Chapter 4

Creating Spatial Data

Necessary Information

```
Datatype::Method( Coordinates, SRID )
```

- Spatial Reference System
  - Example: 4326 (WGS84)
- Type of geometry
  - `Geometry / Geography`
- Datatype
  - `Point, LineString, Polygon`, etc.
- Coordinates
  - `‘Point(23,32, 4326)’`

Example

```
SELECT
geography::STPointFromText(‘POINT(153 -27.5)’, 4326);
```

Methods of Creation

- Directly Create SQL Server `SQLGeometry` type
  - `geography::Point(40, -100, 4269)`
- Parse from several formats using geometry methods
  - WKT (Well Known Text)
  - WKB (Well Known Binary)
  - GML (Geography Markup Language
- API to build programmatically
  - Classes `SqlGeometryBuilder` and `SqlGeographyBuilder`

Well-Known Text Methods

- Simple format that we have seen in `sys.spatial_reference_systems`
- Advantages:
  - Common and simple format
  - Easy to read and identify information in markup
- Disadvantages
  - Creating objects through parsing into internal binary format is slower
  - Rounding errors on floating point values being represented in text format

```
geometry::Parse('POINT(30 40)')
```

↓

```
0x00000000010C0000000000003E400000000000004440
```

SQLQuery1.sql - WIN...ael Beisiegel (52))* | SQLQuery2.sql - WIN...ael Beisiegel (53))* | GeocacheCentralTest...ael Beisiegel (54)) | Schema.sql - WIN-SI...nael Beisiegel (60))*

```sql
SELECT *
FROM sys.spatial_reference_systems
```

100 %

Results | Messages

	spatial_reference_id	authority_name	authorized_spatial_reference_id	well_known_text	unit_of_measure	unit_conversion_factor
159	4299	EPSG	4299	GEOGCS["TM65", DATUM["TM65", ELLIPSOID["Airy Modif...	metre	1
160	4300	EPSG	4300	GEOGCS["TM75", DATUM["Geodetic Datum of 1965", ELL...	metre	1
161	4301	EPSG	4301	GEOGCS["Tokyo", DATUM["Tokyo", ELLIPSOID["Bessel 1...	metre	1
162	4302	EPSG	4302	GEOGCS["Trinidad 1903", DATUM["Trinidad 1903", ELLIP...	Clarke's foot	0.304797265
163	4303	EPSG	4303	GEOGCS["TC(1948)", DATUM["Trucial Coast 1948", ELLIP...	metre	1
164	4304	EPSG	4304	GEOGCS["Voirol 1875", DATUM["Voirol 1875", ELLIPSOID[...	metre	1
165	4306	EPSG	4306	GEOGCS["Bern 1938", DATUM["Bern 1938", ELLIPSOID["...	metre	1
166	4307	EPSG	4307	GEOGCS["Nord Sahara 1959", DATUM["Nord Sahara 1959...	metre	1
167	4308	EPSG	4308	GEOGCS["RT38", DATUM["Stockholm 1938", ELLIPSOID[...	metre	1
168	4309	EPSG	4309	GEOGCS["Yacare", DATUM["Yacare", ELLIPSOID["Interna...	metre	1
169	4310	EPSG	4310	GEOGCS["Yoff", DATUM["Yoff", ELLIPSOID["Clarke 1880 (...	metre	1
170	4311	EPSG	4311	GEOGCS["Zanderij", DATUM["Zanderij", ELLIPSOID["Inter...	metre	1
171	4312	EPSG	4312	GEOGCS["MGI", DATUM["Militar-Geographische Institut", E...	metre	1
172	4313	EPSG	4313	GEOGCS["Belge 1972", DATUM["Reseau National Belge 1...	metre	1
173	4314	EPSG	4314	GEOGCS["DHDN", DATUM["Deutsches Hauptdreiecksnetz...	metre	1
174	4315	EPSG	4315	GEOGCS["Conakry 1905", DATUM["Conakry 1905", ELLIP...	metre	1
175	4316	EPSG	4316	GEOGCS["Dealul Piscului 1933", DATUM["Dealul Piscului 1...	metre	1
176	4317	EPSG	4317	GEOGCS["Dealul Piscului 1970", DATUM["Dealul Piscului 1...	metre	1
177	4318	EPSG	4318	GEOGCS["NGN", DATUM["National Geodetic Network", EL...	metre	1
178	4319	EPSG	4319	GEOGCS["KUDAMS", DATUM["Kuwait Utility", ELLIPSOID[...	metre	1
179	4322	EPSG	4322	GEOGCS["WGS 72", DATUM["World Geodetic System 197...	metre	1
180	4324	EPSG	4324	GEOGCS["WGS 72BE", DATUM["WGS 72 Transit Broadca...	metre	1
181	4326	EPSG	4326	GEOGCS["WGS 84", DATUM["World Geodetic System 198...	metre	1
182	4600	EPSG	4600	GEOGCS["Anguilla 1957", DATUM["Anguilla 1957", ELLIPS...	metre	1

