https://is.gd/1uzKi5

PostGIS 3.0 Deep Dive

paul.ramsey@crunchydata.ca

PostGIS Day STL - November 14, 2019

crunchy data

What’s in a Number?

PostGIS 0.X

- First release in May 2001
- Proof of concept
  - Can PostgreSQL store GIS data?
  - Will it perform?
- Release of Mapserver driver shortly after

- All coordinates stored 3D (24 bytes per coordinate)
- Double aligned storage fields for
  - 3D bounding box (48 bytes)
  - object count (4 bytes)
  - type (4 bytes)
  - dimensionality flag (1 byte)
- 2D Point > 81 bytes

- **Used in roads and watersheds project work**

PostGIS 1.X

- Released in April 2005
- Arnulf Christl
  - “Just release 1.0 so the users aren’t scared!”
- Response to operational experience
  - Lots of points, and short lines
  - Not much use of 3D
  - Header and 3D overhead weighted down performance
- 0.X was “heavyweight”, so 1.X became “lightweight”
  - LWGEOM structures
  - lw* naming prefix
- Upgrade from 0.X required **dump and restore**
  - On-disk format change

- Header in just one byte!
  - Bit flags for dimensions, box, SRID
  - 4 bits for geometry type
- Non-aligned storage
- Optional float bounding box
  (16 bytes)
- Optional SRID
  (4 bytes)
- Higher dimensionality optional
  (16 bytes per coordinate)
- 2D Point = 17 bytes!

PostGIS 2.X

- Released in April 2012
- Support for PostgreSQL extension framework
  - "CREATE EXTENSION postgis"
- Response to limitations in 1.X
  - Out of space for type numbers
    4 bits means max of 16 types
  - Unaligned coordinates mean all accesses require copying to aligned memory
  - Mixture of direct and indirect access to storage structures in code base
  - "Optional" SRID was always there
- Upgrade from 1.X required **dump and restore**
  - On-disk format change

- Header in eight bytes!
  - 1 byte for flags
  - 3 bytes for (mandatory) SRID
  - 4 bytes for type number
- Aligned storage
  - Direct access to coordinates
- Optional float bounding box
  (16 bytes)
- Higher dimensionality optional
  (16 bytes per 2D coordinate)
- 2D Point = 28 bytes

PostGIS 3.X

- Released in September 2019
- Response to limitations in 2.X
  - Out of space for flags!
    - 2 bits for version
    - 1 bit for bounding box
    - 2 bits for dimensionality
    - 1 bit for geography
    - 1 bit for solid
    - 1 bit for readonly
  - Solid? Readonly?
- Re-organize flags
  - Bump version number
  - Move solid flag to extra flag space
- Upgrade from 2.X **does not** require **dump and restore**
- Reorganize extensions

- Header in eight bytes!
  - 1 byte for flags
  - 3 bytes for (mandatory) SRID
  - 4 bytes for type number
- Aligned storage
- Optional float bounding box (16 bytes)
- Optional extra flag space (8 bytes)
- Higher dimensionality optional (16 bytes per 2D coordinate)
- 2D Point = 28 bytes
- 2D Point = 20 bytes???

Semantic Versioning (Major.Minor.Patch)

Increment:

- **MAJOR** version when you make incompatible API changes,
- **MINOR** version when you add functionality in a backwards compatible manner, and
- **PATCH** version when you make backwards compatible bug fixes.

For PostGIS, increment:

- **MAJOR** version for **on-disk format** changes and major **operational** changes,
- **MINOR** version for batches of new functionality, annual release, and
- **PATCH** version for fixes that do not change behaviour or API.

Major Changes in PostGIS 3?

crunchy data

Raster Extension Split

PostGIS 2.5

```
CREATE EXTENSION postgis;
  → postgis-2.5.so
  → rtpostgis-2.5.so
CREATE EXTENSION postgis_sfcgal;
  → postgis-2.5.so
CREATE EXTENSION postgis_topology;
CREATE EXTENSION address_standardizer;
  → address_standardizer-2.5.so
CREATE EXTENSION postgis_geocoder;
```

