TRATAMENTO DE DADOS DOS CNPJS DO BRASIL COM PYSPARK E ORACLE DATABASE

Lucas dos Santos Dourado[1], Geraldo Henrique Neto[1], Anna Patricia Zakem China[1]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

[1]lucas.dourado2@fatec.sp.gov.br,

[1]geraldo.henrique@fatec.sp.gov.br,

[1]anna.china@fatec.sp.gov.br

Resumo. *Este artigo descreve como realizar o tratamento dos dados disponibilizados no repositório de CNPJs, da plataforma federal "dados.gov.br". Devido ao grande volume de dados, o tratamento foi feito com a ferramenta Pyspark, que possibilitou a leitura dos dados em grande escala, como também o armazenamento, via processamento paralelo, no database da Oracle.*

Abstract. *This article describes how to process the data available in the CNPJs repository, of the federal platform "dados.gov.br". Due to the large volume of data, Pyspark tool was sued to handle the data, which allows large-scale data reading, as well as storage, via parallel processing, in the Oracle database.*

1. Introdução

Nos últimos anos, o crescente volume de dados disponíveis tem se tornado uma valiosa fonte de informações para empresas e organizações governamentais. No Brasil, o repositório de CNPJs (Cadastro Nacional de Pessoas Jurídicas) da plataforma dados.gov.br se destaca como uma importante fonte de dados para análises estatísticas, estudos de mercado e tomada de decisões estratégicas.

No entanto, lidar com grandes volumes de dados requer o uso de ferramentas eficientes e escaláveis. Nesse contexto, o *PySpark*, um módulo do *Apache Spark* que permite o processamento distribuído de dados, surge como uma solução poderosa para lidar com o tratamento e análise de grandes conjuntos de dados. Além disso, o *Oracle Database*, conhecido por sua robustez e desempenho, apresenta-se como uma opção confiável para armazenar essas informações valiosas.

Neste artigo, explora-se o processo de tratamento de dados do repositório de CNPJs do Brasil, disponível na plataforma dados.gov.br, utilizando a ferramenta *PySpark* e armazenando os dados resultantes no *Oracle Database*. Será demonstrado como a combinação dessas tecnologias pode facilitar o tratamento de grandes volumes de dados, garantindo eficiência e escalabilidade.

Primeiramente, aborda-se a importância dos dados do repositório de CNPJs do Brasil como fonte de informações estratégicas. Em seguida, será discutido os conceitos-

chave por trás do *PySpark*, destacando suas capacidades de processamento distribuído e sua integração com o ambiente *Python*. Posteriormente, será explorado como o *Oracle Database* pode ser utilizado para armazenar os dados tratados de forma segura e confiável.

Por fim, será apresentado um estudo de caso, demonstrando passo a passo o processo de tratamento de dados do repositório de CNPJs utilizando *PySpark* e a subsequente carga dos dados no *Oracle Database*. Com isso, pretendemos fornecer uma visão abrangente de como essas tecnologias podem ser empregadas para obter *insights* valiosos a partir de grandes conjuntos de dados, contribuindo para a tomada de decisões informadas e eficazes em diversas áreas.

2. O portal Dados.gov.br

No cenário atual, em que a sociedade está cada vez mais interligada e orientada por dados, a transparência governamental e o acesso às informações públicas são fundamentais para fortalecer a democracia e promover o desenvolvimento de uma sociedade participativa. Nesse contexto, o Brasil deu um passo significativo ao lançar o portal de dados abertos dados.gov.br, uma plataforma digital que concentra uma ampla quantidade de informações e estatísticas produzidas pelo governo federal.

O dados.gov.br foi lançado em maio de 2012, com o objetivo de disponibilizar informações governamentais de forma aberta, livre e acessível a todos os cidadãos brasileiros. Por meio dessa plataforma, é possível acessar conjuntos de dados em diversas áreas, como educação, saúde, segurança, transporte, economia e meio ambiente. A interface intuitiva e de fácil navegação permite que qualquer pessoa interessada possa baixar e utilizar essas informações para diferentes propósitos, como pesquisa acadêmica, desenvolvimento de aplicativos, tomada de decisões embasadas e monitoramento das ações governamentais.

A existência do dados.gov.br fortalece a transparência governamental ao possibilitar que os cidadãos e a sociedade civil acompanhem e fiscalizem as atividades do governo de maneira mais efetiva. Por meio dos dados disponíveis na plataforma, é possível avaliar indicadores sociais, analisar políticas públicas e monitorar os gastos governamentais. Essa transparência contribui para reduzir a corrupção, aumentar a prestação de contas e estimular o engajamento cívico.

