Pro SQL Server Disaster Recovery

ISBN-13: 978-1-59059-967-9

ISBN-10: 1-59059-967-5

ISBN-13 (electronic): 978-1-4302-0601-9

ISBN-10 (electronic): 1-4302-0601-2

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication data is available upon request.

Lead Editor: Jonathan Gennick
Technical Reviewer: Steve Jones
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Nicole Abramowitz
Associate Production Director: Kari Brooks-Copony
Production Editor: Kelly Gunther
Compositor: Linda Weidemann, Wolf Creek Press
Proofreader: Elizabeth Berry
Indexer: Broccoli Information Management
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

CHAPTER 1

What Is Disaster Recovery?

One of the greatest frustrations I've faced is discussing (or arguing) a topic for hours, only to realize near the end that my audience and I have completely different views as to what the topic actually is. With that in mind, I hope to make clear what I consider to be *disaster recovery*. My goal is to establish a common understanding of the topic being discussed.

In this chapter, I'll establish what disaster recovery means for the purposes of this book. To accomplish this successfully, I'll discuss

- Disaster recovery from a procedural perspective
- How disaster recovery relates to similar terminology—specifically, *business continuity* and *high availability*
- Exactly what is considered a disaster
- Disaster recovery from a technical perspective

Defining Disaster Recovery

Working as a consultant, one of the most difficult situations to handle is restoring life to a downed SQL Server. It's a stressful situation for everyone involved. When clients call me, it means they have a significant issue. Usually they're unclear as to what the exact problem is or how to proceed.

There are five words that I never want to ask but sometimes have to: "When was your last backup?" From the answer, I can immediately determine if this client has a clear understanding of disaster recovery. Some immediately spring into action, pulling backup tapes out to begin their own documented restore process. But all too often, the question is met with silence and blank stares.

Over the years, I've been in this situation dozens, if not hundreds, of times. Looking back, what jumps out at me is how differently everyone views SQL Server disaster recovery. Here are some of the various interpretations that I've encountered:

- Making sure your data is backed up
- Having a backup/recovery scheme
- Having a *documented* backup/recovery scheme
- Having a documented backup/recovery scheme with directions so thorough that a ten-year-old could follow them
- Off-site data storage
- Planning and documenting all procedures to respond to any type of outage

As this list implies, some view disaster recovery somewhat simplistically, while others see it as a massive project. All interpretations can be valid, but bear in mind that one interpretation might encompass too much, while another interpretation might leave important aspects out.

For the purposes of this book, I'll define disaster recovery as encompassing the following:

- The process involved in returning a downed instance or server to a functioning state
- The process of restoring a damaged database to a functioning state
- The process of restoring lost data
- Mitigation of risks for downtime or loss of data
- Identification of cost, either due to mitigation steps taken or downtime/data loss
- Some level of planning and documenting these processes and mitigation steps
- Consultation with the business owner of the data

Consulting with the business owner of the data is a critical step. The owner is the only one qualified to determine how much downtime or data loss is acceptable. That ties in closely with cost; usually the business side of the equation wants to see zero data loss and zero downtime. This is often an extremely costly goal to achieve. If you don't put cost into the equation from the beginning, it's unlikely you'll get approval for the cost when it comes time for implementation.

This list is still a little vague, but I'll make things clearer in Chapter 12 when I discuss overall disaster recovery planning.

Disaster Recovery, High Availability, and Business Continuity

These three terms sometimes are used interchangeably and often have "floating" definitions. Of the disaster recovery planning projects I've seen that have failed (or at best, limped), the primary reason for the failure was a lack of consensus as to what the project was trying to accomplish. What usually happens is that the basic disaster recovery portion of the project gets a severe case of scope creep, followed by mass confusion and unrealistic expectations.

The following are my definitions for these three terms:

- *Business continuity*: The process of ensuring that day-to-day activities can continue regardless of the problem. It encompasses both technical and nontechnical disasters, such as a worker strike or a supply-chain issue.
- *High availability*: The process of ensuring that systems remain available as long as possible no matter what the cause might be for downtime. This includes disasters, but it also includes events such as regular maintenance, patches, and hardware migration.
- *Disaster recovery*: The process of mitigating the likelihood of a disaster *and* the process of returning the system to a normal state in the event of a disaster.

