

BEST PRACTICES GUIDE:

# **Nimble Storage Best Practices for Microsoft SQL Server**



## Summary

Microsoft SQL Server databases provide the data storage back end for mission-critical applications. Therefore, it's critical to protect these applications to ensure data integrity and availability for transaction processing. Nimble Storage arrays provide I/O optimization and protection features that greatly benefit SQL Server implementations.

It is important to first understand some of the internal I/O processes that SQL Server uses to store data on disk. Then you can use this fundamental understanding to implement best practices for provisioning a database storage design that will optimize performance and eliminate database backup windows. Using these best practices can also dramatically reduce recovery time objectives for full databases as well as individual database objects and data after user error and system failures.

## How SQL Server Writes Data to Disk

### Database Basics

Databases are primarily composed of tables of data stored in rows. Each column represents a specific field in the row. Each row represents one record in a table, based on the table design. For example, a basic design for a customer table would include the customer's first name, last name and some unique identifier such as a customer

ID to differentiate between customers with the same name (e.g., John Smith). Our columns would be CustomerID, FirstName, and LastName, while each row represents a single customer.

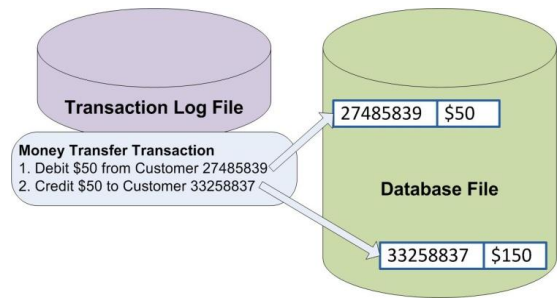
Customers Table		
CustomerID	First Name	Last Name
89437392	John	Smith
27485839	Diane	Jones
75839544	Ellen	Davis
33258837	John	Smith
61382475	Roger	Williams

Databases typically consist of dozens and even hundreds of tables that store specific types of information such as Customers, Products, Orders, etc. Over time our database will grow as we add more data to the tables. It is important to plan and monitor our database to provide the quickest response time to the end-users and to ensure that we properly protect the data assets from loss.

### Write-Ahead Logging (WAL)

To reduce the risk of data loss, Microsoft SQL Server writes data to disk using the Write-Ahead Logging (WAL) algorithm which is common to all major database platforms. The algorithm is designed as a fail-safe process that ensures that interrelated database changes are written completely as a single logical unit of work that can be undone if any of the individual changes fails for any reason. WAL adds a transaction log file to the write process to act as a witness to the database changes that will take place and add a layer of protection ensuring that two files have to agree before the data can be trusted.

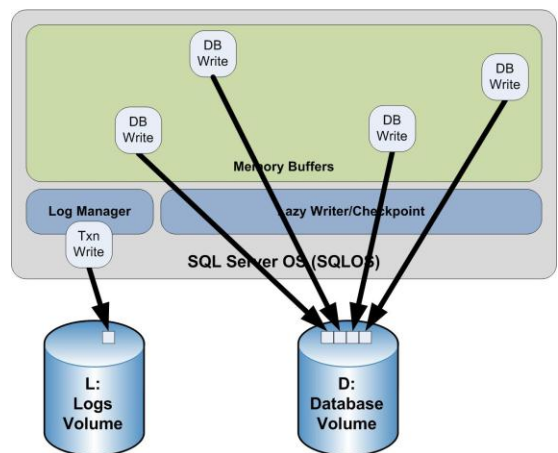
For example, let's assume that we have a banking database that contains the account balances of our customers. Two of the customers have \$100 in their accounts and one of the customers wants to transfer \$50 to the other customer. SQL Server will create an entry in the database transaction log file that contains two steps, a debit of \$50 from the first customer and a credit of \$50 to the other customer. When SQL Server tells the operating system to write the transaction, it also tells it to make sure that no hardware caches the write in memory and that it goes directly to a disk. After the transaction log write completes, SQL Server can update the database file to reflect the new first customer balance of \$50 and then update the database file to \$150 for the second customer.



