Principles of Data Management

Alvin Lin

August 2018 - December 2018

Entity Relationship Models

To reiterate, when planning the logical design of the database, we need to take into account the business need of the database and the computer scientists who need to build the application around the database. The computer scientists need to plan out the structure and relationships of everything in the database.

Entity-Relationship Model

Entity: anything that is distinguishable from other objects
Entity Sets: entities that are related (share some attributes)

$$dog = (name, DOB, gooddog, breed, color, owner)$$

ID	Name
1	Harry
2	Abbey
3	Potato
4	Kali
5	Sade

ID	Name
1	Jeff
2	Pat
3	Alvin
4	Andrew

Relationship Sets: links entity sets together

- Harry - 2005 - Jeff
- Abbey - 2005 - Pat
- Potato - 2016 - Alvin
- Kali - 2018 - Andrew

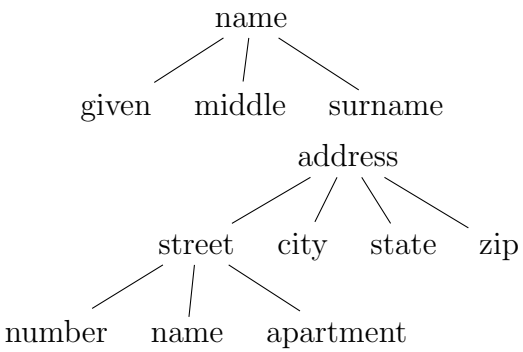
- Sade - 2007 - Andrew

Binary relationships link two entities, but some relationships may have 3 or more entities. Cardinality is used in binary degree relationship sets to define how two entity sets are linked together.

- One to One
- One to Many
- Many to One
- Many to Many

Complex Attributes

Attributes can be simple or complex, single valued or multi-valued. Single valued attributes are things like names, which we only have one of, while emails are multi-valued since we can have multiple. Derived attributes are things like age, which are derived from a date of birth. The domain of an attribute is the set of available values for an attribute. Attributes like names and addresses can be complex attributes:



Redundant Attributes

$$cat = (name, owner.name, id, age)$$

$$owner = (id, name, address)$$

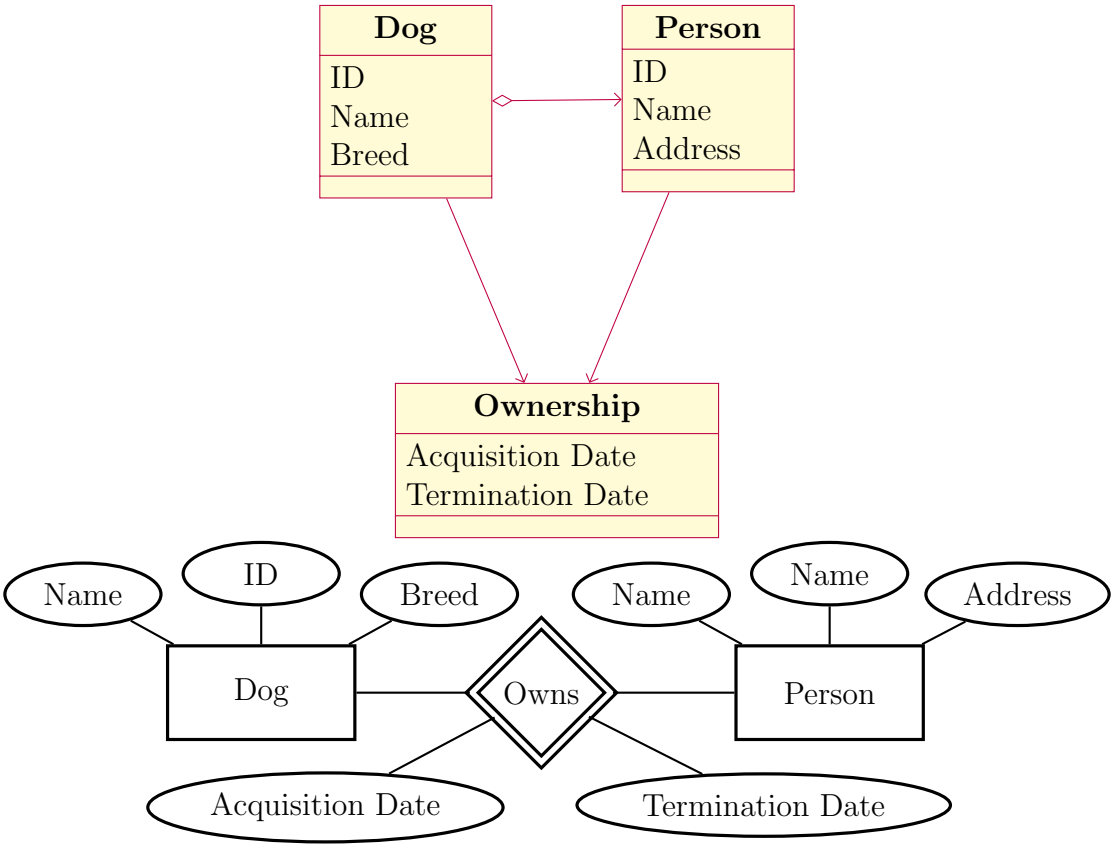
The entities in weak entity sets are not unique.

