U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Base de Dados
Jardim Zoológico

L.EIC · BDAD · Grupo 207 · 2021/2022

Docente Lázaro Costa

Daniela Tomás, up202004946

Nuno Penafort, up202008405

Sofia Sousa, up202005932

Índice

Contexto

Um Jardim Zoológico pretende armazenar informação relativa ao seu funcionamento. O Zoo tem um nome, uma rua, um código postal, o número de telefone, uma área, hora de abertura e de fecho, o número total de animais e uma lotação atual e máxima que, de acordo com esta, o Zoo pode receber inúmeros visitantes. Sobre os visitantes é necessário conhecer o nome, o NIF, o número de telefone e a idade. Cada visitante tem apenas um bilhete, cujo preço varia consoante a idade e tem uma hora de entrada, a data e um código único. Além disso, o Zoo tem disponíveis sessões que são protagonizadas por animais e apresentadas por um tratador e têm um nome, uma descrição e um horário com hora de início e de fim. Os tratadores também são responsáveis por tratar dos animais fora das sessões.

Existem vários tipos de funcionários para além dos tratadores (manutenção, receção, etc…) e é necessário conhecer o nome, NIF, rua, código postal, data de nascimento, número de telefone e email.

Quanto aos animais é importante saber o nome, o nome comum, o nome científico, o género, a idade, data de chegada ao Zoo e a alimentação, onde é preciso ter em conta a quantidade de alimentos necessários, a quantidade de alimentos disponíveis e o tipo de alimentação. Os animais vivem em diferentes tipos de habitats com uma determinada área e com um determinado número de animais e podem ou não ter uma progenitora que pode ter várias crias.

O Zoo tem alguns serviços associados como, por exemplo, hotéis e restaurantes. Ambos possuem um nome, hora de abertura e de fecho, rua, código postal, número de telefone, capacidade e email. Os hotéis têm um preço base e os restaurantes servem um tipo característico de comida e têm só um preço com tudo incluído.

