Database Design

Database Design

- Designing a database is a challenging part of any software development project.
- As the database forms the basis of the whole software structure, errors in the database affect all upper software layers. (Compared to for example an error in the UI.)
- The later a possible design error in the database is noticed, the more expensive it is to fix. Modifying a database often requires changes also to the software layers built on top of the database.

ER Model

- The Entity – Relationship model (ER model) has been developed to help in designing databases.
- In an ER model, things of interest are described with three simple elements:
  - Entity type
  - Entity attributes
  - Entity relationships and relationship types
- An ER model can be converted into a relational model

ER Model: Entity Type

- The interrelated interest domain is often described verbally or in writing.
- The first task of a database designer using the ER model is to find the relationships that exist between entities, the entity types.

ER Model: Entity Type

- Here is a written example description of our Duckburg world:

A fictional world Duckburg created by Carl Banks is the home for various residents, duckburgers. For each duckburger, we know their first name, last name and the name of their pet.

ER Model: Entity Type

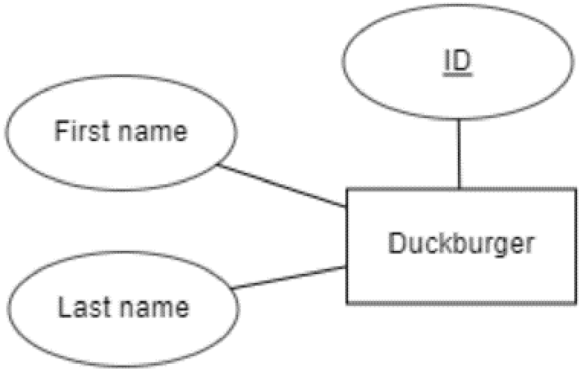
- Based on the description the database designer can identify two entity types:
  - Duckburger
  - Pet
- In the ER model entity types are visualized as a square with a descriptive name.

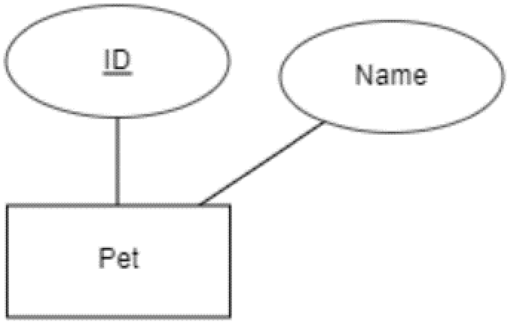
ER Model: Attributes for Entity Types

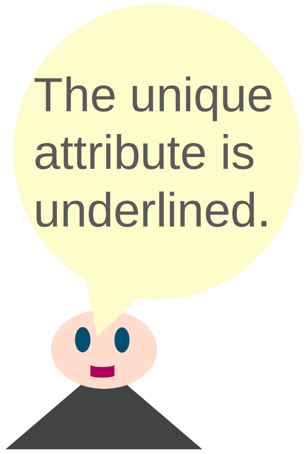
- The database designer can determine that each duckburger has a first and last name as their attributes.
- Each entity must have an unique identifier attribute. The designer contemplates whether the last name could be a good identifier. The answer is no, as there can for example be multiple residents by the name "Duck".
- What about the combination of both the first and last names? The database designer determines this as incorrect as well, since there can be multiple families with the same last name it is possible that two Duck families give the same first name to their child.
- Eventually, the designer settles upon creating a new ID attribute that uniquely identifies each citizen regardless of their name.

ER Model: Attributes for Entity Types

- Now the designer can draw a diagram of the duckburger entity type accompanied by the attributes using the ERDPlus tool.
- In the same way, the designer also identifies the pet entity type and the attributes for a pet entity.







ER Model: Entity Relationships

- The database designer notices a relationship between duckburgers and their pets: Duckburger owns a pet.
- This relationship can be drawn using a "diamond" symbol
- The relationship is given a name.
- The designer names the relationship as "owns".

