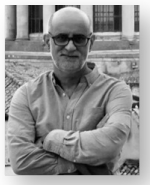
Spatial Data Management for Analytics

Joseph G Vella, Faculty of ICT, UM
JOSEPH.G.VELLA@UM.EDU.MT
L-Università ta' Malta

NextTech 2025
SoftNet 2025
International Academy, Research, and Industry Association
Lisbon, September 2025

IARIA

About



Prof Joseph G Vella is currently an academic member at the University of Malta, emphasising teaching and researching data modelling and DBMS-related technologies. Prof Vella has introduced many teaching units for undergraduate and postgraduate curricula (e.g. Data mining and Warehousing and Digital Forensics). During the same time, Joseph supervised scores of undergraduate projects and a growing number of postgraduate students, authored numerous refereed papers, edited manuscripts, and is asked to review conference and journal submissions.

Prof Vella is involved in several research projects with other academic institutions and industries. The main role in these projects is to apply his expertise in data management to integrate data better and address data quality issues. Another contribution is to advise companies to ensure adequate performance for data-intensive applications running on data servers that are either on-premises or cloud-based.

Research Interest Areas

- Computer Information Systems
- Data and Query Modelling
- DBMS Set-up and Tuning
- Digital Forensics
- Data Science and Analytics

2

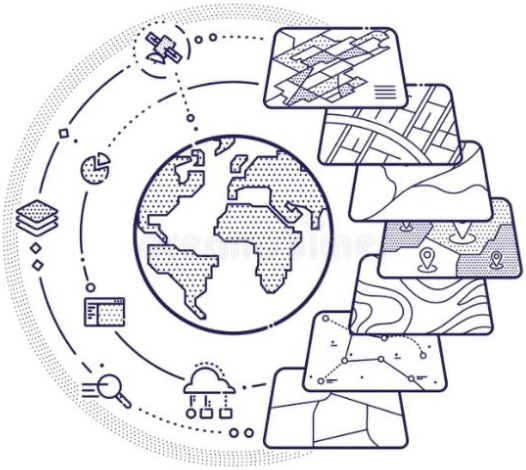
Spatial Data Management for Analytics
Tutorial's Structure

Agenda

- Spatial Data
  - Vector, raster
  - Spatial relationships
- Spatial Operations
  - Create
  - Relate
  - Transform
- Spatial Query Examples
  - Examples
- Spatial Data Analytics
  - Examples
- Wrap-up and Take-Home

3

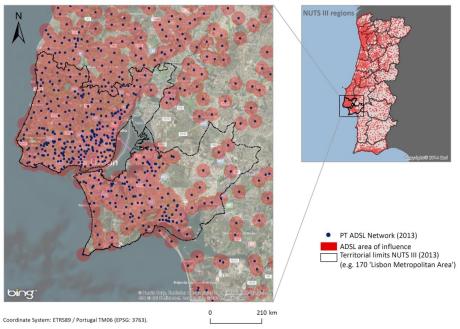
Start Pitch



Case study – from the host country

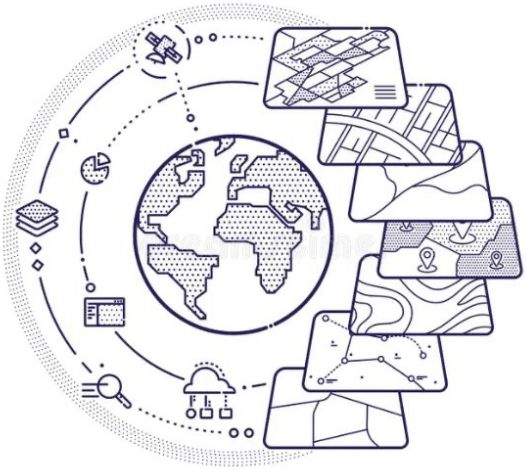
verbatim copy from:
www.efgs.info/information-base/case-study/analyses/potential-territorial-coverage-broadband-internet-access-pt/

- "Statistics Portugal produces a **broadband internet coverage indicator** as a result from the need to include this dimension in the competitiveness index of the **Regional Development Composite Index** at NUTS III level. "
  - "The main goal of Regional Development Composite Index is to provide results that allow for the monitoring of regional disparities and that **support the definition and evaluation of territorial based public policies**."
- "The broadband internet access indicator was created based on **spatial data information** on points of distribution of the broadband internet access network made available by Portugal Telecom."
- "The combination of these buffer zones with the Administrative Map of Portugal and by using Geospatial Analysis techniques, made it possible to obtain the broadband internet access areas of influence associated to each NUTS III region "



5

Spatial Modelling and Data



Spatial modelling deals with defining the structure of spatial and non-spatial data and their relationships (including spatial relationships).

In Biological & Geographical systems, spatial data is layered over the Earth.

Spatial Data

Spatial Data is data associated to a *location in space* (including multi dimensional space – ***vector data***) or about *space* itself (e.g., ***raster image***).

An entity instance can be composed both of non-spatial data (descriptive or categorical properties - what we are accustomed to) and spatial data.

- We couch spatial data in terms of:

  a spatial reference system - the end user's perspective (location, speed).

  And a *co-ordinate system* – on which the spatial data is laid.

  - There are a number of these – e.g., *Euclidean*, *polar*

- *Geospatial data* is spatial data tied to our globe's surface.

  Therefore, geospatial data is spatial data;

  But not all spatial data is geospatial data!

  Also, with geospatial data we have several traditional co-ordinate systems – most coming from **cartography** (the making of maps).

  Furthermore, some basic, but hard, decisions must be made on how to project data and approximate the Earth's surface.

7

Examples of Spatial Data

- Census and surveys data sets
- Satellites imagery - terabytes of data per day
- Water pathways (rivers), farms, ecological sites of importance
- Technical plans of machines, buildings
- Weather and Climate Data
- Medical imaging
- Bodies for animation

8

Spatial Relationships

A **spatial relationship** specifies spatial artefact location in space in relation to another artefact / reference spatial data.

Examples of:

- A spatial object in space is related to other objects by:
  - **vicinity**
    - Metric: which telephone booth is closest to a specific road segment?
  - **coverage**
    - Topological: disjoint, equal, meet, overlap, contains, inside, covers, covered-by are examples
  - **directional**
    - NEWS, on top, topmost, lower, lowest
  - **connectedness**
    - Graph / network model for shortest path
  - **build-up**
    - A spatial object is ***composed*** of other spatial objects - But the higher-level object is "the" object.
    - A spatial object is an ***aggregation*** of other objects - The higher object is the object but:
      it could be dismantled into other objects;
      it could "share" several spatial objects

Next slide

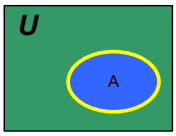
9

Spatial Relationships – Coverage Semantics (*)

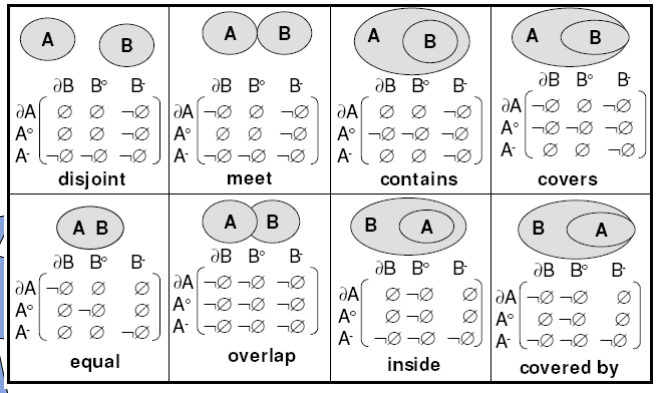
- Simple coverage spatial relationships between objects (no holes and connected 2D objects)

U is the universe (shaded in green).
'**A**' is the shape (e.g. the ellipse interior shaded in blue). Shape A's boundary is drawn in yellow.

A° is the interior;
∂ A is the boundary; and
A⁻ is the exterior in U.



In terms of boundary, interior and exterior!

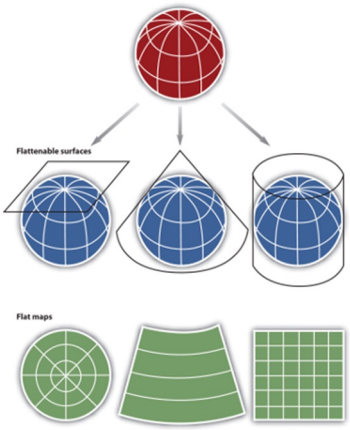


(*) MJ Egenhofer, RD Franzosa, "Point-set topological spatial relations", International Journal of Geographical Information System 5 (2), 1991, 161-174

(*) MJ Egenhofer, "Categorizing binary topological relationship between regions, lines, and points in geographic database", Technical Report, Department of Surveying Engineering, University of Maine, 1991

10

Assigning a projection and coordinate system onto spatial data



We need to ensure two dataset have a common projection for comparison!

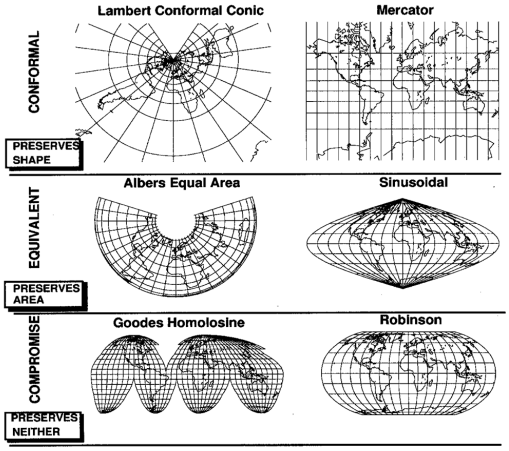


Figure 2.9 Examples of projections classified by their distortions. Conformal projections preserve local shape, equivalent projections preserve area, while compromise projections lie between the two. No projection can be equivalent and conformal.

For GPS data use of WGS 84 SRID 3326 is indicated.

