Universidade Estadual de Londrina

Centro de Ciências Exatas
Departamento de Computação
Curso de Ciência da Computação
Tópicos em Computação

Relatório de construção do

Data Warehouse

Fernando Morgado Pires Neto
Gabriel Ângelo Perez Gasparini Sabaudo
Gabriela Tieko Hirashima
Guilherme Henrique Gonçalves Silva
Sophie Nascimento

Prof. Dr. Vitor Valério de Souza Campos

2022

Introdução

O presente documento busca apresentar de forma simples e clara em uma linguagem mais natural as funções dos scripts executados para os trabalhos requisitados no primeiro semestre da matéria de Tópicos em Computação.

Primeiramente, nos Capítulos 1 e 2 são apresentados todos os scripts necessários para a criação da base de dados do Data Stage e Data Warehouse, respectivamente.

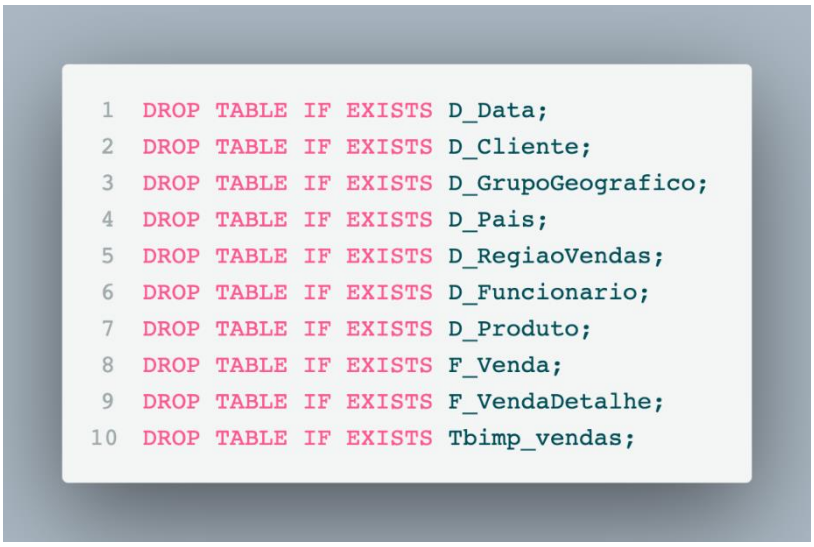
No Capítulo 3 são explicadas todas as transformações referentes a todas as Dimensões, Fatos e demais. Para cada uma delas é dita sua função de forma simples e direta, seguido da sequência de todos os passos, descritos em detalhe.

No penúltimo capítulo é definida a criação da Data Warehouse, explicando como cada parte foi executada e o porquê de sua existência. As dúvidas e questões levantadas durante a realização do trabalho são sumarizadas no penúltimo capítulo. O documento termina na conclusão que faz um apanhado do processo e descreve os frutos do trabalho.

Esquema do Banco de dados do DS

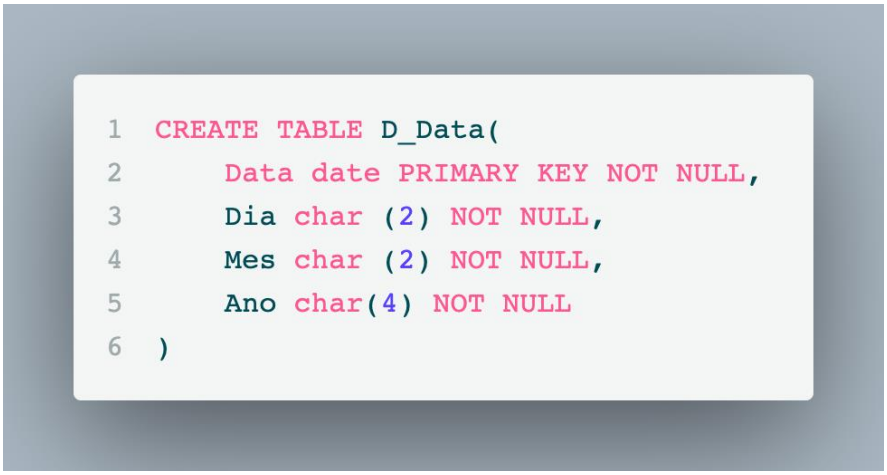
Abaixo estão os trechos de código em SQL relacionados com a construção do Data Storage em PostgreSQL.

Limpeza Inicial do Banco de Dados



```sql
DROP TABLE IF EXISTS D_Data;
DROP TABLE IF EXISTS D_Cliente;
DROP TABLE IF EXISTS D_GrupoGeografico;
DROP TABLE IF EXISTS D_Pais;
DROP TABLE IF EXISTS D_RegiaoVendas;
DROP TABLE IF EXISTS D_Funcionario;
DROP TABLE IF EXISTS D_Produto;
DROP TABLE IF EXISTS F_Venda;
DROP TABLE IF EXISTS F_VendaDetalhe;
DROP TABLE IF EXISTS Tbimp_vendas;
```

Tabela da dimensão Tempo



```sql
CREATE TABLE D_Data(
    Data date PRIMARY KEY NOT NULL,
    Dia char (2) NOT NULL,
    Mes char (2) NOT NULL,
    Ano char(4) NOT NULL
)
```

Tabela da dimensão Cliente

```sql
CREATE TABLE D_Cliente(
    Cod_Cliente varchar(10) NOT NULL,
    Nome varchar(50) NOT NULL,
    Email varchar(50) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL
);

CREATE INDEX IX_Cod_Cliente ON D_Cliente (Cod_Cliente);
```

Tabela da dimensão Geografia - Grupo Geográfico

```sql
CREATE TABLE D_GrupoGeografico(
    Nome varchar (50) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar (50) NOT NULL
);
CREATE INDEX IX_D_GrupoGeografico _nome ON D_GrupoGeografico (Nome);
```

Tabela da dimensão Geografia - País

```sql
CREATE TABLE D_Pais(
    Id_GrupoGeo int NOT NULL,
    Sigla char (2) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar (50) NOT NULL
);
create index IX_D_PaisIdGrupo on D_Pais (Id_GrupoGeo);
create index IX_D_PaisSigla on D_Pais (Sigla);
```

Tabela da dimensão Geografia - Região Vendas

```
CREATE TABLE D_RegiaoVendas(
    Id_Pais int NOT NULL,
    Nome varchar (20) NOT NULL,
    LinData date NOT NULL,
    Linorig varchar (50) NOT NULL
);
create index IX_D_RegiaoIdPais on D_RegiaoVendas (Id_Pais);
create index IX_D_RegiaoNome on D_RegiaoVendas (Nome);
```

Tabela da dimensão Funcionário

```
CREATE TABLE D_Funcionario(
    Nome varchar(50) NOT NULL,
    Login varchar(50) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL
);

CREATE INDEX IX_FuncionarioLogin ON D_Funcionario(Login);
```

Tabela da dimensão Produto

```
CREATE TABLE D_Produto(
    Cod_Produto varchar (20) NOT NULL,
    Nome varchar (50) NOT NULL,
    Tamanho varchar(5) NOT NULL,
    Cor varchar (20) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar (50) NOT NULL
);
create index IX_ProdutoNM on D_Produto (nome);
create index IX_ProdutoCod on D_Produto (Cod_Produto);
```

