I HAVE TERA-BYTES OF POINT CLOUD DATA. WHAT DO I DO NOW?

Speaker notes

Today I am going to talk about point clouds, and the Pointcloud extension for storing point clouds in PostgreSQL.

Speaker notes

This picture here represents a point cloud of the city of Lyon in France. It has multiple billions of points.


FOSS4G 2018
Dar es Salaam

ÉRIC LEMOINE

Developer @ Oslandia

FOSS4G developer and enthusiast since 2007

eric.lemoine@oslandia.com
@elemoine
@erilem

Speaker notes

My name is Éric Lemoine. I work at Oslandia. And I've been working in the FOSS4G field since 2007. I am more known for my work on OpenLayers.

OSLANDIA

https://oslandia.com/

Oslandia provides service on open-source software

- GIS
- 3D
- DATA

Speaker notes

Oslandia is an open-source company working on GIS, 3D and Data Science. QGIS, PostGIS and the iTowns 3D WebGL framework are examples of things we are working on.

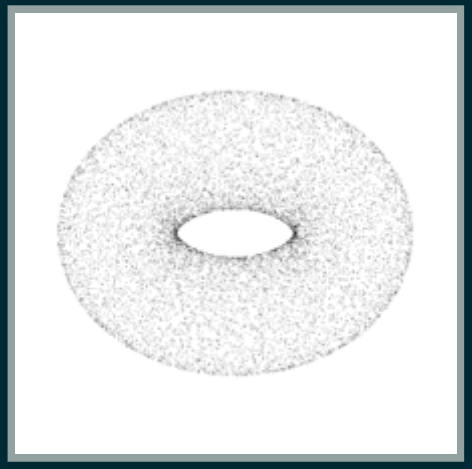
POINT CLOUDS!

Speaker notes

Let's talk about point clouds in general first!

POINT CLOUDS

« A point cloud is a set of data points in space. »



Speaker notes

A point cloud is just a set of data points in space. Nothing more. Point clouds provide a way to represent objects of our environment. A church and streets around it in the previous slide, and just a donut here.

POINT CLOUDS

- Generally produced by 3D scanners (LiDAR)
- Can also be created using Photogrammetry

Speaker notes

What can produce point clouds? Point clouds are generally produced by 3D scanners. This is the LiDAR (Light Detection And Ranging) technology. Point clouds can also be produced using photogrammetry techniques (through homolog points).

LIDAR

Terrestrial, Airborne, Mobile, Unmanned




Speaker notes

There are several types of LiDAR acquisitions: Terrestrial (fixed tripods), Airborne (planes or helicopters), Mobile (Google Car like), and Unmanned (drones).

MANY APPLICATIONS!

- Create Digital Elevation Models (DEMs)
- Create 3D models
- Detect objects and obstacles
- etc.

Speaker notes

Point clouds have a wide range of applications. Examples include creating Digital Evelation Models, Digital Surface Models, 3D models, and detecting objects and obstacles. Autonomous cars use LiDAR! For the creation of 3D models, 3D surfaces are derived from point clouds.

POINT CLOUDS IN POSTGRESQL

Speaker notes

Point clouds are typically stored in LAS files, LAS being a very common point cloud format. Here we're going to discuss storing point clouds in PostgreSQL, through the Pointcloud extension.

POINTCLOUD

"PostgreSQL extension for storing point cloud data"

https://github.com/pgpointcloud/pointcloud

Speaker notes

The Pointcloud extension allows storing point cloud data in PostgreSQL databases. Pointcloud is open-source and available on GitHub. It's easy to build and install, and it's well documented.

POINTCLOUD

- Initially developed by Paul Ramsey (funded by Natural Resources Canada)
- Recently developed / maintained by Oslandia and IGN

Speaker notes

The initial development of Pointcloud was funded by Natural Resources Canada, and done by Paul Ramsey, one of the main PostGIS developers. Recently it's been developed and maintained by Oslandia and IGN, the french institute of geography.

GOALS

- Storing LiDAR data in PostgreSQL
- Leveraging that data for analysis in PostGIS

Speaker notes

Storing LiDAR data in PostgreSQL enables all sort of analysis, by using PostGIS and Pointcloud together. For example determining all the points that are within a polygon is both a very easy and very fast operation.

