# SQL Server version and Edition

# 

# SQL Server Installation



# MS SQL Server - Architecture



# SQL Server – Reading ERRORLOG with xp\_ReadErrorLog

To read error logs in SQL Server using T-SQL you can use extended stored

procedure**xp\_ReadErrorLog**to read SQL Server and SQL Server Agent error logs. **xp\_ReadErrorLog**has seven parameters that can be used to filter error logs.

Syntax for **xp\_ReadErrorLog**:

EXEC xp\_ReadErrorLog    *<LogNumber>,* *<LogType>*,

*<SearchTerm1>*, *<SearchTerm2>*,

*<StartDate>*, *<EndDate>*, *<SortOrder>*

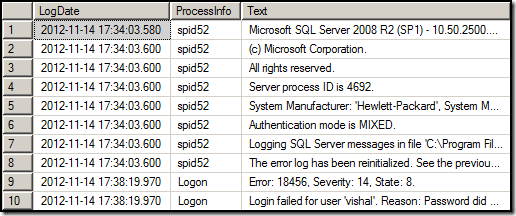
The parameter values can be as follows:

|  |  |
| --- | --- |
| **Parameter** | **Values** |
| *<LogNumber>* | Log number 0, 1, 2 … For example 0 returns current log. 2 returns logs from ERRORLOG.2 |
| *<LogType>* | 1 – Reads SQL Server error logs, 2 – Reads SQL Server Agent error logs |
| *<SearchTerm1>* | Search Term for Text Column |
| *<SearchTerm2>* | Search Term for Text Column **\***When both search terms are specified, it only returns lines containing both terms |
| *<StartDate>* | Start reading logs from specified date |
| *<EndDate>* | Reads logs till this date |
| *<SortOrder>* | ASC – Ascending or DESC – Descending |

You can use the stored procedure as:

EXEC xp\_ReadErrorLog

– Reads current SQL Server error log

[](https://sqlandme.files.wordpress.com/2012/11/image5.png)

Below are some more examples of **xp\_ReadErrorLog**:

EXEC xp\_ReadErrorLog 1

– Reads SQL Server error log from ERRORLOG.1 file

EXEC xp\_ReadErrorLog 0, 1

– Reads current SQL Server error log

EXEC xp\_ReadErrorLog 0, 2

– Reads current SQL Server Agent error log

EXEC xp\_ReadErrorLog 0, 1, 'Failed'

– Reads current SQL Server error log with text 'Failed'

EXEC xp\_ReadErrorLog 0, 1, 'Failed', 'Login'

– Reads current SQL Server error log with text ‘Failed’ AND 'Login'

EXEC xp\_ReadErrorLog 0, 1, 'Failed', 'Login', '20121101', NULL

– Reads current SQL Server error log with text ‘Failed’ AND ‘Login’ from 01-Nov-2012

EXEC xp\_ReadErrorLog 0, 1, 'Failed', 'Login', '20121101', '20121130'

– Reads current SQL Server error log with text ‘Failed’ AND ‘Login’ between 01-Nov-2012 and 30-Nov-2012

EXEC xp\_ReadErrorLog 0, 1, NULL, NULL, '20121101', '20121130'

– Reads current SQL Server error between 01-Nov-2012 and 30-Nov-2012

EXEC xp\_ReadErrorLog 0, 1, NULL, NULL, '20121101', '20121130', 'DESC'

– Reads current SQL Server error log between 01-Nov-2012 and 30-Nov-2012 and sorts in descending order

# SQL Server Services

SQL Server Services :

MS SQL Server provides the following two services which is mandatory for databases creation and maintenance. Other add-on services available for different purposes are also listed.

* SQL Server
* SQL Server Agent

## Other Services

* SQL Server Browser
* SQL Server Full Text Search
* SQL Server Integration Services
* SQL Server Reporting Services
* SQL Server Analysis Services

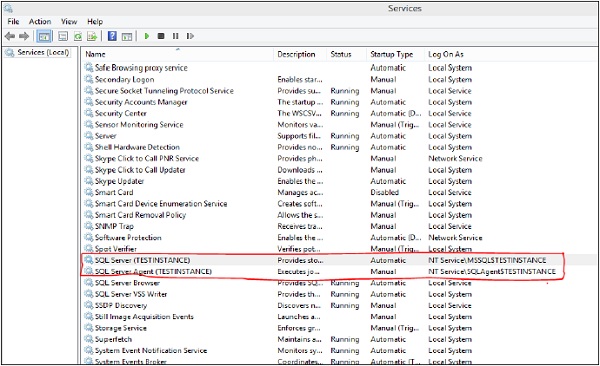
The above services can be availed using the following method.

## Start Services

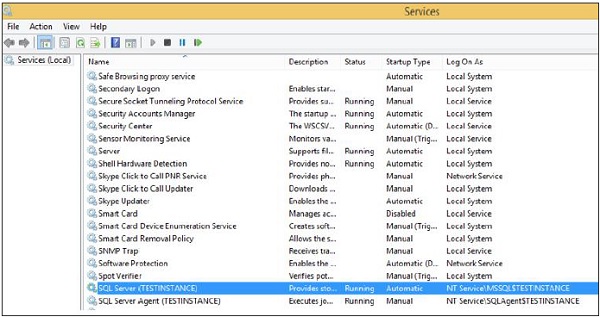
To start any of the services, either of the following two methods can be used.

### Method 1 – Services.msc

**Step 1** − Go to Run, type services.msc and click OK. The following screen appears.



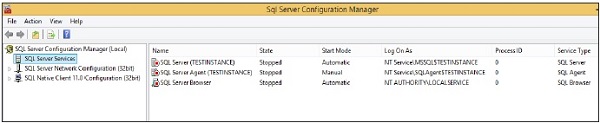
**Step 2** − To start service, right-click on service, click Start button. Services will start as shown in the following snapshot.



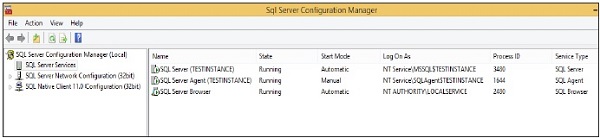
### Method 2 – SQL Server Configuration Manager

**Step 1** − Open configuration manager using the following process.

Start → All Programs → MS SQL Server 2012 → Configuration Tools → SQL Server configuration manager.



**Step 2** − Select the service name, right-click and click on start option. Services will start as shown in the following snapshot.

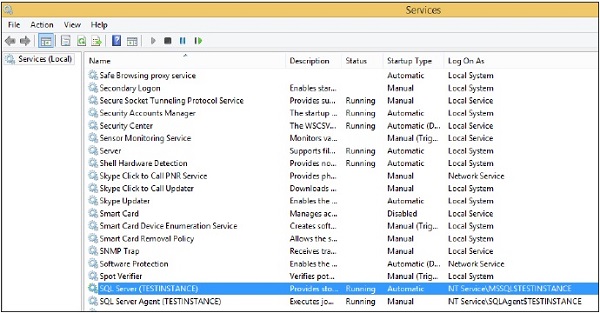


## Stop Services

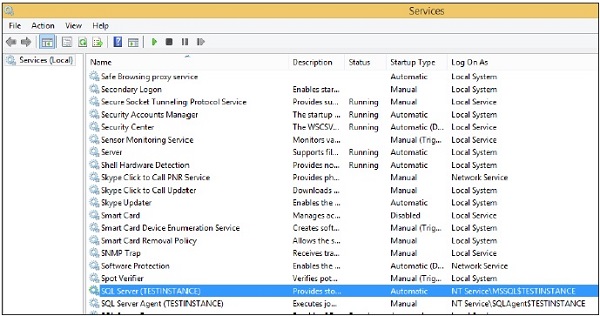
To stop any of the services, either of the following three methods can be used.

### Method 1 - Services.msc

**Step 1** − Go to Run, type services.msc and click OK. The following screen appears.



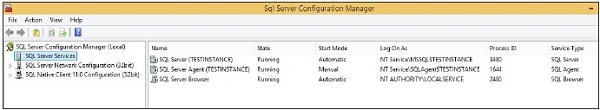
**Step 2** − To stop services, right-click on service and click Stop. The selected service will be stopped as shown in the following snapshot.



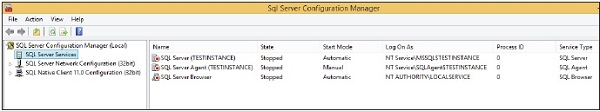
### Method 2 – SQL Server Configuration Manager

**Step 1** − Open configuration manager using the following process.

Start → All Programs → MS SQL Server 2012 → Configuration Tools → SQL Server configuration manager.

