Identificando as entidades e atributos, que representam os dados necessários:

A instituição possui uma vasta gama de recursos, incluindo livros, artigos, periódicos e materiais multimídia. Cada recurso é identificado por um código único, contendo informações sobre título, autor, editora, ano de publicação e palavras-chave relevantes. Além disso, é importante rastrear o número de cópias disponíveis de cada recurso, bem como seu status de disponibilidade.

Recursos

- Código Recursos (ID)
- Título
- Descrição
- Tipo (livro,artigos,periódicos,materiais múltiplos)
- Autor
- Editora
- Ano de publicação
- Palavra-chave
- Número de cópias disponíveis
- Status (Disponível, Indisponível)

Estudantes e professores são usuários essenciais do sistema. Cada usuário é registrado com um número de identificação pessoal e fornece informações como nome, endereço de e-mail e filiação departamental.

Usuários

- Código Usuario (ID)
- Tipo (estudante, Professor)
- Nome
- Email
- Filiação departamental.

O sistema também precisa lidar com empréstimos. Os usuários podem solicitar empréstimos de recursos da biblioteca, indicando a data de retirada. Cada empréstimo é registrado com um número de identificação do empréstimo e deve rastrear a data de retirada, a data de devolução esperada e, posteriormente, a data de devolução real. Atrasos nas devoluções podem resultar em penalidades.

Empréstimos

- Código Empréstimo (ID)
- Código Usuario (ID)
- Código Recursos (ID)
- Data de retirada
- Previsão de devolução
- Data de devolução
- Atrasos

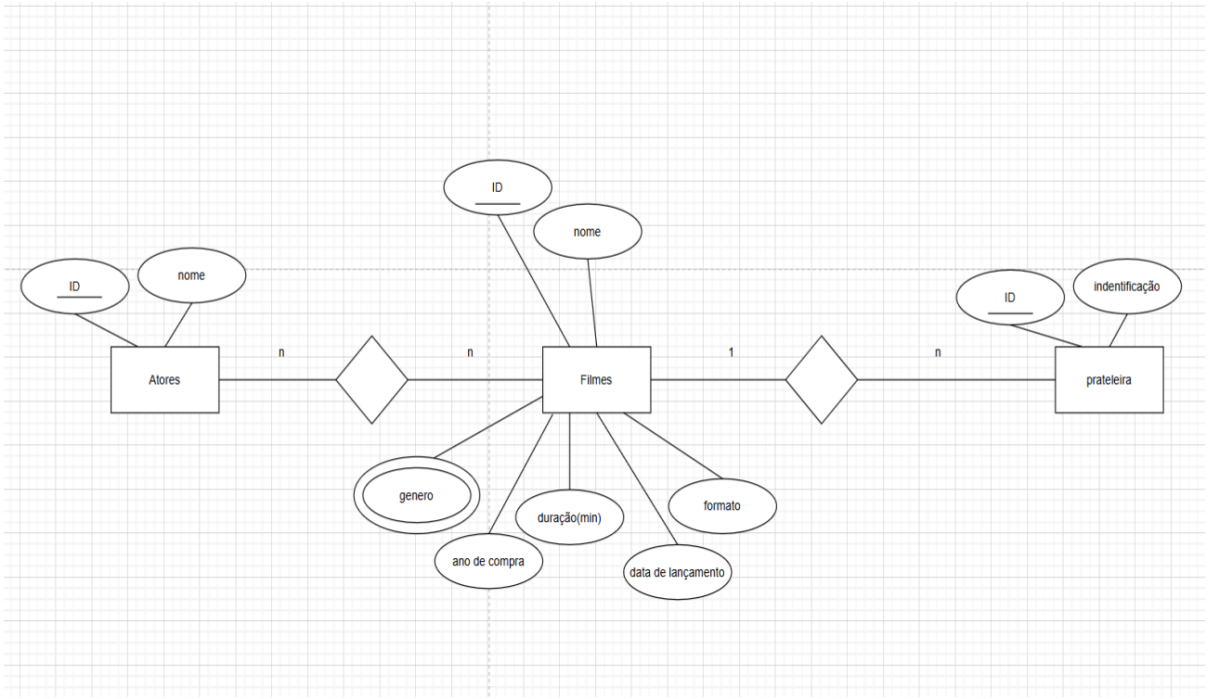
Além disso, há uma equipe de funcionários encarregada de administrar o sistema. Cada funcionário é identificado por um número de funcionário e possui detalhes sobre seu cargo e informações de contato.