If the system crashed in the middle of these SQL Server database writes, the database engine can use the transaction log to undo any partial writes and gracefully complete the transaction writes. Continuing with our money transfer example, when the database restarts it first checks to see if any transactions have been partially written to the database file. It will see that the first customer was debited \$50, but the second customer was not credited \$50. SQL Server will then use the logged transaction to credit the second customer's account with \$50 and complete the money transfer. If database engines did not use this algorithm, a system crash could leave one customer missing \$50 and both customers wondering where the money disappeared. Thus, the write-ahead logging algorithm provides a critical feature for ensuring application consistency for database applications.

### SQL Server File Writes

When SQL Server writes any changes to transaction log or database files, it uses fixed-length blocks of data in order to balance system performance and flexibility. The smallest unit of write that SQL Server uses is an 8 KB Page that can contain one or more rows from a table. For example, if we add a row to our customer's database, SQL Server will first write an 8 KB page to the transaction log file and then use the Lazy Writer or Checkpoint process to write the 8 KB changes to the database file. When a backup or Microsoft VSS snapshot is triggered, SQL Server will also trigger a write of any pending changes to flush data pages to disk. This places the database into a quiesced state that is safe to back up before the backup or snapshot occurs.



## Storage Layout Best Practices

### Use Nimble Protection Manager (NPM)

Nimble Protection Manager (NPM) provides an interface between a Nimble Storage array and the native VSS interfaces of the Windows operating system. This interface places the application data into a consistent state that is safe for backup and recovery, ensuring that the database can safely start and be assured that the data is in a form that SQL Server can read and trust when recovering data after an outage or user error. You can download Nimble Protection Manager from the Nimble Storage support web site.

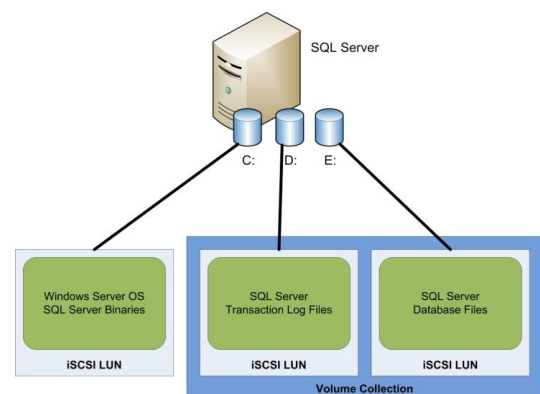
Computer servers continuously write system state and data changes to their storage devices. It's important that when a snapshot is taken, pending writes are flushed and the volume quiesced in a consistent state so that any later restore to that snapshot allows the application to continue working properly. The two primary consistent states are called *crash consistent* and *application consistent*.

- Crash consistency generally refers to the ability for a system to crash and allow the operating system to boot and find its files and core functionality, in a good readable state.
- Application consistency takes additional steps to ensure that an application's data is safe. It is often called transactional consistency when referring to database applications such as Microsoft™ SQL Server.

When you provision storage on a Nimble Storage array, you select one or more protection schedules that specify the intervals at which your data is protected to that point-in-time. When the scheduled time arrives, the Nimble array triggers the Nimble Protection Manager to coordinate a quiesce using VSS. After all writes are quiesced the array performs the snapshot.

### Store Database and Transaction Log Files on Separate Volumes

Nimble Storage arrays have redundant hardware and are highly available by design. They also include high performance auto-tuning features that actively analyze I/O usage patterns. When the Nimble array finds a hot spot in the files, it will optimize them and use the solid state disk for caching high read locations. Because they have different performance characteristics and will therefore tune differently, you should separate database and transaction logs on different volumes. For example, transaction logs have heavy sequential write activity while database files have more random read/write activity.



## Use SQL Server Performance Policy

The Nimble Storage array includes performance profiles that pre-configure new volumes using optimized configuration settings specific for different usage scenarios. For example, the SQL Server performance policy uses optimum block sizes to provide the best performance for SQL Server transaction log and database volumes. As you can see in the screen shot, the SQL Server performance policy also includes in-line compression and high performance caching.

