

Scripts de criação das dimensões:

```
-- Criação das dimensões

CREATE TABLE dim_cliente (
  chave_cliente NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_cliente NUMBER(10) NOT NULL,
  nom_cliente VARCHAR2(50),
  des_racao_social VARCHAR2(50),
  tid_pessoa CHAR(1),
  num_cpf_cnpj NUMBER(15),
  dat_cadastro DATE,
  dat_cancelamento DATE,
  sta_ativo CHAR(1)
);

CREATE TABLE dim_produto (
  chave_produto NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_produto NUMBER(10) NOT NULL,
  nom_produto VARCHAR2(20),
  cod_barra VARCHAR2(20),
  sta_ativo VARCHAR2(20),
  dat_cadastro DATE,
  dat_cancelamento DATE
);

CREATE TABLE dim_vendedor (
  chave_vendedor NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_vendedor NUMBER(4) NOT NULL,
  nom_vendedor VARCHAR2(50),
  sta_ativo CHAR(1)
);

CREATE TABLE dim_estoque (
  chave_estoque NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_estoque NUMBER(4) NOT NULL,
  nom_estoque VARCHAR2(50)
);

CREATE TABLE dim_historico_pedido (
  chave_historico NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_pedido NUMBER(10) NOT NULL,
  cod_cliente NUMBER(10),
  dat_pedido DATE,
  dat_cancelamento DATE,
  dat_entrega DATE,
  val_total_pedido NUMBER(12,2),
  val_desconto NUMBER(12,2),
  cod_vendedor NUMBER(4)
);

CREATE TABLE dim_endereco_cliente (
  chave_endereco NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_cliente NUMBER(10),
  des_endereco VARCHAR2(50),
  num_cep NUMBER(9),
  des_bairro VARCHAR2(50),
  cod_cidade NUMBER(6)
);

CREATE TABLE dim_cidade (
  chave_cidade NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_cidade NUMBER(6) NOT NULL,
  nom_cidade VARCHAR2(50),
  cod_estado CHAR(3)
);

CREATE TABLE dim_estado (
  chave_estado NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_estado CHAR(3) NOT NULL,
  nom_estado VARCHAR2(50),
  cod_pais NUMBER(3)
);

CREATE TABLE dim_pais (
  chave_pais NUMBER
    GENERATED BY DEFAULT AS IDENTITY
  PRIMARY KEY,
  cod_pais NUMBER(3) NOT NULL,
  nom_pais VARCHAR2(50)
);
```

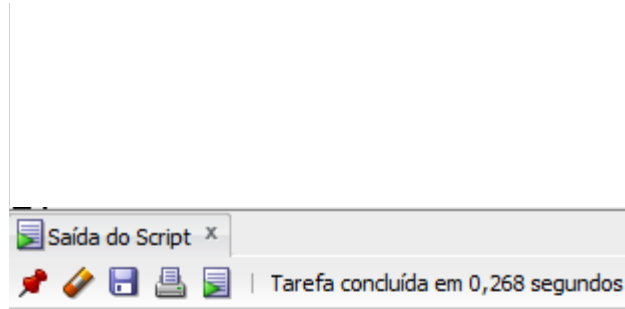


Table DIM_CLIENTE criado.

Table DIM_PRODUTO criado.

Table DIM_VENDEDOR criado.

Table DIM_ESTOQUE criado.

Table DIM_HISTORICO_PEDIDO criado.

Table DIM_ENDERECO_CLIENTE criado.

Table DIM_CIDADE criado.

Table DIM_ESTADO criado.

Table DIM_PAIS criado.

Criação da tabela fato:

```
-- Criação do fato

CREATE TABLE fato_pedido (
  chave_pedido    NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  chave_cliente   NUMBER NOT NULL,
  chave_produto   NUMBER NOT NULL,
  chave_vendedor  NUMBER NOT NULL,
  chave_estoque   NUMBER,
  chave_endereco  NUMBER,
  chave_cidade    NUMBER,
  chave_estado    NUMBER,
  chave_pais       NUMBER,
  chave_historico NUMBER,
  qtd_item        NUMBER(10, 2) NOT NULL,
  valor_uni       NUMBER(10, 2) NOT NULL,
  valor_total     NUMBER(12, 2) NOT NULL,
  data_pedido     DATE NOT NULL,
  sta_pedido      CHAR(1),
  CONSTRAINT fk_pedido_cliente FOREIGN KEY (chave_cliente)
    REFERENCES dim_cliente (chave_cliente),
  CONSTRAINT fk_pedido_produto FOREIGN KEY (chave_produto)
    REFERENCES dim_produto (chave_produto),
  CONSTRAINT fk_pedido_vendedor FOREIGN KEY (chave_vendedor)
    REFERENCES dim_vendedor (chave_vendedor),
  CONSTRAINT fk_pedido_estoque FOREIGN KEY (chave_estoque)
    REFERENCES dim_estoque (chave_estoque),
  CONSTRAINT fk_pedido_endereco FOREIGN KEY (chave_endereco)
    REFERENCES dim_endereco_cliente (chave_endereco),
  CONSTRAINT fk_pedido_cidade FOREIGN KEY (chave_cidade)
    REFERENCES dim_cidade (chave_cidade),
  CONSTRAINT fk_pedido_estado FOREIGN KEY (chave_estado)
    REFERENCES dim_estado (chave_estado),
  CONSTRAINT fk_pedido_pais FOREIGN KEY (chave_pais)
    REFERENCES dim_pais (chave_pais),
  CONSTRAINT fk_pedido_historico FOREIGN KEY (chave_historico)
    REFERENCES dim_historico_pedido (chave_historico)
);
```