Name
Andy
Bob
Andy
Bob

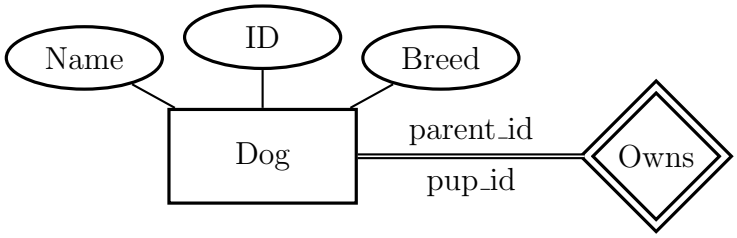
Cat names in this example cannot uniquely identify a cat, so we link it to a strong entity set, which contains an identifying entity. The weak entity set is thus dependent on strong entity set. This provides a discriminating key or attribute that can be added to the weak entity set to make it unique.

ER Diagrams

Entity Set (UML and ER diagram):

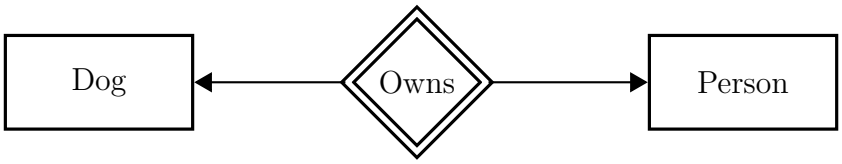


Dogs can also be related to other dogs, which can be represented as follows:

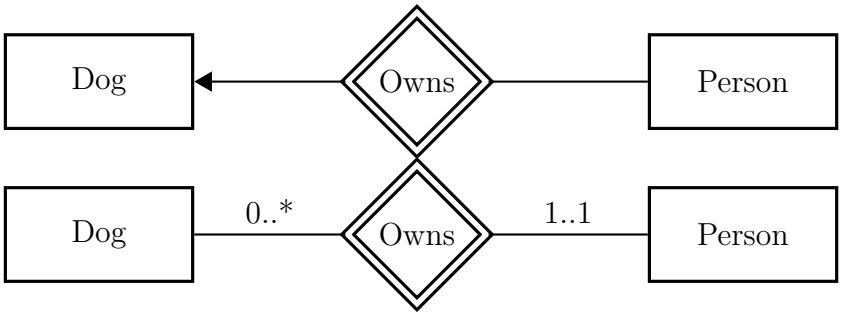


Cardinality

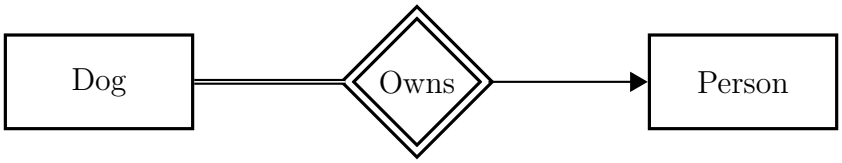
- 1:1



- 1:Many

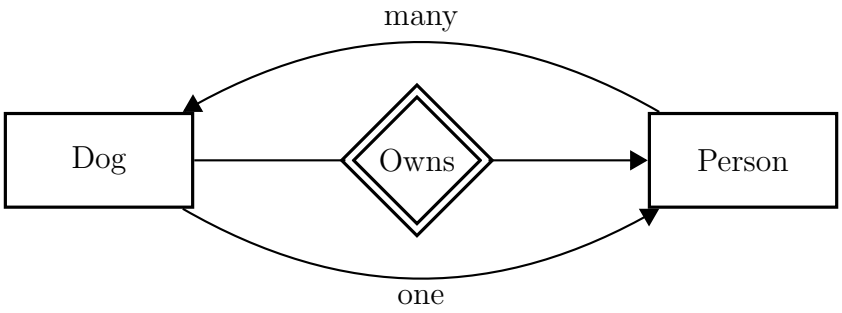


A person has exactly one dog and every dog has zero or more owners.



Every dog has exactly one owner and every owner has zero or more dogs.

- Many:1



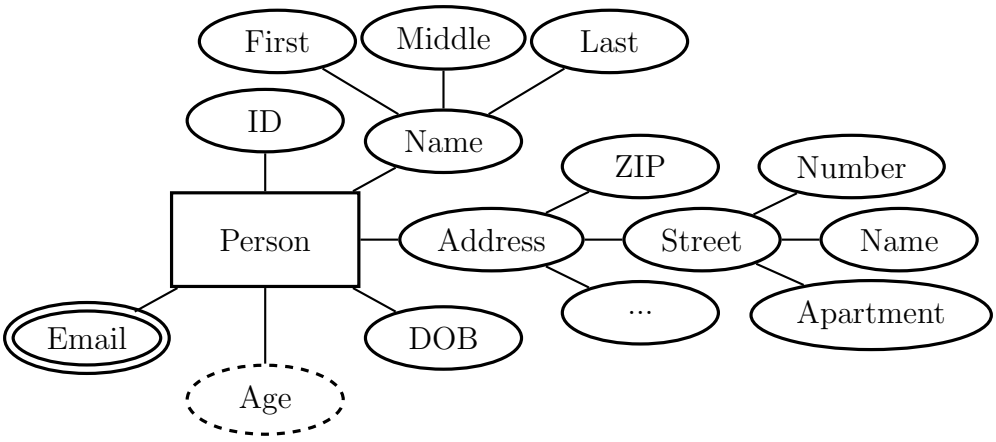
A single line implies a 0..N relationship in that one thing can have 0 or more of another object, while a double line implies 1..N which implies a one or more relationship.

Complex Attributes

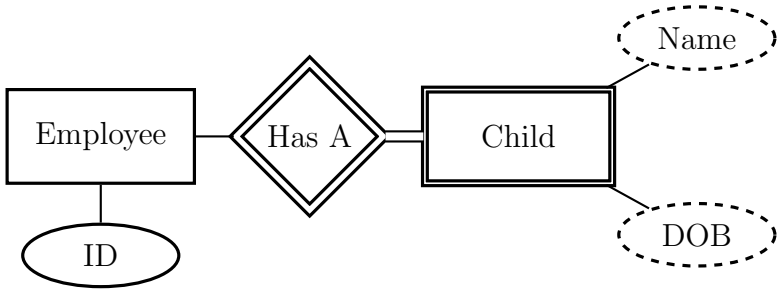
UML:

Person
ID Name: first, middle, last Address: zip, city, state, street, etc {email} DOB
age()

For the accompanying Chen ER diagram, multi-valued attributes like email are double circled, while derived attributes like age are dotted.

