Database normalization

Database normalization is the process of structuring a database, usually 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 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).

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 normalisation 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 Normalisation of the Data Base Relational Model"[3]

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, their details cannot be recorded.*

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

DELETE

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

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, the details of any faculty member who teaches at least one course can be recorded, but a newly hired faculty member who has not yet been assigned to teach any courses cannot be recorded, 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, the last of the records on which that faculty member appears must be deleted, effectively also deleting the faculty member, unless the Course Code field is set to null. This phenomenon is known as a deletion anomaly.

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.

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, updation, 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 (1970)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Primary key (no duplicate tuples)[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Atomic columns (cells cannot have tables as values)[5]	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either does not begin with a proper subset of a candidate key or ends with a prime attribute (no partial functional dependencies of non-prime attributes on candidate keys)[5]	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (no transitive functional dependencies of	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓

non-prime attributes on candidate keys)[5]											
Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute	✘	✘	✘	✘	✓	✓	✓	✓	✓	✓	N/A
Every non-trivial functional dependency begins with a superkey	✘	✘	✘	✘	✘	✓	✓	✓	✓	✓	N/A
Every non-trivial multivalued dependency begins with a superkey	✘	✘	✘	✘	✘	✘	✓	✓	✓	✓	N/A
Every join dependency has a superkey component[8]	✘	✘	✘	✘	✘	✘	✘	✓	✓	✓	N/A
Every join dependency has only superkey components	✘	✘	✘	✘	✘	✘	✘	✘	✓	✓	N/A
Every constraint is a consequence of domain constraints and key constraints	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘

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]

The data in the following example were intentionally designed to contradict most of the normal forms. In real life, it is 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 exist with the following structure:[10]

Title	Author	Author Nationality	Format	Price	Subject	Pages	Thickness	Publisher
Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL Database Design	520	Thick	Apress

For this example, it is assumed that each book has only one author.

As a prerequisite to conform to the relational model, a table must have a primary key, which uniquely identifies a row. Two books could have the same title, but an ISBN number uniquely identifies a book, so it can be used as the primary key:

ISBN#	Title	Author	Author Nationality	Format	Price	Subject	Pages	Thickne
1590593324	Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL Database Design	520	Thick

Satisfying 1NF

To satisfy First normal form, each column of a table must have a single value. Columns which contain sets of values or nested records are not allowed.

In the initial table, **Subject** contains a set of subject values, meaning it does not comply.

To solve the problem, the subjects are extracted into a separate **Subject** table:[10]

Book

ISBN#	Title	Format	Author	Author Nationality	Price	Pages	Thickness	Publishe
1590593324	Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	520	Thick	Apress

Subject

ISBN#	Subject name
1590593324	MySQL
1590593324	Database
1590593324	Design

A foreign key column to the **Subject**-table is added, which refers to the primary key of the row from which the subject was extracted. The same information is therefore represented but without the use of non-simple domains.

Instead of one table in unnormalized form, there are now two tables conforming to the 1NF.

Satisfying 2NF

The **Book** table has one candidate key (which is therefore the primary key), the composite key **{Title, Format}**.[12] Consider the following table fragment:

Book

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

All of the attributes that are not part of the candidate key depend on *Title*, but only *Price* also depends on *Format*. To conform to 2NF and remove duplicities, every non candidate-key attribute must depend on the whole candidate key, not just part of it.

To normalize this table, make **{Title}** a (simple) candidate key (the primary key) so that every non candidate-key attribute depends on the whole candidate key, and remove *Price* into a separate table so that its dependency on *Format* can be preserved:

Book

Title	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*

Format - Pr

Title
Beginning MySQL Database Design and Optimizatio
Beginning MySQL Database Design and Optimizatio
The Relational Model for Database Managemer Version 2
The Relational Model for Database Managemer Version 2

Now, the **Book** table conforms to 2NF.

Satisfying 3NF

The **Book** table still has a transitive functional dependency ({Author Nationality} is dependent on {Author}, which is dependent on {Title}). A similar violation exists for genre ({Genre Name} is dependent on {Genre ID}, which is dependent on {Title}). Hence, the **Book** table is not in 3NF. To make it in 3NF, let's use the following table structure, thereby eliminating the transitive

functional dependencies by placing {Author Nationality} and {Genre Name} in their own respective tables:

Book

Title	**Author**	**Pages**	**Thickness**	**Genre ID**	***Publisher ID***
Beginning MySQL Database Design and Optimization	Chad Russell	520	Thick	1	*1*
The Relational Model for Database Management: Version 2	E.F.Codd	538	Thick	2	*2*

Format - Price

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

Author

Author	**Author Nationality**
Chad Russell	American
E.F.Codd	British

Genre

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

Satisfying EKNF

The elementary key normal form (EKNF) falls strictly between 3NF and BCNF and is not much discussed 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]

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	Title	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, assuming that all available books are offered in each area, the **Title** 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	Title
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.[14]

Satisfying ETNF

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 **title**
- and the **title** is supplied to the **franchisee**
- and the **franchisee** is being supplied by the **supplier,**
- then the **supplier** supplies the **title** to the **franchisee**.[15]

Supplier - Book - Franchisee

Supplier ID	Title	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:[15]

Supplier - Book

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

Book - Franchisee

Title	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	Franchisee ID
1	1
2	2
3	3

The decomposition produces ETNF compliance.