Table FATO_PEDIDO criado.

Criação da tabela de auditoria:

```
144 CREATE TABLE auditoria_dimensoes (  
145     id_auditoria    NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
146     tabela_nome     VARCHAR2(50),  
147     operacao        VARCHAR2(10),  
148     usuario_banco   VARCHAR2(30),  
149     data_operacao    DATE,  
150     chave_registro   VARCHAR2(50)  
151 );
```

Saída do Script x

Tarefa concluída em 0,053 segundos

Table AUDITORIA_DIMENSOES criado.

```
153 CREATE TABLE aud_fato_pedido (  
154     id_auditoria    NUMBER  
155     GENERATED BY DEFAULT AS IDENTITY  
156     PRIMARY KEY,  
157     operacao        VARCHAR2(10),  
158     usuario_bd       VARCHAR2(50),  
159     data_operacao    DATE,  
160     chave_pedido     NUMBER,  
161     chave_cliente    NUMBER,  
162     chave_produto    NUMBER,  
163     chave_vendedor   NUMBER,  
164     chave_estoque    NUMBER,  
165     chave_endereco   NUMBER,  
166     chave_cidade     NUMBER,  
167     chave_estado     NUMBER,  
168     chave_pais       NUMBER,  
169     chave_historico  NUMBER,  
170     qtd_item         NUMBER(10, 2),  
171     valor_uni        NUMBER(10, 2),  
172     valor_total      NUMBER(12, 2),  
173     data_pedido     DATE,  
174     sta_pedido       CHAR(1)  
175 );
```

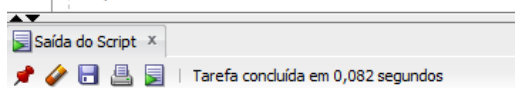
Saída do Script x

Tarefa concluída em 0,045 segundos

Table AUD FATO PEDIDO criado.

Triggers de auditoria:

```
280 CREATE OR REPLACE TRIGGER trg_aud_fato_pedido
281 AFTER INSERT ON fato_pedido
282 FOR EACH ROW
283 BEGIN
284     INSERT INTO aud_fato_pedido (
285         operacao,
286         usuario_bd,
287         data_operacao,
288         chave_pedido,
289         chave_cliente,
290         chave_produto,
291         chave_vendedor,
292         chave_estoque,
293         chave_endereco,
294         chave_cidade,
295         chave_estado,
296         chave_pais,
297         chave_historico,
298         qtd_item,
299         valor_uni,
300         valor_total,
301         data_pedido,
302         sta_pedido
303     ) VALUES (
304         'INSERT',
305         USER,
306         SYSDATE,
307         :NEW.chave_pedido,
308         :NEW.chave_cliente,
309         :NEW.chave_produto,
310         :NEW.chave_vendedor,
311         :NEW.chave_estoque,
312         :NEW.chave_endereco,
313         :NEW.chave_cidade,
314         :NEW.chave_estado,
315         :NEW.chave_pais,
316         :NEW.chave_historico,
317         :NEW.qtd_item,
318         :NEW.valor_uni,
319         :NEW.valor_total,
320         :NEW.data_pedido,
321         :NEW.sta_pedido
322     );
323 END;
```



Trigger TRG_AUD_FATO_PEDIDO compilado

```

CREATE OR REPLACE TRIGGER trg_audit_dim_cliente
AFTER INSERT ON dim_cliente
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_CLIENTE', 'INSERT', USER, SYSDATE, :NEW.chave_cliente);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_produto
AFTER INSERT ON dim_produto
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_PRODUTO', 'INSERT', USER, SYSDATE, :NEW.chave_produto);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_vendedor
AFTER INSERT ON dim_vendedor
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_VENDEDOR', 'INSERT', USER, SYSDATE, :NEW.chave_vendedor);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_estoque
AFTER INSERT ON dim_estoque
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_ESTOQUE', 'INSERT', USER, SYSDATE, :NEW.chave_estoque);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_endereco_cliente
AFTER INSERT ON dim_endereco_cliente
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_ENDERECO_CLIENTE', 'INSERT', USER, SYSDATE, :NEW.chave_endereco);
END;
/

