Normalization (A Different Approach)

Hi! I hope by now normalization is clear to you. We are going to learn and understand with practical examples.

Normalization is the formalization of the design process of making a database compliant with the concept of a **Normal Form**. It addresses various ways in which we may look for repeating data values in a table. There are several levels of the Normal Form, and each level requires that the previous level be satisfied. I have used the wording (indicated in italicized text) for each normalization rule from the *Handbook of Relational Database Design* by Candace C. Fleming and Barbara von Halle.[4]

The normalization process is based on collecting an exhaustive list of all data items to be maintained in the database and starting the design with a few "superset" tables. Theoretically, it may be possible, although not very practical, to start by placing all the attributes in a single table. For best results, start with a reasonable breakdown.

First Normal Form

Reduce entities to first normal form (1NF) by removing repeating or multivalued attributes to another, child entity.

Basically, make sure that the data is represented as a (proper) table. While key to the relational principles, this is somewhat a motherhood statement. However, there are six properties of a relational table (the formal name for "table" is "relation"):

Property 1: Entries in columns are single-valued.

Property 2: Entries in columns are of the same kind.

Property 3: Each row is unique.

Property 4: Sequence of columns is insignificant.

Property 5: Sequence of rows is insignificant.

Property 6: Each column has a unique name.

The most common sins against the first normal form (1NF) are the lack of a Primary Key and the use of "repeating columns." This is where multiple values of the same type are stored in multiple columns. Take, for example, a database used by a company's order system. If the order items were implemented as multiple columns in the Orders table, the database would not be 1NF:

OrderNo	Line1Item	Line1Qty	Line1Price	Line2Item	Line2Qty	Line2Price
245	PN768	1	$35	PN656	3	$15

To make this first normal form, we would have to create a child entity of Orders (Order Items) where we would store the information about the line items on the order. Each order could then have multiple Order Items *related* to it.

OrderNo
245

OrderNo	Item	Qty	Price
245	PN768	1	$35
245	PN656	3	$15

Second Normal Form *Reduce first normal form entities to second normal form (2NF) by removing attributes that are not dependent on the whole primary key.*

The purpose here is to make sure that each column is defined in the correct table. Using the more formal names may make this a little clearer. Make sure each attribute is kept with the entity that it describes.

Consider the Order Items table that we established above. If we place Customer reference in the Order Items table (Order Number, Line Item Number, Item, Qty, Price, Customer)

and assume that we use Order Number and Line Item Number as the Primary Key, it quickly becomes obvious that the Customer reference becomes repeated in the table because it is only dependent on a portion of the Primary Key - namely the Order Number. Therefore, it is defined as an attribute of the wrong entity. In such an obvious case, it should be immediately clear that the Customer reference should be in the Orders table, not the Order Items table.

So instead of:

OrderNo	**ItemNo**	Customer	Item	Qty	Price
245	**1**	SteelCo	PN768	1	$35
245	**2**	SteelCo	PN656	3	$15
246	**1**	Acme Corp	PN371	1	$2.99
246	**2**	Acme Corp	PN015	7	$5

We get:

OrderNo	Customer
245	SteelCo
246	Acme Corp

OrderNo	**ItemNo**	Item	Qty	Price
245	**1**	PN768	1	$35
245	**2**	PN656	3	$15
246	**1**	PN371	1	$2.99
246	**2**	PN015	7	$5

Third Normal Form

Reduce second normal form entities to third normal form (3NF) by removing attributes that depend on other, nonkey attributes (other than alternative keys).

This basically means that we shouldn't store any data that can either be derived from other columns or belong in another table. Again, as an example of derived data, if our

Order Items table includes both Unit Price, Quantity, and Extended Price, the table would not be 3NF. So we would remove the Extended Price (= Qty * Unit Price), unless, of course, the value saved is a manually modified (rebate) price, but the Unit Price reflects the quoted list price for the items at the time of order.

Also, when we established that the Customer reference did not belong in the Order Items table, we said to move it to the Orders table. Now if we included customer information, such as company name, address, etc., in the Orders table, we would see that this information is dependent not so much on the Order per se, but on the Customer reference, which is a nonkey (not Primary Key) column in the Orders table. Therefore, we need to create another table (Customers) to hold information about the customer. Each Customer could then have multiple Orders related to it.

OrderNo	Customer	Address	City
245	SteelCo	Works Blvd	Vinings
246	Acme Corp	North Drive	South Bend
247	SteelCo	Works Blvd	Vinings

OrderNo	Customer
245	SteelCo
246	Acme Corp
247	SteelCo

Customer	Address	City
SteelCo	Works Blvd	Vinings
Acme Corp	North Drive	South Bend

Why Stop Here?

Many database designers stop at 3NF, and those first three levels of normalization do provide the most bang for the buck. Indeed, these were the original normal forms described in E. F. Codd's first papers. However, there are currently four additional levels of normalization, so read on. Be aware of what you don't do, even if you stop with 3NF. In some cases, you may even need to de-normalize some for performance reasons.

