# Normalization

Database management world employs Normalization as an elementary database practice. system (DBMS). A method of data organization builds the foundation of a systematic database approach.The system implements organization structures systematically to reduce dependency and excess data storage. Normalization is Data splitting through Normalization produces one or multiple database tables with related descriptions. relationship between tables. The data storage process becomes possible through this approach. The database system achieves effective organization with no data conflicts occurring in its structure. The main purpose of Normalization is to destruct duplicate database entries and build logical data dependencies. dependency are logical. The databased efficiency increases because it maintains and updates more easily over time. expand overtime. Data manipulation processes experience minimal risk of producing anomalies due to this technique. The procedures for inserting, deleting and updating involve data from these anomalies. Such inconsistencies within the data system can harm database integrity as well as generate inaccuracies. exactness of the database.

A standardized method for Normalization exists under the name Normal One normal form leads to another through the application of standardized procedures that enhance relational database structure. Each stage of normalization improves the structure of the database. The series of normal forms improves database structure through a systematic process. of steps called normalization. This process aims to eliminate particular dependencies of information within the database structure. Second Normal Form (2NF) starts the process of removing different types of redundant and dependency issues in the data. The method progresses through 1NF, 2NF, 3NF, and following normal forms. If we stick to these normal A database design derived from normal forms leads to storage efficiency in addition to data access performance improvements. The database structure becomes more efficient alongside lower performance costs and easier administration capabilities and scalable database possibilities. scalable foundation for database growth. Defining types of normalization:

**Normalization**form

**UN-Normalization Form (UNF):**

The Normalization process initiates using the un-normalized form (UNF). No data redundancy or dependency exists when information does not relate to anything that requires whole or partial data. The existence of multi valued attributes alongside repeating groups in data causes difficulties when querying and maintaining such databases. The normalization procedure begins after collecting UNF data through its raw data extraction. As a primary stage of systematic data organization, the data must be transformed from UNF into the first normal form (1NF). The Unormalization forms lack clear guidelines whereas 1NF represents the roughest normalization stage having duplicate data in its table structure.

|  |
| --- |
| **Users** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry, {admin\_id}, {lawyer\_id, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating, {area\_id, area\_name, description}, {day\_of\_week, is\_available}}, {client\_id, date\_of\_birth, gender, {appointment\_id, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at, {review\_id, rating, comment, review\_date}}}) |

Here the Users is assumed as the non-repetitive entities, so it is stored as primary key, other are the attributes contained in the users table, where none of the rules of normalization are applied properly, and all the attributes are associated with a single key primary value that is users. This is the raw set of data before applying any rules and regulations.

**First Normal Form(1NF):**

Tables following the First Normal Form(1NF) must contain single values in each column without attempted subdivision. be further divided. A table must adhere to the First Normal Form by getting rid of all repeating groups. Every table requires only one distinct value which serves as its primary key according to 1NF. Every field in the database must have a single value to achieve proper data organization. Further Normalization becomes possible while removal of redundant data occurs at this stage. The single entity in the UNF remains non-recurring inside small brackets. bracket and 2 groups of repetitive entity group’s attributes in 1NF the one table from the three group of entity emerge from a single UNF. The rules followed by the first Normal forms are:

1. **Atomic Values**: Each attribute consists of atomic/individual values like in Admin-1 admin\_id only contains single admin ID and username consist of only one username, address and so on. Here a single data, single number or a single word is only included once.
2. **No repeating groups:** Multiple columns containing identical data types should be replaced with multiple rows. Dual-row structures have higher effectiveness than multi-column structures. Commit to maintaining single-value records in every field.
3. **Contains unique identifiers:** Each table consists of different unique identifiers which are known as primary key, and it uniquely identifies each row of the tables so there will be no data collision or any mistakes.

**Notation in 1NF:**

* Each entity name receives the hyphen symbol “-1” as an indicator for First Normal Form (1NF).
* Each row in the table gets its uniqueness from the primary key attribute which is marked with (PK) and underlined.
* The asterisk (\*) notation marks down key foreign attributes because they link to principal key elements of linked tables.

