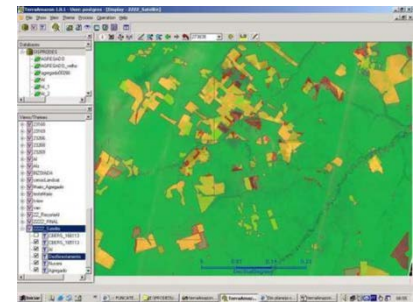
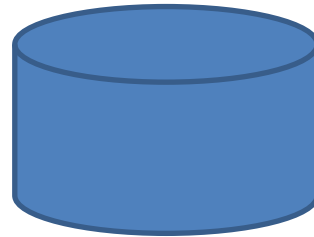
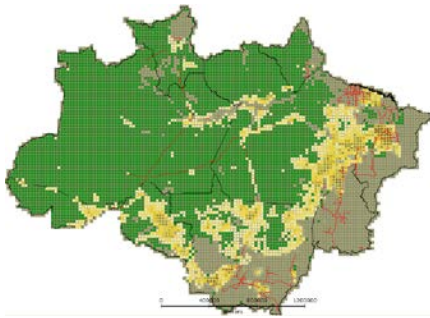


# Spatial Databases: Lecture 6

Institute for Geoinformatics  
Winter Semester 2014



Malumbo Chipofya: room 109

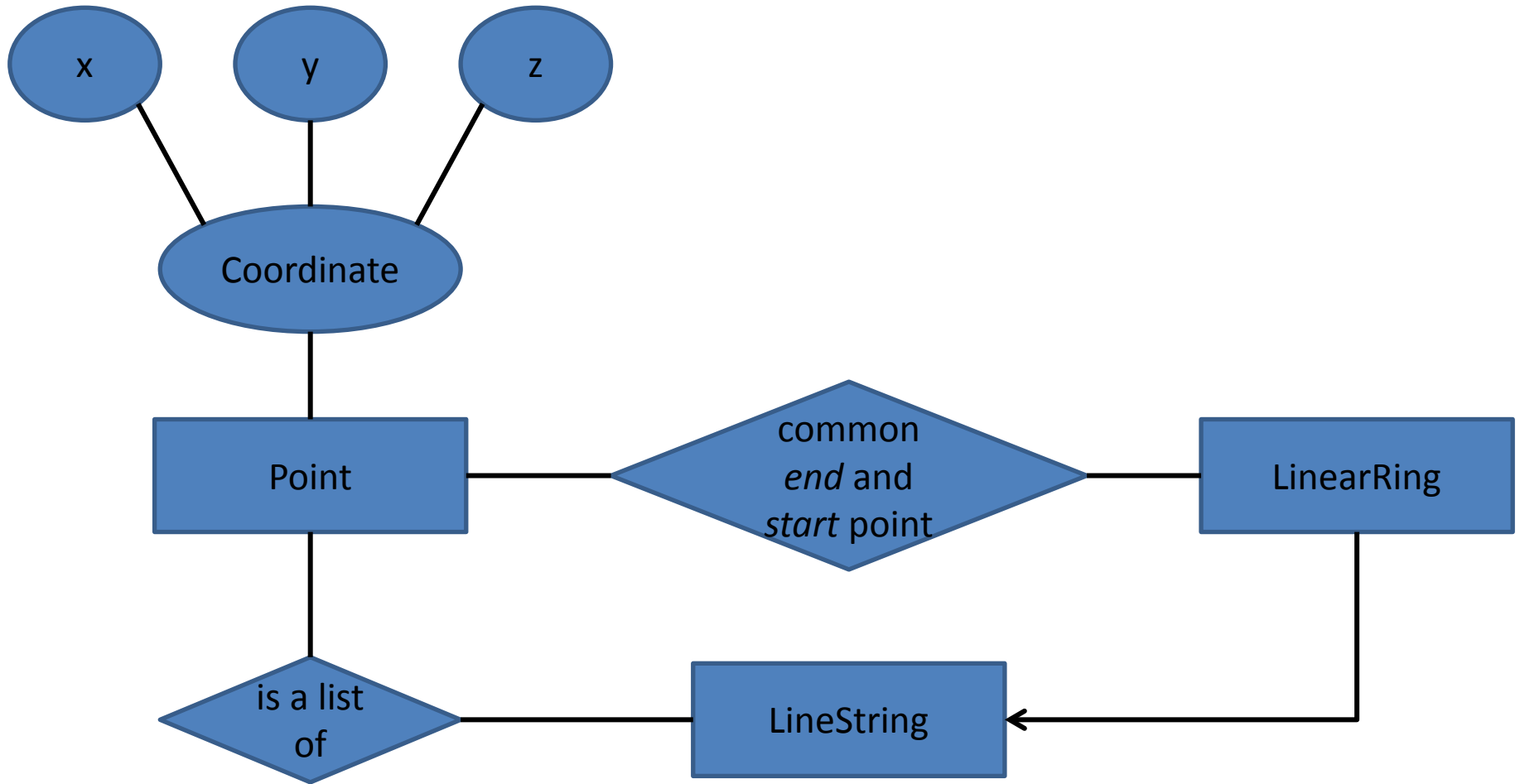
# Topic Overview

1. Prelude: Data and problem solving in science and applications
2. The Relational Database model
3. Interacting with relational databases
