**DATABASE NORMALIZATION**

**Normalization**

* Database normalization is the process of structuring a database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.
* Database normalization is the process of representing a database in terms of relations in standard normal forms.
* The **normal form** is used to reduce **data redundancy** and improve **data integrity**
* The inventor of the relational model, **Edgar Codd** proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of **Boyce-Codd** Normal Form.

**Normal Form**

Certain rules in database management system design have been developed to better organize tables and minimize anomalies. The stage at which a table is organized is known as its **normal form.**

**Types of Normal forms**

* UNF: Unnormalized form
* 1NF: First normal form
* 2NF: Second normal form
* 3NF: Third normal form
* BCNF: Boyce–Codd normal form
* 4NF: Fourth normal form
* 5NF: Fifth normal form
* DKNF: Domain-key normal form

**Data redundancy**

Data redundancy occurs in database systems that have values repeated unnecessarily in one or more records or fields, within a table, or where the field is replicated/repeated in two or more tables.

**Disadvantages of Data redundancy**

**Data inconsistency**

Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.

**Increase in data corruption**

Data corruption is when data becomes damaged as a result of errors in writing, reading, storage, or processing. When the same data fields are repeated in a database or file storage system, data corruption arises. If a file gets corrupted and an employee tries to open it, they may get an error message and not be able to complete their task.

**Increase in database size**

Data redundancy may increase the size and complexity of a database — making it more of a challenge to maintain. A larger database can also lead to longer load times and a great deal of headaches and frustrations for employees as they’ll need to spend more time completing daily tasks.

**Increase in cost**

When more data is created due to data redundancy, storage costs suddenly increase. This can be a serious issue for organizations who are trying to keep costs low in order to increase profits and meet their goals. In addition, implementing a database system can become more expensive.

**How to reduce data redundancy using database normalization**

* Database normalization is the process of efficiently organizing data in a database so that redundant data is eliminated.
* Normalizing data involves organizing the columns and tables of a database to make sure their dependencies are enforced correctly.
* When it comes to normalizing data, each company has their own unique set of criteria.

**Data integrity**

Data integrity is the overall accuracy, completeness and consistency of data. Data integrity also refers to the safety of data and security. It is maintained by a collection of processes, rules, and standards implemented during the design phase. Data integrity also ensures that your data is safe from any outside forces.

**Types of data integrity**

There are two types of data integrity:

* Physical integrity
* Logical integrity.

**Physical integrity**

Physical integrity is the protection of data’s wholeness and accuracy as its stored and retrieved. Protecting data against external factors, such as natural calamities, power outages, falls under the domain of physical integrity.

**Logical integrity**

Logical integrity keeps data unchanged as it is used in different ways in a relational database. Logical integrity protects data from human error and hackers as well.

There are four types of logical integrity.

**Entity integrity**

Entity integrity relies on the creation of primary keys, or unique values that identify pieces of data, to ensure that data isn’t listed more than once and that no field in a table is null. It is a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

**Referential integrity**

Referential integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database’s structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data is accurate, and/or disallow the entry of data that doesn’t apply.

**Domain integrity**

Domain integrity is the collection of processes that ensure the accuracy of each piece of data in a domain. In this context, a domain is a set of acceptable values that a column is allowed to contain. It can include constraints and other measures that limit the format, type and amount of data entered.

**User-defined integrity**

User-defined integrity involves the rules and constraints created by the user to fit their particular needs. Sometimes entity, referential, and domain integrity aren’t enough to safeguard data. Often, specific business rules must be taken into account and incorporated into data integrity measures.

**Advantages of Data Integrity**

Following are the benefits or advantages of Data Integrity:

* It ensures quality in the product and/or service.
* It ensures safety and privacy of customers e.g. patients, social media users etc.
* It increases confidence of consumers to use online digital applications and tools. This helps to increase businesses in digital economy.
* The data integrity helps to protect data from end to end transfer over transmission medium.
* Stored procedures can be used with ease in order to have complete control of data access.

**Data integrity risks**

Factors that can affect the integrity of the data stored in a database.

