# Normal Forms

*by Sophia*

WHAT'S COVERED

In this lesson, you will explore the three main normal forms that are used to normalize a database, in two parts. Specifically, this lesson covers:

# 1. The Steps Involved in Normalization

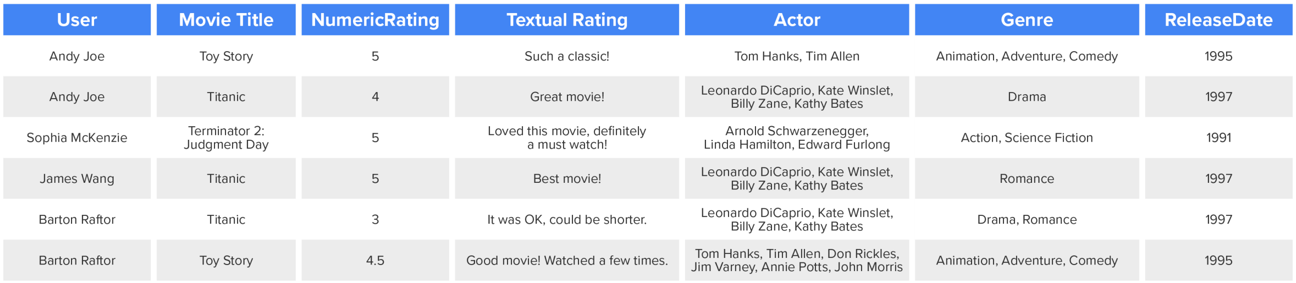
Recall that normalizing the ERD helps database designers focus on evaluating and correcting issues with table structures to help minimize data redundancy and ensure data integrity proactively, before the database is created. The normalization process is a set of steps to work towards refining the database's schema so that it conforms to the rules of each normal form. These steps are fairly standard across database types, so with any database project, you would do these steps in your project to normalize a database.

The following are the steps in the process.

1. Establish the entities: Determine the main entities or objects that need to be represented in the database.
2. Define attributes: Describe each entity's attributes or characteristics.
3. Develop relationships between the entities: Establish relationships between the entities.
4. Apply first normal form (1NF): Ensure each table contains only atomic (indivisible) values. Create separate tables for repeating groups or multivalued attributes.
5. Apply the second normal form (2NF): Ensure that each non-key attribute in a table is functionally dependent on the entire primary key. Create separate tables for partial dependencies if necessary.
6. Apply the third normal form (3NF) and optionally, 4NF and 5NF, if needed: Make sure there are no transitive dependencies among non-key attributes. Create separate tables if necessary to eliminate transitive dependencies. In more complex cases, higher normal forms like the fourth normal form (4NF) or fifth normal form (5NF) may be needed, depending on the data complexity and the database requirements.
7. Testing and refining: Test the normalized database design with sample data and queries. Test results and feedback should be used to refine the design.

To ensure the database remains normalized as the system evolves over time, the database should be reviewed at regular intervals and adjustments to the schema made as needed.

EXAMPLE Each intersection of a row and column in a table should contain only one value, not multiple values. That's what 1NF means by atomicity. In the following spreadsheet that contains data for the movie ratings database, notice that Actor and Genre have multiple values. You will learn in upcoming lessons what needs to be done to correct this problem.



| User | Movie Title | NumericRating | Textual Rating | Actor | Genre | ReleaseDate |
|---|---|---|---|---|---|---|
| Andy Joe | Toy Story | 5 | Such a classic! | Tom Hanks, Tim Allen | Animation, Adventure, Comedy | 1995 |
| Andy Joe | Titanic | 4 | Great movie! | Leonardo DiCaprio, Kate Winslet, Billy Zane, Kathy Bates | Drama | 1997 |
| Sophia McKenzie | Terminator 2: Judgment Day | 5 | Loved this movie, definitely a must watch! | Arnold Schwarzenegger, Linda Hamilton, Edward Furlong | Action, Science Fiction | 1991 |
| James Wang | Titanic | 5 | Best movie! | Leonardo DiCaprio, Kate Winslet, Billy Zane, Kathy Bates | Romance | 1997 |
| Barton Raftor | Titanic | 3 | It was OK, could be shorter. | Leonardo DiCaprio, Kate Winslet, Billy Zane, Kathy Bates | Drama, Romance | 1997 |
| Barton Raftor | Toy Story | 4.5 | Good movie! Watched a few times. | Tom Hanks, Tim Allen, Don Rickles, Jim Varney, Annie Potts, John Morris | Animation, Adventure, Comedy | 1995 |

