

# Chapter 3 Design Theory for Relational Databases

---

# Objectives

---

Understand concepts of:

- Functional Dependencies
- Normalization
- Decomposition
- Multi-valued Dependencies

# Contents

---

- Functional Dependencies
- Rules about FDs
- Key & Super-Key
- Normal forms

# Functional dependency

- A **functional dependency**: constraint between two sets of attributes in a relation
- A set of attributes  $X$  (include  $A_1A_2...A_n$ ) in  $R$  **functionally determine** another attribute  $Y$  (include  $B_1B_2...B_m$ ), also in  $R$ , (written  $X \rightarrow Y$ ) if and only if each  $X$  value is associated with precisely one  $Y$  value
- A functional dependency  $A_1A_2...A_n \rightarrow B_1B_2...B_m$  holds on relation  $R$  if *two tuples of  $R$  agree on all of the attributes  $A_1, A_2, ..., A_n$  then they must also agree on all of the attributes  $B_1, B_2, ..., B_m$*

# Functional dependency

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>	<i>starName</i>
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

© Coursera.org

Easy to see that: the following FD is true

- $\text{title, year} \rightarrow \text{length, genre, studioName}$

Exercise: How about the FD

- $\text{title, year} \rightarrow \text{starName}$

$\text{title, year} \rightarrow \text{starName}$  does not hold in Movies1 relation

# Functional dependency

---

- Review key of relation, candidate keys(alternate keys), primary key
- Super-key
  - A set of attributes that contains a key is called a *super-key*
  - Every super-key satisfies the first condition of a key: it functionally determines all other attributes of the relation
  - If K is a key, L is a super key, then:  $K \subseteq L$
  - A key is also a super key

# Functional dependency

## ■ Armstrong's Axioms

### ■ Fundamental Rules: Let $X$ , $Y$ , $Z$ are sets of attributes

#### ■ Reflexivity:

If  $X$  is a subset of  $Y$ , then  $Y \rightarrow X$

#### ■ Augmentation

If  $X \rightarrow Y$ , then  $XZ \rightarrow YZ$  for any  $Z$

#### ■ Transitivity

If  $X \rightarrow Y$  and  $Y \rightarrow Z$ , then  $X \rightarrow Z$

### ■ Additional rules: Let $X$ , $Y$ , $Z$ , $W$ are sets of attributes

■ Union/Combining: if  $X \rightarrow Y$  AND  $X \rightarrow Z$  then  $X \rightarrow YZ$

■ Decomposition/Splitting: if  $X \rightarrow YZ$ , then  $X \rightarrow Y$  and  $X \rightarrow Z$

■ Pseudotransitivity: If  $X \rightarrow Y$  and  $WY \rightarrow Z$  then  $WX \rightarrow Z$

### ■ Trivial FDs: right side is a subset of left side

■ Ex:  $FLD \rightarrow FD$

# Functional dependency

---

- A set of FD's  $S$  **follows** from a set of FD's  $T$  if every relation instance that satisfies all the FD's in  $T$  also satisfies all the FD's in  $S$
- Two sets of FD's  $S$  and  $T$  are **equivalent** if and only if  $S$  **follows** from  $T$ , and  $T$  **follows**  $S$



# The Closure of Attributes

- The closure of a set of attributes  $\{A_1, A_2, \dots, A_n\}$  under FD's in  $S$  (denoted  $\{A_1, A_2, \dots, A_n\}^+$ ) is the set of attributes  $B$  such that every relation that satisfies all the FD's in set  $S$  also satisfies  $A_1A_2\dots A_n \rightarrow B$
- That is,  $A_1A_2\dots A_n \rightarrow B$  follows from the FD's of  $S$
- $A_1, A_2, \dots, A_n \in \{A_1, A_2, \dots, A_n\}^+$ , because  $A_1A_2\dots A_n \rightarrow A_i$  is trivial

# The Closure of Attributes

## Algorithm 3.7: Closure of a set of attributes

- Input: A set of attributes  $\{A_1, A_2, \dots, A_n\}$  and a set of FD's  $S$
- Output: The closure  $\{A_1, A_2, \dots, A_n\}^+$ 
  1. If necessary, split the FD's of  $S$ , so each FD in  $S$  have singleton right side
  2. Let  $X$  be a set of attributes that will become the closure. Initialize  $X$  to be  $\{A_1, A_2, \dots, A_n\}$
  3. Repeatedly search for some FD:  $B_1 B_2 \dots B_m \rightarrow C$ , such that  $B_1, B_2, \dots, B_m$  are in  $X$ , but  $C$  is not
    - a) If such  $C$  is found, add to  $X$ , and repeat the search
    - b) If such  $C$  is not found, no more attributes can be added to  $X$
  4. The set  $X$  is the correct value of  $\{A_1, A_2, \dots, A_n\}^+$

# The Closure of Attributes

---

$R(A, B, C, D)$

$S = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Compute  $\{A\}^+ ? \{B\}^+ ?$

What are some the keys of  $R$ ?

# Closing Sets of Functional Dependencies

Suppose a set of FD's  $S$ , any set of FD's  $T$  equivalent to  $S$  is said to be a **basis** for  $S$ .

Then we say  $T$  is a **basis** for  $S$

Just work with only FD's that have *singleton right sides*

A **minimal basis** for FD's  $S$  is a **basis**  $B$  that satisfies three conditions:

- All the FD's in  $B$  have singleton right sides
- If any FD is removed from  $B$ , the result is no longer a basis
- If for any FD in  $B$  we remove one or more attributes from the left side, the result is no longer a basis

# Closing Sets of Functional Dependencies

---

## Example

- $R(A, B, C)$
- $S = \{A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B, AB \rightarrow C, BC \rightarrow A, AC \rightarrow B, A \rightarrow BC, B \rightarrow AC, C \rightarrow AB\}$
- $R$  and its FD's have several minimal basis
  - $\{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}$ , or
  - $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

# What happens to ...

---

... a set of FD's  $S$  of  $R$  when we project  $R$  on some attributes?

That is, suppose a relation  $R$  with set of FD's  $S$ , and  $R_1 = \pi_L(R)$ . What FD's hold in  $R_1$ ?

