Normalization Practice

Introduction

Normalization is a process in relational database design used to organize data to reduce redundancy and improve data integrity. It involves decomposing larger tables into smaller, manageable ones and defining relationships among them. Each level of normalization is referred to as a normal form (NF).
This report covers 1NF to 5NF, including BCNF (Boyce-Codd Normal Form), with definitions, use cases, and examples.

Importance of normalized data in databases

Data normalization could be included in your data pipeline, which supports overall visibility into your data, a concept known as data observability.
Ultimately, normalizing your data is one step towards optimizing your data, or maximizing the value you can get from it.
Unfortunately, for many, data optimization is a far-off goal: the data that organizations collect is enormous, but most of that data, in its current form, is rarely useful or valuable on its own. Today, we're living through the early days of AI. If there's one thing we know, it's tha All The Data is needed for AI to succeed.
(Of course, AI needs a lot more than just data: there must be governance, ethics, and frameworks — at bare minimum — to ensure we're getting benefit from AI while reducing harm that we already know it can cause.)
There are many other benefits of normalizing data that we'll explore later on, but first, it's important to explore some key data normalization techniques.

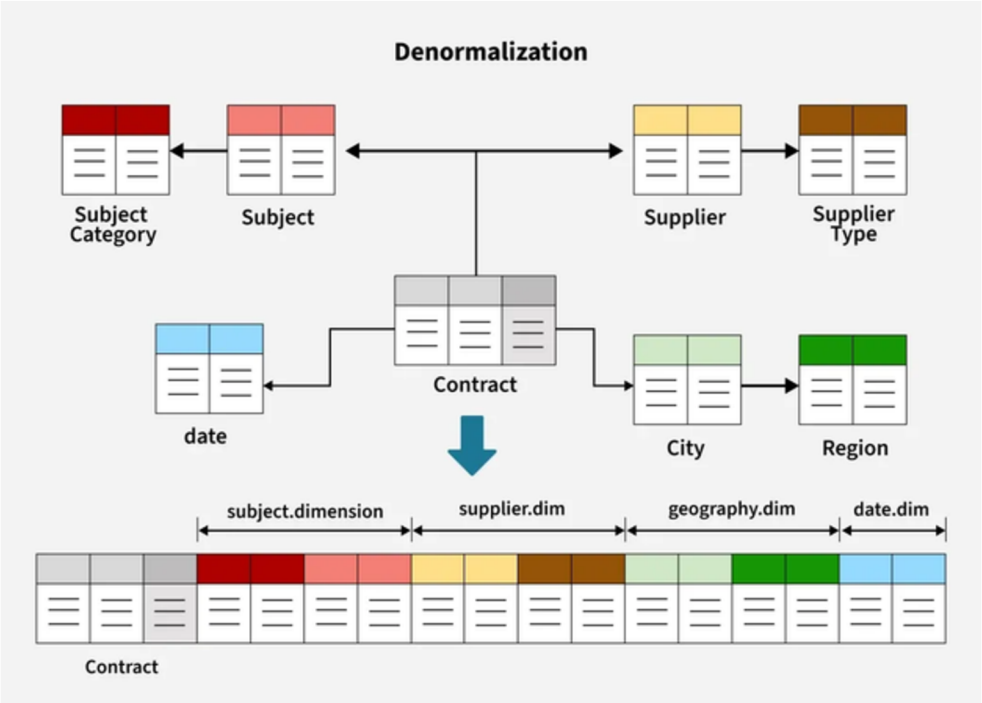
Types of data normalization forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

De-Normalization

Denormalization focuses on combining multiple tables to make queries execute quickly. It adds redundancies in the database though. In this article, we'll explore Denormalization and how it impacts database design. This method can help us to avoid costly joins in a relational database made during normalization.

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean 'reversing normalization' or 'not to normalize'. It is an optimization technique that is applied after normalization.
Basically, The process of taking a normalized schema and making it non-normalized is called denormalization, and designers use it to tune the performance of systems to support time-critical operations.
In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in a database.
For example, in a normalized database, we might have a Courses table and a Teachers table. Each entry in Courses would store the teacherID for a Course but not the teacherName. When we need to retrieve a list of all Courses with the Teacher's name, we would do a join between these two tables.
In some ways, this is great; if a teacher changes his or her name, we only have to update the name in one place. The drawback is that if tables are large, we may spend an unnecessarily long time doing joins on tables. Denormalization, then, strikes a different compromise. Under denormalization, we decide that we're okay with some redundancy and some extra effort to update the database in order to get the efficiency advantages of fewer joins.



❖ **Case1: Company system – UNF table**

EmployeeID	EmployeeName	Department	Projects	Manager
E001	Ahmed	IT	P101:Website, P102:Mobile App	Eng. Khalid
E002	Salim	HR	P103:Recruitment	Ms. Amal
E003	Aisha	IT	P102:Mobile App, P104:Database Upgrade	Eng. Khalid

1NF

EmployeeId	Employee Name	Department	Project	Manger
E001	Ahmed	IT	P101:website	EngKalid
E001	E001	IT	P102:Mobile App	EngKalid
E002	Salim	HR	P103:Recuirment	Ms .Asma
E003	Aisha	IT	P102: Mobile App	EngKalid
E003	Aisha	IT	P104: Mobile App	EngKalid

2NF

Employee Table

EmployeeId	EmployeeName	Department
E001	Ahmed	IT
E001	E001	IT
E002	Salim	HR
E003	Aisha	IT
E003	Aisha	IT

Project Table

Project	Manger
P101:website	EngKalid
P102:Mobile App	EngKalid
P103:Recuirment	Ms .Asma
P102: Mobile App	EngKalid
P104: Mobile App	EngKalid

❖ **Case 2: University System – UNF Table**

StudentID	StudentName	Department	Courses Enrolled	Advisor
S001	Reem	CS	C101:DB, C102:AI	Dr. Omar
S002	Tariq	Business	C103:Marketing	Dr. Sarah
S003	Noura	CS	C101:DB, C104:Cybersecurity	Dr. Omar

1NF

StudentID	StudentName	Department	Courses Enrolled	Advisor
S001	Reem	CS	C101:DB	Dr. Omar
S001	Reem	CS	C102:AI	Dr. Omar
S002	Tariq	Business	C103:Marketing	Dr. Sarah
S003	Noura	CS	C101:DB	Dr. Omar
S003	Noura	CS	C104:Cybersecurity	Dr. Omar

2NF

student Table

StudentID	StudentName	Department
S001	Reem	CS
S001	Reem	CS
S002	Tariq	Business
S003	Noura	CS
S003	Noura	CS

Course Table

Courses Enrolled	Advisor
C101:DB	Dr. Omar
C102:AI	Dr. Omar
C103:Marketing	Dr. Sarah
C101:DB	Dr. Omar
C104:Cybersecurity	Dr. Omar

❖ **Case 3: Airline System – UNF Table**

BookingID	PassengerName	Flights	SeatNumbers	PaymentMethod
B001	Salem	F101:Muscat-Dubai, F102:Dubai-London	12A, 14C	Credit Card
B002	Fatma	F103:Muscat-Istanbul	10B	PayPal
B003	Zayed	F104:Istanbul-Paris, F105:Paris-Oslo	9A, 13A	Credit Card

1NF

BookingID	PassengerName	Flights	SeatNumbers	PaymentMethod
B001	Salem	F101:Muscat-Dubai,	12A	Credit Card
B001	Salem	F102:Dubai London	14C	Credit Card
B002	Fatma	F103:Muscat-Istanbul	10B	PayPal
B003	Zayed	F104:Istanbul-Paris,	9A	Credit Card
B003	Zayed	F105:Paris-Oslo	13A	Credit Card

2NF

BookingID	PassengerName
B001	Salem
B001	Salem
B002	Fatma
B003	Zayed
B003	Zayed

Flights	SeatNumbers	PaymentMethod
F101:Muscat-Dubai,	12A	Credit Card
F102:Dubai London	14C	Credit Card
F103:Muscat-Istanbul	10B	PayPal
F104:Istanbul-Paris,	9A	Credit Card
F105:Paris-Oslo	13A	Credit Card