



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Parque Aquático com Nome Bonito

**André Gonçalves (a80368), Diogo Gonçalves
(a81860), Luís Alves(a80165) e Rafaela Rodrigues
(a80516)**

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Parque Aquático com Nome Bonito

**André Gonçalves (a80368), Diogo Gonçalves
(a81860), Luís Alves (a80165), Rafaela Rodrigues
(a80516)**

Resumo

O trabalho realizado consistiu no desenvolvimento e documentação de um Sistema de Gestão de Base de Dados para um Parque Aquático que pretende obter dados estatísticos fidedignos relativamente aos seus utilizadores.

Numa primeira fase foi clarificado o caso de estudo e, a partir daí, definidos os objetivos que pretendiam ser atingidos com o desenvolvimento do sistema.

De seguida, foram levantados os requisitos do sistema com base em reuniões com o Conselho de Administração do Parque Aquático, tendo sido produzida uma análise que permitiu o desenvolvimento do modelo conceptual de dados.

O modelo conceptual englobou a identificação e caracterização das entidades, dos relacionamentos e dos atributos necessários para representar fielmente o sistema. Findos esses passos, foi produzido um Diagrama ER que sintetiza toda essa análise.

Posteriormente foi feita a transição do modelo conceptual para o modelo lógico, tendo cada tabela sido validada através da normalização até à 3ª forma. Para além disso, também se verificou a capacidade do modelo de suportar as transações e interrogações produzidas pela análise de requisitos.

De seguida, o modelo lógico foi traduzido para um conjunto de instruções SQL que permitem definir a base de dados final. Foram também traduzidas para SQL as transações e interrogações suportadas no modelo lógico, e efetuada uma análise quanto ao espaço ocupado pela base de dados e a sua taxa de crescimento anual.

Por fim, foram definidos perfis de utilização da base de dados bem como caracterizados mecanismos de segurança da mesma.

Área de Aplicação: Desenho e Arquitetura de Sistemas de Bases de Dados

Palavras-Chave: Bases de Dados Relacionais, Diagramas ER, MySQL, MySQL Workbench, Transações, Modelação Lógica, Modelação Conceptual

Índice

Resumo	i
Índice	ii
Índice de Figuras	iv
Índice de Tabelas	vi
1. Definição do Sistema	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	2
1.4. Análise da viabilidade do processo	2
2. Levantamento e análise de requisitos	3
2.1. Método de levantamento e de análise de requisitos adotado	3
2.2. Requisitos levantados	3
2.2.1 Requisitos de descrição	3
2.2.2 Requisitos de exploração	4
2.2.3 Requisitos de controlo	4
2.3. Análise geral dos requisitos	5
3. Modelação Conceptual	7
3.1. Apresentação da abordagem de modelação realizada	7
3.2. Identificação e caracterização das entidades	7
3.3. Identificação e caracterização dos relacionamentos	8
3.4. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos	8
3.5. Detalhe ou generalização de entidades	11
3.6. Apresentação e explicação do diagrama ER	12
3.7. Validação do modelo de dados com o utilizador	12
4. Modelação Lógica	13
4.1. Construção e validação do modelo de dados lógico	13
4.1.1 Entidades fortes e fracas	13
4.1.2 Relações binárias	13
4.2. Desenho do modelo lógico	15
4.3. Validação do modelo através da normalização	16

4.4. Validação do modelo com interrogações do utilizador	16
4.5. Validação do modelo com as transações estabelecidas	18
4.6. Revisão do modelo lógico com o utilizador	20
5. Implementação Física	21
5.1. Seleção do sistema de gestão de bases de dados	21
5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	21
5.3. Tradução das interrogações do utilizador para SQL	25
5.4. Tradução das transações estabelecidas para SQL	25
5.5. Escolha, definição e caracterização de índices em SQL	27
5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual	27
5.7. Definição e caracterização das vistas de utilização em SQL	29
5.8. Definição e caracterização dos mecanismos de segurança em SQL	30
5.9. Revisão do sistema implementado com o utilizador	31
6. Abordagem <i>NoSQL</i>	32
6.1. Introdução e objetivos	32
6.2. Estrutura base do sistema	32
6.3. Migração dos dados	33
6.4. Tradução das interrogações para <i>Cypher</i>	34
6.5. Conclusões sobre a abordagem <i>NoSQL</i>	34
7. Conclusões e Trabalho Futuro	36
Referências	37
Lista de Siglas e Acrónimos	38
Anexos	39
I. Anexo 1 – Script completo de Criação	40
II. Anexo 2 – Interrogações em SQL	44
III. Anexo 3 – Interrogações em <i>Cypher</i>	50

Índice de Figuras

Figura 1 – Diagrama ER	12
Figura 2 – Diagrama do Modelo Lógico	15
Figura 3 - Transação Visitar uma atração	18
Figura 4 - Transação Registrar um Utilizador	18
Figura 5 - Transação Registrar um Funcionário	19
Figura 6 - Transação Registrar uma Atração	19
Figura 7 - Transação Registrar um turno	19
Figura 8 - Transação Registrar saída do Parque	20
Figura 9 – Criação do esquema ParqueAquatico	21
Figura 10 – Criação da tabela Categoria	22
Figura 11 – Criação da tabela Atração	22
Figura 12 – Criação da tabela Funcionário	22
Figura 13 - Criação da tabela Utilizador	23
Figura 14 – Criação da tabela e_visitada_por	24
Figura 15 – Criação da tabela trabalha_em	24
Figura 16 - Transação Visitar uma atração em SQL	25
Figura 17 – Trigger para incrementar o número de Atrações Visitadas	25
Figura 18 - Transação Registrar um Utilizador em SQL	25
Figura 19 - Transação Registrar um Funcionário em SQL	26
Figura 20 - Transação Registrar uma Atração em SQL	26
Figura 21 - Transação Registrar um turno em SQL	26
Figura 22 - Transação Registrar saída do Parque SQL	26
Figura 23 - Vista do estado da fila de espera das diferentes atrações	29
Figura 24 - Vista do nº de utilizadores de cada categoria	30
Figura 25 - Vista do Top 5 de atrações mais visitadas	30
Figura 26 - Privilégios de acordo com o perfil	30
Figura 27 - Criação de um user Funcionário	31
Figura 28 – Instruções <i>MySQL</i> para exportação do povoamento da base de dados	33
Figura 29 – Carregamento da tabela Utilizador	33
Figura 30 - Carregamento da tabela Categoria	33

Figura 31 - Carregamento da tabela Funcionário	33
Figura 32 - Carregamento da tabela Atração	34
Figura 33 - Carregamento da relação entre Utilizador e Categoria	34
Figura 34 - Carregamento da relação entre Utilizador e Atração	34
Figura 35 - Carregamento da relação entre Funcionário e Atração	34

Índice de Tabelas

Tabela 1 – Principais vistas de utilização do sistema	6
Tabela 2 - Caracterização das entidades	8
Tabela 3 – Caracterização dos relacionamentos	8
Tabela 4 – Caracterização dos atributos das entidades	10
Tabela 5 – Caracterização dos atributos dos relacionamentos	11
Tabela 6 – Generalização das entidades Funcionário e Utilizador	12
Tabela 7 - Relacionamento entre Categoria e Utilizador	14
Tabela 8 - Relacionamento trabalha_em	14
Tabela 9 - Relacionamento e_visitada_por	15
Tabela 10 - Tamanho máximo por entrada da tabela Categoria	27
Tabela 11 - Tamanho máximo por entrada da tabela Utilizador	27
Tabela 12 - Tamanho máximo por entrada da tabela e_visitada_por	28
Tabela 13 - Tamanho máximo por entrada da tabela Atracao	28
Tabela 14 - Tamanho máximo por entrada da tabela trabalha_em	28
Tabela 15 - Tamanho máximo por entrada da tabela Funcionário.	28
Tabela 16 - Tamanho máximo de cada entrada por Tabela.	28
Tabela 17 - Tamanho total da BD com informação de 1 ano.	29

1. Definição do Sistema

1.1. Contextualização

Este Sistema de Gestão de Base de Dados foi desenvolvido no âmbito da Unidade Curricular de Base de Dados, inserida no 3º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho, por um grupo de 4 alunos. Teve como Caso de Estudo, um parque aquático que não existe na realidade, mas tomado como real em todas as secções que se seguem.

1.2. Apresentação do Caso de Estudo

O *Parque Aquático com Nome Bonito* localiza-se na região do Algarve, em Portugal. O Parque encontra-se em funcionamento há 10 anos e é o segundo mais popular no país, existindo 5 parques concorrentes na região. Está aberto desde o primeiro dia do mês de maio até ao último dia do mês de outubro anualmente, desde as 9h até às 18h.

Pode usufruir do Parque qualquer pessoa, estando o preço fixado de acordo com a idade da mesma.

Possui atualmente 13 atrações diferentes, entre as quais: *Big Wow*, *Ultra Splash*, *35 and still falling*. A mais popular é a *Big Wow*.

Desde há 5 anos que conta com um fluxo de visitantes superior a 1 milhão anual, situando-se ligeiramente abaixo do concorrente *Parque Aquático com Nome Feio*, que conta com um fluxo superior a 1.2 milhões de visitantes anualmente. Por isso, o Conselho de Administração está à procura de implementar um sistema de informação mais robusto e completo que permita recolher dados mais concretos relativamente ao uso do Parque por parte dos seus utilizadores, de forma a poder colocar-se à frente da concorrência e melhorar as suas perspetivas financeiras futuras.

De maneira a poder alimentar este novo sistema de informação, o Conselho de Administração decidiu colocar sensores à entrada de cada atração do Parque para poder monitorizar os instantes em que os utilizadores entram na fila para uma determinada atração, e os instantes em que começarão a usufruir dela. Para isso, cada utilizador receberá uma pulseira que terá necessariamente de validar em cada um destes momentos, de forma a habilitar a sua entrada na respetiva atração. Estas pulseiras são atribuídas à entrada do

Parque, e permitem também armazenar a chave de um cacifo onde os utilizadores poderão guardar os seus pertences. Deve ser devolvida à saída do Parque, sob a pena de não ser devolvida a caução paga à entrada relativa à pulseira.

1.3. Motivação e Objetivos

O *Parque Aquático com Nome Bonito* pretende encontrar uma forma de otimizar os seus recursos físicos e assim posicionar-se à frente da concorrência com o retorno financeiro que este sistema lhe proporcionará. A ineficiência na alocação de funcionários às diversas atrações, a falta de informação sobre as atrações que apelam a determinadas faixas etárias e o desconhecimento dos tempos de espera em cada atração por altura do dia são os principais problemas que motivaram o desenvolvimento deste sistema informatizado.

Tendo isto em conta, pretendem-se atingir os seguintes objetivos: criação de novas atividades publicitárias dirigidas a faixas etárias específicas, tendo por base a análise dos dados recolhidos; distribuição eficiente dos funcionários pelas diversas atrações de forma a garantir o bom funcionamento do parque e uma alocação economicamente mais sustentável dos recursos.

1.4. Análise da viabilidade do processo

O desenvolvimento deste Sistema de Gestão de Base de Dados é de considerável importância para o Parque Aquático, uma vez que permite a recolha e tratamento de dados essenciais para o desenvolvimento de novos planos estratégicos para o crescimento sustentável do Parque. Apesar do custo de planificação, desenvolvimento, implementação e manutenção do sistema ser também ele considerável, este será rapidamente colmatado com os ganhos operacionais provenientes do processamento e análise dos dados recolhidos relativos ao funcionamento do Parque. Para além disso, a implementação deste sistema poderá provar-se uma vantagem competitiva face aos parques aquáticos concorrentes na região, que não possuem um sistema similar.

2. Levantamento e análise de requisitos

2.1. Método de levantamento e de análise de requisitos adotado

Optamos por utilizar um método misto de levantamento e análise de requisitos, uma vez que consideramos relevante começar por descrever o mais detalhadamente o sistema através da recolha de requisitos junto do Conselho de Administração do Parque, e depois agrupar essa informação de acordo com os diversos atores mencionados nos requisitos e com os perfis de utilização do sistema a desenvolver.

