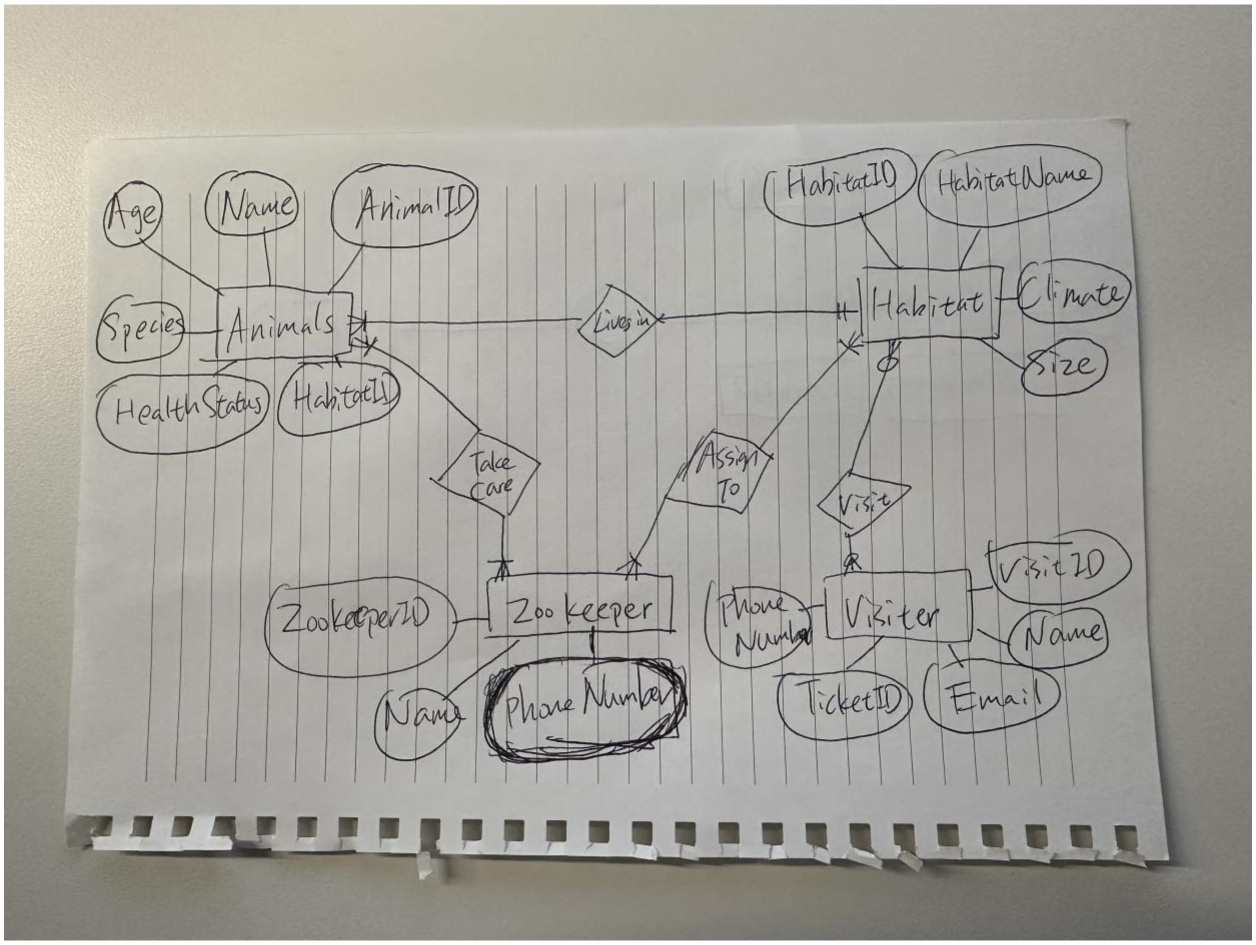
HW4 Part2

ER Diagram for Zoo Database

Muyang Cheng



1. Entities & Attributes

This zoo database features four primary entities, Animal, Habitat, Zookeeper and Visitor, along with their main attributes:

Entities:

1. Animal
    1. AnimalID (Primary Key, Unique)
    2. Name (Animal's name)
    3. Species (Type of animal)
    4. Age (Must be 0 or more)
    5. HealthStatus (Only "Healthy" or "Sick")
    6. HabitatID (Foreign Key → Habitat)
2. Habitat
    1. HabitatID (Primary Key, Unique)
    2. HabitatName (Name of the habitat)
    3. Climate (Hot, Cold, etc.)
    4. Size (Area size)
3. Zookeeper
    1. ZookeeperID (Primary Key, Unique)
    2. Name (Zookeeper's name)
    3. PhoneNumber (Unique contact number)
4. Visitor
    1. VisitorID (Primary Key, Unique)
    2. Name (Visitor’s name)
    3. Email (Optional, could be unique if desired)
    4. PhoneNumber (Optional, could also be unique if desired)
    5. TicketID (Optional, or other ticket-related attributes)

2. Relationships

1. Animal - Habitat (1:N)
    1. One habitat can have many animals, but each animal stays in exactly one habitat.
    2. The Animal table includes HabitatID as a foreign key referencing Habitat(HabitatID).
2. Zookeeper - Habitat (M:N)
    1. One zookeeper may manage multiple habitats, and one habitat can be managed by multiple zookeepers.
    2. Use a junction table (e.g., ZookeeperHabitat) with at least two foreign keys:
       ZookeeperID (FK → Zookeeper.ZookeeperID)
       HabitatID (FK → Habitat.HabitatID)
3. Visitor - Habitat (M:N)

    1. One visitor can visit multiple habitats, and one habitat can be visited by many visitors.
    2. Similarly, use a junction table (e.g., VisitorHabitat or VisitRecord) with:
       VisitorID (FK → Visitor.VisitorID)
       HabitatID (FK → Habitat.HabitatID)

3. Constraints

1. Primary Keys: AnimalID, HabitatID, ZookeeperID, VisitorID must each be unique within their respective tables.
2. Foreign Keys:
    1. Animal(HabitatID) → Habitat(HabitatID)
    2. ZookeeperHabitat(ZookeeperID) → Zookeeper(ZookeeperID) and ZookeeperHabitat(HabitatID) → Habitat(HabitatID)
    3. VisitorHabitat(VisitorID) → Visitor(VisitorID) and VisitorHabitat(HabitatID) → Habitat(HabitatID)
3. Data Rules:
    1. Age ≥ 0
    2. HealthStatus ∈ { "Healthy", "Sick" }
    3. Zookeeper. PhoneNumber must be unique.
    4. Visitor. PhoneNumber or Visitor. Email can be optionally enforced as unique depending on business requirements.
    5. Other constraints (e.g., non-nullable fields, length limits, or check constraints) can be defined as needed.

4. Design

1. Animal - Habitat (1:N)
    1. Each animal resides in exactly one habitat, and a habitat can hold multiple animals.
    2. Storing HabitatID in the Animal table ensures straightforward lookup and preserves data integrity.
2. Zookeeper-Habitat (M:N)
    1. One zookeeper may be in charge of several habitats, and a single habitat might require multiple zookeepers for proper management.
    2. A junction table allows a flexible way to store and query which zookeeper manages which habitat(s).
3. Visitor - Habitat (M:N)
    1. A single visitor often visits multiple habitats during a trip, and a habitat receives many visitors over time.
    2. A junction (link) table keeps track of visits, potentially including timestamps, ticket IDs, or reviews.

4. Enforcing Data Quality
    1. By using primary keys, foreign keys, and check constraints, we could maintain referential integrity (e.g., an animal cannot reference a non-existent habitat) and valid data states (e.g., non-negative ages, consistent phone numbers).