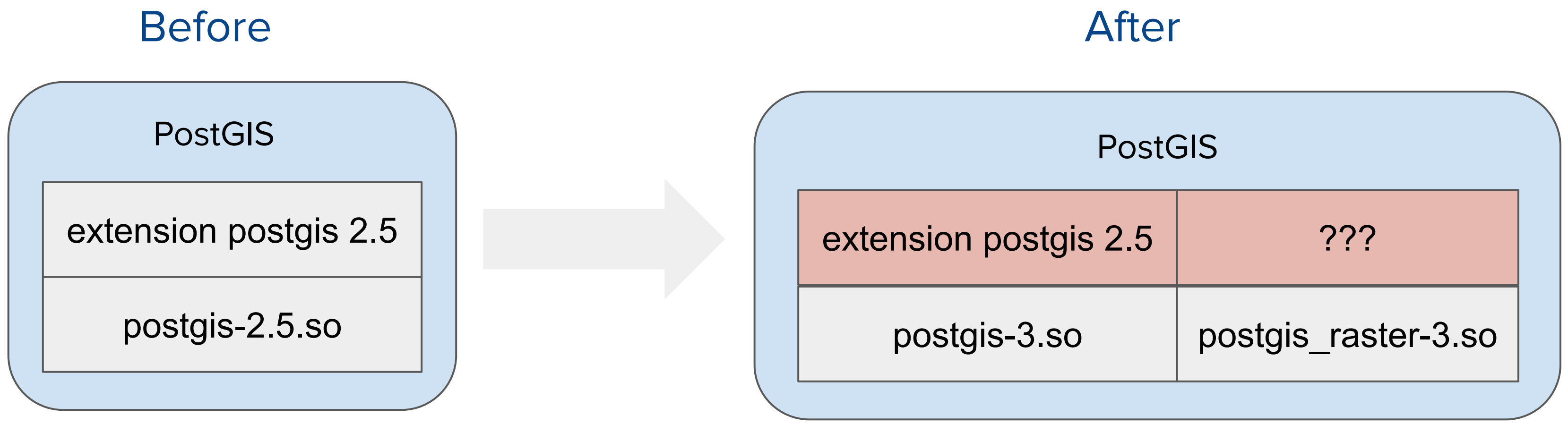
PostGIS 3.0

```
CREATE EXTENSION postgis;
  → postgis-3.so
CREATE EXTENSION postgis_raster;
  → postgis_raster-3.so
CREATE EXTENSION postgis_sfcgal;
  → postgis-3.so
CREATE EXTENSION postgis_topology;
CREATE EXTENSION address_standardizer;
  → address_standardizer-3.so
CREATE EXTENSION postgis_geocoder;
```

Raster Extension Split

Upgrade Software

```
# systemctl stop postgresql-11
# rpm -e postgis25_11 postgis25_11-client
# yum install postgis30_11 postgis30_11-client
# systemctl start postgresql-11
```



Raster Extension Split

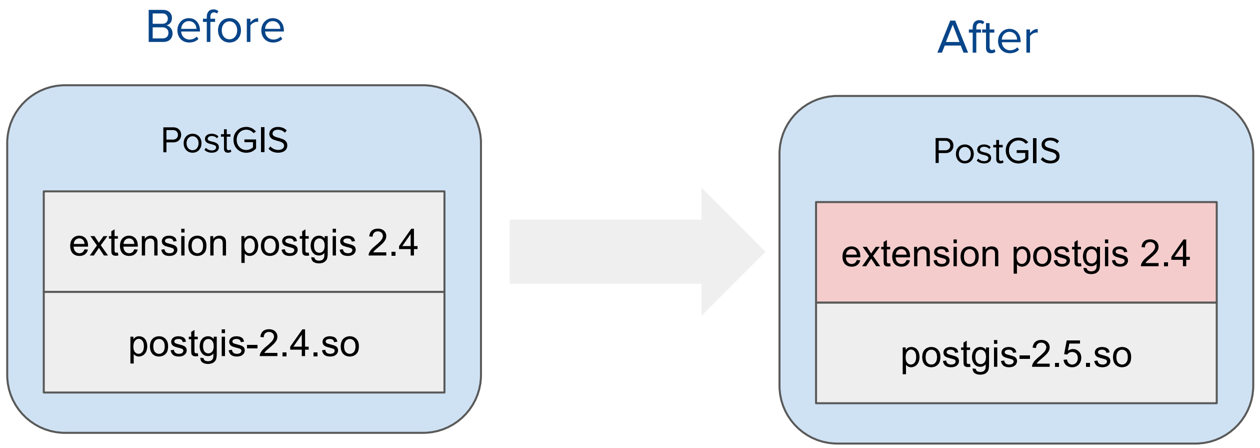
Upgrade Database SQL

```
> ALTER EXTENSION postgis UPDATE TO '3.0.0';
WARNING:  unpackaging raster
WARNING:  PostGIS Raster functionality has been unpackaged
HINT:  type `SELECT postgis_extensions_upgrade(); to finish the upgrade. After upgrading, if you
want to drop raster, run: DROP EXTENSION postgis_raster;
ALTER EXTENSION

> SELECT postgis_extensions_upgrade();
NOTICE:  Packaging extension postgis_raster
NOTICE:  Extension postgis_topology is not available or not packagable for some reason
NOTICE:  Extension postgis_tiger_geocoder is not available or not packagable for some reason
                    postgis_extensions_upgrade
-------------------------------------------------------------------
 Upgrade completed, run SELECT postgis_full_version(); for details

> SELECT postgis_full_version();
POSTGIS="3.0.0 r17983" [EXTENSION] PGSQL="110" GEOS="3.8.0-CAPI-1.11.0 " PROJ="6.2.0" GDAL="GDAL
2.4.0dev-c3279e1, released 2018/06/18" LIBXML="2.9.4" LIBJSON="0.13" LIBPROTOBUF="1.3.1"
WAGYU="0.4.3 (Internal)" RASTER
```

