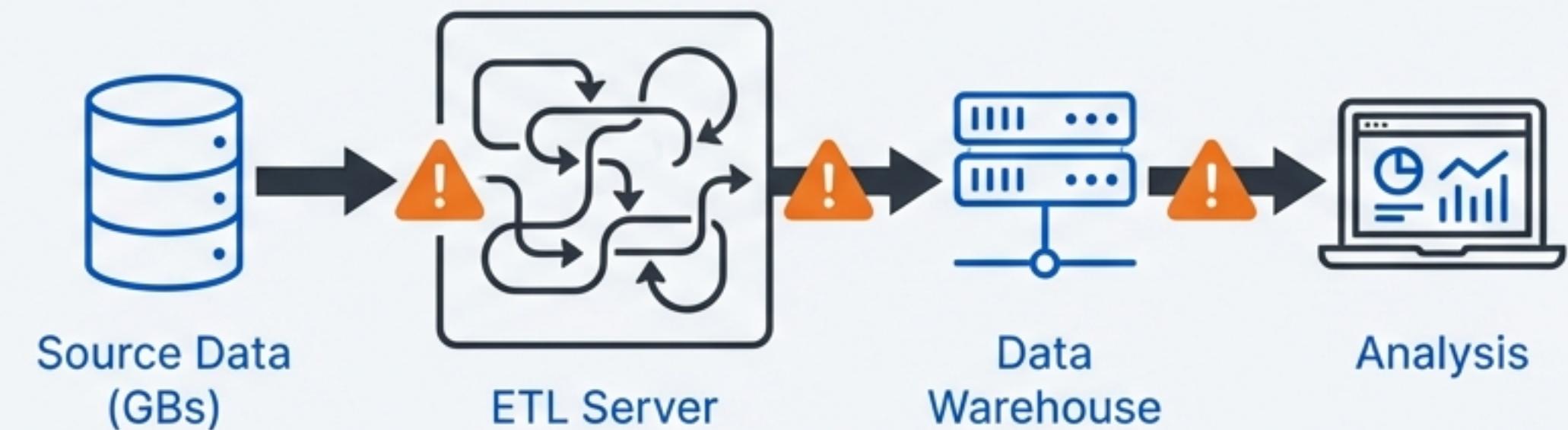


# 대규모 지리 공간 데이터, 아직도 힘겹게 다루고 계신가요?

수백 기가바이트(GB)에 달하는 파일,  
느린 쿼리 속도, 복잡한 데이터 로딩 및  
변환 파이프라인.

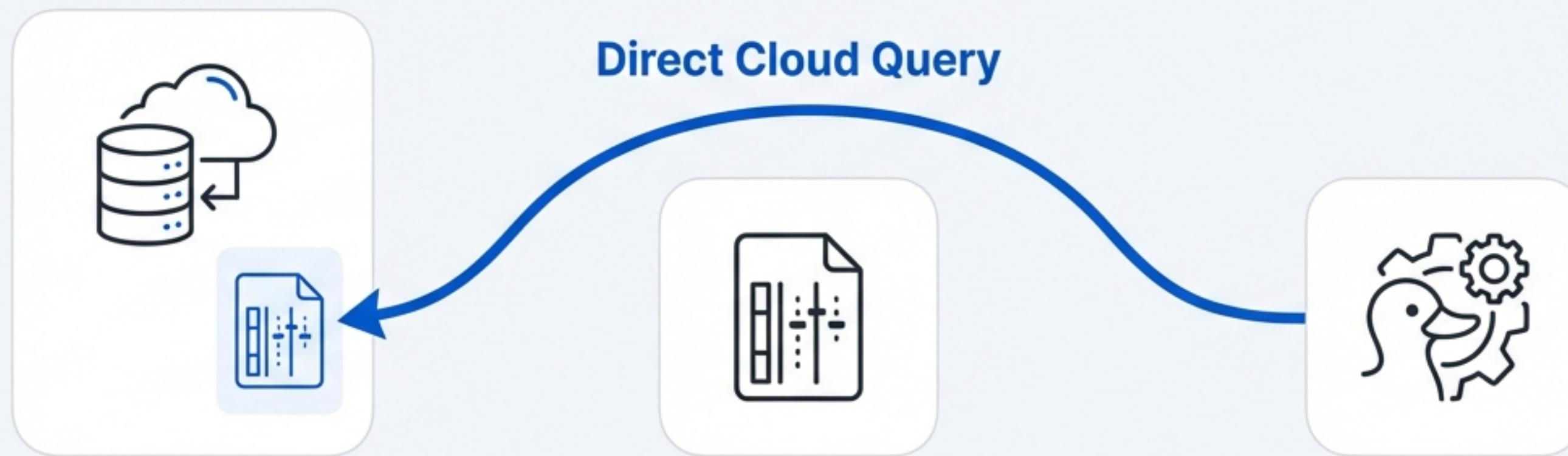
기존의 워크플로우는 데이터의 규모가  
커질수록 한계에 부딪힙니다.

