**UNIVERSIDADE FEDERAL DO PARANÁ**



**ALEXANDRE DIANO DE SOUZA**

**ARTHUR GIAN A. DE OLIVEIRA**

**NICHOLAS MANOEL DE P. ANUNES**

**GABRIEL BISCAIA FERREIRA**

**PROJETO DATA WAREHOUSE**

**CURITIBA**

**2023ALEXANDRE DIANO DE SOUZA**

**ARTHUR GIAN A. DE OLIVEIRA**

**NICHOLAS MANOEL DE P. ANUNES**

**GABRIEL BISCAIA FERREIRA**

**PROJETO DATA WAREHOUSE**

Trabalho acadêmico apresentado ao curso de Graduação em Análise e Desenvolvimento de Sistemas, Universidade Federal do Paraná, referente ao seminário em equipe, como requisito parcial à obtenção de nota na disciplina de Banco de Dados III.

Professor(a): Prof. João Eugenio Marynowski

**CURITIBA**

**2023**

**SUMÁRIO**

[**1. INTRODUÇÃO 4**](#_heading=h.zdwjfw5klee2)

[**2. SITUAÇÃO E MODELO ATUAL 5**](#_heading=h.mpvqij18le7g)

[2.1. Diagrama Relacional - Projeto 5](#_heading=h.wio6ystld8hj)

[2.2. Diagrama Relacional - Realidade 6](#_heading=h.ttxt6fekbxnx)

[**3. MODELO CONCEITUAL E LÓGICO DO BDM 7**](#_heading=h.tbmgbyiqt3br)

[3.1. Modelo Conceitual 7](#_heading=h.ixo2axfoty23)

[3.1.1. Premissas 7](#_heading=h.vqbrp6vs23lu)

[3.1.2. Diagrama Conceitual 8](#_heading=h.8stwtj9s56oa)

[3.2. Modelo Lógico do BDM 9](#_heading=h.ffhvghpkpa8t)

[**4. ARQUITETURA DO DW PROPOSTO 9**](#_heading=h.kdcvd5bj27x4)

[**5. ETL 10**](#_heading=h.6uu1f87otb)

[5.1. Migração de dados com Pentaho 10](#_heading=h.dy2ytaurs0yr)

[5.2. Adaptação da SA 10](#_heading=h.jp56k0dbpq4b)

[5.3. Criação das Tabelas Fato e Dimensões no DW 13](#_heading=h.leoilveqx87v)

[5.4. Inserção de Dados nas Tabelas Dimensão do DW 15](#_heading=h.oqponawd0kft)

[5.5. Inserção de Dados nas Tabelas Fato do DW 16](#_heading=h.80c3l2togjf1)

[5.5.1. Inserção de Dados na Tabela Fato Compra 16](#_heading=h.ime4o5sm9t5l)

[5.5.2. Inserção de Dados na Tabela Fato Venda 19](#_heading=h.ojrpk8u61h0)

[5.5.3. Inserção de Dados na Tabela Fato Lucro 22](#_heading=h.nzng7nfqs1yd)

[**6. ANÁLISE GERENCIAL 23**](#_heading=h.gwudccfx9ph4)

[6.1. Quantidade de produtos vendidos por tipo e categoria ao longo do tempo 23](#_heading=h.7bpvjmqm2bm)

[6.1.1. Evidência Análise Gerencial 24](#_heading=h.af82kwphv0ys)

[6.2. Quantidade e valor total das compras por produto, possibilitando visão hierárquica ao longo do tempo 24](#_heading=h.i2hiiai0xu0y)

[6.2.1. Evidências Análise Gerencial 28](#_heading=h.ppsttku7vji1)

[6.2.1.1. Visão considerando produto, ano e mês 28](#_heading=h.e8s6vme6sqvt)

[6.2.1.2. Visão considerando produto e mês 29](#_heading=h.wqells511q3)

[6.2.1.3. Visão considerando produto e ano 30](#_heading=h.tu1p3z1u4whx)

[6.2.1.4. Visão considerando mês e ano 30](#_heading=h.qxglzw70c6ei)

[6.2.1.5. Visão considerando apenas mês 31](#_heading=h.j3pj5psygoah)

[6.2.1.6. Visão considerando apenas ano 31](#_heading=h.kjnhjd7ch8tg)

[6.2.1.7. Visão considerando apenas produto 31](#_heading=h.3nfo5axumg5r)

[6.2.1.8. Visão sem considerar nada 32](#_heading=h.vglu8f7qtak6)

[6.3. Clientes que mais gastaram na loja virtual com quantidade acumulada, valor acumulado e média em um determinado período 32](#_heading=h.pkr2ydx9zoci)

[6.3.1. Evidências Análise Gerencial 34](#_heading=h.l42avkrzap8h)

[6.3.1.1. Período considerando dia, mês e ano 34](#_heading=h.hxl7fhgeguv7)

[6.3.1.2. Período considerando mês e ano 35](#_heading=h.bsdiav8odaay)

[6.3.1.3. Período considerando apenas ano 35](#_heading=h.43ydrn1xhbvf)

[6.3.1.4. Período total 36](#_heading=h.ilqitua4zzjr)

[6.4. Últimas compras realizadas por cliente, e tempo decorrido até a data atual 36](#_heading=h.dy79585wpm23)

[6.4.1. Evidência Análise Gerencial 37](#_heading=h.6nn78p2r1bex)

[6.5. Lucratividade bruta dos produtos comprados e posteriormente vendidos 38](#_heading=h.q44kpolqtlwr)

[6.5.1. Evidências Análise Gerencial 38](#_heading=h.fpbaomosjd0q)

[6.5.1.1. Lucro por produto 38](#_heading=h.7sloez970b2i)

[6.5.1.2. Lucro total 39](#_heading=h.paohjd6vcqsh)

[6.6. Quantidade de atendimentos realizados ao longo do tempo. 39](#_heading=h.lhtkq23xz8lw)

[6.6.1. Evidência Análise Gerencial 39](#_heading=h.4um0zttsjhdg)

[**7. FREQUÊNCIA DE CARGA 39**](#_heading=h.etkaatknzfl)

[**8. TEMPO DE RETENÇÃO 40**](#_heading=h.lt01x3ze27h7)

[**9. HARDWARE E SGBDS RECOMENDADOS 40**](#_heading=h.qctcpnxe4jgq)

[**10. CONSIDERAÇÕES FINAIS 40**](#_heading=h.pmyvg17n0k05)

[**11. REFERÊNCIAS 40**](#_heading=h.qillrormj911)

[**12. INFORMAÇÕES COMPLEMENTARES 41**](#_heading=h.22rm0iexxwgq)

[12.1. Ferramentas Utilizadas 41](#_heading=h.3fpaqlp8bs1l)

[12.2. Evidências 41](#_heading=h.5s4y4g4bjej2)

[12.2.1. Restauração do Arquivo de Backup no MS SQL Server 41](#_heading=h.k8vqb1711n3b)

[12.2.1.1. Importação Dados no MS SQL Server 42](#_heading=h.tzee5iyrlato)

[12.2.1.2. Evidência da Execução do Script para Restaurar o Backup 43](#_heading=h.cha5biy9yo00)

[12.2.1.3. Evidência da Base e Tabelas Criadas Após Restauração 44](#_heading=h.j4uj517uq73w)

[**ANEXOS 45**](#_heading=h.pje9axd76660)

# INTRODUÇÃO

No contexto atual do mundo, com o avanço da complexidade das organizações junto com a ascensão tecnológica, a relevância da gestão dos repositórios de dados aumentou. Tomar decisões precisas em um mercado cada vez mais competitivo tornou-se crucial para a sobrevivência de uma organização.

Dessa forma, a Ciência de Dados desempenha um papel vital neste cenário, pois permite transformar uma grande quantidade de dados brutos em informação e conhecimento. A implantação de um Data Warehouse (DW) tem como objetivo abordar essa necessidade, reestruturando um backup de um banco de dados operacional de forma a tornar mais eficiente a extração de informações gerenciais com base em fatos.

Na situação apresentada, temos um banco de dados relacional operacional de uma cadeia de lojas de varejo, onde a análise das informações é extremamente demorada, prejudicando efetivamente as operações. Nesse contexto, a criação de um DW a partir desse banco de dados pode fornecer uma fonte de informações gerenciais sem os problemas atuais (processamento e extração de dados lentos), desde que seja utilizada uma infraestrutura apropriada. Atualmente, existem inúmeras opções, ferramentas e soluções de DW disponíveis no mercado com essa infraestrutura, tanto com implementação local quanto em nuvem, como o Amazon Redshift, o Microsoft Azure, entre outras, e cada uma delas possui características adequadas para diferentes aplicações de DW.

