Database normalization

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as an integral part of his relational model.

Normalization entails organizing the columns (attributes) and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of *synthesis* (creating a new database design) or *decomposition* (improving an existing database design).

Contents

Objectives

A basic objective of the first normal form defined by Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic.[1] (SQL is an example of such a data sub-language, albeit one that Codd regarded as seriously flawed.)[2]

The objectives of normalization beyond 1NF (first normal form) were stated as follows by Codd:

> 1. To free the collection of relations from undesirable insertion, update and deletion dependencies.
> 2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs.
> 3. To make the relational model more informative to users.
> 4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.
>
> — E.F. Codd, "Further Normalization of the Data Base Relational Model"[3]

When an attempt is made to modify (update, insert into, or delete from) a relation, the following undesirable side-effects may arise in relations that have not been sufficiently normalized:

- **Update anomaly.** The same information can be expressed on multiple rows; therefore updates to the relation may result in logical inconsistencies. For example, each record in an "Employees' Skills" relation might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee may need to be applied to multiple records (one for each skill). If the update is only partially successful – the employee's address is updated on some records but not others – then the relation is left in an inconsistent state. Specifically, the relation provides conflicting answers to the question of what this particular employee's address is. This phenomenon is known as an update anomaly.
- **Insertion anomaly.** There are circumstances in which certain facts cannot be recorded at all. For example, each record in a "Faculty and Their Courses" relation might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code. Therefore, we can record the details of any faculty member who teaches at least one course, but we cannot record a newly hired faculty member who has not yet been assigned to teach any courses, except by setting the Course Code to null. This phenomenon is known as an insertion anomaly.
- **Deletion anomaly.** Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts. The "Faculty and Their Courses" relation described in the previous example suffers from this type of anomaly, for if a faculty member temporarily ceases to be assigned to any courses, we must delete the last of the records on which that faculty member appears, effectively also deleting the faculty member, unless we set the Course Code to null. This phenomenon is known as a deletion anomaly.

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

An **update anomaly**. Employee 519 is shown as having different addresses on different records.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

An **insertion anomaly**. Until the new faculty member, Dr. Newsome, is assigned to teach at least one course, his or her details cannot be recorded.

Minimize redesign when extending the database structure

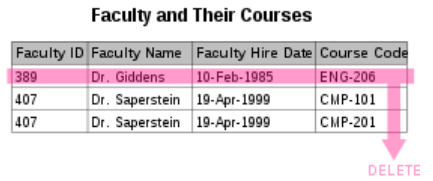
A fully normalized database allows its structure to be extended to accommodate new types of data without changing existing structure too much. As a result, applications interacting with the database are minimally affected.

Normalized relations, and the relationship between one normalized relation and another, mirror real-world concepts and their interrelationships.

Example

Querying and manipulating the data within a data structure that is not normalized, such as the following non-1NF representation of customers, credit card transactions, involves more complexity than is really necessary:

Customer	Cust. ID	Transactions
Abraham	1	Tr. ID / Date / Amount: 12890 / 14-Oct-2003 / −87; 12904 / 15-Oct-2003 / −50
Issac	2	Tr. ID / Date / Amount: 12898 / 14-Oct-2003 / −21
Jacob	3	Tr. ID / Date / Amount: 12907 / 15-Oct-2003 / −18; 14920 / 20-Nov-2003 / −70; 15003 / 27-Nov-2003 / −60



A **deletion anomaly**. All information about Dr. Giddens is lost if he or she temporarily ceases to be assigned to any courses.

To each customer corresponds a 'repeating group' of transactions. The automated evaluation of any query relating to customers' transactions, therefore, would broadly involve two stages:

1. Unpacking one or more customers' groups of transactions allowing the individual transactions in a group to be examined, and
2. Deriving a query result based on the results of the first stage

For example, in order to find out the monetary sum of all transactions that occurred in October 2003 for all customers, the system would have to know that it must first unpack the *Transactions* group of each customer, then sum the *Amounts* of all transactions thus obtained where the *Date* of the transaction falls in October 2003.

