

APIs Estructuradas: DataFrames y SQL

Curso académico 2017/18

The background of the slide features a complex, abstract geometric pattern composed of numerous overlapping wireframe cubes. These cubes are rendered in a dark gray color, creating a sense of depth and perspective. The lighting highlights the edges of the cubes, emphasizing their three-dimensional nature against a lighter gray background.

APIs Estructuradas: Introducción

APIs Estructuradas

- Herramientas para manipular todo tipo de datos
 - No estructurados: Texto, ficheros de Log
 - Semiestructurados: CSV
 - Estructurados: Parquet
- 3 Componentes principales:
 - Datasets
 - DataFrames
 - Tablas y Vistas SQL

APIs Estructuradas

- Aplicables a procesamientos ***batch*** y ***streaming***
- “Unificado”: mismo (o casi igual) código aplicable a procesamiento batch y streaming

DataFrames & Datasets

- Colecciones de datos (tablas) distribuídas
- Filas y columnas
- Cada columna debe tener el mismo número de filas que el resto de columnas (puede haber valores null)
- Cada columna tiene un tipo asociado, que es consistente para todos los valores

DataFrames & Datasets

- Para Spark son “planes de ejecución”, inmutables, evaluados de forma perezosa, que especifican qué operaciones hay que aplicar a un conjunto de datos de origen para generar una salida determinada
- Cuando ejecutamos una acción, estamos diciendo a Spark que ejecute dicho plan y devuelva el resultado.

Ejercicios de Clase: Instrucciones

- Sigue las instrucciones de descarga para los ejercicios de hoy que hay en Aula Virtual

Esquemas y Tipos de Datos



Esquema

- Define los nombres y tipos de las columnas de un DataFrame
- Pueden definirse manualmente o leerlos directamente de una fuente de datos ("schema on read")
- Un esquema consiste en **tipos de datos**

Tipos de Datos en Spark

- Cada lenguaje de programación define sus tipos de datos: diferentes formas y propiedades que puede tener una variable o dato (“integer”, “string”, “array”....)
- Spark también define sus propios tipos de datos, funcionando de forma similar a un lenguaje de programación
- Los tipos de datos Spark se correlacionan con los tipos de cada lenguaje compatible con Spark: Scala, Java, Python, SQL y R

Tipos de Datos en Spark

- Aunque utilizemos la API de DataFrames desde código Python, la mayoría de las manipulaciones de datos utilizan tipos Spark, no tipos de Python.
- Ejemplo: el siguiente código no suma usando Python, sino usando directamente tipos Spark:

```
df = spark.range(500).toDF("number")
df.select(df["number"] + 10).show()
```

Tipos de Datos en Spark

```
df = spark.range(500).toDF("number")
df.select(df["number"] + 10).show()
```

- Spark convierte las expresiones escritas en Python en una representación interna (Catalyst) de los mismos tipos.
- La operación de suma se realiza sobre esa representación interna

DataFrames vs Datasets

- Dentro de las APIs estructuradas de Spark, se distingue entre DataFrames y Datasets
- Ambas tienen tipos, pero en los DataFrames, los tipos se evalúan (se comprueba si se conforman al esquema) de forma dinámica (en tiempo de ejecución)
- En los Datasets se evalúan de forma estática (en tiempo de compilación)
- Datasets sólo están disponibles en la JVM (Java, Scala)

DataFrames vs Datasets

- Nosotros trabajaremos siempre con DataFrames (Python)
- DataFrames = Datasets de Row (en Scala)
- Siempre que usamos DataFrames estamos aprovechando los tipos internos optimizados de Spark (Row) y esto es común a todos los lenguajes desde los que se puede utilizar Spark

Columnas

- Representan un
 - tipo simple (integer, string...)
 - tipo complejo (array, map...)
 - Null
- Similar a las columnas en una tabla relacional

Filas

- Representan un registro de datos
- Todos los registros de un DataFrame son de tipo **Row**

```
spark.range(2).collect()
```

```
[Row(id=0), Row(id=1)]
```

Trabajando Con Tipos

2

3

4

5

6

Trabajando con tipos

```
from pyspark.sql.types import *
b = ByteType()
```

Trabajando con tipos

| Tipo de datos | Tipo Python | Uso |
|---------------|-----------------|---------------|
| Byte | int ó long | ByteType() |
| Short | int ó long | ShortType() |
| Integer | int ó long | IntegerType() |
| Long | long | LongType() |
| Float | float | FloatType() |
| Double | float | DoubleType() |
| Decimal | decimal.Decimal | DecimalType() |
| String | string | StringType() |
| Binary | bytearray | BinaryType() |
| Boolean | bool | BooleanType() |

