Instituto Politécnico de Castelo Branco

Escola Superior de Tecnologia

Mestrado em Engenharia de Software

Data Warehousing (M EINF DSSI)

Docente: Filipe Fidalgo

Ano Letivo 2024/25

Grupo 04:

Alex Seles

Eric Lima

Evaldina Graça

Gerson Muel

Rafael Jesus

Janeiro de 2025

Relatório do TP2 de DW

Implementação da DW

1. Modelo de base de dados de DW

A diagram of a computer

Description automatically generated with medium confidence

O modelo que foi utilizado é sobre o sistema de vendas de uma empresa, onde encontramos o Fato de Vendas e suas dimensões: Tempo, Cliente, Vendedor e Produto.

DML da criação das tabelas:

-- Script para criação das tabelas no SQL Server

-- Dimensão Cliente

CREATE TABLE Dim\_Cliente (

ID\_Cliente INT PRIMARY KEY IDENTITY(1,1),

Nome\_Cliente NVARCHAR(100) NOT NULL,

Data\_Nascimento DATE,

Genero CHAR(1),

Categoria\_Cliente NVARCHAR(50),

Cidade NVARCHAR(100),

Pais NVARCHAR(50)

);

-- Dimensão Produto

CREATE TABLE Dim\_Produto (

ID\_Produto INT PRIMARY KEY IDENTITY(1,1),

Nome\_Produto NVARCHAR(100) NOT NULL,

Categoria\_Produto NVARCHAR(50),

Subcategoria\_Produto NVARCHAR(50),

Marca NVARCHAR(50),

Preco\_Unitario DECIMAL(10, 2),

Data\_Lancamento DATE

);

-- Dimensão Tempo

CREATE TABLE Dim\_Tempo (

ID\_Tempo INT PRIMARY KEY IDENTITY(1,1),

Data DATE NOT NULL,

Dia INT NOT NULL,

Mes INT NOT NULL,

Ano INT NOT NULL,

Trimestre INT NOT NULL,

Semana INT NOT NULL

);

-- Dimensão Vendedor

CREATE TABLE Dim\_Vendedor (

ID\_Vendedor INT PRIMARY KEY IDENTITY(1,1),

Nome\_Vendedor NVARCHAR(100) NOT NULL,

Regiao NVARCHAR(50),

Data\_Contratacao DATE

);

-- Fato Vendas

CREATE TABLE Fato\_Vendas (

ID\_Venda INT PRIMARY KEY IDENTITY(1,1),

ID\_Cliente INT NOT NULL,

ID\_Produto INT NOT NULL,

ID\_Tempo INT NOT NULL,

ID\_Vendedor INT NOT NULL,

Quantidade INT NOT NULL,

Valor\_Unitario DECIMAL(10, 2) NOT NULL,

Valor\_Total AS (Quantidade \* Valor\_Unitario) PERSISTED,

Desconto DECIMAL(10, 2) DEFAULT 0.0,

Receita\_Liquida AS ((Quantidade \* Valor\_Unitario) - Desconto) PERSISTED,

Metodo\_Pagamento NVARCHAR(50),

Status\_Venda NVARCHAR(50),

CONSTRAINT FK\_Cliente FOREIGN KEY (ID\_Cliente) REFERENCES Dim\_Cliente(ID\_Cliente),

CONSTRAINT FK\_Produto FOREIGN KEY (ID\_Produto) REFERENCES Dim\_Produto(ID\_Produto),

CONSTRAINT FK\_Tempo FOREIGN KEY (ID\_Tempo) REFERENCES Dim\_Tempo(ID\_Tempo),

CONSTRAINT FK\_Vendedor FOREIGN KEY (ID\_Vendedor) REFERENCES Dim\_Vendedor(ID\_Vendedor)

);

Views Utilizadas para extração do Gráfico no Power BI

CREATE VIEW [dbo].[Vw\_Participacao\_Categoria] AS

SELECT

DT.Ano,

DP.Categoria\_Produto,

SUM(FV.Receita\_Liquida) AS Receita\_Liquida\_Categoria,

SUM(SUM(FV.Receita\_Liquida)) OVER (PARTITION BY DT.Ano) AS Receita\_Liquida\_Total\_Ano,

ROUND(

SUM(FV.Receita\_Liquida) \* 100.0 / SUM(SUM(FV.Receita\_Liquida)) OVER (PARTITION BY DT.Ano),

