Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

Banco de Dados - IMD0401

RELATÓRIO PROJETO

Banco de Dados Representação Comercial

Adriana Virginia Macedo de Azevedo

Natal - novembro de 2017

1. CARACTERIZAÇÃO DO PROJETO

1.1. Introdução

Esse projeto refere-se ao desenvolvimento de um sistema de banco de dados desenvolvido para uma empresa de representação comercial. O banco criado busca o armazenamento e controle dos dados da empresa, dos clientes, dos fornecedores, dos produtos e auxiliando o acompanhamento e geração de pedidos dos clientes. A empresa trabalha com diversos fornecedores e um grande mix de produtos, não tendo nenhum estoque físico. Os pedidos recebidos dos clientes são digitados por um funcionário que atualmente gera manualmente o pedido para ser enviado ao abastecedor. Alguns dos fornecedores faturam os pedidos de unidades filiais. Na captação de um pedido pode ser necessário que seja gerado uma reserva, que algumas vezes é confirmada e em outras precisa de prévia autorização.
Atualmente a empresa trabalha com controle através de planilhas no Excel. O projeto consiste no desenvolvimento de um banco de dados que atenda a demandas comerciais e geração de relatórios de acompanhamentos. Possibilitando um amplo conhecimento de todos os pontos do processo do recebimento do pedido ao atendimento do mesmo.

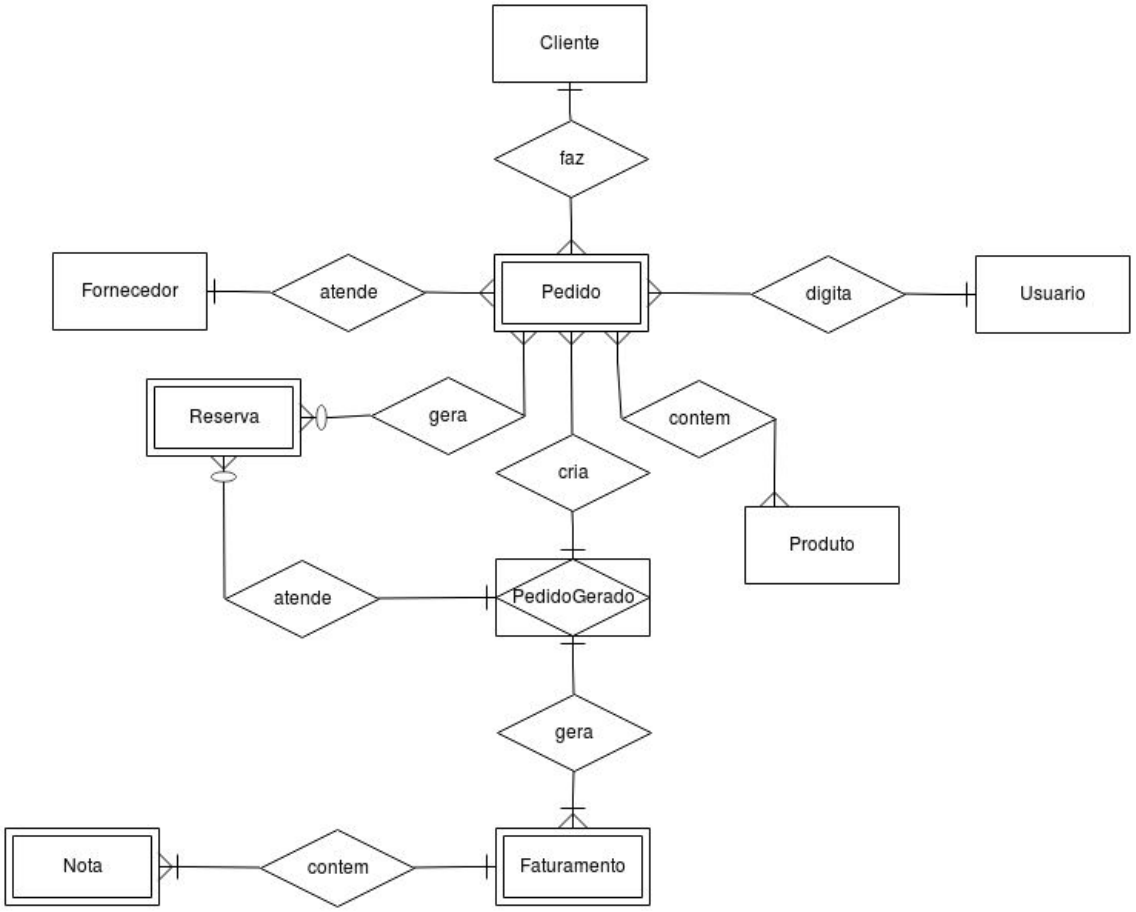
1.2. Objetivos

O projeto busca primeiramente armazenar todos os dados referentes a clientes, fornecedores, produtos e pedidos. De posse desses dados a geração de relatórios referentes a pedidos gerados, volume de compra por cliente, período de compra por cliente, itens mais vendidos, preço médio, volume faturado por fornecedor entre outros. O projeto foi desenvolvido buscando evitar redundâncias, permitindo facilidades de atualizações e evitando inconsistência de dados.
Devido ao processo ser muito grande, com diversas entidades e vários relacionamentos ficou definido que primeiramente seria desenvolvido um banco que fosse possível cadastrar clientes, fornecedores, produtos e pedidos realizados pelos clientes. Tais pedidos são recebidos normalmente semanalmente e agregados para formar diversos pedidos que são