# Projecting Functional Dependencies

---

To find a functional dependencies of projection, we

- Follow from  $S$ , and
- Involve only attributes of  $R_1$

# Projecting Functional Dependencies

---

## Algorithm 3.12: Projecting a Set of FD's

- Input:  $R$ ,  $R_1 = \pi_L(R)$ ,  $S$  a set of FD's that hold in  $R$
- Output: the set of FD's that hold in  $R_1$
- Method:
  - $T$  is the set of FD's that hold in  $R_1$ . Initially,  $T$  is empty
  - For each set of attributes  $X$  of  $R_1$ , compute  $X^+$ . Add to  $T$  all non-trivial FD's  $X \rightarrow A$  such that  $A$  is both in  $X^+$  and an attribute of  $R_1$
  - Construct a minimal basis from  $T$



# Projecting Functional Dependencies

---

## Algorithm 3.12: Projecting a Set of FD's (cont)

- Compute a minimal basis from  $T$ 
  - If there is an FD  $F$  in  $T$  that follows from other FD's in  $T$ , then remove  $F$  from  $T$
  - Let  $Y \rightarrow B$  is a FD in  $T$ , with at least two attributes in  $Y$ , and let  $Z$  is  $Y$  with one of its attributes removed:
    - If  $Z \rightarrow B$  follows from the other FD's in  $T$  (including  $Y \rightarrow B$ ), then replace  $Y \rightarrow B$  by  $Z \rightarrow B$
  - Repeat the above steps in all possible ways until no more changes to  $T$  can be made

# Projecting Functional Dependencies

---

## Two notations

- (1) Closing the empty set and the set of all attributes cannot yield a nontrivial FD
- (2) If we have already knew that the closure of some set  $X$  is all attributes, then we cannot discover any new FD's by closing supersets of  $X$

# Projecting Functional Dependencies

Example: Suppose  $R(A,B,C,D)$  has FD's  $A \rightarrow B$ ,  $B \rightarrow C$ , and  $C \rightarrow D$ .  $R1 = \pi_{A,C,D}(R)$ . Find the FD's of  $R1$ ?

- Compute the closure of the singleton set
  - $\{A\}^+ = \{A,B,C,D\}$ , and B is not in  $R1$ , then new FD's  $A \rightarrow C$ ,  $A \rightarrow D$
  - $\{C\}^+ = \{C,D\}$ , then new FD's  $C \rightarrow D$
  - $\{D\}^+ = \{D\}$ , no new FD's
- Compute the closure of the doubleton set
  - Since  $\{A\}^+$  include all attributes, no care any more for supersets of  $\{A\}$
  - $\{C,D\}^+ = \{C,D\}$ , no new FD's holds in  $R1$
- Finally, there are three FD's  $A \rightarrow C$ ,  $A \rightarrow D$ ,  $C \rightarrow D$  hold in  $R1$
- $A \rightarrow D$  is transitive from  $A \rightarrow C$ , and  $C \rightarrow D$
- So, minimal basis is  $\{A \rightarrow C, C \rightarrow D\}$

# Anomalies introduction

---

Careless selection of a relational database schema can lead to redundancy and related anomalies

So, in this session we shall tackle the problems of relational database designing

Problems such as redundancy that occur when we try to cram too much into a single relation are called *anomalies*

# Anomalies

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>	<i>starName</i>
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With The Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

The principal kinds of anomalies that we encounter are:

**Redundancy:** information maybe repeated unnecessarily in several tuples (exp: the length and genre)

**Update Anomalies:** We may change information in one tuple but leave the same information unchanged in another (exp: if we found that *Star Wars* is 125 minutes long, we may change the length in the first tuple but not in the second and third tuples)

**Deletion Anomalies:** If a set of values becomes empty, we may lose other information as a side effect (exp: if we delete “Fox” from the set of studios, then we have no more studios for the movie “Star Wars”)

# Decomposition

---

The accepted way to eliminate anomalies is the *decomposition* of relations

Decomposition of a relation  $R$  involves splitting the attributes of  $R$  to make the schemas of 2 new relations

**Definition:** Given a relation  $R(A_1, \dots, A_n)$ , we say  $R$  is decomposed into  $S(B_1, \dots, B_m)$  and  $T(C_1, \dots, C_k)$  if:

- +  $\{A_1, \dots, A_n\} = \{B_1, \dots, B_m\} \cup \{C_1, \dots, C_k\}$
- +  $S = \Pi_{B_1, \dots, B_m}(R)$
- +  $T = \Pi_{C_1, \dots, C_k}(R)$

# Example: Decomposition

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>	<i>starName</i>
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With The Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers



<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>
Star Wars	1977	124	SciFi	Fox
Gone With The Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount



<i>title</i>	<i>year</i>	<i>starName</i>
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With	1939	Vivien Leigh
Wayne's W	1992	Dana Carvey
Wayne's W	1992	Mike Meyers

# Discuss

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>
Star Wars	1977	124	SciFi	Fox
Gone With The Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

<i>title</i>	<i>year</i>	<i>starName</i>
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With	1939	Vivien Leigh
Wayne's W	1992	Dana Carvey
Wayne's W	1992	Mike Meyers

The redundancy is eliminated (the length of each film appears only once)

The risk of an update anomaly is gone (we only have to change the length of *Star Wars* in one tuple)

The risk of a deletion anomaly is gone (if we delete all the stars for *Gone with the wind*, that deletion makes the movie disappear from the right but still be found in the left)



