional Replication

Transactional replication is when transactions committed at the publisher are then replicated and committed at the subscriber. This makes no sense in terms of newspapers or magazines, so think of this as if you were reading the online version of your local newspaper. If someone makes a change to the articles on the web site, you will see the changes the next time you refresh your browser.

When would you want to use this type of replication? Often the answer has to do with systems that have a high volume of transactions and the need for low latency, and the fact that one of your subscribers is a not a Microsoft SQL Server. Also, the subscribers in a transactional replication topology are usually considered to be read-only, but it is possible to allow for subscribers to make changes to data that are then pushed back to the publisher.

Transactional replication makes sense when you have a server that is not SQL Server, or when you have the need for more than one subscriber. If you only have one subscriber and you only have SQL Server database servers, then you should go with database mirroring instead, as discussed later in the chapter. But take into consideration the scope of the changes you want replicated. With database mirroring, all transactions are mirrored and thus replicated, but with transactional replication, only those transactions marked for replication will be replicated.

## Merge Replication

This might be the biggest beast of all replication technologies. Changes at both the publisher and subscribers are merged together through the use of triggers. Think of a newspaper needing to publish the evening news that has ten reporters in the field updating stories. Now let's say that two reporters are covering the same story and try to publish different versions. Which version will get written? How would you resolve the conflicts?

Merge replication is not for the faint of heart. When this replication breaks, you had better have a method in place to put it back together again. There can be a lot of administrative overhead with merge replication. So why would anyone want to use this?

Most often I see it used when you have disconnected users from a central publisher. Often times you will see the example of having members of a sales team in the field visiting clients. Their mobile devices and laptops are not always connected, but they may be gathering data. When they do connect to the network, they want their data to update accordingly. Conflicts can (and will) arise if another sales representative tries to update the same data.

This replication topology should only be used if you have the need for many disconnected clients to connect back to the main office. If that is a requirement for your shop, then merge replication may be the answer. I would rather eat shards of broken glass than implement merge replication, but that's just me. Some people make a great living eating broken glass at carnivals. You should choose your own path in life.

## Snapshot Replication

Snapshot replication requires less administrative overhead than the other two forms of replication. Just as it sounds, a snapshot of the database is taken and replicated as a whole to a subscriber. This is very useful when you have data that changes infrequently. Perhaps you might need to update your list of available products to your sales team a few times a year. When the sales representatives connect to your network, a snapshot of your data can be pushed out to them so they see all the new products.

Be careful using snapshot replication on very large data sets. You can easily cause network issues if you were to decide to push a 100GB snapshot out to 100 subscribers in the middle of the day.

If you do not have the multi-server platform requirements that would force you into transactional replication, and if you do not want the hassles of merge replication, then snapshot replication may be the right solution for your shop. While snapshot replication is the easiest type to manage, there's another method of moving data around that requires even less work and time.

## Database Mirroring

Recently I was asked to explain database mirroring in the simplest terms possible. About the only thing I could think to say was the following story:

Long ago, in ancient times, in the time before time, people were asked to make copies of their work. This led to some very interesting inventions throughout history, most notably the printing press, the polygraph, and ultimately the modern day copy machines that became a big hit at holiday office parties. But as society progressed to the point where deliverables were measured in fractions of a second as opposed to days or weeks, these archaic methods were no longer good enough. More and more demand was placed upon people to deliver the same piece of information into many different places all at the same time without any loss of service.

Soon a great battle took place between two giant beasts named Cluster and Log Shipping, who had easily beaten down the inferior SQL Replication in a pay-per-view event two months prior. Cluster and Log Shipping battled for years, neither giving an inch to the other, until such time that they were both badly beaten, weakened, and barely able to stand.

At the time of SQL Server 2005's release, a great hero rose from the ashes of Cluster and the ruins of Log Shipping. The hero's name was Database Mirroring, and promised that everything was going to be different. Database Mirroring promised real-time

synchronization of your data without the hassle of log shipping. Database Mirroring promised faster failover without the need for a cluster. In short, Database Mirroring promised the best of all other technologies, with none of the shortcomings, and even brought along a witness to attest to its abilities.

