University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 1 March 2024

Group Number: 39

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Hubert Wong	82570367	h8f3h	ycwonghubert@gmail.com
Sunny Lau	45195864	g8m3i	lausunny@student.ubc.ca
Veronica Leung	43477207	y0v1e	veronicaxlcw@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 the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

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

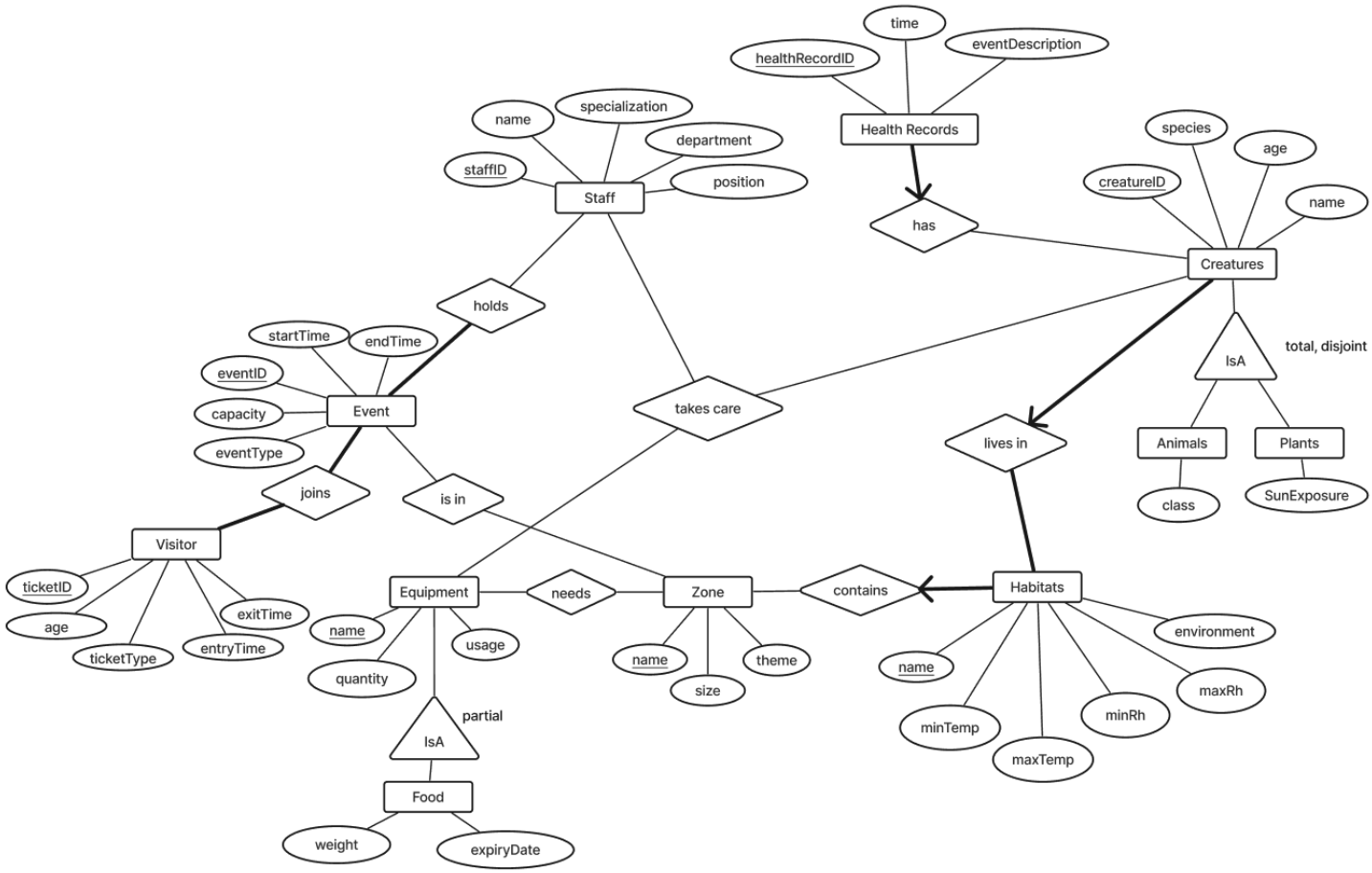
University of British Columbia, Vancouver
Department of Computer Science

---

1. Summary of the project

   This is a database for Botanical Park and Zoo Operations, which allows users to retrieve information about creatures in the botanical park and zoo, operations of the park and visitor statistics for data analysis.

