Práctica 5. MS SQL Server

SQL Server 2008 Spatial Data Types

SQL Server 2008 provides the ***geography*** data type for geodetic spatial data, and the ***geometry*** data type for planar spatial data. Both are implemented as Microsoft .NET Framework Common Language Runtime (CLR) types, and can be used to store different kinds of geographical elements such as points, lines, and polygons. Both data types provide properties and methods that you can use to perform spatial operations such as calculating distances between locations and finding geographical features that intersect one another (such as a river that flows through a town.)

The geography Data Type

The ***geography*** data type provides a storage structure for spatial data that is defined by latitude and longitude coordinates. Typical uses of this kind of data include defining roads, buildings, or geographical features as vector data that can be overlaid onto a raster-based map that takes into account the curvature of the Earth, or for calculating true great circle distances and trajectories for air transport where the distortion inherent in a planar model would cause unacceptable levels of inaccuracy.

The geometry Data Type

The ***geometry*** data type provides a storage structure for spatial data that is defined by coordinates on an arbitrary plane. This kind of data is commonly used in regional mapping systems, such as the state plane system defined by the United States government, or for maps and interior floor plans where the curvature of the Earth does not need to be taken into account.

The ***geometry*** data type provides properties and methods that are aligned with the Open Geospatial Consortium (OGC) *Simple Features Specification for SQL* and enable you to perform operations on geometric data that produce industry-standard behavior.

Spatial Data Type Methods

Both spatial data types in SQL Server 2008 provide a comprehensive set of instance and static methods that you can use to perform queries and operations on spatial data. For example, the following code sample creates two tables for a city mapping application; one contains geometry values for the districts in the

city, and the other contains geometry values for the streets in the city. A query then retrieves the city streets and the districts that they intersect.

```
CREATE TABLE Districts
    (DistrictId int IDENTITY (1,1),
     DistrictName nvarchar(20),
     DistrictGeo geometry);
GO

CREATE TABLE Streets
    (StreetId int IDENTITY (1,1),
     StreetName nvarchar(20),
     StreetGeo geometry);
GO

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Downtown',
      geometry::STGeomFromText
           ('POLYGON ((0 0, 150 0, 150 150, 0 150, 0 0))', 0));

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Green Park',
      geometry::STGeomFromText
           ('POLYGON ((300 0, 150 0, 150 150, 300 150, 300 0))', 0));

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Harborside',
      geometry::STGeomFromText
           ('POLYGON ((150 0, 300 0, 300 300, 150 300, 150 0))', 0));

INSERT INTO Streets (StreetName, StreetGeo)
VALUES ('First Avenue',
      geometry::STGeomFromText
           ('LINESTRING (100 100, 20 180, 180 180)', 0))
GO

INSERT INTO Streets (StreetName, StreetGeo)
VALUES ('Mercator Street',
      geometry::STGeomFromText
           ('LINESTRING (300 300, 300 150, 50 50)', 0))
```

```
GO

SELECT StreetName, DistrictName
FROM Districts d, Streets s
WHERE s.StreetGeo.STIntersects(DistrictGeo) = 1
ORDER BY StreetName
```

The results from this query are shown in the following table.

Query Results

StreetName	DistrictName
First Avenue	Downtown
First Avenue	Harborside
Mercator Street	Downtown
Mercator Street	Green Park
Mercator Street	Harborside

Connecting with Query Editor

SQL Server Management Studio permits you to write or edit code while disconnected from the server. This can be useful when the server is not available or when you want to conserve scarce server or network resources. You can also change the connection of Query Editor to a new instance of SQL Server without opening a new Query Editor window or retyping your code.

To write code offline and then connect to different servers

1. On the Management Studio toolbar, click **Database Engine Query** to open the Query Editor.
2. In the **Connect to Database Engine** dialog box, click **Cancel**. The Query Editor opens, and the title bar for the Query Editor indicates that you are not connected to an instance of SQL Server.
3. In the code pane, type the following Transact-SQL statements:

```
DECLARE @g geometry = 'LINESTRING(9 9, 40 40)'
DECLARE @h geometry = 'POLYGON((15 15, 15 30, 30 30, 30 15, 15 15))'
SELECT @g.STDifference(@h).ToString();
```

At this point you can connect to an instance of SQL Server by clicking **Connect**, **Execute**, **Parse**, or **Display Estimated Execution Plan**, all of which are available from either the **Query** menu, the Query Editor toolbar, or from the shortcut menu when you right-click in the Query Editor window. For this practice, we'll use the toolbar.

4. On the toolbar, click the **Execute** button to open the **Connect to Database Engine** dialog box.
5. In the **Server name** text box, type your server name.

Test the Samples

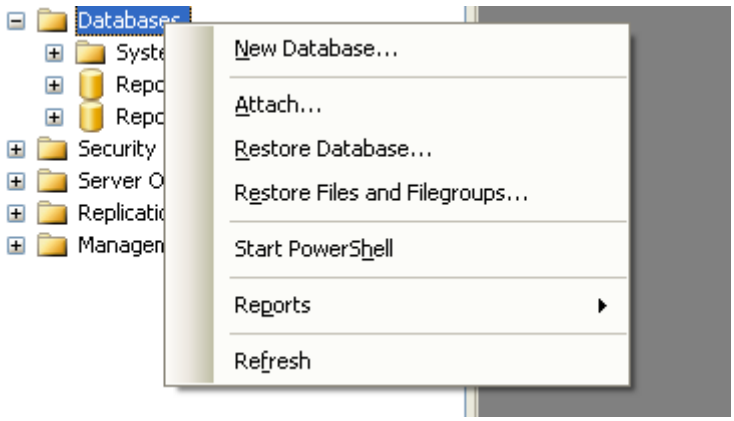
First, try the example in the first Section "Spatial Data Type Methods", then try the samples out from the slides. Copy each one to the Query Editor and Execute one at a time. Compare the results with the expected outcome.

