Best Practices for Storing the GIS Data ReViewer Workspace in an Enterprise Geodatabase for SQL Server®

ESRI 380 New York St., Redlands, CA 92373-8100 USA
TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

J-9786

Best Practices for Storing the GIS Data ReViewer Workspace in an Enterprise Geodatabase for SQL Server

An ESRI White Paper

Contents **Page**

J-9786

Contents **Page**

Best Practices for Storing the GIS Data ReViewer Workspace in an Enterprise Geodatabase for SQL Server

Introduction

GIS Data ReViewer, a data quality control management application, requires that you have a workspace defined when using a review session to store the anomalies found during your data review process. It creates a new feature dataset and tables in the geodatabase that you have identified as your ReViewer workspace. The feature dataset includes point, line, and polygon feature classes, which contain error geometries for the features that have been written to the ReViewer table. The ReViewer workspace can be stored in any existing geodatabase, including the production geodatabase, or a separate file or personal geodatabase created specifically for the data review process. When stored in an enterprise geodatabase, the ReViewer workspace supports versioning. This white paper is intended for database administrators to help them establish the GIS Data ReViewer workspace in an enterprise geodatabase for SQL Server®. The enterprise geodatabase uses ArcSDE® technology as the gateway between GIS clients and SQL Server.

DBTUNE

DBTUNE storage parameters allow you to 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 during the setup phase that follows the installation of ArcSDE technology. ArcSDE installation creates a dbtune file under the etc directory from which the DBTUNE table is populated. If no dbtune file is present during setup, the DBTUNE table will be populated with default values.

If a large number of database connections are accessing the same files in the same location on disk, database performance will be slow 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 ReViewer feature dataset and tables into separate data files across different locations on disk. This will lead to reduced disk contention and improved database I/O.

Standard geographic information system (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.

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 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 0
- 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

Reduce 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 help eliminate 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 same drive that contains the operating system files.

- Put transaction log files on a separate drive from data files. This gives 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 transaction log files in a RAID 10 system and storing data and index files in a RAID 5 system.

J-9786

For more information about RAID, see RAID Levels and SQL Server at http://msdn2.microsoft.com/ms190764.aspx and Microsoft® Windows® 2000 Server Administrator's Companion (Microsoft Press), chapter 7, "Planning Fault Tolerance and Avoidance," by Charlie Russel and Sharon Crawford, at http://www.microsoft.com/technet/prodtechnol/windows2000serv/plan/planning.mspx.

- 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 enable you to 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

Step 1: Data Files

Create new filegroups to store the ReViewer feature dataset and tables.

FILEGROUP	*SDE_PARAMETER*
REV_BDATA	Business table
REV_BINDEX	Business table index
REV_FDATA	Feature table
REV_FINDEX	Feature table index
REV_SDATA	Spatial Index table
REV_SINDEX	Spatial Index table index
REV_ADATA	Adds table (versioned)
REV_AINDEX	Adds table index
REV_DDATA	Deletes table (versioned)
REV_DINDEX	Deletes table index

```
USE MASTER
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_BDATA]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Bdata01', FILENAME =
N'C:\mssql\data\rev\rev_Bdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_BDATA]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_BINDEX]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Bindex01', FILENAME =
N'C:\mssql\data\rev\rev_Bindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_BINDEX]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_FDATA]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Fdata01', FILENAME =
N'C:\mssql\data\rev\rev_Fdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_FDATA]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_FINDEX]
GO
```

```
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Findex01', FILENAME =
N'C:\mssql\data\rev\rev_Findex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_FINDEX]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_SDATA]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Sdata01', FILENAME =
N'C:\mssql\data\rev\rev_Sdata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_SDATA]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_SINDEX]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev_Sindex01', FILENAME =
N'C:\mssql\data\rev\rev_Sindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_SINDEX]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_ADATA]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev__Adata01', FILENAME =
N'C:\mssql\data\rev\rev_Adata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_ADATA]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_AINDEX]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev__Aindex01', FILENAME =
N'C:\mssql\data\rev\rev_Aindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_AINDEX]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_DDATA]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev__Ddata01', FILENAME =
N'C:\mssql\data\rev\rev_Ddata01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_DDATA]
GO
ALTER DATABASE [REV] ADD FILEGROUP [REV_DINDEX]
GO
ALTER DATABASE [REV] ADD FILE(NAME = N'rev__Dindex01', FILENAME =
N'C:\mssql\data\rev\rev_Dindex01.NDF' , SIZE = 1, MAXSIZE = 800, FILEGROWTH = 1) TO FILEGROUP
[REV_DINDEX]
GO
```

