(a) Domain Description

Ecological Data

In an ecosystem, there exist several species. Each species has an ID, one or more common names, a scientific name and a diet.

These species all live in an ecosystem. Ecosystems consist of a name, location and area code as well as a climate and area in km. Each ecosystem will also have a keystone species, that is, a species that has the greatest effect on an ecosystem [2]. There can only be one keystone species in an ecosystem and a species can only be keystone in one ecosystem.

Researchers are assigned to study the species. One researcher can study a variety of species and there is no need for more than one researcher studying a species at any one time. Researchers will have a name, area of expertise and an ORCID ID.

Each researcher will supervise a student. Keep track of the students' names, courses and year of study.

The species in an ecosystem can interact with each other. This interaction type also needs to be recorded (Predation, parisitism, symbiosis etc).

a Entities

- Species
- Ecosystem
- Researcher
- Student (weak)

b 1:1 Binary relationships

- Keystone Species ↔ Ecosystem

c 1:N Binary relationships

- Researcher ↔ Species
- Researcher ↔ Students

d M:N Binary relationships

- Species ↔ Ecosystem

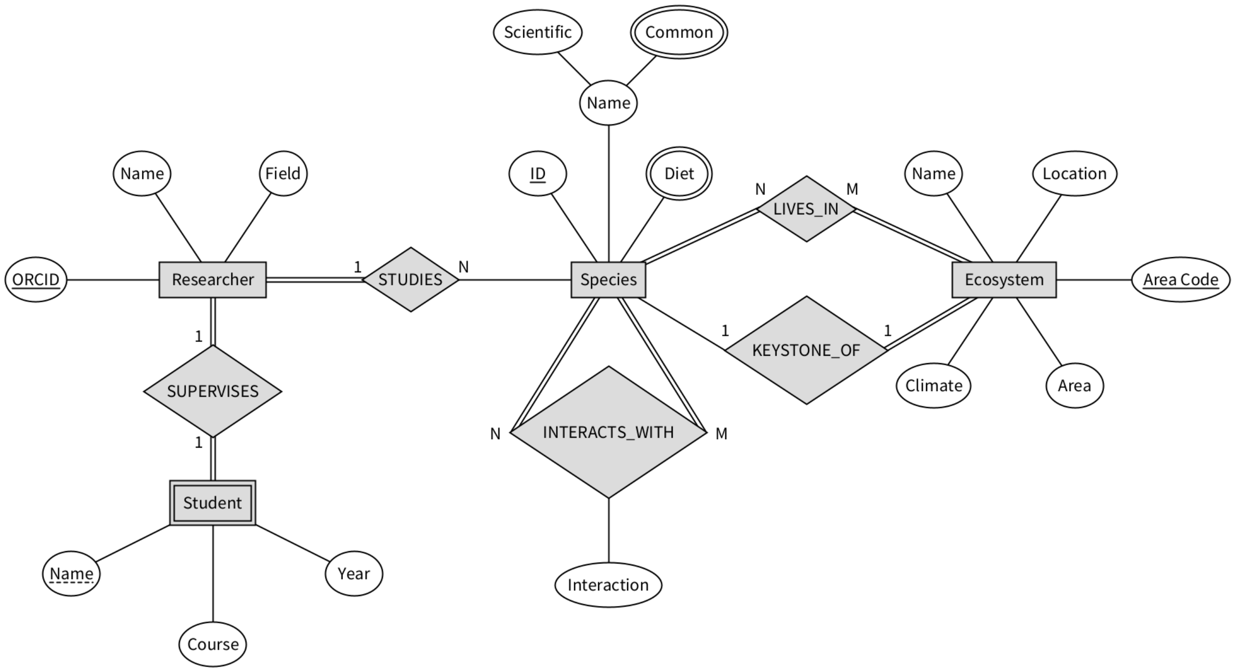
e Recursive relationships

- Species ↔ Species (Interactions)

f Weak entities or multivalued composite attributes

- Species common name (Comp/Multi)
- Student (Weak Entity)

(b) ER Diagram



The Entity relationship diagram seen above was created in Processing in order to give full control over the placement of each entity, attribute and relation. The code used is provided within the *ER_Diagram* directory.

From the domain description, I can extract four distinct terms that I believe can be classified as entities. These are **species**, **ecosystem**, **researcher** and **student**. To start, the species entity contains 5 attributes. ID is a simple attribute we can also assign as a key attribute. Diet likely consists of multiple possible food sources which makes it a multivalued attribute. We know each species has a name, but we are also told that we need to keep track of both scientific and common names. Thus we can create a composite attribute of name with sub-attributes of common and scientific attached to it. A species can have a number of common names in different languages and even in the same language in some cases, making common names a multivalued composite attribute. There is only one fixed scientific name however.

