Best Practices for Storing the Product Library Workspace in an Enterprise Geodatabase for SQL Server

An Esri® White 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 Product Library Workspace in an Enterprise Geodatabase for SQL Server

An Esri White Paper

Contents **Page**

J10022

Contents **Page**

Best Practices for Storing the Product Library Workspace in an Enterprise Geodatabase for SQL Server

Introduction

The product library is part of the Esri® Production Mapping extension. It is a geodatabase that allows multiuser environments to centralize information and behavior for cartographic and digital data production. Production business rules, documents, and spatial information are stored inside the product library, allowing an organization to enforce and standardize production. Data model information, data validation rules, geographic extents, symbology rules, and map documents can all be managed inside the product library as examples of production business rules. In other words, the product library is essentially a geographic document management system. When stored in an enterprise geodatabase, the workspace supports versioning. This white paper is intended to help database administrators establish the product library workspace in an enterprise geodatabase for SQL Server. The enterprise geodatabase uses ArcSDE® technology as the gateway between geographic information system (GIS) clients and SQL Server.

Overview of ArcSDE DBTUNE

DBTUNE storage parameters let you control how ArcSDE technology creates objects within a SQL Server database. You can determine such things as how to allocate space to a table or index, which Filegroup 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 the 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 the disk.

Thus, DBTUNE can be modified to store the product library tables in separate data files in different locations on the 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 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.

With data spread across multiple disks, more spindles 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 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 the 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 lets SQL Server perform tempdb operations in parallel with database operations.

- 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, you can consider keeping the transaction log files in a RAID 10 system and storing the 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/pt-br/library/bb742464(en-us).aspx.

- Use partitioning on large tables. Partitioning lets you split a table across multiple Filegroups by using partitions; 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

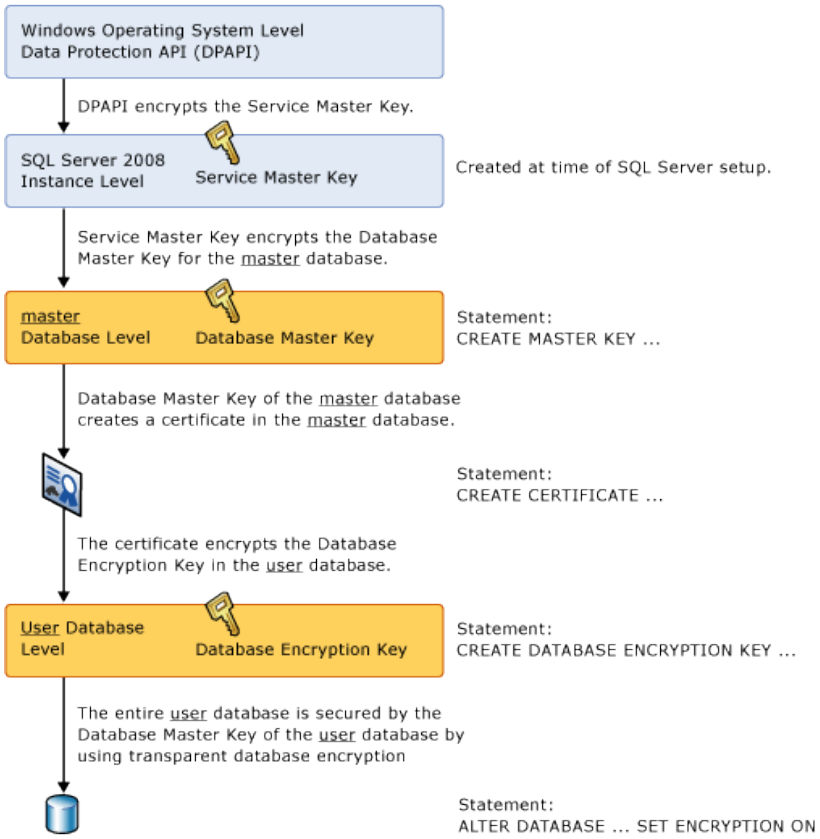
The precautions you can take to help secure the database include designing a secure system, encrypting confidential assets, and building a firewall around the database servers. However, if the physical media (drives or backup tapes) are stolen, a malicious party can just restore or attach the database and browse the data. One solution is to encrypt the 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 in an encrypted database are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database.

J10022

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.

