Middle East Technical University

IS 503

Data Base Concepts and Applications

Term Project

Prepared By: Mert Mecit / 2149219

Table of Contents

List of Figures

Executive Summary

This project is done for Dr. Özden Özcan Top, under IS 503 - Data Base Concepts and Application course's term project. In this project, one of the biggest classes of the animal kingdom, mammals are examined. Within this database **real** 150+ scientists, 100+ families, 250+ genus, 300+ classes, 500+ region-animal relations and 500+ characteristic-animal relations are present. All of the entries are entered by me. I did so much integrity checking on data but ensuring 100% reliable is somewhat impossible in this project, so I used my best effort to detect anomalies. A total of 80+ working hours resulted in this 10+ tables and 1000+ entries huge database. All necessary data as well as pictures are provided from Wikipedia. I also used some information from Animaldiversity.com. Approach to build my database is SQL. Before coding in SQL, I developed a plan with an ER diagram. My ER diagram shows my rationale. Then, I presented how built by SQL Tables with the design I provided in my ER diagram. What that table tries to explain and how all of its attributes are decided on are explained in detail in this report. Then I proposed how I created and inserted my data, how one can update the data, how one can delete the data. At the end, I proposed two different queries to show how this database can be used.

1. Introduction

Mammals have been one of the most extent and interesting classes in animal kingdom. This is mainly both because it includes Homo Sapiens (humans) and it includes wide array of other interesting animals, from little rats to giant whales. First mammals, little rats, were living on the earth about 250 million years ago. After the millions of years of evolutionary progress and enormous number of mutation, thousands of different mammal types emerged. Today, we have about 6500 distinct species in mammals class.

This projects aims to create a huge database related to Mammals class. In this project, information about mammal class, its orders, families, genus, and species; their physical characteristics; their extinction status; their behaviors, as well as information about scientists that named those taxonomy levels are present.

SQL (Structured Query Language) is used to create this database as it offers a portable, faster, and multiple view available data processing.

2. Data, Methodology & Implementation

To create a database, I first introduce my ER diagram in the figure 1. Then I provide my tables that I built based on this ER model. All necessary data explanation, how they are found or calculated are separately explained in each table.

All necessary data is mainly provided from Wikipedia.com and animaldiversity.com. However, for some specific species, I used several sources from the internet such as edgeofexistence.org, researchgate.net, digimorph.org and so on.

