

Introduction to Database Systems

IDBS – Fall 2024

- Week 6:
- Normalization

Eleni Tzirta Zacharatou

Readings
PDBM 6.2

Database Design

How do we design a “good” database schema?

We want to ensure the integrity of the data.

We also want to get good performance.

Example Database

student(sid,cid,room,grade,name,address)

sid	cid	room	grade	name	address
123	15-445	GHC6115	A	Andy	Pittsburgh
456	15-721	GHC8102	B	Tupac	Los Angeles
789	15-445	GHC6115	A	Obama	Chicago
012	15-445	GHC6115	C	Waka Flocka	Atlanta
789	15-721	GHC8102	A	Obama	Chicago

Example Database

student(sid,cid,room,grade,name,address)

sid	cid	room	grade	name	address
123	15-445	GHC6115	A	Andy	Pittsburgh
456	15-721	GHC8102	B	Tupac	Los Angeles
789	15-445	GHC6115	A	Obama	Chicago
012	15-445	GHC6115	C	Waka Flocka	Atlanta
789	15-721	GHC8102	A	Obama	Chicago

Redundancy Problems

Update Anomalies

→ If the room number changes, we need to make sure that we change all student records.

Insert Anomalies

→ May not be possible to add a student unless they are enrolled in a course.

Delete Anomalies

→ If all the students enrolled in a course are deleted, then we lose the room number.

Example Database

student(sid,name,address)

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta

rooms(cid,room)

cid	room
15-415	GHC6115
15-721	GHC8102

courses(sid,cid,grade)

sid	cid	grade
123	15-415	A
456	15-721	B
789	15-415	A
012	15-415	C
789	15-721	A

Why this decomposition is better and how to find it.

Video on LearnIT

All about Functional Dependencies and how to derive them

Functional Dependencies

A **functional dependency** (FD) is a form of a constraint. Part of a relation's schema to define a valid instance.

Definition: $X \rightarrow Y$

→ The value of X functionally defines the value of Y .

Functional Dependencies

Formal Definition:

$\rightarrow X \rightarrow Y \Rightarrow (t_1[x]=t_2[x] \Rightarrow t_1[y]=t_2[y])$

If two tuples (t_1, t_2) agree on the X attribute, then they must agree on the Y attribute too.

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta

X

Y

✓ $sid \rightarrow name$

Functional Dependencies

FD is a constraint that allows instances for which the FD holds.

You can check if an instance violates an FD, but you **cannot** prove that an FD is part of the schema using an instance.

R1(sid,name,address)		
sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta
555	Andy	Providence

??? name → address

Why Should you Care?

FDs seem important, but what can we do with them?

They allow us to decide whether a database design is correct.
→ Note that this is different than whether it's a good idea for performance.

Normal Forms

Identify “bad” functional dependencies

=>

Check for ***normal forms!***

to ensure your relational schema design is free of certain issues
functional dependencies are used to define normal forms

Normal Forms

first normal form (1NF)

second normal form (2NF)

third normal form (3NF)

