Restore a SQL Server Database to a Point in Time (Full Recovery Model)

03/17/2017 • 8 minutes to read • Contributors

In this article

Before You Begin
Using SQL Server Management Studio
Using Transact-SQL
Related Tasks
See Also

THIS TOPIC APPLIES TO: SQL Server Azure SQL Database Azure SQL Data Warehouse Parallel Data Warehouse

This topic describes how to restore a database to a point in time in SQL Server 2017 by using SQL Server Management Studio or Transact-SQL. This topic is relevant only for SQL Server databases that use the full or bulk-logged recovery models.

> **Important**
>
> Under the bulk-logged recovery model, if a log backup contains bulk-logged changes, point-in-time recovery is not possible to a point within that backup. The database must be recovered to the end of the transaction log backup.

- **Before you begin:**

  Recommendations

  Security

- **To restore a SQL Server database to a point in time, using:**

  SQL Server Management Studio

FEEDBACK

Transact-SQL

Before You Begin

Recommendations

- Use STANDBY to find unknown point in time.
- Specify the point in time early in a restore sequence

Security

Permissions

If the database being restored does not exist, the user must have CREATE DATABASE permissions to be able to execute RESTORE. If the database exists, RESTORE permissions default to members of the **sysadmin** and **dbcreator** fixed server roles and the owner (**dbo**) of the database (for the FROM DATABASE_SNAPSHOT option, the database always exists).

RESTORE permissions are given to roles in which membership information is always readily available to the server. Because fixed database role membership can be checked only when the database is accessible and undamaged, which is not always the case when RESTORE is executed, members of the **db_owner** fixed database role do not have RESTORE permissions.

Using SQL Server Management Studio

To restore a database to a point in time

1. In Object Explorer, connect to the appropriate instance of the SQL Server Database Engine, and expand the server tree.
2. Expand **Databases**. Depending on the database, either select a user database or expand **System Databases**, and then select a system database.
3. Right-click the database, point to **Tasks**, point to **Restore**, and then click **Database**.
4. On the **General** page, use the **Source** section to specify the source and location of the backup sets to restore. Select one of the following options:

   - **Database**

     Select the database to restore from the drop-down list. The list contains only databases that have been backed up according to the **msdb** backup history.

> ⓘ **Note**
>
> If the backup is taken from a different server, the destination server will not have the backup history information for the specified database. In this case, select **Device** to manually specify the file or device to restore.

- **Device**

  Click the browse (**...**) button to open the **Select backup devices** dialog box. In the **Backup media type** box, select one of the listed device types. To select one or more devices for the **Backup media** box, click **Add**.

  After you add the devices you want to the **Backup media** list box, click **OK** to return to the **General** page.

  In the **Source: Device: Database** list box, select the name of the database which should be restored.

  Note This list is only available when **Device** is selected. Only databases that have backups on the selected device will be available.

5. In the **Destination** section, the **Database** box is automatically populated with the name of the database to be restored. To change the name of the database, enter the new name in the **Database** box.

6. Click **Timeline** to access the **Backup Timeline** dialog box.

7. In the **Restore to** section, click **Specific date and time**.

8. Use either the **Date** and **Time** boxes or the slider bar to specify a specific date and time to where the restore should stop. Click **OK**.

   > ⓘ **Note**
   >
   > Use the **Timeline Interval** box to change the amount of time displayed on the timeline.

9. After you have specified a specific point in time, the Database Recovery Advisor ensures that only backups that are required for restoring to that point in time are selected in the **Restore** column of the **Backup sets to restore** grid. These selected backups make up the recommended restore plan for your point-in-time restore. You should use only the selected backups for your point-in-time restore operation.

   For information about the columns in the **Backup sets to restore** grid, see Restore Database (General Page). For information about the Database Recovery Advisor, see Restore and Recovery Overview (SQL Server).

10. On the **Options** page, in the **Restore options** panel, you can select any of the following options, if appropriate for your situation:

    - **Overwrite the existing database (WITH REPLACE)**
    - **Preserve the replication settings (WITH KEEP_REPLICATION)**
    - **Restrict access to the restored database (WITH RESTRICTED_USER)**

      For more information about these options, see Restore Database (Options Page).

11. Select an option for the **Recovery state** box. This box determines the state of the database after the restore operation.

    - **RESTORE WITH RECOVERY** is the default behavior which leaves the database ready for use by rolling back the uncommitted transactions. Additional transaction logs cannot be restored. Select this option if you are restoring all of the necessary backups now.
    - **RESTORE WITH NORECOVERY** which leaves the database non-operational, and does not roll back the uncommitted transactions. Additional transaction logs can be restored. The database cannot be used until it is recovered.
    - **RESTORE WITH STANDBY** which leaves the database in read-only mode. It undoes uncommitted transactions, but saves the undo actions in a standby file so that recovery effects can be reverted.

      For descriptions of the options, see Restore Database (Options Page).

12. **Take tail-log backup before restore** will be selected if it is necessary for the point in time that you have selected. You do not need to modify this setting, but you can choose to backup the tail of the log even if it is not required.

