



# Hands-on: Understanding the importance of SQL Server Database Log

*Palak Patel*  
[palak@sqlsquare.com](mailto:palak@sqlsquare.com)

The hands-on exercise is designed to help learn following SQL Server skills

- Understand the importance of Log files in your disaster recovery strategy
- Create Database using T-SQL
- Create Backup Device using T-SQL
- Backup Database and Log using T-SQL
- Restore & Recover Database using T-SQL

## Resources & Prerequisites:

- Single Computer running
  - Windows 2000 Server or Windows 2003 Server
  - SQL Server 2000 Standard Edition (Database Server and Client Tools required)
- Knowledge of SQL Server Database Files, File groups, Backups Query Analyzer is recommended.

## Hands-on:

### Step 1: Create a folder on your hard drive to store the testing database.

Create a Folder called **SQLTest** on your **C:** Drive. If you would like to name the folder differently and place it somewhere else than C: drive, you will be required to change some of the scripts accordingly.

### Step 2: Run the Following Script in Query Analyzer to create the database.

We will create the database with 2 data files (i.e. one mandatory primary data file - .mdf extension and one secondary data file - .ndf extension). We will have two file groups namely, PRIMARY (SQL Server creates it by default and user created FGROUP). We will place the secondary data file in the File Group FGROUP. We are creating our database with this structure to simulate crash later on. We will switch of the SQL Server Service and simply rename this secondary data file. When the SQL Server is started again, it will not load the database as usual as the file is missing. We will assume that the file that we renamed was deleted due to let's say a hard disk crash.

```
CREATE DATABASE UTest
ON
PRIMARY ( NAME = UTestData1,
          FILENAME = 'C:\SQLTest\UTestData1.mdf',
          SIZE = 1MB,
          MAXSIZE = 10,
          FILEGROWTH = 1),
FILEGROUP FGROUP
( NAME = UTestData2,
  FILENAME = 'C:\SQLTest\UtestData2.ndf',
  SIZE = 1MB,
  MAXSIZE = 10,
  FILEGROWTH = 1)
```

