Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Trabalho Prático

Tiago Baptista, João Leal, Fábio Silva

Novembro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

LEIParking

Tiago Baptista, João Leal, Fábio Silva

Novembro, 2018

Resumo

Área de Aplicação: Desenho e arquitectura de Sistemas de Bases de Dados.

Este documento retrata de forma aprofundada a organização e implementação de uma base de dados que suporta o funcionamento de uma aplicação, que permite reservar lugares de estacionamento com antecedência, de uma empresa responsável pela gestão de parques de estacionamento.

Numa parte inicial do relatório serão retratadas as fases do projeto, entre elas, contextualização do tema do relatório, motivação, objetivos, análise de requisitos e a estrutura do trabalho.

De seguida, vão ser identificadas as entidades envolvidas no projeto, os seus atributos e relacionamentos entre elas. Posteriormente vão ser identificadas as chaves candidatas e as chaves primárias escolhida para cada entidade. Após a conclusão do modelo concetual, é apresentado o modelo lógico onde será implementada toda a estrutura da base de dados.

Por fim, os dados são analisados e validados através do método de normalização das relações existentes.

Palavras-Chave: Bases de Dados Relacionais, modelo conceptual e lógico, análise de requisitos, entidades, atributos, relacionamentos,chave primária,chave estrangeira.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Neste trabalho é-nos pedido que façamos uma análise, um planeamento, seguidos da estruturação e implementação de um sistema de base de dados relacional que sirva de suporte e optimize um sistema de smart parking. À priori sabemos que a nossa base de dados será um sistema que gere informação de modo a guardá-la e com que o utilizador possa extrair uma parte dessa informação de modo eficaz, rápido e preciso. Esta base de dados é composta por um conjunto de tabelas e associações entre as mesmas. As linhas representam instâncias de entidades e nas colunas estão representados os dados. O sistema de smart parking centra-se na base de dados portanto é extremamente importante haver um bom planeamento e estruturação desta, para que tenha a eficácia que pretendemos.

Com isto, neste primeiro capítulo vamos introduzir, de forma geral, o nosso projeto. Vamos também introduzir o contexto no qual se desenvolve o caso em estudo.
Relativamente ao resto do relatório, apresentaremos ambos os modelos conceptual e lógico, um após o outro, respetivamente. Por fim, passaremos a implementar fisicamente os modelos anteriormente apresentados e analisados através das relações existentes neles.

1.1. Contextualização

Hoje em dia, o congestionamento do tráfego de veículos é uma das grandes razões que contribuem para a frustração dos condutores, mercadores, empregados e funcionários públicos, em grande parte das cidades pelo mundo inteiro. Como tal, e uma vez que vivemos na era da IoT, é fácil perceber o porquê de algo como o “Smart Parking” ser visto como uma solução, uma vez que oferece bastantes vantagens ao condutor, uma vez que este encontra um lugar livre efetivamente, poupando tempo e dinheiro, uma vez que não tem de andar à procura de um lugar Além disso, e em prol de algo maior, o “Smart Parking” contribui para um menor congestionamento do tráfego de veículos e uma menor poluição no ambiente em geral, uma vez uma solução de estacionamento ótima leva a um tempo de condução reduzido.

Sensores embutidos no chão, ou câmeras montadas em postes de luz ou estruturas de edifícios, determinam se os espaços de estacionamento estão ocupados ou disponíveis. Esses dados são transmitidos para uma plataforma central de estacionamento inteligente que agrupa todas as informações e cria um mapa de estacionamento em tempo real. Assim, os motoristas

usam esse mapa para localizar o lugar livre mais próximo do lugar em que se encontram, em vez de perderem o seu tempo para encontrar um.

Assim sendo, e com estes objetivos em mente, Tomás Taveira, um empresário novo e inovador comprou a empresa *"BelosParques"* , que tinha em sua posse 3 Parques, dos quais 2 deles localizados em Braga e 1 no Porto, e por conseguinte, decidiu implementar um sistema de smart parking, de forma a otimizar como o estacionamento é realizado nos "seus" espaços, dando origem à sua nova cara, a "LEIParking".

Após a implementação desta tecnologia nos seus Parques recém-adquiridos, **basta a um condutor reservar um lugar, para um determinado intervalo de tempo e pagar no ato da reserva, tudo de forma online**, **ficando com um QR Code guardado, para depois fazer o check-In da sua reserva, nesse lugar.** Sem ser necessária a procura ou algum tipo de deslocamento. De notar que, de forma a contribuir ainda mais para uma menor poluição em geral e como "incentivo" aos condutores, **a "LEIParking" aplica uma taxa de estacionamento menor aos carros Híbridos/Elétricos (Tipo 1) do que aos carros a Gasóleo/Gasolina (Tipo 2).**

Depois desta atualização tecnológica, a afluência de veículos nos parques da "LEIParking" aumentou, conseguindo neste momento estacionar facilmente naquela área, devido ao facto de já saber onde ter de estacionar e pelo quão simples o processo de reserva de lugar é.

1.2. Apresentação do Caso de Estudo

O corpo administrativo da empresa pretende desenvolver um Sistema de Base de Dados que permita implementar o sistema de smart parking. Com esta implementação consegue-se minimizar quaisquer problemas de organização e recolha de dados que possa ocorrer, bem como facilita reduz significativamente o tempo que demora o estacionamento.

Nos tempos que correm tempo é o que falta à maioria das pessoas que vivem o seu quotidiano. Sendo assim, é crucial para esta empresa que o tempo de estacionamento de cada condutor seja o mínimo possível, de modo que não só aumenta o lucro obtido, mas também aumenta a satisfação dos seus clientes.

Com esta implementação será então possível efetuar uma gestão eficiente dos dados de cada lugar de estacionamento, tendo uma visão muito mais geral e objetiva de todos os lugares em si, dando ao cliente um conjuntos de informações úteis para que este tenha o maior proveito do serviço prestado.

1.3. Motivação e Objectivos

O ato de estacionar é algo fundamental no dia-a-dia de cada pessoa. Um indivíduo que trabalha e tem uma vida familiar à parte do trabalho, não pode perder tempo no simples ato de estacionar o seu veículo, o que se pode tornar numa tarefa bastante complexa em muitas cidades atualmente. Por esta razão e para proporcionar o melhor serviço possível aos seus clientes, a nossa empresa tomou a decisão de atualizar o seu serviço para um serviço muito mais eficiente e fácil.

O objetivo fundamental desta implementação é atribuir ao cliente um mapa detalhado dos lugares de estacionamento mais próximos, para que estes tenham maior facilidade em estacionar. Assim, esta empresa vai contribuir para uma melhor qualidade de vida dos seus clientes, bem como o bem-estar dos mesmos.

1.4. Estrutura do Relatório

O conteúdo deste relatório vai conter uma descrição detalhada do caso de estudo, elaboração e implementação da base de dados utilizada pela nossa empresa.

No primeiro capítulo é feita uma introdução a este trabalho apresentando-se a contextualização do nosso caso de estudo, bem como os objetivos e motivação da empresa pelo desenvolvimento deste projeto.

No capítulo 2 é feita uma recolha de todos os requisitos, que consiste em apresentar e descrever os requisitos do sistema e de todas as entidades da Base de Dados do caso.
Posteriormente à análise detalhada dos requisitos, no terceiro capítulo apresentamos o modelo conceptual deste caso. Aqui são identificadas as entidades e os atributos de cada entidade. São analisados os relacionamentos e determinadas as chaves candidatas, primárias e alternadas. Por fim, são apresentadas as queries (sendo estas perguntas efetuadas pelo utilizador) e é verificada a não existência de redundância.

Quanto ao modelo lógico, este é apresentado no capítulo 4. A sua primeira parte é dedicada à conversão de modelo conceptual para modelo lógico e apresentação do mesmo. Nos tópicos seguintes é feita a análise das entidades e dos relacionamentos contidos no modelo. É também realizada a validação, reavaliação e revisão do modelo.

No quinto capítulo, é apresentada a tradução do modelo lógico para o SGBD escolhido, sendo esta a implementação física da nossa Base de Dados. São descritas as relações base inicialmente, seguidas da tradução das interrogações do utilizador para SQL. Posteriormente, é feita a análise das transações e definidos os índices em SQL.

Com isto, é necessário fazer uma estimativa do espaço em disco ocupado pela nossa Base de Dados, bem como definir os mecanismos de segurança em SQL. Assim, é realizada a revisão do sistema com a vista do utilizador.

Por fim, são apresentadas as ferramentas utilizadas e a conclusão do relatório, bem como a bibliografia e uma lista de sinónimos.

2. Análise e levantamento de Requisitos

Neste capítulo são abordados e definidos todos os requisitos necessários para o bom funcionamento do sistema apresentado.

O projeto representa a base de dados de uma aplicação que permite reservar lugares de estacionamento, sendo que o domínio de dados desta empresa inclui os diversos parques, as reservas,o cliente, o seu veículo e os pagamentos.