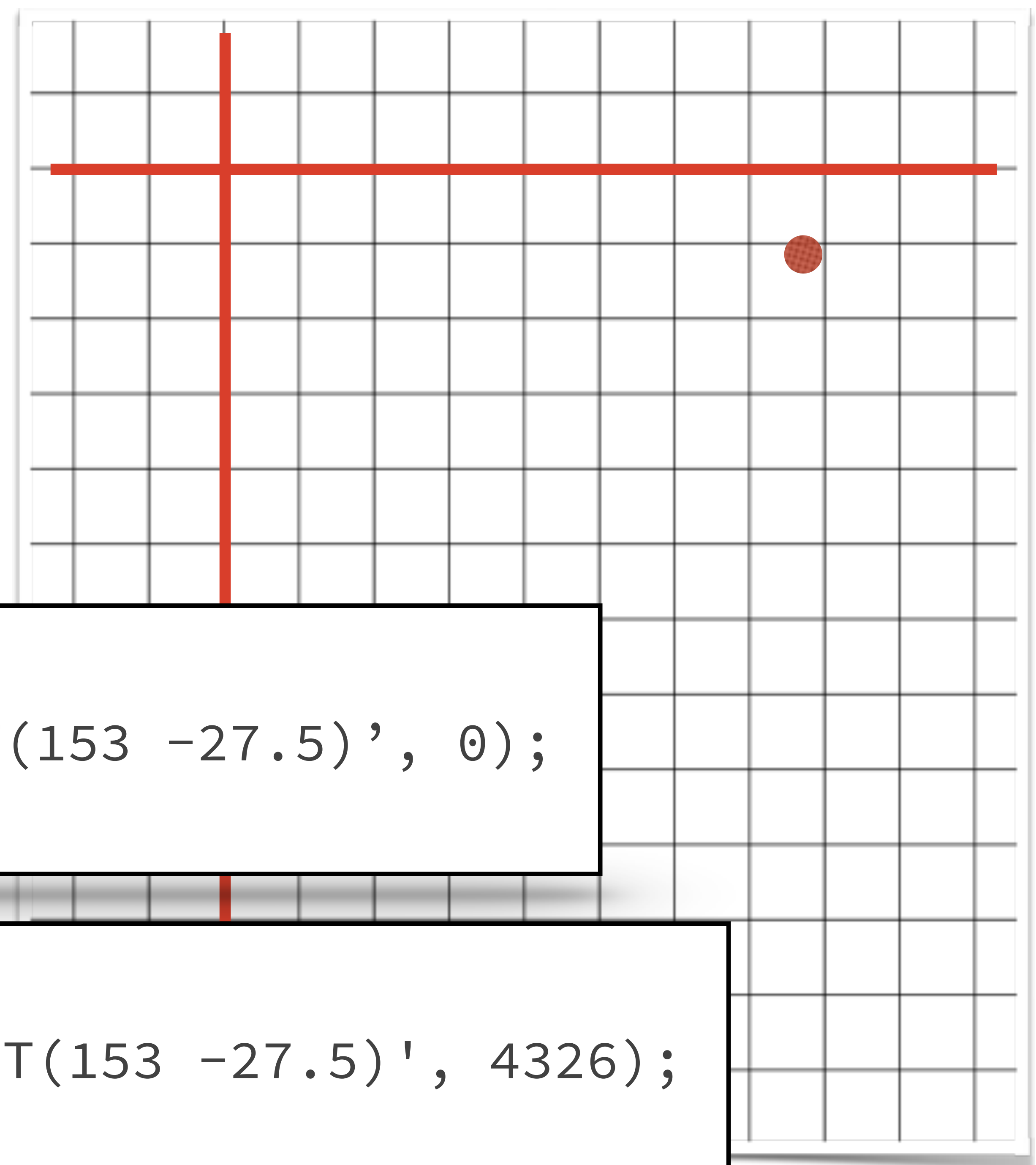
Query executed successfully. | WIN-SISEQ04RCRM\SQLEXPRESS ... | WIN-SISEQ04RCRM\Nathan...

Table 4-1. Methods to Instantiate Spatial Data from Well-Known Text

Geometry	Static Method
Point	`STPointFromText()`
LineString	`STLineFromText()`
Polygon	`STPolyFromText()`
MultiPoint	`STMPointFromText()`
MultiLineString	`STMLineFromText()`
MultiPolygon	`STMPolyFromText()`
GeometryCollection	`STGeomCollFromText()`
Any supported geometry	`STGeomFromText() / Parse()`