É importante ressaltar que o portal dados.gov.br está alinhado com a legislação brasileira de dados abertos, que estabelece os princípios e diretrizes para a disponibilização de informações públicas. A Lei de Acesso à Informação (Lei nº 12.527/2011) e o Decreto nº 8.777/2016 são marcos legais que asseguram o acesso à informação e a transparência no setor público. Essa legislação estabelece a obrigatoriedade de disponibilizar dados em formato aberto, facilitando o acesso, a reutilização e a redistribuição das informações por parte dos cidadãos.

3. O repositório de CNPJ

Dentre diversos repositórios existentes no portal "Dados.gov.br", há o repositório de "Cadastro Nacional da Pessoa Jurídica (CNPJ)". Neste repositório estão os dados das empresas e organizações ativas e inativas no Brasil. Atualizado mensalmente, esse banco de dados é mantido pela Secretaria Especial da Receita Federal do Brasil (RFB), e contém

dados essenciais sobre o funcionamento do sistema tributário e empresarial do país.

O CNPJ é emitido pela Receita Federal como uma identificação única para pessoas jurídicas, proporcionando transparência e rastreabilidade das informações relacionadas a essas entidades. No repositório do CNPJ no portal dados.gov.br, é possível acessar informações atualizadas sobre as empresas brasileiras de forma regular.

Esse repositório de dados é extremamente valioso para diversos usuários, incluindo pesquisadores, acadêmicos, empreendedores e desenvolvedores de aplicativos. Pesquisadores e acadêmicos podem utilizar essas informações para realizar estudos socioeconômicos, análises de mercado e pesquisas acadêmicas. Já empreendedores e desenvolvedores podem explorar o CNPJ para obter *insights* sobre o mercado empresarial e construir soluções inovadoras, como também, gerar *leads* por características do CNPJ. Além disso, o público em geral também pode utilizar esses dados para verificar a autenticidade e obter informações sobre empresas específicas.

4. Disponibilização, Modelagem e Carga dos Dados

Neste item, é abordado com mais detalhes a forma que o repositório de CNPJ é disponibilizado no portal e como pode-se realizar o download dele. Será demonstrado também como a modelagem destes dados foi estruturada e como é possível ler estes dados e carregá-lo em um banco de dados relacional.

4.1. Disponibilização dos dados no portal

Acessando o portal “dados.gov.br”, pode-se navegar até o repositório de CNPJ, cujo nome estará "Cadastro Nacional da Pessoa Jurídica - CNPJ". Por ser um dos repositórios mais acessados do portal, ele estará logo na primeira página. A nomenclatura que tem se utilizado neste artigo é “repositório”, e no portal é equivalente a “Conjunto de dados”.

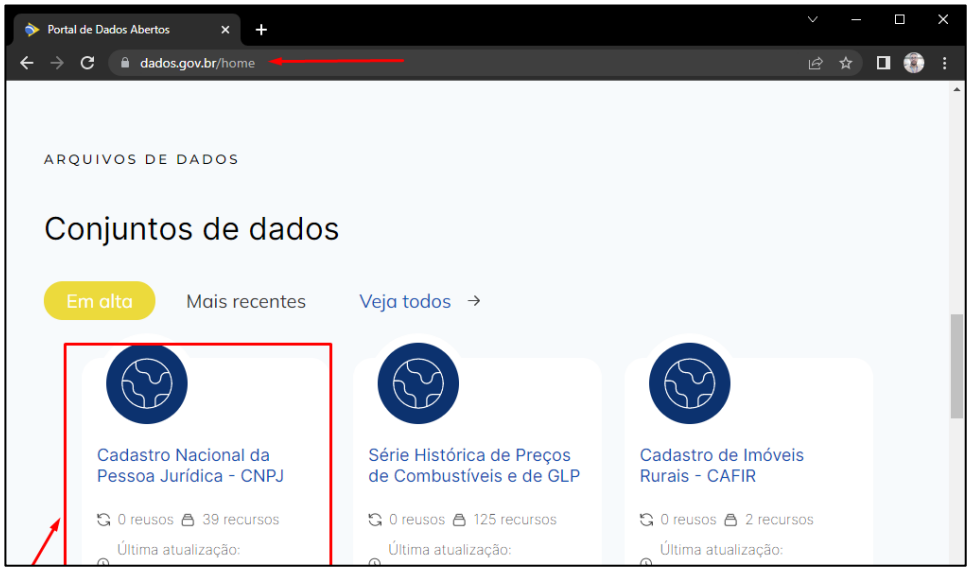

Figura 1. Página inicial do portal “dados.gov.br”, demonstrando que o repositório de CNPJ aparece logo na primeira página.
Fonte: (Autoria própria, 2023)

Dentro de cada conjunto de dados, há o que o portal chama de “Recursos”. Estes

recursos são os arquivos propriamente ditos, onde estão todos os dados referente ao repositório. Na Figura 2 abaixo, pode-se observar que o repositório de CNPJ tem 39 recursos, ou seja, todos os dados referentes a este repositório estão distribuídos em 39 arquivos.