**Final first normalization form:**

|  |
| --- |
| **Users-1** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry)  Admins-1 (**admin\_id, user\_id\***)  **Lawyers-1** (**lawyer\_id**, **user\_id\***, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating)  **PracticeAreas-1** (**area\_id**, area\_name, description)  **LawyerPracticeAreas-1** (**lawyer\_id\***, **area\_id\***)  **LawyerAvailability-1** (**lawyer\_id\***, day\_of\_week, is\_available)  **Clients-1** (**client\_id**, **user\_id\***, date\_of\_birth, gender)  **Appointments-1** (**appointment\_id**, **lawyer\_id\***, **client\_id\***, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at)  **Reviews-1** (**review\_id**, **appointment\_id\***, rating, comment, review\_date) |

**Second normal form (2NF):**

Second Normal Form or 2NF constitutes a vital stage of data normalization which aims to remove redundancy while fixing distinct forms of database anomalies. A partial dependency in data occurs when a non-key database attribute requires all or part of a composite primary key for dependency. The table shows such dependency when it uses composite keys that hold multiple primary key attributes although some non-key attributes depend only on selected elements of the composite key. A table with multiple attributes forming the key contains some non-key fields which depend solely on specific key attributes instead of stringing together with all key attributes. Second normal form (2NF) ensures that non-prime attributes depend on the full primary key and not just a partial key through its set of requirements which are described below.

1. **Must be in 1NF**: A first normal form database implementation is essential since all attributes need to represent indivisible entities and rows should not include repeating value groups.
2. **No Partial Dependencies:** The database design violates the rule when any attribute relies on a portion of the composite key instead of depending on the whole key structure.

**Notation in 1NF:**

1. The table requires first meeting the requirement of 1NF which means every attribute should only have minimal atomic values without any repeated column groupings.

2. A primary key composed of multiple attributes forms the basis of such a key structure.

3. A non-key attribute becomes partially dependent on one or multiple segments(s) of the composite primary key when it meets this requirement.

4. The table contains features which do not function as primary key components.

In Users-1,

There is only one single key attribute which uniquely identify all the other non-key attributes thus they are fully functionally dependent on the primary key which is user\_id. Hence,

**FFD: admin\_id** → username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry

Therefore,

|  |
| --- |
| **Users-2** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry) |

In **Admins-1,**

Both attributes present in Admins-1 are keyed attributes. Hence,

FFD:

**admin\_id, user\_id** → X

Therefore,

|  |
| --- |
| **Admins-2** (**admin\_id**,**user\_id\***) |

In **Lawyers-1,**

There are also two key attributes but they both cannot fully functionally depend in any other attribute rather lawyer\_id defines all non-key attributes. Hence,

**PFD:** **lawyer\_id** → specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating

**FFD: Lawyer\_id, user\_id** → X

**Therefore:**

|  |
| --- |
| **Lawyers\_2** (**lawyer\_id**, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating, **user\_id\***) |

In **LawyerPractiseAreas-1,**

The table establishes an extensive connection between Lawyers and PractiseAreas because one lawyer can handle multiple practice areas, and one area belongs to multiple lawyers. It consist of:

**Composite key:** **area\_id\*, lawyer\_id\***

**Partial functional dependency (PFD): area\_id** → area\_name, description

**Full functional dependency (FFD): lawyer\_id, area\_id** → X

Therefore,

|  |
| --- |
| **LawyerPractiseAreas-2** (**area\_id\***, **lawyer\_id\***) **PractiseAreas-2** (**area\_id**, area\_name, description) |

In **Lawyer\_Availability-1,**

The table contains the days during which lawyers make themselves available. The table operates with a composite key structure because lawyers participate in multiple availability days while each day may belong to multiple lawyers. It consists of:

**Composite key:** **lawyer\_id, day\_of\_week**

**Fully functional dependency (FFD):** **lawyer\_id, day\_of\_week** → is\_available

Therefore,

|  |
| --- |
| **Lawyer\_Availability\_2** (**day\_of\_week**, is\_available, **lawyer\_id**\*) |

In **Clients-1,**

There are also two key attributes but they both cannot fully functionally depend on any other attribute rather client\_id defines all non-key attributes. Hence,

**PFD:**  
**client\_id** → date\_of\_birth, gender

client\_id, user\_id → X

**Therefore:**

|  |
| --- |
| **Clients-2** (**client\_id**, date\_of\_birth, gender, **user\_id\***) |

In **Appointments-1,**

The table contains the days during which lawyers make themselves available. The table operates with a composite key structure because lawyers participate in multiple availability days while each day may belong to multiple lawyers.