분석을 시작하기도 전에 데이터 준비에  
대부분의 시간을 소모하고 있지는  
않으신가요?



# 클라우드 네이티브 시대의 지리 공간 분석: 더 빠르고, 더 간단하게

로컬에 데이터를 다운로드하거나 복잡한 데이터베이스를 설정할 필요 없이, 클라우드에 저장된 대규모 데이터를 직접, 그리고 즉시 쿼리하는 새로운 워크플로우를 소개합니다.



**The Source:**  
Source Cooperative  
(데이터 소스)

**The Format:**  
GeoParquet  
(데이터 포맷)

**The Engine:**  
DuckDB  
(쿼리 엔진)

# The Format: 열 지향(Columnar) 스토리지 GeoParquet로 데이터를 효율적으로

기존의 행(Row) 기반 포맷과 달리, Parquet는 데이터를 열(Column) 단위로 저장합니다.  
이는 분석 쿼리에 혁신적인 이점을 제공합니다.

## 행 지향 (Row-Oriented)

ID	City	Country
1	Oslo	Norway
2	Seoul	Korea
3	Tokyo	Japan

SELECT City



## 열 지향 (Column-Oriented)

ID	City	Country
1	Oslo	Norway
2	Seoul	Korea
3	Tokyo	Japan

SELECT City



**높은 압축률 (Higher Compression):**  
동일한 데이터 타입의 열을 함께 압축하여  
파일 크기를 획기적으로 줄입니다.



**빠른 읽기 속도 (Faster Read Speed):**  
쿼리에 필요한 열만 읽어들여 I/O를 최소화하고  
분석 속도를 극대화합니다.

# The Engine: 제로-셋업 고성능 분석 엔진 DuckDB

DuckDB는 서비스 인메모리(in-memory) 분석 데이터베이스(OLAP)입니다. 복잡한 설치나 외부 종속성 없이, 단일 파일로 실행되어 데이터 분석을 위한 민첩하고 강력한 SQL 환경을 제공합니다.

**OLTP vs. OLAP:** PostgreSQL과 같은 OLTP 데이터베이스가 잦은 삽입/수정/삭제(transacional)에 최적화된 반면, DuckDB와 같은 OLAP 데이터베이스는 대규모 데이터의 복잡한 분석 쿼리(analytical)에 특화되어 있습니다.

"분석을 위한 SQLite"로 생각할 수 있습니다.



