UMWELT BUNDES AMT

DuckBB

DUCK TALES

Open Source mit Open Data: GIS Datenverarbeitung in DuckDB, PostGIS und Sedona

Gregor Rosenauer, Umweltbundesamt

2.12.2025

INTRO

DI Gregor B. Rosenauer

Senior Software Engineer Geosemantik & AI

Fokus: Geo-Datenanalyse & Softwareentwicklung im Team Fernerkundung (UBA)

Hintergrund: Informatik, DI (TU Wien)

Spezialisierung: Semantik und Service Integration, Open Source und Open Data (FAIR)

Erfahrung: Architektur komplexer, skalierbarer Plattformen (z.B. Ontologie-Management, Cloud-Lösungen, Integrationsplattformen)

verschiedene Formate, Brüche, schwer zu lesen und zu nutzen, unterschiedl. Systeme und Ablagen.

Netzlaufwerk, lokale Daten und Tools, lokale embedded DBs,...

- Versionskonflikte
- Reproduzierbarkeit?
- Austausch schwierig
- Sicherungen?
- Zugriffsrechte, tracing,...

geordnete, miteinander verbundene Datenströme, die durch klare Strukturen und Standards fließen

Effizienz und universelle Zugänglichkeit durch die Nutzung von offenen Standards und Open Data:

- Heterogene Daten in gemeinsames Standardformat umgewandelt
- Innerhalb des Systems einheitliche Verarbeitung
- Daten zentralisiert, Tools greifen geregelt darauf zu
- Versionierung, Zugriffskontrolle, etc.

GIS für alle Fälle

PostGIS (PostgreSQL Extension)

Ausgereiftes freies RDBMS. Seit 20+ Jahren etabliert.

Persistente Speicherung mit umfangreicher Indizierung.

DuckDB Spatial (In-Process Analytics)

Modern und leichtgewichtig (seit 2023).

Ideal für explorative Analyse ohne Infrastruktur-Aufwand.

Apache Sedona (Spark-Native und lokal)

Enterprise-Scale, verteilt im Cluster oder lokal.

Sowohl für kleine bis mittlere Datenmengen (SedonaDB) als auch für Petabyte-Datenmengen auf Cluster-Infrastruktur optimiert.

Vergleich 1/2: Grundlagen & Setup

Feature	PostGIS	DuckDB Spatial	Apache Sedona
Typ	Relationale DB + Extension, transaktional	In-Process OLAP + Extension	Native geospatial DB (SQL + Vector)
Setup	Docker/Server Installation	Python (pip install)	Spark Cluster / lokal (SedonaDB)
Use Case	OLTP + GIS	Ad-hoc Analyse, lokale Entwicklung	Big Data ETL / kleine bis mittelgroße Daten
Kopplung Daten – Verarbeitung	Enge Kopplung (Speicherung und Berechnung in DB	Eingebettet (Berechnung am Client, Daten nur importiert nicht referenziert)	lose, flexibel (lokale Dateien oder Data Lake)
Installation & Skalierung	Zentrale Instanz, volle Replika bzw. Sharding	Eingebettet am Client, native Python/R/CLI Integration	Binär- oder Python-Paket, keine Abhängigk.

Vergleich 2/2: Features & Performance

Feature	PostGIS	DuckDB Spatial	Apache Sedona
Spatial Native	Ausgereifte Geometrie und Geografie Unterst., Raster und Index GiST	DuckDB-spatial Erweiterung, relativ jung und tlw. noch unvollständig bzw. nicht so ausgereift (CRS, KNN)	Spatial Data und Abfragen, Checks und Index nativ unterstützt
Setup	Docker/Server Installation	Python (pip install)	Spark Cluster
Fokus	Universale Datenbank für OLTP und GIS	OLAP für lokale bzw Parquet Daten, schnelle Scans+Aggregate	Vektorbasiert, kleine bis mittlere Datenmengen, optimierte Abfragen
Rasterdaten Unterstützung	PostGIS Raster	Derzeit nur experimentell durch die Community	geplant