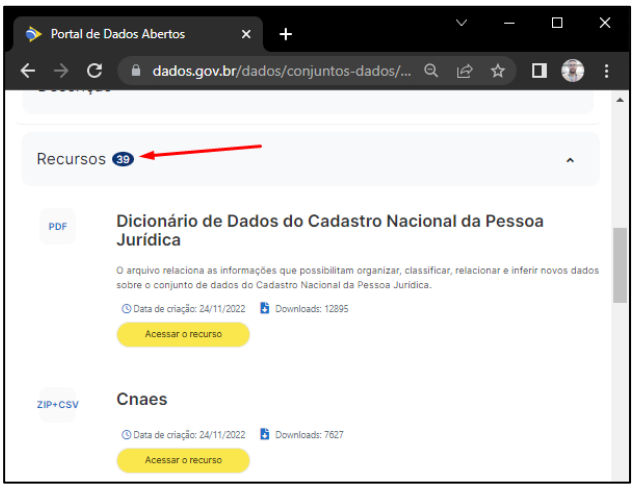

Figura 2. Página do repositório de CNPJ, demonstrando que há 39 recursos disponíveis.
Fonte: (Autoria Própria, 2023)

Para realizar o *download* destes recursos, basta clicar sobre o botão amarelo "Acessar recurso", e então, o arquivo será baixado. Até o momento, não há uma forma de baixar todos os recursos de uma só vez, logo, será necessário clicar em 39 botões para baixar todos os recursos. Outra forma de baixar os dados é acessando diretamente o *link* do servidor do repositório, que é: https://dadosabertos.rfb.gov.br/CNPJ/. Com este *link* direto, é possível criar um simples código que realize a listagem do diretório e efetue o *download* um a um.

4.2. Leitura consolidada dos arquivos com PySpark

Com o *download* de todos os recursos concluído, precisa-se então dar o início ao tratamento destes dados. Na Figura 3 a seguir, pode-se visualizar os recursos baixados, e observar que todos os arquivos estão no formato "ZIP", ou seja, compactados. Dentro de cada arquivo compactado, há outro arquivo no formato "CSV", e é aí que estão os dados. Logo, precisa-se primeiramente descompactar o arquivo para depois realizar a leitura dos dados.

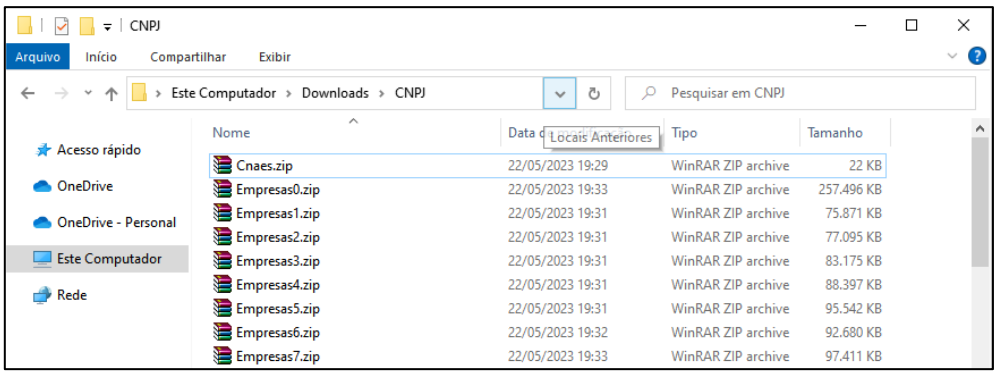

Figura 3. Demonstração dos arquivos do portal em formato compacto.
Fonte: (Autoria Própria, 2023)

Pode-se observar na Figura 3 acima que o mesmo recurso está subdivido em vários arquivos compactados, por exemplo, o recurso “Empresa”, que se refere aos dados “raiz” do CNPJ, estão separados em 10 arquivos compactos, sendo eles: “Empresa0.zip”, “Empresa1.zip”, ..., “Empresa9.zip”. Desta forma, precisa-se descompactar esses conjuntos de arquivos que se referem aos mesmos recursos, e então, realizar a leitura consolidada deles. Faremos isso utilizando *Python* e *Pyspark*.

