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

Unidade Curricular de Bases de Dados

Ano Letivo de 2017/2018

Desenvolvimento de uma base de dados relacional
Jardim Zoológico

Carlos Pedrosa – a77320
David Sousa – a78938
Francisco Matos – a77688
Manuel Sousa – a78869

Novembro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Desenvolvimento de uma base de dados relacional
Jardim Zoológico

Carlos Pedrosa – a77320
David Sousa – a78938
Francisco Matos – a77688
Manuel Sousa – a78869

Novembro, 2017

Resumo

Este projeto tem como principal objetivo o desenvolvimento de um sistema de gestão de bases de dados relacional (SGBDR), que irá permitir administrar, de forma adequada, os recursos de um Jardim zoológico.

De forma a obter um modelo de base de dados viável, procuramos seguir um conjunto de passos bem definidos. Nesse sentido, começamos por definir o sistema sobre o qual se debruçaria a nossa base de dados. Assim, percorremos um longo caminho desde a contextualização até à viabilidade do projeto, passando pela fundamentação do mesmo. De seguida, procedemos ao levantamento de requisitos. Esta fase prende-se principalmente com a escolha de informação relevante e, por conseguinte, a eliminação de informação redundante fornecida pelo dono do jardim zoológico na entrevista realizada pelo grupo. Seguiu-se o desenho do modelo conceptual caracterizado por ser um modelo de dados independente da implementação lógica e física. A partir deste, concebeu-se o modelo lógico, modelo esse que refina o modelo conceptual. Por fim, gerou-se o modelo físico, onde já é possível a manipulação de dados. Foram consideradas ainda outras sub-etapas, como é o caso da normalização, que garantiram o desenvolver correto do trabalho.

Em suma, podemos concluir que o trabalho aqui apresentado cumpre as condições necessárias para que seja gerada uma base de dados simples de acordo com o problema inicialmente proposto.

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

Palavras-Chave: Sistema de gestão de bases de dados relacional, base de dados, contextualização, viabilidade, fundamentação, levantamento de requisitos, modelo conceptual, modelo lógico, manipulação de dados, modelo físico, normalização.

Índice

Anexos

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

1.1. Contextualização de aplicação do sistema

O Zoo do Paulo situado nos arredores da cidade de Lisboa pertence à família Sousa há várias gerações. O Zoo foi criado em 1957 pelo patriarca da família, com o intuito de se tornar um dos primeiros jardins zoológicos a operar em Portugal. Sendo este um negócio bastante arcaico, sempre foi gerido com recurso a tecnologias ultrapassadas para os dias que correm. Efetivamente, apesar da insistência das gerações mais novas, o proprietário da atividade que sustenta a família revelava-se bastante conservador quando confrontado com a mudança, provocando, deste modo, um entrave ao desenvolvimento. Paradoxalmente, as instalações mostravam-se bastante modernas, com jaulas de qualidade e decoradas de acordo com os habitats naturais dos animais, que fazem transparecer os bons cuidados a que estes estão sujeitos. Mesmo com obstáculos ao longo da sua vida, o zoo foi crescendo de década em década, tornando cada vez mais complicada a sua gestão, tanto em termos de biodiversidade, como, por consequência, em termos de visitantes.

1.2. Fundamentação da implementação da base de dados

Com o falecimento do patriarca foi necessário que houvesse a passagem do testemunho para um novo elemento da família. Era a vez do seu filho mais velho assumir a posição que seu pai tão bem desempenhou durante muitos anos. No entanto, o assumir deste cargo não foi tarefa fácil. Sendo este um adulto bastante inovador e um dos apologistas à informatização do Jardim Zoológico, não demorou muito a querer implementar um sistema que tornasse mais fácil a gestão deste, não só na atualidade, mas também no futuro.

1.3. Motivação e Objetivos

Uma das motivações que levou o novo dono do Zoo, Sr. Filipe Sousa, a recorrer a um grupo de engenheiros informáticos foi o facto de o seu falecido pai guardar todas as informações em formato de papel o que tornava difícil a sua compreensão, uma vez que o papel se tinha degradando ao longo dos anos. Outro dos motivos apresentado, foi o facto de este querer aumentar o número de espécies e de recursos humanos no negócio da família.

O principal objetivo na elaboração deste projeto é o desenvolvimento de um sistema de gestão de bases de dados relacional, que irá permitir administrar, de forma adequada, os recursos de um Jardim zoológico para que desta forma seja possível:

- Reduzir os desperdícios associados a alguns recursos, permitindo desta forma a maximização dos lucros;
- Tornar mais eficiente a gestão do jardim zoológico, não só na atualidade, mas também no futuro uma vez que, desta forma, torna-se mais fácil a compreensão dos recursos do negócio por todos.

1.4. Análise da viabilidade do processo

Desde o início o projeto foi visto como viável. Na verdade, a pequena dimensão do projeto leva a que seja possível o desenvolvimento do sistema num curto período de tempo, uma vez que este não requer grandes recursos quer financeiros quer humanos. Por conseguinte, espera-se que o dinheiro investido no novo sistema seja rapidamente recuperado com a gestão eficiente dos recursos.

1.5. Estrutura do Relatório

Depois de apresentado o contexto do problema, as suas motivações e a sua viabilidade segue-se agora os passos dados para o desenvolvimento da base de dados propriamente dita. Nesse sentido, começamos por levantar os requisitos. Posteriormente, desenhamos o modelo concetual, passando para o lógico terminando no físico através da utilização da ferramenta de “forward engineering”. Por último, este apresentará ainda uma secção dedicada à análise crítica do projeto sumarizando desta forma todos os esforços efetuados para alcançar o objetivo proposto.

2. Levantamento de Requisitos

2.1. Método de levantamento e análise de requisitos adotado

No intuito de identificar os requisitos necessários para a elaboração do projeto, reunimo-nos com o novo dono do Zoo, o Sr. Filipe Sousa. Nesse sentido, apresentamos em seguida um texto por nós elaborado onde se sintetiza aqueles que consideramos ser os aspetos mais importantes da entrevista feita ao novo dono do jardim zoológico. De facto, consideramos a entrevista o método mais adequado uma vez que se trata de um Zoo familiar e todas as informações necessárias podem ser obtidas em conjunto com o proprietário. Esta entrevista, pode ser consultada na sua integra nos anexos deste relatório.

2.2. Requisitos levantados

2.2.1 Requisitos de descrição

- **RD1:** Todos os animais devem ser registados no sistema;
- **RD2:** Os animais devem ser caracterizados por: nome, idade, género, hora da última refeição e espécie
- **RD3:** Todos os espaços devem ser registadas no sistema;
- **RD4:** Os espaços devem ser caracterizadas por: tamanho, quantidades de animais e tipo;
- **RD5:** Os funcionários atualmente empregues também devem ser registados no sistema;
- **RD6:** Os funcionários devem ser caracterizados por: nome, número de funcionário, data de nascimento, contactos e ainda morada e salário;
- **RD7:** O sistema deverá ser capaz de registar os alimentos consumidos por cada animal;
- **RD8:** Cada alimento deve ser caracterizado por: nome, quantidade e ainda a sua data de validade.

2.2.2 Requisitos de exploração

- **RE1:** O sistema deverá ser capaz de identificar os alimentos com défice de quantidade (≤100kg);
- **RE2:** O sistema deverá ser capaz de agrupar os funcionários de acordo com a cidade e organizá-los de acordo com o salário auferido;
- **RE3**: O sistema deverá ser capaz de identificar os espaços de um tipo específico com mais de uma certa quantidade de animais;
- **RE4:** O sistema deverá ser capaz de identificar quais os animais presentes num certo espaço;
- **RE5**: O sistema deverá ser capaz de identificar os animais que consumiram uma grande quantidade de alimento numa única refeição.
- **RE6**: O sistema deverá ser capaz de identificar os funcionários que forneceram determinado alimento aos animais.

2.2.3 Requisitos de controlo

- **RC1:** O dono do jardim zoológico deverá possuir controlo total à base dados;
- **RC2:** Os funcionários do Zoo não podem aceder a qualquer informação dos outros funcionários.

2.3. Análise geral dos requisitos

De acordo com a entrevista feita ao Sr. Filipe Sousa percebemos que este pretende privilegiar a gestão dos seus recursos, não estando interessado em obter qualquer informação relativa aos seus clientes. Assim, este referiu que pretendia consultar informação que envolvesse os seus animais, bem como, informação relativa aos seus funcionários.

3. Desenvolvimento do Modelo Conceptual

Depois de desenvolvidas todas as tarefas acima referidas, passamos para a conceção do modelo concetual. Assim, modelação concetual consiste no processo de construção de um modelo que é capaz de relacionar informação independentemente da implementação.

3.1. Apresentação da abordagem de modelação realizada

No desenvolvimento do modelo conceptual usaremos três peças chave. Nesse sentido, uma destas designa-se Entidade. Estas representam um grupo de objetos no "mundo real" com as mesmas propriedades e que existem independentemente. A segunda peça chave tem o nome de Atributo, e tem como objetivo a caracterização das entidades. Por último, é necessária uma peça que faça uma associação entre entidades, esta tem o nome de Relacionamento. Para a elaboração deste projeto usaremos a ferramenta *TerraER* que irá facilitar o desenho do modelo conceptual. Em suma, o problema será abordado todo de uma vez, isto é, com uma única vista de desenvolvimento.

3.2. Identificação e caracterização das entidades

Depois de analisados os requisitos chegamos a um conjunto de entidades que especificamos na seguinte tabela:

Nome da Entidade	Descrição	Sinónimo	Ação/Ocorrência
Alimento	Termo geral que descreve todas as características associadas a um alimento.	Ração	Cada alimento é consumido por vários animais.
Animal	Termo geral que descreve os seres vivos irracionais que habitam nos diferentes espaços do Zoo.	-	Cada animal tem um espaço associado, e um animal consome uma série de alimento.
Espaço	Termo geral que descreve o local onde os animais se encontram.	Local, Recinto, Área	Cada espaço tem associado vários animais.
Funcionário	Termo geral que descreve todos os utilizadores do sistema	Utilizador	Cada funcionário fornece os alimentos necessários para os animais.