Figure 1-1 shows the relationship between the three terms.

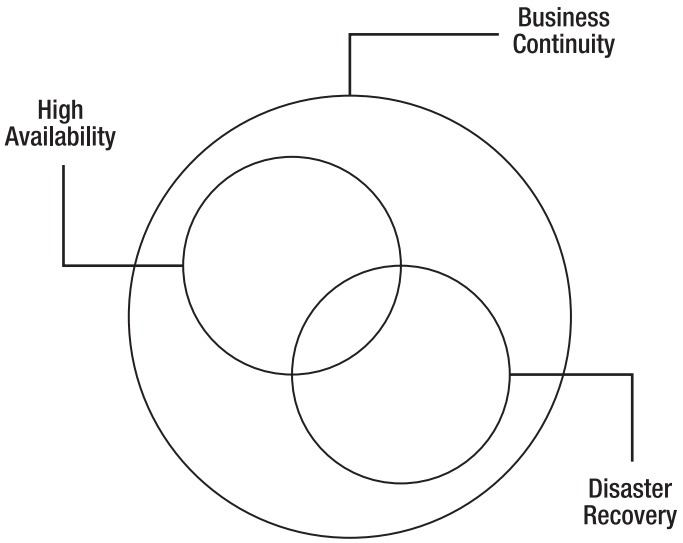

Figure 1-1. *The relationship between business continuity, high availability, and disaster recovery*

Remember, the result of failed disaster recovery projects is often massive scope creep and communication chaos. In the following sections are some specific examples of such failures, taken from real situations that I've either faced or helplessly witnessed.

The Commandeered Project

At a large company, the database administration (DBA) team decided it needed to formalize its backup and recovery operations. The goal was to clearly document its procedures and establish a periodic disaster recovery drill. All of this was confined to the context of the database environment. This was a large undertaking for the team, but it had a clear goal and realistic deliverables.

Every item of work had to be associated with a project, so the team leader entitled the project "Database Disaster Recovery." Eventually, through the miracle of status reports, this project title reached an executive who was so impressed with the initiative and dedication of the DBA team that he rewarded them with the usual: more work. The company announced that the DBA team leader would become the project manager for a new company-wide disaster recovery project.

As you might imagine, the result was the Never-Ending Project. The DBA team leader was not a project manager by profession. Nearly every technical department in the company had a representative present at meetings, which were consumed by continual discussion of every possible scenario that the project needed to address. There was little structure to the project, no clearly defined goals or deliverables, and certainly no sense of order.

After two years and only a list of horrific disaster scenarios to show for it, the project finally faded into memory as fewer and fewer departments sent representatives to meetings. The DBA team was finally able to return to its initial project, which was newly entitled "Database Documentation."

The "We Were Supposed to Do That?" Project

A much smaller manufacturing company took a top-down approach to disaster recovery. At a company meeting, the owner of the company asked the staff to put together a plan of action to return the business to a functioning status in the event of a disaster. The technical staff did a fine job assessing risks and documenting workarounds and recovery steps for a variety of disaster scenarios. The documentation was assembled, and the task was considered complete.

Months later, workers at the trucking company that delivered raw materials to the plant walked off the job. The owner of the manufacturing company had orders to fill without the ability to manufacture the products. He quickly picked up the disaster recovery documentation and looked for a section regarding an interruption in the supply chain. You guessed it: there was no such section. The technical staff addressed only the

technical issues. When the owner asked for a disaster recovery plan, he was really looking for a business continuity plan. As a result, the company lost customers who couldn't wait three weeks while the manufacturer found a secondary supplier.

The High Availability/Disaster Recovery Project

The owner of a promising Internet startup hired a consulting company (for the record, not me) to help ensure that the startup could get as close as possible to 99.999% uptime for its web site. The contract specified that the consulting company was to design the system such that there would be no more than 30 minutes of downtime in the event of any single failure. The consultants came in and did a fantastic job of setting up redundant servers, network connections, and even a colocation. The 30-minute threshold would easily be met.