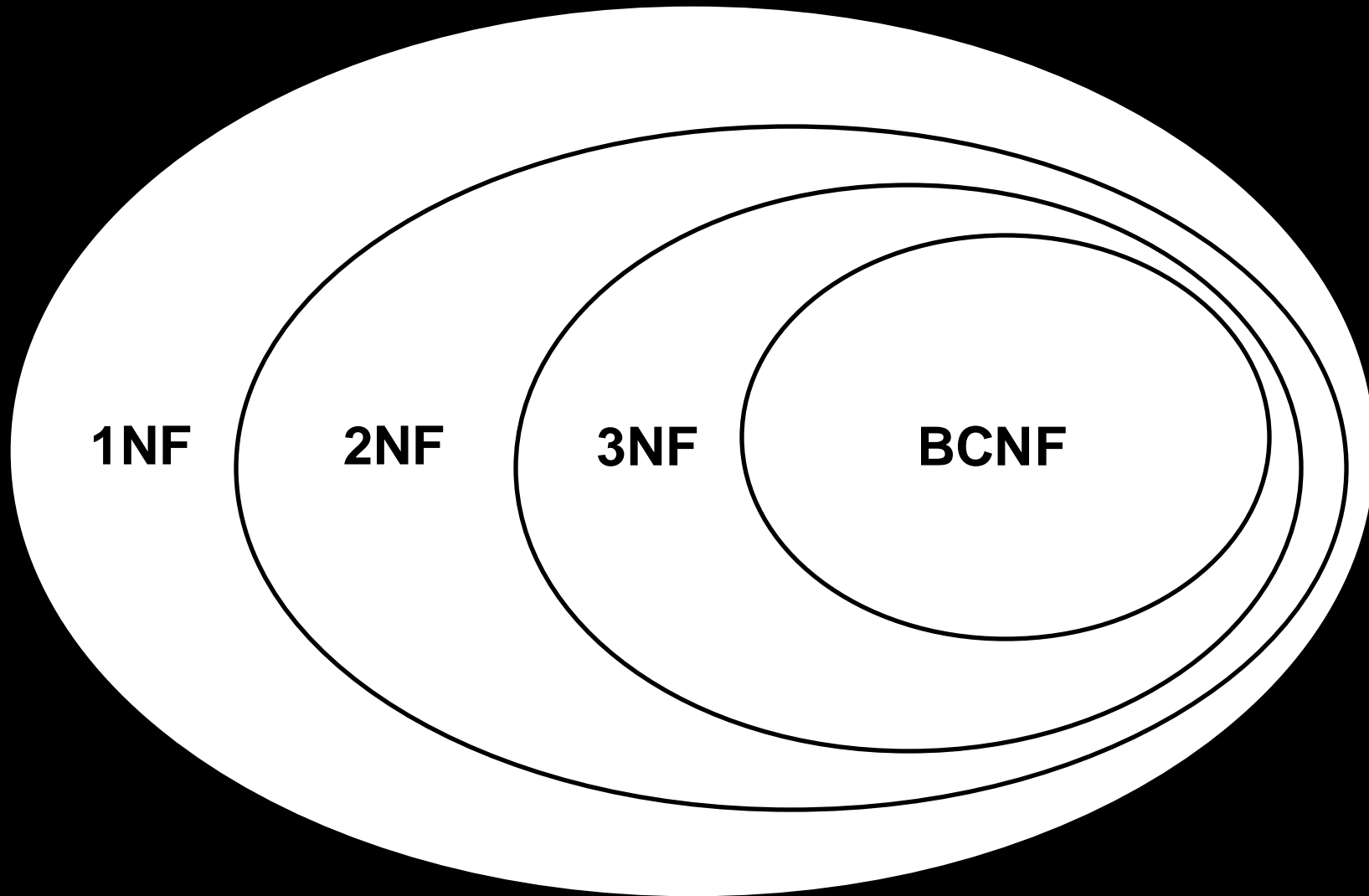
Boyce-Codd normal form (BCNF)

most
common


more restrictive

there are more,
but you don't ever need to know them (i hope)

Normal Forms: Venn Diagram



all attributes are of atomic types
no lists or sets as attributes

id	name	email	office	age	position
108	rory	{rory@itu.dk, lorelai@itu.dk} 	414	35	assoc
118	marvin	marvin@itu.dk	556	35	assist
80	sabina	sabina@itu.dk	112	47	full
128	frodo	frodo@itu.dk	311	30	assoc

2NF

1NF &
non-key attributes *fully (not partially)* depend on the
whole candidate key(s)

if *manufacturer* \rightarrow *country* & key: *manufacturer, model*



manufacturer	model	country	amount
a	v1	italy	500\$
b	v1	denmark	400\$
c	v1	germany	400\$
a	v2	italy	600\$

2NF

1NF &

non-key attributes *fully (not partially)* depend on the whole candidate key(s)

if *manufacturer* \rightarrow *country*

manufacturer	country
a	italy
b	denmark
c	germany

key: *manufacturer*



manufacturer	model	amount
a	v1	500\$
b	v1	400\$
c	v1	400\$
a	v2	600\$

key: *manufacturer, model*

3NF

for all $X \rightarrow A$ in F^+ for a relation R
 $A \subseteq X$ (trivial functional dependency), or
 X contains key for R , or
 A is part of some key for R .