Parsing a Point

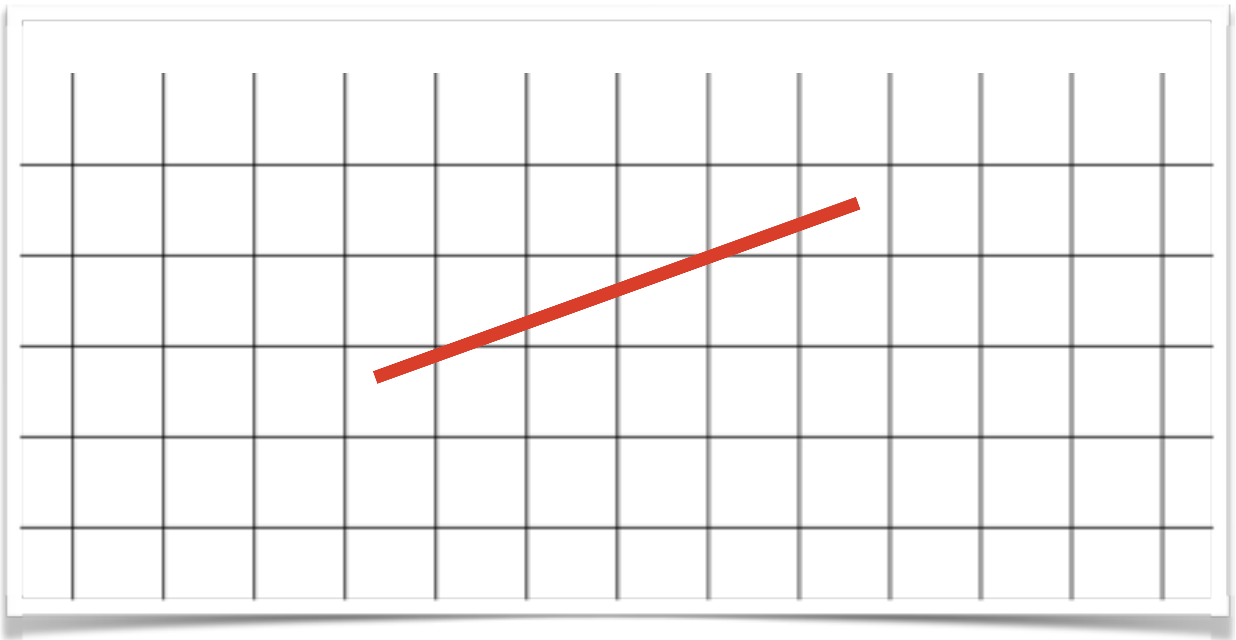
- Well-known text representation
  - `POINT(153,-27.5)`
- Calling `STPointFromText` method to parse (string of SqlChars)

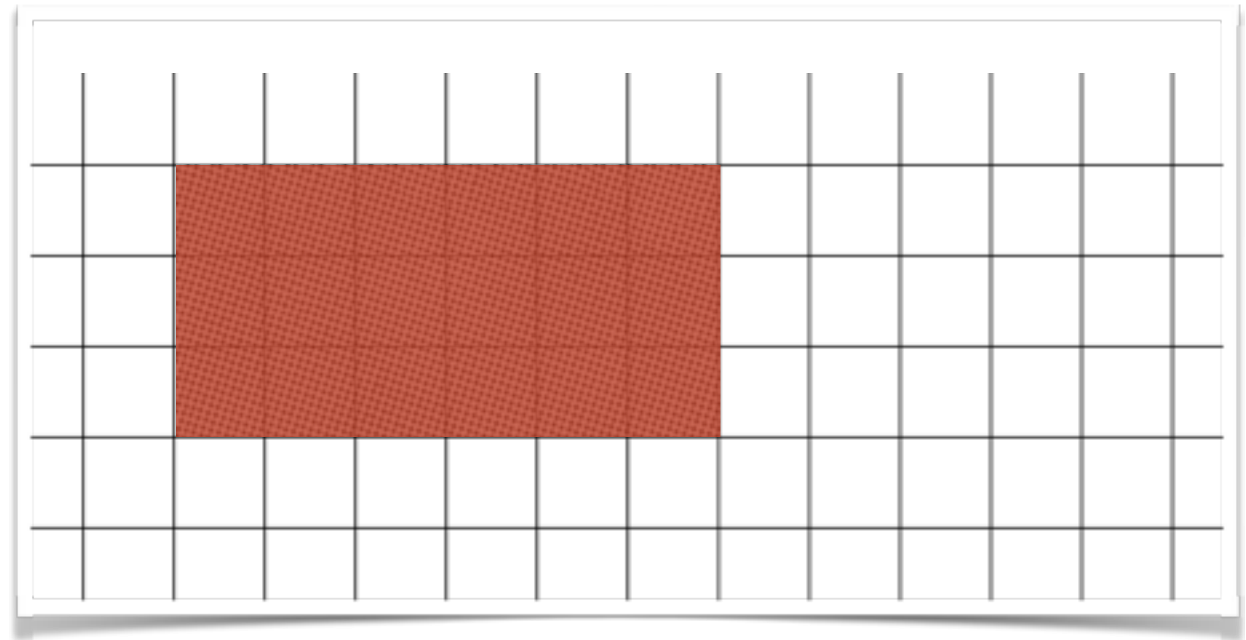


```
SELECT
geometry::STPointFromText('POINT(153 -27.5)’, 0);
```

```
SELECT
geography::STPointFromText('POINT(153 -27.5)', 4326);
```

Other Examples





```
SELECT
geometry::STLineFromText('LINESTRING(300500 600150, 310200 602500)',
27700);
```

```
SELECT
geometry::STPolygonFromText('POLYGON((1 1, 6 1, 6 4, 1 4, 1 1))',
27700);
```

Recall our Types….