Tabela 1 - Identificação das entidades

3.3. Identificação e caracterização dos relacionamentos

Nome da Entidade	Multiplicidade	Relação	Nome da Entidade	Multiplicidade
Alimento	1..*	É consumido	Animal	1..*
Espaço	1..*	Contém	Animal	1..*
Funcionário	1..*	Fornece	Alimento	1..*

Tabela 2-Caracterização dos relacionamentos

3.4. Identificação e caracterização das associações dos atributos com as entidades e relacionamentos

Entidade	Atributos	Descrição	Dados	Null	Multi-Valorado
Animal	idAnimal	Identifica unicamente um animal	Inteiro entre 1 e MAXINT	Não	Não
	Nome	Nome do animal	Variável até 15 caracteres	Sim	Não
	Especie	Espécie do Animal	Variável até 50 caracteres	Não	Não
	Genero	Género do Animal	2 Caracteres (M/F/ND)	Não	Não
	Última Refeição	Data da última refeição	DATETIME	Sim	Não
Alimento	Nome	Nome do Alimento	Variável até 20 caracteres	Não	Não
	Numero de serie	Número que identifica a série do alimento;	Variável até 20 caracteres	Não	Não
	Quantidade	Especifica a quantidade de alimento existente;	Inteiro entre 1 e MAXINT	Não	Não
	idAlimento	Identifica o alimento (valor único)	Inteiro entre 1 e MAXINT	Não	Não

Funcionário	idFuncionario	Identifica o funcionário (valor único)	Inteiro entre 1 e MAXINT;	Não	Não
	Nome	Nome do funcionário	Variável até 45 caracteres	Não	Não
	Salario	Salário do funcionário	Double com uma parte decimal com 2 dígitos	Sim	Não
	Morada -Detalhes	Código Postal/Rua/ Porta/…	Variável até 45 caracteres	Sim	Não
	-Cidade	Cidade onde o funcionário vive;	Variável até 20 caracteres	Não	Não
	Contacto	Contactos do funcionário.	Variável até 20 caracteres	Não	Sim
Espaço	Tamanho	Tamanho do espaço	Char, apenas um caracter	Não	Não
	Quantidade	Quantidade disponível para animais no espaço	Inteiro entre 1 e MAXINT	Não	Não
	Tipo	Tipo de espaço	Variável até 20 caracteres	Não	Não
	idEspaco	Identifica unicamente um espaço	Inteiro entre 1 e MAXINT	Não	Não

Tabela 3-Identificação e associação dos atributos com as entidades

3.5. Detalhe ou generalização de entidades

Depois de construído o modelo concetual procuramos verificar se haveria qualquer detalhe ou generalização das entidades envolvidas. No entanto, concluímos que estas não estavam presentes.

3.6. Apresentação e explicação do diagrama ER

O modelo concetual da nossa base de dados baseia-se na gestão dos recursos associados a um Jardim zoológico. Depois de devidamente analisados os requisitos que nos foram apresentados conseguimos inferir quatro entidades fundamentais nas quais se sustenta o nosso projeto. Tendo isto em conta, começamos por definir a entidade Animal. Esta representa um dos recursos fulcrais que o dono do Zoo gostaria de guardar. Nesse sentido, este indicou que seria importante associar a cada animal uma certa quantidade de características. Entre elas, temos a espécie que serve para identificar o tipo de animal de que se trata, segue-se o nome que permite singular cada um destes dentro da própria espécie, bem como o género que possibilita ao proprietário distinguir o sexo de cada animal. Por último, a hora da última refeição que tem como propósito identificar a última vez que um animal no Jardim Zoológico foi alimentado. Cada animal consome múltiplos alimentos e cada alimento pode ser consumido por vários animais. Deste modo, introduzimos a entidade Alimento, uma vez que este representa outro dos recursos que o proprietário do Zoo pretende gerir. No sentido a minimizar os desperdícios achamos deveras importante, controlar a quantidade de alimentos disponível. Para além disto, é também relevante guardar o nome destes e o número de série associado. O alimento é fornecido por um funcionário que é caracterizado pela morada, especifica-se assim, onde este habita. Outros aspetos importantes de se guardar, segundo o proprietário do Zoo, são os vários contactos que um funcionário possui bem como o nome e o salário que aufere. Numa última instância, consideramos que o espaço onde o animal se encontra deveria ser qualificado. Cada espaço tem um ou vários animais e cada animal encontra-se num único espaço. O tamanho deste, a quantidade e o tipo são fatores fundamentais para o funcionamento do Jardim Zoológico. Por motivos administrativos, decidimos acrescentar o id nas diferentes entidades para desta forma ser possível a sua individualização. Nesse sentido, este atributo funcionará como uma chave primária. No entanto, no decorrer de todo o processo, identificamos algumas chaves candidatas. No caso da entidade funcionário poderíamos ter o nome como uma chave candidata, mas,

poderíamos ter casos em que dois funcionários possuam mesmo nome pelo que este atributo não funcionaria como chave primária. No caso da entidade alimento o único atributo que poderia ser candidato a chave primária seria o número de série. Contudo, depois de nos informarmos com o nosso cliente apercebemo-nos que este é fornecido por diversas empresas e que empresas diferentes podem possuir o mesmo número de série. Já no animal, não se verifica nenhum atributo que permita a unicidade da entidade. Por último, na entidade espaço nenhum dos atributos serviria para se tornar uma chave primária, pois existem espaços com o mesmo tamanho, quantidade e tipo.

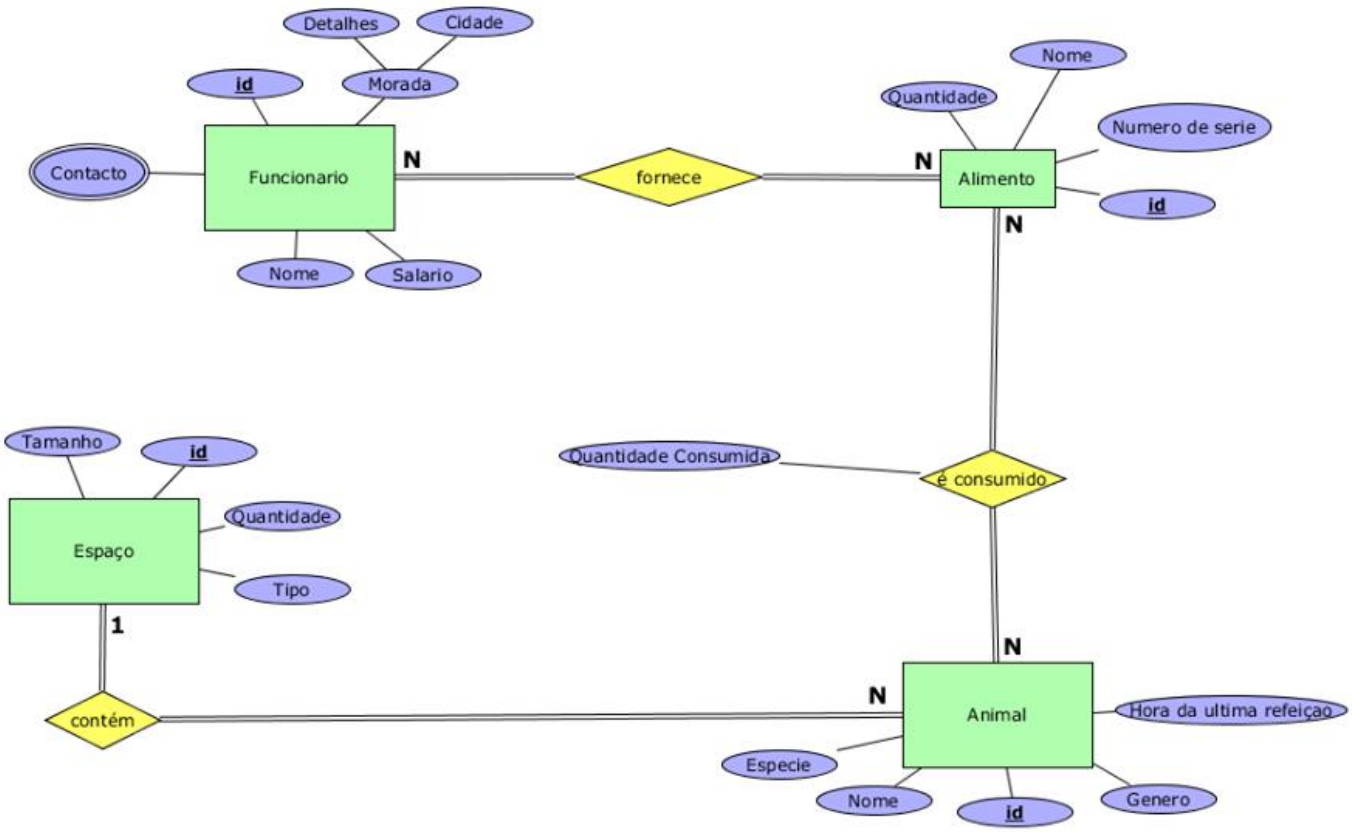


Ilustração 1 - Modelo Concetual

3.7. Validação do modelo de dados com o utilizador

Depois de efetuados todos os procedimentos necessários decidimos reunir-nos junto do nosso cliente para, deste modo, validarmos o modelo desenvolvido. Demonstramos todas as potencialidades do nosso projeto, indicando ao dono do Zoo tudo o que poderia fazer com a sua nova base de dados. Efetivamente, este indicou que o trabalho ia de encontro ao que inicialmente tinha pedido e que todos os propósitos tinham sido cumpridos.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

De forma a construir o modelo lógico, utilizamos o concetual para nos orientarmos, no entanto, apesar de, o modelo lógico ser equivalente a este, algumas opções têm de ser tomadas de modo independente por forma a garantir consistência e coerência nos dados. Neste sentido, especificamos de seguida todo o processo.

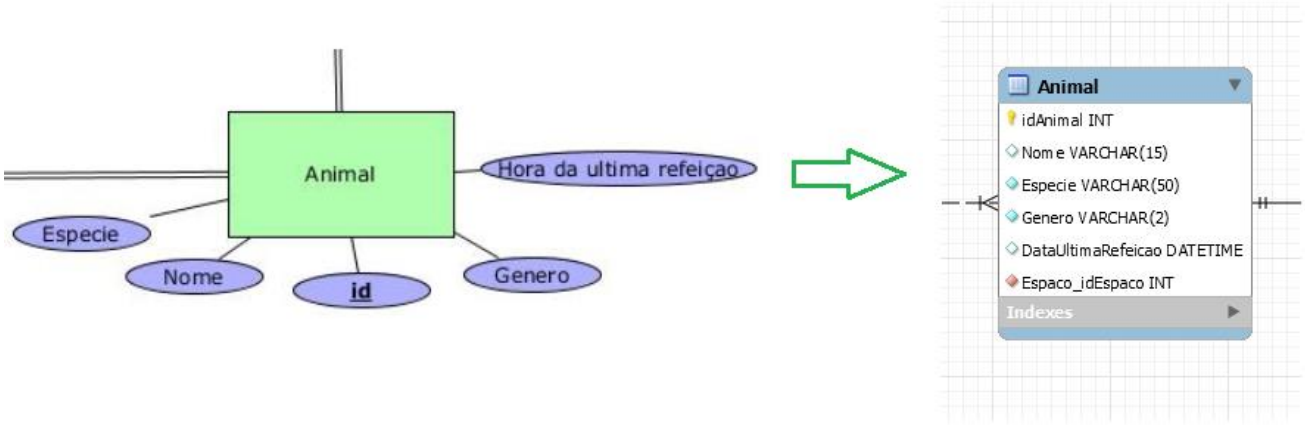


Ilustração 2 - Passagem da entidade animal para o modelo lógico

No que diz respeito ao Animal, o seu idAnimal será representado como um inteiro entre 1 e *MAXINT*, pois este será incrementado de acordo com o número de animais presentes no zoo, sendo uma chave primária, não poderá ter valor nulo. O seu nome será representado como um *VARCHAR(15)*, visto que o nome dado ao animal não será muito extenso, neste caso, o campo poderá ser nulo uma vez que durante o intervalo entre a sua adoção no zoo e a sua identificação nominal o animal será registado sem este reconhecimento. No caso do atributo especie decidimos que este seria representado por um *VARCHAR(50)* pois terá o nome em latim, o qual é

bastante extenso. Este não será nulo, visto que é importante para o dono do zoo estar ciente das espécies que possui. Relativamente ao género do animal, consideramos que este será identificado por apenas um caracter VAR*CHAR(2)* (M para masculino ou F para feminino e ainda ND para não definido para abranger casos em que os animais não tenham o sexo definido). O género segue a mesma lógica da espécie no caso da nulidade até porque, no momento da adoção podemos facilmente identificar e inseri-lo. Por último a data e hora da última refeição é representada como um *DATETIME* com o formato "YYYY-MM-DD HH:MM:SS", e pode variar entre "1000-01-01 00:00:00" até "9999-12-31 23:59:59". Este, eventualmente, será nulo, para abranger as situações em que o animal ainda não foi alimentado.