Tabela do fato Vendas

```
CREATE TABLE F_Venda(
    Data date NOT NULL,
    Nr_NF varchar (10) NOT NULL,
    Id_Cliente int NOT NULL,
    Id_Funcionario int NOT NULL,
    Id_RegiaoVendas int NOT NULL,
    Vlr_Imposto decimal(18, 2) NOT NULL,
    Vlr_Frete decimal(18, 2) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_F_Venda PRIMARY KEY(Data ,Nr_NF, Id_Cliente, Id_Funcionario, Id_RegiaoVendas)
);
```

Tabela do fato Vendas – Detalhe

```
CREATE TABLE F_VendaDetalhe(
    Data date NOT NULL,
    Nr_NF varchar(10) NOT NULL,
    Id_Cliente int NOT NULL,
    Id_Funcionario int NOT NULL,
    Id_RegiaoVendas int NOT NULL,
    Id Produto int NOT NULL,
    Vlr_Unitario decimal (18, 2) NOT NULL,
    Qtd_Vendida int NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar (50) NOT NULL,
    CONSTRAINT PK_F_VendaDetalhe PRIMARY KEY(
        Data,
        Nr_NF,
        Id_Cliente,
        Id Funcionario,
        Id_RegiaoVendas,
        Id Produto
    )
);
```

Tabela de importação das Vendas

```sql
CREATE TABLE Tbimp_vendas(
    nrnf character varying (15),
    datavenda character varvina (15),
    codcliente character varying(15),
    nomecliente character varying (50),
    emailcliente character varying(50),
    regiaovendas character varying(15),
    pais character varying(2),
    grupogeografico character varying(15),
    vendedorlogin character varying(24),
    vendedornome character varying (50),
    vendedorchefenome character varying(50),
    cod_produto character varying(30),
    produto character varying(50),
    tamanho character varying(6),
    Linha character varying(6),
    cor character varying(15),
    subtotal character varying(15),
    imptotal bigint,
    frete bigint,
    precounitario bigint,
    qtd bigint
);
```

Esquema do Banco de dados do DW

Abaixo estão os trechos de código em SQL relacionados com a construção do Data Warehouse em PostgreSQL.

Remoção das chaves estrangeiras das tabelas, caso existam

```sql
ALTER TABLE IF EXISTS D_Pais
    DROP CONSTRAINT
    IF EXISTS FK_D_GrupoGeografico;

ALTER TABLE IF EXISTS D_RegiaoVendas
    DROP CONSTRAINT
    IF EXISTS FK_D_Pais;

ALTER TABLE IF EXISTS D_Pais
    DROP CONSTRAINT
    IF EXISTS FK_D_Pais_D_GrupoGeografico;

ALTER TABLE IF EXISTS D_RegiaoVendas
    DROP CONSTRAINT
    IF EXISTS FK_D_RegiaoVendas_D_Pais;

ALTER TABLE IF EXISTS F_Venda
    DROP CONSTRAINT
    IF EXISTS FK_F_Venda_D_Cliente;

ALTER TABLE IF EXISTS F_Venda
    DROP CONSTRAINT
    IF EXISTS FK_F_Venda_D_Data;

ALTER TABLE IF EXISTS F_Venda
    DROP CONSTRAINT
    IF EXISTS FK_F_Venda_D_Funcionario;

ALTER TABLE IF EXISTS F_Venda
    DROP CONSTRAINT
    IF EXISTS FK_F_Venda_D_RegiaoVendas;

    ALTER TABLE IF EXISTS F_VendaDetalhe
    DROP CONSTRAINT
    IF EXISTS FK_F_VendaDetalhe_D_Produto;

ALTER TABLE IF EXISTS F_VendaDetalhe
    DROP CONSTRAINT
    IF EXISTS FK_F_VendaDetalhe_F_Venda;
```

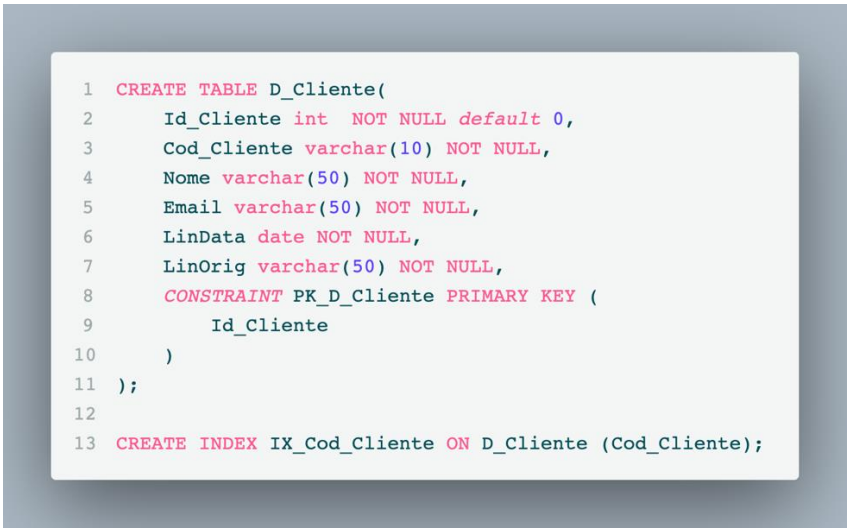
Limpeza inicial do banco de dados

```sql
DROP TABLE IF EXISTS D_Data;
DROP TABLE IF EXISTS D_Cliente;
DROP TABLE IF EXISTS D_GrupoGeografico;
DROP TABLE IF EXISTS D_Pais;
DROP TABLE IF EXISTS D_RegiaoVendas;
DROP TABLE IF EXISTS D_Funcionario;
DROP TABLE IF EXISTS D_Produto;
DROP TABLE IF EXISTS F_Venda;
DROP TABLE IF EXISTS F_VendaDetalhe;
DROP TABLE IF EXISTS Tbimp_vendas;
```

Tabela da dimensão Tempo

```sql
CREATE TABLE D_Data(
    Id_Data int  NOT NULL default 0,
    Data date NOT NULL,
    Dia char(2) NOT NULL,
    Mes char(2) NOT NULL,
    Ano char(4) NOT NULL,
 CONSTRAINT PK_D_Data PRIMARY KEY
(
    Data
)
);
```

