Cover Page

Contents

[Environment Setup 2](#_Toc323392639)

[Introduction to SQL Server 2](#_Toc323392640)

[Spatial Data Definition 3](#_Toc323392641)

[Spatial Data Manipulation 3](#_Toc323392642)

[Insertion Operation 3](#_Toc323392643)

[Selection Operation 3](#_Toc323392644)

[Map Viewer 3](#_Toc323392645)

[Solutions to Assignment 2 3](#_Toc323392646)

[References 4](#_Toc323392647)

# Environment Setup

The following are the software and hardware requirements for developing the solution of this assignment[1]:

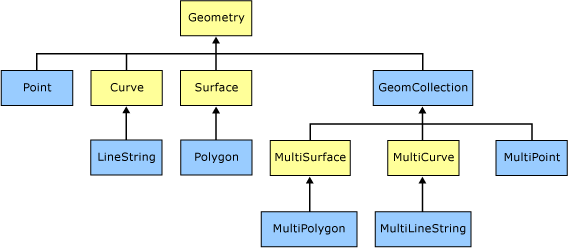
* DBMS: SQL Server 2012 Express Edition
* Operating System: Windows 7 Service Pack 1, Windows Server 2008 R2, Windows Server 2008 Server Pack 2, Windows Vista Service Pack 2
* 32 - bit systems: Computer with Intel or compatible 1GHz or faster processor (2 GHz or faster is recommended.)
* Minimum of 512 MB of RAM (2 GB or more is recommended.)
* 2.2 GB of available hard disk space

# Introduction to SQL Server 2012 Spatial

SQL Server 2012 is the Microsoft's latest cloud ready information platform[2]. Microsoft included support for spatial data since the release of the product SQL Server 2008. There are two types of spatial data[3].

* Geometry
* Geography

We use only "GEOMETRY" data type for develop solutions in our assignment.  SQL Server supports the geography data type, which stores ellipsoidal (round-earth) data, such as GPS latitude and longitude coordinates. But we do not require this in our assignment. The below figure[3] depicts the geometry hierarchy. The  instantiable types of **geometry**and **geography** are indicated in blue.



# Defining Tables

We create the following tables:

CREATE TABLE LandBlock( land\_id varchar(2) PRIMARY KEY, area geometry);

CREATE TABLE FuelLoadRegion ( fuel\_id varchar(2) PRIMARY KEY, area geometry, fuel\_load int CHECK (fuel\_load >= 0 and fuel\_load <= 100));

The first table defines the LandBlock. It has the following attributes:

* **land\_id** - Uniquely identifies a land block and contains a two character string. Hence the datatype is varchar(2). Also it is made PRIMARY KEY so that it uniquely identifies the record, ensures the value is not null and also satisfies the constraint of having primary key for creating indexes, which we will see below
* **area** - Includes the points to be stored for a unique land block. Hence the spatial data type is geometry.

The second table defines the FuelLoadRegion. It has the following attributes:

* **fuel\_id** - Uniquely identifies a fuel load block and just like land\_id, this also is named using two character string. Hence it is also declared as varchar(2) datatype. It is made PRIMARY KEY to help index on FuelLoadRegion table
* **area** - Includes the points to be stored for a unique fuel load block. It contains the spatial data with co-ordinate points, hence it is declared of type "geometry"
* **fuel\_load** - Indicates the amount of combustible fuel in integer units between 0 and 100. Hence is declared as integer and the constraint is specified to accept values only between 0 and 100.

