**BG-BASE Connector Project**

Table of Contents

1. Overview 2

2. MS SQL Server 2008 2

2.1. Alter X\_COORD, Y\_COORD and Z\_COORD field type 2

2.2 Warehouse Database Enable CDC 2

2.1.1 SELECT statements 3

2.1.2 CDC SYNC TABLES 3

2.2 SQL Server Agent 4

2.3. Create Multiversion view 4

3. ArcGIS configuration 5

3.1 Creating Feature Class 5

3.2. Python scripts 5

3.2.1 Log files 5

f. XML File- disconnected synchronization 5

2. Windows Task Scheduler 6

4. Warehouse to SDE 8

a. Schematic 8

Multiple Users 9

Compress in ArcCatalog 9

GUIDs 10

5. SDE to Warehouse 11

6. ArcGIS Tools 12

7. ArcGIS Server Admin Notes 12

References 17

Appendix 1: SQL Server System Configuration 17

Appendix 2: Cleaning CDC Tables in SQL 18

Appendix 3: Troubleshooting 19

# 1. Overview

In 2005, BG-BASE and the Arboretum entered into an agreement to get data out of BG-BASE, and into a relational database that could be used with ArcGIS tools. MS SQL Server was chosen.

In 2011, Putnam Fellow Brian Morgan, created the ArcGIS services that became the basis of web applications.

# 2. BG-BASE

https://github.com/arnarb/BG-Connector

https://github.com/arnarb/aanav/blob/master/doc/bgbase.md

The SQL bond is built atop OLEDB/ADO. Because those technologies work with Oracle the bond should work with Oracle. That said, I have not tested it with Oracle yet. I have tested with MSSQL, MySQL and PostGres.

The SQL tables I believe you are working with came about as a result of our having to create separate tables from data that we store in OpenInsight (our database manager) as multi-value (MV) fields.  While these MV fields work great for us in BG-BASE, they do not translate natively to the SQL world.  We therefore had to create a separate table for each one of our MV fields.  This is certainly not the schema we would have designed had we been working in native SQL!  But, it is what it is, and it \*does\* make the data available in the SQL environment.

In general, any table that contains an underscore in the table name will have a many-to-one relationship with a parent table.

For any tables related to PLANTS (such as PLANTS\_CONDITION), the key will \*always\* be ACC\_NUM\_AND\_QUAL

For any tables related to ACCESSIONS (such as ACCESSIONS\_ASSOCIATED\_TAXA), the key will \*always\* be ACC\_NUM

For any tables related to NAMES (such as NAMES\_ALT\_NAME), the key will \*always\* be NAME\_NUM

Slight disclaimer, we also have a few parent tables that \*already\* have underscores in them (such as HORT\_TASKS).

One other potentially important piece of information in the SQL data is line\_seq.  For example, let's say you have 5 records in the PLANTS\_CONDITION table with the same ACC\_NUM\_AND\_QUAL value.  The record with line\_seq = 1 would be the most recent observation, while records with line\_seq 2-5 would be historical observations.

Currently, *rep\_id* is set as the primary key for each table.

Each table has a combined *ACC\_NUM\_AND\_QUAL* field AND a *NAME\_NUM* field.  These fields would be ideal choices for building the required relationships between the 276 tables.

MV Table                 Key to Parent

ACCESSIONS\_*table*ACC\_NUM

CONTACT*\_table*         CONTACT\_NUM

GENERA\_*table*             GENUS

HORT\_TASKS\_*table*         CODE

NAMES\_*table*NAME\_NUM

PLANTS\_*table* ACC\_NUM\_AND\_QUAL

PROPAGATIONS\_*table* PROP\_NUM

PSOURCES\_table           PS\_NUM

SHIPMENTS\_*table* SHIPMENT\_NUM

SPECIMENS\_*table* SPECIMEN\_NUM

VERIFICATIONS\_*table* VERIF\_NUM

PLANTS has a many-to-one relationship with ACCESSIONS, with ACC\_NUM and ACC\_NUM\_QUAL as a two-part key; ACC\_NUM\_AND\_QUAL is actually a symbolic field we created to concatenate the two.

ACCESSIONS then has a many-to-one relationship with NAMES, with NAME\_NUM as the relational key into NAMES

NAMES then has a many-to-one relationship GENERA, with GENUS as the relational key into GENERA

GENERA then has a many-to-one relationship with FAMILIES, with FAMILY as the relational key into FAMILIES.