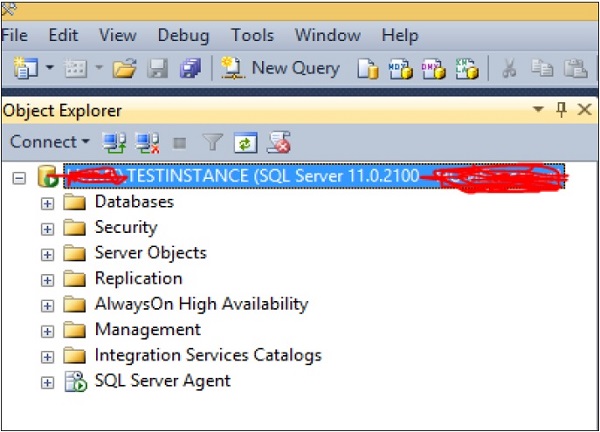


**Step 2** − Select the service name, right-click and click Stop option. The selected service will be stopped as shown in the following snapshot.

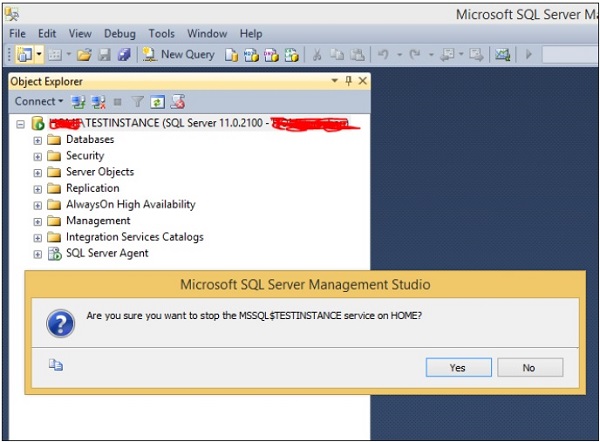


### Method 3 – SSMS (SQL Server Management Studio)

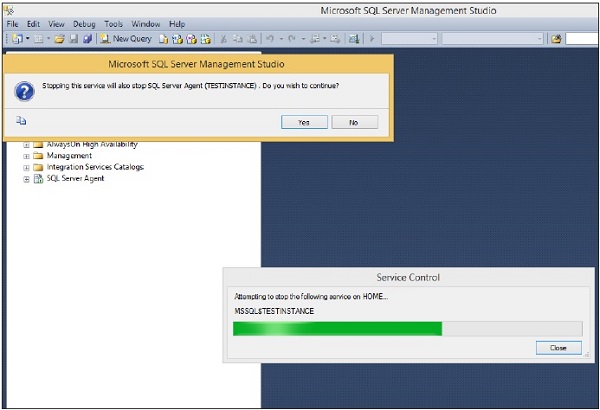
**Step 1** − Connect to the instance as shown in the following snapshot.



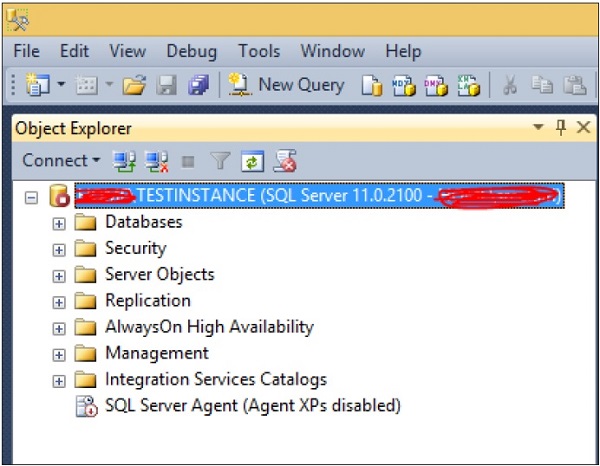
**Step 2** − Right-click on instance name and click Stop option. The following screen appears.



**Step 3** − Click Yes button and the following screen will open.



**Step 4** − Click Yes option on the above screen to stop SQL Server agent service. The services will be stopped as shown in the following screenshot.



**What if SQL Server Browser service is stopped?**

The SQL Server Browser service is installed with installation of SQL Server 2005/2008. It also serves earlier versions installed on the same machine if there any. The SQL Server Browser service has different impact in different situations, such as:

* If you have just one instance installed on machine and it is running on default port 1433, then status of SQL Server Browser service does not make any difference in your connection parameters.
* If there are more than one instances running on the same machine, in that case either you have to start SQL Server Browser service or provide the port number along with IP (or server name) and instance name, to access any other instance than default.
* If SQL Server Browser service is stopped and IP along with port number is not provided then connection will be refused.
* If SQL Server instance is configured using dynamic ports then browser service is required to connect to correct port number.
* Also your named instances will not be published in the list of SQL Server instances on the network (which could be a good thing)

# System Databases

master

The master database contains all of the system level information for SQL Server – all of the logins, linked servers, endpoints, and other system-wide configuration settings. The master database is also where SQL Server stores information about the other databases on this instance and the location of their files. If the master database is not present, SQL Server cannot start.

Always take regular backups of the master database.

Since the master database hold all of the information related to logins, endpoints, linked servers, and user databases, it’s important that you take a backup of the master database after configuring any of these server level changes. Otherwise, if your SQL Server suffers a catastrophic failure, those changes will be lost to the sands of time.

### model

The model database is used as a template whenever a new user database is created. You can change most database properties, create users, stored procedures, tables, views, etc – whatever you do will be applied to any newdatabases.

The nice thing is that you can create a guaranteed set of users, stored procedures, and options (including collation options) by making changes to the model database. Once those changes are in place, they are applied to every new database.

### msdb

msdb is used by the SQL Server Agent, database mail, Service Broker, and other services. If you aren’t actively working with things like jobs, alerts, log shipping, etc you can pretty safely ignore msdb… sort of.

One important item is that msdb holds backup history. Using the msdb tables (you can start by taking a look at msdb.dbo.backupset), it’s possible to determine when each database and filegroup was last backed up. This is very useful, especially when you’ve just started working at a new company or taken over the maintenance of new servers.

### Resource database

The resource database is a hidden system database. This is where system objects are stored. It isn’t possible to see the resource database by normal means. However you can see the data file by navigating to C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Binn. The exact size and modification data of this file will be different from version to version, but the modified date should be the same date that you see when you run SELECT @@version.

It is best to think of the resource database as if it were another system DLL. The resource database is designed to make it easy for quick database upgrades. If new system objects are being put in place, it is only necessary to swap out the resource database MDF file.

### tempdb

We come, at last, to tempdb. Tempdb is the workhorse of the system databases. It is the workspace that SQL Server uses to store the intermediate results of query processing and sorting. You know how you see those spools in your execution plans? When you see one of those, SQL Server is probably spooling the data to a temporary table in the tempdb. Outside of storing temporary results, tempdb is also used during snapshot isolation and for user created temporary tables ([this includes table variables](http://scarydba.wordpress.com/2009/10/13/table-variables-are-only-in-memory-fact-or-myth/)).

One thing that is interesting to note about tempdb is that it is re-created every time the SQL Server service is started. Any objects that you have created in tempdb will be gone once the server restarts. If you want specific tables or stored procedures to always be available in tempdb, you will need to add them to the model database or else use a stored procedure to create them in tempdb when the SQL Server is started.

A properly sized and configured tempdb is vital for effective SQL Server performance. By default tempdb starts at 8MB in size and will continue to grow by ten percent until the drive is full. If the tempdb is sized too small, system resources will be used growing the tempdb file. This overhead can be avoided by increasing the default size of tempdb.

# system databases in detail

****

# Files and File Groups in SQL Server

a very important topic in SQL Server: how SQL Server handles Files and File Groups. When you create a simple database with a **CREATE DATABASE** command, SQL Server only creates 2 files for you:

* A Data File (.mdf)
* A Transaction Log File (.ldf)

The data file itself is created within the one and only PRIMARY file group. Within the PRIMARY file group by default SQL Server stores all the data (user tables, system tables, etc.). So what is the purpose of having additional files and File Groups? Let’s have a look at it.

**Multiple File Groups**

When you create additional File Groups for your database, you are able to store your user defined tables and indexes in them. This helps you in multiple ways:

* You keep the PRIMARY file group small.
* You can split your data across multiple File Groups (e.g. when you use **Table Partitioning** in the Enterprise Edition).
* You can perform Backup and Restore operations at the File Group level. This gives you a more granular control over your Backup and Restore strategy.
* You can run DBCC CHECKDB commands at the File Group level instead of at the database level.