# Decomposition: The Good, Bad and Ugly

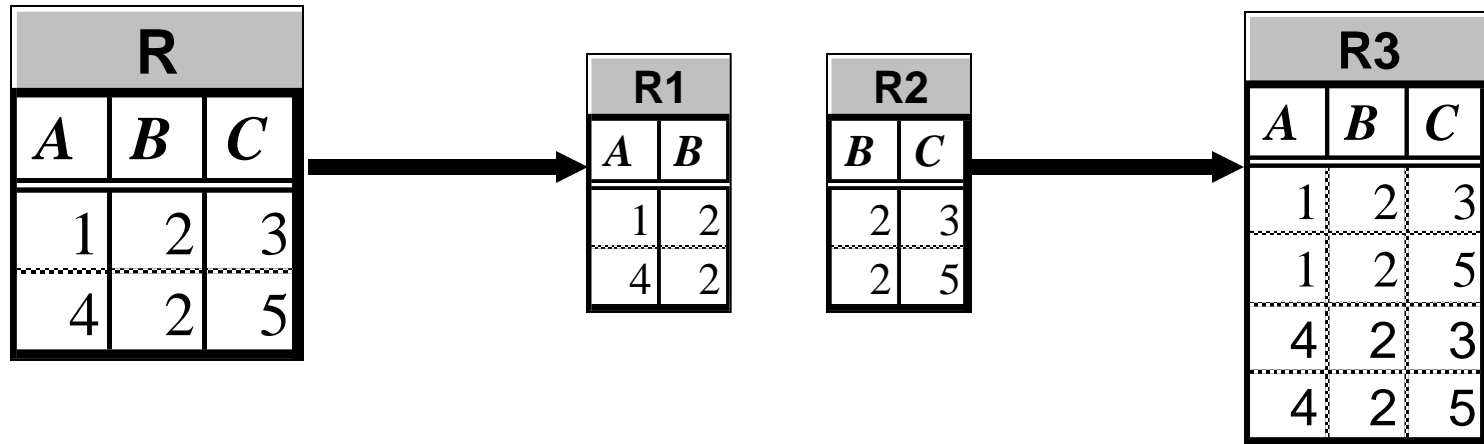
---

We observed that before we decompose a relation schema into BCNF, it can exhibit anomalies; That's the "Good"

However, decomposition can also have some bad:

- Maybe we can't recovery the original information; OR
- After reconstruction, the FDs maybe not hold

# Example: Loss of information after decomposition



Suppose we have  $R(A,B,C)$  but neither of the FD's  $B \rightarrow A$  nor  $B \rightarrow C$  holds.

R is decomposed into R1 and R2 as above

When we try to re-construct R by Natural Join of R1 and R2, we have:  $R3 = R1 \bowtie R2$  (but  $R3 \neq R$   $\Rightarrow$  We lost information)

# Example: Dependency Loss

---

If we check the projected FD's in the relations of the decomposition, can we can be sure that when we reconstruct the original relation from the decomposition by joining, the result will satisfy the original FD's?

# Normal Forms

---

- ☐ First Normal Form
- ☐ Second Normal Form
- ☐ Third Normal Form
- ☐ Boyce-Codd Normal Form
- ☐ Fourth Normal Form
- ☐ Fifth Normal Form
- ☐ Domain-Key Normal Form

# Dependencies: Definitions

---

- ***Multivalued Attributes*** (or ***repeating groups***): non-key attributes or groups of non-key attributes the values of which are not uniquely identified by (directly or indirectly) (not functionally dependent on) the value of the Primary Key (or its part).
- ***Partial Dependency*** – when a non-key attribute is determined by a part, but not the whole, of a **COMPOSITE** primary key.
- ***Transitive Dependency*** – when a non-key attribute determines another non-key attribute.

# 1NF

1NF A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only

Take the following table.

*StudentID is the primary key.*

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	<u>10 Charles Street</u>	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

Is it 1NF?

**No. There are repeating groups (subject, subjectcost, grade)**

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	<u>10 Charles Street</u>	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

How can you make it 1NF?

## Create new rows so each cell contains only one value

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+



StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

But now look – is the *studentID* primary key still valid?



# No – the studentID no longer uniquely identifies each row

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

You now need to declare *studentID* and *subject* together to uniquely identify each row.

So the new key is StudentID *and* Subject.

---

# So. We now have 1NF.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

Is it 2NF?

# 2NF

A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key

**StudentName & Address are dependent on studentID (which is part of the key)**

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

But they are not dependent on *Subject* (the *other* part of the key)

## And 2NF requires...

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

All non-key fields are dependent on the ENTIRE key (studentID + subject)

---

So it's not 2NF

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

How can we fix it?

# Make new tables

---

Make a new table for each primary key field

Give each new table its own primary key

Move columns from the original table to the new table that matches their primary key...

# Step 1

---

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

# Step 2

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)



# Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

# Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

# Step 4 - relationships

STUDENT TABLE (key = StudentID)

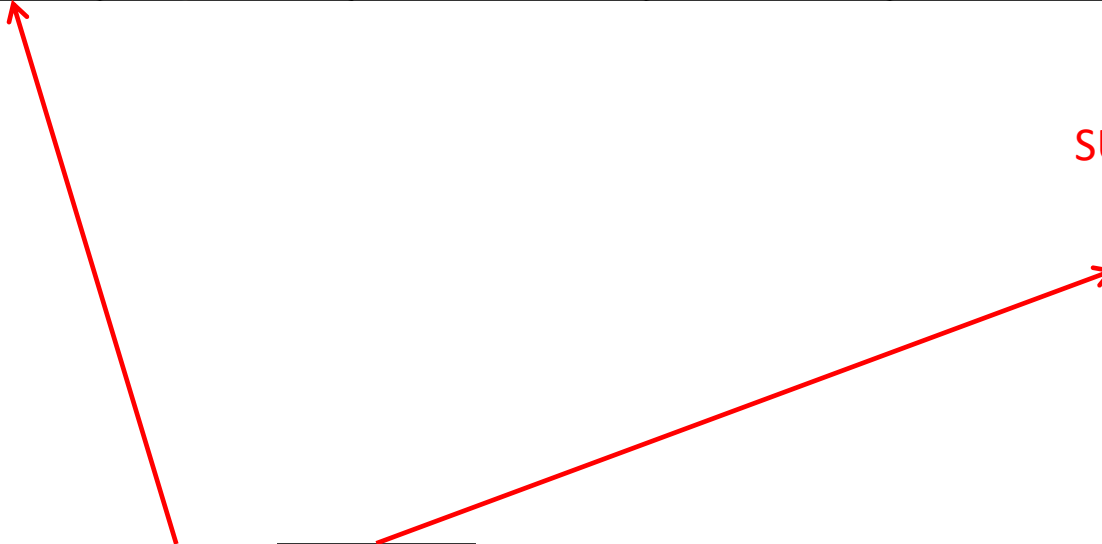
StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Each student can only appear ONCE in the student table