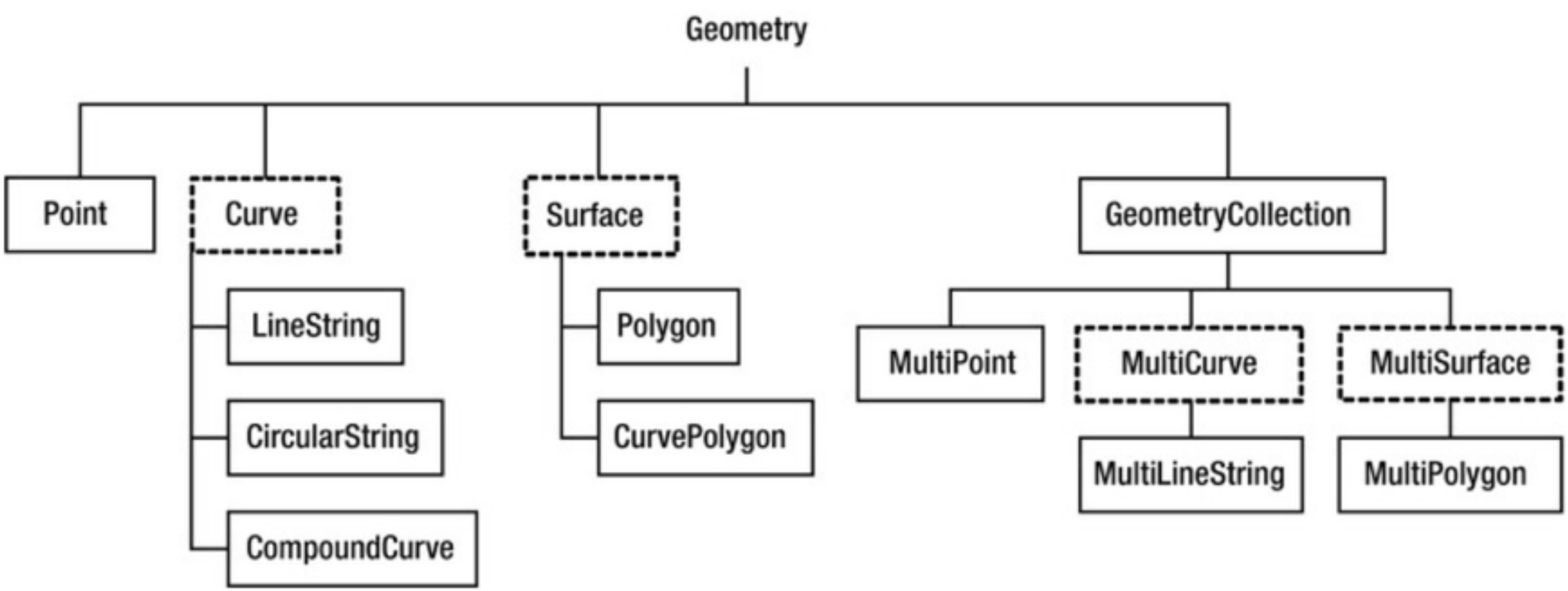


Figure 2-1. *The inheritance hierarchy of geometry types. Instantiable types (those types from which an instance of data can be created in SQL Server 2012) are shown with a solid border.*

Recall our Types….

- More generic method can be used for parsing
  - STGeomFromText
- Useful for parsing variety of WKTs into one table of geometry / geography type
- Even more generic if using SRID 0 or 4326 (WGS84): Parse
- Parse is called by default if we try to set geometry field equal to just a character string