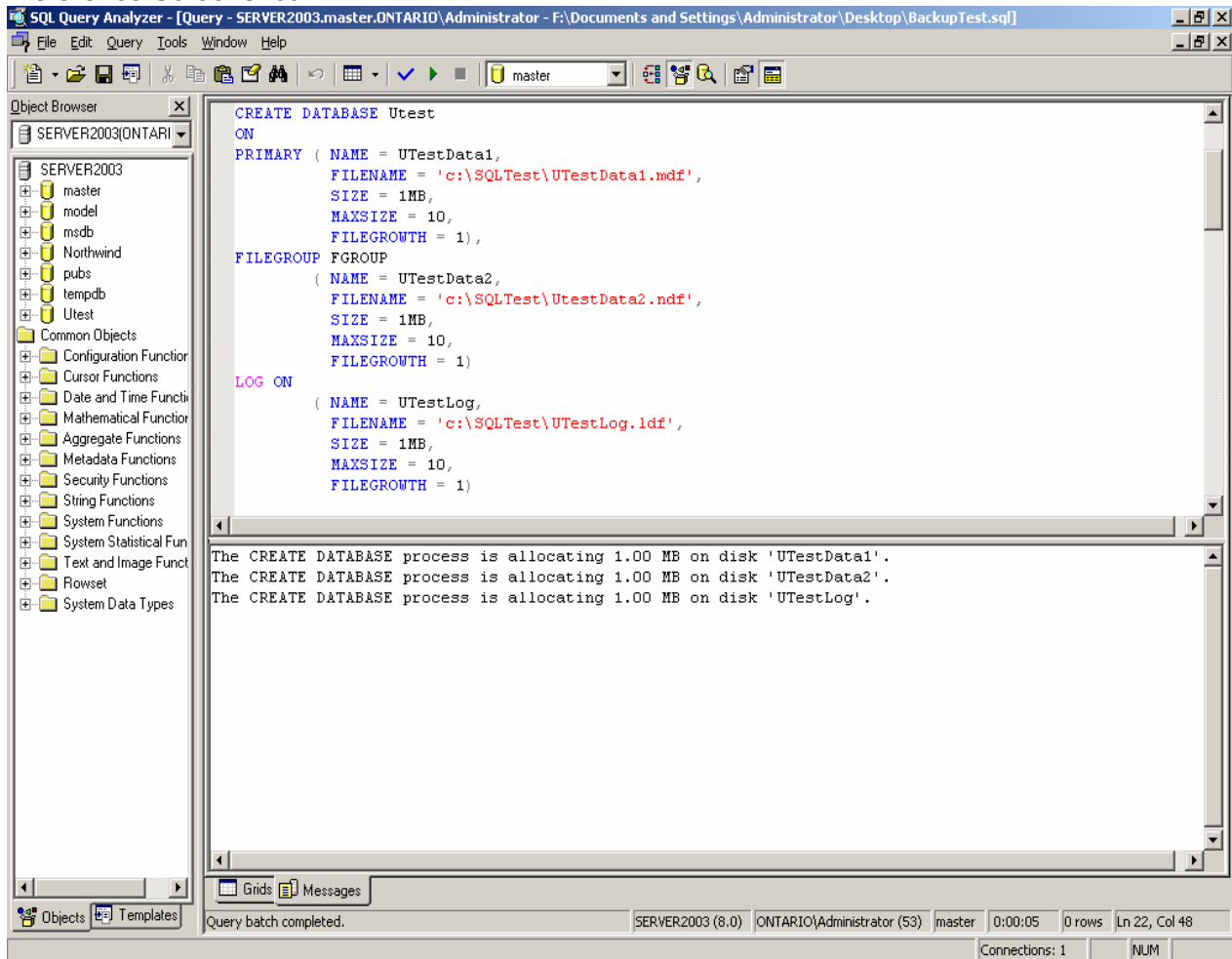
## LOG ON

```
( NAME = UTestLog,  
  FILENAME = 'C:\SQLTest\UTestLog.ldf',  
  SIZE = 1MB,  
  MAXSIZE = 10,  
  FILEGROWTH = 1)
```

## Output:

The CREATE DATABASE process is allocating 1.00 MB on disk 'UTestData1'.  
The CREATE DATABASE process is allocating 1.00 MB on disk 'UTestData2'.  
The CREATE DATABASE process is allocating 1.00 MB on disk 'UTestLog'.

## Reference Screenshot::



A New database called 'Utest' has been created.

**Step 3: We will make sure that the newly created database does not truncate the log on checkpoint.**

Please refer to SQL Server Books online, for more information on Truncate Log on Check point and sp\_dboption system stored procedure.

Run the Following script in Query Analyzer;

```
sp_dboption 'Utest' , 'trunc. log on chkpt.' , 'false'
```

Output:  
The command(s) completed successfully.

#### Step 4: Create Table in the Database

Run the Following script in Query Analyzer to make sure that we are using the newly created database to create and populate the objects.

```
USE UTest  
GO
```

Create a Table on File Group FGROUPE:

```
CREATE TABLE TTable (  
    NID int,  
    NDESC char(20)  
) ON FGROUPE
```

Output:  
The command(s) completed successfully.

#### Step 5: Add four rows in the newly created Table

```
INSERT INTO TTable (NID, NDESC) values (1, 'One')  
INSERT INTO TTable (NID, NDESC) values (2, 'Two')  
INSERT INTO TTable (NID, NDESC) values (3, 'Three')  
INSERT INTO TTable (NID, NDESC) values (4, 'Four')
```

Output:  
(1 row(s) affected)  
  
(1 row(s) affected)  
  
(1 row(s) affected)  
  
(1 row(s) affected)

At this point the following command will list all the rows in the table.

```
SELECT * FROM TTable
```

Output:

NID	NDESC
1	One
2	Two

3 Three  
4 Four

(4 row(s) affected)

### Step 6: Take a Complete Backup of the Database.

Let's say we are taking a complete backup of the database at this point.