Ilustração 3 - Passagem da entidade alimento para o modelo lógico

Já no Alimento, temos o idAlimento que é também representado como um inteiro entre 1 e *MAXINT*. No caso do nome, será um *VARCHAR(20)*, pois foi o que considerámos mais adequado. O número de série do alimento seguirá a mesma lógica, pelo que será caracterizado também como *um VARCHAR(20*), pois este será uma mistura de letras e números. Por último, destacamos a quantidade de alimento existente que é representada como um inteiro entre 1 e *MAXINT*. O alimento possui todos os atributos como não nulos. Assim, é necessário terem um nome para não ocorrerem erros na alimentação dos animais bem como também é preciso saber a quantidade de alimento disponível para poder monitorizar a alimentação destes a qualquer altura. O número de série do alimento, proveniente da empresa irá estar sempre associado a este e por isso também não irá ser nulo.

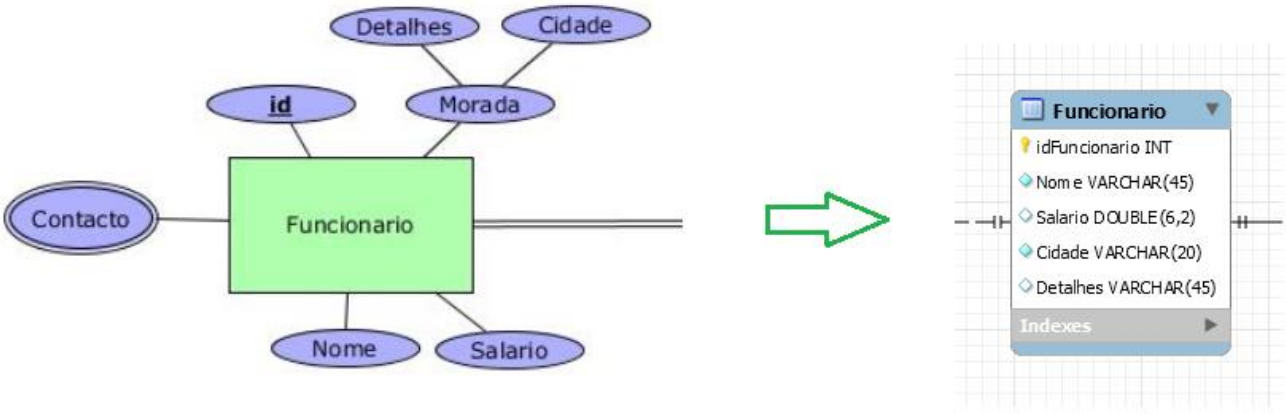


Ilustração 4-Passagem da entidade funcionário para o modelo lógico

Para o Funcionário, o idFuncionario é representado como um inteiro entre 1 e *MAXINT*, este não poderá ser nulo pelo mesmo motivo dos anteriores, ou seja, trata-se de uma chave primária. No caso do atributo contacto, considerámos que este deveria ser um *VARCHAR(20)*, pois chegamos à conclusão que seria uma boa média para qualquer contacto que possamos incluir. O nome do funcionário será representado como um *VARCHAR(45)*, dando algum intervalo para nomes mais longos, este não poderá ter valor nulo. O salário, é representado como um *DOUBLE(6,2)* de valor positivo, visto que falamos de preços, e necessitámos de uma parte decimal com 2 dígitos no máximo. Por fim a morada, sendo esta um atributo composto, será representada pela cidade (por se tratar de informação fundamental, tida em conta, para o bom funcionamento do Zoo, não poderá ter valor nulo), onde esta será um *VARCHAR(20),* e pelos diversos detalhes o qual decidimos considerar como *VARCHAR(45),* para assim termos a possibilidade de introduzir diversas informações sobre a morada. Quanto aos salários e a detalhes das moradas, existem situações em que estes valores poderão ser nulos, no caso de se tratar de um funcionário voluntário ou da família (como vimos na contextualização trata-se de um negócio de família) e também porque os detalhes não são necessários para a identificação da localização do funcionário, pois a partir dos dados da cidade o administrador já obtém as informações necessárias.

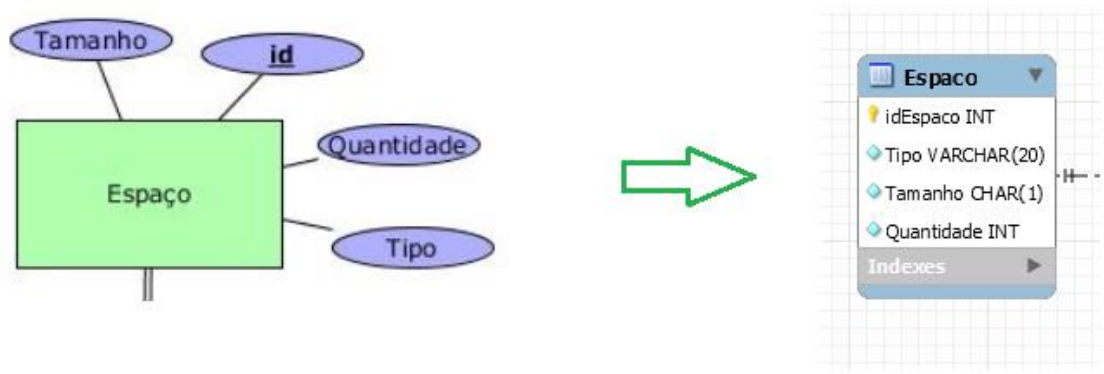


Ilustração 5 - Passagem da entidade espaço para o modelo lógico

Visto que o espaço é introduzido na base de dados com todo o seu conhecimento, nenhum dos seus atributos, Tipo, Tamanho e Quantidade, poderão ser nulos. Assim, no caso do Espaço, temos o idEspaco, que é representado por um inteiro entre 1 e *MAXINT*. O tamanho do espaço, que é identificado unicamente por um *CHAR(1)*, o qual pode ser 'P', 'M', 'G', que corresponde a "Pequeno", "Médio", "Grande", respetivamente e um tipo de espaço o qual é representado como um *VARCHAR(20)*, como por exemplo "Jaula" ou "Aquário".

Todas estas entidades possuem relacionamentos entre elas, que no modelo lógico, poderão dar origem a diferentes tabelas, especificadas de diferentes formas. Para além disso, existem alguns atributos com características especiais que são tratados de modo peculiar neste modelo.

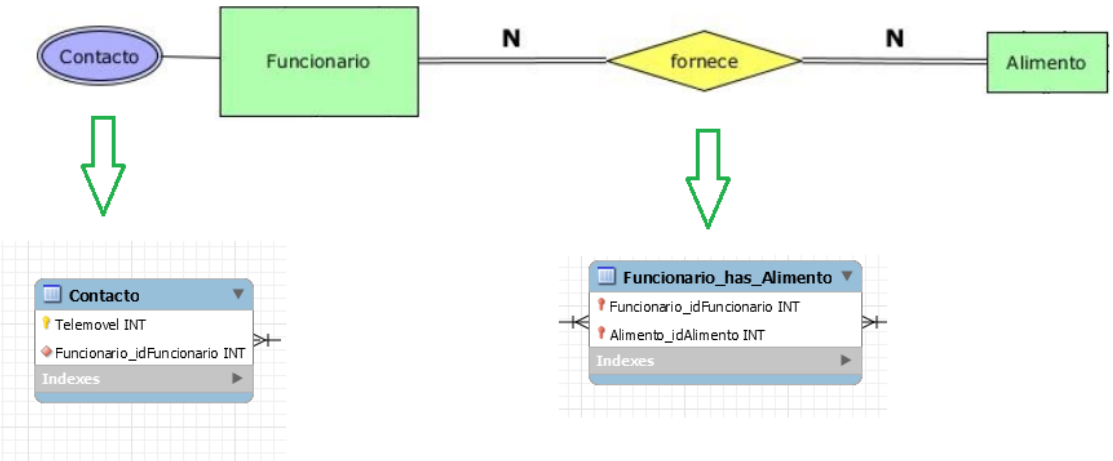


Ilustração 6 - Passagem da relação N para N e do atributo multivalor para o modelo lógico

O funcionário irá possuir duas ligações, uma para o contacto, por se tratar de um atributo multivalorado e outro para a tabela fruto da relação N para N entre o funcionário e o alimento. O contacto possuirá os vários números de telemóvel deste, logo podemos admitir que este atributo será a chave primária desta tabela, por isso não nulo e também terá de ter a chave estrangeira referente ao id do funcionário.

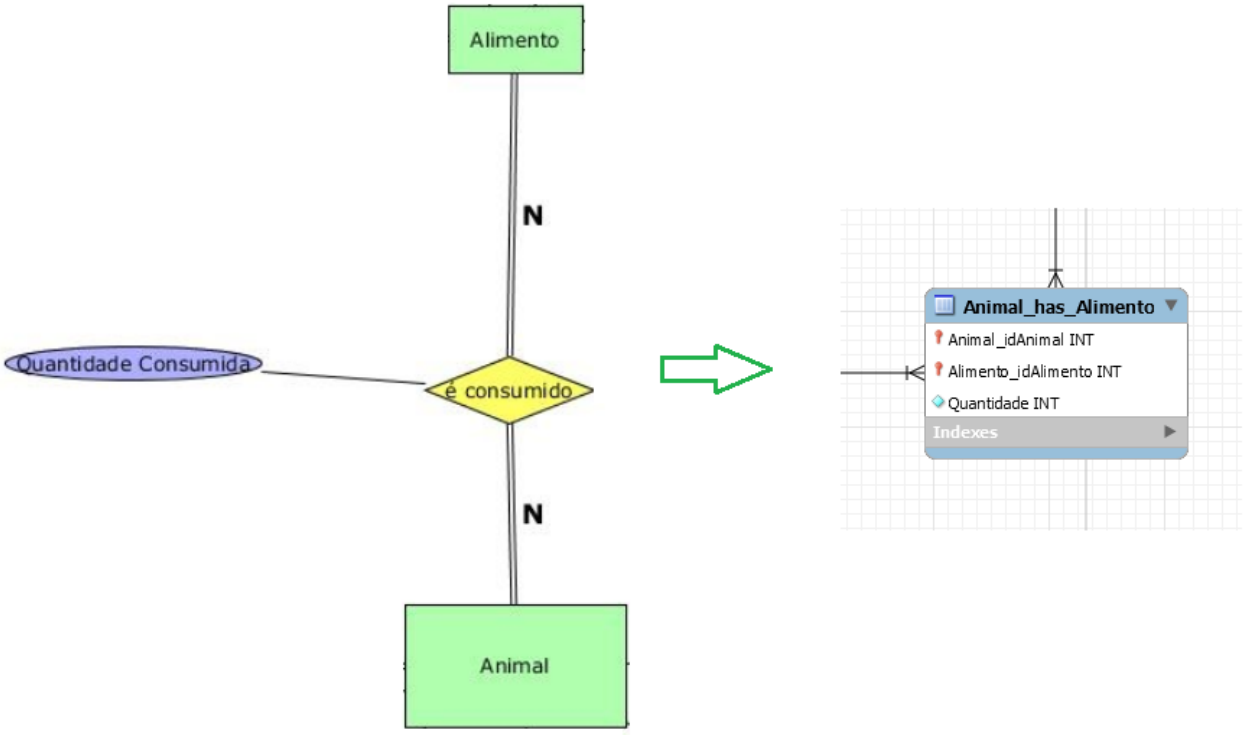


Ilustração 7 - Passagem da relação N para N e do atributo quantidade consumida para o modelo lógico

O alimento possuirá uma outra ligação N para N com Animal, logo iremos ter uma nova tabela que para além de possuir a chave primária composta pelas chaves primárias do alimento e do animal, também possui um atributo importante, a quantidade que como verificamos anteriormente, indica a quantidade ingerida pelo animal, logo terá de ser não nula.