enviados ao Fornecedor podendo ter um ou mais clientes cada pedido, dependendo do volume solicitado.
Os pedidos que são enviados podem ser individuais ou com vários clientes, vai depender para qual fornecedor será encaminhado. As regras para o envio dos pedidos não serão amplamente trabalhadas nessa etapa do projeto, ficando para etapas futuras.
A regra que será desenvolvida é para o recebimento do pedido do cliente e cadastramento do mesmo, porque observou-se que para geração do pedido enviado ao fornecedor tornou-se necessário a criação de outra entidade, PedidoGerado, tal observação surgiu devido aos pedidos gerados de alguns fornecedores possuírem regras distintas e próprias. Entre essas regras foram verificado que fornecedores tem várias unidades cadastradas, tendo os pedidos sendo faturados nas mesmas e precisando atender ao limite de peso médio para envio entre outras coisas. Por exemplo no pedido que deve ser enviado ao Frigorífico precisa ter em média dezessete toneladas.
Nessa etapa do processo será acompanhado o cadastramento de clientes, fornecedores, usuários, unidades e os pedidos dos clientes.
Portanto será desenvolvido as entidades cliente, fornecedor, unidade, usuário, produto e pedido e todas as demais entidades utilizadas para atender as normalizações necessárias.

1.3. Modelo Conceitual

Foi desenvolvido pela identificação dos conjuntos de informações que contém no processo da empresa e as ligações existentes entre esses conjuntos, o conhecimento do funcionamento da empresa auxiliou muito a compreensão e determinação desses conjuntos.
A definição das entidades ocorreu pela percepção dos objetos distinguíveis com características próprias. Inicialmente observou ser necessário cinco entidades, fornecedor, cliente, usuário, produto e pedido, as demais surgiram durante o desenvolvimento.
No processo de criação do banco foi observado que existem quatro entidades fortes, Cliente, Fornecedor, Usuario e Produto e as entidades fracas Pedido, Reserva, Faturamento e Nota que dependem de várias entidades para existir.
A ligação entre as entidades ocorre através das relações que são todas ações, podendo ser também evento, que não ocorreu nesse desenvolvimento.
As entidades têm características próprias que a definem, os atributos. Todas as entidades contém diversos atributos percebidos no processo, porém observe-se que durante a criação do banco novos atributos são descobertos.
Segue a representação gráfica do projeto através do Diagrama de Entidade Relacionamento (DER) principal ferramenta que auxilia o entendimento e compreensão, muitas vezes chamado do modelo do projeto:



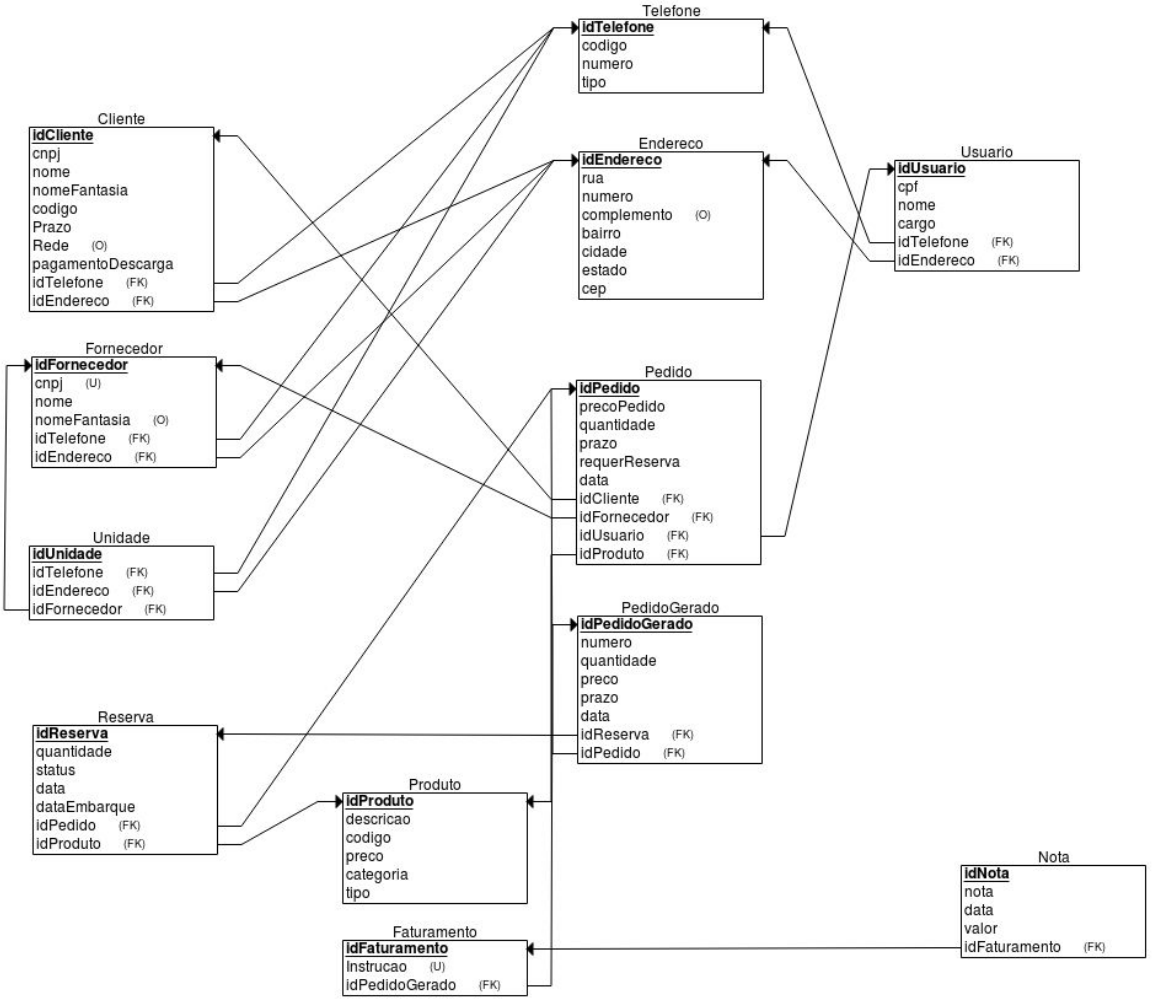
Para tornar a visualização mais clara e limpa os atributos das entidades no Modelo Conceitual, na DER estão relacionados abaixo:

- **Fornecedor** (idFornecedor, cnpj, nome, nomeFantasia, idEndereco, idTelefone)
- **Cliente** (idCliente,cnpj, nome, nomeFantasia, idEndereco, idTelefone, prazo, codigo, rede, pagamentoDescarga)
- **Telefone** (idTelefone, codigo, numero, tipo)
- **Endereco** (idEndereco, rua, numero, complemento, bairro, cidade, estado, cep)
- **Unidade** (idUnidade, idFornecedor, idPessoaJuridica)
- **Usuario** (idUsuario, cpf, nome, idEndereco, idTelefone, cargo)
- **Produto** (idProduto, idFornecedor, descricao, codigo, preco, categoria, tipo)
- **Pedido** (idPedido, idFornecedor, idProduto, idCliente, precoPedido, quantidade, prazo, requerReserva, data)
- **PedidoGerado** (idPedidoGerado, idReserva, idPedido, quantidade, numero, preco, prazo, data)
- **Reserva** (idReserva, idPedido, quantidade, status, data, dataEmbarque)
- **Faturamento** (idFaturamento, idPedido, instrucao)
- **Nota** (idNota, idFaturamento, nota, valor, data)

Como pode ser observado no modelo existem diversos relacionamentos com cardinalidade 1xn e existem os relacionamentos 1x1 que não representados no modelo, como o relacionamento da entidade Usuario com a entidade Endereco, a entidade Fornecedor com a entidade Endereco e a entidade Cliente com a entidade Endereco, pois não é possível o relacionamento com diversas cardinalidades, fornecedor e cliente precisam que o cnpj esteja atrelado a um só endereço. As entidade que tinha relacionamento nxm era Pedido e Faturamento, porém foi alterado para atender demanda do projeto, sendo acrescentado a entidade PedidoGerado.
As entidades PedidoGerado, Reserva, Faturamento e Nota não serão entregues neste projeto, ficaram para desenvolvimento futuro.
O DER foi apresentado a empresa de representação comercial para uma consulta do entendimento do projeto, momento este que auxiliou o complemento e entendimento de todo o processo, após tal consulta verificou-se a necessidade de criação de mais atributos em algumas entidade e também a criação de uma nova entidade, que inicialmente não tinha sido verificado, PedidoGerado (entidade associativa), que antes era o relacionamento entre as entidades Pedido e Faturamento, como esse relacionamento era de muitos para muitos possibilitou a criação da entidade associativa. Comprovando assim a importância e o auxílio do modelo DER para comunicação com o cliente.

1.4. Modelo Lógico

Desenvolvido baseado no modelo conceitual buscou a correta definição das chaves primárias e estrangeiras, a normalização e integridade referencial.
O modelo lógico sofreu diversas alterações no decorrer do desenvolvimento, observando-se a necessidade de contínua adequação e refinamento do processo.
A escolha das chaves primárias aconteceu de forma simples e praticamente todas as entidades tem chave primária simples.
A única entidade que contém chave primária composta é Reserva, pois para tal entidade existir vai depender totalmente de um pedido, sua chave é composta do indexi de identificação da reserva (idReserva) e do indexi de identificação do pedido (idPedido).
O modelo utilizado atende a 1FN, não contém nenhum atributo multivalorado e composto, foram criadas as entidades endereço, telefone, nota e reserva para garantir que a 1FN fosse atendida. Também atende a 2FN, todos os atributos não chaves são completamente dependentes das chaves. E a 3FN também é atendida, como pode ser observado nenhum atributo tem dependência transitiva.



Inicialmente foi prevista duas entidades PessoaJuridica para não haver repetição dos campos de comuns de Fornecedor e Cliente, porém devido a algumas dificuldades tal desenvolvimento foi postergado, para etapas futuras.
Já foi observado a necessidade de alteração do modelo para adequação do desenvolvimento da entidade Reserva, pois a mesma necessita da referência da entidade Produto, que será feito futuramente.

1.5. Modelo Físico

A estrutura física do banco de dados ocorreu com a criação de banco denominando DBRepresentacao, o banco foi criado como usuário postgres, no padrão de codificação Português/Brasil e sem limite de usuário:

```
CREATE DATABASE "DBRepresentacao"
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'pt_BR.UTF-8'
       LC_CTYPE = 'pt_BR.UTF-8'
```

```
CONNECTION LIMIT = -1;
```

Foram criadas todas as tabelas das entidades que constam no modelo lógico que seriam desenvolvidas no projeto, com seus atributos e chaves primárias. A maior parte dos atributos são do tipo varchar, que aceita tanto texto como número.
Nas tabelas Endereco, Telefone, Fornecedor, Cliente, Usuario foram criadas constrains do tipo Check para entrada de valores.

```
create table Cliente(
   idCliente serial,
   cnpj varchar(14) not null check (cnpj similar to '(\d)+'),
   nome varchar(50) null,
   nomeFantasia varchar(30) null,
   codigo integer,
   rede varchar(20),
   pagamentoDescarga boolean,
   idEndereco integer not null references Endereco,
   idTelefone integer not null references Telefone,
   prazo varchar(12) null,
   Primary key(idCliente)
);
```

