http://www.sqlservercentral.com/articles/Database+Snapshots/63004/
Printed 2008/06/05 06:35PM

Database Snapshots

By Basit Ali Farooq, 2008/05/03

Introduction

Microsoft introduced database snapshots with SQL Server 2005. A database snapshot is a static, read-only copy of a source database that gives you a consistent view of a database as of a certain point in time. You can use the snapshot for operations supported by a read-only database, such as reporting applications. You can create multiple database snapshots on a source database. You can recover the source database to the point when the snapshot was created from a database snapshot. However, when recovering from a snapshot, the source database must be available.

Database snapshots are available only in the Enterprise Edition of Microsoft SQL Server 2005 and all recovery models support database snapshots.

In this article I will explain how exactly this new feature works, why you may or may not consider using it, and how to get started using it.

Snapshot file structure

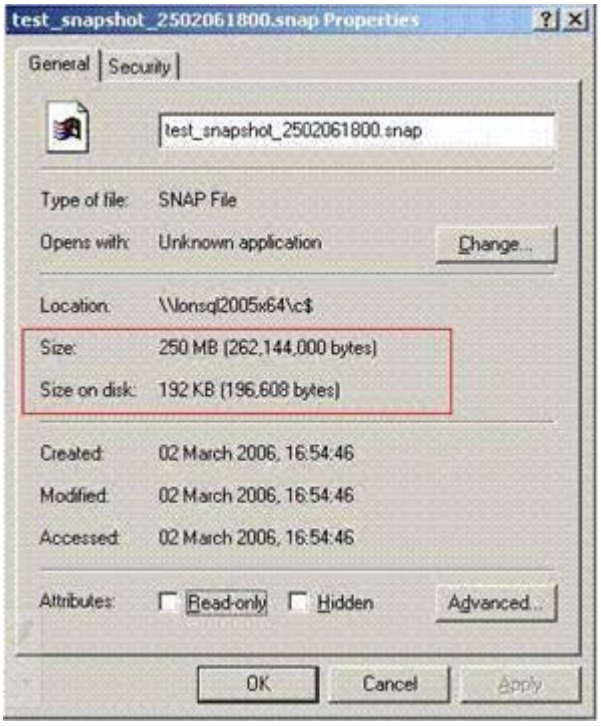
Database snapshots manage database data at the page level. SQL Server doesn't physically create a separate database file when you create a database snapshot. Instead, SQL Server creates a special file called a sparse file. Sparse file filenames are specified when you create the snapshot, and the initial file contains no data and is allocated minimal disk space. Space is allocated to the sparse file as needed.

As a page is modified in the source database, the original (unmodified) version of the page file is written to the sparse file. The process of writing changed files to the sparse file is known as copy-on-write. This allows SQL Server to create a record of the database as it existed at the moment of creation in the sparse file. Unmodified pages are stored only in the source database. Because of this, the source database must be available when you recover a database from a snapshot.

You can retrieve the actual file size of the database snapshot sparse file with the fn_virtualfilestats system table-valued functon the sysntax of the command is:

```
fn_virtualfilestats (databaseID | NULL, fileID | NULL)
```

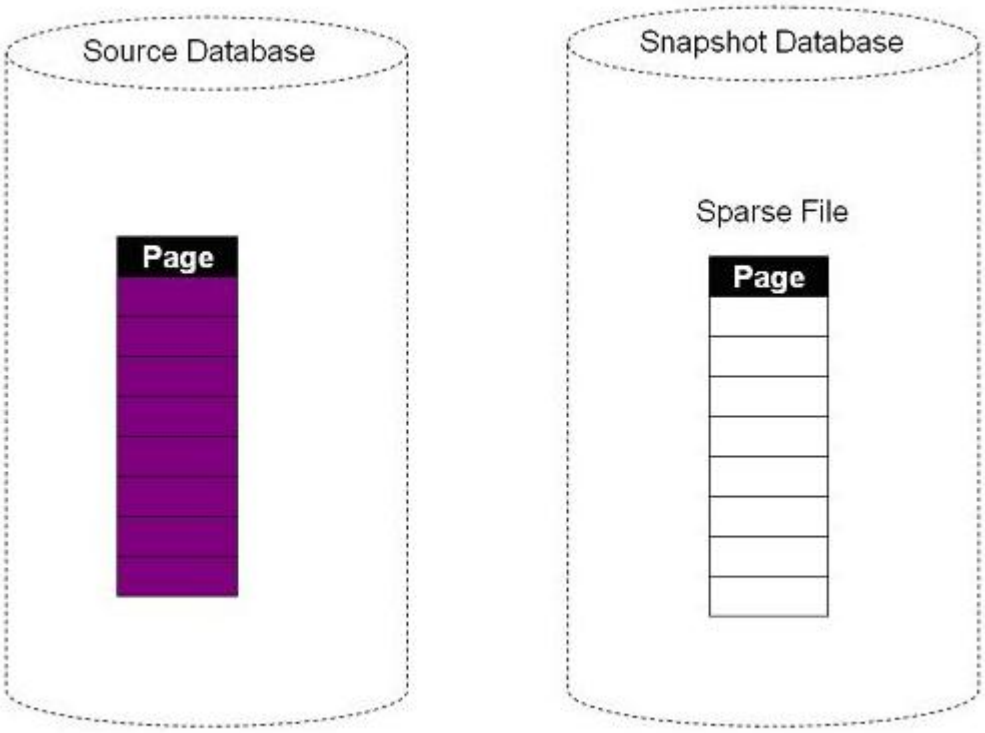
To find the maximum sparse file size, locate the file in the Windows file system and view the Windows file properties as shown in the following figure:



The maximum size isalso saved in sys.database_files and sys.master_files system tables

How Database Snapshots Work?

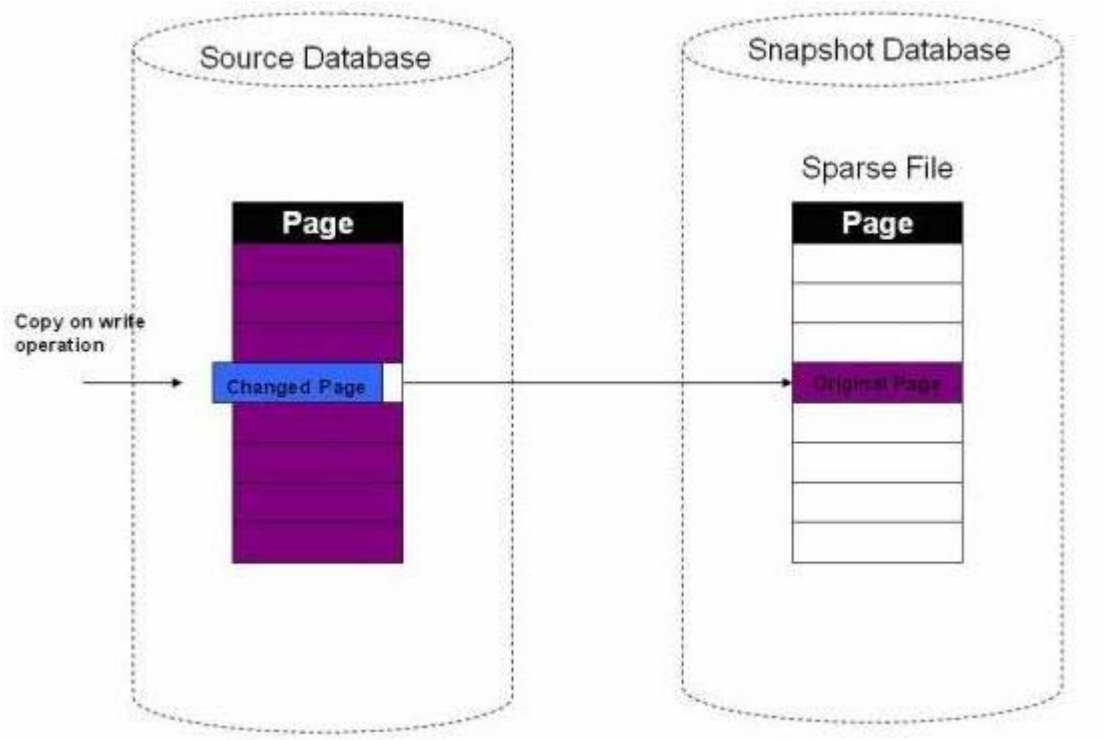
When the snapshot is first created, the sparse file is empty as shown below.



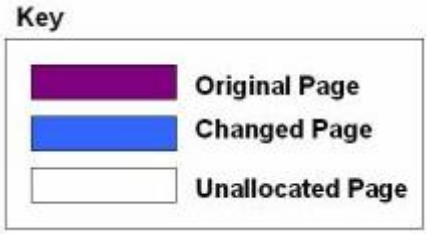
Case 1: When the database has no updations since the snapshot was taken

When the first update operation happens, the original data is copied into the sparse file. The database

engine then updates the data in the source database.



