Case Study

Background

Your client, PetsCare, is a local clinic that takes care of pets, including pet wellness and vaccination, medical services, surgery including spay and neuter, dental cleaning and treatment, grooming, etc. PetsCare has served the community for more than 20 years, and has been successful in the business.

Recently, the current owner, Claire, who is the daughter of the founder of PetsCare, realized that the old file system solution is very inefficient. She and her team have to spend lots of time taking records, querying for information, and maintaining the data. She decided to turn to you, for a Relational Database Management System.

PetsCare wants to record information about customers, pets, staff, visits. Claire will be available to meet via Zoom tomorrow to discuss details. You should prepare questions for building the DBMS.

Step 1

<u>Task:</u> What questions should you ask during the meeting?

You should prepare two categories of questions:

- The interest of the business, what information should be recorded, what might be the entities for holding the information, what might be used as identities, etc.
- The relationship of the entities, especially the mandatory/optional participation, and the cardinalities. We will need them to clear confusions and to create the Entity Relationship Model.

Step 1 Solutions

Category 1: Business Interests, Entities, and Identities

Objective: To identify the "buckets" of data (Entities), the details needed inside them (Attributes), and how to uniquely find a record (Primary Keys).

#	Question	Reasoning & Objective
1.1	**What specific information do you currently collect for a "Customer"?** (e.g., Name, Address, Phone, Email, Emergency Contact, Credit Balance?)	**Objective:** Define the attributes for the Customer entity. This ensures the database captures all contact and billing data needed for communication.
1.2	**How do you currently identify a customer? Do you use a pre-existing "Account Number," or should the system generate a unique ID?**	**Objective:** Identify the **Primary Key** for the Customer entity. We need a unique identifier that never changes, unlike a phone number or name.
1.3	**For "Pets," besides Name and Species, what medical-specific data is vital?** (e.g., Breed, Date of Birth, Gender, Weight, Microchip ID, Allergies?)	**Objective:** To determine the attributes of the Pet entity. "Microchip ID" is particularly important as a potential **Candidate Key**.
1.4	**How do you categorize your "Staff"?** (e.g., Veterinarians, Groomers, Techs, Receptionists?)	**Objective:** To determine if Staff should be one table with a "Role" attribute or if different roles need separate tables due to different data requirements (e.g., Vets have license numbers).
1.5	**What constitutes a "Service"? Do you have a fixed list of prices?** (e.g., "Grooming - Small Dog - $40")	**Objective:** To identify a Service or Procedure Catalog entity. This prevents staff from typing prices manually, which reduces errors.
1.6	**When a "Visit" occurs, what is recorded?** (e.g., Date, Time, Reason for visit, Diagnosis, Total Cost?)	**Objective:** Define the Visit entity. This is a transaction-based entity that links all other data points together.

Category 2: Relationship of Entities, Participation, and Cardinalities

Objective: To determine how entities interact. This defines the "lines" in our ER Diagram, including whether a relationship is 1-to-Many or Many-to-Many.

#	Question	Reasoning & Objective
2.1	**Can a pet have more than one owner (Customer) in your system?**	**Objective:** To determine **Cardinality**. If one owner has many pets, it's 1:M. If multiple owners share a pet, it's M:N, which requires a "Linking Table."
2.2	**Is it possible to have a "Customer" in your system who does not yet have a "Pet" registered?**	**Objective:** To determine **Participation (Optional vs. Mandatory)**. This tells us if a Customer record can exist without a foreign key link to a Pet.
2.3	**During a "Visit," can one pet be treated by multiple "Staff" members?** (e.g., a Vet for surgery and a Tech for recovery?)	**Objective:** To define the relationship between Staff and Visit. This helps determine if we need a many-to-many relationship for staff assignments per appointment.
2.4	**Can one "Visit" record include multiple pets, or does each pet get its own unique Visit ID?**	**Objective:** To clarify the **Degree of the Relationship**. Most systems prefer 1 Pet per Visit for medical record accuracy, but businesses often want 1 Bill for multiple pets.
2.5	**When a "Visit" is created, is it mandatory to assign a "Service" immediately?**	**Objective:** To determine **Participation Constraints**. Can a visit exist as an "Appointment" (empty of services) or is it only created once services are performed?
2.6	**If a "Pet" is deleted (e.g., passes away or moves), what happens to the "Visit" history?**	**Objective:** To understand **Referential Integrity** and **Archiving Rules**. We need to know if we should "Cascade Delete" or keep historical records for tax/medical auditing.