O cliente pretende que a base de dados responda às seguintes interrogações :

-Quais foram os veículos que usaram o parque num determinado dia?

-Quais os parques com lugares livres numa determinada cidade, numa determinada data?

-Que lugares estão livres num determinado parque, numa determinada data?

-Quanto faturou a empresa num dado intervalo de tempo?

Deseja também poder aumentar a capacidade de um parque e consequentemente adicionar lugares de estacionamento.

2.1. Requisitos de descrição

De seguida são apresentados os requisitos necessários para o funcionamento da empresa :

Veículo :

- Possui uma matrícula associada que permite identificar um veículo em específico;
- Possui um identificador para o tipo de combustível usado (gasóleo/gasolina ou híbrido/elétrico);

Cliente:

- Possui um identificador único;
- Possui registo do seu nome;
- Possui registo dos seus contactos (Telemóvel e Email);

Reserva:

- Possui um identificador único;
- Possui uma data de início do lugar reservado;
- Possui um data limite do lugar reservado;
- Possui indicador do lugar a reservar e do piso;
- É efetuada sobre um parque específico;

Pagamento:

- Possui um identificador único;
- Possui a data em que foi efetuado;
- Possui uma taxa horária dependendo do tipo de veículo a que se refere;

Lugar de Estacionamento

- Possui indicação do seu número;
- Possui indicação do piso onde se encontra;
- Possui identificador que permite verificar a sua ocupação

2.2. Requisitos de exploração

De seguida passa-se a explicitar quais foram as funcionalidades que o cliente pretendia que a base de dados fosse capaz de efetuar. Sabendo que, a base de dados foi pedida por uma empresa que pretende a base de dados para suportar o funcionamento de uma aplicação, que permite registos sobre reservas de lugares feitas sobre parques de estacionamento,verificou-se que :

- Para que um cliente possa efetuar uma reserva tem de estar, obrigatoriamente, registado, assim como a sua viatura;
- No registo da viatura tem de estar indicado o tipo de combustível utilizado;
- Ao ser efetuada uma reserva tem de se indicar, obrigatoriamente, a data de entrada e saída do veículo (mês/dia/hora);
- Ao ser efetuada a reserva tem de ser indicado obrigatoriamente o lugar que pretende ocupar;
- A base de dados tem de ser capaz de dar informações sobre os níveis de ocupação de cada parque de estacionamento, das várias regiões do país e quais são efetivamente esses lugares;
- Tem de ser possível verificar o estado das reservas(ativa ou não ativa) para todos os parques espalhados pelo país;

- A numeração dos lugares dos parques segue ordem crescente e não possui repetidos, sendo que de parque para parque não ocorre o reset ao número do lugar;
- Caso a data da saída da viatura seja posterior à data indicada na Reserva é aplicado um agravamento no valor a pagar;
- Veículos com combustível tipo 1 (elétricos ou híbridos) possuem menor taxa por hora do que veículos do tipo 2 (gasóleo ou gasolina);

3. Modelo Conceptual

3.1. Apresentação da abordagem de modelação realizada

Após a análise de requisitos passou-se para a elaboração do modelo conceptual, uma vez que este não depende de considerações físicas nem de detalhes de implementação.

No desenvolvimento deste modelo foram sempre tidos em conta os requisitos necessários para o bom funcionamento do sistema.

3.2. Entidades do Problema

Nesta secção são identificadas as Entidades do sistema, que são os principais componentes para a estruturação e funcionamento da empresa e da app em questão. De seguida são listadas as entidades:

Veículo

Veículo vai ser uma entidade responsável por guardar informações acerca de uma viatura, vai ter dados acerca do tipo de combustível que usa e da sua matrícula. Esta entidade vai ter um papel fundamental pois todas as reservas efetuadas pelo cliente estão associadas a um veículo.

Cliente

Esta entidade vai ser responsável por registar os veículos, possui informações importantes, na medida que vão permitir, em caso de não pagamentos ou de ultrapassagem do limite máximo de tempo pago, que a aplicação possa ter conhecimento e reagir apropriadamente.

Reserva

Esta será a entidade mais importante do sistema, a verdadeira razão para a existência do problema, o bilhete vai conter toda a informação da reserva relativa a um determinado veículo, nomeadamente a data de início da reserva e a data de fim da reserva.

Parque

Esta entidade possibilita a existência de reservas, uma vez que estas reservas são feitas sobre um parque específico.

Pagamento

Esta entidade permite registar os pagamentos feitos sobre as reservas, para que as mesmas possam ser válidas.

LugaresParque

Esta entidade contêm informações sobre os lugares de um determinado parque, nomeadamente, o seu piso e estado de ocupação.

Tabela 1 - Dicionário de entidades

Entidade	Descrição	Alcunha	Ocorrência
Veículo	Entidade representativa de um veículo que tem associado uma ou mais reservas de estacionamento.	Carro	O Veículo é uma entidade fundamental pois é esta que faz uso do sistema e da aplicação em geral.
Cliente	Entidade representativa de um utilizador do sistema que tem associado um veículo.	Utilizador	O cliente é uma entidade bastante importante, pois guarda informações relevantes para o cálculo do pagamento a efetuar.
Reserva	Entidade representativa de uma reserva feita na aplicação, tem informação sobre a hora de início e fim	Marcação	A reserva é a entidade mais importante do sistema, pois é a principal razão para a sua existência

	da estadia do carro no respectivo parque.		
Parque	Entidade representativa dos Parques de estacionamento para os quais podem ser efetuadas reservas	Estacionamento	O Parque é importante pois permite a existência das reservas para um determinado local.
Pagamento	Entidade representativa do pagamento da reserva	Liquidação	O pagamento é crucial pois permite saber se a reserva está validada ou não.
LugaresParque	Entidade representativa dos lugares do Parque	Lugares	Os lugares são importantes pois a possibilidade de escolha de um lugar específico é um requisito pedido pelo cliente e que vai ajudar a determinar a ocupação de um parque num determinado momento

3.3. Identificação dos Relacionamentos

Depois da identificação das entidades presentes no modelo relacional passa-se a explicitar todos os relacionamentos entre elas.

LugaresParque e Reserva:

O relacionamento “processa” entre a entidade Parque e Reserva tem cardinalidade de 1 para N na medida em que podem ser processadas/efetuadas N reservas sobre um parque e cada reserva é processada por um parque.

Figura 1- Relacionamento LugaresParque-Reserva

Reserva e Pagamento:

O relacionamento “tem” entre a entidade Reserva e Pagamento tem cardinalidade de 1 para 1. Na medida em que uma Reserva tem um Pagamento associado, assim como, um Pagamento tem apenas uma Reserva.

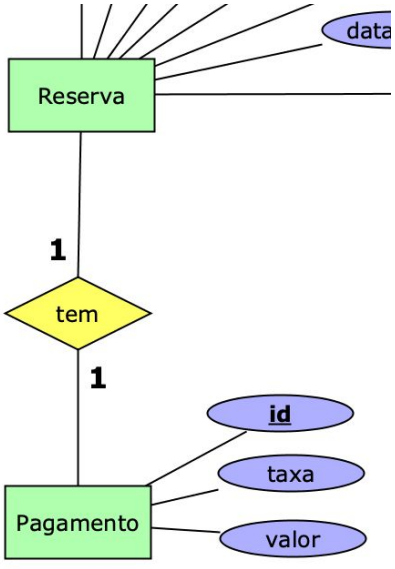

Figura 2 -Relacionamento Reserva-Pagamento

Veículo e Reserva:

O relacionamento “possui” entre a entidade Veículo e Reserva tem cardinalidade de 1 para N. Na medida em que um veículo possui N reservas e por outro lado N reservas podem ser possuídas por um veículo

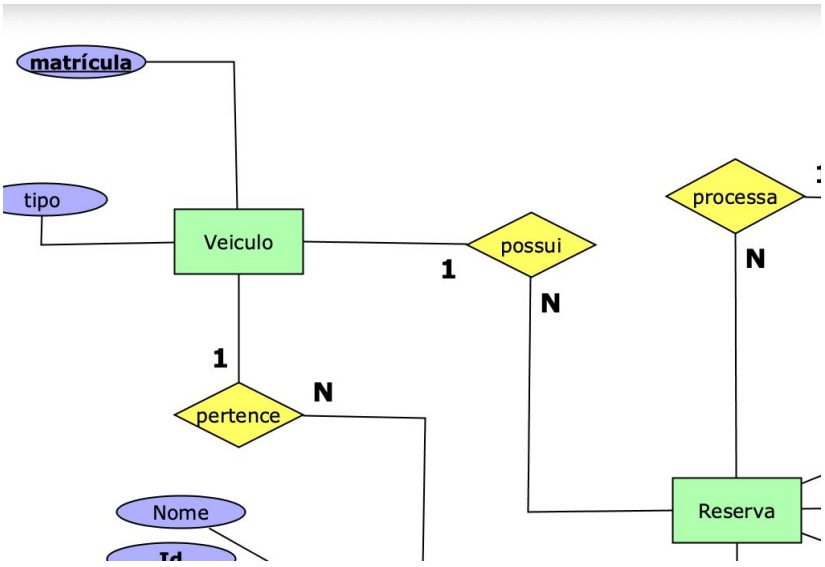

Figura 3 - Relacionamento Veículo-Reserva.

