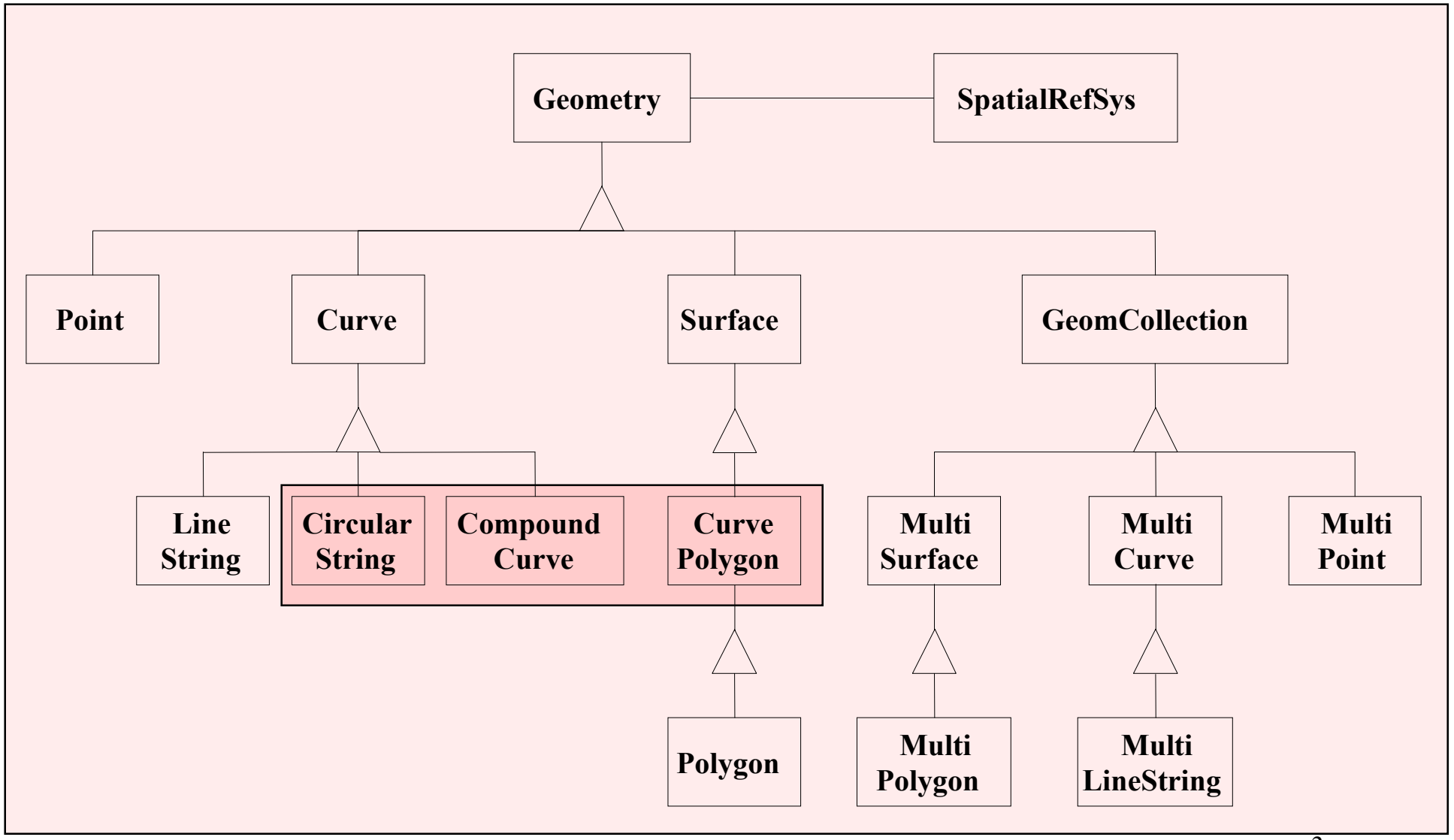
Information technology - Database Languages - SQL Multimedia and Application Packages - Part 3: Spatial

Dr. Paul Scarponcini
Bentley Transportation
paul.scarponcini@bentley.com
March 26, 2000

ISO SQL Follow-On Standard

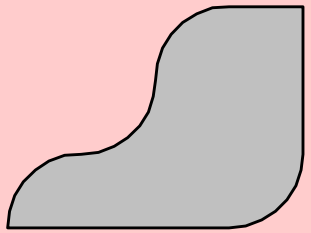
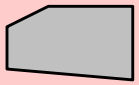
- SQL/MM (Multi Media and Application Packages)
  - Framework
  - Full Text
  - ➔ *Spatial*
  - (General Purpose Facilities)
  - Still Image
  - Data Mining

SQL/MM Geometry Model: UDTs



City Example

CITY TABLE

NAME varchar(30)	POPULATION integer	CITY PARKS varchar(30) array[10]	LOCATION st_geometry
My Little Town	1042	['Lincoln Park']	
The Big City	1450000	['City Park', 'Grant Park', 'Castle Park', 'Forest Park']	
A Big Town	4900	[ ]	

City Example - Table Create

```
CREATE TABLE CITY (
   NAME              VARCHAR(30),
   POPULATION        INTEGER,
   CITY_PARKS        VARCHAR(30) ARRAY[10],
   LOCATION          ST_GEOMETRY )
```

City Example - Insert

```
BEGIN
  DECLARE  city_location  ST_POINT;
  SET city_location =
              NEW  ST_POINT(100.0, 200.0);
  INSERT INTO CITY VALUES (
      'My Little Town',
      1042,
      ARRAY ['Lincoln Park'],
      city_location );
END
```

City Example - Insert

```
INSERT INTO CITY VALUES (
    'My Little Town',
    1042,
    ARRAY ['Lincoln Park'],
    NEW  ST_POINT(100.0, 200.0));
```

City Example - Query

```
SELECT location.area
FROM CITY
WHERE NAME = 'The Big City'
```

Exercise - 1

Exercise 1

PERSON TABLE

NAME	ADDRESS	PHONE
Salmonn	5 Washington St.	555-1234
Fig	1201 Maple Street	
Freezon	27 Olive Drive	555-5976

Exercise 1

- Entities
  1. **What entity is contained within the table?**

  2. **How many of these entities exist?**

- Attributes
  3. **What attributes are contained in the table?**

  4. **What is the type of the third column?**

Exercise 1

- **Entities**
  1. **What entity is contained within the table?**
     Person
  2. **How many of these entities exist?**
     3 (in the table)
- **Attributes**
  3. **What attributes are contained in the table?**
     name, address, phone
  4. **What is the type of the third column?**
     varying of character (VARCHAR)

Exercise 1

- Attribute values

  5. What is the value of PHONE for Salmonn?

  6. For Fig?

- Query

  7. In normal English wording, write a query which returns all of the names in the table.

  8. Using SQL, write the query which returns all of the names in the table.

Exercise 1

- **Attribute values**
  - **5. What is the value of PHONE for Salmonn?**

    555-1234
  - **6. For Fig?**

    NULL (not known)
- **Query**
  - **7. In normal English wording, write a query which returns all of the names in the table.**

    Select names from the PERSON table
  - **8. Using SQL, write the query which returns all of the names in the table.**

    SELECT NAME FROM PERSON

Exercise 1

- Query

9. Write a query to return all of the information about Freezon.

10. Write the query to return all of the people on Maple Street.

Exercise 1

- **Query**

9. Write a query to return all of the information about Freezon.

```
SELECT NAME, ADDRESS, PHONE
   FROM PERSON
   WHERE NAME = 'Freezon'
```

10. Write the query to return all of the people on Maple Street.

```
SELECT NAME
   FROM PERSON
   WHERE ADDRESS LIKE '%Maple Street%'
```

Exercise 1

+ Bonus

11. How far is it from Salmonn's place to Fig's place?

12. Write the query to return all of Fig's next door neighbors.

13. Write a query to return who has the largest property.

Exercise 1

- **Bonus**
  - **11. How far is it from Salmonn's place to Fig's place?**
    ???
  - **12. Write the query to return all of Fig's next door neighbors.**
    ???
  - **13. Write a query to return who has the largest property.**
    ???

International Standard

- Completed in 1999 after 5 years of effort
- Contributions by all major GIS vendors
- Major contributing National Bodies include US, Japan, Canada, UK, Germany

- Based upon SQL'99 (9075) with parallel development schedules
- ISO/IEC 13249-3 (IS) [344 pages]
- Available at: http://global.ihs.com

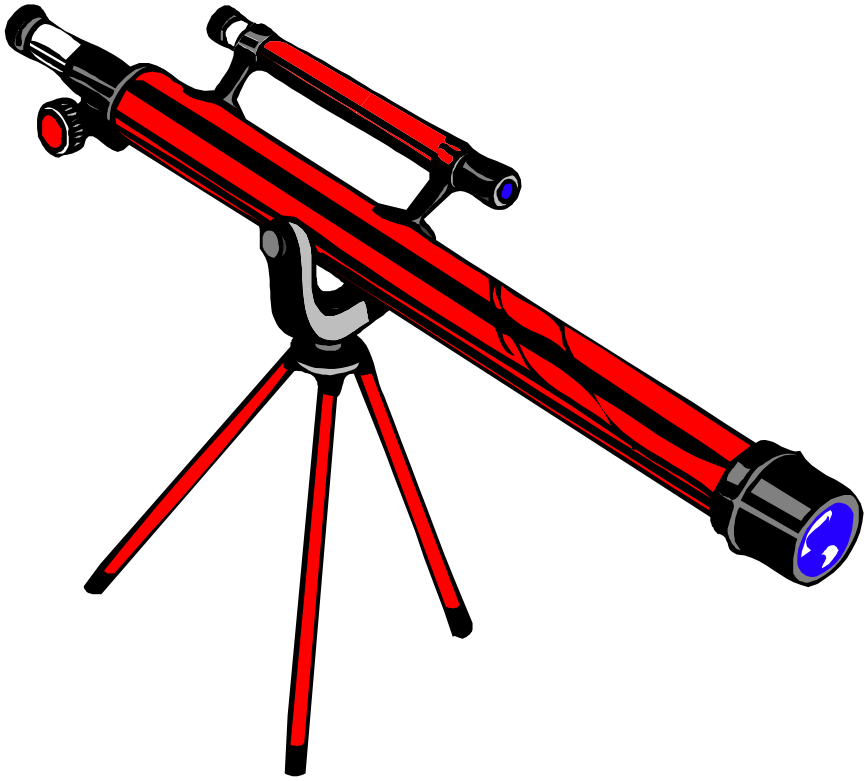
Clauses

1. Scope
2. Normative references
3. Definitions, notations, and conventions
4. Concepts
5. Geometry types
6. Point types
7. Curve types

Clauses (continued)

8. Surface types
9. Geometry collection types
10. Spatial reference system types
11. Support routines
12. Conformance
13. Status codes

Clause 1: Scope



Scope

- generic to spatial data handling
- “ … addresses the need to store, manage, and retrieve information based on aspects of spatial data such as geometry, location, and topology.”
- does not redefine the database language SQL

Scope

- Implementations may exist in environments that also support geographic information, decision support, data mining, and data warehousing
- Application areas include automated mapping, desktop mapping, facilities management, geoengineering, graphics, multimedia, and resource management

Clause 3: Definitions, notations, and conventions

Definitions

OOP	RDBMS	O-R
class	entity or table	type
object (instance)	entity instance or row	value
attribute	attribute or column	attribute
method	procedures	method

Definitions

- **(geometric) dimension** - largest number *n* such that each direct position in a geometric set is associated with a subset that has the direct position in its interior and is similar (isomorphic) to R^n, Euclidean *n*-space

  0 = point

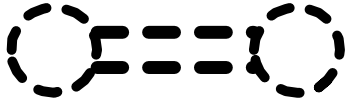
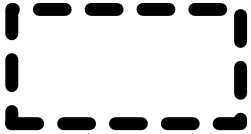
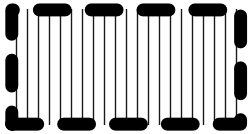
  1 = curve

  2 = surface

Definitions

- **point** - 0-dimensional geometric primitive, representing a position, but not having extent
- **curve** - bounded, connected 1-dimensional geometric primitive, representing the continuous image of a line, and therefore fully realizable as a 1-parameter set of points
- **surface** - bounded, connected 2-dimensional geometric primitive, representing the continuous image of region of a plane, and therefore fully realizable locally as a 2-parameter set of points

Presentation graphic conventions

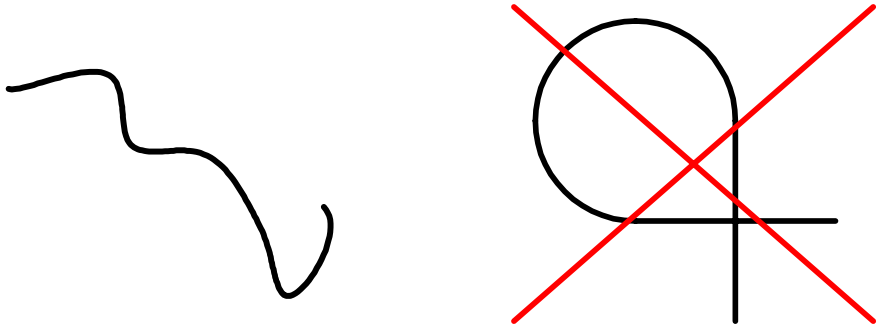
Point:		
Curve:		
Closed Curve:		
Surface:		

Definitions

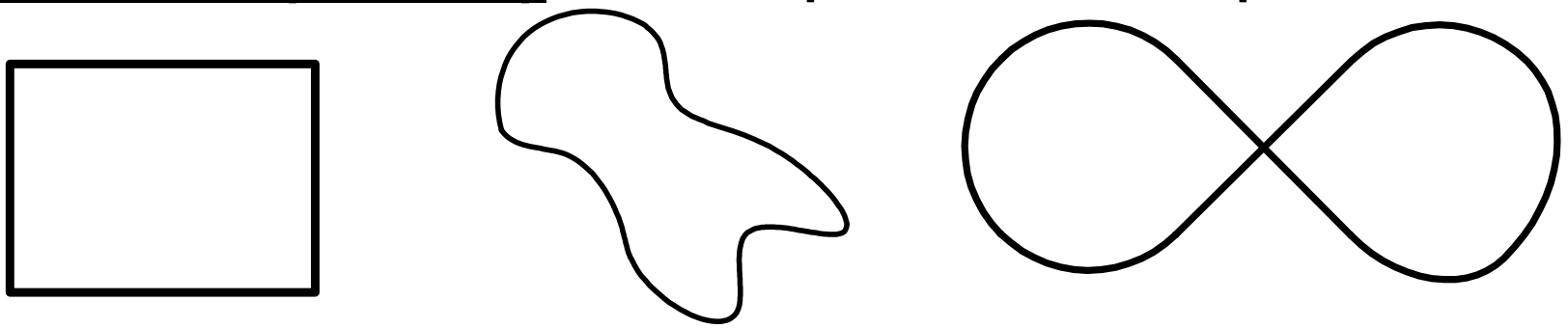
- **coordinate** - one of a set of *N* numbers designating the position of a point in N-dimensional space
- **coordinate system** - set of (mathematical) rules for specifying how coordinates are to be assigned to points
- **coordinate reference system** - coordinate system which is related to the earth by a datum
- **coordinate dimension** - number of measurements or ordinates needed to describe a position in a coordinate system

Definitions

- **simple (geometry)** - no anomalous points, such as self intersection or self tangency



- **closed (curve)** - start point = end point



- **ring** - curve value that is closed and simple

Definitions

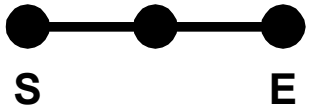
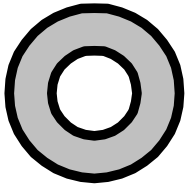
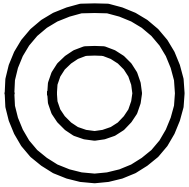
- **point set** - representation of a geometry as a (in)finite set of points

  mathematical set intersection, union, and difference apply
- **closure** - topological function to cause an open point-set to include its boundary making the point-set topologically closed
- **topologically closed** - characteristic of a geometry type that every value includes its own boundary

Definitions ("well defined in general topology")

- **boundary** - "set of geometry values of the next lower dimension":
  - of a point: empty set
  - of a curve: empty set if closed; else set containing start and end points
  - of a surface: set of curves that delineates the edge, including interior and exterior rings
- **interior** - set of all points that comprise a geometric object but which are not in its boundary
- **exterior** - points not in the boundary or interior

Boundary and Interior

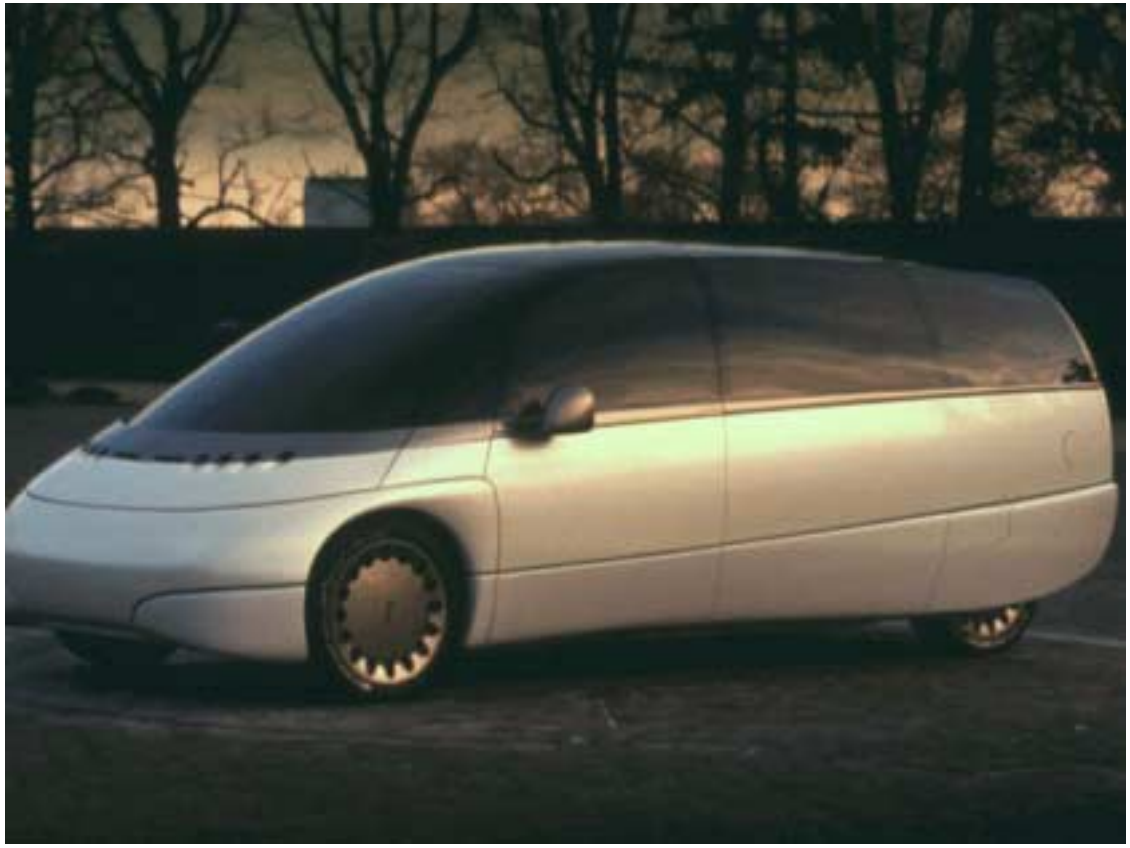
		Boundary	Interior
Point		{ }	
Curve		{ $\bullet_S$, $\bullet_E$ }	
Curve		{ $\bullet_S$, $\bullet_E$ }	
Closed Curve		{ }	
Surface			
Surface			