# 3. MS SQL Server 2008

(see [Appendix 1](#_Appendix_1:_System) for Server details)

The following are steps in order, to set up the SQL Server to work as a Connector.

## 3.1. Alter X\_COORD, Y\_COORD and Z\_COORD field type

If you use SQL Server spatial types (geometry) you could create the point from the x,y on the fly as well during the insert statement. The multiversioned view will allow you to do the update into the versioned gdb through sql.

USE *DBNAME*

ALTER TABLE PLANTS\_LOCATION

ALTER COLUMN Z\_COORD (X\_COORD, Y\_COORD) numeric (16,8) null

GO

## 3.2 Warehouse Database Enable CDC

There are several steps to follow in order to create tables that are editable by ArcGIS. The first step involves enabling CDC (Change Detection Capture) on a database.[1,2](#_References)

Enable database:

USE **databasename**

GO

EXEC sys.sp\_cdc\_enable\_db

GO

If an error message is encountered, you can try:[7,8](#_References_1)

USE **databasename**

GO

EXEC sp\_changedbowner 'sa'

GO

Enable table(s):

USE **databasename**

GO

EXEC sys.sp\_cdc\_enable\_table

@source\_schema = N'dbo',

@source\_name = N'**TABLE\_NAME’**,

@role\_name = N'CDC\_admin'

GO

PLANTS table: when enable cdc on table get this message:

Job 'cdc.Production\_capture' started successfully.

Job 'cdc.Production\_cleanup' started successfully.

Note: @role\_name. If there is any restriction of how data should be extracted from database, this option is used to specify any role which is following restrictions and gating access to data to this option if there is one.  If you do not specify any role and, instead, pass a NULL value, data access to this changed table will not be tracked and will be available to access by everybody.

<http://lennilobel.wordpress.com/2010/02/13/using-sql-server-2008-change-data-capture/>

Change tables are generated in the database System Tables with unique fields that indicate what action was performed:

INSERT (\_$operation = 2)

UPDATE (\_$operation = 3 [before]; \_$operation = 4 [after])

DELETE (\_$operation = 1)

<http://msdn.microsoft.com/en-us/library/bb500305.aspx>

These fields become important when you begin to create stored procedures, to parse through the different processes.

### 3.1.1 SELECT statements

#returns true (1) or false (0), making it easy to find out which databases are CDC-enabled and which aren’t:

SELECT DBNAME is\_cdc\_enabled FROM sys.databases

#show how the new cdc schema and user can be found in sys.schemas and sys.database\_principals

SELECT \* FROM sys.schemas WHERE name = 'cdc'

SELECT \* FROM sys.database\_principals WHERE name = 'cdc'

#find out which tables are CDC-enabled and which aren’t.

SELECT name, is\_tracked\_by\_cdc FROM sys.tables

### 3.1.2 CDC SYNC TABLES

There's a table named dbo.SDE\_SYNC\_TABLES in Warehouse SQLServer 2008 database that contains records of CDC table names and functions, and SDE feature class names that controls the python code. The python code iterates through this table, and calls the CDC function to get the changes from the CDC table and put them in the feature class. -6/7/13 Jason Sardano

## 3.2 SQL Server Agent

Once you have the database and desired tables with CDC’s enabled, you must start SQL Server Agent **or the changes will not be captured**. To start the Agent: 2

1. On the Start menu, point to All Programs, point to Microsoft SQL Server 2008 R2, point to Configuration Tools, and then click SQL Server Configuration Manager.
2. In SQL Server Configuration Manager, expand Services, and then click SQL Agent.
3. In the results pane, right-click any instance, and then click Start. A green arrow on the icon next to the SQL Server Agent and on the toolbar indicates that SQL Server Agent started successfully.
4. Click OK.

Our SQL server Agent keeps turning off. Tried configuring according to this document: <http://technet.microsoft.com/en-us/magazine/gg313742.aspx>, but already had this configuration. I went into SQL Server Configuration Manager and change the SQLServerAgent Start Mode from Manual to Automatic.

## 3.3. Create Multiversion view

Just a view or query that looks like a table

<http://resources.arcgis.com/content/kbase?fa=articleShow&d=24647>

<http://webhelp.esri.com/arcgisserver/9.3.1/java/index.htm#geodatabases/using_-1884018468.htm>

To create a multiversion table, execute the following code in a Command Prompt window:

>sdetable -o create\_mv\_view -T **TABLE\_NAME\_MV** -t **TABLE\_NAME** -i sde:sqlserver:**SERVERNAME** -D **DATABASENAME** -u **USERNAME** -p **PASSWORD**

Do not delete a multiversion table view from SQL directly, you should use the delete command from the Command Prompt.