PostGIS: The Elephant in the room

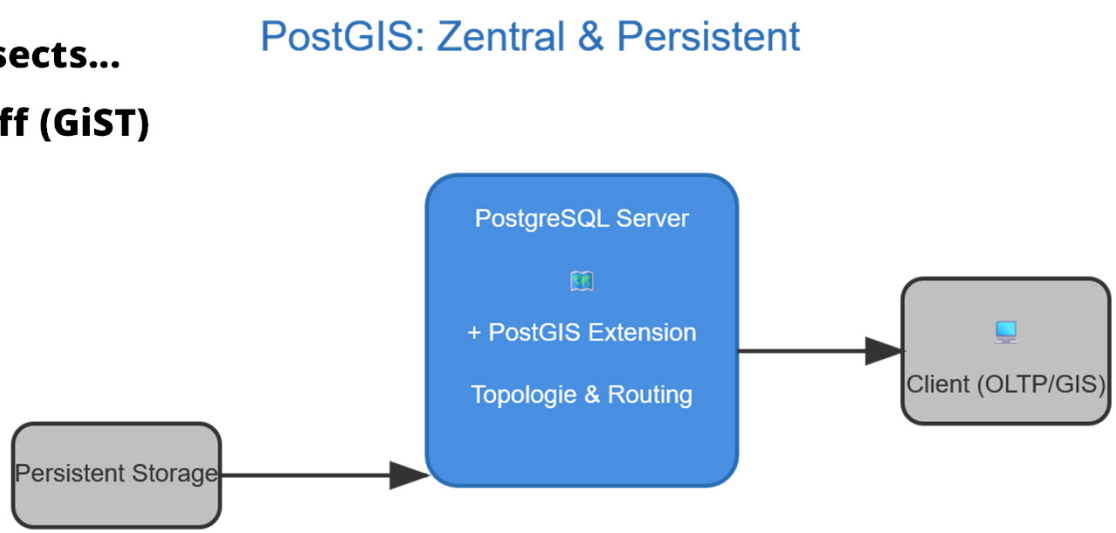
Geometry: Kartesische Koordinaten (projiziert)

Geography: Ellipsoidale Koordinaten (WGS84)

50+ Funktionen: ST_Distance, ST_Within, ST_Intersects…

Vorteil: Spatial Joins mit optimiertem Index-Zugriff (GiST)

```
SELECT ST_Length(geom), ST_AsGeoJSON(geom)
FROM streets
WHERE ST_DWithin(
    geom,
    ST_Point(16.37, 48.20)::geography,
    500
);
-- Ergebnis: <200ms Index-Zugriff
```



DuckDB Spatial: lokale Daten erkunden

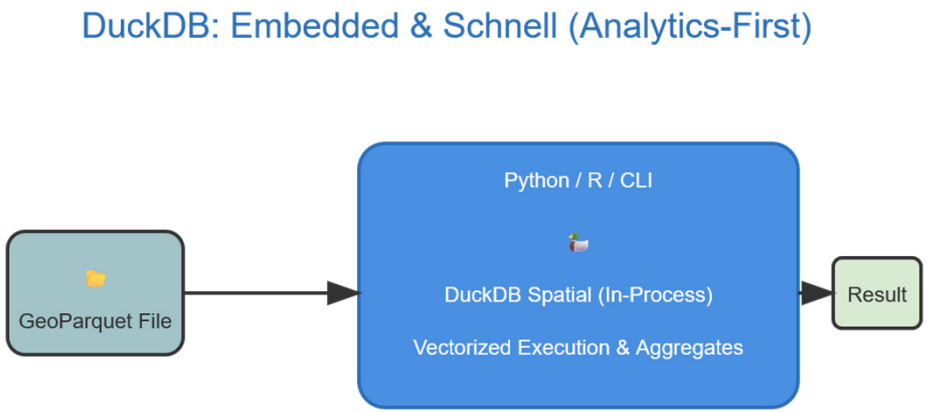
Philosophie: Analytics-First Workflow ohne DB-Setup

Integration: Nahtlos mit JSON, Zeitreihen, Parquet

Vectorized Execution: Für schnelle Prototypen

Ideal für: Exploratory Analysis ohne Infrastruktur

```
INSTALL spatial; LOAD spatial;
SELECT ST_Length(geom)
FROM ST_Read('gip_streets.gpkg')
WHERE ST_DWithin(
    geom,
    ST_Point(16.37, 48.20),
    0.005
);
-- Query-Zeit: 100-500ms (Zero Setup)
```



