

ESTUDO DE CASO: UBER

Big Data

RELEMBRANDO...



CONTEXTUALIZAÇÃO

PIONEIRA NO SEGMENTO DE TRANSPORTE

Desenvolve serviços de mobilidade, entregas e transporte de mercadorias (freight)

PRESENTE EM 72 PAÍSES

Atua em todos os continentes urbanos e processa uma enorme volume de dados dado seu alcance global

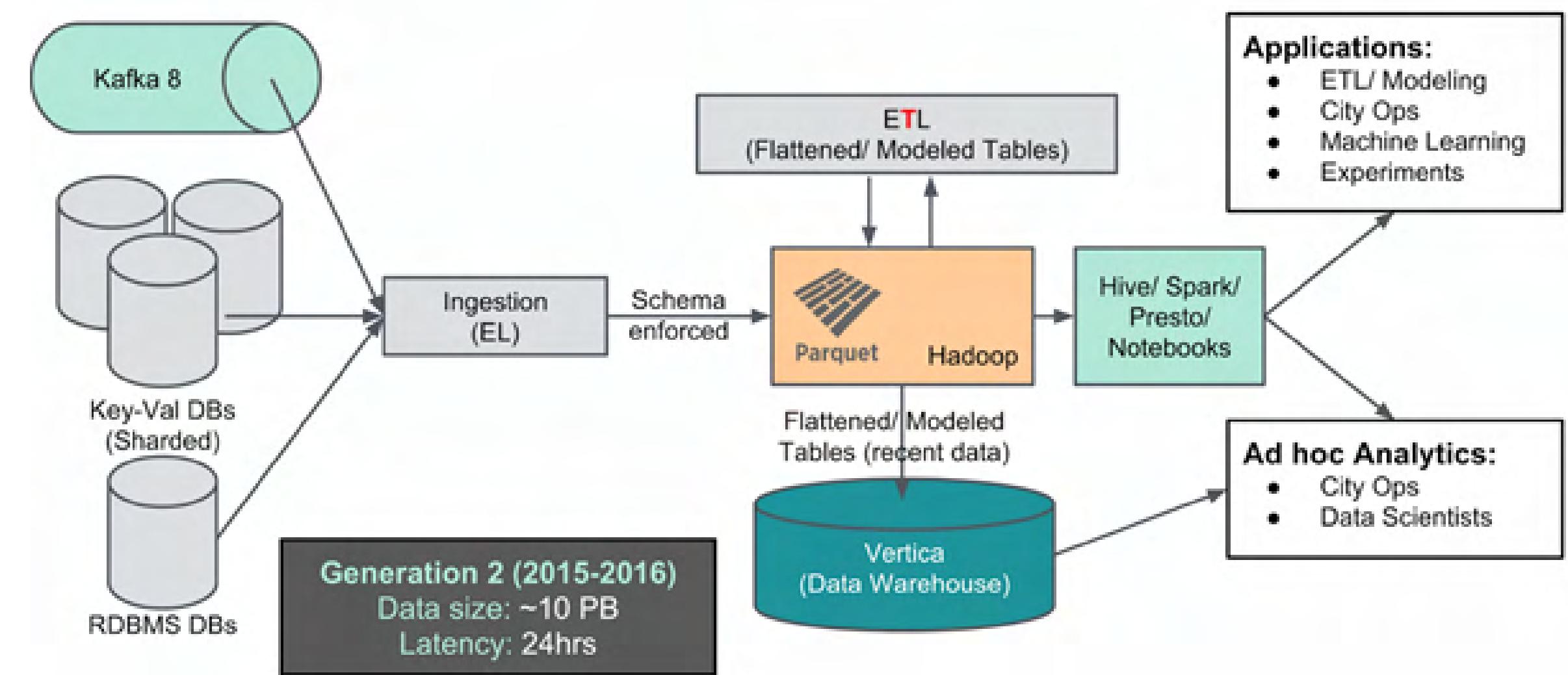
MASSIVA REDE DE COLABORADORES

A Uber possui milhões de colaboradores, entre eles: motoristas, consumidores, comerciantes, transportadoras e entregadores

Arquitetura dos Sistemas de Big Data

Ecossistema Hadoop

Nova abordagem permitiu ganhos de performance na ingestão e transformações desnecessárias de dados. Tal estrutura foi desenhada para ser utilizada em conjunto com o Presto, Apache Spark, Apache Hive e Apache Parquet