Major-only Library Version Number

Upgrade 2.4 to 2.5

```
# systemctl stop postgresql-11
# rpm -e postgis24_11 postgis24_11-client
# yum install postgis25_11 postgis25_11-client
# systemctl start postgresql-11
```



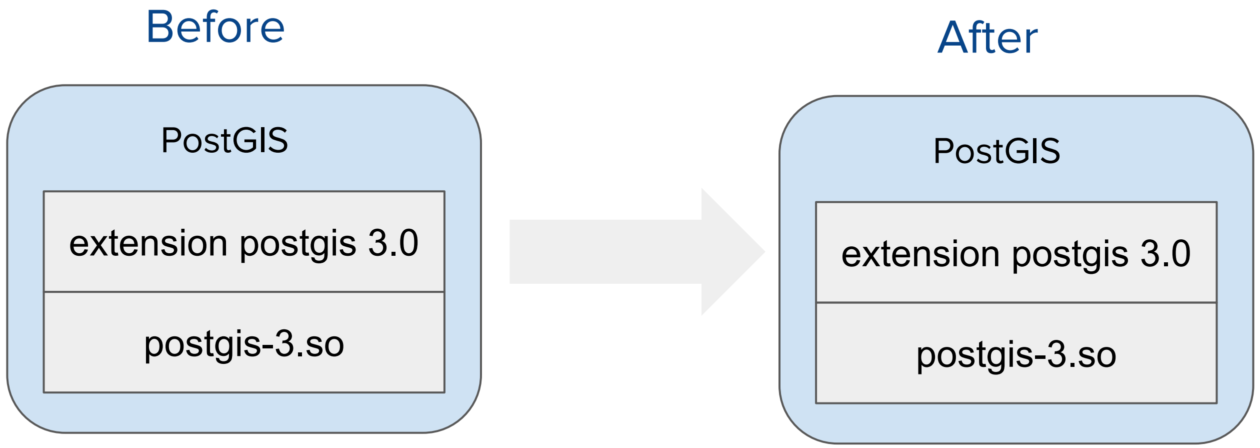
Action:

`ALTER EXTENSION UPDATE` to fix your **broken system**.

Major-only Library Version Number

Upgrade 3.0 to 3.1

```
# systemctl stop postgresql-11
# rpm -e postgis30_11 postgis30_11-client
# yum install postgis31_11 postgis31_11-client
# systemctl start postgresql-11
```



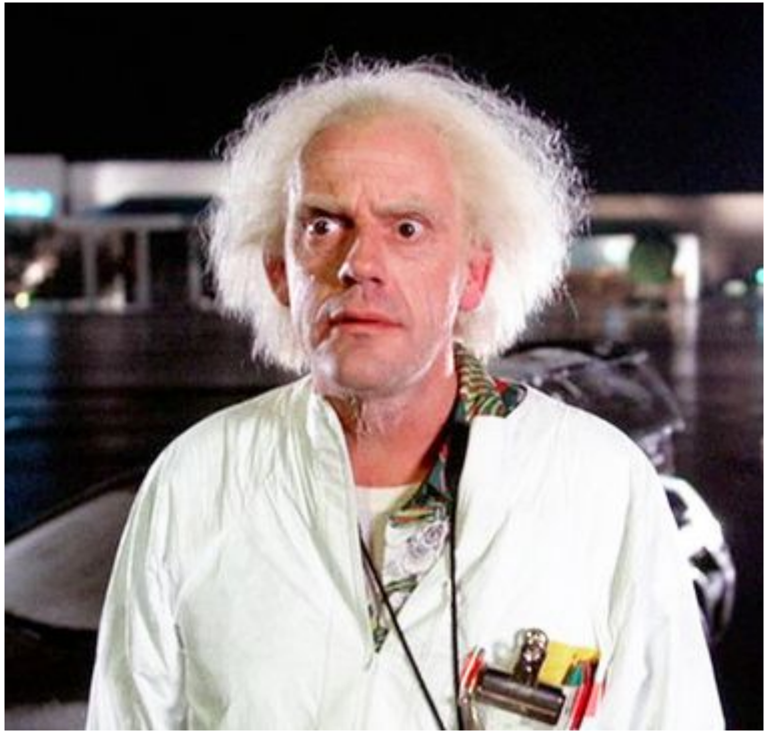
Action:

`ALTER EXTENSION UPDATE` to **add the new functions** to your system.

Serialization (aka “on disk format”)

- No new features
- No operational implications
  - Old format is still read
  - New writes use new format

- “Light point” (20 bytes!)
- Accelerated spatial joins against large geometries
- Optional sidecar index structures
- Specialized geometry compression


“But in the **future...**!”