2. ER Diagram



Remarks: The following changes have been made to the ER diagram.

- ISA constraints have been added to the diagram as advised.

- The attributes in the Habitat entity set have been changed from {<u>name</u>, temperature range, humidity range, environment} to {<u>name</u>, minTemp, maxTemp, minRh, maxRh, environment} for better data representation.
- In the Visitor entity set, new attributes *age* and *ticketType* are added.
- In the Staff entity set, new attributes *department* and *position* are added.
- In the Event entity set, a new attribute *eventType* has been added.
- The total participation constraint for Staff in Event has been removed, as not all the staff hold events.

3. Schema Derived from ER Diagram

   Below are the schema derived from ER diagram, with the underlined attributes as Primary Key, bolded attributes as Foreign Key.

   - Staff(<u>staffID: INTEGER</u>, name: VARCHAR(50), specialization: VARCHAR(50), position: VARCHAR(50), department: VARCHAR(50))
     - Name, specialization, position, department NOT NULL
     - Candidate Key: {staffID}
   - Event(<u>eventID: INTEGER</u>, startTime: DATETIME, endTime: DATETIME, capacity: INTEGER, eventType: ENUM('promotion', 'private party', 'holiday special', 'fundraising', 'conference'))
     - startTime, endTime, capacity, eventType NOT NULL
     - Candidate Key: {eventID}
   - Holds(**<u>staffID: INTEGER, eventID: INTEGER</u>**)
     - Candidate Key: {staffID, eventID}
   - Visitor(<u>ticketID: INTEGER</u>, entryTime: DATETIME, exitTime: DATETIME, age: INTEGER, ticketType: ENUM("Child", "Teen", "Adult", "Senior"))
     - age, ticketType NOT NULL
     - entryTime, exitTime can be NULL, as people can purchase entry tickets for future date
     - Candidate Key: {ticketID}
   - Joins(**<u>ticketID: INTEGER, eventID: INTEGER</u>**)
     - Candidate Key: {ticketID, eventID}
   - Zone(<u>name: VARCHAR(50)</u>, size: INTEGER, theme: VARCHAR(50))
     - size, theme NOT NULL
     - theme: UNIQUE
     - Candidate Key: {name}, {theme}
   - IsIn(**<u>eventID: INTEGER, zoneName: VARCHAR(50)</u>**)
     - Candidate Key: {eventID, zoneName}

    - zoneName is a Foreign Key which references the attribute name in Table Zone
- Equipment(name: VARCHAR(50), quantity: INTEGER, usage: VARCHAR(50))
    - Candidate Key: {name}
- Food(**name: VARCHAR(50)**, weight: FLOAT, expiryDate: DATE)
    - expiryDate NOT NULL
    - Candidate Key: {name}
- Needs(**zoneName: VARCHAR(50), equipmentName: VARCHAR(50)**)
    - Candidate Key: {zoneName, equipmentName}
    - zoneName is a Foreign Key which references the attribute name in table Zone
    - equipmentName is a Foreign Key which references the attribute name in table Equipment
- HasHealthRecord(healthRecordID: INTEGER, time: DATETIME, eventDescription: VARCHAR(100), **creatureID: INTEGER**)
    - time, eventDescription, creatureID is NOT NULL
    - Candidate Key: {healthRecordID}
- CreaturesLivesIn(creatureID: INTEGER, species: VARCHAR(50), age: INTEGER, name: VARCHAR(50), class: VARCHAR(50), sunExposure: ENUM("FullSun", "PartSun", "PartShade", "FullShade") , **habitatName: VARCHAR(50)**)
    - species, age, name, habitatName NOT NULL
    - *class* is NULL for Plants, *sunExposure* is NULL for Animal
    - Candidate Key: {creatureID}
    - habitatName is a Foreign Key which references the attribute name in table HabitatsContained
    - Remarks: *Animals* and *Plants* are not modelled into two separate tables because Creatures participate in the relations *Has*, *LivesIn* and *TakesCare*. Moreover, each ISA entity has one to two fields only, so not much spaces will be wasted on the NULL values.
- HabitatsContained(name:VARCHAR(50), minTemp: FLOAT, maxTemp: FLOAT, minRh: INTEGER, maxRh: INTEGER, environment: VARCHAR(50), **zoneName: VARCHAR(50)**)
    - minTemp, maxTemp, minRh, maxRh, environment, zoneName NOT NULL
    - Candidate Key: {name}
    - zoneName is a Foreign Key which references the attribute name in Table Zone
- TakesCare(**staffID: INTEGER, equipmentName: VARCHAR(50), creatureID: INTEGER**)
    - Candidate Key: {staffID, equipmentName, creatureID}
    - equipmentName is a Foreign Key which references the attribute name in table Equipment