LIMITAÇÃO ESCOLHIDA:

Arquivos .Parquet não podem ser alterados

Imutabilidade dos arquivos Parquet

THESE THINGS HAPPEN

fixing bugs and chewing bubblegum

[Blog](#) [Archives](#) [Date/time Cheat Sheet](#)

Search

So You Need to Edit a Parquet File

Aug 4th, 2017

You've uncovered a problem in your beautiful parquet files, some piece of data either snuck in, or was calculated incorrectly, or there was just a bug. You know exactly how to correct the data, but how do you update the files?

tl;dr: [spark-edit-examples](#)

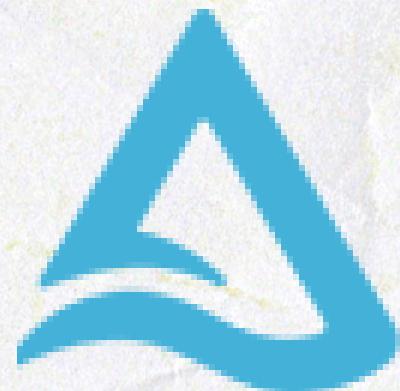
It's all immutable

The problem we have when we need to edit the data is that our data structures are immutable.

You can add partitions to Parquet files, but you can't edit the data in place. Spark DataFrames are immutable.

But ultimately we can mutate the data, we just need to accept that we won't be doing it in place. We will need to recreate the Parquet files using a combination of schemas and UDFs to correct the bad data.

COMO A LIMITAÇÃO FORA MITIGADA?



