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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

ZMS (Zoological Management Service)

João Correia, Marco Pereira, Pedro António, Rúben Cerqueira

Dezembro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

ZMS (Zoological Management Service)

João Correia, Marco Pereira, Pedro António, Rúben Cerqueira

Dezembro, 2020

Resumo

Este relatório foi desenvolvido no âmbito do desenvolvimento de uma Base de Dados para um Jardim Zoológico local. Este foca a sua atividade na preservação das espécies e foi já condecorado várias vezes pela maneira como com escassos recursos tem vindo a destacar-se a nível nacional pelo esforço que tem feito na manutenção das melhores condições para os animais, assim como a maneira como tem passado a mensagem de sensibilização aos bons tratos animais.

O software visa facilitar e tornar mais eficiente a gestão dos mais diversos aspectos relacionados com o Jardim Zoológico, desde a venda dos diferentes bilhetes, até à documentação de cada animal, passando por todo o processo de vacinação que é conduzido no recinto.

No curso deste relatório serão apresentadas na íntegra todas as etapas do processo de desenvolvimento desta Base de Dados, desde a sua incepção, até à implementação física da mesma.

Primeiramente, foram investigados os requisitos propostos pelo Jardim Zoológico, bem como o meio onde este software será inserido. Averiguou-se seu método de trabalho e quais as suas motivações e objetivos para a implementação de uma base de dados, verificando se esta seria ou não economicamente viável.

Em seguida, através de métodos de análise e tendo em vista o que será esperado no presente, assim como ideias para expansão futura, foram levantados e aprovados os requisitos para o sistema que guiaram todo o nosso processo de desenvolvimento .

Finda esta fase, passou-se para a modelação conceptual, elaborada recorrendo à ferramenta “brModelo”, a partir da qual nos foi possível, seguindo as regras do mapeamento ER, fazer a transposição deste para um modelo lógico. Tal foi executado utilizando a ferramenta “MySQL Workbench”, que foi também a nossa escolha para a geração e implementação física do Base de Dados pretendida. A utilização de uma só ferramenta para o efeito garantiu que o resultado é o correto mediante as especificações estipuladas, e verifica-se assim como uma solução robusta e fiável para a utilização pretendida.

Estando o processo de desenvolvimento da Base de Dados terminado, conclui-se também o presente projecto.

Área de Aplicação: Desenho, Arquitetura, Desenvolvimento e Implementação de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Análise de Requisitos, Entidades, Atributos, Relacionamentos, Modelo Conceptual, Modelo Lógico, Modelo Físico, Normalização, Interrogações, Índices, Vistas de Utilização, *MySQL Workbench*, *SQL*.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização de aplicação do sistema

O planeta Terra configura-se como um sistema imensamente vasto que em si abriga uma quantidade enormemente diversa de organismos vivos. A forma como estes se relacionam entre si, inseridos no contexto de ecossistemas, revela a verdadeira complexidade e fragilidade da vida terrestre.

Devido à supramencionada diversidade de ecossistemas, configurar-se-ia como uma tarefa herculana satisfazer o desejo humano de interagir com os diversos animais, caso o humano tivesse de se deslocar a cada um destes ecossistemas.

É neste contexto que surgem os jardins zoológicos, com o intuito de permitir uma maior proximidade de interação entre a fauna terrestre e a grande porção da humanidade, cujos meios financeiros não permitem uma interação num contexto de habitat natural.

Quando geridos de forma responsável, estes podem-se revelar bastante benéficos para a conservação da vida animal terrestre. Primeiramente, como um meio de sensibilizar o público geral acerca das ameaças que esta enfrenta e, em segundo, pois poderão servir como santuários de vida selvagem onde esta é preservada e prolongada em comparação com as condições que os animais enfrentariam no meio selvagem.

Sendo assim, a gestão de um jardim zoológico revela-se como um desafio com uma certa complexidade inerente, devido à necessidade de balançar o aspeto negocial do serviço, com toda a logística associada à preservação da vida animal.

Fica assim, evidente, a necessidade de um serviço de base de dados que seja capaz de guardar, gerir e processar dados relativos a toda a esfera de um jardim zoológico, de forma a garantir o bom funcionamento do mesmo.

1.2. Fundamentação da Implementação da Base de Dados

O Jardim Zoológico começou a sua atividade servindo-se de outro sistema para armazenar os seus dados. Este era feito apenas com folhas de cálculo, e tinha obviamente bastante restrições associadas em termos de organização e manutenção.

A gerência tem como objetivo o acesso aos dados, com fácil manutenção, e que permita a sua manipulação com vista a poder responder às necessidades dos seus clientes e continuar a poder ter o crescimento que tem vindo a ter nos últimos anos.

Para responder a estas necessidade, a melhor alternativa é a implementação de uma Base de Dados que facilite o armazenamento de toda a informação.

A conceção da Base de Dados deve ser considerada por forma a cumprir os requisitos da infraestrutura de sistema de informação que o jardim zoológico já contém, para que a transição para o novo sistema de armazenamento e tratamento dos dados sejam o menos dispendiosos e atribulados o possível.

O presente sistema, permite aos funcionários da empresa armazenar a informação relativa aos visitantes, animais que tenham passado pelo zoológico e os recintos em que se encontram, que animais foram apadrinhados, o estado das suas vacinas, os veterinários que as administram, bem como todas as vendas de bilhetes.

O desejo do nosso Jardim Zoológico no futuro é de expandir a base de dados de vacinação dos animais para que se possa criar uma base de dados conjunta com outros jardins zoológicos. Esta medida levará a que seja possível a migração de animais para os jardins zoológicos em que tenham as melhores condições, com a garantia de que nenhuma informação será perdida na transação e que o animal terá o seu bem-estar assegurado.

1.3. Análise da Viabilidade do Processo

O projeto que temos em mãos assenta na implementação de uma Base de Dados Relacional que possibilite estudar e relacionar os dados que o Jardim Zoológico possui de modo a compreender e melhorar as condições do espaço quer para os visitantes, quer para os animais.

A utilização de uma Base de Dados Relacional é absolutamente fulcral pois permite a manipulação e a extração de dados sobre o nosso sistema de forma célere e com a complexidade que é exigida para as necessidades presentes.

Este investimento, embora considerável, vai significar ganhos largamente superiores no futuro, pois ter-se-á uma melhor noção das preferências e particularidades dos visitantes. Outra vantagem, e esta seguramente mais importante, é a de garantir que as condições de vida dos animais presentes no Jardim Zoológico são as melhores, através da inserção criteriosa na Base de Dados das vacinas tomadas por cada animal, quem a administrou, e qual será a próxima data de administração.

A abordagem atual, que utiliza um ficheiro para guardar toda esta informação, é algo que não é centralizado, e que, portanto, obriga a que cada inserção tenha que ser feita em vários locais. Tal não só não garante que a informação esteja disponível quando é necessária em cada situação, e também promove o erro, uma vez que não há qualquer sistema para garantir que a informação é inserida corretamente em todos os ficheiros.

Com uma Base de Dados implementada num Sistema de Gestão de Base de Dados Relacional, vai ser possível fazer todas as operações de gestão do Jardim Zoológico de forma simultânea, sem necessidade de reescrita, e com a segurança de que todos os intervenientes dispõem da mesma informação, tornando o sistema muito mais rápido e fiável.

Com a crescente afluência aos jardins zoológicos e preocupação com a vida animal, é cada vez mais importante assegurar a integridade da informação disponibilizada no sistema, bem como possuir mecanismos que nos permitam detetar aspectos em que as condições ou foco do Jardim Zoológico possam ser melhorados.

1.4. Estrutura do Relatório

O presente relatório será composto por seis capítulos:

Na **Introdução** ocorre a contextualização do projeto, seguida da apresentação do caso de estudo e menção das motivações e objetivos.

Ao longo do segundo capítulo, **Levantamento e Análise de Requisitos**, menciona-se a metodologia de levantamento e análise dos requisitos, seguida da apresentação dos requisitos de descrição, exploração e controlo, que serão analisados e validados.

No terceiro capítulo, **Modelação Conceptual**, são tratados os aspetos relativos à modelação das entidades que irão integrar o sistema. Estas serão identificadas e caracterizadas em termos de atributos, assim como acontecerá com os relacionamentos entre estas. Na fase seguinte, é apresentado o diagrama ER que caracterizará o sistema. Este modelo será validado em relação aos requisitos apresentados no capítulo anterior.

Durante o quarto capítulo, **Modelação Lógica**, o modelo conceptual é convertido na sua versão lógica que é, de seguida, validado através da normalização e de interrogações do utilizador.

No quinto capítulo, **Implementação Física,** o modelo lógico é convertido para uma versão física no sistema de gestão de base de dados relacionais escolhido.

Para finalizar na **Conclusão e Trabalho Futuro,** o nosso sexto capítulo, apresentam-se ilações resultantes da elaboração do trabalho e são propostas formas de aprimorar o projeto elaborado.

2. Levantamento e Análise de Requisitos

2.1. Método de Levantamento e de Análise de Requisitos Adotados

No domínio da metodologia de levantamento e análise de requisitos, o foco incidiu em dois aspetos principais de um jardim zoológico, mencionados no capítulo 1: o **aspeto negocial** e a **gestão da vida animal**.

Quanto ao **aspeto negocial,** o método principal de derivação dos requisitos centrou-se na simulação de quais seriam as métricas benéficas à análise das perspetivas financeiras do negócio, como, por exemplo, o número total de clientes num certo período de tempo.

Relativamente à **gestão da vida animal**, a metodologia centrou-se no levantamento de quais informações seriam relevantes para garantir a saúde e bem-estar dos animais. Esta assegura ainda a informação relativa à permanência dos animais no jardim zoológico, tendo em foco a organização espacial dos recintos onde estes se encontram, assim como relações de familiaridade cria/progenitor.

2.2. Requisitos Levantados

2.2.1. Requisitos de Descrição

1. Cada Bilhete é caracterizado pelo seu momento de aquisição.
2. Cada Bilhete é caracterizado por um Tipo;
3. Cada Tipo é descrito pelo seu nome e preço;
4. Cada Zona é caracterizada pelo seu nome e derivará o número de Animais nela presentes;
5. Cada Zona pode ser acedida por um conjunto de Tipos de Bilhete;
6. Cada Recinto é descrito pelas suas coordenadas, a sua área, o bioma que simula, assim como o número de Animais vivos que alberga;
7. Cada Recinto pertence a uma Zona;
8. Cada Animal é caracterizado pelas suas medidas, género, indicador se ainda se encontra vivo e a sua data de nascimento. Poderá adicionalmente ter um nome atribuído caso tenha sido adotado;
9. Cada animal relacionar-se-á com as suas crias, caso as tenha;
10. Cada animal encontra-se abrigado num recinto;
11. Existe uma espécie relativa a cada animal;

12. A espécie é caracterizada pelo seu nome comum, nome científico e qual é o valor monetário mensal necessário para apadrinhar um animal pertencente à espécie;
13. Um padrinho será alguém que faz contribuições monetárias mensais a um ou mais animais;
14. Cada padrinho será caracterizado pelo seu nome, uma lista de contactos, o seu NIB, cartão de cidadão e NIF;
15. Cada vacina é descrita pela doença a que é relativa;
16. Uma vacina é obrigatoriamente administrada numa ou mais espécies. Este relacionamento é descrito pela data limite em que se tem de tomar a primeira dose, quantas doses são necessárias e a dosagem por cada vacina (Ex: 10 ml). Caso o número de doses necessárias seja superior a 1, será indicado um intervalo temporal para a tomada de cada dose subsequente;
17. Cada vacina é administrada a um ou mais animais, por um ou mais veterinários;
18. Cada veterinário é descrito pelo seu nome e lista de contactos.

2.2.2 Requisitos de Exploração

1. É possível consultar que animais um padrinho apadrinhou;
2. Podemos calcular qual o valor mensal total que um padrinho deve pagar;
3. A base de dados permite saber as crias de um animal existentes no zoo;
4. Pode-se determinar todos os descendentes de um animal no zoo;
5. É possível consultar os progenitores de um animal existentes no zoo;
6. Disponibilidade para mostrar todos os ascendentes de um animal existentes no zoo;
7. Existe a possibilidade de conhecer todas as espécies que um tipo de bilhete dá acesso;
8. Podemos consultar quantos animais, existem em cada bioma do jardim zoológico;
9. É possível calcular o top 3 dos tipos de bilhetes mais comprados;
10. Permite conhecer quantos bilhetes de cada tipo foram vendidos;
11. Disponibilizar a faturação total do jardim zoológico;
12. Calcular a faturação do zoo num intervalo de tempo;
13. Podemos saber o crescimento de visitas anual;
14. É possível consultar que veterinários administraram qual vacina a um animal;
15. Permite conhecer quais vacinas já foram administradas a um animal e o seu momento de administração;
16. É possível consultar quais vacinas, incluindo doses que se devem repetir, um animal ainda deve tomar;
17. Podemos apresentar os contactos de um veterinário.
18. Calcular a média de pesos de animais de uma dada espécie.

2.2.3 Requisitos de Controlo

1. O jardim zoológico encontra-se aberto ao público entre as 9h e as 17h;
2. Não se pode eliminar informação relativa à vacinação;
3. Quando se regista um novo apadrinhamento, deve-se adicionar um nome ao animal apadrinhado;
4. Um empregado poderá registar novos clientes e bilhetes;
5. O registo de um novo apadrinhamento pode ser adicionado por um empregado, alterando o nome de um animal;
6. Um gestor tem controlo sobre todos os dados.

2.3. Análise de Requisitos

O processo de levantamento e análise de requisitos configura-se como um elemento nuclear no desenho e posterior implementação de um sistema de gestão de bases de dados, devido ao facto das posteriores modelações terem como base os critérios definidos nesta fase.

Quanto ao processo de levantamento, este foi dividido em três fases, que passamos a enumerar: **requisitos de descrição**, **requisitos de exploração** e **requisitos de controlo**.

Na fase de levantamento dos **requisitos de descrição**, o foco consistiu em analisar quais os elementos que consideramos fulcrais ao funcionamento de um jardim zoológico e qual a forma como estes se deviam relacionar entre si, dando assim origem aos requisitos apresentados.

Seguidamente, para a elaboração dos **requisitos de exploração**, analisou-se qual informação seria relevante extrair acerca do nosso jardim zoológico, tendo em conta os elementos e relacionamentos apresentados na fase anterior.

Na fase final, em que se procedeu ao levantamento dos **requisitos de controlo**, decidiu-se quais seriam as restrições de administração da nossa base de dados, de forma a criar coerência entre todo o sistema.

Este processo foi acompanhado da análise dos requisitos levantados, implicando uma constante discussão acerca de quanto cada um dos requisitos realmente se adapta a um cenário real, tendo em foco evitar cenários pouco adequados à realidade da gestão do serviço proposto.

Desta forma, os requisitos apresentados têm como intuito principal adicionar valor real ao controlo da logística animal e negocial do tema apresentado.

3. Modelação Conceptual

3.1. Apresentação da Abordagem de Modelação Realizada

Tendo em conta os requisitos levantados e validados no capítulo anterior, deve-se iniciar a conceção de um plano de design da Base de Dados que será implementada.

A primeira fase deste design será a criação de um **Modelo Conceptual**, um modelo da Base de Dados independente do Sistema de Gestão a utilizar. O intuito deste modelo será representar as várias entidades que irão compor o nosso sistema, assim como os atributos relativos a cada uma destas e os relacionamentos entre as diversas entidades.

Para tal, será usado um **Diagrama Entity-Relationship.** A estratégia de criação deste diagrama será incremental: em primeiro lugar serão representadas as entidades que foram identificadas como necessárias para modelar o sistema. De seguida representaremos os atributos de cada uma destas entidades e procederemos à identificação das relações, tomando especial atenção a adição de atributos às entidades, no caso de ser uma relação N para N. Concluiremos o processo com a representação das multiplicidades de cada relação modelada.

3.2. Identificação e Caracterização das Entidades

Atendendo aos requisitos levantados foi identificada a necessidade de criação das seguintes entidades:

- **Bilhete**

A entidade Bilhete representa cada passe de acesso vendido no nosso jardim zoológico. Esta será caracterizada pelo seu **id** e **momento de aquisição.**

- **Tipo**

O Tipo representa os vários tipos de Bilhete que o zoo suportará. Cada Tipo é caracterizado por um **nome** e um **preço**.

- **Zona**

A entidade Zona representa as diversas zonas em que os recintos de animais poderão estar inseridos. Estas são caracterizadas pelo seu **nome**. Derivam a quantidade de animais existentes dentro dos seus recintos.

- **Recinto**

A entidade Recinto representa os espaços físicos que poderão abrigar animais. Estes são representados pelas suas **coordenadas**, **área** e **bioma.**

- **Animal**

Esta entidade é caracterizada pelas suas **medidas**, **género**, uma marca a indicar se o animal se encontra **vivo** e a sua **data de nascimento**. Opcionalmente, pode ainda ter um **nome**.

- **Padrinho**

Um Padrinho será alguém que adota um ou mais animais, pagando um valor mensal associado por cada animal apadrinhado. Este é identificado por um **nome**, uma **lista de contactos**, **NIB**, **morada**, **cartão de cidadão** e **NIF.**

- **Espécie**

Esta entidade representa cada espécie de animal presente no nosso jardim zoológico. Ela é caracterizada pelo **nome comum** da espécie em questão, o seu **nome científico** e o **valor de apadrinhamento.** Este último traduz-se na quantidade monetária representativa das despesas associadas ao tratamento de um animal desta espécie e que será pago por um padrinho.

- **Vacina**

Entidade que representa vacinas que os animais deverão ter de tomar. É identificada pela **doença** que procura inocular.

- **Veterinário**

Um veterinário é responsável por administrar vacinas a animais. É caracterizado pelo seu **nome** e uma **lista de contactos**.

3.3. Identificação e Caracterização dos Relacionamentos

- **Relacionamento Tipo–Bilhete**

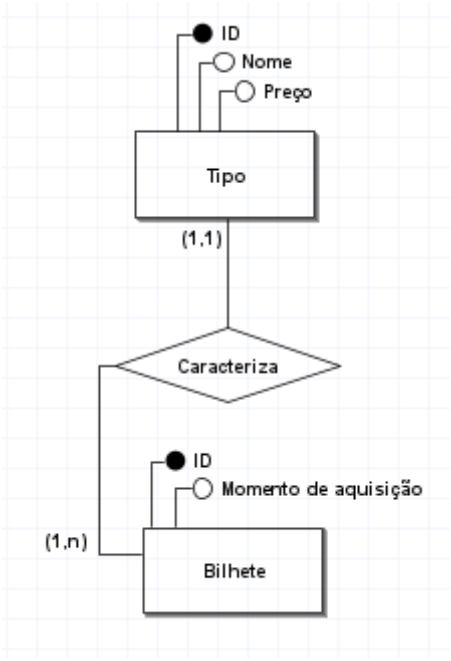

Figura 1 - Relacionamento Tipo-Bilhete.

Relacionamento: Tipo Caracteriza Bilhete

Descrição: Cada Bilhete é caracterizado por um Tipo. Este relacionamento permitirá delimitar a quais zonas do jardim zoológico um tipo de bilhete dará acesso.

Cardinalidade: Tipo (1) - Bilhete (1..*).

Cada Bilhete tem apenas um Tipo, porém todos os Bilhetes de acesso terão um dos Tipos pré-determinados.

Atributos: Este relacionamento não possui atributos devido à sua cardinalidade.

- **Relacionamento Zona-Tipo**

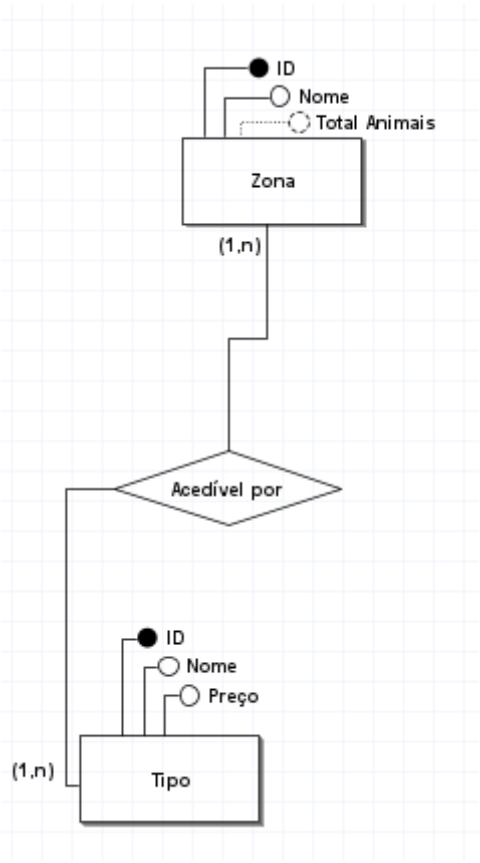

Figura 2 - Relacionamento Zona-Tipo.

Relacionamento: Cada Zona é acedida com um Tipo de Bilhete.

Descrição: Cada Tipo de Bilhete dá acesso a zonas pré-determinadas do zoo.

Cardinalidade: Tipo (1..*) – Zona (1..*).

Cada Tipo de Bilhete poderá limitar o Visitante a aceder certas Zonas do jardim zoológico, consoante o seu tema. Como exemplo, um Bilhete geral poderá dar acesso a todas as Zonas, mas um Bilhete do Tipo “Zona dos Répteis” poderá dar apenas acesso às Zonas associadas a répteis.

Assim sendo e, visto que várias Zonas poderão ser acedidas por diferentes Tipos de Bilhetes, esta relação configura-se como N para N.

Atributos: Este relacionamento não possui atributos.