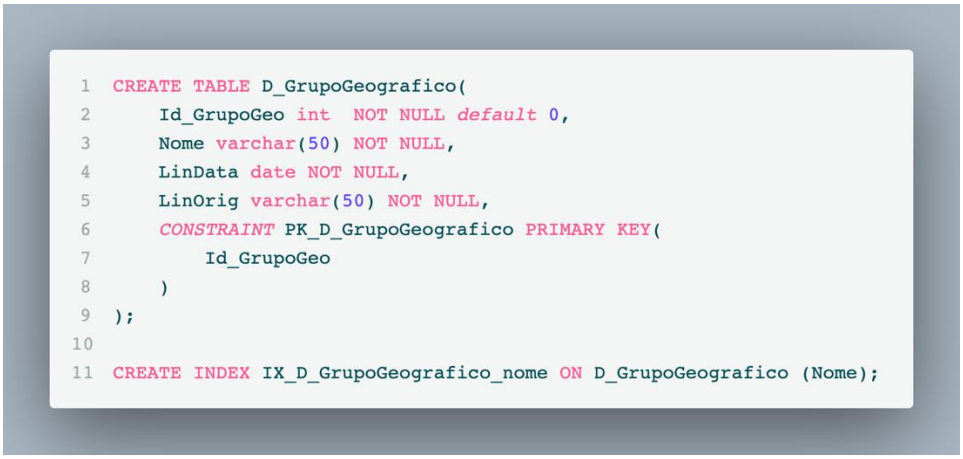
Tabela da dimensão Cliente



```sql
CREATE TABLE D_Cliente(
    Id_Cliente int  NOT NULL default 0,
    Cod_Cliente varchar(10) NOT NULL,
    Nome varchar(50) NOT NULL,
    Email varchar(50) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_D_Cliente PRIMARY KEY (
        Id_Cliente
    )
);

CREATE INDEX IX_Cod_Cliente ON D_Cliente (Cod_Cliente);
```

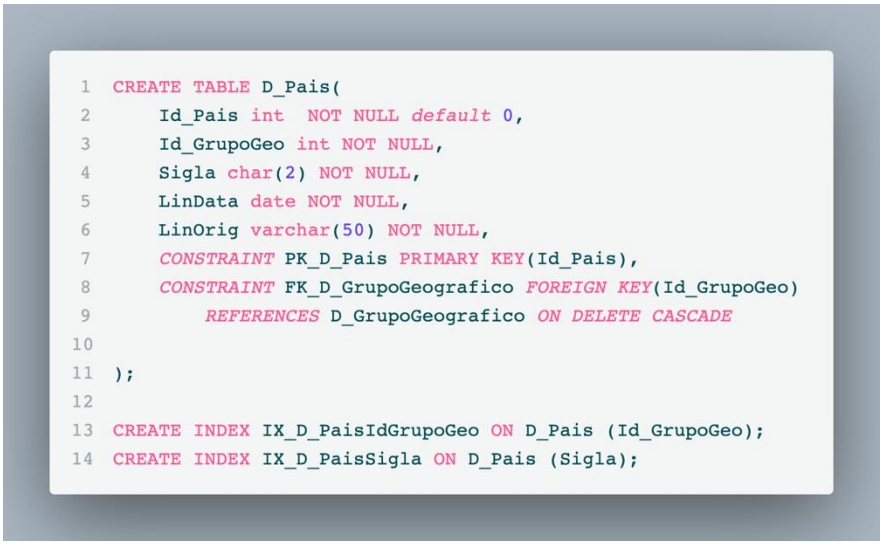
Tabela da dimensão Geografia – Grupo Geográfico



```sql
CREATE TABLE D_GrupoGeografico(
    Id_GrupoGeo int  NOT NULL default 0,
    Nome varchar(50) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_D_GrupoGeografico PRIMARY KEY(
        Id_GrupoGeo
    )
);

CREATE INDEX IX_D_GrupoGeografico_nome ON D_GrupoGeografico (Nome);
```

Tabela da dimensão Geografia – País



```sql
CREATE TABLE D_Pais(
    Id_Pais int  NOT NULL default 0,
    Id_GrupoGeo int NOT NULL,
    Sigla char(2) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_D_Pais PRIMARY KEY(Id_Pais),
    CONSTRAINT FK_D_GrupoGeografico FOREIGN KEY(Id_GrupoGeo)
        REFERENCES D_GrupoGeografico ON DELETE CASCADE

);

CREATE INDEX IX_D_PaisIdGrupoGeo ON D_Pais (Id_GrupoGeo);
CREATE INDEX IX_D_PaisSigla ON D_Pais (Sigla);
```

Tabela da dimensão Geografia – Região Vendas

```sql
CREATE TABLE D_RegiaoVendas(
    Id_RegiaoVendas int  NOT NULL default 0,
    Id_Pais int NOT NULL,
    Nome varchar(20) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_D_RegiaoVendas PRIMARY KEY(Id_RegiaoVendas),
    CONSTRAINT FK_D_Pais FOREIGN KEY(Id_Pais)
        REFERENCES D_Pais ON DELETE CASCADE
);

CREATE INDEX IX_D_RegiaoIdPais ON D_RegiaoVendas (Id_Pais);
CREATE INDEX IX_D_RegiaoNome   ON D_RegiaoVendas (Nome);
```

Tabela da dimensão Funcionário

```sql
CREATE TABLE D_Funcionario(
    Id_Funcionario int  NOT NULL default 0,
    Nome varchar(50) NOT NULL,
    Login varchar(50) NOT NULL,
    Id_Chefe int NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_D_Funcionario PRIMARY KEY(Id_Funcionario)
);

CREATE INDEX IX_FuncionarioLogin ON D_Funcionario(Login);
```

Tabela da dimensão Produto

```sql
CREATE TABLE D_Produto(
    Id_Produto int NOT NULL default 0,
    Cod_Produto varchar(20) NOT NULL default 0,
    Nome varchar(50) NULL,
    Tamanho varchar(5) NULL,
    Cor varchar(20) NULL,
    LinData date NULL,
    LinOrig varchar(50) NULL,
    version int NULL,
    date_from timestamp NULL,
    date_to timestamp NULL,
    CONSTRAINT PK_D_Produto PRIMARY KEY(Id_Produto)
);

CREATE INDEX IX_ProdutoNM ON D_Produto (nome);
CREATE INDEX IX_ProdutoCod ON D_Produto (Cod_Produto);
```

Tabela do fato Vendas

```sql
CREATE TABLE F_Venda(
    Id_Fato int  NOT NULL,
    Data date NOT NULL,
    Nr_NF varchar (10) NOT NULL,
    Id_Cliente int NOT NULL,
    Id_Funcionario int NOT NULL,
    Id_RegiaoVendas int NOT NULL,
    Vlr_Imposto decimal(18, 2) NOT NULL,
    Vlr_Frete decimal(18, 2) NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_F_Venda PRIMARY KEY(
        Data ,
        Nr_NF ,
        Id_Cliente ,
        Id_Funcionario ,
        Id_RegiaoVendas)
);
```

Tabela do fato Vendas – Detalhe

```sql
CREATE TABLE F_VendaDetalhe(
    Id_FatoDetalhe int  NOT NULL,
    Data date NOT NULL,
    Nr_NF varchar (10) NOT NULL,
    Id_Cliente int NOT NULL,
    Id_Funcionario int NOT NULL,
    Id_RegiaoVendas int NOT NULL,
    Id_Produto int NOT NULL,
    Vlr_Unitario decimal(18, 2) NOT NULL,
    Qtd_Vendida int NOT NULL,
    LinData date NOT NULL,
    LinOrig varchar(50) NOT NULL,
    CONSTRAINT PK_F_VendaDetalhe PRIMARY KEY(
        Data ,
        Nr_NF ,
        Id_Cliente ,
        Id_Funcionario ,
        Id_RegiaoVendas ,
        Id_Produto)
);
```