PostgreSQL 12 “Support Functions”

PostGIS 2.5

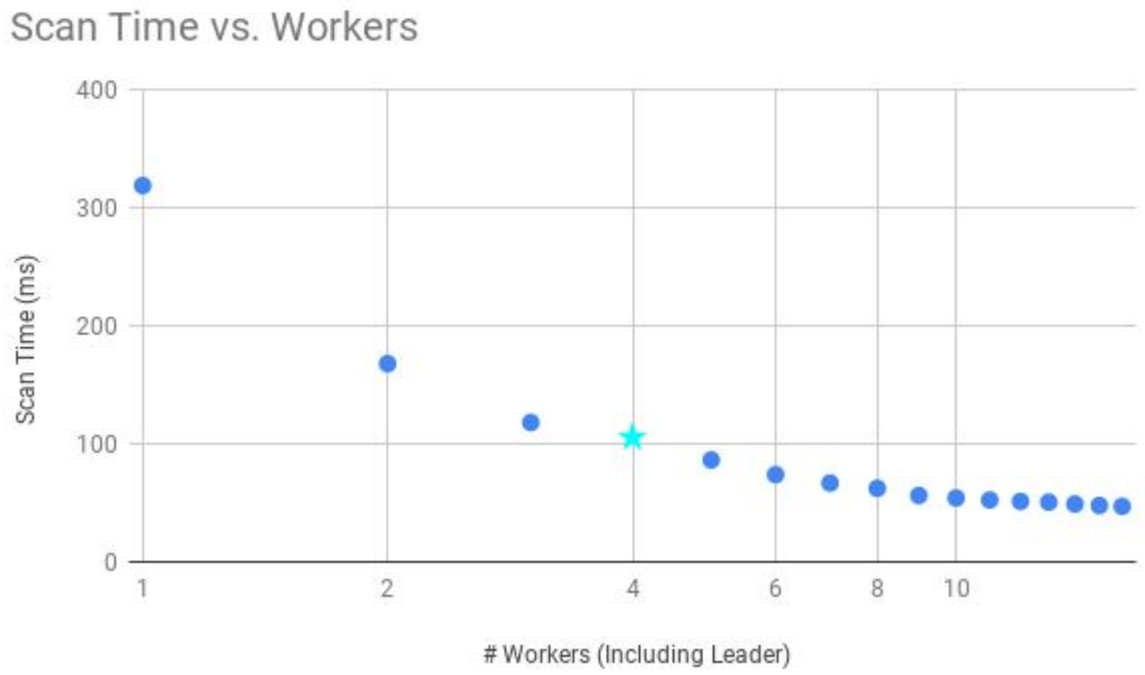
```
CREATE FUNCTION
  ST_Intersects(g1 geometry, g2 geometry)
    RETURNS boolean AS
    'SELECT $1 && $2 AND
            _ST_Intersects($1,$2)'
    LANGUAGE 'sql'
    IMMUTABLE
    PARALLEL SAFE;

CREATE FUNCTION
  _ST_Intersects(g1 geometry, g2 geometry)
    RETURNS boolean
    AS '$libdir/postgis-2.5','ST_Intersects'
    LANGUAGE 'c'
    IMMUTABLE STRICT
    PARALLEL SAFE
    COST 10000;
```

PostGIS 3.0 + PostgreSQL 12

```
CREATE FUNCTION
  ST_Intersects(g1 geometry, g2 geometry)
    RETURNS boolean
    AS '$libdir/postgis-3','ST_Intersects'
    SUPPORT postgis_index_supportfn
    LANGUAGE 'c'
    IMMUTABLE STRICT
    PARALLEL SAFE
    COST 10000;
```