And there you have it. Database mirroring looks to give you the very best in HA by combining all of the good in other tools and none of the bad. However, there is one tiny little thing that makes database mirroring not as perfect as people may think: you only get one mirror.

Oh sure, you can go to a clothing store and stand in those fancy mirrors that let you see parts of your body you didn't even know existed. Well, database mirroring gives you one mirror, no more. So, if you have the need to mirror your data to more than one location, then you need a different solution. You could, if so desired, configure your mirror and also configure log shipping to other servers. But the one mirror itself will not provide that option for multiple servers.

Other than that one issue, database mirroring is a fine alternative to clustering, log shipping, and replication. It truly does offer a wonderful HA solution that will satisfy the HA requirements for many shops. In fact, asynchronous mirroring *could* be used for DR purposes, which is important as you begin to plan for what to do when disaster strikes.

## Disaster Recovery Planning

DR planning is essential; not only for your job security, but for the entire company's existence. You can do a quick search on the Internet regarding the number of companies that have gone out of business simply due to their failure to plan for a disaster. Most commonly you will find companies that simply did not have any idea about the need to recover their data in a timely fashion. I see a lot of cases where companies go under because of something boneheaded, such as relying on RAID 5 as the only means for preventing a disaster.

**TIP:** When it comes to DR, failure to plan is planning to fail.

You simply must plan for a disaster. No scenario is out of the question. Whenever I see a new design, I usually point to a particular section and ask, "What if that server disappears? What then?" Asking those types of questions are the only way to effectively plan for a disaster.

One of the better methods I have come across when it comes to DR planning is to simply ask the question, what will we do if . . . ? You ask the question until you reach a point where there is nothing left that you could effectively plan for.

Here is an example:

Q: What will we do if that server melts?

A: We have SAN replication and would just need to boot up the DR server.

Q: What will we do if the SAN replication stops working?

A: We would recover from tape to restore the DR server to last night.

Q: What will we do if the tapes are not good?

A: We would recover from the last known good tapes.

Q: What will we do if the DR server does not boot in a timely manner?

A: We could build another server.

Q: What will we do if there is no other server available to build?

A: We could go buy another server.

Q: What will we do if we cannot buy another server?

A: We could take the test server and use that one as production for the time being.

Now, compare that series of questions and answers to this type of DR planning:

Q: What will we do if that server melts?

A: We have the disks in a RAID 5 array; the chances of two disks failing is minimal.

Q: What will we do if two disks fail?

A: There is little chance of that happening.

Q: Um . . . OK . . . so, what will we do if that does happen?

A: Don't worry about it; it won't happen.

Sorry, but that does not leave me warm and fuzzy inside, nor should it make you feel comfortable either. Another good question to ask yourself and others would be, what are we trying to protect against? When it comes to DR, you can never ask enough questions, and never settle for an answer similar to "You don't have to worry about that, it will never happen."

## GET OUT OF JAIL FREE

**by Sylvester Carstarphen**

One thing I like to get is written documentation—what I refer to as my "get out of jail free" card—when management insists a particular scenario will never happen.

---

## DR Testing

If your company is wise, then it will be conducting regular DR tests. In some cases, regular DR testing is required by internal and external auditors in order for your company to retain a specific level of compliance. You should be aware of the DR tests and look to actively participate so that you can learn from each and every test experience.

Each DR test exercise gives you the opportunity to prepare your own DR test plans. These details are going to be quite valuable to you should a disaster happen. As you participate in the test exercise, you will inevitably come across some hidden nuance of your systems. For example, you may find a handful of servers that need to be booted in a particular order.

**TIP:** Practice makes perfect. Give yourself an opportunity to succeed during a crisis by practicing the steps needed to recover.