2

) AS Participacao\_Percentual

FROM

Fato\_Vendas FV

INNER JOIN

Dim\_Produto DP ON FV.ID\_Produto = DP.ID\_Produto

INNER JOIN

Dim\_Tempo DT ON FV.ID\_Tempo = DT.ID\_Tempo

GROUP BY

DT.Ano,

DP.Categoria\_Produto

CREATE VIEW [dbo].[Vw\_Receita\_Agregada] AS

SELECT

DT.Ano,

DT.Trimestre,

DP.Categoria\_Produto,

SUM(FV.Receita\_Liquida) AS Receita\_Liquida\_Total

FROM

Fato\_Vendas FV

INNER JOIN

Dim\_Produto DP ON FV.ID\_Produto = DP.ID\_Produto

INNER JOIN

Dim\_Tempo DT ON FV.ID\_Tempo = DT.ID\_Tempo

GROUP BY

GROUPING SETS (

(DT.Ano, DT.Trimestre, DP.Categoria\_Produto), -- Receita por Ano, Trimestre e Categoria

(DT.Ano, DT.Trimestre), -- Receita por Ano e Trimestre

(DT.Ano, DP.Categoria\_Produto), -- Receita por Ano e Categoria

(DT.Ano), -- Receita por Ano

(DP.Categoria\_Produto), -- Receita por Categoria

() -- Total Geral

)

;

CREATE VIEW [dbo].[Vw\_Receita\_Agrupada] AS

SELECT

DT.Ano,

DT.Trimestre,

DP.Categoria\_Produto,

SUM(FV.Receita\_Liquida) AS Receita\_Liquida\_Total

FROM

Fato\_Vendas FV

INNER JOIN

Dim\_Produto DP ON FV.ID\_Produto = DP.ID\_Produto

INNER JOIN

Dim\_Tempo DT ON FV.ID\_Tempo = DT.ID\_Tempo

GROUP BY

GROUPING SETS (

(DT.Ano, DT.Trimestre, DP.Categoria\_Produto), -- Receita por Ano, Trimestre e Categoria

(DT.Ano, DT.Trimestre), -- Receita por Ano e Trimestre

(DT.Ano, DP.Categoria\_Produto), -- Receita por Ano e Categoria

(DT.Ano), -- Receita por Ano

(DP.Categoria\_Produto), -- Receita por Categoria

() -- Total Geral

)

;

CREATE VIEW [dbo].[Vw\_Receita\_Cubo] AS

SELECT

DT.Ano,

DP.Categoria\_Produto,

FV.Metodo\_Pagamento,

SUM(FV.Receita\_Liquida) AS Receita\_Liquida\_Total

FROM

Fato\_Vendas FV

INNER JOIN

Dim\_Produto DP ON FV.ID\_Produto = DP.ID\_Produto

INNER JOIN

Dim\_Tempo DT ON FV.ID\_Tempo = DT.ID\_Tempo

GROUP BY

CUBE(DT.Ano, DP.Categoria\_Produto, FV.Metodo\_Pagamento)

CREATE VIEW [dbo].[Vw\_Receita\_Vendedores\_Recentes] AS

WITH Vendedores\_Recente AS (

SELECT

ID\_Vendedor,

Nome\_Vendedor,

Data\_Contratacao

FROM

Dim\_Vendedor

WHERE

Data\_Contratacao >= DATEADD(YEAR, -5, GETDATE())

),

Vendas\_Filtradas AS (

SELECT

FV.ID\_Produto,

FV.ID\_Tempo,

FV.Receita\_Liquida,

DP.Categoria\_Produto,

DT.Ano

FROM

Fato\_Vendas FV

INNER JOIN

Dim\_Produto DP ON FV.ID\_Produto = DP.ID\_Produto

INNER JOIN

Dim\_Tempo DT ON FV.ID\_Tempo = DT.ID\_Tempo

INNER JOIN

Vendedores\_Recente VR ON FV.ID\_Vendedor = VR.ID\_Vendedor

)

SELECT

Ano,

Categoria\_Produto,

SUM(Receita\_Liquida) AS Receita\_Liquida\_Total

FROM

Vendas\_Filtradas

GROUP BY

Ano,

Categoria\_Produto

;

Gráficos Gerados no Power BI

A screenshot of a computer

Description automatically generated

Os dados utilizados nos nossos exemplos foram criados utilizando um script em python