4. Functional Dependencies

   Primary keys and foreign keys remain unchanged. Hence, they are not stated explicitly in this section.

   Staff

   staffID → name, specialization, position, department

   specialization → department

   position → department

   Candidate Key: {staffID}

   Event

   eventID → startTime, endTime, capacity, eventType

   eventType → capacity

   Candidate Key: {eventID}

   Holds

   Since there are no non-key attributes, there are no non-trivial FDs.

   Candidate Key: {staffID, eventID}

   Visitor

   ticketID → entryTime, exitTime, age, ticketType

   age → ticketType

   Candidate Key: {ticketID}

Joins

Since there are no non-key attributes, there are no non-trivial FDs.

Candidate Key: {ticketID, eventID}

Zone

name → size, theme

theme → name, size

Candidate Key: {name}, {theme}

IsIn

Since there are no non-key attributes, there are no non-trivial FDs.

Candidate Key: {eventID, zoneName}

Equipment

name → quantity, usage

Candidate Key: {name}

Food

name → weight, expiryDate

Candidate Key: {name}

Needs

Since there are no non-key attributes, there are no non-trivial FDs.

Candidate Key: {zoneName, equipmentName}

HasHealthRecord

healthRecordID → time, eventDescription, creatureID

Candidate Key: {healthRecordID}

CreaturesLivesIn

creatureID → species, age, name, class, sunExposure, habitatName

name, age, species → creatureID, class, sunExposure, habitatName

Candidate Key: {creatureID}

HabitatsContained

name → minTemp, maxTemp, minRh, maxRh, environment, zoneName

Candidate Key: {name}

TakesCare

Since there are no non-key attributes, there are no non-trivial FDs.

Candidate Key: {staffID, equipmentName, creatureID}

5. Normalization

<u>BCNF Decomposition: Staff</u>

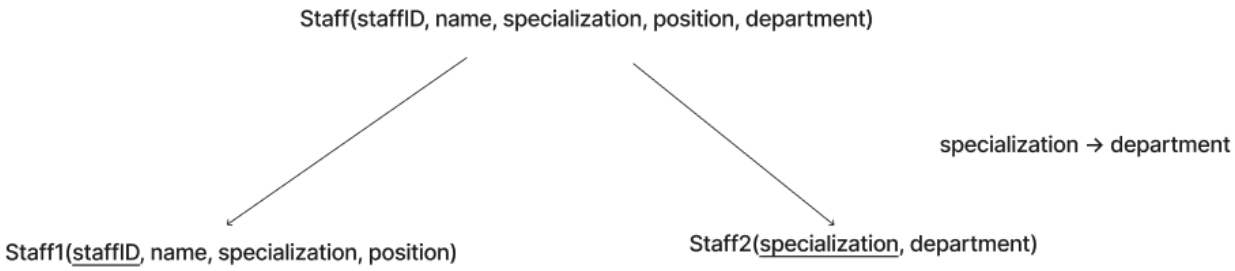
1. The relation *Staff* is not in BCNF.

   $\{specialization\}^+ = \{specialization, department\}$

   $\{position\}^+ = \{position, department\}$

   FDs *specialization* → *department* and *position* → *department* violate BCNF.