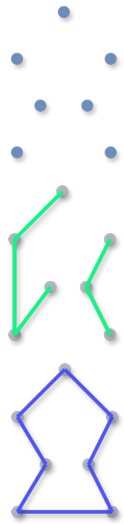
11

Spatial Data - Vector types

- **Points**, **lines**, and **polygons**
  - 2D and 3D
- Common Representations:
  - Well-Known Text (**WKT**) (and Well-Known Binary (WKB))
  - **Shape Files** (i.e., ESRI)
  - GeoJSON
  - GeoPACKAGE
  - **GPX** (i.e., Garmin)
  - KML (i.e., Keyhole Markup Language)
  - MVT (Mapbox Vector Tiles)

12

2D Vector Data Types (structure)

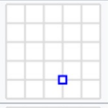
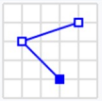
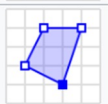
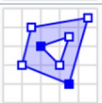


- Point (x, y)
  Example: Restaurants abstraction
  Aka node, vertex
  Can have non-spatial attributes (e.g., colour)
- Line – a sequence of points
  Example: pathway abstraction
  Aka linestring, chain
  Two consecutive points are connected by (straight) edge
  Can have non-spatial attributes
  Note:
  - Edges can cross in complex linestrings
  - Edges can form a closed loop in complex linestrings
- Polygon – a closed loop of points enclosing an area
  Example: recreational area abstraction
  Two consecutive points are connected by (straight) edge
  Can have non-spatial attributes
  Note:
  - A polygon can have one, or more, holes in complex polygons

13

Well-Known Text (WKT) Representation of Vector Data Types

Geometry primitives (2D)

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

From Wikipedia: https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

These structures are based on the **Open Geospatial Consortium** (OGC) - HTTPS://WWW.OGC.ORG/.

Well-known binary is a systematic conversion/encoding of WKT.

14

Well-Known Text (WKT) Representation of Vector Data Types - Complex

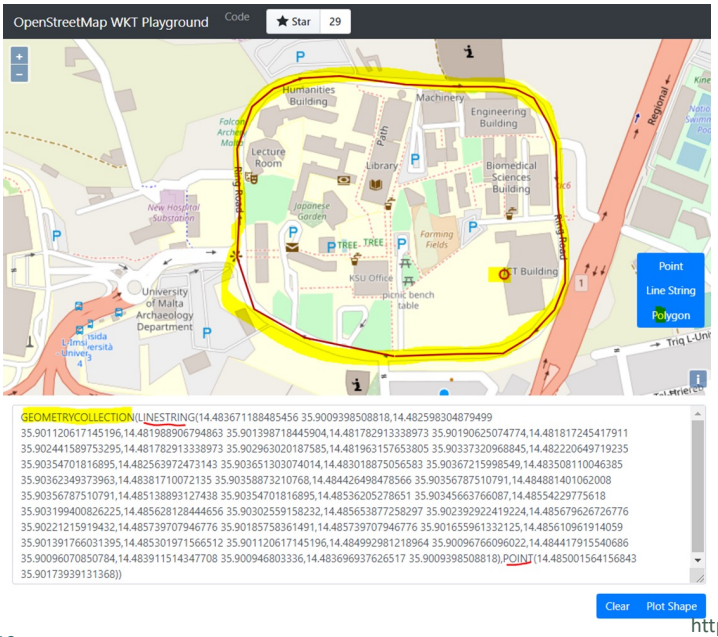
Multipart geometries (2D)

Type	Examples
MultiPoint	 `MULTIPOINT ((10 40), (40 30), (20 20), (30 10))` `MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString	 `MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon	 `MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`  `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)))`
GeometryCollection	 `GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`

From Wikipedia: https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

15

WKT Playground (many available)



https://clydedacruz.github.io/openstreetmap-wkt-playground/

16

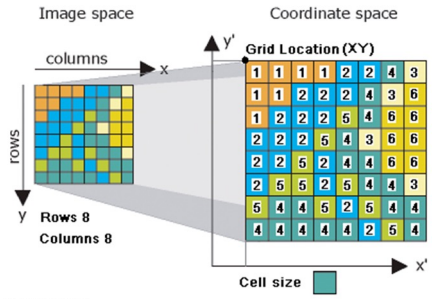
Spatial Data Type - Raster Data

- Single **band/layer** (sometimes *dimension* is used instead of band)
  - One datum vs multiple datums
- Multi band/layer
  - One datum vs multiple datums
- Common representations
  - **Tiff**, JPEG, etc.
  - **GeoTiFF**
  - **Network Common Data Form** (NetCDF)
    - Self describing
    - With built in timeseries encoding
- Sources of Raster Data include:
  - *Satellite imagery*
  - *Aerial photography*
  - *Processed data* (e.g., from satellite data collection, to weather projections)
  - *Vector data composition mapped into a raster*

17

Raster Data Files

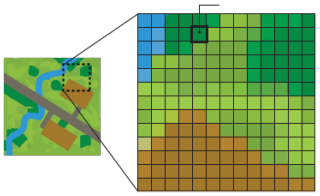
- Raster files (or Raster formats) are used for storing dense data over all its coverage.
  - *This technique is helpful when a vector representation is inadequate and unavailable.*
- An example of a raster is denoting an area's mixed usage – e.g., composition of farmlands, roads, built structures, natural boundaries (rivers).
- Rasters are also used as background over which spatial data is superimposed – e.g., OpenStreetMap.
- A raster file is a grid representation, having:
  - Rows (r) and columns (c) with a pixel/cell in each.
    - Pixel/cell have a size (i.e., area) – this is related to the file's **resolution**.
    - Values are assigned to a pixel
      - **single**, or **multi band**
    - A raster file can have multiple **layers**!
- The dataset is **georeferenced**, e.g., pinned to the Earth's surface (usually the first pixel, plus a skew angle).
- Note the row and column counting practice
  - i.e., the first pixel is the upper left corner
- Size of a raster is: [x] * [y] * Dutum size * layers



List of cell values

[1111224311222436122254662225436622524466255254435445254444444254]

https://desktop.arcgis.com/en/arcmap/latest/manage-data/geodatabases/raster-basics.htm

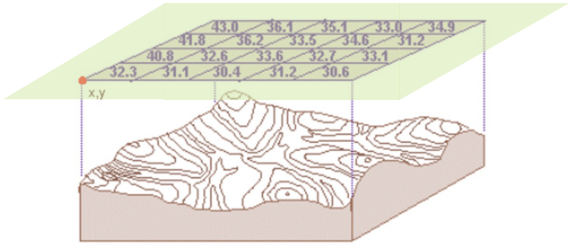


https://desktop.arcgis.com/en/arcmap/latest/manage-data/geodatabases/raster-basics.htm

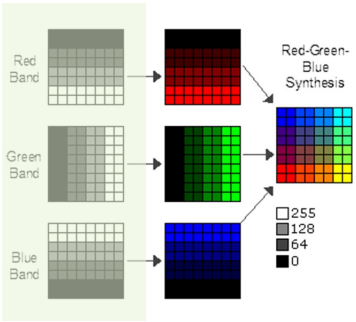
18

Raster Data (continued)

Single band
Discretization of a landscape showing elevation



Multi band
Discretization of colour sensing in 3 channels (RGB)



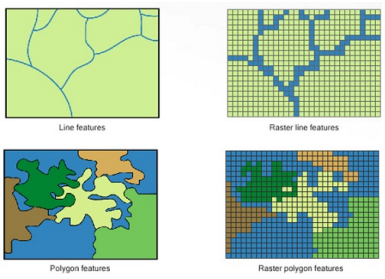
19

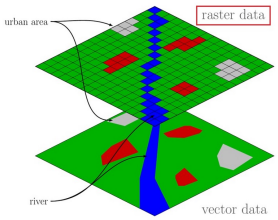
Raster Data Operations

- **Raster Management**
  - Calculate the area
  - Pixel type, Pixel size
  - Georeference position, Spatial reference system

  Raster Conversion
  - toTIFF, toJPEG, toPNG
  - fromVector
    - ! This is not a straightforward conversion: i.e.,
      - Overlaps are lost, resolution differences, one raster per vector type
  - toVector

https://gsp.humboldt.edu/olm/Lessons/GIS/08%20Rasters/RasterToVector.html





20

https://cartong.pages.gitlab.cartong.org/learning-corner/en/3_key_gis_concepts/3_3_key_concepts/3_3_3_vector_raster_data

Spatial Operations

On vector data

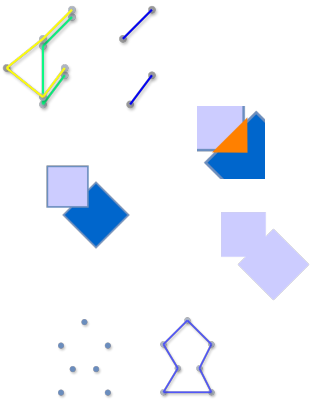
- Examples
  - Retrieve artefact directly
  - Aggregate
    - Total
    - Group by
    - Windowing
  - Retrieve through spatial relationships
  - Generate
    - Through spatial operations
    - Through aggregations (pts -> line string)
    - Through data reduction (line string -> line strings)

22

2D Vector Data Types (Operations)

- Point
  - **Distance** between two points -> number
  - **Extract** a coordinate (e.g., latitude) -> number
  - **Move** a point -> point
- Line
  - **Length** of a line -> number
  - **Closest distance** between two lines -> number
    - Closest distance between a line and a point
  - **Intersection** of two lines -> null | point+ | line+
  - **Combine** two lines -> line
  - **Build a line** from a sequence of points -> line
- Polygon
  - **Area** of a polygon -> number
  - **Centre** of the polygon -> point
  - **Intersection** of two polygons -> null | point | line+ | polygon+
  - **Combine** two polygons -> polygon
  - **Build a polygon** from a set of points -> polygon

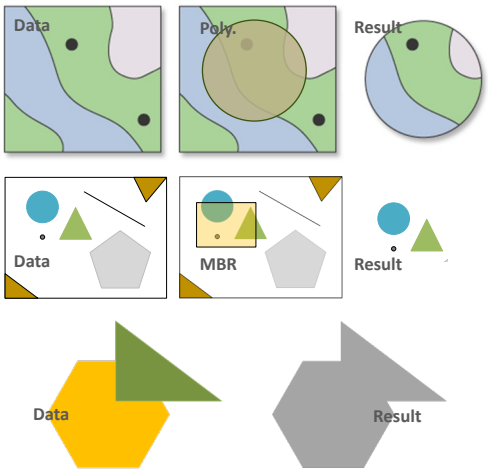
Note: that from simple structures, e.g. lines, an operation can yield a "complex structure" – set of lines. Yellow and green line intersection produces two lines (the blue).



