© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 11 Issue: 02 | April - June 2020
Paper is available at http://www.jrps.in | Email : info@jrps.in

Database Normalization: A Review

Vinita Panwar, Assistant Professor,
RD engineering college, duhai, Ghaziabad

Abstract

Database normalization theory offers formalized guidelines how to reduce data redundancy and thus problems that it causes in databases. More lately, researchers have started to formalize ideas that the problems caused by unwanted dependencies and redundancy can be observed in case of any modular system like software, hardware, or organization. To tackle these problems, they have proposed the theory of normalized systems that complete application frees systems of combinatorial effects and thus the impact of a change in a system does not depend on the system size. The theory of normalized systems do not say much about database normalization, the theories are deeply related. The Common ground of the database normalization theory and the theory of normalized systems.

ISSN : 2278-6848
03
9 772278 684800
© International Journal for Research Publication and Seminar

Keywords: Database Normalization, Formalized Guidelines, Modular System, Organization, Software, Theories, etc.

Introduction

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

Databases have evolved dramatically since their inception in the early 1960s. Navigational databases such as the hierarchical database (which relied on a tree-like model and allowed only a one-to-many relationship), and the network database (a more flexible model that allowed multiple relationships), were the original systems used to store and manipulate data. Although simple, these early systems were inflexible. In the 1980s, relational databases became popular, followed by object-oriented databases in the 1990s. More recently, NoSQL databases came about as a response to the growth of the internet and the need for faster speed and processing of unstructured data. Today, cloud databases and self-driving databases are breaking new ground when it comes to how data is collected, stored, managed, and utilized.

Types of Databases

There are many different types of databases. The best database for a specific organization depends on how the organization intends to use the data.

Relational databases: Relational databases became dominant in the 1980s. Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information.

Object-oriented databases: Information in an object-oriented database is represented in the form of objects, as in object-oriented programming.

Distributed databases: A distributed database consists of two or more files located in different sites. The database may be stored on multiple computers, located in the same physical location, or scattered over different networks.

Data warehouses: A central repository for data, a data warehouse is a type of database specifically designed for fast query and analysis.

NoSQL databases: A NoSQL, or nonrelational database, allows unstructured and semistructured data to be stored and manipulated (in contrast to a relational database, which defines how all data inserted into the database must be composed). NoSQL databases grew popular as web applications became more common and more complex.

Graph databases: A graph database stores data in terms of entities and the relationships between entities.

OLTP databases: An OLTP database is a speedy, analytic database designed for large numbers of transactions performed by multiple users.

These are only a few of the several dozen types of databases in use today. Other, less common databases are tailored to very specific scientific, financial, or other functions. In addition to the different database types, changes in technology development approaches and dramatic advances such as the cloud and automation are propelling databases in entirely new directions. Some of the latest databases include

Open source databases: An open source database system is one whose source code is open source; such databases could be SQL or NoSQL databases.

Cloud databases: A cloud database is a collection of data, either structured or unstructured, that resides on a private, public, or hybrid cloud computing platform. There are two types of cloud database models: traditional and database as a service (DBaaS). With DBaaS, administrative tasks and maintenance are performed by a service provider.

Multimodel database: Multimodel databases combine different types of database models into a single, integrated back end. This means they can accommodate various data types.

Document/JSON database: Designed for storing, retrieving, and managing document-oriented information, document databases are a modern way to store data in JSON format rather than rows and columns.

Self-driving databases: The newest and most groundbreaking type of database, self-driving databases (also known as autonomous databases) are cloud-based and use machine learning to automate database tuning, security, backups, updates, and other routine management tasks traditionally performed by database administrators.

Database Management System

A database typically requires a comprehensive database software program known as a database management system (DBMS). A DBMS serves as an interface between the database and its end users or programs, allowing users to retrieve, update, and manage how the information is organized and optimized. A DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, and backup and recovery.

Some examples of popular database software or DBMSs include MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and DBASE.

Database Challenges

Today's large enterprise databases often support very complex queries and are expected to deliver nearly instant responses to those queries. As a result, database administrators are constantly called upon to employ a wide variety of methods to help improve performance. Some common challenges that they face include:

Absorbing significant increases in data volume: The explosion of data coming in from sensors, connected machines, and dozens of other sources keeps database administrators scrambling to manage and organize their companies' data efficiently.

Ensuring data security: Data breaches are happening everywhere these days, and hackers are getting more inventive. It's more important than ever to ensure that data is secure but also easily accessible to users.

Keeping up with demand: In today's fast-moving business environment, companies need real-time access to their data to support timely decision-making and to take advantage of new opportunities.

Managing and maintaining the database and infrastructure: Database administrators must continually watch the database for problems and perform preventative maintenance, as well as apply software upgrades and patches. As databases become more complex and data volumes grow, companies are faced with the expense of hiring additional talent to monitor and tune their databases.