Creating a Database

Once SQL Server 2008 express is installed with management tools do the following

1. On windows Inicio->Todos los programas->Microsoft SQL Server 2008->SQL Server Management Studio.
2. You should see something like *computernamehere\SPATIAL* and now login with sa (Standard mode) or just the windows account assuming you gave current user admin rights.

Select Databases -> Right mouse click -> New Database



3. Give the database the name MTIG and click the **OK** button.

Loading GIS Data Into the Database

Now we have a nice fully functional GIS database with no spatial data. So to do some neat stuff, we need to get some data to play with.

First off we have to install a loader. You can use the freely available http://www.sharpgis.net/page/SQL-Server-2008-Spatial-Tools.aspx Which comes with both an ESRI shape loader and SQL Server spatial viewer. To use simply download and extract.

Get the Data

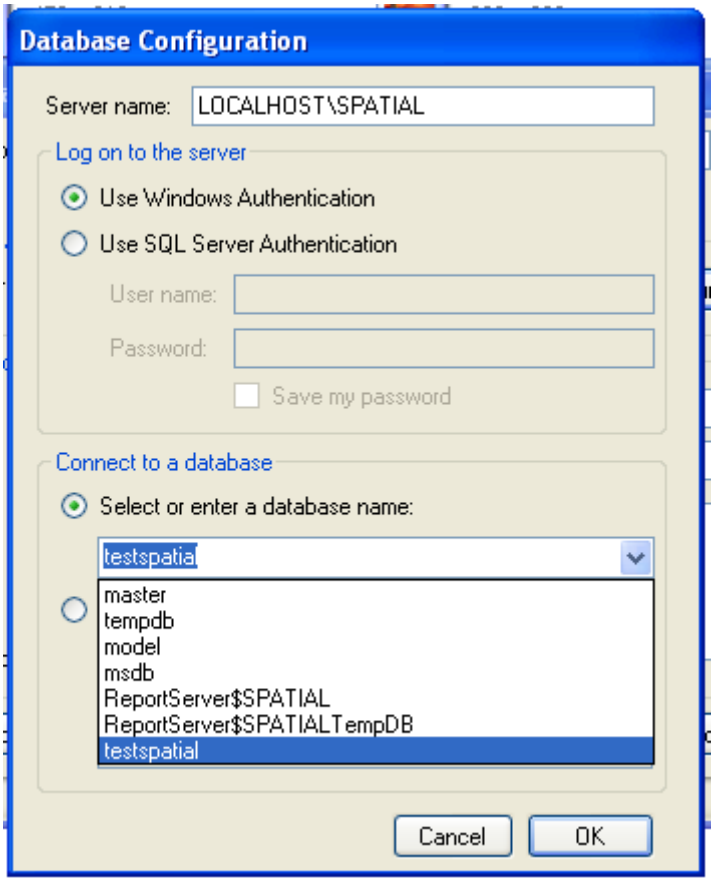
Download data from the MassGIS site. For this simple exercise just download Towns with Coast:
ftp://data.massgis.state.ma.us/pub/shape/state/towns.exe

Extract the file into some folder. We will only be using the _POLY files for these exercises.

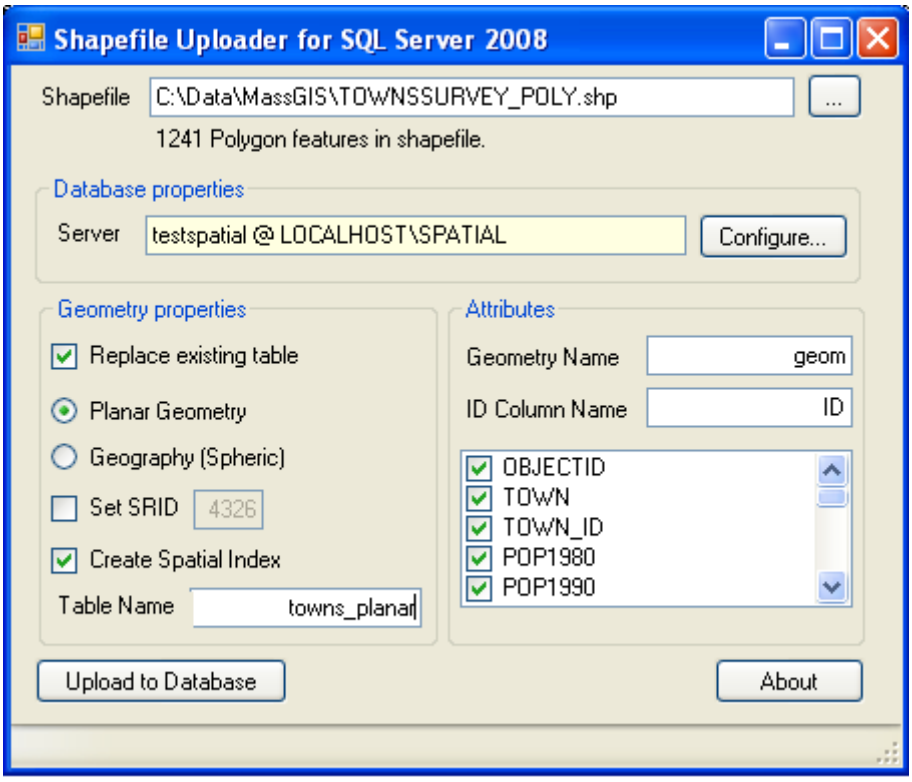
Bringing in Towns As Planar

First of all, what is Planar - Planar is a class of spatial reference systems that projects a round earth onto a flat model. SQL Server 2008 supports both a Geometry (Planar) and a Geography (round-earth model). For data brought in as Planar, SQL Server 2008 does not do any validation to ensure it is in the sys.spatial_reference_systems table, and in fact SQL Server 2008 only contains spherical spatial reference systems in that meta table. So if you want to bring in as planar, as long as all your data is in the same planar projection, you should be fine. SQL Server 2008 has no mechanism of transforming data from one planar projection to another.
Those who are familiar with the PostGIS equivalent exercise of this know that MassGIS data is in Massachusetts state plane Meters (Spatial_Reference_ID = 26986 which is a planar projection) so bringing it in as Geometry works fine, but trying to push it into Geodetic we shall find is a little trickier.
Now lets move some data:

1. Launch the *Shape2Sql.exe* packaged in the SharpGIS tools zip file
2. Your screen should look something like this



3. Point at the towns file you downloaded - Your screen should look something like this when you are done:



4. Uncheck Create Spatial Index
5. Now click the Upload to Database

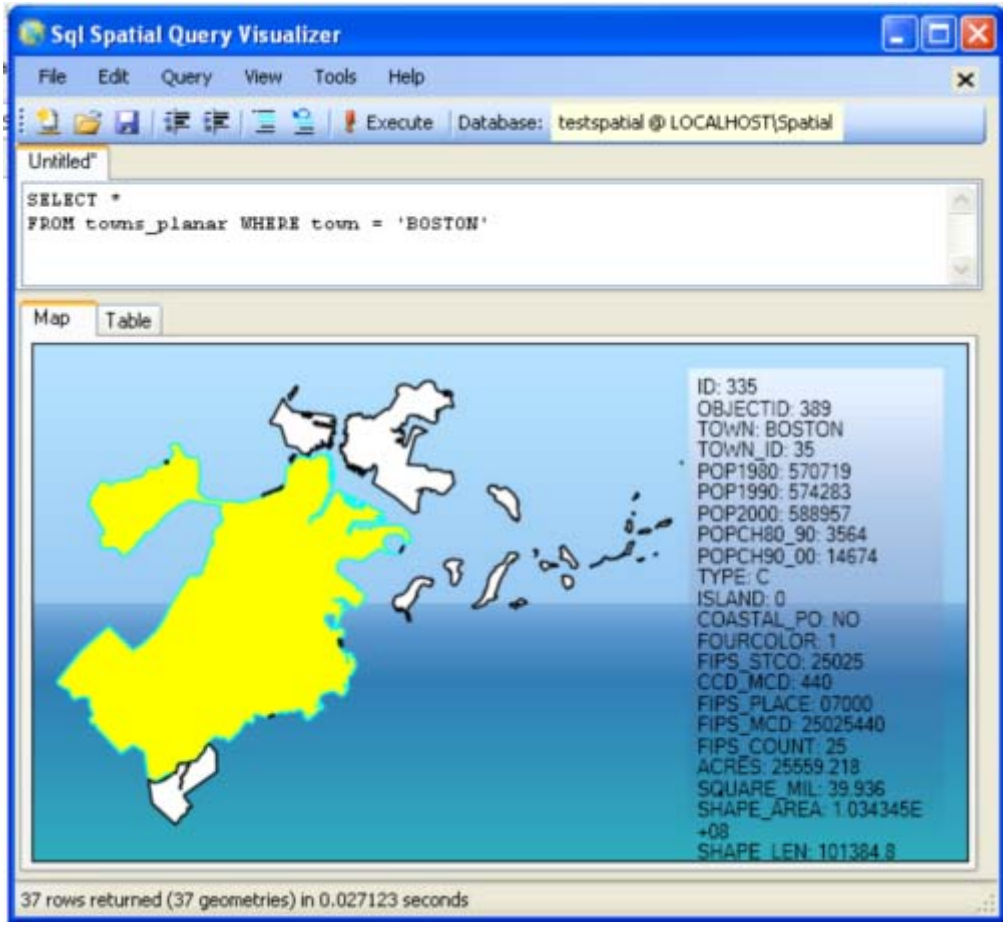
Querying the data and visualizing it

What good is spatial data if you can't drop your jaws at its fantastic beauty. So lets look at what this animal looks like:

1. Launch the SQLSpatial.exe which is also packaged in the SharpGIS tools.
2. Type in the following SQL statement:

```
SELECT * FROM towns_planar WHERE town = 'BOSTON'
```

3. Click the **!Execute** button, and mouse over a geometry and you should see something like this:



4. File New Query and type this:

```
SELECT TOP 1 geom.STCentroid().STAsText() FROM towns_planar
WHERE town = 'BOSTON'
```

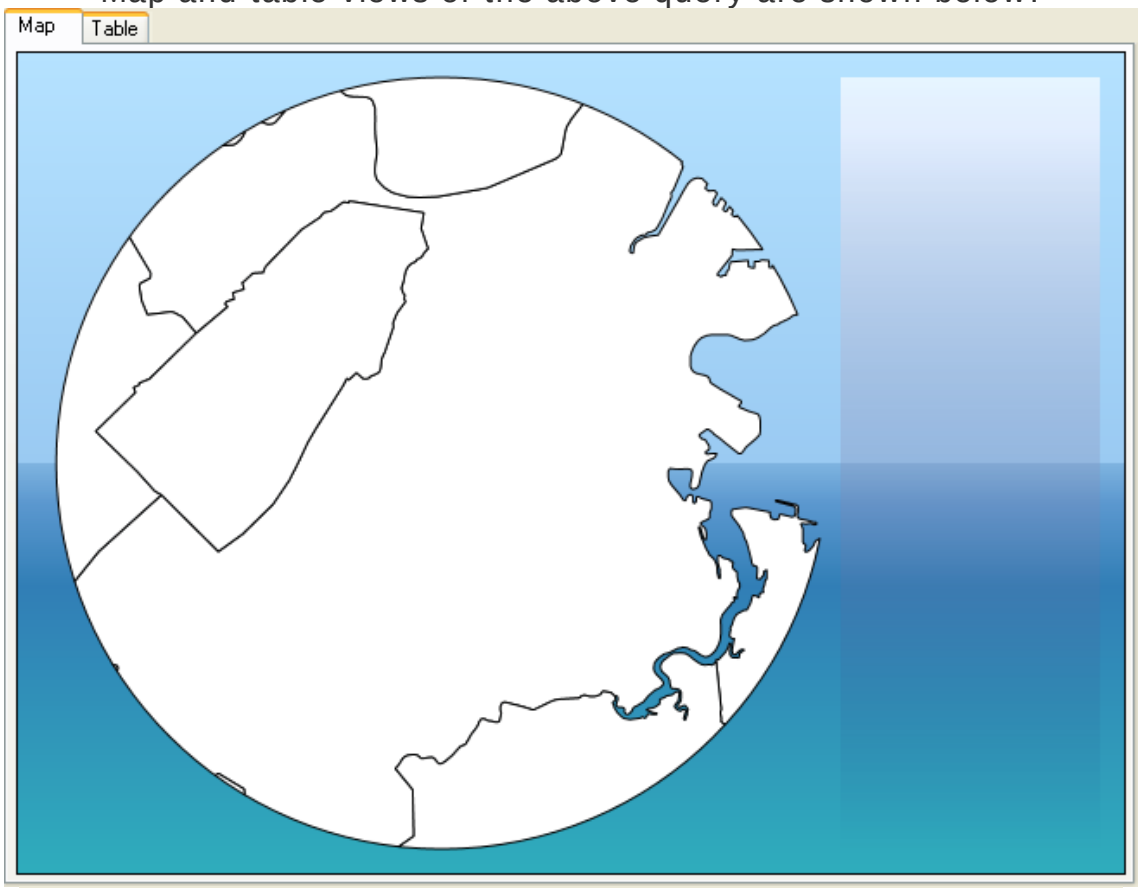
Should toggle to the table view and give you this:

POINT (232749.48473676728 895447.62452343013)

5. Now lets pick the first geometry in Boston, find the centroid, buffer the centroid 7000 meters and find all fragments of towns in the buffer. People familiar with spatial queries will recognize this as clipping geometries to a buffer.
File-> New Query and do this: - evidentally there are some Massachusetts towns that SQL Server doesn't like thus the need for the IsValid check.

```
SELECT town, geom.STIntersection(buf.aBuffer) As newgeom
FROM towns_planar INNER JOIN
(SELECT TOP 1 geom.STCentroid().STBuffer(7000) As aBuffer
FROM towns_planar WHERE town = 'BOSTON') As buf
ON (towns_planar.geom.STIntersects(buf.aBuffer) = 1)
WHERE geom.STIsValid() = 1
```

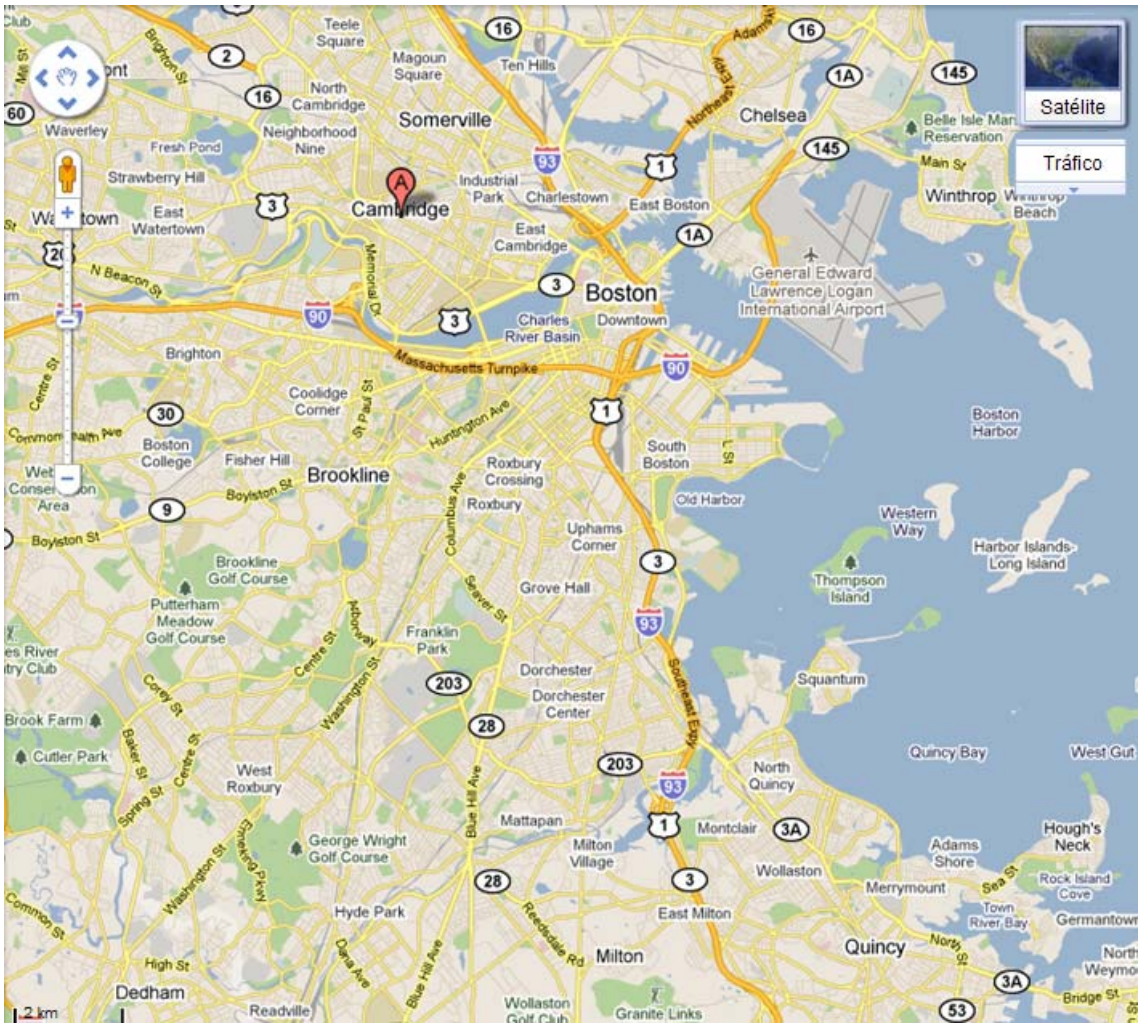
Map and table views of the above query are shown below:



Map | Table

	town	newgeom
▶	BOSTON	POLYGON ((231973.69904296211 888491.39859053551, 232246.7336425781...
	WATERTOWN	POLYGON ((228773.56081612138 901207.93344519113, 228785.6538085937...
	NEWTON	POLYGON ((226085.05990538746 893304.36202476593, 226519.3237304687...
	BROOKLINE	POLYGON ((228695.82348632813 893844.81079101562, 229125.7036132812...
	QUINCY	POLYGON ((237889.99728268082 890697.96425473469, 237931.01171875 8...
	DEDHAM	MULTIPOLYGON (((226855.67299943022 891673.155365011, 226864.313476...
	MILTON	POLYGON ((232749.48461914063 888447.62451171875, 233109.7036132812...
	CAMBRIDGE	MULTIPOLYGON (((229607.23364257813 901596.31103515625, 229630.1135...