Ecosystem also contains 5 attributes. Name, location climate and area make up the simple attributes while area code can be used as a key attribute.

There are three attributes associated with the researcher entity. Name and field of study are simple attributes and the ORCID works well as a key attribute.

The student entity contains three simple attributes, name, course and year. There doesn't seem to be a good candidate for a key attribute and students only exist in the database if they study under a researcher, so student is likely a weak entity with a composite key made from a combination of the primary key from researcher and the student's name.

Moving on to relationships, every species must live in an ecosystem. Similarly, every ecosystem must have at least one species to be classified as such. This indicates a full participation from both with an N:M cardinality as multiple species can reside in multiple ecosystems.

There is also the keystone relationship between species and ecosystem. By definition, there can only be one keystone species in an ecosystem and as the keystone species is a defining characteristic of the ecosystem itself, each species can be keystone of only one ecosystem. Every ecosystem must have a keystone but not every species can be one.

Every researcher must study a species to be in the database, but not every species is studied. For the purposes of this database, one researcher is responsible for multiple species with no overlap. This creates a 1:N cardinality.

Each researcher will supervise one student. Every researcher must take part, and every student in the database will have a supervisor.

There is a recursive relationship between each species and a number of others. This involves inter-species interactions where every species takes part by the fact that they exist in nature. The interaction type is also recorded.

(c) Relational Schema Mapping

Step 1 Regular Entity Types

The following tables can be mapped from regular entities and simple attributes from the ER diagram:

SPECIES

ID	ScientificName

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name

RESEARCHER

ORCID	Name	Field

Step 2 Weak Entity Types

From this, the weak entity that is student can be integrated:

SPECIES

ID	ScientificName

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

Step 3 1:1 Relation Types

There are two 1:1 relations in the ER diagram, supervises and keystone of. We will begin with keystone of as that is the simplest. The first method works best in this case as we have a full and partial participation.

Taking ecosystem as S as it has full participation, we can take ID of the species as a foreign key in ecosystem and call it KeystoneID.

Supervises is a bit different. As I have already mapped the weak entities in step 2, supervises as a relation is already mapped to the schema.

SPECIES

ID	ScientificName

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name	KeystoneID

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

Step 4 1:N Relation Types

Studies is the singular 1:N relation in the diagram. We can take species as S and researcher as T and take the primary key, ORCID as a foreign key in the species table. There are no attributes in the relation to add to species.

SPECIES

ID	ScientificName	ResearcherID

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name	KeystoneID

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

Step 5 M:N Relation Types

Interacts with and lives in are the two M:N relations in the database. For each, we create a new table and include the primary keys of the participating entities as foreign keys which will make up the composite primary key. The new interacts with table will also include the interaction attribute.

SPECIES

ID	ScientificName	ResearcherID

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name	KeystoneID

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

INTERACTS_WITH

SpeciesID1	SpeciesID2	Interaction

LIVES_IN

SpeciesID	AreaCode

Step 6 Multivalued Attributes

There are two multivalued attributes, diet and common name and both belong to species. These are mapped by creating a new table with the primary key of the entity the attribute belongs to taken as a foreign key. This is combined with the attribute to form a composite primary key for the table.

SPECIES

ID	ScientificName	ResearcherID

DIET

SpeciesID	Diet

COMMON_NAME

SpeciesID	CommonName

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name	KeystoneID

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

INTERACTS_WITH

SpeciesID1	SpeciesID2	Interaction

LIVES_IN

SpeciesID	AreaCode

Step 7 N-ary Relation Types

There are no N-ary relation types in this ER diagram, meaning we can safely skip this step. Therefore the following is the fully constructed mapping of the ER diagram to a schema:

SPECIES

ID	ScientificName	ResearcherID

DIET

SpeciesID	Diet

COMMON_NAME

SpeciesID	CommonName

ECOSYSTEM

AreaCode	AreaKM	Location	Climate	Name	KeystoneID

RESEARCHER

ORCID	Name	Field

STUDENT

ORCID	Name	Course	Year

INTERACTS_WITH

SpeciesID1	SpeciesID2	Interaction

LIVES_IN

SpeciesID	AreaCode

(d) SQL Implementation

1 Creation of Tables

The SQL code was implemented in separate files in order to maintain modularity. I began with the researcher and species tables as many of the other tables relied on the IDs of both as foreign keys. These IDs were assigned automatically with AUTO_INCREMENT to make sure they were unique. In both tables I used UNIQUE to prevent accidentally adding the same information again. Without this, if we were to run the same insert command twice it would give two ID values to the same entry. However, in the unlikely scenario that a researcher has the same name and field of study, they will be prevented from entering the database. Using a composite key of the ID and name may solve this issue.

