Tips for the PostGIS Power User

Topics

- PostGIS functions
  - Geometry constructors / deconstructors accessors / spatial predicates
  - Walk through a few examples.
- DE-9IM
  - Fine-tuning spatial predicates
- PostgreSQL
  - Table inheritance / partitioning
  - Database tuning

Introduction

- What is PostGIS?
  - A PostgreSQL database extension that "spatially enables" the server back-end to support the storage of geometric objects in an object-relational PostgreSQL database.
  - http://postgis.refractions.net/docs/

Introduction

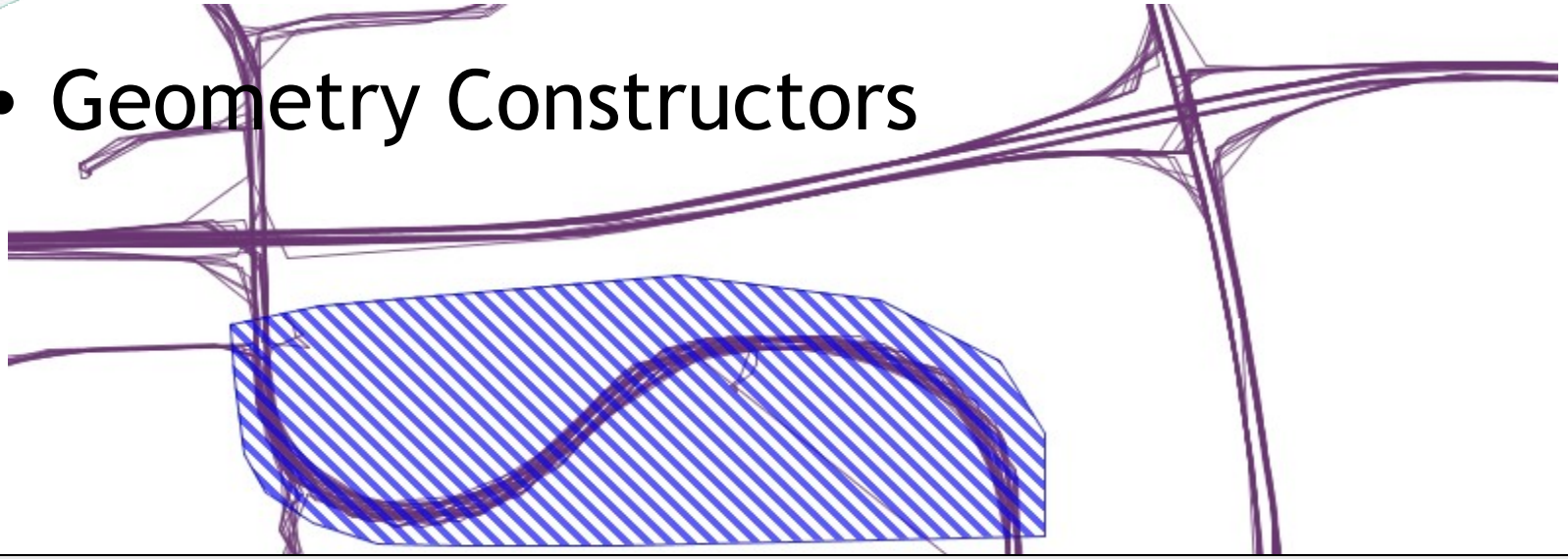
- Geometry Constructors
  - ST_GeomFromText(text)
  - ST_GeomFromWKB(bytea)

```
SELECT ST_GeomFromText( 'POINT(1718098 616348)', 3005 );
```