J10022

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

/* Create Certificate */
CREATE CERTIFICATE prodlib_cert WITH SUBJECT = 'PRODLIB Server Certificate';
GO

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

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

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

/* Encrypt database */
ALTER DATABASE prodlibdb 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 ='prodlibdb'
GO
```

Step 1: Create Data Files

Create new Filegroups to store the product library feature classes and tables.

FILEGROUP	ArcSDE_PARAMETER
PRODLIB_BDATA	Business table
PRODLIB_BINDEX	Business table index
PRODLIB_FDATA	Feature table
PRODLIB_FINDEX	Feature table index
PRODLIB_SDATA	Spatial Index table
PRODLIB_SINDEX	Spatial Index table index
PRODLIB_ADATA	Adds table (versioned)
PRODLIB_AINDEX	Adds table index
PRODLIB_DDATA	Deletes table (versioned)
PRODLIB_DINDEX	Deletes table index

```
USE MASTER
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_BDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Bdata01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Bdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_BDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_BINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Bindex01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Bindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_BINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_FDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Fdata01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Fdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_FDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_FINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Findex01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Findex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_FINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_SDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Sdata01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Sdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_SDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_SINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Sindex01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Sindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_SINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_ADATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Adata01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Adata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_ADATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_AINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Aindex01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Aindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_AINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_DDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Ddata01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Ddata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_DDATA]
GO
ALTER DATABASE [PRODLIBDB] ADD FILEGROUP [PRODLIB_DINDEX]
GO
ALTER DATABASE [PRODLIBDB] ADD FILE(NAME = N'prodlib_Dindex01', FILENAME =
N'C:\mssql\data\prodlibdb\prodlib_Dindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[PRODLIB_DINDEX]
GO
```

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

```
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Bdata01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Bindex01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Fdata01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Findex01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Sdata01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Sindex01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Adata01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Aindex01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Ddata01', SIZE = 400MB )
ALTER DATABASE [PRODLIBDB] MODIFY FILE ( NAME = N'prodlib_Dindex01', SIZE = 400MB )
```

Verify Filegroups and data files:

```
EXEC sp_helpdb prodlibdb
GO
```

Step 2: Create the Prodlib User

Create a new database user to store the product library feature classes and tables; grant the appropriate permissions.

Create user and schema:

```
USE [prodlibdb]
GO
EXEC sp_addlogin N'prodlib', 'prodlib', @logindb, @loginlang
GO
CREATE USER [prodlib] FOR LOGIN [prodlib]
GO
CREATE SCHEMA [prodlib] AUTHORIZATION [prodlib]
GO
ALTER USER [prodlib] WITH DEFAULT_SCHEMA=[prodlib]
GO
```

Grant privileges:

```
USE [prodlibdb]
GO
EXEC sp_droprolemember 'gis_data_creator', 'prodlib'
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', 'prodlib'
GO
```

Verify roles:

```
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) = 'prodlib'
```

Associate login prodlib with user prodlib:

```
USE [prodlibdb]
GO
EXEC sp_change_users_login 'update_one','prodlib','prodlib'
GO
EXEC sp_helpuser 'prodlib'
GO
```

J10022

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:mcsdbsrv1 –D prodlibdb
```

Copy ***dbtune_exp.sde*** to ***dbtune_prodlib.sde***.

- Modify the ##DEFAULTS configuration keywords.

```
dbtune_prodlib.sde

##DEFAULTS
A_INDEX_RASTER "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_INDEX_ROWID  "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_INDEX_SHAPE  "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_INDEX_STATEID     "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_INDEX_USER "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_INDEX_XML  "WITH FILLFACTOR = 75 ON PRODLIB_AINDEX"
A_STORAGE    "ON PRODLIB_ADATA"
B_INDEX_RASTER "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_INDEX_ROWID  "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_INDEX_SHAPE  "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_INDEX_TO_DATE     "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_INDEX_USER "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_INDEX_XML  "WITH FILLFACTOR = 75 ON PRODLIB_BINDEX"
B_STORAGE    "ON PRODLIB_BDATA
D_INDEX_ALL  "WITH FILLFACTOR = 75 ON PRODLIB_DINDEX"
D_INDEX_DELETED_AT  "WITH FILLFACTOR = 75 "ON PRODLIB_DINDEX"
D_STORAGE    "ON PRODLIB_DDATA"
F_INDEX_AREA "WITH FILLFACTOR = 75 ON PRODLIB_FINDEX"
F_INDEX_FID  "WITH FILLFACTOR = 75 ON PRODLIB_FINDEX"
F_INDEX_LEN  "WITH FILLFACTOR = 75 ON PRODLIB_FINDEX"
F_STORAGE    "ON PRODLIB_FDATA"
GEOMETRY_STORAGE    "GEOMETRY"
GEOMTAB_PK   "WITH FILLFACTOR = 75 ON PRODLIB_FINDEX"
GEOMTAB_STORAGE     "ON PRODLIB_FDATA"
I_STORAGE    "ON PRODLIB_FDATA"
S_INDEX_ALL  "WITH FILLFACTOR = 75 ON PRODLIB_SINDEX"
S_INDEX_SP_FID "WITH FILLFACTOR = 75 ON PRODLIB_SINDEX"
S_STORAGE    "ON PRODLIB_SDATA"
END
```