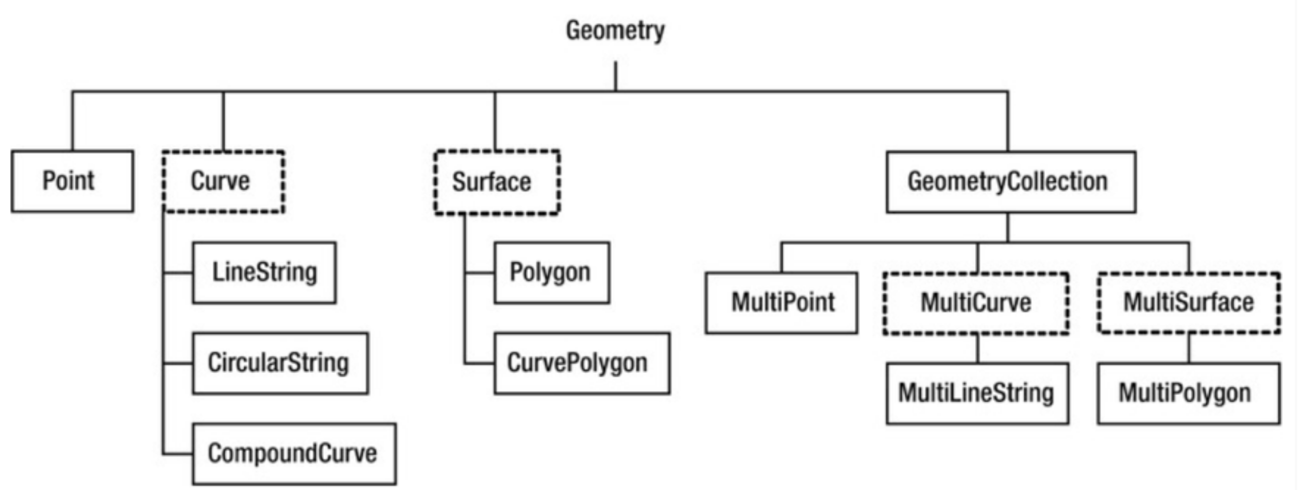
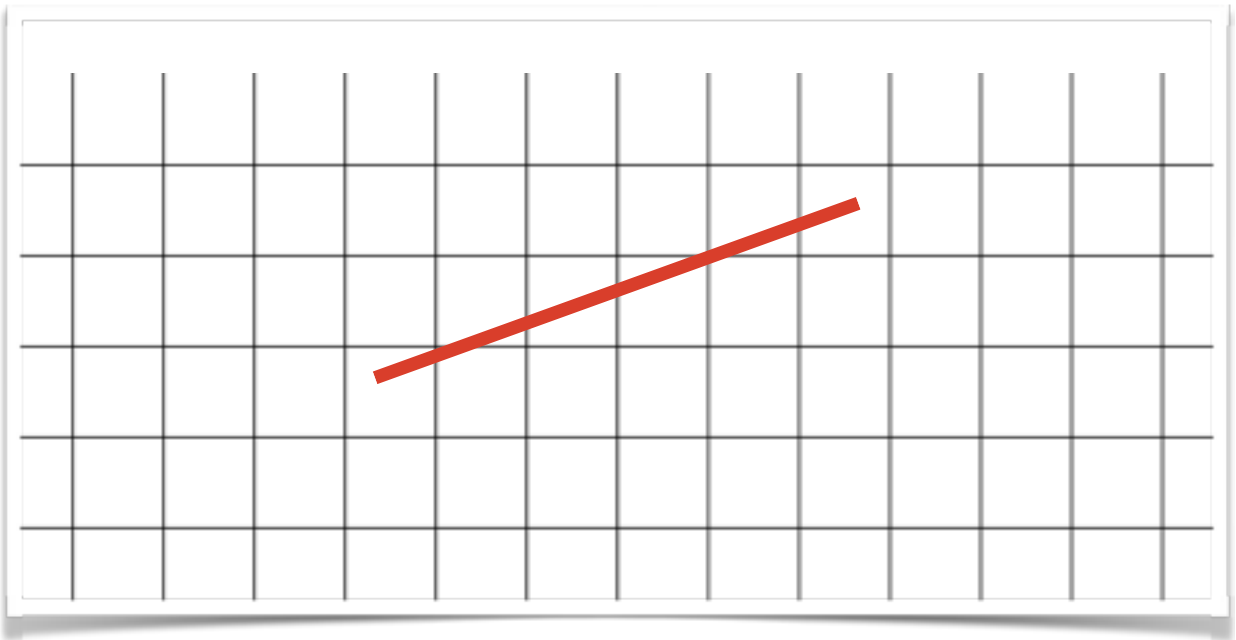
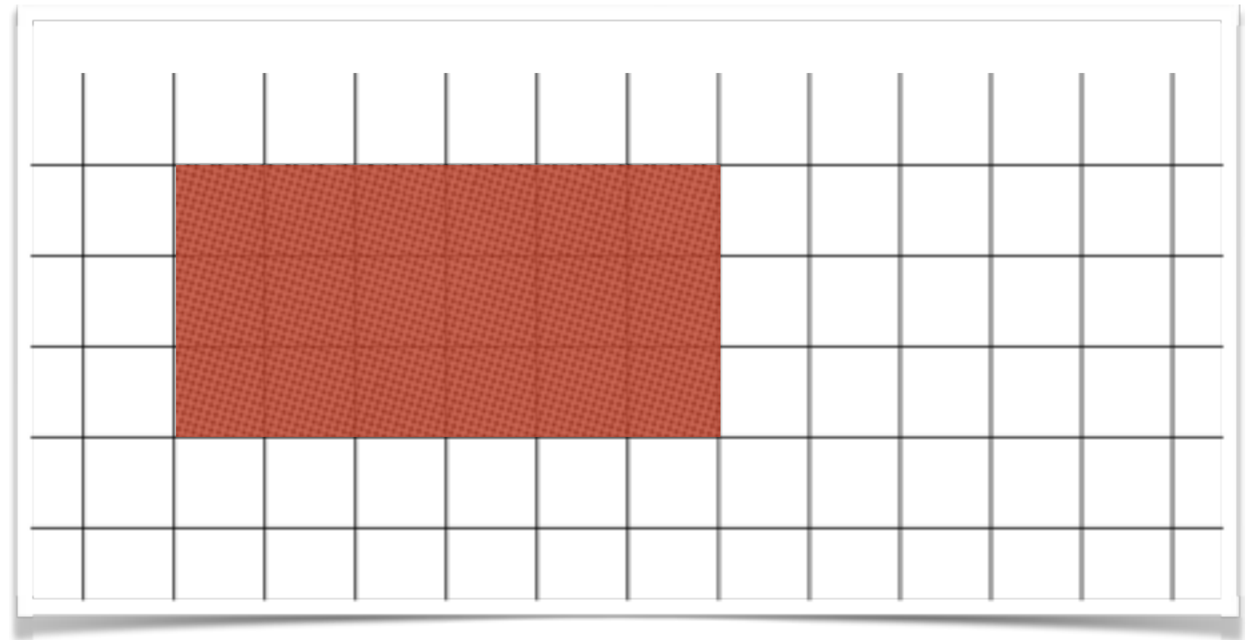