Adição das chaves estrangeiras nas tabelas

```
ALTER TABLE D_Pais ADD CONSTRAINT FK_D_Pais_D_GrupoGeografico
    FOREIGN KEY(Id_GrupoGeo)
    REFERENCES D_GrupoGeografico (Id_GrupoGeo)
    ON DELETE CASCADE;


ALTER TABLE D_RegiaoVendas ADD CONSTRAINT FK_D_RegiaoVendas_D_Pais
    FOREIGN KEY(Id_Pais)
    REFERENCES D_Pais (Id_Pais)
    ON DELETE CASCADE;


ALTER TABLE F_Venda ADD  CONSTRAINT FK_F_Venda_D_Cliente
    FOREIGN KEY(Id_Cliente)
    REFERENCES D_Cliente (Id_Cliente)
    ON DELETE CASCADE;


ALTER TABLE F_Venda ADD  CONSTRAINT FK_F_Venda_D_Data
    FOREIGN KEY(Data)
    REFERENCES D_Data (Data)
    ON DELETE CASCADE;


ALTER TABLE F_Venda ADD CONSTRAINT FK_F_Venda_D_Funcionario
    FOREIGN KEY(Id_Funcionario)
    REFERENCES D_Funcionario (Id_Funcionario)
    ON DELETE CASCADE;


ALTER TABLE F_Venda ADD  CONSTRAINT FK_F_Venda_D_RegiaoVendas
    FOREIGN KEY(Id_RegiaoVendas)
    REFERENCES D_RegiaoVendas (Id_RegiaoVendas)
    ON DELETE CASCADE;


ALTER TABLE F_VendaDetalhe ADD  CONSTRAINT FK_F_VendaDetalhe_D_Produto
    FOREIGN KEY(Id_Produto)
    REFERENCES D_Produto (Id_Produto)
    ON DELETE CASCADE;


ALTER TABLE F_VendaDetalhe ADD CONSTRAINT FK_F_VendaDetalhe_F_Venda
    FOREIGN KEY(Data, Nr_NF,Id_Cliente, Id_Funcionario, Id_RegiaoVendas)
    REFERENCES F_Venda (Data, Nr_NF,Id_Cliente, Id_Funcionario, Id_RegiaoVendas)
    ON DELETE CASCADE;
```

Descrição das Execuções

- **Transformação 01 – Importação de dados para o Data Stage:**

    - **O que faz:** Carrega todos os dados do arquivo de entrada (1), seleciona e renomeia os campos (2) e insere no banco de dados do DS (3).

    - Passos:
        - **(1) Text file input:** Lê os dados do arquivo CSV (massadados2005.csv)
        - **(2) Select values:** Seleciona e renomeia todos os campos que foram lidos no passo anterior
        - **(3) Table output:** É realizada a conexão com o banco de dados para carregar os atributos que vêm do fluxo para a tabela TbImp_Vendas do DS.

    - Número de registros inseridos: 5151 (Cinco mil cento e cinquenta e um).

- **Transformação 02 – Dimensão Tempo para o Data Stage:**

    - **O que faz:** Lê os dados da tabela TbImp_Vendas do DS (1), recebe a string DataVenda, formatando a data por ano, mês e dia (2). É feita a substituição de '.' por '– '(3). Após isso, seleciona os valores necessários (4), verifica a existência de nulos e preenche caso encontre (5), ordena os registros (6), elimina os duplicados (7) e salva as modificações no banco DS (8).

    - Passos:
        - **(1) Text file input:** Obtém os nomes dos campos da tabela TbImp_Vendas no banco de dados DS
        - **(2) Strings cut:** Recebe a string DataVenda, e a separa por ano, mês e dia. Ele começa a contagem a partir de 0 e vai a até 10 exclusive.
        - **(3) Replace in string:** Recebe a string DataVenda, e procura na string os '.' para substituir por '– '.
        - **(4) Select values BI:** Na aba Meta-data, clicando no botão "get fields to change" se obtém 4 campos, além disso são removidos os campos não pertencentes a dimensão Tempo, alterando também o campo DataVenda para o tipo Date com tamanho 10.
        - **(5) If field value if null:** Verifica se os campos DataVenda, ano, mês e dia são nulos. Caso sejam, insere respectivamente os valores "1900/01/01", "1900", "01" e "01".
        - **(6) Sort rows:** É feita a ordenação das linhas de acordo com o campo DataVenda de forma ascendente.
        - **(7) Unique rows:** Elimina linhas duplicadas dos campos DataVenda, a partir do ano, mês e dia.

        - **(8) Table output:** Utilizado para criar uma conexão entre o DS e o banco de dados, associando os dados que vem do passo anterior à tabela alvo D_Data.

    - Número de registros inseridos: 181 (Cento e oitenta e um).

- **Transformação 03 – Dimensão Tempo para o Data Warehouse:**

    - **O que faz:** Lê os dados da tabela D_Data do DS (1) e insere os valores na tabela D_Data do DW caso não existam, gerando um ID incremental para cada registro (2).

    - Passos:
        - **(1) Table input:** Obtém os nomes dos campos da tabela D_Data no banco de dados DS.
        - **(2) Select values:** Na aba Meta-data, clicando no botão “get fields to change” se obtém 4 campos, além disso são removidos os campos não pertencentes a dimensão Tempo, alterando também o campo DataVenda para o tipo Date com tamanho 10.
        - **(3) Combination lookup/update:** Insere os valores obtidos até aqui na tabela alvo (D_Data do Data Warehouse), realizando algumas verificações de duplicação e criando o campo Id_Data, que pode ser incrementado.

    - Número de registros inseridos: 181 (Cento e oitenta e um).

- **Transformação 04 – Dimensão Cliente para o Data Stage:**

    - **O que faz:** Lê os dados da tabela TbImp_Vendas do DS, selecionando os valores pertinentes. Em seguida, utiliza a data e hora do sistema para preencher o campo LinData e define o campo LinOrig como “Arquivo de Vendas”. Então, verifica se algum campo é nulo e o preenche com valores padrões. Por fim, ordena as entradas e elimina instâncias duplicadas.

    - Passos:
        - **(1) Table input:** Contém os dados de entrada pertinentes para a realização das operações através de um comando SELECT tendo como alvo a tabela TbImp_Vendas do banco DS.
        - **(2) Select values:** Seleciona os campos CodCliente, NomeCliente e EmailCliente, renomeando-os para Cod_Cliente, Nome e Email respectivamente.
        - **(3) Get system info:** É feito o preenchimento do campo LinData utilizando a data do sistema (system date), informando a data em que ocorreu a carga de dados.
        - **(4) Add constants:** É adicionado a String “Arquivo de Vendas” no campo LinOrig.

