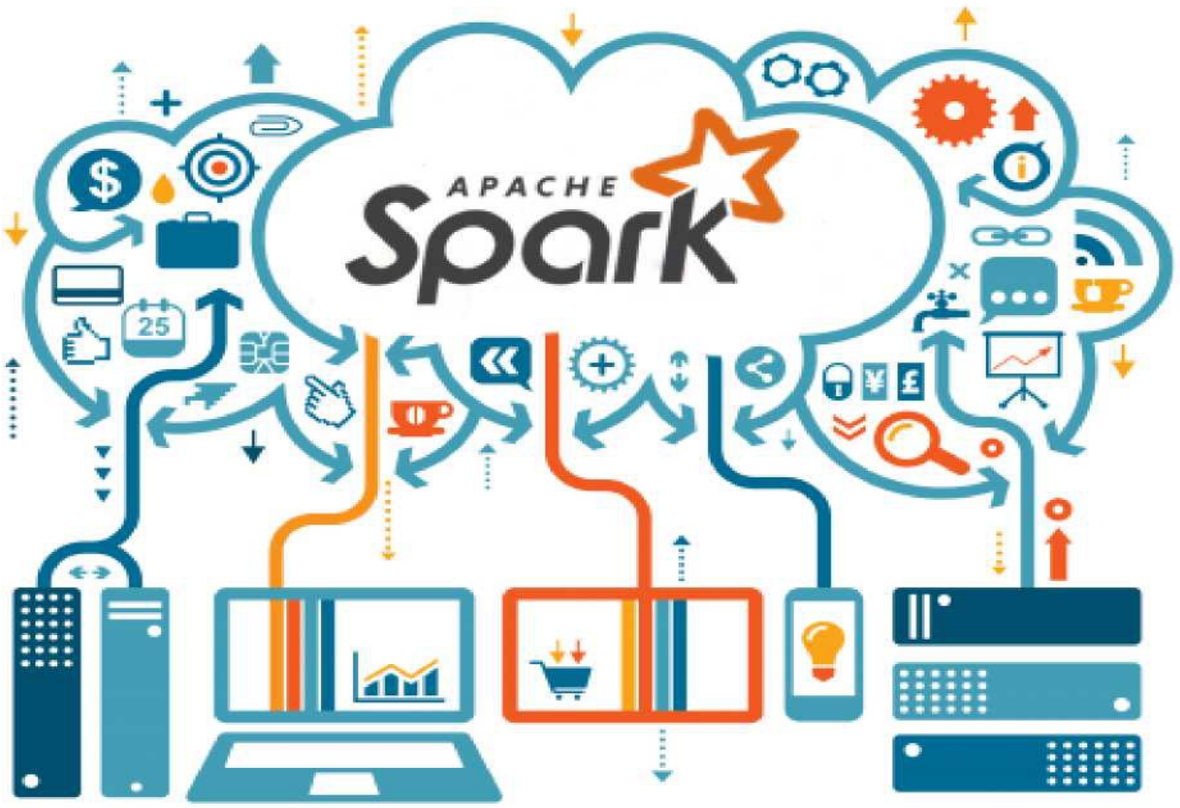
Spark SQL



Spark SQL

Los datos estructurados **son los datos que tienen un esquema, es** decir, un conjunto conocido de campos para cada registro.

Spark SQL proporciona una abstracción de conjunto de datos que simplifica el trabajo con conjuntos de datos estructurados. **El conjunto de datos es similar a las tablas de una base de datos relacional.**
Cada vez más flujo **de trabajo de Spark se está moviendo hacia Spark SQL**.

El conjunto de datos tiene un esquema natural y esto permite a Spark almacenar datos de una manera más eficaz y **puede ejecutar consultas SQL en él mediante comandos SQL reales.**

DATAFRAME

Spark SQL introduce una abstracción de datos tabulares denominada **DataFrame** desde Spark 1.3
Un DataFrame es una abstracción de datos o un lenguaje específico de dominio para trabajar con datos estructurados y semiestructurados.

