Relatório Trabalho 1 INF1341

Gabriel Augusto Silva de Aquino - 1521617
Pedro Felipe Santos Magalhães - 1611074

A “Compra e Vende” é uma empresa recém-criada que trabalha no atacado e no varejo de materiais de construção. Seus sócios possuem forte interesse em automatizar seu negócio o máximo possível. Por isso, logo na criação da empresa, contrataram dois pequenos pacotes de software relacionados com a área de vendas e de controle de funcionários. Tais pacotes foram implantados na empresa e entraram em operação.

No entanto, os donos da “Compra e Vende” perceberam dois problemas:

(i) Os produtos adquiridos possuem bases de dados desintegradas, existindo inconsistência entre dados dos produtos;

(ii) Os produtos adquiridos não suprem as necessidades da empresa. Você foi contratado para
trabalhar na empresa “Compra e Vende”, sendo o responsável por resolver esses problemas.

Essas abaixo são as tabelas originais recuperadas do Banco de Dados deles:

NotasVenda (Numero, DataEmissao, FormaPagamento, CodigoCliente*, CPFVendedor)

ItensNota (Numero, NumeroMercadoria*, Quantidade, ValorUnitario)

Mercadorias (NumeroMercadoria, Descricao, QuantidadeEstoque)

Cliente (Codigo, Nome, Telefone, Logradouro, Numero, Complemento, Cidade, Estado, NumeroContribuinte)

Funcionario (CPF, Nome, Telefone, Logradouro, Numero, Complemento, Cidade, Estado, CodigoDepartamento*)

CargosFunc (CPF*, CodigoCargo*, DataInicio, DataFim)

Departamento (CodigoDepartamento, Nome, CPF_Chefe)

Cargo (Codigo, Descrição, Salario_Base)

Após uma revisão dos requisitos da empresa e analisando as tabelas, nós realizamos algumas mudanças em algumas das tabelas e criamos algumas visões para garantir que as aplicações do cliente ainda funcionam corretamente.

Essas abaixo são as novas tabelas e modificações

notascompras(numero, dataEmissao,codigoFornecedor*, Entregue) Tabela nova

itensnotacompra(numero_nota*, numero_mercadoria*, quantidade, valorUnitario) Tabela nova

NotasVenda (Numero, DataEmissao, FormaPagamento, CodigoCliente*, CPFVendedor*)

Mercadorias_nova (NumeroMercadoria, Descricao, QuantidadeEstoque, Estoque_Max) Tabela Nova

ItensNota (Numero*, NumeroMercadoria*, Quantidade, ValorUnitario)

Cliente_Fornecedor (Codigo, Nome, Telefone, Logradouro, Numero, Complemento, Cidade, Estado, NumeroContribuinte, CNPJ) Tabela nova

Funcionario_nova (CPF, Nome, Telefone, Logradouro, Numero, Complemento, Cidade, Estado, CodigoDepartamento*, Email)Tabela nova

CargosFunc (CPF*, CodigoCargo*, DataInicio, DataFim)

Departamento (CodigoDepartamento, Nome, CPF_Chefe*)

Cargo (Codigo, Descrição, Salario_Base)

Preco_min_venda (CodigoMercadoria*, ultimopreco) Tabela nova

Essas são as visões que criamos:

Cliente (Codigo, Nome, Telefone, Logradouro, Numero, Complemento, Cidade,Estado, CodigoDepartamento*) Sem CNPJ

Funcionario (CPF, Nome, Telefone, Logradouro, Numero, Complemento, Cidade, Estado, CodigoDepartamento*) Sem Email

Mercadorias (NumeroMercadoria, Descricao, QuantidadeEstoque)

Produtos_nao_recebidos (numero, dataEmissao,codigoFornecedor*, Entregue)

1. O cliente quer que exista um controle de compras, onde será possivel cadastrar notas fiscais referentes às mercadorias adquiridas:

Para isso criamos as tabelas

- Notas_Compras
- Itens_Nota_Compra

As informações do fornecedor podem ser obtidas com um join da tabela **Notas_Compras** com a tabela **Cliente_Fornecedor**.

