Практика SPARK.

.

Neoflex
IT-решения для бизнеса

DataFrame

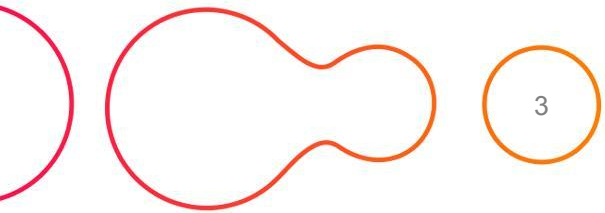
DATAFRAME

DataFrame – распределенный набор данных сгруппированный по именованным столбцам.

DataFrame – объект SPARK, позволяющий работать с набором данных в табличном представлении.

Полезные ссылки:

1) pyspark.sql module — PySpark 2.4.8 documentation (apache.org)
2) https://sparkbyexamples.com/

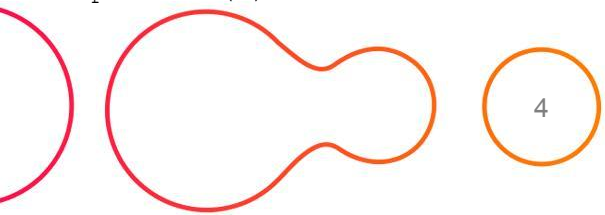


Neoflex
IT-решения для бизнеса

СОЗДАНИЕ

```
emp = [(1,"Smith",-1,"2018","10","M",3000), \
     (2,"Rose",1,"2010","20","M",4000), \
     (3,"Williams",1,"2010","10","M",1000), \
     (4,"Jones",2,"2005","10","F",2000), \
     (5,"Brown",2,"2010","40","",-1), \
       (6,"Brown",2,"2010","50","",-1) \
  ]
empColumns = ["emp_id","name","superior_emp_id","year_joined", \
       "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show(3)
```



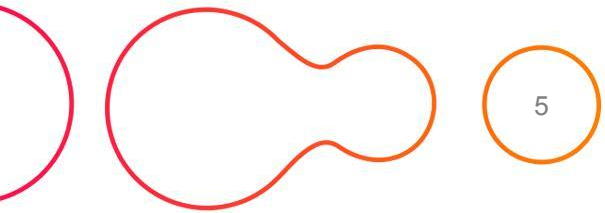
Neoflex
IT-решения для бизнеса

СОЗДАНИЕ

```
df_customers=spark.read.csv('/tmp/spark_practics/data/customers', header=True, sep=',')
df_billings=spark.read.csv('/tmp/spark_practics/data/billings', header=True, sep=',')

df_customers.printSchema()
df_customers.show(3)

df_billings.printSchema()
df_billings.show(3)
```



ОБРАЩЕНИЕ К СТОЛБЦАМ

1) dataframe_name.column_name

```
df_customers = df_customers.withColumnRenamed("customer_id", "id")

df_billings.filter(df_billings.amount < 5000).join(df_customers, df_customers.id ==
df_billings.customer_id) \

.groupBy(df_customers.country).agg({"amount": "avg", "balance": "max", "customer_id":
"count"}).show(3)
```

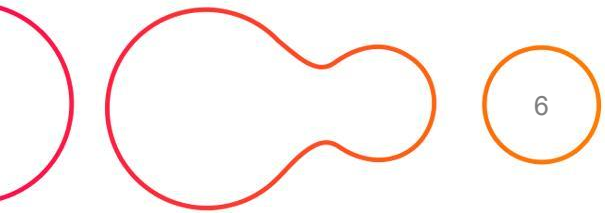
2) “column_name”

```
df_billings.filter("amount < 5000").where("balance > 0").show(3)
```

3) col(“column_name”)

```
from pyspark.sql import functions as f

df_billings.filter(f.col("amount") < 5000).where(f.col("balance") > 0).show(3)
```



Neoflex
IT-решения для бизнеса

ОПЕРАЦИИ С DATAFRAME

1) Выбор столбцов

```
df_billings.select('customer_id', 'balance').show(3)
```

2) Переименование столбцов

```
df_billings.withColumnRenamed('customer_id', 'id').show(3)
```

3) Добавление новых столбцов

```
df_billings.withColumn("cat", f.col('balance')+500).show(3)
```