**Partial functional dependency (PFD): appointment\_id** → appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at

**Full functional dependency (FFD): appointment\_id,** **lawyer\_id, client\_id** → X

Every appointment needs to be associated with a lawyer and client thus the table contains a lawyer\_id field to identify the responsible lawyer for each appointment, so the lawyer\_id and client\_id are foreign keys in this table.

**Therefore:**

|  |
| --- |
| **Appointments-2** (**appointment\_id**, **lawyer\_id\***, **client\_id\***, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at**)** |

In **Reviews-1,**

The Reviews-1 table serves as a repository for client evaluations following their lawyer appointment meetings. A single appointment links each review entry through the appointment\_id foreign key. The table contains no reviews not attached to an appointment. It ensures referential integrity.

**Partial functional dependency (PFD):** review\_id → rating, comment, review\_date

**Full functional dependency (FFD):** review\_id, appointment\_id → X

During a one-to-one relationship an appointment includes zero or one review entry. A review needs to be connected to a single appointment based on the appointment\_id\* constraint which maintains one-to-one relations.

Therefore:

|  |
| --- |
| **Reviews-2** (**review\_id**, rating, comment, review\_date, **appointment\_id**\*) |

**Final 2NF:**

|  |
| --- |
| **Users-2** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry)  **Admins-2** (**admin\_id**,**user\_id\***)  **Lawyers-2** (**lawyer\_id**, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating, **user\_id\***)  **LawyerPractiseAreas-2** (**area\_id\***, **lawyer\_id\***) **PractiseAreas-2** (**area\_id**, area\_name, description)  **Lawyer\_Availability\_2** (**day\_of\_week**, is\_available, **lawyer\_id**\*)  **Clients-2** (**client\_id**, date\_of\_birth, gender, **user\_id\***)  **Appointments-2** (**appointment\_id**, **lawyer\_id\***, **client\_id\***, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at**)**  **Reviews-2** (**review\_id**, rating, comment, review\_date, **appointment\_id**\*) |

**Third normal form 3NF:**

3NF prevents referenced data from including any columns that are not reliant on the primary key because the solution requires foreign keys as references instead of using data from either table. The relationship link between the three tables serves as the best model to demonstrate transitive functional dependency. A functional dependency between tables A and B will make C transitively dependent on the combination of A and B when B depends functionally on C. The removal of transitive functional dependencies of non-prime attributes with respect to every super key serve as another way to describe the transformation. Any RDBMS utilizes super keys as combined column systems that function to identify rows uniquely. Each row requires the minimum number of columns for identification which makes up a super key. (Thakur​, 2016) The rules followed by 3NF are:

1. **ACID Properties:** follows Atomicity, Consistency, Isolation, Durability
2. **Data integrity rules:** This rule maintains the accuracy and consistency of the data in the database which contains entity integrity, referential integrity, domain integrity and user-defined integrity.

Notation followed by 3NF:

1. **Hyphen symbol “-3”:** The table naming convention includes a “-3” suffix which shows that the database follows Third Normal Form (3NF) standards.
2. **Primary Key Notation:** All primary keys receive both underline as well as a PK mark. The table contains a built-in mechanism which grants one-of-a-kind identification to all its rows.
3. **Foreign Key Notation:** After an attribute becomes a Foreign Key it receives an asterisk (\*) as an additional symbol after its name.
4. **Functional Dependencies Notation We use:** The partial dependency found in PFD is one of the conditions that lead to removing tables in 2NF.