# Inserting Data into Tables

The data is inserted into the two tables by executing the below SQL statements:

### Inserting data into "LandBlock"

INSERT INTO LandBlock(land\_id, area) VALUES

('L1',geometry::STGeomFromText('POLYGON ((0 22, 0 29, 15 22, 0 22))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES ('L2',geometry::STGeomFromText('CURVEPOLYGON(CIRCULARSTRING(3 17, 6 20, 9 17, 6 14, 3 17))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L3',geometry::STGeomFromText('POLYGON ((17 17, 20 17, 22 14, 22 19, 25 11, 17 12, 17 17))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L4',geometry::STGeomFromText('POLYGON ((15 27,25 27, 25 22, 15 22, 15 27))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L5',geometry::STGeomFromText('POLYGON ((7 14, 17 14, 17 9, 7 9, 7 14))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L6',geometry::STGeomFromText('POLYGON ((1 7, 1 12, 7 12, 7 7, 1 7))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L7',geometry::STGeomFromText('POLYGON ((8 7, 14 7, 12.5 0, 8 7))',0));

### Inserting data into "FuelLoadRegion"

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F1', geometry::STGeomFromText('POLYGON((3 29, 13 29, 13 23, 9 23,9 26, 3 26, 3 29))',0), 70);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F2', geometry::STGeomFromText('POLYGON((3 25, 5 21, 5 18, 2 18, 2 21, 1 21, 3 25))',0), 20);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F3', geometry::STGeomFromText('POLYGON((15 10, 20 10, 20 8, 24 8, 24 6, 29 6, 29 5, 20 5, 20 4, 18 4, 18 8, 15 8, 15 10))',0), 35);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F4', geometry::STGeomFromText('CURVEPOLYGON(COMPOUNDCURVE(CIRCULARSTRING(17 23,17.5 22.04, 18.5 21.47),CIRCULARSTRING(18.5 21.47,21 21.04, 22 21),CIRCULARSTRING(22 21,23 21.04, 25.5 21.47),CIRCULARSTRING(25.5 21.47,26.5 22.04, 27 23),CIRCULARSTRING(27 23,26.5 23.96, 25.5 24.53),CIRCULARSTRING(25.5 24.53,23 24.96, 22 25),CIRCULARSTRING(22 25,21 24.96, 18.5 24.53),CIRCULARSTRING(18.5 24.53,17.5 23.96, 17 23)))',0), 40);

The insert operation is used to insert data into the tables. The method " geometry::STGeomFromText" returns  a geometry instance from an Open Geospatial Consortium (OGC) Well-Known Text (WKT) representation.[4]

**Syntax:**

STGeomFromText ( 'geometry\_tagged\_text' , SRID )

**Arguments:**

*geometry\_tagged\_text*

Is the WKT representation of the **geometry** instance you wish to return. *geometry\_tagged\_text* is an **nvarchar(max)** expression.

*SRID*

Is an **int** expression representing the spatial reference ID (SRID) of the **geometry** instance you wish to return.

The following 'geometry\_tagged\_text' can be defined:

* Point
* LineString
* CircularString
* CompoundCurve
* Polygon
* CurvePolygon

Collection types include:

* MultiPoint
* MultiLineString
* MultiPolygon
* GeometryCollection

We have used "**Polygon**" to create all the shapes but ellipse and circle. Ellipse and circle is created using '**CURVEPOLYGON**' and ' **CIRCULARSTRING**' as there is no direct support by SQL Server 2012 to implement a ellipse or circle.

**Polygon** instances are instances that can be stored in a geometry or geography variable without throwing an exception. The following are accepted Polygon instances[5]:

* An Empty Polygon instance
* A Polygon instance that has an acceptable exterior ring and zero or more acceptable interior rings

The following criteria are needed for a ring to be acceptable.

* The LineString instance must be accepted.
* The LineString instance must have at least four points.
* The starting and ending points of the LineString instance must be the same.

'POLYGON((1 1, 3 3, 3 1, 1 1))';

**Circular strings** are the basic curve subtype, corresponding to a LineString for linear data. Three points are used to define a segment with the start (first point) and end (third point) points and another point (second point) anywhere along the circular arc. Circular strings can be linked together where the last point of the previous curve becomes the first point of the next curve. Valid circular strings will always

CIRCULARSTRING(0 -23.43778, 0 0, 0 23.43778);

CURVEPOLYGON  is a topologically closed surface defined by an exterior bounding ring and zero or more interior rings These polygons are similar to polygons, having at least one ring and zero or more holes (inner rings). Curve polygons are composed of linear strings, circular strings, and/or compound curves. Within a given ring, the first point as defined in a curve polygon component must be identical to the last point in a curve polygon component, just like standard polygon rings[6].