- **Relacionamento Recinto-Zona**

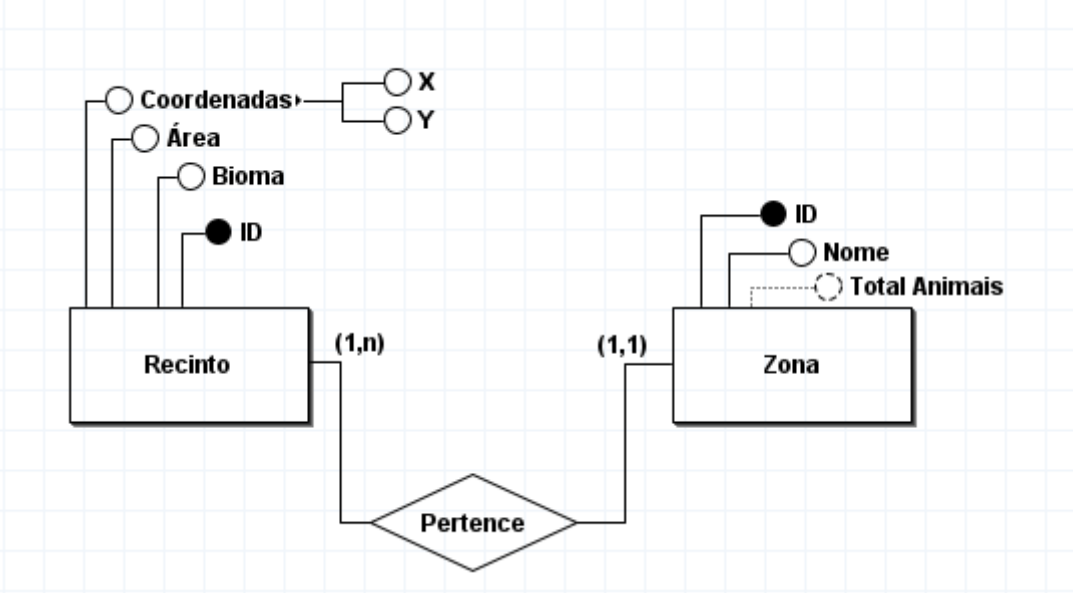

Figura 3 - Relacionamento Recinto-Zona.

Relacionamento: Recinto Pertence a Zona

Descrição: Cada Recinto deve pertencer a uma zona.

Cardinalidade: Recinto (1..*) - Zona (1).

O jardim zoológico estará dividido em Zonas animais. Cada uma comportará um conjunto de Recintos.

Atributos: Este relacionamento não possui atributos devido à sua cardinalidade.

- **Relacionamento Animal-Recinto**

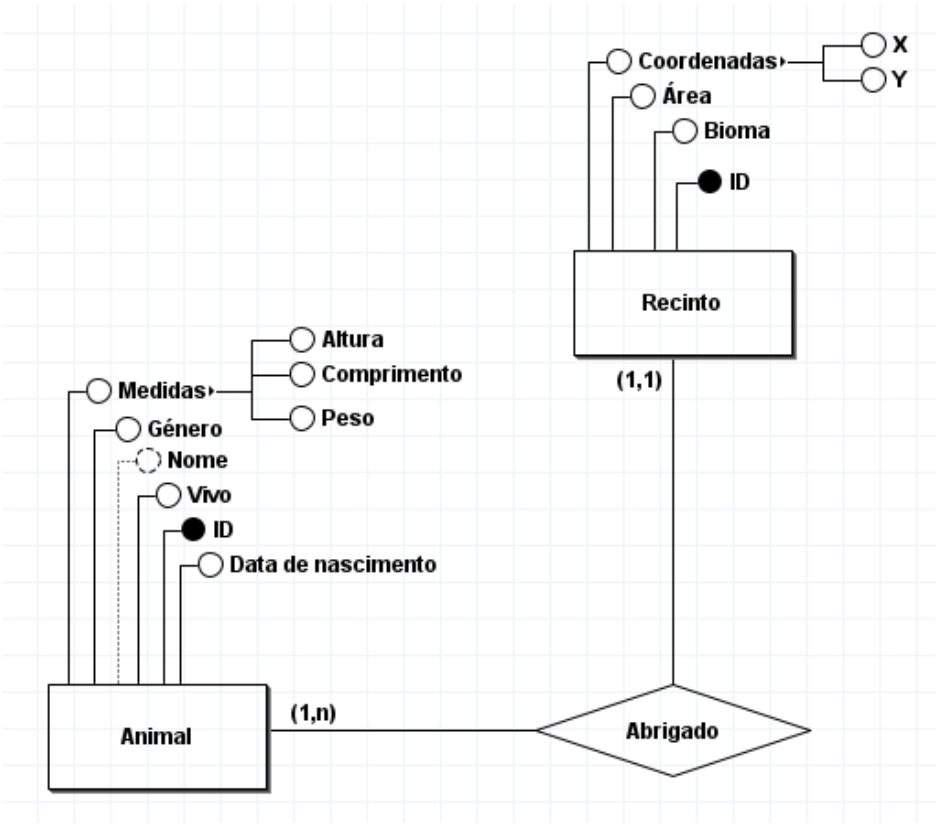

Figura 4 - Relacionamento Animal-Recinto.

Relacionamento: Animal é abrigado em Recinto

Descrição: Vários animais podem ser abrigados no mesmo recinto.

Cardinalidade: Animal (1..*) - Recinto (1).

Vários animais estarão abrigados em cada um dos recintos.

Atributos: Este relacionamento não possui atributos devido à sua cardinalidade.

- **Relacionamento Padrinho-Animal**

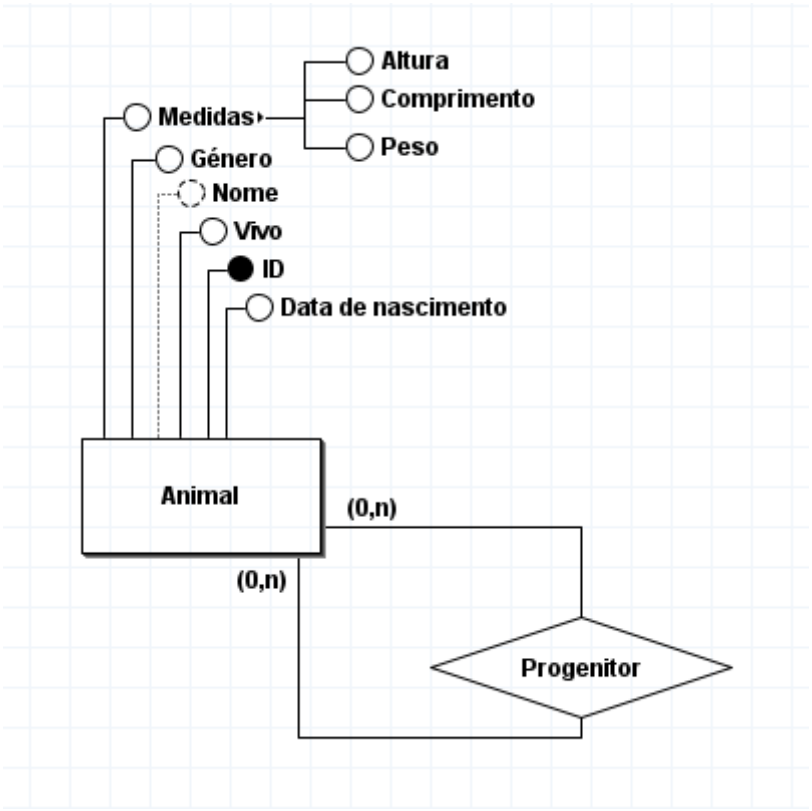

Figura 5 - Relacionamento Animal-Animal

Relacionamento: Um animal pode ser progenitor de vários animais.

Descrição: Durante a vida dos animais no jardim zoológico, estes poderão dar origem a crias.

Cardinalidade: Animal (0..*) – Animal (0..*).

Cada Animal poderá não ter crias ou poderá ter um conjunto delas. Observando no sentido inverso, um Animal pode ou não ter progenitores no jardim zoológico.

Atributos: Este relacionamento não possui atributos.

- **Relacionamento Animal-Padrinho**

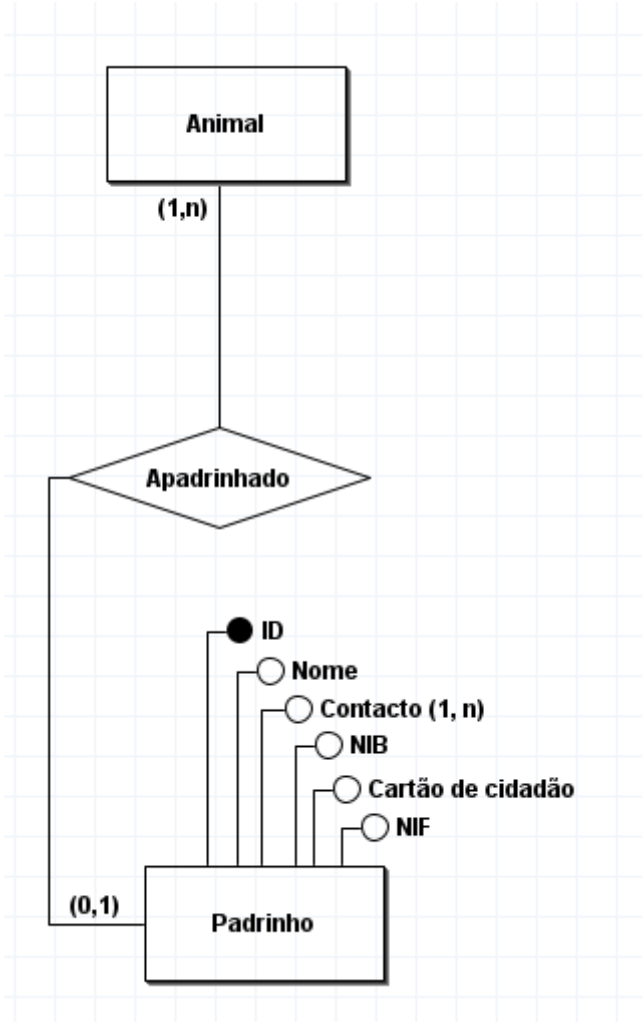

Figura 6 - Relacionamento Padrinho-Animal.

Relacionamento: Um animal é apadrinhado por um animal.

Descrição: Uma pessoa pode optar por pagar uma quantia mensal para suportar os custos associados a um ou mais Animais (custos derivados da Espécie do Animal).

Cardinalidade: Animal (1..*) - Padrinho (0,1).

Cada Animal poderá ter apenas um Padrinho, mas o mesmo padrinho pode suportar mais do que um Animal.

Atributos: Este relacionamento não possui atributos devido à sua cardinalidade.

- **Relacionamento Espécie-Animal**

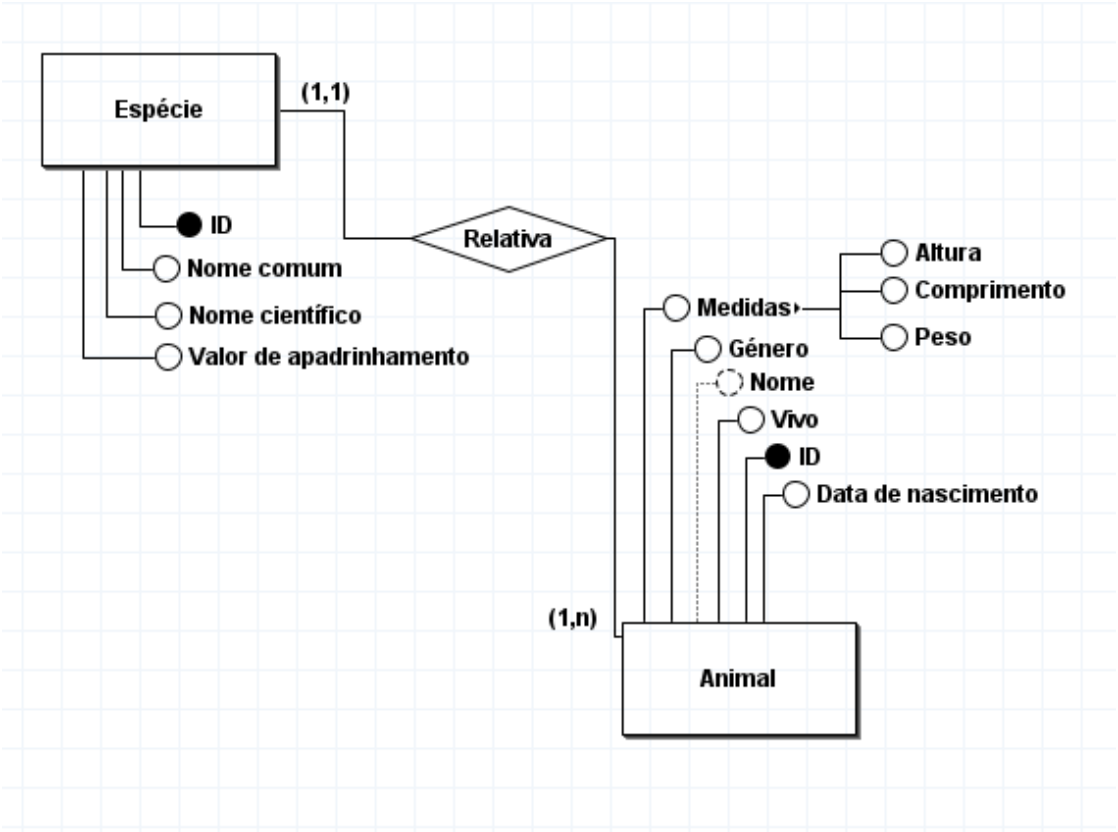

Figura 7 - Relacionamento Espécie-Animal.

Relacionamento: Espécie Caracteriza Animal.

Descrição: Cada Animal deve pertencer uma Espécie, de onde irá derivar alguns atributos.

Cardinalidade: Espécie (1) – Animal (1..*).

Cada Animal tem apenas uma Espécie. Pode, porém, podem existir vários animais da mesma espécie no jardim zoológico.

Atributos: Este relacionamento não possui atributos devido à sua cardinalidade.

- **Relacionamento Vacina-Espécie**

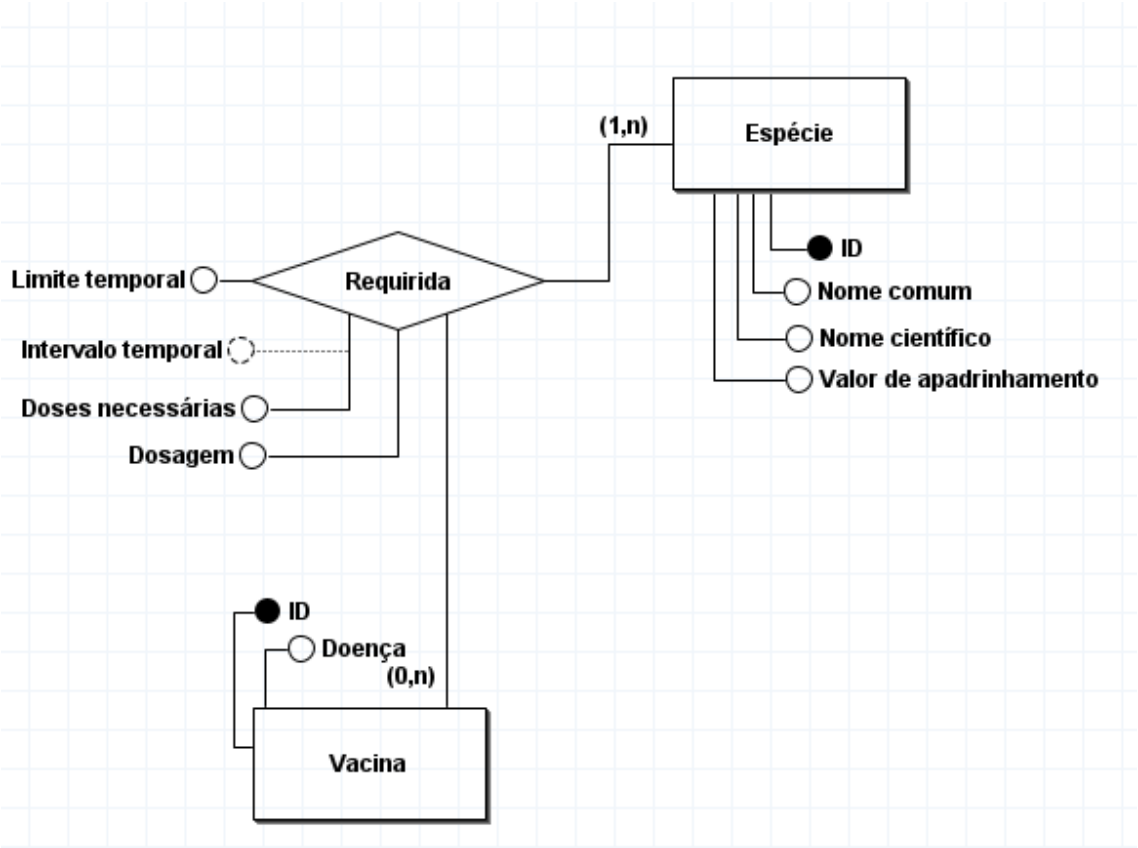

Figura 8 - Relacionamento Vacina-Espécie.

Relacionamento: Vacina requerida por Espécie.

Descrição: Para garantir a saúde dos Animais há um conjunto de Vacinas que estes devem tomar. Estas estarão associadas à Espécie do animal.

Cardinalidade: Vacina (0..*) – Espécie(1..*).

Cada Vacina pode ser dada a um conjunto de Espécies e cada Espécie pode ter ou não de tomar uma ou mais vacinas.

Atributos: O **limite temporal** indica a idade (em meses) máxima a que um animal da espécie deve tomar a primeira dose da vacina. A **dosagem** indica, em ml, a quantidade de vacina a tomar por dose. O atributo **doses necessárias** indica quantas doses da vacina devem ser tomadas. Opcionalmente, caso o valor de **doses necessárias** seja superior a 1, o atributo **intervalo temporal** indica de quantos em quantos meses a vacina deve ser tomada até que seja desenvolvida a imunidade.

- **Relacionamento Vacina-Veterinário–Animal**

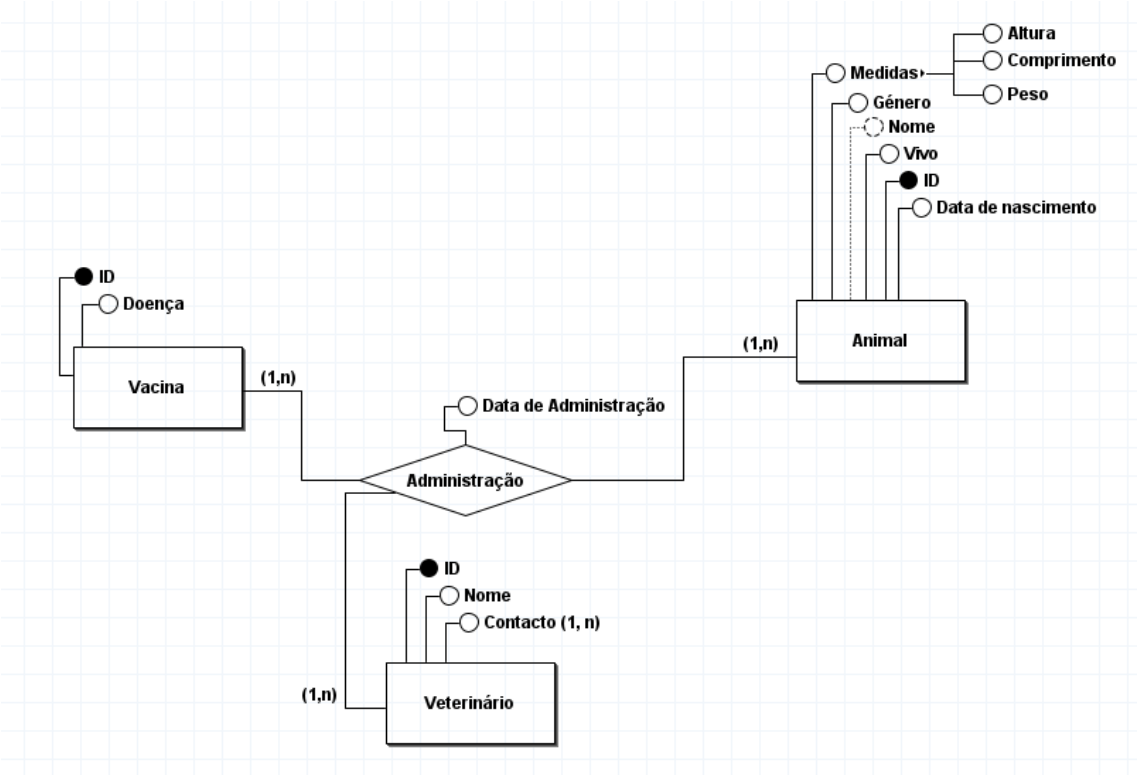

Figura 9 - Relacionamento Vacina-Veterinário–Anima**l**

Relacionamento: Vacinas é Administradas a Animais por Veterinários.

Descrição: Este relacionamento representa o ato de vacinar um Animal. Associa uma Vacina ao Veterinário que a administrou e ao Animal foi vacinado, mantendo um registo da data da inoculação.

Cardinalidade: Vacina (1..*) - Veterinário (1..*) - Animal.(1..*)

No ato de vacinação estas três entidades estão relacionadas no intuito de se manter um registo completo da vacinação para consulta futura, na eventualidade de alguma complicação médica. Cada Veterinário pode vacinar um ou mais Animais, administrando-lhes uma ou mais Vacinas. Cada Animal pode receber várias Vacinas administradas por vários Veterinários.