One of Codd's important insights was that structural complexity can be reduced. Reduced structural complexity gives users, application, and DBMS more power and flexibility to formulate and evaluate the queries. A more normalized equivalent of the structure above might look like this:

Customer	Cust. ID
Abraham	1
Issac	2
Jacob	3

Cust. ID	Tr. ID	Date	Amount
1	12890	14-Oct-2003	−87
1	12904	15-Oct-2003	−50
2	12898	14-Oct-2003	−21
3	12907	15-Oct-2003	−18
3	14920	20-Nov-2003	−70
3	15003	27-Nov-2003	−60

In the modified structure, the key is {Cust. ID} in the first relation, {Cust. ID, Tr ID} in the second relation.

Now each row represents an individual credit card transaction, and the DBMS can obtain the answer of interest, simply by finding all rows with a Date falling in October, and summing their Amounts. The data structure places all of the values on an equal footing, exposing each to the DBMS directly, so each can potentially participate directly in queries; whereas in the previous situation some values were embedded in lower-level structures that had to be handled specially. Accordingly, the normalized design lends itself to general-purpose query processing, whereas the unnormalized design does not. The normalized version also allows the user to change the customer name in one place and guards against errors that arise if the customer name is misspelled on some records.

Normal forms

Codd introduced the concept of normalization and what is now known as the first normal form (1NF) in 1970.[4] Codd went on to define the second normal form (2NF) and third normal form (3NF) in 1971,[5] and Codd and Raymond F. Boyce defined the Boyce-Codd normal form (BCNF) in 1974.[6]

Informally, a relational database relation is often described as "normalized" if it meets third normal form.[7] Most 3NF relations are free of insertion, update, and deletion anomalies.

The normal forms (from least normalized to most normalized) are:

- UNF: Unnormalized form
- 1NF: First normal form
- 2NF: Second normal form
- 3NF: Third normal form
- EKNF: Elementary key normal form
- BCNF: Boyce–Codd normal form
- 4NF: Fourth normal form
- ETNF: Essential tuple normal form
- 5NF: Fifth normal form
- DKNF: Domain-key normal form
- 6NF: Sixth normal form

	UNF (1970)	1NF (1971)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Primary key (no duplicate tuples)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No repeating groups	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Atomic columns (cells have single value)	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No partial dependencies (values depend on the whole of every Candidate key)	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
No transitive dependencies (values depend only on Candidate keys)	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency involves either a superkey or an elementary key's subkey	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	N/A
No redundancy from any functional dependency	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	N/A
Every non-trivial, multi-value dependency has a superkey	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	N/A
A component of every explicit join dependency is a superkey[8]	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	N/A
Every non-trivial join dependency is implied by a candidate key	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	N/A
Every constraint is a consequence of domain constraints and key constraints	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	N/A
Every join dependency is trivial	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Example of a step by step normalization

Normalization is a database design technique, which is used to design a relational database table up to higher normal form. [9] The process is progressive, and a higher level of database normalization cannot be achieved unless the previous levels have been satisfied. [10]

That means that, having data in unnormalized form (the least normalized) and aiming to achieve the highest level of normalization, the first step would be to ensure compliance to first normal form, the second step would be to ensure second normal form is satisfied, and so forth in order mentioned above, until the data conform to sixth normal form.

However, it is worth noting that normal forms beyond 4NF are mainly of academic interest, as the problems they exist to solve rarely appear in practice [11]

Please note that the data in the following example were intentionally designed to contradict most of the normal forms. In real life, it's quite possible to be able to skip some of the normalization steps because the table doesn't contain anything contradicting the given normal form. It also commonly occurs that fixing a violation of one normal form also fixes a violation of a higher normal form in the process. Also one table has been chosen for normalization at each step, meaning that at the end of this example process, there might still be some tables not satisfying the highest normal form.