Cliente e Veículo

O relacionamento “pertence” entre as entidades Cliente e Veículo possui uma cardinalidade de 1 para N. Na medida em que N veículos pertencem a um cliente e um cliente tem como pertença N Veículos.

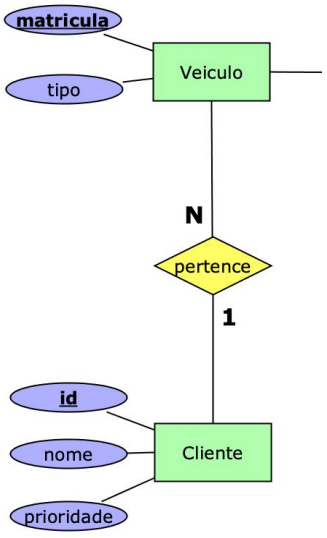

Figura 4 - Relacionamento Cliente-Veículo.

Parque e LugaresParque

O relacionamento “pertence” entre as entidades Parque e LugaresParque possui uma cardinalidade de 1 para N. Na medida em que N Lugares pertencem a um Parque e um Parquetem como pertença N Lugares.

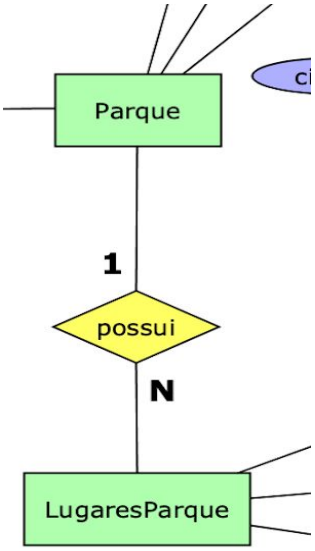

Figura 5 - Relacionamento Parque-LugaresParque.

Tabela 2 - Dicionário de Relacionamento

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Reserva	1	tem	1	**Pagamento**
Veiculo	1	possui	N	**Reserva**
Veiculo	N	pertencem	1	**Cliente**
Parque	1	possui	N	**LugaresParque**
LugaresParque	1	possui	N	**Reserva**

3.4. Identificação e associação de atributos com entidades e relacionamentos e respetivos domínios de valores

De seguida passa-se a explicitar todos os atributos/ propriedades de cada entidade que foram achados relevantes para o problema em causa.

Tabela 3 - Dicionário de dados dos atributos das entidades do modelo

Entidade	Atributo	Descrição	Tipo de dados	NULL	Tipo de Atributo	Domínio
Parque	id	Identificador único que cada parque registado no sistema possui.	INT	Não	Chave Primária	Inteiro
	capacidade	Número de lugares que um determinado parque possui.	INT	Não	Simples	Inteiro
	zona (composto) distrito	Aponta a localização geográfica de um parque, neste caso o seu distrito.	VARCHAR (24)	Não	Simples	24 Char variáveis
	zona (composto) cidade	Aponta a localização geográfica de um parque, neste caso a sua cidade.	VARCHAR (24)	Não	Simples	24 Char variáveis

Reserva	id	Identificador único que cada parque registado no sistema possui.	INT	Não	Chave Primária	Inteiro
	data_inicio	Indica a data de início de uma determinada reserva.	DATETIME	Não	Simples	Data
	data_fim	Indica a data de fim de uma determinada reserva.	DATETIME	Não	Simples	Data
	ativa	indica o estado da reserva.	BOOLEAN	Não	Simples	Tiny Int
Veiculo	matricula	Matrícula de um determinado veículo.	VARCHAR (24)	Não	Chave Primária	6 Char variáveis
	tipo	Tipo de combustível que o automóvel utiliza	INT	Não	Simples	Inteiro
Cliente	id	Identificador único que cada cliente registado no sistema possui.	INT	Não	Chave Primária	Inteiro
	nome	Indica o nome de um cliente.	VARCHAR (24)	Não	Simples	24 Char variáveis

Pagamento	id	Identificador único que cada cliente registado no sistema possui.	INT	Não	Chave Primária	Inteiro
	preco	Indica o valor a ser cobrado pela reserva	FLOAT	Não	Simples	Float
	taxa	Indica a taxa aplicada ao veículo usado na Reserva	FLOAT	Não	Simples	Float
	multa	Indica o valor da multa a pagar (caso exceda o tempo de reserva)	FLOAT	Sim	Simples	Float
LugaresParque	id	Identificador único que cada lugar registado no sistema possui.	INT	Não	Chave Primária	Inteiro
	piso	Indica o piso em que o lugar se encontra num parque.	INT	Não	Simples	Inteiro
	ocupado	Indica se o lugar se encontra ocupado por uma reserva.	BOOLEAN	Não	Simples	Tiny Int

3.5. Determinação das chaves candidatas, chaves primárias e chaves alternadas

Após a identificação de todas as entidades, os seus respectivos atributos e o relacionamentos entre elas passa-se a explicitar o processo de escolha das chaves primárias de cada entidade.

Sabendo que o conjunto de todos atributos de uma entidade são chaves candidatas do modelo e que uma chave primária é um campo(s) (de uma tabela) que nunca se repete, esta característica possibilita-o a tornar-se o identificador único de um registo. Pode-se agora explicitar as escolhas feitas para este problema.

Cliente:

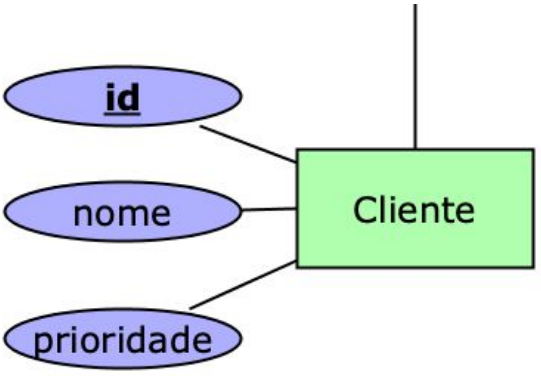

Figura 6 - Chaves Cliente.

Chaves Candidatas: {Nome,Id}

Chave Primária: As duas chaves candidatas são ambas únicas e invariáveis para cada cliente, mas elegemos o Id como chave primária .

Veículo:

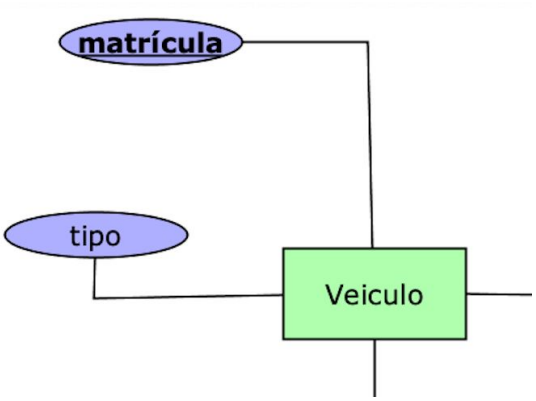

Figura 7- Chaves Veículo.

Chaves Candidatas: {matrícula}

Chave Primária: A única chave candidata foi a que foi eleita como chave primária.

Parque

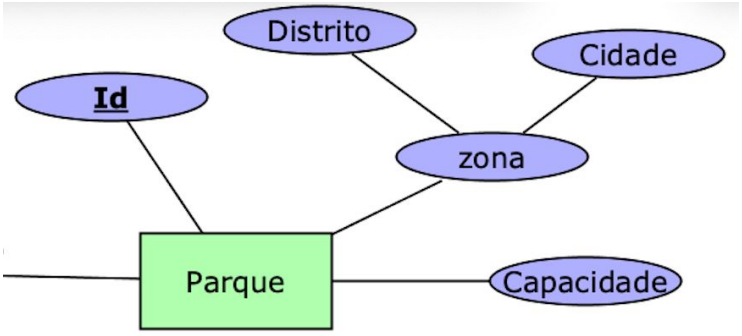

Figura 8 - Chaves Parque.

Chaves Candidatas: {Id}

Chave Primária: A única chave candidata foi a que foi eleita como chave primária.

Reserva

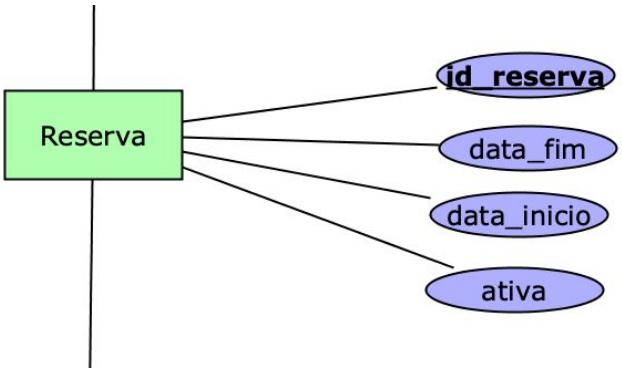

Figura 9 - Chaves Reserva.