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

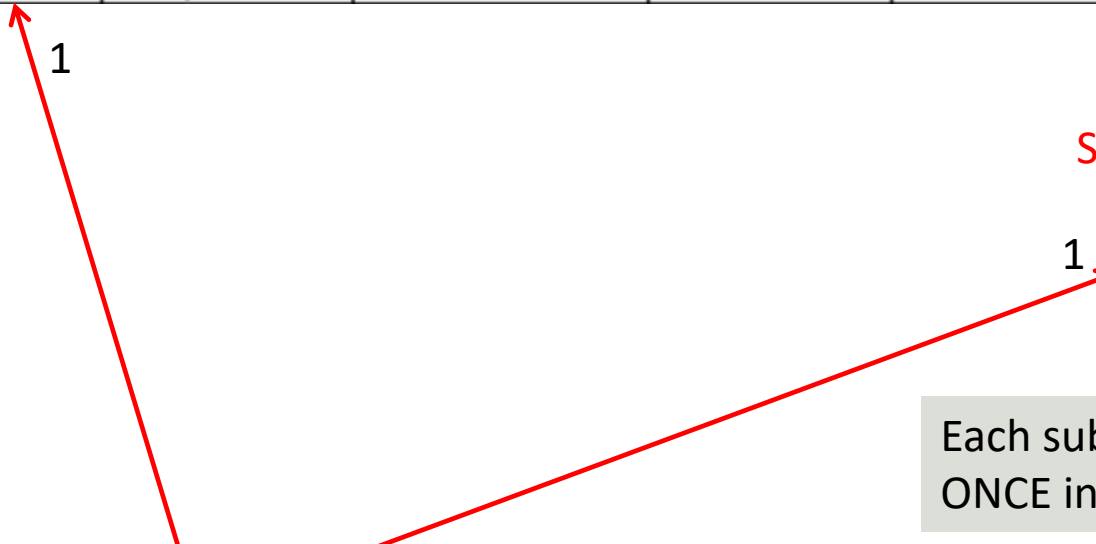
SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

Each subject can only appear  
ONCE in the subjects table

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

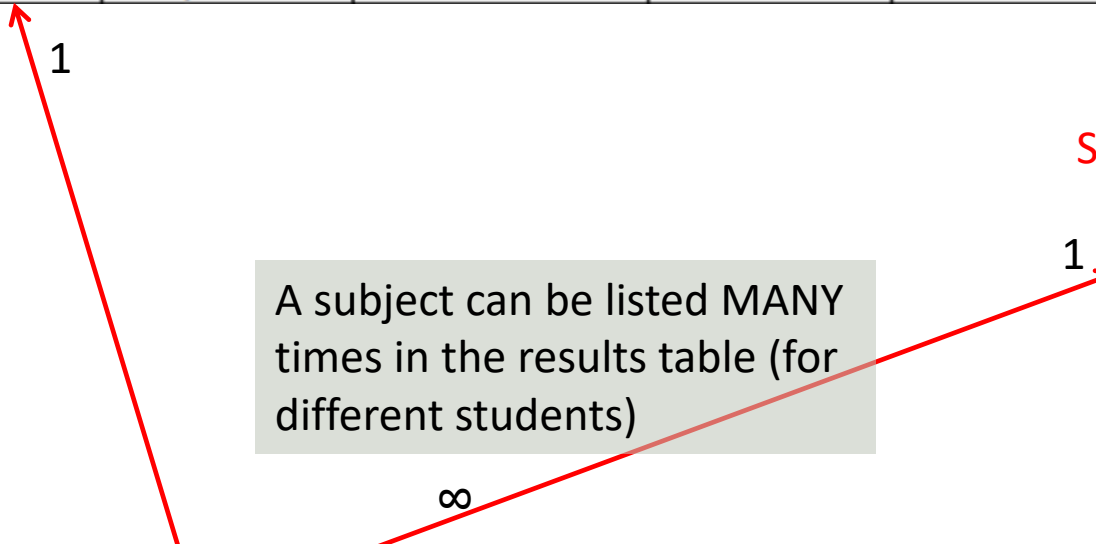
SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

A subject can be listed MANY times in the results table (for different students)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

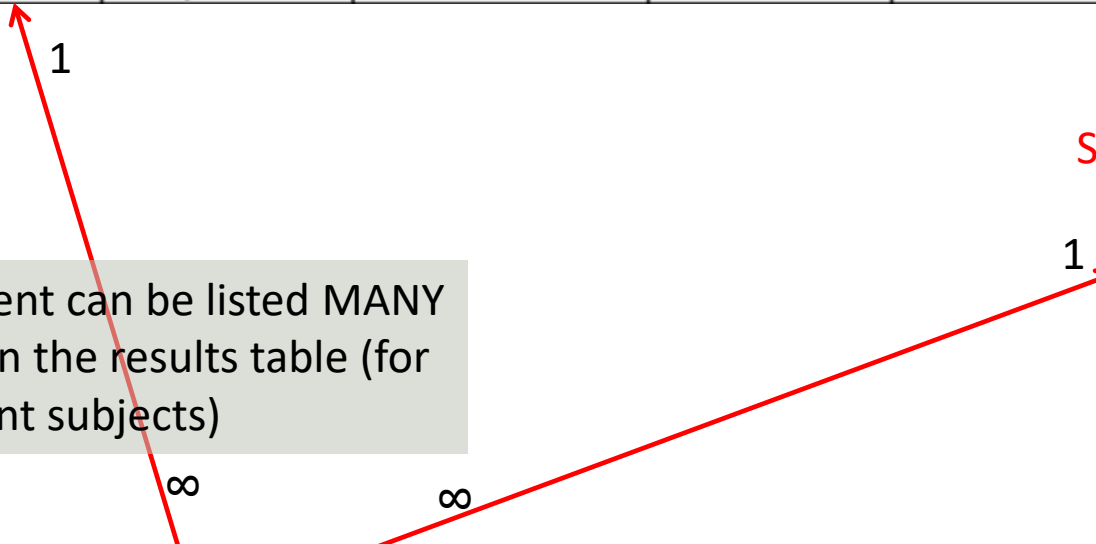
SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

A student can be listed MANY times in the results table (for different subjects)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

SubjectCost is only dependent on the primary key, *Subject*



RESULTS TABLE (key = StudentID+Subject)



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

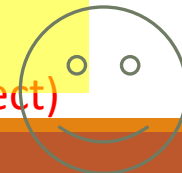
SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

Grade is only dependent  
on the primary key  
(*studentID + subject*)

RESULTS TABLE (key = StudentID+Subject)



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Name, Address are only dependent on the primary key (*StudentID*)



SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

1

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

So it is  
2NF!

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

But is it  
3NF?

