



Semantix

Spark

Aula 10

Quem sou eu?

Eu sou Rodrigo Augusto Rebouças.

Engenheiro de dados da Semantix
Instrutor do Semantix Academy

Você pode me encontrar em:
rodrigo.augusto@semantix.com.br





Esquemas

Introdução



Spark Esquemas

- Spark pode inferir os esquemas para os dados estruturados
 - Parquet
 - Tabela Hive
- Spark pode tentar inferir os esquemas para os dados semi estruturados
 - csv
 - json

Definição Esquemas

- Inferir esquema automaticamente em dados sem cabeçalho

setor.csv

```
1, vendas  
2, TI  
3, RH
```

```
scala> val sDF = spark.read.csv("setor.csv").printSchema()
```

```
_c0: string (nullable = true)
```

```
_c1: string (nullable = true)
```

```
scala> val sDF = spark.read.option("inferSchema","true"). \
```

```
csv("setor.csv").printSchema()
```

```
_c0: integer (nullable = true)
```

```
_c1: string (nullable = true)
```

Definição Esquemas

- Inferir esquema automaticamente em dados com cabeçalho

setor.csv

```
id, setor  
1, vendas  
2, TI  
3, RH
```

```
scala> val sDF = spark.read.csv("setor.csv").printSchema()
```

```
_c0: string (nullable = true)
```

```
_c1: string (nullable = true)
```

```
scala> val sDF = spark.read.option("inferSchema","true"). \
```

```
option("header","true").csv("setor.csv").printSchema()
```

```
id: integer (nullable = true)
```

```
vendas: string (nullable = true)
```



Join

Introdução

- Tipos de Join
 - inner (padrão)
 - outer
 - left_outer
 - right_outer
 - Leftsemi

Exemplo Inner Join

cliente.csv

id_c, nome
15, João
20, Carlos
, Marcos
30, Ana
15, Maria

cidade.csv

id_c, cidade
15, São Paulo
20, Santos
25, Campinas

id_c	nome	cidade
15	João	São Paulo
20	Carlos	Santos
30	Ana	null
15	Maria	São Paulo

```
scala> val clienteDF = spark.read.option("header","true").csv("cliente.csv")
```

```
scala> val cidadeDF = spark.read.option("header","true").csv("cidade.csv")
```

```
scala> clienteDF.join(cidadeDF, "id_c").show()
```

Exemplo Left Outer Join

cliente.csv

id_c, nome
15, João
20, Carlos
, Marcos
30, Ana
15, Maria

cidade.csv

id_c, cidade
15, São Paulo
20, Santos
25, Campinas

id_c	nome	cidade
15	João	São Paulo
20	Carlos	Santos
null	Marcos	null
30	Ana	null
15	Maria	São Paulo

```
scala> val clienteDF = spark.read.option("header","true").csv("cliente.csv")
```

```
scala> val cidadeDF = spark.read.option("header","true").csv("cidade.csv")
```

```
scala> clienteDF.join(cidadeDF, clienteDF("id_c") === cidadeDF("id_c"), "left_outer").show()
```



Laboratório

Resolução de Exercícios

Exercícios Join e Esquema

1. Criar o DataFrame **alunosDF** para ler o arquivo no hdfs “/user/aluno/<nome>/data/escola/alunos.csv” sem usar as “option”
2. Visualizar o esquema do alunosDF
3. Criar o DataFrame **alunosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/alunos.csv” com a opção de Incluir o cabeçalho
4. Visualizar o esquema do alunosDF
5. Criar o DataFrame **alunosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/alunos.csv” com a opção de Incluir o cabeçalho e inferir o esquema
6. Visualizar o esquema do alunosDF
7. Salvar o DataFrame alunosDF como tabela Hive “tab_alunos” no banco de dados <nome>
8. Criar o DataFrame **cursosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/cursos.csv” com a opção de Incluir o cabeçalho e inferir o esquema
9. Criar o DataFrame **alunos_cursosDF** com o inner join do alunosDF e cursosDF quando o id_curso dos 2 forem o mesmo
10. Visualizar os dados, o esquema e a quantidade de registros do alunos_cursosDF



Spark SQL Queries

Introdução

Spark SQL queries

- Realizar consultas com comandos SQL
 - Tabelas Hives
 - Views de DataFrame e dataset
 - Outros formatos de arquivos
- Usar a função `SparkSession.sql` para executar uma consulta SQL
 - Retornar um DataFrame

```
scala> val myDF = spark.sql("select * from people")  
scala> myDF.printSchema()  
scala> myDF.show()
```

people está localizado onde?