23

2D Vector Data Types (Operations - continued)

- **Selection**
  Which objects are at Point (x,y)?
  Returns all objects: point+ | line+ | polygon+
- **Clipping**
  Which objects in *Most Bounding Box* (MBR) or polygon?
  Returns an object or part-of (delimited by MBR) that overlap and MBR.
- **Windowing**
  Which objects in MBR or polygon?
  Returns all object that overlap with MBR (i.e. output area could be greater than MBR's.
- **Merger of objects**
- **Projection**
  Redraw artefact with an ad hoc style



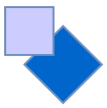
24

2D Vector Data Types (Operations - Predicates)

- **Spatial predicates**, i.e., returns true if it is the case that a spatial relationships holds between two static spatial objects.

Examples of spatial predicates include:

Intersection (object, object) -> true | false



intersection(poly lilac, poly blue) -> true

Inside (object, object) -> true | false

inside(point blue, poly lilac) -> true

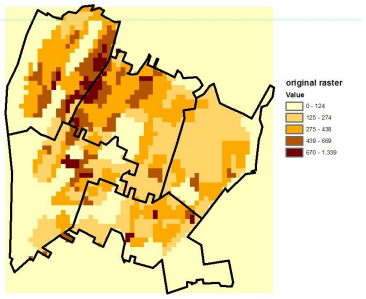
Disjoint(object, object) -> true | false

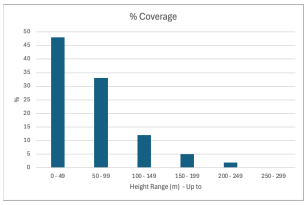
disjoint(line blue, line orange) -> false

25

Raster Data Operations (overview)

- **Raster Transformation**
  - Set a pixel value
  - Translate
  - Resample
  - Clip
    - Vector and Raster interaction
      - Assigned a value to the vector points from the raster
      - Clip a vector polygon and the raster and apply processing on the clip
  - Apply a function (MapAlgebra)
- **Rater Processing**
  - Summary Statistics
  - Histogram
  - Value count
- **Raster Composition**
  - Coalesce multi bands





26

Spatial Database and DBMS

Spatial Database

- A **spatial database** supplants a database by *modelling* and *managing* of **spatial representations of artefacts** belonging to a space, anchored by a coordinate reference system, and providing *query primitives* for accessing and evaluating all data.
- Many spatial databases enable the creation of more intricate spatial artefacts using basic spatial data, such as *points*, *lines*, and *polygons*, and spatial relationships, such as *overlap* and *inside*.
- To store and modify geographic data, particularly in Geographic Information Systems (GIS), a geographic database, that contains georeferenced spatial data and relationships, is used.

Standardisation:

The *Simple Features specification* was created by the **Open Geospatial Consortium (OGC)** and was initially published in 1997.

- It establishes guidelines for integrating spatial capabilities into database systems.

The SQL and multimedia standard extends the Simple Features is the **SQL/MM Spatial ISO/IEC standard**.

28

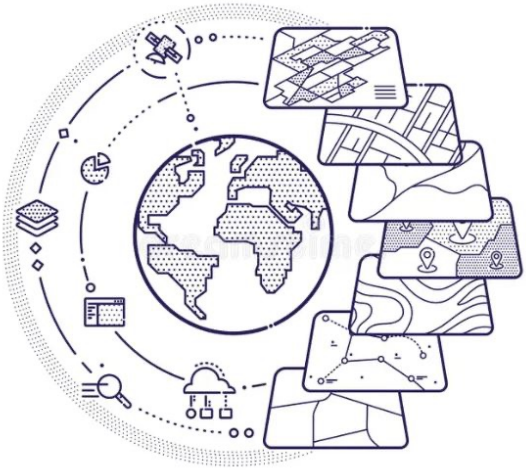
Spatially Enabled DBMS

- What, when, and where can spatial DBMS help?
- Modelling spatial structures, relationships, and constraints
  - With implicit support to spatial data types
- Adopt and follow known standards and best practices.
- Offer efficient solutions to spatial data storage, indexing and retrieval
  - We need specialised multi-dimensional indexing techniques. A common example is Guttman's R-Tree (*).
  - PostgreSQL has a very good variety of these.
- "Joining" data on spatial attributes
- Requires handing of models that are not strictly geographical in nature

(*) A. Guttman. 1984. R-trees: a dynamic index structure for spatial searching. SIGMOD Rec. 14, 2 (June 1984), 47–57.

29

Spatial Queries



Spatial Retrieval (Guade, 1998*)

Exact – return objects exactly matching given artefact
Point – return objects overlapping given point artefact
Window – return objects overlapping given polygon artefact
Intersection – return objects overlapping at least one point
Enclosure – return objects that overlap all the given polygon's points
- Dual of *containment*

Containment – return objects whose all points overlap given polygon's points
- Dual of *enclosure*

Adjacency – return objects that do not have interior overlap but have boundary interaction with a given polygon
Nearest-Neighbour – given an artefact, find objects that are closest (minimum distance).
Spatial Join – given a predicate with a spatial conjunct, return all object that satisfy this predicate.

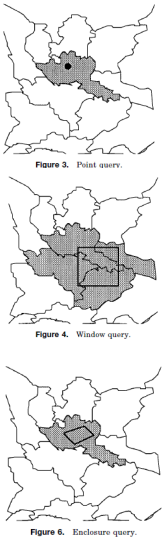


Figure 3. Point query.
Figure 4. Window query.
Figure 6. Enclosure query.

(*) V Gaede and O Günther. 1998. Multidimensional access methods. ACM Computer Surveys 30, 2 (June 1998), 170–231.

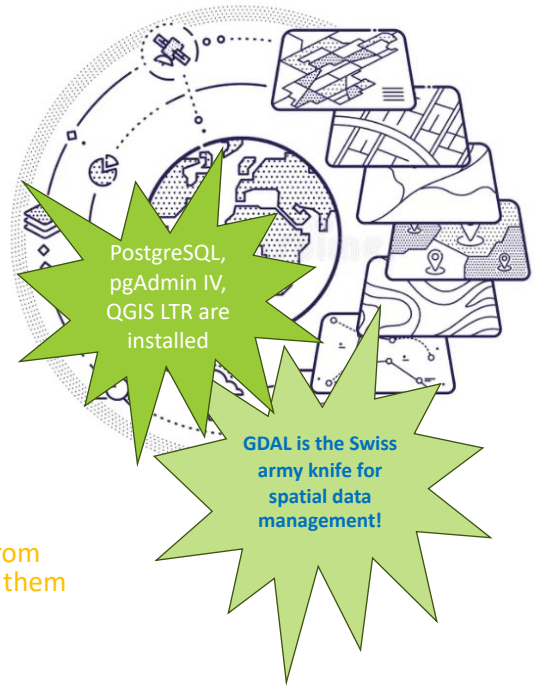
31

Spatial Queries: Examples

Based on PostgreSQL + PostGIS extension

- Vector (Geometry and Geography)
- Raster
- Mixed

All functions and predicates starting with 'ST_' are from the OGS standard – i.e., any langauge implementing them must implement the standard's semantics!



Setup – Database, Schema, Tables

```
Create the Database through pgAdmin IV or an SQL shell (e.g., psql)
   Give it any name, mine is spatial_jgv, and set owner as postgres

-- Enable the postgis extension:
    CREATE EXTENSION postgis;

-- Create Schema called geometry
    CREATE SCHEMA geometry AUTHORIZATION postgres;

-- Create table called geometries (this will be an abstract table)
-- (an abstract table is a table that is not meant to have rows)
    CREATE TABLE geometry.geometries(
       geo_serno bigserial primary key,
       geo_test character varying (255),
       geo_obj geometry not null,
       geo_weight numeric);
```

Opt for **geography** type if distances are important!

33

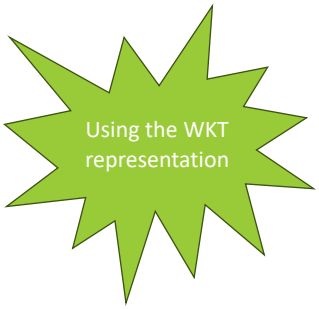
Setup – Database, Schema, Tables (continued)

```
-- Create a table for a few types of geomerties - point, line, polygon.  Each will inherit from
   abstract table geometry.geometries.

   CREATE TABLE geometry.pt() INHERITS (geometry.geometries);
   ALTER TABLE geometry.pt
     ADD CONSTRAINT geometry_pt_point CHECK (ST_GeometryType(geo_obj)='ST_Point');  -- Point
   Geometry

   CREATE TABLE geometry.ln() INHERITS (geometry.geometries);
   ALTER TABLE geometry.ln
        ADD CONSTRAINT geometry_ln_line CHECK (ST_GeometryType(geo_obj)='ST_LineString'); -- Linestring
   Geometry

   CREATE TABLE geometry.pg() INHERITS (geometry.geometries);
   ALTER TABLE geometry.pg
        ADD CONSTRAINT geometry_ln_line CHECK (ST_GeometryType(geo_obj)='ST_Polygon'); -- Polygon Geometry

   CREATE TABLE geometry.gc() INHERITS (geometry.geometries); -- Vector collection Geometry
   ALTER TABLE geometry.gc
        ADD CONSTRAINT geometry_ln_line CHECK (ST_GeometryType(geo_obj)='ST_GeometryCollection');
```

34

Setup – Populate database and tables

```
-- The following is test data (email me for the SQL script, if interested):
-- Remember, table geometry.geometries will not be our target for inserts as it is an
   abstract table (not meant to store rows)!

INSERT INTO geometry.pt(geo_test,geo_obj,geo_weight)
VALUES ('pt_a','POINT( 0 0 )',0);

INSERT INTO geometry.ln(geo_test,geo_obj,geo_weight)
VALUES ('ln_a','LINESTRING ( 1  1, 10 10  )',1);

INSERT INTO geometry.pg(geo_test,geo_obj,geo_weight)
VALUES ('po_a','POLYGON(( 1 1, 3 1, 3 3, 1 3, 1 1  ))',10),
```



35

Performance enchancement

- One definite imnprovement is spatial indexing
  i.e., indexes of multidimensional data are a breed of their own
  Most popular being R-tree (*)
- Spatial indexing is not a silver bullet for performance!