Diagrama UML

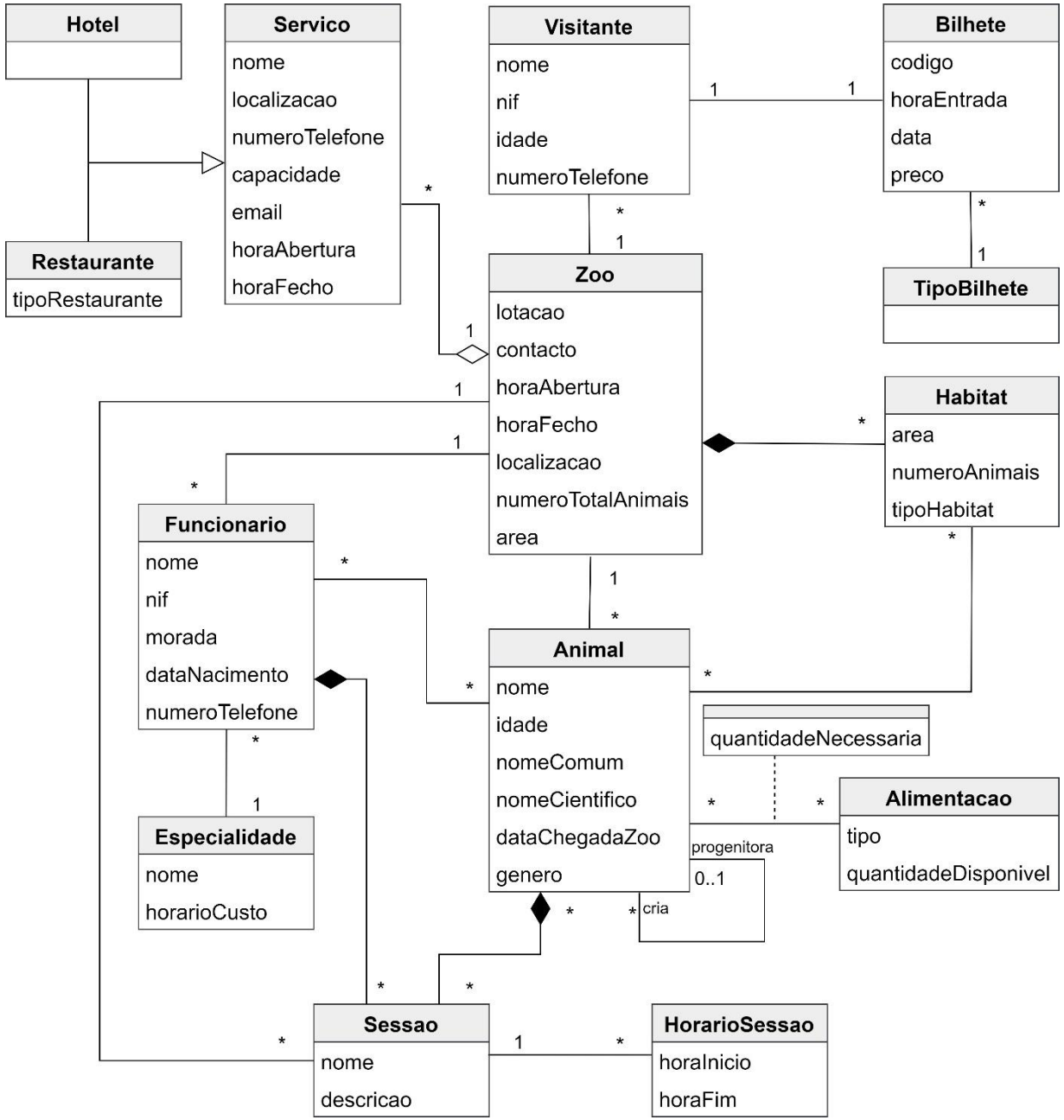
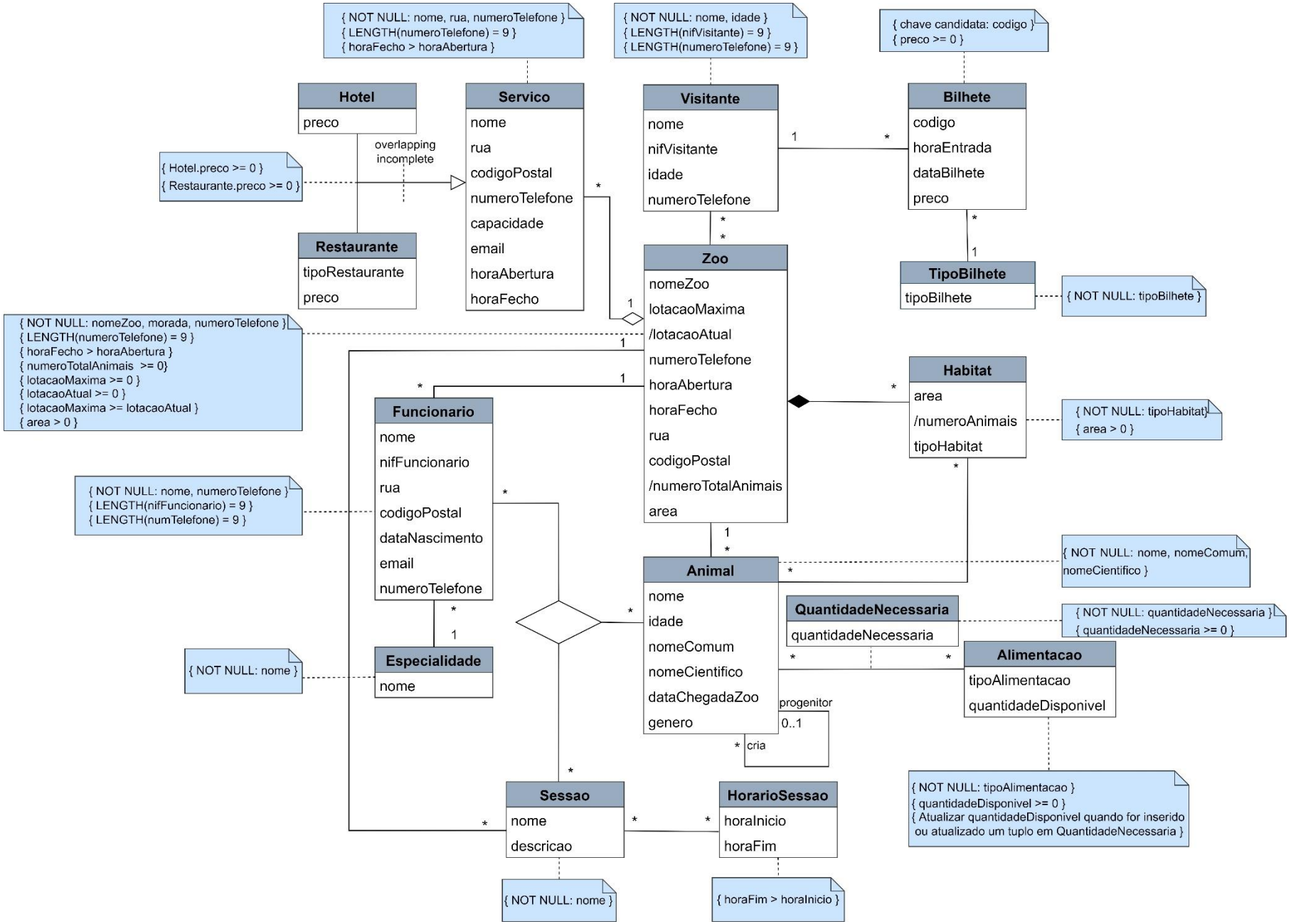