- 4. Spatial Relational Database Management Systems**
5. Applications: Terraview and Terralib: Prof. Dr. Gilberto Camara
6. A sample of Nosql Databases: brief introductions + example applications
  - a. Array databases: SciDB
  - b. Document databases: MongoDB
  - c. Graph databases: Neo4J
7. Summary of all lectures given.

# Recap

# Assignment 2: hints

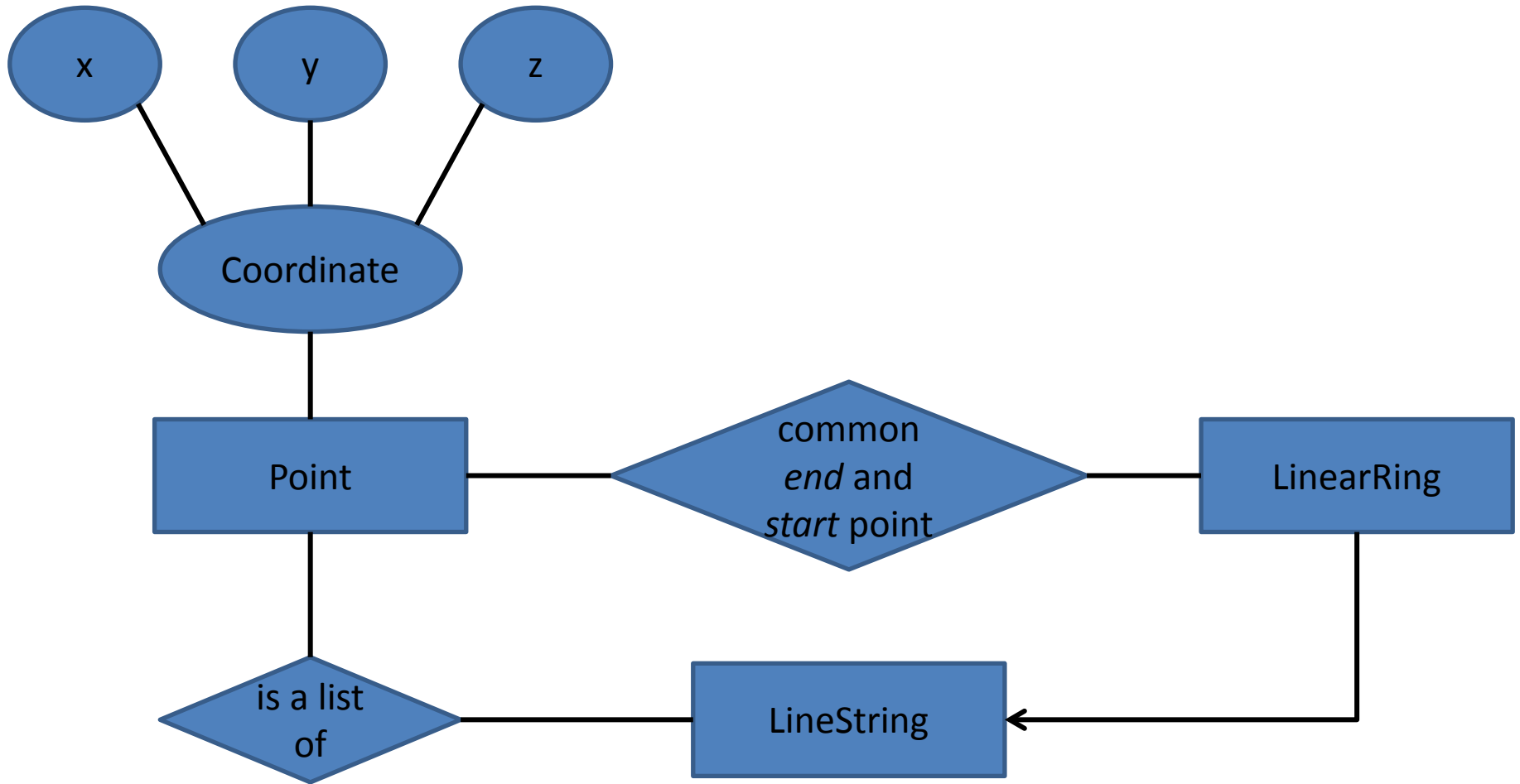
# Creating of Geometry Storage in Postgresql



