Restoring Microsoft SQL Server 7 Master Databases

https://campus.barracuda.com/doc/43843639/

A damaged master database is evident by the failure of the SQL Server to start, by segmentation faults or input/output errors or by a report from DBCC. An example of an error might be damage caused by media failure in the area in which master database is stored.

The procedure used to recover a damaged master database is different from the procedure used to recover user databases. If the master database becomes unusable, it must be restored from a previous dump. All changes made to the master database after the last dump are lost when the dump is reloaded and therefore must be reapplied.

It is strongly recommended that the master database be backed up each time it is changed. This is best accomplished by prohibiting the creation of user-defined objects in the master database and by being aware of the statements and system procedures, and the equivalent actions in SQL Enterprise Manager, that modify it.

The most common statements and system procedures that modify master are:

- **DISK INIT**
- **CREATE DATABASE**
- **ALTER DATABASE**
- **DISK MIRROR**
- **DISK UNMIRROR**
- **DISK REMIRROR**
- **sp_dropremotelogin**
- **sp_addumpdevice**
- **sp_dropdevice**
- **sp_addlogin**
- **sp_droplogin**
- **sp_addserver**
- **sp_dropserver**
- **sp_addremotelogin**

If a user database is created, expanded, or shrunk after the most recent dump (backup) of the master database and if it becomes necessary to reload the master database, then that user database and all data in it will be lost and must be restored from backup. Because of this, always dump (back up) the master database after creating, expanding, or shrinking user databases.

You must rebuild using the same character set and sort order as the master database dump that will be reloaded.

Complete the following steps to recover a damaged master database:

Step 1. Rebuild Master Database

1. In Windows Explorer, go to **Start** > **Programs** > **Microsoft SQL Server** , and then click **SQL Setup** , or, from the distribution media, navigate to the directory containing the software compatible with your hardware platform's processor architecture, and run **SETUP.EXE**.
2. Follow the on-screen instructions until the **Options** window displays.
3. Select **Rebuild Master Database**, and then click **Continue** . A confirmation window displays.
4. Click **Resume** ; the **Rebuild Options** window displays.
5. To specify the character set, click **Sets** and make any necessary changes in the **Select Character Set** window; skip this step if you are using the default character set (ISO 8859-1).

   You must use the same character set and sort order previously used for this master database.
6. To specify the sort order, click **Orders** and make any necessary changes in the **Select Sort Order** window; skip this step if you are using the default sort order (dictionary order, case-insensitive).
7. In the **Rebuild Options** window, click **Continue** . The **SQL Server Installation Path** window displays.
8. If not correctly displayed in the **SQL Server Installation Path** window, enter the location of the existing SQL Server installation, and then click **Continue** .
9. The **Rebuild Master Database** window displays. If it is not correctly displayed in the **Rebuild Master Database** window, enter the location, name, and device of the existing MASTER device, and then click **Continue** .
10. The setup program rebuilds the master database. When rebuilding is complete and the completion window displays, click **Exit** .

The files `MASTER.DA@` and `MASTER.AL@` are stored in the `\MSSQL\INSTALL` directory. When rebuilding the master database (or when installing SQL Server), one of these two files is used by the setup program. When the default sort order and character set are selected, `MASTER.DA@` is expanded and copied onto the server, replacing `MASTER.DAT`. When an alternate character set and/or sort order is selected, `MASTER.AL@` is expanded, copied onto the server, and several SQL scripts are run.

Step 2. Restart Microsoft SQL Server in Single-User Mode

Before you can restore the master database, you must start Microsoft SQL Server in single-user mode.

1. Go to the **SQL Server Manage** , right-click the SQL server instance, and click **Properties** ; the

SQL Server Properties window displays.

2. In the **General** tab, open the **Startup Parameters** window, and in the **Parameter** field, type: `—m`
3. Click **Add** , and then click **OK** . Click **OK** to close the **SQL Server Properties** window .
4. Right-click the SQL server instance, and click **Stop** , and then right-click the SQL server instance, and click **Start** to restart the instance.

