Database Backups- The Crib Sheet

For things you need to know rather than the things you want to know

Robyn Page's crib sheet was a Simple-Talk classic, providing a terse but thorough roadmap of all of the important SQL Server backup-related considerations. It returns, newly revised.

Contents

Core Backup Strategies

- Database Restore Strategies
- Why back up?
- What is a backup?
- When to do backups?
- Where should backups be placed?
- What needs backing up?
- Backup Policies and Procedures
- Who should perform backups
- Backup Media and Media Rotation

SQL Server Backup Knowledge

- Backup types
- Performing backups
- System Database backups
- Recovery Models
- Replicated Databases
- Backup to URL
- Striped Backups
- Backup History
- Backup file Information
- Basic backup process
- Backup Compression

Review Questions

Core Backup Knowledge

Database Restore Strategies

Plan immediately for how you want to bring databases back online from the backups, a restore strategy, not a backup strategy. This must be a written plan for methods and best practices to get databases back online. Different classes of data maintained (e.g. financial, corporate, departmental, personal), the risk of its loss, its value to the enterprise, and the general strategy for minimizing the damage or cost that would follow the loss of part, or all, of that data. A written agreement must exist for all production systems which defines the maximum tolerable loss of data and the database downtime. The restore strategy must be tested against various eventualities to verify that a full recovery of the database can be made in all circumstances. The strategy must be available for inspection by external auditors, insurers, or assessors.

Why back-up?

In order to arrive at a restored database, backups must first be taken in order to protect against hardware failure, malicious damage, user-error, fraud-detection, or database corruption. Backups can be used to undo transactions within the system, maintain copies of a database, archive a system at a point in time, and meet regulatory requirements for certain kinds of data storage and retrieval. Think of this like a spare tire on the car, better to have it and never need it, than need it and not have it

What is a backup?

There are several different kinds of backups. Starting with the most fundamental, a full backup, it is a complete, page-by-page copy of an entire database including all the data, all the structures and all other objects stored within the database. There are also differential backups that are a page by page copy of all pages in a database that have been modified in any way since the last full backup. You can also have file/filegroup backups which are page by page copies of specific files or filegroups that define a database. Finally, there is another backup type called a log backup that is a backup of all the committed transactions currently stored in the transaction log. These backups, which are written to files, are used to restore a database in various ways depending on the types of backups available and the goal of the restore operation.

When to do backups?

This is largely a business driven decision. Since the entire purpose of a backup is to be able to restore a database, in order to understand when you need to take what kind of backup depends on what kind of restores you intend to perform. The business will have to define two concepts, the Recovery Point Objective (RPO) and the Recovery Time Objective (RTO). The RPO is basically the amount of information that the business is willing to risk losing. This definition will drive a large number of decisions around how often different kinds of backups are taken. The RTO is defined as the amount of time it takes to perform the restore operation. Combining the RTO and the RPO will largely define how often each of the different types of backups are used within your system. A rough definition that will meet the RPO and RTO of many businesses is to run a full backup once every 24 hours. Then, all day long, run log backups every ½ hour. This will ensure that no more than ½ an hour will be lost.

Where should backups be placed?

The one place you shouldn't put backups is on the same drives as your data. The whole idea is to have a secondary storage location for the important business data. Initial backups should generally be done to a local drive on the system. Secondary copies should be regularly created and stored in locations other than the original location of the database. Cloud-based storage systems are an excellent secondary storage location.

What needs backing up?

Any data whose loss could lead to lost time or lost revenue for the business should have a backup available. In addition to the business data, it's a very good idea to also backup the system databases in order to be able to have a speedier recovery if you need to rebuild an entire system. It's also a good idea to have backups of all your maintenance scripts and programs.

Backup policies and procedures

Since backups are fundamentally a protection for the business, part of that protection has to be working with the business to arrive at a documented set of processes. This should include Recovery Point Objectives and Recovery Time Objectives as well as precise, step-by-step instructions, storage locations, and all other information necessary to successfully restore a database.

Who should perform backups?