In general, you should always have at least one secondary File Group where you store your own database objects. You should never store anything in the PRIMARY file group besides the system objects that SQL Server creates for you.

**Multiple Files**

When you have created your own File Group you also have to place at least one file into it. In addition, you can add additional files into the File Group. This can also improve the performance of your workload, because SQL Server evenly spreads the data between all files through a so-called Round Robin Allocation Algorithm. The first extent of 64K is stored in the first file, the second extent of 64K is stored in the second file, the third extent of 64K is again stored in the first file (when you have 2 files in your File Group).

With that approach SQL Server can latch multiple copies of the allocation bitmap pages (PFS, GAM, SGAM) in the Buffer Pool and increase the throughput of your workload. You solve the same problem that can occur in a default configuration of [TempDb](http://www.sqlpassion.at/archive/2013/07/23/sql-server-quickie-7-tempdb-latch-contention/). In addition, SQL Server also makes sure that all the files within the File Group are full at the same point in time – through a so-called Proportional Fill Algorithm. Therefore, it is also very important that all your files within the File Group have the same initial size and the same Auto Growth factors. Otherwise the Round Robin Allocation can’t work very well.

**A concrete example**

Now let’s have a look at a concrete example of how you can create a database with an additional File Group with multiple files within it. The following code shows you the CREATE DATABASE command that is necessary to accomplish this task.

-- Create a new database

CREATE DATABASE MultipleFileGroups ON PRIMARY

(

-- Primary File Group

NAME = 'MultipleFileGroups',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MultipleFileGroups.mdf',

SIZE = 5MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1024KB

),

-- Secondary File Group

FILEGROUP FileGroup1

(

-- 1st file in the first secondary File Group

NAME = 'MultipleFileGroups1',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MultipleFileGroups1.ndf',

SIZE = 1MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1024KB

),

(

-- 2nd file in the first secondary File Group

NAME = 'MultipleFileGroups2',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MultipleFileGroups2.ndf',

SIZE = 1MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1024KB

)

LOG ON

(

-- Log File

NAME = 'MultipleFileGroups\_Log',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MultipleFileGroups.ldf',

SIZE = 5MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1024KB

)

GO

After the creation of the database, the question is then how to put a table or index into a specific file group? You can specify the File Group manually with the **ON** keyword as the following code shows.

-------------------------------------------------------------------------------------------------------------------------

CREATE TABLE Customers

(

FirstName CHAR(50) NOT NULL,

LastName CHAR(50) NOT NULL,

Address CHAR(100) NOT NULL,

ZipCode CHAR(5) NOT NULL,

Rating INT NOT NULL,

ModifiedDate DATETIME NOT NULL,

)

ON [FileGroup1]

GO

# Understanding how SQL Server stores data in data files



# Database Page Type and read page information



# SQL Server transaction logs

A transaction log is a file – integral part of every SQL Server database. It contains log records produced during the logging process in a SQL Server database. The transaction log is the most important component of a SQL Server database when it comes to the disaster recovery – however, it must be uncorrupted. After each database modification – transaction occurrence, a log record is written to the transaction log. All the changes are written sequentially

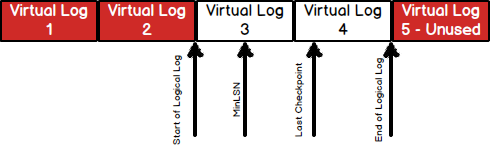
## What does a SQL Server transaction log store?

A transaction log stores every transaction made to a SQL Server database, except some which are minimally logged like BULK IMPORT or SELECT INTO. Internally it is split into the smaller parts called Virtual Log Files (VLFs). When one VLF becomes full logging continue to write into the next available in the transaction log. The transaction log file can be represented as a circular file. When the logging reaches the end of the file it starts again from the beginning, but only if all the requirements has been met and the inactive parts has been truncated. The truncation process is necessary to mark all inactive parts so they can be used again and overwritten

*A log record is no longer needed in the transaction log if all of the following are true:*

* *The transaction of which it is part has committed*
* *The database pages it changed have all been written to disk by a checkpoint*
* *The log record is not needed for a backup (full, differential, or log)*
* *The log record is not needed for any feature that reads the log (such as database mirroring or replication)* [1]

Logical log is an active part of the transaction log. A Log Sequence Number (LSN) identifies every transaction in the transaction log. The MinLSN is the starting point of the oldest active transaction in the online transaction log



## Can SQL Server database work without a transaction log?

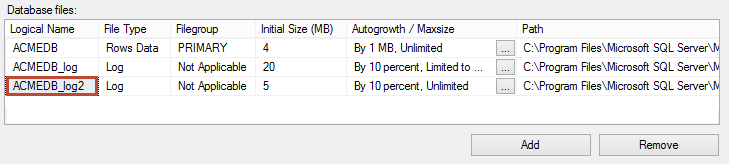
No, that is not possible due to the SQL Server design and ACID compliance. ACID stands for atomicity, consistency, isolation, and durability. All the transactions must fullfil these characteristics:

* *An****atomic transaction****is either fully completed, or is not begun at all*
* *A transaction enforces****consistency****in the system state by ensuring that at the end of any transaction the system is in a valid state*
* *When a transaction runs in****isolation****, it appears to be the only action that the system is carrying out at one time*
* *A transaction is****durable****meaning that once it has been successfully completed, all of the changes it made to the system are permanent* [2]

## Can one SQL Server database have more than one transaction log?

Yes, that is possible, but it’s only recommended in the specific situations. Adding multiple transaction log files will not improve performance of the SQL Server database in any way. Writing can occur only into one file at the time, so parallel I/O operations are not possible

Having multiple transaction log files is recommended only if the first one is full or the disk drive is running out of space. Either way, these problems should be addressed earlier and handled by creating transaction log backups and monitoring available space on the disk drive

[](https://www.sqlshack.com/wp-content/uploads/2014/02/databasefiles.png)

## How to Read transaction log file

1.

select [current lsn],

[operation],

[transaction name],

[transaction id],

[transaction sid],

[spid],

[begin time]

From fn\_dblog(null,null)

2. DBCC LOGINFO

## Why is the SQL Server transaction log growing?

Every transaction is followed by logging into the online transaction log. During SQL Server work, the transaction log grows if changes are made into a database, therefore maintaining the transaction log is crucial for proper SQL Server operation

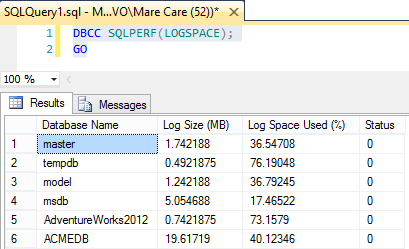
**There are three recovery models available in SQL Server, depending on which one is used transaction log growth is manifested differently:**

* **Simple recovery model** – the transaction log backups are not supported. The truncation process is automated and space is reclaimed for reuse. There is data loss risk because changes since the most recent database backup are exposed. In the Simple recovery there is little chance for the transaction log growing – just in specific situations when there is a long running transaction or transaction that creates many changes
* **Bulk-logged recovery model** – the transaction log backups are supported and needed on regular basis. There is no automated process of transaction log truncation, the transaction log backups must be made regularly to mark unused space available for overwriting. Bulk-logged recovery model reduces transaction log space usage by using minimal logging for most bulk operations
* **Full recovery model** – the transaction log backups are supported and needed on regular basis. There is no data loss risk under the normal circumstances. There is no automated process of transaction log truncation, the transaction log backups must be made regularly to mark unused space available for overwriting. In the Full recovery there is biggest chance of the transaction log growing because all transactions are logged

## How to maintain a transaction log in SQL Server?

A transaction log maintenance is important task in SQL Server administration. Monitoring is recommended on daily basis or even more often is a SQL Server database has high amount of traffic. The transaction log space can be monitored by using the DBCC SQLPREF command:

|  |
| --- |
| DBCC SQLPERF(LOGSPACE);  GO |



* *Database Name – Name of the database for the log statistics displayed*
* *Log Size (MB) – Current size allocated to the log. This value is always smaller than the amount originally allocated for log space because the Database Engine reserves a small amount of disk space for internal header information*
* *Log Space Used (%) – Percentage of the log file currently occupied with transaction log information*
* *Status – Status of the log file. Always 0* [3]

The transaction log should be backed up on the regular basis to avoid the auto growth operation and filling up a transaction log file. Space in a transaction log can be truncated (cleared) through SQL Server Management Studio by selecting Transaction Log as backup type or through CLI by executing the following command:

|  |
| --- |
| BACKUP LOG ACMEDB  TO DISK = 'C:\ACMEDB.TRN'  GO |

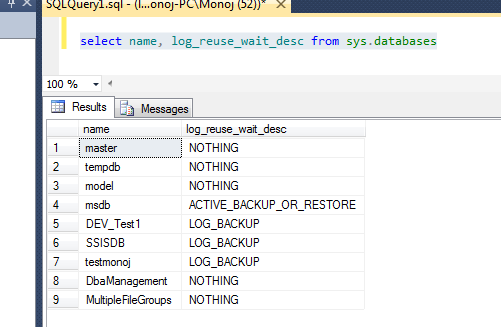
That backed up space can be reused again and it’ll be overwritten with new transactions. Some operations can’t be combined and must be executed separately:

## How To know why VLF cannot reuse

Run the below command to know the reason.

select name, log\_reuse\_wait\_desc from sys.databases

OutPut :



**Nothing :** don't required to do for VLF reuse

**Log-Backup :** You have to take transaction log backup. Generally when database is in full recovery model this may happen.