Business took off for the startup until one day when the main data center suffered a massive power outage caused by flooding in the building. Even the backup generators failed. Luckily, the company had an identical system on standby at another facility. A few name changes and domain name system (DNS) entries later, the system was back online.

Weeks later, the owner started receiving hundreds of complaints from customers who had ordered products but never received them. Upon investigation, he discovered that neither the customers nor their orders were in the system. Frustrated, he called the consulting company, demanding an explanation. It turned out that the system that was colocated was only updated every 30 minutes. When the flood occurred and the standby system was activated, the company ended up losing 20 minutes' worth of data. The ultimate financial impact: $79,000, plus more than 200 customers who would never return.

The issue of data loss and its financial impact was never discussed. The consulting company did its job, but no one even thought to question the risk of data loss.

The Price of Misunderstanding

The previous scenarios are just a few examples of what can happen when terms aren't clearly defined. Wasted time, frustration, financial loss, and the perception of failure can all be avoided by simply being clear when using terminology.

Now that I've established what I mean by the term *disaster recovery*, I'll clarify what constitutes a disaster.

Disaster Categories

I find it useful to categorize various types of disaster. This isn't simply due to my compulsive need to categorize (hey, I'm a database guy). It's useful to understand exactly what is meant by a *media failure* or a *hardware failure* and to appreciate what sort of events can

cause them. I'll walk through each of the categories and offer some real examples that I've encountered over the years. Then I'll compare the categories in terms of the following criteria:

- Probability of occurrence
- Predictability of the event
- Overall impact (usually measured from a financial perspective)

■ **Tip** I find it useful to clearly identify the problem before even thinking about a solution. When looking at a disaster scenario, be sure you know the root cause. If you don't, your solution might just be a Band-Aid on a larger issue.

Environmental

As the name implies, environmental disasters are ones in which the environment of the server has been affected in some way. These can come in a wide range of variations. Here are a few real-life scenarios I've encountered:

- *Poor server-room design*: I once encountered a client who had poor ventilation within the server room. It wasn't out of negligence; the server environment simply grew more quickly than the client's capacity to house it. Due to the heat, servers would fail randomly throughout the day.
- *Natural disaster*: A tornado swept through southern Wisconsin, randomly destroying buildings (as they like to do). One of the buildings it hit had servers in it, which ended up in pieces in trees more than a mile away.
- *Accident*: A water pipe running close to a rather large (and well-constructed) server room burst, flooding the floor. Luckily, the flooding was contained and only a few items were affected; however, some of those items were terminals to manage the servers themselves. This was before Microsoft came out with its Terminal Services product, so there was no means of managing them other than hooking heavy CRT monitors to each server.

Hardware

Hardware includes not only the server itself, but also any associated hardware, including network devices and other dependent hardware such as domain controllers. The following are some examples of hardware disasters:

- *Failed motherboard*: A server with a single power supply simply shut down. After multiple attempts to boot, the client called a vendor technician to bring in a replacement power supply. Still, the server wouldn't clear a *power-on self-test* (*POST*). After further investigation, it appeared the problem was a damaged motherboard.

- *Damaged cables*: A newly purchased cluster was implemented for redundancy. The client tested it for months before moving it to production. Once in production, nodes would fail randomly. It turned out that the SCSI cables were bound so tightly that they were shorting out, briefly disrupting communication with the disk array and causing random failure events.

- *Complete network failure*: Construction crews accidentally severed a primary fiber-optic line that carried all network traffic between a larger organization and its more than 4,000 satellite offices. The primary business tools were located at the main office, so all work essentially stopped until the line was repaired.

Media

Hard drives and tape drives have one major weakness—they're magnetic media, which means they can be corrupted easily if put in the proximity of a magnetic field. These devices also contain moving parts, which are prone to break. Here's a list of some types of media failures that you might encounter in the field:

- *Failing disk drive*: In this simple and common failure, a single drive in a disk array begins to acquire bad sectors. Within a week, it fails outright.

- *Corrupted active backup tape*: An active tape used in a backup rotation becomes corrupted, retaining only a portion of the information it should be storing.