  - Creates a geometry data type in EPSG:3005 projection

Introduction

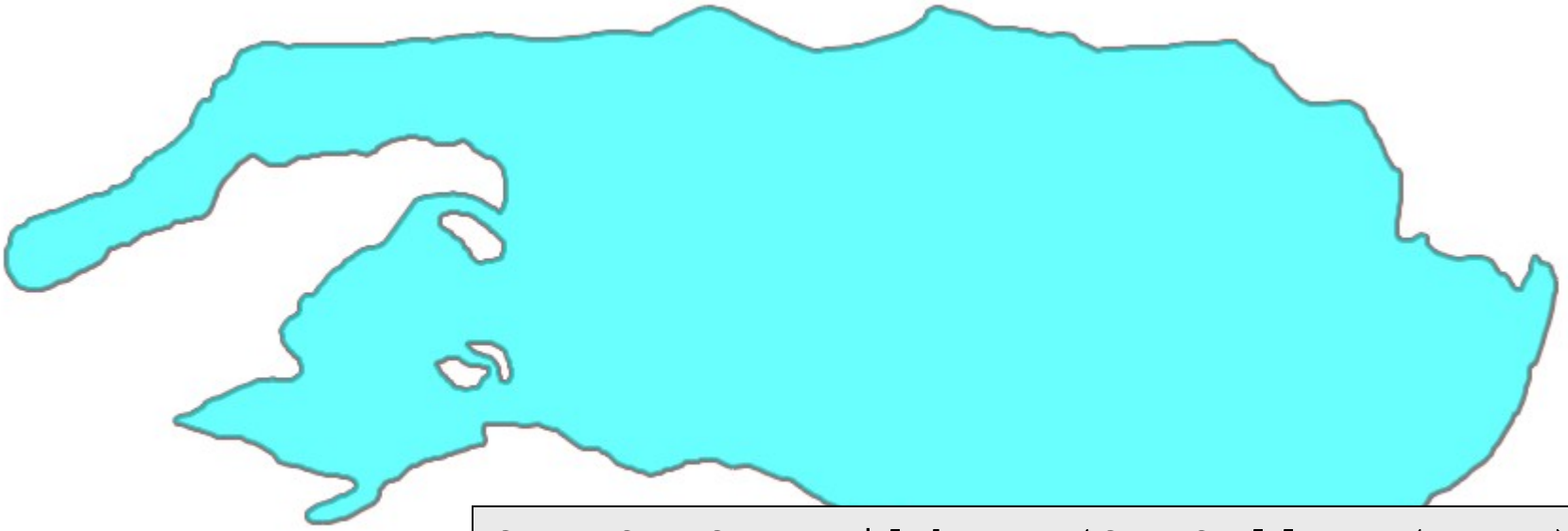
- Geometry Constructors



```
SELECT ST_MakeLine(gps_points.geom)
FROM (SELECT geom
      FROM gps_point_data
      WHERE time_stamp::date = '2007-09-22'::date
      ORDER BY time_stamp
      ) AS gps_points;
```

Introduction

- Geometry Constructors
  - ST_BuildArea()



```
SELECT ST_BuildArea(ST_Collect(geom))
FROM ...
```

Introduction

- Geometry Accessors / Deconstructors
  - ST_StartPoint()
  - ST_PointN(geometry, int)
  - ST_ExteriorRing(geometry)

```
SELECT ST_StartPoint(geom)
FROM my_lines;
```

Introduction

- Geometry Accessors / Deconstructors

```
SELECT point
FROM (SELECT ST_StartPoint(geom) AS point
      FROM my_lines
      UNION ALL
      SELECT ST_EndPoint(geom) AS point
      FROM my_lines) AS a
GROUP BY point
HAVING count(*) = 4;
```

Caution: GROUP BY uses a geometry's bounding box

Introduction

- Geometry Accessors / Deconstructors

```
SELECT ST_GeometryN( geom, 1 ) FROM my_multilines;
```

  - How to explode a MULTI* table

```
SELECT generate_series( 1, 5 );
```

```
 generate_series
-----------------
```