STGeomFromText('CURVEPOLYGON(COMPOUNDCURVE(CIRCULARSTRING(17 23,17.5 22.04, 18.5 21.47),CIRCULARSTRING(18.5 21.47,21 21.04, 22 21),CIRCULARSTRING(22 21,23 21.04, 25.5 21.47),CIRCULARSTRING(25.5 21.47,26.5 22.04, 27 23),CIRCULARSTRING(27 23,26.5 23.96, 25.5 24.53),CIRCULARSTRING(25.5 24.53,23 24.96, 22 25),CIRCULARSTRING(22 25,21 24.96, 18.5 24.53),CIRCULARSTRING(18.5 24.53,17.5 23.96, 17 23)))'

# Index Creation

# Query Execution

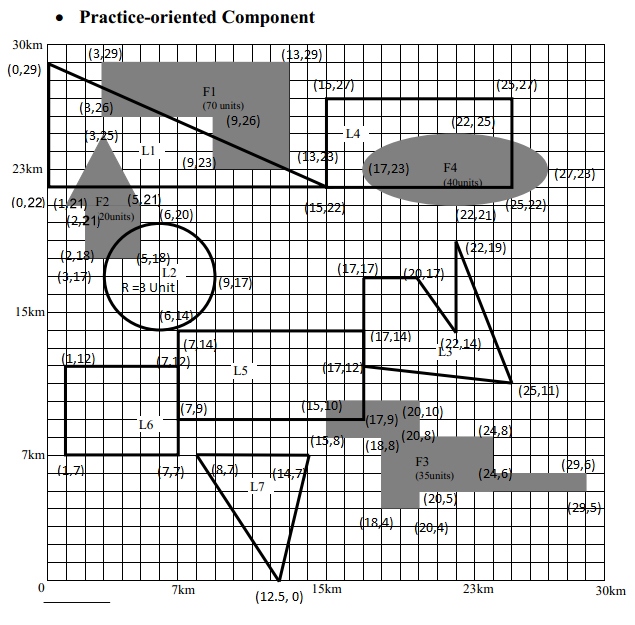
Spatial Operations like Union, Intersection, Contains

# Query Execution Plan

# Map Viewer - Andreas

# Summarized Solutions to Assignment 2

The below diagram shows the land blocks and the regions of fuel load **MARKED WITH POINTS**.



## You are required to create two tables. One table is called "LandBlock" to store land blocks and the other table is called "FuelLoadRegion" to store fuel load regions. In these tables, one attribute must be GEOMETRY data type to describe spatial objects.

### Creating Tables:

CREATE TABLE LandBlock( land\_id varchar(2) PRIMARY KEY, area geometry);

CREATE TABLE FuelLoadRegion ( fuel\_id varchar(2) PRIMARY KEY, area geometry, fuel\_load int CHECK (fuel\_load >= 0 and fuel\_load <= 100));

### Inserting data into "LandBlock"

INSERT INTO LandBlock(land\_id, area) VALUES

('L1',geometry::STGeomFromText('POLYGON ((0 22, 0 29, 15 22, 0 22))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES ('L2',geometry::STGeomFromText('CURVEPOLYGON(CIRCULARSTRING(3 17, 6 20, 9 17, 6 14, 3 17))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L3',geometry::STGeomFromText('POLYGON ((17 17, 20 17, 22 14, 22 19, 25 11, 17 12, 17 17))', 0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L4',geometry::STGeomFromText('POLYGON ((15 27,25 27, 25 22, 15 22, 15 27))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L5',geometry::STGeomFromText('POLYGON ((7 14, 17 14, 17 9, 7 9, 7 14))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L6',geometry::STGeomFromText('POLYGON ((1 7, 1 12, 7 12, 7 7, 1 7))',0));