        - **(5) If field value if null:** Verifica se os campos Cod_Cliente, Nome, Email e LinOrig são nulos. Caso sejam, insere respectivamente os valores 999999, “Não Aplica”, “Não Aplica” e “Registro padrão inserido manualmente”.
        - **(6) Sort rows:** Neste passo, é executada a ordenação das linhas de acordo com o campo Cod_Cliente de forma ascendente.
        - **(7) Unique rows:** Elimina entradas com campos Cod_Cliente, Nome, Email, LinData e LinOrig duplicados.
        - **(8) Table output:** Cria uma conexão entre o DS e o banco de dados, associando os dados que vem do passo anterior à tabela alvo D_Cliente.

    - Número de registros inseridos: 1216 (mil duzentos e dezesseis).

- **Transformação 05 – Dimensão Cliente para o Data Warehouse:**

    - **O que faz:** Lê os dados da tabela D_Cliente do DS e insere os valores na tabela D_Cliente do DW, gerando um Id_Cliente incremental para cada registro.

    - Passos:
        - **(1) Table input:** Obtém os nomes dos campos da tabela D_Cliente no DS.
        - **(2) Combination lookup/update:** Insere os valores obtidos até aqui na tabela D_Cliente do DW, realizando algumas verificações de duplicação e criando o campo Id_Cliente como uma chave incremental.

    - Número de registros inseridos: 1216 (Mil duzentos e dezesseis).

- **Transformação 06 – Dimensão Geografia (Grupo Geográfico) para o DS:**

    - **O que faz:** Lê os dados da tabela TbImp_Vendas do DS (1), seleciona os valores necessários (2) e preenche os campos que faltam ser preenchidos (3) e (4). Verifica a existência de nulos e preenche caso encontre (5). Por fim, ordena os registros (6), elimina os duplicados (7) e salva as modificações no banco DS (8).

    - Passos:
        - **(1) Table Input:** É feita uma conexão com o banco de dados DS para buscar os campos desejados da tabela TbImp_Vendas, por meio de um SQL de SELECT.
        - **(2) Select Values:** São selecionados (levados para o fluxo) apenas os campos que pertencem a tabela D_GrupoGeografico. Nesse caso específico, nenhum campo é renomeado.
        - **(3) Get System Info:** Neste passo é feito o preenchimento dos campos que necessitam de algum valor do sistema. No caso o campo LinData é preenchido com a data do sistema, informando a data em que ocorreu a carga de dados.

- **(4) Add constants:** Aqui são preenchidos os campos que terão seus valores constantes. No caso o campo LinOrig recebe o valor “Arquivo de Vendas”.
- **(5) If field value is null:** Agora, é feita a verificação de campos com os valores nulos, e, caso existam, são definidos outros valores para esses campos. No caso, o campo GrupoGeografico e LinOrig recebem, respectivamente, os valores “Não Aplica” e “Registro padrão inserido manualmente”.
- **(6) Sort rows:** Esse passo ordena os registros de acordo com determinados atributos. No caso, a tributo é o GrupoGeografico, de forma ascendente.
- **(7) Unique rows:** Agora, são eliminados os registros duplicados de acordo com determinados atributos. No caso, o atributo é o GrupoGeografico.
- **(8) Table output:** Nesse passo é feita a conexão com o banco de dados, para carregar os atributos que vêm do fluxo à tabela D_GrupoGeografico do DS.

- Número de registros inseridos: 3 (três)

● Transformação 07 – Dimensão Geografia (Grupo Geográfico) para o DW:

- **O que faz:** Lê os dados da tabela D_GrupoGeografico do DS (1) e insere os valores na tabela D_GrupoGeografico do DW caso não existam, gerando um ID incremental para cada registro (2).

- Passos:
  - **(1) Table input:** Cria a conexão com o DS para obter os campos da tabela D_GrupoGeografico.
  - **(2) Combination lookup/update:** Insere os valores na tabela D_GrupoGeografico do DW, somente se já não existirem. Também é inserido o valor da chave primária, que é incremental. Nesse caso, a chave primária é o Id_GrupoGeo.

- Número de registros inseridos: 3 (três).

● Transformação 08 – Dimensão Geografia (País) para o DS:

- **O que faz:** Lê os dados da tabela TbImp_Vendas do DS (1), faz uma busca na tabela D_GrupoGeografico (2), seleciona os valores necessários (3) e preenche os campos que faltam ser preenchidos (4) e (5). Verifica a existência de nulos e preenche caso encontre (6). Por fim, ordena os registros (7), elimina os duplicados (8), faz novamente uma seleção dos valores necessários (9) e salva as modificações no banco DS (10).

- Passos:
  - **(1) Table input:** Por meio de um SQL, um SELECT preenche os campos desejados. Possui os mesmos dados de entrada da dimensão tempo.

- **(2) Combination lookup/update:** Database lookup: Faz uma busca de valores na tabela do Banco de dados, nesse caso a tabela é a D_GrupoGeografico.
- **(3) Select values:** Na aba Meta-data, clicando no botão “get fields to change” se obtém 4 campos, além disso são removidos os campos não pertencentes a tabela GrupoGeografico da dimensão Geografia.
- **(4) Get System Info:** Neste passo é feito o preenchimento do campo LinData utilizando a data do sistema (system date), informando a data em que ocorreu a carga de dados.
- **(5) Add constraints:** É adicionado a String “Arquivo de Vendas” no campo LinOrig.
- **(6) if field value if null:** Verifica se os campos Pais, Id_GrupoGeo e LinOrig são nulos. Caso sejam, insere respectivamente os valores “ZZ”, “1” e “Registro padrão inserido manualmente”.
- **(7) Sort rows:** É feita a ordenação das linhas de acordo com o campo Pais de forma ascendente.
- **(8) Unique rows:** Elimina linhas duplicadas.
- **(9) Select values 2:** Na aba Meta-data, clicando no botão “get fields to change” se obtém 4 campos, além disso são removidos os campos não pertencentes à tabela D_Pais da dimensão Geografia.
- **(10) Table output:** Utilizado para criar uma conexão entre o DS e o banco de dados, associando os dados que vem do passo anterior à tabela alvo D_Pais.

- Número de registros inseridos: 6 (seis).

● Transformação 09 – Dimensão Geografia (País) para o DW:

- **O que faz:** Lê os dados da tabela D_Pais do DS (1) e insere os valores na tabela D_Pais do DW caso não existam, gerando um ID incremental para cada registro (2).

- Passos:
  - **(1) Table input:** Obtém os nomes dos campos da tabela D_Pais no banco de dados DS.
  - **(2) Combination lookup/update:** Insere os valores obtidos até aqui na tabela alvo (D_Pais do Data Warehouse), realizando algumas verificações de duplicação e criando o campo Id_Pais que pode ser incrementado.

- Número de registros inseridos: 6 (seis)

● Transformação 10 – Dimensão Geografia (Região Vendas) para o DS

- **O que faz:** Seleciona os dados da tabela TbImp_Vendas do DS (1). Em seguida, seleciona os dados Id_Pais da tabela D_Pais no DW (2, 3). Checa se existem valores nulos (alterando-os para os respectivos valores) (4), ordena as instâncias e elimina valores duplicados (5, 6). Seleciona os campos da tabela D_RegiaoVendas do DS (7), altera o nome do campo Regiao_Vendas e o salva no banco DS (8).