Figura

Chaves Candidatas: {Id}

Chave Primária: As duas chaves candidatas são ambas únicas e invariáveis para cada cliente, mas elegemos o Id como chave primária .

Pagamento

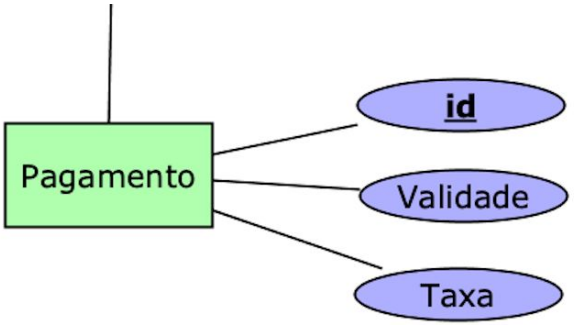

Figura 10 - Chaves Pagamento.

Chaves Candidatas: {Id}

Chave Primária: A única chave candidata foi a que foi eleita como chave primária.

LugaresParque

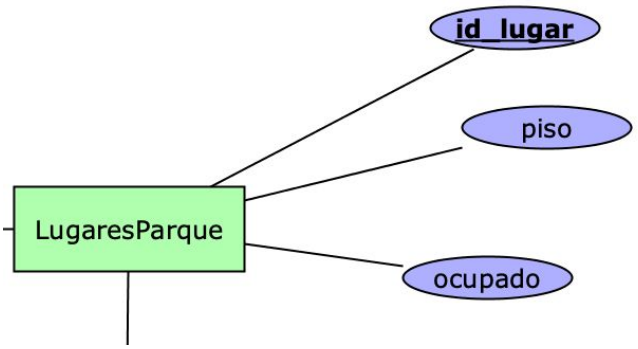

Figura 11 - Chaves LugaresParque.

Chaves Candidatas: {Id}

Chave Primária: A única chave candidata foi a que foi eleita como chave primária.

3.6. Revisão e validação do modelo com o utilizador

Dado como terminado a elaboração do modelo conceptual é necessário a sua validação. E quando falamos em validação dizemos que esta só é obtida se o modelo responder a todas as perguntas que o utilizador posso fazer. Assim, foram selecionadas algumas perguntas que consideramos adequadas, cuja viabilidade vai ser verificada através da comparação com o modelo conceptual.

1. Quais foram os veículos que usaram o parque numa determinada data?
2. Quais os parques com lugares livres numa determinada zona a uma dada hora ?
3. Que lugares estão livres num determinado parque a uma dada hora?
4. Quanto faturou a empresa num dado intervalo de tempo?

Validação

1. Quantos ,quais foram os veículos e os respetivos donos que usaram um parque num determinada data?

De maneira a saber quantos e quais foram os veículos que usaram um parque num determinado dia, são necessárias as entidades Reserva, Parque e Veículo.

Para saber as reservas efetuadas num determinado dia,ou seja, o número de veículos que usaram o parque necessitamos da Reserva.De maneira a saber os veículos associados a essas Reservas é necessário a entidade Veículo e de maneira a saber o parque o qual o utilizador quer procurar é necessária a entidade Parque.

2. Quais os parques com lugares livres numa determinada zona a uma dada hora?

De maneira a saber quais os parques livres numa determinada zona, é necessário a entidade Parque, LugaresParque e Reserva. A entidade LugaresParque, juntamente com a Reserva, fornecem a informação necessária para descobrir quais os lugares que são efetivamente livres, enquanto que a entidade Parque possui a informação necessária sobre a localização deste.

3. Que lugares estão livres num determinado parque a uma dada hora ?

De maneira a saber quais os lugares livres de um dado parque , é necessário a entidade Parque, LugaresParque e Reserva. A entidade LugaresParque, juntamente com a Reserva, fornecem a informação necessária para descobrir quais os lugares que são efetivamente livres, enquanto que a Parque possui a informação necessária sobre o Parque e a sua localização.

Estes lugares livres, são aqueles que não têm Reservas ativas na hora dada pelo Cliente e cuja ocupação do lugar está efetivamente livre (ocupado = 0).

4. Quanto faturou a empresa num dado intervalo de tempo?

De maneira a determinar a faturação total de uma empresa num determinado intervalo de tempo será necessário o uso das entidades Pagamento e Reserva. A entidade Pagamento reúne todos os pagamentos feitos de todas as reservas e a entidade Reserva dá-nos a informação das reservas efetuadas num determinado intervalo de tempo.

3.7. Apresentação e explicação do diagrama ER

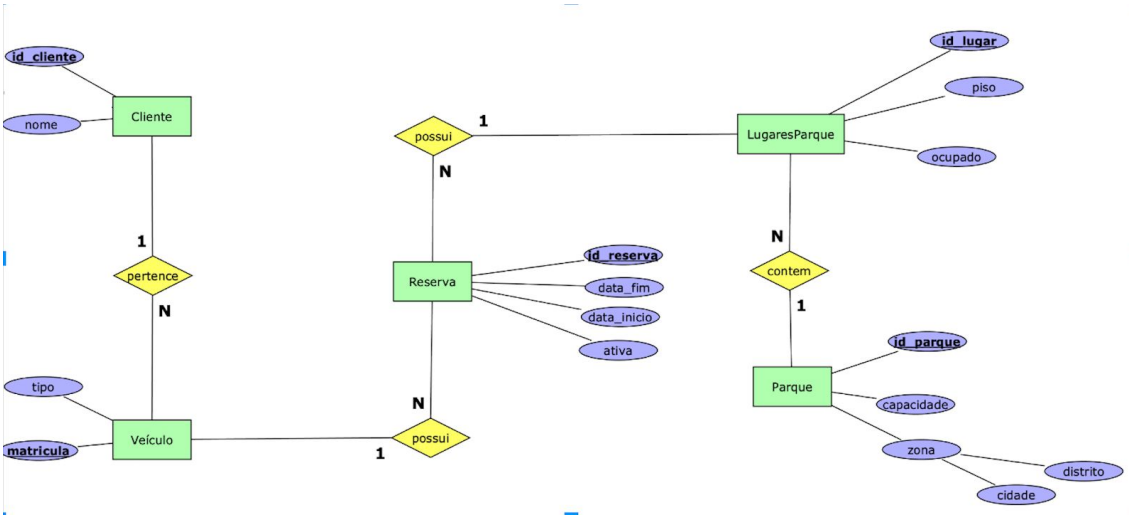

Figura 12 - Diagrama ER.

A entidade central,e que fundamenta a criação deste sistema, é a Reserva que possui o id como identificador principal, sendo que esta entidade se relaciona com o Pagamento e LugaresParque.

A entidade Pagamento é identificado pelo seu id e possui dois atributos, a validade e a Taxa a pagar pela reserva. Esta entidade relaciona-se com a Reserva.

A entidade Parque é identificado pelo seu id e possui como atributos a zona(atributo composto), que possui Distrito e Cidade e Capacidade. Esta entidade relaciona-se apenas com a Reserva.

A entidade Cliente é identificado pelo seu id e possui como atributos o nome e a prioridade. Esta Entidade relaciona-se com a entidade Veiculo.

A entidade veículo possui como identificador a matricula e ainda o atributo tipo.Sendo que se relaciona com a entidade Reserva e Cliente.

A entidade LugaresParque é identificada pelo seu id e possui como atributos o piso e ocupado.

4. Modelo Lógico

Após a modelação do problema apresentado foi preciso “traduzi-lo” para o modelo lógico.

De maneira a construir o modelo lógico houve necessidade de analisar o modelo conceptual e as suas entidades, atributos e relacionamentos. As entidades presentes no modelo são as mesmas das do modelo lógico : Veículo,Reserva,Parque,Cliente e Pagamento.

Sendo que existem 2 tipos de relacionamentos dentro do modelo.O primeiro é o de 1-N sendo que a entidade com multiplicidade N vai ter uma chave estrangeira que corresponde à chave primária da entidade de multiplicidade 1.O segundo relacionamento é o de 1-1 sendo que a entidade Reserva vai ficar com a chave estrangeira que irá identificar o pagamento associado a uma reserva.