> You may find it convenient to start the SQL Server in single-user mode using the command line program, **SQLSERVER.EXE** , with option "/m". This procedure only works, however, if the SQL Server is configured to start using the current interactive user’s account.

Step 3. Restore Master Database from Most Recent Backup

1. Create a restore job and select the most recent backup version of the master database.
2. Run the restore job.

> This may take some time, typically 10 to 15 minutes, depending on the size of the master database. Restore only the master database while in single user mode. Do not restore any other databases.

Step 4. Apply Changes to Master Database

1. Go to the SQL Server Enterprise Manger and right-click the SQL server instance. Select Properties to open the SQL Server Properties window.
2. Under the **General** tab in the SQL Server Properties window, open the Startup Parameters window and remove "–m" from the list of existing parameters.
3. Right-click the SQL server instance, and click **Stop** , and then right-click the SQL server instance and click **Start** to restart the instance. If there have been no changes to the master database since the last dump, then proceed to *Step 5. Drop Invalid Databases and Database Devices.*
4. If login IDs or devices have been added to or dropped from the master database since the last backup, those changes must be reapplied. Restart the server and reapply the changes manually or from saved batch files.
5. If databases have been created, expanded or shrunk since the last dump of master, those databases must be dropped and then restored.
6. If you have made many changes and have no recent dump, it is possible that by reloading master in some cases you can regain data in user databases that has been lost. This technique requires the use of **DISK REINIT** and **DISK REFIT** and can involve manual modifications to the master database tables.

- Use **DISK REINIT** to recreate rows in sysdevices for all database devices that have been added after the most recent dump. **DISK REINIT** updates sysdevices just as **DISK INIT** does, but it does not format the physical disk file, so existing data is preserved.
- Use **DISK REFIT** to recreate rows in sysusages and sysdatabases for all **CREATE** and **ALTER DATABASE** statements that were performed after the most recent dump.
- **DISK REFIT** scans the physical file associated with each space that is allocated to databases. It also adds the corresponding sysdatabases entries. Some of the information is not reconstructed perfectly.
  For example, the original virtual device number is not assigned, because it is not known. Instead, virtual device numbers are assigned sequentially. The database owner is not extracted while scanning the physical files; ownership is set to the system administrator. It is also not possible to determine how many sysusages entries originally existed. **DISK REFIT** inserts a separate entry for each different segment type.
- When this is done, correct the entries made by **DISK REFIT** to sysdatabases and sysusages (if desired) and also add to syslogins any login IDs that were not retained. Then shut down and restart SQL Server.

  Capturing the latest changes made to a database by using **DISK REFIT** and **DISK REINIT** to recreate the master database is possible, but it is preferable to keep the master database current by dumping it after creating or altering databases. Using **DISK REFIT** and **DISK REINIT** is a complicated process that can result in data loss because many of the changes made to a database often must be reconstructed manually in the master database. If you feel this technique is necessary, contact your primary support provider before beginning the recovery process.

Step 5. Drop Invalid Databases and Database Devices

Use the SQL Enterprise manager to drop any invalid database devices and databases from the newly restored master database.

If you are recovering from a disaster where you have lost a database device file, the master database you have just restored still contains a reference to it. Yosemite Server Backup will not be able to restore any databases contained on the database device until the file is restored or the database device is dropped. If the database device is dropped, Yosemite Server Backup will automatically recreate the device when a database contained on the device is restored.

Step 6. Restore msdb Database

When restoring the msdb database, keep the following considerations in mind:

- The msdb database supports SQL Executive and provides a storage area for scheduling information. The schedules that you implement using SQL Enterprise Manager are maintained in the msdb database. This includes such things as the tasks that you schedule from the Task Scheduling window, the automatic backups you schedule from the Database Backup/Restore window and all replication tasks, which are automatically created by the system if the server is configured as a replication distributor.
- During installation of a server, the setup program automatically creates two devices (of 2MB and 1MB) on the same disk drive as the master database and then places the msdb database on the 2MB device (MSDBDATA) and its transaction log on the 1MB device (MSDBLOG). Scheduling information is then stored in this database.
- During a rebuild of the master database, the setup program drops and recreates the msdb database, which results in a loss of all scheduling information.