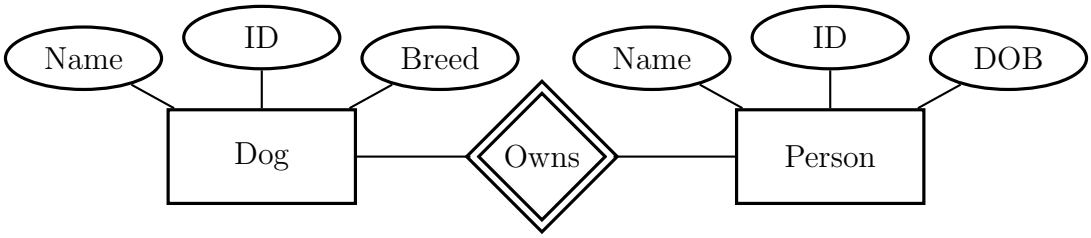


Sometimes, weak entity sets must rely on another entity in order to be uniquely identified. For example:



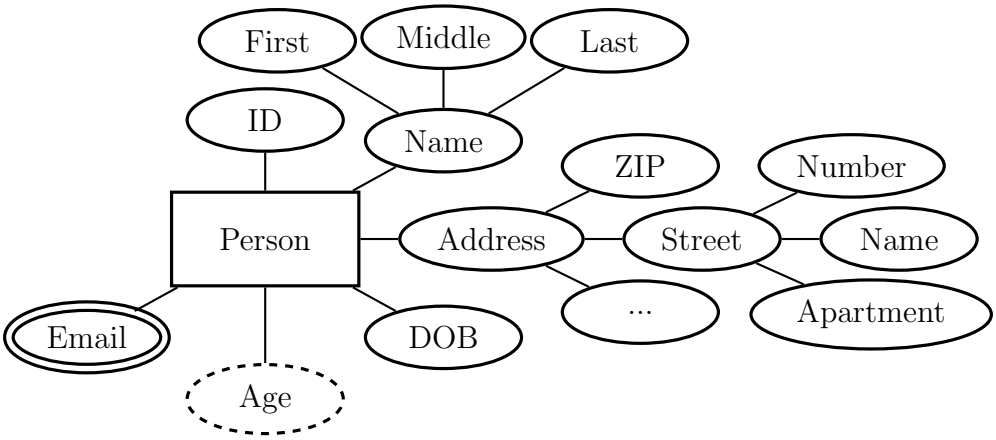
If there are two children (of two employees) with the same name and date of birth, then they are reliant on their relationship with the employee and the employee's ID in order to be uniquely identified.

Reduction into Tables



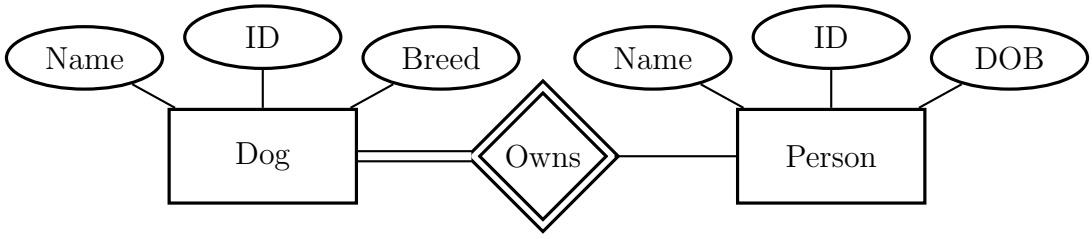
$$Person = (\mathbf{id}, name, DOB)$$
$$Dog = (\mathbf{id}, name, breed)$$
$$owns = (\mathbf{person.id}, \mathbf{dog.id})$$

While the primary key of people and dogs can just be their id, the "owns" table's primary key is both the person and dog id of the association in order to prevent ambiguity.



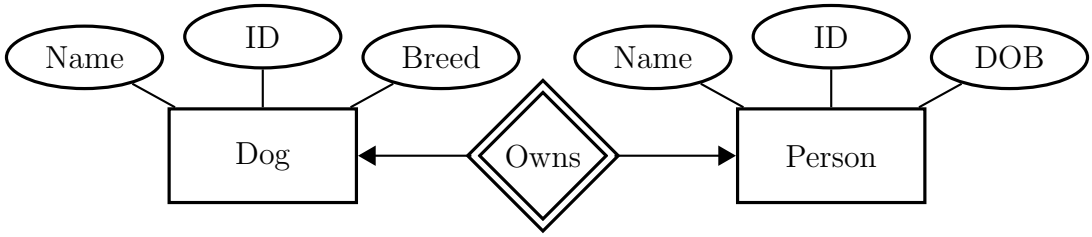
$$Person = (\mathbf{id}, dob, name\_first, name\_middle, name\_last, zip,$$
$$state, city, street\_number, street\_name, apartment)$$
$$Email = (\mathbf{person\_id}, \mathbf{email\_address})$$

