U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BASE DE DADOS
ZOOLÓGICO

Carlos Rafael Barbosa Madaleno
Marcos William Ferreira Pinto
Pedro Nuno Batista Bastos

Porto, 2021

Índice

1 Descrição do problema

Neste trabalho pretende-se realizar a implementação de uma base de dados com a finalidade de administrar e organizar um zoológico.

Nessa base de dados representa-se os vários **animais [Animal]**, que são organizados em **espécies [Species]** que por sua vez é uma composição de **classe [Animal Class]**.

Um **animal** pode ter graus de parentesco com outros **animais** do Zoo, se esse for o caso, essa relação é guardada.

Com organização em mente; para cada **animal**, é designado um **habitat [Habitat]**, que por sua vez é uma **estrutura [Structure]** dentro do zoológico. Essa organização é realizada considerando os parâmetros ambientais necessários a sobrevivência do **animal**, tendo por referência a sua **espécie**.

Um habitat pode ser dividido em **terrestre [Terrarium]** ou **aquático [Aquarium]**, de modo que os parâmetros ambientais possam adequar-se ao maior número de **espécies** possível.

Além disso, uma **estrutura** não necessita de conter necessariamente um **habitat**, podendo também agregar **estruturas** destinado para a oferta de **serviços [Services Building]** ao público (**visitante [visitor]**).

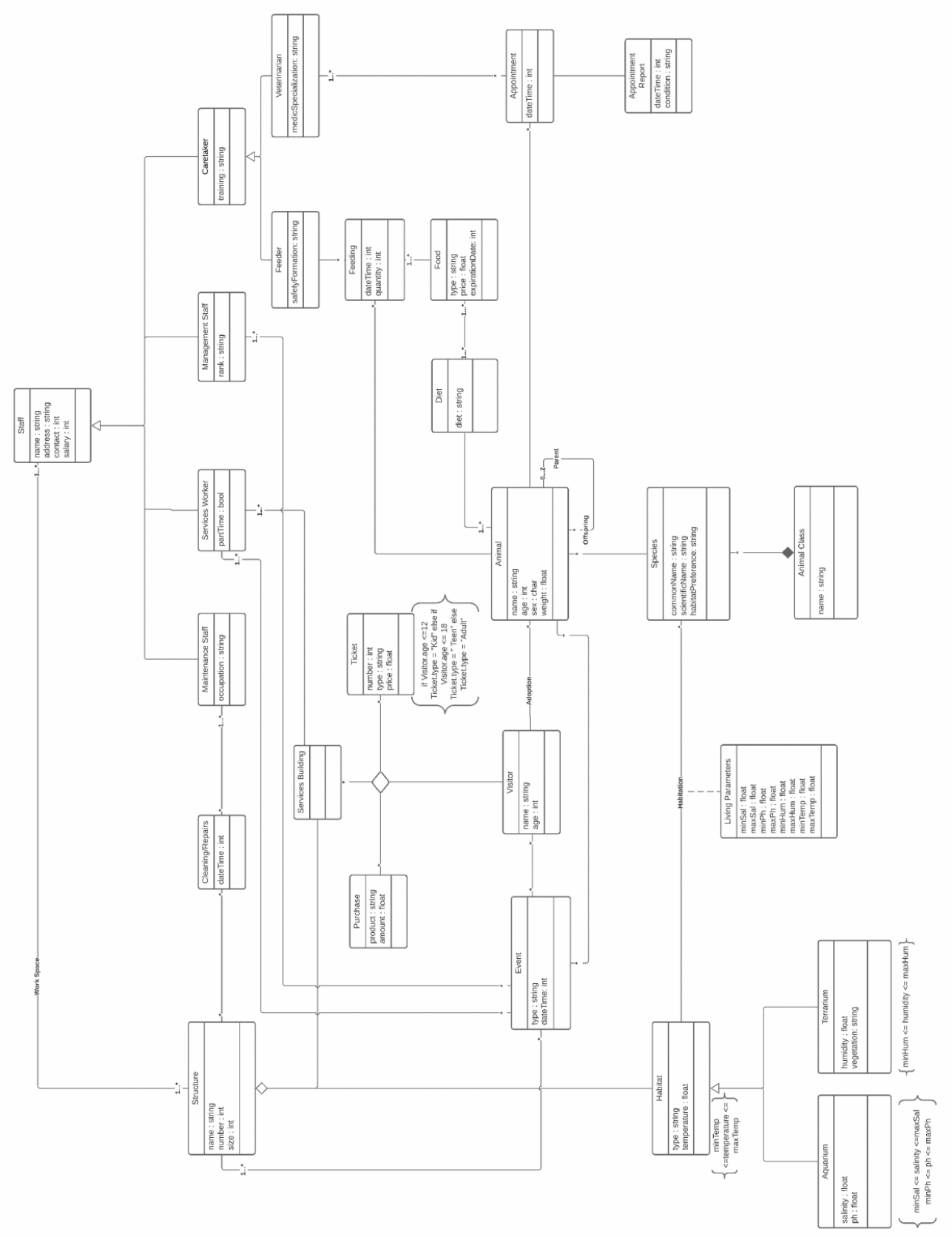
A existência de **staff [Staff]** também é essencial para o funcionamento do Zoo, e cada empregado é dividido consoante a sua função (**manutenção [Maintenance Staff]**, **atendimento ao público [Services Worker]**, **administração [Management Staff]** e **tratamento de animais [Caretaker]**). Cada **funcionário** trabalha em uma ou mais **localidade (estruturas)**. Além do mais, cada **trabalhador**, dependendo da sua área de atuação, realizará funções de modo que o Zoo possa funcionar.