**Simple:** 쉬운 설치, 외부 종속성 없음



**Fast:** 벡터화된 쿼리 실행



**Feature-Rich:** 원격 Parquet 파일 직접 쿼리



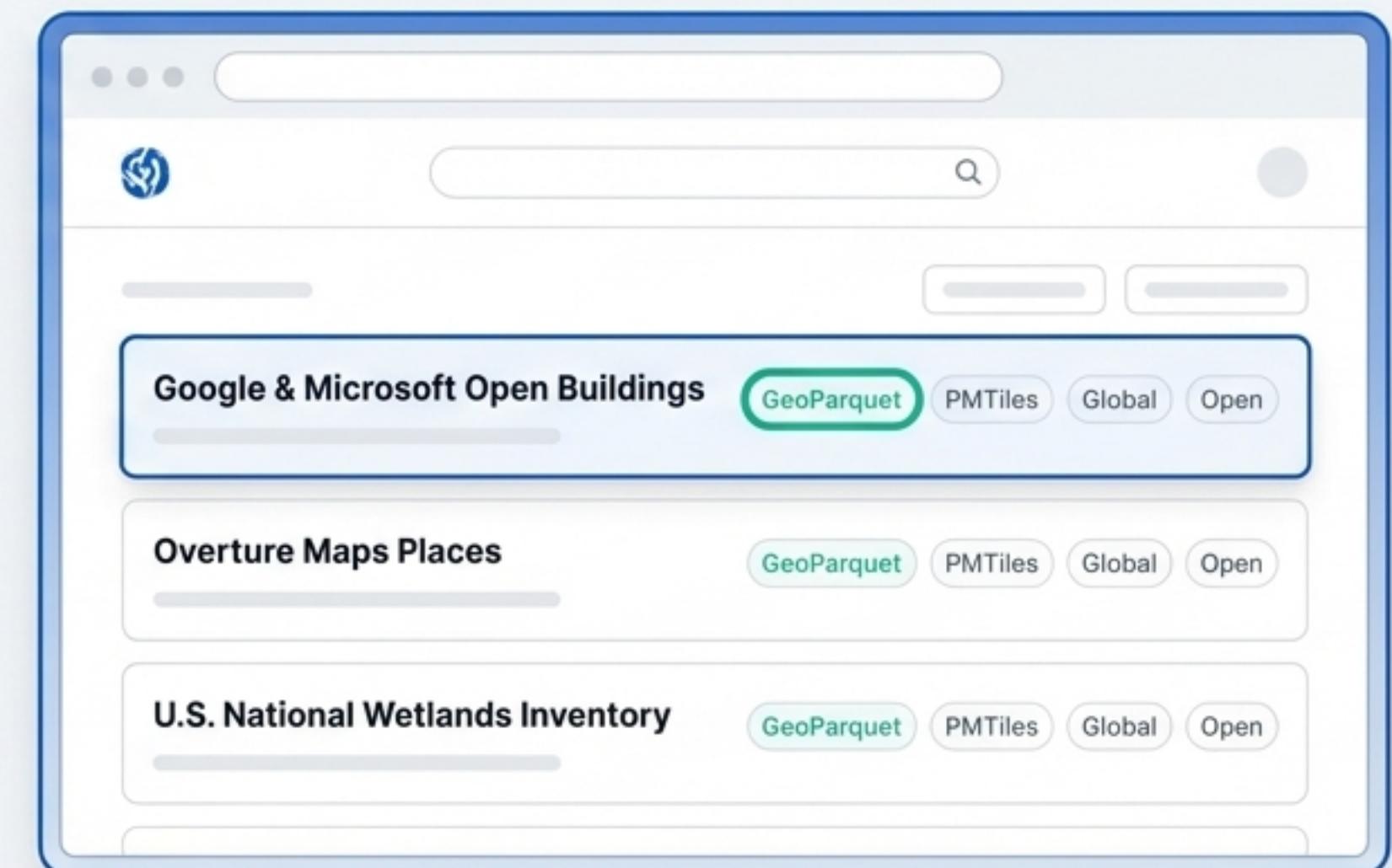
**Extensible:** 'spatial', 'httpfs' 등 강력한 확장 기능

# The Source: 분석에 최적화된 데이터 플랫폼 Source Cooperative

Source Cooperative는 대규모 지리 공간 데이터셋을 클라우드에 최적화된 포맷(GeoParquet, PMTiles 등)으로 제공하는 데이터 플랫폼입니다. 데이터를 다운로드할 필요 없이 누구나 자유롭게 접근하고 분석할 수 있습니다.