Compare this with:



Bringing in Towns As Geodetic

If you have data measured in degrees e.g. WGS84 longlat (4326) or NAD 83 LongLat (4269 standard TIGER US Census format), bringing in your data as geodetic is simple since 4326 and 4269 are already listed in the sys.spatial_reference_systems. A simple query confirms that –

```
SELECT * FROM sys.spatial_reference_systems WHERE spatial_reference_id
IN(4269,4326);
```

Looking at the Table tab:

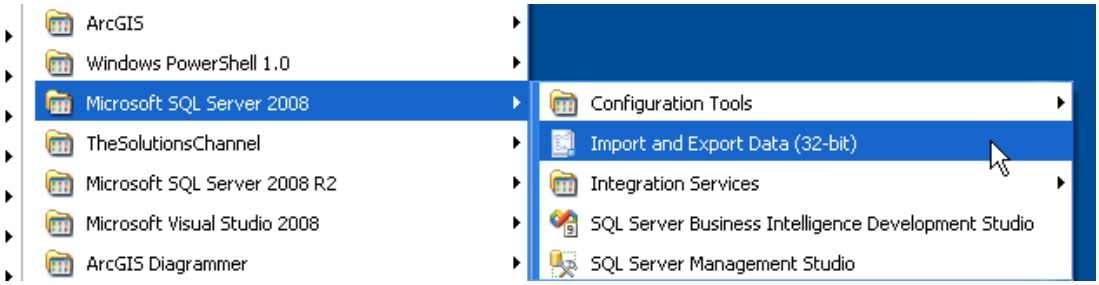
To do so - you simply follow the prior steps but choose **Geography (Spheric)** instead.

Repeat the same queries under Management Studio

Repeat the previous SELECT statements under SQL Management Studio. Look at the **Results** and **Spatial results** tabs.
Notice that you can zoom and drag the image. If you rest the mouse pointer on any surface, you get additional data.

Import and Export SQL Server data

While SQL Server can be used to house the data that your company maintains on a daily basis and offers a way to effectively and efficiently manage the objects and data within that database, there are often times when getting initial data into a database can be a hassle.
However, not only does SQL Server manage the data and objects contained within the database, it has tools that can be used to get data into tables quickly from a variety of external sources.
SQL Server provides to those who use it, the ability to import and export in SQL Server to many different formats. This can be extremely useful for someone wanting to quickly get data into the database or out of the database for any reason.
For example, an agency that sells mailing lists could easily export the data from a SQL Server database to a format that could be burned onto a CD and delivered to the customer very quickly.
On the other hand, the same company could buy a mailing list file from another agency or third party and import the data into their database and save the time of having to key the data into the database or an application.
So how can the import and export in SQL Server data tools that are provided by SQL Server be used to get data into a database quickly?
In order to use the import and export in SQL Server Data functions, select Start -> All Programs -> Microsoft SQL Server 2008 -> Import and Export Data as shown in the image below:

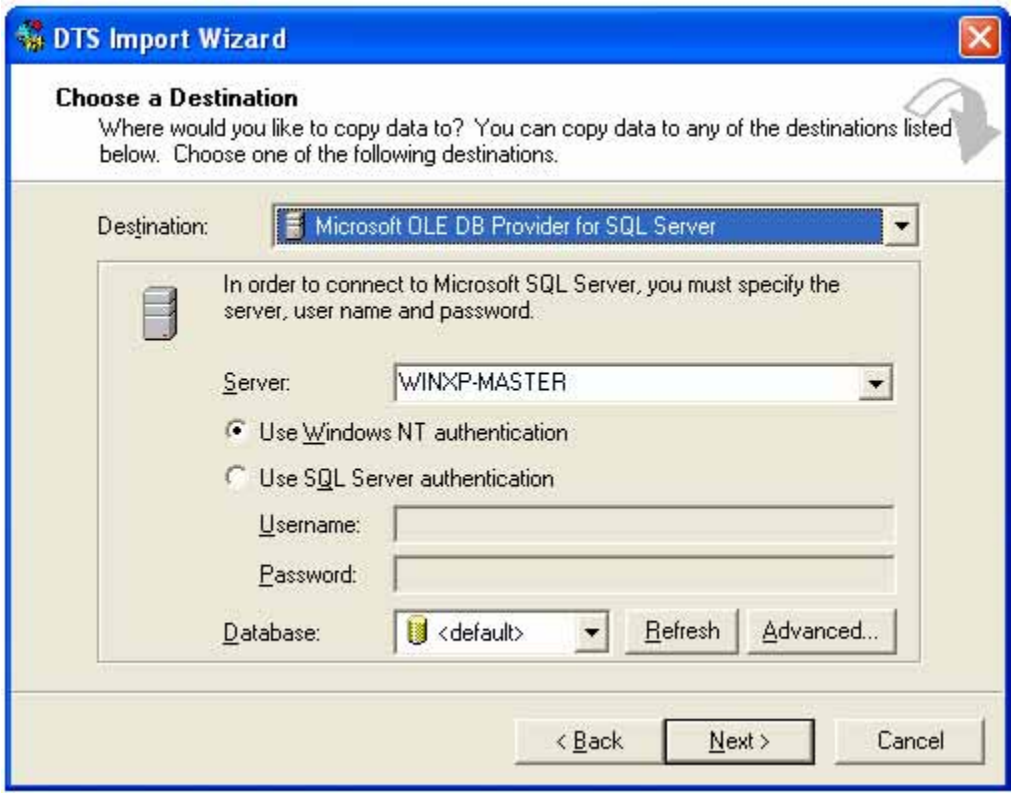


The interface for Import Data and Export Data is the same, each presents the user with different options for the destination and source for the data involved in the operation. Regardless of what operation you wish to conduct when transferring the data, an import or an export, each option will provide you with the functionality you need.
For clarification, regardless of whether you want to import or export data, the options are provided that will allow an import or an export to occur once the operation is initiated. Once the selection is made, the following window appears:

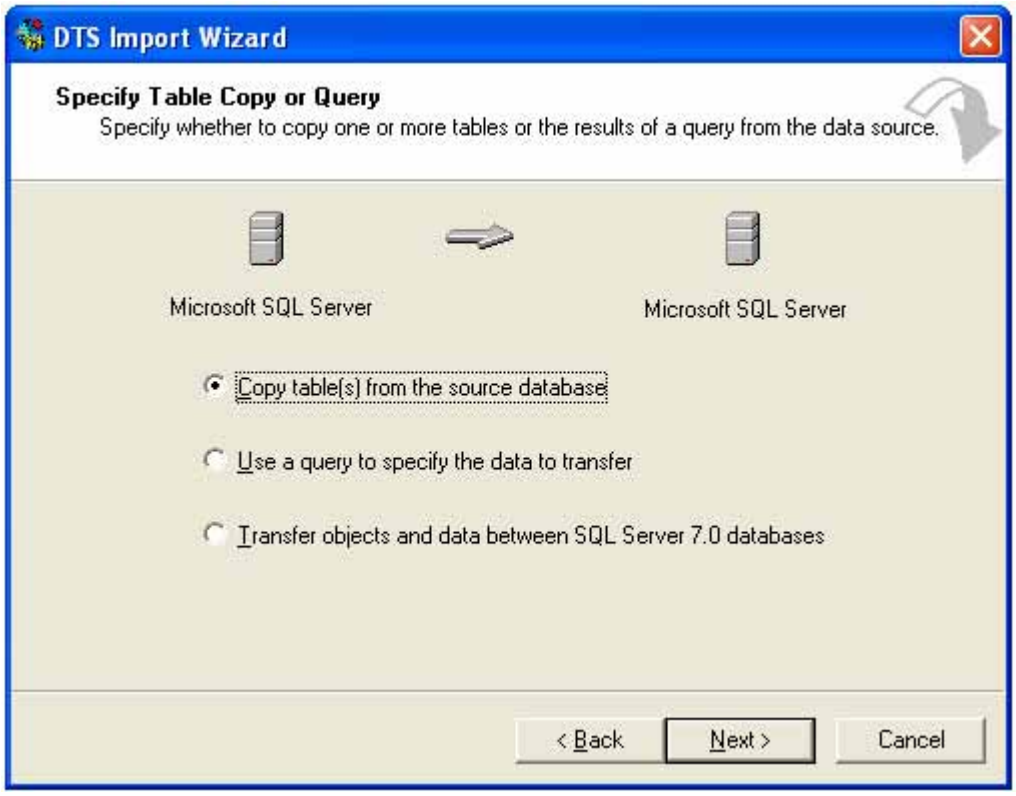
Select the Next button to continue with the Import/Export Wizard. This will bring up the following screen:

This screen will allow the user to select the data source, which is a drop down list of all the types of data that can be used with the import/export.
These include many different formats, not just SQL Server data. The source type can be SQL Server, Access, Oracle or any other data store that can be accessed via ODBC as

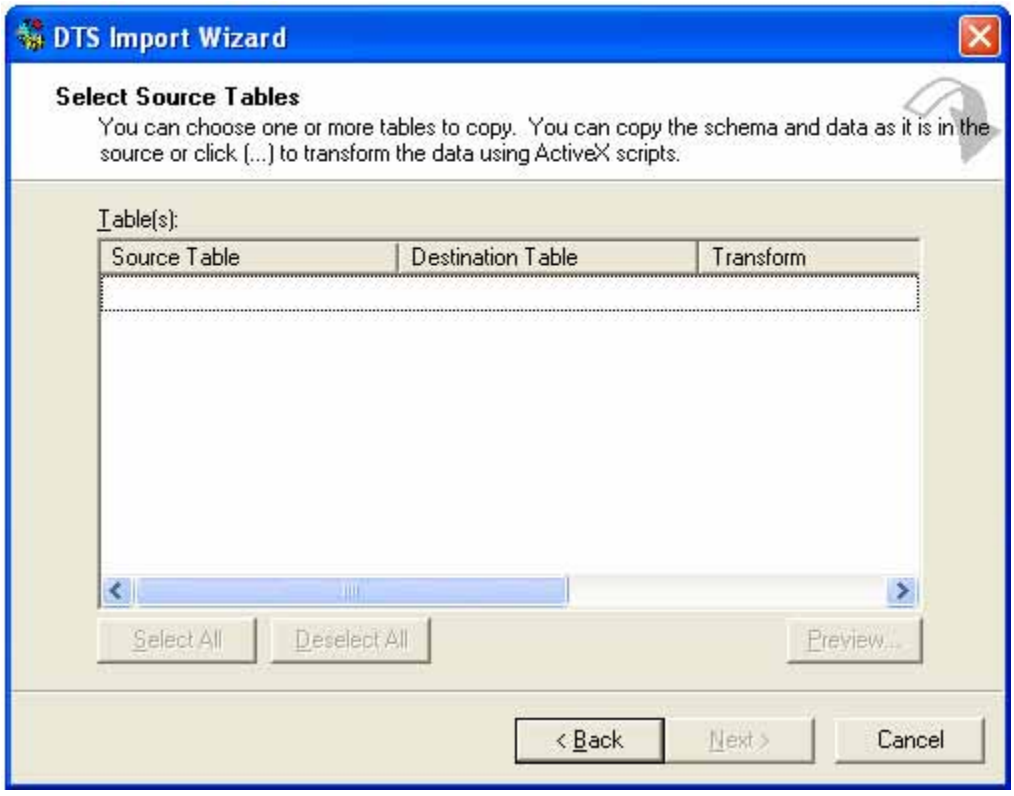
well. After the data source type is selected, the user is then presented with the specifications needed to specify the location of the source data.
Each option comes with different parameters needed to locate, connect and open the data source. Once the data source is configured, click next which will allow the selection of the destination for the data.
In most instances you will be required to input information that will allow the database to verify who you are and securely control access to that particular repository of data. Each type of data has differing parameters and must be experimented with in order to learn the full functionality of the import/export capabilities of SQL Server and the tools it offers. The following graphic illustrates this screen:



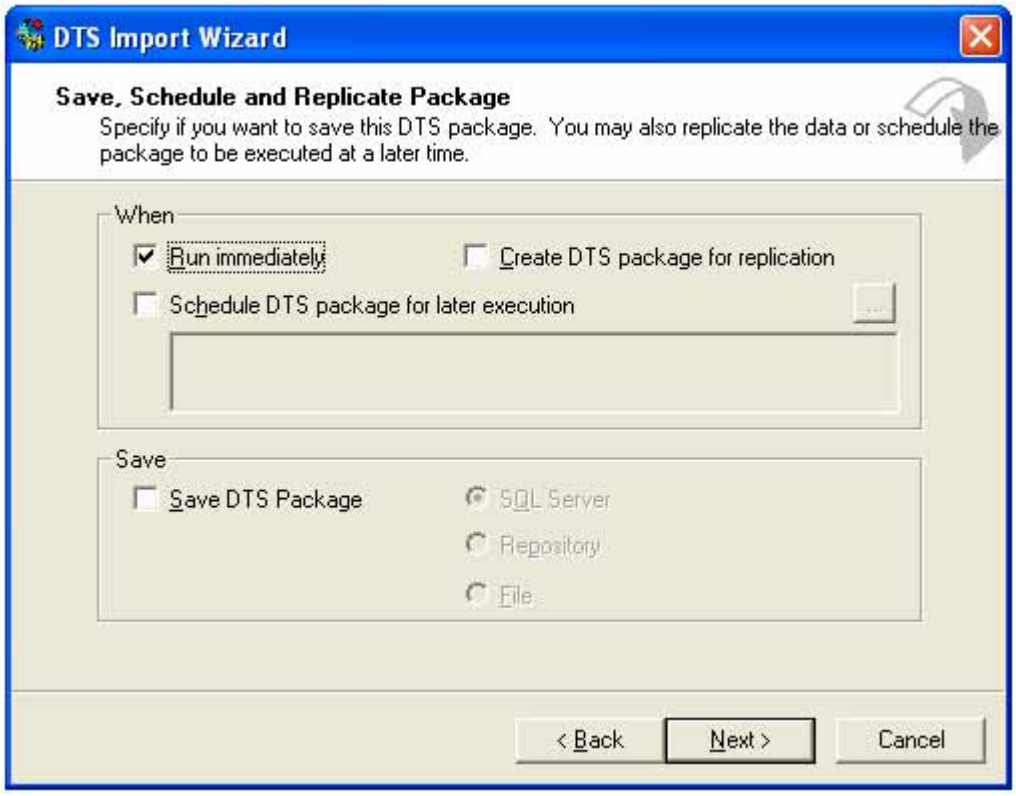
This screen is used the same way as the previous one used to input the source of the information for the import and export in SQL Server.
It functions the exact same way the source screen works except that it represents the connection used to access the destination database. If the user wants to go back to the source screen, click the back button. Clicking next will bring up the next screen.
This will allow the user to select the type of objects that they want to move.



The user can select to move tables, use a query to specify the data or move data and objects between databases.
Selecting the copy tables option will allow the user to select tables to move in their entirety to the destination. The use of the query option will allow the user to enter a query to specify the data that needs be moved. This can be used to limit the fields or the amount of data being moved.
The transfer objects option will allow the selection of objects to be moved between SQL Server databases.
All of these options are very important when trying to determine how to import and export data in SQL Server. After the choice is made, the user can click next to move on to the next screen, which is shown below.



To continue with the import and export in SQL Server operation, this screen will allow the selection of the source and destination objects and the data transformations that need to occur in order to move the data to the correct location.
These transformations can be a straight copy or they can be the result of data manipulation via scripts in order to make the data fit the format you need or transform the data into other elements. Once this is complete, click the Next button, which will bring up the following screen.
After this information is input, our import/export operation is almost ready to run on the server to complete the movement of the data.



This will allow the user to run the import/export now or save the configured import/export as a DTS package to be run later or saved on the server to be run as needed.
You may be asking yourself why this utility to import and export in SQL Server is so advantageous, imagine if you had to use data entry tasks in order to manually enter a list of thousands of records of data.
How much time could be saved if the data could be loaded via a file in the import/export data utility?
Experience has proven to me that this tool can prove to be invaluable if the developer or database administrator becomes familiar with the power that it provides. It has saved me numerous hours on projects and has been used exclusively for data transformation tasks that would not have been possible without a tool such as this.
Moving data using the import and export in SQL Server can prove very useful along with the rest of the DTS features in SQL Server.

Copy databases with the Wizard

Warning: SQL Server 2008 Express does not include this wizard.
The Copy Database Wizard lets you move or copy databases and their objects easily from one server to another, with no server downtime. Using this wizard, you can do the following:

- Pick a source and destination server.
- Select databases to move or copy.
- Specify the file location for the databases.
- Create logins on the destination server.
- Copy additional supporting objects, jobs, user-defined stored procedures, and error messages.

- Schedule when to move or copy the databases.

In addition to copying databases, you can copy associated metadata, for example, logins and objects from the **master** database that are required by a copied database.

Note

The **model**, **msdb**, and **master** databases cannot be copied or moved by the Copy Database Wizard.

Also, you can move and copy databases between different instances of SQL Server, and you can upgrade databases from SQL Server 2000 to SQL Server 2005 or later. The destination server must be SQL Server 2005 or later. For more information, see "Upgrading SQL Server by Using the Copy Database Wizard" later in this topic.