Os **empregados** que trabalham na área de **tratamento de animais** ainda são divididos em **tratadores [Feeder]** ou **veterinários [Veterinarian]**. O primeiro, cuidará da alimentação dos **animais**, enquanto o segundo da saúde dos mesmos.

Ademais, pretende-se registar as compras efetuadas por um **visitante [Visitor]** e os **eventos** a que ele compareceu. Um **visitante** ainda tem a possibilidade de adotar um ou mais **animais**.

Por fim, é possível a realização de **eventos [Event]** (como visitas guiadas, exposições ou shows); para tal, um evento poderá acontecer em 1 ou mais **edifícios (estruturas)**, ter ou não a participação de **animais**, e deverá ter no mínimo 1 **gerente (administração)** e **1 funcionário (atendimento ao público)** a organizá-lo.

2 Diagrama UML



3 Esquema relacional e dependências funcionais

Staff(id, name, address, contact, salary)
Dependência Funcional: {id} → {name, address, contact, salary}

MaintenanceStaff(id → Staff, occupation, name→Staff.name, address→Staff.address, contact→Staff.contact,salary→Staff.salary)
Dependência Funcional: {id} → {occupation, name, address, contact, salary}

ManagementStaff(id → Staff, rank, name→Staff.name, address→Staff.address, contact→Staff.contact,salary→Staff.salary)
Dependência Funcional: {id} → {rank, name, address, contact, salary}

Caretaker(id → Staff, training, name→Staff.name, address→Staff.address, contact→Staff.contact,salary→Staff.salary)
Dependência Funcional: {id} → {training, name, address, contact, salary}

Feeder(id → Caretaker, safetyFormation, name→Staff.name, address→Staff.address, contact→Staff.contact,salary→Staff.salary, training→Caretaker.training)
Dependência Funcional: {id} → {safetyFormation, name, address, contact, salary, training}

Veterinarian(id → Caretaker, medicSpecialization, name→Staff.name, address→Staff.address, contact→Staff.contact,salary→Staff.salary, training→Caretaker.training)
Dependência Funcional: {id} → {medicSpecialization, name, address, contact, salary, training}

Structure(id, number, name, size)
Dependência Funcional: {id} → {number, name, size}

WorkSpace(staff→ Staff, structure -> Structure)
Dependência Funcional: non-existant

CleaningRepairs(id, dateTime)
Dependência Funcional: {id} → {dateTime}

StructureCleaning(cleaning→ CleaningRepairs, structure -> Structure)
Dependência Funcional: non-existant

Upkeep(cleaning→ CleaningRepairs, staff -> MaintenanceStaff)
Dependência Funcional: non-existant

ServicesBuilding(id)
Dependência Funcional: non-existant

ServicesWorker(id → Staff, building → ServicesBuilding, partTime)
Dependência Funcional: {id,building} → {partTime}

Ticket(id,number, type, price)
Dependência Funcional: {id} → {number, type, price}