# 3NF

**A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key.**

An attribute C is transitively dependent on attribute A if there exists an attribute B such that:  $A \rightarrow B$  and  $B \rightarrow C$

**Note that** 3NF is concerned with transitive dependencies which do not involve candidate keys. A relation with more than one candidate key will clearly have transitive dependencies of the form:  $\text{primary\_key} \rightarrow \text{other\_candidate\_key} \rightarrow \text{any\_non-key\_column}$

# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

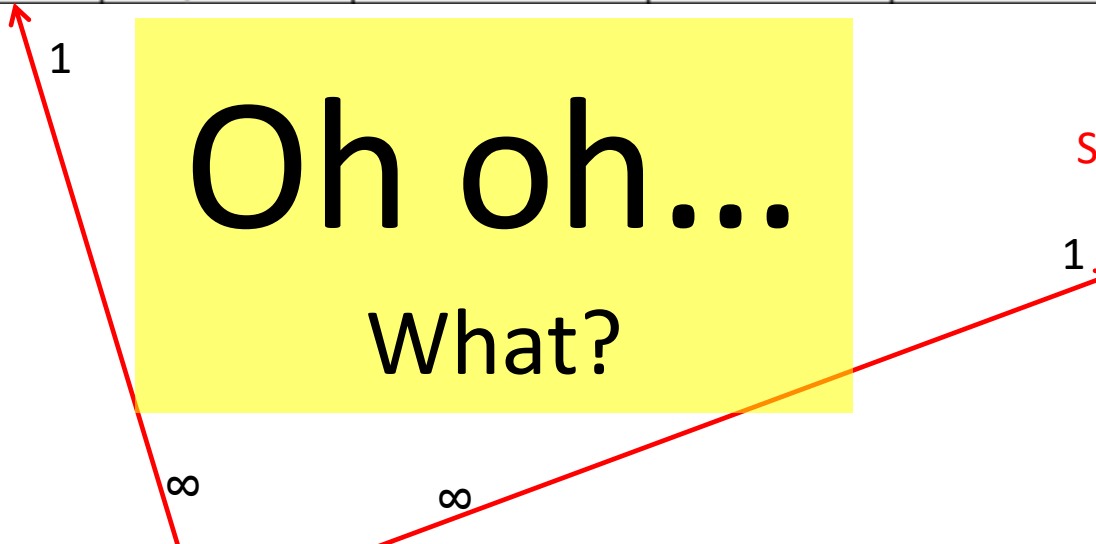
Oh oh...  
What?

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

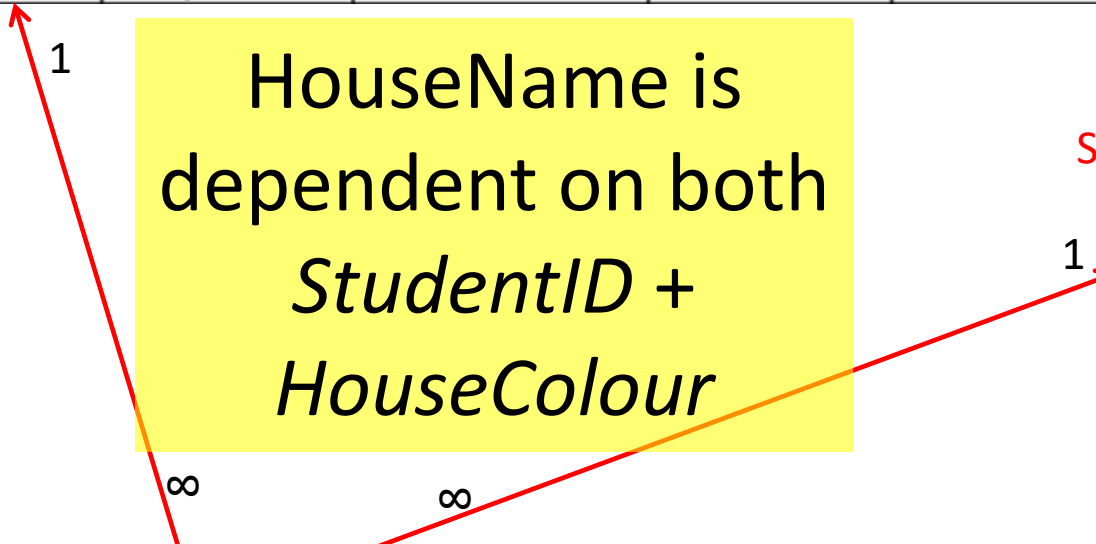
HouseName is  
dependent on both  
*StudentID* +  
*HouseColour*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

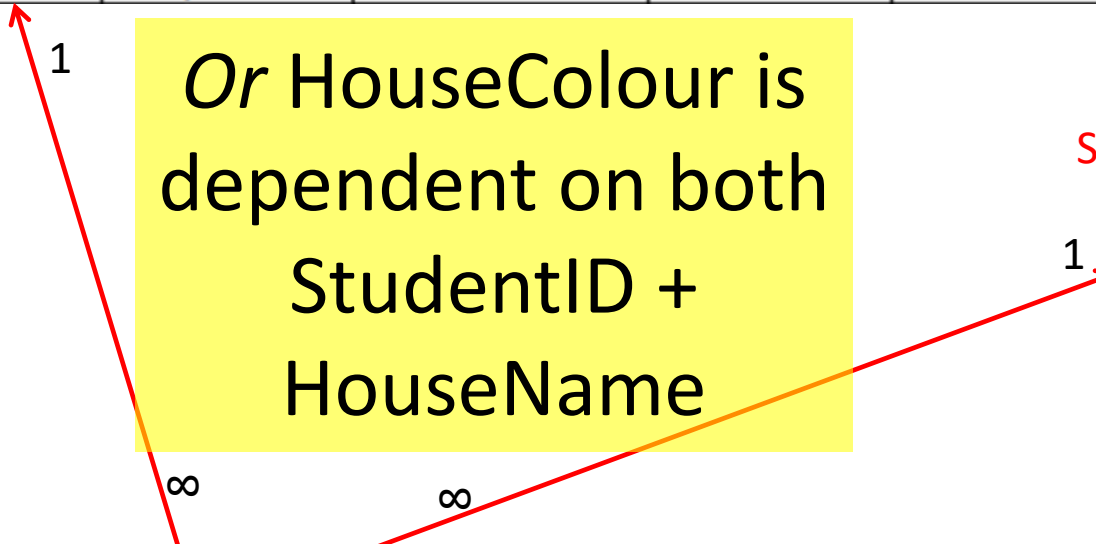
Or HouseColour is  
dependent on both  
StudentID +  
HouseName

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

*But either way,  
non-key fields are  
dependent on MORE  
THAN THE PRIMARY  
KEY (studentID)*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