Initial data

Let a database table with the following structure: [10]

Book	Author	Author Nationality	Book Type	Price	Subject	Pages	Thickness	Publisher	Publisher Country	Publication Type	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL, Database, Design	520	Thick	Apress	USA	E-book	1	Tutorial

Satisfying 1NF

To satisfy 1NF, we need to ensure that the values in each column of a table are atomic. In the initial table, **Subject** contains a set of values, meaning it does not comply.

One way to achieve the first normal form would be to separate the duplicities into multiple columns:

Book	Book Type	Author	Author Nationality	Price	Subject 1	Subject 2	Subject 3	Pages	Thickness	Publisher	Publisher country	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	MySQL	Database	Design	520	Thick	Apress	USA	1	Tutorial

Although now the table formally complies to the 1NF (is atomic), the problem with this solution is obvious - if a book has more than three subjects, it cannot be added to the database without altering its structure.

To solve the problem in a more elegant way, it is necessary to identify entities represented in the table and separate them into their own respective tables. In this case, it would result in **Book**, **Subject** and **Publisher** tables: [10]

Book

Book	Book Type	Author	Author Nationality	Price	Pages	Thickness	Genre ID	Genre Name	*Publisher ID*
Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	520	Thick	1	Tutorial	*1*

Subject

Subject ID	Subject name
1	MySQL

Publisher

Publisher_ID	Name	Country
1	Apress	USA

2	Database
3	Design

Simply separating the initial data into multiple tables would break the connection between the data. That means we also need to determine the relationships between the newly introduced tables. You might have noticed the *Publisher ID* column in the Book's table - it is a foreign key realizing one-to-many relation between a book and a publisher.

A book can fit many subjects, as well as a subject may correspond to many books. That means we also need to define a many-to-many relationship. We achieve that by creating a link table: [10]

Book - Subject

Book	***Subject ID***
Beginning MySQL Database Design and Optimization	*1*
Beginning MySQL Database Design and Optimization	*2*

Instead of one table in unnormalized form, we now have 4 tables conforming to the 1NF.

Satisfying 2NF

The Book's table has two candidate keys - **{Book}** and **{Book Type}**. There are more rows in the book table (omitted in the previous data for brevity):

Book

Book	**Book Type**	**Author**	**Author Nationality**	**Price**	**Pages**	**Thickness**	**Genre ID**	**Genre Name**	***Publisher ID***
Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	Paperback	E.F.Codd	British	39.99	538	Thick	2	Popular science	*2*
Beginning MySQL Database Design and Optimization	E-book	Chad Russell	American	22.34	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	E-book	E.F.Codd	British	13.88	538	Thick	2	Popular science	*2*

To conform to 2NF and remove duplicities, all the non-key attributes must be dependent on the whole key. This table does not satisfy 2NF because the **Author**, **Author Nationality**, **Pages, Thickness, Genre ID**, **Genre Name** and **Publisher ID** depend only on a part of the compound key - **{Book}** - and only **Price** uniquely depends on the whole keyset (the price depends on the book itself and also on its type - an e-book is usually the cheapest and hadcover the most expensive).

In order to have the table in second normal form, it is therefore necessary to break the table in two:

Book

Book	**Author**	**Author Nationality**	**Pages**	**Thickness**	**Genre ID**	**Genre Name**	***Publisher ID***
Beginning MySQL Database Design and Optimization	Chad Russell	American	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	E.F.Codd	British	538	Thick	2	Popular science	*2*

Book Type - Prices

Book	**Book Type**	**Price**
Beginning MySQL Database Design and Optimization	Hardcover	49.99
The Relational Model for Database Management: Version 2	Paperback	39.99
Beginning MySQL Database Design and Optimization	E-book	22.34
The Relational Model for Database Management: Version 2	E-book	13.88

Now, the book table conforms to 2NF.