Step 2: Build the Entity Relationship Model

Congratulations! Thanks to the efficient communication between Claire and you, now you understand the details of the business, and you are ready to write down narrative statements for your Entity Relationship Model.

You should lay out your own assumptions and your own ER Model.

Step 2 Solution

After several rounds of discussion, below is the one confirmed with the client.

Note: Don't worry if your ER model is different than this one -- as long as you have clear assumptions, and your model is built up these assumptions).

Here is what you summarized about the entities:

- Customers: CustomerID, FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate. CustomerID is the identifier.
- Pets: CustomerID, Pet#, NickName, Address, Email, Phone, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes. CustomerID and Pet# together as the identifier.
- Staff: EmployeeID, FirstName, LastName, SSN, Address, Email, Phone, DoB. EmployeeID is the identifier.
- Visit: VisitID, Date, Time, Customer, Pet, ServiceID, ServiceName, ServicePrice, ServiceDescription, Staff, Bill, Paid. VisitID is the identifier.

Their relationships are:

- A customer may have one or more (zero or more) pets; A pet must belong to one and only one (exactly one) customer.
- A staff may treat one or more pets (zero or more); and A pet may be familiar with one or more (zero or more) staff.
- A customer may have one or more (zero or more) visits; and each visit must be done by one and only one (exactly one) customer.
- A pet may have one or more (zero or more) visits; and each visit may involve one (zero or one) pet.
- A staff must be in charge of one or more (one or more) visits; and a visit must be charged by one and only one (exactly one) staff.

- A customer may be referred by one (zero or one) customer; and a customer may refer one or more (one or more) customers.
- A staff must have one and only one (exactly one) supervisor; A staff may supervise one or more (zero or more) staff.

Step 3: Create the Entity Relationship Diagram

In step 2, we got:

Here is what you summarized about the entities:

- Customers: CustomerID, FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate. CustomerID is the identifier.
- Pets: CustomerID, Pet#, NickName, Address, Email, Phone, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes. CustomerID and Pet# together as the identifier.
- Staff: EmployeeID, FirstName, LastName, SSN, Address, Email, Phone, DoB. EmployeeID is the identifier.
- Visit: VisitID, Date, Time, Customer, Pet, ServiceID, ServiceName, ServicePrice, ServiceDescription, Staff, Bill, Paid. VisitID is the identifier.

Their relationships are:

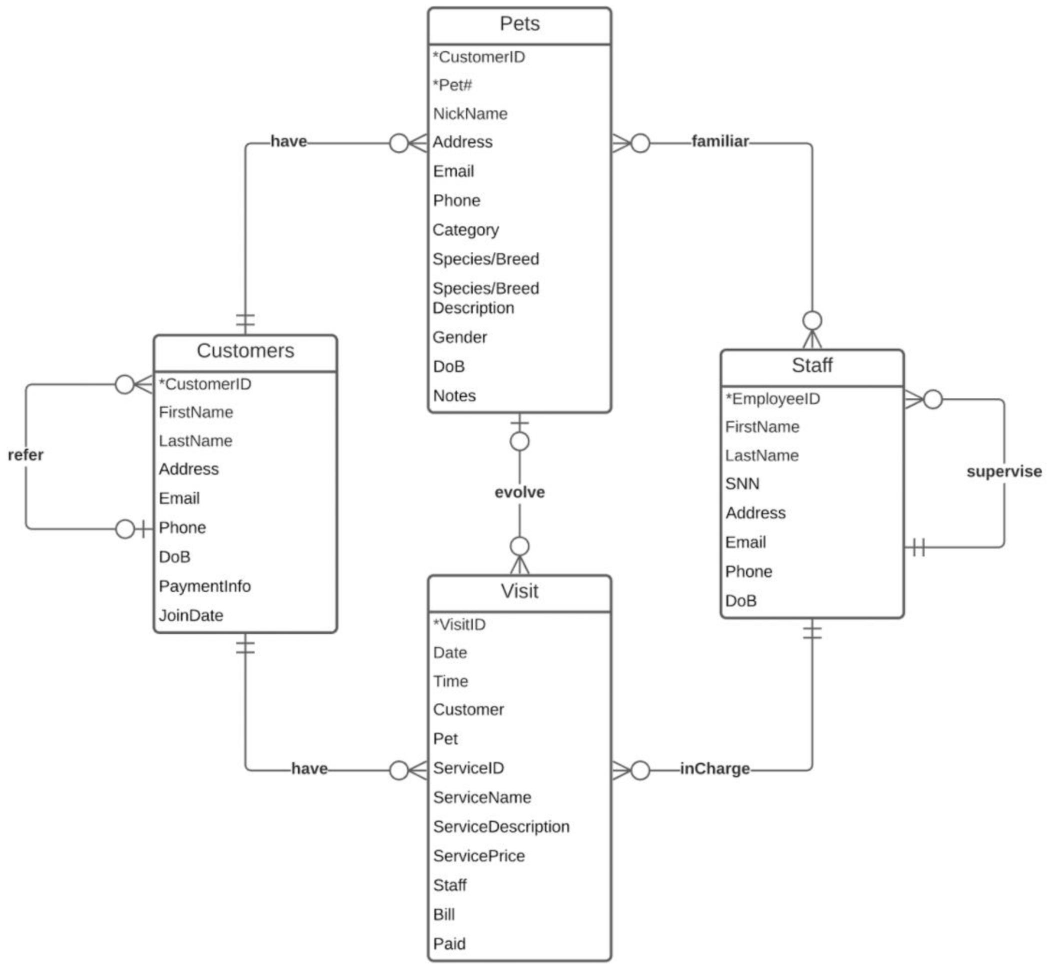
- A customer may have one or more (zero or more) pets; A pet must belong to one and only one (exactly one) customer.
- A staff may treat one or more pets (zero or more); and A pet may be familiar with one or more (zero or more) staff.
- A customer may have one or more (zero or more) visits; and each visit must be done by one and only one (exactly one) customer.
- A pet may have one or more (zero or more) visits; and each visit may involve one (zero or one) pet.
- A staff may be in charge of one or more (one or more) visits; and a visit must be charged by one and only one (exactly one) staff.
- A customer may be referred by one (zero or one) customer; and a customer may refer one or more (one or more) customers.

- A staff must have one and only one (exactly one) supervisor; A staff may supervise one or more (zero or more) staff.

Now you have collected the information needed for the ER Model, and you are ready to draw the ERD.