2.2. Requisitos levantados

2.2.1 Requisitos de descrição

Os requisitos de descrição são os seguintes:

1. Uma atração tem uma duração fixa, uma capacidade máxima de utilizadores, uma altura mínima, uma designação, uma fila de espera de utilizadores, um conjunto de funcionários que a monitorizam e localiza-se numa zona do Parque;
2. Cada atração tem uma numeração única;
3. Um utilizador tem associada uma categoria, a listar: Normal, Júnior, Infantil e Sénior, de acordo com a idade do utilizador (11 aos 64, 5 aos 10, 0 aos 4 e mais de 65 anos respetivamente);
4. A cada categoria, corresponde um preço de bilhete.
5. Cada utilizador, ao entrar e sair do Parque, valida o seu bilhete na entrada principal, sendo registada a hora em que o faz;
6. Cada utilizador possui uma pulseira que tem de passar à entrada da fila de cada atração e à entrada da atração em si, ficando registada a hora em que o faz;
7. Cada utilizador pode visitar qualquer uma das atrações;
8. Cada utilizador, ao comprar o seu bilhete, indica a sua nacionalidade e o seu nome;
9. O Parque possui 4 zonas, a listar: Cabeça, Ombro, Joelho e Pé;

10. Um funcionário trabalha como monitor de apenas uma determinada atração num turno;
11. Um funcionário pode trabalhar mais de um turno por dia;
12. Os funcionários registam a data e hora de entrada e saída de cada turno;
13. Um funcionário tem um salário mensal e um identificador único;
14. Cada funcionário trabalha, em média, 4 turnos de 2 horas por dia, tendo 1 uma de descanso.

2.2.2 Requisitos de exploração

Os requisitos de exploração são os seguintes:

1. Obter uma listagem dos utilizadores que frequentaram uma atração num intervalo de tempo;
2. Obter o tempo médio de espera dos utilizadores de uma atração num intervalo de tempo;
3. Obter o número de utilizadores em fila numa atração num instante de tempo;
4. Obter uma listagem de utilizadores de uma categoria;
5. Obter uma listagem das atrações mais visitadas por utilizadores de uma categoria;
6. Obter a hora de entrada média dos utilizadores de uma categoria;
7. Obter o número total de utilizadores que visitaram o parque num intervalo de tempo em dias (inclusive);
8. Obter o número total de utilizadores que visitaram o parque por categoria num intervalo de tempo em dias (inclusive);
9. Obter o top de utilizadores que mais frequentaram as atrações num dia;
10. Obter as atrações mais visitadas num determinado intervalo de tempo;
11. Obter uma listagem de todos os utilizadores que frequentaram o Parque, por ordem decrescente de tempo permanecido no Parque;
12. Obter a designação da categoria do maior número de visitantes da atração monitorizada por um dado funcionário, num dado turno.

2.2.3 Requisitos de controlo

Os requisitos de controlo são os seguintes:

1. Os funcionários registam novos utilizadores e modificam as suas informações;
2. O departamento de marketing consulta dados estatísticos sobre a frequência das atrações por parte dos utilizadores e os seus tempos de espera;
3. A administração regista e modifica funcionários e atrações;
4. Os sensores, que se encontram no início da fila de espera e na entrada da atração, registam a hora de entrada dos utilizadores na fila e na atração;

2.3. Análise geral dos requisitos

Tendo em conta os requisitos recolhidos acima, é possível agrupá-los de acordo com os atores do sistema:

1. Utilizador

- A cada utilizador está associado o nome, nacionalidade e categoria (sendo a categoria uma característica que define a faixa etária em que o utilizador se insere). De acordo com a altura em que compra o bilhete, é-lhe associado um identificador único. Para além disso, são registados automaticamente os momentos em que o utilizador visita as diversas atrações do parque pelo sistema informático, bem como os momentos em que entra e sai do parque.

2. Categoria

- Uma categoria tem uma designação, uma faixa etária associada e um preço. Um utilizador pertence necessariamente a uma categoria.

3. Atração

- A cada atração está associada uma designação, uma capacidade máxima, uma zona do parque, uma altura mínima e uma duração. Uma atração é monitorizada por um ou mais funcionários a qualquer altura, mas pode não ter utilizadores a visitá-la. São registadas as entradas dos utilizadores na fila para a atração e para a atração em si pelo sistema informático automaticamente.

4. Funcionário

- A cada funcionário está associado o seu nome, o seu salário e o seu identificador único. Trabalha um ou mais turnos por dia, registado a data de início e de fim do seu turno. Pode trabalhar em qualquer uma das atrações.

Para além disso, podemos verificar as principais vistas relativamente à informação que será armazenada no sistema de acordo com os diversos perfis de utilização:

Informação	Acesso	Administração	Marketing	Funcionário	Sensores
Funcionários	Criar/Remover	X			
	Alterar	X			
	Consultar	X			
Utilizadores	Criar/Remover	X		X	
	Alterar	X			X
	Consultar	X	X		
Atrações	Criar/Remover	X			
	Alterar	X			
	Consultar	X	X		
Turno	Criar/Remover	X			X
	Consultar	X			
Visita a atração	Criar/Remover	X			X
	Alterar	X			X

	Consultar	X	X		
--	-----------	---	---	--	--

Tabela 1 – Principais vistas de utilização do sistema

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Para modelarmos conceptualmente o Caso de Estudo em causa, produzimos uma análise geral dos requisitos (que se encontra em 2.3.) que permitirá a identificação das principais entidades envolvidas no sistema e respetivos relacionamentos.

De seguida, através dos requisitos de descrição, poderemos reconhecer os atributos associados a cada entidade e relacionamento e, posteriormente, desenvolver o Diagrama ER.

3.2. Identificação e caracterização das entidades

A partir da análise de requisitos foram identificadas 4 entidades, que se caracterizam na Tabela 2 - Caracterização das entidades.

Nome da Entidade	Descrição	Apelidos	Ocorrência
Funcionário	Termo geral para descrever todos os empregados do Parque Aquático	Empregados, Staff	Cada Funcionário trabalha numa determinada Atração
Atração	Termo geral para descrever todas as atividades possíveis do Parque Aquático	Diversões	Cada Atração é controlada por um Funcionário e pode ser usada por um utilizador
Utilizador	Termo geral para descrever todas as pessoas que frequentam o Parque Aquático	Cliente	Cada Utilizador tem associada a si uma Categoria e pode usufruir de uma ou mais Atrações
Categoria	Termo geral para descrever a faixa etária dos Utilizadores	Faixa etária	Cada Categoria representa um tipo de Utilizador do Parque, de acordo com a Administração

3.3. Identificação e caracterização dos relacionamentos

De acordo com a análise de requisitos, foram identificados 3 relacionamentos que se encontram na Tabela 3 – Caracterização dos relacionamentos. Os requisitos que deram origem a estes relacionamentos foram o 3, o 6, o 7, o 10 e o 12, todos eles de descrição.

Quanto às multiplicidades, tiveram-se em consideração os seguintes factos:

- Cada atração terá de ter pelo menos um funcionário a trabalhar nela (requisito de descrição 1), e cada funcionário poderá trabalhar ao longo dos turnos em atrações diferentes (requisito de descrição 11);
- Um utilizador poderá visitar o Parque mas não visitar nenhuma atração, tal como uma atração poderá não ser visitada por qualquer utilizador;
- Cada utilizador terá de pertencer necessariamente a uma categoria (requisito de descrição 3), mas poderão não haver utilizadores de uma dada categoria.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Funcionário	1.. n	Trabalha em	1.. n	Atração
Atração	0.. n	É visitado por	0.. n	Utilizador
Utilizador	0.. n	Pertence a	1.. 1	Categoria

Tabela 3 – Caracterização dos relacionamentos

3.4. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

Na tabela seguinte, estão devidamente identificados e caracterizados todos os atributos relativos às entidades identificadas no sistema projetado. É feita referência ao nome da entidade, e para cada atributo a sua descrição, tipo, tamanho, se é chave primária, se é um atributo composto, multivalorado, possivelmente nulo ou derivado. Inclui-se também um valor *default*, se aplicável.

Entidade	Atributos	Descrição	Tipo e tamanho	Chave primária	Atributo Composto	Atributo Nulo	Atributo Multivalorado	Atributo derivado	Valor default
Funcionário	Id	Identificador único do funcionário	Inteiro	Sim	Não	Não	Não	Não	0
	Nome	Nome completo do funcionário	64 caracteres variáveis	Não	Não	Não	Não	Não	FSN (funcionário sem nome)
	Salário	Salário mensal do funcionário, em euros (€)	Decimal (6,2)	Não	Não	Não	Não	Não	0
Atração	Id	Identificador único da atração	Inteiro	Sim	Não	Não	Não	Não	0
	Designação	Nome da atração	32 caracteres variáveis	Não	Não	Não	Não	Não	ASN (atração sem nome)
	Capacidade	Número máximo de utilizadores por viagem na atração	Inteiro	Não	Não	Não	Não	Não	0
	Zona	Zona do parque onde a atração se localiza	64 caracteres variáveis	Não	Não	Não	Não	Não	ZSN (zona sem nome)
	Duração	Duração, em minutos, de uma viagem na atração	Time	Não	Não	Não	Não	Não	0
	Altura mínima	Altura mínima que um utilizador necessita de ter para frequentar a atração	Inteiro	Não	Não	Não	Não	Não	0
Utilizador	Id	Identificador único do utilizador	Inteiro	Sim	Não	Não	Não	Não	0
	Nome	Nome completo do utilizador	64 caracteres variáveis	Não	Não	Não	Não	Não	USN (utilizador sem nome)
	Nacionalidade	País de origem do utilizador	32 caracteres variáveis	Não	Não	Não	Não	Não	USNA (sem nacionalidade)

	Hora de entrada no parque	Data e hora que o utilizador entrou no parque	DateTime	Não	Não	Não	Não	Não	Data atual
	Hora de saída do Parque	Data e hora que o utilizador saiu no parque	DateTime	Não	Não	Sim	Não	Não	NULL
	Nº de Atrações Visitadas	Número de atrações que o utilizador visitou	Inteiro	Não	Não	Não	Não	Sim	0
Categoria	Id	Identificador único da categoria	Inteiro	Sim	Não	Não	Não	Não	0
	Designação	Nome da categoria	32 caracteres variáveis	Não	Não	Não	Não	Não	CSN (categoria sem nome)
	Preço	Preço, em euros, que os utilizadores pertencentes a esta categoria pagam	Decimal (5,2)	Não	Não	Não	Não	Não	0
	Idade inferior	Idade mínima para pertencer a esta categoria	Inteiro	Não	Não	Não	Não	Não	0
	Idade Superior	Idade máxima para pertencer a esta categoria	Inteiro	Não	Não	Não	Não	Não	0

Tabela 4 – Caracterização dos atributos das entidades

Quanto aos relacionamentos identificados na secção anterior, os seus atributos encontram-se devidamente reconhecidos e caracterizados na tabela que se segue, que apresenta a descrição, tipo e tamanho de cada um, bem como se é uma chave primária, um atributo composto, nulo, multivalorado ou derivado. Para além disso, inclui também um valor *default*, para os casos em que se aplica.

Relacionamento	Atributos	Descrição	Tipo e tamanho	Chave primária	Atributo Composto	Atributo Nulo	Atributo Multivalorado	Atributo derivado	Valor default
----------------	-----------	-----------	----------------	----------------	-------------------	---------------	------------------------	-------------------	---------------

Funcionário- Atração	Data de início	Data e hora do início do turno de um funcionário numa determinada atração	DateTime	Sim	Não	Não	Não	Não	Data Atual
	Data de fim	Data e hora do fim do turno de um funcionário numa determinada atração	DateTime	Não	Não	Sim	Não	Não	NULL
Atração- Utilizador	Data de entrada na fila	Data e hora da entrada de um utilizador na fila de espera numa determinada atração	DateTime	Sim	Não	Não	Não	Não	Data Atual
	Data de entrada na atração	Data e hora da entrada de um utilizador numa determinada atração	DateTime	Não	Não	Sim	Não	Não	NULL

Tabela 5 – Caracterização dos atributos dos relacionamentos

3.5. Detalhe ou generalização de entidades

Tendo em conta que o nosso diagrama possui 4 entidades, o número de superclasses e subclasses de entidades nunca poderia ser elevado. No entanto, podemos generalizar as entidades Funcionário e Utilizador numa superclasse Pessoa.