4.2. Desenho do modelo lógico

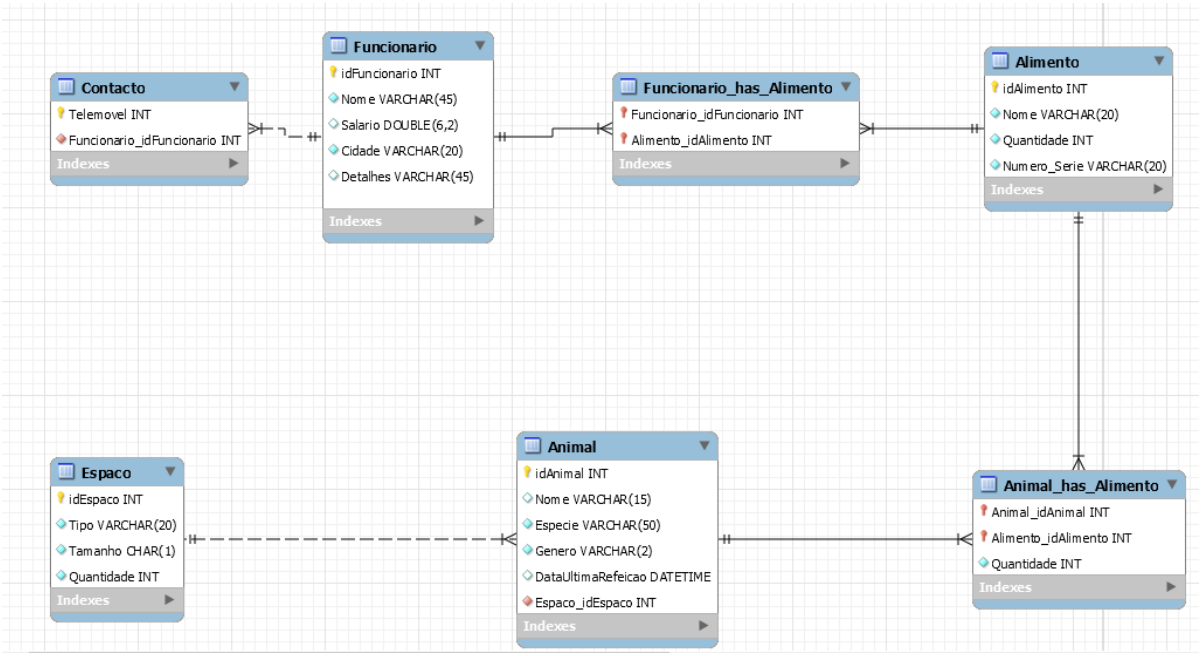


Ilustração 8 - Modelo Lógico

4.3. Validação do modelo através da normalização

- **Primeira Forma Normal (1FN):**

Pela análise do modelo desenvolvido chegamos à conclusão que existem vários casos comprometedores. No caso das relações, identificamos duas relações N para N, tendo isto em conta, irão ser criadas tabelas para estas relações. Ao possuírem os atributos que representam chaves estrangeiras, a identificação será segura e única, conseguindo assim manter o esquema normalizado.

No caso do atributo multivalorado, Contacto, a nossa resolução passa por atribuir a chave primária ao número de telemóvel e chave estrangeira referente ao Id_Funcionario. Desta forma, é possível que cada funcionário possua vários números de telemóvel sem por em causa a consistência da base de dados, continuando assim o sistema normalizado. Especifica-se de seguida, as propriedades acima enunciadas:

Funcionario_idFuncionario[1]	Alimento_idAlimento[1]
2	1
3	1
1	2
2	2

Tabela 4 - Prova de normalização da tabela do modelo lógico Funcionario_has_Alimento

Animal_idAnimal[1]	Alimento_idAlimento[1]	Quantidade[1]
2	1	4
3	5	26
6	3	21
9	5	13

Tabela 5 - Prova de normalização da tabela do modelo lógico Animal_has_Alimento

Telemovel[1]	Funcionario_idFuncionario[1]
911121121	1
911121122	1
912345678	2
912345679	2

Tabela 6 - Prova de normalização do atributo multivalorado Contacto

- **Segunda Forma Normal (2FN):**

De forma a provar que a nosso modelo vai de encontro à segunda e terceira forma normal, decidimos construir uma tabela onde se explicitam as dependências funcionais deste.

Espaco = {idEspaco, Tipo, Tamanho, Quantidade};
Animal = {idAnimal, Nome, Espécie, Género, HoraDaUltimaRefeicao, idEspaco};
Alimento = {idAlimento, Nome, Quantidade, NumeroDeSerie};
Funcionario = {idFuncionario, Nome, Cidade, Detalhes};
Funcionario_Alimento = {idFuncionario, idAlimento};
Animal_Alimento = {idAnimal, idAlimento, Quantidade}.
Contacto = {Telemovel, idFuncionario}

[1] Assumindo que os dados utilizados como exemplo foram inseridos na base de dados

Relacionamento	Dependência
Espaço	IdEspaco → Tipo, Tamanho, Quantidade
Animal	IdAnimal → Nome, Especie, Genero, HoraDaUltimaRefeicao, idEspaco
Alimento	IdAlimento → Nome, Quantidade, NuemeroDeSerie
Funcionario	IdFuncionario → Nome, Cidade, Detalhes
Funcionario_Alimento	IdFuncionario, idAlimento → IdFuncionario, IdAlimento
Animal_Alimento	IdAnimal, IdAlimento → Quantidade
Contacto	Telemovel→ idFuncionario

Tabela 7 - Tabela de dependências

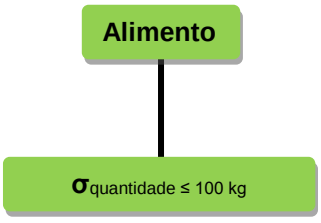
Tal como podemos verificar o esquema encontra-se de acordo com a segunda forma normal, já que se encontra na primeira e uma vez que todas as *Non-Primary-Keys* estão completamente dependentes a nível funcional da *Primary Key.*

- **Terceira Forma Normal (3FN):**

Como vimos anteriormente, o nosso esquema encontra-se na primeira e na segunda forma normal. Assim sendo estas condições necessárias para a prova da terceira forma normal basta apenas comprovar que nenhuma *non-primary-key* é transitivamente dependente de uma *primary key.* De facto, isto significa que uma chave não primária não pode determinar outra o que também se verifica no nosso modelo.

4.4. Validação do modelo com as interrogações do utilizador

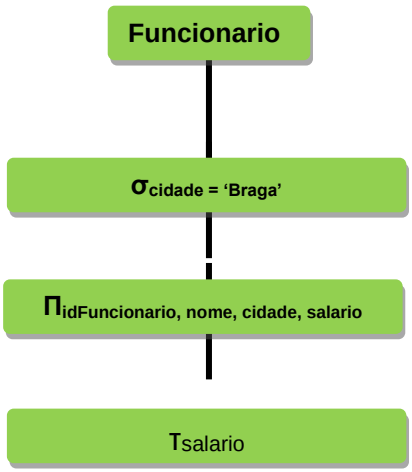
- **Interrogação 1:**



(Alimento)

$\sigma_{quantidade \leq 100\ kg}$

➢ **Interrogação 2:**



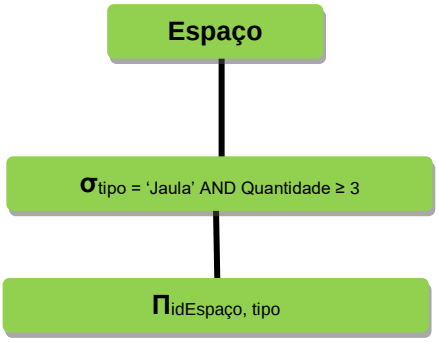
(Funcionário)

σcidade = 'Braga'

ΠidFuncionario, nome, cidade, salario

Tsalario

➢ **Interrogação 3:**

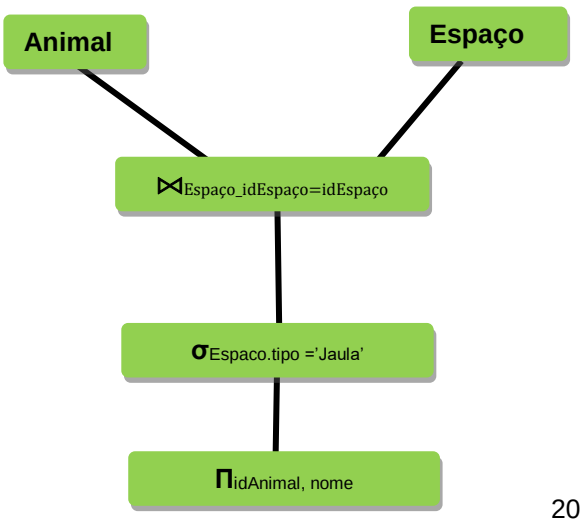


(Espaço)

σtipo = 'Jaula' AND Quantidade ≥ 3

ΠidEspaço, tipo

➢ **Interrogação 4:**

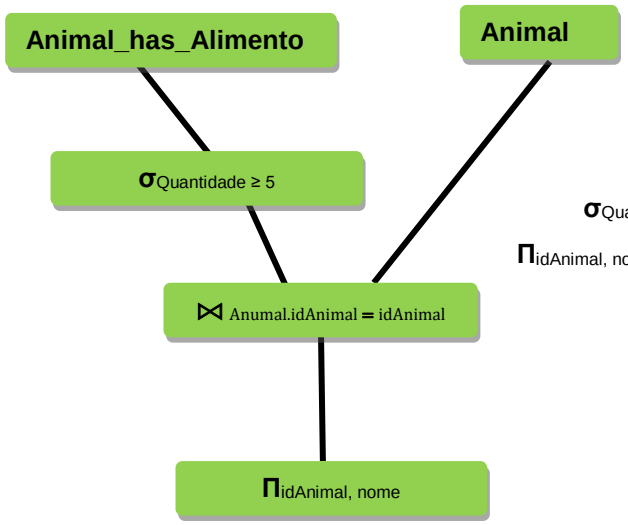


(Animal $\bowtie_{Espaço\_idEspaço=idEspaço}$ Espaço)