**Active transaction :** Active transactions are going on.

so on .......

## Do I need SQL Server transaction log backups?

Yes, that is one of the most important resource when it comes to disaster recovery. They are not needed (and available) only if the Simple recovery model is used – but there is data loss exposure. Majority of Database Administrators uses 15 minutes interval or even less for a high traffic SQL Server databases. The transaction log backups are important because when taken they mark inactive VLFs that can be used for writing down new transactions

## How to Clear SQL Server Transaction Log File with DBCC Shrinkfile

**SQL Server Transaction Log**

In SQL, transaction log file is used to save records that are produced during logging process in an SQL Server database. The transaction log is basically an integral part of [SQL Server database](https://www.janbasktraining.com/blog/create-database-microsoft-sql-server/). It is not only integral but also a most important component of the SQL Server database especially when it is about disaster recovery. As in SQL, as soon as the database is modified the log record is written to the transaction log. Here all the changes are recorded in the transaction file in a sequential order. In this article, we are going to discuss SQL Server transaction log and its features.

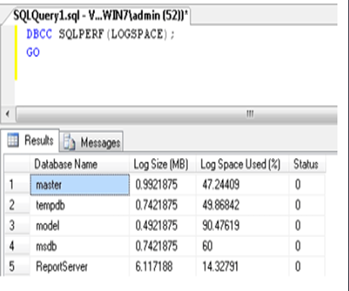
**Why should we clear the Transaction log?**

If any database change occurs than SQL Server transaction log grows. Regular maintenance of transaction log can prevent it from becoming full. Log truncation technique is used to keep the log empty or prevent it from being filled up. This process deletes the inactive log files from the logical transaction log and freed up space from logical transaction log that can be reused by the physical transaction log. If the log will not be truncated then it can fill all disk space allocated to the physical transaction log.

Truncation process for the three [recovery models of SQL](https://www.janbasktraining.com/blog/sql-server-recovery-models/) is different and explained as below:

* **Simple Recovery Model**: In this model Transaction log backup is not supported and truncation process executes automatically as a result space is freed up automatically
* **Bulk-logged Recovery Model**: In this automatic truncation is not available and so regular backups are needed to mark unused space and hereby space is created for overwriting purpose. The size of the log file can be here reduced for bulk operations
* **Full Recovery Model**: In this model again, the truncation process is not automatic and the chances for growing the log files are higher.

One can know the available and used transaction log space with the help of a single line command as shown in the following figure:



Now after getting the information, you can take action accordingly to clear the log file.

**How to clear the Transaction Log File of SQL Server?**

As it has been clear so far that log truncation process creates more free space in the transaction log file so is also called **log clearing**. A transaction log file is usually o small chunks that are also called Virtual Log Files or VLF. These VLFs are marked if are available for reuse and if not, then also they will be marked accordingly by using active and inactive tags respectively.

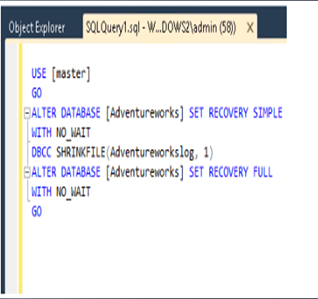
To clear the transaction log, one has to search these VLF files and make them free or available. To reduce the size of physical log files, Log Shrinking process is used as Log Truncation does not reduce the physical log storage size.

**Two steps are involved in the process of clearing SQL Server transaction log:**

* TRUNCATE\_ONLY option is used to perform a log backup
* DBCC SHRINKFILE function shrinks the file to the required size

Here BACKUP LOG TRUNCATE\_ONLY option can also be used for this process but it empties all transaction log content without backing it up. As TRUNCATE\_ONLY option is not available in advance version of SQL Server, so many users use this option along with DBCC SHRINKFILE function.

Simple recovery model can be utilized to generate the logs that will be used rather than opting truncation transaction logs. Like shown in the following window:



Here, two options can be used to shrink the log file:

* Shrinking of Log File by using SQL Server Management Studio or SSMS
* Shrink the transaction log file by using T-SQL

## Syntax

SQL

DBCC SHRINKFILE

(

{ file\_name | file\_id }

{ [ , EMPTYFILE ]

| [ [ , target\_size ] [ , { NOTRUNCATE | TRUNCATEONLY } ] ]

}

)

[ WITH NO\_INFOMSGS ]

## Arguments

file\_name  
The file to be shrunk's logical name.

file\_id  
The file to be shrunk's identification (ID) number. To get a file ID, use the [FILE\_IDEX](https://docs.microsoft.com/en-us/sql/t-sql/functions/file-idex-transact-sql?view=sql-server-2017) system function or query the [sys.database\_files](https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-database-files-transact-sql?view=sql-server-2017) catalog view in the current database.

target\_size  
An integer - the file's new megabyte size. If not specified, DBCC SHRINKFILE reduces to the file creation size.

**Note**

You can reduce an empty file's default size using DBCC SHRINKFILE target\_size. For example, if you create a 5-MB file and then shrink the file to 3 MB while the file is still empty, the default file size is set to 3 MB. This applies only to empty files that have never contained data.

This option isn't supported for FILESTREAM filegroup containers.  
If specified, DBCC SHRINKFILE tries to shrink the file to target\_size. Used pages in the file's area to be freed are moved to free space in the file's kept areas. For example, with a 10-MB data file, a DBCC SHRINKFILE operation with an 8 target\_size moves all used pages in the file's last 2 MB into any unallocated pages in the file's first 8 MB. DBCC SHRINKFILE doesn't shrink a file past the needed stored data size. For example, if 7 MB of a 10-MB data file is used, a DBCC SHRINKFILE statement with a target\_size of 6 shrinks the file to only 7 MB, not 6 MB.

EMPTYFILE  
Migrates all data from the specified file to other files in the **same filegroup**. In other words, EMPTYFILE migrates data from a specified file to other files in the same filegroup. EMPTYFILE assures you that no new data gets added to the file, despite this file not being read-only. You can use the [ALTER DATABASE](https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql?view=sql-server-2017) statement to remove a file. If you use the [ALTER DATABASE](https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql?view=sql-server-2017) statement to change file size, the read-only flag is reset and data can be added.

For FILESTREAM filegroup containers, you can't use ALTER DATABASE to remove a file until the FILESTREAM Garbage Collector has run and deleted all the unnecessary filegroup container files that EMPTYFILE has copied to another container. For more information, see[sp\_filestream\_force\_garbage\_collection (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/filestream-and-filetable-sp-filestream-force-garbage-collection?view=sql-server-2017)

**Note**

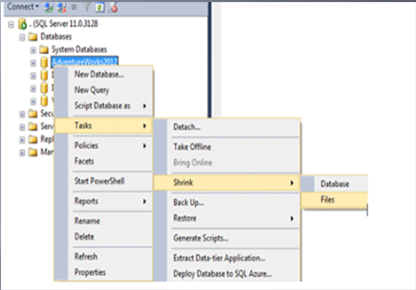
For information on removing a FILESTREAM container, see the corresponding section in [**ALTER DATABASE File and Filegroup Options (Transact-SQL)**](https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql-file-and-filegroup-options?view=sql-server-2017)

NOTRUNCATE  
Moves allocated pages from a data file's end to unallocated pages in a file's front with or without specifying target\_percent. The free space at the file's end isn't returned to the operating system, and the file's physical size does not change. Therefore, if NOTRUNCATE is specified, the file appears not to shrink. NOTRUNCATE is applicable only to data files. The log files are not affected. This option isn't supported for FILESTREAM filegroup containers.