if *position* \rightarrow *salary* (neither attributes are keys)



id	name	office	age	position	salary
108	rory	414	35	assoc	50000
118	marvin	556	35	assist	40000
80	sabina	112	47	full	70000
128	frodo	311	30	assoc	50000

3NF

for all $X \rightarrow A$ in F^+ for a relation R
 $A \subseteq X$ (trivial functional dependency), or
 X contains key for R , or
 A is part of some key for R .

position \rightarrow salary

position	salary
assoc	50000
assist	40000
full	70000



id	name	office	age	position
108	rory	414	35	assoc
118	marvin	556	35	assist
80	sabina	112	47	full
128	frodo	311	30	assoc

BCNF

for all $X \rightarrow A$ in F^+ for a relation R
 $A \subseteq X$ (trivial functional dependency), or
 X contains key for R .

$fID, progID \rightarrow start_date$ (a faculty can start at a program at most once)

$start_date \rightarrow progID$ (on a given day, at most one faculty can start at a program)

candidate keys: $fID, progID$ or $progID, start_date$

works_in

fID	progID	start_date
108	DS	01/01/2015
118	DS	02/02/2017
80	CS	03/03/2017
108	CS	02/03/2017

☹️ **start_date is not a key!**

**BCNF is ideal but
can sometimes be
too restrictive!**

3NF

for all $X \rightarrow A$ in F^+ for a relation R

$A \subseteq X$ (trivial functional dependency), or

X contains key for R, or

A is part of some key for R.

$fID, progID \rightarrow start_date$ (a faculty can start at a program at most once)

$start_date \rightarrow progID$ (on a given day, at most one faculty can start at a program)