To comply with 2NF, we will also need to correct the problem with the Actor and Genre attributes not being functionally dependent on the primary key or composite key. They should each be separate tables. We will tackle this work in upcoming lessons as we work through the normalization process.

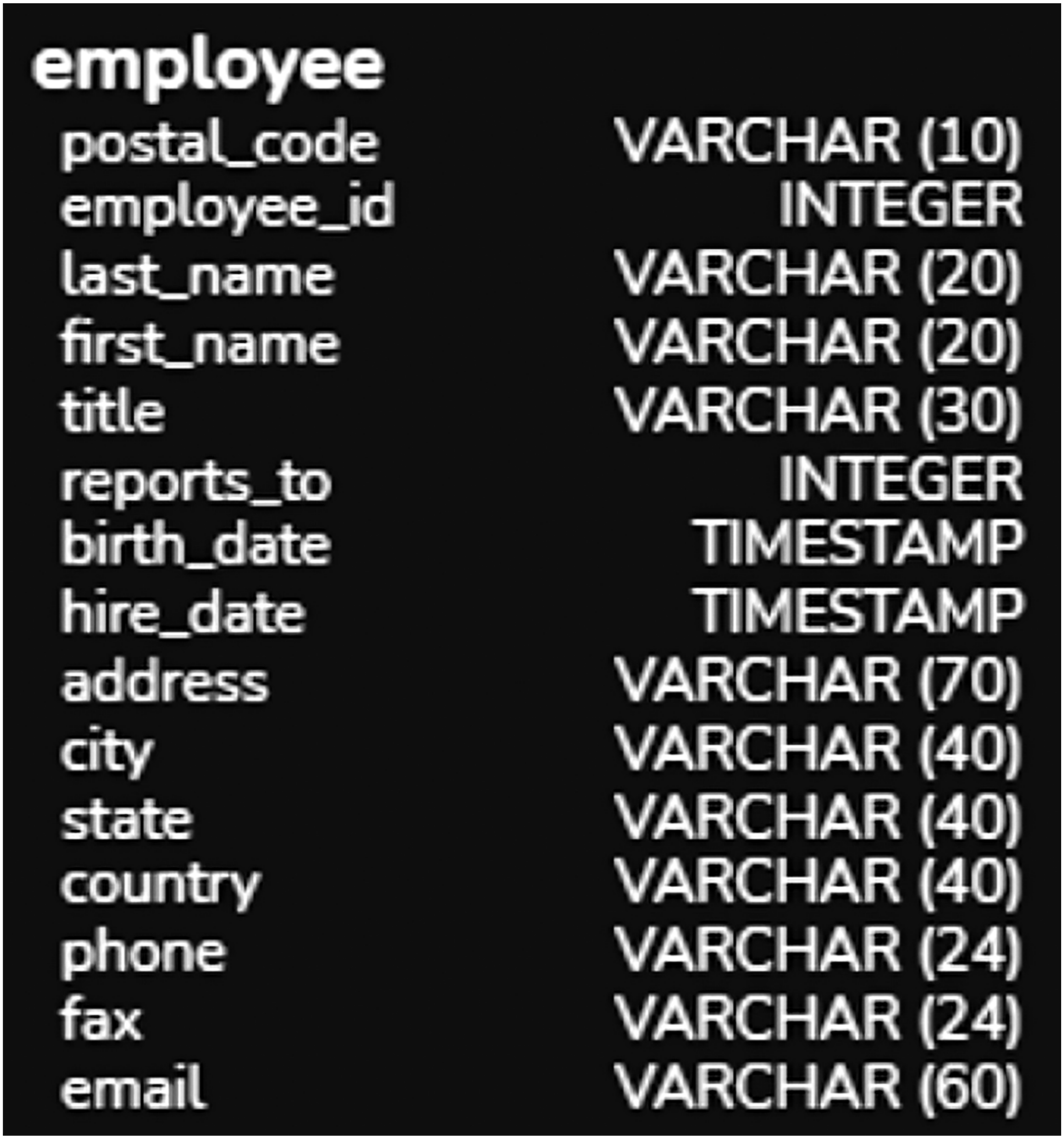
## 2. A Normalization Example

To help reach these objectives, database designers go through the normalization process one stage at a time. For the first normal form (1NF), you ensure that each column in your tables is single valued, meaning it does not contain multiple values like in the Actor column in the above spreadsheet. You also ensure that each attribute or column in a table has a unique name. The first normal form (1NF) also requires you to identify the primary key of each table, with the attributes being dependent on the primary key.

KEY CONCEPT

A commonsense rule with the first normal form (1NF) is to ensure that each of the column values is of the same data type and purpose. This means that if you have a phone number column in the customer table, that column should contain only phone numbers, not names or addresses.

EXAMPLE For the second normal form (2NF), our tables should begin in the first normal form (1NF), but you also ensure there is no partial dependency.



employee

| | |
|---|---|
| postal_code | VARCHAR (10) |
| employee_id | INTEGER |
| last_name | VARCHAR (20) |
| first_name | VARCHAR (20) |
| title | VARCHAR (30) |
| reports_to | INTEGER |
| birth_date | TIMESTAMP |
| hire_date | TIMESTAMP |
| address | VARCHAR (70) |
| city | VARCHAR (40) |
| state | VARCHAR (40) |
| country | VARCHAR (40) |
| phone | VARCHAR (24) |
| fax | VARCHAR (24) |
| email | VARCHAR (60) |

The primary key in the above table is the employee_id. The employee_id is unique for every row, and you can reference a specific record by using it. This allows us to uniquely identify each row, even if you happen to have two employees with the same name.

