**Normalization Example**

Our first task is to present the data in a tabular format as shown below.

Looking at this data, we can see that we are not in first normal form because we have no keys, repeating groups and multi-valued fields.

Things you should consider to understand why this data is not normalized:

* What happens when a customer has a fifth pet? Do we re-size the entire database to add that column? What about a sixth, seventh or more?
* When most customers only have one or two pets, we still have additional space being used for pet 3, pet 4 and so on.
* How do we search for values in a multi-valued field like visits? This can be a processing nightmare and involves a lot of overhead.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Client** | **Address** | **Phone** | **Pet 1** | **Pet 2** | **Pet 3** | **Pet 4** | **Visits** |
| Mary Jones | 55 Rhodes St | 555-290-3083 | Boomer, Chihuahua | Trixie, Schnauzer | Fred, Mixed |  | March 10 at 2:00pm (check up)  Boomer, Trixie, Fred  March 25 at 8:00am (spay)  Trixie |
| Jerome Franklin | 37583 Respite Pines Lane | 555-450-4999 | Esmerelda, Bulldog |  |  |  | May 27 at 1:00pm (check up)  June 15 at 8:00am (grooming)  August 5 at 8:00am (grooming) |
| Pat Cooper | 1250 50th Avenue | 555-408-3803 | Snots, Mixed | Spot, Mixed | Sam, Poodle | Suzy, Great Dane | September 12 at 4:00pm (check up)  Snots, Suzy |

How do we get our vet database to first normal form (1NF)? To be in first normal form we need:

* Unique primary key
* One set of values per column
* One value per cell
* Eliminate repeating data

To improve upon this, we will start by normalizing the data into first normal form.

1nf:

* Each table cell should contain a single value.
* Each record needs to be unique.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Client** | **Address** | **Phone** | **Pet** | **Breed** | **Visits** |
| Mary Jones | 55 Rhodes St | 555-290-3083 | Boomer | Chihuahua | March 10 at 2:00pm (check up) |
| Mary Jones | 55 Rhodes St | 555-290-3083 | Trixie | Schnauzer | March 10 at 2:00pm (check up) |
| Mary Jones | 55 Rhodes St | 555-290-3083 | Trixie | Schnauzer | March 25 at 8:00am (spay) |
| Mary Jones | 55 Rhodes St | 555-290-3083 | Fred | Mixed | March 10 at 2:00pm (check up) |
| Jerome Franklin | 37583 Respite Pines Lane | 555-450-4999 | Esmerelda | Bulldog | May 27 at 1:00pm (check up) |
| Jerome Franklin | 37583 Respite Pines Lane | 555-450-4999 | Esmerelda | Bulldog | June 15 at 8:00am (grooming) |
| Jerome Franklin | 37583 Respite Pines Lane | 555-450-4999 | Esmerelda | Bulldog | August 5 at 8:00am (grooming) |
| Pat Cooper | 1250 50th Avenue | 555-408-3803 | Snots | Mixed | September 12 at 4:00pm (check up) |
| Pat Cooper | 1250 50th Avenue | 555-408-3803 | Spot | Mixed |  |
| Pat Cooper | 1250 50th Avenue | 555-408-3803 | Sam | Poodle |  |
| Pat Cooper | 1250 50th Avenue | 555-408-3803 | Suzy | Great Dane | September 12 at 4:00pm (check up) |

When we look at the data normalized to first normal form, we see that we still have some issues.

* Insertion anomalies
  + data about more than one entity in the relation forces you to insert data about an unrelated entity
* Deletion anomalies
  + part of the primary key of a row becomes null when the data are deleted, forcing you to remove the entire row. The result of a deletion anomaly is the loss of data that you would like to keep.
* Update anomalies
  + if every row is not changed, then data that should be the same are no longer the same. The potential for these inconsistent data is the modification anomaly

To alleviate some of the issues we find in first normal form, we will continue normalizing the data to second normal form.

2nf:

* The relation is in first normal form.
* All non-key attributes are functionally dependent on the entire primary key.

