Database Design

The Database Design Process

1. Requirements Analysis — What does the user want? What tasks/apps will the database support?
2. Conceptual Design — High-level description of the data and its relation to the real-world.
3. Logical Design — Translate the conceptual model into a DBMS data model e.g., relational model (logical schema)
4. Schema Refinement — Ensure data integrity
5. Physical Design — Select physical layouts, indexes, etc.
6. Security Design — Access Control, Privileges

Entity Relationship Modelling

Translation to relational

Functional Dependencies
Normalization & Decomposition

Entity-Relationship Model

Conceptual Design

What are the *entities*?

- Real world objects that we want to store information about

What are the *relations*?

- What are the associations between the entities?

How much of the real world needs to be modeled and stored in the DB?

What are the *integrity constraints*?

- What are the business rules that should always hold?

The Zoo

Entities:

Animals, Keepers, Enclosures, …

Relations:

LivesIn, FedBy, …

What to store?

- Dimensions of an enclosure?
- Medical history?
- Nutrition tables

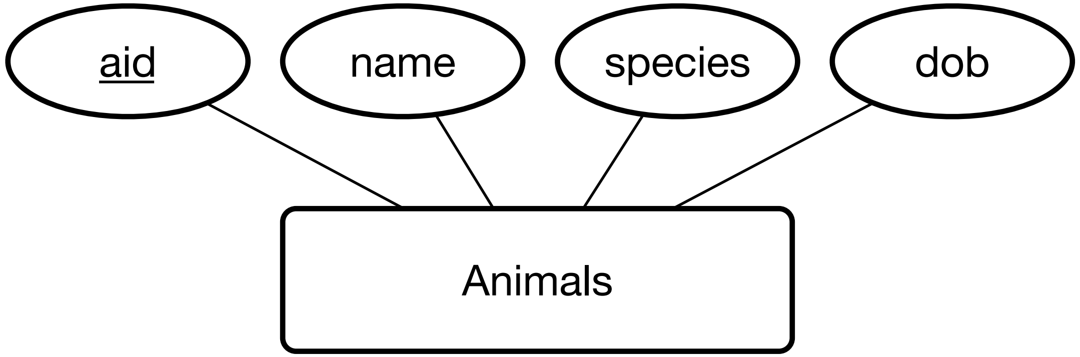
Constraints?

- Should animals always have an enclosure?
- Can an enclosure have more than one animal?
- What rules determine the base salary of a keeper?

Entity Relationship (ER) Modelling

Why can't we just use the relational model?

1. ER model is a high-level data model primarily used for database design
2. It is visual and simple
   - Matches how users think of their data
   - Facilitates discussion (inclusive of users with no tech background)
3. Easy to translate to a DBMS data model



Entities

Entity: A real-world object described by a set of attribute values.

Entity Set: A collection of similar entities.

Each entity set has a *key* (underlined); a set of attributes that uniquely identify each entity.