If your database only stores the Product Library repository, you can edit ##DEFAULTS; otherwise, create a new configuration keyword.

- Import the modified ***dbtune_prodlib.sde*** file.

```
sdedbtune -o import -f dbtune_prodlib.sde -u sde -p sde -i sde:sqlserver:mcsdbsrv1 –D prodlibdb
```

Step 4: Create the Product Library Database Connection

Create a database connection in ArcCatalog™ with the prodlib user; this will be the product library workspace location.

Step 5: Configure SQL Server Parameters

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

SQL Server Parameters for Product Library

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 6: Configure ArcSDE Parameters

You need to 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 from 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 7: Create the CKB_USERS Role

A role needs to be assigned to the users who are going to be working with the product library so they can view or add components, information, and data. The role CKB_USERS must be created for the users to be recognized by the product library. This can be done by using the following statement:

```
USE [prodlibdb]
GO
EXEC sp_droprole 'ckb_users'
GO
EXEC sp_addrole 'ckb_users', 'prodlib'
GO
```

Step 8: Create the Product Library Workspace

After the geodatabase has been created, various tables and feature classes that are part of the product library need to be added to it. This process can be completed in ArcMap™.

The steps in this section are for defining and upgrading the geodatabase as a product library in ArcMap.

Steps

1. Start ArcMap.
2. On the menu bar, click **Customize > Production > Product Library**.

Tips:

- If the **Product Library** command is not enabled, you may need to enable the Production Mapping extension by clicking **Customize > Extensions** and checking the check box for **Production Mapping**.
- You can also open the window by clicking the **Product Library** button on the **Production Cartography** or **Production Editing** toolbar.

3. Right-click **Product Library** and click **Select Product Library**.

   The **Choose Product Library Workspace** dialog box appears.

4. Navigate to the product library database.
5. Click **Open**.

   The **Upgrade Product Library Workspace** or the **Create Product Library Workspace** dialog box appears if the geodatabase does not have all the components necessary to perform as a product library. Continue to step 6 if one of these dialog boxes appears.

Note: If you also have the Esri Nautical Solution installed, tables are added to the product library schema that are necessary for nautical production but do not impact nonnautical production.

If this is not a new product library, and if an error appears when you choose a geodatabase as the product library, it may need to be compacted or compressed to restore its integrity. Compacting applies to personal and file geodatabases, and compressing applies to enterprise geodatabases.

If you select an enterprise workspace, the **Select Product Library Owner** dialog box appears, and you can continue to step 7.

6. Perform the following steps to upgrade or create the geodatabase you want to use as the product library:

   a. If necessary, click the drop-down arrow in the **Select Configuration Keyword** area and choose an option.

      Options vary based on whether the database is a personal, file, or ArcSDE geodatabase.

   b. Click **OK**.

   c. Click **OK** once the upgrade or creation completes.

7. If the product library is an enterprise geodatabase, click the owner and click **OK**.

   The **Upgrade Product Library Workspace** or the **Create Product Library Workspace** dialog box appears if the geodatabase does not have all the components necessary to perform as a product library. Perform the following steps if you are an administrator and want to upgrade or create the geodatabase to use as the product library.

   a. If necessary, click the drop-down arrow in the Select Configuration Keyword area and choose an option.

      Options vary based on whether the database is a personal, file, or ArcSDE geodatabase.

   b. Click **OK**.

   c. If necessary, click **OK** once the upgrade or creation completes.

Step 9: Verify the Storage

Run the SQL queries below to verify that the product library workspace was created under the correct Filegroups.

```
USE [prodlibdb]
GO
```

List Filegroups and data files:

```
EXEC sp_helpdb prodlibdb
GO
```

List Filegroup data files:

```
USE [prodlibdb]
GO
```

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

List tables by Filegroup:

```
USE [prodlibdb]
GO
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) = 'prodlib'
 ORDER BY 1,3,2
GO
```

List indexes by table and Filegroup:

```
USE [prodlibdb]
GO
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) = 'prodlib'
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 SQL Server Books Online 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 Filegroups. This is particularly useful when tables are empty and you are allowed to re-create database objects.

Step 10: Register as Versioned

If you manually import the product library tables and feature classes, such as importing an XML file, you need to verify that all the tables and feature classes are registered as versioned. This allows the software to edit the tables as you create and work with your product library. However, you shouldn't create new versions of the product library tables; this can result in inconsistencies within the versions.

Steps

1. Expand **Database Connections** in the **Catalog Tree** window.
2. Double-click the product library administrator connection geodatabase to connect to it.
3. Right-click each feature class in your product library and click **Manage > Register as Versioned**.

   Do not check the **Register the selected objects with the option to move edits to base** check box.

4. Click **OK**.

5. Right-click each table in your product library, except for those listed below, and click **Manage > Register as Versioned**.

   Do not check the **Register the selected objects with the option to move edits to base** check box.

   Do not register the following tables for versioning:

   - ELM_CATEGORIES
   - ELM_ELEMENTS
   - ELM_PRODUCTS
   - ELM_SOLUTIONS

6. Click **OK**.

Step 11: 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.

You can learn more about heaps 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. You 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 prodlib.TableA table of the prodlibdb database by using row compression, use the following query:

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

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

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

You can learn more about row-level compression 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.

You can learn more about Unicode compression 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 compressed. Data that involves many repeating values will be more compressed than data populated by more unique values. You 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 prodlib.TableA table of the prodlibdb database by using page compression, use the following query:

```
USE [prodlibdb]
ALTER TABLE [prodlib].[TableA] REBUILD PARTITION = ALL
```

```
WITH
(DATA_COMPRESSION = PAGE)
```

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

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

You can learn more about page-level compression 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.

You can learn more about data compression 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.

You can learn more about the Data Compression Wizard 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 prodlib.TableA table in the prodlibdb database, execute the following Transact-SQL statement:

```
USE prodlibdb;
GO
EXEC sp_estimate_data_compression_savings 'prodlib', '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 prodlibdb;
GO
EXEC sp_estimate_data_compression_savings 'prodlib', 'TableA', NULL,
NULL,
'PAGE';
GO
```

You can learn more about how to estimate compression savings at http://msdn.microsoft.com/en-us/library/cc280574(v=sql.110).aspx.

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

```
/*---------------------------------------------------------------------
-- Verify PRODLIB Schema Storage
---------------------------------------------------------------------*/
USE [prodlibdb]
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) = 'prodlib'
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) = 'prodlib'
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 PRODLIB.PL_NEWCLASS_AOI
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) = 'prodlib'
AND o.type <> 'S' and o.type = 'U'
ORDER BY o.name,o.type;
GO

/*-----------------------------------------------------------------------
--Index page compression
--Example:
/*
ALTER INDEX R222_pk
ON PRODLIB.PL_NEWCLASS_AOI
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) = 'prodlib'
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 in a database 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. Therefore, using TDE and backup compression together is not recommended.

SQL Server 2012—Data Compression

Step 12: Validate Permissions and Roles

All the tables in the product library need to have read/write privileges assigned to them except the PCAT_PERMISSION table, which only needs read privileges assigned to it. The permissions need to be assigned to the CKB_USERS role. You can re-create the CKB_USERS role and grant the right permissions to the role by using the following script:

```
/*RECREATE ckb_users role */
USE [prodlibdb]
GO
EXEC sp_droprolemember 'ckb_users', 'prodlibuser'
GO
```

J10022

```
EXEC sp_droprole 'ckb_users'
GO
EXEC sp_addrole 'ckb_users', 'prodlib'
GO
EXEC sp_addrolemember 'ckb_users', 'prodlibuser'
GO
DECLARE @OWNER varchar(10)
SET @OWNER = 'PRODLIB'
DECLARE Tables_Cursor CURSOR
READ_ONLY
FOR SELECT a.name as table_name, a.xtype as type
    FROM dbo.sysobjects a, dbo.sysusers b
    WHERE a.uid = b.uid and a.xtype in ('U','P') and b.name = @OWNER ORDER BY a.name
DECLARE @name varchar(100), @type varchar(1)
OPEN Tables_Cursor
FETCH NEXT FROM Tables_Cursor INTO @name, @type
WHILE (@@fetch_status <> -1)
BEGIN
  IF (@@fetch_status <> -2)
   BEGIN
      --PRINT @owner + '.' + @name + ' ' + @type
      -- GRANT PERMISSIONS TO TABLE
    IF @type = 'U'
      BEGIN
      --EXECUTE ('GRANT SELECT ON ' + @OWNER + '.' + @name + ' TO pl_user')
      EXECUTE ('GRANT SELECT, INSERT, UPDATE, DELETE ON ' + @OWNER + '.' + @name + ' TO ckb_users')
    END
      ELSE
        --GRANT PERMISSION TO STORE PROCEDURE
        IF @type = 'P'
        BEGIN
          EXECUTE ('GRANT EXEC ON ' + @OWNER + '.' + @name + ' TO ckb_users')
        END
  END
  FETCH NEXT FROM Tables_Cursor INTO @name, @type
END
CLOSE Tables_Cursor
DEALLOCATE Tables_Cursor
GO
REVOKE INSERT, UPDATE, DELETE ON PRODLIB.PCAT_PERMISSION FROM ckb_users;
GO
```

Grant Permissions Using ArcCatalog

Both the administrator and other user accounts in the underlying database management system should have appropriate privileges and roles assigned to them. When you set up your connection to your spatial database, ensure that you are connecting as the appropriate user.

Steps

1. Start ArcCatalog.
2. Expand **Database Connections** in the **Catalog Tree** window.
3. Double-click the product library administrator connection geodatabase to connect to it.
4. Select all tables except PCAT_PERMISSION, right-click, then click **Manage** > **Privileges**.
5. Type CKB_USERS into the **User** text box on the **Change Privileges** dialog box.
6. Click the **View (Select)** drop-down arrow and choose **GRANT**.
7. Click the **Edit (Update/Insert/Delete)** drop-down arrow and choose **GRANT**.
8. Click **OK**.
9. Select the **PCAT_PERMISSION** table, right-click, then click **Privileges**.
10. Type CKB_USERS into the **User** text box on the **Change Privileges** dialog box.

11. Click the **View** drop-down arrow and choose **GRANT**.

12. Click **OK**.

Step 13: 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® software, 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 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.

The DBTUNE SESSION_TEMP_TABLE parameter must be set to 1 (true) to allow the session-based log file table to be created in tempdb. 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.

Learn more about ArcSDE log file tables at resources.arcgis.com/en/help/main/10.2/index.html#/What_are_ArcSDE_log_file_tables/002q00000011000000/.

Step 14: Create the ArcSDE Product Library User

The example below shows how to create an ArcSDE user to access the product library:

```
USE master
GO
EXEC sp_addlogin N'prodlibuser', 'prodlibuser', @logindb, @loginlang
GO
```

Create user:

```
USE [prodlibdb]
GO
CREATE USER [prodlibuser] FOR LOGIN [prodlibuser]
GO
```

Grant privileges:

```
USE [prodlibdb]
GO
EXEC sp_addrolemember N'ckb_users', N'prodlibuser'
GO
```

Verify user permissions:

```
USE [prodlibdb]
GO
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) = 'prodlibuser'
GO
```

Step 15: Create Database Connections for Product Library Users

Database connections need to be created for the other product library users if the product library is stored in SQL Server. Create a database connection in ArcCatalog with the PRODLIBUSER user; this will be the product library ArcSDE connection.

Note: You can create a login for each user in the product library database and, if using database authentication, type the user name of the product library user for whom you want to create a database connection.

Step 16: Assign Product Library Permissions

There are two different levels of interaction with the product library in an ArcSDE implementation of the product library: administrators and users. These levels of access are controlled through ArcCatalog database connections. The administrators manage the overall product library including the structure, components, and user permissions. This level of permissions through ArcCatalog database connections is related to the database role CKB_USERS.

Users can have varying degrees of access to parts of the product library based on whether they have edit, read/write, or read-only permissions based on their Windows login. Using the administrator's database connection, user accounts are created for anyone who is going to have access to the product library. To create a new user, first add the person as a product library user, then assign permissions.

Learn more about product library permissions at resources.arcgis.com/en/help/main/10.2/index.html#/Product_library_permissions/010300000043000000/.

Step 17: Add New Users to the Product Library

Using the administrator's database connection, user accounts are created for anyone who is going to have access to the product library. To create a new user, the person must first be added as a product library user, then permissions can be assigned.

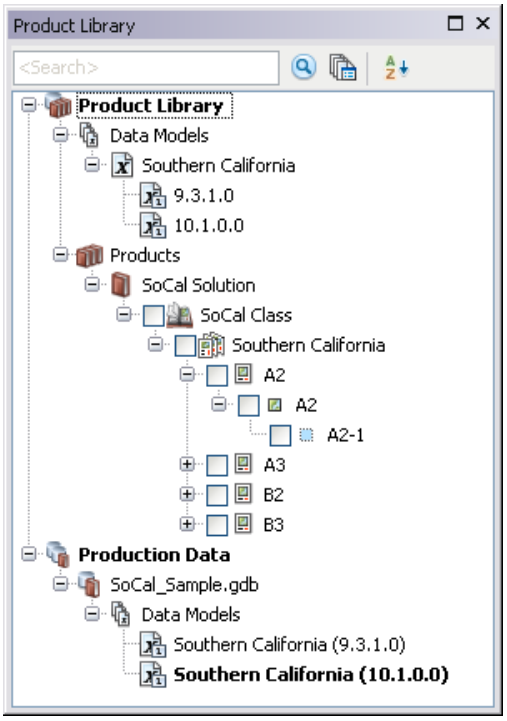
Initially, the user must be added to the geodatabase by the administrator. Each user is added using the first name, last name, and Windows user name.

Note: This only applies if you are using an ArcSDE geodatabase as your product library. Personal and file geodatabase permissions are defined by the user's permissions at the operating system level.

Steps

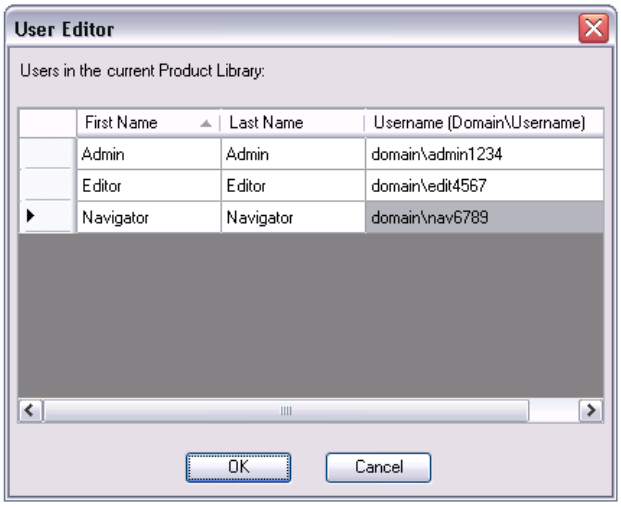
1. Start ArcMap.
2. If necessary, open the **Product Library** window by clicking **Customize > Production > Product Library** on the main menu.

A tree view of the product library appears.



3. If necessary, define the product library workspace.
4. Right-click **Product Library** and click **Configure > Users**.

   The **User Editor** dialog box appears.



Tip: If you are using an ArcSDE geodatabase as your product library, you can also add new users to the product library by right-clicking a series and clicking **Permissions**.

5. Right-click anywhere in the **Users in the current Product Library** list and click **New User**.

   A new row appears in the list.

Tip: If you are using an ArcSDE geodatabase as your product library, you can also add users when you are assigning permissions to existing users.

6. Type the user's first name in the **First Name** cell.
7. Type the user's last name in the **Last Name** cell.
8. Type the user's Windows login name in the **Username (Domain\Username)** cell.
9. Repeat steps 5 through 8 for each user you need to add to the product library.
10. Click **OK**.

If the user name(s) is valid, the user(s) is added to the product library.