```

```

CREATE OR REPLACE TRIGGER trg_audit_dim_cidade
AFTER INSERT ON dim_cidade
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_CIDADE', 'INSERT', USER, SYSDATE, :NEW.chave_cidade);
END;
/






CREATE OR REPLACE TRIGGER trg_audit_dim_estado
AFTER INSERT ON dim_estado
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_ESTADO', 'INSERT', USER, SYSDATE, :NEW.chave_estado);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_pais
AFTER INSERT ON dim_pais
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_PAIS', 'INSERT', USER, SYSDATE, :NEW.chave_pais);
END;
/

CREATE OR REPLACE TRIGGER trg_audit_dim_historico_pedido
AFTER INSERT ON dim_historico_pedido
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_dimensoes
        (tabela_nome, operacao, usuario_banco, data_operacao, chave_registro)
    VALUES
        ('DIM_HISTORICO_PEDIDO', 'INSERT', USER, SYSDATE, :NEW.chave_historico);
END;
/

```

Saída do Script x

 | Tarefa concluída em 0,295 segundos

```
Trigger TRG_AUDIT_DIM_CLIENTE compilado

Trigger TRG_AUDIT_DIM_PRODUTO compilado

Trigger TRG_AUDIT_DIM_VENDEDOR compilado

Trigger TRG_AUDIT_DIM_ESTOQUE compilado

Trigger TRG_AUDIT_DIM_ENDERECO_CLIENTE compilado

Trigger TRG_AUDIT_DIM_CIDADE compilado

Trigger TRG_AUDIT_DIM_ESTADO compilado

Trigger TRG_AUDIT_DIM_PAIS compilado

Trigger TRG_AUDIT_DIM_HISTORICO_PEDIDO compilado
```