DataFrames almacenan datos de una manera más eficiente que los RDD nativos, **aprovechando su esquema utiliza las capacidades inmutables, en memoria, resilientes, distribuidas y paralelas de RDD,** y aplica una estructura llamada esquema a los datos, lo que permite a Spark administrar el esquema y solo pasar datos entre nodos, de una manera mucho más eficiente que el uso de Java Serialización.

A diferencia de un RDD, **los datos se organizan en columnas con nombre, como una tabla en una base de datos relacional.**

DATASET

Un DATASET es un conjunto de datos estructurados, no necesariamente una fila, pero podría ser de un tipo determinado. Java y Spark conocerán el tipo de datos de un dataset en tiempo de compilación.

SPARKSESSION

Con el SparkSession haremos lo mismo que hacíamos con sqlContext por ejemplo obtener un Dataset

```
//obtenemos el dataset de tipo People
val ds = sparkSession.read.json("src/main/resources/people.json").
```

O por el contrario obtener un **DataFrame**

```
val df = sparkSession.read.format("fichero.csv")
.format("fichero.csv")
.option("header", "true") // Use first line of all files as header
.option("inferSchema", "true") // Automatically infer data types
.load("src/main/resources/datos.csv")
```

DATAFRAME Y DATASET

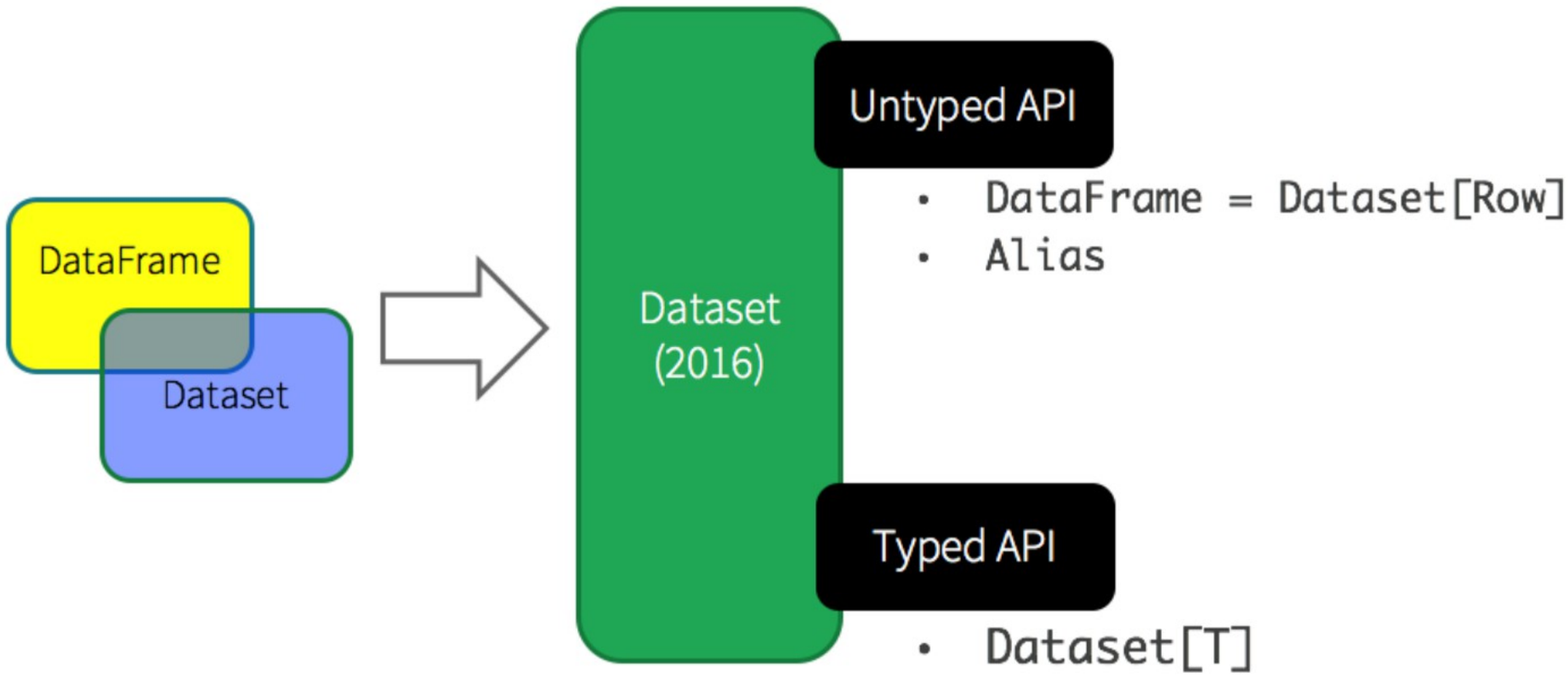
A partir de Spark 2.0, las API de DataFrame se combinan con las API de Dataset