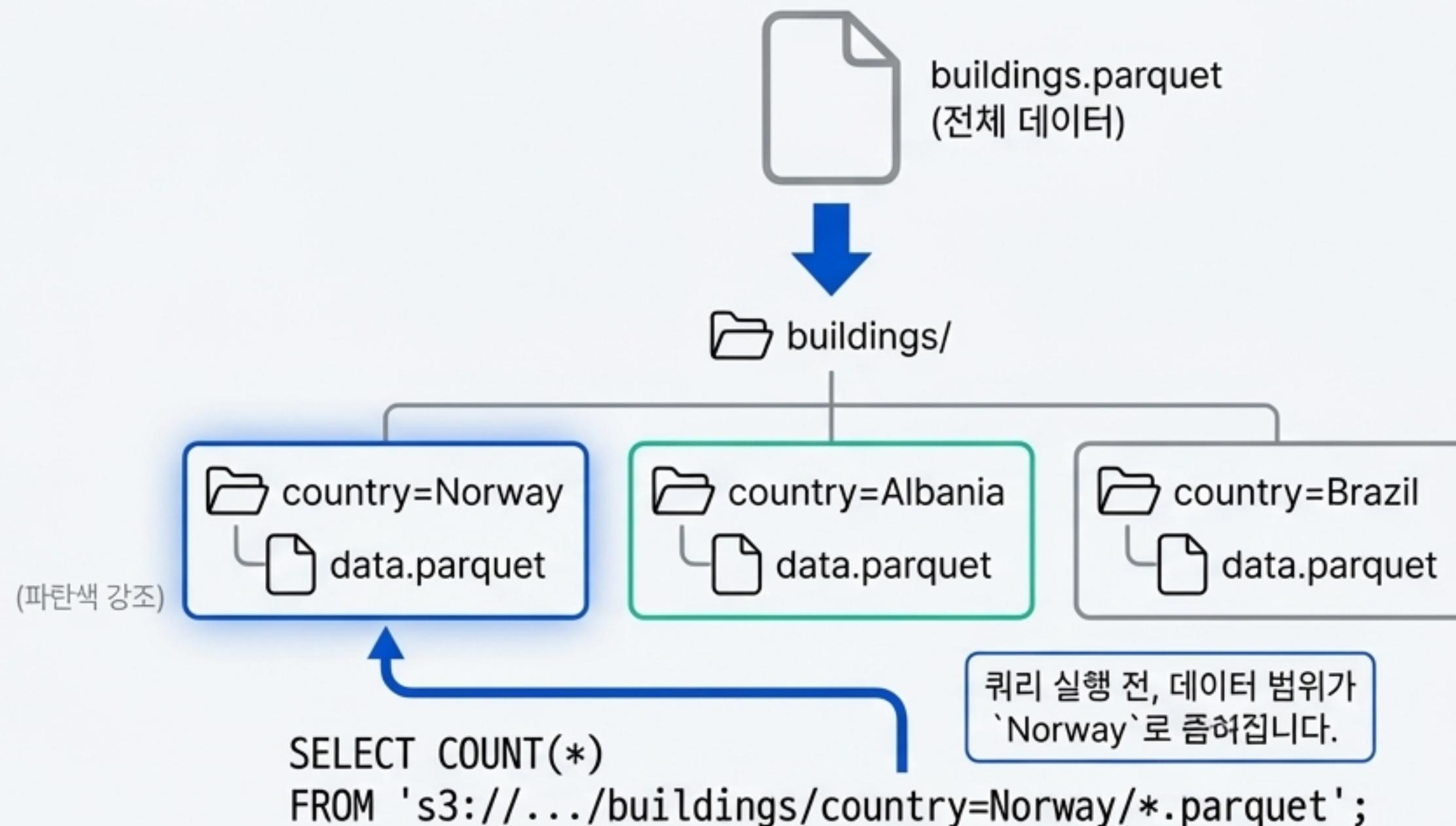
## Key Datasets Available

- **Google & Microsoft Open Buildings**
- **Overture Maps Places**
- **U.S. National Wetlands Inventory**
- **...and more.**



