Best Practices for Storing the Esri® Production Mapping Workspace in an Enterprise Geodatabase for SQL Server

An Esri® Technical Paper
September 2013

380 New York Street
Redlands, California 92373-8100 USA

800 447 9778
info@esri.com

esri.com

Best Practices for Storing the Esri Production Mapping Workspace in an Enterprise Geodatabase for SQL Server

An Esri Technical Paper

Contents **Page**

Contents **Page**

Best Practices for Storing the Esri Production Mapping Workspace in an Enterprise Geodatabase for SQL Server

Introduction

Esri® Production Mapping, an extension to ArcGIS® 10.1 for Desktop, streamlines your geographic information system (GIS) data and map production. It provides tools that facilitate data creation, maintenance, and validation as well as tools for producing high-quality cartographic products. Each organization has workflows that are unique to the type of data it collects and the products it delivers. These workflows can be generalized into a basic production workflow consisting of the steps used to create the geodatabase, capture or load the initial data, perform edits to the data, ensure data validity and accuracy, and produce digital or hard-copy output. Production Mapping can streamline each step while remaining flexible enough to adapt to your organization's business rules and workflows. This technical paper offers best practices for database administrators when establishing the Production Mapping workspace in an enterprise geodatabase for Microsoft SQL Server®. The enterprise geodatabase uses ArcSDE® technology as the gateway between GIS clients and SQL Server.

The Geodatabase

The geodatabase is central to any production workflow. Your geodatabase design is determined by the data that will be captured and edited, the logic rules used for validating data, and the types of output to be created.

When designing a geodatabase, certain types of validation are built in, such as the ability to limit the valid attribute values for a field by using a domain. You can also model the geometric relationships of features through topologies or networks. Production Mapping supports these forms of validation and allows you to define additional business rules using ArcGIS Data Reviewer checks stored in a batch job. The batch jobs can be run when you update the attributes of a feature or template, like domains, or as a postprocess such as validating the topology.

The product library in Production Mapping provides a framework for managing business rules, data, and map documents in a secure, centralized location. By leveraging the rules stored in the product library, data editing tools are tailored to ensure that features conform to your data collection standards. During cartographic production, the product library can act as a management system for your map documents and data, allowing you to check in and check out files and restore historical versions. The cartographic tools in Production Mapping can also be used to ensure consistent and repeatable symbology as well as provide a number of dynamic surround elements.

See the white paper *Best Practices for Storing the Product Library Workspace in an Enterprise Geodatabase for SQL Server*.

Production Database

A production database contains the data used for production tasks such as creating and updating features. Depending on the data model you are using, data in a production database can be used to create a digital or hard-copy map/chart or a specific type of data.

The data in a production database usually corresponds with a data model and product class in the product library. This technical paper is intended to help you establish the production database in an enterprise geodatabase for SQL Server.

Capture

The purpose of data capture is to consume existing data or create new data in your geodatabase. Data can be captured in many different ways, such as extracting new data from imagery, gathering information in the field with a mobile device, or converting existing data.

Production Mapping provides data loading automation tools that allow you to define the mapping ahead of time between source data, such as shapefiles or coverages, and your geodatabase to ensure consistency when converting large amounts of data. Feature Manager allows you to quickly and easily create new features by using feature templates and construction tools within a centralized editing environment that leverages your enhanced validation rules.

Editing

The editing stage of the production workflow involves adjusting existing features and adding new features to ensure that your data is up-to-date and accurate. This can involve updating data to match a new source or modifying attributes of data that was collected for a different purpose.

When editing data within Production Mapping, feature attribution is managed through Feature Manager, which allows you to update fields while ensuring the attributes are valid in accordance with your validation rules as well as update attributes in batches and create new features. Production Mapping also provides a number of editing tools for batch geometry updates, as well as tools for specific types of data or industries, including tools for linear referencing, utilities, contour lines, and z-enabled data.

Review

Data review is important to ensure that the data being created is accurate and appropriate for its intended purpose before it is delivered or used for making a map product. The data review or quality control stage of the production workflow often involves three phases: finding issues with the data, whether through manual or automated methods; fixing issues or marking them as exceptions; and verifying that the issues are resolved.

The Data Reviewer component of Production Mapping provides the ability to track records of all issues found in the Reviewer table, which maintains the status of the record as well as a link between the record and the feature with the issue. Data Reviewer offers the ability to automate data validation through configurable checks that can be run when you choose, scheduled through a Microsoft® Windows® service to run at specified times, or run through Python scripts. Automated checks may not be able to find all problems with the data, such as missing features. Data Reviewer also has a number of tools to help with manual or visual quality control, including tools to flag missing features, and the ability to create a sample set of data for in-depth validation.

Note: Data Reviewer is available with Production Mapping or as a stand-alone extension.

See the white paper *Best Practices for Storing the ArcGIS Data Reviewer Workspace in an Enterprise Geodatabase for SQL Server*.

Create Output

Typically, the final stage of a production workflow is the creation of the output that will be delivered. This stage may be repeated if delivery is contingent on approval or if the product requires regular updates. Many types of output may be produced: hard-copy maps or data exported to a certain format or served over the web.

Many of the hard-copy products produced—from one-off maps to map books to highly detailed charts—require version tracking. Production Mapping provides check-in and checkout capabilities for map documents as well as the ability to track history and roll back to a previous version.

Data visualization and symbology are important when producing a hard-copy product or serving data over the web. Production Mapping provides tools for consistent, repeatable, rule-based symbology, where you define what symbol or representation should be applied to features based on their attribute combination. Production Mapping also provides a number of custom surround elements, such as the graphic table element, which allows you to create a table or legend that automatically updates based on the data being displayed.

Workflow Management

When managing production, it is important to be able to allocate resources and track the status of the project. Being able to ensure that work is being done consistently and steps are not being skipped is also essential. Production Mapping allows you to tie all the components of data capture, editing, validation, and cartography together in high-level workflows with ArcGIS Workflow Manager and in detailed workflows with Task Assistant Manager.

Note: ArcGIS Workflow Manager is available with Production Mapping and as a stand-alone extension.

Workflow Manager allows you to create a job (unit of work), assign or reassign the job to a user, and track the overall status of the project. Each job includes a predefined workflow that you build to represent your processes. When assigned a job, you execute the steps in the workflow to launch the appropriate tools, send e-mail notifications, or ask questions to determine the path the job should take.

Task Assistant Manager allows you to define workflows in ArcMap™ that guide you through various tasks. Task Assistant workflows can be used to provide step-by-step instructions for complex tasks, to minimize confusion for new users, or simply as a reference. Clicking a task in a workflow can execute either a tool in ArcMap or geoprocessing tools, set up your environment by specifying layer or snapping properties, or provide a description of what needs to be done.

See the white paper *Best Practices for Storing the ArcGIS Workflow Manager Workspace in an Enterprise Geodatabase for SQL Server*.

Mapping and Charting Solutions

Production Mapping can be used to build your own solutions by creating a product library to meet the standards of an industry or business. However, Esri has built three commercial off-the-shelf solutions for the defense mapping, maritime, and aviation industries that utilize and expand on Production Mapping functionality.

ArcSDE DBTUNE

DBTUNE storage parameters let you control how ArcSDE technology creates objects within a SQL Server database. You can determine things such as how to allocate space to a table or index, the Filegroup that a table or index is created in, and other SQL Server-specific storage attributes. They also allow you to specify one of the available storage formats for the geometry of a spatial column.

The DBTUNE storage parameters are stored in the DBTUNE table. The DBTUNE table, along with all other metadata tables, is created in the database when the Create Enterprise Geodatabase or Enable Enterprise Geodatabase tool is executed.

When a large number of database connections access the same files in the same location on disk, database performance is slower because the connections are competing with one another for the same resources. To reduce this competition, you can store database files in different locations on disk.

Thus, DBTUNE can be modified to store the Production feature dataset and tables in separate data files across different locations on disk. This will reduce disk contention and improve database input/output (I/O).

Standard GIS storage recommendations favor keeping index and log files separate from vector and tabular business tables. For performance reasons, it is better to position the business, feature, and spatial index tables separately and to position Filegroup data files based on their usage pattern. For a multiversioned, highly active editing geodatabase, database files of the VERSIONS Filegroup may be separated and dispersed across available disks to avoid I/O contention.

Disk Configuration

Large production enterprise geodatabase systems should employ a hardware striping solution. Your best disk and data organization strategies involve spreading your data across multiple disks.

When data is spread across multiple disks, more spindles can actively search for it. This can increase disk read time and decrease disk contention. However, too many disks can slow down a query. There are two main ways of achieving striping: Filegroups and redundant array of independent disks (RAID). You can also combine the two by creating Filegroups within disk arrays. You can employ data segregation strategies; keeping tables from indexes or certain types of tables from other tables will improve performance and alleviate administrative burdens.

