Instituto Superior de Engenharia do Porto

Relatório
BDDAD

Turma: 2DK
José Mota 1161263
Patrick Timas 1171352
João Flores 1171409

2018/2019

➢ A IsepBricolage possui vários clientes, os quais são identificados por um código, um nome, uma morada, um código postal, um telemóvel, um número de contribuinte e uma zona geográfica. Estes são ainda classificados por tipo, de acordo com o volume de negócio que efetuam (por exemplo, VIP, grande cliente, pequeno cliente). Semanalmente, os clientes são visitados por vendedores da mesma zona geográfica, os quais registam os artigos encomendados numa nota de encomenda, que posteriormente enviam para o armazém da sua zona geográfica.

Em relação a adição dos clientes à base dados temos a tabela com os dados cod_cliente do tipo inteiro que é a chave primária para identificação do cliente único e não vazio porque não faria sentido ter repetidos, os dados nome e morada tipo Varchar pois é a variável mais adequada para este tipo ambas não podem estar vazias e o nome tem de ser único e o código postal, telemóvel e numero contribuinte sendo todas do integer para guardar os valores de cada atributo com tamanho adequado ao atributo, nenhum do anteriores podem ser vazios e o telemóvel e número contribuinte único.

```
create table Cliente(
cod_cliente Integer constraint pk_cod_cliente primary key,
id_zona_geografica Integer CONSTRAINT fk_id_zona_cliente REFERENCES Zona_Geografica(id_zona_geografica) on delete cascade,
nome varchar(40) constraint nm_cliente not null,
morada varchar (40) constraint morada_cliente not null,
cod_postal integer constraint cod_postal_cliente not null,
telefone Integer constraint telef_cliente not null check (telefone between 900000000 and 999999999),
num_contribuiente Integer check(num_contribuiente between 100000000 and 999999999)
);
```

Existe também uma tabela histórico tipo de cliente usada para guardar todos os tipos que cada cliente teve e datas de início e fim ambas não podem ser vazio e a data de fim tem de ser maior que a data início, tendo como chave cod_cliente que corresponde a uma chave primaria da tabela cliente sendo assim um modo de relacionar a tabela cliente à tabela histórico. Esta tabela usa uma ligação do tipo um para um pois um cliente tem apenas um histórico.

```
create table Historico_tipo_Cliente (
cod_cliente Integer constraint  pk_cod_cliente_historico primary key REFERENCES Cliente(cod_cLiente) on delete cascade,
tipo_cliente varchar(40) constraint nn_tipo_cliente not null Check (REGEXP_LIKE(tipo_cliente,'VIP||pequeno cliente||grande cliente|| medio cliente')),
data_inicio_tipo TIMESTAMP  constraint nn_data_inicio_tipo not null ,
data_fim_tipo TIMESTAMP  constraint nn_data_fim_tipo not null
);
```

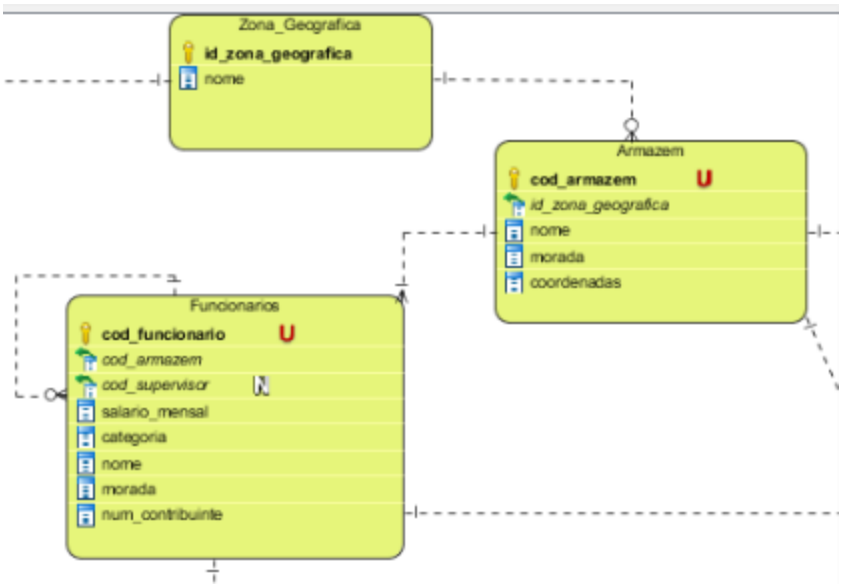
Como os clientes são visitados por vendedores da mesma zona geográfica logo os clientes têm um atributo id zona que é uma chave estrangeira para relacionar o cliente com uma zona geográfica, este atributo não pode ser nulo. Um cliente só pode ter uma zonas geográfica por isso a ligação para a zona geográfica só é para um.

```
create table Zona_Geografica (
id_zona_geografica Integer constraint pk_id_zonas_geografica primary key,
nome varchar(10) constraint nm_nome not null
);
```

