

## Using MapX and MapXtreme to Load and Query SpatialWare for SQL Server Layers

**MAPINFO® SPATIALWARE® FOR SQL SERVER ENABLES GEOGRAPHIC OBJECTS, LIKE LINES, POLYLINES AND REGIONS, TO BE STORED IN SQL SERVER TABLES. THIS POWERFUL ADDITION TO THE DATABASE ALLOWS USERS TO SEND GEOGRAPHIC QUERIES TO A TABLE IN SQL SERVER; PROCESS THE RESULTS ON THE SERVER; AND, THEN DISPLAY RESULTS IN A CLIENT APPLICATION LIKE MAPINFO MAPX® OR MAPINFO® MAPXTREME®.**

The screen shot to the right is taken from an application developed using MapX and Spatialware for SQL Server. This application will load a schools table stored in SQL Server. Then, it will ask the user to select a school and a radius value in miles. Once this information is selected, a query will be sent to SQL Server to return the census collection districts that fall within this radius around the school. A theme will be created based on the total weekly income for the census collection districts that were returned. The census collection districts are stored in a table in SQL Server.

The following is the relevant code used in MapX to return the census collection districts from SQL Server that are contained within the 50 mile radius around the selected Tara Girls School.

```
Sub createtheme()
```

```
'this converts the radius in miles to meters for the calculation in SQL Server
```

```
If CombRadius.Text = "20" Then
```

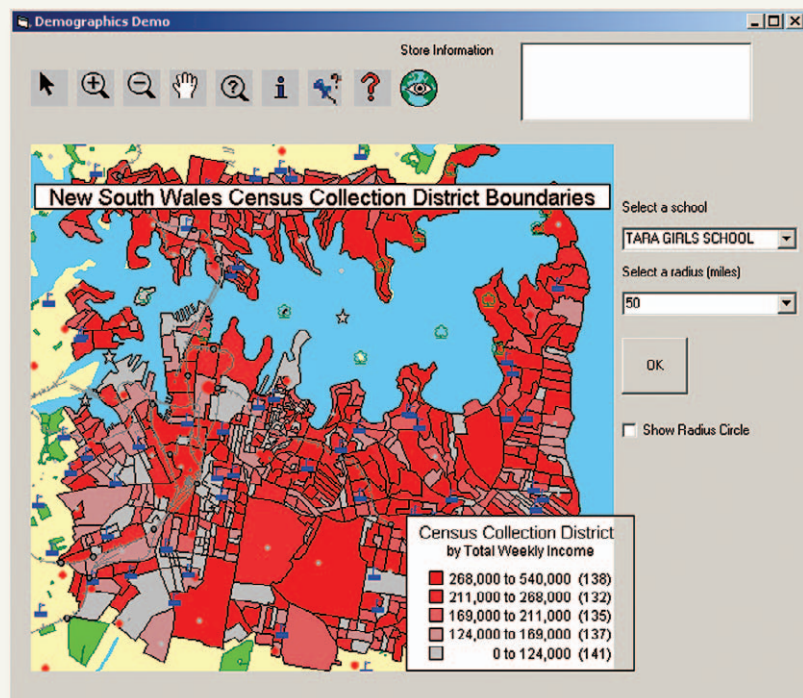
```
    i = Val(CombRadius) * 1609.344
```

```
Elseif CombRadius.Text = "30" Then
```

```
    i = Val(CombRadius) * 1609.344
```

```
Elseif CombRadius.Text = "40" Then
```

```
    i = Val(CombRadius) * 1609.344
```



```
Elseif CombRadius.Text = "50" Then
```

```
    i = Val(CombRadius) * 1609.344
```

```
End If
```

```
'adds the school layer as a dataset so we can search on it
```

```
Set lyrstore =  
Map1.Layers.Item("schools")
```

```
Set ds =  
Map1.Datasets.Add(miDataSetLayer, lyrstore, "demo")
```

```
'searches for the store the user chose
```

```
Set ftrs = lyrstore.Search("name=" &  
"" & CombStore.Text & """)
```

```
Set ftr = ftrs.Item(1)
```

```
'creates a point from the coordinates of the store found which will be used to create the radius circle
```

```
pnt.Set ftr.CenterX, ftr.CenterY
```

```
Map1.Layers.CreateLayer "temp", , 1  
'temporary layer to store radius circle
```

```
Map1.Layers.Item("temp").Visible = False
```

```
'code to create circle
```

```
Set ftrcir =  
Map1.FeatureFactory.CreateCircularRegion(1, pnt, Val(CombRadius.Text),  
miUnitMile, 50)
```

```
ftrcir.Style.RegionPattern = 0
```

```
ftrcir.Style.RegionBorderWidth = 2
```

```
'adds circle feature to the temporary  
layer
```

```
Set ftrcir =  
Map1.Layers.Item("temp").AddFeature  
(ftrcir)
```

```
'displays the check box for displaying  
the circle
```

```
CheckRadius.Visible = True
```

```
CheckRadius.Value = 0
```

```
'creates a query to send to SQL Server  
to select the blocks that fall within the  
radius circle, uses HG_LL_Circle because  
considers the curvature of the earth
```

```
str = "exec sp_spatial_query 'select  
Tot_Wkly_Income,hg_morph_out(sw_geom  
etry) from demographics where  
ST_Overlaps(sw_geometry,HG_LL_Circle  
(" & ftr.CenterX & "," & ftr.CenterY & ","  
& i & ")')"
```

```
LayerInfoObject.Type =  
miLayerInfoTypeServer
```

```
LayerInfoObject.AddParameter "name",  
"blocks"
```

```
LayerInfoObject.AddParameter  
"ConnectionString",  
"DSN=sqlserver;UID=ts;PWD=ts"
```

```
LayerInfoObject.AddParameter "Query",  
str
```

```
LayerInfoObject.AddParameter "toolkit",  
"ODBC"
```

```
'adds the selected blocks to the map
```

```
Set lyrstore =  
Map1.Layers.Add(LayerInfoObject, 3)
```

Here is the equivalent code for adding the results of the same query above in MapXtreme 3.0:

```
dim lInfo
```

```
dim str
```

```
'this is the query being sent to SQL  
Server to return the census collection  
districts
```

```
str = "exec sp_spatial_query 'select  
Tot_Wkly_Income,hg_morph_out(sw_geom  
etry) from demographics where  
ST_Overlaps(sw_geometry,HG_LL_Circle  
(" & fMapX & "," & fMapY & "," &  
cMapXRadius & ")'"
```

```
Set lInfo =  
Session(cMapXCourierObject).CreateMap  
XLayerInfo
```

```
'layer type is RDB
```

```
lInfo.Type = 4
```

```
lInfo.AddParameter "Name", "blocks"
```

```
'connection string
```

```
lInfo.AddParameter "ConnectionString",  
"DSN=sqlserver;UID=ts;PWD=ts;"
```

```
'toolkit name
```

```
lInfo.AddParameter "ToolKit", "ODBC"
```

```
'layer server query
```

```
lInfo.AddParameter "Query", str
```

```
set
```

```
lyrstore=Session(cMapXObject).Layers.  
Add (lInfo)
```

As demonstrated with this code, users may send geographic queries to a table stored in SQL Server, process results via a server and display results via a mapping server. ★

---

*Cindy Makarowsky is a technical support specialist with MapInfo Corporation.*