Clase 033 — Polars: DataFrames modernos

Parte: 0 — Prerrequisitos · Fuente: Polars User Guide + Vink (2020+) · Duración estimada: 75 min.

Clase 033 — Polars: DataFrames modernos

Parte: 0 — Prerrequisitos · Fuente: Polars User Guide + Vink (2020+) · Duración estimada: 75 min.

Objetivo

Conocer Polars — la librería de DataFrames moderna (Rust + Arrow) que está reemplazando a pandas en proyectos donde performance importa. Aprender su API (similar a pandas pero con expresiones lazy y paralelismo automático) y entender cuándo conviene Polars sobre pandas (datasets > 1 GB, pipelines con muchas transformaciones, multi-core).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Instalar Polars (pip install polars) y leer datos con pl.read_csv, pl.read_parquet, pl.scan_csv (lazy).
- Aplicar la API de expresiones: df.select(pl.col('precio').sum()), pl.col('x').filter(pl.col('y') > 0).mean().
- Diferenciar eager (DataFrame) de lazy (LazyFrame) — y por qué lazy permite optimizaciones del query planner.
- Hacer groupby, join, pivot, unpivot con sintaxis Polars y comparar con pandas.
- Reconocer el speedup típico: 5-30× sobre pandas en operaciones comunes (single-machine, multi-core).

Temas

- Polars vs pandas vs DuckDB: el panorama 2026.
- Arrow como formato columnar in-memory.
- Eager (DataFrame) vs Lazy (LazyFrame).
- Expresiones encadenables: pl.col(...).operation().
- Query optimization automática: predicate pushdown, projection pushdown.
- Multi-threading automático.

Definiciones y características

- Polars: librería de DataFrames escrita en Rust, ABI estable en Python.
- pl.DataFrame: estructura eager — los datos están en RAM.
- pl.LazyFrame: descripción de pipeline; ejecución diferida hasta .collect().
- Expresión (pl.col, pl.lit): describe operación sobre columna; se compone y optimiza.
- pl.scan_csv / scan_parquet: leer lazy — solo se materializa lo que necesitás.
- Query optimization: el planner reordena filtros y projections para minimizar trabajo.
- streaming: para datasets mayores que RAM (collect(streaming=True)).

Dataset / recursos

- NYC Taxi (~100 MB Parquet) o cualquier CSV > 100 MB.
- Librerías: polars, pandas (para comparar), pyarrow.

Ejercicios

1. Eager básico: df = pl.read_csv('archivo.csv'); df.head(); df.describe(). Comparar con pandas.
2. Expresiones: df.filter(pl.col('precio') > 100).group_by('categoria').agg(pl.col('precio').mean().alias('precio_medio')).
3. Lazy + collect: lf = pl.scan_csv('big.csv').filter(...).group_by(...).agg(...); result = lf.collect(). Comparar tiempo vs eager.
4. Query plan: lf.explain() muestra el plan optimizado. Identificar predicate pushdown.
5. Pandas ↔ Polars: df.to_pandas() y pl.from_pandas(pd_df). Útil para mantener compatibilidad gradual.

Homework verificable

Sobre un dataset > 500 MB (NYC Taxi recomendado):

1. Pipeline en pandas y en Polars (lazy): filtrar viajes con tip_amount > 5, agrupar por hora del día, calcular media y desviación.
2. Medir wall time de cada uno.
3. Reportar speedup.

Criterio de aceptación: Polars debe ser ≥ 3× más rápido que pandas; los resultados numéricos deben coincidir (±0.001).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Mezclar APIs (df['col'].sum() estilo panda	Funciona pero no usa optimizaciones. Fix:
LazyFrame no devuelve datos	Es perezoso. Fix: .collect() al final del
Sin streaming=True en datasets enormes	OOM. Fix: lf.collect(streaming=True).
Columnas con tipos string mezclados → sche	Polars es estricto con tipos. Fix: pl.read
Esperar índices como en pandas	Polars NO tiene índice. La columna que qui

Preguntas frecuentes

¿Polars reemplaza a pandas?

Para data engineering / pipelines: cada vez más sí. Para ad-hoc analysis en Jupyter y compatibilidad con todo el ecosistema (sklearn, statsmodels, plotting): pandas sigue siendo el default. Ambos coexisten.

¿Polars o DuckDB?

Polars: API DataFrame Python-first. DuckDB: SQL embebido, mejor para joins masivos y queries SQL. Se complementan: muchos pipelines usan DuckDB para reads grandes + Polars para transformaciones.

¿pl.col vs df['col']?

pl.col es una expresión (reusable, optimizable). df['col'] materializa una Series — útil para inspección pero no para pipelines.

¿Polars en producción?

Sí. Empresas como Goldman Sachs, Citadel, Bank of America lo usan. Estable, mantenimiento activo,

sponsoreado por la compañía Polars Cloud.

¿Polars soporta lazy streaming desde S3?

Sí: pl.scan_parquet('s3://bucket/*.parquet') con credentials configuradas. Lee solo lo que necesita.

Referencias

- Polars User Guide
- Polars Cookbook
- McKinney (2022 blog), Apache Arrow and the "10 Things I Hate About pandas".
- Comparativa: <https://h2oai.github.io/db-benchmark/> — Polars consistentemente top.

Siguiente clase

Clase 034 — Parquet, Arrow, PyArrow, DuckDB

Apéndice: notebook (primer bloque)

Comparamos Polars vs pandas sobre un dataset sintético generado en disco. Requiere: pip install polars pandas pyarrow numpy.

```
import time, tempfile, os
from pathlib import Path
import numpy as np
import pandas as pd

try:
    import polars as pl
except ImportError as e:
    raise ImportError('Instalá polars: pip install polars pyarrow') from e

print(f'polars {pl.__version__} | pandas {pd.__version__} | numpy {np.__version__}')
```

Archivos complementarios

- notebook.ipynb