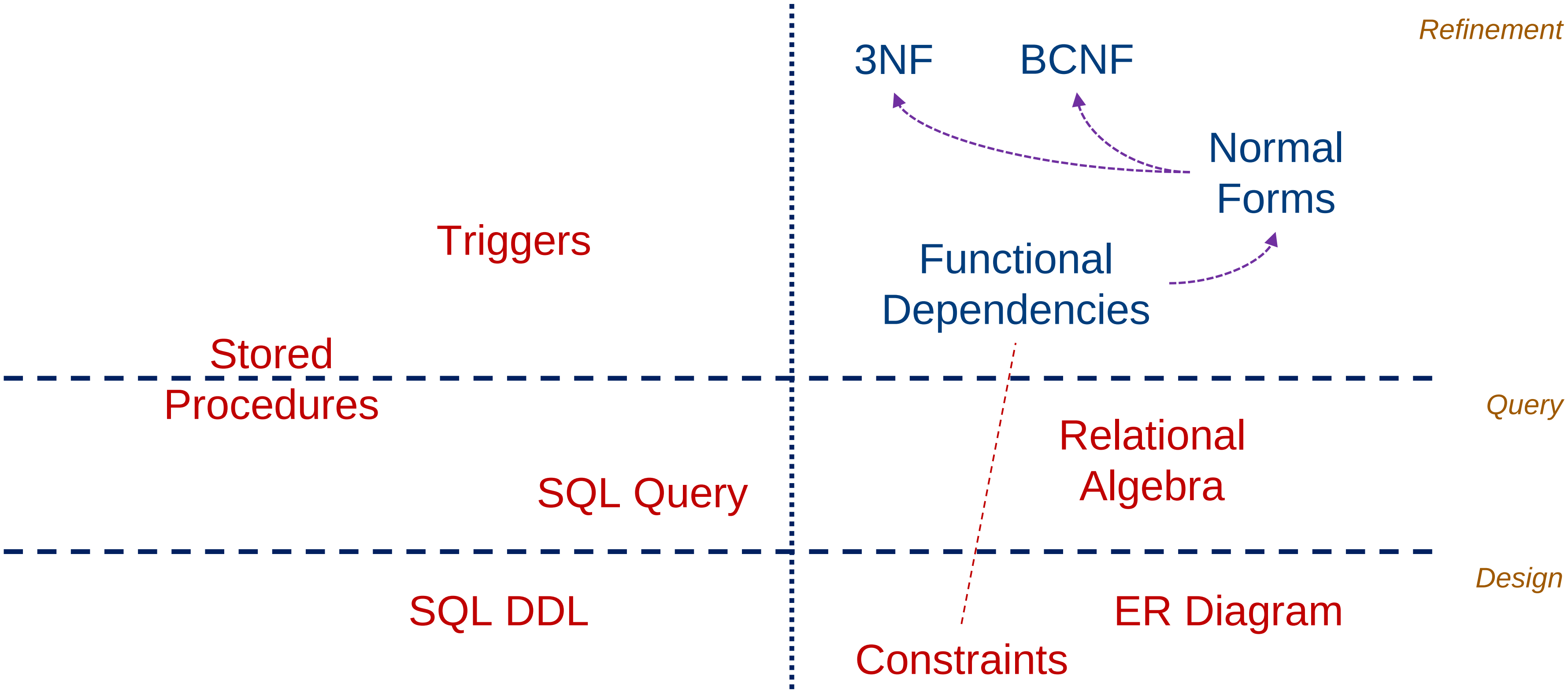
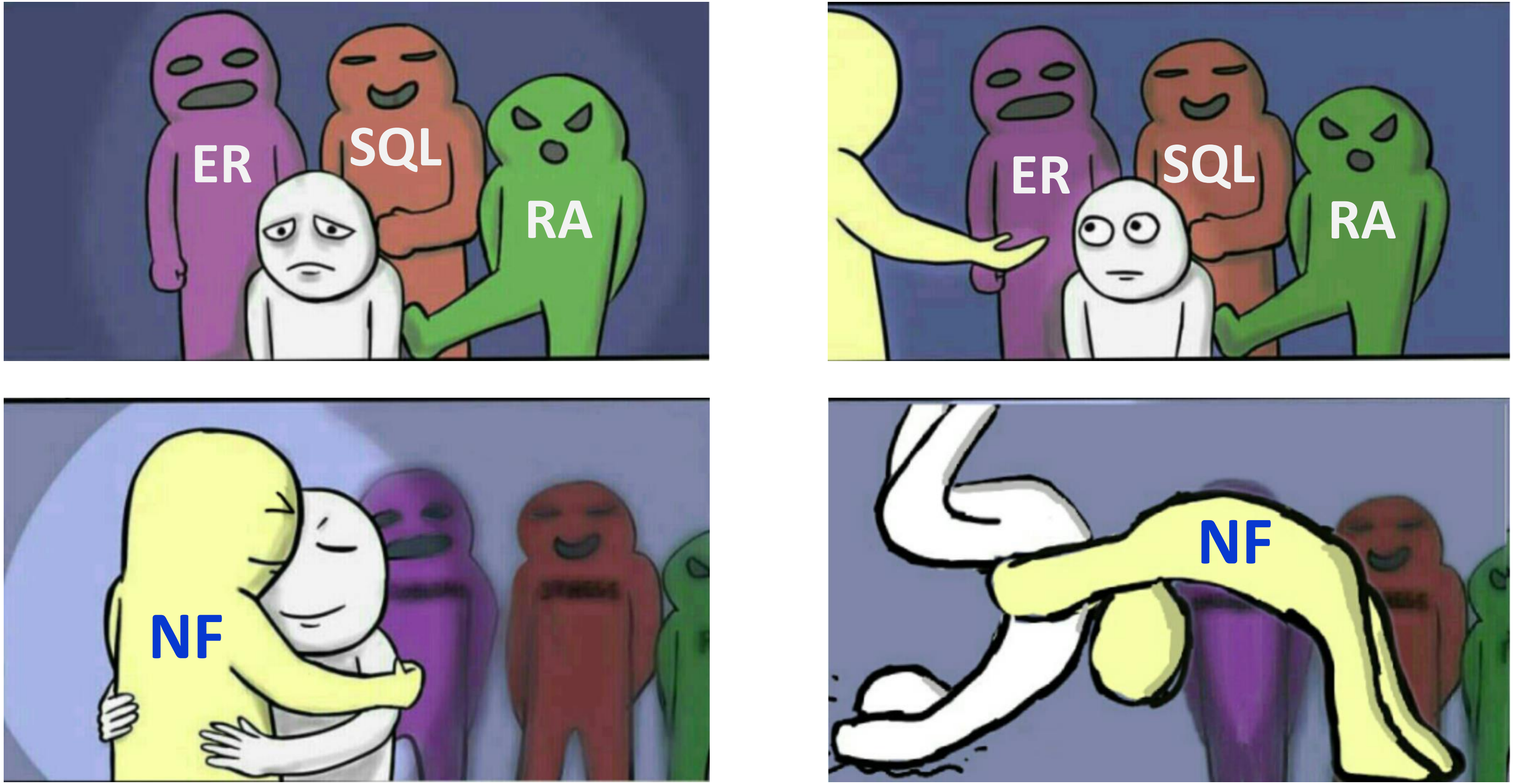
CS2102 Database Systems

Lecture 10 – Functional Dependencies

Roadmap

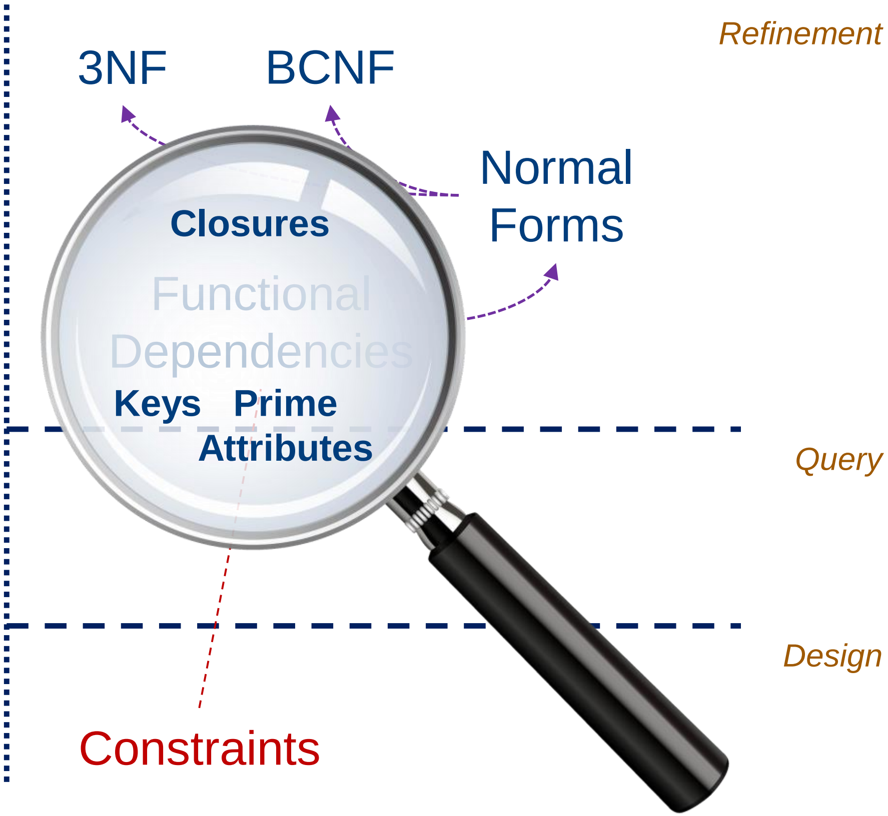

Refinement
3NF
BCNF
Normal Forms
Functional Dependencies
Triggers
Stored Procedures
Query
SQL Query
Relational Algebra
Design
SQL DDL
ER Diagram
Constraints