Trabajando con tipos

| Tipo de datos | Tipo Python | Uso |
|---------------|---------------------------------------------|------------------------------------------------------------------|
| Timestamp | datetime.datetime | TimestampType() |
| Date | datetime.Date | DateType() |
| Array | list, tuple, array | ArrayType(elementType) |
| Map | dict | MapType(keyType, valueType) |
| Struct | list, tuple | StructType(fields) fields es una lista de StructFields |
| StructField | depende del dataType del StructField | StructField(name, dataType) |

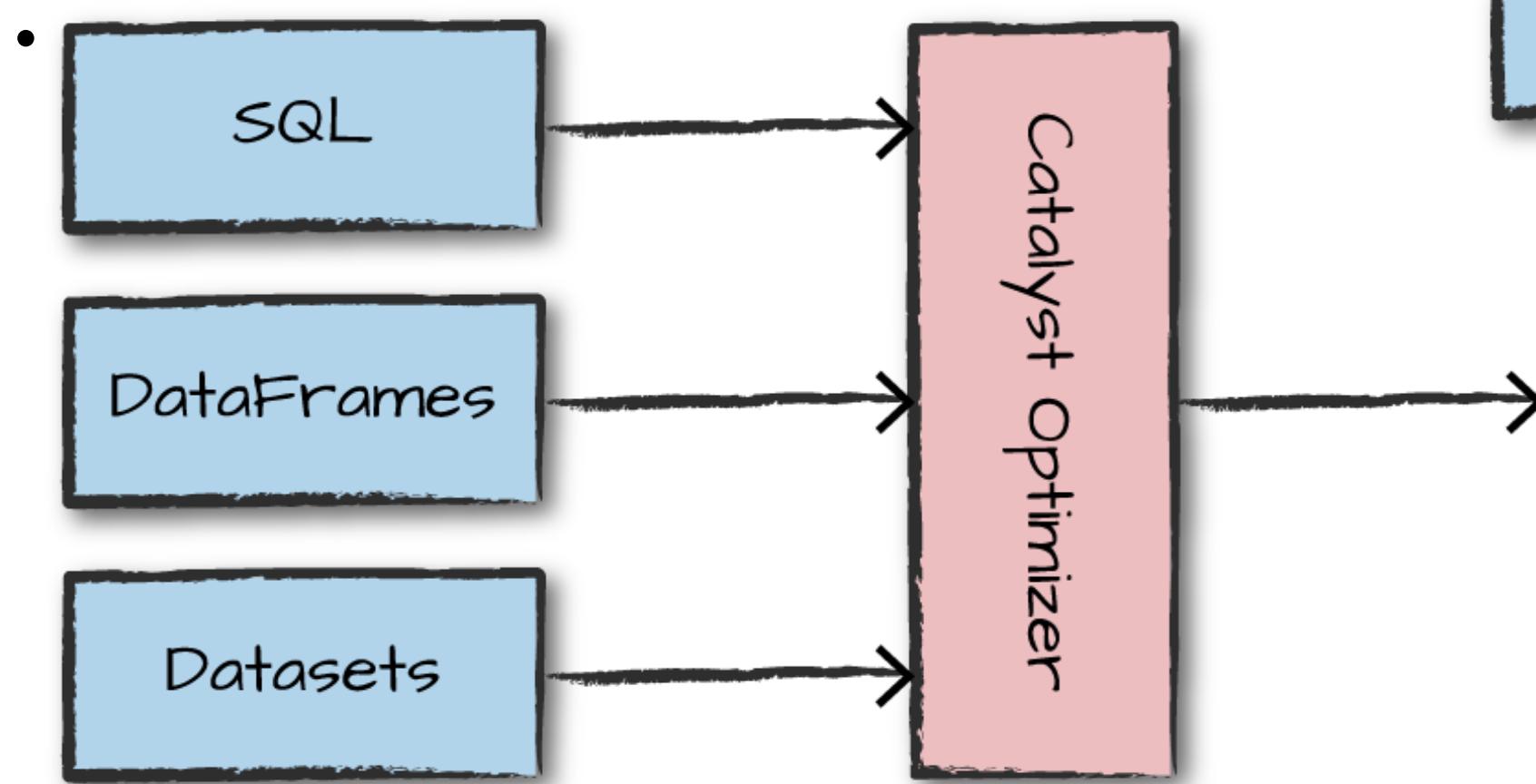
Flujo de Ejecución en las APIs Estructuradas