4) Агрегатные ф-ии

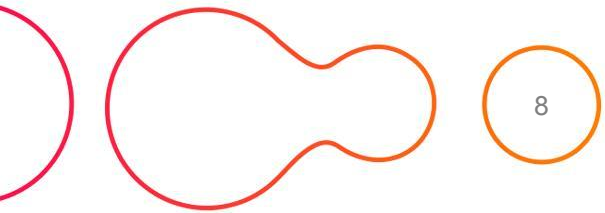
```
df_customers = df_customers.withColumnRenamed("customer_id", "id")
df_billings.filter(df_billings.amount < 5000).join(df_customers, df_customers.id == df_billings.customer_id) \
.groupBy(df_customers.country)\
.agg(f.avg("amount").alias("avg_am"),\
          f.max("balance").alias("max_bal"),\
#fergergetgrtg
          f.count("customer_id").alias("c"))\
.where(f.col("c")>7000).show(3)
```

Neoflex
IT-решения для бизнеса

SPARK SQL

```
df_customers.createOrReplaceTempView('cust')

df = spark.sql('select country, min(id), count(id) from cust group by country having count(id)<1450')
df.show(3)
```



Neoflex
IT-решения для бизнеса

Neoflex
IT-решения для бизнеса

Создание DataFrame из данных разных форматов

CSV

```
DF=spark.read.csv('hdfs://cluster/user/hdfs/test/example.csv', schema=None, sep = '\t',
encoding='UTF-8', escape='\', comment='--', header='True', inferSchema='False',
ignoreLeadingWhiteSpace='True', ignoreTrailingWhiteSpace='True', nullValue='Null',
nanValue='Nan', positiveInf='-oo', negativeInf='+oo', dateFormat='yyyy-MM-dd HH:mm:ss',
timestampFormat='yyyy-MM-dd'T'HH:mm:ss.SSSZZ', maxColumns= 100, maxCharsPerColumn=1000,
maxMalformedLogPerPartition=1, mode='PERMISSIVE')
```

CSV

Параметры:

path – строка или список строк, указывающие, где лежат файлы.

schema – опционально, схема которая описывает данные в файле.

esep – символ, который описывает разделитель в файле. По умолчанию, ‘,’.

encoding – указывает из какой кодировки надо декодировать файл. По умолчанию, ‘UTF-8’.

escape – символ, используемый для экранирования кавычек внутри уже заключенного в кавычки значения. По умолчанию '\'.

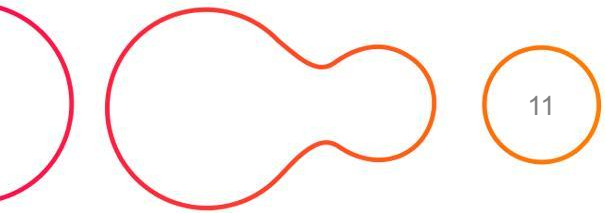
comment – символ, указывающий начало строки комментария, которые не требуется загружать, по умолчанию выключено.

header – логическое значение, указывающее использовать первую строку в качестве названий столбцов или нет. По умолчанию ‘False’.

inferSchema – логическое значение, указывающее автоматическое получение схемы из данных, требует дополнительного сканирования всего файла. По умолчанию, ‘False’.

ignoreLeadingWhiteSpace – определяет пропускать или нет пробелы перед значениями. По умолчанию, ‘False’.

ignoreTrailingWhiteSpace – определяет пропускать или нет пробелы после значений. По умолчанию, ‘False’.



CSV

Параметры:

nullValue – устанавливает строку, которая обозначает значение null. По умолчанию пустая строка.

nanValue – устанавливает строку, которая обозначает нечисловое значение. По умолчанию ‘NaN’.

positiveInf – устанавливает строку, которая обозначает плюс бесконечность. По умолчанию ‘Inf’.

negativeInf – устанавливает строку, которая обозначает минус бесконечность. По умолчанию ‘Inf’.

dateFormat – устанавливает формат даты в соответствии форматам java.text.SimpleDateFormat, поля со значениями соответствующие этому формату загружаются с типом дата. По умолчанию, ‘yyyy-MM-dd’.

timestampFormat – устанавливает формат timestamp в соответствии форматам java.text.SimpleDateFormat, поля со значениями соответствующие этому формату загружаются с типом timestamp . По умолчанию, ‘yyyy-MM-dd'T'HH:mm:ss.SSSZZ’.

maxColumns – указывает максимальное возможное количество столбцов, которое может иметь файл, по умолчанию 20480.

maxCharsPerColumn – указывает максимальное возможное количество символов у столбцов, которое может иметь файл, по умолчанию , -1 (неограничено).

maxMalformedLogPerPartition – указывает максимальное количество не соответствующие схеме строк, которые будут записаны в лог, остальные проигнорированы.