Purchase(id**,** product, amount)
Dependência Funcional: {id} →{product, amount}

Visitor(id, name, age)
Dependência Funcional: {id} → {name, age}

Acquisition(purchase → Purchase, servicesBuilding → ServicesBuilding, ticket → Ticket, visitor → Visitor)
Dependência Funcional: {purchase, servicesBuilding, ticket} → {visitor}

Habitat(id, type, temperature)
Dependência Funcional: {id} → {type,temperature}

Aquarium(id → Habitat, salinity, ph, type→Habitat.type, temperature→ Habitat.temperature)
Dependência Funcional: {id} → {salinity, ph, type, temperature}

Terrarium(id → Habitat, humidity, vegetation, type→Habitat.type, temperature→Habitat.temperature)
Dependência Funcional: {id} → {humidity, vegetation, type, temperature}

AnimalClass(id, name)
Dependência Funcional: {id} → {name}

Species(id, scientificName, commonName, habitatPreference, id → AnimalClass)
Dependência Funcional: {id} → {scientificName, commonName, habitatPreference}

Habitation(species → Species, habitat- > Habitat, minSal, maxSal, minPh, maxPh, minHum, maxHum, minTemp, maxTemp)
Dependência Funcional: {species, habitat} → {minSal, maxSal, minPh, maxPh, minHum, maxHum, minTemp, maxTemp}

Animal(id, name, age, sex, weight, species → Species, adopter → Visitor, diet → Diet)
Dependência Funcional: {id} → {name, age, sex, weight, species, adopter, diet}

Relationship(mother→ Animal, father→ Animal, child→ Animal)
Dependência Funcional: {child} → {mother, father}

Event(id,type, dateTime)
Dependência Funcional: {id} → { type, dateTime}

StructureHabitat(id,habitat,structure)
Dependência Funcional: {id} → { habitat, structure}

StructureServicesBuilding(id,servicesbuilding,structure)
Dependência Funcional: {id} → { servicesbuilding, structure}

EventStructure(event->Event, structure -> Structure)
Dependência Funcional: non-existant

EventVisitor(event->Event, visitor -> Visitor)
Dependência Funcional: non-existant

EventAnimal(event->Event, animal -> Animal)
Dependência Funcional: non-existant

EventServiceWorker(event->Event, worker -> ServicesWorker)
Dependência Funcional: non-existant

EventManagementWorker(event->Event, worker -> ManagementStaff)
Dependência Funcional: non-existant

Food(id, type, price, expirationDate)
Dependência Funcional: {id} → {type, price, expirationDate}

Diet(id, diet)
Dependência Funcional: {id} → {diet}

DietFood(diet-> Diet, food -> Food)
Dependência Funcional: non-existant

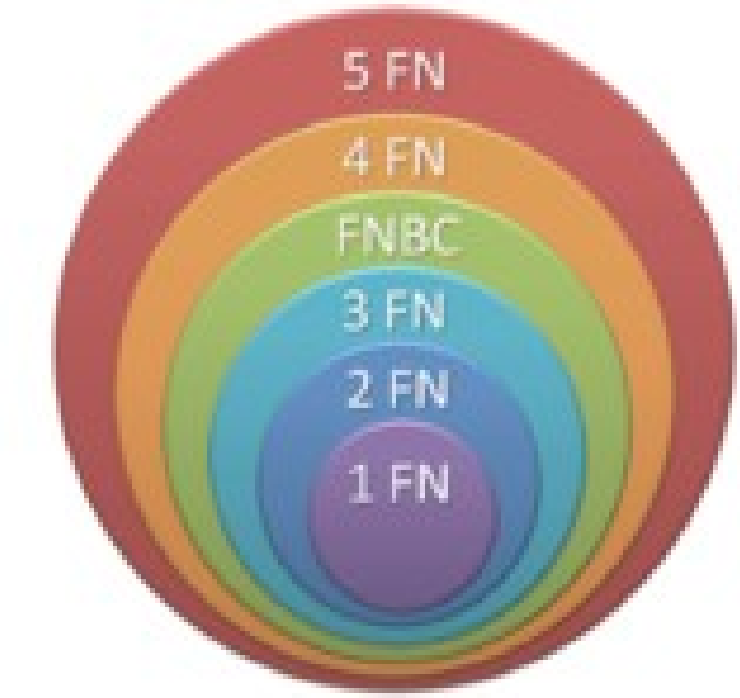
Feeding(id, dateTime, quantity, food -> Food, feeder ->Feeder, animal -> Animal)
Dependência Funcional: {id} → {dateTime, quantity, food, feeder, animal}

Appointment(id, dateTime, animal -> Animal)
Dependência Funcional: {id} → {dateTime, animal}

VetAppointment(vet -> Veterinarian, appointment -> Appointment)
Dependência Funcional: non-existant

AppointmentReport(id, dateTime, condition, appointment -> Appointment)
Dependência Funcional: {id} → {dateTime, condition, appointment}.