Diagrama UML (revisto)



Esquema Relacional

- **Zoo** (idZoo, lotaçãoMaxima, lotacaoAtual, numeroTelefone, horaAbertura, horaFecho, rua, codigoPostal, numeroTotalAnimais, area)
- **Visitante** (nifVisitante, nome, idade, numeroTelefone)
- **Bilhete** (codigo, horaEntrada, dataBilhete, preco, nifVisitante -> Visitante, idTipoBilhete -> TipoBilhete)
- **TipoBilhete** (idTipoBilhete, tipoBilhete)
- **Servico** (idServico, nome, numeroTelefone, capacidade, email, horaAbertura, horaFecho, rua, codigoPostal, idZoo -> Zoo)
- **Hotel** (idServico -> Servico, preco)
- **Restaurante** (idServico -> Servico, tipoRestaurante, preco)
- **Animal** (idAnimal, nome, idade, nomeComum, nomeCientifico, dataChegadaZoo, genero, idZoo -> Zoo, idProgenitor -> Animal)
- **Funcionario** (nifFuncionario, nome, rua, codigoPostal, dataNascimento, email, numeroTelefone, idEspecialidade -> Especialidade, idZoo -> Zoo)
- **Especialidade** (idEspecialidade, nome)
- **Habitat** (idHabitat, area, numeroAnimais, tipoHabitat, idZoo -> Zoo)
- **Sessao** (idSessao, nome, descricao, idZoo -> Zoo)
- **HoraioSessao** (idHorarioSessao, horaInicio, horaFim)
- **SessaoHoraioSessao** (idSessao -> Sessao, idHorarioSessao -> HorarioSessao)
- **Alimentacao** (idAlimentacao, tipoAlimentacao, quantidadeDisponivel)
- **VisitanteZoo** (nifVisitante -> Visitante, idZoo -> Zoo)
- **FuncionarioAnimalSessao** (idAnimal -> Animal, nifFuncionario -> Funcionario, idSessao -> Sessao)
- **AnimalHabitat** (idAnimal -> Animal, idHabitat -> Habitat)
- **QuantidadeNecessaria** (idAnimal -> Animal, idAlimentacao -> Alimentacao, quantidadeNecessaria)

Análise Dependências Funcionais e Formas Normais

- **Zoo:**

  FDs:

  - idZoo -> nomeZoo, lotacaoMaxima, lotacaoAtual, numeroTelefone, rua, codigoPostal, horaAbertura, horaFecho, numeroTotalAnimais, area

- **Visitante:**

  FDs:

  - nifVisitante -> nome, idade, numeroTelefone

- **Bilhete:**

  FDs:

  - codigo -> horaEntrada, dataBilhete, preco, idTipoBilhete

- **TipoBilhete:**

  FDs:

  - idTipoBilhete -> tipoBilhete
  - tipoBilhete -> idTipoBilhete

- **Servico:**

  FDs:

  - idServico -> nome, numeroTelefone, capacidade, email, horaAbertura, horaFecho, rua, codigoPostal, idZoo

- **Hotel:**

  FDs:

  - idServico -> preco

- **Restaurante:**

  FDs:

  - idServico -> tipoRestaurante, preco

- **Animal:**

  FDs:

  - idAnimal -> nome, idade, nomeComum, nomeCientifico, dataChegadaZoo, genero, idZoo, idProgenitor

- **Funcionario:**

  FDs:

  - nifFuncionario -> nome, rua, codigoPostal, dataNascimento, email, numeroTelefone, idEspecialidade, idZoo

- **Especialidade:**

  FDs:

  - idEspecialidade -> nome

- **Habitat:**

  FDs:

  - idHabitat -> area, numeroAnimais, tipoHabitat, idZoo
- **Sessao:**

  FDs:

  - idSessao -> nome, descricao, idZoo

- **HoraioSessao:**

  FDs:

  - idHorarioSessao -> horaInicio, horaFim

- **Alimentacao:**

  FDs:

  - idAlimentacao -> tipoAlimentacao, quantidadeDisponivel

- **QuantidadeNecessaria:**

  FDs:

  - idAnimal, idAlimentacao -> quantidadeNecessaria

As relações estão todas na 3ª Forma Normal e na Forma Normal de Boyce-Codd, porque a partir lado esquerdo de cada dependência funcional podemos obter todos os atributos da relação, ou seja, existe sempre uma (super)chave do lado esquerdo.

Restrições

- **Zoo:**
  - Não podem existir zoos com o mesmo id:

    idZoo PRIMARY KEY
  - Não podem existir nomes e números de telefone iguais:

    nomeZoo constraint nome_zoo_u UNIQUE

    numeroTelefone constraint numero_telefone_zoo_u UNIQUE
  - Os zoos têm sempre nome e número de telefone atribuídos:

    nomeZoo constraint nome_zoo_nn NOT NULL

    numeroTelefone constraint numero_telefone_zoo_nn NOT NULL
  - A hora de fecho é maior do que a hora de abertura:

    constraint horas_zoo_c CHECK (horaFecho > horaAbertura)
  - O número de telefone tem de ter 9 caracteres:

    constraint numero_telefone_zoo_c CHECK (LENGTH (numeroTelefone) = 9)
  - O número total de animais, a lotação máxima e a lotação atual são maiores ou iguais a zero e a área é maior que zero:

    constraint numero_total_animais_zoo_c CHECK (numeroTotalAnimais >= 0)

    contraint lotação_maxima_zoo_c CHECK (lotacaoMaxima >= 0)

    constraint lotacao_atual_zoo_c CHECK (lotacaoAtual >= 0)

    constraint área_zoo_c CHECK (area > 0)
  - A lotação máxima é maior que a lotação atual:

    constraint lotacao_zoo_c CHECK (lotacaoMaxima >= lotacaoAtual)
- **Visitante:**
  - Não podem existir visitantes com o mesmo id:

    nifVisitante PRIMARY KEY
  - Não podem existir números de telefone e NIFs iguais:

    numeroTelefone constraint numero_telefone_visitante_u UNIQUE
  - Os visitantes têm sempre nome e idade atribuídos:

    nome constraint nome_visitante_nn NOT NULL

    idade constraint idade_nn NOT NULL
  - O número de telefone tem de ter 9 caracteres e o NIF 9 caracteres:

    constraint numero_telefone_visitante_c CHECK (LENGTH (numeroTelefone) = 9)

    constraint nif_visitante_c CHECK (LENGTH (nifVisitante) = 9)
- **Bilhete:**
  - Não podem existir visitantes com o mesmo id:

    codigo PRIMARY KEY
  - O preço é maior ou igual a zero:

    constraint preco_bilhete_c CHECK (preco >= 0)
  - O id do visitante corresponde a um id da classe Visitante e o id do zoo corresponde a um id da classe Zoo:

    nifVisitante REFERENCES Visitante (nifVisitante) NOT NULL

    idTipoBilhete REFERENCES TipoBilhete (idTipoBilhete) NOT NULL
- **TipoBilhete:**
  - Não podem existir tipos de bilhete com o mesmo id:

    idTipoBilhete PRIMARY KEY
  - Os tipos de bilhete têm sempre um tipo de bilhete atribuído:

    tipoBilhete constraint tipo_bilhete_nn NOT NULL
- **Servico:**
  - Não podem existir serviços com o mesmo id:

    idServico PRIMARY KEY

– Não podem existir rua e números de telefone iguais:

numeroTelefone constraint numero_telefone_servico_u UNIQUE

– Os serviços têm sempre um nome, rua e número de telefone atribuídos:

nome constraint nome_servico_nn NOT NULL

rua constraint rua_servico_ nn NOT NULL

numeroTelefone constraint numero_telefone_servico_nn NOT NULL

– A hora de fecho é maior do que a hora de abertura:

constraint horas_servico_c CHECK (horaFecho > horaAbertura)

– O número de telefone tem de ter 9 caracteres:

constraint numero_telefone_servico_c CHECK (LENGTH (numeroTelefone) = 9)