σEspaco.tipo ='Jaula'

ΠidAnimal, nome

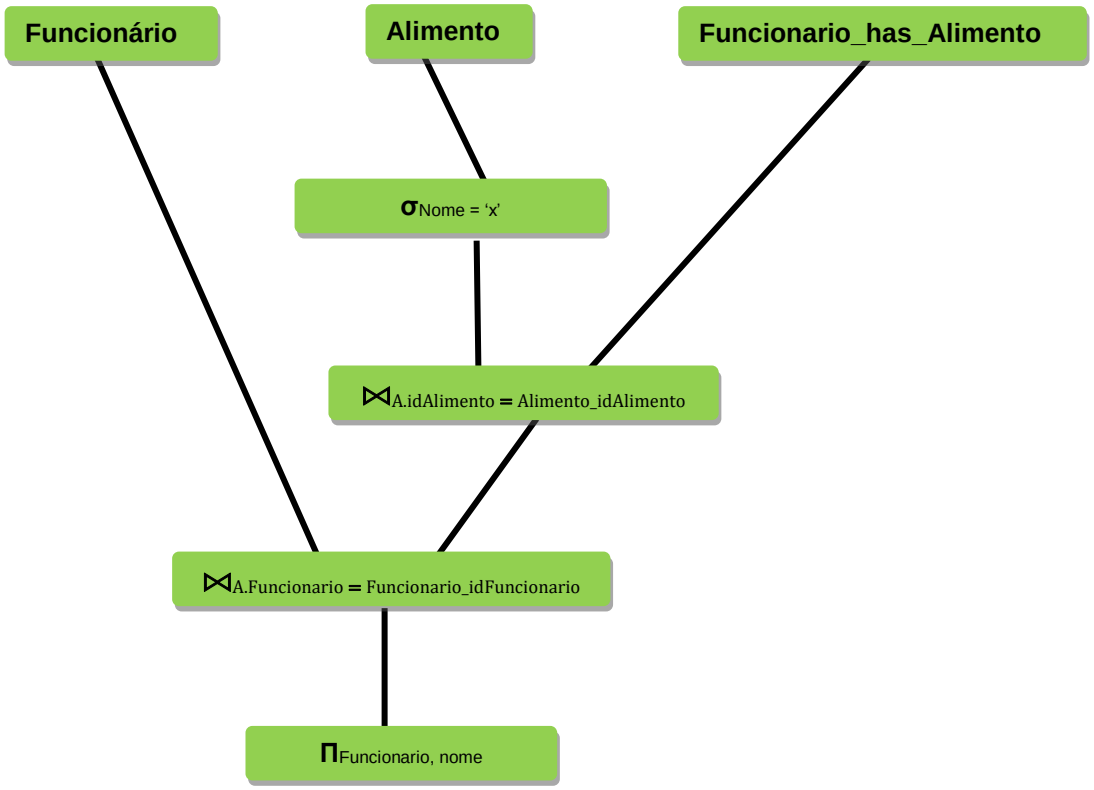
- **Interrogação 5:**



$\sigma_{Quantidade \geq 5}$ Animal_has_Alimento $(\bowtie_{Anumal.idAnimal = idAnimal})$ Animal

$\Pi_{idAnimal, nome}$

- **Interrogação 6:**



Funcionario $\bowtie_{A.\ Funcionario = Funcionario\_idFuncionario}$ ($\sigma_{Nome = 'x'}$ (Alimento) $\bowtie_{A.\ idAlimento = Alimento\_idAlimento}$ (Funcionario_has_Alimento))
$\Pi_{Funcionario, nome}$

4.5. Validação do modelo com as transações estabelecidas

De forma a tratar a situação em que o zoo recebe uma espécie bastante rara para adoção e tem de ignorar a quantidade máxima possível de um respetivo espaço, decidimos implementar uma transação. Assim, o mapa desta especifica-se de seguida.

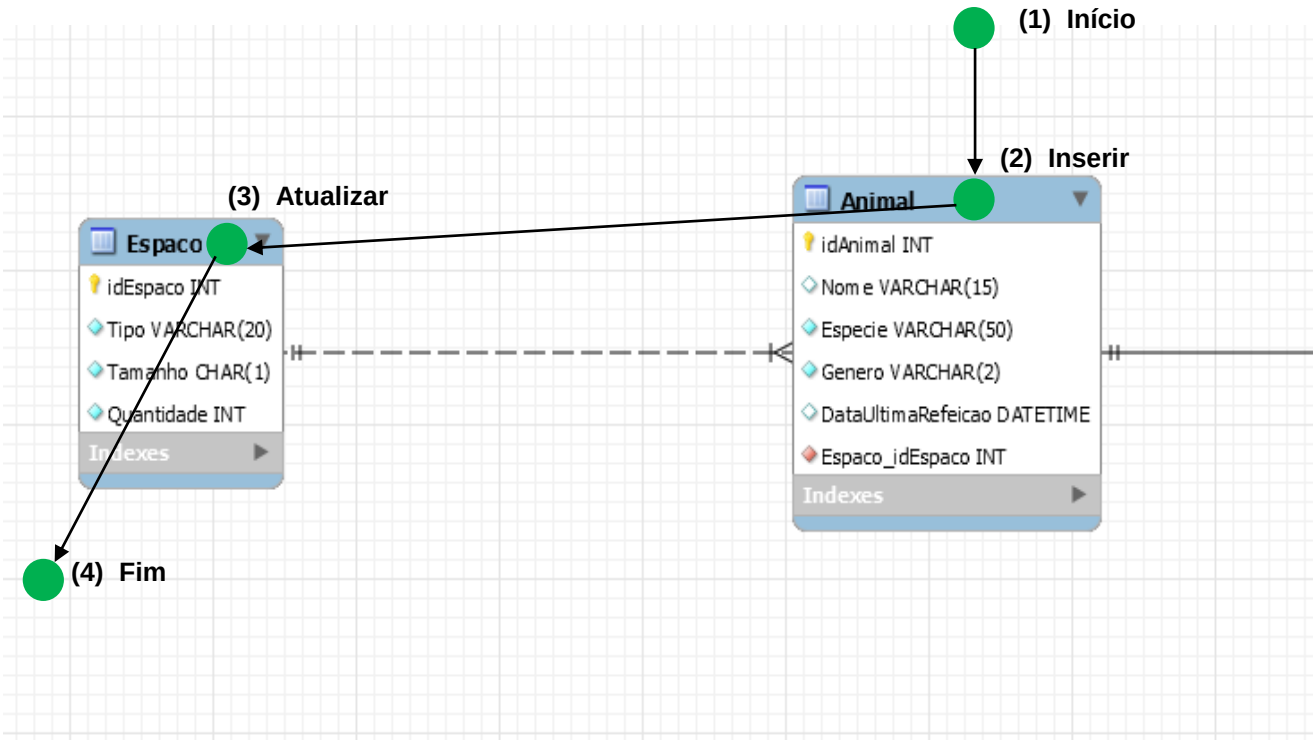


Ilustração 9 - Mapa de transação, Inserir espécie rara

4.6. Reavaliação do modelo lógico

Depois de concluído todo o processo que deu origem ao modelo lógico, verificamos se era necessário haver uma reavaliação deste. No entanto, tal não foi necessário pelo que não efetuamos nenhuma alteração no modelo lógico até então obtido.

5. Implementação Física

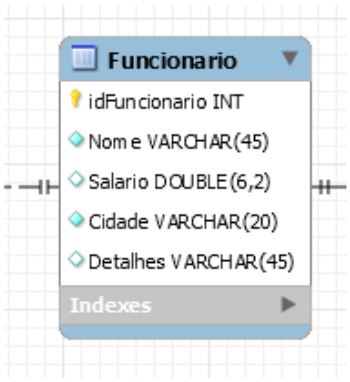
5.1. Seleção do sistema de gestão de base de dados

Para o sistema de gestão de bases de dados, resolvemos optar pelo MySQL. Esta decisão deveu-se ao facto de este se tratar de um software open source (apesar de ter sido adquirido pela Oracle) pelo que seria uma ferramenta bastante acessível. Outra das vantagens que o MySQL traria era o facto de existir o *MySQL Workbench*, o que nos permitia escrever as nossas instruções SQL tendo ao lado uma visualização do nosso modelo lógico, tornando mais fácil raciocinar sobre as interrogações que pretendíamos escrever. Por fim, sendo este apenas um trabalho académico e pelo que o MySQL está disponível para diferentes plataformas, facilitou na elaboração do nosso projeto, visto que diferentes membros do grupo usavam diferentes sistemas operativos (nomeadamente Linux e Windows).

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

De modo a construir o modelo físico, usamos a ferramenta de *Forward Engineering* disponibilizada pelo *MySQL*. Assim, o código necessário para esta transição foi gerado automaticamente. Explicitamos de seguida algum do código criado, o qual poderá ser consultado na integra em anexo.

- **Funcionario:**



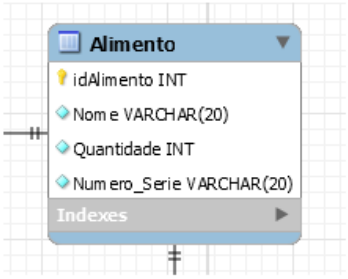

```
20  CREATE TABLE IF NOT EXISTS `ZooDB`.`Funcionario` (
21    `idFuncionario` INT NOT NULL,
22    `Nome` VARCHAR(45) NOT NULL,
23    `Salario` DOUBLE(6,2) NULL,
24    `Cidade` VARCHAR(20) NOT NULL,
25    `Detalhes` VARCHAR(45) NULL,
26    PRIMARY KEY (`idFuncionario`))
27  ENGINE = InnoDB;
28
```

Ilustração 10 - Tabela Funcionário no modelo físico


```
33  CREATE TABLE IF NOT EXISTS `ZooDB`.`Contacto` (
34    `Telemovel` INT NOT NULL,
35    `Funcionario_idFuncionario` INT NOT NULL,
36    PRIMARY KEY (`Telemovel`),
37    CONSTRAINT `fk_Contacto_Funcionario`
38      FOREIGN KEY (`Funcionario_idFuncionario`)
39      REFERENCES `ZooDB`.`Funcionario` (`idFuncionar
40      ON DELETE NO ACTION
41      ON UPDATE NO ACTION)
42  ENGINE = InnoDB;
43
```

Ilustração 11 - Tabela contacto no modelo físico

- **Alimento:**




```
49  CREATE TABLE IF NOT EXISTS `ZooDB`.`Alimento` (
50    `idAlimento` INT NOT NULL,
51    `Nome` VARCHAR(20) NOT NULL,
52    `Quantidade` INT NOT NULL,
53    `Numero_Serie` VARCHAR(20) NOT NULL,
54    PRIMARY KEY (`idAlimento`))
55  ENGINE = InnoDB;
```