In the above employee table, each attribute is clearly an attribute that an employee can have. Every attribute depends on the primary key. For example, if you need to find the address of an employee, you can look up the employee's information via their employee_id to find it.

**Partial dependency** can occur when an attribute in a table depends on only a part of the primary key (or composite key) and not the whole key. If this occurs, you need to remove the attribute that is causing partial dependency and move it to another table where it is fully dependent on the primary key.

For the third normal form (3NF), the tables should be in the second normal form (2NF) and additionally have no transitive dependencies. A transitive dependency exists when one or more non-prime attributes is dependent on another non-prime attribute. This is also called a functional dependency.

> EXAMPLE Column X in a table may be dependent on the primary key, but then column Y is dependent on column X. This is a transitive dependency because column Y is only indirectly dependent on the primary key. The solution ends up simple: Remove the columns creating a transitive dependency into another table.

You will learn about higher normal forms, such as 4NF, in a later lesson in which no independent multivalued dependencies exist. But remember that the third normal form (3NF) is typically where the normalization process ends.

TERM TO KNOW

**Partial Dependency**

A condition in which an attribute in a table depends on only a part of the primary key and not the whole key.

## SUMMARY

In this lesson, you learned the **steps involved in normalizing** a database, applying 1NF, 2NF, and 3NF rules in order. You also learned about the three stages of the normalization process, with specific rules to adhere to at each stage. Finally, you reviewed some **examples of normalization** issues and their solutions at each stage.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

## TERMS TO KNOW

**Partial Dependency**

A condition in which an attribute in a table depends on only a part of the primary key and not the whole key.