Quando o cliente realiza uma ou mais encomendas o vendedor regista o que o cliente necessita, logo usamos uma ligação um para zero ou mais. A tabela nota de encomenda tem os atributos estado, quantidade e preço sendo eles todos do tipo integer que não podem ser nulo e em relação a quantidade e preço têm de ser positivos. A id_nota_encomenda representa a chave primaria para podermos relacionar esta tabela com as necessárias. Declara-se como chave estrangeiras o cod_funcionario para se saber que vendedor realizou a encomenda, usa-se o cod_cliente do mesmo modo que o funcionário de modo para saber que cliente realizou a encomenda, a referência é usada para obter o artigo e o número fatura para obter a fatura.

```
create table Nota_Encomenda(
id_nota_encomenda integer generated as identity constraint pk_id_nota_encomenda primary key,
cod_funcionario Integer constraint  fk_cod_funcionario_nota_Encomenda REFERENCES Funcionario(cod_Funcionario) on delete cascade,
cod_cliente Integer constraint  fk_cod_cliente_nota_Encomenda REFERENCES Cliente(cod_cliente) on delete cascade,
referencia Integer constraint fk_referencia_nota_encomenda References Artigo(referencia) on delete cascade,
num_fatura Integer constraint fk_num_fatura_nota_encomenda References Fatura(num_fatura) on delete cascade,
estado varchar (20) constraint estado_nota_encomenda not null,
quantidade integer constraint qnt_nota_enquantidade check(quantidade>=0),
preco integer constraint preco_nota_encomenda check(preco>=0)
);
```

➢ Na IsepBricolage trabalham vários funcionários, os quais são identificados por um número de funcionário (único). Para cada funcionário é registada a seguinte informação: o cartão de cidadão (único), o nome, a morada, o n.º de contribuinte, o salário mensal, a categoria a que pertence (por exemplo, vendedor, motorista, fiel de armazém, etc.) e o código do seu supervisor. Um supervisor é um funcionário da empresa e só pode ser supervisor de tiver a mesma categoria e idade superior ao seu supervisionado. Os funcionários da IsepBricolage pertencem a uma e só uma zona geográfica e estão afetos a um e só um armazém. No entanto, os motoristas apesar de pertencerem a uma só zona geográfica podem efetuar o transporte de mercadorias para as várias zonas geográficas desde que estas não distanciem da sua em mais de 200 kms.



Em relação à tabela funcionários existe uma chave primaria cod_funcionario que serve para identificar um certo funcionário, sendo um integer único. Contem o salário mensal integer que representa o salário para um determinado funcionário e respetiva categoria do tipo Varchar.

```
create table Funcionario (
cod_funcionario Integer constraint  pk_cod_funcionario primary key,
cod_armazem Integer constraint fk_cod_armazem_funcionario REFERENCES Armazem(cod_armazem) on delete cascade,
cod_supervisor Integer DEFAULT null,
salario_mensal integer constraint  nn_salario_mensal not null Check (salario_mensal>0),
categoria varchar(40) constraint nn_categoria not null Check (REGEXP_LIKE(categoria,'motorista||fiel armazem||vendedor|| supervisor'))
nome varchar(40) constraint  nn_nome_funcionario not null,
morada varchar(40) constraint nn_morada_funcionario not null,
num_contribuinte Integer constraint nn_num_contribuinte not null
);
```