To take Backup of the database, let's add a Backup Device.

```
EXEC sp_addumpdevice 'disk', 'UTestDump', 'C:\SQLTest\UtestDump.bak'
```

Output:

(1 row(s) affected)

'Disk' device added.

Backup the database.

```
BACKUP DATABASE UTest to UTestDump
```

Output:

Processed 80 pages for database 'UTest', file 'UTestData1' on file 1.

Processed 16 pages for database 'UTest', file 'UTestData2' on file 1.

Processed 1 pages for database 'UTest', file 'UTestLog' on file 1.

BACKUP DATABASE successfully processed 97 pages in 0.614 seconds (1.286 MB/sec).

### Step 7: Add two more rows in to the table:

```
INSERT INTO TTable (NID, NDESC) values (5, 'Five')
```

```
INSERT INTO TTable (NID, NDESC) values (6, 'Six')
```

Output:

(1 row(s) affected)

(1 row(s) affected)

At this point the following command will list all the rows in the table.

```
SELECT * FROM TTable
```

Output:

NID	NDESC
-----	-------

-----

1	One
---	-----

2	Two
---	-----

3	Three
---	-------

4	Four
---	------

5	Five
---	------

6	Six
---	-----

(6 row(s) affected)

### Step 8: Take backup of the Transaction Log.

We are taking backup of the log at this point. To make sure that the last two records (namely, 5 and 6) are backed up as part of this log backup.

#### BACKUP LOG UTest to UTestDump

Output:

Processed 1 pages for database 'UTest', file 'UTestLog' on file 2.  
BACKUP LOG successfully processed 1 pages in 0.403 seconds (0.016 MB/sec).

### Step 9: Add two more records into the table.

```
INSERT INTO TTable (NID, NDESC) values (7, 'Seven')  
INSERT INTO TTable (NID, NDESC) values (8, 'Eight')
```

Output:

(1 row(s) affected)

(1 row(s) affected)

At this point the following command will list all the rows in the table.

```
SELECT * FROM TTable
```

Output:

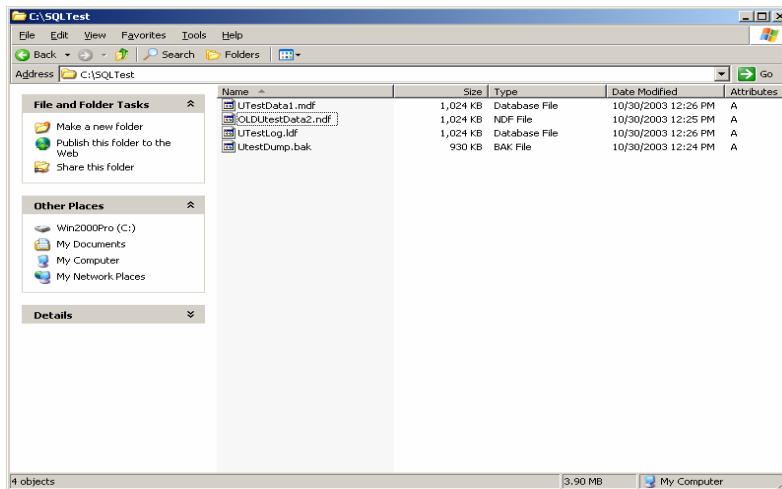
NID	NDESC
1	One
2	Two
3	Three
4	Four
5	Five
6	Six
7	Seven
8	Eight

(8 row(s) affected)

