**DB Recovery - when DB State is Recovery\_pending - or Suspect**

<http://www.sqlskills.com/blogs/paul/post/Search-Engine-QA-4-Using-EMERGENCY-mode-to-access-a-RECOVERY-PENDING-or-SUSPECT-database.aspx>

<http://www.sqlskills.com/blogs/paul/post/checkdb-from-every-angle-emergency-mode-repair-the-very-very-last-resort.aspx>

By far the most common search engine query leading to the blog is about fixing a suspect or unrecovered database. The very best way to do this is to use your backups, and to have a backup strategy that allows you to recover in the smallest amount of time and with no data loss. But what if you don't have a backup for some reason? Well, it depends what's damaged in the database and when the damage is noticed.

There are three states the database can be in when its damaged:

ONLINE

* If it's one of the data files that is damaged and the damage is hit during a query or some other operation then the database will remain online and accessible.

RECOVERY PENDING

* The database will be in this state if SQL Server knows that recovery needs to be run on the database but something is preventing recovery from starting. This is different from SUSPECT because there's nothing to say that recovery is going to fail - it just hasn't started yet.
* An example of this is when the database wasn't cleanly shut down (i.e. there was at least one uncommitted transaction active at the time the database was shut down) and the log file has been deleted.

SUSPECT

* The database will be in this state if the transaction log is damaged and it prevents recovery or a transaction rollback from completing.
* Some examples of this are:
* When the database wasn't cleanly shut down and recovery tries to read a corrupt data page or comes across a corrupt log record.
* A regular transaction rolls back and tries to read a corrupt data page or comes across a corrupt log record.

You can check the state of a database in the sys.databases catalog view:

SELECT state\_desc FROM sys.databases WHERE name = 'master';

GO

or by using the DATABASEPROPERTYEX function:

SELECT DATABASEPROPERTYEX ('master', 'STATUS');

GO

Beware however, as DATABASEPROPERTYEX returns SUSPECT when the database is RECOVERY PENDING, as I'll show you below.

So the state the database is in determines what you can do if you don't have a backup. The easiest case is when it's still ONLINE. In this case you can probably run repair to remove the damage, most likely with some data loss (see my [previous post](http://www.sqlskills.com/blogs/paul/2007/09/15/CorruptionLastResortsThatPeopleTryFirst.aspx) on REPAIR\_ALLOW\_DATA\_LOSS for more details), and then take steps to prevent the damage occuring again. If repair can't fix all the errors then your only option without a backup is to extract as much data as you can into a new database.

The other two database states are more difficult and are what's causing people to search for help. In this case the database isn't accessible at all, because recovery hasn't run or completed and so the database is in an inconsistent state. It could just be logically inconsistent (e.g. a transaction modifying data hasn't recovered) or worse it could structurally inconsistent (e.g. a system transaction modifying index linkages has't recovered). Either way, SQL Server wants to prevent you from getting into the database because it doesn't know what state the data and structures in the database are in. But if you don't have a backup, you need to get into the database, no matter what state things are in.

You can do this using EMERGENCY mode. In versions prior to SQL Server 2005, EMERGENCY mode wasn't documented and you had to hack the sysdatabases table to get a database into it (worse still, the exact hack to use was changed from version to version in a bid to obfuscate things). In 2005 though, EMERGENCY mode was documented and proper syntax added to support it. Members of the sysadmin role can put the database into EMERGENCY mode using:

ALTER DATABASE foo SET EMERGENCY;

GO

Once in EMERGENCY mode, the database is accessible only by members of the sysadmin role. The database is also read-only as nothing can be written to the transaction log.

Let's see an example of this (based on a demo from my [*Secrets of Fast Detection and Recovery from Database Corruptions*](http://www.microsoft.com/emea/itsshowtime/sessionh.aspx?videoid=549) presentation). I'm going to create a database and a sample table:

IF DATABASEPROPERTY (N'emergencydemo', 'Version') > 0 DROP DATABASE emergencydemo;

GO

CREATE DATABASE emergencydemo;

GO

USE emergencydemo;

GO

CREATE TABLE salaries (

FirstName CHAR (20),

LastName CHAR (20),

Salary INT);

GO

INSERT INTO salaries VALUES ('John', 'Williamson', 10000);

INSERT INTO salaries VALUES ('Stephen', 'Brown', 12000);

INSERT INTO salaries VALUES ('Jack', 'Bauer', 10000);

GO

I'm going to start an explicit user transaction and update a row in the table:

BEGIN TRANSACTION;

GO

UPDATE salaries SET Salary = 0 WHERE LastName='Brown';

GO

Now I'm going to force the data page holding the updated row to be written to disk:

CHECKPOINT;

GO

So we have an active, uncommitted transaction that's modified the table, and the table modification has been written to disk. If the power failed at this point, crash recovery would run and the transaction would be rolled back. I'm going to simulate this by shutting down SQL Server. In another connection:

SHUTDOWN WITH NOWAIT;

GO

Server shut down by NOWAIT request from login ROADRUNNERPR\paul.