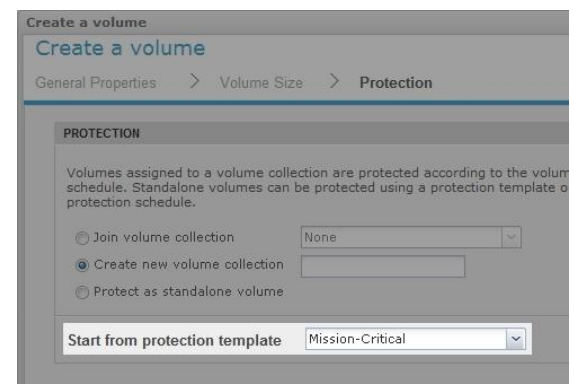
Performance Policies > SQL Server



PERFORMANCE PARAMETERS	
Block size	8192 bytes
Compress	Yes
Cache	Yes

## Use Protection Templates

Nimble Storage arrays provide Protection Templates that consist of pre-configured schedules for snapshots, replication, and retention policies. When creating a new volume collection you can select a Protection Template that will insert a default schedule based on existing business rules. For example, you could create Protection Templates based on the criticality of the application data. Less critical applications such as middleware servers can use longer snapshot schedule intervals (4 hours) and shorter retention schedules (10 days). However, more critical applications such as databases whose data frequently changes will usually require shorter snapshot schedule intervals (15 minutes or less) and longer retention schedules (90 days). In this case you will want to use a Protection Template with shorter snapshot schedules and longer retention schedules. Using Protection Templates reduces the amount of work required to create storage volumes and provide consistency for managing similar applications.



PROTECTION

Volumes assigned to a volume collection are protected according to the volume schedule. Standalone volumes can be protected using a protection template or protection schedule.

Join volume collection

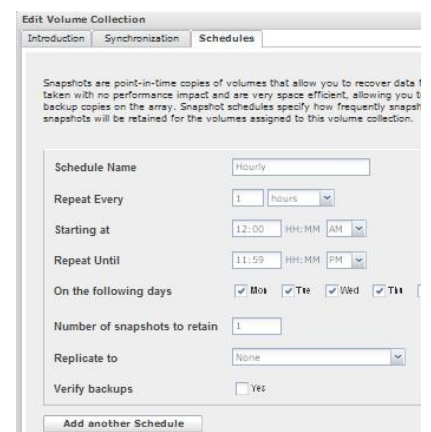
Create new volume collection

Protect as standalone volume

Start from protection template: Mission-Critical

## Use Volume Collections

A Volume Collection allows you to schedule the frequency and retention of snapshots as well as replication to other Nimble Storage arrays. A volume collection can coordinate protection activities between separate yet related volumes—such as a database’s transaction log and database file volumes—to ensure that databases are snapshotted with application consistency. The volume collection integrates with Microsoft VSS, which triggers them to flush SQL Server pending database writes to disk and pause the write activity of the transaction log and database files into a momentarily quiesced state. This ensures the data integrity of the point-in-time snapshot backup.



Introduction | Synchronization | Schedules

Snapshots are point-in-time copies of volumes that allow you to recover data if taken with no performance impact and are very space efficient, allowing you to backup copies on the array. Snapshot schedules specify how frequently snapshots will be retained for the volumes assigned to this volume collection.