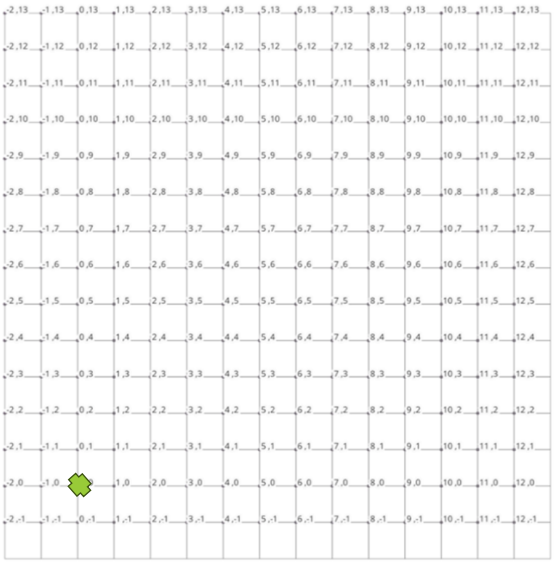
```
CREATE INDEX idex_geom_pt_obj
ON geometry.ptx
USING gist(geo_obj);

VACUUM ANALYZE geometry.ptx;
```

(*) A. Guttman. 1984. R-trees: a dynamic index structure for spatial searching. SIGMOD Rec. 14, 2 (June 1984), 47–57.

36

Visualization of test data coverage

Grid (of interest):
Lower left corner (longitude, latitude)

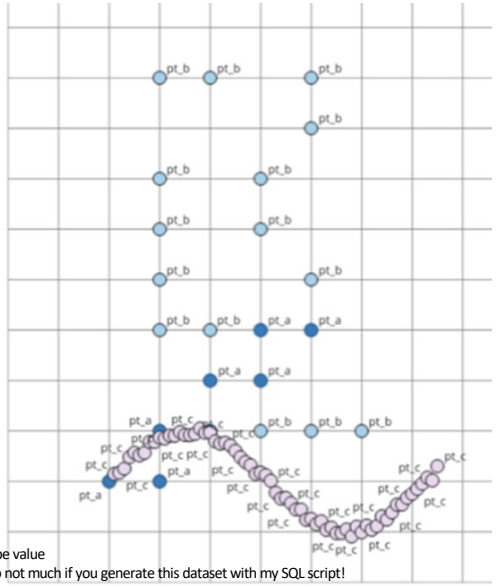


37

Visualizations

Points:
Colour and label by attribute geo_test.

```
-- refresh
CREATE TABLE geometry.geometries(
       geo_serno bigserial primary key,
       geo_test character varying (255),
       geo_obj geometry not null,
       geo_weight numeric);
CREATE TABLE geometry.pt() INHERITS
   (geometry.geometries);
   ALTER TABLE geometry.pt
     ADD CONSTRAINT geometry_pt_point
     CHECK (ST_GeometryType(geo_obj)='ST_Point');
INSERT INTO
   geometry.pt(geo_test,geo_obj,geo_weight)
VALUES ('pt_a','POINT( 0 0 )',0);
```

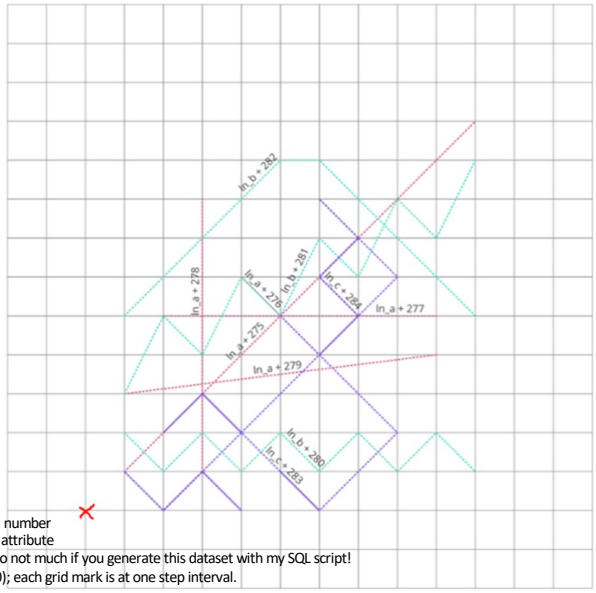


Notes:
Legend – geo test text
Colour coded by geo test type value
! Your serial numbers will do not much if you generate this dataset with my SQL script!

38

Visualisations

Lines:

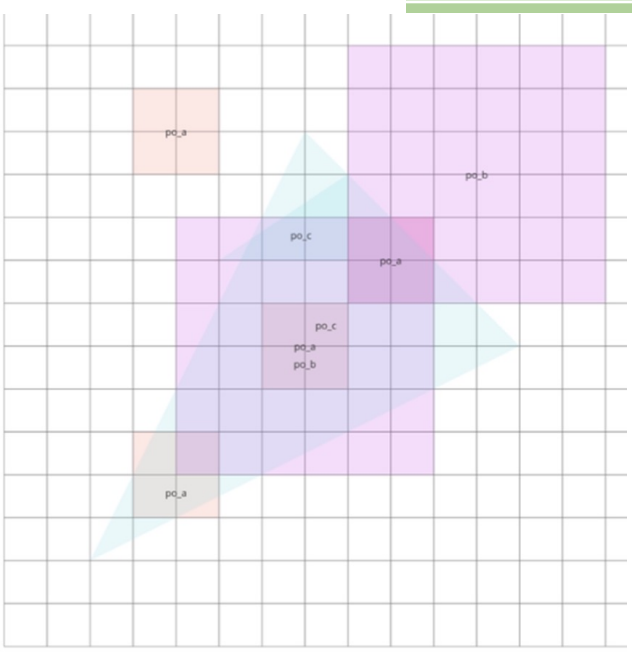


Notes:
Legend – geo type + serial number
Colour coded by geo type attribute
Your serial numbers will do not much if you generate this dataset with my SQL script!
Red cross is the origin (0,0); each grid mark is at one step interval.

39

Visualizations

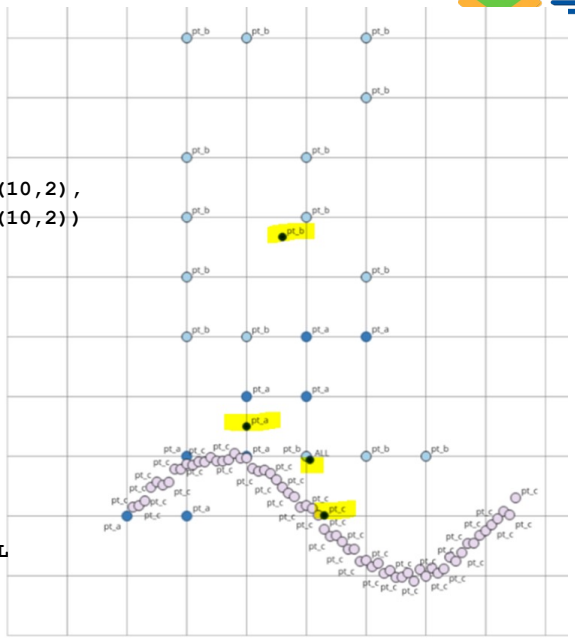
Polygons:



40

Query point - centroids

```
SELECT coalesce(p.geo_test, 'ALL')
      ::character varying(255) AS geo_test,

   ST_MakePoint(avg(ST_X(geo_obj))::numeric(10,2),
                avg(ST_Y(geo_obj))::numeric(10,2))
   AS pt_centroid_by_test
 FROM geometry.pt p
GROUP BY ROLLUP (p.geo_test);
```



Notes:
Input: str x point
Output: str x point
Renaming of rollup entry Null value with ALL
! Rounding is present

41

Query Point - Translate

```
SELECT
   p.geo_serno,
   p.geo_test,
   ST_Translate(p.geo_obj, 2, 3)
    AS geo_obj
FROM geometry.pt p;
```

	From: text	To: text
1	POINT(1 8)	POINT(3 11)
2	POINT(4 7)	POINT(6 10)
3	POINT(1 6)	POINT(3 9)
4	POINT(1 3)	POINT(3 6)
5	POINT(1 4)	POINT(3 7)
6	POINT(5 1)	POINT(7 4)
7	POINT(4 1)	POINT(6 4)
8	POINT(2 3)	POINT(4 6)
9	POINT(2 8)	POINT(4 11)
10	POINT(4 8)	POINT(6 11)
11	POINT(3 5)	POINT(5 8)
12	POINT(1 5)	POINT(3 8)
13	POINT(4 4)	POINT(6 7)
14	POINT(3 6)	POINT(5 9)
15	POINT(3 1)	POINT(5 4)

Notes:
Query to reproduce given output presented is

```
SELECT
   ST_AsText(p.geo_obj) AS 'From',
   ST_AsText(ST_Translate(p.geo_obj, 2, 3)) AS 'To'
FROM geometry.pt p;
```

42

Query point – points to multipoint

```
-- Collect all points, by geo_obj attribute value, into one point (with multipoint).
   Collect all table points into one (call it 'ALL').

SELECT coalesce(p.geo_test, 'ALL')
      ::character varying(255)  AS geo_test,
       ST_AsText(ST_Union(p.geo_obj)::geometry(MultiPoint)) AS geo_obj
  FROM geometry.pt p
 GROUP BY ROLLUP (p.geo_test);
```

