Projet de Base de Données Spatiales
Analyse agricole et territoriale par région au Maroc

Réalisé par :

Salwa Afroukh 02

Salma Yahyaoui 58

Sous la direction de :

M. Hajji Hicham

Année universitaire 2025-2026

1 CONTENTS

1 INTRODUCTION

L'agriculture occupe une place stratégique dans l'économie du Maroc. Elle contribue de manière significative au PIB national, emploie une grande partie de la population active et structure profondément les territoires ruraux. Cependant, la diversité géographique et climatique du pays, la répartition inégale des ressources en eau et des infrastructures, ainsi que la spécialisation régionale des filières agricoles, rendent nécessaire une analyse spatiale fine pour mieux comprendre et valoriser le potentiel agricole de chaque région.

Aujourd'hui, le Ministère de l'Agriculture publie des informations régionales riches (superficies agricoles, surfaces irriguables, cheptel, infrastructures de formation, etc.), mais ces données restent souvent fragmentées, non structurées et difficilement exploitables pour des analyses croisées. En parallèle, de nombreuses données spatiales ouvertes (réseaux routiers, hydrographie, limites administratives) sont disponibles sur des plateformes publiques telles qu'OpenStreetMap ou HydroSHEDS.

Dans ce contexte, ce projet vise à concevoir et implémenter une base de données spatiale intégrée permettant de rassembler, structurer et analyser ces différentes sources d'information. Grâce à l'utilisation de PostgreSQL/PostGIS, il s'agira de rendre possible des requêtes spatiales et thématiques utiles à la planification agricole, à la gestion du territoire et à la prise de décision publique.

2 PROBLEMATIQUE

Comment mettre en place une base de données spatiale cohérente et interopérable permettant de croiser des informations agricoles, géographiques et infrastructurelles pour analyser les dynamiques territoriales régionales au Maroc ?

Plus précisément :

- Comment intégrer des données hétérogènes (tableaux descriptifs, couches géographiques, indicateurs économiques) dans un même système spatial cohérent ?
- Quels modèles conceptuels et relationnels adopter pour représenter les entités agricoles et territoriales de manière efficace et normalisée ?
- Comment exploiter les fonctions spatiales de PostGIS pour répondre à des questions concrètes telles que la densité routière, l'accessibilité des zones agricoles ou la répartition des infrastructures de formation ?
- En quoi les indexations spatiales (R-tree, R-tree)* peuvent-elles optimiser les performances des requêtes géographiques sur de grands volumes de données ?
- Et enfin, dans quelle mesure des solutions émergentes comme DuckDB, grâce à sa rapidité en traitement analytique et son intégration avec des formats modernes (Parquet, GeoParquet, etc.), pourraient-elles constituer une alternative ou un complément à PostGIS dans le cadre d'analyses spatiales régionales ?

3 SOURCES DE DONNEES ET ETAPES DE PREPARATION.

3.1 SOURCES DE DONNEES

Donnée	Source	Description / Utilisation
Limites administratives des régions	https://sig-maroc.com/	Délimitation des 12 régions administratives du Maroc. Sert de couche de base pour toutes les analyses spatiales.
Cours d'eau	HydroSHEDS	Fournit le réseau hydrographique à l'échelle du continent africain. Les données ont été découpées (clip) selon les limites régionales du Maroc.
Etablissements de formation	https://download.geofabrik.de/africa/morocco.html	Contient les établissements (scolaires, hôteliers, agricoles…).
Réseau routier (routes et autoroutes)	https://www.geodatika.com/routes	Données vectorielles du réseau routier national : routes principales, autoroutes, pistes secondaires. Utilisées pour les calculs de densité et de proximité.
Données agricoles régionales	https://www.agriculture.gov.ma/fr	Indicateurs régionaux : superficie agricole utile (SAU), superficie irrigable, cheptel, etc. Données extraites depuis les pages régionales du site.

3.2 ÉTAPES DE PREPARATION DES DONNEES

1. Téléchargement et organisation des données

Les différentes couches spatiales ont été téléchargées aux formats Shapefile , puis organisées.

2. Extraction et découpage des cours d'eau (HydroSHEDS)

La couche HydroSHEDS couvre tout le continent africain. Une opération de clip spatial a été effectuée avec la couche des régions du Maroc pour ne conserver que les cours d'eau situés sur le territoire national.

Résultat : une couche « hydrographie_maroc » adaptée à notre échelle d'analyse.

3. Collecte des données agricoles (site du Ministère)

Les indicateurs régionaux (SAU, superficie irriguable, cheptel, etc.) ont été extraits manuellement à partir des pages du site du Ministère de l'Agriculture.

Une tentative de web scraping a été réalisée pour automatiser la collecte, mais les résultats n'ont été que partiellement exploitables en raison de la structure HTML variable des pages.

Les données validées ont été saisies dans une table Excel structurée par région.

4. Intégration dans la base de données

Toutes les couches spatiales (shapefiles) ont été importées dans PostgreSQL à l'aide de pgShapeLoader.

La table Excel contenant les indicateurs agricoles a été importée comme table non spatiale.

Une jointure entre la table Excel et la couche des limites administratives a été effectuée (via le code de la région) pour produire une table enrichie nommée regions, intégrant à la fois les géométries et les données descriptives.

5. Création des index spatiaux GiST pour chaque table.

-Exemple :

```
CREATE INDEX gist
  ON cours_d_eau
  USING gist
  (geom);
```

-Utilité :

Afin d'optimiser les performances des requêtes spatiales dans la base de données,on crée des index spatiaux de type GiST (Generalized Search Tree) sur les colonnes géométriques principales (geom) des tables.

Ces index reposent sur une structure de type R-tree, adaptée aux données spatiales.

Le principe est de stocker, pour chaque entité, un rectangle englobant minimal (MBR) et de les organiser de manière hiérarchique pour faciliter les recherches spatiales.

Grâce à ces index :

- Les requêtes telles que la recherche des routes intersectant une région, ou les établissements situés à moins de 5 km d'une autoroute, sont exécutées beaucoup plus rapidement.
- PostGIS peut exclure rapidement les objets non pertinents (ceux dont le MBR ne croise pas la zone de recherche), avant de calculer précisément les géométries réelles.
- Sur des jeux de données volumineux (plusieurs milliers d'objets), le temps d'exécution des requêtes est réduit de plusieurs ordres de grandeur.
- Ainsi, les index GiST permettent de rendre la base performante et scalable, condition essentielle pour des analyses spatiales à l'échelle régionale ou nationale.

4 SCHEMA DE BASE DE DONNEES ET JUSTIFICATION.

4.1 SCHEMA GENERAL

Table	Type de géométrie	Champs principaux	Description
regions	MULTIPOLYGON (EPSG:4326)	gid, nom_region, sau, superficie_irrigable, cheptel_bovins, etc.	Contient les limites administratives régionales et les indicateurs agricoles (SAU, cheptel, irrigation). C'est la table de référence pour les analyses spatiales.
roads	MULTILINESTRING (EPSG:4326)	gid, fclass, name, ref, maxspeed, etc	Représente le réseau routier national (routes principales, secondaires, autoroutes).
cours_d_eau	MULTILINESTRING (EPSG:4326)	gid, hyriv_id, length_km, dis_av_cms, etc	Contient le réseau hydrographique extrait d'HydroSHEDS, découpé selon les régions du Maroc.
gis_osm_pois_a_free_1	MULTIPOLYGON (EPSG:4326)	gid, fclass, name, etc	Regroupe les points d'intérêt extraits d'OpenStreetMap, notamment les établissements de formation.

4.2 JUSTIFICATION :

Séparation thématique claire :
Chaque table représente une thématique géographique spécifique (administration, transport, hydrologie, services).

Structure adaptée aux analyses spatiales :

- regions → unité spatiale d'agrégation.
- roads / cours_d_eau → couches linéaires servant à calculer des **indicateurs de densité**.
- gis_osm_pois_a_free_1 → couche polygonale pour des **comptages ou analyses de proximité**.

5 RESULTATS DES REQUETES

5.1 REQUETE 1 :

- Longueur totale du réseau routier par région.

- Requête :

```
SELECT
```

```
    r.nom_region,
    SUM(ST_Length(ST_Intersection(ro.geom, r.geom)::geography) / 1000) AS
longueur_routes_km
FROM
    regions AS r
JOIN
    roads AS ro ON ST_Intersects(r.geom, ro.geom)
GROUP BY
    r.nom_region
ORDER BY
    longueur_routes_km DESC;
```

-Explication :

- ST_Intersects(r.geom, ro.geom): Vérifie si la géométrie d'une route intersecte la géométrie d'une région.
- ST_Intersection(ro.geom, r.geom): Calcule la portion de la route qui se trouve à l'intérieur de la région. C'est important car une route peut traverser plusieurs régions.
- ST_Length(...::geography) / 1000: Calcule la longueur de la géométrie résultante en mètres (par défaut pour geography), puis la convertit en kilomètres.
- GROUP BY r.nom_region: Regroupe les résultats par le nom de la région pour sommer les longueurs.

-Résultat :

	nom_region	longueur_routes_km
1	Région de Marrakech-Safi	69073.83162479027
2	Région de Casablanca-Settat	54251.677127468545
3	Région de Souss-Massa	53795.08625221068
4	Région de Drâa-Tafilalet	47085.536324252134
5	Région de Fès-Meknès	43282.15780393782
6	Région de l'Oriental	42292.364353206445
7	Région de Béni Mellal-Khénifra	36924.34110872824

Table

nom_region longueur_routes_km

Région de Marrakech-Safi 69073.83162479027

Région de Casablanca-Settat 54251.677127468545

Région de Souss-Massa 53795.08625221068

Région de Drâa-Tafilalet 47085.536324252134

Région de Fès-Meknès 43282.15780393782

Région de l'Oriental 42292.364353206445

Région de Béni Mellal-Khénifra 36924.34110873824

Région de Rabat-Salé-Kénitra 33749.215210407616

Région de Tanger-Tétouan-Al Hoceima 31047.2460629283

Région de Laâyoune-Sakia El Hamra 21501.348850546063

Région de Guelmim-Oued Noun 18512.183965716522

Région de Dakhla-Oued Ed-Dahab 8557.01329510078

5.2 REQUETE2 :

- Densité routière (km de routes par 100 km²) pour chaque région.

-Requête :

```
WITH longueur_routes AS (
    SELECT
        r.nom_region,
        SUM(ST_Length(ST_Intersection(ro.geom, r.geom)::geography)) / 1000 AS
longueur_km,
        ST_Area(r.geom::geography) / 1000000 AS superficie_km2
    FROM
        regions AS r
    JOIN
        roads AS ro
        ON r.geom && ro.geom                -- Filtrage rapide par bounding box
        AND ST_Intersects(r.geom, ro.geom)
    GROUP BY
        r.nom_region, r.geom
)
SELECT
    nom_region,
    longueur_km,
    superficie_km2,
    (longueur_km / superficie_km2) * 100 AS densite_routiere_par_100km2
FROM
```

```
    longueur_routes
ORDER BY
    densite_routiere_par_100km2 DESC;
```

-Explication :

- ST_Area(r.geom::geography) / 1000000: Calcule la superficie de la région en mètres carrés, puis la convertit en kilomètres carrés (1 km² = 1 000 000 m²).
- La formule de densité est appliquée pour obtenir le ratio par 100 km².
- GROUP BY r.nom_region, r.geom: Il est nécessaire de regrouper par r.geom également car ST_Area est une fonction agrégée si elle n'est pas dans le GROUP BY et la géométrie de la région est unique pour chaque nom de région.

-Résultat :

Exécuter | 12 lignes, 971.248 secondes | Créer une vue | Effacer

	nom_region	longueur_km	superficie_km2	ensite_routiere_par_100km
1	Région de Casablanca-Settat	54251.67712816803	20295.889862421947	267.30376197308584
2	Région de Tanger-Tétouan-Al Hoceima	31047.246063235463	16163.758862527331	192.07936920670556
3	Région de Rabat-Salé-Kénitra	33749.21521079873	17650.843846638258	191.20454242320278
4	Région de Marrakech-Safi	69073.83162579437	38988.93456838201	177.16265496982737
5	Région de Béni Mellal-Khénifra	36924.34110925215	27654.297107932278	133.52117020056488
6	Région de Fès-Meknès	43282.157804448754	38858.45992925593	111.38413072274717
7	Région de Souss-Massa	53795.086253133806	53630.675685544826	100.30656068656114
8	Région de l'Oriental	42292.36435369626	66100.30588959141	63.98210081559682
9	Région de Drâa-Tafilalet	47085.53632500675	85961.03414511976	54.77544191187459
10	Région de Guelmim-Oued Noun	18512.183966081575	44622.9943057624	41.48575023727397
11	Région de Laâyoune-Sakia El Hamra	21501.348851062492	143451.79271039303	14.988553607322558
12	Région de Dakhla-Oued Ed-Dahab	8557.013295356823	129256.17211382477	6.620197051651351

5.3 REQUETE 3 :

-Établissements de formation situés à moins de 5 km d'une autoroute.