The suggested SQL Server optimal configuration is as follows:

- Disk 0—SQL Server/Application software
- Disk 1—master, model, msdb
- Disk 2—tempdb
- Disk 3—Log files
- Disk 4—Feature data tables
- Disk 5—Spatial index data tables
- Disk 6—Attribute data/Business tables
- Disk 7—SQL Server indexes

Reducing Disk I/O Contention

As a rule, you should create database files that are as large as possible, based on the maximum amount of data you estimate the database will contain, to accommodate future growth. By creating large files, you can avoid file fragmentation and gain better database performance. In many cases, you can let data files grow automatically; just be sure to limit autogrowth by specifying a maximum growth size that leaves some hard disk space available. By putting different Filegroups on different disks, you can also minimize physical fragmentation of your files as they grow.

To configure data and log files for best performance, follow these best practices:

- To avoid disk contention, do not put data files on the drive that contains the operating system files.

- Put transaction log files and data files on separate drives. This will give you the best performance by reducing disk contention between data and transaction log files.

- Put the tempdb database on a separate drive if possible, preferably on a RAID 10 or RAID 5 system. In environments in which there is intensive use of tempdb databases, you can get better performance by putting tempdb on a separate drive, which would let SQL Server perform tempdb operations in parallel with database operations.

- Understand that the RAID configuration that is best for your database files depends on several factors, including performance and recoverability needs. RAID 10 is the recommended RAID system for transaction log, data, and index files. If you have budget restrictions, consider keeping transaction log files in a RAID 10 system and storing data and index files in a RAID 5 system.

  For more information about RAID, see RAID Levels and SQL Server at http://technet.microsoft.com/en-us/library/ms190764(SQL.105).aspx and chapter 7, "Planning Fault Tolerance and Avoidance," by Charlie Russel and Sharon Crawford, from *Microsoft Windows 2000 Server Administrator's Companion* (Microsoft Press) at http://technet.microsoft.com/en-us/library/bb742464.aspx.

- Use partitions on large tables. Partitioning lets you split a table across multiple Filegroups; you can place a subset of a table or index on a designated Filegroup. This capability lets you separate specific pieces of a table or index onto individual Filegroups and effectively manage file I/O for volatile tables. Partitions let you easily manage archival routines and data loading operations.

Below is a suggested design to reduce disk I/O contention:

File Type	Database Activity	Move File to Disk With
Transaction log files	Frequent edits	Relatively low I/O
Transaction log files	Few or no edits	Moderate I/O
tempdb	Frequent edits	Low I/O but separate from transaction log files
master, model, msdb	Few edits	Moderate I/O
Data	Frequent edits	Relatively low I/O

Transparent Data Encryption

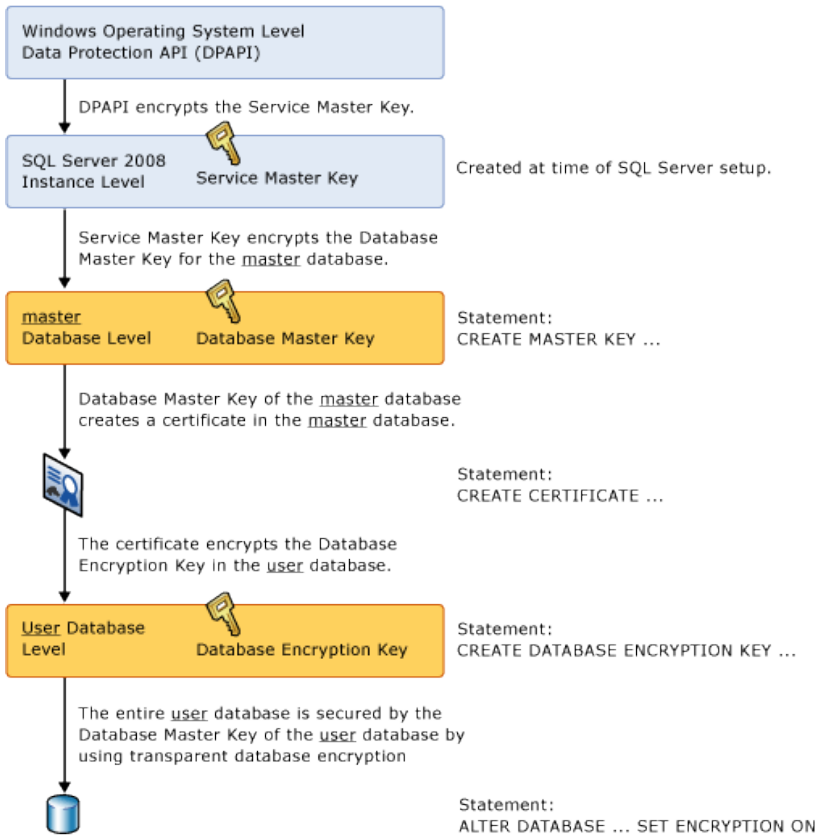
The precautions you can take to help protect the database include designing a secure system, encrypting confidential assets, and building a firewall around the database servers. However, if the physical media (drives or backup tapes) are stolen, a malicious party could just restore or attach the database and browse the data. One solution is to encrypt sensitive data in the database and protect the keys that are used to encrypt the data with a certificate. This prevents anyone without the keys from using the data, but this kind of protection must be planned in advance.

Transparent data encryption (TDE) performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is either a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an extensible key management (EKM) module. TDE protects data "at rest," meaning the data and log files. It provides the ability to comply with many laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using Advanced Encryption Standard

(AES) and Triple Data Encryption Standard (3DES) encryption algorithms without changing existing applications.

Database files are encrypted at the page level. The pages are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database.

The following illustration shows the architecture of TDE encryption:



TDE Encryption Architecture

Learn more about TDE at http://msdn.microsoft.com/en-us/library/bb934049.aspx.

To use TDE, follow these steps:

- Create a master key.
- Create or obtain a certificate protected by the master key.
- Create a database encryption key and protect it with the certificate.
- Set the database to use encryption.

```
USE master
GO
/* Verify master key */
SELECT * FROM sys.symmetric_keys WHERE name LIKE '%MS_DatabaseMasterKey%'
GO

/* if there are no records found, then it means there was no predefined Master Key.
 To create a Master Key, you can execute the below mentioned TSQL code. */

/* Create master key */
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'pmadmin';
GO
/* Backup master key */
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'pmadmin';
GO
BACKUP MASTER KEY TO FILE = 'D:\mssqlbackup\master\masterkey.mk'
    ENCRYPTION BY PASSWORD = 'pmadmin';
GO

/* Create Certificate */
CREATE CERTIFICATE pm_cert WITH SUBJECT = 'PM Server Certificate';
GO

/* Verify Certificate */
SELECT * FROM sys.certificates where [name] = 'pm_cert'
GO

/* Backup certificate */
BACKUP CERTIFICATE pm_cert TO FILE = 'D:\mssqlbackup\master\pm.cer'
   WITH PRIVATE KEY (
         FILE = 'D:\mssqlbackup\master\pm.pvk',
         ENCRYPTION BY PASSWORD = 'pmadmin');
GO

USE pmdb
GO
/* Create Encryption key */
CREATE DATABASE ENCRYPTION KEY
   WITH ALGORITHM = AES_256
   ENCRYPTION BY SERVER CERTIFICATE pm_cert;
GO

/* Encrypt database */
ALTER DATABASE pmdb SET ENCRYPTION ON;
GO

/* Verify Encryption */
SELECT
DB_NAME(database_id) AS DatabaseName
,Encryption_State AS EncryptionState
,key_algorithm AS Algorithm
,key_length AS KeyLength
FROM sys.dm_database_encryption_keys
GO
SELECT
NAME AS DatabaseName
,IS_ENCRYPTED AS IsEncrypted
FROM sys.databases where name ='pmdb'
GO
```