***Figure 2-1.** The inheritance hierarchy of geometry types. Instantiable types (those types from which an instance of data can be created in SQL Server 2012) are shown with a solid border.*

Generic Parsing Methods





```
SELECT
geometry::STGeomFromText('LINESTRING(300500 600150, 310200 602500)',
4326);
```

```
DECLARE @Square geography =
geometry::Parse('POLYGON((1 1, 6 1, 6 4, 1 4, 1 1))');
```

```
DECLARE @Square geography = 'POLYGON((1 1, 6 1, 6 4, 1 4, 1 1))';
```

SQL Server Demo

Use of .NET classes

- `SqlGeometry` and `SqlGeography` Classes
- `STGeomFromText` method requires `SqlChars` to be passed in, where as `Parse` can take a C# String.

```
SqlGeography Delhi = SqlGeography.STGeomFromText( new
SqlChars("POINT(77.25 28.5)"), 4326);
```

```
SqlGeography Delhi = SqlGeography.Parse("POINT(77.25 28.5)");
```

Retrieving WKT from SQL Server Types

- Recall that SQL Server stores all `geometry` and `geography` objects in a binary format
- Methods are provided to convert binary back into WKT format
  - `STAsText()`
    - OGC-compliant method, returns `SqlChars` (`nvarchar`). Only returns 2D coordinates, will ignore *z* or *m* values.
  - `AsTextZM()`
    - Same as `STAsText()`, but includes *z* and *m* values
  - `ToString()`
    - .NET base class `Object` defines this method for displaying object ivars, etc. Calls `AsTextZM()`, but will return a C# `string` type rather than `SQLChars` if in .NET code

Retrieving WKT from SQL Server Types

```
DECLARE @Point geometry =
   geometry::STPointFromText('POINT(14 9 7)', 0);
SELECT
   @Point.STAsText() AS STAsText,
   @Point.AsTextZM() AS AsTextZM,
   @Point.ToString() AS ToString;
```

STAsText	AsTextZM	ToString
POINT (14 9)	POINT (14 9 7)	POINT (14 9 7)

Creating Spatial Data from Well-Known Binary

- Another standard way of representing data, defined by OGC
- Contains header and stream of 8 byte values representing coordinates
- Unfortunately different from internal SQL Server binary format, still need to use methods for input conversion

Creating Spatial Data from Well-Known Binary

- Advantages
  - Faster than parsing WKT, as coordinates are 8 bytes in both internal format and WKB so parsing can be efficient
  - Floating point values do not lose precision with rounding to decimal format
- Disadvantages
  - Not human readable

Table 4-2. Methods to Instantiate Spatial Data from Well-Known Binary

Geometry	Static Method
Point	STPointFromWKB()
LineString	STLineFromWKB()
Polygon	STPolyFromWKB()
MultiPoint	STMPointFromWKB()
MultiLineString	STMLineFromWKB()
MultiPolygon	STMPolyFromWKB()
GeometryCollection	STGeomCollFromWKB()
Any supported geometry	STGeomFromWKB()

WKB Representation of a Point

```
0x000000000014001F5C28F5C28F6402524DD2F1A9FBE
```

Table 4-3. Elements Contained Within an Example WKB Geometry Representation

Value	Description
0x	Hexadecimal notation identifier
00	Byte order marker. 0×00 indicates little-endian byte order
00000001	This geometry is a Point, denoted as type 1
4001F5C28F5C28F6	x-coordinate (10.572)
402524DD2F1A9FBE	y-coordinate (2.245)

SQL WKB Methods

```
SELECT
geometry::STGeomFromWKB(0x00000000014001F5C28F5C28F6402524DD
2F1A9FBE, 2099);
```