Essa superclasse requer que todos os seus membros sejam membros de uma das subclasses (restrição de participação), e que as suas subclasses sejam disjuntas, já que caso os funcionários utilizem o Parque como utilizadores, tal utilização não deverá ser contabilizada para efeitos estatísticos por ser diminuta e não representar o público-alvo do Parque.

Pessoa		Funcionário	Utilizador				
ID	Nome	Salário	Nacionalidade	Hora de saída do Parque	Hora de entrada no Parque	Categoria	Nº de atrações visitadas

Tabela 6 – Generalização das entidades Funcionário e Utilizador

Apesar de se poder optar por esta generalização, o diagrama ER produzido tem apenas 4 entidades e 3 relacionamentos que traduzem fielmente o Parque e o seu funcionamento segundo os requisitos recolhidos, e, como a leitura do diagrama é bastante clara sem a introdução da superclasse acima mencionada, optamos por não considerar o uso da generalização no diagrama final.

3.6. Apresentação e explicação do diagrama ER

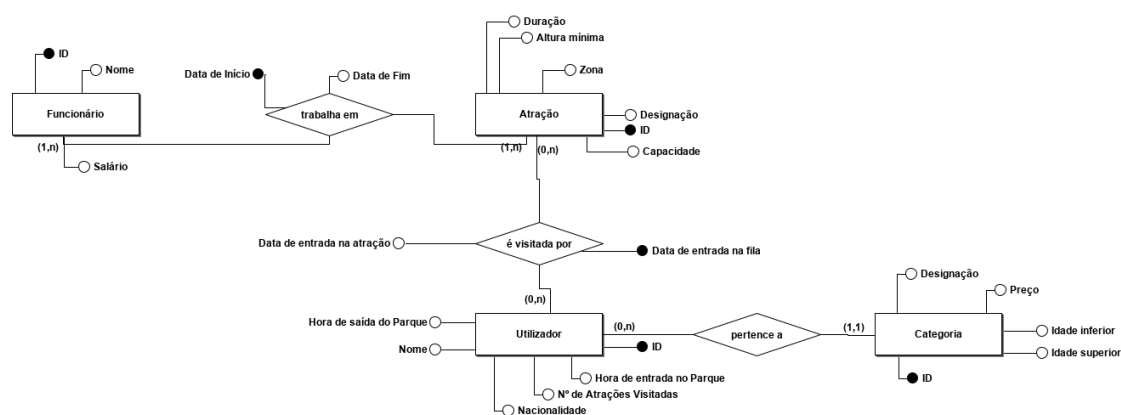


Figura 1 – Diagrama ER

Tendo em conta tudo aquilo que foi apresentado até este ponto no relatório, concebemos o diagrama ER que se encontra acima. Neste, estão representadas as 4 entidades com os respetivos atributos e as 3 relações que as unem e representam uma parte importante do sistema.

3.7. Validação do modelo de dados com o utilizador

Após nova reunião com o Conselho de Administração do Parque, foi-nos indicado que o modelo se adequava às necessidades do sistema a ser implementado.

No entanto, uma reunião intermédia permitiu que fossem corrigidas algumas limitações resultantes da omissão de alguns requisitos quanto ao funcionamento das categorias de utilizadores. Inicialmente a Categoria era apenas um atributo do Utilizador, mas fomos posteriormente informados que o número e nome de cada Categoria não era estático e poderia vir a ser alterado no futuro, bem como o respetivo preço e intervalo de idades. Por isso, revimos o nosso modelo e optamos por colocar uma nova entidade (Categoria), tal como está representado na Figura 1 – Diagrama ER, com os atributos que lhe estão associados.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Para traduzir o modelo conceptual para o modelo lógico, é necessário que o último seja validado para garantir a sua correção estrutural e capacidade de suportar as transações e interrogações requeridas pelo sistema.

4.1.1 Entidades fortes e fracas

Todas as entidades do modelo são entidades fortes, pois a sua existência não depende da existência de outras entidades. Enumeram-se de seguida:

Funcionário (Id, Nome, Salario)

Chave Primária Id

Atração (Id, Designacao, Zona, Capacidade, Duracao, Altura_Minima)

Chave Primária Id

Utilizador (Id, Nome, Nacionalidade, Hora_entrada_parque, Hora_saida_parque, Categoria_Id, N_Atracoes_Visitadas)

Chave Primária Id

Chave Estrangeira Categoria_Id **referente** Categoria (Id)

Categoria (Id, Designacao, Preco, Idade_inferior, Idade_superior)

Chave Primária Id

4.1.2 Relações binárias

Nas 3 relações existentes entre as entidades, apenas verificamos 2 tipos de relações: duas de N:M e uma de 1:N, que se listam de seguida:

1. 1:N (1 para Muitos)

Identificamos esta relação entre a entidade Categoria (1) e a entidade Utilizador (N). A relação entre estas entidades é estabelecida através da inserção de uma cópia da chave primária (Id) da entidade Categoria (pai), na entidade Utilizador (filho).

(Id=Categoria_Id)	
Categoria (Id, Designacao, Preco, Idade_inferior, Idade_superior) Chave Primária Id	Utilizador (Id, Nome, Nacionalidade, Hora_entrada_parque, Hora_saida_parque, Categoria_Id, N_Atracoes_Visitadas) Chave Primária Id Chave Estrangeira Categoria_Id referente Categoria(Id)

Tabela 7 - Relacionamento entre Categoria e Utilizador

2. N:M (Muitos para muitos)

Encontramos 2 relações deste tipo no nosso modelo conceptual: entre as entidades Funcionário e Atração, e entre as entidades Atração e Utilizador.

No primeiro caso (Funcionário-Atração), é criada uma nova tabela, onde as chaves primárias das entidades Funcionário e Atração serão inseridas como chave estrangeira. Neste caso, para além de serem chaves estrangeiras, também serão parte da chave primária, uma vez que esta será composta. Para além disso, esta relação possui dois atributos: Data de início e Data de fim, sendo que a primeira será também uma parte da chave primária composta.

A chave primária desta relação será então composta pelo Id do Funcionário, pelo Id da Atração e pela Data de Início o turno, uma vez que um turno é definido univocamente pelo Funcionário que trabalha nele, pela Atração em que trabalha e pela hora em que inicia o turno.

(Id = Funcionario_ID)	Funcionário (Id, Nome, Salario) Chave Primária Id	Atração (Id, Designacao, Zona, Capacidade, Duracao, Altura_Minima) Chave Primária Id	(Id = Atracao_ID)
	trabalha_em (Funcionario_ID, AtracaoID, Data_de_Inicio, Data_de_fim) Chave Primária Funcionario_ID, AtracaoID, Data_de_Inicio Chave Estrangeira Funcionario_ID referente Funcionário (Id) Chave Estrangeira AtracaoID referente Atração (Id)		

Tabela 8 - Relacionamento trabalha_em

No segundo caso (Atração - Utilizador), é criada uma nova tabela onde as chaves primárias de cada entidade envolvida serão inseridas como chave estrangeira. Tal como no caso anterior, para além de serem chaves estrangeiras, também serão parte da chave primária composta. Para além disso, esta relação possui dois atributos: Data de entrada na fila e Data de entrada na atração, sendo que a primeira também será parte da chave primária da relação.

A chave primária desta relação será então composta pelo Id da Atração, pelo Id do Utilizador, e pela Data de Entrada na fila, uma vez que uma visita a uma atração apenas é definida univocamente pelo Utilizador que a visita, pela Atração visitada e pela Data em que entra na fila.

(Id = Atracao_ID)	Atração (Id, Designacao, Zona, Capacidade, Duracao, Altura_Minima) Chave Primária Id	Utilizador (Id, Nome, Nacionalidade, Hora_entrada_parque, Hora_saida_parque, Categoria_Id, N_Atracoes_Visitadas) Chave Primária Id Chave Estrangeira Categoria_Id	(Id = Utilizador_ID)
	e_visitado por (Atracao_Id, Utilizador_Id, Data_de_entrada_na_fila, Data_de_entrada_na_Atracao) Chave Primária Atracao_Id, Utilizador_Id, Data_de_entrada_na_fila Chave Estrangeira Atracao_Id referente Atração (Id) Chave Estrangeira Utilizador_Id referente Utilizador (Id)		

Tabela 9 - Relacionamento e_visitada_por

4.2. Desenho do modelo lógico

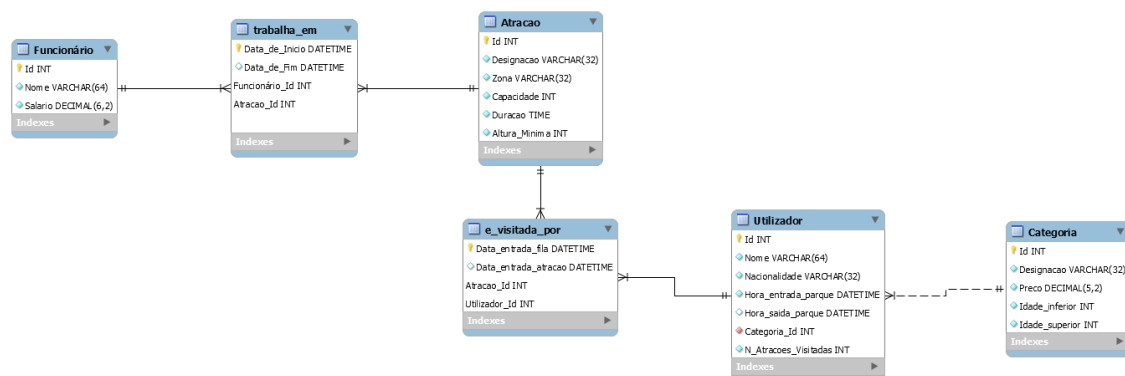


Figura 2 – Diagrama do Modelo Lógico

4.3. Validação do modelo através da normalização

Para validar o modelo através da normalização é necessário converter o modelo lógico em tabelas relacionais normalizadas:

- **Funcionário:**
id_funcionario -> salario, nome.
- **Atracao:**
id_atracao -> designacao, zona, capacidade, duracao, altura_minima.
- **Utilizador:**
id_utilizador -> nome, nacionalidade, hora_entrada_parque, hora_saida_parque, categoria, n_atracoes_visitadas.
- **Categoria:**
id_categoria -> designacao, preco, Idade_inferior, Idade_superior.
- **Trabalha_em:**
atracao_id, funcionario_id data_de_inicio, -> data_de_fim.
- **E_visitada_por:**
data_entrada_fila, atracao_id, utilizador_id -> data_entrada_atracao.

As tabelas relacionais normalizadas respeitam a Primeira Forma Normal porque todos os atributos são atômicos, ou seja, não é possível decompô-los e não existe nenhum conjunto de atributos repetidos que descrevem a mesma característica.

Para seguir a Segunda Forma Normal o modelo tem de respeitar a Primeira Forma Normal e todos os atributos não-chave não têm uma dependência funcional com os atributos chave, o que se verifica.

Podemos concluir que este modelo também respeita a Terceira Forma Normal porque não existe qualquer dependência funcional entre os atributos não chave e está na Segunda Forma Normal.

4.4. Validação do modelo com interrogações do utilizador

O nosso modelo é válido porque da forma como está estruturado consegue responder às interrogações do levantadas em 2.2.

1. **Obter uma listagem dos utilizadores que frequentaram uma atração num intervalo de tempo**
Para responder a esta interrogação, consultamos a tabela Utilizador e juntamos a tabela e_visitada_por, filtrada pelo intervalo de tempo dado.
2. **Obter o tempo médio de espera dos utilizadores de uma atração num intervalo de tempo**

Para responder a esta interrogação, consultamos a tabela e_visitada_por filtrada pelo intervalo de tempo dado e calculamos o tempo médio de espera.

3. Obter o número de utilizadores em fila numa atracção num intervalo de tempo

Para responder a esta interrogação, consultamos a tabela e_visitada_por filtrada pelo intervalo de tempo dado e contamos o número de entradas da tabela.

4. Obter uma listagem de utilizadores de uma categoria

Para responder a esta interrogação, consultamos a tabela Utilizador filtrada pela categoria dada.

5. Obter uma listagem das atracções mais visitadas por utilizadores de uma categoria

Para responder a esta interrogação, consultamos a tabela Utilizador, juntamos a tabela e_visitada_por filtrada pela categoria e por fim juntamos também a tabela Atracao, agrupando-a pelo Id da atracção e ordenando o resultado por ordem decrescente.