DELTA LAKE

Experimentos



Alterações em um Conjunto de Dados



Salvar os Dados Alterados no Banco de Dados

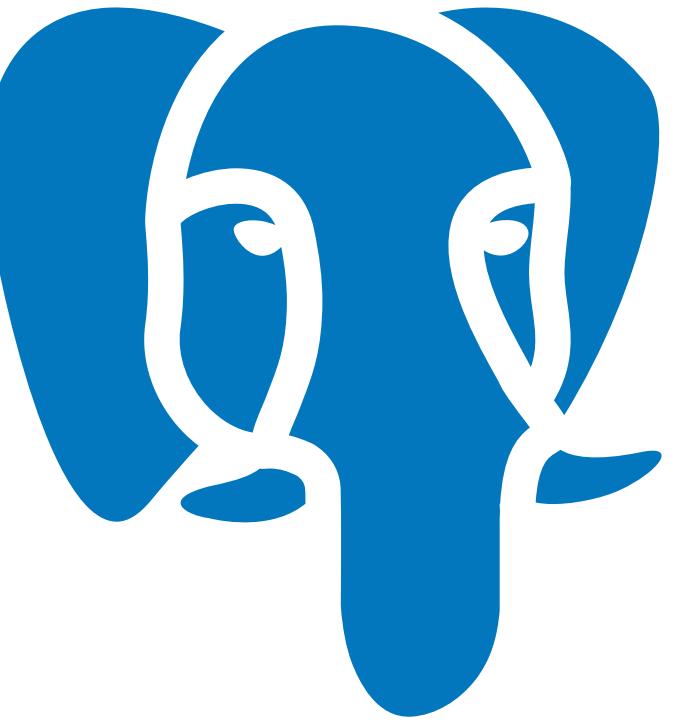
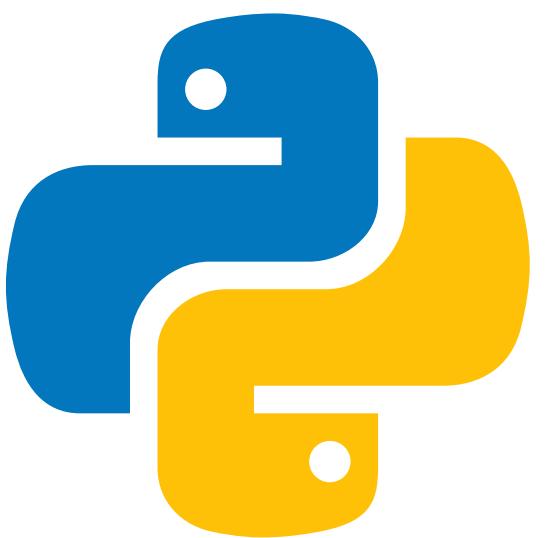
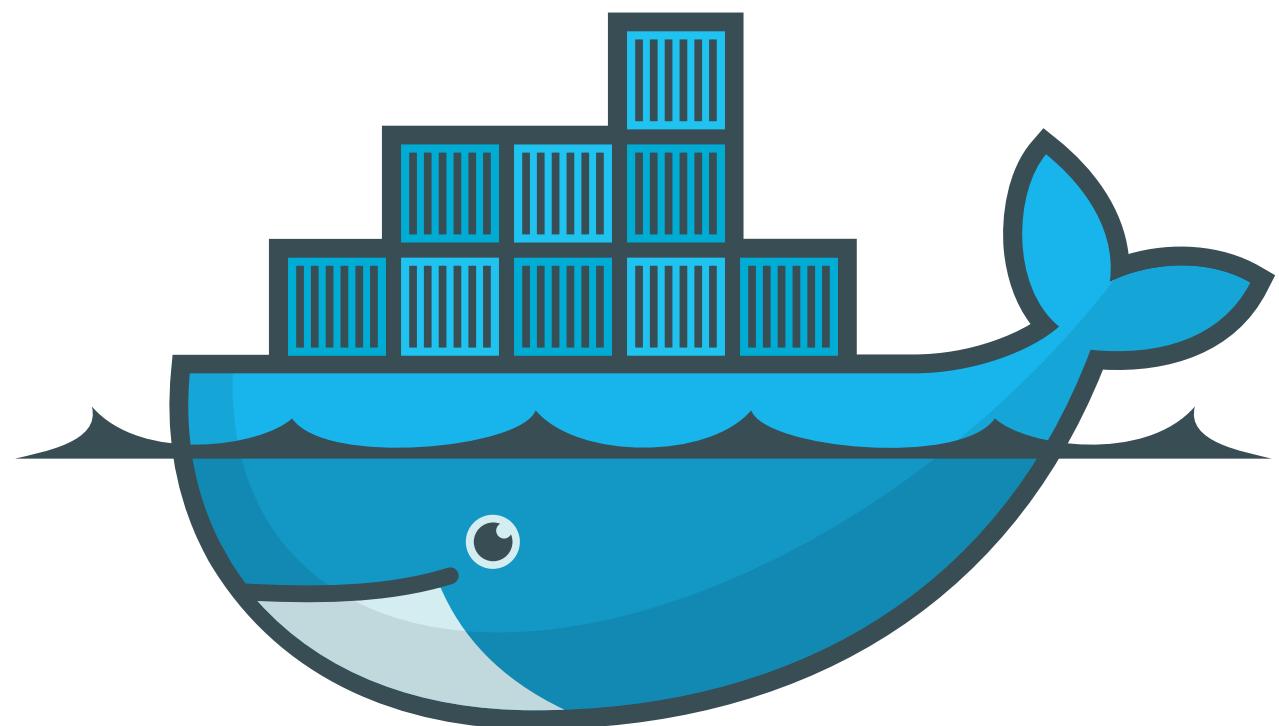