SQL Server is terminating this process.

I'm also going to simulate damage to the transaction log:

C:\Documents and Settings\paul>del "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\emergencydemo\_log.LDF"

C:\Documents and Settings\paul>

Now when I start up SQL Server again, we see the following in the error log:

2007-10-02 11:39:47.14 spid18s Starting up database 'emergencydemo'.  
2007-10-02 11:39:47.46 spid18s Error: 17207, Severity: 16, State: 1.  
2007-10-02 11:39:47.46 spid18s FileMgr::StartLogFiles: Operating system error 2(The system cannot find the file specified.) occurred while creating or opening file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\emergencydemo\_log.LDF'. Diagnose and correct the operating system error, and retry the operation.  
2007-10-02 11:39:47.60 spid18s File activation failure. The physical file name "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\emergencydemo\_log.LDF" may be incorrect.  
2007-10-02 11:39:47.60 spid18s The log cannot be rebuilt because the database was not cleanly shut down.

The database wasn't cleanly shut down and the transaction log isn't available so recovery couldn't run. The final message is interesting - there's a feature in SQL Server 2005 that if you attach or startup a database without a transaction log file, and the database was cleanly shut down, SQL Server will create a new log file automatically. In our case that can't happen though.

What happens if I try to get into the database?

USE emergencydemo;

GO

Msg 945, Level 14, State 2, Line 1

Database 'emergencydemo' cannot be opened due to inaccessible files or insufficient memory or disk space. See the SQL Server errorlog for details.

So what state is the database in?

SELECT DATABASEPROPERTYEX ('emergencydemo', 'STATUS');

GO

returns SUSPECT. But checking the sys.databases table

SELECT state\_desc FROM sys.databases WHERE name='emergencydemo';

GO

returns RECOVERY PENDING. This is what I'd expect, as recovery didn't get a chance to even start.

Now I'll set the database into EMERGENCY mode so I can get in and see what state things are in:

ALTER DATABASE emergencydemo SET EMERGENCY;

GO

In the errorlog you can tell when a database has been put into EMERGENCY mode:

2007-10-02 11:53:52.57 spid51 Setting database option EMERGENCY to ON for database emergencydemo.  
2007-10-02 11:53:52.59 spid51 Starting up database 'emergencydemo'.  
2007-10-02 11:53:52.62 spid51 The database 'emergencydemo' is marked EMERGENCY\_MODE and is in a state that does not allow recovery to be run.

Let's try that again:

USE emergencydemo;

GO

This time it works. What's the state of the data?

SELECT \* FROM salaries;

GO

FirstName LastName Salary  
-------------------- -------------------- -----------  
John Williamson 10000  
Stephen Brown 0  
Jack Bauer 10000

It's inconsistent, as I'd expect.

That's the catch with EMERGENCY mode - you can get into the database but recovery hasn't run or completed so you don't know whether the database is logically or structurally consistent. However, at least you *can* get into the database to extract data out or repair any damage.

In the next post (later today) I'll show you how to repair any damage using the emergency-mode repair feature of DBCC CHECKDB.

**SELECT state\_desc FROM sys.databases WHERE name='ax40\_kni\_drift';**

**ALTER DATABASE emergencydemo SET EMERGENCY;**

**GO**

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

**GO**

**DBCC CHECKDB (emergencydemo, REPAIR\_ALLOW\_DATA\_LOSS) WITH NO\_INFOMSGS;**

**GO**

**if you get "Database 'db\_name' is being recovered. Waiting until recovery is finished. "**

**Follow the following steps:**

**1 stop sql server**

**2 rename your db files or move them to another location**

**3 start sql server**

**4 remove the database from the list**

**5 create a new database with the same name and the same datafiles**

**6 set this database in emergency mode**

**7 stop sql server**

**8 copy your original db files back**

**9 start sql server.**

**Your original db is now in emergency mode.**