Satisfying 3NF

In order to conform to 3NF, the table should not contain transitive dependencies. Note the book table with more rows (previously omitted for brevity):

Book

Book	**Author**	**Author Nationality**	**Pages**	**Thickness**	**Genre ID**	**Genre Name**	***Publisher ID***
Beginning MySQL Database Design and Optimization	Chad Russell	American	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	E.F.Codd	British	538	Thick	2	Popular science	*2*
Learning SQL	Alan Beaulieu	American	338	Slim	1	Tutorial	*3*
SQL Cookbook	Anthony Molinaro	American	636	Thick	1	Tutorial	*3*

The primary key **{Book}** decides the genre - **Genre ID** - that means the **Genre ID** is functionally dependent on the **Book**. and **Genre Name** depends on **Genre ID**. Therefore, there is also a third relation defining a transitive dependency of **Genre Name** on **Book**. That's why this table doesn't satisfy third normal form.

A decomposition of the table needs to be performed in order to get rid of the transitive dependency to satisfy 2NF:

Book

Book	**Author**	**Author Nationality**	**Pages**	**Thickness**	***Genre ID***	***Publisher ID***
Beginning MySQL Database Design and Optimization	Chad Russell	American	520	Thick	*1*	*1*
The Relational Model for Database Management: Version 2	E.F.Codd	British	538	Thick	*2*	*2*
Learning SQL	Alan Beaulieu	American	338	Slim	*1*	*3*
SQL Cookbook	Anthony Molinaro	American	636	Thick	*1*	*3*

Book Genres

Genre ID	**Genre Name**
1	Tutorial
2	Popular science

The data now satisfy third normal form. According to some sources, we could say the data are now "normalized",[7] but the definition of "fully normalized" is somewhat fuzzy and C. J. Date, for example, claims that (loosely put) a database is fully normalized when all the tables are at least in 5NF.[12]

Satisfying EKNF

EKNF falls strictly between 3NF and BCNF and isn't much referenced in the literature. It is intended *"to capture the salient qualities of both 3NF and BCNF"* while avoiding the problems of both (namely, that 3NF is "too forgiving" and BCNF is "prone to computational complexity"). Since it is rarely mentioned in literature, it is not included in this example. [13] For more information on the EKNF, see its own Wikipedia page.

Satisfying BCNF

A relational schema R is considered to be in **Boyce–Codd normal form (BCNF)** if, for every one of its dependencies X → Y, one of the following conditions hold true:

- X → Y is a trivial functional dependency (i.e., Y is a subset of X)
- X is a superkey for schema R [14]

Consider the table in 3NF from the previous step:

Book

Book	**Author**	**Author Nationality**	**Pages**	**Thickness**	***Genre ID***	***Publisher ID***
Beginning MySQL Database Design and Optimization	Chad Russell	American	520	Thick	*1*	*1*
The Relational Model for Database Management: Version 2	E.F.Codd	British	538	Thick	*2*	*2*
Learning SQL	Alan Beaulieu	American	338	Slim	*1*	*3*
SQL Cookbook	Anthony Molinaro	American	636	Thick	*1*	*3*

There is a non-trivial dependency violating BCNF - ***{Book} → {Pages, Thickness, Genre ID, Publisher ID}***. Therefore, the table should be decomposed: [14]

Book

Book	**Pages**	**Thickness**	***Genre ID***	***Publisher ID***
Beginning MySQL Database Design and Optimization	520	Thick	*1*	*1*
The Relational Model for Database Management: Version 2	538	Thick	*2*	*2*
Learning SQL	338	Slim	*1*	*3*
SQL Cookbook	636	Thick	*1*	*3*

Author - Nationality - Book

Author	**Author Nationality**	**Book**
Chad Russell	American	Beginning MySQL Database Design and Optimization
E.F.Codd	British	The Relational Model for Database Management: Version 2
Alan Beaulieu	American	Learning SQL
Anthony Molinaro	American	SQL Cookbook