Normal Forms vs ER, SQL and RA



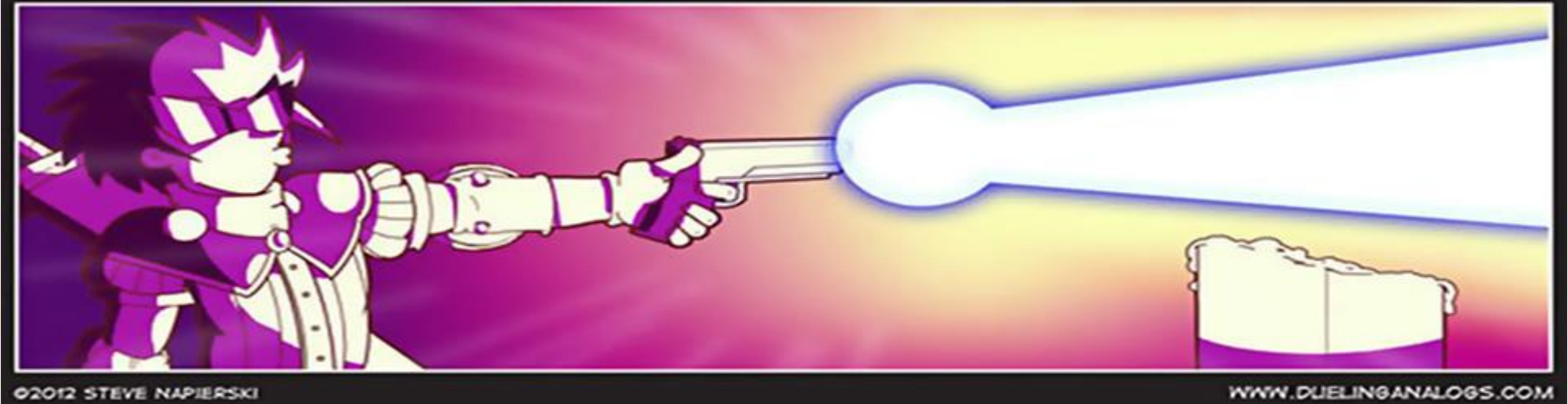
Roadmap

- We will do this step by step
  - Functional dependencies
  - Closures
  - Keys, superkeys and prime attributes
  - Normal forms and schema refinement *(next week)*



Doing normal forms **without** knowing functional dependencies, closures, keys, ...

Doing normal forms **after** knowing functional dependencies, closures, keys, ...



Anomalies

Anomalies

- **Motivation**
  - Suppose that we give ER diagram to both Alice and Bob
  - Each of them translates the diagram into a relational schema
  - Both claims that theirs is the best relational schema of all time
  - How to decide which one is better?

Anomalies

- **Motivation**
  - How to decide which one is better?
    - There could be many different ways to evaluate whether a relational schema is “good”
      - Different people may have different opinions
    - But there are things that should not be done
      - There are some *minimum* requirements to be met

  - A normal form is a definition of minimum requirements in terms of redundancy

A DB admin trying to explain why their relational schema is good without knowing FD

Anomalies

- **Redundancy**

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

  - Primary key
    - `(NRIC, Phone)`
  - There is some redundancy in terms of Alice's address
    - It is unnecessarily stored twice
    - This could lead to several anomalies

Anomalies

- **Update Anomalies**

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Clementi
Bob	5678	98765432	Pasir Ris

- Primary key
  - `(NRIC, Phone)`

❖ We may accidentally update one of Alice’s addresses, leaving the other unchanged

Anomalies

- **Deletion Anomalies**

Table "Student_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

Table "Student_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	`NULL`	Pasir Ris

- Primary key
  - `(NRIC, Phone)`
- Let's say Bob no longer uses a phone
  - Can we remove Bob's phone number?
  - NO! *(primary key attributes cannot be NULL)* *(otherwise we remove Bob completely)*

Anomalies

- **Insertion Anomalies**

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris
Cathy	9876	NULL	Yishun

- Primary key
  - (NRIC, Phone)
- Let’s say we have a new student Cathy
  - But Cathy does not use phone, can we add Cathy?
  - NO! *(primary key attributes cannot be NULL)*

Anomalies

- **Normalization**

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

Table "Student_Info"

Name	NRIC	Address
Alice	1234	Jurong East
Bob	5678	Pasir Ris

Table "Student_Contact"

NRIC	Phone
1234	67899876
1234	83838484
5678	98765432

- How do we get rid of those anomalies?
  - Split the table *(normalize it)*
- *Redundancy?* → *No.* *(Alice's address no longer duplicated)*
- *Update anomalies?* → *No.* *(only one place to update Alice's address)*
- *Deletion anomalies?* → *No.* *(can delete from `Student_Contact` freely)*
- *Insertion anomalies?* → *No.* *(entry in `Student_Contact` is optional)*
- *Can we get back `Student_Data`?* → *Yes.* *(by performing natural join)*

Anomalies

- **Normalization**

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

Table "Student_Info"

Name	NRIC	Address
Alice	1234	Jurong East
Bob	5678	Pasir Ris

Table "Student_Contact"

NRIC	Phone
1234	67899876
1234	83838484
5678	98765432

- How do we do such normalizations?
  - *Following some procedures designed according to normal forms*
  - Don’t be hasty, we will do this step-by-step

Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → Normal Forms → BCNF / 3NF

Functional Dependencies

Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → Normal Forms → BCNF, 3NF

Functional Dependencies

- ## Previous Example

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

An Apology

- I will inadvertently forgot to use `{}`
- I will sometimes use a single letter to indicate an attribute
- I will inadvertently forgot to use `,` to separate these single letter attributes

  - We mentioned that this table is bad because of the redundancy in `Address`
    - What causes this redundancy?
      - Some dependency between `NRIC` and `Address`
      - In particular, `NRIC` *uniquely identifies* `Address` *(but primary key is (`NRIC, Phone`))*
      - This is called *functional dependencies* (FD)
        - Denoted by `{NRIC}` → `{Address}`

Functional Dependencies

tl;dr

Given `{A,B} → {C,D}`
- If we know the value of both attributes A and B
- Then we know the value of both attributes C and D
- For all possible rows
- ❖ *But if we know the value of attributes C and D, we may not know the value of attributes A and B*

- **Formal Definition of *Uniquely Identifies***
  - Let $A_1, A_2, \cdots, A_m, B_1, B_2, \cdots, B_n$ be some attributes
  - We say that $\{A_1 A_2 \cdots A_m\} \rightarrow \{B_1 B_2 \cdots B_n\}$, if:
    - Whenever two tuples have the same values on A_1, A_2, ... and A_m
    - They always have the same values on B_1, B_2, ... and B_n
  - Example: `{NRIC} → {Name}`
    - Reads as *"`NRIC` decides `Name`"* or *"`NRIC` determines `Name`"*
      - Informally, *"FD `NRIC` to `Name`"* *(I will basically accidentally say it like this)*
    - Meaning:
      - If two tuples have the same `NRIC` value, then they have the same `Name` value

❖ ***Note the asymmetry***

Why is it nice to have such definition?

Functional Dependencies

- **Examples**
  - Which of the following functional dependencies are FALSE?