Case 2: When the database has updations since the snapshot was taken



During read operations on the snapshot, the original data is read from the source database. For the data that has been changed, the snapshot reads the information from the sparse file.

Using database snapshots

You can use a database snapshot in nearly any situation where you could use the database file, except when you require write access. Typical uses include:

- When you query a snapshot, you will not experience blocking due to update/insert operations that you might have to contend with when querying the source database (assuming snapshot isolation is not in use).
- Initially the snapshot data files are small and are very quick to create. They should only become large if your database is subject to frequent modifications.
- Data source for reporting applications.
- Data source for data export export operations.
- Historic data source.
- Redundant file copy for data redundancy or archive.
- File copy for test and validations.

Database snapshot limits

After you create a database snapshot, and while the snapshot exists, there are limits on what you can do with the source database. You can't:

- move the snapshot to another server. Because of the reliance on the source database, the snapshot has to exist on the same server as that source database which in our case is fine.
- include data changes from transactions in process. Snapshots are read-only and doesn't include source database uncommitted transactions.
- keep the snapshot if you must restore the source. If the source database becomes unavailable, all snapshots of the database become unavailable so can't drop or detach the source database
- separate performance. Where data pages have not changed you will be accessing the source database file, which may cause contention at the file level since both the source database and the snapshot will be accessing the same MDF.
- prevent the overhead of the snapshot. Every update/insert transaction on the source server potentially bears the overhead of the page copy operation to maintain the snapshot(s).
- You cannot grant a new user access to the data in a snapshot. Permissions are inherited form the source database as it existed at the time of snapshot creation and subsequent changes to security in the source database do not filter down to the snapshots.
- Full text indexes are not available on snapshots so if you require full text searching for your reporting then snapshots are not an option.

Managing database snapshots

Your only limit on creating database snapshots is available disk space. As a way of avoiding confusion, use database snapshot names that identify the:

- Source database.
- File as a snapshot.
- Date and time when you created the snapshot.

Creating snapshots

You can't create database snapshots with SQL Server Management Studio. You use the CREATE DATABASE command with the AS SNAPSHOT OF clause. The general syntax is:

```
CREATE DATABASE database_snapshot_name ON
(NAME = source_logical_filename,
FILENAME = operating_system_file)
AS SNAPSHOT OF source_database_name;
```

You must specify a snapshot file for each database file, except the transaction log file, any offline files, or the files in the process of being restored. When specifying the snapshot file, you must:

- Include each logical file in the source database.
- Specify a unique path and filename for the snapshot file.
- Enclose the operating system path and filename in quotes.
- Enclose each logical file/physical file pair in parentheses.
- Separate multiple files by commas.

For example, the following statement creates a snapshot for a database with a single data file:

```
CREATE DATABASE TestA_snap ON
(NAME = TEST,
FILENAME = 'C:\Snaps\snap.ssf')
```

```
AS SNAPSHOT OF TEST
```

The following statement creates a snapshot with multiple data files:

```
CREATE DATABASE TestB_snap_morn ON
(NAME = Spice,
FILENAME = 'C:\Snaps\snapmornA.ssf'),
(NAME = Spiceix,
FILENAME = 'C:\Snaps\snapmornB.ssf')
AS SNAPSHOT OF TestDb
```

Viewing snapshots

You can view database snapshots in SQL Server Management Studio. To view all snapshots on an instance, expand Databases > Database Snapshots. Expand the snapshot itself to view database objects in the snapshot.

Dropping snapshots

You can drop a snapshot with SQL Server Management Studio or DROP DATABASE command. This operations doesn't affect the source database. To use SQL Server Management Studio,Expand Databases > Database Snapshots > Right-click a snapshot and choose Delete > Click OK.

To use the DROP DATABASE on a snapshot, you run:

```
DROP DATABASE database_snapshot_name
```

Revert to snapsot

Reverting to a snapshot is the same as a restore to the point in time when the snapshot was created. To revert to snapshot you use RESTORE DATABASE as shown below:

Syntax

```
RESTORE DATABASE database_name
FROM DATABASE_SNAPSHOT = 'database_snapshot_name'
```

Example

```
RESTORE DATABASE TEST
FROM DATABASE_SNAPSHOT = 'Test_snap'
```

Conclusion

I welcome snapshots with open arms to my DBA tool box and think they can be used in many situation. I do believe it's a pity that this is an Enterprise edition only feature, but I do see it as a tool I will make good use of when doing system upgrades on my Enterprise edition servers. Snapshots are in their infancy but I feel that with a little more work they will become a widely-used tool.