Notations (symbols)

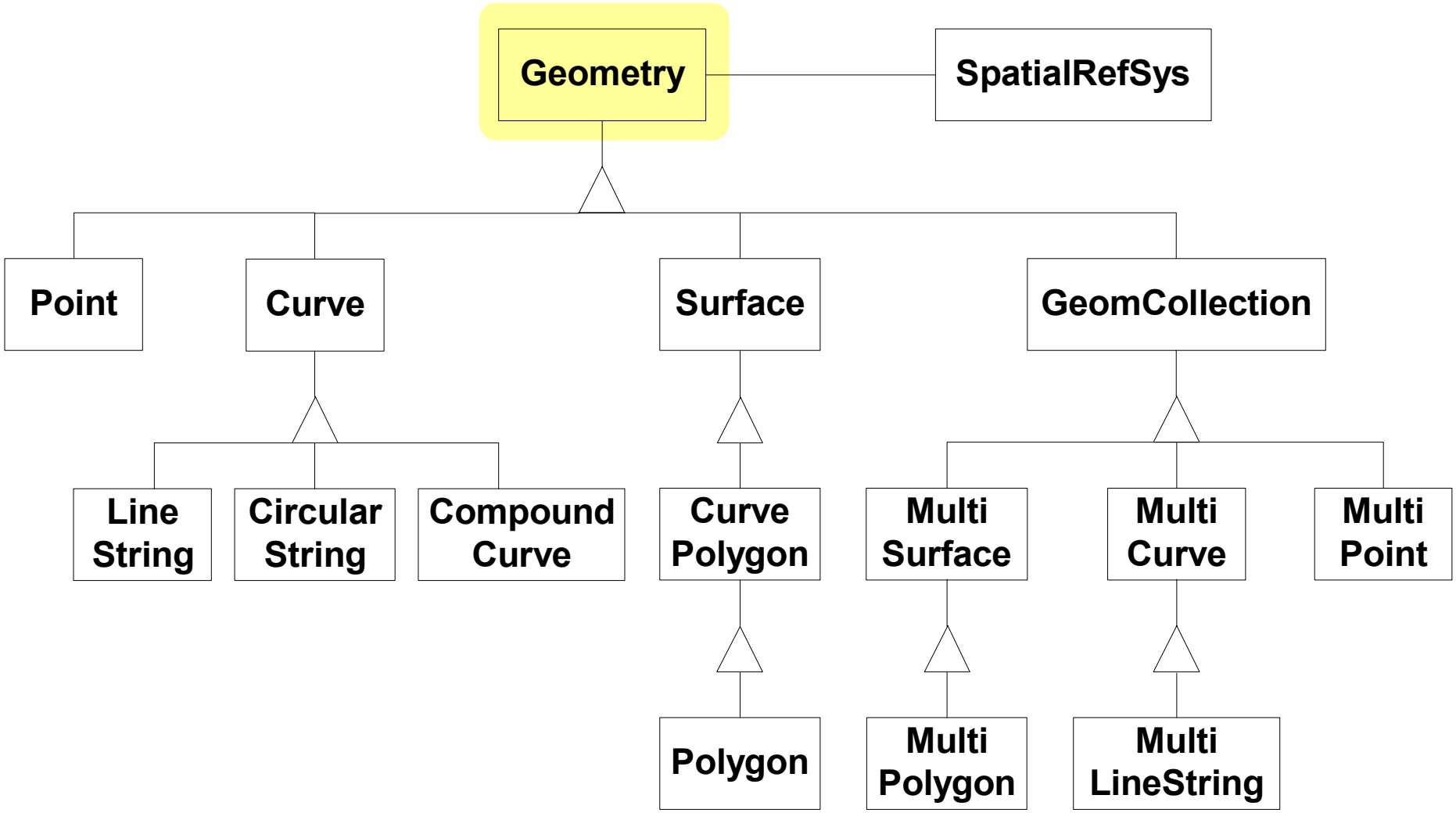
$\varnothing$	empty set
$\cap$	Intersection
$\cup$	Union
—	Difference
$\in$	is a member of
$\notin$	is not a member of
$\subset$	is a proper subset of
$\subseteq$	is a subset of

$\Leftrightarrow$	if and only if
$\Rightarrow$	Implies
$\forall$	for all
$\{ x \mid ... \}$	set of all x such that ...
$\wedge$	And
$\vee$	Or
$\neg$	Not

Clause 4: Concepts



Geometry Type Hierarchy



Concepts
ST_Geometry

- non-instantiable root type
- subtypes are 0-, 1-, and 2-dimensional geometry types in 2-D coordinate space
- all instantiable type values are topologically closed
- all locations in a geometry value are in the same spatial reference system
- all calculations and returned values are in the SRS of the first parameter

Concepts
ST_Geometry Methods

- dimension
- coordinate dimension
- type
- SRID
- transform
- is empty
- is simple
- boundary
- envelope
- convex hull
- buffer
- intersection
- union
- difference
- symmetric difference
- distance
- WKT to SQL
- as text
- WKB to SQL
- as binary

Concepts
Spatial Relationships

- Dimensionally extended 9 Intersection Model (DE-9IM)
- compares boundaries, interiors, and exteriors of two geometry values
- spatial relationships have pre-defined “patterns”
- ST_Relates tests spatial relationships by testing BIE intersections

Concepts
DE-9IM Intersection Matrix

	Interior	Boundary	Exterior
Interior	**(I(*a*) ∩ I(*b*)). ST.Dimension**	**(I(*a*) ∩ B(*b*)). ST.Dimension**	**(I(*a*) ∩ E(*b*)). ST.Dimension**
Boundary	**(B(*a*) ∩ I(*b*)). ST.Dimension**	**(B(*a*) ∩ B(*b*)). ST.Dimension**	**(B(*a*) ∩ E(*b*)). ST.Dimension**
Exterior	**(E(*a*) ∩ I(*b*)). ST.Dimension**	**(E(*a*) ∩ B(*b*)). ST.Dimension**	**(E(*a*) ∩ E(*b*)). ST.Dimension**

DE-9IM Pattern Matrix

- consists of a set of 9 pattern-values (*p*), one for each cell in the matrix
- meanings for any cell where *x* is the intersection set for the cell are:
  - if *p* = T, then *x*.ST_Dimension $\in$ { 0, 1, 2 }, i.e. $x \neq \varnothing$
  - if *p* = F, then *x*.ST_Dimension = -1, i.e. $x = \varnothing$
  - if *p* = 0, then *x*.ST_Dimension = 0 (zero)
  - if *p* = 1, then *x*.ST_Dimension = 1 (one)
  - if *p* = 2, then *x*.ST_Dimension = 2
  - if *p* = *, then *x*.ST_Dimension $\in$ { -1, 0, 1, 2 }, i.e. any value

DE-9IM Pattern Matrix

- pattern matrix can be represented as a character string of 9 characters, representing DE-9IM in row major order
- for example, ‘ **012TF**** ’ means

	I	**B**	**E**
I	0	1	2
B	T	F	*
E	*	*	*

(I(*a*) ∩ I(*b*)).ST_Dimension = 0

(I(*a*) ∩ B(*b*)).ST_Dimension = 1

(I(*a*) ∩ E(*b*)).ST_Dimension = 2

(B(*a*) ∩ I(*b*)).ST_Dimension = 0 or 1 or 2

(B(*a*) ∩B(*b*)).ST_Dimension = -1, i.e. $x = \varnothing$

all others can equal any value {-1,0,1,2,3}

Concepts
Common Spatial Relationships

- Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps
- mutually exclusive
- complete covering
- apply to pairs of geometries
- apply across dimensions
- expressible in terms of DE-9IM patterns

ST_Disjoint

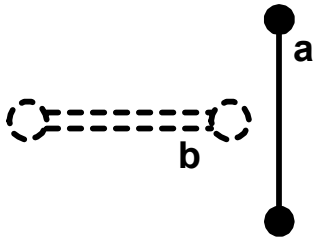
Given two geometries *a* and *b*, ST_Disjoint is defined as:

a.ST_Disjoint(b) $\Leftrightarrow$ $a \cap b = \varnothing$

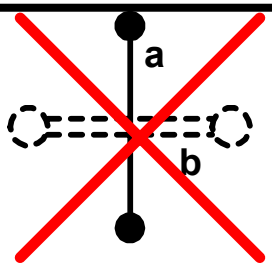
Expressed in terms of the DE-9IM:

a.ST_Disjoint(b) $\Leftrightarrow$
(Interior(a) $\cap$ Interior(b) = $\varnothing$) $\wedge$
(Interior(a) $\cap$ Boundary(b) = $\varnothing$) $\wedge$
(Boundary(a) $\cap$ Interior(b) = $\varnothing$) $\wedge$
(Boundary(a) $\cap$ Boundary(b) = $\varnothing$)
$\Leftrightarrow$ a.ST_Relate(b, 'FF*FF****')

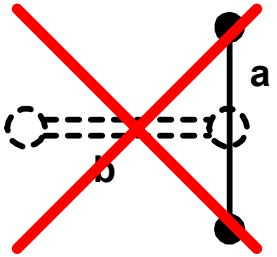
ST_Disjoint $a \cap b = \varnothing$



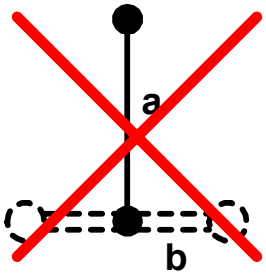
$(\text{Interior}(a) \cap \text{Interior}(b) = \varnothing) \wedge$



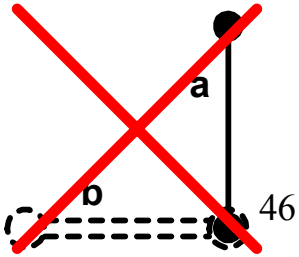
$(\text{Interior}(a) \cap \text{Boundary}(b) = \varnothing) \wedge$



$(\text{Boundary}(a) \cap \text{Interior}(b) = \varnothing) \wedge$

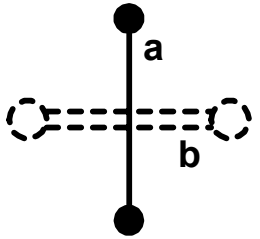


$(\text{Boundary}(a) \cap \text{Boundary}(b) = \varnothing)$

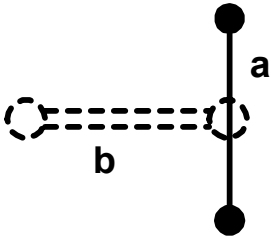


ST_Intersects $a \cap b \neq \emptyset$

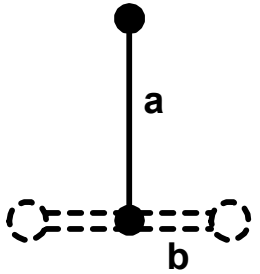
$(\text{Interior}(a) \cap \text{Interior}(b) \neq \emptyset) \vee$



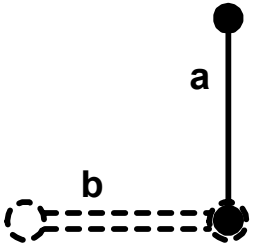
$(\text{Interior}(a) \cap \text{Boundary}(b) \neq \emptyset) \vee$



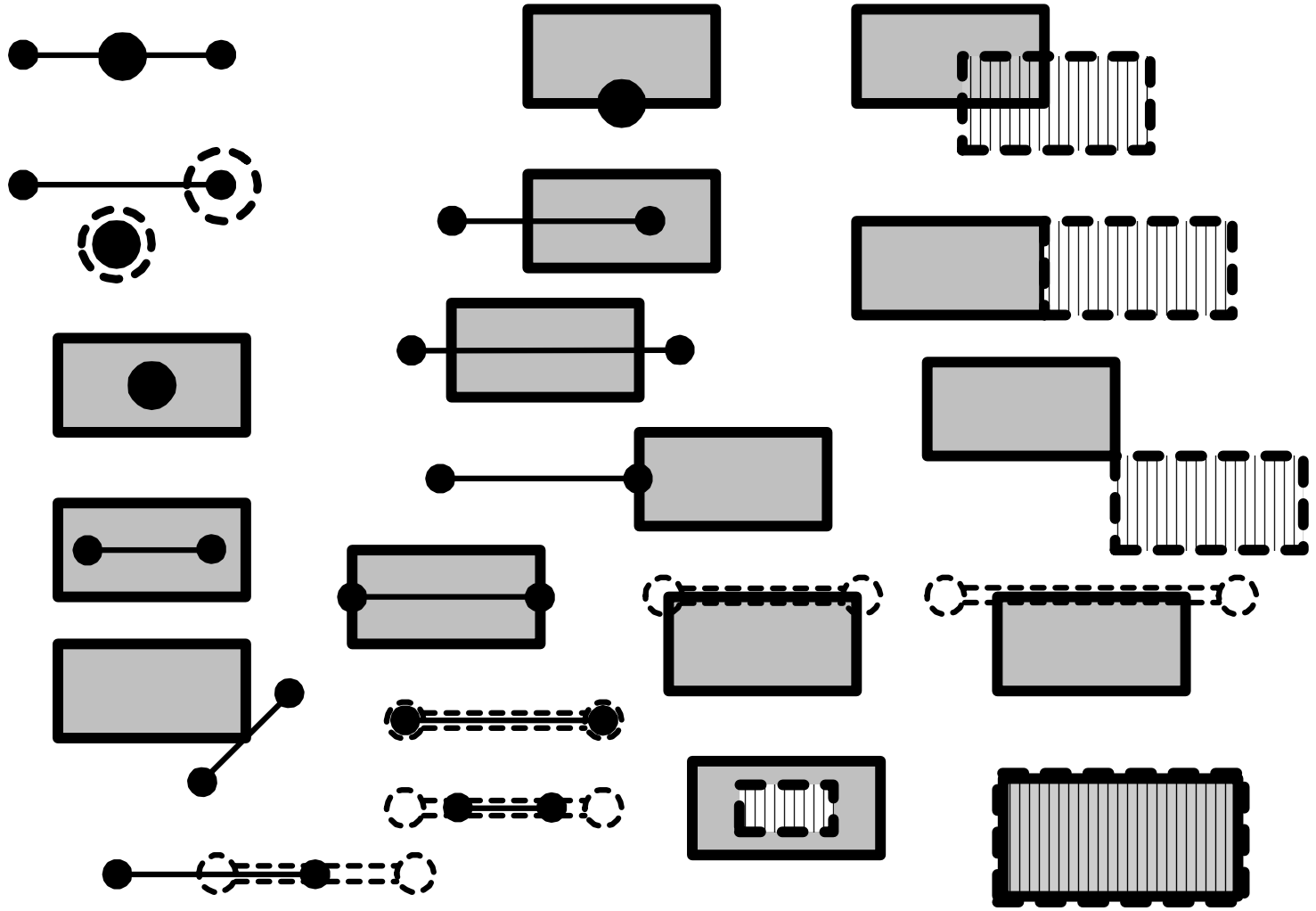
$(\text{Boundary}(a) \cap \text{Interior}(b) \neq \emptyset) \vee$



$(\text{Boundary}(a) \cap \text{Boundary}(b) \neq \emptyset)$



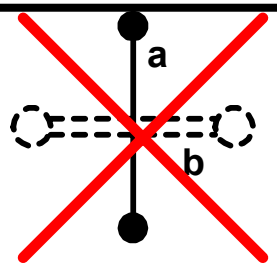
Other ST_Intersects Examples



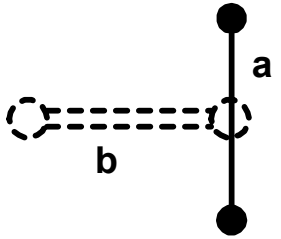
ST_Touches

$$(a \cap b) \neq \varnothing \wedge (\text{Interior}(a) \cap \text{Interior}(b) = \varnothing)$$

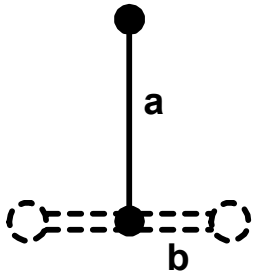
$$(\text{Interior}(a) \cap \text{Interior}(b) = \varnothing) \wedge$$



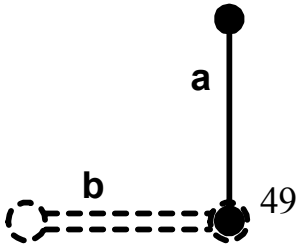
$$((\text{Interior}(a) \cap \text{Boundary}(b) \neq \varnothing) \vee$$



