Recovering a SQL Server Database from Suspect Mode

Recovering a SQL Server Database from Suspect Mode

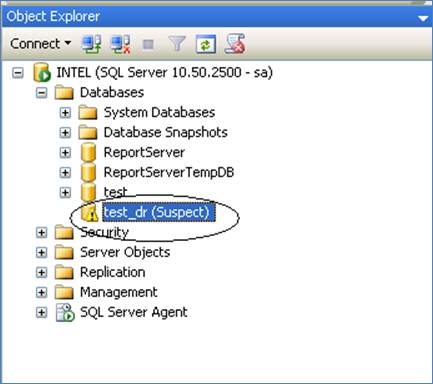
BY [SATNAM SINGH](http://www.sql-server-performance.com/author/satnamsingh/)

A couple of days back at I got a call from my support team informing me that one of our database located on the Production Server went into **Suspect Mode.**The version used was SQL Server 2005 Service Pack 3. Being a Production Database server, it was a Priority 1 incident and the expected time of resolution was 4 hours..

**Solution:**

The first step was to identify why this incident occured and after investigation it was found that it was due to the corruption of the transactional log file of the database.

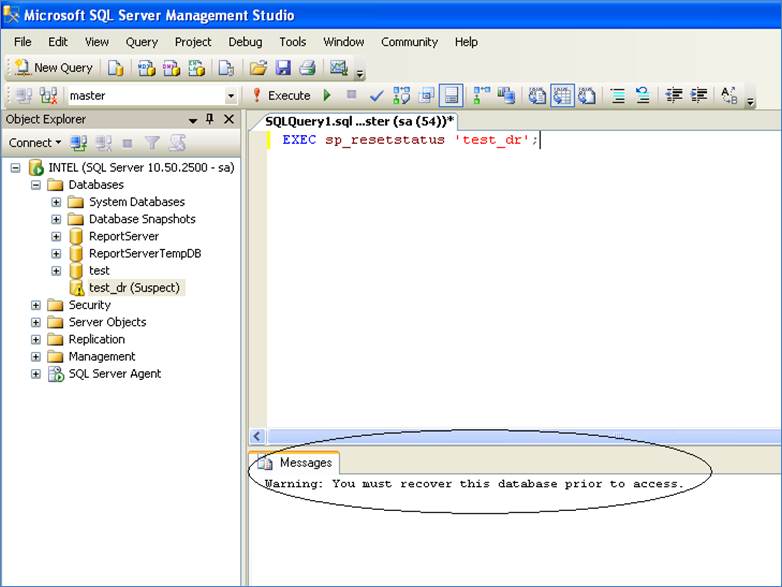
I connected to SSMS using the **sa**login credentials and located the **SUSPECT** database:



I then reset the status of the **SUSPECT** Database by executing the below T-SQL query against the master database.

EXEC sp\_resetstatus 'test\_dr';

**sp\_resetstatus** turns off the suspect flag on a database. This procedure updates the mode and status columns of the named database in sys.databases. Also note that only logins having sysadmin priveleges can perform this :



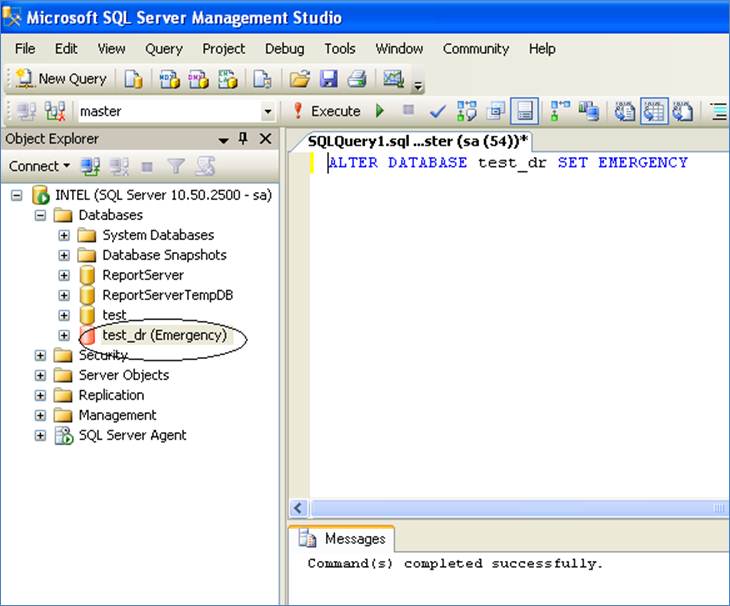
As you can see in the above screen capture, the T-SQL query gave the warning message upon execution:

**You must recover this database prior to access**

The next step was to set the SUSPECT database into an **EMERGENCY** mode. This was done by executing the below SQL query against the master database.