Step 1: Create Data Files

Create new file groups to store the production data.

```
TABLESPACE        DATAFILE_NAME     REMARKS
PM_BDATA          PM_Bdata          Business table
PM_BINDEX         PM_Bindex         Business table index
PM_FDATA          PM_Fdata          ST_Geomery Lob storage
PM_FINDEX         PM_Findex         ST_Geomery Lob index
PM_SDATA          PM_Sdata          Spatial Index Tables
PM_SINDEX         PM_Sindex         Spatial Index Features
PM_ADATA          PM_Adata          Adds table (versioned)
PM_AINDEX         PM_Aindex         Adds table index
PM_DDATA          PM_Ddata          Delete table (versioned)
PM_DINDEX         PM_Dindex         Delete table index
PM_RASTER         PM_Raster         Raster Datasets
PM_RINDEX         PM_Rindex         Raster Indexes
PM_RBLK           PM_Rblk           Raster BLK table
PM_RBLKIDX        PM_Rblkidx        Raster BLK table Indexes
PM_XMLDOC         PM_Xmldoc         XML Documents
PM_XMLIDX         PM_Xmlidx         XML Documents Indexes
PM_TOPO_BDATA     PM_Topo_Bdata     Topology Business table
PM_TOPO_BINDEX    PM_Topo_Bindex    Topology Business table index
PM_TOPO_FDATA     PM_Topo_Fdata     Topology ST_Geomery Lob storage
```

```
PM_TOPO_FINDEX    PM_Topo_Findex    Topology ST_Geomery Lob index
PM_TOPO_SDATA     PM_Topo_Sdata     Topology Spatial Index Tables
PM_TOPO_SINDEX    PM_Topo_Sindex    Topology Spatial Index Features
PM_TERRAIN_BDATA  PM_Terrain_Bdata  Terrain Business table
PM_TERRAIN_BINDEX PM_Terrain_Bindex Terrain Business table index
PM_TERRAIN_FDATA  PM_Terrain_Fdata  Terrain ST_Geomery Lob storage
PM_TERRAIN_FINDEX PM_Terrain_Findex Terrain ST_Geomery Lob index
PM_TERRAIN_SDATA  PM_Terrain_Sdata  Terrain Spatial Index Tables
PM_TERRAIN_SINDEX PM_Terrain_Sindex Terrain Spatial Index Features
PM_NET_BDATA      PM_Net_Bdata      Network Business table
PM_NET_BINDEX     PM_Net_Bindex     Network Business table index
PM_NET_FDATA      PM_Net_Fdata      Network ST_Geomery Lob storage
PM_NET_FINDEX     PM_Net_Findex     Network ST_Geomery Lob index
PM_NET_SDATA      PM_Net_Sdata      Network Spatial Index Tables
PM_NET_SINDEX     PM_Net_Sindex     Network Spatial Index Features

USE MASTER
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Bdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Bdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Bdata_02', FILENAME =
N'D:\mssqldata\pmdb\pm_Bdata_02.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Bindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Bindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Bindex_02', FILENAME =
N'D:\mssqldata\pmdb\pm_Bindex_02.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Fdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Fdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Findex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Findex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Sdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Sdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Sindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Sindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_XML_DOC]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Xml_doc_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Xml_doc_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_XML_DOC]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_XML_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Xml_index_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Xml_index_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_XML_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_ADATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Adata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Adata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_ADATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_AINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Aindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Aindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_AINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_DDATA]
GO
```

J10067

```
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Ddata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Ddata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_DDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_DINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Dindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Dindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_DINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_RASTER]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Raster_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Raster_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_RASTER]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_RASTER_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Raster_Index_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Raster_Index_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_RASTER_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Bdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Bdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Bindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Bindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Fdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Fdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Findex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Findex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Sdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Sdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TOPO_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Topo_Sindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Topo_Sindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_TOPO_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Bdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Bdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Bindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Bindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Fdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Fdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Findex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Findex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Sdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Sdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_SDATA]
GO
```

```
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_TERRAIN_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Terrain_Sindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Terrain_Sindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO
FILEGROUP [PM_TERRAIN_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Bdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Bdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_BDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Bindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Bindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_BINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Fdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Fdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_FDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Findex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Findex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_FINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Sdata_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Sdata_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_SDATA]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_NET_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_Net_Sindex_01', FILENAME =
N'D:\mssqldata\pmdb\pm_Net_Sindex_01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PM_NET_SINDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SDE_DICT]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_sde_dict_01', FILENAME =
N'D:\mssqldata\pmdb\pm_sde_dict_01.NDF' , SIZE = 1, MAXSIZE = 300, FILEGROWTH = 1) TO FILEGROUP
[PM_SDE_DICT]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SDE_DICT_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_sde_dict_index_01', FILENAME =
N'D:\mssqldata\pmdb\pm_sde_dict_index_01.NDF' , SIZE = 1, MAXSIZE = 100, FILEGROWTH = 1) TO
FILEGROUP [PM_SDE_DICT_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SDE_LOG]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_sde_log_01', FILENAME =
N'D:\mssqldata\pmdb\pm_sde_log_01.NDF' , SIZE = 1, MAXSIZE = 300, FILEGROWTH = 1) TO FILEGROUP
[PM_SDE_LOG]
GO
ALTER DATABASE [pmdb] ADD FILEGROUP [PM_SDE_LOG_INDEX]
GO
ALTER DATABASE [pmdb] ADD FILE(NAME = N'pm_sde_log_index_01', FILENAME =
N'D:\mssqldata\pmdb\pm_sde_log_index_01.NDF' , SIZE = 1, MAXSIZE = 100, FILEGROWTH = 1) TO
FILEGROUP [PM_SDE_LOG_INDEX]
GO
```

By setting the data files initial size to 1MB, there is no delay in the creation of the Filegroups; to avoid fragmentation, you can resize the data files.

```
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Bdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Bindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Fdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Findex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Sdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Sindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Adata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Aindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Ddata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Dindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Raster_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Rindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Rblk_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Rblkidx_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Xmldoc_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Xmlidx_01', SIZE = 400MB )
```

```
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Bdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Bindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Fdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Findex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Sdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Topo_Sindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Bdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Bindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Fdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Findex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Sdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Terrain_Sindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Bdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Bindex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Fdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Findex_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Sdata_01', SIZE = 400MB )
ALTER DATABASE [PMDB] MODIFY FILE ( NAME = N'pm_Net_Sindex_01', SIZE = 400MB )
```

Verify Filegroups and data files:

```
EXEC sp_helpdb pmdb
GO
```

Step 2: Create PM User

Create a new database user to store the production data, then grant the appropriate permissions.

Create user and schema:

```
USE master
GO
CREATE LOGIN pm WITH PASSWORD = 'pmadmin', DEFAULT_DATABASE=[pmdb],DEFAULT_LANGUAGE=[us_english],
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE PMDB
GO
CREATE USER [pmdb] FOR LOGIN [pmdb]
GO
CREATE SCHEMA [pmdb] AUTHORIZATION [pmdb]
GO
ALTER USER [pmdb] WITH DEFAULT_SCHEMA=[pmdb]
GO
```

Grant privileges:

```
USE PMDB
GO
EXEC sp_droprolemember 'gis_data_creator', 'pm'
GO
EXEC sp_droprole 'gis_data_creator'
GO
CREATE ROLE gis_data_creator AUTHORIZATION dbo
GO
GRANT CREATE TABLE TO gis_data_creator
GO
GRANT CREATE PROCEDURE TO gis_data_creator
GO
GRANT CREATE VIEW TO gis_data_creator
GO
EXEC sp_addrolemember 'gis_data_creator', 'pm'
GO
```

Verify role:

```
EXEC sp_helprolemember 'gis_data_creator'
GO
```

Verify role permissions:

```
select dp.NAME AS principal_name,
 dp.type_desc AS principal_type_desc,
 o.NAME AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
 from sys.database_permissions p
 left OUTER JOIN sys.all_objects o
 on p.major_id = o.OBJECT_ID
 inner JOIN sys.database_principals dp
 on p.grantee_principal_id = dp.principal_id
 where dp.NAME = 'gis_data_creator'
GO
```

Verify user permissions:

```
select USER_NAME(p.grantee_principal_id) AS principal_name,
 dp.type_desc AS principal_type_desc,
 p.class_desc,
 OBJECT_NAME(p.major_id) AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
from sys.database_permissions p
inner JOIN sys.database_principals dp
on p.grantee_principal_id = dp.principal_id
where USER_NAME(p.grantee_principal_id) = 'pm'
```

Associate login PM with user PM:

```
USE PMDB
GO
EXEC sp_change_users_login 'update_one','pm','pm'
GO
EXEC sp_helpuser 'pm'
GO
```

Step 3: Modify DBTUNE

Export the dbtune file before making any modification.

```
sdedbtune -o export -f dbtune_exp.sde -u sde -p sde -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb
```

Copy ***dbtune_exp.sde*** to ***dbtune_pm.sde***.