Backups should not be run by anyone. The backup process should be automated so that it runs on a scheduled basis. Further, there should be monitoring or auditing in order to alert the necessary parties in the event of errors or failures. There must be someone within the organization who is tasked with setting up and monitoring the backup processes. Traditionally this role is taken on by someone designated as a Database Administrator (DBA). Whoever is responsible will need to be able to log onto the systems they are responsible for in order to perform any tasks needed as a part of the goal of having a successful restore in the event of trouble. This role would have to be fairly well defined and its access monitored in order to be compliant with various legal restrictions such as HIPAA, Sarbannes-Oxley and others.

Non-production systems may also need to have backups in place. A similar level of access to the system itself will be necessary for those whose role requires that they maintain the backups and restores on those systems.

Backup media and media rotation

The backup itself is a file or a set of files. These files can be stored on any media on which there is adequate space. Regardless of the media in use, it’s always a good idea to have more than one copy of the primary, full, backups as part of a fundamental data protection strategy. It is unwise to delete all but the most recent backup, as the integrity of a particular backup cannot be assumed, unless a test restore has been undertaken, followed by a DBCC CHECKDB to ensure database integrity. The period for which backups are stroed is a matter determined by the backup strategy and the needs of the business.

SQL Server Backup Knowledge

Backup Types

Different types of backups provide varying methods for protecting the data within the database. Traditionally, two backup types are used, a full backup and log backups. These satisfy most of the fundamental protections backups offer. But, depending on the size of the database, the number of transactions within the system and the RPO and RTO defined within the backup policies, other types of backups will be used. System databases are dependent on full backups only.

Full backups

Full backup is completely independent of the database recovery model. Production systems should have a full backup regularly scheduled in order to protect the data. System databases (including master and msdb) should be backed up after changes are made to the server, such as creating or removing databases, creating or modifying SQL Agent jobs, configuring security, adding or removing linked servers, etc.

Differential backups

The differential backup is linked to the last full backup. The differential backup will backup all pages that have been modified since that last full backup. Because the differential backup is usually much smaller than the full backup, it’s faster. The differential backup is used to offset the number of log backups that would be needed to restore a database to a point in time between full backups. This can make the overall recovery time faster.

The restore of a differential backup functions in much the same way as a restore from a full backup. It’s a complete copy of all the modified pages, so you can’t stop at a point in time with this type of backup. You can, however, combine a differential backup with log backups to restore to a point in time.

Differential backups cannot be used for the master database. For differential backups to be applied on a database, the database has to be in the NORECOVERY state. But SQL Server does not allow MASTER to be restored without recovery, thus you cannot perform a differential backup on MASTER.

If you are using differential backups, since they are linked to the last full backup, if you need to take an additional full backup, you can use COPY_ONLY with the backup command. This will prevent a new link being created to the full backup you are taking that is outside the normal schedule.

Transaction log backups

A transaction log backup represents all committed transactions from the transaction log between the last log backup and the point at which the current log backup completed. The first time you take a full backup, the log backup will be all the transactions between that full backup and the current moment in time. This is also applicable if you change the recovery model of the database. Otherwise, the log backups are completely decoupled from full and differential backups. Transaction log backups are used in conjunction with full and differential backups to restore a database to a point in time.

Transaction log backups are not linked to any particular full backup or differential backup, but, they are linked to each other. This means that log backups must be restored in a continuous chain. Any missing log backups means that you can only restore to the last log backup that you have in that chain.

Each time the transaction log is backed up, all of the committed transactions in the log will be cleared from the transaction log. These transactions will be the ones included in the log backups. This process will free that space within the log, allowing it to be reused, but it won't shrink the size of the log file. Transactions logs are combined with full and differential backups to achieve a more thorough backup process allowing for recovery to a point in time. Consider the following set of backups:

> 02:00 – full backup
> 02:15 – transaction log backup 1
> 02:30 – transaction log backup 2
> 02:45 – transaction log backup 3
> etc

To restore the database as of 02:35, you first restore the full backup, but leave the database in the recovering state. You then apply each of the transaction log backups (1,2,3) with a STOPAT 02:35 command. Each transaction log backup will then be able to restore the transactions for the time period it covers except the final one which will only restore 5 minutes of transactions (02:30 to 02:35 part of the 02:30 to 02:45 covered by log backup 3). Your final step is to issue a recover command to the database to conclude the restore process.