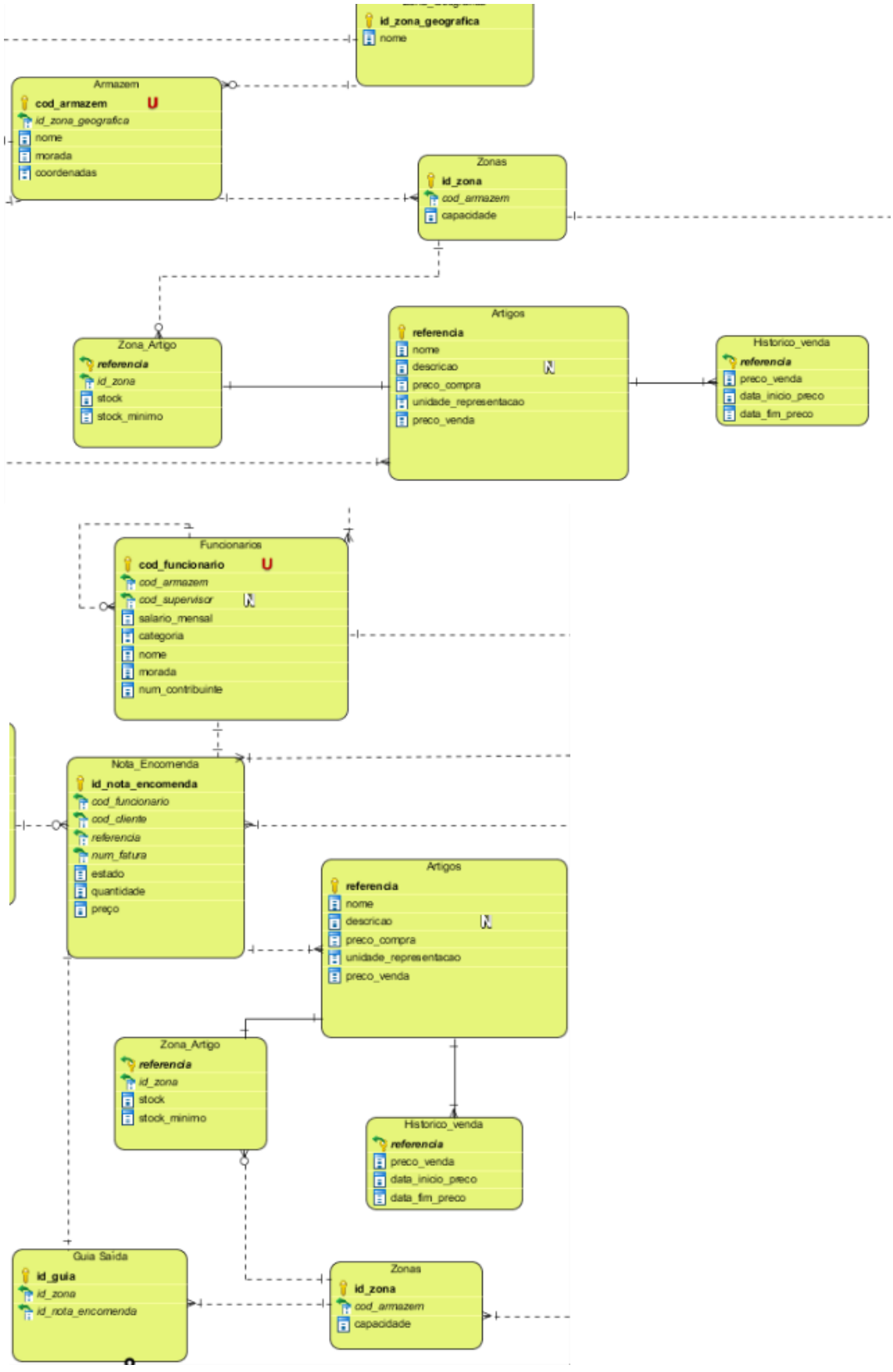
Tem uma relação a si próprio para se poder saber quem são os supervisores, sendo que só se pode ser supervisor caso tenha a mesma categoria e idade superior ao seu supervisionado neste caso isto é uma restrição de integridade. A ligação é de um para muitos porque o um supervisor pode ter vários supervisionados.

Um cod_supervisor, este código serve para identificar o supervisor do funcionário respeitando a restrição de integridade anterior.

Um cod_armazem para relacionar um funcionário a um armazém daí a usarmos relação de um para muitos porque um armazém pode ter vários funcionários, mas um funcionário só pode trabalhar num armazém.

Caso o funcionário seja motorista pode realizar viagens, na qual tem o cod_funcionario para se saber quem realizou uma certa viagem. Um motorista só pode realizar uma viagem caso a zona geográfica do seu armazém se encontre num raio de duzentos kms aos locais da viagem. Sendo esta mais uma restrição de integridade.

➢ A IsepBricolage está sediada na zona norte e possui vários armazéns instalados em várias zonas geográficas. Os armazéns da IsepBricolage são identificados por um código (único), um nome, uma morada e uma localização em coordenadas geográficas WGS84. Cada armazém está dividido em várias zonas físicas com uma dada capacidade (volume total). Os artigos podem estar armazenados em várias zonas físicas em diferentes quantidades e cada zona física pode ter vários artigos. Um artigo tem uma referência única, um nome, uma descrição, um preço de compra, um preço de venda, uma unidade de representação (por exemplo: m, kg, caixa, unidades), uma quantidade stock mínimo e uma quantidade em stock. A quantidade de stock mínimo e a quantidade de stock varia de armazém para armazém. Os preços de venda dos artigos são periodicamente alterados e por isso interessa manter o histórico dos preços e do intervalo de datas em que o preço se encontrava válido.


id_zona_geografica
nome
Armazem
cod_armazem
U
id_zona_geografica
nome
morada
coordenadas
Zonas
id_zona
cod_armazem
capacidade
Zona_Artigo
referencia
id_zona
stock
stock_minimo
Artigos
referencia
nome
descricao
preco_compra
unidade_representacao
preco_venda
Historico_venda
referencia
preco_venda
data_inicio_preco
data_fim_preco
Funcionarios
cod_funcionario
U
cod_armazem
cod_supervisor
salario_mensal
categoria
nome
morada
num_contribuinte
Nota_Encomenda
id_nota_encomenda
cod_funcionario
cod_cliente
referencia
num_fatura
estado
quantidade
preço
Artigos
referencia
nome
descricao
preco_compra
unidade_representacao
preco_venda
Zona_Artigo
referencia
id_zona
stock
stock_minimo
Historico_venda
referencia
preco_venda
data_inicio_preco
data_fim_preco
Guia Saída
id_guia
id_zona
id_nota_encomenda
Zonas
id_zona
cod_armazem
capacidade

