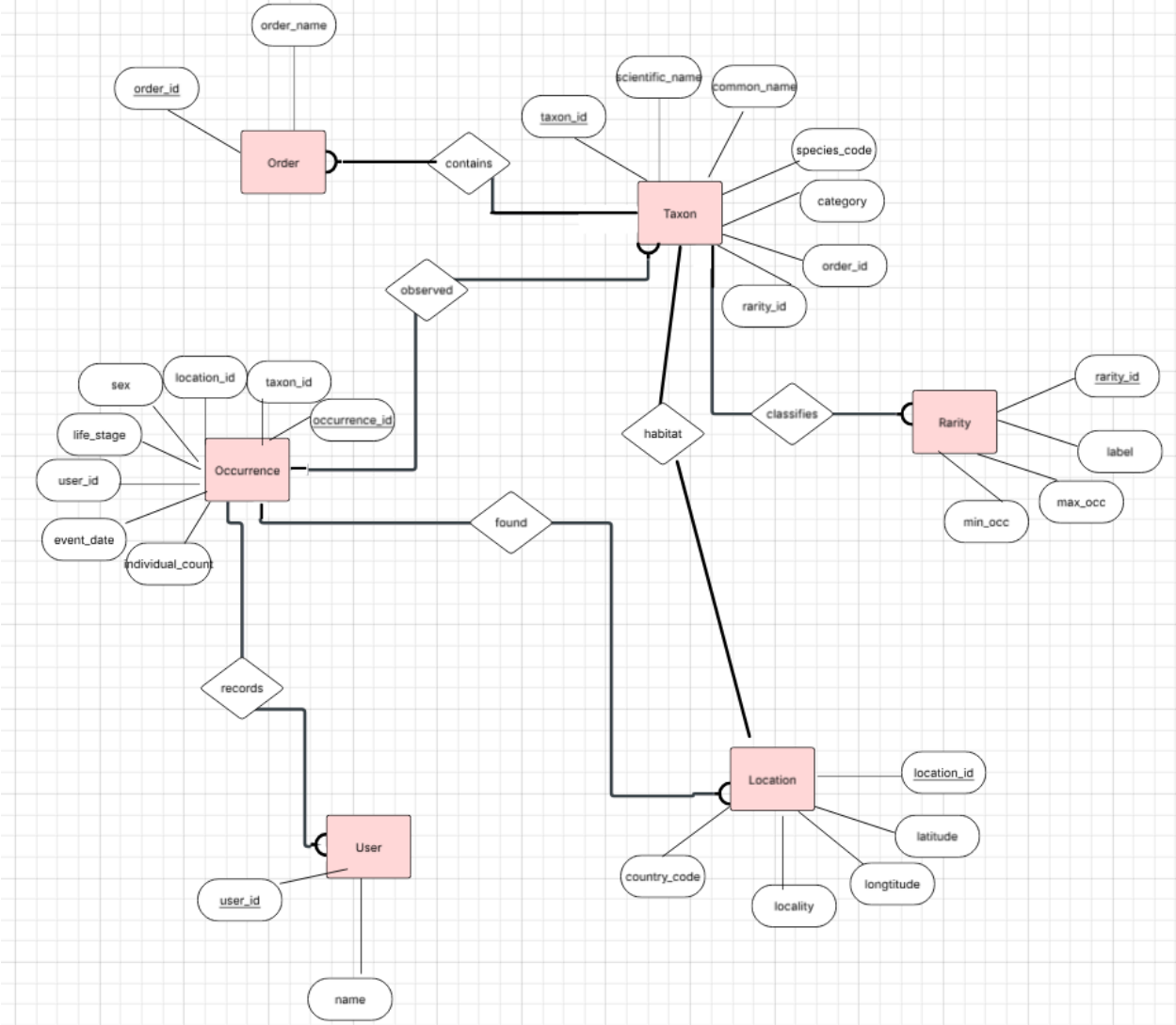
Stage 2 UrbanBird — Conceptual Design Documentation

ER Diagram:

We created a conceptual design of the biodiversity database using an ER diagram. The design includes six entities: Order, Rarity, Taxon, Location, Occurrence, and User, with only one entity dedicated to user information. It clearly models both 1-to-many and many-to-many relationships as required.



Assumptions and Explanations for Each Entity:

1. **Order**

The taxonomy of each bird is divided into 44 orders - this is the information the order table stores. We selected Order as an entity because a single order contains the taxonomy of many birds. Therefore, storing it in a separate table that is related to taxon with a 1-to-many relationship allows us to remove the order attribute from the taxon table. Thus, we do not have to repeat the same string multiple times and save storage space.

2. **Rarity**

We selected Rarity as an entity because it essentially functions as a reusable lookup table that allows us to assign various tiers to taxons. This table represents a rarity scale defined by common and rare / legendary birds. The table's attributes are as follows, rarity_id (Primary Key), label, min_occ, max_occ. Furthermore, this table is related to taxon with a 1-to-many relationship. The rarity-scoring aggregates occurrence counts per species and updates the `rarity_id` accordingly, ensuring queries can surface tier badges instantly.

3. **Taxon**

We selected Taxon as an entity because we wanted to capture all the specific information about bird species and families. Combination of all the attributes beget all the biological and taxonomical information that we need for a specific bird (e.g. category, species_name, scientific_name). It has a many to many relationship with the Location entity to capture the necessary background information about a bird's natural habitat and taxonomy.

4. **Location**

We selected Location as an entity because we wanted to capture all the specific geological information about each unique bird. We modeled as an entity to normalize and avoid redundant coordinate data across multiple occurrences. All the attributes combined together generate a map of a bird's natural whereabouts, which leads to the many to many relationship entity with taxon: habitat. With this many to many relationship, we will be able to derive a lot of information about a bird's habitat.

5. **User**

We selected User as our entity rather than an attribute because each user represents a real individual who logs information about birds. It's important to keep track of who recorded which observation, so we need a separate User entity to maintain their identity. Also, creating a User as an entity removes duplicate information since a user is

able to log many occurrences, meaning if there was no User entity, Occurrence would have many user_id duplicates. Since a user is able to log many occurrences because birds are constantly seen outside, it creates a one-to-many relationship with Occurence.

6. **Occurrence**

We selected Occurence as our entity because each User needs to be able to describe what they have seen and it needs to be stored independently allowing for each occurrence to be associated with exactly one user. An occurrence can't link to multiple users, as an occurrence has specific attributes such as event date and location_id that are unique to that user. This creates a one-to-many relationship with User.

Description of Relationships and Cardinalities:

- Order contains Taxon: one-to-many. Each order has many taxa, and each taxon belongs to exactly one order.
- Rarity classifies Taxon: one-to-many. One rarity tier applies to many taxa, and each taxon belongs to one tier.
- Taxon observed in Occurrence: one-to-many. A taxon can appear in many occurrences, and each occurrence involves one taxon.
- Location found in Occurrence: one-to-many. A location can have many occurrences, and each occurrence is tied to one location.
- User records Occurrence: one-to-many. A user can record many occurrences, and each occurrence is attributed to one user.
- Taxon and Location are related many-to-many through Occurrence, representing habitat associations.

Relational Schema:

ORDER(order_id:INT [PK], order_name:VARCHAR(40));

RARITY(rarity_id:INT [PK], label:VARCHAR(15), min_occ:INT, max_occ:INT);

TAXON(taxon_id:INT [PK], scientific_name:VARCHAR(100), common_name:VARCHAR(100), species_code:CHAR(6), category:VARCHAR(20), order_id:INT [FK to ORDER.order_id], rarity_id:INT [FK to RARITY.rarity_id]);

LOCATION(location_id:INT [PK], latitude:DECIMAL(9,6), longitude:DECIMAL(9,6), locality:VARCHAR(200), country_code:CHAR(2));

USER(user_id:INT [PK], name:VARCHAR(60));

OCCURRENCE(occurrence_id:BIGINT [PK], taxon_id:INT [FK to TAXON.taxon_id], location_id:INT [FK to LOCATION.location_id], user_id:INT [FK to USER.user_id], event_date:DATE, individual_count:INT, sex:VARCHAR(15), life_stage:VARCHAR(15), basis_of_record:VARCHAR(30))

HABITAT(taxon_id:INT [FK to TAXON.taxon_id], location_id:INT [FK to LOCATION.location_id], PRIMARY KEY (taxon_id, location_id))

Normalization:
For a database schema to satisfy the Third Normal Form (3NF) it has to satisfy the following requirements:

1. Each table has to have a primary key - this requirement is satisfied since each of our tables contain a clearly defined primary key (order_id, rarity_id, taxon_id, location_id, user_id, occurrence_id).
2. Each attribute has to be atomic (i.e. there has to be one value per cell) - this requirement is satisfied within our tables.
3. Every attribute that is not a key is dependent on the whole primary key - this is true within our tablets. For example, in Occurrance the event_date, individual_count, sex, etc. depend on only occurance_id not other primary keys, such as, location_id or user_id.
4. Furthermore, the final requirement is that each attribute that is not a key does not depend on any other attribute that is a key. For example, in taxon, order_id and rarity_id are foreign keys that refer to the Order table and Rarity table only. Furthermore, order_name is not duplicated within the Taxon table, it is only stored within Order. Therefore, our database satisfied this requirement.

Thus, our database satisfies the 3NF requirements.