Ilustração 12 - Tabela alimento no modelo físico

- **Funcionario_has_Alimento:**

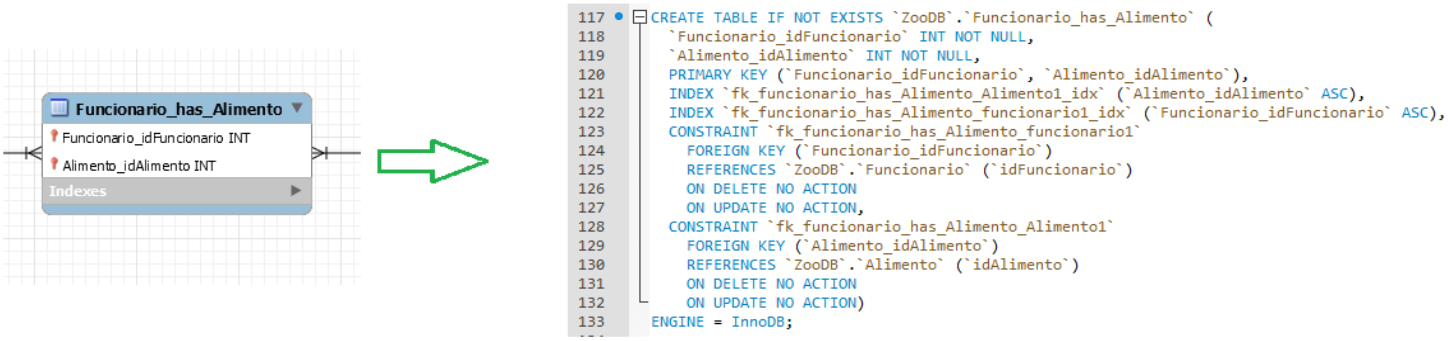


Ilustração 13 -Tabela Funcionario_has_Alimento no modelo físico

- **Espaço:**

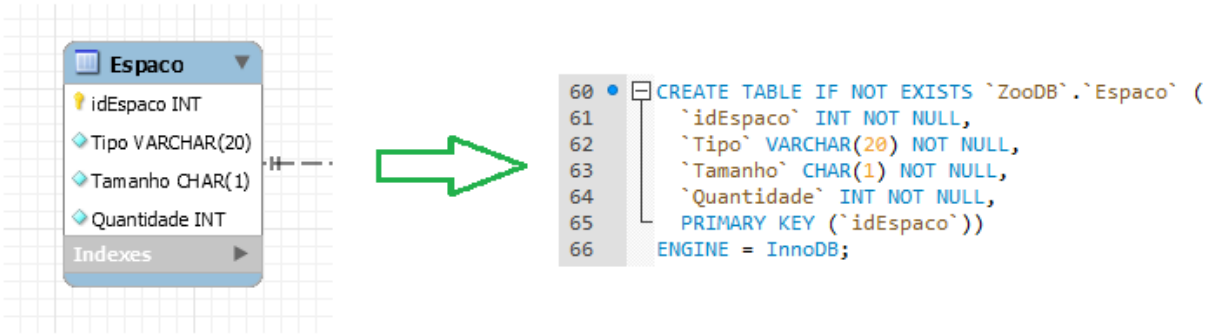


Ilustração 14 - Tabela Espaco no modelo físico

- **Animal:**

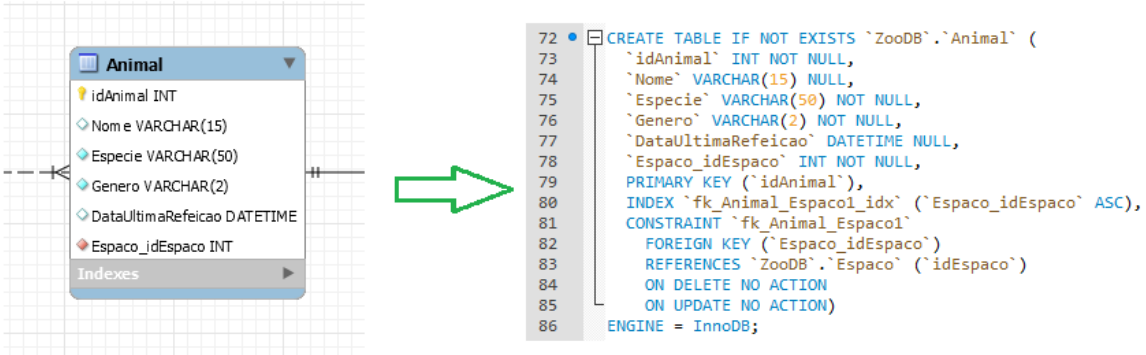
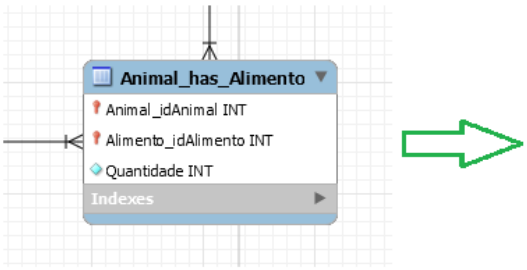


Ilustração 15 - Tabela Animal no modelo físico

➢ **Animal_has_Alimento:**




```
CREATE TABLE IF NOT EXISTS `ZooDB`.`Animal_has_Alimento` (
  `Animal_idAnimal` INT NOT NULL,
  `Alimento_idAlimento` INT NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`Animal_idAnimal`, `Alimento_idAlimento`),
  INDEX `fk_Animal_has_Alimento_Alimento1_idx` (`Alimento_idAlimento` ASC),
  INDEX `fk_Animal_has_Alimento_Animal1_idx` (`Animal_idAnimal` ASC),
  CONSTRAINT `fk_Animal_has_Alimento_Animal1`
    FOREIGN KEY (`Animal_idAnimal`)
    REFERENCES `ZooDB`.`Animal` (`idAnimal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_has_Alimento_Alimento1`
    FOREIGN KEY (`Alimento_idAlimento`)
    REFERENCES `ZooDB`.`Alimento` (`idAlimento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Ilustração 16 - Tabela Animal_has_Alimento no modelo físico.

5.3. Tradução das interrogações do utilizador para o SQL

Depois de povoada a base de dados (ver anexo), implementamos as interrogações necessárias em *MySQL*. Assim, no sentido a apresentar apenas algumas interrogações selecionamos aquelas que nos pareceram ser mais complexas de implementar.

➢ **Passagem do requisito de exploração quatro(RE4) para *MySQL*:**

Visto que pretendemos identificar os animais presentes num determinado espaço (iremos usar jaula para exemplificar) tivemos a necessidade de utilizar um *INNER JOIN.* Assim, este, quando aplicado corretamente (ligando o que há de comum entre a entidade espaço e a entidade animal – animal_Espaco = Espaco_idEspaco) irá recolher a informação pretendida.

```
SELECT idAnimal, nome
    FROM Animal
        INNER JOIN Espaco
            ON animal.Espaco_idEspaco = Espaco.idEspaco
                WHERE Espaco.tipo = 'Jaula';
```

Ilustração 17 - Interrogação *MySQL* (RE4)

- **Passagem do requisito de exploração seis(RE6) para *MySQL*:**

Quanto à *query* 6 pretendemos identificar os funcionários que forneceram um determinado alimento aos animais, sendo assim, precisaremos de utilizar dois INNER JOIN. Inicialmente com o *funcionario_has_alimento* e alimento utilizando *idAlimento* como atributo comum e depois entre *funcionario_has_alimento* e *funcionario* que terão o *idFuncionario* em comum. Filtrando pelo alimento desejado (usando “Frango” como exemplo) obteremos os nomes e os ids dos funcionários que forneceram estes alimentos, como desejado.

```
49 •    SELECT idFuncionario, funcionario.Nome
50          FROM Alimento
51              INNER JOIN Funcionario_has_alimento
52                  ON ALimento.idAlimento = Alimento_idAlimento
53                      INNER JOIN Funcionario
54                          ON Funcionario.idFuncionario = Funcionario_idFuncionario
55                              WHERE alimento.Nome = 'Frango';
```

Ilustração 18 - Interrogação *MySQL* (RE6)

Para além destas interrogações, desenvolvemos todos os mecanismos os necessários para que os requisitos de exploração fossem todos cumpridos. De facto, verificamos que algumas destas *querys* poderiam ser aliadas com outras funcionalidades que facilitarão o dia-a-dia do dono do Zoo. Assim, exemplificamos de seguida algumas delas:

- **Criação de um gatilho:**

Efetivamente, na formulação do modelo físico da base de dados, sentimos a necessidade da criação de um *trigger*, pois irá ser necessário atualizar a quantidade de comida disponível na tabela Alimento sempre que um animal for alimentado.

Com *atualiza_quantidade*, declaramos inicialmente duas variáveis inteiras, *Num* e *Id_Al* e definimos o seu valor para a quantidade ingerida pelo animal e o identificador desse mesmo alimento, respetivamente. Na tabela alimento, retiramos ao seu atributo quantidade (diferente do atributo quantidade presente em Animal_has_Alimento) e a quantidade consumida, presente em Num. Por fim, filtramos o alimento que foi comido através de *WHERE* idAlimento = id_Al.

```
60
61  DELIMITER $$
62  CREATE TRIGGER Atualiza_quantidade
63      AFTER INSERT ON Animal_has_Alimento
64      FOR EACH ROW
65      BEGIN
66          DECLARE Num INT;
67          DECLARE Id_Al INT;
68
69          SET Num = NEW.Quantidade;
70          SET Id_Al = NEW.Alimento_idAlimento;
71
72          UPDATE Alimento
73              SET Quantidade = Quantidade - Num
74                  WHERE idAlimento = Id_Al;
75      END
76  $$
```

Ilustração 19 - Criação de um *trigger* em *MySQL*

- **Criação de um procedimento:**

De forma a permitir possíveis aumentos de salários, provenientes do sucesso do jardim zoológico, fornecemos um *procedure* que o permite realizar.

Através de *atualizarSalario* podemos fornecer apenas o id do funcionário e o seu salário desejado e, caso se verifique o aumento de salário, efetuá-lo e caso contrário informar o utilizador que se trata de um salário inválido. Para isso começamos por declarar uma variável *double antigoSalario*, onde guardaremos o salário atual do funcionário utilizando o método INTO, após a identificação do funcionário correto. De forma a verificar se o salário inserido é válido, utilizamos um *if statement*, comparando-o com o *salarioAntigo* e realizamos um *update* ao salário do funcionário ou caso contrário, enviamos uma mensagem de "Salário invalido" através de *SIGNAl SQLSTATE '45000' SET MESSAGE_TEXT = 'Salario Invalido'.*

```
96
97  DELIMITER $$
98  CREATE PROCEDURE atualizarSalario(IN idFunc Int, IN salarioNovo DOUBLE(6,2))
99  BEGIN
100     DECLARE antigoSalario DOUBLE(6,2);
101
102     SELECT Salario INTO antigoSalario
103         FROM Funcionario
104             WHERE idFuncionario = idFunc;
105
106     IF (salarioNovo > antigoSalario) THEN
107         UPDATE Funcionario
108             SET Salario=salarioNovo
109                 WHERE idFuncionario = idFunc;
110     ELSE
111         SIGNAL SQLSTATE '45000'
112             SET MESSAGE_TEXT = 'Salario invalido';
113     END IF;
114 END;
115 $$
```

Ilustração 20 - Criação de um *procedure* no *MySQL*

5.4. Tradução das transações estabelecidas para SQL

A transação que se segue diz respeito, ao primeiro mapa de transação especificado mais acima neste relatório. Assim, explicamos de seguida os passos dados para a criação desta transação. Fornecemos, em primeiro lugar, o *PROCEDURE addEspecieRara* que recebe os dados do animal em questão. Realizámos assim dois processos fundamentais, a inserção do animal e a atualização dos dados do espaço. Nesta transação garantimos a sua atomicidade, através da *IF CLAUSE* no final do código, que permite revertê-la caso algum erro aconteça, ou validá-la caso contrário. A consistência também é garantida, pois mantemos um estado válido na base de dados independentemente do sucesso da transação. A isolação é respeitada pois o dado alterado (quantidade no espaço) apenas será visível após o *commit* e por sua vez toda a transação. Por fim, a durabilidade é garantida devido à sua gravação em memória não volátil.
Sendo assim garantimos a validade da transação visto que obedecem às propriedades ACID.