2.1 ER Diagram

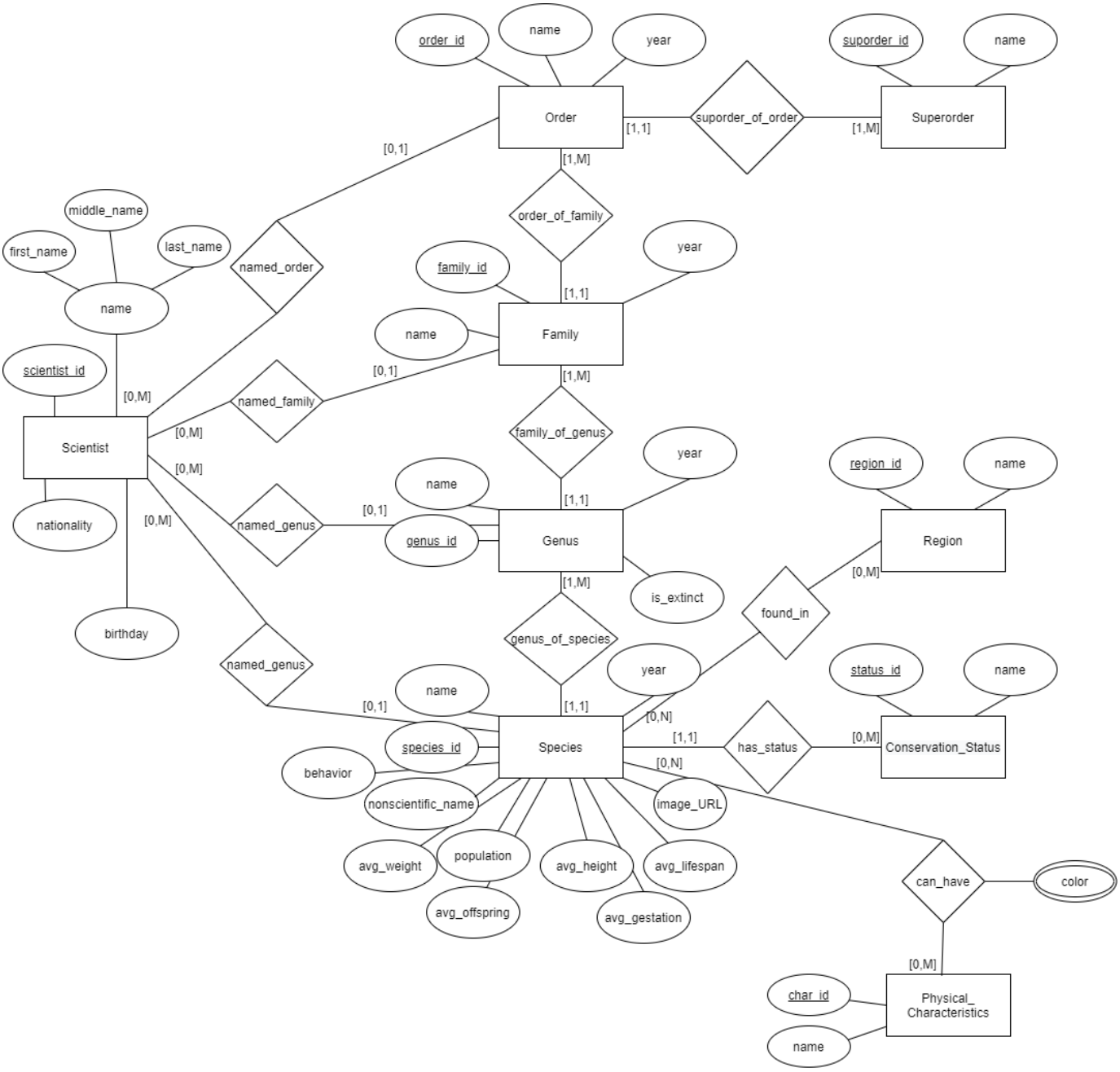

Figure 1. ER Diagram

Assumptions made in this ER diagram are as follows:

- Each order is under one specific suborder, each family is under one specific order and so on. However, each suborder can have several orders, each order can have several families and so on. That is why those kind of relationships have [1,1] and [1,M] relationships.
- Each scientist can name several order, families, genus, or species as well as they may not be named any. However, one assumption of our model is each of those taxonomy levels are named by at most one person (details are explained in Tables section) or it could be named anonymously.

- Species can be found in several regions or they cannot be found anywhere (they are extinct). One region can be home for zero to many number of different species.
- Each species have a certain conservation status. One conservation status can be found in zero to many number of species.
- Physical characteristics table is for unique characteristics. One species can have zero to many different kind of those characteristics as well as one characteristics can be found in zero to many different kind of species.

2.2 SQL Tables

From the ER diagram above, I have built 12 tables. All tables and their data explanations are explained in detail in this section.

2.2.1 Table "Scientist"

This table is for keeping information of scientists that named (and probably discovered order, family, genus, or species as well). For simplicity purposes, when two or more scientists worked on same animal, only one of them is considered. Composite attribute of name is divided into 3 columns as first, middle, and last name here. 191 scientists are present in my database.

Scientist_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

First_name = To keep first name of the scientist. Stored as VARCHAR.

Middle_name = To keep middle name of the scientist. If someone have a multiple middle names, one that is suggested by Wikipedia is used. If there was no suggested one, randomly one is picked. For some scientists, if I cannot retrieve full name, abbreviated middle name such as 'J' or 'M' is used. Stored as VARCHAR.

Last_name = To keep last name of the scientist. Stored as VARCHAR.

Nationality = To keep nationality of the scientist. Can be useful to identify which countries are the ones that contributed most to that area. Stored as CHAR with 3 letters by using Alpha-3 codes of the country.

Birthday = To keep birthday of the scientist. It can be useful to identify young discoverers as well as for fun queries such as finding whether the discoverer of a mammal is actually of that zodiac sign as zodiac signs have mammal names commonly: Aries, Taurus, Leo and so on. Stored as DATE and storing format is 'YYYY-MM-DD'.