6. Obter a hora de entrada média dos utilizadores de uma categoria

Para responder a esta interrogação, consultamos a tabela Utilizador filtrada pela categoria dada e calculamos a hora de entrada média.

7. Obter o número de utilizadores do parque num intervalo de tempo

Para responder a esta interrogação, consultamos a tabela Utilizador filtrada pelo intervalo de tempo dado e contamos o número de entradas da tabela filtrada.

8. Obter o número de utilizadores do parque por categoria num intervalo de tempo

Para responder a esta interrogação, consultamos a tabela Utilizador filtrada pela categoria e intervalo de tempo dados.

9. Obter os utilizadores que mais frequentaram as atracções num dia

Para responder a esta interrogação, consultamos a tabela Utilizador filtrada pelo intervalo de tempo dado e ordenamos pelo atributo N_Atracoes_Visitadas por ordem decrescente.

10. Obter as atracções mais visitadas num determinado intervalo de tempo

Para responder a esta interrogação, consultamos a tabela Atracao e juntamos a tabela e_visitada_por filtrada pelo intervalo de tempo dado.

11. Obter uma listagem de todos os utilizadores que frequentaram o Parque, por ordem decrescente de tempo permanecido no Parque

Para responder a esta interrogação, consultamos a tabela Utilizador ordenada pela diferença da hora de entrada e saída do Parque, dada pelos atributos (Hora_entrada_parque e Hora_saida_parque).

12. Obter a designação da categoria do maior número de visitantes da atracção monitorizada por um dado funcionário, num dado turno

Para responder a esta interrogação, consultamos a tabela Atracao, juntamos as tabelas e_visitada_por, Utilizador e Categoria filtrando-as pelo turno dado. De seguida agrupamos o resultado por Categoria e ordenamos por ordem decrescente.

4.5. Validação do modelo com as transações estabelecidas

Tendo em conta os requisitos recolhidos no capítulo 2. , teremos de validar o nosso modelo lógico de acordo com as seguintes transações principais:

- **Visitar uma atração**

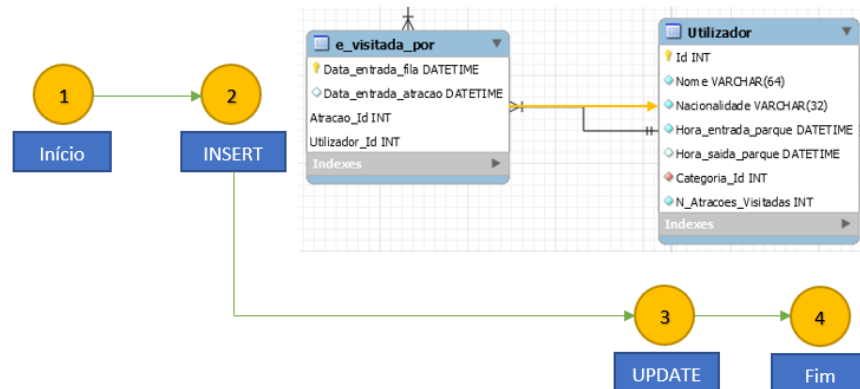


Figura 3 - Transação Visitar uma atração

Quando um Utilizador visita uma atração, criará uma nova entrada na tabela e_visitada_por, marcando a hora em que entra na fila. Para além disso, também deverá incrementar o atributo N_Atracoes_Visitas numa unidade.

- **Registar um Utilizador**

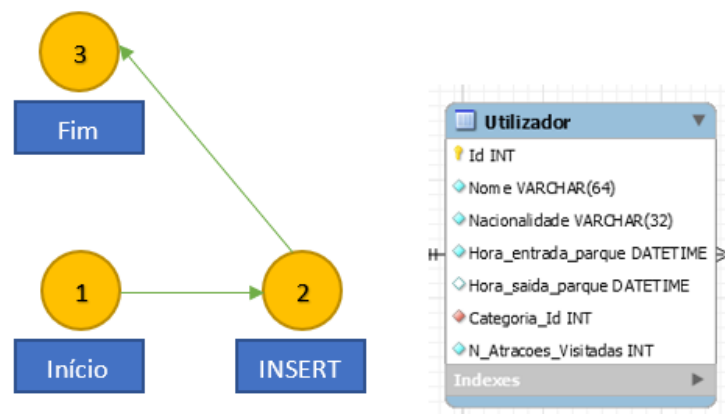


Figura 4 - Transação Registar um Utilizador

Para registar um Utilizador, o Funcionário precisa de inserir o seu Nome, Nacionalidade e Categoria e Hora de Entrada no Parque, sendo uma inserção de uma nova entrada na tabela Utilizador.

- **Registar um Funcionário**

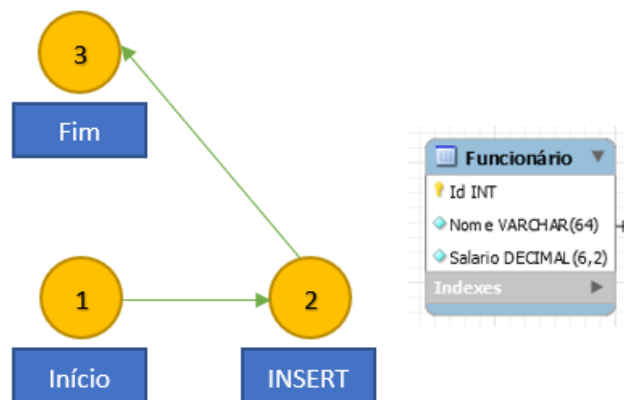


Figura 5 - Transação Registrar um Funcionário

Para registrar um Funcionário, a Administração precisa de inserir o seu Nome e Salário, sendo uma inserção de uma nova entrada na tabela Funcionário.

- **Registrar uma Atração**

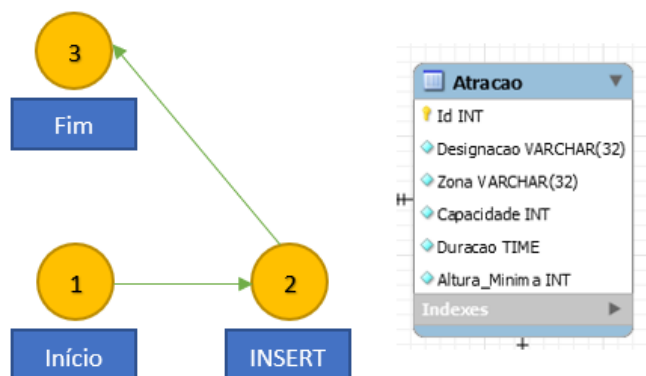


Figura 6 - Transação Registrar uma Atração

Para registrar uma nova Atração, a Administração precisa de inserir a sua Designação, Id, Zona, Capacidade, Duração e Altura Mínima, sendo uma inserção de uma nova entrada na tabela Atracao.

- **Registrar um turno**

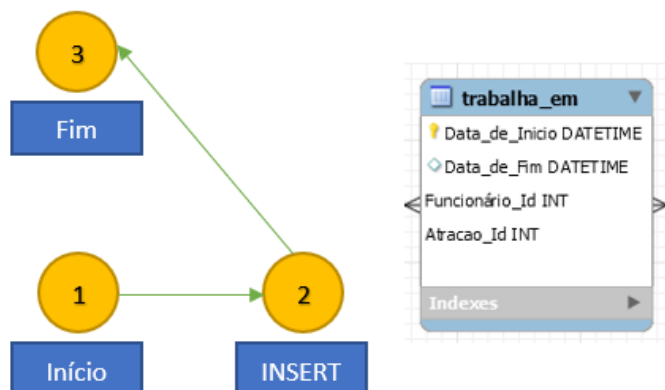


Figura 7 - Transação Registrar um turno

Para se registrar um turno, o Sensor insere uma Data de Início do turno e uma Data de Fim, que serão necessariamente consecutivas, para além de incluir o Id do Funcionário que realizou o turno, e o Id da Atração na qual trabalhou o funcionário. Trata-se também de uma inserção de uma nova entrada na tabela trabalha_em.

- **Registrar saída do Parque**

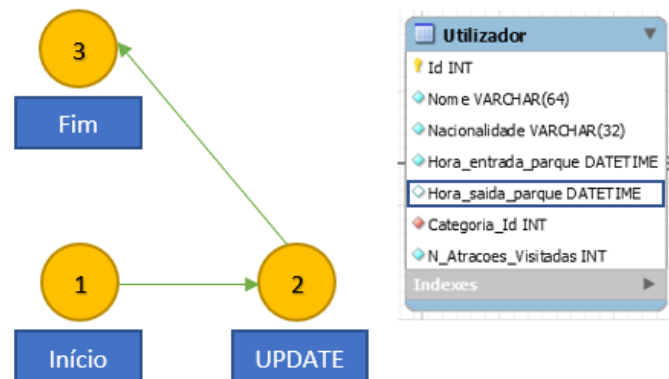


Figura 8 - Transação Registrar saída do Parque

Para o Sensor registrar a saída do Parque de um Utilizador, terá de alterar a tabela Utilizador e colocar a Hora de Saída do Parque de acordo com o momento em que recebeu essa informação, sendo necessariamente posterior à Hora de Entrada no Parque. Trata-se de uma alteração à entrada do Utilizador que saiu do parque, no campo Hora_saida_parque.

4.6. Revisão do modelo lógico com o utilizador

Após a definição do modelo lógico, reunimos novamente com o Conselho de Administração de forma a validar o modelo produzido. Como tinha sido validado anteriormente o modelo conceptual, também este foi validado, dado que foi obtido seguindo as regras de transição do modelo conceptual para o modelo lógico.

5. Implementação Física

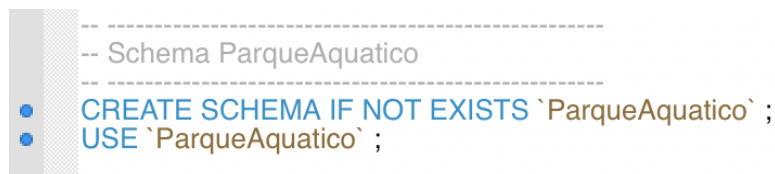
5.1. Seleção do sistema de gestão de bases de dados

Para implementarmos fisicamente o sistema modelado logicamente, optamos por usar o MySQL da Oracle, uma vez que é o sistema com o qual os elementos que desenvolveram este sistema estão mais experienciados e foi aquele usado ao longo da UC de Base de Dados relativamente aos sistemas de gestão de bases de dados relacionais.

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

Usando o MySQL como sistema de gestão, começamos por criar um script que irá ser a base do nosso sistema.

Após criado o esquema na base de dados, através dos comandos da Figura 9 – Criação do esquema ParqueAquatico, que será referenciado como ParqueAquatico, precisamos de criar e definir cada entidade e relacionamento definidos no passo anterior de modelação.



```
-- -----  
-- Schema ParqueAquatico  
-- -----  
• CREATE SCHEMA IF NOT EXISTS `ParqueAquatico` ;  
• USE `ParqueAquatico` ;
```

Figura 9 – Criação do esquema ParqueAquatico

Para as entidades Categoria, Atracao e Funcionario, inserimos todos os seus atributos com os seus domínios, não podendo nenhum deles tomar o valor nulo (possui sempre um valor *default*). Como também referido anteriormente, para todas as entidades acima a sua chave primária é o atributo “Id”, sendo que para o caso do Funcionário, este atributo é automaticamente incrementado a cada inserção na tabela.

```

-----
-- Table `ParqueAquatico`.`Categoria`
-----
• DROP TABLE IF EXISTS `ParqueAquatico`.`Categoria` ;
• CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Categoria` (
  `Id` INT NOT NULL DEFAULT 0,
  `Designacao` VARCHAR(32) NOT NULL DEFAULT 'CSN (categoria sem nome)',
  `Preco` DECIMAL(5,2) NOT NULL DEFAULT 0,
  `Idade_inferior` INT NOT NULL DEFAULT 0,
  `Idade_superior` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

```

Figura 10 – Criação da tabela Categoria

```

-----
-- Table `ParqueAquatico`.`Atracao`
-----
• DROP TABLE IF EXISTS `ParqueAquatico`.`Atracao` ;
• CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Atracao` (
  `Id` INT NOT NULL DEFAULT 0,
  `Designacao` VARCHAR(32) NOT NULL DEFAULT 'ASN (atração sem nome)',
  `Zona` VARCHAR(32) NOT NULL DEFAULT 'ZSN (zona sem nome)',
  `Capacidade` INT NOT NULL DEFAULT 0,
  `Duracao` TIME NOT NULL DEFAULT 0,
  `Altura_Minima` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

```

Figura 11 – Criação da tabela Atração

```

-----
-- Table `ParqueAquatico`.`Funcionario`
-----
• DROP TABLE IF EXISTS `ParqueAquatico`.`Funcionario` ;
• CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Funcionario` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(64) NOT NULL DEFAULT 'FSN (funcionário sem nome)',
  `Salario` DECIMAL(6,2) NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

```

Figura 12 – Criação da tabela Funcionário

Relativamente à tabela Utilizador, são adicionados todos os seus atributos, que não podem tomar o valor nulo à exceção do atributo “Hora_saida_parque”. Este atributo pode ter um valor nulo uma vez que o sistema poderá ser consultado num momento em que o Utilizador poderá ainda não ter saído do recinto. A sua chave primária será o atributo “Id” que, tal como o Funcionário, será incrementado automaticamente aquando de cada nova inserção.