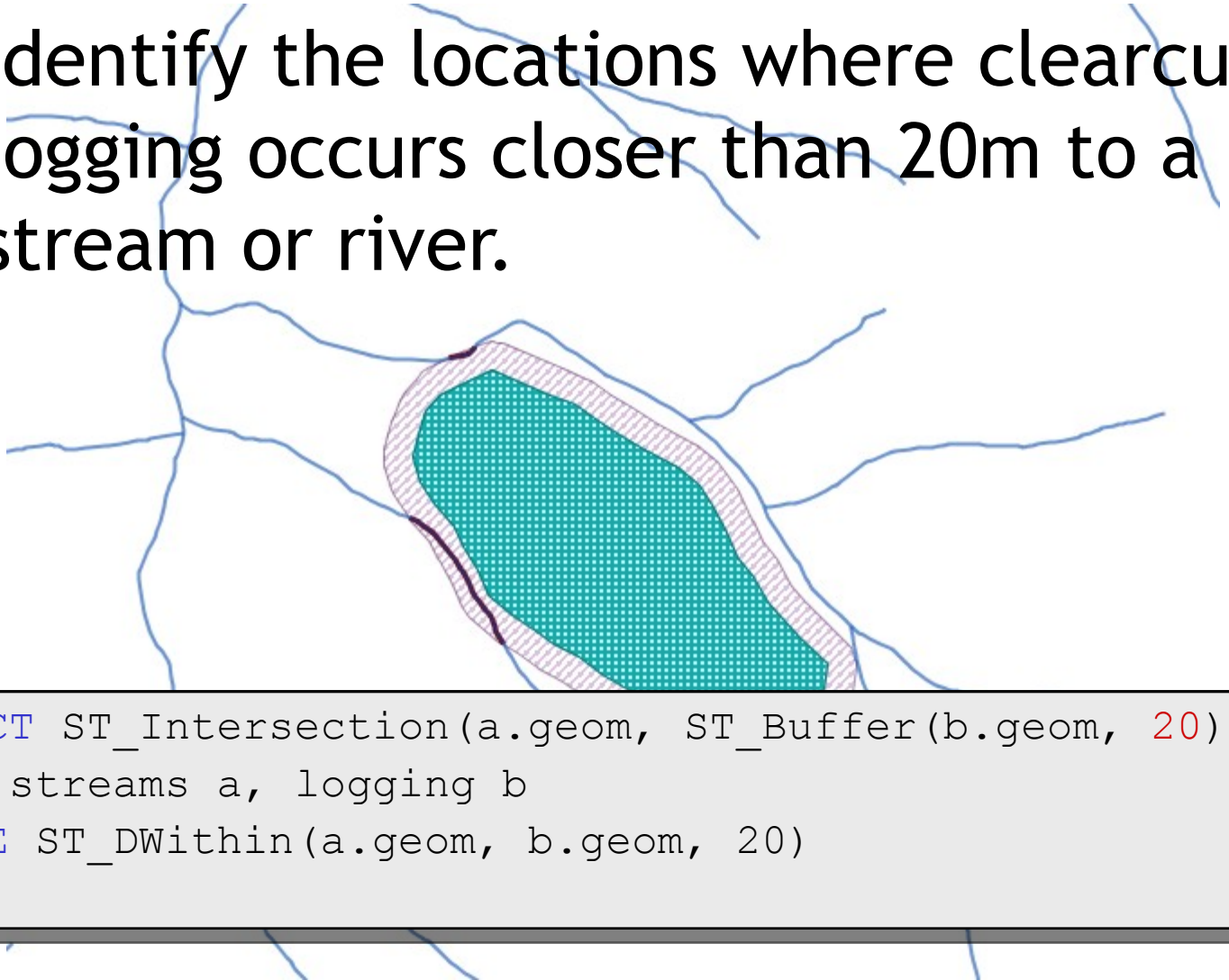
```
CREATE TABLE my_lines AS
SELECT geom
FROM (SELECT ST_GeometryN(geom,
               generate_series(1, ST_NumGeometries(geom)))
             AS geom
      FROM my_multilines
      ) AS foo;
```

Introduction

- Geometry Spatial Predicates / Functions
  - ST_Intersects()
  - ST_Within()
  - ST_Touches()
  - ST_GeomUnion()
  - ST_SymmetricDifference()
  - ST_ConvexHull()
  - …

Sample PostGIS Queries

1. Identify the locations where clearcut logging occurs closer than 20m to a stream or river.



```
SELECT ST_Intersection(a.geom, ST_Buffer(b.geom, 20))
FROM streams a, logging b
WHERE ST_DWithin(a.geom, b.geom, 20)
```

Sample PostGIS Queries

1. What is the average elevation of a lake digitized in 3D?

```
SELECT avg(ST_Z(ST_PointN(ring,
                          generate_series(1, ST_NumPoints(ring))
           )
         )
FROM (
   SELECT ST_ExteriorRing(geom) AS ring
   FROM lakes
   WHERE lake_id = 1

   UNION ALL

   SELECT ST_InteriorRingN(geom,
                           generate_series(1, ST_NumInteriorRings(geom))
                          ) AS ring
   FROM lakes
   WHERE lake_id = 1
) AS foo
```

Sample PostGIS Queries

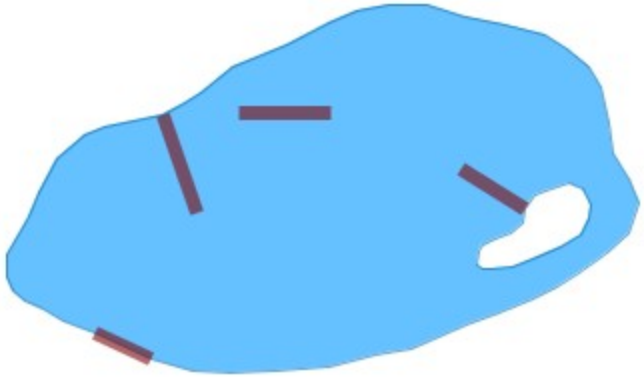
1. Efficiently, union a set of polygons.


```
SELECT ST_Union(the_geom)
FROM ...

(takes ~16.7 seconds)
```

```
SELECT ST_Buffer(ST_Collect(geom), 0.0)
FROM ...

(takes ~4.1 seconds)
```

Sample PostGIS Queries

1. Find all docks that are contained completely within a lake, not touching a lake bank.



What PostGIS functions would you use?

