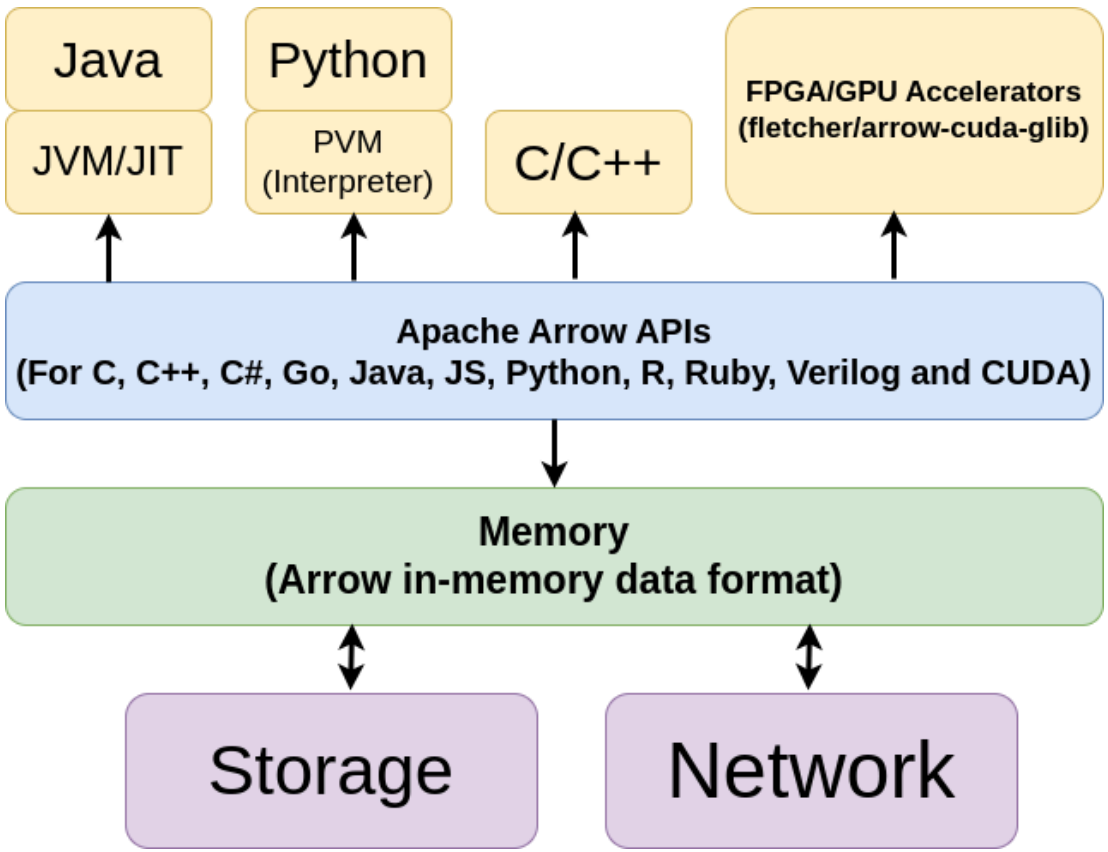
Apache Arrow

Hermilo

4 de Septiembre 2025

¿Qué es Apache Arrow?



APACHE ARROW

- Estandar de intercambio de datos.
- Un formato in-memory.
- Un formato de Red.
- Un formato de Almacenamiento.
- Biblioteca de I/O
- Biblioteca de cómputo vectorizado.
- Biblioteca de manejo de dataframes.
- Un query engine.
- Administrador de datasets particionados.

- El proyecto **Apache Arrow** es un gran esfuerzo dirigido a resolver los problemas fundamentales de la analítica de datos.
- Está dirigido a proporcionar una experiencia **write everywhere, run everywhere**, por lo que es fácil perderse si nos se sabe dónde empezar.
- **PyArrow** es el punto de entrada al ecosistema de Apache Arrow para desarrolladas de Python, y es el medio que ofrece el acceso a los muchos beneficios de Arrow.

PyArrow

- Apache Arrow nació como un **Formato Columnar de Datos**.
- De manera que el tipo fundamental de datos en PyArrow es una **columna de datos** la cual es expuesta por mendio de un objeto `pyarrow.Array`.
- A este nivel, PyArrow es similar a los arreglos 1D de **NumPy**.