# Creating of Geometry Storage in Postgresql

- **Create database**
- Create functions
  - make\_world()
  - line\_string\_verbose()
  - get\_vertices()
  - get\_vertices\_recurse()
  - line\_string\_insert()
- Create a view for LineStrings
- Create trigger
- **Test in between**

# Creating of Geometry Storage in Postgresql



# Creating of Geometry Storage in Postgresql

- Create database
- **Create functions**
  - **make\_world()**
  - **line\_string\_verbose()**
  - **get\_vertices()**
  - **get\_vertices\_recurse()**
  - line\_string\_insert()
- Create a view for LineStrings
- Create trigger
- Test in between

# Creating of Geometry Storage in Postgresql

- Create database
- Create functions
  - make\_world()
  - line\_string\_verbose()
  - get\_vertices()
  - get\_vertices\_recurse()
  - **line\_string\_insert()**
- **Create a view for LineStrings**
- **Create a trigger**
- Test in between



# Creating of Geometry Storage in Postgresql

- VIEWS

ID#	Skill	M.St	#Chd	#Yrs	M.€	Date	#sticks	Wgt.	Hrs
1	Medium	M	0	2	40	1.06	55	9	6
2	Low	S	0	1	30	7.05	34	5	5
3	High	S	2	3	45	1.06	54	9	6
4	High	M	3	4	50	3.11	61	12	8

- A view is a tailor made presentation of data in one or more tables or view (WHY VIEWS?)