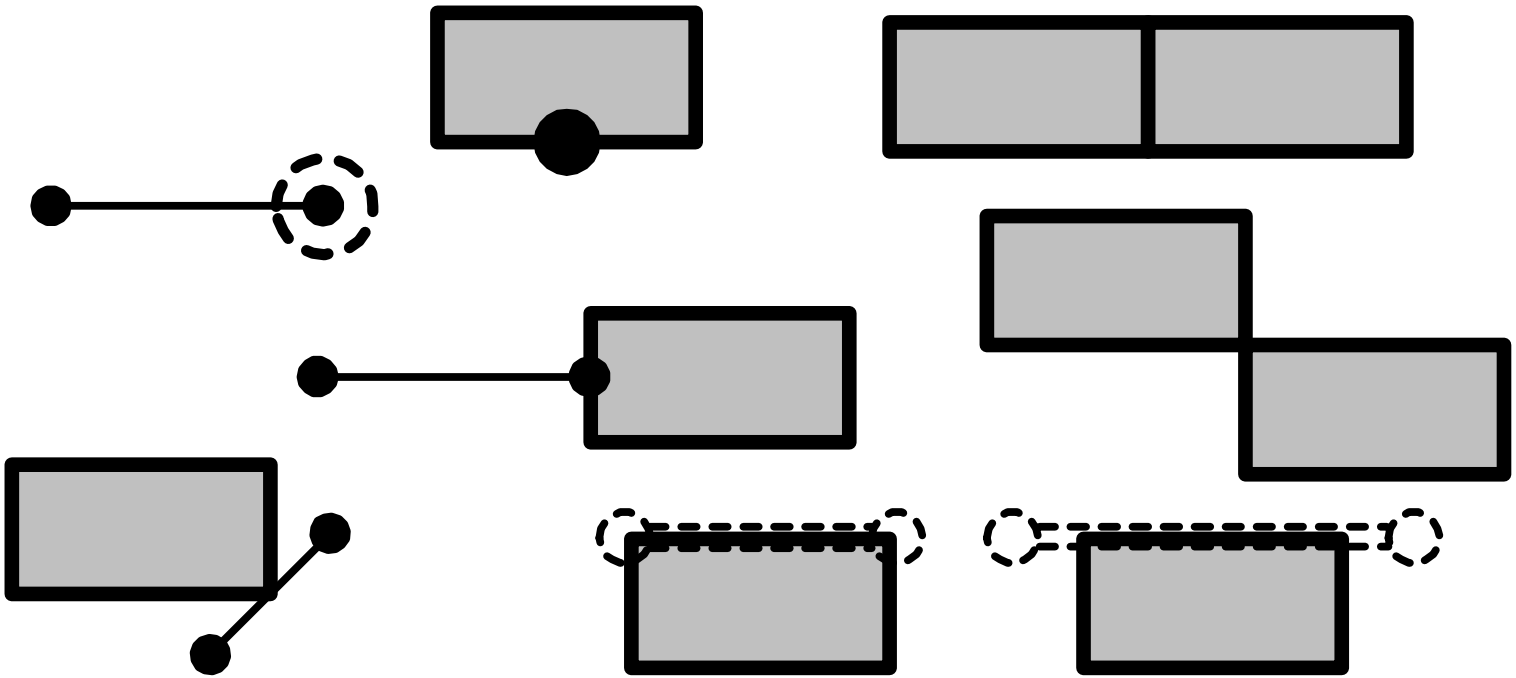
$$(\text{Boundary}(a) \cap \text{Interior}(b) \neq \varnothing) \vee$$



$$(\text{Boundary}(a) \cap \text{Boundary}(b) \neq \varnothing))$$



More ST_Touches Examples



ST_Crosses

a.ST_Crosses(b) $\Leftrightarrow$

if a and b both have dimension 0 or 2:

null value

if a and b both have dimension 1:

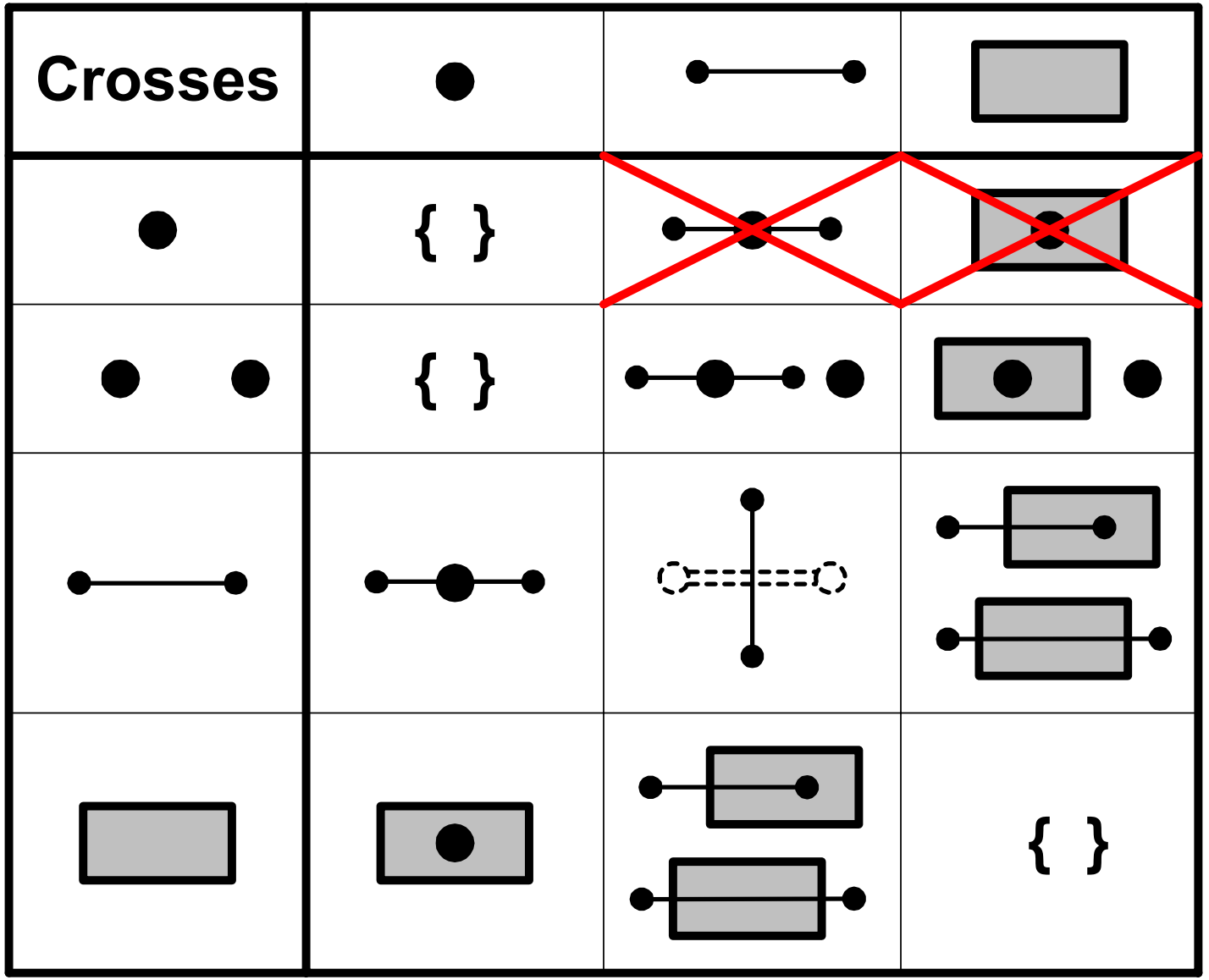
(Interior(*a*) $\cap$ Interior(*b*)).ST_Dimension = 0

else:

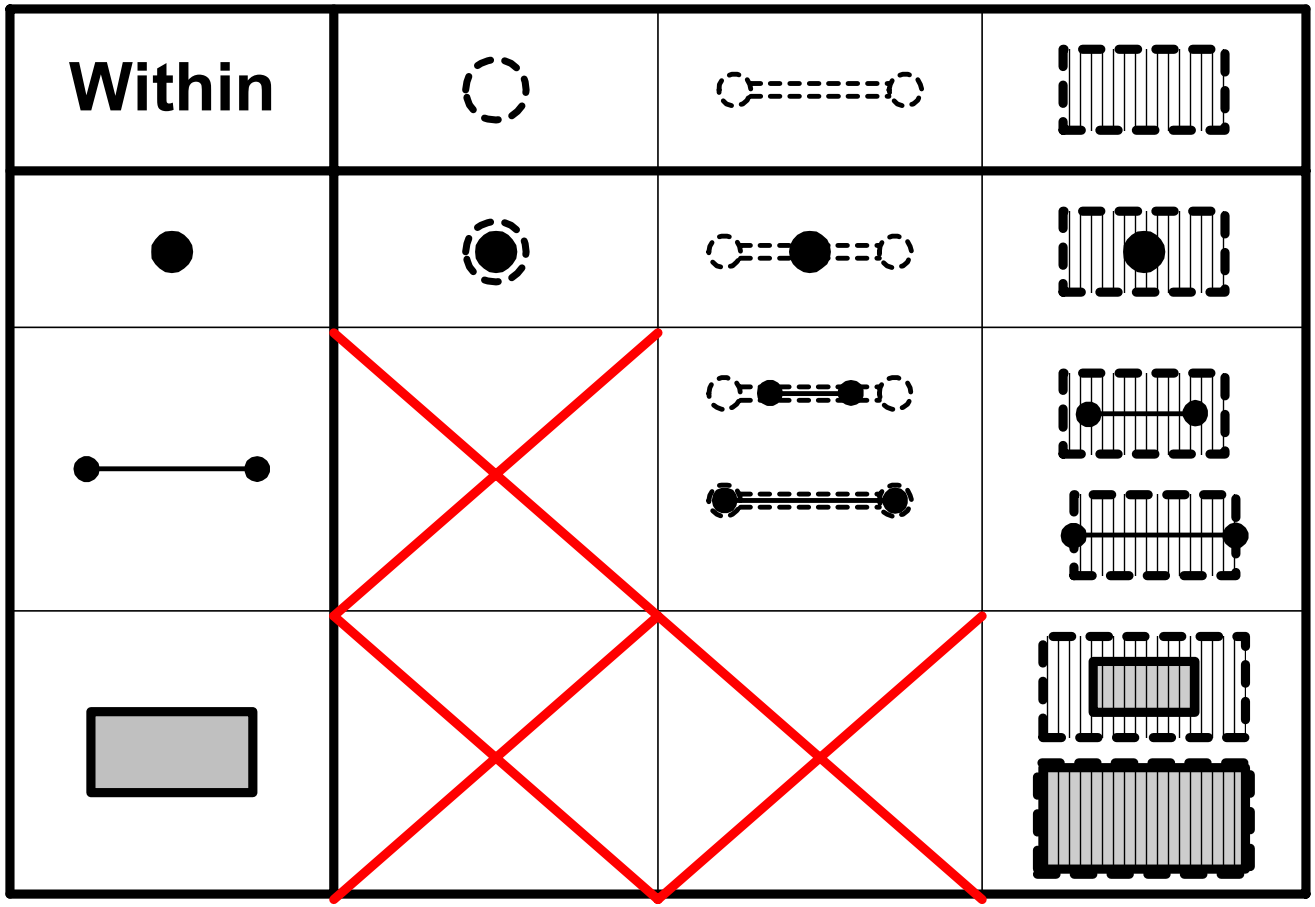
(Interior(*a*) $\cap$ Interior(*b*) $\neq \emptyset$) $\wedge$

(Interior(*a*) $\cap$ Exterior(*b*) $\neq \emptyset$)

ST_Crosses

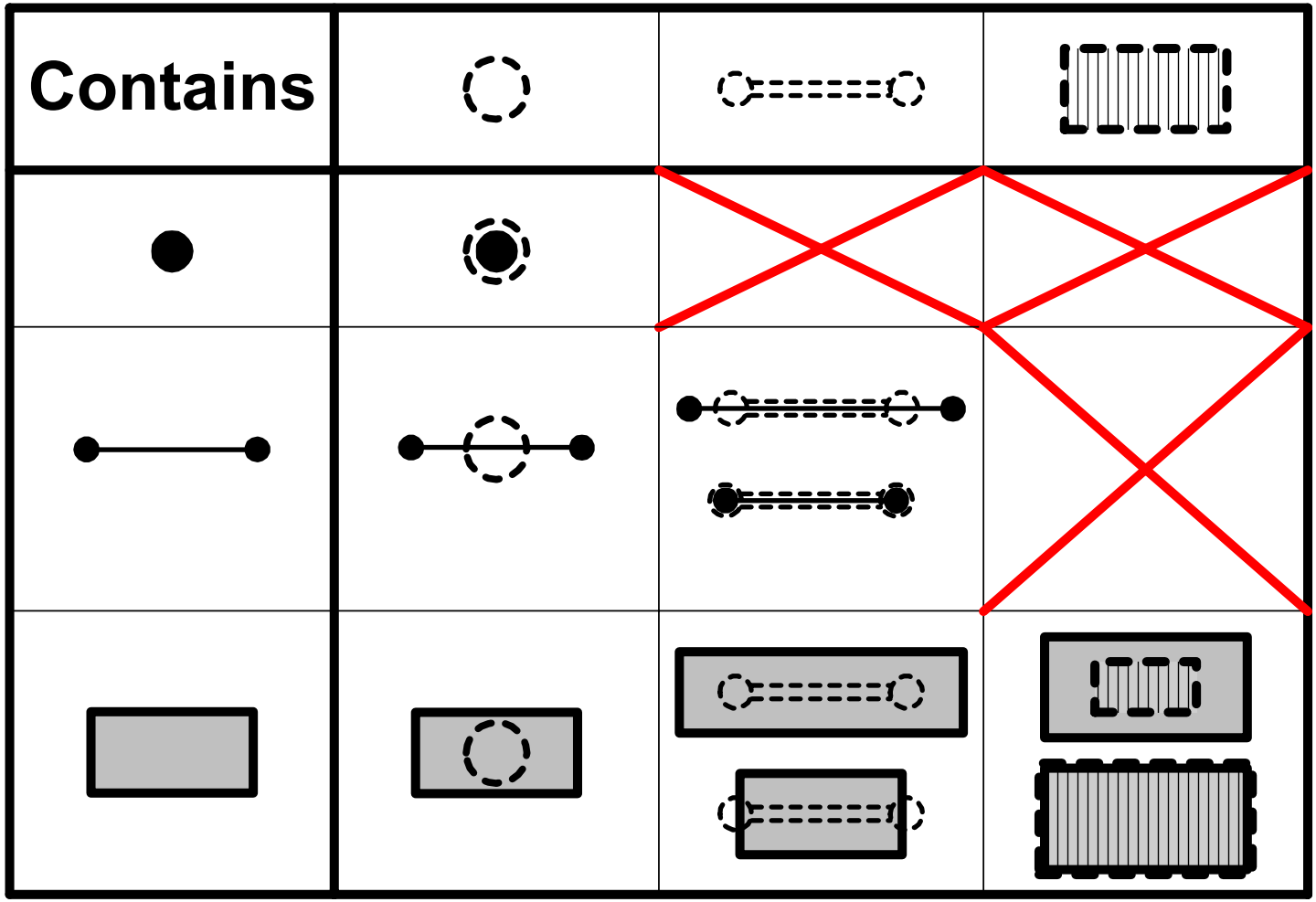


ST_Within



$$(\text{Interior}(a) \cap \text{Interior}(b) \neq \varnothing) \wedge$$
$$(\text{Interior}(a) \cap \text{Exterior}(b) = \varnothing) \wedge$$
$$(\text{Boundary}(a) \cap \text{Exterior}(b) = \varnothing))$$

ST_Contains



a.ST_Contains(*b*) ⇔ *b*.ST_Within(*a*)

ST_Overlaps

a.ST_Overlaps(b) $\Leftrightarrow$

if a and b both have dimension 0 or 2:

$(\text{Interior}(a) \cap \text{Interior}(b) \neq \emptyset) \wedge$
$(\text{Interior}(a) \cap \text{Exterior}(b) \neq \emptyset) \wedge$
$(\text{Exterior}(a) \cap \text{Interior}(b) \neq \emptyset)$

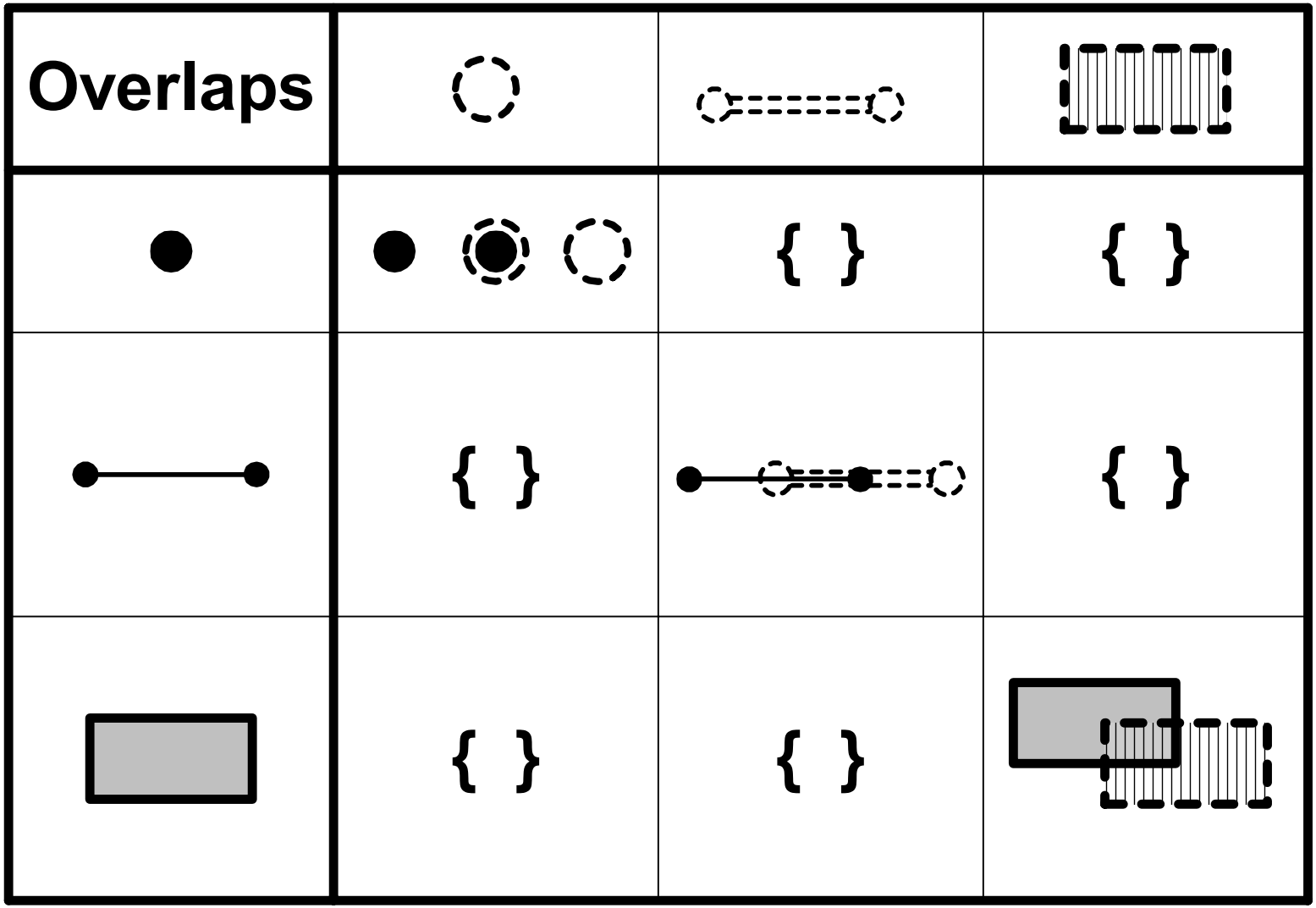
if a and b both have dimension 1:

$((\text{Interior}(a) \cap \text{Interior}(b)).\text{ST\_Dimension} = 1) \wedge$
$(\text{Interior}(a) \cap \text{Exterior}(b) \neq \emptyset) \wedge$
$(\text{Exterior}(a) \cap \text{Interior}(b) \neq \emptyset)$

else:

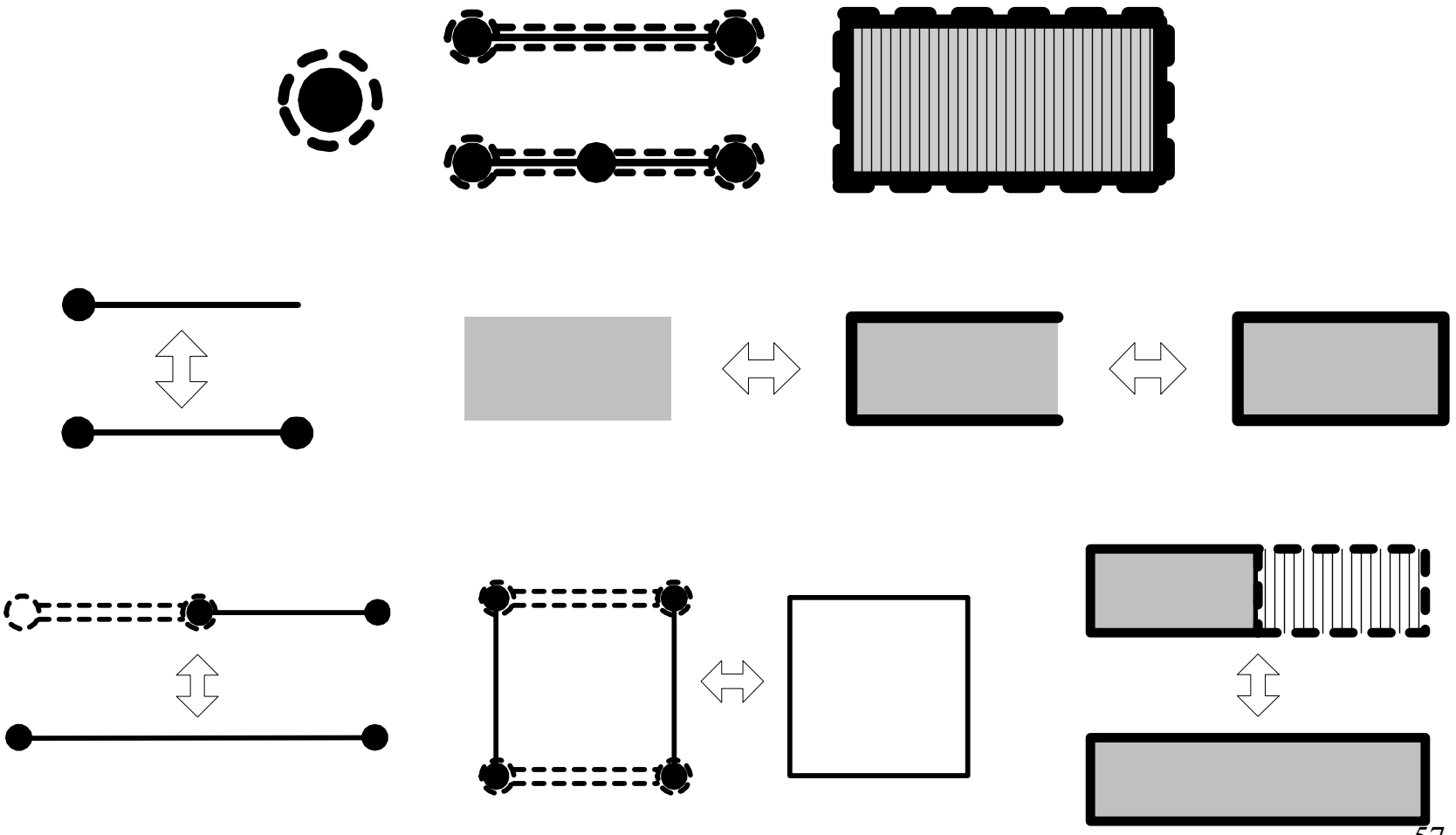
null value

ST_Overlaps



ST_Equals

$$(a - b) \cup (b - a) = \varnothing$$



Clause 5: Geometry Types

Subclause layout

5.1 ST_Geometry Type and Routines

5.1.1 ST_Geometry Type

Purpose

Definition

attributes, method declarations

Definitional Rules

Description

5.1.2-n Methods, functions

Geometry Types
ST_Geometry

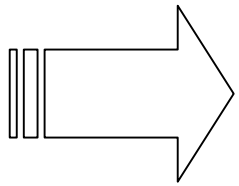
Purpose

The ST_Geometry type is the root type for geometry types. All subtypes have position specified in their attributes.

ST_Geometry

Definition

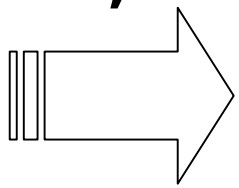
```
CREATE TYPE ST_Geometry
  AS (
    ST_PrivateDimension SMALLINT
     DEFAULT -1,
    ST_PrivateCoordinateDimension
     SMALLINT DEFAULT 2
  )
  NOT INSTANTIABLE
  NOT FINAL
```



ST_Geometry

Definition (cont'd)

```
METHOD ST_SRID()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```



ST_Geometry

Definition (cont'd)

```
METHOD ST_SRID(asrid INTEGER)
  RETURNS ST_Geometry
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```

ST_Geometry

Definitional Rules

. . .

3) The attribute *ST_PrivateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateDimension*.

ST_Geometry

Description

1) The *ST_Geometry* type provides for public use:

   a) a method *ST_Dimension()*,

. . .

9) An *ST_Geometry* value has an associated spatial reference system specified by a spatial reference system identifier.

ST_SRID Methods

Purpose

Observe and mutate the spatial reference system identifier of the ST_Geometry value.

ST_SRID Methods

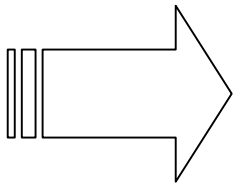
Definition

```
CREATE METHOD ST_SRID()
  RETURNS INTEGER
  FOR ST_Geometry
  -- See Description
CREATE METHOD ST_SRID(asrid
  INTEGER)
  RETURNS ST_Geometry
  FOR ST_Geometry
  -- See Description
```

ST_SRID Methods

Description

1) The method *ST_SRID()* has no input parameters.
2) The null-call method *ST_SRID()* returns the spatial reference system identifier for the *ST_Geometry* value.



ST_SRID Methods

Description (cont'd)

3) The method *ST_SRID(INTEGER)* takes the following input parameters:

   a) an INTEGER value *asrid*.

4) The parameter *asrid* is a spatial reference system identifier.

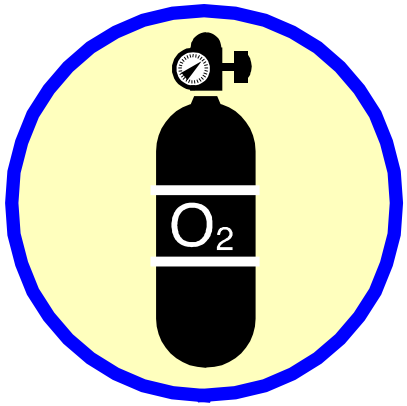
5) The type preserving method *ST_SRID(INTEGER)* returns an *ST_Geometry* value with the spatial reference system identifier set to *asrid*.

Breather

All elements of the specification of a type have been shown except:

constructor methods

WHY ?



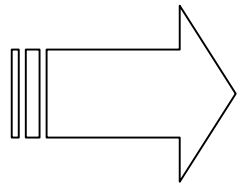
Method types

- Constructor
  - returns a value of a given type
  - equivalent to creating an (object) instance of a class in OOP
  - a type may have multiple constructors
- Observer
  - returns the value of an attribute(s)
  - e.g., the coordinates of a point
- Mutator
  - changes the value of an attribute(s)

ST_Point

Definition

```
CREATE TYPE ST_Point
  UNDER ST_Geometry
  AS (
    ST_PrivateX DOUBLE PRECISION
     DEFAULT NULL,
    ST_PrivateY DOUBLE PRECISION
     DEFAULT NULL)
  INSTANTIABLE
  NOT FINAL
```



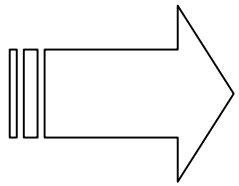
ST_Point

Definition (cont'd)

```
METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION)
  RETURNS ST_Point
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

ST_Point Method

```
CREATE METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION)
  RETURNS ST_Point
  FOR ST_Point
```

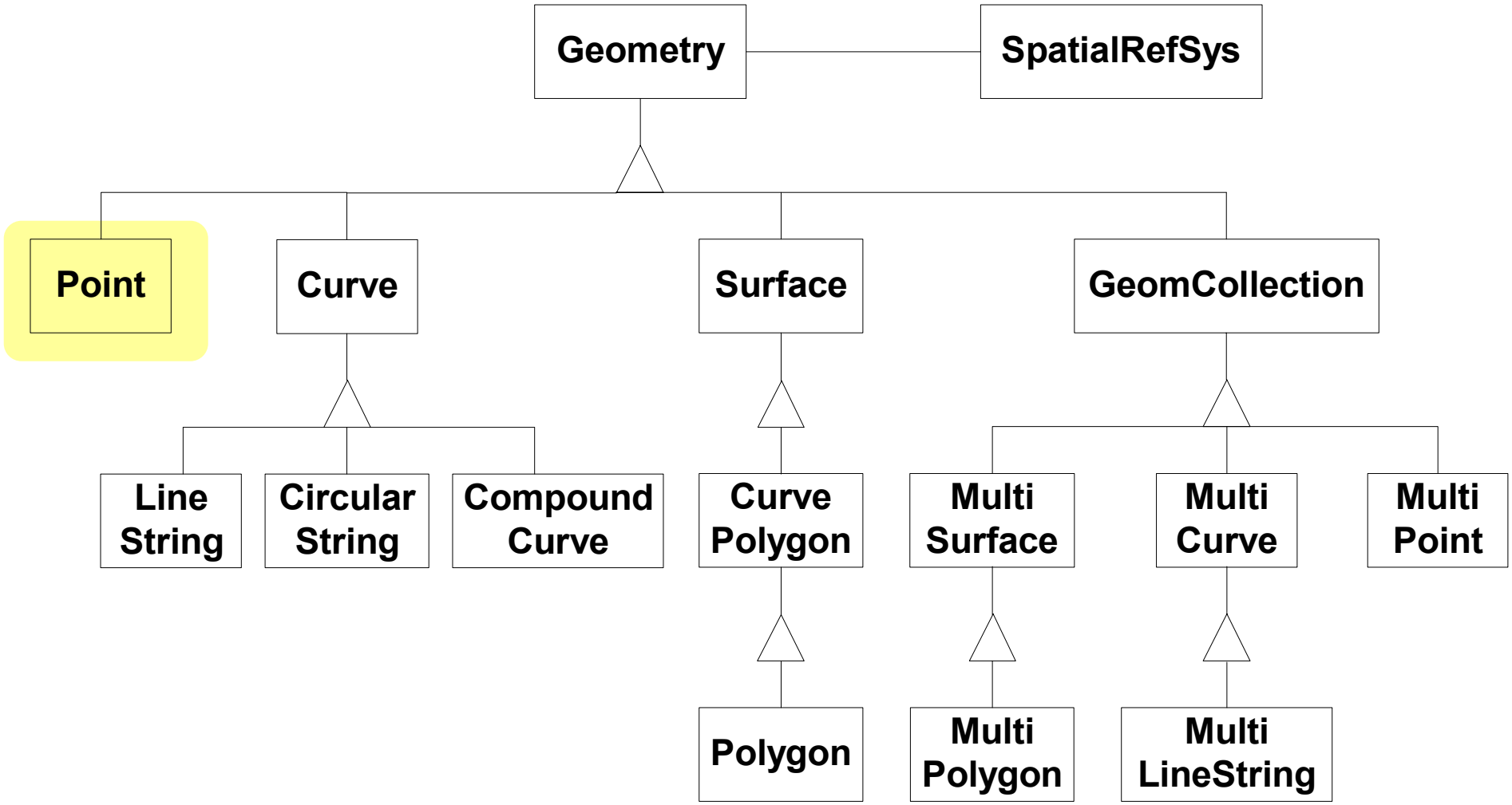


ST_Point Method (Cont'd)