Para além disso, esta entidade possui um atributo derivado: “N_Atracoes_Visitadas”. Foi criado um *trigger* para que cada vez que o utilizador entra numa atração, o valor do atributo seja incrementado numa unidade.

Também possui uma chave estrangeira “Utilizador_Categoria”, que referencia o “Id” da entidade Categoria. Caso a chave original seja alterada, essa alteração é refletida também nos utilizadores, uma vez que devem ter sempre associados valores válidos de uma Categoria. No entanto, caso seja pretendido apagar um registo da tabela Categoria, a ação não será permitida caso existam registos de utilizadores com essa Categoria, já que pode remover informação relevante para as consultas realizadas na Base de Dados.

```
-- Table `ParqueAquatico`.`Utilizador`
--
DROP TABLE IF EXISTS `ParqueAquatico`.`Utilizador` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Utilizador` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(64) NOT NULL DEFAULT 'USN (utilizador sem nome)',
  `Nacionalidade` VARCHAR(32) NOT NULL DEFAULT 'USNA (sem nacionalidade)',
  `Hora_entrada_parque` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Hora_saida_parque` DATETIME NULL DEFAULT NULL,
  `Categoria_Id` INT NOT NULL,
  `N_Atracoes_Visitadas` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`),
  INDEX `fk_Utilizador_Categoria1_idx` (`Categoria_Id` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_Categoria1`
    FOREIGN KEY (`Categoria_Id`)
      REFERENCES `ParqueAquatico`.`Categoria` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Figura 13 - Criação da tabela Utilizador

Para a criação da tabela e_visitada_por, adicionam-se os seus atributos, sendo que apenas o atributo “Hora_entrada_atracao” pode tomar o valor nulo, uma vez que um utilizador pode entrar na fila da atração e serem realizadas consultas à base de dados enquanto este ainda não entrou na atração em si.

Relativamente às chaves, a sua chave primária é composta pelos atributos “Data_entrada_fila”, “Atracao_Id” e “Utilizador_Id”, sendo que os dois últimos atributos são também chaves estrangeiras. Como na tabela anterior, caso a chave primária da tabela que origina esta chave estrangeira seja alterada, essa alteração reflete-se nesta tabela. Caso alguma das chaves estrangeiras, por algum motivo, queira ser removida, essa ação estará limitada pela inexistência de registos na tabela “e_visitada_por” com referência às chaves a eliminar.


```

-----
-- Table `ParqueAquatico`.`e_visitada_por`
-----
• DROP TABLE IF EXISTS `ParqueAquatico`.`e_visitada_por` ;
• CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`e_visitada_por` (
  `Data_entrada_fila` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Data_entrada_atracao` DATETIME NULL DEFAULT NULL,
  `Atracao_Id` INT NOT NULL,
  `Utilizador_Id` INT NOT NULL,
  PRIMARY KEY (`Data_entrada_fila`, `Atracao_Id`, `Utilizador_Id`),
  INDEX `fk_e_visitada_por_Atracao1_idx` (`Atracao_Id` ASC) VISIBLE,
  INDEX `fk_e_visitada_por_Utilizador1_idx` (`Utilizador_Id` ASC) VISIBLE,
  CONSTRAINT `fk_e_visitada_por_Atracao1`
    FOREIGN KEY (`Atracao_Id`)
      REFERENCES `ParqueAquatico`.`Atracao` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_e_visitada_por_Utilizador1`
    FOREIGN KEY (`Utilizador_Id`)
      REFERENCES `ParqueAquatico`.`Utilizador` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Figura 14 – Criação da tabela e_visitada_por

Por fim, é criada a tabela da relação trabalha_em com os seus respetivos atributos, em que apenas o atributo “Data_de_Fim” pode ser nulo pois no momento em que se consulta a nossa base de dados, o funcionário pode não ter terminado o seu turno.

Possui uma chave primária composta pelos atributos “Data_de_inicio”, “Funcionario_Id” e “Atracao_Id”. Os últimos dois são simultaneamente chave primária e chave estrangeira. Aplicam-se os mesmos princípios de atualização/remoção de chaves estrangeiras que na relação anterior.

```

-----
-- Table `ParqueAquatico`.`trabalha_em`
-----
• DROP TABLE IF EXISTS `ParqueAquatico`.`trabalha_em` ;
• CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`trabalha_em` (
  `Data_de_Inicio` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Data_de_Fim` DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  `Funcionario_Id` INT NOT NULL,
  `Atracao_Id` INT NOT NULL,
  PRIMARY KEY (`Data_de_Inicio`, `Funcionario_Id`, `Atracao_Id`),
  INDEX `fk_trabalha_em_Funcionario_idx` (`Funcionario_Id` ASC) VISIBLE,
  INDEX `fk_trabalha_em_Atracao1_idx` (`Atracao_Id` ASC) VISIBLE,
  CONSTRAINT `fk_trabalha_em_Funcionario`
    FOREIGN KEY (`Funcionario_Id`)
      REFERENCES `ParqueAquatico`.`Funcionario` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_trabalha_em_Atracao1`
    FOREIGN KEY (`Atracao_Id`)
      REFERENCES `ParqueAquatico`.`Atracao` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Figura 15 – Criação da tabela trabalha_em

5.3. Tradução das interrogações do utilizador para SQL

Todas as interrogações referidas em 4.4. foram traduzidas para SQL. Dada a extensão da listagem de todas elas, optou-se por colocá-las na íntegra na secção dos Anexos (ver Anexo 2 – Interrogações em SQL).

5.4. Tradução das transações estabelecidas para SQL

As transações estabelecidas em 4.5. foram traduzidas para SQL.

- **Visitar uma atração**

```
CREATE PROCEDURE `VisitaAtracao` (in UtilizadorID INT, AtracaoID INT )
begin
  insert into e_visitada_por (Data_entrada_fila, Data_entrada_atracao, Atracao_Id, Utilizador_Id)
  VALUES
  (NOW(), NULL ,AtracaoID,UtilizadorID);
end
```

Figura 16 - Transação Visitar uma atração em SQL

Quando um Utilizador visita uma atração, criará uma nova entrada na tabela e_visitada_por, marcando a hora em que entra na fila, o seu ID e o ID da atração que está a visitar. Para além disso, é usado um *trigger* para incrementar em 1 valor o número de atrações visitadas pelo Utilizador.

```
create trigger AtualizaAtracoesVisitadas after insert on e_visitada_por
for each row
begin
  update Utilizador
  set Utilizador.N_Atracoes_Visitadas = Utilizador.N_Atracoes_Visitadas + 1
  where Utilizador.Id = NEW.Utilizador_Id;
end
```

Figura 17 – Trigger para incrementar o número de Atrações Visitadas

- **Registar um Utilizador**

```
CREATE PROCEDURE `InsertUser` (in UserNome VARCHAR(64), UserNacionalidade VARCHAR(32), UserCategoria_id INT)
begin
  INSERT INTO Utilizador (Nome,Nacionalidade,Hora_entrada_parque,Hora_saida_parque,Categoria_Id,N_Atracoes_Visitadas)
  VALUES
  (UserNome,UserNacionalidade,NOW(), NULL,UserCategoria_id,0);
end
```

Figura 18 - Transação Registar um Utilizador em SQL

Para registar um Utilizador novo, é necessário inserir o Nome, Nacionalidade e Categoria e será criada uma nova entrada na tabela Utilizador.

- **Registar um Funcionário**

```
CREATE PROCEDURE `InsertFuncionario` (in FunNome VARCHAR(64),FunSalario DECIMAL(5,2))
begin
insert into Funcionario (Nome, Salario)
VALUES
(FunNome, FunSalario);
end
```

Figura 19 - Transação Registrar um Funcionário em SQL

Para registrar um Funcionário, é necessário inserir o seu Nome e Salário, sendo a inserção uma nova entrada na tabela Funcionario.

- **Registrar uma Atração**

```
CREATE PROCEDURE `InsertAtracao` (in id INT, ATDesignacao VARCHAR(32),ATZona VARCHAR(32), ATCapacidade INT , ATDuracao TIME, ATAltura INT)
begin
insert into Atracao (Id, Designacao, Zona, Capacidade, Duracao, Altura_Minima)
VALUES
(id, ATDesignacao,ATZona,ATCapacidade,ATDuracao,ATAltura);
end
```

Figura 20 - Transação Registrar uma Atração em SQL

Para registrar uma nova Atração, é necessário inserir a sua Designação, Id, Zona, Capacidade, Duração e Altura Mínima, sendo a inserção uma nova entrada na tabela Atracao.

- **Registrar um turno**

```
CREATE PROCEDURE `InsertTurno` (in DataBegin DATETIME, DataEnd DATETIME, FunID INT, AtracaoID INT )
begin
insert into trabalha_em (Data_de_Inicio, Data_de_Fim, Funcionario_Id, Atracao_Id)
VALUES
(DataBegin, DataEnd, FunID, AtracaoID);
end
```

Figura 21 - Transação Registrar um turno em SQL

Para registrar um novo Turno, é necessário inserir a sua Data de Inicio e de Fim do turno, o ID da Atração e o ID do funcionário, sendo a inserção uma nova entrada na tabela trabalha_em.

- **Registrar saída do Parque**