Segundo Drabas e Lee (2017), no livro "*Learning Spark*", o tratamento de dados utilizando a biblioteca *PySpark* é explorado de forma abrangente e esclarecedora. São abordados conceitos fundamentais, como a criação de *DataFrames*, expressões SQL e a execução distribuída de operações em *cluster*. O livro oferece exemplos práticos e casos de uso reais, demonstrando como aplicar o *PySpark* em diferentes cenários de análise e preparação de dados. Destacam a importância do *PySpark* como uma ferramenta eficiente e escalável para processamento de grandes volumes de dados. Essa obra se torna uma referência indispensável para aqueles que desejam aprofundar seus conhecimentos no tratamento de dados com o uso do *PySpark* (Drabas e Lee, 2017).

A Figura 4, ilustra um exemplo de leitura consolidada, para o conjunto de recursos “Empresas”, utilizando *Python*. Vejamos com mais detalhes todas as bibliotecas utilizadas neste código.

```
1   import os
2   import zipfile
3   import findspark
4   findspark.init()
5   import pyspark
6   from pyspark.sql import SparkSession
7
8   def extrairAquivos(diretorio,priNome,extpara):
9       filtroarquivos = []
10      for diretorio, subpastas, arquivos in os.walk(diretorio):
11          for arquivo in arquivos:
12              if(str(arquivo[:len(priNome)]).upper() == priNome.upper()):
13                  filtroarquivos.append(f'{diretorio}{arquivo}')
14      for arq in filtroarquivos:
15          zipfile.ZipFile(arq,'r').extractall(extpara)
16          print(f'Extraido arquivo: {arq}')
17
18  pasta = 'D:\\Meus Documentos\\CNPJ\\'
19  pastatemp = 'D:\\Meus Documentos\\Temp\\'
20  primNome = 'Empresas'
21  extrairAquivos(pasta,primNome,pastatemp)
22
23  spark = SparkSession.builder \
24      .master('local[*]') \
25      .appName("Carga CNPJ") \
26      .config('spark.ui.port', '4050') \
27      .config("spark.driver.extraClassPath", "C:\\app\\oracle\\19c\\jdbc\\lib\\ojdbc8.jar") \
28      .getOrCreate()
29
30  dados = spark.read.format("csv")\
31                    .option("encoding","iso-8859-1")\
32                    .option("inferSchema","false")\
33                    .option("sep", ";")\
34                    .load(pastatemp)
```

Figura 4. Código fonte para descompactar arquivo e ler CSVs de forma consolidada.
Fonte: (Autoria Própria, 2023)

Da linha 1 até a linha 6 estão as importações de bibliotecas e módulos. A primeira importação, biblioteca “os”, será utilizada para realizar a leitura dos arquivos no diretório do sistema. A “*zipfile*” será utilizada para descompactar os arquivos. A “*findspark*” ajuda a localizar onde o aplicativo “*Spark*” está instalado na máquina. A “*pyspark*” é a biblioteca do “*spark*” disponibilizada no *Python*, que será utilizada para carregar os dados. Na linha

6, realiza-se a importação do módulo "*SparkSession*", que será útil para criação da sessão de *clusters* do *Spark*.

Da linha 8 até a linha 16 está a criação da função "extrairArquivos", que irá descompactar os arquivos selecionados de um diretório especificado. A função pede como argumento o diretório os arquivos compactados estão, qual o nome que se inicia os arquivos, (que em nosso exemplo será "Empresa") e em qual diretório deseja descompactar estes arquivos. Na linha 18, 19 e 20, cria-se as três variáveis exigidas nos argumentos desta função mencionada, e na linha 21, executa-se esta função para descompactar o arquivo.

Da linha 23 até a linha 28 é demonstrado a inicialização do "*SparkSession*", ou seja, iniciará o aplicativo *Spark* dentro do código de aplicação *Python*, tudo isso através da biblioteca *Pyspark*, que realiza a união dessas tecnologias. Essa sessão *Spark* iniciada é atribuída a variável que de nome "*spark*", conforme descrita na linha 23, e a partir desta variável, pode-se executar todos os recursos disponíveis no *Spark*.

A partir da linha 30, há o início então da leitura dos arquivos CSVs que foram descompactados na pasta informada, através da execução da função na linha 21. A leitura destes arquivos está sendo realizada através do método "*read*" do *Spark*, que possibilita a leitura de diversos formatos de arquivo, conforme descrito na documentação (Apache Software Foundation, 2023). Neste caso, como o formato dos arquivos é "CSV", atribuí-se o "CSV" no argumento "*format*". Na linha 34, está informado qual é o diretório que estão os arquivos que devem ser lidos. A leitura consolidada destes arquivos é atribuída a variável de nome "dados", que terá o tipo "*Dataframe*", tipo no qual possibilita diversas análises de dados.

