# First normal form

From Wikipedia, the free encyclopedia

**First normal form** (**1NF or Minimal Form**) is a normal form used in database normalization. A relational database table that adheres to 1NF is one that meets a certain minimum set of criteria. These criteria are basically concerned with ensuring that the table is a faithful representation of a **relation**[1] and that it is free of **repeating groups**[2].

The concept of a "repeating group" is, however, understood in different ways by different theorists. As a consequence, there is not universal agreement as to which features would disqualify a table from being in 1NF. Most notably, 1NF as defined by some authors (for example, Ramez Elmasri and Shamkant B. Navathe[3], following the precedent established by E.F. Codd) excludes **relation-valued attributes** (tables within tables); whereas 1NF as defined by other authors (for example, Chris Date) permits them.

## Contents



## 1NF tables as representations of relations

According to Date's definition of 1NF, a table is in 1NF if and only if it is "isomorphic to some relation", which means, specifically, that it satisfies the following five conditions:

> 1. There's no top-to-bottom ordering to the rows.
> 2. There's no left-to-right ordering to the columns.
> 3. There are no duplicate rows.
> 4. Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
> 5. All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].
>
> —Chris Date, "What First Normal Form Really Means", pp. 127-8[4]

Violation of any of these conditions would mean that the table is not strictly relational, and therefore that it is not in 1NF.

Examples of tables (or views) that would not meet this definition of 1NF are:

- A table that lacks a unique key. Such a table would be able to accommodate duplicate rows, in violation of condition 3.
- A view whose definition mandates that results be returned in a particular order, so that the row-ordering is an intrinsic and meaningful aspect of the view.[5] This violates condition 1. The tuples in true relations are not ordered with respect to each other.
- A table with at least one nullable attribute. A nullable attribute would be in violation of condition 4, which requires every field to contain exactly one value from its column's domain. It should be noted, however, that this aspect of condition 4 is controversial. It marks an important departure from Codd's original vision of the relational model, which made explicit provision for nulls.[6]

# Repeating groups

Date's fourth condition, which expresses "what most people think of as *the* defining feature of 1NF"[7], is concerned with repeating groups. The following example illustrates how a database design might incorporate repeating groups, in violation of 1NF.

### 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 | Rachel | Ingram | 555-861-2025 |
| 456 | James | 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 | Rachel | Ingram | 555-861-2025 |
| 456 | James | Wright | 555-403-1659<br>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 | Osvaldo | Ingram | 555-861-2025 | | |
| 456 | James | 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 | Rachel | Ingram | 555-861-2025 |
| 456 | James | Wright | 555-403-1659, 555-776-4100 |
| 789 | Maria | Fernandez | 555-808-9633 |

This is arguably the worst design of all, and again not in keeping with the spirit of 1NF. 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 virtually impossible 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 impossible 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 table and a Customer Telephone Number table.

**Customer**

| Customer ID | First Name | Surname |
|---|---|---|
| 123 | Rachel | Ingram |
| 456 | James | 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.

# Atomicity

Some definitions of 1NF, most notably that of E.F. Codd, make reference to the concept of **atomicity**. Codd states that the "values in the domains on which each relation is defined are required to be atomic with respect to the DBMS."[8] Codd defines an atomic value as one that "cannot be decomposed into smaller pieces by the DBMS (excluding certain special functions)."[9]

Hugh Darwen and Chris Date have suggested that Codd's concept of an "atomic value" is ambiguous, and that this ambiguity has led to widespread confusion about how 1NF should be understood.[10][11] In particular, the notion of a "value that cannot be decomposed" is problematic, as it would seem to imply that few, if any, data types are atomic:

- A character string would seem not be atomic, as the RDBMS typically provides operators to decompose it into substrings.
- A date would seem not to be atomic, as the RDBMS typically provides operators to decompose it into day, month, and year components.
- A fixed-point number would seem not to be atomic, as the RDBMS typically provides operators to decompose it into integer and fractional components.