adm_system

- Código funcionário (ID)
- Cargo
- Contato

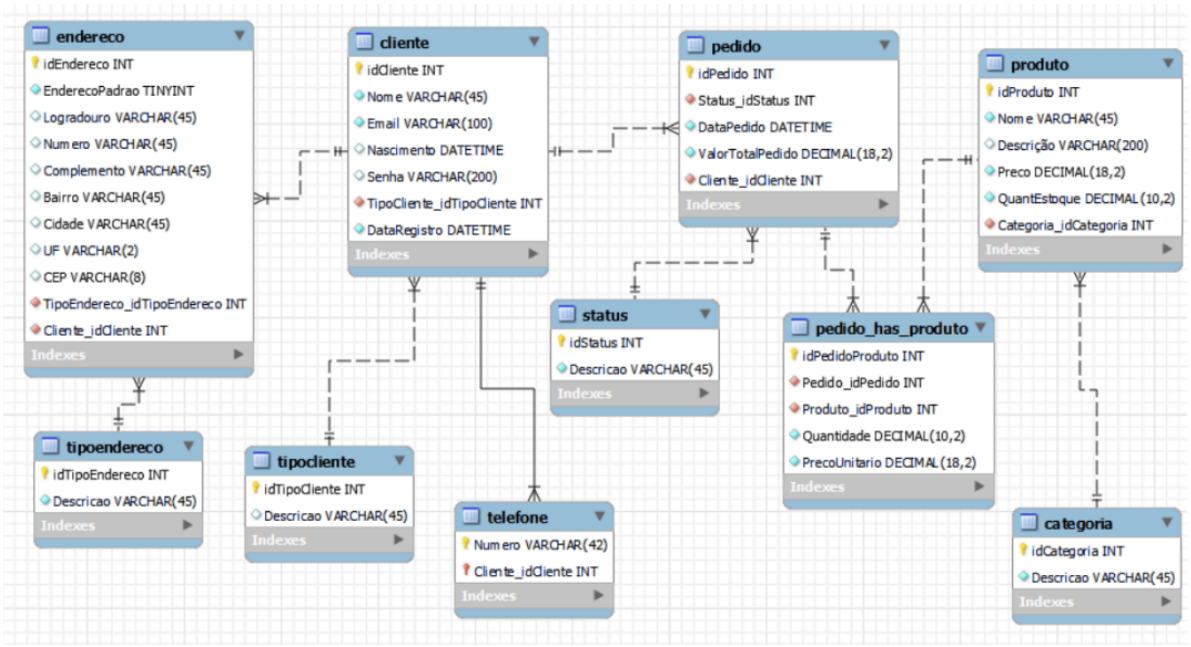
Construindo um Diagrama de Entidade e Relacionamento –DER, que represente os dados necessários a este controle, juntamente com as Integridades:

Carlos é um amante de filmes. Ele gosta tanto de filmes que embora assine vários serviços de Stream, continua comprando os DVDs ou BlueRays dos filmes que ele mais gosta. Fato que isso gerou um problema de armazenamento e organização. Por várias vezes ele sabe que tem um filme mas não tem a mínima ideia onde ele pode estar. Já aconteceu inclusive situações em que ele comprou um filme mais de uma vez simplesmente porque não lembrava que já havia comprado o filme. Neste cenário complexo ele decidiu criar um sistema para armazenar seus filmes. Para cada filme ele pretende armazenar o Nome, formato (DVD, BlueRay, etc) data de lançamento, tempo de duração em minutos, gênero (ação, drama, ficção científica, etc.) e o ano que ele comprou o filme. Um filme pode ter mais de um gênero. Podemos ter uma comédia romântica ou um filme de ação que também é um drama. Os atores também terão um destaque pois existem atores que participam de diversos filmes e um filme tem diversos atores. Outra informação que precisa ser armazenada é o local onde o filme foi guardado. Assim, precisamos saber qual Prateleira é usada para armazenar o filme. Essa prateleira deve ter uma identificação (prateleira1, Prateleira 2, etc.) Desta forma um filme estará posicionado em uma prateleira e uma prateleira teremos diversos filmes.