#Yrs	M.€
2	40
1	30
3	45
4	50

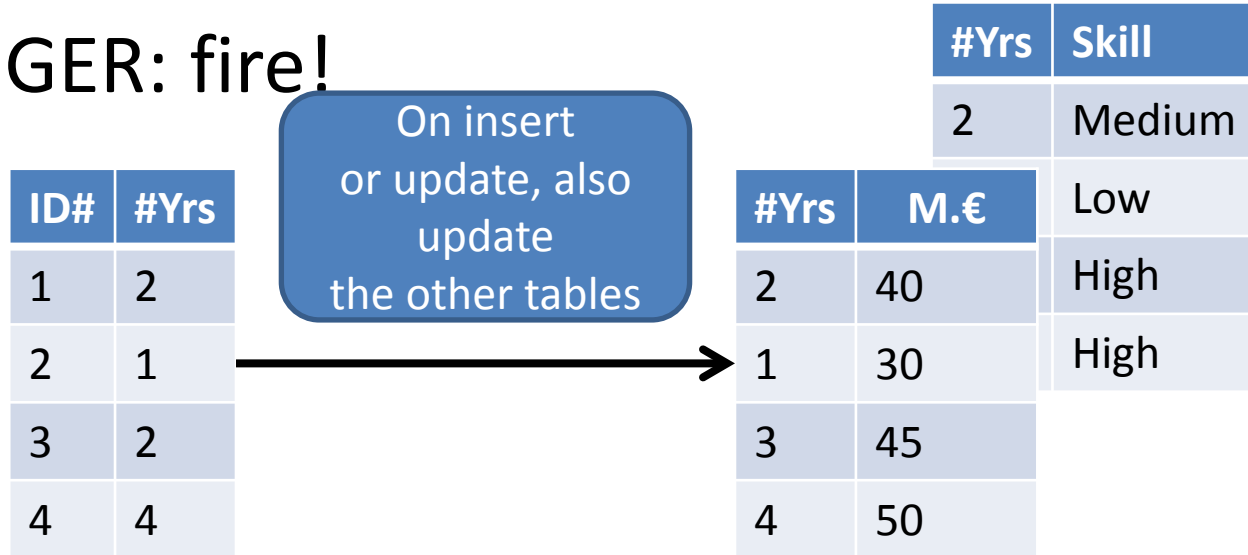
#Yrs	Skill
2	Medium
1	Low
3	High
4	High

ID#	Date	#sticks	Wgt.	Hrs
1	1.06	55	9	6
2	7.05	34	5	5
3	1.06	54	9	6
4	3.11	61	12	8

ID#	#Yrs
1	2
2	1
3	2
4	4

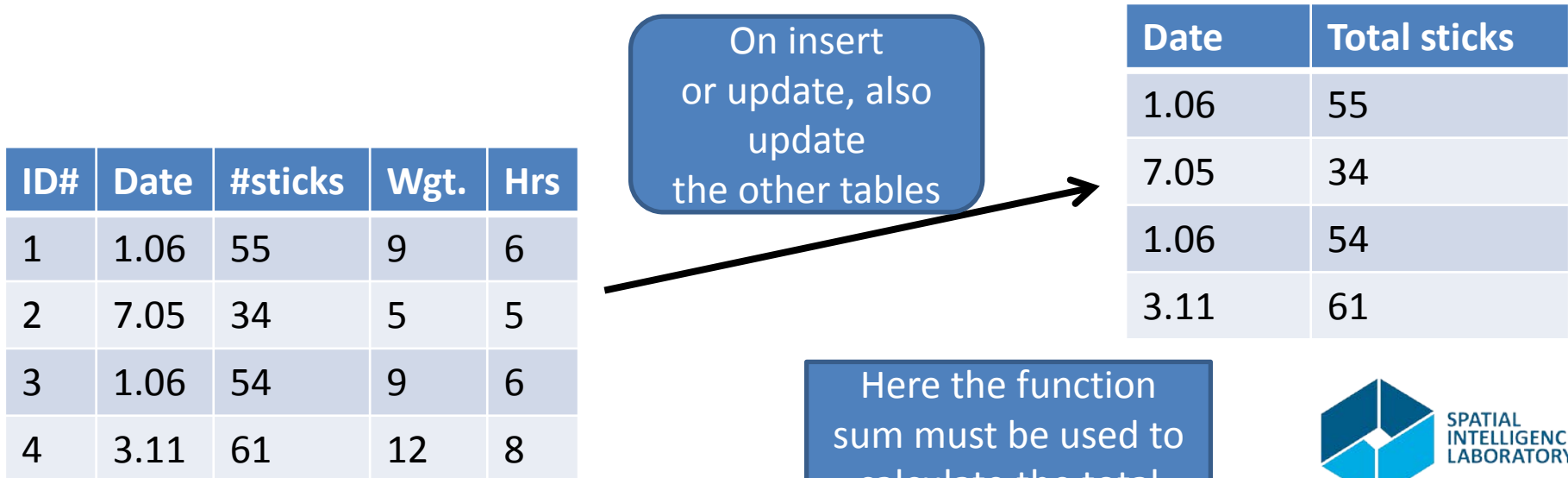
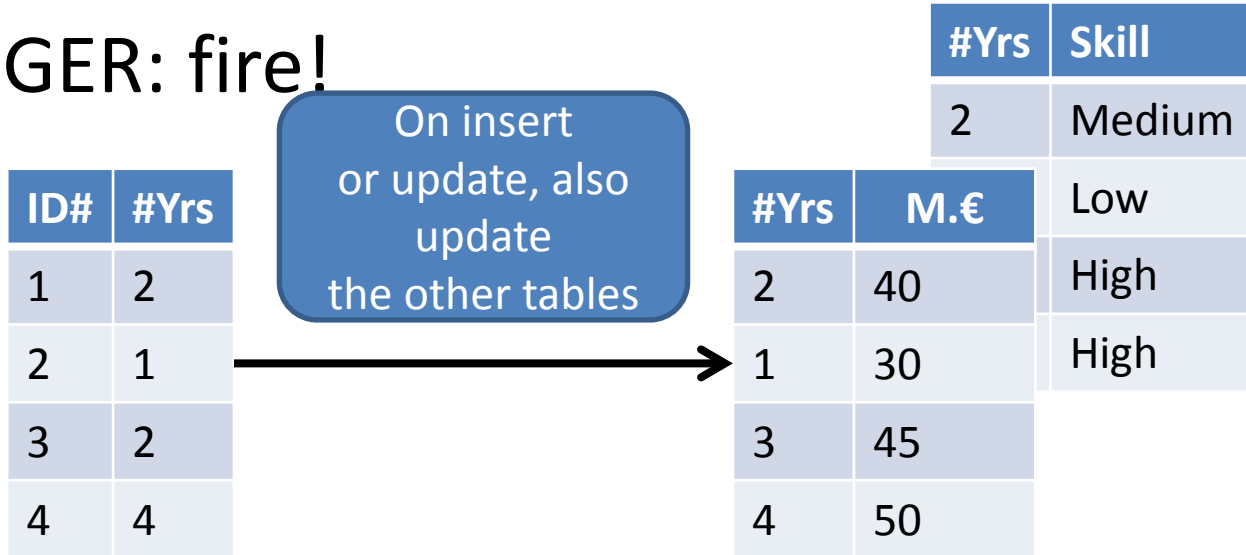
# Creating of Geometry Storage in Postgresql