ALTER DATABASE test\_dr SET EMERGENCY

Once the database is set to EMERGENCY mode it becomes a **READ\_ONLY** copy and only members of**sysadmin** fixed server roles have privileges to access it. The basic purpose for this is to facilitate troubleshooting. I did not want other users updating the database while it was being worked on.

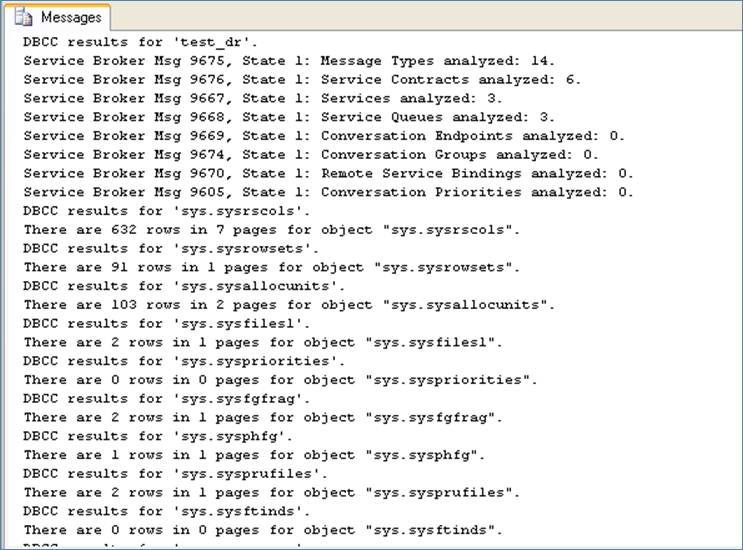


As you can see from the above screen capture, once the T-SQL query got executed successfully the state of the database changed from **SUSPECT** to **EMERGENCY**.

Once the database state was changed to **EMERGENCY**. I then performrf a consistency check by executing the below T-SQL query against the master database.

DBCC checkdb('test\_dr')

Which resulted in the below output:



As seen from the above screen capture there is no issue with respect to consistency of the **test\_dr** database. Also, this confirmed that the logical and physical integrity of the database was intact.

The next step was to set the database to **SINGLE USER** mode with **ROLLBACK IMMEDIATE**. To do this the below SQL query was executed against the master database.

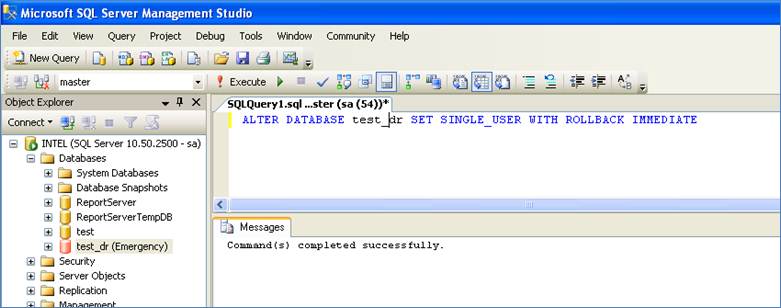
ALTER DATABASE

test\_dr SET SINGLE\_USER

WITH ROLLBACK IMMEDIATE

The above query will rollback any transactions if any are present in the **test\_dr**database and will bring the database named **test\_dr** into **Single User** mode.

