Advanced Spatial Data Access

Introduction

Tabular queries can access spatial data stored in the Soil Data Mart (SDM) database. Spatial data is stored in tables such as mupolygon, mupoint and muline as pairs of columns, one with the data in WGS84 geographic coordinates (epsg:4326) and one with the data in Web Mercator (WM, epsg:3857 – this is not quite accurate as true epsg:3857 uses a flattened sphere for coordinate projection, in the SDM database the WM data are actually projected using a perfect sphere, this pseudo-Web Mercator projection matches that used by common Web sources such as Google Maps and Microsoft Bing maps). The column names are formed as follows: <table-name><[geo|proj>. For example in table "mupolygon", the WGS84 data are held in column "mupolygongeo".

A set of SQL tabular functions have been defined to simplify navigation between coordinates and SDM table data keys. For more advanced applications a series of macros have been included to provide extra functionality that may not normally be available through queries. In addition there is no restriction on referencing the spatial tables through normal queries.

Tabular Functions

These functions return a "table" so may be treated in SQL statements as a "FROM" table source.

Note that the WKT must follow the ISO standard definition (see the Wikipedia page at http://en.wikipedia.org/wiki/Well-known_text for a simple introduction), for example the list of vertices that defines a polygon must end at the same point as the beginning point).

`SDA_Get_Areasymbol_from_intersection_with_WktWgs84`
`SDA_Get_Areasymbol_from_intersection_with_WktWm`

Given a WKT geometry definition in WGS84 or Web Mercator (WM), returns a table of "areasymbol" intersected by the geometry.

`SDA_Get_AreasymbolWKTWgs84_from_Areasymbol`
`SDA_Get_AreasymbolWKTWm_from_Areasymbol`

Given an areasymbol, returns a table of the areasymbol's outline as WKT in WGS84 ("AreasymbolWkt84") or Web Mercator (WM) ("AreasymbolWktWm").

`SDA_Get_Mukey_from_intersection_with_WktWgs84`
`SDA_Get_Mukey_from_intersection_with_WktWm`

Given a WKT geometry definition in WGS84 or Web Mercator (WM), returns a table of the "mukey" for mapunits intersected by the geometry.

`SDA_Get_MupolygonWktWgs84_from_Mukey`
`SDA_Get_MupolygonWktWm_from_Mukey`

Given an mukey, returns a table of all the corresponding mapunit's mupolygons as WKT in WGS84 ("MupolygonWktWgs84") or Web Mercator (WM) ("MupolygonWktWm").

SDA_Get_MupointWktWgs84_from_Mukey
SDA_Get_MupointWktWm_from_Mukey

Given an mukey, returns a table of all the corresponding mapunit's mupoints as WKT in WGS84 ("MupointWktWgs84") or Web Mercator (WM) ("MupointWktWm").

SDA_Get_MulineWktWgs84_from_Mukey
SDA_Get_MulineWktWm_from_Mukey

Given an mukey, returns a table of all the corresponding mapunit's mulines as WKT in WGS84 ("MulineWktWgs84") or Web Mercator (WM) ("MulineWktWm").

Sample SQL statements using the above new functions:

```
select * from SDA_Get_Areasymbol_from_intersection_with_WktWgs84(
  'polygon((
     -121.77100 37.368402,
     -121.77100 37.373473,
     -121.76000 37.373473,
     -121.76000 37.368402,
     -121.77100 37.368402))')

select * from SDA_Get_Areasymbol_from_intersection_with_WktWm(
  'polygon((
    -13555610.9782664 4490483.16765171,
    -13555610.9782664 4491391.31746713,
    -13554135.9341103 4491391.31746713,
    -13554135.9341103 4490483.16765171,
    -13555610.9782664 4490483.16765171))')
```

And for the mukeys specifically,

```
select * from SDA_Get_Mukey_from_intersection_with_WktWgs84(
  'polygon((
     -121.77100 37.368402,
     -121.77100 37.373473,
     -121.76000 37.373473,
     -121.76000 37.368402,
     -121.77100 37.368402))')

select * from SDA_Get_Mukey_from_intersection_with_WktWm(
  'polygon((
    -13555610.9782664 4490483.16765171,
    -13555610.9782664 4491391.31746713,
    -13554135.9341103 4491391.31746713,
    -13554135.9341103 4490483.16765171,
    -13555610.9782664 4490483.16765171))')
```

Combining with an areasymbol test for 'CA646',

```
select S.mukey, M.musym
from SDA_Get_Mukey_from_intersection_with_WktWm(
  'polygon((
    -13555610.9782664 4490483.16765171,
    -13555610.9782664 4491391.31746713,
    -13554135.9341103 4491391.31746713,
    -13554135.9341103 4490483.16765171,
    -13555610.9782664 4490483.16765171))') as S,
```

```
legend as L,
mapunit M
where M.mukey = S.mukey
and M.lkey = L.lkey
and L.areasymbol = 'CA646'
```

Macros

Macros are shorthand statements that are “expanded” into normal SQL statements. They allow creation of SQL statements that would normally not be allowed or present short sequences of SQL statements that address specific needs.

In all cases the macro name and arguments are surrounded by immediately-adjacent “tilde” characters (“~”). No embedded whitespace is allowed. SQL variable names are indicated by an at-sign (“@”) followed by an identifier. The identifier may start with an alphabetic character or underscore and be followed by zero or more alphanumeric characters and/or underscores. The macro names and variable names are case-insensitive.