Atributos: O relacionamento é descrito pela **data de administração** da Vacina, de modo a ser possível registar várias doses da mesma Vacina no mesmo animal.

3.4. Identificação e Caracterização da Associação dos Atributos com as Entidades e Relacionamentos

Entidade	Atributos	Tipo de Dados	Nulo	Composto	Multivalor	Derivado	Candidato
Bilhete	ID	INT	Não	Não	Não	Não	Sim
	Momento de aquisição	DATETIME	Não	Não	Não	Não	Não
Tipo	ID	INT	Não	Não	Não	Não	Sim
	Nome	VARCHAR(45)	Não	Não	Não	Não	Sim
	Preço	FLOAT	Não	Não	Não	Não	Não
Zona	ID	INT	Não	Não	Não	Não	Sim
	Nome	VARCHAR(45)	Não	Não	Não	Não	Sim
	Total Animais	INT	Não	Não	Não	Sim	Não
Recinto	ID	INT	Não	Não	Não	Não	Sim
	Coordenadas	FLOAT	Não	Sim	Não	Não	Sim
	Área	FLOAT	Não	Não	Não	Não	Não
	Bioma	VARCHAR (100)	Não	Não	Não	Não	Não
Animal	ID	INT	Não	Não	Não	Não	Sim
	Nome	VARCHAR (100)	Sim	Não	Não	Não	Não
	Medidas	Decimal()	Não	Sim	Não	Não	Não
	Género	VARCHAR(9)	Não	Não	Não	Não	Não
	Vivo	BOOLEAN/ TINYINT	Não	Não	Não	Não	Não
	Data de Nascimento	DATE	Não	Não	Não	Não	Não
Padrinho	ID	INT	Não	Não	Não	Não	Sim
	Nome	VARCHAR(50)	Não	Não	Não	Não	Não
	NIB	VARCHAR(39)	Não	Não	Não	Não	Sim
	NIF	VARCHAR(8)	Não	Não	Não	Não	Sim
	Cartão de Cidadão	VARCHAR(13)	Não	Não	Não	Não	Sim
	Contacto	VARCHAR()	Não	Não	Sim	Não	Não
Espécie	ID	INT	Não	Não	Não	Não	Sim
	Nome Comum	VARCHAR(45)	Não	Não	Não	Não	Não
	Nome científico	VARCHAR(45)	Não	Não	Não	Não	Sim
	Valor de apadrinhamento	INT	Não	Não	Não	Não	Não
Vacina	ID	INT	Não	Não	Não	Não	Sim
	Doença	VARCHAR(60)	Não	Não	Não	Não	Não
Veterinário	ID	INT	Não	Não	Não	Não	Sim
	Nome	VARCHAR(50)	Não	Não	Não	Não	Não

	Contacto	VARCHAR()	Não	Não	Sim	Não	Não
Vacina - Espécie	Limite temporal	INT	Não	Não	Não	Não	Não
	Doses necessárias	INT	Não	Não	Não	Não	Não
	Dosagem	INT	Não	Não	Não	Não	Não
	Intervalo Temporal	INT	Sim	Não	Não	Não	Não
Vacina - Veterinário - Animal	Data de Administração	DATE	Não	Não	Não	Não	Não

Tabela 1 - Caracterização de todos os atributos existentes.

Os atributos guardam valores que descrevem a ocorrência das entidades e representam a parte principal dos dados guardados na base de dados.

A maioria dos atributos identificados são simples, com um único valor (*single-valued*), não são derivados e não podem ser nulos. Porém, existem algumas exceções, que passaremos a explicar em seguida.

- **Zona**

  Esta entidade apresenta ainda o atributo **total animais,** indicativo do número de Animais que se encontram abrigado nessa zona. Este será calculado a partir do número de Animais *vivos* existentes em cada recinto relacionado com a zona.

- **Recinto**

O atributo **coordenadas** será composto. Esta composição será formada pelo seu valor em x e em y.

- **Animal**

O atributo **medidas** é composto. Será constituído pelos atributos **altura**, **comprimento** e **peso.**

O atributo **nome** de um Animal é opcional. Isto deve-se ao facto de este só ser atribuído quando um Animal é adotado.

- **Padrinho**

O atributo **contacto** será multivalorado pois um Padrinho poderá ter mais do que uma forma de contacto.

- **Veterinário**

À semelhança da entidade Padrinho, também a entidade Veterinário terá o atributo multivalorado **contacto**, de forma a se puderem realizar diversas tentativas de contacto caso algum dos meios de comunicação não esteja funcional.

- **Vacina-Espécie**

O atributo **intervalo temporal** será opcional. Caso seja necessária mais do que uma dose da vacina este atributo indicará o numero de meses necessário entre a tomada de cada dose posterior.

3.4.1 Domínio dos Atributos

Descrição dos tipos de dados de cada atributo e dos domínios de valores que estes podem tomar:

- **Bilhete**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Momento de aquisição	DATETIME	Formato YYYY-MM-DD HH:MI:SS.

Tabela 2 - Caracterização dos atributos de Bilhete.

- **Tipo**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Nome	VARCHAR(45)	Sequência de palavras.
Preço	FLOAT	Valor positivo em euros

Tabela 3 - Caracterização dos atributos de Tipo.

- **Zona**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Nome	VARCHAR(45)	Sequência de palavras.
Total Animais	INT	Número inteiro positivo.

Tabela 4 - Caracterização dos atributos de Zona.

- **Recinto**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Coordenadas	FLOAT	Tuplo de coordenadas
Área	FLOAT	Número positivo em hectares
Bioma	VARCHAR (100)	Sequência de palavras.
Total Animais	INT	Número inteiro positivo.

Tabela 5 - Caracterização dos atributos de Recinto.

- **Animal**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Nome	VARCHAR (100)	Sequência de palavras.
Medidas	Decimal()	Atributo multivalorado composto por 3 valores decimais positivos: altura, peso e comprimento.
Género	VARCHAR(9)	Masculino/Feminino
Vivo	BOOLEAN/ TINYINT	0/1 (falecido/vivo)
Data de Nascimento	DATE	Formato YYYY-MM-DD.

Tabela 6 - Caracterização dos atributos de Divulgação.

- **Padrinho**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Nome	VARCHAR(50)	Sequência de palavras.
NIB	VARCHAR(39)	Sequência IBAN: 2 caracteres para país, 2 de validação e até 35 de código
NIF	VARCHAR(8)	Sequência de 8 dígitos válidos em m*ódulo 11*
Cartão de Cidadão	VARCHAR(13)	Sequência de 7 dígitos seguidos de um espaço e 4 dígitos
Contacto	VARCHAR()	Lista de sequências alfanuméricas

Tabela 7 - Caracterização dos atributos de Padrinho.

- **Espécie**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.

Nome Comum	VARCHAR(45)	Sequência de palavras.
Nome científico	VARCHAR(45)	Sequência de palavras.
Valor de apadrinhamento	INT	Número inteiro positivo.

Tabela 8 - Caracterização dos atributos de Espécie.

- **Vacina**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Doença	VARCHAR(60)	Sequência de palavras.

Tabela 9 - Caracterização dos atributos de Vacina.

- **Veterinário**

Atributos	Tipo de Dados	Domínio de Valores
ID	INT	Número inteiro positivo.
Nome	VARCHAR(50)	Sequência de palavras.
Contacto	VARCHAR()	Lista de sequências alfanuméricas

Tabela 10 - Caracterização dos atributos de Veterinário.

- **Relacionamento Vacina-Espécie**

Atributos	Tipo de Dados	Domínio de Valores
Limite temporal	DATE	Número inteiro positivo simbolizando o número de meses
Doses necessárias	INT	Número inteiro positivo
Dosagem	INT	Número inteiro positive em mililitros.
Intervalo Temporal	DATE	Número inteiro positivo simbolizando o número de meses

Tabela 11 - Caracterização dos atributos do relacionamento Vacina-Espécie.

- **Relacionamento Vacina-Veterinário-Animal**

Atributos	Tipo de Dados	Domínio de Valores
Data de Administração	DATE	Formato YY-MM-DD

Tabela 12 - Caracterização dos atributos do relacionamento Vacina-Veterinário-Animal.

3.4.2. Chaves Candidatas, Primárias e Alternativas

A coluna **candidato** permite sinalizar quais atributos de uma entidade terão possibilidade de ser chames primárias.

Nas entidades em que apenas o atributo ID se configura como candidato, será esse o atributo utilizado como chave primária. Este será o caso nas entidades **Bilhete, Animal, Vacina** e **Veterinário**. As chaves primárias dos relacionamentos serão compostas pelas chaves primárias das entidades relacionadas, em particular no caso dos relacionamentos **Tipo-Zona**, **Animal-Animal**, **Vacina-Veterinário-Animal** e **Vacina-Espécie.**

Para as restantes entidades em que existe mais do que uma chave candidata é necessário avaliar qual a melhor candidata. A filosofia adotada pelo grupo foi a seguinte: "Qualquer valor que tenha um significado fora da base de dados não deve ser usado como chave primária", o que implicou que a chave primária de qualquer entidade fosse um ID, abstrato de significado fora da base de dados. Procede-se à identificação das chaves alternativas das entidades em que, embora se tenha escolhido o ID como chave primária, havia outras chaves candidatas:

- **Tipo**

  Chaves Candidatas: ID, Nome.

  Chave Primária: ID.

  Chaves Alternativas: Nome.

- **Zona**

  Chaves Candidatas: ID, Nome.

  Chave Primária: ID.

  Chaves Alternativas: Nome.

- **Recinto**

  Chaves Candidatas: ID, Coordenadas.

  Chave Primária: ID.

  Chaves Alternativas: Coordenadas.

- **Padrinho**

  Chaves Candidatas: ID, NIB, NIF, Cartão de cidadão.

  Chave Primária: ID.

  Chaves Alternativas: NIB, NIF, Cartão de cidadão.

Embora existam vários valores únicos associados a um Padrinho, todos eles se revelam bastante complexos e com significado inerente.

- **Espécie**

Chaves Candidatas: ID, Nome científico.

Chave Primária: Id.

Chaves Alternativas: Nome científico.

3.5 Detalhe ou Generalização de Entidades

Não se detetaram semelhanças significativas entre as várias entidades do modelo conceptual para se justificar a utilização de herança.

Numa iteração anterior, em que se considerou a entidade Cliente, esta poderia possivelmente estar associada hierarquicamente à entidade Padrinho. Porém, visto que a sua utilização foi descartada após discussão, essa hipótese de generalização não se encontra presente no diagrama conceptual.

Veterinário e Padrinho partilham o atributo **contacto,** porém também se julgou ineficaz abrigar as duas entidades sob generalização pois o número de atributos em conjunto não se revela justificativo para tal.

3.6 Apresentação e Explicação do Diagrama ER

O Diagrama ER representativo do nosso modelo conceptual configura-se do seguinte modo:

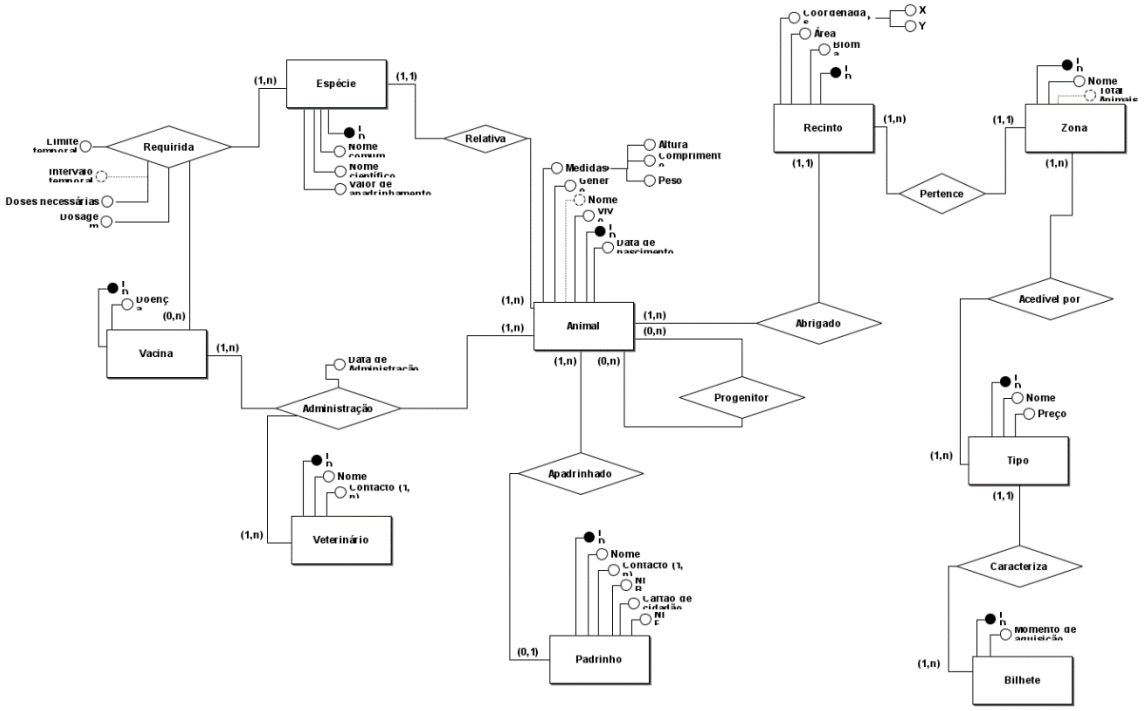

Figura 10 - Modelo Conceptual.

O presente diagrama pode ser interpretado da seguinte forma: cada Bilhete tem um tipo que garante acesso a certas Zonas. Cada Zona tem Recintos que abrigam Animais. Estes Animais podem ter crias e podem também ser apadrinhados por Padrinhos. Cada Animal pertence a uma Espécie e cada Espécie terá como requerimento tomar um conjunto de Vacinas. Cada Vacina é administrada numa data a um Animal por um Veterinário.

3.7 Validação do Modelo de Dados com o Utilizador

Devido ao facto de o Utilizador no contexto deste projeto ser o próprio grupo, verificar-se-á internamente se cada um dos Requisitos de Descrição foi satisfeito, pois são estes que inspiram a criação do modelo conceptual.

1. **Cada Bilhete é caracterizado pelo seu momento de aquisição.**

Figura 11 - Atributos de Organizador.

A entidade bilhete tem o atributo mencionado.

2. **Cada Bilhete é caracterizado por um Tipo.**

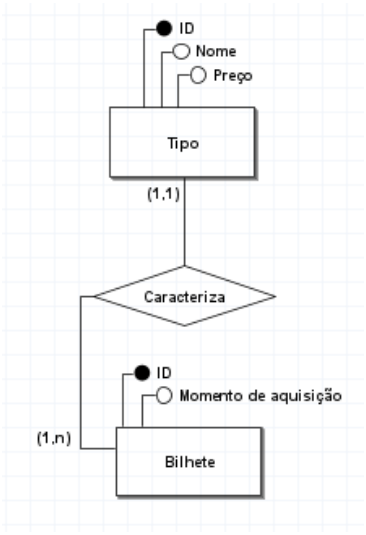

Figura 12 - Relacionamento Bilhete - Tipo

O relacionamento atribui o Tipo ao Bilhete.

3. **Cada Tipo é descrito pelo seu nome e preço.**

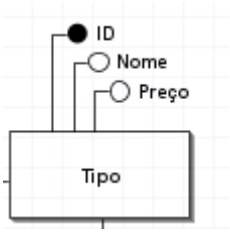

Figura 13 - Atributos de Tipo.

A entidade Tipo tem os atributos mencionados.

4. **Cada Zona é caracterizada pelo seu nome e derivará o número de Animais nela presentes;**

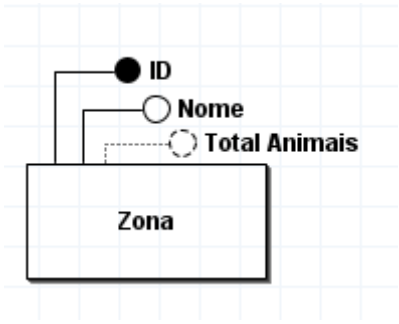

Figura 14 - Atributos de Zona

A entidade Zona tem os atributos mencionados. Total Animais será um atributo derivado que representa o número de Animais englobados numa zona.

5. **Cada Zona pode ser acedida por um conjunto de Tipos de Bilhete.**

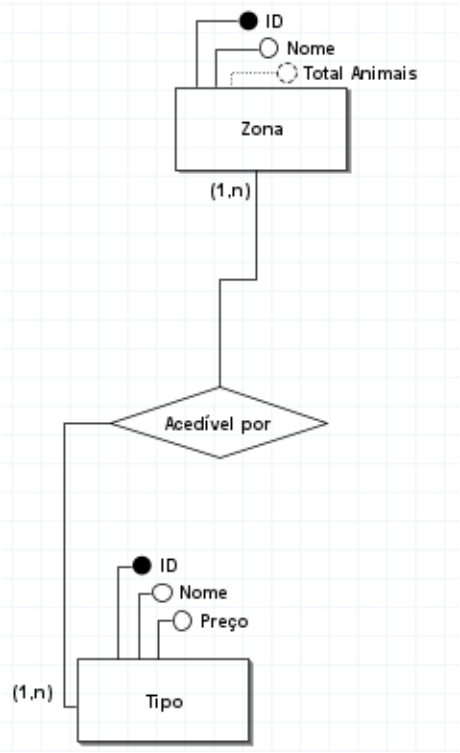

Figura 15 - Relacionamento Zona-Tipo

A relação garante acesso a cada Zona a um grupo diverso de Tipos de Bilhetes.

6. **Cada Recinto é caracterizado pelas suas coordenadas, a sua área e o bioma que simula. É caracterizado também pelo número de Animais vivos que alberga**.

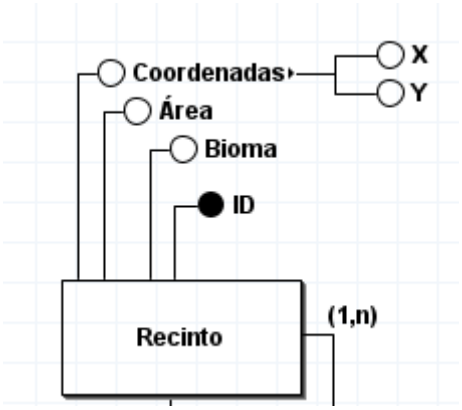

Figura 16 - Atributos de Recinto.

A entidade Recinto tem os atributos mencionados. O atributo relativo ao número de animais é derivado e, portanto, não se representa no diagrama conceptual.

7. **Cada Recinto pertence a uma Zona.**

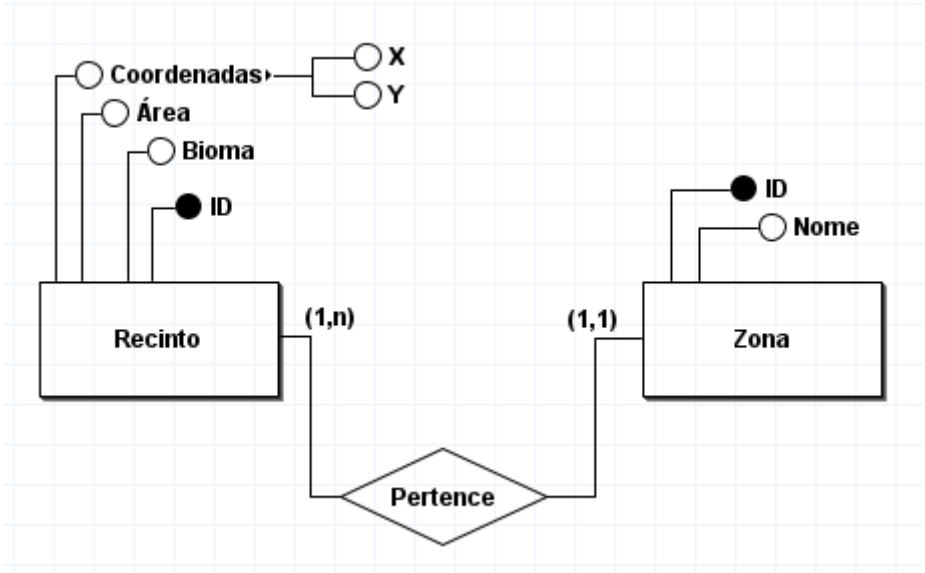

Figura 17 - Relacionamento Recinto-Zona

A relação garante que cada Recinto deve pertencer a uma Zona.

8. **Cada Animal é caracterizado pelas suas medidas, género, um indicador se o Animal ainda se encontra vivo e a sua data de nascimento. Poderá ter um nome atribuído caso tenha sido adotado.**

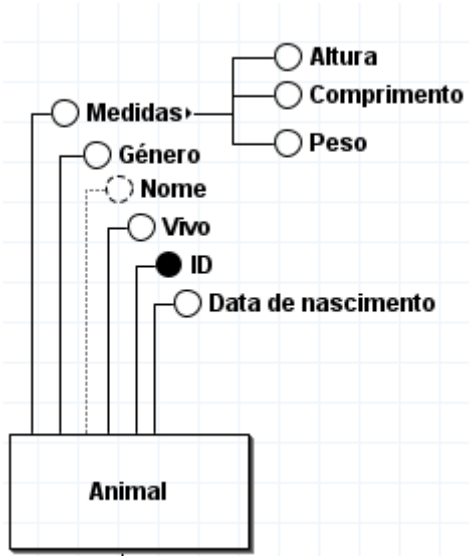

Figura 18 - Atributos de Animal

A entidade Animal tem os atributos mencionados. As medidas serão um atributo composto e o nome opcional.