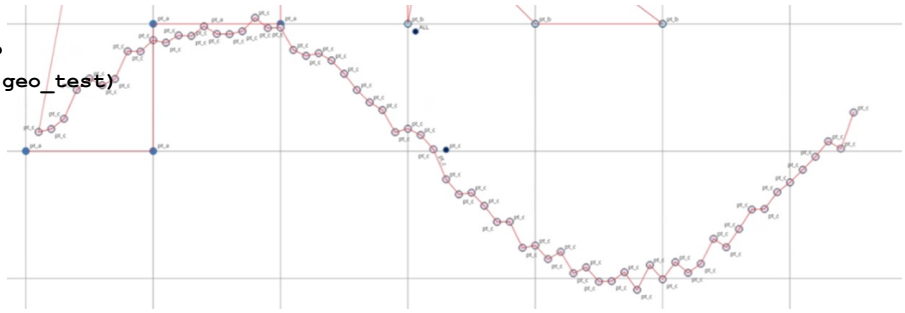
geo_test character varying (255)	st_astext text
ALL	MULTIPOINT((0 0),(0.1 0.15),(0.2 0.174),(0.3 0.254),(0.4 0.481),(0.5 0.57),(0.6 0.518),(0.7 0.567),(0.8 0.785),(0.9 0.783),(1 0),(1 0.8...
pt_c	MULTIPOINT((0.1 0.15),(0.2 0.174),(0.3 0.254),(0.4 0.481),(0.5 0.57),(0.6 0.518),(0.7 0.567),(0.8 0.785),(0.9 0.783),(1 0.872),(1.1 0....
pt_b	MULTIPOINT((1 3),(1 4),(1 5),(1 6),(1 8),(2 3),(2 8),(3 1),(3 5),(3 6),(4 1),(4 4),(4 7),(4 8),(5 1))
pt_a	MULTIPOINT((0 0),(1 0),(1 1),(2 1),(2 2),(3 2),(3 3),(4 3))

Notes:
INPUT: str x point
OUTPUT: str x multipoint
We convert the data type by aggregating (using the Group By Rollup operator)

43

Query point – points to linestring

```
SELECT coalesce(p.geo_test, 'ALL')
      ::character varying(255)
       AS geo_test,
       ST_AsText(ST_MakeLine(p.geo_obj ORDER BY p.geo_serno)
      ::geometry(LineString))
     AS geo_obj
  FROM geometry.pt p
 GROUP BY ROLLUP (p.geo_test);
```

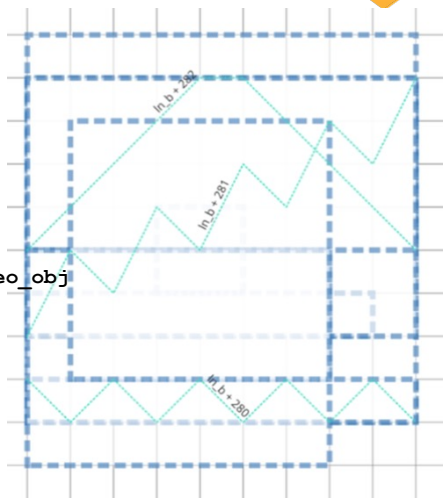


Notes:
INPUT: str x point
OUTPUT: str x linestring
We convert the data type by aggregating (Group By)
Stitching order is by the table's PK (ie a serial)

44

Linestring Envelope

```
SELECT l.geo_serno,
       l.geo_test,
       ST_Envelope(l.geo_obj)::geometry AS geo_obj
  FROM geometry.ln as l;

SELECT ll.*, ST_GeometryType(ll.geo_obj)
  FROM (SELECT l.geo_serno, l.geo_test,
               ST_Envelope(l.geo_obj)::geometry AS geo_obj
          FROM geometry.ln as l) as ll;
```



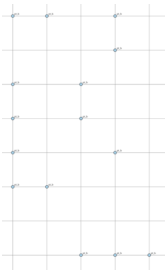
Notes:

Use the function ST_GeometryType to check the type of the output. For example, straight lines will produce a Linestring data type.

45

Query – Convex Hull of a set of points

```
SELECT coalesce(p.geo_test, 'ALL')::text
       AS geo_test,

   ST_ConvexHull(ST_Collect(p.geo_obj))::geometry(Poly
   gon)
        AS geo_obj
  FROM geometry.pt p
 GROUP BY ROLLUP (p.geo_test);
```



Notes:

input: str x pt

output: str x polygon

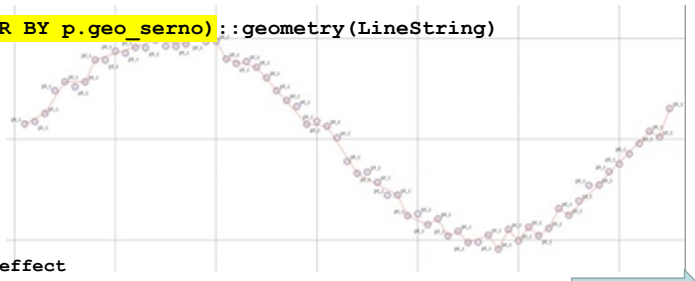
Visual on 'pt_b' test only!

Look at ST_ConcaveHull() and ST_MinimumBoundingCircle()

46

Simplify a set of points (data reduction in quantity)

```
SELECT ls.geo_test,
       ST_NPoints(geo_obj) AS np_orig,
       ST_NPoints(ST_Simplify(geo_obj,0.066)) AS np_orig33,
       ST_NPoints(ST_Simplify(geo_obj,0.100)) AS np_orig13,
       ST_NPoints(ST_Simplify(geo_obj,0.500)) AS np_orig04,
       ST_Simplify(geo_obj,0.066)::geometry(LineString) as geo_obj
FROM (
   SELECT  p.geo_test,
           ST_MakeLine(p.geo_obj ORDER BY p.geo_serno)::geometry(LineString)
           AS geo_obj
     FROM geometry.pt p
    WHERE p.geo_test='pt_c'
    GROUP BY p.geo_test ) AS ls;
```



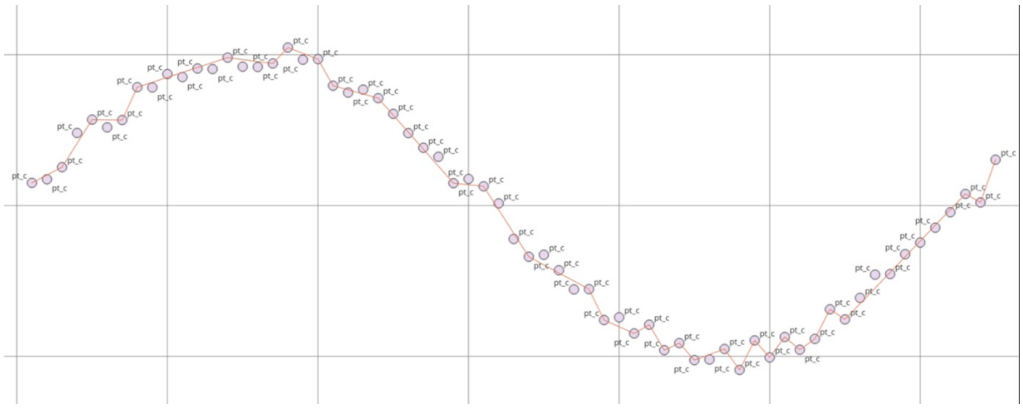
```
Notes:
input:  str x pt
output: str x linestring
Trying different tolerances and see their effect
Eg. use 0.066 for tolerance ST_Simplify reduces number of points by half.
! ST_Simplify is being phased out and ST_SimplifyPreserveTopology is recommended.
```

Next slide

47

Simplify a set of points (data reduction in number)

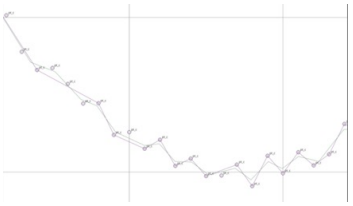
- Note the linestring (red line) has less ‘points’ then the initial number of points.



48

Smooth and Simplify a set of points

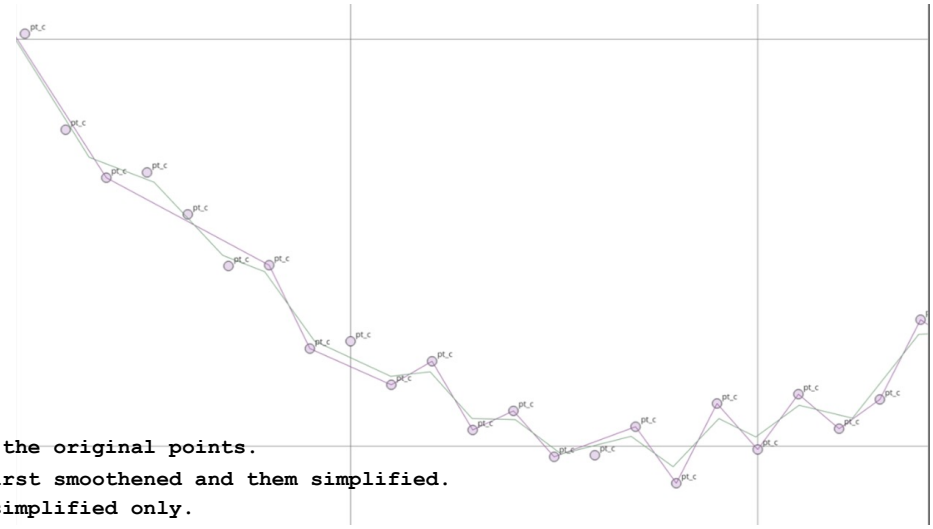
```
SELECT smooth_ls.geo_test,
      ST_NPoints(smooth_ls.geo_obj) AS np_orig,
      ST_NPoints(ST_SimplifyPreserveTopology(smooth_ls.geo_obj,0.010)) AS np_orig33,
      ST_NPoints(ST_SimplifyPreserveTopology(smooth_ls.geo_obj,0.020)) AS np_orig13,
      ST_NPoints(ST_SimplifyPreserveTopology(smooth_ls.geo_obj,0.033)) AS np_orig04,
      ST_SimplifyPreserveTopology(smooth_ls.geo_obj,0.033)::geometry(LineString) AS
   geo_obj
  FROM (  SELECT ls.geo_test, ST_ChaikinSmoothing(ls.geo_obj,5) as geo_obj
        FROM ( SELECT p.geo_test,
                      ST_MakeLine(p.geo_obj ORDER BY p.geo_serno)::geometry(LineString)
                      AS geo_obj
                 FROM geometry.pt p
                WHERE p.geo_test='pt_c'
                GROUP BY p.geo_test
                  ) AS ls
       ) AS smooth_ls;
```



Next slide

49

Smooth and Simplify a set of points (continued)

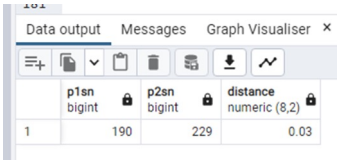


Violet dots are the original points.
Green line is first smoothened and them simplified.
Violet line is simplified only.