Schedule Name: Hourly

Repeat Every: 1 hours

Starting at: 12:00 HH:MM AM

Repeat Until: 11:59 HH:MM PM

On the following days:  Mon  Tue  Wed  Thu

Number of snapshots to retain: 1

Replicate to: None

Verify backups:  Yes

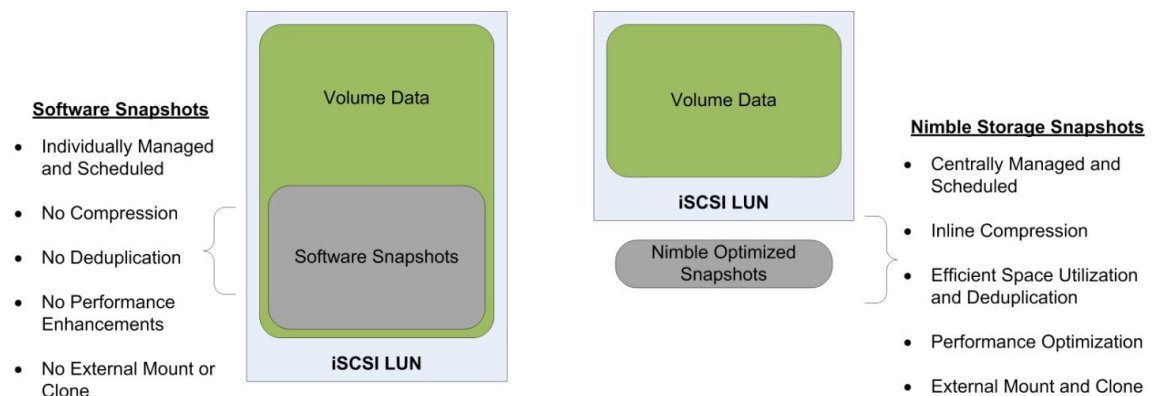
Add another Schedule

Management of a volume collection allows you to quickly change protection schedules for all related volumes. For example, suppose that you have created a SQL Server database protection schedule for several databases on a SQL Server supporting an eCommerce application. It is common for databases to be partitioned into different data files with differing performance characteristics. The initial business requirements called for a configuration based on hourly snapshots for backup and replication off-site every six hours for disaster recovery. As business has increased, management has decided that the database has become more critical and thus needs more frequent backup and more frequent replication to reduce potential data loss. You can change the protection schedule for all of the associated files for the database at the same time by changing the Volume Collection properties, thus saving time and eliminating configuration errors that might inhibit recoverability.

### Prefer Hardware Snapshots versus Software and SQL Server Snapshots

Snapshots are the basis for creating point-in-time versions of storage volumes and backups that can be mounted and accessed just like any other iSCSI volume. You can create snapshots at different layers of virtualization architectures, including within the Guest Software, within the Host Software and within the Storage Hardware. Connecting data volumes directly to the guest allows NPM to trigger snapshots that use the Nimble Storage hardware provider rather than inefficient software-based snapshots.

Nimble Storage arrays provide highly efficient hardware snapshot functionality that is optimized by Nimble’s inline compression and block incremental efficiencies. This differs from operating system native software snapshots such as Microsoft™ VSS, which are not efficiently stored within their volumes. Software snapshots don’t take advantage of Nimble Storage array optimized snapshot backup functionality. The following diagram shows the differing locations in which snapshots are stored. It is preferable to use hardware-based snapshots in the Nimble Storage array that take advantage of performance, in-line compression, and cloning capabilities rather than performing software snapshots with far less flexibility.



Beginning with SQL Server 2005 Enterprise, SQL Server also contains snapshot features that require SQL scripting skills. These types of SQL Server application-native snapshots are stored within the database files and use copy-on-write methodology that is very inefficient when compared with Nimble Storage array snapshot backups. Using Nimble Storage arrays, Database Administrators can perform database snapshots much more frequently and with more granular recovery points. These maintain the

same recovery methods available as SQL-native snapshots. Using Nimble array snapshots for restoration, reporting, and other use cases is covered in the Better SQL Server Recovery section.

## Better SQL Server Backups

SQL Server databases use the Write Ahead Logging algorithm to protect against system failure, and Nimble Storage arrays are hardware redundant to protect against storage failure. However, databases should be backed up from time to time to protect against logical failures such as accidental deletion of data. SQL Server provides several native features for backup and you can take advantage of Nimble Storage arrays for enhanced snapshot backup to avoid interruption of service.

The following timelines and graphs show the relative impacts of the different SQL Server backup and restoration methodologies after a database failure. These demonstrate the recovery point and recovery time associated with restoring a database back to full service. These timelines make certain assumptions that the hardware, operating system, and application software are unaffected by the outage and that the database failed because of a logical failure; an erroneous deletion of tables or files, for example. The system load metric is a composite of total production system involvement in the backup process, which includes the CPU, Memory, System Bus, Network, and Storage sub-system. Production database processing must compete with the system load imposed by the backup process, and reduces the transaction processing performance.