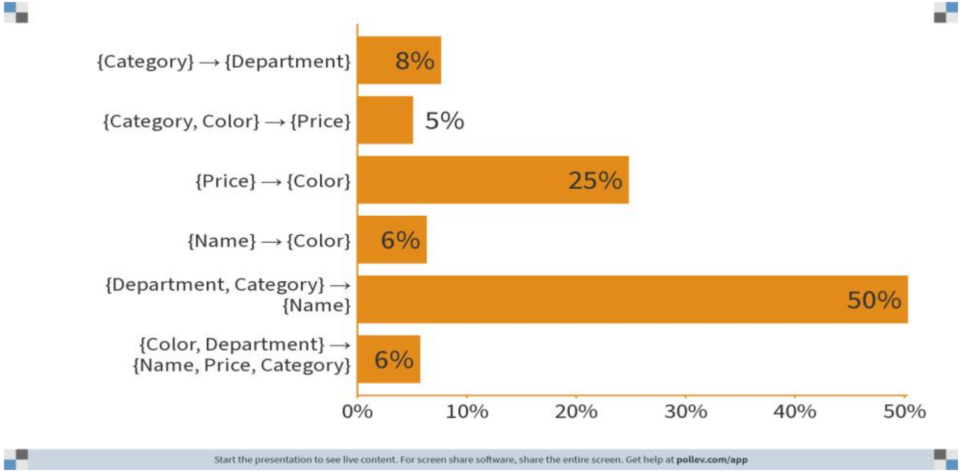
Table "Shops"

Name	Category	Color	Department	Price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office Supplies	59

Note

How do we solve this kind of question?

- Find the actual violation to the definition of functional dependencies LHS → RHS
- If there are two rows such that:
  - The values of the attributes on the LHS are the same
  - The values of the attributes on the RHS are different
- Otherwise, not enough information and the FD ***may*** be true



Functional Dependencies

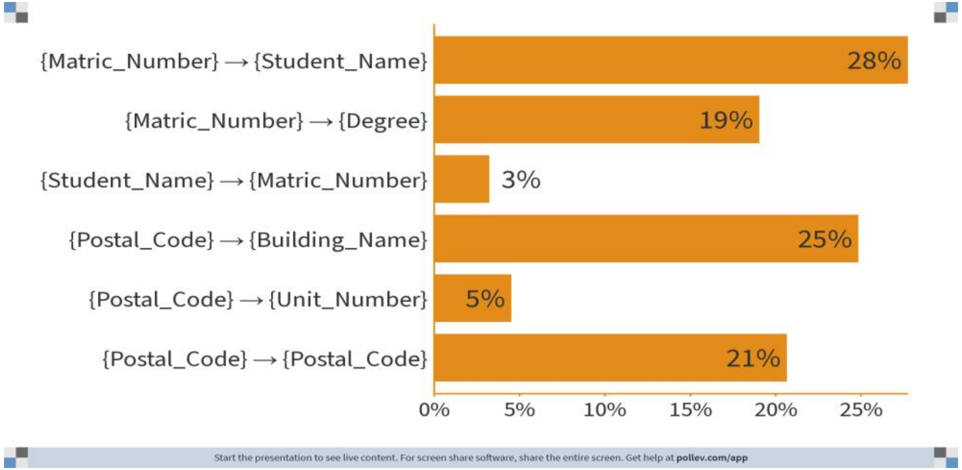
- **Examples**
  - Which of the following functional dependencies are *likely* `TRUE`?

Assumes real-life constraints

Note

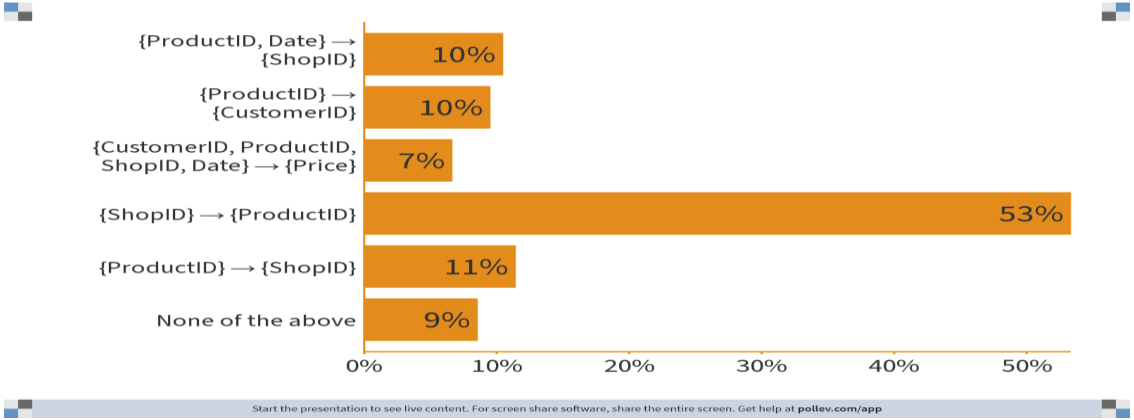
How do we solve this kind of question?

- Similar to before, but if we cannot definitely say that these are false, then they ***may*** be true
- Of course, the question here is "ambiguous" because of the word *likely*
- In fact, the last option is ***definitely*** true



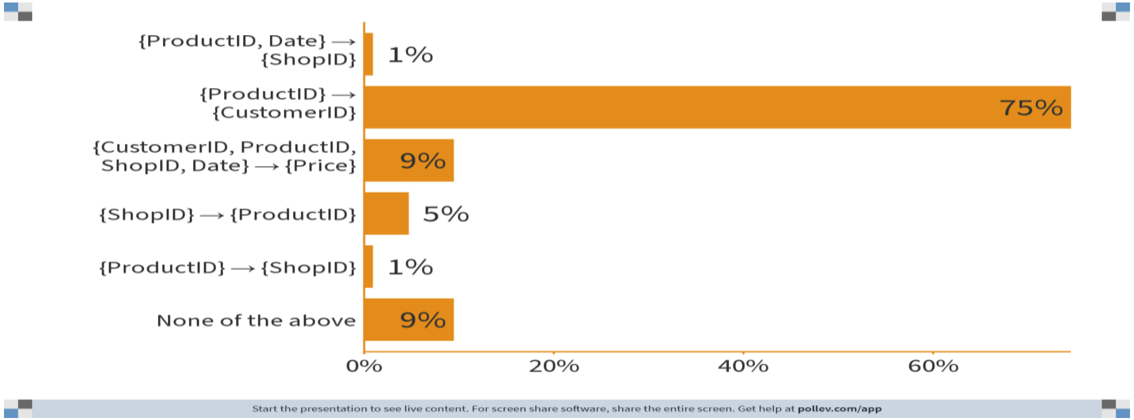
Functional Dependencies

- **Where Do FDs Come From?**
  - From common sense *(see previous slide)*
  - From the application's requirements
    - `Purchase(CustomerID, ProductID, ShopID, Price, Date)`
    - **Requirement #1** *Each shop can sell at most one product*



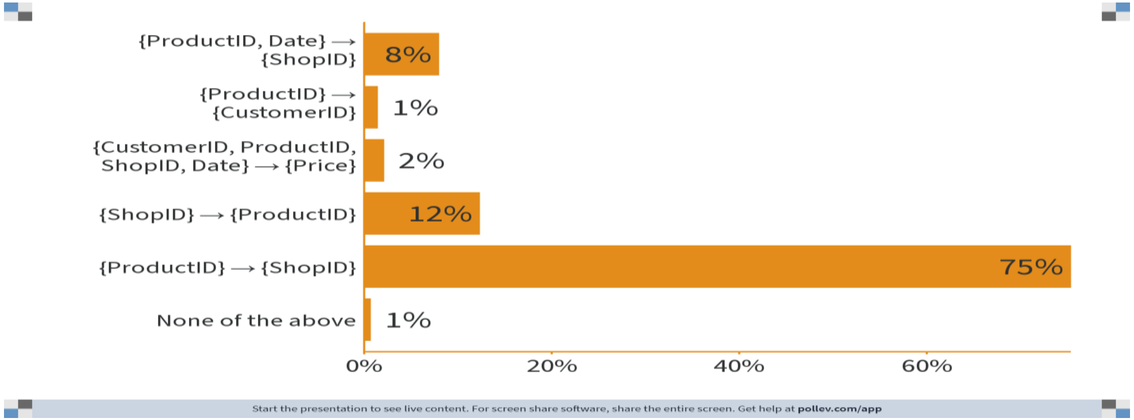
Functional Dependencies

- **Where Do FDs Come From?**
  - From common sense *(see previous slide)*
  - From the application's requirements
    - `Purchase(CustomerID, ProductID, ShopID, Price, Date)`
    - **Requirement #2** *No two customers buy the same product*