It is also important that you have very clearly defined recovery times for your systems. While you may be pleased to know that you can have your DR servers functional in 45 minutes, that may be an unacceptable amount of time to your end users. Make certain that time objectives are clearly stated and that your actual recovery times are clearly recorded for everyone to see. Otherwise, it will be difficult to know during a real disaster whether things are going well (but it will be painfully obvious if things are going poorly).

This reference will also be handy when you are recovering from disaster. If you know your testing took 30 minutes to recover, then you can reference your earlier testing and notify your customers as to when they can expect their systems to be back online.

## Carnegie Hall

There is an old joke about a young man asking for directions in New York City. He stops an older man and asks the question, “How do I get to Carnegie Hall?” The old man replies, “Practice, practice, practice.” When I was coaching basketball, I would always tell my team to practice “game shots” from “game spots” at “game speed.” In other words, you practice how you play, which should be all out, all the time. If you practice at half speed, you end up playing at half speed.

All of that holds true for DR preparation. You need to practice and keep your skills sharp. When was the last time you tried to recover the master database? Or, better yet, have you ever tried to recover the master database? Believe me, when the pressure is on during a disaster event, the last thing you want to be doing is fumbling around with manuals and pounding your fist against your monitor because you do not know how to get your instance started.

Practice recovering the master database. Try recovering the msdb database sometime as well. Spot check your backup files and do a restore to make certain you are familiar with the process. Pay particular attention to your important systems. If you do nothing else right, recovering the important systems without any problems just may be enough to save your job, so practice with those and make certain you are familiar with any subtleties that they may contain.

You simply cannot practice enough when it comes to a disaster. As a DBA, your primary responsibility is to recover the data in the event of a disaster. Therefore, it is the one thing you need to plan for and practice most often.

## Networks Are Like Bathrooms

When issues arise, and *issues will always arise*, people will start to blame the pieces of their environment they know the least about. As the DBA you will find people inevitably blaming the database server for all sorts of problems. In my experience I have seen people blame the database server for a host of problems that have nothing to do with the database server. The fact is that they know the database server exists, but they don't know if it is truly the problem, so they just point their finger in your direction first.

One of the last things that anyone ever thinks about is the network. Your network has ups and downs all the time. There are lots of wires, routers, and switches that make up the backbone of your network. With so many moving parts that need to go together in order to make your systems work, it is easy to understand that issues will arise from time to time. I once read a study that said how database servers are the number one piece of infrastructure that people are ordered to troubleshoot, but over 70 percent of all issues are directly related to bad application code.

As a DBA you may sometimes find yourself blaming the network simply because you cannot possibly blame bad code all of the time. And it is fine for you to blame the network, because the network team will usually just blame management for not agreeing to buy the right equipment. But the truth is that people will often misunderstand the difference between networks and computers.

It's common to hear people in offices yell out, "The database server is down!" when the problem is something completely different. Networks are a lot like your bathroom at home.

## Call a Plumber

I could not tell you the first thing about plumbing, but it would not take more than a few nights of listening to a leaky faucet before I would head down to the hardware store to start asking questions. But a leaky faucet does not mean that you have a problem with your plumbing. Likewise with database servers; if your database server crashes, that does not mean your network is offline. And if one workstation is showing signs of a problem and others are not, then chances are it is not the network either. (For the record, if one workstation has problems and others do not, the problem is not likely the database server either.) And if that workstation cannot print to a network printer, you will not solve that problem by going into the network closet.

If your faucet were to spit out some dirty water, what would you do? Probably let it run for a few minutes to see if it clears up, right? Perhaps the city was working on the water lines, and after a few minutes the issue will go away. If you cannot get to a web site or a particular network server, maybe someone is moving some cables around. People need to perform maintenance on a regular basis, and it is not practical to expect everything to be flawless all the time. Wait a few minutes and try again before sounding the alarm.

You don't have to be a plumber, however, to know that if you run out of hot water after a 5-minute shower, you are going to need to spend some money to fix the problem. And most businesses will put up with a slow network for a period of time rather than face the

expense of actually fixing the problem. Having the network administrators work an extra day a week is not going to solve the problem, either.