9. Cada animal relacionar-se-á com as suas crias, caso as tenha.

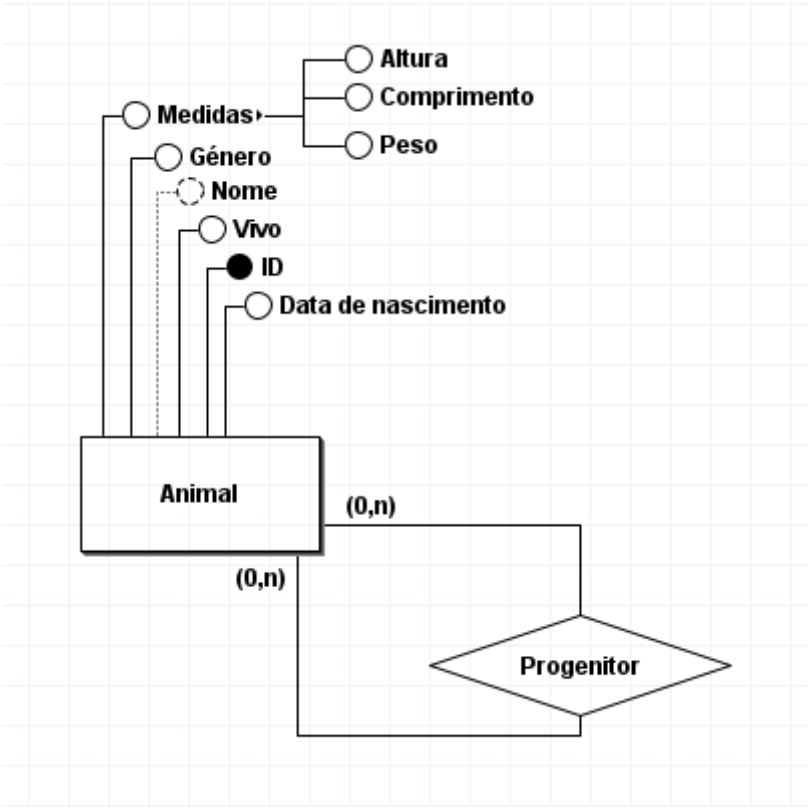

Figura 19 - Relacionamento Animal - Animal

Cada Animal poderá ser progenitor de um conjunto de outros Animais

10. Cada animal encontra-se abrigado num recinto.

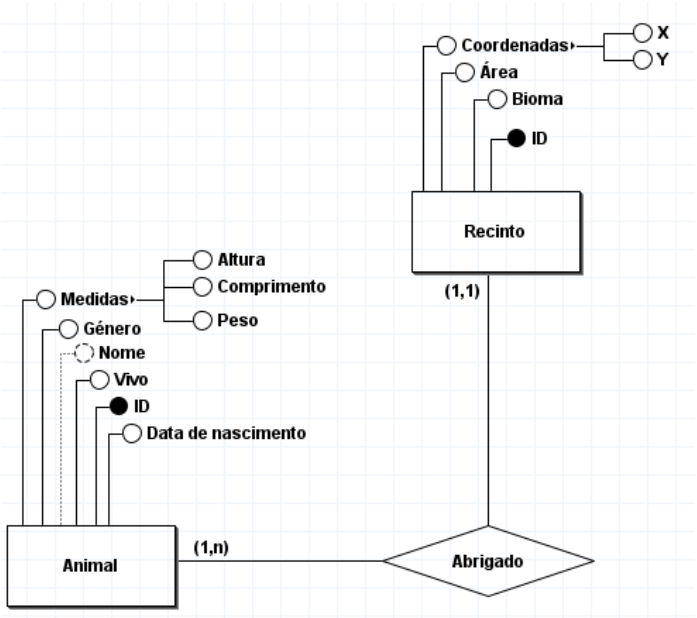

Figura 20 - Relacionamento Animal - Recinto

O relacionamento comprova a existência de N animais em cada Recinto.

11. Existirá uma Espécie relativa a cada Animal

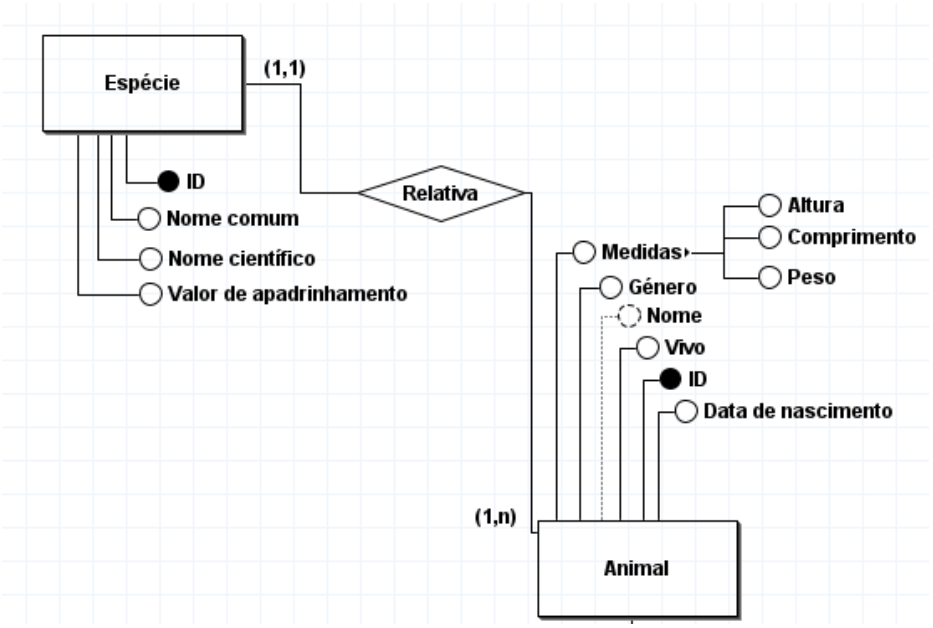

Figura 21- Relacionamento Espécie - Animal

O relacionamento comprova que cada Animal tem uma Espécie.

12. **A Espécie é caracterizada pelo seu nome comum, nome científico e qual é o valor monetário mensal necessário para apadrinhar um Animal pertencente à Espécie.**

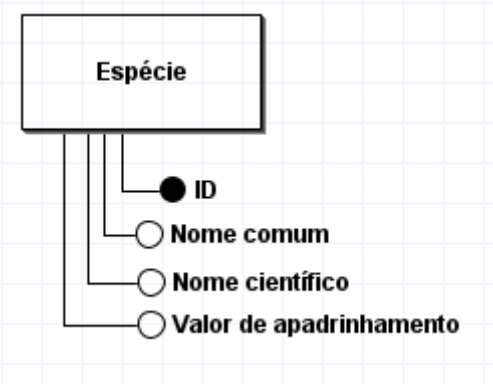

Figura 22 - Atributos de Espécie

13. **Um Padrinho será alguém que faz contribuições monetárias mensais a um ou mais Animais.**

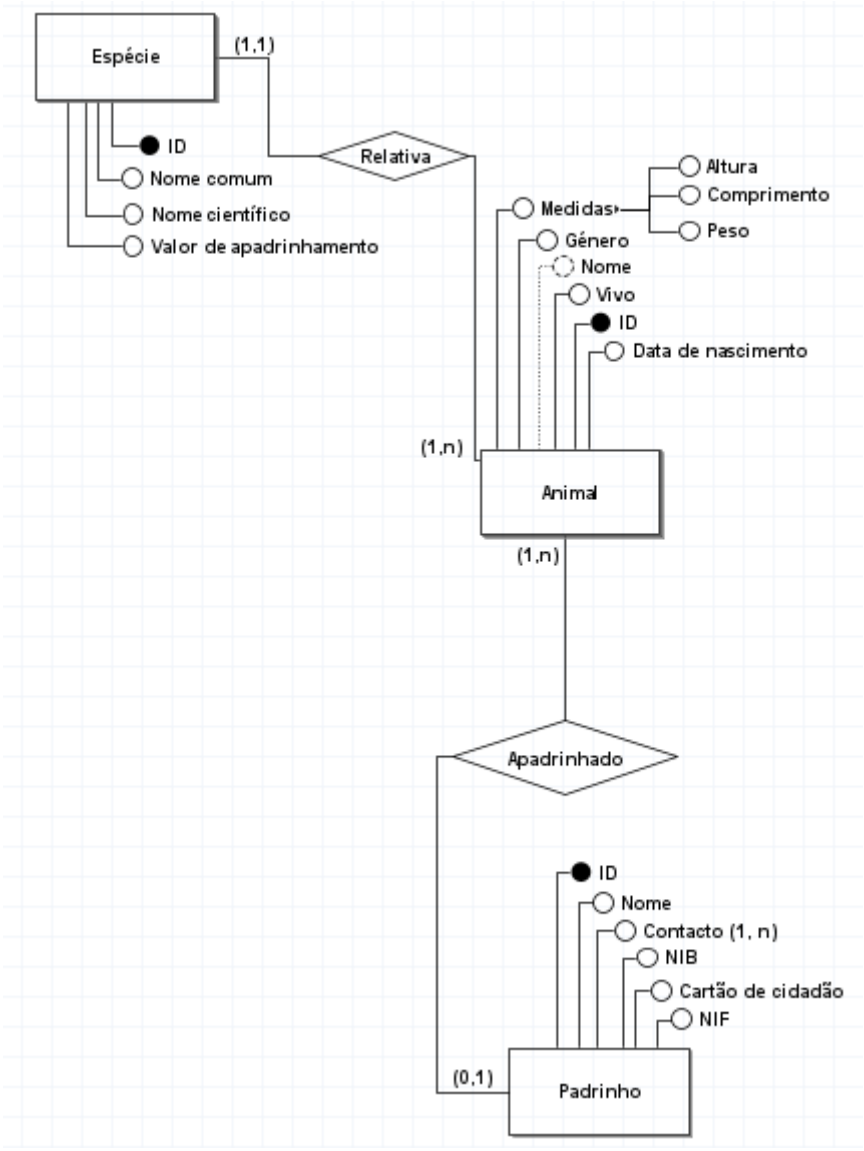

Figura 23 - Relacionamento Padrinho - Animal

O relacionamento indica que um Padrinho poderá ter um ou mais Animais. O valor a pagar por cada animal é obtido a partir do atributo **valor de apadrinhamento** da entidade Espécie que caracteriza Animal.

14. **Cada Padrinho será caracterizado pelo seu nome, uma lista de contactos, o seu NIB, cartão de cidadão e NIF.**

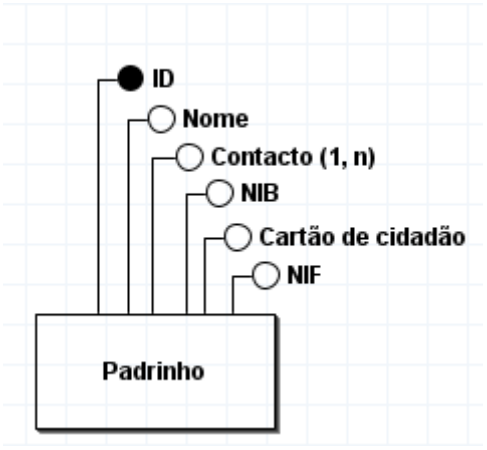

Figura 24 – Atributos de Padrinho

A entidade Padrinho tem os atributos mencionados. Como indicado, os contactos estarão inseridos numa lista.

15. **Cada Vacina é descrita pela doença a que é relativa**.

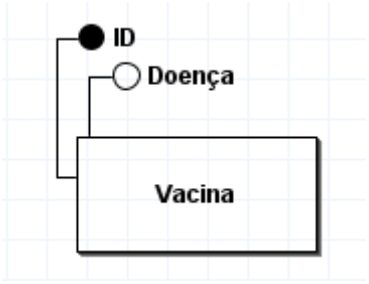

Figura 25 - Atributos de Vacina

A entidade Vacina tem o atributo mencionado.

16. **Uma vacina terá que ser administrada por uma ou mais espécies. Este relacionamento é descrito pela data limite em que se tem de tomar a primeira dose, quantas doses são necessárias e a dosagem por cada vacina (Ex: 10 ml). Caso as doses necessárias sejam mais do que uma, será indicado um intervalo temporal para a tomada de cada dose seguinte.**

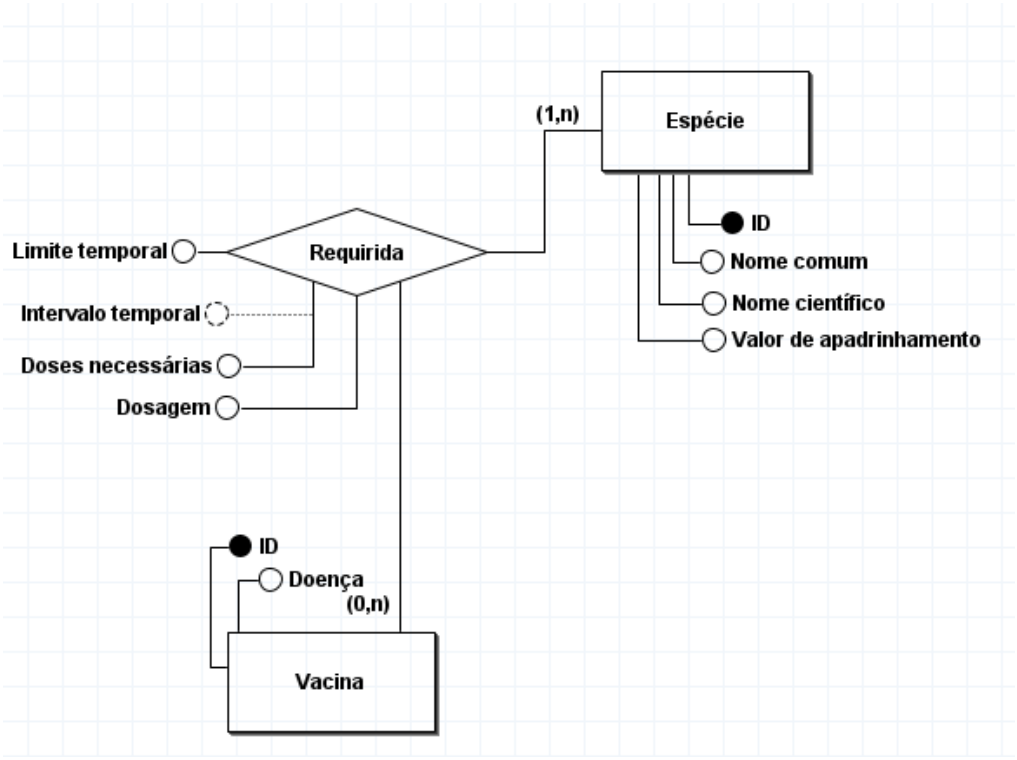

Figura 26 - Relação Vacina - Espécie

O relacionamento Vacina-Espécie indica que uma Espécie pode não ter Vacinas como pode também ter um conjunto delas. Cada Vacina deve estar associada a pelo menos uma Espécie. O atributo **intervalo temporal** está marcado como opcional.

17. Cada Vacina é administrada a um ou mais Animais, por um ou mais Veterinários.

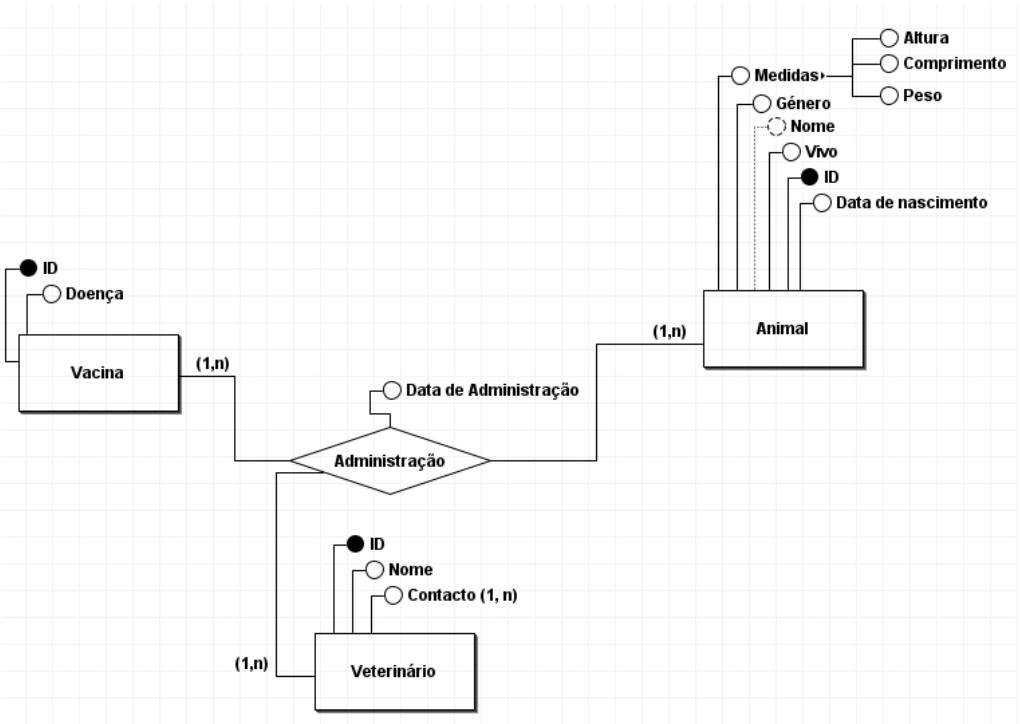

Figura 27 - Relacionamento Vacina-Veterinário-Animal

O relacionamento triplo liga as três entidades no ato de vacinação.

18. **Cada Veterinário é descrito pelo seu nome e lista de contactos.**

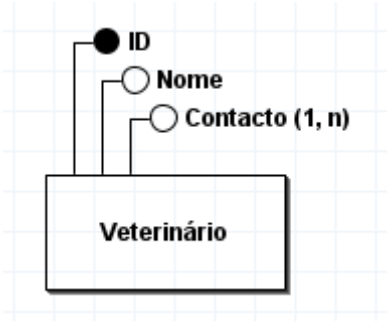

Figura 28 - Atributos de Veterinário.

A entidade Veterinário tem os atributos mencionados. Como indicado, os contactos estarão inseridos numa lista.

4 Modelação Lógica

4.1. Construção e Validação do Modelo de Dados Lógico

Para começar a conversão do modelo de dados conceptual para modelo de dados lógico agrupámos cada entidade e seus atributos numa tabela. Após este procedimento, atendendo às regras de derivação dos relacionamentos e de atributos multivalorados do modelo de dados conceptual, conclui-se assim a construção do modelo de dados lógico.

1. **Relacionamento binário de grau 1:N:**

Neste relacionamento são necessárias duas entidades lógicas. A entidade que identifica o grau N da relação terá que ter a referência da outra entidade. Essa referência será a chave primária que constará num atributo da entidade de multiplicidade de grau N.

Ocorrências:

- “Abrigado” entre Animal(N) e Recinto(1)
- “Pertence” entre Recinto(N) e Zona(1)
- “Carateriza” entre Zona(N) e Tipo(1)
- “Relativa” entre Animal(N) e Espécie(1)

2. **Relacionamento binário de grau N:M:**

Neste relacionamento serão necessárias três entidades lógicas. Para representar este relacionamento terá que ser criada uma entidade lógica do relacionamento que vai conter as chaves primárias das duas entidades intervenientes e os respetivos atributos da relação.

Ocorrências:

- “Acedível por” entre Tipo e Zona
- “Requerida” entre Vacina e Espécie

3. **Relacionamento ternário**

Neste relacionamento serão necessárias quatro entidades lógicas. Este relacionamento será semelhante à relação de grau N:M, apenas difere no número de entidades envolvidas na relação. Logo, é criada uma entidade lógica contendo as chaves primárias como atributos das três entidades da relação.

Ocorrência:

- “Administração” entre Vacina, Veterinário e Animal

4. **Autorrelacionamento de grau N:M:**

Neste relacionamento serão necessárias 2 entidades lógicas. Uma vez que este relacionamento se relaciona com uma entidade apenas, vamos ter uma entidade lógica que representará o autorrelacionamento, contendo as chaves primárias como atributo das duas entradas da entidade relacionadora.

Ocorrência:

- "Progenitor" entre duas entidades Animal

5. **Atributo multivalorado:**

Uma vez que um atributo apenas pode ter um valor em cada entidade, é necessário criar uma tabela para esse atributo multivalorado.

