



Normalization

Overview

# Objective

- Normalization presents a set of rules that tables and databases must follow to be well structured.
- Historically presented as a sequence of normal forms

# First Normal Form, 1NF

- A table is in the first normal form if
  - The domain of each attribute contains only atomic values, and
  - The value of each attribute contains only a single value from that domain.

In simple terms it means every column of your table should only contain single values

# Example

- For a library

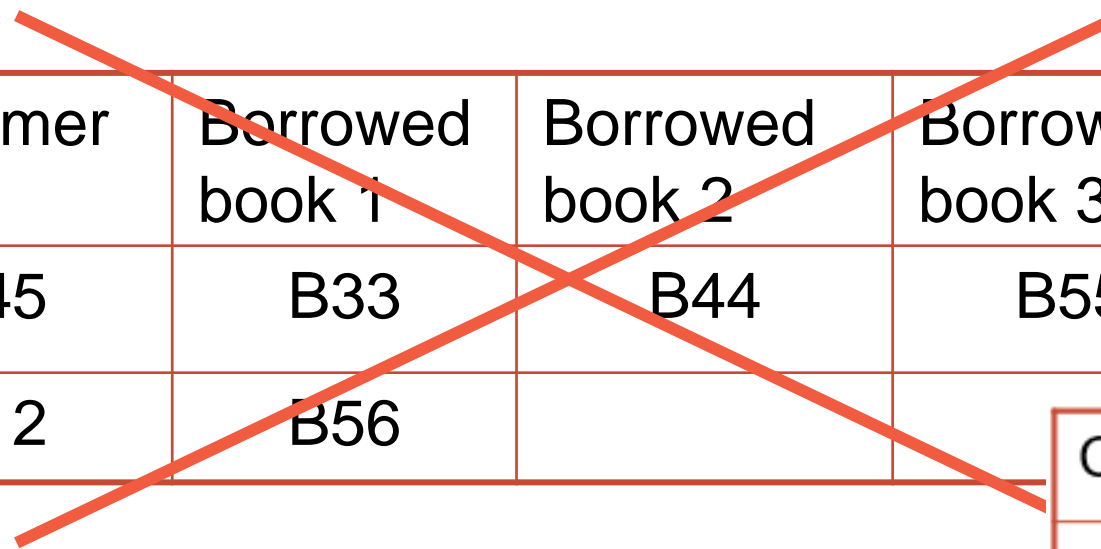
Customer ID	Borrowed books
C45	B33, B44, B55
C12	B56

# 1-NF Solution

Customer ID	Borrowed book
C45	B33
C45	B44
C45	B55
C12	B56

# 1-NF Solution

- “Columns are expensive, rows are cheap”



Customer ID	Borrowed book 1	Borrowed book 2	Borrowed book 3
C45	B33	B44	B55
C12	B56		

Customer ID	Borrowed book
C45	B33
C45	B44
C45	B55
C12	B56

# Example

- For an airline

Flight	Weekdays
UA59	Mo We Fr
UA73	Mo Tu We Th Fr

# 1 NF Solution

Flight	Weekday
UA59	Mo
UA59	We
UA59	Fr
UA73	Mo
UA73	We
...	...



# Implication for the ER model

- Watch for entities that can have multiple values for the same attribute
  - Phone numbers, addresses, favorite foods, cars owned
- What about single attributes with multiple data elements?
  - Lecture time 5:30-7:00pm
- How do we record this?
  - Separate start-end times
  - Text content
  - Atomic time slot
  - Named time slot

# Functional Dependencies

- We say an attribute B has a functional dependency on another attribute A if, for any two records which have the same value for A, the values for B in these two records are always the same
- We illustrate this as:  
 $A \rightarrow B$
- Example: Suppose we keep track of employee home addresses for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of home address on employee number:
  - employee number  $\rightarrow$  email address

# Functional Dependencies

<u>EmpNum</u>	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

- If EmpNum is the PK then the FDs are:  
EmpNum → EmpEmail  
EmpNum → EmpFname  
EmpNum → EmpLname

