

Traitement de données géospatiales avec Parquet et DuckDB



Développeur SIG chez Oslandia

@florentfougeres

@florentfougeres

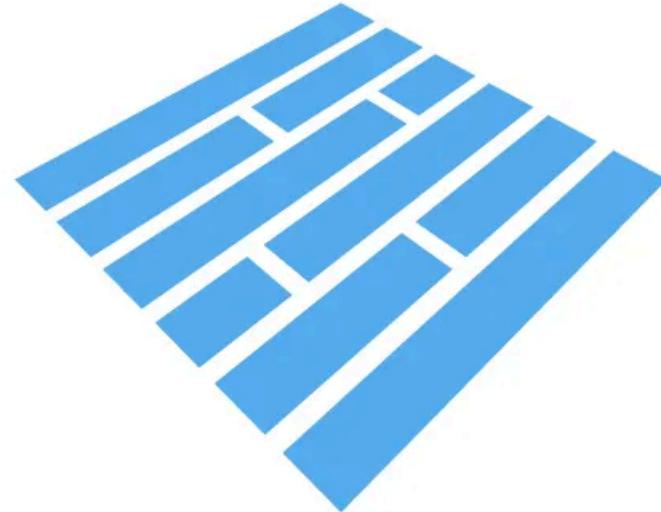
florent.fougeres@gmail.com

Florent Fougères

Plan du cours

1. Introduction au format Parquet
2. Le SGBD DuckDB : présentation et caractéristiques
3. Performances et benchmarks
4. Cas d'usage en SIG
5. La complémentarité Parquet + DuckDB
6. Exercices pratiques

1. Introduction au format Parquet



Parquet

Qu'est-ce que Parquet ?

- Format de fichier **open source**
- Développé par la **Fondation Apache**
- Conçu pour stocker et accéder à de **grandes quantités de données**
- **Architecture orientée colonnes**
- **Compression efficace**
- Extension **Geoparquet** pour les géométries spatiales

Architecture orientée colonnes

Format traditionnel (orienté lignes) :

```
Ligne 1: nom, prénom, ville, code_postal  
Ligne 2: nom, prénom, ville, code_postal  
Ligne 3: nom, prénom, ville, code_postal
```

Parquet (orienté colonnes) :

```
Colonne nom: [valeur1, valeur2, valeur3, ...]  
Colonne prénom: [valeur1, valeur2, valeur3, ...]  
Colonne ville: [valeur1, valeur2, valeur3, ...]
```

Avantages de l'architecture colonnes

Meilleure compression

- Les données d'une même colonne sont souvent similaires

Lecture optimisée

- Ne lit que les colonnes nécessaires

Performance analytique

- Idéal pour les agrégations et statistiques

Organisation en dataset

Un dataset Parquet peut être un **dossier de fichiers** :

```
/mon_dataset/
└── partie_1.parquet
└── partie_2.parquet
└── partie_3.parquet
└── partie_4.parquet
```

Avantages :

- Traitement en parallèle (chunks)
- Gestion efficace de gros volumes
- Meilleure performance globale

Geoparquet

<https://geoparquet.org>

Extension de Parquet pour les **données géospatiales**

- Stockage des géométries spatiales
- Compatible avec tous les types géométriques
- Métadonnées spatiales (CRS, bbox, etc.)
- Tous les avantages de Parquet préservés

Benchmark : Comparaison des formats

Données : Localités mondiales > 1 000 habitants

Format	Taille du fichier
GeoJSON	~100 MB
CSV	~80 MB
Parquet	~20 MB

→ Gain de taille : 5x plus compact que GeoJSON

Limite importante de Parquet

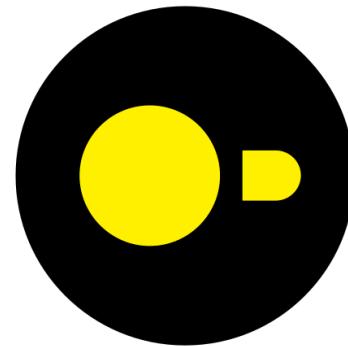
⚠️ Parquet = format d'échange

- Pas un format de travail
- Pas un format de traitement
- Nécessite un système optimisé pour l'exploiter

Solution : DuckDB

02

Le SGBD DuckDB : présentation et caractéristiques



DuckDB

Qu'est-ce que DuckDB ?