– O id do zoo corresponde a um id da classe Zoo:

idZoo REFERENCES Zoo (idZoo) NOT NULL

- **Hotel / Restaurante:**

– Não podem existir hotéis/restaurantes com o mesmo id e estes ids correspondem a um id da classe Servico:

idServico PRIMARY KEY REFERENCES Servico (idServico) NOT NULL

– O preço tanto dos hotéis como dos restaurantes é maior ou igual a zero:

constraint preco_hotel_c / preco_restaurante_c CHECK (preco >= 0)

- **Animal:**

– Não podem existir animais com o mesmo id:

idAnimal PRIMARY KEY

– Os animais têm sempre nome, nome comum e nome científico atribuídos:

nome constraint nome_animal_nn NOT NULL

nomeComum constraint nome_comum_animal_nn NOT NULL

nomeCientifico constraint nome_cientifico_animal_nn NOT NULL

– O id do zoo corresponde a um id da classe Zoo e o id do progenitor corresponde a um id da classe Animal:

idZoo REFERENCES Zoo (idZoo) NOT NULL

idProgenitor REFERENCES Progenitor (idAnimal) NOT NULL

- **Funcionario:**

– Não podem existir funcionários com o mesmo id:

nifFuncionario PRIMARY KEY

– Não podem existir números de telefone, NIFs e emails iguais:

numeroTelefone constraint numero_telefone_funcionario_u UNIQUE

email constraint email_funcionario_u UNIQUE

– Os funcionários têm sempre nome e número de telefone atribuídos:

nome constraint nome_funcionario_nn NOT NULL

numeroTelefone constraint numero_telefone_funcionario_nn NOT NULL

– O número de telefone tem de ter 9 caracteres e o NIF 9 caracteres:

constraint numero_telefone_funcionario_c CHECK (LENGTH (numeroTelefone) = 9)

constraint nif_funcionario_c CHECK (LENGTH (nifFuncionario) = 9)

– O id do zoo corresponde a um id da classe Zoo e o id da especialidade corresponde a um id da classe Especialidade:

idZoo REFERENCES Zoo (idZoo) NOT NULL

idEspecialidade REFERENCES Especialidade (idEspecialidade) NOT NULL

- **Especialidade:**

– Não podem existir especialidades com o mesmo id:

idEspecialidade PRIMARY KEY

– A especialidade tem sempre um nome atribuído:

nome constraint nome_especialidade_nn NOT NULL

- **Habitat:**

– Não podem existir habitats com o mesmo id:

idHabitat PRIMARY KEY

- Os habitats têm sempre um tipo:

  tipoHabitat constraint tipo_habitat_nn NOT NULL

- O número de animais é maior ou igual a zero e a área é maior que zero:

  constraint numero_animais_habitat_c CHECK (numeroAnimais >= 0)

  CHECK (area > 0)

- O id do zoo corresponde a um id da classe Zoo:

  idZoo REFERENCES Zoo (idZoo) NOT NULL

- **Sessao:**

- Não podem existir sessões com o mesmo id:

  idSessao PRIMARY KEY

- As sessões têm sempre nome:

  nome constraint nome_sessao_nn NOT NULL

- O id do zoo corresponde a um id da classe Zoo:

  idZoo REFERENCES Zoo (idZoo) NOT NULL

- **HorarioSessao:**

- Não podem existir horários de sessões com o mesmo id:

  idHorarioSessao PRIMARY KEY

- Os horários das sessões têm sempre nome:

  nome constraint nome_horario_sessao_nn NOT NULL

- A hora de fim é maior do que a hora de início:

  constraint horas_horario_sessao_c CHECK (horaFim > horaInicio)

- **SessaoHorarioSessao:**

- Não podem existir instâncias com o mesmo conjunto {idSessao, idHorarioSessao}:

  PRIMARY KEY (idSessao, idHorarioSessao)

- O id da sessão corresponde a um id da classe Sessao, o id do horário da sessão corresponde a um id da classe HorarioSessao:

  idSessao REFERENCES Sessao (idSessao) NOT NULL

  idHorarioSessao REFERENCES HorarioSessao (idHorarioSessao) NOT NULL

- **Alimentacao:**

- Não podem existir alimentações com o mesmo id:

  idAlimentacao PRIMARY KEY

- A alimentação tem sempre um tipo e quantidade disponível:

  tipoAlimentacao constraint tipo_alimentacao_nn NOT NULL

  quantidadeDisponivel constraint quantidade_disponivel_nn NOT NULL

- A quantidade disponível é maior ou igual a zero:

  constraint quatidade_disponivel_c CHECK (quantidadeDisponivel >= 0)

