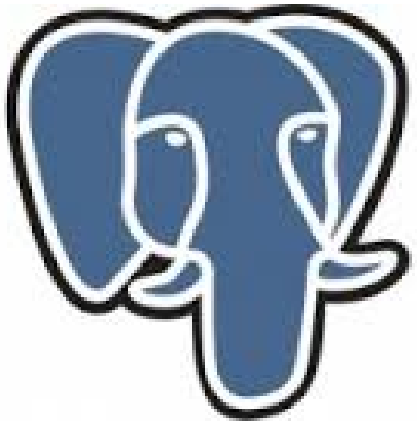
Gimme some YESQL !



- and a GIS -

Vincent Picavet – Oslandia – www.oslandia.com

PostgreSQL ?
What's that ?

« Oh, yeah, this thing bundled with PostGIS installer ! »

RODBMS

Relational
Object
Database Management System

SQL compliant
Transactionnal
ACID
MVCC
Extensible
...
Spatial !

PostgreSQL New stuff

(..or some not that old features..)

PG 9.x

PostgreSQL

- PG 9.4 beta 2

- Not long before release
- Good foundation work
- Lots of new features
- Some of them useful for GIS

SQL powah !

LATERAL JOIN

LATERAL JOIN

- Access to other tables in subqueries
- Fix the < - > operator constant issue

```
-- find 2 bars closest to each bus stop (just in case one is closed)
select
        bus.gid, bus.name, lat.gid, lat.name, lat.dist
from
        points as bus
, lateral (
        select
                bar.gid
                , st_distance(bar.geom, bus.geom) as dist
                , bar.name
        from
                points as bar
        where
                bar.type = 'bar'
        order by
                bar.geom <-> bus.geom -- forbidden without lateral
        limit 2
) as lat
where
        bus.type = 'bus_stop'
order by
        bus.gid, lat.dist desc;
```

AUTO-UPDATE VIEWS

Auto-updateable views

- 9.3+
- No trigger !
- 9 .4 : column-level management
  - Mix updateable/non updateable columns

- Fine-tuning spatial right management
- With check option

Auto-updateable views

```
create or replace view
        ways_v as
select
        *
from
        ways
where
        st_contains(st_setsrid(st_geomfromtext(
                        'polygon((4.8300 45.7562, 4.8300 45.7616, 4.8385 45.7616
                        , 4.8385 45.7562, 4.8300 45.7562))'), 4326)
                , linestring)
with check option;

-- simplify geometry
update ways_v set linestring = st_simplify(linestring, 0.00001);

-- Check option will prevent this one
update ways_v set linestring = st_translate(linestring, -10, -10);
```

MATERIALIZED VIEWS

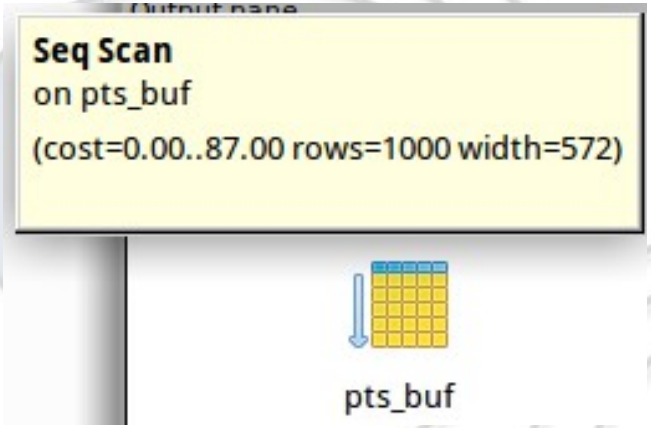
Materialized views

- 9.3+
- Actually compute the view & store results
- « Cache views »
- Allows indexing
- Indexes & Constraints NOT copied
- 9 .4 : Refresh concurrently
- Use cases
  - MV of complex joins
  - MV of complex spatial operations
  - MV of FDW tables

