# Normalization

When Codd invented relational databases in 1970 he described, not only relational algebra, but a process by which complicated real data could be constrained into the restricted structure of relations. He called this process normalization. Normalization consists of putting all the data you have got into tables where each element is atomic. This is called **first normal form**. Putting data into first normal form has the disadvantage that it introduces a great deal of duplication into the data and so he invented second and third normal form to remove the duplication he introduced with first normal form. Subsequently Codd and others added Boyce/Codd, forth and fifth normal forms but since each normal form simply adds an extra condition to its predecessors we do not need to go into all of them in detail. On the other hand we cannot simply rely on Entity / Attribute analysis either because it is designed for an object oriented environment not a relational one, it is not rigorous enough to use reliably on very large designs and because, in some circumstances, normalization will produce tables with less duplication than Entity / Attribute analysis.

Putting data into tables produces duplication because data tends not to be naturally atomic. For example we could have three different attributes, A, B and C. Now suppose that every entity had one attribute value for attributes A and B but might have a list of values for attribute C. This could be transformed into a table as follows:

| A | B | C |
|---|---|---|
| a1 | b1 | c1,c2,c3 |
| a2 | b2 | c4 |

becomes

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a1 | b1 | c2 |
| a1 | b1 | c3 |
| a2 | b2 | c4 |

This is the conventional technique for transforming unnormalized data into first normal form but the same problem occurs when turning an object oriented design into a set of tables.

Most of the normal forms are based on the concept of functional dependence. A column X is said to be **Functionally Dependent** on a column Y if, for any given value of Y, there is only one possible value X at any particular time. E.g. a column containing someone's age could be functionally dependent on a column containing his name because people only have one age. The age will change with time but nobody has two ages at once. On the other hand the name column is not functionally dependent on the age column because several people can have the same age.

Given a table of data with no idea about its meaning it is possible to say which columns cannot be functionally dependent on other columns but not to say which columns are functionally dependent on other columns unless the table contains all the rows it could ever contain. Functional dependence is related to the meaning of the data.

A column can be functionally dependent on a group of columns and several columns can be functionally dependent on the same column or group of columns. A column, or a group of columns, X is said to be **Fully Functionally Dependent** on a group of columns Y if X is functionally dependent on Y but not on any subset of Y.

In normalization data is split into tables in such a way that all non-key columns must be fully functionally dependent on the primary key. Furthermore all functional dependencies within the table must be full functional dependencies on a candidate key. One possible starting point is to imagine all related data as one table and then identify functional dependencies within it. If any of them break the functional

dependency rules above then the table must be split with the functional dependency represented in one table but the column or columns depended on appearing in both tables. The same rules are applied to the two new tables until there is a set of tables all of which conform to these rules. Alternatively it is possible to start from tables designed as result of Entity / Attribute analysis and consider each column in the light of the functional dependency rules but be careful to note the attributes for which entities, in unnormalized form, may sometimes have lists of attribute value because these are the ones that might cause duplication.

This process puts the tables into Boyce/Codd normal form. There are two more normal forms, forth and fifth. The fifth normal form rarely causes problems and anyway those problems are very difficult to identify so we will ignore it. However forth normal form anomalies occur relatively frequently which means it is important to be aware of them. For that we need a new definition.

In a table with at least three columns A, B and C if, for a given value in A, there are rows associating it with a set of values in B irrespective of the value in C and rows associating it with a set of values in C irrespective of the value in B then B and C are said to have a **Multivalued Dependency** on A.

A, B and C can be groups of columns not just single columns. Multivalued dependencies can be identified easily in an unnormalized table because there are two or more columns both dependent on the same key column and both containing lists of values which have to be turned into atomic values to put the table into first normal form.

For example an unnormalized table describing lecture courses may have a column containing a course name which is the key, a column containing a list of all the students attending the course and a column containing a list of the lecturers and demonstrators concerned in delivering the course. A version of this table in first normal form will contain three columns; course name, student name and teacher name

COURSES (STUDENT, COURSE, TEACHER)

But it will display a great deal of duplication even though there are no functional dependencies. This duplication results from a multivalued dependency; student name and teacher name have a multivalued dependency on course name.

Multivalued dependencies can be eliminated by splitting the table into two tables, one for each of the multivalued columns but both containing the common key.

For example above we have

ATTENDS (COURSE, STUDENT)

and

TEACHING (COURSE, TEACHER)

The point of normalization is to change complicated data into tables in which all data is atomic and which have no duplication in them. Duplication is a bad thing because it wastes space and produces the risk of inconsistent data appearing in the database; if a piece of information occurs twice it could become inconsistent if one version were changed and not the other.

Like Entity /Attribute analysis, there is usually no definitive right answer to a normalization problem although there are wrong answers. Functional dependencies and multivalued dependencies are associated with the meaning of the data. If the meaning of the data changes the data tables may no longer be normalized so it is important to document any dependencies you detect and build in to the design of your database. Finally remember that normalization is not compulsory; if you follow the rules slavishly you can get very silly results.