les tronçons de centralité fonctionnelle

Version stable du 260305

Tableau Drive

	Fonctionnel		Paramètres géométriques
Niveau 1	Centre Urbain	Présence continue ou presque d'activités commerciales / tertiaires / équipements publics / espaces publics	présence d'au moins 18 items pour chaque linéaire de voie de 200 mètres
Niveau 2	Centre Local	Agrégations de fonctionnalités commerciales ou tertiaires couplées à un intensité démographique INSEE Carroyé	présence d'au moins 10 items pour chaque linéaire de voie de 200 mètres
Niveau 3	Noyau de vie quotidienne	Ponctualité de services formant une centralité	présence de 8 items dans un rayon de 200 mètres
Niveau 4	Espaces Résidentiels	Restent des zones agglomérées OSM ne figurant dans aucun des 3 premiers niveaux	autres

```
DROP TABLE IF EXISTS toulon_hierarchie_fonctionnelle_v3;

CREATE TABLE toulon_hierarchie_fonctionnelle_v3 AS
WITH stats_troncons AS (
    SELECT
        f.osm_id,
        f.nom_voie,
        f.code_pattern,
        -- On récupère la longueur de la rue d'origine pour calculer la densité
        ST_Length(ST_Transform(l.geom, 2154)) as longueur_rue_m,

        -- 1. Le Filet Strict : On compte les items DANS le buffer (Filaire V3)
        (SELECT COUNT(i.osm_id)
         FROM toulon_items_nomenclature_full i
         WHERE ST_Intersects(ST_Transform(i.geom, 2154), f.geom)) as total_items_strict,

        -- 2. Le Filet Radar (Niveau 3) : On lance un radar à 75m autour du tronçon
        (SELECT COUNT(i.osm_id)
         FROM toulon_items_nomenclature_full i
         WHERE ST_DWithin(ST_Transform(i.geom, 2154), ST_Transform(l.geom, 2154), 75)) as
total_items_radar,

        f.geom
    FROM zones_motifs_filaires_v3 f
    JOIN osm_lignes l ON f.osm_id = l.osm_id
),
classification AS (
    SELECT
        osm_id, nom_voie, code_pattern,
        total_items_strict, total_items_radar,
        ROUND(longueur_rue_m::numeric) as longueur_rue_m,

        -- Calcul mathématique : Ramener la densité sur un standard de 200 mètres
        CASE WHEN longueur_rue_m > 0
             THEN ROUND(((total_items_strict::numeric / longueur_rue_m::numeric) * 200), 1)
             ELSE 0
        END as items_pour_200m,

        geom,
```

```
        -- Les règles d'or des centralités Alteris
        CASE
            WHEN (longueur_rue_m > 0 AND (total_items_strict::numeric /
longueur_rue_m::numeric) * 200 >= 18) THEN 1 -- Niveau 1
            WHEN (longueur_rue_m > 0 AND (total_items_strict::numeric /
longueur_rue_m::numeric) * 200 >= 10) THEN 2 -- Niveau 2
            WHEN total_items_radar >= 8 THEN 3 -- Niveau 3 (Pôle radar)
            ELSE 4 -- Tissu neutre
        END as niveau_fonctionnel
    FROM stats_troncons
)
-- On ne garde que les niveaux 1, 2 et 3 !
SELECT * FROM classification WHERE niveau_fonctionnel < 4;

-- Index pour la rapidité
CREATE INDEX idx_hierarchie_v3_geom ON toulon_hierarchie_fonctionnelle_v3 USING GIST(geom);
```

Voici la requête mise à jour qui intègre votre règle du rayon de 100 m pour le Niveau 3 et le nettoyage des entités (une seule commande)