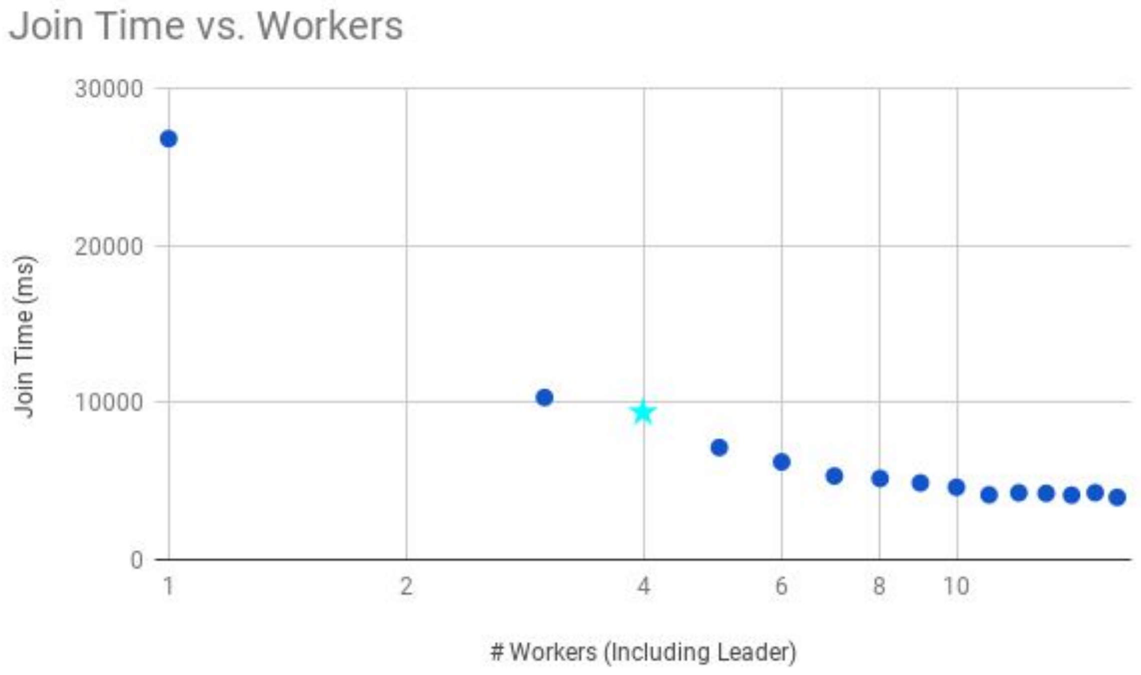
PostgreSQL 12 Parallel Spatial Scan

```
SELECT Sum(ST_Area(geom)) FROM pd;
```



PostgreSQL 12 Parallel Spatial Join

```
SELECT * FROM pd JOIN pts_10 pts
  ON ST_Intersects(pd.geom, pts.geom);
```



Smaller Changes in PostGIS 3?

Updated Support Libraries

Proj 6.0 and up

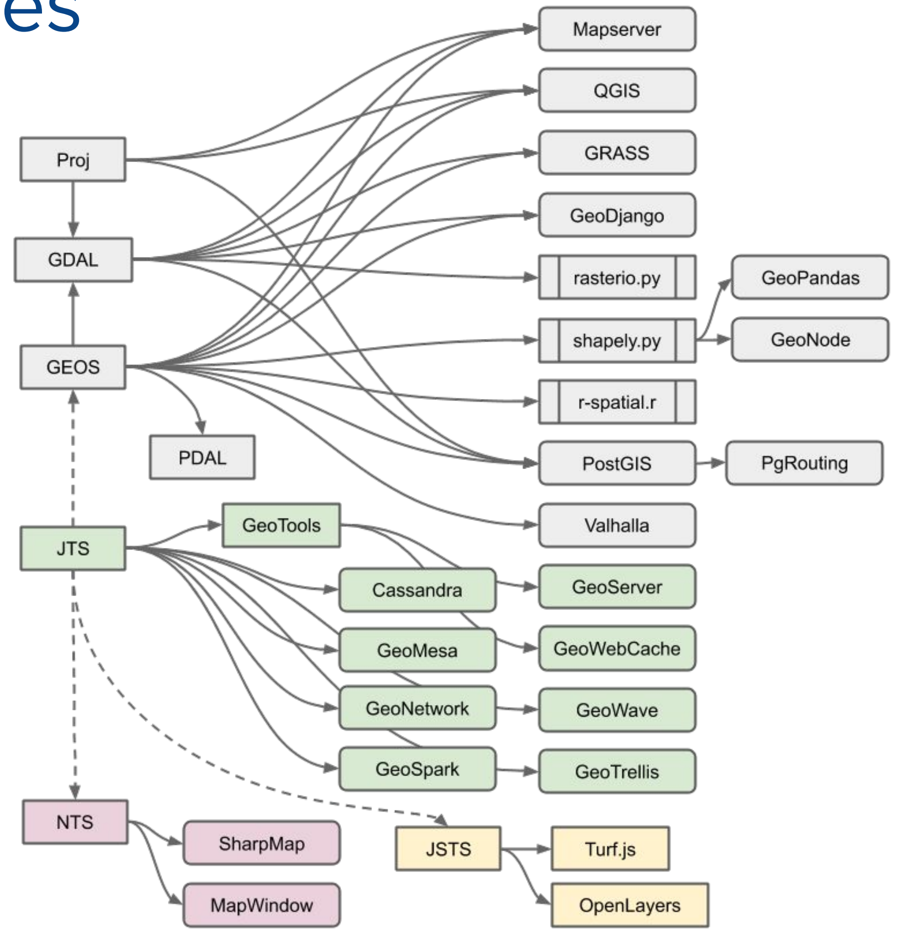
- New library API
- Old API to be deprecated with **Proj 7**
- Direct geographic transformation
  - No more pivot on WGS84
- Vertical datum transformations
  - Real problem!
- Time-dependent datums
  - Real problem!