Foi apresentado o exemplo de leitura consolidados com os 10 arquivos do recurso "Empresas", porém, para os demais recursos do repositório, é utilizado o mesmo método para leitura de dados, mudando apenas o nome do recurso. Sendo assim, feita a leitura dos dados, veremos a seguir como foi feita a modelagem dos dados para armazenamento no banco de dados relacional.

4.3. Modelagem dos dados

Neste item, será abordado com mais detalhes todos os recursos disponíveis no repositório, demonstrando toda modelagem dos dados, estruturação de tabelas e os respectivos metadados de cada coluna. No conjunto de dados do portal, há um arquivo disponibilizado com informações sobre os metadados, porém, as informações não são tão precisas, por exemplo, não há descrição do tamanho do campo. Sendo assim, após a leitura consolidada de cada recurso, cria-se a modelagem de dados.

Tabela 1. Consolidação e descrição dos recursos do repositório de CNPJ.

Recurso Consolidado	Descrição do Recurso	Qtd. Recursos
CNAE	Tabela de CNAE - Classificação Nacional de Atividades Econômicas	1
SIMPLES	Tabela de CNPJ optantes pelo regime tributário "Simples Nacional"	1
EMPRESAS	Tabela com dados raiz do CNPJ	10
ESTABELECIMENTOS	Tabela com dados analíticos de cada CNPJ e as respectivas filiais	10
MOTIVOS	Tabela de motivo de baixa de CNPJ	1
MUNICÍPIOS	Tabela de cadastro de municípios brasileiros	1
NATUREZAS	Tabela de cadastro de natureza jurídica para CNPJ	1
PAÍSES	Tabela de cadastro de países do mundo	1
QUALIFICAÇÕES	Tabela de cadastro de tipos de qualificações de sócio	1
REGIMES	Tabela com indicação regime tributário de CNPJ por exercício fiscal	4
SÓCIOS	Tabela com dados de sócios por CNPJ	10
Total	**11**	**41**

Fonte: (Autoria Própria, 2023)

Inicialmente, na Figura 2, demonstrou que havia 39 recursos, porém, na Tabela 1 está demonstrando 41 recursos. Isso é pelo fato de dois recursos serem apenas informativos, não se referem aos dados, e sim ao *layout*, e quatro recursos estavam compactados dentro do recurso “Regimes”, logo, 39 - 2 + 4 = 41.

Além da consolidação dos recursos, criou-se um apelido abreviado para cada um deles, apelido no qual será utilizado para criação das tabelas no banco de dados *Oracle*. Na Tabela 2, é possível visualizar estes apelidos.

Tabela 2. Apelidos de tabela por recurso.

Recurso	Apelido
CNAE	CNA
SIMPLES	SMN
EMPRESAS	EMP
ESTABELECIMENTOS	EST
MOTIVOS	MOT
MUNICÍPIOS	MUN
NATUREZAS	NAT
PAÍSES	PAI
QUALIFICAÇÕES	QUA
REGIMES	RGM
SÓCIOS	SOC
Total	**11**

Fonte: (Autoria Própria, 2023)

A modelagem de dados é uma etapa crucial no processo de desenvolvimento de um banco de dados. Nesse contexto, Helskyaho (2015) fornece orientações práticas e valiosas no livro "*Oracle SQL Developer Data Modeler for Database Design Mastery*". A obra aborda conceitos fundamentais, como entidades, relacionamentos e atributos, e

apresenta técnicas avançadas para otimizar a estrutura de um banco de dados *Oracle*.

Partindo destes princípios de modelagem, a Figura 5 a seguir representa como os recursos foram estruturados no banco de dados. Todas as onze tabelas mencionadas anteriormente foram criadas no banco de dados *Oracle*, com suas respectivas colunas e *datatype*. Para auxiliar no controle de tuplas, em cada tabela também foi adicionada a coluna "REGNUM" e "DELETADO". A coluna de "REGNUM" serve para controlar cada tupla da tabela por um identificado único, é a *Primary Key*. Já a coluna "DELETADO", serve controle de exclusão de tuplas.