Materialized views

```
-- initial table
create table pts (id serial, geom geometry(Point, 2154));
--fill it
insert into
        pts (geom)
select
        st_setsrid(st_makepoint(random() * 100000, random() * 100000), 2154)
from
        generate_series(1, 1000);
```

```
-- MV for long computation
create materialized view
        pts_buf as
select
        id
        , st_buffer(geom, 10) as geom
from
        pts;

-- Data is updated
update pts set geom = st_translate(geom, 100, 100);

-- Refresh the MV
refresh materialized view pts_buf;
```



RANGES

Ranges

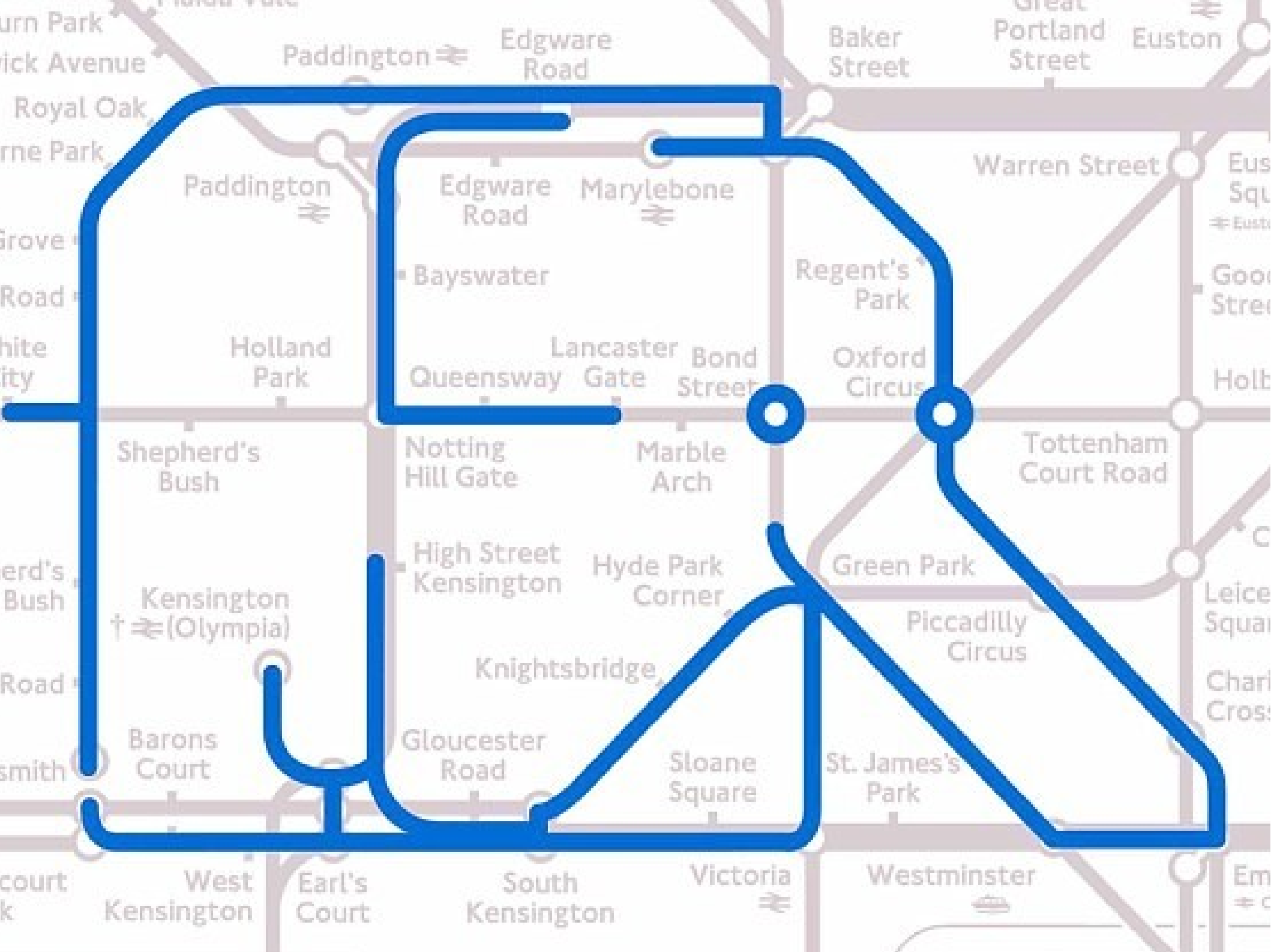
- Datatype(s)
- Integer, numeric, dates
- Open, closed, infinite
- Use indexes
- Good for time data

```
=# SELECT * from test_range ;
                        period
-------------------------------------------------------
 ["2012-01-01 00:00:00+01","2012-01-02 12:00:00+01"]
 ["2012-01-01 00:00:00+01","2012-03-01 00:00:00+01"]
 ["2008-01-01 00:00:00+01","2015-01-01 00:00:00+01"]
(3 rows)


=# SELECT * FROM test_range WHERE period && '[2012-01-03 00:00:00,2012-01-03 12:00:00]';
                        period
-------------------------------------------------------
 ["2012-01-01 00:00:00+01","2012-03-01 00:00:00+01"]
 ["2008-01-01 00:00:00+01","2015-01-01 00:00:00+01"]
(2 rows)
```

