SL-01 6.3 DataFrame的创建和保存

6.3.1 Parquet
6.3.2 JSON
6.3.3 CSV
6.3.4 文本文件
6.3.5 序列集合

SL-02 6.3.1 Parquet



1.从Parquet文件创建DataFrame

Parquet是Spark的默认数据源，很多大数据处理框架和平台都支持Parquet格式，它是一种开源的列式存储文件格式，提供多种I/O优化措施（比如压缩，以节省存储空间，支持快速访问数据列）。
存储Parquet文件的目录中包含了_SUCCESS文件和很多像part-XXXXX这样的压缩文件。要把Parquet文件读入DataFrame，需要指定格式和路径，具体实例如下：

```
>>> df.show()
+------+--------------+----------------+
|  name|favorite_color|favorite_numbers|
+------+--------------+----------------+
|Alyssa|          null|  [3, 9, 15, 20]|
|   Ben|           red|              []|
+------+--------------+----------------+
```



```
Terminal File Edit View Search Terminal Help
(base) hadoop@hadoop01:/usr/local/spark/examples/src/main/resources$ /usr/local/
spark/bin/pyspark
Python 3.8.20 (default, Oct  3 2024, 15:24:27)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
Welcome to

      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.4.0
      /_/

Using Python version 3.8.20 (default, Oct  3 2024 15:24:27)
Spark context Web UI available at http://192.168.254.129:4040
Spark context available as 'sc' (master = local[*], app id = local-1751944987537
).
SparkSession available as 'spark'.
>>> filePath="file:///usr/local/spark/examples/src/main/resources/users.parquet"
>>> df=spark.read.format("parquet").load(filePath)
>>> df.show()
+------+--------------+----------------+
|  name|favorite_color|favorite_numbers|
+------+--------------+----------------+
|Alyssa|          null|  [3, 9, 15, 20]|
|   Ben|           red|              []|
+------+--------------+----------------+
```

6.3.1 Parquet



或者也可以使用如下方式读取Parquet文件生成DataFrame：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/users.parquet"
>>> df = spark.read.parquet(filePath)
>>> df.show()
```

6.3.1 Parquet



或者也可以使用如下方式读取Parquet文件生成DataFrame：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/users.parquet"
>>> df = spark.read.parquet(filePath)
>>> df.show()
```

6.3.1 Parquet



2.将DataFrame保存为Parquet文件

将DataFrame保存为Parquet文件的具体方法如下（在上面的代码基础上继续执行下面代码）：

```
>>> df.write.format("parquet").mode("overwrite").option("compression","snappy").
... save("file:///home/hadoop/otherusers")
```

上面代码执行以后，在本地文件系统中的“/home/hadoop/”目录下会生成一个名称为“otherusers”的子目录，该目录下包含两个文件，即_SUCCESS文件和像part-00000-XXXX.snappy.parquet这样的文件，后者是使用snappy压缩算法得到的压缩文件。如果要再次读取文件生成DataFrame，load()中可以直接使用目录“file:///home/hadoop/otherusers”，也可以使用文件“file:///home/hadoop/otherusers/part-00000-XXXX.snappy.parquet”。

或者也可以使用如下方式把DataFrame保存为Parquet文件：

```
>>> df.write.parquet("file:///home/hadoop/otherusers")
```

6.3.2 JSON



1.从JSON文件创建DataFrame

JSON（JavaScript Object Notation）是一种常见的数据格式，与XML相比，JSON的可读性更强，更容易解析。JSON有两种表示格式，即单行模式和多行模式，这两种模式Spark都支持。
从JSON文件创建DataFrame的具体方法如下：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.json"
>>> df = spark.read.format("json").load(filePath)
>>> df.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

6.3.2 JSON



或者也可以使用如下方式读取JSON文件生成DataFrame：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.json"
>>> df = spark.read.json(filePath)
```

6.3.2 JSON



2.将DataFrame保存为JSON文件

将DataFrame保存为JSON文件的具体方法如下（在上面的代码基础上继续执行下面代码）：