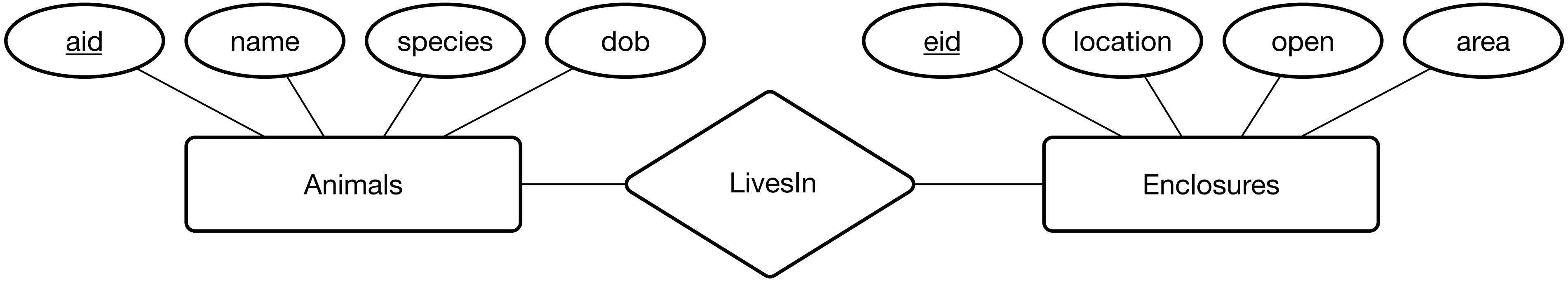
Each attribute has a *domain*

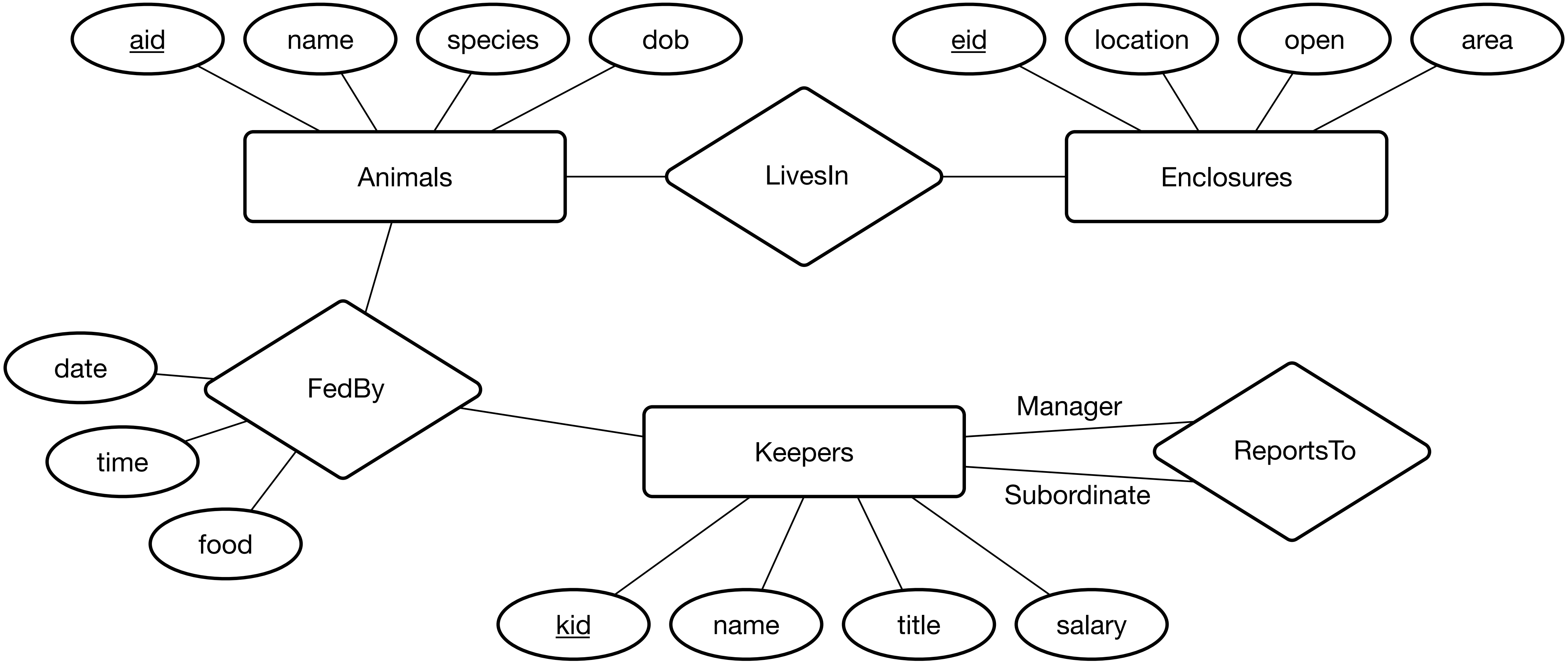
Relationship: An association among two or more entities $E_1, \ldots, E_n$

Relationships can have their own attributes $a_1, \ldots a_m$

Relationship Set: A collection of similar relationships.
$\{(e_1, \ldots, e_n, a_1, \ldots, a_m) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$

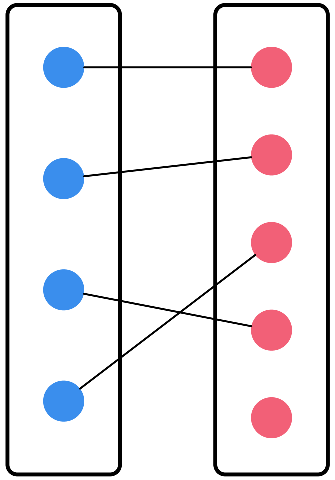
Relationships




aid
name
species
dob
eid
location
open
area
Animals
LivesIn
Enclosures
date
FedBy
time
food
Keepers
Manager
ReportsTo
Subordinate
kid
name
title
salary