-Requête :

```
SELECT
DISTINCT p.name AS nom_etablissement,p.fclass AS type_etablissement,p.geom
FROM
    gis_osm_pois_a_free_1 AS p JOIN roads AS r ON ST_DWithin(p.geom::geography,
r.geom::geography, 5000)
WHERE p.fclass IN ('university') AND r.fclass = 'motorway';
```

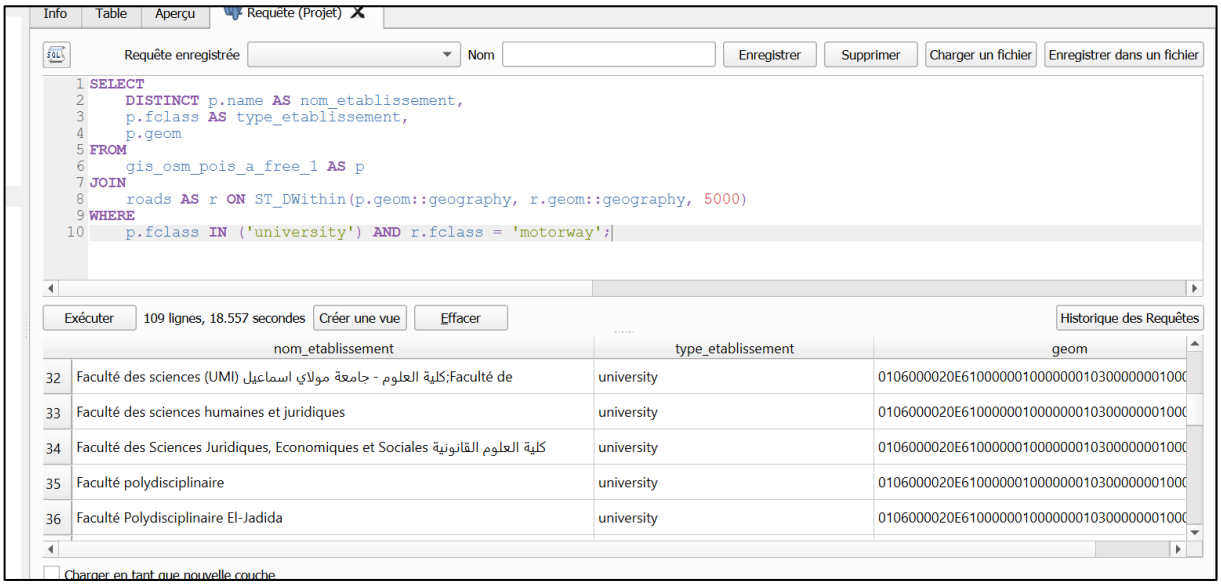
-Explication :

- p.fclass IN ('university'): Filtre les points d'intérêt pour inclure uniquement les établissements de formation.
- r.fclass = 'motorway': Filtre les routes pour inclure uniquement les autoroutes.

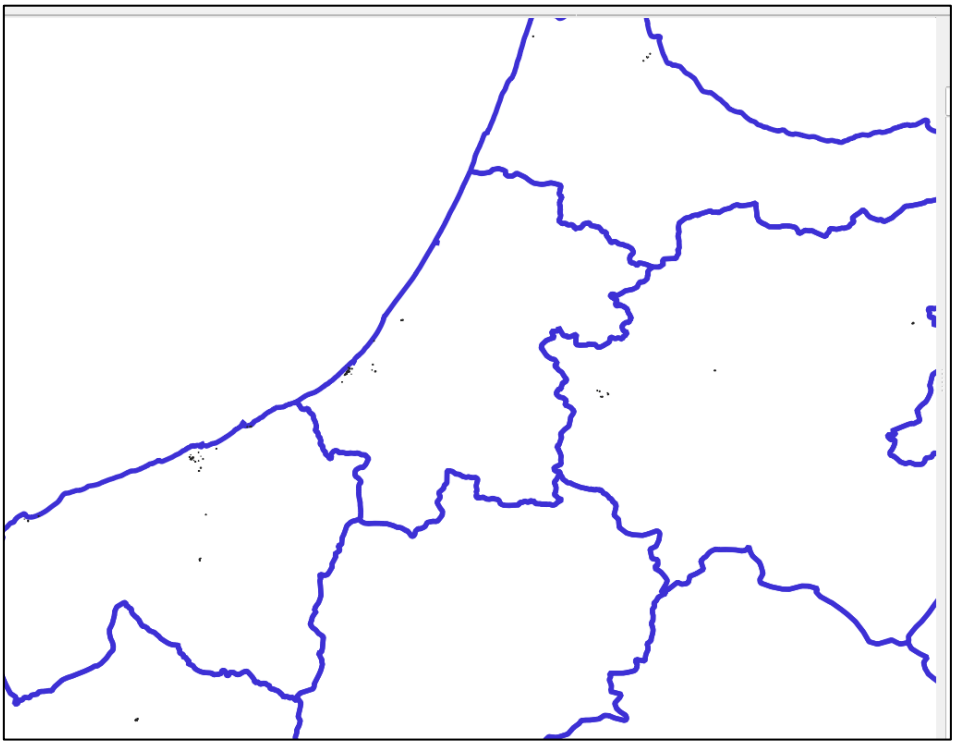
- ST_DWithin(p.geom::geography, r.geom::geography, 5000): Cette fonction spatiale vérifie si la distance entre la géométrie d'un établissement (p.geom) et la géométrie d'une route (r.geom) est inférieure ou égale à 5000 mètres (5 km). L'utilisation de ::geography permet des calculs de distance précis sur la surface terrestre.
- DISTINCT p.name, p.fclass: Assure que chaque établissement est listé une seule fois, même s'il est proche de plusieurs tronçons d'autoroute.

-Résultat :

Tableau :



Carte :



5.4 REQUETE 4 :

- Longueur des cours d'eau traversant les régions les plus agricoles (selon SAU).

-Requête :

```
WITH TopAgriRegions AS (
    SELECT nom_region,geom,SAU
    FROM regions
    ORDER BY SAU DESC
    LIMIT 5 -- On sélectionne les 5 régions les plus agricoles
)
SELECT tar.nom_region,tar.geom,
SUM(ST_Length(ST_Intersection(c.geom, tar.geom)::geography) / 1000) AS
longueur_cours_eau_km
FROM TopAgriRegions AS tar
JOIN
    cours_d_eau AS c ON ST_Intersects(tar.geom, c.geom)
GROUP BY tar.nom_region,tar.geom
ORDER BY longueur_cours_eau_km DESC;
```

-Explication :

- WITH TopAgriRegions AS (...): Une Common Table Expression (CTE) est utilisée pour d'abord identifier les régions avec la plus grande SAU. LIMIT 3 sélectionne les trois premières.
- ST_Intersects(tar.geom, c.geom): Joint les cours d'eau avec ces régions.
- ST_Intersection(c.geom, tar.geom): Calcule la partie du cours d'eau qui se trouve dans la région.
- ST_Length(...::geography) / 1000: Calcule la longueur de ces segments de cours d'eau en kilomètres.
- GROUP BY tar.nom_region: Somme les longueurs par région.

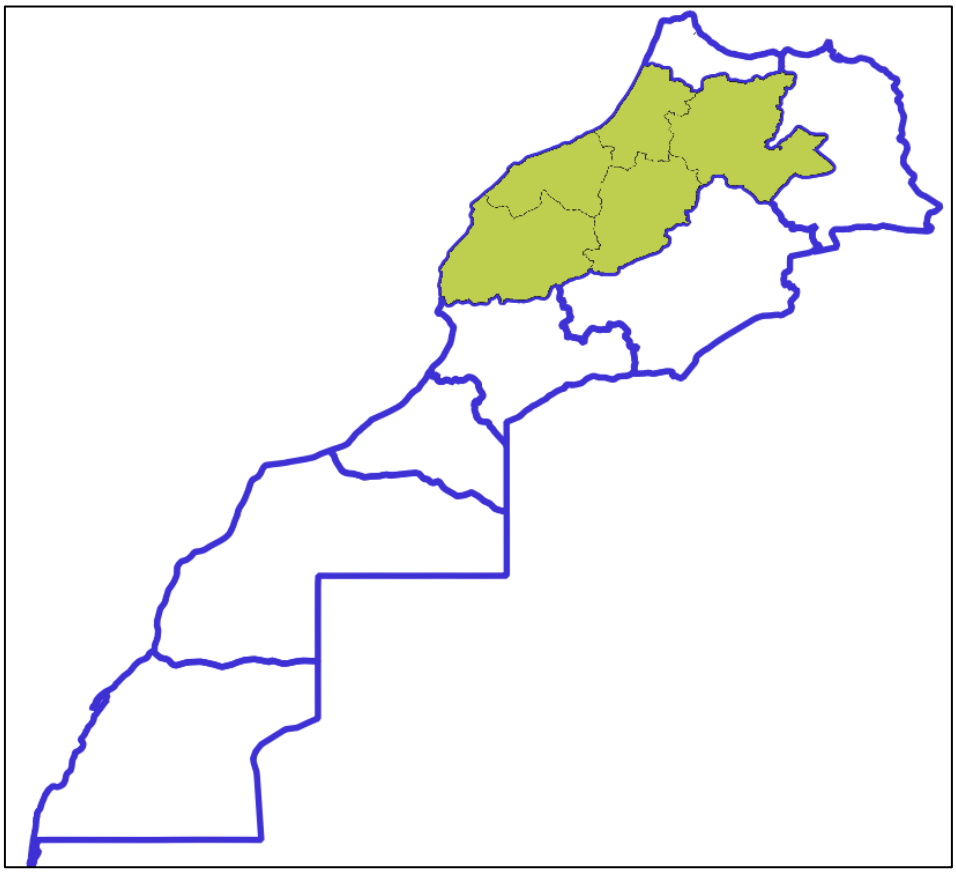
-Résultat :

Table :

Exécuter | 5 lignes, 85.778 secondes | Créer une vue | Effacer

	nom_region	geom	longueur_cours_eau_km
1	Région de Marrakech-Safi	0106000020E61...	9734.766194410748
2	Région de Fès-Meknès	0106000020E61...	9430.17112853662
3	Région de Béni Mellal-Khénifra	0106000020E61...	6599.259978715173
4	Région de Casablanca-Settat	0106000020E61...	5024.355754291521
5	Région de Rabat-Salé-Kénitra	0106000020E61...	4298.9319405235965

carte :



5.5 REQUETE 5 :

- Analyse croisée entre superficie irrigable et densité routière.

-Requête :

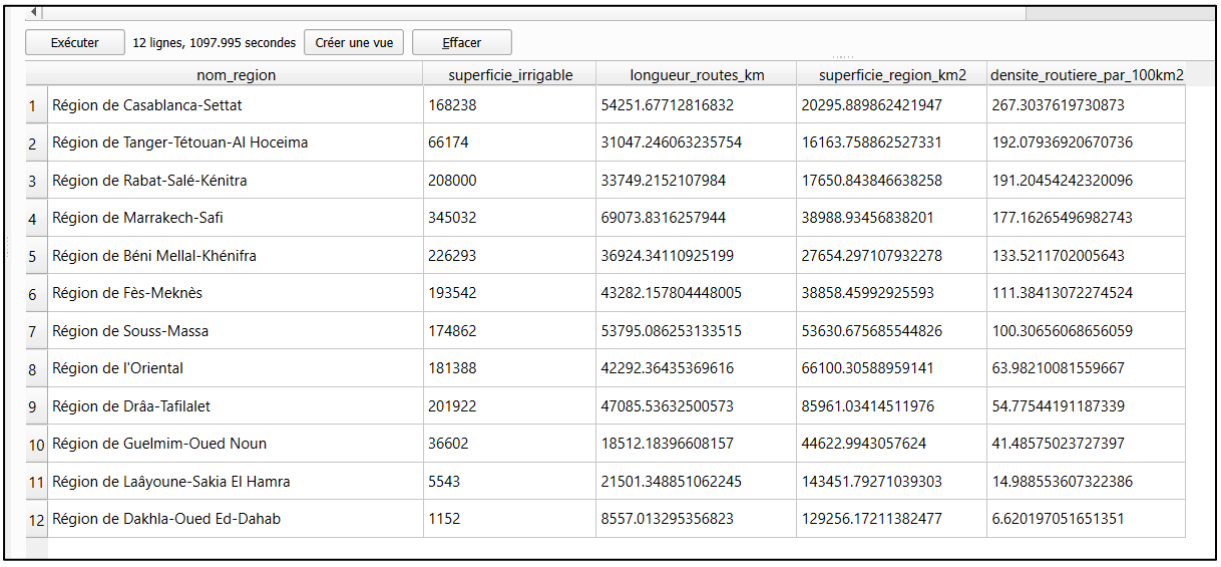
```
SELECT
    r.nom_region,
    r.superficie_irrigable,
    SUM(ST_Length(ST_Intersection(ro.geom, r.geom)::geography) / 1000) AS
longueur_routes_km,
    (ST_Area(r.geom::geography) / 1000000) AS superficie_region_km2,
    (SUM(ST_Length(ST_Intersection(ro.geom, r.geom)::geography) / 1000) /
     (ST_Area(r.geom::geography) / 1000000)) * 100 AS
densite_routiere_par_100km2
FROM
    regions AS r
JOIN
    roads AS ro
```

```
    ON ro.geom && r.geom -- on a utilisé un filtre de Bounding box afin de pouvoir accelere le temps d'exécution.
    AND ST_Intersects(ro.geom, r.geom)
GROUP BY
    r.nom_region, r.superficie_irrigable, r.geom
ORDER BY
    densite_routiere_par_100km2 DESC;
```

-Explicatif :

- Cette requête est très similaire à la deuxième, mais elle ajoute la colonne r.superficie_irrigable dans la sélection et dans le GROUP BY.
- Elle permet de visualiser et d'analyser côte à côte la capacité d'irrigation d'une région et sa densité de réseau routier, ce qui pourrait être utile pour des études de développement agricole et d'accessibilité.

