INFORMATION SYSTEM DEPARTMENT
INFORMATION TECHNOLOGY FACULTY
UNIVERSITY OF SCIENCE – VNUHCM

ADVANCED DATABASE

Chapter 06

FUNCTIONAL DEPENDENCY & NORMALIZATION

Dr. **VU** Thi My Hang

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

Recall:

Relational Model

(Mô hình quan hệ)

fit@hcmus

Recall: Relational Model

(Mô hình quan hệ)

- The relational data model represents data in the form of relations based on mathematical foundation.
- A relation is a **named**, two-dimensional table of data, with rows (tuples) and **named** columns (attributes).
  - Relation ⇔ Table.
  - Tuples in a relation ⇔ Rows of a table.
  - Attributes of a relation ⇔ Columns of a table.

The relation **Employee** with four attributes (columns) and six tuples (rows).

Employee

staffNo	sName	position	salary
SL21	John White	Manager	30000
SG37	Ann Beech	Assistant	12000
SG14	David Ford	Supervisor	18000
SA9	Mary Howe	Assistant	9000
SG5	Susan Brand	Manager	24000
SL41	Julie Lee	Assistant	9000

Recall: Relational Schema

(Lược đồ CSDL quan hệ)

- A relational (database) schema is a set of table definitions and integrity constraints for organizing data in relations.

(Ràng buộc toàn vẹn)

- Integrity constraints specify (business) rules to maintain the accuracy and integrity of data stored in relations.
  - **Domain Constraints.** All values appear in a column of a relation must be from the same domain.
    - Domain: Set of values that may be assigned to an attributes.
    - **Domain definition**: Domain name, meaning, data type and size.
  - **Entity Integrity.** A relation must have a **non-null primary key**.
  - **Referential Integrity**. A **foreign key** value must match a primary key value in another relation or must be null to maintain consistency among the rows of the two relations.

cdio 4.0

fit@hcmus

Recall: Relational Schema

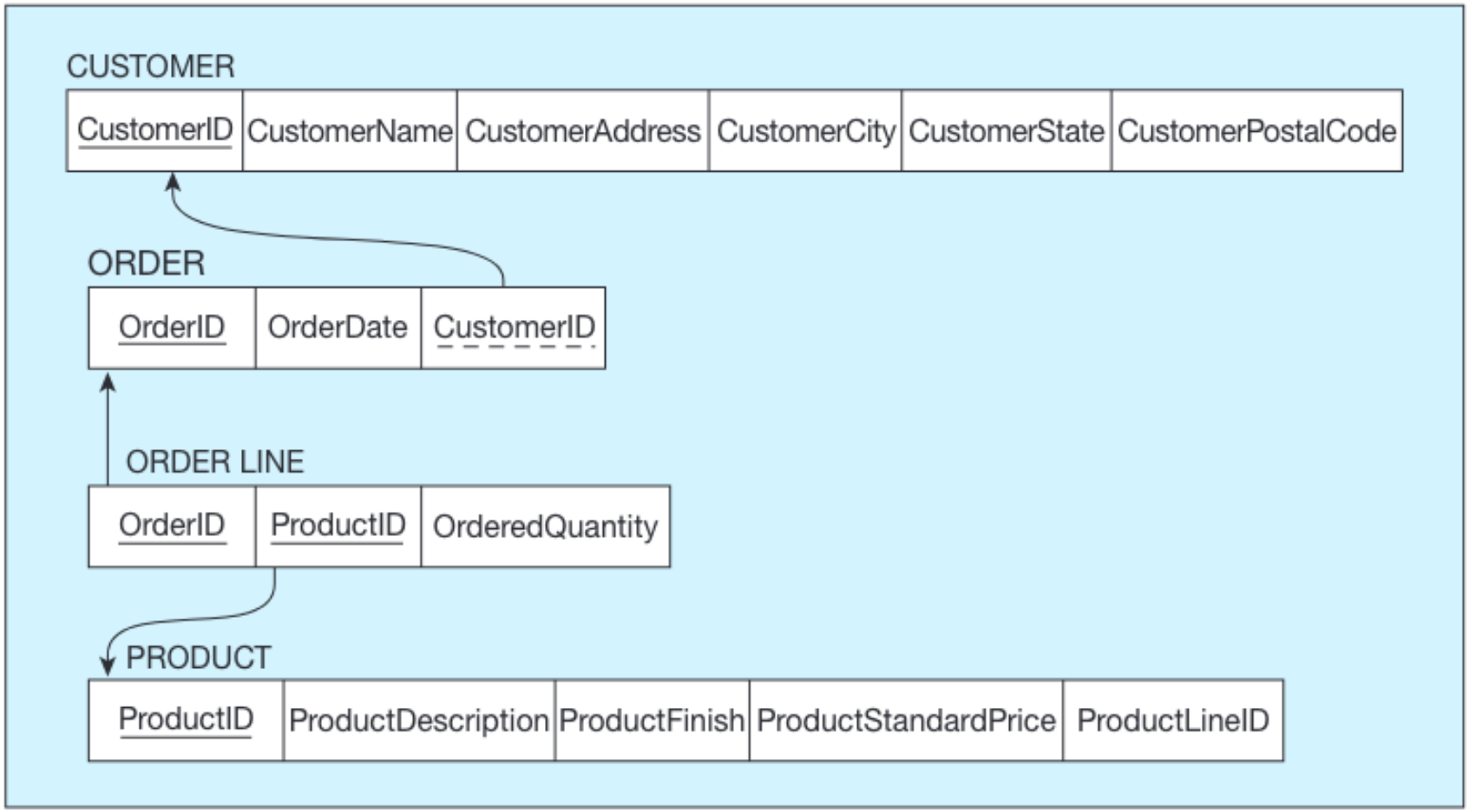
Example of domain definition

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Recall: Relational Schema

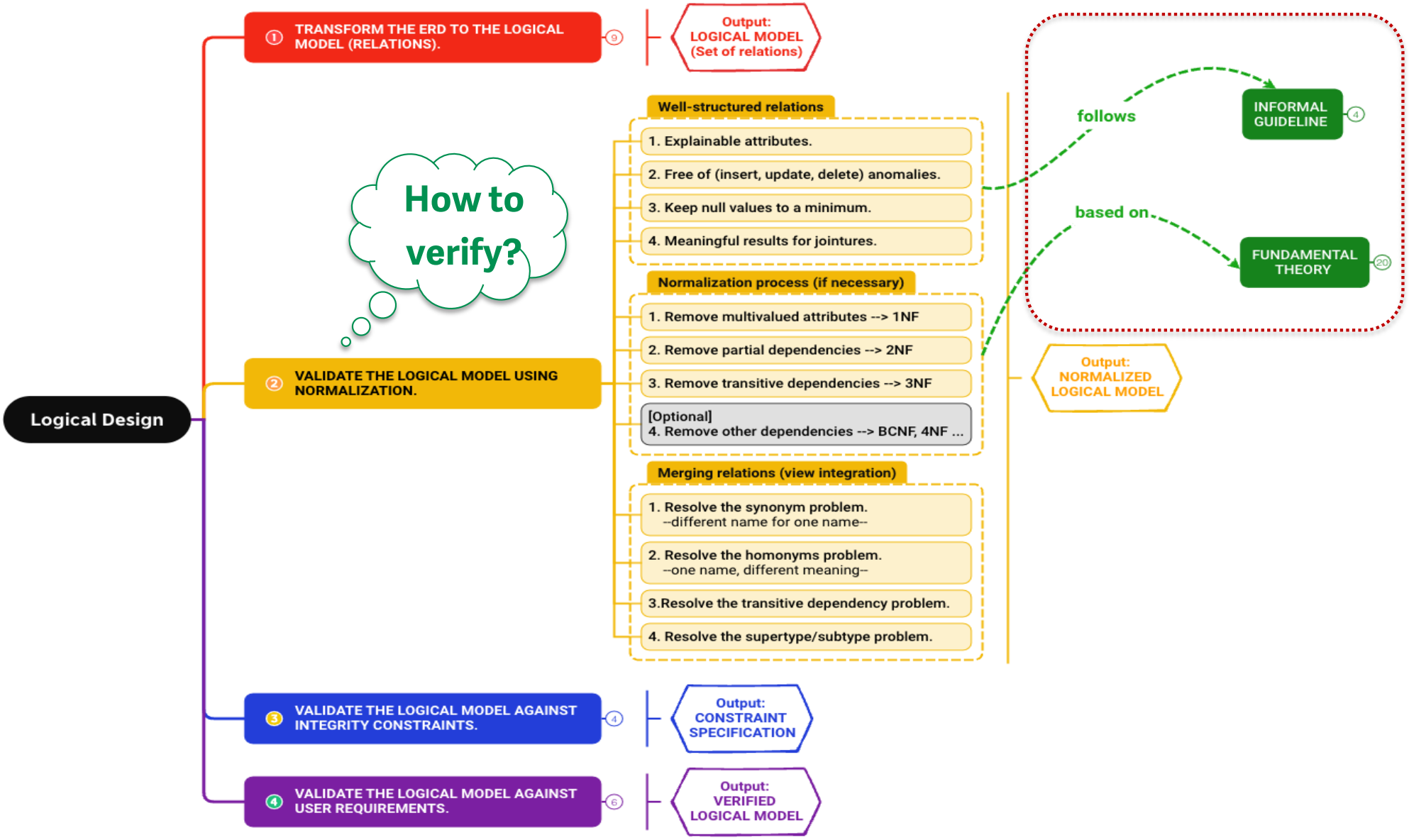
Example of primary and foreign keys



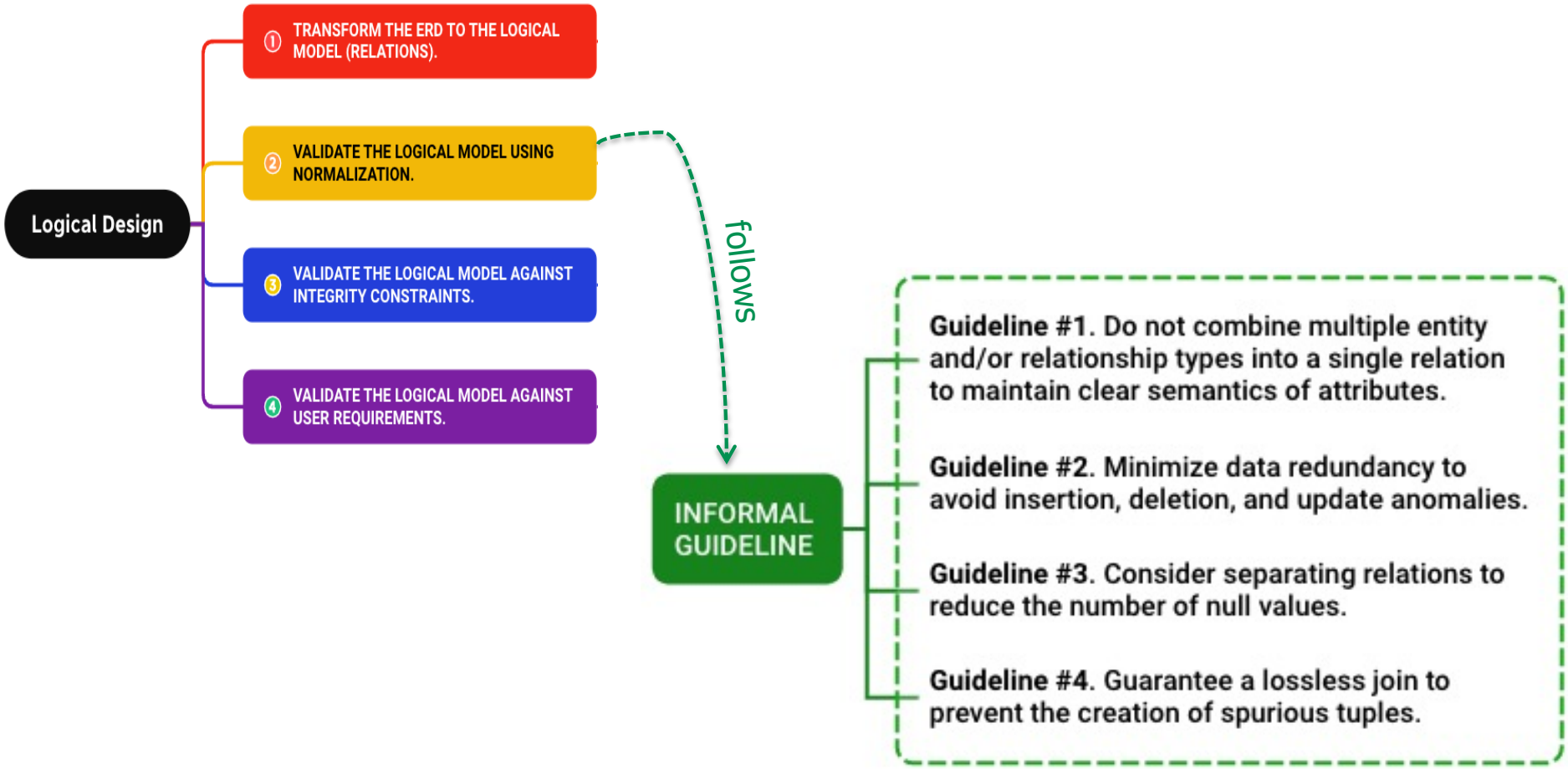
Recall:

Logical Design Process

Logical Design Steps

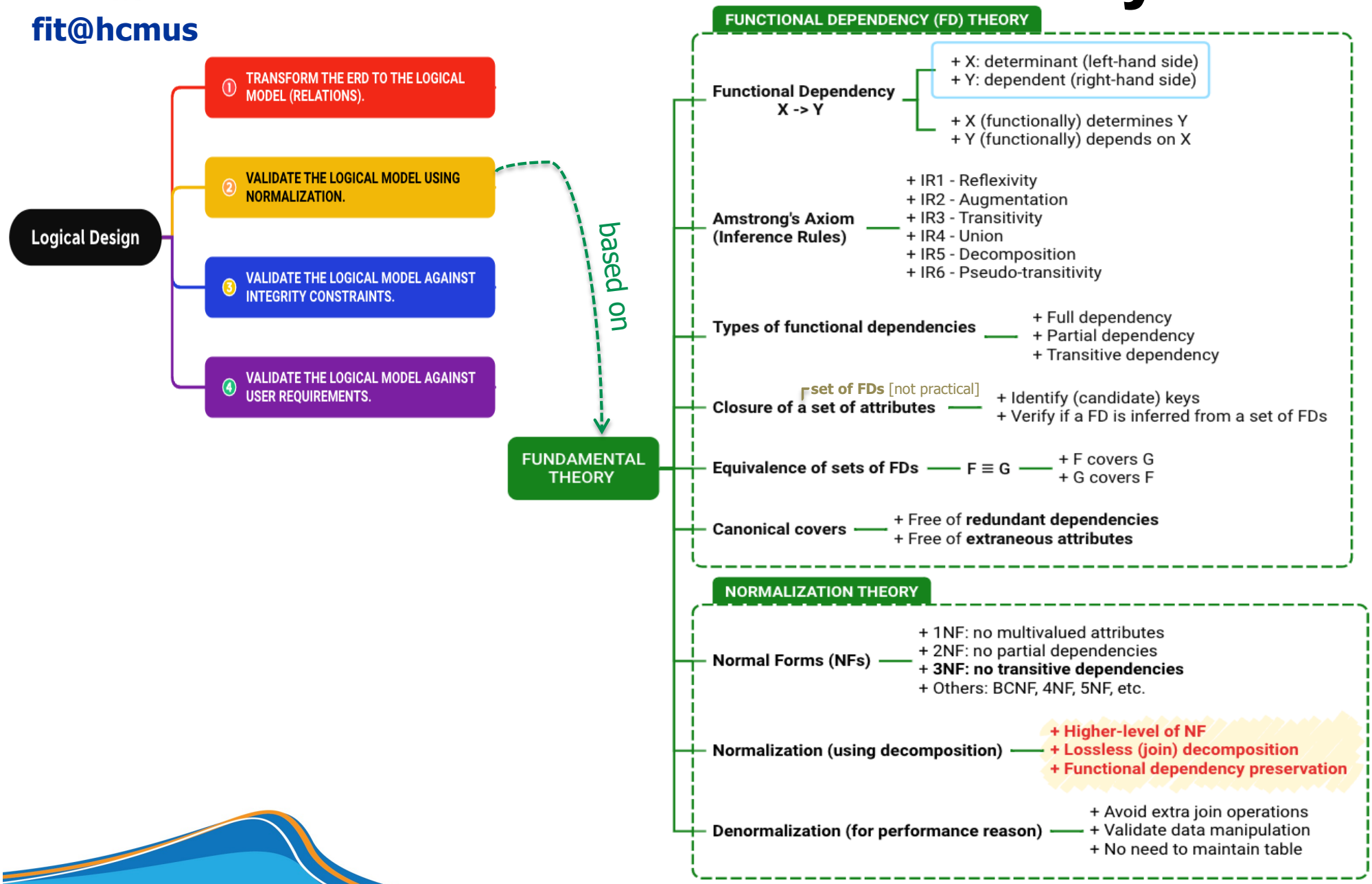


Informal Guideline



Fundamental Theory

fit@hcmus



cdio 4.0

fit@hcmus

Validate?

How to verify?