```
RETURN SELF.
    -- Return an ST_Point value with
ST_PrivateDimension(0).
                    -- dimension = 0,
ST_PrivateCoordinateDimension(2).
        -- coordinate dimension = 2,
ST_SRID(0).         -- SRID = 0,
ST_X(xcoord).       -- ST_X = xcoord,
ST_Y(ycoord)        -- ST_Y = ycoord
```

Back to Concepts

ST_Point



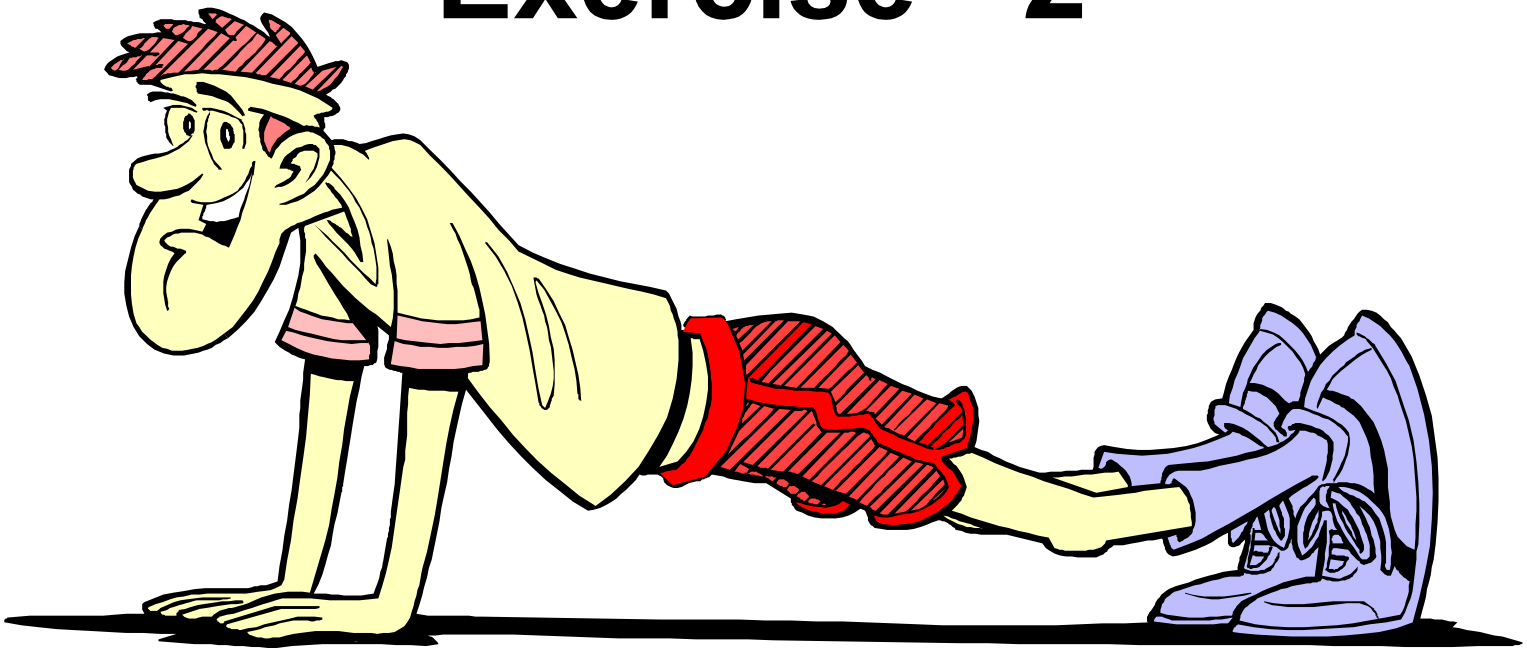
Concepts
ST_Point

- instantiable subtype of ST_Geometry
- 0-dimensional geometry
- represents a single location in two-dimensional coordinate space (R^2)
- has an x coordinate value and a y coordinate value
- boundary of an ST_Point value is the empty set
- ST_Point values are simple

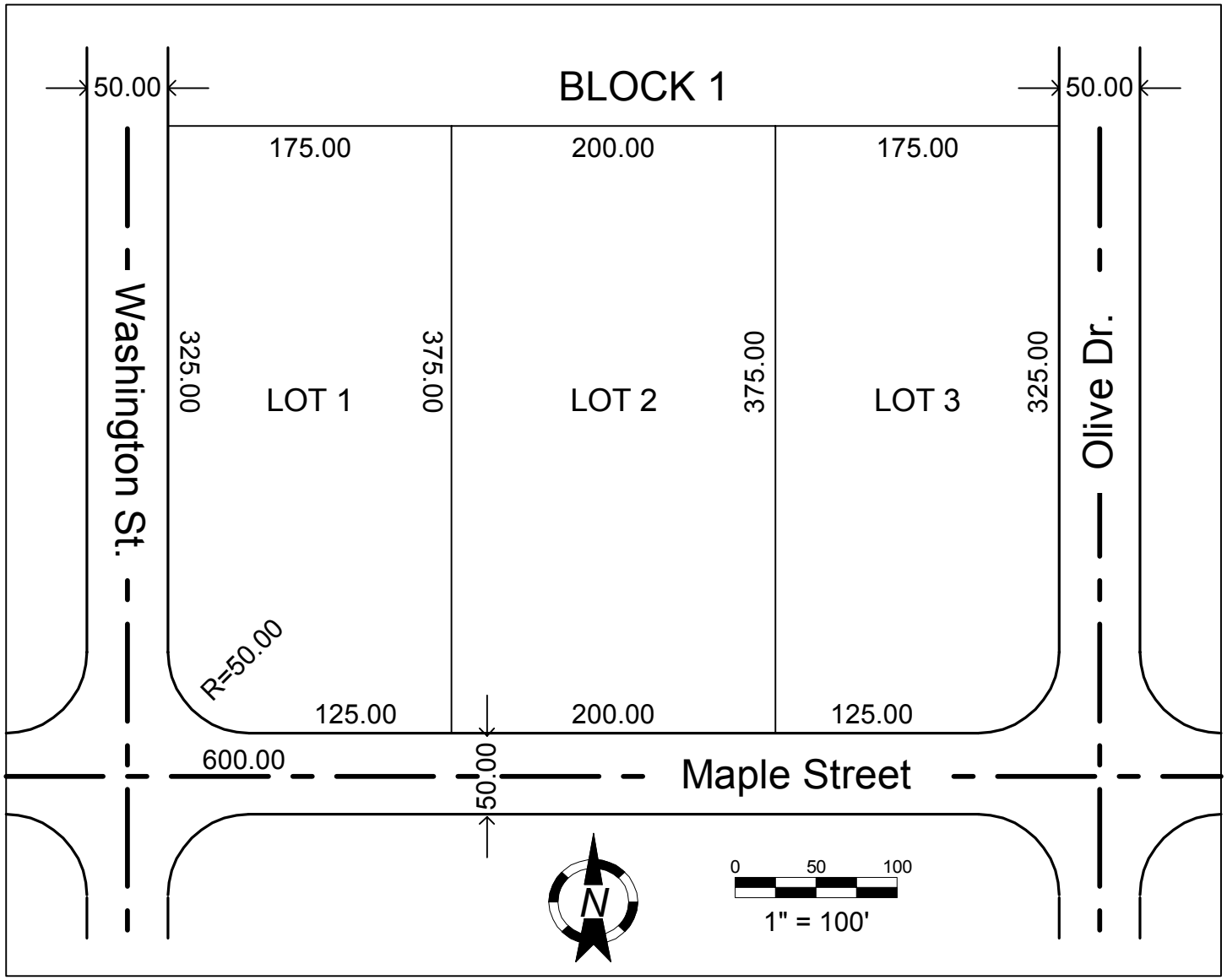
Concepts
ST_Point Methods

- ST_Point: returns the specified ST_Point value.
- ST_X: observes and mutates the x coordinate value of an ST_Point value.
- ST_Y: observes and mutates the y coordinate value of an ST_Point value.
- ST_ExplicitPoint: returns the coordinate values as a DOUBLE PRECISION ARRAY value.

Exercise - 2



Exercise 2



Exercise 2

- Provide the SQL command to create a table of intersections whose attributes include two crossing street names and a geographic location

Exercise 2

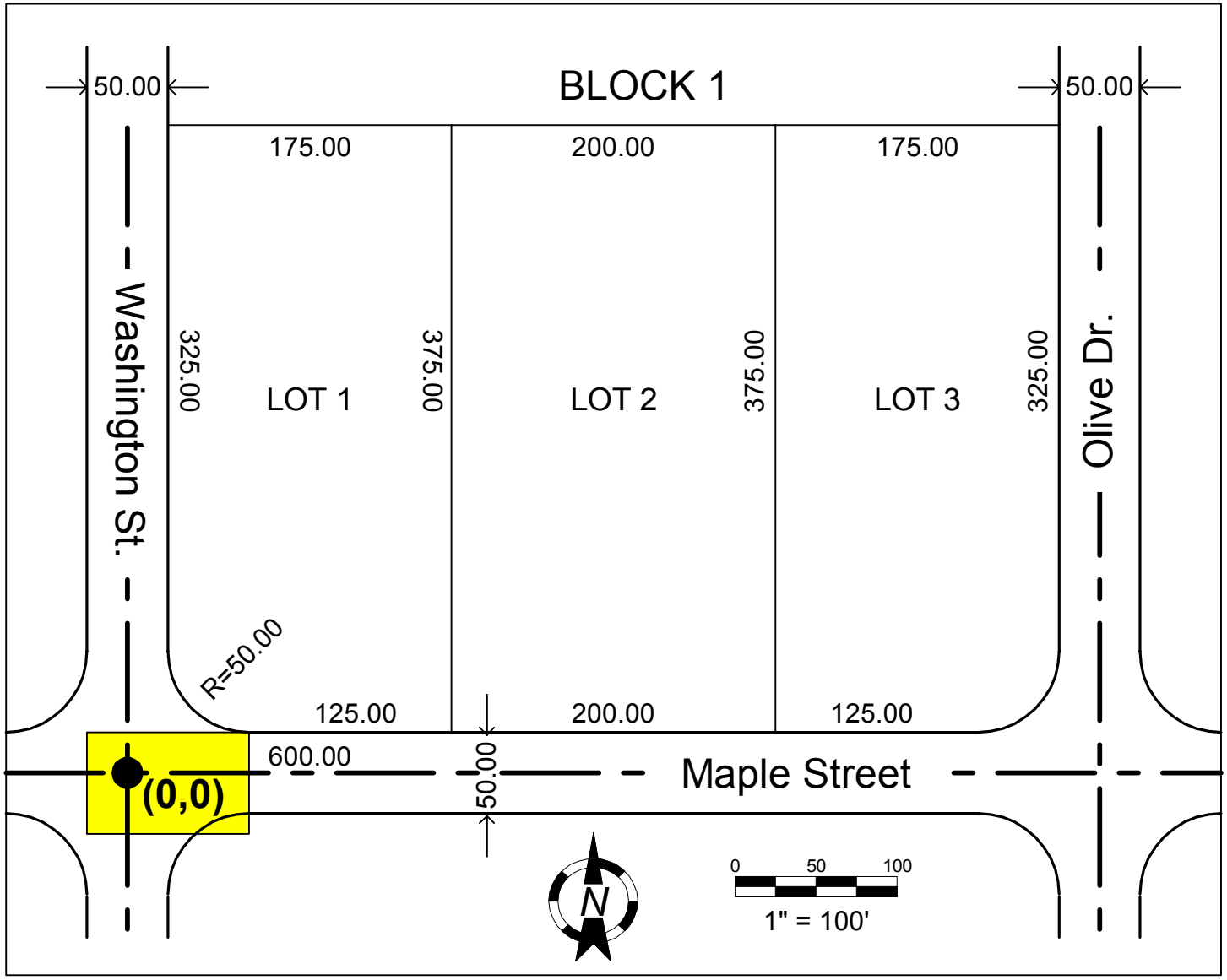
- Provide the SQL command to create a table of intersections whose attributes include two crossing street names and a geographic location

```
CREATE TABLE INTERSECTION  (
        ID                      INTEGER,
        STREET1                 VARCHAR(30),
        STREET2                 VARCHAR(30),
        LOCATIONST_Point )
```

Exercise 2

- Using SQL, add a row for the intersection of Maple Street and Washington St. Assume the map origin (0,0) is at this point.

Exercise 2

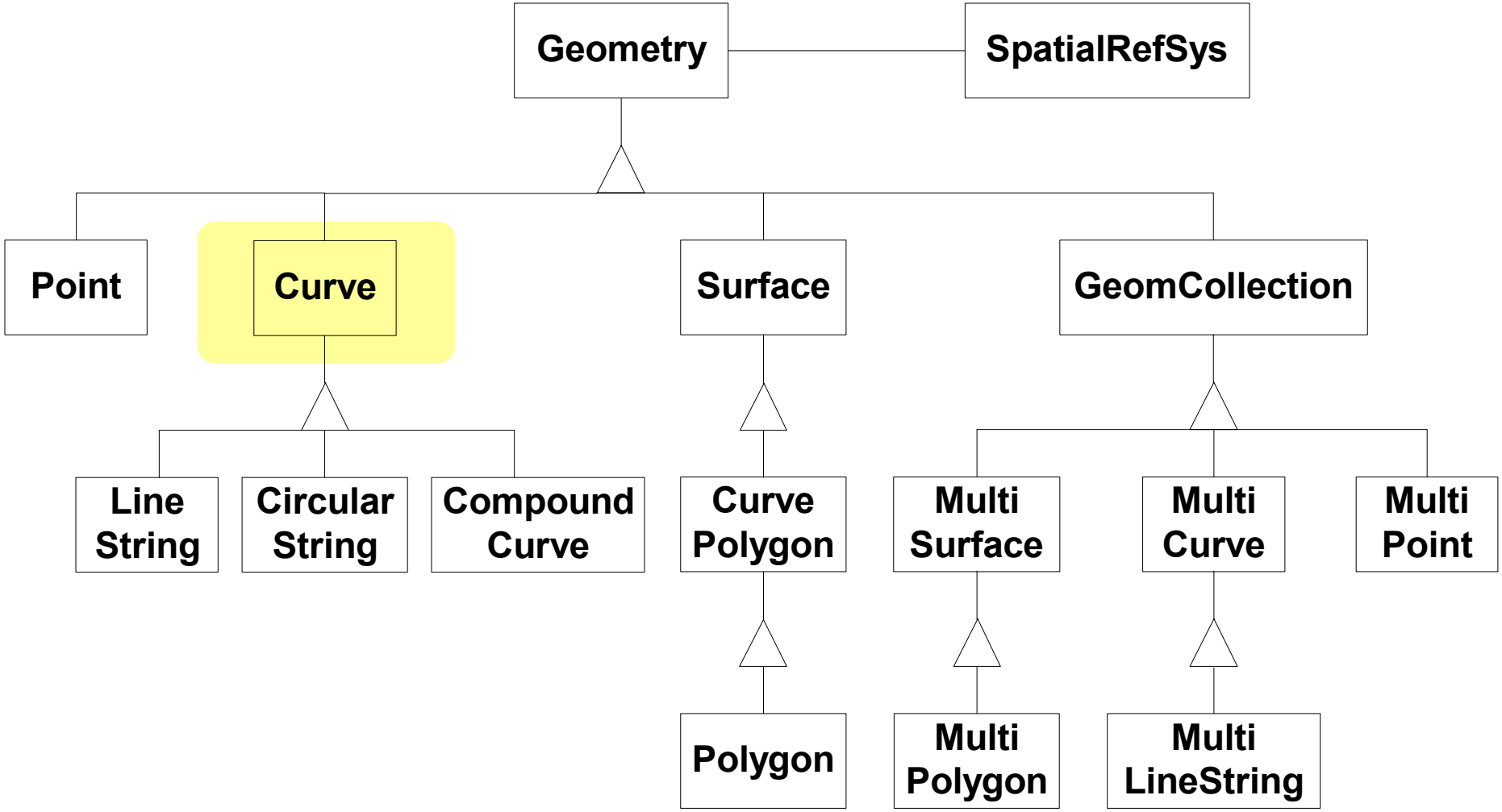


Exercise - 2

- Using SQL, add a row for the intersection of Maple Street and Washington St. Assume the map origin (0,0) is at this point.