Résultat



Exécuter | 12 lignes, 1097.995 secondes | Créer une vue | Effacer

	nom_region	superficie_irrigable	longueur_routes_km	superficie_region_km2	densite_routiere_par_100km2
1	Région de Casablanca-Settat	168238	54251.67712816832	20295.889862421947	267.3037619730873
2	Région de Tanger-Tétouan-Al Hoceima	66174	31047.246063235754	16163.758862527331	192.07936920670736
3	Région de Rabat-Salé-Kénitra	208000	33749.2152107984	17650.843846638258	191.20454242320096
4	Région de Marrakech-Safi	345032	69073.8316257944	38988.93456838201	177.16265496982743
5	Région de Béni Mellal-Khénifra	226293	36924.34110925199	27654.297107932278	133.5211702005643
6	Région de Fès-Meknès	193542	43282.157804448005	38858.45992925593	111.38413072274524
7	Région de Souss-Massa	174862	53795.086253133515	53630.675685544826	100.30656068656059
8	Région de l'Oriental	181388	42292.36435369616	66100.30588959141	63.98210081559667
9	Région de Drâa-Tafilalet	201922	47085.53632500573	85961.03414511976	54.77544191187339
10	Région de Guelmim-Oued Noun	36602	18512.18396608157	44622.9943057624	41.48575023727397
11	Région de Laâyoune-Sakia El Hamra	5543	21501.348851062245	143451.79271039303	14.988553607322386
12	Région de Dakhla-Oued Ed-Dahab	1152	8557.013295356823	129256.17211382477	6.620197051651351

5.6 REQUETE 6 :

- Proximité des POIs et des Routes par Région

-Requête :

```
SELECT
    r.nom_region,
    COUNT(DISTINCT p.osm_id) AS nombre_pois_proches_routes
FROM
    regions AS r
JOIN
```

```
    gis_osm_pois_a_free_1 AS p
    ON r.geom && p.geom       -- filtre rapide sur bounding box
    AND ST_Intersects(r.geom, p.geom)  -- intersection exacte
JOIN
    roads AS ro
    ON ro.geom && ST_Expand(p.geom, 0.001)  -- bbox étendue autour du POI (~100m)
    AND ST_DWithin(p.geom::geography, ro.geom::geography, 100)
GROUP BY
    r.nom_region
ORDER BY
    nombre_pois_proches_routes DESC;
```

-Explication :

- p.geom::geography et ro.geom::geography : C'est la modification clé. L'opérateur ::geography "transtype" (convertit à la volée) les colonnes de géométrie en type geography.
- ST_DWithin(..., ..., 100) : Lorsque ST_DWithin est utilisé avec des objets de type geography, le paramètre de distance (ici 100) est automatiquement interprété en mètres. Cela rend la requête plus simple et plus précise qu'une reprojection manuelle. La base de données calcule la distance réelle sur la surface de la Terre.
- ST_Intersects(r.geom, p.geom) : Pour la simple vérification de savoir si un point est à l'intérieur d'un polygone, il est souvent plus performant de rester sur le type geometry car les calculs sont plus simples. On ne fait la conversion en geography que lorsque la précision métrique est nécessaire.

-Resultat :

Exécuter | 12 lignes, 396.473 secondes | Créer une vue | Effacer

	nom_region	nombre_pois_proches_routes
1	Région de Marrakech-Safi	4490
2	Région de Casablanca-Settat	3885
3	Région de Rabat-Salé-Kénitra	2996
4	Région de Souss-Massa	2059
5	Région de Fès-Meknès	1731
6	Région de Tanger-Tétouan-Al Hoceima	1503
7	Région de l'Oriental	1465
8	Région de Drâa-Tafilalet	1048
9	Région de Béni Mellal-Khénifra	401
10	Région de Laâyoune-Sakia El Hamra	290
11	Région de Guelmim-Oued Noun	286
12	Région de Dakhla-Oued Ed-Dahab	118

5.7 REQUETE 7 :

- Cours d'eau Significatifs, Irrigation et Bétai

-Requête :

```
SELECT
   r.nom_region,
   r.superficie_irrigable,
   (r.cheptel_bovins + r.cheptel_ovins + r.cheptel_caprins + r.cheptel_camelins)
AS cheptel_total,
   SUM(
      ST_Length(
         ST_Intersection(c.geom, r.geom)::geography
      )
   ) / 1000 AS longueur_cours_eau_km
FROM
   regions AS r
JOIN
   cours_d_eau AS c ON ST_Intersects(r.geom, c.geom)
WHERE
   c.dis_av_cms > 0.1 -- Filtre crucial pour ne garder que les cours d'eau avec un
débit
GROUP BY
   r.nom_region, r.superficie_irrigable, r.cheptel_bovins, r.cheptel_ovins,
r.cheptel_caprins, r.cheptel_camelins
ORDER BY
   longueur_cours_eau_km DESC;
```

-Explication :

- WHERE c.dis_av_cms > 0.1 : C'est la clause la plus importante de cette requête. Elle filtre les données pour ne conserver que les cours d'eau ayant un débit moyen (dis_av_cms) supérieur à 0.1 m^3/s. Cela permet d'exclure les oueds (rivières à sec) et de ne mesurer que les ressources en eau réellement disponibles, rendant l'analyse géographiquement pertinente.
- ST_Intersection(c.geom, r.geom) : Calcule la géométrie exacte du segment de cours d'eau qui se trouve à l'intérieur des frontières d'une région.
- SUM(ST_Length(...)) / 1000 : ST_Length calcule la longueur de ce segment en mètres (car les données sont projetées en UTM). SUM additionne les longueurs de tous les segments de cours d'eau pertinents dans une région, puis le tout est divisé par 1000 pour obtenir des kilomètres.

- GROUP BY r.nom_region, ... : Regroupe les résultats par région pour permettre l'agrégation par SUM. Toutes les colonnes non agrégées dans le SELECT doivent apparaître dans le GROUP BY.
  -Résultat :

Exécuter | 12 lignes, 9.958 secondes | Créer une vue | Effacer

	nom_region	superficie_irrigable	cheptel_total	igueur_cours_eau_l
1	Région de Drâa-Tafilalet	201922	2186318	10217.1256095...
2	Région de Souss-Massa	174862	2633800	6856.31939543...
3	Région de l'Oriental	181388	4283400	6776.19631117...
4	Région de Fès-Meknès	193542	3840270	6460.66866332...
5	Région de Dakhla-Oued Ed-Dahab	1152	110000	5387.64767494...
6	Région de Marrakech-Safi	345032	5064000	5364.30722797...
7	Région de Laâyoune-Sakia El Hamra	5543	613000	5196.34791154...
8	Région de Béni Mellal-Khénifra	226293	4115000	4002.80180777...
9	Région de Guelmim-Oued Noun	36602	490697	3618.06310875...
10	Région de Rabat-Salé-Kénitra	208000	2698827	3498.23582011...
11	Région de Tanger-Tétouan-Al Hoceima	66174	1626180	2949.92024863...
12	Région de Casablanca-Settat	168238	3416934	2219.64036285...

5.8 REQUETE 8 :

- Indice de Développement Agricole Routier

-Requête :

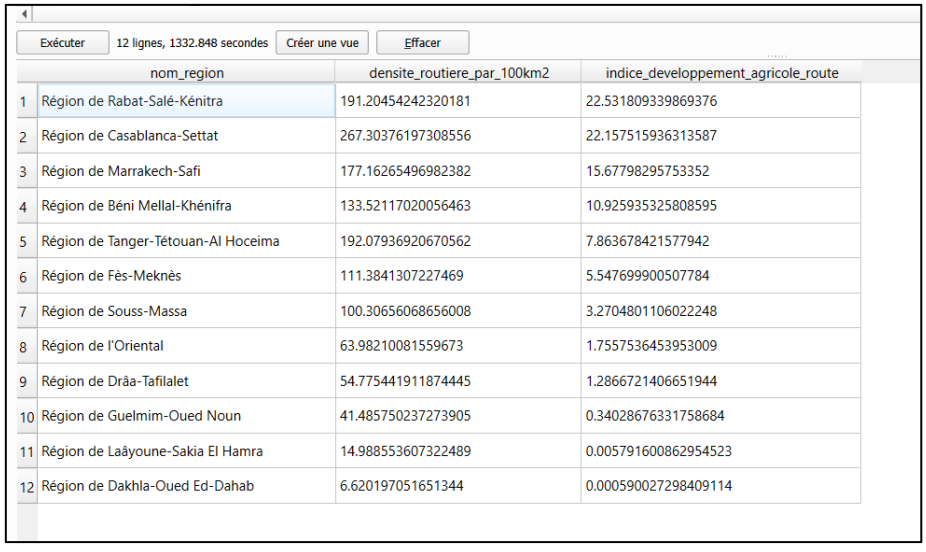
```
WITH longueur_routes AS (
    SELECT
        r.nom_region,
        r.superficie_irrigable,
        SUM(
            ST_Length(
                ST_Intersection(ro.geom, r.geom)::geography
            )
        ) / 1000 AS longueur_km,
        ST_Area(r.geom::geography) AS superficie_m2
    FROM regions r
    JOIN roads ro
        ON r.geom && ro.geom
```

```
        AND ST_Intersects(r.geom, ro.geom)
    GROUP BY r.nom_region, r.superficie_irrigable, r.geom
)
SELECT
    nom_region,
    (longueur_km / (superficie_m2 / 1000000) * 100) AS
densite_routiere_par_100km2,
    (longueur_km / (superficie_m2 / 1000000) * 100) *
    ((CAST(superficie_irrigable AS DOUBLE PRECISION) * 10000.0) /
CAST(superficie_m2 AS DOUBLE PRECISION))
        AS indice_developpement_agricole_route
FROM longueur_routes
ORDER BY indice_developpement_agricole_route DESC;
```

-Explicatif :

- ST_Length(ST_Intersection(...)::geography) : Comme pour la requête 2, on calcule la longueur précise des routes en mètres.
- ST_Area(r.geom::geography) : C'est ici que le type geography montre toute sa puissance. En appliquant ST_Area à une région convertie en geography, la fonction retourne sa **superficie en mètres carrés** calculée sur le globe.
- **Calcul de la Densité Routière** : La formule (SUM(...) / 1000) / (ST_Area(...) / 1000000) * 100 devient extrêmement fiable. On divise des **kilomètres** réels par des **kilomètres carrés** réels, sans aucune approximation liée à une projection.
- **Calcul de la Proportion Irrigable** : De même, la division de la superficie irrigable (convertie en m^2) par ST_Area(r.geom::geography) donne un ratio très précis.
- GROUP BY ..., r.geom : Il est nécessaire d'inclure r.geom dans le GROUP BY car la fonction ST_Area est appliquée directement à cette colonne avant l'agrégation par SUM.

-Résultat :



Exécuter | 12 lignes, 1332.848 secondes | Créer une vue | Effacer

	nom_region	densite_routiere_par_100km2	indice_developpement_agricole_route
1	Région de Rabat-Salé-Kénitra	191.20454242320181	22.531809339869376
2	Région de Casablanca-Settat	267.30376197308556	22.157515936313587
3	Région de Marrakech-Safi	177.16265496982382	15.67798295753352
4	Région de Béni Mellal-Khénifra	133.52117020056463	10.925935325808595
5	Région de Tanger-Tétouan-Al Hoceima	192.07936920670562	7.863678421577942
6	Région de Fès-Meknès	111.3841307227469	5.547699900507784
7	Région de Souss-Massa	100.30656068656008	3.2704801106022248
8	Région de l'Oriental	63.98210081559673	1.7557536453953009
9	Région de Drâa-Tafilalet	54.775441911874445	1.2866721406651944
10	Région de Guelmim-Oued Noun	41.485750237273905	0.34028676331758684
11	Région de Laâyoune-Sakia El Hamra	14.988553607322489	0.005791600862954523
12	Région de Dakhla-Oued Ed-Dahab	6.620197051651344	0.000590027298409114

6 ANALYSE DES INDEX SPATIAUX (R-TREE, R*-TREE).

6.1 INDEXATION R-TREE :

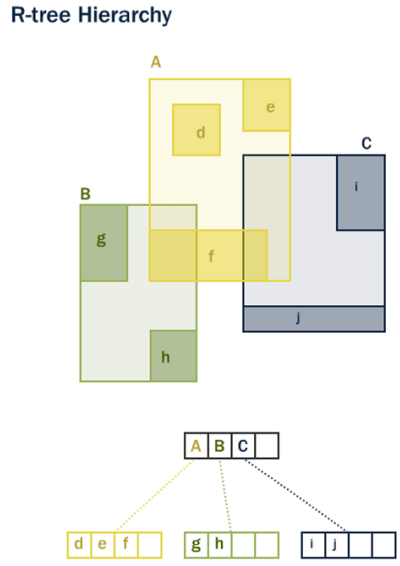


Figure : R-tree indexing visualized from PostGIS Workshops (postgis.net) - Introduction to PostGIS - 15. Spatial Indexing

Les R-arbres sont des structures de données sous forme d'arbre utilisées comme méthodes d'exploration spatiale. Elles servent à indexer des informations multidimensionnelles (coordonnées géographiques, rectangles ou polygones).

Les étapes suivantes liées à la construction et à l'utilisation d'un **index spatial de type R-tree** :

- Concept **du rectangle englobant minimal (MBR)**
  - Chaque objet spatial (point, ligne, polygone) est associé à un MBR (bbox)
  - Les nœuds internes de l'arbre contiennent des MBR couvrant les enfants
- Insertion **successive d'objets**
  - On descend l'arbre pour trouver la feuille la plus appropriée (celle où l'augmentation du MBR est minimale)
  - On insère l'objet dans la feuille correspondante
- Dépassement **de capacité dans un nœud feuille**
  - Quand le nombre d'entrées dépasse M, un **split** est déclenché
- Répartition **des entrées restantes**

- Pour chaque entrée restante, on évalue l'augmentation d'aire si on l'ajoute à groupe A vs groupe B

- On l'affecte au groupe minimisant cette augmentation, en respectant la contrainte m (minimum)

• **Propagation vers les niveaux supérieurs**

- Si le nœud parent a un 'overflow' en recevant le nouveau pointeur, on applique aussi split interne
- On remonte jusqu'à la racine, éventuellement en créant une nouvelle racine.