The stuff that matters most is invisible and expensive. Beautiful faucets won't impress anybody if the entire house is running off a one-half-inch garden hose. The real backbones of your plumbing system are the water heater and the pipes running behind the walls and under the floorboard. Those pieces are hard to go back and fix later, so you don't want to make those purchasing decisions without qualified help. If you're not buying this kind of equipment on a regular basis, you don't know the ins and outs of the brands, features, and even the local regulations involved. Network hardware is the same way: the basic plumbing guidelines stay the same, but things change every year; and without help, you can end up buying gear that you can't easily replace later.

## Fix It Now!

Good plumbing requires a lot of knowledge and planning. If you need to water your lawn, you can slap some hose extensions together, scatter some sprinklers around, and turn the faucet on whenever the grass starts to turn brown. But if you want to do a good job, get your lawn looking perfect, and not have to hassle with moving the sprinklers around and manually turning the faucet off and on, you have to do your homework. When it comes time to build a network, you want to research how the pipes work and what kinds of connectors to use, strike a balance between reliability and affordability, and make sure you never have to touch the equipment again. None of that is guesswork.

Even the most minor plumbing problem is perceived as an emergency. When somebody can't get hot water to take a shower, they get angry quick. They might be able to make do with cold water, but they're not happy about it, and they proceed to tell everybody they meet about what a lousy morning they're having, and how the landlord better fix it or there's going to be trouble. When a user is having a problem, they yell and pull their hair out. Problems from "I can't print on the network!" to "The database server is slow!" are all spoken with the same urgent tone of voice, and everyone will expect you to drop everything immediately.

## DBA Stands For . . .

Now that you have some technical basics, let's review some professional basics. To start with, let's look at the following question:

*What does DBA stand for?*

Depending on whom you ask, you will get a different answer. You might respond with "database administrator," and technically you would be right. To some the answer would be "default blame acceptor." And to others it could be an acronym for "don't bother asking." Keep poking around and you will find that there are many different points of view with regard to what DBAs do, what DBA stands for, and what DBAs should be doing that they are not currently doing already.

Why so many viewpoints? DBAs touch many different areas of the company in many different ways. To many end users, we are the ones that reset their passwords and the ones that fix things when the system is running slowly. As such, we are also blamed for the systems running slowly even if we had nothing to do with the design of the system (especially with vendor applications). In short, DBAs get blamed for a lot of things that they have no control over, but because they have been helpful at one point or another when troubleshooting, they are ultimately seen as the owners.

**TIP:** No one knows what you really do, not even your boss. They only know what you do not do, or to be more precise, they only remember your failures and easily forget all your success.

Ask a developer about what a DBA does and you will be told that they are a roadblock to progress. The implementation of standards and guidelines for database usage only slows down development work. Of course, the upside is having a stabler environment, but that hardly matters to the developer screaming that they need sys admin access or else their project will be late.

With these different touch points around your role, you also have different perceptions, and you need to be aware of these perceptions so that you can make an effort to mitigate them before they drive a wedge between you and your customers. There are three things to keep in mind for all of your interactions daily.

## People Will Resist Change

Change happens, it's a fact. So why do so many people resist change, despite the fact that it happens all the time? You would think that everyone would be used to having things change, and yet some people stubbornly refuse to let go.

There are three main drivers of change: people, technology, and information. In case you were not aware, as a DBA you will (most likely) be working in a division of a company called Information Technology. That title has two of the three drivers alone! Toss in the fact that *people* are employed there, and it's no wonder you will see so much change on a daily basis. To quote Alan Zimmerman, "The real problem isn't the change . . . it's people's reaction to the change."