2. We decompose on *specialization* → *department*:



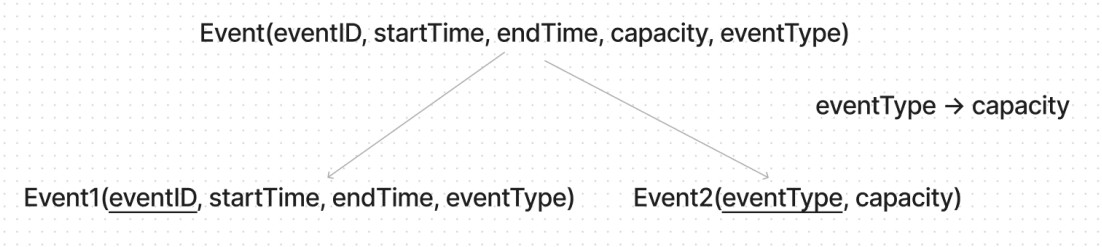
<u>BCNF Decomposition: Event</u>

1. The relation *Event* is not in BCNF.

   $\{eventType\}^+ = \{eventType, capacity\}$

   FD *eventType* → *capacity* violates BCNF.

2. We decompose on *eventType* → *capacity*:



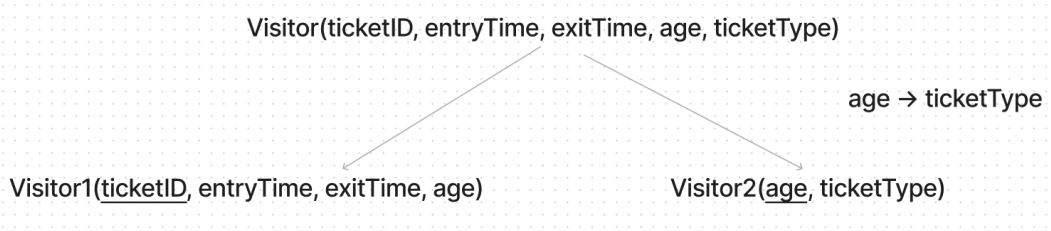
<u>BCNF Decomposition: Visitor</u>

1. The relation *Visitor* is not in BCNF.

   $\{age\}^+ = \{age, ticketType\}$

   FD age ⟶ ticketType violates BCNF.

2. We decompose on *age ⟶ ticketType*



Other relations with FDs of the form X ⟶ b have X as the superkey of each relation respectively. Hence, they are all in BCNF.

Below is a list of tables after normalization, with the underlined attributes as Primary Key, bolded attributes as Foreign Key.

- Staff1(staffID: INTEGER, name: VARCHAR(50), **specialization: VARCHAR(50)**, position: VARCHAR(50))
  - Candidate Key: {staffID}
- Staff2(specialization: VARCHAR(50), department: VARCHAR(50))
  - Candidate Key: {specialization}
- Event1(eventID: INTEGER, startTime: DATETIME, endTime: DATETIME, **eventType: ENUM('promotion', 'private party', 'holiday special', 'fundraising', 'conference')**)
  - Candidate Key: {eventID}
- Event2(eventType: ENUM('promotion', 'private party', 'holiday special', 'fundraising', 'conference'), capacity: INTEGER)
  - Candidate Key: {eventType}
- Holds(**staffID: INTEGER, eventID: INTEGER**)
  - Candidate Key: {staffID, eventID}
- Visitor1(ticketID: INTEGER, entryTime: DATETIME, exitTime: DATETIME, **age: INTEGER**)
  - Candidate Key: {ticketID}
- Visitor2(age: INTEGER, ticketType: ENUM("Child", "Teen", "Adult", "Senior"))
  - Candidate Key: {age}
- Joins(**ticketID: INTEGER, eventID: INTEGER**)
  - Candidate Key: {ticketID, eventID}