- *Damaged archival tape*: An archived tape, being stored offsite, becomes damaged and entirely unreadable, probably due to being dropped or placed on top of a TV or monitor.

Process

I define process errors a little differently than most. A process can be one of these two things:

- An automated program or script that runs unattended

- A manual process performed on a scheduled basis, either reoccurring or once

You might argue that a mistakenly executed manual process is a form of user error, but these types of errors fit more closely with automation in terms of likelihood, impact, and predictability. For example, when performing a database upgrade for an application, you usually have the opportunity to assess the likelihood and impact of a failure. You can plan accordingly to mitigate risks. Since you know exactly when this process is going to occur, you can, in a sense, "predict" a failure.

Here are some examples of process failure:

- *Service pack installation issues*: While installing a service pack, a cryptic error message occurs. From that point on, SQL Server doesn't start in normal mode.

- *Manual task not performed*: At a satellite office, one of the formal jobs of the office manager is to change the backup tape daily. The office manager goes on vacation and forgets to instruct someone else to change the tapes. A failure occurs, prompting a restore to two days ago. Unfortunately, there is only one tape, and it has been continually overwritten each morning. Thus, the only backup you have is from that same morning; you have nothing from two days previous.

- *Automated backup failure*: A SQL Server backup job has been automated to run every night at 4 a.m. Failure notifications go to a single DBA. The DBA has been out sick for four days when a failure occurs and a restore is required. Nobody was aware of the problem, because the one person who received the failure notification e-mails had been out sick and not checking his e-mail. A system administrator goes to the server to restore the SQL databases only to discover that the automated job has failed the past three days.

User

User error is the most difficult category to deal with. User errors are unpredictable and potentially devastating in their impact. Users can come up with a seemingly unending number of creative ways to cause havoc. Here are a few examples:

- *"Where is the* `WHERE`*?"*: Most of us are guilty of this at one time or another. We forget to put a `WHERE` clause on a `DELETE` or `UPDATE` statement. In one particular case, an ad hoc change was required to the database—three child records needed to be associated with a different parent record. Needless to say, the change occurred for every record in the database.

- *"I didn't delete* that*"*: A data entry clerk accidentally deletes the largest customer from the database. Cascading deletes had been turned on, so every order that customer made was deleted.

- *Too much power in the wrong hands*: My favorite disaster of all time happened to me personally. I was working as a DBA at a small consulting company with only one SQL Server. A colleague came into my office with the following speech: "Hey, I was going to upload those new files we got from that client, and there wasn't enough room, so I tried to delete *something something dot something*, but it said it was in use, so I disabled a service and then I was able to delete it. Now I can't connect to the SQL Server. Did you make some changes?" Enough said.

Predictability, Probability, and Impact

I'm sure many of you are reading through these examples and thinking "This one is easy to remedy" or "Well, if you designed the application properly. . . ." If you're thinking that, fantastic! Problem solvers unite! I, however, approach things a bit more cynically. I look at each example and think about what other things might go wrong or roadblocks to implementing some sort of mitigation solution. Whatever your outlook is, let's take things one step at a time and simply identify potential disasters.

I previously said these categories revolve around the likelihood of occurrence, the predictability of an event, and the impact of the failure. Table 1-1 summarizes the category breakdown.

Table 1-1. *Probability, Predictability, and Impact of Various Failure Types*

Failure Type	Probability	Predictability	Impact
Environment	Very low	Natural disasters are impossible to predict, but a poorly constructed server room may be an entirely different matter.	Usually catastrophic.
Hardware	Low	Some server monitoring tools warn of impending failure.	Downtime and data loss; how much depends on what failed.
Media	Low	Most RAID controller software provides some means of predicting an impending drive failure. Type storage is rarely accessed, making it extremely difficult to predict impending failure.	Ranges from relatively no impact when losing a single RAID 5 drive to significant downtime and potential data loss.

Continued

Table 1-1. *Continued*

Failure Type	Probability	Predictability	Impact
Process	High	There's always some level of predictability, because the events happen at a fixed time.	Could range from embarrassment to major downtime and data loss.
User	Usually low, depending on training and application design	Almost impossible to predict, although poorly trained staff and a poorly designed application may be a hint.	Could range from a minor annoyance to catastrophic.

