Proper Care and Feeding of your SQL MDB

(Recommendations for General MDB Maintenance)

A well thought-out MDB *installation* is only the beginning. There are regular maintenance tasks that must be performed – including TEMPDB and logging considerations - as well as recommended backup/recovery tasks.

This document reviews those tasks (and provides recommendations for them) and also highlights performance aspects that should be considered. It was last revised May 15, 2006.

Maintenance Considerations

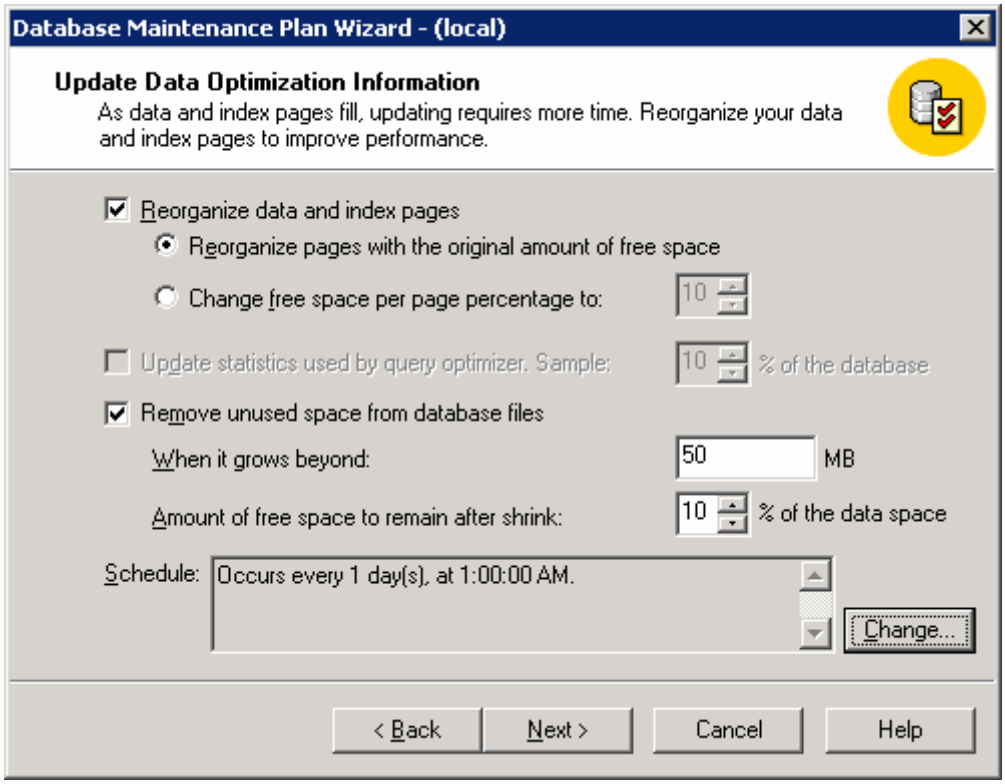
As part of your day to day maintenance tasks you should carefully monitor the following to ensure the MDB server's capacity is not exceeded or its performance compromised:

- SQL memory config/usage
- Indexes
- Data files
- Transaction logs

Of course, you should also regularly (and frequently) backup critical files.

Monitor Indexes

Fragmented indexes should be rebuilt using SQL Server facilities such as Database Maintenance Planner.



This dialog includes the following options:

- **Reorganize data and index pages -** Reorganizing data and index pages can reestablish the free space.
- **Reorganize pages with the original amount of free space -** Causes the indexes on the tables in the database to be dropped and re-created with the original FILLFACTOR that was specified when the indexes were created.
- **Remove unused space from database files -** Remove any unused space from the database, thereby allowing the size of the data files to be reduced.
- **When it grows beyond** - Remove unused space from the database only if the database exceeds the specified size, in MB.
- **Amount of free space to remain after shrink** - Determine the amount of unused space to remain in the database after the database is shrunk (the larger the percentage, the less the database can shrink, and the less it'll expand - avoiding performance hits).
- **Schedule** - Set the frequency that the data optimization tasks.