Triggers para tratamento de dados:

```
CREATE OR REPLACE TRIGGER trg_dim_cliente_bi
BEFORE INSERT ON dim_cliente
FOR EACH ROW
BEGIN
    IF :NEW.cod_cliente IS NULL THEN
        RAISE_APPLICATION_ERROR(-20010, 'cod_cliente nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_cliente IS NULL THEN
        :NEW.nom_cliente := 'NAO INFORMADO';
    END IF;
    IF :NEW.sta_ativo IS NULL THEN
        :NEW.sta_ativo := 'A';
    END IF;
END;
/

-- Produto
CREATE OR REPLACE TRIGGER trg_dim_produto_bi
BEFORE INSERT ON dim_produto
FOR EACH ROW
BEGIN
    IF :NEW.cod_produto IS NULL THEN
        RAISE_APPLICATION_ERROR(-20011, 'cod_produto nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_produto IS NULL THEN
        :NEW.nom_produto := 'NAO INFORMADO';
    END IF;
    IF :NEW.sta_ativo IS NULL THEN
        :NEW.sta_ativo := 'A';
    END IF;
END;
/

-- Vendedor
CREATE OR REPLACE TRIGGER trg_dim_vendedor_bi
BEFORE INSERT ON dim_vendedor
FOR EACH ROW
BEGIN
    IF :NEW.cod_vendedor IS NULL THEN
        RAISE_APPLICATION_ERROR(-20012, 'cod_vendedor nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_vendedor IS NULL THEN
        :NEW.nom_vendedor := 'NAO INFORMADO';
    END IF;
    IF :NEW.sta_ativo IS NULL THEN
        :NEW.sta_ativo := 'A';
    END IF;
END;
/

-- Estoque
CREATE OR REPLACE TRIGGER trg_dim_estoque_bi
BEFORE INSERT ON dim_estoque
FOR EACH ROW
BEGIN
    IF :NEW.cod_estoque IS NULL THEN
        RAISE_APPLICATION_ERROR(-20013, 'cod_estoque nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_estoque IS NULL THEN
        :NEW.nom_estoque := 'NAO INFORMADO';
    END IF;
END;
/
```

```
-- Endereco Cliente
CREATE OR REPLACE TRIGGER trg_dim_endereco_cliente_bi
BEFORE INSERT ON dim_endereco_cliente
FOR EACH ROW
BEGIN
    IF :NEW.cod_cliente IS NULL THEN
        RAISE_APPLICATION_ERROR(-20014, 'cod_cliente do endereco nao pode ser nulo.');
```

```
END IF;
    IF :NEW.des_endereco IS NULL THEN
        :NEW.des_endereco := 'NAO INFORMADO';
    END IF;
    IF :NEW.num_cep IS NULL THEN
        :NEW.num_cep := 0;
    END IF;
    IF :NEW.des_bairro IS NULL THEN
        :NEW.des_bairro := 'NAO INFORMADO';
    END IF;
END;
/

-- Cidade
CREATE OR REPLACE TRIGGER trg_dim_cidade_bi
BEFORE INSERT ON dim_cidade
FOR EACH ROW
BEGIN
    IF :NEW.cod_cidade IS NULL THEN
        RAISE_APPLICATION_ERROR(-20015, 'cod_cidade nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_cidade IS NULL THEN
        :NEW.nom_cidade := 'NAO INFORMADO';
    END IF;
    IF :NEW.cod_estado IS NULL THEN
        :NEW.cod_estado := 'XX';
    END IF;
END;
/

-- Estado
CREATE OR REPLACE TRIGGER trg_dim_estado_bi
BEFORE INSERT ON dim_estado
FOR EACH ROW
BEGIN
    IF :NEW.cod_estado IS NULL THEN
        RAISE_APPLICATION_ERROR(-20016, 'cod_estado nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_estado IS NULL THEN
        :NEW.nom_estado := 'NAO INFORMADO';
    END IF;
    IF :NEW.cod_pais IS NULL THEN
        :NEW.cod_pais := 0;
    END IF;
END;
/

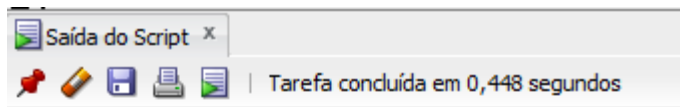
-- Pais
CREATE OR REPLACE TRIGGER trg_dim_pais_bi
BEFORE INSERT ON dim_pais
FOR EACH ROW
BEGIN
    IF :NEW.cod_pais IS NULL THEN
        RAISE_APPLICATION_ERROR(-20017, 'cod_pais nao pode ser nulo.');
```

```
END IF;
    IF :NEW.nom_pais IS NULL THEN
        :NEW.nom_pais := 'NAO INFORMADO';
    END IF;
END;
/

-- Historico Pedido
CREATE OR REPLACE TRIGGER trg_dim_historico_pedido_bi
BEFORE INSERT ON dim_historico_pedido
FOR EACH ROW
BEGIN
    IF :NEW.cod_pedido IS NULL THEN
        RAISE_APPLICATION_ERROR(-20018, 'cod_pedido nao pode ser nulo.');
```

```
END IF;
    IF :NEW.cod_cliente IS NULL THEN
        RAISE_APPLICATION_ERROR(-20019, 'cod_cliente no historico nao pode ser nulo.');
```

```
END IF;
    IF :NEW.dat_pedido IS NULL THEN
        :NEW.dat_pedido := SYSDATE;
    END IF;
    IF :NEW.val_total_pedido IS NULL THEN
        :NEW.val_total_pedido := 0;
    END IF;
    IF :NEW.val_desconto IS NULL THEN
        :NEW.val_desconto := 0;
    END IF;
END;
/
```

Trigger TRG_DIM_CLIENTE_BI compilado

Trigger TRG_DIM_PRODUTO_BI compilado

Trigger TRG_DIM_VENDEDOR_BI compilado

Trigger TRG_DIM_ESTOQUE_BI compilado

Trigger TRG_DIM_ENDERECO_CLIENTE_BI compilado

Trigger TRG_DIM_CIDADE_BI compilado

Trigger TRG_DIM_ESTADO_BI compilado

Trigger TRG_DIM_PAIS_BI compilado

Trigger TRG_DIM_HISTORICO_PEDIDO_BI compilado

Carregamentos de dados:

```
-- Carregamento de dados

INSERT INTO dim_cliente (
    cod_cliente,
    nom_cliente,
    des_razao_social,
    tip_pessoa,
    num_cpf_cnpj,
    dat_cadastro,
    dat_cancelamento,
    sta_ativo
)
SELECT
    cod_cliente,
    nom_cliente,
    des_razao_social,
    tip_pessoa,
    num_cpf_cnpj,
    dat_cadastro,
    dat_cancelamento,
    sta_ativo
FROM
    cliente;

INSERT INTO dim_produto (
    cod_produto,
    nom_produto,
    cod_barra,
    sta_ativo,
    dat_cadastro,
    dat_cancelamento
)
SELECT
    cod_produto,
    nom_produto,
    cod_barra,
    sta_ativo,
    dat_cadastro,
    dat_cancelamento
FROM
    produto;

INSERT INTO dim_vendedor (
    cod_vendedor,
    nom_vendedor,
    sta_ativo
)
SELECT
    cod_vendedor,
    nom_vendedor,
    sta_ativo
FROM
    vendedor;

INSERT INTO dim_estoque (
    cod_estoque,
    nom_estoque
)
SELECT
    cod_estoque,
    nom_estoque
FROM
    estoque;

INSERT INTO dim_historico_pedido (
    cod_pedido,
    cod_cliente,
    dat_pedido,
    dat_cancelamento,
    dat_entrega,
    val_total_pedido,
    val_desconto,
    cod_vendedor
)
SELECT
    cod_pedido,
    cod_cliente,
    dat_pedido,
    dat_cancelamento,
    dat_entrega,
    val_total_pedido,
    val_desconto,
    cod_vendedor
FROM historico_pedido;

INSERT INTO dim_endereco_cliente (
    cod_cliente,
    des_endereco,
    num_cep,
    des_bairro,
    cod_cidade
)
SELECT
    cod_cliente,
    des_endereco,
    num_cep,
    des_bairro,
    cod_cidade
FROM
    endereco_cliente;
```

```

3) INSERT INTO dim_cidade (
    cod_cidade,
    nom_cidade,
    cod_estado
)
SELECT
    cod_cidade,
    nom_cidade,
    cod_estado
FROM
    cidade;

4) INSERT INTO dim_estado (
    cod_estado,
    nom_estado,
    cod_pais
)
SELECT
    cod_estado,
    nom_estado,
    cod_pais
FROM
    estado;

5) INSERT INTO dim_pais (
    cod_pais,
    nom_pais
)
SELECT
    cod_pais,
    nom_pais
FROM
    pais;