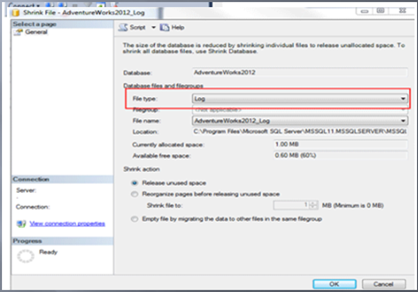
TRUNCATEONLY  
Releases all free space at the file's end to the operating system but does not perform any page movement inside the file. The data file is shrunk only to the last allocated extent. target\_size is ignored if specified with TRUNCATEONLY.  
The TRUNCATEONLY option does not move information in the log, but does remove inactive VLFs from the end of the log file. This option isn't supported for FILESTREAM filegroup containers.

WITH NO\_INFOMSGS  
Suppresses all informational messages.

In the above-mentioned option, SSMS is a server management tool that is easier to use can instantly reduce or shrink the size of log files. Here the database for which the shrink operation has to be performed can be chosen and by the right-clicking on it Shrink operation can be performed as shown in the following figure:



Here in this tool, you may have to choose the file type as log:



If you want to shrink the file using T-SQL then the following command will have to be used:

**DBCC SHRINKFILE (Log Filename, Desired Size in MB)**

In case of Full Recovery Model, when we are not bothered about losing data then the following commands can be used for shrinking the file:

*ALTER DATABASE nameDB*

*SET RECOVERY SIMPLE*

*GO*

*DBCC SHRINKFILE (Log Filename, Desired Size in MB)*

*Go*

*ALTER DATABASE nameDB*

*SET RECOVERY FULL*

Another option is also available to shrink the SQL file and that is by using a command:

**BACKUP LOG nameDB TO Backup Device**

In this way, anyone can clear the transaction log file of SQL Server to create free space in the log and reuse that space. As transactions are very much important and the user must keep track of them to make changes in the transaction and to keep the space available for further storing.

**Conclusion**

Here we can say that as transactions are an imperative and essential part of SQL Server so one must keep track of them. But as many transactions take place so to improve the efficiency of transaction processing it is must keep clean the transaction log files. Clear files will not only provide free space for information storage but also will improve the [performance of SQL Server](https://www.janbasktraining.com/blog/popular-sql-server-performance-tuning-tips/). As per the recovery model used by SQL Server, users can perform these operations. The user can clean the log in three ways choose the best one and keep the log clean and empty.

**SQL Server Checkpoint in details**



**DataBase State Options**

The state options control who can use the database and for what operations. There are three aspects to usability: The user access state determines which users can use the database; the status state determines whether the database is available to anybody for use; and the updateability state determines what operations can be performed on the database. You control each of these aspects by using the ALTER DATABASE command to enable an option

for the database.None of the state options uses the keywords ON and OFF to control the state value.

**SINGLE\_USER | RESTRICTED\_USER | MULTI\_USER**

The three options SINGLE\_USER, RESTRICTED\_USER, and MULTI\_USER describe the user

Access property of a database. They are mutually exclusive; setting any one of them unsets

the others. To set one of these options for your database, you just use the option name. For

example, to set the AdventureWorks2008 database to single-user mode, use the following code:

**ALTER DATABASE AdventureWorks2008 SET SINGLE\_USER;**

A database in SINGLE\_USER mode can have only one connection at a time. A database in RESTRICTED\_USER mode can have connections only from users who are considered “qualified”—those who are members of the dbcreator or sysadmin server role or the db\_owner

role for that database. The default for a database is MULTI\_USER mode, which means anyone with a valid user name in the database can connect to it. If you attempt to change a database’s state to a mode that is incompatible with the current conditions—for example, if you try to change the database to SINGLE\_USER mode when other connections exist—the behavior of SQL Server is determined by the TERMINATION option you specify. I’ll discuss termination

options shortly. To determine which user access value is set for a database, you can examine the sys.databases

catalog view, as shown here:

**SELECT USER\_ACCESS\_DESC FROM sys.databases**

**WHERE name = '<name of database>';**

This query will return one of MULTI\_USER, SINGLE\_USER, or RESTRICTED\_USER.

**OFFLINE | ONLINE | EMERGENCY**

You use the OFFLINE, ONLINE, and EMERGENCY options to describe the status of a database.

They are mutually exclusive. The default for a database is ONLINE. As with the user access

options, when you use ALTER DATABASE to put the database in one of these modes, you

don’t specify a value of ON or OFF—you just use the name of the option. When a database

is set to OFFLINE, it is closed and shut down cleanly and marked as offline. The database

cannot be modified while the database is offline. A database cannot be put into OFFLINE

mode if there are any connections in the database. Whether SQL Server waits for the other

connections to terminate or generates an error message is determined by the TERMINATION

option specified.The following code examples show how to set a database’s status value to OFFLINE and how

to determine the status of a database:

**ALTER DATABASE AdventureWorks2008 SET OFFLINE;**

**SELECT state\_desc from sys.databases**

**WHERE name = 'AdventureWorks2008';**

A database can be explicitly set to EMERGENCY mode, and that option will be discussed later

in conjunction with DBCC commands.As shown in the preceding query, you can determine the current status of a database by examining the state\_desc column of the sys.databases view. This column can return status values

other than OFFLINE, ONLINE, and EMERGENCY, but those values are not directly settable using ALTER DATABASE. A database can have the status value RESTORING while it is in the process of being restored from a backup. It can have the status value RECOVERING during a restart of SQL Server.

The recovery process is performed on one database at a

time, and until SQL Server has finished recovering a database, the database has a status of

RECOVERING. If the recovery process cannot be completed for some reason (most likely

Because one or more of the log files for the database is unavailable or unreadable), SQL

Server gives the database the status of RECOVERY\_PENDING. Your databases can also be put

into RECOVERY\_PENDING mode if SQL Server runs out of either log or data space during

rollback recovery, or if SQL Server runs out of locks or memory during any part of the startup

process. I’ll go into more detail about the difference between rollback recovery and startup

recovery in later If all the needed resources, including the log files, are available, but corruption is detected

during recovery, the database may be put in the SUSPECT state. You can determine the state

value by looking at the state\_desc column in the sys.databases view. A database is completely

unavailable if it’s in the SUSPECT state, and you will not even see the database listed if you

run sp\_helpdb. However, you can still see the status of a suspect database in the sys.databases

view. In many cases, you can make a suspect database available for read-only operations by setting its status to EMERGENCY mode. If you really have lost one or more of the log files

for a database, EMERGENCY mode allows you to access the data while you copy it to a new

location. When you move from RECOVERY\_PENDING to EMERGENCY, SQL Server shuts down

the database and then restarts it with a special flag that allows it to skip the recovery process.

Skipping recovery can mean you have logically or physically inconsistent data—missing index

rows, broken page links, or incorrect metadata pointers. By specifically putting your database

in EMERGENCY mode, you are acknowledging that the data might be inconsistent but that you want access to it anyway.

**READ\_ONLY | READ\_WRITE**

These options describe the updatability of a database. They are mutually exclusive. The default

for a database is READ\_WRITE. As with the user access options, when you use ALTER DATABASE

to put the database in one of these modes, you don’t specify a value of ON or OFF, you just

use the name of the option. When the database is in READ\_WRITE mode, any user with the appropriate

permissions can carry out data modification operations. In READ\_ONLY mode, no INSERT, UPDATE, or DELETE operations can be executed. In addition, because no modifications are done when a database is in READ\_ONLY mode, automatic recovery is not run on this database when SQL Server is restarted, and no locks need to be acquired during any SELECT operations. Shrinking a database in READ\_ONLY mode is not possible.A database cannot be put into READ\_ONLY mode if there are any connections to the database. Whether SQL Server waits for the other connections to terminate or generates an error message is determined by the TERMINATION option specified.

The following code shows how to set a database’s updatability value to READ\_ONLY and how

to determine the updatability of a database:

**ALTER DATABASE AdventureWorks2008 SET READ\_ONLY;**

**SELECT name, is\_read\_only FROM sys.databases**

**WHERE name = 'AdventureWorks2008';**

When READ\_ONLY is enabled for database, the is\_read\_only column returns 1; otherwise, for

a READ\_WRITE database, it returns 0.