13. Restore operations may fail if there are active connections to the database. Check the **Close existing connections option** to ensure that all active connections between Management Studio and the database are closed. This check box sets the database to single user mode before performing the restore operations, and sets the database to multi-user mode when complete.

14. Select **Prompt before restoring each backup** if you wish to be prompted between each restore operation. This is not usually necessary unless the database is large and you wish to monitor the status of the restore operation.

Using Transact-SQL

Before you begin

A specified time is always restored from a log backup. In every RESTORE LOG statement of the restore sequence, you must specify your target time or transaction in an identical STOPAT clause. As a prerequisite to a point-in-time restore, you must first restore a full database backup whose end point is earlier than your target restore time. That full database backup can be older than the most recent full database backup as long as you then restore every subsequent log backup, up to and including the log backup that contains your target point in time.

To help you identify which database backup to restore, you can optionally specify your WITH STOPAT clause in your RESTORE DATABASE statement to raise an error if a data backup is too recent for the specified target time. The complete data backup is always restored, even if it contains the target time.

Basic Transact-SQL syntax

RESTORE LOG *database_name* FROM <backup_device> WITH STOPAT **=** *time***,** RECOVERY...

The recovery point is the latest transaction commit that occurred at or before the **datetime** value that is specified by *time*.

To restore only the modifications that were made before a specific point in time, specify WITH STOPAT **=** *time* for each backup you restore. This makes sure that you do not go past the target time.

To restore a database to a point in time

> ⓘ **Note**
>
> For an example of this procedure, see Example (Transact-SQL), later in this section.

1. Connect to server instance on which you want to restore the database.

2. Execute the RESTORE DATABASE statement using the NORECOVERY option.

   > ⓘ **Note**
   >
   > If a partial restore sequence excludes any FILESTREAM filegroup, point-in-time restore is not supported. You can force the restore sequence to continue. However the FILESTREAM filegroups that are omitted from your RESTORE statement can never be restored. To force a point-in-time restore, specify the CONTINUE_AFTER_ERROR option together with the STOPAT, STOPATMARK, or STOPBEFOREMARK option, which you must also specify in your subsequent RESTORE LOG statements. If you specify CONTINUE_AFTER_ERROR, the partial restore sequence succeeds and the FILESTREAM filegroup becomes unrecoverable.

3. Restore the last differential database backup, if any, without recovering the database (RESTORE DATABASE *database_name* FROM *backup_device* WITH NORECOVERY).

4. Apply each transaction log backup in the same sequence in which they were created, specifying the time at which you intend to stop restoring log (RESTORE DATABASE *database_name* FROM <backup_device> WITH STOPAT= *time*, RECOVERY).

> **Note**
>
> The RECOVERY and STOPAT options. If the transaction log backup does not contain the requested time (for example, if the time specified is beyond the end of the time covered by the transaction log), a warning is generated and the database remains unrecovered.

Example (Transact-SQL)

The following example restores a database to its state as of `12:00 AM` on `April 15, 2020` and shows a restore operation that involves multiple log backups. On the backup device, `AdventureWorksBackups`, the full database backup to be restored is the third backup set on the device (`FILE = 3`), the first log backup is the fourth backup set (`FILE = 4`), and the second log backup is the fifth backup set (`FILE = 5`).

> **Important**
>
> The **AdventureWorks2012** database uses the simple recovery model. To permit log backups, before taking a full database backup, the database was set to use the full recovery model, using `ALTER DATABASE AdventureWorks SET RECOVERY FULL`.

Copy

```
RESTORE DATABASE AdventureWorks
   FROM AdventureWorksBackups
   WITH FILE=3, NORECOVERY;

RESTORE LOG AdventureWorks
   FROM AdventureWorksBackups
   WITH FILE=4, NORECOVERY, STOPAT = 'Apr 15, 2020 12:00 AM';

RESTORE LOG AdventureWorks
   FROM AdventureWorksBackups
   WITH FILE=5, NORECOVERY, STOPAT = 'Apr 15, 2020 12:00 AM';
RESTORE DATABASE AdventureWorks WITH RECOVERY;
GO
```

Related Tasks

- Restore a Database Backup Using SSMS

- Back Up a Transaction Log (SQL Server)
- Restore a Database to the Point of Failure Under the Full Recovery Model (Transact-SQL)
- Restore a Database to a Marked Transaction (SQL Server Management Studio)
- Recover to a Log Sequence Number (SQL Server)
- ToPointInTime (SMO)

See Also

backupset (Transact-SQL)
RESTORE (Transact-SQL)
RESTORE HEADERONLY (Transact-SQL)

> **Note**
>
> The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, we invite you to read our blog post.

12 Comments

Sign in **4 people listening**

+ Follow

Post comment as...

Mohammed_naufal Dec 28, 2017

```
RESTORE DATABASE AdventureWorks
 FROM AdventureWorksBackups
   WITH FILE=3, NORECOVERY;

RESTORE LOG AdventureWorks
   FROM AdventureWorksBackups
   WITH FILE=4, NORECOVERY, STOPAT = 'Apr 15, 2020 12:00 AM';

RESTORE LOG AdventureWorks
```