Normal Forms

So last week be began to normalize our data which is the process of separating the data into related tables. Normalization goes on to then reduce data redundancy. If data is repeated when it shouldn't be it can cause storage and maintenance problems.

When a table contains data that is unnormalized it might contain data on two or more entities. It could also contain repeating values or columns or data that is repeated in two or more rows.

There are 7 normal forms and you must apply them in order as you go through the process of normalization. We will be going over just the first 3 for now. Usually your database will be considered normalized at that point. The 4th through the 6th are applied infrequently so we will just briefly describe them.

First normal form states that "The value stored at the intersection of each row and column must be a scalar value, and table must not contain any repeating columns. Scalar means – being described as a single value that might belong to a number of fields, but itself is one value.

Order#	**Cust#**	**Last**	**First**	**Address**	**Zip**	**Order_date**	**Products**
05	123	Smith	Sue	60 Main	83406	12/14/19	2 bracelets, necklace, earrings
06	124	Thomas	Tom	99 South St.	84304	01/05/20	ring, watch
07	123	Smith	Sue	60 Main	83406	01/20/20	tiara

This table doesn't follow 1NF because the products field has multiple values. We can also see that there is probably more than one entity involved here as well, but that will be solved as we normalize as well.

Order#	**Cust#**	**Last**	**First**	**Address**	**Zip**	**Order_date**	**Product1**	**Product2**	**Product3**
05	123	Smith	Sue	60 Main	83406	12/14/19	2 bracelets	necklace	earrings
06	124	Thomas	Tom	99 South St.	84304	01/05/20	ring	watch	null
07	123	Smith	Sue	60 Main	83406	01/20/20	tiara	null	null

What if we just separate those products into their own column? Are we now in first normal form. No, remember it states also that "The table must not contain any repeating columns" We are repeating the products again and again. How can we predict how many product columns we should have. If a customer orders a hundred different products should be have 100 columns ready to go? Then if another person only orders one product, they'd have 99 columns with a null value. It just doesn't make sense.

Now it is in 1NF. There is only one value in each cell and no columns are repeating.

Order#	**Cust#**	**Last**	**First**	**Address**	**Zip**	**Order_date**	**Products**
05	123	Smith	Sue	60 Main	83406	12/14/19	2 bracelets
05	123	Smith	Sue	60 Main	83406	12/14/19	necklace
05	123	Smith	Sue	60 Main	83406	12/14/19	earrings
06	124	Thomas	Tom	99 South St.	84304	01/05/20	ring
06	124	Thomas	Tom	99 South St.	84304	01/05/20	watch

07	123	Smith	Sue	60 Main	83406	01/20/20	tiara

But we can see a problem because now some of the data in each row is repeated and our primary key is not unique. Even if we used Order# and Cust# as a composite, it's still not unique. So let's see if 2NF will help with this.

Second Normal Form 2NF states that "Every non-key column must depend on the entire primary key. " (this has to do with composite keys where each field in the row has to rely on composite primary key)

Even if we used Order# and Cust# as a composite key that uniquely identifies each row, each field does not rely on entire primary key.

So let's break it down to two different entities and that will solve one table—the customer table.

Cust#	Last	First	Address	Zip
123	Smith	Sue	60 Main	83406
124	Thomas	Tom	99 South St.	84304

So Customer is in 2NF, but not the Order table.

So we need to apply the 3NF now. 3NF states that "Every non-key must depend only on the primary key.

The Order table still has a non-key (products) that doesn't depend on the primary key. Product names don't depend on the order number. They can be used in lots of different orders. But our order date does depend on that particular order and the foreign key of customer needs to be there to relate the two tables. And again our primary key is repeating which is not following what primary keys do—they need to be unique.

Order#	Cust#	Order_date	Products
05	123	12/14/19	2 bracelets
05	123	12/14/19	necklace
05	123	12/14/19	earrings
06	124	01/05/20	ring
06	124	01/05/20	watch
07	123	01/20/20	tiara

If you see a column that doesn't depend on the primary key, you separate it out into it's own table.

Product_id	Name	Quantity
004	bracelet	2
005	necklace	1
006	earrings	1
007	ring	1
008	watch	1

009	tiara	1

Order#	Cust#	Order_date
05	123	12/14/19
06	124	01/05/20
07	123	01/20/20

So now we have a products table, but the products table is still not in 3NF. The quantity field does not depend on the primary key. Also what is the relationship between these two tables. (1:1, 1:M, or M:M)? It's many-to-many. Orders can have many products and products can belong to many orders. So to resolve that and help put it into 3NF at the same time we are going to create a linking table.

Order#	Cust#	Order_date
05	123	12/14/19
06	124	01/05/20
07	123	01/20/20

OrderLine#	Order#	Product_id	Quantity
1	05	004	2
2	05	005	1
3	05	006	1
4	06	007	1
5	06	008	1
6	07	009	1

Product_id	Name	Price
004	bracelet	25.99
005	necklace	35.99
006	earrings	15.99
007	ring	129.99
008	watch	89.99
009	tiara	1999.99

Now we can place quantity inside the linking table where it makes more sense. Now each table is in 3NF. I did add a field onto products called price which does depend on the product_id so that fits. And you could also have other fields like product category, etc.

Also notice the linking table, I didn't necessarily need a new orderline# field as primary key, I could have used the order# and the product_id as a composite key and that would have worked as well.

Boyce-Codd normal form (which is named after Codd who invented the relational database model and Boyce who helped create SQL and came up with this normal form) The Boyce Codd states that "A non-key column can't be dependent on another non-key column" For example you could say that with the zipcode you can look up city and state for city and state are dependent on zipcode. So really you could store city and state in a separate table and use zip code to look them up.

4NF states that "A table must not have more than one multivalued dependency." If you wanted to store the customers phone numbers and they had a cell number and a home number and a work number they you've created a multivalued dependency on the customer_id primary key. To be in 4NF you have to store the phone numbers in a separate table and use the customer_id as a foreign key.

5NF exists when “The tables have been split until there is redundancy, but they can still be joined to recreate the original.”

6NF states that “Every constraint on the relationship is dependent only on key constraints and domain constraints” This normal form has not been implemented in a database. It is only an abstract idea and not a practical design model. It is only of academic interest and is beyond the scope of this class.

So there we have the Normal Forms that helps us create a proper database.