~~ST_Within~~? ~~ST_Contains~~? ~~ST_Touches~~?

DE-9IM The Dimensionally Extended – Nine Intersection Model

- Approach
  - make pair-wise tests of the intersections between the Interiors, Boundaries, and Exteriors of two geometries and to represent these relationships in an "intersection" matrix

DE-9IM The Dimensionally Extended – Nine Intersection Model

	Interior	Boundary	Exterior
Interior	*dim(**I**(a) ∩ **I**(b))*	*dim(**I**(a) ∩ **B**(b))*	*dim(**I**(a) ∩ **E**(b))*
Boundary	*dim(**B**(a) ∩ **I**(b))*	*dim(**B**(a) ∩ **B**(b))*	*dim(**B**(a) ∩ **E**(b))*
Exterior	*dim(**E**(a) ∩ **I**(b))*	*dim(**E**(a) ∩ **B**(b))*	*dim(**E**(a) ∩ **E**(b))*

Possible values:

{T, F, *, 0, 1, 2}

Where:

T == {0,1,2}

F == empty set

* == don't care

0 == dimensional 0 – point

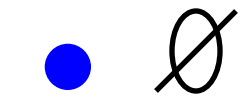
1 == dimensional 1 – line

2 == dimensional 2 - area

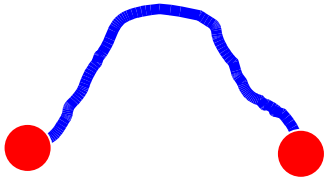
DE-9IM The Dimensionally Extended – Nine Intersection Model

Geometry Topology

- Boundary
  - the set of geometries of the next lower dimension



Point
(dim-0)



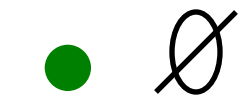
Line
(dim-1)

Polygon
(dim-2)

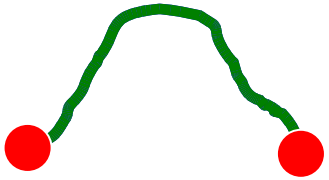
DE-9IM The Dimensionally Extended – Nine Intersection Model

Geometry Topology

- Interior
  - the points that are left when the boundary points are removed



Point
(dim-0)



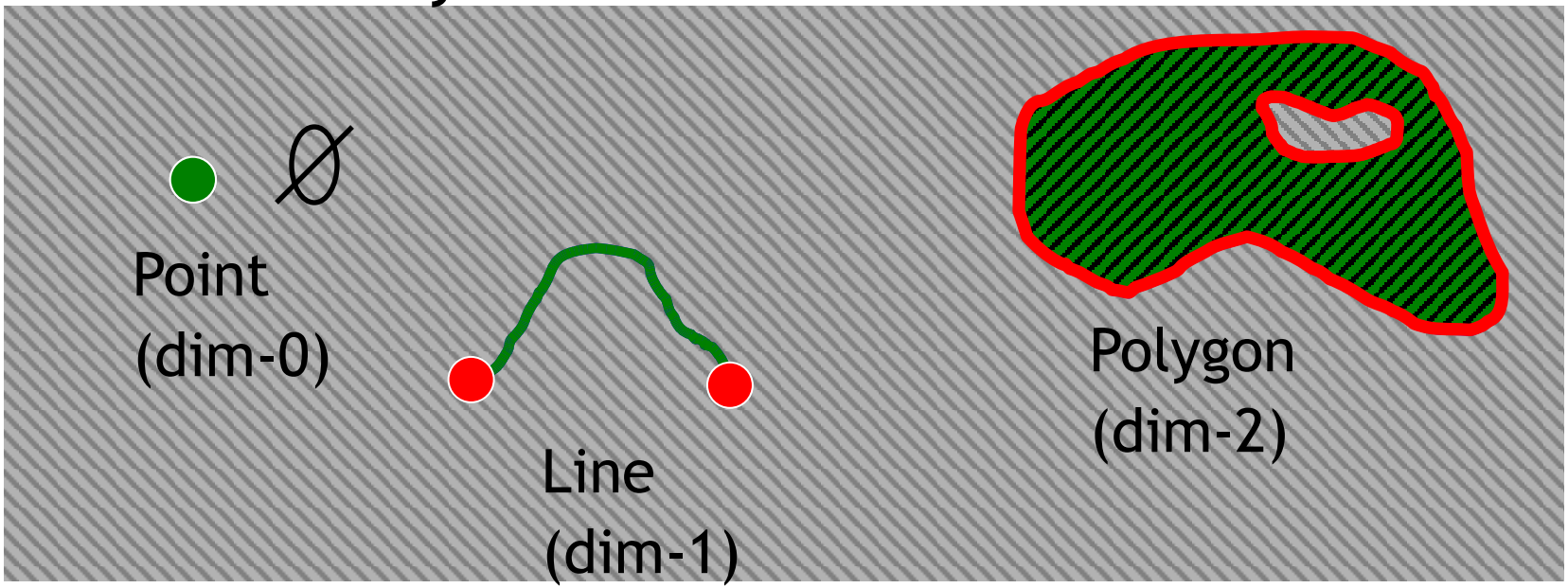
Line
(dim-1)