Some products provide specific database index maintenance tools. Unicenter Desktop and Server Management, for example, provides the dsmmssqlopt script that monitors state for performance problems and conducts a two level defragmentation and index rebuild strategy. Run this script anytime report or UI performance deteriorates after adding or updating many machines.

Note: It does not perform maintenance on tables/indices unless needed.

Monitor Data Files and Disk Space

Operating system data files should be defragmented as often as necessary using system tools such as Disk Defragmenter. Disk space should be kept at least 20% free.

Monitor Transaction Logs

For best performance you should increase the initial size of the transaction log file based on estimated usage.

Using a specific growth increment amount, such as 100MB, instead of a percentage increment may be beneficial. Automatically growing transaction logs does cause some performance degradation, therefore you should select a reasonable size for the autogrow increment to avoid automatic growing too often. At most, automatic growing should only arise once per week. This also applies to the MDB database.

Manually shrink the transaction log files based on monitoring observations and defragment the transaction log file as necessary.

Regularly Backup Critical Files

The following files should be regularly backed up to assist in recovery procedures:

- Database and transaction log files should be scheduled for regular backups
- The master database should be included in the backup plan

Multiple simultaneous backup devices may be used for improved performance.

Further backup and recovery guidelines are provided later in this document.

Note: For More Information, see Microsoft SQL Server Books Online – Database Maintenance Plan Wizard, `DBCC SHOWCONTIG`, and `DBCC_INDEXDEFRAG`.

Performance Considerations

Depending on the number of processors, memory, and disks, installing Microsoft SQL Server on a dedicated server is not required. Network latency between the application and a separate SQL server may, in fact, degrade application performance.

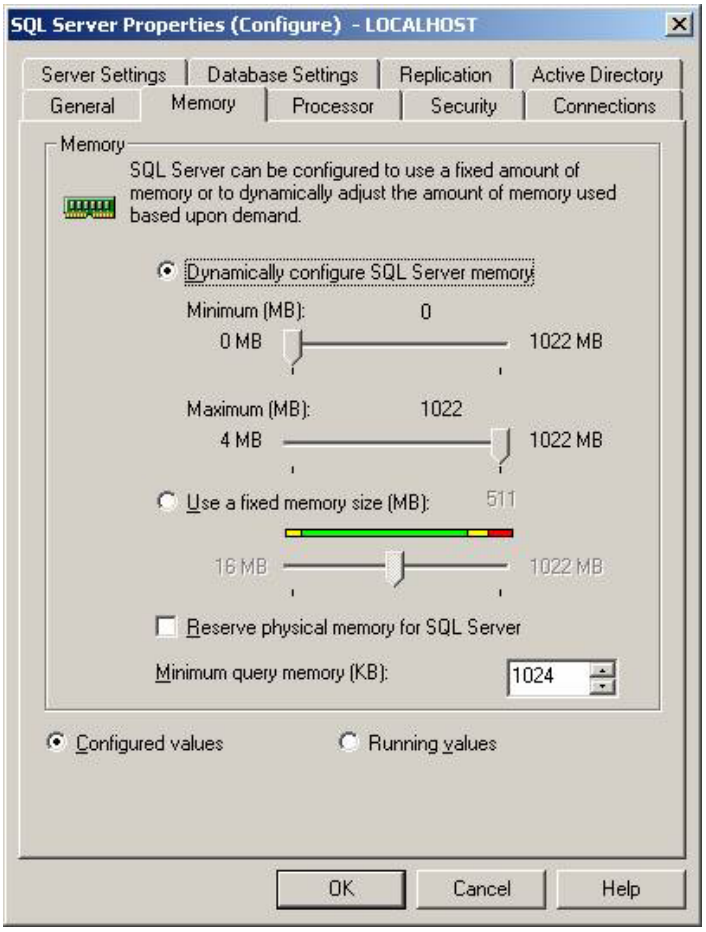
It is best practice to always run SQL Server on the same box as a mid-tier manager (for example a Unicenter Desktop and Server Management Domain Manager should run on the same box as its MDB).