Também foi colocado como comentário o comando necessário para apagar todas as tabelas.
A inserção de dados na maior parte das tabelas foi realizado pela criação de uma procedure function para otimizar o cadastramento, a única tabela que não é preenchida através de procedure é a tabela Pedido.

```
--Procedure para cadastrar telefones
create or replace function cadastrar_telefone_s(codigoT varchar, numeroT integer, tipoT
varchar) returns void AS $$
Begin
   insert into telefone (codigo, numero, tipo) values (codigoT, numeroT, tipoT);
End;
$$ Language 'plpgsql';

--Povoando tabela telefone
select cadastrar_telefone_s('84','987654321','Cel');
```

Foi criado o gatilho pedido_insert para garantir que quando seja necessário fazer uma reserva, receba-se a mensagem e o pedido será incluído juntamente com a inclusão da reserva. O processo de inclusão da reserva não faz parte do escopo do projeto, por isso esses pedidos não são incluídos ainda.

```
--Criando função testar_reserva para testar se existe necessidade de fazer a reserva
create or replace function testar_reserva() returns trigger AS $testar_reserva$
Begin
   If NEW.requerReserva is true then
         Raise Exception 'Precisa ser feita uma reserva';
   End If;
   Return NEW;
End;
```

```
$testar_reserva$ Language 'plpgsql';

--Criando a trigger para inserção de pedido
create trigger pedido_insert before insert or update on pedido
for each row execute
procedure testar_reserva();
```

Para o relatório da relação dos clientes da cidade de Natal um view foi criada:

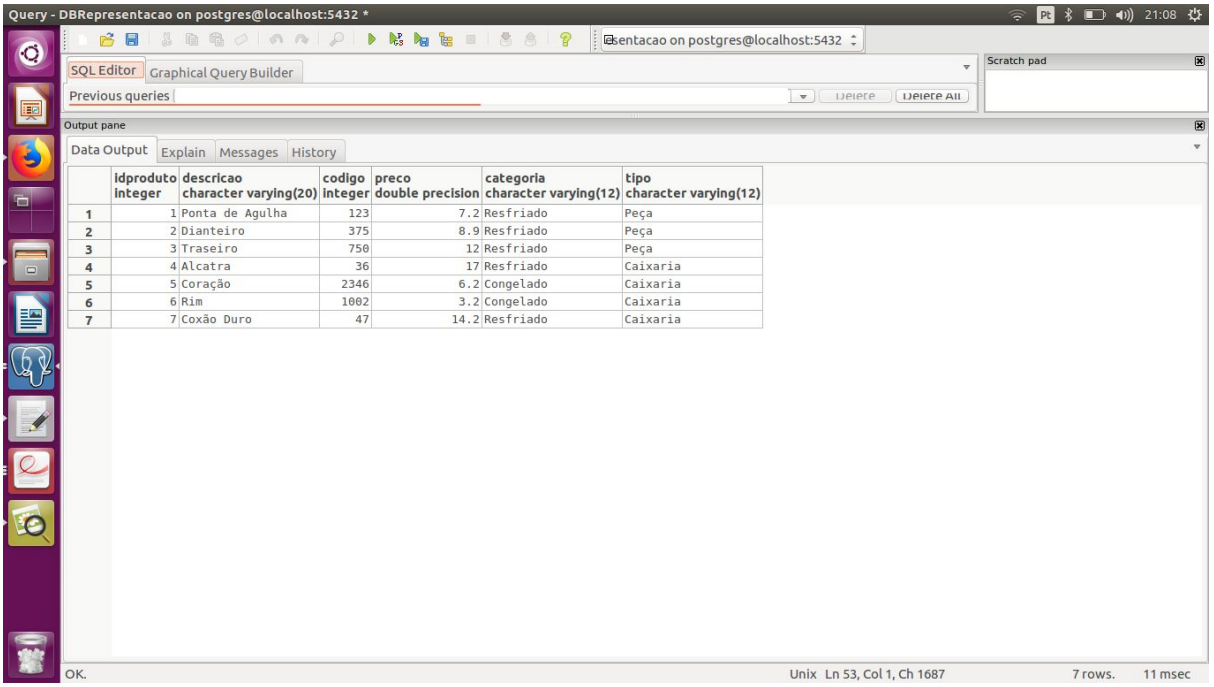
```
--Criação de view com relação dos cliente de Natal
create view vw_relacaoClientesNatal as
select c.nome, c.nomeFantasia, e.bairro, t.codigo, t.numero
from cliente c
   join endereco e on c.idendereco = e.idendereco
   join telefone t on c.idtelefone = t.idtelefone
where e.cidade = 'Natal'
order by c.nome;

select * from vw_relacaoClientesNatal;
```

Outro relatório foi criado através de view, pedidos dos cliente, onde está relacionado o nome do cliente, produto pedido e quantidade:

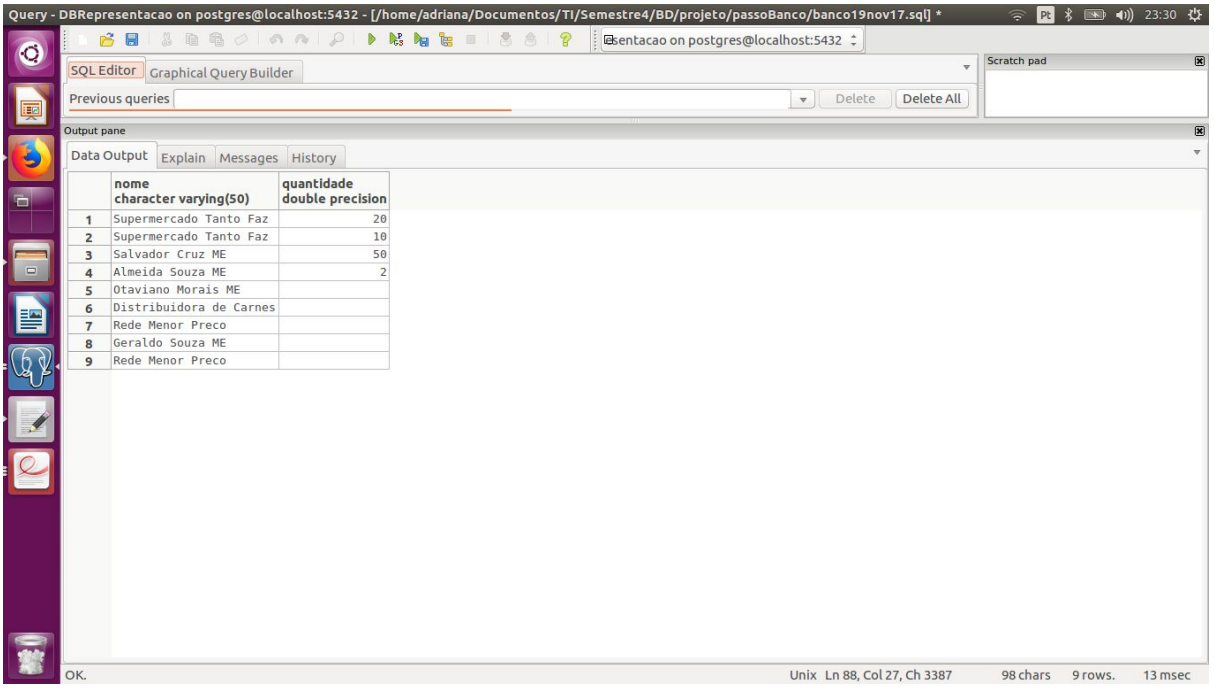
```
--Criação de view com os pedidos dos clientes
create view vw_pedidoCliente as
select c.nome, p.quantidade, pd.descricao
from pedido p
   join produto pd on p.idproduto = pd.idproduto
   join cliente c on c.idcliente = p.idcliente
order by c.nome;

select * from vw_pedidoCliente;
```

Também foi implementado consultadas diretas através do comando select e consultas utilizando join e produto cartesiano:

```
select * from produto;
```




```
select c.nome, p.quantidade from pedido p
    full outer join cliente c on
    c.idcliente = p.idcliente;
```



1.6. Conclusão

Durante o desenvolvimento do projeto foram feitas diversas alterações do modelo inicial. Tais atualizações foram surgindo com uma melhor compreensão do funcionamento do banco de dados e infelizmente também pela limitação de conhecimento de programação.

Devido a não conseguir trabalhar com procedure foi alterado o melhor modelo encontrado, no anexo.
Durante o desenvolvimento foi criado uma função para inserção de dados do fornecedor em uma só ação, como tal entidade tem dependência das entidades PessoaJuridica, Endereco e Telefone, a execução só conseguia inserir a primeira tupla, não conseguindo inserir outras, o que provocou a alteração do modelo.
As etapas propostas foram todas atendidas, porém o projeto demonstrou ter uma dimensão muito grande para realizar que precisa ser desenvolvida para o sistema funcionar adequadamente no escritório de representação.
O entendimento da linguagem SQL foi desenvolvido mais facilmente com a busca de alcançar os objetivos propostos, ocorreu um maior aprofundamento nos comandos principalmente das funções. Porém ainda existe uma extensa necessidade de compreensão e utilização dos comandos SQL para alcançar o desenvolvimento completo do projeto aqui inicialmente proposto.
O método de desenvolvimento seguindo as regras do banco de dados auxiliou muito a todo processo, facilitando o entendimento e orientando a criação e expansão gradual do banco. Com essa metodologia torna-se mais fácil direcionar o desenvolvimento do banco.
A desvantagem observada foi a necessidade de sensibilidade para não ficar estacionado numa das etapas do sistema, nesse desenvolvimento foi na elaboração do modelo conceitual e lógico, despendeu-se muito tempo, tendo sido criados mais de quinze alterações.
Portanto a criação e desenvolvimento do projeto foi muito proveitoso e instrutivo, pois solidificou o conhecimento adquirido.

Scripts

Para o SQL Postgres não importa se é letra maiúscula ou minúscula, por isso em alguns momentos dos scripts existem as duas opções. O script não se encontra completo, os scripts completos estão nos arquivos anexos.