PyArrow Arrays

```python
import pyarrow as pa

## Arrays can be made of numbers
>>> pa.array([1, 2, 3, 4, 5])

## Or strings
>>> pa.array(["A", "B", "C", "D", "E"])

## Or even complex objects
>>> pa.array([{"a" : 5}, {"b" : 10}])

## Arrays can also be masked
>>> pa.array([1, 2, 3, 4, 5],
mask = pa.array([True, False, True, False,
True]) )
```

Comparado a los arreglos clásicos de NumPy, los arreglos de PyArrow son un poco más complejos.

- Estandar de intercambio de datos.

Manipulación de Arreglos

- La comunidad Arrow ha creado una implementación de referencia de código abierto de un motor de cálculo y consulta basado en el formato Arrow llamado **Acero**.
- PyArrow proporciona también acceso al motor de cálculo **Acero** mediante el módulo `pyarrow.compute`.
- Para este fin, existe la biblioteca Acero para facilitar diversas implementaciones de alto rendimiento de funciones que operan con datos con formato Arrow, junto con la creación de planes de ejecución de filtrados, agregaciones y transformaciones para flujos de datos.

```python
import pyarrow.compute as pc

>>> arr = pa.array([1, 2, 3, 4, 5])
>>> pc.multiply(arr, 2)
>>> pc.value_counts(arr)
>>> pc.min_max(arr)
```

PyArrow Tables

- Como los arreglos son “columnas”, estos puden ser agrupados para formar `pyarrow.Table`.
- Las tablas son constituidas por `pyarrow.ChunkedArray` de manera que concatenar filas a ellas resultan una operación barata.
- A este nivel, PyArrow es similar a los **DataFrames de Pandas**.

PyArrow Tables

```python
>>> table = pa.table([
  pa.array([1, 2, 3, 4, 5]),
  pa.array(["a", "b", "c", "d", "e"]),
  pa.array([1.0, 2.0, 3.0, 4.0, 5.0]),
], names = ["col1", "col2", "col3"])
>>> table.take([0, 1, 4])
col1: [[1,2,5]]
col2: [["a","b","e"]]
col3: [[1,2,5]]
>>> table.schema
col1: int64
col2: string
col3: double
```

- Comparado a Pandas, las tablas de PyArrow son completamente implementadas en C++ y nunca modifican datos **in-place**.
- `Tables` están basadas en **ChunkedArrays** de manera que concatenar datos es una operación **zero copy**.
- El motor de cómputo Acero en Arrow es capaz de proporcionar muchas de las operaciones comunes de la analítica de datos como joining, filtrados y agregaciones de datos.

Ejecutando Analítica de Datos

- El motor de cómputo de Acero potencia las capacidades de análisis y transformación disponibles en las tablas
- Muchas funciones de `pyarrow.compute` proporcionan kernels que operan en tablas y las `Table` exponen métodos de join, filtrado y agrupación.

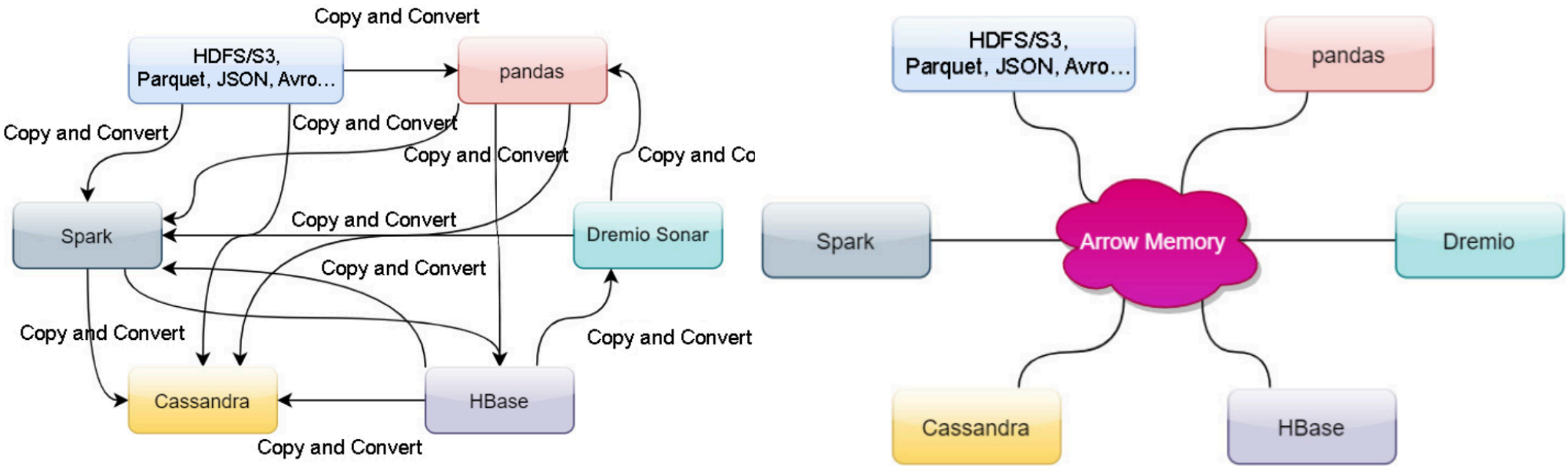
```python
import pyarrow as pa
import pyarrow.compute as pc

>>> table = pa.table([
  pa.array(["a", "a", "b", "b", "c", "d", "e", "c"]),
  pa.array([11, 20, 3, 4, 5, 1, 4, 10])
], names = ["keys", "values"])
>>> table.filter(pc.field("values") == )
>>> table.group_by("keys").aggregate(["keys", "sum"])
>>> table1 = pa.table({"id" : [1, 2, 3],
  "year" : [2020, 2022, 2019]
})
>>> table2 = pa.table({"id" : [3, 4],
  "n_legs" : [5, 100],
  "animal" : ["Brittle Stars", "Centipede"]
})
>>> table1.join(table2, keys = "id")
```