• **Recherche spatiale**

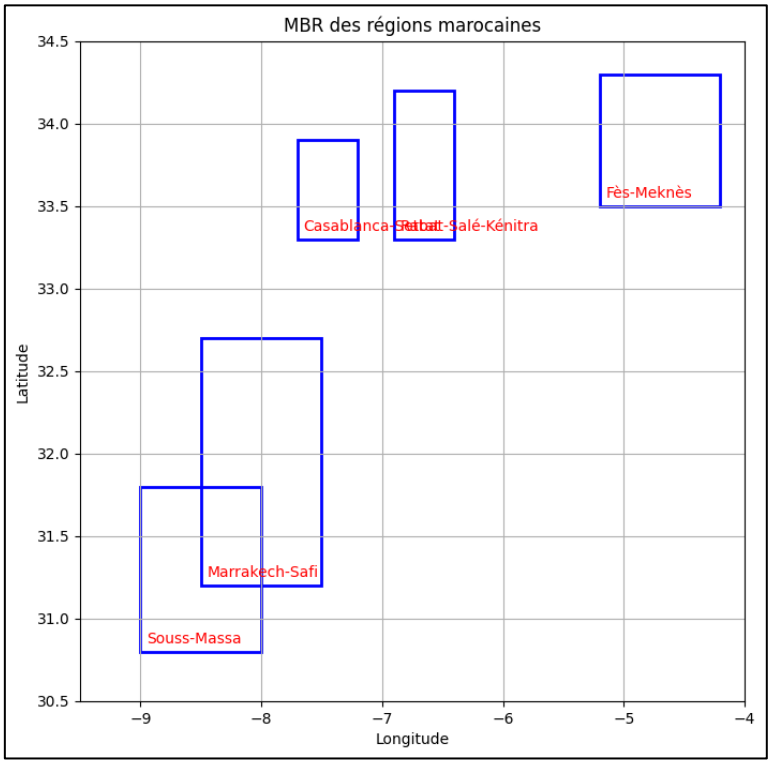
- Pour interroger un rectangle ou intersection : on descend les nœuds dont MBR intersecte la requête.
- On parcourt seulement les branches pertinentes.

Exemple : régions :

Pour illustrer, on utilise 5 régions (MBR simplifiés — coordonnées fictives proches des valeurs du rapport pour trois d'entre elles) :

1. **Casablanca-Settat (C)** : xmin = −7.7, ymin = 33.3, xmax = −7.2, ymax = 33.9
2. **Rabat-Salé-Kénitra (R)** : xmin = −6.9, ymin = 33.3, xmax = −6.4, ymax = 34.2
3. **Marrakech-Safi (M)** : xmin = −8.5, ymin = 31.2, xmax = −7.5, ymax = 32.7
4. **Fès-Meknès (F)** : xmin = −5.2, ymin = 33.5, xmax = −4.2, ymax = 34.3
5. **Souss-Massa (S)** : xmin = −9.0, ymin = 30.8, xmax = −8.0, ymax = 31.8

Chaque MBR devient une entrée feuille à insérer dans l'arbre.

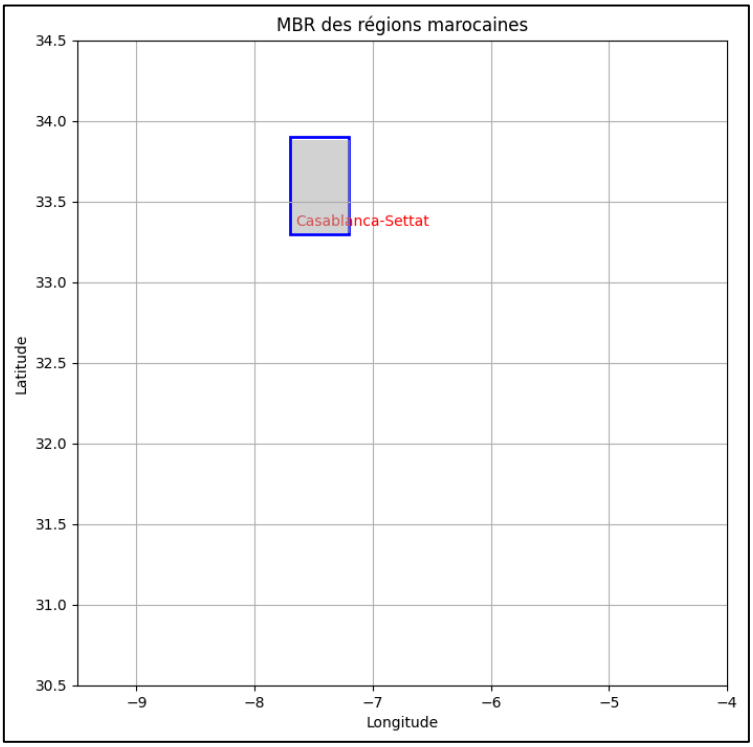


Étape A : création du premier nœud feuille

[Feuille] : { C }

MBR = bbox(C)

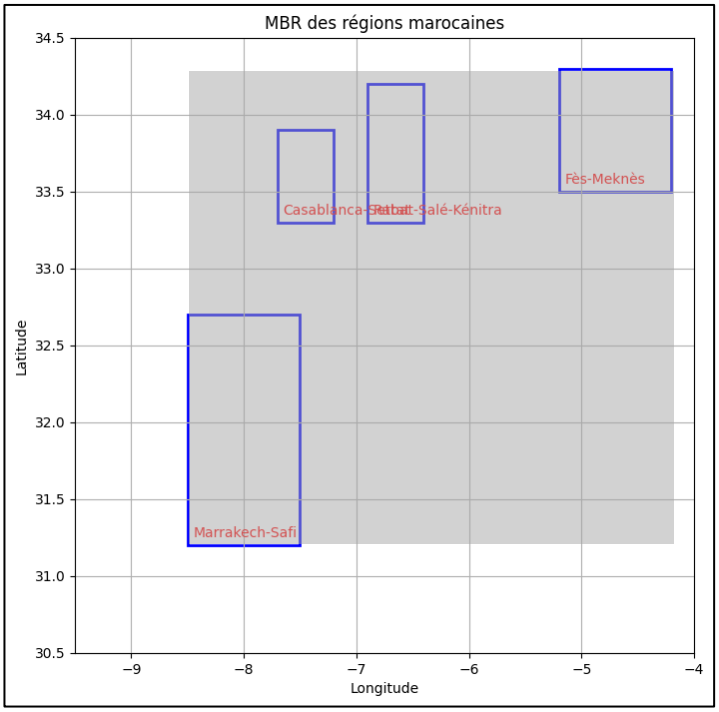
Schéma MBR : un rectangle autour de Casablanca-Settat.



Étape B : insertion jusqu'à M=4 (remplissage) :

[Feuille] : {C, R, M, F} (plein)

MBR = bbox(C ∪ R ∪ M ∪ F)



Étape C : insertion qui déclenche le split :

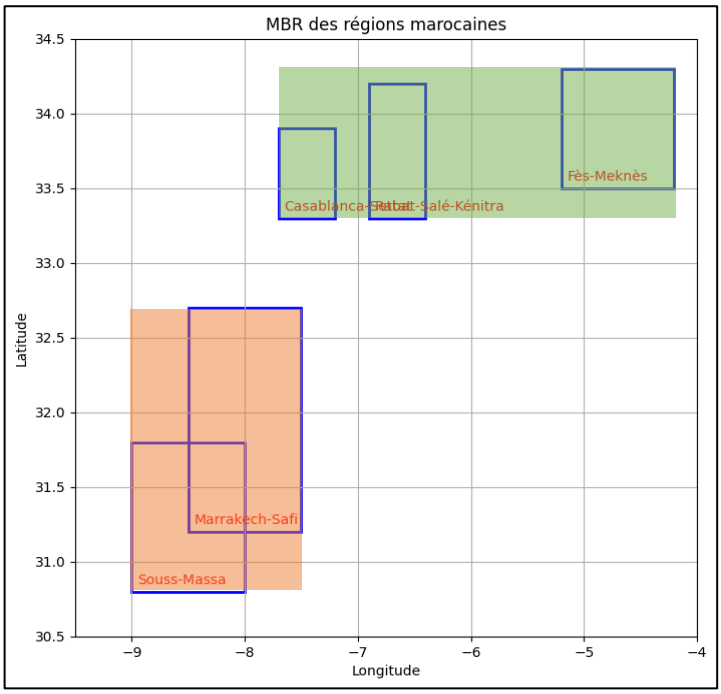
On veut insérer **S.**

La feuille contient déjà 4 entrées (**M + 1 = 5**) entrées dans la feuille donc il y'aura un **split** obligatoire qui respecte les contraintes suivantes :

- Distribuer les autres entrées (C, R, S) en comparant l'augmentation d'aire (MBR) que causerait leur ajout à chacun des deux groupes. On affecte à la partie qui fait la plus faible augmentation.
- Contraintes m = 2 : Chaque groupe doit finir avec au moins 2 entrées.

résultat :

- **Nœud A** (feuille) : { M, S } → MBR_A = bbox(M ∪ S)
- **Nœud B** (feuille) : { C, R, F } → MBR_B = bbox(C ∪ R ∪ F)



Étape D : création d'un nouveau niveau si on split la racine :

Si le nœud scindé était la racine (c'est le cas ici), on crée une **nouvelle racine** qui devient parent des deux feuilles A et B.

[Racine]

/ \

[Feuille A] [Feuille B]

{ M, S } { C, R, F }

6.2 INDEXATION R*-TREE:

Un *R-tree** est une variante améliorée qui, au lieu de splitter immédiatement, peut **réinsérer** certaines entrées et utilise des critères de split qui minimisent le **recouvrement** et l'aire totale ; ceci réduit les chevauchements et améliore les performances de recherche.

- Principales différences opérationnelles :

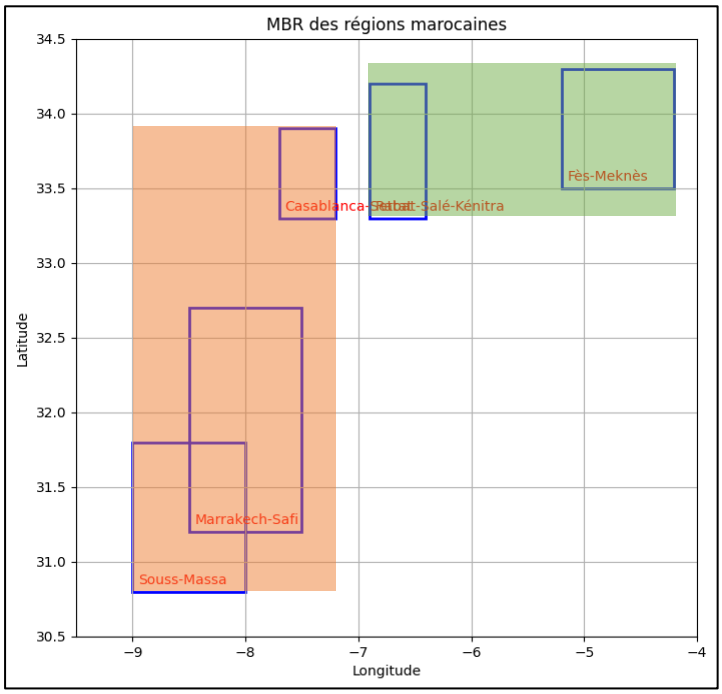
  - **Réinsertion avant split** : quand un nœud déborde, R*-tree **réinsère** à partir du niveau racine plutôt que de splitter immédiatement. Cela permet de redistribuer des entrées vers des nœuds plus adaptés et souvent d'éviter le split.
  - **Critère de split** : quand split nécessaire, R*-tree choisit la répartition qui **minimise le recouvrement** (overlap) des MBRs enfants, puis l'aire totale.
  - **Conséquence** : moins de chevauchement entre MBRs, arborescence plus compacte, meilleures performances pour requêtes spatiales.

Application sur l'exemple précédant :

Par réinsertion, on peut obtenir une répartition :

- **Nœud 1** : { S, M , C }
- **Nœud 2** : { R , F }

Plus généralement, le R*-tree cherchera une répartition qui **réduit le chevauchement** entre les rectangles comparativement au R-tree classique.



7 DUCKDB POUR L'ANALYSE SPATIALE

7.1 PRESENTATION GENERALE DE DUCKDB

DuckDB est un moteur de base de données analytique moderne, orienté OLAP (Online Analytical Processing), conçu pour le traitement efficace de grands volumes de données locales.

Contrairement à des systèmes serveurs comme PostgreSQL/PostGIS, DuckDB est un moteur embarqué : il s'exécute directement dans une application ou un environnement (comme Python, R, QGIS ou un notebook Jupyter) sans nécessiter d'installation serveur.

Son architecture colonnaire (stockage par colonne et non par ligne) le rend particulièrement performant pour les opérations analytiques, les jointures lourdes et les agrégations massives.

DuckDB supporte nativement les formats modernes tels que Parquet, CSV, et désormais GeoParquet pour les données spatiales, ce qui facilite l'échange et le traitement de gros volumes de données géographiques.

7.2 AVANTAGES DE DUCKDB DANS LE CONTEXTE DE CE PROJET

Dans le cadre d'une analyse agricole et territoriale à l'échelle nationale, les données peuvent devenir très volumineuses (couches géographiques complexes, indicateurs multiples, calculs de distances et d'intersections).

DuckDB offre plusieurs avantages :

- Performance élevée : traitement rapide de jeux de données volumineux directement depuis le disque, sans serveur.
- Simplicité d'intégration : peut être utilisé avec des langages comme Python pour effectuer des requêtes SQL locales.
- Compatibilité avec les formats spatiaux récents : prise en charge de GeoParquet et des types géométriques via des extensions.
- Faible consommation de ressources : pas besoin d'un serveur PostgreSQL, idéal pour une exécution sur un ordinateur personnel.
- Interopérabilité : peut lire et traiter des fichiers déjà utilisés dans PostGIS (exportés en CSV, GeoPackage ou Parquet).