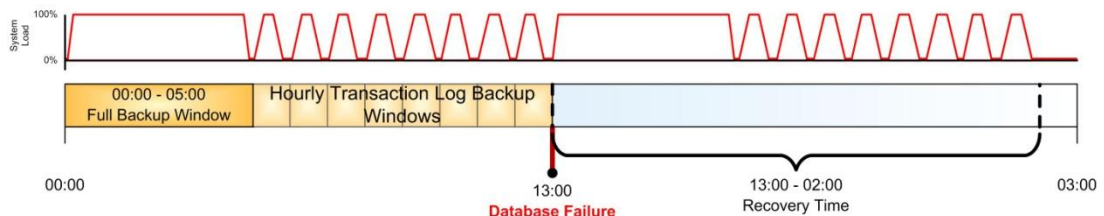
## SQL Server Backup to Disk or Tape

### Traditional Backup to Disk/Tape

Max RPO = 1 Hour

Max RTO = 24 Hours

System/Storage Impact = Heavy



The Traditional Backup to Disk/Tape timeline shows a typical daily backup schedule which performs a full daily database backup followed by hourly transaction log backups. We can see that the system load is high during the full backup when entire database and transaction log are copied to a backup disk or tape device. Later we see spikes hourly when the transaction log backup copies are made. Finally, the transaction logs are truncated on the successful completion of either the full or transaction log backup.

Recovery of a database begins with restoration of the most recent successful full backup, followed by restoration of each transaction log backup. The restoration time is considerable because all database objects (Tables, Indexes, Security, etc.) must be copied from the backup media to the server. Thus, if it takes 5 hours to back up, then it will take at least 5 hours to restore. Transaction logs are cumulative, so they can be restored as soon as the previous log completes restoration.



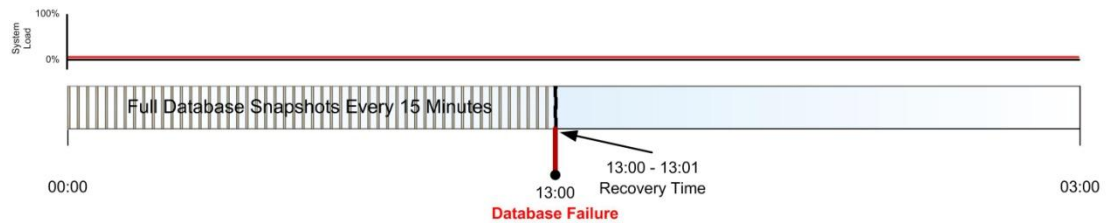
## SQL Server Backup Using Nimble Storage

### Nimble Storage Snapshot Backup

Max RPO = 15 Minutes

Max RTO = 1 Minute

System/Storage Impact = Minimal



Nimble Storage arrays are optimized to perform full database snapshot backups that minimize the impact on the production system. The Nimble array integrates with Microsoft VSS using Nimble Protection Manager (NPM) to periodically request the database to flush pending database writes to disk. Then the array creates an optimized snapshot. Recovery of a database using Nimble Storage snapshots is a very quick process in which an administrator creates a clone of the snapshot and mounts it directly for data restoration.

## SQL Server Database Recovery Models

SQL Server database recovery models were created to simplify the administration of databases. It's important to understand recovery models, because they affect your backup process. There are three recovery models available, Full, Bulk-Logged, and Simple. The primary differences between the recovery models are whether or not transactions are logged and if they truncate the transaction log when a checkpoint occurs. A checkpoint is marked in the transaction log after all database changes have been written to disk.

### Nimble Snapshot Backups for Simple Recovery Model Databases

For most SQL Server protection scenarios, the simple recovery model provides the best combination of management simplicity and data protection. The simple recovery model logs transactions

All Transactions Logged  
Truncate on Checkpoint

and keeps them until a checkpoint occurs. It then truncates the transaction log up to the checkpoint which keeps them from growing out of control. Using the simple recovery model does not permit restoration of transactions that have occurred in between backups, so some data loss is assumed using this recovery model. Nimble snapshot backups perform a nearly instant full backup of the database, which provides a maintenance-free method of managing a SQL Server database backup because logs are truncated regularly. SQL Server System databases (Master, Model, MSDB, etc.) are configured to use the simple recovery model by default. The diagram shows Nimble snapshot backups of the entire database at 15 minute intervals. Using Nimble snapshot backup at short and regular intervals provides a good balance of managability and very granular point-in-time restoration in the shortest time possible. If your database cannot permit loss of the most recent transactions that occur



between Nimble snapshot backups then you must use the SQL Server Full Recovery Model.

