Database Normal Forms (1)

Normalization is a process of organizing the attributes and tables of a database to minimize data redundancy and improve data integrity. The higher the normal form, the less redundancy the database has. In normalization, the data is divided into several tables linked together with relationships.

what are normal forms?

- they are a key element of database theory
- how do we design tables that are easy to use?
  - set of rules for designing databases
  - they have mathematical underpinnings
- querying a no form database isnt good

Using Normal forms prevents issues

First Normal Form

Each Column shall contain one, and only one, value

table is in the First Normal Form if:

- All entries in a column (attribute) are of the **same data type.**
- Each column contains atomic (indivisible) values, and there are no repeating groups or arrays.
  - a single cell must not hold more than one value (atomicity)

- Each row is unique and identified by a primary key
- no duplicated rows or columns

the below example is bad form, as each row describes *multiple* albums per artist:

Artist	Albums
The Beatles	Yellow Submarine, White Album, Rubber Soul
Milk Can	Make It Sweet
Dresden Dolls	Yes Virginia, No Virginia, The Dresden Dolls

Applying the first normal form:

Lets fix that...

Artist	Album
The Beatles	Yellow Submarine
The Beatles	White Album
The Beatles	Rubber Soul
Milk Can	Make It Sweet
Dresden Dolls	Yes Virginia
Dresden Dolls	No Virginia
Dresden Dolls	The Dresden Dolls

Lets add somne extra info to this table: the year the album released, and the Prime Minister of that year

Artist	Album	Year	Prime Minister
The Beatles	Yellow Submarine	1969	Harold Wilson
The Beatles	White Album	1968	Harold Wilson
The Beatles	Rubber Soul	1965	Harold Wilson
Milk Can	Make It Sweet	1999	Tony Blair
Dresden Dolls	Yes Virginia	2006	Tony Blair
Dresden Dolls	No Virginia	2008	Gordon Brown
Dresden Dolls	The Dresden Dolls	2003	Tony Blair

Second Normal Form

The 1NF only eliminates repeating groups, not redundancy. That's why there is 2NF.

A table is only in 2NF when:

- it is already in 1NF
- it has no partial dependancy that is: **Every non-key attribute is fully dependant on the primary key**
    - bascially every attribute should tell you something about the primary key

    In the above example the key is: *Artist, Album*

        - arguably year as well
- is prime minister dependant on the key? - **No it isnt, so put it in a different table**

below is two tables split between the years

Artist	Album	Year
The Beatles	Yellow Submarine	1969
The Beatles	White Album	1968
The Beatles	Rubber Soul	1965
Milk Can	Make It Sweet	1999
Dresden Dolls	Yes Virginia	2006
Dresden Dolls	No Virginia	2008
Dresden Dolls	The Dresden Dolls	2003

Year	Prime Minister
1969	Harold Wilson
1968	Harold Wilson
1965	Harold Wilson
1999	Tony Blair
2006	Tony Blair
2008	Gordon Brown
2003	Tony Blair

- this seperates the data, say if we got the prime minister of a certain year wrong, we now only have to check that row of that table, and can leave the artist and album table untouched

Third Normal Form

Every non-key attribute must provide a fact about the key, the whole key and nothing but the key

The phrase "have no transitive partial dependency" refers to a condition related to the normalization of databases, specifically addressing the requirements of higher normal forms, such as the Third Normal Form (3NF). To understand this concept, it's essential to break down the terms involved:

- **Dependency**: In the context of databases, a dependency is a relationship where a value of one attribute (column) is determined by the value of another attribute. For example, if we have a table with columns for `EmployeeID`, `EmployeeName`, and `Department`, the `EmployeeName` might be dependent on the `EmployeeID` because the ID uniquely identifies the employee and thus their name.
- **Partial Dependency**: This occurs when a non-primary key attribute is dependent on part of a composite primary key, rather than the whole key. For example, if a table has a composite key consisting of `StudentID` and `CourseID`, and another attribute, `CourseName`, depends only on `CourseID`, then `CourseName` has a partial dependency on the composite key.

- **Transitive Dependency**: A transitive dependency in a database table is a type of functional dependency that occurs when one non-key attribute indirectly depends on the primary key through another non-key attribute. For instance, if `A` determines `B`, and `B` determines `C`, then `C` has a transitive dependency on `A` through `B`.

To "have no transitive partial dependency" means a table is structured such that:

1. **No Partial Dependencies**: Non-key attributes depend on the entire primary key for their values, not just a part of it. This requirement is met in the Second Normal Form (2NF), where the table is already free from partial dependencies—that is, every non-key attribute must depend on the whole of a composite key, not just part of it.
2. **No Transitive Dependencies**: Additionally, no attribute should depend on another attribute through a third attribute, which is the condition for Third Normal Form (3NF). In 3NF, every non-key attribute must be directly dependent on the primary key and not through some transitive chain of dependencies.