4.1. Construção e validação do modelo de dados lógico

Para a construção do modelo lógico foi necessário efetuar uma revisão geral no modelo conceptual de modo a verificar todos as entidades,atributos e relacionamentos existentes. A nossa primeira preocupação é criar as relações do modelo lógico que representam todas as entidades, atributos e relacionamentos do modelo conceptual

As entidades presentes no modelo conceptual são as mesmas que figuram no modelo lógico:

Veículo;

Cliente;

Reserva;

Pagamento;

Parque;

LugaresParque;

Além disto, quando analisamos o modelo conceptual definimos e classificamos as entidades e relacionamentos do mesmo. Tendo por base a *Database Definition Language* (DDL):

- Entidades Fortes
- Entidades Fracas
- Relacionamentos binários um para muitos (1-N)
- Relacionamentos binários um para um (1-1)
- Relacionamentos recursivos um para um (1-1)
- Relacionamentos superclasse/subclasse
- Relacionamentos binários muitos para muitos (N-M)
- Relacionamentos complexos
- Atributos multivalor

4.1.1. Entidades Fortes

Resumidamente, uma entidade forte é aquela a qual tem atributos suficientes para criar uma chave primária e não tem dependência com nenhuma outra entidade. Ao analisar o modelo conceptual, concluímos que todas as entidades são entidades fortes e também identificamos as chaves primárias e alternativas de cada entidade. Assim, temos:

- **Veículo** (matrícula, tipo, Cliente_idCliente)
  Chave primária (matrícula)

- **Cliente** (idCliente, nome, prioridade)
  Chave primária (idCliente)
  Chave alternativa (nome)

- **Reserva** (idReserva, data_inicio, data_fim, Veiculo_matricula, Pagamento_idPagamento, LugaresParque_lugar, ativa)
  Chave primária (idReserva)

- **Pagamento** (idPagamento, taxa, preco)
  Chave primária (idPagamento)

- **LugaresParque** (id_lugar, piso, ocupado, Parque_idParque)
  Chave primária (id_lugar)

- **Parque** (idParque, capacidade, distrito, cidade)
  Chave primária (idParque)

4.1.2. Entidades Fracas

No nosso modelo conceptual não identificamos nenhuma entidade fraca, ou seja, nenhuma das entidades deste modelo depende de outra sendo que todas elas fazem sentido isoladas.

4.1.3. Relacionamentos binários um para muitos (1-N)

Neste tipo de relacionamentos, a entidade de multiplicidade N tem um dos atributos que é considerado uma chave estrangeira, que é a chave principal da entidade com multiplicidade 1.

- **Veículo** (matricula, tipo, Cliente_idCliente)
  Chave Primária (matrícula)
  Chave Estrangeira (Cliente_idCliente)

- **LugaresParque** (id_lugar, piso, ocupado, Parque_idParque)
  Chave Primária (id_lugar)
  Chave Estrangeira (Parque_idParque)

- **Reserva** (idReserva, data_inicio, data_fim, Veiculo_matricula, Pagamento_idPagamento, LugaresParque_lugar, ativa)
  Chave Primária (idReserva)
  Chave Estrangeira (Veiculo_matricula)
  Chave Estrangeira (Pagamento_idPagamento)
  Chave Estrangeira (LugaresParque_lugar)

4.1.4. Relacionamentos binários um para um (1-1)

Este tipo de relacionamentos pode ser de participação dos dois lados ou só de um deles. No caso do nosso modelo conceptual, verificamos que existia um relacionamento destes entre as entidades *Reserva* e *Pagamento*. Neste caso a entidade *Reserva* fica com a chave estrangeira *Pagamento_idPagamento* que identifica um pagamento de uma dada reserva.

4.1.5. Relacionamentos recursivos um para um (1-1)

Não identificamos qualquer tipo de relacionamento deste género, com base no nosso modelo conceptual.

4.1.6. Relacionamentos superclasse/subclasse

No nosso modelo conceptual não identificamos a existência de nenhuma classe dominante (superclasse) nem subordinada (subclasse). Como tal, não existe nenhum relacionamento superclasse/subclasse no nosso modelo.

4.1.7. Relacionamentos binários de muitos para muitos (N-M)

De acordo com o nosso modelo conceptual, não existe qualquer relacionamento binário de muitos para muitos (N-M).

4.1.8. Relacionamentos Complexos

Acerca deste tipo de relacionamentos, podemos aferir que nenhum destes se encontra no modelo feito.

4.1.9. Atributos Multivalor

Estes atributos podem assumir diferentes valores para a mesma entidade. No caso em estudo, analisando o modelo conceptual, não se verifica a existência de qualquer atributo com esta particularidade, daí concluirmos que não existem atributos multivalor no nosso modelo.

4.2. Desenho do modelo lógico

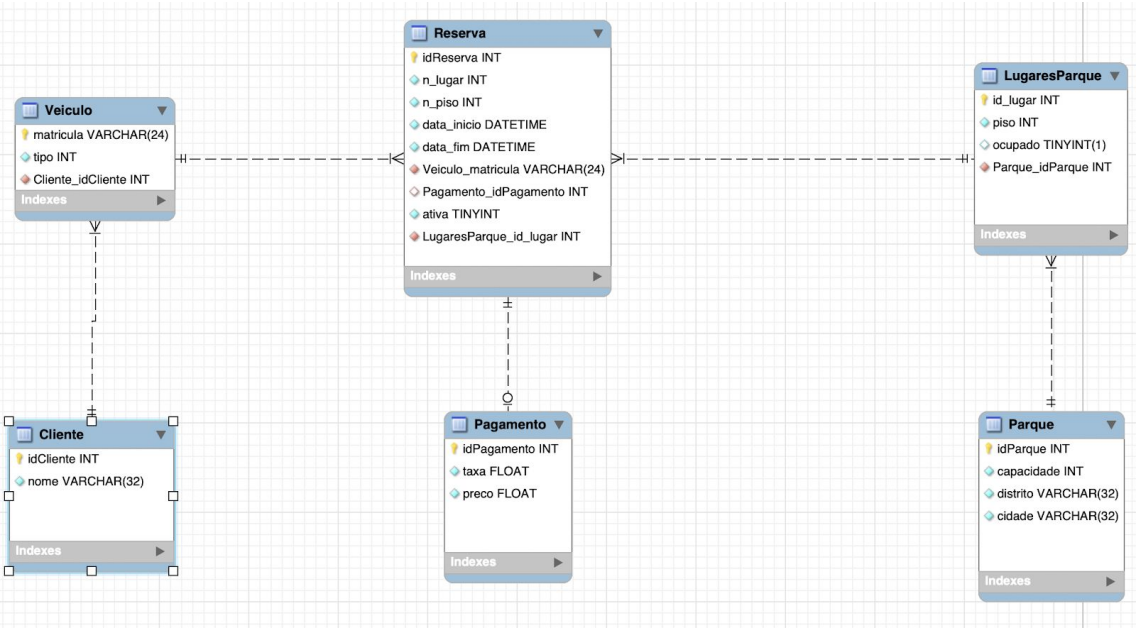

Figura 13 - Modelo Lógico.

4.3. Validação do modelo através da normalização

O objetivo do modelo relacional de uma base de dados é agrupar atributos em relações, por existir uma dependência funcional entre eles. O resultado permite a criação de um modelo lógico estruturalmente consistente. No entanto o modelo normalizado não corresponde, por

vezes, à versão de máxima eficiência. A normalização permite que se perceba melhor cada atributo e o que representa.
Para percebermos melhor esta técnica, exemplificaremos uma dependência funcional entre dois atributos:
Reserva (idReserva, data_inicio, data_fim, Veiculo_matricula, Pagamento_idPagamento, ativa, LugaresParque_ativa)
Como na relação Reserva conseguimos obter o preço da mesma através do Pagamento_idPagamento, que contém a informação do preço da reserva associada, então existe uma dependência funcional entre Pagamento_idPagamento e o preço.

4.3.1. 1FN - 1ª Forma Normal

Uma relação está na 1ª Forma Normal quando não é possível decompor os atributos mais do que já estão. Quando falamos em dividir os atributos, falamos quando há, por exemplo, atributos multivalorados ou atributos compostos.
Assim, como não há nenhum atributo multivalorado ou composto, podemos afirmar que o nosso modelo já se encontra de acordo com a 1ª Forma Normal.

4.3.2. 2FN - 2ª Forma Normal

Uma relação está na 2ª forma normal quando possui uma chave primária complexa e não existem atributos dessa relação apenas dependentes de uma das chaves primárias.
Podemos, assim, concluir que o nosso modelo respeita a 2ª forma normal, pois não existe nenhuma chave primária complexa no modelo.

4.3.3. 3FN - 3ª Forma Normal

Uma relação está na 3ª forma normal quando nenhum dos atributos não-chave pode influenciar outro atributo não-chave. Apesar disto, para ser considerado da 3ª forma normal, o sistema precisa de estar na 2ª forma normal. Como no nosso modelo não existem relações de 2ª forma normal também não existem de 3ª forma normal, consequentemente.
Com isto, podemos concluir que o nosso modelo se encontra de acordo com a 3ª Forma Normal.

4.4. Validação do modelo com interrogações do utilizador

Com o termino do modelo lógico, vem associada a validação do modelo. Assim, é necessário que responda às perguntas que utilizadas para validar o modelo conceptual.