DML Contextualização

Carlos precisa de um novo emprego. Ele está estressado e cansado de um trabalho repetitivo e sem perspectivas. Fato que ele decide largar o emprego e montar uma lojinha na internet. Para isso precisa de um sistema que permita a ele cadastrar os cliente, registrar seus pedidos, cadastrar produtos, registrar seus preços de venda, etc. Nesta ideia, o Carlos desenvolveu o modelo de entidade e relacionamento que segue.



Os scripts de criação deste banco estão disponíveis no AVA no bloco "Linguagem de Manipulação de Dados - DML - SQL", indicado pelo rótulo "Banco Exemplo – Vendas"

Utilizando o banco de dados definido no modelo de entidade e relacionamento acima, escreva as consultas SQL solicitadas em cada uma das questões.
Construir Cláusulas SQL para:

1. Listar todos os clientes que fazem aniversário no mês junho;

```
select Nome from Cliente where Month(Nascimento) = 4;
```

2. Listar quantos dias de vida cada cliente tem no dia de hoje;

```
select datediff(NOW(), Nascimento) 'Dias de diferenças' from
Cliente;
```

3. Criar uma consulta SQL que liste todos os clientes, seus pedidos e no pedido indique a descrição do status do pedido;

```
select c.Nome, p.idPedido, s.Descricao  from Cliente c
join Pedido p
on c.idCliente = p.Cliente_idCliente
join Status s
on p.Status_idStatus = s.idStatus;
```

4. Construir uma consulta SQL que liste o pedido, o produto, a quantidade de produtos do pedido, o valor unitário do produto e o valor total (quantidade x valor unitário).

```
select p.idPedido, po.Nome, pp.Quantidade,
pp.PrecoUnitario, (pp.Quantidade * pp.PrecoUnitario) as
valor_total
from Cliente c
join Pedido p
on c.idCliente = p.Cliente_idCliente
join Status s
on p.Status_idStatus = s.idStatus
join Pedido_has_Produto pp
on p.idPedido = pp.Pedido_idPedido
join Produto po
on pp.Produto_idProduto = po.idProduto
;
```

5. Gerar uma relação com as quantidades e valores totais dos pedidos agrupados pela descrição do status;

```
SELECT s.Descricao, COUNT(p.idPedido) Quantidade,
SUM(p.ValorTotalPedido) "Valor total"
FROM Pedido p
JOIN Status s ON s.idStatus=p.Status_idStatus
GROUP BY s.Descricao;
```

6. Listar todos os clientes e seus endereços. Para cada endereço, mostrar de forma descritiva qual o tipo do endereço;

```
SELECT c.idCliente,
      c.Nome,
      e.UF,
      e.Cidade,
      e.Bairro,
      e.CEP,
      e.Logradouro,
      e.Complemento,
      e.Numero,
      IF(e.EnderecoPadrao, 'É o endereço padrão', 'Não é o
endereço padrão') "Endereço Padrão?",
      t.Descricao
"Tipo de endereço"
FROM Cliente c
JOIN Endereco e ON c.idCliente = e.idEndereco
JOIN TipoEndereco t on t.idTipoEndereco =
e.TipoEndereco_idTipoEndereco;
```

7. Criar uma listagem com a contagem de todos os produtos, soma do preço x quatestoque, agrupados pela descrição da categoria;

```
select c.Descricao,
      count(p.idProduto) as Quantidade,
      sum(p.Preco * p.QuantEstoque) as Total
from Produto p
join Categoria c on p.Categoria_idCategoria = c.idCategoria
group by c.Descricao;
```

8. Criar um script SQL para listar todos os clientes que moram na Cidade de São Paulo;

```
select c.Nome,
      e.Cidade
from Cliente c
join Endereco e on c.idCliente = e.Cliente_idCliente
where e.Cidade = 'São Paulo';
```