Recuperar Dados Erroneamente Deletados ou alterados

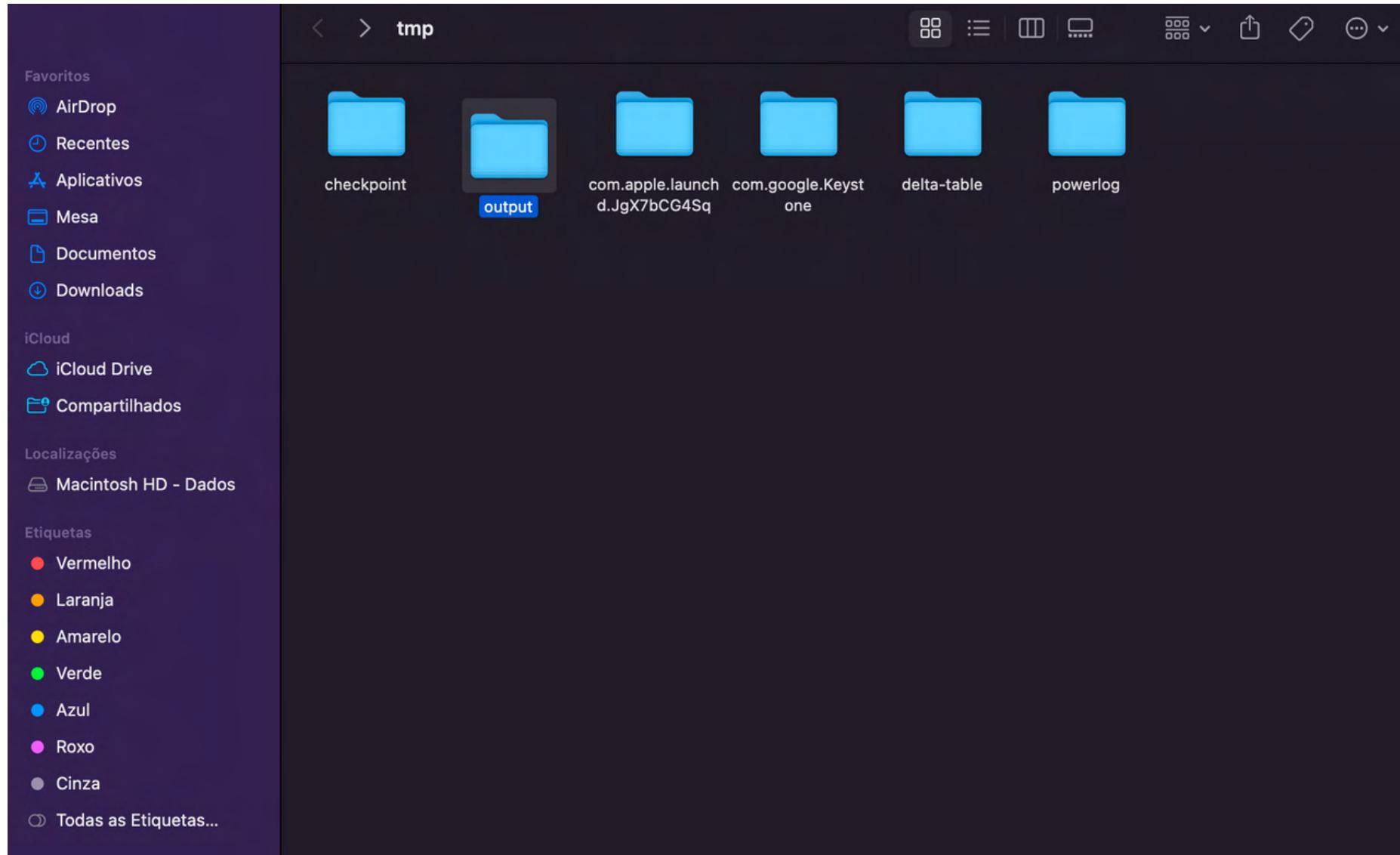
Preparação



Java



Preparação



The screenshot shows a Mac OS X Finder window with a dark theme. The left sidebar contains 'Favoritos' (AirDrop, Recentes, Aplicativos, Mesa, Documentos, Downloads), 'iCloud' (iCloud Drive, Compartilhados), 'Localizações' (Macintosh HD - Dados), and 'Etiquetas' (Vermelho, Laranja, Amarelo, Verde, Azul, Roxo, Cinza, Todas as Etiquetas...). The main pane is titled 'tmp' and contains several folder icons: 'checkpoint', 'output' (highlighted in blue), 'com.apple.launchd.JgX7bCG4Sq', 'com.google.Keystone', 'delta-table', and 'powerlog'. The top bar has standard OS X icons for file operations.

Importar as bibliotecas necessárias.

```
[1]: import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta.tables import *
from delta import *
import pyspark.sql.functions as PSF
import psycopg2
import pandas as pd
import os
import time
```

Coletar as variáveis de ambiente definidas em Variáveis de Ambiente.

```
[5]: env_vars = !cat .../Environment/.env
for var in env_vars:
    key, value = var.split('=')
    os.environ[key] = value
```

Configuração do Ambiente

```
[6]: builder = pyspark.sql.SparkSession.builder.appName("PySpark_Postgres_test").config("spark.jars", os.environ.get('SPARKJAR')) \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .master("local").appName("PySpark_Postgres_test")

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

Preparação

Ler o arquivo a fim de enviá-lo para o banco de dados.