Ocorrência:

- "Contacto" em Padrinho
- "Contacto" em Veterinário

Entidades resultantes:

- **Bilhete** = {idBilhete, momento_aquisicao, Tipo_idTipo}
- **Tipo** = {idTipo, nome, preco}
- **Zona** = {idZona, nome}
- **Tipo_has_Zona** = {Zona_idZona, Tipo_idTipo}
- **Recinto** = {idRecinto, coord_y, coord_x, area, bioma, Zona_idZona}
- **Animal** = {idAnimal, genero, nome, data_nascimento, altura, comprimento, peso, vivo, Padrinho_idPadrinho, Recinto_idRecinto, Especie_idEspecie}
- **Animal_has_Animal** = {Animal_idAnimalProgenitor, Animal_idAnimalFilho}
- **Padrinho** = {idPadrinho, nome, nib, morada, cartao_cidadao, nif}
- **Contacto_Padrinho** = {Padrinho_idPadrinho, contacto}
- **Contacto_Veterinario** = {Veterinario_idVeterinario, contacto}
- **Veterinario** = {idVeterinario, nome}
- **Vacina** = {idVacina, doenca}
- **Especie** = {idEspecie, nome_comum, nome_cientifico, valor_de_apadrinhamento}
- **Animal_has_Vacina_has_Veterinario** = {Animal_idAnimal, Vacina_idVacina, Veterinario_idVeterinario, data_administracao}
- **Especie_has_vacina** = {Especie_idEspecie, Vacina_idVacina, limite_temporal, intervalo_temporal, doses_necessarias, dosagem}

4.2. Desenho do Modelo Lógico

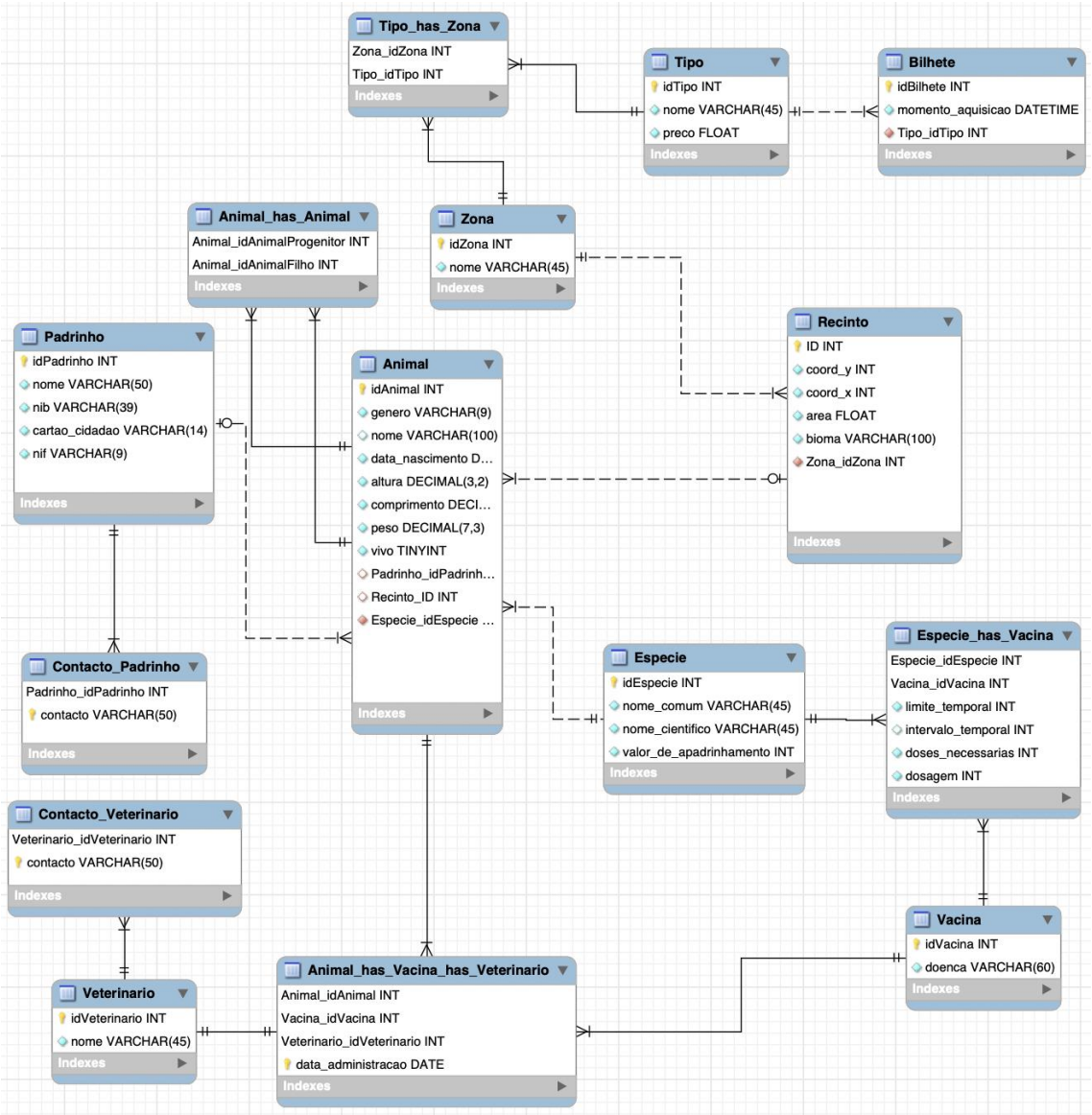

Figura 29 - Modelo Lógico.

4.3. Validação do Modelo através da Normalização

O modelo lógico encontra-se normalizado até à primeira forma normal. Pode-se ver, por exemplo, a relação Contacto_Padrinho e Padrinho. Como contacto é um atributo multivalorado de padrinho iria arrecadar em vários grupos repetidos se este atributo estivesse na entidade lógica Padrinho, logo, foi criada uma nova tabela separada que para conter essa informação.

O modelo lógico encontra-se normalizado até à segunda forma normal. Um exemplo do uso desta regra é na relação Animal para Recinto. Se se agrupasse estas duas tabelas numa só de Animal, iriam conter informações que não dependeriam do idAnimal, como as dimensões do Recinto, logo, estas informações foram separadas para contribuir para a normalização e coerência do modelo.

Por fim, o modelo encontra-se também normalizado até à terceira forma normal. No modelo não temos qualquer atributo que dependa de um outro. Um caso que poderia acontecer era haver um atributo idade na tabela Animal. Isso seria redundante uma vez que com a data de nascimento já seria possível chegar à idade do animal, não haveria por isso qualquer razão para ter esse atributo.

4.4. Validação do Modelo com as Interrogações do Utilizador

Como forma de validação do modelo lógico, verificou-se se seria possível satisfazer as interrogações definidas sob a forma de requisitos de exploração. Para tal, prosseguiu-se à representação, em álgebra relacional, de algumas das interrogações.

A ferramenta utilizada foi o **RelaX – Relational Algebra Calculator**, que permite a representação de árvores lógicas. Este programa apresenta, no entanto, algumas limitações no que toca à disponibilização de operadores de lógica extendida, tendo o grupo ficado limitado na quantidade de interrogações que conseguiu modelar, visto que muitos operadores disponibilizados pelo *MySQL*, como a cláusula WITH RECURSIVE não têm equivalente lógico no RelaX.

1. **Consultar qual o valor mensal total que cada padrinho deve pagar.**

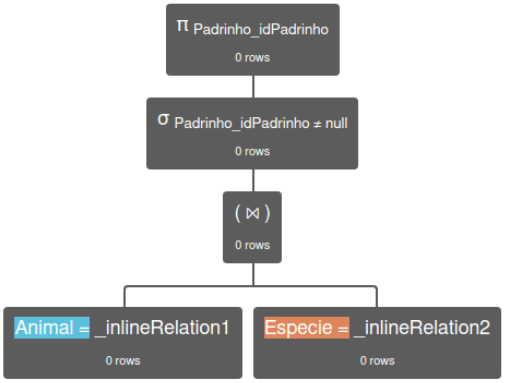

Figura 30 - Árvore representativa da Interrogação n.º 2.

πPadrinho_idPadrinho,sum(valor_de_apadrinhamento) (σPadrinho_idPadrinho≠null (Animal ⋈ Especie))

Em primeiro lugar, é feito um Inner Join entre as tabelas Animal e Especie através da coluna que representa o identificador de espécie removendo depois todas as linhas desta tabela que possuiam o idPadrinho a null. Depois disto agrupa as linhas por idPadrinho e apresenta o id do padrinho e a soma dos valores de cada linha das colunas de valor_de_apadrinhamento para apresentar o valor total a pagar nesse mês.

2. **Conhecer todas as espécies que um tipo de bilhete dá acesso.**

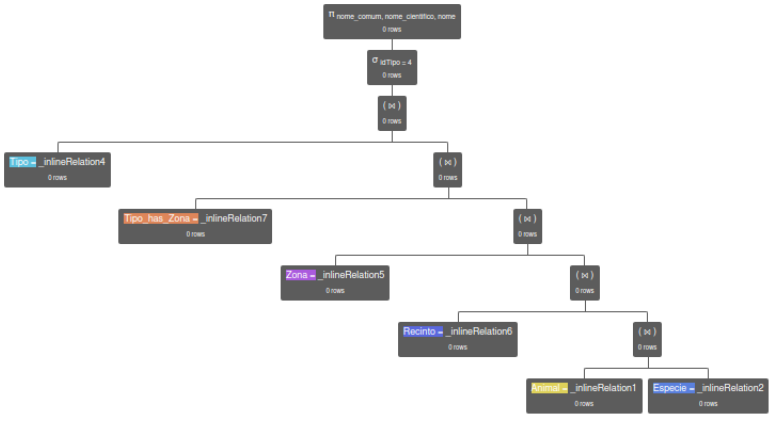

Figura 31 - Árvore representativa da Interrogação n.º 7.

πnome_comum,nome_cientifico,nome (σidTipo=idTipoDado (Tipo ⋈ (Tipo_has_Zona ⋈ (Zona ⋈ (Recinto ⋈ (Animal ⋈ Especie))))))

Seja idTipoDado o id do tipo dado como argumento na chamada do procedure, esta query faz um Inner Join entre Animal e Espécie através do identificador de espécie de cada tabela, após isto é feito um novo Inner Join entre recinto e a tabela anteriormente obtida através do identificador de recinto, seguidamente é feito um novo Inner Join entre a tabela Zona e a tabela que obtemos através do identificador de zona, posteriormente foi feito um Inner Join entre a tabela Tipo_has_Zona e a tabela deste momento através do identificador de Zona, por último foi feito um Inner Join entre a tabela Tipo e a tabela obtida através do identificador de Tipo, selecionando depois aqueles em que o identificador do tipo é igual ao argumento idTipoDado, sendo finalmente feito uma projeção do nome_comum,nome_científico e nome projetando depois aqueles em que o identificador do tipo é igual.

4.5. Revisão do Modelo Lógico produzido

Após conversão do modelo conceptual para modelo lógico, seguindo as normas estabelecidas para o efeito, verifica-se que todas as entidades do modelo anterior constam no modelo lógico, tal como as relações, respeitando, assim, as regras delimitadas para este tipo de conversão.

Depois das várias análises referidas nos capítulos anteriores concluímos que conseguimos construir um modelo lógico consistente que respeita os requisitos impostos no início do projeto e que nos permitirá criar um modelo físico que consiga suportar estas funcionalidades.

Com isto estamos assim em posição para avançar para o desenvolvimento do respetivo modelo físico do projeto.

5 Implementação Física

5.1. Seleção do Sistema de Gestão de Base de Dados

Por sugestão da equipa docente da unidade curricular de Base de Dados, o Sistema de Gestão de Base de Dados selecionado para implementação do projeto foi o *MySQL.*

O *MySQL* é um Sistema de Gestão de Bases de Dados *open-source* cujo paradigma é relacional. Este é um dos sistemas mais utilizados no mundo, devido à sua facilidade de utilização e às qualidades supramencionadas.

O *engine* utilizado será o *InnoDB,* as funcionalidades padrões de transação complacentes com *ACID: atomicity, consistency, isolation, durability.*

5.2. Tradução do Esquema Lógico para o Sistema de Gestão de Bases de Dados escolhido em SQL

Em conjunção com o *MySQL*, foi utilizada a ferramenta *MySQL Workbench*. Esta ferramente permite, após a definição do modelo lógico, utilizar o mecanismo de *Forward Engineering.*

Este permite, quando lhe é dado o modelo lógico, gerar o código de criação do sistema de gestão de base de dados.

Assim sendo, o modelo lógico do projeto foi representado utilizando a ferramenta e posteriormente recorreu-se à funcionalidade de *Forward Engineering.*

O código do programa usado para povoar a base de dados encontra-se disponível no **Anexo 3.** Resumidamente este escreve para um ficheiro “PovoamentoBD.sql” as várias linhas de inserção de informação para a nossa base de dados tendo sido criado de forma a que as linhas emitidas fossem o mais aleatórias possível mas mantendo uma base bem assente na realidade: as datas das vacinas dadas a um animal não são inferiores à idade do mesmo, os dados de cada animal são calculados com base nos valores médios de altura, comprimento e peso da sua espécie para ambos os sexos (as fêmeas pesam tendencialmente menos 10% que os machos). Este código, entre outros permite que a nossa base de dados possa ser testada com um elevado número de entradas gerado de forma aleatória mas que faça sentido, tendo nomes, contactos, NIF,CC,etc.

5.3. Tradução das Interrogações do Utilizador para SQL

Após a definição da álgebra relacional das diferentes queries, foram adotados os métodos de conversão indicados para originar o código SQL correspondente.

Na definição das interrogações originadas recorremos às *stored procedures* que consistem em conjuntos de comandos que, dado um ou mais parâmetros de entrada, produz um output.

Optamos por este método uma vez que, na necessidade de correr uma dada interrogação, apenas será necessário chamar a *stored procedure* correspondente, acompanhada com os seus argumentos. Além disso, pode-se dizer que queries em *stored procedures* serão mais eficientes, pois são pré-processadas, sendo assim otimizadas no futuro.

Para certas interrogações também recorremos às vistas, que funcionam como tabelas auxiliares do sistema de base de dados. Com estas vistas foram possíveis realizar cálculos intermédios de modo a que o código seja, não só mais legível e apresentável, como eficiente e direto. Estas tabelas são geradas no momento de seleção, não sendo guardadas no sistema.

- **Exemplos de interrogações que foram implementadas:**

1. **É possível consultar que animais um padrinho apadrinhou.**

```
DELIMITER $$
CREATE PROCEDURE animaisApadrinhados
    (IN idPadrinho INT)
BEGIN
    SELECT idAnimal AS IDAnimal, nome AS Nome
        FROM Animal
        WHERE Padrinho_idPadrinho = idPadrinho;
END $$
```

Figura 32 - Imagem do código obtido para a *query* n.º 1.

2. **Pode-se determinar todos os descendentes de um animal no zoo.**

```
DELIMITER $$
CREATE PROCEDURE descendentesAnimal
    (IN idAnimal INT)
BEGIN
    WITH RECURSIVE arvoreDescendente AS (
    SELECT Animal_idAnimalProgenitor AS IDProgenitor,Animal_idAnimalFilho AS IDCria, 1 AS Profundidade_Relativa
    FROM Animal_has_Animal
    WHERE Animal_idAnimalProgenitor = idAnimal

    UNION ALL

    SELECT pais.Animal_idAnimalProgenitor, pais.Animal_idAnimalFilho, aD.Profundidade_Relativa + 1
    FROM Animal_has_Animal pais, arvoreDescendente aD
    WHERE pais.Animal_idAnimalProgenitor = aD.IDCria
    )
    SELECT * FROM arvoreDescendente;
END $$
```

Figura 33 - Imagem do código obtido para a *query* n.º 4.

3. **É possível calcular o top 3 dos tipos de bilhetes mais comprados.**

```
DELIMITER $$
CREATE PROCEDURE top3TiposBilheteMaisComprados
    ()
BEGIN
SELECT T.nome AS TipoBilhete, COUNT(B.tipo_idTipo) AS NBilhetesVendidos
    FROM Bilhete as B
        INNER JOIN Tipo AS T
        ON B.Tipo_idTipo = T.idTipo
    GROUP BY B.tipo_idTipo
    ORDER BY COUNT(B.tipo_idTipo) DESC
    LIMIT 3;
END $$
```

Figura 34 - Imagem do código obtido para a *query* n.º 9.

4. **Calcular a média de pesos de animais de uma dada espécie.**

```
DELIMITER $$
CREATE PROCEDURE averageWeightSpecies
    (IN idEspecie INT)
BEGIN
SELECT e.nome_comum, e.nome_cientifico, AVG(a.peso) AS MediaPeso FROM Especie e, Animal a
    WHERE e.idEspecie = a.Especie_idEspecie
        AND e.idEspecie = idEspecie
            GROUP BY e.idEspecie;
END $$
```

Figura 35 - Imagem do código obtido para a *query* n.º 18.

5.4. Escolha, Definição e Caracterização de Índices em SQL

A utilização de índices em SQL permite reduzir o tempo gasto na procura de informação numa base de dados, tendo como custo um maior gasto em termos de espaço no armazenamento da informação à base de dados. A analogia de um índice de um livro que permite encontrar secções mais rápido mas que aumentará o tamanho de um livro devido às páginas gastas para a sua exposição é bastante adequada à utilização de índices em SQL.

Quando é utilizada a funcionalidade *Forward Engineering* no *SQL Workbench*, o script de inicialização da Base de Dados devolvido já criará um conjunto de índices, associados às chaves primárias e chaves estrangeiras.

A adição de mais índices teria o propósito de aumentar a velocidade de execução de interrogações associadas ao nosso Sistema de Gestão de Base de Dados, porém, o grupo constatou que nenhuma das interrogações apresenta um tempo de execução sub-satisfatório.

Para além do mais, como será discutido na secção seguinte, a única entidade que revelará um crescimento considerável na base de dados, ao longo do tempo, será a entidade Bilhete. No entanto, esta entidade já tem tanto a sua chave primária como a chave estrangeira que refere a entidade Tipo assinaladas como índices, sendo que o único outro atributo da entidade Bilhete será o seu momento de aquisição, cuja utilização como índice não se revela justificada.

Dado os factos apresentados, o grupo decidiu que não se revelaria como benéfico a adição de índices além dos já existentes.

5.5. Estimativa do Espaço em Disco da Base de Dados e Taxa de Crescimento Anual

De forma a estimar o espaço gasto por uma entrada de cada entidade existente na nossa Base de Dados foi necessário calcular quantos bytes serão gastos por cada tipo de dados. Para tal, foram consultadas as secções 11.1, 11.2, 11.3 e 11.7 do *MySQL 8.0 Reference Manual.*

Dessa consulta, estabeleceram-se as seguintes métricas de bytes por tipo de dado:

- **Int**: 4 bytes;
- **Float**:4 bytes;
- **Datetime**: 5 bytes;
- **Date:** 3 bytes;
- **Decimal(X,Y) –** 8 bytes quando X – Y = 9 e Y <9, para valores superiores consultar a tabela disponível na secção 11.7 do manual.

- **Varchar(N) –** N+1 bytes caso a string esteja codificada em ASCII, 2*N + 1 caso a codificação seja mais complexa mas o valor de 2*N seja menor que 255 e 2*N + 2 caso 2*N > 255;
- **TINYINT –** 1 byte.

Tendo em conta os valores definidos, explicitam-se os valores máximos que uma entrada de cada entidade e relação com atributos ocupará.

- **Bilhete**

Atributos	Tipo de Dados	Tamanho (em bytes)
idBilhete	INT	4
momento_aquisicao	DATETIME	5
Tipo_idTipo	INT	4

Tabela 13 – Tamanho (em bytes) dos atributos de Bilhete.

Uma entrada da entidade Bilhete ocupará 13 bytes.

- **Tipo**

Atributos	Tipo de Dados	Tamanho (em bytes)
idTipo	INT	4
nome	VARCHAR(45)	45*2+1 = 91
preco	FLOAT	4
Tipo_idTipo	INT	4

Tabela 14 - Tamanho (em bytes) dos atributos de Bilhete e entradas em tabelas associadas.

O campo nome poderá utilizar caracteres externos à tabela ASCII, sendo utilizado o segundo caso referido para VARCHAR, utilizando-se 2 bytes por caractere + 1 pois 45*2<255. O tamanho total de uma entrada da entidade Tipo ocupará 99 bytes. Tendo esta entidade uma referência obrigatória na tabela Tipo_has_Zona, adicionam-se 4 bytes ao custo total associado à inserção de uma entrada de uma entidade Tipo, totalizando 103 bytes.

- **Zona**

Atributos	Tipo de Dados	Tamanho (em bytes)
idZona	INT	4
nome	VARCHAR(45)	45*2+1 = 91
Zona_idZona	INT	4

Tabela 15 – Tamanho (em bytes) dos atributos de Zona.

O campo nome poderá utilizar caracteres externos à tabela ASCII, sendo utilizado o segundo caso referido para VARCHAR, utilizando-se 2 bytes por caracter + 1 pois 45*2<255. O tamanho total de uma entrada da entidade Zona ocupará 95 bytes. Tendo esta entidade uma referência obrigatória na tabela Tipo_has_Zona, adicionam-se 4 bytes ao custo total associado à inserção de uma entrada de uma entidade Tipo, totalizando 99 bytes.

- **Recinto**

Atributos	Tipo de Dados	Tamanho (em bytes)
ID	INT	4.
coord_y	FLOAT	4
coord_y	FLOAT	4
bioma	VARCHAR(100)	100*2+1=201
Zona_idZona	INT	4

Tabela 16 - Tamanho (em bytes) dos atributos de Recinto.

Uma entrada da entidade Recinto ocupará 217 bytes.

- **Animal**

Atributos	Tipo de Dados	Tamanho (em bytes)
idAnimal	INT	4
genero	VARCHAR (9)	9+1
nome	VARCHAR(100)	100*2+1
data_nascimento	DATETIME	5
altura	DECIMAL(3,2)	8
comprimento	DECIMAL(3,2)	8
peso	DECIMAL(7,3)	8
vivo	TINYINT	1
Padrinho_idPadrinho	INT	4
Recinto_ID	INT	4
Especie_idEspecie	INT	4
Animal_idAnimalProgenitor	INT	4
Animal_idAnimal	INT	4

Tabela 17 - Caracterização dos atributos de Divulgação.

Quanto a chaves estrangeiras, admitiu-se que todas se encontraram preenchidas.

Quanto a chaves pertencentes a tabelas representativas de relações N para N, como é impossível prever o número de relações progenitor-cria, assumiu-se que em média cada animal iria ter um progenitor e uma cria.

Totalizando os valores de cada atributo, uma entrada de Animal ocupará em média 260 bytes.

