06

第6章 Spark SQL

授课教师：[李阳源]
ywlee1212@gmail.com
主页：https://sites.google.com/view/dalguri



SL-02 提纲

6.1 Spark SQL简介
6.2 结构化数据DataFrame
6.3 DataFrame的创建和保存
6.4 DataFrame的基本操作
6.5 从RDD转换得到DataFrame
6.6 使用Spark SQL读写数据库
6.7 PySpark和pandas的整合
6.8 综合实例

SL-03 6.1 Spark SQL简介

6.1.1 Spark SQL架构
6.1.2 为什么推出Spark SQL
6.1.3 Spark SQL的特点
6.1.4 Spark SQL简单编程实例

SL-04 6.1.1 Spark SQL架构



Spark SQL在Hive兼容层面仅依赖HiveQL解析、Hive元数据，也就是说，从HQL被解析成抽象语法树（AST）起，就全部由Spark SQL接管了。Spark SQL执行计划生成和优化都由Catalyst（函数式关系查询优化框架）负责

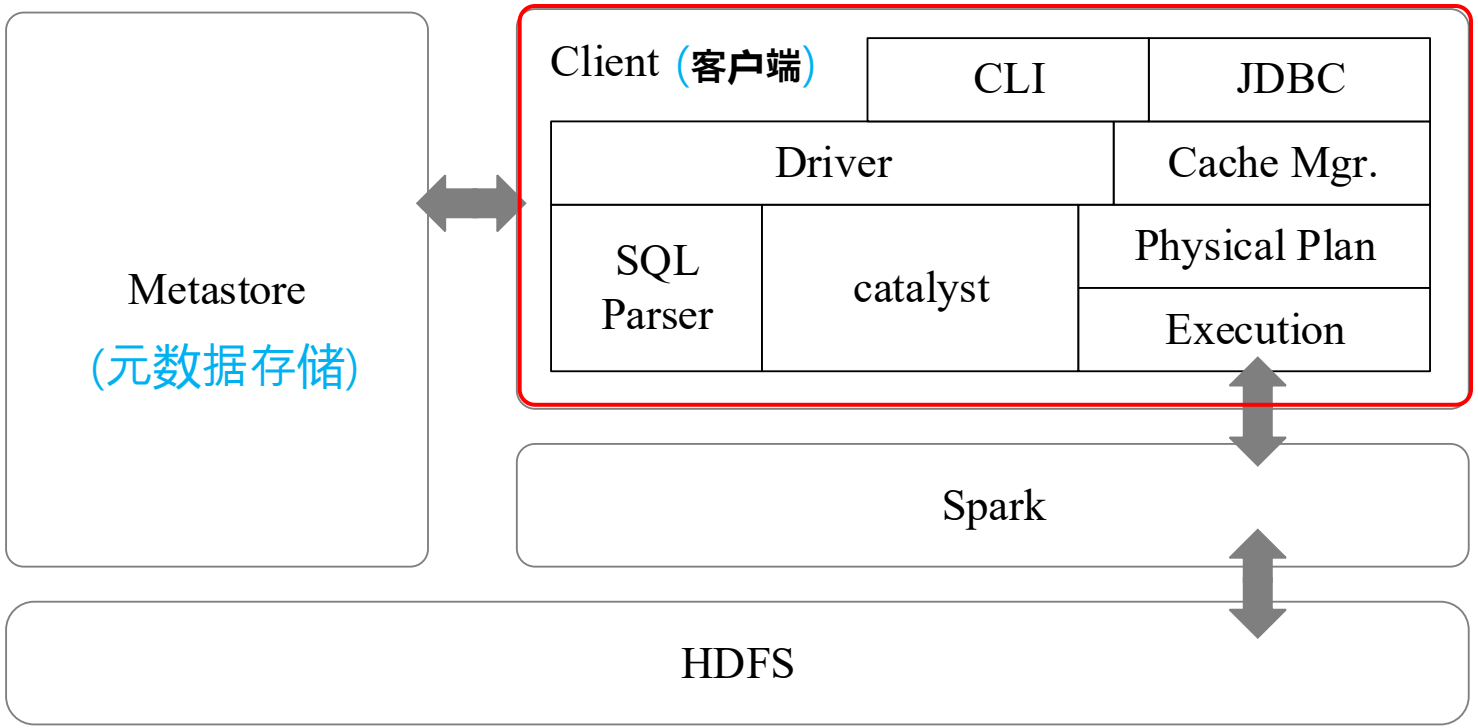