- Passos**:**
  - **(1) Table input**: Dados de entrada necessários para realizar usando um comando SELECT, tendo como alvo a tabela TbImp_Vendas do banco DS.
  - **(2) Select values**: Seleciona campos de RegiaoVendas e Pais.
  - **(3) Database lookup**: Utiliza o campo Pais para comparar com o campo Sigla da tabela D_Pais do DW. Caso sejam iguais, busca o valor de Id_Pais.
  - **(4) If field value is null:** Verifica se os campos RegiaoVendas, Id_Pais e LinOrig são nulos. Caso os valores sejam nulos, insere os valores "ZZ" em RegiaoVendas, 1 em Id e "Registro padrão inserido manualmente" em LinOrig.
  - **(5) Sort rows:** Neste passo, ordena RegiaoVendas por ascendencia.
  - **(6) Unique rows:** Elimina entradas de RegiaoVendas duplicadas.
  - **(7) Select values 2:** Seleciona Regiao_Vendas, Id_Pais, LinOrig e LinData, e altera o campo de Regiao_Vendas para nome.
  - **(8) Table output:** Cria uma conexão entre o DS e o banco de dados, associando os dados que vem do passo anterior à tabela alvo D_RegiaoVendas.

- Número de registros inseridos: 10 (dez) registros.

● **Transformação 11 – Dimensão Geografia (Região Vendas) para o DW:**

- **O que faz:** Coleta os dados da tabela D_RegiaoVendas do DS (1) inserindo nos valores na tabela D_RegiaoVendas do DW, após, cria um Id_RegiaoVendas que é incrementado a cada registro (2).

- Passos**:**
  - **(1) Table Input:** Dados da tabela D_Cliente no Data Stage.
  - **(2) Combination lookup/update:** Insere os valores obtidos na tabela alvo (D_RegiaoVendas do Data Warehouse), verifica duplicação de instâncias e após, cria um campo Id_RegiaoVendas que é incrementado a cada registro.

- Número de registros inseridos**:** 10 (dez) registros.

● **Transformação 12 – Dimensão Funcionário para o DS:**

- **O que faz:** Coleta os dados da tabela TbImp_Vendas do DS (1), seleciona os valores que serão trabalhados (2) e preenche os campos que possuem algum valor faltando (3) e (4). Quando um valor nulo é encontrado, o preenche com outro valor (5). Ordena os registros de acordo com seus atributos (6), elimina instâncias duplicadas (7) e salva as modificações no banco DS a partir de uma conexão com o banco de dados (8).

- Passos:

- **(1) Table Input:** Conexão com o banco de dados DS para coletar dados da tabela TbImp_Vendas, por meio de um SQL SELECT.
- **(2) Select Values:** Os campos pertencentes a tabela D_Funcionario são selecionados. Nesse caso, não há alterações.
- **(3) Get System Info:** Os campos que possuem valores faltando são preenchidos neste passo. O campo LinData, por exemplo, recebe a data do sistema, informando a data em que ocorreu a carga de dados.
- **(4) Add constants:** Os campos que possuem valores faltando e terão seus valores constantes são preenchidos neste passo. O campo LinOrig recebe o valor “Arquivo de Vendas”.
- **(5) If field value is null**: Checagem de campos com valores nulos, se existirem. Neste passo são definidos outros valores para os campos. Os campos são definidos da seguinte forma: VendedorLogin -> “Não Aplica”, VendedorNome -> “Não Aplica” e LinOrig -> “Registro padrão inserido manualmente”.
- **(6) Sort rows:** Neste passo os registros são ordenados de acordo com os atributos. O utilizado para ordenar é o VendedorLogin, de forma ascendente.
- **(7) Unique rows:** Os registros duplicados são deletados de acordo com atributos. O atributo para este passo é o VendedorLogin.
- **(8) Table output:** Uma conexão com o banco de dados é realizada para carregar os atributos que vêm do fluxo à tabela D_Funcionario do DS.

- Número de registros inseridos**:** 11 (onze) registros.

● Transformação 13 – Dimensão Funcionário para o DW:

- **O que faz:** Coleta dados da tabela D_Funcionario do DS (1) e insere valores na tabela D_Funcionario do DW se não existirem, gerando um ID que incrementa a cada instância (2).

- Passos**:**
  - **(1) Table input:** Cria a conexão com o DS para coletar os campos da tabela D_Funcionario.
  - **(2) Combination lookup/update:** Insere os valores na tabela D_Funcionario do DW, quando não existirem. Também é inserido o valor da chave primária, que incrementa a cada instância. A chave primária é o Id_Funcionario.

- Número de registros inseridos**:** 11 (onze) registros.

● Transformação 14 – Update ID Chefe do Funcionário:

- **O que faz:** Para realizar o update, é preciso buscar o Id do Chefe gerado anteriormente. A transformação lê os dados da tabela D_Funcionario do DW (1), e verifica o nome do chefe do funcionário na tabela TbImp_Vendas (2). Sabendo o nome, é possível buscar o ID do chefe na tabela D_Funcionario

do DW (3). Seleciona os valores necessários (4) e filtra os resultados para checar se o funcionário realmente está subordinado àquele chefe (5). Por fim, atualiza o campo Id_Chefe nos registros da tabela D_Funcionario do DW (6).

- Passos**:**
  - **(1) Table input:** Ccom o banco de dados DW para obter os dados de D_Funcionario que foram carregados anteriormente.
  - **(2) Database lookup:** Verifica a igualdade entre os campos de chave primária do D_Funcionario (Login) e TbImp_Vendas para recuperar o VendedorChefeNome.
  - **(3) Database lookup 2:** Sabendo do campo VendedorChefeNome, é possível verificar D_Funcionario, para obter o Id_Funcionario do chefe correspondente ao nome encontrado (VendedorChefeNome).
  - **(4) Select values:** Os campos que serão necessários são selecionados para fazer a atualização de Id_Chefe.
  - **(5) Filter rows:** Esse passo garante que o registro correto seja atualizado, filtrando os registros em que o VendedorChefeNome corresponde ao NomeChefeDW. Caso não sejam correspondentes, o fluxo segue para um passo Dummy, que não possui instruções de alteração.
  - **(6) Update:** Por fim, o Id_Chefe do usuário igual ao Id_Funcionario é atualizado na tabela D_Funcionario do DW.

- Número de registros atualizados: 8 (oito) registros.

● **Transformação 15 – Dimensão Produto para o DS:**

- **O que faz:** Coleta dados da tabela TbImp_Vendas do DS (1), seleciona os valores necessários (2) e preenche os campos que possuem algum valor faltando (3) e (4). Checa se existem nulos e os preenche caso encontre (5). No final, ordena os registros (6), elimina os duplicados (7), e salva as modificações no banco DS (8).