ER Model: Entity Relationship Types

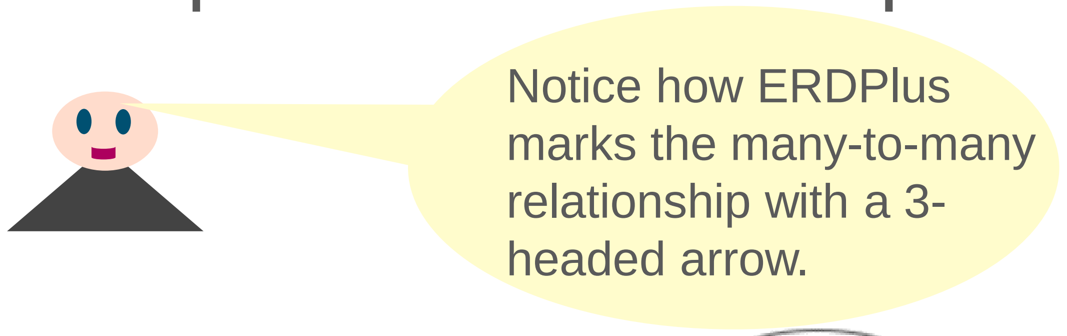
- Next the database designer must define the relationship type between duckburgers and their pets.
- For example, the family unit of Donald Duck and his nephews owns a dog called Bolivar.
- One option would be to create the model so that only Donald owns Bolivar.
- However, this would not be entirely true and we do not want to upset the nephews.

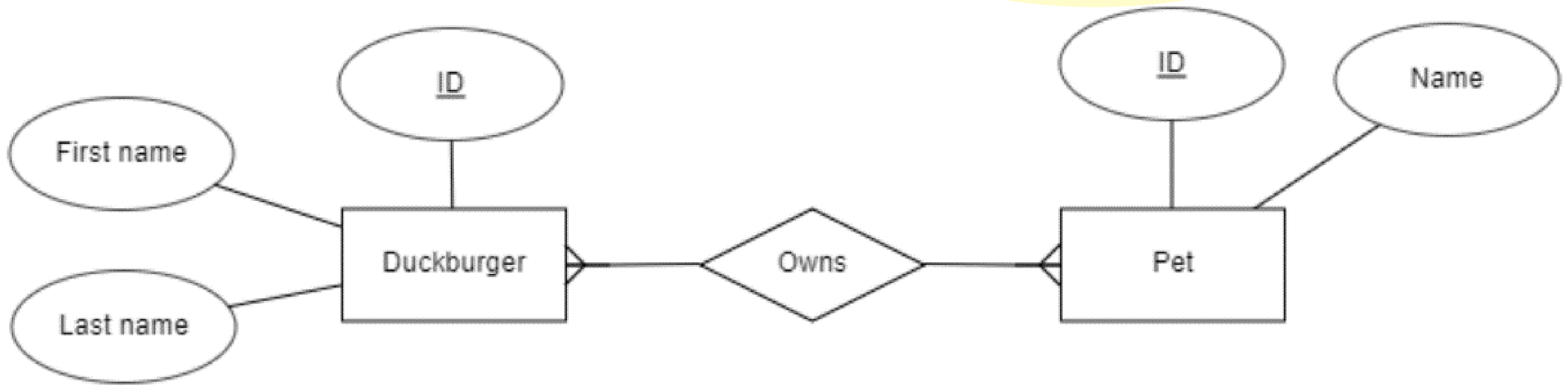
ER Model: Entity Relationship Types

- On the other hand, can each duckburger only own one pet?
- This would be an unnecessary limitation, and thus the database designer decides to allow duckburgers to own multiple pets.

ER Model: Entity Relationship Types

- The resulting relationship type is **many-to-many**. Each duckburger can own multiple pets and each pet can have multiple owners.


Notice how ERDPlus marks the many-to-many relationship with a 3-headed arrow.