A Isep bricolage contem vários armazéns, esta tabela contem a chave primaria cod_armazem para relacionar este armazém com outras tabelas. Um armazém está sediado numa zona geográfica por isso o armazém contem um id_zona, com uma ligação um para um ou mais porque uma zona geográfica pode conter vários armazéns, mas um armazém não pode ter mais do que uma zona.

```
create table Armazem(
cod_armazem Integer generated as identity constraint pk_cod_armazem primary key,
id_zona_geografica Integer CONSTRAINT fk_id_zona REFERENCES Zona_Geografica(id_zona_geografica) on delete cascade,
nome varchar(20) constraint nn_nome not null,
morada varchar(20) constraint nn_morada not null,
coordenadas varchar(20) constraint nn_coordenadas  not null
);
```

Os atributos nome, morada e coordenadas são Varchar não podendo serem nulas e repetidas.

Os armazéns estão relacionados com a tabela funcionários um para muitos, porque um armazém pode ter varios funcionarios.

Um armazem relaciona-se com as zonas físicas a partir do cod_armazem na tabela zonas, sendo este cod_armazem diferente de nulo pelo facto de uma zona estar contida num armazém.

```
create table Zonas (
id_zonas Integer constraint pk_id_zonas primary key,
cod_armazem Integer CONSTRAINT fk_cod_armazem REFERENCES Armazem(cod_armazem) on delete cascade,
capacidade Integer constraint nn_capacidade not null
);
```

A tabela zona contem uma chave primaria com o nome id_zona para identificar a zona e uma capacidade do tipo integer para guardar a capacidade da respetiva zona.

Com a necessidade de guardar o stock mínimo e stock criou-se a tabela zona artigo com chaves id_zona referente a zona física, criada a cima, com dois atributos que integer não nulos e superiores a zero para guardar o stock. Uma chave necessária é referencia para podermos saber qual o stock de um determinado artigo com o nome referencia.

Um requisito necessário é que o stock nunca fica inferior ao stock mínimo caso fique a encomenda fica como pendente.

Em relação a tabela uma referência como chave primaria para distinguir os artigos. Os outros atributos dos artigos é o nome do tipo Varchar não podendo este ser nulo, a descrição também é do tipo Varchar podendo esta ser nula, uma unidade de

representação do tipo Varchar não podendo este ser nulo e ser do tipo de medidas portuguesas. Por fim tem os preços de compra de venda sendo este do tipo numeric para poder guardar os valores com casas decimais e com as restrições de não poder ser nulo e serem positivos.

```
create table Zona_Artigo(
referencia Integer constraint  pk_referencia_Zona_Artigo primary key REFERENCES Artigo(referencia) on delete cascade,
id_zonas Integer constraint  pk_id_zona_artigo  REFERENCES Zonas(id_zonas) on delete cascade,
stock_minimo integer constraint   nn_stock_minimo not null,
stock integer constraint stock not null
);
```

Um aspeto importante em relação ao preço de venda seria guardar o histórico do preço para cada artigo. Nesta tabela é necessária uma chave para saber qual o produto referente ao artigo e os atributos que guardam o preço de venda, data início preço e data fim preço. O preço venda guarda o preço para o produto numa determinada data do tipo numeric não podendo ser nulo e tem de ser superior a zero. Em relação as duas datas ambas têm de ser diferentes de nulas. A data início necessita de ser superior a data fim.