Ainsi, DuckDB se positionne comme une alternative légère à PostGIS pour les analyses ponctuelles et les explorations rapides de données spatiales volumineuses.

7.3 REQUETES SPATIALES REALISEES DUCKDB

On a essayé d'utiliser DuckDB pour l'exécution des requêtes spatiales qui ont pris beaucoup du temps pour donner un résultat su QGIS.

Requête 1 : Longueur totale du réseau routier par région.

-Script :

```
import duckdb
import pandas as pd
```

```
import geopandas as gpd

con = duckdb.connect(database=':memory:', read_only=False)
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
print("Extensions spatiales DuckDB chargées.")

# --- 1. Charger vos données avec GeoPandas ---
regions_gdf = gpd.read_file("regions.shp")
roads_gdf = gpd.read_file("roads.shp")
print("Fichiers SHP chargés avec GeoPandas.")

# --- 2. Reprojeter les GeoDataFrames vers un CRS projeté (par exemple,
UTM) ---
# Le Maroc couvre plusieurs zones UTM. Pour un calcul national,
# il faut choisir une zone représentative ou faire des calculs par
zone.
# Pour simplifier, prenons une zone UTM standard et acceptons une
légère distorsion

target_crs = "EPSG:32629"

regions_gdf_proj = regions_gdf.to_crs(target_crs)
roads_gdf_proj = roads_gdf.to_crs(target_crs)
print(f"Géométries reprojetées en {target_crs} (unités en mètres).")

# --- 3. Convertir la colonne 'geometry' reprojetée en WKT et préparer
les DataFrames Pandas ---

# Pour les régions
regions_df_for_duckdb =
pd.DataFrame(regions_gdf_proj.drop(columns=['geometry']))
regions_df_for_duckdb['geom_wkt'] =
regions_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)

# Pour les routes
roads_df_for_duckdb =
pd.DataFrame(roads_gdf_proj.drop(columns=['geometry']))
roads_df_for_duckdb['geom_wkt'] =
roads_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)

print("Géométries reprojetées converties en WKT et DataFrames Pandas
préparés.")

# --- 4. Créer les tables DuckDB à partir des DataFrames Pandas
standards ---
con.execute("CREATE TABLE regions AS SELECT * FROM
regions_df_for_duckdb;")
con.execute("CREATE TABLE roads AS SELECT * FROM roads_df_for_duckdb;")
```

```
print("Tables 'regions' et 'roads' créées avec succès dans DuckDB.")

# --- 5. Exécution de la requête DuckDB avec ST_GeomFromText ---
query = """
SELECT
    r.nom_region,
    SUM(
        ST_Length(
            ST_Intersection(
                ST_GeomFromText(ro.geom_wkt),
                ST_GeomFromText(r.geom_wkt)
            )
        )
        / 1000.0 -- Maintenant, la division par 1000.0 est correcte car
ST_Length retourne des mètres
    ) AS longueur_routes_km
FROM
    regions AS r
JOIN
    roads AS ro ON ST_Intersects(ST_GeomFromText(r.geom_wkt),
ST_GeomFromText(ro.geom_wkt))
GROUP BY
    r.nom_region
ORDER BY
    longueur_routes_km DESC;
"""

print("\nExécution de la requête SQL...")
result_df = con.execute(query).fetchdf()

print("\nRésultats de la requête (longueurs en km après
reprojection):")
print(result_df)

con.close()
print("\nConnexion DuckDB fermée.")
```

-Table :

	nom_region	longueur_routes_km
0	Région de Marrakech-Safi	69052.029295
1	Région de Casablanca-Settat	54239.541367
2	Région de Souss-Massa	53777.286897
3	Région de Drâa-Tafilalet	47132.773278
4	Région de Fès-Meknès	43349.115721
5	Région de l'Oriental	42450.856186
6	Région de Béni Mellal-Khénifra	36938.439249
7	Région de Rabat-Salé-Kénitra	33761.267121
8	Région de Tanger-Tétouan-Al Hoceima	31080.671948
9	Région de Laâyoune-Sakia El Hamra	21519.580575
10	Région de Guelmim-Oued Noun	18507.533728
11	Région de Dakhla-Oued Ed-Dahab	8591.968677

Requête 2 : Densité routière (km de routes par 100 km²) pour chaque région

-Script :

```
import duckdb
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt # <--- Assurez-vous que cette ligne est
présente !

con = duckdb.connect(database=':memory:', read_only=False)
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
print("Extensions spatiales DuckDB chargées.")

# --- 1. Charger vos données avec GeoPandas ---
try:
    regions_gdf = gpd.read_file("regions.shp")
    roads_gdf = gpd.read_file("roads.shp")
    print("Fichiers SHP chargés avec GeoPandas.")
except Exception as e:
    print(f"Erreur lors du chargement des fichiers SHP: {e}")

# --- 2. Reprojeter les GeoDataFrames vers un CRS projeté (par exemple,
UTM) ---
target_crs = "EPSG:32629" # Assurez-vous que c'est la bonne zone UTM
pour vos données

regions_gdf_proj = regions_gdf.to_crs(target_crs)
roads_gdf_proj = roads_gdf.to_crs(target_crs)
print(f"Géométries reprojetées en {target_crs} (unités en mètres).")

# --- 3. Convertir la colonne 'geometry' reprojetée en WKT et préparer
les DataFrames Pandas ---
regions_df_for_duckdb =
pd.DataFrame(regions_gdf_proj.drop(columns=['geometry']))
regions_df_for_duckdb['geom_wkt'] =
regions_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)

roads_df_for_duckdb =
pd.DataFrame(roads_gdf_proj.drop(columns=['geometry']))
roads_df_for_duckdb['geom_wkt'] =
roads_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)

print("Géométries reprojetées converties en WKT et DataFrames Pandas
préparés.")

# --- 4. Créer les tables DuckDB à partir des DataFrames Pandas
standards ---
```

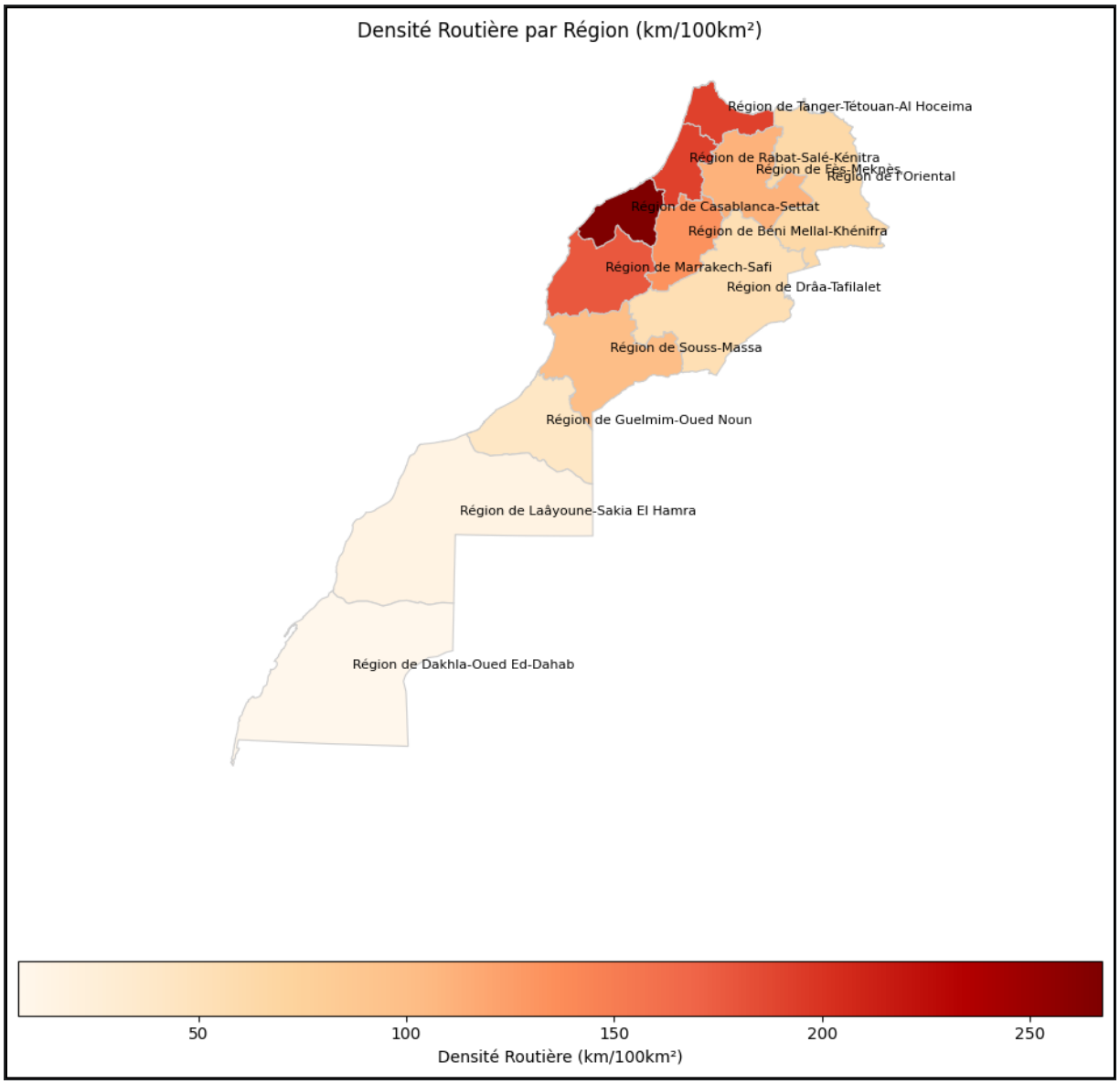
```
con.execute("CREATE TABLE regions AS SELECT * FROM
regions_df_for_duckdb;")
con.execute("CREATE TABLE roads AS SELECT * FROM roads_df_for_duckdb;")
print("Tables 'regions' et 'roads' créées avec succès dans DuckDB.")

# --- 5. Exécution de la deuxième requête DuckDB (Densité Routière) ---
query_densite_routiere = """
SELECT
    r.nom_region,
    (
        (SUM(
            ST_Length(
                ST_Intersection(
                    ST_GeomFromText(ro.geom_wkt),
                    ST_GeomFromText(r.geom_wkt)
                )
            )
        ) / 1000.0)
        /
        (ST_Area(ST_GeomFromText(r.geom_wkt)) / 1000000.0)
    ) * 100 AS densite_routiere_par_100km2
FROM
    regions AS r
JOIN
    roads AS ro ON ST_Intersects(ST_GeomFromText(r.geom_wkt),
ST_GeomFromText(ro.geom_wkt))
GROUP BY
    r.nom_region, ST_GeomFromText(r.geom_wkt)
ORDER BY
    densite_routiere_par_100km2 DESC;
"""

print("\nExécution de la requête SQL pour la densité routière...")
result_df_densite_routiere =
con.execute(query_densite_routiere).fetchdf()

print("\nRésultats de la requête (Densité routière en km/100km² après
reprojection):")
print(result_df_densite_routiere)

con.close()
print("\nConnexion DuckDB fermée.")

# --- 6. Préparation pour la cartographie ---

# Joindre les résultats de densité routière aux géométries des régions
regions_with_density = regions_gdf_proj.merge(
    result_df_densite_routiere,
    on='nom_region',
```

```
    how='left'
)

regions_with_density['densite_routiere_par_100km2'] =
regions_with_density['densite_routiere_par_100km2'].fillna(0)

print("\nGeoDataFrame de régions avec densité routière:")
print(regions_with_density.head())

# --- 7. Création de la carte ---
fig, ax = plt.subplots(1, 1, figsize=(12, 12))

regions_with_density.plot(
    column='densite_routiere_par_100km2',
    cmap='OrRd',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True,
    legend_kwds={'label': "Densité Routière (km/100km²)",
'orientation': "horizontal"}
)

for x, y, label in zip(regions_with_density.geometry.centroid.x,
regions_with_density.geometry.centroid.y,
regions_with_density.nom_region):
    ax.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset
points", fontsize=8, color='black')

ax.set_title("Densité Routière par Région (km/100km²)")
ax.set_axis_off()

plt.show() # Afficher la carte
```

-Table :

	nom_region	densite_routiere_par_100km2
0	Région de Casablanca-Settat	267.353359
1	Région de Tanger-Tétouan-Al Hoceima	191.850180
2	Région de Rabat-Salé-Kénitra	191.108297
3	Région de Marrakech-Safi	177.217715
4	Région de Béni Mellal-Khénifra	133.460936
5	Région de Fès-Meknès	111.156284
6	Région de Souss-Massa	100.326816
7	Région de l'Oriental	63.719386
8	Région de Drâa-Tafilalet	54.699161
9	Région de Guelmim-Oued Noun	41.494012
10	Région de Laâyoune-Sakia El Hamra	14.970201
11	Région de Dakhla-Oued Ed-Dahab	6.598440

-Carte :



Requête 3 : Analyse croisée entre superficie irrigable et densité routière.