CSV

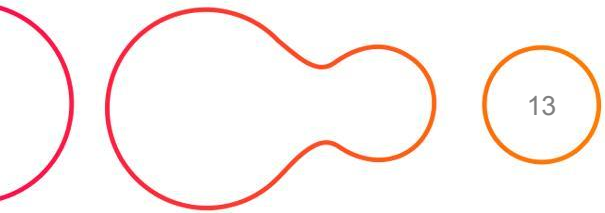
Параметры:

mode – указывает реакцию на ошибки при парсинге строк. По умолчанию, ‘PERMISSIVE’.

PERMISSIVE : устанавливает значения в null в строках, которые не смог распарсить. В случае использования пользовательской схемы, устанавливает в null все значения, которые не соответствуют схеме.

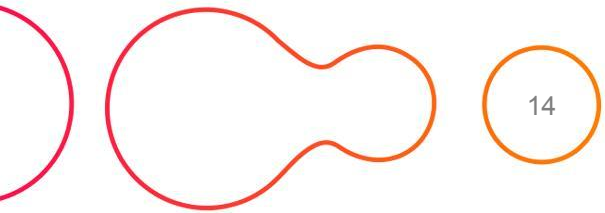
DROPMALFORMED : игнорирует строки, которые не смог распарсить.

FAILFAST : вызывает исключение.



JSON

```
DF=spark.read.json('//home/user/test/example.json', schema=None,
prefersDecimal='True', allowComments='False', allowUnquotedFieldNames='True',
allowSingleQuotes='False', allowNumericLeadingZero='True',
allowBackslashEscapingAnyCharacter='True', mode=None, dateFormat=None, timestampFormat=None)
```



JSON

Параметры:

path – строка или список строк, указывающие, где лежат файлы.

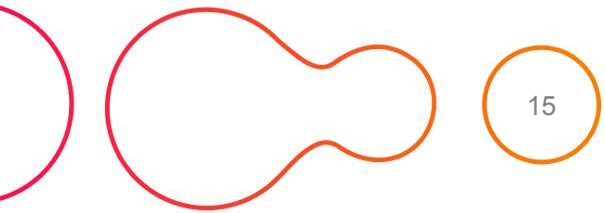
schema – опционально, схема которая описывает данные в файле.

mode – указывает реакцию на ошибки при парсинге строк. По умолчанию, ‘PERMISSIVE’.

> **PERMISSIVE** : устанавливает значения в null в строках, которые не смог распарсить. В случае использования пользовательской схемы, устанавливает в null все значения, которые не соответствуют схеме.
>
> **DROPMALFORMED** : игнорирует строки, которые не смог распарсить.
>
> **FAILFAST** : вызывает исключение.

dateFormat – устанавливает формат даты в соответствии форматам java.text.SimpleDateFormat, поля со значениями соответствующие этому формату загружаются с типом дата. По умолчанию, ‘yyyy-MM-dd’.

timestampFormat – устанавливает формат timestamp в соответствии форматам java.text.SimpleDateFormat, поля со значениями соответствующие этому формату загружаются с типом timestamp . По умолчанию, ‘yyyy-MM-dd'T'HH:mm:ss.SSSZZ’.



JSON

Параметры:

primitivesAsString – устанавливает загружать все примитивы, как строки. По умолчанию, false.

prefersDecimal – устанавливает загружать все числа с дробной частью, как decimal (дробная часть фиксированная), иначе, как вещественное число (число с плавающей точкой). По умолчанию, false.

allowComments – устанавливает игнорировать комментарии в стиле Java/C++. По умолчанию, false.

allowUnquotedFieldNames – разрешает наименования полей без кавычек. По умолчанию, false.

allowSingleQuotes – разрешает апострофы в дополнении к двойным кавычкам. По умолчанию, false.

allowNumericLeadingZero – разрешает лидирующие нули в числах (например, 00012). По умолчанию, false.