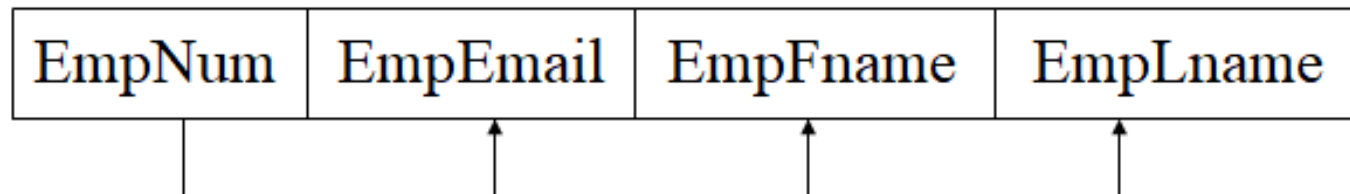
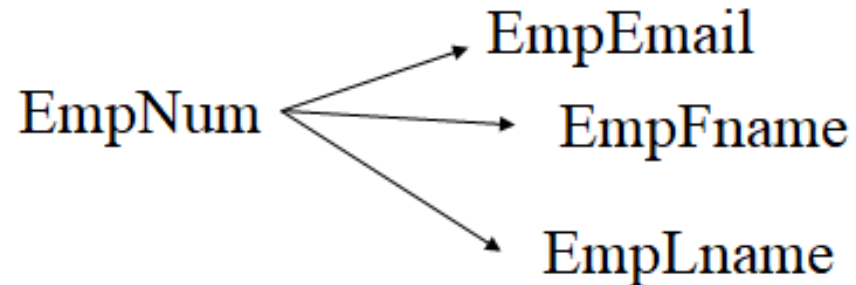
# Functional Dependencies

$\text{EmpNum} \rightarrow \text{EmpEmail}$

$\text{EmpNum} \rightarrow \text{EmpFname}$

$\text{EmpNum} \rightarrow \text{EmpLname}$

*3 different ways  
you might see FDs  
depicted*



# Determinant

- Functional Dependency
  - $\text{EmpNum} \rightarrow \text{EmpEmail}$
- Attribute on the left-hand side is the determinant
  - EmpNum is a determinant of EmpEmail

# Transitive dependency

- Consider the attributes A, B, and C, where  
 $A \rightarrow B$  and  $B \rightarrow C$
- Functional dependencies are *transitive*,
- The above results in the functional dependency  
 $A \rightarrow C$
- We say that C is *transitively dependent* on A through B
- E.g. Person  $\rightarrow$  Car  $\rightarrow$  Car Price

# Example

Book table

<u>BookNo</u>	Title	Author	Year
B1	Moby Dick	H. Melville	1851
B2	Lincoln	G. Vidal	1984

The Author attribute is:

- functionally dependent on Title
- transitively dependent on BookNo (because Title is functionally dependent on BookNo)
- functionally dependent on BookNo

# Why does this matter

table BorrowedBooks

<u>BookNo</u>	<u>Customer</u>	Address	Due
B1	J. Fisher	101 Main Street	3/2/15
B2	L. Perez	202 Market Street	2/28/15

- The Address attribute is
- Functionally dependent on the pair {BookNo, Customer}
- Fully functionally dependent on Customer



# Problems of dependencies

## Insertion anomaly

- Cannot insert new customers in the system until they have borrowed books

## Update anomaly

- Must update all rows involving a given customer if he or she changes address

## Deletion anomaly

- Will lose information about customers that have returned all the books they have borrowed

# Second Normal Form

- A table is in 2NF if and only if
  - **It is in 1NF and**
  - **no non-prime attribute is dependent on any subset of the primary key of the table**
- A non-prime attribute of a table is an attribute that is not a part of the primary key of the table
- If the primary key is not composite, 1NF is also 2NF

# Example

- Library allows customers to request books that are currently out

<u>BookNo</u>	<u>Customer</u>	Due date	PhoneNo
B3	J. Fisher	12/1/2013	555-1234
B2	J. Fisher	16/1/2013	555-1234
B2	M. Amer	29/1/2013	555-4321