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	Title	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

Decomposing this table lowers redundancies, resulting in the following two tables:

Franchisee - Book

Franchisee ID	Title
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

The query joining these tables would return the following data:

Franchisee - Book - Location JOINed

Franchisee ID	Title	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

The JOIN returns three more rows than it should; adding another table to clarify the relation results in three separate tables:

Franchisee - Book

Franchisee ID	Title
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	Title
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.[14]

C.J. Date has argued that only a database in 5NF is truly "normalized".[16]

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

Title	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 other words — nothing prevents us from putting, for example, "Thick" for a book with only 50 pages — and this makes the table violate DKNF.

To solve this, a table holding enumeration that defines the **Thickness** is created, and that column is removed from the original table:

Thickness Enum

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

Book - Pages - Genre - Publisher

Title	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 sixth normal form is that *"a table is in 6NF when **the row contains the Primary Key, and at most one other attribute**"*.[17]

That means, for example, the **Publisher** 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

The obvious drawback of 6NF is the proliferation of tables required to represent the information on a single entity. If a table in 5NF has one primary key column and N attributes, representing the same information in 6NF will require N tables; multi-field updates to a single conceptual record will require updates to multiple tables; and inserts and deletes will similarly require operations across multiple tables. For this reason, in databases intended to serve Online Transaction Processing needs, 6NF should not be used.

However, in data warehouses, which do not permit interactive updates and which are specialized for fast query on large data volumes, certain DBMSs use an internal 6NF representation — known as a columnar data store. In situations where the number of unique values of a column is far less than the number of rows in the table, column-oriented storage allow significant savings in space through data compression. Columnar storage also allows fast execution of range queries (e.g., show all records where a particular column is between X and Y, or less than X.)

In all these cases, however, the database designer does not have to perform 6NF normalization manually by creating separate tables. Some DBMSs that are specialized for warehousing, such as Sybase IQ, use columnar storage by default, but the designer still sees only a single multi-

column table. Other DBMSs, such as Microsoft SQL Server 2012 and later, let you specify a "columnstore index" for a particular table.[18]

See also

- Denormalization
- Database refactoring
- Lossless join decomposition

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) Archived (https://web.archive.org/web/20070612235326/http://www.acm.org/classics/nov95/toc.html) June 12, 2007, at the Wayback Machine, 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 Normalisation of the Data Base Relational Model", p. 34*
4. *Codd, E. F. (June 1970). "A Relational Model of Data for Large Shared Data Banks" (https://web.archive.org/web/20070612235326/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) . S2CID 207549016 (https://api.semanticscholar.org/CorpusID:207549016) . Archived from the original (http://www.acm.org/classics/nov95/toc.html) on June 12, 2007. Retrieved August 25, 2005.*
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 May 22, 2018.
9. Kumar, Kunal; Azad, S. K. (October 2017). Database normalization design pattern. 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. S2CID 24491594 (https://api.semanticscholar.org/CorpusID:24491594) .
10. "Database normalization in MySQL: Four quick and easy steps" (https://web.archive.org/web/20170830224213/https://www.computerweekly.com/tutorial/Database-normalization-in-MySQL-Four-quick-and-easy-steps) . ComputerWeekly.com. Archived from the original (https://www.computerweekly.com/tutorial/Database-normalization-in-MySQL-Four-quick-and-easy-steps) on August 30, 2017. Retrieved March 23, 2021.
11. "Database Normalization: 5th Normal Form and Beyond" (https://mariadb.com/kb/en/library/database-normalization-5th-normal-form-and-beyond/) . MariaDB KnowledgeBase. Retrieved January 23, 2019.
12. The table fragment itself has several candidate keys (simple key **{Price}**, and compound keys of Format together with any column except Price or Thickness), but we assume that in the complete table only **{Title, Format}** will be unique.
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 January 22, 2019.
14. "Normalizace databáze" (https://cs.wikipedia.org/w/index.php?title=Normalizace_datab%C3%A1ze&oldid=16615346) , Wikipedie (in Czech), November 7, 2018, retrieved January 22, 2019
15. Date, C. J. (December 21, 2015). The New Relational Database Dictionary: Terms, Concepts, and Examples (https://books.google.com/books?id=Jx5UCwAAQBAJ&q=etnf%20normalization&pg=PT138) . "O'Reilly Media, Inc.". p. 138. ISBN 9781491951699.
16. Date, C. J. (December 21, 2015). The New Relational Database Dictionary: Terms, Concepts, and Examples (https://books.google.com/books?id=Jx5UCwAAQBAJ&q=etnf%20normalization&pg=PT163) . "O'Reilly Media, Inc.". p. 163. 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 January 23, 2019.
18. Microsoft Corporation. Columnstore Indexes: Overview. https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview . Accessed Mar 23, 2020.

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

- Kent, William (February 1983). "A Simple Guide to Five Normal Forms in Relational Database Theory" (http://www.bkent.net/Doc/simple5.htm) . *Communications of the ACM*. **26** (2): 120–125. doi:10.1145/358024.358054 (https://doi.org/10.1145%2F358024.358054) . S2CID 9195704 (https://api.semanticscholar.org/CorpusID:9195704) .
- 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
- Description of the database normalization basics (http://support.microsoft.com/kb/283878) by Microsoft
- 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)

Last edited 7 days ago by William Avery