Updated Support Libraries

GEOS 3.8

- GEOS is a port of JTS
- GEOS is infrastructure and boring to most developers
- GEOS is really important!

- **Community revitalization**
- Active maintainership
- Crunchy hires JTS developer
  - Martin Davis
- All back-logged JTS improvements ported to GEOS



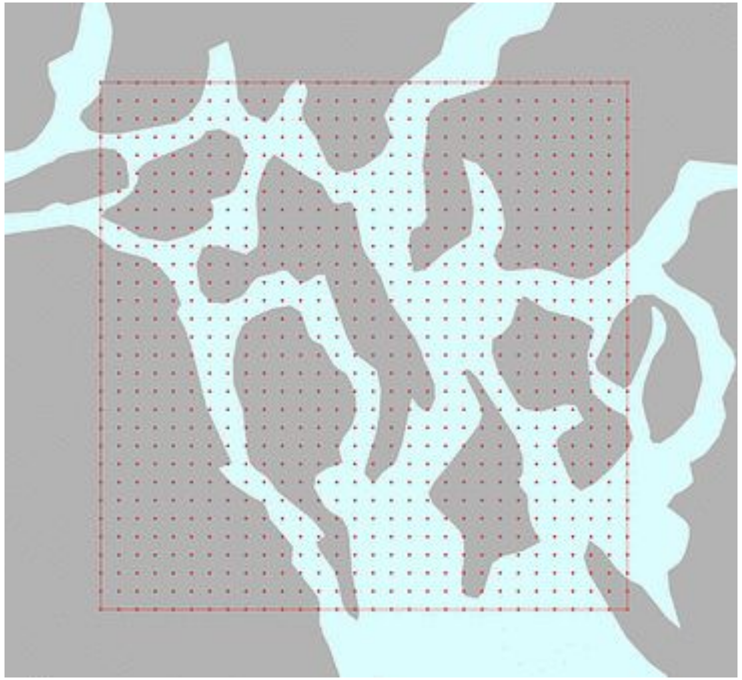
Updated Support Libraries

GEOS 3.8

- Remove Java-isms in favour of C++isms
  - Stack over heap
  - C++11 renovation
- Performance improvements
  - Profiler hot spots
  - JTS algorithmic improvements
- Have seen 30-50% improvements on some workloads
  - Buffer building

MVT Performance

- Profile and improve simplification code (points and lines)
- Replace GEOS rectangle clipping with *wagyu* implementation
    - Fixed-precision clipping routine
    - Does not produce invalid outputs
    - Validity checking can be skipped
    - GEOS 3.9 could replace *wagyu* in turn



Enhanced GeoJSON Support

GeoJSON has a **very stupid** structure for handling attributes.

GeoJSON specifies:

- Geometry ✔️
- Feature ❌
- FeatureCollection ❌

PostGIS has long had support for ST_AsGeoJSON(geometry)

Enhanced GeoJSON Support

GeoJSON has a **very stupid** structure for handling attributes.

name	**geometry**
Didagat Islands	POINT(125.6 10.1)
Discovery Islands	POINT(-123.4 48.4)

Each table row looks like a dictionary entry with two properties, right? So JSON structure is obvious...