```
>>> df.write.format("json").mode("overwrite").save("file:///home/hadoop/otherpeople")
```

上面代码执行以后，在本地文件系统中的“/home/hadoop/”目录下会生成一个名称为“otherpeople”的子目录，该目录下包含两个文件，即_SUCCESS文件和像part-00000-XXXX.json这样的文件。如果要再次读取文件生成DataFrame，load()中可以直接使用目录“file:///home/hadoop/otherpeople”，也可以使用文件“file:///home/hadoop/otherpeople/part-00000-XXXX.json”。

或者也可以采用如下方式把DataFrame保存成JSON文件：

```
>>> df.write.json("file:///home/hadoop/otherpeople")
```

6.3.3 CSV



1.从CSV文件创建DataFrame

CSV是一种将所有的数据字段用逗号隔开的文本文件格式，在这些用逗号隔开的字段中，每行表示一条记录。CSV文件已经和普通的文本文件一样被广泛使用。

从CSV文件创建DataFrame的具体方法如下：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.csv"
>>> schema = "name STRING,age INT,job STRING"
>>> df = spark.read.format("csv").schema(schema).option("header","true").₩
... option("sep",";").load(filePath)
>>> df.show()
+-----+---+---------+
| name|age|      job|
+-----+---+---------+
|Jorge| 30|Developer|
|  Bob| 32|Developer|
+-----+---+---------+
```

6.3.3 CSV



或者也可以使用如下方式读取CSV文件生成DataFrame：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.csv"
>>>  schema = "name STRING,age INT,job STRING"
>>>  df = spark.read.schema(schema).option("header","true").₩
...  option("sep",";").csv(filePath)
```

6.3.3 CSV



2.将DataFrame保存为CSV文件

将DataFrame保存为CSV文件的具体方法如下（在上面的代码基础上继续执行下面代码）：

```
>>> df.write.format("csv").mode("overwrite").save("file:///home/hadoop/anotherpeople")
```

或者也可以使用如下方式把DataFrame保存成CSV文件：

```
>>> df.write.csv("file:///home/hadoop/anotherpeople")
```

6.3.4 文本文件



从一个文本文件创建DataFrame的方法如下：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.txt"
>>> df = spark.read.format("text").load(filePath)
>>> df.show()
+-----------+
|      value|
+-----------+
|Michael, 29|
|   Andy, 30|
| Justin, 19|
+-----------+
```

或者也可以使用如下方式读取文本文件生成DataFrame：

```
>>> filePath = "file:///usr/local/spark/examples/src/main/resources/people.txt"
>>> df = spark.read.text(filePath)
```

6.3.4 文本文件



如果要把一个DataFrame保存成文本文件，则需要使用如下语句格式（在上面代码的基础上继续执行）：

```
>>> df.write.text("file:///home/hadoop/newpeople")
```

或者也可以使用如下语句格式把一个DataFrame保存成文本文件：

```
>>> df.write.format("text").save("file:///home/hadoop/newpeople")
```

6.3.5 序列集合



Spark允许通过序列集合来创建DataFrame，下面是一个具体实例：

```
>>> df = spark.createDataFrame([￦
    ("Xiaomei","Female","21"),￦
    ("Xiaoming","Male","22"),￦
    ("Xiaoxue","Female","23")]).￦
    toDF("name","sex","age")
>>> df.show()
+--------+------+---+
|    name|   sex|age|
+--------+------+---+
| Xiaomei|Female| 21|
|Xiaoming|  Male| 22|
| Xiaoxue|Female| 23|
+--------+------+---+
```

6.3.5 序列集合



在创建DataFrame时，还可以对每个列的数据类型进行定义，具体实例如下：

```
>>> from pyspark.sql import SparkSession,Row
>>> from pyspark.sql.types import IntegerType, StringType, StructField, StructTy
pe
>>> schema = StructType([StructField("name", StringType(), True),₩
                         StructField("age", IntegerType(), True),₩
                         StructField("sex", StringType(), True)])
>>> data = []
>>> data.append(Row("Xiaomei", 21, "Female"))
>>> data.append(Row("Xiaoming", 22, "Male"))
>>> data.append(Row("Xiaoxue", 23, "Female"))
>>> df = spark.createDataFrame(data, schema)
>>> df.show()
```