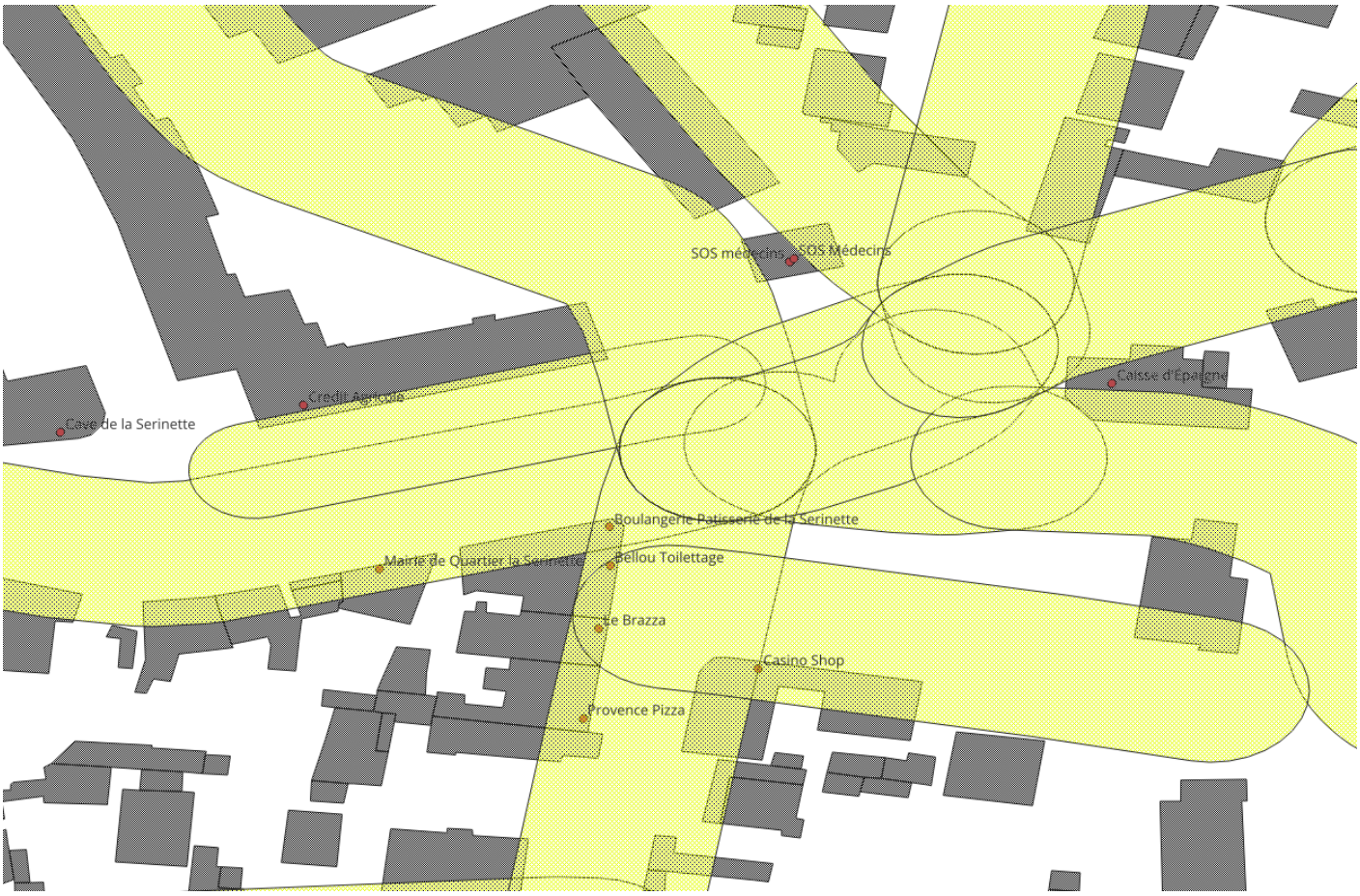
```
-- 1. Calcul de la hiérarchie avec le radar à 100m pour le Niveau 3
DROP TABLE IF EXISTS toulon_hierarchie_fonctionnelle_v2;

CREATE TABLE toulon_hierarchie_fonctionnelle_v2 AS
WITH stats_troncons AS (
    SELECT
        f.osm_id, f.nom_voie, f.code_pattern,
        ST_Length(ST_Transform(l.way, 4326)::geography) as longueur_rue_m,
        (SELECT COUNT(i.osm_id) FROM toulon_items_nomenclature_full i WHERE
ST_Intersects(ST_Transform(i.geom, 3857), f.geom)) as total_items_strict,

        -- LE NOUVEAU RADAR À 100 MÈTRES EST ICI
        (SELECT COUNT(i.osm_id) FROM toulon_items_nomenclature_full i WHERE
ST_DWithin(ST_Transform(i.geom, 3857), ST_Transform(l.way, 3857), 100)) as total_items_rayon,

        f.geom
    FROM zones_motifs_filaires_v2 f
    JOIN planet_osm_line l ON f.osm_id = l.osm_id
),
classification AS (
    SELECT
        osm_id, nom_voie, code_pattern, total_items_strict, total_items_rayon,
        ROUND(longueur_rue_m::numeric) as longueur_rue_m,
        CASE WHEN longueur_rue_m > 0 THEN ROUND(((total_items_strict::numeric /
longueur_rue_m::numeric) * 200), 1) ELSE 0 END as items_pour_200m,
        geom,
        CASE
            WHEN (longueur_rue_m > 0 AND (total_items_strict::numeric / longueur_rue_m::numeric) *
200 >= 18) THEN 1
            WHEN (longueur_rue_m > 0 AND (total_items_strict::numeric / longueur_rue_m::numeric) *
200 >= 10) THEN 2
            WHEN total_items_rayon >= 8 THEN 3
            ELSE 4
        END as niveau_fonctionnel
    FROM stats_troncons
```

```
)
SELECT * FROM classification WHERE niveau_fonctionnel < 4;

CREATE INDEX idx_hierarchie_v2_geom ON toulon_hierarchie_fonctionnelle_v2 USING GIST(geom);

-- 2. Nettoyage Topologique immédiat pour éviter les superpositions
DROP TABLE IF EXISTS toulon_hierarchie_topologique_v2;

CREATE TABLE toulon_hierarchie_topologique_v2 AS
WITH
niv1 AS (SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom FROM
toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 1),
niv2 AS (SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom FROM
toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 2),
niv3 AS (SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom FROM
toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 3),
decoupe_n2 AS (SELECT ST_Difference(n2.geom, COALESCE(n1.geom, 'GEOMETRYCOLLECTION
EMPTY'::geometry)) as geom FROM niv2 n2 CROSS JOIN niv1 n1),
decoupe_n3 AS (SELECT ST_Difference(ST_Difference(n3.geom, COALESCE(n1.geom,
'GEOMETRYCOLLECTION EMPTY'::geometry)), COALESCE(n2.geom, 'GEOMETRYCOLLECTION
EMPTY'::geometry)) as geom FROM niv3 n3 CROSS JOIN niv1 n1 CROSS JOIN niv2 n2)
SELECT 1 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM niv1 WHERE geom IS NOT
NULL AND NOT ST_IsEmpty(geom) UNION ALL
SELECT 2 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM decoupe_n2 WHERE
geom IS NOT NULL AND NOT ST_IsEmpty(geom) UNION ALL
SELECT 3 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM decoupe_n3 WHERE
geom IS NOT NULL AND NOT ST_IsEmpty(geom);

CREATE INDEX idx_topo_v2_geom ON toulon_hierarchie_topologique_v2 USING GIST(geom);
```