Note that SRID is not serialized into WKB

```
DECLARE @g geometry =
   geometry::STPointFromText('POINT(14 9 7)', 0);
SELECT
   @g.STAsBinary();
```

↓

```
0x01010000000000000000002C40000000000000002240
```

SQL WKB Methods

- Note that like with `STAsText()` method, `STAsBinary()` drops the *z* and *m* fields. If *z* and *m* are desired, use the method `AsBinaryZM()`.

```
DECLARE @g geometry =
   geometry::STPointFromText('POINT(14 9 7)', 0);
SELECT
   @g.STAsBinary();
   @g.AsBinaryZM();
```

```
0x01010000000000000000000000002C400000000000002240
```

```
0x01E90300000000000000000000002C40000000000000224000000000000001C40
```

Creating Spatial Data from Geometry Markup Language

- Geometry Markup Language is a XML based format for representing spatial information.
- Be aware: coordinates are in latitude-longitude order rather than longitude-latitude order (what WKT uses). However, geometry is in x-y order just as WKT.
- No support for *z* or *m* coordinates, supports only 2D.
- No commas are necessary for lists of position pairs.

```
<Point xmlns="http://www.opengis.net/gml">
   <pos>47.6 -122.3</pos>
</Point>
```

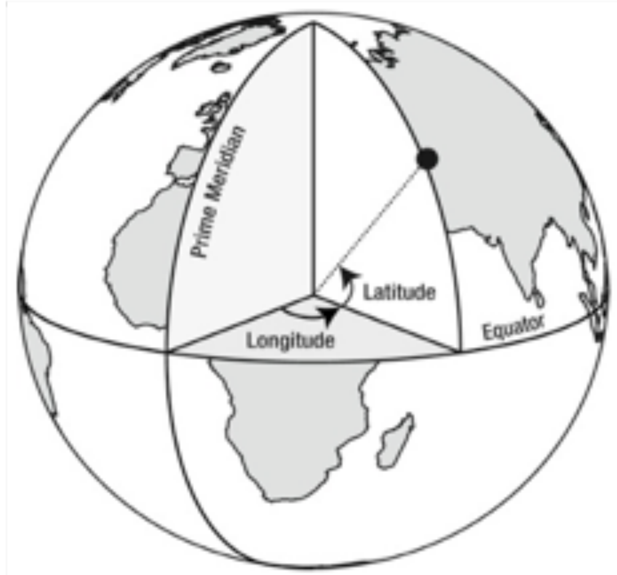


Figure 1-4. Describing positions using a geographic coordinate system.

Creating Spatial Data from Geometry Markup Language

- Typically used to transmit information over the internet (see also GeoJSON)
- Namespace required to be valid GML, otherwise just XML document
  xmlns="http://www.opengis.net/gml"

```
DECLARE @NoGMLNameSpace xml =
'<LineString>
    <posList>-6 4 3 -5</posList>
</LineString>';
SELECT geometry::GeomFromGml(@NoGMLNameSpace, 0);
```

↓

```
System.FormatException: 24129: The given XML instance is not valid because the
top- level tag is LineString. The top-level element of the input Geographic
Markup Language (GML) must contain a Point, LineString, Polygon, MultiPoint,
MultiGeometry, MultiCurve, MultiSurface, Arc, ArcString, CompositeCurve,
PolygonPatch or FullGlobe (geography Data Type only) object.
```

GML Advantages and Disadvantages

- Advantages
  - Easy to read like with WKT
  - Well structured XML format defines structure of geometry with sensible nesting
- Disadvantages
  - Very verbose, requires substantially more space to represent the same geometry
  - Also suffers from floating point rounding
  - SQL Server implements only a subset of full standard Importing some GML files may not be possible

```
<LineString xmlns=“http://www.opengis.net/gml">
   <posList>-6 4 3 -5 10 8</posList>
</LineString>
```

Inputting and Outputting GML

- Only one method for importing: `GeomFromGml()`
  - Must be the top-level `geometry` or `geography` type
- To obtain GML from SQL Server: `AsGml()6 -122.3</pos> </Point>'; SELECT`

```
DECLARE @gml xml =
'<Point xmlns="http://www.opengis.net/gml">
   <pos>47.6 -122.3</pos>
</Point>';

SELECT
   geography::GeomFromGml(@gml, 4269);
```

Inputting and Outputting GML

```
DECLARE @polygon geography =
   'POLYGON((-4 50, 2 50, 2 60, -4 60, -4 50))';
SELECT
   @polygon.AsGml();
```

↓

```
<Polygon xmlns="http://www.opengis.net/gml">
   <exterior>
      <LinearRing>
         <posList>50 -4 50 2 60 2 60 -4 50 -4</posList>
      </LinearRing>
   </exterior>
</Polygon>
```