El conjunto de datos adopta dos características de API distintas: una API fuertemente tipada y una API sin tipo.

Considera DataFrame como vista sin tipo de un conjunto de datos, que es un conjunto de datos de fila donde una fila es un objeto JVM genérico sin tipo.

Dataset, por el contrario, es una colección de objetos JVM fuertemente tipados.

DATAFRAME Y DATASET



CATALYST . Optimizador Spark

Spark SQL utiliza un **optimizador llamado Catalyst** para optimizar todas las consultas escritas tanto en **Spark SQL como en DataFrame DSL.**

- Este optimizador hace que las consultas se ejecuten mucho más rápido que sus homólogos RDD.

- El Catalyst es una biblioteca modular que se construye como un sistema basado en reglas. Cada regla del marco de trabajo se centra en la optimización específica.

Por ejemplo, una regla como **ConstantFolding** se centra en quitar la expresión constante de la consulta.

DATASET O RDD

Tenemos que tener en cuenta que:

- MLlib está en un cambio a LA API basada en conjunto de datos.

- Spark streaming también se está moviendo hacia, algo llamado streaming estructurado que se basa en gran medida en Dataset API.

CUANDO USAR DATASETS

Se necesitan semánticas enriquecidas, abstracciones de alto nivel y API específicas de dominio.

• Nuestro procesamiento requiere agregación, promedios, suma, consultas SQL y acceso en columnas en datos semi-estructurados.
• Queremos un mayor grado de seguridad de tipos en tiempo de compilación, objetos JVM escritos y el beneficio de la optimización de Catalyst.

• Se necesita la unificación y simplificación de las API en las bibliotecas de Spark.

OPERACIONES SPARK SQL

a. En caso de que tenga datos estructurados o semiestructurados con datos simples e inequívocos tipos, puede deducir un esquema mediante una reflexión.
Ejemplo →
import spark.implicits._
// for implicit conversions from Spark RDD to Dataframe
val dataFrame = rdd.toDF()

b. En el caso de querer cargar datos de un fichero:
val dataFrame = spark.read.json("example.json")

val dataFrame = spark.read.csv("example.csv")

AGREGACION

PARA PODER TRABAJAR CON LAS FUNCIONES DE AGREGADO :
import org.apache.spark.sql.functions._

Su Sintaxis :

SimpleData= seq con muchos datos
val df = simpleData.toDF("employee_name", "department", "salary")
df.select(avg("Salary")).show(false)

OPERACIONES JOIN

JoinType	Join String	Equivalent SQL Join
Inner.sql	inner	INNER JOIN
FullOuter.sql	outer, full, fullouter, full_outer	FULL OUTER JOIN
LeftOuter.sql	left, leftouter, left_outer	LEFT JOIN
RightOuter.sql	right, rightouter, right_outer	RIGHT JOIN

```
empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"inner")
    .show(false)
```