Tendo sido a mecânica completamente explicitada com os exemplos em cima, são expostos agora as restantes entidades e os seus custos associados.

- **Padrinho** – 170 bytes + (4 + 101)*3 = 485 bytes. Assumiu-se uma média de 3 contactos associados a um padrinho.
- **Espécie –** 190 bytes
- **Vacina –** 125 bytes
- **Veterinário –** 95 + (4 + 101)*3 = 410 bytes. Assumiu-se uma média de 3 contactos associados a um veterinário.
- **Relacionamento Vacina-Espécie –** 24 bytes
- **Relacionamento Vacina-Veterinário-Animal –** 15 bytes.

Utilizando as funcionalidades disponibilizadas pelo **MySql Workbench**, este revela que o tamanho atual da base de dados é de **1.4 Mib**.

Tendo em conta o contexto do nosso projeto, um jardim zoológico, é de esperar que a única entidade que tenha uma quantidade de inserções considerável na base de dados anualmente.

Este foi um fator tido em consideração na fase de modelação conceptual, pretendendo-se modelar a entidade Bilhete de forma a ocupar o menos espaço possível.

Estipulou-se uma venda anual de 100 000 bilhetes, o que representa um aumento anual de 100 000 *13 = 1 300 000 bytes, ou seja, **1 300 KB.**

Assumindo um crescimento de 10% anual no número de animais e tendo atualmente 248 animais vivos no sistema, este configura-se como um crescimento de round(248*0.1)*260 = 6 500 bytes, ou seja**, 6.5 KB**.

Assumindo que cada um destes animais será vacinado 5 vezes no seu primeiro ano de vida, configura-se como um crescimento de round(248*0.1)*5*15 = 1 875 bytes, **1.875 KB**.

O crescimento relativo às restantes entidades revela-se bastante imprevisível, no entanto é possível assumir que a sua variação será reduzida o suficiente de modo a que não terá um real impacto no crescimento da base de dados. A título de exemplo, não se observará um crescimento exponencial no número de novos tipos de bilhete, espécie ou veterinários.

Tendo em conta a informação acima mencionada, o crescimento total anual traduz-se no valor de 1 308.375 KB por ano, ou seja, **1277.70 KiB**.

5.6. Definição e Caracterização das vistas de utilização em SQL

- **Vista do Número de Bilhetes Anuais**

  Número de Bilhetes comprados anualmente desde a criação do Jardim Zoológico.

```
CREATE VIEW vwNBilhetesAno AS
    SELECT YEAR(B.momento_aquisicao) AS Ano,COUNT(B.idBilhete) AS NumeroBilhetes
        FROM Bilhete as B
        GROUP BY YEAR(B.momento_aquisicao)
        ORDER BY YEAR(B.momento_aquisicao) DESC;
```

Figura 36 - Vista de Bilhetes por Ano.

- **Vista do Número de Bilhetes Anuais em comparação ao ano Anterior**

  Esta vista foi construída a partir da vista anterior de forma a auxiliar na criação da interrogação número 13.

```
CREATE VIEW vwNbilhetesAnos AS
    SELECT Atual.Ano AS Ano, Atual.NumeroBilhetes AS NumeroBilhetes, Anterior.NumeroBilhetes AS NumeroBilhetesAnterior
        FROM (SELECT * FROM vwNBilhetesAno) AS Atual
        INNER JOIN (SELECT * FROM vwNBilhetesAno) AS Anterior
        ON Atual.Ano= Anterior.Ano+1;
```

Figura 37 - Vista de comparação de venda de Bilhetes por Ano.

- **Vista do Número de Animais por Zona**

  Esta vista irá conter o número de animais vivos por Zona do Jardim Zoológico. Foi construída pois achamos que esta informação pode ser útil de forma a ter uma melhor noção da distribuição por zonas dos animais.

```
CREATE VIEW vwNAnimaisZona AS
SELECT Z.nome AS NomeDeZona, COUNT(Z.idZona) AS NúmeroAnimais
    FROM Zona AS Z
        INNER JOIN Recinto AS R
        ON Z.idZona = R.Zona_idZona
        INNER JOIN Animal AS A
        ON A.Recinto_ID = R.ID
    WHERE A.vivo=1
    GROUP BY Z.idZona;
```

Figura 38 - Vista do Número de Animais por Zona.

5.7. Revisão do Sistema Implementado com o Utilizador

Após a conclusão da implementação da base de dados numa situação real seria agendada uma reunião final com a equipa de gestão do Jardim Zoológico de forma a validar a base de dados e possivelmente dar como concluído o desenvolvimento do sistema.

A revisão da implementação consistiria em 3 fases.

Na primeira fase iriamos mostrar que a base de dados conseguia responder às várias interrogações às quais nós nos tínhamos comprometido a poder responder com o cliente.

A segunda fase constataria em explicar ao cliente as estimativas de espaço ocupado da base de dados e da taxa de crescimento esperada tendo como fundação a informação anteriormente referida.

A fase final consistiria numa discussão com o cliente sobre as várias dúvidas que a apresentação poderia ter deixado acerca da implementação, bem como futuras possibilidades de crescimento e possíveis interrogações a adicionar.

Após terem sido esclarecidas todas as possíveis dúvidas e ter sido acordado que a atual implementação era a esperada partiríamos para a fase de instalação do sistema.

6. Conclusões e Trabalho Futuro

Através do desenvolvimento desta Base de Dados, o Jardim Zoológico vai inquestionavelmente ter uma capacidade de crescimento acrescida, bem como a segurança de que os seus animais têm toda a sua vacinação assegurada e no tempo regulamentado.

Estes benefícios advêm de finalmente ser feito uso de uma tecnologia de gestão, exploração e armazenamento de dados que seja capaz de cumprir com todos os requisitos de organização que o Jardim Zoológico impõe. Deste modo, podemos avançar com a confiança de que estamos a utilizar um sistema que nos permitirá resolver todos os desafios do presente, assim com os do futuro.

Através do desenvolvimento desta Base de Dados, o Jardim Zoológico vai inquestionavelmente ter uma capacidade de crescimento acrescida, bem como a segurança de que os seus animais têm toda a sua vacinação assegurada e no tempo regulamentado.

Ao longo deste relatório foram apresentadas as razões que levaram à criação e subsequente implementação desta Base de Dados, e quais os problemas que esta tem por vista solucionar, passando pela formalização do seu modelo conceptual e posterior conversão para modelo lógico, com todas as transformações associadas, e terminando com a formulação do modelo físico.

A aplicação deste método de desenvolvimento permite que, a qualquer momento, sejam implementadas novas funcionalidades de um modo simples e seguro, assegurando que qualquer expansão futura desta Base de Dados pode ser executada.

Dada a elegância do método adotado, e os esforços do Jardim Zoológico para a crescente utilização dos seus serviços, a migração, resultante da execução da solução de software desenvolvida, foi suave e não desencadeou qualquer tipo de complicações.

No futuro pretende-se que seja possível cumprir com as especificações delineadas na Introdução, como é o caso da Base de Dados ser expandida para ser algo utilizado a nível nacional conjuntamente com outros jardins zoológicos, e esta foi já desenvolvida com vista a suportar esta e qualquer expansão que da Base de Dados possa vir a sofrer durante quer a sua evolução, quer a do Jardim Zoológico.

7. Referências Bibliográficas

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

Mysql. (2018). MySql. [Online]. [04 December 2020]. Available from: https://dev.mysql.com/doc/refman/8.0/en/.

Lista de Siglas e Acrónimos

SQL *Structured Query Language*

ER *Entity–Relationship*

I. Anexo 1 – *Script* de Inicialização da Base de Dados

```
-- 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_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema BD_Zoologico
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema BD_Zoologico
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `BD_Zoologico` ;
CREATE SCHEMA IF NOT EXISTS `BD_Zoologico` ;
-- -----------------------------------------------------
-- Schema test
-- -----------------------------------------------------
USE `BD_Zoologico` ;

-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Padrinho`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Padrinho` (
  `idPadrinho` INT NOT NULL,
  `nome` VARCHAR(50) NOT NULL,
  `nib` VARCHAR(39) NOT NULL,
  `cartao_cidadao` VARCHAR(14) NOT NULL,
  `nif` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`idPadrinho`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Zona`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Zona` (
  `idZona` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idZona`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Recinto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Recinto` (
  `ID` INT NOT NULL,
  `coord_y` INT NOT NULL,
  `coord_x` INT NOT NULL,
  `area` FLOAT NOT NULL,
  `bioma` VARCHAR(100) NOT NULL,
  `Zona_idZona` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Recinto_Zona1_idx` (`Zona_idZona` ASC) VISIBLE,
  CONSTRAINT `fk_Recinto_Zona1`
    FOREIGN KEY (`Zona_idZona`)
    REFERENCES `BD_Zoologico`.`Zona` (`idZona`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
ROW_FORMAT = Default;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Especie`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Especie` (
  `idEspecie` INT NOT NULL,
  `nome_comum` VARCHAR(45) NOT NULL,
  `nome_cientifico` VARCHAR(45) NOT NULL,
  `valor_de_apadrinhamento` INT NOT NULL,
  PRIMARY KEY (`idEspecie`))
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Animal`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Animal` (
  `idAnimal` INT NOT NULL,
  `genero` VARCHAR(9) NOT NULL,
  `nome` VARCHAR(100) NULL,
  `data_nascimento` DATE NOT NULL,
  `altura` DECIMAL(3,2) NOT NULL,
  `comprimento` DECIMAL(3,2) NOT NULL,
  `peso` DECIMAL(7,3) NOT NULL,
  `vivo` TINYINT NOT NULL,
  `Padrinho_idPadrinho` INT NULL,
  `Recinto_ID` INT NULL,
  `Especie_idEspecie` INT NOT NULL,
  PRIMARY KEY (`idAnimal`),
  INDEX `fk_Animal_Padrinho1_idx` (`Padrinho_idPadrinho` ASC) VISIBLE,
  INDEX `fk_Animal_Recinto1_idx` (`Recinto_ID` ASC) VISIBLE,
  INDEX `fk_Animal_Especie1_idx` (`Especie_idEspecie` ASC) VISIBLE,
  CONSTRAINT `fk_Animal_Padrinho1`
    FOREIGN KEY (`Padrinho_idPadrinho`)
    REFERENCES `BD_Zoologico`.`Padrinho` (`idPadrinho`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_Recinto1`
    FOREIGN KEY (`Recinto_ID`)
    REFERENCES `BD_Zoologico`.`Recinto` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_Especie1`
    FOREIGN KEY (`Especie_idEspecie`)
    REFERENCES `BD_Zoologico`.`Especie` (`idEspecie`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Tipo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Tipo` (
  `idTipo` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `preco` FLOAT NOT NULL,
  PRIMARY KEY (`idTipo`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Bilhete`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Bilhete` (
  `idBilhete` INT NOT NULL,
  `momento_aquisicao` DATETIME NOT NULL,
  `Tipo_idTipo` INT NOT NULL,
  PRIMARY KEY (`idBilhete`),
  INDEX `fk_Bilhete_Tipo1_idx` (`Tipo_idTipo` ASC) VISIBLE,
  CONSTRAINT `fk_Bilhete_Tipo1`
    FOREIGN KEY (`Tipo_idTipo`)
    REFERENCES `BD_Zoologico`.`Tipo` (`idTipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Vacina`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Vacina` (
  `idVacina` INT NOT NULL,
  `doenca` VARCHAR(60) NOT NULL,
  PRIMARY KEY (`idVacina`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Veterinario`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Veterinario` (
  `idVeterinario` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idVeterinario`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Animal_has_Vacina_has_Veterinario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Animal_has_Vacina_has_Veterinario` (
  `Animal_idAnimal` INT NOT NULL,
  `Vacina_idVacina` INT NOT NULL,
  `Veterinario_idVeterinario` INT NOT NULL,
  `data_administracao` DATE NOT NULL,
  INDEX `fk_Animal_has_Vacina_Vacina1_idx` (`Vacina_idVacina` ASC) VISIBLE,
  INDEX `fk_Animal_has_Vacina_Animal1_idx` (`Animal_idAnimal` ASC) VISIBLE,
  INDEX `fk_Animal_has_Vacina_has_Veterinario_Veterinario1_idx` (`Veterinario_idVeterinario`
ASC) VISIBLE,
  PRIMARY KEY (`Animal_idAnimal`, `Vacina_idVacina`, `Veterinario_idVeterinario`,
`data_administracao`),
  CONSTRAINT `fk_Animal_has_Vacina_Animal1`
    FOREIGN KEY (`Animal_idAnimal`)
    REFERENCES `BD_Zoologico`.`Animal` (`idAnimal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_has_Vacina_Vacina1`
    FOREIGN KEY (`Vacina_idVacina`)
    REFERENCES `BD_Zoologico`.`Vacina` (`idVacina`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_has_Vacina_has_Veterinario_Veterinario1`
    FOREIGN KEY (`Veterinario_idVeterinario`)
    REFERENCES `BD_Zoologico`.`Veterinario` (`idVeterinario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Tipo_has_Zona`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Tipo_has_Zona` (
  `Zona_idZona` INT NOT NULL,
  `Tipo_idTipo` INT NOT NULL,
  INDEX `fk_Tipo_has_Zona_Zona1_idx` (`Zona_idZona` ASC) VISIBLE,
  INDEX `fk_Tipo_has_Zona_Tipo1_idx` (`Tipo_idTipo` ASC) VISIBLE,
  PRIMARY KEY (`Zona_idZona`, `Tipo_idTipo`),
  CONSTRAINT `fk_Tipo_has_Zona_Tipo1`
    FOREIGN KEY (`Tipo_idTipo`)
    REFERENCES `BD_Zoologico`.`Tipo` (`idTipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tipo_has_Zona_Zona1`
    FOREIGN KEY (`Zona_idZona`)
    REFERENCES `BD_Zoologico`.`Zona` (`idZona`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Contacto_Padrinho`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Contacto_Padrinho` (
  `Padrinho_idPadrinho` INT NOT NULL,
  `contacto` VARCHAR(50) NOT NULL,
  INDEX `fk_Contacto_Padrinho1_idx` (`Padrinho_idPadrinho` ASC) VISIBLE,
  PRIMARY KEY (`Padrinho_idPadrinho`, `contacto`),
  CONSTRAINT `fk_Contacto_Padrinho1`
    FOREIGN KEY (`Padrinho_idPadrinho`)
    REFERENCES `BD_Zoologico`.`Padrinho` (`idPadrinho`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Especie_has_Vacina`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Especie_has_Vacina` (
  `Especie_idEspecie` INT NOT NULL,
  `Vacina_idVacina` INT NOT NULL,
  `limite_temporal` INT NOT NULL,
  `intervalo_temporal` INT NULL,
  `doses_necessarias` INT NOT NULL,
  `dosagem` INT NOT NULL,
  INDEX `fk_Especie_has_Vacina_Vacina1_idx` (`Vacina_idVacina` ASC) VISIBLE,
  INDEX `fk_Especie_has_Vacina_Especie1_idx` (`Especie_idEspecie` ASC) VISIBLE,
  PRIMARY KEY (`Especie_idEspecie`, `Vacina_idVacina`),
  CONSTRAINT `fk_Especie_has_Vacina_Especie1`
    FOREIGN KEY (`Especie_idEspecie`)
    REFERENCES `BD_Zoologico`.`Especie` (`idEspecie`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Especie_has_Vacina_Vacina1`
    FOREIGN KEY (`Vacina_idVacina`)
    REFERENCES `BD_Zoologico`.`Vacina` (`idVacina`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Animal_has_Animal`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Animal_has_Animal` (
  `Animal_idAnimalProgenitor` INT NOT NULL,
  `Animal_idAnimalFilho` INT NOT NULL,
  INDEX `fk_Animal_has_Animal_Animal2_idx` (`Animal_idAnimalFilho` ASC) VISIBLE,
  INDEX `fk_Animal_has_Animal_Animal1_idx` (`Animal_idAnimalProgenitor` ASC) VISIBLE,
  PRIMARY KEY (`Animal_idAnimalProgenitor`, `Animal_idAnimalFilho`),
  CONSTRAINT `fk_Animal_has_Animal_Animal1`
    FOREIGN KEY (`Animal_idAnimalProgenitor`)
    REFERENCES `BD_Zoologico`.`Animal` (`idAnimal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Animal_has_Animal_Animal2`
    FOREIGN KEY (`Animal_idAnimalFilho`)
```

```
    REFERENCES `BD_Zoologico`.`Animal` (`idAnimal`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BD_Zoologico`.`Contacto_Veterinario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BD_Zoologico`.`Contacto_Veterinario` (
  `Veterinario_idVeterinario` INT NOT NULL,
  `contacto` VARCHAR(50) NOT NULL,
  INDEX  `fk_Contacto_Padrinho_copy1_Veterinario1_idx`  (`Veterinario_idVeterinario`  ASC)
VISIBLE,
  PRIMARY KEY (`Veterinario_idVeterinario`, `contacto`),
  CONSTRAINT `fk_Contacto_Padrinho_copy1_Veterinario1`
    FOREIGN KEY (`Veterinario_idVeterinario`)
    REFERENCES `BD_Zoologico`.`Veterinario` (`idVeterinario`)
    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;
```

II. Anexo 2 – *Script* Fundamental de Povoamento Inicial

```
USE `BD_Zoologico` ;

INSERT INTO Zona VALUES (1,"Aviário");
INSERT INTO Zona VALUES (2,"Templo dos Primatas");
INSERT INTO Zona VALUES (3,"Encosta dos Felinos");
INSERT INTO Zona VALUES (4,"Reptilarium");
INSERT INTO Zona VALUES (5,"África Selvagem");
INSERT INTO Zona VALUES (6,"Austrália");
INSERT INTO Zona VALUES (7,"Pólos");
INSERT INTO Zona VALUES (8,"Ásia");
INSERT INTO Zona VALUES (9,"Atlântida");

INSERT INTO Tipo VALUES (1,"Dança das Aves",14.2);
INSERT INTO Tipo VALUES (2,"Caminhantes Terrestres",22.2);
INSERT INTO Tipo VALUES (3,"Zona dos Répteis",7.2);
INSERT INTO Tipo VALUES (4,"Paraíso Submerso",16.2);
INSERT INTO Tipo VALUES (5,"Acesso Total",30.4);
INSERT INTO Tipo VALUES (6,"Criança",10.4);

INSERT INTO Tipo_has_Zona VALUES (1,1);
INSERT INTO Tipo_has_Zona VALUES (1,5);
INSERT INTO Tipo_has_Zona VALUES (2,5);
INSERT INTO Tipo_has_Zona VALUES (3,5);
INSERT INTO Tipo_has_Zona VALUES (4,5);
INSERT INTO Tipo_has_Zona VALUES (5,5);
INSERT INTO Tipo_has_Zona VALUES (6,5);
INSERT INTO Tipo_has_Zona VALUES (7,5);
INSERT INTO Tipo_has_Zona VALUES (8,5);
INSERT INTO Tipo_has_Zona VALUES (9,5);
INSERT INTO Tipo_has_Zona VALUES (7,4);
INSERT INTO Tipo_has_Zona VALUES (9,4);
INSERT INTO Tipo_has_Zona VALUES (4,3);
INSERT INTO Tipo_has_Zona VALUES (2,2);
```

```
INSERT INTO Tipo_has_Zona VALUES (3,2);
INSERT INTO Tipo_has_Zona VALUES (4,2);
INSERT INTO Tipo_has_Zona VALUES (5,2);
INSERT INTO Tipo_has_Zona VALUES (6,2);
INSERT INTO Tipo_has_Zona VALUES (7,2);
INSERT INTO Tipo_has_Zona VALUES (8,2);
INSERT INTO Tipo_has_Zona VALUES (1,6);
INSERT INTO Tipo_has_Zona VALUES (2,6);
INSERT INTO Tipo_has_Zona VALUES (3,6);
INSERT INTO Tipo_has_Zona VALUES (4,6);
INSERT INTO Tipo_has_Zona VALUES (5,6);
INSERT INTO Tipo_has_Zona VALUES (6,6);
INSERT INTO Tipo_has_Zona VALUES (7,6);
INSERT INTO Tipo_has_Zona VALUES (8,6);
INSERT INTO Tipo_has_Zona VALUES (9,6);

INSERT INTO Vacina VALUES (1,"Vacina de Temperamento"); -- dog like animals and bears  -- 21,50,36
INSERT INTO Vacina VALUES (2,"Vacina da Raiva"); -- all
INSERT INTO Vacina VALUES (3,"Vacina da Panleukopenia"); -- cat like animals -- 1,2,5,16,17,40,48
INSERT INTO Vacina VALUES (4,"Vacina da Parvovirus"); -- raccoon like animals -- 3,6,15
INSERT INTO Vacina VALUES (5,"Vacina da Brucellosis"); -- ursos -- 21,50
INSERT INTO Vacina VALUES (6,"Vacina da Diphtheria"); -- apes (primates) -- 3,4,7,40,44
INSERT INTO Vacina VALUES (7,"Vacina do Tetano"); -- all
INSERT INTO Vacina VALUES (8,"Vacina da Clostridial"); -- cow like and sheep (bovid and ovicaprin) girafes and okapis (giraffids) -- 8,14,29,33,34,45
INSERT INTO Vacina VALUES (9,"Vacina da Febre Maligna Catarrhal"); -- deer (cervid) -- 25,8
INSERT INTO Vacina VALUES (10,"Vacina de Clostridium perfringens"); -- camels like (camelids) --34
INSERT INTO Vacina VALUES (11,"Vacina do Virus do Nilo Oeste"); -- horses like (equids) -- 8,20,45
INSERT INTO Vacina VALUES (12,"Vacina da Leptospirosis "); -- rhinos tapirs pig like animals e animais marinhos 10,11,12,43
INSERT INTO Vacina VALUES (13,"Vacina do polyomavirus"); -- birds 24,26,27,31,41,42
INSERT INTO Vacina VALUES (14,"Vacina destinada a Micobacteriose"); -- marine mammals 43
INSERT INTO Vacina VALUES (15,"Vacina destinada a Infeções Respiratórias"); -- all tirando 43,10
```