```

Saída do Script x

146 linhas inserido.

49 linhas inserido.

30 linhas inserido.

10 linhas inserido.

40.500 linhas inserido.

46 linhas inserido.

269 linhas inserido.

47 linhas inserido.

19 linhas inserido.

Packages de dimensões:

```
CREATE OR REPLACE PACKAGE pkg_dimensoes AS
  PROCEDURE inserir_cliente(
    p_cod_cliente NUMBER,
    p_nom_cliente VARCHAR2,
    p_des_razao_social VARCHAR2,
    p_tip_pessoa CHAR,
    p_num_cpf_cnpj NUMBER,
    p_dat_cadastro DATE,
    p_dat_cancelamento DATE,
    p_sta_ativo CHAR
  );

  PROCEDURE inserir_produto(
    p_cod_produto NUMBER,
    p_nom_produto VARCHAR2,
    p_cod_barra VARCHAR2,
    p_sta_ativo VARCHAR2,
    p_dat_cadastro DATE,
    p_dat_cancelamento DATE
  );

  PROCEDURE inserir_vendedor(
    p_cod_vendedor NUMBER,
    p_nom_vendedor VARCHAR2,
    p_sta_ativo CHAR
  );

  PROCEDURE inserir_estoque(
    p_cod_estoque NUMBER,
    p_nom_estoque VARCHAR2
  );

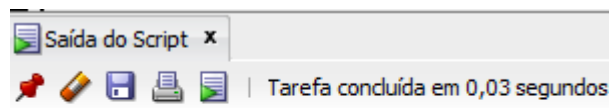
  PROCEDURE inserir_endereco_cliente(
    p_cod_cliente NUMBER,
    p_des_endereco VARCHAR2,
    p_num_cep NUMBER,
    p_des_bairro VARCHAR2,
    p_cod_cidade NUMBER
  );

  PROCEDURE inserir_cidade(
    p_cod_cidade NUMBER,
    p_nom_cidade VARCHAR2,
    p_cod_estado CHAR
  );

  PROCEDURE inserir_estado(
    p_cod_estado CHAR,
    p_nom_estado VARCHAR2,
    p_cod_pais NUMBER
  );

  PROCEDURE inserir_pais(
    p_cod_pais NUMBER,
    p_nom_pais VARCHAR2
  );

  PROCEDURE inserir_historico_pedido(
    p_cod_pedido NUMBER,
    p_cod_cliente NUMBER,
    p_dat_pedido DATE,
    p_dat_cancelamento DATE,
    p_dat_entrega DATE,
    p_val_total_pedido NUMBER,
    p_val_desconto NUMBER,
    p_cod_vendedor NUMBER
  );
END pkg_dimensoes;
/
```