A proposta da nossa equipe é configurar uma instância de MS SQL Server e realizar a restauração do arquivo de backup BD\_VAREJO.sql. Usaremos o DBeaver para conectar-se à base de dados da instância do MS SQL Server e criar o diagrama lógico relacional das tabelas com a finalidade de comparar os relacionamentos das tabelas com o diagrama do arquivo BD\_VAREJO.png. Assim que o backup estiver restaurado no MS SQL Server, a proposta é transferir todos os dados do MS SQL Server para uma área de staging em um banco de dados PostgreSQL.

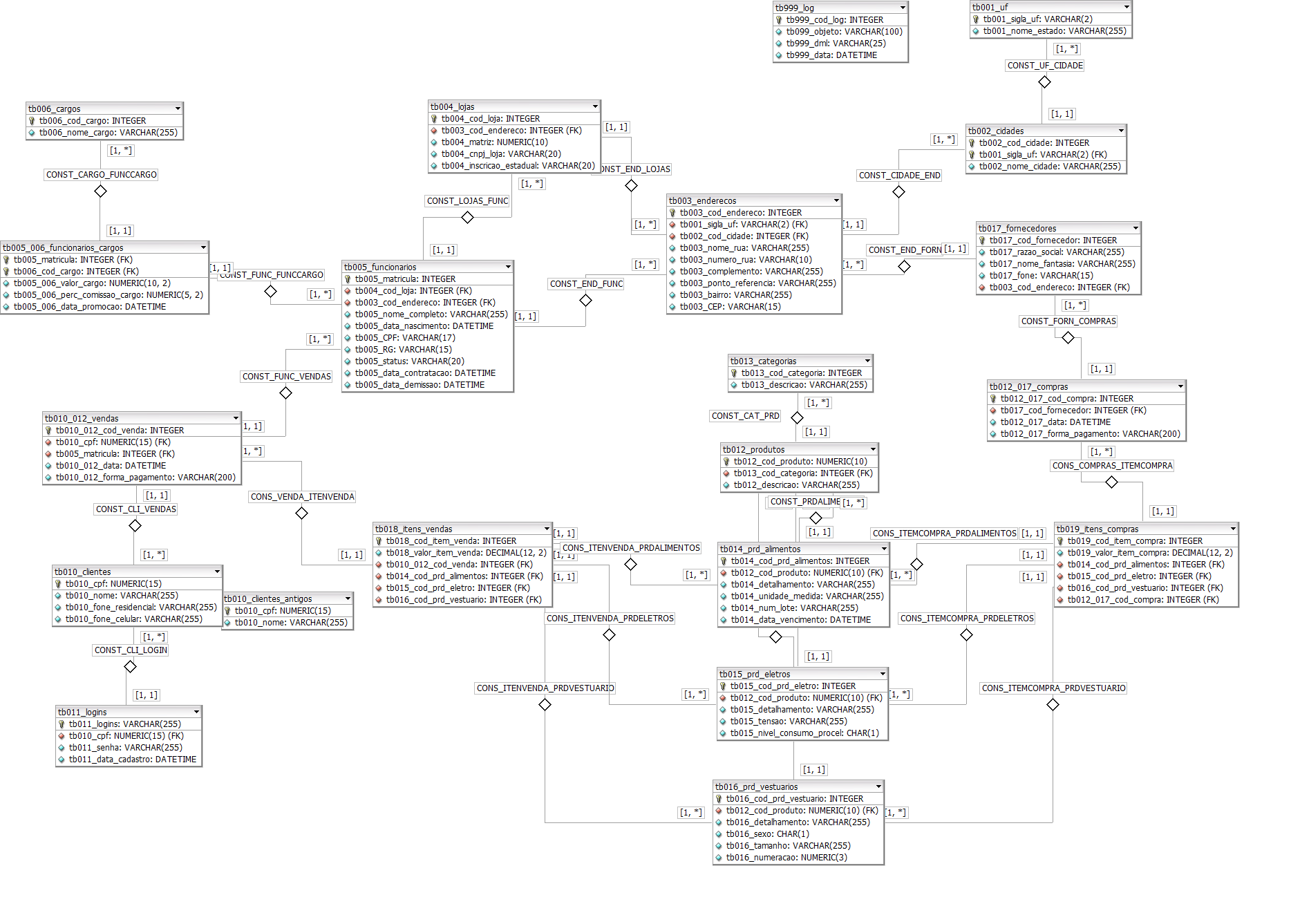
Após a transferência dos dados para a área de staging, o DW será desenvolvido no PostgreSQL a partir dos dados carregados na área de staging. A ferramenta Pentaho será usada para realizar o processo de ETL.

# SITUAÇÃO E MODELO ATUAL

## Diagrama Relacional - Projeto

O diagrama lógico relacional a seguir, trata-se do diagrama do projeto do BD, que não necessariamente representa como o BD realmente se encontra atualmente.

Figura 1 - Diagrama relacional atual

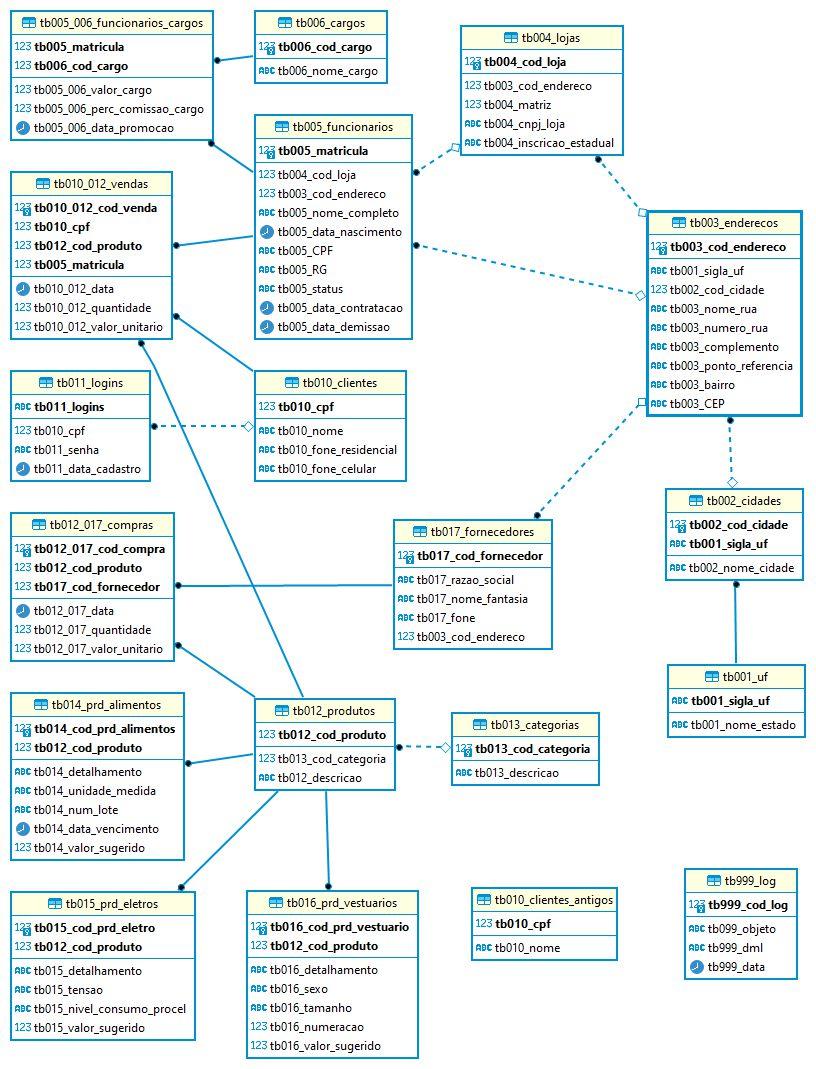
****

Fonte: Material base disponibilizado

## Diagrama Relacional - Realidade

O diagrama lógico relacional a seguir trata-se do diagrama do projeto do BD que foi obtido a partir da realidade de como o BD encontra-se atualmente.