For the enterprise tier you may wish to run the MDB on a dedicated SQL Server box – however, this decision depends upon the number of enterprise components and the peak load.

An Enterprise MDB may coexist with Enterprise Managers but, for large clients, this may not be feasible. DBAs tend to prefer dedicated database hardware, but with SQL 2005 and modern hardware this is not required. It is relatively easy to allocate hardware to SQL and a shared box for Enterprise Manager software and the MDB may be the best choice (it often simplifies configuration and administration of fault tolerant environments). However, where a number of large enterprise managers are involved the best choice for over all performance and ease of management may be a separate dedicated SQL server. Caution: it is generally unwise to consolidate an r11 MDB on the same SQL server as used for other databases, especially application databases.

Manage Memory

You should set reserved physical memory for SQL server and specify an amount – on a 4GB box we suggest reserving at least 2GB (monitor memory usage to determine if more memory can be reserved for SQL)

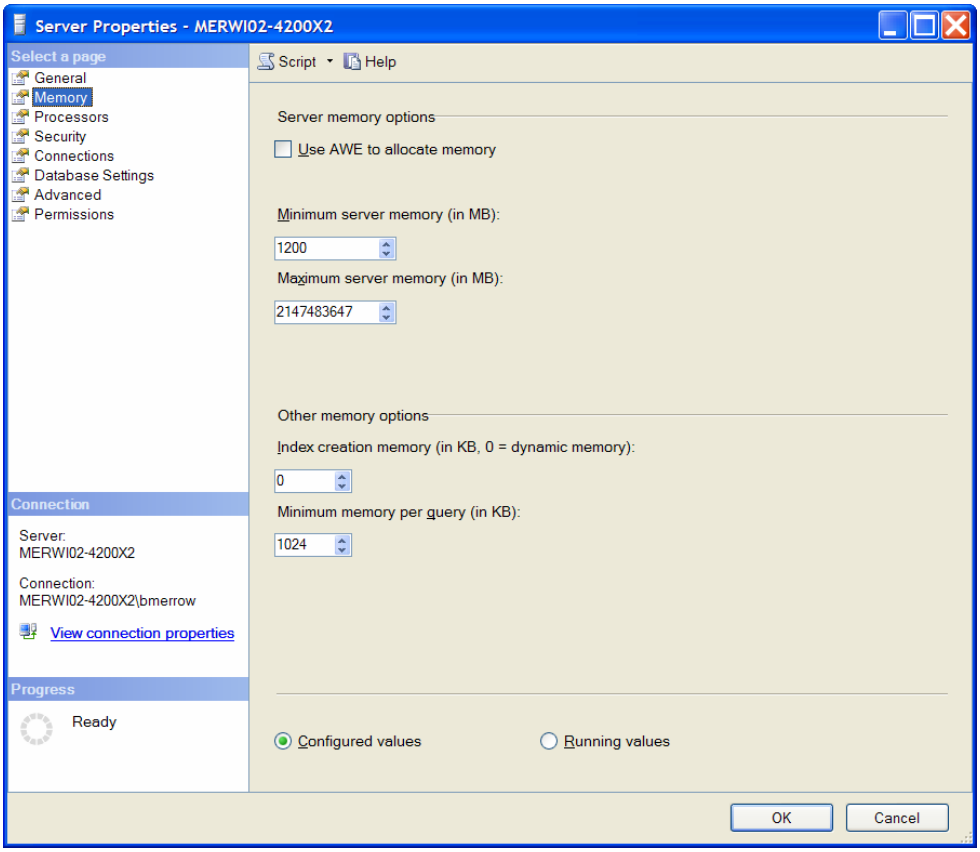


As highlighted above, our best practice is to run middle tier managers on the same box as their MDB. In general, but especially in such cases, it is best to reserve memory for SQL Server use.

