



Global Knowledge®

Expert Reference Series of White Papers

Restoring Data with Database Snapshots

Restoring Data with Database Snapshots

Neil Tucker, MCT, MCITP, MCDBA, MCTS, MCSE

Introduction

What DBA wouldn't appreciate a process that allows for fast and easy restoration of lost data? If the process allows you and your users to examine the information before restoring it, to use a minimal amount of drive space, and to carry out all the necessary steps with only local resources, then you have a solution that not only saves you time, but improves the options you have for supporting users and developers.

All of these options are available when using SQL Server's Database Snapshot functionality. When configured, it gives you access to a point-in-time, read-only view of the database and its records. Multiple snapshots can be created on the same database. Database snapshots require the original database to function, and you will be required to delete them before you can remove the original database.

Snapshots use a copy-on-write operation that minimizes the drive space required for each snapshot. Unless there are changes in the original database, the snapshots will remain empty. Queries to the snapshot for unchanged records will automatically be redirected to the original database. Queries to the snapshot for modified records will show the data as it existed at the time of the snapshot's creation. Modified records are copied from the original database before they are changed. The copy-on-write process copies data at the page level, so some unchanged records will likely end up in the snapshot.

How To Use Database Snapshots

Database Snapshots require the Enterprise Edition of SQL Server and can only be created using T-SQL. The steps below show you how to do this on a test server. You will need the Enterprise or Developer editions of SQL Server 2005 or 2008. To use the scripts provided, create two folders on the root of the C: drive named **Data-base** and **Snapshot**.

- 1. Create the test database.** Use the script in Figure 1 (below) to create a database named **DB1** and a table named **dbo.Contacts** with five (5) records.

```
/* Create a new database named DB1 */
Use Master
GO
IF Exists (Select Name From sys.databases Where Name = 'db1')
Drop Database DB1
```

```

GO
Create Database DB1 on Primary
( Name='DB1_Data',Filename='C:\Database\DB1.mdf')
LOG ON
( Name='DB1_log',Filename='C:\Database\DB1_log.ldf')
GO

/* Create a table named Contacts in DB1 */
USE DB1
GO
Create Table dbo.Contacts
(ID nchar(5) NOT NULL,
FirstName nvarchar(50),
LastName nvarchar(50),
Constraint PK_Contacts Primary Key Clustered (ID Asc))
GO

/* Insert 5 records into the Contacts Table */
Insert DB1.dbo.Contacts
Values('101','John','Harrison')
Insert DB1.dbo.Contacts
Values('102','Jessica','Forthwright')
Insert DB1.dbo.Contacts
Values('103','Earl','Russell')
Insert DB1.dbo.Contacts
Values('104','Stanley','McDonald')
Insert DB1.dbo.Contacts
Values('105','Mary','Kellerman')
GO

```

2. Create a snapshot of DB1. Use the Figure 2 script to create a snapshot of the database. After it is created, verify that the snapshot file was created (C:\Snapshot\DB1_Snapshot.ss). Notice that no log file is needed. This is a read-only version of DB1 as it existed at the time it was created. In the Figure 2 script, notice that the **NAME** parameter points to the actual name of the database file and not the name of the database as specified in the **AS SNAPSHOT OF** parameter. Databases with multiple data files would, therefore, need multiple **NAME** parameters, one for each file.

```

/* Create a snapshot of the DB1 database */
USE master
GO

```

```

CREATE DATABASE DB1_Snapshot ON
( NAME = DB1_Data, FILENAME =
'C:\Snapshot\DB1_Snapshot.ss' )
AS SNAPSHOT OF DB1;
GO

```

3. Change records in DB1. Modify any record from the Contacts table in the DB1 database. Verify that the information in the DB1_Snapshot database is unchanged, using the script provided in Figure 3. The changes to DB1 have caused the original version of the data pages changed to be written to the DB1_Snapshot.ss file.

```

/* Verify that the information in the
snapshot and the original database are the same. */
Use DB1
Select * From DB1.dbo.Contacts
Select * From DB1_Snapshot.dbo.Contacts
Go

/* Update the records in the Contacts table and compare
it to the Snapshot version again */
Use DB1
Go
--Insert dbo.Contacts
--Values('106','Chris','Andrews')
--Go
Update dbo.Contacts
set FirstName='Jonathan'
where ID=101
Go
Delete dbo.Contacts
where ID=102
Go
/* Verify that the information in the
snapshot and the original database are different. */
Select * From DB1.dbo.Contacts
Select * From DB1_Snapshot.dbo.Contacts
Go