* Human error
* Transfer errors
* Bugs and viruses
* Compromised hardware

**Risks to data integrity can easily be minimized or eliminated by doing the following:**

* Limiting access to data and changing permissions to restrict changes to information by unauthorized parties
* Validating data to make sure it is correct both when it is gathered and used
* Backing up data
* Using logs to keep track of when data is added, modified, or deleted
* Conducting regular internal audits
* Using error detection software

**Types of Normal forms**

|  |
| --- |
| * What is Unnormalized form * What are the advantages * What are the disadvantages * Example |

**UNF: Unnormalized form**

* Unnormalized form (UNF), also known as an unnormalized relation or non-first normal form, is a simple database data model lacking the efficiency of database normalization.
* Companies like Google, Amazon and Facebook deal with large amounts of data that are difficult to store efficiently. They use NoSQL databases, which are based on the principles of the unnormalized relational model, to deal with the storage issue.
* Some examples of NoSQL databases are **MongoDB, Apache Cassandra** and **Redis**. These databases are more scalable and easier to query with as they do not involve expensive operations like JOIN.

**Advantages of unnormalized form over normalized forms are:**

* Since there is no relation, querying this data model is simpler.
* Restructuring data is easier.

**Disadvantages of unnormalized form are:**

* Absence of relations means that data is often redundant.
* CRUD operations are plagued by anomalies, and if they are not handled correctly, can result in data inconsistency.

**Example**

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Course** |
| 1. | Jack | * Mathematics * Chemistry |
| 2. | Tim | Chemistry  Physics  English |
| 3. | Ana | * Physics * Chemistry |

**1NF: First normal form**

|  |
| --- |
| * **Criteria of first normal form** * **Example** |

**Criteria of first normal form**

* Every column must be unique in each table.
* Create a separate table for each set of related data.
* All entries must be single-valued and atomic
* There's no top-to-bottom ordering to the rows.
* There's no left-to-right ordering to the columns.
* There are no duplicate rows.

**Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer ID** | **First Name** | **Surname** | **Telephone Number** |
| 123 | Anji | Madamanchi | 9999999999, 9999988888 |
| 456 | Priyanka | Monala | 8888888888, 8888899999 |

**Reason -** The column values are not atomic

Adding more columns

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Customer ID** | **First Name** | **Surname** | **Telephone Number1** | **Telephone Number2** |
| 123 | Anji | Madamanchi | 9999999999 | 9999988888 |
| 456 | Priyanka | Monala | 8888888888 | 8888899999 |

**Reason -** The columns are same (repeated)

Adding another row

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer ID** | **First Name** | **Surname** | **Telephone Number** |
| 123 | Anji | Madamanchi | 9999999999 |
| 123 | Anji | Madamanchi | 9999988888 |
| 456 | Priyanka | Monala | 8888888888 |
| 456 | Priyanka | Monala | 8888899999 |

|  |  |  |
| --- | --- | --- |
| **Telephone Number ID** | **Customer ID** | **Telephone Number** |
| 1 | 123 | 9999999999 |
| 2 | 123 | 9999988888 |
| 3 | 456 | 8888888888 |
| 4 | 456 | 8888899999 |

**Reason –** the "ID" is no longer unique in this solution with duplicated customers.

|  |  |  |
| --- | --- | --- |
| **Customer ID** | **First Name** | **Surname** |
| 123 | Anji | Madamanchi |
| 456 | Priyanka | Monala |

|  |
| --- |
| * **Criteria of second normal form** * **Example** |

**2NF: Second normal form**

**Criteria of second normal form**

* It is in first normal form.
* Every non-prime attribute of the relation is dependent on the whole (fully) of every candidate key.

**Differences between primary key and candidate key**

| **Key** | **Primary Key** | **Candidate key** |
| --- | --- | --- |
| Definition | Primary Key is a unique and non-null key which identify a record uniquely in table. A table can have only one primary key. | Candidate key is also a unique key to identify a record uniquely in a table but a table can have multiple candidate keys. |
| Null | Primary key column value cannot be null. | Candidate key column can have null value. |
| Objective | Primary key is most important part of any relation or table. | Candidate key signifies as which key can be used as Primary Key. |
| Use | Primary Key is a candidate key. | Candidate key may or may not be a primary key. |

**Example**

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Subjects** |
| 111 | Aarya | Math, Science |
| 222 | Anji | English |
| 333 | Priyanka | History, Social Studies |
| 444 | Sonu | Economics |