Now, the **Book** table is BCNF compliant. But what about the **Author - Nationality - Book** table? That one is not in BCNF yet, because there is a functional dependency ***{Author} → {Nationality}*** together with the trivial dependency ***{Book} → {Book}.*** The table needs to be decomposed one more time:

Book

Book	**Pages**	**Thickness**	***Genre ID***	***Publisher ID***
Beginning MySQL Database Design and Optimization	520	Thick	*1*	*1*
The Relational Model for Database Management: Version 2	538	Thick	*2*	*2*
Learning SQL	338	Slim	*1*	*3*
SQL Cookbook	636	Thick	*1*	*3*

Author - Nationality

Author	**Author Nationality**
Chad Russell	American
E.F.Codd	British
Alan Beaulieu	American
Anthony Molinaro	American

Author - Book

Author	**Book**
Chad Russell	Beginning MySQL Database Design and Optimization
E.F.Codd	The Relational Model for Database Management: Version 2
Alan Beaulieu	Learning SQL
Anthony Molinaro	SQL Cookbook

Now, *each attribute represents a fact about the key, the whole key, and nothing but the key*. Therefore BCNF has been achieved. [14]

Satisfying 4NF

Assume the database is owned by a book retailer franchise that has several franchisees that own shops in different locations. And therefore the retailer decided to add a table that contains data about availability of the books at different locations:

Franchisee - Book Location

Franchisee ID	Book	Location
1	Beginning MySQL Database Design and Optimization	California
1	Beginning MySQL Database Design and Optimization	Florida
1	Beginning MySQL Database Design and Optimization	Texas
1	The Relational Model for Database Management: Version 2	California
1	The Relational Model for Database Management: Version 2	Florida
1	The Relational Model for Database Management: Version 2	Texas
2	Beginning MySQL Database Design and Optimization	California
2	Beginning MySQL Database Design and Optimization	Florida
2	Beginning MySQL Database Design and Optimization	Texas
2	The Relational Model for Database Management: Version 2	California
2	The Relational Model for Database Management: Version 2	Florida
2	The Relational Model for Database Management: Version 2	Texas
3	Beginning MySQL Database Design and Optimization	Texas

As this table structure consists of a compound primary key, it doesn't contain any non-key attributes and it's already in BCNF (and therefore also satisfies all the previous normal forms). However, if we assume that all available books are offered in each area, we might notice that the **Book** is not unambiguously bound to a certain **Location** and therefore the table doesn't satisfy 4NF.

That means that, to satisfy the fourth normal form, this table needs to be decomposed as well:

Franchisee - Book

Franchisee ID	Book
1	Beginning MySQL Database Design and Optimization
1	The Relational Model for Database Management: Version 2
2	Beginning MySQL Database Design and Optimization
2	The Relational Model for Database Management: Version 2
3	Beginning MySQL Database Design and Optimization

Franchisee - Location

Franchisee ID	Location
1	California
1	Florida
1	Texas
2	California
2	Florida
2	Texas
3	Texas

Now, every record is unambiguously identified by a superkey, therefore 4NF is satisfied. [15]

Satisfying ETNF

Suppose the franchisees can also order books from different suppliers. Suppose the franchisees can also order books from different suppliers. Let the relation also be subject to the following constraint:

- If a certain **supplier** supplies a certain **book**
- and the **book** is supplied to the **franchisee**
- and the **franchisee** is being supplied by the **supplier,**
- then the **supplier** supplies the **book** to the **franchisee**. [16]

Supplier - Book - Franchisee

Supplier ID	Book	Franchisee ID
1	Beginning MySQL Database Design and Optimization	1
2	The Relational Model for Database Management: Version 2	2
3	Learning SQL	3

This table is in 4NF, but the Supplier ID is equal to the join of its projections: **{ { Supplier ID , Book } , { Book, Franchisee ID } , { Franchisee ID , Supplier ID } }.** No component of that join dependency is a superkey (the sole superkey being the entire heading), so the table does not satisfy the ETNF and can be further decomposed: [16]