```
120  DELIMITER $$
121  CREATE PROCEDURE addEspecieRara(IN idAni Int, IN Nom VARCHAR(15),
122                                  IN Especi VARCHAR(50), IN Gen VARCHAR(2),
123                                  IN DUltimaRefeicao DATETIME, IN Espa_idEspa INT)
124  BEGIN
125      DECLARE Quanti INT;
126      DECLARE ErroTransaction BOOL DEFAULT 0;
127      DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET ErroTransaction = 1;
128
129      START TRANSACTION;
130
131      INSERT INTO Animal
132          (idAnimal, Nome, Especie, Genero, DataUltimaRefeicao, Espaco_idEspaco)
133          VALUES
134              (idAni, Nom, Especi, Gen, DUltimaRefeicao, Espa_idEspa);
135
136      UPDATE Espaco
137          SET Quantidade = Quantidade + 1
138              WHERE idEspaco = Espa_idEspa;
139
140      IF ErroTransaction THEN
141          ROLLBACK;
142      ELSE
143          COMMIT;
144      END IF;
145  END
```

Ilustração 21 - Implementação de uma transação em *MySQL*

5.5. Escolha, definição e caracterização de índices em SQL

Devido à dimensão da nossa base de dados consideramos que não é necessária a criação de índices. De facto, trata-se de um Zoo familiar pelo que, neste momento, a quantidade de dados não é muito elevada como veremos de seguida.

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

Tabela	Atributos	Tipo	Tamanho
Funcionario	idFuncionario	*INT*	4 Bytes
	Nome	*VARCHAR(45)*	45 Bytes
	Salario	*DOUBLE(6, 2)*	8 Bytes
	Cidade	*VARCHAR(20)*	20 Bytes
	Detalhes	*VARCHAR(45)*	45 Bytes

Tabela 8 - Tabela estimativa de tamanho para a entidade Funcionario

Deste modo, como existem inicialmente 6 funcionários, a tabela *Funcionario* irá ocupar 6 * 122 Bytes, o que perfaz 732 Bytes.

Tabela	Atributos	Tipo	Tamanho
Alimento	idAlimento	*INT*	4 Bytes
	Nome	*VARCHAR(20)*	20 Bytes
	Quantidade	*INT*	4 Bytes
	Numero_Serie	*VARCHAR(20)*	20 Bytes

Tabela 9 - Tabela estimativa de tamanho para a entidade Alimento

Deste modo, como existem inicialmente 5 tipos de alimento registados, a tabela *Alimento* irá ocupar 5 * 48 Bytes, o que perfaz 240 Bytes.

Tabela	Atributos	Tipo	Tamanho
Animal	idAnimal	*INT*	4 Bytes
	Nome	*VARCHAR(15)*	15 Bytes
	Especie	*VARCHAR(50)*	50 Bytes
	Genero	*VARCHAR(2)*	2 Bytes
	DataUltimaRefeicao	*DATETIME*	5 Bytes
	Espaco_idEspaco	*INT*	4 Bytes

Tabela 10 -Tabela estimativa de tamanho para a entidade Animal

Deste modo, como existem inicialmente 11 espécies de animais registados, a tabela *Animal* irá ocupar 11 * 80 Bytes, o que perfaz 880 Bytes.

Tabela	Atributos	Tipo	Tamanho
Espaco	idEspaco	*INT*	4 Bytes
	Tipo	*VARCHAR(20)*	20 Bytes
	Tamanho	*CHAR(1)*	1 byte
	Quantidade	*INT*	4 Bytes

Tabela 11 - Tabela estimativa de tamanho para a entidade Espaco

Deste modo, como existem inicialmente 3 tipos de espaços, a tabela *Espaco* irá ocupar 3 * 29 Bytes, o que perfaz 87 Bytes.

Tabela	Atributos	Tipo	Tamanho
Contacto	Telemovel	*INT*	4 Bytes
	Funcionario_idFuncionario	*INT*	4 Bytes

Tabela 12 - Tabela estimativa de tamanho para o Contacto

Deste modo, como existem inicialmente 7 contactos registados, a tabela *Contacto* irá ocupar 7 * 8 Bytes, o que perfaz 56 Bytes.

Tabela	Atributos	Tipo	Tamanho
Funcionario_has_Alimento	Funcionario_idFuncionario	*INT*	4 Bytes
	Alimento_idAlimento	*INT*	4 Bytes

Tabela 13 - Tabela estimativa de tamanho para Funcionario_has_Alimento

Deste modo, como existem inicialmente 8 entradas na tabela, correspondente a diversos funcionários que fornecem diferentes alimentos, a tabela *Funcionario_has_Alimento* irá ocupar 8 * 8 Bytes, o que perfaz 64 Bytes.

Tabela	Atributos	Tipo	Tamanho
Animal_has_Alimento	Animal_idAnimal	*INT*	4 Bytes
	Alimento_idAlimento	*INT*	4 Bytes
	Quantidade	*INT*	4 Bytes

Tabela 14 - Tabela estimativa de tamanho para Animal_has_Alimento

Deste modo, como existem 5 entradas na tabela, correspondente aos animais aos quais foram fornecidos um certo tipo de comida, a tabela *Animal_has_Alimento* irá ocupar 5 * 12 Bytes, o que perfaz 60 Bytes.

Logo, a base de dados necessita de 2119 Bytes ≈ 2.069 Kbytes.
Supondo um crescimento de 10% ao ano é possível concluir que a base de dados aumentará em aproximadamente 0.21Kbytes por ano.

5.7. Definição e caracterização dos mecanismos de segurança em SQL

Para irmos ao encontro dos requisitos de controlo, resolvemos criar 2 tipos de utilizadores. Um deles tem o nome de “admin”, e tem o acesso completo à base de dados. Este utilizador é suposto ser usado pelo dono do Zoo. O segundo, foi criado com intuito para qualquer funcionário, e tem o nome de “func” (como especificado no ponto a seguir). Este tem acesso a apenas algumas tabelas, e dentro de essas tabelas apenas pode executar certas instruções *MySQL*, nomeadamente o SELECT e o UPDATE.

```
4  CREATE USER 'admin'@'localhost'
5      IDENTIFIED BY 'admin';
6
7  GRANT ALL PRIVILEGES ON ZooDB.*
8      TO 'admin'@'localhost';
9
10 CREATE USER 'func'@'localhost'
11     IDENTIFIED BY 'func';
12
13
14 GRANT SELECT, UPDATE ON ZooDB.viewFunc_Animal TO
15     'func'@'localhost';
16
17
18 GRANT SELECT, UPDATE ON ZooDB.viewFunc_Alimento TO
19     'func'@'localhost';
20
21
22 GRANT SELECT ON ZooDB.viewFunc_Espaco TO
23     'func'@'localhost';
24
25
26 GRANT SELECT, UPDATE ON ZooDB.viewFunc_Animal_has_Alimento TO
27     'func'@'localhost';
```

Ilustração 22 - Criação de utilizadores no *MySQL*

5.8. Definição e caracterização das vistas de utilização em SQL

Como já referimos tivemos a necessidade de criar diferentes tipos de utilizadores, para tal, decidimos criar as vistas que achamos necessários para que estes vejam apenas os pontos de que têm acesso.

```
15 CREATE VIEW viewFunc_Animal AS
16     SELECT idAnimal, Nome, DataUltimaRefeicao
17         FROM Animal;
```

Ilustração 23 - Vista do funcionário em relação à tabela Animal

```
24 CREATE VIEW viewFunc_Alimento AS
25     SELECT idAlimento, Nome, Quantidade
26         FROM Alimento;
```

Ilustração 24 - Vista do funcionário em relação à tabela alimento

```
33 •  CREATE VIEW viewFunc_Espaco AS
34        SELECT Tipo, Tamanho, Quantidade
35            FROM Espaco;
```

Ilustração 25 - Vista do funcionário em relação á tabela espaço