2. O cliente informou que um fornecedor pode ser cliente e que deve ser possivel identificar todos os produtos que ele forneceu/fornecerá para a “Compra e Vende” e todas as compras que realizou.

Para isso nós mudamos a tabela cliente para **Cliente_Fornecedor** com a adição de um **CNPJ**, e para não atrapalhar as aplicações nós também criamos a visão **Cliente** com os dados apenas os dados anteriores a mudança.

Para conseguir os itens fornecidos, podemos dar um join da tabela **Cliente_Fornecedor** com as tabelas **Itens_Nota, Notas_Compra** e **Mercadorias_nova**.

Temos também na tabela **Notas_Compra** uma coluna para “Entregue”, assim saberemos os produtos que um devido fornecedor ainda deve entregar.

3. O cliente quer que toda vez que o estoque de uma mercadoria estiver abaixo de um dado valor, deve ser enviado um e-mail para o administrador do sistema de compras, indicando que o pedido de um produto deve ser realizado.

- Nós criamos a nova tabela **Funcionarios_nova** que contém o email de todos os funcionários, assim criamos a visão **Funcionarios** com todas as colunas originais.
- Em linhas gerais seria criado um alerta na coluna de estoque e quando esse estoque ficasse baixo demais chamariamos um processo batch que enviaria um email para os funcionários responsáveis.

4. Deve ser implementada regra que impeça que o preço de venda de um produto seja menor que o preço da última compra realizada para o produto em questão.

- Criamos a tabela preco_min_mercadoria (CodigoMercadoria*, ultimopreco)
- Criamos a trigger **atualiza_preco_min** que atualiza o preço min a cada compra de mercadoria

5. Não deve ser possível realizar vendas de produtos sem estoque.

- Criamos uma trigger que impede (raise application error) update na tabela mercadorias quando o estoque é menor que 0, como o comando da venda de mercadoria deve estar em transação, a geração da nota também deverá ser bloqueada pois a transação vai dar rollback.

6. Não deve ser possível realizar aquisições que façam com que um produto ultrapasse seu estoque máximo.

- Criamos uma trigger que impede (raise application error) update na tabela mercadorias quando o estoque é maior que o máximo, a geração da nota também deverá ser bloqueada pois a transação vai dar rollback.

7. Deve ser criado um mecanismo para garantir a integridade dos dados de vendedores com a base de funcionários.

- Criamos a chave estrangeira de CPF na tabela **NotasVenda** referenciando o CPF do funcionário.

8. Deve ser criada uma forma de geração de prévia de valores a pagar aos funcionários, considerando-se o salário base dos mesmos e o valor de 5% de comissão sobre as vendas realizadas pelos vendedores. Deverão ser armazenados, para cada funcionário, em um dado mês/ano, o valor do salário base, o valor total das comissões e o total dos proventos a serem pagos. Isto deve ser calculado até o segundo dia do mês subseqüente.

- Procedimento batch que vai pegarSUM(vendas)*0.05+cargos.salario, das tabelas **Cliente_Funcionario**, dar join com **NotasVenda** on cpf , join com **CargosFunc** on cpf join com **Cargos** on codigo group by cpf, pegando assim as vendas. Esse processo batch vai se repetir todo dia primeiro de cada mês.

9. As funcionalidades mais frequentemente utilizadas do sistema (e que merecem atenção especial no que se refere ao desempenho) são:

- **Consulta de aquisições em atraso de um determinado fornecedor;**

Criamos uma visão chamada **Produtos_nao_recebidos** que contém apenas os produtos que os fornecedores não entregaram, para um determinado fornecedor basta fazer um select by ID.

- **Cadastro de produtos com todas as suas informações;**

Criamos uma Procedure **Cadastra_produto** que recebe os dados do produto e o cadastra.