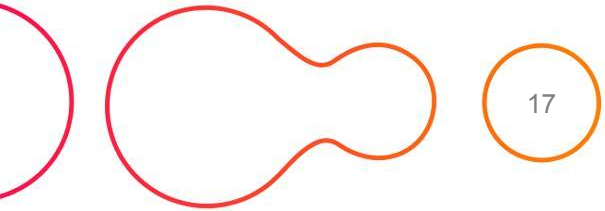
allowBackslashEscapingAnyCharacter – разрешает использование обратной черты в качестве символа экранирования. По умолчанию, false.

PARQUET

```
DF=spark.read.parquet('//home/user/test/exampleparquet')
```

Параметры:

path – строка или список строк, указывающие, где лежат данные.

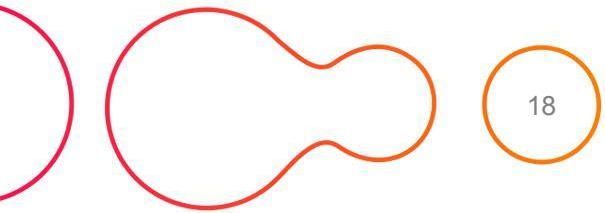


ORC

```
DF=spark.read.orc('//home/user/test/exampleorc')
```

Параметры:

path – строка, указывающая, где лежат данные.

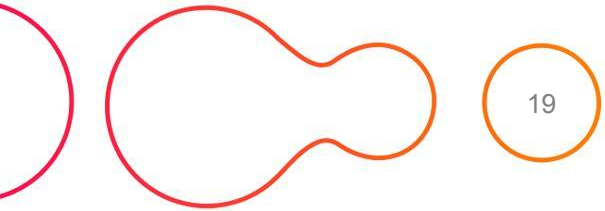


TEXT

```
DF=spark.read.text('//home/user/test/example.txt')
```

Параметры:

path – строка или список строк, указывающие, где лежат данные.



HIVE

```
DF=spark.read.table(tableName = 'test.table')
```

или

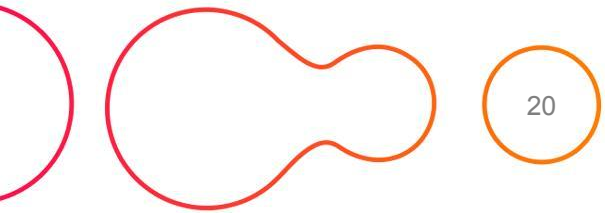
```
DF=spark.sql(sqlQuery = 'select * from test.tavble')
```

Параметры:

tableName – строка название таблицы

или

sqlQuery – sql запрос



JDBC

```
DF=spark.read.jdbc('jdbc:postgresql://localhost:5432/db_test', 'table_test',
column='date_from', lowerBound=None, upperBound=None, numPartitions=10, predicates=None,
properties={ 'user' : 'SYSTEM', 'password' : 'mypassword' })
```

Примеры строк подключения:

СУБД	Драйвер JDBC	URL
Oracle	oracle.jdbc.OracleDriver	jdbc:oracle:oci[OCI_VERSION]:@[HOST_NAME]
Oracle	oracle.jdbc.OracleDriver	jdbc:oracle:thin:@ [HOST_NAME]:[PORT_NUMBER]:[DATABASE_NAME]
MSSQL	com.microsoft.jdbc.sqlserver.SQLServerDriver	jdbc:microsoft:sqlserver: //[HOST_NAME]:[PORT_NUMBER]
PostgreSQL	org.postgresql.Driver	jdbc:postgresql: //[HOST_NAME]:[PORT_NUMBER] /[DATABASE_NAME]
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://[HOST_NAME]:[PORT_NUMBER] /[DATABASE_NAME]

JDBC

Параметры:

url – JDBC URL

table – название таблицы

column – наименование целочисленного поля, который будет использовать для партицирования; если параметр установлен, то numPartitions, lowerBound (inclusive), and upperBound (exclusive) устанавливают ограничения на набор данных в таблице, по которым формируется DataFrame.