```
[8]: dataf = spark.read.format("csv").option("delimiter", ",").option("header", "true").option("inferSchema", "true") \
    .load("../data/200ksales.csv")
dataf.show()
```

	Region	Country	Item_Type	Sales_Channel	Order_Priority	Order_Date	Order_ID	Ship_Date	Units_Sold	Unit_Price	Unit_Cost	Total_Revenue	Total_Cost	Total_Profit
00	Australia and Oce...	Australia	Meat	Online	C	4/4/2011	451691138	5/23/2011	43	1814127	1568167	245960		
45	Asia	Tajikistan	Personal Care	Online	L	7/12/2018	144177377	8/1/2018	41	338770	234897	103873		
07	Sub-Saharan Africa	Mozambique	Cosmetics	Online	H	7/6/2011	982716166	7/17/2011	64	2801140	1687155	1113985		
10	Central America and...	Panama	Personal Care	Offline	L	5/1/2011	784543836	5/11/2011	28	229661	159242	70418		
10	North America	Canada	Fruits	Online	H	11/15/2013	137209212	12/29/2013	21	19686	14601	5085		
92	Europe	Italy	Fruits	Offline	C	7/16/2016	148573625	7/18/2016	50	47508	35236	12271		
74	Europe	Romania	Beverages	Offline	C	11/15/2012	196810901	12/1/2012	95	454286	304357	149928		
04	Australia and Oce...	Australia	Cosmetics	Offline	L	4/30/2019	180407369	5/25/2019	54	2362628	1423035	939593		
63	Europe Bosnia and Herzeg...		Cosmetics	Offline	C	5/29/2016	968161857	6/8/2016	94	4137223	2491891	1645331		
61	Asia	Japan	Clothes	Online	M	3/25/2018	349185693	4/8/2018	6	72234	23690	48543		
41	Australia and Oce...	Papua New Guinea	Beverages	Online	H	8/28/2013	682902705	9/18/2013	49	234450	157074	77376		

```
[9]: lista_de_colunas_inteiro = ["Units_Sold", "Unit_Price", "Unit_Cost", "Total_Revenue", "Total_Cost", "Total_Profit"]

def convert_data_types(df, columns_list):
    for column_name in columns_list:
        df.select(PSF.col(column_name).cast('bigint').alias(column_name))
    return df

[10]: convert_data_types(dataf, lista_de_colunas_inteiro)

DataFrame[Region: string, Country: string, Item_Type: string, Sales_Channel: string, Order_Priority: string, Order_Date: string, Order_ID: int, Ship_Date: string, Units_Sold: int, Unit_Price: int, Unit_Cost: int, Total_Revenue: int, Total_Cost: int, Total_Profit: int]
```

Salvar o arquivo 200ksales.csv no banco de dados.

```
# O parâmetro OVERWRITE somente será True se já
# existir uma tabela cujo nome é o passado no
# parâmetro DBNAME.

def save_data_to_database(df, dbname: str, overwrite: bool):
    if overwrite:
        df.write.format("jdbc").option("driver", "org.postgresql.Driver").option("url", os.environ.get('DB_CONNECTION_STRING')) \
            .option("dbtable", dbname).option("user", os.environ.get('USER')).option("password", os.environ.get('PASSWORD')) \
            .mode("overwrite").save()
        return print("Overwrite done!")

    else:
        df.write.format("jdbc").option("driver", "org.postgresql.Driver").option("url", os.environ.get('DB_CONNECTION_STRING')) \
            .option("dbtable", dbname).option("user", os.environ.get('USER')).option("password", os.environ.get('PASSWORD')).save()
        return print("Saved without overwrite!")

save_data_to_database(dataf, "sales", True)
```

Experimentos

```
[4]: dataf = spark.read.format("jdbc").option("url", os.environ.get('DB_CONNECTION_STRING')) \
    .option("driver", "org.postgresql.Driver").option("dbtable", "sales") \
    .option("user", os.environ.get('USER')).option("password", os.environ.get('PASSWORD')).load()

[5]: dataf.where(dataf.Order_ID == 451691138).show()

+-----+-----+-----+-----+-----+-----+
| Region|Country|Item_Type|Sales_Channel|Order_Priority|Order_Date|Order_ID|Ship_Date|Unit
it_Cost|Total_Revenue|Total_Cost|Total_Profit|
+-----+-----+-----+-----+-----+-----+
|Australia and Oce...|Australia|Meat|Online|C| 4/4/2011|451691138|5/23/2011|
364| 1814127| 1568167| 245960|
+-----+-----+-----+-----+-----+-----+-----+
```



