



Integração de diferentes bancos de dados via Python

João Victor Guimarães de Oliveira

Leila Moreira Gomes Roque

Robson Motta

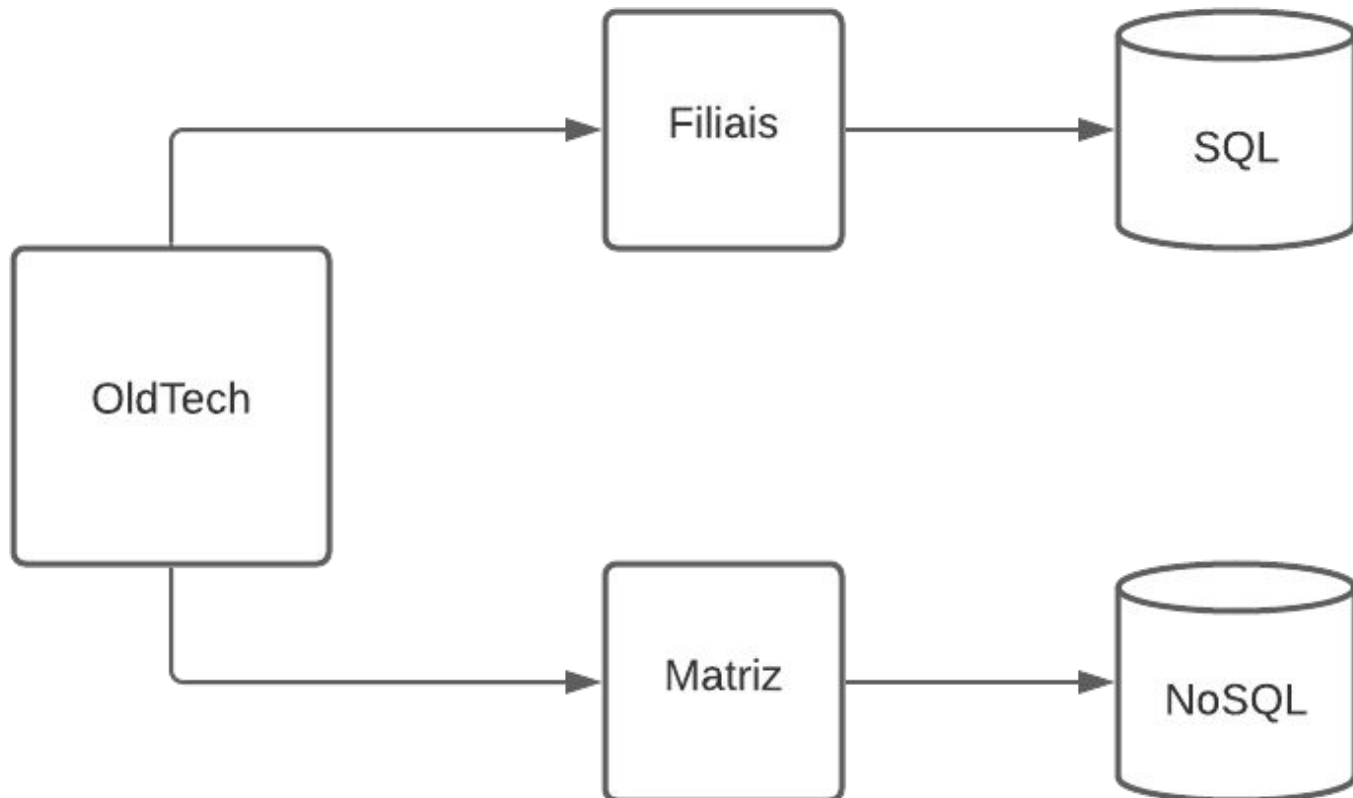
SoulCode Academy

BC8 - Engenharia de Dados

13 de dezembro de 2021


Problema

A empresa **OLDTech Ltda** utiliza dois sistemas distintos para cadastro de vendas, um em sua matriz e outro em suas filiais. O sistema da matriz utiliza um banco de dados NoSQL enquanto os das filiais ainda utilizam um SQL



