Normalizing a Database

(Another Perspective)

By Nathan Hardy

Welcome to the wonderful world of databases, as well as the normalization process. To normalize a database means to increase its flexibility, as well as protection against anomalies.

Hopefully you will learn the process by which a database is put into the 1st, 2nd and 3rd normal forms.

First of all, there are actually 7 normal forms in the order as follows-

First Normal Form	(1NF)
Second Normal Form	(2NF)
Third Normal Form	(3NF)
Boyce-Codd Normal Form	(BCNF)
Fourth Normal Form	(4NF)
Fifth Normal Form	(5NF)
Domain/Key Normal Form	(DKNF)

But you don't have to worry about anything beyond 3rd normal form, as this is the most commonly used of the normal forms.

1st Normal Form

First rule of first normal form is basically data must be in a database (tables), makes sense doesn't it. Beyond this are a few other rules which must be observed.

1. Each column must have a unique name
2. Each column must have only one data type
3. No two rows can have identical data
4. Each column can have only a single value
5. Columns cannot contain repeating groups

So what does all that mean at this time, well the first two rules are pretty self-explanatory, and automatically observed if you use software like Access, SQL Server, etc... So we can skip these rules and concentrate on the remaining.

The third rule simply means no row can have values identical to another row, otherwise how would you tell them apart. So this again is a fairly easy to follow, and to confirm yourself. Now let us focus on the last two rules of 1st normal form.

Moving onto the fourth rule, you can see it is pretty easy to confirm. If a cell contains more than one entry, then we have to move this info into its own row. To help to illustrate this point an example-**1-1**

Example 1-1

Name	**Phone**	**Address**	**Email**
Jonny Appleseed	888-123-5675, 908-123-3456	2324 west	ruty@ruty.com

Dave Crocket	990-678-3456, 801-334-3456	123 Fake street	mo@email.com
Paul Bunyan	678-453-6754, 567-345-2178	742 Nowhereville	blue@ox.com

As you can see in example 1-2, the phone numbers are now contained in multiple rows, even though it seems counter-intuitive, you can do it this way.

Example 1-2

Name	Phone	Address	Email
Jonny Appleseed	888-123-5675	2324 west	ruty@ruty.com
Jonny Appleseed	908-123-3456	2324 west	ruty@ruty.com
Dave Crocket	990-678-3456	123 Fake street	mo@email.com
Dave Crocket	801-334-3456	123 Fake street	mo@email.com
Paul Bunyan	678-453-6754	742 Nowhereville	blue@ox.com
Paul Bunyan	567-345-2178	742 Nowhereville	blue@ox.com

Though it seems this is adding duplicate data, it is one way to achieve first normal form. Another way (some would say better) will be discussed after the last rule.

The last rule sounds complicated but is really quite simple. So what does it mean then to us? Well, using the previous table as an example you might have had the thought to add the emails to more columns. Such as phone1, phone2, phone3 etc.... This is what we mean by a repeating group, and again you can separate them like example 1-2.

But wait, didn't I mention a better way, well what would that be. Well the very foundation of a relational database is a relation. But how does that apply here you say, take a look at example 1-3.

Example 1-3

Customer Table

Name	Address	Email
Jonny Appleseed	2324 west	ruty@ruty.com
Dave Crocket	123 Fake street	mo@email.com
Paul Bunyan	742 Nowhereville	blue@ox.com

Phone Table

Name	Phone
Jonny Appleseed	888-123-5675
Jonny Appleseed	908-123-3456

Dave Crocket	990-678-3456
Dave Crocket	801-334-3456
Paul Bunyan	678-453-6754
Paul Bunyan	567-345-2178

As you can see in the previous example (example 1-3), we take out the repeating values and place them in their own table. We also place a copy of the primary key of the other table, giving us a relation back to our main table.

2nd Normal Form

The first and foremost rule of 2nd normal form is that the database must be in 1st normal form. Easy enough if you follow the rules to achieve this. The second rule is that non-key fields must depend completely on key fields (our concatenated key). But wait, what exactly does that mean? Again an example is the best way to learn this, so take a look at an example that needs a composite key (example 2-1).