# Example

- Candidate key (primary key) is {BookNo, Customer}
- We have
  - Customer  $\rightarrow$  PhoneNo
- Table is not 2NF
  - Potential for
    - Insertion anomalies
    - Update anomalies
    - Deletion anomalies

# 2NF Solution

- Put telephone number in separate Customer table

<u>BookNo</u>	<u>Customer</u>	Due date
B3	J. Fisher	12/1/2013
B2	J. Fisher	16/1/2013
B2	M. Amer	29/1/2013

<u>Customer</u>	PhoneNo
J. Fisher	555-1234
M. Amer	555-4321

# Third Normal Form

- A table is in 3NF if and only if
  - **it is in 2NF and**
  - **all its attributes are determined only by its primary key and not by any non-prime attributes**

# Example

- Table Rented Cars

<u>Hire Date</u>	<u>Customer</u>	License Plate	Car Fuel
16/1/2017	J. Fisher	AB1234F	Hybrid
22/2/2017	L. Perez	ST7890Q	Petrol

- Candidate key is {Hire Date, Customer}
- License Plate → Car Fuel

# 3NF Solution

<u>Hire Date</u>	<u>Customer</u>	License Plate
16/1/2017	J. Fisher	AB1234F
22/2/2017	L. Perez	ST7890Q

<u>License Plate</u>	Car Fuel
AB1234F	Hybrid
ST7890Q	Petrol



# Another example

- Tournament winners

Tournament	Year	Winner	DOB
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 Sept. 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975

- Candidate key is {Tournament, Year}
- Winner → DOB

# Normalisation Example

- We have a table representing orders in an online store
- Each row represents an item on a particular order
- Primary key is {Order, Product}
- Columns
  - Order
  - Product
  - Quantity
  - UnitPrice
  - Customer
  - Address

# Functional Dependencies

- Each order is for a single customer:
  - Order  $\rightarrow$  Customer
- Each customer has a single address
  - Customer  $\rightarrow$  Address
- Each product has a single price
  - Product  $\rightarrow$  UnitPrice
- As Order  $\rightarrow$  Customer and Customer  $\rightarrow$  Address
  - Order  $\rightarrow$  Address

# 2NF Solution (I)

- First decomposition
  - First table

<u>Order</u>	<u>Product</u>	Quantity	UnitPrice
--------------	----------------	----------	-----------

- Second table

<u>Order</u>	Customer	Address
--------------	----------	---------

# 2NF Solution (II)

- Second decomposition

- First table

<u>Order</u>	<u>Product</u>	Quantity
--------------	----------------	----------

- Second table

<u>Order</u>	Customer	Address
--------------	----------	---------

- Third table

<u>Product</u>	UnitPrice
----------------	-----------

# 3NF

- In second table
  - Customer → Address

<u>Order</u>	Customer	Address
--------------	----------	---------

- Split second table into

<u>Order</u>	Customer
--------------	----------

<u>Customer</u>	Address
-----------------	---------

# Normalisation

- 1NF:
  - (Order, Product, Customer, Address, Quantity, UnitPrice)
- 2NF:
  - (Order, Customer, Address)
  - (Product, UnitPrice)
  - (Order, Product, Quantity)
- 3NF:
  - (Product, UnitPrice)
  - (Order, Product, Quantity)
  - (Order, Customer)
  - (Customer, Address)

# Normalization considerations

- Normalization is performed to reduce or eliminate Insertion, Deletion or Update anomalies
- However, a completely normalized database may not be the most efficient or effective implementation
- “Denormalization” is sometimes used to improve efficiency