Name	Time	Status	Schedule	New Data	Compression
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::11:31:11.1 12/20 03:31 AM	12/20 03:31 AM	-	fifteen	50.88 KB	5.57
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::11:15:05.9 12/20 03:15 AM	12/20 03:15 AM	-	fifteen	35.31 KB	5.19
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::11:01:10.9 12/20 03:01 AM	12/20 03:01 AM	-	fifteen	46.25 KB	5.66
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::10:45:05.9 12/20 02:45 AM	12/20 02:45 AM	-	fifteen	55.38 KB	4.82
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::10:31:11.1 12/20 02:31 AM	12/20 02:31 AM	-	fifteen	44.63 KB	4.55
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::10:15:06.0 12/20 02:15 AM	12/20 02:15 AM	-	fifteen	66.94 KB	4.73
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::10:01:10.9 12/20 02:01 AM	12/20 02:01 AM	-	fifteen	40.69 KB	5.17
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::09:45:06.1 12/20 01:45 AM	12/20 01:45 AM	-	fifteen	47.63 KB	6.0
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::09:31:11.1 12/20 01:31 AM	12/20 01:31 AM	-	fifteen	34.44 KB	5.24
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::09:15:06.0 12/20 01:15 AM	12/20 01:15 AM	-	fifteen	40.0 KB	4.97
<input type="checkbox"/> ns-sql-dbandlog-fifteen-2010-12-20::09:01:11.1 12/20 01:01 AM	12/20 01:01 AM	-	fifteen	46.88 KB	5.86

### Protecting Full Recovery Model Databases

If you require recovery of transactions that have occurred after the most recent snapshot, then you should use the Full Recovery model and periodically perform full backup and

All Transactions Logged  
No Truncate on Checkpoint

transaction log backups using SQL Server native tools. SQL Server native backup tools permit recovery of its native full backup and subsequent transaction log backups, including the tail log. Refer to the Microsoft SQL Server documentation for the version of your database software, the SQL Server 2008 R2 documentation is located at the following link:

<http://msdn.microsoft.com/en-us/library/ms190217.aspx>.

In addition to SQL Server native backups, use Nimble snapshot backups to provide replication functionality for disaster recovery.

### Nimble Snapshot Backups for Bulk-Logged Recovery Model Databases

The bulk-logged recovery model is used primarily for inserting a large amount of data into a database as quickly as possible. This model is used for objects such as large databases (LDB) for Data Marts and Data Warehouses. Bulk-logged backups can also perform quickly by avoiding transaction log writes for database operations such as SELECT INTO, CREATE INDEX, as well as text and image operations. If

the database is damaged for any reason during the bulk-logged operation, then work may have to be redone, which is why this model isn't used as frequently outside of Large

Some Transactions Logged  
No Truncate on Checkpoint

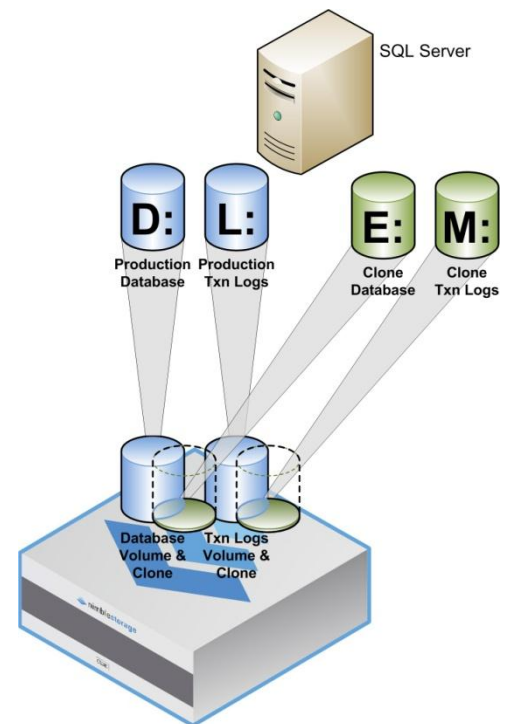
Database applications. To minimize the impact of an error during a bulk-logged operation, you should always perform a backup using the methods appropriate for the recovery model that your database uses if it is not always in the Bulk-Logged Recovery model.

## Better SQL Server Recovery

Restoration of data is the true test of a backup solution; especially how quickly data can be recovered and put back into production. As we've seen in the *Better Database Backups* section, Nimble Storage arrays can perform near-instantaneous full database backups at short, regular intervals. Nimble arrays also allow you to recover your data by creating a zero-copy clone and mounting it to a SQL Server host. Zero-copy clones provide instantaneous and extremely space efficient 'clones' of the production database. These clones are fully read-writable copies of the original database, but have the benefit of occupying almost zero storage capacity thanks to Nimble's efficient cloning technology. The Nimble Storage CS-Series Administrator's Guide details the steps to create a clone for restoration in the section "Using a Nimble array snapshot to recover a SQL server database." Using clones allows you to perform restorations and other use cases that will maximize the benefit of Nimble Storage within your organization.