>sdetable -o delete\_mv\_view -t **TABLE\_NAME\_MV** -i sde:sqlserver:**SERVERNAME** -D **DATABASENAME** -u **USERNAME** -p **PASSWORD**

If having trouble, check spelling of tables, or Need to register the table(s) first: <http://edndoc.esri.com/arcsde/9.0/admin_cmd_refs/sdetable.htm>

## 3.3 Make a change to the table

USE Warehouse

GO

UPDATE PLANTS\_LOCATION

SET X\_COORD='702736', Y\_COORD='474163', Z\_COORD='555555', Grid = 'NE', Location = '27'

WHERE (Acc\_num = '1-2' AND ACC\_NUM\_QUAL = 'A')

DELETE from PLANTS\_LOCATION

WHERE ACC\_NUM = '444-44' AND ACC\_NUM\_QUAL = '44'

INSERT INTO PLANTS\_LOCATION (ACC\_NUM, ACC\_NUM\_QUAL, line\_seq, rep\_id, X\_COORD, Y\_COORD, Z\_COORD)

VALUES ('444-44', '44', '99', '999999999', '701582', '473546', '0' )

# 4. ArcGIS configuration

## 4.1 Creating Feature Class

Staging is used by Production via ESRI Replication to determine the changes that are made during the synchronization process. If we don't have staging, then we won't be able to get a change file. It might be possible to create a replica between 2 versions (Production Default version and Production GIS Edit version, I would have to research this)

## 4.2. Python scripts

Scripts need to be updated after the July 2013 upgrade to ArcGIS 10.1. Jason Sardano created new python files located on arcgis10.arbweb.harvard.edu: C:\Users\Public\Documents\BGBase Connector

The connector sub-folder has the worker classes that are performing the majority of the work. - 8/14/13 DJT

created by Jason Sardano, May, 31 2012

The scripts are on the arcgis server, located at C:\Python26\ArcGIS10.0\Scripts\bg-sde-sync\sde\_to\_bg\_import\aa\_sde\_to\_bg\_sync.py

The file is currently configured to write the contents to C:\\Python26\\ArcGIS10.0\\Scripts\\bg-sde-sync\\sde\_to\_bg\_import\\changes.xml. This script will create the XML data change file that should be placed in the BGBASE6\IMPORT folder.

The code looks like this:

#bg\_xml\_output: Path to BG BASE XML File

#bg\_xml\_output commented out for testing purposes

#bg\_xml\_output = "M:\\BGBASE6\\IMPORT\\changes.xml"

bg\_xml\_output = "C:\\Python26\\ArcGIS10.0\\Scripts\\bg-sde-sync\\sde\_to\_bg\_import\\changes.xml"

Run with using scheduled tasks, see that section for details.

### 4.2.1 Log files

## f. XML File- disconnected synchronization

We can create an XML file by doing the synchronization between the Production default version and the GDB\_Edits version in staging through the disconnected process vs connected.   It does require that we setup a few more processes (creation of data change message – xml file, the management of the acknowledgement files, etc…).  Once we create these though we can place them in a model and automate them, to run either automatically or via the push of a button.

Read this topic first: acknowledgement and data change messages?

<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00270000003n000000.htm>

Then this on the data change message (xml file):

<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Exporting_a_data_change_message/00270000003p000000/>

Then the rest of the articles under the disconnected editing section. The change message may be zipped, which we can unzip the file once we put it somewhere.

# 5. Windows Task Scheduler

|  |  |  |
| --- | --- | --- |
| Windows Task Scheduler | PSEXEC command | Python script |



|  |  |  |
| --- | --- | --- |
| new task in Windows Scheduler, and used these instructions, <http://blogs.msdn.com/b/davethompson/archive/2011/10/25/running-a-scheduled-task-after-another.aspx> to have it fire after the Bgbase Warehouse task is completed | psexec command to execute warehouse\_to\_sde.bat; log files at ARCGIS:\Users\Public\Documents\BG-BASE\python\_logs\warehouse\_to\_sde.log | takes records from CDC and populates ArcGIS database and then clears CDC table |