- TRIGGER: fire!



# Creating of Geometry Storage in Postgresql

- TRIGGER: fire!



Here the function sum must be used to calculate the total

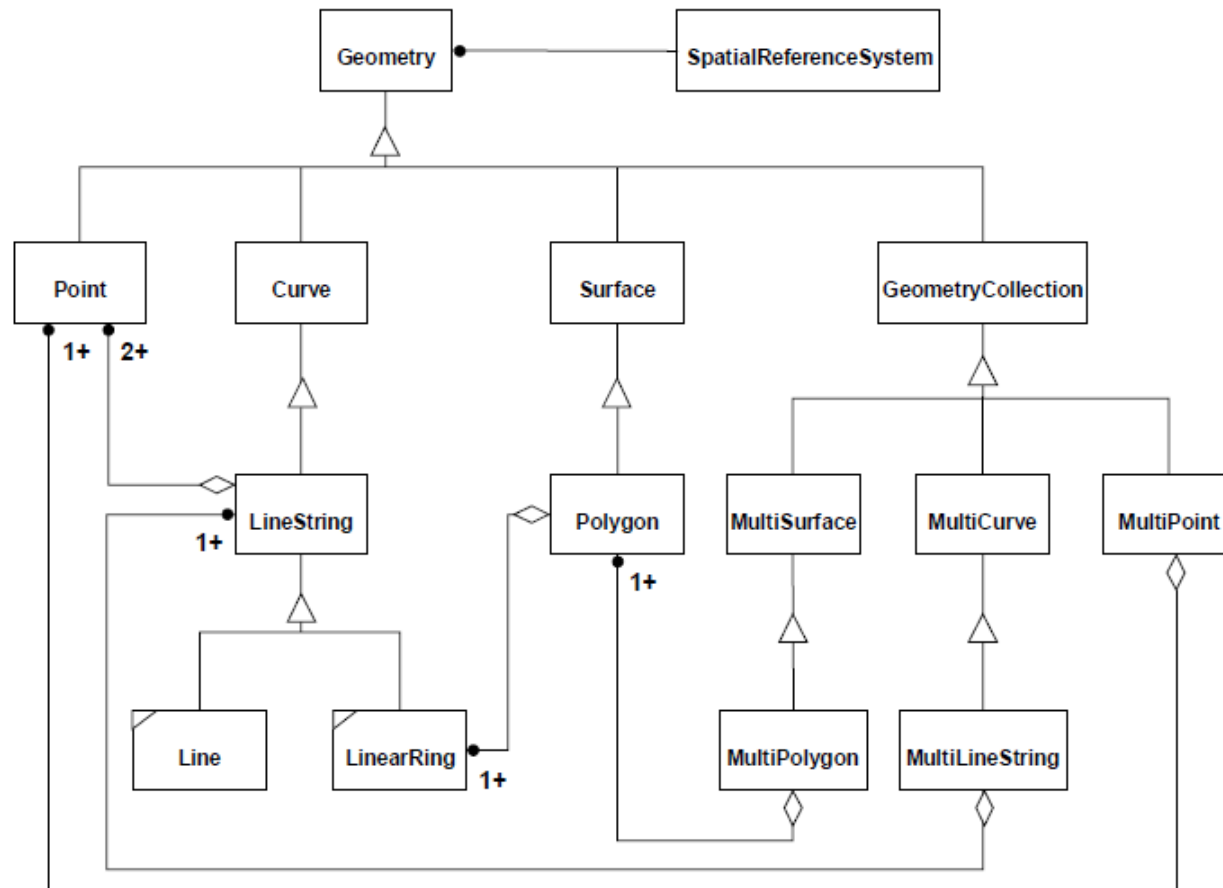
# Creating of Geometry Storage in Postgresql