```
CREATE PROCEDURE `SaidaUser` (in UID INT)
begin
UPDATE `Utilizador`
SET Hora_saida_parque = NOW()
WHERE ID = UID;
end
```

Figura 22 - Transação Registrar saída do Parque SQL

Para registrar a saída do parque de um Utilizador, é necessário inserir o ID do utilizador. O parâmetro Hora_saida_parque referente a esse Utilizador, é então alterado para a hora atual.

5.5. Escolha, definição e caracterização de índices em SQL

Dado que a definição de índices extra aumenta o tempo despendido pelo sistema a realizar inserções, optamos por não colocar quaisquer índices extra em nenhuma das relações. Isto porque a maior parte das operações realizadas no sistema são de inserção de dados por parte dos Sensores, e apenas são consultados os valores (ação que beneficiaria da adição de possíveis índices) esporadicamente, quer pelo Departamento de Marketing, quer pela Administração.

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

Com o desenvolvimento de uma Base de Dados é necessário ter em conta o espaço em disco que será necessário para armazenar todos os dados produzidos. Cada registo de informação de uma determinada tabela ocupa espaço físico na memória, dependendo do tipo de dados a que este está associado, tal como apresentado nas seguintes tabelas.

Atributo	Tamanho máximo (Bytes)
Id	4
Designação	32
Preço	5
Idade_inferior	4
Idade_superior	4

Tabela 10 - Tamanho máximo por entrada da tabela Categoria

Atributo	Tamanho máximo (Bytes)
Id	4
Nome	64
Nacionalidade	32
Hora_entrada_parque	8
Hora_saida_parque	8
Categoria	4
N_atracoes_visitadas	4

Tabela 11 - Tamanho máximo por entrada da tabela Utilizador

Atributo	Tamanho máximo (Bytes)
Data_entrada_fila	8
Data_entrada_atracao	8
Atracao_id	4

Utilizador_id	4
---------------	---

Tabela 12 - Tamanho máximo por entrada da tabela e_visitada_por

Atributo	Tamanho máximo (Bytes)
Id	4
Designação	32
Zona	32
Capacidade	4
Duracao	3
Altura_Minima	4

Tabela 13 - Tamanho máximo por entrada da tabela Atracao

Atributo	Tamanho máximo (Bytes)
Data_de_Inicio	8
Data_de_Fim	8
Funcionario_Id	4
Atracao_id	4

Tabela 14 - Tamanho máximo por entrada da tabela trabalha_em

Atributo	Tamanho máximo (Bytes)
Id	4
Nome	64
Salário	6

Tabela 15 - Tamanho máximo por entrada da tabela Funcionário.

- **Tamanho dos registos nas tabelas da Base de Dados**

Através das tabelas anteriores conseguimos calcular o tamanho que cada registo ocupa em cada tabela da Base de Dados. Com esta informação é possível estimar o tamanho das tabelas e como o crescimento dos dados irá afetar o espaço utilizado em disco.

Tabela	Tamanho (Bytes)
Categoria	49
Utilizador	124
E_visitado_por	24
Atracao	79
Trabalha_em	24
Funcionário	76

Tabela 16 - Tamanho máximo de cada entrada por Tabela.

- **Tamanho inicial e escalabilidade do sistema**

Considerando um caso de estudo inicial de grande dimensão que pretende representar um cenário real e fazendo consulta da tabela anteriormente apresentada, pode-se inferir facilmente um tamanho inicial para a base de dados em questão.

Assume-se um universo de 1 milhão de utilizadores que, em média, visitam as 13 atrações do parque, e que as atrações são vigiadas pelos funcionários em turnos de 3 horas. No parque trabalham um total de 40 funcionários ao longo de 214 dias. O parque divide os clientes em 4 categorias (Infantil, Júnior, Normal e Sénior).

Tabela	Tamanho (Bytes)
Categoria	$49 \times 4 = 196$
Utilizador	$124 \times 1000000 = 124,000,000$
E_visitado_por	$24 \times 1000000 \times 18 = 432,000,000$
Atracao	$79 \times 13 = 1,027$
Trabalha_em	$24 \times 4 \times 214 = 20,544$
Funcionário	$76 \times 40 = 3,040$
Total	556,024,807

Tabela 17 - Tamanho total da BD com informação de 1 ano.

Tendo isto em conta, no primeiro ano de implementação do sistema até agora projetado para o Parque Aquático, este requeria 556 Gigabytes para armazenar toda a informação acima mencionada. Tendo em conta que o principal objetivo na adoção deste sistema é o aumento do número de utilizadores do Parque, é importante ter em consideração que os requisitos físicos para o armazenamento desta quantidade de informação devem estar preparados para um aumento considerável de ano para ano da capacidade de armazenamento requerida pelo sistema projetado.

5.7. Definição e caracterização das vistas de utilização em SQL

Tendo em conta uma possível utilização do sistema em painéis publicitários digitais, optou-se pela implementação de algumas vistas de utilização que permitem a qualquer instante obter uma série de informações úteis relativamente ao estado atual do Parque. Descrevem-se de seguida:

```
Create View estado_fila_atracoes
AS SELECT V.Atracao_Id as "Atração",
count(V.Utilizador_Id) as "Nº de utilizadores em espera" from e_visitada_por as V
where (V.data_entrada_atracao = null);
```

Figura 23 - Vista do estado da fila de espera das diferentes atrações

```
Create View categoria_numbers
AS SELECT Utilizador.Categoria_Id as "Categoria" ,
COUNT(Utilizador.Categoria_Id) as "Nº de visitas" FROM Utilizador
INNER JOIN Categoria ON Categoria.Id=Utilizador.Categoria_Id
GROUP BY (Categoria.Id);
```

Figura 24 - Vista do nº de utilizadores de cada categoria

```
Create View top5_atracoes
as SELECT A.Designacao as "Nome da Atração",
COUNT(A.Id) as "Nº de visitantes" FROM Atracao As A
INNER JOIN e_visitada_por ON e_visitada_por.Atracao_Id= A.Id
WHERE (e_visitada_por.Data_entrada_atracao IS NOT NULL)
GROUP BY (A.Id)
ORDER BY COUNT(A.Id) DESC
Limit 5;
```

Figura 25 - Vista do Top 5 de atrações mais visitadas

5.8. Definição e caracterização dos mecanismos de segurança em SQL

De forma a protegermos o nosso sistema de usos indevidos, optamos por criar diferentes perfis de acordo com a utilização da base de dados.

```
create role marketing, funcionario, sensor;
grant select on ParqueAquatico.Utilizador to marketing;
grant select on ParqueAquatico.Atracao to marketing;
grant select on ParqueAquatico.e_visitada_por to marketing;

grant insert on ParqueAquatico.Utilizador to funcionario;
grant select on ParqueAquatico.Categoria to funcionario;

grant insert on ParqueAquatico.trabalha_em to sensor;
grant insert on ParqueAquatico.e_visitada_por to sensor;
grant update on ParqueAquatico.e_visitada_por to sensor;
grant update on ParqueAquatico.Utilizador to sensor;
```

Figura 26 - Privilégios de acordo com o perfil

Para os utilizadores do Departamento de Marketing, é permitida a consulta de informação relativa aos Utilizadores, Atrações e visitas dos Utilizadores a Atrações, já que é a única informação relevante para o desempenho das suas atividades publicitárias e de promoção da imagem do Parque.

Para os funcionários, é permitida a inserção de novos Utilizadores no sistema, já que necessitam dessa funcionalidade aquando da compra dos bilhetes por parte dos Utilizadores. Para além disso, também podem consultar os diferentes tipos de Categoria de bilhetes que a Administração definiu para venda.

De forma a limitar também um possível acesso indevido na parte do *hardware* instalado à entrada do Parque e das Atrações, também foi criado um perfil específico para esses sensores, podendo apenas inserir novos dados na base de dados e atualizar o perfil do Utilizador aquando da sua saída/entrada no Parque e atualizar o estado de uma visita a uma Atração, quando o Utilizador entra na Atração em si (saindo da fila).

Note-se também que apenas o perfil de Administração permite a eliminação de registos da base de dados, uma vez que a integridade e persistência dos dados é um fator chave para o sucesso deste sistema.

```
create user if not exists funcionarioOne  
identified with sha256_password by 'IRanOutOfIdeas34';
```

Figura 27 - Criação de um user Funcionário

Para além disso, de forma a manter estes utilizadores seguros, as palavras-passe de cada *user* são implementadas com recurso ao método de autenticação SHA-256. Na Figura 27 - Criação de um user Funcionário, é identificado o comando que cria um utilizador Funcionário a título de exemplo do que é feito para os restantes utilizadores referidos anteriormente.

5.9. Revisão do sistema implementado com o utilizador

Após terminada a implementação física, foi apresentado o sistema final ao Conselho de Administração do Parque Aquático. Após povoarem a base de dados com alguns dados guardados em ficheiros *.txt*, testaram o correto funcionamento dos procedimentos implementados e verificaram também que a documentação produzida correspondia ao que havia sido encomendado. Por isso, aprovaram o projeto desenvolvido e aguardam agora pela apresentação de uma alternativa ao sistema relacional, para que possam implementar a melhor das opções a tempo da próxima época balnear.

6. Abordagem NoSQL

6.1. Introdução e objetivos

Tendo em conta que o sistema pretendido envolve uma grande quantidade de escritas na base de dados, consideramos útil o desenvolvimento de um sistema *NoSQL* recorrendo à plataforma *Neo4J*, orientada a grafos.

Desenvolvendo este sistema, será possível armazenar a grande quantidade de informação gerada diariamente no Parque Aquático de uma forma adaptada ao esperado rápido crescimento, de acordo com a direção. Por isso, desenvolvemos um sistema comparável com o relacional, respondendo exatamente às mesmas interrogações propostas em 5.3. .

Como o sistema é comparável, as aplicações e utilizadores poderão ser os mesmos que na base de dados relacional, estando ambas a funcionar simultaneamente, mas tendo em conta que esta recebe a informação diariamente à meia-noite armazenada no sistema em *MySQL*. Por isso, para analisar a utilização do Parque recorrendo a este sistema, a análise deverá ser feita para dias anteriores ao dia em que a consulta está a ser realizada.

6.2. Estrutura base do sistema

Tendo em conta que o sistema utilizado é baseado em grafos, definimos a seguinte estrutura para a base de dados:

1. **Nodos**

- a. Utilizador (mapeamento direto da tabela Utilizador no sistema relacional)
- b. Categoria (mapeamento direto da tabela Categoria no sistema relacional)
- c. Atração (mapeamento direto da tabela Atração no sistema relacional)
- d. Funcionário (mapeamento direto da tabela Funcionário no sistema relacional)

2. **Relacionamentos**

- a. Pertence_a (mapeamento entre a chave estrangeira Categoria_Id da tabela Utilizador e a chave primária da tabela Categoria. Assim, cada nodo Utilizador pertence a um nodo Categoria)
- b. E_visitada_por (mapeamento entre a chave estrangeira Atracao_Id da tabela E_visitada_por e a chave primária da tabela Atração; e a chave estrangeira

Utilizador_Id da tabela E_visitada_por e a chave primária da tabela Utilizador. Assim, cada nodo Atração é visitado por um nodo Utilizador)

- c. Trabalha_em (mapeamento entre a chave estrangeira Atracao_Id da tabela Trabalha_em e a chave primária da tabela Atração; e a chave estrangeira Funcionário_Id da tabela Trabalha_em e a chave primária da tabela Funcionário. Assim, cada nodo Funcionário trabalha num nodo Atração).

6.3. Migração dos dados

De forma a migrar a informação presente no sistema *MySQL* para o sistema *Neo4J*, exporta-se o povoamento das relações existentes para ficheiros *.csv*.

Para isso, foram definidas 5 instruções SQL cujo *Result Set* é a tabela que se pretende exportar. Para a migração inicial, usou-se a opção “Exportar resultados” (que se encontra no barra de ferramentas, na parte relativa a *Queries*).

```
-- Exportar tabela "Utilizador"
SELECT Id, Nome, Nacionalidade, DATE_FORMAT(Hora_entrada_parque, '%Y-%m-%dT%H:%M:%S') AS Hora_entrada_parque, DATE_FORMAT(Hora_saida_parque, '%Y-%m-%dT%H:%M:%S') AS Hora_saida_parque,
Categoria_Id, N_Atracoes_Visitadas
FROM Utilizador;

-- Exportar tabela "Categoria"
SELECT * FROM Categoria;

-- Exportar tabela "Atração"
SELECT * FROM Atracao;

-- Exportar tabela "e_visitada_por"
SELECT DATE_FORMAT(Data_entrada_fila, '%Y-%m-%dT%H:%M:%S') AS Data_entrada_fila, DATE_FORMAT(Data_entrada_atracao, '%Y-%m-%dT%H:%M:%S') AS Data_entrada_atracao, Atracao_Id, Utilizador_Id
FROM e_visitada_por;

-- Exportar tabela "trabalha_em"
SELECT DATE_FORMAT(Data_de_inicio, '%Y-%m-%dT%H:%M:%S') AS Data_de_inicio, DATE_FORMAT(Data_de_fim, '%Y-%m-%dT%H:%M:%S') AS Data_de_fim, Funcionario_Id, Atracao_Id
FROM trabalha_em;
```

Figura 28 – Instruções *MySQL* para exportação do povoamento da base de dados

Findo esse processo, os ficheiros *.csv* gerados deverão ser colocados na pasta *import*, localizada na raiz do programa *Neo4J*. De seguida, esses ficheiros serão importados utilizando a linguagem *Cypher*, através das instruções que se apresentam de seguida:

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///Utilizador.csv" AS row
3 CREATE (:Utilizador {id: toInteger(row.Id), nome: row.Nome, nacionalidade: row.Nacionalidade, hora_entrada_parque:
row.Hora_entrada_parque, hora_saida_parque: row.Hora_saida_parque, n_atracoes_visitadas:
toInteger(row.N_Atracoes_Visitadas)});
```

Figura 29 – Carregamento da tabela Utilizador

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///Categoria.csv" AS row
3 CREATE (:Categoria {id: toInteger(row.Id), designacao: row.Designacao, idade_inferior:
toInteger(row.Idade_inferior), idade_superior: toInteger(row.Idade_superior)});
```

Figura 30 - Carregamento da tabela Categoria

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///Funcionario.csv" AS row
3 CREATE (:Funcionario {id: toInteger(row.Id), nome: row.Nome, salario: toFloat(row.Salario)});
```