# script para criar os ficheiros com os dados

import pandas as pd

import numpy as np

# Configurações gerais

num\_clientes = 100

num\_produtos = 50

num\_vendedores = 20

num\_dias = (365 \* 4) + 2

num\_vendas = 15000

# Geração da Dimensão Cliente

clientes = {

    "ID\_Cliente": np.arange(1, num\_clientes + 1),

    "Nome\_Cliente": [f"Cliente\_{i}" for i in range(1, num\_clientes + 1)],

    "Data\_Nascimento": pd.date\_range(start="1970-01-01", periods=num\_clientes, freq="M"),

    "Genero": np.random.choice(["M", "F"], size=num\_clientes),

    "Categoria\_Cliente": np.random.choice(["Regular", "VIP"], size=num\_clientes),

    "Cidade": np.random.choice(["Lisboa", "Porto", "Castelo Branco", "Fátima"], size=num\_clientes),

    "Pais": ["Portugal"] \* num\_clientes

}

df\_clientes = pd.DataFrame(clientes)

# Geração da Dimensão Produto

produtos = {

    "ID\_Produto": np.arange(1, num\_produtos + 1),

    "Nome\_Produto": [f"Produto\_{i}" for i in range(1, num\_produtos + 1)],

    "Categoria\_Produto": np.random.choice(["Eletrônicos", "Roupas", "Alimentos"], size=num\_produtos),

    "Subcategoria\_Produto": np.random.choice(["Premium", "Básico"], size=num\_produtos),

    "Marca": [f"Marca\_{i}" for i in range(1, num\_produtos + 1)],

    "Preco\_Unitario": np.random.uniform(10.0, 1000.0, num\_produtos).round(2),

    "Data\_Lancamento": pd.date\_range(start="2020-01-01", periods=num\_produtos, freq="M")

}

df\_produtos = pd.DataFrame(produtos)

# Geração da Dimensão Tempo

data\_calculada = pd.date\_range(start="2020-01-01", periods=num\_dias, freq="D")

tempo = {

    "ID\_Tempo": np.arange(1, num\_dias + 1),

    "Data": data\_calculada,

    "Dia": data\_calculada.day,

    "Mes": data\_calculada.month,

    "Ano": data\_calculada.year,

    "Trimestre": data\_calculada.quarter,

     "Semana": data\_calculada.isocalendar().week.astype(int)

}

df\_tempo = pd.DataFrame(tempo)

# Geração da Dimensão Vendedor

vendedores = {

    "ID\_Vendedor": np.arange(1, num\_vendedores + 1),

    "Nome\_Vendedor": [f"Vendedor\_{i}" for i in range(1, num\_vendedores + 1)],

    "Regiao": np.random.choice(["Norte", "Centro", "Algarve"], size=num\_vendedores),

    "Data\_Contratacao": pd.date\_range(start="2015-01-01", periods=num\_vendedores, freq="Y")

}

df\_vendedores = pd.DataFrame(vendedores)

# Geração da Fato Vendas

vendas = {

    "ID\_Venda": np.arange(1, num\_vendas + 1),

    "ID\_Cliente": np.random.choice(df\_clientes["ID\_Cliente"], size=num\_vendas),

    "ID\_Produto": np.random.choice(df\_produtos["ID\_Produto"], size=num\_vendas),

    "ID\_Tempo": np.random.choice(df\_tempo["ID\_Tempo"], size=num\_vendas),

    "ID\_Vendedor": np.random.choice(df\_vendedores["ID\_Vendedor"], size=num\_vendas),

    "Quantidade": np.random.randint(1, 10, num\_vendas),

    "Valor\_Unitario": np.random.uniform(10.0, 1000.0, num\_vendas).round(2),

    "Desconto": np.random.uniform(0.0, 50.0, num\_vendas).round(2),

    "Metodo\_Pagamento": np.random.choice(["Cartão de Crédito", "Paypal", "MBWay", "Dinheiro"], size=num\_vendas),

    "Status\_Venda": np.random.choice(["Concluída", "Pendente", "Cancelada"], size=num\_vendas)

}

df\_vendas = pd.DataFrame(vendas)

df\_vendas["Valor\_Total"] = (df\_vendas["Quantidade"] \* df\_vendas["Valor\_Unitario"]).round(2)

df\_vendas["Receita\_Liquida"] = (df\_vendas["Valor\_Total"] - df\_vendas["Desconto"]).round(2)