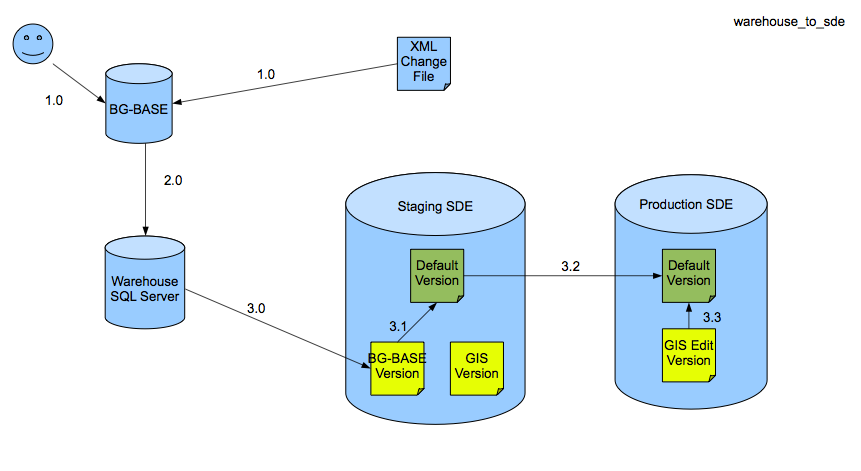
|  |  |  |
| --- | --- | --- |
| Windows Task Scheduler | BG-BASE VB script | BG-BASE VB script |



|  |  |  |
| --- | --- | --- |
| Task in Windows scheduler runs every half hour | .vbs script that processes records from BG-BASE to SQL Warehouse | .vbs script looks for XML file and processes |

# 4. Warehouse to SDE

## a. Schematic



|  |  |
| --- | --- |
| 1.0 | User makes edit in BG-BASE and/or BG-BASE receives edits from SDE via XML change file. BG-BASE processes changes |
| 2.0 | Performed by BG-BASE, a scheduled task (polling mechanism) to check for the XML changes. BG-BASE writes data changes to Warehouse db [using Open Insight data warehousing, automated synchronization. |
| 3.0 | After updates to the Warehouse are made, BG-BASE calls “warehouse\_to\_sde.py”. The python script reads the changes from SQL (Warehouse CDC) using the Microsoft SQL Server Python API and pushes the changes directly into Staging SDE via an ESRI Python Cursor. CDC is only enabled in the SQL Server Warehouse database. |
| 3.1 | “warehouse\_to\_sde.py” script reconciles changes in Staging BG-BASE Version to Staging Default Version. The code reconciles and posts from BG-BASE to Default during the warehouse to SDE process. |
| Staging | Staging is used by Production via ESRI Replication to determine the changes that are made during the synchronization process. If we don't have staging, then we won't be able to get a change file. |
| 3.2 | “warehouse\_to\_sde.py” script uses ESRI Python Synchronization between Staging GIS Version and Production SDE Default Version |
| 3.3 | “warehouse\_to\_sde.py” script reconciles between GIS Edit Version and Default Version on Production SDE. |

Notes:

1. An UPDATE in one table in SQL may be an INSERT in another table, we have to be able to capture this. For example edits in PLANTS\_CONDITION and MEASUREMENTS tables may lead to new conditions and measurements being created, this would be an INSERT.
2. SQL records with line\_seq = 1 represent the current data. If a plant is \*moved\*, the line\_seq numbers should be changed in SQL (just as if ctrl-n was pressed in BG-BASE to add a new line) and the new SQL record should be assigned a line\_seq of 1 and a new rep\_id. This would keep the data in synch with BG-BASE after we pick up the insert, and keep the history updated in SQL as well (as opposed to losing the history in SQL by just over-writing the current record). LOCATION\_CHANGE\_TYPE field updated to tell us what kind of an update it is.

## Multiple Users

If different users are executing different steps:

This version was looked up in a sql system table, when selecting from the multi view, based on the user that is logged in. The state\_id is then returned based on that version.  When the same user was used, it would have worked just fine because the version was already set, but with different users, you need to set the current version first using the following stored procedure:

EXEC dbo.set\_current\_version @version\_name = 'BG-BASE'

Any future processes or queries will need the above statement added (e.g., a process to reconcile all the changes made). Running that before executing code which relies on the version should fix it, either that or create a single user for all the processes and use that one user to run the various steps.  Once you run that stored procedure to set the version, it's set for the duration of that users session.

The reason that you can't see the data in SQL Server is because the data isn't really in the table. The database is versioned. This means that there are A and D tables (add and drop tables, respectively), that actually hold the data. ArcGIS sees the add/drop tables and pulls together the data to give you the view that you are seeing. SQL Server doesn't recognize these tables, so it doesn't use them when viewing the base table.