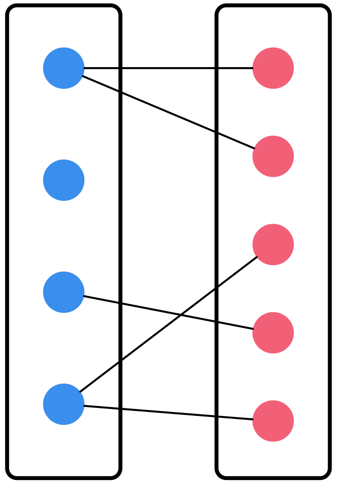
Relationships & Constraints

Types of Relationships



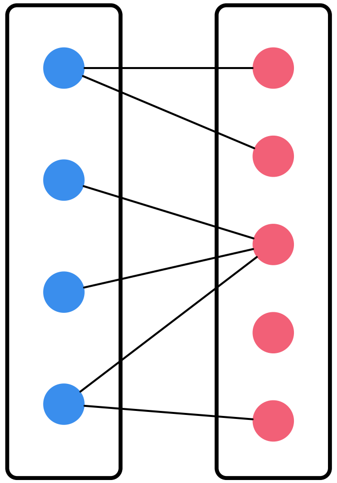
1-1

An animal can live in at most one enclosure only and each enclosure can have at most one animal



1-many | many-1

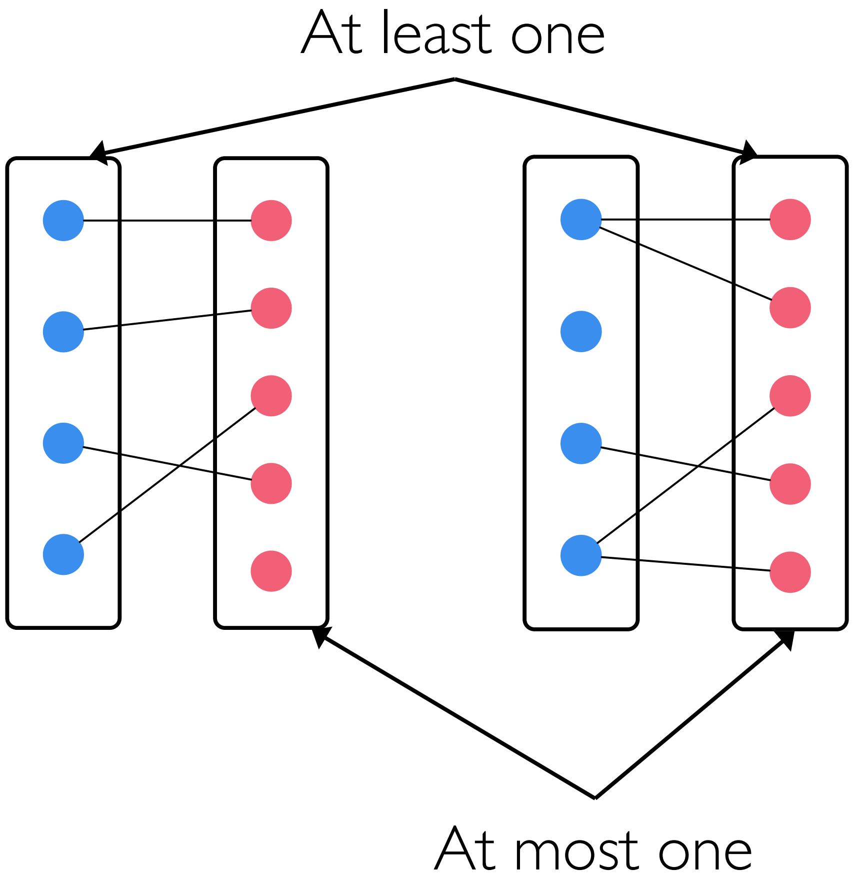
An animal can live across many enclosures, but each enclosure can have at most one animal