4 Formas normais



Todas as relações obedecem a primeira forma normal (1FN) por cada atributo contém apenas valores
atômicos e cada atributo contém apenas valores monovalorados (valores simples, sem conjuntos ou listas). Além disso todo atributo não chave é dependente funcional da chave primaria, por isso podemos dizer que todas as relações também obedecem a segunda forma normal (2FN). Ainda mais, para todas as relações, para cada dependência funcional não trivial (os atributos não são um subconjunto da chave primária) $\{\overline{A}\} \rightarrow \{\overline{B}, \overline{A}\}$ é uma super chave, mais especificamente uma chave. Logo, podemos dizer que as relações obedecem a terceira forma normal (3FN).

Por fim, como todas as relações obedecem a terceira forma normal (3FN) e para cada dependência funcional $\{\overline{A}\} \rightarrow \{\overline{B}, \overline{A}\}$ é uma super chave, mais especificamente uma chave, as relações também obedecem a forma normal de Boyce-Codd.

As relações que não possuem dependências funcionas não violam as regras mencionadas acima, logo, concluímos que obedecem a 2FN, 3FN e a FNBC.

5 Implementação de restrições

No ficheiro “criar.sql” foi necessário a utilização de algumas restrições (constraints) para que o funcionamento da base de dados seja como o esperado para o funcionamento na vida real. Abaixo são mostradas as restrições impostas no script.

Para a restrição da chave primária [PRIMARY KEY] não foram mencionadas as restrições implícitas. Ademais, as chaves primárias com nomes diferente de “id” são consideradas um número de identificação.

Quando não se menciona a integridade referencial duma foreign key, optou-se por restringir a sua eliminação e atualização **[ON DELETE RESTRICT ON UPDATE RESTRICT]**

Staff

A chave primária **[PRIMARY KEY]** é id.

Os funcionários (Staff) precisam de ter nome (name) , morada (adress) e contacto(contact) logo, estes campo não podem ficar vazios **[NOT NULL].** Além disto, dois trabalhadores não podem ter o mesmo contato, por isso este atributo é único **[UNIQUE]**.

O salário (salary) deve ser sempre maior que zero (a razão para não ser maior do que o salário-mínimo é a possível alteração dele e estágios remunerados que recebem menos de 1 salário), adicionalmente pode existir estágios não remunerados logo salário não necessita de ser “NOT NULL **[CHECK > 0].**

MaintenanceStaff

A chave primária **[PRIMARY KEY]** é id.

Cada funcionário tem um cargo definido, logo a sua ocupação (occupation) não pode ser nula **[NOT NULL]**.

Pretende-se que seja possível atualizar o id do Staff alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

ManagementStaff

A chave primária **[PRIMARY KEY]** é id.

Todos os que possuem cargos em gestão ou administração devem possuir rank, isso para que possa ser definido que tem o poder de tomar decisões finais e a área de atuação de cada gerente **[NOT NULL].**

Pretende-se que seja possível atualizar o id do Staff alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

Caretaker

A chave primária **[PRIMARY KEY]** é id.

O cuidador (caretaker) pode possuir algum treinamento específico (para animais mais delicados, ou perigosos, ou que requeiram necessidades especiais) mas isso não é obrigatório, logo pode ser um valor nulo.

Pretende-se que seja possível atualizar o id do Staff alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

Feeder

A chave primária **[PRIMARY KEY]** é id.

É necessário que o alimentador (feeder) possua treinamento de segurança para lidar com os animais tanto para a segurança dele próprio como dos animais em questão. O valor default será o nível mais baixo - "Basic" **[DEFAULT].**

Pretende-se que seja possível atualizar o id do Staff alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

Veterinarian

A chave primária **[PRIMARY KEY]** é id.

