University of British Columbia, Department of Computer Science

CPSC 304

Cover Page for Project Part "Project Design (E/R Diagram, Schemas, Normalization)"

Date: Oct 14, 2018

Group Members:

Name	Student Number	CS Userid	Tutorial Section	Email Address
Sophia Shen	14747159	q8s0b	T1E	sophiashenziyi@gmail.com
Owen Tsai	26515155	f6c1b	T1G	tsaiyicheng3@gmail.com
Charlotte Zhu	53587151	r3s0b	T1A	coco99166@outlook.com
Brandon Djokic	26172056	t7s6	T1A	btdjokic@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

2. Relational Model - NOTE: IF YOU CHANGE ANYTHING HERE, PLEASE CHANGE THE FINALIZED TABLES AS WELL

PK: underlined
FK: **Bold**

Animal (name : CHAR(20), animal_id : INTEGER, species : CHAR(20), sex: CHAR(20), age: INTEGER, height : INTEGER, weight : INTEGER, eat_freq_week : INTEGER, eat_amount : INTEGER, **enclosure_id**: INTEGER)
FK enclosure_id REFERENCES Habitat. enclosure_id not null

Employee(f_name : CHAR(20), l_name : CHAR(20), employee_id : INTEGER, pay : INTEGER, walkeetalkee_num : INTEGER, **zoo_address**: CHAR(50)
FK zoo_address references Employee. Zoo_address not null

Keeper(**employee_id**: INTEGER, duty: CHAR(20))
FK employee_id references Employee

Trainer(**id_employee**: INTEGER, specialty : CHAR(20))
FK employee_id references Employee

Site(site_id: INTEGER, location: CHAR(20), used_for: CHAR(20), **zoo_address**: CHAR(50))
FK zoo_address references Zoo

Food(food_id : INTEGER, stock_serving : INTEGER, date_purch : DATE, date_expiry: DATE, **site_id**: INTEGER)
Site_id references Site. site_id not null

Habitat(enclosure_id: INTEGER, biome: CHAR(20), temp: INTEGER, humidity: INTEGER, sq_ft: INTEGER, depth: INTEGER, **site_id**: INTEGER)
site_id references Site. site_id not null

Zoo(name : CHAR(20), phone : INTEGER, address: CHAR(50), city : CHAR(20), country : CHAR(20))
Alt Key: phone

Show(start_time: INTEGER, duration: INTEGER, name: CHAR(20), type: CHAR(20), **site_id**: INTEGER)
site_id references Site. site_id not null

Performs(**start_time**: INTEGER, **show_name**: CHAR(20) , **employee_id**: INTEGER, **animal_id**: INTEGER, role: CHAR(50))
FK start_time and show_name references Show, animal_id references Animal, employee_id references Employee

CaresFor(**employee_id**: INTEGER, **animal_id**: INTEGER)
FK employee_id references Keeper, animal_id references Animal)

Trains(**employee_id**: INTEGER, **animal_id**: INTEGER, skill: CHAR(50))
FK employee_id references Trainer, animal_id references Animal)

Trades(**zoo_from_address**: CHAR(50) , **zoo_to_address**: CHAR(50), **animal_id**: INTEGER, date: DATE)
FK zoo_from_address references Zoo, zoo_to_address references Zoo, animal_id references Animal

Eats(**animal_id**: INTEGER, **food_id**: INTEGER)
FK animal_id references Animal, food_id references Food

3 + 4. Functional Dependencies & Tables

Animal
animal_id → *name, species, sex, age, height, weight, eat_freq_week, eat_amount, enclosure_id*

Employee
employee_id → *f_name, l_name, pay, walkeetalkee_num,* ***zoo_address***
f_name, l_name→ walkeetalkee_num
Employee(employee_id, f_name, l_name, pay, zoo_address)
EmployeeCommunication(f_name, l_name, walkeetalkeeno)

Keeper
employee_id → *duty*

Trainer
employee_id → *specialty*

Site
site_id → *location, used_for, zoo_address*

Food
food_id → *stock_serving, date_purch, date_expiry, site_id*

Habitat
enclosure_id → *biome, temp, humidity, sq_ft, depth, site_id*
Biome → temperature, humidity
Habitat(enclosure_id, biome, sq_ft, depth, id_number)
HabitatBiome(biome, temperature, humidity)

Zoo
address → *name, phone, city, country*

Show
Start time, name → *duration, type, location, site_id*

Performs, CaresFor, Trains, Trades, Eats
No unique FDs (the minimal key determines everything)

FINALIZED TABLES (AFTER DECOMPOSITION):

Animal (name : CHAR(20), animal_id : INTEGER, species : CHAR(20), sex: CHAR(20), age: INTEGER, height : INTEGER, weight : INTEGER, eat_freq_week:INTEGER, eat_amount:INTEGER, **enclosure_id**: INTEGER)
FK enclosure_id REFERENCES Habitat. enclosure_id not null

Employee(**f_name** : CHAR(20), **l_name** : CHAR(20), employee_id : INTEGER, pay : INTEGER, **zoo_address**: CHAR(50))
FK zoo_address references Zoo. Zoo_address not null
FK f_name, l_name references EmployeeCommunication. F_name, l_name not null

EmployeeCommunication(f_name:CHAR(20), l_name:CHAR(20), walkeetalkee_num : INTEGER)

Keeper(**employee_id**: INTEGER, duty: CHAR(20))
FK employee_id references Employee

Trainer(**employee_id**: INTEGER, specialty : CHAR(20))
FK employee_id references Employee