INSERT INTO Especie VALUES (1,"Lince-ibérico","Lynx pardinus",24); -- Comprimento: 68-82 cm Altura: 40-50 cm PESO: 7-14 kg Floresta

INSERT INTO Especie VALUES (2,"Leão-africano","Panthera leo bleyenberghi",58); -- Comprimento: 1,6-1,9 m Altura: 1,2 m Peso: 120-180 kg||| F :Comprimento: 1,7-2,5 m Altura: 1,2 m Peso: 150-240 kg Savana

INSERT INTO Especie VALUES (3,"Macaco-capuchinho-de-peito-amarelo","Cebus xanthosternos",59); -- Comprimento: 36-42 cm Peso: 3 kg

INSERT INTO Especie VALUES (4,"Macaco-do-japão","Macaca fuscata",21); -- Comprimento: ≤65 cm Peso: ≤18 kg

INSERT INTO Especie VALUES (5,"Pantera-nebulosa","Neofelis nebulosa",37); -- Comprimento: 68-106 cm Peso:

INSERT INTO Especie VALUES (6,"Panda-vermelho","Ailurus fulgens",46); -- Comprimento: 51-73 cm Peso:

INSERT INTO Especie VALUES (7,"Orangotango-de-sumatra","Pongo abelii",41); -- Peso:

INSERT INTO Especie VALUES (8,"Okapi","Okapia johnstoni",30); -- Comprimento: ≤2 m Peso:

INSERT INTO Especie VALUES (9,"Pitão-real","Python regius",48); -- Comprimento: 1,2 – 1,8 m Peso:

INSERT INTO Especie VALUES (10,"Piranha-vermelha","Pygocentrus nattereri",31); -- Comprimento: 33 cm Peso:

INSERT INTO Especie VALUES (11,"Rinoceronte-branco","Ceratotherium simum",20); -- Comprimento: ≤4 m Peso:

INSERT INTO Especie VALUES (12,"Rinoceronte-indiano","Rhinoceros unicornis",60); -- Comprimento: ≤3,5 m Peso:

INSERT INTO Especie VALUES (13,"Tartaruga-aligátor-comum","Macrochelys temminckii",50); -- Comprimento: 35-80 cm Peso:

INSERT INTO Especie VALUES (14,"Girafa-de-angola","Giraffa camelopardalis angolensis",40); -- Comprimento: 14 cm Peso:

INSERT INTO Especie VALUES (15,"Suricata","Suricata suricatta",30); -- Peso:

INSERT INTO Especie VALUES (16,"Tigre-branco","Panthera tigris",35); -- F: Comprimento: 2,1-2,7 m Peso: M:Comprimento: 2,6-3,1 m Peso:

INSERT INTO Especie VALUES (17,"Tigre-da-sibéria","Panthera tigris altaica",45); -- Comprimento: 2,4-3,3 m Peso:

INSERT INTO Especie VALUES (18,"Tartaruga-estrela-indiana","Geochelone elegans",55); -- Comprimento: 15-38 cm Peso:

INSERT INTO Especie VALUES (19,"Tartaruga-espinhosa","Heosemys spinosa",56); -- Comprimento: 17-22 cm Peso:

INSERT INTO Especie VALUES (20,"Zebra-de-grevy","Equus grevyi",22); -- Comprimento: ≤3 m Peso:

INSERT INTO Especie VALUES (21,"Urso-pardo","Ursus arctos",38); -- Comprimento: 1,5-2,8 m Peso:

INSERT INTO Especie VALUES (22,"Aligátor-americano","Alligator mississippiensis",42); -- F: Comprimento: 3 m Peso: | M: Comprimento: 4,5 m Peso:

INSERT INTO Especie VALUES (23,"Anaconda-amarela","Eunectes notaeus",49); -- Comprimento: 3-4 m Peso:

INSERT INTO Especie VALUES (24,"Araçari-verde","Pteroglossus viridis",23); -- Comprimento: 30-39 cm Peso:

INSERT INTO Especie VALUES (25,"Adax","Addax nasomaculatus",58); -- Comprimento: <1,30 m Peso:

INSERT INTO Especie VALUES (26,"Arara-jacinta","Anodorhynchus hyacinthinus",47); -- Comprimento: 100 cm Peso:

INSERT INTO Especie VALUES (27,"Arara-escarlate","Ara macao",25); -- Comprimento: 84-89 cm Peso:

INSERT INTO Especie VALUES (28,"Axolote","Ambystoma mexicanum",34); -- Comprimento: <30cm Peso:

INSERT INTO Especie VALUES (29,"Bisonte-americano","Bison bison bison",22); -- Comprimento: <3,5 m Peso:

INSERT INTO Especie VALUES (30,"Boa-da-jamaica","Epicrates subflavus",51); -- Comprimento: 2 - 2,5 m

INSERT INTO Especie VALUES (31,"Calau-da-papuásia","Aceros plicatus",59); -- Comprimento: 65-85 cm

INSERT INTO Especie VALUES (32,"Boa-de-madagáscar","Sanzinia madagascariensis",36); -- Comprimento: até 2,5 m

INSERT INTO Especie VALUES (33,"Búfalo-africano","Syncerus caffer caffer",26); -- Comprimento: 2,4-3,4 m

INSERT INTO Especie VALUES (34,"Camelo","Camelus ferus",47); -- Comprimento: 250-300 cm

INSERT INTO Especie VALUES (35,"Canguru-vermelho","Macropus rufus / Osphranter rufus",35); --

INSERT INTO Especie VALUES (36,"Dragão-de-komodo","Varanus komodoensis",27); -- Comprimento: ≤3 m

INSERT INTO Especie VALUES (37,"Elefante-africano","Loxodonta africana",42); -- Comprimento: 6-7,5 m

INSERT INTO Especie VALUES (38,"Crocodilo-do-nilo","Crocodylus niloticus",38); -- Comprimento: 5,5 m

INSERT INTO Especie VALUES (39,"Chimpanzé","Pan troglodytes",46); --

INSERT INTO Especie VALUES (40,"Chita","Acinonyx jubatus",27); -- Comprimento: 1,1-1,3 m

INSERT INTO Especie VALUES (41,"Ema","Dromaius novaehollandiae",52); --

INSERT INTO Especie VALUES (42,"Flamingo-rubro","Phoenicopterus ruber",35); --

INSERT INTO Especie VALUES (43,"Golfinho-roaz","Tursiops truncatus",42); -- Comprimento: ≤3,9 m

```
INSERT INTO Especie VALUES (44,"Gorila-ocidental-das-terras-baixas","Gorilla gorilla gorilla",51); -- F: Comprimento: 1,5 m | M: Comprimento: 1,7 m
INSERT INTO Especie VALUES (45,"Hipopótamo","Hippopotamus amphibius",48); -- Comprimento: ≤5,5 m
INSERT INTO Especie VALUES (46,"Iguana-rinoceronte","Cyclura cornuta",35); -- Comprimento: 1,2m
INSERT INTO Especie VALUES (47,"Iguana-verde","Iguana iguana",58); -- Comprimento: 1,5-2 m
INSERT INTO Especie VALUES (48,"Jaguar","Panthera onca onca",38); -- Comprimento: 1,15-1,70 cm
INSERT INTO Especie VALUES (49,"Pinguim-do-Cabo","Spheniscus demersus",25); -- Comprimento: 55-76 cm
INSERT INTO Especie VALUES (50,"Urso-polar","Ursus maritimus",43);
```

III. Anexo 3 – Programa em Java para geração do *Script* remanescente do Povoamento Inicial

```
import java.io.*;
import java.time.LocalDateTime;
import java.util.*;

public class main {
   static String NOMETABELA="Bilhete";
   static String VACANIVET="";
   static Map<Integer,List<Quad<Integer,Integer,Integer,Integer>>> ESPECIEVACINA = new HashMap<>();
   static Map<Integer,List<Animal>> arvores = new HashMap<>();
   static Map<Integer,Triple<Double,Double,Double>> species = new HashMap<>();
   static Map<String, int[]> biomeDist = new HashMap<>();
   static Map <Integer,int[]> lista = new HashMap<>();
   static String bancos[] = {"0007", "0010", "0018", "0019", "0022", "0023", "0025", "0032", "0033", "0034", "0035", "0235",
"0036", "0038", "0043", "0045", "0046", "0059", "0061", "0065", "0079", "0193", "0269", "0781", "5180"};
   static String petnames[] = {"Sonya", "Dove", "Fang", "Fuzz", "Chloe", "Bonkers", "Crook", "Donut", "Spectral", "Teapot",
"Hugo", "Shredder", "Tinkerbell", "Sugar", "Galileo", "Gumby", "Hunter", "Chakra", "Zion", "Sally"};
   static String first_name[] = {"Águeda", "Amélia", "Ângela", "Alessandro", "Alessandra", "Alexandre", "Aline", "Antônio",
"Breno", "Bruna", "Carlos", "Carla", "Célia", "Cecília", "César", "Danilo", "Dalila", "Deneval", "Eduardo", "Eduarda", "Esther",
"Elísio", "Fábio", "Fabrício", "Fabrícia", "Félix", "Felícia", "Feliciano", "Frederico", "Fabiano", "Gustavo", "Guilherme", "Gúbio",
"Heitor", "Hélio", "Hugo", "Isabel", "Isabela", "Ígor", "João", "Joana", "Júlio César", "Júlio", "Júlia", "Janaína", "Karla", "Kléber",
"Lucas", "Lorena", "Lorraine", "Larissa", "Ladislau", "Marcos", "Meire", "Marcelo", "Marcela", "Margarida", "Mércia", "Márcia",
"Marli", "Morgana", "Maria", "Norberto", "Natália", "Nataniel", "Núbia", "Ofélia", "Paulo", "Paula", "Pablo", "Pedro", "Raul",
"Rafael", "Rafaela", "Ricardo", "Roberto", "Roberta", "Sílvia", "Sílvia", "Silas", "Simão", "Suélen", "Sara", "Salvador", "Sirineu",
"Talita", "Tertuliano", "Vicente", "Víctor", "Vitória", "Yango", "Yago", "Yuri", "Washington", "Warley"};
   static String last_name[] = {"Araújo", "D'cruze", "Estéves", "Silva", "Souza", "Carvalho", "Santos", "Reis", "Xavier", "Franco",
"Braga", "Macedo", "Batista", "Barros", "Moraes", "Costa", "Pereira", "Carvalho", "Melo", "Lemos", "Saraiva", "Nogueira",
"Oliveira", "Martins", "Moreira", "Albuquerque"};
   static String emails[] = {"@hotmail.com", "@gmail.com", "@yahoo.com", "@outlook.com"};
   static String nums[] = {"91", "92", "93", "96", "259", "253", "256"};

   static void inicia() {
      species.put(1, new Triple<Double, Double, Double>(0.5, 0.82, 14.0));
      species.put(2, new Triple<Double, Double, Double>(1.2, 2.5, 240.0));
      species.put(3, new Triple<Double, Double, Double>(1.0, 0.42, 3.0));
      species.put(4, new Triple<Double, Double, Double>(1.0, 0.65, 18.0));
      species.put(5, new Triple<Double, Double, Double>(0.65, 1.06, 23.0));
      species.put(6, new Triple<Double, Double, Double>(0.73, 0.73, 6.0));
      species.put(7, new Triple<Double, Double, Double>(1.0, 0.4, 90.0));
      species.put(8, new Triple<Double, Double, Double>(2.0, 1.8, 300.0));
      species.put(9, new Triple<Double, Double, Double>(0.2, 2.0, 15.0));
      species.put(10, new Triple<Double, Double, Double>(0.1, 0.33, 3.8));
      species.put(11, new Triple<Double, Double, Double>(2.0, 4.0, 3500.0));
      species.put(12, new Triple<Double, Double, Double>(1.9, 3.5, 2000.0));
      species.put(13, new Triple<Double, Double, Double>(0.2, 0.8, 113.0));
      species.put(14, new Triple<Double, Double, Double>(5.0, 4.7, 1900.0));
      species.put(15, new Triple<Double, Double, Double>(0.35, 0.1, 0.9));
      species.put(16, new Triple<Double, Double, Double>(1.1, 3.1, 260.0));
      species.put(17, new Triple<Double, Double, Double>(1.1, 3.3, 300.0));
      species.put(18, new Triple<Double, Double, Double>(0.2, 0.38, 6.0));
      species.put(19, new Triple<Double, Double, Double>(0.3, 0.22, 2.0));
      species.put(20, new Triple<Double, Double, Double>(1.6, 3.0, 450.0));
      species.put(21, new Triple<Double, Double, Double>(1.5, 2.8, 600.0));
      species.put(22, new Triple<Double, Double, Double>(0.3, 4.5, 450.0));
      species.put(23, new Triple<Double, Double, Double>(0.15, 4.0, 30.0));
      species.put(24, new Triple<Double, Double, Double>(0.25, 0.35, 0.162));
      species.put(25, new Triple<Double, Double, Double>(1.15, 1.3, 90.0));
      species.put(26, new Triple<Double, Double, Double>(0.25, 1.0, 1.7));
      species.put(27, new Triple<Double, Double, Double>(0.25, 0.89, 1.4));
      species.put(28, new Triple<Double, Double, Double>(0.2, 0.3, 0.18));
      species.put(29, new Triple<Double, Double, Double>(1.95, 3.5, 1000.0));
      species.put(30, new Triple<Double, Double, Double>(0.2, 2.5, 30.0));
      species.put(31, new Triple<Double, Double, Double>(0.25, 0.85, 2.0));
      species.put(32, new Triple<Double, Double, Double>(0.2, 2.5, 6.0));
      species.put(33, new Triple<Double, Double, Double>(1.8, 3.4, 900.0));
```

```
        species.put(34, new Triple<Double, Double, Double>(1.8, 3.0, 600.0));
        species.put(35, new Triple<Double, Double, Double>(1.5, 2.0, 92.0));
        species.put(36, new Triple<Double, Double, Double>(0.4, 3.0, 150.0));
        species.put(37, new Triple<Double, Double, Double>(3.7, 7.5, 7500.0));
        species.put(38, new Triple<Double, Double, Double>(0.3, 5.5, 1000.0));
        species.put(39, new Triple<Double, Double, Double>(0.95, 0.4, 60.0));
        species.put(40, new Triple<Double, Double, Double>(0.94, 1.3, 43.0));
        species.put(41, new Triple<Double, Double, Double>(1.9, 1.0, 55.0));
        species.put(42, new Triple<Double, Double, Double>(1.45, 0.3, 2.8));
        species.put(43, new Triple<Double, Double, Double>(0.6, 3.9, 500.0));
        species.put(44, new Triple<Double, Double, Double>(2.0, 1.7, 180.0));
        species.put(45, new Triple<Double, Double, Double>(1.6, 5.5, 4500.0));
        species.put(46, new Triple<Double, Double, Double>(0.2, .2, 9.0));
        species.put(47, new Triple<Double, Double, Double>(0.2, 2.0, 5.0));
        species.put(48, new Triple<Double, Double, Double>(1.0, 1.7, 120.0));
        species.put(49, new Triple<Double, Double, Double>(0.7, 0.3, 3.0));
        species.put(50, new Triple<Double, Double, Double>(1.3, 3.0, 450.0));
        biomeDist.put("Savana", new int[]{2, 41, 40, 37, 33, 20, 18, 15, 14, 12, 11});
        biomeDist.put("Floresta", new int[]{6, 35, 1});
        biomeDist.put("Floresta Tropical", new int[]{44, 39, 3, 4, 7, 47, 46, 32, 30, 19, 24, 26, 27, 31});
        biomeDist.put("Pântano", new int[]{28, 23, 22, 13, 9});
        biomeDist.put("Planície Temperada", new int[]{29, 16, 8});
        biomeDist.put("Pólo", new int[]{49, 50});
        biomeDist.put("Oceano", new int[]{43, 10});
        biomeDist.put("Rio", new int[]{45, 42, 38});
        biomeDist.put("Montanha", new int[]{48, 25, 21, 17, 5});
        biomeDist.put("Deserto", new int[]{36, 34});
        lista.put(1, new int[]{21, 50, 36});
        lista.put(2, new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50});
        lista.put(3, new int[]{1, 2, 5, 16, 17, 40, 48});
        lista.put(4, new int[]{3, 6, 15});
        lista.put(5, new int[]{21, 50});
        lista.put(6, new int[]{3, 4, 7, 40, 44});
        lista.put(7, new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50});
        lista.put(8, new int[]{8, 14, 29, 33, 34, 45});
        lista.put(9, new int[]{25, 8});
        lista.put(10, new int[]{34});
        lista.put(11, new int[]{8, 20, 45});
        lista.put(12, new int[]{10, 11, 12, 43});
        lista.put(13, new int[]{24, 26, 27, 31, 41, 42});
        lista.put(14, new int[]{43});
        lista.put(15, new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50});
    }


    public static class Animal {
        public int ano;
        public int idAn;
        public boolean sexo;
        public boolean mae;
        public boolean pai;

        public Animal(int first, int animal,boolean second) {
            this.ano = first;
            this.sexo = second;
            this.idAn=animal;
            this.pai = this.mae = false;
        }
    }
```

```
public static class Triple<T, U, V> {

    private final T first;
    private final U second;
    private final V third;

    public Triple(T first, U second, V third) {
        this.first = first;
        this.second = second;
        this.third = third;
    }

    public T getFirst() { return first; }
    public U getSecond() { return second; }
    public V getThird() { return third; }
}

public static class Quad<T, U, V, W> {

    public final T first;
    public final U second;
    public final V third;
    public final W forth;

    public Quad(T first, U second, V third, W forth) {
        this.first = first;
        this.second = second;
        this.third = third;
        this.forth = forth;
    }
}

public static String getRelacoes() {
    String relac="";
    for (List<Animal> a : arvores.values()) {
        for (Animal aa : a) {
            for (Animal aaa : a) {
                if (aa.idAn!=aaa.idAn) {
                    if (aaa.ano>aa.ano+2) {
                        if (aa.sexo && !aaa.pai) {
                            aaa.pai=true;
                            relac = relac.concat("INSERT INTO Animal_has_Animal VALUES ("+aa.idAn+","+aaa.idAn+");\n");
                        }
                        else if (!aa.sexo && !aaa.mae) {
                            aaa.mae=true;
                            relac = relac.concat("INSERT INTO Animal_has_Animal VALUES ("+aa.idAn+","+aaa.idAn+");\n");
                        }
                    }
                }
            }
        }
    }
    return relac;
}


public static String getRandomTicket(Random rand) {
    int ano = rand.nextInt(30)+1990;
    int mês = rand.nextInt(12)+1;
    int dia = rand.nextInt(27+(mês==2 ? 1 : 3))+1;
    String nascimento = "'"+ano+"-"+mês+"-"+dia+"'";

    return nascimento+","+(rand.nextInt(6)+1);
}
```

```
  public static String getRandomNameAnimal(Random rand) {
    return petnames[rand.nextInt(petnames.length)];
  }
  public static String getRandomAnimal(Random rand,int idEspecie,int padrinho,int animal) {
     String ANIMAL ="";

     Triple<Double,Double,Double> info = species.get(idEspecie);
     String gender = rand.nextBoolean() ? "\"Masculino\"" : "\"Feminino\"";
     int vivo = rand.nextInt(100)>79 ? 0 : 1;
     int ano = rand.nextInt(30)+1990;
     List<Animal> l;
     if (arvores.containsKey(idEspecie)) {
        l = arvores.get(idEspecie);
     }
     else {
        l = new ArrayList<>();
     }
     l.add(new Animal(ano,animal,gender.equals("\"Masculino\"")));
     arvores.put(idEspecie,l);
     int mês = rand.nextInt(12)+1;
     int dia = rand.nextInt(27+(mês==2 ? 1 : 3))+1;
     String nascimento = "'"+ano+"-"+mês+"-"+dia+"'";

     List<Quad<Integer,Integer,Integer,Integer>> ll = ESPECIEVACINA.get(idEspecie);

     for (Quad<Integer,Integer,Integer,Integer> q : ll) {
        int vet = rand.nextInt(25)+1;
        int anoV=ano;
        int mesV = mês+q.second;
        for (int contador=0;contador<q.forth;contador++) {
           while (mesV>12) {
              anoV++;
              mesV-=12;
           }
           if (anoV>2020 | mesV>11) break;
           String data = "'"+ anoV +"-"+ mesV +"-1'";
           VACANIVET = VACANIVET.concat("INSERT INTO Animal_has_Vacina_has_Veterinario VALUES
("+animal+","+q.first+","+vet +"," + data +");\n");
           mesV+=q.third;
        }
     }

     double altura=info.getFirst()*(0.95+rand.nextDouble()*0.1);
     double comprimento=info.getSecond()*(0.9+rand.nextDouble()*0.2);
     double peso=info.getThird()*(0.8+rand.nextDouble()*0.4)*(gender.equals("\"Feminino\"") ? 0.90 : 1);

     String cS=String.valueOf(comprimento).substring(0,4);
     String aS=String.valueOf(altura).substring(0,4);
     String pS;
     if (peso>=1000)
        pS=String.valueOf(peso).substring(0,8);
     else if (peso>=100)
        pS=String.valueOf(peso).substring(0,7);
     else if (peso>=10)
        pS=String.valueOf(peso).substring(0,6);
     else
        pS=String.valueOf(peso).substring(0,5);
     String nameAnimal="";
     if (padrinho>0) {
        nameAnimal="\""+getRandomNameAnimal(rand).concat("\",");
     }
     ANIMAL = ANIMAL.concat(gender+","+nameAnimal+nascimento+","+aS+","+cS+","+pS+","+vivo+",");
     if (padrinho>0) {
        ANIMAL = ANIMAL.concat(padrinho + ",");
     }
```

```
        return ANIMAL;
    }

    public static Map.Entry<String,String> getRecintos(Random rand) {
        String RECINTO = "";
        int nanimas=1;
        int pad=1;
        String ANIMAL = "";
        int i=1,k=0,j=0,zona;
        for (Map.Entry<String,int[]> e : biomeDist.entrySet()) {
            for (Integer f : e.getValue()) {
                if (f==49 || f==50) zona=7;
                else if (f==43 || f== 10) zona=9;
                else if (f==4 || f==12 || f==18 || f==31 || f==7) zona=8;
                else if (f==6 || f==35) zona=6;
                else if (f==9 || f== 13 || f==19 || f==23|| f==30 || f==32 || f==46 || f==47) zona=4;
                else if (f==1 || f==5 || f==16 || f==17 || f==40 || f==48) zona=3;
                else if (f==3 || f==39 || f==44) zona=2;
                else if (f==24 || f==26 || f== 27 || f==42) zona=1;
                else zona=5;
                String size = String.valueOf(rand.nextFloat()*4 ).substring(0,3);
                RECINTO = RECINTO.concat("INSERT INTO Recinto VALUES
("+i+","+j*10+","+k*10+","+size+",\""+e.getKey()+"\","+zona+"); -- "+f+"\n");
                j++;
                if (j==5) {
                    k++;
                    j=0;
                }
                int n=rand.nextInt(10)+1;
                for (int l=0;l<n;l++) {
                    int idPadrinho = 0;
                    if (rand.nextInt(100) > 75) {
                        if (pad==26) pad=1;
                        idPadrinho = pad;
                        ANIMAL=ANIMAL.concat("INSERT INTO Animal VALUES (");
                        pad++;
                    }
                    else {
                        ANIMAL=ANIMAL.concat("INSERT INTO Animal
(idAnimal,genero,data_nascimento,altura,comprimento,peso,vivo,Recinto_ID,Especie_idEspecie) VALUES (");


                    }
                    String animal=getRandomAnimal(rand,f,idPadrinho,nanimas);
                    ANIMAL=ANIMAL.concat(String.valueOf(nanimas).concat(","+animal+i+","+f+");\n"));
                    nanimas++;
                }
                i++;
            }
        }
        return new AbstractMap.SimpleEntry<>(RECINTO, ANIMAL);
    }

    public static String getRandomName(Random rand) {
        String nome = first_name[rand.nextInt(first_name.length)]+" "+last_name[rand.nextInt(last_name.length)];
        return nome;
    }

    public static String getRandomNIF(Random rand) {
        String NIF ="";
        for (int i=0;i<9;i++) {
            NIF = NIF.concat(String.valueOf(rand.nextInt(10)));
        }
        return NIF;
    }
```

```
public static String getRandomNIB(Random rand) {
    String NIB="";
    int i=rand.nextInt(25);
    NIB = NIB.concat(bancos[i]);
    for (int j=0;j<16;j++) {
        int k=rand.nextInt(10);
        NIB = NIB.concat(String.valueOf(k));
    }
    return NIB;
}

public static String getRandomCC(Random rand) {
    String CC="";
    for (int j=0;j<8;j++) {
        CC=CC.concat(String.valueOf(rand.nextInt(10)));
    }
    CC = CC.concat(" ");
    CC = CC.concat(String.valueOf(rand.nextInt(9)+1));
    CC = CC.concat(" ");

    String alphabet = "ZYX";
    for (int i = 0; i < 2; i++) {
        CC = CC.concat(String.valueOf(alphabet.charAt(rand.nextInt(alphabet.length()))));
    }

    CC = CC.concat(String.valueOf(rand.nextInt(2)+1));

    return CC;
}

public static String getRandomEMAIL(Random rand,String name) {
   name = name.toLowerCase().replace(" ","");
    return "\"" + name + rand.nextInt(1000) +emails[rand.nextInt(emails.length)]+"\"";
}

public static String getRandomNum(Random rand) {
    String num;
    for (num = nums[rand.nextInt(nums.length)];num.length()<10;) {
        num=num.concat(String.valueOf(rand.nextInt(10)));
    }
    return "\""+num+"\"";
}

public static String getRandomCont(Random rand,String name) {
    int n = rand.nextInt(100);
    if (n>60) {
        return getRandomEMAIL(rand,name);
    }
    else {
        return getRandomNum(rand);
    }
}


public static String getRandomVacinaEEspecie(Random rand) {
    String vEe="";
    Boolean b1,b2;
    String dd1,dd2;
    for (Map.Entry<Integer,int[]> e : lista.entrySet()) {
        for (int especie : e.getValue()) {
            int limite = rand.nextInt(24);
            int dNN=rand.nextInt(5)+1;
            int espaco = (rand.nextInt(19) + 6) * (dNN==1? 0 : 1);
            String dQ = String.valueOf(rand.nextInt(15));
            String idd;
            if (dNN==1)
```

```
                idd = "(Especie_idEspecie,Vacina_idVacina,limite_temporal,doses_necessarias,dosagem) ";
            else idd="";
            List<Quad<Integer,Integer,Integer,Integer>> lis;
            if (ESPECIEVACINA.containsKey(especie)) {
                lis = ESPECIEVACINA.get(especie);
            }
            else {
                lis = new ArrayList<>();
            }
            lis.add(new Quad<>(e.getKey(),limite,espaco,dNN));
            ESPECIEVACINA.put(especie,lis);
            String space="";
            if (espaco!=0) {
                space = espaco+ ",";
            }
            vEe = vEe.concat("INSERT INTO Especie_has_Vacina "+ idd+ "VALUES
("+especie+","+e.getKey()+","+limite+","+space+dNN+","+dQ+");\n");
        }
    }

    return vEe;
}

public static void main(String[] args) throws IOException {
    int id;
    String padrinhos="";
    String pc="";
    Random rand = new Random();
    rand.setSeed(LocalDateTime.now().hashCode()*rand.nextLong());

    File myObj = new File("PovoamentoBD.sql");
    myObj.createNewFile();
    FileWriter myWriter = new FileWriter("PovoamentoBD.sql",true);
    BufferedWriter bw = new BufferedWriter(myWriter);
    PrintWriter out = new PrintWriter(bw);

    inicia();

    for (id=1;id<26;id++) {
        String name = getRandomName(rand);
        padrinhos = padrinhos.concat("INSERT INTO Padrinho VALUES
("+id+",\""+name+"\",\""+getRandomNIB(rand)+"\",\""+getRandomCC(rand)+"\",\""+getRandomNIF(rand)+"\");\n");
        int random = rand.nextInt(3)+2;
        for (int y=1;y<random;y++) {
            pc = pc.concat("INSERT INTO Contacto_Padrinho VALUES ("+id+","+getRandomCont(rand,name)+");\n");
        }
    }

    padrinhos = padrinhos.concat("\n");
    pc = pc.concat("\n");

    for (id=1;id<26;id++) {
        String name = getRandomName(rand);
        padrinhos = padrinhos.concat("INSERT INTO Veterinario VALUES
("+id+",\""+name+"\");\n");//",\""+getRandomNIB(rand)+"\",\""+getRandomCC(rand)+"\",\""+getRandomNIF(rand)+"\");\n");
        int random = rand.nextInt(3)+2;
        for (int y=1;y<random;y++) {
            pc = pc.concat("INSERT INTO Contacto_Veterinario VALUES ("+id+","+getRandomCont(rand,name)+");\n");
        }
    }

    out.print(padrinhos+"\n");
    out.print(pc+"\n");

    out.print(getRandomVacinaEEspecie(rand)+"\n");
```

```
        Map.Entry<String,String> p = getRecintos(rand);
        out.print(p.getKey()+"\n"+p.getValue()+"\n");
        out.print(getRelacoes()+"\n");
        out.print(VACANIVET);

        for  (id =1 ;id<10001; id++) {
            out.print("INSERT INTO Bilhete VALUES ("+id+","+getRandomTicket(rand)+");\n");
        }

        out.close();

    }
}
```