- Passos**:**
  - **(1) Table input:** Por meio de um SQL SELECT coleta os campos desejados.
  - **(2) Select values:** Os campos necessários são selecionados, excluído os não pertencentes a dimensão Tempo. O campo DataVenda é alterado para Date de tamanho 10.
  - **(3) Get System Info:** Neste passo, os campos que possuem valores faltados são substituídos. Em LinData, a data do sistema (system date) é inserida, informando a data em que ocorreu a carga de dados.
  - **(4) Add constraints:** Neste passo, os campos que possuem valores faltando são substituídos. Em LinOrig -> “Arquivo de Vendas” é inserido.
  - **(5) if field value if null:** Checa se Cod_Produto, Nome, Tamanho, Cor e LinOrig são nulos. Caso sejam, Cod_Produto -> “999999”, Nome -> “Não

Aplica", Tamanho-> "NA", Cor-> "NA" e LinOrig-> "Registro padrão inserido manualmente" são inseridos nos campos vazios.

- **(6) Sort rows:** Ordenação das linhas é realizada de acordo com os campos Cod_Produto e Nome de forma ascendente.
- **(7) Unique rows:** Elimina instâncias repetidas dos campos Cod_Produto e Nome.
- **(8) Table output:** Cria uma conexão entre o DS e o banco de dados, carregando dados na tabela alvo D_Produto.

- Número de registros inserido**s:** 60 (sessenta) registros.

● **Transformação 16 – Dimensão Produto para o DW:**

- **O que faz:** O que faz: Lê os dados da tabela D_Produto do DS (1) e insere os valores na tabela D_Produto do DW caso não existam, gerando um ID incremental para cada registro (2).

- Passos:
  - **(1) Table input:** Obtém os nomes dos campos da tabela D_Produto no banco de dados DS.
  - **(2) Select values: Combination lookup/update:** Insere os valores obtidos até aqui na tabela alvo (D_Produto do Data Warehouse), realizando algumas verificações de duplicação e criando o campo Id_Produto que pode ser incrementado.

- Número de registros inseridos: 60 (sessenta).

● **Transformação 17 – Fato Venda para o DS:**

- **O que faz:** Seleciona valores iniciais da tabela TbImp_Vendas para poder buscar os Ids correspondentes das tabelas do DW. Além disso, realiza as operações padrão de ordenação, verificação de nulo e, por fim, integração com o banco de dados.

- Passos:
  - **(1) Table input:** Contém os dados de entrada pertinentes para a realização das operações através de um comando SELECT tendo como alvo a tabela TbImp_Vendas do banco DS.
  - **(2) Replace in string:** Substitui "." por "–" da mesma forma que na dimensão Tempo.
  - **(3) Select values:** Seleciona os campos NrNf, DataVenda, CodCliente, RegiaoVendas, VendedorLogin, ImpTotal e Frete.
  - **(4) Sort rows:** Ordena os valores de forma ascendente de acordo com o campo RegiaoVendas.
  - **(5) Group by 2:** Os campos ImpTotal e Frete são agrupados pela função de soma.

- **(6) Database lookup:** Utiliza o campo CodCliente para comparar com o campo Cod_Cliente da tabela D_Cliente do DW. Caso sejam iguais, busca o valor de Id_Cliente.
- **(7) Database lookup 2:** Utiliza o campo VendedorLogin para comparar com o campo login da tabela D_Funcionario do DW. Caso sejam iguais, busca o valor de Id_Funcionario.
- **(8) Database lookup 3:** Utiliza o campo RegiaoVendas para comparar com o campo Nome da tabela D_RegiaoVendas do DW. Caso sejam iguais, busca o valor de Id_RegiaoVendas.
- **(9) Select values 2:** Seleciona os campos DataVenda (renomeado para Data), NrNf (renomeado para Nr_Nf), Id_Cliente, Id_Funcionario, Id_RegiaoVendas, ImpTotal e Frete.
- **(10) Get system info:** tem a mesma informação da transformação cliente e ela é armazenada na tabela F_Venda.
- **(11) Add constants:** tem a mesma informação da transformação cliente e ela é armazenada na tabela F_Venda.
- **(12) If field value is null:** Verifica se algum dos campos selecionados é nulo. Caso seja, insere um valor padrão.
- **(13) Insert/update:** Associa os atributos que vem do fluxo com a tabela F_Venda, inserindo ou atualizando conforme o necessário.

○ Número de registros inseridos: 1382 (mil trezentos e oitenta e dois).

● **Transformação 18 – Fato Venda para o DW:**

- **O que faz:** Busca os dados da tabela F_Venda do DS, seleciona os campos desejados e insere os valores na tabela F_Venda do DW, criando um ID incremental para cada instância.

- Passos:
  - **(1) Table input:** Contém os campos da tabela da tabela F_Vendas no DS.
  - **(2) Select values:** Seleciona todos os campos da tabela.
  - **(3) Combination lookup/update:** Insere os valores na tabela F_Venda do DW. Além disso, cria um campo Id_Fato que funciona como um incremental.

- Número de registros inseridos: 1379 (mil trezentos e setenta e nove).

● **Transformação 19 – Fato Venda Detalhe para o DS:**

- **O que faz:** Lê os dados da tabela TbImp_Vendas do DS (1), substitui '.' por '-' nos campos que representam uma data (2) e seleciona os valores necessários (3). Então, os registros são ordenados (4) e agrupados (5). Em seguida, busca os ID's das chaves estrangeiras que referenciam as tabelas D_Cliente (6), D_Funcionario (7), D_RegiaoVendas (8) e D_Produto (9). Após isso, seleciona os valores necessários (10) e preenche os campos que

faltam (11) e (12). Por fim, verifica a existência de nulos, preenche caso encontre (13), e salva as modificações no banco DS (14).

- Passos:
  - **(1) Table input:** É feita uma conexão com o banco de dados DS para buscar os campos desejados da tabela TbImp_Vendas, por meio de um SQL de SELECT.
  - **(2) Replace in string:** Esse passo simplesmente substitui caracteres por outros. No caso, substitui '.' por '-' em DataVenda, para encaixar no formato date do banco de dados.
  - **(3) Select values:** São selecionados (levados para o fluxo) apenas os campos que pertencem a tabela F_VendaDetalhe. Nesse caso, nenhum campo é renomeado.
  - **(4) Sort rows:** Esse passo ordena os registros de acordo com determinados atributos. No caso, o atributo é o Cod_Produto, de forma ascendente.
  - **(5) Group by 2:** Aqui são escolhidos os atributos para serem agrupados, e os atributos que vão ser agregados (e a função de agregação). No caso, os campos PrecoUnitario e Qtd serão agregados pela função de soma.
  - **(6) Database lookup:** Busca na tabela D_Cliente do DW o atributo Id_Cliente, a partir da igualdade de CodCliente.
  - **(7) Database lookup 2:** Busca na tabela D_Funcionario do DW o atributo Id_Funcionario, a partir da igualdade do Login.
  - **(8) Database lookup 3:** Busca na tabela D_RegiaoVendas do DW o atributo Id_RegiaoVendas, a partir da igualdade do nome da região.
  - **(9) Database lookup 4:** Busca na tabela D_Produto do DW o atributo Id_Produto, a partir da igualdade de CodProduto.
  - **(10) Select values 2:** São selecionados (levados para o fluxo) apenas os campos que pertencem a tabela F_VendaDetalhe. Nesse caso, o atributo DataVenda é renomeado para Data.
  - **(11) Get System Info:** Neste passo é feito o preenchimento dos campos que necessitam de algum valor do sistema. No caso o campo LinData é preenchido com a data do sistema, informando a data em que ocorreu a carga de dados.
  - **(12) Add constants:** Aqui são preenchidos os campos que terão seus valores constantes. No caso o campo LinOrig recebe o valor "Arquivo de Vendas".
  - **(13) If field value is null:** Agora, é feita a verificação de campos com os valores nulos, e, caso existam, são definidos outros valores para esses campos. No caso os campos receberão os seguintes valores:
    - Data = 1900/01/01
    - Id_Cliente = 1
    - Id_Funcionario = 1
    - Id_RegiaoVendas = 1
    - Id_Produto = 1
    - LinOrig = Registro padrão inserido manualmente

- NrNf = 0
- PrecoUnitario = 0
- Qtd = 0

▪ **(14) Insert / Update:** Associa os atributos que vem do fluxo com os atributos da tabela F_VendaDetalhe que vão ser inseridos / atualizados.

o Número de registros inseridos: 5151 (cinco mil cento e cinquenta e um).

