Database backup and restore

Overview

The main purpose of backing up the Verba database is to provide increased reliability and system recovery (fault tolerance) of the system. In case of a media or database failure, data can be restored from earlier database backups.

It is important to note, that the Verba media files (recordings) are stored in the file system, and backing up the database does not provide fault tolerance for the media files. Verba administrators have to choose another method to backup media files regularly to provide the reliability of the system. The Verba media file backup procedures have to be synchronized with the Verba database backup in order to provide the ability to restore a consistent Verba system in case of a failure.

Microsoft SQL Server provides high-performance backup and restore capabilities. The SQL Server backup and restore component provides an essential safeguard for protecting critical data stored in SQL Server databases. Implementing a well-planned backup and restore strategy helps protect databases against data loss because of damage caused by a variety of failures. Testing your strategy by restoring a set of backups and recovering your database prepares you to respond effectively to a disaster.

The following topics provide a brief explanation of the most important backup topics.

For detailed information about SQL Server backup and restore features, refer to https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases.

This page has the following content:

For further information:

- Creating a one-off full database backup
- Scheduling database backups
- Restoring a full database backup
- Changing the database recovery model

Selecting a recovery model

SQL Server provides three recovery models to:

- Simplify recovery planning.
- Simplify backup and recovery procedures.
- Clarify tradeoffs between system operational requirements.

Each of these models addresses different needs for performance, disk and tape space, and protection against data loss. For example, when you choose a recovery model, you must consider the tradeoffs between the following business requirements:

- Performance of large-scale operation (for example, index creation or bulk loads).
- Data loss exposure (for example, the loss of committed transactions).
- Transaction log space consumption.
- The simplicity of backup and recovery procedures.

Depending on what operations you are performing, more than one model may be appropriate. After you have chosen a recovery model or models, plan the required backup and recovery procedures.

This table provides an overview of the benefits and implications of the three recovery models.

Recovery models comparison table

Recovery model	Benefits	Work loss exposure	Recover to point in time?

Simple	Permits high-performance bulk copy operations. Reclaims log space to keep space requirements small.	Changes since the most recent database or differential backup must be redone.	Can recover to the end of any backup. Then changes must be redone.
Full	No work is lost due to a lost or damaged data file. Can recover to an arbitrary point in time (for example, prior to application or user error).	Normally none. If the log is damaged, changes since the most recent log backup must be redone.	Can recover to any point in time.
Bulk-Logged	Permits high-performance bulk copy operations. Minimal log space is used by bulk operations.	If the log is damaged, or bulk operations occurred since the most recent log backup, changes since that last backup must be redone. Otherwise, no work is lost.	Can recover to the end of any backup. Then changes must be redone.

Simple Recovery

Simple Recovery requires the least administration. In the Simple Recovery model, data is recoverable only to the most recent full database or differential backup. Transaction log backups are not used and minimal transaction log space is used. After the log space is no longer needed for recovery from server failure, it is reused.

The Simple Recovery model is easier to manage than the Full or Bulk-Logged models, but at the expense of higher data loss exposure if a data file is damaged.

Simple Recovery is not an appropriate choice for production systems where the loss of recent changes is unacceptable.

When using Simple Recovery, the backup interval should be long enough to keep the backup overhead from affecting production work, yet short enough to prevent the loss of significant amount of data.

The backup strategy for simple recovery consists of:

- Database backups.
- Differential backups (optional).

To recover in the event of media failure:

- Restore the most recent full database backup.
- If differential backups exist, restore the most recent one. Changes since the last database or differential backup are lost.

Full and Bulk-Logged Recovery

Full Recovery and Bulk-Logged Recovery models provide the greatest protection for data. These models rely on the transaction log to provide full recoverability and to prevent work loss in the broadest range of failure scenarios. The Full Recovery model provides the most flexibility for recovering databases to an earlier point in time.

The Bulk-Logged model provides higher performance and lower log space consumption for certain large-scale operations (for example, create index or bulk copy). It does this at the expense of some flexibility of point-in-time recovery.

The backup strategy for full recovery consists of:

- Database backups.
- Differential backups (optional).
- Transaction log backups.

Full and bulk-logged recovery are similar and many users of the Full Recovery model will use the Bulk-Logged model on occasion.

You can restore a database to the state it was in at the point of failure if the current transaction log file for the database is available and undamaged.

To restore the database to the point of failure:

- Back up the currently active transaction log.
- Restore the most recent database backup without recovering the database.
- If differential backups exist, restore the most recent one.
- Restore each transaction log backup created since the database or differential backup in the same sequence in which they were created without recovering the database.

- Apply the most recent log backup (created in), and recover the database.

The backup strategy for bulk-logged recovery consists of:

- Database backups.
- Differential backups (optional).
- Log backups.

Backing up a log that contains bulk-logged operations requires access to all data files in the database. If the data files are not accessible, the final transaction log cannot be backed up and all committed operations in that log will be lost.

To recover in the event of media failure:

- Back up the currently active transaction log.
- Restore the most recent full database backup.
- If differential backups exist, restore the most recent one.
- Apply in sequence all transaction log backups created since the most recent differential or full database backup.
- Manually redo all changes since the most recent log backup

Database backups

Full database backups

A database backup creates a duplicate of the data that is in the database when the backup completes. This is a single operation, usually scheduled at regular intervals. Database backups are self-contained.

You can re-create the entire database from a database backup in one step by restoring the database. The restore process overwrites the existing database or creates the database if it does not exist. The restored database will match the state of the database at the time the backup completed, minus any uncommitted transactions. Uncommitted transactions are rolled back when the database is recovered.

A database backup uses more storage space per backup than transaction log and differential database backups. Consequently, database backups need more time to complete the backup operation and so are typically created less frequently than a differential database or transaction log backups.

Differential database backups

A differential database backup records only the data that has changed since the last database backup. You can make more frequent backups because differential database backups are smaller and faster than database backups. Making frequent backups decreases your risk of losing data.

If you have created any file backups since the last full database backup, those files will be scanned by SQL Server at the beginning of a differential database backup. This may cause some degradation of performance in the differential database backup.

You use differential database backups to restore the database to the point at which the differential database backup was completed. To recover to the exact point of failure, you must use transaction log backups.

Consider using differential database backups when:

- Only a relatively small portion of the data in the database has changed since the last database backup. Differential database backups are particularly effective if the same data is modified many times.
- You are using the Simple Recovery model and want more frequent backups, but don't want to do frequent full database backups.
- You are using the Full or Bulk-Logged Recovery model and want to minimize the time it takes to roll forward transaction log backups when restoring a database.

A recommended process for implementing differential database backups is:

- Create regular database backups.
- Create a differential database backup periodically between database backups, such as every four hours or more for highly active systems.
- If using Full or Bulk-Logged Recovery, create transaction log backups more frequently than differential database backups, such as every 30 minutes.

The sequence for restoring differential database backups is:

- Restore the most recent database backup.
- Restore the last differential database backup.

- Apply all transaction log backups created after the last differential database backup was created if you use Full or Bulk-Logged Recovery.