50

Are there any two points close to each other?

```
SELECT p1.geo_serno as p1sn, p2.geo_serno as p2sn, ST_Distance(p1.geo_obj, p2.geo_obj)
  FROM geometry.pt AS p1,
       geometry.pt AS p2
 WHERE ST_Distance(p1.geo_obj, p2.geo_obj) <= 0.1
   AND p1.geo_serno < p2.geo_serno;
```

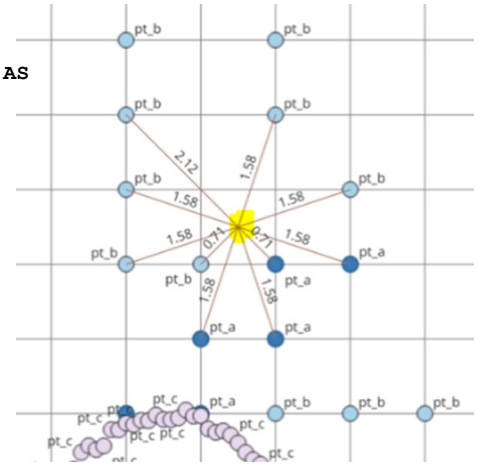


Notes:
input pt x pt x numeric
output pt x pt x numeric

51

Top N Closest Points – Which ten points are closest to a given point?

```
SELECT p.geo_serno "to",
       ST_MakeLine(t.geo_obj,p.geo_obj),
       ST_Distance(p.geo_obj,t.geo_obj)::numeric(10,2) AS
   distance
  FROM (VALUES (-1::integer,
                'my position'::character varying(255),
                ST_GeomFromText('POINT(2.5 3.5)'))
       ) AS t(geo_serno,geo_test,geo_obj),
       geometry.pt AS p
 ORDER BY distance ASC LIMIT 10;
```



Notes:
input: (str x pt) X pt
output: str x number x linestring

Point of interest is given by POINT(2.5 3.5)- yellow highlight.

52

Which two points are closest to each other?

```
SELECT p1.geo_serno,  st_astext(p1.geo_obj),
      p2.geo_serno,  st_astext(p2.geo_obj),
      st_distance(p1.geo_obj, p2.geo_obj) as "distapart"
  FROM geometry.pt p1, geometry.pt p2
 WHERE p1.geo_serno > p2.geo_serno
   AND geo_test = 'pt_b'
   AND st_distance(p1.geo_obj, p2.geo_obj) <= ALL (
        SELECT st_distance(p1.geo_obj, p2.geo_obj)
          FROM geometry.pt p1, geometry.pt p2
         WHERE p1.geo_serno > p2.geo_serno
           AND geo_test = 'pt_b')
 ORDER BY distapart;
```

geo_serno bigint	st_astext text	geo_serno bigint	st_astext text	distapart double precision
229	POINT(2 0.971)	190	POINT(2 1)	0.029000000000000026

53

Length of Linestrings

```
SELECT
   l.geo_serno,
   l.geo_test,
   ST_Length(l.geo_obj)::numeric(8,2) as "distar
   ST_NumPoints(l.geo_obj) as "npoints",
   ST_IsSimple(l.geo_obj) as "is_simple"
FROM geometry.ln as l;
```

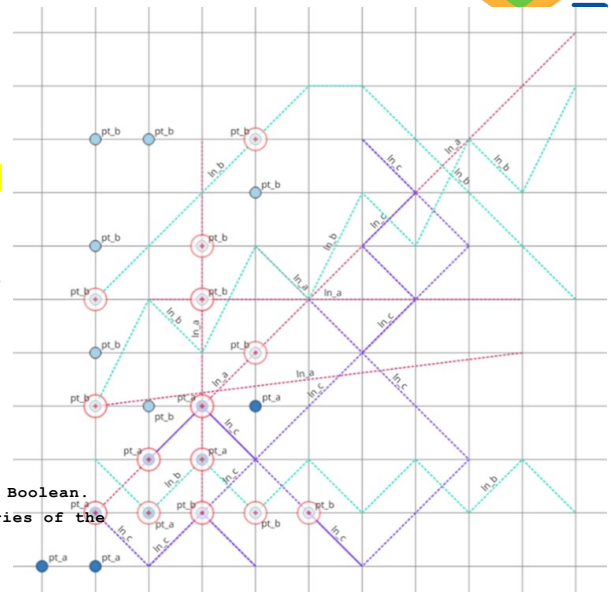
	geo_serno bigint	geo_test character varying (255)	distance numeric (8,2)	npoints integer	is_simple boolean
1	275	ln_a	12.73	2	true
2	276	ln_a	2.83	2	true
3	277	ln_a	7.00	2	true
4	278	ln_a	7.00	2	true
5	279	ln_a	8.06	2	true
6	280	ln_b	12.73	10	true
7	281	ln_b	16.84	10	true
8	282	ln_b	12.31	10	true
9	283	ln_c	26.87	20	false
10	284	ln_c	26.87	20	false

Notes:
The distance unit is related to the geometry's reference system.
E.g., meters, kilometers, miles, degrees.
Use ST_LengthSpheroid() for ellipsoid distance (and 3D).

54

Linestring and Point intersection

```
SELECT row_number() OVER() AS serno,
       l.geo_serno as l_serno,
      p.geo_serno as p_serno,
      ST_Intersection(l.geo_obj,p.geo_obj)
      ::geometry(Point) AS geo_obj
  FROM geometry.ln AS l,
       geometry.pt AS p
 WHERE ST_Intersects(l.geo_obj,p.geo_obj);
```

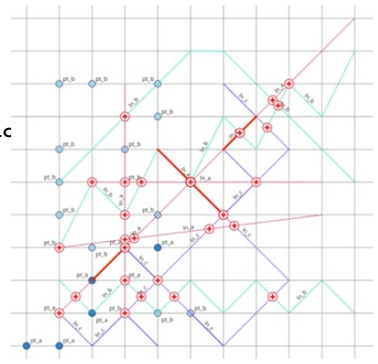


Notes:
ST_Intersects is a test/predicate that returns a Boolean.
ST_Intersection is a function that return geometries of the intersection

55

Linestrings intersection

```
SELECT row_number() OVER() AS serno,
       l1.geo_serno as l1_serno, l2.geo_serno as l2_serno,
      ST_AsText(ST_Intersection(l1.geo_obj,l2.geo_obj))
  FROM geometry.ln AS l1,
       geometry.ln AS l2
 WHERE ST_Intersects(l1.geo_obj,l2.geo_obj)
   AND l1.geo_serno > l2.geo_serno;  -- ignore self intersectic
```



Notes:
Input: [serno/integer , pt] x [pt , linestring]
Output id/integer x pt | linestring

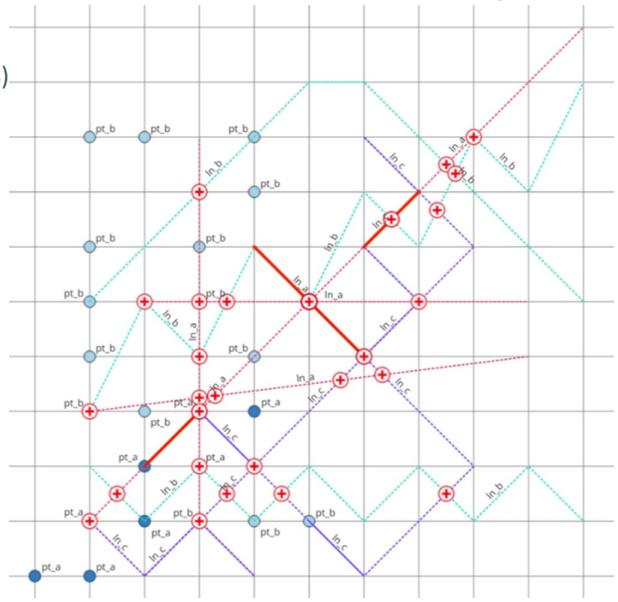
Output geometry types include Point, Linestring, MultiPoint, and MultiLinestring.

Next slide

56

Linestrings intersection (continued)

- Thick red lines (n=3) and thick red crosses (n=28)

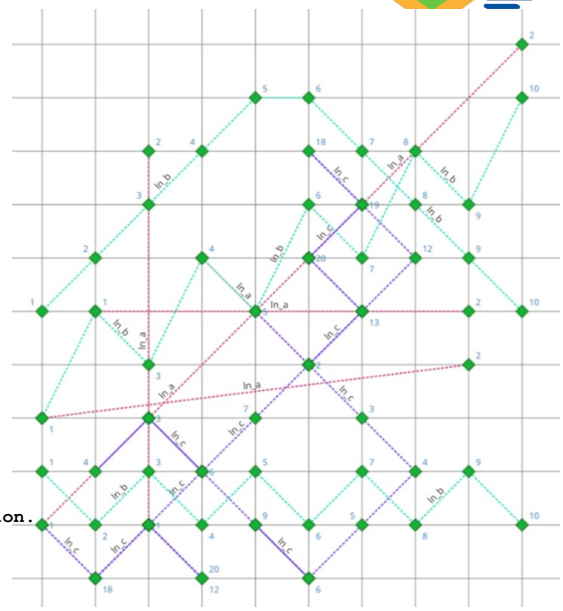


57

Linestring to Points

```
SELECT row_number() OVER() AS serno,
       (ln).geo_serno,
       (ln).path[1] As index,
       ST_AsText((ln).geom)::geometry(Point)
       as geo_obj
  FROM (SELECT (ST_DumpPoints(geo_obj)).*,
               geo_serno
          FROM geometry.ln) AS ln;
```

Notes:
path[n] and geom are related to the ST_DumpPoints function.