Polygon
(dim-2)

DE-9IM The Dimensionally Extended – Nine Intersection Model

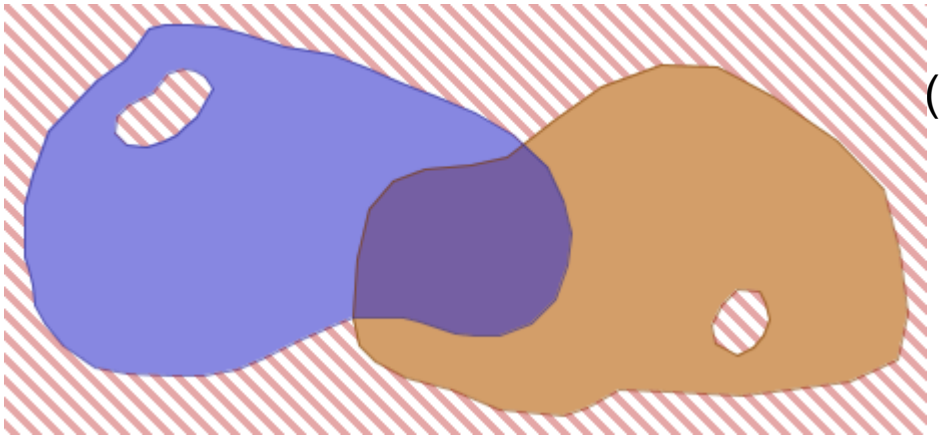
Geometry Topology

- Exterior
  - consists of points not in the interior and boundary



DE-9IM The Dimensionally Extended – Nine Intersection Model

(a)



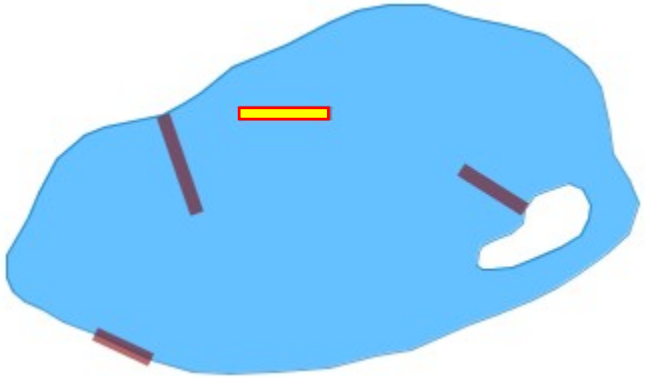
(b)

	Interior	Boundary	Exterior
Interior	2	1	2
Boundary	1	0	1
Exterior	2	1	2

ST_Relate(a, b) = ‘212101212’

Sample PostGIS Queries

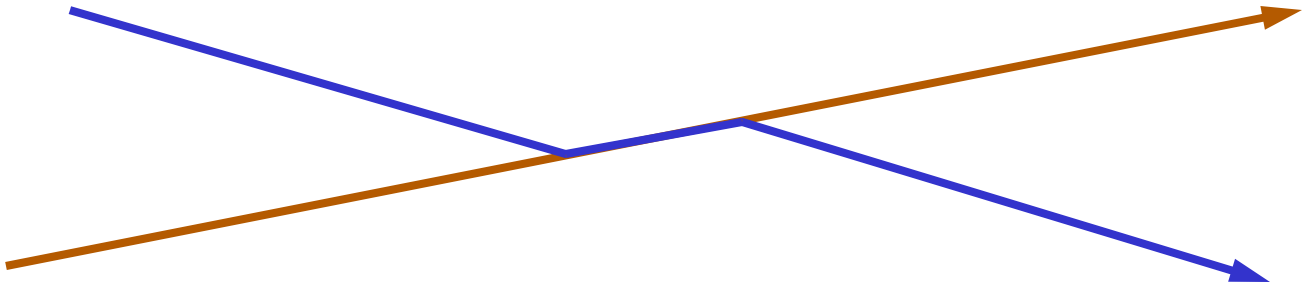
1. Find all docks that are contained completely within a lake, not touching a lake bank.



```
SELECT a.id
FROM docks a, lakes b
WHERE a.geom && b.geom
AND ST_Relate(a.geom, b.geom, 'TFFTFF212');
```