Alguns veterinários especializam-se em diversas áreas da saúde (medicSpecialization). O default value será um de clínica geral - "General" **[DEFAULT].**

Pretende-se que seja possível atualizar o id do Staff alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

Structure

A chave primária **[PRIMARY KEY]** é id.

Cada estrutura do zoológico deve possuir um número de identificação (structureNumber), além disso cada número necessita de ser único **[NOT NULL] and [UNIQUE].**

WorkSpace

{staff, structure} é a chave primária **[PRIMARY KEY].**

Para determinar o espaço de trabalho de cada funcionário do zoo é importante que possa haver essa ligação logo staff e structure não podem ser nulos **[NOT NULL]**.

Além disso, ambos não precisam ser únicos pois tanto funcionários podem ter mais de um espaço de trabalho como um espaço de trabalho pode ter mais de um trabalhador.

CleaningRepairs

A chave primária **[PRIMARY KEY]** é id.
dateTime deve conter a data e hora da limpeza/reparação realizada **[NOT NULL].**

StructureCleaning

{cleaning, structure} é a chave primária **[PRIMARY KEY].**
São necessárias as restrições de ambos os valores abaixo para a relação entre Structure e cleaningRepairs, logo cleaning e strucuture não podem ser nulos [NOT NULL].
Isso tem o intuito de registar as limpezas e reparos feitas nas diversas estruturas (edifícios e habitats) do zoológico.

Upkeep

{cleaning, staff} é a chave primária **[PRIMARY KEY].**
cleaning deve ser um valor não nulo **[NOT NULL].**
staff também deve ser um valor válido **[NOT NULL].**

ServicesBuilding

A chave primária **[PRIMARY KEY]** é id.

ServicesWorker

{id, building} é a chave primária **[PRIMARY KEY].**
partTime não deve ser nulo, além disso foi utilizado um integer para determinar se o serviceWorker é part-time ou não. Nesse caso 1 representa parttime e 0 representa que o trabalhador não é parttime **[NOT NULL] and [CHECK == 1 OR CHECK ==0].**

Ticket

A chave primária **[PRIMARY KEY]** é id.
ticketNumber deve ser o número único de um bilhete e todo bilhete dever possuir um número **[NOT NULL] and [UNIQUE].**
Os tickets possuem diferentes tipos (Kid, Teen e Adult), por padrão o default é o ‘’Adult” **[DEFAULT “Adult”].**
O preço do bilhete também é armazenado, o default é 20. Ademais o preço de um bilhete deve sempre ser 0 ou superior **[DEFAULT 20] and [CHECK >=0].**

Purchase

A chave primária **[PRIMARY KEY]** é id.
product não pode ser nulo **[NOT NULL].**
amount não pode ser nulo e deve ser um valor igual ou superior a 0 **[NOT NULL] and [CHECK >=0].**

Visitor

A chave primária **[PRIMARY KEY]** é id.
O nome (name) de um visitante deve ser sempre registrado **[NOT NULL].**
A idade (age) de um visitante deve ser sempre registrada e deve possuir um valor superior ou igual a 0 **[NOT NULL] and [CHECK >=0].**

Acquisition

{purchase, servicesBuilding, ticket} é a chave primária **[PRIMARY KEY].**
ServicesBuilding não deve ser um valor nulo (deve sempre haver o lugar ou se realizou a compra) **[NOT NULL].**
Ticket não deve ser um valor nulo (o número do bilhete é registrado na compra, ou é adicionado no momento quando se compra o próprio bilhete) **[NOT NULL].**
Visitor não pode ser nulo (deve sempre haver alguém a realizar a compra) **[NOT NULL].**

Habitat

A chave primária [PRIMARY KEY] é id.
É necessário saber sempre o tipo de habitat logo não poder ser um valor nulo **[NOT NULL].**
A temperatura média (temperature) de um habitat deve ser sempre conhecida, por isso deve ser sempre mencionada **[NOT NULL].**

Aquarium

A chave primária **[PRIMARY KEY]** é id.
A salinidade de um aquário deve ser conhecida para determinar que tipo de animais aquáticos podem habitar nele **[NOT NULL].**
Da mesma forma que a salinidade, é preciso ter o pH, tendo em mente os parâmetros para a sobrevivência de um animal **[NOT NULL].**
Pretende-se que seja possível atualizar o id do Habitat alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

Terrarium

A chave primária **[PRIMARY KEY]** é id.