-Script :

```
import duckdb
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt

# Connexion à Duckdb
con = duckdb.connect(database=':memory:', read_only=False)
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
print("Extensions spatiales DuckDB chargées.")

# --- 1. Charger vos données avec GeoPandas ---
```

```
try:
    regions_gdf = gpd.read_file("regions.shp")
    roads_gdf = gpd.read_file("roads.shp")
    print("Fichiers SHP chargés avec GeoPandas.")
except Exception as e:
    print(f"Erreur lors du chargement des fichiers SHP: {e}")
    exit()

# --- 2. Reprojeter les GeoDataFrames vers un CRS projeté (par exemple,
UTM) ---
target_crs = "EPSG:32629"

regions_gdf_proj = regions_gdf.to_crs(target_crs)
roads_gdf_proj = roads_gdf.to_crs(target_crs)
print(f"Géométries reprojetées en {target_crs} (unités en mètres).")

# --- 3. Convertir la géométrie en WKT et préparer les DataFrames pour
DuckDB ---
regions_df_for_duckdb = pd.DataFrame(regions_gdf_proj)
regions_df_for_duckdb['geom_wkt'] =
regions_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)
regions_df_for_duckdb =
regions_df_for_duckdb.drop(columns=['geometry'])

roads_df_for_duckdb = pd.DataFrame(roads_gdf_proj)
roads_df_for_duckdb['geom_wkt'] =
roads_gdf_proj['geometry'].apply(lambda g: g.wkt if g else None)
roads_df_for_duckdb = roads_df_for_duckdb.drop(columns=['geometry'])

print("Géométries reprojetées converties en WKT et DataFrames Pandas
préparés.")

# --- 4. Créer les tables DuckDB ---
con.execute("CREATE TABLE regions AS SELECT * FROM
regions_df_for_duckdb;")
con.execute("CREATE TABLE roads AS SELECT * FROM roads_df_for_duckdb;")
print("Tables 'regions' et 'roads' créées avec succès dans DuckDB.")

# --- 5. Exécution de la requête SQL (Analyse croisée)
query_analyse_croisee = """
SELECT
    r.nom_region,
    -- MODIFIÉ : Renommage de la colonne pour éviter le conflit lors de
la fusion
    MAX(r.superficie) AS superficie_irrigable,
    SUM(ST_Length(ST_Intersection(ST_GeomFromText(ro.geom_wkt),
ST_GeomFromText(r.geom_wkt))) / 1000) AS longueur_routes_km,
    (MAX(ST_Area(ST_GeomFromText(r.geom_wkt))) / 1000000) AS
superficie_region_km2,
```

```
    (SUM(ST_Length(ST_Intersection(ST_GeomFromText(ro.geom_wkt),
ST_GeomFromText(r.geom_wkt))) / 1000) /
(MAX(ST_Area(ST_GeomFromText(r.geom_wkt))) / 1000000)) * 100 AS
densite_routiere_par_100km2
FROM
    regions AS r
JOIN
    roads AS ro ON ST_Intersects(ST_GeomFromText(r.geom_wkt),
ST_GeomFromText(ro.geom_wkt))
-- CORRIGÉ : On groupe uniquement par nom de région pour sommer TOUTES
les routes d'une région.
GROUP BY
    r.nom_region
ORDER BY
    densite_routiere_par_100km2 DESC;
"""

print("\nExécution de la requête SQL pour l'analyse croisée (superficie
irrigable et densité routière)...")
result_df_analyse_croisee =
con.execute(query_analyse_croisee).fetchdf()

print("\nRésultats de l'analyse croisée (Densité routière et Superficie
Irrigable):")
print(result_df_analyse_croisee)

# --- 6. Préparation pour la cartographie ---

# Joindre les résultats de l'analyse croisée aux géométries des régions
regions_with_analysis = regions_gdf_proj.merge(
    result_df_analyse_croisee,
    on='nom_region',
    how='left'
)

# Gérer les cas où il n'y aurait pas de routes ou de superficie dans
les résultats
regions_with_analysis['longueur_routes_km'] =
regions_with_analysis['longueur_routes_km'].fillna(0)
regions_with_analysis['superficie_region_km2'] =
regions_with_analysis['superficie_region_km2'].fillna(0)
regions_with_analysis['densite_routiere_par_100km2'] =
regions_with_analysis['densite_routiere_par_100km2'].fillna(0)
regions_with_analysis['superficie_irrigable'] =
regions_with_analysis['superficie_irrigable'].fillna(0)
# S'assurer que les NaN sont gérés

print("\nGeoDataFrame de régions avec les résultats de l'analyse:")
print(regions_with_analysis.head())
```

```
# --- 7. Création des deux cartes côte à côte ---
fig, axes = plt.subplots(1, 2, figsize=(20, 10))
fig.suptitle("Analyse croisée entre Densité Routière et Superficie
Irrigable", fontsize=16)

# --- Carte 1: Densité Routière ---
regions_with_analysis.plot(
    column='densite_routiere_par_100km2',
    cmap='OrRd', # Palette de couleurs du jaune au rouge
    linewidth=0.8,
    ax=axes[0],
    edgecolor='0.8',
    legend=True,
    legend_kwds={
        'label': "Densité Routière (km/100km²)",
        'orientation': "horizontal",
        'shrink': 0.7,
        'pad': 0.05
    }
)
for x, y, label in zip(regions_with_analysis.geometry.centroid.x,
regions_with_analysis.geometry.centroid.y,
regions_with_analysis.nom_region):
    axes[0].annotate(label, xy=(x, y), xytext=(3, 3),
textcoords="offset points", fontsize=7, color='black', ha='center')

axes[0].set_title("Densité Routière par Région", fontsize=14)
axes[0].set_axis_off()

# --- Carte 2: Superficie Irrigable ---
regions_with_analysis.plot(
    column='superficie_irrigable',
    cmap='GnBu', # Palette de couleurs du vert au bleu (souvent associé à
l'eau/végétation)
    linewidth=0.8,
    ax=axes[1],
    edgecolor='0.8',
    legend=True,
    legend_kwds={
        'label': "Superficie Irrigable (hectares)",
        'orientation': "horizontal",
        'shrink': 0.7,
        'pad': 0.05
    }
)
for x, y, label in zip(regions_with_analysis.geometry.centroid.x,
regions_with_analysis.geometry.centroid.y,
regions_with_analysis.nom_region):
```

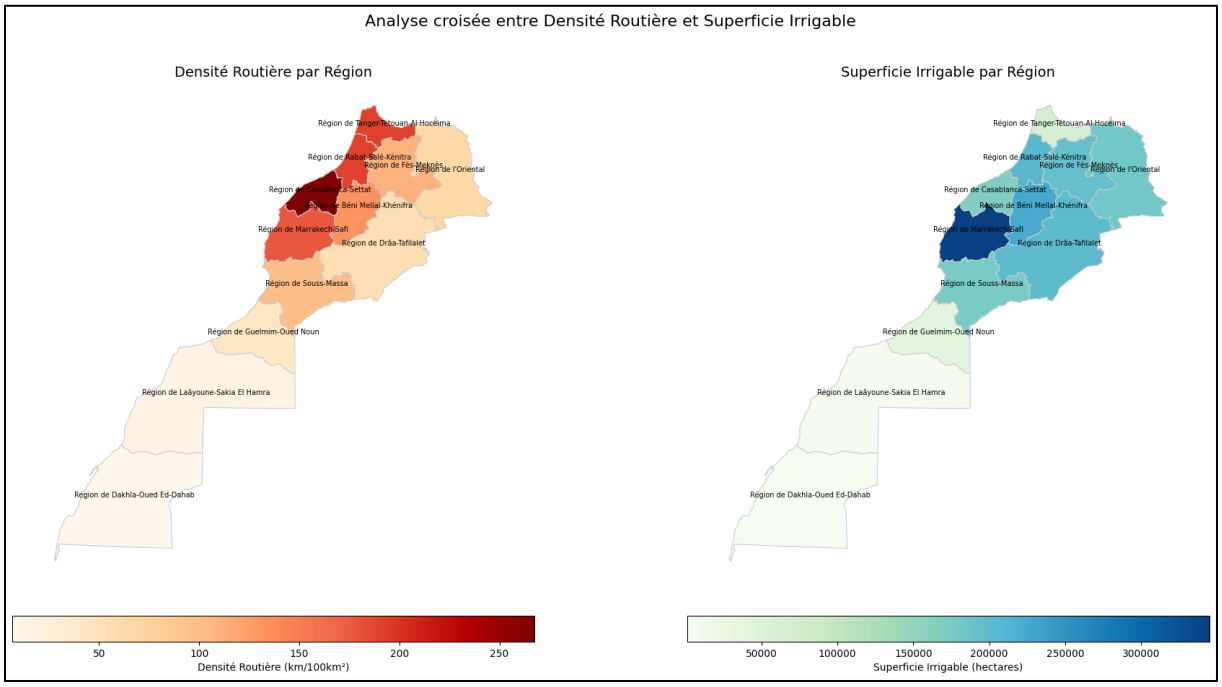
```
    axes[1].annotate(label, xy=(x, y), xytext=(3, 3),
textcoords="offset points", fontsize=7, color='black', ha='center')

axes[1].set_title("Superficie Irrigable par Région", fontsize=14)
axes[1].set_axis_off()

plt.tight_layout(rect=[0, 0, 1, 0.96]) # Ajuste la mise en page pour le
titre principal
plt.show()

# --- Fermeture de la connexion ---
con.close()
print("\nConnexion DuckDB fermée.")
```

-Tables :

	nom_region	superficie_irrigable	\
0	Région de Casablanca-Settat	168238	
1	Région de Tanger-Tétouan-Al Hoceima	66174	
2	Région de Rabat-Salé-Kénitra	208000	
3	Région de Marrakech-Safi	345032	
4	Région de Béni Mellal-Khénifra	226293	
5	Région de Fès-Meknès	193542	
6	Région de Souss-Massa	174862	
7	Région de l'Oriental	181388	
8	Région de Drâa-Tafilalet	201922	
9	Région de Guelmim-Oued Noun	36602	
10	Région de Laâyoune-Sakia El Hamra	5543	
11	Région de Dakhla-Oued Ed-Dahab	1152	

	longueur_routes_km	superficie_region_km2	densite_routiere_par_100km2
0	54239.541367	20287.585538	267.353359
1	31080.671948	16200.491418	191.850180
2	33761.267121	17666.039440	191.108297
3	69052.029295	38964.518576	177.217715
4	36938.439249	27677.341743	133.460936
5	43349.115721	38998.349269	111.156284
6	53777.286897	53602.106421	100.326816
7	42450.856186	66621.571705	63.719386
8	47132.773278	86167.268820	54.699161
9	18507.533728	44602.902370	41.494012
10	21519.580575	143749.447657	14.970201
11	8591.968677	130212.114590	6.598440

-Cartes :



-Interprétation : On peut dire qu'on a à peu près une corrélation positive entre densité routière et superficie irrigable par régions.

Trois requêtes originales impliquant des jointures spatiales :

-Script :

```
import duckdb
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt

# Connexion à DuckDB
con = duckdb.connect(database=':memory:', read_only=False)
con.execute("INSTALL spatial;")
con.execute("LOAD spatial;")
print("Extensions spatiales DuckDB chargées.")

# --- 1. Charger vos données avec GeoPandas ---
try:
    regions_gdf = gpd.read_file("regions.shp")
    roads_gdf = gpd.read_file("roads.shp")
    pois_gdf = gpd.read_file("gis_osm_pois_a_free_1.shp") # Charger les
POIs
    cours_d_eau_gdf = gpd.read_file("cours_d_eau.shp") # Charger les
cours d'eau
    print("Fichiers SHP chargés avec GeoPandas.")
except Exception as e:
    print(f"Erreur lors du chargement des fichiers SHP: {e}")
    exit()
```

```
# --- 2. Reprojeter les GeoDataFrames vers un CRS projeté (par exemple,
UTM) ---
target_crs = "EPSG:32629" # Assurez-vous que c'est la bonne zone UTM
pour vos données

regions_gdf_proj = regions_gdf.to_crs(target_crs)
roads_gdf_proj = roads_gdf.to_crs(target_crs)
pois_gdf_proj = pois_gdf.to_crs(target_crs) # Reprojeter les POIs
cours_d_eau_gdf_proj = cours_d_eau_gdf.to_crs(target_crs) # Reprojeter
les cours d'eau
print(f"Géométries reprojetées en {target_crs} (unités en mètres).")

# --- 3. Convertir la colonne 'geometry' reprojetée en WKT et préparer
les DataFrames Pandas ---
def prepare_df_for_duckdb(gdf_proj):
    df = pd.DataFrame(gdf_proj)
    df['geom_wkt'] = gdf_proj['geometry'].apply(lambda g: g.wkt if g
else None)
    return df.drop(columns=['geometry'])

regions_df_for_duckdb = prepare_df_for_duckdb(regions_gdf_proj)
roads_df_for_duckdb = prepare_df_for_duckdb(roads_gdf_proj)
pois_df_for_duckdb = prepare_df_for_duckdb(pois_gdf_proj) # Préparer
les POIs
cours_d_eau_df_for_duckdb = prepare_df_for_duckdb(cours_d_eau_gdf_proj)
# Préparer les cours d'eau

print("Géométries reprojetées converties en WKT et DataFrames Pandas
préparés.")

# --- 4. Créer les tables DuckDB à partir des DataFrames Pandas
standards ---
con.execute("CREATE TABLE regions AS SELECT * FROM
regions_df_for_duckdb;")
con.execute("CREATE TABLE roads AS SELECT * FROM roads_df_for_duckdb;")
con.execute("CREATE TABLE gis_osm_pois_a_free_1 AS SELECT * FROM
pois_df_for_duckdb;")
con.execute("CREATE TABLE cours_d_eau AS SELECT * FROM
cours_d_eau_df_for_duckdb;")
print("Tables 'regions', 'roads', 'gis_osm_pois_a_free_1' et
'cours_d_eau' créées avec succès dans DuckDB.")
#
================================================================================
============================
```

Requête 1 : Proximité POIs et Routes

- **Ce que ça fait :** Compte le nombre de Points d'Intérêt (POI) situés très près (moins de 100m) des routes dans chaque région.