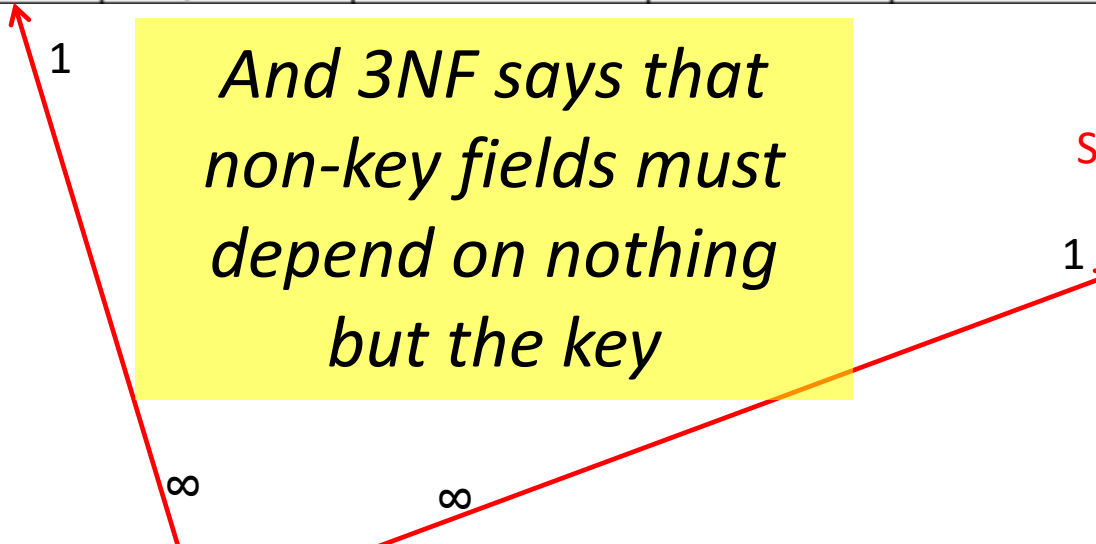
*And 3NF says that  
non-key fields must  
depend on nothing  
but the key*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

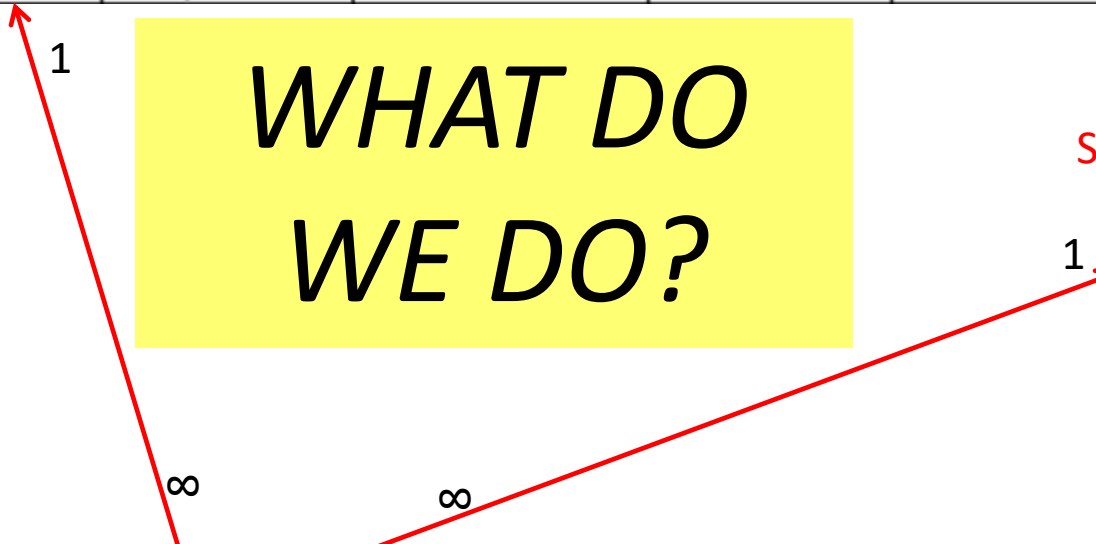
*WHAT DO  
WE DO?*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# Again, carve off the offending fields

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

1

∞

∞

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

# A 3NF fix

StudentTable

StudentID	StudentName	Address
19594332X	Mary Watson	10 Charles Street

Primary key: StudentID

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

1

∞

∞

1

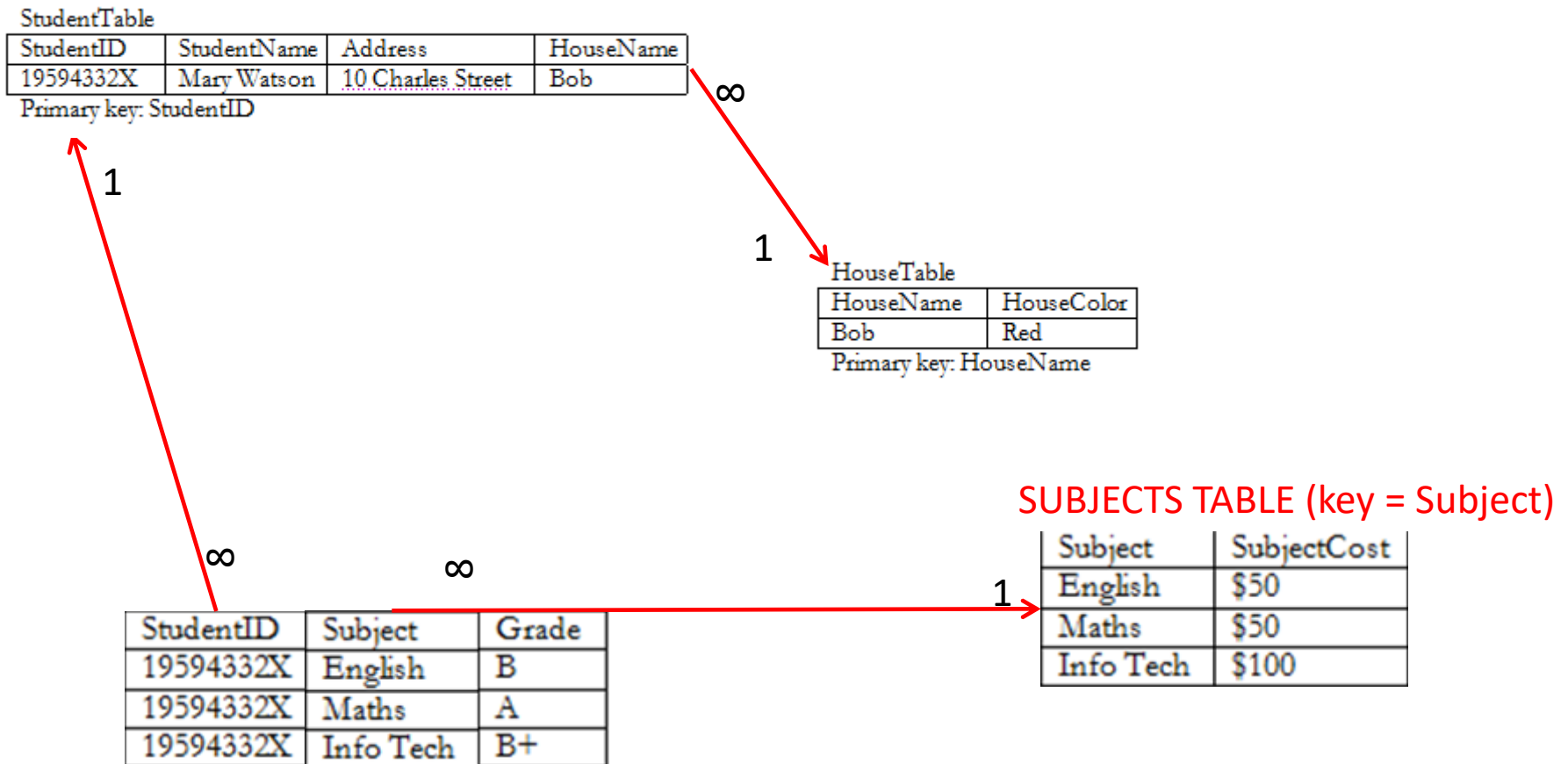
StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

