Real-Time Updating of ArcSDE through SQL

An Introduction
By Chad D. Cooper, GISP, University of Arkansas

Pushing real-time data into an ArcSDE geodatabase for immediate consumption by end users can be challenging. However, with the advent of Microsoft’s SQL Server 2008 and Esri support for the built-in SQL Server 2008 spatial data types, real-time updating of ArcSDE feature classes in SQL Server 2008 has gotten easier. By utilizing the SQL Server geometry and geography data types, SQL stored procedures, and triggers, we can essentially bypass the Esri ArcSDE stack and push attribute and spatial updates directly to nonversioned feature class tables. Simple to complex insertions and updates can be performed through SQL, allowing us to provide (upon map refresh or redraw) instant feature class updates to end users.

Why Do This?
Real-time data is becoming more and more prevalent in the world around us and therefore in greater demand within GIS applications. So how can real-time data be pushed to an ArcSDE geodatabase? Third-party applications, such as Informatica and Safe Software’s FME, can do this. Both products work very nicely (I’ve used them both) but can be too costly for many organizations. ArcObjects can also be used, but not all organizations have access to a developer capable of creating an application for updating ArcSDE. Instead, let’s look at a simpler method using SQL to directly update the feature class table.

Using the SQL Server Geometry and Geography Data Types
By default, ArcSDE databases in SQL Server use the ArcSDE compressed binary type for feature geometry storage within the SHAPE field. To use the Microsoft geometry and geography data types instead for feature geometry storage, we need to specify them when creating our feature class. Since we cannot change the geometry field type of an existing feature class, we will create a new feature class and specify the GEOGRAPHY configuration keyword during the process.

In ArcCatalog, create a new polygon feature class as you normally would, but when you get to the database storage configuration, use either the GEOGRAPHY or GEOMETRY configuration keyword instead of taking the DEFAULTS option. We will use GEOGRAPHY, which instructs ArcSDE to use the GEOGRAPHY binary data type, to store our spatial information in the SHAPE field (Figure 1).

A Word of Caution
What we are about to tackle can be very dangerous, as changes to nonversioned data cannot be rolled back. It is highly recommended that the methodology about to be discussed be thoroughly tested in a development and/or testing environment before ever being attempted in production.
What You Will Need

- Microsoft SQL Server 2008 (Enterprise or Express; Express used for the examples)
- Enterprise ArcSDE for your DBMS
- SQL Server Management Studio (available for both SQL Server Enterprise and Express)
- Command line access to ArcSDE with necessary account permissions
- Testing or development ArcSDE database environment (Do not attempt this in a production environment.)

Populating Our Feature Class with Data

With SQL Server Spatial, you create and update features through SQL statements issued against the feature class table. These SQL statements can be issued to your ArcSDE database (in this example, it is called City) through SQL Server Management Studio. Listing 1 demonstrates how to add a record into the Tracts table, populating the OBJECTID, TRACT_ID, TRACT_NAME—and most importantly—SHAPE fields. This looks like any other SQL INSERT statement with one difference: the usage of the geography::STPolyFromText method to create the feature geometry from well-known text (WKT).

The Tract polygon is represented by coordinate pairs, each representing a polygon vertex, led by the POLYGON keyword, passed into the geography::STPolyFromText method. The coordinate string is followed by the spatial reference identifier (SRID), which in this case is EPSG:4326, WGS84.

```
BEGIN
    INSERT INTO [City].[dbo].[Tracts]
    (
        OBJECTID,
        TRACT_ID,
        TRACT_NAME,
        SHAPE
    )
    SELECT 1,1000,'Smith',
    geography::STPolyFromText(
        'POLYGON(
            -77.0532238483429 38.870863029297695,
            -77.05468297004701 38.87304314667469,
            -77.05788016319276 38.872800914712734,
            -77.05849170684814 38.870219840133124,
            -77.05556273460388 38.8690670969385,
            -77.0532238483429 38.870863029297695
        )',
        4326)
END
```

Listing 1. Inserting a record into the Tracts feature class

Figure 1. Specifying the GEOGRAPHY storage configuration keyword
Querying the Data

Once we have data in our table, we can query it using SQL Server Management Studio.

```
SELECT [OBJECTID]
  , [TRACT_ID]
  , [TRACT_NAME]
  , [SHAPE]
FROM [City].[dbo].[TRACTS]
```

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>TRACT_ID</th>
<th>TRACT_NAME</th>
<th>SHAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1000</td>
<td>Smith</td>
<td>0xE610000000104060000008610937078EF424000000005684353C07C78B9E08BEF424000100000ED7F4353C0174EBCF0B7EF42400100004FB44353C0DFD91C5D63E4F42000000054BE43530033973DEF4240000000578E4353C08610937078EF424000000005684353C00100000020000000001000000FFFFFFFF000000003</td>
</tr>
</tbody>
</table>

What about that SHAPE field data? Data contained in the Microsoft SQL Server 2008 geometry and geography data types are stored as a stream of binary data, which is what was returned from the query in the SHAPE field. We won't go into the details of binary storage (which you can find on the web). Luckily, there are built-in methods that allow us to retrieve the WKT representations of instances of spatial data. Here are three of these methods: STAsText(), STAsTextZM(), and ToString(). To get each point in our geometry, we can modify our query slightly by adding the STAsText() method to the SHAPE instance as shown in Listing 3.

```
SELECT [OBJECTID]
  , [TRACT_ID]
  , [TRACT_NAME]
  , [SHAPE].STAsText()
FROM [City].[dbo].[TRACTS]
```

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>TRACT_ID</th>
<th>TRACT_NAME</th>
<th>SHAPE</th>
</tr>
</thead>
</table>