图6-3 Spark SQL架构

SL-05 6.1.1 Spark SQL架构



- Spark SQL增加了DataFrame（即带有Schema信息的RDD），使用户可以在Spark SQL中执行SQL语句，数据既可以来自RDD，也可以是Hive、HDFS、Cassandra等外部数据源，还可以是JSON格式的数据
- Spark SQL目前支持Scala、Java、Python三种语言，支持SQL-92规范

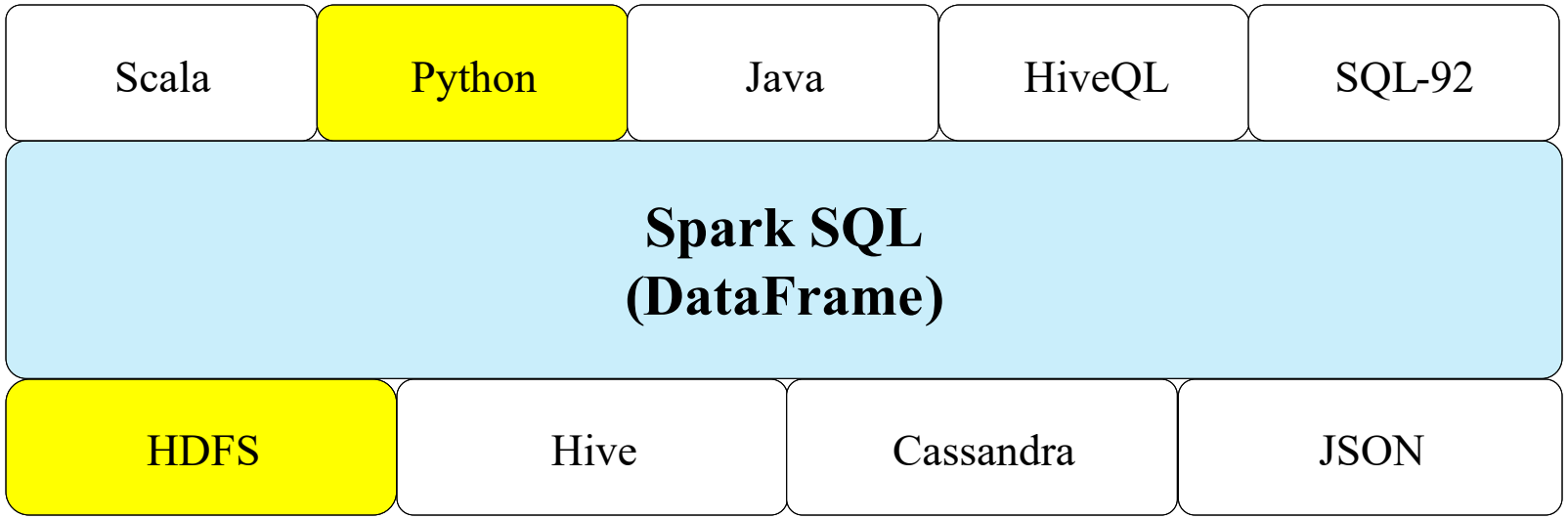


图6-4 Spark SQL支持的数据格式和编程语言

SL-06 6.1.2 为什么推出Spark SQL



Spark SQL: Relational Data Processing in Spark

Michael Armbrust[†], Reynold S. Xin[†], Cheng Lian[†], Yin Huai[†], Davies Liu[†], Joseph K. Bradley[†], Xiangrui Meng[†], Tomer Kaftan[‡], Michael J. Franklin[†‡], Ali Ghodsi[†], Matei Zaharia[†*]