9. Criar um relatório mostrando o nome do cliente e os produtos que ele comprou em quaisquer pedidos com status fechado;

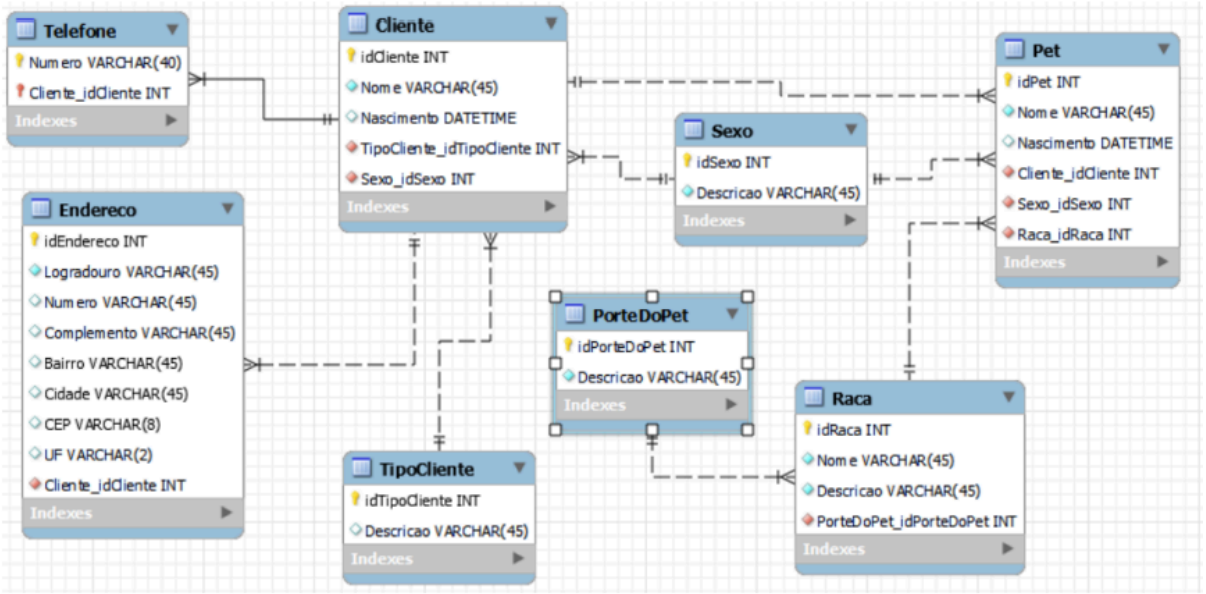
```
select cli.nome as 'Nome do Cliente', prod.nome as 'Produto',
st.Descricao as 'Situação do Pedido'
from Cliente cli
join Pedido pe on pe.cliente_idCliente = cli.idCliente
join Status st on st.idStatus = pe.status_idStatus
join Pedido_has_Produto php on php.pedido_idPedido = pe.idPedido
join Produto prod on prod.idProduto = php.Produto_idProduto
where st.Descricao = 'Fechado'
ORDER BY cli.Nome;
```

10. Mostrar mês a mês quantos (quantidade) pedidos foram feitos.

```
select month(DataPedido) as 'Mês', year(DataPedido) as 'Ano',
count(*) as 'Quantidade de Pedidos'
from Pedido
group by year(DataPedido), month(DataPedido)
order by Ano, Mês;
```

Construir os Scripts de criação do Banco de dados mostrado no MER - Modelo de Entidade e Relacionamento e os scripts de inserção de dados nas tabelas.