## Compress in ArcCatalog

Reconcile-Post-Rereconcile and Compress

If multiple versions then need to reconcile-post-rereconcile and compress.

<http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#geodatabases/reconciling_a_version.htm>

If Kyle is supposed to be querying on the MV (view) and not the actual table, then he needs to run EXEC dbo.set\_current\_version @version\_name = 'BG-BASE'

before he does anything.

Or are we supposed to be getting data out of the view and into the table?

A Geodatabase Compress moves edits down to the base tables.

<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00170000000p000000>

<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00290000005v000000>

Good, yes, dropping the version would also do the same as it would drop its related state lineage and the states contained in it….those would then be orphaned states, that should be cleaned up by a compress.

Manually (not STORED PROCEDURE) you track changes in the CDC with the following code:

**Go on STAGING database in SQL and execute:**

exec dbo.set\_current\_version 'BG-BASE'

exec dbo.edit\_version @name='BG-BASE', @edit\_action=1

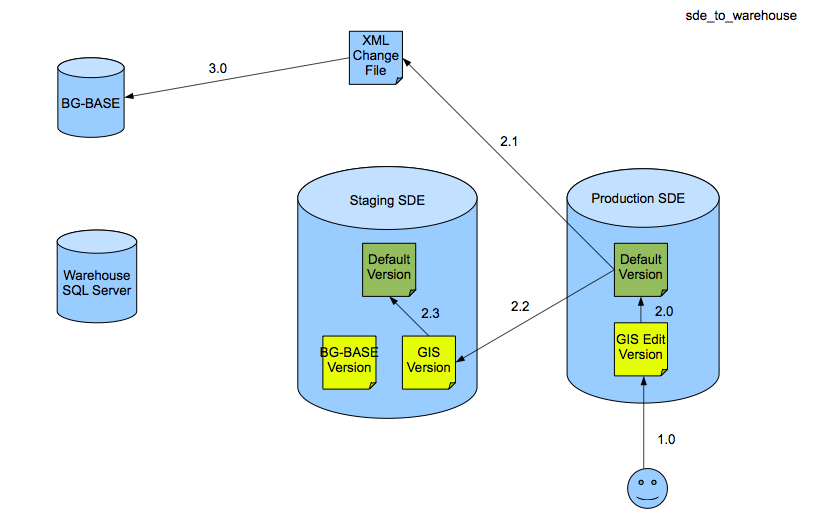
## GUIDs

The GUID’s are only useful within the Geodatabase’s replication processes that are used. According to the diagram we are only bringing attribute updates from BG-BASE over into the “staging” geodatabase.   GUID’s would be used to keep track of the features between the “staging” and production editing geodatabases.  The production editing geodatabase would then have edit versions – both in-house and field.

The key area is the staging geodatabase.  The best way to probably deal with the attribute update coming in from BG-BASE would be as a “bulk” update, which I think is what I was thinking about doing.   You would create a version off of default in which this bulk attribute update would be made to the necessary feature classes.  We set conflict detection to be by field.  So that bulk attribute update coming in can be reconciled and posted to the default version after the staging database has been synchronized from its “parent” production geodatabase with all the in-house and field edits.   GUID’s are being used within the geodatabase replication process and should not need to be tracked outside of the geodatabase environment.   The catch is that outside of the geodatabase environment there has to be some unique identifier on the feature that you are using to get back to it and bring those updates from the BG-BASE side in…   I would think though that you are only editing specific attributes on the features and don’t need to touch the geometry or GUID.    Thus you should not need the GUID at all on the BG-BASE side, unless you are using it as the identifier…then, yes.

There is a knowledge base article on populating the GUID through SQL, but I would think we would only want it populated and managed from the Geodatabase/ArcObjects side only (Production Editing Geodatabase) and pushed from there back to the staging geodatabase (where the synchronization with the incoming BG-BASE bulk attribute update takes place) and then finally back to BG-BASE – if necessary.