Enhanced GeoJSON Support

GeoJSON for a “feature” should look like row_to_json() output, using the GeoJSON encoding for geometry.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
    },
  "name": "Dinagat Islands"
}
```

↑ But this is not what GeoJSON specifies!

Enhanced GeoJSON Support

GeoJSON has a **very stupid** structure for handling attributes.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

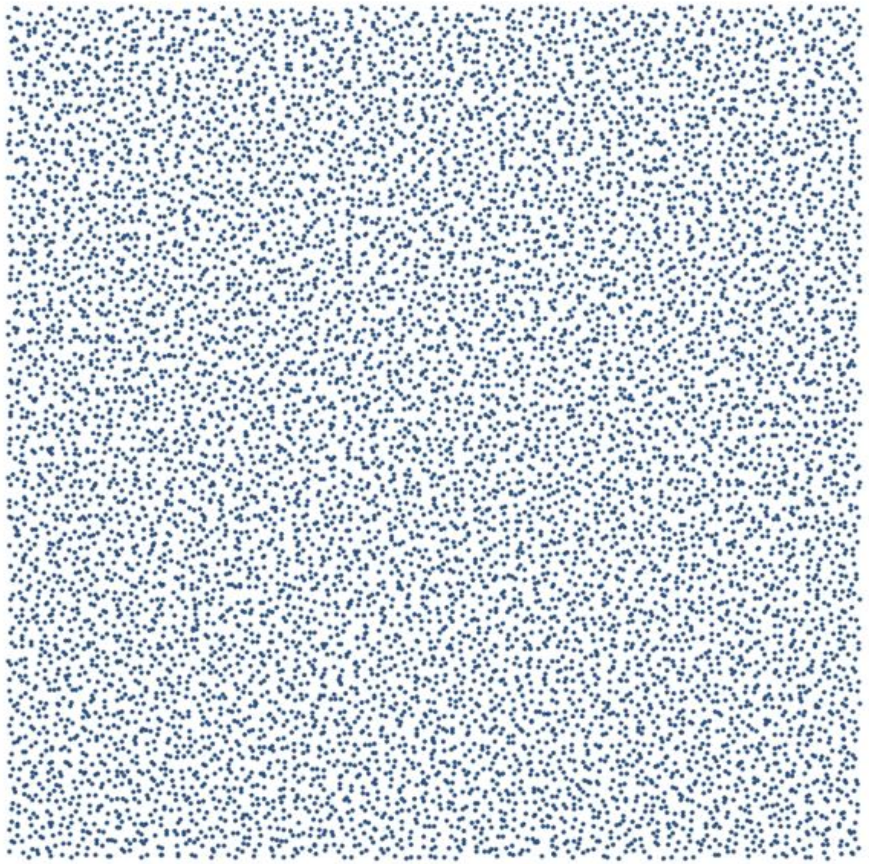
↑ It does this! (This is your brain on GIS!)

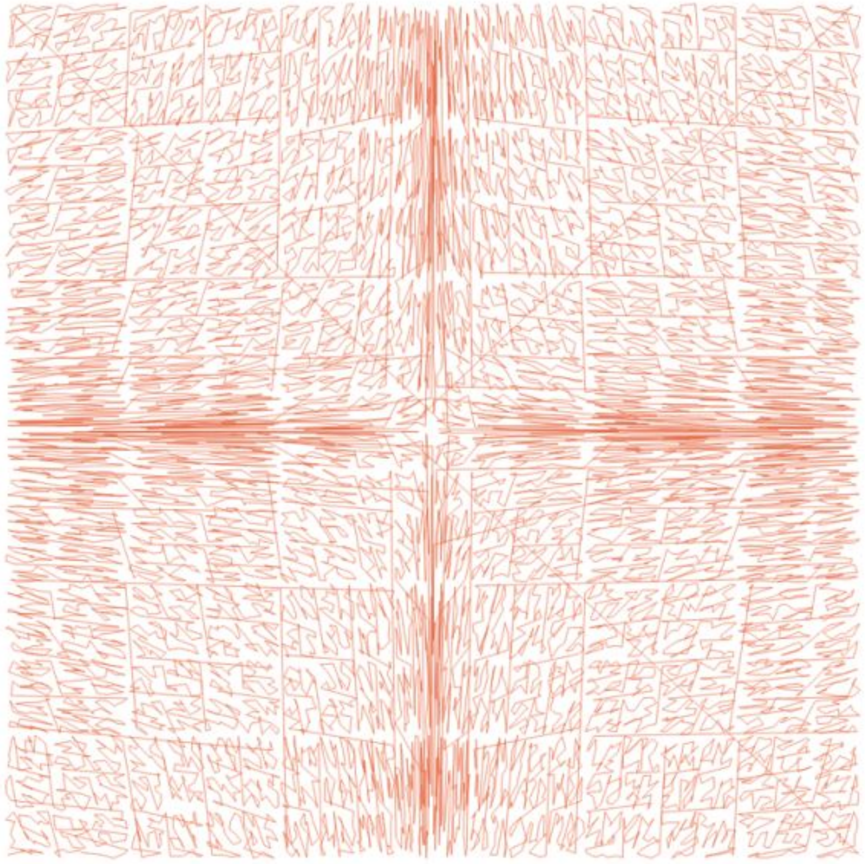
Enhanced GeoJSON Support

- geometry::json cast
- geometry::jsonb cast
  - Allows row_to_json(row) function to handle geography columns transparently
- ST_AsGeoJSON(row)
  - Returns GeoJSON “Feature”

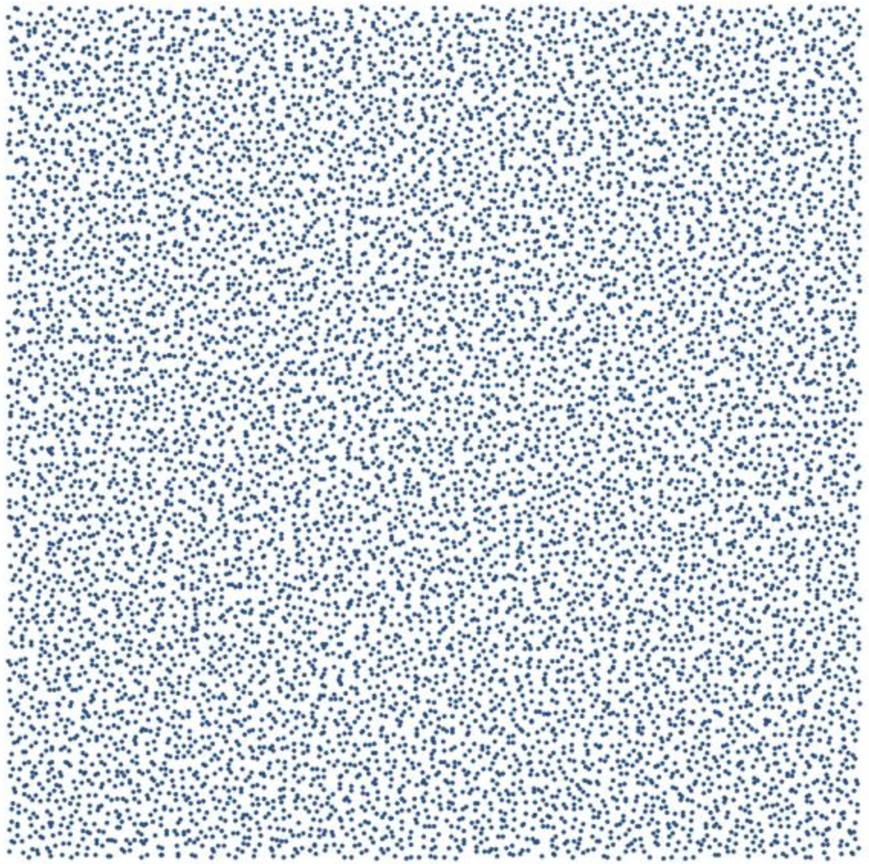
```
SELECT ST_AsGeoJSON(r.*)
FROM r
WHERE r.name = 'foobar';