```
INSERT INTO INTERSECTION VALUES
        ( 1, 'Maple Street', 'Washington St.',
          NEW ST_Point (0,0))
```

ST_Curve



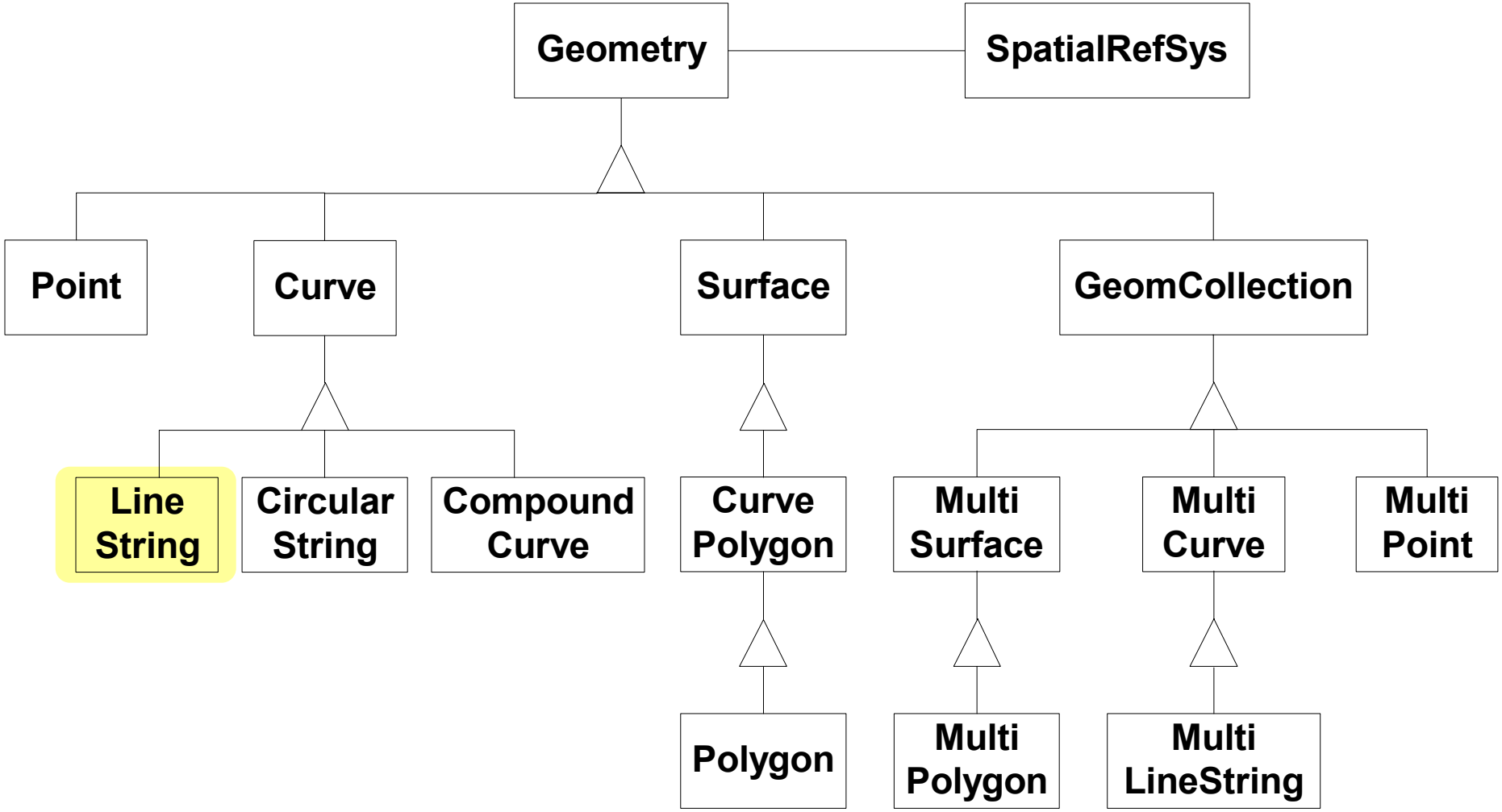
Concepts
ST_Curve

- non-instantiable subtype of ST_Geometry
- 1-D geometry; usually a sequence of points
- interpolation method defined by subtypes
- simple if it does not pass through the same point twice
- closed if start point equals end point
- ring is simple and closed
- if closed, boundary is the empty set; else, start point and end point

Concepts
ST_Curve Methods

- ST_Length: returns the length
- ST_StartPoint: returns the ST_Point value that is the start point
- ST_EndPoint: returns the ST_Point value that is the end point.
- ST_IsClosed: tests if closed.
- ST_IsRing: tests if a ring.
- ST_CurveToLine: returns the ST_LineString value approximation

ST_LineString



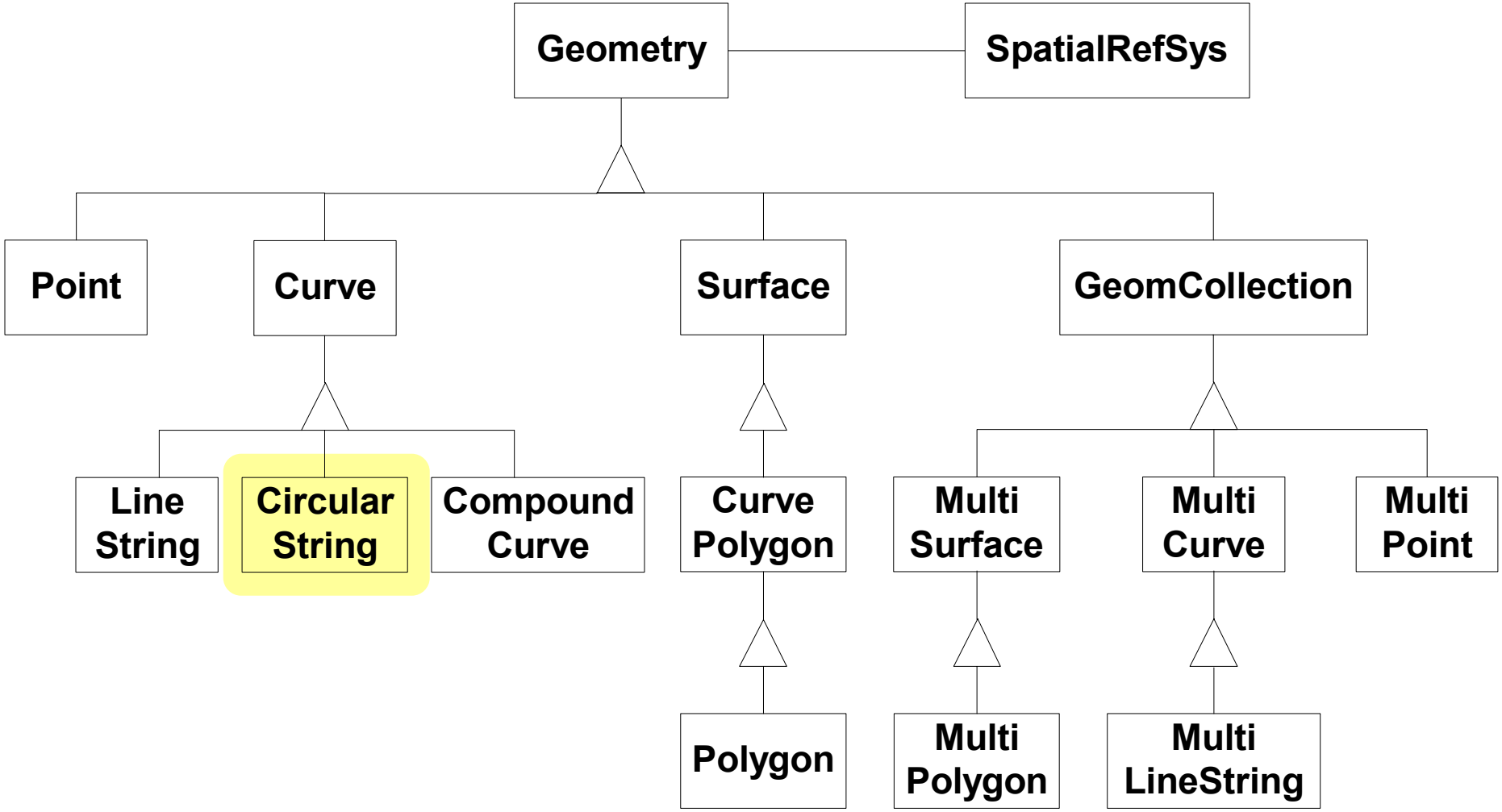
Concepts
ST_LineString

- instantiable subtype of ST_Curve
- sequence of at least two ST_Point values
- linear interpolation between the ST_Point values
- each pair of ST_Points defines a line segment
- line if only two ST_Points
- linear ring if simple and closed

Concepts
ST_LineString Methods

- ST_LineString: returns the specified ST_LineString value
- ST_Points: observes and mutates the ST_Point collection
- ST_NumPoints: returns the cardinality of the ST_Point collection
- ST_PointN: returns the specified element in the ST_Point collection

ST_CircularString



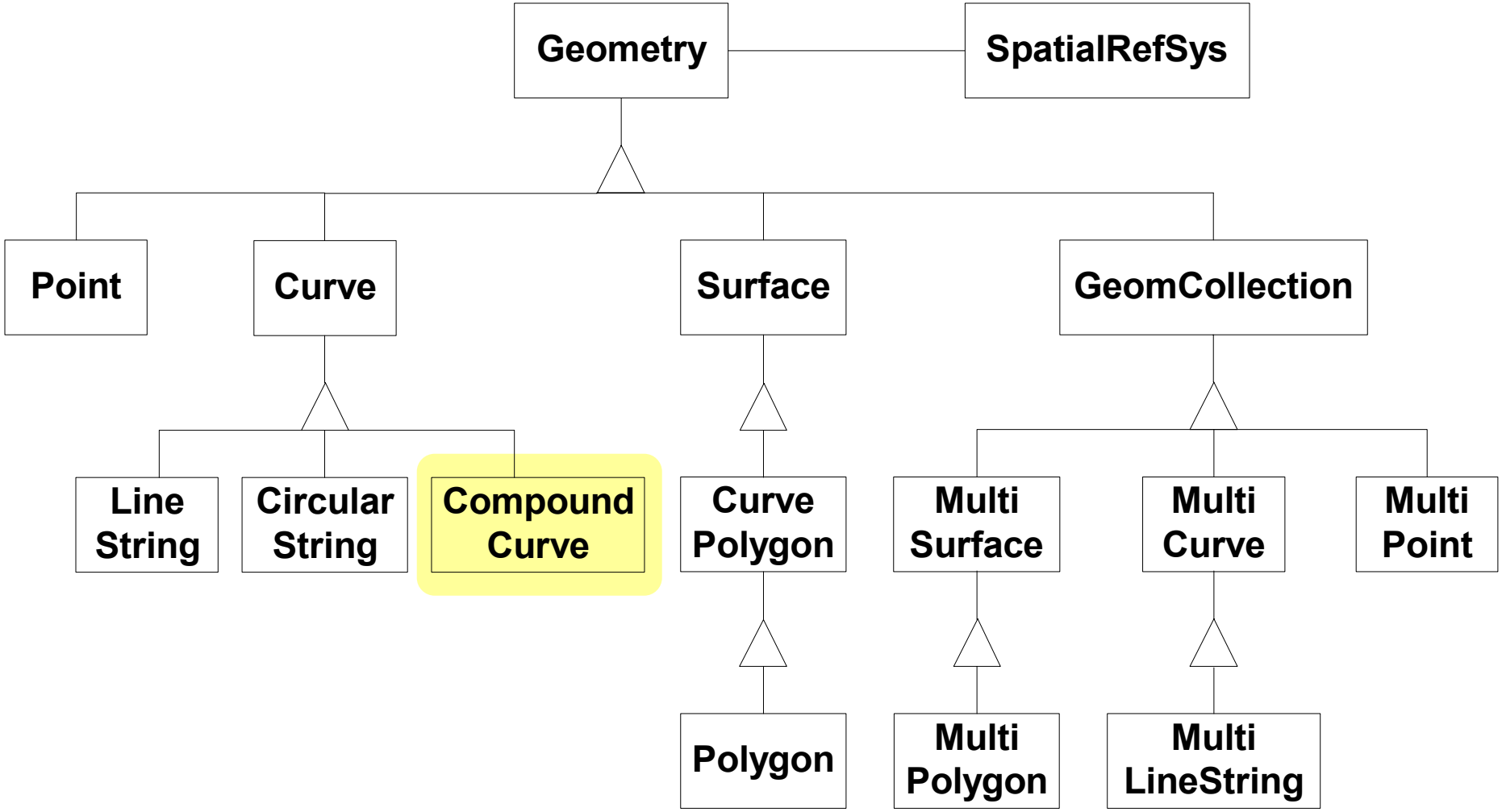
Concepts
ST_CircularString

- circular interpolation subtype of ST_Curve
- sequence of ST_Points defining arc segments
- first three points are start, intermediate, and end points of the first arc segment
- subsequent points are intermediate and end points of the next arc segment
- circular arc if only three ST_Points
- circular ring if simple and closed
- arc can degenerate to a straight line

Concepts
ST_CircularString Methods

- ST_CircularString: returns the specified ST_CircularString value.
- ST_Points: observes and mutates the ST_Point collection
- ST_NumPoints: returns the cardinality
- ST_PointN: returns the specified ST_Point
- ST_MidPointRep: returns the array of points which identify an ST_CircularString value including start, mid, and end points

ST_CompoundCurve



Concepts
ST_CompoundCurve

- variable interpolation subtype of ST_Curve
- sequence of 1 or more contiguous curves of type ST_LineString and/or ST_CircularString
- end point of each curve equals the start point of the next curve

Concepts
ST_CompoundCurve Methods

- ST_CompoundCurve: returns the specified ST_CompoundCurve value.
- ST_Curves: observes and mutates the ST_Curve collection
- ST_NumCurves: returns the cardinality of the ST_Curve collection
- ST_CurveN: returns the specified element in the ST_Curve collection

Exercise - 3

Exercise 3

- Provide the SQL command to create a table of streets whose attributes include street name and centerline geometry.

Exercise 3

- Provide the SQL command to create a table of streets whose attributes include street name and centerline geometry

```
CREATE TABLE STREET  (
        NAME                VARCHAR(30),
        GEOMETRY        ST_Curve )
```

Exercise 3

- Using SQL, add a row for Maple Street between Washington St. and Olive Dr. (The ST_LineString constructor is on the next page).

Exercise 3

METHOD ST_LineString

(apointarray ST_Point
ARRAY[*ST_MaxGeometryArrayElements*])

RETURNS ST_LineString

SELF AS RESULT

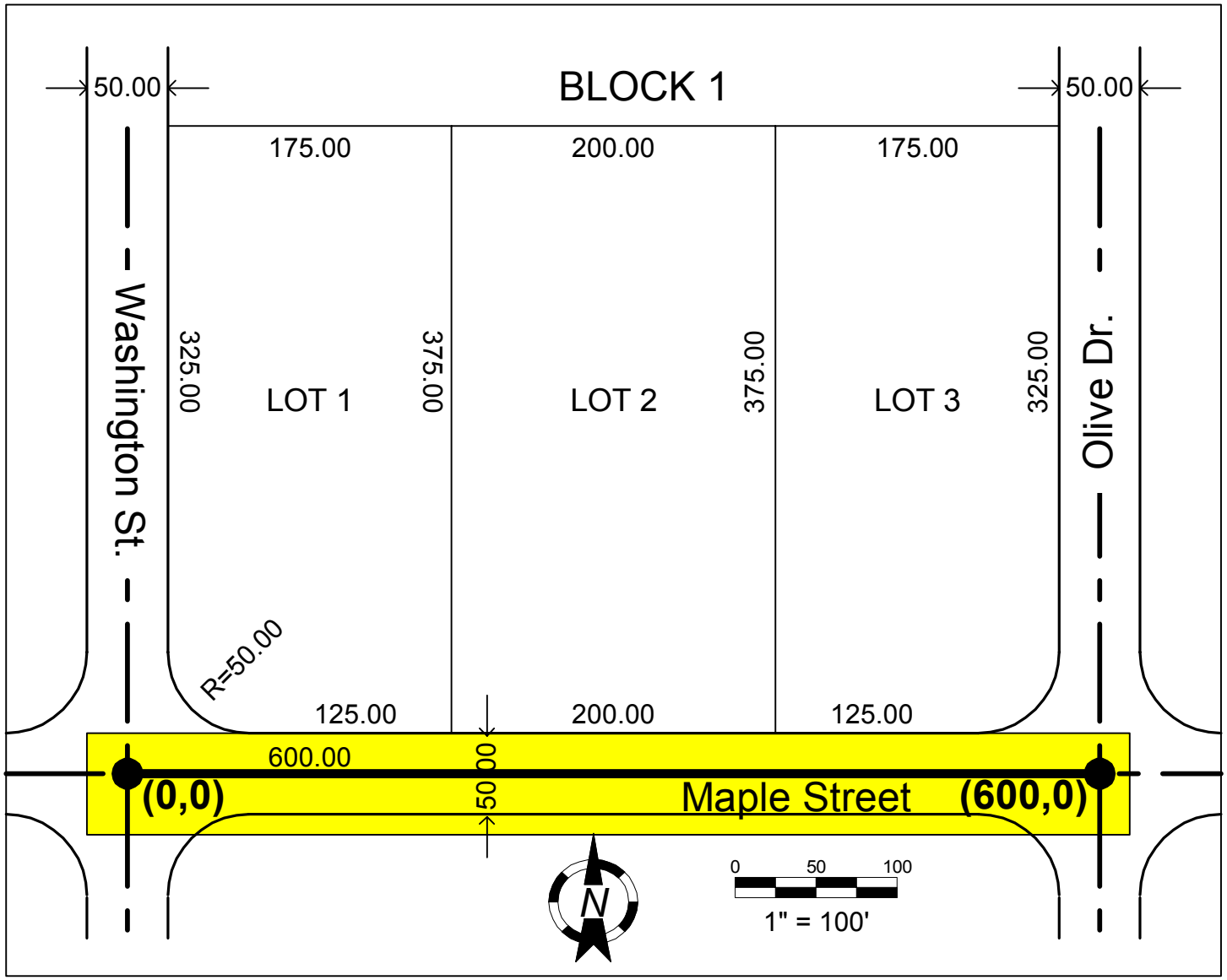
LANGUAGE SQL

DETERMINISTIC

CONTAINS SQL

RETURNS NULL ON NULL INPUT

Exercise - 3

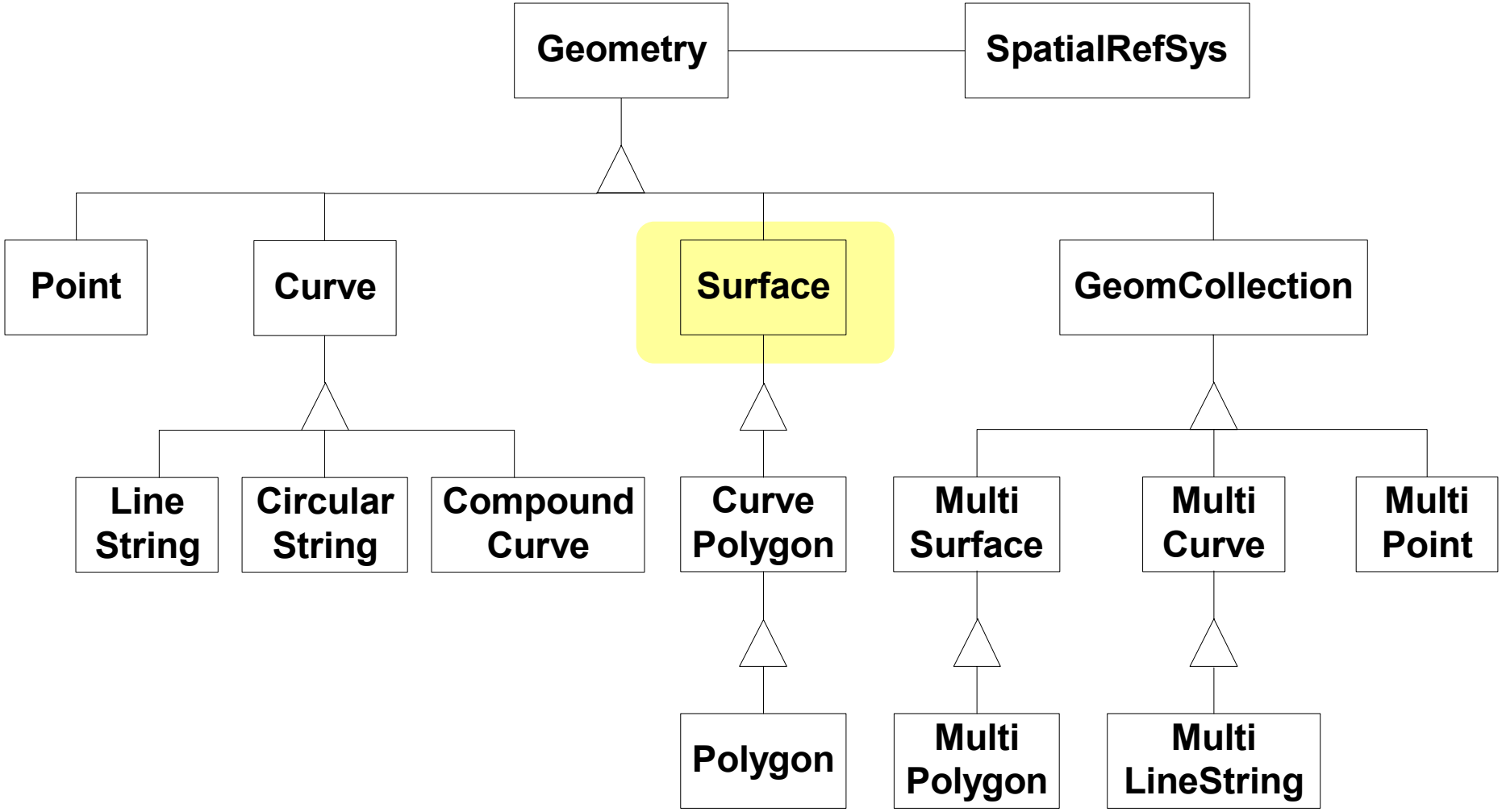


Exercise 3

- Using SQL, add a row for Maple Street between Washington St. and Olive Dr.