Supplier - Book

Supplier ID	Book
1	Beginning MySQL Database Design and Optimization
2	The Relational Model for Database Management: Version 2
3	Learning SQL

Book - Franchisee

Book	Franchisee ID
Beginning MySQL Database Design and Optimization	1
The Relational Model for Database Management: Version 2	2
Learning SQL	3

Franchisee - Supplier

Supplier ID	Book	Franchisee ID
1	Beginning MySQL Database Design and Optimization	1
2	The Relational Model for Database Management: Version 2	2
3	Learning SQL	3

After the decomposition, compliance to ETNF is ensured.

Satisfying 5NF

To spot a table not satisfying the 5NF, it is usually necessary to examine the data thoroughly. Suppose the table from 4NF example with a little modification in data and let's examine if it satisfies 5NF:

Franchisee - Book Location

Franchisee ID	Book	Location
1	Beginning MySQL Database Design and Optimization	California
1	Learning SQL	California
1	The Relational Model for Database Management: Version 2	Texas
2	The Relational Model for Database Management: Version 2	California

If we decompose this table, we lower redundancies and get the following two tables:

Franchisee - Book

Franchisee ID	Book
1	Beginning MySQL Database Design and Optimization
1	Learning SQL
1	The Relational Model for Database Management: Version 2
2	The Relational Model for Database Management: Version 2

Franchisee - Location

Franchisee ID	Location
1	California
1	Texas
2	California

What happens if we try to join these tables? The query would return the following data:

Franchisee - Book - Location JOINed

Franchisee ID	Book	Location
1	Beginning MySQL Database Design and Optimization	California
1	Learning SQL	California
1	The Relational Model for Database Management: Version 2	California
1	The Relational Model for Database Management: Version 2	Texas
1	Learning SQL	Texas
1	Beginning MySQL Database Design and Optimization	Texas
2	The Relational Model for Database Management: Version 2	California

Apparently, the JOIN returns three more rows than it should - let's try to add another table to clarify the relation. We end up with three separate tables:

Franchisee - Book

Franchisee ID	Book
1	Beginning MySQL Database Design and Optimization
1	Learning SQL
1	The Relational Model for Database Management: Version 2
2	The Relational Model for Database Management: Version 2

Franchisee - Location

Franchisee ID	Location
1	California
1	Texas
2	California

Location - Book

Location	Book
California	Beginning MySQL Database Design and Optimization
California	Learning SQL
California	The Relational Model for Database Management: Version 2
Texas	The Relational Model for Database Management: Version 2

What will the JOIN return now? It actually is not possible to join these three tables. That means it wasn't possible to decompose the **Franchisee - Book Location** without data loss, therefore the table already satisfies 5NF. [15]

Satisfying DKNF

Let's have a look at the **Book** table from previous examples and see if it satisfies the Domain Key Normal Form:

Book

Book	Pages	Thickness	*Genre ID*	*Publisher ID*
Beginning MySQL Database Design and Optimization	520	Thick	*1*	*1*
The Relational Model for Database Management: Version 2	538	Thick	*2*	*2*
Learning SQL	338	Slim	*1*	*3*
SQL Cookbook	636	Thick	*1*	*3*

Logically, **Thickness** is determined by number of pages. That means it depends on **Pages** which is not a key. Let's set an example convention saying a book up to 350 pages is considered "slim" and a book over 350 pages is considered "thick".

This convention is technically a constraint but it is neither a domain constraint nor a key constraint; therefore we cannot rely on domain constraints and key constraints to keep the data integrity.

In another words - nothing prevents us to put, for example, "Thick" for a book with 50 pages and that makes the table violate DKNF.