PyArrow, NumPy y Pandas

- Uno de las metas de diseño originales de Apache Arrow fue permitir el fácil intercambio de datos sin el costo de convertir a través de múltiples formatos o serializarlos antes de transferirlos.
- PyArrow proporciona el soporte de conversión de datos copy-free de y hacía pandas y numpy.
- Si tenemos datos en PyArrow podemos invocar al método `to_numpy` del objeto `pyarrow.Array` y `to_pandas` en `pyarrow.Array` y `pyarrow.Table` para convertirlos a objetos de pandas o numpy sin enfrentar ningún costo adicional de conversión.

PyArrow, NumPy y Pandas



Tomado de **In-Memory Analytics with Apache Arrow**

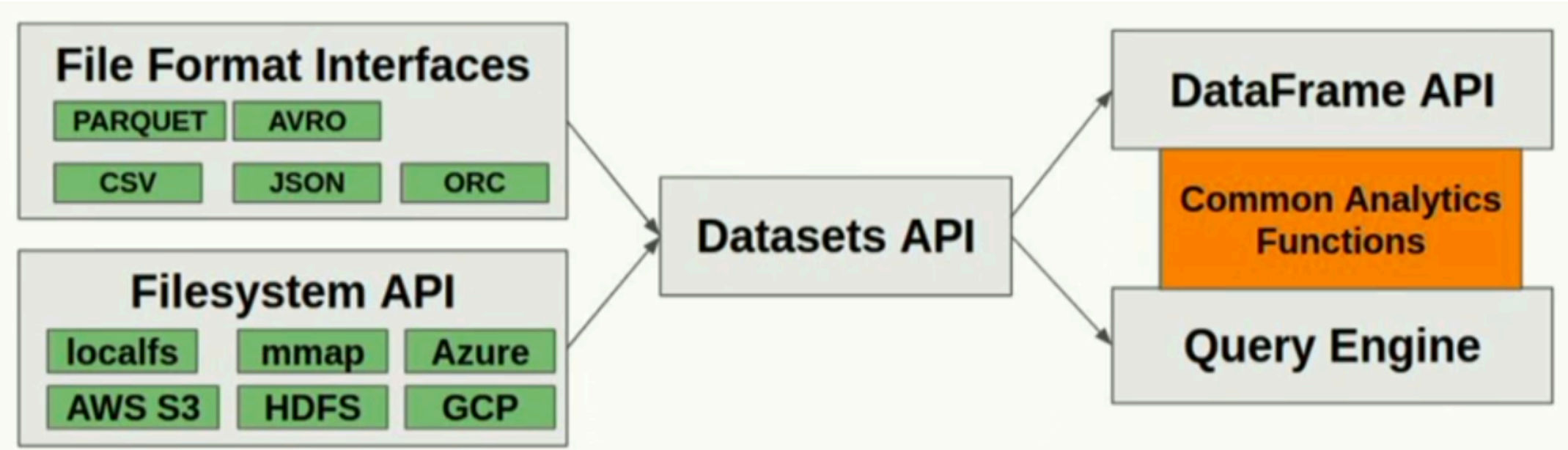
Y es rápido!!

```python
>>> data = [a % 5 for a in range(100_000_000)]
>>> npdata = np.array(data)
>>> padata = np.array(data)
>>> import timeit
>>> timeit.timeit(
  lambda: np.unique(npdata, return_counts = True),
  number = 1
)
>>> timeit.timeit(
  lambda: pc.value_counts(padata),
  number=1
)
```