The two multivalued attributes, diet and common names were implemented with composite private keys made up of the species ID and the attribute itself. I realise now that it is this composite key that allows the one entity to have multiple of the same attribute.

The ecosystem table has the most information out of any of the tables. The area code is kept as a small varchar to prevent mistakes and the area in km is an unsigned int to make sure there isn't a negative area. The five most common climate types are provided as an ENUM. The Species ID of the keystone species is recorded here as a foreign key.

As there is a 1:1 relation between researchers who supervise students, the students can use their supervisor's ORCID in combination with their names to create a unique primary key. There is a check on the inserted year value to ensure no impossible values are input.

As the interacts with table takes in two foreign keys of the same type, there is a check to make sure that the two IDs are not identical. These IDs become a composite primary key. Another ENUM controls the interaction types that can be input into the system.

The lives in table is similar but without a need for as many controls as it is not a recursive relation.

2 Population of the Database

Populating the tables in the database was done in the same order that the tables were created. The researcher table was filled first, as species relied on the value of the researcher's ID. Although this was not strictly necessary as the researcher ID column in the species table could be left as NULL or populated later. Not all species were expected to be studied.

Species was inserted next as many entities in the database did rely on the species ID. The ID values in both species and researcher were left as NULL in the INSERT function to let SQL automatically populate the ID values with incrementing, unique numbers. Names were chosen at random from a-z animals.com [1]. The rest of the data was made up for the purposes of demonstrating the database.

I tried to assign multiple values in the diet and common name tables whenever I could to demonstrate that this was possible. Some of the species names may be a bit of a stretch as a result. I kept the ecosystem table short with six values that could be reused for all of the species in the demonstration. There was one student for every researcher due to the 1:1 relation. Some sample interactions were added between species and every species was included in an ecosystem, though they could have been added to more than one.

3 Data Integrity Constraints and Triggers

The first trigger is a simple one intended to make sure there are no orphan processes when trying to delete a researcher. The user is reminded to delete the corresponding student and species entries that use the ORCID as a foreign key. This works by checking those foreign key values before the delete function is executed and stopping the process and sending a custom error message if they exist.

The second comes into play after every creation of an entry in the ecosystem table. This trigger automatically adds the included speciesID in the keystoneID section of the ecosystem table to the LIVES_IN table. The idea is that there should never be a case where the keystone species of a particular ecosystem does not live in said ecosystem. Implementing this trigger makes logical sense and reduces the number of INSERT functions done manually.

This trigger is easily tested with a SELECT and INNER JOIN.

```
SELECT SPECIES.ID, SPECIES.ScientificName, LIVES_IN.AreaCode
FROM SPECIES
INNER JOIN LIVES_IN
ON SPECIES.ID = LIVES_IN.SpeciesID
WHERE SPECIES.ID = 11
ORDER BY SPECIES.ID ASC;
```

The script shows Amanita Muscaria and the area codes it lives in. After inserting a new ecosystem with it as the keystone species, we can run the script again and see that Amanita Muscaria belongs to the new ecosystem.

```
INSERT INTO ECOSYSTEM VALUES ('EU092', 1000, 'Germany', 'Temperate', 'Mixed Forest', 11);
```

The temporary ecosystem can be removed with the following:

```
DELETE FROM LIVES_IN WHERE AreaCode = 'EU092';
DELETE FROM ECOSYSTEM WHERE AreaCode = 'EU092';
```

4 SQL Queries

Aside from some testing queries, I implemented a SELECT query with INNER JOIN across three tables displaying species information along with the ecosystem in which they live and the researcher studying them. For convenience, the table is then displayed in ascending order of the species ID. This provides all the information closest to species at a glance.

The next query counts the number of diet types of each species and displays only the ones with more than two diet types. This can be used to isolate the species with more varied and diverse diets. It achieves this by counting the number of diet types in each species and grouping by the species' scientific name as long as the count was greater than 1.

The final query uses a subquery to display every case of an ecosystem which contains more than 5 species. The inner WHERE ECOSYSTEM.AreaCode IN () selects the area codes where SpeciesID has a count of more than 5 which is then joined with further ecosystem and species information.

References

[1] Animals by scientific name: A complete list. https://a-z-animals.com/animals/scientific/. Accessed: 2025-11-15.

[2] R. T. Paine. A Note on Trophic Complexity and Community Stability. *The American Naturalist*, 103(929):91–93, January 1969. Publisher: The University of Chicago Press.