



Toulouse-DataViz

DuckDB nous ouvre grand les portes de l'open data

Meetup Toulouse Dataviz

25 janvier 2024

Éric Mauvière, icem7



What the duck is that !?!

Data Science is Getting Ducky

19 Dec 2023 PAUL RAMSEY Cocréateur de PostGIS



Tim Schaub

@tschaub · Follow

I can't believe I survived without @DuckDB for so long. Such an awesome piece of tech. Feels indispensable already.

10:39 PM · Sep 8, 2023



55



Reply



Share

Duck DB brings the excitement back to data warehouses (almost)



Hugo Lu · Follow

9 min read · Sep 21, 2023



159



2



DuckDB ouvre grand les portes de l'open data

Ingénieur IGN



Jean-Marc Viglino @jmviglino · 9 janv.

Belle utilisation de @duckdb pour l'analyse du réseau de transport 🚗
#GTFS #BAN #Macarte #SpatialAnalysis 🗺️ #mapping #OpenData 📄



DuckDB: The Indispensable Geospatial Tool You Didn't Know You Were Missing

September 12, 2023

By [Chris Holmes](#), Radiant Earth Technical Fellow



Jared Lander

@jaredlander

Just used @duckdb take 20 million rows out of Postgres into a parquet file in 18 seconds. So much faster than anything I have done before. This tool is getting better and better all the time.

[Traduire le post](#)

3:27 PM · 22 févr. 2023 · 25,5 k vues

Vous avez dit bluffant ?

Ca y est !! J'ai réussi à reproduire ton output.

C'est absolument bluffant de puissance.

Sur ma machine, en 30 sec d'exécution de la requête il parse les 512Mo comme une fleur.

Alain Roan <alain.roan@

J'ai fait quelques tests avec duckdb, c'est bluffant et j'ai encore beaucoup de points à éclaircir. Vivement le meetup.

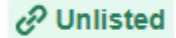
aottenheimer <aottenheimer@

Une comparaison dans Observable



Alain Ottenheimer

Data Analysis, Data Viz Javascript, python, QGIS, Tableau, illustrator...



Edited Jan 19 2 forks 1 Like

EXPLORATEUR DE L'HISTORIQUE DES DONNÉES CLIMATIQUES (beta)

Historique des températures de localités françaises de 1900-2020 à partir des données Météo-France.

Exploitation d'un fichier
de 570 000 observations de 1900 à 2023

» Explorateur de l'historique des données climatiques - Alain Ottenheimer

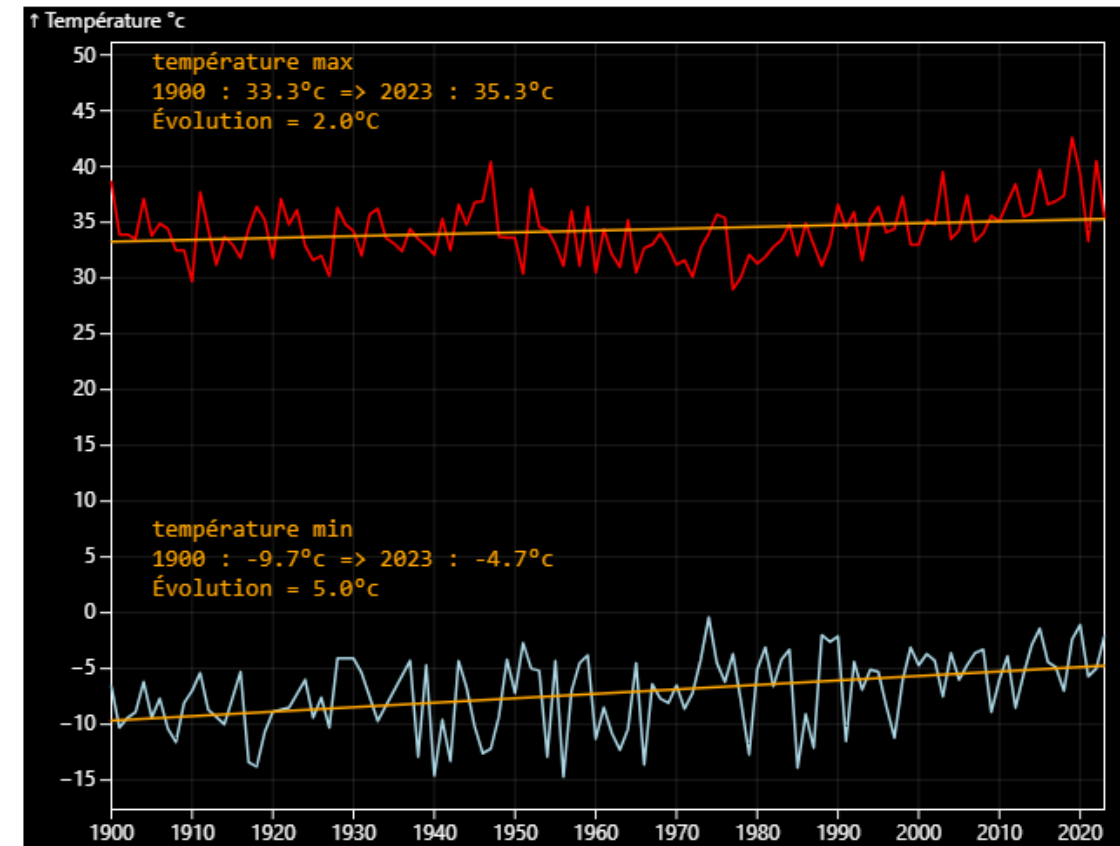
Choisir une ville

PARIS-MONTSOURIS (75)

Choisir l'année de
départ

1900

GRAPHICS



Dans Observable, DuckDB est 3 fois plus rapide que JavaScript

Pur JavaScript + CSV

```
meteo_fr = FileAttachment("Meteo_france_selection_historique@5.csv").csv()
```

```
data = {  
  const parseDate = d3.utcParse("%Y-%m-%d");  
  const temp = meteo_fr.map((row) => ({  
    localite: row.NOM_USUEL,  
    date: parseDate(row.DATE),  
    annee: +d3.timeFormat("%Y")(parseDate(row.DATE)),  
    mois: d3.timeFormat("%m")(parseDate(row.DATE)),  
    jour: d3.timeFormat("%d")(parseDate(row.DATE)),  
    jour_annee: d3.timeFormat("%j")(parseDate(row.DATE)),  
    tmin: row.TN !== "" ? +row.TN : -99,  
    tmax: row.TX !== "" ? +row.TX : -99,  
    dg: row.DG !== "" ? +row.DG : -99,  
    rr: row.RR !== "" ? +row.RR : -99  
  }));  
  return temp.filter((d) => d.annee >= annee_debut && d.localite == localite);  
}
```

» Historique des données climatiques - test JS

DuckDB + Parquet

```
db = DuckDBClient.of({  
  meteo_fr: FileAttachment("Meteo_france_selection_historique.parquet")  
})
```

```
data = db.query(`FROM meteo_fr  
SELECT nom_usuel as localite, "DATE" AS "date",  
year(date) AS annee, month(date) AS mois,  
day(date) AS jour, dayofyear(date) AS jour_annee,  
IFNULL(TN, -99) AS tmin,  
IFNULL(TX, -99) AS tmax,  
IFNULL(DG, -99) AS dg,  
IFNULL(RR, -99) AS rr  
WHERE annee >= ? AND localite = ?`, [+annee_debut, localite])
```

Lisibilité ++
Bande passante -
Temps d'exécution ---

» Historique des données climatiques - test DuckDB

Qui suis-je ?

Chargé d'études,
responsable
de **publications**
régionales



10 ans

Développeur
d'un logiciel
de **cartographie**
statistique



20 ans

Formateur en sémiologie,
rédacteur de récits
statistiques et graphiques ;
formateur R-tidyverse
certifié RStudio



3 ans

Programme de l'intervention

- 1 – Comment utiliser DuckDB dans divers environnements
- 2 – Quelques points forts de DuckDB
- 3 – D'où ça sort ?
- 4 – Formats orientés colonnes et streamables
- 5 – Limites et perspectives

#1

Comment manier DuckDB dans différents environnements

DuckDB quesaco ?

Un moteur SQL dans votre navigateur web, Observable

The screenshot shows the DuckDB Web Shell interface. At the top, the browser address bar displays 'shell.duckdb.org'. The main content area has a dark background and shows the following text:

DuckDB Web Shell
 Database: v0.9.2
 Package: @duckdb/duckdb-wasm@1.28.1-dev73.0

Connected to a local transient in-memory database.
 Enter .help for usage hints.

The user has entered the following SQL query in the prompt:

```
duckdb> FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')
...> WHERE CODGEO = '31410';
```







The query results are displayed in a table with 7 columns: CODGEO, CODDEP, LIBGEO, PMUN, PTOT, PCAP, and ANNEE_RP. The results show 6 rows of data for the CODGEO '31410'.

CODGEO	CODDEP	LIBGEO	PMUN	PTOT	PCAP	ANNEE_RP
31410	31	Pechbonnieu	4425	4511	86	2017
31410	31	Pechbonnieu	4429	4515	86	2018
31410	31	Pechbonnieu	4544	4633	89	2019
31410	31	Pechbonnieu	4624	4714	90	2020
31410	31	Pechbonnieu	4662	4752	90	2021

On the right side, there is a preview panel showing the first two columns: CODGEO and CODDEP. It indicates that there are 34,997 categories for CODGEO and 100 categories for CODDEP.

Appli web WASM

<https://shell.duckdb.org>

	CODGEO string	CODDEP string	LIBGEO string	PMUN number	PTOT number	PCAP number
	 34,997 categories	 100 categories	 33,232 categories	 0 550k	 0 550k	 0 550
0	01001	01	L'Abergement-Clémenci	776	794	
1	01001	01	L'Abergement-Clémenci	771	789	
2	01001	01	L'Abergement-Clémenci	779	798	
3	01001	01	L'Abergement-Clémenci	806	821	

174 928 rows

pop_legales memory_db

```
FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9') ;
```

DuckDB quesaco ?

Un moteur SQL sans dépendance = facilement installable.

Un petit programme autonome, 20 Mo : exécutable Windows, Mac, Linux

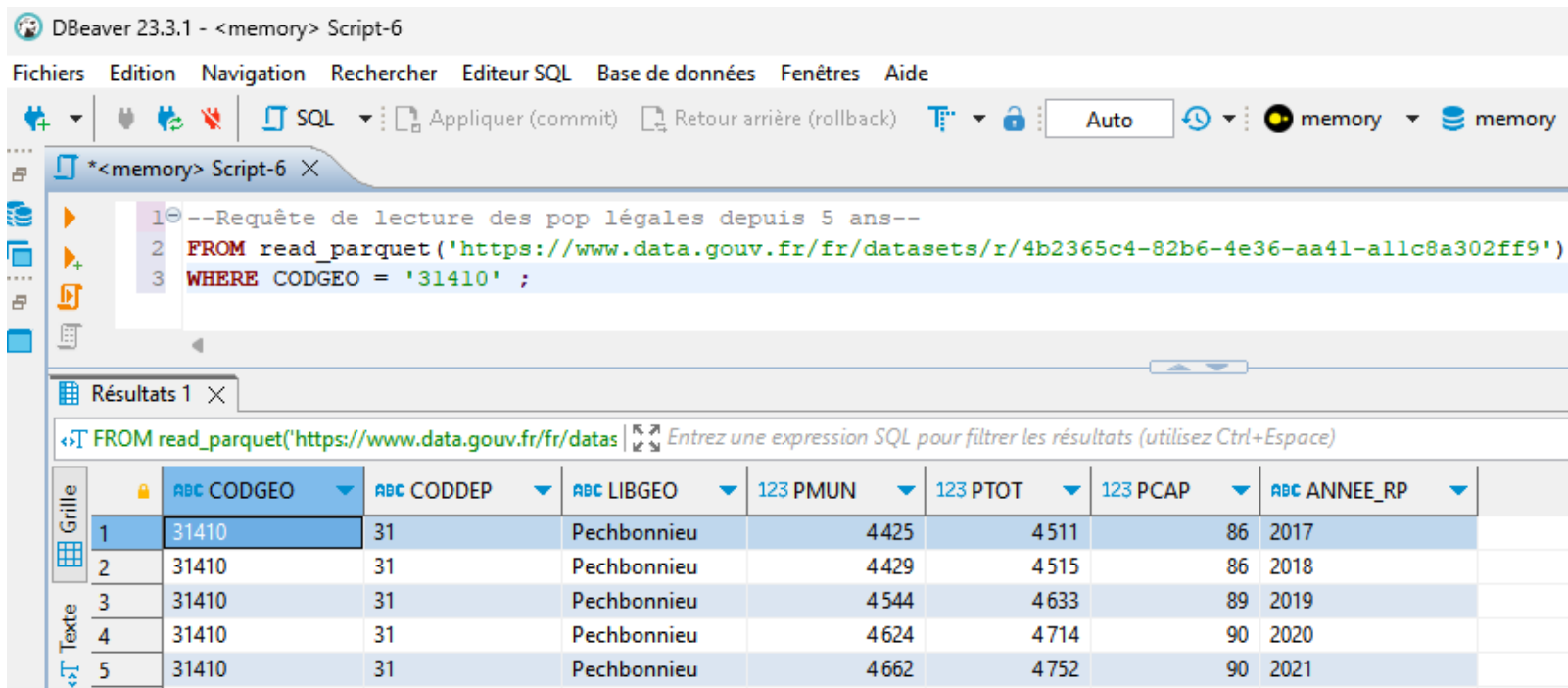
```
C:\apps\duckdb\duckdb.exe X + v
v0.9.2 3c695d7ba9
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
D FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')
> WHERE CODGEO = '31410' ;
```

CODGEO varchar	CODDEP varchar	LIBGEO varchar	PMUN int64	PTOT int64	PCAP int64	ANNEE_RP varchar
31410	31	Pechbonnieu	4425	4511	86	2017
31410	31	Pechbonnieu	4429	4515	86	2018
31410	31	Pechbonnieu	4544	4633	89	2019
31410	31	Pechbonnieu	4624	4714	90	2020
31410	31	Pechbonnieu	4662	4752	90	2021

» DuckDB installation

DuckDB quesaco ?

Un moteur SQL utilisable dans un éditeur convivial et libre, pour gérer ses scripts.



The screenshot shows the DBeaver 23.3.1 interface. The top menu bar includes Fichiers, Edition, Navigation, Rechercher, Editeur SQL, Base de données, Fenêtres, and Aide. Below the menu is a toolbar with icons for file operations and a status bar showing 'Auto' and 'memory'. The main editor displays a SQL script:

```
1 --Requête de lecture des pop légales depuis 5 ans--
2 FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')
3 WHERE CODGEO = '31410' ;
```

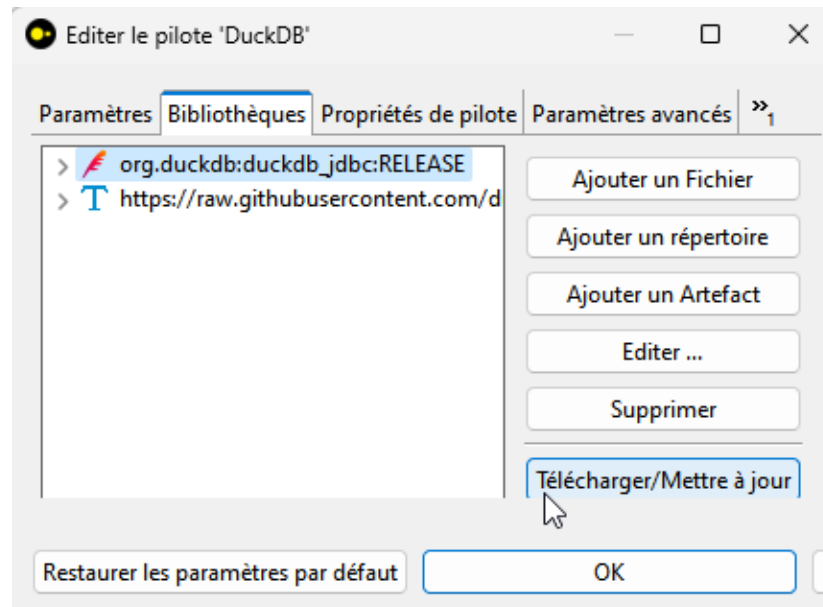
Below the script, the 'Résultats 1' tab shows a table with 8 columns: ABC CODGEO, ABC CODDEP, ABC LIBGEO, 123 PMUN, 123 PTOT, 123 PCAP, and ABC ANNEE_RP. The table contains 5 rows of data.

	ABC CODGEO	ABC CODDEP	ABC LIBGEO	123 PMUN	123 PTOT	123 PCAP	ABC ANNEE_RP
1	31410	31	Pechbonnieu	4425	4511	86	2017
2	31410	31	Pechbonnieu	4429	4515	86	2018
3	31410	31	Pechbonnieu	4544	4633	89	2019
4	31410	31	Pechbonnieu	4624	4714	90	2020
5	31410	31	Pechbonnieu	4662	4752	90	2021

DBeaver

» DBeaver Community

Librairie jdbc
facile à mettre à jour



The screenshot shows the 'Edit driver DuckDB' dialog box. The 'Bibliothèques' tab is selected, showing a list of libraries with the following entries:

- > org.duckdb:duckdb_jdbc:RELEASE
- > https://raw.githubusercontent.com/d

On the right side, there are buttons for 'Ajouter un Fichier', 'Ajouter un répertoire', 'Ajouter un Artefact', 'Editer ...', and 'Supprimer'. At the bottom, there is a button labeled 'Télécharger/Mettre à jour' which is highlighted by a mouse cursor. Other buttons at the bottom include 'Restaurer les paramètres par défaut' and 'OK'.

DuckDB dans R

```
library(duckdb)
```

```
con <- dbConnect(duckdb())  
dbExecute(con, 'load httpfs')
```

```
dbGetQuery(con, "  
FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')  
WHERE codegeo = '31410'  
ORDER BY annee_rp  
")
```

	codegeo	libgeo	coddep	codreg	pmun	ptot	pcap	annee_rp
1	31410	Pechbonnieu	31	76	4425	4511	86	2017
2	31410	Pechbonnieu	31	76	4429	4515	86	2018
3	31410	Pechbonnieu	31	76	4544	4633	89	2019
4	31410	Pechbonnieu	31	76	4624	4714	90	2020
5	31410	Pechbonnieu	31	76	4662	4752	90	2021

```
library(duckdb)  
library(dplyr)
```

```
con <- dbConnect(duckdb())  
dbExecute(con, 'load httpfs')
```

```
tbl(con,  
  "read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')") |>  
  filter(codegeo == "31410") |>  
  arrange(annee_rp) |>  
  collect()
```

Syntaxe SQL

Syntaxe dplyr

DuckDB dans Python

```
import duckdb
import pandas as pd

rp_df = duckdb.sql("""
    FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')
    WHERE codegeo = '31410'
    ORDER BY annee_rp
    """).df()

rp_df
```

	codegeo	libgeo	coddep	codreg	pmun	ptot	pcap	annee_rp
0	31410	Pechbonnieu	31	76	4425	4511	86	2017
1	31410	Pechbonnieu	31	76	4429	4515	86	2018
2	31410	Pechbonnieu	31	76	4544	4633	89	2019
3	31410	Pechbonnieu	31	76	4624	4714	90	2020
4	31410	Pechbonnieu	31	76	4662	4752	90	2021

```
import duckdb
import polars as pl

rp_df = (
    duckdb.sql("FROM read_parquet('https://www.data.gouv.fr/fr/datasets/r/4b2365c4-82b6-4e36-aa41-a11c8a302ff9')")
    .filter("codegeo = '31410'")
    .sort('annee_rp')
    .pl()
)

rp_df
```

shape: (5, 8)

codegeo	libgeo	coddep	codreg	pmun	ptot	pcap	annee_rp
str	str	str	str	i64	i64	i64	str
"31410"	"Pechbonnieu"	"31"	"76"	4425	4511	86	"2017"
"31410"	"Pechbonnieu"	"31"	"76"	4429	4515	86	"2018"
"31410"	"Pechbonnieu"	"31"	"76"	4544	4633	89	"2019"
"31410"	"Pechbonnieu"	"31"	"76"	4624	4714	90	"2020"
"31410"	"Pechbonnieu"	"31"	"76"	4662	4752	90	"2021"

Syntaxe SQL
et sortie pandas

Syntaxe mixte
et sortie polars

Open data Météo-France

```
FROM 'https://object.files.data.gouv.fr/meteofrance/data/synchro_ftp/BASE/HOR/H_31_latest-2023-2024.csv.gz'  
SELECT NOM_USUEL, ALTI, AAAAMMJJHH,  
strptime(AAAAMMJJHH, '%Y%m%d%H') AS horaire, T  
WHERE NOM_USUEL ILIKE '%blagnac%' AND T IS NOT NULL;
```

ABC NOM_USUEL	123 ALTI	123 AAAAMMJJHH	🕒 horaire	123 T
TOULOUSE-BLAGNAC	151	2023 010 100	2023-01-01 00:00	12,8
TOULOUSE-BLAGNAC	151	2023 010 101	2023-01-01 01:00	12,9
TOULOUSE-BLAGNAC	151	2023 010 102	2023-01-01 02:00	12,8
TOULOUSE-BLAGNAC	151	2023 010 103	2023-01-01 03:00	12,6
TOULOUSE-BLAGNAC	151	2023 010 104	2023-01-01 04:00	12,7
TOULOUSE-BLAGNAC	151	2023 010 105	2023-01-01 05:00	12,6

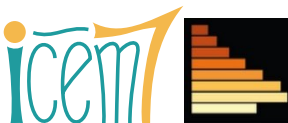
Copier/coller à partir
de DBeaver -> Datawrapper

Courbe de température à Toulouse-Blagnac

Ces 10 derniers jours, à partir de meteo.data.gouv.fr



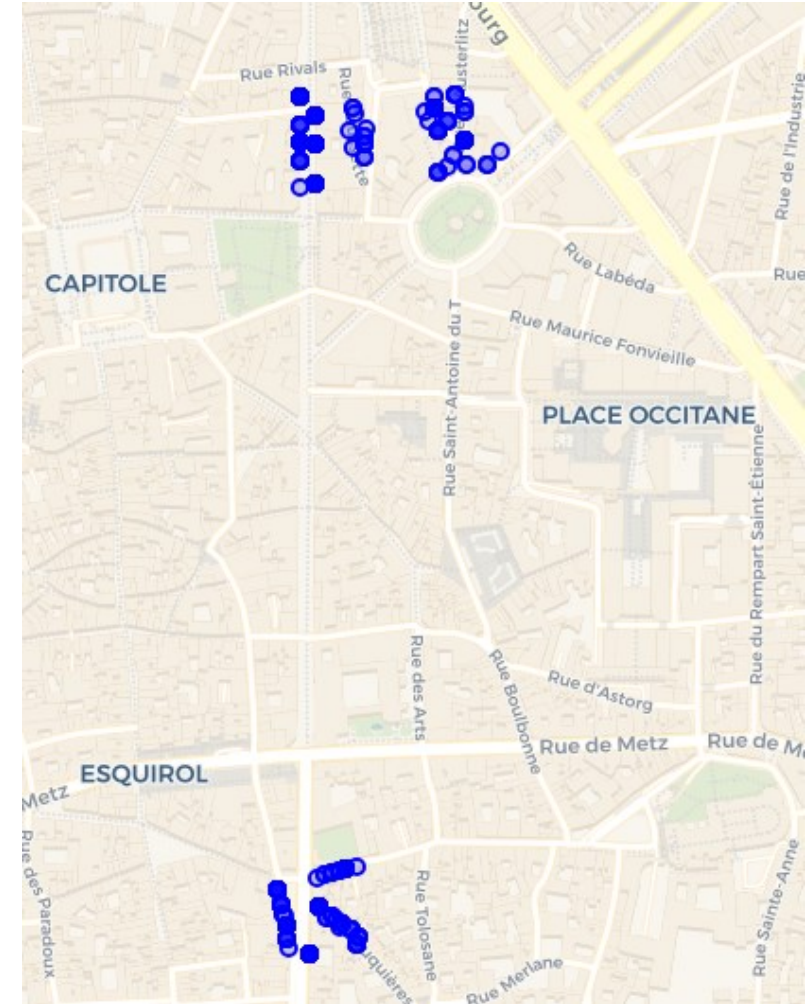
Source: Météo - France



Sirene géolocalisé – autour d'Étincelle coworking à Toulouse

```
FROM sirenegeo CROSS JOIN etincelle
SELECT ST_distance(
etincelle.geometry.ST_geomFromWkb().ST_transform('EPSG:4326', 'EPSG:2154', true),
sirenegeo.geometry.ST_geomFromWkb().ST_transform('EPSG:4326', 'EPSG:2154', true)
)::int AS distance,
siret, denominationUniteLegale AS nom, activitePrincipaleEtablissement AS NIV5,
geo_adresse, sirenegeo.geometry
WHERE codeCommuneEtablissement = '31555' AND distance < 50 ;
```

Une requête spatiale pour lister et visualiser dans DBeaver les établissements Sirene à proximité des 3 sites d'Étincelle.



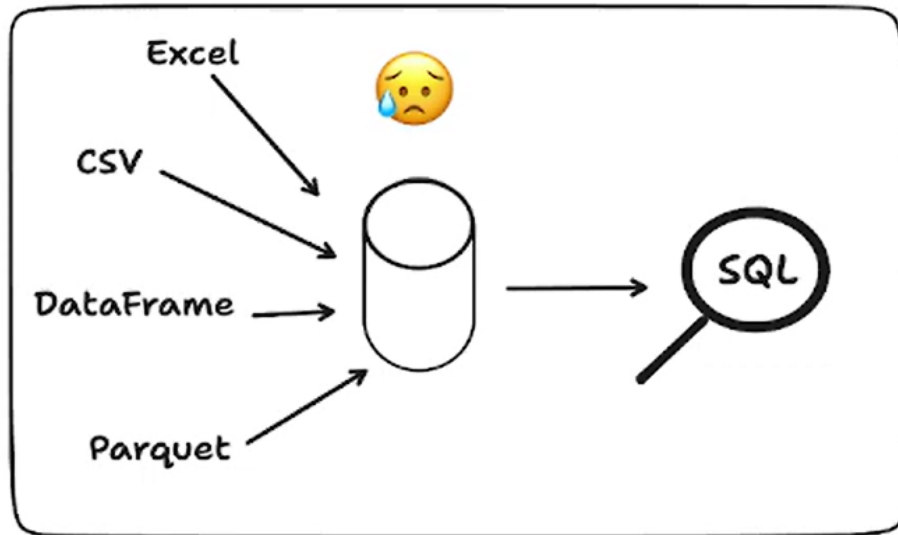


Quelques points forts de DuckDB

dans différents environnements

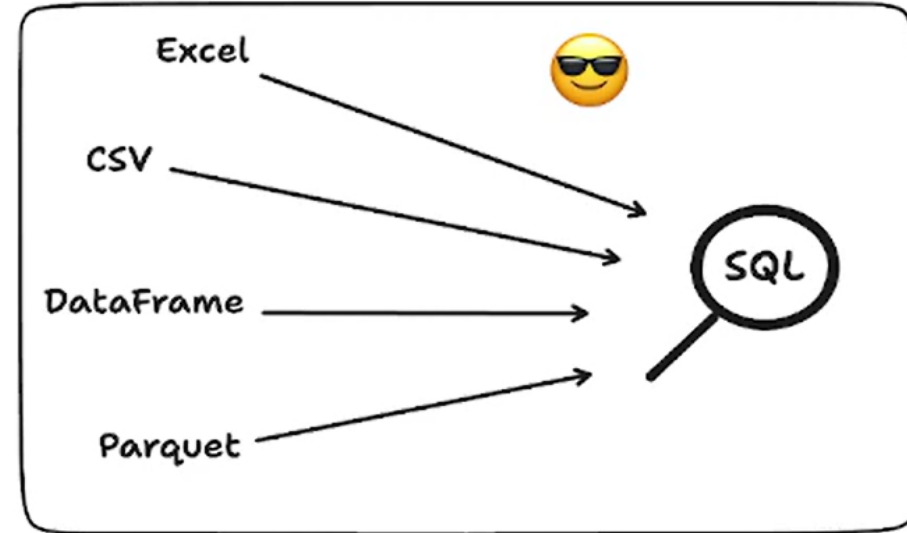
Séparer stockage et requêtes, simplifier les deux univers