You can only take transaction log backups if the database is in the full or bulk-logged recovery model. The MASTER and MSDB databases are in simple recovery mode, so transaction logs cannot be made for them.

Once a log backup completes, part of the process is to remove all the completed transactions that were just backed up. This frees the space within the log for reuse later. It doesn't shrink the size of the log file itself. If your database is in Full

Recovery, the one way to free that space within the log is through the log backup process.

Tail-Log Backup

When your database is in full recovery, you will be taking regular log backups. Before you restore the database, in the event of some types of error where the database is still online and available, you should run one more log backup. This is referred to as the tail log backup. It captures all remaining committed transactions in the system so that you can recover to a point in time. It’s possible to restore the database without this log backup being taken. If the database is offline for any reason, you won’t be able to perform a tail log backup.

File and filegroup backups

Backing up a database through the full backup or even the differential backup process is a relatively lightweight and fast operation. However, it’s not free. When you get into very large database sizes (at least 500gb and greater), you will find that you need to break apart the storage of the database into multiple files and file groups. Once you have done that, you can take backups of the file/filegroup which will be faster and cause less load than backing up the entire database.

These work the same way as differential database backups. All the pages that have been modified on the file since the last full backup will be written off to the backup file. This is very useful for large databases that are relatively static.

Snapshots

A database snapshot is not technically a backup like the other types listed here. However, it is a mechanism for creating a copy of the backup that can be used to restore a database to an earlier moment in time. Snapshots are only available with Enterprise version of SQL Server. When a snapshot of the database is created, a read only copy of your database is created by simply recording pages that have been modified and supplying those pages, plus the original database pages as a means of having a database. You can then restore to the point prior to the pages being modified, the moment of the snapshot. No point in time recovery within the range of time that the snapshot existed is possible. This is extremely useful for situations like deployments where you need a temporary and fast backup that will also make for a very fast restore process. It doesn’t work at all well for disaster recovery scenarios.

Partial Database Backups

If you have broken up your database into files and filegroups and some of those file/filegroups are marked as read only, it becomes possible to backup each of the file/filegroups independently, backing up the read only objects once, and then backing up the non-read-only objects over and over. These independent file and filegroup backups are referred to as a partial backup. They can then be used in a partial restore process. This makes for faster backups and restore operations because you only have to worry about a portion of the database. But, most systems won't have read only portions making this a somewhat rare situation.

Performing Backups

The following table shows standard, basic, methods for performing different types of backups within SQL Server. The preferred method is to use T-SQL because of the direct control it confers. Use of the graphical user interface through the SQL Server Management Studio is included but not recommended.

Backup Type	TSQL	console
Full Backup	`BACKUP DATABASE [databasename] TO [device]`	1. Expand the Databases folder in the Object Explorer window. Right click on the database you wish to backup. Select Tasks then Backup from the context menu. 2. Verify that the right database is in the Source section in the drop down labeled Database. 3. The default is Full, so you'll need to create the Destination next. By default it will be set as Disk and a backup file will already be defined based on the server backup location defaults. The name of the file is defaulted to your database name.

		4. If you wish to change the location or the name of the backup, you'll need to first click the Remove button for the default value. You can then click the Add button to create your own
Transaction log	`BACKUP LOG [databasename] TO [device]`	As above, but on step 3 you'll need to change the backup type in the drop down to Log. NOTE: If your database is in the Simple recovery model, Log will not be a choice
Differential backup	`BACKUP DATABASE {databasename}TO {device} WITH DIFFERENTIAL`	As above, but in step 3 you'll need to change the backup type to Differential
Filegroup Backup	`BACKUP DATABASE [databasename] FILE = [filename],` --or `FILEGROUP = [filegroupname] TO [device]`	As above, but, you'll need to modify the Backup Component, which was not addressed above. You can check "Files and filegroups" instead of "Database." From there you can select which of the file or filegroups you wish to back up.
Filegroup Differential backup	`BACKUP DATABASE {databasename} FILEGROUP = {filegroup}TO {device} WITH DIFFERENTIAL`	Just as with the Differential backup, you'll change the type of backup.