INSERT INTO LandBlock(land\_id, area) VALUES

('L7',geometry::STGeomFromText('POLYGON ((8 7, 14 7, 12.5 0, 8 7))',0));

### Inserting data into "FuelLoadRegion"

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F1', geometry::STGeomFromText('POLYGON((3 29, 13 29, 13 23, 9 23,9 26, 3 26, 3 29))',0), 70);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F2', geometry::STGeomFromText('POLYGON((3 25, 5 21, 5 18, 2 18, 2 21, 1 21, 3 25))',0), 20);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F3', geometry::STGeomFromText('POLYGON((15 10, 20 10, 20 8, 24 8, 24 6, 29 6, 29 5, 20 5, 20 4, 18 4, 18 8, 15 8, 15 10))',0), 35);

INSERT INTO FuelLoadRegion(fuel\_id, area, fuel\_load) VALUES

('F4', geometry::STGeomFromText('CURVEPOLYGON(COMPOUNDCURVE(CIRCULARSTRING(17 23,17.5 22.04, 18.5 21.47),CIRCULARSTRING(18.5 21.47,21 21.04, 22 21),CIRCULARSTRING(22 21,23 21.04, 25.5 21.47),CIRCULARSTRING(25.5 21.47,26.5 22.04, 27 23),CIRCULARSTRING(27 23,26.5 23.96, 25.5 24.53),CIRCULARSTRING(25.5 24.53,23 24.96, 22 25),CIRCULARSTRING(22 25,21 24.96, 18.5 24.53),CIRCULARSTRING(18.5 24.53,17.5 23.96, 17 23)))',0), 40);

## Create spatial index on the attribute GEOMETRY data type to describe the spatial object

CREATE SPATIAL INDEX LandBlock\_Index

ON LandBlock(area)

WITH ( BOUNDING\_BOX = ( 0, 0, 30, 30));

CREATE SPATIAL INDEX FuelLandBlock\_Index

ON FuelLoadRegion(area)

WITH ( BOUNDING\_BOX = ( 0, 0, 30, 30));

## Answer the following on attribute GEOMETERY in each table using SPATIAL queries:

### a. Retrieve all regions in LandBlock and FuelLoadRegion

SELECT area FROM FuelLoadRegion

UNION ALL

SELECT area FROM LandBlock

### b. Find the area size for each region in LandBlock and FuelLoadRegion

SELECT land\_id as id,ROUND(area.STArea(),2) AS block\_size FROM LandBlock

UNION ALL

SELECT fuel\_id as id, ROUND(area.STArea(),2) AS block\_size FROM FuelLoadRegion;

### c. Find all regions of fuel load higher than 30 units

SELECT F.fuel\_id, F.area

FROM FuelLoadRegion As F

WHERE F.fuel\_load > 30;

### d. Find all land blocks which have some region containing fuel load of 30 units or higher

SELECT L.land\_id, L.area

FROM LandBlock L, FuelLoadRegion F

WHERE F.fuel\_load>=30 AND (L.area.STIntersects(F.area)=1)

### e. Find all land blocks within 11 km of the area containing fuel load over 70 units.

SELECT L.land\_id, L.area

FROM LandBlock L, FuelLoadRegion F

WHERE L.area.STDistance(f.area)<=11 and F.fuel\_load>=70

# References

1. System Requirements - <http://msdn.microsoft.com/library/ms143506(v=SQL.110).aspx>
2. Introducing Microsoft SQL Server 2012 - Ross Mistry and Stacia Misner
3. Data types - <http://msdn.microsoft.com/en-us/library/bb964711(v=sql.100).aspx>
4. <http://msdn.microsoft.com/en-us/library/bb933823.aspx>
5. <http://msdn.microsoft.com/en-us/library/bb895267.aspx>
6. New Spatial Features in SQL Server Code-Named “Denali” [Ed Katibah (Microsoft) and Milan Stojic (Microsoft)] - SQL Server Technical Article