The change cycle itself has three stages: a beginning, a middle, and an end. When change happens we can dive right into it at the beginning, while in the middle we feel tied to both our past (known) and our future (unknown) states, and at the end we finally let go of the known state, which is often something we found dependable and reliable. As people pass through these three stages, they react in different ways. Consider the following type of groups of people you may already know:

- Innovators
- Early adopters
- Early majority

- Late majority
- Late adopters
- Diehards

Probably sounds like just about everyone you know, right? And there is a good chance that those classifications apply to yourself as well, and you fall into a different class for different things. Perhaps you never upgrade to a new version of software until the first service pack is released. Sound familiar? And perhaps you are also first in line to buy the latest iPhone. In some cases you embrace change and in others you proceed with more caution.

The value that one places on the items and things most familiar to them is what will lead them to either embrace or resist change. Think about that for a moment. If you place more value on the known than the unknown, would you not also resist change? The same holds true for the end users that you support. If you propose that they start doing a lot of things differently, you had better be prepared to show them how much more valuable the future state will be than the current one.

Why do you stand in line for the new iPhone? Because you value the new model more than your current one. Why do you wait for the first service pack to be released before even considering upgrading your software? Because you value the stability of your current system more than the new version.

As a DBA you are going to be asked to lead change on a number of occasions. And, on some occasions, you won't be *asked* to lead change—you will need to get up out of your cube and do it yourself without being told. When the time comes, follow these simple guidelines:

1. Define the need from the organization's point of view.
2. Define the need from the individual (or end user's) point of view.
3. Describe the desired outcome.
4. Define a timeline.

Believe me, this will take practice. Very few people have innate soft skills that allow for them to easily get people to embrace change. But for those that do, and for those that learn them, the result is a powerful way to motivate and energize people to new places.

You should also be aware of the types of resistance behaviors. You get two flavors for that: passive and active. Passive resistance would be a person who agrees verbally to everything you say but never follows through on what they agree to, or who feigns ignorance and perhaps withholds information. Active resistance would be a person who looks to find fault and ridicules change; who manipulates and appeals to a sense of fear. If you are aware of these behaviors when dealing with people, you can head off problems long before they become bigger problems.

## Having Standards and Processes Is Not a Bad Thing

It is inevitable that at some point you will be approached by someone who says they do not understand why you are doing things in a particular way. Chances are that this is a result of your having put into practice some standards that help you to provide a stabler environment. And these same people will be dumbfounded that (1) you need to have such a standard and (2) that it should really have to apply to *them*.

For example, suppose you are the new DBA in a shop that had no previous DBA. Everyone had full access to the database servers and could do whatever they wanted. To say things were a mess might be too kind. Being the good DBA that you are, you go through your checklist and find a few ways to make immediate improvements that would be of a great benefit to the entire company.

First, you go through each server and database and correctly place the data and log files on separate drives in an effort to avoid a large disaster. While getting the files moved around, you revoke the ability for anyone except yourself to create a new database. You do this because your root cause analysis of the problem led you to the conclusion that the reason there were so many databases set up incorrectly was because too many people did not understand the implications of their actions. In an effort to help them avoid hurting themselves, you make the necessary changes.

About an hour after you start revoking access, you get a phone call from a developer that cannot understand why they cannot create a database. “It’s just a database—why should I call you to create one when I am capable of doing it myself?” You will resist the urge to respond with, “If you could do it yourself, then I would not have spent a week cleaning up after you like the guy following the horses on the Fourth of July,” and will do your best to explain that in an effort to have a stable environment, it is best if database creation were in the hands of the few, not the many.

Try to understand that most people have a horizontal view or experience in their daily life. Developers need to get to work, log into the network, access the database server, make changes, test their changes (hopefully), and deploy changes to production. Why is that a horizontal view? Well, think of all the vertical silos they went through during the day:

- They drove to work, parked their car, and entered the building (facilities).
- They logged into the network (domain controllers, network).
- They accessed the database servers to make changes (you).
- They test their changes (get customer sign-off).
- They deploy their changes to production (you again).