Fill the new dbtune file with the proper Filegroups:

```
dbtune_pm.sde

##DATA_DICTIONARY
B_CLUSTER_ROWID           0
B_CLUSTER_USER            0
B_INDEX_ROWID             "WITH FILLFACTOR = 75 ON PM_SDE_DICT_INDEX"
B_INDEX_USER              "WITH FILLFACTOR = 75 ON PM_SDE_DICT_INDEX"
B_STORAGE                 "ON PM_SDE_DICT"
I_STORAGE                 "ON PM_SDE_DICT"
MVTABLES_MODIFIED_INDEX   "WITH FILLFACTOR = 75 ON PM_SDE_DICT_INDEX"
MVTABLES_MODIFIED_TABLE   "ON PM_SDE_DICT"
STATE_LINEAGES_INDEX      "WITH FILLFACTOR = 75 ON SDE_STATE_INDEX"
STATE_LINEAGES_TABLE      "ON PM_SDE_STATE"
STATES_INDEX              "WITH FILLFACTOR = 75 ON SDE_STATE_INDEX"
STATES_TABLE              "ON PM_SDE_STATE"
VERSIONS_INDEX            "WITH FILLFACTOR = 75 ON SDE_STATE_INDEX"
VERSIONS_TABLE            "ON PM_SDE_STATE"
XML_INDEX_TAGS_INDEX      "WITH FILLFACTOR = 75 ON SDE_DICT_INDEX"
XML_INDEX_TAGS_TABLE      "ON PM_SDE_DICT"
END


##DEFAULTS
A_CLUSTER_RASTER          0
A_CLUSTER_ROWID           0
A_CLUSTER_SHAPE           1
A_CLUSTER_STATEID         0
A_CLUSTER_USER            0
A_CLUSTER_XML             0
A_INDEX_RASTER            "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_ROWID             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID           "WITH FILLFACTOR = 75 ON PM_AINDEX"
```

J10067

```
A_INDEX_USER                "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_XML                 "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_MS_SPINDEX                "GRIDS = (MEDIUM, MEDIUM, MEDIUM, MEDIUM), CELLS_PER_OBJECT = 16"
A_OUT_OF_ROW                0
A_STORAGE                   "ON PM_ADATA"
AUX_CLUSTER_COMPOSITE       1
AUX_INDEX_COMPOSITE         "WITH FILLFACTOR = 75 ON PM_RASTER_INDEX"
AUX_STORAGE                 "ON PM_RASTER"
B_CLUSTER_RASTER            0
B_CLUSTER_ROWID             0
B_CLUSTER_SHAPE             1
B_CLUSTER_TO_DATE           0
B_CLUSTER_USER              0
B_CLUSTER_XML               0
B_INDEX_RASTER              "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_INDEX_ROWID               "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_INDEX_SHAPE               "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_INDEX_TO_DATE             "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_INDEX_USER                "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_INDEX_XML                 "WITH FILLFACTOR = 75 ON PM_BINDEX"
B_MS_SPINDEX                "GRIDS = (MEDIUM, MEDIUM, MEDIUM, MEDIUM), CELLS_PER_OBJECT = 16"
B_OUT_OF_ROW                0
B_STORAGE                   "ON PM_BDATA"
BLK_CLUSTER_COMPOSITE       1
BLK_INDEX_COMPOSITE         "WITH FILLFACTOR = 75 ON PM_RASTER_INDEX"
BLK_OUT_OF_ROW              0
BLK_STORAGE                 "ON PM_RASTER"
BND_CLUSTER_COMPOSITE       0
BND_CLUSTER_ID              0
BND_INDEX_COMPOSITE         "WITH FILLFACTOR = 75 ON PM_RASTER_INDEX"
BND_INDEX_ID                "WITH FILLFACTOR = 75 ON PM_RASTER_INDEX"
BND_STORAGE                 "ON PM_RASTER"
COLLATION_NAME              ""
CROSS_DB_QUERY_FILTER       0
D_CLUSTER_ALL               0
D_CLUSTER_DELETED_AT        1
D_INDEX_ALL                 "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT          "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                   "ON PM_DDATA"
F_CLUSTER_FID               1
F_INDEX_AREA                "WITH FILLFACTOR = 75 ON PM_FINDEX"
F_INDEX_FID                 "WITH FILLFACTOR = 75 ON PM_FINDEX"
F_INDEX_LEN                 "WITH FILLFACTOR = 75 ON PM_FINDEX"
F_OUT_OF_ROW                0
F_STORAGE                   "ON PM_FDATA"
GEOM_SRID_CHECK             1
GEOMETRY_STORAGE            "GEOMETRY"
GEOMTAB_OUT_OF_ROW          0
GEOMTAB_PK                  "WITH FILLFACTOR = 75 ON PM_FINDEX"
GEOMTAB_STORAGE             "ON PM_FDATA"
I_STORAGE                   "ON PM_FDATA"
NUM_DEFAULT_CURSORS         -1
PERMISSION_CACHE_THRESHOLD 250
RAS_CLUSTER_ID              1
RAS_INDEX_ID                "WITH FILLFACTOR = 75 ON PM_RASTER_INDEX"
RAS_STORAGE                 "ON PM_RASTER"
RASTER_STORAGE              "BINARY"
S_CLUSTER_ALL               1
S_CLUSTER_SP_FID            0
S_INDEX_ALL                 "WITH FILLFACTOR = 75 ON PM_SINDEX"
S_INDEX_SP_FID              "WITH FILLFACTOR = 75 ON PM_SINDEX"
S_STORAGE                   "ON PM_SDATA"
UI_TEXT                     "User Interface text for DEFAULTS"
UNICODE_STRING              "TRUE"
XML_COLUMN_STORAGE          "DB_XML"
XML_DOC_INDEX               "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_DOC_OUT_OF_ROW          0
XML_DOC_STORAGE             "ON PM_XML_DOC"
XML_DOC_UNCOMPRESSED_TYPE   "BINARY"
XML_IDX_CLUSTER_DOUBLE      0
XML_IDX_CLUSTER_ID          0
XML_IDX_CLUSTER_PK          1
XML_IDX_CLUSTER_STRING      0
XML_IDX_CLUSTER_TAG         0
XML_IDX_FULLTEXT_CAT        "SDE_DEFAULT_CAT"
XML_IDX_FULLTEXT_LANGUAGE   ""
XML_IDX_FULLTEXT_TIMESTAMP 1
XML_IDX_FULLTEXT_UPDATE_METHOD "CHANGE_TRACKING BACKGROUND"
XML_IDX_INDEX_DOUBLE        "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_IDX_INDEX_ID            "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_IDX_INDEX_PK            "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_IDX_INDEX_STRING        "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_IDX_INDEX_TAG           "WITH FILLFACTOR = 75 ON PM_XML_INDEX"
XML_IDX_OUT_OF_ROW          0
XML_IDX_STORAGE             "ON PM_XML_INDEX"
END
```

```
##GEOGRAPHY
GEOMETRY_STORAGE            "GEOGRAPHY"
UI_TEXT                     "User Interface text description for GEOGRAPHY"
END

##GEOMETRY
GEOMETRY_STORAGE            "GEOMETRY"
UI_TEXT                     "User Interface text description for GEOMETRY"
END

##IMS_GAZETTEER
XML_COLUMN_STORAGE          "SDE_XML"
END
```

##LOGFILE_DEFAULTS

SESSION_TEMP_TABLE parameter must be set to 1 (true) to allow the session-based log file table to be created in tempdb. in this case to not fill the DBTUNE filegroups.

Example:

```
##LOGFILE_DEFAULTS
LD_INDEX_ALL          "WITH FILLFACTOR = 75"
LD_STORAGE            ""
LF_CLUSTER_ID          0
LF_CLUSTER_NAME        0
LF_INDEX_ID           "WITH FILLFACTOR = 75 "
LF_INDEX_NAME         "WITH FILLFACTOR = 75 "
LF_STORAGE            ""
SESSION_INDEX         "WITH FILLFACTOR = 75 "
SESSION_STORAGE         ""
SESSION_TEMP_TABLE      1
END
```

If you change the SESSION_TEMP_TABLE parameter to 0 (false), the SDE_LOGFILES, SDE_LOGFILE_DATA, and SDE_SESSION<SDE_ID> tables will be created in the connecting user's schema; hence, the user requires CREATE TABLE permission.