- **Consulta dos N produtos mais vendidos;**

Criamos a função get_best_sellers(N) que vai retornar os N itens mais vendidos em valor absoluto

- **Consulta dos N maiores clientes;**
  Criamos a função get_best_clientes(N) que retorna o id e valor total de compras de cada cliente em ordem crescente até o Nº elemento
- **Consulta dos N maiores fornecedores;**
  Criamos a função get_best_fornecedor(N) que retorna o id e valor total de compras com o fornecedor

- **Consulta a dados de um fornecedor/cliente X.**

Criamos uma função **GET_CLIENTE_FORNECEDOR** que recebe o ID e retorna os dados do cliente/fornecedor.

- **Consulta a estoque e preço de um produto.**

Criamos uma visão **ESTOQUE_PRECO_MERCADORIA** que guarda o estoque e o preço mínimo de uma mercadoria, em seguida a função **GET_ESTOQUE_PRECO** que recebe o ID de um produto e retorna seu estoque atual e seu preço mínimo de venda.

- **Consulta a dados de um pedido X.**

Criamos a função **GET_DADOS_PEDIDO_COMPRA** que recebe o ID da nota de compra e retorna os dados referentes a essa nota.

- **Consulta a dados de uma Nota Fiscal X.**

Criamos a função **GET_DADOS_NOTA_FISCAL** que recebe o ID da nota fiscal desejada e retorna os dados referentes a essa nota.

---------- ABAIXO ESTÃO TODOS OS CÓDIGOS----------

Criação das tabelas:

-- Cliente_Fornecedor

```
CREATE TABLE CLIENTE_FORNECEDOR
(
  CODIGO INTEGER NOT NULL,
  NOME VARCHAR2(30) ,
  TELEFONE VARCHAR2(15) ,
  LOGRADOURO VARCHAR2(50) ,
  NUMERO INTEGER ,
  COMPLEMENTO VARCHAR2(100) ,
  CIDADE VARCHAR2(100) ,
  ESTADO VARCHAR2(30) ,
  NUMEROCONTRIBUINTE INTEGER ,
```

```
  CNPJ VARCHAR(18) ,

  CONSTRAINT CLIENTE_FORNECEDOR_PK PRIMARY KEY
  (
        CODIGO
  ) ENABLE
);
```

-- Funcionario_nova:

```
CREATE TABLE funcionario_nova
(
cpf integer not null,
nome VARCHAR2(50),
telefone VARCHAR2(15),
email VARCHAR(50),
logradouro VARCHAR2(50),
numero integer,
complemento VARCHAR2(80),
Cidade VARCHAR2(80),
Estado VARCHAR2(30),
CodigoDepartamento integer,
CONSTRAINT funcionario_pk PRIMARY KEY (cpf)
);

alter table funcionario_nova add
constraint funcionario_codigodep_fk foreign key (CodigoDepartamento)
REFERENCES departamento (codigoDepartamento);
```

-- CargosFunc:

```
CREATE TABLE cargosFunc
(
cpf integer not null,
CodigoCargo integer,
dataInicio date,
dataFim date,
```

```
CONSTRAINT cargosFunc_pk PRIMARY KEY (cpf,CodigoCargo,dataInicio)
);

alter table cargosfunc add
constraint cargosfunc_codigocargo_fk foreign key (CodigoCargo)
REFERENCES cargo (codigo);
```
-- Cargo:

```
CREATE TABLE cargo
(
codigo integer not null PRIMARY KEY,
descricao varchar2(50),
salario_base integer
);
```

-- Departamento:

```
CREATE TABLE departamento
(
codigoDepartamento integer not null,
nome varchar2(50),
cpf_chefe integer,
CONSTRAINT departamento_pk PRIMARY KEY (codigoDepartamento),
constraint departamento_fk foreign key (cpf_chefe) REFERENCES
funcionario_nova (cpf)
);
```

-- Mercadorias:

```
CREATE TABLE MERCADORIAS_NOVA
(
  NUMEROMERCADORIA INTEGER NOT NULL,
  QUANTIDADE INTEGER ,
  DESCRICAO  VARCHAR2(50) ,
  ESTOQUE_MAX INTEGER,

  CONSTRAINT MERCADORIAS_PK PRIMARY KEY
  (
```

```
        NUMEROMERCADORIA
  ) ENABLE
);
```