# 쿼리가 아닌 파일 시스템으로 필터링: 파티셔닝의 힘

파티셔닝은 대규모 데이터셋을 논리적인 하위 폴더(예: 국가, 날짜)로 미리 분할하는 기술입니다. 쿼리 엔진은 전체 데이터를 스캔할 필요 없이 필요한 폴더의 데이터만 읽어 성능을 극대화합니다.



# 이제 직접 증명해 보겠습니다: 실전 쿼리 준비

별도의 서버 설정 없이, DuckDB CLI 또는 Python 환경에서 몇 줄의 명령어로 모든 준비가 끝납니다. 원격 파일 시스템과 공간 분석 기능을 활성화합니다.

-- 1. DuckDB 실행 (Start DuckDB)

```
> duckdb
```

-- 2. 필수 확장 기능 설치 및 로드 (Install & Load required extensions)

```
INSTALL spatial;
```

```
LOAD spatial;
```

```
INSTALL httpfs;
```

```
LOAD httpfs;
```

-- 3. AWS S3 리전 설정 (Set the AWS S3 Region for Source Coop)

```
SET s3_region = 'us-west-2';
```

이것이 전부입니다. 이제 수백만 건의 데이터를 쿼리할 준비가 되었습니다.

# 노르웨이의 건물 370만 개, 단 몇 초 만에 확인

데이터 다운로드 없이, Source Cooperative에 호스팅된 GeoParquet 파일을 직접 쿼리하여 노르웨이의 전체 건물 수를 집계합니다.

```
SELECT COUNT(*)  
FROM 's3://us-west-  
2.beta.source.coop/google-  
microsoft-open-  
buildings/v3/geoparquet/by-  
country/country=Norway/part-  
0.parquet';
```

**3,789,171**

실행 시간: ~4.5초

로컬 데이터베이스나 파일 없이,  
클라우드 데이터를 즉시 분석했습니다.

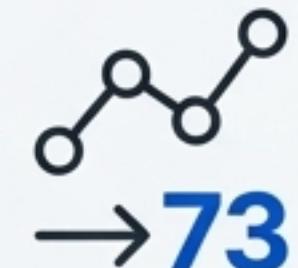
# 와일드카드를 이용한 탐색: 'S'로 시작하는 모든 국가의 건물 7,300만 개

DuckDB는 파일 경로에 와일드카드(`\*`)를 사용하여 여러 파티션의 데이터를 동시에 쿼리할 수 있습니다. 이를 통해 즉각적인 대규모 탐색 분석이 가능해집니다.

```
SELECT COUNT(*)  
FROM 's3://.../by-country/country=S*/*.parquet';
```

~73,000,000

남아프리카, 스페인, 스웨덴 등  
'S'로 시작하는 모든 국가의 데이터를  
한 번의 쿼리로 집계합니다.



# 공간 필터링: 오슬로 지역 건물 데이터만 추출하여 테이블 생성

원격 파일 전체를 대상으로 `ST\_Within`과 같은 공간 함수를 직접 실행할 수 있습니다. 특정 지역(오슬로) 내의 건물 데이터만 필터링하여 분석을 위한 인메모리 테이블 `oslo\_buildings`를 생성합니다.

```
CREATE OR REPLACE TABLE oslo_buildings AS
SELECT
    source,
    area_in_meters,
    ST_GeomFromWKB(geometry) as geom -- WKB를 지오메트리로 변환
FROM 's3://.../country=Norway/*.parquet'
WHERE ST_Within(
    ST_GeomFromWKB(geometry),
    ST_Buffer(ST_Point(10.75, 59.91), 0.1) -- 오슬로 중심점 버퍼
);
```

결과: **62,841개의** 건물이 필터링되어 `oslo\_buildings` 테이블에 저장되었습니다.