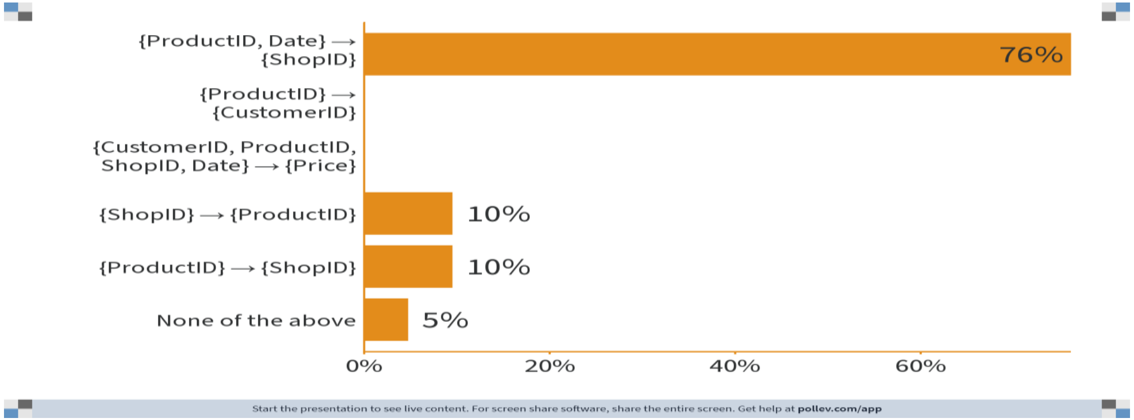
Functional Dependencies

- **Where Do FDs Come From?**
  - From common sense *(see previous slide)*
  - From the application's requirements
    - `Purchase(CustomerID, ProductID, ShopID, Price, Date)`
    - **Requirement #3** *No two shops sell the same product*



Functional Dependencies

- **Where Do FDs Come From?**
  - From common sense *(see previous slide)*
  - From the application's requirements
    - `Purchase(CustomerID, ProductID, ShopID, Price, Date)`
    - **Requirement #4** *No two shops sell the same product on the same date*



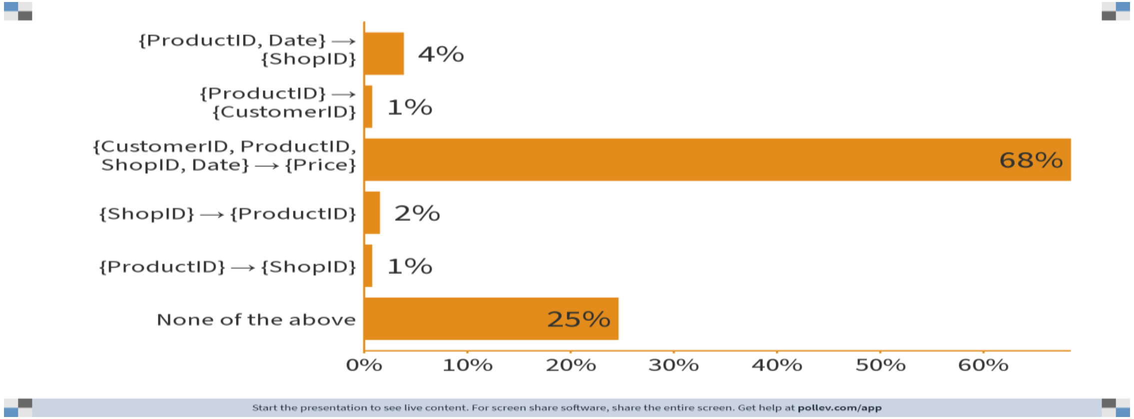
Functional Dependencies

- **Where Do FDs Come From?**
  - From common sense *(see previous slide)*
  - From the application's requirements
    - `Purchase(CustomerID, ProductID, ShopID, Price, Date)`
    - **Requirement #5** *No two shops sell the same product to the same customer on the same date at two different prices*

Note

As per discussion during the lecture, the problem here is the ambiguity in the English words used. To resolve this (e.g., for projects), do the following:

1. Check in "Module Details" in LumiNUS, there is a "Project Q&A" card
2. Ask in forum if still not clear
3. Write down the constraints in FD



Closures

Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → Normal Forms → BCNF, 3NF

Closures

- **FD Reasoning**
  - Now that we know what FDs are
  - Next, we will discuss how to do reasoning with FDs
  - **Example:**
    - We know that
      - `{NRIC}` → `{Matric_Number}`
      - `{Matric_Number}` → `{Name}`
    - What else can we know?
      - `{NRIC}` → `{Name}` *(by transitivity)*
  - **Objective:** Given a set of FDs, figure out what other FDs they can imply
    - This is ***important*** for normal forms

Closures

- **FD Reasoning**
  - If we know that
    - `{NRIC}` → `{Matric_Number}`
    - `{Matric_Number}` → `{Name}`
  - How do we know?
    - `{NRIC}` → `{Name}` *(by transitivity)*
- **Proof:** by *contradiction*
  1. Assume not `{NRIC}` → `{Name}` *(denoted by `{NRIC}` ↛ `{Name}`)*
  2. Then there is a tuple $\langle N, MN_1, M_1 \rangle$ and $\langle N, MN_2, M_2 \rangle$ such that $M_1 \neq M_2$
  3. By `{NRIC}` → `{Matric_Number}`,
     we know that $MN_1 = MN_2 = MN$
  4. Since $MN_1 = MN_2 = MN$, by `{Matric_Number}` → `{Name}`,
     we know that $M_1 = M_2 = M$
  5. This contradicts (2), so we retract on (1) and conclude that `{NRIC}` → `{Name}`

Note

You do ***NOT*** have to understand the proof but it will be beneficial if you do. The point of this "proof" is to show the difficulty in proving anything in FD. However, we do have "tools" to help us:

1. Armstrong's Axioms
2. Closure

Closures

- **Armstrong's Axioms**
  - Three fundamental axioms for FD reasoning
  1. **Axiom of Reflexivity**
     - A set of attributes → A subset of the attributes
  - **Example:**
    - `{NRIC, Name} → {NRIC}`
    - `{StudentID, Name, Age} → {Name, Age}`
    - `{ABCD} → {ABC}`
    - `{ABCD} → {BCD}`
    - `{ABCD} → {AD}`
    - `{ABCD} → {ABCD}`

Closures

Armstrong's Axioms

1. Reflexivity {AB} → {A}

- **Armstrong's Axioms**
  - Three fundamental axioms for FD reasoning
  - **2. Axiom of Augmentation**
    - If {A} → {B}
    - Then {AC} → {BC} *(for any C)*
  - **Example:** if {NRIC} → {Name} then
    - {NRIC, Age} → {Name, Age}
    - {NRIC, Salary, Weight} → {Name, Salary, Weight}
    - {NRIC, Address, Postal} → {Name, Address, Postal}

Closures

Armstrong's Axioms

1. **Reflexivity** $\{AB\} \rightarrow \{A\}$
2. **Augmentation** $\{A\} \rightarrow \{B\} \Rightarrow \{AC\} \rightarrow \{BC\}$

- **Armstrong's Axioms**
  - Three fundamental axioms for FD reasoning

  3. **Axiom of Transitivity**
     - If $\{A\} \rightarrow \{B\}$ and $\{B\} \rightarrow \{C\}$
     - Then $\{A\} \rightarrow \{C\}$

  - **Example:**
     - if $\{NRIC\} \rightarrow \{Address\}$
     - and $\{Address\} \rightarrow \{Postal\}$
     - then $\{NRIC\} \rightarrow \{Postal\}$

Closures

Armstrong's Axioms

1. **Reflexivity** $\{AB\} \rightarrow \{A\}$
2. **Augmentation** $\{A\} \rightarrow \{B\} \Rightarrow \{AC\} \rightarrow \{BC\}$
3. **Transitivity** $\{A\} \rightarrow \{B\}$ & $\{B\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{C\}$

- ## Extended Armstrong's Axioms
  - Two additional theorems for FD reasoning

A. Rule of Decomposition

- If $\{A\} \rightarrow \{BC\}$
- Then $\{A\} \rightarrow \{B\}$ and $\{A\} \rightarrow \{C\}$

- **Proof:**

1. $\{A\} \rightarrow \{BC\}$ Assume
2. $\{BC\} \rightarrow \{B\}$ Reflexivity $B \subseteq BC$
3. $\{A\} \rightarrow \{B\}$ Transitivity (1) and (2) ■
4. $\{BC\} \rightarrow \{C\}$ Reflexivity $C \subseteq BC$
5. $\{A\} \rightarrow \{C\}$ Transitivity (1) and (4) ■

Closures

Armstrong's Axioms

1. Reflexivity	$\{AB\} \rightarrow \{A\}$
2. Augmentation	$\{A\} \rightarrow \{B\} \Rightarrow \{AC\} \rightarrow \{BC\}$
3. Transitivity	$\{A\} \rightarrow \{B\}$ & $\{B\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{C\}$

- **Extended Armstrong's Axioms**
  - Two additional theorems for FD reasoning

  B. **Rule of Union**
    - If $\{A\} \rightarrow \{B\}$ and $\{A\} \rightarrow \{C\}$
    - Then $\{A\} \rightarrow \{BC\}$
  - **Proof:**
    1. $\{A\} \rightarrow \{B\}$ Assume
    2. $\{A\} \rightarrow \{C\}$ Assume
    3. $\{A\} \rightarrow \{AB\}$ Augmentation (1) with A
    4. $\{AB\} \rightarrow \{BC\}$ Augmentation (2) with B
    5. $\{A\} \rightarrow \{BC\}$ Transitivity (3) and (4) ■