Package PKG_DIMENSOES compilado

Inserções das dimensões:

```
CREATE OR REPLACE PACKAGE BODY pkg_dimensoes AS
-- Inserir Cliente
PROCEDURE inserir_cliente(
    p_cod_cliente NUMBER,
    p_nom_cliente VARCHAR2,
    p_des_razao_social VARCHAR2,
    p_tip_pessoa CHAR,
    p_num_cpf_cnpj NUMBER,
    p_dat_cadastro DATE,
    p_dat_cancelamento DATE,
    p_sta_ativo CHAR
) IS
BEGIN
    IF p_cod_cliente IS NULL OR p_nom_cliente IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001,'Campos obrigatórios do cliente ausentes!');
    END IF;

    INSERT INTO dim_cliente(
        cod_cliente, nom_cliente, des_razao_social, tip_pessoa,
        num_cpf_cnpj, dat_cadastro, dat_cancelamento, sta_ativo
    )
    VALUES (
        p_cod_cliente, p_nom_cliente, p_des_razao_social, p_tip_pessoa,
        p_num_cpf_cnpj, p_dat_cadastro, p_dat_cancelamento, p_sta_ativo
    );
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_cliente: '||SQLERRM);
END;

-- Inserir Produto
PROCEDURE inserir_produto(
    p_cod_produto NUMBER,
    p_nom_produto VARCHAR2,
    p_cod_barra VARCHAR2,
    p_sta_ativo VARCHAR2,
    p_dat_cadastro DATE,
    p_dat_cancelamento DATE
) IS
BEGIN
    IF p_cod_produto IS NULL OR p_nom_produto IS NULL THEN
        RAISE_APPLICATION_ERROR(-20002,'Campos obrigatórios do produto ausentes!');
    END IF;

    INSERT INTO dim_produto(
        cod_produto, nom_produto, cod_barra, sta_ativo, dat_cadastro, dat_cancelamento
    )
    VALUES (
        p_cod_produto, p_nom_produto, p_cod_barra, p_sta_ativo, p_dat_cadastro, p_dat_cancelamento
    );
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_produto: '||SQLERRM);
END;
```

```

PROCEDURE inserir_vendedor(
    p_cod_vendedor NUMBER,
    p_nom_vendedor VARCHAR2,
    p_sta_ativo CHAR
) IS
BEGIN
    IF p_cod_vendedor IS NULL OR p_nom_vendedor IS NULL THEN
        RAISE_APPLICATION_ERROR(-20003,'Campos obrigatórios do vendedor ausentes!');
    END IF;

    INSERT INTO dim_vendedor(
        cod_vendedor, nom_vendedor, sta_ativo
    )
    VALUES (
        p_cod_vendedor, p_nom_vendedor, p_sta_ativo
    );
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_vendedor: '||SQLERRM);
END;

-- Inserir Estoque
PROCEDURE inserir_estoque(
    p_cod_estoque NUMBER,
    p_nom_estoque VARCHAR2
) IS
BEGIN
    IF p_cod_estoque IS NULL OR p_nom_estoque IS NULL THEN
        RAISE_APPLICATION_ERROR(-20004,'Campos obrigatórios do estoque ausentes!');
    END IF;

    INSERT INTO dim_estoque(cod_estoque, nom_estoque)
    VALUES (p_cod_estoque, p_nom_estoque);
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_estoque: '||SQLERRM);
END;

-- Inserir Endereco Cliente
PROCEDURE inserir_endereco_cliente(
    p_cod_cliente NUMBER,
    p_des_endereco VARCHAR2,
    p_num_cep NUMBER,
    p_des_bairro VARCHAR2,
    p_cod_cidade NUMBER
) IS
BEGIN
    IF p_cod_cliente IS NULL OR p_des_endereco IS NULL THEN
        RAISE_APPLICATION_ERROR(-20005,'Campos obrigatórios do endereco ausentes!');
    END IF;

    INSERT INTO dim_endereco_cliente(
        cod_cliente, des_endereco, num_cep, des_bairro, cod_cidade
    )
    VALUES (
        p_cod_cliente, p_des_endereco, p_num_cep, p_des_bairro, p_cod_cidade
    );
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_endereco_cliente: '||SQLERRM);
END;

```

```

PROCEDURE inserir_cidade(
    p_cod_cidade NUMBER,
    p_nom_cidade VARCHAR2,
    p_cod_estado CHAR
) IS
BEGIN
    IF p_cod_cidade IS NULL OR p_nom_cidade IS NULL THEN
        RAISE_APPLICATION_ERROR(-20006,'Campos obrigatórios da cidade ausentes!');
    END IF;

    INSERT INTO dim_cidade(cod_cidade, nom_cidade, cod_estado)
    VALUES (p_cod_cidade, p_nom_cidade, p_cod_estado);
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_cidade: '||SQLERRM);
END;

-- Inserir Estado
PROCEDURE inserir_estado(
    p_cod_estado CHAR,
    p_nom_estado VARCHAR2,
    p_cod_pais NUMBER
) IS
BEGIN
    IF p_cod_estado IS NULL OR p_nom_estado IS NULL THEN
        RAISE_APPLICATION_ERROR(-20007,'Campos obrigatórios do estado ausentes!');
    END IF;

    INSERT INTO dim_estado(cod_estado, nom_estado, cod_pais)
    VALUES (p_cod_estado, p_nom_estado, p_cod_pais);
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_estado: '||SQLERRM);
END;

-- Inserir Pais
PROCEDURE inserir_pais(
    p_cod_pais NUMBER,
    p_nom_pais VARCHAR2
) IS
BEGIN
    IF p_cod_pais IS NULL OR p_nom_pais IS NULL THEN
        RAISE_APPLICATION_ERROR(-20008,'Campos obrigatórios do pais ausentes!');
    END IF;

    INSERT INTO dim_pais(cod_pais, nom_pais)
    VALUES (p_cod_pais, p_nom_pais);
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_pais: '||SQLERRM);
END;