**ROLLBACK AFTER integer [SECONDS]**

This option causes SQL Server to wait for the specified number of seconds and then break unqualified connections. Incomplete transactions are rolled back. When the transition is to SINGLE\_USER mode, all connections

are unqualified except the one issuing the ALTER DATABASE command.When the transition is to RESTRICTED\_USER mode, unqualified connections are those of users who are not members of the db\_owner fixed database role or the dbcreator and sysadmin fixed server roles.

**ROLLBACK IMMEDIATE**

This option breaks unqualified connections immediately. All incomplete transactions are rolled back. Keep in mind that although the connectionmay be broken immediately, the rollback might take some time to complete. All work

done by the transaction must be undone, so for certain operations, such as a batch update of millions of rows or a large index rebuild, you could be in for a long wait.Unqualified connections are the same as those described previously.

**NO\_WAIT**

This option causes SQL Server to check for connections before attempting to change the database state and causes the ALTER DATABASE command to fail if certain connections exist. If the database is being set to SINGLE\_USER mode, the

ALTER DATABASE command fails if any other connections exist. If the transition is to RESTRICTED\_USER mode, the ALTER DATABASE command fails if any unqualified connections exist.The following command changes the user access option of the AdventureWorks2008 database to SINGLE\_USER and generates an error if any other connections to the AdventureWorks2008 database exist:

**ALTER DATABASE AdventureWorks2008 SET SINGLE\_USER WITH NO\_WAIT;**

**Database Page verify Option**

**Checksum vs. Torn Page Detection**

You might wonder why the switch from Torn Page Detection to Checksums, and what’s the difference.  The difference is in how they validate IO.  When it writes a page, Torn Page Detection grabs the first 2 bits of every 512 byte sector on each page and stores those bits in the page header.  When the page is later read back in from disk, SQL Server compares those header bits with the bits from the sector, to make sure they’re still the same.  Now, that check is better than nothing, but you can see where it might potentially miss corruption in the rest of the sector, right?

The Checksum option, on the other hand, creates a checksum value using the content of the entire page, and saves that value in the header.  When a page is read from disk, a checksum is created again and compared to the saved checksum.  Since any difference in the bytes read for that page will result in a different checksum value, this is a far more thorough method for validating IO.

|  |  |
| --- | --- |
|  | **select object\_id('tab1')**    **DBCC TRACEON (3604);**  **GO**    **-- get page numbers**  **DBCC IND ('Checksum\_test', 350624292, 1)**    **-- view page**  **DBCC PAGE ('Checksum\_test', 1, 154, 3)** |

SQL Server Recovery model



SQL Server Backup and Restoration





## How to Backup All SQL Server Databases

1. Specify path to store database backups
2. Specify backup file name format
3. Select list of databases to backup
4. Loop through databases
5. Programmatically create database backup command using database name, path and file name format
6. See attached SQL Server backup script

## File Naming Format DBname\_YYYYDDMM.BAK

Here is the script that will allow you to backup each database within your instance of SQL Server.  You will need to change the**@path** to the appropriate backup directory.

DECLARE @name VARCHAR(50) -- database name

DECLARE @path VARCHAR(256) -- path for backup files

DECLARE @fileName VARCHAR(256) -- filename for backup

DECLARE @fileDate VARCHAR(20) -- used for file name

-- specify database backup directory

SET @path = 'C:\Backup\'

-- specify filename format

SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112)

DECLARE db\_cursor CURSOR READ\_ONLY FOR

SELECT name

FROM master.dbo.sysdatabases

WHERE name NOT IN ('master','model','msdb','tempdb') -- exclude these databases

OPEN db\_cursor

FETCH NEXT FROM db\_cursor INTO @name

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @fileName = @path + @name + '\_' + @fileDate + '.BAK'

BACKUP DATABASE @name TO DISK = @fileName

FETCH NEXT FROM db\_cursor INTO @name

END

CLOSE db\_cursor

DEALLOCATE db\_cursor

## File Naming Format DBname\_YYYYDDMM\_HHMMSS.BAK

If you want to also include the time in the filename you can replace this line in the above script:

-- specify filename format

SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112)

with this line:

-- specify filename format

SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112) + REPLACE(CONVERT(VARCHAR(20),GETDATE(),108),':','')

# Automating Transaction Log Backups for All SQL Server Databases

With the use of T-SQL you can generate your transaction log backups and with the use of cursors you can cursor through all of your databases to back them up one by one. With the use of the DATABASEPROPERTYEX function we can also just address databases that are either in the FULL or BULK\_LOGGED recovery model since you can not issue transaction log backups against databases in the SIMPLE recovery mode.

Here is the script that will allow you to backup the transaction log for each database within your instance of SQL Server that is either in the FULL or BULK\_LOGGED recovery model.

You will need to change the @path to the appropriate backup directory and each backup file will take on the name of "DBname\_YYYDDMM\_HHMMSS.TRN".

DECLARE @name VARCHAR(50) -- database name

DECLARE @path VARCHAR(256) -- path for backup files

DECLARE @fileName VARCHAR(256) -- filename for backup

DECLARE @fileDate VARCHAR(20) -- used for file name

SET @path = 'C:\Backup\'

SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112)

+ '\_'

+ REPLACE(CONVERT(VARCHAR(20),GETDATE(),108),':','')

DECLARE db\_cursor CURSOR FOR

SELECT name

FROM master.dbo.sysdatabases

WHERE name NOT IN ('master','model','msdb','tempdb')

AND DATABASEPROPERTYEX(name, 'Recovery') IN ('FULL','BULK\_LOGGED')

OPEN db\_cursor

FETCH NEXT FROM db\_cursor INTO @name

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @fileName = @path + @name + '\_' + @fileDate + '.TRN'

BACKUP LOG @name TO DISK = @fileName

FETCH NEXT FROM db\_cursor INTO @name

END

CLOSE db\_cursor

DEALLOCATE db\_cursor

**Copy Only Backup**

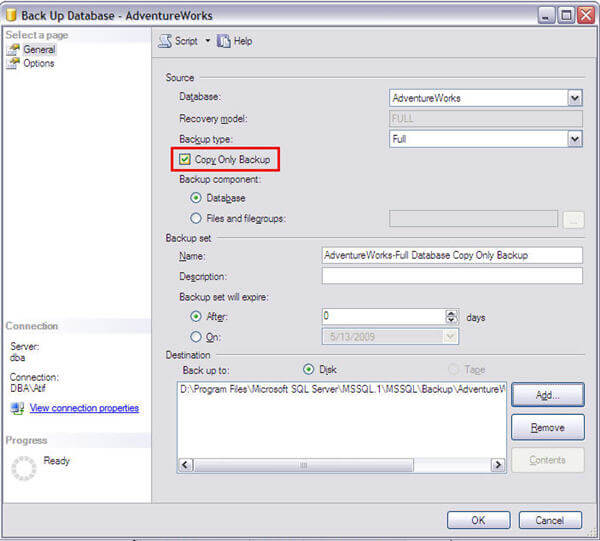
Fortunately in SQL Server 2005 and onwards we have a backup option for such a situation. This backup option is known as Copy Only backups. This option is specifically for creating an ad hoc backup that will not disturb the planned restore sequence for that database.

Copy Only backups can be used for creating a full backup or a transaction log backup. This option is not implemented for differential backups. In practical scenarios you will rarely need to create a Copy Only log backup, however the copy only option may be frequently used with full backups.

Although the Copy Only option is available for SQL Server 2005 there is not a way to create them using SSMS in SQL 2005. If you have SSMS 2008 you can use the GUI or you have to use a T-SQL statement to create Copy Only backups.

First let's look at how this can be done using SSMS 2008 to create Copy Only backup

* Go to the backup database window as you would for a normal backup



* Just below the "Backup type" menu, you will find a check box for "Copy Only Backup"
* Click this check box
* Fill out all other related information as you would for a normal backup and click OK

Note: The check box for "Copy Only Backup" will also be active for differential backups, but it will have no affect other than creating a normal differential backup.

Second, to create a Copy Only backup with T-SQL you can issue the following command:

-- Create full backup with Copy Only option

BACKUP DATABASE AdventureWorks

TO DISK = 'D:\WithoutCopyOnly\_AdventureWorks.bak'

WITH COPY\_ONLY

GO

## Proving The Concept

Before going further it will be good to summarize the concept that a full backup with the copy only option is independent of the sequence of your other normal backups. So after you create a backup with the copy only option, you would be able to work with the recovery plan based on your scheduled backups without this impacting your restore process.