Snapshot	`CREATE DATABASE [newdbname] ON (NAME = [logicalfilename], FILENAME = [filename]) AS SNAPSHOT OF [databasename]`	T-SQL only
Partial Backup	Same as FileGroup Backup	

System database backups

To restore a database server, the system databases will need a backup regime that includes at least the MASTER and MSDB databases.

MASTER
Must be backed up after creating, altering, or dropping a database (but not the objects within a database); altering a data or log file; changing logins; changing server configuration optons; altering linked servers, etc. Rather than trying to keep track of each and every change, schedule the master backup to be done regularly. It’s a very small database in most cases, so you can back it up frequently.

MSDB
Must be backed up to capture SQL Server Agent configurations.

Model
Only back this up if you modify it.

TEMPDB
Never back this up. Tempdb is recreated each time SQL Server is restarted.

Recovery models

There are three recovery models that can be set on each user database. The choice of recovery model drives decisions and requirements around the types of backups you’ll use. While setting the recovery model is strictly a technology option, the decisions for the type of recovery model to use are driven by the needs of the business in terms of the Recovery Point Objective outlined above.

You set the recovery model through the ALTER DATABASE command:

```
ALTER DATABASE [databasename] SET RECOVERY FULL/SIMPLE/BULK_LOGGED
```

By default, the model database, from which all other databases in the system are created, is set to FULL recovery. This means any new database you create will be in FULL recovery unless you choose a different recovery model while creating it or afterwards.

Simple
In simple recovery model, the committed transactions are removed from the log and the space is reclaimed at each checkpoint within the database. When in simple recovery, no log backups are possible. This means that with simple recovery, recovery to a point in time is not possible and all backups are retained in the full or differential backups only. Depending on the frequency of those backups, you could lose substantial amounts of data under simple recovery.

Bulk-logged
Bulk-logged recovery model requires that log backups be taken regularly in order to reclaim log space. This recovery model reduces the size of the transaction log by minimally logging some operations such as bulk inserts. The problem is, recovery to a point in time is not possible with this recovery model because any minimally logged operations cannot be restored. This means that bulk-logged has all the overhead of Full Recovery, but effectively works like Simple Recovery. Unless you have specific needs or are willing to perform lots of extra backup operations, it’s not recommended to use bulk-logged recovery.

Full
In full recovery you must run a log backup on a regular basis in order to reclaim log space. Recovery to a point in time is fully supported. For any production system that has data that is vital to the business, full recovery should be used.

How to Set the Recovery Model

Recovery model	**TSQL**	**console**
Full	`ALTER DATABASE {databasename}SET RECOVERY FULL`	Open the 'Databases' folder. Once the database folder is expanded, right click on the database and select the 'Properties' option. The 'Database Properties' window will open. Click on the 'Options' tab and the recovery model will be listed in the middle of the screen. Click on the drop down box to select the needed recovery model. On the bottom of the screen click 'OK' to save the Recovery Model
Simple	`ALTER DATABASE {databasename}SET RECOVERY SIMPLE`	
Bulk Logged	`ALTER DATABASE {databasename}SET RECOVERY BULK_LOGGED`	

Replicated databases

Replicated databases (both system and user) require different strategies for backup which are beyond the scope of this article. Please refer to Backing up and Restoring Replicated Databases in BOL for details at Back Up and Restore Replicated Databases.

Backup to URL

In addition to backing up to a file, with SQL Server 2014, it's possible to backup to a file using a URL to Windows Azure Blob Storage. This provides a means of both backing up your databases and backing them up to an offsite storage location in order to provide added protection to your businesses data. For more details, check out the Books Online entry (SQL Server Backup to URL).