Amostra dos dados

- Dois arquivos .csv com as seguintes colunas
 - nota_fiscal
 - vendedor
 - total
- Amostra 'vendedor' com campos nulos
- 2000 dados no total

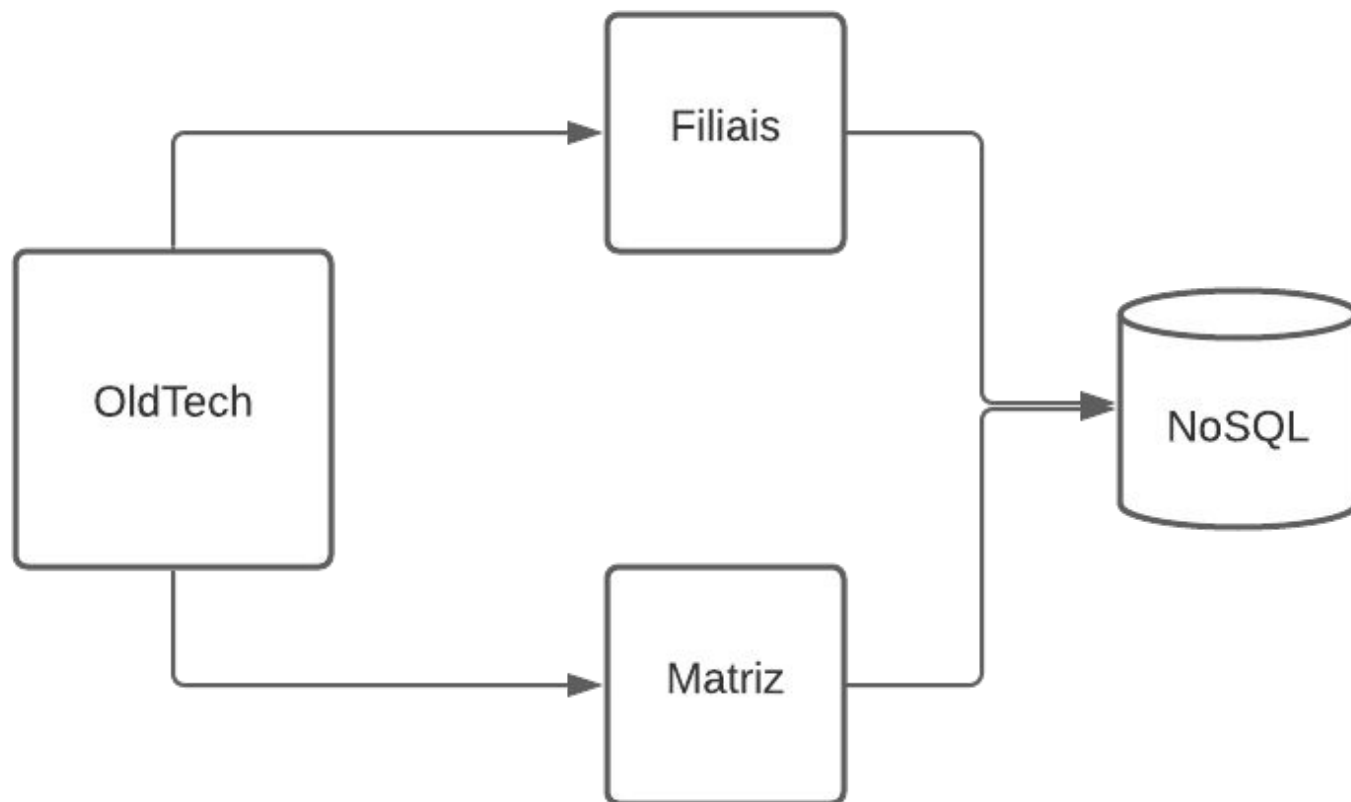


	A	B
1	nota_fiscal,vendedor,total	
2	1,Clerkclaude,260.85	
3	2,Karly,217.45	
4	3,Xerxes,462.32	
5	4,Kaia,290.52	
6	5,Myron,518.35	
7	6,Eddy,320.92	
8	7,Suellen,464.28	
9	8,Doreen,406.99	
10	9,Mira,579.04	
11	10,Waring,225.81	
12	11,Ardelia,574.07	
13	12,Ariadne,396.09	
14	13,Johnathon,369.16	
15	14,Joline,349.68	
16	15,Brianne,326.26	
17	16,Elvin,336.53	
18	17,Demetre,452.96	
19	18,Karie,251.07	
20	19,Dimitry,447.47	
21	20,Julina,146.43	
22	21,Felicia,142.56	
23	22,Wikki,259.26	



Solução

Realizar a leitura do banco de dados da filial, corrigir, padronizar e inserir os dados no banco de dado da matriz





Interface de conexão com os bancos de dados

- Criadas duas classes para a conexão com os bancos de dados

```
import mysql.connector

class Interface_db_mysql():

    usuario, senha, host, banco = "", "", "", ""

    def __init__(self, usuario, senha, host, banco):
        """Construtor da classe Interface_db_mysql

        Args:
            usuario (string): usuario do banco
            senha (string): senha de acesso ao banco
            host (string): ip de acesso ao banco
            banco (string): nome do banco
        """
        try:
            self.usuario = usuario
            self.senha = senha
            self.host = host
            self.banco = banco
        except Exception as e:
            print(str(e))
```

```
from cassandra.cluster import Cluster

class Interface_db_cassandra():
    cluster = ""
    session = ""

    def __init__(self, database = "soulcode"):
        """Construtor da classe Interface_db_cassandra

        Args:
            database (string, optional): nome da database. Defaults to "soulcode".
        """
        try:
            self.cluster = Cluster()
            self.set_session(database)
        except Exception as e:
            print(str(e))

    def set_session(self, database):
        """Realiza a conexão com a database

        Args:
            database (string): nome da database
        """
        try:
            self.session=self.cluster.connect(database)
        except Exception as e:
            print(str(e))
```



Interface de conexão com os bancos de dados

- Métodos exclusivos de cada classe

```
def conectar(self):
    """Função genérica para conectar ao banco

    Returns:
        con : conector mysql
        cursor : cursor para leitura do banco
    """
    try:
        con = mysql.connector.connect(user = self.usuario, password = self.senha,
                                      host = self.host, database=self.banco)
        cursor = con.cursor()
        return con, cursor
    except Exception as e:
        print(str(e))

def desconectar(self, con, cursor):
    """Função genérica para desconectar do banco

    Args:
        con : conector mysql
        cursor : cursor para leitura do banco
    """
    try:
        cursor.close()
        con.commit()
        con.close()
    except Exception as e:
        print(str(e))
```

```
def fetchall(self, dados):
    """Função que retorna uma lista a partir
    de um objeto da classe Cluster

    Args:
        dados (Object Cluster): Objeto cluster
        com os dados buscados do banco de dados

    Returns:
        list : Lista com os dados lidos do banco de dados
    """
    try:
        lista = []
        for d in dados:
            lista.append(d)
        return lista
    except Exception as e:
        print(str(e))
```