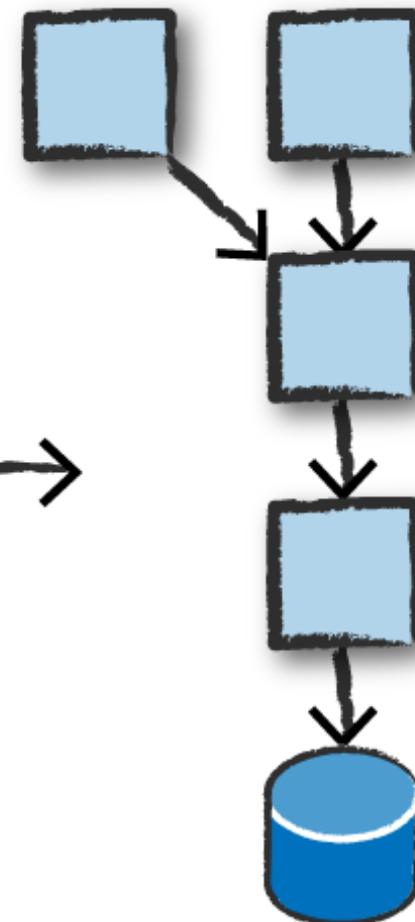
Flujo de Ejecución

- Pasos desde que escribimos nuestro código hasta que se ejecuta de forma distribuída en el clúster
 1. Escribir código (DataFrame / SQL)
 2. Si es código válido => Spark lo convierte en un **Plan Lógico**
 3. Spark transforma el **Plan Lógico** a **Plan Físico**, aplicando todas las optimizaciones posibles
 4. Spark ejecuta el **Plan Físico** (manipulaciones a nivel de RDDs) en el clúster

Flujo de Ejecución



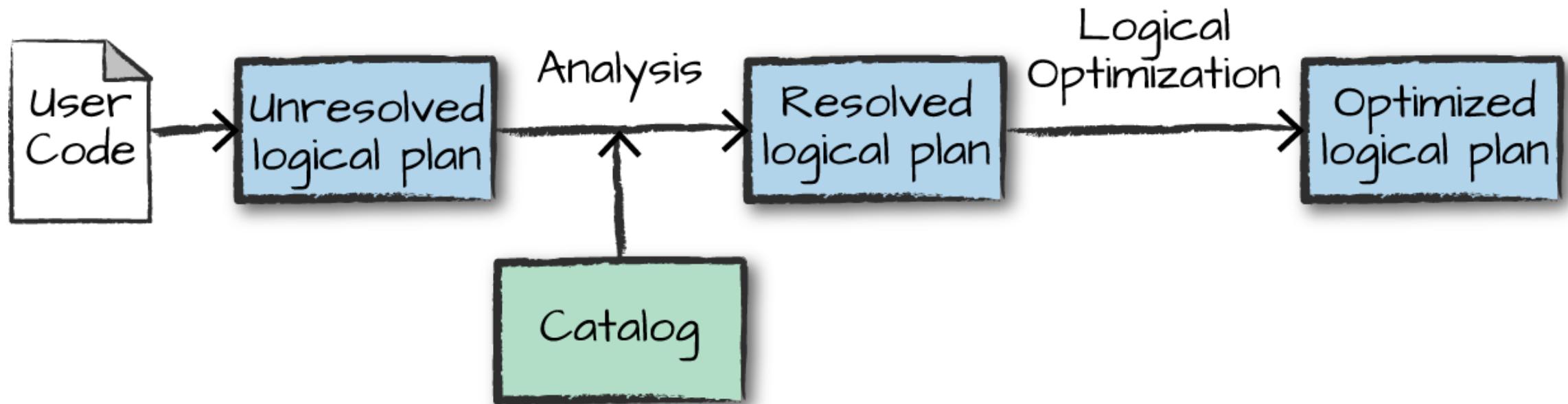
Physical Plan



Flujo de Ejecución: Plan Lógico

1. Transformación directa de código de usuario a “Plan Lógico no resuelto” (conjunto de transformaciones abstractas traducidas a una representación optimizada). “No resuelto” porque aún no se ha verificado si los DataFrames, tablas y columnas existen
2. El **analyzer** consulta el catálogo y crea un “Plan Lógico Resuelto”, en el que los DFs/tablas y columnas han sido ya verificados.
3. Por último el **optimizador Catalyst** aplica una serie de reglas que resultan en un **Plan Lógico Optimizado**