1. Quais foram os veículos que usaram o parque num determinado dia?
2. Quais os parques com lugares livres numa determinada cidade ?
3. Que lugares estão livres num determinado parque a uma dada hora?
4. Quanto faturou a empresa num dado intervalo de tempo?

Passa-se agora a validar o modelo:

1- Quais foram os veículos que usaram o parque num determinado dia?

Para determinar quais os veículos que usaram um parque num determinado dia, precisamos das entidades Reserva, Veículo e Parque.

Tendo a entidade Reserva, sabemos as reservas efetuadas num determinado dia. A entidade Veículo dá-nos a informação dos veículos os quais estão associadas as reservas desse dia. Para completar, a entidade Parque tem a informação do parque que queremos procurar. Assim, o utilizador tem toda a informação que precisa para conseguir obter a sua resposta.

2- Quais os parques com lugares livres numa determinada cidade numa dada data?

Para determinar quais os parques com lugares disponíveis numa determinada cidade precisamos da informação das entidades Parque e LugarParque. A primeira entidade referida fornece-nos a informação dos parques que existem na cidade que o utilizador quer procurar. A última diz-nos se um lugar está ocupado. Assim, com toda a informação que é fornecida, o utilizador consegue encontrar a resposta à sua pergunta.

3- Que lugares estão livres num determinado parque numa dada data?

Para determinar quais os lugares livres de um parque são necessárias as entidades Parque e LugaresParque. A entidade Parque seleciona qual o parque que o utilizador quer procurar e a entidade LugaresParque revela quais os lugares livres no parque que o utilizador quer. Assim, este tem toda a informação para ter a resposta à sua pergunta.

4- Quanto faturou a empresa num dado intervalo de tempo?

Para determinar qual o valor faturado pela empresa num dado intervalo de tempo é necessária a informação das entidades Reserva e Pagamentos. A primeira entidade fornece-nos a informação das reservas feitas num determinado intervalo de tempo. Por fim, a entidade tem informação de todos os pagamentos feitos em todas as reservas. Assim, o utilizador terá toda a informação necessária para ter a sua resposta.

4.5. Validação do modelo com as transações estabelecidas

Como estabelecido com o cliente é necessário verificar se o modelo suporta as seguintes transações:

- Inserir um lugar num parque : Para executar este processo é necessário aumentar a capacidade do parque e inserir um novo lugar.
- Dar entrada de uma reserva, check-in: Para executar este processo tem de existir uma reserva feita e apenas é necessário marcar o lugar da reserva como ocupado.
- Gerar um pagamento associado a uma determinada reserva: Para tal o cliente tem de efetuar uma reserva ao que a base de dados irá automaticamente gerar um pagamento da mesma.

4.6. Reavaliação do modelo lógico

Em relação à reavaliação do nosso modelo lógico, não há necessidade de o fazer, pois o nosso modelo já estava normalizado.

4.7. Revisão do modelo lógico com o utilizador

Nesta fase final do modelo lógico surgiu a necessidade de realizar uma revisão do modelo com a finalidade de verificar a integridade da estrutura deste modelo.
Assim, após essa mesma revisão, o modelo foi aprovado, pois este verifica todos os requisitos desejados pelo utilizador.

5. Modelo Físico

5.1 Seleção do sistema de gestão de bases de dados

Após a aceitação do modelo lógico, avançou-se com a tradução para o modelo físico, usando um sistema de gestão de base de dados.

Para este projeto foi decidido utilizar o sistema de gestão de base de dados o MYSQL que utiliza o motor InnoDB.As razões para esta escolha deveram-se principalmente à sua fiabilidade, alto desempenho e nível de segurança.

5.2 Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

Relações Base

Relação Cliente

Domínio idCliente	Inteiro
Domínio nome	String com tamanho até 32
Domínio prioridade	Boolean

Cliente(

idCliente	Identificador de um cliente	NOT NULL,
nome	Indica o nome de um cliente	NOT NULL,

PRIMARYKEY(IdCliente));

```
CREATE TABLE IF NOT EXISTS `parking`.`Cliente` (
  `idCliente` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`idCliente`))
ENGINE = InnoDB;
```

Figura 14 - Tabela Cliente.

Relação Veiculo

Domínio matricula	String com tamanho até 32
Domínio tipo	Inteiro
Cliente_idCliente	Inteiro

Veiculo(

matricula	Identificador de um veículo	NOT NULL,
tipo	Combustível dos veiculos	NOT NULL,
Cliente_idCliente	identificador do dono do veículo	NOT NULL,

PRIMARYKEY(matricula));

```
CREATE TABLE IF NOT EXISTS `parking`.`Veiculo` (
  `matricula` VARCHAR(24) NOT NULL,
  `tipo` INT NOT NULL,
  `Cliente_idCliente` INT NOT NULL,
  PRIMARY KEY (`matricula`),
  INDEX `fk_Veiculo_Cliente1_idx` (`Cliente_idCliente` ASC) VISIBLE,
  CONSTRAINT `fk_Veiculo_Cliente1`
    FOREIGN KEY (`Cliente_idCliente`)
    REFERENCES `parking`.`Cliente` (`idCliente`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Figura 15 - Tabela Veículo.

Relação Pagamento

Domínio id_Pagamento	Inteiro
Domínio taxa	Float
Domínio preco	Float

Pagamento(

id_Pagamento	Identificador de um pagamento	NOT NULL,

taxa	Valor aplicado a cada tipo veículo	NOT NULL,
preco	Valor final da reserva	NOT NULL,
PRIMARYKEY(id_pagamento))		

```
CREATE TABLE IF NOT EXISTS `parking`.`Pagamento` (
  `idPagamento` INT NOT NULL AUTO_INCREMENT,
  `taxa` FLOAT NOT NULL,
  `preco` FLOAT NOT NULL,
  PRIMARY KEY (`idPagamento`))
ENGINE = InnoDB;
```

Figura 16 - Tabela Pagamento.

Relação Parque

Dominio idParque	Inteiro
Dominio capacidade	Inteiro
Dominio distrito	String com tamanho até 32
Dominio cidade	String com tamanho até 32

Parque(		
idParque	Identificador de um parque	NOT NULL,
capacidade	nºde lugares de um parque	NOT NULL,
distrito	distrito onde se localiza	NOT NULL,
cidade	cidade onde se localiza	NOT NULL,
PRIMARYKEY(idParque))		

```
CREATE TABLE IF NOT EXISTS `parking`.`Parque` (
  `idParque` INT NOT NULL AUTO_INCREMENT,
  `capacidade` INT NOT NULL,
  `distrito` VARCHAR(32) NOT NULL,
  `cidade` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`idParque`))
ENGINE = InnoDB;
```

Figura 17 - Tabela Parque.

Relação LugaresParque

Dominio id_lugar	Inteiro
Dominio piso	Inteiro
Dominio ocupado	BOOLEAN
Dominio Parque_idParque	Inteiro

LugaresParque(

id_lugar	Identificador de um parque	NOT NULL,
piso	piso onde se encontra o lugar	NOT NULL,
ocupado	estado do lugar em questão	NOT NULL,
Parque_idParque	id do parque a que pertence	NOT NULL,

PRIMARYKEY(id_lugar))

```
CREATE TABLE IF NOT EXISTS `parking`.`LugaresParque` (
  `id_lugar` INT NOT NULL,
  `piso` INT NOT NULL,
  `ocupado` TINYINT(1) NULL,
  `Parque_idParque` INT NOT NULL,
  INDEX `fk_LugaresPaque_Parque1_idx` (`Parque_idParque` ASC) VISIBLE,
  PRIMARY KEY (`id_lugar`),
  CONSTRAINT `fk_LugaresParque_Parque1`
    FOREIGN KEY (`Parque_idParque`)
    REFERENCES `parking`.`Parque` (`idParque`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Figura 18 -Tabela LugaresParque.