Apache Sedona: Expertenleichtgewicht

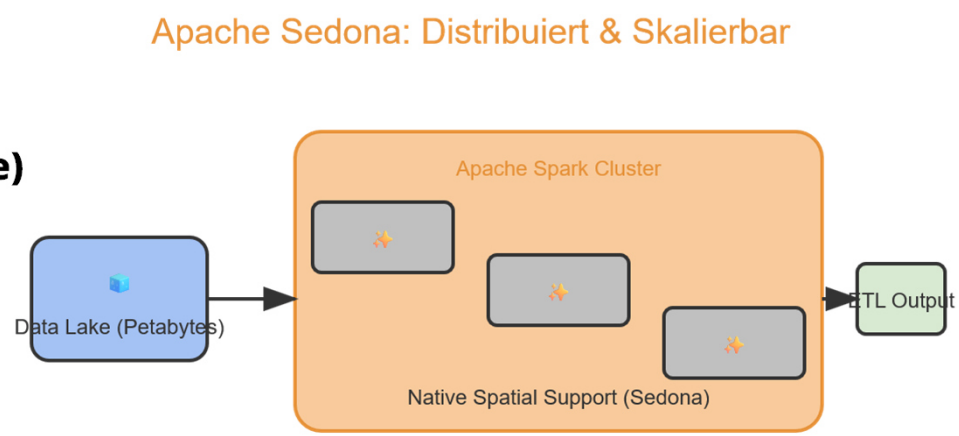
Philosophie: Nativer räumlicher Support in Apache Spark

Skalierung: Designed für Milliarden-Reihen Datensätze

GeoParquet: Integration mit Spatial Predicate Push-Down (Räumliche Prädikats-Ablagerung)

Ideal für: Kontinentale Straßennetzwerke (100M+ Objekte)

```
SELECT ST_Length(geom), ST_GeoHash(geom, 5)
FROM large_road_network
WHERE ST_Within(
    geom,
    ST_Buffer(ST_Point(x, y), 500)
);
-- Typisch: 5-30s auf 8-Knoten-Cluster mit großen Datenmengen (Petabyte scale)
```



Fazit PostGIS/DuckDB/Sedona

Keine "beste" Lösung: Abhängig von Skalierung, Persistenz, Workflow.

Komplementär: Viele Organisationen nutzen alle drei parallel.

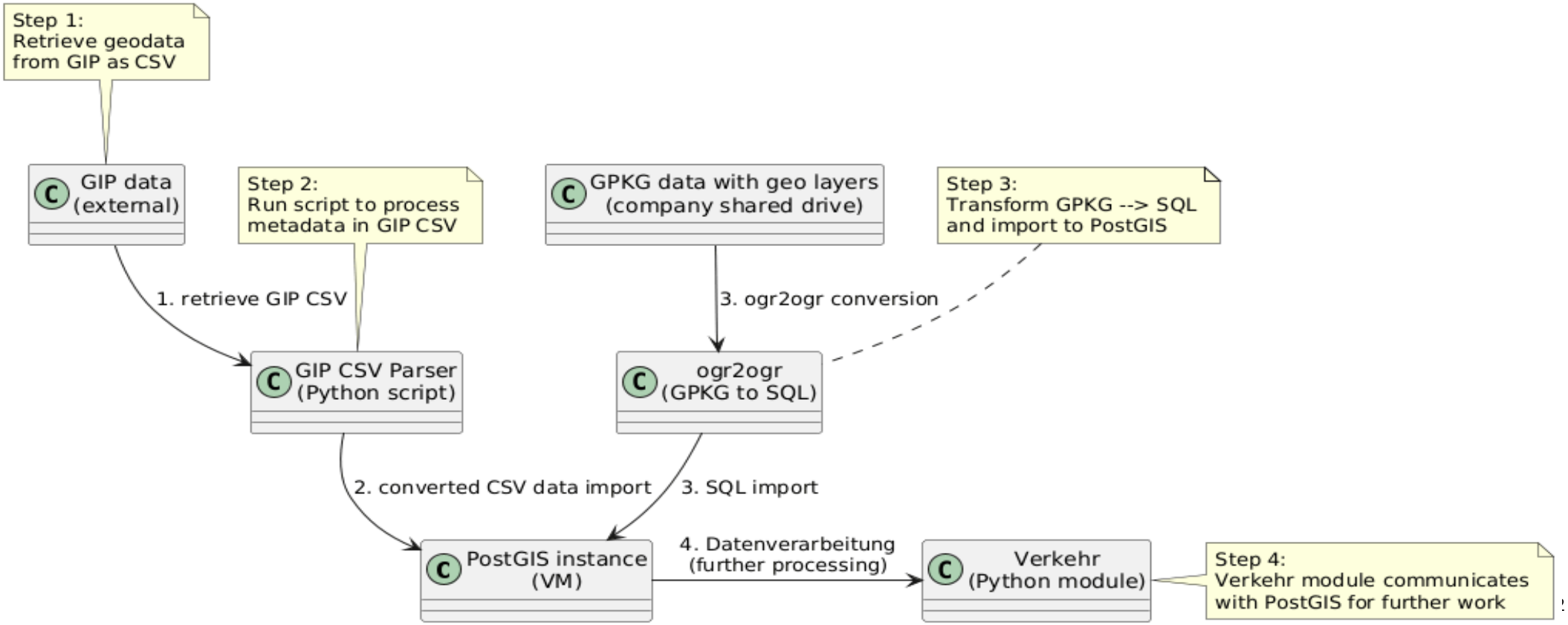
GIP Fazit: **PostGIS** für traditionelle GIS-Stabilität,
DuckDB für schnelle Analyse.
SedonaDB zur Evaluierung und Skalierung