-- NotasVenda

```
CREATE TABLE NOTASVENDA
(
  NUMERO INTEGER NOT NULL,
  DATAEMISSAO DATE ,
  FORMAPAGAMENTO VARCHAR(15),
  CODIGOCLIENTE INTEGER ,
  CPFVENDEDOR INTEGER,

  CONSTRAINT CLIENTE_FK
        FOREIGN KEY (CODIGOCLIENTE)
        REFERENCES CLIENTE_FORNECEDOR(CODIGO),

  CONSTRAINT FUNCIONARIO_FK
        FOREIGN KEY (CPFVENDEDOR)
        REFERENCES FUNCIONARIO_nova(CPF),

  CONSTRAINT NOTASVENDA_PK PRIMARY KEY
  (
        NUMERO
  ) ENABLE

);
```

-- ItensNota

```
CREATE TABLE ITENSNOTA
(
  NUMERO INTEGER NOT NULL,
  NUMEROMERCADORIA INTEGER ,
  QUANTIDADE INTEGER ,
  VALORUNITARIO NUMBER(*,2) ,
```

```
  CONSTRAINT MERCADORIA_FK
        FOREIGN KEY (NUMEROMERCADORIA)
        REFERENCES MERCADORIAS_NOVA(NUMEROMERCADORIA),


  CONSTRAINT ITENSNOTA_PK PRIMARY KEY
  (
        NUMERO,
        NUMEROMERCADORIA
  ) ENABLE

);
```

-- NotasCompra:

```
CREATE TABLE NOTASCOMPRA
(
  NUMERO INTEGER NOT NULL,
  DATAEMISSAO DATE ,
  CODIGOFORNECEDOR INTEGER,

  CONSTRAINT NotaCompra_fornecedor_FK
        FOREIGN KEY (CODIGOFORNECEDOR)
        REFERENCES CLIENTE_FORNECEDOR(CODIGO),

  CONSTRAINT NOTASCOMPRA_PK PRIMARY KEY
  (
        NUMERO
  ) ENABLE

);
ALTER TABLE NOTASCOMPRA
  ADD entregue  NUMBER(1) DEFAULT 0;
```

-- Itens_Nota_Compra:

```
CREATE TABLE ITENSNOTACOMPRA
(
```

```
  NUMERO INTEGER NOT NULL,
  NUMEROMERCADORIA INTEGER ,
  QUANTIDADE INTEGER ,
  VALORUNITARIO NUMBER(*,2) ,

  CONSTRAINT MERCADORIA_COMPRA_FK
        FOREIGN KEY (NUMEROMERCADORIA)
        REFERENCES MERCADORIAS_NOVA(NUMEROMERCADORIA),
CONSTRAINT NUMERO_COMPRA_FK
        FOREIGN KEY (NUMERO)
        REFERENCES NOTASCOMPRA(NUMERO),

  CONSTRAINT ITENSNOTACOMPRA_PK PRIMARY KEY
  (
        NUMERO,
        NUMEROMERCADORIA
  ) ENABLE

);
```

-- preco_min_venda:

```
CREATE TABLE  "PRECO_MIN_VENDA"
   (      "CODMERCADORIA" NUMBER(*,0) NOT NULL ENABLE,
          "ULTIMOPRECO" NUMBER(*,2),
           CONSTRAINT "PRECO_MIN_VENDA_PK" PRIMARY KEY
("CODMERCADORIA") ENABLE
   ) ;ALTER TABLE  "PRECO_MIN_VENDA" ADD CONSTRAINT
"PRECO_MIN_VENDA_FK" FOREIGN KEY ("CODMERCADORIA")
            REFERENCES  "MERCADORIAS_NOVA"
("NUMEROMERCADORIA") ENABLE;
```

Criação das Visões:

-- Mercadorias

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW
"C##1611074"."Mercadorias" ("NumeroMercadoria", "Descricao",
"QuantidadeEstoque") AS
  SELECT NumeroMercadoria, Descricao, Quantidade
FROM Mercadorias_Nova;
```

-- Clientes

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW
"C##1611074"."CLIENTE" ("CODIGO", "NOME", "TELEFONE",
"LOGRADOURO","NUMERO" ,"COMPLEMENTO", "CIDADE", "ESTADO",
"NUMEROCONTRIBUINTE") AS
  SELECT CODIGO, NOME, TELEFONE, LOGRADOURO, NUMERO
,COMPLEMENTO, CIDADE, ESTADO, NUMEROCONTRIBUINTE
FROM CLIENTE_FORNECEDOR;
```

-- Funcionario

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW
"C##1611074"."FUNCIONARIO" ("CPF", "NOME", "TELEFONE",
"LOGRADOURO","NUMERO" ,"COMPLEMENTO", "CIDADE", "ESTADO",
"CodigoDepartamento") AS
  SELECT "CPF", "NOME", "TELEFONE", "LOGRADOURO","NUMERO"
,"COMPLEMENTO", "CIDADE", "ESTADO", "CODIGODEPARTAMENTO"
FROM FUNCIONARIO_NOVA;
```

Criação das Triggers:

-- TRIGGER ATUALIZA_PRECO_MIN:

```
create or replace Trigger atualiza_preco_min AFTER INSERT
ON ITENSNOTACOMPRA
REFERENCING NEW AS Novo
FOR EACH ROW
BEGIN
  INSERT INTO preco_min_venda (codmercadoria,ultimopreco)
  VALUES (:Novo.numeromercadoria, :Novo.valorunitario);
EXCEPTION
```

```
  WHEN DUP_VAL_ON_INDEX THEN
        UPDATE preco_min_venda
        SET   ultimopreco = :Novo.valorunitario
        WHERE CODMERCADORIA = :Novo.numeromercadoria;
end atualiza_preco_min;
```

-- TRIGGER IMPEDE_VENDA_SEM_ESTOQUE: -20001 -> sem estoque

```
create or replace Trigger impede_venda_sem_estoque BEFORE UPDATE
ON Mercadorias_Nova
REFERENCING NEW AS Novo
FOR EACH ROW
BEGIN
IF :Novo.Quantidade < 0 then
raise_application_error( -20001, 'Não há estoque disponivel' );
END IF;
end IMPEDE_VENDA_SEM_ESTOQUE;
```

-- TRIGGER impede_compra_acima_max: -20002 -> estoque max atingido

```
create or replace Trigger impede_compra_acima_max BEFORE UPDATE
ON Mercadorias_Nova
REFERENCING NEW AS Novo OLD AS Velho
FOR EACH ROW
BEGIN
IF :Novo.Quantidade > :Velho.estoque_max then
raise_application_error( -20002, 'Estoque maximo atingido' );
END IF;
end impede_compra_acima_max;
```

-- Criação das Funções:

-- GET_CLIENTE_FORNECEDOR Recebe o id e retorna os dados do cliente ou fornecedor com esse ID

```
create or replace function get_cliente_fornecedor
(cf_id in CLIENTE_FORNECEDOR.CODIGO%Type)
```

```
return CLIENTE_FORNECEDOR%rowtype
as
l_cust_record CLIENTE_FORNECEDOR%rowtype;
begin
select * into l_cust_record from CLIENTE_FORNECEDOR
where CODIGO=cf_id;
return(l_cust_record);
end;
```

-- get_preco_min Recebe o id do item

```
create or replace function get_preco_min
(item_id in MERCADORIAS_NOVA.NUMEROMERCADORIA%Type)
return PRECO_MIN_VENDA.ULTIMOPRECO%type
as
preco_min PRECO_MIN_VENDA.ULTIMOPRECO%type;
begin
select ULTIMOPRECO into preco_min from PRECO_MIN_VENDA
where CODMERCADORIA=item_id;
return(preco_min);
end;
```

-- get_estoque_preco Recebe o id do item