To solve this, we can create a table holding enumeration that defines the **Thickness** and remove that column from the original table:

Thickness Enum

Thickness	Min pages	Max pages
Slim	1	350
Thick	351	999,999,999,999

Book - Pages - Genre - Publisher

Book	Pages	*Genre ID*	*Publisher ID*
Beginning MySQL Database Design and Optimization	520	*1*	*1*
The Relational Model for Database Management: Version 2	538	*2*	*2*
Learning SQL	338	*1*	*3*
SQL Cookbook	636	*1*	*3*

That way, the domain integrity violation has been eliminated, and the table is in DKNF.

Satisfying 6NF

A simple and intuitive definition of the sitxth normal form is that *"a table is in 6NF when* ***the row contains the Primary Key, and at most one, non-key attribute".*** [17]

That means, for example, the **Publishers** table designed while creating the 1NF

Publisher

Publisher_ID	Name	Country
1	Apress	USA

needs to be further decomposed into two tables:

Publisher

Publisher_ID	Name
1	Apress

Publisher country

Publisher_ID	Country
1	USA

Such normalization to 6NF is mostly used in data warehouses where the benefits outweigh the drawbacks. [18]

See also

- Denormalization
- Database refactoring

Notes and references

1. "The adoption of a relational model of data ... permits the development of a universal data sub-language based on an applied predicate calculus. A first-order predicate calculus suffices if the collection of relations is in first normal form. Such a language would provide a yardstick of linguistic power for all other proposed data languages, and would itself be a strong candidate for embedding (with appropriate syntactic modification) in a variety of host languages (programming, command- or problem-oriented)." Codd, "A Relational Model of Data for Large Shared Data Banks" (http://www.acm.org/classics/nov95/toc.html), p. 381
2. Codd, E.F. Chapter 23, "Serious Flaws in SQL", in *The Relational Model for Database Management: Version 2*. Addison-Wesley (1990), pp. 371–389
3. Codd, E.F. "Further Normalization of the Data Base Relational Model", p. 34
4. Codd, E. F. (June 1970). "A Relational Model of Data for Large Shared Data Banks" (http://www.acm.org/classics/nov95/toc.html). *Communications of the ACM*. **13** (6): 377–387. doi:10.1145/362384.362685 (https://doi.org/10.1145%2F362384.362685).
5. Codd, E. F. "Further Normalization of the Data Base Relational Model". (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems", New York City, May 24–25, 1971.) IBM Research Report RJ909 (August 31, 1971). Republished in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6*. Prentice-Hall, 1972.
6. Codd, E. F. "Recent Investigations into Relational Data Base Systems". IBM Research Report RJ1385 (April 23, 1974). Republished in *Proc. 1974 Congress* (Stockholm, Sweden, 1974), N.Y.: North-Holland (1974).
7. Date, C. J. (1999). *An Introduction to Database Systems*. Addison-Wesley. p. 290.
8. Darwen, Hugh; Date, C. J.; Fagin, Ronald (2012). "A Normal Form for Preventing Redundant Tuples in Relational Databases" (https://researcher.watson.ibm.com/researcher/files/us-fagin/icdt12.pdf) (PDF). *Proceedings of the 15th International Conference on Database Theory*. EDBT/ICDT 2012 Joint Conference (http://edbticdt2012.dima.tu-berlin.de/). ACM International Conference Proceeding Series. Association for Computing Machinery. p. 114. doi:10.1145/2274576.2274589 (https://doi.org/10.1145%2F2274576.2274589). ISBN 978-1-4503-0791-8. OCLC 802369023 (https://www.worldcat.org/oclc/802369023). Retrieved 2018-05-22.
9. Kumar, Kunal; Azad, S. K. (October 2017). "Database normalization design pattern" (http://dx.doi.org/10.1109/upcon.2017.8251067). *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*. IEEE. doi:10.1109/upcon.2017.8251067 (https://doi.org/10.1109%2Fupcon.2017.8251067). ISBN 9781538630044.
10. "Database normalization in MySQL: Four quick and easy steps" (https://www.computerweekly.com/tutorial/Database-normalization-in-MySQL-Four-quick-and-easy-steps). *ComputerWeekly.com*. Retrieved 2019-01-21.
11. "Database Normalization: 5th Normal Form and Beyond" (https://mariadb.com/kb/en/library/database-normalization-5th-normal-form-and-beyond/). *MariaDB KnowledgeBase*. Retrieved 2019-01-23.
12. Date, C. J. (2015-12-21). *The New Relational Database Dictionary: Terms, Concepts, and Examples* (https://books.google.cz/books?id=Jx5UCwAAQBAJ&lpg=PT163&ots=-EeyL8dBkn&dq=etnf%20normalization&hl=cs&pg=PT163#v=onepage&q=etnf%20normalization&f=false). "O'Reilly Media, Inc.". p. 163. ISBN 9781491951699.
13. "Additional Normal Forms - Database Design and Relational Theory - page 151" (http://what-when-how.com/Tutorial/topic-1114galv/Database-Design-and-Relational-Theory-167.html). *what-when-how.com*. Retrieved 2019-01-22.
14. Kozubek, Agnieszka (2014-04-03). "The Boyce-Codd Normal Form (BCNF)" (https://www.vertabelo.com/blog/technical-articles/boyce-codd-normal-form-bcnf). *vertabelo*. Retrieved 2019-01-22.
15. "Normalizace databáze" (https://cs.wikipedia.org/w/index.php?title=Normalizace_databáze&oldid=16615346), *Wikipedie* (in Czech), 2018-11-07, retrieved 2019-01-22
16. Date, C. J. (2015-12-21). *The New Relational Database Dictionary: Terms, Concepts, and Examples* (https://books.google.cz/books?id=Jx5UCwAAQBAJ&lpg=PT163&dq=etnf%20normalization&hl=cs&pg=PT138#v=onepage&q=etnf&f=false). "O'Reilly Media, Inc.". p. 138. ISBN 9781491951699.
17. "normalization - Would like to Understand 6NF with an Example" (https://stackoverflow.com/questions/4824714/would-like-to-understand-6nf-with-an-example). *Stack Overflow*. Retrieved 2019-01-23.