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 11 Issue: 02 | April - June 2020
Paper is available at http://www.jrps.in | Email : info@jrps.in

Removing limits on scalability: A business needs to grow if it's going to survive, and its data management must grow along with it. But it's very difficult for database administrators to predict how much capacity the company will need, particularly with on-premises databases.

Addressing all of these challenges can be time-consuming and can prevent database administrators from performing more strategic functions.

Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

What is an "inconsistent dependency"? While it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

Some Facts about Database Normalization

- The words normalization and normal form refer to the structure of a database.

- Normalization was developed by IBM researcher E.F. Codd In the 1970s.
- Normalization increases clarity in organizing data in Databases.

Normalization of a Database is achieved by following a set of rules called 'forms' in creating the database.

Database Normalization Rules

The database normalization process is divided into following the normal form:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

First Normal Form (1NF)

Each column is unique in 1NF.

Example:

Sample Employee table, it displays employees are working with multiple departments.

Employee	Age	Department
Melvin	32	Marketing, Sales
Edward	45	Quality Assurances
Alex	36	Human Resource

Employee table following 1NF:

Employee	Age	Department
Melvin	32	Marketing
Melvin	32	Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

Second Normal Form (2NF)

The entity should be considered already in 1NF, and all attributes within the entity should depend solely on the unique identifier of the entity.

Example:

Sample Products table:

Product ID	product	Brand
1	Monitor	Apple
2	Monitor	Samsung
3	Scanner	HP
4	Head phone	JBL

Product table following 2NF:

Products Category table:

Product ID	Product
1	Monitor
2	Scanner
3	Head phone

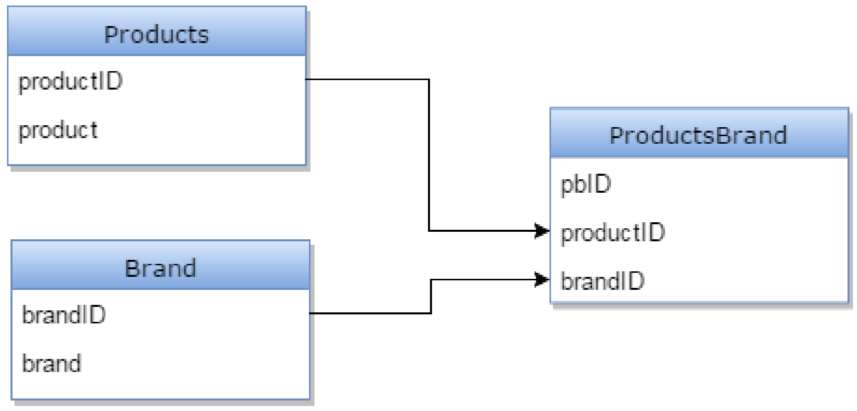
Brand table:

Brand ID	Brand
1	Apple
2	Samsung
3	HP
4	JBL

Products Brand table:

Pb ID	Product ID	Brand ID
1	1	1
2	1	2

3	2	3
4	3	4



Second Normal Form (2NF)

Third Normal Form (3NF)

The entity should be considered already in 2NF, and no column entry should be dependent on any other entry (value) other than the key for the table.

If such an entity exists, move it outside into a new table.

3NF is achieved, considered as the database is normalized.

Boyce-Codd Normal Form (BCNF)

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key.

Example 1 – Find the highest normal form of a relation R(A,B,C,D,E) with FD set as {BC->D, AC->BE, B->E}

Step 1. As we can see, (AC) + ={A,C,B,E,D} but none of its subset can determine all attribute of relation, So AC will be candidate key. A or C can't be derived from any other attribute of the relation, so there will be only 1 candidate key {AC}.

Step 2. Prime attributes are those attribute which are part of candidate key {A, C} in this example and others will be non-prime {B, D, E} in this example.

Step 3. The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attribute.

The relation is in 2nd normal form because BC->D is in 2nd normal form (BC is not a proper subset of candidate key AC) and AC->BE is in 2nd normal form (AC is candidate key) and B->E is in 2nd normal form (B is not a proper subset of candidate key AC).

The relation is not in 3rd normal form because in BC->D (neither BC is a super key nor D is a prime attribute) and in B->E (neither B is a super key nor E is a prime attribute) but to satisfy 3rd normal for, either LHS of an FD should be super key or RHS should be prime attribute.

So the highest normal form of relation will be 2nd Normal form.

Fourth Normal Form (4NF)

Tables cannot have multi-valued dependencies on a Primary Key.

Fifth Normal Form (5NF)

A composite key shouldn't have any cyclic dependencies.

Review of literature

(Jmeter, n.d.) Studied "*What is Normalization*" and found that NORMALIZATION is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data. Normalization divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically. The inventor of the relational model Edgar Codd proposed the theory of normalization with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form.