# “Normalizing to death”

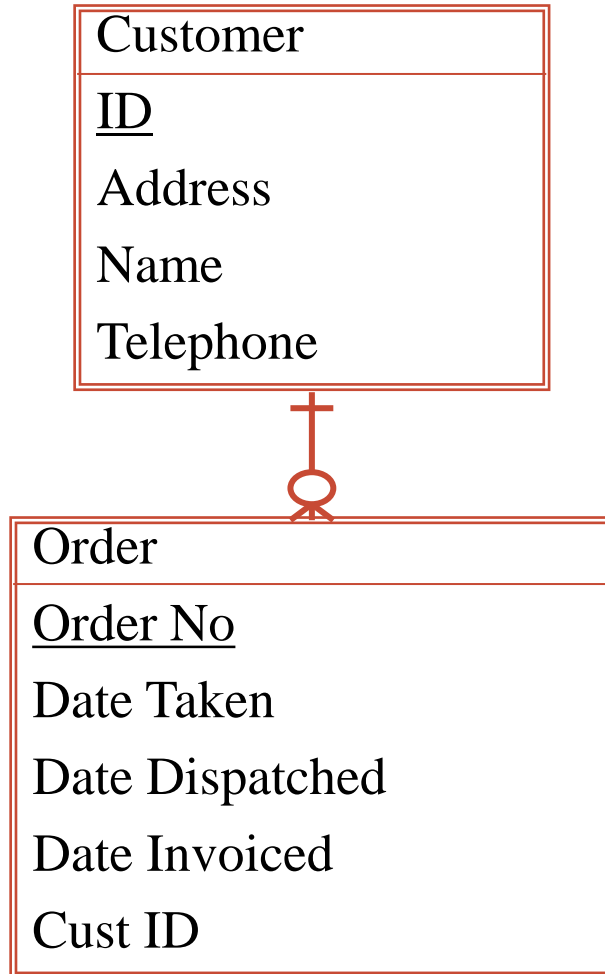
- Normalization splits database information across multiple tables
- To retrieve complete information from a normalized database, the JOIN operation must be used
- JOIN tends to be expensive in terms of processing time, and very large joins are even more expensive

# Denormalization

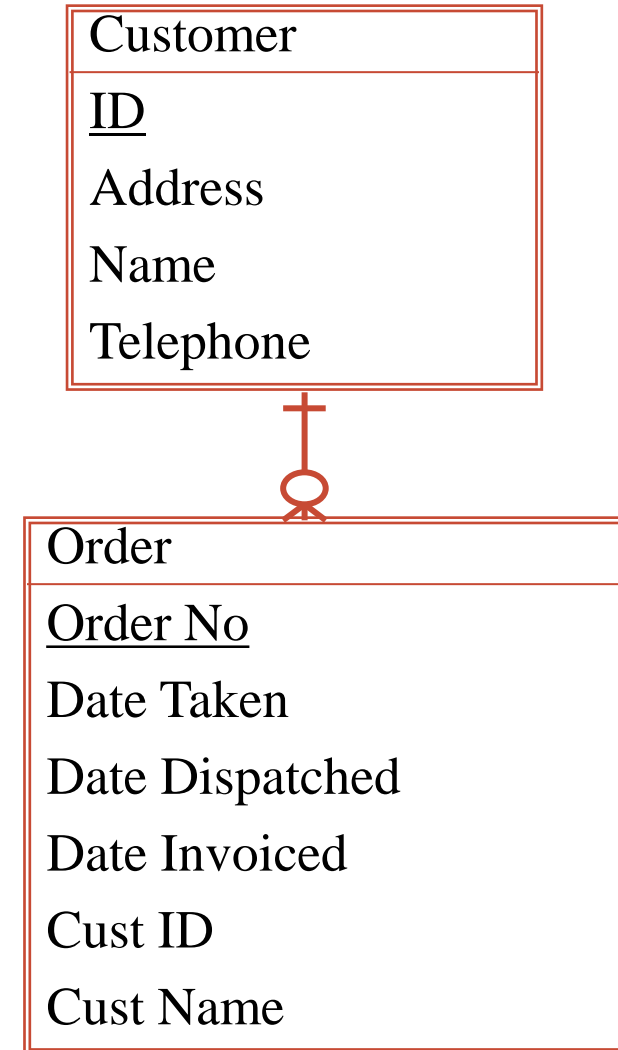
- A normalized design will often store different but related pieces of information in separate tables which makes access of the complete information slower than optimum
- From Wikipedia: denormalization is the process of attempting to optimize the read performance of a database by adding copies of data from one table to another or by grouping data
- To do this, constraints must be added to the database to show how the copies of data will be updated
- Eventually queries run faster but the design is more complex or even problematic for updates, deletions and insertions
- A denormalized data model is not the same as a data model that has not been normalized