```
[37]: dataf.write.format("delta").save("/tmp/output/experimentodelta")

[38]: parDF = spark.read.format("delta").load("/tmp/output/experimentodelta")
parDF.where(parDF.Order_ID == 451691138).show()

+-----+-----+-----+-----+-----+-----+
| Region|Country|Item_Type|Sales_Channel|Order_Priority|Order_Date|Order_ID|Ship_Date|Unit
it_Cost|Total_Revenue|Total_Cost|Total_Profit|
+-----+-----+-----+-----+-----+-----+
|Australia and Oce...|Australia|Meat|Online|C| 4/4/2011|451691138|5/23/2011|
364| 1814127| 1568167| 245960|
+-----+-----+-----+-----+-----+-----+
```

Leitura dos dados presentes no banco de dados

```
[39]: teste = DeltaTable.forPath(spark, "/tmp/output/experimentodelta")
start = time.time()
for i in range(100):
    teste.update(
        condition = PSF.expr("Order_ID == 451691138"),
        set = { "Unit_Price": PSF.expr("Unit_Price + 1") })
    teste.update(
        condition = PSF.expr("Order_ID == 451691138"),
        set = { "Units_Sold": PSF.expr("Units_Sold + 1") })
end = time.time()
print(end - start)

761.0740370750427

[41]: parDF = spark.read.format("delta").load("/tmp/output/experimentodelta")
parDF.where(parDF.Order_ID == 451691138).show()

+-----+-----+-----+-----+-----+-----+
| Region|Country|Item_Type|Sales_Channel|Order_Priority|Order_Date|Order_ID|Ship_Date|Units_Sold|Unit_Price|Un
it_Cost|Total_Revenue|Total_Cost|Total_Profit|
+-----+-----+-----+-----+-----+-----+
|Australia and Oce...|Australia|Meat|Online|C| 4/4/2011|451691138|5/23/2011| 4400| 521|
364| 1814127| 1568167| 245960|
+-----+-----+-----+-----+-----+-----+
```

Alterações nas colunas Unit_Price e Units_Sold

Experimentos

The screenshot shows a PostgreSQL client interface with a table named "sales". The table has columns: Order_ID, Ship_Date, Units_Sold, Unit_Price, Unit_Cost, Total_Revenue, and Total_Cost. A specific row is selected, and its details are shown in a sidebar. The sidebar also lists other columns and their values.

Order_ID	Ship_Date	Units_Sold	Unit_Price	Unit_Cost	Total_Revenue	Total_Cost
138	5/23/2011	4400	521	364	1814127	1568167
377	8/1/2018	4145	81	56	338770	234897
166	7/17/2011	6407	437	263	2801140	1687155
336	5/11/2011	2810	81	56	229661	159242
212	12/29/2013	2110	9	6	19686	14601
625	7/18/2016	5092	9	6	47508	35236
901	12/1/2012	9574	47	31	454286	304357
369	5/25/2019	5404	437	263	2362628	1423035
357	6/8/2016	9463	437	263	4137223	2491891
693	4/8/2018	661	109	35	72234	23690
705	9/18/2013	4941	47	31	234450	157074
222	7/10/2014	276	9	6	2575	1909
071	11/16/2017	1672	81	56	136652	94752
581	12/10/2011	6164	421	364	2600529	2247949

Dados atualizados no banco

```
[42]: start = time.time()
parDF.write.format("jdbc").option("driver", "org.postgresql.Driver").option("url", os.environ.get('DB_CONNECTION_STRING')) \
    .option("dbtable", "sales").option("user", os.environ.get('USER')).option("password", os.environ.get('PASSWORD')) \
    .mode("overwrite").save()
end = time.time()
print(end - start)

[Stage 16378:> (0 + 1) / 1]
22.51351499557495
```

Salvando os dados no banco

Experimentos

```
[49]: deleting = DeltaTable.forPath(spark, "/tmp/output/experimentodelta")
```

```
[52]: deleting.delete(PSF.col('Order_ID') == 451691138)
```



• • •

```
[60]: deletedDF = spark.read.format("delta").load("/tmp/output/experimentodelta")
deletedDF.count()
```