- Create database
- Create functions
  - make\_world()
  - line\_string\_verbose()
  - get\_vertices()
  - get\_vertices\_recurse()
  - **line\_string\_insert()**
- **Create a view for LineStrings**
- **Create a trigger**
- Test in between

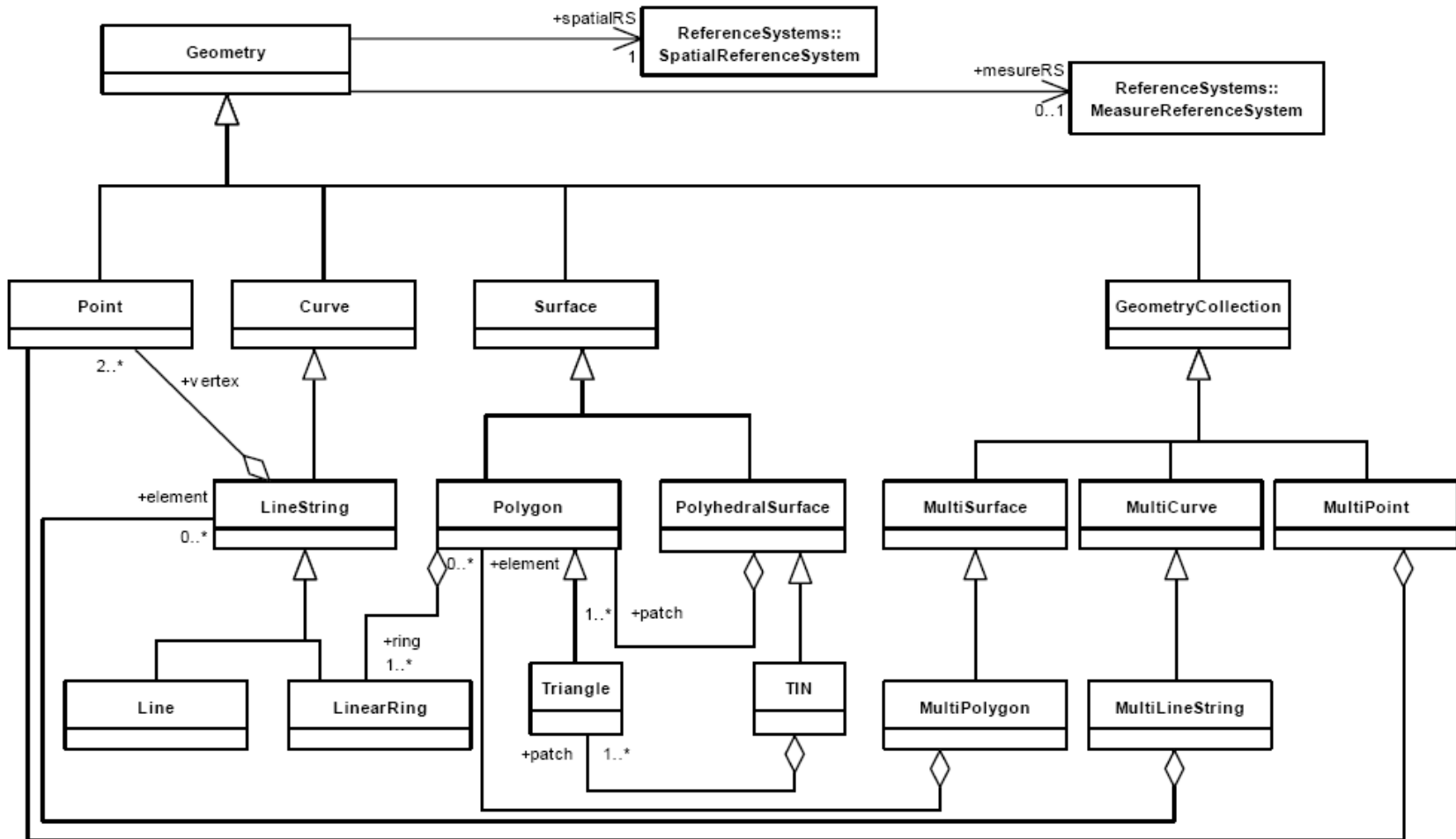
# Spatial Relational Databases

Postgres/Postgis as an example

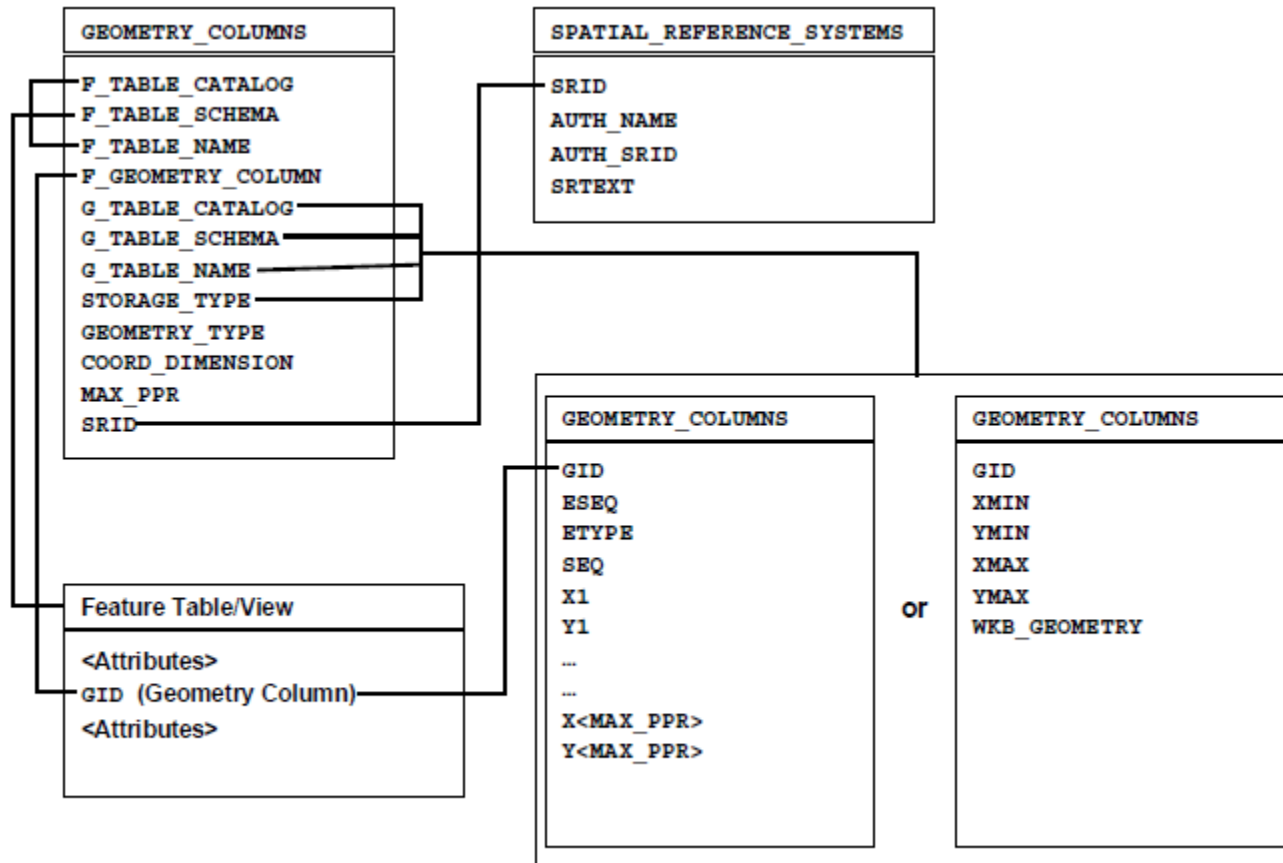
# OGC Simple Features



# OGC *less* Simple Features



# Database Structure





# Creating Geometry Columns

- Start PSQL and login to “mygeoms” database
  - If logged in to the default database postgres, type “\c mygeoms” to change databases
- Install the postgis extension in mygeoms
- In postgis geometries are types so they can be referenced in DDL statements as directly
  - `CREATE TABLE test ( ID int4 , test_label varchar(25), geom geometry(LINESTRING,4326));`