lowerBound – минимальное значение поля, по которому формируются партиции.

upperBound – максимальное значение поля, по которому формируются партиции.

numPartitions – количество партиций, которое может быть создано (при параллельной обработке большое количество партиций может уронить источник данных).

predicates – список выражений, которые будут применены в условиях WHERE, каждое выражение определяет одну партицию.

properties – дополнительные аргументы при подключении к jdbc источнику. Например { ‘user’ : ‘SYSTEM’, ‘password’ : ‘mypassword’ }

AVRO

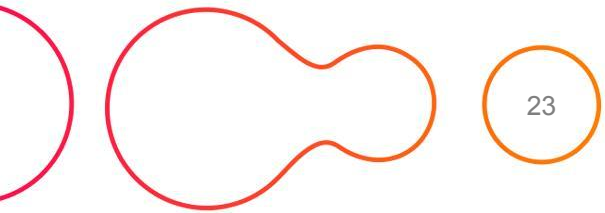
Работа с avro форматом не включена в поставку spark, поэтому поддержку необходимо дополнительно подключать.

```
./bin/spark-submit --packages org.apache.spark:spark-avro_2.12:3.1.1 ...
```

После этого этот формат будет доступен в методе:

```
DF=spark.read.load('hdfs://cluster/user/hdfs/test/exampleavro', format='avro', schema=None, **options)
```

Для чтения avro-файлов – параметр format='avro'



AVRO, LOAD()

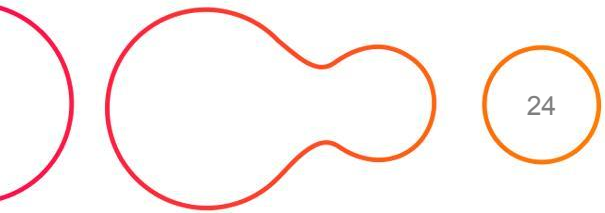
Параметры:

path – строка или список строк, указывающие, где лежат файлы.

format – Строка указывающая на формат файла (avro/csv/json/parquet/orc…). Значение по умолчанию ‘parquet’.

schema – опционально, схема которая описывает данные в файле или DDL форматированная строка, например ‘col0 INT, col1 DOUBLE’.

options – опции, которые можно задать при вызове соответствующего формату метода (например, `load(test.csv, format='csv', sep='\t')`)



Neoflex
IT-решения для бизнеса

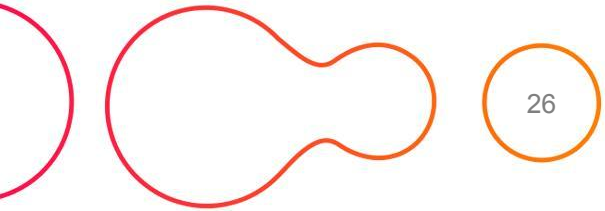
Создание RDD (sequence)

SEQUENCE

Sequence файлы используются для хранения пар “ключ-значение”, в pyspark возможность работать с ними присутствует у RDD, более низкоуровневой структуры данных нежели DataFrame.

```
rdd = spark.sparkContext.sequenceFile('path')
```

path – путь до файла



Neoflex
IT-решения для бизнеса

Сохранение DataFrame в различные форматы данных

CSV

```
DF=spark.write.csv('hdfs://cluster/user/hdfs/test/example.csv‘, mode=‘overwrite’,
 compression= snappy, sep=‘|’, escape=None, header=‘True’, nullValue=‘null’,
 escapeQuotes=None, quoteAll=None, dateFormat=None, timestampFormat=None,
ignoreLeadingWhiteSpace=None, ignoreTrailingWhiteSpace=None, charToEscapeQuoteEscaping=None,
 encoding=None, emptyValue=None)’)
```

CSV

Параметры:

path – по какому пути сохранять файл.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

append: Добавляет содержимое DataFrame к существующим данным.

overwrite: Перезаписывает существующие файлы.

ignore: Отменяет сохранение.

error или **errorifexists** : Генерирует исключение.

compression – кодек сжатия при сохранении файла (none, bzip2, gzip, lz4, snappy and deflate).