WRITABLE CTE

```
CREATE TABLE bad_data (
        id bigint geom geometry(MultiPolygon, 2154)
);

WITH deleted AS (
        DELETE FROM
                good_gis_data
        WHERE
                not st_isvalid(geom)
        RETURNING
                id, geom)
 INSERT INTO
        bad_data
SELECT
        *
FROM
        deleted;
```

RECURSIVE CTE


Paddington
Edgware Road
Baker Street
Great Portland Street
Euston
Royal Oak
Paddington
Edgware Road
Marylebone
Warren Street
Bayswater
Regent's Park
Holland Park
Lancaster Gate
Queensway
Bond Street
Oxford Circus
Shepherd's Bush
Notting Hill Gate
Marble Arch
Tottenham Court Road
High Street Kensington
Hyde Park Corner
Green Park
Kensington (Olympia)
Piccadilly Circus
Knightsbridge
Barons Court
Gloucester Road
Sloane Square
St. James's Park
West Kensington
Earl's Court
South Kensington
Victoria
Westminster

Recursive CTEs

- Common Table Expression
- SQL Standard
- « WITH RECURSIVE »

- CTE Syntax :-)
- CTE Performances /!\

- Recursive → Graph analysis features

```
CREATE RECURSIVE VIEW t(n) AS
    VALUES (1)
  UNION ALL
    SELECT n+1 FROM t WHERE n < 100;
```

```
create table
        rec_res as
with recursive
        search_graph(gid, source, depth, path, length, cycle) as (
                select
                        g.gid, g.source, 1 as depth, ARRAY[g.gid] as path
                        , cost, false as cycle
                from
                        tr as g
                where
                        gid = 31913
                union all
                select
                        g.gid
                        , g.source
                        , sg.depth + 1 as depth
                        , path || g.gid as path
                        , sg.length + g.cost as length
                        , g.gid = ANY(path) as cycle
                from
                        tr as g
                join
                        search_graph as sg
                on
                        sg.source = g.target
                where
                        not cycle
        )
select
        sg.*
        , tr.geom
from
        search_graph as sg
join
        tr
on
        sg.gid = tr.gid
limit 1000;
```