One example entry:

Scientist_id	First_name	Middle_name	Last_name	Nationality	Birthday
7	Carl	NULL	Linnaeus	SWE	1707-05-23

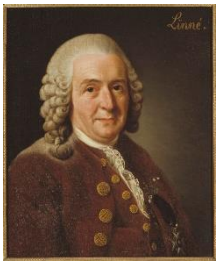

Figure 2. Carl Linnaeus

2.2.2 Table “Superorder”

This table is for keeping information of superorders. There are only 4 superorders exist in the mammals class and all are present in my database.

Superorder_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the superorder. Stored as VARCHAR.

One example entry:

Superorder_id	Name
3	Laurasiatheria

2.2.3 Table “Orderr”

This table is for keeping information of orders. Name is chosen as “Orderr” as order is a reserved word in SQL. There are 19 orders exist in the mammals class and all are present in my database.

Order_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the order. Stored as VARCHAR.

Year = To keep the year that order is found/named. Stored as INTEGER.

Named_scientist = To keep the ID of the scientist that is found/named this order. Stored as INTEGER and is a foreign key to the table Scientist.

Superorder = To keep the ID of the superorder that order belongs to. Stored as INTEGER and is a foreign key to the table Superorder.

One example entry:

Order_id	Name	Year	Named_scientist	Superorder
6	Sirenia	1811	4	1

Figure 3. Sirenia

2.2.4 Table “Family”

This table is for keeping information of orders. There are many families exist in the mammals class and 110 of them (generally the biggest ones and I tried to include all of the orders) are present in my database

Family_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the family. Stored as VARCHAR.

Year = To keep the year that family is found/named. Stored as INTEGER.

Named_scientist = To keep the ID of the scientist that is found/named this family. Stored as INTEGER and is a foreign key to the table Scientist.

Orderr = To keep the ID of the order that family belongs to. Stored as INTEGER and is a foreign key to the table Orderr.

One example entry:

Family_id	Name	Year	Named_scientist	Orderr
68	Talpidae	1814	37	14

Figure 4. Talpidae

2.2.5 Table "Genus"

This table is for keeping information of genus. There are many genus exist in the mammals class and 290 of them are present in my database. If there are less than or equal to 3 genus in a family in my database, I included all of them. If there are more than 3, I only included the 3 biggest one in terms of species amount.

Genus_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the genus. Stored as VARCHAR.

Year = To keep the year that genus is found/named. Stored as INTEGER.

Is_extinct = Shows whether the genus is completely extinct or not. Stored as TINYINT and 1 shows that genus is extinct.

Named_scientist = To keep the ID of the scientist that is found/named this genus. Stored as INTEGER and is a foreign key to the table Scientist.

Family = To keep the ID of the family that genus belongs to. Stored as INTEGER and is a foreign key to the table family.

One example entry:

Genus_id	Name	Year	Is_extinct	Named_scientist	Family
38	Stegodon	1847	1	71	12

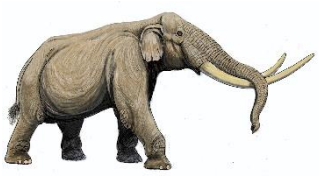

Figure 5. Stegodon

2.2.6 Table "Region"

This table is for keeping information of regions. I have divided regions by using figure below. Additionally, as some mammals are living in glaciers, I have added them and as Madagascar is home for distinct species, I have added it as a separate region. Total of 19 regions are in DB.

Region_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the region. Stored as VARCHAR.

One example entry:

Superorder_id	Name
16	Japan

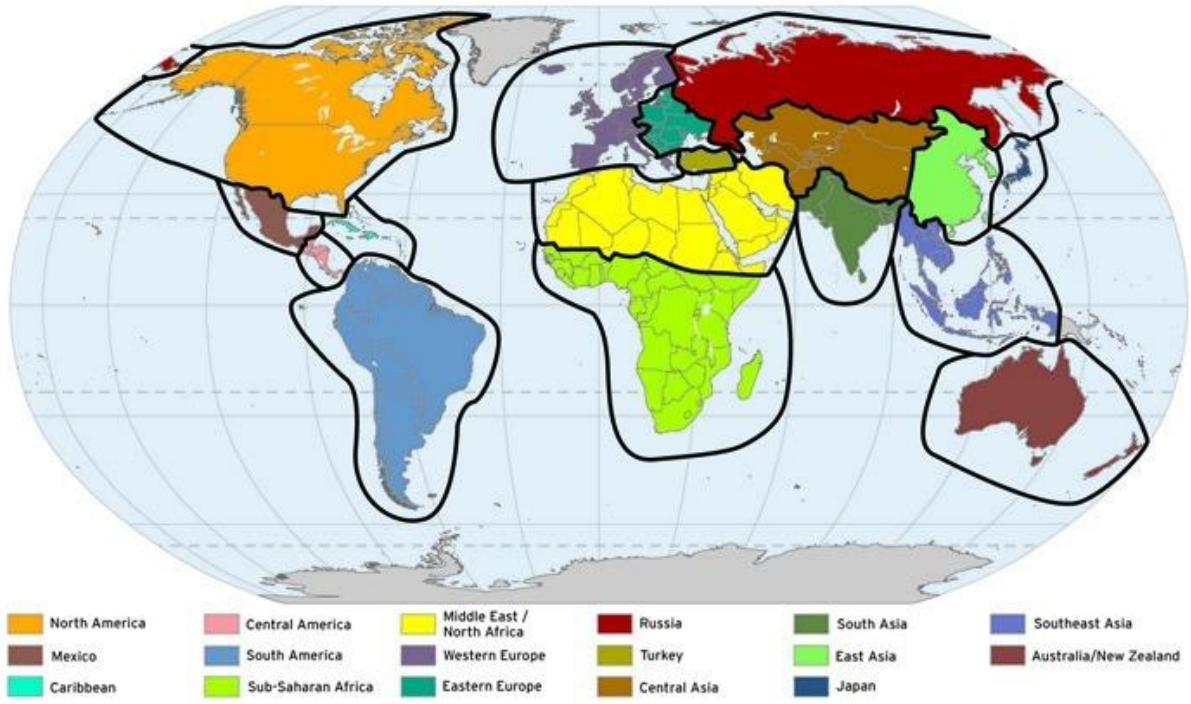

Figure 6. World Regions

2.2.7 Table "Conservation Status"

This table is for keeping conservation statuses. As there are many institutions that does conservation classification for mammals, I have used International Union for Conservation of Nature (IUCN) classification. It basically have 9 levels, from 1-7 it defines threat level, 1 being totally extinct and 7 being least concern, 8 means data deficient and 9 means not evaluated.

Status_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the status. Stored as VARCHAR.

One example entry:

Superorder_id	Name
3	Critically Endangered

2.2.8 Table "Species"

This table is for keeping information of species. There are many species exist in the mammals class and 328 of them are present in my database. If there are less than or equal to 3 species in a genus in my database, I included all of them. If there are more than 3, I only included the 3 biggest one in terms of information available. As getting information from extinct genus is very hard, many of them are not included in database, this is why species table is not approximately 3 times higher than family table.

Species_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the species. Stored as VARCHAR.

Nonscientific_name = This is for practical purposes. For example, 'Panthera Leo' is known as Lion for most of the people. Lion is nonscientific name here.

Image = An image URL is provided for every species. This is stored as a TEXT.

Year = To keep the year that species is found/named. Stored as INTEGER.

Avg_weight = To store the average weight of the species. If a correct average is provided, it is used. If I have only information about max and min weights, I have used a formula as follows:

$$0.25 * Weight_{max} + 0.75 * Weight_{min}$$

As generally upper limit tends to be a huge outlier. Weight is expressed in grams in my database and stored as DOUBLE.

Avg_height = To store the average height of the species. If a correct average is provided, it is used. If I have only information about max and min heights, I have used a formula as follows:

$$0.4 * Weight_{max} + 0.6 * Weight_{min}$$

As generally upper limit tends to be a slight outlier. Height is expressed in centimeters in my database and stored as DOUBLE.

Avg_lifespan = To store the average lifespan of the species. If only information for under captivity or in wild provided, its used. However, if both information are available, higher one is used. Lifespan is expressed in years and stored as DOUBLE.