### Step 10: Simulate Crash.

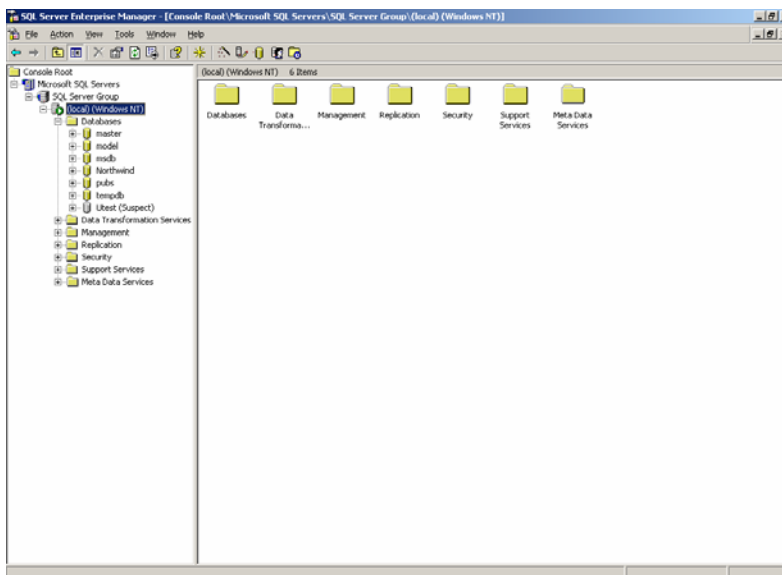
(a) Stop SQL Server Service using any method that is convenient to you.

(b) Go to Windows Explorer and browse to C:\SQLTest Folder. Rename UTestData2.ndf file to something else.



(c) Start SQL Server Service using any method that is suitable to you.

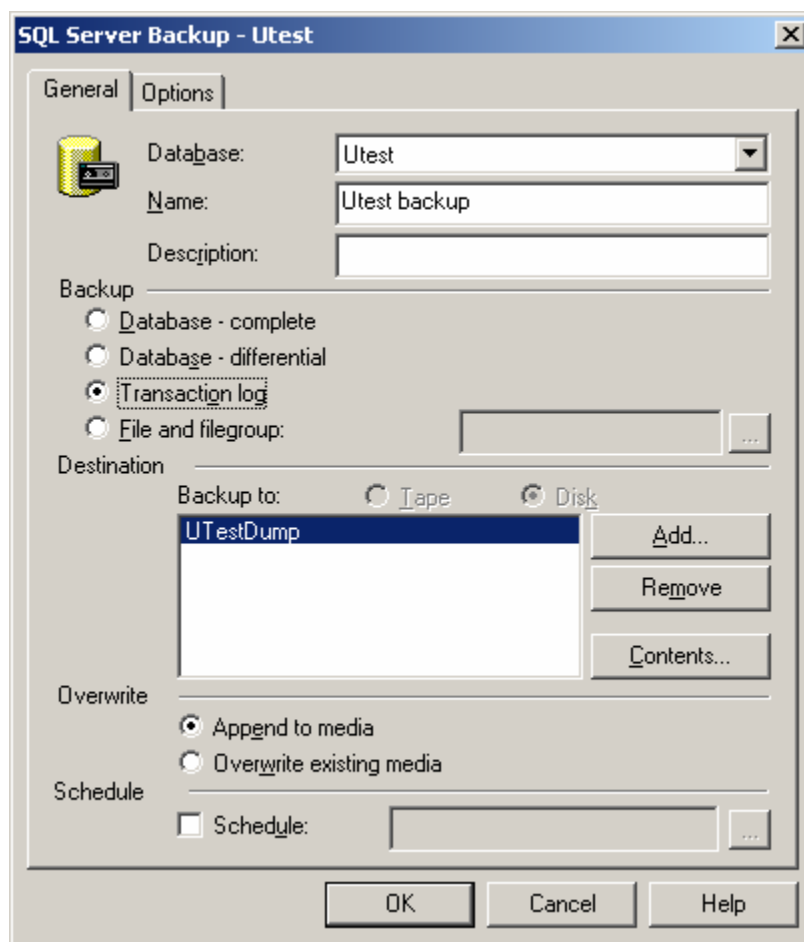
(d) When you start the SQL Server & inspect Enterprise Manager, it should display (suspect) next to UTest Database. Since, SQL Server could not find UTestData2.ndf file while loading this database, it put the database into Suspect mode. See the screenshot.