many-many

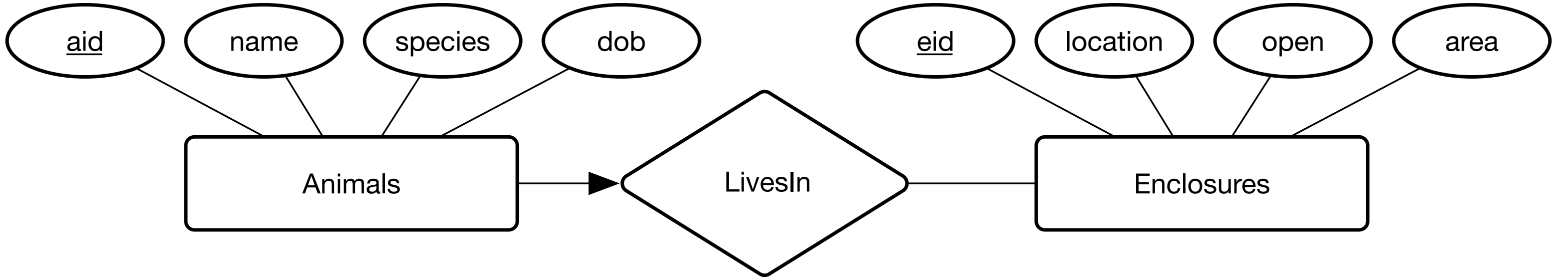
An animal can live across many enclosures, and each enclosure can have many animals

Types of Relationships



Key Constraints

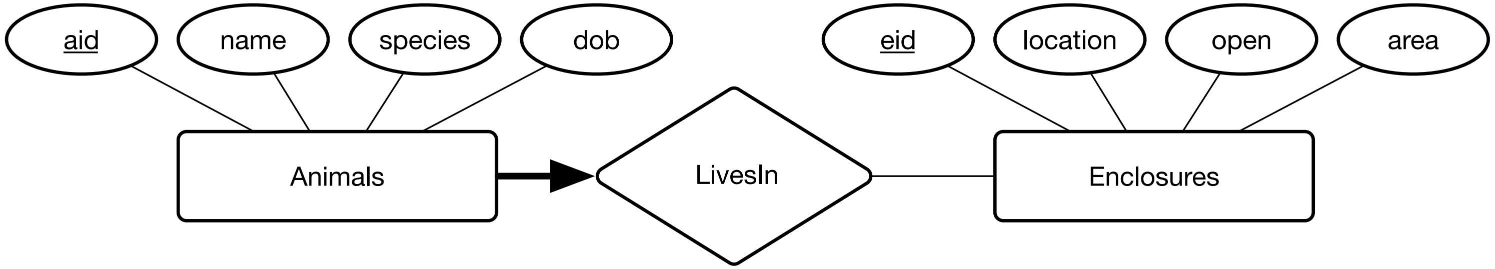
- Gives us a 1-many relationship constraints
- Enforces "**at most**" one constraint



An animal lives in at most one enclosure!

Participation Constraints

- Enforces **"at least"** one constraint
- Partial vs. total participation



Participation Constraint: An animal lives in at least one enclosure!

Key Constraint: An animal lives in at most one enclosure

An animal lives in exactly one enclosure

Design Choices

Modelling Design Choices

ER Modeling is not always straightforward

CHOICES

Entity vs. Attribute?

Entity vs. Relationship?

Binary vs. N-ary Relationships?

Aggregations; Generalizations & Specializations; …

GOALS

Capture the semantics of the real world & its constraints as closely as possible

Eliminate Redundancy

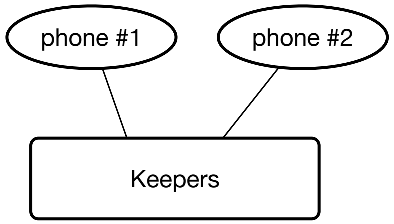
Performance

Simplicity

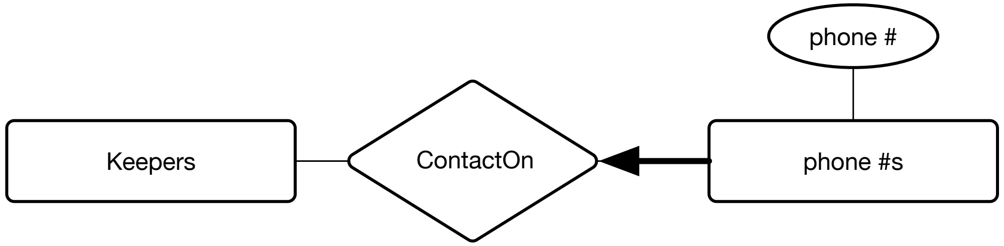
Entity vs. Attribute?

