Database Normalization

# Introduction

Database normalization was conceived by E.F. Codd. Prevent complexity from becoming unmanageable and keeping copies of things you don’t know you already had somewhere in storage. Permit data to be precisely found. This is the definition from his original paper.

## Codd’s justification for the model

*1. To free the collection of relations from undesirable insertion, update and deletion dependencies;*

*2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs;*

*3. To make the relational model more informative to users;*

*4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.*

— E.F. Codd, "Further Normalization of the Data Base Relational Model" (1970)

In simpler terms it might be rewritten as:

1. Reduce unnecessary insertions, updates and deletions
2. Reduce redundancy
3. Prevent restructuring the data as new information is added
4. Make the data more clear and readily accessible to users

Even more simply it might be:

Maximize atomicity (search accuracy), minimize redundancy (efficiency).

## The Normal Forms

In the original work by EF Codd he describes 5 forms:

* First Normal Form (1NF)
* Second Normal Form (2NF)
* Third Normal Form (3NF)
* Forth Normal Form (4NF)
* Fifth Normal Form (5NF)

Most companies accept databases in 3NF.

In the book normalization is mentioned in only the most generalized way and very briefly. In some cases less is more confusing. There are no practice exam questions that mention normalization; however, if there is one thing to take away for the exam it is the answer to the following question.

Which normal form is most common in databases today? (Answer: 3NF)

I have never heard an interview question referencing “Normal Form”. Occasionally the topic broached by asking about data modeling. We will go over the Data Modeler in this class and reference these rules to confirm our model is appropriate.

Data normalization is complicated and often misunderstood. Many sites contradict each other and have confusing definitions. At the graduate level the rules are written in predicate calculus and proven with math.

## Some Videos for more information

## Are these two videos teaching the same concepts and rules?

<https://www.youtube.com/watch?v=fg7r3DgS3rA>

<https://www.youtube.com/watch?v=UrYLYV7WSHM>

## How is this series of videos different from the previous

[UHCL 28a Graduate Database Course - First and Second Normal Forms](https://www.youtube.com/watch?v=cbJ-xaBfWLM)

[UHCL 29a Graduate Database Course - Third Normal Form and Database Design Shortcuts](https://www.youtube.com/watch?v=nUbp9MRN0To)

Here are some simple examples.

# Example 1

Involves decomposing a table into less redundant (and smaller) tables without losing information, and then linking the data back together by defining foreign keys in the old table referencing the primary keys of the new ones. The objective is to isolate data so that additions, deletions, and modifications of an attribute can be made in just one table and then propagated through the rest of the database using the defined foreign keys.

Here is a table someone created in Excel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Age | Pet\_1 | Pet\_2 | Pet\_Name\_1 | Pet\_name\_2 |
| Heather | 10 | Dog | Cat | Leo | Thomas |
| Bobby | 12 | Dog |  | Leo |  |
| Rachel | 10 | Cat |  | Fluff |  |
| Jimmy | 11 | Dog |  | Kimba |  |
| Heather | 11 | Cat |  | Leo |  |
| Lola | 10 | Cat |  | Thomas |  |

1NF

Make values atomic. Make sure no table contains multiple columns for the same data. Each should have a primary key that distinguishes it as unique. The primary key is usually a single column, but sometimes more than one column can be combined to create a single primary key. Using the rules of first normal form, there may be redundant information across multiple rows, but each row will be unique.

STUDENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student\_ID** | Name | Age | Pet | Pet Name |
| **101** | **Heather** | **10** | Dog | Leo |
| **101** | **Heather** | **10** | Cat | Thomas |
| **465** | Bobby | 12 | Dog | Leo |
| **17** | Rachel | 10 | Cat | Fluff |
| **22** | Jimmy | 11 | Dog | Kimba |
| **156** | Heather | 11 | Cat | Leo |
| **24** | Lola | 10 | Cat | Thomas |

What columns would you have to use to uniquely identify one row?

2NF

1NF is somewhat searchable but had redundant data and therefore is an inefficient use of space. It would be helpful to split out the pets into an independent table, and match them up using the student names as foreign keys.

STUDENTS

|  |  |  |
| --- | --- | --- |
| **Student\_ID** | Name | Age |
| **101** | Heather | 10 |
| **465** | Bobby | 12 |
| **17** | Rachel | 10 |
| **22** | Jimmy | 11 |
| **156** | Heather | 11 |
| **24** | Lola | 10 |

PETS

|  |  |  |  |
| --- | --- | --- | --- |
| **Pet\_ID** | Pet Type | Pet Name | Owner |
| **923** | Dog | Leo | Heather |
| **923** | Dog | Leo | Bobby |
| **71** | Cat | Thomas | Heather |
| **8** | Cat | Fluffy | Rachel |
| **101** | Cat | Leo | Heather |
| **452** | Dog | Kimba | Jimmy |
| **359** | Cat | Thomas | Lola |

3NF

Avoids repeating the age of the student with two pets, so better. But dogs and cats are repeated several times in the pets table. Uncertainty arises when certain data is the same.