58

Buffer a Linestring

```
SELECT l.geo_serno,
       l.geo_test,
       ST_Buffer(l.geo_obj, .10, 'endcap=round join=round')
       AS geo_obj
  FROM geometry.ln as l;
```

Notes:

The 0.10 is the buffer radius; endcap and join define the appearance of the generated polygon at endpoints and sharp turns (joins).

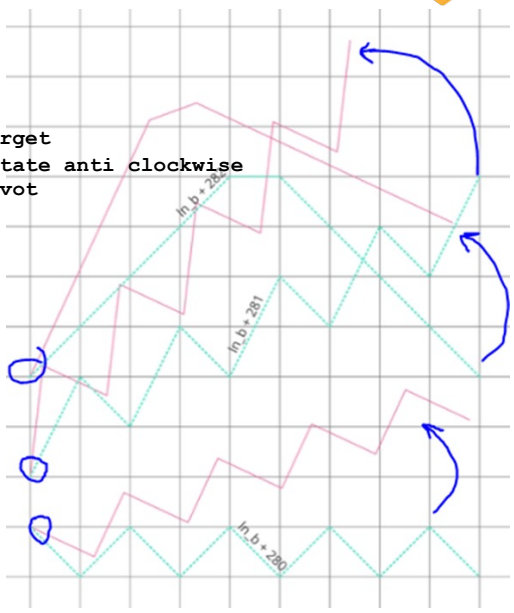
If the input geometry is a point, the output is a circle (with a default of 8 edges).

Check the manual for other options.

59

Rotate a Linestring

```
SELECT l.geo_serno,
       l.geo_test,
       ST_Rotate(l.geo_obj,                -- target
                 pi()/9,                   -- rotate anti clockwise
                 ST_StartPoint(l.geo_obj)) -- pivot
     AS geo_obj
    FROM geometry.ln as l
   WHERE geo_test='ln_b';
```



60

Intersection – Polygon vs Linestring

```
SELECT g.geo_serno, g.geo_test,
       l.geo_serno, l.geo_test,
       ST_GeometryType(
         ST_Intersection(
           g.geo_obj,
           l.geo_obj)  )
FROM geometry.pg as g INNER JOIN
     geometry.ln as l
  ON ST_Intersects(g.geo_obj, l.geo_obj);
```

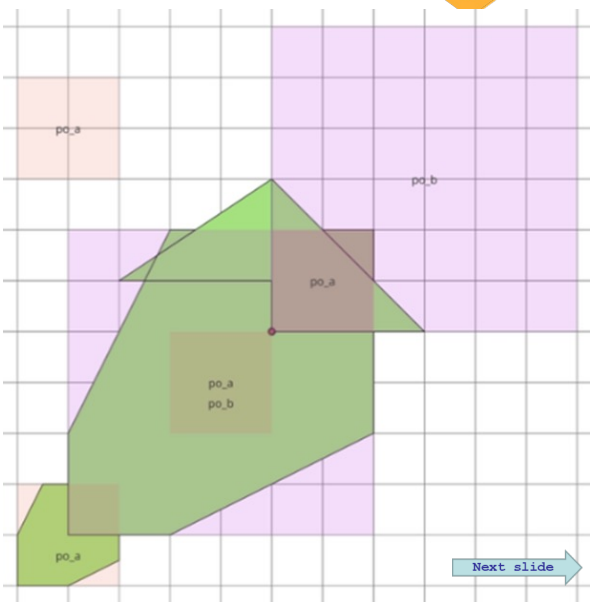
	geo_serno bigint	geo_test character varying (255)	geo_serno bigint	geo_test character varying (255)	st_geometrytype text
1	285	po_a	275	ln_a	ST_LineString
2	285	po_a	278	ln_a	ST_LineString
3	285	po_a	279	ln_a	ST_Point
4	285	po_a	280	ln_b	ST_MultiLineString
5	285	po_a	281	ln_b	ST_Point
6	285	po_a	283	ln_c	ST_MultiPoint

Notes:
The data types of the intersections are: Point, LineStrings, MultiPoint, MultiLineStrings, GeometryCollection.
ST_Intersects() is a Boolean function / predicate.
ST_Intersection() is a function that builds structure out of two object's intersection.

61

Intersection – Polygon vs Polygon

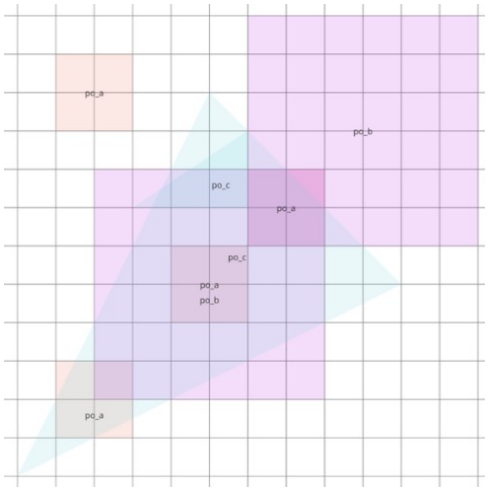
```
SELECT row_number() OVER() AS serno,
       ST_Intersection(
        g1.geo_obj,
        g2.geo_obj)
  FROM geometry.pg as g1 INNER JOIN
       geometry.pg as g2
    ON ST_Intersects(
        g1.geo_obj,
        g2.geo_obj)
 WHERE g1.geo_serno>g2.geo_serno;
    -- ignore self intersection
```

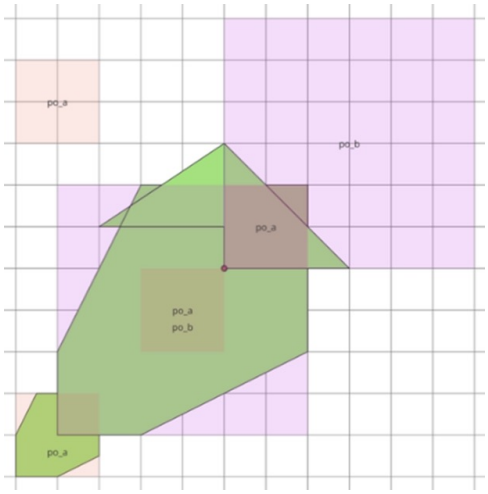


Notes:
The data types of the intersections are: Point, Linestring, Polygon, and their combinations

62

Intersection – Polygon vs Polygon (continued)

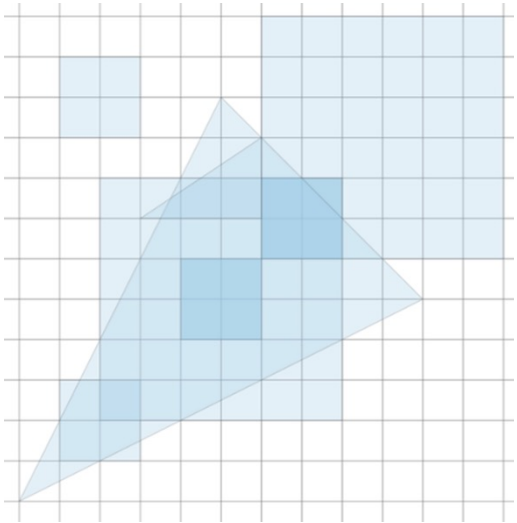




63

Polygon Within a Polygon

```
SELECT row_number() OVER() AS serno,
       ST_Intersection(g1.geo_obj, g2.geo_obj)
  FROM geometry.pg as g1 INNER JOIN geometry.pg as
    ON ST_Within(g1.geo_obj, g2.geo_obj);
```



```
Notes: ST_Within is not symmetric
       ST_Contains is an inverse of ST_Within
```

64

Which points are within '3' distance units from our polygons?

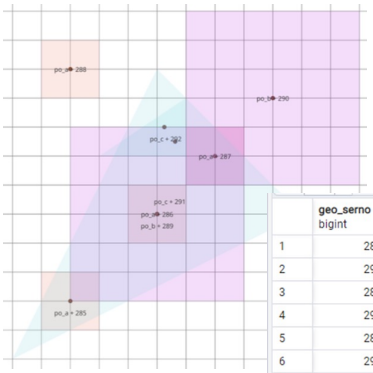
```
SELECT g.geo_serno, g.geo_test,
       p.geo_serno, p.geo_test,
      ST_Distance(g.geo_obj, p.geo_obj) as distance
  FROM geometry.pg as g INNER JOIN geometry.pt as p
    ON ST_DWithin(g.geo_obj, p.geo_obj, 3)
 WHERE p.geo_test<>'pt_c'
 ORDER BY distance DESC;
```

	geo_serno bigint	geo_test character varying (255)	geo_serno bigint	geo_test character varying (255)	distance double precision
1	287	po_a	196	pt_b	3
2	286	po_a	202	pt_b	3
3	286	po_a	203	pt_b	3
4	285	po_a	199	pt_b	3
5	288	po_a	196	pt_b	3
6	288	po_a	199	pt_b	3
7	285	po_a	196	pt_b	3
8	290	po_b	196	pt_b	3
9	292	po_c	205	pt_b	3
10	286	po_a	209	pt_b	3
11	286	po_a	199	pt_b	3
12	286	po_a	201	pt_b	3
13	289	po_b	187	pt_a	2.8284271247461903
14	290	po_b	205	pt_b	2.8284271247461903

65

How are two Polygons Spatially Related?

```
SELECT g1.geo_serno, g2.geo_serno,
       ST_Relate(g1.geo_obj,g2.geo_obj)
  FROM geometry.pg as g1, geometry.pg as g2
 WHERE g1.geo_serno in (291, 286);
```



	geo_serno bigint	geo_serno bigint	st_relate text
1	286	285	FF2FF1212
2	291	285	212101212
3	286	286	2FFF1FFF2
4	291	286	212FF1FF2
5	286	287	FF2F01212
6	291	287	212101212
7	286	288	FF2FF1212
8	291	288	FF2FF1212
9	286	289	2FF1FF212
10	291	289	212101212
11	286	290	FF2F01212
12	291	290	212101212
13	286	291	2FF1FF212
14	291	291	2FFF1FFF2
15	286	292	FF2FF1212
16	291	292	212101212

Notes:

ST_Relate() describes a spatial relationship through interior, exterior, and boundary interactions of the two artefacts – as in our slide 'Spatial Relationships – Coverage Semantics'.