# 5. SDE to Warehouse



|  |  |
| --- | --- |
| Production | Production database: The reconcile, post, conflict management and compress processes which will manage all the field and desktop spatial edits to the production editing geodatabase. Reconcile changes from GIS Edit to Default in Prod |
| 1.0 | User makes edits in geodatabase.  Ready for future implementation: Field and desktop edits, editors are isolated into their own versions. Field edits can take place through 2-way disconnected (very long term or large geographic distance field edits), or check-out/check-in. |
| 2.0 | User calls “sde\_to\_warehouse.py” script, script reconciles edits to default. The code reconciles and posts from the GIS Edit Version to the Default version. |
| 2.1 | “sde\_to\_warehouse.py” script creates XML change file. |
| 2.2 | “sde\_to\_warehouse.py” script uses ESRI Synchronization between production SDE Default Version and Staging GIS Version – purpose is to detect any conflicts in incoming BG-BASE data (now in a version off Default) against edits made in field or desktops. |
| 2.3 | “sde\_to\_warehouse.py” script reconciles changes in staging GIS Version to Staging Default Version |
| 3.0 | “sde\_to\_warehouse.py” script calls BG-BASE web service with XML change file name as a parameter |

# 6. ArcGIS Tools

2/12/13 Jason Sardano

* On arcgis.arbweb, C:\Users\Public\Documents\ArcGIS, contains a Toolbox, SDE connection files and the python files.
* The toolbox is located at C:\Users\Public\Documents\ArcGIS\BG-BASE Tools.tbx (you have to view it with ArcCatalog).

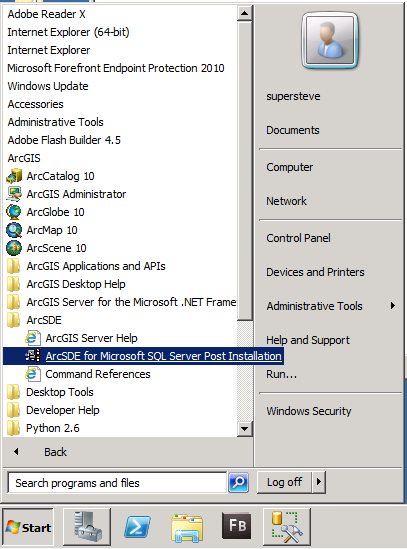
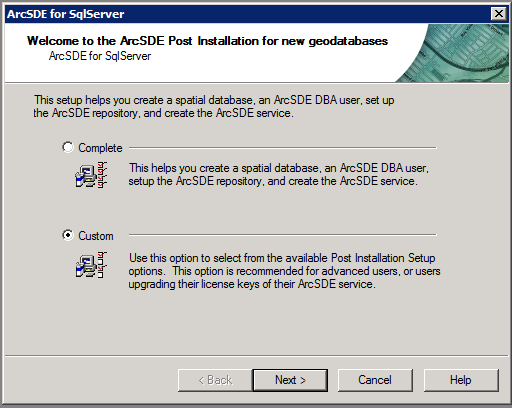
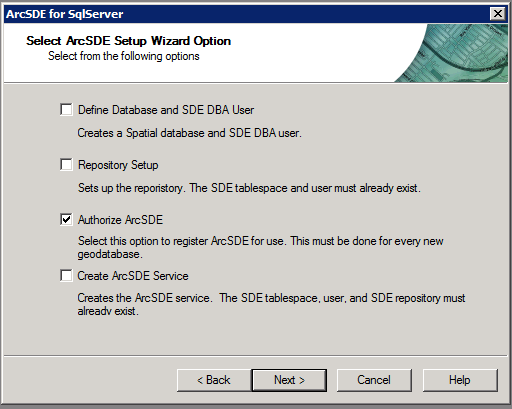
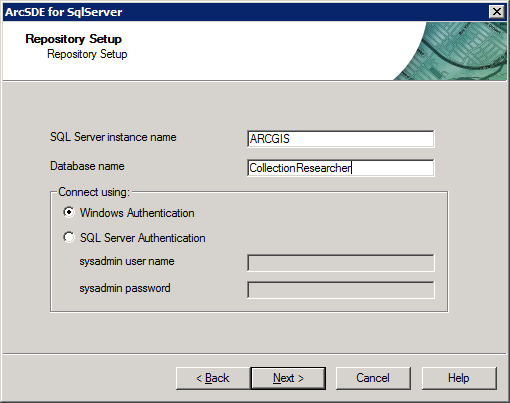
To receive data from Warehouse (BG-BASE) into SDE, you have 2 options:

1. Open the toolbox with ArcCatalog, double-click the "BG-BASE to SDE" model, and run the model.
2. Open Windows Explorer, navigate to C:\Users\Public\Documents\ArcGIS, double-click the warehouse\_to\_sde.bat file.