[†]Databricks Inc. [*]MIT CSAIL [‡]AMPLab, UC Berkeley

While the popularity of relational systems shows that users often prefer writing declarative queries, the relational approach is insufficient for many big data applications. First, users want to perform ETL to and from various data sources that might be semi- or unstructured, requiring custom code. Second, users want to perform advanced analytics, such as machine learning and graph processing, that are challenging to express in relational systems. In practice, we have observed that most data pipelines would ideally be expressed with a combination of both relational queries and complex procedural algorithms. Unfortunately, these two classes of systems—relational and procedural—have until now remained largely disjoint, forcing users to choose one paradigm or the other.

SL-07 6.1.2 为什么推出Spark SQL



Spark SQL: Relational Data Processing in Spark

Michael Armbrust[†], Reynold S. Xin[†], Cheng Lian[†], Yin Huai[†], Davies Liu[†], Joseph K. Bradley[†], Xiangrui Meng[†], Tomer Kaftan[‡], Michael J. Franklin[†‡], Ali Ghodsi[†], Matei Zaharia[†*]

[†]Databricks Inc. [*]MIT CSAIL [‡]AMPLab, UC Berkeley

Spark SQL bridges the gap between the two models through two contributions. First, Spark SQL provides a *DataFrame API* that can perform relational operations on both external data sources and Spark's built-in distributed collections. This API is similar to the widely used data frame concept in R [32], but evaluates operations lazily so that it can perform relational optimizations. Second, to support the wide range of data sources and algorithms in big data, Spark SQL introduces a novel extensible optimizer called *Catalyst*. Catalyst makes it easy to add data sources, optimization rules, and data types for domains such as machine learning.

SL-08 6.1.2 为什么推出Spark SQL



- 关系数据库在大数据时代已经不能满足要求
  - ✓ 首先，用户需要从不同数据源执行各种操作，包括结构化、半结构化和非结构化数据
  - ✓ 其次，用户需要执行高级分析，比如机器学习和图像处理
- 在实际大数据应用中，经常需要融合关系查询和复杂分析算法（比如机器学习或图像处理），但是，缺少这样的系统

- Spark SQL填补了这个鸿沟：
  - ✓ 首先，可以提供DataFrame API，可以对内部和外部各种数据源执行各种关系型操作
  - ✓ 其次，可以支持大数据中的大量数据源和数据分析算法
  - ✓ Spark SQL可以融合：传统关系数据库的结构化数据管理能力和机器学习算法的数据处理能力

SL-09 6.1.3 Spark SQL的特点



Spark SQL的特点如下：

（1）容易整合（集成）。Spark SQL可以将SQL查询和Spark程序无缝集成，允许我们使用SQL或熟悉的DataFrame API在Spark程序中查询结构化数据。

（2）统一的数据访问方式。Spark SQL可以以相同方式连接到任何数据源，DataFrame和SQL提供了访问各种数据源的方法，包括Hive、Avro、Parquet、ORC、JSON和JDBC。

（3）兼容Hive。Spark SQL支持HiveQL语法以及Hive SerDes和UDF（用户自定义函数），允许我们访问现有的Hive仓库。

（4）标准的数据库连接。Spark SQL支持JDBC或ODBC连接。

SL-10 6.1.4 Spark SQL简单编程实例



- 在RDD编程中，使用的是SparkContext接口，在Spark SQL编程中，需要使用SparkSession接口。
- 从Spark2.0以上版本开始，Spark使用全新的SparkSession接口替代Spark1.6中的SQLContext及HiveContext接口，来实现其对数据加载、转换、处理等功能。
- SparkSession实现了SQLContext及HiveContext所有功能。
- 此外，SparkSession也封装了SparkContext、SparkConf和StreamingContext等。也就是说，在Spark1.x中，需要创建多种上下文对象（比如，创建SparkContext对象用于RDD编程，创建SQLContext对象用于SQL编程），这会让代码显得很烦琐；
- 而在Spark2.x和Spark3.x中，应用只需要为每个JVM创建一个SparkSession对象，然后就可以用其执行各种Spark操作。