candidate keys: $fID, progID$ or $progID, start_date$

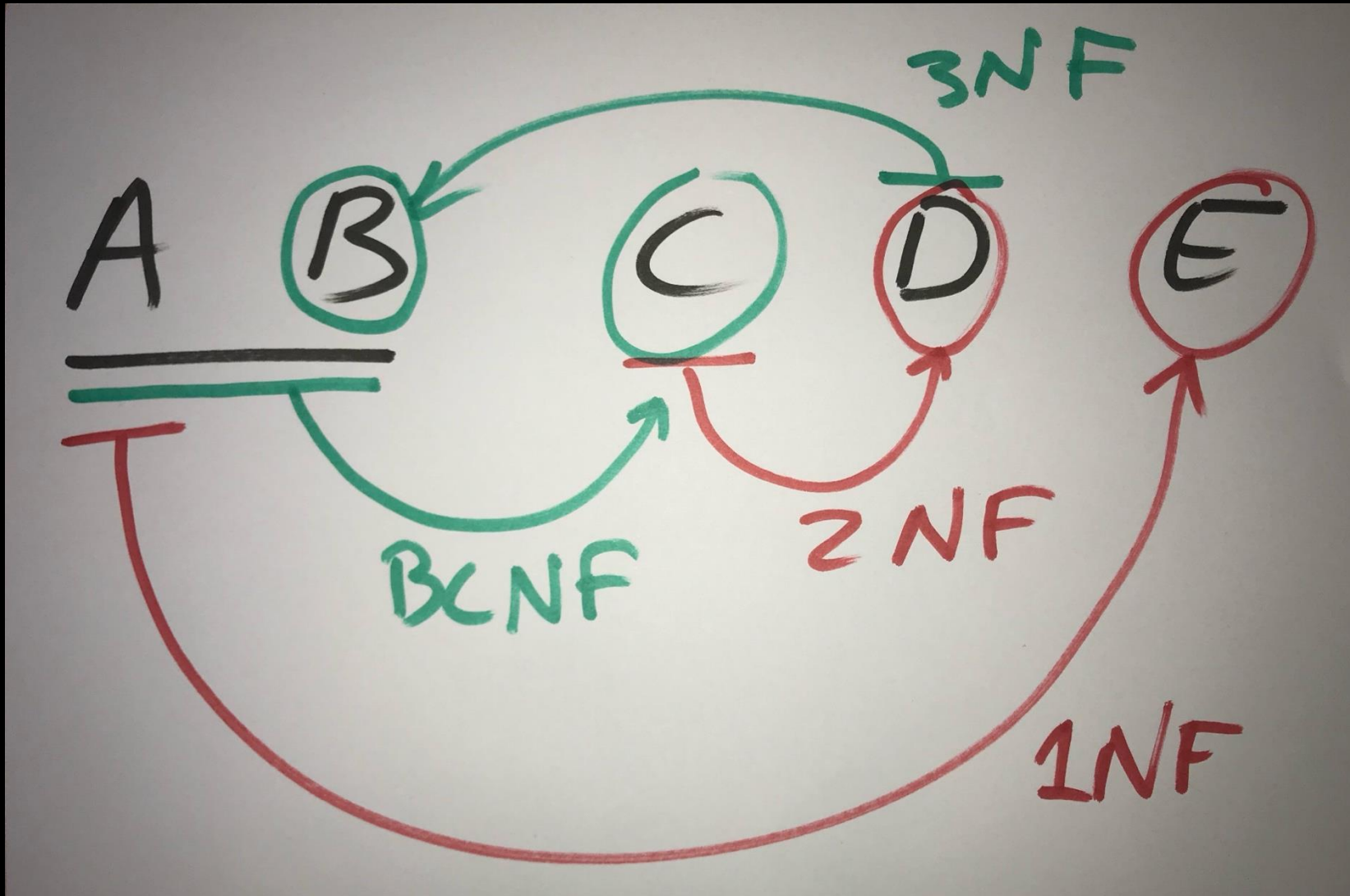
works_in

fID	progID	start_date
108	DS	01/01/2015
118	DS	02/02/2017
80	CS	03/03/2017
108	CS	02/03/2017



3NF allows this!

Visualizing FDs and Normal Forms



Normalization

Look for ***functional dependencies***! (video on LearnIT!)

Identify “bad” functional dependencies that break BCNF/3NF

If there are such, then decompose the tables into two tables

Check again and repeat for sub-tables

When done, the database schema is normalized

Simple Decomposition Algorithm

1. Find all FDs
2. While FD \prec 3NF exists:
Decompose!

Will result in BCNF or 3NF

No way from 3NF to BCNF ...

... except by losing a dependency

... so we don't decompose 3NF!!!

Practical Decomposition

Consider relation R and (important) functional dependency $X \rightarrow Y$ that violates 3NF/BCNF

Decompose R into R1 and R2 where

R1 = R – Y (everything but Y = the right side)

R2 = XY (the whole FD = both left and right side)

This has the following nice properties

R2 is (normally) in BCNF

Joining R1 and R2 (with = on all X attributes) yields R

Example: Person(ID, Name, ZIP, City), $ZIP \rightarrow City$

X = ZIP, Y = City

R-Y = Person(ID, Name, ZIP)

XY = ZIP(ZIP, City)

Example Decomposition I

$R = \underline{A}BCD$

$AB \rightarrow CD$

$C \rightarrow D$

New Table: $\underline{C}D$

Old Table: $\underline{A}BC$

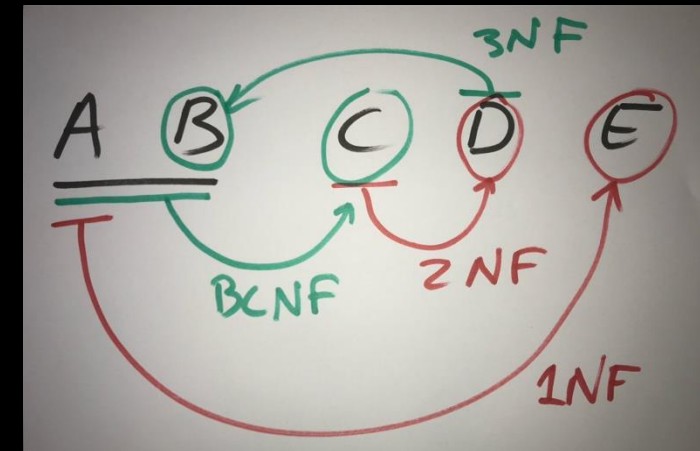
Both in BCNF

What happened to $AB \rightarrow D$?