1. DDL

```
-- Database: "DBRepresentacao"

-- DROP DATABASE "DBRepresentacao";

CREATE DATABASE "DBRepresentacao"
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'pt_BR.UTF-8'
       LC_CTYPE = 'pt_BR.UTF-8'
       CONNECTION LIMIT = -1;
```

```
--Criando as tabelas

--Criando tabela Endereço
create table Endereco(
    idEndereco serial,
    rua varchar(50) not null,
    numero integer null,
    complemento varchar(15) null,
    bairro varchar(20) null,
    cidade varchar(15) not null,
    uf char(2) not null,
    cep varchar(8) null check (cep similar to '(\d)+'),
    primary key (idEndereco)
);

--Caso precise apagar a tabela
--drop table endereco;

--Alterando o tipo de dados da coluna numero da tabela endereco
alter table endereco alter numero type varchar(12);

--Criando tabela telefone
create table Telefone(
    idTelefone serial,
    codigo varchar(2) null check (codigo similar to '(\d)+'),
    numero varchar(9) not null check (numero similar to '(\d)+'),
    tipo varchar(10) not null check (tipo = 'fixo' or tipo = 'cel'),
    primary key (idTelefone)
);

--Caso precise apagar a tabela
--drop table telefone;

--Apagando a coluna tipo da tabela telefone
alter table telefone drop tipo;

--Criando uma constraint do tipo Check para o atributo tipo da tabela telefone
alter table telefone add tipo varchar(10) check (tipo = 'fixo' or tipo = 'cel');

--Criando tabela Fornecedor
create table Fornecedor(
    idFornecedor serial,
    cnpj varchar(14) not null check (cnpj similar to '(\d)+'),
    nome varchar(50) null,
    nomeFantasia varchar(30) null,
    idEndereco integer not null references Endereco,
```

```
    idTelefone integer not null references Telefone,
    Primary key(idFornecedor)
);

--Caso precise apagar a tabela
--drop table Fornecedor;

--Caso seja necessário apagar um atributo referenciado
alter table Fornecedor drop idtelefone cascade;

--Criando tabela Cliente
create table Cliente(
    idCliente serial,
    cnpj varchar(14) not null check (cnpj similar to '(\d)+'),
    nome varchar(50) null,
    nomeFantasia varchar(30) null,
    codigo integer,
    rede varchar(20),
    pagamentoDescarga boolean,
    idEndereco integer not null references Endereco,
    idTelefone integer not null references Telefone,
    Primary key(idCliente)
);

--Caso precise apagar a tabela
--drop table cliente;

--Acrescentando a tabela Fornecedor o atributo nomeFantasia
alter table Cliente add prazo varchar (12);

--Criando tabela Usuário
create table Usuario(
    idUsuario serial,
    cpf varchar not null check(cnpj similar to '(\d)+'),
    nome varchar(50) not null,
    cargo varchar (15) null,
    idEndereco integer not null references Endereco,
    idTelefone integer not null references Telefone,
    Primary key(idUsuario)
);

--Caso precise apagar a tabela
-- drop table usuario;

--Criando tabela Unidade
create table Unidade(
```

```
    idUnidade serial,
    idFornecedor integer not null references Fornecedor,
    idEndereco integer not null references Endereco,
    idTelefone integer not null references Telefone,
    Primary key(idUnidade)
);

--Caso precise apagar a tabela
-- drop table unidade;


--Criando tabela Produto
create table Produto(
    idProduto serial,
    descricao varchar(20),
    codigo integer,
    preco float,
    categoria varchar(12),

    tipo varchar(12),
    primary key(idProduto)
);

--Caso precise apagar a tabela
-- drop table produto;

--Criando tabela Pedido
create table Pedido(
    idPedido serial,
    idProduto integer not null references Produto (idProduto) on delete
cascade,
    idCliente integer not null references Cliente (idCliente) on delete cascade,
    idFornecedor integer not null references Fornecedor (idFornecedor) on
delete cascade,
    precoPedido float not null,
    quantidade float not null,
    prazo varchar(12) not null,
    requerReserva boolean,
    data date,
    primary key(idPedido)
);

--Caso precise apagar a tabela
-- drop table pedido;
```