Probability, predictability, and impact together help to prioritize disaster recovery planning. If a disaster has a low probability, is difficult to predict, and has a relatively low impact, there's no sense in placing it at the top of the list of action items (or placing it on the list at all).

I'll refer back to these categories and scenarios throughout the book, applying specific technical features to each particular category and the example scenarios.

Disaster Recovery from a Technical Perspective

Up to this point, I've been approaching the question of disaster recovery from an abstract, procedural level. While it's important to think about the subject in an abstract way, this is a technical book. I established that, for the purposes of this book, disaster recovery encompasses reducing the likelihood of the disaster and returning the system to a functioning state. Simply put, disaster recovery is mitigation and response.

SQL Server has long had technologies in place to handle mitigation and response. SQL Server 2005 includes new technologies and improvements that have completely changed the way we should think about disaster recovery. Having a backup and recovery plan can and should be augmented by other techniques. Given the increasing size of the average database, a straightforward full backup is becoming an untenable technique on which to rely.

Mitigation Technologies

Certain technologies center only on reducing the likelihood or impact of any particular disaster. These I classify as *mitigation technologies*. Here are some examples:

- *Clustering*: A longtime feature in SQL Server, clustering allows you to set up additional failover servers to take control of the database should the primary server fail.
- *Log shipping*: A technique that has been used manually in the past, log shipping is the process of copying log backups and moving them to a standby server that continually restores them. If the primary server fails, users can be redirected to the standby server manually.
- *Database mirroring*: A completely new technology in SQL Server 2005, database mirroring provides automatic failover. Unlike clustering, there is no shared data, and the standby database can be on any server in any location.

Response Technologies

If this were a perfect world, mitigation techniques would always protect our systems and data from disaster. This is not a perfect world. While mitigation techniques in disaster recovery planning are useful, having a response plan is a requirement. Here are some examples of response technologies:

- *Backup and restore*: No database platform would be complete without functionality to back up and restore a database. SQL Server 2005 provides additional functionality such as mirrored backups and checksum validation.
- *File/filegroup backup and restore*: SQL Server allows you to back up individual data files or filegroups, though this method isn't used frequently. It can aid tremendously in designing a backup scheme for a very large database (VLDB).
- *Database snapshots*: Also new to SQL Server 2005, a database snapshot lets you revert back to a specific point in time without having to go through an entire restore process.

A wide range of technologies apply to disaster recovery. As responsible DBAs, we should be using every option at our disposal to approach any potential disaster situation. Given the number of disaster categories and the technologies available to address them, it's time to be creative and think outside of the "backup/restore" box.

Caveats and Recommendations

Common understanding is the key to any meaningful discussion. Before undertaking any major initiative, it's critical that all parties be clear on terminology and objectives. In this chapter, I've attempted to clarify disaster recovery as it pertains to this book.

The following are some additional thoughts to keep in mind as you work to implement a sound disaster recovery process in your own environment:

- *Stay simple*: Don't make things too elaborate or define terms in a complex way. The key is to break everything down into base concepts, leaving as little room as possible for implicit interpretation. Complex terminology or objectives are usually the primary cause of misunderstanding.

- *Agreement is not a requirement for action*: I'm sure that some of you who are reading this don't quite agree with how I've categorized things or how I've defined disaster recovery. That doesn't mean the rest of the book doesn't have value. The key is that you understand my position before moving forward. The same applies to any undertaking.

- *Categorization isn't just show*: Being a database professional, categorization is almost a compulsive need for me, but there is a real purpose behind it. If individual topics or problems have commonality, it is likely that approaches to discussing them have the same commonality.

Summary

I've established that disasters can be categorized in five basic ways: environmental, hardware, media, process, and user. Each individual disaster scenario has a certain probability, predictability, and impact, which together determine the priority of actions taken. Disaster recovery is the process of reducing the probability or impact of a disaster and the actions taken to respond to that event.

Now that I've spelled out the basic procedural structure for disaster recovery, I'll explain the technical options available to you and how they relate to this basic structure.