We start by isolating each group of data into its own entity. What is functionally dependent upon client id? The information about the client itself (name, address, phone):

**Client**

|  |  |  |  |
| --- | --- | --- | --- |
| **ClientID** | **Client** | **Address** | **Phone** |
| 1 | Mary Jones | 55 Rhodes St | 555-290-3083 |
| 2 | Jerome Franklin | 37583 Respite Pines Lane | 555-450-4999 |
| 3 | Pat Cooper | 1250 50th Avenue | 555-408-3803 |

The pets can be isolated to their own entity as well. We’ll use the primary key from the client entity, ClientID, to tie the clients to their pets. Remember – they could have multiple pets. This structure allows any given client to have any number of entries in the pet entity without worrying about having to resize the database again and again.

**Pet**

|  |  |  |  |
| --- | --- | --- | --- |
| **PetID** | **Pet** | **Breed** | **ClientID** |
| 1 | Boomer | Chihuahua | 1 |
| 2 | Trixie | Schnauzer | 1 |
| 3 | Fred | Mixed | 1 |
| 4 | Esmerelda | Bulldog | 2 |
| 5 | Snots | Mixed | 3 |
| 6 | Spot | Mixed | 3 |
| 7 | Sam | Poodle | 3 |
| 8 | Suzy | Great Dane | 3 |

The visits can be isolated to their own entity. We’ll use the primary key from the client entity, ClientID, to tie the clients to their visits. Remember – they could have multiple pets. This structure allows any given client to have any number of visit records.

**Visit**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VisitID** | **VisitDate** | **VisitTime** | **VisitReason** | **ClientID** |
| 1 | March 10 | 2:00pm | Check Up | 1 |
| 2 | March 25 | 8:00am | Spay | 1 |
| 3 | May 27 | 1:00pm | Check Up | 2 |
| 4 | June 15 | 8:00am | Grooming | 2 |
| 5 | August 5 | 8:00am | Grooming | 2 |
| 6 | September 12 | 4:00pm | Check Up | 3 |

Next we’ll deal with how pets are tied to their visits. Recall that our conceptual diagram depicted a many-to-many relationship between pets and visits. In order to create this type of relationship, we need another table to serve as the go between so that one pet can tie to zero or more visit records and one visit record can tie to one or more pets. We can accomplish this by creating a new table as shown below.

**PetVisit**

|  |  |
| --- | --- |
| **PetID** | **VisitID** |
| 1 | 1 |
| 2 | 1 |
| 2 | 2 |
| 3 | 1 |
| 4 | 3 |
| 4 | 4 |
| 4 | 5 |
| 5 | 6 |
| 8 | 6 |

In order to reach third normal form, we are going to break out the pets and their breeds. In theory, the vet could store informaton about various breeds unrelated to the actual client’s pets. So we will create a new entity to store breeds and modify the pet entity to relate to it.

3nf

* The relation is in second normal form.
* There are no transitive dependencies.

**Pet**

|  |  |  |  |
| --- | --- | --- | --- |
| **PetID** | **Pet** | **BreedID** | **ClientID** |
| 1 | Boomer | 2 | 1 |
| 2 | Trixie | 6 | 1 |
| 3 | Fred | 4 | 1 |
| 4 | Esmerelda | 1 | 2 |
| 5 | Snots | 4 | 3 |
| 6 | Spot | 4 | 3 |
| 7 | Sam | 5 | 3 |
| 8 | Suzy | 3 | 3 |

**Breeds**

|  |  |
| --- | --- |
| **BreedID** | **Breed** |
| 1 | Bulldog |
| 2 | Chihuahua |
| 3 | Great Dane |
| 4 | Mixed |
| 5 | Poodle |
| 6 | Schnauzer |
| 7 | Siberian Husky |
| 8 | Shih Tzu |

Note: For the purposes of this demonstration, two additional dog breeds are listed here to show the scenario where the vet database may store more breeds than shown in their pet data.

Keep in mind that this part of the modeling process requires you to think about the scenarios that might exist using sample data but you’re not dealing with the exact data when designing a database. Multiply this scenario by thousands of records for any given entity and you’ll easily see why we use such small samples to work through this process.

Next will document our logical design. The logical design will depict the entities and relationships but will also include the attributes and primary/foreign key information for this normalized design.

**The Logical Diagram for our normalized vet database:**

We will document the following entities to include: clients, pets, visits, petvisit, and breeds.