```
INSERT INTO STREET VALUES
        ( 'Maple Street',
          NEW ST_LineString
                (ARRAY
                  [ NEW ST_Point (0,0),
                    NEW ST_Point (600,0) ] ) )
```

ST_Surface



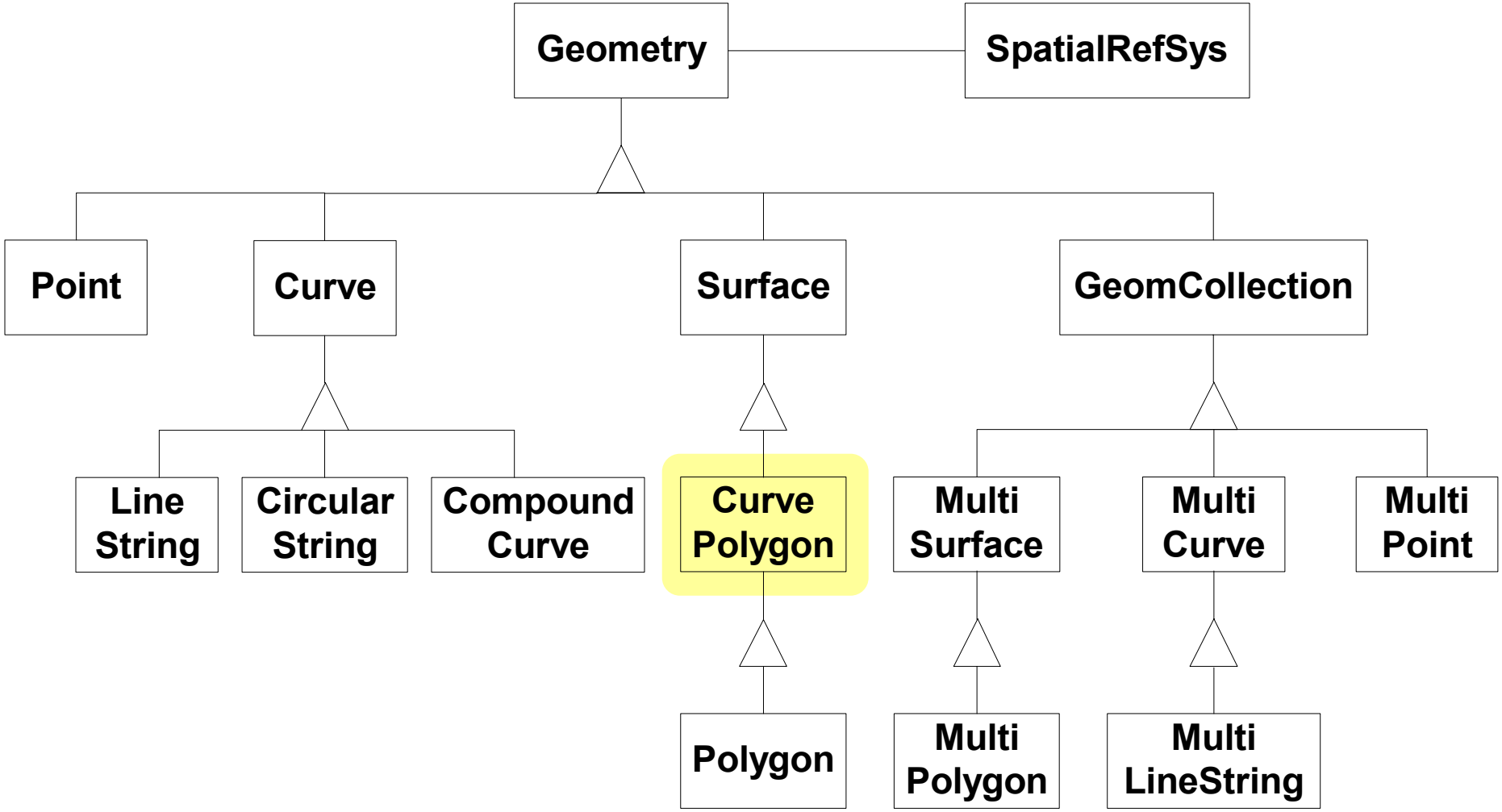
Concepts
ST_Surface

- non-instantiable subtype of ST_Geometry
- 2-dimensional geometry defined as a simple surface consisting of a single patch whose boundary is specified by one exterior ring and zero or more interior rings
- boundary is the set of closed curves corresponding to its exterior and interior rings

Concepts
ST_Surface Methods

- ST_Area: returns the area
- ST_Perimeter: returns the length of the perimeter of the ST_Surface value.
- ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_Surface value.
- ST_PointOnSurface: returns the ST_Point value that is guaranteed to be on the ST_Surface value

ST_CurvePolygon



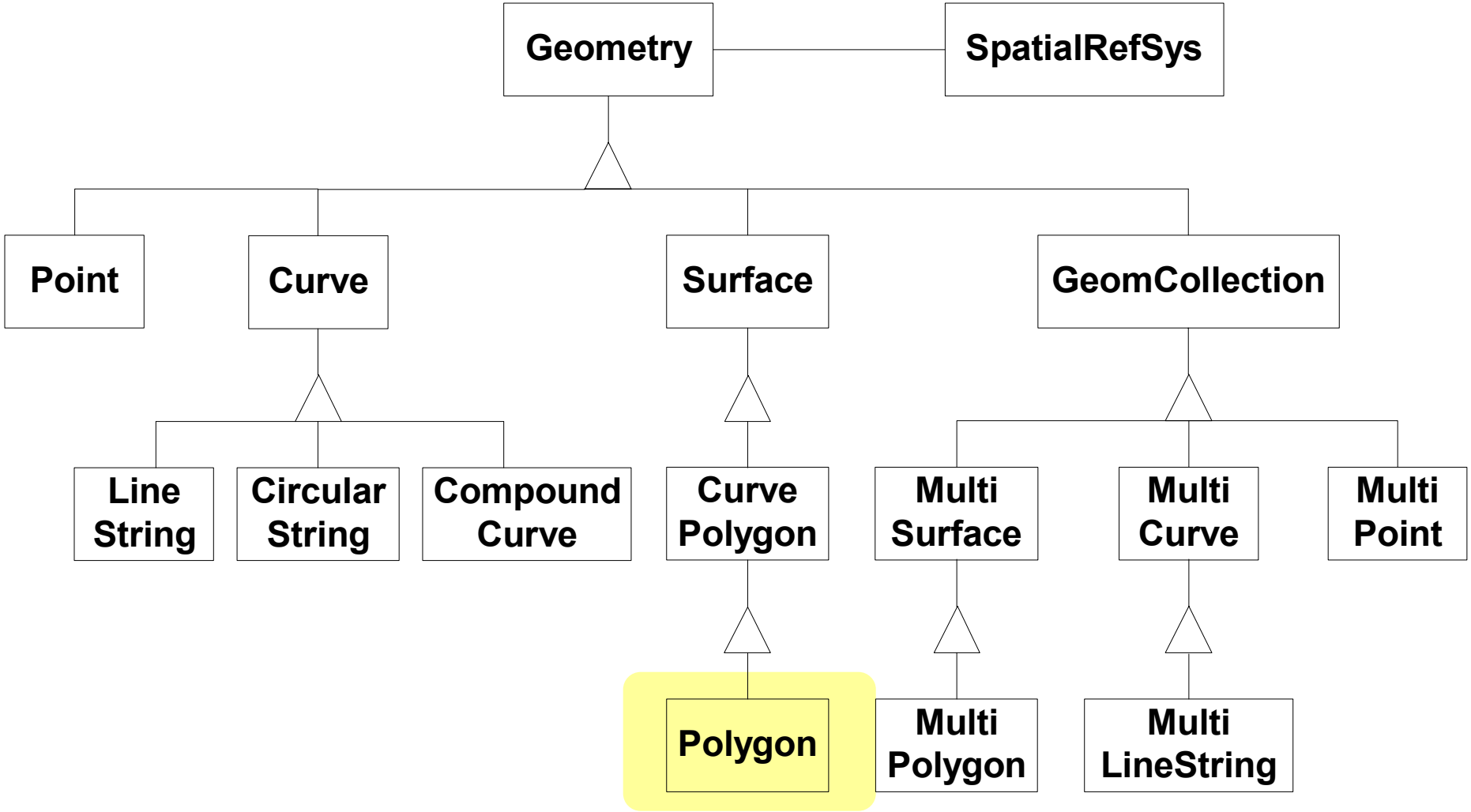
Concepts
ST_CurvePolygon

- instantiable subtype of ST_Surface
- planar surface defined by one exterior and zero or more interior boundaries
- boundary is the set of exterior and interior rings
- no two rings cross but may intersect tangentially at a point
- interior is a connected point set; exterior is not if interior rings exist (defining holes)
- ST_CurvePolygon values are simple

Concepts
ST_CurvePolygon Methods

- ST_CurvePolygon: returns the specified ST_CurvePolygon value
- ST_ExteriorRing: observes, mutates the exterior ring
- ST_InteriorRings: observes and mutates the collection of interior rings
- ST_NumInteriorRing: returns the cardinality of the collection of interior rings
- ST_InteriorRingN: returns the specified interior ring
- ST_CurvePolyToPoly: returns an ST_Polygon value approximating the ST_CurvePolygon value

ST_Polygon



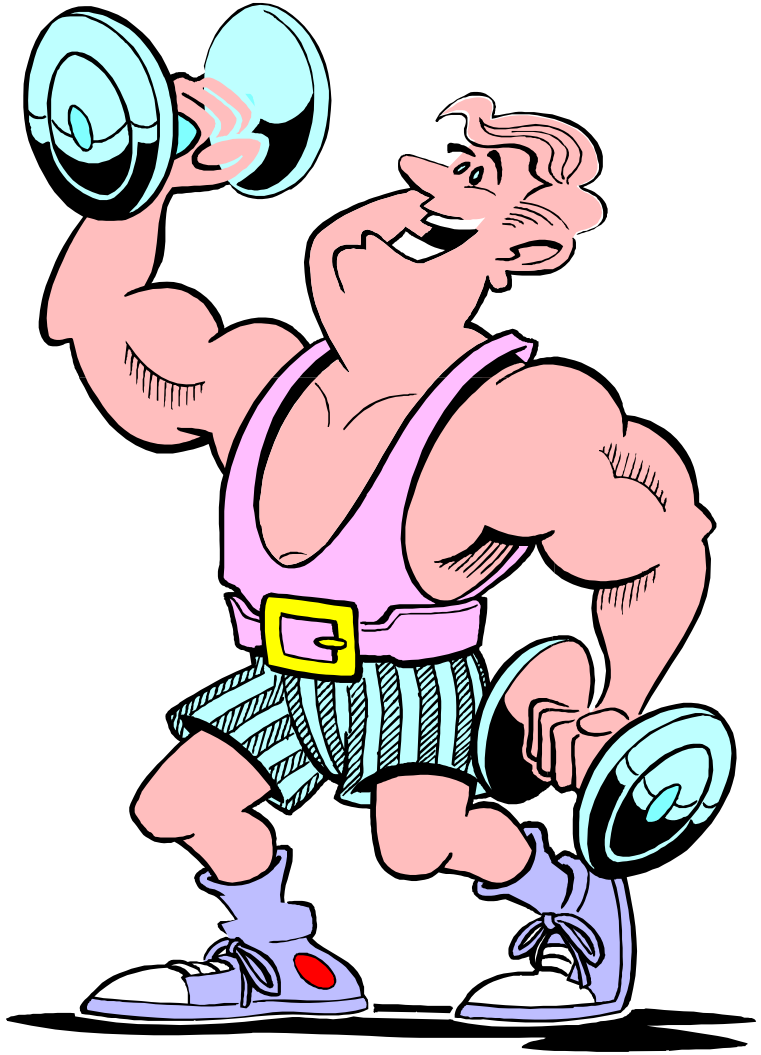
Concepts
ST_Polygon

- instantiable subtype of ST_CurvePolygon
- boundary is defined by linear rings

Concepts
ST_Polygon Methods

- ST_Polygon: returns the specified ST_Polygon value

Exercise - 4



Exercise 4

- Provide the SQL command to create a table of land parcels (lots) whose attributes include lot and block numbers and geometry

Exercise 4

- Provide the SQL command to create a table of land parcels (lots) whose attributes include lot and block numbers and geometry.

```
CREATE TABLE PARCEL  (
        BLOCK               INTEGER,
        LOT                 INTEGER,
        GEOMETRY            ST_CurvePolygon )
```

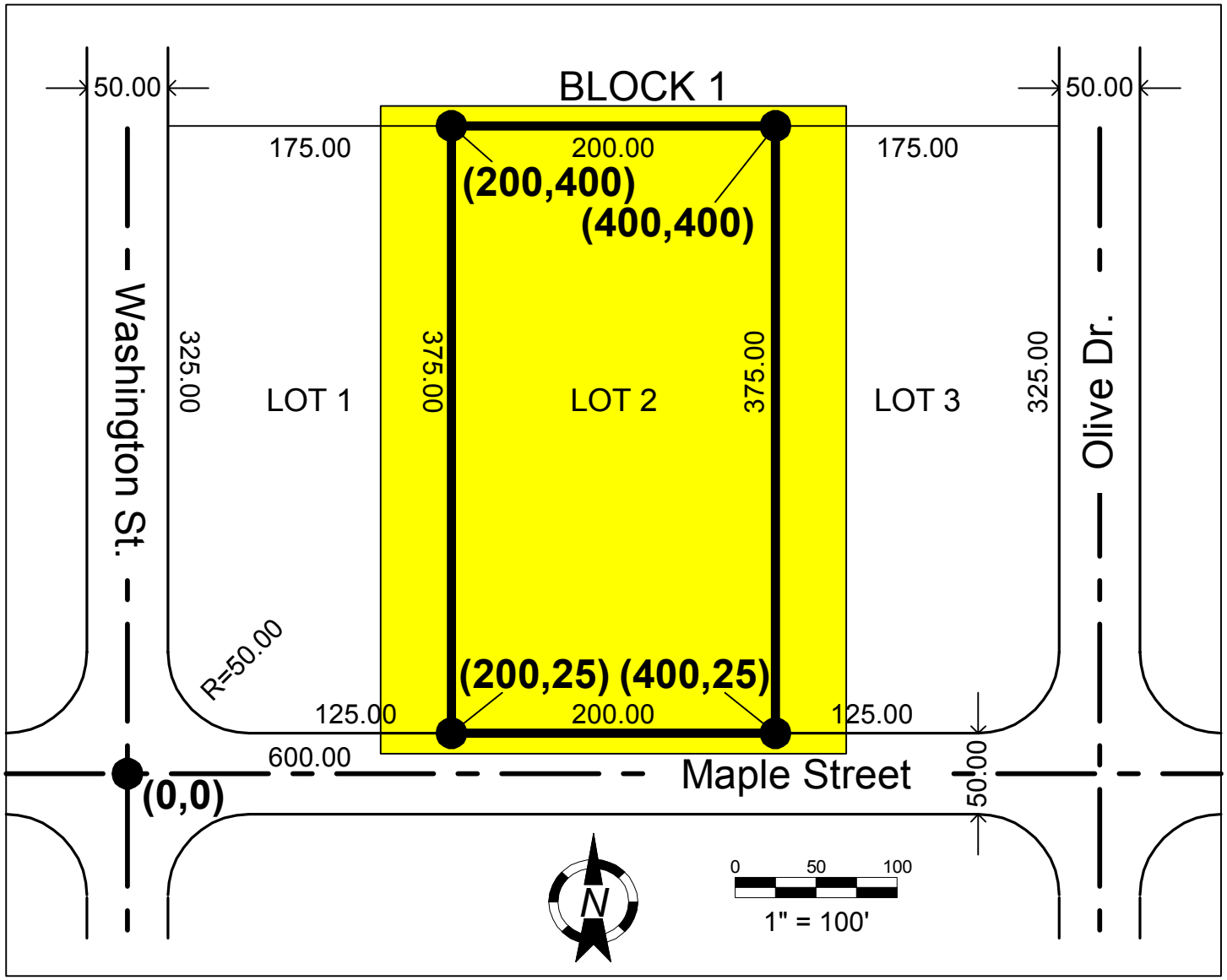
Exercise 4

- Using SQL, add a row for Lot 2. (The ST_Polygon constructor is on the next page).

Exercise 4

```
METHOD ST_Polygon
    (alinestring ST_LineString)
    RETURNS ST_Polygon
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
```

Exercise - 4



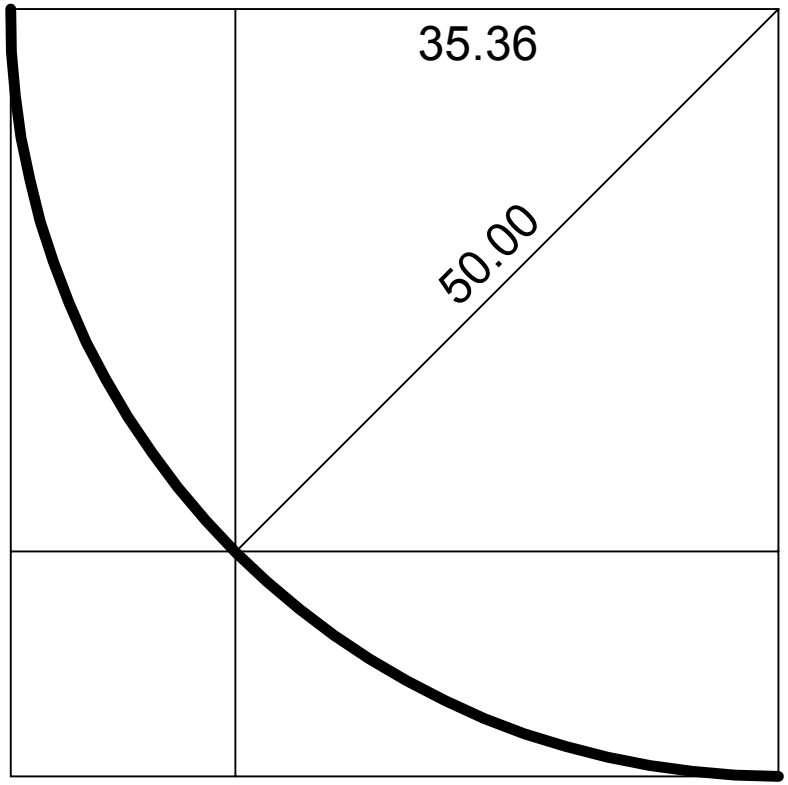
Exercise 4

- Using SQL, add a row for Lot 2.