# Downward Denormalization

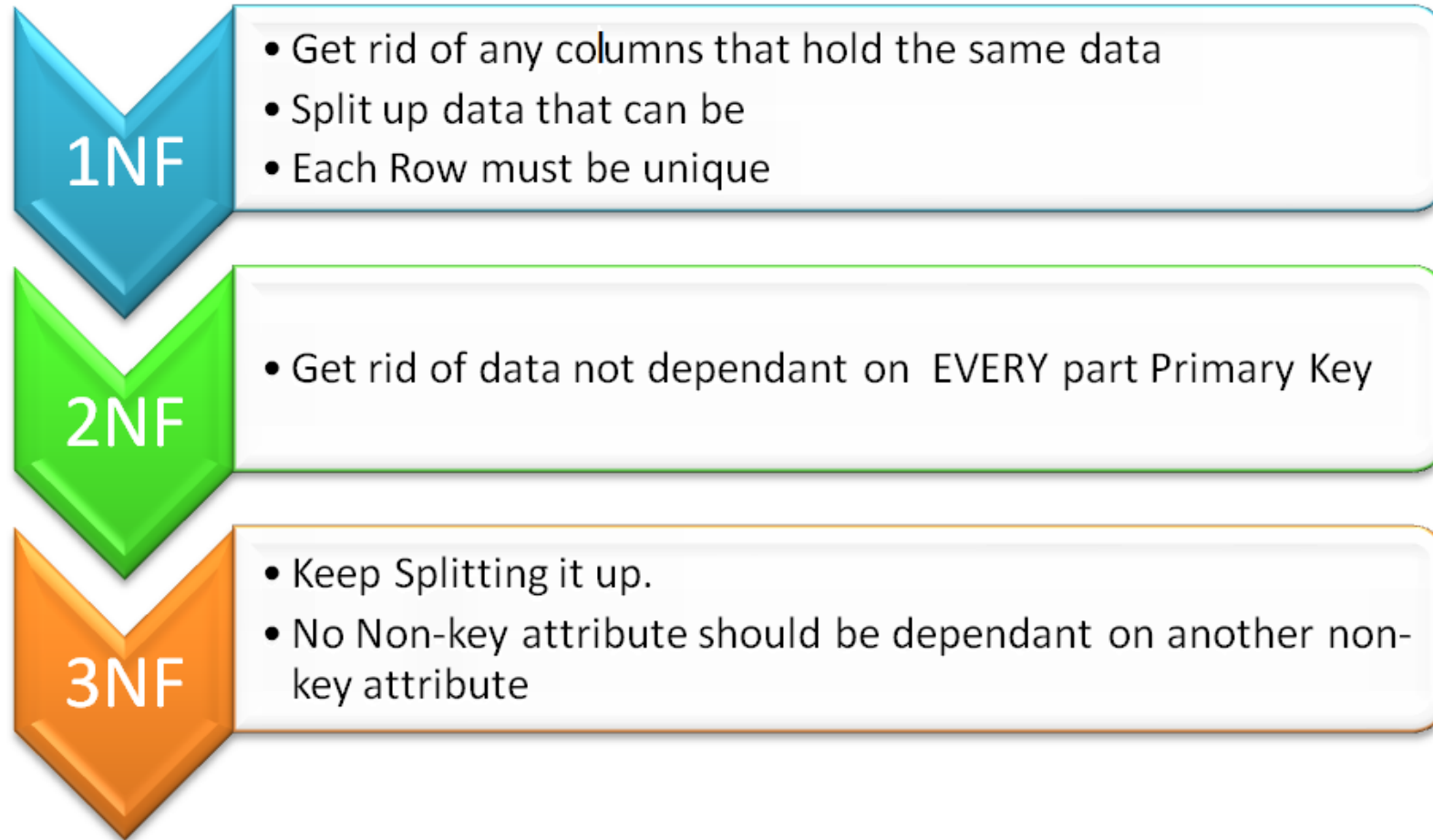
Before:



After:



# Summary of Normalization



# Thank you!