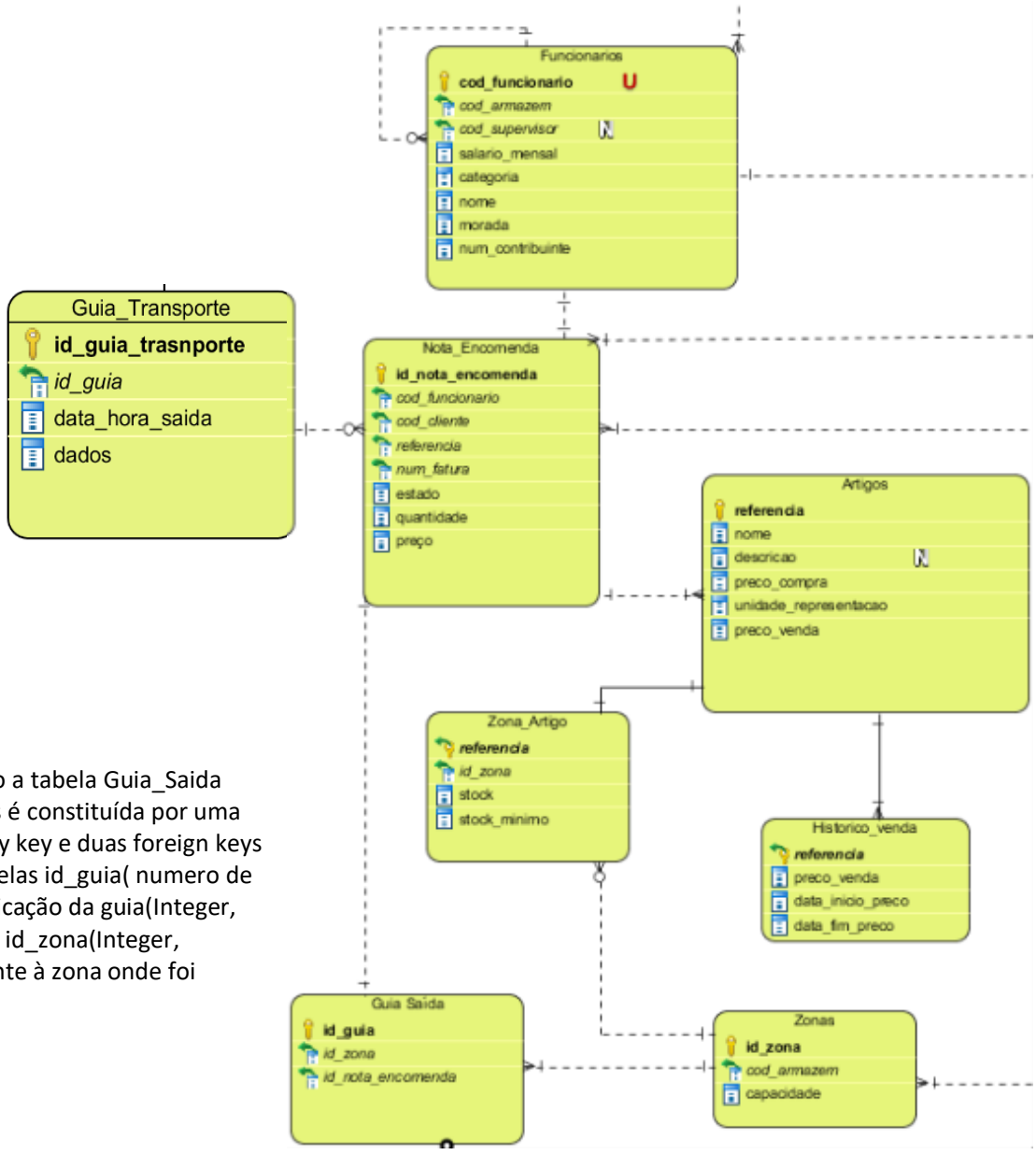
```
create table Historico_Venda(
referencia Integer constraint  pk_referencia_historico primary key REFERENCES Artigo(referencia) on delete cascade,
id_zonas Integer constraint fk_id_zona_Historico REFERENCES Zonas(id_zonas) on delete cascade,
preco_venda integer constraint nn_preco_venda_Historico not null check (preco_venda>0),
data_inicio_preco TIMESTAMP  constraint nn_data_inicio_preco not null ,
data_fim_preco TIMESTAMP  constraint nn_data_fim_preco not null

);
```

Um aspeto importante nas tabelas é a utlização do on-delet-cascade que caso uma tabela seja eliminada a base de dados procura todos que tivessem esse elemento e elimina-os também.

Quando as notas de encomendas chegam ao armazém, um dos funcionários do armazém analisa a existência de artigos em stock. Se algum artigo, de uma dada encomenda, atingiu o stock mínimo a nota de encomenda é colocada como pendente. Caso contrário, ela será processada e dará origem a uma guia de saída de artigos. A guia de saída de artigos tem um identificador gerado automaticamente pelo SGBD e é referente a um conjunto de artigos. Para cada artigo da guia de saída interessa saber em que zona(s) foi retirado e qual as respetivas quantidades. Logo que as encomendas estejam processadas, os artigos são enviados para a secção de expedição juntamente com os dados dos clientes e a lista de artigos adquiridos. Desta secção são enviados os artigos para os clientes acompanhados da respetiva guia de transporte.

Para este ponto, temos de verificar se o artigo se encontra disponível. Caso não se verifique isso, o estado da encomenda passa para pendente (atributo estado da tabela Nota_Encomenda), caso contrário a encomenda é processada. Com isto são criadas as seguintes tabelas: Guia_Saida, Guia_Transporte.