Figura 31 - Carregamento da tabela Funcionário

```

1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///Atracao.csv" AS row
3 CREATE (:Atracao {id: toInteger(row.Id), designacao: row.Designacao, zona: row.Zona, capacidade:
  toInteger(row.Capacidade), duracao: row.Duracao, altura_minima: toInteger(row.Altura_Minima)});

```

Figura 32 - Carregamento da tabela Atração

```

1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///Utilizador.csv" AS row
3 MATCH (categoria:Categoria {id: toInteger(row.Categoria_Id)})
4 MATCH (utilizador:Utilizador {id: toInteger(row.Id)})
5 MERGE (utilizador)-[:PERTENCE_A]->(categoria);

```

Figura 33 - Carregamento da relação entre Utilizador e Categoria

```

1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///e_visitada_por.csv" AS row
3 MATCH (atracao:Atracao {id: toInteger(row.Atracao_Id)})
4 MATCH (utilizador:Utilizador {id: toInteger(row.Utilizador_Id)})
5 MERGE (atracao)-[:E_VISITADA_POR {data_entrada_fila: row.Data_entrada_fila, data_entrada_atracao:
  row.Data_entrada_atracao}]->(utilizador);

```

Figura 34 - Carregamento da relação entre Utilizador e Atração

```

1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///trabalha_em.csv" AS row
3 MATCH (atracao:Atracao {id: toInteger(row.Atracao_Id)})
4 MATCH (funcionario:Funcionario {id: toInteger(row.Funcionario_Id)})
5 MERGE (funcionario)-[:TRABALHA_EM {data_de_inicio: row.Data_de_Inicio, data_de_fim: row.Data_de_Fim}]->(atracao);

```

Figura 35 - Carregamento da relação entre Funcionário e Atração

Após estas instruções serem executadas, a base de dados no sistema *NoSQL* encontra-se povoada e pronta a ser usada.

6.4. Tradução das interrogações para *Cypher*

Uma vez que replicamos todo o povoamento no sistema relacional para o sistema orientado a grafos, também traduzimos todas as interrogações definidas em 2.2.2 para *Cypher*. Dada a extensão das mesmas, optamos por incluí-las no Anexo 3 – Interrogações em *Cypher*.

6.5. Conclusões sobre a abordagem *NoSQL*

Finalizada a implementação do sistema em *Neo4J*, apresentam-se então algumas conclusões sobre o trabalho desenvolvido.

Em primeiro lugar, foi possível recriar o modelo definido para o sistema relacional no sistema orientado a grafos, o que foi um ponto positivo e favorável ao rápido desenvolvimento deste último. Por isso mesmo, também optamos por passar as interrogações estabelecidas na íntegra para o sistema *NoSQL*.

Apesar disso, a inexistência de procedimentos similares aos disponibilizados em *MySQL* revelou-se um entrave à replicação 100% idêntica, tendo de se alterar os parâmetros de cada interrogação em cada um dos comandos executados.

Em segundo lugar, a baixa utilização deste sistema a nível global, comparativamente com um sistema *MySQL*, também gerou mais dificuldades na procura de informação relevante para resolver alguns problemas aquando da definição das interrogações em *Cypher* e também no que toca ao carregamento dos dados do sistema relacional para o orientado a grafos.

Ainda assim, foi possível atingir os objetivos pretendidos aquando do planeamento deste novo sistema, e poderá funcionar como um complemento ou alternativa ao sistema relacional anteriormente desenvolvido.

7. Conclusões e Trabalho Futuro

A implementação da base de dados apresentada neste relatório resulta de uma abordagem minuciosa para resolver o problema em causa. Acreditamos que esta solução, apesar da sua simplicidade, cumpriu os principais objetivos. No entanto, numa análise mais cuidadosa foi possível reparar nalguns pontos menos positivos, bem como pontos fortes.

Começando pelos pontos fortes, consideramos que o facto de o sistema não estar excessivamente complexo pode ser considerado uma vantagem, uma vez que revela uma maior simplicidade que possibilita a expansão do mesmo. Ainda assim, contém diversidade suficiente para aplicar os conhecimentos adquiridos ao longo das aulas de Base de Dados. Para além disso, foi possível responder a todas as interrogações definidas inicialmente.

Relativamente aos pontos fracos, será importante ressaltar que não foi definido qualquer atributo multivalorado, uma vez que não consideramos necessário em qualquer parte do sistema.

Quanto ao trabalho futuro, será possível adicionar o serviço de cafetaria também ao sistema, com o respetivo stock de produtos. Uma outra possível adição seria a de incluir também a loja de merchandising, também com o stock de produtos.

Conclui-se, portanto, que os diversos passos de modelação se apresentam como ferramentas essenciais para a correta definição de uma Base de Dados que se adeque aos requisitos indicados pela entidade que procura uma implementação do sistema de informação.

Referências

Connolly, T. and Begg, C. (2005). *Database Systems, A Practical Approach to Design, Implementation, and Management*. 4^a ed. Addison-Wesley.

Lista de Siglas e Acrónimos

BD	Base de Dados
UC	Unidade Curricular
ER	Entidade Relacionamento

Anexos

I. Anexo 1 – Script completo de Criação

```
-- MySQL Script generated by MySQL Workbench
-- qui 22 nov 2018 11:20:59 WET
-- Model: Parque Aquático Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE
RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema ParqueAquatico
-- -----

DROP SCHEMA IF EXISTS `ParqueAquatico` ;

-- -----
-- Schema ParqueAquatico
-- -----

CREATE SCHEMA IF NOT EXISTS `ParqueAquatico` ;
USE `ParqueAquatico` ;

-- -----
-- Table `ParqueAquatico`.`Atracao`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`Atracao` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Atracao` (
  `Id` INT NOT NULL DEFAULT 0,
  `Designacao` VARCHAR(32) NOT NULL DEFAULT 'ASN (atração sem nome)',
  `Zona` VARCHAR(32) NOT NULL DEFAULT 'ZSN (zona sem nome)',
  `Capacidade` INT NOT NULL DEFAULT 0,
```



```

`Duracao` TIME NOT NULL DEFAULT 0,
`Altura_Minima` INT NOT NULL DEFAULT 0,
PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `ParqueAquatico`.`Categoria`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`Categoria` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Categoria` (
  `Id` INT NOT NULL DEFAULT 0,
  `Designacao` VARCHAR(32) NOT NULL DEFAULT 'CSN (categoria sem
nome)',
  `Preco` DECIMAL(5,2) NOT NULL DEFAULT 0,
  `Idade_inferior` INT NOT NULL DEFAULT 0,
  `Idade_superior` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `ParqueAquatico`.`e_visitada_por`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`e_visitada_por` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`e_visitada_por` (
  `Data_entrada_fila` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Data_entrada_atracao` DATETIME NULL DEFAULT NULL,
  `Atracao_Id` INT NOT NULL,
  `Utilizador_Id` INT NOT NULL,
  PRIMARY KEY (`Data_entrada_fila`, `Atracao_Id`, `Utilizador_Id`),
  INDEX `fk_e_visitada_por_Atracao1_idx` (`Atracao_Id` ASC) VISIBLE,
  INDEX `fk_e_visitada_por_Utilizador1_idx` (`Utilizador_Id` ASC)
VISIBLE,
  CONSTRAINT `fk_e_visitada_por_Atracao1`
    FOREIGN KEY (`Atracao_Id`)
    REFERENCES `ParqueAquatico`.`Atracao` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,

```

```

CONSTRAINT `fk_e_visitada_por_Utilizador1`
  FOREIGN KEY (`Utilizador_Id`)
  REFERENCES `ParqueAquatico`.`Utilizador` (`Id`)
  ON DELETE RESTRICT
  ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----
-- Table `ParqueAquatico`.`Funcionário`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`Funcionário` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Funcionário` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(64) NOT NULL DEFAULT 'FSN (funcionário sem nome)',
  `Salario` DECIMAL(6,2) NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `ParqueAquatico`.`trabalha_em`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`trabalha_em` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`trabalha_em` (
  `Data_de_Inicio` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Data_de_Fim` DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  `Funcionário_Id` INT NOT NULL,
  `Atracao_Id` INT NOT NULL,
  PRIMARY KEY (`Data_de_Inicio`, `Funcionário_Id`, `Atracao_Id`),
  INDEX `fk_trabalha_em_Funcionário_idx` (`Funcionário_Id` ASC)
  VISIBLE,
  INDEX `fk_trabalha_em_Atracao1_idx` (`Atracao_Id` ASC) VISIBLE,
  CONSTRAINT `fk_trabalha_em_Funcionário`
    FOREIGN KEY (`Funcionário_Id`)
    REFERENCES `ParqueAquatico`.`Funcionário` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_trabalha_em_Atracao1`

```

```

        FOREIGN KEY (`Atracao_Id`)
        REFERENCES `ParqueAquatico`.`Atracao` (`Id`)
        ON DELETE RESTRICT
        ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----
-- Table `ParqueAquatico`.`Utilizador`
-- -----

DROP TABLE IF EXISTS `ParqueAquatico`.`Utilizador` ;

CREATE TABLE IF NOT EXISTS `ParqueAquatico`.`Utilizador` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(64) NOT NULL DEFAULT 'USN (utilizador sem nome)',
  `Nacionalidade` VARCHAR(32) NOT NULL DEFAULT 'USNA (sem
nacionalidade)',
  `Hora_entrada_parque` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Hora_saida_parque` DATETIME NULL DEFAULT NULL,
  `Categoria_Id` INT NOT NULL,
  `N_Atracoes_Visitadas` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Id`),
  INDEX `fk_Utilizador_Categorial_idx` (`Categoria_Id` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_Categorial`
    FOREIGN KEY (`Categoria_Id`)
    REFERENCES `ParqueAquatico`.`Categoria` (`Id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

II. Anexo 2 – Interrogações em SQL

```
-- 1. Obter uma listagem dos utilizadores
-- que frequentaram uma atração num intervalo de tempo;
USE ParqueAquatico;

DELIMITER //

DROP PROCEDURE IF EXISTS whoVisited //

CREATE PROCEDURE whoVisited(id INT, fromWhen DATETIME, toWhen
DATETIME)
BEGIN
    SELECT distinct U.Id "Id do Visitante", U.Nome as "Nome do
Visitante" from Utilizador as U
    INNER JOIN e_visitada_por as V on U.Id = V.Utilizador_Id
    WHERE (V.Data_entrada_fila between fromWhen and toWhen)
        and id = V.Atracao_Id;
END //

-- 2. Obter o tempo médio de espera
-- dos utilizadores de uma atração num intervalo de tempo;
DELIMITER //

DROP FUNCTION IF EXISTS averageWait //

CREATE FUNCTION averageWait(id INT, fromWhen DATETIME, toWhen
DATETIME)
    RETURNS TIME
    DETERMINISTIC
BEGIN
    DECLARE result TIME(0);
```

```

        SELECT
sec_to_time(avg(time_to_sec(timediff(V.Data_entrada_atracao,V.Data_entrada_fila)))) as "Média" from e_visitada_por as V
        WHERE (V.Data_entrada_fila between fromWhen and toWhen)
            and id = V.Atracao_Id
            and (V.Data_entrada_atracao between fromWhen and toWhen)
            and (V.Data_entrada_atracao is not null)
        into result;
    RETURN result;
END //
```

```

-- 3. Obter o número de utilizadores em fila numa atração
-- num intervalo de tempo;
DELIMITER //
```

```

DROP FUNCTION IF EXISTS countWaiting //
```

```

CREATE FUNCTION countWaiting(idAtracao INT, whenWait DATETIME)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE result INT;
    SELECT count(V.Utilizador_Id) as "Nº de utilizadores em espera"
from e_visitada_por as V
    where (V.data_entrada_atracao = null)
    and idAtracao = V.Atracao_Id
    and (V.Data_entrada_fila <= whenWait) into result;
    RETURN result;
END //
```

```

-- 4 Obter uma listagem de utilizadores de uma categoria;
DELIMITER //
```

```

DROP PROCEDURE IF EXISTS UtilizadorCat //
```

```

CREATE PROCEDURE UtilizadorCat(id INT)
BEGIN
    SELECT distinct U.Id "Id do Visitante", U.Nome "Nome do
Visitante" from Utilizador as U
    WHERE (id = U.Categoria_Id);
```