Seconde version SQL basée sur les filaires de seconde version - plus restrictifs

Nouvelle version qui intégre un cercle de 75 mètres

Cas d'école sur la Serinette - niveau 3 non reconnu !



Pourquoi Alteris "rate" ce pôle actuellement ?

> Votre diagnostic est 100% exact. C'est ce qu'on appelle la **"Fragmentation des carrefours"**. Dans OpenStreetMap, un carrefour complexe comme celui-ci n'est pas "une seule ligne". La rue est découpée en 4, 5 ou 6 petits tronçons qui se rejoignent au centre. Conséquence : l'algorithme a compté les items *tronçon par tronçon*. Il a trouvé 2 items sur la branche Ouest, 3 sur la branche Sud, 1 sur la branche Est... **Aucun petit segment n'a réussi à atteindre le score de 8 à lui tout seul**, donc tout le carrefour est passé à la trappe !
>
> Votre solution est la bonne : Le Rayon de Recherche (`ST_DWithin`)
>
> Votre intuition de SIGiste est brillante. Au lieu de compter aveuglément dans le "scalpel" de 30m, nous allons doter l'algorithme d'une double vision grâce à des sous-requêtes :
>
> 1. **Une vision au microscope (Le Scalpel F_D)** : Pour calculer la densité linéaire (Niveaux 1 et 2).