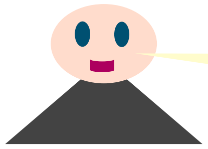


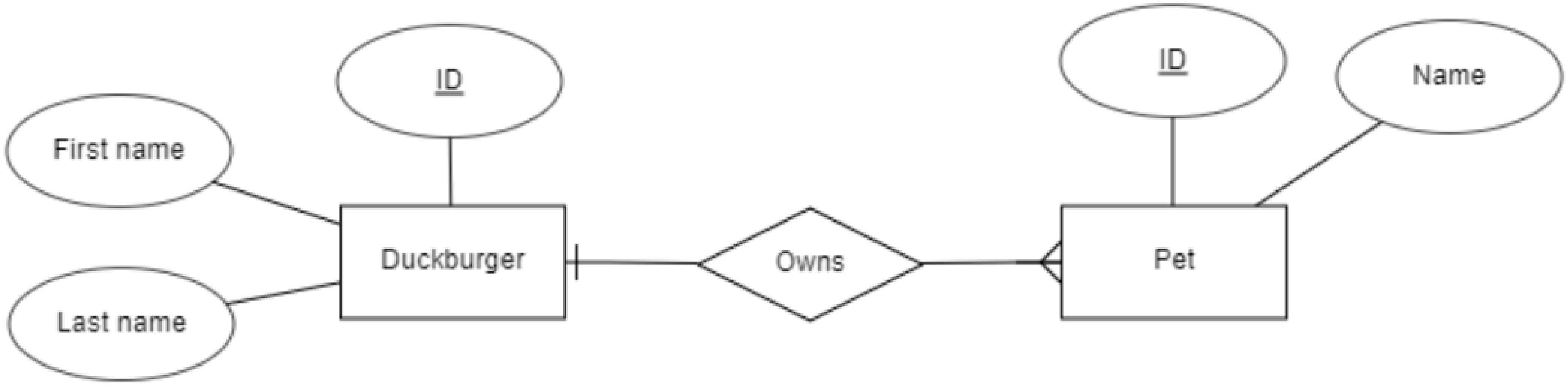
ER Model: Entity Relationship Types

- If the database designer chose a design where only Donald would own Bolivar, there would instead exist a **one-to-many** relationship between a duckburger and their pet
- This was not the case, but the relationship is introduced here as:
  - It is also commonly used.
  - The conversion of a one-to-many relationship from the ER model to the relational model is different compared to the many-to-many relationship.

ER Model: Entity Relationship Types

- Here is another example of an ER model where each duckburger can still own multiple pets but each pet only has one owner.


ERDPlus marks the "one" end of the relationship with a crossed line.