Trend: GeoParquet entwickelt sich zum Standard für Cloud Analytics.

Ressourcen: postgis.net • duckdb.org/spatial • sedona.apache.org

Datenfluss: GiP CSV > Python > PostGIS

Implementierung am Beispiel GIP (Graphenintegrations-Plattform)



GIP-Workflow: Datenfluss+Architektur

1. **Import von GIP Datenquelle: data.gv.at (CSV)**
2. **Transformation GIP CSV Dump mit Python**
   1. **Datenmodell aus CSV parsen (extra Zeile mit Fields und Types)**
   2. **Schema in PostgreSQL erstellen**
   3. **Daten in DB importieren**
3. **Transformation GPKG Layer: ogr2ogr (Python)**
   1. Open source CLI Tool ogr2ogr wandelt GPKG in standard SQL um
   2. Import in PostgreSQL
4. **Gemeinsame PostGIS Instanz als zentraler Hub zur weiteren Geodaten-Verarbeitung**
   1. Alle Clients können nun sicher und transaktional auf eine gemeinsame, konsistente und aktuelle Datenbasis zugreifen und die Verarbeitung in fachspezifischen Modulen (Verkehr, Windkraft,...) abgewickelt werden.
5. **Analyse: Verkehr-Modul (Python: Datenverarbeitung & Analyse)**

Wunschszenario

1. **zentrale Verwaltung**
   auch von großen Daten an *einem* Ort
2. **Versionierung und Tagging**
   sauberer Stand (zB "ÖROK 2025")
3. **Nachvollziehbarkeit**
   gesamte Berechnung mit allen Daten
4. **keine Konflikte**
   jeder im Team hat den selben Stand
5. **Einheitlicher Zugriff**
   Entwickler (Tools), Experten (TortoiseGit)

Entscheidungs-Framework: Wann nutzen wir was?

Nutze PostGIS wenn...

Persistente DB nötig ist

Komplexe Multi-Step Transformationen & Topologie (Routing)

Nutze DuckDB wenn...

Exploratory Analysis & Prototyping

Kein Infrastructure-Setup gewünscht (GeoParquet Workflows)

Nutze Sedona wenn...

Distribuierte Verarbeitung erforderlich (Petabyte-Scale)

Data Lake Architektur (Delta, Iceberg)

UMWELT BUNDES AMT

umweltbundesamt.at
instagram.com/umweltbundesamt_at/
bsky.app/profile/umweltbundesamt-at.bsky.social
linkedin.com/company/umweltbundesamt

DuckDB, PostGIS und Sedona
Salzburg, 2.12.2025

KONTAKT

DI Gregor B. Rosenauer

Team: Fernerkundung und Raumanalyse
Softwareentwicklung Geosemantik
Kontakt: gregor.rosenauer@umweltbundesamt.at
Telefon: +43 664 620 68 56

umweltbundesamt.at
instagram.com/umweltbundesamt_at/
bsky.app/profile/umweltbundesamt-at.bsky.social
linkedin.com/company/umweltbundesamt

Salzburg, 1.12.2025

GIP-Implementation: Details

Docker Setup:

PostgreSQL 18 mit PostGIS Extension

Python 3 mit psycopg2 Bindings

GDAL/ogr2ogr für Format-Transformation

Daten-Transformationen:

EPSG:4326 (WGS84) → EPSG:3857 (Web Mercator)

Räumliche Indizierung (GiST/BRIN)

Netzwerk-Topologie für Routing-Analyse