Table Pets Suggestion No. 1

Type and Owner

|  |  |  |
| --- | --- | --- |
| **Type** | Pet Name | **Owner** |
| **Dog** | Leo | **Heather** |
| **Dog** | Leo | **Bobby** |
| **Cat** | Thomas | **Heather** |
| **Cat** | Fluffy | **Rachel** |
| **Dog** | Rex | **Bobby** |
| **Cat** | Leo | **Heather** |
| **Dog** | Kimba | **Jimmy** |
| **Cat** | Thomas | **Lola** |

|  |  |
| --- | --- |
| **Suggestion** | **Problem** |
| 1 | 2 Heathers with a Cat |
| 2 | 2 Leos that are Dogs |
| 3 | 2 Heathers with a Leo |

Table Pets Suggestion No. 2

Type and Name

|  |  |  |
| --- | --- | --- |
| **Type** | **Pet Name** | Owner |
| **Dog** | **Leo** | Heather |
| **Dog** | **Leo** | Bobby |
| **Cat** | **Thomas** | Heather |
| **Cat** | **Fluffy** | Rachel |
| **Cat** | **Leo** | Heather |
| **Dog** | **Kimba** | Jimmy |
| **Cat** | **Thomas** | Lola |

|  |  |  |
| --- | --- | --- |
| Type | **Pet Name** | **Owner** |
| Dog | **Leo** | **Heather** |
| Dog | **Leo** | **Bobby** |
| Cat | **Thomas** | **Heather** |
| Cat | **Fluff** | **Rachel** |
| Cat | **Leo** | **Heather** |
| Dog | **Kimba** | **Jimmy** |
| Cat | **Thomas** | **Lola** |

Table Pets Suggestion No. 3

Name and Owner

Solution

Third normal form makes sure each non-key element in each row provides information about the key in the row. In order to establish an unambiguous unique identifier for each pet, it is useful to include a unique primary key that distinguishes each pet from all the others

STUDENTS: 3NF

|  |  |  |
| --- | --- | --- |
| **Student\_ID** | Name | Age |
| **101** | Heather | 10 |
| **465** | Bobby | 12 |
| **17** | Rachel | 10 |
| **22** | Jimmy | 11 |
| **89** | Heather | 11 |
| **24** | Lola | 10 |
| **156** | Heather | 11 |

|  |  |
| --- | --- |
| **Suggestion** | **Problem** |
| 1 | 2 Heathers with a Cat |
|  |  |
|  |  |

PETS: 3NF

|  |  |  |  |
| --- | --- | --- | --- |
| **Pet\_ID** | **Type** | Pet Name | **Owner** |
| **923** | Dog | Leo | Heather |
| **923** | Dog | Leo | Bobby |
| **71** | Cat | Thomas | Heather |
| **8** | Cat | Fluff | Rachel |
| **101** | Cat | Leo | Heather |
| **452** | Dog | Kimba | Jimmy |
| **339** | Cat | Thomas | Lola |

4NF

Which pet owned by which student is clear But what about Heather and Bobby co-owning Leo? We need one more table.

STUDENTS: 4NF

|  |  |  |
| --- | --- | --- |
| **Student\_ID** | Name | Age |
| **101** | Heather | 10 |
| **465** | Bobby | 12 |
| **17** | Rachel | 10 |
| **22** | Jimmy | 11 |
| **89** | Heather | 11 |
| **24** | Lola | 10 |
| **156** | Heather | 11 |

OWNERSHIP: 4NF

|  |  |
| --- | --- |
| **Pet\_ID** | **Own\_ID** |
| **923** | **101** |
| **923** | **465** |
| **71** | **101** |
| **8** | **17** |
| **101** | **89** |
| **452** | **22** |
| **339** | **24** |

PETS: 4Nf

|  |  |  |
| --- | --- | --- |
| **Pet\_ID** | Pet Type | Pet Name |
| **923** | Dog | Leo |
| **923** | Dog | Leo |
| **71** | Cat | Thomas |
| **8** | Cat | Fluffy |
| **101** | Cat | Leo |
| **452** | Dog | Kimba |
| **359** | Cat | Thomas |

# Example 2

<http://agiledata.org/essays/dataNormalization.html>

1NF  
Atomic

No repeating groups

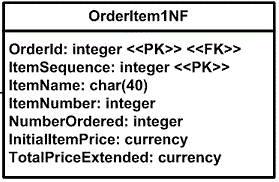
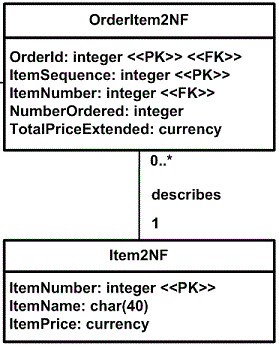
2NF

Each field **fully** **dependent** on the key identifier

Independent of Order

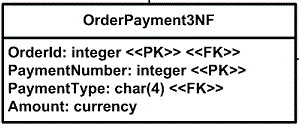
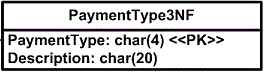
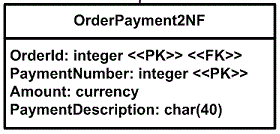
Name Bronze 3” paper clip

Price $0.79



3NF

Each field **directly** **dependent** on the primary key



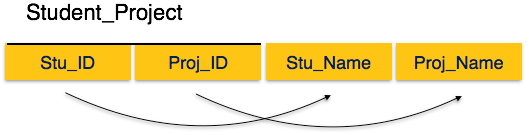
# Example 3

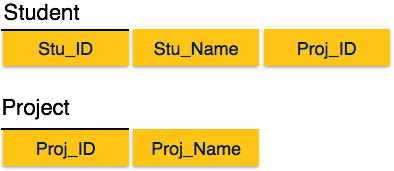
1NF  
Atomic

No repeating groups

2NF

Each field **fully** **dependent** on the key identifier





3NF

Each field **directly dependent** on the primary key

City can be identified by the student ID here and by Zip

1. City is not a prime attribute of Student ID. Many Students can have the same city
2. City can also be found by ZIP which coud also be found by Student ID