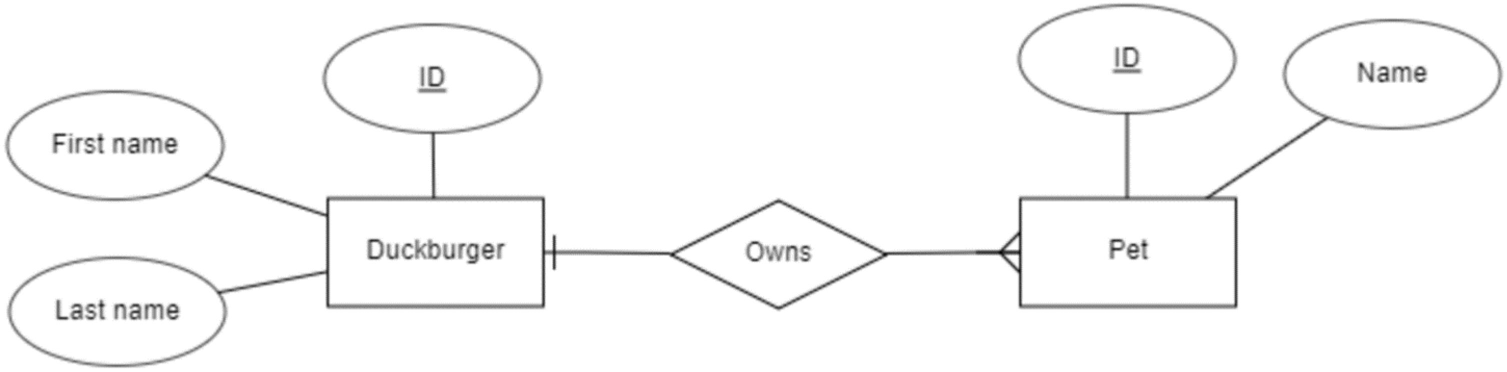


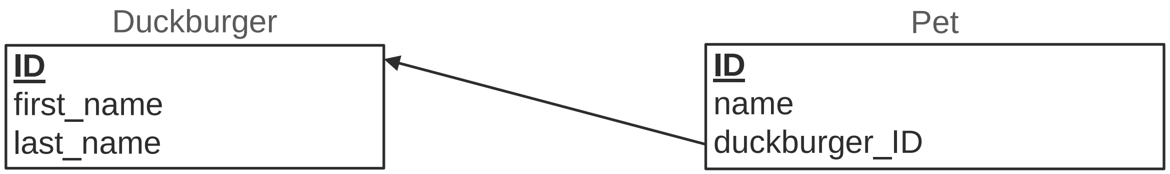
From ER Model to Relational Model

- Each entity type forms their own table in the database.
- The entity attributes are columns in the table.
  - The unique identifier attribute is used as the primary key

From ER Model to Relational Model

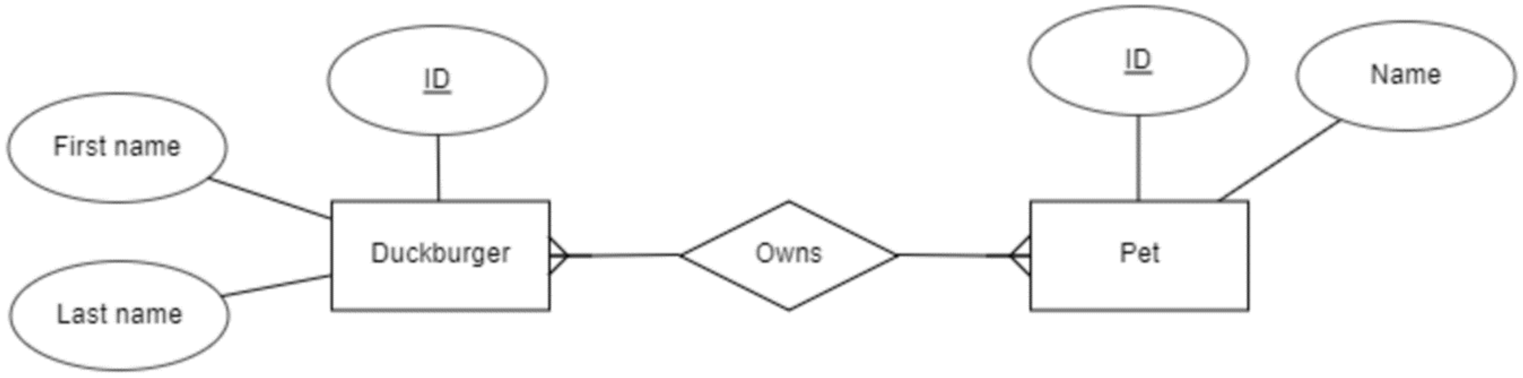
- The catch is how to convert the relationship type limitations into the table format
- In a one-to-many relationship the "many" end of the relationship gets a foreign key pointing to the "one" end.

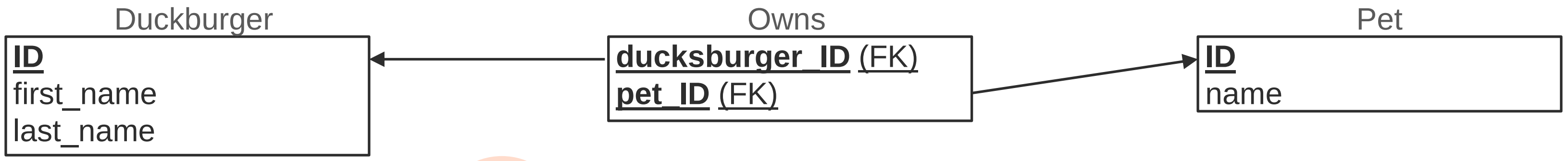




From ER Model to Relational Model

- A one-to-many is transformed by adding another table for the relationship specified inside the diamond shape. This table has two foreign keys each pointing to the the respective primary keys in the duckburger and pet tables.





Duckburger
<u>ID</u>
first_name
last_name

Owns
<u>ducksburger_ID</u> <u>(FK)</u>
<u>pet_ID</u> <u>(FK)</u>

Pet
<u>ID</u>
name