Figura 2 - Diagrama Relacional modificado



Fonte: Os Autores (2023)

* 1. **Comparação Entre os Diagramas Real e de Projeto**

Comparando-se o diagrama relacional do projeto com o diagrama relacional de fato implementado no BD físico, constatou-se que:

* Não existe a tabela tb018\_itens\_vendas;
* Não existe a tabela tb019\_itens\_compras;
* Tabela tb012\_017\_compras possui:
  + Chave estrangeira tb012\_cod\_produto;
  + Coluna tb012\_017\_quantidade;
  + Coluna tb012\_017\_valor\_unitario.
* Tabela tb012\_017\_compras não possui:
  + Coluna tb012\_017\_forma\_pagamento;
* Tabela tb019\_012\_vendas possui:
  + Chave estrangeira tb012\_cod\_produto;
  + Coluna tb010\_012\_quantidade;
  + Coluna tb010\_012\_valor\_unitario;
* Tabela tb019\_012\_vendas não possui:
  + Coluna tb010\_012\_forma\_pagamento;
* Tabela tb012\_017\_compras não possui relacionamento com tabela
* tb019\_itens\_compras, mas sim com a tabela tb012\_produtos;
* Tabela tb010\_tb012\_vendas não possui relacionamento com a tabela
* tb018\_itens\_venda;
* Tabela tb012\_produtos se relaciona com a tabela tb019\_012\_vendas;
* Tabela tb014\_prd\_alimentos possui uma coluna tb014\_valor\_sugerido;
* Tabela tb015\_prd\_eletros possui uma coluna tb015\_valor\_sugerido;
* Tabela tb016\_prd\_vestuarios possui uma coluna tb016\_valor\_sugerido;

# MODELO CONCEITUAL E LÓGICO DO BDM

## Modelo Conceitual

### Premissas

**Tipos**:

* Alimentos;
* Eletros;
* Vestuário.

**Categorias**:

* Alimentos Perecíveis;
* Alimentos Não Perecíveis;
* Eletrodomésticos;
* Eletrônicos;
* CD e DVD;
* Roupas Unissex;
* Roupas Infantis.

Analisando o repositório de informações, observou-se que todos os clientes estão registrados. Consequentemente, tornou-se desafiante distinguir entre uma transação realizada virtualmente ou fisicamente. Portanto, assumimos como suposição que todas as transações ocorrem virtualmente e que todas as compras são efetuadas em plataformas digitais.

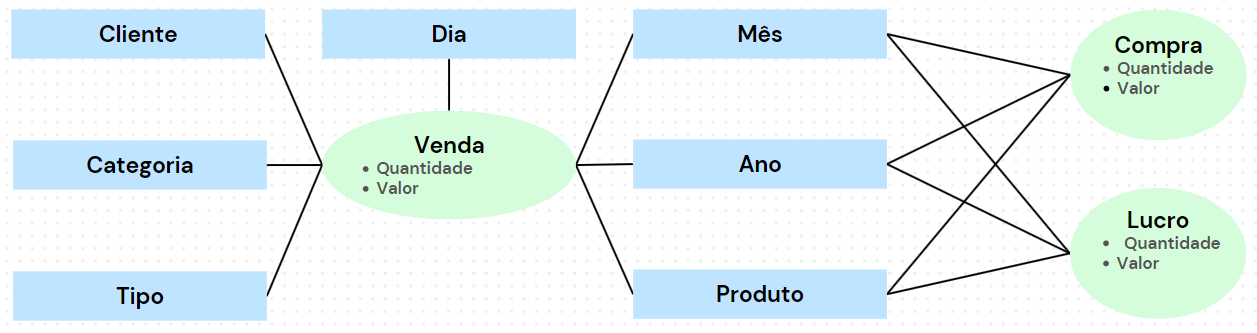
Chegamos à conclusão de que a dimensão temporal, especificamente o "dia," não possui a mesma relevância para a categoria "Compra" em comparação com a categoria "Venda." Portanto, optamos por eliminar a dimensão "dia" associada à categoria "Compra."

Consideramos que as vendas representam o único tipo de interação com os clientes. Portanto, a questão de gestão relacionada à "Quantidade de interações realizadas ao longo do período" se traduz na quantidade de vendas efetuadas durante esse período.

### Diagrama Conceitual

Decidimos criar um diagrama seguindo o modelo Constelação, levando em consideração que a estrutura disponível nos permite ter dois eventos, Venda e Compra, ambos compartilhando características (mês, ano e produto). Além disso, temos um evento Lucro, que também compartilha as mesmas características de produto, mês e ano, acompanhado pelo atributo "valor" conforme requerido para as análises de gestão. Como ilustrado na Figura 3, Venda, Compra e Lucro apresentam configurações do tipo constelação, interconectadas através das dimensões em comum, formando assim uma estrutura de constelação.

Figura 3 - Diagrama Conceitual

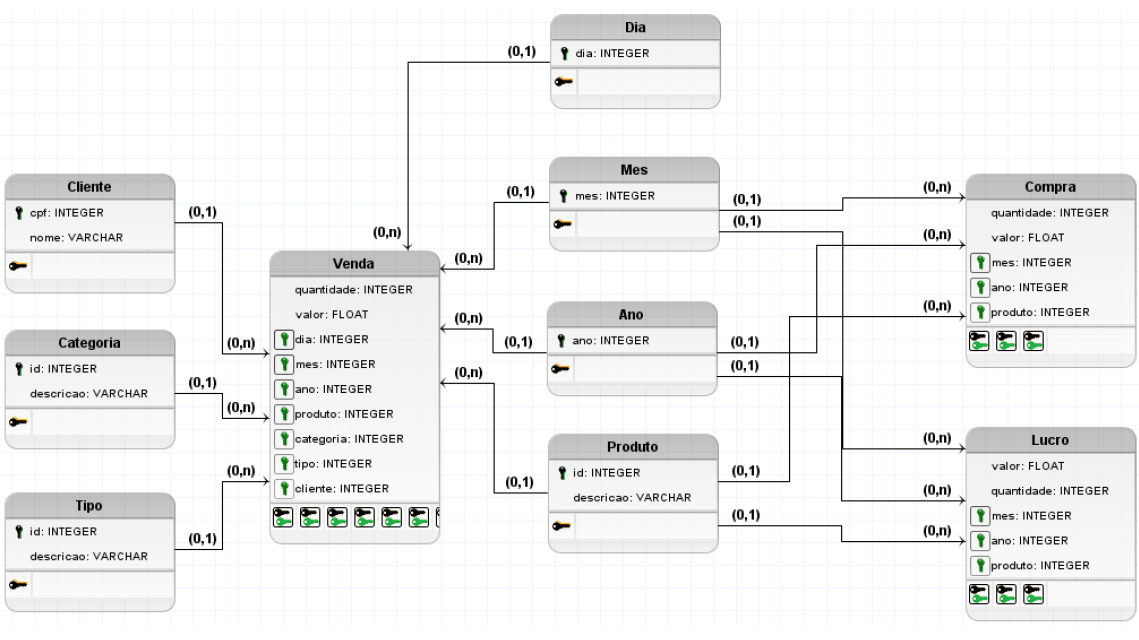


Fonte: Os Autores (2023)

## Modelo Lógico do BDM

Com base no modelo conceitual previamente formulado, o modelo prático de concretização do Data Warehouse foi concebido. Neste esquema, foram estabelecidas as conexões e suas quantidades, além dos formatos de dados, chaves principais e chaves secundárias.

Figura 4 - Modelo Lógico



Fonte: Os Autores (2023)

# ARQUITETURA DO DW PROPOSTO

Podemos observar na Figura 5 o modelo da arquitetura, onde foi utilizada a ferramenta Pentaho para realizar a extração dos dados da base relacional operacional (MS SQL Server) e inserção dos dados extraídos em uma Staging Area (SA) relacional (PostreSQL). Uma vez osdados carregados na SA foram utilizados queries SQL para a realização da extração e estruturação dos dados da SA e alimentação do DW.

****

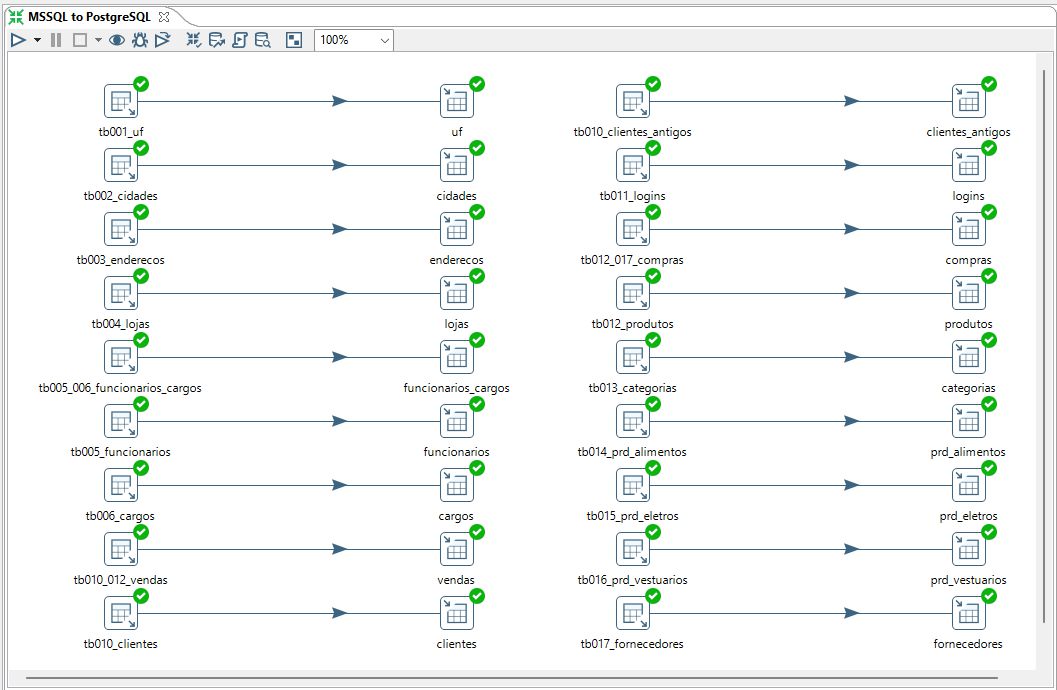
Fonte: Os Autores (2023)

# ETL

## Migração de dados com Pentaho

Antes da execução das ETLs para a alimentação do DW, realizamos a migração dos dados através do Pentaho para a nossa Staging Area (Figura 6), visando facilitar a aplicação das ETLs e geração dos Datamarts posteriormente.