A humidade (humidity) deve sempre ser conhecida (para determinar a sobrevivência dos animais), além disso, como é medida em porcentagem deve estar entre 0 e 100 **[NOT NULL] and [CHECK >=0 and CHECK <=100].**

Pretende-se que seja possível atualizar o id do Habitat alterando-o nesta relação **[ON UPDATE CASCADE]**, mas proibir a sua eliminação **[ON DELETE RESTRICT].**

AnimalClass

A chave primária **[PRIMARY KEY]** é id.

O nome (name) da classe deve sempre conhecido e ser diferente de outras classes **[NOT NULL] and [UNIQUE].**

Species

A chave primária **[PRIMARY KEY]** é id.

O nome científico (scientific name) de uma espécie deve sempre ser conhecido para ajudar na organização, além disso, como é utilizado para identificação de um animal deve ser único **[NOT NULL] and [UNIQUE].**

Habitation

{species, habitat} é a chave primaria **[PRIMARY KEY].**

species, habitat, minTemp, maxTemp devem ser conhecidos **[NOT NULL].**

minSal, maxSal, minPh, maxPh, minHum, maxHum podem ser nulos (por exemplo um terrário não necessita de se verficar salinidade ou pH.

Para além disso, devemos verificar (**[CHECK]**):

- minSal >= 0;
- maxSal >= 0;
- minSal <= maxSal (A salinidade mínima deve ser menor que a máxima);
- 0 <= minPh <=14;
- 0 <= maxPh <=14;
- minPh <= maxPh (O pH mínimo deverá ser menor que máximo);
- 0 <= minHum <=100;
- 0 <= maxHum <=100;
- minHum <= maxHum (A humidade mínima deverá ser menor que a máxima);
- minTemp <= maxTemp (A temperatura mínima deverá ser menor que a máxima).

Animal

A chave primária **[PRIMARY KEY]** é id.
O nome do animal pode ser nulo pois ele pode ainda não possuir nome.
A idade de um animal deve ser conhecida, além disso a idade deve ser necessariamente um valor maior ou igual a 0 **[NOT NULL] and [CHECK >=0].**
O sexo (sex)de um animal de um animal deve ser conhecido **[NOT NULL].**
O peso (weight) de um anima deve ser conhecido **[NOT NULL].**
A espécie (species) a que um animal pertence também deverá ser conhecida **[NOT NULL].**
A dieta (diet) deverá ser conhecida para ter-se os devidos cuidados ao animal **[NOT NULL].**
Um adotante (adopter) pode não existir, nesse o valor poderá ser nulo.

Relationship

A chave primária **[PRIMARY KEY]** é id.
A mãe e/ou o pai de um animal poderá não ser conhecido. Nesse caso poderá ser deixado em branco (animais que chegaram ou zoológico, por exemplo).

Event

A chave primária **[PRIMARY KEY]** é id.
O tipo (type) de evento realizado deverá sempre ser conhecido **[NOT NULL].**
A data e o horário de um evento também são devem ser sabidas, por isso dateTime deve ser conhecida **[NOT NULL].**

EventStructure

{event, structure} é a chave primaria **[PRIMARY KEY].**
Para se determinar o lugar onde um evento se realizará é necessário saber event e strucuture, logo estes não podem ser nulos **[NOT NULL]**.

EventVisitor

{event, visitor} é a chave primaria **[PRIMARY KEY].**
Para se determinar os visitantes que participaram ou participarão de um evento é necessário saber event e visitor, logo estes não podem ser nulos **[NOT NULL]**.

EventAnimal

{event, animal} é a chave primaria **[PRIMARY KEY].**
Para se determinar os animais que participaram/ participarão de um evento é necessário saber event e animal, logo estes não podem srer nulos **[NOT NULL]**.

EventServiceWorker

{event, worker} é a chave primária **[PRIMARY KEY].**

Para se determinar os funcionários que participaram/ participarão de um evento é necessário saber event e worker, logo estes não podem ser nulos **[NOT NULL]**.

EventManagementWorker

{event, admin} é a chave primária **[PRIMARY KEY].**

Para se determinar os gerentes/ administradores que participaram/ participarão de um evento é necessário saber event e admin, logo estes não podem ser nulos **[NOT NULL]**.

Food

A chave primária **[PRIMARY KEY]** é id.