Avg_offspring = To store the average offspring of the species. It basically means the mother gives birth to this number of children in each labor. Expressed in number and stored as DOUBLE.

Avg_gestation = To store the average gestation of the species. It basically means the mother gives birth to in this amount of time. Expressed in days and stored as DOUBLE.

Behavior = To store the key behavior of the given species. Expressed in keywords. For example, keyword 'arboreal' means the species live in trees. Those keywords are extracted from long texts of Wikipedia and Animaldiversity. Stored as TEXT.

Named_scientist = To keep the ID of the scientist that is found/named this species. Stored as INTEGER and is a foreign key to the table Scientist.

Status_id = To keep the status for the species. Stored as INTEGER and is a foreign key to the table Conservation Status.

Genus = To keep the ID of the genus that species belongs to. Stored as INTEGER and is a foreign key to the table genus.

One example entry:

Species_id	Name	Nonscientific_name	Image	Year	Avg_weight	Avg_height	Avg_lifespan
265	Felis Catus	Cat	https://upload...	1758	4750	76.2	14

Avg_offspring	Avg_gestation	Behavior	Genus	Status_id	Named_scientist
4	63	Scansorial, cursorial…	239	7	7

Figure 7. Cat

2.2.9 Table "Found In"

This table is for keeping species' locations. Species ID and region ID both constitute primary key here. As we have mentioned, a region can have none to many species as well as a species can have none (extinct) to many regions. A total of 505 entries are here.

Region_id = Identifier for region. Stored as an INTEGER.

Species_id = Identifier for species. Stored as an INTEGER.

One example entry:

Region_id	Species_id
17	243

Figure 8. Ursus Maritimus (Polar Bear) in Arctics

2.2.10 Table "Physical Characteristics"

This table is for keeping unique physical characteristics. By unique, I mean, conspicuous characteristics such as mane of a lion, fin of a dolphin, tail of a rat or hunch of a camel. Characteristics such as body hair (unless its very obvious like wool) are not included in that list. A total of 21 characteristics are present in my DB.

Char_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the characteristic. Stored as VARCHAR.

One example entry:

Char_id	Name
19	Fin

Figure 9. Shark fin

2.2.11 Table "Color"

As it is a multi-value attribute in my ER diagram, I needed to create a color entity. Most basic 22 colors are chosen for this purpose.

Color_id = Used as primary key and unique identifier in this table. Stored as INTEGER.

Name = To keep the name of the color. Stored as VARCHAR.

One example entry:

Color_id	Name
13	Cyan

Figure 10. Cyan Colored Impostor

2.2.12 Table "Can Have"

This table merges Species, Color, and Physical Characteristics table and creates a new entity. This table keeps information that a species can have a physical characteristics and also indicates its color as it can be different colors. I have a total of 599 entries here.

Species_id = Identifier for species. Stored as an INTEGER.

Char_id = Identifier for character. Stored as an INTEGER.

Color_id = Identifier for color. Stored as an INTEGER.

One example entry:

Species_id	Char_id	Color_id
173	11	6

Figure 11. Petaurista Elegans (Flying Squirrel) can have a brown patagium (wing like things)

2.3 SQL Queries

To get meaningful information related to my database, I have to perform SQL queries. First, I will introduce create, insert, update, and delete, then I will provide complex queries.

2.3.1 Create Queries

Beginning of the create queries and an example (family table) is provided to show how I used AUTO_INCREMENT function, primary key and foreign key constraints. All of the tables were created with a similar manner and complete create queries are provided in MySQL file.

```
DROP DATABASE Mammals;

CREATE DATABASE Mammals;

USE Mammals;

.

.

.

CREATE TABLE Family(
```

```
family_id INTEGER NOT NULL AUTO_INCREMENT,
name VARCHAR(30) NOT NULL,
year INTEGER,
order_id INTEGER NOT NULL,
named_scientist INTEGER,
PRIMARY KEY (family_id),
FOREIGN KEY (order_id) REFERENCES Orderr(order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
FOREIGN KEY (named_scientist) REFERENCES Scientist(scientist_id)
        ON UPDATE CASCADE,
        ON DELETE SET NULL
);
.
.
.
```

2.3.2 Insert Queries

To add data, I used insert queries instead of exporting from an excel file. As I have provided family table above, I will provide how I inserted data into it.

```
INSERT INTO Family(name, year, order_id, named_scientist) VALUES ('Macroscelididae', 1836, 1, 14),('Potamogalidae', 1865, 2, 15)… ,('Cetartiodactyla', 1848, 13, 4),('Perissodactyla', 1848, 13, 4);
```

As I used AUTO_INCREMENT function, I did not have to specify ID for them, SQL did it for me. I provided the example with … above but full code is provided in homework file.