# Salvando os dados em arquivos CSV (opcional)

df\_clientes.to\_csv("Dim\_Cliente.csv", index=False)

df\_produtos.to\_csv("Dim\_Produto.csv", index=False)

df\_tempo.to\_csv("Dim\_Tempo.csv", index=False)

df\_vendedores.to\_csv("Dim\_Vendedor.csv", index=False)

df\_vendas.to\_csv("Fato\_Vendas.csv", index=False)

# Exibindo os primeiros registros de cada tabela

print("Clientes:\n", df\_clientes.head())

print("Produtos:\n", df\_produtos.head())

print("Tempo:\n", df\_tempo.head())

print("Vendedores:\n", df\_vendedores.head())

print("Vendas:\n", df\_vendas.head())

# script para importar os ficheiros para as tabelas de DW

import os

import pandas as pd

from dotenv import load\_dotenv

import pymssql

# Carregar variáveis de ambiente do arquivo .env

load\_dotenv()

def conectar\_bd():

"""Conecta ao banco de dados usando as configurações do arquivo .env."""

server = os.getenv("DB\_SERVER")

user = os.getenv("DB\_USER")

password = os.getenv("DB\_PASSWORD")

database = os.getenv("DB\_NAME")

try:

conn = pymssql.connect(server, user, password, database)

return conn

except pymssql.InterfaceError as e:

print(f"Erro de conexão: {e}")

return None

# Função para carregar CSV e inserir dados no banco de dados

def carregar\_dados(tabela, arquivo\_csv, colunas\_excluir=None):

"""

Carrega os dados de um arquivo CSV e insere na tabela do banco de dados.

:param tabela: Nome da tabela no banco de dados.

:param arquivo\_csv: Caminho do arquivo CSV.

:param colunas\_excluir: Lista de colunas a serem excluídas do insert.

"""

conn = conectar\_bd()

if not conn:

return

cursor = conn.cursor()

try:

# Habilitar inserção em colunas identity

cursor.execute(f"SET IDENTITY\_INSERT {tabela} ON")

# Carregar dados do CSV

df = pd.read\_csv(arquivo\_csv)

# Remover colunas calculadas

if colunas\_excluir:

df = df.drop(columns=colunas\_excluir, errors='ignore')

# Gerar a string de inserção dinâmica

colunas = ", ".join(df.columns)

valores = ", ".join(["%s"] \* len(df.columns))

query = f"INSERT INTO {tabela} ({colunas}) VALUES ({valores})"

# Inserir dados linha a linha

for \_, row in df.iterrows():

cursor.execute(query, tuple(row))

conn.commit()

print(f"Dados inseridos na tabela {tabela} com sucesso.")

# Desabilitar inserção em colunas identity

cursor.execute(f"SET IDENTITY\_INSERT {tabela} OFF")

except Exception as e:

print(f"Erro ao inserir dados na tabela {tabela}: {e}")

conn.rollback()

finally:

cursor.close()

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

# Exemplo de carga para cada tabela

carregar\_dados("Dim\_Cliente", "Dim\_Cliente.csv")

carregar\_dados("Dim\_Produto", "Dim\_Produto.csv")

carregar\_dados("Dim\_Tempo", "Dim\_Tempo.csv")

carregar\_dados("Dim\_Vendedor", "Dim\_Vendedor.csv")

carregar\_dados("Fato\_Vendas", "Fato\_Vendas.csv", colunas\_excluir=["Valor\_Total", "Receita\_Liquida"])

todos os codigos que foram criados para este trabalho encontram-se disponiveis em:

<https://github.com/ericlima/etl_dados_trabalho_dw>

* CTE

"Qual é a receita líquida total por ano e categoria de produto, considerando apenas as vendas realizadas por vendedores contratados nos últimos 5 anos?"

Utilizamos a view : [Vw\_Receita\_Vendedores\_Recentes]

* ROLLUP

"Qual é a receita líquida total por ano, trimestre e categoria de produto, incluindo totais acumulados por ano e categoria?"

Utilizamos a view : [Vw\_Receita\_Agregada]

* CUBE

"Qual é a receita líquida total por ano, categoria de produto e método de pagamento, incluindo todas as combinações possíveis de totais acumulados?"

Utilizamos a view : [Vw\_Receita\_Agrupada]

* GROUPING SETS

"Qual é a receita líquida total por ano, trimestre e categoria de produto, mas também os totais separados por ano, por categoria e os totais gerais?"

Utilizamos a view : [Vw\_Receita\_Vendedores\_Recentes]