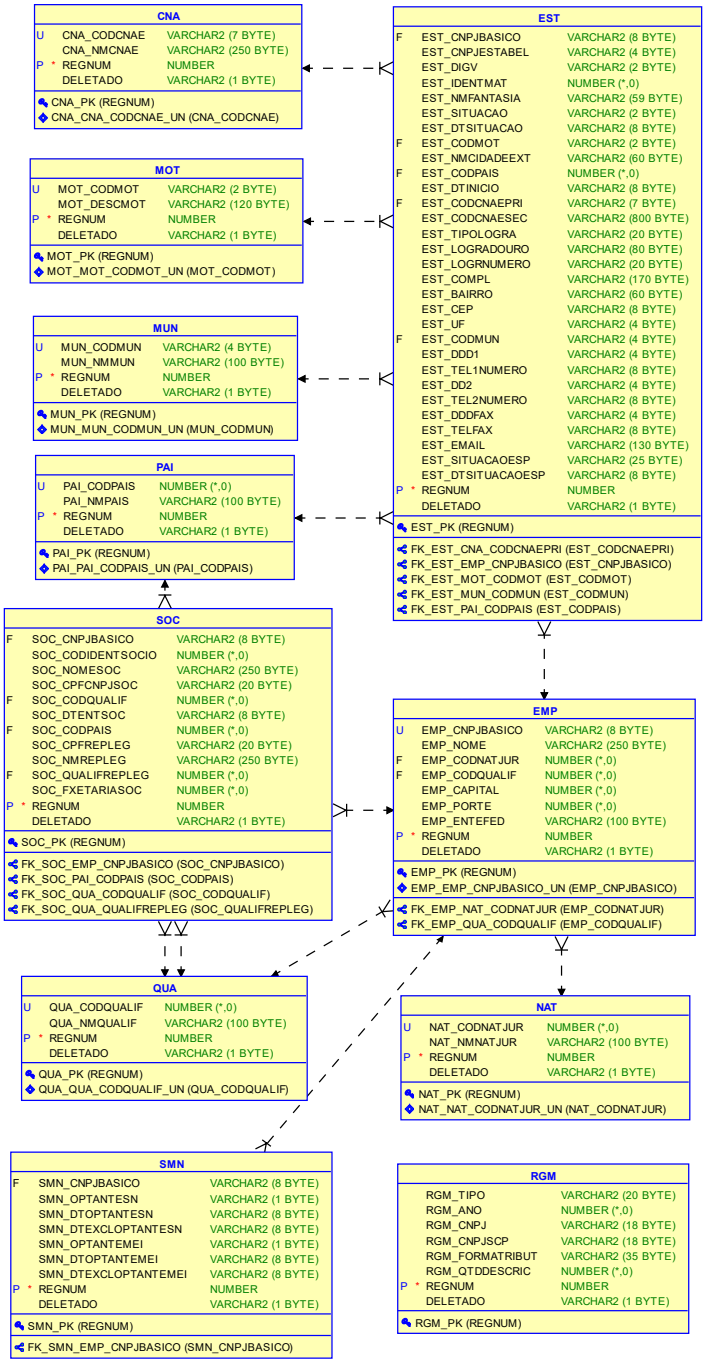

Figura 5. Modelagem completa do banco de dados de CNPJ.
Fonte: (Autoria Própria, 2023)

4.4. Carga dos dados no Oracle através do PySpark

Conforme demonstrado na Figura 4, o código estava até a linha 34, pois foi abordando a leitura dos dados CSV. Agora, na sequência da abordagem do código, será apresentado as linhas referentes a carga de dados no banco de dados *Oracle*.

```
35  from pyspark.sql.functions import lit
36  tabela = "emp"
37  dadosColNames = [
38      'emp_cnpjbasico',
39      'emp_nome',
40      'emp_codnatjur',
41      'emp_codqualif',
42      'emp_capital',
43      'emp_porte',
44      'emp_entefed'
45  ]
46
47  # renomeando colunas do dataframe
48  for index,NomeColuna in enumerate(dadosColNames):
49      dados = dados.withColumnRenamed(f"_c{index}",NomeColuna)
50
51  # adicionando coluna de deletado
52  dados = dados.withColumn("DELETADO",lit(" "))
53
54  # modo append serve para input, sem ele, será criado uma nova tabela no banco
55  dados.write \
56      .format("jdbc") \
57      .option("url", "jdbc:oracle:thin:@127.0.0.1/orclbd") \
58      .option("dbtable", tabela) \
59      .option("user", "cnpgolden") \
60      .option("password", "system") \
61      .mode("append")\
62      .save()
```

Figura 6. Sequência do código fonte da figura 4, carga de dados no Oracle.
Fonte: (Autoria Própria, 2023)

Na linha 35, realiza-se a importação do módulo "lit" das funções do *Spark*. Esse módulo irá auxiliar na criação de uma nova coluna com preenchimento de um caractere específico para todas as tuplas. A variável "tabela" da linha 36, armazena o nome da tabela do banco de dados do *Oracle*. Na linha 37, criamos a lista "dadosColNames", que armazena todas as colunas da tabela, vamos utilizar essa lista na sequência.