Mid Section Summary

Tools

There is a system available online to help check the correctness of a proof using these axioms
https://www.comp.nus.edu.sg/~adi-yoga/CS2102/armstrong/
It only works with the basic and NOT the extended version

- **Armstrong's Axioms**
  - Three fundamental axioms for FD reasoning

1. **Axiom of Reflexivity**
   - A set of attributes → A subset of the attributes
2. **Axiom of Augmentation**
   - If {A} → {B}
   - Then {AC} → {BC} *(for any C)*
3. **Axiom of Transitivity**
   - If {A} → {B} and {B} → {C}
   - Then {A} → {C}

Implication

Due to the soundness and completeness property, repeated usage of the 3 axioms will eventually gives no new FD. This is called the "closure" (i.e., maximal information)

Notes *(no need to know in details)*

- **Sound**
  - All FD that can be *derived* using this are correct FD
- **Complete**
  - All FD that can be derived *can be derived* using these rules

❖ This is not the only sound and complete rules
❖ We are not interested in proving these, only using them

Mid Section Summary

- **Extended Armstrong's Axioms**
  - Two additional theorems for FD reasoning

A. Rule of Decomposition

- If $\{A\} \rightarrow \{BC\}$
- Then $\{A\} \rightarrow \{B\}$ and $\{A\} \rightarrow \{C\}$

B. Rule of Union

- If $\{A\} \rightarrow \{B\}$ and $\{A\} \rightarrow \{C\}$
- Then $\{A\} \rightarrow \{BC\}$

Please be careful on assignments/assessments for which we specify whether you can use the extended version on not
⇒ but Assignment 2 is not yet created, so stay tuned

Notes *(no need to know in details)*

- **Sound**
  - Since the rules can be derived from Armstrong's axioms
- **Complete**
  - Since Armstrong's axioms already complete and these only add

❖ Note that you do not need to use these because they can be derived by the three fundamental axioms
❖ These are useful for simplification

Closures

Armstrong's Axioms

1. **Reflexivity** $\{AB\} \rightarrow \{A\}$
2. **Augmentation** $\{A\} \rightarrow \{B\} \Rightarrow \{AC\} \rightarrow \{BC\}$
3. **Transitivity** $\{A\} \rightarrow \{B\}$ & $\{B\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{C\}$

Extended Armstrong's Axioms

A. **Decomposition** $\{A\} \rightarrow \{BC\} \Rightarrow \{A\} \rightarrow \{B\}$ & $\{A\} \rightarrow \{C\}$
B. **Union** $\{A\} \rightarrow \{B\}$ & $\{A\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{BC\}$

- **Example**
  - **Given:**
    1. $\{A\} \rightarrow \{B\}$
    2. $\{BC\} \rightarrow \{D\}$
  - **Target:** $\{AC\} \rightarrow \{D\}$
  - **Proof:**
    3. $\{AC\} \rightarrow \{BC\}$ Augmentation (1) with C
    4. $\{AC\} \rightarrow \{D\}$ Transitivity (3) and (2) ■

Closures

- **Exercise #1**
  - **Given:**
    1. {A} → {B}
    2. {D} → {C}
  - **Target:** {AD} → {BC}
  - **Proof:**
    3. {AD} → {BD} Augmentation (1) with D
    4. {AD} → {B} Decomposition of (3)
    5. {AD} → {AC} Augmentation (2) with A
    6. {AD} → {C} Decomposition of (5)
    7. {AD} → {BC} Union of (4),(6) ■

Armstrong's Axioms

1. **Reflexivity** {AB} → {A}
2. **Augmentation** {A} → {B} ⇒ {AC} → {BC}
3. **Transitivity** {A} → {B} & {B} → {C} ⇒ {A} → {C}

Extended Armstrong's Axioms

A. **Decomposition** {A} → {BC} ⇒ {A} → {B} & {A} → {C}
B. **Union** {A} → {B} & {A} → {C} ⇒ {A} → {BC}

Closures

Armstrong's Axioms

1. Reflexivity	$\{AB\} \rightarrow \{A\}$
2. Augmentation	$\{A\} \rightarrow \{B\} \Rightarrow \{AC\} \rightarrow \{BC\}$
3. Transitivity	$\{A\} \rightarrow \{B\}$ & $\{B\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{C\}$

Extended Armstrong's Axioms

A. Decomposition	$\{A\} \rightarrow \{BC\} \Rightarrow \{A\} \rightarrow \{B\}$ & $\{A\} \rightarrow \{C\}$
B. Union	$\{A\} \rightarrow \{B\}$ & $\{A\} \rightarrow \{C\} \Rightarrow \{A\} \rightarrow \{BC\}$

- **Exercise #2**
  - **Given:**
    1. $\{A\} \rightarrow \{C\}$
    2. $\{AC\} \rightarrow \{D\}$
    3. $\{AD\} \rightarrow \{B\}$
  - **Target:** $\{A\} \rightarrow \{B\}$
  - **Proof:**
    4. $\{A\} \rightarrow \{AC\}$ Augmentation (1) with A
    5. $\{A\} \rightarrow \{D\}$ Transitivity (4) and (2)
    6. $\{A\} \rightarrow \{AD\}$ Augmentation (5) with A
    7. $\{A\} \rightarrow \{B\}$ Transitivity (6) and (3) ■

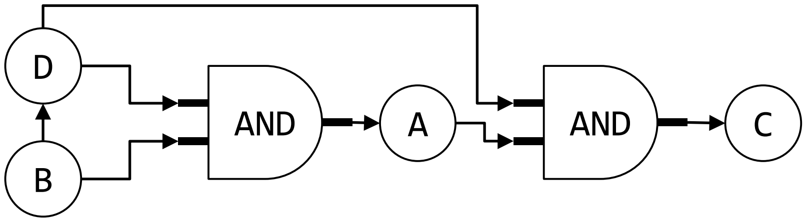
Closures

- **Armstrong's Axioms**
  - Using (*extended*) Armstrong's axioms to do FD reasoning is a bit cumbersome
    - **Proof:** *Previous few slides* ■
  - We will describe a *mechanical* approach called *closure*
  - **Observation:**
    1. By *Rule of Union*, we can find the largest $B_1 B_2 \cdots B_n$ in $\{A_1 A_2 \cdots A_m\} \rightarrow \{B_1 B_2 \ldots B_n\}$
    2. By *Rule of Decomposition*, the largest $B_1 B_2 \ldots B_n$ implies individual $\{A_1 A_2 \cdots A_m\} \rightarrow \{B_1\}$,$\{A_1 A_2 \cdots A_m\} \rightarrow \{B_2\}$, $\cdots$,$\{A_1 A_2 \cdots A_m\} \rightarrow \{B_n\}$
    3. So, knowing $B_1 B_2 \ldots B_n$ is sufficient to know all that is determined by $A_1 A_2 \ldots A_m$

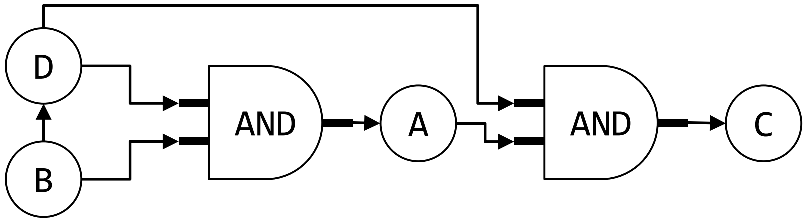
❖ We call $B_1 B_2 \ldots B_n$ the closure of $A_1 A_2 \ldots A_m$ denoted by $\{A_1 A_2 \ldots A_m\}^+$

Closures

- **Armstrong's Axioms**
  - Using (*extended*) Armstrong's axioms to do FD reasoning is a bit cumbersome
    - **Proof:** *Previous few slides* ■
  - We will describe a *mechanical* approach called *closure*
  - **Intuition:**
    - FDs are kind of like components on a circuit board



Closures

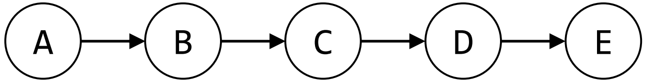


- **Motivating Example**
  - Four attributes A, B, C and D
  - Given {B} → {D}, {BD} → {A} and {AD} → {C}
  - Check {B} → {C}
  - **Steps:**
    1. Activate B — Activated set = {B}
    2. Activate whatever {B} can activate — Activated set = {B,D}
    3. Activate whatever {B,D} can activate — Activated set = {A,B,D}
    4. Activate whatever {A,B,D} can activate — Activated set = {A,B,C,D}
    5. Until no more can be activated