```
INSERT INTO PARCEL VALUES
   ( 1, 2, NEW ST_Polygon
        (NEW ST_LineString  (ARRAY
                [ NEW ST_Point (200,25),
                  NEW ST_Point (400,25),
                  NEW ST_Point (400,400),
                  NEW ST_Point (200,400),
                  NEW ST_Point (200,25) ] ) ) )
```

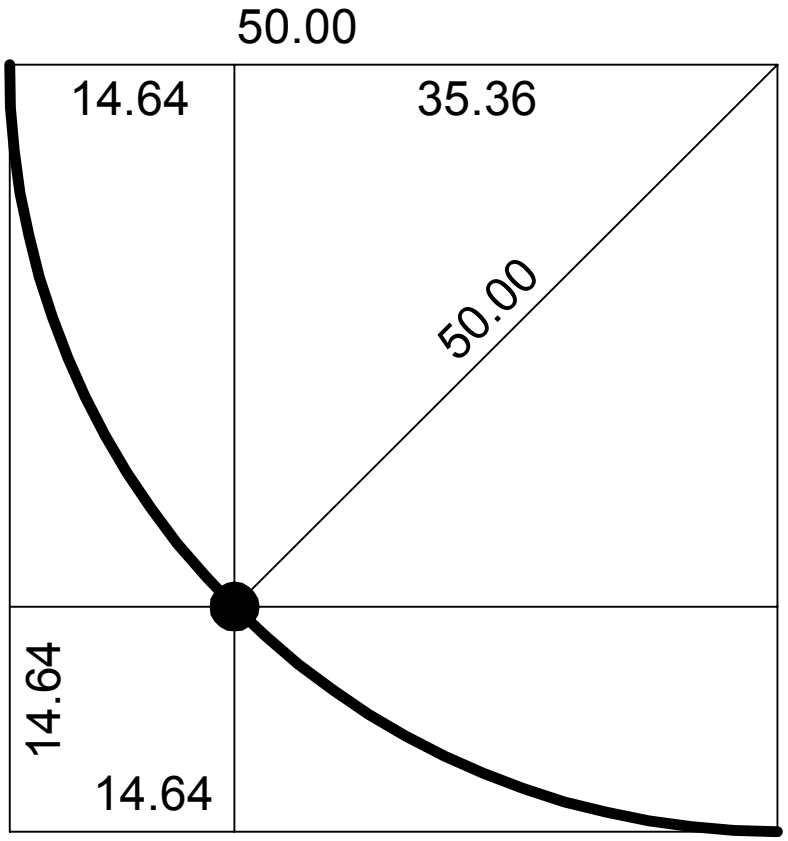
Exercise 4

- Using SQL, add a row for Lot 1. The following sketch will be helpful:

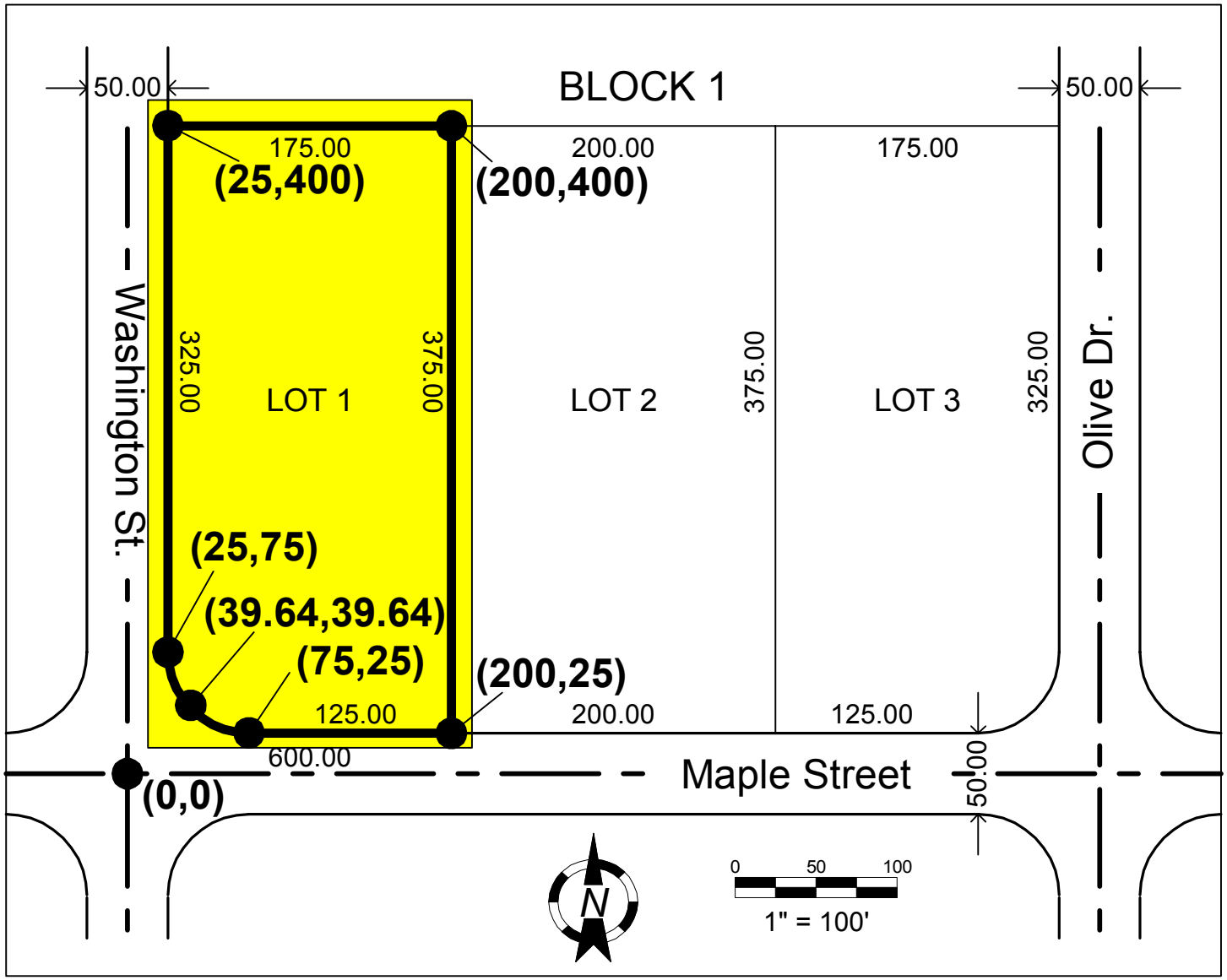


Exercise 4

- Using SQL, add a row for Lot 1.



Exercise 4

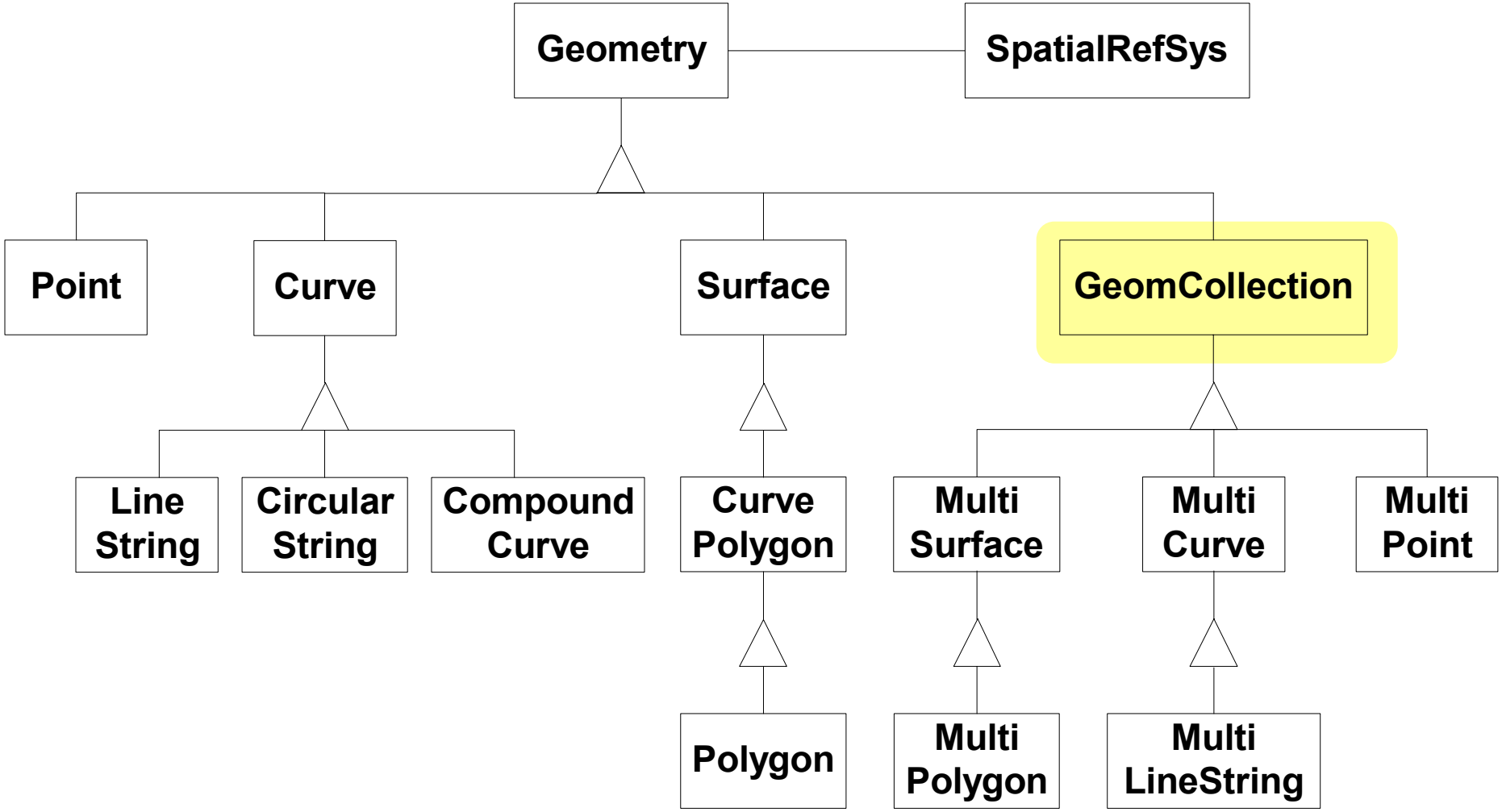


```
INSERT INTO PARCEL VALUES
   ( 1, 1, NEW ST_CurvePolygon
        ( NEW ST_CompoundCurve ( ARRAY
                [ NEW ST_CircularString (ARRAY
                      [ NEW ST_Point (25,75),
                        NEW ST_Point (39.64,39.64),
                        NEW ST_Point (75,25) ] ) ,
                  NEW ST_LineString  (ARRAY
                      [ NEW ST_Point (75,25),
                        NEW ST_Point (200,25),
                        NEW ST_Point (200,400),
                        NEW ST_Point (25,400),
                        NEW ST_Point (25,75)
    ] ) ] ) ) )
```

Alternative, Using WKT

```
INSERT INTO PARCEL VALUES
   ( 1, 1, ST_CPolyFromText
        ( 'CURVEPOLYGON
            ( COMPOUNDCURVE
                ( CIRCULARSTRING
                    ( 25 75, 39.64 39.64, 75 25 ),
                ( 75 25, 200 25, 200 400, 25 400, 25 75 )
   ))'))
```

ST_GeomCollection



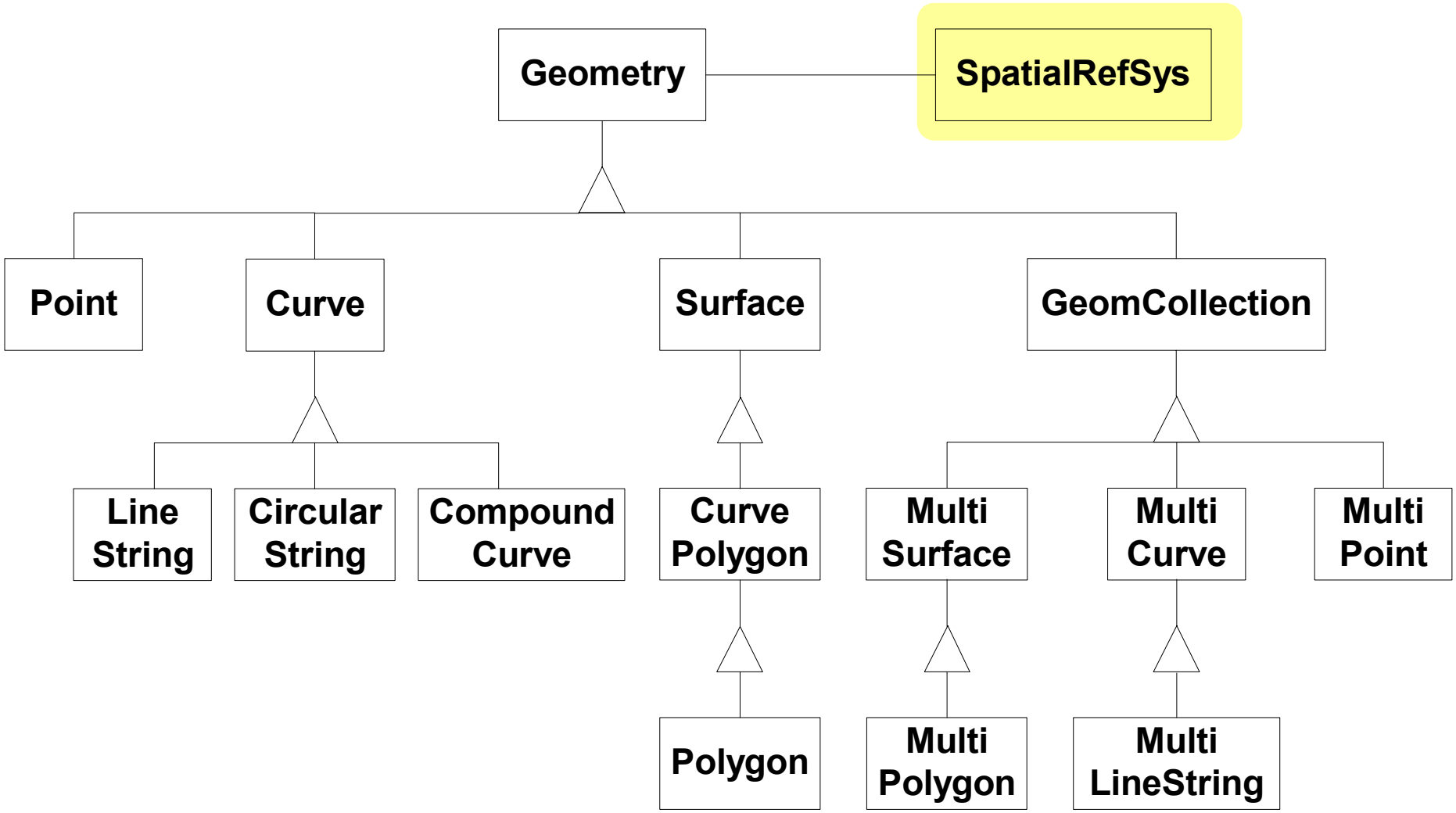
Concepts
ST_GeomCollection

- instantiable subtype of ST_Geometry
- collection of zero or more ST_Geometry values
- all geometries are in the SRS of the collection
- no other member constraints
- subtypes can constrain membership by dimension or overlap

Concepts
ST_GeomCollection Methods

- ST_GeomCollection: returns the specified ST_GeomCollection value.
- ST_Geometries: observes and mutates the ST_Geometry collection
- ST_NumGeometries: returns the cardinality of the ST_Geometry collection
- ST_GeometryN: returns the specified element in the ST_Geometry collection

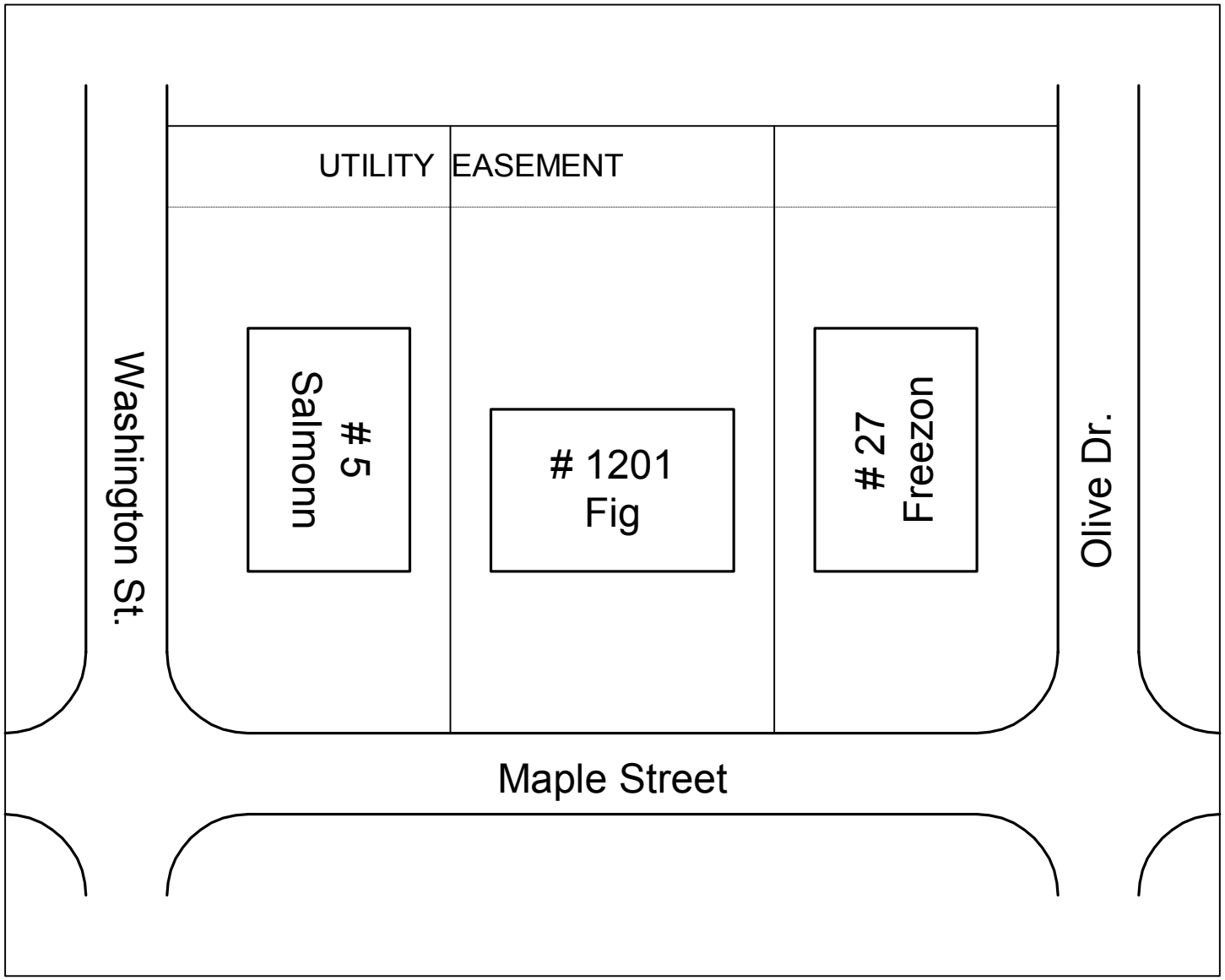
ST_SpatialRefSys



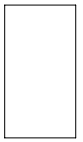
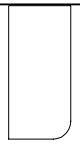
Concepts
ST_SpatialRefSys Methods

- ST_SpatialRefSys: returns the specified ST_SpatialRefSys value
- ST_AsWKTSRS: returns the well-known text representation of a spatial reference system
- ST_WKTSRSToSQL: returns the ST_SpatialRefSys value of a well-known text representation
- ST_SRID: returns the integer identifier of an ST_SpatialRefSys value
- ST_Equals: tests if two ST_SpatialRefSys values are equal

Exercise 5



Exercise 5

NAME	ADDRESS	PHONE	LOCATION
Salmonn	5 Washington St.	555-1234	
Fig	1201 Maple Street		
Freezon	27 Olive Drive	555-5976	

Exercise 5

- Query
  1. How far is it from Salmonn's place to Fig's place?

Exercise 5

- Query
  1. How far is it from Salmonn's place to Fig's place?

```
SELECT F.LOCATION.ST_Distance
  (T.LOCATION)
  FROM PERSON F, PERSON T
  WHERE F.NAME = 'Salmonn'
    AND T.NAME = 'Fig'
```

Exercise 5

- Query
  2. Write the query to return all of Fig's next door neighbors.

Exercise 5

- Query
  2. Write the query to return all of Fig's next door neighbors.

```
SELECT A.NAME
  FROM PERSON A, PERSON B
  WHERE B.NAME = 'Fig'
     AND B.LOCATION.ST_Touches
  (A.LOCATION)
```

Exercise 5

- Query
  3. Write a query to return who has the largest property.

Exercise 5

- Query

  3. Write a query to return who has the largest property.

```
SELECT A.NAME
   FROM PERSON A
   WHERE A.LOCATION.ST_Area =
    (SELECT MAX (B.LOCATION.ST_Area)
          FROM PERSON B)
```

SQL/MM Standard

- Comments / corrections are welcome

  paul.scarponcini@bentley.com
- Work continues on “Later Progression”
- Input / assistance are greatly appreciated