Example:

```
##LOGFILE_DEFAULTS
LD_INDEX_ALL          "WITH FILLFACTOR = 75 ON
PM_SDELOG_INDEX"
LD_STORAGE            "ON PM_SDELOG"
LF_CLUSTER_ID          0
LF_CLUSTER_NAME        0
LF_INDEX_ID           "WITH FILLFACTOR = 75 ON
PM_SDELOG_INDEX"
LF_INDEX_NAME         "WITH FILLFACTOR = 75 ON
PM_SDELOG_INDEX"
LF_STORAGE            "ON PM_SDELOG"
SESSION_INDEX         "WITH FILLFACTOR = 75 ON
PM_SDELOG_INDEX"
SESSION_STORAGE       "ON PM_SDELOG"
SESSION_TEMP_TABLE      0
END
```

```
##LOGFILE_DEFAULTS
LD_INDEX_ALL                "WITH FILLFACTOR = 75"
LD_STORAGE                  ""
LF_CLUSTER_ID               0
LF_CLUSTER_NAME             0
LF_INDEX_ID                 "WITH FILLFACTOR = 75"
LF_INDEX_NAME               "WITH FILLFACTOR = 75"
LF_STORAGE                  ""
SESSION_INDEX               "WITH FILLFACTOR = 75"
SESSION_STORAGE             ""
SESSION_TEMP_TABLE          1
END

##NETWORK_DEFAULTS
A_CLUSTER_ROWID             0
A_CLUSTER_SHAPE             1
A_CLUSTER_STATEID           0
A_CLUSTER_USER              0
A_INDEX_ROWID               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER                "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW                0
A_STORAGE                   "ON PM_ADATA"
B_CLUSTER_ROWID             0
B_CLUSTER_SHAPE             1
B_CLUSTER_USER              0
```

J10067

```
B_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_INDEX_SHAPE              "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_OUT_OF_ROW               0
B_STORAGE                  "ON PM_NET_BDATA"
COMMENT                    "The base system initialization parameters for NETWORK_DEFAULTS"
D_CLUSTER_ALL              0
D_CLUSTER_DELETED_AT       1
D_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT         "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                  ""
F_CLUSTER_FID              1
F_INDEX_AREA               "WITH FILLFACTOR = 75 ON PM_NET_FINDEX"
F_INDEX_FID                "WITH FILLFACTOR = 75 ON PM_NET_FINDEX"
F_INDEX_LEN                "WITH FILLFACTOR = 75 ON PM_NET_FINDEX"
F_OUT_OF_ROW               0
F_STORAGE                  "ON PM_NET_FDATA"
S_CLUSTER_ALL              1
S_CLUSTER_SP_FID           0
S_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_NET_SINDEX"
S_INDEX_SP_FID             "WITH FILLFACTOR = 75 ON PM_NET_SINDEX"
S_STORAGE                  "ON PM_NET_SDATA"
UI_NETWORK_TEXT            "The network default configuration"
END

##NETWORK_DEFAULTS::DESC
A_CLUSTER_ROWID            1
A_CLUSTER_STATEID          0
A_CLUSTER_USER             0
A_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID            "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW               0
A_STORAGE                  "ON PM_ADATA"
B_CLUSTER_ROWID            1
B_CLUSTER_USER             0
B_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_OUT_OF_ROW               0
B_STORAGE                  "ON PM_NET_BDATA"
D_CLUSTER_ALL              0
D_CLUSTER_DELETED_AT       1
D_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT         "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                  "ON PM_DDATA"
END

##NETWORK_DEFAULTS::NETWORK
A_CLUSTER_ROWID            1
A_CLUSTER_STATEID          0
A_CLUSTER_USER             0
A_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID            "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW               0
A_STORAGE                  "ON PM_ADATA"
B_CLUSTER_ROWID            1
B_CLUSTER_USER             0
B_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_NET_BINDEX"
B_OUT_OF_ROW               0
B_STORAGE                  "ON PM_NET_BDATA"
D_CLUSTER_ALL              0
D_CLUSTER_DELETED_AT       1
D_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT         "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                  "ON PM_DDATA"
END

##NETWORK_GEOGRAPHY
GEOMETRY_STORAGE           "GEOGRAPHY"
UI_NETWORK_TEXT            "User Interface description for SQL Server GEOGRAPHY"
END

##NETWORK_GEOGRAPHY::DESC
GEOMETRY_STORAGE           "GEOGRAPHY"
END

##NETWORK_GEOGRAPHY::NETWORK
GEOMETRY_STORAGE           "GEOGRAPHY"
END

##NETWORK_GEOMETRY
GEOMETRY_STORAGE           "GEOMETRY"
UI_NETWORK_TEXT            "User Interface description for SQL Server GEOMETRY"
END
```

```
##NETWORK_GEOMETRY::DESC
GEOMETRY_STORAGE           "GEOMETRY"
END

##NETWORK_GEOMETRY::NETWORK
GEOMETRY_STORAGE           "GEOMETRY"
END

##NETWORK_SDEBINARY
GEOMETRY_STORAGE           "SDEBINARY"
UI_NETWORK_TEXT            "User Interface description for SQL Server SDEBINARY"
END

##NETWORK_SDEBINARY::DESC
GEOMETRY_STORAGE           "SDEBINARY"
END

##NETWORK_SDEBINARY::NETWORK
GEOMETRY_STORAGE           "SDEBINARY"
END

##SDEBINARY
GEOMETRY_STORAGE           "SDEBINARY"
UI_TEXT                    "User Interface text description for SDEBINARY"
END

##TERRAIN_DEFAULTS
A_CLUSTER_ROWID            0
A_CLUSTER_SHAPE            1
A_CLUSTER_STATEID          0
A_CLUSTER_USER             0
A_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID            "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW               0
A_STORAGE                  "ON PM_ADATA"
B_CLUSTER_ROWID            0
B_CLUSTER_SHAPE            1
B_CLUSTER_USER             0
B_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_INDEX_SHAPE              "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_OUT_OF_ROW               0
B_STORAGE                  "ON PM_TERRAIN_BDATA"
D_CLUSTER_ALL              0
D_CLUSTER_DELETED_AT       1
D_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT         "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                  "ON PM_DDATA"
F_CLUSTER_FID              1
F_INDEX_AREA               "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_INDEX_FID                "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_INDEX_LEN                "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_OUT_OF_ROW               0
F_STORAGE                  "ON PM_TERRAIN_FDATA"
S_CLUSTER_ALL              1
S_CLUSTER_SP_FID           0
S_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_TERRAIN_SINDEX"
S_INDEX_SP_FID             "WITH FILLFACTOR = 75 ON PM_TERRAIN_SINDEX"
S_STORAGE                  "ON PM_TERRAIN_SDATA"
UI_TERRAIN_TEXT            "The terrain default configuration"
END

##TERRAIN_DEFAULTS::EMBEDDED
A_CLUSTER_ROWID            0
A_CLUSTER_SHAPE            1
A_CLUSTER_STATEID          0
A_CLUSTER_USER             0
A_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID            "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW               0
A_STORAGE                  "ON PM_ADATA"
B_CLUSTER_ROWID            0
B_CLUSTER_SHAPE            1
B_CLUSTER_USER             0
B_INDEX_ROWID              "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_INDEX_SHAPE              "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_INDEX_USER               "WITH FILLFACTOR = 75 ON PM_TERRAIN_BINDEX"
B_OUT_OF_ROW               0
B_STORAGE                  "ON PM_TERRAIN_BDATA"
D_CLUSTER_ALL              0
D_CLUSTER_DELETED_AT       1
D_INDEX_ALL                "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT         "WITH FILLFACTOR = 90 ON PM_DINDEX"
D_STORAGE                  "ON PM_DDATA"
```

J10067

```
F_CLUSTER_FID             1
F_INDEX_AREA              "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_INDEX_FID               "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_INDEX_LEN               "WITH FILLFACTOR = 75 ON PM_TERRAIN_FINDEX"
F_OUT_OF_ROW              0
F_STORAGE                 "ON PM_TERRAIN_FDATA"
S_CLUSTER_ALL             1
S_CLUSTER_SP_FID          0
S_INDEX_ALL               "WITH FILLFACTOR = 75 ON PM_TERRAIN_SINDEX"
S_INDEX_SP_FID            "WITH FILLFACTOR = 75 ON PM_TERRAIN_SINDEX"
S_STORAGE                 "ON PM_TERRAIN_SDATA"
END

##TERRAIN_GEOGRAPHY
GEOMETRY_STORAGE          "GEOGRAPHY"
UI_TERRAIN_TEXT           "User Interface description for SQL Server GEOGRAPHY"
END

##TERRAIN_GEOGRAPHY::EMBEDDED
GEOMETRY_STORAGE          "GEOGRAPHY"
END

##TERRAIN_GEOMETRY
GEOMETRY_STORAGE          "GEOMETRY"
UI_TERRAIN_TEXT           "User Interface description for SQL Server GEOMETRY"
END

##TERRAIN_GEOMETRY::EMBEDDED
GEOMETRY_STORAGE          "GEOMETRY"
END

##TERRAIN_SDEBINARY
GEOMETRY_STORAGE          "SDEBINARY"
UI_TERRAIN_TEXT           "User Interface description for SQL Server SDEBINARY"
END

##TERRAIN_SDEBINARY::EMBEDDED
GEOMETRY_STORAGE          "SDEBINARY"
END

##TOPOLOGY_DEFAULTS
A_CLUSTER_ROWID           0
A_CLUSTER_SHAPE           1
A_CLUSTER_STATEID         0
A_CLUSTER_USER            0
A_INDEX_ROWID             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID           "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW              0
A_STORAGE                 "ON PM_ADATA"
B_CLUSTER_ROWID           0
B_CLUSTER_SHAPE           1
B_CLUSTER_USER            0
B_INDEX_ROWID             "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_INDEX_SHAPE             "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_INDEX_USER              "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_OUT_OF_ROW              0
B_STORAGE                 "ON PM_TOPO_BDATA"
D_CLUSTER_ALL             0
D_CLUSTER_DELETED_AT      1
D_INDEX_ALL               "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT        "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_STORAGE                 "ON PM_DDATA"
F_CLUSTER_FID             1
F_INDEX_AREA              "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_INDEX_FID               "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_INDEX_LEN               "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_OUT_OF_ROW              0
F_STORAGE                 "ON PM_TOPO_FDATA"
S_CLUSTER_ALL             1
S_CLUSTER_SP_FID          0
S_INDEX_ALL               "WITH FILLFACTOR = 75 ON PM_TOPO_SINDEX"
S_INDEX_SP_FID            "WITH FILLFACTOR = 75 ON PM_TOPO_SINDEX"
S_STORAGE                 "ON PM_TOPO_SDATA"
UI_TOPOLOGY_TEXT          "The topology default configuration"
END

##TOPOLOGY_DEFAULTS::DIRTYAREAS
A_CLUSTER_ROWID           0
A_CLUSTER_SHAPE           1
A_CLUSTER_STATEID         0
A_CLUSTER_USER            0
A_INDEX_ROWID             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_SHAPE             "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_STATEID           "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_INDEX_USER              "WITH FILLFACTOR = 75 ON PM_AINDEX"
A_OUT_OF_ROW              0
```

