

Introduction to normal forms

First Normal Form

Let's consider the following Student table:

Student ID	First Name	Surname	Phone number	Course
1234	John	Smith	012345678	CAIS
4567	James	Wright	014634967	CASE
8901	Fiona	Murphy	013679845	CASE

What if we want to add several contact numbers for each student?

Solution 1: allowing the "phone number" field to contain more than one value

Student ID	First Name	Surname	Phone number	Course
1234	John	Smith	012345678 0860123456	CAIS
4567	James	Wright	014634967	CASE
8901	Fiona	Murphy	013679845	CASE

→ Not possible in RDBMS.

Solution 2: creating additional “phone number” fields

Student ID	First Name	Surname	Phone1	Phone2	Course
1234	John	Smith	012345678	0860123456	CAIS
4567	James	Wright	014634967		CASE
8901	Fiona	Murphy	013679845		CASE

➔ Possible in RDBMS, but creates a number of difficulties:

- Queries such as finding the student with a given phone number, or checking whether two students have the same number become very complex.
- The link between a student and a phone number is no longer unique, (i.e. there is no way to enforce that the second phone number for John Smith is different than the first one).
- What happens if we need to record three phone numbers for a given student? Or four, five?

Solution 3: creating a longer “phone number” field, to fit two numbers in it

Student ID	First Name	Surname	Phone Number	Course
1234	John	Smith	012345678, 0860123456	CAIS
4567	James	Wright	014634967	CASE
8901	Fiona	Murphy	013679845	CASE

➔ Possible in RDBMS, but creates a number of difficulties:

- What does the field mean? It can be a number, a list of numbers, anything.
- How to specify constraints on the phone numbers?
- How to efficiently query this field, e.g. to find a specific number?

Solution 4: using two tables, Student and Contact Details

Student ID	First Name	Surname	Course	Student ID	Phone Number
1234	John	Smith	CAIS	1234	012345678
4567	James	Wright	CASE	1234	0860123456
8901	Fiona	Murphy	CASE	4567	014634967
				8901	013679845

→ Works fine, and only requires (Student ID, Phone Number) to be the primary key in the second table.

This solution only contains tables which are in "1NF".

Chris Date's definition: a table is in 1NF if, and only if,

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].

Edgar F. Codd's definition: "values in the domains on which each relation is defined are required to be atomic with respect to the DBMS."

Second Normal Form

Let's consider another Student table:

Student	Module	Home Address
Jones	CA218	Jones' home
Smith	CA218	Another place
Wright	CA218	Elsewhere
Murphy	CA220	Right here
Jones	CA220	Jones' home
Kearney	CA220	Nowhere

Neither {Student} nor {Module} is a candidate key for the table:

- a given Student needs to appear more than once (he is studying several Modules)
- a given Module needs to appear more than once (it is followed by several Students)

Only the composite key {Student, Module} qualifies as a candidate key for the table.

The remaining attribute, Home Address, is dependent on only part of the candidate key, namely Student.

There is redundancy, as the address of a given student appears several times as soon as he is registered to several modules.

This makes the table vulnerable: it is possible to change Jones' home address for CA218 and not for CA220, which would imply contradictory answers to a query on his address.

Therefore the table is not in 2NF.

2NF = 1NF + one extra property

Definition: a 1NF table is in 2NF if and only if, given any candidate key and any attribute that is not a constituent of a candidate key, the non-key attribute depends upon the whole of the candidate key rather than just a part of it.