18. "Sixth normal form" (https://en.wikipedia.org/w/index.php?title=Sixth_normal_form&oldid=873004660), *Wikipedia*, 2018-12-10, retrieved 2019-01-23

Further reading

- Date, C. J. (1999), *An Introduction to Database Systems (https://web.archive.org/web/20050404010227/http://www.aw-bc.com/catalog/academic/product/0%2C1144%2C0321197844%2C00.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
- H.-J. Schek, P. Pistor Data Structures for an Integrated Data Base Management and Information Retrieval System

External links

- Database Normalization Basics (http://databases.about.com/od/specificproducts/a/normalization.htm) by Mike Chapple (About.com)
- Database Normalization Intro (http://www.databasejournal.com/sqletc/article.php/1428511), Part 2 (http://www.databasejournal.com/sqletc/article.php/26861_1474411_1)
- An Introduction to Database Normalization (http://mikehillyer.com/articles/an-introduction-to-database-normalization/) by Mike Hillyer.
- A tutorial on the first 3 normal forms (http://phlonx.com/resources/nf3/) by Fred Coulson
- DB Normalization Examples (http://www.dbnormalization.com/)
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft
- Database Normalization and Design Techniques (http://www.barrywise.com/2008/01/database-normalization-and-design-techniques/) by Barry Wise, recommended reading for the Harvard MIS.
- A Simple Guide to Five Normal Forms in Relational Database Theory (http://www.bkent.net/Doc/simple5.htm)
- Normalization in DBMS by Chaitanya (beginnersbook.com) (http://beginnersbook.com/2015/05/normalization-in-dbms/)
- A Step-by-Step Guide to Database Normalization (https://www.databasestar.com/normalization-in-dbms/)
- ETNF – Essential tuple normal form (http://researcher.watson.ibm.com/researcher/files/us-fagin/icdt12.pdf)