Système de gestion de base de données relationnelle (SGBDR)

- Écrit en C++
- Licence MIT (open source)
- Base de données en fichier portable
- Version stable : 1.4.4
- Sans serveur (serverless)

Pourquoi choisir DuckDB ?

Sans serveur (Serverless)

- Pas de processus serveur à gérer
- Configuration minimale

Installation rapide

- Mise en place en quelques minutes

Fichier portable

- Facilite le partage et la sauvegarde

Pourquoi choisir DuckDB ? (suite)

Langage SQL standard

- Facile si vous connaissez SQL
- Nombreuses fonctions analytiques

Lecture directe de fichiers

- Requêtes directes sur Parquet, CSV, etc.
- Pas d'import nécessaire

Extension spatiale

- Fonctions GIS basées sur GEOS
- Compatible avec PostGIS

Extension spatiale de DuckDB

Fonctionnalités avancées pour le SIG :

- Grande variété de fonctions spatiales (GEOS)
- **Noms identiques à PostGIS** (migration facilitée)
- **Indexation spatiale**
- Lecture/écriture de nombreux formats
- Support des **drivers GDAL**

Installation de l'extension spatial

```
INSTALL spatial;
```

A faire qu'une seule fois sur le poste

Chargement de l'extension spatial

```
LOAD spatial;
```

Il s'agit de l'équivalent du `CREATE EXTENSION postgis` sur une base dans PostgreSQL

OLAP vs OLTP

DuckDB est optimisé pour l'OLAP

https://fr.wikipedia.org/wiki/Traitement_analytique_en_ligne

En informatique, et plus particulièrement dans le domaine des bases de données, le traitement analytique en ligne (anglais online analytical processing, OLAP) est un type d'application informatique orienté vers l'analyse sur-le-champ d'informations.

- OLAP: Traitement analytique en ligne (*online analytical processing*)
- OLTP: Traitement transactionnel en ligne (*online transaction processing*)

OLAP vs OLTP

OLAP (DuckDB)	OLTP (PostgreSQL)
Requêtes analytiques complexes	Transactions rapides
Lecture de gros volumes	Nombreuses écritures
Peu de mises à jour	Mises à jour fréquentes
Agrégations et analyses	Intégrité transactionnelle

Performance : Architecture colonnes

DuckDB excelle pour :

- Traitement de **grands volumes** de données
- Données de **plusieurs gigaoctets**
- Tables de **plusieurs millions de lignes**
- **Requêtes analytiques** et agrégations

Benchmark : DuckDB vs PostGIS

Données : 13,1 millions de bâtiments (GeoJSON 6,3 GB)

Opération	DuckDB	PostGIS
Intégration	4 min	5 min 30 s
Calcul de buffer	38 s	1 min 54 s
Ajout colonne + surface	2 s	2 min 58 s

⚠ DuckDB ne remplace pas PostGIS

Ils ont des cas d'usage différents et complémentaires.

Limitations de DuckDB (1/2)

✗ Pas de support des projections

- Pas de transformation de systèmes de coordonnées
- Données doivent être dans la même projection

✗ Mono-utilisateur et local

- Usage local uniquement
- Pas de multi-utilisateurs simultanés

✗ Verrouillage en lecture

- Base verrouillée lors des lectures

Limitations de DuckDB (2/2)

✗ Pas de gestion des droits

- Pas de système de permissions
- Pas de gestion de rôles

✗ Projet jeune

- Évolution rapide
- API peut changer

✗ OLAP uniquement

- Pas pour : écritures fréquentes, transactions concurrentes, haute disponibilité

Utilisation de DuckDB

Interface en ligne de commande (CLI)

- [Linux, macOS, Windows](#)

Langages de programmation

- [Python, R, Java, Node.js, Rust...](#)

Dans QGIS

- [QDuckDB](#)