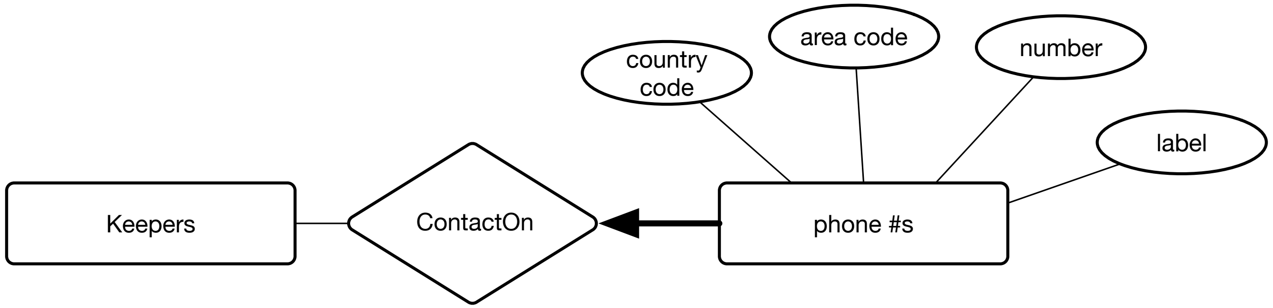
- Semantics - Is the item of direct interest to the database?
- Does the object have single or multiple instances?
- Does the object have components of its own? Atomic or tuple-valued?
- Is the object often non-existent or unknown?