LATEST RELEASES

- v1.1.0 (2018-04-31)
- v1.1.1 (2018-06-12)
- v1.2.0 (2018-08-22)

Speaker notes

The Pointcloud extension includes many PcPoint and PcPatch manipulation functions. The 1.1.0 version brings new functions useful for both analysis and visualization. The 1.1.1 version includes bug fixes. The 1.2.0 version adds new functions and mark most functions as PARALLEL SAFE, for parallel executions in PostgreSQL.

WHY NOT USE POSTGIS?

```
     Column       |       Type
------------------+------------------
 id               | integer
 geom             | geometry(PointZ)
 intensity        | double precision
 returnnumber     | double precision
 numberofreturns  | double precision
 classification   | double precision
 scananglerank    | double precision
 red              | double precision
 green            | double precision
 blue             | double precision
```

Speaker notes

By the way, why not using PostGIS instead of creating a specific extension? PostGIS has a PointZ geometry type that could be used, hasn't it?

WHY NOT USE POSTGIS?

- One point per row means billions of rows
- Does not work!

Speaker notes

Because point clouds may have billions of points, which would translate into billions of database rows, which wouldn't work.

PATCHES OF POINTS

- Organize the points into patches
- → Millions of rows instead of billions

```
 Column |    Type
--------+------------
 id     | integer
 pa     | pcpatch(1)
```

Speaker notes

For that reason Pointcloud organizes points into patches. A patch typically includes several hundreds or several thousands points, which translates into millions of rows rather than billions of rows. This is still big, but manageable.

TWO TYPES

- `PcPoint(pcid)`
- `PcPatch(pcid)`

Speaker notes

Pointcloud actually defines two new types: PcPoint and PcPatch. PcPatches are collections of PcPoints. PcPoints are packings of point dimensions (X, Y, Z, ...). Dimensions are packed in byte arrays, so in a very compact way.

USE POINTCLOUD

```
CREATE EXTENSION pointcloud;
CREATE EXTENSION postgis;             -- optional
CREATE EXTENSION pointcloud_postgis; -- optional
```

Speaker notes

Enabling Pointcloud in a database is done the same way as enabling PostGIS.

USE POINTCLOUD

```
 Schema |        Name         |   Type
--------+---------------------+----------
 public | geography_columns   | view
 public | geometry_columns    | view
 public | pointcloud_columns  | view
 public | pointcloud_formats  | table
 public | raster_columns      | view
 public | raster_overviews    | view
 public | spatial_ref_sys     | table
```

Speaker notes

After enabling Pointcloud in a database the pointcloud_colums view and the pointcloud_formats table are added to the database. The poincloud_colums view includes information about all the PcPoint and PcPatch columns that exist in the database. The pointcloud_formats table includes XML documents that define how dimensions are encoded in PcPoints.

SCHEMA

```
 pcid | srid |                                  schema
------+------+-------------------------------------------------------------------------
    1 | 4326 | <?xml version="1.0" encoding="UTF-8"?>                                 +
      |      | <pc:PointCloudSchema xmlns:pc="http://pointcloud.org/schemas/PC/1.1"   +
      |      |     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">             +
      |      |   <pc:dimension>                                                       +
      |      |     <pc:position>1</pc:position>                                       +
      |      |     <pc:size>4</pc:size>                                               +
      |      |     <pc:description>X coordinate as a long integer. You must use the   +
      |      |                     scale and offset information of the header to      +
      |      |                     determine the double value.</pc:description>       +
      |      |     <pc:name>X</pc:name>                                               +
      |      |     <pc:interpretation>int32_t</pc:interpretation>                     +
      |      |     <pc:scale>0.01</pc:scale>                                          +
      |      |   </pc:dimension>                                                      +
      |      |   <pc:dimension>                                                       +
      |      |     <pc:position>2</pc:position>                                       +
      |      |     <pc:size>4</pc:size>                                               +
      |      |     <pc:description>Y coordinate as a long integer. You must use the   +
      |      |                     scale and offset information of the header to      +
      |      |                     determine the double value.</pc:description>       +
      |      |     <pc:name>Y</pc:name>                                               +
      |      |     <pc:interpretation>int32_t</pc:interpretation>                     +
      |      |     <pc:scale>0.01</pc:scale>                                          +
      |      |   </pc:dimension>                                                      +
      |      |   <pc:dimension>                                                       +
```

Speaker notes

This is an example of an PointCloudSchema XML document. The document defines how dimensions are encoded within PCPoints.

```
SELECT pa FROM patches LIMIT 1;
--------------------------------------------------------------------
 01010000000200000009000000021000000004000000060CEFFFFBC9A78560000
(1 row)
```

Speaker notes

An SQL query that selects a patch (PcPatch) and returns a sort of WKB (Well Known Binary) string representing the patch.

```
SELECT PC_AsText(pa) FROM patches LIMIT 1;
------------------------------------------------------------------------
 {"pcid":1,"pts":[[-126.99,45.01,1,0],[-126.98,45.02,2,0],[-126.9
(1 row)
```

Speaker notes

The Pointcloud extension provides functions for manipulating points and patches. For example the `PC_AsText` function returns a JSON representation of patches.

WORKING WITH REAL DATA

Speaker notes

Let's look at how real point cloud data can be inserted into a PostgreSQL database?

PDAL

https://www.pdal.io/

Speaker notes

PDAL can be used for that! PDAL is a library and tool for converting and manipulating point cloud data. It is like GDAL but for point cloud data rather than raster and vector data.

LOAD DATA USING PDAL

1. Define a PDAL pipeline definition in JSON

```
{
"pipeline": [
    {
        "type": "readers.las",
        "filename":"arles_church.las",
        "spatialreference":"EPSG:3946"
    },
    {
        "type": "filters.chipper",
        "capacity": "400"
    },
    {
        "type":"writers.pgpointcloud",
        "connection":"dbname=lopocs host=localhost port=5432 user=elemoine",
        "schema": "public",
        "table":"arles_church",
        "compression":"none",
        "srid":"3946",
        "overwrite":"true",
        "column": "points",
        "scale_x": "0.01",
        "scale_y": "0.01",
```

Speaker notes

This creates a PDAL pipeline whose source is a LAS file and sink is a Pointcloud database table. The filter in between the source and the sink is a so-called "chipper" filter. The "chipper" filter is responsible for creating patches of points – 400-point patches here. "post_sql" is used on the writer side to create an index on the "arles_church" table.

LOAD DATA USING PDAL

2. Execute the PDAL pipeline

```
pdal pipeline pipeline-definition.json
```

Speaker notes

This executes the PDAL pipeline defined in the pipeline-definition.json file.

QUERYING

```
SELECT count(*) num_patches,
       sum(PC_NumPoints(points)) num_points
FROM arles_church;

 num_patches | num_points
-------------+------------
       45952 |   18380597
(1 row)
```

Speaker notes

The point cloud has been loaded into PostgreSQL. We can start throwing some SQL at it! The above SQL query just counts the total number of patches and points.

VISUALIZE IN QGIS

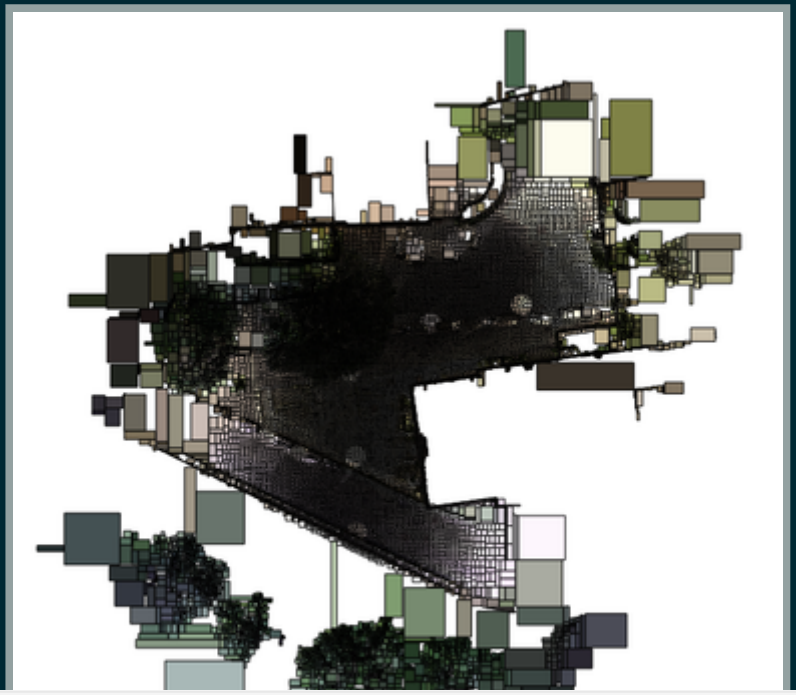
```
SELECT id, points FROM arles_church
```

Speaker notes

Point clouds in PostgreSQL can be displayed in QGIS! As a 2D viewer QGIS actually displays the 2D bounds (X/Y bounds) of patches. This is actually very useful for testing and debugging.

VISUALIZE IN QGIS

```
SELECT id, points, PC_PatchAvg(points, 'red')   || ',' ||
                   PC_PatchAvg(points, 'green') || ',' ||
                   PC_PatchAvg(points, 'blue')  || ',255' AS color
FROM arles_church;
```



Speaker notes

We can even add some colors!

VISUALIZE IN QGIS

```
SELECT id, points, PC_PatchAvg(points, 'red')   || ',' ||
                   PC_PatchAvg(points, 'green') || ',' ||
                   PC_PatchAvg(points, 'blue')  || ',255' AS color
FROM arles_church
WHERE PC_Intersects(points, ST_GeomFromText('POLYGON(…)'))
```

Speaker notes

And we can select the patches that intersect a given polygon. This query uses the geometry index that was created on the arles_church table when the PDAL pipeline was run.

LOPOCS

Light OpenSource PointCloud Server

https://github.com/Oslandia/lopocs

Speaker notes

Interestingly people often want to visualize their data! At Oslandia we created LOPoCS for that. LOPoCS is not viewer, it's a light open-source pointcloud streaming server.

LOPOCS

- Streams point cloud data stored in PostgreSQL
- Supports multiple streaming protocols (Greyhound and 3D Tiles currently supported)
- → works with Potree, Cesium, iTowns

Speaker notes

LOPoCS is able to stream point cloud data stored in PostgreSQL/Pointcloud. LOPoCS implements existing protocols. The Greyhound and 3D Tiles protocols are currently supported. This makes LOPoCS work with various point cloud web viewers, including Potree, Cesium and iTowns.

MOTIVATION

- Stream point cloud data directly from Postgres
- No export/indexing step required
- Nice for pre-visualization and prototyping

Speaker notes

The motivation is to be able to stream point cloud data directly from PostgreSQL, without having to export and index the data outside the database. In particular this is useful for pre-visualization and prototyping. For serious visualizations exporting the data to indexed files (3D Tiles for example) will always lead to better performance.

CESIUM / 3D TILES

Speaker notes

This is a Cesium application displaying a point cloud streamed by LOPoCS.

ITOWNS / 3D TILES

Speaker notes

This is an iTowns application displaying a point cloud streamed by LOPoCS.

ITOWNS

A Three.js-based framework for visualizing 3D geospatial data

https://github.com/iTowns/itowns

Speaker notes

iTowns is a Three.js-based web framework for visualizing 3D geospatial data. It is not specific to point cloud data, and can be used to represent various types of 2D and 3D data (meshes, oriented photos, etc.).

ITOWNS / 3D TILES

Speaker notes

And now a video showing up an iTowns application displaying a point cloud of the city of Lyon in France. This point cloud has multiple billions of points. The points are colored by iTowns using images from a WMS.

SUMMARY / REVIEW

- Use Pointcloud to store point clouds in PostgreSQL
- Use QGIS to visualize them in 2D
- Use LOPoCS to stream them for 3D visualization
- Use iTowns to visualize them in 3D


THANKS!