Boyce/Codd Normal Form

Reduce third normal form entities to Boyce/Codd normal form (BCNF) by ensuring that they are in third normal form for any feasible choice of candidate key as primary key.

In short, Boyce/Codd normal form (BCNF) addresses dependencies between columns that are part of a Candidate Key.

Some of the normalizations performed above may depend on our choice of the Primary Key. BCNF addresses those cases where applying the normalization rules to a Candidate Key other than the one chosen as the Primary Key would give a different result. In actuality, if we substitute any Candidate Key for Primary Key in 2NF and 3NF, 3NF would be equivalent with BCNF.

In a way, the BCNF is only necessary because the formal definitions center around the Primary Key rather than an entity item abstraction. If we define an entity item as an object or information instance that correlates to a row, and consider the normalization rules to refer to entity items, this normal form would not be required.

In our example for 2NF above, we assumed that we used a composite Primary Key consisting of Order Number and Line Item Number, and we showed that the customer reference was only dependent on a portion of the Primary Key - the Order Number. If we had assigned a unique identifier to every Order Item independent of the Order Number, and used that as a single column Primary Key, the normalization rule itself would not have made it clear that it was necessary to move the Customer reference.

There are some less obvious situations for this normalization rule where a set of data actually contains more than one relation, which the following example should illustrate.

Consider a scenario of a large development organization, where the projects are organized in project groups, each with a team leader acting as a liaison between the overall project and a group of developers in a matrix organization. Assume we have the following situation:

- Each Project can have many Developers.
- Each Developer can have many Projects.
- For a given Project, each Developer only works for one Lead Developer.
- Each Lead Developer only works on one Project.
- A given Project can have many Lead Developers.

In this case, we could theoretically design a table in two different ways:

ProjectNo	**Developer**	Lead Developer
20020123	**John Doe**	Elmer Fudd
20020123	**Jane Doe**	Sylvester
20020123	**Jimbo**	Elmer Fudd
20020124	**John Doe**	Ms. Depesto

Case 1: Project Number and Developer as a Candidate Key can be used to determine the Lead Developer. In this case, the Lead Developer depends on both attributes of the key, and the table is 3NF if we consider that our Primary Key.

Lead Developer	Developer	ProjectNo
Elmer Fudd	John Doe	20020123
Sylvester	Jane Doe	20020123
Elmer Fudd	Jimbo	20020123
Ms. Depesto	John Doe	20020124

Case 2: Lead Developer and Developer is another Candidate Key, but in this case, the Project Number is determined by the Lead Developer alone. Thus it would not be 3NF if we consider that our Primary Key.

In reality, these three data items contain more than one relation (Project - Lead Developer and Lead Developer - Developer). To normalize to BCNF, we would remove the second

relation and represent it in a second table. (This also illustrates why a *table* is formally named a *relation*.)

ProjectNo	Lead Developer
20020123	Elmer Fudd
20020123	Sylvester
20020123	Elmer Fudd
20020124	Ms. Depesto

Lead Developer	Developer
Elmer Fudd	John Doe
Elmer Fudd	Jimbo
Sylvester	Jane Doe
Ms. Depesto	John Doe

Fourth Normal Form

Reduce Boyce/Codd normal form entities to fourth normal form (4NF) by removing any independently multivalued components of the primary key to two new parent entities. Retain the original (now child) entity only if it contains other, nonkey attributes.

Where BCNF deals with dependents of dependents, 4NF deals with multiple, independent dependents of the Primary Key. This is a bit easier to illustrate.

Let us say we wanted to represent the following data: Manager, Manager Awards, and Direct Reports. Here, a Manager could have multiple Awards, as well as multiple Direct Reports. 4NF requires that these be split into two separate tables, one for Manager - Awards, and one for Manager - Direct Reports. We may need to maintain a Managers table for other Manager attributes.

This table:

Manager	Awards	Direct Reports
Scrooge McDuck	Stingy John	Donald Duck
Minnie Mouse	Mouse of the Month	Mickey Mouse
Minnie Mouse	Mouse of the Year	Pluto
Clara		Goofy

becomes two tables:

Manager Awards Table

Manager	Awards
Scrooge McDuck	Stingy John
Minnie Mouse	Mouse of the Month
Minnie Mouse	Mouse of the Year
Clara	

Direct Reports Table

Manager	Direct Reports
Scrooge McDuck	Donald Duck
Minnie Mouse	Mickey Mouse
Minnie Mouse	Pluto
Clara	Goofy