O tipo (type) de comida deve ser sempre mencionado (para definir o que um animal pode ou não comer, ou se a comida é para animais carnívoros ou herbívoros) **[NOT NULL].**

O preço (price) de uma comida deve ser sempre conhecido e deve ser um valor igual ou superior a 0 **[NOT NULL] and [CHECK >= 0].**

A data de validade (expirationDate) de um determinado alimento deve sempre ser conhecida **[NOT NULL].**

Diet

A chave primária **[PRIMARY KEY]** é id.

A dieta (diet) deve ser um valor conhecido **[NOT NULL].**

DietFood

{diet, food} é a chave primária **[PRIMARY KEY].**

diet e food devem sempre ser conhecidos **[NOT NULL].**

Feeding

A chave primária **[PRIMARY KEY]** é id.

A data e o horário de alimentação devem ser conhecidos, por isso dateTime deve ser **[NOT NULL].**

A quantidade (quantity)para a alimentação também deve ser sempre conhecida **[NOT NULL].**

É necessário conhecermos sempre o animal, o alimentador, e a comida a ser dado, por isso food, feeder e animal não podem ser nulos **[NOT NULL]**.

Appointment

A chave primária **[PRIMARY KEY]** é id.
A data e o horário da consulta devem ser conhecidos, por isso dateTime deve ser **[NOT NULL].**
O animal para a consulta deve ser conhecido por isso, animal deve ser **[NOT NULL].**

VetAppointment

{vet, appointment} é a chave primária **[PRIMARY KEY].**
Tanto vet como appointment devem ser conhecidos **[NOT NULL].**

AppointmentReport

A chave primária **[PRIMARY KEY]** é id.
A data e o horário do relatório médico devem ser conhecidos, por isso dateTime deve ser **[NOT NULL].**
A condição (condition) do animal deve sempre ser conhecida, caso o animal esteja saudável o default é "healthy" **[DEFAULT "HEALTHY"].**
A consulta relacionada a esse relatório também deverá ser conhecida, nesse caso apppointment deve ser **[NOT NULL].**

6 Interrogações

- Quais os animais que têm pais conhecidos e em quem são?
  Realizado com o documento **int1.sql**

- Os eventos que aconteceram num determinado mês e ano?
  Realizado com o documento **int2.sql** (foi utilizado o mês de Abril /2021 na query como exemplo por causa do povoar.sql)

- O que cada animal pode comer?
  Realizado com o documento **int3.sql**

- Quanto foi gasto com comida no último mês?
  Realizado com o documento **int4.sql**

- Qual a lista de pessoas com ligações ao zoológico e qual essa ligação e quantas pessoas são?
  Realizado com o documento **int5.sql**

- Quais são os animais que apresentaram alguma complicação no último exame veterinário?
  Realizado com o documento **int6.sql**

- Qual o melhor habitat para uma determinada espécie?
  Realizado com o documento **int7.sql**

- Onde cada funcionário trabalha e qual a descrição desse funcionário?
  Realizado com o documento **int8.sql**

- Qual Structure necessita de limpeza, e qual a data em que foi limpa anteriormente?
  Realizado com o documento **int9.sql**

- Quanto cada visitante gastou no zoológico e quais os dados desses visitantes?
  Realizado com o documento **int10.sql**

7 Gatilhos

Gatilho 1:

O primeiro gatilho é usado para detectar o deletar de uma linha na tabela “Purchase”, que tem a utilização para cancelar compras erradas. Quando isso acontece a linha relacionada a ID da compra cancelada em “Acquisition” também é deletada.

Gatilho 2:

Quando ocorre a alteração do número de contato de algum “Staff” o segundo gatilho verifica se o novo número é valido, caso não for a operação não é realizada e uma mensagem de que o novo contato não é valido é mostrada.

Gatilho 3:

Quando alguém tenta inserir um novo valor a tabela “Relationship” (que guardar os animais que possuem conhecidos) o terceiro gatilho verifica se a mãe tem sexo feminino, o pai tem sexo masculino, ambos os pais têm idade superior a criança e se pais e filhos são da mesma espécie. Caso uma ou mais dessas verificações falhem a mensagem “RELATIONSHIP NOT VALID” é mostrada.

Gatilho 4:

Ao inserir algum tuplo na tabela Habitation, verifica se o Habitat está de acordo com os Living Parameters. Se não, aborta a operação e mostra uma mensagem de erro.