```

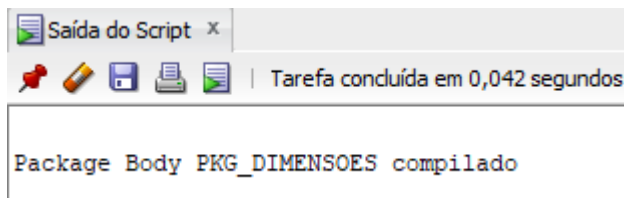
```

PROCEDURE inserir_historico_pedido(
    p_cod_pedido NUMBER,
    p_cod_cliente NUMBER,
    p_dat_pedido DATE,
    p_dat_cancelamento DATE,
    p_dat_entrega DATE,
    p_val_total_pedido NUMBER,
    p_val_desconto NUMBER,
    p_cod_vendedor NUMBER
) IS
BEGIN
    IF p_cod_pedido IS NULL OR p_cod_cliente IS NULL OR p_dat_pedido IS NULL THEN
        RAISE_APPLICATION_ERROR(-20009, 'Campos obrigatórios do historico de pedido ausentes!');
    END IF;

    INSERT INTO dim_historico_pedido(
        cod_pedido, cod_cliente, dat_pedido, dat_cancelamento, dat_entrega,
        val_total_pedido, val_desconto, cod_vendedor
    ) VALUES (
        p_cod_pedido, p_cod_cliente, p_dat_pedido, p_dat_cancelamento, p_dat_entrega,
        p_val_total_pedido, p_val_desconto, p_cod_vendedor
    );
    COMMIT;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Erro inserir_historico_pedido: '||SQLERRM);
END;

END pkg_dimensoes;
/

```



Package de fato:

```
CREATE OR REPLACE PACKAGE BODY pkg_fato AS
```

```
    PROCEDURE inserir_fato_pedido (
        p_chave_cliente    NUMBER,
        p_chave_produto     NUMBER,
        p_chave_vendedor    NUMBER,
        p_chave_estoque     NUMBER,
        p_chave_endereco    NUMBER,
        p_chave_cidade      NUMBER,
        p_chave_estado      NUMBER,
        p_chave_pais        NUMBER,
        p_chave_historico   NUMBER,
        p_qtd_item          NUMBER,
        p_valor_uni         NUMBER,
        p_data_pedido       DATE,
        p_sta_pedido        CHAR
    ) IS
        v_valor_total NUMBER(12, 2);
    BEGIN
        IF p_chave_cliente IS NULL
        OR p_chave_produto IS NULL
        OR p_qtd_item IS NULL
        OR p_valor_uni IS NULL THEN
            raise_application_error(-20001, 'Campos obrigatórios ausentes!');
        END IF;

        -- Calcula o valor total automaticamente
        v_valor_total := p_qtd_item * p_valor_uni;

        INSERT INTO fato_pedido (
            chave_cliente,
            chave_produto,
            chave_vendedor,
            chave_estoque,
            chave_endereco,
            chave_cidade,
            chave_estado,
            chave_pais,
            chave_historico,
            qtd_item,
            valor_uni,
            valor_total,
            data_pedido,
            sta_pedido
        ) VALUES ( p_chave_cliente,
                    p_chave_produto,
                    p_chave_vendedor,
                    p_chave_estoque,
                    p_chave_endereco,
                    p_chave_cidade,
                    p_chave_estado,
                    p_chave_pais,
                    p_chave_historico,
                    p_qtd_item,
                    p_valor_uni,
                    v_valor_total,
                    p_data_pedido,
                    p_sta_pedido );

        COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            dbms_output.put_line('Erro inserir_fato_pedido: ' || sqlerrm);
    END;
END pkg_fato;
/
```

```
CREATE OR REPLACE PACKAGE pkg_fato AS
```

```
    PROCEDURE inserir_fato_pedido (
```

```
        p_chave_cliente    NUMBER,
        p_chave_produto     NUMBER,
        p_chave_vendedor    NUMBER,
        p_chave_estoque     NUMBER,
        p_chave_endereco    NUMBER,
        p_chave_cidade      NUMBER,
        p_chave_estado      NUMBER,
        p_chave_pais        NUMBER,
        p_chave_historico   NUMBER,
        p_qtd_item          NUMBER,
        p_valor_uni         NUMBER,
        p_data_pedido       DATE,
        p_sta_pedido        CHAR
```

```
    );
```

```
END pkg_fato;
```

```
/
```