IV. Anexo 4 - *Script* de Implementação de Interrogações e Funcionalidades para o Funcionário

```
-- ------------Requisitos de Exploração------------ --


-- ----------------------QUERY 1-----------------------
-- Consultar que animais um padrinho apadrinhou     --
-- ------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE animaisApadrinhados
   (IN idPadrinho INT)
BEGIN
   SELECT idAnimal AS IDAnimal, nome AS Nome
      FROM Animal
      WHERE Padrinho_idPadrinho = idPadrinho;
END $$

-- -------------------------QUERY 2-----------------------------
-- Consultar qual o valor mensal total que cada padrinho deve pagar   --
-- ------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE valorMensalAPagar
   ()
BEGIN
   SELECT Padrinho_idPadrinho AS IDPadrinho, SUM(E.valor_de_apadrinhamento) AS
ValorAPagar
      FROM Animal AS A INNER JOIN Especie AS E
         ON A.Especie_idEspecie=E.idEspecie
      WHERE Padrinho_idPadrinho IS NOT NULL
      GROUP BY Padrinho_idPadrinho;
END $$

-- --------------------QUERY 3--------------------------
-- Consultar as crias de um animal existentes no zoo   --
```

```
-- -----------------------------------------------------

DELIMITER $$
CREATE PROCEDURE criasDeAnimal
   (IN idAnimal INT)
BEGIN
   SELECT Animal_idAnimalFilho AS IDCria
      FROM Animal_has_Animal
      WHERE Animal_idAnimalProgenitor=idAnimal;
END $$
-- ----------------------QUERY 4-------------------------
-- Determinar todos os descendentes de um animal no zoo --
-- -----------------------------------------------------

DELIMITER $$
CREATE PROCEDURE descendentesAnimal
   (IN idAnimal INT)
BEGIN
   WITH RECURSIVE arvoreDescendente AS (
   SELECT Animal_idAnimalProgenitor AS IDProgenitor,Animal_idAnimalFilho AS IDCria, 1 AS
Profundidade_Relativa
   FROM Animal_has_Animal
   WHERE Animal_idAnimalProgenitor = idAnimal

   UNION ALL

   SELECT pais.Animal_idAnimalProgenitor, pais.Animal_idAnimalFilho,
aD.Profundidade_Relativa + 1
   FROM Animal_has_Animal pais, arvoreDescendente aD
   WHERE pais.Animal_idAnimalProgenitor = aD.IDCria
   )
   SELECT * FROM arvoreDescendente;
END $$

-- -------------------------QUERY 5---------------------------
-- Consultar os progenitores de um animal existentes no zoo --
-- -------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE progenitoresDeAnimal
```

```
   (IN idAnimal INT)
BEGIN
SELECT Animal_idAnimalProgenitor AS IDCria
   FROM Animal_has_Animal
   WHERE Animal_idAnimalFilho=idAnimal;
END $$

-- ----------------------------QUERY 6-------------------------
-- Mostrar todos os ascendentes de um animal existentes no zoo
-- -------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE ascendentesAnimal
   (IN idAnimal INT)
BEGIN
   WITH RECURSIVE arvoreAscendente AS (
   SELECT Animal_idAnimalFilho AS IDCria, Animal_idAnimalProgenitor AS IDProgenitor, -1 AS
Profundidade_Relativa
   FROM Animal_has_Animal
   WHERE Animal_idAnimalFilho = idAnimal

   UNION ALL

   SELECT pais.Animal_idAnimalFilho, pais.Animal_idAnimalProgenitor,
aA.Profundidade_Relativa - 1
   FROM Animal_has_Animal pais, arvoreAscendente aA
   WHERE pais.Animal_idAnimalFilho = aA.IDProgenitor
   )
   SELECT * FROM arvoreAscendente;
END $$

-- ----------------------QUERY 7-----------------------------
-- Conhecer todas as espécies que um tipo de bilhete dá acesso
-- -------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE especiesPorTipoBilhete
   (IN idTipo INT)
BEGIN
```

```
SELECT E.nome_comum AS NomeComum, E.nome_cientifico AS NomeCientifico, Z.nome AS
NomeZona
   FROM Tipo AS T
   INNER JOIN Tipo_has_Zona AS TZ
      ON T.idTipo = TZ.Tipo_idTipo
      INNER JOIN Zona AS Z
         ON Z.idZona=TZ.Zona_idZona
         INNER JOIN Recinto AS R
            ON R.Zona_idZona=Z.idZona
            INNER JOIN Animal AS A
               ON A.Recinto_ID = R.ID
               INNER JOIN Especie AS E
                  ON E.idEspecie = A.Especie_idEspecie
   WHERE T.idTipo=idTipo
   GROUP BY E.idEspecie;
END $$

-- ------------------------QUERY 8-----------------------------
-- Consultar quantos animais, existem em cada bioma do jardim zoológico --
-- ------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE nAnimaisPorBioma
   ()
BEGIN
SELECT bioma AS Bioma, SUM(A.vivo) AS NAnimaisVivos ,SUM(CASE WHEN A.vivo=0 THEN
1 ELSE 0 END)AS NAnimaisMortos
   FROM Recinto as R INNER JOIN Animal AS A
   ON R.ID = A.Recinto_ID
   GROUP BY bioma;
END $$

-- --------------------QUERY 9-----------------------
-- Calcular o TOP 3 tipo de bilhetes mais comprados --
-- ------------------------------------------------

DELIMITER $$
CREATE PROCEDURE top3TiposBilheteMaisComprados
   ()
BEGIN
```

```
SELECT T.nome AS TipoBilhete, COUNT(B.tipo_idTipo) AS NBilhetesVendidos
   FROM Bilhete as B
      INNER JOIN Tipo AS T
      ON B.Tipo_idTipo = T.idTipo
   GROUP BY B.tipo_idTipo
   ORDER BY COUNT(B.tipo_idTipo) DESC
   LIMIT 3;
END $$

-- ---------------------QUERY 10----------------------------
-- Conhecer quantos bilhetes de cada tipo foram vendidos --
-- ----------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE nBilhetesTipoSold
   ()
BEGIN
SELECT idTipo AS Tipo, COUNT(*) AS Número FROM Bilhete b,Tipo t
   WHERE b.Tipo_idTipo = t.idTipo
   GROUP BY idTipo;
END $$

-- -------QUERY 11-----------
-- Calcula faturação total --
-- --------------------------

DELIMITER $$
CREATE PROCEDURE faturacaoGlobal
   ()
BEGIN
SELECT SUM(preco) AS Faturacao_Global
   FROM (SELECT * FROM Bilhete b,Tipo t
   WHERE b.Tipo_idTipo = t.idTipo) as aux;
END $$

-- ---------------QUERY 12--------------------
-- Calcula faturação num intervalo de tempo --
-- ------------------------------------------

DELIMITER $$
```

```
CREATE PROCEDURE faturacaoIntervalo
   (IN dataInicio DATE, dataFim Date)
BEGIN
SELECT SUM(preco) AS Faturacao_Intervalo
   FROM (SELECT * FROM Bilhete b, Tipo t
      WHERE b.Tipo_idTipo = t.idTipo
         AND momento_aquisicao BETWEEN dataInicio AND dataFim) AS aux;
END $$

-- ------------QUERY 13---------------------
-- Saber o crescimento de visitas anual  --
-- ----------------------------------------

DELIMITER $$
CREATE PROCEDURE aumentoVisitasPorAno
   ()
BEGIN
SELECT Ano, NumeroBilhetes, ((NumeroBilhetes-
NumeroBilhetesAnterior)/NumeroBilhetesAnterior)*100 AS CrescimentoPercentagem
   FROM vwNbilhetesAnos;
END $$

-- ---------------------------QUERY 14-------------------------
-- Saber que veterinarios administraram qual vacina a um animal
-- ------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE veterinarioVacinaPorAnimal
   (IN idAnimal INT)
BEGIN
SELECT v.nome AS Nome, va.doenca AS Nome_Vacina
   FROM Veterinario v, Animal_has_Vacina_has_Veterinario vc, Animal a, Vacina va
      WHERE v.idVeterinario = vc.Veterinario_idVeterinario
         AND vc.Animal_idAnimal = a.idAnimal
         AND va.idVacina = vc.Vacina_idVacina
         AND a.idAnimal = idAnimal;
END $$

-- ------------------------QUERY 15----------------------------
```

```
-- Calcula quais vacinas já foram administradas a um animal e o seu momento de administração
--
-- -------------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE vacinaDataPorAnimal
   (IN idAnimal INT)
BEGIN
SELECT va.doenca AS Nome_Vacina, vc.data_administracao AS Data
   FROM Animal_has_Vacina_has_Veterinario vc, Animal a, Vacina va
      WHERE vc.Animal_idAnimal = a.idAnimal
         AND va.idVacina = vc.Vacina_idVacina
         AND a.idAnimal = idAnimal;
END $$

-- ---------------------QUERY 16---------------------------------
-- Consulta quais vacinas, incluindo doses que se devem repetir, um animal ainda deve tomar --
-- -------------------------------------------------------------

DELIMITER $$
DROP PROCEDURE IF EXISTS vacinasPorDarAnimal;
CREATE PROCEDURE vacinasPorDarAnimal
   (IN idAnimal INT)
BEGIN
SELECT v.doenca as Vacina, calculaDosesRestantes(a.data_nascimento,ev.limite_temporal,ev.intervalo_temporal,ev.doses_necessarias) as Doses_Em_Falta FROM Animal a, Especie e, Especie_has_Vacina ev, Vacina v
   WHERE a.Especie_idEspecie = e.idEspecie
      AND e.idEspecie = ev.Especie_idEspecie
      AND ev.Vacina_idVacina = v.idVacina
      AND a.idAnimal = idAnimal
      AND a.vivo = 1
      AND (TIMESTAMPDIFF(MONTH,a.data_nascimento,CURDATE())) <= (ev.limite_temporal + (ev.intervalo_temporal * (ev.doses_necessarias-1)));
END $$

-- -----------------QUERY 17------------------
-- Apresenta os contactos de um veterinário --
-- -------------------------------------------
```

```
DELIMITER $$
CREATE PROCEDURE contactoVeterinario
   (IN idVeterinario INT)
BEGIN
SELECT * FROM Veterinario v, Contacto_Veterinario cv
   WHERE v.idVeterinario = cv.Veterinario_idVeterinario
      AND v.idVeterinario = idVeterinario;
END $$

-- ----------------------QUERY 18---------------------------
-- Calcula a média de pesos de animais de uma dada espécie --
-- ---------------------------------------------------------

DELIMITER $$
CREATE PROCEDURE averageWeightSpecies
   (IN idEspecie INT)
BEGIN
SELECT e.nome_comum, e.nome_cientifico, AVG(a.peso) AS MediaPeso FROM Especie e,
Animal a
   WHERE e.idEspecie = a.Especie_idEspecie
      AND e.idEspecie = idEspecie
         GROUP BY e.idEspecie;
END $$
```

V. Anexo 5 – *Script* de Criação da Função definida que auxilia na resposta a parte das Interrogações

```
-- ------------------------------------------------------------
-- --------------------FUNÇÕES-----------------------------
-- ------------------------------------------------------------


-- Função que calcula quantas doses faltam administrar dada uma data de nascimento
DELIMITER $$
CREATE FUNCTION calculaDosesRestantes(dataNascimento DATE,limiteTemporal INT, intervaloTemporal INT, dosesNecessarias INT) RETURNS INT
BEGIN
   RETURN CEIL(((limiteTemporal + (intervaloTemporal * (dosesNecessarias-1))) - (TIMESTAMPDIFF(MONTH,dataNascimento,CURDATE())))/intervaloTemporal);
END $$
```

VI. Anexo 6 - *Script* de Criação das Vistas que respondem a parte das Interrogações

```
-- ------------------------------------------------------------
-- ------------------------VIEWS-------------------------------
-- ------------------------------------------------------------


DROP VIEW IF EXISTS vwNBilhetesAno;
DROP VIEW IF EXISTS vwNbilhetesAnos;

-- Numero de bilhetes por ano
CREATE VIEW vwNBilhetesAno AS
   SELECT YEAR(B.momento_aquisicao) AS Ano,COUNT(B.idBilhete) AS NumeroBilhetes
      FROM Bilhete as B
      GROUP BY YEAR(B.momento_aquisicao)
      ORDER BY YEAR(B.momento_aquisicao) DESC;

-- Usa view anterior para ter uma tabela com o ano, o n de bilhetes desse ano e o n de bilhetes do ano anterior
CREATE VIEW vwNbilhetesAnos AS
   SELECT Atual.Ano AS Ano, Atual.NumeroBilhetes AS NumeroBilhetes, Anterior.NumeroBilhetes AS NumeroBilhetesAnterior
      FROM (SELECT * FROM vwNBilhetesAno) AS Atual
      INNER JOIN (SELECT * FROM vwNBilhetesAno) AS Anterior
      ON Atual.Ano= Anterior.Ano+1;

-- Apresenta o número de animais vivos presentes em cada zona
CREATE VIEW vwNAnimaisZona AS
SELECT Z.nome AS NomeDeZona, COUNT(Z.idZona) AS NúmeroAnimais
   FROM Zona AS Z
      INNER JOIN Recinto AS R
      ON Z.idZona = R.Zona_idZona
      INNER JOIN Animal AS A
      ON A.Recinto_ID = R.ID
   WHERE A.vivo=1
   GROUP BY Z.idZona;
```