● **Transformação 20 – Fato Venda Detalhe para o DW:**

o **O que faz:** Lê os dados da tabela F_VendaDetalhe do DS (1), seleciona os campos necessários (3) e insere os valores na tabela F_VendaDetalhe do DW caso não existam, gerando um ID incremental para cada registro (3).

o Passos:
▪ **(1) Table input:** Cria a conexão com o DS para obter os campos da tabela F_VendaDetalhe.
▪ **(2) Select values:** São selecionados (levados para o fluxo) apenas os campos que pertencem a tabela F_DetalheVenda. Nesse caso específico, nenhum campo é renomeado.
▪ **(3) Combination lookup/update:** Insere os valores na tabela F_DetalheVenda do DW, somente se já não existirem. Também é inserido o valor da chave primária, que é incremental. Nesse caso, a chave primária é o Id_GrupoGeo.

o Número de registros inseridos: 5151 (cinco mil cento e cinquenta e um).

● **Job:**
o **O que faz:** Configura um job (1) para rodar e executar todas as transformações (2 - 21) automaticamente.

o Passos:
▪ **(1) Start:** Configura o job. Todas as configurações do job se encontram nesse passo, como agendamentos e repetições.
▪ **(2) Transformation Importa Dados 2005 para DS:** Executa a transformação de importação dos dados para o DS.
▪ **(3) Transformation D_Data DS:** Executa a transformação da dimensão Tempo para o DS.
▪ **(4) Transformation D_Data DW:** Executa a transformação da dimensão Tempo para o DW.
▪ **(5) Transformation D_Cliente DS:** Executa a transformação da dimensão Cliente para o DS.
▪ **(6) Transformation D_Cliente DW:** Executa a transformação da dimensão Cliente para o DW.
▪ **(7) Transformation GrupoGeografico DS:** Executa a transformação da tabela de Grupo Geográfico da dimensão Geografia para o DS.

- **(8) Transformation GrupoGeografico DW:** Executa a transformação da tabela de Grupo Geográfico da dimensão Geografia para o DW.
- **(9) Transformation Geografia - PAIS DS:** Executa a transformação da tabela de País da dimensão Geografia para o DS.
- **(10) Transformation Geografia - PAIS DW:** Executa a transformação da tabela de País da dimensão Geografia para o DW.
- **(12) Transformation Geografia - REGIÃO VENDAS DS:** Executa a transformação da tabela de Região Vendas da dimensão Geografia para o DS.
- **(13) Transformation Geografia - REGIÃO VENDAS DS:** Executa a transformação da tabela de Região Vendas da dimensão Geografia para o DW.
- **(13) Transformation D_Funcionário DS:** Executa a transformação da dimensão Funcionário para o DS.
- **(14) Transformation D_Funcionário DW:** Executa a transformação da dimensão Funcionário para o DW.
- **(15) Transformation Funcionário DW UPDATE ID_CHEFE:** Executa a transformação de atualização do atributo Id_Chefe da dimensão Funcionário do DW.
- **(16) Transformation Produto DS:** Executa a transformação da dimensão Produto para o DS.
- **(17) Transformation Produto DW:** Executa a transformação da dimensão Produto para o DW.
- **(18) Transformation Fatos Vendas DS:** Executa a transformação do fato Vendas para o DS.
- **(19) Transformation Fatos Vendas DW:** Executa a transformação da do fato Vendas para o DW.
- **(20) Transformation Fatos Vendas DS:** Executa a transformação da do fato Vendas Detalhes para o DS.
- **(21) Transformation Fatos Vendas DW:** Executa a transformação do fato Vendas Detalhes para o DW.

Construção do Data Warehouse

A primeira transformação envolve transferir os dados de um arquivo CSV contendo a massa de dados relevante para uma tabela denominada TbImp_Vendas presente no Data Stage, de onde serão selecionados todos os fatos e dimensões relevantes para as demais tabelas do DS e, posteriormente, do Data Warehouse, onde serão gerados os identificadores únicos dos registros.

Cada transformação executada nos passos seguintes pertence a uma dimensão ou fato e faz as inserções no banco do DS ou do DW. Sendo assim, cada transformação executada no DS valida e trata os dados antes de inseri-los na tabela. As transformações do DW, por sua vez, leem os dados das tabelas do DS, selecionam os valores necessários e inserem diretamente na respectiva tabela do DW. Essa inserção pode ser feita diretamente, pois os dados já foram tratados e validados ao serem inseridos no DS.

É interessante ressaltar que a Transformação 14, que insere o Id_Chefe dos funcionários na tabela, se difere das demais pois a dimensão Funcionário possui um relacionamento com si própria (o chefe de um funcionário também é um funcionário). Sendo assim, os dados dos funcionários devem ser inseridos no DW primeiramente (para ser gerado o id de cada registro) e somente depois ser buscado o id do respectivo chefe de cada funcionário.

Por fim, temos também o *Job* (passo seguinte da Transformação 20), que nada mais é que a realização em sequência e automatizada de todas as transformações até o momento, ou seja, desde a importação da massa de dados até a Transformação 20 – Fato Venda Detalhe para o DW.

Dificuldades e Dúvidas

No início houve dificuldade em entender a necessidade do Data Stage, e o carregamento de todos os dados ser feito em uma só tabela (TbImp_Vendas) deixava isso ainda mais confuso. Mas com um estudo mais aprofundado das transformações e a finalidade de cada uma delas, conseguimos entender que o Data Stage é essencial para garantir que todos os dados cheguem validados no Data Warehouse. Isso é importante pois, uma vez que um registro foi inserido no DW, ele é imutável e não pode ser removido.

Conclusão

Foi possível ter uma ideia melhor do funcionamento de uma estrutura de um Data Warehouse e sua ligação com um Data Stage. A validação na etapa do DS garante consistência para os dados que serão passados para o DW, e a partir disso, é possível agrupar os dados de acordo com as suas áreas, gerar valores e informações seguras e analisá-las de maneira mais eficiente.

O Data Warehouse serve como uma ferramenta fundamental para contribuir com o recolhimento de informações em bancos de dados. Assim, podemos concluir que o Data Warehouse é a peça mais importante de uma infraestrutura de Business Intelligence.