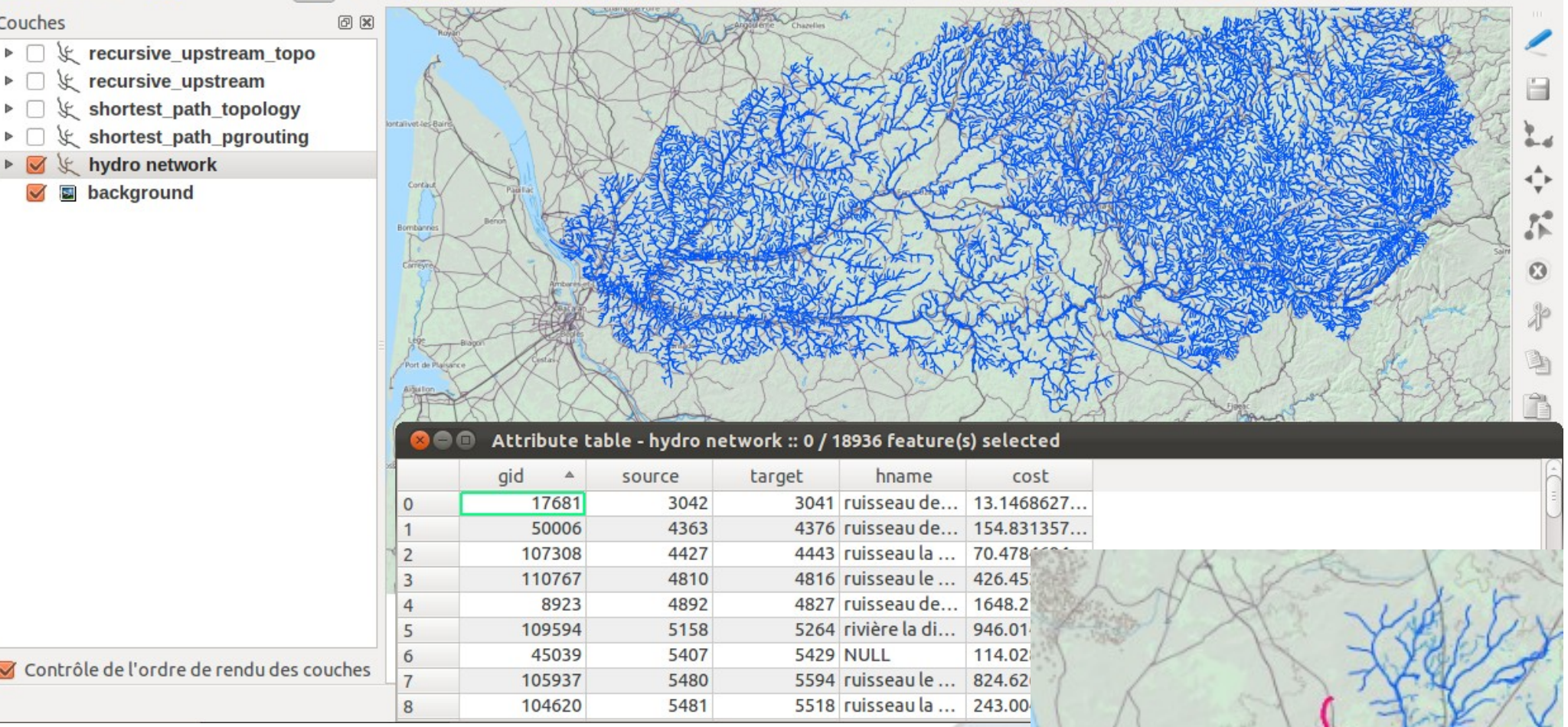
1

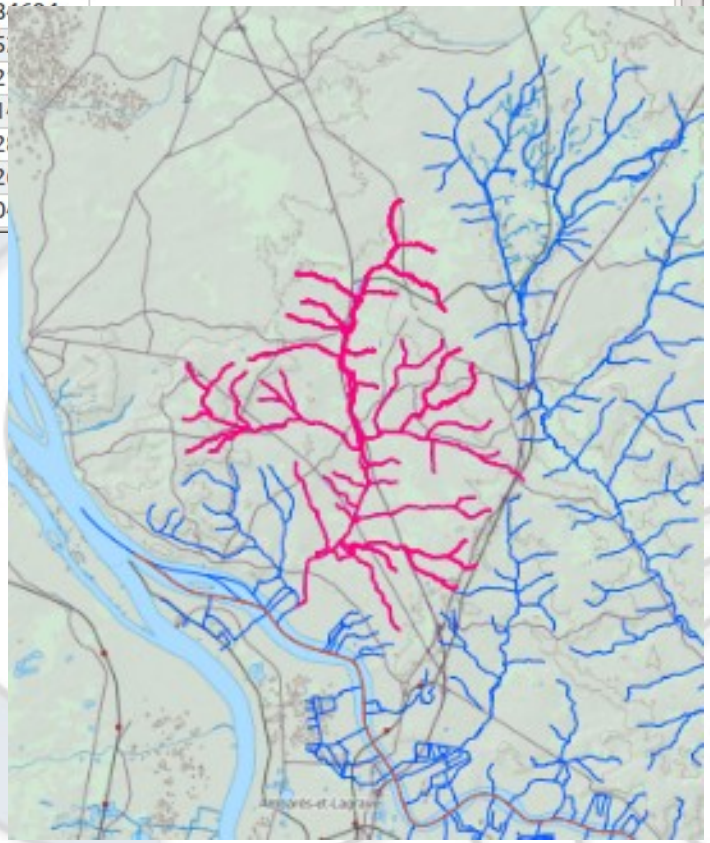
2

3

Recursive CTE

gid integer	source integer	depth integer	path integer[]	length double precision	cycle boolean	geom geometry(MultiLineString,2154)
31913	20850	1	{31913}	2666.0523017	f	01050000206A08000001000
33855	20735	2	{31913,	3473.3086319	f	01050000206A08000001000
32477	20845	2	{31913,	2725.7640259	f	01050000206A08000001000
33854	19909	3	{31913,	7183.7295195	f	01050000206A08000001000


Couches
recursive_upstream_topo
recursive_upstream
shortest_path_topology
shortest_path_pgrouting
hydro network
background
Contrôle de l'ordre de rendu des couches
Attribute table - hydro network :: 0 / 18936 feature(s) selected
gid
source
target
hname
cost
0 17681 3042 3041 ruisseau de... 13.1468627...
1 50006 4363 4376 ruisseau de... 154.831357...
2 107308 4427 4443 ruisseau la ... 70.478
3 110767 4810 4816 ruisseau le ... 426.45
4 8923 4892 4827 ruisseau de... 1648.2
5 109594 5158 5264 rivière la di... 946.01
6 45039 5407 5429 NULL 114.02
7 105937 5480 5594 ruisseau le ... 824.62
8 104620 5481 5518 ruisseau la ... 243.00



UNLOGGED TABLES

Unlogged tables

- No WAL
- No maintenance
- Faster to write to
- will not survive a crash
- will not survive a crash
- WILL NOT SURVIVE A CRASH

- Temporary results
- Reproduceable results

Unlogged tables

```
-- Unlogged table
CREATE UNLOGGED table
        testu (a int, g geometry(LineString, 2154));

-- 9.5 :
alter table testu set logged;
```

LOGICAL DECODING

Logical Decoding

- Probably biggest 9.4 feature
- First step of many
- A kind of « database modification logger»

- Open path to :
  - Logical replication
  - Multi-master replication
  - Logging, versioning, incremental updates…

Logical Decoding

```
pggis=# SELECT 'init' FROM pg_create_logical_replication_slot('oslandia', 'test_decoding');
 ?column?
----------
 init
(1 row)

pggis=# delete from points where gid = 1;
DELETE 1
pggis=# SELECT * FROM pg_logical_slot_get_changes('oslandia', null, null, 'include-xids', '0');
  location   | xid |                    data
------------+-----+-----------------------------------------------
 0/5A799658 | 931 | BEGIN
 0/5A799658 | 931 | table public.points: DELETE: gid[integer]:1
 0/5A79B6E8 | 931 | COMMIT
(3 rows)
```

```
pggis=# update points set geom = st_translate(geom, 1, 1) where gid = 2;
UPDATE 1
pggis=# SELECT * FROM pg_logical_slot_get_changes('oslandia', null, null, 'include-xids', '0');
pggis=#
 location   | xid |
------------+-----+-------------------------------------------------------------------------------
 0/5A79E108 | 933 | BEGIN
 0/5A79E108 | 933 | table public.points: INSERT: gid[integer]:999999999
        osm_id[character varying]:'99999999' "timestamp"[character varying]:''
        name[character varying]:'testrep'
        type[character varying]:'notype'
        geom[geometry]:'0101000020E6100000000000286A37B43F000000BD5B0BE53F'
 0/5A7A0B08 | 933 | COMMIT
(3 rows)
```

YESQL !

NoSQL is the future ?
What NoSQL ?

→ PostgreSQL document-oriented data support

Unstructured data,
key-pair values

« MongoDB on ACID »

~~XML~~, [hstore], JSON, JSONB

JSON

JSON

- 9.2+
- Document storage
- key order + duplicates preserved
- Indexing
- Text storage
- I/O fast, size overhead
- Operations < hstore
- Expression index
- No GIN / GIST

```
% CREATE INDEX idx_dvd_reviews
        ON reviews ((review#>>'{product,group}'));
```

JSON

- **Operators & Functions (9.3, 9.4)**

```
% SELECT '[{"a":"foo"},{"b":"bar"},{"c":"baz"}]'::json->2;
?column?
-------------
{"c":"baz"}
% SELECT '{"a": {"b":"foo"}}'::json->'a';
?column?
-------------
{"b":"foo"}
% SELECT '{"a":1,"b":2}'::json->>'b';
?column?
----------
2
% SELECT '[{"a": {"b":{"c": "foo"}}}]'::json#>'{a,b}';
?column?
-------------
{"c": "foo"}
```

```
pggis=# select to_json(points) from points limit 1;
{"gid":3,"osm_id":"124585","timestamp":"2014-04-04T08:17:23Z",
"name":"Espace Comboire Nord","type":"motorway_junctio",
"geom":"0101000020E61000009F2523C21ACA1640DE5DC2FC70934640"}
```

JSON

Other functions

- to_json()
- json_build_array()
- json_build_object()
- json_each()
- json_each_text()
- json_array_length()
- json_object_keys()
- json_array_elements()
- json_array_elements_text()
- json_typeof()
- json_to_record()

JSONB

JSONB

- 9.4+
- Binary storage
- No key duplicates & order preservation
- Slower I/O, bigger than JSON
- Hstore-like operators
- JSON operators
- .. and more : =, @>, ?, ?|, ?&
- GIN indexing

- Use it !
  - Except I/O, key preservation (JSON)
    or flat + strings (hstore)

JSONB

- GIN indexing : @> ? ?| ?&
- Fast
- Expression indexes
- jsonb_path_ops for better @>

```
CREATE INDEX idx_docs_gin
        ON docs USING gin(doc jsonb_path_ops);
```

JSONB

```
% SELECT '{"a": 1, "b": 2}'::jsonb =
        '{"b": 2, "a": 1}'::jsonb;
?column?
----------
t
% SELECT '{"a":1, "b":2}'::jsonb @>
        '{"b":2}'::jsonb;
?column?
----------
t
% SELECT '{"a":1, "b":2}'::jsonb ? 'b';
?column?
----------
t
%
% SELECT '{"a": {"b": 2, "c": 3}}'::jsonb @>
        '{"a": {"c": 3}}'::jsonb;
?column?
----------
.
% SELECT '{"a":1, "b":2, "c":3}'::jsonb ?| ARRAY['b', 'd'];
?column?
------------
t
```

GeoJSON

- **PostGIS plays too !**

```
select st_asgeojson(geom)::json from points limit 1;
                st_asgeojson
-----------------------------------------------------------
 {"type":"Point","coordinates":[5.6973677,45.1518856]}
(1 row)
```

```
select
        row_to_json(f) as feature
from (
        select
                'Feature' as type
                , st_asgeojson(geom)::json as geometry
                , '{"gid":1, "color" : "green"}'::json as properties
        from
                points
        limit 1
) as f;
```

```
{
        "type":"Feature",
        "geometry":{
                "type":"Point",
                "coordinates":[5.6973677,45.1518856]
        },
        "properties":{
                "gid":1,
                "color" : "green"
        }
}
(1 row)
```

EASY TESTING & DEPLOY

APT

- apt.postgresql.org

- Supported PostgreSQL versions
- For various apt-based distributions
- + extensions
- Maintained & Trusted

- PostGIS is in !
- Use it !

docker-pggis

Done

- Docker container
- Based on Phusion Baseimage
- All GIS-related PG packages
  - PG 9.4
  - PostGIS (+SFCGAL), PgRouting
  - PointCloud, PDAL

- Oneliner run

```
docker run --rm -P --name pggis_test oslandia/pggis /sbin/my_init
```

https://github.com/vpicavet/docker-pggis

More to come...

ORACLE SPATIAL FDW

Oracle Spatial FDW

- Oracle FDW
- Transparent access to Oracle tables
- No Oracle spatial support
  - WKT as intermediary format
  - Complex and inefficient

WIP

- Native support
- SDO → PostGIS
- Replication, cross-db queries…
- Still WIP

https://github.com/laurenz/oracle_fdw/pull/7

Oracle Spatial FDW

```
-- enable extension
create extension oracle_fdw;

-- Oracle server parameters and user mapping
create server orcl foreign data wrapper oracle_fdw options (dbserver '//10.0.0.1/FG');
grant usage on foreign server orcl to pggis;
create user mapping for pggis server orcl options (user 'PGGIS', password 'PGGISORAPASS');

-- create foreign table
CREATE FOREIGN TABLE ora_ways (
    way_id bigint,
    description varchar,
    geom geometry(LineString, 2154)
) SERVER orcl OPTIONS(schema 'MYUSER', table 'ways');

-- Use the foreign table just as another table
select * from ora_ways where st_length(geom) < 0.00003;
```

BETTER SPATIAL INDEXES

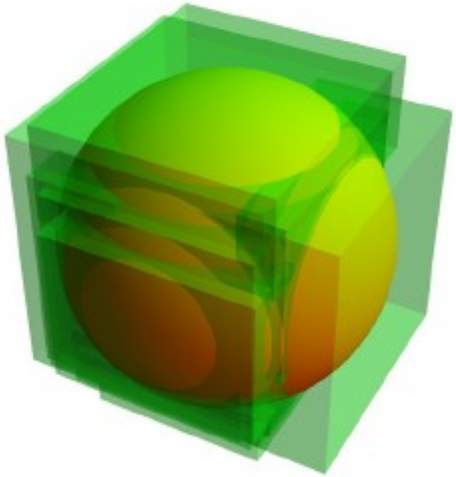
SP-Gist

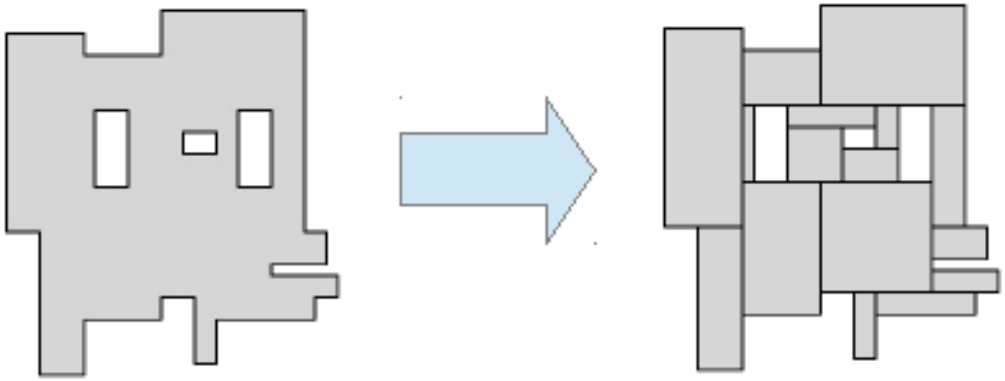
- « SP » = SPatial
- New PG index type
- Faster to read
- 3x faster to build
- Good fit for spatial data
- GSoC 2014
- Commitable code ?

VODKA

- Korotkov, Bartunov, Sigaev
- create index .. using vodka
- Derivation of JSONB indexing
- R-Tree based on GiST as entry tree
- Use multiple boxes per polygons

WIP





Beware of the spaghetti monster killing your R-Trees !

Thank you

vincent.picavet@oslandia.com

@vpicavet

www.oslandia.com

OSLANDIA