- **Pourquoi c'est utile :** Évalue l'accessibilité des services (magasins, écoles, etc.) via le réseau routier. Plus le nombre est élevé, mieux les services sont connectés aux routes.
- **La carte montre :** Les régions où les POIs sont les plus accessibles par la route.

Requête 2 : Cours d'eau, Irrigation et Bétail

- **Ce que ça fait :** Calcule la longueur totale des cours d'eau dans chaque région et la met en parallèle avec la superficie irrigable et le total du cheptel (bovins, ovins, etc.).
- **Pourquoi c'est utile :** Permet de comprendre quelles régions ont de bonnes ressources en eau pour l'agriculture (irrigation) et l'élevage, et si ces secteurs sont développés là où l'eau est présente.
- **La carte montre :** La répartition des cours d'eau par région, à comparer avec les chiffres d'irrigation et de bétail pour trouver des synergies ou des manques.

Requête 3 : Indice de Développement Agricole Routier

- **Ce que ça fait :** Crée un score unique pour chaque région en multipliant sa densité routière par la proportion de sa superficie qui est irrigable.
- **Pourquoi c'est utile :** Identifie les régions où le réseau routier est le plus pertinent pour soutenir l'agriculture irriguée. Un score élevé signifie à la fois de bonnes routes et un fort potentiel d'irrigation.
- **La carte montre :** Les régions avec le meilleur "potentiel" combiné de routes et d'agriculture irriguée, idéales pour cibler les investissements.

Requête 1 :

```
# --- REQUÊTE 1: Proximité des POIs avec les Routes par Région ---
print("\n--- Requête 1: Proximité des POIs avec les Routes par Région -
--")
query_pois_roads_proximity = """
SELECT
    r.nom_region,
    COUNT(DISTINCT p.osm_id) AS nombre_pois_proches_routes --
CORRECTION ICI: Remplacer p.gid par p.osm_id
FROM
    regions AS r
JOIN
    gis_osm_pois_a_free_1 AS p ON
ST_Intersects(ST_GeomFromText(r.geom_wkt), ST_GeomFromText(p.geom_wkt))
JOIN
    roads AS ro ON ST_DWithin(ST_GeomFromText(p.geom_wkt),
ST_GeomFromText(ro.geom_wkt), 100)
GROUP BY
    r.nom_region
ORDER BY
    nombre_pois_proches_routes DESC;
"""
```

```
result_df_pois_roads =
con.execute(query_pois_roads_proximity).fetchdf()
print("\nRésultats de la Requête 1 (Nombre de POIs à <100m des routes
par région):")
print(result_df_pois_roads)
# Préparation pour la carte 1
regions_map1 = regions_gdf_proj.merge(result_df_pois_roads,
on='nom_region', how='left')
regions_map1['nombre_pois_proches_routes'] =
regions_map1['nombre_pois_proches_routes'].fillna(0)

fig1, ax1 = plt.subplots(1, 1, figsize=(10, 8))
regions_map1.plot(
    column='nombre_pois_proches_routes',
    cmap='viridis',
    linewidth=0.8,
    ax=ax1,
    edgecolor='0.8',
    legend=True,
    legend_kwds={'label': "Nombre de POIs proches des routes",
'orientation': "horizontal", 'shrink': 0.7}
)
for x, y, label in zip(regions_map1.geometry.centroid.x,
regions_map1.geometry.centroid.y, regions_map1.nom_region):
    ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset
points", fontsize=7, color='black', ha='center')
ax1.set_title("Requête 1: Nombre de POIs proches des routes par
Région", fontsize=14)
ax1.set_axis_off()
plt.tight_layout()
plt.show() # Afficher la carte pour la Requête 1


#
==============================================================================
============================
```

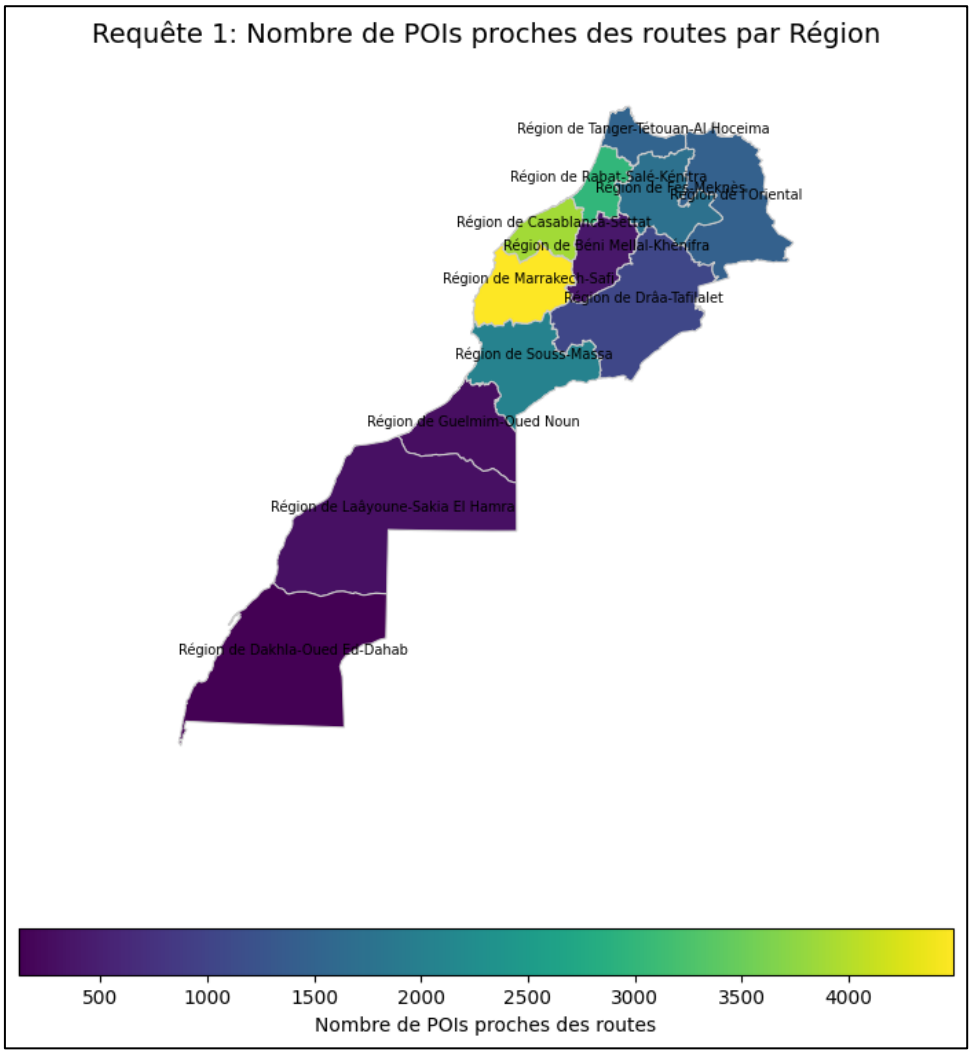
Table :

```
Résultats de la Requête 1 (Nombre de POIs à <100m des routes par région):
```

	nom_region	nombre_pois_proches_routes
0	Région de Marrakech-Safi	4490
1	Région de Casablanca-Settat	3885
2	Région de Rabat-Salé-Kénitra	2996
3	Région de Souss-Massa	2059
4	Région de Fès-Meknès	1731
5	Région de Tanger-Tétouan-Al Hoceima	1503

```
6                  Région de l'Oriental                    1465
7              Région de Drâa-Tafilalet                    1049
8         Région de Béni Mellal-Khénifra                    401
9      Région de Laâyoune-Sakia El Hamra                    290
10           Région de Guelmim-Oued Noun                    286
11        Région de Dakhla-Oued Ed-Dahab                    118
```

Carte :



Requête 2 :

```
# --- REQUÊTE 2: Impact Potentiel des Cours d'Eau sur l'Irrigation et
le Bétail par Région ---
print("\n--- Requête 2: Impact Potentiel des Cours d'Eau sur
l'Irrigation et le Bétail par Région ---")
query_water_impact = """
SELECT
    r.nom_region,
    r.superficie,
```

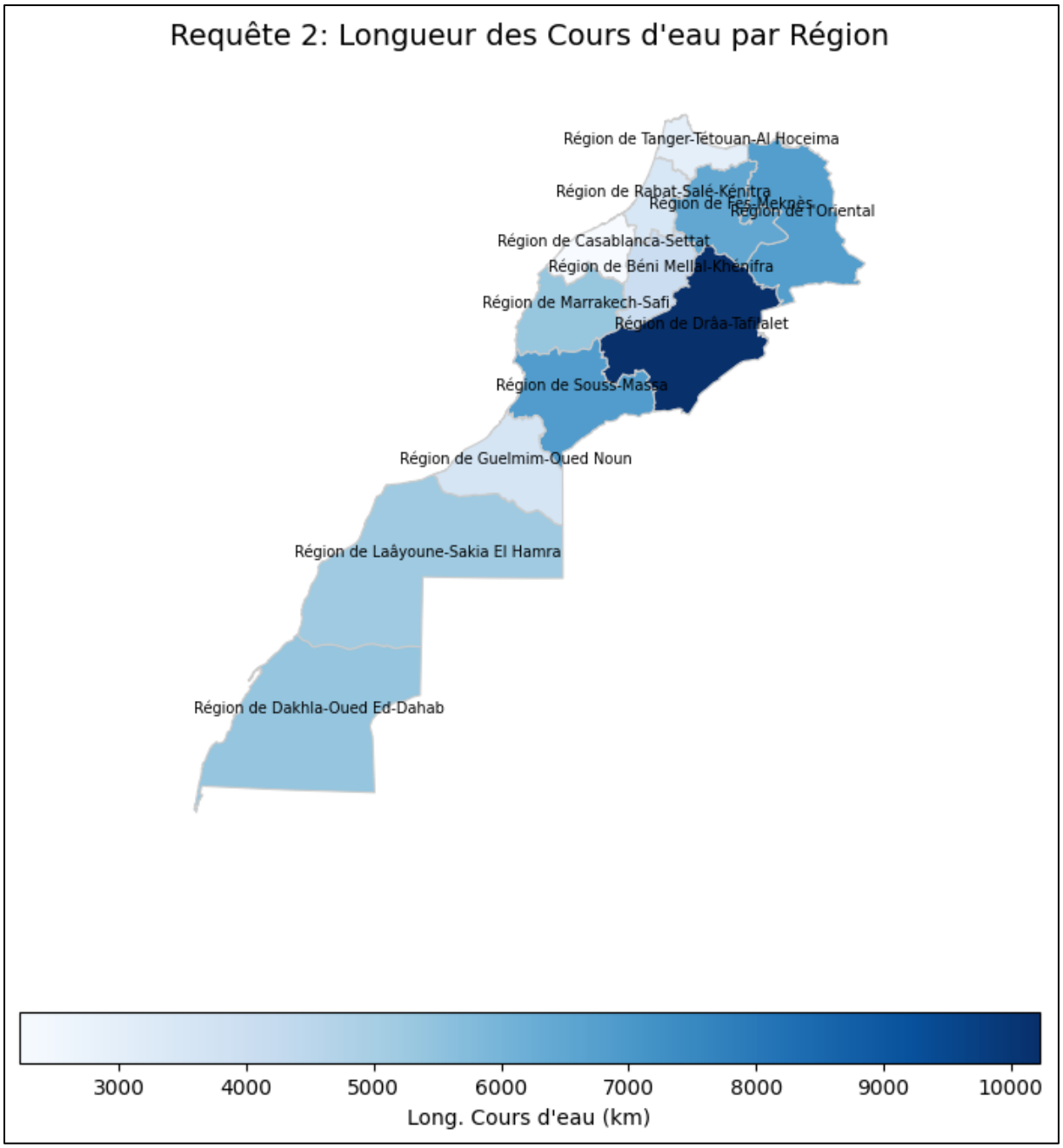
```
    COALESCE(r.cheptel_bo, 0) + COALESCE(r.cheptel_ov, 0) +
COALESCE(r.cheptel_ca, 0) + COALESCE(r.cheptel__1, 0) AS cheptel_total,
    SUM(ST_Length(ST_Intersection(ST_GeomFromText(ce.geom_wkt),
ST_GeomFromText(r.geom_wkt))) / 1000) AS longueur_cours_eau_km
FROM
    regions AS r
JOIN
    cours_d_eau AS ce ON ST_Intersects(ST_GeomFromText(r.geom_wkt),
ST_GeomFromText(ce.geom_wkt))
WHERE
    ce.dis_av_cms > 0.1 -- FILTRE CRUCIAL : On ne garde que les cours
d'eau avec un débit non nul (seuil à ajuster)
GROUP BY
    r.nom_region, r.superficie, r.cheptel_bo, r.cheptel_ov,
r.cheptel_ca, r.cheptel__1
ORDER BY
    longueur_cours_eau_km DESC;
"""
result_df_water_impact = con.execute(query_water_impact).fetchdf()
print("\nRésultats de la Requête 5 (Long. Cours d'eau, Superficie
Irrigable, Cheptel Total par région):")
print(result_df_water_impact)

# Préparation pour la carte 2 (Ex: longueur des cours d'eau)
regions_map2 = regions_gdf_proj.merge(result_df_water_impact,
on='nom_region', how='left')
regions_map2['longueur_cours_eau_km'] =
regions_map2['longueur_cours_eau_km'].fillna(0)

fig2, ax2 = plt.subplots(1, 1, figsize=(10, 8))
regions_map2.plot(
    column='longueur_cours_eau_km',
    cmap='Blues',
    linewidth=0.8,
    ax=ax2,
    edgecolor='0.8',
    legend=True,
    legend_kwds={'label': "Long. Cours d'eau (km)", 'orientation':
"horizontal", 'shrink': 0.7}
)
for x, y, label in zip(regions_map2.geometry.centroid.x,
regions_map2.geometry.centroid.y, regions_map2.nom_region):
    ax2.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset
points", fontsize=7, color='black', ha='center')
ax2.set_title("Requête 2 : Longueur des Cours d'eau par Région",
fontsize=14)
ax2.set_axis_off()
plt.tight_layout()
plt.show() # Afficher la carte pour la Requête
```