↑ The return from the query in Listing 2

↑ Listing 2: Querying Tracts feature class

↑ Listing 3. Adding the STAsText() method
<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>TRACT_ID</th>
<th>TRACT_NAME</th>
<th>SHAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1000</td>
<td>Smith</td>
<td>POLYGON (((-77.0532238483429, 37.870863029297695), (-77.054682970047011, 37.873043146674689), (-77.057880163192763, 37.872800914712734), (-77.058491706848145, 37.870219840133124), (-77.055562734603882, 37.8690670969385), (-77.0532238483429, 37.870863029297695)))</td>
</tr>
</tbody>
</table>

The return from the query in Listing 3

Setting the Feature Class Spatial Extent and Adding a Spatial Index

An important step in this process is setting the spatial extent of our feature class and adding a spatial index. I will set the spatial extent at the command line via the sdelayer alter command and let ArcSDE calculate the extent for me using the -E calc flag:

```
C:\>sdelayer -o alter -l Tracts,Shape -E calc -i sde:sqlserver:servername\sqlexpress -s server_name -D City -u user -p pass
```

Listing 4: Setting the extent

Next, we can add a spatial index using the Add Spatial Index ArcToolbox tool. (Other ways to set the spatial extent and spatial indexes can be found in the ArcGIS documentation.)

Editing the Feature Class

One of the benefits of editing an enterprise ArcSDE feature class through SQL is that the editing can be performed while the feature class is being consumed by multiple clients such as ArcMap or ArcGIS for Server map services. When a feature is updated or inserted, it is visible to the client upon map refresh (caused by pan, zoom, or refresh map commands). Listing 5 shows the SQL statements used to edit the feature we inserted in Listing 1. Figure 2 shows the result. Again, this is a typical SQL UPDATE statement with the exception of the geography::STPolyFromText method call.

↑ The return from the query in Listing 3

↑ Figure 2. Our feature after insert and after edit
BEGIN
UPDATE [City].[dbo].[Tracts]
SET SHAPE = geography::STPolyFromText(
  'POLYGON((-77.0532238483429 38.870863029297695,
  -77.05468297004701 38.87304314667469,
  -77.05788016319276 38.872800914712734,
  -77.05849961836481 38.869157633013312,
  -77.05556273460388 38.8690670969385,
  -77.0532238483429 38.870863029297695
  ))', 4326)
WHERE TRACT_ID = 1
END

Listing 5: Editing a vertex (highlighted line) in our feature class

Expanding the Possibilities
In addition to adding and updating features, SQL Server has many other spatial functions such as the ability to perform spatial analyses. SQL Server 2008 includes support for methods that are defined by the Open Geospatial Consortium (OGC) standard and a set of Microsoft extensions to support that standard. Full documentation of these methods can be found on the MSDN site (msdn.microsoft.com; search for “spatial data type method reference”). Some of the more interesting methods are listed below in Table 1.

<table>
<thead>
<tr>
<th>Method</th>
<th>What It Does</th>
</tr>
</thead>
<tbody>
<tr>
<td>STIntersects()</td>
<td>Tests whether two objects intersect</td>
</tr>
<tr>
<td>STDistance()</td>
<td>Returns the shortest LineString between two object instances</td>
</tr>
<tr>
<td>STIntersection()</td>
<td>Returns an object representing the points where an object instance intersects another object instance</td>
</tr>
</tbody>
</table>

Table 1. Some of the spatial methods available in SQL Server 2008

SQL Server spatial methods can be incorporated into stored procedures and, along with database triggers, can be a powerful way to programatically update features. Spatial operations, such as testing if a point location lies within a polygon boundary, can be quickly performed directly in the database. Stored procedures in SQL Server can be called programatically with .NET (the System.Data.SqlClient namespace) and Python (pymssql, pyodbc libraries), along with other methods in other languages.

Editing Caveats
As was pointed out earlier, directly editing your enterprise database is not to be taken lightly, nor is it for the faint of heart. This article was written as a brief introduction to utilizing the Microsoft spatial data types with ArcSDE. It is highly recommended that if you are interested in using the methods described here, you educate yourself on the inner workings of ArcSDE, versioning, concurrency, locking, editing constraints, isolation levels, and spatial data types and how these all apply to and work with your flavor of geodatabase. Of particular importance is your dataset extent, which must be carefully set because adding features outside of the current extent can cause issues. Detailed information on these topics can be found in the Professional and Administrator libraries in the ArcGIS 10 for Desktop online help at the ArcGIS Resource Center.

Conclusion
Using SQL to directly edit spatial data in ArcSDE is a very powerful concept. It is a fast and lightweight method to provide timely updates to end users. Features can be both inserted and updated. A plethora of static and extended methods exist to help in manipulating and analyzing spatial data. SQL statements can be incorporated into stored procedures, enabling your SQL code to be called programatically. Lastly, ArcGIS 10.1 will allow for direct access to SQL Server spatial database tables, thereby making utilization of the SQL Server spatial data types a more viable option for many organizations.

With great power comes great responsibility. Know the limitations and requirements of your database and use them to your advantage. For more information, contact Chad Cooper at chad@coopergeo.com.
GeoCollector for ArcPad

GeoCollector™ for ArcPad® is an end-to-end field data collection solution that merges Esri® mobile geographic information system (GIS) technology with Trimble® professional-grade global GPS capabilities.

This offer includes Trimble’s GeoExplorer 6000 series and two choices of accuracy (after postprocessing)

- 10-cm capable (GeoCollector XH)
- 50-cm capable (GeoCollector XT)

You can also include Floodlight™ satellite shadow reduction technology, which enhances GNSS performance to deliver more positions and better accuracies in difficult conditions.

This is an exclusive offer from Esri and is packaged for US customers.

Learn more about GeoCollector for ArcPad at esri.com/geocollector.