Requisitos do trabalho:
1. Sintaxe do MySql
2. Criar as FKs em Script separado da criação das tabelas (alter table)
3. Criar os scripts de inserção dos dados nas tabelas usando o comando Insert Into.




```
-- Criação do banco de dados
create database taismoreira;

-- criação da tabela (create table)
use taismoreira;

create table Cliente(
idCliente int auto_increment,
nome varchar(45) not null,
nascimento datetime,
primary key(idCliente));

create table Telefone(
numero varchar(15),
primary key(numero));

create table Endereco(
idEndereco int auto_increment,
logradouro varchar(45) not null,
```

```
numero varchar(6) not null,
complemento varchar(45),
bairro varchar(45) not null,
cidade varchar(45) not null,
cep varchar(8),
uf varchar(2),
primary key(idEndereco));

create table TipoCliente(
idTipoCliente int auto_increment,
descricao varchar(45),
primary key(idTipoCliente));

create table Pet(
idPet int auto_increment,
nome varchar(45) not null,
nascimento datetime not null,
primary key(idPet));

create table PorteDoPet(
idPorteDoPet int auto_increment,
descricao varchar(45) not null,
primary key(idPorteDoPet));

create table Raca(
idRaca int auto_increment,
nome varchar(45) not null,
descricao varchar(45),
primary key(idRaca));

create table Sexo(
idSexo int auto_increment,
descricao varchar(45),
primary key(idSexo));

-- Alterar as tabelas(Adicionando Foreign Key)
-- telefone(n) : cliente(1)
alter table Telefone
add column cliente_idCliente int not null,
add constraint fk_telefone_cliente
foreign key (cliente_idCliente)
references Cliente(idCliente);

-- endereco(n) : cliente(1)
alter table Endereco
add column cliente_idCliente int not null,
add constraint fk_endereco_cliente
foreign key(cliente_idCliente)
```

```
references Cliente(idCliente);

-- raca(n) : porteDoPet(1)
alter table Raca
add column porteDoPet_idporteDoPet int not null,
add constraint fk_raca_porte_do_pet
foreign key(porteDoPet_idPorteDoPet)
references PorteDoPet(idPorteDoPet);

-- pet(n) : raca(1)
alter table Pet
add column raca_idRaca int not null,
add constraint fk_pet_raca
foreign key(raca_idRaca)
references Raca(idRaca),

-- pet(n) : cliente(1)
add column cliente_idCliente int not null,
add constraint fk_pet_cliente
foreign key(cliente_idCliente)
references Cliente(idCliente),

-- pet(n) : sexo(1)
add column sexo_idSexo int not null,
add constraint fk_pet_sexo
foreign key(sexo_idSexo)
references Sexo(idSexo);

-- cliente(n) : sexo(1)
alter table Cliente
add column sexo_idSexo int not null,
add constraint fk_cliente_sexo
foreign key(sexo_idSexo)
references Sexo(idSexo),

-- cliente(n) : tipoCliente(1)
add column tipoCliente_idTipoCliente int not null,
add constraint fk_cliente_tipo_cliente
foreign key(tipoCliente_idTipoCliente)
references TipoCliente(idTipoCliente);

-- Inserção dos dados
start transaction;

insert into Sexo(descricao)
values('Feminino'),
('Feminino');
```

```
insert into TipoCliente (descricao)
values('Pessoa Jurídica'),
('Pessoa Física');

insert into Cliente(nome, nascimento, sexo_idSexo, tipoCliente_idTipoCliente)
values('Taís Moreira', '2002-07-30', 1,1),
('Jujuba Silva', '1970-01-07',2,2);

insert into Telefone(numero,cliente_idCliente)
values('987654321',1),
('123456789',2);

insert into Endereco(logradouro, numero, complemento, bairro, cidade, cep, uf,
cliente_idCliente)
values('Rua da limpeza', '1000', 'Bananeirado lado','Dados de
Lar','Fortaleza','15786254','CE',1),
('Rua da manipulação', '0001', 'Mercado Rosa','Lar de
Dados','Fortaleza','98765432','CE',2);

insert into PorteDoPet(descricao)
values('Porte Grande'),
('Porte Pequeno');

insert into Raca(nome, descricao, porteDoPet_idporteDoPet)
values('Gato','dócil e experiente no banho',1),
('Papagaio','agressivo',2);

insert into Pet (nome, nascimento, raca_idRaca,cliente_idCliente,sexo_idSexo)
values('Lila','2024-03-28',1,1,1),
('Bubu','2024-07-28',2,2,2);

-- commit;
```

Descrição do problema:

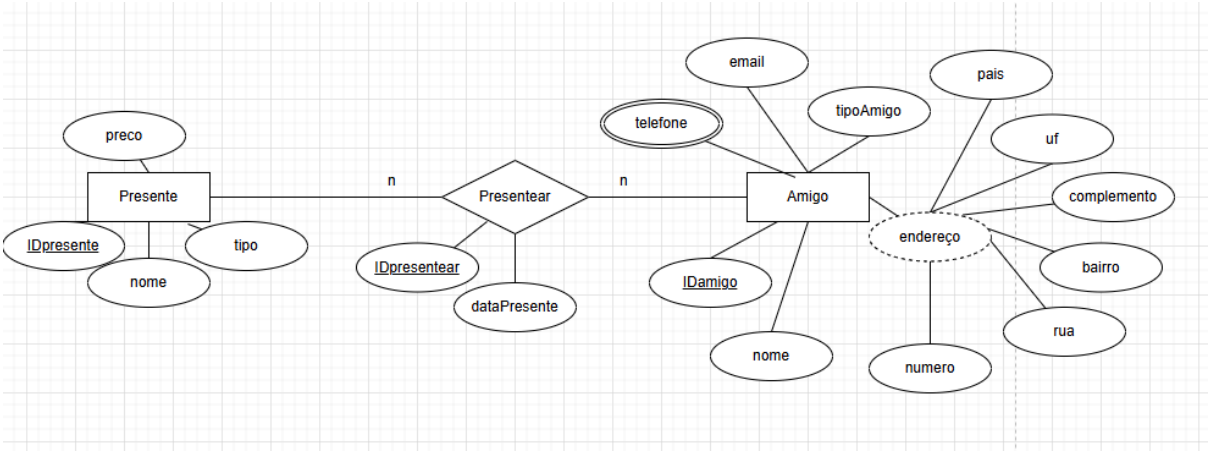
Cha Hae-In é uma caçadora de nível S mas tem uma memória não muito boa e acaba tendo dificuldade de lembrar que presentes ela deu para seus amigos ou parentes. As vezes ela até lembra que presenteou, mas não tem certeza do que foi, a data que presenteou ou quanto custou e ela precisa registrar isso em algum lugar. Cada ***presente*** *tem um nome, preço e um tipo do presente.*

Ela também tem uma classificação de pessoas/amigos, em que ela divide por ***tipos****, para facilitar escolher o valor que pretende gastar com cada presente. Uma pessoa da família sempre irá receber um presente melhor que um colega de trabalho. Um amigo do peito receberá um presente bom mas não tão bom quanto o que a família recebe. O Amigo recebedor de presente tem um nome, endereço,* ***telefones****, e-mail, Tipo de Amigo.*

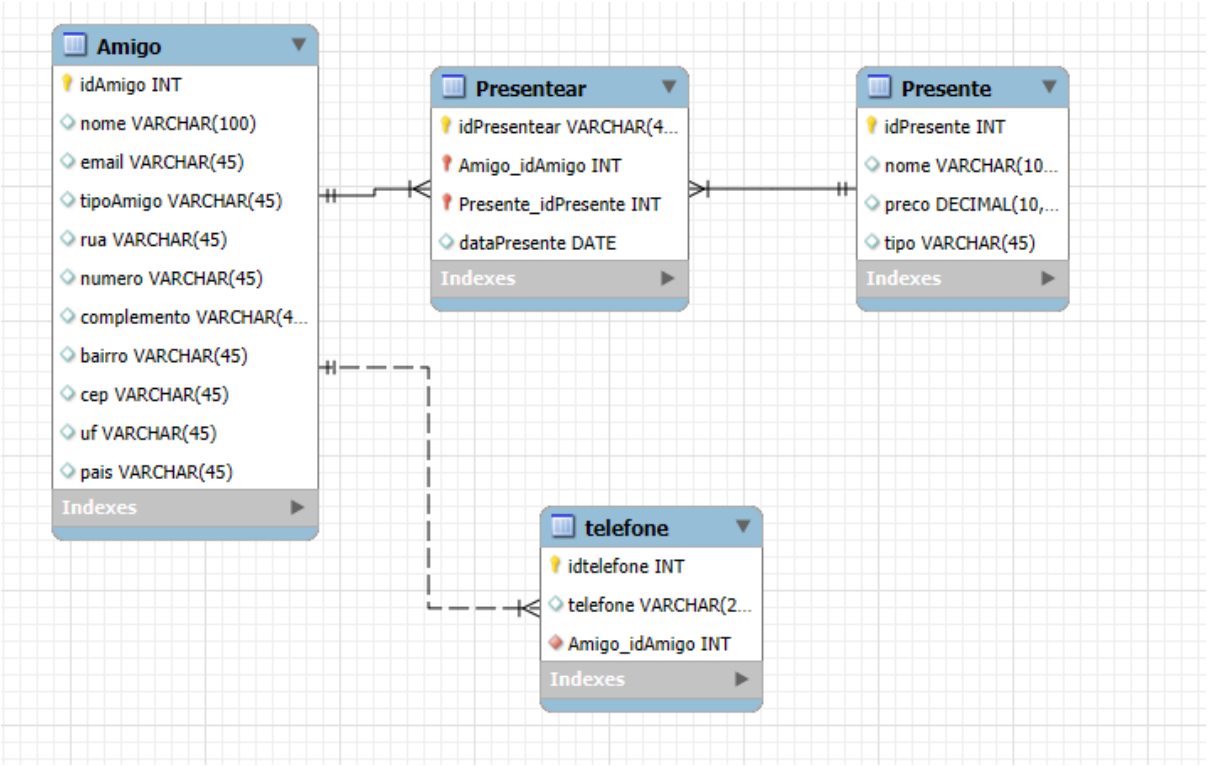
Importante ressaltar que ela pode dar, por exemplo, um perfume para várias pessoas mas ela não quer repetir um presente que já deu. Assim, precisa controlar de alguma forma a data que ela deu um presente para um amigo e assim ela conseguirá consultar no sistema uma lista de todos os presentes que já deu para cada amigo e ver a data que presentou.