Quanto a tabela Guia_Saida apenas é constituída por uma primary key e duas foreign keys snedo elas id_guia(numero de identificação da guia(Integer, único), id_zona(Integer, referente à zona onde foi

retirado) e o id_nota_encomenda(Integer,referente à encomenda em questão),respetivamente.

```
create table Guia_Saida(
id_guia Integer generated as identity constraint pk_id_guia primary key,
id_zonas Integer CONSTRAINT fk_id_zona_guia_saida REFERENCES Zonas(id_zonas) on delete cascade,
id_nota_encomenda Integer CONSTRAINT fk_id_nota_incomneda REFERENCES Nota_Encomenda(id_nota_encomenda) on delete cascade

);
```

Quanto a tablea Guia_Transporte, esta é constituída por uma primary key, id_guia_transporte, sendo esta uma identificador da guia de transporte. Como foreign key temos o id_guia, ou seja temos a guia de saída associada. Com esta chave estrangeira temos acesso aos dados presentes na nota de encomenda, como os dados relativos ao armazém, ao cliente e do funcionário que está encarregue da entrega, quem como os artigos. Como atributos desta tabela temos a data_hora_saida (TIMESTAMP para assim obtermos a data e a hora) e dados_empresa(varchar, sendo estes os dados relativos a empresa).

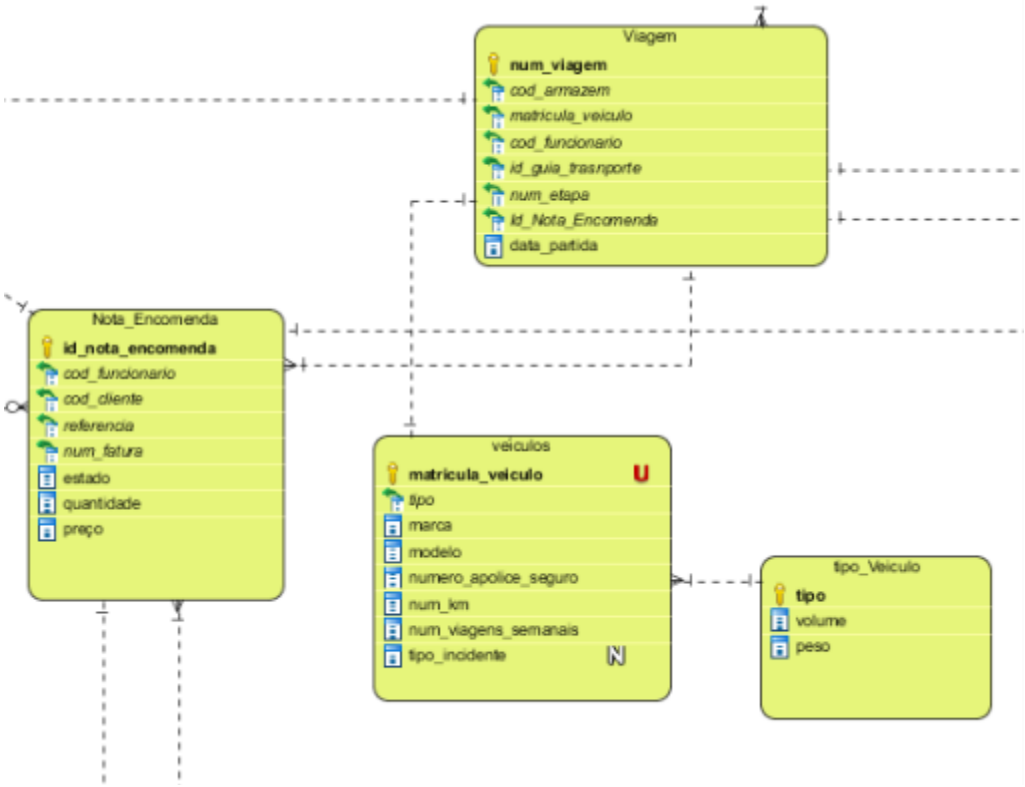
```
create table Guia_Transporte(
id_guia_transporte Integer constraint  pk_id_guia_transporte primary key,
id_guia  Integer constraint fk_id_guia REFERENCES Guia_Saida(id_guia) on delete cascade,
data_hora_saida Timestamp constraint nn_data_hora_saida not null,
dados_empresa varchar(40) constraint nn_dados_empresa not null
);
```

Nesta seção temos como restrições de integridade :

- Se o stock for nulo, a encomenda não será processada e será considerada como pendente

O transporte de mercadorias é feito pela própria IsepBricolage. IsepBicolage possui uma frota de veículos que são de um determinado tipo (2 eixos, 3 eixos, 4 eixos, etc.), têm um número de matrícula, uma marca, um modelo, um número da apólice de seguro e um determinado número de quilómetros. Os tipos de veículos têm diferentes capacidades, quer relativas ao volume ou ao peso que podem transportar. Cada veículo só pode fazer três a quatro viagens por semana. Um veículo pode transportar várias encomendas numa mesma viagem e entregar as encomendas a diferentes clientes. Numa viagem a mercadoria deverá ocupar toda ou quase toda a capacidade do veículo, isto porque a IsepBricolage pretende minimizar os custos inerentes ao transporte de mercadorias.

Cada viagem inicia-se com um carregamento num armazém e é identificada por um número de viagem e data de partida. Cada viagem está associada a um veículo e é composta de etapas, de um cliente a outro cliente. Cada etapa é identificada por um número de etapa.