Fifth Normal Form

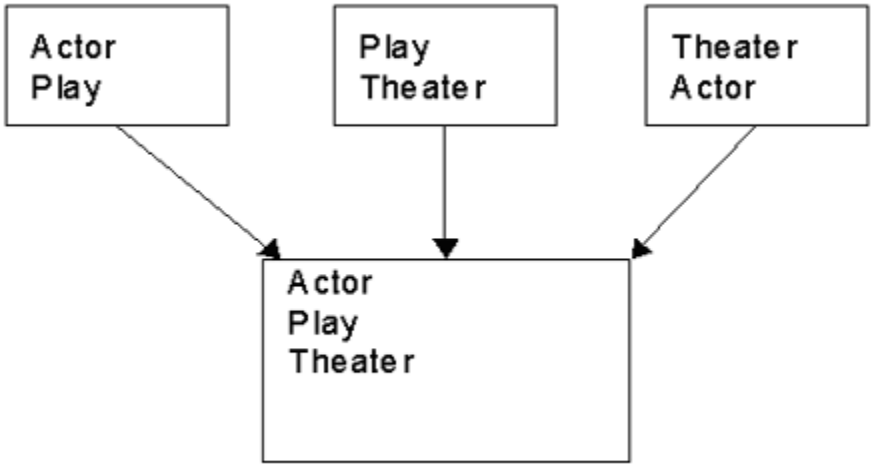
Reduce fourth normal form entities to fifth normal form (5NF) by removing pairwise cyclic dependencies (appearing within composite primary keys with three or more component attributes) to three or more parent entities.

This addresses problems that arise from representing associations between multiple entities with interdependencies. Making it 5NF consists of adding parent tables, one for each meaningful combination that has children in the original table.

A table with such information is 5NF if the information cannot be represented in multiple smaller entities alone.

An example of such a situation may be the representation of Actors, Plays, and Theaters. In order to know who plays what and where, we need the combination of these three attributes. However, they each relate to each other cyclically. So to resolve this, we would need to establish parent tables with Actor - Play, Play - Theater, and Theater - Actor. These would each contain a portion of the Primary Key in the Actor, Play, and Theater table.

Actor	**Play**	**Theater**
Billy Bob	**Catcher in the Rye**	**West 42nd**
Ann	**Catcher in the Rye**	**West 42nd**
John	**Catch-22**	**Broadway**
Lily	**Hamlet**	**Broadway**
Lisa	**Cats**	**West 42nd**
Andy	**Cats**	**Darlington**



Domain Key Normal Form

(Not defined in "Handbook of Relational Database Design."[5]).

The simplest description I have found is at Search Database.com at http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci212669,00.html:

"A key uniquely identifies each row in a table. A domain is the set of permissible values for an attribute. By enforcing key and domain restrictions, the database is assured of being freed from modification anomalies."

This appears to differ from the other normal forms in that it does not seek to introduce additional tables, but rather ensures that columns are restricted to valid values.

According to http://www.cba.nau.edu/morgan-j/class/subtop2_3/tsld023.htm, "...there is no known process for ensuring that tables are in Domain Key Normal Form."

Conclusion

While we may not always observe all the rules or normalize our databases to the fifth and domain key normal form, it is important to have a basic understanding of the theoretical principles of database design. It will help us not only design normalized databases, but to build more powerful and flexible applications. Also, it will help us ensure that our data

remains usable. Now that we have laid the theoretical foundation and defined the formal database design methods for normalization, it may be time to take a break. I need one anyway. :->

When we resume with Part 2 of this article, I will show how we can design a fairly well normalized database using nothing but some common sense, a few simple rules, and a piece of string, so get ready! Part 2 of this article will address a different approach to designing the database, normalization through synthesis, and will describe the SQL language.

The relational model has three major aspects:

Structures	Structures are well-defined objects (such as tables, views, indexes, and so on) that store or access the data of a database. Structures and the data contained within them can be manipulated by operations.
Operations	Operations are clearly defined actions that allow users to manipulate the data and structures of a database. The operations on a database must adhere to a predefined set of integrity rules.
Integrity Rules	Integrity rules are the laws that govern which operations are allowed on the data and structures of a database. Integrity rules protect the data and the structures of a database.

Relational database management systems offer benefits such as:

- independence of physical data storage and logical database structure
- variable and easy access to all data
- complete flexibility in database design
- reduced data storage and redundancy

Review questions

1. Explain first, second and third normal forms
2. Explain BCNF
3. Explain fourth and fifth normal forms
4. Define Domain key normal form

Sources

1. C. J. Date, "There's Only One Relational Model!"(see http://www.pgro.uk7.net/cjd6a.htm).
2. Dr. E. F. Codd's 12 rules for defining a fully relational database (see http://www.cis.ohio-state.edu/~sgomori/570/coddsrules.html).
3. C.J.Date *Handbook of Relational Database Design* by Candace C. Fleming and Barbara von Halle (Addison Wesley, 1989).5. *Ibid.*

References

1. Characteristics of a Relational Database by David R. Frick & Co., CPA.
2. Dr. Morgan at Northern Arizona University - College of Business Administration
3. SearchDatabase.com