If you do not reserve memory for SQL, Windows will swap out SQL pages that could be readily available in memory – resulting in SQL performance swings.

Since reserving memory for SQL takes memory away from the middle tier manager you should monitor memory use and paging. But, in general, SQL Server always benefits from reserved memory.

You should set the minimum memory for SQL 2005 – on a 4GB box we suggest reserving 2GB (monitor memory usage to determine if more memory can be reserved for SQL).



SQL Server Enterprise Editions can use more than 2GB of memory so if you have more than 2GB of memory available for SQL you may wish to enable AWE (refer to http://support.microsoft.com/kb/274750/en-us for details).

Performance Considerations for Storing Data

Store the MDB data and log on separate disk drives for improved performance (keeping the log on a separate disk can be a significant tuning benefit). The same applies to TEMPDB.

Maintaining SQL data on striped disks also provides a major tuning benefit – if you have limited drives available and a choice between separate log/data disks or one striped drive the striped drive is usually the better option.

More about TEMPDB

SQL Server uses the TEMPDB database as a scratch area for MDB temporary tables, sorting, subqueries, and so forth. TEMPDB should be stored on its own drive away from other databases whenever possible (default is SQL install disk, so it is common to add a new separate tempdb on a different drive).

The size of the TEMPDB database should be increased based on the amount of available disk space and expected usage. If TEMPDB often grows to the same size, you may want to leave it that way as performance is hurt if it has to grow constantly.

The big question here is how big the MDB is and how big is its biggest table (that should be the guide for the smallest TEMPDB can be for safety – i.e. "`order by`" clauses, etc). TEMPDB file growth is expensive each time it occurs. At the very least, you should set the file growth increment percentage to a reasonable size to prevent TEMPDB files from growing by too small a value.

Logging Considerations

Be sure to install SQL Server on a disk with sufficient available space – this is the default log location. SQL Server allows transaction log files to exist on multiple devices – this improves logging system performance by allowing it to write to multiple disk devices.

MDB Transaction Log automatically grows by 10% and its growth is unrestricted, but using a specific growth increment amount, such as 100MB, instead of a percentage increment may be beneficial.

In some situations the Transaction Log may become very large, leading to the following situations:

- Run out of space
- Transactions may fail and may start to roll back
- Transactions may take a long time to complete

When that happens, shrink it with the following:

`DBCC SHRINKFILE`: `DBCC SHRINKFILE(mdb_log, TRUNCATEONLY)`.

The TRUNCATEONLY option causes any unused space in the files to be released to the operating system and shrinks the file to the last allocated extent, reducing the file size without moving any data.

For recovery purposes, immediately execute `BACKUP DATABASE`.

IMPORTANT: shrinking the transaction log every day may impact the performance of your database due to fragmentation!

`BACKUP LOG MDB WITH TRUNCATE_ONLY` could also be used instead of `DBCC SHRINKFILE`. `TRUNCATE_ONLY` removes the inactive part of the log without making a backup copy of it and truncates the log, thereby freeing up space. Specifying a backup device is unnecessary because the log backup is not saved. **For recovery purposes, immediately execute `BACKUP DATABASE`.**

If you are having difficulty controlling log growth, the best thing that you can do is increase the frequency of your log backups. Log backups cause inactive portions to be moved to the END of the log file. That won't cost you much more disk space for the backups, and will allow your transaction log to stay much, much smaller because you are just splitting up those transactions among more backup files. You will find that when you arrive at that “magic” log backup frequency, your log file will stabilize at a size that you are satisfied with and will quit expanding.

When to Backup

Using a combination of database, differential database, and transaction log backups minimizes the amount of time needed to recover from a failure.