We will use LSN (log sequence number) information to track the full backup that is the base for the differential backups. First we will note the LSN for the differential base of the AdventureWorks database.

--Script # 1: Note current differential base LSN

SELECT DB\_NAME(database\_id) AS [DB Name],

differential\_base\_lsn AS 'Note differential base LSN'

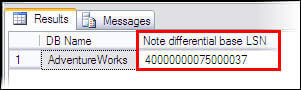
FROM sys.master\_files

WHERE database\_id = DB\_ID('AdventureWorks')

AND type\_desc = 'ROWS'

GO

The differential\_base\_lsn affects the sequence in which combination of backups are to be restored in a recovery. You may also get detailed information about the LSN of any database from table msdb..backupset.



Now we have to confirm that a full backup without the Copy Only option will update the differential\_base\_lsn. For this purpose we will issue a full backup command without the Copy Only option and we will note the change in the LSN to prove that the updated differential base LSN is the LSN of our last full backup.

--Script # 2: Create full backup and compare LSN information

-- Create full backup

-- Run script after changing Backup path

BACKUP DATABASE AdventureWorks

TO DISK = 'D:\WithoutCopyOnly\_AdventureWorks.bak'

GO

-- Get differential\_base\_lsn after full backup

SELECT DB\_NAME(database\_id) AS [DB Name],

differential\_base\_lsn AS 'Updated differential base LSN'

FROM sys.master\_files

WHERE database\_id = DB\_ID('AdventureWorks')

AND type\_desc = 'ROWS'

GO

-- Get LSN of recent full backup for match purpose

SELECT database\_name, backup\_start\_date, is\_copy\_only,

first\_lsn as 'LSN of full bakup'

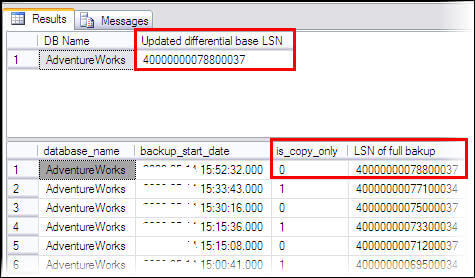
FROM msdb..backupset

WHERE database\_name = 'AdventureWorks'

ORDER BY backup\_start\_date DESC

GO

In the following result set, we can verify that LSN has been changed for the differential backup restore base and matches our full backup LSN.  Both marked LSNs below are the same which confirms that the last backup is our differential base.



Now we will create a full backup with the Copy Only option and it will be proved that the full backup with the Copy Only option will not affect the differential base LSN of our database. In other words the full backup with the Copy Only option will not affect the base full backup for the differential backups.

--Script # 3: Create full backup with copy only option and compare LSN information

-- Create full backup with copy only option

-- Run script after changing Backup path

BACKUP DATABASE AdventureWorks

TO DISK = 'D:\CopyOnly\_AdventureWorks.bak'

WITH COPY\_ONLY

GO

-- Get differential\_base\_lsn after full backup with copy only option

SELECT DB\_NAME(database\_id) AS [DB Name],

differential\_base\_lsn AS 'Un changed differential base LSN'

FROM sys.master\_files

WHERE database\_id = DB\_ID('AdventureWorks')

AND type\_desc = 'ROWS'

GO

-- Get LSN of recent full backup with copy only option for match purpose

SELECT database\_name, backup\_start\_date, is\_copy\_only,

first\_lsn as 'LSN of last full bakup'

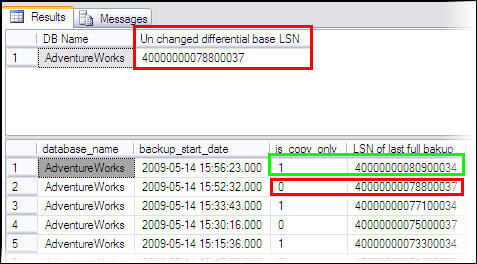
FROM msdb..backupset

WHERE database\_name = 'AdventureWorks'

ORDER BY backup\_start\_date DESC

GO

In the following image we can verify that after a full backup with the Copy Only option the differential base LSN is unchanged and it matches the LSN of the previous full backup (both are marked red). Also note that the last full backup with the Copy Only option is also there (marked green).



Following considerations will be helpful while using backups with the Copy Only option.

* Copy Only option will also work for compatibility level 80 databases in a SQL Server 2005 instance
* Transaction log backups with the Copy Only option preserves the existing log archive point, hence it will not truncate the transaction logs of that database.
* There is no enhanced consideration required while restoring a backup created with the Copy Only option.
* A full backup with Copy Only option can not be used as a base for restoring differential backups
* A log backup with Copy Only option may be created for databases with recovery model full or bulk logged only.
* A full backup with Copy Only option may be created for databases with any recovery model.

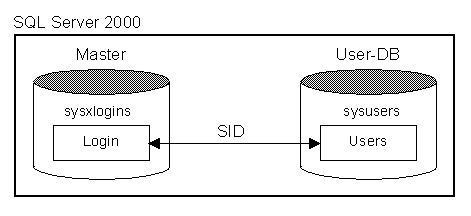
# SQL Server Logins and Users

**Overview**

Although the terms **login** and **user** are often used interchangeably, **they are very different**.

* A login is used for user authentication
* A database user account is used for database access and permissions validation.

Logins are **associated to users by the security identifier (SID)**. A login is required for access to the SQL Server server. The process of verifying that a particular login is valid is called "authentication". This login must be associated to a SQL Server database user. You use the user account to control activities performed in the database. If no user account exists in a database for a specific login, the user that is using that login cannot access the database even though the user may be able to connect to SQL Server. The single exception to this situation is when the database contains the "guest" user account. A login that does not have an associated user account is mapped to the guest user. Conversely, if a database user exists but there is no login associated, the user is not able to log into SQL Server server.



When a database is restored to a different server it contains a set of users and permissions but there **may not be any corresponding logins** or the logins may not be associated with the same users. This condition is known as having **"orphaned users."**

**Troubleshooting Orphaned Users**

When you restore a database backup to another server, you may experience a problem with orphaned users. The following scenario illustrates the problem and shows how to resolve it.

**Use master  
sp\_addlogin 'test', 'password', 'Northwind'**

**SELECT** **sid FROM dbo.sysxlogins WHERE name = 'test'**  
0xE5EFF2DB1688C246855B013148882E75

Grant access to the user you just created

**Use Northwind  
sp\_grantdbaccess 'test'**

**SELECT** **sid FROM dbo.sysusers WHERE name = 'test'**  
0xE5EFF2DB1688C246855B013148882E75

As you can see, both SID's are identical.

Backup the database

**Use master  
BACKUP DATABASE Northwind  
TO DISK = 'C:\Northwind.bak'**

Copy the Backupfile to another Maschine and SQL Server and restore it as follows:

**RESTORE FILELISTONLY  
FROM DISK = 'C:\Users\Zahn\Work\Northwind.bak'**  
Northwind  
Northwind\_log

**RESTORE DATABASE TestDB  
FROM DISK = 'C:\Users\Zahn\Work\Northwind.bak'  
WITH  
 MOVE 'Northwind' TO 'D:\DataMSSQL\Data\northwnd.mdf',  
 MOVE 'Northwind\_log' TO 'D:\DataMSSQL\Data\northwnd.ldf'**

The restored database contains a user named "test" without a corresponding login, which results in "test" being orphaned.

Check the SID's

**Use master  
SELECT** **sid FROM dbo.sysxlogins WHERE name = 'test'**  
0x39EE98D37EAC2243B7833705EC1C60E3

**Use TestDB  
SELECT** **sid FROM dbo.sysusers WHERE name = 'test'**  
0xE5EFF2DB1688C246855B013148882E75

Now, to detect orphaned users, run this code

**Use TestDB  
sp\_change\_users\_login 'report'**  
test 0xE5EFF2DB1688C246855B013148882E75

The output lists all the logins, which have a mismatch between the entries in the **sysusers** system table, of the TestDB database, and the**sysxlogins** system table in the master database.

Resolve Orphaned Users

**Use TestDB  
sp\_change\_users\_login 'update\_one', 'test', 'test'**

**SELECT** **sid FROM dbo.sysusers WHERE name = 'test'**  
0x39EE98D37EAC2243B7833705EC1C60E3  
  
**use master  
SELECT** **sid FROM dbo.sysxlogins WHERE name = 'test'**  
0x39EE98D37EAC2243B7833705EC1C60E3

This relinks the server login "test" with the the TestDB database user "test". The **sp\_change\_users\_login** stored procedure can also perform an update of all orphaned users with the "auto\_fix" parameter but this is not recommended because SQL Server attempts to match logins and users by name. For most cases this works; however, if the wrong login is associated with a user, a user may have incorrect permissions.