```
create or replace function get_estoque_preco
(item_id in mercadorias_nova.NUMEROMERCADORIA%Type)
return "C##1611074"."Estoque_preco_mercadorias"%rowtype
as
preco_estoque "C##1611074"."Estoque_preco_mercadorias"%rowtype;
begin
select * into preco_estoque from
"C##1611074"."Estoque_preco_mercadorias"
where
"C##1611074"."Estoque_preco_mercadorias"."NumeroMercadoria"=item_id;
return(preco_estoque);
end;
```

-- get_dados_pedido_compra Recebe o id da nota de compra

```
create or replace function get_dados_pedido_compra
(pedido_id in NOTASCOMPRA.NUMERO%Type)
return NOTASCOMPRA%rowtype
as
dados_compra NOTASCOMPRA%rowtype;
begin
select * into dados_compra from NOTASCOMPRA
where NUMERO =pedido_id;
return(dados_compra);
end;
```

-- get_dados_nota_fiscal Recebe o id da nota fiscal desejada

```
create or replace function get_dados_nota_fiscal
(pedido_id in NOTASVENDA.NUMERO%Type)
return NOTASVENDA%rowtype
as
dados_venda NOTASVENDA%rowtype;
begin
select * into dados_venda from NOTASVENDA
where NUMERO =pedido_id;
return(dados_venda);
end;
```

-- **GET_BEST_SELLERS:**
- recebe N, até qual elemento devemos retornar

```
create or replace function get_best_sellers
(N in NUMBER)
return SYS_REFCURSOR
as
  rf_cur   SYS_REFCURSOR;
begin
open rf_cur for
SELECT * FROM
```

```
        (Select
        ItensNota.NUMEROMERCADORIA,sum(ItensNota.QUANTIDADE *
        VALORUNITARIO) as vendas From NotasVenda join ItensNota on
        NotasVenda.numero = ItensNota.numero group by
        ItensNota.NUMEROMERCADORIA order by vendas DESC)
WHERE rownum <= N

return(rf_cur);
end get_best_sellers;
```

-- GET_BEST_CLIENTES

-recebe N, até qual elemento devemos retornar

```
create or replace function get_best_clientes
(N in NUMBER)
return SYS_REFCURSOR
as
  rf_cur   SYS_REFCURSOR;
begin
open rf_cur for
SELECT * FROM
        (Select Cliente_Fornecedor.Codigo,sum(ItensNota.QUANTIDADE *
        VALORUNITARIO) as vendas From NotasVenda join ItensNota on
        NotasVenda.numero = ItensNota.numero
         join Cliente_Fornecedor on NotasVenda.CodigoCliente =
        Cliente_Fornecedor.Codigo group by Cliente_Fornecedor.Codigo order
        by vendas DESC)
WHERE rownum <= N
return(rf_cur);
end get_best_clientes;
```

-- get_best_fornecedores

- recebe N que indica quantas rows vamos retornar

```
create or replace function get_best_fornecedores
(N in NUMBER)
return SYS_REFCURSOR
as
  rf_cur   SYS_REFCURSOR;
```

```
begin
open rf_cur for
SELECT * FROM
        (Select Cliente_Fornecedor.Codigo,sum(QUANTIDADE *
        VALORUNITARIO) as vendas From NotasCompra join
        ItensNotaCompra on NotasCompra.numero =
        ItensNotaCompra.numero
         join Cliente_Fornecedor on NotasCompra.CodigoCliente =
        Cliente_Fornecedor.Codigo group by Cliente_Fornecedor.Codigo order
        by vendas DESC)
WHERE rownum <= N

return(rf_cur);
end get_best_fornecedores;
```

-- Stored Procedure

```
create or replace PROCEDURE cadastra_produto (id_M IN NUMBER, descri
IN
VARCHAR2, qt IN NUMBER, e_max IN NUMBER)
as
begin
insert into MERCADORIAS_NOVA (
NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO )
VALUES ( id_M,qt, e_max ,descri );
end cadastra_produto;
```

----------INSERTS PARA TESTE----------

-- CLIENTE_FORNECEDOR