Sample PostGIS Queries

1. Identify linear spatial features that intersect on a line and not at a point.



```
SELECT a.id, intersection(a.geom, b.geom)
FROM mylines a, mylines b
WHERE a.id != b.id
AND a.geom && b.geom
AND ST_Relate(a.geom, b.geom, '1*1***1**');
```

Table Inheritance

```
CREATE TABLE cities (
    name            text,
    population      real,
    altitude        int
);
```

```
CREATE TABLE capitals (
    province    text
) INHERITS (cities);
```

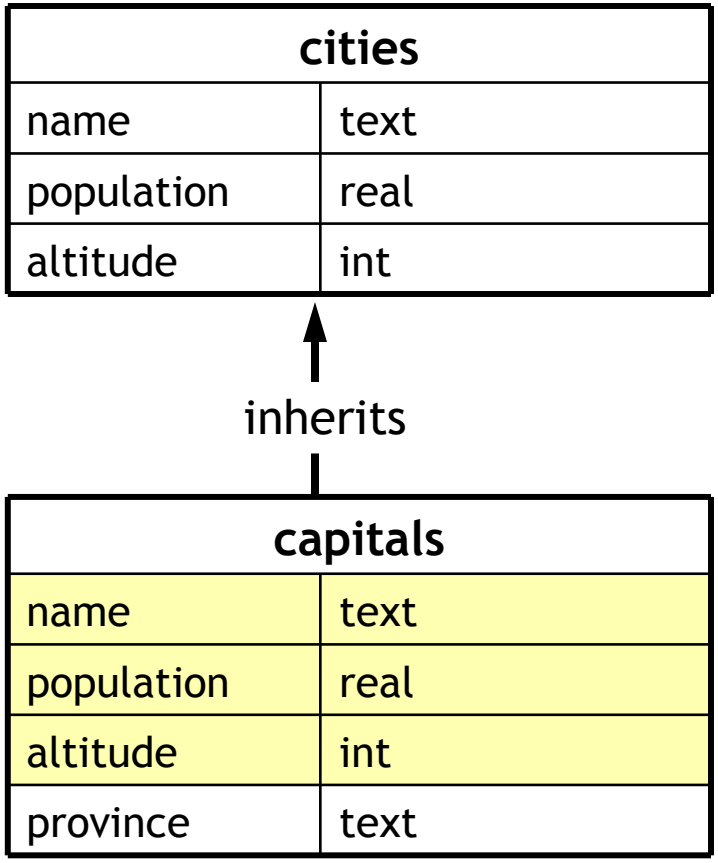


Table Inheritance

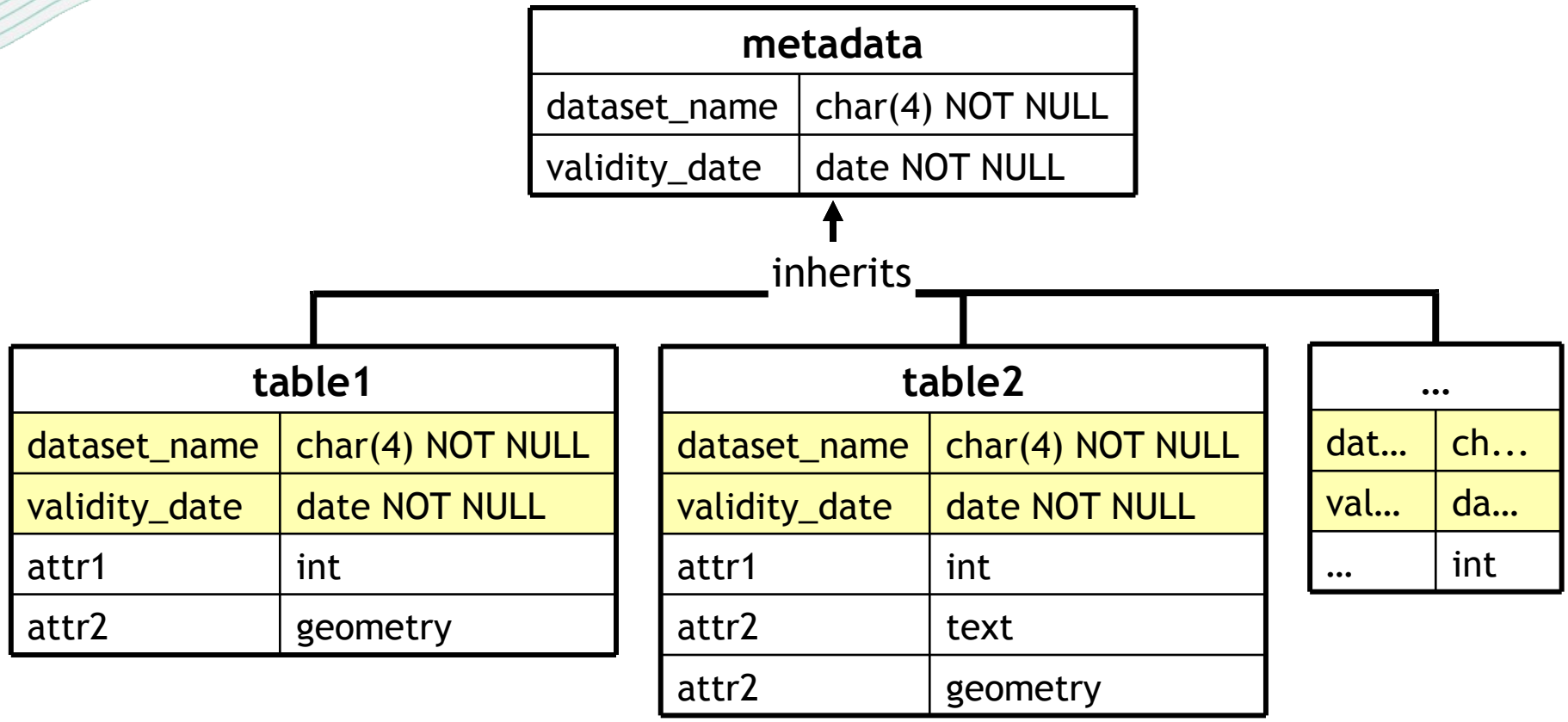


Table Partitioning

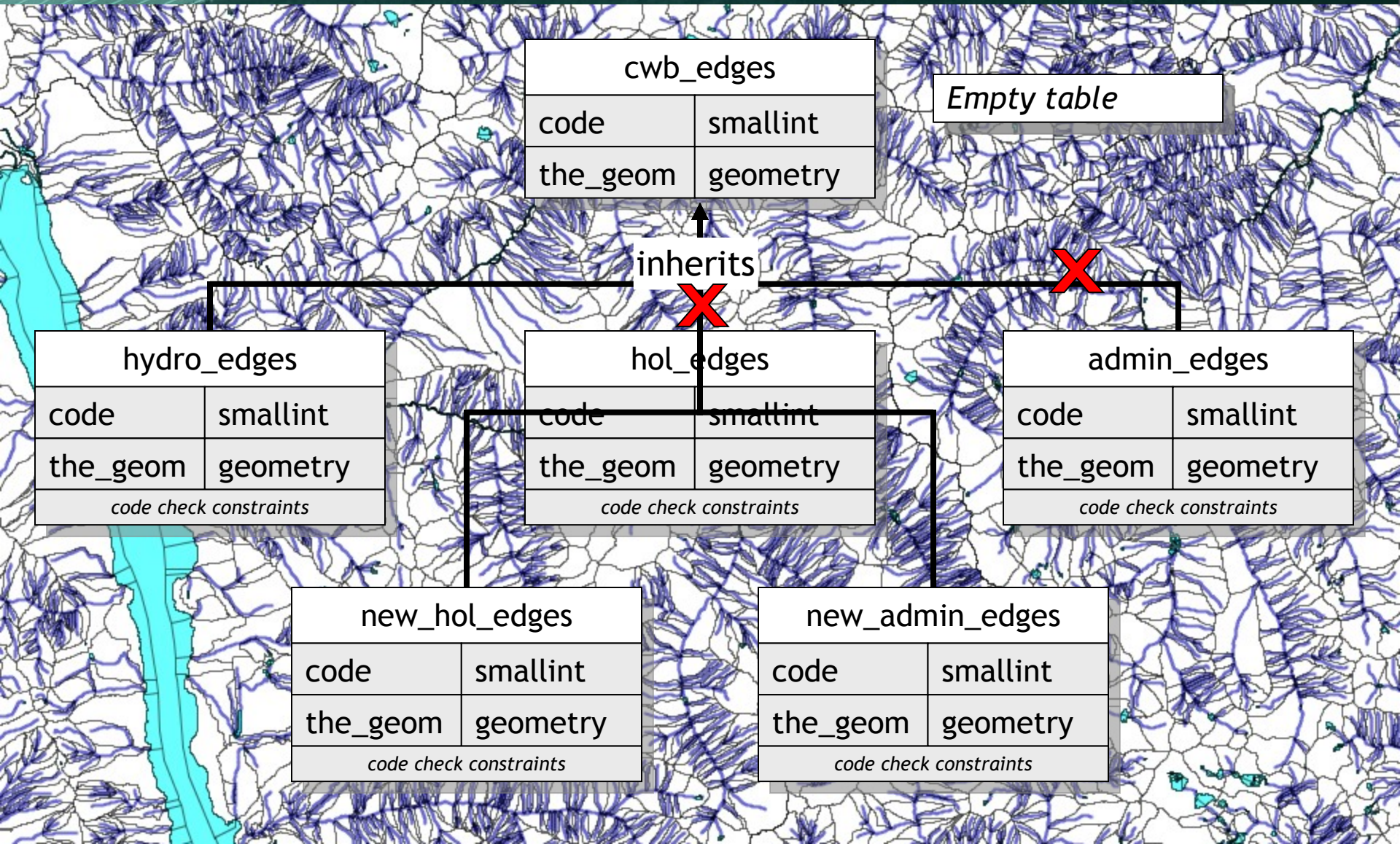


Table Partitioning

cwb_edges	
code	smallint
the_geom	geometry

hydro_edges	
code	smallint
the_geom	geometry
code check constraints	

```
-- ADD Rules to parent table
CREATE RULE insert_hydro AS
    ON INSERT TO cwb_edges WHERE code = ...
    DO INSTEAD
    INSERT INTO hydro_edges (code, geom)
    VALUES (NEW.code, NEW.geom);
```

```
CREATE TABLE hydro_edges (
) INHERITS cwb_edges;

ALTER TABLE hydro_edges
    ADD CONSTRAINT code_check
    CHECK (code = ...);
```

PostgreSQL Tuning

- The biggest bottleneck in a spatial database is I/O
- When setting up a server, invest in a:
  - ***great*** file system
    - RAID 5 – good for web servers, **not** spatial DBs
    - Recommend RAID 1/0
  - ***good*** memory
  - ***adequate*** CPU(s)

