Universitatea Tehnică „Gheorghe Asachi" din Iași

Facultatea de Automatică și Calculatoare

Domeniul: Calculatoare și Tehnologia Informației

ZOO
~Proiect la disciplina~
~Baze de Date~

Studenta:Pasa Larisa
Anul: 3
Grupa: 1306B
Profesor Coordonator: Mironeanu Catalin

1. Descrierea proiectului :

Scopul:

Scopul aplicatiei este de a crea o baza de date care sa tina evidenta animalelor din gradina Zoologica. Acestea sunt tinute in tarcuri specifice fiecarui tip de animal. Tarcurile pot contine si corpuri anexe. Tipurile de animal sunt standardizate, limitate, in raport cu tarcurile existente. Animalele trebuie sa aiba informatii generale, cum ar fi numele si ziua de nastere. Este de interes si dieta animalelor, tipul acesteia, dar si ce mananca si cantitatea pentru animalul in cauza. La ZOO exista un cabinet veterinar in care trebuie sa existe un registru cu toate inregistrarile cu vizitele animalelor. O inregistrare trebuie sa cuprinda informatii medicale de interes, precum posibile operatii, medicatie, data de nastere si altele.

Datele de interes:

-Animals: elementul principal al unei gradini Zoologice sunt tocmai animalele. Avem nevoie sa stocam animalele din zoo intr-o entitate Animals

-Animal_Info:stocarea informatiilor referitoate la fiecare animal in parte se va realiza in entitatea Animal_Info in care date de interes sunt numele si data de nastere ale animalului.

-Animal_Type: avem nevoie sa stocam datele despre tipurile de animale existente in zoo pentru o imagine de ansamblu asupra gradinii zoologice

-Animal_Pen: trebuie sa stim tarcurile in care animalele pot locui pentru a le organiza corespunzator

-Medical_Info: ne intereseaza sa avem o evidenta asupra starii de sanatate a animalelor pentru tratament si dieta potrivite

-Food: este de interes stocul de mancare din gradina zoologica. In aceasta entitate se stocheaza ce tipuri de mancare sunt disponibile.

-Diet: vom folosi entitatea diet pentru stocarea datelor despre nutritia fiecarui animal

Functiile principale ale proiectului:

- ✓ Inregistrarea animalelor
- ✓ Evidenta nutritiei acestora
- ✓ Evidenta tipurilor de animale
- ✓ Evidenta tarcurilor
- ✓ Inregistrarea aspectelor legate de sanatate

2. Structura si relatiile dintre entitati :

Sunt prezente urmatoarele tipuri de relatii:

➢ One-To-One (1 : 1) :

“Animal_Info” are o legatura de **1:1** cu entitatea **“Animals”** . In cea de-a doua tabela mentionata se afla doar numarul de identificare al animalului, iar in “Animal

Info” este o inregistrare cu datele sale generale. Legatura **1:1** ajuta la mentinerea constrangerii pentru a nu aparea inregistrari multiple si diferite pentru un singur animal. Aceasta s-a realizat prin setarea drept cheie primara si unica a “ID_Animals”.

➢ One-To-Many (1 : n):

Entitatea **“Animals”** este entitatea principala care contine toate animalele pe baza carora sunt create si au sens si celelalte entitati. Aceasta detine o relatie **1:n** cu entitatile **“Animal_Type”** si **“Animal_Pen”**. Mai multe animale pot fi de acelasi tip (canin, felin,rozator) si mai multe animale pot sta in acelasi tarc, dar un animal nu poate sta in >=2 tarcuri simultan.

Entitatea **“Medical_Info”** poate avea mai multe inregistrari cu acelasi animal din entitatea **“Animals”**, dar un animal nu poate fi repartizat decat la un singur cabinet, cel al gradinii zoologice. Astfel se implementeaza o relatie **1:n**.

➢ Many-To-Many (n : n) :

Entitatea **“Diet”** creeaza o legatura **n:n** cu entitatea **“Animals”** din care preia ID_Animals pentru identificare si cu entitatea**“Food”** de unde extrage tipurile de mancare cu ID_Food . Un animal poate aparea in mai multe inregistrari din entitata “Diet” si un tip de macarea poate fi mancat de mai multe animale. Deci entitatea “Diet” este intermediara in vederea stabilirii nutritiei animalelor.

3. Descrierea constrangerilor :

Constrangerile de orice tip (not null, check, primary key etc) sunt necesare pentru a asigura integritatea si corectitudinea datelor introduse.