**Reason** - ‘Subjects’ does not have atomic value

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Subjects** |
| 111 | Aarya | Math |
| 111 | Aarya | Science |
| 222 | Anji | English |
| 333 | Priyanka | History |
| 333 | Priyanka | Social Studies |
| 444 | Sonu | Economics |

Now the above table is in First Normal Form

**Reason -** here the column ‘Subjects’ is only dependent upon the column ‘Name’

|  |  |
| --- | --- |
| **Id** | **Name** |
| 111 | Aarya |
| 222 | Anji |
| 333 | Priyanka |
| 444 | Sonu |

|  |  |
| --- | --- |
| **Id** | **Subjects** |
| 111 | Math |
| 111 | Science |
| 222 | English |
| 333 | History |
| 333 | Social Studies |
| 444 | Economics |

**3NF: Third normal form**

**Criteria of Third normal form**

* It is in second normal form.
* The table or relation should not contain any transitive partial dependency.

**Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 111 | Aarya | 523001 | AP | Ongole |
| 222 | Anji | 500032 | TS | Hyderabad |
| 333 | Priyanka | 534204 | AP | Bhimavaram |
| 444 | Sonu | 524001 | AP | Nellore |

**Reason** - EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key (EMP\_ID). It violates the rule of third normal form.

|  |  |  |
| --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** |
| 111 | Aarya | 523001 |
| 222 | Anji | 500032 |
| 333 | Priyanka | 534204 |
| 444 | Sonu | 524001 |

|  |  |  |
| --- | --- | --- |
| **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 523001 | AP | Ongole |
| 500032 | TS | Hyderabad |
| 534204 | AP | Bhimavaram |
| 524001 | AP | Nellore |

**BCNF: Boyce–Codd normal form**

* BCNF is the advance version of 3NF.
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.

**Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author** | **Nationality** | **Book title** | **Genre** | **Number of pages** |
| William Shakespeare | English | The Comedy of Errors | Comedy | 100 |
| Markus Winand | Austrian | SQL Performance Explained | Textbook | 200 |
| Jeffrey Ullman | American | A First Course in Database Systems | Textbook | 500 |
| Jennifer Widom | American | A First Course in Database Systems | Textbook | 500 |

Two candidate keys – Author and title

|  |  |
| --- | --- |
| **Author** | **Nationality** |
| William Shakespeare | English |
| Markus Winand | Austrian |
| Jeffrey Ullman | American |
| Jennifer Widom | American |

|  |  |  |
| --- | --- | --- |
| **Book title** | **Genre** | **Number of pages** |
| The Comedy of Errors | Comedy | 100 |
| SQL Performance Explained | Textbook | 200 |
| A First Course in Database Systems | Textbook | 500 |

|  |  |
| --- | --- |
| **Author** | **Book title** |
| William Shakespeare | The Comedy of Errors |
| Markus Winand | SQL Performance Explained |
| Jeffrey Ullman | A First Course in Database Systems |
| Jennifer Widom | A First Course in Database Systems |

**4NF: Fourth normal form**

* A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
* Introduced by Ronald Fagin in 1977
* For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

**Example**

|  |  |  |
| --- | --- | --- |
| **STU\_ID** | **COURSE** | **HOBBY** |
| 21 | Computer | Dancing |
| 21 | Math | Singing |
| 34 | Chemistry | Dancing |
| 74 | Biology | Cricket |
| 74 | Physics | Cricket |

|  |  |
| --- | --- |
| **STU\_ID** | **COURSE** |
| 21 | Computer |
| 21 | Math |
| 34 | Chemistry |
| 74 | Biology |
| 74 | Physics |

|  |  |
| --- | --- |
| **STU\_ID** | **HOBBY** |
| 21 | Dancing |
| 21 | Singing |
| 34 | Dancing |
| 74 | Cricket |

**5NF: Fifth normal form**

* The fifth normal form was first described by Ronald Fagin in his 1979.
* Fifth normal form (5NF), also known as project-join normal form (PJ/NF).
* A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