Schéma classique : on importe (duplique) un tas de ressources dans une base, puis on requête sur/dans cette base.



Coûteux en espace, infra, maintenance, mises à jour, sauvegardes...

Avec un client intelligent, les données sont là où leurs producteurs savent les gérer, l'appareil de l'utilisateur les interroge directement.

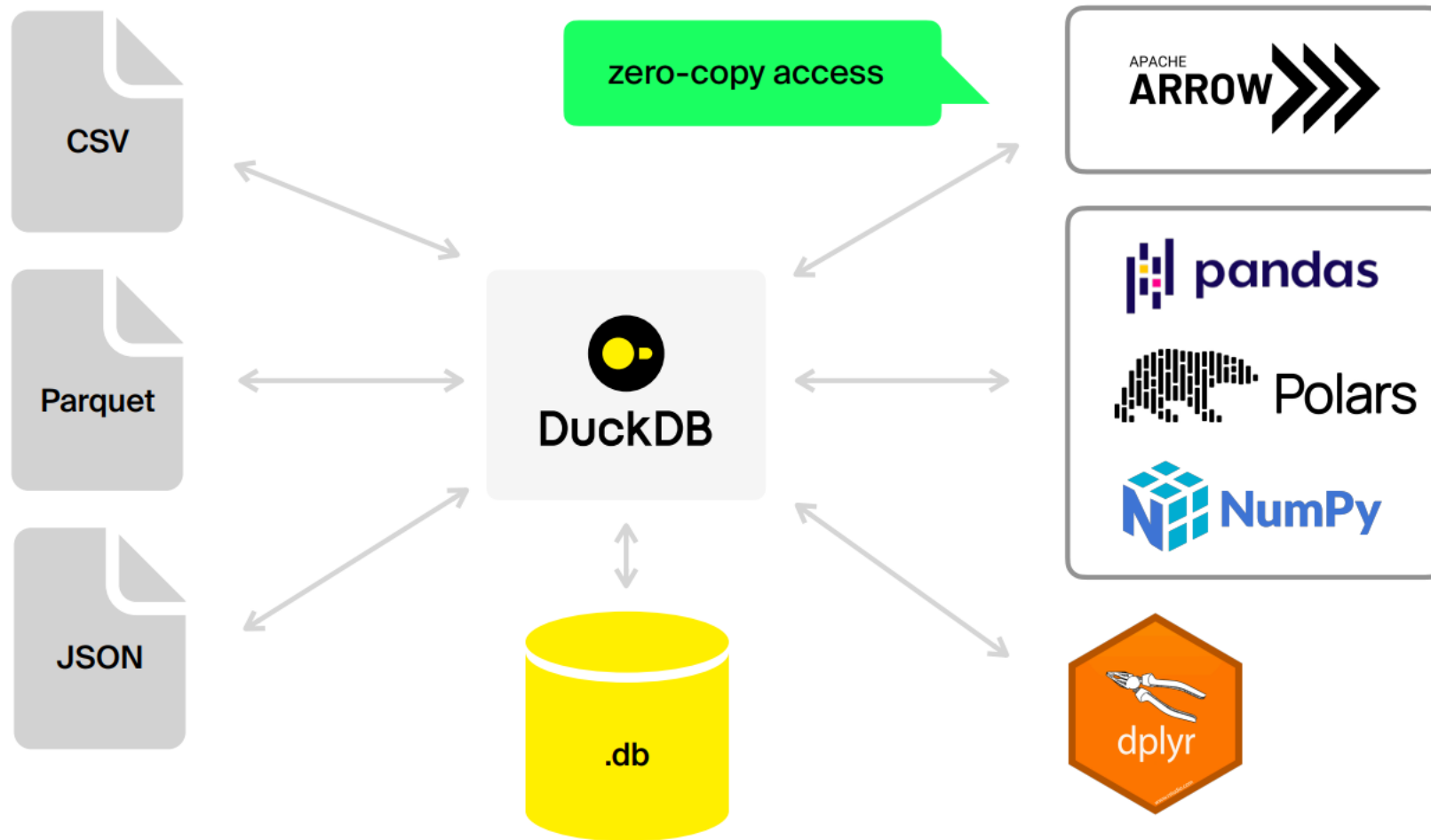


Décentralisation, économie, agilité

» [Unlocking Valuable Sports Insights: Analyzing SportsCode XML Files with DuckDB and Kloppey](#)

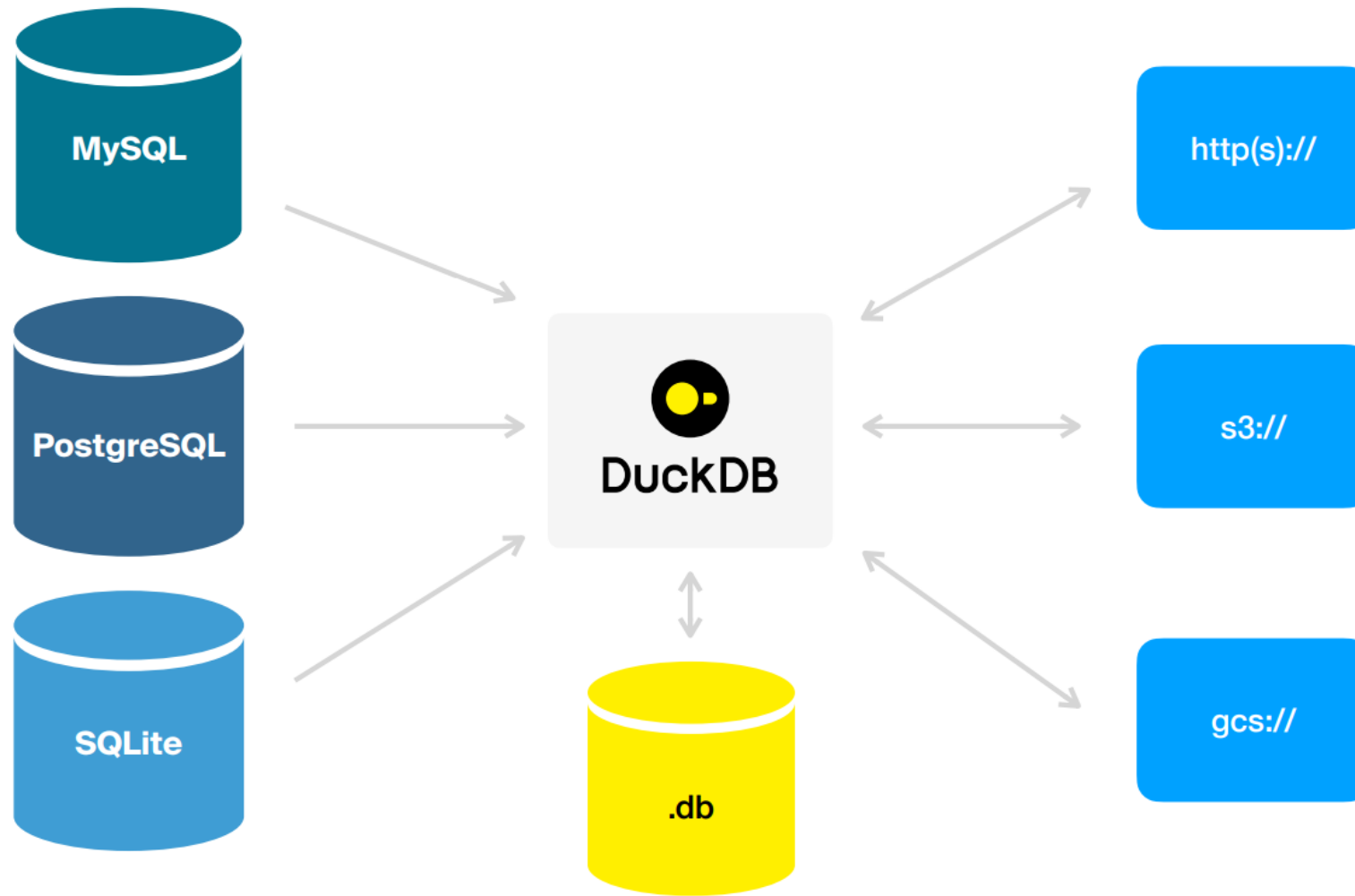
DuckDB ouvre grand les portes de l'open data

DuckDB lit et écrit de nombreux formats de fichiers



» [DuckDB: Harnessing in-process analytics for data science and beyond, 2023](#)

Il se connecte aussi à des bases et services web distants



» DuckDB: Harnessing in-process analytics for data science and beyond, 2023

DuckDB ouvre grand les portes de l'open data

Je gère déjà en R ou Python, alors DuckDB, ça m'apporterait quoi ?