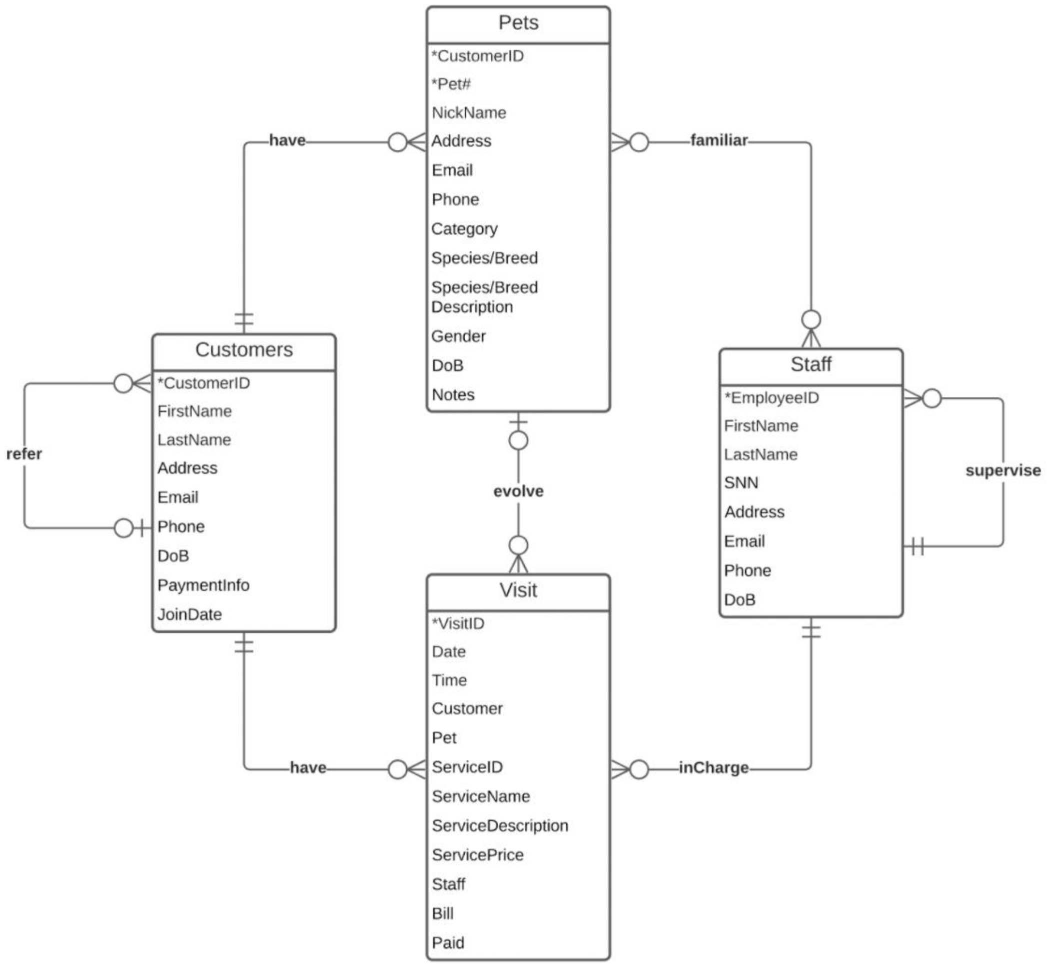
Step 3 Solution

Here is the solution



Step 4

In step 3, we got:



Now you can convert the ERD to a Relational Model.

Step 4 Solution

Here is the relational model you converted from the ERD:

- Customers (CustomerID, FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate, ReferredByCustomerID(fk)).
- Pets (CustomerID(fk), Pet#, NickName, Address, Email, Phone, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes).
- Staff (EmployeeID, FirstName, LastName, SSN, Address, Email, Phone, DoB, SupervisorID(fk)).
- Visit (VisitID, Date, Time, CustomerID(fk), Pet(fk), ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID(fk), Bill, Paid).
- Pets_Staff(CustomerID(fk), Pet#(fk),EmployeeID(fk))

Step 5

Normalize the Relational Model to 3NF

Before normalization, you should ask Claire about the Functional Dependencies of the relations. Here is what you got from her:

- Customers (CustomerID, FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate, ReferredByCustomerID(fk)).
  - FD1: CustomerID → FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate, ReferredByCustomerID
- Pets (CustomerID(fk), Pet#, NickName, Address, Email, Phone, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes).
  - FD1: CustomerID, Pet# → NickName, Address, Email, Phone, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes.
  - FD2: CustomerID → Address, Email, Phone
  - FD3: Species/Breed → Species/Breed Description
- Staff (EmployeeID, FirstName, LastName, SSN, Address, Email, Phone, DoB, SupervisorID(fk)).
  - FD1: EmployeeID → FirstName, LastName, SSN, Address, Email, Phone, DoB, SupervisorID
- Visit (VisitID, Date, Time, CustomerID(fk), Pet(fk), ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID(fk), Bill, Paid).

    - FD1: VisitID → Date, Time, CustomerID, Pet, ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID, Bill, Paid
    - FD2: ServiceID → ServiceName, ServicePrice, ServiceDescription.
- Pets_Staff(<u>CustomerID(fk)</u>, <u>Pet#(fk)</u>,<u>EmployeeID(fk)</u>)
    - There is no non-primary-key attribute.

Now you can normalize this Relational Model to 3NF.