Single value



Multiple values

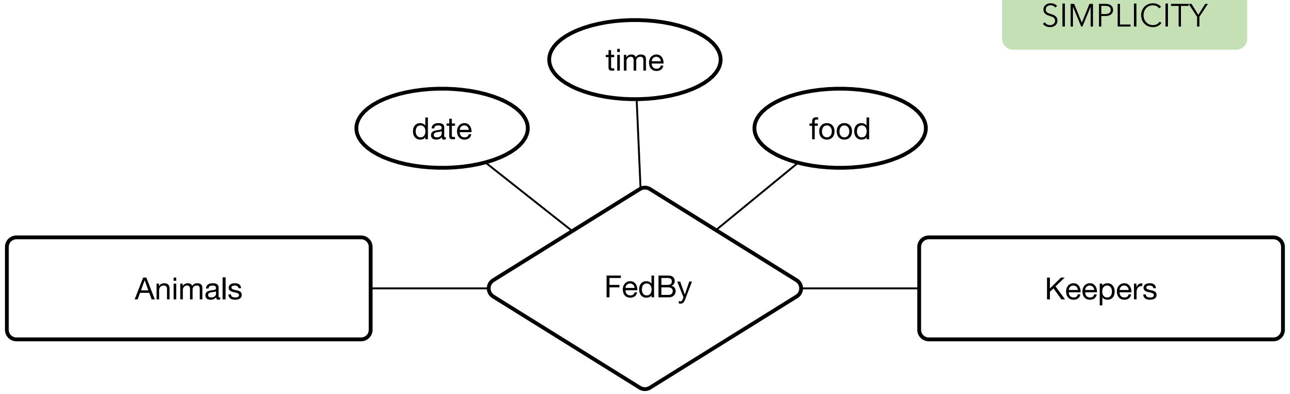




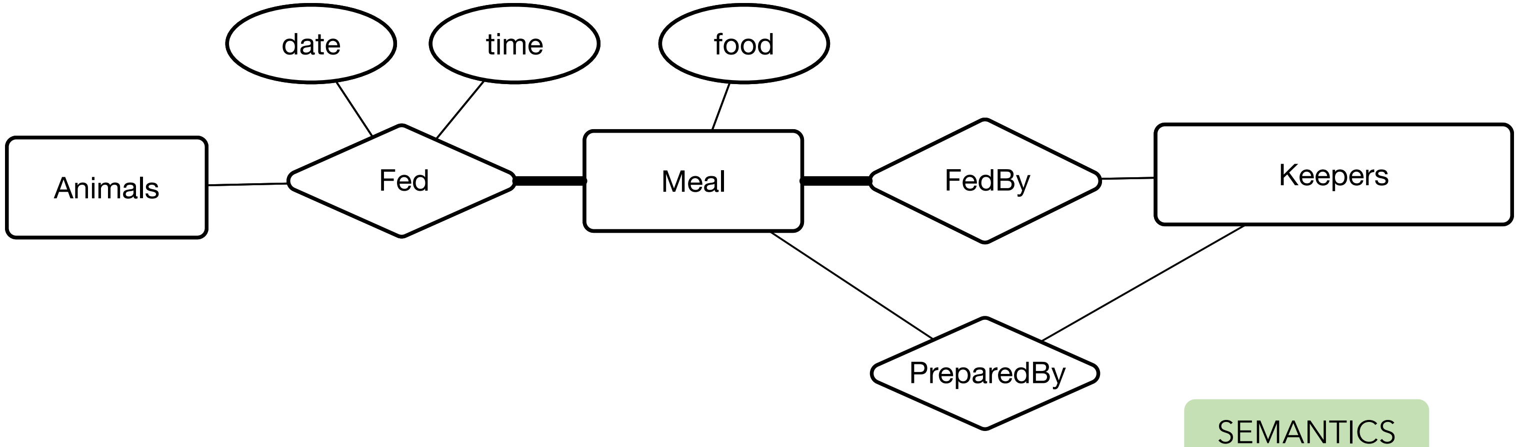
Has structure, tuple-valued

Entity vs. Relationship?

- Semantics
- A relationship is a more compact and preferable option here unless …
- We associate other information with the Meal record



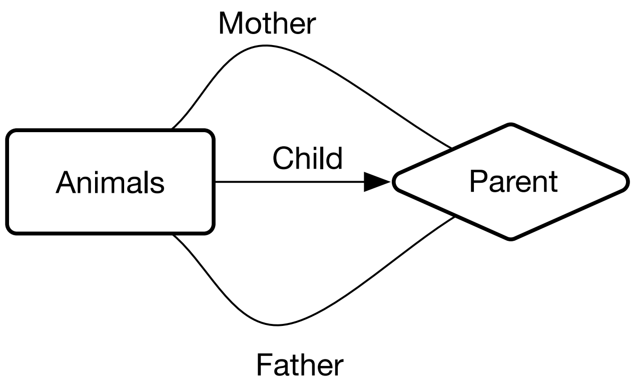
SIMPLICITY



SEMANTICS

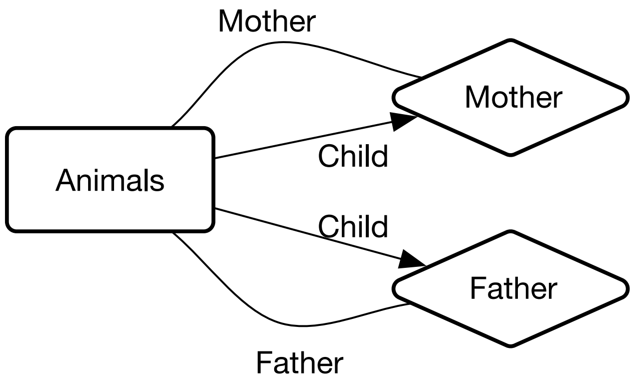
Binary vs. *n*-ary relationship?

It depends…

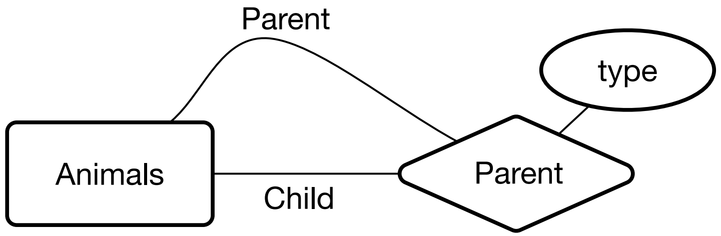


Ternary: what if we don't have information on one of the parents? Store as NULLs!

SEMANTICS

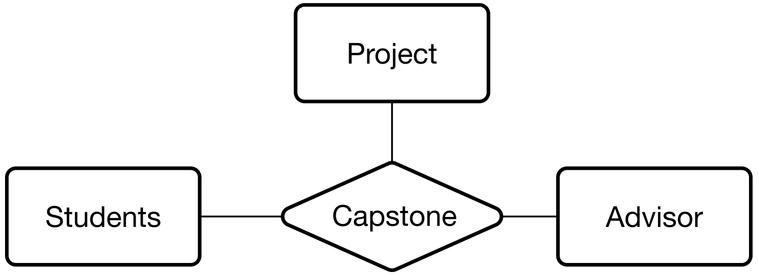


Is this binary relationship preferrable?