Datasets

- **Datasets** son una abstracción que permite trabajar con grandes conjuntos tabulares de datos, potencialmente más grandes que la memoria y distribuidos a través de múltiples archivos.
- Datasets proporciona un acceso **lazy**, evitando la necesidad de cargar todos los datos en memoria inmediatamente.
- Datasets son compatibles con el motor de cómputo Acero en la mayoría de los casos en lugar de las tablas.

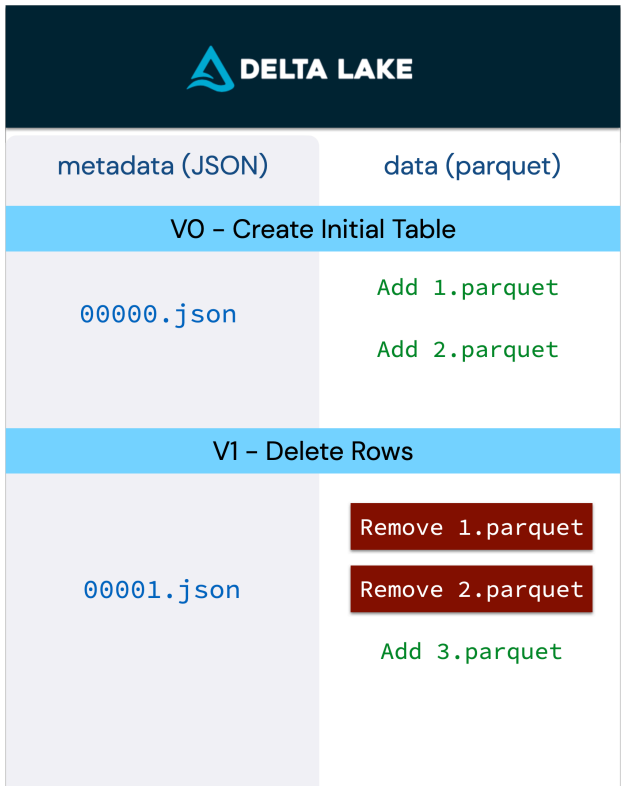
Datasets

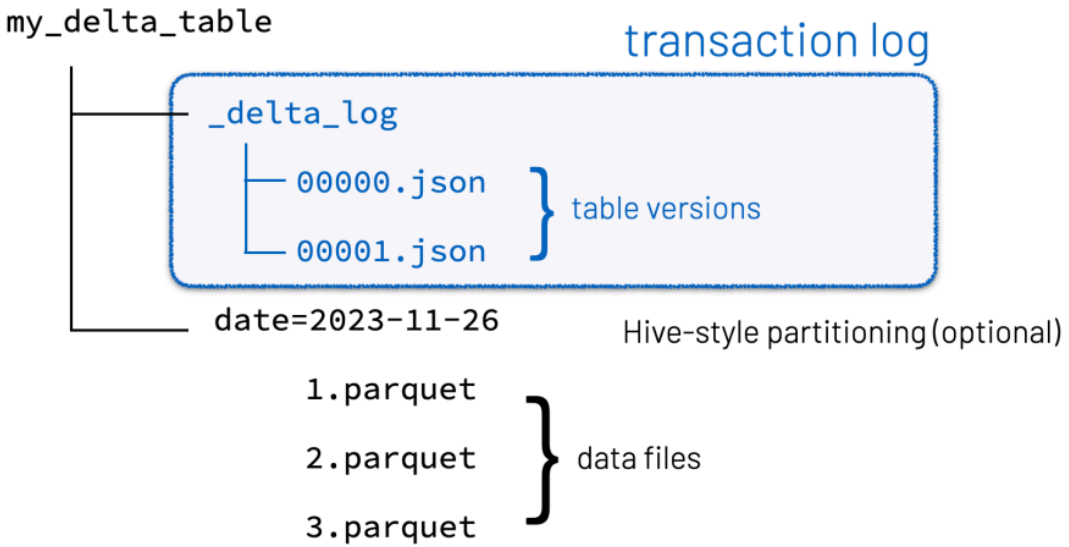

File Format Interfaces
PARQUET
AVRO
CSV
JSON
ORC
Filesystem API
localfs
mmap
Azure
AWS S3
HDFS
GCP
Datasets API
DataFrame API
Common Analytics Functions
Query Engine

Datasets

- En el actual ecosistema de **data lakes** y **lakehouses**, muchos conjuntos de datos son ahora enormes colecciones de archivos en estructuras de directorios particionados en lugar de un solo archivo.
- La API Datasets proporciona una serie de utilidades para interactuar fácilmente con conjuntos de datos grandes, distribuidos y posiblemente particionados que se distribuyen en múltiples archivos.