```
insert into cliente_fornecedor values (1,'ARI','(21)1111-1111','RUA A', 1,'AAA',
'RJ', 'RJ', 1, '1');
```

```
insert into cliente_fornecedor values (2,'BER','(21)2222-2222','RUA B',
2,'BBB', 'RJ', 'RJ', 2, '2');
insert into cliente_fornecedor values (3,'CER','(21)3333-3333','RUA C',
3,'CCC', 'RJ', 'RJ', 3, '3');
insert into cliente_fornecedor values (4,'EDU','(21)4444-4444','RUA D',
4,'DDD', 'RJ', 'RJ', 4, '4');
insert into cliente_fornecedor values (5,'ROFL','(21)5555-5555','RUA E',
5,'EEE', 'RJ', 'RJ', 5, '5');
insert into cliente_fornecedor values (6,'KAKA','(21)6666-6666','RUA F',
6,'FFF', 'RJ', 'RJ', 6, '6');
insert into cliente_fornecedor values (7,'JOHN','(21)7777-7777','RUA G',
7,'GGG', 'RJ', 'RJ', 7, '7');
insert into cliente_fornecedor values (8,'RAPHA','(21)8888-8888','RUA H',
8,'HHH', 'RJ', 'RJ', 8, '8');
insert into cliente_fornecedor values (9,'BOB','(21)9999-9999','RUA I', 9,'III',
'RJ', 'RJ', 9, '9');
```

-- CARGO

```
insert into CARGO values (1, 'AAA', 100);
insert into CARGO values (2, 'BBB', 200);
insert into CARGO values (3, 'CCC', 300);
insert into CARGO values (4, 'DDD', 400);
insert into CARGO values (5, 'EEE', 500);
insert into CARGO values (6, 'FFF', 600);
insert into CARGO values (7, 'GGG', 700);
insert into CARGO values (8, 'HHH', 800);
insert into CARGO values (9, 'III', 900);
```

-- MERCADORIAS_NOVA

```
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (1, 100, 100, 'AAA');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (2, 200, 200, 'BBB');
```

```
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (3, 300, 300, 'CCC');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
 values (4, 400, 400, 'DDD');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (5, 500, 500, 'EEE');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (6, 600, 600, 'FFF');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (7, 700, 700, 'GGG');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (8, 800, 800, 'HHH');
insert into MERCADORIAS_NOVA
(NUMEROMERCADORIA,QUANTIDADE,ESTOQUE_MAX,DESCRICAO)
values (9, 900, 900, 'III');
```

-- CARGOSFUNC

```
insert into CARGOSFUNC values (1, 1, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (2, 2, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (3, 3, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (4, 4, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (5, 5, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (6, 6, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (7, 7, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (8, 8, '1/1/1991', '10/10/2010');
insert into CARGOSFUNC values (9, 9, '1/1/1991', '10/10/2010');
```

-- DEPARTAMENTO

```
insert into DEPARTAMENTO values (1,'AAA', null);
insert into DEPARTAMENTO values (2,'BBB', null);
insert into DEPARTAMENTO values (3,'CCC', null);
```

```
insert into DEPARTAMENTO values (4,'DDD', null);
insert into DEPARTAMENTO values (5,'EEE', null);
insert into DEPARTAMENTO values (6,'FFF', null);
insert into DEPARTAMENTO values (7,'GGG', null);
insert into DEPARTAMENTO values (8,'HHH', null);
insert into DEPARTAMENTO values (9,'III', null);
```

-- FUNCIONARIO_NOVA