> 2. **Une vision radar (75 mètres)** : Chaque segment va "regarder" autour de lui à 75 mètres de distance. S'il voit 8 items dans ce cercle, il s'auto-déclare en **Niveau 3**, même si les items n'appartiennent pas strictement à son petit bout de trottoir !

```
DROP TABLE IF EXISTS toulon_hierarchie_fonctionnelle_v2;

CREATE TABLE toulon_hierarchie_fonctionnelle_v2 AS
WITH stats_troncons AS (
    SELECT
        f.osm_id,
        f.nom_voie,
        f.code_pattern,
        COUNT(i.osm_id) as total_items,
        ST_Length(ST_Transform(l.way, 4326)::geography) as longueur_rue_m,
        CASE
            WHEN ST_Length(ST_Transform(l.way, 4326)::geography) > 0
            THEN (COUNT(i.osm_id)::numeric / ST_Length(ST_Transform(l.way,
4326)::geography)::numeric) * 200
            ELSE 0
        END as items_pour_200m,
        f.geom
    FROM zones_motifs_filaires_v2 f
    JOIN planet_osm_line l ON f.osm_id = l.osm_id
    LEFT JOIN toulon_items_nomenclature_full i ON ST_Intersects(ST_Transform(i.geom, 3857),
f.geom)
    GROUP BY f.osm_id, f.nom_voie, f.code_pattern, f.geom, l.way
),
classification AS (
    SELECT
        osm_id, nom_voie, code_pattern, total_items,
        ROUND(longueur_rue_m::numeric) as longueur_rue_m,
        ROUND(items_pour_200m::numeric, 1) as items_pour_200m,
        geom,
        CASE
            -- NIVEAU 1 : Hyper-Centre Urbain (>= 18 items pour 200m)
            WHEN items_pour_200m >= 18 THEN 1

            -- NIVEAU 2 : Centre Local (>= 10 items pour 200m)
            WHEN items_pour_200m >= 10 THEN 2

            -- NIVEAU 3 : Noyau de vie quotidienne (>= 8 items ponctuels sur le tronçon)
            WHEN total_items >= 8 THEN 3

            -- NIVEAU 4 : Espaces Résidentiels (le reste)
```

```
        ELSE 4
    END as niveau_fonctionnel
  FROM stats_troncons
)
SELECT * FROM classification
WHERE niveau_fonctionnel < 4;

CREATE INDEX idx_hierarchie_v2_geom ON toulon_hierarchie_fonctionnelle_v2 USING GIST(geom);
```

Ce qui est nouveau c'est le nettoyage - pas de superposition

L'étape de Nettoyage (L'emporte-pièce V2)

Comme pour la version précédente, ces 233 tronçons se chevauchent sûrement aux carrefours. Nous devons appliquer notre requête "topologique" sur cette V2 pour que le Niveau 1 découpe le Niveau 2, etc.

Toujours sur `alteris_db=#`, lancez ce bloc :