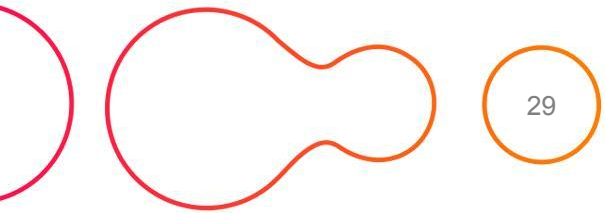
sep – символ, который описывает разделитель в файле. По умолчанию, ‘,’.

header – логическое значение, указывающее записывать названия столбцов в первую строку или нет. По умолчанию ‘False’.

encoding – указывает в какой кодировке сохранять файл. По умолчанию, ‘UTF-8’.

nullValue – устанавливает строку, которая обозначает значение null. По умолчанию пустая строка.

emptyValue – указывает строку, которая будет использоваться при записи пустых значений. По умолчанию пустая строка.



CSV

Параметры:

escape – символ, используемый для экранирования кавычек внутри уже заключенного в кавычки значения. По умолчанию '\'.

escapeQuotes – флаг, включающий обрамление в кавычки всех значений, внутри которых есть кавычки. По умолчанию true, (все кавычки в значениях экранируются).

quoteAll – включает обрамление в кавычки всех значений. По умолчанию, false (все кавычки в значениях экранируются).

dateFormat – устанавливает формат даты в соответствии форматам java.text.SimpleDateFormat. По умолчанию, ‘yyyy-MM-dd’.

timestampFormat – устанавливает формат timestamp в соответствии форматам java.text.SimpleDateFormat. По умолчанию, ‘yyyy-MM-dd'T'HH:mm:ss.SSSZZ’.

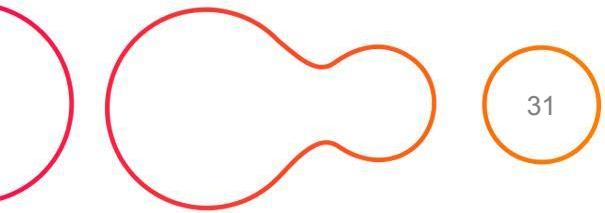
ignoreLeadingWhiteSpace – определяет пропускать или нет пробелы перед значениями. По умолчанию, ‘True’.

ignoreTrailingWhiteSpace – определяет пропускать или нет пробелы после значений. По умолчанию, ‘True’.

charToEscapeQuoteEscaping –символ, используемый для экранирования, если символ экранирования являются кавычки. По умолчанию используется

JSON

```
DF=spark.write.json('hdfs://cluster/user/hdfs/test/examplejson‘, mode=‘overwrite’,
 compression= snappy, dateFormat=None, timestampFormat=None, lineSep=None,
encoding=None)’)
```



Neoflex
IT-решения для бизнеса

JSON

Параметры:

path – по какому пути сохранять файл.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

append: Добавляет содержимое DataFrame к существующим данным.

overwrite: Перезаписывает существующие файлы.

ignore: Отменяет сохранение.

error или **errorifexists** : Генерирует исключение.

compression – кодек сжатия при сохранении файла (none, bzip2, gzip, lz4, snappy and deflate).

lineSep – символ, который описывает разделитель строк в файле. По умолчанию, ‘\n’.

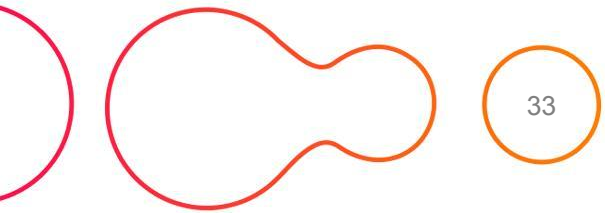
encoding – указывает в какой кодировке сохранять файл. По умолчанию, ‘UTF-8’.

dateFormat – устанавливает формат даты в соответствии форматам java.text.SimpleDateFormat. По умолчанию, ‘yyyy-MM-dd’.

timestampFormat – устанавливает формат timestamp в соответствии форматам java.text.SimpleDateFormat. По умолчанию, ‘yyyy-MM-dd'T'HH:mm:ss.SSSZZ’.

