

## BANCO DE DADOS

### Trabalho – Relatório

<b>Curso:</b>	Análise e Desenvolvimento de Sistemas
<b>Aluno(a):</b>	Isabelly Pereira Neto
<b>RU:</b>	5161877

### 1. 1ª Etapa – Modelagem

**Pontuação:** 30 pontos.

Dadas as regras de negócio abaixo listadas, referentes ao estudo de caso de uma Rede de Hotéis, elabore o Modelo Entidade-Relacionamento (MER), isto é, o modelo conceitual.

O Modelo Entidade-Relacionamento (MER) deve contemplar os seguintes itens:

- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

Uma Rede de Hotéis necessita controlar os dados dos funcionários, das unidades, dos quartos, dos hóspedes, das reservas e dos pagamentos. Para isso, contratou um profissional de Banco de Dados, a fim de modelar o Banco de Dados que armazenará todos os dados.

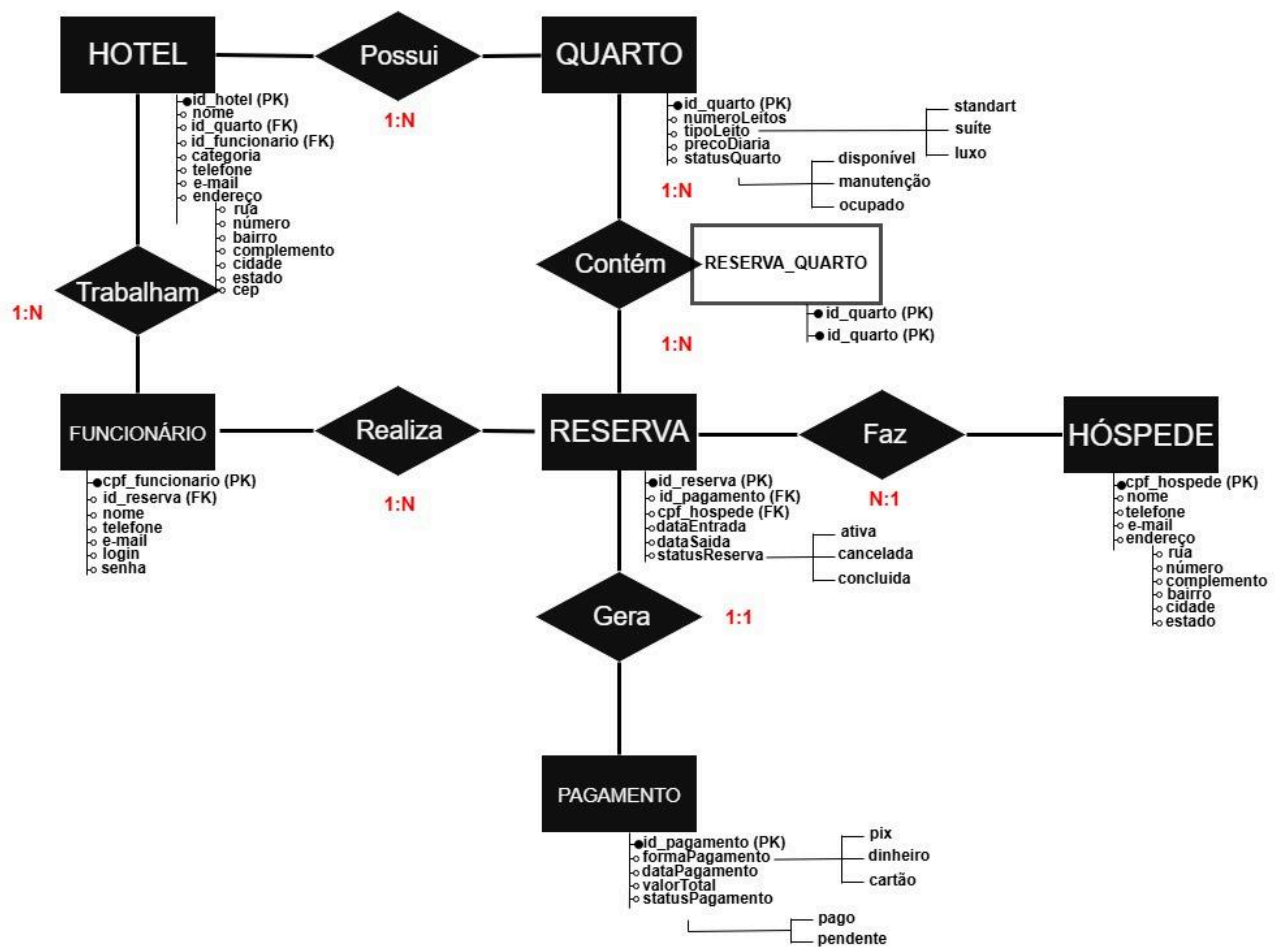
As regras de negócio são:

- Funcionário – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail, login e senha;

- Hotel – Deverão ser armazenados os seguintes dados: identificação do hotel, nome, categoria, telefone, e-mail e endereço, sendo o endereço composto por rua, número, complemento, bairro, CEP, cidade e estado;
- Quarto – Deverão ser armazenados os seguintes dados: identificação do quarto, número de leitos, tipo (*standard*, luxo ou suíte), preço da diária e *status* (disponível, ocupado ou manutenção);
- Hóspede – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail e endereço, sendo o endereço composto por rua, número, complemento, bairro, CEP, cidade e estado;
- Reserva – Deverão ser armazenados os seguintes dados: identificação da reserva, data de entrada, data de saída e *status* (ativa, cancelada ou concluída);
- Pagamento – Deverão ser armazenados os seguintes dados: identificação do pagamento, forma de pagamento (cartão, pix ou dinheiro), data do pagamento, valor total e *status* (pago ou pendente);
- Um hotel possui um ou vários quartos;
- Um ou vários funcionários trabalham em um hotel;
- Um funcionário realiza uma ou várias reservas;
- Um ou vários quartos fazem parte de uma ou várias reservas;
- Um hóspede pode fazer uma ou várias reservas;
- Uma reserva gera um pagamento.

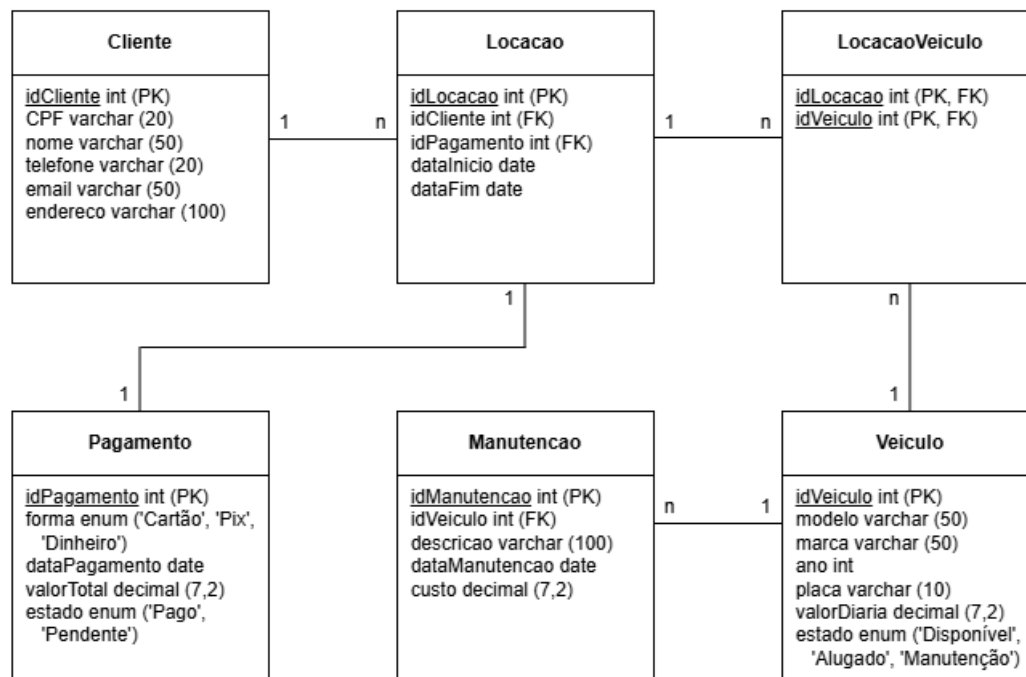
**Importante:**

- O Modelo Entidade-Relacionamento (MER) deve considerar somente as regras de negócio dadas, não podendo ser criada nenhuma outra entidade ou atributo que não estejam nas regras de negócio;
- Em caso de haver entidade associativa, a mesma deve ser representada pela “Representação 1” (texto da Aula 1 – Fundamentos de Banco de Dados, Figura 25);
- Em caso de haver cardinalidade (1,1), a chave estrangeira deve fazer parte da entidade que possui o maior número de chaves estrangeiras.



## 2. 2ª Etapa – Implementação

Considere o seguinte Modelo Relacional (modelo lógico), referente ao estudo de caso de uma Locadora de Veículos:



Com base no Modelo Relacional dado e utilizando a *Structured Query Language* (SQL), no MySQL Workbench, implemente o que se pede.

**Importante:** Para testar o Banco de Dados após a implementação, utilize os comandos contidos no arquivo “Trabalho – Populando o Banco de Dados” para popular as tabelas. Tal arquivo contém todos os comandos de inserção dos dados (fictícios) necessários para a realização dos testes.

**Pontuação:** 30 pontos.

1. Implemente um Banco de Dados chamado “LocadoraVeiculos”. Após, implemente as tabelas, conforme o Modelo Relacional dado, observando as chaves primárias e as chaves estrangeiras. Todos os campos, de todas as tabelas, não podem ser nulos (*not null*).