Figura 6 - Migração com Pentaho



Fonte: Os Autores (2023)

## Adaptação da SA

Vale ressaltar que na Staging Area foi adicionada uma coluna relacionada com o tipo do produto, com o intuito de auxiliar nas Análises Gerenciais.

// Adição da coluna tipo\_produto na tabela produtos de staging area

ALTER TABLE staging.produtos add tipo\_produto int;

// Update da coluna tipo\_produto na staging area conforme tipo de cada produto

// Considerando produtos do tipo alimento

UPDATE staging.produtos

SET tipo\_produto = 1

WHERE tb012\_cod\_produto IN (SELECT p.tb012\_cod\_produto FROM

staging.prd\_alimentos pa, staging.produtos p

WHERE pa.tb012\_cod\_produto = p.tb012\_cod\_produto);

// Considerando produtos do tipo eletro

UPDATE staging.produtos

SET tipo\_produto = 2

WHERE tb012\_cod\_produto IN (SELECT p.tb012\_cod\_produto FROM

staging.prd\_eletros pa, staging.produtos p

WHERE pa.tb012\_cod\_produto = p.tb012\_cod\_produto);

// Considerando produtos do tipo vestuario

UPDATE staging.produtos

SET tipo\_produto = 3

WHERE tb012\_cod\_produto IN (SELECT p.tb012\_cod\_produto FROM

staging.prd\_vestuarios pa, staging.produtos p

WHERE pa.tb012\_cod\_produto = p.tb012\_cod\_produto);

Dada a necessidade de obter a Lucratividade na análise gerencial, tangenciamos a alternativa de realizar os cálculos através das médias de valor de compra e média de valor de venda, dessa maneira esse processamento teria que ocorrer no momento do ETL, com a criação de duas novas colunas na tabela Produto com os valores médios que serão utilizados posteriormente no DW para a realização do cálculo da lucratividade final.

// Adição da coluna preco\_medio\_compra na tabela produtos de staging area

ALTER TABLE staging.produtos add preco\_medio\_compra float;

// Adição da coluna preco\_medio\_venda na tabela produtos de staging area

ALTER TABLE staging.produtos add preco\_medio\_venda float;

// Cálculo da média ponderada do valor pago pelo produto na compra e inserção

dos valores na coluna preco\_medio\_compra da tabela staging.produtos

UPDATE staging.produtos p

SET preco\_medio\_compra = subquery.preco\_medio\_compra

FROM (

SELECT

p.tb012\_cod\_produto,

p.tb012\_descricao,

sum((c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario))/sum(c.tb012\_017\_quanti

dade) AS preco\_medio\_compra

FROM

staging.compras c,

staging.produtos p

GROUP BY

p.tb012\_cod\_produto,

c.tb012\_cod\_produto,

p.tb012\_descricao

HAVING

c.tb012\_cod\_produto = p.tb012\_cod\_produto

) AS subquery

WHERE

p.tb012\_cod\_produto = subquery.tb012\_cod\_produto;

// Cálculo da média ponderada do valor pago pelo produto na venda e inserção

dos valores na coluna preco\_medio\_venda da tabela staging.produtos

UPDATE staging.produtos p

SET preco\_medio\_venda = subquery.preco\_medio\_venda

FROM (

SELECT

p.tb012\_cod\_produto,

p.tb012\_descricao,

sum((v.tb010\_012\_quantidade

\*v.tb010\_012\_valor\_unitario))/sum(v.tb010\_012\_quantidade) AS preco\_medio\_venda

FROM

staging.vendas v,

staging.produtos p

GROUP BY

p.tb012\_cod\_produto,

v.tb012\_cod\_produto,

p.tb012\_descricao

HAVING

v.tb012\_cod\_produto = p.tb012\_cod\_produto

) AS subquery

WHERE

p.tb012\_cod\_produto = subquery.tb012\_cod\_produto;

## Criação das Tabelas Fato e Dimensões no DW

A seguir são apresentados os SQL de criação das tabelas fato e das tabelas

dimensões, bem como os alter tables relativos à aplicação das chaves estrangeiras nas tabelas fato.

// Primeiro Passo

// Criar o schema e as tabelas

CREATE SCHEMA dw;

CREATE TABLE dw.Cliente (

cpf BIGINT PRIMARY KEY,

nome VARCHAR

);

CREATE TABLE dw.Ano (

ano INTEGER PRIMARY KEY

);

CREATE TABLE dw.Mes (

mes INTEGER PRIMARY KEY

);

CREATE TABLE dw.Produto (

id INTEGER PRIMARY KEY,

descricao VARCHAR,

preco\_medio\_compra FLOAT,

preco\_medio\_venda FLOAT

);

CREATE TABLE dw.Categoria (

id INTEGER PRIMARY KEY,

descricao VARCHAR

);

CREATE TABLE dw.Dia (

dia INTEGER PRIMARY KEY

);

CREATE TABLE dw.Tipo (

id INTEGER PRIMARY KEY,

descricao VARCHAR

);

CREATE TABLE dw.Venda (

quantidade INTEGER,

valor FLOAT,

dia INTEGER,

mes INTEGER,

ano INTEGER,

produto INTEGER,

categoria INTEGER,

tipo INTEGER,

cliente BIGINT

);

CREATE TABLE dw.Compra (

quantidade INTEGER,

valor FLOAT,

mes INTEGER,

ano INTEGER,

produto INTEGER

);

CREATE TABLE dw.Lucro (

produto INTEGER,

quantidade INTEGER,

lucro FLOAT

);

// Segundo Passo

// Adicionar as constraints

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaCliente

FOREIGN KEY (cliente)

REFERENCES dw.Cliente (cpf);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaDia

FOREIGN KEY (dia)

REFERENCES dw.Dia (dia);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaMes

FOREIGN KEY (mes)

REFERENCES dw.Mes (mes);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaAno

FOREIGN KEY (ano)

REFERENCES dw.Ano (ano);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaProduto

FOREIGN KEY (produto)

REFERENCES dw.Produto (id);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaCategoria

FOREIGN KEY (categoria)

REFERENCES dw.Categoria (id);

ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaTipo

FOREIGN KEY (tipo)

REFERENCES dw.Tipo (id);

ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraMes

FOREIGN KEY (mes)

REFERENCES dw.Mes (mes);

ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraAno

FOREIGN KEY (ano)

REFERENCES dw.Ano (ano);

ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraProduto

FOREIGN KEY (produto)

REFERENCES dw.Produto (id);

ALTER TABLE dw.Lucro ADD CONSTRAINT fkLucroProduto

FOREIGN KEY (produto)

REFERENCES dw.Produto (id);

## Inserção de Dados nas Tabelas Dimensão do DW

Os SQL apresentados a seguir são referentes a inserção dos dados nas tabelas Dimensão do DW. Os dados inseridos nestas tabelas do DW foram obtidos a partir da SA, mediante processo de ETL.