Interface de conexão com os bancos de dados

- Métodos de consulta dos dados

```
def buscar(self, query):  
    """Função genérica para uma consulta no banco de dados  
  
    Args:  
        query (string): Query de busca  
    Returns:  
        cursor.fetchall(): retorna tudo que for encontrado  
        pelo cursor  
    """  
    try:  
        con, cursor = self.conectar()  
        cursor.execute(query)  
        return cursor.fetchall()  
    except Exception as e:  
        print(str(e))  
    finally:  
        self.desconectar(con, cursor)
```

```
def buscar(self, query):  
    """Função genérica para consulta de dados no  
    banco de dados  
  
    Args:  
        query (string): query SQL completa para  
        consultas no banco de dados  
    Returns:  
        list: lista com todos os dados lidos do  
        banco de dados  
    """  
    try:  
        dados = self.session.execute(query)  
        lista = self.fetchall(dados)  
        return lista  
    except Exception as e:  
        print(str(e))
```




Interface de conexão com os bancos de dados

- Método de inserção de dados

```
def inserir(self, query):
    """Função genérica para inserir um dado no
    | banco de dados

    Args:
    | query (string): Query de inserção
    Returns:
    | cursor.fetchall(): retorna tudo que for
    | encontrado pelo cursor
    """
    try:
        con, cursor = self.conectar()
        cursor.execute(query)
        return cursor.fetchall()
    except Exception as e:
        print(str(e))
    finally:
        self.desconectar(con, cursor)
```

```
def inserir(self, query):
    """Função genérica para inserção de dados no
    | banco de dados

    Args:
    | query (string): query SQL completa para inserir
    | dados no banco de dados
    """
    try:
        self.session.execute(query)
    except Exception as e:
        print(str(e))
```




Interface de conexão com os bancos de dados

- Método de atualização dos dados

```
def atualizar(self, query):
    """Função genérica para alterar um dado no
    | banco de dados

    Args:
    | query (string): query de atualização
    Returns:
    | cursor.fetchall(): retorna tudo que for
    | encontrado pelo cursor
    """
    try:
        con, cursor = self.conectar()
        cursor.execute(query)
        return cursor.fetchall()
    except Exception as e:
        print(str(e))
    finally:
        self.desconectar(con, cursor)
```

```
def atualizar(self, query):
    """Função genérica para atualização de dados
    | no banco de dados

    Args:
    | query (string): query SQL completa para
    | atualizar dados no banco de dados
    """
    try:
        self.session.execute(query)
    except Exception as e:
        print(str(e))
```



Interface de conexão com os bancos de dados

- Método de exclusão dos dados

```
def deletar(self, query):  
    """Função genérica para um delete de algum dado  
    no banco de dados  
  
    Args:  
        query (string): query de inserção  
    Returns:  
        cursor.fetchall(): retorna tudo que for  
        encontrado pelo cursor  
    """  
    try:  
        con, cursor = self.conectar()  
        cursor.execute(query)  
        return cursor.fetchall()  
    except Exception as e:  
        print(str(e))  
    finally:  
        self.desconectar(con, cursor)
```

```
def deletar(self, query):  
    """Função genérica para deleção de dados no  
    banco de dados  
  
    Args:  
        query (string): query SQL completa para  
        deletar dados no banco de dados  
    """  
    try:  
        self.session.execute(query)  
    except Exception as e:  
        print(str(e))
```



Resolução do problema

- Importação dos conectores dos bancos e da biblioteca pandas
- Leitura dos arquivos .csv via pandas

```
from modules.conector_cassandra import Interface_db_cassandra
from modules.conector_mysql import Interface_db_mysql
import pandas as pd

if __name__ == "__main__":
    try:
        #Importando os dados do arquivo CSV e salvando em um DataFrame
        Sistema_A_SQL = pd.read_csv("Sistema_A_SQL.csv", sep=",")
        Sistema_B_NoSQL = pd.read_csv("Sistema_B_NoSQL.csv", sep=",")
```