SQL

```
CREATE DATABASE locadoraVeiculos;  
USE locadoraVeiculos;
```

```
CREATE TABLE IF NOT EXISTS Pagamento (  
    idPagamento int NOT NULL UNIQUE,  
    forma enum ('Pix', 'Cartão', 'Dinheiro') NOT NULL,  
    dataPagamento date NOT NULL,  
    valorTotal decimal(7,2) NOT NULL,  
    estado enum ('Pago', 'Pendente') NOT NULL,  
    PRIMARY KEY (idPagamento)  
);  
  
CREATE TABLE IF NOT EXISTS Cliente (  
    idCliente int NOT NULL UNIQUE,  
    CPF varchar(20) NOT NULL,  
    nome varchar(50) NOT NULL,  
    telefone varchar(20) NOT NULL,  
    email varchar(50) NOT NULL,  
    endereco varchar(100) NOT NULL,  
    PRIMARY KEY (CPF)  
);  
  
CREATE TABLE IF NOT EXISTS Locacao (  
    idLocacao int AUTO_INCREMENT NOT NULL UNIQUE,  
    idCliente int NOT NULL,  
    idPagamento int NOT NULL,  
    dataInicio date NOT NULL,  
    dataFim date NOT NULL,  
    PRIMARY KEY (idLocacao),  
    CONSTRAINT fkPagamento FOREIGN KEY (idPagamento) REFERENCES  
Pagamento(idPagamento),  
    CONSTRAINT fkCliente FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente)  
);  
  
CREATE TABLE IF NOT EXISTS LocacaoVeiculo (  
    idLocacao int NOT NULL,  
    idVeiculo int NOT NULL,  
    PRIMARY KEY (idLocacao, idVeiculo),  
    CONSTRAINT fkLocacao FOREIGN KEY (idLocacao) REFERENCES Locacao(idLocacao),  
    CONSTRAINT fkVeiculo FOREIGN KEY (idVeiculo) REFERENCES Veiculo(idVeiculo)  
);  
  
CREATE TABLE IF NOT EXISTS Veiculo (  
    idVeiculo int AUTO_INCREMENT NOT NULL UNIQUE,  
    modelo varchar(50) NOT NULL,  
    marca varchar(50) NOT NULL,  
    ano int NOT NULL,  
    placa varchar(10) NOT NULL,  
    valorDiaria decimal(7,2) NOT NULL,
```

```
estado enum ('Alugado', 'Disponível', 'Manutenção') NOT NULL,  
PRIMARY KEY (idVeiculo)  
);  
  
CREATE TABLE IF NOT EXISTS Manutencao (  
    idManutencao int AUTO_INCREMENT NOT NULL UNIQUE,  
    idVeiculo int NOT NULL,  
    descricao varchar(100) NOT NULL,  
    dataManutencao date NOT NULL,  
    custo decimal(7,2) NOT NULL,  
    PRIMARY KEY (idManutencao),  
    CONSTRAINT fkVeiculoManutencao FOREIGN KEY (idVeiculo) REFERENCES  
Veiculo(idVeiculo)  
);
```