De la vitesse

De la robustesse

De l'élégance

Vitesse : ordres de grandeur du temps de chargement d'un CSV

M2Pro CPU, 32GB RAM, DuckDB v0.9.1

1 Go de CSV = 1 seconde

20 Go de CSV = 20 secondes

360 Go de CSV = 5 mn

» [DuckDB: Harnessing in-process analytics for data science and beyond, 2023](#)

Robustesse des écritures

```
df_stations_service = df_bpe.lazy().filter( # 1.
    pl.col("TYPEQU") == "B316"
).groupby( # 2.
    "DEP"
).agg( # 3.
    pl.count().alias("NB_STATION_SERVICE")
).collect() # 4.

df_stations_service.head(5)
```

```
/tmp/ipykernel_238/1940419069.py:3: DeprecationWarning: `groupby` is deprecated. It has been renamed to `group_by`.
).groupby( # 2.
```

En lecture directe depuis un CSV

Pour faciliter l'usage des données de la Base Permanente des Equipements (BPE), une copie de la version 2019 est disponible à l'adresse : https://minio.lab.sspcloud.fr/donnees-insee/diffusion/BPE/2019/BPE_ENS.csv

```
url = "https://minio.lab.sspcloud.fr/donnees-insee/diffusion/BPE/2019/BPE_ENS.csv"
df_bpe = pl.read_csv(
    url, sep = ";",
    dtypes={
        "DEP": pl.Categorical,
        "DEPCOM": pl.Categorical
    })
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[6], line 2
      1 url = "https://minio.lab.sspcloud.fr/donnees-insee/diffusion/BPE/2019/BPE_ENS.csv"
----> 2 df_bpe = pl.read_csv(
      3     url, sep = ";",
      4     dtypes={
      5         "DEP": pl.Categorical,
      6         "DEPCOM": pl.Categorical
      7     })
```

La galère de la maintenance avec les API de dataframes :

Par exemple dans polars, *groupby* est devenu *group_by*, *sep* est devenu *separator*...

Mais dans d'autres librairies, ce sera *delim* ou *delimiter*, tout comme une agrégation s'écrira *agg*, *summarize* ou *rollup*, un tri *sort*, *orderby*, *arrange*...

» [Prise en main de Polars en Python - Insee](#)

Efficacité et concision

```
url = "https://minio.lab.sspcloud.fr/donnees-insee/diffusion/BPE/2019/BPE_ENS.csv"
df_bpe = pl.read_csv(
    url, separator = ";",
    dtypes={
        "DEP": pl.Categorical,
        "DEPCOM": pl.Categorical
    })
df_bpe
```

✓ 7.5s

shape: (1_060_614, 7)

REG	DEP	DEPCOM	DCIRIS	AN	TYPEQU	NB_EQUIP
i64	cat	cat	str	i64	str	i64
84	"01"	"01001"	"01001"	2019	"A401"	2
84	"01"	"01001"	"01001"	2019	"A404"	4
84	"01"	"01001"	"01001"	2019	"A405"	2

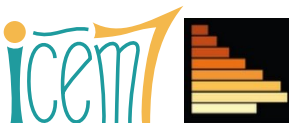
» [Prise en main de Polars en Python - Insee](#)

Lecture d'un fichier CSV dans **polars** : configuration obligatoire des colonnes de codes géographiques et spécification du délimiteur.

FROM 'https://minio.lab.sspcloud.fr/donnees-insee/diffusion/BPE/2019/BPE_ENS.csv' ;

123 REG	ABC DEP	ABC DEPCOM	ABC DCIRIS	123 AN	ABC TYPEQU	123 NB_EQUIP
84	01	01001	01001	2019	A401	2
84	01	01001	01001	2019	A404	4
84	01	01001	01001	2019	A405	2
84	01	01001	01001	2019	A504	1
84	01	01001	01001	2019	A507	1
84	01	01001	01001	2019	B203	1
84	01	01001	01001	2019	C104	1

Lecture d'un fichier CSV avec **DuckDB** : rien à faire !



Élégance

Calcul d'une somme de nombre d'équipements

```
df_bpe.lazy()\n    .with_columns(pl.col('NB_EQUIP').cast(pl.Int64, strict=False))\n    .select(\n        pl.col("NB_EQUIP").sum().alias("NB_EQUIP_TOT")\n    ).head(5).collect()
```

Polars

NB_EQUIP_TOT
i64
2607293

```
FROM df_bpe\nSELECT sum(NB_EQUIP) NB_EQUIP_TOT ;
```

DuckDB

123 NB_EQUIP_TOT
2 607 293

» [Prise en main de Polars en Python - Insee](#)

Les + du SQL dans DuckDB

GROUP BY ALL

GROUP BY GROUPING SETS ((), ())

FROM <table> SELECT ...

FROM 'https://monsite.fr/rp.parquet'

SELECT * EXCLUDE ()

SELECT * REPLACE ()

SELECT COLUMNS(c -> c ILIKE '%pop%')

SELECT REG, sum(COLUMNS(c -> c ILIKE '%pop%')) GROUP BY REG

Colonnes calculées immédiatement réutilisables



DuckDB, d'où ça sort ?

Les papas de DuckDB : 2 chercheurs en bases de données



Hannes Mühleisen et Mark Raasveldt

Ils ont collaboré au sein du CWI (Centrum Wiskunde & Informatica) à Amsterdam.

Ils ont écrit ensemble plusieurs articles de recherche.

Ils ont aussi créé [MonetDBLite](#), avant de se lancer en 2018 dans l'aventure [DuckDB](#).

Pourquoi ce nom « DuckDB » ?

CWI

DuckDB

- ▶ Why “Duck” DB?
- ▶ Ducks are amazing animals
- ▶ They can fly, walk and swim
- ▶ They are resilient
- ▶ They can live off anything
- ▶ Also Hannes used to own a pet duck



Qu'est-ce que le CWI (Centrum Wiskunde & Informatica) ?

CWI

L'INRIA néerlandais : l'**institut national de recherche en informatique**

Depuis 40 ans, le CWI est un centre d'excellence et d'innovation de niveau mondial, basé à Amsterdam :

- **Python** : créé par Guido van Rossum (1991)
- **MonetDB** : 1^{re} base de données « orientée colonnes », Martin Kersten (1993)
- **X100** : exécution de requêtes vectorisées, Peter Boncz, Niels Nes et Marcin Zukowski (2005) -> Vectorwise puis **Snowflake**
- **DuckDB** : 2018, prend la suite de **MonetDBLite**, se définit comme un moteur de BDD autonome, embarquable partout (comme SQLite), simple et véloce





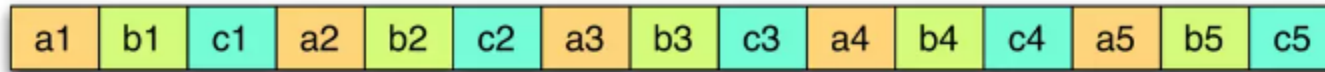
Savoir utiliser les nouveaux formats orientés colonnes et streamables

Formats orientés colonnes : l'exemple de parquet

Une table
logique

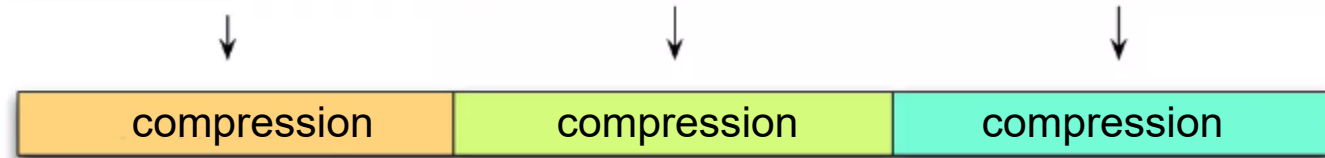
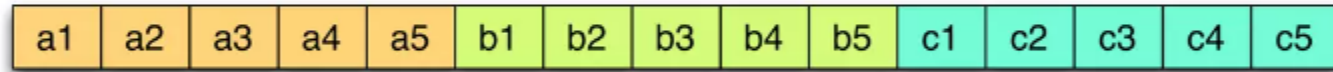
a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Stockage physique **en lignes**



→ Insertion de nouveaux enregistrements aisée.

Stockage physique **en colonnes**



→ Encodage et compression ++
Il devient plus facile de lire les seules colonnes utiles dans une requête.

