MBIT School
Madrid Business Intelligence Technology

Apache Spark SQL

Diego J. Bodas Sagi

Índice

¿Por qué?

MBIT School
Madrid Business Intelligence Technology

- Porque nos nutrimos de datos que están almacenados persistentemente
- Porque SQL es un estándar extendido y fácil de usar
- Porque recurrir a Hive implica depender de Hadoop
- Porque consideramos que se puede mejorar lo que tenemos…

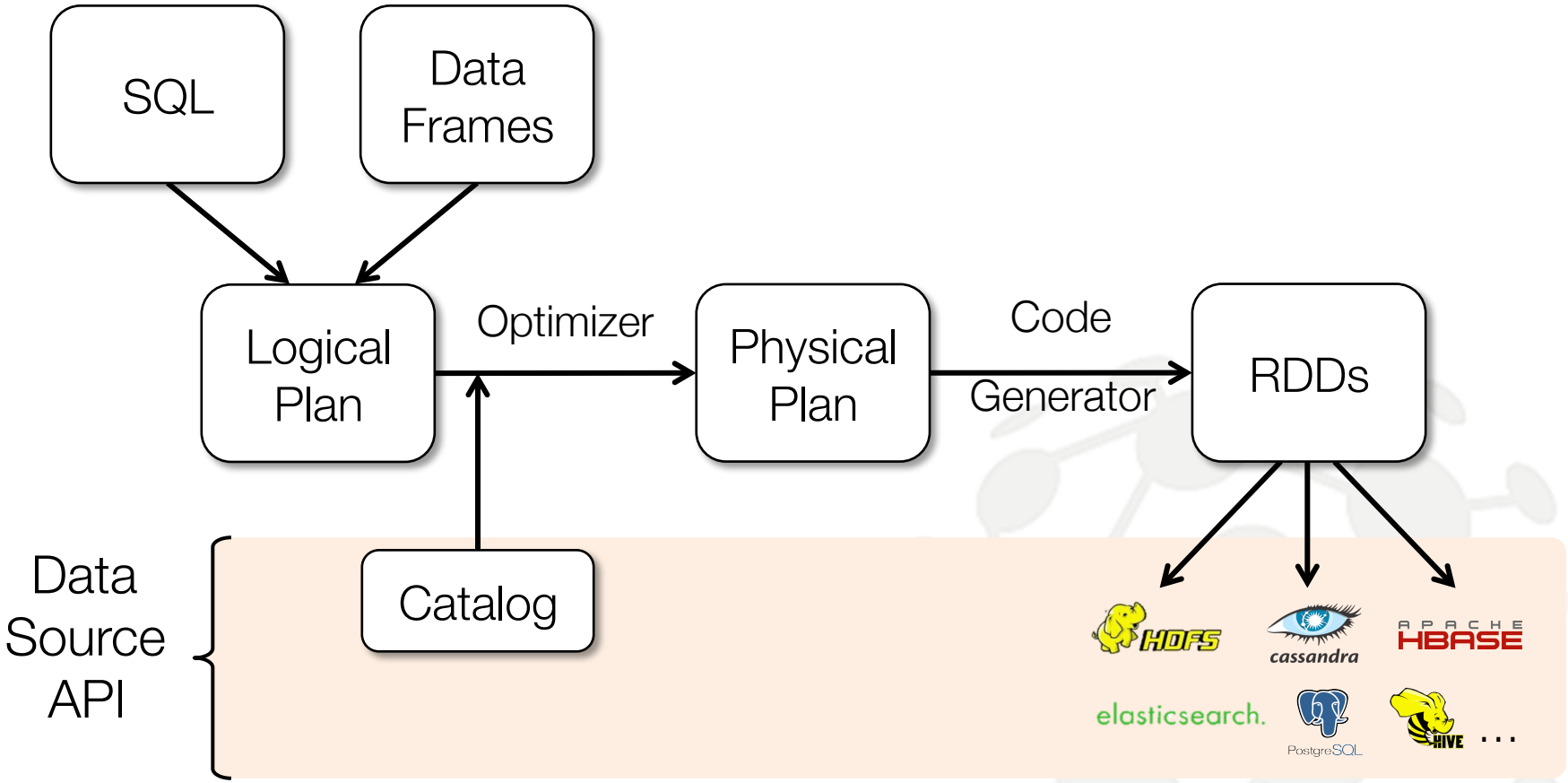
Contextos SQL

MBIT School
Madrid Business Intelligence Technology

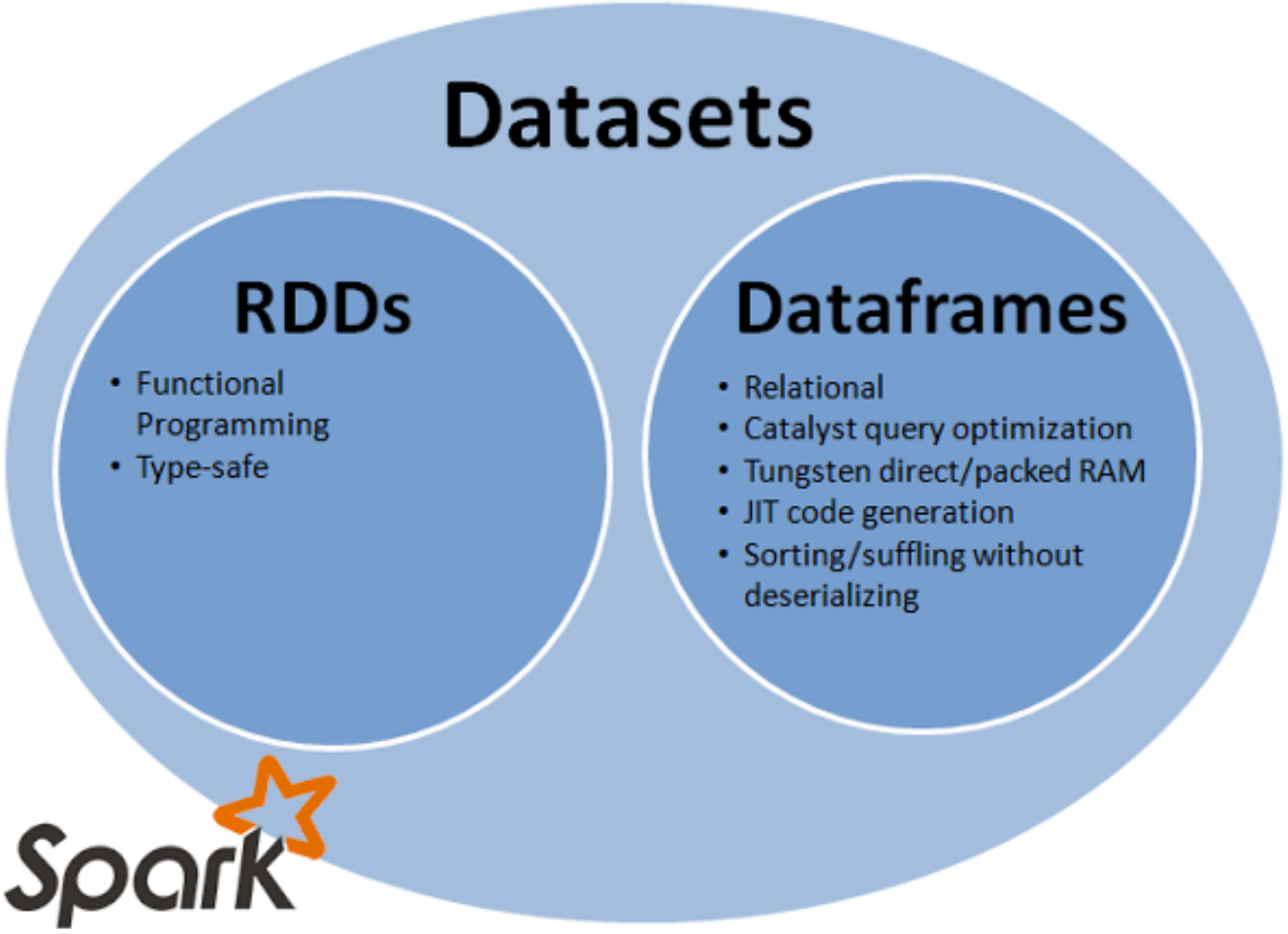
- Toda aplicación Spark depende de un contexto
- El contexto recoge toda la información necesaria para acceder al clúster y usar sus recursos
- El *master* es el encargado de gestionar el cluster
- En este caso, necesitamos un sqlContext
- Este contexto nos permite crear ***DataFrames***