```
[60]: 254159
```

```
[54]: deletedDF.where(deletedDF.Order_ID == 451691138).count()
```

```
[54]: 0
```

```
[64]: save_data_to_database(deletedDF, "sales", True)
```

```
[Stage 16481:> (0 + 1) / 1]
Overwrite done!
```

Subindo no banco de dados uma versão sem o pedido número 451691138

Experimentos

```
[49]: deleting = DeltaTable.forPath(spark, "/tmp/output/experimentodelta")
```

```
[52]: deleting.delete(PSF.col('Order_ID') == 451691138)
```

• • •

```
[60]: deletedDF = spark.read.format("delta").load("/tmp/output/experimentodelta")
deletedDF.count()
```

```
[60]: 254159
```

```
[54]: deletedDF.where(deletedDF.Order_ID == 451691138).count()
```

```
[54]: 0
```

```
[64]: save_data_to_database(deletedDF, "sales", True)
```

```
[Stage 16481:> (0 + 1) / 1]
Overwrite done!
```

Subindo no banco de dados uma versão sem o pedido número 451691138

Experimentos

The screenshot shows the pgAdmin 4 interface for a PostgreSQL 12.10 database named 'Tendencias'. In the left sidebar, under the 'tendencias' database, the 'Tables' section is expanded, showing the 'sales' table. The main window displays an SQL query in the 'SQL Query' tab:

```
1 SELECT * from sales where "Order_ID" = 451691138;
```

The results pane below shows a table header with columns: Region, Country, Item_Type, Sales_Channel, Order_Priority, and Order_. A message 'No row selected' is displayed. At the bottom of the results pane, there is a log of system messages:

```
-- 2022-05-18 09:11:31.1570
select reltuples::int8 as count from pg_class c JOIN
pg_catalog.pg_namespace n ON n.oid=c.relnamespace where
nspname='public' AND relname='sales';

-- 2022-05-18 09:11:31.1570
SELECT COUNT(*) as count FROM "public"."sales";

-- 2022-05-18 09:22:34.5420
SELECT * from sales where "Order_ID" = 451691138 LIMIT 100;
```

The status bar at the bottom indicates 'Enable syntax highlighting' is checked.

Pedido número 451691138 devidamente excluído do banco

Experimentos

```
[56]: history = spark.sql("DESCRIBE HISTORY delta.`/tmp/output/experimentodelta`")
latest_version = history.selectExpr("max(version)").collect()
print(latest_version[0][0])
201

[65]: check_versions = spark.read.format("delta").option("versionAsOf", 200).load("/tmp/output/experimentodelta")
check_versions.count()

[65]: 254160

[73]: df_final = check_versions.where(check_versions.Order_ID == 451691138)
df_final.count()

[73]: 1

[74]: save_data_to_database(df_final, "sales", False, True)
```



Resgate de uma versão anterior do pedido número 451691138

Experimentos

```
1 SELECT
2   *
3   FROM
4     sales
5 WHERE
6   "Order_ID" = 451691138;
```

_Type	Sales_Channel	Order_Priority	Order_Date	Order_ID	Ship_Date	Units_Sold
	Online	C	4/4/2011	451691138	5/23/2011	44

— 2022-05-18 09:34:19.4210
SELECT * from sales where "Order_ID" = 451691138 LIMIT 100;
— 2022-05-18 09:34:31.8400
SELECT
 *
FROM
 sales
WHERE
 "Order_ID" = 451691138 LIMIT 100;

public Enable syntax highlighting

Region: Australia and Oceania
Country: Australia
Item_Type: Meat
Sales_Channel: Online
Order_Priority: C
Order_Date: 4/4/2011
Order_ID: 451691138
Ship_Date: 5/23/2011
Units_Sold: 4400
Unit_Price: 521
Unit_Cost: 364
Total_Revenue: 1814127
Total_Cost: 1568167

Pedido número 451691138 reinserido no banco

Resultados Alcançados

- 01 3+ milhões de dados inseridos no banco em segundos
- 02 100 alterações em dados específicos em cerca de 10 minutos
- 03 1000 alterações em dados específicos em cerca de 1 Hora
- 04 Exclusão de dados específicos em segundos
- 05 Recuperação de dados outrora excluídos

Obrigado!

Grupo:

Renato Gabriel Ferreira