Differential database backups reduce the amount of transaction log that must be applied to recover the database. This is normally faster than creating a full database backup. The MDB uses full recovery model. Suggested backup plan:

- Use Database, Differential and Transaction
- A full backup should be created at least once a day
- Transaction log backups should occur every hour
- Differential backups should occur every three hours
- The full backup should occur during off-hours when there is minimal database use
- The transaction and differential backups should be on a set schedule based on when your full backup occurs

Backup and Recovery Options

MDB Full Backup

For a full MDB backup, do the following:

```
BACKUP DATABASE DB_NAME TO BACKUP_DEVICE
    USE MASTER
    EXEC SP_DROPDEVICE 'MDB_BKP1'
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP1', 'C:\PROGRAM FILES\MICROSOFT
    SQL SERVER\MSSQL\BACKUP\MDB_BKP1.BAK'
```

```
USE MDB
BACKUP DATABASE MDB TO MDB_BKP1
```

`EXEC SP_DROPDEVICE 'MDB_BKP1'`: Drops a backup device, deleting the entry from **`master.dbo.sysdevices`**.

`EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP1', 'C:\PROGRAM FILES\MICROSOFT SQL SERVER\MSSQL\BACKUP\MDB_BKP1.BAK'`: adds a backup device (disk file as backup device in this case) to SQL Server.

Valid device types are:

- **disk -** Hard disk file as a backup device.
- **pipe -** Named pipe - If you are **backing up your MDB directly over a network** (not really recommended as it hurts performance), one way to boost throughput is to perform the backup over a dedicated network which is devoted to backups and restores only. All devices should be on the same high speed switch. Avoid going over a router when backing up over a network, as they can greatly slow down backup speed.
- **Tape -** Any tape devices supported by Windows. If device is **tape**, **noskip** is the default.

For fastest backups, perform a disk backup to a local drive array (ideally, to an array dedicated to backups only), then move the backup file(s) over the network to another server where the file can be stored, or to a tape device (this also ensures you will be able to recover in case of hardware failure). Backing up a database directly to tape on a local device, or to a tape device over the network, or to directly to a hard disk over the network, is generally slower. However, there's nothing wrong with keeping one copy of MDB backup on the local server (even though it has been moved to another disk/tape or off the server) should you need to restore the backup quickly.

`BACKUP DATABASE MDB TO MDB_BKP`: Backs up the entire MDB database.

Differential Backup

For a differential backup, do **either** of the following:

- BACKUP DATABASE MDB TO MDB_BKP1 WITH DIFFERENTIAL
- BACKUP DATABASE MDB TO DISK = 'C:\PROGRAM FILES\MICROSOFT SQL SERVER\MSSQL\BACKUP\ MDB_LOG_BKP3.BAK' WITH DIFFERENTIAL

The `WITH DIFFERENTIAL` parameter specifies that the database or file backup should consist only of the portions of the database or file changed since the last full backup. A differential backup usually takes up less space than a full backup. If you use this option all individual log backups since the last full backup do not need to be applied.

Notice that in the first differential backup example, we used the same backup device (i.e. MDB_BKP1). Keep reading for more on RESTORE DATABASE DB_NAME FROM BACKUP_DEVICE WITH FILE = FILE_NUMBER.

In the second differential backup example, we're not defining a logical backup device. We are using the TO DISK=PHYICAL_BACKUP_DEVICE_NAME syntax.

Using Multiple Backup Devices in Parallel

When dealing with a large MDB, the backup process can be optimized by using multiple backup devices in parallel:

```
USE MASTER
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP2', 'C:\PROGRAM FILES\MICROSOFT
    SQL SERVER\MSSQL\BACKUP\MDBBKP2.BAK'
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP3', 'C:\PROGRAM FILES\MICROSOFT
    SQL SERVER\MSSQL\BACKUP\MDBBKP3.BAK'
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP4', 'C:\PROGRAM FILES\MICROSOFT
    SQL SERVER\MSSQL\BACKUP\MDBBKP4.BAK'
    USE MDB
    BACKUP DATABASE MDB TO MDB_BKP2, MDB_BKP3, MDB_BKP4
```

Using multiple backup devices allows backups to be written to all devices in parallel. Similarly, the backup can be *restored* from multiple devices in parallel.