- Zone(name: VARCHAR(50), size: INTEGER, theme: VARCHAR(50))
    - Candidate Key: {name}, {theme}
- IsIn(**eventID: INTEGER, zoneName: VARCHAR(50)**)
    - Candidate Key: {eventID, zoneName}
- Equipment(name: VARCHAR(50), quantity: INTEGER, usage: VARCHAR(50))
    - Candidate Key: {name}
- Food(**name: VARCHAR(50)**, weight: FLOAT, expiryDate: DATE)
    - Candidate Key: {name}
- Needs(**zoneName: VARCHAR(50), equipmentName: VARCHAR(50)**)
    - Candidate Key: {zoneName, equipmentName}
- HasHealthRecord(healthRecordID: INTEGER, time: DATETIME, eventDescription: VARCHAR(100), **creatureID: INTEGER**)
    - Candidate Key: {healthRecordID}
- CreaturesLivesIn(creatureID: INTEGER, species: VARCHAR(50), age: INTEGER, name: VARCHAR(50), class: VARCHAR(50), sunExposure: Enum (“FullSun”, “PartSun”, “PartShade”, “FullShade”) , **habitatName: VARCHAR(50)**)
    - Candidate Key: {creatureID}
- HabitatsContained(name:VARCHAR(50), minTemp: FLOAT, maxTemp: FLOAT, minRh: INTEGER, maxRh: INTEGER, environment: VARCHAR(50), **zoneName: VARCHAR(50)**)
    - Candidate Key: {name}
- TakesCare(**staffID: INTEGER, equipmentName: VARCHAR(50), creatureID: INTEGER**)
    - Candidate Key: {staffID, equipmentName, creatureID}

6. SQL DDL Statement

```
CREATE TABLE Staff1 (

    staffID INTEGER PRIMARY KEY,

    name VARCHAR(50) NOT NULL,

    specialization VARCHAR(50) NOT NULL,

    Position VARCHAR(50) NOT NULL,

    FOREIGN KEY specialization REFERENCES Staff2(specialization)
```

```
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);

CREATE TABLE Staff2 (
    specialization VARCHAR(50) PRIMARY KEY,
    department VARCHAR(50),
);

CREATE TABLE Event1 (
    eventID INTEGER PRIMARY KEY,
    startTime DATETIME NOT NULL,
    endTime DATETIME NOT NULL,
    eventType ENUM('promotion', 'private party', 'holiday special', 'fundraising',
'conference') NOT NULL,
    FOREIGN KEY eventType REFERENCES Event2(eventType)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);

CREATE TABLE Event2 (
    eventType ENUM('promotion', 'private party', 'holiday special', 'fundraising',
'conference') PRIMARY KEY,
    capacity INTEGER NOT NULL
```

```
);


CREATE TABLE Holds (
        staffID INTEGER,
        eventID INTEGER,
        PRIMARY KEY (staffID, eventID),
        FOREIGN KEY (staffID) REFERENCES Staff1(staffID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (eventID) REFERENCES Event1(eventID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
);


CREATE TABLE Visitor1 (
        ticketID INTEGER PRIMARY KEY,
        entryTime DATETIME,
        exitTime DATETIME,
        age INTEGER NOT NULL,
        FOREIGN KEY age REFERENCES Visitor2(age)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
);
```

```
CREATE TABLE Visitor2 (
        age INTEGER PRIMARY KEY,
        ticketType ENUM(“Child”, “Teen”, “Adult”, “Senior”) NOT NULL
);

CREATE TABLE Joins (
        ticketID INTEGER,
        eventID INTEGER,
        PRIMARY KEY (ticketID, eventID),
        FOREIGN KEY (ticketID) REFERENCES Visitor(ticketID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (eventID) REFERENCES Event1(eventID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
);

CREATE TABLE Zone (
        name VARCHAR(50) PRIMARY KEY,
        size INTEGER NOT NULL,
        theme VARCHAR(50) UNIQUE NOT NULL
);
```

```
CREATE TABLE IsIn (
        eventID INTEGER,
        zoneName VARCHAR(50),
        PRIMARY KEY (zoneName, eventID),
        FOREIGN KEY (eventID) REFERENCES Event1(eventID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (zoneName) REFERENCES Zone(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
);

CREATE TABLE Equipment (
        name VARCHAR(50) PRIMARY KEY,
        quantity INTEGER,
        usage VARCHAR(50)
);

CREATE TABLE Food (
        name VARCHAR(50) PRIMARY KEY,
        weight FLOAT,
        expiryDate DATE NOT NULL,
```

```
        FOREIGN KEY (name) REFERENCES Equipment(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);
CREATE TABLE Needs (
        zoneName VARCHAR(50),
        equipmentName VARCHAR(50)
        PRIMARY KEY (zoneName, equipmentName),
        FOREIGN KEY (equipmentName) REFERENCES Equipment(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (zoneName) REFERENCES Zone(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
);

CREATE TABLE HasHealthRecord (
        healthRecordID INTEGER PRIMARY KEY,
        time DATETIME NOT NULL,
        eventDescription VARCHAR(100) NOT NULL,
        creatureID INTEGER NOT NULL,
        FOREIGN KEY (creatureID) REFERENCES CreaturesLivesIn(creatureID)
                ON DELETE NO ACTION
```

```
        ON UPDATE CASCADE,
);

CREATE TABLE CreaturesLivesIn (
    creatureID INTEGER PRIMARY KEY,
    species VARCHAR(50) NOT NULL,
    age INTEGER NOT NULL,
    name VARCHAR(50) NOT NULL,
    class VARCHAR(50),
    sunExposure ENUM (“FullSun”, “PartSun”, “PartShade”, “FullShade”) ,
    habitatName VARCHAR(50) NOT NULL,
    FOREIGN KEY (habitatName) REFERENCES HabitatsContained(name)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);

CREATE TABLE HabitatsContained (
    name VARCHAR(50) PRIMARY KEY,
    minTemp FLOAT NOT NULL,
    maxTemp FLOAT NOT NULL,
    minRh INTEGER NOT NULL,
    maxRh INTEGER NOT NULL,
    environment VARCHAR(50) NOT NULL,
```

```
    zoneName VARCHAR(50) NOT NULL,
    FOREIGN KEY (zoneName) REFERENCES Zone(name)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);

CREATE TABLE TakesCare (
    staffID INTEGER,
    equipmentName VARCHAR(50),
    creatureID INTEGER,
    PRIMARY KEY (staffID, equipmentName, creatureID),
    FOREIGN KEY (staffID) REFERENCES Staff1(staffID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (equipmentName) REFERENCES Equipment(name)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (creatureID) REFERENCES CreaturesLivesIn(creatureID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
)
```

7. INSERT

   INSERT INTO Staff1

Values (101, “White Ross”, “Customer Relations”, “Customer Service”),

(102, “Alex Poon”, “Veterinary Medicine”, “Veterinarian”),

(103, “Allen Iverson”, “Zoology”, “Keeper”),

(104, “Taylar Swift”, “Customer Relations”, “Customer Service”),

(105, “Elon Musk”, “Horticulture”, “Gardener”)

(106, “Jasmine Kaur”, “Botany”, “Researcher”);

INSERT INTO Staff2

Values (“Customer Relations”, 'Visitor Services'),

(“Veterinary Medicine”, 'Veterinary Services'),

(“Zoology”, 'Animal Care'),

(“Horticulture”, 'Plant Care'),

(“Botany”, “Conservation”);

INSERT INTO Event1

Values (1, “2023-06-18 10:00:00”, “2023-06-18 17:00:00”, “promotion”),

(2, “2023-09-05 15:00:00”, “2023-09-05 17:00:00”, “fundraising”),

(3, “2023-12-24 15:00:00”, “2023-12-24 19:00:00”, “holiday special”),

(4, “2023-08-12 12:00:00”, “2023-08-12 17:00:00”, “private party”),

(5, “2023-07-18 15:00:00”, “2023-07-18 17:00:00”, “conference”),

(6, “2023-06-18 15:00:00”, “2023-06-18 17:00:00”, “conference”);

INSERT INTO Event2

Values (“promotion”, 1500),

(“private party”, 50),

(“holiday special”, 2000),

(“fundraising”, 1000),

(“conference”, 100);

INSERT INTO Holds

Values (101, 2), (102, 5), (103, 1), (104, 4), (105, 3), (106, 6), (101, 1);