J10067

```
A_STORAGE                   "ON PM_ADATA"
B_CLUSTER_ROWID             0
B_CLUSTER_SHAPE             1
B_CLUSTER_USER              0
B_INDEX_ROWID               "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_INDEX_SHAPE               "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_INDEX_USER                "WITH FILLFACTOR = 75 ON PM_TOPO_BINDEX"
B_OUT_OF_ROW                0
B_STORAGE                   "ON PM_TOPO_BDATA"
D_CLUSTER_ALL               0
D_CLUSTER_DELETED_AT        1
D_INDEX_ALL                 "WITH FILLFACTOR = 75 ON PM_DINDEX"
D_INDEX_DELETED_AT          "WITH FILLFACTOR = 90 ON PM_DINDEX"
D_STORAGE                   "ON PM_DDATA"
F_CLUSTER_FID               1
F_INDEX_AREA                "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_INDEX_FID                 "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_INDEX_LEN                 "WITH FILLFACTOR = 75 ON PM_TOPO_FINDEX"
F_OUT_OF_ROW                0
F_STORAGE                   "ON PM_TOPO_FDATA"
S_CLUSTER_ALL               1
S_CLUSTER_SP_FID            0
S_INDEX_ALL                 "WITH FILLFACTOR = 75 ON PM_TOPO_SINDEX"
S_INDEX_SP_FID              "WITH FILLFACTOR = 75 ON PM_TOPO_SINDEX"
S_STORAGE                   "ON PM_TOPO_SDATA"
END

##TOPOLOGY_GEOGRAPHY
GEOMETRY_STORAGE            "GEOGRAPHY"
UI_TOPOLOGY_TEXT            "User Interface description for SQL Server GEOGRAPHY"
END

##TOPOLOGY_GEOGRAPHY::DIRTYAREAS
GEOMETRY_STORAGE            "GEOGRAPHY"
END

##TOPOLOGY_GEOMETRY
GEOMETRY_STORAGE            "GEOMETRY"
UI_TOPOLOGY_TEXT            "User Interface description for SQL Server GEOMETRY"
END

##TOPOLOGY_GEOMETRY::DIRTYAREAS
GEOMETRY_STORAGE            "GEOMETRY"
END

##TOPOLOGY_SDEBINARY
GEOMETRY_STORAGE            "SDEBINARY"
UI_TOPOLOGY_TEXT            "User Interface description for SQL Server SDEBINARY"
END

##TOPOLOGY_SDEBINARY::DIRTYAREAS
GEOMETRY_STORAGE            "SDEBINARY"
END

##WKB_GEOMETRY
GEOMETRY_STORAGE            "OGCWKB"
UI_TEXT                     "User Interface text description for OGC WKB"
END
```

```
##LOGFILE_DEFAULTS
SESSION_TEMP_TABLE parameter must be set to 1
(true) to allow the session-based log file table to be
created in tempdb. in this case to not fill the DBTUNE
filegroups.

Example:

##LOGFILE_DEFAULTS
LD_INDEX_ALL              "WITH FILLFACTOR = 75"
LD_STORAGE                ""
LF_CLUSTER_ID             0
LF_CLUSTER_NAME           0
LF_INDEX_ID               "WITH FILLFACTOR = 75 "
LF_INDEX_NAME             "WITH FILLFACTOR = 75 "
LF_STORAGE                ""
SESSION_INDEX             "WITH FILLFACTOR = 75 "
SESSION_STORAGE           ""
SESSION_TEMP_TABLE        1
END

If you change the SESSION_TEMP_TABLE parameter
to 0 (false), the SDE_LOGFILES,
SDE_LOGFILE_DATA, and SDE_SESSION<SDE_ID>
tables will be created in the connecting user's schema;
hence, the user requires CREATE TABLE permission.

Example:

##LOGFILE_DEFAULTS
LD_INDEX_ALL              "WITH FILLFACTOR = 75
ON PM_SDELOGIDX"
LD_STORAGE                "ON PM_SDELOG"
LF_CLUSTER_ID             0
LF_CLUSTER_NAME           0
LF_INDEX_ID               "WITH FILLFACTOR = 75
ON PM_SDELOGIDX"
LF_INDEX_NAME             "WITH FILLFACTOR = 75
ON PM_SDELOGIDX"
LF_STORAGE                "ON PM_SDELOG"
SESSION_INDEX             "WITH FILLFACTOR = 75
ON PM_SDELOGIDX"
SESSION_STORAGE           "ON PM_SDELOG"
SESSION_TEMP_TABLE        0
END
```

Import the modified ***dbtune_pm.sde*** file.

```
sdedbtune -o import -f dbtune_pm.sde -u sde -p sde -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb
```

Step 4: Configure SQL Server Parameters

It is recommended that you use the following parameter values when creating a SQL Server database:

SQL Server Parameters

Parameter Name	Value
Server Memory: Use AWE to allocate memory	Enabled
Processors: Boost SQL Server Priority	Enabled
Security SQL Server and Windows Authentication mode	Enabled
Connections: Maximum number of concurrent connections	0 = unlimited
Connections: Allow remote connections to this server	Enabled

Step 5: Configure ArcSDE Parameters

You must configure the MAXBLOBSIZE and TCPKEEPALIVE parameters for the ArcSDE geodatabase used as the product library. The MAXBLOBSIZE value is -1 by default. However, if you are using SQL Server or another enterprise DBMS, make sure that this value is set to -1 and the TCPKEEPALIVE value is set to TRUE. This command should be used at the command prompt of a machine where ArcSDE is installed.

```
sdeconfig -o alter -v MAXBLOBSIZE=-1 -i <service> -u sde -p <sde_password>
sdeconfig -o alter -v TCPKEEPALIVE=TRUE -i <service> -u sde -p <sde_password>
```

For more information, see the ArcSDE Administration Command Reference.

Step 6: Load the Data

Prepare your geodatabase for data loading.

Back up your database.

Change the ArcSDE buffer size:

```
sdeconfig -o alter -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin  -N -q
-v MINBUFSIZE=409600
sdeconfig -o alter -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin -N -q
-v MAXBUFSIZE=819200
```

Set the ArcSDE temp folder:

```
sdeconfig -o alter -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin -N -q
-v TEMP=D:\TEMP
```

List the ArcSDE parameters:

```
sdeconfig -o list -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin
```

Set the database recovery model to simple:

```
ALTER DATABASE [pmdb] SET RECOVERY SIMPLE WITH NO_WAIT
```

Increase the log file size:

```
ALTER DATABASE [pmdb] MODIFY FILE ( NAME = N'pmdb_Log01', MAXSIZE = 20480000KB )  --20GB
```

Load 10 percent of the data, estimate the total size of each data file, then resize the data files accordingly.

If loading or appending data to an existing feature class, even if the feature class is empty but you have to load a large amount of data, change the layer I/O mode to load_only_io; loading will be faster because indexes are disabled.

```
sdelayer -o  load_only_io -l contour_l,shape -D pmdb -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -s mysrv
-u sde -p sde
```

Load your data model with the PM user, then load the production mapping data.

Back up your database.

Step 7: Register as Versioned

In ArcCatalog™, register the PM schema as versioned.