(SINGH, n.d.) studied "*Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database*" and observed that Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let's discuss about anomalies first then we will discuss normal forms with examples. There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly.

(Pratiyushsah, 2020) studied "*Normal Forms in DBMS*" and found that Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updating anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables. If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

(Ebrahim, n.d.) Studied "*Project-Database Normalization*" and found that A functional dependency is a constraint between two sets of attributes from the database. Suppose that our relational database schema has n attributes A1, A2, ..., An; let us think of the whole database as being described by a single universal relation schema R = {A1, A2, ..., An}. We do not imply that we will actually store the database as a single universal table; we use this concept only in developing the formal theory of data dependencies.

(Eessaar, 2016) studied "*The Database Normalization Theory and the Theory of Normalized Systems: Finding a Common Ground*" and observed that Normalization theory of relational databases dates back to the E.F. Since then it has been extended a lot and the work is ongoing. There are proposals how to apply similar principles in case of other data models like object-oriented data model XML data model or document data model . Database normalization process helps database developers to reduce data redundancy and thus avoid certain update anomalies that appear because there are combinatorial effects (CEs) between propositions that are recorded in a database.

(Bahmani, Naghibzadeh, & Bahmani, 2008) studied "*Automatic database normalization and primary key generation*" and found that Normalization as a method of producing good relational database designs is a well-understood topic in the relational database field. The goal of normalization is to create a set of relational tables with minimum amount of redundant data that can be consistently and correctly modified. The main goal of any normalization technique is to design a database that avoids redundant information and update anomalies. The process of normalization was first formalized by E.F. Codd Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form.

(Wang, Du, & Lehmann, 2010) studied "*Accounting For The Benefits Of Database Normalization*" and found that today's accountants are increasingly called upon to participate in decision-making about the design and development of their company's information systems (IS) due to their knowledge of the business processes and expertise in accounting. Consequently, it would be very beneficial for accountants to have understanding of the tasks and processes of the IS design and implementation, including the pros and cons of database normalization. Moreover, twenty-first century accountants, as end-users of a database system, often need to create, maintain, and query their database.

(Albarak & Bahsoon, 2018) studied "*Prioritizing Technical Debt in Database Normalization Using Portfolio Theory and Data Quality Metrics*" and found that Information systems evolve in response to data growth, improving quality, and changes in users' requirements. The evolution process face many risks, such as cost overruns, schedule delays and increasing chances of failure; it is believed that 80% of the cost is spent on system's evolution. Databases are the core of information systems; evolving databases schemas through refactoring is common practice for improving data qualities and performance, among other structural and behavioural qualities. Database normalization is one of the main principles for relational database design, invented by the Turing Award winner Ted Codd.

Conclusion

This 1st normal form, 2nd normal form and 3rd normal form. These phases are used to normalize the database tables automatically. As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

References

[1] Ernst, N.A. 2012. On the role of requirements in understanding and managing technical debt. Proceedings of the Third International Workshop on Managing Technical Debt (2012), 61–64.

[2] Albarak, M., & Bahsoon, R. (2018). Prioritizing technical debt in database normalization using portfolio theory and data quality metrics. Proceedings - International Conference on Software Engineering, 31–40. https://doi.org/10.1145/3194164.3194170

[3] Bahmani, A. H., Naghibzadeh, M., & Bahmani, B. (2008). Automatic database normalization and primary key generation. Canadian Conference on Electrical and Computer Engineering, (June 2008), 11–16. https://doi.org/10.1109/CCECE.2008.4564486

[4] Ebrahim, M. E. A. (n.d.). Project-DatabaseNormalization. Retrieved from https://www.researchgate.net/publication/333972824_Project-Database_Normalization

[5] Eessaar, E. (2016). The Database Normalization Theory and the Theory of Normalized Systems: Finding a Common Ground. Baltic J. Modern Computing, 4(1), 5–33. Retrieved from https://www.researchgate.net/publication/297731569_The_Database_Normalization_Theory_and_the_Theory_of_Normalized_Systems_Finding_a_Common_Ground/link/56e18d9508ae40dc0abf50a1/download

[6] Jmeter. (n.d.). What is Normalization. Retrieved from https://www.guru99.com/database-normalization.html

[7] Pratiyushsah. (2020). Normal Forms in DBMS. Retrieved from https://www.geeksforgeeks.org/normal-forms-in-dbms/

[8] SINGH, C. (n.d.). Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database. Retrieved from https://beginnersbook.com/2015/05/normalization-in-dbms/

[9] Wang, T. J. (T. J. ., Du, H., & Lehmann, C. M. (2010). Accounting For The Benefits Of Database Normalization. American Journal of Business Education (AJBE), 3(1), 41–52. https://doi.org/10.19030/ajbe.v3i1.371