Issues to Consider

Consider the following issues before you use the Copy Database Wizard.

Area	Consideration
Required permissions	You must be a member of the **sysadmin** fixed server role on both the source and destination servers.
Required components	SQL Server 2005 Integration Services (SSIS) or later.
model, **msdb** and **master** databases	The **model**, **msdb**, and **master** databases cannot be copied or moved by the Copy Database Wizard.
Database on source server	If you select the **Move** option, the wizard deletes the source database automatically after moving the database. The Copy Database Wizard does not delete a source database if you select the **Copy** option.
Full-text catalogs	If you use the SQL Server Management Object method to move the full-text catalog, you must repopulate the index after the move. If you use the detach-and-attach method, full-text catalogs must be moved manually. For more information about how to move full-text catalogs, see Moving Database Files.

Starting the Copy Database Wizard

In SQL Server Management Studio, in Object Explorer, expand **Databases**, right-click a database, point to **Tasks**, and then click **Copy Database**.

Copying and Moving Databases

To use the Copy Database Wizard, you must specify the following:

- The source server where the databases to be copied reside.
- The destination server to which the databases are to be copied or moved.
- The databases to be moved or copied.
- The name of a target database, if different than the name of the source database.

The source database name can be use for the copied or moved database only if name conflicts do not exist on the destination server. If name conflicts exist, you must resolve them manually on the destination server before you can use the source database name there.

- Other objects to be copied or moved; for example, logins, shared objects from the **master** database, jobs and maintenance plans, and user-defined error messages.
- The schedule for the copy or move operation, if you want it to run at a later time.
- If you are not a system administrator, you must specify a SQL Server Agent Proxy account that has access to the Integration Services (SSIS) Package execution subsystem.

The detach-and-attach method, detaches the database, moves or copies the database .mdf, .ndf, .ldf files and reattaches the database in the new location. For the detach-and-attach method, to avoid data loss or inconsistency, active sessions cannot be attached to the database being moved or copied. If any active sessions exist, the Copy Database Wizard does not execute the move or copy operation.

Note

For the SQL Server Management Object method, active sessions are allowed because the database is never taken offline.

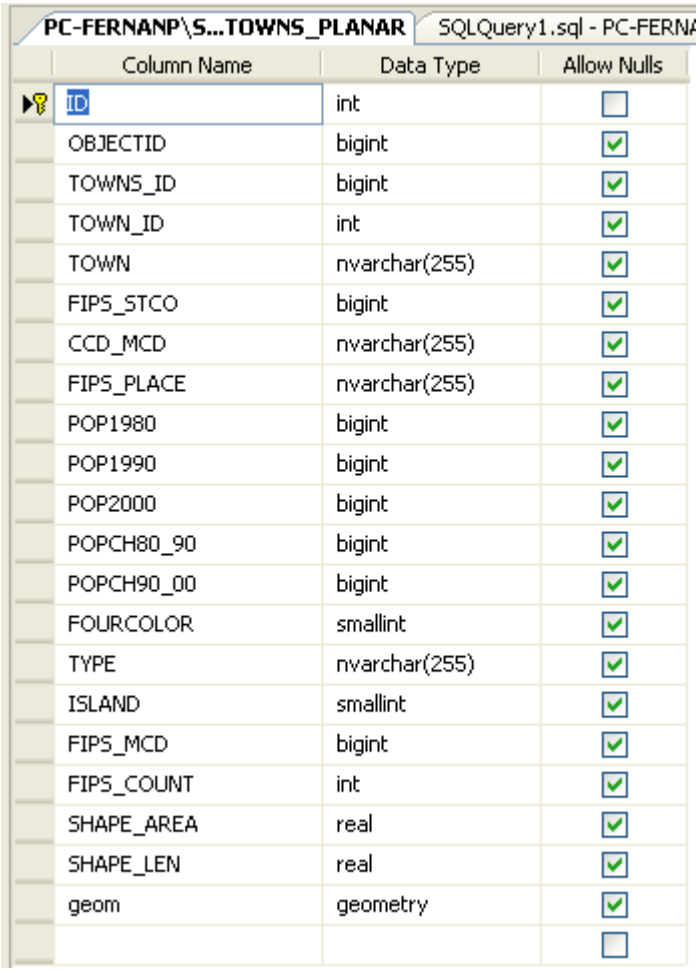
When moving databases between different servers or disk drives, the Copy Database Wizard copies the database to the destination server and verifies that it is online. When moving databases between two instances on the same server, the file system move operation is performed.

Managing Metadata When Restoring to Another Server Instance

When you copy a database to another server instance, to provide a consistent experience to users and applications, you might have to re-create some or all of the metadata for the database, such as logins and jobs, on the other server instance.

Tabla TOWNS_PLANAR

Dado que no se puede usar el asistente de copia de base de datos y la exportación no reconoce el tipo geometry, se puede definir es esquema de la tabla y después incorporar los datos desde una hoja Excel. Los datos se pueden encontrar en el archivo Datos Práct. 5. TOWNS_PLANAR.xls. Y el esquema de la tabla es como se muestra en la siguiente figura:



PC-FERNANP\S...TOWNS_PLANAR | SQLQuery1.sql - PC-FERNA

Column Name	Data Type	Allow Nulls
ID	int	☐
OBJECTID	bigint	☑
TOWNS_ID	bigint	☑
TOWN_ID	int	☑
TOWN	nvarchar(255)	☑
FIPS_STCO	bigint	☑
CCD_MCD	nvarchar(255)	☑
FIPS_PLACE	nvarchar(255)	☑
POP1980	bigint	☑
POP1990	bigint	☑
POP2000	bigint	☑
POPCH80_90	bigint	☑
POPCH90_00	bigint	☑
FOURCOLOR	smallint	☑
TYPE	nvarchar(255)	☑
ISLAND	smallint	☑
FIPS_MCD	bigint	☑
FIPS_COUNT	int	☑
SHAPE_AREA	real	☑
SHAPE_LEN	real	☑
geom	geometry	☑
		☐