```
insert into FUNCIONARIO_NOVA values
(1,'ARI','(21)1111-1111','AAA@HOTMAIL.COM','RUA A', 1,'AAA', 'RJ', 'RJ', 1);
insert into FUNCIONARIO_NOVA values
(2,'BER','(21)2222-2222','BBB@HOTMAIL.COM','RUA B', 2,'BBB', 'RJ', 'RJ',
2);
insert into FUNCIONARIO_NOVA values
(3,'CER','(21)3333-3333','CCC@HOTMAIL.COM','RUA C', 3,'CCC', 'RJ', 'RJ',
3);
insert into FUNCIONARIO_NOVA values
(4,'EDU','(21)4444-4444','DDD@HOTMAIL.COM','RUA D', 4,'DDD', 'RJ', 'RJ',
4);
insert into FUNCIONARIO_NOVA values
(5,'ROFL','(21)5555-5555','EEE@HOTMAIL.COM','RUA E', 5,'EEE', 'RJ', 'RJ',
5);
insert into FUNCIONARIO_NOVA values
(6,'KAKA','(21)6666-6666','FFF@HOTMAIL.COM','RUA F', 6,'FFF', 'RJ', 'RJ',
6);
insert into FUNCIONARIO_NOVA values
(7,'JOHN','(21)7777-7777','GGG@HOTMAIL.COM','RUA G', 7,'GGG', 'RJ',
'RJ', 7);
insert into FUNCIONARIO_NOVA values
(8,'RAPHA','(21)8888-8888','HHH@HOTMAIL.COM','RUA H', 8,'HHH', 'RJ',
'RJ', 8);
insert into FUNCIONARIO_NOVA values
(9,'BOB','(21)9999-9999','III@HOTMAIL.COM','RUA I', 9,'III', 'RJ', 'RJ', 9);
```

-- PRECO_MIN_VENDA

```
  INSERT ALL
```

```
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (1, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (2, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (3, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (4, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (5, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (6, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (7, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (8, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (9, 0)
  INTO PRECO_MIN_VENDA ( CODMERCADORIA,ULTIMOPRECO)
VALUES (12, 0)
  SELECT * FROM dual;
```

-- NOTASCOMPRA:

```
insert into NOTASCOMPRA values (1, '27/09/2018',1);
insert into NOTASCOMPRA values (2, '11/11/2011',2);
insert into NOTASCOMPRA values (3, '11/11/2011',3);
insert into NOTASCOMPRA values (4, '11/11/2011',4);
insert into NOTASCOMPRA values (5, '11/11/2011',5);
insert into NOTASCOMPRA values (6, '11/11/2011',6);
insert into NOTASCOMPRA values (7, '11/11/2011',7);
insert into NOTASCOMPRA values (8, '11/11/2011',8);
insert into NOTASCOMPRA values (9, '11/11/2011',9);
```

-- ITENSNOTACOMPRA :

```
insert into ITENSNOTACOMPRA values (2, 1,20,250);
```

--ITENSNOTA

```
insert into ITENSNOTA values (1, 1,20,250);
insert into ITENSNOTA values (2, 2,20,250);
insert into ITENSNOTA values (3, 3,20,250);
insert into ITENSNOTA values (4, 4,20,250);
insert into ITENSNOTA values (5, 5,20,250);
insert into ITENSNOTA values (6, 6,20,250);
insert into ITENSNOTA values (7, 7,20,250);
insert into ITENSNOTA values (8, 8,20,250);
insert into ITENSNOTA values (9, 9,20,250);
```

--NOTASVENDA

```
insert into NOTASVENDA values (1, '11/11/2011','Cartao', 1, 1);
insert into NOTASVENDA values (2, '11/11/2011','Cartao', 2, 2);
insert into NOTASVENDA values (3, '11/11/2011','Cartao', 3, 3);
insert into NOTASVENDA values (4, '11/11/2011','Cartao', 4, 4);
insert into NOTASVENDA values (5, '11/11/2011','Cartao', 5, 5);
insert into NOTASVENDA values (6, '11/11/2011','Cartao', 6, 6);
insert into NOTASVENDA values (7, '11/11/2011','Cartao', 7, 7);
insert into NOTASVENDA values (8, '11/11/2011','Cartao', 8, 8);
insert into NOTASVENDA values (9, '11/11/2011','Cartao', 9, 9);
```