Nesta etapa a empresa prepara-se para a viagem de entrega das encomendas. Para tal iremos necessitar da tabela Viagem. Esta tabela é constituída por uma primary key ,num_viagem(integer,refere-se ao numero da viagem), e por varias foreign keys: cod_armazem(chave referente ao armazém de onde é expedida a encomenda), a matricula do veículo que a vai transportar (matricula_veiculo), o motorista,funcionário que a transporta(código funcionário), a guia de transporte(através do id_guia_transporte), o número da etapa correspondentes a entrega da encomenda,(num_etapa), e os dados referentes à encomenda(através do id_nota_encomenda, conseguimos obter os dados da encomenda).

```
create table Viagem(
num_viagem integer constraint pk_num_viagem primary key,
cod_armazem Integer CONSTRAINT fk_cod_armazem_viagem REFERENCES Armazem(cod_armazem) on delete cascade,
matricula_veiculo varchar(15) constraint fk_matricula_veiculo_viagem references Veiculos(matricula_veiculo) on delete cascade,
cod_funcionario Integer constraint  fk_cod_funcionario_nota_viagem REFERENCES Funcionario(cod_Funcionario) on delete cascade,
id_guia_transporte Integer constraint  id_guia_transporte_viagem REFERENCES Guia_Transporte(id_guia_transporte) on delete cascade,
num_etapas Integer constraint  num_etapas_viagem REFERENCES Etapas(num_etapas) on delete cascade,
id_nota_encomenda integer constraint id_nota_encomenda_viagem References Nota_Encomenda(id_nota_encomenda) on delete cascade,
data_partida TimeStamp CONSTRAINT dt_ptd_viagem NOT NULL
);
```

Para a mesma ser transportada, é necessário um veículo. A tabela veículo e composta por uma chave primaria, que é a matrícula do veículo visto que esta é única. Como foreign key temos o tipo de veículo. Como atributos temos a marca do veiculo(varchar), o modelo(varchar), o numero_apolice_seguro(integer),numero_km feitos pelo veículo, num_viagens_semanais realizadas pelo mesmo e tipo_incidente.

```
create table Veiculos(
matricula_veiculo varchar(15) constraint pk_matricula_veicula primary key CHECK (REGEXP_LIKE(matricula_veiculo, '[0-9][2]-[A-Z][2]-[0-9][2]|[0-9][2]-[0-9][2]-[A-Z][2]|[A-Z][2]-[0-9][2]-[0-9][2]')),
tipo varchar(20) constraint fk_tipo_veiculo references Tipo_veiculo(tipo) on delete cascade,
marca varchar(20) constraint marca_veiculo not null,
modelo varchar(20) constraint modelo_veiculo not null,
numero_apolice_seguro integer constraint nm_ap_sg_veiculo not null,
num_km integer constraint nm_km_veiculo not null,
num_viagem_semanais integer constraint nm_vgm_smn_veiculo not null check(num_viagem_semanais<=4),
tipo_incidente varchar(40)
);
```

O tipo de veiculo e caracterizado pelo peso (integer) e pelo volume(integer), sendo como primary key o tipo, sendo o tipo o numero de eixos

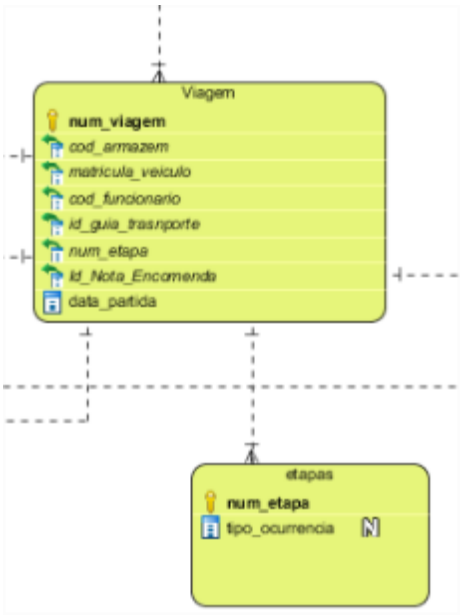
```
create table Tipo_veiculo(
tipo varchar(20) constraint pk_tipo primary key,
volume integer constraint volume_tipo_veiculo not null,
peso integer constraint peso_tipo_veiculo not null
);
```

Como restrições de integridade neste ponto temos:

- Cada veículo só pode fazer no máximo 3 a 4 viagens semanais,
- O veículo só sai para entregar encomendas só e só se este ocupar toda ou quase toda a capacidade

Sempre que um funcionário (condutor) termina as entregas faz uma notificação de fim de serviço e informa se ocorreu algum incidente (por exemplo: greve, acidente de aviação, cliente não estava, etc.). Em caso de incidente o condutor regista o tipo de ocorrência e elabora uma pequena descrição do incidente. Caso o tipo de incidente seja um acidente de aviação o veículo fica indisponível para uma nova entrega de mercadoria.

No fim de cada viagem, é registada na tabela etapa, a ocorrência de algum incidente. A disponibilidade do veículo é anotada no atributo etapa.