```

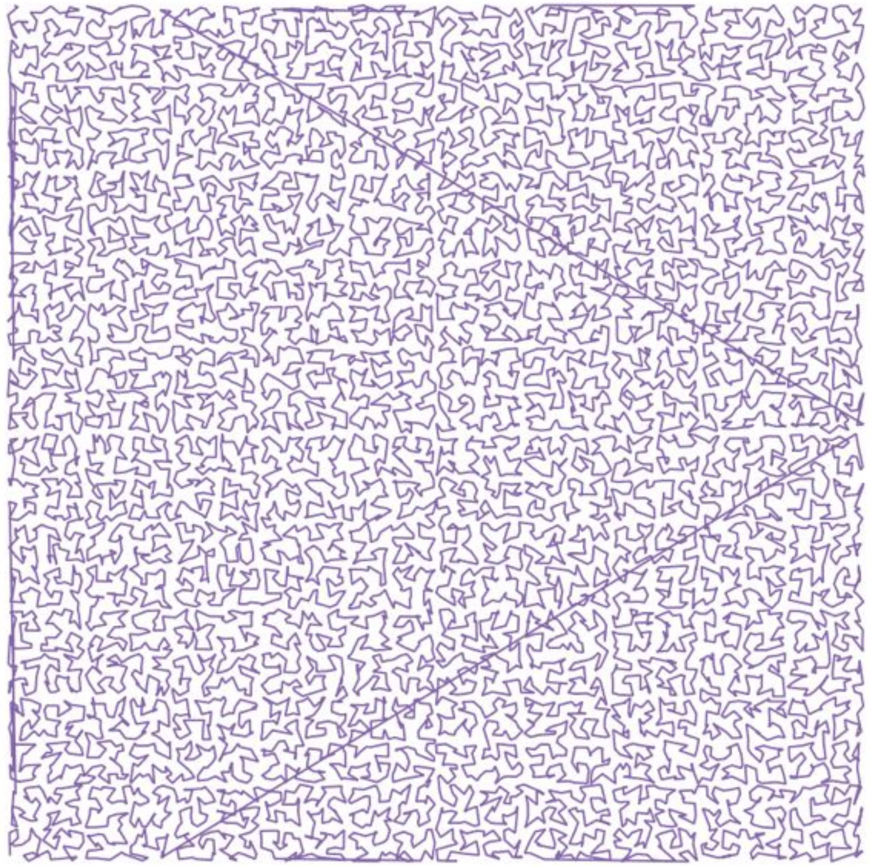
ORDER BY geometry (2.x)





ORDER BY geometry (3.x)





Dropped Functions!!!!

- **THIS NEVER HAPPENS!**
- But it’s a major version number change, so…
- ST_Accum()
  - use array_agg()
- ST_AsGeoJSON(version, geometry)
- ST_AsKML(version, geometry)
- SFCGAL bindings for
  - ST_Area, ST_Distance, ST_Intersection, ST_Difference, ST_Union, ST_Intersects

What about PostGIS 3.1?

crunchy data

PostGIS **3.1?**

- Accelerated spatial join for large objects
  - Hash key for object identity
- Surface analysis functions
  - Weighted surface (point density)
  - Kriged surface (point intensity)
  - Interpolation and contouring of surfaces
- GEOS 3.9
  - Robust overlay
  - Geometry cleaning
  - Deterministic precision reduction

Questions?

paul.ramsey@crunchydata.ca

https://is.gd/1uzKi5