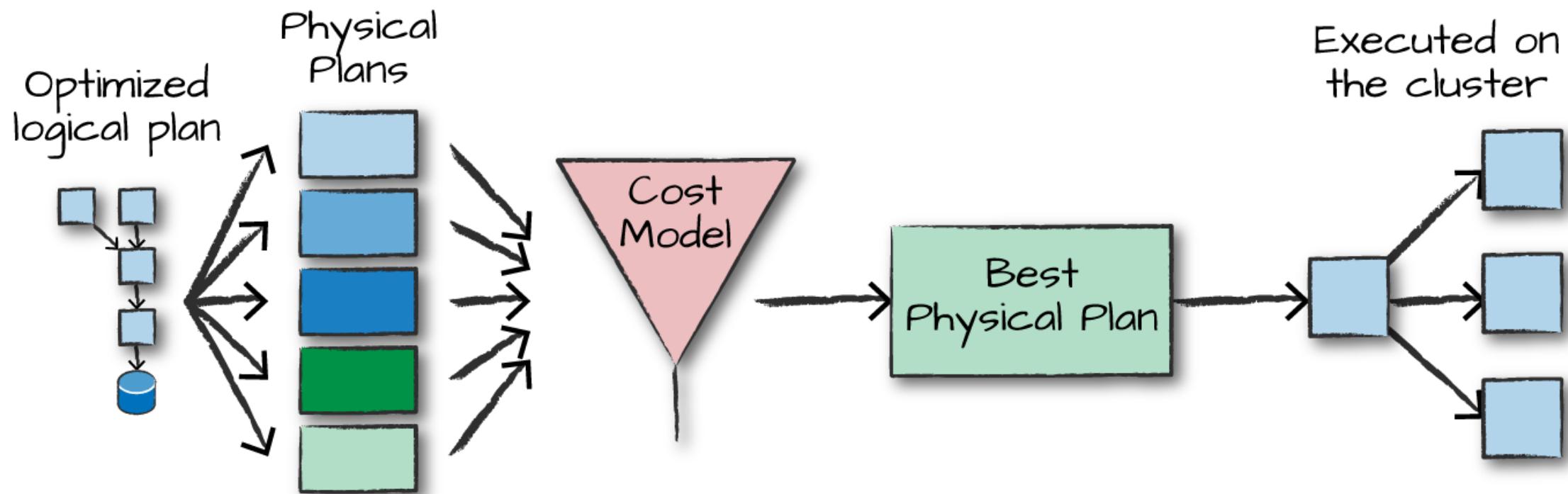
Flujo de Ejecución: Plan Lógico



Flujo de Ejecución: Plan Físico

1. Una vez obtenido un Plan Lógico, Spark comienza la creación de un Plan Físico de ejecución. Para ello genera diferentes estrategias físicas de ejecución y las evalúa contra un modelo de coste (e.g: diferentes estrategias para realizar el “join” de dos DataFrames)
2. El **Plan Físico** resultante consiste en una serie de RDDs y transformaciones sobre los mismos. Es decir, Spark **“compila” nuestras queries sobre DataFrames y SQL a transformaciones de más bajo nivel sobre RDDs**

Flujo de Ejecución: Plan Físico



Flujo de Ejecución: Ejecución

- Cuando Spark ha seleccionado un Plan Físico, ejecuta ese código sobre RDDs, la API de bajo nivel de Spark.
- En esta fase también se producen optimizaciones en tiempo de ejecución
- Por último, Spark devuelve el resultado al usuario.