❖ Animals:

Primary Key: id_animals
Foreign Keys: id_animal_type (animal_type) si id_animal_pen (animal_pen)

❖ Animal_Info:

Primary Key: id_animals (animals)
Foreign Keys: id_animals (animals)
Constrangere Unique:id_animals (animals)
Constrangere check:

1. name (regexp_like(Name, '^[a-zA-Z_]*$’) -> numele nu trebuie sa contina cifre sau caractere speciale precum acolade etc. Numele poate contine spatii
2. Gender (lista de valori <’Male’, ‘Female’>) -> sunt doua tipuri , in acord cu realitatea, de oameni pe acest criteriu

❖ Animal_Pen:

Primary Key: id_animal_pen
Constrangere Unique: name_pen -> numele tarcului trebuie sa fie unic pentru o identificare mai usoara a zonelor locuibile si specifice pentru animale si tipurile lor

Constrangere check: name_pen (regexp_like (Name_Pen, '^[a-zA-Z_]*$')) -> numele nu trebuie sa contina caractere speciale care sa duca la crearea unui nume de tarc irelevant

❖ Animal_Type:

Primary Key: id_animal_type
Constrangere Unique: type_animal -> exista cateva categorii de animale privite in ansamblul lor, de aceea datele introduse vor fi unice. Daca exista tipul canin, nu mai e nevoie de o alta definire a sa
Constrangere check: type_animal -> lista valori (<'Bird', 'Canin', 'Rodent' etc>)

❖ Medical_Info:

Primary Key: id_medical_info
Constrangere check: Children -> lista de valori(<'Yes','No'>) -> raspuns care ajuta la determinarea posibilelor probleme de sanatate

❖ Food:

Primary Key:id_food
Constrangere check:food_type (regexp_like (Food_Type, '^[a-zA-Z_]*$')) -> mancarea nu trebuie sa contina numere sau alte caractere speciale

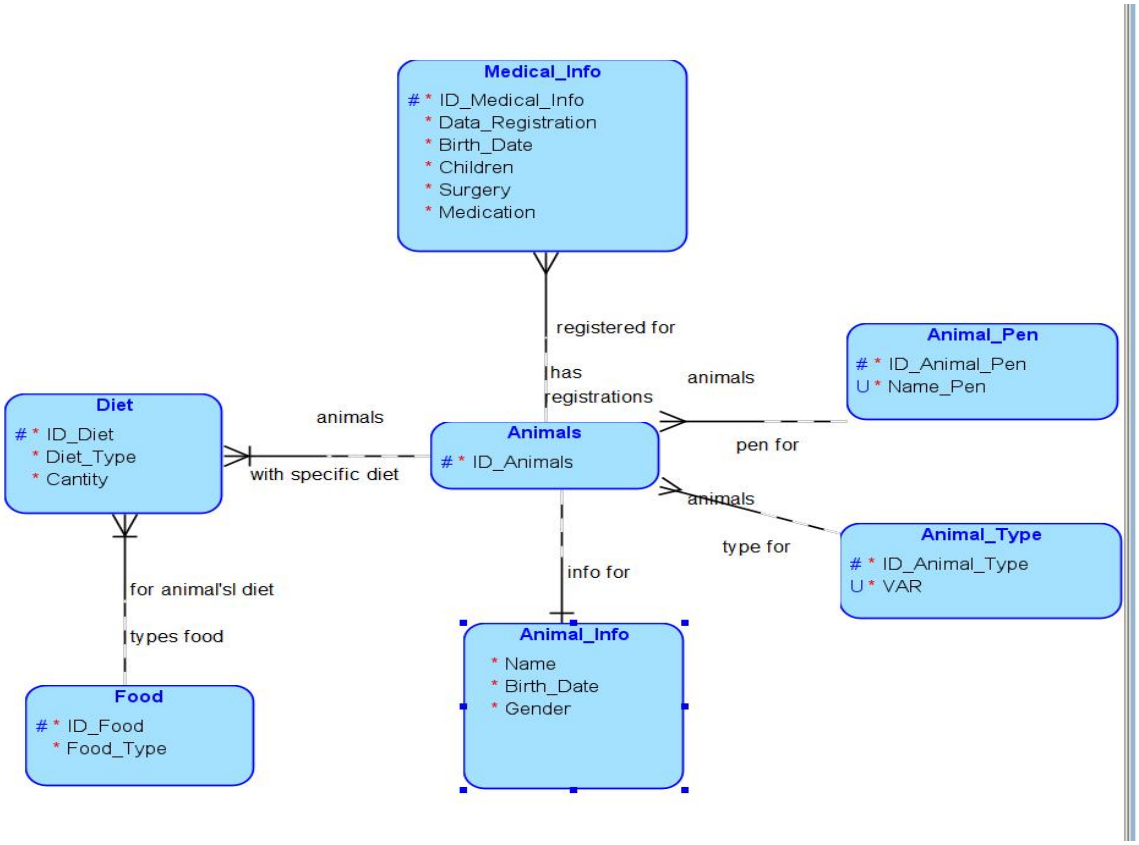
❖ Diet :