Backup device speed is one potential bottleneck in backup throughput. Using multiple devices can increase throughput in proportion to the number of devices used.

IMPORTANT: When using multiple backup devices, they must be all of the same type (i.e. disk)!

Transaction Log Backup

To backup the transaction log, do the following:

```
BACKUP LOG DB_NAME TO BACKUP_DEVICE
    USE MASTER
    EXEC SP_DROPDEVICE 'MDB_LOG_BKP1'
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_LOG_BKP1', 'C:\PROGRAM
    FILES\MICROSOFT SQL SERVER\MSSQL\BACKUP\MDB_LOG_BKP1.BAK'
    USE MDB
    BACKUP LOG MDB TO MDB_LOG_BKP1
```

MDB Restore

TO restore the MDB, do the following:

```
RESTORE DATABASE DB_NAME FROM BACKUP_DEVICE
```

```
RESTORE DATABASE MDB FROM MDB_BKP1
```

This example demonstrates restoration of a full database backup followed by a differential backup. A second backup set is also restored on the media. The differential backup was appended to the backup device that contains the full database backup:

- RESTORE DATABASE MDB FROM MDB_BKP1 WITH NORECOVERY
- RESTORE DATABASE MDB FROM MDB_BKP1 WITH FILE = 2

The NORECOVERY option instructs the restore operation to not roll back any uncommitted transactions. Either the NORECOVERY or STANDBY option must be specified if another transaction log has to be applied. If neither NORECOVERY, RECOVERY, or STANDBY is specified, RECOVERY (instructs the restore operation to roll back any uncommitted transactions) is the default –

SQL Server requires that the WITH NORECOVERY option be used on all but the final RESTORE statement when restoring a database backup and multiple transaction logs, or when multiple RESTORE statements are needed (for example, a full database backup followed by a differential database backup).

Note: When specifying the NORECOVERY option, the database is not usable in this intermediate, nonrecovered state.

The FILE option identifies the backup set to be restored. For example, a file number of “2” indicates the second backup set.

During the restore, the specified database must not be in use.

Transaction Log Restore

To restore the transaction log, do the following:

```
RESTORE LOG DB_NAME FROM BACKUP_DEVICE
    RESTORE LOG MDB FROM MDB_LOG_BKP1
```

This example restores a full database backup followed by a differential backup.