### Step 11: Start the Recovery by taking a backup of the Transaction Log

We have a problem with the database. Since, we do not have any backup of the last two rows added in the TTable table of our database and we have log files that are not damaged, we can try to take backup of the Log.

To take backup of the log, let's use Enterprise Manager this time, right click the database 'UTest' in Enterprise Manager, click All Task and Backup. The Backup Dialog box will appear as shown below. Make sure that you choose the appropriate backup device (refer to screenshot). Click OK and SQL Server will perform backup of the log for the database (this will take care of the last two rows, namely 7 & 8 that we added).



## Step 12: Restore the Full Backup and the Transaction Log Backups

Since, we have successfully taken backup of the log after the system has crashed, we can try to restore the database, let's do it using Query Analyzer this time.

```

RESTORE DATABASE Utest
FROM UtestDump
WITH NORECOVERY
RESTORE LOG Utest
FROM UtestDump
WITH FILE=2, NORECOVERY
RESTORE LOG Utest
FROM UtestDump
WITH FILE=3, RECOVERY

```

### Output:

```

Processed 80 pages for database 'Utest', file 'UtestData1' on file 1.
Processed 16 pages for database 'Utest', file 'UtestData2' on file 1.
Processed 1 pages for database 'Utest', file 'UtestLog' on file 1.
RESTORE DATABASE successfully processed 97 pages in 0.360 seconds (2.194 MB/sec).
Processed 1 pages for database 'Utest', file 'UtestLog' on file 2.
RESTORE LOG successfully processed 1 pages in 0.067 seconds (0.099 MB/sec).

```

Processed 1 pages for database 'Utest', file 'UtestLog' on file 3.  
RESTORE LOG successfully processed 1 pages in 0.075 seconds (0.027 MB/sec).

### Step 13: Check!

We need to check if all the rows were restored properly or not.

Let's see:

```
USE Utest
GO
SELECT * FROM TTable
GO
```

Output:

NID	NDESC
-----	-------

1	One
2	Two
3	Three
4	Four
5	Five
6	Six
7	Seven
8	Eight

(8 row(s) affected)

That's Amazing! Isn't it?

### Conclusion:

- The actual recovery process of the database can start with a further backup! It all depends on your backup strategy.
- Placement of Physical Database files with in file group and physical hard drives and placement of objects in the file group does matter a lot from backup/recovery perspective apart from performance tuning perspectives.

### Questions:

Please email your questions, comments, suggestions and difficulties with this article to [articles@sqlsquare.com](mailto:articles@sqlsquare.com)

### Cleanup

Drop the device and delete the file

```
sp_dropdevice 'UtestDump'
```

Output

File 'c:\SQLTest\UtestDump.bak' closed.

Device dropped.

Successfully deleted the physical file 'c:\SQLTest\UtestDump.bak'.

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

Drop the Database:

### **DROP DATABASE UTest**

Output:

Deleting database file 'c:\SQLTest\UTestData1.mdf'.

Deleting database file 'c:\SQLTest\UTestLog.ldf'.

Deleting database file 'c:\SQLTest\UtestData2.ndf'.

Delete the Folder C:\SQLTest. (Or whatever folder that you created to store the database files)

---

Send your comments and feedback to [palak@sqlsquare.com](mailto:palak@sqlsquare.com)

Author's Web Site: [www.sqlsquare.com/palak](http://www.sqlsquare.com/palak)