If you plan to perform a full database restoration, then you can detach the existing database that you wish to restore. Copy the database and transaction log files from the cloned restoration volume to the production volume and overwrite the existing files. Attach the recovered database and continue working.

For more advanced database object-level restoration capabilities, you can leave the production database untouched and attach the recovery database temporarily from the cloned restoration volume. Open the SQL Server Management Studio and click the New Query button on the toolbar. Then use the following SQL Server commands to attach the recovery database with a different database name (RecoveryDB) than the production database.



```
CREATE DATABASE RecoveryDB
ON
(NAME = RecoveryDB, FILENAME =
'E:\SQL Server\Databases\ProductionDB.mdf'),
(NAME = RecoveryDB_Log, FILENAME =
'M:\SQL Server\Transaction Logs\ProductionDB_log.ldf')
FOR ATTACH
```

### Database Object Restoration

Use this scenario to recover from erroneous database object operations such as truncating a table or deleting a stored procedure. Use a SQL query to copy the database object from the cloned recovery database to the production database. For example:

```
insert into ProductionDB.dbo.Catalog
select * from RecoveryDB.dbo.Catalog
```

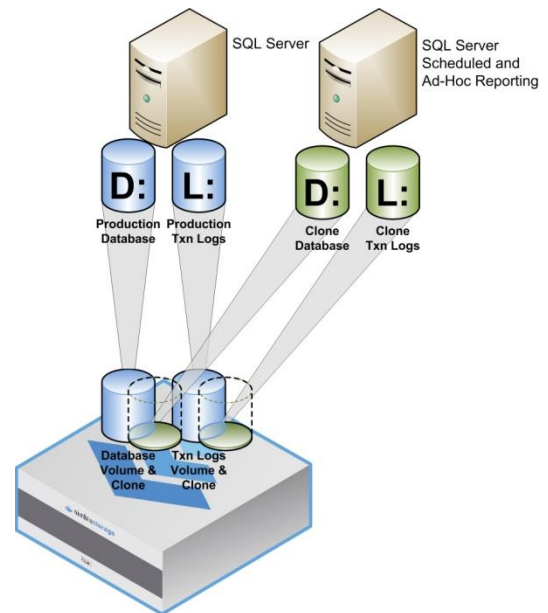
## Data Restoration

This works similarly to Database Object Restoration but uses a SQL query that restores certain data points that were lost due to user error, for example, the recovery of an accidental Order deletion in an ERP or CRM database from a recent snapshot backup.

```
update ProductionDB.dbo.Customer
set LastName = (
    select LastName
    from RecoveryDB.dbo.LastName
    where CustomerID = 8309485)
where CustomerID = 8309485
```

## Report Server Off-loading

Many organizations find that as their databases grow over time, their reporting needs begin to adversely compete for processing and storage resources with the production use of the system. Off-loading scheduled and ad-hoc reporting to another system reduces the competition for system resources and extends to useful life of the production system. It also gives end-users quicker response times that increase their productivity. You can create a volume clone of your production database and attach it to the reporting server. This use case also provides off-site reporting functionality using Nimble's WAN-optimized replication technology. This replication technology dramatically reduces the TCO (total cost of ownership) for off-site replication.



## Development and Q/A Testing

Another popular use case is to use a production clone for Quality Assurance and Development Testing. This model is similar to the Report Server Off-loading such that the Nimble volume clone is attached to another server, but it is usually refreshed less frequently than the Report Server. The database is an identical copy of the production database and allows more complete testing without the risk of adversely affecting the production system.



**Nimble Storage, Inc.**

2740 Zanker Road, San Jose, CA 95134

Tel: 408-432-9600; 877-364-6253 | [www.nimblestorage.com](http://www.nimblestorage.com) | [info@nimblestorage.com](mailto:info@nimblestorage.com)

© 2012 Nimble Storage, Inc. All rights reserved. CASL is a trademark of Nimble Storage Inc. BPG-SQL-0812