```

4. Revert to the original version of the table. You can restore individual records in a table from the snapshot. Use the script in Figure 4 to see how to fix incorrect deletions or updates.

```

/* Reverse deletions and updates
   using information in the snapshot */
Use DB1
go
Insert dbo.Contacts
Select * From DB1_Snapshot.dbo.Contacts
Where ID=102
Go
Update C
Set C.FirstName = S.FirstName
From DB1.dbo.Contacts C
Inner Join DB1_Snapshot.dbo.Contacts S
ON C.ID = S.ID
Where C.ID = 101

/* Verify that the records in DB1 and the
   snapshot are the same */
Select * From DB1.dbo.Contacts
Select * From DB1_Snapshot.dbo.Contacts
Go

```

5. Revert to the original database. As long as the original database is still intact and online, you can restore all changes made since the snapshot. These database snapshot restores cannot be done when more than one snapshot exists. As with a normal database restore, exclusive access to the database is needed during this operation. To test this process, use Figure 3 script to create differences between both databases, then use Figure 5 to restore the entire database from the snapshot.

```

-- Restore the DB1 database from the snapshot
-- The restore operation will fail if there are other connections
to the DB1 database

Use Master
Go
Restore Database DB1 From
Database_Snapshot = 'DB1_Snapshot'
Go
Select * From DB1.dbo.Contacts
Select * From DB1_Snapshot.dbo.Contacts

```

Caveats when Using Database Snapshots

Like any other useful DBA tool, database snapshots can be overused or misused. Here are a few precautions to keep in mind before implementing this functionality in a production environment.

Backups are still required: There is no substitute for regular, tested backups that are stored in a remote, secure location. The database snapshot requirement that the files be stored on the same server as the original database prevents them from being used in this manner. The fact that the snapshots become obsolete if the original database is lost also limits their use for this purpose. The strength of snapshots is in their ability to provide point-in-time views of your data and ad-hoc restore options.

Beware of too many snapshots: Although the empty state of a new snapshot file makes their size negligible, take care when creating multiple snapshots on the same database. Theoretically, a database with 10 GB of data and 24 snapshots can suddenly and unexpectedly require up to 250 GB of drive space. The ability to quickly restore a database to how it looked at a particular hour of the day is very convenient, but if this functionality is not needed, the additional resource requirements can tax a mission-critical server beyond acceptable levels. A snapshot should only be created to meet specific administrative needs and then deleted when it is no longer necessary.

Using Snapshots for Read-only Operations: The point-in-time, read-only nature of snapshots makes them a candidate for reporting or similar solutions. Using them in this way, however, limits their use as an ad-hoc restoration solution. Before allowing others to use your snapshots for development or business solutions, consider how this will affect the options available to you when administering your databases. Other dedicated read-only solutions, like **Log Shipping** might be considered.

Conclusion

If you decide to take advantage of Database Snapshots, keep its limitations in mind. It cannot replace your backup solution, and you must carefully consider the additional resource demands of each additional snapshot. It can, however, provide a fast and simple solution for viewing point-in-time data and restoring lost information from databases. Overall, it's a useful tool for reducing downtime and improving the availability of your databases.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge. Check out the following Global Knowledge course(s):

[Developing and Implementing a SQL Server 2008 Database \(M6232\)](#)

[SQL Server 2008 for Administration \(M6231, M6232\)](#)

For more information or to register, visit www.globalknowledge.com or call **1-800-COURSES** to speak with a sales representative.

Our courses and enhanced, hands-on labs and exercises offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 1,200 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and business training needs.

About the Author

Neil Tucker MCT, MCITP, MCDBA, MCSE, MCDST. Neil is a technical trainer, writer, and consultant with 16 years of experience in the IT industry. He has authored white papers and books on SQL Server, SharePoint, Windows Server, and Windows 7.