Verify filegroups and data files:

```
EXEC sp_helpdb rev
GO
```

Step 2: REV User

Create a new database user to store the ReViewer feature dataset and tables; grant the appropriate permissions.

Create user and schema:

```
USE REV
GO
CREATE USER [rev] FOR LOGIN [rev]
GO
CREATE SCHEMA [rev] AUTHORIZATION [rev]
GO
ALTER USER [rev] WITH DEFAULT_SCHEMA=[rev]
GO
```

Grant privileges:

```
USE REV
GO
EXEC sp_droprolemember 'gis_data_creator', 'rev'
```

J-9786

```
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', 'rev'
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) = 'rev'
```

Associate Login rev with User rev:

```
USE REV
GO
EXEC sp_change_users_login 'update_one','rev','rev'
GO
EXEC sp_helpuser 'rev'
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 5151 –D rev

GIS Data ReViewer 9.3

Copy ***dbtune_exp.sde*** to ***dbtune_rev.sde***.

Modify the ***dbtune_rev.sde*** *##DEFAULTS* configuration keyword.

J-9786

dbtune_rev.sde

```
##DEFAULTS
A_INDEX_RASTER"WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_ROWID "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_SHAPE "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_STATEID "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_USER "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_XML "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_STORAGE "ON REV_ADATA"
AUX_INDEX_COMPOSITE "WITH FILLFACTOR = 75 ON GIS_RASTERIDX"
AUX_STORAGE "ON GIS_RASTER"
B_INDEX_RASTER "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_INDEX_ROWID "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_INDEX_SHAPE "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_INDEX_TO_DATE "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_INDEX_USER "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_INDEX_XML "WITH FILLFACTOR = 75 ON REV_BINDEX"
B_STORAGE "ON REV_BDATA"
BLK_INDEX_COMPOSITE "WITH FILLFACTOR = 75 ON GIS_RASTERIDX"
BLK_STORAGE "ON GIS_RASTER"
BND_INDEX_COMPOSITE"WITH FILLFACTOR = 75 ON GIS_RASTERIDX"
BND_INDEX_ID "WITH FILLFACTOR = 75 ON GIS_RASTERIDX"
BND_STORAGE "ON GIS_RASTER"
D_INDEX_ALL "WITH FILLFACTOR = 75 ON REV_DINDEX"
D_INDEX_DELETED_AT "WITH FILLFACTOR = 75 ON REV_DINDEX"
D_STORAGE "ON REV_DDATA"
F_INDEX_AREA "WITH FILLFACTOR = 75 ON REV_FINDEX"
F_INDEX_FID "WITH FILLFACTOR = 75 ON REV_FINDEX"
F_INDEX_LEN "WITH FILLFACTOR = 75 ON REV_FINDEX"
F_STORAGE "ON REV_FDATA"
GEOMETRY_STORAGE "SDEBINARY"
GEOMTAB_PK "WITH FILLFACTOR = 75"
RAS_INDEX_ID "WITH FILLFACTOR = 75 ON GIS_RASTERIDX"
RAS_STORAGE "ON GIS_RASTER"
S_INDEX_ALL "WITH FILLFACTOR = 75 ON REV_SINDEX"
S_INDEX_SP_FID "WITH FILLFACTOR = 75 ON REV_SINDEX"
S_STORAGE "ON REV_SDATA"
END
```

Import the modified ***dbtune_rev.sde*** file.

```
sdedbtune -o import -f dbtune_rev.sde -u sde -p sde -i 5151 –D rev
```

GIS Data ReViewer 9.3.1

From the GIS Data ReViewer 9.3.1 release forward, you can select the DBTUNE configuration keyword on the *ReViewer Workspace Properties* dialog box.

Instead of modifying ##DEFAULTS, you can simply create a new DBTUNE keyword following the steps below.

- Copy the ##DEFAULTS keyword and paste it at the end of the *dbtune_rev.sde* file.
- Rename it ##REVIEWER and change the filegroup name for the appropriate parameters.