PostgreSQL Tuning

- postgresql.conf
  - Startup
    - checkpoint_segment_size
      - # of WAL files - 16MB each
      - Default: 3
      - Set to at least 10 or 30 for databases with heavy write activity or more for large database loads
      - Possibly store the xlog on a separate disk device
    - shared_buffers
      - Default: ~32MB
      - About 1/3 to 3/4 of available RAM

PostgreSQL Tuning

- postgresql.conf
  - Startup
    - constraint_exclusion
      - Default: "off"
      - Set to "on" to ensure the query planner will optimize as desired.

PostgreSQL Tuning

- postgresql.conf
  - Runtime
    - work_mem
      - Memory used for sort operations and complex queries
      - Default: 1MB
      - Adjust up for large dbs, complex queries, lots of RAM
      - Adjust down for many concurrent users or low RAM

PostgreSQL Tuning

- postgresql.conf
  - Runtime
    - maintainence_work_mem
      - Memory used for VACUUM, CREATE INDEX, etc.
      - Default:16MB
      - Generally too low – ties up I/O, locks objects while swapping memory.
      - Recommend 32MB to 256MB on production servers with lots of RAM, but depends on number of concurrent users.

PostgreSQL Tuning

- postgresql.conf
  - Runtime
    - On development systems with lots of RAM and few developers…

```
SET work_mem TO 1200000;
```

```
SET maintainence_work_mem TO 1200000;
```

PostgreSQL Tuning

- postgresql.conf