Well-structured relations

1. Explainable attributes.
2. Free of (insert, update, delete) anomalies.
3. Keep null values to a minimum.
4. Meaningful results for jointures.

follows some experiences

Informal Design Guideline

- **Guideline #1**. Do not combine multiple entity and/or relationship types into a single relation to maintain clear semantics of attributes.
- **Guideline #2**. Minimize data redundancy to avoid insertion, deletion, and update anomalies.
- **Guideline #3**. Consider separating relations to reduce the number of null values.
- **Guideline #4**. Guarantee a lossless join to prevent the creation of spurious tuples.

based on fundamental theory

NORMALIZATION THEORY

- **Normal Forms (NFs)**
  - + 1NF: no multivalued attributes
  - + 2NF: no partial dependencies
  - + **3NF: no transitive dependencies**
  - + Others: BCNF, 4NF, 5NF, etc.
- **Normalization (using decomposition)**
  - + Higher-level of NF
  - + Lossless (join) decomposition
  - + Functional dependency preservation
- **Denormalization (for performance reason)**
  - + Avoid extra join operations
  - + Validate data manipulation
  - + No need to maintain table

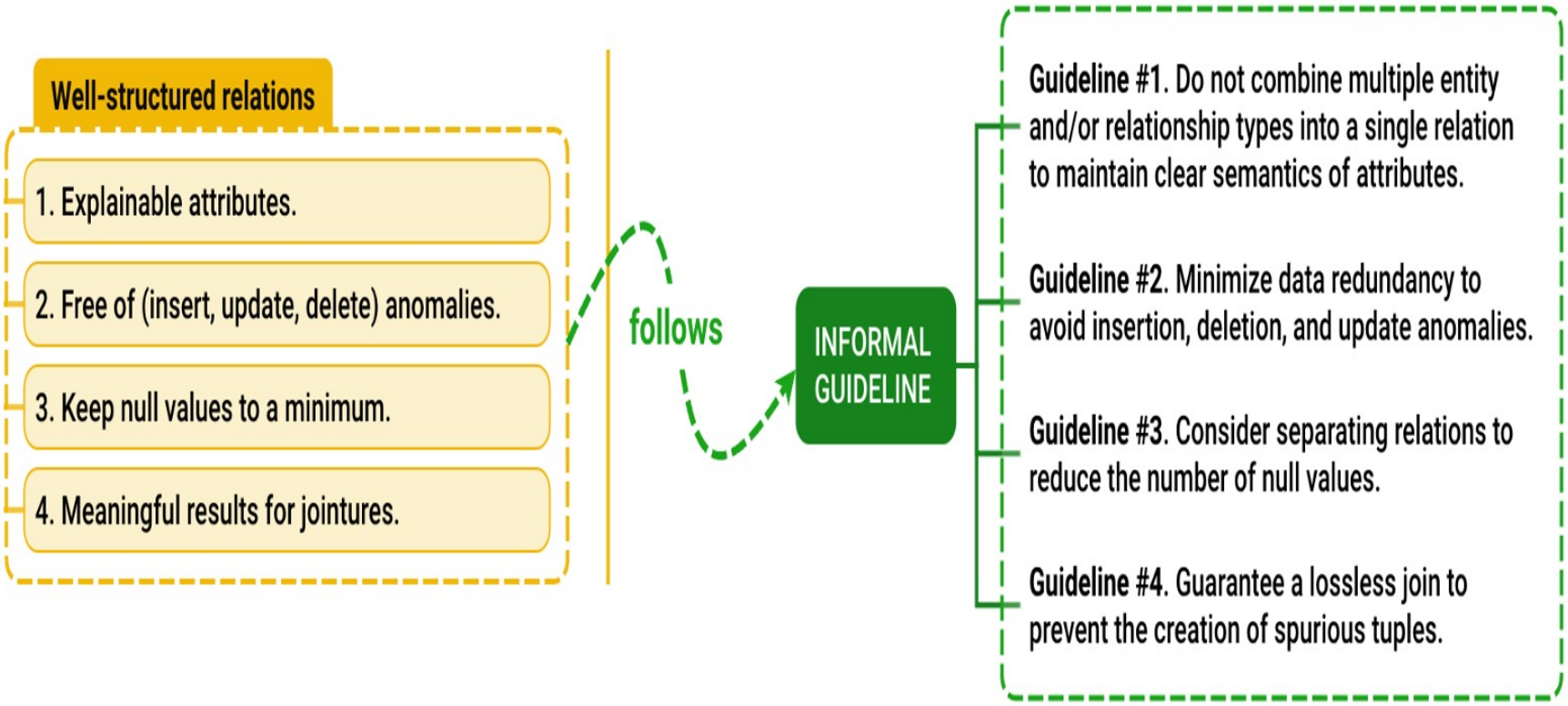
FUNCTIONAL DEPENDENCY (FD) THEORY

- **Functional Dependency X -> Y**
  - + X: determinant (left-hand side)
  - + Y: dependent (right-hand side)
  - + X (functionally) determines Y
  - + Y (functionally) depends on X
- **Amstrong's Axiom (Inference Rules)**
  - + IR1 - Reflexivity
  - + IR2 - Augmentation
  - + IR3 - Transitivity
  - + IR4 - Union
  - + IR5 - Decomposition
  - + IR6 - Pseudo-transitivity
- **Types of functional dependencies**
  - + Full dependency
  - + Partial dependency
  - + Transitive dependency
- **Closure of a set of attributes**
  - + Identify (candidate) keys
  - + Verify if a FD is inferred from a set of FDs
- **Equivalence of sets of FDs** — F ≡ G
  - + F covers G
  - + G covers F
- **Canonical covers**
  - + Free of **redundant dependencies**
  - + Free of **extraneous attributes**

(Coronel and Morris, 2015) (Hoffer et al., 2019)

A. Informal Design Guideline

A. Informal Design Guidelines



Outline

- What is a “good” database structure?
- Design Guidelines for “Good” Relational Schemas

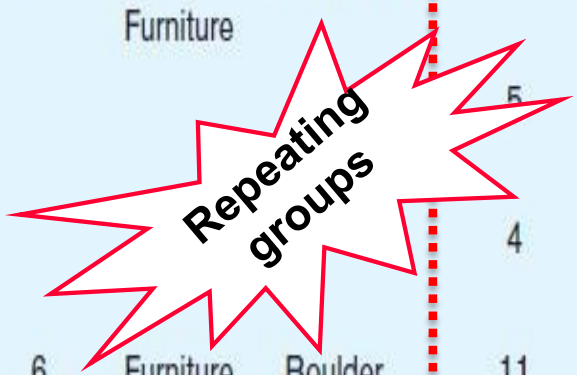
Outline

- What is a “good” database structure?
- Design Guidelines for “Good” Relational Schemas

Well-Structured Relations?

- Is this a relation?
  No, a table with repeating groups (multivalued attributes).
- What's the primary key? Indeterminable with clarity.

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3



Well-Structured Relations?

- Is this a relation?
  Yes, unique rows with no repeating groups.
- What's the primary key? OrderID, ProductID.
- Well-structured? No, data redundancy/duplication exists.

Redundancy

Order ID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Well-Structured Relations?

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

- What if a new product is added to the order 1007?
- What if the Dining Table is deleted from the order 1006?
- What if the price of the product 4 is modified?

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Well-Structured Relations?

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

- What if a new product is added to the order 1007?

➔ Customer data must be re-entered (insertion anomaly).

- What if the Dining Table is deleted from the order 1006?

➔ Product finish & price are also deleted (deletion anomaly).

- What if the price of the product 4 is modified?

➔ Multi-rows must be updated (update anomaly).

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

cdio 4.0

fit@hcmus

Well-Structured Relations?

Relation: **Yes**, Primary key: **MAGV**

MAGV	TENGV	NGSINH	DCHI	MABM	TENBM	TRGBM
009	Tiên	11/02/1960	119 Cống Quỳnh, Tp HCM	5	HTTT	005
005	Tùng	20/08/1962	222 Nguyễn Văn Cừ, Tp HCM	5	HTTT	005
007	Hằng	11/3/1954	332 Nguyễn Thái Học, Tp HCM	4	MMT	008
001	Như	01/02/1967	291 Hồ Văn Huê, Tp HCM	4	MMT	008
004	Hùng	04/03/1967	95 Bà Rịa, Vũng Tàu	5	HTTT	005
003	Tâm	04/05/1957	34 Mai Thị Lự, Tp HCM	5	HTTT	005
008	Quang	01/09/1967	80 Lê Hồng Phong, Tp HCM	4	MMT	008
006	Vinh	01/01/1965	45 Trưng Vương, Hà Nội	1	CNPM	006

- Update the head of the department (**TRGBM**) **HTTT**? **Update anomaly**
- Add a new department **KHMT**? **Insertion anomaly**
- Delete the lecturer **006**? **Deletion anomaly**

(Coronel and Morris, 2015) (Hoffer et al., 2019)

cdio 4.0

fit@hcmus

Well-Structured Relations?

Insertion Anomaly: Adding new rows forces users to create duplicate data.

Deletion Anomaly: Deleting rows may cause a loss of data that would be needed for other future rows.

Modification Anomaly: Changing data in a row forces changes to other rows due to duplication.

General Rule:

A relation should contain minimal data redundancy and pertain one type of entity/relationship types to ensure:

- Explainable attributes.
- Free of insertion, deletion, and update anomalies.
- Minimal numbers of null values.
- Meaningful results of jointure.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Outline

- What is a “good” database structure?
- Design Guidelines for “Good” Relational Schemas

cdio 4.0

fit@hcmus

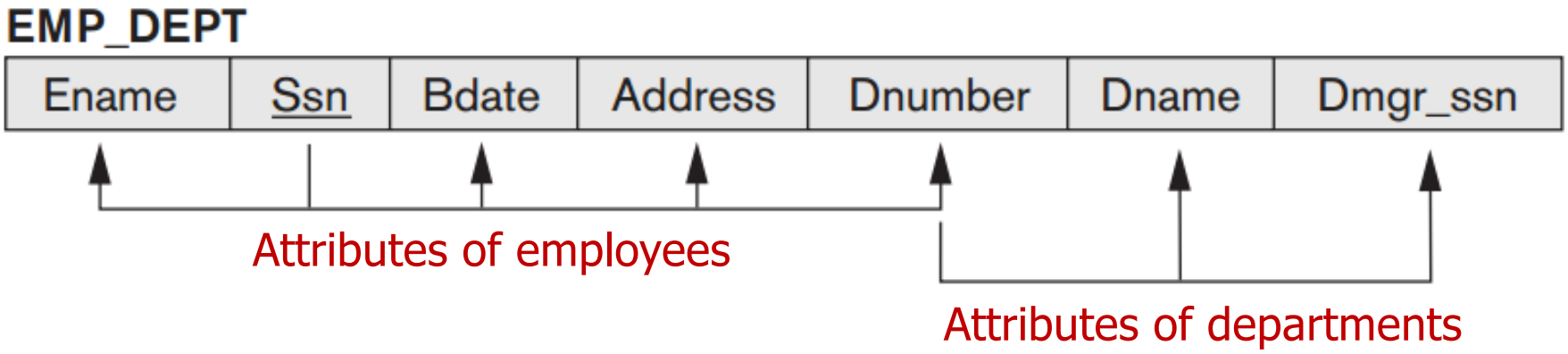
Semantics of Attributes

Guideline #1

Design a relation schema so that it is easy to explain its meaning (clear semantics of attributes).

→ Do not combine attributes from multiple entity types and/or relationship types into a single relation.

- Each EMPLOYEE tuple represents an employee.
- Each DEPARTMENT tuple represents a department.



(Coronel and Morris, 2015) (Hoffer et al., 2019)

fit@hcmus

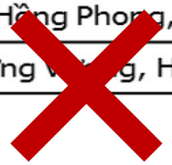
Redundant Information

Guideline #2

Design a relation schema so that no insertion, deletion, or update anomalies exist. If any anomalies are present, document them clearly and make sure that database modifications will be executed properly.

→ Minimalize data redundancy to avoid anomalies.

MAGV	TENGV	NGSINH	DCHI	MABM	TENBM	TRGBM
009	Tiên	11/02/1960	119 Cống Quỳnh, Tp HCM	5	HTTT	005
005	Tùng	20/08/1962	222 Nguyễn Văn Cừ, Tp HCM	5	HTTT	005
007	Hằng	11/3/1954	332 Nguyễn Thái Học, Tp HCM	4	MMT	008
001	Như	01/02/1967	291 Hồ Văn Huê, Tp HCM	4	MMT	008
004	Hùng	04/03/1967	95 Bà Rịa, Vũng Tàu	5	HTTT	005
003	Tâm	04/05/1957	34 Mai Thị Lự, Tp HCM	5	HTTT	005
008	Quang	01/09/1967	80 Lê Hồng Phong, Tp HCM	4	MMT	008
006	Vinh	01/01/1965	45 Trưng Vương, Hà Nội	1	CNPM	006



(Coronel and Morris, 2015) (Hoffer et al., 2019)

NULL Values

Guideline #3

Design a relation schema so that tuples have minimal null values as much as possible.

→ Consider separating relations to reduce the number of null values.

- Attributes that are NULL frequently could be placed in separate relations (with the primary key).
- Setting a NULL value is appropriate WHEN the actual value is unknown, absent or not applicable.

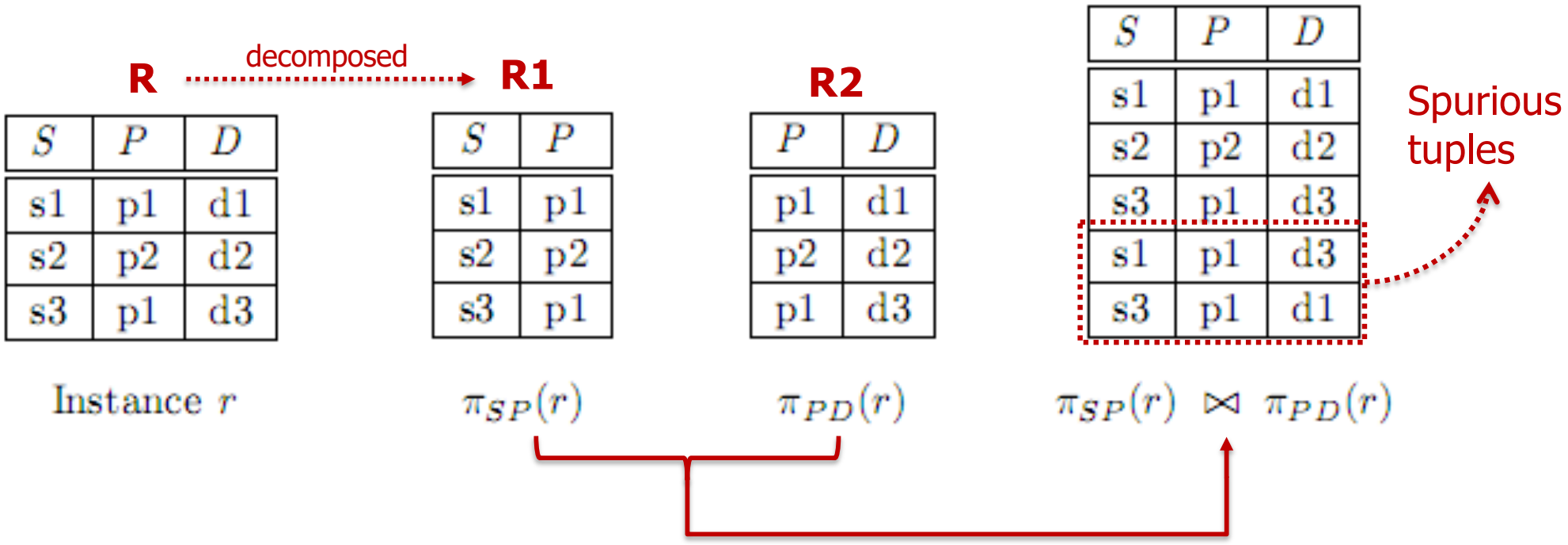
(Coronel and Morris, 2015) (Hoffer et al., 2019)

Spurious Tuples

Guideline #4

(Bộ dữ liệu giả)

- **Spurious tuples**: Additional tuples that were not in the original relation formed by linking the two tables incorrectly.