As linhas 48 e 49, exibem uma estrutura de repetição que altera a nomenclatura das colunas do *Dataframe* "dados", de acordo com as colunas indicadas na lista "dadosColNames". Essa renomeação de colunas é necessária para que o método "*write*" seja executado com sucesso. Ainda sobre colunas, na linha 52 adiciona-se mais uma coluna no *Dataframe*, que é a coluna "DELETADO", conforme explanamos no item 4.3

Por fim, da linha 55 até 62, tem-se a execução do método "*write*", que é um método do *Dataframe Spark*. Ele serve para gravar os dados do *Dataframe*, tendo diversas possibilidades, como CSV, Parquet, dentre outros. Neste caso, utilizou-se o modo JDBC (*Java Database Connectivity*) para se conectar no banco Oracle e realizar o *insert* do *Dataframe*. As opções de configuração para a escrita são definidas através dos parâmetros

"*option*". O parâmetro "*url*" define qual a URL de conexão com o banco de dados *Oracle*. O "*dbtable*" especifica a tabela de destino onde os dados serão gravados. O "*user*" e "*password*" definem o usuário e a senha para a autenticação no banco de dados. O parâmetro ".*mode*("*append*")" indica que os dados serão anexados à tabela existente, preservando os dados anteriores, sem ele, será criado uma nova tabela. Por fim, o método "*save*" é chamado para iniciar o processo de escrita dos dados no banco de dados *Oracle*, seguindo as parametrizações fornecidas anteriormente.

5. Estatísticas de processamento e volumetria de dados

Finalizado a etapa de carga dos dados, conforme demonstrado a aplicação no item anterior, será demonstrado as estatísticas desse salvamento de dados, exibindo quanto tempo foi gasto para executar cada tratamento de dados e qual a volumetria final atingida.

5.1. Tempo de processamento dos dados

A carga de dados foi realizada separadamente para cada recurso, como havia 11 recursos, logo, foram executadas 11 aplicações isoladamente, seguindo a estrutura do código apresentado nas Figuras 3 e 4.

Tabela 3. Etapas da carga de dados e tempo de processamento.

Etapa	Descrição Etapa	Processamento	Percentual Processamento
1	Descompactando Arquivos	00:03:30	2,00%
2	Inicializando Spark	00:02:46	1,58%
3	Lendo dados	00:01:49	1,04%
4	Verificando Inconsistências Chaves Estrangeiras	00:00:09	0,09%
5	Gravando Log de inconsistências	00:29:53	17,08%
6	Excluindo inconsistências do Dataframe	00:00:02	0,02%
7	Gravando dados no Oracle	02:16:44	78,16%
8	Excluindo arquivos descompactados	00:00:04	0,04%
Total		**02:54:57,00**	**100,00%**

Fonte: (Autoria Própria, 2023)

Dentre as etapas destacadas na Tabela 3 acima, destaca-se a etapa principal, que é a número 7 – "Gravando dados no Oracle". É neste momento que o *Spark* grava os dados no banco relacional *Oracle*. Na Tabela 4, pode-se visualizar o desdobramento desta etapa 7, que totalizou 2 horas, 16 minutos e 44 segundos.

Tabela 4. Processamento da etapa 7 por tabela.

Tabela	Processamento	Percentual Processamento	Linhas Processadas
EST	01:09:20	50,71%	56.520.512
EMP	00:29:49	21,81%	53.671.265
SOC	00:18:18	13,38%	22.775.626
SMN	00:17:42	12,94%	35.333.862
RGM	00:01:22	1,00%	8.706.612
PAI	00:00:03	0,04%	255
QUA	00:00:02	0,02%	68
MOT	00:00:02	0,02%	61
CNA	00:00:02	0,02%	1.359
MUN	00:00:02	0,02%	5.571
NAT	00:00:02	0,02%	90
Total	**02:16:44**	**100,00%**	**177.015.281**

Fonte: (Autoria Própria, 2023)

Dentre todas as tabelas, aquela que mais levou tempo para concluir a carga de dados foi a "EST", que é a tabela de estabelecimentos. Isso ocorreu pelo fato desta tabela ter várias chaves estrangeiras, conforme demonstrado na Figura 5, e outro ponto a se ressaltar é a quantidade de tuplas (linhas) dessa tabela, totalizou 56.520.512 tuplas.

5.2. Volumetria dos dados

Por fim, ao consolidar todos os recursos estruturados no banco de dados, tem-se um total de 177.015.281 tuplas (linhas), conforme demonstrado na Tabela 5 abaixo, e todos esses registros são dados relacionados ao CNPJ de uma empresa, podendo ser utilizado e escalado em qualquer aplicação.

Tabela 5. Volumetria total do repositório de CNPJ.