❖ Everything that is activated is the closure of {B} denoted by $\{B\}^+$

Closures

- **Definition**
  - Let $S = \{A_1, A_2, \ldots, A_n\}$ be a set of attributes
  - The closure of S is the set of attributes that can be decided by A_1, A_2, ..., A_n
    - Directly or indirectly
    - Denoted by $\{A_1, A_2, \ldots, A_n\}^+$
  - **Example:**
    - Given: $\{A\} \rightarrow \{B\}$, $\{B\} \rightarrow \{C\}$, $\{C\} \rightarrow \{D\}$, $\{D\} \rightarrow \{E\}$
      - $\{A\}^+ = \{A, B, C, D, E\}$
      - $\{B\}^+ = \{B, C, D, E\}$
      - $\{C\}^+ = \{C, D, E\}$
      - $\{D\}^+ = \{D, E\}$
      - $\{E\}^+ = \{E\}$



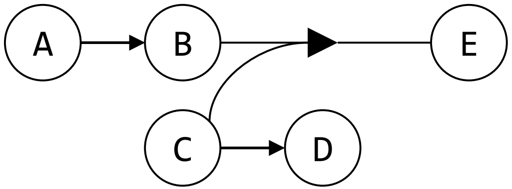
Closures

Tools

There is a separate system that allows computation of closure plus some additional tools not yet covered in lecture
https://www.comp.nus.edu.sg/~adi-yoga/CS2102/FD/
It has Armstrong's axioms + computation of closures, keys, etc
See notes at the end of the lecture slides

- **Computing Closures**
  - Let $S = \{A_1, A_2, \ldots, A_n\}$ be a set of attributes
  - The closure of S denoted by S^+ or $\{A_1, A_2, \ldots, A_n\}^+$ can be computed by
    1. Initialize the closure to $\{A_1, A_2, \ldots, A_n\}$
    2. If there is an FD: $A_i, A_j, \ldots, A_m \rightarrow B$ such that $A_i, A_j, \ldots, A_m$ are all in the closure, then put B into the closure
    3. Repeat step 2 until we cannot find any new attribute to put into the closure
  - **Example:**
    - A table with five attributes A, B, C, D and E
    - FD: {A} → {B}, {C} → {D} and {BC} → {E}

1. $\{A\}^+$ = {A, B}
2. $\{A,C\}^+$ = {A, B, C, D, E}
3. $\{B\}^+$ = {B}



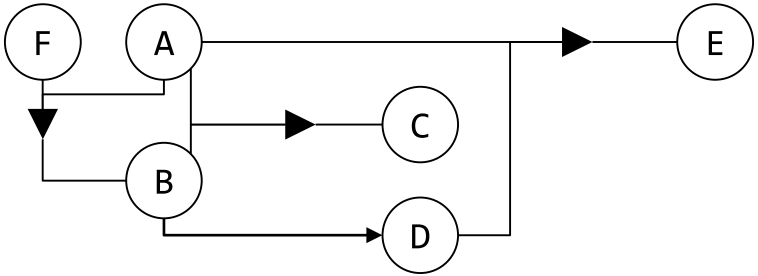
Closures

- To prove that {X} → {Y} holds
  - Show that $\{X\}^+$ contains Y
- To prove that {X} → {Y} doesn't hold

  (i.e., {X} ↛ {Y})
  - Show that $\{X\}^+$ does not contain Y

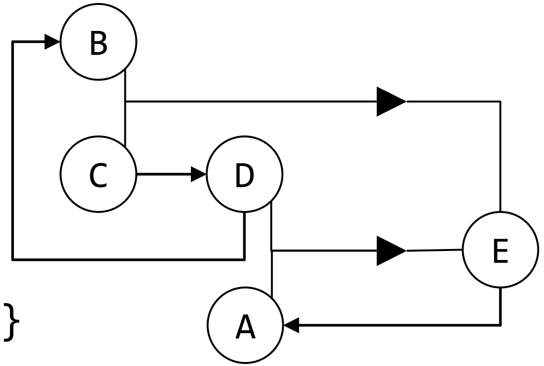
Example: $AB \rightarrow C$, $AD \rightarrow E$, $B \rightarrow D$ and $AF \rightarrow B$

- Prove that {A, F} → {D} holds
  - $\{A, F\}^+ = \{A, B, C, \mathbf{D}, E, F\}$
  - $D \in \{A, F\}^+$
  - ∴ {A, F} → {D} holds
- Prove that {A, D} → {F} does not hold
  - $\{A, D\}^+ = \{A, D, E\}$
  - $F \notin \{A, D\}^+$
  - ∴ {A, D} → {F} does not hold



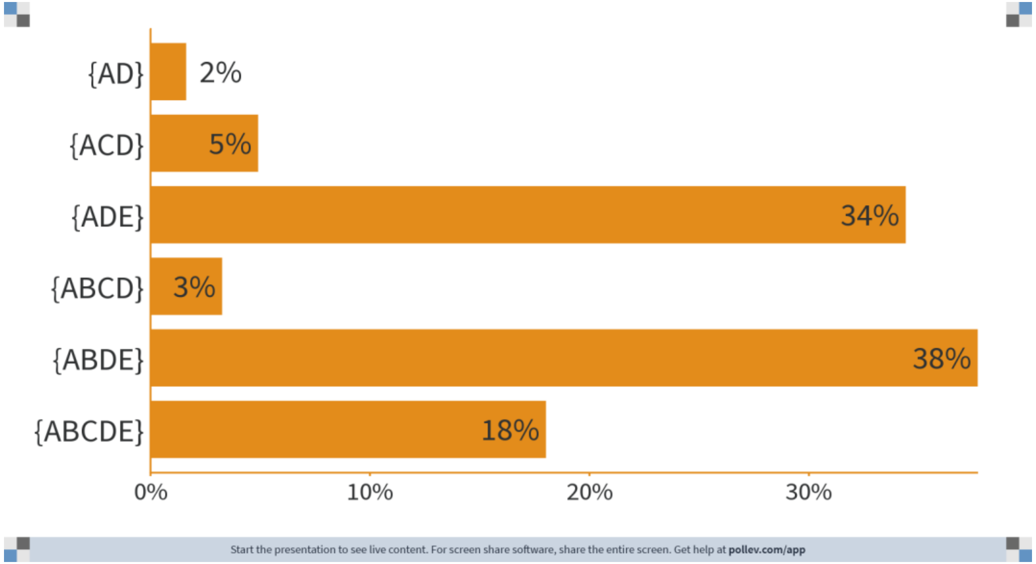
Closures

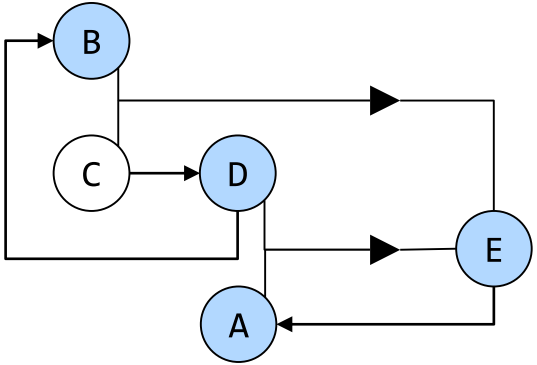
- **Exercise #1**
  - **Given:** {C} → {D}, {AD} → {E}, {BC} → {E}, {E} → {A} and {D} → {B}
  - **Check:** does {C} → {A} holds
  - **Steps:**
    1. Compute $\{C\}^+$
       a) Starts with {C}
       b) Since {C} → {D}, we have {C,D}
       c) Since {D} → {B}, we have {B,C,D}
       d) Since {BC} → {E}, we have {B,C,D,E}
       e) Since {E} → {A}, we have {A,B,C,D,E}
    2. Since A ∈ $\{C\}^+$, then {C} → {A} holds



Closures

- **Exercise #2**
  - **Given:** {C} → {D}, {AD} → {E}, {BC} → {E}, {E} → {A} and {D} → {B}
  - **Question:** what is {AD}$^{+}$?