```
##REVIEWER
A_INDEX_RASTER "WITH FILLFACTOR = 75 ON REV_AINDEX"
A_INDEX_ROWID "WITH FILLFACTOR = 75 ON REV_AINDEX"
...
END
```

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

J-9786

Step 4: ReViewer Workspace

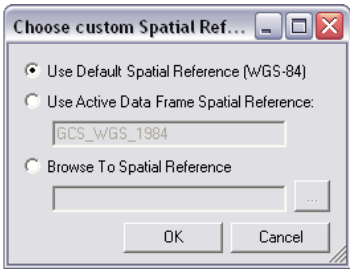
Create a database connection in ArcCatalog™ with the REV user; this will be the ReViewer workspace location.

GIS Data ReViewer 9.3

In ArcMap™, click the ReViewer Session Manager button on the PLTS™ GIS Data ReViewer toolbar.

- Browse to the ReViewer workspace location.
- Navigate to the database connection created in ArcCatalog.
- Click New to start a new ReViewer session.

The *Choose custom Spatial Reference* dialog box appears.



Choose custom Spatial Reference

- Choose the custom spatial reference.

  When you click *Browse To Spatial Reference*, the New Spatial Reference wizard appears and allows you to choose the spatial reference you want to use with the ReViewer feature dataset.

- Click OK.

  This automatically creates the ReViewer feature dataset and tables.

- The ID and Name fields are automatically populated in the Session area.

- If necessary, type a custom name for the current ReViewer session in the Name text box.

  By default, the name matches the ID.

- If you are working with an enterprise database, click the ReViewer Dataset Version drop-down arrow and choose the version to be used.

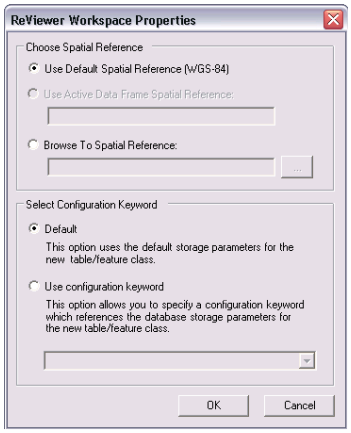
- Click Start Session.

  The button appears to be selected, and its name changes to End Session.

- Click Close.

GIS Data ReViewer 9.3.1

When you click New to start a new session in GIS Data ReViewer 9.3.1, the *Reviewer Workspace Properties* dialog box appears, and you can select the DBTUNE configuration keyword. Select the ##REVIEWER configuration keyword created in step 3.



ReViewer Workspace Properties

Step 5: Set the Feature Dataset and Tables as Versioned

After the new session is created, the REV user must register the following as versioned in ArcCatalog:

- RevDataset (feature dataset)
- RevTableMain table
- RevTableLocation table

Step 6: Verify Storage

Run the SQL queries below to verify if the ReViewer dataset was created under the correct filegroups.

```
USE REV
GO
```

List filegroups and data files:

```
exec sp_helpdb rev
go
```

List filegroup data files:

```
exec sp_helpfilegroup 'PRIMARY'
go
```

List tables by filegroup:

```
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) = 'rev'
 ORDER BY 1,3,2
GO
```

J-9786

List indexes by table and filegroup:

```
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) = 'rev'
order by object_name(i.id),i.name,f.name
go
```

If any tables or indexes are stored in the wrong filegroup, then ALTER TABLE and ALTER INDEX can be used to change the filegroup (see SQL Server Books Online for more details).

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 7: Restore DBTUNE

Once the ReViewer feature dataset and tables are created, you can restore the original DBTUNE parameters.

GIS Data ReViewer 9.3

Import the original dbtune file.

*sdedbtune -o import -f **dbtune_exp.sde** -u sde -p sde -i 5151 –D rev*

GIS Data ReViewer 9.3.1

Skip this step if you have created the *##REVIEWER* DBTUNE configuration keyword.

Step 8: Permissions and Roles

Grant permissions to the REV tables through the schema.