Para resolver esse problema, vamos dimensionar um sistema para ajudar a Carol a resolver esse problema:
1. Criar o Diagrama de Entidade e Relacionamento (feito usando um software de desenho como o Draw.Io);
2. Criar o Modelo de Entidade e Relacionamento (feito no Workbench);
3. Criar os Scripts de Criação do Banco de Dados.

Diagrama



Workbench



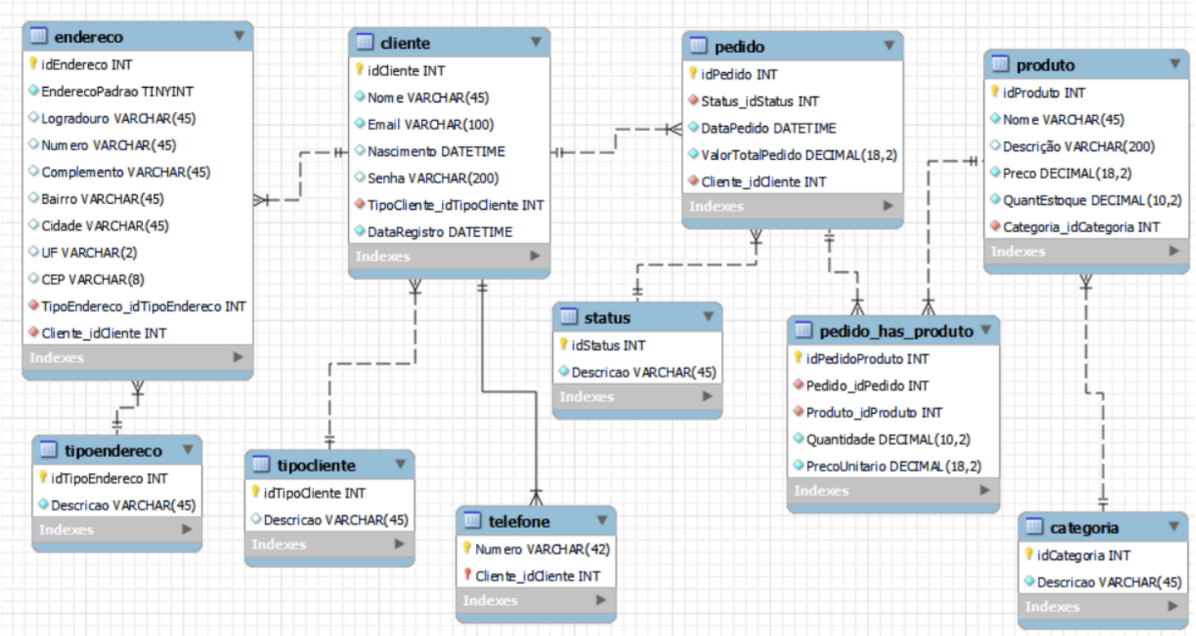
SQL

```sql
CREATE DATABASE PresentesAmigos;
USE PresentesAmigos;
-- DROP DATABASE presentesamigos;

CREATE TABLE Amigo(
    idAmigo int auto_increment,
    nome varchar(100) not null,
    email varchar(50) not null,
    tipoAmigo varchar(20) not null,

    pais varchar (30) not null,
    cep varchar(10) not null,
    cidade varchar(40) not null,
    UF varchar(2) not null,
    bairro varchar(100) not null,
    rua varchar(100) not null,
    numero varchar(10),
    complemento varchar(100),
    PRIMARY KEY (idAmigo)
);

CREATE TABLE Presente(
```

```
    idPresente int auto_increment,
    nome varchar(100) not null,
    preco decimal(10,2),
    tipo varchar(50) not null,

    PRIMARY KEY (idPresente)
);

CREATE TABLE Telefone(
    idTelefone int auto_increment,
    telefone varchar(20) not null,

    amigo_idAmigo int not null,
    PRIMARY KEY (idTelefone),
    INDEX fk_telefone_amigo1_idx (amigo_idAmigo ASC)
);

CREATE TABLE Presentear(
    idPresentear int,
    presente_idPresente int,
    amigo_idAmigo int,
    dataPresente date not null,

    PRIMARY KEY (idPresentear),
    INDEX fk_presentear_amigo1_idx (amigo_idAmigo ASC)
);

ALTER TABLE Telefone
    ADD CONSTRAINT fk_telefone_amigo1
        FOREIGN KEY (amigo_idAmigo)
        REFERENCES Amigo(idAmigo);

ALTER TABLE Presentear
    ADD CONSTRAINT presentear_ibfk_1
        FOREIGN KEY (presente_idPresente)
        REFERENCES Presente (idPresente),
    ADD CONSTRAINT fk_presentear_amigo1
        FOREIGN KEY (amigo_idAmigo)
        REFERENCES Amigo (idAmigo);
```