Step 8: Verify Storage

Run the SQL queries below to verify that the production tables and indexes were created under the correct Filegroups:

```
USE PMDB
GO
```

List Filegroups and data files:

```
EXEC sp_helpdb pmdb
GO
```

List Filegroups data files:

```
EXEC sp_helpfilegroup 'PRIMARY'
GO
```

List tables by Filegroups:

```
SELECT USER_NAME(o.uid) [Owner],
 OBJECT_NAME(i.id) [Table Name],
 FILEGROUP_NAME(groupid) AS [Filegroup Name]
 FROM sysindexes i inner join sysobjects o
 ON i.id = o.id
 WHERE i.indid IN (0, 1) AND OBJECTPROPERTY(i.id, 'IsMSShipped') = 0 AND
 USER_NAME(o.uid) = 'pm'
 ORDER BY 1,3,2
GO
```

List indexes by table and Filegroups:

```
select 'owner'=user_name(o.uid)
,'table_name'=object_name(i.id),i.indid
,'index_name'=i.name ,i.groupid
,'filegroup'=f.name ,'file_name'=d.physical_name
,'dataspace'=s.name from sys.sysindexes i
,sys.sysobjects o,sys.filegroups f
,sys.database_files d, sys.data_spaces s
where objectproperty(i.id,'IsUserTable') = 1
and i.id = o.id
and f.data_space_id = i.groupid
and f.data_space_id = d.data_space_id
and f.data_space_id = s.data_space_id
and user_name(o.uid) = 'pm'
order by object_name(i.id),i.name,f.name
go
```

If any tables or indexes are stored in the wrong Filegroup, use ALTER TABLE and ALTER INDEX to change the Filegroup (see Books Online for SQL Server 2012 at http://msdn.microsoft.com/en-us/library/ms130214.aspx).

Also, in Management Studio, you can re-create the DDL script of tables and indexes. Then, within *create script*, you can modify the Filegroup parameter and re-create the tables and indexes in the correct file groups. This is particularly useful when tables are empty and you are allowed to re-create database objects.

Step 9: Using Data Compression

Row and page compression for tables and indexes enables you to save storage space by reducing the size of the database. Data compression has the drawback of increasing CPU usage because the data must be compressed and decompressed when being accessed. You cannot use data compression with system tables, and only the Enterprise and Developer editions of SQL Server 2012 support data compression.

You can configure data compression on the following:

- Clustered tables
- Heap tables (A heap is a table without a clustered index.)
- Nonclustered indexes
- Indexed views
- Individual partitions of a partitioned table or index

There are three forms of data compression you can use with SQL Server 2012: row-level compression, Unicode compression, and page-level compression.

Note: More information on heaps is available at http://msdn.microsoft.com/en-us/library/hh213609(v=sql.110).aspx.

Row-Level Compression
Row-level compression works by using more efficient storage formats for fixed-length data.

Row-level compression uses the following strategies to save space:

- Storing fixed-length numeric data types and CHAR data types as though they were variable-length data types
- Not storing NULL or 0 values
- Reducing metadata required to store data

Although it does reduce the amount of space that data uses, row-level compression does not provide the storage improvements of page-level compression. The advantage of row-level compression is that it requires less CPU usage than page-level compression. Use the following syntax to compress a table by using row-level compression:

```
ALTER TABLE tableName REBUILD WITH (DATA_COMPRESSION=ROW)
```

For example, to rebuild all partitions of the pm.TableA table of the PMDB database by using row compression, use the following query:

```
USE [PMDB]
ALTER TABLE [pmdb].[TableA] REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW)
```

Use the following syntax to configure an index with row-level compression:

```
ALTER INDEX indexName ON tableName REBUILD PARTITION ALL WITH
(DATA_COMPRESSION=ROW)
```

Note: More information on row-level compression is available at http://msdn.microsoft.com/en-us/library/cc280576(v=sql.110).aspx.

Unicode Compression
Unicode compression enables the database engine to compress Unicode values stored in page- or row-compressed objects. You can use Unicode compression with the fixed-length nchar(n) and nvarchar(n) data types. Unicode compression is automatically used where appropriate when you enable row and page compression.

Note: More information on Unicode compression is available at http://msdn.microsoft.com/en-us/library/ee240835(SQL.110).aspx.

Page-Level Compression
Page-level compression compresses data by storing repeating values and common prefixes only once and then making references to those values from other locations within the table. When page compression is applied to a table, row compression techniques are also applied. Page-level compression uses the following strategies:

- Row-level compression is applied to maximize the number of rows stored on a page.
- Column prefix compression is applied by replacing repeating data patterns with references.

- This data is stored in the page header.
- Dictionary compression scans for repeating values and then stores this information in the page header.

The benefits of page compression depend on the type of data that is compressed. Data that involves many repeating values will be more compressed than data populated by more unique values. Use the following general syntax to apply page-level compression:

```
ALTER TABLE name REBUILD WITH (DATA_COMPRESSION=PAGE)
```

For example, to rebuild all partitions of the pm.TableA table of the PMDB database by using page compression, use the following query:

```
USE [PMDB]
ALTER TABLE [pmdb].[TableA] REBUILD PARTITION = ALL
WITH
(DATA_COMPRESSION = PAGE)
```

Use the following syntax to configure an index with page-level compression:

```
ALTER INDEX indexName ON tableName REBUILD PARTITION ALL WITH
(DATA_COMPRESSION=PAGE)
```

Note: More information on page-level compression is available at http://msdn.microsoft.com/en-us/library/cc280464(v=sql.110).aspx.

If tables or indexes are partitioned, you can configure compression on a per-partition basis. If you split a partition by using the ALTER PARTITION statement, the new partitions inherit the data compression attribute of the original partition. If you merge two partitions, the resultant partition has the compression attribute of the destination partition. Although compression does allow more rows to be stored on a page, it doesn't alter the maximum row size of a table or index. You can't enable a table for compression if the maximum row size and the compression overhead exceed 8,060 bytes. The default compression setting for indexes is NONE, and you must specify the compression property for indexes when you create them. Nonclustered indexes do not inherit the compression property of the table, but clustered indexes created on a heap inherit the compression state of the heap. Data compression applies only at the source, so when you export data from a compressed source, SQL Server will output the data in uncompressed row format. Importing uncompressed data into a target table enabled for compression will compress the data.

Note: More information on data compression is available at http://msdn.microsoft.com/en-us/library/cc280449(v=sql.110).aspx.

You can configure compression by using the preceding Transact-SQL statements or from SQL Server Management Studio by using the Data Compression Wizard on either tables or indexes. You can use the Data Compression Wizard to add and remove compression. To use the Data Compression Wizard to change the compression settings for both tables and indexes, perform the following steps:

1. In SQL Server Management Studio, right-click the table or index you want to compress, choose Storage, and then select Manage Compression.

2. On the Welcome To The Data Compression Wizard page, click Next.

3. On the Select Compression Type page, you can choose to use the same compression type for all partitions or choose among Row, Page, and None on a per-partition basis. Click Calculate to determine the difference between current space usage and compressed usage.

4. On the Select An Output Option page, choose whether to create a script, to perform the operation immediately, or to perform the option according to a schedule. Click Next and then click Finish to complete the wizard.

Note: More information on the Data Compression Wizard is available at http://msdn.microsoft.com/en-us/library/cc280496(v=SQL.110).aspx.

Estimating Compression
The best way to determine the benefits of compression on an object is to use the sp_estimate_data_compression_savings stored procedure. The benefits of compression depend on factors such as the uniqueness of data. The sp_estimate_data_compression_savings stored procedure is available in the Enterprise edition of SQL Server only.

The syntax of the stored procedure is as follows:

```
sp_estimate_data_compression_savings[ @schema_name = ]
'schema_name', [ @object_name = ]
'object_name', [@index_id = ] index_id,[@partition_number = ]
partition_number,
[@data_compression = ] 'data_compression'
```

For example, to configure an estimate of the compression benefits of using row compression on the pm.TableA table in the PMDB database, execute the following Transact-SQL statement:

```
USE PMDB;
GO
EXEC sp_estimate_data_compression_savings 'pm', 'TableA', NULL,
NULL,
'ROW';
GO
```

To configure an estimate of the compression benefits of using page compression on the same table, execute the following Transact-SQL statement:

```
USE PMDB;
GO
EXEC sp_estimate_data_compression_savings 'pm', 'TableA', NULL,
NULL,
'PAGE';
GO
```

Note: More information about SP_ESTIMATE_DATA_COMPRESSION_SAVINGS is available at http://msdn.microsoft.com/en-us/library/cc280574(v=sql.110).aspx.