Tabela	Colunas	Tuplas	Percentual Tuplas
EST	32	56.520.512	31,9297%
EMP	9	53.671.265	30,3201%
SMN	9	35.333.862	19,9609%
SOC	13	22.775.626	12,8665%
RGM	8	8.706.612	4,9186%
MUN	4	5.571	0,0031%
CNA	4	1.359	0,0008%
PAI	4	255	0,0001%
NAT	4	90	0,0001%
QUA	4	68	0,0000%
MOT	4	61	0,0000%
Total	**95**	**177.015.281**	**100%**

Fonte: (Autoria Própria, 2023)

6. Considerações Finais

Em conclusão, o presente artigo abordou o tratamento de dados de CNPJ disponíveis no portal dados.gov.br, utilizando a poderosa biblioteca *PySpark*. Com um volume impressionante de mais de 177 milhões de linhas de dados, divididas em 11 tabelas, o processamento dessas informações foi desafiador, exigindo uma abordagem eficiente e escalável.

A utilização do *PySpark* mostrou-se uma escolha assertiva, permitindo o processamento distribuído dos dados em um *cluster* de computadores, resultando em um desempenho significativamente superior. As funcionalidades avançadas do *PySpark*, como a capacidade de lidar com dados em tempo real e o suporte para operações complexas de transformação e agregação, foram fundamentais para o sucesso desse projeto de tratamento de dados.

Após o processamento, os dados foram salvos no *Oracle Database*, uma solução confiável e robusta para armazenamento de grandes volumes de dados. A escolha dessa ferramenta permitiu o armazenamento e a recuperação eficientes dos dados tratados, além de oferecer recursos adicionais, como a capacidade de consultas SQL complexas e integração com outras ferramentas de análise de dados.

Em suma, o tratamento de dados de CNPJ do portal dados.gov.br foi um desafio significativo, dada a magnitude do volume de informações disponíveis. No entanto, graças à utilização do *PySpark* e à estratégia de processamento distribuído, foi possível lidar com eficiência com o grande volume de dados, realizando transformações e agregações necessárias. O armazenamento desses dados no *Oracle Database* proporcionou uma solução confiável e escalável, permitindo consultas e análises futuras para a obtenção de *insights* valiosos. Esse trabalho destaca a importância do tratamento adequado dos dados para a tomada de decisões informadas e o avanço da análise de dados no contexto empresarial.

A próxima etapa compreende a criação de uma plataforma de consulta sintética de CNPJ, por agrupamento de informações, por exemplo, uma plataforma que possibilite filtrar o total de cadastros por CNAE ou por munícipio, possibilitando a comercialização estruturada dos dados analíticos, caso o usuário tenha interesse.

7. Referências

BRASIL. Ministério do Planejamento. Portal brasileiro de dados abertos: Relato da iniciativa. [Brasília]: Ministério do Planejamento, 20 ago. 2015. Disponível em: https://www.gov.br/governodigital/pt-br/dados-abertos/portalbrasileirodadosabertos.pdf. Acesso em: 22 mai. 2023.

BRASIL. Lei nº 12.527, de 18 de novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5º, no inciso II do § 3º do art. 37 e no § 2º do art. 216 da Constituição Federal; altera a Lei nº 8.112, de 11 de dezembro de 1990; revoga a Lei nº 11.111, de 5 de maio de 2005, e dispositivos da Lei nº 8.159, de 8 de janeiro de 1991; e dá outras providências. Diário Oficial da União, Brasília, DF, 18 nov. 2011. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm. Acesso em: 22 mai. 2023.

BRASIL. Decreto nº 8.777, de 11 de maio de 2016. Regulamenta a Lei nº 12.527, de 18 de novembro de 2011, que dispõe sobre o acesso a informações previsto no inciso

XXXIII do caput do art. 5º, no inciso II do § 3º do art. 37 e no § 2º do art. 216 da Constituição. Diário Oficial da União, Brasília, DF, 11 maio 2016. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2016/decreto/d8777.htm. Acesso em: 22 mai. 2023.

BRASIL. Ministério da Economia. Conjunto de Dados: Cadastro Nacional da Pessoa Jurídica - CNPJ. [Brasília]: Ministério da Economia, 24 nov. 2022. Disponível em: https://dados.gov.br/dados/conjuntos-dados/cadastro-nacional-da-pessoa-juridica---cnpj. Acesso em: 22 mai. 2023.

LEE, Denny; DRABAS, Tomasz. Learning PySpark: Build data-intensive applications locally and deploy at scale using the combined powers of Python and Spark. Sebastopol, CA: Packt Publishing, 2017. 274 p. ISBN 978-1786463708.

APACHE Software Foundation. Apache Spark - Documentation. Disponível em: https://spark.apache.org/documentation.html. Acesso em: 23 maio 2023.

HELSKYAHO, Heli. Oracle SQL Developer Data Modeler: for Database Design Mastery. 1. ed. [S.l.]: McGraw-Hill Education, 2015. 366 p. ISBN 978-0071850094.