SUBJECTS TABLE (key = Subject)

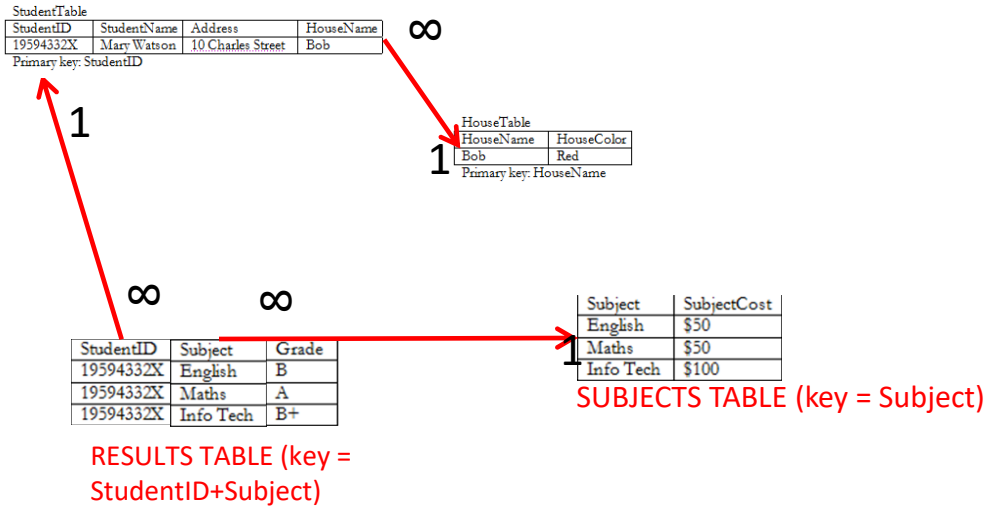
Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

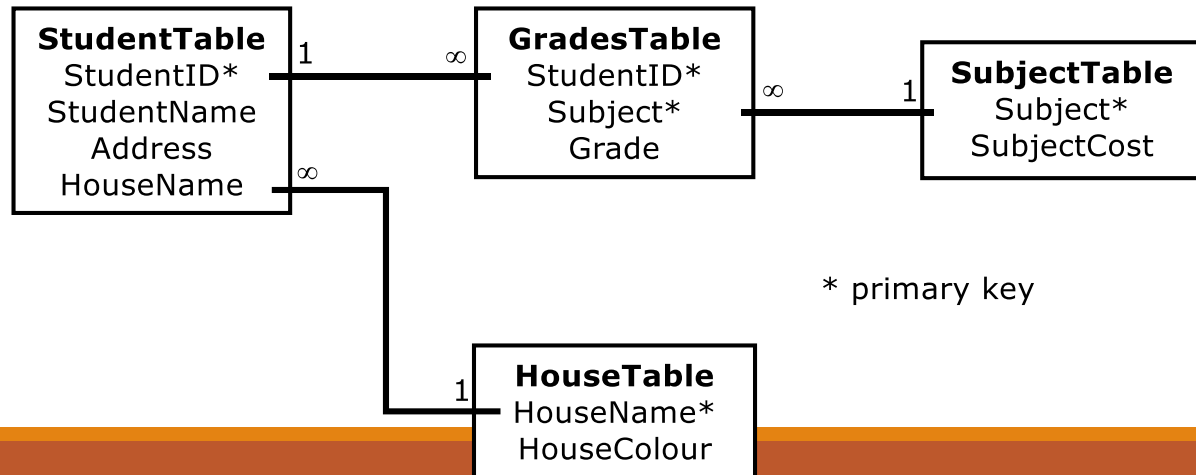
# A 3NF fix



# A 3NF win!



Or...



# The Reveal

Before...

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

After...