SQL

```
DROP TABLE IF EXISTS toulon_hierarchie_topologique_v2;

CREATE TABLE toulon_hierarchie_topologique_v2 AS
WITH
niv1 AS (
    SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom
    FROM toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 1
),
niv2 AS (
    SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom
    FROM toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 2
),
niv3 AS (
    SELECT ST_Buffer(ST_Union(ST_MakeValid(geom)), 0) as geom
    FROM toulon_hierarchie_fonctionnelle_v2 WHERE niveau_fonctionnel = 3
```

```
),
decoupe_n2 AS (
    SELECT ST_Difference(n2.geom, COALESCE(n1.geom, 'GEOMETRYCOLLECTION EMPTY'::geometry)) as
geom
    FROM niv2 n2 CROSS JOIN niv1 n1
),
decoupe_n3 AS (
    SELECT ST_Difference(
              ST_Difference(n3.geom, COALESCE(n1.geom, 'GEOMETRYCOLLECTION EMPTY'::geometry)),
              COALESCE(n2.geom, 'GEOMETRYCOLLECTION EMPTY'::geometry)
           ) as geom
    FROM niv3 n3 CROSS JOIN niv1 n1 CROSS JOIN niv2 n2
)
SELECT 1 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM niv1 WHERE geom IS NOT NULL
AND NOT ST_IsEmpty(geom)
UNION ALL
SELECT 2 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM decoupe_n2 WHERE geom IS NOT
NULL AND NOT ST_IsEmpty(geom)
UNION ALL
SELECT 3 as niveau_fonctionnel, (ST_Dump(geom)).geom as geom FROM decoupe_n3 WHERE geom IS NOT
NULL AND NOT ST_IsEmpty(geom);

CREATE INDEX idx_topo_v2_geom ON toulon_hierarchie_topologique_v2 USING GIST(geom);
```

Exportation de la version finale

Une fois terminé, quittez (`\q`) et lancez l'export :

Bash

```
sudo -u postgres ogr2ogr -f "GPKG" \
    "/tmp/Toulon_Hierarchie_Propre_V2.gpkg" \
    PG:"dbname=alteris_db" \
    "toulon_hierarchie_topologique_v2" && \
sudo mv /tmp/Toulon_Hierarchie_Propre_V2.gpkg /home/debian/osm_data/ && \
sudo chown debian:debian /home/debian/osm_data/Toulon_Hierarchie_Propre_V2.gpkg
```

Première version - trop large

```
DROP TABLE IF EXISTS toulon_hierarchie_fonctionnelle;

CREATE TABLE toulon_hierarchie_fonctionnelle AS
WITH stats_troncons AS (
    SELECT
        f.osm_id,
        f.nom_voie,
        f.code_pattern,
        COUNT(i.osm_id) as total_items,
        ST_Length(ST_Transform(l.way, 4326)::geography) as longueur_rue_m,
        CASE
            WHEN ST_Length(ST_Transform(l.way, 4326)::geography) > 0
            THEN (COUNT(i.osm_id)::numeric / ST_Length(ST_Transform(l.way,
4326)::geography)::numeric) * 200
            ELSE 0
        END as items_pour_200m,
        f.geom
    FROM zones_motifs_filaires f
    JOIN planet_osm_line l ON f.osm_id = l.osm_id
    LEFT JOIN toulon_items_nomenclature_full i ON ST_Intersects(ST_Transform(i.geom, 3857),
f.geom)
    GROUP BY f.osm_id, f.nom_voie, f.code_pattern, f.geom, l.way
),
classification AS (
    SELECT
        osm_id, nom_voie, code_pattern, total_items,
        ROUND(longueur_rue_m::numeric) as longueur_rue_m,
        ROUND(items_pour_200m::numeric, 1) as items_pour_200m,
        geom,
        CASE
            -- NIVEAU 1 : Hyper-Centre Urbain (>= 18 items pour 200m)
            WHEN items_pour_200m >= 18 THEN 1

            -- NIVEAU 2 : Centre Local (>= 10 items pour 200m)
            WHEN items_pour_200m >= 10 THEN 2

            -- NIVEAU 3 : Noyau de vie quotidienne (>= 8 items ponctuels sur le tronçon)
            WHEN total_items >= 8 THEN 3
```

```
            -- NIVEAU 4 : Espaces Résidentiels (le reste)
            ELSE 4
        END as niveau_fonctionnel
    FROM stats_troncons
)
SELECT * FROM classification
WHERE niveau_fonctionnel < 4;

CREATE INDEX idx_hierarchie_geom ON toulon_hierarchie_fonctionnelle USING GIST(geom);
```

Encore une subtilité de PostGIS ! L'erreur `Only lon/lat coordinate systems are supported in geography` est très parlante quand on connaît le moteur interne de la base de données.