```
USE [rev]
GO
GRANT DELETE ON SCHEMA::[jtx] TO [giseditor]
GO
GRANT EXECUTE ON SCHEMA::[jtx] TO [giseditor]
GO
GRANT INSERT ON SCHEMA::[jtx] TO [giseditor]
GO
GRANT SELECT ON SCHEMA::[jtx] TO [giseditor]
GO
GRANT UPDATE ON SCHEMA::[jtx] TO [giseditor]
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) = 'giseditor'
```

Tips for Granting Permissions

Grant a user only select, update, insert, and delete privileges to the RevAdminCustomFields table if the user has privileges to modify the database schema.

Grant a user only select, update, insert, and delete privileges to the RevAdminDescriptions table if you want the user to be able to add customized error descriptions. Note that all other users accessing the ReViewer workspace will see the customized error descriptions.

Step 9: 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 are greater than 100 for each connected SDE 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.

You can find out more about ArcSDE log file tables by visiting the ArcGIS® Desktop 9.3 Help under Geodatabases and ArcSDE > Administering ArcSDE geodatabases > Configuring an ArcSDE geodatabase/Log file configuration options.

Step 10: ArcSDE User

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

Editor User

```
USE master
GO
EXEC sp_addlogin N'giseditor', 'gis$editor', @logindb, @loginlang
GO
```

J-9786

Create user:

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

Grant privileges:

```
USE [rev]
GO
GRANT DELETE ON SCHEMA::[rev] TO [giseditor]
GO
GRANT EXECUTE ON SCHEMA::[rev] TO [giseditor]
GO
GRANT INSERT ON SCHEMA::[rev] TO [giseditor]
GO
GRANT SELECT ON SCHEMA::[rev] TO [giseditor]
GO
GRANT UPDATE ON SCHEMA::[rev] TO [giseditor]
GO
```

Viewer User

```
USE master
GO
EXEC sp_addlogin N'gisviewer', 'gis$viewer', @logindb, @loginlang
GO
```

Create user:

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

Grant privileges:

```
USE [rev]
GO
GRANT SELECT ON SCHEMA::[rev] TO [gisviewer]
GO
```

Conclusion

You can reduce disk contention and improve database I/O by storing the GIS Data ReViewer workspace in different locations on disk. However, this practice alone does not guarantee optimal database performance, and additional tuning tasks may need to be taken into consideration.

You can find out more about the recommended tuning tasks in the ArcGIS Desktop 9.3 Help under Geodatabases and ArcSDE > Administering ArcSDE geodatabases > Tuning an ArcSDE geodatabase.

ArcGIS Desktop 9.3 Help (http://webhelp.esri.com/arcgisdesktop/9.3)
or visit www.esri.com/support.

For more information on GIS Data ReViewer, visit www.esri.com/datareviewer or e-mail datareviewer@esri.com.

About ESRI

For four decades, ESRI has been helping people make better decisions through management and analysis of geographic information. Our culturally diverse staff work with our business partners and hundreds of thousands of people who use GIS to make a difference in our world.

A full-service GIS company, ESRI offers support for implementing GIS technology from the desktop to enterprise-wide servers, online services, and mobile devices. GIS solutions are flexible and customizable to meet the needs of all our users.

Our Focus

At ESRI, we focus on promoting the value of GIS and its applications throughout the world and pay close attention to our users' needs. Our software development and services respond to our customers with products that are easy to use, flexible, and integrated. Our technology is multidisciplinary, productive, and valuable to our users.

We have a strong commitment to educating our customers through ESRI's various training programs. ESRI is a socially conscious business and invests heavily in issues regarding education, conservation, sustainable development, and humanitarian affairs.

Contact ESRI

1-800-GIS-XPRT (1-800-447-9778)
Phone: 909-793-2853
Fax: 909-793-5953
info@esri.com
www.esri.com

Offices worldwide
www.esri.com/locations

ESRI
380 New York Street
Redlands, California
92373-8100 USA

GSA Contract Holder
Contract GS-35F-5086H

Printed in USA