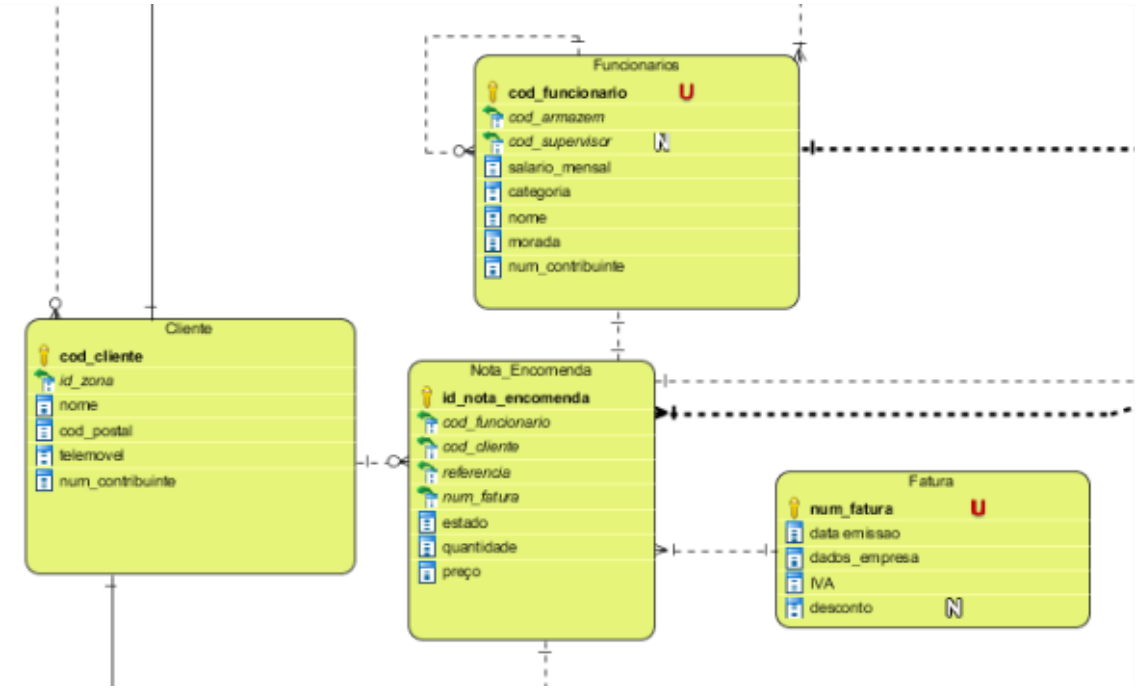



```
create table Etapas(
num_etapas integer constraint pk_num_etapas primary key,
tipo_ocurrencia varchar(50)
);
```

Para este ponto, temos como restrição de integridade:

- Caso o incidente seja um acidente de viação, o veículo fica indisponível para viagens de entrega futuras.

Ao dia 15 de cada mês, todas as encomendas expedidas são faturadas ao cliente, numa única fatura. A fatura emitida deverá indicar no cabeçalho, para além da informação da IsepBricolage, os dados do cliente, o código do vendedor, a data de emissão, e o número da fatura. A fatura deve indicar a lista de artigos adquiridos, com as respectivas quantidades, valor de base a pagar, valor do IVA e descontos.



Como todas as encomendas tem que ter um comprovativo, e enviado junto com a mesma uma fatura. Para registar tudo criamos a tabela Fatura. Esta e constituída pelo um identificador(num_fatura),pela data de emissão da mesma(data_emissao), pelo valor do IVA(IVA) e pelo desconto aplicado (desconto). Através da tabela Nota_Encomenda conseguimos obter os dados relativos à aos artigos, nomeadamente a quantidade, preco_venda, referencia do mesmo, dados do cliente.

```
create table Fatura(
num_fatura Integer  generated as identity constraint pk_num_fatura primary key,
dados_empresa varchar(100) constraint dados_empresa_fatura not null,
data_emissao TimeStamp CONSTRAINT nn_fatura NOT NULL,
iva integer constraint iva_fatura check(iva>0),
desconto integer constraint desconto_fatura check(desconto>=0)
);
```

Para um melhor controlo financeiro a IsepBricolage pretende que seja registada na base de dados todos os pagamentos efetuados pelos clientes.

Para este passo criamos um registo de pagamentos para todos os clientes da empresa. Assim criamos a tabela Pagamentos que contem como primary key o cod_cliente e como atributos tem atributo o valor_pago(integer) e a data_pagamento(TimeStamo para assim sabermos a data e a hora de pagamento).

```
create table Pagamentos(
cod_cliente Integer constraint  pk_cod_cliente_pagamentos primary key not null REFERENCES Cliente(cod_cliente),
valor_pago Integer constraint valor_pago_cliente not null check(valor_pago>=0),
data_pagamento TimeStamp CONSTRAINT dt_pagamento NOT NULL
);
```