INSERT INTO Visitor1

Values (1001, “2023-06-18 10:00:00”, “2023-06-18 13:00:00”, 6),

(1002, “2023-09-05 12:00:00”, “2023-09-05 18:00:00”, 70),

(1003, “2023-12-24 10:00:00”, “2023-12-24 19:30:00”, 18),

(1004, “2023-08-12 12:00:00”, “2023-08-12 17:00:00”, 18),

(1005, “2023-07-18 14:00:00”, “2023-07-18 19:00:00”, 45),

(1006, “2023-06-18 14:30:00”, “2023-06-18 17:30:00”, 38);

INSERT INTO Visitor2

Values (6, “child”),

(70, “senior”),

(18, “teen”),

(45, “adult”),

(38, “adult”);

INSERT INTO Joins

Values (1001, 1), (1002, 2), (1003, 3), (1004, 4), (1005, 5), (1006, 6);

INSERT INTO Zone

Values (“Wild Amazon”, 10000, “jungle”),

(“Arid Africa”, 8000, “Africa”),

(“Subtropical Asia”, 4800, “Asia”),

(“Warm Garden”, 5000, “Garden”),

(“Freezing Igloo”, 3000, ”Polar Region“);

INSERT INTO IsIn

Values(1, “Wild Amazon”), (1, “Arid Africa”), (1, “Subtropical Asia”), (3, “Freezing Igloo”), (4, “Warm Garden”), (2, “Warm Garden”);

INSERT INTO Equipment

Values(“Syringe”, 100, “injecting drugs”), (“Carrot”, 5, “food”), (“Fish”, 50, “food”), (“Meat”, 10, “food”), (“Fresh leave”, 1, “food”), (“Insect”, 100, “food”), (“Dissecting Kit”, 5, “dissecting plants”), (“Combs”, 10, “grooming furred animals”);

INSERT INTO Food

Values(“Carrot”, 1000, “2024-03-01”), (“Fish”, 2000, “2024-03-03”),

(“Meat”, 3000, “2024-03-08”), (“Fresh leave”, 500, “2024-03-03”),

(“Insect”, 200, “2024-04-01”);

INSERT INTO Needs

Values(“Fish”, “Freezing Igloo”), (“Meat”, “Subtropical Asia”)), (“Meat”, “Wild Amazon”), (“Meat”, “Arid Africa”), (“Fresh leave”, “Arid Africa”), (“Syringe”, “Freezing Igloo”);

INSERT INTO HasHealthRecord

Values (1, “2024-01-01”, “annual vaccination“, 10001),

(2, “2024-01-01”, “annual vaccination“, 10002),

(3, “2023-09-04”, “dying leaves”, 10004),

(4, “2024-01-04”, “egg laid on leaves”, 10003),

(5, “2023-10-28”, “treatment for a skin infection”, 10001),

(6, “2023-08-14”, “treatment for a flipper injury”, 10002);

INSERT INTO CreaturesLivesIn

Values

(10001, “Capuchin monkey”, 10, “Star”, “Mammalia”, NULL, “Subtropical”),

(10002, “Emperor penguin”, 4, “Pingu”, “Aves”, NULL, “Polar”),

(10003, “Atelopus spumarius harlequin frog”, 2, “Prince”, NULL, “Tropical”),

(10004, “Lysiana exocarpi”, 22, “Red Mistletoe”, “PartSun”, NULL, “Temporate”),

(10005, “Saguaro Cactus”, 17, “Big Saguaro”, “FullSun”, NULL, “Desert”);

INSERT INTO HabitatsContained

Values (“Tropical”, 22.0, 24.5, 80, 90, “land”, “Wild Amazon”),

(“Desert”, 28.5, 35.0, 22, 28, “sand”, “Arid Africa”),

(“Subtropical”, 15.0, 20.5, 60, 80, “land and lake”, “Subtropical Asia”),

(“Temporate”, 2, 12.5, 45, 70, “land”, “Warm Garden”),

(“Polar”, -50.0, -30.5, 75, 85, “ice and water”,“Freezing Igloo”);

INSERT INTO TakesCare

Values (102,”Syringe”, 10001),

(102, “Syringe”, 10002),

(103, “Fish”, 10002),

(103, “Meat”, 10001),

(105, “Dissecting Kit”, 10004);