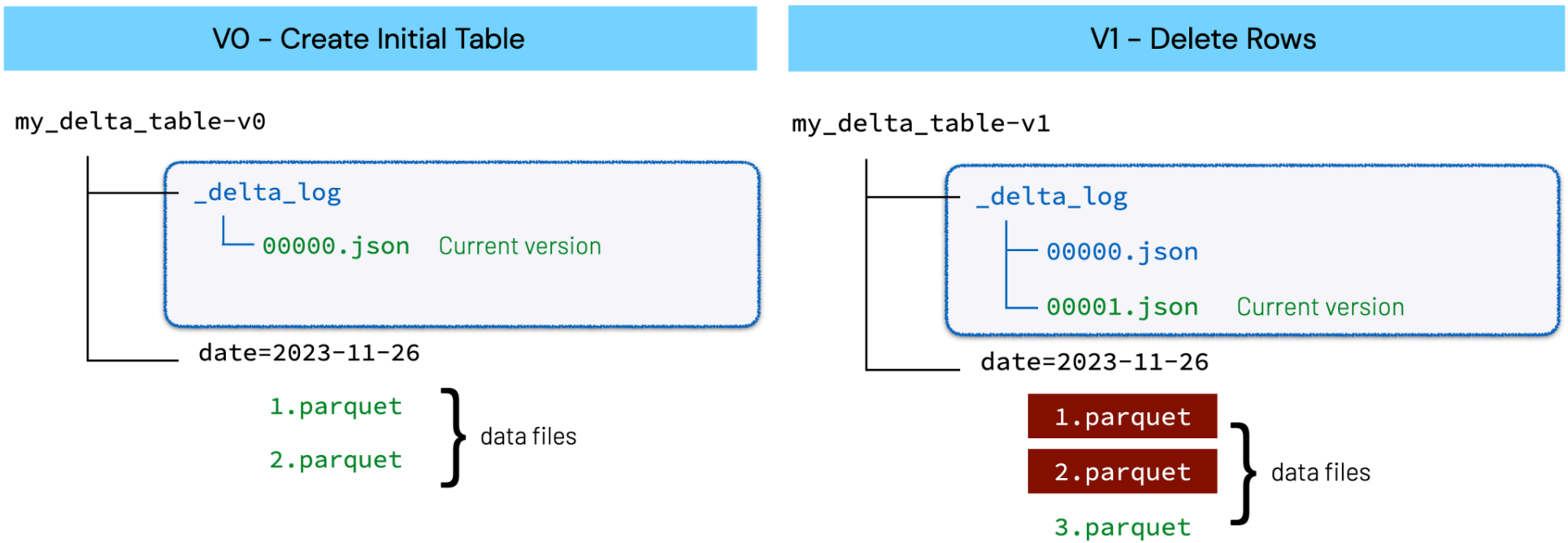
Data Lakes : Delta Lake





Tomado de Denny Lee, “Understanding the Delta Lake Transaction Log at the File Level”, Denny Lee (blog), November 26, 2023.

Data Lakes : Delta Lake



Tomado de Denny Lee, “Understanding the Delta Lake Transaction Log at the File Level”, Denny Lee (blog), November 26, 2023.

PyArrow Datasets

```python
>>> from deltalake import DeltaTable
>>> import pyarrow.dataset as ds

# Cargamos la Delta table
>>> dt = DeltaTable("datos/atlas")

# Convertimos a PyArrow Dataset
>>> dataset = dt.to_pyarrow_dataset()

# Observamos los primeros 10 registros
>>> dataset.head(10)

# Filtramos sólo registros de México en 2024
```

- `Datasets` proporciona acceso lazy a grandes volúmenes de datos guardados en cualquiera de los formatos soportados por

```
>>> filtrado =
dataset.scanner(filter=ds.field("year") ==
2024)#.to_table()
```