2. DML

```
--Procedure para inserir dados nas tabelas



--Procedure para cadastrar produtos
create or replace function cadastrar_produto(descricaoP varchar, codigoP
integer,precoP float, categoriaP varchar, tipoP varchar) returns void AS $$
Begin
   insert into Produto(descricao, codigo, preco, categoria, tipo) values
(descricaoP, codigoP, precoP, categoriaP, tipoP);
End;
$$ Language 'plpgsql';

--Povoando a tabela de produtos
select cadastrar_produto('Ponta de Agulha', 123, 7.2, 'Resfriado', 'Peça');

select cadastrar_produto('Dianteiro', 375, 8.9, 'Resfriado', 'Peça');



--Procedure para cadastrar telefones
create or replace function cadastrar_telefone_s(codigoT varchar, numeroT
integer, tipoT varchar) returns void AS $$
Begin
   insert into telefone (codigo, numero, tipo) values (codigoT, numeroT,
tipoT);
End;
$$ Language 'plpgsql';

--Povoando tabela telefone
select cadastrar_telefone_s('84','987654321','Cel');

select cadastrar_telefone_s('84','982354321','Cel');



--Procedure para cadastrar endereços
create or replace function cadastrar_endereco_s(ruaE varchar, numeroE
integer,complementoE varchar, bairroE varchar, cidadeE varchar, ufE char,
cepE varchar) returns void AS $$
Begin
   insert into Endereco(rua, numero, complemento, bairro, cidade, uf, cep)
values (ruaE, numeroE, complementoE, bairroE, cidadeE, ufE, cepE);
End;
$$ Language 'plpgsql';

--Povoando tabela endereço
```

```
select cadastrar_endereco_s('Av. São José', 543,' ','Quintas','Natal',
'RN','59123000');

select cadastrar_endereco_s('Rua Estreita', 1020,'Lado sul ','Planalto','Natal',
'RN','59025000');


--Procedure para cadastrar fornecedor
create or replace function cadastrar_fornecedor_s(cnpjF varchar, nomeF
varchar, nomeFantasiaF varchar, idEnderecoF integer, idTelefoneF integer)
returns void AS $$
Begin
   insert into fornecedor(cnpj, nome, nomeFantasia, idEndereco, idTelefone)
values (cnpjF, nomeF, nomeFantasiaF, idEnderecoF, idTelefoneF);
End;
$$ Language 'plpgsql';

--Povoando tabela fornecedor
select cadastrar_fornecedor_s('01234567890123', 'Frigorifico', '', 12,13);

select cadastrar_fornecedor_s('34565437890123', 'Cereal', '', 13,14);

--Apagando dupla repetida
delete from fornecedor where idfornecedor = 4;

--Procedure para cadastrar usuário
create or replace function cadastrar_usuario(cpfU varchar, nomeU varchar,
cargoU varchar, idEndereco integer, idTelefone integer) returns void AS $$
Begin
   insert into usuario(cpf, nome, cargo, idEndereco, idTelefone) values (cpfU,
nomeU, cargoU, idEndereco, idTelefone);
End;
$$ Language 'plpgsql';

--Povoando tabela usuário
select cadastrar_usuario('01234567890','João Maria', 'vendedor', 18,16);

select cadastrar_usuario('98765432100', 'Paulo', 'gerente',19,17);

select cadastrar_usuario('65478392736', 'Maria', 'auxiliar',20,18);

--Procedure para cadastrar cliente
create or replace function cadastrar_cliente_s(cnpjC varchar, nomeC varchar,
nomeFantasiaC varchar, codigoC integer, redeC varchar,
pagamentoDescargaC boolean, prazoC varchar, idEnderecoC integer,
idTelefoneC integer) returns void AS $$
```

```
Begin
   insert into cliente(cnpj, nome, nomeFantasia, codigo, rede,
pagamentoDescarga, prazo, idEndereco, idTelefone) values (cnpjC, nomeC,
nomeFantasiaC, codigoC,  redeC, pagamentoDescargaC, prazoC,
idEnderecoC, idTelefoneC);
End;
$$ Language 'plpgsql';

--Povoando tabela cliente
select * from endereco;

select cadastrar_cliente_s('01020203040506', 'Almeida Souza
ME','Supermercado Quero mais', 324134,  ' ', true, '21/28/35', 2, 2);

select cadastrar_cliente_s('76543213040506', 'Supermercado Tanto Faz','',
546372,  ' ', true, '07', 9, 6);


--Procedure para cadastrar unidade
create or replace function cadastrar_unidade(idFornecedorU integer,
idEnderecoU integer, idTelefoneU integer) returns void AS $$
Begin
   insert into unidade(idFornecedor, idEndereco, idTelefone) values
(idFornecedorU, idEnderecoU, idTelefoneU);
End;
$$ Language 'plpgsql';

--Povoando tabela unidade
select cadastrar_unidade(2,22,15);

select cadastrar_unidade(2,23,16);

--Criando função testar_reserva para testar se existe necessidade de fazer a
reserva
create or replace function testar_reserva() returns trigger AS $testar_reserva$
Begin
   If NEW.requerReserva is true then
         Raise Exception 'Precisa ser feita uma reserva';
   End If;
   Return NEW;
End;
$testar_reserva$ Language 'plpgsql';

--Criando a trigger para inserção de pedido
create trigger pedido_insert before insert or update on pedido
for each row execute
```

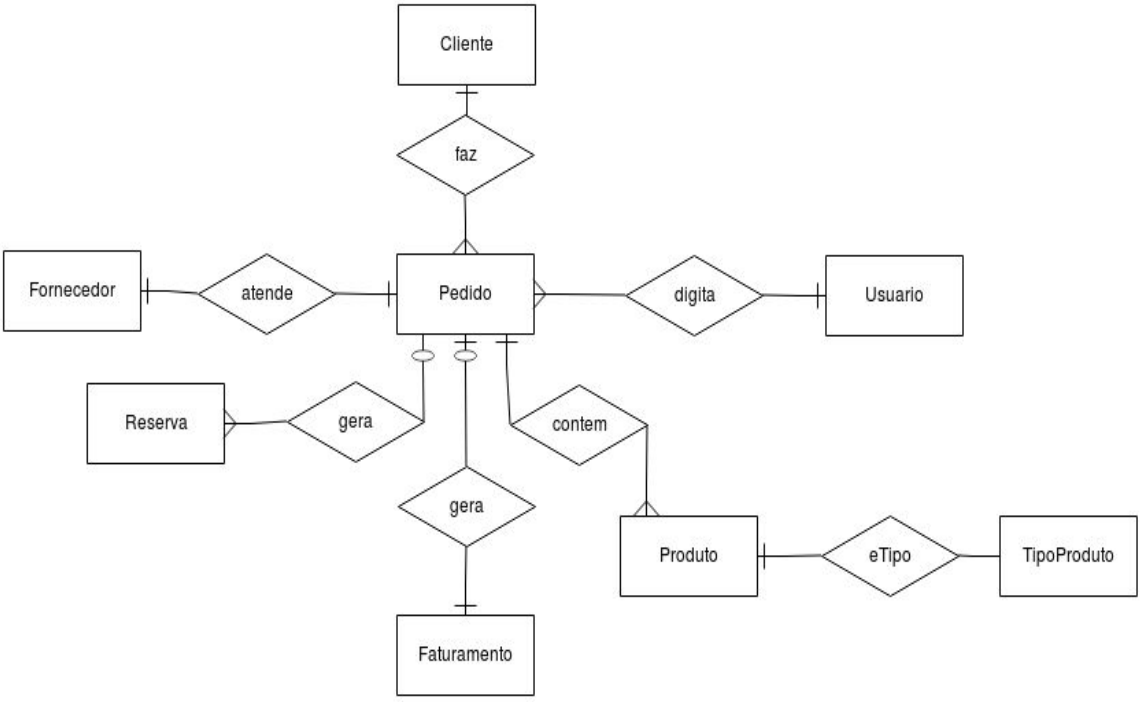
```
procedure testar_reserva();

--Povoando tabela pedido
insert into pedido (idProduto, idCliente, idFornecedor, precoPedido,
quantidade, prazo, requerReserva, data) values (1, 4, 2, 7.5, 20, '07', false,
'19/11/2017');

insert into pedido (idProduto, idCliente, idFornecedor, precoPedido,
quantidade, prazo, requerReserva, data) values (7, 4, 2, 17.5, 10, '21/28/35',
false, '19/11/2017');
```