-Tables :

```
Résultats de la Requête 2 (Long. Cours d'eau, Superficie Irrigable,
Cheptel Total par région):
                            nom_region  superficie  cheptel_total  \
0              Région de Drâa-Tafilalet      201922        2186318
1                 Région de Souss-Massa      174862        2633800
2                  Région de l'Oriental      181388        4283400
3                  Région de Fès-Meknès      193542        3840270
4        Région de Dakhla-Oued Ed-Dahab        1152         110000
5              Région de Marrakech-Safi      345032        5064000
6     Région de Laâyoune-Sakia El Hamra        5543         613000
7        Région de Béni Mellal-Khénifra      226293        4115000
8           Région de Guelmim-Oued Noun       36602         490697
9          Région de Rabat-Salé-Kénitra      208000        2698827
10  Région de Tanger-Tétouan-Al Hoceima       66174        1626180
11          Région de Casablanca-Settat      168238        3416934
```

```
    longueur_cours_eau_km
0            10227.851405
1             6854.418434
2             6801.593223
3             6471.607482
4             5403.971011
5             5362.649736
6             5199.486629
7             4004.511608
8             3617.277542
9             3499.778607
10            2953.079553
11            2219.220292
```

-Carte :



Requête 3 :

```
# --- REQUÊTE 3: Analyse de la Densité Routière Pondérée par la
Superficie Irrigable ---
print("\n--- Requête 3: Analyse de la Densité Routière Pondérée par la
Superficie Irrigable ---")
query_agri_road_index = """
SELECT
    r.nom_region,
    r.superficie,
    (ST_Area(ST_GeomFromText(r.geom_wkt)) / 1000000) AS
superficie_region_km2,
    (SUM(ST_Length(ST_Intersection(ST_GeomFromText(ro.geom_wkt),
ST_GeomFromText(r.geom_wkt))) / 1000) /
```

```
(ST_Area(ST_GeomFromText(r.geom_wkt)) / 1000000)) * 100 AS
densite_routiere_par_100km2,
    (
        (SUM(ST_Length(ST_Intersection(ST_GeomFromText(ro.geom_wkt),
ST_GeomFromText(r.geom_wkt))) / 1000) /
(ST_Area(ST_GeomFromText(r.geom_wkt)) / 1000000)) * 100
    ) * (CAST(r.superficie AS DOUBLE) /
((ST_Area(ST_GeomFromText(r.geom_wkt)) / 1000000.0) * 100.0)) AS
indice_developpement_agricole_route
FROM
    regions AS r
JOIN
    roads AS ro ON ST_Intersects(ST_GeomFromText(r.geom_wkt),
ST_GeomFromText(ro.geom_wkt))
GROUP BY
    r.nom_region, r.superficie, ST_GeomFromText(r.geom_wkt)
ORDER BY
    indice_developpement_agricole_route DESC;
"""
result_df_agri_road_index =
con.execute(query_agri_road_index).fetchdf()
print("\nRésultats de la Requête 3 (Indice de Développement Agricole
Routier par région):")
print(result_df_agri_road_index)

# Préparation pour la carte 3
regions_map3 = regions_gdf_proj.merge(result_df_agri_road_index,
on='nom_region', how='left')
regions_map3['indice_developpement_agricole_route'] =
regions_map3['indice_developpement_agricole_route'].fillna(0) # Gérer
les NaN si une région n'a pas de routes

fig3, ax3 = plt.subplots(1, 1, figsize=(10, 8))
regions_map3.plot(
    column='indice_developpement_agricole_route',
    cmap='YlGn', # Palette du jaune au vert pour un indice de
développement
    linewidth=0.8,
    ax=ax3,
    edgecolor='0.8',
    legend=True,
    legend_kwds={'label': "Indice de Développement Agricole Routier",
'orientation': "horizontal", 'shrink': 0.7}
)
for x, y, label in zip(regions_map3.geometry.centroid.x,
regions_map3.geometry.centroid.y, regions_map3.nom_region):
    ax3.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset
points", fontsize=7, color='black', ha='center')
```

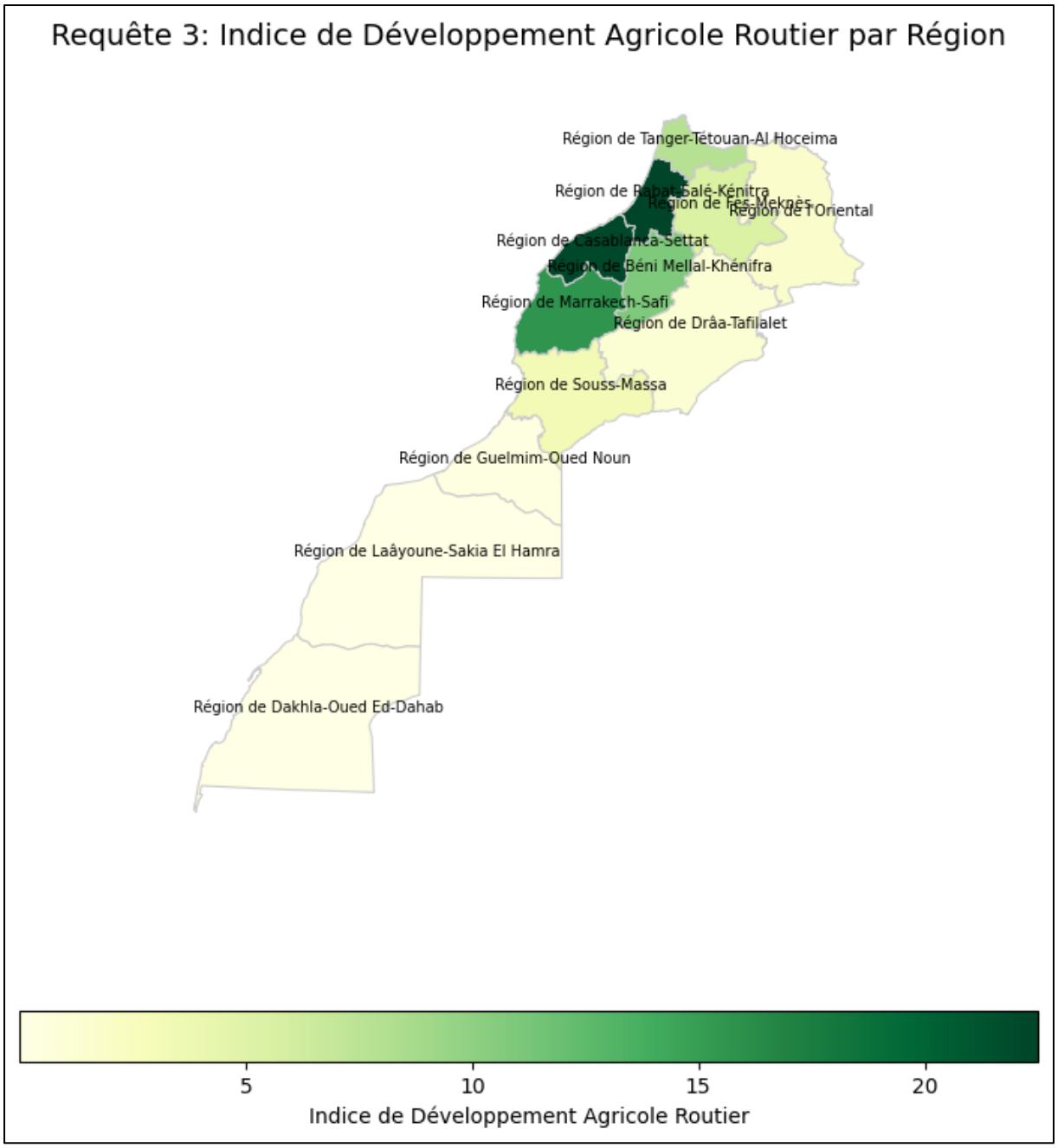
```
ax3.set_title("Requête : Indice de Développement Agricole Routier par
Région", fontsize=14)
ax3.set_axis_off()
plt.tight_layout()
plt.show() # Afficher la carte pour la Requête 3


con.close()
print("\nConnexion DuckDB fermée.")
```

-Tables :

```
Résultats de la Requête 3 (Indice de Développement Agricole Routier par
région):
                          nom_region  superficie  superficie_region_km2
\
0          Région de Rabat-Salé-Kénitra      208000           17666.039440
1           Région de Casablanca-Settat      168238           20287.585538
2              Région de Marrakech-Safi      345032           38964.518576
3        Région de Béni Mellal-Khénifra      226293           27677.341743
4   Région de Tanger-Tétouan-Al Hoceima       66174           16200.491418
5                 Région de Fès-Meknès      193542           38998.349269
6               Région de Souss-Massa      174862           53602.106421
7                Région de l'Oriental      181388           66621.571705
8            Région de Drâa-Tafilalet      201922           86167.268820
9          Région de Guelmim-Oued Noun       36602           44602.902370
10    Région de Laâyoune-Sakia El Hamra        5543          143749.447657
11       Région de Dakhla-Oued Ed-Dahab        1152          130212.114590
```

```
    densite_routiere_par_100km2  indice_developpement_agricole_route
0                    191.108297                            22.501096
1                    267.353359                            22.170699
2                    177.217715                            15.692683
3                    133.460936                            10.911913
4                    191.850180                             7.836487
5                    111.156284                             5.516492
6                    100.326816                             3.272884
7                     63.719386                             1.734863
8                     54.699161                             1.281805
9                     41.494012                             0.340508
10                    14.970201                             0.005773
11                     6.598440                             0.000584
```

-Carte :



Remarque : Ces requêtes ont été exécuter dans Googlecolab et n'ont pas pris beaucoup temps, maximum 30 à 40 min au contraire à QGIS qui prend des heures d'exécution.

8 DISCUSSION CRITIQUE (LIMITES DES DONNEES, DIFFICULTES, PERSPECTIVES).

La réalisation de ce projet a mis en évidence plusieurs limites liées aux données et difficultés techniques rencontrées tout au long du processus :

8.1 LIMITES DES DONNEES

- Les données issues du Ministère de l'Agriculture ne sont pas fournies dans un format homogène ni structuré, ce qui a rendu leur extraction et leur intégration complexes.
- Certaines couches présentaient des incohérences spatiales : différences de systèmes de projection, géométries incomplètes ou mal définies.
- D'autres limitaient l'analyse temporelle à cause de mises à jour irrégulières.
- L'intégration de sources multiples a nécessité un travail important de nettoyage, reprojection et harmonisation pour assurer la cohérence spatiale des couches.

8.2 DIFFICULTES TECHNIQUES

- La manipulation de grandes couches vectorielles dans PostgreSQL/PostGIS a engendré des temps de calcul élevés, notamment lors d'opérations de jointures et d'intersections spatiales complexes.
- L'optimisation des requêtes a demandé une bonne compréhension de la logique spatiale, ainsi l'utilisation du concept du 'bounding box' a contribué d'une manière remarquable dans la diminution du temps d'exécution de quelques requêtes.

> **5.6.2 Bounding box and geometry operators**
>
> PostGIS offers a number of geometry bounding box comparators that work exclusively with box2d objects and one comparator that works against the actual geometry. Some but not all of these operators have functional counterparts that apply to the entire geometry. As a convenient shorthand, PostGIS uses various operators to symbolize comparators. For example, `A && B` returns true if bounding box of geometry A intersects bounding box of geometry B or vice versa where the double ampersand operator (&&) is the intersection comparator. The && operator is the one most commonly used as a precheck for spatial relationships.

Post GIS in Action_page169

- Parfois l'échange de la BDS était difficile à cause de sa taille, ainsi son intégration au niveau de PgAdmin.
- Difficultés : L'exécution dans Qgis prend beaucoup de temps (requête 1 a pris 6heures)

8.3 PERSPECTIVES D'AMELIORATION

- Une meilleure intégration des outils de visualisation interactive pourrait renforcer la dimension décisionnelle du SIG.
- L'usage futur de moteurs analytiques modernes (DuckDB, BigQuery GIS) en complément de PostGIS pourrait améliorer la vitesse de traitement pour des analyses à grande échelle.
- On peut ainsi tirer d'autres avantages à partir de l'utilisation de DuckDB si on veut créer une base de données dynamique, qui peut êtres mise à jour d'une manière continue (possibilité d'intégration de l'IA pour remplire la base), à travers les avantages suivants :

    - Performances locales et temps réel
    - Mise à jour facile des données (sans serveur)

- Interopérabilité et formats modernes
- Combinaison parfaite avec PostGIS
- Intégration facile avec des outils de visualisation

Et de cette manière on peut créer une carte qui peut être plus durable.

9 DEPOT GITHUB DU PROJET

Le script SQL, les fichiers de données ainsi que le projet QGIS associés à ce travail sont accessibles en ligne :

- Titre du dépôt : Projet-BDS
- Lien : https://github.com/salwaafroukh/Projet-BDS.git

10 BIBLIOGRAPHIE :

- *Post GIS in Action*
- *SPATIAL SQL APRACTICAL APPROACH TO MODERN GIS USING SQL_* MATTHEW FORREST _
- *https://fr.wikipedia.org/wiki/R-arbre*
- *https://en.wikipedia.org/wiki/R*-tree*
- *https://colab.research.google.com/*
- https://download.geofabrik.de/africa/morocco.html
- https://www.geodatika.com/routes
- https://www.agriculture.gov.ma/fr
- https://sig-maroc.com/