If your query includes the line:

```
~DeclareGeometry(@aoi)~
```

That line will be translated into the following lines:

```
-- ~DeclareGeometry(@aoi)~
-- begin macro substitution
declare @aoi geometry;
-- end macro substitution
```

(The result of the macro expansion is not displayed. If you include macros in a query for queued execution the completion email will show the expanded SQL text.)

Following the above macro statement conventional SQL statements may reference the defined variable “@aoi”. For example spatial coordinates may be assigned by use of “Well Known Text”, here a polygonal boundary is defined in WGS84 (epsg:4326):

```
select @aoi = geometry::STPolyFromText('polygon((
-121.157072910308 46.0181639308995,
-121.321280753631 45.9248106152548,
-121.348997869021 45.9168439802811,
-121.157072910308 46.0181639308995
))', 4326)
```

A few of the macros also reference an “IdGeom” or “IdGeog” table variable. These are defined by macros, the statements are shown here:

```
~DeclareIdGeomTable(@intersectedPolygonGeometries)~
~DeclareIdGeogTable(@intersectedPolygonGeographies)~
```

These two macro statements are expanded into:

```
-- ~DeclareIdGeomTable(@intersectedPolygonGeometries)~
-- begin macro substitution
declare @intersectedPolygonGeometries table (id int, geom geometry);
-- end macro substitution
-- ~DeclareIdGeogTable(@intersectedPolygonGeographies)~
-- begin macro substitution
declare @intersectedPolygonGeographies table (id int, geog geography);
-- end macro substitution
```

These 2-tuple tables are used, for example, to associate a mapunit polygon's spatial definition with a mukey for joining against other tables in the SDM database. A fully worked example will be presented after the following lists of macros.

A number of the macros define simple SQL variables:

```
~DeclareBigint(@varname)~
~DeclareBit(@varname)~
~DeclareInt(@varname)~
~DeclareSmallint(@varname)~
~DeclareTinyint(@varname)~
~DeclareFloat(@varname)~
~DeclareReal(@varname)~
~DeclareDate(@varname)~
~DeclareDatetime(@varname)~
~DeclareTime(@varname)~
~DeclareChar(@varname,n)~
~DeclareVarchar(@varname,[n|max])~
~DeclareGeometry(@varname)~
~DeclareGeography(@varname)~
```

In the case of the Char and Varchar definitions a number (1..8000) must be supplied as the second argument, the Varchar definition also allows a MAX specification. For example, to declare a char(3), varchar(4) and varchar(max), use:

```
~DeclareChar(@a_char_variable,3)~
~DeclareVarchar(@a_varchar_variable,4)~
~DeclareVarchar(@a_second_varchar_variable,max)~
```

As noted previously, 2-tuple id and geometry or geography tables are declared with:

~DeclareIdGeomTable(@varname)~
~DeclareIdGeogTable(@varname)~

Mapunit polygons (WGS84) clipped to a filtering boundary (WGS84) are returned by:

```
~GetClippedMapunits(@filter,[point|line|polygon],[geo|proj],@outtable)~
```

Before this statement, create the @filter with macro DeclareGeometry and declare @outtable with macro DeclareIdGeomTable.

The statements used by the above macro will return the 2-tuple table of mukeys and associated clipped geometry. Note that the geometries may include geometry collections, points, multipoints, lines, multilines, polygons and multipolygons. The result may have composites split apart by means of:

```
~SplitIdGeomTable(@intable, @outtable)~
```

Both table variables must be created by macro DeclareIdGeomTable before the above statement us employed.

The final macro converts an IdGeom table into an IdGeog table. This is especially useful for determining areas of the spatial objects in the tables.

```
~GetGeogFromGeomWgs84(@intable,@outtable)~
```

The table variables must be created by macros DeclareIdGeomTable and DeclareIdGeogTable before the macro statement is used.

Here's an example of an area of interest (AOI) specification followed by retrieval of data from other tables in the SDM database and aggregation of area by mukey:

```
-- Define a triangular AOI in WGS84
~DeclareGeometry(@aoi)~
select @aoi = geometry::STPolyFromText('polygon((
-121.157072910308 46.0181639308995,
-121.321280753631 45.9248106152548,
-121.348997869021 45.9168439802811,
-121.157072910308 46.0181639308995
))', 4326)

-- Extract all intersected polygons
~DeclareIdGeomTable(@intersectedPolygonGeometries)~
~GetClippedMapunits(@aoi,polygon,geo,@intersectedPolygonGeometries)~

-- Convert geometries to geographies so we can get areas
~DeclareIdGeogTable(@intersectedPolygonGeographies)~
```

```
~GetGeogFromGeomWgs84(@intersectedPolygonGeometries,@intersectedPolygonGeog
raphies)~

-- Return the polygonal geometries
select * from @intersectedPolygonGeographies
where geog.STGeometryType() = 'Polygon'

-- get aggregated areas and associated mukey, musym, nationalmusym,
areasymbol, mucertstat (Map Unit Certification Status)
select id, sum(geog.STArea()) as area
into #aggarea from @intersectedPolygonGeographies
group by id;

-- Return the polygons with joined data
select mukey, area, musym, nationalmusym, areasymbol, mucertstat
from #aggarea A, mapunit M, legend L
where A.id = M.mukey and M.lkey = L.lkey
order by id;

-- Return the aggregated area by mukey
select id as mukey, area from #aggarea
```