Please refer to the screen capture below:



# Recovering a SQL Server Database from Suspect Mode

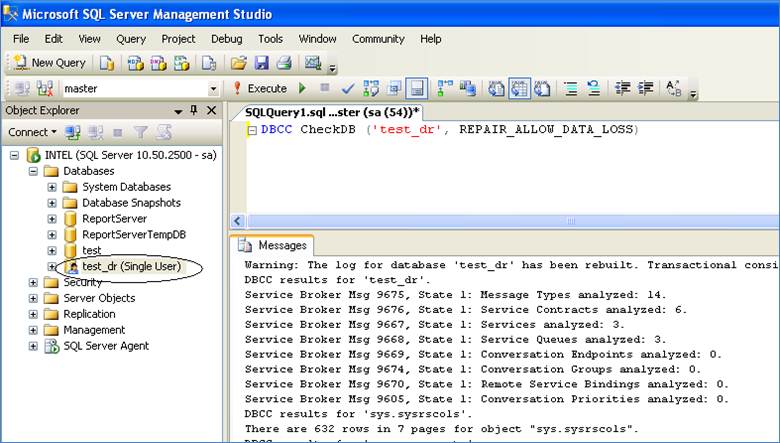
BY [SATNAM SINGH](http://www.sql-server-performance.com/author/satnamsingh/)

The next step was to perform a DBCC Checkdb along with **Repair with Data Loss** by executing the below T-SQL query against the master database.

DBCC CheckDB ('test\_dr', REPAIR\_ALLOW\_DATA\_LOSS)

This query will attempt to repair all reported errors. These repairs can cause some data loss.

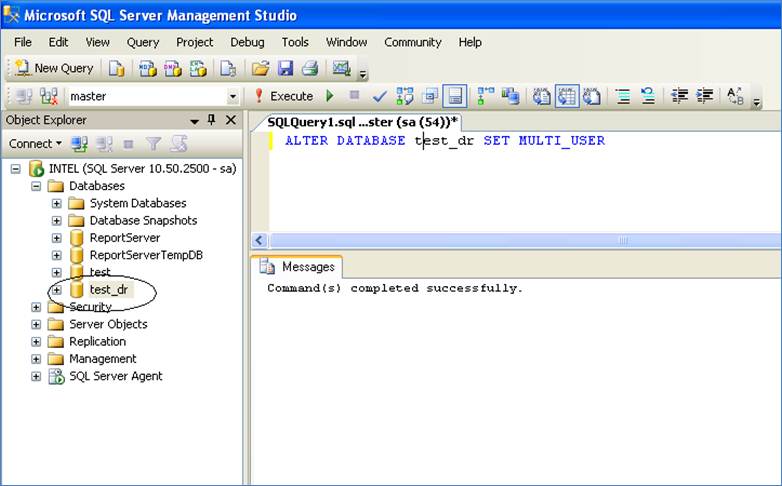
Once the DBCC CheckDB with the **Repair with Data Loss** option were executed, the Database went into Single User mode as shown below:



After performing the above step the database was brought ONLINE and Multiple Users access was enabled by executing the below T-SQL query against the master database.

ALTER DATABASE test\_dr SET MULTI\_USER

Please refer the screen capture below.

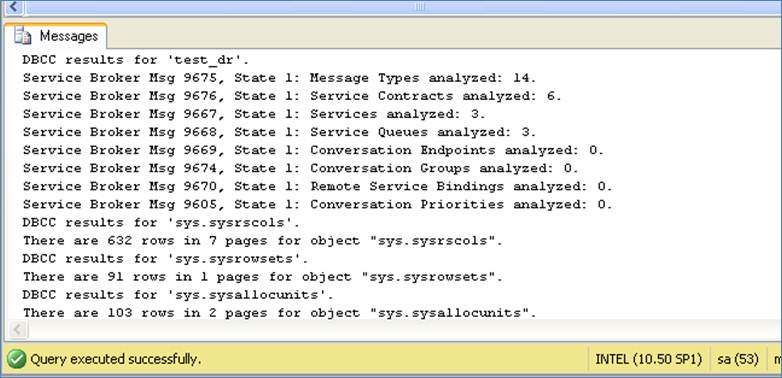


As you can see from the above screen capture the database named **test\_dr** is back ONLINE. I am even able to view its objects as shown below:



As final step for safety, I again checked the consistency of the database which was just repaired and brought ONLINE (i.e. the **test\_dr** database) by executing the below T-SQL query against the master database.

 DBCC CheckDB ('test\_dr')



After performing the above steps I ensured that all the required logins had access to the database with proper privileges. The application started working fine and the business was back on track. It took just 38 minutes to bring the SUSPECT database back ONLINE.

Please let me know if you have any comments on this approach or alternative approaches you have used in the past.