# 분석에서 시각화로: 쿼리 결과를 GeoJSON 파일로 내보내기

DuckDB의 강력한 `COPY` 명령어를 사용하면 테이블의 데이터를 다양한 포맷으로 손쉽게 내보낼 수 있습니다. GDAL 드라이버를 활용하여 공간 데이터를 GeoJSON 형식으로 직접 변환합니다.

```
COPY oslo_buildings TO 'oslo.geojson'  
WITH (  
    FORMAT GDAL,  
    DRIVER 'GeoJSON'  
)
```



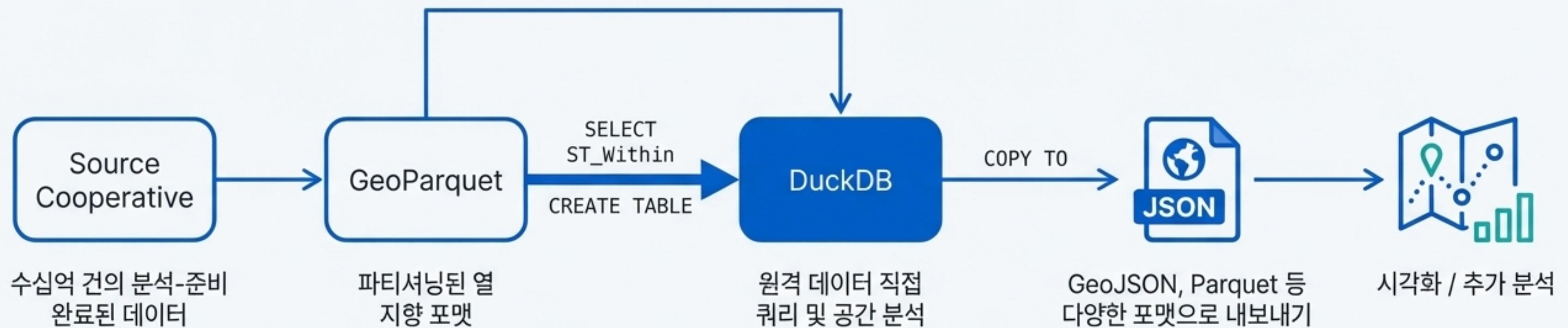
2.3초 만에 6만여 개의 건물 폴리곤이 포함된 `oslo.geojson` 파일 생성 완료.

# 최종 결과: 단 몇 분 만에 클라우드 데이터에서 지도로



데이터 검색, 다운로드, 변환, 로딩의 복잡한 과정 없이, 몇 줄의 SQL만으로 클라우드의 원시 데이터로부터 직접 시각적인 결과물을 얻었습니다.

# Modern Geospatial Workflow 요약



**Speed. Scale. Simplicity.** 이 세 가지가 새로운 워크플로우의 핵심입니다.

# 직접 시작해 보세요

오늘 소개된 모든 코드와 자료는 아래 리소스를 통해  
직접 확인하고 실행해 보실 수 있습니다.



## Demo Code

[github.com/mattforrest/  
youtube-tutorials](https://github.com/mattforrest/youtube-tutorials)

오늘 시연에 사용된 전체 SQL 코드를  
제공합니다.



## DuckDB & Extensions

[duckdb.org](https://duckdb.org)

DuckDB를 설치하고 `spatial` 확장  
기능 문서를 확인하세요.



Source  
Cooperative

## Explore Datasets

[source.coop](https://source.coop)

계정을 만들고 다양한 클라우드 최적화  
데이터셋을 탐색해 보세요.