Step 5 Solutions

Normalize the Relational Model to 3NF

Here is the normalization process:

- Customers, Staff, and Pets_Staff relations are in 3NF, because they are in 1NF; they have no partial functional dependencies so they are in 2NF; and they have no transitive functional dependencies so they are in 3NF.
- Pets relation is in 1NF. However, it is not in 2NF because FD2: CustomerID → Address, Email, Phone. CustomerID as part of the primary key, determines non-primary-key attributes. This leads to a partial functional dependency. We need to normalize Pets to 2NF:
    - Create a new relation to put CustomerID, Address, Email, Phone. Since Customer relation has these attributes, we can simply remove them from Pets, and keep CustomerID as a foreign key.
    - Pets (<u>CustomerID(fk)</u>, <u>Pet#</u>, NickName, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes).
        - FD1: CustomerID, Pet# → NickName, Category, Species/Breed, Species/Breed Description, Gender, DoB, Notes.
        - FD2: Species/Breed → Species/Breed Description
- Pets relation now is in 2NF. However, it is not in 3NF because of FD2: Species/Breed → Species/Breed Description. (CustomerID, Pet#) →Species/Breed, and Species/Breed → Species/Breed Description is a transitive functional dependency. We need to normalize Pets to 3NF:
    - Create a new relation to put Species/Breed and Species/Breed Description and modify Pets:

  - Species (<u>Species/Breed</u>, Species/Breed Description)
    - FD1: Species/Breed → Species/Breed Description
  - Pets (<u>CustomerID(fk)</u>, <u>Pet#</u>, NickName, Category, Species/Breed(fk), Gender, DoB, Notes).
    - FD1: CustomerID, Pet# → NickName, Category, Species/Breed, Gender, DoB, Notes.
- Now Pets relation is in 3NF.
- Visit relation is in 1NF, and 2NF. However, it is not in 3NF because of FD2: ServiceID → ServiceName, ServicePrice, ServiceDescription. VisitID → ServiceID, and ServiceID → ServiceName, ServicePrice, ServiceDescription is a transitive functional dependency. We need to normalize Visit to 3NF:
  - Create a new relation to put ServiceID, ServiceName, ServicePrice, ServiceDescription and modify visit.
  - Service (<u>ServiceID</u>, ServiceName, ServicePrice, ServiceDescription)
    - FD1: ServiceID → ServiceName, ServicePrice, ServiceDescription
  - Visit (<u>VisitID</u>, Date, Time, CustomerID(fk), Pet(fk), ServiceID(fk), EmployeeID(fk), Bill, Paid).
    - FD1: VisitID → Date, Time, CustomerID, Pet, ServiceID, EmployeeID, Bill, Paid

Step 6

Final Output for Implementation

After the normalization process, summarize the relational model in 3NF.

Step 6 Solutions

Now the final Relational Model in 3NF is as below:

- Customers (<u>CustomerID</u>, FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate, ReferredByCustomerID(fk)).
    - FD1: CustomerID → FirstName, LastName, Address, Email, Phone, DoB, PaymentInfo, JoinDate, ReferredByCustomerID
- Pets (<u>CustomerID(fk)</u>, <u>Pet#</u>, NickName, Category, Species/Breed(fk), Gender, DoB, Notes).
    - FD1: CustomerID, Pet# → NickName, Category, Species/Breed, Gender, DoB, Notes.
- Species (<u>Species/Breed</u>, Species/Breed Description)
    - FD1: Species/Breed → Species/Breed Description
- Staff (<u>EmployeeID</u>, FirstName, LastName, SSN, Address, Email, Phone, DoB, SupervisorID(fk)).
    - FD1: EmployeeID → FirstName, LastName, SSN, Address, Email, Phone, DoB, SupervisorID
- Visit (<u>VisitID</u>, Date, Time, CustomerID(fk), Pet(fk), ServiceID(fk), EmployeeID(fk), Bill, Paid).
    - FD1: VisitID → Date, Time, CustomerID, Pet, ServiceID, EmployeeID, Bill, Paid
- Service (<u>ServiceID</u>, ServiceName, ServicePrice, ServiceDescription)
    - FD1: ServiceID → ServiceName, ServicePrice, ServiceDescription
- Pets_Staff(<u>CustomerID(fk)</u>, <u>Pet#(fk)</u>,<u>EmployeeID(fk)</u>)
    - There is no non-primary-key attribute.

Congratulations! Now you have a Relational Model in 3NF, and can be implemented as a real database. Good job!