Explication de l'erreur : Dans la base OSM de votre VPS, les géométries (`l.way`) sont stockées nativement en "Web Mercator" (EPSG:3857, le système de Google Maps), dont les coordonnées sont déjà exprimées en mètres (et non en degrés de longitude/latitude). Or, quand on utilise la fonction `::geography` pour obtenir une distance ultra-précise (qui prend en compte la courbure de la Terre), PostgreSQL exige que la donnée source soit en Longitude/Latitude (EPSG:4326).

Il suffit donc de transformer la géométrie en 4326 *juste avant* de la passer en geography.

Voici la requête avec la petite correction magique `ST_Transform(l.way, 4326)` intégrée au calcul de longueur.

La Requête SQL (La Bonne !)

SQL

```
DROP TABLE IF EXISTS toulon_hierarchie_fonctionnelle;

CREATE TABLE toulon_hierarchie_fonctionnelle AS
WITH stats_troncons AS (
    SELECT
        f.osm_id,
        f.nom_voie,
        f.code_pattern,
        COUNT(i.osm_id) as total_items,
        -- On transforme d'abord en 4326 (lon/lat) pour que la conversion geography fonctionne
parfaitement
        ST_Length(ST_Transform(l.way, 4326)::geography) as longueur_rue_m,
        -- Calcul de la densité pour 200m
        CASE
            WHEN ST_Length(ST_Transform(l.way, 4326)::geography) > 0
            THEN (COUNT(i.osm_id)::numeric / ST_Length(ST_Transform(l.way,
4326)::geography)::numeric) * 200
            ELSE 0
        END as items_pour_200m,
        f.geom
    FROM zones_motifs_filaires f
    JOIN planet_osm_line l ON f.osm_id = l.osm_id
    LEFT JOIN toulon_items_nomenclature_full i ON ST_Intersects(ST_Transform(i.geom, 3857),
f.geom)
    GROUP BY f.osm_id, f.nom_voie, f.code_pattern, f.geom, l.way
),
classification AS (
    SELECT
        osm_id, nom_voie, code_pattern, total_items,
        ROUND(longueur_rue_m::numeric) as longueur_rue_m,
        ROUND(items_pour_200m::numeric, 1) as items_pour_200m,
        geom,
        CASE
            -- NIVEAU 1 : Centre Urbain (>= 10 items pour un linéaire de 200m)
            WHEN items_pour_200m >= 10 THEN 1

            -- NIVEAU 2 : Centre Local (>= 5 items pour un linéaire de 200m)
            WHEN items_pour_200m >= 5 THEN 2

            -- NIVEAU 3 : Noyau de vie quotidienne (5 items ou plus sur le tronçon)
```

```
            WHEN total_items >= 5 THEN 3

            -- NIVEAU 4 : Espaces Résidentiels (le reste)
            ELSE 4
        END as niveau_fonctionnel
    FROM stats_troncons
)
SELECT * FROM classification
WHERE niveau_fonctionnel < 4;

CREATE INDEX idx_hierarchie_geom ON toulon_hierarchie_fonctionnelle USING GIST(geom);
```

Exportation

Puis vous pouvez relancer l'export avec la commande que l'on connaît bien maintenant (après avoir fait `\q`) :

Bash

```
sudo -u postgres ogr2ogr -f "GPKG" \
    "/tmp/Toulon_Hierarchie_Fonctionnelle.gpkg" \
    PG:"dbname=alteris_db" \
    "toulon_hierarchie_fonctionnelle" && \
sudo mv /tmp/Toulon_Hierarchie_Fonctionnelle.gpkg /home/debian/osm_data/ && \
sudo chown debian:debian /home/debian/osm_data/Toulon_Hierarchie_Fonctionnelle.gpkg
```

Cette fois-ci, le calcul mathématique de la longueur devrait passer sans aucune erreur et générer votre cartographie des Niveaux 1, 2 et 3 !

Est-ce que la requête s'est bien exécutée cette fois ? J'ai hâte de savoir comment Toulon "réagit" à vos seuils stricts !

Revision #8
Created 2026-03-04 09:59:14 UTC by Julien
Updated 2026-03-06 14:41:47 UTC by Julien