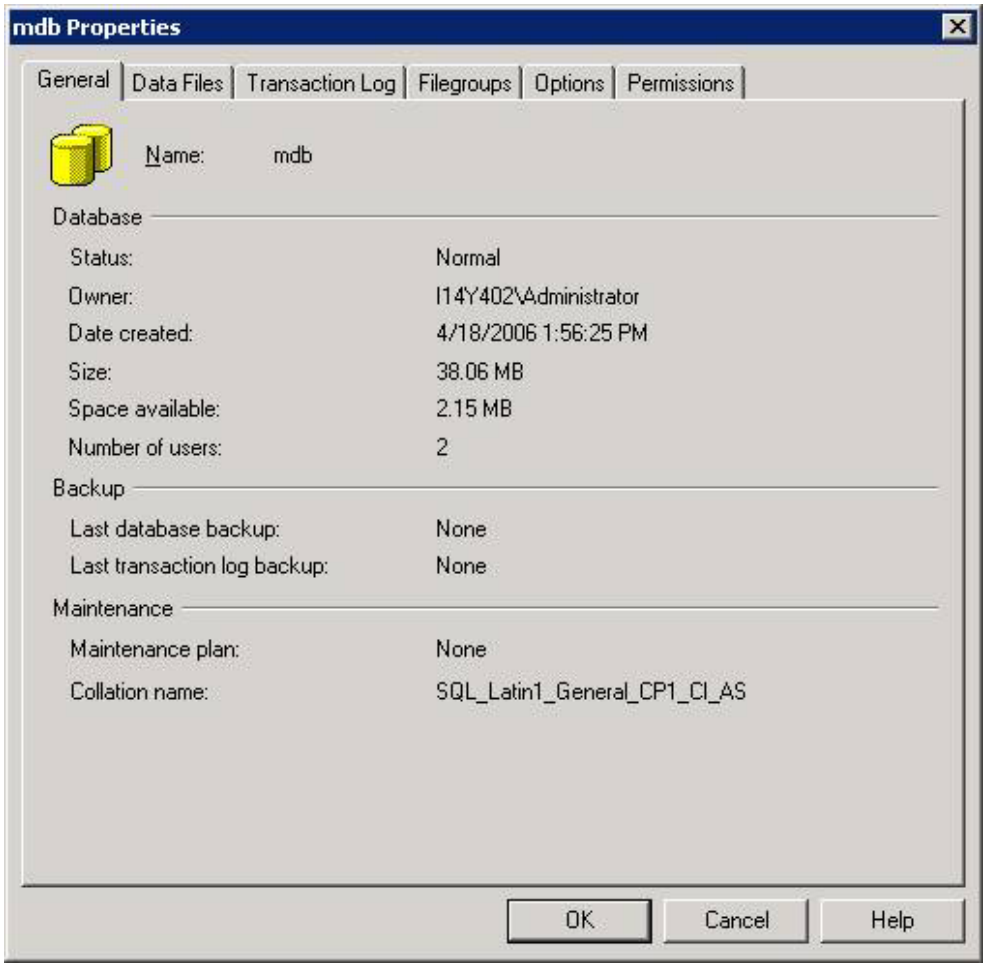
In addition, this example includes restoration of the second backup/log set on the media. The differential and second log backups were appended to the backup devices that contain the full database/first log backups:

```
RESTORE DATABASE MDB FROM MDB_BKP1 WITH NORECOVERY
    RESTORE DATABASE MDB FROM MDB_BKP1 WITH NORECOVERY, FILE=2
    RESTORE LOG MDB FROM MDB_LOG_BKP1 WITH FILE=2, RECOVERY
```

Restoring MDB Backups to Different SQL Servers

In order to restore the MDB from a backup do the following:

1. Create a database called mdb. (this value must be case insensitive, accent sensitive – i.e. …_CI_AS.)



   If your Server is using a different collation and the DB got created with a different collation name, please, run the following SQL statement in SQL Query Analyzer:

   `ALTER DATABASE MDB COLLATE SQL_Latin1_General_CP1_CI_AS.`

   Note that "`SQL_Latin1_General_CP1_`" can be replaced by the Server's default collation.

2. Launch SQL Query Analyzer and restore the MDB by running the following statements:

```
USE MASTER
    EXEC SP_DROPDEVICE 'MDB_BKP1' -- >i.e. just in case there is dump
    device called MDB_BKP1
    EXEC SP_ADDUMPDEVICE 'DISK', 'MDB_BKP1', 'Backup file path' --
    i.e. 'C:\PROGRAM FILES\MICROSOFT SQL
    SERVER\MSSQL\BACKUP\MDB_BKP1.BAK'
    RESTORE DATABASE MDB FROM MDB_BKP1
```

IMPORTANT: The RESTORE statement always produces a database that is identical to the one that was backed up. The backup file contains information on the name, number, size, and location of all files for the database at the time the backup was created. Therefore, if your location is different from the default one (e.g., i.e. C:\PROGRAM FILES\MICROSOFT SQL SERVER\MSSQL\DATA), you'll have to use the WITH MOVE clause. For example:

```
RESTORE DATABASE MDB FROM MDB_BKP1 WITH MOVE 'mdb' TO 'Your custom
location – i.e. D:\MSSQL\DATA\MDB.MDF', MOVE 'mdb_log' TO 'Your custom
location – i.e. D:\MSSQL\DATA\MDB_LOG.LDF'
```