Therefore, a table that "has no transitive partial dependency" effectively meets the criteria for being in Third Normal Form (3NF). It ensures a more robust design by eliminating unnecessary redundancy and potential anomalies in data manipulation (insertion, deletion, and update anomalies). This design enhances the integrity and consistency of the database.

lets add some new information to our newly created prime ministers table, each ones birthday

Year	Prime Minister	Birthday
1969	Harold Wilson	1916-03-11
1968	Harold Wilson	1916-03-11
1965	Harold Wilson	1916-03-11
1999	Tony Blair	1953-05-06
2003	Tony Blair	1953-05-06
2006	Tony Blair	1953-05-06
2008	Gordon Brown	1951-02-20

so our key is (year, prime minister)

- birthday depends upon PART of the key — it depends on Prime minister
    - however it doesnt tell you something about the **whole** key (it doesnt tell you about the year they were in power as well)

    So it is in second normal form but not third normal form

    Say if we had to change an incorrect birthday..

Split it up again, have one table for Prime Minister and Year, and a seperate one for Prime Minister and Birthday

This table is now in 3NF

Year	Prime Minister
1969	Harold Wilson
1968	Harold Wilson
1965	Harold Wilson
1999	Tony Blair
2003	Tony Blair
2006	Tony Blair
2008	Gordon Brown

Prime Minister	Birthday
Harold Wilson	1916-03-11
Tony Blair	1953-05-06
Gordon Brown	1951-02-20

Why is this better?

- Now if we need to alter the birthday of a PM (or any other fact about that key)...
- ...then we only need to alter it in one place.

So stopping here is alright, it is a good database design. However you could take things further...

Other Normal Forms:

Boyce-Codd Normal Form / 3.5 Normal Form

- slightly stronger version of third normal form

Every possible candidate key for a table is also in third normal form

- split up a 3nf table into tables with single candidate kets to get 3.5 normal form

4th Normal Form

If multiple attributes in a table depend on the saem key:

- those attributes should be dependant too
- otherwise split them up into their own tables
    - e.g. A pizza restaurant chain that do different styles of pizza and delivers to certain regions then as long as each branch of this chain can always deliver the same style of pizza but to different regions → the style of

pizza and the region are independant therefore should be in seperate tables

5th Normal Form

its in 4th normal form and cannot be seperated into any more tables

- **you split it up as much as possible**
-

To conclude

- as long as you keep things as seperate as possible, you usually hit third normal form by accident
  - practically you are usually at a good point then
  - 5th normal form in the long run is more flexible
  - **3.5 NF database is usually enough**
- design is subjective (usually)
  - mathematical proof of flexibility is however usually a good thing...

GOLDEN RULE OF DATABASES:

Every non key attribute must provide a fact about the key

Examples and Practise Questions

First Normal Form (1NF)

Rule: A table is in 1NF if it has no repeating groups. Each column must contain atomic values, and each column must be unique.

Example:

Imagine a table that tracks customer orders.

- Before 1NF (with repeating groups):

CustomerID	Name	OrderIDs
1	John Doe	101, 102
2	Jane Smith	103
3	Bob Johnson	104, 105, 106

- After applying 1NF:

CustomerID	Name	OrderID
1	John Doe	101
1	John Doe	102
2	Jane Smith	103
3	Bob Johnson	104
3	Bob Johnson	105
3	Bob Johnson	106

In the 1NF version, the OrderIDs column has been split into separate rows to ensure that each row/column intersection has one and only one value.

Second Normal Form (2NF)

Rule: A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

Example:

- Before 2NF (not all non-key attributes fully functionally dependent on the primary key):

OrderID	CustomerName	ProductID	ProductName
101	John Doe	P1	Laptop
102	John Doe	P2	Smartphone

- After applying 2NF (split into two tables):

Table 1: Orders

OrderID	CustomerName	ProductID
101	John Doe	P1
102	John Doe	P2

Table 2: Products

ProductID	ProductName
P1	Laptop
P2	Smartphone

In the 2NF version, we've removed the partial dependency of ProductName on OrderID by splitting the table into two tables: Orders and Products. ProductName is now only dependent on ProductID.

Third Normal Form (3NF)

Rule: A table is in 3NF if it is in 2NF and all its attributes are not only fully functionally dependent on the primary key but also non-transitively dependent on the primary key.

Example:

- Before 3NF (there is a transitive dependency):

Table: Orders

OrderID	CustomerName	CustomerAddress	ProductID
101	John Doe	123 Elm St	P1
102	Jane Smith	456 Pine St	P2

- After applying 3NF (transitive dependency removed):

Table 1: Orders

OrderID	CustomerID	ProductID
101	C1	P1
102	C2	P2

Table 2: Customers

CustomerID	CustomerName	CustomerAddress
C1	John Doe	123 Elm St
C2	Jane Smith	456 Pine St

In the 3NF version, the transitive dependency of CustomerAddress on OrderID (via CustomerName) is removed by splitting the table into Orders and Customers, ensuring each non-key attribute depends only on the primary key.

These examples illustrate the progression through the first three normal forms by eliminating repeating groups, partial dependencies, and transitive dependencies, respectively, to enhance database design and integrity.