# Creating Geometry Columns

- Add a new table called pgGeomCols with attributes
  - i. gcid: serial
  - ii. label: varchar(80)
- Check the columns you have in your table
  - \d pgGeomCols

# Creating Geometry Columns

- Add a new Geometry columns called poi and poi\_neighborhood as follows
  - SELECT **AddGeometryColumn** ('pggeomcols', 'poi', 4326, 'POINT',2);
  - SELECT **AddGeometryColumn** ('pggeomcols', 'poi\_neighborhood', 4326, 'Polygon',2);
- Now check the columns in your table again
  - \d pgGeomCols

# Creating Geometry Columns

- Geometries are stored in the table `geometry_columns`
  - `SELECT f_geometry_column As cols, type, srid, coord_dimension As dims FROM geometry_columns WHERE f_table_name = 'pggeomcols';`

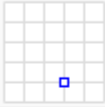
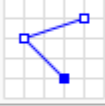
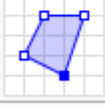

# Creating Geometries

- A geometry column can be of any geometry type.
  - `SELECT AddGeometryColumn ('pggeomcols', 'poi_full_neighborhood', 4326, 'Geometry',2);`

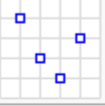
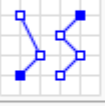
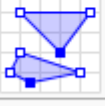

# Referring to Geometries

- Well Known Text format

Geometry primitives (2D)

Type	Examples	
Point		<code>POINT (30 10)</code>
LineString		<code>LINESTRING (30 10, 10 30, 40 40)</code>
Polygon		<code>POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))</code>
		<code>POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))</code>

Multipart geometries (2D)

Type	Examples	
MultiPoint		<code>MULTIPOINT ((10 40), (40 30), (20 20), (30 10))</code>
		<code>MULTIPOINT (10 40, 40 30, 20 20, 30 10)</code>
MultiLineString		<code>MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))</code>
MultiPolygon		<code>MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))</code>
		<code>MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))</code>

# Referring to Geometries

```
INSERT INTO pgGeomCols (label, poi, poi_neighborhood,  
poi_full_neighborhood) VALUES  
(  
'Point3', 'POINT(0.5 0.5)', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))',  
'GEOMETRYCOLLECTION(POINT(0 0), POLYGON((0 0, 1 0, 1 1, 0 1, 0  
0)))'),  
(  
'Point2', 'POINT(2 0)', 'POLYGON((0 0, 3 0, 3 3, 0 3, 0 0))',  
'GEOMETRYCOLLECTION(POINT(2 0), POLYGON((0 0, 1 0, 1 1, 0 1, 0  
0)))'),  
(  
'PolygonWithHole', 'POINT(0.5 1)', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0  
0),(1 1, 1 2, 2 2, 2 1, 1 1))', 'GEOMETRYCOLLECTION(POINT(0.5 1),  
POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1)),  
LINESTRING(0 0, 1 1, 2 1, 2 2))');
```

```
SELECT label, poi_full_neighborhood FROM pgGeomCols;
```

```
SELECT label, ST_AsText(poi_full_neighborhood) FROM pgGeomCols;
```

# PostGIS functions

<http://postgis.net/docs/reference.html>

- PostgreSQL PostGIS Geometry/Geography/Box Types
- Management Functions
- Geometry Constructors
- Geometry Accessors
- Geometry Editors
- Geometry Outputs
- Operators
- Spatial Relationships and Measurements
- SFCGAL Functions
- Geometry Processing
- Linear Referencing
- Long Transactions Support
- Miscellaneous Functions
- Exceptional Functions



# References

- <http://postgis.net/docs/>
- <http://www.postgresql.org/docs/9.3/static/index.html>
- <http://workshops.boundlessgeo.com/postgis-intro/index.html>
- <http://www.opengeospatial.org/standards/sfa>  
(OGC SFS - Common architecture)
- <http://www.opengeospatial.org/standards/sfs>  
(OGC SFS - SQL option)

That's all for today

Thank you!

Questions?