Operaciones Básicas



DataFrames: operaciones básicas

- Ya sabemos que un DataFrame consiste en:
 - N filas ó registros, de tipo Row
 - M columnas, de varios tipos
- Un **esquema** define el nombre y los tipos de dato de cada columna
- El **particionamiento** de un DataFrame define la distribución física en el clúster del contenido del DataFrame
- El **esquema de particionamiento** define en base a qué se realiza el particionamiento (por ejemplo, en base a valores de una columna o aleatoriamente)

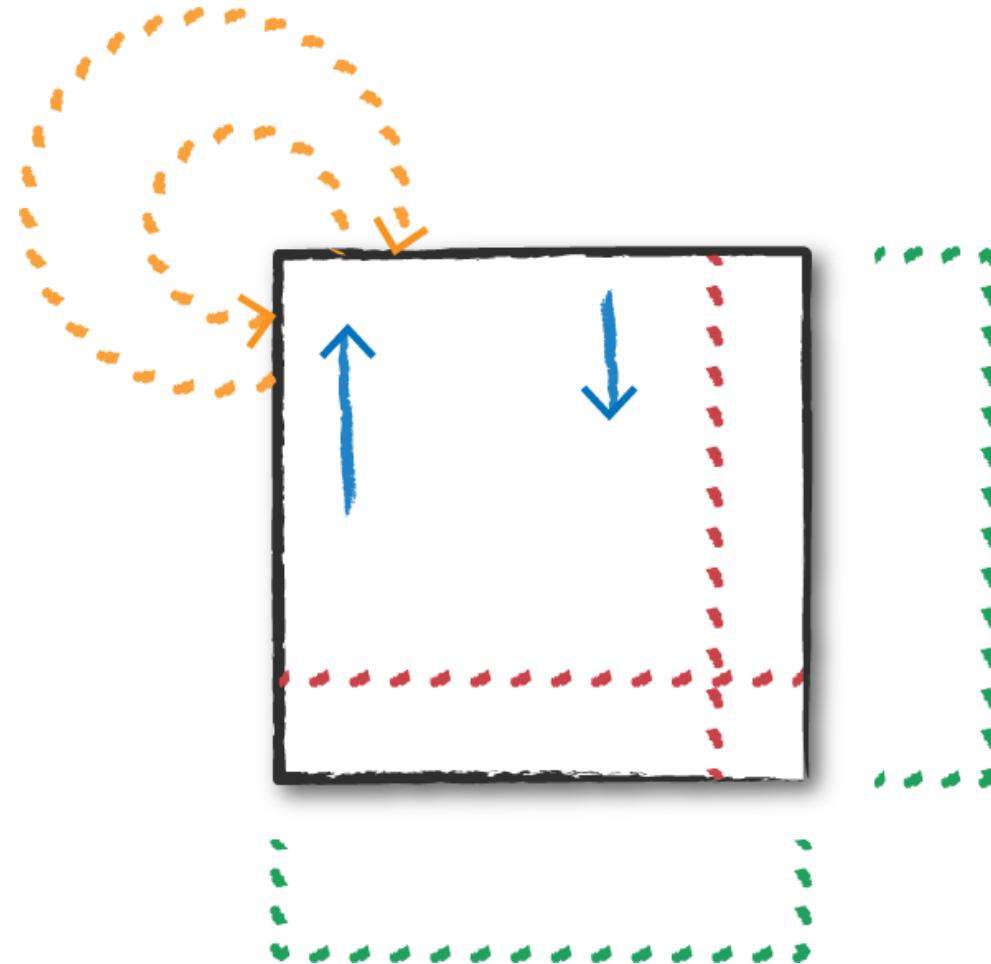
DataFrames: esquemas

- Podemos forzar el esquema de un DataFrame o podemos dejar que Spark lo infiera de los datos de origen (*schema-on-read*)
- La inferencia del esquema es muy cómoda para análisis ad-hoc, pero en cargas ETL de producción es preferible definir el esquema manualmente

DataFrames: transformaciones

- Son las manipulaciones que haremos sobre un DataFrame para obtener el resultado deseado
- Varios tipos:
 - Añadir filas o columnas
 - Eliminar filas o columnas
 - Transformar una fila en una columna
 - Cambiar el orden de las filas en base a valores de columnas

DataFrames: transformaciones



- Remove columns or rows
- Transform a row into a column or a column into a row
- Add rows or columns
- Sort data by values in rows

DataFrames: transformaciones

- Son las manipulaciones que haremos sobre un DataFrame para obtener el resultado deseado
- Varios tipos:
 - Añadir filas o columnas
 - Eliminar filas o columnas
 - Transformar una fila en una columna
 - Cambiar el orden de las filas en base a valores de columnas

DataFrames: Tipos de Datos

- Esta materia la vemos en el Notebook “Tipos de Datos”

DataFrames: Agregaciones Básicas

- Esta materia la vemos en el Notebook “Agregaciones Básicas”