SQL Queries

Leitura de dados

- Default: Tabelas Hive
- Arquivo Parquet
- Arquivo Json

```
scala> val testDF = spark.sql("SELECT * FROM tabela")
```

```
scala> val testDF = spark.sql("SELECT * FROM parquet.`/bd/tabela.parquet`")
```

```
scala> val testDF = spark.sql("SELECT * FROM json.`/bd/tabela.json`")
```



Spark SQL Queries

Consultas e Views

SQL Queries Exemplo de consultas

- Select
- Where
- Group by
- Having
- Order by
- Limit
- Count
- Sum
- Mean
- As
- Subqueries

```
scala> val maAgeDF = spark.sql("SELECT MEAN(age) AS mean_age, STDDEV(age)  
AS sdev_age FROM people WHERE pcode IN (SELECT pcode FROM pcodes  
WHERE state='MA')")
```

SQL Queries Views

- Possibilidade para executar SQL queries em DataFrame e DataSet
- São temporárias
 - Views Regular - usar em apenas uma seção Spark
 - Views Global - usar em múltiplas seções Spark através de uma aplicação Spark
- Criar uma view
 - `DataFrame.createTempView(view-name)`
 - `DataFrame.createOrReplaceTempView(view-name)`
 - `DataFrame.createGlobalTempView(view-name)`

```
scala> val clienteDF = spark.read.json("cliente.json").createTempView("clienteView")
scala> val tabDF = spark.sql("select * from clienteView ").show(10)
```




API Catalog



API Catalog

- Explorar tabelas e gerenciar Views
- Usar spark.catalog
- Funções
 - listDatabases
 - Listas banco de dados
 - Caso não apareça o BD desejado
 - Problema de configuração do spark
 - Importar package
 - setCurrentDatabase(nomeBD)
 - Setar o banco de dados padrão de leitura
 - listTables
 - Listar tabelas e views do BD atual
 - listColumns(nomeTabela)
 - Listar colunas de uma tabela ou view
 - dropTempView(nomeView)
 - Remover view

Exemplo API Catalog

```
scala> val tabDF = spark.sql("select * from bdtest.user").show(10)
```

```
scala> spark.catalog.listDatabases.show()
```

```
scala> spark.catalog.setCurrentDatabase("bdtest")
```

```
scala> spark.catalog.listTables.show()
```

```
scala> val tabDF = spark.sql("select * from user").show(10)
```



Laboratório


Resolução de Exercícios



Exercícios API Catalog

Realizar os exercícios usando a API Catalog.

1. Visualizar todos os banco de dados
2. Definir o banco de dados “seu-nome” como principal
3. Visualizar todas as tabelas do banco de dados “seu-nome”
4. Visualizar as colunas da tabela tab_alunos
5. Visualizar os 10 primeiros registros da tabela "tab_alunos" com uso do spark.sql



SQL Queries vs Operações de DataFrame

SQL Queries vs Operações de DataFrame

- Funcionalidade equivalente
- Otimizadas pelo Catalyst optimizer
- Transformação DataFrame
 - Maior separabilidade do que as queries
- SQL queries
 - Facilidade em programação com conhecimento apenas em SQL

```
// Transformação DataFrame
```

```
scala> val testDF = spark.read.table("cliente").where("id=10255")
```

```
// SQL queries
```

```
scala> val testDF = spark.sql("SELECT * FROM cliente WHERE id = 10255")
```



Laboratório

Resolução de Exercícios

Exercícios SQL Queries vs Operações de DataFrame

Realizar as seguintes consultas usando SQL queries e transformações de DataFrame na tabela Hive “tab_alunos” no banco de dados <nome>

1. Visualizar o id e nome dos 5 primeiros registros
2. Visualizar o id, nome e ano quando o ano de ingresso for maior ou igual a 2018
3. Visualizar por ordem alfabética do nome o id, nome e ano quando o ano de ingresso for maior ou igual a 2018
4. Contar a quantidade de registros do item anterior



Semantix

Obrigado!

Alguma pergunta?



Você pode me encontrar em:
rodrigo.augusto@semantix.com.br

GET SMARTER