OBS: a princípio utilizei o CPF como PK, mas ao popular o banco de dados, notei que a PK estava sendo o idCliente, por isso decidi alterar meu código para seguir o mesmo padrão.

**Pontuação:** 10 pontos.

2. Implemente uma consulta para listar a descrição, a data e o custo de todas as manutenções realizadas nos veículos.

SQL

```
SELECT descricao, dataManutencao, custo  
FROM Manutencao;
```

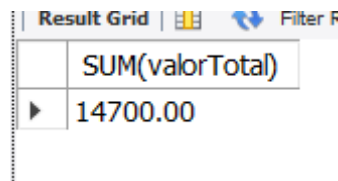
	descricao	dataManutencao	custo
▶	Troca de óleo e revisão geral	2024-12-09	200.00
	Substituição de pneu	2024-12-10	600.00
	Troca de pastilhas de freio	2024-12-14	450.00
	Alinhamento e balanceamento	2024-12-18	150.00
	Revisão elétrica completa	2024-12-28	500.00
	Reparo na suspensão	2025-01-05	700.00
	Troca do sistema de escapamento	2025-01-07	750.00
	Troca de bateria	2025-01-17	400.00
	Substituição do filtro de ar	2025-01-17	120.00
	Pintura e retoques na lataria	2025-01-28	900.00

**Pontuação:** 10 pontos.

3. Implemente uma consulta para listar o valor total arrecadado pela locadora. Lembre-se que pagamentos “pendentes” não fazem parte da soma.

SQL

```
SELECT SUM(valorTotal) FROM pagamento  
WHERE estado = 'Pago';
```



SUM(valorTotal)
14700.00

**Pontuação:** 10 pontos.

4. Implemente uma consulta para listar o modelo e a marca dos veículos, bem como o número de vezes que cada um foi locado. A listagem deve ser mostrada em ordem decrescente pelo número de aluguéis.

Dica: Utilize a cláusula *group by*.

SQL

```
SELECT veiculo.marca, veiculo.modelo,  
COUNT(LocacaoVeiculo.idVeiculo) AS totalDeLocacao  
FROM veiculo  
INNER JOIN LocacaoVeiculo ON veiculo.idVeiculo = LocacaoVeiculo.idVeiculo  
GROUP BY veiculo.marca, veiculo.modelo  
ORDER BY totalDeLocacao DESC;
```

	marca	modelo	totalDeLocacao
▶	Hyundai	HB20	4
	Renault	Duster	3
	Volkswagen	Gol	2
	Toyota	Corolla	2
	Ford	Fiesta	2
	Fiat	Toro	2
	Jeep	Compass	2
	Chevrolet	Onix	1
	Honda	Civic	1
	Chevrolet	Cruze	1

**Pontuação:** 10 pontos.

5. Implemente uma consulta para listar o nome dos clientes que possuem pagamento “pendente”, bem como o valor devido por eles. A listagem deve ser mostrada em ordem alfabética crescente pelo nome dos clientes.

Dica: Utilize a cláusula *group by*.

SQL

```
SELECT cliente.nome,  
SUM(pagamento.valorTotal) AS valorPendente  
FROM cliente  
JOIN locacao ON cliente.idCliente = locacao.idCliente  
JOIN pagamento ON locacao.idPagamento = pagamento.idPagamento  
WHERE pagamento.estado = 'pendente'  
GROUP BY cliente.nome  
ORDER BY cliente.nome;
```

Result Grid | Filter Rows:

	nome	valorPendente
▶	João da Silva	880.00
	Lucas Martins	2220.00
	Pedro dos Santos	280.00