$X \rightarrow Y$

$R_1 = R - Y$

$R_2 = XY$



Example Decomposition II

$R = \underline{AB}CD \ (AB \rightarrow CD)$

$A \rightarrow D$

A = personID, B = projectID,
C = start date, D = name of person

New Table: \underline{AD}

Old Table: \underline{ABC}

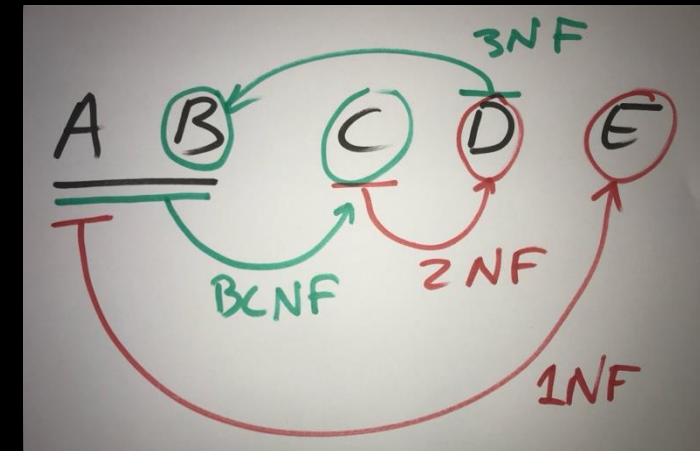
Both in BCNF

What happened to $AB \rightarrow D$?

$X \rightarrow Y$

$R_1 = R - Y$

$R_2 = XY$



Example Decomposition III

$R = \underline{AB}CD$

$A \rightarrow C$

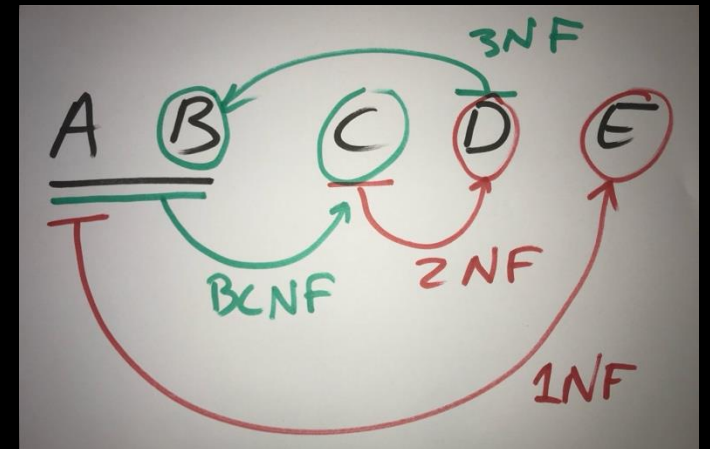
$A \rightarrow D$

New Table: \underline{ACD} // Merge tables with identical keys

Old Table: \underline{AB}

Both in BCNF

$X \rightarrow Y$
 $R1 = R - Y$
 $R2 = XY$



Example Decomposition IV

$R = \underline{A}BCD$

$D \rightarrow A$

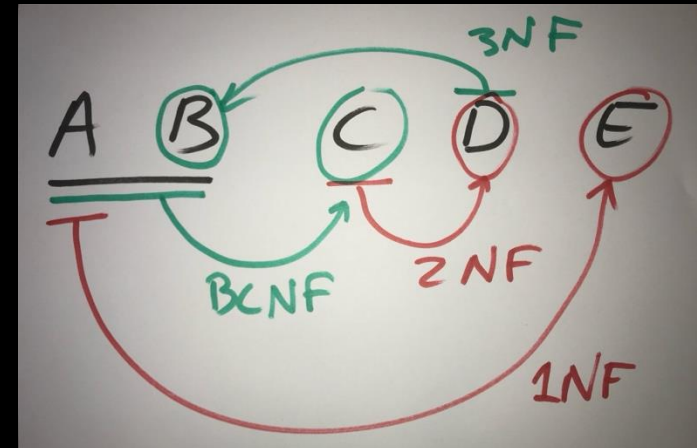
R is in 3NF

No decomposition into BCNF

$X \rightarrow Y$

$R_1 = R - Y$

$R_2 = XY$



Example Decomposition V

$R = \underline{A}BCD$

$A \rightarrow C, C \rightarrow D$

Which one do we use first? ... and why?