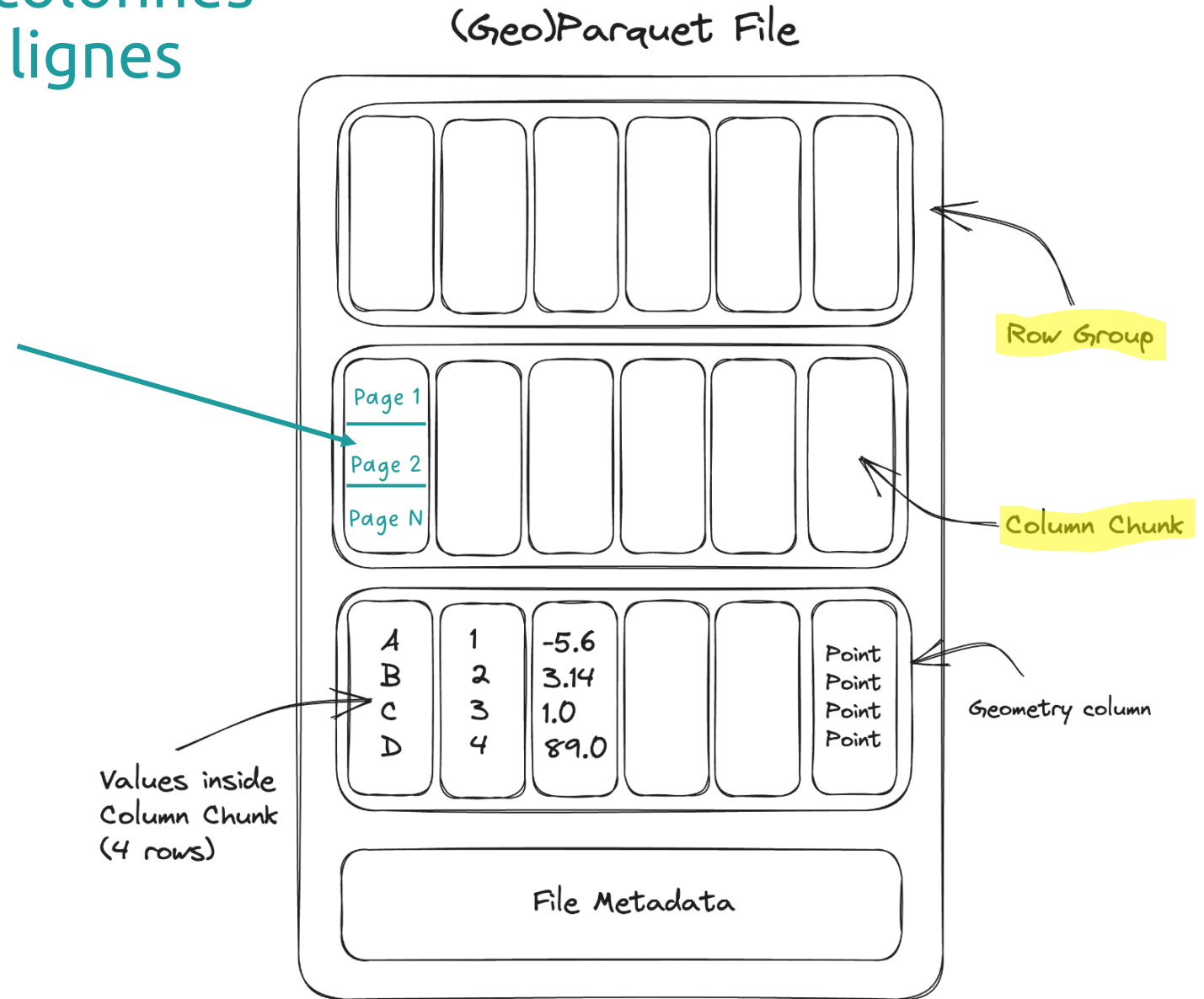
» How to use Parquet

Un fichier parquet sépare les colonnes et se structure en groupes de lignes

D'où la dénomination « **parquet** ».

Un « column chunk » est lui-même découpé en « pages », et chacun de ces blocs rejoint le concept de « vecteur ».

Des **métadonnées** sont stockées dans chaque **row-group**, et même pour chaque « page ».

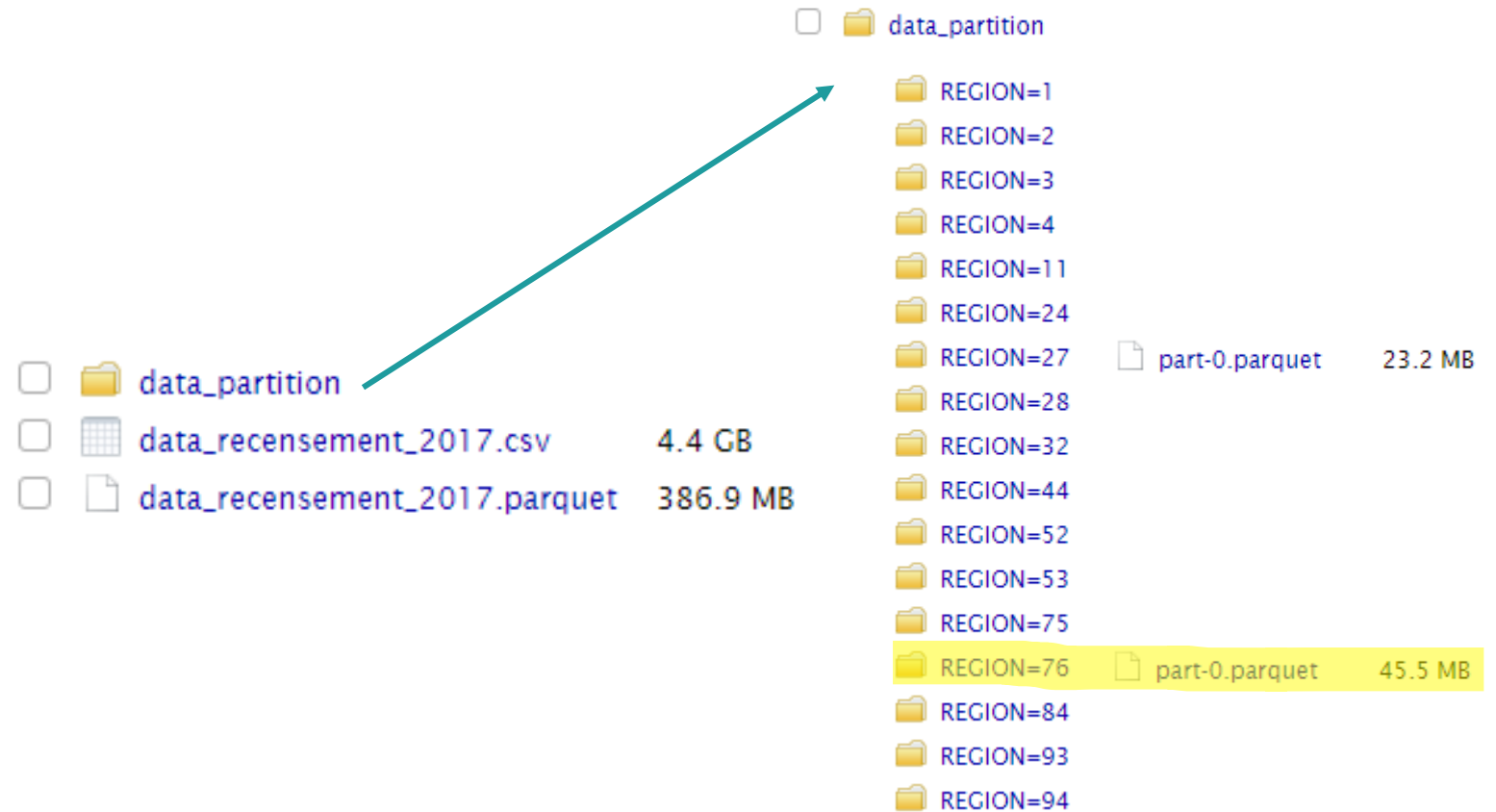


» Guidelines for GeoParquet

Comment éliminer de la recherche les lignes inutiles - 1

D'abord en découpant un gros fichier selon les modalités d'un champ clé.

Ce que l'on appelle une partition.



DuckDB peut parcourir toute une partition, ne retenant que ce qu'il faut.

```
SELECT COMMUNE, COUNT(*) FILTER (WHERE TYPL = '5') AS nb_logements_fortune
FROM read_parquet('data_partition/**/*.parquet', hive_partitioning = 1)
WHERE REGION = '76'
GROUP BY ALL ;
```

Comment éliminer de la recherche les lignes inutiles - 2

On s'appuie sur les « **statistiques** » de chaque row-group.

En particulier, on y trouve **la valeur min et la valeur max** de chaque colonne dans un groupe.

Ces métadonnées se lisent très vite et permettent d'écarter du scan les row-groups non pertinents.

```
SELECT * FROM table WHERE x = 5
```

```
Row-group 0: x: [min: 0, max: 9]
```

```
Row-group 1: x: [min: 3, max: 7]
```

```
Row-group 2: x: [min: 1, max: 4]
```

De l'intérêt de ces filtres précoces (filter push-down)...

