Planning for Disaster Recovery

SQL Server 2008 R2

When you are administrating a SQL Server database, preparing to recovery from potential disasters is important. A well-designed and tested backup and restore plan for your SQL Server backups is necessary for recovering your databases after a disaster. For more information, see Introduction to Backup and Restore Strategies in SQL Server. In addition, to make sure that all your systems and data can be quickly restored to regular operation if a natural disaster occurs, you must create a disaster recovery plan. When you create this plan, consider scenarios for different types of disasters that might affect your shop. These include natural disasters, such as a fire, and technical disasters, such as a two-disk failure in a RAID-5 array. When you create a disaster recovery plan, identify and prepare for all the steps that are required to respond to each type of disaster. Testing the recovery steps for each scenario is necessary. We recommend that you verify your disaster recovery plan through the simulation of a natural disaster.

When you are designing your backup and restore plan, you should consider your disaster recovery planning with regard to your particular environmental and business needs. For example, suppose a fire occurs and wipes out your 24-hour data center. Are you certain you can recover? How long will it take you to recover and have your system available? How much data loss can your users tolerate?

Ideally, your disaster recovery plan states how long recovery will take and the final database state the users can expect. For example, you might determine that after the acquisition of specified hardware, recovery will be completed in 48 hours, and data will be guaranteed only until the end of the previous week.

A disaster recovery plan can be structured in many different ways and can contain many types of information. Disaster recovery plan types include the following:

- A plan to acquire hardware.
- A communication plan.
- A list of people to be contacted if a disaster occurs.
- Instructions for contacting the people involved in the response to the disaster.
- Information about who owns the administration of the plan.
- A checklist of required tasks for each recovery scenario. To help you review how disaster recovery progressed, initial each task as it is completed, and indicate the time when it finished on the checklist.

SQL Server Recovery Models

SQL Server provides three alternative recovery models: simple, full, and bulk-logged. A recovery model is a database property that controls the basic behavior of backup and restore operations for a database. Selecting the optimal recovery model for each of your databases is a required part of planning your backup and restore strategy. The choice of recovery model for a given database depends somewhat on your availability and recovery requirements. The choice of recovery model, in turn, affects the possibilities for disaster recovery for a database.

For an introduction to recovery models, see Recovery Model Overview.

Managing Backup Media

We recommend that your backup plan include provisions for managing backup media, such as the following:

- A tracking and management plan for storing and recycling backup sets.
- A schedule for overwriting backup media.
- In a multiserver environment, a decision to use either centralized or distributed backups.
- A means of tracking the useful life of media.
- A procedure to minimize the effects of the loss of a backup set or backup media (for example, the loss of a tape).
- A decision to store backup sets on or offsite, and an analysis of how this will affect recovery time.

For information about how SQL Server uses backup devices and media, see Working with Backup Media in SQL Server.

Running a Base-Functionality Script

Usually, you include a base-functionality script as part of your disaster recovery plan to confirm that everything is working as intended. A base-functionality script provides a dependable tool for the system administrator or database administrator to verify that the database is back in a workable state, without depending on end users for verification.

A base-functionality script is application-specific and can take many different forms. For example, on a decision support or reporting system, the script might just be a copy of several of your key reporting queries. For an online transaction processing (OLTP) application, the script may execute a batch of stored procedures that execute INSERT, UPDATE, and DELETE statements. For example, a base-functionality script might be as simple as an .sql file that sends batched SQL statements to the server from the **sqlcmd** utility. Another example is using a .bat file that contains both **bcp** and **sqlcmd** commands.

Ensuring Disaster Readiness

To make sure that you are ready for disaster, we recommend that you periodically perform the following activities:

- Test your backup and recovery procedures thoroughly before a real failure occurs. Testing helps ensure that you have the required backups to recover from various failures, that your procedures are clearly defined and documented, and can that they can be executed smoothly and quickly by any qualified operator.
- Perform regular database and transaction log backups to minimize the amount of lost data. We recommend that you back up both system and user databases.
- Maintain system logs in a secure manner. Keep records of all service packs installed on Microsoft Windows and SQL Server. Keep records of network libraries used and the security mode. Also, if SQL Server is running in Mixed Mode Authentication (SQL Server and Windows Authentication Mode), record the **sa** password in a secure location. For more information, see Security and Protection (Database Engine).

Important
Windows Authentication is much more secure than SQL Server Authentication. When possible, you should use Windows Authentication.

- On another server, assess the steps that you have to take to recover from a disaster. If it is necessary, amend the steps as necessary to suit local server environment, and test the amended steps.
- Maintain a base-functionality script for quickly assessing minimal capability.

Auditing and Reducing Potentially Disastrous User Errors

One of the more challenging recovery scenarios is recovering from a serious user error, such as accidentally dropped database objects. This section lists tools that can help you in auditing, and in some cases regulating, changes to databases.

- Data Definition Language (DDL) triggers

  These triggers can be created for auditing and regulating certain changes to your database schema. DDL triggers fire stored procedures in response to a variety of DDL statements. These are primarily statements that start with CREATE, ALTER, and DROP. The scope of a DDL trigger is either a particular database or a whole server instance. For more information, see Understanding DDL Triggers.

- Event notifications

  Event notifications execute in response to a variety of Transact-SQL DDL statements and SQL Trace events, and send information about these events to a Service Broker service.

  Event notifications can be programmed against many of the same events captured by SQL Trace. But unlike creating traces, you can use event notifications to perform an action inside an instance of SQL Server in response to events. Because event notifications execute asynchronously, these actions do not consume any resources defined by the immediate transaction. For more information, see Event Notifications (Database Engine).

Note
Not all DDL events can be used in DDL triggers. Some events are intended for asynchronous, non-transacted statements only. For example, a CREATE DATABASE event cannot be used in a DDL trigger. You should use event notifications for such events.

- SQL Server Agent

  This is a Windows service that executes scheduled administrative tasks, which are called jobs. SQL Server Agent uses SQL Server to store job information. Among other things, SQL Server Agent can run a job in response to a specific event, such as errors that have a specific severity level or message number.

  For an introduction to SQL Server Agent, see Automating Administrative Tasks (SQL Server Agent). For information about how to use SQL Server Agent for events, see Monitoring and Responding to Events.

- SQL Trace

  SQL Trace provides Transact-SQL system stored procedures to create traces on user-selected event classes in an instance of the SQL Server Database Engine. These system stored procedures can be used within your own applications to create traces manually. For more information, see Introducing SQL Trace.

Note
SQL Server Profiler is a graphical user interface to SQL Trace for monitoring an instance of the Database Engine or Analysis Services. For more information, see Using SQL Server Profiler.

See Also

Reference

sqlservr Application

Concepts
Permissions (Database Engine)
Recovery Model Overview

Community Additions