In Users-3,

None of the transitive dependency exists. All the attributes are depended on the primary key which is user\_id so the Users-2 is already in 3NF so no changes are needed:

|  |
| --- |
| **Users-3** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry) |

In Admins-2,

None of the transitive dependency exists. All the attributes are depended on the primary key which is admin\_id so the Admins-2 is already in 3NF so no changes are needed:

|  |
| --- |
| **Admins-3** (**admin\_id**, **user\_id\***) |

In Lawyers-2,

All the non-key attributes in the table rely wholly on the lawyer\_id field without transitive dependency relationships between attributes. The admin\_id\* is a foreign key that points to the Admins-3 table to indicate administrative control. It is already on 3NF so no changes are needed.

|  |
| --- |
| **Lawyers-3** (**lawyer\_id**, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating, **user\_id\***) |

For LawyerPractiseAreas-3,

This is a junction table for the many-to-many relationship between lawyers and practice areas. In this table there is no transitive dependency so on the 3NF is already applied here so no changes are needed

|  |
| --- |
| **LawyerPractiseAreas-2** (**area\_id\***, **lawyer\_id\***) |

For PractiseAreas-3,

This table is already in 3NF state as there is no transitive dependency, and all the attributes are depended upon area\_id.

|  |
| --- |
| **PractiseAreas-2** (**area\_id**, area\_name, description) |

For Lawyer\_Availability-3,

A specified lawyer's availability appears in every row according to the selected day. There exists no dependency among non-key attributes where one attribute depends upon another attribute. The table currently exists in 3NF. Already in 3NF.

|  |
| --- |
| **Lawyer\_Availability\_2** (**day\_of\_week**, is\_available, **lawyer\_id**\*) |

For Clients-3,

None of the transitive dependency exists. All the attributes are depended on the primary key which is client\_id so the Clients-2 is already in 3NF so no changes are needed:

|  |
| --- |
| **Clients-3** (**client\_id**, date\_of\_birth, gender, **user\_id\***) |

For Appointments-3,

The appointment\_date and appointment\_time together with notes fields depend on the appointment while lawyer\_id\* and client\_id\* maintain the relationships in the system but do not form transitive dependencies. Already in 3NF.

|  |
| --- |
| **Appointments-3** (**appointment\_id**, **lawyer\_id\***, **client\_id\***, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at**)** |

For Reviews-3,

The review\_id determines the dependencies for rating, comment and review\_date. Appointment\_id\* creates a relation to the appointment table but the review attributes do not rely on appointment attributes. Already in 3NF no need of any changes

|  |
| --- |
| **Reviews-2** (**review\_id**, rating, comment, review\_date, **appointment\_id**\*) |

Final 3NF:

|  |
| --- |
| **Users-3** (**user\_id**, username, password, password\_salt, email, full\_name, phone, address, role, registration\_date, last\_login, is\_active, profile\_image, session\_token, session\_expiry, reset\_token, reset\_token\_expiry)  **Admins-3** (**admin\_id**, **user\_id\***)  **Lawyers-3** (**lawyer\_id**, specialization, practice\_areas, experience\_years, education, license\_number, about\_me, consultation\_fee, is\_verified, is\_available, rating, **user\_id\***)  **LawyerPractiseAreas-2** (**area\_id\***, **lawyer\_id\***)  **PractiseAreas-2** (**area\_id**, area\_name, description)  **Lawyer\_Availability\_2** (**day\_of\_week**, is\_available, **lawyer\_id**\*)  **Clients-3** (**client\_id**, date\_of\_birth, gender, **user\_id\***)  **Appointments-3** (**appointment\_id**, **lawyer\_id\***, **client\_id\***, appointment\_date, appointment\_time, duration, status, notes, created\_at, updated\_at**)**  **Reviews-2** (**review\_id**, rating, comment, review\_date, **appointment\_id**\*) |

# References

Thakur​, S. (2016, 04 14). *Database Normalization: Explain 1NF, 2NF, 3NF, BCNF With Examples*. Retrieved from what is Dbms: : https://whatisdbms.com/database-normalization-explain-1nf-2nf-3nf-bcnf-with-examples/