Relação Reserva

Dominio idReserva	Inteiro
Dominio data_inicio	DATETIME
Dominio data_fim	DATETIME
Dominio Veiculo_matricula	String com tammanho 24
Dominio Pagamento_idPagamento	Inteiro
Dominio ativa	BOOLEAN
Dominio LugaresParque_id_lugar	Inteiro

Reserva(

idReserva	Identificador de um parque	NOT NULL,
data_inicio	data de início de uma reserva	NOT NULL,
data_fim	data de término de uma reserva	NOT NULL,
Veiculo_matricula	identificador da viatura que efetua reserva	NOT NULL,
Pagamento_idPagamento	identificador do pagamento da reserva	NULL,
ativa	indica se está ativa ou arquivada	NULL,
LugaresParque_id_lugar	indica o lugar reservado	NULL,

PRIMARYKEY(idReserva))

```
CREATE TABLE IF NOT EXISTS `parking`.`Reserva` (
  `idReserva` INT NOT NULL,
  `data_inicio` DATETIME NOT NULL,
  `data_fim` DATETIME NOT NULL,
  `Veiculo_matricula` VARCHAR(24) NOT NULL,
  `Pagamento_idPagamento` INT NULL,
  `ativa` TINYINT NOT NULL,
  `LugaresParque_id_lugar` INT NOT NULL,
  PRIMARY KEY (`idReserva`),
  INDEX `fk_Reserva_Pagamento_idx` (`Pagamento_idPagamento` ASC) VISIBLE,
  INDEX `fk_Reserva_Veiculo1_idx` (`Veiculo_matricula` ASC) VISIBLE,
  INDEX `fk_Reserva_LugaresParque1_idx` (`LugaresParque_id_lugar` ASC) VISIBLE,
  CONSTRAINT `fk_Reserva_Pagamento`
    FOREIGN KEY (`Pagamento_idPagamento`)
    REFERENCES `parking`.`Pagamento` (`idPagamento`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Reserva_Veiculo1`
    FOREIGN KEY (`Veiculo_matricula`)
    REFERENCES `parking`.`Veiculo` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Reserva_LugaresParque1`
    FOREIGN KEY (`LugaresParque_id_lugar`)
    REFERENCES `parking`.`LugaresParque` (`id_lugar`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 19 - Tabela Reserva.

5.3 Tradução das interrogações do utilizador para SQL

Quais foram os veículos que usaram o parque num determinado dia?

Para a resolução desta interrogação foi necessário recolher informações das tabelas Veiculo, Reserva e Cliente.Sendo que retiramos informação acerca da matricula e nome da junção que é feita das três tabelas garantindo que estamos a limitar o intervalo de tempo com o input do procedure.Em seguida está explicitada a resolução da query em MYSQL.

```
USE parking;

-- QUERY 1 : Quais foram os veículos que usaram o parque num determinado dia

DELIMITER $$

CREATE PROCEDURE veiculos (IN d DATETIME)
BEGIN
   SELECT matricula,nome
   FROM Veiculo
   INNER JOIN Reserva
   ON Reserva.Veiculo_matricula = Veiculo.matricula
   INNER JOIN Cliente
   ON Cliente.idCliente=Veiculo.Cliente_idCliente
   WHERE d BETWEEN Reserva. data_inicio AND Reserva.data_fim;


END $$

CALL veiculos('2018-12-25 22:23:00');
DROP PROCEDURE veiculos;
```

Figura 20 - Query 1.

Quais os parques com lugares livres num determinado distrito a uma dada hora ?

Para a resolução desta interrogação foi necessário recolher informações das tabelas Parque e LugaresParque, no processo foi tido em consideração que um lugar que esteja reservado para a data do input mas ainda não esteja ocupado,ou seja, o check-in ainda não foi efetuado, é considerado um lugar ocupado.Recolheu -se apenas informação sobre os id dos Parques.Em seguida está explicitada a resolução da query em MYSQL.

```
-- QUERY 2 : Quais os parques com lugares liveres reserváveis num determinado distrito ?

DELIMITER $$

CREATE PROCEDURE parques_livres_zona (IN dist VARCHAR(16), IN cur_date DATETIME)
BEGIN

   SELECT DISTINCT idParque FROM Parque

   INNER JOIN(
      SELECT Parque_idParque
      FROM LugaresParque
      WHERE(LugaresParque.id_lugar NOT IN
         (SELECT id_lugar
         FROM LugaresParque
         INNER JOIN Reserva ON Reserva.LugaresParque_lugar = LugaresParque.id_lugar
         WHERE(NOW() BETWEEN Reserva.data_inicio AND Reserva.data_fim) && (Reserva.ativa = 1)))) AS tab_livres
   ON Parque.idParque = tab_livres.Parque_idParque
   WHERE Parque.distrito = dist;

END $$

CALL parques_livres_zona('Braga',NOW());
DROP PROCEDURE parques_livres_zona;
```

Figura 21 - Query 2.

Que lugares estão livres num determinado parque a uma dada hora?

Para a resolução desta interrogação foi necessário recolher informações das tabelas Reserva e LugaresParque, no processo usou-se o mesmo raciocínio usado na query 2, sendo que selecionamos os lugares e o seu piso.Em seguida está explicitada a resolução da query em MYSQL.

```
-- QUERY 3: Que lugares estão livres num determinado parque,  ?

DELIMITER $$

CREATE PROCEDURE lugares_livres_Parque(IN p_id INT, IN cur_date DATETIME)
BEGIN

    SELECT id_lugar, piso
    FROM LugaresParque
    WHERE(LugaresParque.Parque_idParque = p_id AND LugaresParque.id_lugar NOT IN
        (SELECT id_lugar
         FROM LugaresParque
         INNER JOIN Reserva
         ON Reserva.LugaresParque_lugar = LugaresParque.id_lugar
         WHERE(cur_date BETWEEN Reserva.data_inicio AND Reserva.data_fim) &&
             (Reserva.ativa = 1)));

END $$

CALL lugares_livres_Parque(1,NOW());
DROP PROCEDURE lugares_livres_Parque;
```

Figura 22 - Query 3.

Quanto faturou a empresa num dado intervalo de tempo?

Para a resolução desta interrogação foi necessário recolher informações das tabelas Reserva e Pagamento, de maneira a que as informações estivessem compreendidas no intervalo de tempo do input e fazendo a correspondência entre o id dos pagamentos e o id das Reservas, fazendo uma soma de todos estes valores.Em seguida está explicitada a resolução da query em MYSQL.

```
-- QUERY 4: Quanto faturou a empresa num dado intervalo de tempo?

DELIMITER $$

CREATE PROCEDURE lucro_intervalado(IN d_inicio DATETIME, IN d_fim DATETIME)
BEGIN
    SELECT SUM(preco)
    FROM Reserva
    INNER JOIN Pagamento
    ON Reserva.Pagamento_idPagamento = Pagamento.idPagamento
    WHERE Reserva.data_inicio BETWEEN d_inicio AND d_fim;

END $$

CALL lucro_intervalado('2018-11-25 22:20:00','2018-11-26 22:00:00');
DROP PROCEDURE lucro_intervalado;
```

Figura 23 - Query 4.

5.4 Tradução das transações estabelecidas para SQL

De maneira a aumentar a *performance* do sistema de gestão de base de dados a ser desenvolvido,houve necessidade de analisar todas as transações/queries que ocorrem no modelo desenvolvido.

De seguida estão traduzidas todas essas transações em código MYSQL.

```
-- PROCEDURE: INSERIR LUGAR NUM PARQUE --

DELIMITER $$

CREATE PROCEDURE inserirLugar (
    IN flr INT,
    IN lug INT,
    IN p_ID INT)

BEGIN
    DECLARE Erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
    START TRANSACTION;

INSERT INTO LugaresParque (id_lugar, piso, ocupado, Parque_idParque)
        VALUES (lug, flr, 0, p_ID);

IF Erro
THEN ROLLBACK;
ELSE COMMIT;
END IF;
END $$
```

Figura 24 - Aumentar um lugar num parque de estacionamento .

```
DELIMITER $$

CREATE PROCEDURE checkIN (IN reserva_id INT)

BEGIN
    DECLARE lugar_id INT;
    SET lugar_id = (SELECT LugaresParque_lugar FROM Reserva WHERE Reserva.idReserva=reserva_id);

    UPDATE LugaresParque
    SET ocupado = 1
    WHERE id_lugar = lugar_id;
END$$
```

Figura 25 - Processo de dar entrada de uma reserva.

Verificou-se a necessidade de criar um trigger para que após a inserção de uma reserva seja gerado um registo do valor a pagar pela estadia do carro,tendo em conta o tipo de veículo e o tempo que o mesmo esteve no parque.

```
DELIMITER ||

CREATE TRIGGER geraPagamentoCliente
AFTER INSERT ON Reserva
FOR EACH ROW
BEGIN
    DECLARE tipo_veiculo INT;
    DECLARE pagamento_id INT;
    DECLARE tempo FLOAT;
    DECLARE tx_1 FLOAT;
    DECLARE tx_2 FLOAT;

    SET tx_1 = 1.2;
    SET tx_2 = 1.5;
    SET tipo_veiculo = tipoveiculo(NEW.idReserva);
    SET tempo = tempoveiculo(NEW.idReserva);

    IF (tipo_veiculo = 1) THEN
        INSERT INTO Pagamento (idPagamento,taxa, preco, multa)
            VALUES (idPagamento, tx_1-1, tx_1*0.1*tempo, 0);
    ELSE
        INSERT INTO Pagamento (idPagamento,taxa, preco, multa)
            VALUES (idPagamento, tx_2-1, tx_2*0.1*tempo, 0);
    END IF;
END ||

DELIMITER $$
```

Figura 26 - Trigger para gerar pagamento

De maneira a permitir que o parque seja aumentado na sua capacidade, como pedido pelo cliente criou-se o seguinte trigger.