To generate the XML file containing SDE changes that will be consumed by BG-BASE, you have the following 2 options:

1. Open the toolbox with ArcCatalog, double-click the "SDE to BG-BASE" model, and run the model.
2. Open Windows Explorer, navigate to C:\Users\Public\Documents\ArcGIS, double-click the sde\_to\_warehouse.bat file.

# 7. ArcGIS Server Admin Notes

* **Configuration**
  + Spatial databases are in Microsoft SQL Server on arcgis.arbweb.harvard.edu, and use a DBO schema versus a SDE schema.
  + Connection files, map documents, and project files are stored in [R:\ArbMaps\GIS](file:///R:\ArbMaps\GIS).
  + To connect to a database with a computer running ArcGIS Desktop, create a folder connection in ArcMap or ArcCatalog to [R:\ArbMaps\GIS\Connections](file:///R:\ArbMaps\GIS\Connections) and select the appropriate connection file.
  + To connect to a database using ArcGIS Explorer, copy connection details from connection file using the procedure above.
    - Only authenticated users may connect.
* **Security & Permissions**
  + [Instructions for adding users to a SQL Server database.](http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#/Adding_Windows_authenticated_users_or_groups_to_a_SQL_Server_database/002q00000029000000/)
    - Readers should be granted the *datareader* role.
    - Editors should be granted the *datawriter* role.
  + Once you have created SQL accounts, you will need to give a user permissions to access each dataset in the geodatabase.
  + [Instructions for granting and revoking privileges.](http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#/Granting_and_revoking_privileges_on_datasets/002q0000002z000000/)
* **Software Updates**
  + Subscribe to patches and service packs RSS feed at <http://rss.esri.com/support/patchesandservicepacks>.
  + Install services packs for ArcGIS Desktop 10, ArcGIS Server 10, and ArcSDE 10 for Microsoft SQL Server 64.
  + All products installed on the same machine must have the same service pack installed, but different machines may have different versions.
* **License Update**
  + Obtain new license codes from Wendy Guan at the Center for Geographic Analysis in July of each year.
  + Run GIS Server Post Install with the Authorize GIS Server option only.
  + 
  + 
  + 
  + 
  + License codes and configuration files located in [R:\ArbMaps\GIS\Configuration](file:///R:\ArbMaps\GIS\Configuration).
  + Run ArcSDE for Microsoft SQL Server Post-Install with the Authorize ArcSDE option only for each spatial database.
    - ArnoldArboretumGIS
    - CollectionResearcher
    - Staging
    - Production

Extensions http://www.esri.com/software/arcgis/about/desktop-extensions

Availability node in the ArcGIS Administrator shows which extensions are currently checked out

ArcGIS Desktop, Network Analyst and Maplex

Brian Morgan

Putnam Research Fellow

# References

1. Dave, Pinal, 2009. Introduction to Change Data Capture (CDC) in SQL Server 2008. <http://www.simple-talk.com/sql/learn-sql-server/introduction-to-change-data-capture-%28cdc%29-in-sql-server-2008/>
2. Microsoft TechNet, SQL server. How to: Start SQL Server Agent (SQL Server Configuration Manager), <http://technet.microsoft.com/en-us/library/ms191454.aspx>
3. ESRI Support, HowTo: Create and use multiversion views for ArcSDE for MS SQL Server 2000, <http://support.esri.com/en/knowledgebase/techarticles/detail/24647>
4. ArcGIS Server 9.3.1 Help, Using multiversioned views, <http://webhelp.esri.com/arcgisserver/9.3.1/java/index.htm#geodatabases/using_-1884018468.htm>
5. ArcSDE Administration Command Reference, <http://edndoc.esri.com/arcsde/9.1/admin_cmd_refs/sdetable.htm>
6. <http://sql-articles.com/blogs/change-data-capture/>
7. <http://blogs.msdn.com/b/ramoji/archive/2009/11/20/could-not-update-the-metadata-that-indicates-database-is-enabled-for-change-data-capture-the-failure-occurred-when-executing-the-command-setcdctracked-value-1.aspx>
8. <http://support.microsoft.com/kb/913423>

# Appendix 1: SQL Server System Configuration

Microsoft SQL Server Management Studio 2008 10.0.4000.0

Microsoft Analysis Services Client Tools 10.0.4000.0

Microsoft Data Access Components (MDAC) 6.1.7601.17514

Microsoft MSXML 3.0 4.0 5.0 6.0

Microsoft Internet Explorer 8.0.7601.17514

Microsoft .NET Framework 2.0.50727.5444

Operating System 6.1.7601

Using Windows Authentication

Microsoft SQL Server Enterprise Edition (64-bit) 10.0.4000 SP2

Platform: NT x64

Operating System: 6.1 (7601)