3. DQL

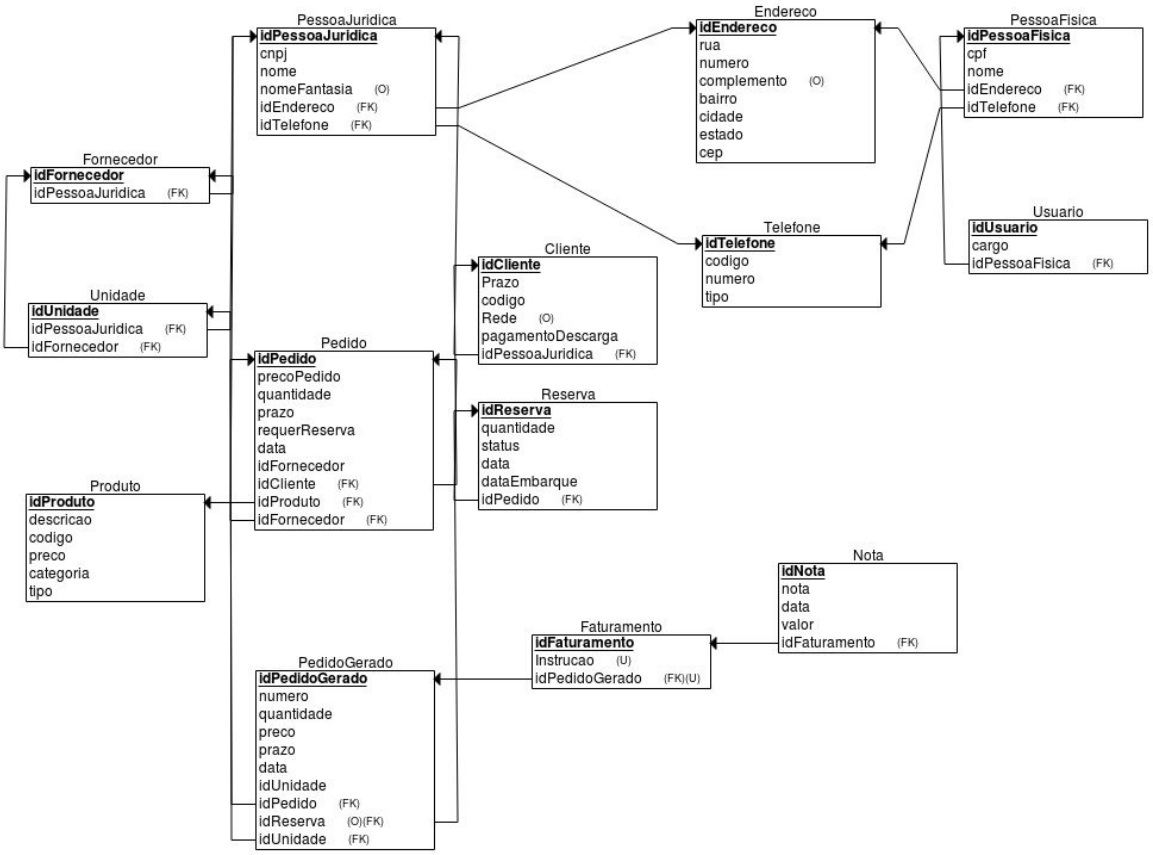
```
--Comando full join
select c.nome, p.quantidade from pedido p
full outer join cliente c on
c.idcliente = p.idcliente;

--Comando join
select u.nome, t.codigo,t.numero from usuario u join telefone t on u.idtelefone
= t.idtelefone;

--Criação de view com relação dos cliente de Natal
create view vw_relacaoClientesNatal as
select c.nome, c.nomeFantasia, e.bairro, t.codigo, t.numero
from cliente c
   join endereco e on c.idendereco = e.idendereco
   join telefone t on c.idtelefone = t.idtelefone
where e.cidade = 'Natal'
order by c.nome;

select * from vw_relacaoClientesNatal;

--Criação de view com os pedidos dos clientes
create view vw_pedidoCliente as
select c.nome, p.quantidade, pd.descricao
from pedido p
   join produto pd on p.idproduto = pd.idproduto
   join cliente c on c.idcliente = p.idcliente
order by c.nome;

select * from vw_pedidoCliente
```

Anexos

- Modelo conceitual inicial



- Modelo lógico aperfeiçoado



/

Referências

https://www.postgresql.org/docs/9.5

https://www.devmedia.com.br/guia/postgresql/34328

https://stackoverflow.com/questions/tagged/sql