In this example, age is a derived attribute, so it is not included in the table, and email is a multi-valued attribute, so we have a separate table for it. This table's primary key is both the person id and their email address.



$$Dog = (\mathbf{dog\_id}, dog\_name, breed, person\_id, person\_name)$$

In this example with total participation, a dog must have one and only one owner, while a person may have multiple dogs. Thus, the dog table can include the person id.

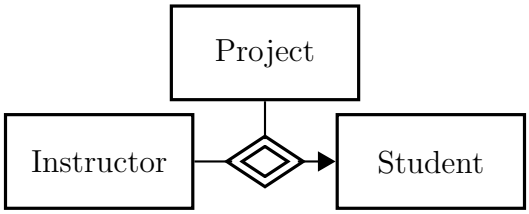


$$Person = (\mathbf{id}, name, dob, \mathbf{dog\_id}, dog\_name)$$

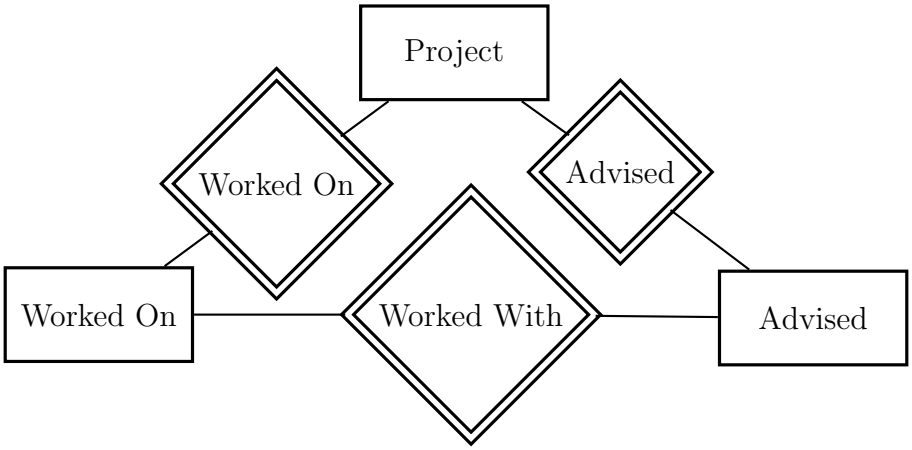
If we decide that the relationship must be one to one, then a person may have zero or one dogs, and a dog may have zero or one owners. Without total participation, both the dog id and the person's id must be part of the primary key of the table.

Ternary Relationships

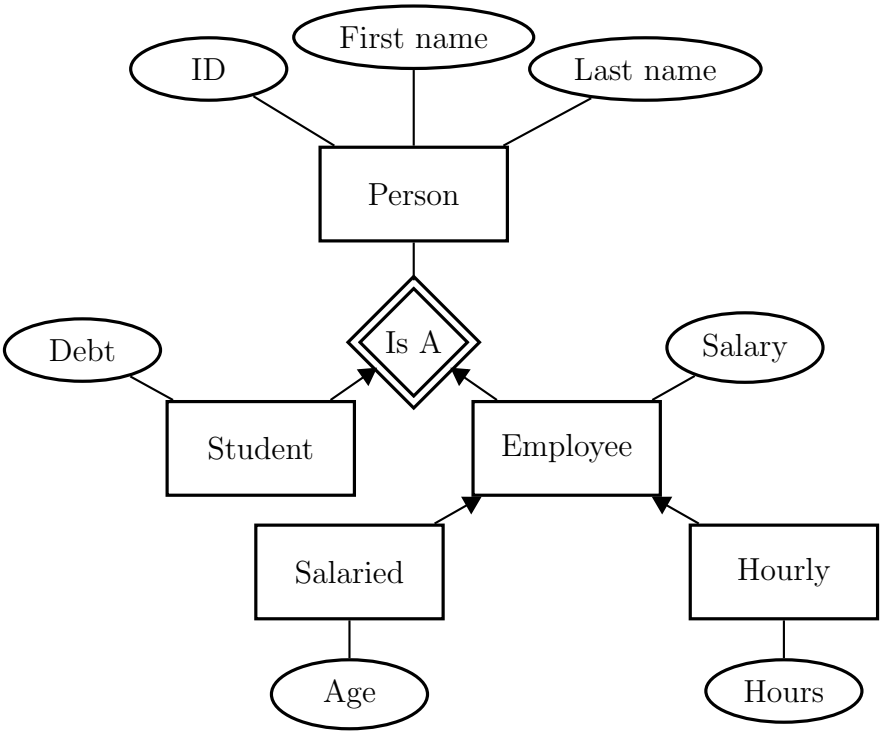
Because this can be ambiguous, we only allow for one arrow indicating ownership.