Assigning Permissions to Users

Once the user is added, the permissions can be granted at the series level of the product library. Permissions are passed down to all products within a given series. Permissions are also passed up from the series to the class and the solution. For example, if a user is given permission to one or more series below a particular class or solution, the user has access to those entries. By default, the permissions for a user are set to Not Available, but there are four different levels:

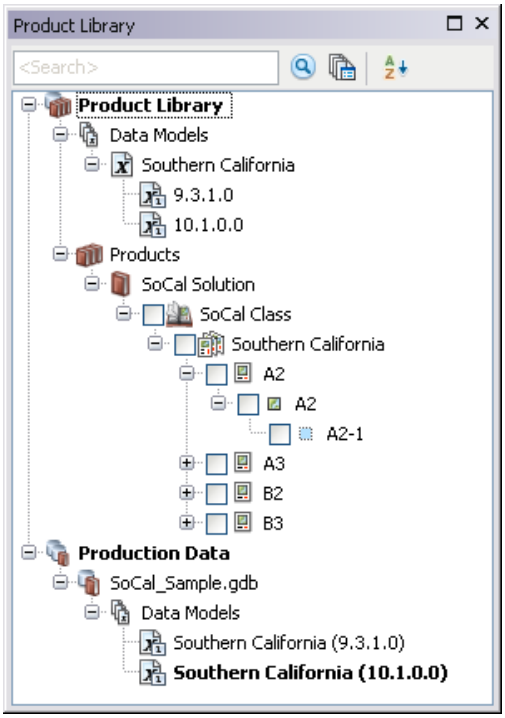
- Not Available—The series and all components beneath it are hidden from the user.
- Read Only—Properties can be viewed for all levels of the product library.
- Check In/Check Out—Files can be checked in and out.
- Edit—Product library levels can be added, modified, and removed, and files can be checked in and out.

Note: This only applies if you are using an ArcSDE geodatabase as your product library. Personal and file geodatabase permissions are defined by the user's permissions at the operating system level.

Steps

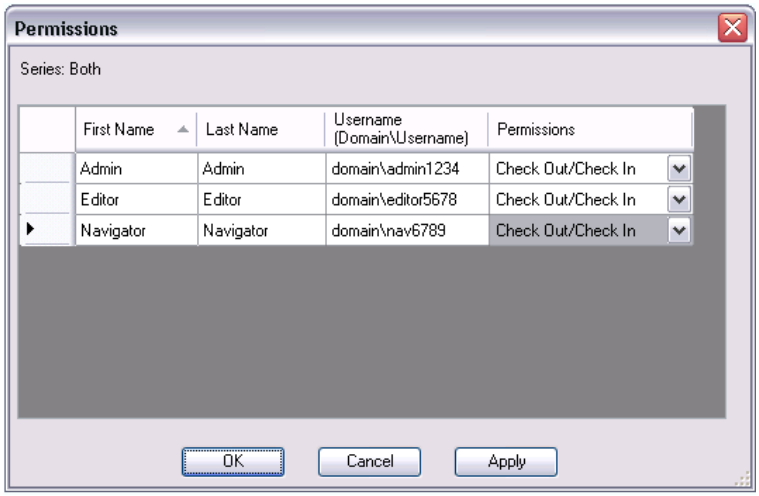
1. Start ArcMap.
2. If necessary, open the **Product Library** window by clicking **Customize > Production > Product Library** on the main menu.

A tree view of the product library appears.



3. If necessary, expand **Product Library** and **Products**.
4. Navigate to the series level of the product class for which you want to assign permissions.
5. Right-click the series name and click **Permissions**.

   The **Permissions** dialog box appears.



J10022

6. Click the **Permissions** drop-down arrow for the user to whom you want to grant permissions and choose an option.

7. Repeat step 6 for all users to whom you want to give permissions.

8. Click **OK**.

Replication

You can use geodatabase replication to replicate the Product Library workspace (one-way, two-way replication), but you can create a *new* solution, product class, series, or product only in the parent geodatabase *or* only in the child geodatabase.

Conclusion

You can reduce disk contention and improve database I/O by storing the product library workspace in different locations on the 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:
resources.arcgis.com/en/help/main/10.2/index.html#/Minimize_disk_I_O_contention_in_SQL_Server/002q00000021000000/

For more information on the product library, visit the Esri Production Mapping page:
esri.com/software/arcgis/extensions/production-mapping/index.html

Learn about setting up the product library in an ArcSDE environment:
resources.arcgis.com/en/help/main/10.2/index.html#/in_a_geodatabase_in_SQL_Server_Express/010300000299000000/

Access blogs, forums, downloads, and more, via the Esri Production Mapping resource center:
resources.arcgis.com/en/communities/production-mapping/

You can access other resources at ArcGIS 10.2 for Desktop Help:
resources.arcgis.com/en/help/main/10.2/#/Welcome_to_the_ArcGIS_Professional_Help_Library/00qn0000001p000000/ and Esri Support (support.esri.com).

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