2.3.3 Update Queries

One update can be done is for my Species table. If someone from USA wants to work with my table, it will be convenient for him or her to use imperial system instead of metric system.

So, he or she can convert my gram variable to ounces and centimeter variable to inches.

```
UPDATE Species
SET avg_weight = avg_weight*0.04, avg_height = avg_height*0.4;
```

With a very crude approximation we can say 1 cm is 0.4 inches and 1 gram is 0.04 ounces. If we update our tables like this and run a query, we can see our queries are in ounces and inches. However, MySQL can give an error that requires you to turn off the safe mod to perform that update query. If this is the case, at the top of your update query, you should add

```
SET SQL_SAFE_UPDATES = 0;
```

line to be able to perform update.

2.3.4 Delete Queries

This database have many taxonomy levels but the one that has detailed information is species. So, I want to delete all upmost levels that does not have a single species. As I notice, each order have at least one family. But this is not the case with families. I want to delete all families that does not contain a single genus, hence a single species.

```
DELETE FROM Family
WHERE family_id NOT IN (SELECT family
                        FROM genus);
```

When I perform the nested delete above, I can delete all the families that does not contain a single genus, hence a species.

2.3.5 Complex Queries

When I was doing homework, I noticed that this area is dominated by scientist from 4 countries: Germany, France, UK, and USA. While choosing lower-level animals from upper-level ones, I was unbiased, I lastly looked at the scientist that discovered it. According to the internet, those countries are somewhat similar in their forest areas, they have similar number of trees in terms of percentage. I wonder, which of those countries' scientists are mostly productive in discovery of arboreal (tree-climbing) animals in my database. As for species in same genus, generally same scientist is named, I will look for genus. I will use all of the countries but result is likely to be France, Germany, UK, or USA is the best in discovering arboreal mammals.

```
SELECT nationality, count(*)
FROM scientist s, genus g1
WHERE s.scientist_id = g1.named_scientist
        AND g1.genus_id IN (SELECT genus_id
                            FROM genus g2, species s2
                            WHERE s2.genus = g2.genus_id
        AND behavior LIKE '%arboreal%')
GROUP BY nationality;
```

This query does the job. Winner is France. French scientists have discovered 18 different genus that includes at least one arboreal species.

Throughout the project, I have realized that generally species are named earlier than genus, genus are named earlier than families and so on. I believe this is mostly because of the fact that first there was only species and in time new taxonomy levels emerged that needs to be named. I wonder, where in the world are those whose genus names are given before their species name can be found. I will return only top 5 regions that has the most species that is named after genus.

```
SELECT r.name, count(*)
FROM Found_in F, Region R
WHERE r.region_id = f.region_id
    AND EXISTS
        (SELECT s2.species_id
        FROM species s2
        WHERE f.species_id = s2.species_id
            AND EXISTS (SELECT s.species_id
                        FROM species s, genus g
                        WHERE s.genus = g.genus_id
                        AND s2.species_id = s.species_id
                        AND s.year > g.year))
GROUP BY r.name
ORDER BY count(*) DESC
LIMIT 5;
```

Above query does the job and we can find out that in Southeast Asia, we have the most species that are named after its genus. I did that query because I assumed in the developed regions such as North America and Europe, taxonomy would be more developed and there will not be much difference between the genus and species' discovery times.

3. Conclusion

In this project, I have tried to create a database for mammals. I have used 4 taxonomy levels and I tried to include as much information as possible to honor people that worked on them. First, I provided an ER diagram to show my rationale in this database. Then I provided tables and their data details to show how data are provided.

As this page will be mostly blank, I will provide a last Felis Catus (cat) photo that is saying goodbye to finish my 3-week and ~84 human hours (all by myself) spent project.

Figure 12. Goodbye Cat