Master Parameter	Description
`local`	run Spark locally with one worker thread (no parallelism)
`local[K]`	run Spark locally with K worker threads (ideally set to number of cores)
`spark://HOST:PORT`	connect to a Spark standalone cluster; PORT depends on config (7077 by default)
`mesos://HOST:PORT`	connect to a Mesos cluster; PORT depends on config (5050 by default)

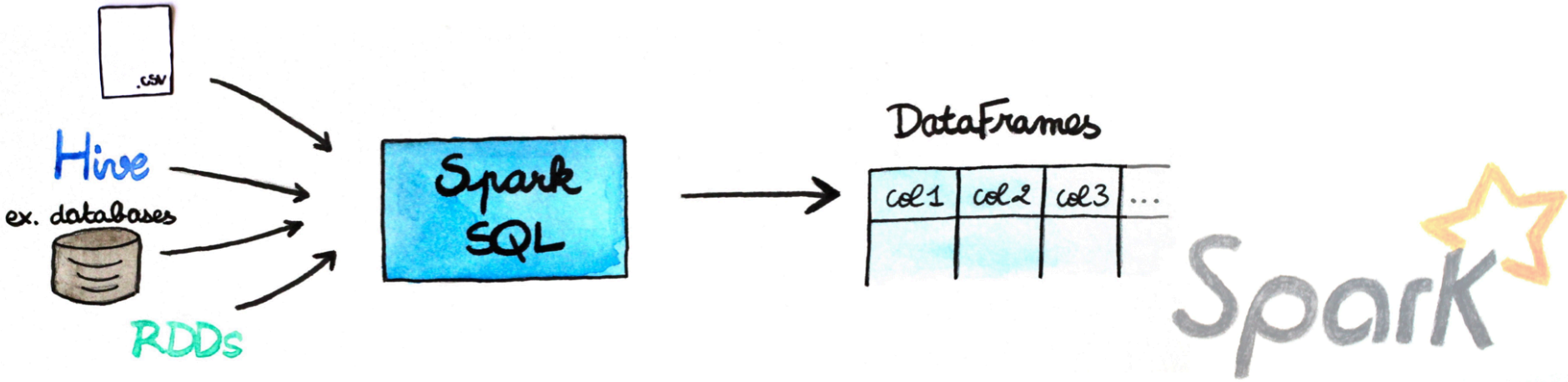
Arquitectura



Datasets


Datasets
RDDs
• Functional Programming
• Type-safe
Dataframes
• Relational
• Catalyst query optimization
• Tungsten direct/packed RAM
• JIT code generation
• Sorting/suffling without deserializing
Spark

DataFrames



Cada fila de un dataset es un objeto **Row**

DataFrames: transformations and actions

MBIT School
Madrid Business Intelligence Technology

Transformations	Actions
filter	count
select	collect
drop	show
join	take

Lazy evaluation

Transformed DF is executed when action runs on it

DataFrames: transformations and actions

show action causes **createDataFrame**, **filter**, and **select** transforms to be executed

result

Ejemplo

MBIT School
Madrid Business Intelligence Technology

```
>>> data = [('Alice', 1), ('Bob', 2)]
>>> data
[('Alice', 1), ('Bob', 2)]
>>> df = sqlContext.createDataFrame(data)
```