Often, it is better to reduce this to multiple binary relationships:



Sometimes, we may use a different notation for a use case like this:



When reducing this to tables we have a few options:

1. Take all the attributes from every level, and make entities with lower levels have a higher level primary key.

$$\begin{aligned} Person &= (ID, fname, lname) \\ Student &= (ID, debt) \\ Employee &= (ID, salary) \\ Salaried &= (ID, age) \\ Hourly &= (ID, hours) \end{aligned}$$

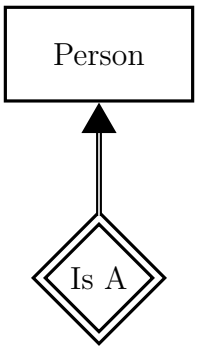
2. Create entity "leafs", grabbing attributes from higher levels.

$$\begin{aligned} Student &= (ID, fname, lname, debt) \\ Hourly &= (ID, fname, lname, salary, hours) \\ Salaried &= (ID, fname, lname, salary, age) \end{aligned}$$

3. Throw everything into one table (which is usually a bad choice).

$$Person = (ID, fname, lname, salary, hours, age, debt)$$

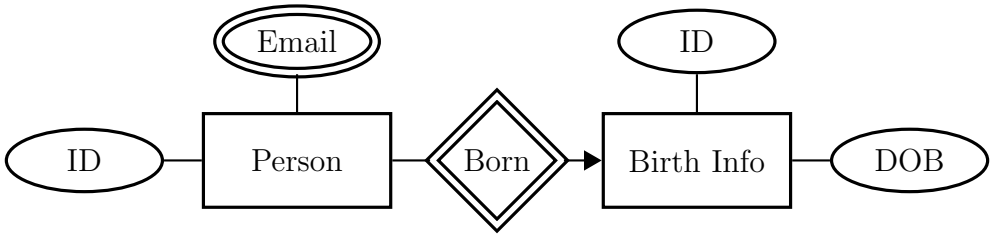
The concept of total and partial completeness still exists for these relationships. With total completeless, the higher level MUST belong to the lower level. This would be represented as:



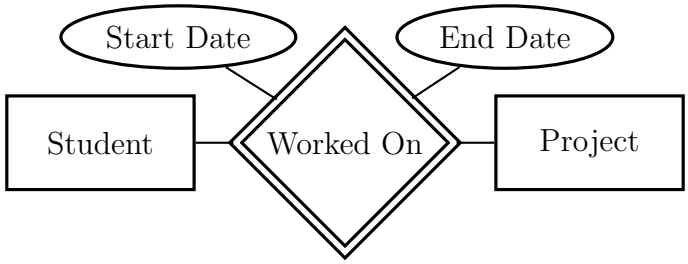
With the salaried/hourly employee dichotomy, total completeness is implied because they are disjoint categories of employees.

Entity vs Attribute

Sometimes, we may make a choice like so:



This has the advantage of reducing some redundancy as multiple people can share entries in the birth info table. For complex attributes, it may sometimes be prudent to separate out the attribute into its own entity. As another example:



You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech