Learn about the First Normal Form and Database Design

This is the second in a series of posts teaching normalization.

The first post introduced database normalization, its importance, and the types of issues it solves.

In this article we'll explore the first normal form. For the examples, we'll use the Sales Staff Information shown below as a starting point. As we pointed out in the last post's modification anomalies section, there are several issues to keeping the information in this form. By normalizing the data you see we'll eliminate duplicate data as well as modification anomalies.

SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

1NF – First Normal Form Definition

The first steps to making a proper SQL table is to ensure the information is in first normal form. Once a table is in first normal form it is easier to search, filter, and sort the information. The rules to satisfy 1st normal form are:

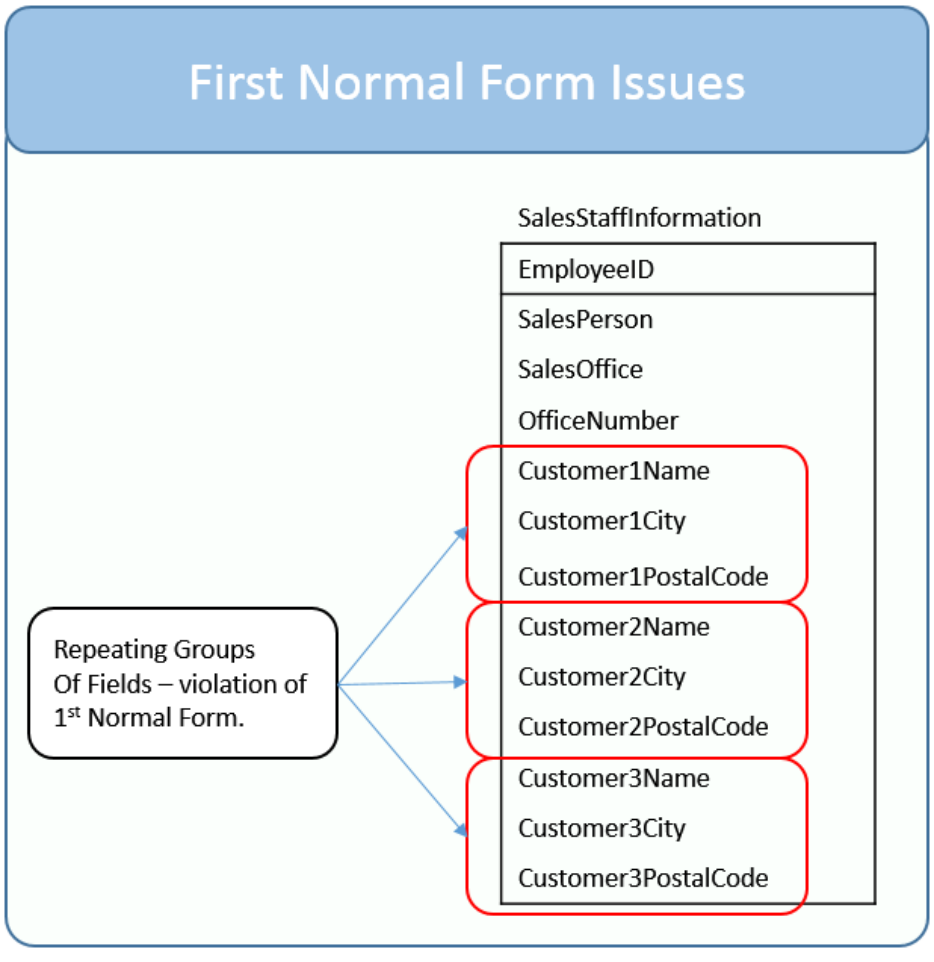
- That the data is in a database table. The table stores information in rows and columns where one or more columns, called the primary key, uniquely identify each row.
- Each column contains atomic values, and there are not repeating groups of columns.

Tables in first normal form cannot contain sub columns. That is, if you are listing several cities, you cannot list them in one column and separate them with a semi-colon.
Get your **Free Coupon** Today

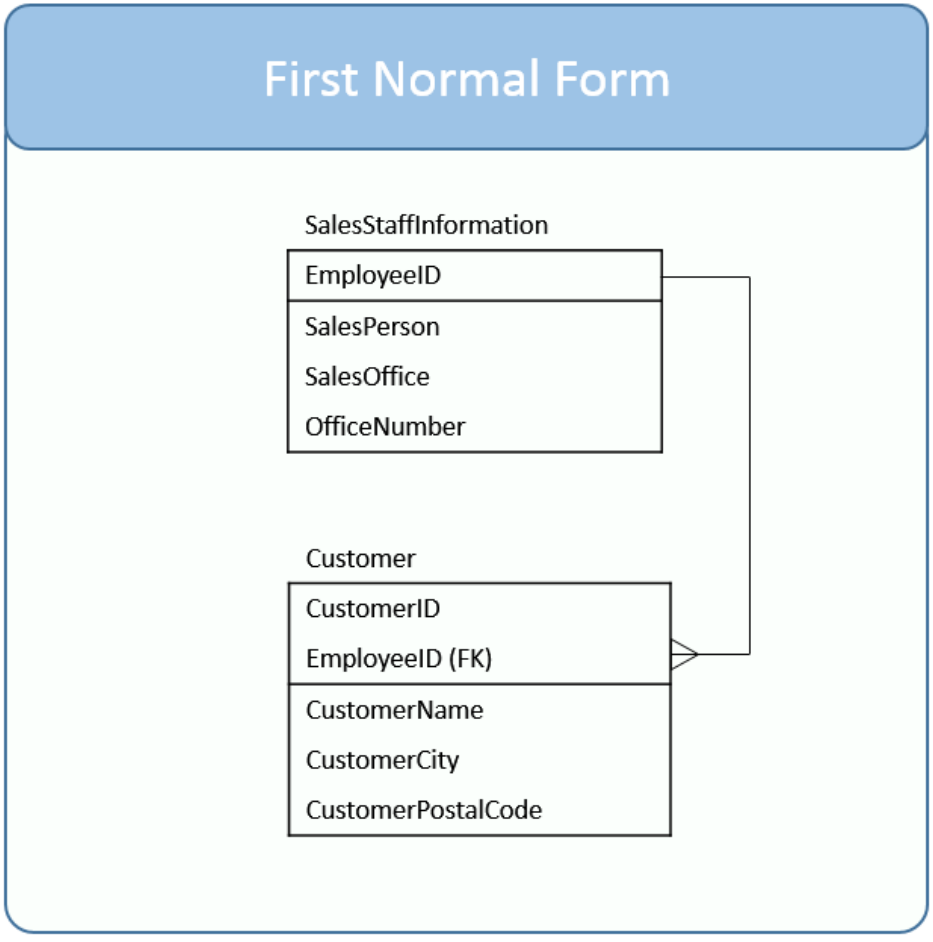
It's my best deal possible to enroll in

Database Normalization Simplified.

When a value is atomic, the values cannot be further subdivided. For example, the value “Chicago” is atomic; whereas “Chicago; Los Angeles; New York” is not. Related to this requirement is the concept that a table should not contain repeating groups of columns such as Customer1Name, Customer2Name, and Customer3Name.



Our example table is transformed to first normal form by placing the repeating customer related columns into their own table. This is shown below:



The repeating groups of columns now become separate rows in the Customer table linked by the EmployeeID foreign key. As mentioned in the lesson on Data Modeling, a foreign key is a value which matches back to another table's primary key. In this case, the customer table contains the corresponding EmployeeID for the SalesStaffInformation row. Here is our data in first normal form.

SalesStaffInformation			
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**
1003	Mary Smith	Chicago	312-555-1212
1004	John Hunt	New York	212-555-1212
1005	Martin Hap	Chicago	312-555-1212

Note: Primary Key: EmployeeID

Customer				
CustomerID	**EmployeeID**	**CustomerName**	**CustomerCity**	**PostalCode**
C1000	1003	Ford	Dearborn	48123
C1010	1003	GM	Detroit	48213
C1020	1004	Dell	Austin	78720
C1030	1004	HP	Palo Alto	94303
C1040	1004	Apple	Cupertino	95014
C1050	1005	Boeing	Chicago	60601

This design is superior to our original table in several ways:

1. The original design limited each SalesStaffInformation entry to three customers. In the new design, the number of customers associated to each design is practically unlimited.
2. It was nearly impossible to Sort the original data by Customer. You could, if you used the UNION statement, but it would be cumbersome. Now, it is simple to sort customers.
3. The same holds true for filtering on the customer table. It is much easier to filter on one customer name related column than three.
4. The insert and deletion anomalies for Customer have been eliminated. You can delete all the customer for a SalesPerson without having to delete the entire SalesStaffInformaiton row.

Modification anomalies remain in both tables, but these are fixed once we reorganize them as 2nd normal form.

It's my best deal possible to enroll in

Database Normalization Simplified.

More tutorials are to follow! Remember! I want to remind you all that if you have other questions you want answered, then post a comment or tweet me. I’m here to help you. What other topics would you like to know more about?