- **VisitanteZoo:**

- Não podem existir instâncias com o mesmo conjunto {nifVisitante, idZoo}:

  PRIMARY KEY (nifVisitante, idZoo)

- O id do visitante corresponde a um id da classe Visitante e o id do zoo corresponde a um id da classe Zoo:

  nifVisitante REFERENCES Visitante (nifVisitante) NOT NULL

  idZoo REFERENCES Zoo (idZoo) NOT NULL

- **FuncionarioAnimalSessao:**

- Não podem existir instâncias com o mesmo conjunto {nifFuncionario, idAnimal, idSessao}:

  PRIMARY KEY (nifFuncionario, idAnimal, idSessao)

- O id do funcionário corresponde a um id da classe Funcionário, o id do animal corresponde a um id da classe Animal e o id da sessão corresponde a um id da classe Sessao:

  nifFuncionario REFERENCES Funcionario (nifFuncionario) NOT NULL

  idAnimal REFERENCES Animal (idAnimal) NOT NULL

  idSessao REFERENCES Zoo (idSessao) NOT NULL

- **AnimalHabitat:**

– Não podem existir instâncias com o mesmo conjunto {idAnimal, idHabitat}:

PRIMARY KEY (idAnimal, idHabitat)

– O id do animal corresponde a um id da classe Animal e o id do habitat corresponde a um id da classe Habitat:

idAnimal REFERENCES Animal (idAnimal) NOT NULL

idHabitat REFERENCES Habitat (idHabitat) NOT NULL

- **QuantidadeNecessaria:**

– Não podem existir instâncias com o mesmo conjunto {idAnimal, idAlimentacao}:

PRIMARY KEY (idAnimal, idAlimentacao)

– O id do animal corresponde a um id da classe Animal e o id da alimentação corresponde a um id da classe Alimentacao:

idAnimal REFERENCES Animal (idAnimal) NOT NULL

idAlimentacao REFERENCES Alimentacao (idAlimentacao) NOT NULL

– A quantidade necessária não pode ser nula:

quantidadeNecessaria constraint quatidade_necessaria_nn NOT NULL

– A quantidade necessária tem de ser maior ou igual a zero:

constraint quantidade_necessaria_c CHECK (quantidadeNecessaria >= 0)

Interrogações

1. Quantidade necessária média de cada tipo de alimentação. Retorna a quantidade por ordem decrescente.

2. Alimentos que necessitam de reabastecimento, ou seja, caso a quantidade necessária por tipo de alimentação seja maior que a quantidade disponível. Retorna a quantidade disponível, quantidade necessária total e o idAlimentacao por ordem crescente.

3. Número de horas que cada animal participa em sessões por dia. Retorna por ordem crescente de horas.

4. Número de estabelecimentos por cada tipo de serviço. Retorna os nomes dos serviços por ordem alfabética.

5. Lotação atual em cada zoo. Retorna por ordem de lotação atual decrescente.

6. Zoo que tem mais lucro na venda de bilhetes. Retorna o lucro e, caso haja zoos com o mesmo lucro, retorna o idZoo por ordem crescente.

7. Nome dos tratadores responsáveis por mais do que dois animais. Retorna os nomes dos funcionários por ordem alfabética.

8. Percentagem que cada habitat ocupa no zoo. Retorna o nome do zoo, o idHabitat, tipo de habitat por ordem alfabética e a percentagem.

9. Animais cuja idade seja menor do que um e que não tenham um progenitor.

10. Os funcionários que já visitaram o zoo sem ser em trabalho. Retorna o nome dos funcionários por ordem alfabética.

Gatilhos

1. A quantidade disponível atualiza sempre que for atualizado um registo em quantidade necessária.

2. Impede que seja inserido um funcionário com idade inferior a 18 anos.

   Nota: Este gatilho pode ser violado caso seja atualizada a data de nascimento para um valor inválido. Para o evitar, teríamos de criar um gatilho adicional que impedisse essa atualização.

3. Impede que seja atualizado um funcionário em FuncionarioAnimalSessao que não tenha como especialidade “tratador”, porque apenas os tratadores podem apresentar sessões.

   Nota: Este gatilho pode ser violado caso seja inserido um funcionário que não seja um tratador. Para o evitar, teríamos de criar um gatilho adicional que impedisse essa inserção.

Autoavaliação

Daniela Tomás: 33,3%

Nuno Penafort: 33,3%

Sofia Sousa: 33,3%