Processors: 2

Operating System Memory: 8190

3Ghz Pentium 4 64bit capable machine with 8GB of ram

# Appendix 2: Cleaning CDC Tables in SQL

Safest way to do this is delete and recreate the feature class, that will remove all the tables and A and D tables.  So you recreate and re-register as versioned which will create new A and D tables (new numbers as well, since the Feature Class will have a new registration id in the SDE\_table\_registry)  …..that is the cleanest method.   Others – you could potentially do a truncate table if you are using a spatial type in SQL Server, but care needs to be taken as there may be relationships built against fields that may be getting removed.   So yes you could truncate as long as you truncate the D table as well, and run a compress to clean things up in the GDB/SDE administrative schema tables.  Then you might want to run these queries, but never delete state\_ids out of any of the administrative tables, or A and D tables or truncate them without first checking with ESRI, AND having a backup…   so typically I think deleting the feature class is the safest…

Not even sure if these still work ok in 10, they should but have not re-visited/used these in a long time…

a) Check for Incomplete or missing lineages:

select state\_id from sde.states ST where not exists

(select \* from sde.state\_lineages SL

where ST.lineage\_name = SL.lineage\_name and SL.lineage\_id = 0);

b) Check for Invalid parent state ids:

select state\_id from sde.states where parent\_state\_id not in

(select state\_id from sde.states)

order by state\_id;

c) Check for States with no lineages:

select distinct state\_id from sde.states where lineage\_name not in

(select lineage\_name from sde.state\_lineages)

order by state\_id;

select distinct state\_id from sde.states where state\_id not in

(select lineage\_id from sde.state\_lineages)

order by state\_id;

d)Check for edits referencing states with no parent state:

select state\_id from sde.MVTABLES\_MODIFIED

where state\_id in (select state\_id from sde.states where parent\_state\_id not in

(select state\_id from sde.states));

# Appendix 3: Troubleshooting

The key thing we need to provide back to BG is the xml change file.   We can create an XML file by doing the synchronization between the production default version and the GDB\_Edits version in staging through the disconnected process vs connected.   It does require that we setup a few more processes (creation of data change message – xml file, the management of the acknowledgement files, etc…).  Once we create these though we can place them in a model and automate them, to run either automatically or via the push of a button.

Read this topic first: <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00270000003n000000.htm>

then this on the data change message (xml file)

<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Exporting_a_data_change_message/00270000003p000000/>

 then the rest of the articles under the disconnected editing section

Think we've figured a little more out. It seems like our changes are there, the stored procedure is working, but they are not making it out of the A12 table.

The PLANTS\_LOCATION\_MV table is using the following code:

WHERE   s.state\_id = Staging.dbo.SDE\_get\_view\_state()

But Staging.dbo.SDE\_get\_view\_state() is returning as 71

but the state\_id's of the changed records are up to 104

How can we get them back in sync?

Compress

You may need to run a compress to move them out.

Are there multiple versions?  Or are we just going to Default version at this point?  Can’t remember, however, if multiple versions then we need to reconcile-post-rereconcile and compress.  If not just a compress should work

Jason: We're doing reconcile, post  and compress before running the sync.

I believe that the process isn't working because different users are executing different steps.  The version is "BG-BASE".  This version was looked up in a sql system table, when selecting from the multi view, based on the user that is logged in. The state\_id is then returned based on that version.  When the same user was used, it would have worked just fine because the version was already set, but with different users, you need to set the current version first using the following stored procedure:

EXEC dbo.set\_current\_version @version\_name =  'BG-BASE'

Running that before executing code which relies on the version should fix it, either that or create a single user for all the processes and use that one user to run the various steps.  Once you run that stored procedure to set the version, it's set for the duration of that users session.  It's any future processes or queries you run that need that statement added.  I assume there is a process to reconcile all the changes made.  That process would need this added.

Jason: I saw the change in a12 (which gets in there from the stored procedure). I then see the change get moved from the a12 table to the plants\_location feature class on staging (which is a result of my python script calling a geoprocessing function). The problem is syncing the data from Staging to Production. I couldn't get this to work even if I made the change to the staging data using ArcGIS; which leads me to believe that there is a problem with the replicas. I am not familiar with ArcGIS synchronization