Striped backups

In order to reduce the time taken to perform a backup, SQL Server can write in parallel to up to 32 different devices (physical files), grouped together as a "Media Set". (A media family is a set of backup units grouped together to make one logical device.) To do the backup, the devices must be defined within a backup device, and must all be of the same type.

Backup History

The history of every SQL Server backup is written to the MSDB database. This can be accessed easily via using the following system views.

dbo.backupfile
Contains one row for each data or log file that is backed up

dbo.backupmediafamily
Contains one row for each media family

dbo.backupmediaset
Contains one row for each backup media set

dbo.backupset
Contains a row for each backup set

dbo.backupfilegroup
Contains one row for each filegroup in a database at the time of backup

dbo.logmarkhistory
Contains one row for each marked transaction that has been committed

Backup File Information

It is possible to look at a backup file itself to get information about the backup. The header of the backup stores data like when the backup was created, what type of backup it is, the user who took the backup and all other sorts of information. The basic commands available are:

LABELONLY	`RESTORE LABELONLY FROM DISK = 'somepath'`	Simple overview of what is on the backup file in question
HEADERONLY	`RESTORE HEADERONLY FROM DISK = 'somepath'`	A very thorough listing of all the information contained within the header.
FILELISTONLY	`RESTORE FILELISTONLY FROM DISK = 'somepath'`	A listing of the files, their locations and their types that defined the database that was backed up. This is useful when you need to restore a database on a different server.

For more details on the use of these commands, consult the Books Online and the article How to Get Information About Your Backups.

Basic Backup Process

The best way to set up your backup processes is to automate them using some type of scheduler and T-SQL. SQL Agent is the built-in scheduling process within SQL Server. You can schedule standard commands for your databases. This represents a fundamental approach to a backup schedule that ensures a decent recovery plan:

- **Sunday Night**: Full Backup and database consistency check
- **Monday-Saturday:** Differential Backup
- **Midnight-11:45PM:** Log Backup every 15 minutes

With this approach, you're ensuring that the business will lose no more than 15 minutes of data since you have a full backup, a differential backup for each day of the week, and log backups, providing a Recovery Point Objective. Have enough disk space to keep, at minimum, the last full backup, the last two differential backups, the last 48 hours' worth of log backups. This means, in a worst case scenario, you can restore the full backup, a differential and then up to 192 log restores (assuming the failure occurred in the last 15 minutes of the day), establishing a Recovery Time Objective.

Further, a testing process should be set up for your backups. The main point is not to backup your databases, but to recover your databases in the event of a failure. Where possible, you should have a second server available to automate running restores of, at minimum, the full backup and the differential backups. Further, you should practice regularly doing a point in time restore with the log backups to ensure you know how to do it when in an emergency situation.

Backup Compression

Backup compression should be used, where available, for all backups. Compression actually speeds up the backup process by sacrificing CPU, one of the most available resources on most systems, to reduce the amount of I/O necessary to write to disk, the most used resource on most systems. Native compression is available in SQL Server 2008 Enterprise or SQL Server 2008R2 Standard or Enterprise and all higher versions of SQL Server. SQL Server 2014 introduces encryption in addition to compression.

Managed Backup

Introduced in SQL Server 2014, Managed Backup is a process that backs up your database and logs to Azure Blob Storage in a completely automated fashion based on both time and the amount of transactions you have processed in your system. This is not for everyone, especially if you have very large databases, or very poor internet connectivity. But if you want to set up a basic backup process and you have little knowledge of best practices or ability with T-SQL, this is a good alternative. For an introduction to the concepts, read this blog post; How to Set Up Managed Backups in SQL Server 2014.

Review Questions

- What is the difference between a differential and and transaction log backup operation?
- What operations would make a backup of the master database a good idea?
- What is a filegroup?
- What is the drawback to backing up a database merely by backing up its data files?
- When might one consider using a file differential backup?
- Why would you need to initiate a backup manually when you already have an automated backup in place?
- Why can't you do a point-in-time recovery from a backup to a bulk-logged database?

Related Content:

For more articles on SQL Backup and Restore, see here Simple Talk Backup and Recovery