Primary Key: id_diet
Foreign Keys: id_food (food), id_animals (animals)
Constrangere check:

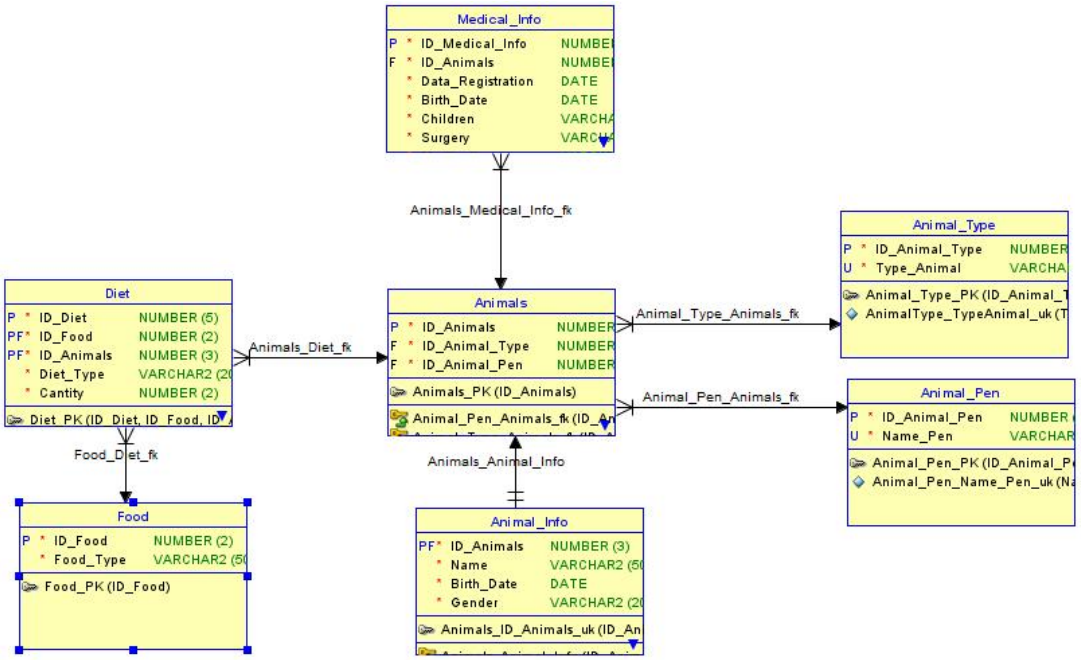
1. diet_type -> lista de valori (<'Carnivor','Omnivor','Erbivor'>)
   -> sunt 3 tipuri care cuprind particularitati care ajuta veterinarul in privinta consultatiilor
2. Cantity (regexp_like (Cantity,'[+-]?([0-9]*[.])?[0-9]+')) ->
   cantitatea trebuie sa contina doar numere. Reprezinta numarul de bucati mancate de animal intr-o zi.

4. Descrierea constrangerilor :

Modelul logic:



Modelul relational:



5. Normalizare:

Prima forma normala:

Toate tabelele respectă condițiile primei forme normale:

- un atribut conține valori atomice din domeniul său (și nu grupuri de astfel devalori)
- nu conține grupuri care se repetă

Exemplu:

Pentru tabela “Diet”: un animal poate manca mai multe tipuri de mancare, insa pentru fiecare tip nou (preluat din tabela “Food”) , se creaza o noua inregistrare cu acelasi animal si noul tip de mancare .

A doua forma normala:

Toate tabelele respectă condițiile celei de a doua forme normale:

- este prima formă normală
- toate atributele non
-cheie depind de toate cheile candidat

Exemplu:

Tabela “Animals” are doar un camp cheie primara, la fel ca celelalte tabele -> ID_Animals.

A treia forma normala:

Toate tabelele respectă a treia formă normală:

- este în a doua formă normală
- toate atributele non-cheie sunt direct (non-tranzitiv) dependențe de toate cheile candidat

Exemplu:

Tabela “Animals” contine cheia primara din tabela “Animal_Type”. Deci cu ajutorul foreign key-ului de la cea de-a doua tabela mentionata, tipul animalului va fi preluat.