PARQUET

```
DF=spark.write.parquet('hdfs://cluster/user/hdfs/test/exampleparq‘, mode=‘overwrite’,
partitionBy=‘None’, compression= snappy)’)
```



Neoflex
IT-решения для бизнеса

PARQUET

Параметры:

path – по какому пути сохранять файл.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

append: Добавляет содержимое DataFrame к существующим данным.

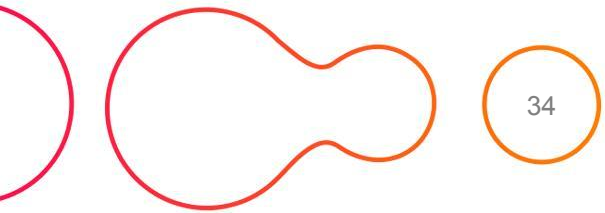
overwrite: Перезаписывает существующие файлы.

ignore: Отменяет сохранение.

error или **errorifexists** : Генерирует исключение.

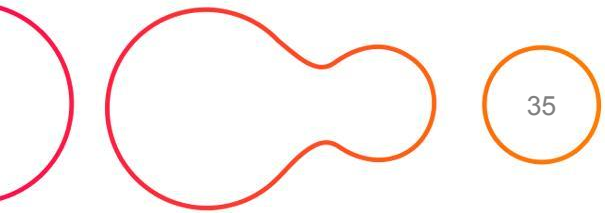
compression – кодек сжатия при сохранении файла (none, bzip2, gzip, lz4, snappy and deflate).

partitionBy – названия столбцов, по которым создавать партиции.



ORC

```
DF=spark.write.orc('hdfs://cluster/user/hdfs/test/exampleorc‘, mode=‘overwrite’,
partitionBy=‘None’, compression= snappy)’)
```



Neoflex
IT-решения для бизнеса

ORC

Параметры:

path – по какому пути сохранять файл.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

append: Добавляет содержимое DataFrame к существующим данным.

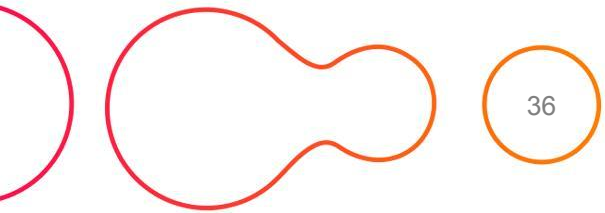
overwrite: Перезаписывает существующие файлы.

ignore: Отменяет сохранение.

error или **errorifexists** : Генерирует исключение.

compression – кодек сжатия при сохранении файла (none, bzip2, gzip, lz4, snappy and deflate).

partitionBy – названия столбцов, по которым создавать партиции.



TEXT

```
DF=spark.write.text('//home/user/test/example.txt', compression='gzip', lineSep='\n\r')
```

Параметры:

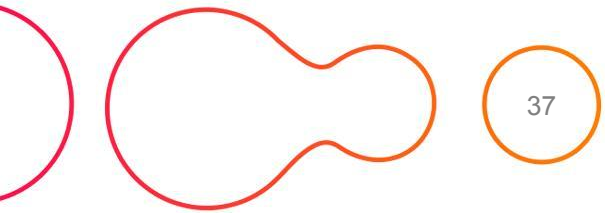
path – по какому пути сохранять файл.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

- **append**: Добавляет содержимое DataFrame к существующим данным.
- **overwrite**: Перезаписывает существующие файлы.
- **ignore**: Отменяет сохранение.
- **error** или **errorifexists** : Генерирует исключение.

compression – кодек сжатия при сохранении файла (none, bzip2, gzip, lz4, snappy and deflate).

lineSep – символ, который описывает разделитель строк в файле. По умолчанию, '\n'.



HIVE

```
DF=spark.write.saveAsTable(name, format=None, mode=None, partitionBy=None, **options)
```

Параметры:

name – имя таблицы

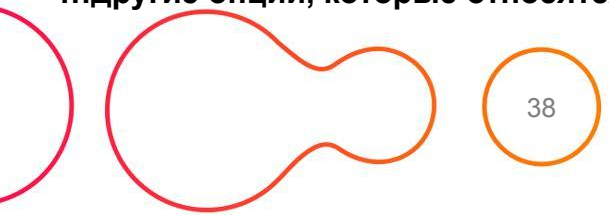
format – в каком формате будет создаваться файл (parquet/csv…)

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

- **append**: Добавляет содержимое DataFrame к существующим данным.
- **overwrite**: Перезаписывает существующие файлы.
- **ignore**: Отменяет сохранение.
- **error** или **errorifexists** : Генерирует исключение.

partitionBy – названия полей, по которым будет партицирование

…другие опции, которые относятся к форматам файлов.