Resolução do problema

- Realiza o tratamento dos dados
- Instância os objetos de conexão com os bancos de dados

```
#Excluindo a coluna nota_fiscal do DataFrame para inserção nos databases
Sistema_A_SQL.drop(['nota_fiscal'], axis=1, inplace=True)
Sistema_B_NoSQL.drop(['nota_fiscal'], axis=1, inplace=True)

#Excluindo as linhas que contem campos vazios para inserção no database SQL
Sistema_A_SQL = Sistema_A_SQL.dropna()

#Instancia a interface para comunicação com os bancos de dados/keyspace já criados anteriormente
interface_mysql = Interface_db_mysql("root","root","127.0.0.1","OldTech_Ltda")
interface_cassandra = Interface_db_cassandra('oldtech_ltda')
```



Resolução do problema

- Insere os dados tratados nos bancos de dados SQL e NoSQL

```
#Insere os dados do DataFrame Sistema_A_SQL no banco de dados SQL - MySQL
for index, row in Sistema_A_SQL.iterrows():
    query=f"INSERT INTO vendas (nome_vendedor, total) VALUES ('{row['vendedor']}', {int(row['total'])})"
    interface_mysql.inserir(query)

#Insere os dados do DataFrame Sistema_B_NoSQL no banco de dados NoSQL - Cassandra, verificando se existe algum campo nulo
for index, row in Sistema_B_NoSQL.iterrows():
    if pd.isnull( row['vendedor'] ):
        query=f"INSERT INTO vendas (nota_fiscal, total) VALUES ({'uuid()'}, {int(row['total'])})"
        interface_cassandra.inserir(query)
    elif pd.isnull( row['total'] ):
        query=f"INSERT INTO vendas (nota_fiscal, nome_vendedor) VALUES ({'uuid()'}, {int(row['vendedor'])})"
        interface_cassandra.inserir(query)
    else:
        query=f"INSERT INTO vendas (nota_fiscal, nome_vendedor, total) VALUES ({'uuid()'}, '{row['vendedor']}', {int(row['total'])})"
        interface_cassandra.inserir(query)
```



Resolução do problema

- Busca os dados no banco de dados SQL e armazena em um DataFrame
- Insere os dados buscados no banco de dados NoSQL

```
#Consulta os dados do banco MySQL salvando em um DataFrame
df_vendas = pd.DataFrame( interface_mysql.buscar("select * from vendas") )
df_vendas.columns = ['nota_fiscal','vendedor','total']

#Insere os dados lidos do banco de dados SQL no banco de dados NoSQL
for index, row in df_vendas.iterrows():
    query=f"INSERT INTO vendas (nota_fiscal, nome_vendedor, total) VALUES ({'uuid()'},{row['vendedor']}, {int(row['total'])})"
    interface_cassandra.inserir(query)

print("Fim da execução!")

except Exception as e:
    print(str(e))
```


CRONOGRAMA DO PROJETO

INTEGRAÇÃO DE DIFERENTES DATABASES VIA PYTHON

TÍTULO DO PROJETO	Atividade 16	NOME DA EMPRESA	OLDTech Ltda
RESPONSÁVEIS DO PROJETO	João Victor Guimarães de Oliveira Leila Moreira Gomes Roque Robson Motta	DATA	13/12/21

FASE		DETALHES	10/12/2021	11/12/2021	12/12/2021	13/12/2021
1	Elaboração e início do projeto	- Avaliação do plano - Início				ENTREGA DO PROJETO
2	Definição e plano do projeto	- Definição de ferramentas (Linguagem e SGBD) - Programação da divisão do trabalho				
3	Lançamento e execução do projeto	- Criação do banco de dados no MySQL / Workbench - Criação do banco de dados NoSQL - Cassandra - Leitura dos dados dos arquivos CSV - Salvar os arquivos CSV em Dataframe via biblioteca Pandas - Tratamento dos dados - Inserção dos dados no MySQL e no Cassandra - Consulta dos dados no MySQL e retorno deles em Dataframe - Inserção dos dados do banco MySQL no Cassandra				
4	Controle e desempenho do projeto	- Alcance do objetivo - Resultados de qualidade				
5	Fechamento do projeto	- Verificação do projeto - Apresentação				



Obrigado!