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

1

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

∞

1

∞

∞

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

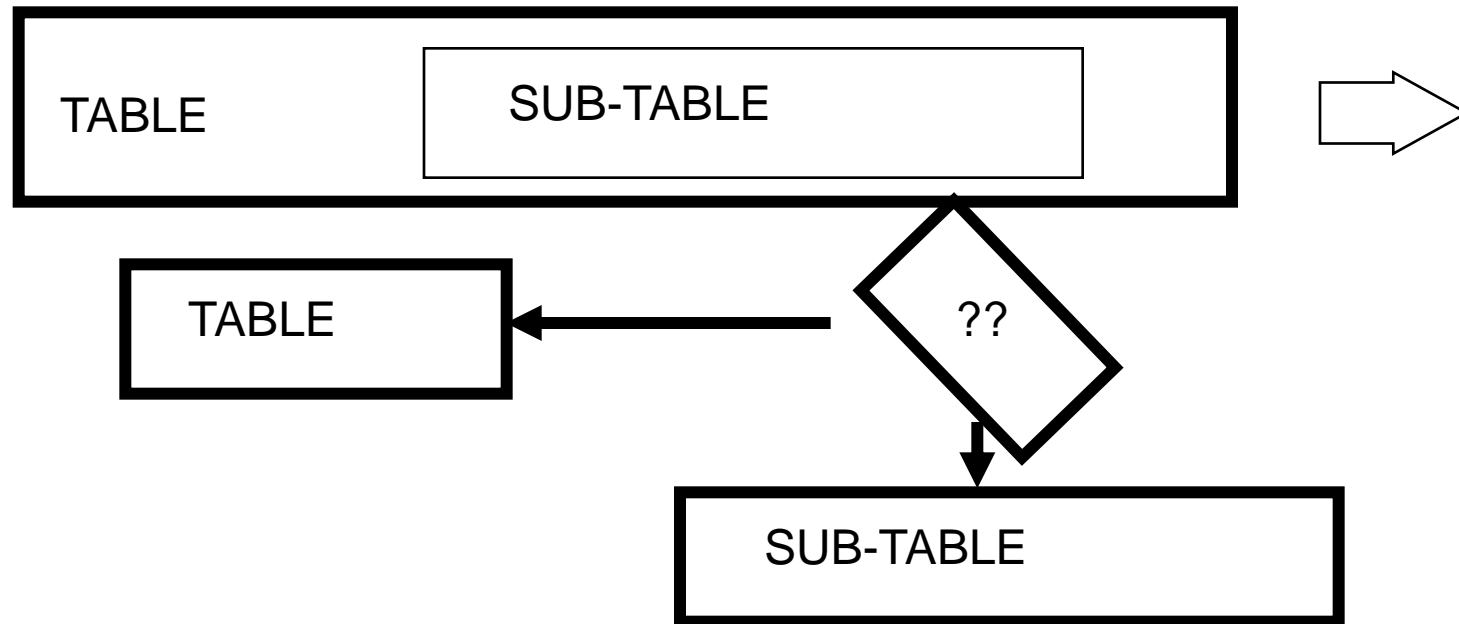
SUBJECTS TABLE (key = Subject)

# 3NF

## No transitive dependencies

---

Table contains data from an embedded entity with non-key attributes.



BCNF is the same, but the embedded table may involve key attributes.



# BCNF

---

A relation R is in BCNF if and only if: **Whenever there is a Non-Trivial FD**

**$A_1A_2..A_n \rightarrow B_1B_2..B_m$  for R, it is the case that:  
 $\{A_1,..,A_n\}$  is a super-key for R**

That is: the left side of every Non-Trivial FD must be a super-key

## Example: BCNF or not

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>	<i>starName</i>
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With The Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

The above relation is not in BCNF because:  
Consider the FD:

$\{title, year\} \rightarrow \{length, genre, studioName\}$

We know that  $\{title, year\}$  is not a super-key

## Example: BCNF or not

<i>title</i>	<i>year</i>	<i>length</i>	<i>genre</i>	<i>studioName</i>
Star Wars	1977	124	SciFi	Fox
Gone With The Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

The above relation is in BCNF because:  
 $\{title, year\} \rightarrow \{length, genre, studioName\}$

And: neither title nor year by itself functionally determines any of the other attributes

## Example: BCNF or not

<i>title</i>	<i>year</i>	<i>starName</i>
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With	1939	Vivien Leigh
Wayne's W	1992	Dana Carvey
Wayne's W	1992	Mike Meyers

The above relation is in BCNF because: it has no Non-Trivial FD

# Differences between BCNF and 3NF

3NF	Boyce-Codd
<p>a <i>nontrivial</i> functional dependency: <math>X \Rightarrow A</math> holds in <math>R</math>, either</p> <p>(a) <math>X</math> is a superkey of <math>R</math>, or</p> <p>(b) <math>A</math> is a prime attribute of <math>R</math>.</p>	<p>a <i>nontrivial</i> functional dependency <math>X \Rightarrow A</math> holds in <math>R</math>, then:</p> <p>a) <math>X</math> is a superkey of <math>R</math></p>
<p><b><u>Note</u>:</b> A functional dependency <math>X \Rightarrow Y</math> is a <b>full functional dependency</b> if removal of any attribute <math>A</math> from <math>X</math> means that the dependency does not hold any more; A <b><u>partial functional dependency</u></b> is not a <b><u>full functional dependency</u></b></p>	

# BCNF decomposition algorithm (self studying)

---

**Input:** A relation R with a set of FD's F

**Output:** A BCNF decomposition of R with lossless join

**Method:**

- At each step compute the key for the sub-relation R
- if not in BCNF, pick any FD  $X \rightarrow Y$  which violates
- break the relation into 2 sub-relations
  - $R_1(XY)$
  - $R_2(S - Y)$
  - this has a lossless join
  - project FD's onto each sub-relation
- continue until no more offending FD's

# 3NF decomposition algorithm – self studying

---

**Input:** A relation  $R$  with a set of FD's  $F$

**Output:** A decomposition of  $R$  into a collection of relations, all of which are in 3NF. This decomposition has a lossless join and dependency-preservation.

**Method:**

- Find minimal basic for  $F$ , say  $G$ .
- $\forall X-A \in G$ , use  $XA$  as the schema of one relations in the decomposition.
- If none of the sets of relations from Step 2 is a super key for  $R$ , add another relation whose schema is a key for  $R$ .

# Summary 1

---

Decompose a relation into BCNF is a solution for eliminating anomalies

But BCNF can cause information loss and dependency loss

3NF is a relax solution of BCNF that keep loss-less join and dependency-preservation properties



# Summary 2:

2NF	3NF	Boyce-Codd
every nonprime attribute $A$ in $R$ is not partially dependent on <i>any</i> key of $R$	a <i>nontrivial</i> functional dependency: $X \Rightarrow A$ holds in $R$ , either (a) $X$ is a superkey of $R$ , or (b) $A$ is a prime attribute of $R$ .	a <i>nontrivial</i> functional dependency $X \Rightarrow A$ holds in $R$ , then: a) $X$ is a superkey of $R$
<b><u>Note:</u></b> A functional dependency $X \Rightarrow Y$ is a <b>full functional dependency</b> if removal of any attribute $A$ from $X$ means that the dependency does not hold any more; A <b><u>partial functional dependency</u></b> is not a <b><u>full functional dependency</u></b>		