ST_Relate() output is a 9-character representing a 3x3 matrix Interior, Exterior, and Boundary.

66

Linestring and Polygon with a specific spatial relationship

```
-- Given lines and polygons, which line is
   in a polygon and a line's boundary
   must touch the polygon's boundary?

SELECT g1.geo_serno, l1.geo_serno,
  FROM geometry.pg as g1, geometry.l as l1
 WHERE ST_Relate(l1.geo_obj, g1.geo_obj,'1FF00F212')
   AND g1.geo_serno in (291, 286);
```

		Polygon		
		Interior	Boundary	Exterior
Linestring	Interior	1	F	F
	Boundary	0	0	F
	Exterior	2	1	2

Notes:		
Interaction	T	Yes/No
	0	'T' and Point
	1	'T' and Line
	2	'T' and Area
	F	Null
	*	Yes/Null

Spatial Relation				
Equals	T*F**FFF*			
Disjoint	FF*FF****			
Touches	FT*******	F**T*****	F***T****	
Contains	T*****FF*			
Covers	T*****FF*	*T****FF*	***T**FF*	****T*FF*
Intersects	T********	*T*******	***T*****	****T****
Within	T*F**F***			
Covered By	T*F**F***	*TF**F***	**FT*F***	**F*TF***
Crosses	T*T******	T*****T**	0********	
Overlaps	T*T***T**	1*T***T**		

T = 0|1|2

From "PostGIS in Action" – Obe & Hsu, 2021

67

Try Out!

- For intersection (ST_Intersection) rewrite good examples with:
  Union (ST_Union)
  - Use ST_Union as an aggregating operator too

  Difference (ST_Difference) And the related (ST_SymDifference)

68

Spatial Data Analytics

Data Clustering

- We need some distance measure
  Technically a distance measure needs to be **metric**:
  - *Non−negative, Identity has no distance, Symmetry, and Triangular Inequality.*
- **Points**
  K-means
  Voronoi Diagrams
- **Linestrings**
  How similar are two paths?
- **Polygon**
  How similar are two shapes?

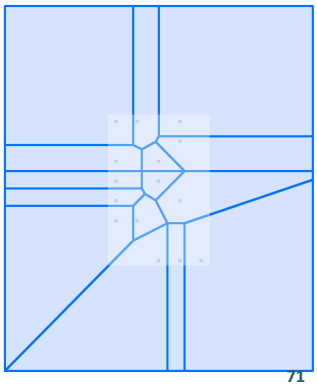
70

Point clustering - K-Means and Voronoi Diagram (based on Cartesian Distance)

```
-- k-means (k=3)
SELECT
   geo_serno, st_astext(geo_obj),
   ST_ClusterKMeans(geo_obj, 3) OVER () AS
   cluster
 FROM geometry.pt
WHERE geo_test='pt_b'
ORDER BY "cluster" ASC;
```

geo_serno bigint	st_astext text	cluster integer
205	POINT(4 4)	0
202	POINT(5 1)	0
203	POINT(4 1)	0
197	POINT(3 1)	0
195	POINT(1 8)	1
198	POINT(4 7)	1
199	POINT(1 6)	1
206	POINT(2 8)	1
207	POINT(4 8)	1
196	POINT(3 6)	1
208	POINT(3 5)	2
209	POINT(1 5)	2
204	POINT(2 3)	2
201	POINT(1 4)	2
200	POINT(1 3)	2

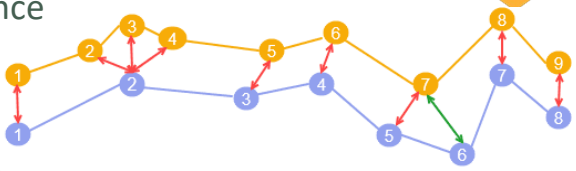
```
-- Points to Areas with Voronoi's method
SELECT ST_VoronoiPolygons(multi_geo_obj)
  FROM (SELECT ST_Multi(ST_Collect(geo_obj))
          AS multi_geo_obj
          FROM geometry.pt
         WHERE geo_test='pt_b')
         AS t(multi_geo_obj);
```



71

Linestring Clustering – Frechet Distance

Two linestrings (orange and blue). A distance (red line) is calculated between the closest vertices in the two linestrings. The green line is the discrete **Fréchet distance**.



developers.arcgis.com/geoanalytics/sql-functions/st_frechet_distance/

```
SELECT ST_FrechetDistance(v_pt_c.geo_obj, line.geom)
       as "FrechetDistance"
  FROM geometry.vw_simplifytopoorder_by_test_pt_c as v_pt_
      (SELECT geo_obj
         FROM geometry.vw_simplifytopoorder_by_test_pt_c
        UNION
       SELECT geo_obj
         FROM geometry.ln) as line(geom)
 ORDER BY "FrechetDistance" ASC;
```

FrechetDistance double precision
0
3.568531350569867
5.861530175645265
6.762089100270715
6.894381770688362
7.030824987154779
7.712225100449285
8.454490877634205
8.824421567445652
9.373922124703192
10.308366310914645

72

Polygon Clustering – Turning Distance

- The **turning distance** between two polygon objects is a measure of how closely their shapes match, regardless of rotation or scaling. A turning distance close to 0 indicates a near match. The larger the value, the more the two shapes differ.

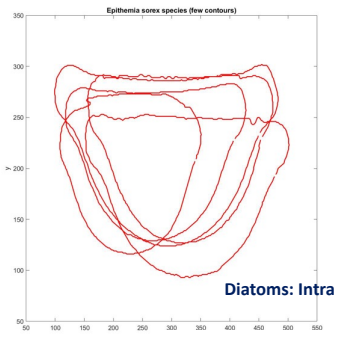
  From(Matlab)

  www.mathworks.com/help/matlab/ref/polyshape.turningdist.html

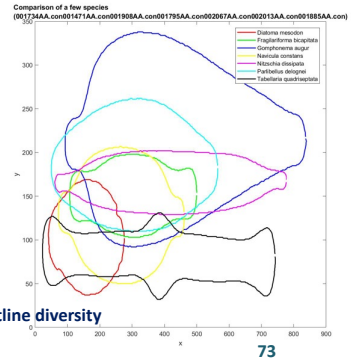
  Reference:

  Arkin, E.M., Chew, L.P., Huttenlocher, D.P., Kedem, K., and Mitchell, J.S.B. "An efficiently computable metric for comparing polygonal shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol. 13, Number 3, 1991, pp. 209-16.

Not in PostGIS!



Diatoms: Intra species outline diversity



Diatoms: Inter species outline diversity

73

Issues

Additional challenges

- **Spatial Data Quality**
  - Intrinsic quality
    - **Provenance** of the data source
  - Multiple sources
    - Variety
      - Formats
      - Resolution – as in resolution of location (x,y)
      - Resolution in frequency of capture – e.g., number of points making up a linestring
  - Integrity Constraints
    - How much can integrity constraints help to maintain data quality?
  - Can the spatial data be integrated with non-spatial data?
    - How much are we loosing (data items) on joining?
    - Do we need to change the resolution of the spatial (e.g., from a neighbourhood to city extent?)

75

Additional challenges (continued)

- Technical
  - **Performance**
    - Ingress/import of data
      - Also, consider in database vs out-of-database
    - Indexing (i.e., multidimensional (MD)) vs Non-Indexing
      - Creating costs (time and resources)
      - Maintenance (including re-index)
    - Views and Materialised Views (MV)
      - E.g., Use MVs when updates are rare and computation is long winded
    - Rewrite queries (includeing transactions)
      - Use Common Table Expressions (CTE)
      - Batch updates (run transactions with small number of updates)
      - Use two pass Query Processing (i.e., assume MD indexes are in place) features in SQL
        - 1st pass query by MBBs
        - 2nd for hits in the first pass, do an exact match
    - Choice of programming langauges
      - Back-end (for PostgreSQL) – do not assume *one* with suffice or is adequate!?
        - PL/pgSQL, PL/Python
      - Front-end
        - Any really – same warning as for back-end

76

Additional challenges (continued)

- Technical
  - Training
  - Compentences (ICT and domain)
    - Data Modelling
    - Query Modelling
      - Including Integrity constraints
    - Batch processing (to address computational bottlenecks)
    - Physical design
    - Perfomance profiling
    - Coupling with tools (e.g., QGIS for visualisalitioins and analysis)
  - Quality assurance of software related activities

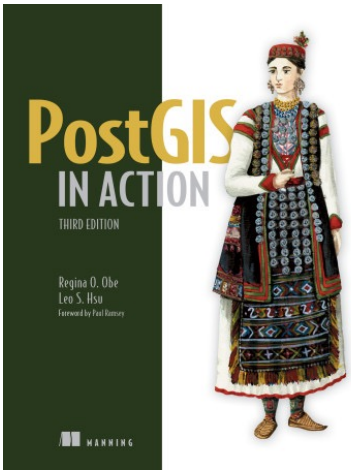
77

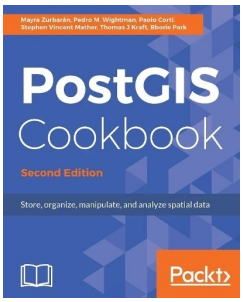
Thanks

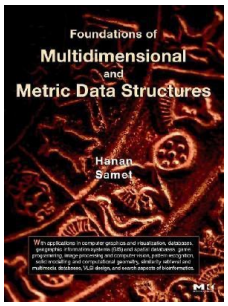
Do start a discussion with me if interested on getting a few hints to start your first spatial data analytics project …

References (other than directly cited in the slide deck)

- Main text books and references







- Blogs, Technical Papers from these sources are always very insightfull:
  - **Paul Ramsey**
  - **Pierre Racine**
  - Crunchy Data Blog
  - https://gis.stackexchange.com/
  - Postgis (and PostgreSQL) on Redit