  - Runtime
    - client_min_messages

```
SET client_min_messages to DEBUG;
```

      - Useful when writing PL/Pgsql functions.

```
CREATE FUNCTION my_function () RETURNS TEXT AS
$BODY$
BEGIN
        ...
        RAISE DEBUG 'myvar: %' var;
        ...
```

Performance Tips

- Spatial function calls can be expensive. Be efficient in their use - avoid unnecessary/duplicate function calls.
  - Use St_Expand where appropriate
  - Use one relate call instead of 2 or 3 other spatial calls.
  - Use St_Distance()==0 instead of intersects() on large geometries
  - Avoid St_Buffer() unless you need a buffered geometry

Performance Tips

- Partition your data into Most Frequently Used (MFU) and Least Frequently Used (LFU).


Questions

Appendex A

```
// PostGIS and JTS
Class.forName("org.postgresql.Driver");
Connection conn =
   DriverManager.getConnection("jdbc:postgresql://...");

WKBReader wkbReader = new WKBReader();
WKBWriter wkbWriter = new WKBWriter();

String query =
   "SELECT the_geom FROM my_spatial_table
    WHERE the_geom && ST_GeomFromWKB(?, 3005)");
PreparedStatement pstmt = conn.prepareStatement(query);
pstmt.setBytes(1, wkbWriter.write(myJTSPolygon);

ResultSet rs = pstmt.executeQuery();
while(rs.next) {
   Geometry g = wkbReader.read(WKBReader.hexToBytes(
                    rs.getString(1)));
   ...
   // Do stuff with Geometry
}
```

Appendex B