SL-11 6.1.4 Spark SQL简单编程实例



- 需要注意的是，虽然SparkSession对象已经包含了其他所有的上下文对象（SparkContext、SparkConf和StreamingContext等），可以为所有的Spark应用程序提供统一的入口，但你仍然可以访问那些上下文对象及其方法，比如，可以继续使用SparkContext对象作为RDD编程的入口，因此，使用SparkContext或SQLContext的基于1.x版本的旧代码，仍然可以正常运行在2.x和3.x上。

- SparkSession支持从不同的数据源加载数据，以及把数据转换成DataFrame，并且支持把DataFrame转换成SQLContext自身的表，然后使用SQL语句来操作数据。SparkSession亦提供了HiveQL以及其他依赖于Hive的功能的支持。

SL-12 6.1.4 Spark SQL简单编程实例



可以通过如下语句创建一个SparkSession对象：

```
from pyspark.sql import SparkSession
spark=SparkSession ₩
            .builder ₩
            .master("local[*]") ₩
            .appName("SparkSessionExample") ₩
            .getOrCreate()
```

实际上，在启动进入spark-shell以后，spark-shell就默认提供了一个SparkContext对象（名称为**sc**）和一个SparkSession对象（名称为spark），因此，也可以不用自己声明一个SparkSession对象，而是直接使用spark-shell提供的SparkSession对象，即**spark**。

SL-13 6.1.4 Spark SQL简单编程实例



```
# coding:utf8
from pyspark.sql import SparkSession
if __name__ == '__main__':
    spark = SparkSession₩
        .builder₩
        .master("local[*]")₩
        .appName("Simple Application")₩
        .getOrCreate()
    logFile = "file:///usr/local/spark/README.md"
    logDF = spark.read.text(logFile)
    numAs = logDF.filter(logDF["value"].contains("a")).count()
    numBs = logDF.filter(logDF["value"].contains("b")).count()
    print('Lines with a: %s, Lines with b: %s' % (numAs, numBs))
    spark.stop()
```

6.1.4 Spark SQL简单编程实例



File Edit View Navigate Code Refactor Run Tools VCS Window Help

ch05SparkSQL 〉 p136_SparkSQLExample.py

p136_SparkSQLExample

Project

ch05SparkSQL ~/PycharmProjects/ch05S
main.py
p136_SparkSQLExample.py
External Libraries
Scratches and Consoles

main.py × p136_SparkSQLExample.py ×

```
1  from pyspark.sql import SparkSession
2
3  if __name__ == '__main__':
4      spark = SparkSession \
5          .builder \
6          .master("local[*]") \
7          .appName("SparkSessionExample") \
8          .getOrCreate()
9      logFile = 'file:///usr/local/spark/README.md'
10     logDF = spark.read.text(logFile)
11     numAs = logDF.filter(logDF['value'].contains('a')).count()
12     numBs = logDF.filter(logDF['value'].contains('b')).count()
13     print('Lines with a: %s, Lines with b: %s' %(numAs, numBs))
14     spark.stop()
15
```

if __name__ == '__main__'

Run: p136_SparkSQLExample ×

```
/home/hadoop/anaconda3/envs/nanyang/bin/python /home/hadoop/PycharmProjects/ch05SparkSQL/p136_S
Lines with a: 72, Lines with b: 39

Process finished with exit code 0
```

Version Control | Run | TODO | Problems | Terminal | Python Packages | Python Console | Services

5:1 LF UTF-8 4 spaces Python 3.8 (nanyang)