Filtrage de colonnes
(*projection push down*)

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

+

Filtrage de lignes
(*predicate push down*)

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

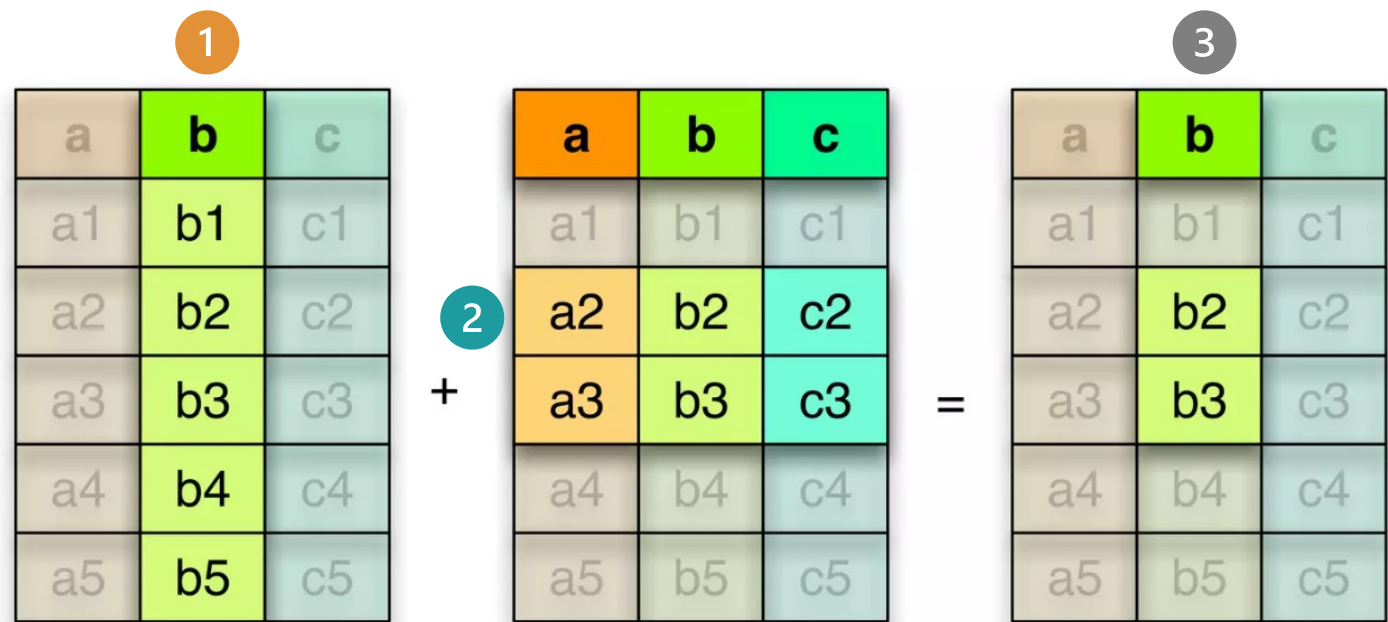
=

Pour ne considérer que
les données a priori pertinentes

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

» How to use Parquet

... qui accélèrent nos requêtes



```
FROM 'https://static.data.gouv.fr/resources/sirene-geolocalise-parquet/20240107-143656/sirene2024-geo.parquet'  
SELECT geometry  
WHERE codeCommuneEtablissement = '31555'  
AND siren = '808183610'
```

3 5 Mo chargés seulement d'un fichier de 1 Go

sirene2024-geo.parquet	206	xhr	duckdb-browser...	17.1 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	66.3 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	263 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	1.0 MB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	17.1 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	17.1 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	66.3 kB
sirene2024-geo.parquet	206	xhr	duckdb-browser...	263 kB

Multiple coups de sonde

Chaque appel lit une portion de ce fichier parquet, un **byte-range**

sirene2024-geo.parquet	Content-Length:	262144
sirene2024-geo.parquet	Content-Range:	bytes 175276032-175538175/972726667
sirene2024-geo.parquet	Content-Type:	application/octet-stream

Avec parquet, le tri du fichier et l'ordre des filtres important

Interrogation d'un fichier Sirène Geoparquet (1 Go)
trié par code commune, puis code Siret

```
FROM 'https://.../sirene2024-geo.parquet'  
SELECT geometry  
WHERE codeCommuneEtablissement = '31555'  
AND siren = '808183610'
```

HTTP Stats:

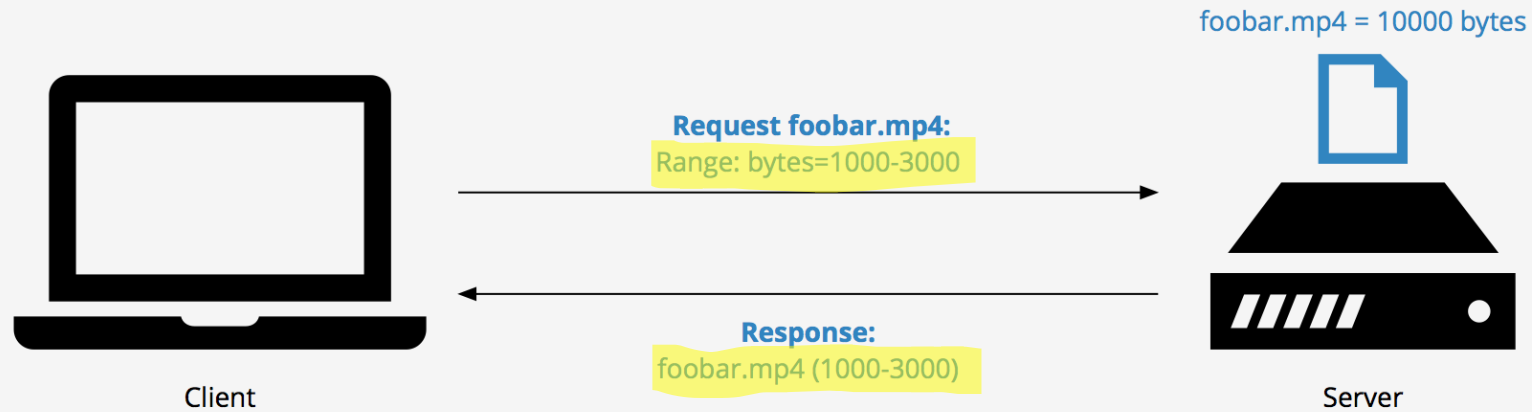
in: 4.9MB
out: 0 bytes
#HEAD: 0
#GET: 32
#PUT: 0
#POST: 0

```
FROM 'https://.../sirene2024-geo.parquet'  
SELECT geometry  
WHERE siren = '808183610'  
AND codeCommuneEtablissement = '31555'
```

HTTP Stats:

in: 20.4MB
out: 0 bytes
#HEAD: 0
#GET: 28
#PUT: 0
#POST: 0

Requêtes de plages : c'est comme cela que se lit une vidéo



Byte-Range Requests

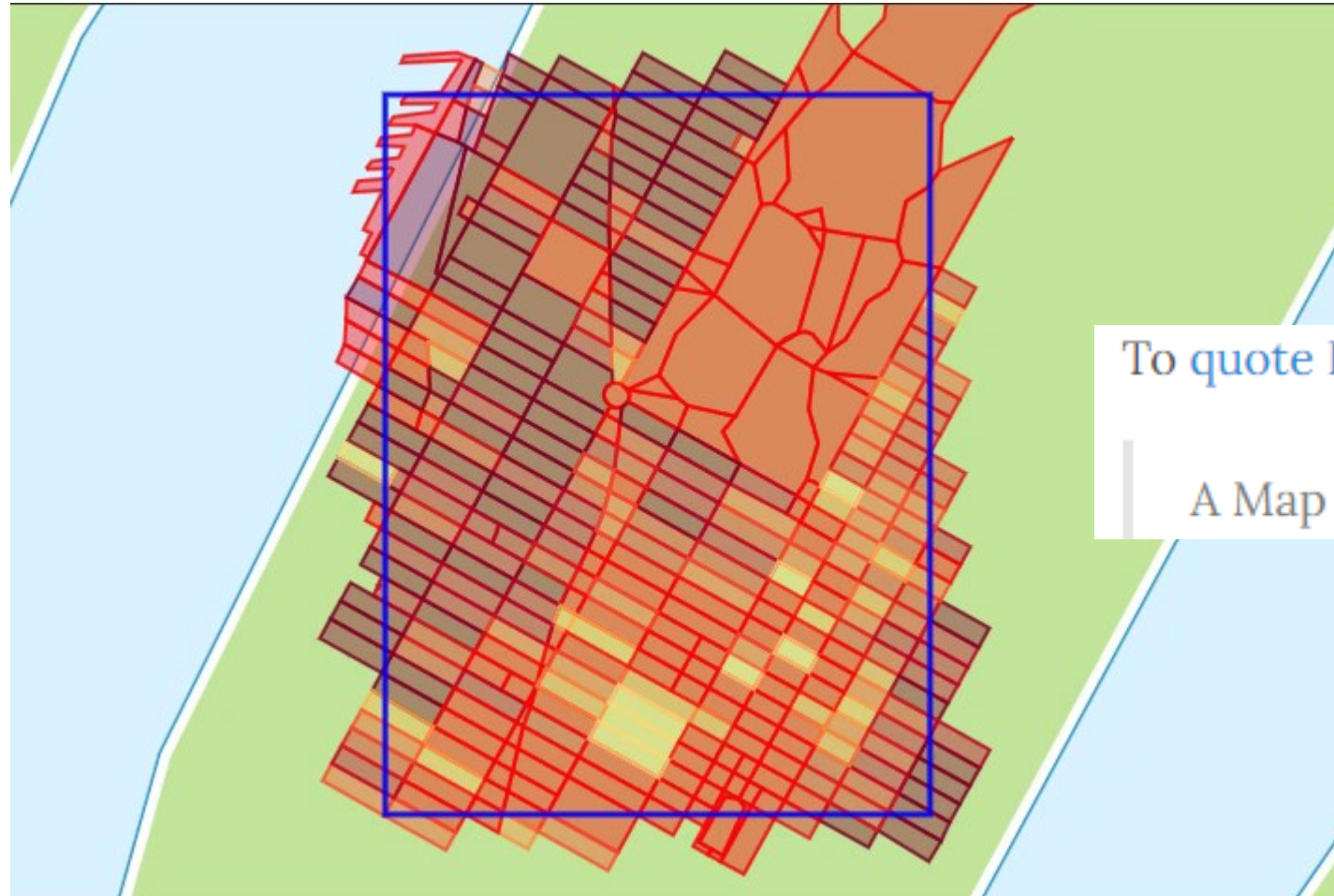
Le monde SIG est déjà en avance avec les formats streamés