JDBC

```
DF=spark.write.jdbc('jdbc:postgresql://localhost:5432/db_test', 'table_test', properties={
'user' : 'SYSTEM', 'password' : 'mypassword' })
```

Параметры:

url – JDBC URL

table – название таблицы

properties – дополнительные аргументы при подключении к jdbc источнику. Например { 'user' : 'SYSTEM', 'password' : 'mypassword' }

Например:

СУБД	Драйвер JDBC	URL
Oracle	oracle.jdbc.OracleDriver	jdbc:oracle:oci[OCI_VERSION]:@[HOST_NAME]
Oracle	oracle.jdbc.OracleDriver	jdbc:oracle:thin:@ [HOST_NAME]:[PORT_NUMBER]:[DATABASE_NAME]
MSSQL	com.microsoft.jdbc.sqlserver.SQLServerDriver	jdbc:microsoft:sqlserver: //[HOST_NAME]:[PORT_NUMBER]
PostgreSQL	org.postgresql.Driver	jdbc:postgresql: //[HOST_NAME]:[PORT_NUMBER] /[DATABASE_NAME]
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://[HOST_NAME]:[PORT_NUMBER] /[DATABASE_NAME]

AVRO

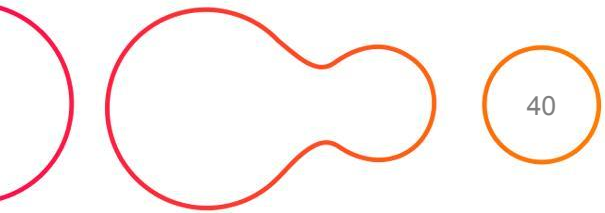
Работа с avro форматом не включена в поставку spark, поэтому поддержку необходимо дополнительно подключать.

```
./bin/spark-submit --packages org.apache.spark:spark-avro_2.12:3.1.1 ...
```

После этого этот формат будет доступен в методе:

```
DF=spark.write.save('hdfs://cluster/user/hdfs/test/exampleavro', format='avro',
mode='overwrite', partitionBy='None', **options)
```

Для чтения avro-файлов – параметр format='avro'



Neoflex
IT-решения для бизнеса

AVRO, SAVE()

Параметры:

path – строка или список строк, указывающие, где лежат файлы.

format – Строка указывающая на формат файла (avro/csv/json/parquet/orc…). Значение по умолчанию ‘parquet’.

mode – определяет поведение, если файлы с таким именем уже созданы, по умолчанию «error».

- **append**: Добавляет содержимое DataFrame к существующим данным.
- **overwrite**: Перезаписывает существующие файлы.
- **ignore**: Отменяет сохранение.
- **error** или **errorifexists** : Генерирует исключение.

partitionBy – названия столбцов, по которым создавать партиции.

options – опции, которые можно задать при вызове соответствующего формату метода (например, `load(test.csv, format='csv', sep='\t')`)

Neoflex
IT-решения для бизнеса

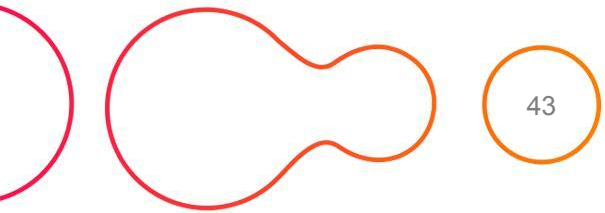
Сохранеие RDD (sequence)

SEQUENCE

Sequence файлы используются для хранения пар “ключ-значение”, в pyspark возможность работать с ними присутствует у RDD, более низкоуровневой структуры данных нежели DataFrame.

```
rdd.saveAsSequenceFile('path')
```

path – путь до файла



Neoflex
IT-решения для бизнеса

КОНТАКТЫ

www.neoflex.ru

Neoflex
IT-решения для бизнеса