S	P	D
s1	p1	d1
s2	p2	d2
s3	p1	d3

S	P
s1	p1
s2	p2
s3	p1

P	D
p1	d1
p2	d2
p1	d3

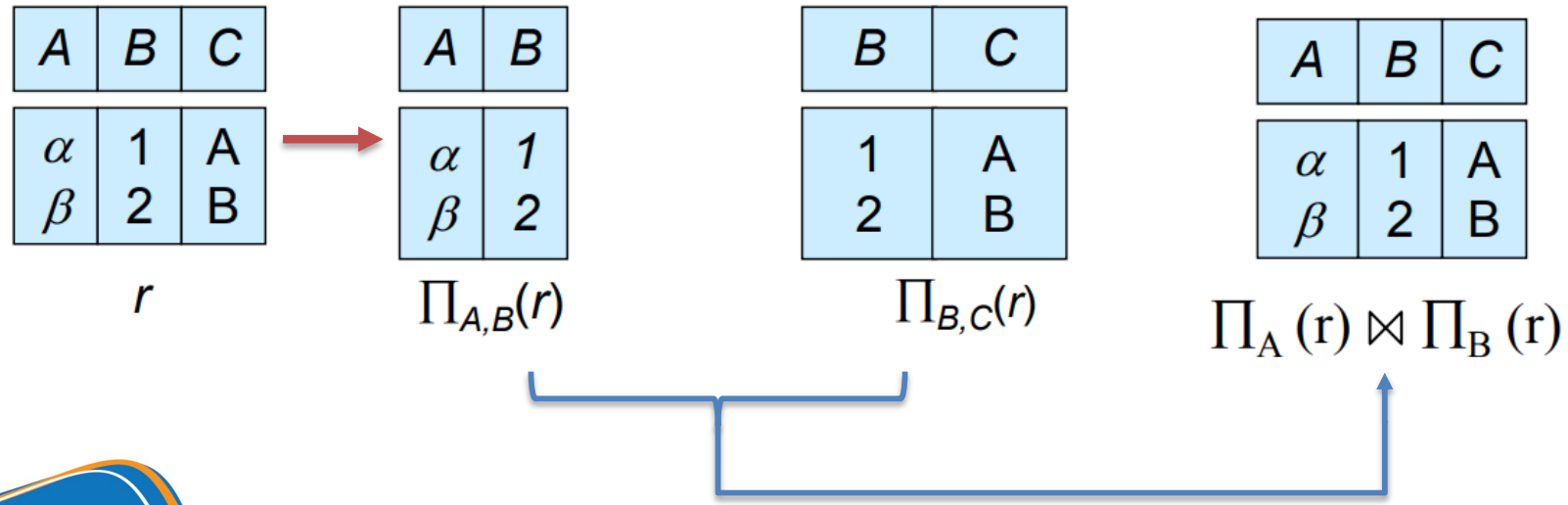
S	P	D
s1	p1	d1
s2	p2	d2
s3	p1	d3
s1	p1	d3
s3	p1	d1

(Coronel and Morris, 2015) (Hoffer et al., 2019)

Spurious Tuples

Guideline #4

- The "**Lossless join**" property is used to guarantee meaningful results for join operations.
- **Lossless-join decomposition** means splitting a table in a way so that we do not lose information (i.e., able to reconstruct the original table from the decomposed table using natural joins).



(Coronel and Morris, 2015) (Hoffer et al., 2019)

Spurious Tuples

Guideline #4

Design a relation schema so that no spurious tuples should be generated using natural joins of any relations

→ Guarantee a lossless join to prevent the creation of spurious tuples.

(Coronel and Morris, 2015) (Hoffer et al., 2019)

cdio 4.0

fit@hcmus

Summary

How to verify?

Well-structured relations

1. Explainable attributes.
2. Free of (insert, update, delete) anomalies.
3. Keep null values to a minimum.
4. Meaningful results for jointures.

follows some experiences

Informal Design Guideline

- **Guideline #1**. Do not combine multiple entity and/or relationship types into a single relation to maintain clear semantics of attributes.
- **Guideline #2**. Minimize data redundancy to avoid insertion, deletion, and update anomalies.
- **Guideline #3**. Consider separating relations to reduce the number of null values.
- **Guideline #4**. Guarantee a lossless join to prevent the creation of spurious tuples.

based on fundamental theory

NORMALIZATION THEORY

- **Normal Forms (NFs)**
  - + 1NF: no multivalued attributes
  - + 2NF: no partial dependencies
  - + **3NF: no transitive dependencies**
  - + Others: BCNF, 4NF, 5NF, etc.
- **Normalization (using decomposition)**
  - + Higher-level of NF
  - + Lossless (join) decomposition
  - + Functional dependency preservation
- **Denormalization (for performance reason)**
  - + Avoid extra join operations
  - + Validate data manipulation
  - + No need to maintain table

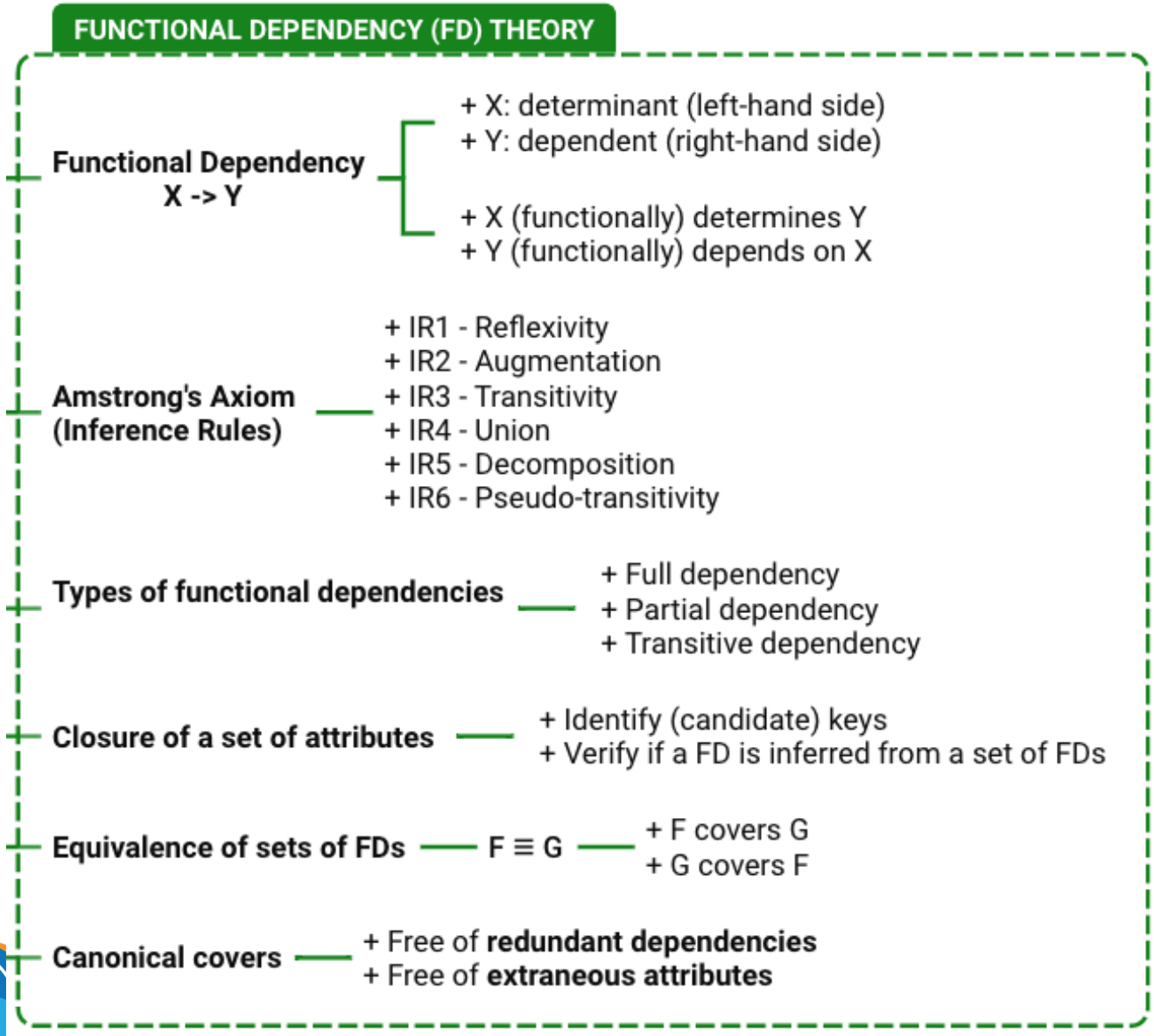
FUNCTIONAL DEPENDENCY (FD) THEORY

- **Functional Dependency X -> Y**
  - + X: determinant (left-hand side)
  - + Y: dependent (right-hand side)
  - + X (functionally) determines Y
  - + Y (functionally) depends on X
- **Amstrong's Axiom (Inference Rules)**
  - + IR1 - Reflexivity
  - + IR2 - Augmentation
  - + IR3 - Transitivity
  - + IR4 - Union
  - + IR5 - Decomposition
  - + IR6 - Pseudo-transitivity
- **Types of functional dependencies**
  - + Full dependency
  - + Partial dependency
  - + Transitive dependency
- **Closure of a set of attributes**
  - + Identify (candidate) keys
  - + Verify if a FD is inferred from a set of FDs
- **Equivalence of sets of FDs** — F ≡ G
  - + F covers G
  - + G covers F
- **Canonical covers**
  - + Free of **redundant dependencies**
  - + Free of **extraneous attributes**

(Coronel and Morris, 2015) (Hoffer et al., 2019)

B. Functional Dependency Theory

B. Functional Dependency Theory



Outline

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

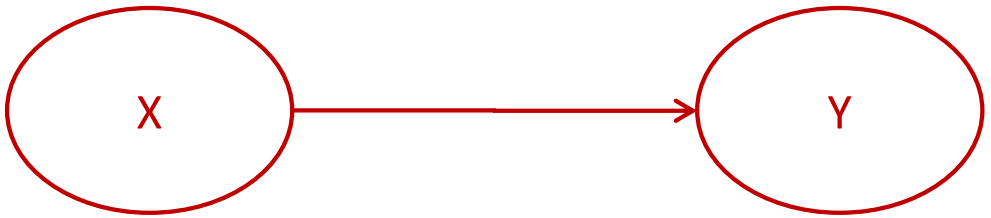
Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

Functional Dependency (FD)

(Phụ thuộc hàm)

- Specifying a **constraint** between two sets of attributes (X, Y) of a relational schema R.
- Denoted as: **X → Y**
  - **X**, **Y** are two sets of attributes of the **same schema R**
  - **Y** (functionally) **depends on X** (on R).
  - **X** (functionally) **determines Y** (on R).
  - **X holds Y** (on R).



Left-side
(Determinant)
(Định thuộc)

Right-side
(Dependent)
(Phụ thuộc)

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Functional Dependency (FD)

(Phụ thuộc hàm)

Let **R(A1, A2, …, An)**, R^+ = **{A1, …, An}**

X, Y $\subseteq R^+$, X → Y if and only if:

$\forall r \in R,\ t_1, t_2 \in r$, **if** $t_1[X] = t_2[X]$ **then** $t_1[Y] = t_2[Y]$

Interpretation: X → Y holds on R (X determines Y on R) whenever two tuples have the same value for X, they must have the same value for Y.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

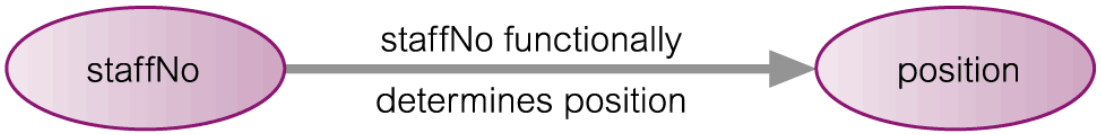
fit@hcmus

Functional Dependency (FD)

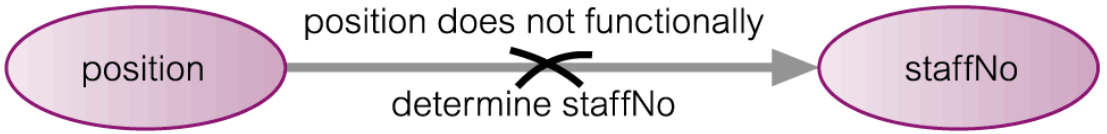
(Phụ thuộc hàm)

$\forall r \in R,\ t_1, t_2 \in r$, if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London



SL21 → Manager



Manager → SL21
→ SG5

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Functional Dependency (FD)

(Phụ thuộc hàm)

$\forall r \in R,\ t_1, t_2 \in r,$ if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

FD Identification based on the relation Staff:

staffNo → sName

sName ↛ staffNo

Business rules

The identification of FDs within a relational schema must be based on the semantics of the attributes and their interrelationships, rather than the data itself.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

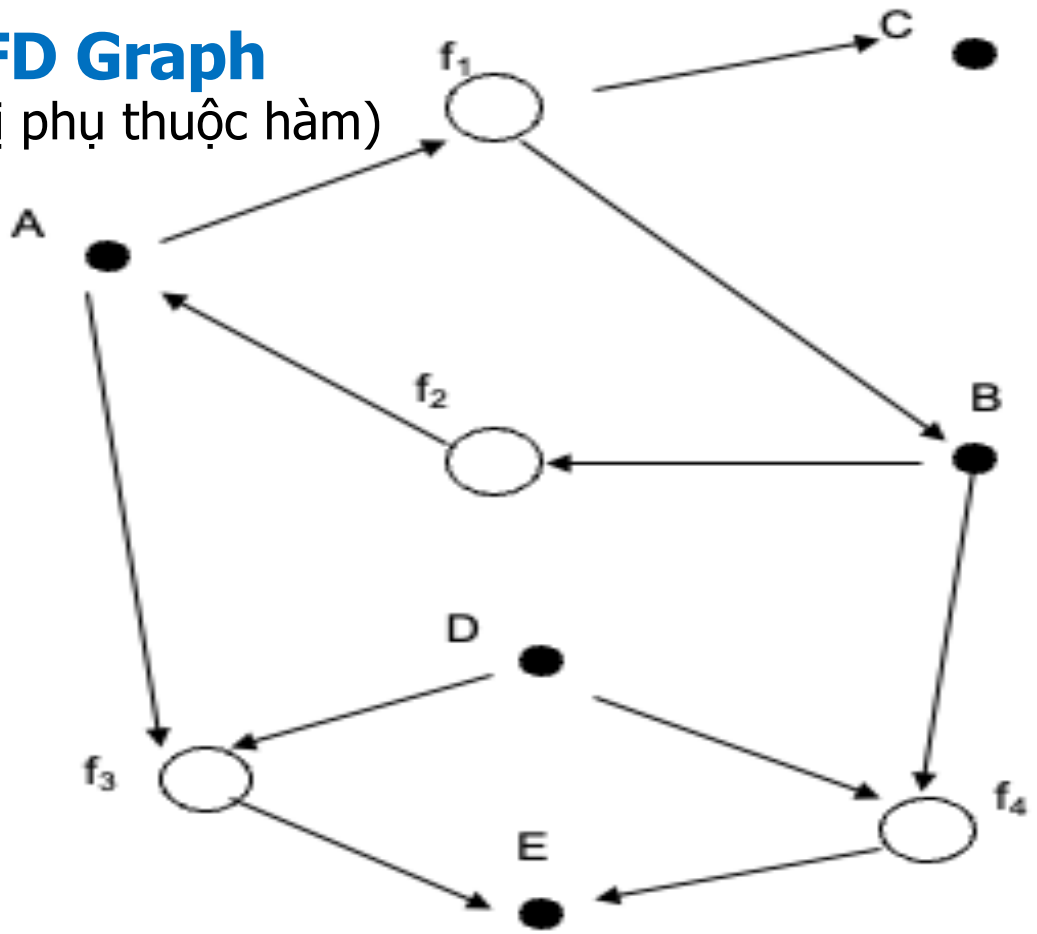
Visual Representation of FDs

Given R = <U, F> with: U = {ABCDE}

F = { f_1: A→BC; f_2: B →A; f_3: AD →E; f_4: BD →E }

FD Graph

(Đồ thị phụ thuộc hàm)



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

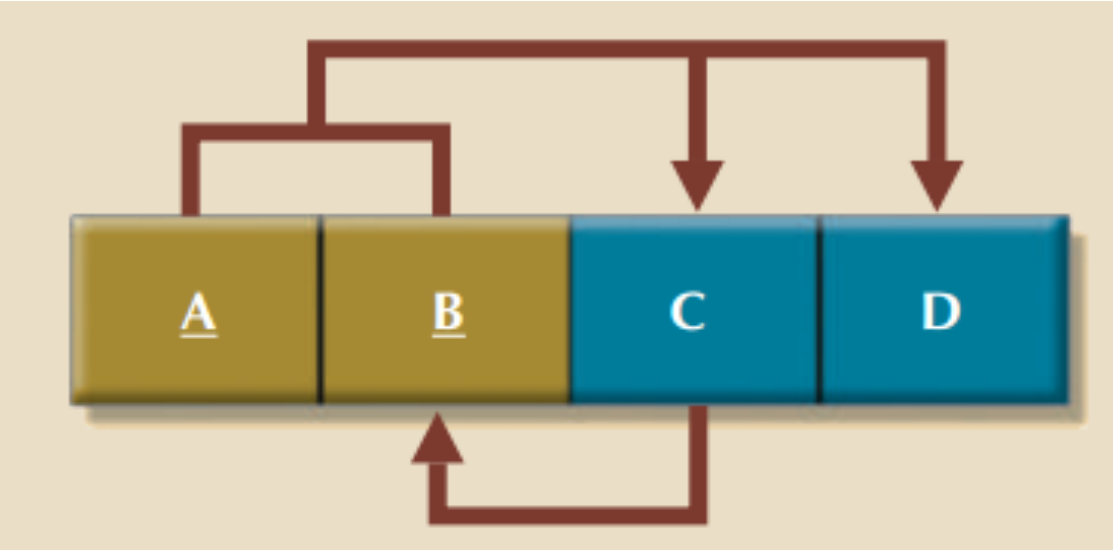
Visual Representation of FDs

Given R = <U, F> with: U = { ABCD }

F = { f1: AB →CD; f2: AC →BD; f3: C →B }

FD Diagrams

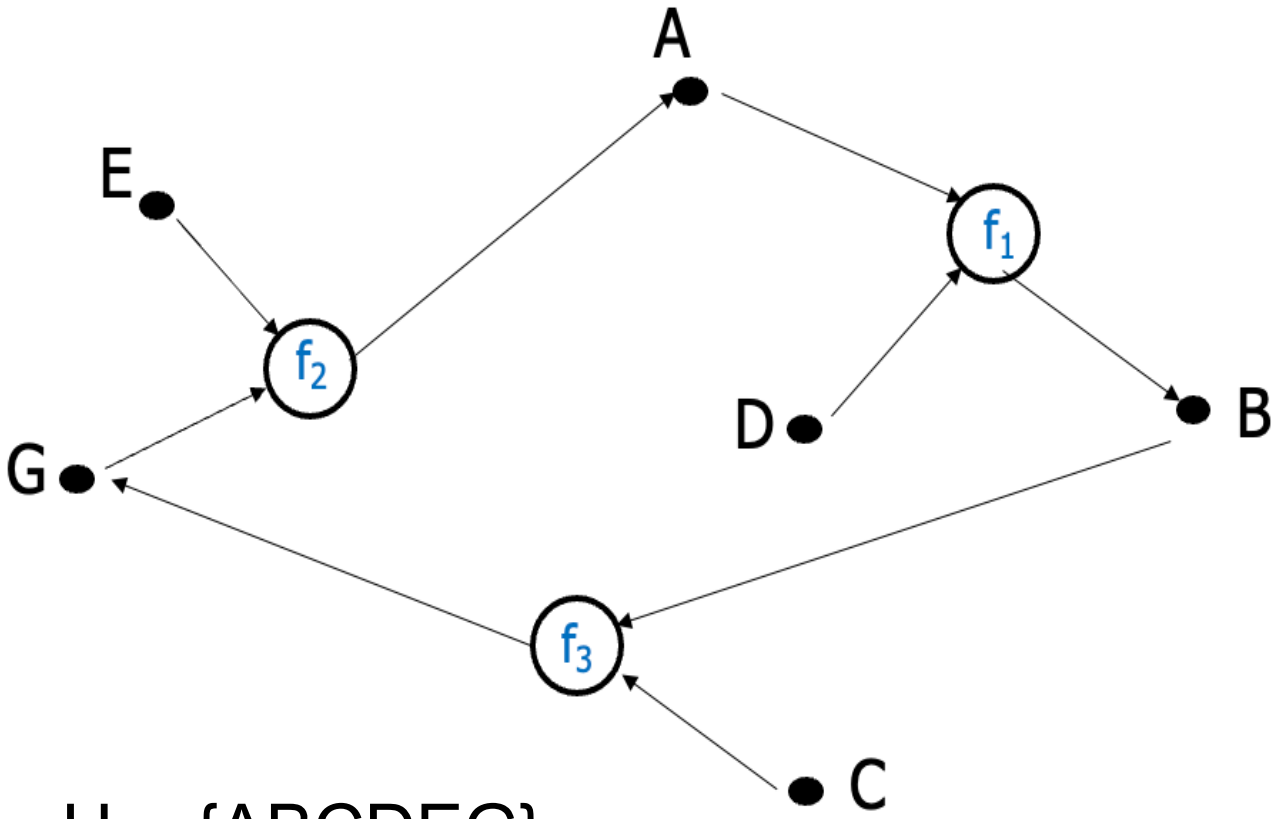
(Sơ đồ phụ thuộc hàm)



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Practice 1.1: **FD Graph/Diagram**

Let R = <U, F> with U: set of R's attributes, F: set of Rs' FDs.
Determine U, F through the following FD graphs/diagrams.



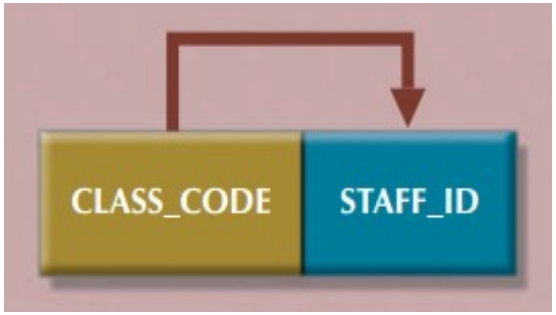
U = {ABCDEG}

F = { f_1: AD → B; f_2: EG → A; f_3: BC → G }

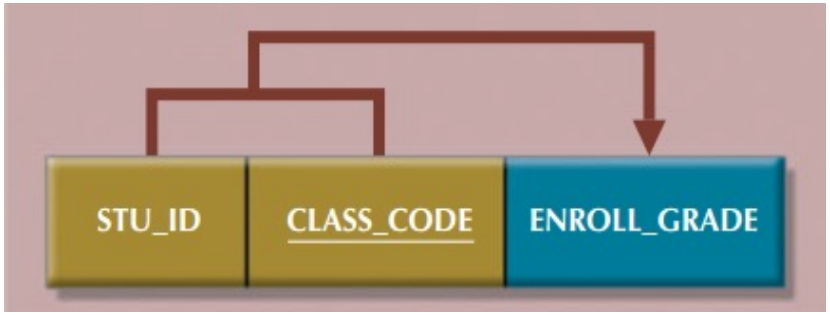
Practice 1.2: **FD Graph/Diagram**

Let R = <U, F> with U: set of R's attributes, F: set of Rs' FDs.
Determine U, F through the following FD graphs/diagrams.

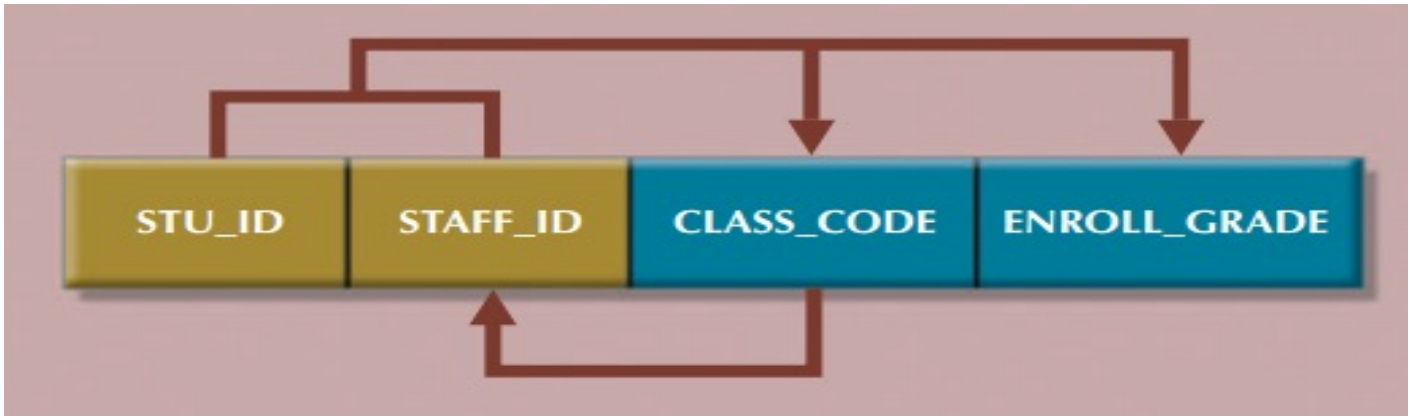
a)



b)



c)



Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

Inference Rules for FDs

(Luật dẫn/Tiên đề Amstrong)

- Armstrong's Inference Rules (or Axioms)

(Luật phản xạ)

- Reflexivity (**IR1**)

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

(Luật gia tăng)

- Augmentation (**IR2**)

$$X \rightarrow Y \Rightarrow XZ \rightarrow YZ$$

(Luật bắc cầu)

- Transitivity (**IR3**)

$$X \rightarrow Y \text{ and } Y \rightarrow Z \Rightarrow X \rightarrow Z$$

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

cdio 4.0

fit@hcmus

Inference Rules for FDs

(Hệ quả từ luật dẫn/Tiên đề Amstrong)

- Other Inference Rules

(Luật hội)

- Additivity or Union (**IR4**)

$$X \rightarrow Y \text{ and } X \rightarrow Z \Rightarrow X \rightarrow YZ$$

(Luật phân rã)

- Decomposition or Projectivity (**IR5**)

$$X \rightarrow YZ \Rightarrow X \rightarrow Y \text{ and } X \rightarrow Z$$

(Luật bắc cầu giả)

- Pseudo Transitivity (**IR6**)

$$X \rightarrow Y \text{ and } YZ \rightarrow W \Rightarrow XZ \rightarrow W$$

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Implied Functional Dependency

(Phụ thuộc hàm suy dẫn/hệ quả)

- Let **F** be a set of functional dependencies on the schema **R**, and **U** be the set of attributes of R, denoted as **R = <U, F>.**
- **f** is an implied FD from F, denoted as **F ⊢ f**, if **f** can be logically inferred from the original set F using inference rules on the given FDs in F.
- Example:

  R(ABCGHI),

  F = { f1: A→B, f2: A→C, f3: CG→H, f4: CG→I, f5: B→H }

  f6: A → H (IR3) ⇒ **F ⊢ f6**

  f7: AG → I (IR6) ⇒ **F ⊢ f7**

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

Closure of a Set of FDs

(Bao đóng của tập phụ thuộc hàm)

- The closure of F, denoted by **F^+**, is the set of **ALL** functional dependencies logically inferred from the original set F using inference rules on the given FDs in F.
- Critical to ensure that a database schema is free from redundancy and anomalies and to understand the relationships between attributes of a given schema.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Closure of a Set of FDs

(Bao đóng của tập phụ thuộc hàm)

- **Example**:

  R(ABCGHI)

  F = { f1: A→B, f2: A→C, f3: CG→H, f4: CG→I, f5: B→H }

  F^+ = F ∪ { f6: A → H, f7: AG → I } ∪ …

```
F+ = F
apply the reflexivity rule /* Generates all trivial dependencies */
repeat
    for each functional dependency f in F+
        apply the augmentation rule on f
        add the resulting functional dependencies to F+
    for each pair of functional dependencies f1 and f2 in F+
        if f1 and f2 can be combined using transitivity
            add the resulting functional dependency to F+
until F+ does not change any further
```

☹ $F \subseteq F^+$; F^+ is very large.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Closure of a Set of FDs

(Bao đóng của tập phụ thuộc hàm)

- **Example**:

  R(ABCGHI)

  F = { f1: A→B, f2: A→C, f3: CG→H, f4: CG→I, f5: B→H }

  F^+ = F ∪ { f6: A → H, f7: AG → I } ∪ …

$F^+ = F$
apply the reflexivity rule /* Generates all trivial dependencies */
repeat
  for each functional dependency f in F^+

☹ neither practical nor applicable.

  for each pair of functional dependencies f_1 and f_2 in F^+
    if f_1 and f_2 can be combined using transitivity
      add the resulting functional dependency to F^+
until F^+ does not change any further

☹ $F \subseteq F^+$; F^+ is very large.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Closure of a Set of Attributes

(Bao đóng của tập thuộc tính)

- For each such set of attributes X, we determine the set X^+ of attributes that are functionally determined by X based on F; X^+_F is called the closure of X under F.

$$X^+_F = \{ Y \mid X \rightarrow Y \text{ logically inferred from } F \}$$

$$X \subseteq X^+_F$$

$$X \subseteq R^+$$

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Closure of a Set of Attributes

(Bao đóng của tập thuộc tính)

- **Example**:

  R(ABCGHI)

  F = { f1: A→B, f2: A→C, f3: CG→H, f4: CG→I, f5: B→H }

  A^+ = { ABCH }; $(BC)^+$ = { BCH }

closure = X;
repeat until there is no change: {
    if there is an FD $U \rightarrow V$ in F such that $U \subseteq$ *closure*,
        then set *closure* = *closure* $\cup$ V
    }

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Closure of a Set of Attributes

(Bao đóng của tập thuộc tính)

- Use of attribute closures:
  - Test if K is a super key.
  - Test if X → Y is an implied FD from the original set F (i.e., included in F^+).
  - An alternative way to compute F^+.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Closure of a Set of Attributes

(Bao đóng của tập thuộc tính)

- **Example**:

  R(ABCGHI)

  F = { f1: A→B, f2: A→C, f3: CG→H, f4: CG→I, f5: B→H }

  - Which of the following attribute sets is a candidate key?
    - AG ✓
    - AB
    - CGB

    $(AG)^+ = \{ AGBCHI \} = U$

  - Which of the following FDs is included in F^+?
    - A→I
    - AG→H ✓
    - BG→H

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Practice 2.1: **Find the closure of a set of attribute(s)**

Given R(ABCDEF); F = { AB→C, BC→AD, D→E, CF→B }

a. Find the closure of the following attribute sets:

- D^+_F = ?
- BC^+_F = ?
- CF^+_F = ?

b. Which of the following FDs is inferred from F?

- f1: DF→EF
- f2: BC→E
- f3: BC→F
- f4: CF→A

c. Find a possible candidate key?

Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

Keys

(Siêu khoá)

- A **super key** of a relation schema R = { A_1, A_2, ..., A_n } **is** a set of attributes from R (S ⊆ R) with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$.
  - Uniqueness: Identify each row in the relation.

(Khoá) (Khoá ứng viên)

- A **key** (or **candidate key**) K is a **super key** with the additional property that removal of any attribute from K will cause K not to be a super key anymore.
  - Uniqueness: Identify each row in the relation.
  - Minimality: $\neg\exists X' \subset X$ such as X' is a super key.

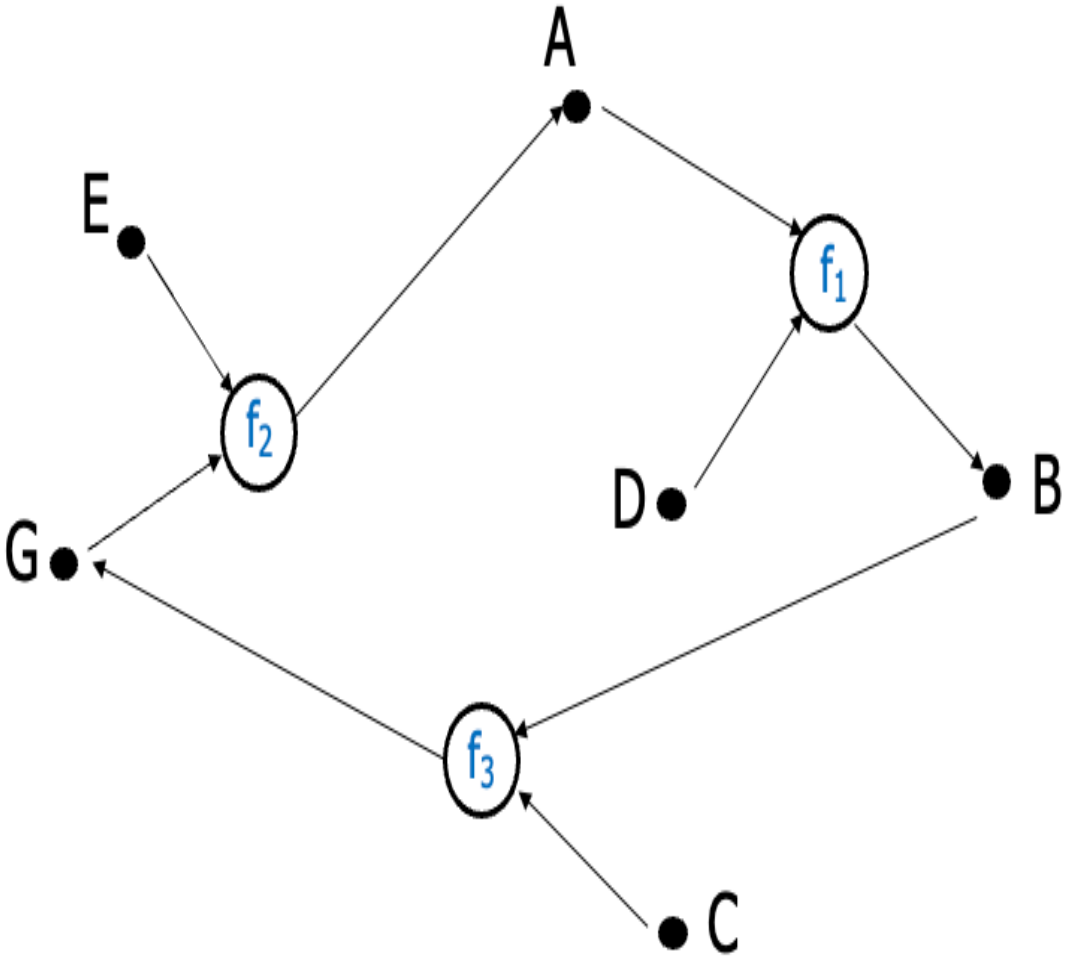
(Coronel and Morris, 2015) (Silberschatz et al. 2020)

FDs and Keys

R = <U, F> with:
U = ABCDEG, F = { f_1: AD → B; f_2: EG → A; f_3: BC → G }
Key of Q?

N (left-only) = { EDC }
D (right-only) = Ø
L (other) = { A, B, G }

$(EDCA)^+$ = U ➔ K1
$(EDCB)^+$ = U ➔ K2
$(EDCG)^+$ = U ➔ K2



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Algorithm for Computing Keys

B1 Gọi *left*(*f*) là vế trái, *right*(*f*) là vế phải của phụ thuộc hàm *f*.

B2 - Tính $\mathcal{N} := U - \cup_{\forall f \in F} right(f)$

B3 - Nếu $\mathcal{N}^+_{\mathcal{F}} = U$ thì *R* chỉ có một khóa là $\mathcal{N}$.

B4 - Nếu $\mathcal{N}^+_{\mathcal{F}} \subset U$ thì

B4.1 Tính $\mathcal{D} := \cup_{\forall f \in \mathcal{F}} right(f) - \cup_{\forall f \in F} left(f)$

B4.2 Tính $\mathcal{L} := U - \mathcal{N}^+_{\mathcal{F}} \mathcal{D}$

B5 $\mathcal{K} := \emptyset$;

Với mỗi tập con $\mathcal{L}_i \subseteq \mathcal{L}$, nếu $\{\mathcal{N}\mathcal{L}_i\}^+_{\mathcal{F}} = U$: $\mathcal{K} := \mathcal{K} \cup \{\mathcal{N}\mathcal{L}_i\}$

Trong khi mà tồn tại $\mathcal{K}_i, \mathcal{K}_j$ thuộc $\mathcal{K}$ sao cho $\mathcal{K}_i \subset \mathcal{K}_j$ thì

$$\mathcal{K} := \mathcal{K} - \{\mathcal{K}_j\}$$

R có tập khóa là $\mathcal{K}$.

Example #1

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

  U = { ABCDEF }

- F= { f1: D → B ,
  f2: A → C ,
  f3: AD → E ,
  f4: C → F }

B1 Gọi *left(f)* là vế trái, *right(f)* là vế phải của phụ thuộc hàm *f*.
B2 - Tính $\mathcal{N} := U - \cup_{\forall f \in F} right(f)$
B3 - Nếu $\mathcal{N}^+_{\mathcal{F}} = U$ thì *R* chỉ có một khóa là $\mathcal{N}$.
B4 - Nếu $\mathcal{N}^+_{\mathcal{F}} \subset U$ thì
B4.1 Tính $\mathcal{D} := \cup_{\forall f \in \mathcal{F}} right(f) - \cup_{\forall f \in F} left(f)$
B4.2 Tính $\mathcal{L} := U - \mathcal{N}^+_{\mathcal{F}} \mathcal{D}$
B5 $\mathcal{K} := \emptyset$;
Với mỗi tập con $\mathcal{L}_i \subseteq \mathcal{L}$, nếu $\{\mathcal{NL}_i\}^+_{\mathcal{F}} = U$: $\mathcal{K} := \mathcal{K} \cup \{\mathcal{NL}_i\}$
Trong khi mà tồn tại $\mathcal{K}_i, \mathcal{K}_j$ thuộc $\mathcal{K}$ sao cho $\mathcal{K}_i \subset \mathcal{K}_j$ thì
$\mathcal{K} := \mathcal{K} - \{\mathcal{K}_j\}$
R có tập khóa là $\mathcal{K}$.

B1: **left(f) = { DAC }, right(f) = { BCEF }**

B2: **N = U – right(f) = { ABCDEF } – { BCEF } = { AD }**

B3: N^+_F = { *tập thuộc tính xác định/ được dẫn ra từ N* }
= { ADCBEF } = U

R có duy nhất 1 khoá là AD

Example #2

- Tìm khóa của lược đồ quan hệ :

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

  U = { ABCDEF }

B1: **left(f) = { ABC }, right(f) = { ACDEF }**

B2: **N = U – right(f) = { ABCDEF } – { ACDEF } = { B }**

B3: N^+_F = { *tập thuộc tính xác định/ được dẫn ra từ N* }
= { B } ⊂ U

- Có tập phụ thuộc hàm sau:

  F= { f1: A → D ,
  f2: C → AF ,
  f3: AB → EC
  }

Example #2

- Tìm khóa của lược đồ quan hệ :

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

  U = { ABCDEF }

- Có tập phụ thuộc hàm

  F= { f1: A → D ,
  f2: C → AF ,
  f3: AB → EC
  }

B1: **left(f) = { ABC }, right(f) = { ACDEF }**

B2: **N = U – right(f) = { ABCDEF } – { ACDEF } = { B }**

B3: $\mathbf{N^+_F}$ **= {** *tập thuộc tính xác định/ được dẫn ra từ N* **}**
= { B } ⊂ U

B4.1: **D = right(f) – left(f)** *--tìm tập chỉ bên vế phải*
= { ACDEF } – { ABC } = { DEF }

B4.2: **L = U –** $\mathbf{D.N^+_F}$ *--tìm tập có khả năng làm khoá*
= { ABCDEF } – { DEFB } = { AC }

Example #2

- Tìm khóa của lược đồ quan hệ :

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

  U = { ABCDEF }

- Có tập phụ thuộc hàm

  F= { f1: A → D ,
  f2: C → AF ,
  f3: AB → EC
  }

B1: **left(f) = { ABC }, right(f) = { ACDEF }**

B2: **N = U – right(f) = { ABCDEF } – { ACDEF } = { B }**

B3: N^+_F = { *tập thuộc tính xác định/ được dẫn ra từ N* }
= { B } ⊂ U

B4.1: **D = right(f) – left(f)** *--tìm tập chỉ bên vế phải*
= { ACDEF } – { ABC } = { DEF }

B4.2: $L = U - D.N^+_F$ *--tìm tập có khả năng làm khoá*
= { ABCDEF } – { DEFB } = { AC }

B5: **Tìm Li từ tổ hợp các thuộc tính trong L**
Số lượng Li = n = 2^n – 1, Li = { A, C, AC }

Example #2

- Tìm khóa của lược đồ quan hệ :

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

  U = { ABCDEF }

- Có tập phụ thuộc hàm

  F= { f1: A → D ,
  f2: C → AF ,
  f3: AB → EC
  }

B1: **left(f) = { ABC }, right(f) = { ACDEF }**

B2: **N = U – right(f) = { ABCDEF } – { ACDEF } = { B }**

B3: N^+_F = { *tập thuộc tính xác định/ được dẫn ra từ N* }
= { B } ⊂ U

B4.1: **D = right(f) – left(f)** *--tìm tập chỉ bên vế phải*
= { ACDEF } – { ABC } = { DEF }

B4.2: **L = U – D.N^+_F** *--tìm tập có khả năng làm khoá*
= { ABCDEF } – { DEFB } = { AC }

B5: **Tìm Li từ tổ hợp các thuộc tính trong L**
Số lượng Li = n = 2^n – 1, Li = { A, C, AC }

Với Li = { A }
$\{ NLi \}^+_F$ = { BADECF } = U ➔ **K1 = { BA } là khoá**
➔ Loại tất cả tập liên quan A ra khỏi Li

Với Li = { C }
$\{ NLi \}^+_F$ = { BCAFED } = U ➔ **K2 = { BC } là khoá**
➔ Loại tất cả tập liên quan C ra khỏi Li

Practice 3: **Key Computing**

Find all keys of R = <U, F> with:

a. U = {ABCD}, F = { A→B, BC→D, D→A }

b. U = {ABCDEG}, F = { AD→B, EG→A, BC→G }

Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

cdio 4.0

fit@hcmus

Equivalence of Sets of FDs

(Tập phụ thuộc hàm tương đương)

- Notation: **F ≡ G** with F, G are two sets of FDs.

- Def #1: F and G are equivalent if $F^+ = G^+$.

$$\mathbf{F \equiv G \Leftrightarrow F^+ = G^+}$$

- Def #2: F and G are equivalent if they meet two following conditions:
  - Every FD in F can be inferred from G (i.e., G covers F)
  - Every FD in G can be inferred from F (i.e., F covers G)

$$\mathbf{F \equiv G \Leftrightarrow F \text{ covers } G \text{ and } G \text{ covers } F}$$

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Example

R(ABCDE)

F = { A→BC, A→D, CD→E }

G = { A→BCE, A→ABD, CD→E }

F ≡ G ? ☑

- { A→BCE, A→ABD, CD→E } → { A→BC, A→D, CD→E }
  - F ⊆ G ➔ G obviously covers F.
- { A→BC, A→D, CD→E } → { A→BCE, A→ABD, CD→E }
  - In G, the FD A→E can be found in F using inference rules?
  - F covers G

Practice 4: **Test the equivalence of FD sets**

Given the two sets of FDs: F, G as follows. F ≡ G ?

a. F = { A→BC, A→D, CD→E }
   G = { A→BCDE }

b. F = { M→NE, EG→CD, E→M }
   G = { M→NE, MG→CDN, E→MN, EG→N }

Functional Dependency Theory

- Functional Dependency (FD)
- Inference Rules for FDs
- Closure of a Set of FDs and/or Attributes
- Use of Closure for Key Identification
- Equivalence of a Set of FDs
- Canonical Covers

fit@hcmus

Canonical Cover

Minimal Sets of FDs

(Phủ tối thiểu PTT)

- Sets of FDs may have redundant dependencies that can be inferred from the others.
  - Example: Given F = { f1: A → B, f2: B → C, f3: A → C }, f3: A → C is a redundant dependency since it can be inferred from f1, f2 through IR3 (transitive rule).

- Parts of a functional dependency may also be redundant (extraneous attributes).
  - Example: F = { f1: A → C, f2: AB → C } B is extraneous attribute in f2 since C can be fully functionally inferred from a.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

cdio 4.0

fit@hcmus

Canonical Cover

Minimal Sets of FDs

(Phủ tối thiểu PTT)

Intuitively, a canonical cover of F, denoted as F_c, is a "minimal" set of FDs equivalent to F, having no redundant dependencies or redundant parts (extraneous attributes) of dependencies.

- A set of FDs is **minimal** must satisfies :
  - Every FD has a single attribute for its right-hand side.
  - We cannot remove any attribute of any determinant (the left-hand side of a FD) and still have a set of dependencies that is equivalent to F (free of LEFT extraneous attributes).
  - We cannot remove any dependency from F and still have a set of dependencies that are equivalent to F (free of redundant dependencies).

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Compute a Canonical Cover

Minimal Sets of FDs

(Phủ tối thiểu PTT)

- Step 0: Start with the original set of FDs (Fc = F)
- Step 1: Create a single attribute right-hand side for all FDs.
- Step 2: Remove all extraneous attributes.
- Step 3: Remove all redundant functions.
- Step 4: Form a canonical cover from the remaining FDs.

Example:

R(ABCD) with F = { $A \rightarrow BC$, $B \rightarrow AC$, $C \rightarrow A$ }. Compute F_C?

S1: Fc = { f1: $A \rightarrow B$, f2: $A \rightarrow C$, f3: $B \rightarrow A$, f4: $B \rightarrow C$, f5: $C \rightarrow A$ }

S2: -

S3: Which FDs can be inferred from others when removed?

f1 ~~f2~~ ~~f3~~ f4 f5

S4: Fc = { f1: $A \rightarrow B$, f4: $B \rightarrow C$, f5: $C \rightarrow A$ }

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Practice 5.1: **Compute a canonical cover F_C.**

F = { A→BC, B→C, AB→C, A→B }

S1: Fc = { f1: A → B,
f2: A → C,
f3: B → C,
f4: AB → C,
f5: A → B }

S2: AB → C (since A → C)
Fc = { f1: A → B, f2: A → C, f3: B → C, f4: A → C; f5: A → B }

S3: Redundant FD? f1 f2 f3 f4 f5
Fc = { f1: A → B, f3: B → C }

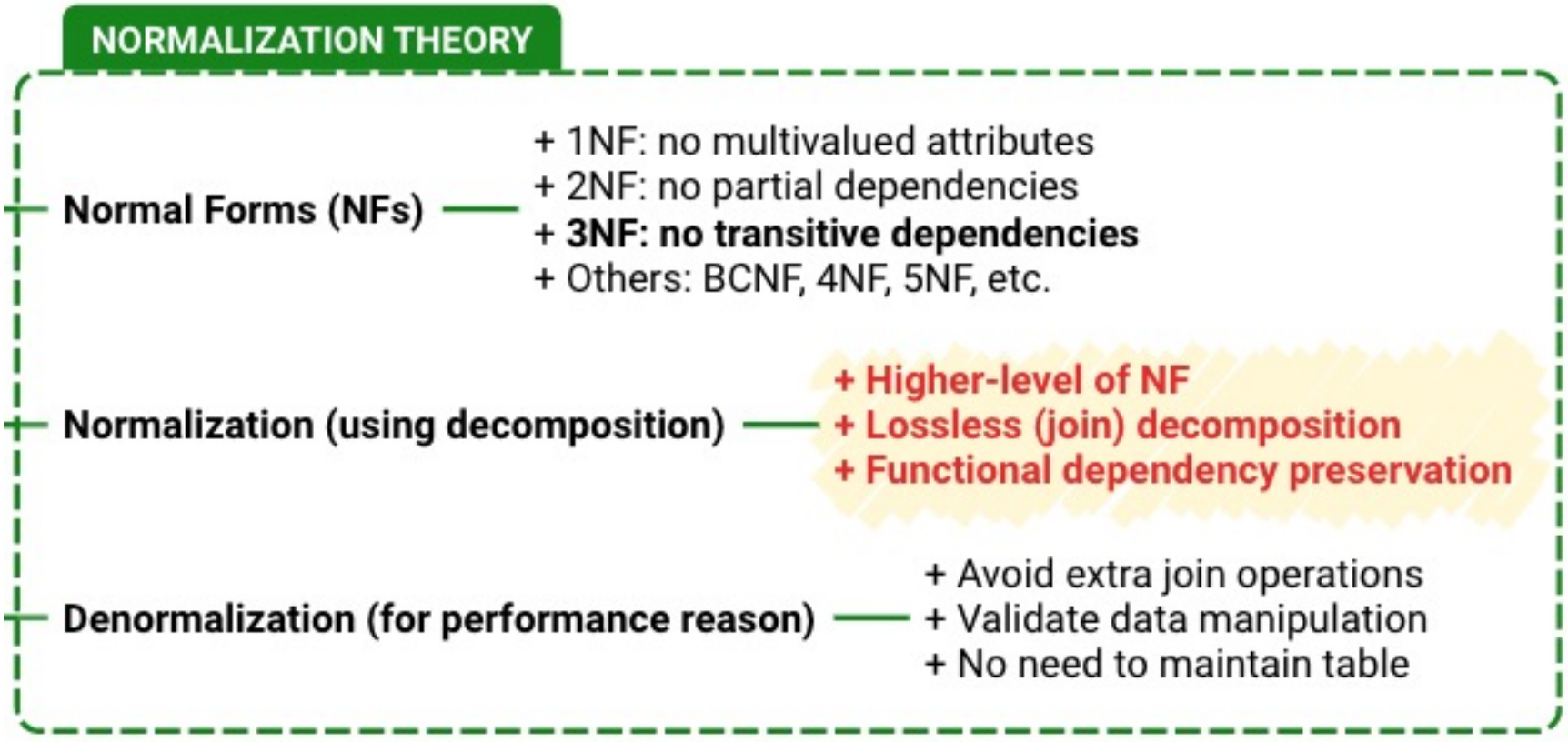
S4: Fc = { f1: A → B, f3: B → C }

Practice 5.2: **Compute a canonical cover F_C.**

F = { A→BCD, BCD→E, CD→EI }

C. Normalization Theory

C. Normalization Theory



Outline

- Fundamental Concepts
- Normal Forms (NFs)
- Normalization using Decomposition

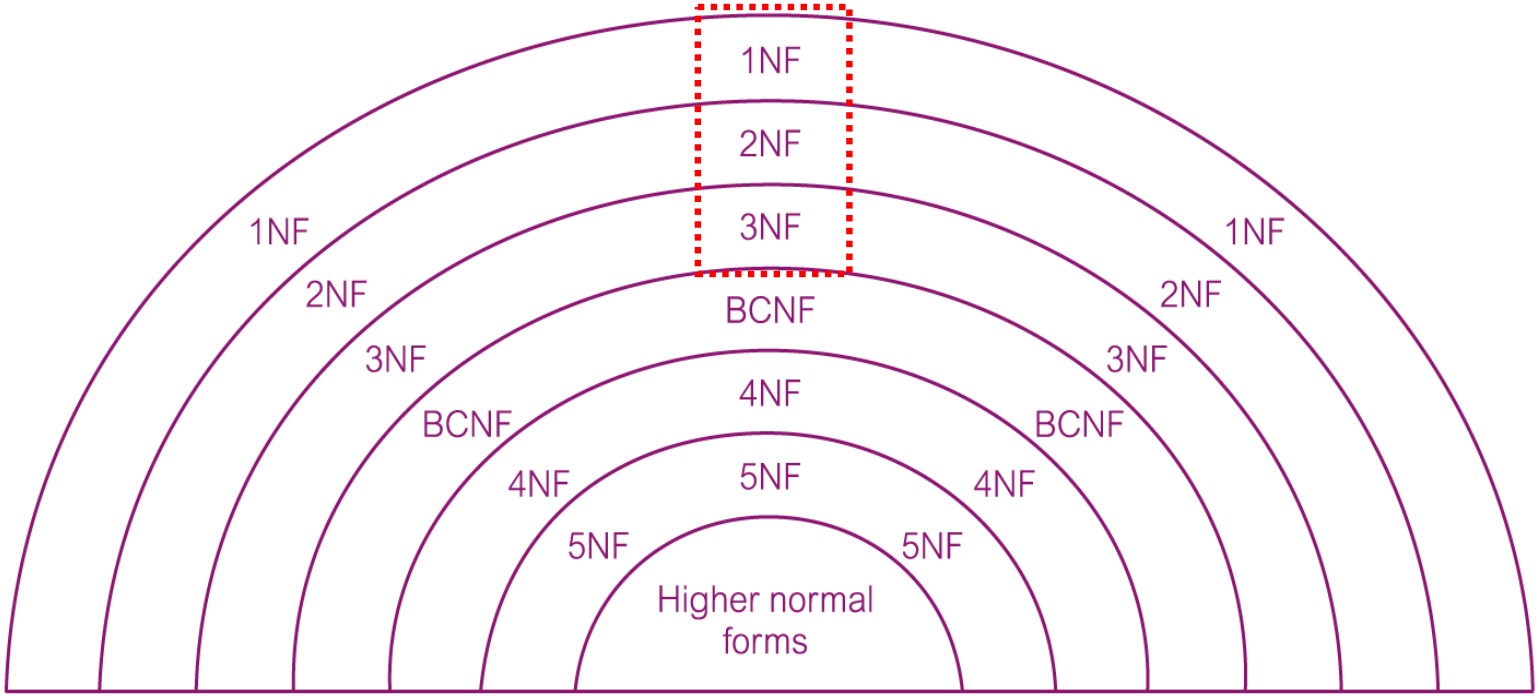
Outline

- Fundamental Concepts
- Normal Forms (NFs)
- Normalization using Decomposition

Normal Form

(Dạng chuẩn)

- Normal Form (NF): Conditions using keys and FDs of a relation to certify whether a relation schema is in a particular normal form (1NF, 2NF, 3NF, BCNF, …).



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Normalization

(Chuẩn hoá dữ liệu)

- Normalization: A process of evaluating and correcting unsatisfactory ("bad") relations to minimize data redundancies. (Trùng lắp dữ liệu)

- Use of Normalization:
  - Analyze the relationships among the attributes within a relation.
  - Evaluate whether the existing structure can be enhanced through normalization.
  - Enhance the current structure as needed.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Decomposition

(Phân rã)

- Decomposition: Normalization technique allows breaking up the attributes of a "bad" relation schema into smaller relations to improve its quality.

- Criteria for a decomposition:
  - Dependency-preserving decomposition

    (Bảo toàn PTH)
  - Lossless-join decomposition (≠ lossy decomposition)

    (Bảo toàn thông tin)

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Trivial vs. non-trivial functional dependencies

(Phụ thuộc hàm hiển nhiên)

- **Trivial functional dependency**

  Given $X \rightarrow Y$: **If** $Y \subseteq X$ **Then** $X \rightarrow Y$ is a trivial FD (IR1).

(Phụ thuộc hàm không hiển nhiên)

- **Non-trivial functional dependency**

  Given $X \rightarrow Y$: **If** $Y \not\subseteq X$ **Then** $X \rightarrow Y$ is a non-trivial FD.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Full vs. partial functional dependencies

(Phụ thuộc hàm đầy đủ)

- **Full** (functional) **dependency**

  Given $X \rightarrow Y$:

  If $\neg\exists X' \subset X$ **such as** $X' \rightarrow Y$

  Then Y depends fully functionnally on X

(Phụ thuộc hàm riêng phần / KHÔNG đầy đủ)

- **Partial** (functional) **dependency**

  Given $X \rightarrow Y$:

  If $\exists X' \subset X$ **such as** $X' \rightarrow Y$

  Then Y depends partially functionally on X

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Example: SV (mãsv, họtên, mãlớp, tênlớp, mônhọc, điểm)
Which of the following FDs are full FDs? partial FDs?

- f1: Mãsv → Họtên, mãlớp
- f2: Mãlớp → Tênlớp
- f3: Mãsv, Mônhọc → Điểm
- f4: Mãsv, Tênsv, Mônhọc → Điểm
- f5: Mãsv, Mãlớp → Tênlớp

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Example: SV (mãsv, họtên, mãlớp, tênlớp, mônhọc, điểm)
Which of the following FDs are full FDs? partial FDs?

- f1: Mãsv → Họtên, mãlớp
- f2: Mãlớp → Tênlớp
- f3: Mãsv, Mônhọc → Điểm
- f4: Mãsv, Tênsv, Mônhọc → Điểm
- f5: Mãsv, Mãlớp → Tênlớp

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Example:

R(ABCDEI), F = { A→BCD, BCD→E, CD→EI }

Which of the following FDs is a full dependency?

- BCD→E ✗
- A→D ✓
- CD→I ✓
- AC→I ✗

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Complementary Concepts

Transitive functional dependencies

(Phụ thuộc bắc cầu)

- **Transitive functional dependency**

A FD $X \rightarrow A$ is a transitive dependency if it satisfies all four of the following conditions:

(1) $X \rightarrow Y$,

(2) $Y \rightarrow A$,

(3) $Y \not\rightarrow X$,

(4) $A \notin XY$.

SinhVien (mãsv, họtên, mãlớp, tênlớp, mônhọc, điểm, sốcccd, ngàycấp_sốcccd)

Xác định phụ thuộc hàm bắc cầu?

f1: Mãsv → tênlớp

f2: Mãsv → ngàycấp_sốcccd

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Complementary Concepts

Transitive functional dependencies

SinhVien (mãsv, họtên, mãlớp, tênlớp, mônhọc, điểm, sốcccd, ngàycấp_sốcccd)

Xác định phụ thuộc hàm bắc cầu?

f1: Mãsv → tênlớp

(1) Mãsv → Mãlớp ☑

(2) Mãlớp → Tênlớp ☑

(3) Mãlớp ↛ Mãsv ☑

(4) Tênlớp ∉ {Mãsv, Mãlớp} ☑

→ *Phụ thuộc bắc cầu*

f2: Mãsv → ngàycấp_sốcccd

(1) Mãsv → Sốcccd ☑

(2) Sốcccd → Ngàycấp_sốcccd ☑

(3) Sốcccd → Mãsv ✗

→ ***KHÔNG*** *phụ thuộc bắc cầu*

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Recall: Key Concepts

- If a relation schema has more than one key, each is called a candidate key. One of the candidate keys is arbitrarily designed to be the primary key, and the others are called secondary keys.
  - A prime attribute (or key attribute) must be a member of some candidate key. (Thuộc tính khoá)
  - A non-prime attribute (or non-key attribute) is not a prime attribute, i.e., it is not a member of any candidate key. (Thuộc tính KHÔNG khoá)

R(AB CDEI)

Keys: AB or BD

✓ Key attributes: ABD;

✓ Non-key attributes: CEI

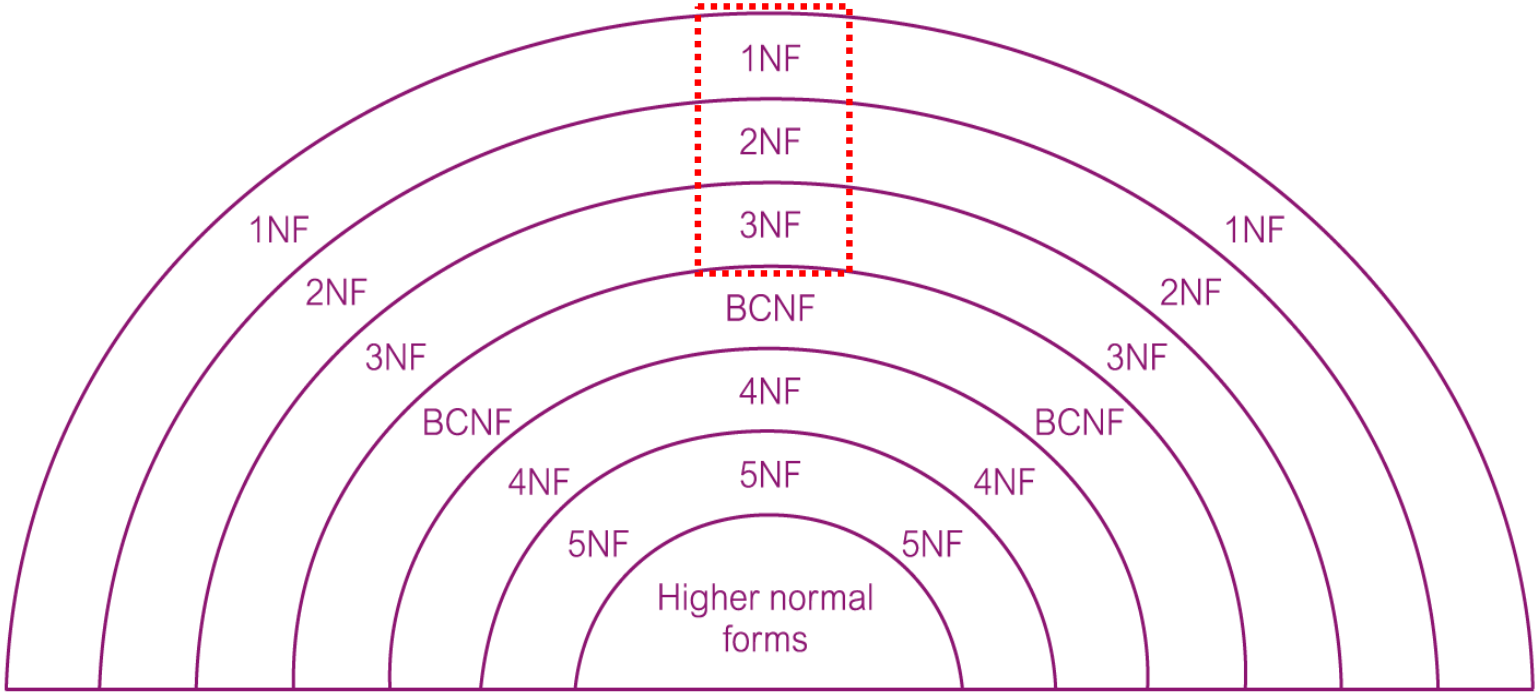
(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Outline

- Fundamental Concepts
- Normal Forms (NFs)
- Normalization using Decomposition

Types of Normal Forms

Normal Form (NF): Conditions based on keys and FDs of a relation to verify whether a relation schema is in a particular normal form (1NF, 2NF, 3NF, BCNF, …), which can help evaluate level of data redundancies in a database schema.



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Table with Repeating Groups

1NF

erID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
						Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Repeating groups

Complex structure with repeating groups (multivalued attributes).

Indeterminable keys.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

fit@hcmus

Recall: Properties of Relations

- Allow distinguishing relations from nonrelation tables:
  - Each relation (or table) in a database has a unique name.
  - No multivalued attributes are allowed in a relation.
  - Each attribute (or column) within a table has a unique name.
  - The order of columns (left to right) is insignificant.
  - No two rows in a relation can be identical.
    - **Each relation must have a determined primary key.**
  - The order of rows (top to bottom) is insignificant.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

First Normal Form (1NF)

Def: A relation is in **1NF** if the following two constraints both apply:

- There are no repeating groups (no multivalued attributes) in the relation (i.e., every attribute value is atomic).
- A primary key has been defined clearly, which uniquely identifies each row in a two-dimensional table.

1NF

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

First Normal Form (1NF)

1NF

Redundancy

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
		2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

☹ **Insertion, deletion, and update anomalies!**

There are multiple entity types in one relation.

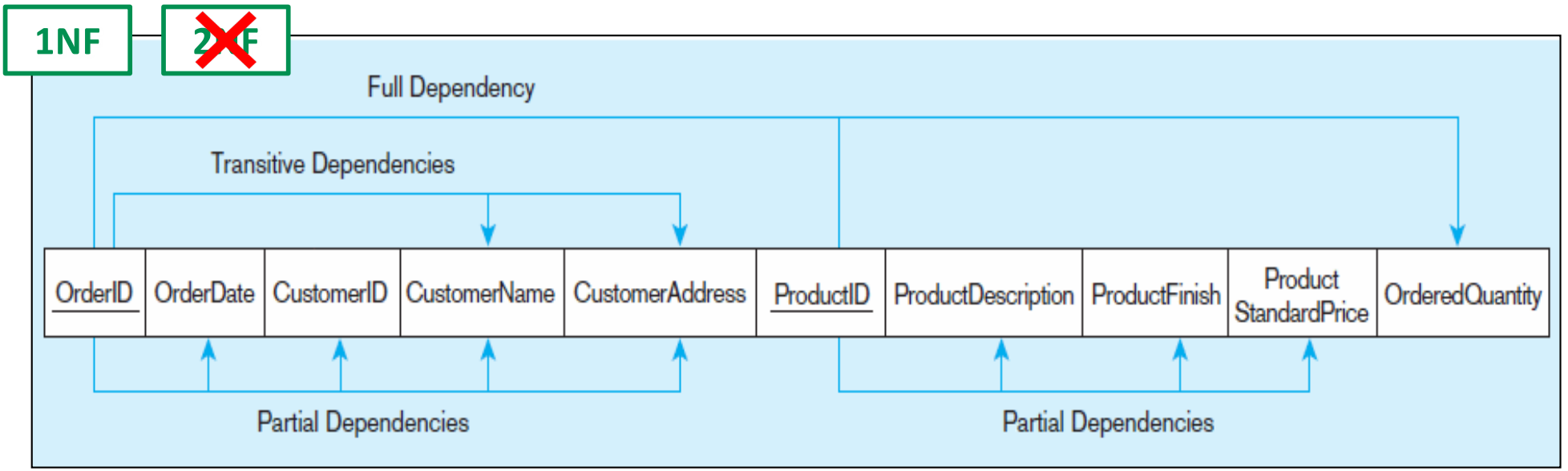
Convert the above relation to second normal form (2NF) to remove the entity duplication which results in anomalies.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Second Normal Form (2NF)

Def: A relation is in **2NF** if it meet two following conditions:

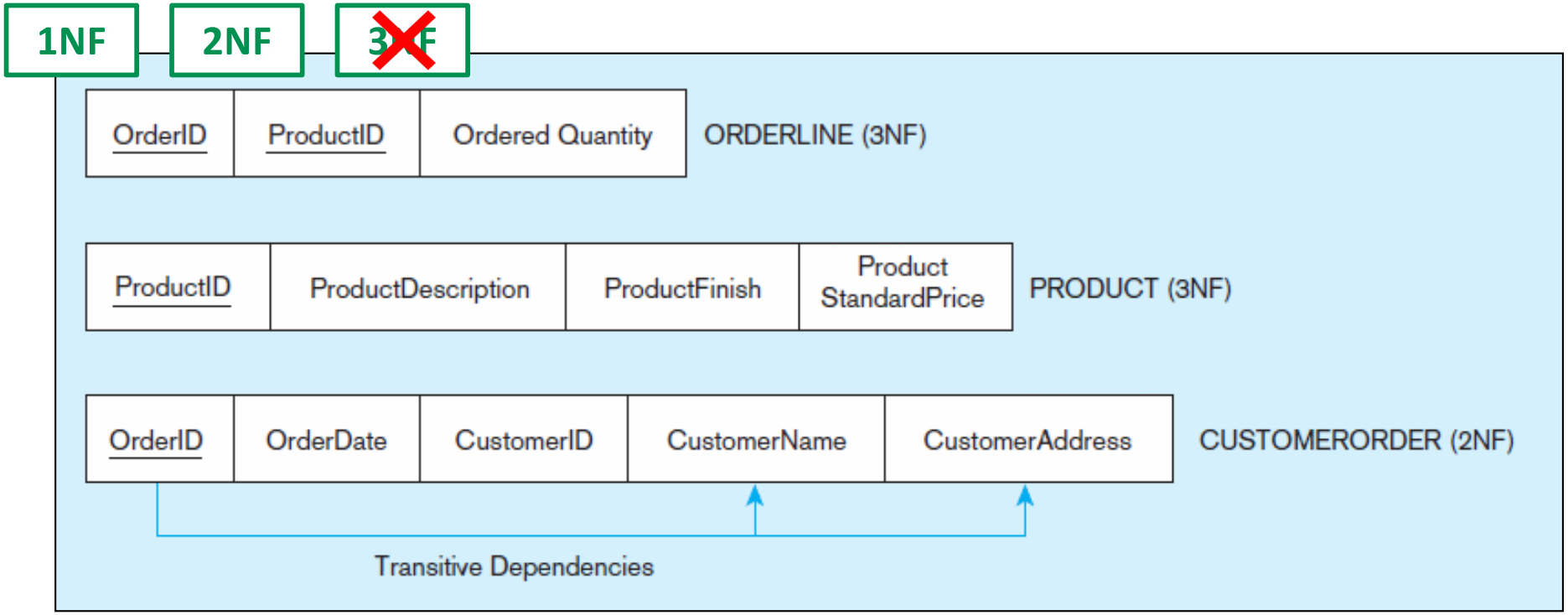
- It is in first normal form (1NF).
- **No** non-prime attribute is partially dependent on any key.



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Second Normal Form (2NF)

- Removing partial dependencies.



☹ **Data duplication still exists** due to transitive dependencies.

Convert the above relation to third normal form (3NF) to remove transitive dependencies.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Third Normal Form (3NF)

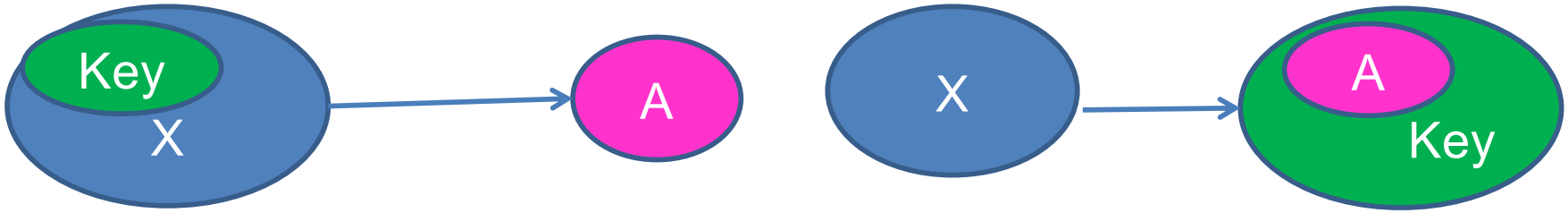
Appropriate for the relation with only one candidate key.

Def #1: A relation is in **3NF** if it meet two following conditions:

- It is in second normal form (2NF)
- And no transitive dependencies exist between the primary key and one or more non-prime (non-key) attributes.

Appropriate for the relation with at least two candidate keys.

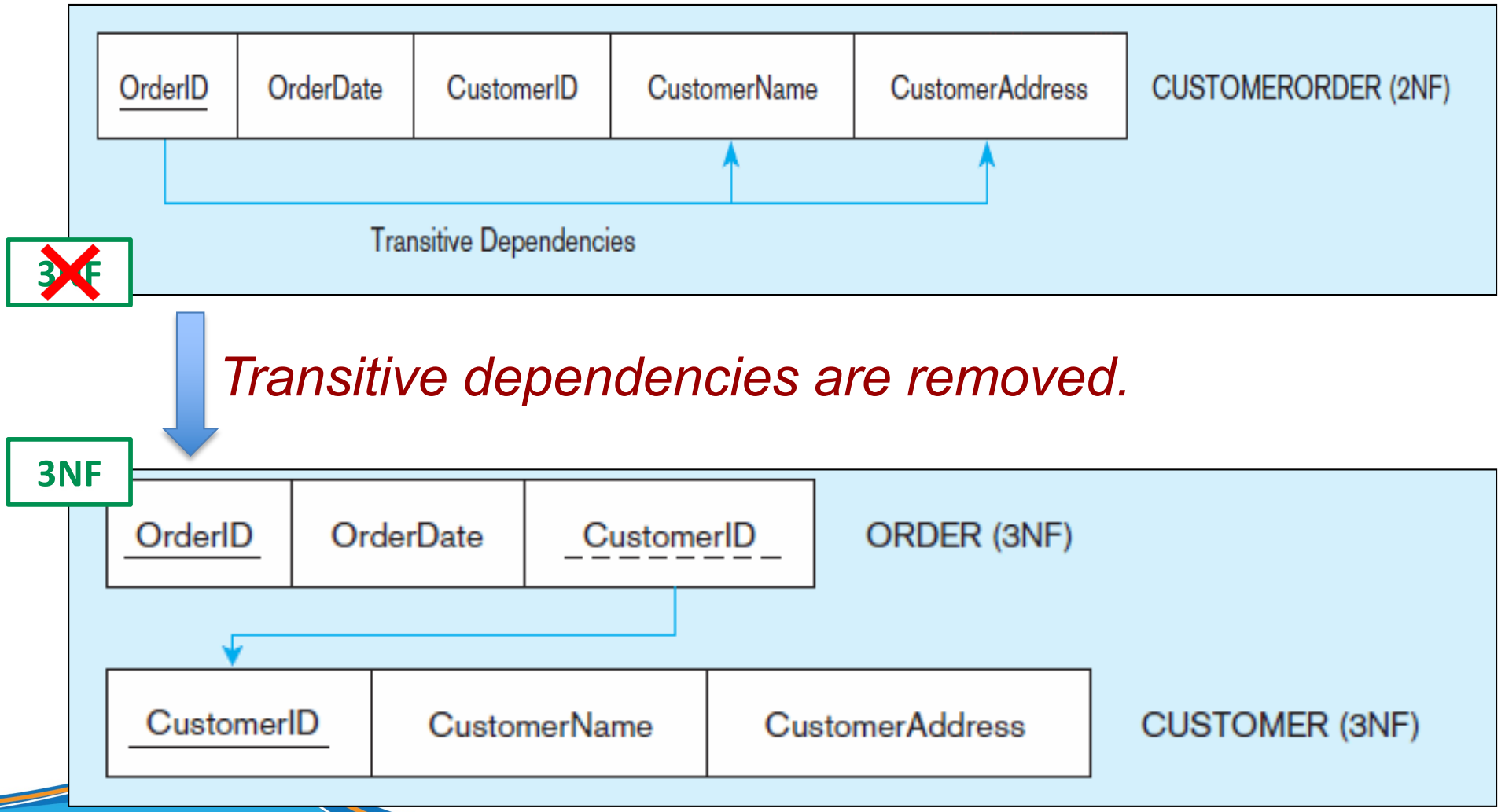
Def #2: A relation is in **3NF** if, for every non-trivial FD X→A in the set of FDs of the relation, X is a super key or A is a part of a key.



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Third Normal Form (3NF)

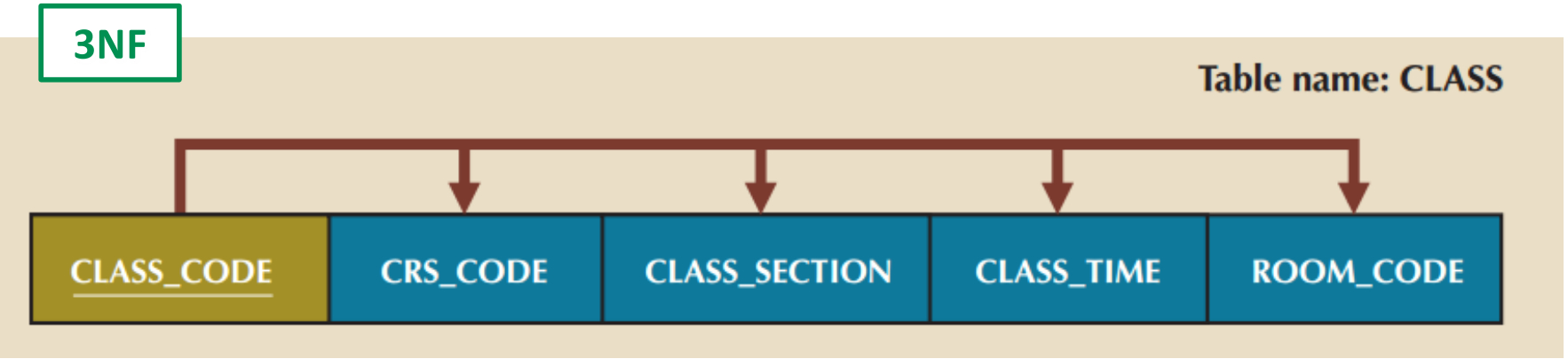
- Removing transitive dependencies.



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Example

- The CLASS table has two candidate keys:
  - **CLASS_CODE**
  - (**CRS_CODE, CLASS_SECTION**)



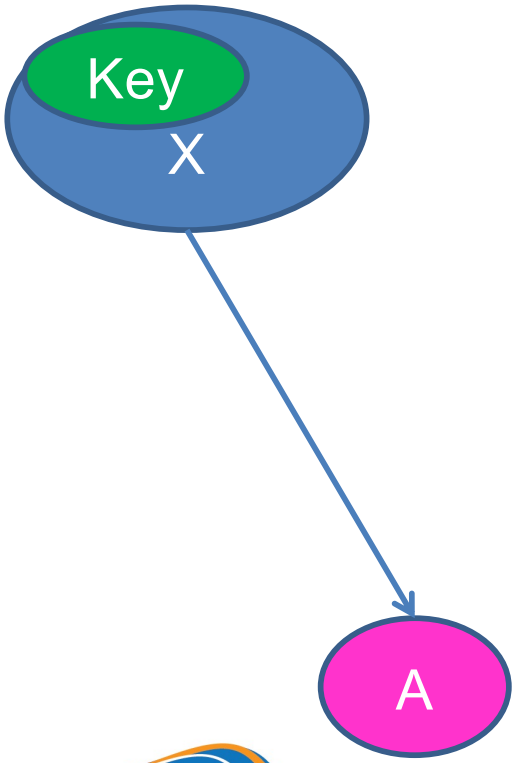
Higher-Level Normal Forms

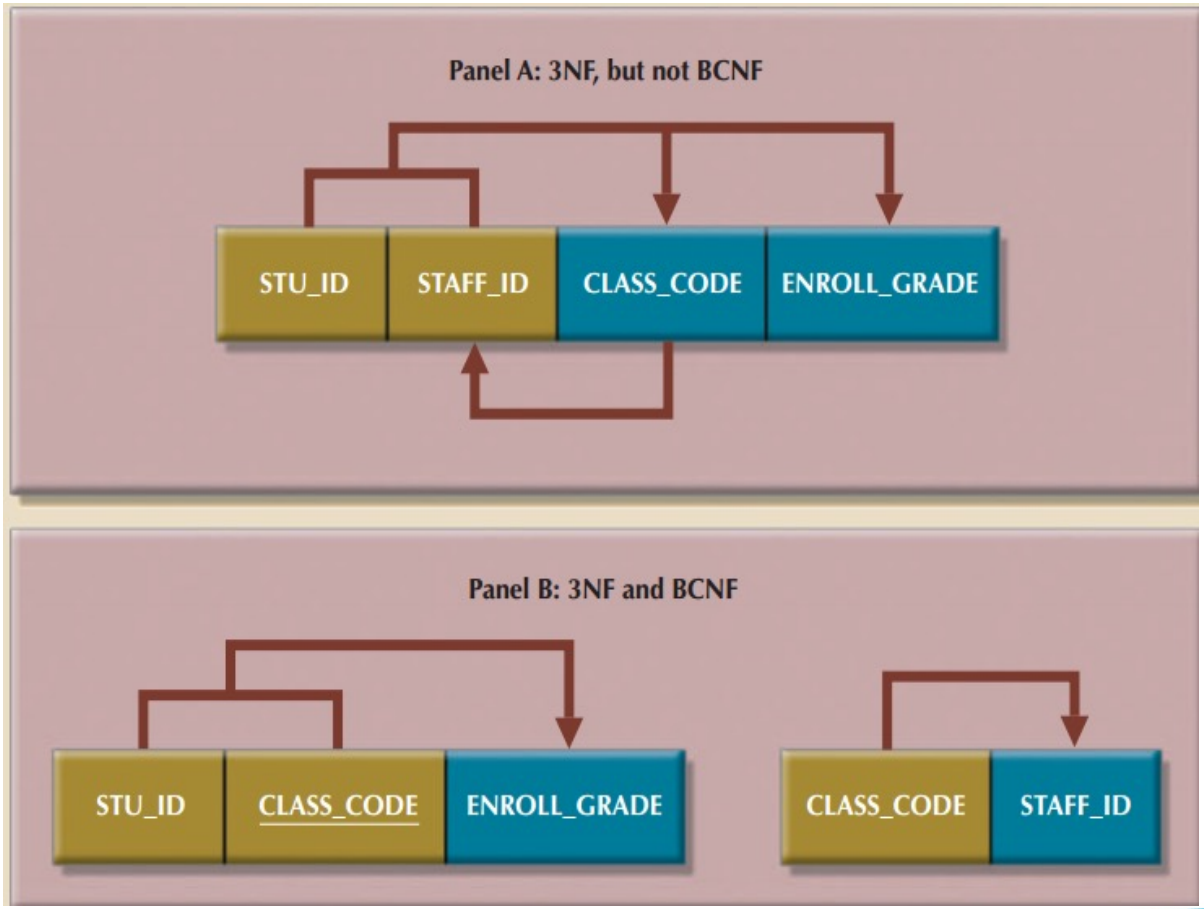
- Tables in **3NF** will perform suitably in business transactional databases. However, higher normal forms are sometimes useful.
  - **Boyce-Codd normal form (BCNF)**
  - Fourth normal form (4NF)
  - Fifth normal form (5NF)

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Boyce-Codd Normal Form (BCNF)

Def: A relation is in **BCNF** if it is in 3NF, and all determinants must be super keys.



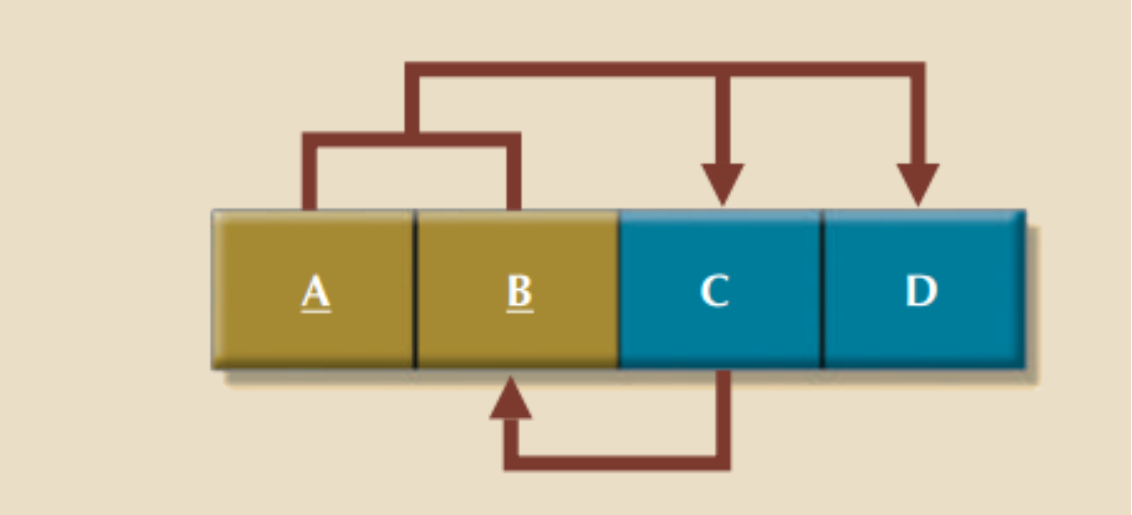


(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Example #1

R (ABCD) with F = { AB →CD, AC →BD, C →B }

Normal form of R?



- Candidate keys: AB or AC (key attributes: A, B, C)
- R satisfies 3NF (no partial or transitive dependencies between non-key attributes and any keys).
- R does not satisfy BCNF due to C → B (C is not a super key).

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Example #2

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

  U = { ABCDEF }

- F= { f1: D → B ,
  f2: A → C ,
  f3: AD → E ,
  f4: C → F }

Key: AD. Normal form of R?

Example #3

- Tìm khóa của lược đồ quan hệ :

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

  U = { ABCDEF }

- Có tập phụ thuộc hàm sau:

  F= { f1: A → D ,
  f2: C → AF ,
  f3: AB → EC
  }

Key: BA or BC. Normal form of R?

cdio 4.0

fit@hcmus

Summary of Normal Form

Normal Form	Characteristic
1NF	Table format, no repeating groups (or multivalued attributes), and PK identified.
2NF	1NF and no partial dependencies (between non-key attributes and any keys). **Hint**: If there is a functional dependency X -> Y, where X is part of any keys **and** Y is a non-key attribute holding in relation R, then R does not satisfy 2NF.
3NF	2NF and no transitive dependencies (no FD exists between non-key attributes). **Hint**: If there is a functional dependency X -> Y, where X is not a super key **and** Y is a non-key attribute holding in relation R, then R does not satisfy 3NF.
BCNF	Every determinant is a super key (special case of 3NF). **Hint**: If there is a functional dependency X -> Y, where X is not a super key holding in R, then R does not satisfy BCNF.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Practice 6: Identify keys and normal forms R = <U, F>

a. U = {ABCD}, F = { A→B, BC→D, D→A }

b. U = {ABCDEG}, F = { AD→B, EG→A, BC→G }

c. U = {ABCDE}, F = { A→BC, A→D, CD→E }

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Outline

- Fundamental Concepts
- Normal Forms (NFs)
- Normalization using Decomposition (Based on FD Theory)

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Recall: Decomposition

(Based on FD Theory)

- Decomposition: Normalization technique allows breaking up the attributes of a "bad" relation schema into smaller relations to improve its quality.

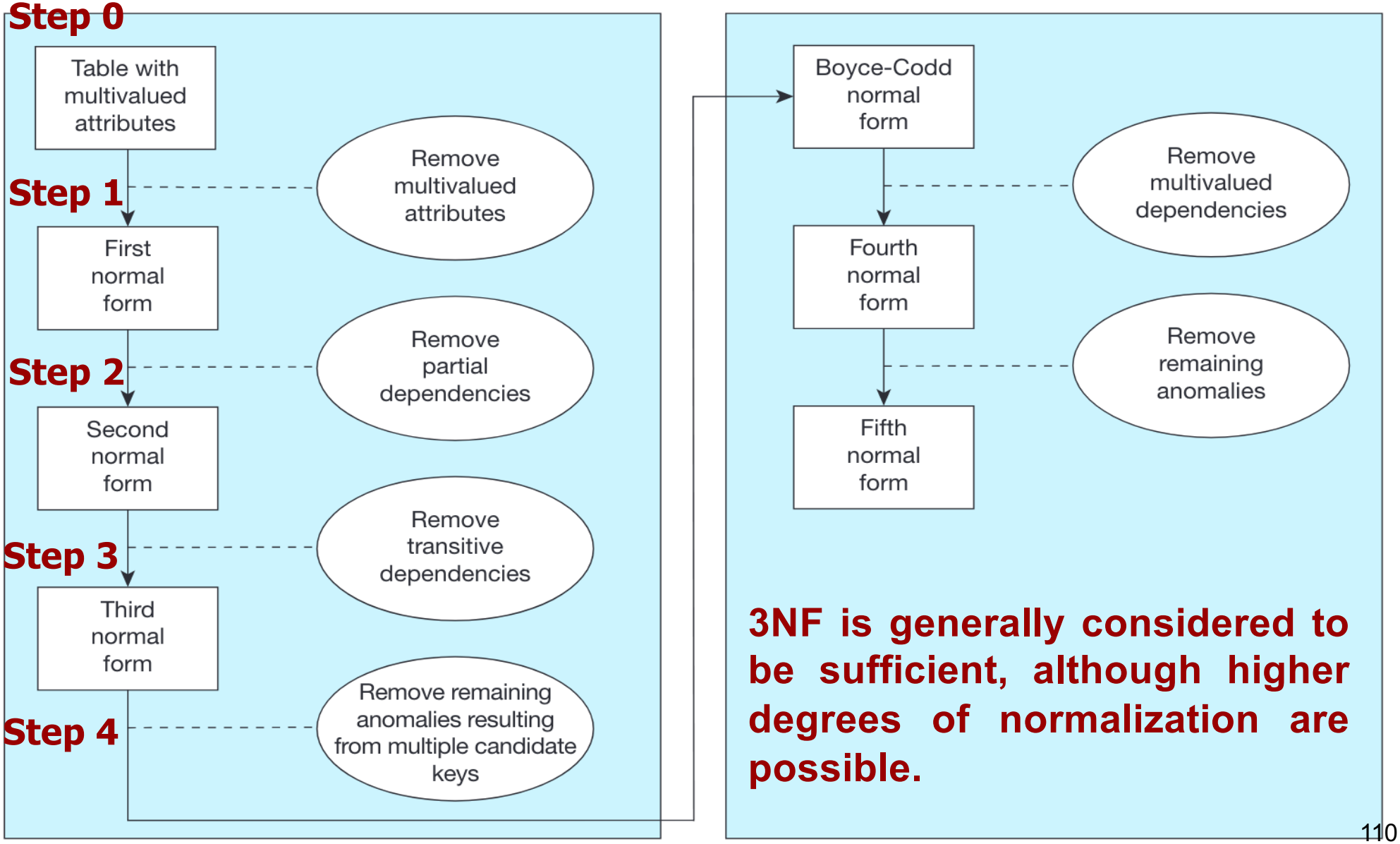
- Criteria for a decomposition:
  - Higher-level NF (3NF is acceptable).
  - Dependency-preserving decomposition.
  - Lossless-join decomposition (≠ lossy decomposition).

- A FD-based decomposition is always a lossless-join decomposition.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Decomposition Process

(Based on FD Theory)



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Case Study: Employee DB

Step 0: **Identifier** of the table Employee? **Repeating groups**?

Employee

ID	Last-Name	Department	Dependent-Name	Dependent-DOB	Dependent-Sex
322135609	Cordani	CS	Mary Cindy John	01/12/60 04/24/65 07/12/68	F F M
423542641	Strange	VP	Fern Victoria	03/28/62 11/12/84	F F
536234809	VanKlaveren	Sales	Sadie	08/31/65	F
632390802	Miller	Sales	Sallie	09/21/45	F
980772345	Kroeger	MIS	Jonathan	08/15/85	M

Case Study: Employee DB

Step 1: Remove **repeating groups** to form a schema in 1NF.

Employee (ID, Dependent-Name, …)

1NF

ID	Last-Name	Department	Dependent-Name	Dependent-DOB	Dependent-Sex
322135609	Cordani	CS	Mary	01/12/60	F
322135609	Cordani	CS	Cindy	04/24/65	F
322135609	Cordani	CS	John	07/12/68	M
423542641	Strange	VP	Fern	03/28/62	F
423542641	Strange	VP	Victoria	11/12/84	F
536234809	VanKlaveren	Sales	Sadie	08/31/65	F
632390802	Miller	Sales	Sallie	09/21/45	F
980772345	Kroeger	MIS	Jonathan	08/15/85	M

Case Study: Employee DB

Step 2: Remove **partial dependencies** to form a schema in 2NF.

Employee (ID, Dependent-Name, …)

1NF

ID	Last-Name	Department	Dependent-Name	Dependent-DOB	Dependent-Sex
322135609	Cordani	CS	Mary	01/12/60	F
322135609	Cordani	CS	Cindy	04/24/65	F
322135609	Cordani	CS	John	07/12/68	M
423542641	Strange	VP	Fern	03/28/62	F
423542641	Strange	VP	Victoria	11/12/84	F
536234809	VanKlaveren	Sales	Sadie	08/31/65	F
632390802	Miller	Sales	Sallie	09/21/45	F
980772345	Kroeger	MIS	Jonathan	08/15/85	M

Partial Dependency

Case Study: Employee DB

fit@hcmus

Step 2: Remove **partial dependencies** to form a schema in 2NF.

ID	Last-Name	Department	Dependent-Name	Dependent-DOB	Dependent-Sex

1NF

Employee

3NF

Id	Last-Name	Department
322135609	Cordani	CS
423542640	Strange	VP
536234809	VanKlaveren	Sales
632390802	Miller	Sales
980772345	Kroeger	MIS

Dependent

3NF

Emp-Id	Dependent-Name	Dependent-DOB	Dependent-Sex
322135609	Mary	01/12/60	F
322135609	Cindy	04/24/65	F
322135609	John	07/12/68	M
423542641	Fern	03/28/62	F
423542641	Victoria	11/12/84	F
536234809	Sadie	08/31/65	F
632390802	Sallie	09/21/45	F
980772345	Jonathan	08/15/85	M

fit@hcmus

Dependency Preservation

(Bảo toàn phụ thuộc hàm)

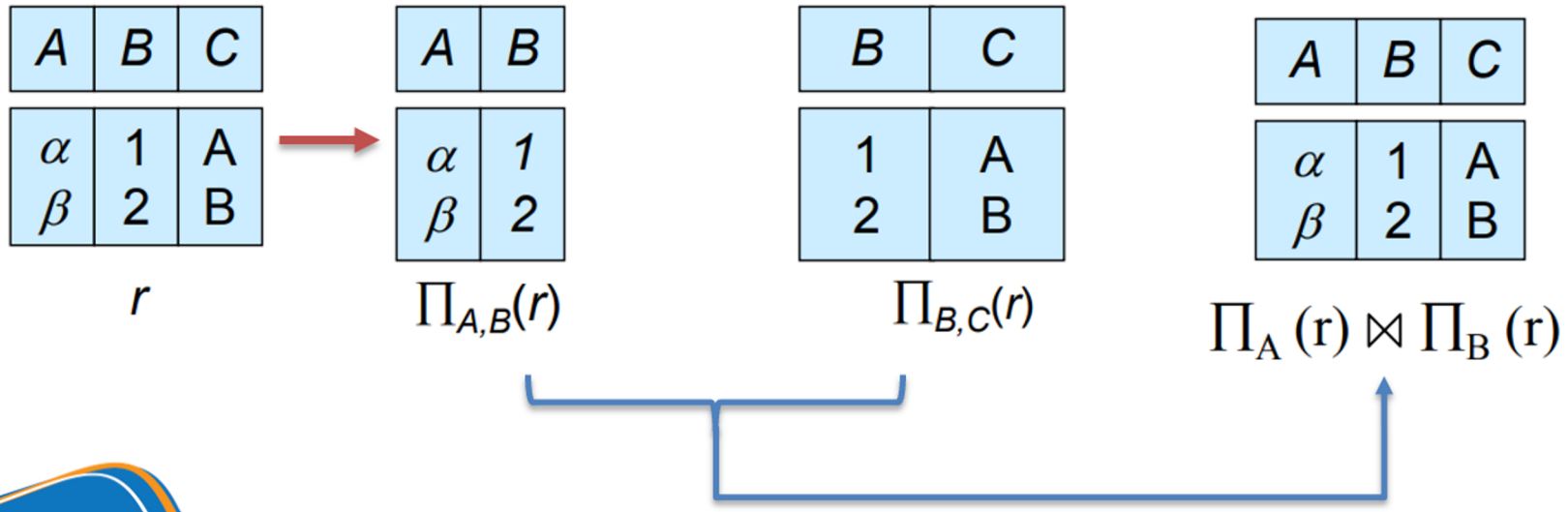
- Given R = <U, F>, suppose that R is decomposed into sub-relations R_1, R_2 , …, R_n. Let F_i be the set of dependencies F^+ that include only attributes in R_i.
- R_1, …R_n form a dependency-preserving decomposition of R if: $\mathbf{(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+}$
- If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive.

(Coronel and Morris, 2015) (Silberschatz et al. 2020)

Lossless Decomposition

(Bảo toàn thông tin)

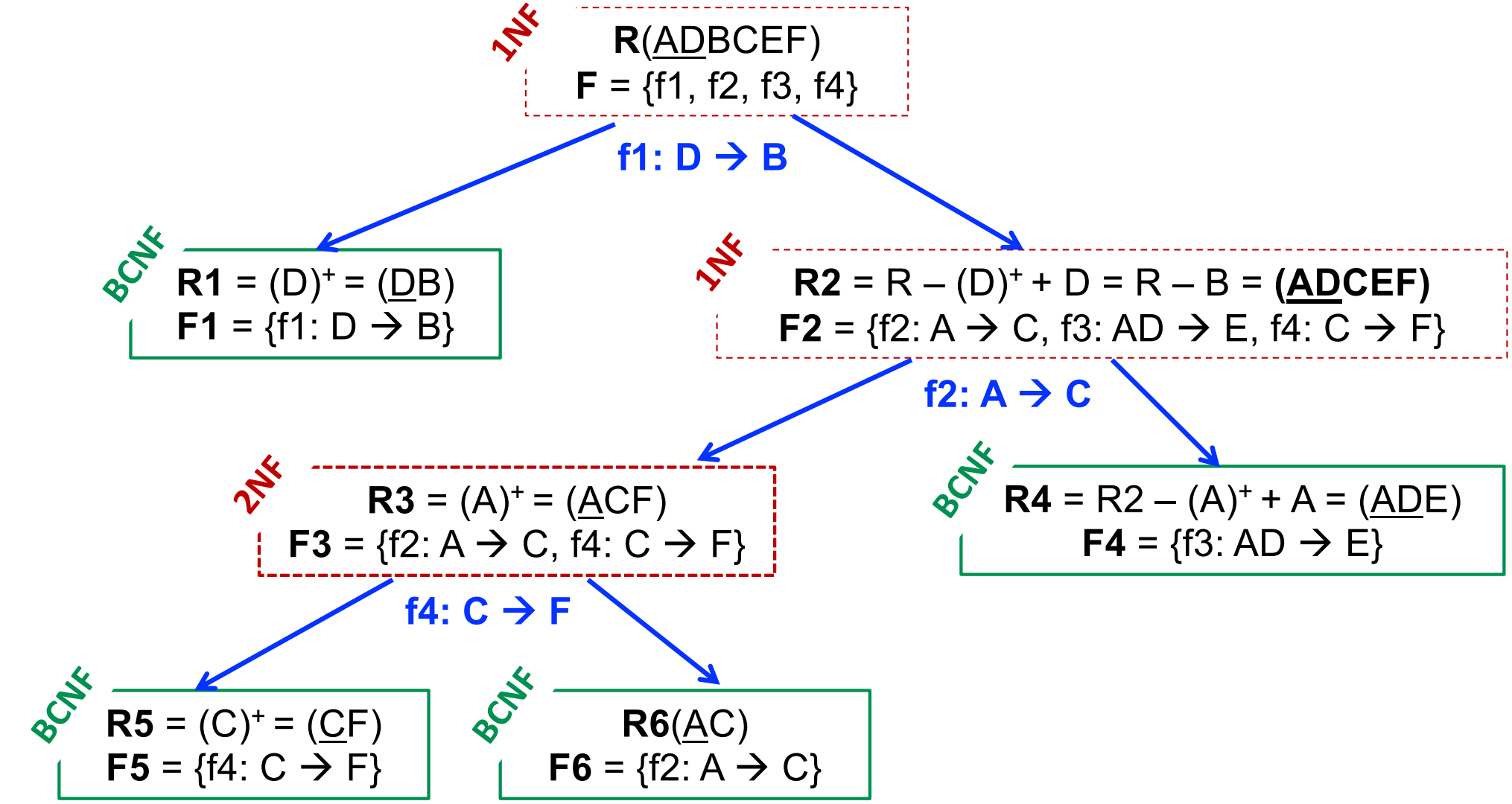
- Given R = <U, F>, suppose that R is decomposed into sub-relations R_1, R_2 , …, R_n. Let F_i be the set of dependencies F^+ that include only attributes in R_i.
- R_1, …R_n form a form a lossless decomposition of R if they can be recombined through natural join operations to form R.



(Coronel and Morris, 2015) (Silberschatz et al. 2020)

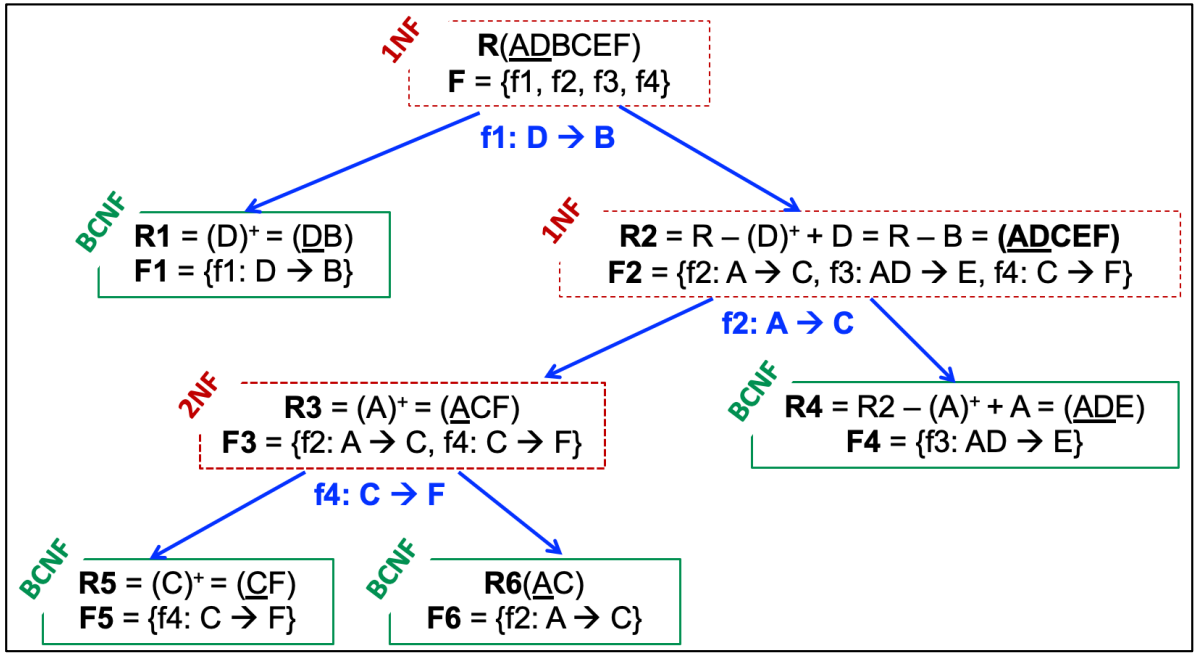
Practice 7: Decomposition

R (ADBCEF), F = { f1: D → B, f2: A → C, f3: AD → E, f4: C → F }



Practice 7: Decomposition (cont.)

R (ADBCEF), F = { f1: D → B, f2: A → C, f3: AD → E, f4: C → F }



R(ADBCEF), **F** = {f1, f2, f3, f4} 1NF

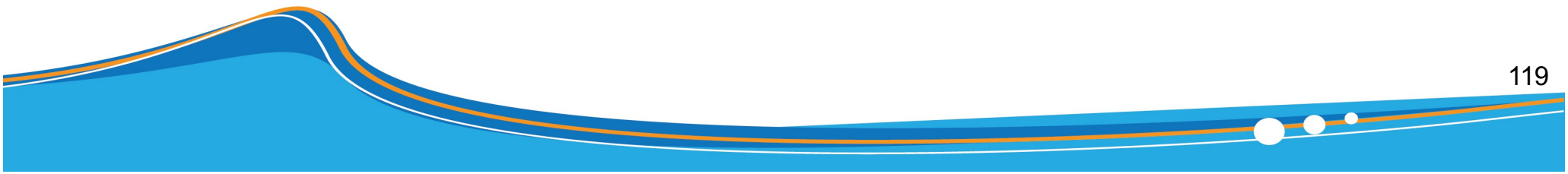
- Higher-level (always) ☑
- Lossless join (always) ☑
- Dependency preservation? ☑

BCNF

R1(DB), **F1** = {f1: D → B} BCNF
R4(ADE), **F4** = {f3: AD → E} BCNF
R5(CF), **F5** = {f4: AD → E} BCNF
R6(AC), **F6** = {f2: A → C} BCNF

Practice 8: Normalization

R (ABCDEFGHIJ)
F = { AB → C, A → DE, B → F, F → GH, D → IJ }
a. Determine keys of R.
b. Determine the normal form of R?
c. Decompose R to obtain a schema in 3NF or higher.
d. Is this a lossless decomposition? Explain.
e. Is this a dependency-preserving decomposition? Explain.



THE END