Utilisation de DuckDB

IDE

- DBeaver
- DataGrip (freeware)
- Beekeeper Studio

Web

Permet grâce à DuckDB WASM

- DuckDB Shell
- DuckUI
- Avec la CLI via `duckdb -ui`

03

Cas d'usage en SIG

Plugin QDuckDB pour QGIS

Intégration de DuckDB dans QGIS

- Développement financé par l'IFREMER
- Mode **lecture seule** actuellement
- Chargement de couches depuis DuckDB
- Disponible dans le gestionnaire d'extensions QGIS

 Vidéo de démonstration

Exemple 1 : Créer une table géographique depuis un CSV

Créer une couche spatiale depuis un CSV avec X,Y :

```
SELECT
  *,
  ST_Point(x, y) as geometry
FROM read_csv('fichier.csv')
WHERE x IS NOT NULL
  AND y IS NOT NULL;
```

Exemple 2 : Conversion Shapefile → GPKG

Convertir un Shapefile en GeoPackage :

```
COPY (
    SELECT * FROM ST_Read('input.shp')
) TO 'output.gpkg'
WITH (FORMAT GDAL, DRIVER 'GPKG');
```

Exemple 3 : Parquet → Shapefile avec filtre

Créer un Shapefile depuis Parquet avec filtre spatial :

```
COPY (
    SELECT *
    FROM read_parquet('data.parquet')
    WHERE ST_Within(
        geometry,
        ST_MakeEnvelope(xmin, ymin, xmax, ymax)
    )
) TO 'filtered.shp'
WITH (FORMAT GDAL, DRIVER 'ESRI Shapefile');
```

Génération de tuiles vectorielles

Nouveauté depuis DuckDB 1.4

Avantages :

- Performance élevée pour la génération de tuiles
- Intégration facile dans des pipelines
- Alternative légère aux solutions traditionnelles

Exemple par Max Gabrielsson

04

Complémentarité Parquet + DuckDB

Le duo gagnant

Format ouvert (Parquet)

- Stockage compact
- Compression efficace
- Portabilité

+

Base de données in-process (DuckDB)

- Requêtes SQL puissantes
- Performance élevée
- Facilité d'utilisation

= Efficacité maximale

Workflow recommandé

1. Stockage en Parquet

- Compression et archivage
- Partage et échange de données

2. Traitement avec DuckDB

- Analyses et requêtes complexes
- Transformations et conversions

3. Export vers format métier

- GeoPackage pour QGIS
- Shapefile pour compatibilité
- GeoJSON pour le web

Cas d'usage idéaux

DuckDB + Parquet est parfait pour :

- Analyse exploratoire de données (EDA)
- Prototypage rapide d'analyses spatiales
- Traitement batch de grandes données
- Conversions de formats en masse
- Génération de rapports analytiques
- Scripts ETL
- Travail en local sur données volumineuses
- Recherche et développement

Cas d'usage à éviter

Ne PAS utiliser pour :

- Applications web avec accès concurrent
- Systèmes nécessitant haute disponibilité
- Applications transactionnelles (CRUD intensif)
- Cas nécessitant gestion fine des droits utilisateurs

05

Conclusion

Points clés à retenir

1. **Parquet** : format de stockage efficace pour grandes données
2. **DuckDB** : SGBD performant pour l'analyse volumineuse
3. **Extension spatiale** : capacités GIS avancées
4. **Parquet + DuckDB** : duo idéal pour l'analyse géospatiale
5. **Complémentarité** : ne remplace pas PostgreSQL/PostGIS

Perspectives

DuckDB est un projet jeune et dynamique

- Évolution rapide
- Écosystème grandissant
- Nouvelles fonctionnalités régulières
- Intégrations multiples

→ Un outil à surveiller de près dans le domaine géospatial

Ressources

Documentation et communauté :

- Site officiel : <https://duckdb.org>
- Documentation : <https://duckdb.org/docs>
- Extension spatiale : <https://duckdb.org/docs/extensions/spatial>
- GitHub : <https://github.com/duckdb/duckdb>
- Une liste sélectionnée de bibliothèques, d'outils et de ressources DuckDB :
<https://github.com/davidgasquez/awesome-duckdb>
- Plugin QGIS QDuckDB : <https://plugins.qgis.org/plugins/qduckdb/>
- Article Géotribu : https://geotribu.fr/articles/2023/2023-12-19_duckdb-donnees-spatiales/