- COG (cloud optimized Geotiff)
- FlatGeoBuf
- PMTiles
- Geoparquet

Tous ces formats se « streament » via des range-requests.

Et Geoparquet s'interroge en SQL avec DuckDB.

Here's a [demo](https://flatgeobuf.org) on the flatgeobuf.org site that shows this in action: live slippy map filtering of a 12GB US Census Block dataset direct from storage over HTTP



To quote Brandon Liu:

A Map is just a Video

» flatgeobuf.org



Perspectives et limites

DuckDB a des marges de progression, et aussi des limites

SQL a ses limites (comparé à R/dplyr, notamment les renommages dynamiques de colonnes)

DuckDB est optimisé pour requêter des BDD en lecture seule (OLAP – on line analytical processing)

Il ne fait que du calcul simple, pas de modélisation ni de dataviz

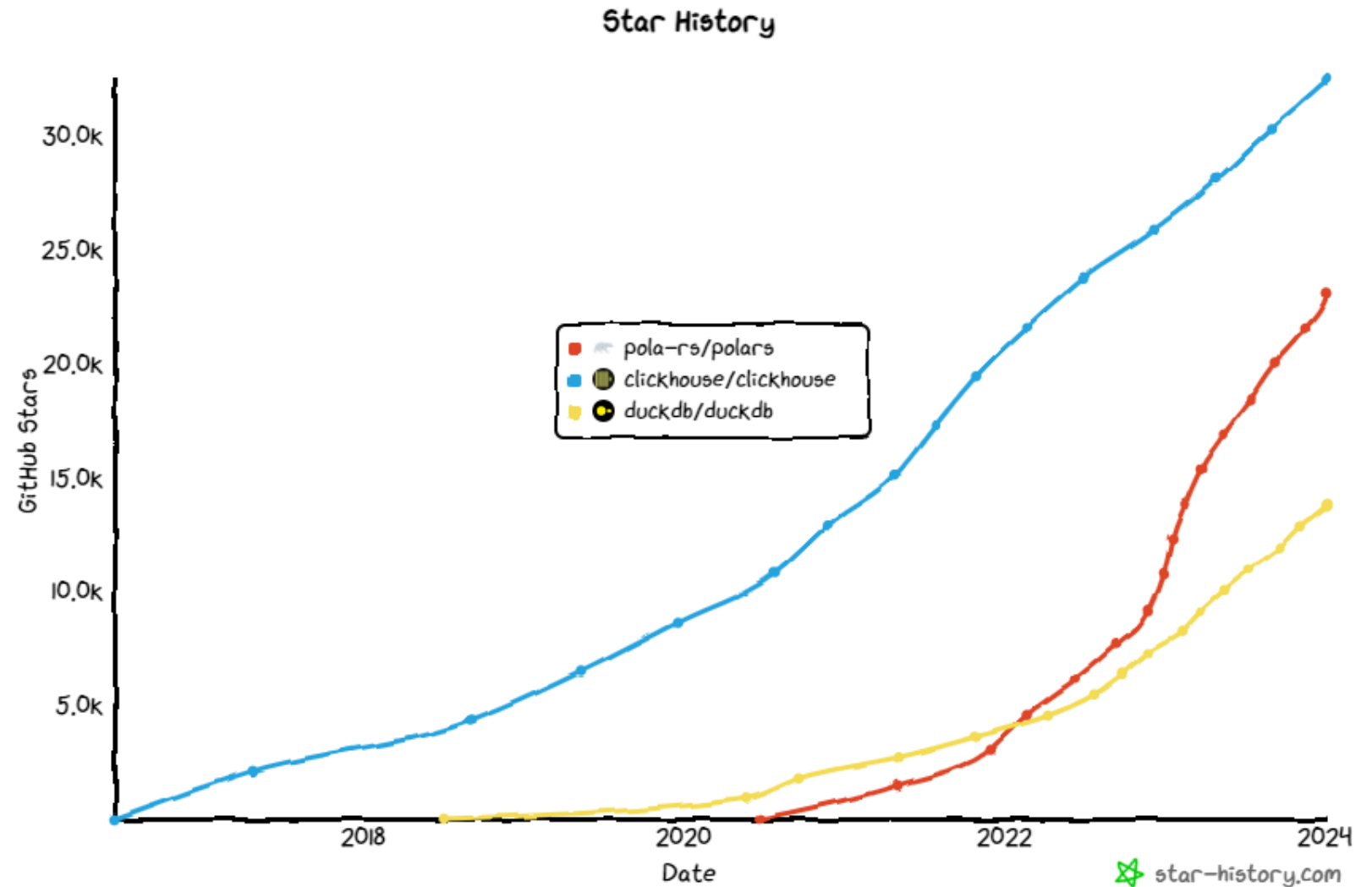
Le support spatial est à améliorer encore (indexation spatiale), en liaison avec les évolutions du format geoparquet

La concurrence : Polars et Clickhouse

Polars est aussi basé
à Amsterdam !



Richie Vink



» [star-history](https://star-history.com)

DuckDB ouvre grand les portes de l'open data






DuckDB est dans le peloton de tête pour la rapidité...

Les BDD orientées colonnes font la différence.

0.5 GB 5 GB **50 GB**

basic questions

Input table: 1,000,000,000 rows x 9 columns (50 GB)

	duckdb-latest	0.9.1.1	2023-10-26	24s
	DuckDB	0.8.1.3	2023-10-26	25s
	ClickHouse	23.10.4.25	2023-11-30	29s
	Polars	0.19.8	2023-10-17	32s
	DataFrames.jl	1.6.1	2023-10-17	84s

System & Machine

Relative time (lower is better)

ClickHouse 23.11 (c6a.metal, 500gb gp2):	×1.43
Databend (c6a.metal, 500gb gp2):	×1.55
DuckDB (c6a.metal, 500gb gp2):	×1.69
SelectDB (c6a.metal, 500gb gp2):	×1.94
Snowflake (8×L):	×5.92
MonetDB (c6a.4xlarge, 500gb gp2):	×11.58
MariaDB ColumnStore (c6a.4xlarge, 500gb gp2) [†] :	×60.98
MongoDB (c6a.4xlarge, 500gb gp2):	×412.49
TimescaleDB (c6a.4xlarge, 500gb gp2):	×877.04
SQLite (c6a.4xlarge, 500gb gp2):	×887.82
PostgreSQL (c6a.4xlarge, 500gb gp2):	×909.36
MySQL (MyISAM) (c6a.4xlarge, 500gb gp2):	×1360.92

First time
Second time

2.5 3.0

» The Return of the
H2O.ai Database-like Ops
Benchmark

» ClickBench - a Benchmark For Analytical DBMS

DuckDB ouvre grand les portes de l'open data

... mais son charme résulte d'un ensemble de qualités

Se lance en un clin d'œil, dans un tas d'environnements (yc web),
aucun pb d'installation, pas de droits à demander

Conçu par un tandem chevronné top niveau en recherche BDD, s'appuie sur des
dizaines d'années d'expertise au CWI, résolument opensource et libre (MIT).




Utilise un langage éprouvé, pérenne (SQL), rendu plus amical encore pour l'utilisateur

Intègre l'analyse spatiale

Lit tous les formats ou presque : CSV, parquet, JSON, formats géo, le tout en https

Exploite à fond la puissance de votre ordi personnel. Autonomie +++

Perspectives 2024

DuckDB versions			
v0.9	Version courante		v1.0
v0.10	Tôt en 2024		Format de base stabilisé
v1.0	Courant 2024		Améliorations en stabilité et robustesse
			Optimisations de performance



Quel est ma
conclusion ?

« Avec **DuckDB**, je me gave d'open data
tellement **c'est devenu facile**
de **butiner** les sources de données. »

Merci pour votre attention !

Site et blog : <https://www.icem7.fr/>

Twitter : [@ericmauviere](https://twitter.com/ericmauviere)

LinkedIn : www.linkedin.com/in/ericmauviere

Merci