```
END //
```

```
-- 5. Obter uma listagem das atrações mais visitadas por utilizadores  
de uma categoria;
```

```
DELIMITER $$
```

```
drop Procedure if exists AtracoesMaisVisitadasCat $$
```

```
CREATE PROCEDURE AtracoesMaisVisitadasCat (id Int)
```

```
BEGIN
```

```
    SELECT Atracao.Designacao, COUNT(e_visitada_por.Utilizador_Id)  
as "Nº de visitas" FROM Utilizador
```

```
    INNER JOIN e_visitada_por ON  
e_visitada_por.Utilizador_Id=Utilizador.Id
```

```
    INNER JOIN Atracao ON Atracao.Id=e_visitada_por.Atracao_Id
```

```
    WHERE (Utilizador.Categoria_Id=id)
```

```
    GROUP BY (Atracao.Id)
```

```
    ORDER BY COUNT(e_visitada_por.Utilizador_Id) DESC;
```

```
END
```

```
$$
```

```
-- 6. Obter a hora de entrada média dos utilizadores de uma categoria;
```

```
DELIMITER //
```

```
DROP FUNCTION IF EXISTS averageEntry//
```

```
CREATE FUNCTION averageEntry(id INT)
```

```
    RETURNS TIME
```

```
    DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE result TIME(0);
```

```
    SELECT sec_to_time(avg(time_to_sec(U.Hora_Entrada_parque))) as  
"Média" from Utilizador as U
```

```
    WHERE U.Categoria_Id=id
```

```
    into result;
```

```
    RETURN result;
```

```

END //

-- 7. Obter o número total de utilizadores que visitaram o parque num
intervalo de tempo em dias (inclusive);

DELIMITER //

DROP FUNCTION IF EXISTS NumUtilizadoresAtTime //

CREATE FUNCTION  NumUtilizadoresAtTime (inicio DATE, fim DATE)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE result INT;
    SELECT count(U.Id) from Utilizador as U
    WHERE ((Date(U.Hora_entrada_parque) )BETWEEN inicio and fim)
    into result;
    RETURN result;
END //

-- 8.      Obter o número total de utilizadores que visitaram o
parque por categoria num intervalo de tempo em dias (inclusive);

DELIMITER //

DROP FUNCTION IF EXISTS NumUtilizadoresAtTimeCat //

CREATE FUNCTION  NumUtilizadoresAtTimeCat (id INT, inicio DATE, fim
DATE)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE result INT;
    SELECT count(U.Id) from Utilizador as U
    WHERE ((Date(U.Hora_entrada_parque) )BETWEEN inicio and fim) AND
(U.Categoria_Id= id)
    into result;
    RETURN result;
END //

```

-- 9. Obter o top n utilizadores que mais frequentaram as atrações num dia;

DELIMITER //

DROP PROCEDURE IF EXISTS BestUsers //

CREATE PROCEDURE BestUsers (day DATE, limite INT)

BEGIN

SELECT U.Nome as "Nome" , U.N_Atracoes_Visitadas as "N° atrações visitadas" FROM Utilizador As U

WHERE (Date(U.Hora_entrada_parque)=day)

ORDER BY N_Atracoes_Visitadas DESC

LIMIT limite;

END //

-- 10. Obter as atrações mais visitadas num determinado intervalo de tempo;

DELIMITER //

DROP PROCEDURE IF EXISTS attractionsMostVisitedByTime //

CREATE PROCEDURE attractionsMostVisitedByTime (fromWhen DATETIME,

toWhen DATETIME)

BEGIN

SELECT A.Designacao as "Nome da Atração", COUNT(A.Id) as "N° de visitantes" FROM Atracao As A

INNER JOIN e_visitada_por ON e_visitada_por.Atracao_Id= A.Id

WHERE (e_visitada_por.Data_entrada_atracao IS NOT NULL) AND

(e_visitada_por.Data_entrada_atracao BETWEEN fromWhen AND toWhen)

GROUP BY (A.Id)

ORDER BY COUNT(A.Id) DESC;

END //

-- 11. Obter uma listagem de todos os utilizadores que frequentaram o Parque, por ordem decrescente de tempo permanecido no Parque;


```
DELIMITER //
```

```
DROP PROCEDURE IF EXISTS allUsersByTimeOnPark //
```

```
CREATE PROCEDURE allUsersByTimeOnPark ()
```

```
BEGIN
```

```
    SELECT U.Nome as "Nome do Utilizador",  
    timediff(U.Hora_saida_parque,U.Hora_entrada_parque) as "Tempo que  
    permaneceu no Parque" FROM Utilizador AS U  
    WHERE U.Hora_saida_parque IS NOT NULL  
    ORDER BY timediff(U.Hora_saida_parque,U.Hora_entrada_parque) DESC;  
END //
```

-- 12. Obter a designação da categoria do maior número de visitantes da atração monitorizada por um dado funcionário, num dado turno.

```
DELIMITER //
```

```
DROP PROCEDURE IF EXISTS categoryMostVisitedOnShift //
```

```
CREATE PROCEDURE categoryMostVisitedOnShift (funcionario INT, shift  
DATETIME)
```

```
BEGIN
```

```
    SELECT Categoria.Designacao as "Nome da Categoria ",  
    COUNT(Categoria.Designacao) as "Nº visitantes da categoria" FROM  
    trabalha_em AS T  
    INNER JOIN Atracao ON Atracao.Id=T.Atracao_Id  
    INNER JOIN e_visitada_por ON e_visitada_por.Atracao_Id=Atracao.Id  
    INNER JOIN Utilizador ON Utilizador.Id=  
    e_visitada_por.Utilizador_Id  
    INNER JOIN Categoria On Categoria.Id = Utilizador.Categoria_Id  
    WHERE (T.Data_de_Fim IS NOT NULL)  
    AND (T.Funcionário_Id=funcionario)  
    AND (shift BETWEEN T.Data_de_Inicio AND T.Data_de_Fim)  
    AND (e_visitada_por.Data_entrada_atracao IS NOT NULL)  
    AND (e_visitada_por.Data_entrada_atracao BETWEEN T.Data_de_Inicio  
    AND T.Data_de_Fim)  
    GROUP BY (Categoria.Id)  
    ORDER BY COUNT(Categoria.Designacao) DESC;  
END //
```

III. Anexo 3 – Interrogações em Cypher

// 1. Obter uma listagem dos utilizadores que frequentaram uma atração num intervalo de tempo;

```
MATCH (a:Atracao)-[r:E_VISITADA_POR]->(u:Utilizador)
WHERE a.id = 1 AND localdatetime(r.data_entrada_fila) >=
localdatetime('2017-06-15T09:5:20') AND
localdatetime(r.data_entrada_fila) <= localdatetime('2017-06-
15T18:5:00')
RETURN u.id AS Id, u.nome AS Nome
ORDER BY u.id
```

// 2. Obter o tempo médio de espera dos utilizadores de uma atração num intervalo de tempo;

```
MATCH q1 = (a:Atracao)-[r:E_VISITADA_POR]->(u:Utilizador)
WHERE a.id = 1 AND localdatetime(r.data_entrada_fila) >=
localdatetime('2017-06-15T09:05:20') AND
localdatetime(r.data_entrada_fila) <= localdatetime('2017-06-
15T18:05:00')
WITH duration.between(localdatetime(r.data_entrada_fila),
localdatetime(r.data_entrada_atracao)) AS diff
WITH AVG(diff) AS time
RETURN time.minutesOfHour + ':' + time.secondsOfMinute AS TempoMedio;
```

// 3. Obter o número de utilizadores em fila numa atração num intervalo de tempo;

```
MATCH q1 = (a:Atracao)-[r:E_VISITADA_POR]->(u:Utilizador)
WHERE a.id = 1 AND localdatetime(r.data_entrada_fila) >=
localdatetime('2017-06-15T10:05:20') AND
localdatetime(r.data_entrada_atracao) = null
```

```
RETURN COUNT(*) AS PessoasEmFila;
```

```
// 4. Obter uma listagem de utilizadores de uma categoria;
```

```
MATCH (u:Utilizador)-[r:PERTENCE_A]->(c:Categoria)
WHERE c.id = 2 // inserir aqui ID pretendido
RETURN u.id AS IdVisitante, u.nome AS NomeVisitante;
```

```
// 5. Obter uma listagem das atrações mais visitadas por utilizadores
de uma categoria;
```

```
MATCH (a:Atracao)-[r:E_VISITADA_POR]-(u:Utilizador)-[p:PERTENCE_A]-
(c:Categoria)
WHERE c.id = 2
RETURN a.designacao AS Atracao, COUNT(u) AS Visitas
ORDER BY COUNT(u) DESC
```

```
// 6. Obter a hora de entrada média dos utilizadores de uma categoria;
```

```
MATCH (u:Utilizador)-[r:PERTENCE_A]->(c:Categoria)
WHERE c.id = 2 // inserir aqui ID pretendido
with localtime(u.hora_entrada_parque) AS entrada
with AVG(duration({hours: entrada.hour, minutes: entrada.minute,
seconds: entrada.second})) AS media
RETURN media.hours + ':' + media.minutesOfHour + ':' +
media.secondsOfMinute AS HoraMediaEntrada;
```

```
// 7. Obter o número total de utilizadores que visitaram o parque num
intervalo de tempo em dias (inclusive);
```

```
MATCH (u:Utilizador)
WITH localtime(u.hora_entrada_parque) AS entrada
WHERE entrada >= localtime('2017-06-15T09:00:00') AND entrada <=
localtime('2017-06-15T19:00:00')
RETURN COUNT(entrada) AS NumUtilizadores;
```

// 8. Obter o número total de utilizadores que visitaram o parque por categoria num intervalo de tempo em dias (inclusive);

```
MATCH (u:Utilizador)-[:PERTENCE_A]->(c:Categoria)
WHERE c.id = 1
with localtime(u.hora_entrada_parque) AS entrada
WHERE entrada >= localtime('2017-06-15T09:00:00') AND entrada <=
localtime('2017-06-15T19:00:00')
RETURN COUNT(entrada) AS NumUtilizadores;
```

// 9. Obter o top n utilizadores que mais frequentaram as atrações num dia;

```
MATCH (u:Utilizador)
WHERE localtime(u.hora_entrada_parque).year = 2017 AND
localtime(u.hora_entrada_parque).month = 6 AND
localtime(u.hora_entrada_parque).day = 15
RETURN u.nome AS Nome, u.n_atracoes_visitadas AS NAtacoesVisitadas
ORDER BY u.n_atracoes_visitadas DESC
LIMIT 5;
```

// 10. Obter as atrações mais visitadas num determinado intervalo de tempo;

```
MATCH (a:Atracao)-[r:E_VISITADA_POR]->(u:Utilizador)
WHERE localtime(r.data_entrada_atracao) >= localtime('2017-06-
15T09:00:00') AND localtime(r.data_entrada_atracao) <=
localtime('2017-06-17T23:00:00')
RETURN a.designacao AS NomeAtracao, COUNT(a.id) AS NumVisitas
ORDER BY NumVisitas DESC;
```

// 11. Obter uma listagem de todos os utilizadores que frequentaram o Parque, por ordem decrescente de tempo permanecido no Parque;

```
MATCH (u:Utilizador)
RETURN u
ORDER BY duration.between(localtime(u.hora_saida_parque),
localtime(u.hora_entrada_parque)) ASC;
```

// 12. Obter a designação da categoria do maior número de visitantes da atração monitorizada por um dado funcionário, num dado turno.

```
MATCH (f:Funcionario)-[t:TRABALHA_EM]-(a:Atracao)-[r:E_VISITADA_POR]-(u:Utilizador)-[p:PERTENCE_A]-(c:Categoria)
WHERE f.id = 1
AND localdatetime(r.data_entrada_fila) >=
localdatetime(t.data_de_inicio)
AND localdatetime(r.data_entrada_fila) <= localdatetime(t.data_de_fim)
AND localdatetime(t.data_de_inicio) <= localdatetime("2017-06-
15T18:00:00")
AND localdatetime(t.data_de_fim) >= localdatetime("2017-06-
15T09:00:00")
RETURN c.designacao AS Categoria, COUNT(c.id) AS NumVisitas
ORDER BY COUNT(c.id) DESC
```