New Table: $\underline{C}D$

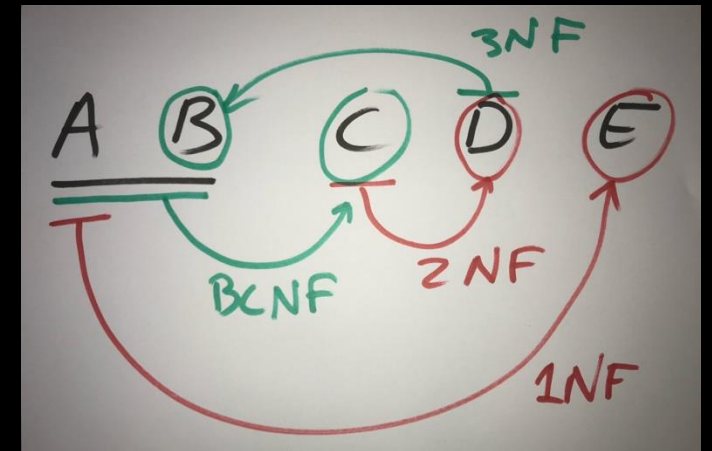
New Table: $\underline{A}C$

Old Table: $\underline{A}B$

All three in BCNF

What happened to $AB \rightarrow D$

$X \rightarrow Y$
 $R_1 = R - Y$
 $R_2 = XY$



Example Normalization

Advices(Teacher, Student, Dept, Program, Course)

Teacher \rightarrow Dept

Course \rightarrow Program

$X \rightarrow Y$

$R1 = R - Y$

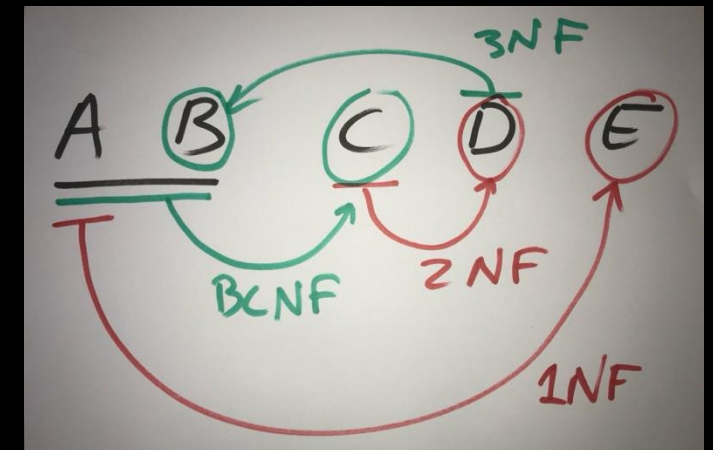
$R2 = XY$

Outcome:

Faculty(Teacher, Dept)

Catalog(Course, Program)

Advices(Teacher, Student, Dept, Program, Course)



The Inconvenient Truth


or why many people don't like **normalization**

most NoSQL systems denormalize instead of decomposing a relation into two!

id	name	office	age	position	salary
108	rory	414	35	assoc	50000
118	marvin	556	35	assist	40000
80	sabina	112	47	full	70000
128	frodo	311	30	assoc	50000

joining two tables (especially in a distributed environment) is expensive!!

Takeaways

- 
- **Schema Refinement**
 - **Functional Dependencies**
 - Unavoidable, trivial, and redundant FDs
 - **Normal forms are important**
 - they form the foundation for schema refinement
 - always decompose 1NF and 2NF
 - but keep in mind you might have to take more things into account
 - **Normalization Process**
 - decomposition of relations to BCNF (or 3NF)