Functional dependency: Attribute B has a functional dependency on attribute A (i.e., **A** → **B** if, for each value of attribute A, there is exactly one value of attribute B). If value of A is repeating in tuples then value of B will also repeat. In our example, Employee Address has a functional dependency on Employee ID, because a particular Employee ID value corresponds to one and only one Employee Address value. (Note that the reverse need not be true: several employees could live at the same address and therefore one Employee Address value could correspond to more than one Employee ID. Employee ID is therefore **not** functionally dependent on Employee Address.)

What is Normalization?

Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

The Normal Forms

The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and 3NF along with the occasional 4NF. Fifth normal form is very rarely seen and won't be discussed in this article.

Before we begin our discussion of the normal forms, it's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible ramifications they could have on your system and account for possible inconsistencies. That said, let's explore the normal forms.

First Normal Form (1NF)

First normal form (1NF) sets the very basic rules for an organized database:

- Eliminate duplicative columns from the same table.
- Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).

Example 1: Domains and values

Suppose a novice designer wishes to record the names and telephone numbers of customers. He defines a customer table which looks like this:

Customer

Customer ID	First Name	Surname	Telephone Number

123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659
789	Maria	Fernandez	555-808-9633

The designer then becomes aware of a requirement to record **multiple** telephone numbers for some customers. He reasons that the simplest way of doing this is to allow the "Telephone Number" field in any given record to contain more than one value:

Customer

Customer ID	First Name	Surname	Telephone Number
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659 555-776-4100
789	Maria	Fernandez	555-808-9633

Assuming, however, that the Telephone Number column is defined on some Telephone Number-like domain (e.g. the domain of strings 12 characters in length), the representation above is not in 1NF. 1NF (and, for that matter, the RDBMS) prohibits a field from containing more than one value from its column's domain.

Example 2: Repeating groups across columns

The designer might attempt to get around this restriction by defining multiple Telephone Number columns:

Customer

Customer ID	First Name	Surname	Tel. No. 1	Tel. No. 2	Tel. No. 3

123	Robert	Ingram	555-861-2025		
456	Jane	Wright	555-403-1659	555-776-4100	
789	Maria	Fernandez	555-808-9633		

This representation, however, makes use of nullable columns, and therefore does not conform to Date's definition of 1NF. Even if the view is taken that nullable columns are allowed, the design is not in keeping with the spirit of 1NF. Tel. No. 1, Tel. No. 2., and Tel. No. 3. share exactly the same domain and exactly the same meaning; the splitting of Telephone Number into three headings is artificial and causes logical problems. These problems include:

- Difficulty in querying the table. Answering such questions as "Which customers have telephone number *X*?" and "Which pairs of customers share a telephone number?" is awkward.
- Inability to enforce uniqueness of Customer-to-Telephone Number links through the RDBMS. Customer 789 might mistakenly be given a Tel. No. 2 value that is exactly the same as her Tel. No. 1 value.
- Restriction of the number of telephone numbers per customer to three. If a customer with four telephone numbers comes along, we are constrained to record only three and leave the fourth unrecorded. This means that the database design is imposing constraints on the business process, rather than (as should ideally be the case) vice-versa.

Example 3: Repeating groups within columns

The designer might, alternatively, retain the single Telephone Number column but alter its domain, making it a string of sufficient length to accommodate multiple telephone numbers:

Customer			
Customer ID	**First Name**	**Surname**	**Telephone Number**
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659, 555-776-4100
789	Maria	Fernandez	555-808-9633

This design is not consistent with 1NF, and presents several design issues. The Telephone Number heading becomes semantically woolly, as it can now represent either a telephone number, a list of telephone numbers, or indeed anything at all. A query such as "Which pairs of customers share a telephone number?" is more difficult to formulate, given the necessity to cater for lists of telephone numbers as well as individual telephone numbers. Meaningful constraints on telephone numbers are also very difficult to define in the RDBMS with this design.

A design that complies with 1NF

A design that is unambiguously in 1NF makes use of two tables: a Customer Name table and a Customer Telephone Number table.

Customer Name

Customer ID	First Name	Surname
123	Robert	Ingram
456	Jane	Wright
789	Maria	Fernandez

Customer Telephone Number

Customer ID	Telephone Number
123	555-861-2025
456	555-403-1659
456	555-776-4100
789	555-808-9633

Repeating groups of telephone numbers do not occur in this design. Instead, each Customer-to-Telephone Number link appears on its own record.

Second Normal Form (2NF)

Second normal form (2NF) further addresses the concept of removing duplicative data:

- Meet all the requirements of the first normal form.
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Create relationships between these new tables and their predecessors through the use of foreign keys.

Example

Consider a table describing employees' skills:

Employees' Skills

Employee	**Skill**	**Current Work Location**
Jones	Typing	114 Main Street
Jones	Shorthand	114 Main Street
Jones	Whittling	114 Main Street
Roberts	Light Cleaning	73 Industrial Way
Ellis	Alchemy	73 Industrial Way
Ellis	Juggling	73 Industrial Way
Harrison	Light Cleaning	73 Industrial Way

The table's only candidate key is {Employee, Skill}.

The remaining attribute, Current Work Location, is dependent on only part of the candidate key, namely Employee. Therefore the table is not in 2NF. Note the redundancy in the way Current Work Locations are represented: we are told three times that Jones works at 114 Main Street, and twice that Ellis works at 73 Industrial Way. This redundancy makes the table vulnerable to update anomalies: it is, for example, possible to update Jones' work location on his "Typing" and "Shorthand" records and not update his "Whittling" record. The resulting data would imply contradictory answers to the question "What is Jones' current work location?"

A 2NF alternative to this design would represent the same information in two tables:

Employees

Employee	**Current Work Location**
Jones	114 Main Street

Employees' Skills

Employee	**Skill**
Jones	Typing

Roberts	73 Industrial Way
Ellis	73 Industrial Way
Harrison	73 Industrial Way

Jones	Shorthand
Jones	Whittling
Roberts	Light Cleaning
Ellis	Alchemy
Ellis	Juggling
Harrison	Light Cleaning

Update anomalies cannot occur in these tables, which are both in 2NF.

Third Normal Form (3NF)

Third normal form (3NF) goes one large step further:

- Meet all the requirements of the second normal form.
- Remove columns that are not dependent upon the primary key.

Example

An example of a 2NF table that fails to meet the requirements of 3NF is:

Tournament Winners

Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968

Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

The only candidate key is {Tournament, Year}.

The breach of 3NF occurs because the non-prime attribute Winner Date of Birth is transitively dependent on {Tournament, Year} via the non-prime attribute Winner. The fact that Winner Date of Birth is functionally dependent on Winner makes the table vulnerable to logical inconsistencies, as there is nothing to stop the same person from being shown with different dates of birth on different records.

In order to express the same facts without violating 3NF, it is necessary to split the table into two:

Tournament Winners

Tournament	**Year**	**Winner**
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Player Dates of Birth

Player	**Date of Birth**
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Update anomalies cannot occur in these tables, which are both in 3NF.

- courses - tutors - reference books
- assumptions
    - for a course - any number of tutors and any numbers of books
    - a book can be used for any number of courses
    - a tutor can teach any number of courses
    - no link between tutors and books (!!!)