```
CREATE TRIGGER updateCapacidade
AFTER INSERT ON LugaresParque
FOR EACH ROW
BEGIN
    UPDATE Parque
    SET capacidade = capacidade + 1
    WHERE Parque.idParque = NEW.Parque_idParque;
END  $$
```

Figura 26 - Trigger para efetuar update na capacidade

5.5 Escolha, definição e caracterização de índices em SQL

Após processo de pesquisa chegou-se à conclusão que índices são uma estrutura de dados que aumentam a velocidade de operações numa tabela.No *MYSQL* os indices são guardados em *B-Trees*, sendo que todas as tabelas *InnoDB* tem um índice especial que é denominado por *clustered index* no qual os registos são armazenados, ou seja, este índice é a chave primária da tabela.Para além deste índice é também utilizado um índice especial para otimizar as operações de *Data Manipulation Language*, entre elas *SELECT,INSERT,UPDATE,DELETE,...*

Quando se define uma chave primária, o *InnoDB* utiliza-a como *clustered index*, o que significa que são definidos tantos *clustered index* quantas chaves primárias para a tabela em causa.

5.6 Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Tabela 4 - Tamanho da Tabela Veículo

Atributo	Tipo de dados	Espaço ocupado
matricula	CHAR (24)	24 bytes
tipo	Inteiro	4 bytes
Cliente_idCliente	Inteiro	4 bytes
Total		nº de veiculos * 32 bytes

Tabela 5 - Tamanho da Tabela Cliente

Atributo	Tipo de dados	Espaço ocupado
idCliente	Inteiro	4 bytes
nome	CHAR(32)	32 bytes
Total		nºclientes * 36 bytes

Tabela 6 - Tamanho da Tabela Reserva

Atributo	Tipo de dados	Espaço ocupado
idReserva	Inteiro	4 bytes
LugaresParque_lugar	Inteiro	4 bytes
data_inicio	DATETIME	8 bytes
data_fim	DATETIME	8 bytes
Veiculo_matricula	CHAR(24)	24 bytes
Pagamento_idPagamento	Inteiro	4 bytes
ativa	BOOLEAN	1 byte
Total		nºreservas*53 bytes

Tabela 7 - Tamanho da Tabela Pagamento

Atributo	Tipo de dados	Espaço ocupado
idPagamento	Inteiro	4 bytes
taxa	Float	4 bytes
preco	Float	4 bytes
multa	Float	4 bytes
Total		nºpagamentos *16 bytes

Tabela 8 - Tamanho da Tabela LugaresParque

Atributo	Tipo de dados	Espaço ocupado
id_lugar	Inteiro	4 bytes
piso	Inteiro	4 bytes
ocupado	BOOLEAN	1 byte
Parque_idParque	Inteiro	4 bytes
Total		nºlugares* 13 bytes

Tabela 9 - Tamanho da tabela Parque

Atributo	Tipo de dados	Espaço ocupado
idParque	Inteiro	4 bytes
capacidade	Inteiro	4 bytes
distrito	CHAR(32)	32 bytes
cidade	CHAR(32)	32 bytes
Total		nº parques*72 bytes

5.7 Definição e caracterização das vistas de utilização em SQL

Aquando do desenvolvimento de um SGBD, as vistas de cada um dos utilizadores, ou seja, os acessos e as ações que cada um pode ter, são algo de grande importância.

Uma vista pode ser considerada como uma tabela, por vezes designada de tabela virtual, uma vez que as colunas destas vistas são colunas presentes em outras tabelas do nosso SGBD. As vistas permitem simplificar e evitar escrever consultas frequentes, escondendo também certos detalhes que não têm interesse ou não dizem respeito a um dado utilizador.

Consideremos, portanto, algumas vistas do nosso modelo físico:

```sql
-- VIEW COM AS RESERVAS DE CADA CLIENTE --

CREATE VIEW r_Cliente AS
SELECT idCliente AS 'ID Cliente',
        nome AS 'Nome do Cliente',
        veic.matricula AS 'Matrícula',
        veic.tipo AS 'Tipo de Veículo',
        res.idReserva AS 'ID Reserva',
        res.data_inicio AS 'Data de Entrada',
        res.data_fim AS 'Data de Saída',
        pag.preco AS 'Preço',
        pag.taxa AS 'Taxa',
        pag.multa AS 'Multa'
FROM Cliente
INNER JOIN Veiculo AS veic ON Cliente.idCliente = veic.Cliente_idCliente
INNER JOIN Reserva AS res ON veic.matricula = res.Veiculo_matricula
INNER JOIN Pagamento As pag ON res.Pagamento_idPagamento = pag.idPagamento
ORDER BY idCliente;
```

Figura 27 - View reservas de cada cliente

Figura 28 - View reservas dos parques

```sql
-- VIEW COM AS INFORMAÇOES DOS LUGARES LIVRES DOS PARQUES --
CREATE VIEW v_Lugares AS
SELECT  Parque_idParque AS 'ID Parque',
                id_lugar AS 'Lugar',
                piso AS 'Piso',
                park.distrito AS 'Distrito',
                park.cidade AS 'Cidade'
FROM LugaresParque
INNER JOIN Parque AS park ON LugaresParque.Parque_idParque = park.idParque
WHERE (LugaresParque.ocupado = 0)
ORDER BY Parque_idParque;
```

5.8 Definição e caracterização dos mecanismos de segurança em SQL

Nesta fase fala-se sobre as medidas tomadas para garantir o bom funcionamento, segurança e integridade da base de dados.

As medidas tomadas no projeto foram a criação de utilizadores para garantir a segurança dos dados e a utilização de transações para garantir a integridade da base de dados.

Dentro dos utilizadores identificou-se dois tipos, o administrador e o cliente.

O administrador tem permissão para realizar qualquer ação (*CREATE,DROP,SELECT,DELETE,INSERT,UPDATE*) e, por isso, tem as seguintes permissões/regras de acesso:

```
-- ADMIN

CREATE USER 'administrador' @'localhost' IDENTIFIED BY 'administrador';
GRANT ALL PRIVILEGES ON parking.* TO 'administrador' @'localhost';
```

O cliente tem possibilidade de consultar todos os lugares disponíveis, consultar todos os parques existentes e ter informações da sua localização, assim como adicionar registos de cliente e Veículo.

```
CREATE USER 'cliente' @'localhost' IDENTIFIED BY 'cliente';
GRANT SELECT ON parking.LugaresParque TO 'cliente' @ 'localhost';
GRANT SELECT (distrito,cidade) ON parking.Parque TO 'cliente' @ 'localhost';
GRANT INSERT ON parking.Veiculo TO 'cliente' @ 'localhost';
GRANT INSERT ON parking.Cliente TO 'cliente' @ 'localhost';
REVOKE DROP,DELETE,UPDATE ON parking.* FROM 'cliente'@'cliente';
```

5.9 Revisão do sistema implementado com o utilizador

Reuniu-se com o cliente onde foram testadas as funcionalidades de toda a implementação física com a verificação da resposta às interrogações e ainda a estimativa quanto ao espaço físico que a base de dados irá ocupar.

O Cliente mostrou-se agradado com o projecto desenvolvido, apontando que espera um bom crescimento da sua empresa assim como futuras implementações.

Assim sendo, obteve-se a aprovação do Cliente.

6. Ferramentas utilizadas

Para a elaboração deste projeto recorreu-se a algumas ferramentas entre elas:

- Word Online: Elaboração do relatório apresentado;
- TerraER: Desenho do modelo conceptual;
- MySQL Workbench: Desenvolvimento do modelo lógico e físico.

7. Conclusão

O desenvolvimento de um SGBD é uma tarefa que requer diferentes cuidados ao longo da mesma. Dessa forma, ao longo do nosso percurso, cada passo dado em prol desse mesmo desenvolvimento, teve sempre como objetivo a melhor performance possível para o sistema, assim como os requisitos do sistema, analisados junto do cliente. tendo sido definidas entidades e relacionamentos de forma a otimizar a gerência e armazenamento dos dados em causa. Além disso, e a favor da minimização de erros de organização foi seguida a metodologia sugerida pela livro Database Systems- Thomas Connolly.Carolyn Begg.

Assim sendo, numa fase inicial do projeto foi elaborado um modelo conceptual, com proximidade ao Cliente, onde este consentiu com o modelo desenvolvido. Posteriormente, foram elaborados os modelos lógico e esquema físico, novamente validados pelo Cliente.

A base de dados em questão foi marcada como um sistema simples, mas não muito simples, do pretendido, uma vez que se mostra flexível, no caso do Cliente querer adotar outras áreas de funcionamento da empresa, que o utilizador deseje adquirir, em caso de expansão.

Em conclusão, e tendo em conta os resultados em mão, podemos comprovar que os objetivos foram concluídos, pois conseguiu-se elaborar uma base de dados funcional e representativa dos requisitos da leiParking, cumprindo qualquer requisito imposto pelo Cliente.

8. Referências

Thomas M. Connolly, Carolyn E. Begg - Database Systems: A Practical Approach to Design, Implementation and Management - 4th Edition.

9. Lista de Siglas e Acrónimos

BD Base de Dados

ER Entidade-Relacionamento

SGBD Sistema de Gestão de Base de Dados