Ambiente de Dados – DML 2

Alan está querendo criar um sistema que permita ele controlar seus gastos e despesas. Neste sistema deverão ser controladas as contas que ele tem e seus tipos que podem ser cartões de crédito e contas correntes em bancos. Fato que o sistema precisará ter um cadastro de usuários pois ele deverá permitir que mais de uma pessoa (usuário) utilize o sistema. Cada conta deve ter o registro do seu saldo inicial para sabermos quanto havia disponível no momento do início do uso do sistema. Fato também que cada conta deverá ter uma descrição que indicará qual é a conta (cartão visa, conta corrente no BNB, etc). Para cada conta deverei registrar as movimentações que serão nada mais que os registros de cada transação que o usuário fizer. Se a pessoa utilizar o cartão de crédito para comprar um lanche, deverá registrar no sistema, incluindo a data da movimentação, uma descrição do que foi, o valor e indicar o tipo da movimentação que pode ser um débito ou crédito. Claro que preciso também relacionar uma categoria a cada transação de movimentação pois se eu precisar, por exemplo, saber quanto gastei em lanches no semestre, preciso deixar já registrado na tabela de movimentação. Nesta ideia, desenvolvi o modelo de entidade e relacionamento que segue. Utilizando o banco de dados definido no modelo de entidade e relacionamento acima, escreva as consultas SQL solicitadas em cada uma das questões.



01 - Crie um relatório mostrando o valor total que cada usuário movimentou, agrupado por conta. Deve ser mostrado: Nome do usuário, Nome da Conta, Valor total.

02 - Crie um relatório mostrando o valor total e a média das movimentações em cada mês/ano, agrupado por Categoria.
Deve ser mostrado: Categoria, Mês/Ano, Valor total, Média.

03 - Listar Todos os usuários que fizeram supermercado no mês de novembro de 2019, totalizando as compras deles em novembro de 2019, calculando a média das compras, contando a quantidade de compras, mostrando o maior valor de compra e o menor valor de compra.

04 - Listar todos os usuários que não têm contas relacionadas.

05 - Listar todos os Tipos de Movimento que não tiverem nenhum registro associado na tabela de movimentação.

06 – Liste as contas que tiveram mais de 1.500,00 em movimentações no ano de 2020.