(we will see a more formal definition later on)

Solution: creating two tables, Student with candidate key {Student} and Student Modules candidate key {Student, Module}

Student	Home Address
Jones	Jones' home
Smith	Another place
Wright	Elsewhere
Murphy	Right here
Jones	Jones' home
Kearney	Nowhere

Student	Module
Jones	CA218
Smith	CA218
Wright	CA218
Murphy	CA220
Jones	CA220
Kearney	CA220

Third Normal Form

Let's consider another table:

Module	Year	Best Student	Best Student's DoB
CA218	2008	Jones	23/5/1988
CA220	2008	Smith	13/4/1987
CA222	2008	Jones	23/5/1988
CA218	2007	Murphy	3/11/1986
CA220	2007	Kearney	15/12/1986
CA222	2007	Smith	13/4/1987

Even though Best Student and Best Student's Date of Birth are determined by the whole key {Module, Year} and not part of it, particular Best Student / Best Student Date of Birth combinations are shown redundantly on multiple records, so there is a vulnerability in the table.

This problem is addressed by third normal form (3NF).

3NF = 2NF + one extra property

Definition: a 2NF table is in 3NF if and only if, every non-prime attribute of the table is directly dependent on every key of the table.

Non-prime attribute: any attribute that does not belong to any candidate key.

Directly dependent (a.k.a. non-transitively dependent): $X \rightarrow Z$ (X determines Z) indirectly, by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$ (where it is not the case that $Y \rightarrow X$).

Well-known summary: every non-key attribute "must provide a fact about the key, the whole key, and nothing but the key."

("whole key" implies 2NF, "nothing but the key" implies 3NF)

Solution: creating two tables, ModuleYear and Best Student

Module	Year	Best Student
CA218	2008	Jones
CA220	2008	Smith
CA222	2008	Jones
CA218	2007	Murphy
CA220	2007	Kearney
CA222	2007	Smith

Best Student	Best Student's DoB
Jones	23/5/1988
Smith	13/4/1987
Jones	23/5/1988
Murphy	3/11/1986
Kearney	15/12/1986
Smith	13/4/1987

Boyce-Codd Normal Form

Let's consider yet another table:

Room	Start Time	End Time	Booking Type
Q119	09:00	9:50	Lecture
Q119	10:30	12:15	Exam
Q119	14:00	15:35	Lecture
L221	10:00	11:00	Student Presentation
L221	13:00	14:15	Meeting
L221	15:00	17:30	Meeting

The table lists bookings for two rooms, Q119 and L221. A booking can be defined by the room number and the period for which the room is requested.

For each booking, we also specify the type of booking. Room Q129 is only available for lecture-related bookings (lectures or exams), while room L221 is available only for student presentation and meetings of research groups.

Rooms for lectures and meetings can be booked by regular staff members, while student presentation and exams are organised only by senior staff members.

The table's candidate keys are:

- {Room, Start Time}
- {Room, End Time}
- {Booking Type, Start Time}
- {Booking Type, End Time}

In this table, there is no attribute not belonging to any candidate key (a.k.a non-prime attributes).

A direct consequence is that there is no "partial functional dependencies of non-prime attributes on candidate keys" (i.e. the table is 2NF) and no "transitive functional dependencies of non-prime attributes on candidate keys" (i.e. the table is 3NF).

However, the table is still vulnerable, in the sense that there is nothing to guarantee that the dependency of the booking type on the room is respected. It is, for instance, possible to request Q119 for a lab exam.

Solution: BCNF

Definition: a table is in BCNF if and only if for every one of its non-trivial functional dependencies $X \rightarrow Y$, X is a superkey (i.e. X is either a candidate key or a superset thereof).

Solution: creating two tables, Booking Types and Current Bookings, and using a flag

Booking Type	Room	Senior-only
Lecture	Q119	No
Exam	Q119	Yes
Meeting	L221	No
Student Presentation	L221	Yes

Room	Start Time	End Time	Senior-only
Q119	09:00	9:50	No
Q119	10:30	12:15	Yes
Q119	14:00	15:35	No
L221	10:00	11:00	Yes
L221	13:00	14:15	No
L221	15:00	17:30	No

Note:

- 1NF, 2NF and 3NF are always achievable.
- BCNF is not always achievable.
- there are higher Normal Forms