Example 2-1

Time	Band Name	Style	No. Of Members
01:00:00 PM	Red and Rock	Rock	3
01:00:00 PM	SQL Attack	Rock	5
02:00:00 PM	Deletion Anomaly	Metal	3
02:00:00 PM	Trumpet Call	Jazz	5
04:00:00 PM	Harley Road	Pop	4

Now the time column is not unique enough to be used as the primary key. But if we add the Band Name column, it does qualify, creating a concatenated key or composite key. So how does that relate to 2nd normal form? Well now is the time to apply your 2nd normal form. Run each column that is not in the concatenated key through this process. Does the Style column depend on Band Name column? Does it depend on the time column? Does it depend on the number of members column? Sounds easy but may be more complicated than you think.

So let's try this together then. First off does the Band name column depend on the time column? The short answer is not really. A concert time can exist without any particular band playing. Now then does the style depend on the band? Yes, style does depend on our band. Moving to the member's column, does it depend on the time? It is fairly obvious it does not. Does it depend on the band? Sure does. This is a partial dependency on a composite key.

So now what? Let's take out the second half of our concatenated key, Band Name, and put it in its very own table creating a Band Info table. But wait! Act now and I will throw in a few extra columns too. Just kidding. Each column that depends (partially or completely) on the band name follows it into the new table, Band Info. Our database now looks like example 1-2.

Example 1-2

Concert Time Table

Time	Band Name
01:00:00 PM	Red and Rock
01:00:00 PM	SQL Attack
02:00:00 PM	Deletion Anomaly
02:00:00 PM	Trumpet Call
04:00:00 PM	Harley Road

Band Info Table

Band Name	Style	No. Of Members
Red and Rock	Rock	3
SQL Attack	Rock	5
Deletion Anomaly	Metal	3
Trumpet Call	Jazz	5
Harley Road	Pop	4

After all of this, keep in mind this form only applies to concatenated primary keys, if you only have a primary key it already is in 2nd normal form.

3rd Normal Form

The first rule of creating the 3rd normal form is that the data must be in 2nd normal form. Makes sense, but the second rule is that no non-key value depends on another non-key value (transitive dependency). This one may sound complex but it relatively simple in practice. But the best way to show you is, you guessed it, an example (example 3-1).

Example 3-1

Member	Book Title	Author	Pages	Year Published
Jonny	Program with D	Randal Gently	234	2001
Bill	Not in mySQL	Willy B Dilliams	889	2012
Rebecca	Program with D	Randal Gently	234	2001
Christy	Everybody Loops	Rose Garden	453	1999
Jess	Not in mySQL	Willy B Dilliams	889	2012

When you see a lot of duplication in a table it is often an indication of the need for 3rd normal form. So we use a similar process as 2nd normal form and see if each column is dependent on the primary key. We ask is the book title dependent on the member? Yes, but the author, pages and published year is not. So we have values

dependent on the non-key value, Book Title and we continue with a similar process again as in 2nd normal form, placing the member and title in a table and the title, author, pages and published year in another.

Member Table

Member	**Book Title**
Jonny	Program with D
Bill	Not in mySQL
Rebecca	Program with D
Christy	Everybody Loops
Jess	Not in mySQL

Book Table

Book Title	**Author**	**Pages**	**Year Published**
Program with D	Randal Gently	234	2001
Everybody Loops	Rose Garden	453	1999
Not in mySQL	Willy B Dilliams	889	2012

Now we have the Member Table relating to the Book Table through the book title, this is by way of a foreign key (Book Title). Each value is now dependent on the primary key. Now we have finally achieved the process of 3rd normal form. Not so hard after all.

Final Thoughts on the 3 Normal Form Process

1. When normalizing a database, don't be afraid to add and subtract columns. Yes you can do this as long as the data is represented somewhere.
2. Just because we are in 3rd Normal Form does not necessarily mean things are perfectly placed.
3. Remember to keep an eye out for upward mobility. By that I mean your design will continue to work no matter how much data it contains.
4. Always keep in mind it's supposed to be a science but ends up being more of an art form.
5. Finally, consider that 3rd Normal Form is more of a flexible guideline, not the final word.

Couple of SQL Jokes to Finish Off

1. A SQL query walks into a bar and sees two tables. He walks ups and says, “Can I join you?” The waitress then walks over and says, “Nice View!”
2. A SQL query walks into a bar on Valentine’s Day and sees two tables, she says, “Insert all this, you Cartesian pigs.”
3. Why don't you ask SQL people to help you move? Because they sometimes drop tables.