Dynamically Generate WKT

- May have data not already in WKT, WKB, or GML
- Can use string manipulation to make WKT in SQL

```
CREATE TABLE GPSLog (
  Latitude float,
  Longitude float,
  LogTime datetime
);
INSERT INTO GPSLog VALUES
  (51.868, -1.198, '2011-06-02T13:47:00'),
  (51.857, -1.182, '2011-06-02T13:48:00'),
  (51.848, -1.167, '2011-06-02T13:49:00'),
  (51.841, -1.143, '2011-06-02T13:50:00'),
  (51.832, -1.124, '2011-06-02T13:51:00');
```

```
SELECT geography::STGeomFromText(
  'POINT(' + CAST(Longitude AS varchar(32)) + ' ' + CAST(Latitude AS varchar(32)) + ')',
  4326
  )
FROM GPSLog;
```

Dynamically Generate WKT

- Although simple internal constructors could be easier for simple cases. Book shows example of building up a LineString

```
SELECT geography::Point(Latitude, Longitude, 4326) FROM GPSLog;
```

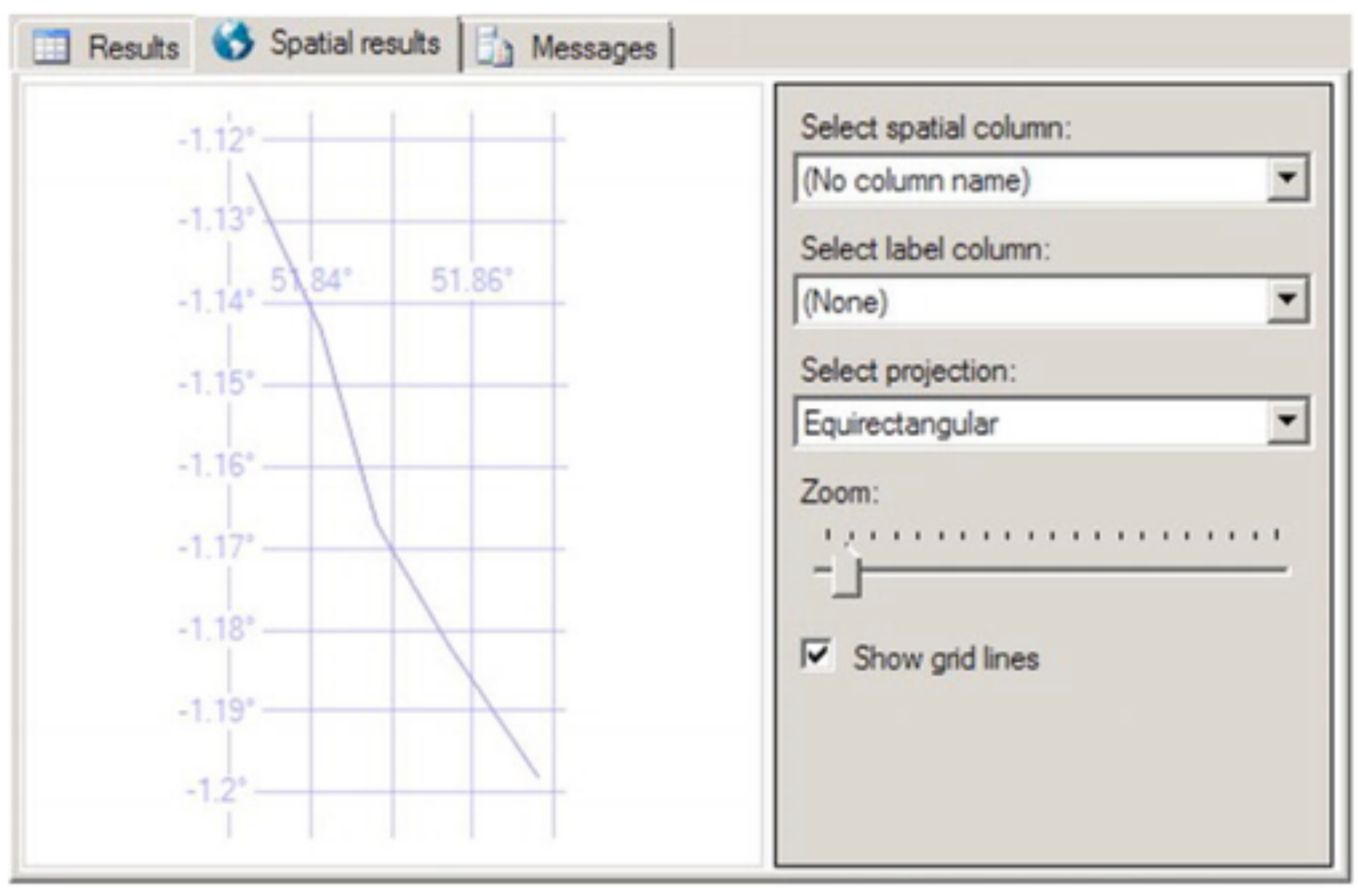


Figure 4-1. A LineString created from dynamically constructed WKT.

.NET Console Application Demo

Forming Well-Known Text