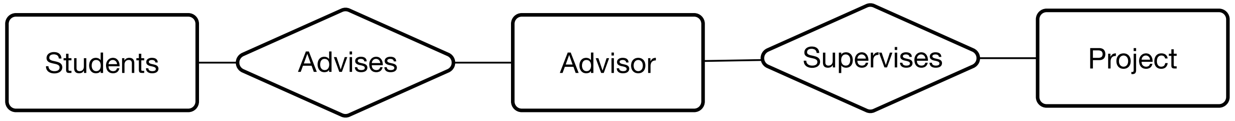
PERFORMANCE

Better Design?
Can encode asexual reproduction better. But, how can I ensure that an animal has at most **two** parents?



We may have to repeat information (e.g. when multiple students work on the same project)

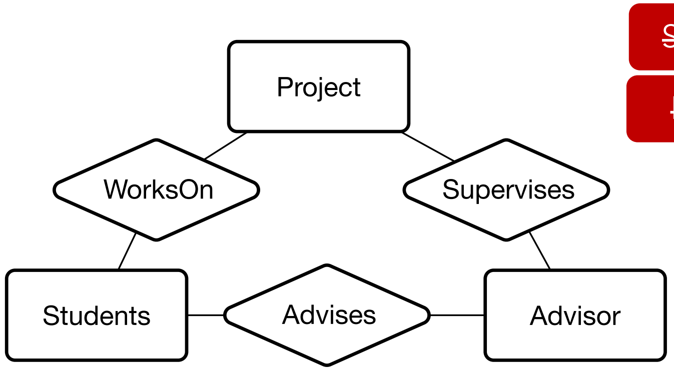
~~SEMANTICS~~



This binary relationship loses important information. We know that an advisor supervises certain projects, and advices certain students, but we don't know which students work on which projects

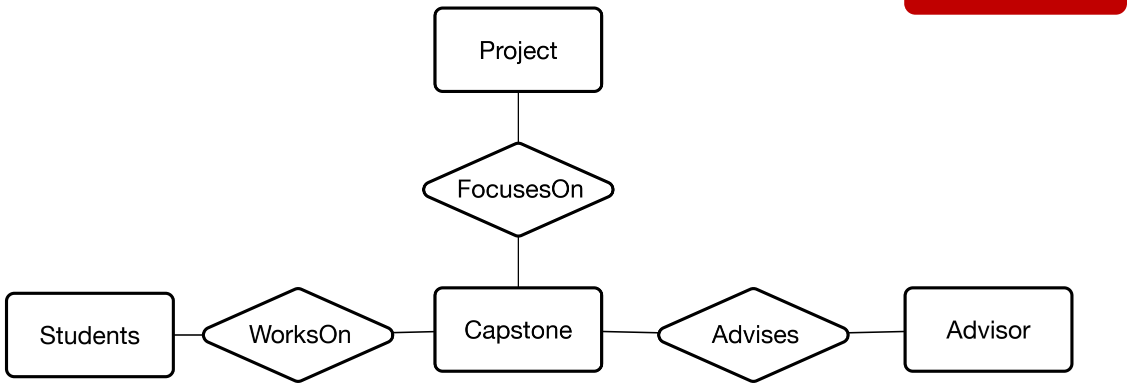
~~SEMANTICS~~

~~PERFORMANCE~~



Difficult to pull information that relates all three entities together. We cannot associate information with the capstone itself such as a "grade"

~~SIMPLICITY~~



Accurate but unnatural

Binary vs. *n*-ary relationship?

It depends …

Conceptual Design with ER

- Expressive, graphical model captures application semantics well!
- Basic constructs (entities & relationships) are easy to communicate and understand
- Additional constructs exist: ISA hierarchies, Aggregation, Weak Entities, etc.
- Captures some but not all constraints such as functional dependencies.
- Constraints play an important in database design

DATABASE SCHEMA
THAT WAS A GOOD IDEA.
NOW I SEE IT...

Dataedo /cartoon

Piotr@dataedo