You have nothing to do with their access to the building or their access to the network in general, but you do have some control over how they are allowed to make changes to the database servers. In this scenario your view is a vertical one: you are mostly concerned with your silo (database servers), and have no idea what the horizontal view looks like. And this can be a very hard thing to reconcile; you really need to work with

your end users and communicate with them on a frequent basis to understand all of their pain points.

Once you start to understand that most of your end users are taking a horizontal view, it is easy to see why they might be frustrated when you start to enforce policies, guidelines, and standards. Remember that people may resist change and use this as an opportunity to lead change. Over time your soft skills will improve to the point that people will begin to trust that your actions are best for the company as a whole, and not intended as a slight to any one individual.

## People Will Blame What They Do Not Understand

This one is fairly self-explanatory. People will always tend to blame something they are aware of but do not fully understand. It is only natural, right? I mean, if you think you already know everything about nine out of ten items, then your mind will focus on that tenth item and you will spend far too much time on why the tenth item is causing you so much heartache at the moment.

You will frequently be told such things as, “Our code hasn’t changed in years,” “Everything ran fine last night,” and “You guys must have done something yesterday because now all of our stuff runs much slower.” Get used to being the focus of attention for any problems with any system. Start developing some thick skin because you are going to need it, and soon.

Any time there is even one hint that something is amiss with a system, the first thing people will do is blame the database server. Despite the fact that there are many layers between their screen and the database server, your phone will be the first to ring. You will be expected to investigate the issue immediately, and you will also be surprised to see e-mails with sentences that read, “I called the DBA and they are going to fix the problem.” Wait a minute! We never said there was a problem with the database server—why are you telling people we are going to fix anything? The problem could be the network, or a poor design that didn’t scale, or *anything*. And yet people will fixate on the database server because (1) it is known and (2) most people do not understand how it works. And in some cases, if they get an error message such as, “Could not connect to the database” (or the word “database” is simply in the error message somewhere), people automatically assume that the problem *must* be with the database.

I have lost count of the number of times I have been told there is something wrong with the server only to find that the issue is that the person or account did not have rights to log in. Sorry, but that is not a problem with the server. And it is also not a problem with the server if you try to load 100GB of data onto a disk that only has 10GB of space free. Same for filling up a 33GB tempdb drive; the issue is not with the server, it is with inefficient code. And yet your server (and ultimately yourself) will be forced to carry the burden of fault.

That’s OK, because one of the reasons you are a DBA is because you are able to carry such a burden as would crush most of your peers.

## Blame vs. Credit

In the preceding section, we discussed how people tend to blame things they do not understand. Another fact that you need to get used to is that people rarely go out of their way to give you credit for the things you do or suggest should be done. I have never once come to work to be told, “Nice job last night, our batch load ran two seconds faster than expected.” But if it runs two seconds longer, then you may be asked to start on a Star Trek–type Level 5 diagnostic, whatever that means.

You are going to find out quickly that the only time you get noticed is when things go wrong. Some DBAs start to think that the only way they can show value is if they are always around to fix the things that go wrong. Personally I would rather build out an environment that hums along nicely without anything ever going wrong to begin with, even if that means I will never be noticed.

If you do find yourself in the situation where you are not being noticed, then what you need to do is start building out some metrics. Start with something simple, like your database backups. Track how many backups were taken in the past week and how many backups failed. At some point, perhaps a month or two, you can use the data to report to your manager. If you have reduced the number of backup failures over time, then this report is going to help you demonstrate your value to the company in a way that did not require something to go wrong before you got noticed.

**TIP:** Metrics and reporting are ways for you to demonstrate your value; use them wisely.

Metrics are a wonderful way for you to build up a stable environment, report on the good work you are doing, and show value besides during a disaster. Besides tracking backups and backup failures, here are a few other good items to track over time:

- Number of databases (broken down to production, nonproduction, etc.)
- Amount of space used on disk by database files
- Amount of space used by database backups
- Amount of time taken to back up databases
- Number of database restores
- Number of new databases created

All of this information is readily available in the system databases, and will go a long way to help you make certain that people are aware of your work behind the scenes.