You can use the SQL script below to list the PM user tables and indexes and to generate the SQL statements to set page compression for the tables and indexes.

```
/*-----------------------------------------------------------------------
-- Verify PM Schema Storage
-----------------------------------------------------------------------*/
USE [pmdb]
GO
/*-----------------------------------------------------------------------
--List all tables
-----------------------------------------------------------------------*/
SELECT USER_NAME(o.uid) [owner], o.name,o.id,o.type,o.status
FROM sysobjects o
WHERE USER_NAME(o.uid) = 'pm'
AND o.type <> 'S' and o.type = 'U'
ORDER BY o.name,o.type;
GO
/*-----------------------------------------------------------------------
--List all indexes
-----------------------------------------------------------------------*/
SELECT USER_NAME(o.uid) [owner], OBJECT_NAME(i.id) [table], i.name
[index],o.type [type]
FROM sysindexes i inner join sysobjects o ON i.id = o.id
WHERE USER_NAME(o.uid) = 'pm'
AND o.type <> 'S' and o.type = 'U' and i.indid = 1
ORDER BY USER_NAME(o.uid),OBJECT_NAME(i.id),i.name;
GO

/*-----------------------------------------------------------------------
--Table page compression
--Example:
/*
ALTER TABLE PM.CITIES
REBUILD WITH (DATA_COMPRESSION = PAGE);
GO
*/
-----------------------------------------------------------------------*/
--Generate script to set table page compression:
SELECT 'ALTER TABLE ' + USER_NAME(o.uid) + '.' + o.name + ' REBUILD WITH
(DATA_COMPRESSION = PAGE);' [TXTSQL]
FROM sysobjects o
WHERE USER_NAME(o.uid) = 'pm'
AND o.type <> 'S' and o.type = 'U'
ORDER BY o.name,o.type;
GO

/*-----------------------------------------------------------------------
--Index page compression
--Example:
/*
ALTER INDEX R125_pk
ON PM.CITIES
REBUILD WITH ( DATA_COMPRESSION = PAGE ) ;
GO
*/
-----------------------------------------------------------------------*/
--Generate script to set index page compression:
SELECT 'ALTER INDEX ' + i.name + ' ON ' + USER_NAME(o.uid) + '.' +
OBJECT_NAME(i.id) +
       ' REBUILD WITH ( DATA_COMPRESSION = PAGE );' [TXTSQL]
FROM sysindexes i inner join sysobjects o ON i.id = o.id
WHERE USER_NAME(o.uid) = 'pm'
AND o.type <> 'S' and o.type = 'U' and i.indid = 1
ORDER BY USER_NAME(o.uid),OBJECT_NAME(i.id),i.name;
GO
```

Compression and TDE

Encryption of the database file is performed at the page level. The pages are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database.

SQL Server 2012—Transparent Data Encryption (TDE)

Encrypted data compresses significantly less than equivalent unencrypted data.

If TDE is used to encrypt a database, backup compression will not be able to significantly compress the backup storage.

SQL Server 2012—Data Compression

Step 10: Prepare Geodatabase for Editing

Prepare the geodatabase for normal online transaction processing (OLTP) editing.

After loading the data, change the layer to normal I/O:

```
sdelayer -o  normal_io -l contour_l,shape -D pmdb -i sde:sqlserver:mcsdbsrv -s mcsdbsrv
-u sde -p sde
```

Change the ArcSDE buffer size:

```
sdeconfig -o alter -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin  -N -q -v
MINBUFSIZE=16384
sdeconfig -o alter -i sde:sqlserver:mcsdbsrv -s mcsdbsrv -D pmdb -u sde -p sdeadmin -N -q -v
MAXBUFSIZE=65536
```

Shrink the log file:

```
DBCC SHRINKFILE (pmdb_Log01)
```

Shrink the database:

```
DBCC SHRINKDATABASE (pmdb)
```

Update the statistics:

```
USE pmdb
GO
EXEC sp_updatestats
GO
```

Set the database recovery model to full:

```
ALTER DATABASE [pmdb] SET RECOVERY FULL WITH NO_WAIT
```

Back up your database.

Step 11: Grant Permissions and Verify Roles

Grant permissions to the PM tables through the schema:

```
USE [pmdb]
GO
```

```
EXEC sp_addrole 'pm_editor', 'pm'
GO
GRANT DELETE ON SCHEMA::[pmdb] TO [pm_editor]
GO
GRANT EXECUTE ON SCHEMA::[pmdb] TO [pm_editor]
GO
GRANT INSERT ON SCHEMA::[pmdb] TO [pm_editor]
GO
GRANT SELECT ON SCHEMA::[pmdb] TO [pm_editor]
```

```
GO
GRANT UPDATE ON SCHEMA::[pmdb] TO [pm_editor]
GO
EXEC sp_addrole 'pm_viewer', 'pm'
GO
GRANT SELECT ON SCHEMA::[pmdb] TO [pm_viewer]
GO
```

Verify role permissions:

```
select USER_NAME(p.grantee_principal_id) AS principal_name,
 dp.type_desc AS principal_type_desc,
 p.class_desc,
 OBJECT_NAME(p.major_id) AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
from sys.database_permissions p
inner JOIN sys.database_principals dp
on p.grantee_principal_id = dp.principal_id
where USER_NAME(p.grantee_principal_id) in ('pm_editor','pm_viewer')
GO
```

Grant the PM_EDITOR role to ArcSDE editor users and the PM_VIEWER role to ArcSDE viewer users.

Step 12: Configure Log File Tables

Enterprise geodatabases use log file tables to maintain lists of selected records. Records are written to log file tables for later use by the application whenever a selection of a specific size is made, a reconciliation or post on a versioned database is performed, or a disconnected editing checkout is done in a client application. The log file tables store the ObjectIDs of the selected features so they can be redisplayed. This allows faster analysis and processing of information.

In ArcGIS, by default, log file tables are used if the selection set contains 100 or more records. This selection threshold of 100 features is set in the registry. It can be changed; however, Esri does not recommend doing so. There is no proven performance reason for changing it, and doing so could even cause performance problems. Thus, log file tables store feature selections in ArcMap that have more than 100 records for each connected ArcSDE editor/viewer user.

Log file options are set using specific parameters in the SERVER_CONFIG and DBTUNE tables (sde_server_config and sde_dbtune in a SQL Server database). Parameters in these tables are altered using the sdeconfig and sdedbtune commands, respectively.

In SQL Server, one table is created in tempdb in the format ##SDE_session<sde_id>. This table is truncated when the connecting application deletes its log files, and the table is dropped when the session disconnects. When using the default setting, users do not require CREATE TABLE permission in the database for the session table to be created in tempdb.

Learn more about ArcSDE log file tables at
Log file table configuration options for geodatabases in SQL Server.

Step 13: Create ArcSDE Users

The example below shows how to create an editor and viewer ArcSDE user.

Editor User

```
USE master
GO
CREATE LOGIN giseditor WITH PASSWORD = 'giseditor',
DEFAULT_DATABASE=[pmdb],DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
```

Create user:

```
USE [pmdb]
GO
CREATE USER [giseditor] FOR LOGIN [giseditor]
GO
```

Grant privileges:

```
USE [pmdb]
GO
EXEC sp_addrolemember N'pm_editor', N'giseditor' GO
GO
```

Viewer User

```
USE master
GO
CREATE LOGIN gisviewer WITH PASSWORD = 'gisviewer',
DEFAULT_DATABASE=[pmdb],DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
```

Create user:

```
USE [pmdb]
GO
CREATE USER [gisviewer] FOR LOGIN [gisviewer]
GO
```

Grant privileges:

```
USE [pmdb]
GO
EXEC sp_addrolemember N'pm_viewer', N'gisviewer' GO
GO
```

Conclusion

You can reduce disk contention and improve database I/O by storing the production mapping data in different locations on disk. However, this practice alone does not guarantee optimal database performance, and additional tuning tasks may be needed.

Learn more about the recommended tuning tasks:

Minimize disk I/O contention in SQL Server
What type of maintenance is needed for a geodatabase?

For more information on Esri Production Mapping, visit esri.com/productionmapping or e-mail productionmapping@esri.com.

Access blogs, forums, downloads, and more, from the Esri Production Mapping resource center.

You can access other resources at ArcGIS 10.2 for Desktop Help and Esri Support.

Esri inspires and enables people to positively impact their future through a deeper, geographic understanding of the changing world around them.

Governments, industry leaders, academics, and nongovernmental organizations trust us to connect them with the analytic knowledge they need to make the critical decisions that shape the planet. For more than 40 years, Esri has cultivated collaborative relationships with partners who share our commitment to solving earth's most pressing challenges with geographic expertise and rational resolve. Today, we believe that geography is at the heart of a more resilient and sustainable future. Creating responsible products and solutions drives our passion for improving quality of life everywhere.

Contact Esri

380 New York Street
Redlands, California 92373-8100 USA

1 800 447 9778
T 909 793 2853
F 909 793 5953
info@esri.com
esri.com

Offices worldwide
esri.com/locations

Printed in USA