**Example**

|  |  |  |
| --- | --- | --- |
| **Traveling Salesman** | **Brand** | **Product Type** |
| Jack Schneider | Acme | Vacuum Cleaner |
| Jack Schneider | Acme | Breadbox |
| Mary Jones | Robusto | Pruning Shears |
| Mary Jones | Robusto | Vacuum Cleaner |
| Mary Jones | Robusto | Breadbox |
| Mary Jones | Robusto | Umbrella Stand |
| Louis Ferguson | Robusto | Vacuum Cleaner |
| Louis Ferguson | Robusto | Telescope |
| Louis Ferguson | Acme | Vacuum Cleaner |
| Louis Ferguson | Acme | Lava Lamp |
| Louis Ferguson | Nimbus | Tie Rack |

|  |  |
| --- | --- |
| **Traveling Salesman** | **Product Type** |
| Jack Schneider | Vacuum Cleaner |
| Jack Schneider | Breadbox |
| Mary Jones | Pruning Shears |
| Mary Jones | Vacuum Cleaner |
| Mary Jones | Breadbox |
| Mary Jones | Umbrella Stand |
| Louis Ferguson | Telescope |
| Louis Ferguson | Vacuum Cleaner |
| Louis Ferguson | Lava Lamp |
| Louis Ferguson | Tie Rack |

|  |  |
| --- | --- |
| **Brand** | **Product Type** |
| Acme | Vacuum Cleaner |
| Acme | Breadbox |
| Acme | Lava Lamp |
| Robusto | Pruning Shears |
| Robusto | Vacuum Cleaner |
| Robusto | Breadbox |
| Robusto | Umbrella Stand |
| Robusto | Telescope |
| Nimbus | Tie Rack |

|  |  |
| --- | --- |
| **Traveling Salesman** | **Brand** |
| Jack Schneider | Acme |
| Mary Jones | Robusto |
| Louis Ferguson | Robusto |
| Louis Ferguson | Acme |
| Louis Ferguson | Nimbus |

**DKNF: Domain-key normal form**

* Domain-Key Normal Form is the highest form of Normalization.
* Every constraint should be a logical sequence of the domain constraints and key constraints applied to the relation.
* The practical utility of DKNF is less.

|  |  |  |
| --- | --- | --- |
| **Wealthy Person** | **Wealthy Person Type** | **Net Worth in Dollars** |
| Aarya | Millionaire | 124,543,621 |
| Anji | Billionaire | 6,553,228,893 |
| Priyanka | Billionaire | 8,829,462,998 |
| Sonu | Millionaire | 495,565,211 |

|  |  |
| --- | --- |
| **Wealthy Person** | **Net Worth in Dollars** |
| Aarya | 124,543,621 |
| Anji | 6,553,228,893 |
| Priyanka | 8,829,462,998 |
| Sonu | 495,565,211 |

|  |  |  |
| --- | --- | --- |
| **Status** | **Minimum** | **Maximum** |
| Millionaire | 1,000,000 | 999,999,999 |
| Billionaire | 1,000,000,000 | 999,999,999,999 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [**UNF**](https://en.wikipedia.org/wiki/Unnormalized_form) **(1970)** | [**1NF**](https://en.wikipedia.org/wiki/First_normal_form) **(1970)** | [**2NF**](https://en.wikipedia.org/wiki/Second_normal_form) **(1971)** | [**3NF**](https://en.wikipedia.org/wiki/Third_normal_form) **(1971)** | [**BCNF**](https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form) **(1974)** | [**4NF**](https://en.wikipedia.org/wiki/Fourth_normal_form) **(1977)** | [**5NF**](https://en.wikipedia.org/wiki/Fifth_normal_form) **(1979)** | [**DKNF**](https://en.wikipedia.org/wiki/Domain-key_normal_form) **(1981)** |
| Primary key (no duplicate tuples) | Maybe | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| No repeating groups | Maybe | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Atomic columns (cells have single value)[8] | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| 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)[8] | No | No | Yes | Yes | Yes | Yes | Yes | Yes |
| 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)[8] | No | No | No | Yes | Yes | Yes | Yes | Yes |
| Every non-trivial functional dependency begins with a superkey[8] | No | No | No | No | Yes | Yes | Yes | Yes |
| Every non-trivial multivalued dependency begins with a superkey[8] | No | No | No | No | No | Yes | Yes | Yes |
| Every join dependency has only superkey components[8] | No | No | No | No | No | No | Yes | Yes |
| Every constraint is a consequence of domain constraints and key constraints[8] | No | No | No | No | No | No | No | Yes |