No computation occurs with `sqlContext.createDataFrame()`

- Spark only records how to create the DataFrame

```
[Row(_1=u'Alice', _2=1), Row(_1=u'Bob', _2=2)]
>>> sqlContext.createDataFrame(data, ['name', 'age'])
[Row(name=u'Alice', age=1), Row(name=u'Bob', age=2)]
```

Crear un DataFrame desde Pandas o R

MBIT School
Madrid Business Intelligence Technology

- No confundir DataFrames de Spark con DataFrames de Python o R

```
# Create a Spark DataFrame from Pandas
>>> spark_df = sqlContext.createDataFrame(pandas_df)
```

Leer y cargar datos en el DataFrame

MBIT School
Madrid Business Intelligence Technology

```
>>> df = sqlContext.read.text("README.txt")
>>> df.collect()
[Row(value=u'hello'), Row(value=u'this')]
```

Más datos

MBIT School
Madrid Business Intelligence Technology

- Persistencia de un DataFrame: **`cache()`**
- Nunca usar collect() (*todos*) en producción, en su lugar usar take(n)
- Lanzar una consulta es muy sencillo

```
# SQL can be run over SchemaRDDs that have been registered as a table.
results = sqlContext.sql("SELECT name FROM people")
```

- Pero primero se debe haber registrado como tabla

```
// Register the DataFrames as a table.
peopleDataFrame.registerTempTable("people")
```

Salvar y persistir

MBIT School
Madrid Business Intelligence Technology

```
# DataFrames can be saved as Parquet files, maintaining the schema information.
schemaPeople.write.parquet("people.parquet")
```

Simplifica el código

MBIT School
Madrid Business Intelligence Technology

Using RDDs

```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [int(x[1]), 1])) \
    .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
    .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
    .collect()
```

Using SQL

```
SELECT name, avg(age)
FROM people
GROUP BY name
```

Using Pig

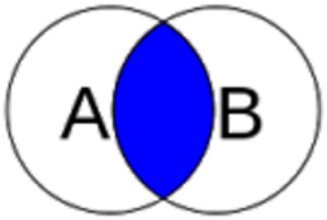
```
P = load '/people' as (name, name);
G = group P by name;
R = foreach G generate … AVG(G.age);
```

Using DataFrames

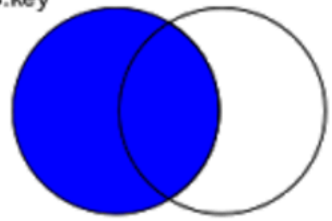
```
sqlCtx.table("people") \
    .groupBy("name") \
    .agg("name", avg("age")) \
    .collect()
```

SQL JOINS

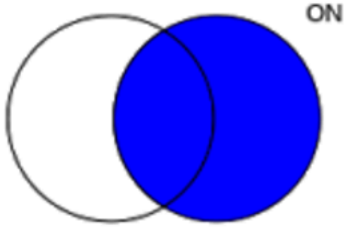
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



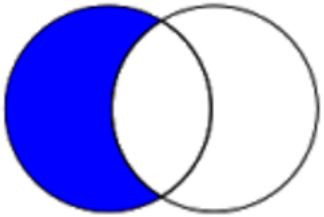
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



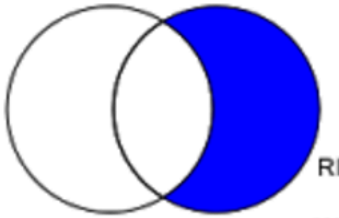
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



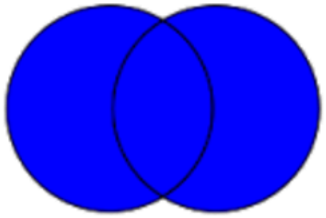
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



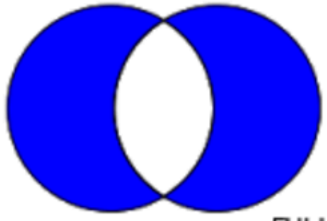
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.kev iIS NULL



Your turn

MBIT School
Madrid Business Intelligence Technology