Keys, Superkeys and Prime Attributes

Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → Normal Forms → BCNF, 3NF

Superkeys

- **Superkeys of a Table**

Table "Student_Data"

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3579	420923	Yishun

- **Definition:** A set of attributes in a table that **decides** all other attributes
- **Example:**
  - `{NRIC}` is a superkey since `{NRIC}` → `{Name, Postal, Address}`
  - `{NRIC, Name}` is a superkey
    - Since `{NRIC, Name}` → `{Postal, Address}`

Keys

- **Keys of a Table**

Table "Student_Data"

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3579	420923	Yishun

  - **Definition:** A superkey that is **minimal**
  - **Example:**
    - `{NRIC}` is a superkey since `{NRIC} → {Name, Postal, Address}`
    - `{NRIC, Name}` is a superkey
      - Since `{NRIC, Name} → {Postal, Address}`
    - `{NRIC}` is key, BUT `{NRIC, Name}` is NOT a key

Keys

- **Keys of a Table**

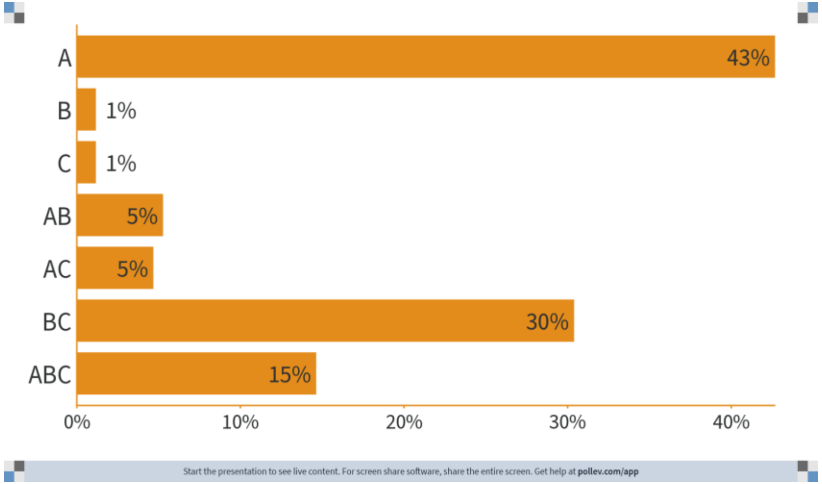
Table "Student_Data"

Name	NRIC	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3579	3	420923	Yishun

  - A table may have multiple keys
  - **Example:**
    - `{NRIC}` is a key
      - Since `{NRIC}` → `{Name, StudentID, Postal, Address}`
    - `{StudentID}` is a key
      - Since `{StudentID}` → `{Name, NRIC, Postal, Address}`
    - Both `{NRIC}` and `{StudentID}` are keys

Keys

- **Exercise**
  - **Given:**
    - A table `T(A, B, C)` with three attributes `A`, `B` and `C`
    - Two FDs: `{A}` → `{BC}` and `{BC}` → `{A}`
  - Find the key(s) of `T`



Note

Minimality condition of keys K means the following:

- For every attribute a ∈ K, if we remove a from K (i.e., K – {a}), then the remaining attribute is no longer a superkey
  - In other words, $(K – \{a\})^+$ is not all other attributes in the relations

As you can see, from this definition, {BC} is minimal because removing either B or C from {BC} makes the remaining attribute not a superkey.

Keys

- **Why are We Talking About Keys?**

Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → **Normal Forms** → BCNF / 3NF

  - Because we needed it in our discussions of normal forms
    - Whether or not a table T has redundancy would *partially* depend on what the keys of T are

- **So How do We Compute the Keys?**
  - Check the FDs on the table T and use closures to derive the keys

Keys

- **Finding Keys**
  - **Definition:** A key is a minimal set of attributes that decides all other attributes
  - **Input:** A table `T(A, B, C, ...)` and a set of FDs on `T`
  - **Algorithm:**
    1. Consider every subset of attributes in T
       - A, B, C, ..., AB, AC, BC, ..., ABC, ...
    2. Derive the closure of each subset
       - $\{A\}^+$, $\{B\}^+$, $\{C\}^+$, ..., $\{AB\}^+$, $\{AC\}^+$, $\{BC\}^+$, ..., $\{ABC\}^+$, ...
    3. Identify all superkeys based on the closures
    4. Identify all keys from the superkeys
       - Find all superkeys for which its subset is not a key

Keys

- **Finding Keys**
  - **Example:** A table `R(A, B, C)` with $A \rightarrow B$ and $B \rightarrow C$
  - **Steps:**
    1. Consider every subset of attributes in T
       - A, B, C, AB, AC, BC, ABC
    2. Derive the closure of each subset

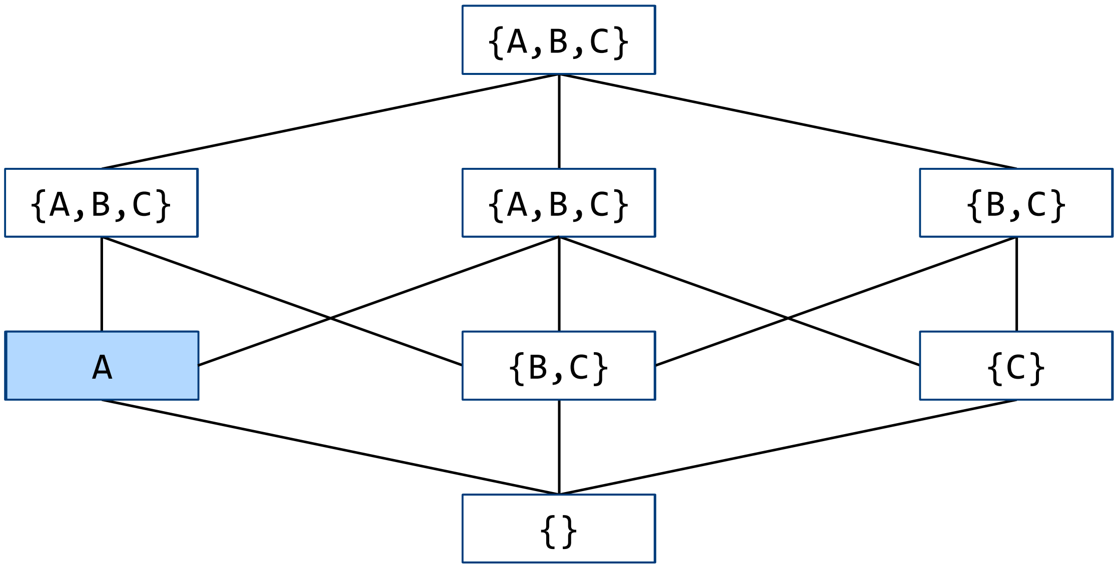
$\{A\}^+ = \{A,B,C\}$	$\{B\}^+ = \{B,C\}$	$\{C\}^+ = \{C\}$	
$\{AB\}^+ = \{A,B,C\}$	$\{AC\}^+ = \{A,B,C\}$	$\{BC\}^+ = \{B,C\}$	$\{ABC\}^+ = \{A,B,C\}$

    3. Identify all superkeys based on the closures
       - A, AB, AC, ABC
    4. Identify all the keys from the superkeys
       - A

Prime Attributes
{A}

Keys

- **Finding Keys**
  - **Example:** A table `R(A, B, C)` with `A → B` and `B → C`
  - **Steps Visualization**



Keys

- ## Finding Keys
  - **Exercise:** A table `R(A, B, C, D)` with `AB → C`, `AD → B` and `B → D`
  - **Steps:**
    1. Enumerate all subset of attributes

{A}	{B}	{C}	{D}
{AB}	{AC}	{AD}	
{BC}	{BD}	{CD}	
{ABC}	{ABD}	{ACD}	{BCD}
{ABCD}			

Keys

- **Finding Keys**
  - **Exercise:** A table `R(A, B, C, D)` with $AB \rightarrow C$, $AD \rightarrow B$ and $B \rightarrow D$
  - **Steps:**
    2. Compute the closures

$\{A\}^+ = \{A\}$	$\{B\}^+ = \{BD\}$	$\{C\}^+ = \{C\}$	$\{D\}^+ = \{D\}$
$\{AB\}^+ = \{ABCD\}$	$\{AC\}^+ = \{AC\}$	$\{AD\}^+ = \{ABCD\}$	
$\{BC\}^+ = \{BCD\}$	$\{BD\}^+ = \{BD\}$	$\{CD\}^+ = \{CD\}$	
$\{ABC\}^+ = \{ABCD\}$	$\{ABD\}^+ = \{ABCD\}$	$\{ACD\}^+ = \{ABCD\}$	$\{BCD\}^+ = \{BCD\}$
$\{ABCD\}^+ = \{ABCD\}$			

Keys

- **Finding Keys**
  - **Exercise:** A table `R(A, B, C, D)` with $AB \rightarrow C$, $AD \rightarrow B$ and $B \rightarrow D$
  - **Steps:**
    3. Identify the superkeys

$\{A\}^+ = \{A\}$	$\{B\}^+ = \{BD\}$	$\{C\}^+ = \{C\}$	$\{D\}^+ = \{D\}$
$\{AB\}^+ = \{ABCD\}$	$\{AC\}^+ = \{AC\}$	$\{AD\}^+ = \{ABCD\}$	
$\{BC\}^+ = \{BCD\}$	$\{BD\}^+ = \{BD\}$	$\{CD\}^+ = \{CD\}$	
$\{ABC\}^+ = \{ABCD\}$	$\{ABD\}^+ = \{ABCD\}$	$\{ACD\}^+ = \{ABCD\}$	$\{BCD\}^+ = \{BCD\}$
$\{ABCD\}^+ = \{ABCD\}$			

Keys

- **Finding Keys**
  - **Exercise:** A table `R(A, B, C, D)` with `AB → C`, `AD → B` and `B → D`
  - **Steps:**
    4. Identify the keys