```
42 •  CREATE VIEW viewFunc_Animal_has_Alimento AS
43        SELECT *
44            FROM Animal_has_Alimento;
```

Ilustração 26 - Vista do funcionário em relação à tabela Animal_has_Alimento

5.9. Revisão do sistema implementado com o utilizador

Quando apresentado o sistema final ao utilizador foi demonstrada a sua satisfação não só por implementar tudo aquilo que desejava, mas também por fornecer extras que certamente se iriam mostrar bastante úteis para o futuro do funcionamento do Zoo.

6. Conclusões e trabalho futuro

Ao longo do projeto deparamo-nos com diversos desafios que pretendemos caracterizar de forma exaustiva nesta secção.

Relativamente à primeira parte do trabalho sentimos dificuldades no que diz respeito à fixação dos requisitos. De facto, uma definição inicialmente instável destes, fez com que tivéssemos de retroceder algumas vezes no projeto. No que diz respeito à segunda parte, no modelo lógico, não sentimos grandes adversidades uma vez que a passagem do concetual para este fez-se à custa de certos procedimentos apreendidos nas aulas teórico-práticas. No entanto, o desafio instalou-se quando o problema a tratar eram as transações. Sendo um conceito pouco consolidado tivemos alguns problemas em enquadrar este mecanismo no nosso esquema. Efetivamente, consideramos que o número de transações presentes no trabalho poderia ser maior. Já no modelo físico os requisitos foram na sua generalidade cumpridos indo, deste modo, ao encontro da satisfação do problema inicialmente definido. Consideramos, contudo, que o trabalho tem os seus pontos fortes. Nesse sentido, a contextualização e a fundamentação foram definidas de maneira exímia permitindo maior estabilidade em alguns aspetos. Consideramos também que relativamente ao futuro o processo é viável pois apesar de se tratar de uma base de dados de pequena dimensão permite adicionar os dados necessários ao negocio para a qual foi construída.

Em suma, consideramos que um trabalho como este deveria ser tratado com mais antecedência, de modo a, prevenir eventuais erros. Assim, mesmo sendo um trabalho de complexidade reduzida foi essencial para um aprimoramento das nossas aptidões relativamente aos conteúdos da UC

Referências

- Thomas Connolly, TC, Carolyn Begg, CB, 2005, *Database System – A Pratictal Approach to Designm Implementation, and Managements*, 4ªedição, Harlow: Pearson Education Limited

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBDR	Sistema de Gestão de Bases de Dados Relacional
RD	Requisitos de Descrição
RE	Requisitos de Exploração
RC	Requisitos de Controlo
ID	Identidade
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
SQL	*Structured Query Language*

Anexos

I. Anexo 1 – Entrevista

1ª Pergunta: Porque é que recorreu aos nossos serviços?

R.: Pretendo um sistema que seja capaz de registar novos animais que receba no zoo, registar os alimentos que aqui temos, registar os funcionários "e os fornecedores existentes".

2ª Pergunta: O que precisa de saber de cada animal?

R.: Cada animal possui características e cuidados diferentes. É necessário ser possível saber o nome do animal, a idade que este possui, o género, os alimentos que come, a hora da última refeição bem como a espécie a que este pertence é também importante guardar o número do lugar onde este se encontra. Os espaços em que este se encontra também são importantes para mim, existem vários tipos deste e vários animais se encontram no mesmo espaço, com tamanhos diferentes.

3ª Pergunta: E sobre os alimentos?

R.: Como deve saber, cada animal possui uma alimentação diferente. É assim necessário, saber o nome do alimento, a quantidade existente no armazém e a data de validade.

Já agora, o funcionário deve ser registado com nome, número de funcionário, data de nascimento, contactos, morada e salário.

5ª Pergunta: Sendo o seu zoo bastante popular, pretende guardar algum tipo de informação referente aos seus clientes?

R.: Sinceramente, o motivo pelo qual recorri aos vossos serviços foi mesmo pelo simples facto de o meu falecido pai guardar toda a informação relevante referente aos nossos animais num caderno o que me dificulta bastante a concretização do meu trabalho. Os clientes não interferem em nada naquilo que pretendo, o meu único fim é poder através de um sistema eficaz minimizar os custos e facilitar o trabalho para as gerações seguintes.

6ª Pergunta: Para além de guardar todas as informações que o Sr. referiu, há algum tipo de funcionalidade extra que pretende que a base de dados faça a gestão?

R.: Sim, dava-me imenso jeito que a base de dados me permitisse consultar quais os alimentos que estão em falta. Geralmente quando tenho uma quantidade inferior a 100kg de alimentos este esgota-se muito rapidamente pelo que tenho de comprar mais. De facto, se fosse possível este tipo de consulta no sistema facilitava imenso o meu trabalho pois já não teria de me deslocar até ao armazém e verificar todos os alimentos um a um. Efetivamente, isto traz-me outra preocupação que é saber se os animais consumiram o que foi estipulado numa refeição.

Estes são alguns aspetos, outros serão relativamente aos meus funcionários. Era benéfico para mim determinar quais funcionários forneceram determinado alimento a determinado animal. Assim, desta forma, poderia apurar-se responsabilidades caso algum erro fosse cometido.

Sendo um Zoo com um caracter mais familiar, gosto de estar a par de todas as características dos meus funcionários. De facto, se a base de dados me permitisse consultar a informação referente a estes e agrupa-los por cidade organizando-os pelo salário auferido era bastante útil. Sabe, é importante para mim que os meus funcionários não tenham muitas despesas com o transporte, uma vez que não é possível facultarmos o transporte necessário, deste modo, uma pequena parte do salario é determinado tendo em conta este aspeto.

7ª Pergunta: E relativamente aos animais?

R.: Os animais são a parte fundamental deste negócio, é sempre importante determinar quais os animais num determinado tipo de espaço e verificar se o espaço contém mais de uma determinada quantidade. Temos sempre situações em que o Zoo recebe uma espécie rara e, independentemente, de a quantidade estar ultrapassada, esta tem de ser adicionada a um dos espaços existentes. Sendo assim, existem casos onde gostaria apenas de saber quais os animais presentes num espaço. Mais uma vez, na situação em que me encontro tenho de procurar tudo no caderno do meu falecido pai, o que não é fácil, pois devido a recentes mudanças todo o caderno se encontra um pouco confuso.

8ª Pergunta: E quem poderá ter acesso à base de dados?

R.: Apenas eu poderei aceder às informações dos funcionários e claro a tudo o que diz respeito ao Zoo. Os funcionários só podem aceder a aspetos relacionados com os animais, onde estes se encontram ou até mesmo à alimentação.

II. Anexo 2 – Modelo Físico no *MySQL*

```
-- 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='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema ZooDB
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema ZooDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `ZooDB` DEFAULT CHARACTER SET utf8 ;
USE `ZooDB` ;

-- -----------------------------------------------------
-- Table `ZooDB`.`Funcionario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ZooDB`.`Funcionario` (
  `idFuncionario` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Salario` DOUBLE(6,2) NULL,
  `Cidade` VARCHAR(20) NOT NULL,
  `Detalhes` VARCHAR(45) NULL,
  PRIMARY KEY (`idFuncionario`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `ZooDB`.`Contacto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ZooDB`.`Contacto` (
  `Telemovel` INT NOT NULL,
  `Funcionario_idFuncionario` INT NOT NULL,
  PRIMARY KEY (`Telemovel`),
  CONSTRAINT `fk_Contacto_Funcionario`
    FOREIGN KEY (`Funcionario_idFuncionario`)
    REFERENCES `ZooDB`.`Funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ZooDB`.`Alimento`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `ZooDB`.`Alimento` (
  `idAlimento` INT NOT NULL,
  `Nome` VARCHAR(20) NOT NULL,
  `Quantidade` INT NOT NULL,
  `Numero_Serie` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`idAlimento`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ZooDB`.`Espaco`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ZooDB`.`Espaco` (
  `idEspaco` INT NOT NULL,
  `Tipo` VARCHAR(20) NOT NULL,
  `Tamanho` VARCHAR(10) NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`idEspaco`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `ZooDB`.`Animal`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ZooDB`.`Animal` (
  `idAnimal` INT NOT NULL,
  `Nome` VARCHAR(15) NULL,
  `Especie` VARCHAR(50) NOT NULL,
  `Genero` VARCHAR(2) NOT NULL,
  `DataUltimaRefeicao` DATETIME NULL,
  `Espaco_idEspaco` INT NOT NULL,
  PRIMARY KEY (`idAnimal`, `Espaco_idEspaco`),
  INDEX `fk_Animal_Espaco1_idx` (`Espaco_idEspaco` ASC),
  CONSTRAINT `fk_Animal_Espaco1`
    FOREIGN KEY (`Espaco_idEspaco`)
    REFERENCES `ZooDB`.`Espaco` (`idEspaco`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ZooDB`.`Animal_has_Alimento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ZooDB`.`Animal_has_Alimento` (
  `Animal_idAnimal` INT NOT NULL,
  `Alimento_idAlimento` INT NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`Animal_idAnimal`, `Alimento_idAlimento`),
  INDEX `fk_Animal_has_Alimento_Alimento1_idx` (`Alimento_idAlimento` ASC),
  INDEX `fk_Animal_has_Alimento_Animal1_idx` (`Animal_idAnimal` ASC),
  CONSTRAINT `fk_Animal_has_Alimento_Animal1`
    FOREIGN KEY (`Animal_idAnimal`)
    REFERENCES `ZooDB`.`Animal` (`idAnimal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_has_Alimento_Alimento1`
    FOREIGN KEY (`Alimento_idAlimento`)
    REFERENCES `ZooDB`.`Alimento` (`idAlimento`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ZooDB`.`Funcionario_has_Alimento`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `ZooDB`.`Funcionario_has_Alimento` (
  `Funcionario_idFuncionario` INT NOT NULL,
  `Alimento_idAlimento` INT NOT NULL,
  PRIMARY KEY (`Funcionario_idFuncionario`, `Alimento_idAlimento`),
  INDEX `fk_funcionario_has_Alimento_Alimento1_idx` (`Alimento_idAlimento` ASC),
  INDEX `fk_funcionario_has_Alimento_funcionario1_idx` (`Funcionario_idFuncionario` ASC),
  CONSTRAINT `fk_funcionario_has_Alimento_funcionario1`
    FOREIGN KEY (`Funcionario_idFuncionario`)
    REFERENCES `ZooDB`.`Funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_funcionario_has_Alimento_Alimento1`
    FOREIGN KEY (`Alimento_idAlimento`)
    REFERENCES `ZooDB`.`Alimento` (`idAlimento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


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

III. Anexo 3 – Povoamento da base de dados

```
USE ZooDB;

INSERT INTO Espaco
    (idEspaco, Tipo, Tamanho, Quantidade)
        VALUES
    (1, 'Jaula', 'G', 9),
    (2, 'Aquario', 'P', 20),
    (3, 'Jaula', 'M', 4);


INSERT INTO Animal
        (idAnimal, Nome, Especie, Genero, DataUltimaRefeicao, Espaco_idEspaco)
        VALUES
    (1, 'Joao', 'Panthera leo', 'M', NULL, 1),
    (2, 'Oscar', 'Orcinus orca', 'M', '2017-11-24 22:36:12', 2),
    (3, 'Ricardo', 'Panthera pardus', 'M', '2017-11-25 11:21:53', 1),
    (4, 'Napoleão', 'Balaenoptera physalus', 'M', NULL, 2),
    (5, 'Paula', 'Equus asinus', 'F', NULL, 1),
    (6, 'Kika', 'Macropus rufus', 'F', '2017-11-23 16:45:21', 3),
    (7, 'Laila', 'Pan troglodytes', 'F', NULL, 1),
    (8, 'Maria', 'Loxodonta africana', 'F', '2017-11-25 07:43:54', 1),
    (9, 'Eddy', 'Ailuropoda melanoleuca', 'M', '2017-11-23 12:48:32', 2),
    (10, 'Lara', 'Carcharodon carcharias', 'F', NULL, 2),
    (11, 'Laila', 'Apis mellifera', 'F', NULL, 1);


INSERT INTO Alimento
        (idAlimento, Nome, Quantidade, Numero_Serie)
        VALUES
            (1, 'Frango', 10, '4312'),
            (2, 'Peixinhos', 230, '2042'),
            (3, 'Frutos', 589, '324'),
```

```
(4, 'Bamboo', 20, '1324'),
(5, 'Atum', 1000, '642');


INSERT INTO Funcionario
    (idfuncionario, Nome, Salario, Cidade, Detalhes)
    VALUES
(1, 'André Marques', 250, 'Braga', 'Tenões'),
(2, 'Mariana Fonte', 2000, 'Porto', 'Estádio do Dragao'),
(3, 'Filipe Sousa', 9500, 'Braga', 'Partes desconhecidas'),
(4, 'Nuno Gonçalves', 700, 'Lisboa', 'Praça do Rato'),
(5, 'Ricardo Martins', 1200, 'Vila Franca de Xira', NULL),
(6, 'Inês Ferreira', 9500, 'Guimarães', 'Partes desconhecidas');


INSERT INTO Animal_has_Alimento
    (Animal_idAnimal, Alimento_idAlimento, Quantidade)
    VALUES
(2, 1, 4),
(3, 5, 26),
(6, 3, 21),
(8, 4, 13),
(9, 5, 13);


INSERT INTO Funcionario_has_Alimento
    (Funcionario_idfuncionario, Alimento_idAlimento)
    VALUES
(1, 1),
(2, 1),
(3, 2),
(3, 3),
(4, 4),
(5, 5),
(5, 1),
(6, 5);
```

```
INSERT INTO Contacto
        (Telemovel, Funcionario_idfuncionario)
        VALUES
    (911121121, 1),
    (962345678, 2),
    (913323321, 3),
    (934442454, 4),
    (916231524, 1),
    (932136555, 5),
    (917646664, 6);
```