Date suggests that "the notion of atomicity *has no absolute meaning*"[12]: a value may be considered atomic for some purposes, but may be considered an assemblage of more basic elements for other purposes. If this position is accepted, 1NF cannot be defined with reference to atomicity. Columns of any conceivable data type (from string types and numeric types to array types and table types) are then acceptable in a 1NF table—although perhaps not always desirable. Date argues that relation-valued attributes, by means of which a field within a table can contain a table, are useful in rare cases.[13]

# Notes and References

1. ^ "[T]he overriding requirement, to the effect that the table must directly and faithfully represent a relation, follows from the fact that 1NF was originally defined as a property of relations, not tables." Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) in *Date on Database: Writings 2000-2006* (Springer-Verlag, 2006), p. 128.

2. ^ "First normal form excludes variable repeating fields and groups." Kent, William. "A Simple Guide to Five Normal Forms in Relational Database Theory" (http://www.bkent.net/Doc/simple5.htm), *Communications of the ACM* **26** (2), Feb. 1983, pp. 120-125.
3. ^ Elmasri, Ramez and Navathe, Shamkant B. *Fundamentals of Database Systems, Fourth Edition* (Addison-Wesley, 2003), p. 315.
4. ^ Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) pp. 127-128.
5. ^ Such views cannot be created using SQL that conforms to the SQL:2003 standard.
6. ^ The third of Codd's 12 rules states that "Null values ... [must be] supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type." Codd, E.F. "Is Your DBMS Really Relational?" *Computerworld*, October 14, 1985.
7. ^ Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) p. 128.
8. ^ Codd, E.F. *The Relational Model for Database Management Version 2* (Addison-Wesley, 1990).
9. ^ Codd, E.F. *The Relational Model for Database Management Version 2* (Addison-Wesley, 1990), p. 6.
10. ^ Darwen, Hugh. "Relation-Valued Attributes; or, Will the Real First Normal Form Please Stand Up?", in C. J. Date and Hugh Darwen, *Relational Database Writings 1989-1991* (Addison-Wesley, 1992).
11. ^ "[F]or many years," writes Date, "I was as confused as anyone else. What's worse, I did my best (worst?) to spread that confusion through my writings, seminars, and other presentations." Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) in *Date on Database: Writings 2000-2006* (Springer-Verlag, 2006), p. 108
12. ^ Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) p. 112.
13. ^ Date, C.J. "What First Normal Form Really Means" (http://www.dbdebunk.com/page/page/629796.htm) pp. 121-126.

# See also

- Attribute-value system

# Further reading


- Litt's Tips: Normalization (http://www.troubleshooters.com/littstip/ltnorm.html)
- Rules Of Data Normalization (http://www.datamodel.org/NormalizationRules.html)
- Date, C. J., & Lorentzos, N., & Darwen, H. (2002). *Temporal Data & the Relational Model (http://www.elsevier.com/wps/product/cws_home/680662)* (1st ed.). Morgan Kaufmann. ISBN 1-55860-855-9.
- Date, C. J. (1999), *An Introduction to Database Systems (http://www.aw-bc.com/catalog/academic/product/0,1144,0321197844,00.html)* (8th ed.). Addison-Wesley Longman. ISBN 0-321-19784-4.
- Kent, W. (1983) *A Simple Guide to Five Normal Forms in Relational Database Theory (http://www.bkent.net/Doc/simple5.htm)*, Communications of the ACM, vol. 26, pp. 120-125
- Date, C.J., & Darwen, H., & Pascal, F. *Database Debunkings (http://www.dbdebunk.com/)*


# External links

- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- An Introduction to Database Normalization (http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html) by Mike Hillyer.
- Normalization (http://www.utexas.edu/its/windows/database/datamodeling/rm/rm7.html) by ITS, University of Texas.
- Rules of Data Normalization (http://www.datamodel.org/NormalizationRules.html) by Data Model.org

- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- Free PDF poster available (http://www.marcrettig.com/poster/) by Marc Rettig
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft

**Topics in Database normalization**

**First normal form** | Second normal form | Third normal form
Boyce-Codd normal form | Fourth normal form | Fifth normal form | Domain/key normal form | Sixth normal form
Denormalization

Retrieved from "http://en.wikipedia.org/wiki/First_normal_form"

Category: Database normalization