Inserindo dados:

```
BEGIN
-- Inserir País
pkg_dimensoes.inserir_pais(1, 'Brasil');

-- Inserir Estado
pkg_dimensoes.inserir_estado('SP', 'Sao Paulo', 1);

-- Inserir Cidade
pkg_dimensoes.inserir_cidade(1001, 'Sao Paulo', 'SP');

-- Inserir Cliente
pkg_dimensoes.inserir_cliente(
    101, 'Joao Silva', 'Joao Silva LTDA', 'J', 12345678901,
    SYSDATE, NULL, 'A'
);

-- Inserir Endereço do Cliente
pkg_dimensoes.inserir_endereco_cliente(
    101, 'Rua das Flores, 123', 12345678, 'Centro', 1001
);

-- Inserir Produto
pkg_dimensoes.inserir_produto(
    201, 'Notebook', '1234567890123', 'A', SYSDATE, NULL
);

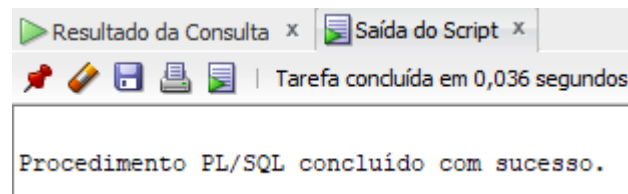
-- Inserir Vendedor
pkg_dimensoes.inserir_vendedor(301, 'Carlos Souza', 'A');

-- Inserir Estoque
pkg_dimensoes.inserir_estoque(401, 'Estoque Central');

-- Inserir Histórico de Pedido
pkg_dimensoes.inserir_historico_pedido(
    501, 101, SYSDATE, NULL, SYSDATE + 3, 7000, 0, 301
);

-- Inserir Fato de Pedido
pkg_fato.inserir_fato_pedido(
    1, -- chave_cliente
    1, -- chave_produto
    1, -- chave_vendedor
    1, -- chave_estoque
    1, -- chave_endereco
    1, -- chave_cidade
    1, -- chave_estado
    1, -- chave_pais
    1, -- chave_historico
    2, -- qtd_item
    3500, -- valor_uni
    SYSDATE, -- data_pedido
    'S' -- sta_pedido
);

COMMIT;
END;
```



8701 | select * from fato_pedido;

CHAVE_PEDIDO	CHAVE_CLIENTE	CHAVE_PRODUTO	CHAVE_VENDEDOR	CHAVE_ESTOQUE	CHAVE_ENDERECO	CHAVE_CIDADE	CHAVE_ESTADO	CHAVE_PAIS	CHAVE_HISTORICO	QTD_ITEM	VALOR_UNI	VALOR_TOTAL	DATA_PEDIDO	STA_PEDIDO
1	4	1	1	1	1	1	1	1	1	2	3500	7000	29/08/25	S

Tabelas de auditorias:

1056 | `select * from aud_fato_pedido;`

Saída do Script x Resultado da Consulta x

Todas as Linhas Extraídas: 1 em 0,018 segundos

ID_AUDITORIA	OPERACAO	USUARIO_BD	DATA_OPERACAO	CHAVE_PEDIDO	CHAVE_CLIENTE	CHAVE_PRODUTO	CHAVE_VENDEDOR	CHAVE_ESTOQUE	CHAVE_ENDERECO	CHAVE_CIDADE	CHAVE_ESTADO	CHAVE_PAIS	CHAVE_HISTORICO	QTD_ITEM	VALOR_LINHA	VALOR_TOTAL	DATA_PEDIDO	STA_PEDIDO
1	INSERT	RM555678	29/08/25											1	3500	7000	29/08/25	S

1056 | `select * from auditoria_dimensoes;`

Saída do Script x Resultado da Consulta x

Todas as Linhas Extraídas: 9 em 0,013 segundos

ID_AUDITORIA	TABELA_NOME	OPERACAO	USUARIO_BANCO	DATA_OPERACAO	CHAVE_REGISTRO
1	1 DIM_PAIS	INSERT	RM555678	29/08/25	27
2	2 DIM_ESTADO	INSERT	RM555678	29/08/25	54
3	3 DIM_CIDADE	INSERT	RM555678	29/08/25	277
4	4 DIM_CLIENTE	INSERT	RM555678	29/08/25	153
5	5 DIM_ENDERECO_CLIENTE	INSERT	RM555678	29/08/25	52
6	6 DIM_PRODUTO	INSERT	RM555678	29/08/25	55
7	7 DIM_VENDEDOR	INSERT	RM555678	29/08/25	36
8	8 DIM_ESTOQUE	INSERT	RM555678	29/08/25	16
9	9 DIM_HISTORICO_PEDIDO	INSERT	RM555678	29/08/25	40506

Dashboard:

