# Cơ sở dữ liệu

TS. Hồ Mạnh Tài

Khoa CNTT2

Học viện Công nghệ Bưu chính Viễn thông
2018

# Chương 4: Phụ thuộc hàm
# Functional Dependencies

# 1. Dạng chuẩn và phụ thuộc hàm
# Normal forms & functional dependencies

# Dạng chuẩn 1 (1st Normal Forms – 1NF)

| Student | Courses |
|---|---|
| Mary | {CS145,CS229} |
| Joe | {CS145,CS106} |
| … | … |

*Violates 1NF.*

| Student | Courses |
|---|---|
| Mary | CS145 |
| Mary | CS229 |
| Joe | CS145 |
| Joe | CS106 |

In 1st NF

**1NF Constraint:** Types must be atomic!

# Các ràng buộc ngăn ngừa bất thường dữ liệu

A poorly designed database causes ***anomalies***:

| Student | Course | Room |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

If every course is in only one room, contains ***redundant*** information!

# Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

| **Student** | **Course** | **Room** |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | C12 |
| Sam | CS145 | B01 |
| .. | .. | .. |

If we update the room number for one tuple, we get inconsistent data = an ***update*** **anomaly**

# Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

| Student | Course | Room |
|---|---|---|
| .. | .. | .. |

If everyone drops the class, we lose what room the class is in! = a <u>***delete* anomaly**</u>

# Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

| Student | Course | Room |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

| | | |
|---|---|---|
| ... | CS229 | C12 |

Similarly, we can't reserve a room without students = an ***insert* anomaly**

# Constraints Prevent (some) Anomalies in the Data

| Student | Course |
|---|---|
| Mary | CS145 |
| Joe | CS145 |
| Sam | CS145 |
| .. | .. |

| Course | Room |
|---|---|
| CS145 | B01 |
| CS229 | C12 |

Is this form better?

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

Today: develop theory to understand why this design may be better **and** how to find this *decomposition*…

# 2. Phụ thuộc hàm - FDs

# Định nghĩa

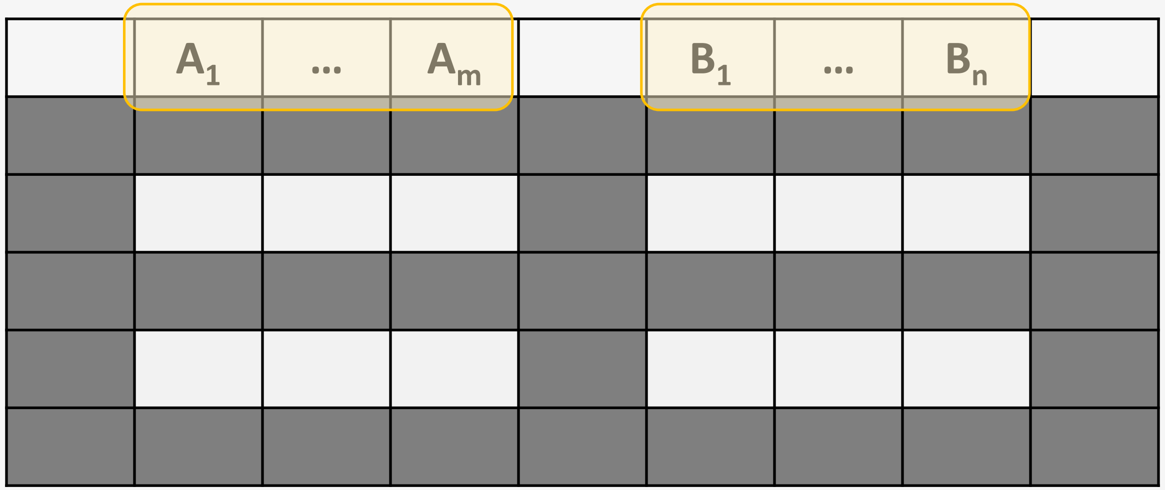
**Def:** Let A,B be *sets* of attributes
We write A → B or say A ***functionally determines*** B if, for any tuples $t_1$ and $t_2$:

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call A → B a functional dependency

*A->B means that*
*“whenever two tuples agree on A then they agree on B.”*

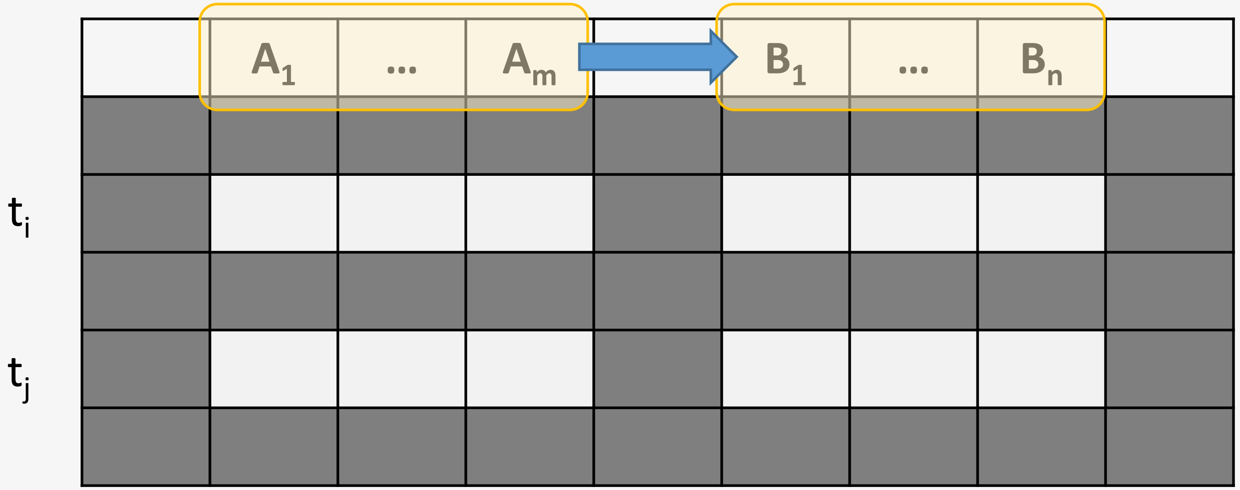
# A Picture Of FDs




Defn (again):

Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

# A Picture Of FDs

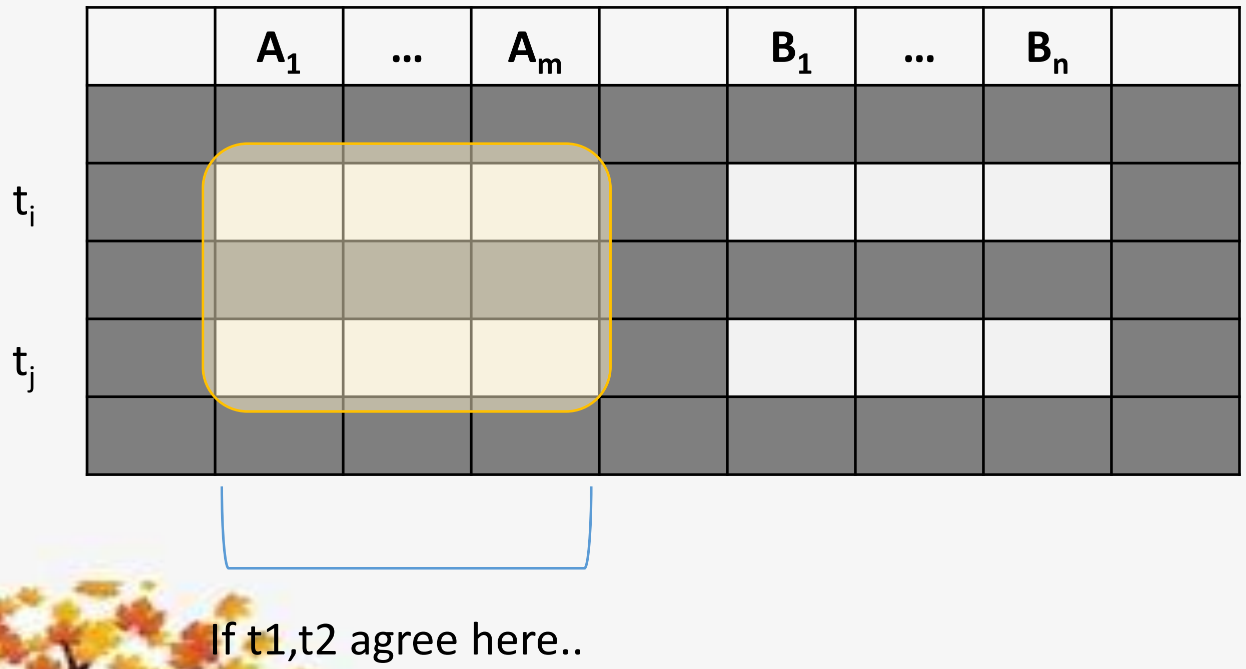



<u>Defn (again):</u>

Given attribute sets **$A=\{A_1,\ldots,A_m\}$** and **$B = \{B_1,\ldots B_n\}$** in **R**,

The ***functional dependency*** $A \rightarrow B$ on R holds if for ***any*** $t_i,t_j$ in R:

# A Picture Of FDs



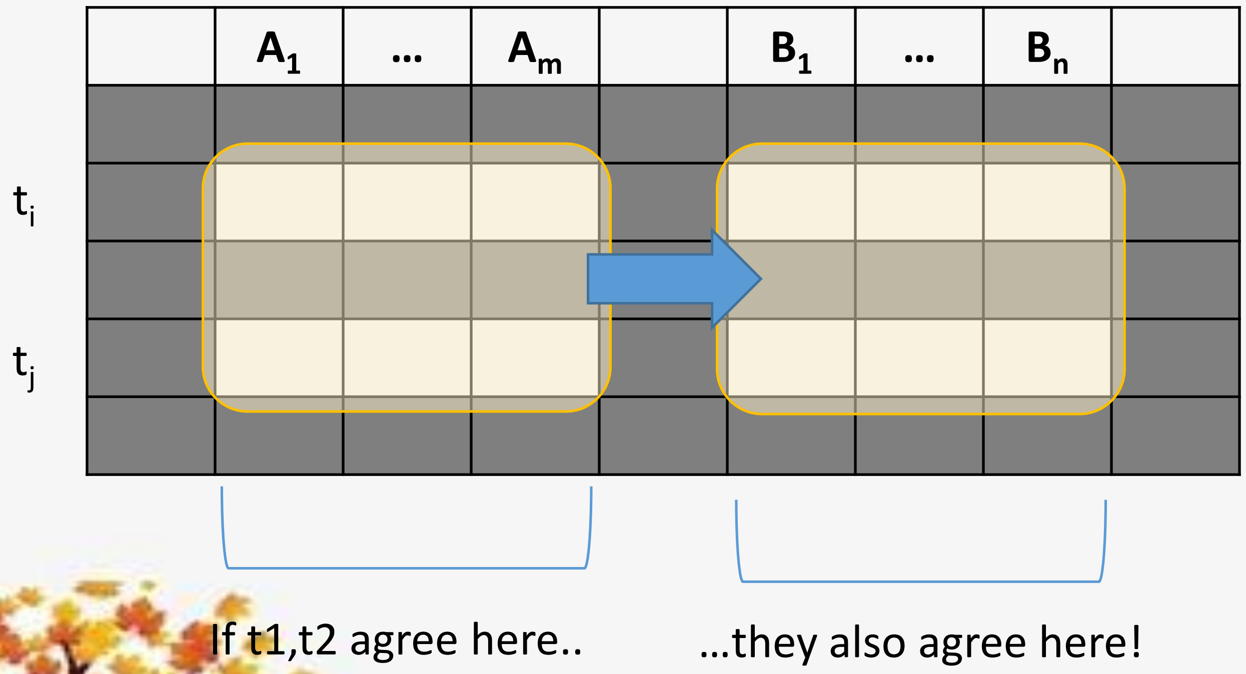

<u>Defn (again):</u>

Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

The *functional dependency* $A \rightarrow B$ on R holds if for ***any*** $t_i,t_j$ in R:

$t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

# A Picture Of FDs




<u>Defn (again):</u>

Given attribute sets $\mathbf{A=\{A_1,\ldots,A_m\}}$ and $\mathbf{B = \{B_1,\ldots B_n\}}$ in **R,**

The *functional dependency* $A \rightarrow$ **B on R** holds if for ***any*** $t_i, t_j$ in R:

<u>**if**</u> $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

<u>**then**</u> $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2]=t_j[B_2]$ AND … AND $t_i[B_n] = t_j[B_n]$

# FDs for Relational Schema Design

- High-level idea: **why do we care about FDs?**

  1. Start with some relational *schema*

  2. Find out its *functional dependencies (FDs)*

  3. Use these to *design a better schema*
     1. One which minimizes the possibility of anomalies

# Functional Dependencies as Constraints

A **functional dependency** is a form of **constraint**

- *Holds* on some instances (but not others) – can check whether there are violations
- Part of the schema, helps define a *valid* instance

| Student | Course | Room |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

Note: The FD {Course} -> {Room} ***holds on this instance***

*Recall: an **<u>instance</u>** of a schema is a multiset of tuples conforming to that schema, **i.e. a table***

# Functional Dependencies as Constraints

Note that:

- You can check if an FD is **violated** by examining a single instance;
- However, you **cannot prove** that an FD is part of the schema by examining a single instance.
  - *This would require checking every valid instance*

| Student | Course | Room |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

However, cannot *prove* that the FD {Course} -> {Room} is ***part of the schema***

# More Examples

An FD is a constraint which holds, or does not hold on an instance:

| EmpID | Name | Phone | Position |
|---|---|---|---|
| E0045 | Smith | 1234 | Clerk |
| E3542 | Mike | 9876 | Salesrep |
| E1111 | Smith | 9876 | Salesrep |
| E9999 | Mary | 1234 | Lawyer |

# More Examples

| EmpID | Name | Phone | Position |
|---|---|---|---|
| E0045 | Smith | 1234 | Clerk |
| E3542 | Mike | 9876 ← | Salesrep |
| E1111 | Smith | 9876 ← | Salesrep |
| E9999 | Mary | 1234 | Lawyer |

{Position} → {Phone}

# More Examples

| EmpID | Name | Phone | Position |
|---|---|---|---|
| E0045 | Smith | 1234 → | Clerk |
| E3542 | Mike | 9876 | Salesrep |
| E1111 | Smith | 9876 | Salesrep |
| E9999 | Mary | 1234 → | Lawyer |

but *not* {Phone} → {Position}

# 2. Tìm phụ thuộc hàm

# “Good” vs. “Bad” FDs

- We can start to develop a notion of **good** vs. **bad** FDs:

| EmpID | Name | Phone | Position |
|---|---|---|---|
| E0045 | Smith | 1234 | Clerk |
| E3542 | Mike | 9876 | Salesrep |
| E1111 | Smith | 9876 | Salesrep |
| E9999 | Mary | 1234 | Lawyer |

Intuitively:

EmpID -> Name, Phone, Position is *“good FD”*

- ***Minimal redundancy, less possibility of anomalies***

# "Good" vs. "Bad" FDs

We can start to develop a notion of **good** vs. **bad** FDs:

| EmpID | Name | Phone | Position |
|---|---|---|---|
| E0045 | Smith | 1234 | Clerk |
| E3542 | Mike | 9876 | Salesrep |
| E1111 | Smith | 9876 | Salesrep |
| E9999 | Mary | 1234 | Lawyer |

<u>Intuitively:</u>

EmpID -> Name, Phone, Position is *"good FD"*

But Position -> Phone *is a "bad FD"*

- ***Redundancy! Possibility of data anomalies***

# “Good” vs. “Bad” FDs

| Student | Course | Room |
|---|---|---|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

Returning to our original example… can you see how the “bad FD” {Course} -> {Room} could lead to an:

- Update Anomaly
- Insert Anomaly
- Delete Anomaly
- …

Given a set of FDs (from user) our goal is to:

1. Find all FDs, and
2. Eliminate the “Bad Ones".

# FDs for Relational Schema Design

- High-level idea: **why do we care about FDs?**

1. Start with some relational *schema*
2. Find out its *functional dependencies (FDs)*
3. Use these to *design a better schema*
   1. One which minimizes possibility of anomalies

*This part can be tricky!*

# Finding Functional Dependencies

Example:

Products

| Name | Color | Category | Dep | Price |
|---|---|---|---|---|
| Gizmo | Green | Gadget | Toys | 49 |
| Widget | Black | Gadget | Toys | 59 |
| Gizmo | Green | Whatsit | Garden | 99 |

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

Given the provided FDs, we can see that {Name, Category} → {Price} must also hold on **any instance**…

Which / how many other FDs do?!?

# Finding Functional Dependencies

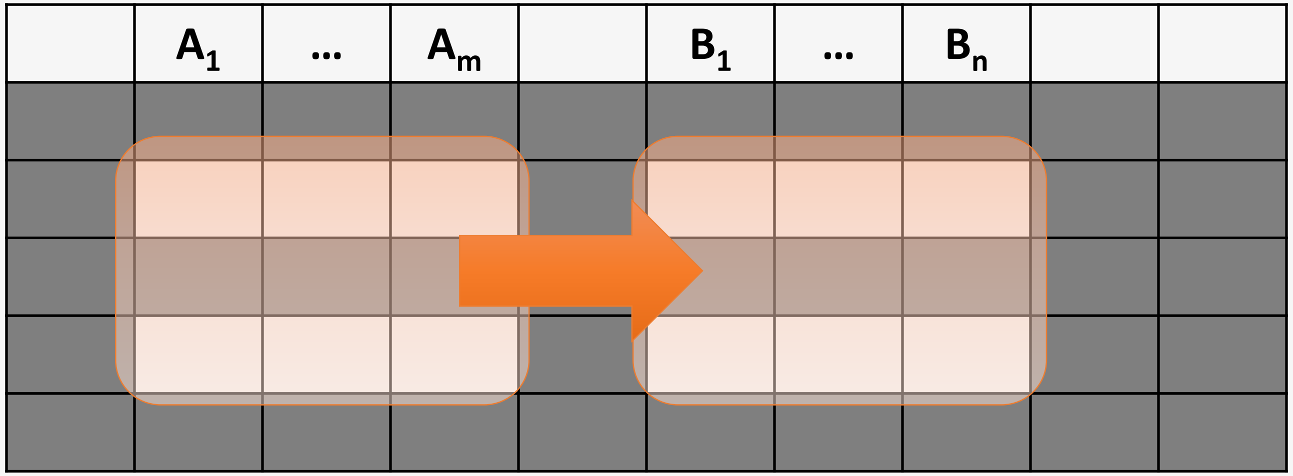
Equivalent to asking: Given a set of FDs, $F = \{f_1,\ldots f_n\}$, does an FD g hold?

**Inference problem**: How do we decide?

Answer: Three simple rules called **Armstrong's Rules.**

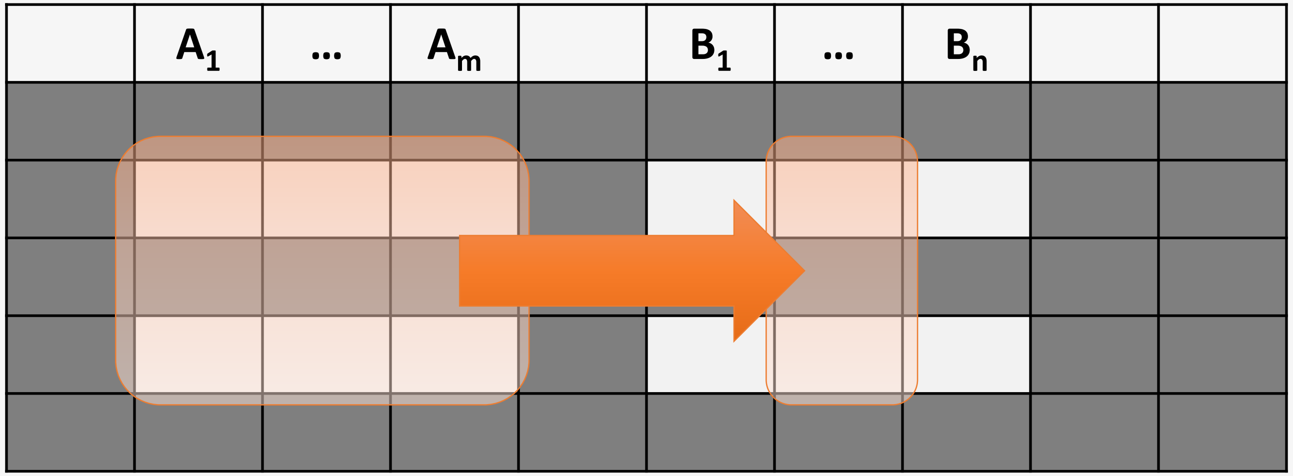
1. **Split/Combine,**
2. **Reduction, and**
3. **Transitivity…** *ideas by picture*

# 1. Split/Combine




$$A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$$

# 1. Split/Combine

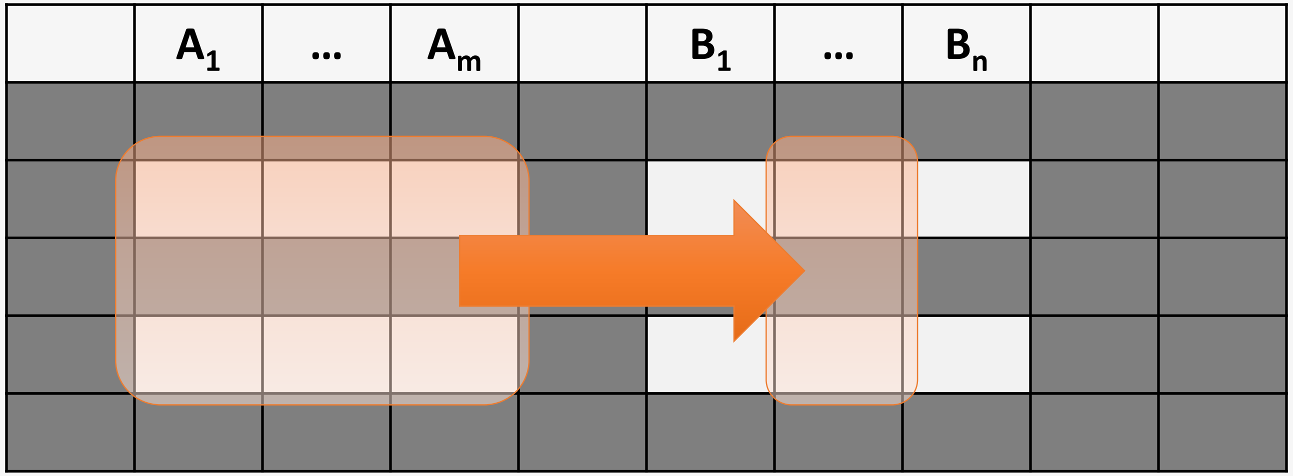



$$A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$$

… is equivalent to the following *n* FDs…

$$A_1, \ldots, A_m \rightarrow B_i \text{ for } i=1,\ldots,n$$

# 1. Split/Combine

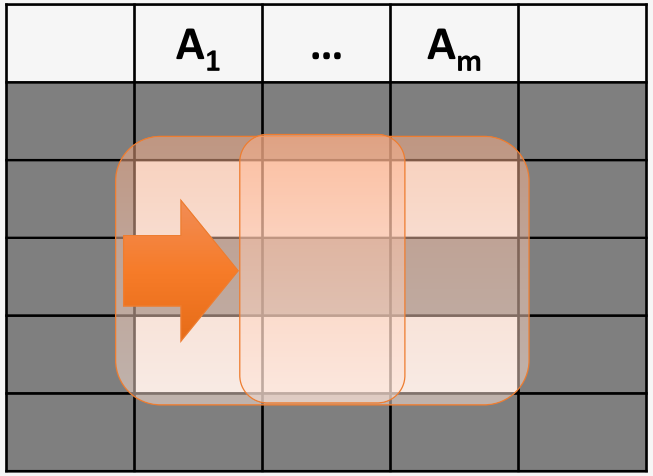



***And vice-versa,*** $A_1,\dots,A_m \rightarrow B_i$ for $i=1,\dots,n$

… is equivalent to …

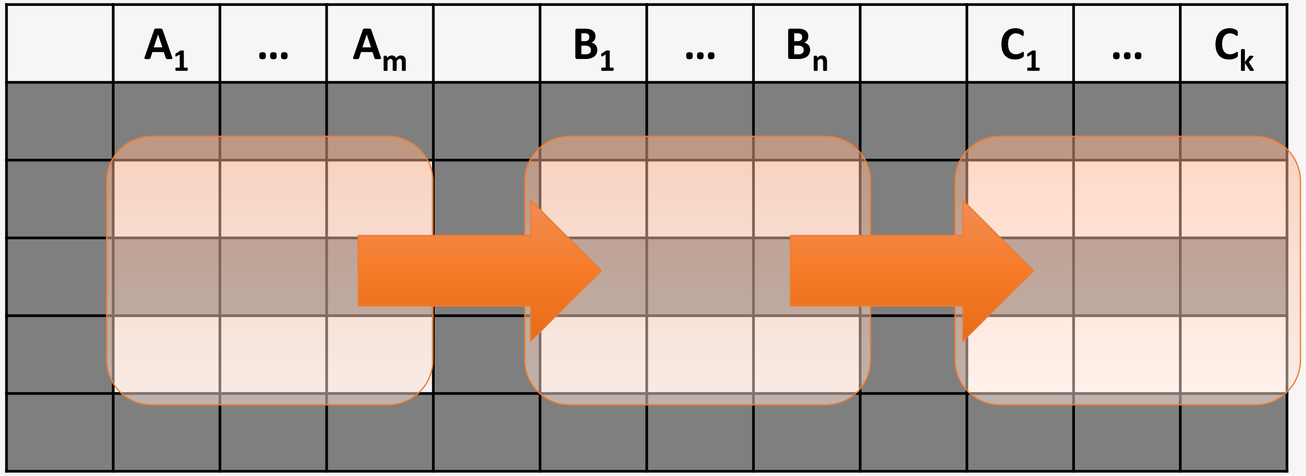
$A_1, \dots, A_m \rightarrow B_1,\dots,B_n$

# 2. Reduction/Trivial



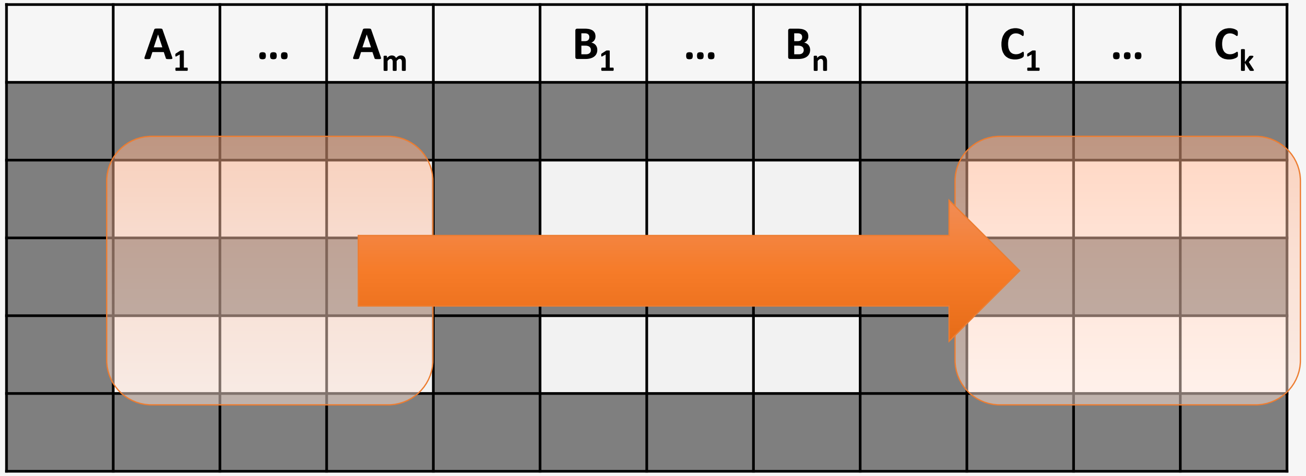

$A_1,\ldots,A_m \rightarrow A_j$ for any $j=1,\ldots,m$

# 3. Transitive Closure




$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and
$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

# 3. Transitive Closure




$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and
$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

implies

$A_1,\ldots,A_m \rightarrow C_1,\ldots,C_k$

# Finding Functional Dependencies

Example:

Products

| Name | Color | Category | Dep | Price |
|---|---|---|---|---|
| Gizmo | Green | Gadget | Toys | 49 |
| Widget | Black | Gadget | Toys | 59 |
| Gizmo | Green | Whatsit | Garden | 99 |

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

# Finding Functional Dependencies

<u>Example:</u>

**Inferred FDs:**

| Inferred FD | Rule used |
|---|---|
| 4. {Name, Category} -> {Name} | ? |
| 5. {Name, Category} -> {Color} | ? |
| 6. {Name, Category} -> {Category} | ? |
| 7. {Name, Category -> {Color, Category} | ? |
| 8. {Name, Category} -> {Price} | ? |

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

# Finding Functional Dependencies

Example:

**Inferred FDs:**

| Inferred FD | Rule used |
|---|---|
| 4. {Name, Category} -> {Name} | Trivial |
| 5. {Name, Category} -> {Color} | Transitive (4 -> 1) |
| 6. {Name, Category} -> {Category} | Trivial |
| 7. {Name, Category -> {Color, Category} | Split/combine (5 + 6) |
| 8. {Name, Category} -> {Price} | Transitive (7 -> 3) |

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Can we find an algorithmic way to do this?

# Bao đóng - Closures

# Closure of a set of Attributes

**Given** a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F:**
**Then** the <u>**closure**</u>, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

<u>Example:</u> F =

{name} → {color}
{category} → {department}
{color, category} → {price}

*Example Closures:*

$\{\text{name}\}^+$ = {name, color}
$\{\text{name, category}\}^+$ =
{name, category, color, dept, price}
$\{\text{color}\}^+$ = {color}

# Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$ and set of FDs F.

**Repeat until** X doesn't change; **do**:

**if** $\{B_1, \ldots, B_n\} \rightarrow C$ is entailed by F

**and** $\{B_1, \ldots, B_n\} \subseteq X$

**then** add C to X.

**Return** X as $X^+$

# Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
**Repeat until** X doesn't change; **do**:
  **if** $\{B_1, \ldots, B_n\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_n\} \subseteq X$:
    **then** add C to X.
**Return** X as $X^+$

{name, category}$^+$ =
{name, category}

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

# Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
**Repeat until** X doesn't change; **do**:
**if** $\{B_1, \ldots, B_n\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_n\} \subseteq X$:
**then** add C to X.
**Return** X as $X^+$

$\{name, category\}^+ =$
$\{name, category\}$

$\{name, category\}^+ =$
$\{name, category, color\}$

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

# Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
**Repeat until** X doesn't change; **do**:
**if** $\{B_1, \ldots, B_n\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_n\} \subseteq X$:
**then** add C to X.
**Return** X as $X^+$

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

$\{\text{name, category}\}^+ =$ {name, category}

$\{\text{name, category}\}^+ =$ {name, category, color}

$\{\text{name, category}\}^+ =$ {name, category, color, dept}

# Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
**Repeat until** X doesn't change; **do**:
**if** $\{B_1, \ldots, B_n\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_n\} \subseteq X$:
**then** add C to X.
**Return** X as $X^+$

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

$\{\text{name, category}\}^+ =$
{name, category}

$\{\text{name, category}\}^+ =$
{name, category, color}

$\{\text{name, category}\}^+ =$
{name, category, color, dept}

$\{\text{name, category}\}^+ =$
{name, category, color, dept, price}

# Example

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^+ = \{A, B, \qquad \}$

Compute $\{A, F\}^+ = \{A, F, \qquad \}$

# Example

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^{+} = \{A, B, C, D \qquad\qquad \}$

Compute $\{A, F\}^{+} = \{A, F, B \qquad\qquad \}$

# Example

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Compute $\{A, F\}^+ = \{A, B, C, D, E, F\}$

# 3. Closures, Superkeys & Keys

# Why Do We Need the Closure?

- With closure we can find all FD's easily

- To check if $X \to A$

  1. Compute $X^+$

  2. Check if $A \in X^+$

Note here that **X** is a *set* of attributes, but **A** is a *single* attribute. Why does considering FDs of this form suffice?

Recall the <u>**Split/combine**</u> rule:
$X \rightarrow A_1, \ldots, X \rightarrow A_n$
*implies*
$X \rightarrow \{A_1, \ldots, A_n\}$

# Using Closure to Infer ALL FDs

**Example:**

Given F =

$\{A,B\} \to C$
$\{A,D\} \to B$
$\{B\} \to D$

Step 1: Compute $X^+$, for every set of attributes X:

$\{A\}^+ = \{A\}$
$\{B\}^+ = \{B,D\}$
$\{C\}^+ = \{C\}$
$\{D\}^+ = \{D\}$
$\{A,B\}^+ = \{A,B,C,D\}$
$\{A,C\}^+ = \{A,C\}$
$\{A,D\}^+ = \{A,B,C,D\}$
$\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$
$\{B,C,D\}^+ = \{B,C,D\}$
$\{A,B,C,D\}^+ = \{A,B,C,D\}$

No need to compute all of these- why?

# Using Closure to Infer ALL FDs

Example:

Given F =

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

Step 1: Compute $X^+$, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \varnothing$:

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

# Using Closure to Infer ALL FDs

**Example:**

Given F =

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

Step 1: Compute $X^+$, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

*"Y is in the closure of X"*

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

# Using Closure to Infer ALL FDs

Example:

Given F = 

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

Step 1: Compute $X^+$, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

*The FD $X \rightarrow Y$ is non-trivial*

# Superkeys and Keys

# Keys and Superkeys

A <u>superkey</u> is a set of attributes $A_1, \ldots, A_n$ s.t. for *any other* attribute B in R,
we have $\{A_1, \ldots, A_n\} \rightarrow B$

I.e. all attributes are *functionally determined* by a superkey

A <u>key</u> is a *minimal* superkey

This means that no subset of a key is also a superkey (i.e., dropping any attribute from the key makes it no longer a superkey)

# Finding Keys and Superkeys

- For each set of attributes X

    1. Compute $X^+$

    2. If $X^+$ = set of all attributes then X is a **superkey**

    3. If X is minimal, then it is a **key**

# Example of Finding Keys

Product(name, price, category, color)

{name, category} → price
{category} → color

What is a key?

# Example of Keys

Product(name, price, category, color)

{name, category} → price
{category} → color

$\{name, category\}^+$ = {name, price, category, color}
= the set of all attributes
⟹ this is a **superkey**
⟹ this is a **key**, since neither name nor category alone is a superkey