Site(site_id: INTEGER, location: CHAR(20), used_for: CHAR(20), **zoo_address**: CHAR(50))
FK zoo_address references Zoo

Food(food_id : INTEGER, stock_serving : INTEGER, date_purch : DATE, date_expiry: DATE, **site_id**: INTEGER)
site_id references Site. site_id not null

Habitat(enclosure_id: INTEGER, **biome**: CHAR(20), sq_ft: INTEGER, depth: INTEGER, **site_id**: INTEGER)
site_id references Site. site_id not null
Biome references HabitatBiome. Biome not null

HabitatBiome(biome: CHAR(20), temp: INTEGER, humidity: INTEGER)
site_id references Site. site_id not null

Zoo(name : CHAR(20), phone : INTEGER, address: CHAR(50), city : CHAR(20), country : CHAR(20))

Show(start_time: INTEGER, duration: INTEGER, name: CHAR(20), type: CHAR(20), **site_id**: INTEGER) *site_id references Site. site_id not null*

Performs(**start_time**: INTEGER, show_name: CHAR(20), **employee_id**: INTEGER, **animal_id**: INTEGER, role: CHAR(50))
FK start_time and show_name references Show, animal_id references Animal, employee_id references Employee

CaresFor(**employee_id**: INTEGER, **animal_id**: INTEGER)
FK employee_id references Keeper, animal_id references Animal)

Trains(**employee_id**: INTEGER, **animal_id**: INTEGER, skills: CHAR(50))
FK employee_id references Trainer, animal_id references Animal)

Trades(**zoo_from_address**: CHAR(50) , **zoo_to_address**: CHAR(50), **animal_id**: INTEGER, date: INTEGER)
FK zoo_from_address references Zoo, zoo_to_address references Zoo, animal_id references Animal

Eats(**animal_id**: INTEGER, **food_id**: INTEGER)
FK animal_id references Animal, food_id references Food

5. SQL DDLs - DIRECTLY UPDATED ON SQL FILE:

```
create table zoo(
    address varchar(40),
    name char(40) not null,
    phone int,
    city char(20),
    country char(20),
    primary key (address)
    );

create table site(
    site_id int,
    location char(20),
    used_for char(20),
    zoo_address varchar(40),
    primary key (site_id),
    foreign key (zoo_address) references zoo ON DELETE CASCADE
    );

create table habitatbiome(
    biome char(20),
    temp int,
    humidity int,
    primary key (biome)
    );

create table habitat(
    enclosure_id int,
    biome char(20),
    sq_ft int,
    depth int,
    site_id int not null,
    primary key (enclosure_id),
    foreign key (biome) references habitatbiome,
    foreign key (site_id) references site ON DELETE CASCADE
    );

create table animal(
    animal_id int,
    name char(20),
    age int,
    sex char(20),
    height int,
    weight int,
    species char(20),
    eat_freq_week int,
    eat_amount int,
    enclosure_id int,
    primary key (animal_id),
    foreign key (enclosure_id) references habitat ON DELETE SET NULL
    );

create table employeecommunication(
    f_name char(20),
    l_name char(20),
    walkeetalkeeno int,
    primary key (f_name, l_name)
    );

create table employee(
    f_name char(20),
    l_name char(20),
    employee_id int,
    pay int,
    zoo_address varchar(40) not null,
    primary key (employee_id),
    foreign key (zoo_address) references zoo ON DELETE SET NULL,
    foreign key (f_name, l_name) references employeecommunication
    );

create table keeper(
    duty char(20),
    employee_id int,
    primary key (employee_id),
    foreign key (employee_id) references employee ON DELETE CASCADE
    );

create table trainer(
    speciality char(20),
    employee_id int,
    primary key (employee_id),
    foreign key (employee_id) references employee ON DELETE CASCADE
    );


create table food(
    food_id int,
    name char(20),
    stock_serving int,
    date_purchased date,
    date_expired date,
    site_id int not null,
    primary key (food_id),
    foreign key (site_id) references site ON DELETE CASCADE
    );


create table show(
    start_time char(8),
    duration int,
    name char(20),
    type char(20),
    site_id int not null,
    primary key (start_time, name),
    foreign key (site_id) references site ON DELETE CASCADE
    );

create table performs(
    start_time char(8),
    show_name char(20),
    employee_id int,
    animal_id int,
    role char(50),
    primary key (start_time, employee_id, animal_id),
    foreign key (start_time, show_name) references show,
    foreign key (employee_id) references employee,
    foreign key (animal_id) references animal
    );

create table caresfor(
    employee_id int,
    animal_id int,
    primary key (employee_id, animal_id),
    foreign key (employee_id) references employee,
    foreign key (animal_id) references animal
    );

create table trains(
    employee_id int,
    animal_id int,
    skills char(50),
    primary key (employee_id, animal_id),
    foreign key (employee_id) references employee,
    foreign key (animal_id) references animal
    );

create table trades(
    zoo_from_address varchar(40),
    zoo_to_address varchar(40),
    animal_id int,
    trade_date date,
    primary key (zoo_from_address, zoo_to_address, animal_id),
    foreign key (zoo_from_address) references zoo,
    foreign key (zoo_to_address) references zoo,
    foreign key (animal_id) references animal
    );

create table eats(
    animal_id int,
    food_id int,
    primary key (animal_id, food_id),
    foreign key (animal_id) references animal ON DELETE CASCADE,
    foreign key (food_id) references food
    );
```