// Inserção de dados na tabela dimensão Ano

INSERT INTO dw.ano

SELECT DISTINCT EXTRACT(YEAR FROM tb010\_012\_data) FROM staging.vendas

ON CONFLICT DO NOTHING;

INSERT INTO dw.ano

SELECT DISTINCT EXTRACT(YEAR FROM tb012\_017\_data) FROM staging.compras

ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Mes

INSERT INTO dw.mes

SELECT DISTINCT EXTRACT(MONTH FROM tb010\_012\_data) FROM staging.vendas

ON CONFLICT DO NOTHING;

INSERT INTO dw.mes

SELECT DISTINCT EXTRACT(MONTH FROM tb012\_017\_data) FROM staging.compras

ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Dia

INSERT INTO dw.dia

SELECT DISTINCT EXTRACT(DAY FROM tb010\_012\_data) FROM staging.vendas

ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Produto

INSERT INTO dw.produto

SELECT DISTINCT p.tb012\_cod\_produto, p.tb012\_descricao,

p.preco\_medio\_compra, p.preco\_medio\_venda FROM staging.produtos p;

// Inserção de dados na tabela dimensão Cliente

INSERT INTO dw.cliente

SELECT DISTINCT c.tb010\_cpf, c.tb010\_nome FROM staging.clientes c;

// Inserção de dados na tabela dimensão Categoria

INSERT INTO dw.categoria

SELECT DISTINCT c.tb013\_cod\_categoria, c.tb013\_descricao FROM

staging.categorias c;

// Inserção de dados na tabela dimensão Tipo

INSERT INTO dw.tipo (id, descricao)

VALUES (1, 'alimento'), (2, 'eletro'), (3,'vestuario');

## Inserção de Dados nas Tabelas Fato do DW

Os SQL apresentados a seguir são referentes a inserção dos dados nas tabelas Fato do DW. Os dados inseridos nestas tabelas do DW foram obtidos a partir da SA, mediante processo de ETL.

### Inserção de Dados na Tabela Fato Compra

// Inserção de dados na tabela fato Compra

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

c.tb012\_cod\_produto AS produto,

c.tb012\_017\_quantidade AS quantidade,

c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario AS valor,

EXTRACT(MONTH FROM c.tb012\_017\_data) AS mes,

EXTRACT(YEAR FROM c.tb012\_017\_data) AS ano

FROM

staging.compras c

);

// Independente do mês

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

c.tb012\_cod\_produto AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

NULL AS mes,

EXTRACT(YEAR FROM c.tb012\_017\_data) AS ano

FROM

staging.compras c

GROUP BY c.tb012\_cod\_produto, EXTRACT(YEAR FROM c.tb012\_017\_data)

);

// Independente do ano

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

c.tb012\_cod\_produto AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

EXTRACT(MONTH FROM c.tb012\_017\_data) AS mes,

NULL AS ano

FROM

staging.compras c

GROUP BY c.tb012\_cod\_produto, EXTRACT(MONTH FROM c.tb012\_017\_data)

);

// Independente do ano e mês

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

c.tb012\_cod\_produto AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

NULL AS mes,

NULL AS ano

FROM

staging.compras c

GROUP BY c.tb012\_cod\_produto

);

// Independente de produto

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

NULL AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

EXTRACT(MONTH FROM c.tb012\_017\_data) AS mes,

EXTRACT(YEAR FROM c.tb012\_017\_data) AS ano

FROM

staging.compras c

GROUP BY EXTRACT(MONTH FROM c.tb012\_017\_data), EXTRACT(YEAR FROM

c.tb012\_017\_data)

);

// Independente do produto e mês

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

NULL AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

NULL AS mes,

EXTRACT(YEAR FROM c.tb012\_017\_data) AS ano

FROM

staging.compras c

GROUP BY EXTRACT(YEAR FROM c.tb012\_017\_data)

);

// Independente do produto e ano

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

NULL AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

EXTRACT(MONTH FROM c.tb012\_017\_data) AS mes,

NULL AS ano

FROM

staging.compras c

GROUP BY EXTRACT(MONTH FROM c.tb012\_017\_data)

);

// Independente do produto, mês e ano

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)

(

SELECT

NULL AS produto,

SUM(c.tb012\_017\_quantidade) AS quantidade,

SUM(c.tb012\_017\_quantidade\*c.tb012\_017\_valor\_unitario) AS valor,

NULL AS mes,

NULL AS ano

FROM

staging.compras c

);

### Inserção de Dados na Tabela Fato Venda

// Inserção de dados na tabela fato Venda

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

v.tb010\_012\_quantidade AS quantidade,

v.tb010\_012\_quantidade\*v.tb010\_012\_valor\_unitario AS valor,

EXTRACT(DAY FROM v.tb010\_012\_data) AS dia,

EXTRACT(MONTH FROM v.tb010\_012\_data) AS mes,

EXTRACT(YEAR FROM v.tb010\_012\_data) AS ano,

v.tb012\_cod\_produto AS produto,

p.tb013\_cod\_categoria AS categoria,

p.tipo\_produto AS tipo,

v.tb010\_cpf AS cliente

FROM

staging.vendas v,

staging.produtos p

WHERE

v.tb012\_cod\_produto = p.tb012\_cod\_produto

);

// considerando categoria, tipo e quantidade

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

SUM(v.tb010\_012\_quantidade) AS quantidade,

NULL AS valor,

NULL AS dia,

NULL AS mes,

NULL AS ano,

NULL AS produto,

p.tb013\_cod\_categoria AS categoria,

p.tipo\_produto AS tipo,

NULL AS cliente

FROM

staging.vendas v,

staging.produtos p

WHERE

v.tb012\_cod\_produto = p.tb012\_cod\_produto

GROUP BY

p.tb013\_cod\_categoria,

p.tipo\_produto

);

// consideranco quantidade, valor, dia, mes, ano e cliente

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

SUM(v.tb010\_012\_quantidade) AS quantidade,

SUM(v.tb010\_012\_quantidade\*v.tb010\_012\_valor\_unitario) AS valor,

EXTRACT(DAY FROM v.tb010\_012\_data) AS dia,

EXTRACT(MONTH FROM v.tb010\_012\_data) AS mes,

EXTRACT(YEAR FROM v.tb010\_012\_data) AS ano,

NULL AS produto,

NULL AS categoria,

NULL AS tipo,

v.tb010\_cpf AS cliente

FROM

staging.vendas v

GROUP BY

EXTRACT(DAY FROM v.tb010\_012\_data),

EXTRACT(MONTH FROM v.tb010\_012\_data),

EXTRACT(YEAR FROM v.tb010\_012\_data),

v.tb010\_cpf

);

// consideranco quantidade, valor, mes, ano e cliente

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

SUM(v.tb010\_012\_quantidade) AS quantidade,

SUM(v.tb010\_012\_quantidade\*v.tb010\_012\_valor\_unitario) AS valor,

NULL AS dia,

EXTRACT(MONTH FROM v.tb010\_012\_data) AS mes,

EXTRACT(YEAR FROM v.tb010\_012\_data) AS ano,

NULL AS produto,

NULL AS categoria,

NULL AS tipo,

v.tb010\_cpf AS cliente

FROM

staging.vendas v

GROUP BY

EXTRACT(MONTH FROM v.tb010\_012\_data),

EXTRACT(YEAR FROM v.tb010\_012\_data),

v.tb010\_cpf

);

// considerando quantidade, valor, ano e cliente

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

SUM(v.tb010\_012\_quantidade) AS quantidade,

SUM(v.tb010\_012\_quantidade\*v.tb010\_012\_valor\_unitario) AS valor,

NULL AS dia,

NULL AS mes,

EXTRACT(YEAR FROM v.tb010\_012\_data) AS ano,

NULL AS produto,

NULL AS categoria,

NULL AS tipo,

v.tb010\_cpf AS cliente

FROM

staging.vendas v

GROUP BY

EXTRACT(YEAR FROM v.tb010\_012\_data),

v.tb010\_cpf

);

// considerando quantidade, valor e cliente

INSERT INTO

dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,

cliente)

(

SELECT

SUM(v.tb010\_012\_quantidade) AS quantidade,

SUM(v.tb010\_012\_quantidade\*v.tb010\_012\_valor\_unitario) AS valor,

NULL AS dia,

NULL AS mes,

NULL AS ano,

NULL AS produto,

NULL AS categoria,

NULL AS tipo,

v.tb010\_cpf AS cliente

FROM

staging.vendas v

GROUP BY

v.tb010\_cpf

);

### Inserção de Dados na Tabela Fato Lucro

// Inserção de dados na tabela fato Lucro

INSERT INTO dw.lucro (produto, quantidade, lucro)

(

SELECT

v.tb012\_cod\_produto AS produto,

SUM(v.tb010\_012\_quantidade) AS quantidade,

SUM(v.tb010\_012\_quantidade \* (p.preco\_medio\_venda -

p.preco\_medio\_compra)) AS lucro

FROM

staging.vendas v,

staging.produtos p

WHERE

v.tb012\_cod\_produto = p.tb012\_cod\_produto AND

p.preco\_medio\_compra IS NOT NULL

GROUP BY

v.tb012\_cod\_produto

);

// lucro total

INSERT INTO dw.lucro (produto, quantidade, lucro)

(

SELECT

NULL AS produto,

NULL AS quantidade,

SUM(v.tb010\_012\_quantidade \* (p.preco\_medio\_venda -

p.preco\_medio\_compra)) AS lucro

FROM

staging.vendas v,

staging.produtos p

WHERE

v.tb012\_cod\_produto = p.tb012\_cod\_produto AND

p.preco\_medio\_compra IS NOT NULL

);

# ANÁLISE GERENCIAL

Analisando as necessidades da empresa foram mapeadas as seguintes necessidades que devem ser respondidas pelo banco de dados multidimensional a ser implementado.

## Quantidade de produtos vendidos por tipo e categoria ao longo do tempo

SELECT

t.descricao AS tipo,

c.descricao AS categoria,

SUM(v.quantidade) AS quantidade

FROM

dw.venda v,

dw.tipo t,

dw.categoria c

WHERE

v.categoria = c.id AND

v.tipo = t.id AND

v.quantidade IS NOT NULL AND

v.valor IS NULL AND

v.dia IS NULL AND

v.mes IS NULL AND

v.ano IS NULL AND

v.produto IS NULL AND

v.categoria IS NOT NULL AND

v.tipo IS NOT NULL AND

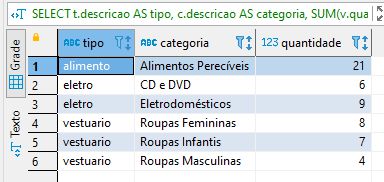
v.cliente IS NULL

GROUP BY

t.descricao,

**c.descricao;**

### Evidência Análise Gerencial



## Quantidade e valor total das compras por produto, possibilitando visão hierárquica ao longo do tempo

// Visão considerando produto, ano e mês

SELECT

c.produto,

p.descricao,

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.ano,

c.mes

FROM

dw.compra c,

dw.produto p

WHERE

p.id = c.produto AND

c.mes IS NOT NULL AND -- \* c.mes = 4

c.ano IS NOT NULL -- \* c.ano = 2010

GROUP BY

c.produto,

p.descricao,

c.ano,

c.mes;

// Visão considerando produto e mês

SELECT

c.produto,

p.descricao,

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.mes

FROM

dw.compra c,

dw.produto p

WHERE

p.id = c.produto AND

c.mes IS NOT NULL AND -- \* c.mes = 4

c.ano IS NULL

GROUP BY

c.produto,

p.descricao,

c.mes;

// Visão considerando produto e ano

SELECT

c.produto,

p.descricao,

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.ano

FROM

dw.compra c,

dw.produto p

WHERE

p.id = c.produto AND

c.ano IS NOT NULL AND -- \* c.ano = 2010

c.mes IS NULL

GROUP BY

c.produto,

p.descricao,

c.ano;

// Visão considerando mês e ano

SELECT

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.mes,

c.ano

FROM

dw.compra c

WHERE

c.produto IS NULL AND

c.mes IS NOT NULL AND -- \* c.mes = 4

c.ano IS NOT NULL -- \* c.ano = 2010

GROUP BY

c.mes,

c.ano;

// Visão considerando apenas mês

SELECT

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.mes

FROM

dw.compra c

WHERE

c.produto IS NULL AND

c.mes IS NOT NULL AND -- \* c.mes = 4

c.ano IS NULL

GROUP BY

c.mes;

// Visão considerando apenas ano

SELECT

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor,

c.ano

FROM

dw.compra c

WHERE

c.produto IS NULL AND

c.mes IS NULL AND

c.ano IS NOT NULL -- \* c.ano = 2010

GROUP BY

c.ano;

// Visão considerando apenas produto

SELECT

c.produto,

p.descricao,

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor

FROM

dw.compra c,

dw.produto p

WHERE

p.id = c.produto AND

c.ano IS NULL AND

c.mes IS NULL

GROUP BY

c.produto,

p.descricao;

// Visão sem considerar nada

SELECT

SUM(c.quantidade) AS quantidade,

SUM(c.valor) AS valor

FROM

dw.compra c

WHERE

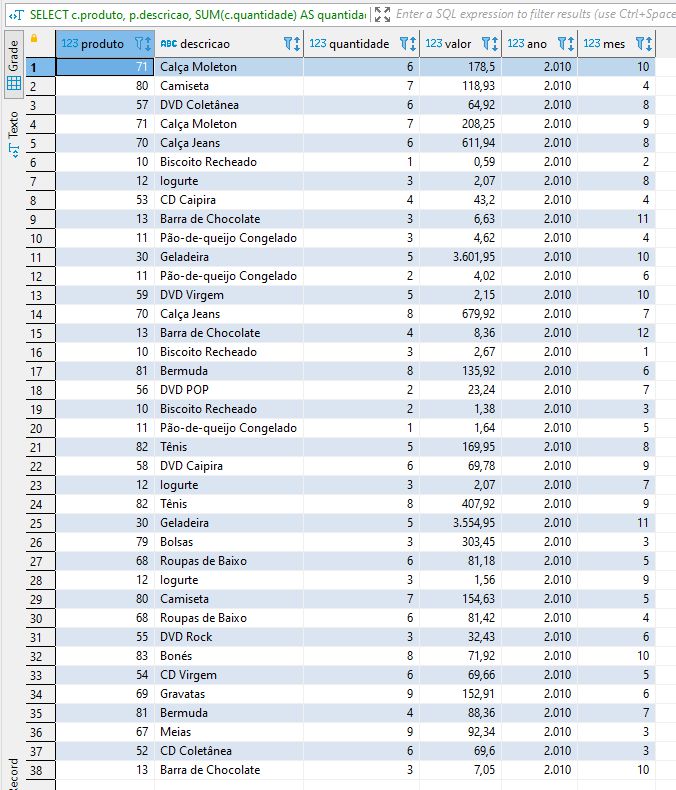
c.produto IS NULL AND

c.mes IS NULL AND

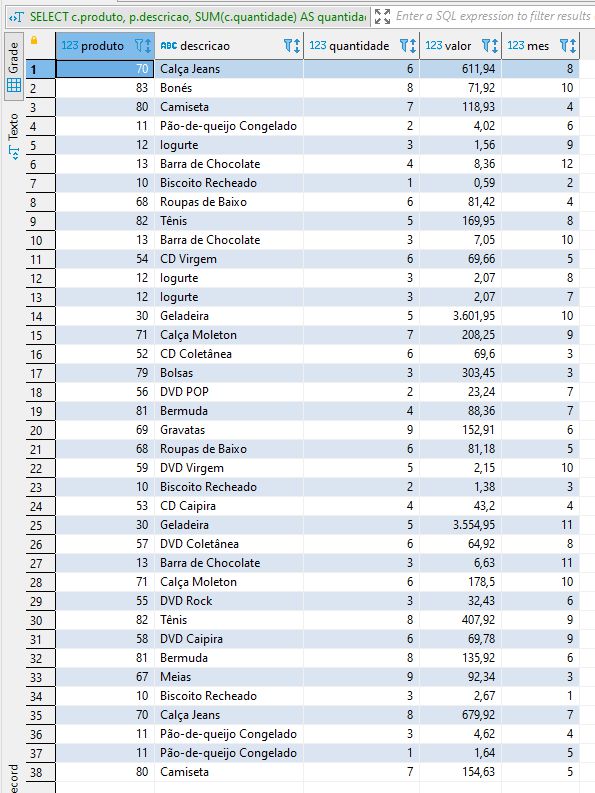
c.ano IS NULL;

### Evidências Análise Gerencial

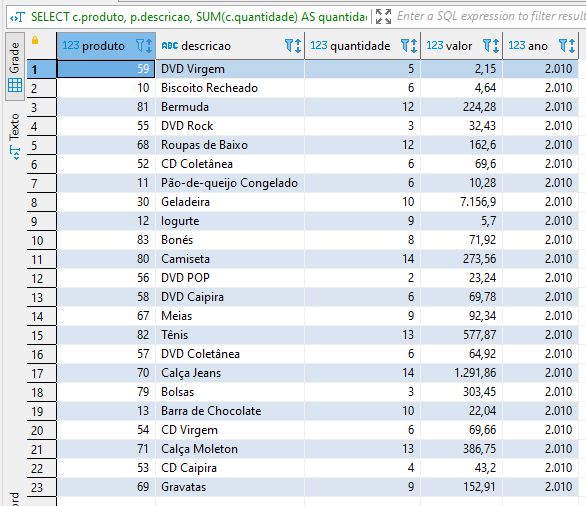
#### Visão considerando produto, ano e mês

****

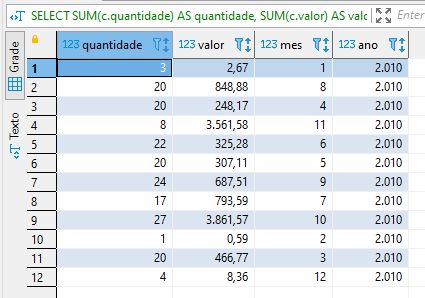
#### Visão considerando produto e mês

****

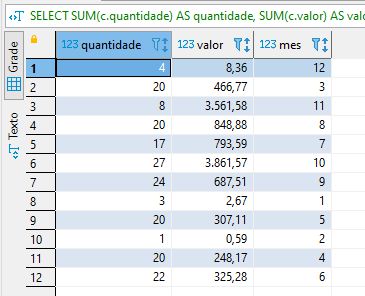
#### Visão considerando produto e ano

****

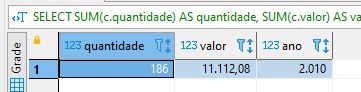
#### Visão considerando mês e ano

****

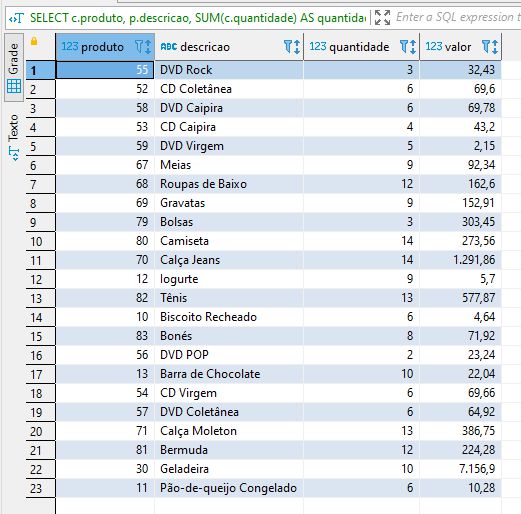
#### Visão considerando apenas mês

****

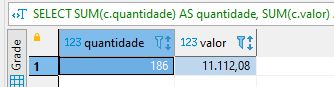
#### Visão considerando apenas ano

****

#### Visão considerando apenas produto



#### Visão sem considerar nada

****

## Clientes que mais gastaram na loja virtual com quantidade acumulada, valor acumulado e média em um determinado período

// Período considerado dia, mes e ano

SELECT

v.cliente as cliente,

SUM(v.quantidade) AS quantidade\_acumulada,

SUM(v.valor) AS valor\_acumulado,

AVG(v.valor) AS valor\_médio

FROM

dw.venda v

WHERE

v.dia BETWEEN 1 AND 31 AND

v.mes BETWEEN 3 AND 5 AND

v.ano BETWEEN 2009 AND 2011 AND

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NOT NULL AND

v.mes IS NOT NULL AND

v.ano IS NOT NULL AND

v.produto IS NULL AND

v.categoria IS NULL AND

v.tipo IS NULL AND

v.cliente IS NOT NULL

GROUP BY

v.cliente

ORDER BY

SUM(v.valor) DESC;

// Período considerando mes e ano

SELECT

v.cliente as cliente,

SUM(v.quantidade) AS quantidade\_acumulada,

SUM(v.valor) AS valor\_acumulado,

AVG(v.valor) AS valor\_médio

FROM

dw.venda v

WHERE

v.mes BETWEEN 3 AND 5 AND

v.ano BETWEEN 2009 AND 2011 AND

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NULL AND

v.mes IS NOT NULL AND

v.ano IS NOT NULL AND

v.produto IS NULL AND

v.categoria IS NULL AND

v.tipo IS NULL AND

v.cliente IS NOT NULL

GROUP BY

v.cliente

ORDER BY

SUM( v.valor) DESC;

// Período considerando apenas ano

SELECT

v.cliente as cliente,

SUM(v.quantidade) AS quantidade\_acumulada,

SUM(v.valor) AS valor\_acumulado,

AVG(v.valor) AS valor\_médio

FROM

dw.venda v

WHERE

v.ano BETWEEN 2009 AND 2011 AND

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NULL AND

v.mes IS NULL AND

v.ano IS NOT NULL AND

v.produto IS NULL AND

v.categoria IS NULL AND

v.tipo IS NULL AND

v.cliente IS NOT NULL

GROUP BY

v.cliente

ORDER BY

SUM(v.valor) DESC;

// Período total

SELECT

v.cliente as cliente,

SUM(v.quantidade) AS quantidade\_acumulada,

SUM(v.valor) AS valor\_acumulado,

AVG(v.valor) AS valor\_médio

FROM

dw.venda v

WHERE

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NULL AND

v.mes IS NULL AND

v.ano IS NOT NULL AND

v.produto IS NULL AND

v.categoria IS NULL AND

v.tipo IS NULL AND

v.cliente IS NOT NULL

GROUP BY

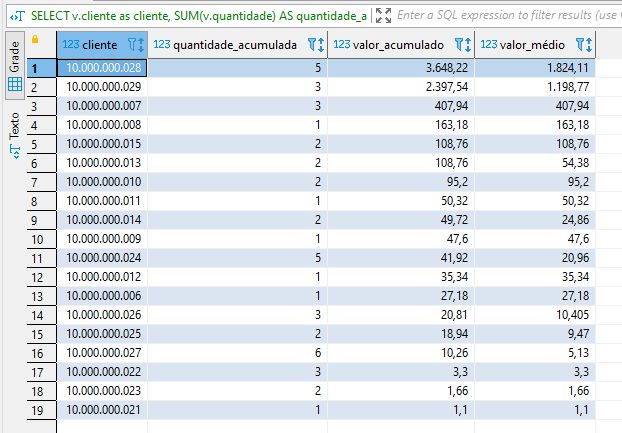
v.cliente

ORDER BY

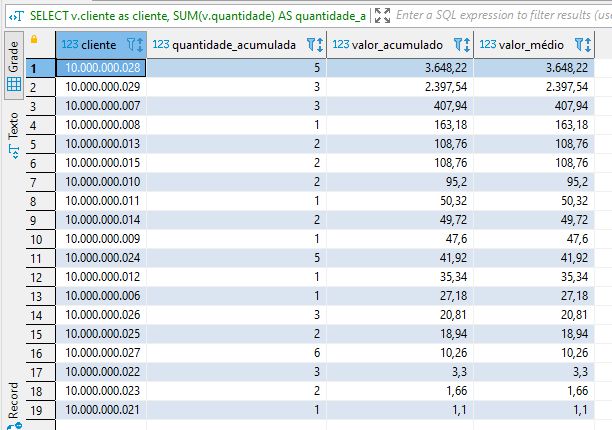
SUM(v.valor) DESC;

### Evidências Análise Gerencial

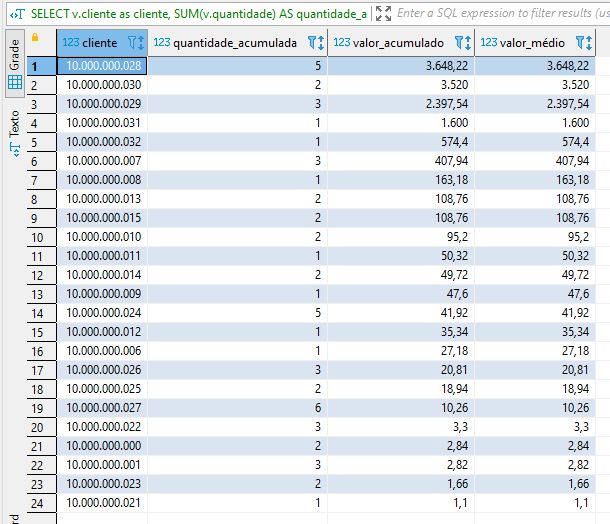
#### Período considerando dia, mês e ano



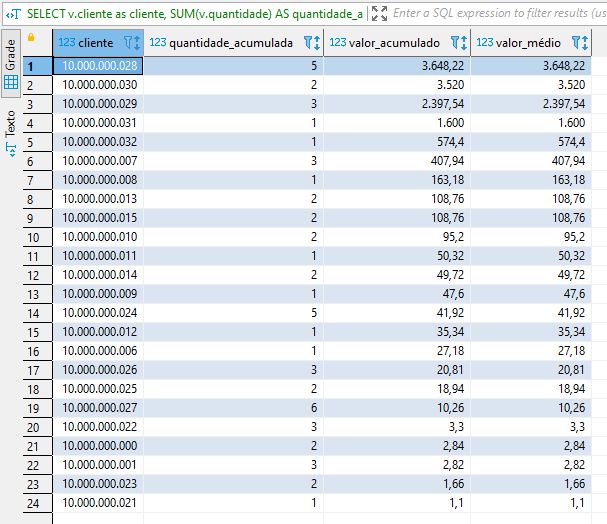
#### Período considerando mês e ano

****

#### Período considerando apenas ano



#### Período total

****

## Últimas compras realizadas por cliente, e tempo decorrido até a data atual

SELECT

v.cliente AS cliente,

c.nome AS nome,

make\_date(v.ano, v.mes, v.dia) AS data\_compra,

AGE(NOW():: date, make\_date(v.ano, v.mes, v.dia)) AS tempo\_decorrido

FROM

dw.venda v,

dw.cliente c

WHERE

v.cliente = c.cpf AND

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NOT NULL AND

v.mes IS NOT NULL AND

v.ano IS NOT NULL AND

v.produto IS NOT NULL AND

v.categoria IS NOT NULL AND

v.tipo IS NOT NULL AND

v.cliente IS NOT NULL

ORDER BY

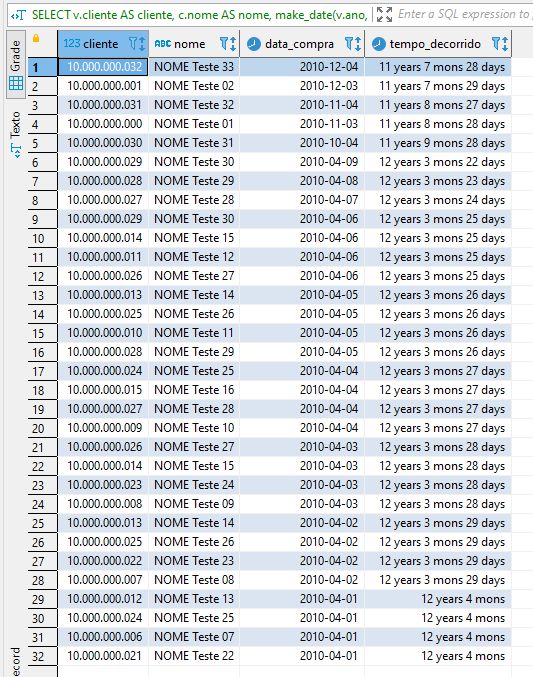
v.ano DESC,

v.mes DESC,

v.dia DESC

;

### Evidência Análise Gerencial



## Lucratividade bruta dos produtos comprados e posteriormente vendidos

// lucro por produto

SELECT

p.descricao AS produto,

l.quantidade AS quantidade\_vendida,

l.lucro AS lucro\_total

FROM

dw.lucro l,

dw.produto p

WHERE

l.produto = p.id AND

l.produto IS NOT NULL AND

l.quantidade IS NOT NULL AND

l.lucro IS NOT NULL

// lucro total

SELECT

l.lucro AS lucro\_total

FROM

dw.lucro l

WHERE

l.produto IS NULL AND

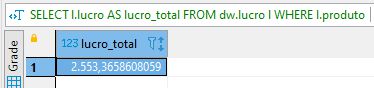
l.quantidade IS NULL AND

l.lucro IS NOT NULL

### Evidências Análise Gerencial

#### Lucro por produto

#### Lucro total



## Quantidade de atendimentos realizados ao longo do tempo.

SELECT

COUNT(\*) AS quantidade\_atendimentos

FROM

dw.venda v

WHERE

v.quantidade IS NOT NULL AND

v.valor IS NOT NULL AND

v.dia IS NOT NULL AND

v.mes IS NOT NULL AND

v.ano IS NOT NULL AND

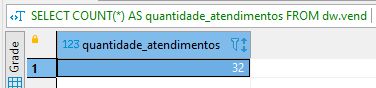
v.produto IS NOT NULL AND

v.categoria IS NOT NULL AND

v.tipo IS NOT NULL AND

v.cliente IS NOT NULL;

### Evidência Análise Gerencial

****

# FREQUÊNCIA DE CARGA

A frequência estabelecida das execuções ocorrerão diariamente, preferencialmente fora do expediente de trabalho a fim de não prejudicar o estabelecimento. A periodicidade foi estabelecida para que a loja possua um controle das vendas. O primeiro levantamento de dados será completo, e demandará mais tempo, já as execuções posteriores apenas incrementarão as cargas e serão mais rápidas.

# TEMPO DE RETENÇÃO

O tempo de retenção do banco de dados da Staging Area foi definido como no máximo uma semana, uma vez que será alimentado também semanalmente.

O tempo de retenção do Data Warehouse terá um tempo de 5 anos, onde a retenção de dados será controlada através de backups diários. Os dados serão distribuídos entre as lojas existentes levando em conta as informações necessárias para cada uma delas, e a sede principal será responsável por armazenar os dados de todas unidades.

# HARDWARE E SGBDS RECOMENDADOS

Para execução da Análises Gerenciais, recomendamos o SGBD PostgreSQL,

uma vez que o próprio DW já foi desenvolvido utilizando essa ferramenta, que apresenta um ótimo desempenho, e é gratuita para uso.

Quanto ao Hardware, a equipe indica a utilização de um servidor dedicado com a seguinte configuração:

* Processador: AMD Epyc 7742 2.25GHz, 64C/128T, 256M Cache.
* Placa Mãe: Supermicro H11SSL-NC Rev 2
* Memória RAM: 128Gb DDR4 ECC
* Armazenamento: SSD NAS 500Gb para o sistema operacional e HDD NAS 16TB
* 2 Fontes 750W com redundância

# CONSIDERAÇÕES FINAIS

Usar um data warehouse para extrair informações de gerenciamento pode não parecer fundamental para uma empresa inicialmente, mas à medida que a esta tem um crescimento, a extração dessas informações diretamente do servidor e de banco de dados operacionais pode ser custoso do ponto de vista de desempenho, pode atrapalhar as tarefas e rotinas do dia-a-dia, podendo até afetar custos, já que essas análises influenciam em tomadas de decisão que podem prejudicar a organização.

As soluções de data warehouse apresentadas aqui demonstram que sua implementação pode ser simples, podendo utilizar inclusive ferramentas gratuitas. Ou pode-se optar pela utilização de ferramentas mais otimizadas e poderosas, levando em consideração que isso deve ser visto como investimento, facilitando o acesso a informações que auxiliam em tomadas de decisão assertivas no que se refere aos objetivos da empresa.

# REFERÊNCIAS

**<**[**https://forums.pentaho.com/threads/217203-Error-while-connecting-to-MSSQL-DB/**](https://forums.pentaho.com/threads/217203-Error-while-connecting-to-MSSQL-DB/)**>**

**<**[**https://sourceforge.net/projects/jtds/**](https://sourceforge.net/projects/jtds/)**>**

**<**[**https://www.tecmint.com/convert-files-to-utf-8-encoding-in-linux/**](https://www.tecmint.com/convert-files-to-utf-8-encoding-in-linux/)**>**

**<**[**https://conteudo.precocerto.co/margem-bruta-margem-liquida/#:~:text=A%20margem%20de%20lucro%20bruta,faturamento%20que%20%C3%A9%20lucro%20l%C3%ADquido**](https://conteudo.precocerto.co/margem-bruta-margem-liquida/#:~:text=A%20margem%20de%20lucro%20bruta,faturamento%20que%20%C3%A9%20lucro%20l%C3%ADquido)**.>**

# INFORMAÇÕES COMPLEMENTARES

## Ferramentas Utilizadas

* Microsoft SQL Server 2019
* Microsoft SQL Server Management Studio
* DBeaver
* Pentaho
* Driver MSSQL Server para o Pentaho:
  + <<https://sourceforge.net/projects/jtds/files/latest/download>>

## Evidências

### Restauração do Arquivo de Backup no MS SQL Server

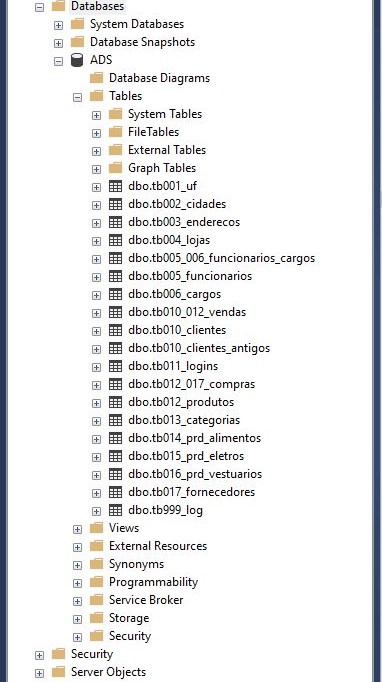
#### Importação Dados no MS SQL Server

****

#### Evidência da Execução do Script para Restaurar o Backup

****

#### Evidência da Base e Tabelas Criadas Após Restauração

****

# ANEXOS

# 