$\{A\}^+$	=	$\{A\}$	$\{B\}^+$	=	$\{BD\}$	$\{C\}^+$	=	$\{C\}$	$\{D\}^+$	=	$\{D\}$
$\{AB\}^+$	=	$\{ABCD\}$	$\{AC\}^+$	=	$\{AC\}$	$\{AD\}^+$	=	$\{ABCD\}$			
$\{BC\}^+$	=	$\{BCD\}$	$\{BD\}^+$	=	$\{BD\}$	$\{CD\}^+$	=	$\{CD\}$			
$\{ABC\}^+$	=	$\{ABCD\}$	$\{ABD\}^+$	=	$\{ABCD\}$	$\{ACD\}^+$	=	$\{ABCD\}$	$\{BCD\}^+$	=	$\{BCD\}$
$\{ABCD\}^+$	=	$\{ABCD\}$									

Prime Attributes
{A, B, D}

Keys

- **Finding Keys**
  - Small tricks to help you
    1. Always check the small attributes sets first
       - **Example:**
         - R(A, B, C, D) with {A} → {B}, {B} → {C}, {C} → {D} and {D} → {A}
         - Compute closures

           {A}⁺ = {ABCD} | {B}⁺ = {ABCD} | {C}⁺ = {ABCD} | {D}⁺ = {ABCD}

         - No need to check other since they will be superkeys but not keys
    2. Any attributes not in right hand side of any FD **must be** in every key
       - **Why?** Not determined by any other attributes
       - **Example:** R(A, B, C, D) with AB → C, AD → B and B → D
         - A does not appear in right hand side of any FDs
         - Must be in keys *({AB} and {AD})*

Prime Attributes
{A, B, C, D}

Prime Attributes
{A, B, D}

Keys

- **Finding Keys**
  - **Exercise:** A table `R(A, B, C, D)` with $A \rightarrow B$, $A \rightarrow C$ and $C \rightarrow D$
  - **Steps:**
    1. `A` must be in every key *(trick #2)*
    2. Compute closure from smallest subset containing `A` *(trick #1)*

       $\{A\}^+ = \{ABCD\}$
    3. That's it, no need to check other subsets
       - **Why?**
         1. Must contain `A`
         2. Must not be superset of `A`
         - ❖ No other subset of `{ABCD}` satisfies these two criteria
       - Keys = `{A}`

Prime Attributes
{A}

Keys

- **Finding Keys**
  - **Exercise:** A table `R(A, B, C, D, E)` with $AB \rightarrow C$, $C \rightarrow B$, $BC \rightarrow D$ and $CD \rightarrow E$
  - **Steps:**
    1. A must be in every key *(trick #2)*
    2. Compute closure from smallest subset containing A *(trick #1)*

       $\{A\}^+ = \{A\}$ *(need to check 2 attributes)*

       $\{AB\}^+ = \{ABCDE\}$ *(found a key)*

       $\{AC\}^+ = \{ABCDE\}$ *(found a key)*

       $\{AD\}^+ = \{AD\}$

       $\{AE\}^+ = \{AE\}$ *(need to check 3 attributes but cannot be superset of AB and AC)*

       $\{ADE\}^+ = \{ADE\}$ *(no other subsets since they must be superset of AB and AC)*
    3. That's it, no need to check other subsets
       - Keys = {AB}, {AC}

Prime Attributes
{A, B, C}

Keys

- **Finding Keys**
  - **Exercise:** A table R(A, B, C, D, E, F) with $AB \rightarrow C$, $C \rightarrow B$, $BCE \rightarrow D$ and $D \rightarrow EF$
  - **Steps:**
    1. A must be in every key *(trick #2)*
    2. Compute closure from smallest subset containing A *(trick #1)*

$\{A\}^+ = \{A\}$		$\{ABC\}^+ = \{ABC\}$	
$\{AB\}^+ = \{ABC\}$		$\{ABD\}^+ = \{ABCDEF\}$	
$\{AC\}^+ = \{ABC\}$		$\{ABE\}^+ = \{ABCDEF\}$	
$\{AD\}^+ = \{ADEF\}$		$\{ACD\}^+ = \{ABCDEF\}$	
$\{AE\}^+ = \{AE\}$		$\{ACE\}^+ = \{ABCDEF\}$	
$\{AF\}^+ = \{AF\}$		$\{ADE\}^+ = \{ADEF\}$	

❖ Keys = {ABD}, {ABE}, {ACD}, {ACE}

Prime Attributes
{A, B, C, D, E}

Prime Attributes

- **Definition**
  - If an attribute appears in a key, then it is a prime attribute
  - Otherwise, it is a non-prime attribute

  - **Why?**
    - Will be used when we talk about normal forms

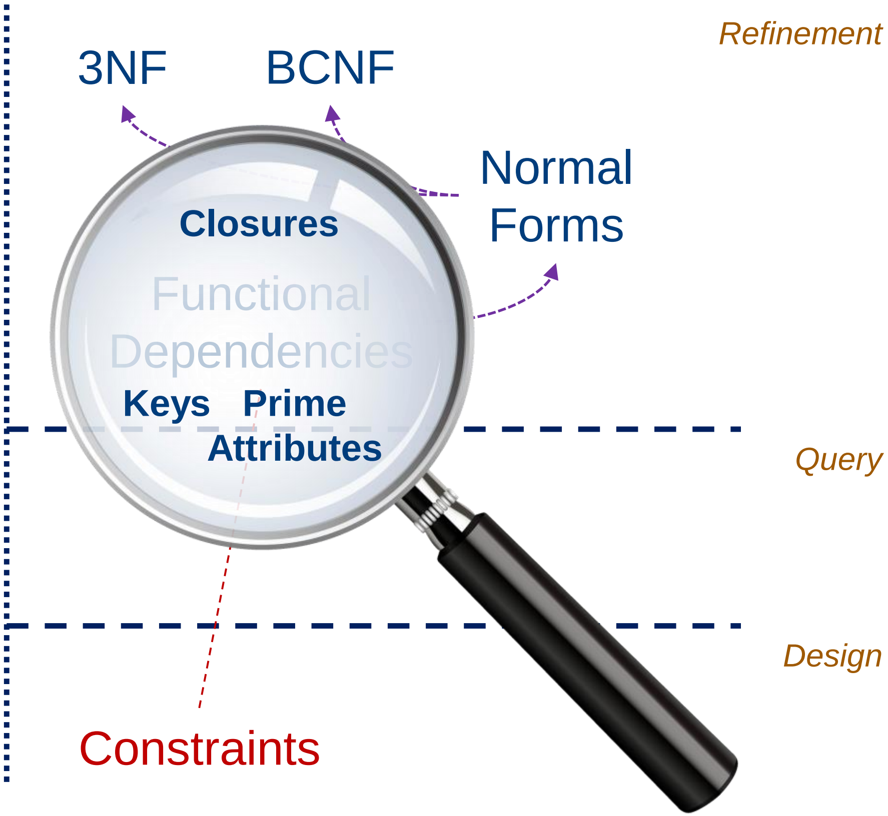
Functional Dependencies → Closures → Keys, Superkeys, Prime Attributes → **Normal Forms** → BCNF, 3NF

  - **Exercises:**
    - *Let's go back to previous slides and find the prime attribute*

See this notes in previous slides

Roadmap

- We will do this step by step
  - Functional dependencies
  - Closures
  - Keys, superkeys and prime attributes
  - **Normal forms and schema refinement** *(next week)*



Algorithm Roadmap

Closures

Input:

- Set of attributes A of R
- Set of FDs F on R

1. Initialize $\theta = A$
2. While (*there exists some FD $B \rightarrow C$ in F such that $B \subseteq \theta$ and $C \not\subseteq \theta$*)
3. $\quad \theta = \theta \cup C$
4. Return θ

Superkeys

Input:

- A table R

1. Initialize $\theta = \emptyset$
2. For every S $\subseteq$ R
3. $\quad$ If (*closure*(*S*) = *R*)
4. $\quad\quad \theta = \theta \cup$ S
5. Return θ

Keys

Input:

- A table R

1. SK = superkey(R)
2. For every sk ∈ SK
3. $\quad$ If (*there are no other sk' ∈ SK such that sk' ⊂ sk*)
4. $\quad\quad \theta = \theta \cup$ sk
5. Return θ

Prime Attributes

Input:

- A table R

1. Return ∪ keys(R)

3NF Check

BCNF Check

Non-Trivial & Decomposed FDs

BCNF Decomposition

3NF Decomposition

Redundant Attributes

Minimal Basis

3NF Synthesis

Redundant FDs

Tools

https://www.comp.nus.edu.sg/~adi-yoga/CS2102/FD/

How do I give the set of FDs to compute the closure

- Use the [Given] reasoning in Armstrong's axiom
- This will generate the basis of discussion
- Computation of closures, keys, superkeys, etc will depend on the basis of discussion PLUS the supplied set of attributes

QUESTION?