**Roles and Schema**

****

**Database Important activity**



**Database Mail**

****

**DBCC CHECKDB**



**Database and Database Server performance monitiring**

**Index and Index optimization**



**Update Statistices**



**Introduction of Locks in SQL Server**



**Mostly used DMV**

* [sys.dm\_exec\_cached\_plans](http://msdn2.microsoft.com/en-us/library/ms187404.aspx) - Cached query plans available to SQL Server
* [sys.dm\_exec\_sessions](http://msdn2.microsoft.com/en-us/library/ms176013.aspx) - Sessions in SQL Server
* [sys.dm\_exec\_connections](http://msdn2.microsoft.com/en-us/library/ms181509.aspx) - Connections to SQL Server
* [sys.dm\_db\_index\_usage\_stats](http://msdn2.microsoft.com/en-us/library/ms188755.aspx) - Seeks, scans, lookups per index
* [sys.dm\_io\_virtual\_file\_stats](http://msdn2.microsoft.com/en-us/library/ms190326.aspx) - IO statistics for databases and log files
* [sys.dm\_tran\_active\_transactions](http://msdn2.microsoft.com/en-us/library/ms174302.aspx) - Transaction state for an instance of SQL Server

[sys.dm\_exec\_sql\_text](http://msdn.microsoft.com/en-us/library/ms181929.aspx) - Returns TSQL code

* [sys.dm\_exec\_query\_plan](http://msdn.microsoft.com/en-us/library/ms189747.aspx) - Returns query plan
* [sys.dm\_os\_wait\_stats](http://msdn.microsoft.com/en-us/library/ms179984.aspx)- Returns information what resources SQL is waiting on
* [sys.dm\_os\_performance\_counters](http://msdn.microsoft.com/en-us/library/ms187743.aspx)- Returns performance monitor counters related to SQL Server

**Frequently Used DBCC Commands**

1.DBCC CHECKALLOC  
DBCC CHECKALLOC checks page usage and allocation in the database. Use this command if allocation errors are found for the database. If you run DBCC CHECKDB, you do not need to run DBCC CHECKALLOC, as DBCC CHECKDB includes the same checks (and more) that DBCC CHECKALLOC performs.  
  
  
2.DBCC CHECKCATALOG  
This command checks for consistency in and between system tables. This command is not executed within the DBCC CHECKDB command, so running this command weekly is recommended.  
  
3.DBCC CHECKCONSTRAINTS  
DBCC CHECKCONSTRAINTS alerts you to any CHECK or constraint violations.  
Use it if you suspect that there are rows in your tables that do not meet the constraint or CHECK constraint rules.  
  
4.DBCC CHECKDB  
A very important DBCC command, DBCC CHECKDB should run on your SQL Server instance on at least a weekly basis. Although each release of SQL Server reduces occurrences of integrity or allocation errors, they still do happen. DBCC CHECKDB includes the same checks as DBCC CHECKALLOC and DBCC CHECKTABLE. DBCC CHECKDB can be rough on concurrency, so be sure to run it at off-peak times.  
  
5.DBCC CHECKTABLE  
DBCC CHECKTABLE is almost identical to DBCC CHECKDB, except that it is performed at the table level, not the database level. DBCC CHECKTABLE verifies index and data page links, index sort order, page pointers, index pointers, data page integrity, and page offsets. DBCC CHECKTABLE uses schema locks by default, but can use the TABLOCK option to acquire a shared table lock. CHECKTABLE also performs object checking using parallelism by default (if on a multi-CPU system).  
  
6.DBCC CHECKFILEGROUP  
DBCC CHECKFILEGROUP works just like DBCC CHECKDB, only DBCC CHECKFILEGROUP checks the specified filegroup for allocation and structural issues. If you have a very large database (this term is relative, and higher end systems may be more apt at performing well with multi-GB or TB systems ) , running DBCC CHECKDB may be time-prohibitive.  
If your database is divided into user defined filegroups, DBCC CHECKFILEGROUP will allow you to isolate your integrity checks, as well as stagger them over time.  
  
7.DBCC CHECKIDENT  
DBCC CHECKIDENT returns the current identity value for the specified table, and allows you to correct the identity value if necessary.  
  
8.DBCC DBREINDEX  
If your database allows modifications and has indexes, you should rebuild your indexes on a regular basis. The frequency of your index rebuilds depends on the level of database activity, and how quickly your database and indexes become fragmented. DBCC DBREINDEX allows you to rebuild one or all indexes for a table. Like DBCC CHECKDB, DBCC CHECKTABLE, DBCC CHECKALLOC, running DBREINDEX during peak activity times can significantly reduce concurrency.  
  
9.DBCC INDEXDEFRAG  
Microsoft introduced the excellent DBCC INDEXDEFRAG statement beginning with SQL Server 2000. This DBCC command, unlike DBCC DBREINDEX, does not hold long term locks on indexes. Use DBCC INDEXDEFRAG for indexes that are not very fragmented, otherwise the time this operation takes will be far longer then running DBCC DBREINDEX. In spite of it's ability to run during peak periods, DBCC INDEXDEFRAG has had limited effectiveness compared to DBCC DBREINDEX (or drop/create index).  
  
10.DBCC INPUTBUFFER  
The DBCC INPUTBUFFER command is used to view the last statement sent by the client connection to SQL Server. When calling this DBCC command, you designate the SPID to examine. (SPID is the process ID, which you can get from viewing current activity in Enterprise Manager or executing sp\_who. )  
  
11.DBCC OPENTRAN  
DBCC OPENTRAN is a Transact-SQL command that is used to view the oldest running transaction for the selected database. The DBCC command is very useful for troubleshooting orphaned connections (connections still open on the database but disconnected from the application or client), and identification of transactions missing a COMMIT or ROLLBACK. This command also returns the oldest distributed and undistributed replicated transactions, if any exist within the database. If there are no active transactions, no data will be returned. If you are having issues with your transaction log not truncating inactive portions, DBCC OPENTRAN can show if an open transaction may be causing it.  
  
12.DBCC PROCCACHE  
You may not use this too frequently, however it is an interesting DBCC command to execute periodically, particularly when you suspect you have memory issues. DBCC PROCCACHE provides information about the size and usage of the SQL Server procedure cache.  
  
13.DBCC SHOWCONTIG  
The DBCC SHOWCONTIG command reveals the level of fragmentation for a specific table and its indices. This DBCC command is critical to determining if your table or index has internal or external fragmentation. Internal fragmentation concerns how full an 8K page is.  
When a page is underutilized, more I/O operations may be necessary to fulfill a query request than if the page was full, or almost full.  
External fragmentation concerns how contiguous the extents are. There are eight 8K pages per extent, making each extent 64K. Several extents can make up the data of a table or index. If the extents are not physically close to each other, and are not in order, performance could diminish.  
  
14.DBCC SHRINKDATABASE  
DBCC SHRINKDATABASE shrinks the data and log files in your database.  
Avoid executing this command during busy periods in production, as it has a negative impact on I/O and user concurrency. Also remember that you cannot shrink a database past the target percentage specified, shrink smaller than the model database, shrink a file past the original file creation size, or shrink a file size used in an ALTER DATABASE statement.  
  
15.DBCC SHRINKFILE  
DBCC SHRINKFILE allows you to shrink the size of individual data and log files. (Use sp\_helpfile to gather database file ids and sizes).  
  
16. DBCC TRACEOFF, TRACEON, TRACESTATUS  
Trace flags are used within SQL Server to temporarily enable or disable specific SQL Server instance characteristics. Traces are enabled using the DBCC TRACEON command, and disabled using DBCC TRACEOFF. DBCC TRACESTATUS is used to displays the status of trace flags. You'll most often see TRACEON used in conjunction with deadlock logging (providing more verbose error information).  
  
17.DBCC USEROPTIONS  
Execute DBCC USEROPTIONS to see what user options are in effect for your specific user connection. This can be helpful if you are trying to determine if you current user options are inconsistent with the database options.

**Important Trace Flag**



**SQL SERVER performance tuning**

****

**Query Execution Plan**



**SQL Server Profiler**

****

**SQL Server High availability**

**Log Shipping All Documents**

****

**Mirroring all documents**

****

**SQL Upgrade**

****