**Database Normalization**

(Source: http://www.1keydata.com/database-normalization/)

In relational database design, we not only want to create a structure that stores all of the data, but we also want to do it in a way that minimize potential errors when we work with the data. The default language for accessing data from a relational database is [SQL](http://www.1keydata.com/sql/sql.html). In particular, SQL can be used to manipulate data in the following ways: [insert](http://www.1keydata.com/sql/sqlinsert.html) new data, [delete](http://www.1keydata.com/sql/sqldelete.html) unwanted data, and [update](http://www.1keydata.com/sql/sqlupdate.html) existing data. Similarly, in an un-normalized design, there are 3 problems that can occur when we work with the data:

**INSERT ANOMALY**: This refers to the situation when it is impossible to insert certain types of data into the database.

**DELETE ANOMALY**: The deletion of data leads to unintended loss of additional data, data that we had wished to preserve.

**UPDATE ANOMALY**: This refers to the situation where updating the value of a column leads to database inconsistencies (i.e., different rows on the table have different values).

To address the 3 problems above, we go through the process of normalization. When we go through the normalization process, we increase the number of tables in the database, while decreasing the amount of data stored in each table. There are several different levels of database normalization:

* [1st Normal Form (1NF)](http://www.1keydata.com/database-normalization/first-normal-form-1nf.php)
* [2nd Normal Form (2NF)](http://www.1keydata.com/database-normalization/second-normal-form-2nf.php)
* [3rd Normal Form (3NF)](http://www.1keydata.com/database-normalization/third-normal-form-3nf.php)
* Bryce-Codd Normal Form (BCNF)
* 4th Normal Form (4NF)
* 5th Normal Form (5NF)

The opposite of normalization is denormalization, where we want to combine multiple tables together into a larger table. Denormalization is most frequently associated with designing the fact table in a [data warehouse](http://www.1keydata.com/datawarehousing/datawarehouse.html).

## 1st Normal Form Definition

A database is in first normal form if it satisfies the following conditions:

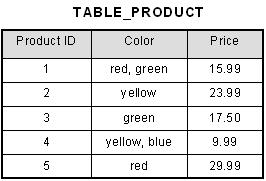
* Contains only atomic values
* There are no repeating groups

An atomic value is a value that cannot be divided. For example, in the table shown below, the values in the [Color] column in the first row can be divided into "red" and "green", hence [TABLE\_PRODUCT] is not in 1NF.

A repeating group means that a table contains two or more columns that are closely related. For example, a table that records data on a book and its author(s) with the following columns: [Book ID], [Author 1], [Author 2], [Author 3] is not in 1NF because [Author 1], [Author 2], and [Author 3] are all repeating the same attribute.

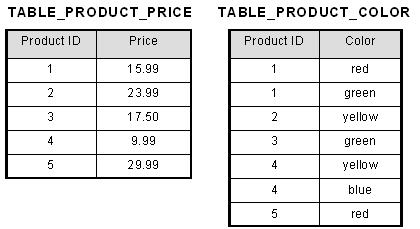
## 1st Normal Form Example

How do we bring an unnormalized table into first normal form? Consider the following example:



This table is not in first normal form because the [Color] column can contain multiple values. For example, the first row includes values "red" and "green."

To bring this table to first normal form, we split the table into two tables and now we have the resulting tables:



Now first normal form is satisfied, as the columns on each table all hold just one value.

## 2nd Normal Form Definition

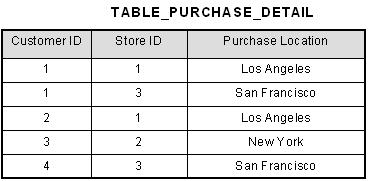
A database is in second normal form if it satisfies the following conditions:

* It is in first normal form
* All non-key attributes are fully functional dependent on the primary key

In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

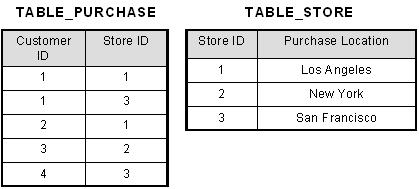
## 2nd Normal Form Example

Consider the following example:



This table has a composite primary key [Customer ID, Store ID]. The non-key attribute is [Purchase Location]. In this case, [Purchase Location] only depends on [Store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we break the table into two tables, and now we have the following:



What we have done is to remove the partial functional dependency that we initially had. Now, in the table [TABLE\_STORE], the column [Purchase Location] is fully dependent on the primary key of that table, which is [Store ID].

## 3rd Normal Form Definition

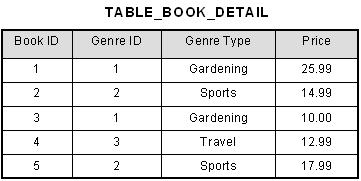
A database is in third normal form if it satisfies the following conditions:

* It is in second normal form
* There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the table: A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

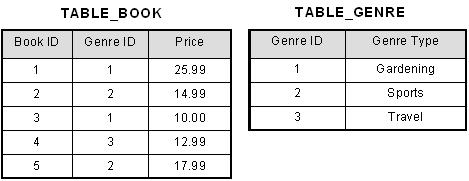
## 3rd Normal Form Example

Consider the following example:



In the table able, [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Type]. Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two as follows:



Now all non-key attributes are fully functional dependent only on the primary key. In [TABLE\_BOOK], both [Genre ID] and [Price] are only dependent on [Book ID]. In [TABLE\_GENRE], [Genre Type] is only dependent on [Genre ID].