INFORMATION SYSTEM DEPARTMENT
INFORMATION TECHNOLOGY FACULTY
UNIVERSITY OF SCIENCE – VNUHCM

ADVANCED DATABASE

Chapter 05

LOGICAL DESIGN

Dr. **VU** Thi My Hang

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

fit@hcmus

Outline

- Overview
- Main Steps
- Fundamentals
- ERD to Relations Mapping

cdio 4.0

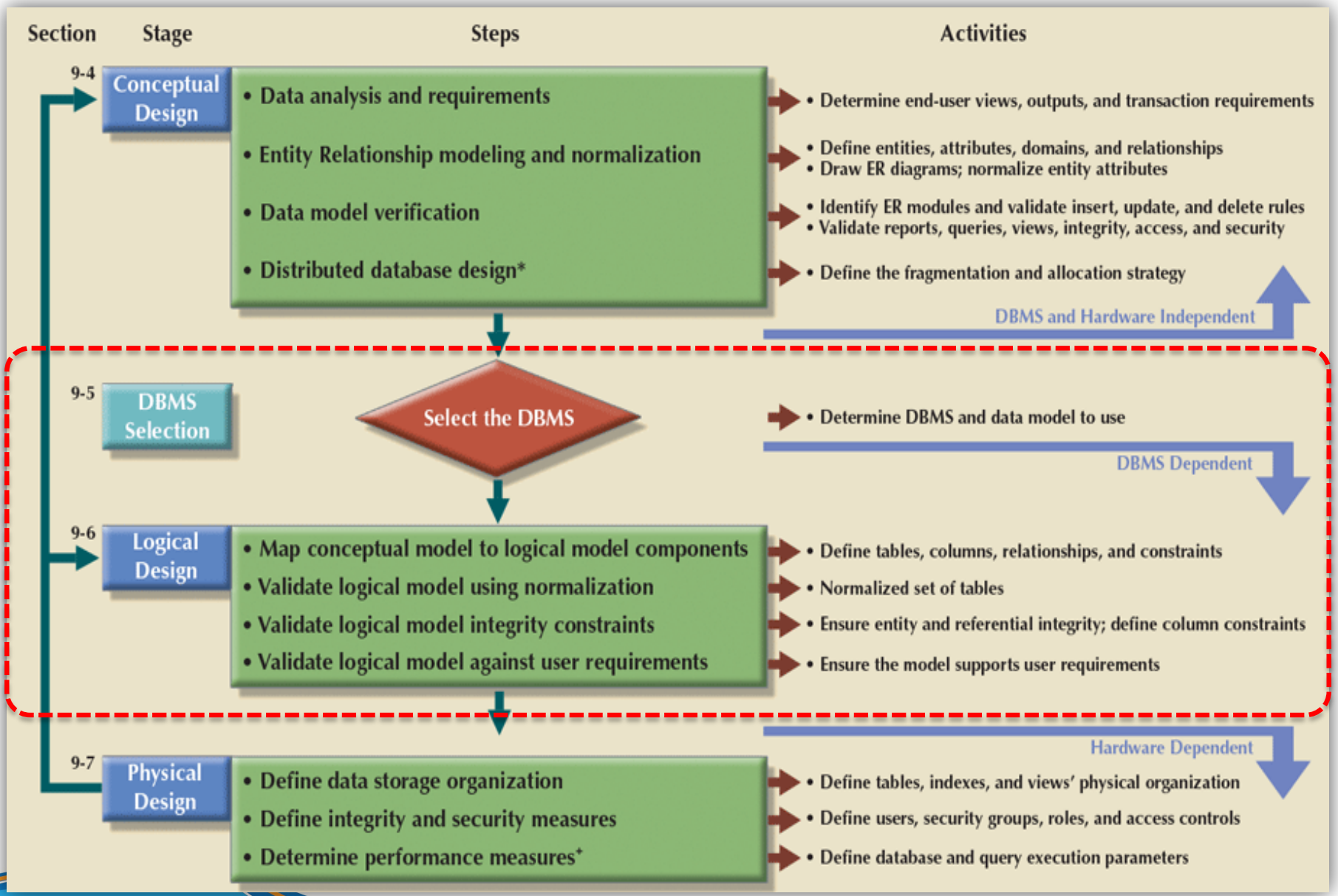
fit@hcmus

Outline

- Overview
- Main Steps
- Fundamentals
- ERD to Relations Mapping

Recall: Database Design Process

fit@hcmus



(Coronel and Morris, 2015; Silberschatz et al., 2020; Hoffer et al., 2019)

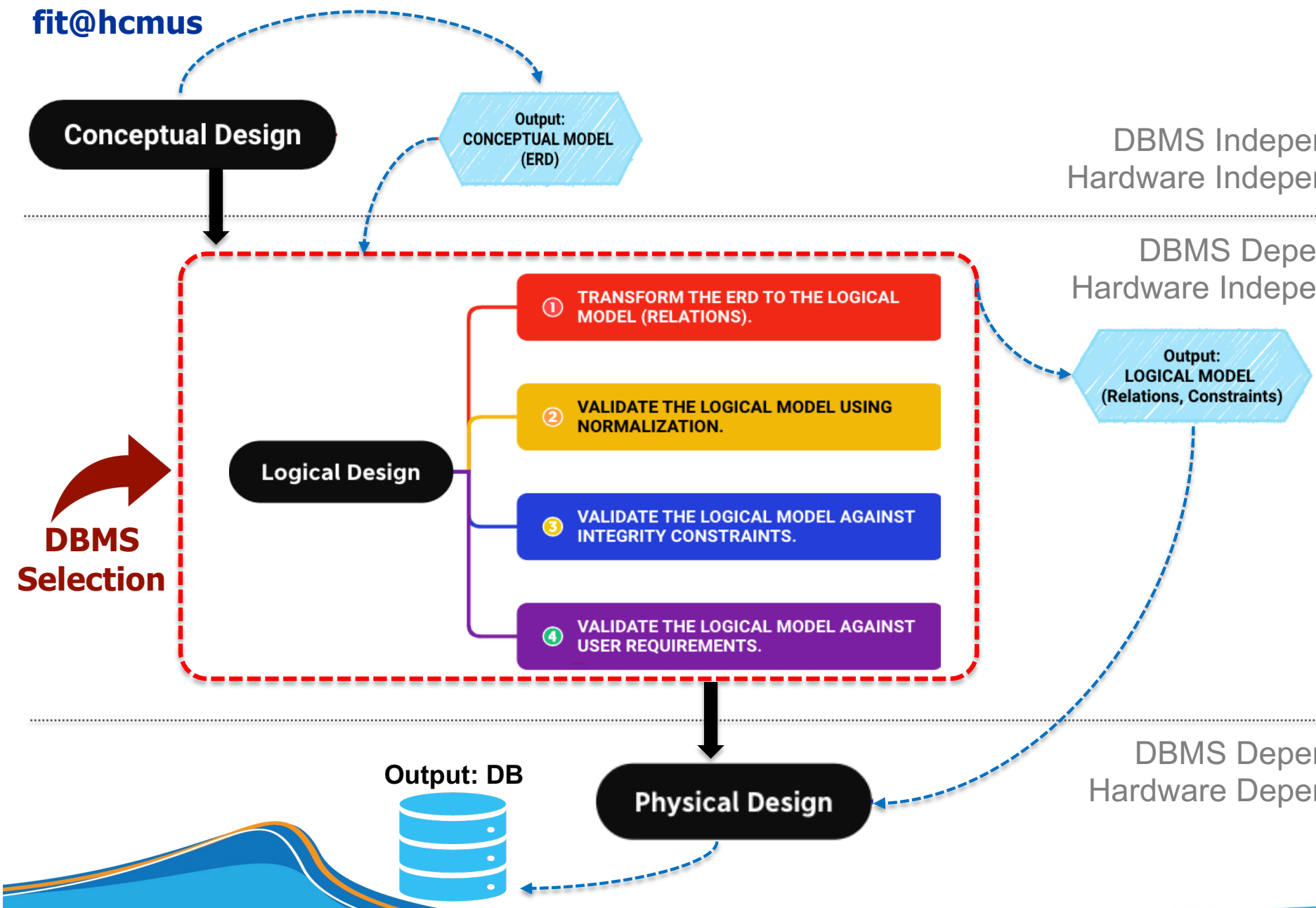
fit@hcmus

Logical Design

- **Definition**: Process of transforming the conceptual model into a **normalized logical model**. The emphasis in this process has been on the relational data model.
- **Goal:** Design an enterprise-wide database that is based on a specific data model and/or DBMS but independent of physical-level details (implementation).
- **Output**: Specifications of relations (tables) & constraints based on a DBMS.
- **Tools**: Relational model, functional dependency theory, and normalization theory.

(Coronel and Morris, 2015; Silberschatz et al., 2020; Hoffer et al., 2019)

Logical Design



(Coronel and Morris, 2015; Silberschatz et al., 2020; Hoffer et al., 2019)

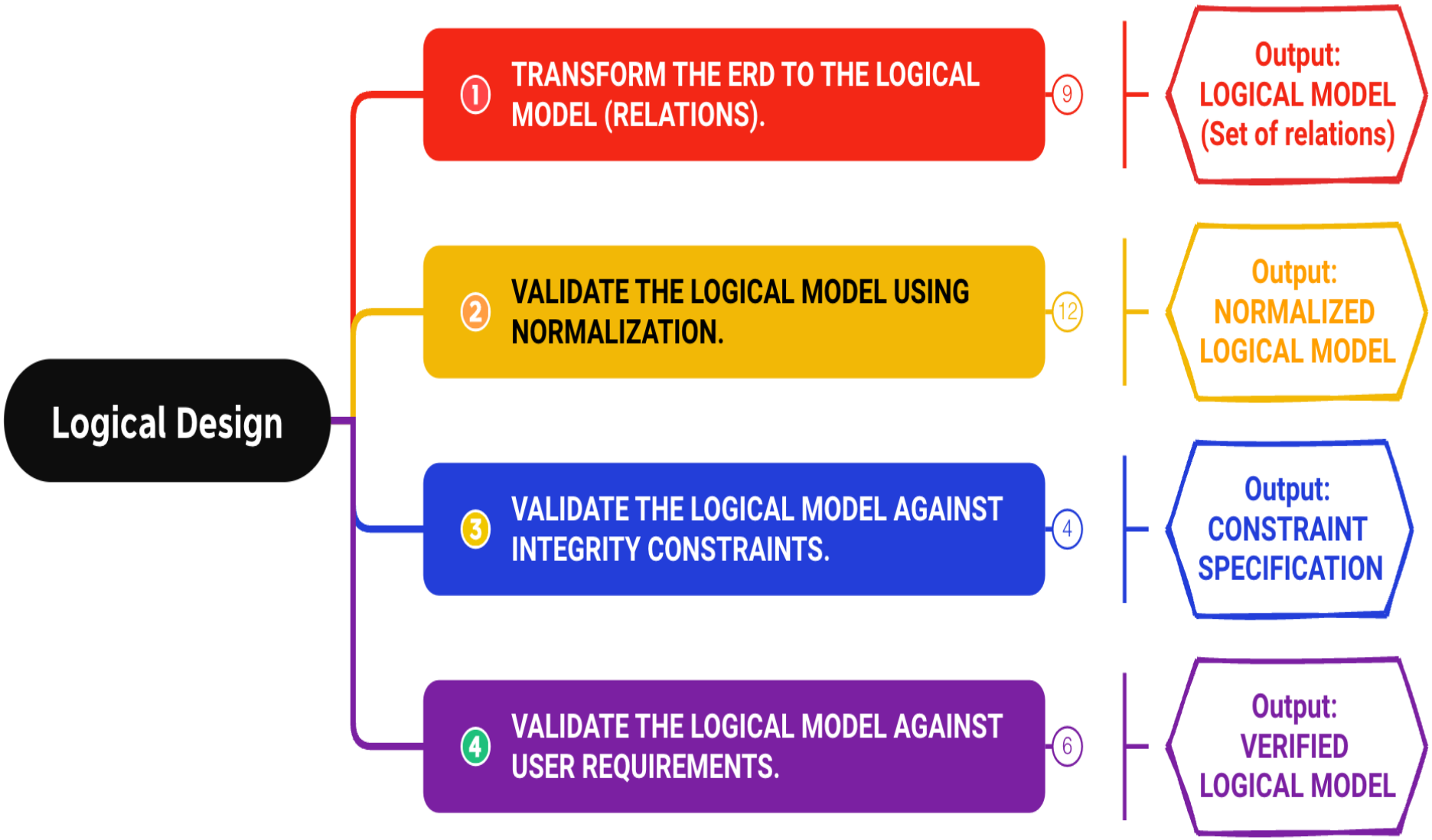
cdio 4.0

fit@hcmus

Outline

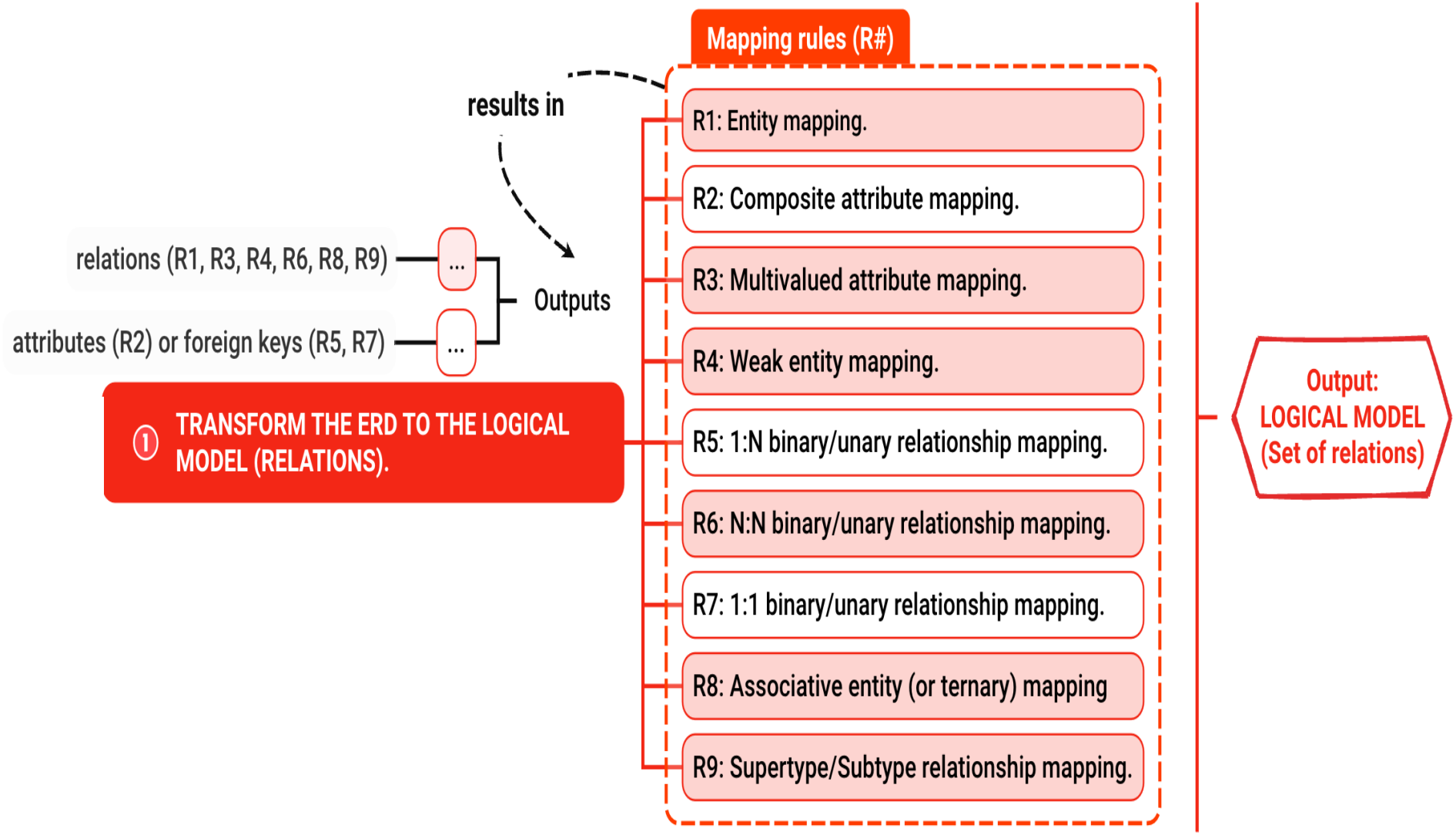
- Overview
- Main Steps
- Fundamentals
- ERD to Relations Mapping

Main Steps


Logical Design
1 TRANSFORM THE ERD TO THE LOGICAL MODEL (RELATIONS).
9
Output: LOGICAL MODEL (Set of relations)
2 VALIDATE THE LOGICAL MODEL USING NORMALIZATION.
12
Output: NORMALIZED LOGICAL MODEL
3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.
4
Output: CONSTRAINT SPECIFICATION
4 VALIDATE THE LOGICAL MODEL AGAINST USER REQUIREMENTS.
6
Output: VERIFIED LOGICAL MODEL

Main Steps

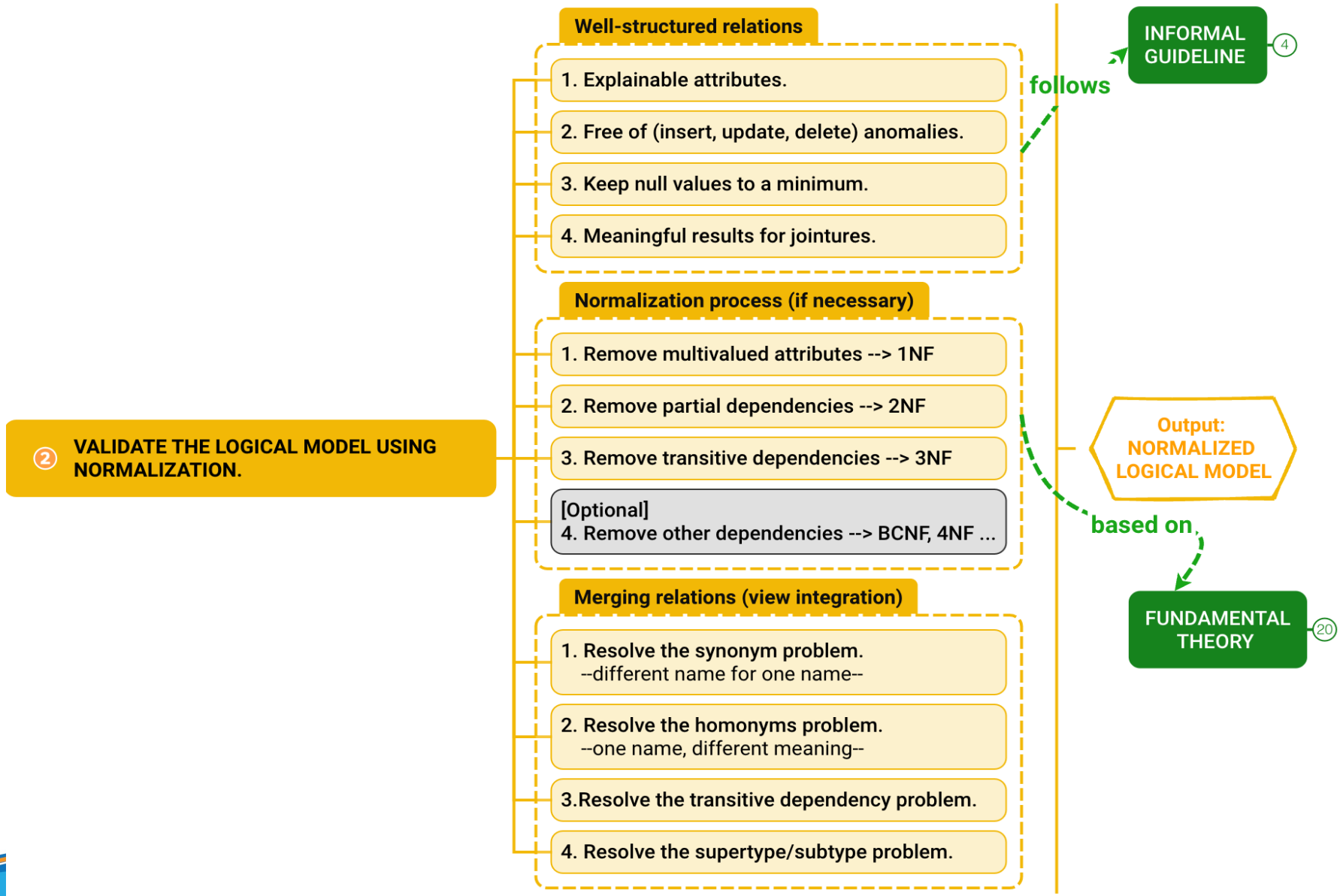
Step 1: Map the conceptual model to logical model.



fit@hcmus

Main Steps

Step 2: Validate the logical model using normalization.



Main Steps

Step 2: Validate the logical model using normalization.

Merging relations (view integration)

- On large projects, the work of several sub-teams comes together during logical design, so there is often a need to merge relations.
- Integrating existing databases with new information requirements often leads to the need to integrate different views.
- New data requirements may arise during the life cycle, so there is a need to merge any new relations with what has already been developed.

(Coronel and Morris, 2015)

fit@hcmus

Main Steps

Step 2: Validate the logical model using normalization.

Merging relations (view integration)

1. Resolve the synonym problem.
   --different name for one name--
2. Resolve the homonyms problem.
   --one name, different meaning--

→ Obtain agreement (if possible) from users on a single, standardized name for the conflict attribute.

Synonym (different names but same meaning) (**Đồng nghĩa** khác âm)

STUDENT1(StudentID, Name)
STUDENT2(MatriculationNo, Name, Address)

→ STUDENT(StudentNo, Name, Address)

Homonym (one attribute name but different meaning) (**Đồng âm** khác nghĩa)

STUDENT1(StudentID, Name, Address)
STUDENT2(StudentID, Name, PhoneNo, Address)

↓

STUDENT(StudentID, Name, PhoneNo, CampusAddress, PermanentAddress)

(Coronel and Morris, 2015)

fit@hcmus

Main Steps

Step 2: Validate the logical model using normalization.

Merging relations (view integration)

3.Resolve the transitive dependency problem.

When two 3NF relations are merged to form a single relation, transitive dependencies may result.
➔ Consider restructure the schema.

STUDENT1(StudentID, MajorName)
STUDENT2(StudentID, Advisor)

STUDENT(StudentID, MajorName, Advisor)

What if we have FD: MajorName → Advisor?

STUDENT(StudentID, MajorName)
MAJOR (MajorName, Advisor)

(Coronel and Morris, 2015)

fit@hcmus

Main Steps

Step 2: Validate the logical model using normalization.

Merging relations (view integration)

4. Resolve the supertype/subtype problem.

These relationships may be hidden in user views or relations.

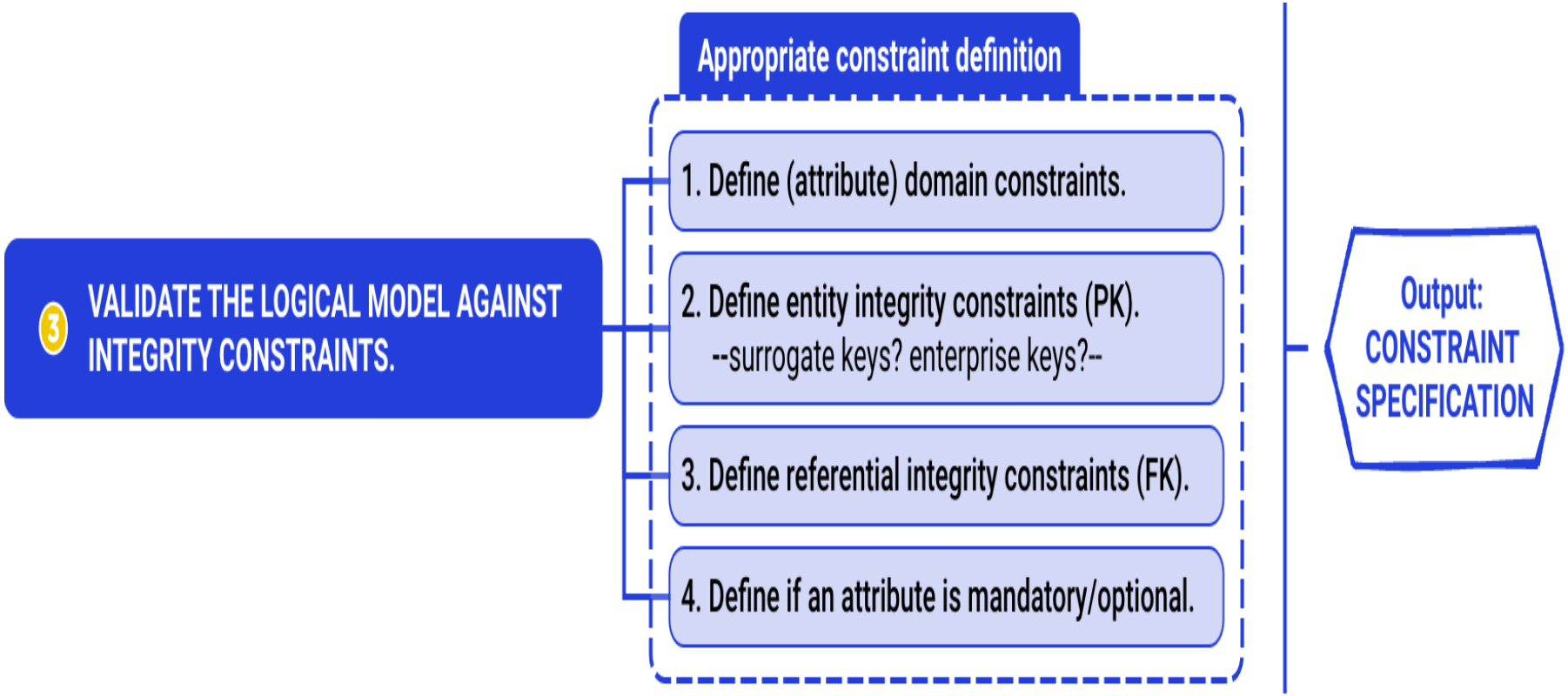
PATIENT1(PatientID, Name, Address)
PATIENT2(PatientID, RoomNo)

PATIENT(PatientID, Name, Address)
RESIDENTPATIENT(PatientID, RoomNo)
OUTPATIENT(PatientID, DateTreated)

(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.



Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

1. Define (attribute) domain constraints.

Domain Constraints

- All of the values that appear in a column of a relation must be from the same domain, i.e., set of values assigned to an attribute.
- A domain definition should clarify domain name, meaning, data type, size, and allowable values/ranges.

CLASS_CODE	is a valid class code.
	Type: numeric
	Range: low value=1000 high value=9999
	Display format: 9999
	Length: 4
CLASS_DAYS	is a valid day code.
	Type: character
	Display format: XXX
	Valid entries: MWF, TR, M, T, W, R, F, S
	Length: 3
CLASS_TIME	is a valid time.
	Type: character
	Display format: 99:99 (24-hour clock)
	Display range: 06:00 to 22:00
	Length: 5

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

2. Define entity integrity constraints (PK).
--surrogate keys? enterprise keys?--

Entity Integrity Constraints

- Every relation has a **primary key** and that the data values for that primary key are all valid.

(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

2. Define entity integrity constraints (PK).
--surrogate keys? enterprise keys?--

Guidelines for Primary Key Selection

- **Unique values**
  --Uniquely identify each entity instance--
- **Non-intelligent**
  --No embedded semantic meaning--
- **No change over time**
  --Should be permanent and unchangeable--
- **(Preferably) single-attribute**
  --Surrogate key could be considered to replace a composite key--
- **(Preferably) numeric value**
  --Better for data processing at the physical level--
- **Security-compliant**
  --No security risk or violation--

(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

2. Define entity integrity constraints (PK).
--surrogate keys? enterprise keys?--

Surrogate Keys

- Key generated by systems to simplify the identification of entity instances.
- Have no meaning outside of the system.
- Useful for replacing composite keys or long, single-string keys.

Patient_Treatment

PatientID	PhysicianID	TreatmentCode	PTreatmentDate	PTreatmentTime	PTreatmentResults

Use **IDTreatment** as a surrogate key

(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

2. Define entity integrity constraints (PK).
--surrogate keys? enterprise keys?--

Enterprise Key (Object Identifier)

- Primary keys which are unique across the whole database.

➔ Facilitate data evolvability.

EMPLOYEE(EmpID, EmpName, DeptName, Salary)
CUSTOMER(CustID, CustName, Address)

Organizations need to recognize that employees can also be customers.

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, EmpName, DeptName, Salary)
CUSTOMER (OID, CustID, CustName, Address)

(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

2. Define entity integrity constraints (PK).
--surrogate keys? enterprise keys?--

Enterprise Key (Object Identifier)

- Primary keys which are unique across the whole database.

➔ Facilitate data evolvability.

EMPLOYEE

OID	EmpID	EmpName	DeptName	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

OID	CustID	CustName	Address
2	100	Fred's Warehouse	Greensboro, NC
3	101	Bargain Bonanza	Moscow, ID
6	102	Jasper's	Tallahassee, FL
7	103	Desks 'R Us	Kettering, OH

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, EmpName, DeptName, Salary)
CUSTOMER (OID, CustID, CustName, Address)

OBJECT

OID	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

(Coronel and Morris, 2015)

Main Steps

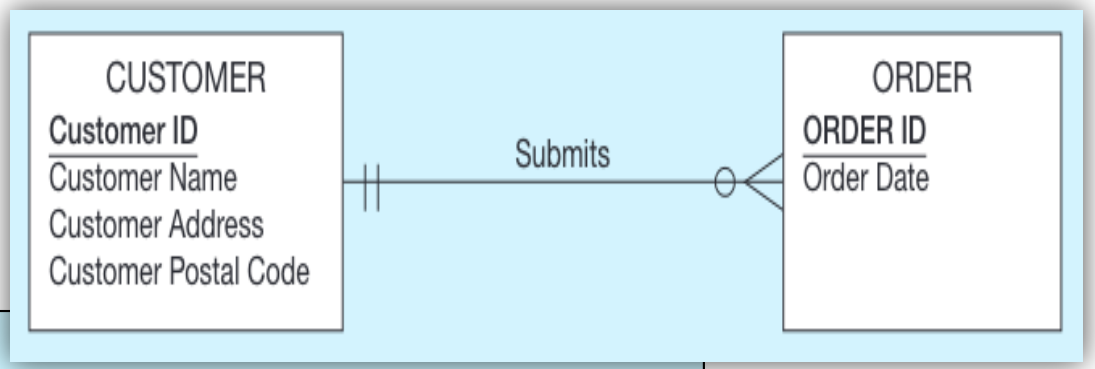
Step 3: Validate the logical model integrity constraints.

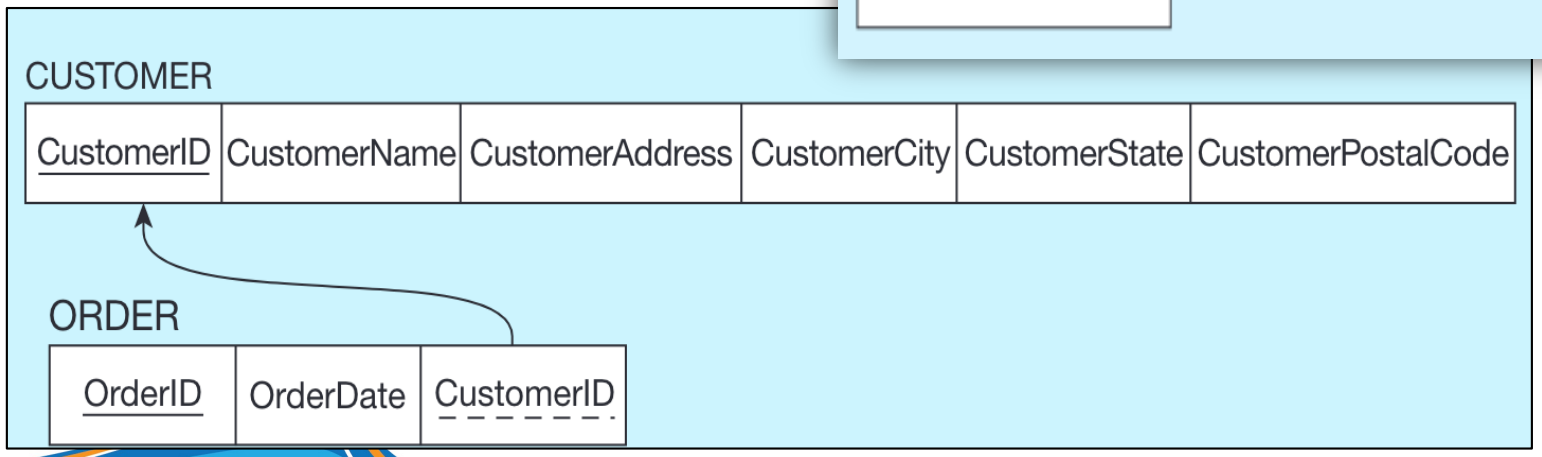
3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

3. Define referential integrity constraints (FK).

Referential Integrity Constraints

- Each foreign key value of a relation **MUST** match a primary key value in another relation **or** be null.





(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

3. Define referential integrity constraints (FK).

Referential Integrity Constraints

- Each foreign key value of a relation **MUST** match a primary key value in another relation **or** be null.

What happens if we delete a customer who has submitted orders?

- **Delete the associated** orders (called a **cascading delete**)
- **Prohibit deletion** of the customer until all associated orders are deleted (a **safety check**).
- **Place a null value** in the foreign key (an exception stating that although an order must have a CustomerID value when the order is created, CustomerID can become null later if the associated customer is deleted).

(Coronel and Morris, 2015)

fit@hcmus

Main Steps

Step 3: Validate the logical model integrity constraints.

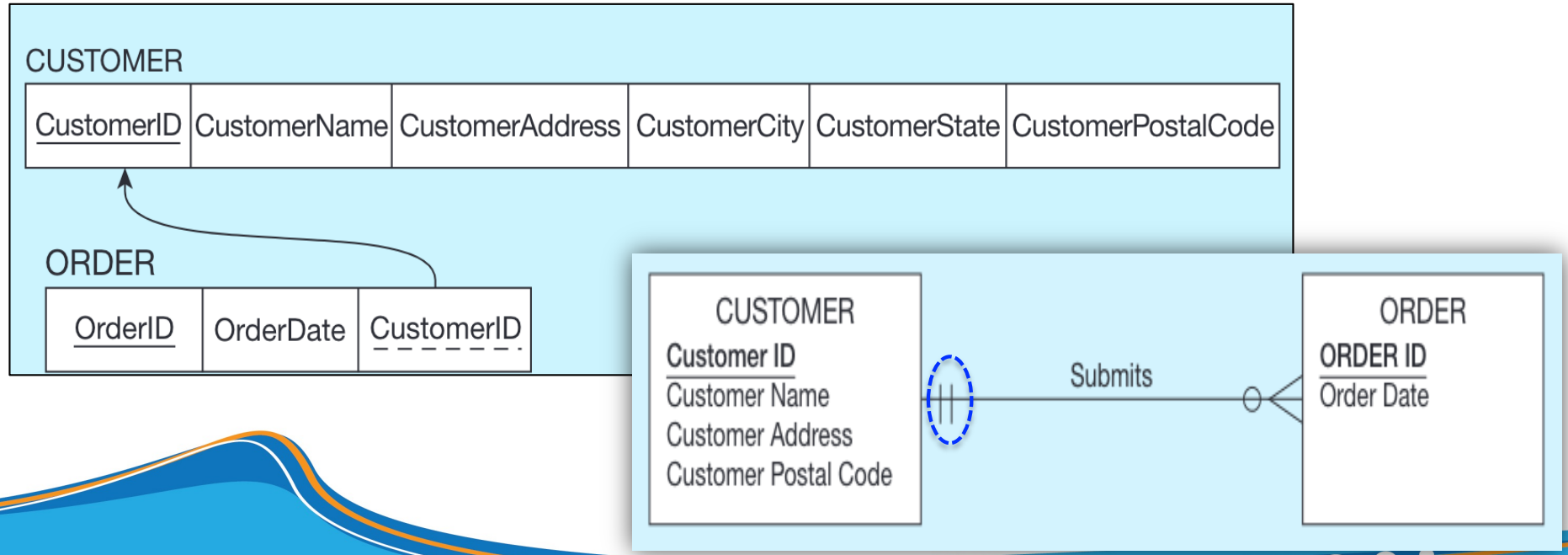
3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

4. Define if an attribute is mandatory/optional.

Mandatory/Optional Constraints

- Define which attributes are mandatory and which are optional while ensuring entity (PK) and referential integrity (FK) constraints.

How do you know whether a foreign key is allowed to be null?



(Coronel and Morris, 2015)

Main Steps

Step 3: Validate the logical model integrity constraints.

3 VALIDATE THE LOGICAL MODEL AGAINST INTEGRITY CONSTRAINTS.

4. Define if an attribute is mandatory/optional.

Mandatory/Optional Constraints

- Define which attributes are mandatory and which are optional while ensuring entity (PK) and referential integrity (FK) constraints.

Sample tables (usually created during the physical design phase)

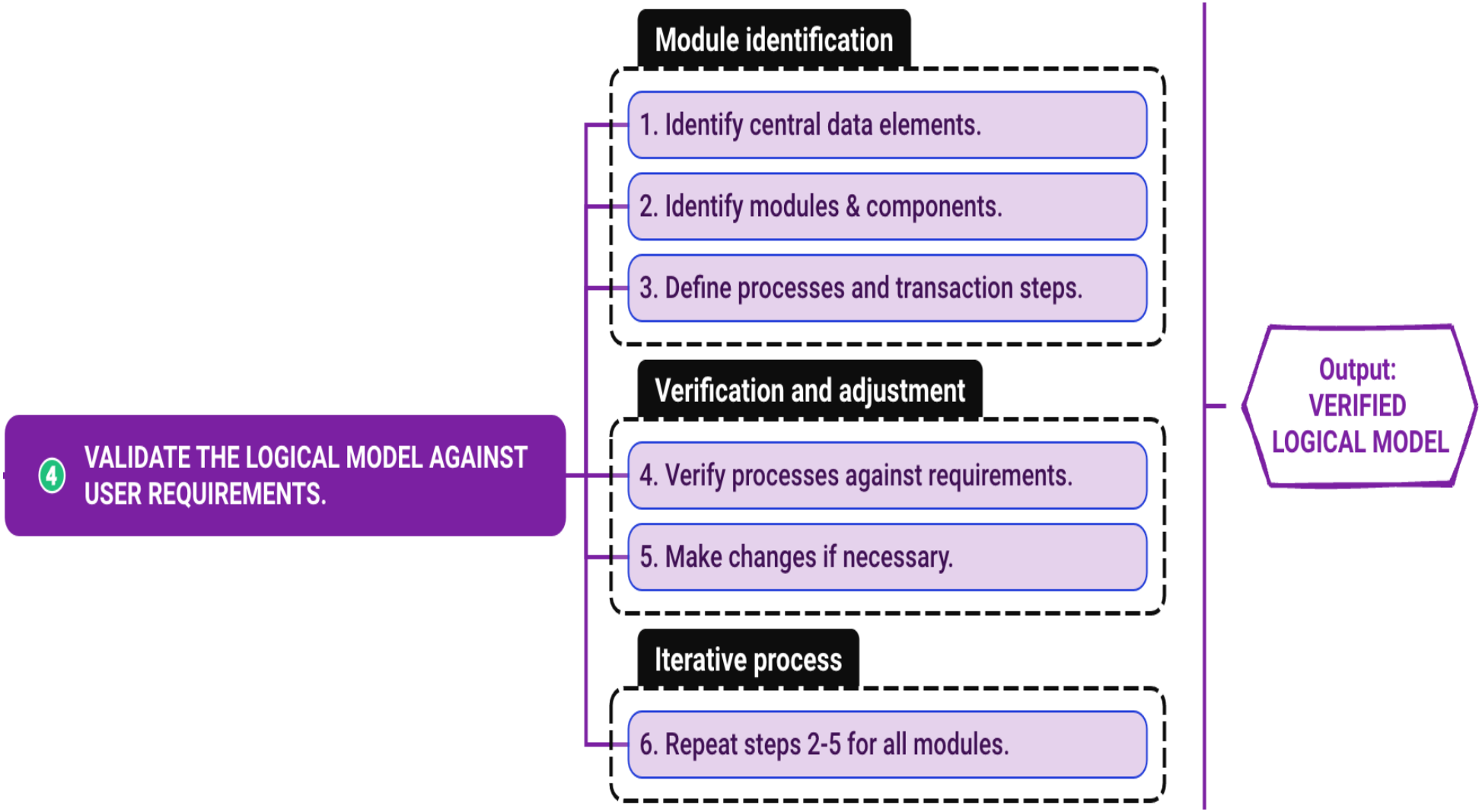
```
CREATE TABLE Customer_T
        (CustomerID                    NUMBER(11,0)      NOT NULL,
        CustomerName                   VARCHAR2(25)      NOT NULL,
        CustomerAddress                VARCHAR2(30),
        CustomerCity                   VARCHAR2(20),
        CustomerState                  CHAR(2),
        CustomerPostalCode             VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                       NUMBER(11,0)      NOT NULL,
        OrderDate                      DATE DEFAULT SYSDATE,
        CustomerID                     NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));
```

(Coronel and Morris, 2015)

Main Steps

Step 4: Validate the logical model against user requirement.



Outline

- Overview
- Main Steps
- Fundamentals
- ERD to Relations Mapping

Recall: Relational Model

- 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

(Coronel and Morris, 2015)

cdio 4.0

fit@hcmus

Recall: Relational Schema

- A relational (database) schema is a set of table definitions and integrity constraints for organizing data in relations.
- 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 the foreign key value must be null. This allows maintaining consistency among the rows of the two relations.

(Coronel and Morris, 2015)

cdio 4.0

fit@hcmus

Recall: Relational Schema

Example of domain definition

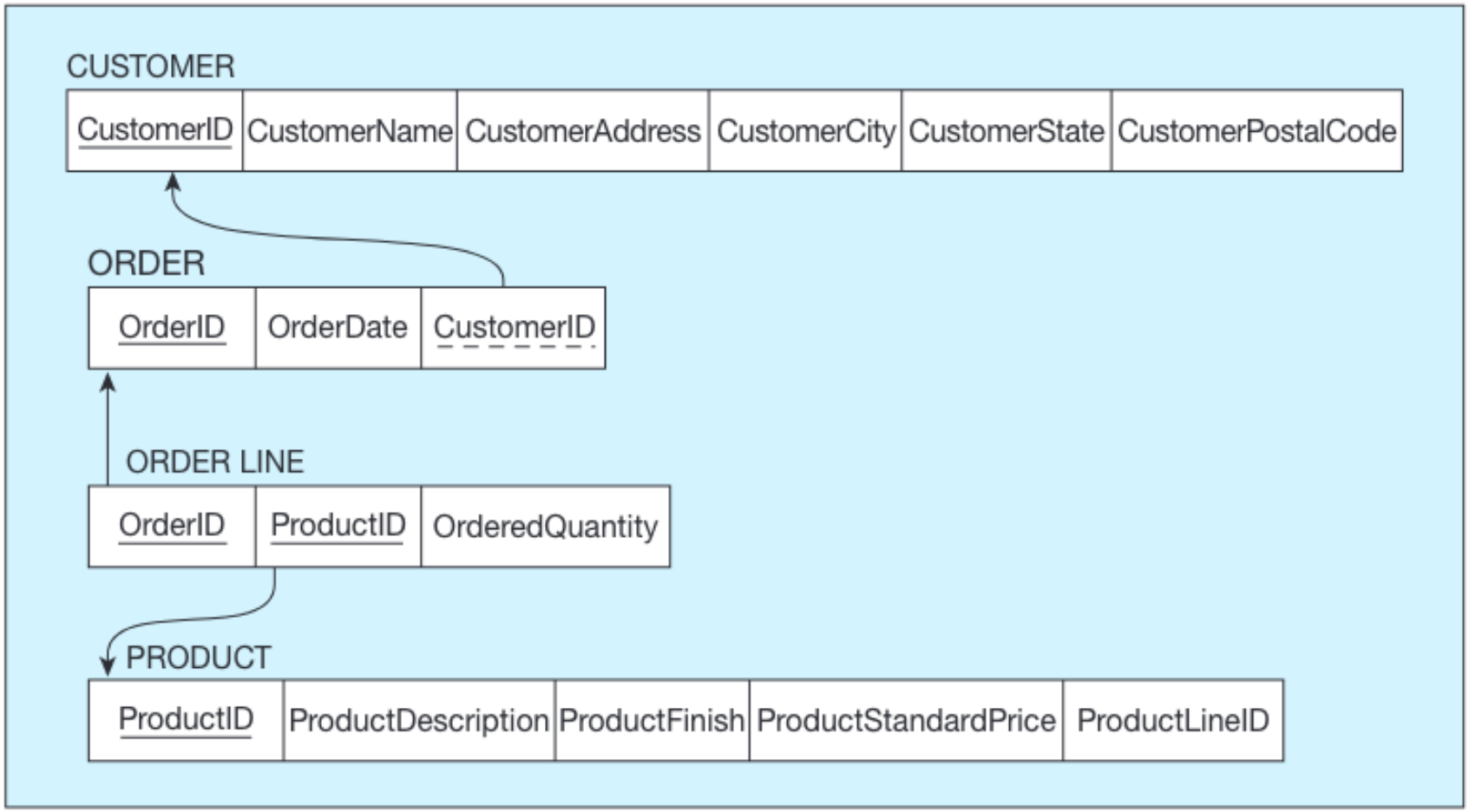
TABLE 4-1 Domain Definitions for INVOICE Attributes

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

(Coronel and Morris, 2015)

Recall: Relational Schema

Example of primary and foreign keys



(Coronel and Morris, 2015)

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)

cdio 4.0

fit@hcmus

Normalization

cf. Chapter 6

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.

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.

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

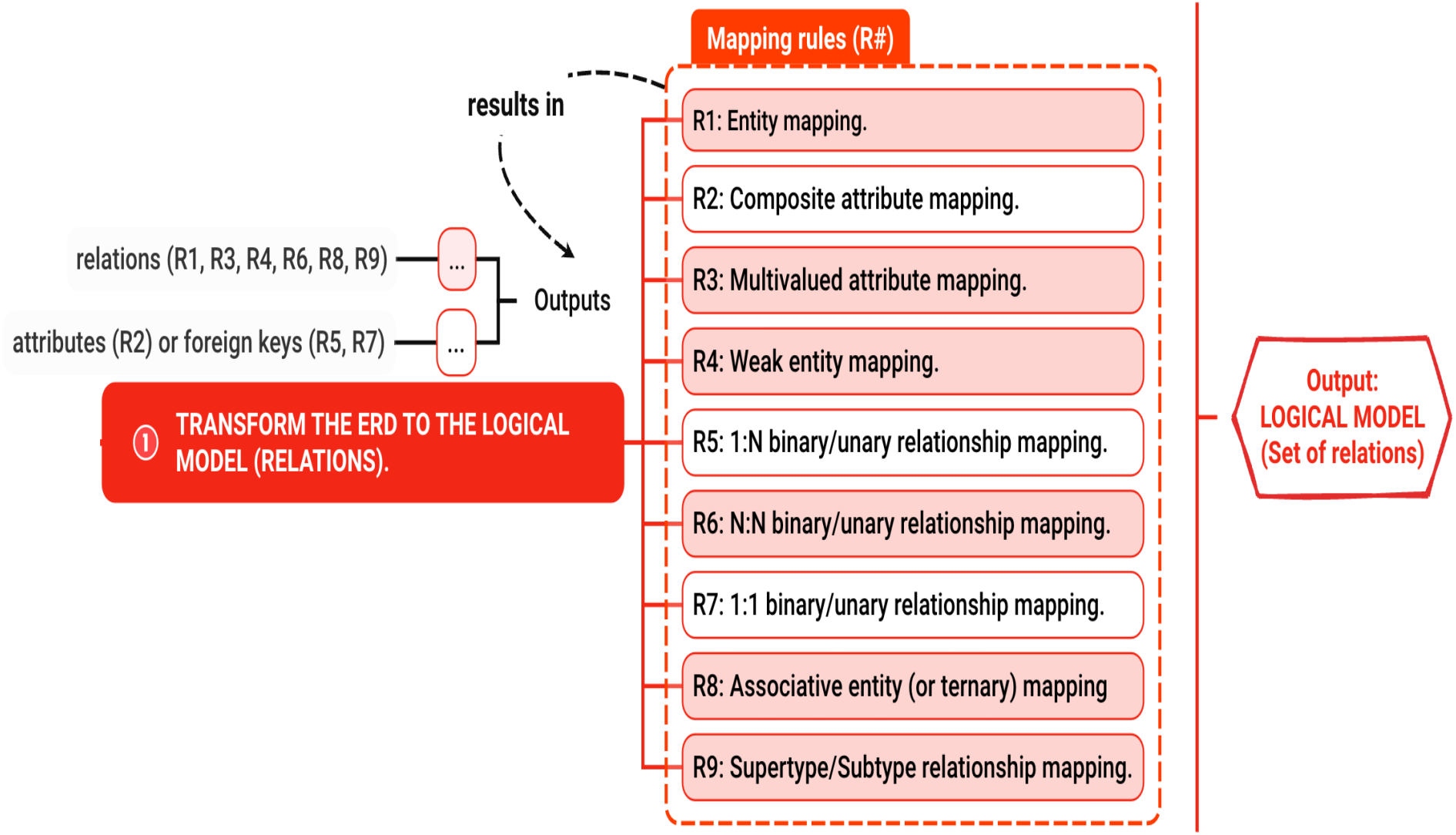
FUNCTIONAL DEPENDENCY (FD) THEORY

- **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**

Outline

- Overview
- Main Steps
- Fundamentals
- ERD to Relations Mapping

Mapping Rules

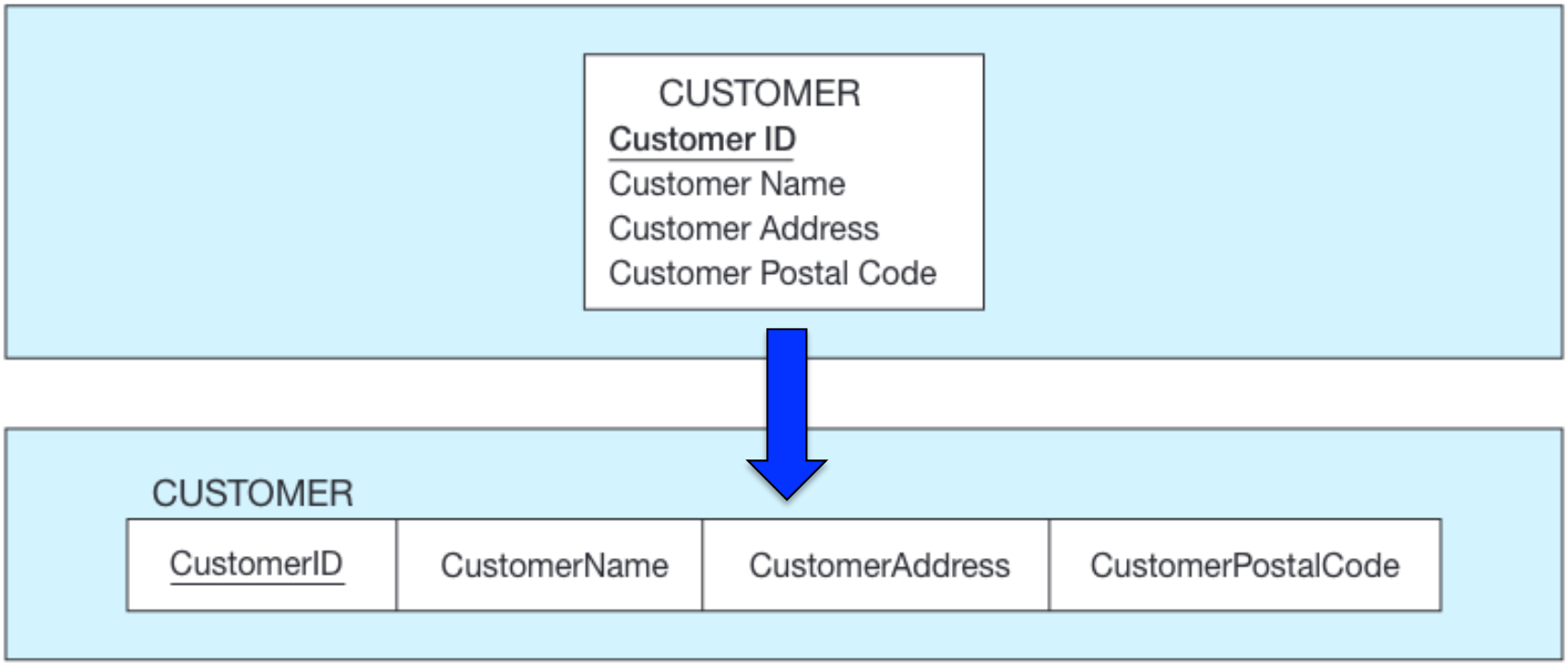


Mapping Rules

R1: (Regular/Strong) Entity mapping

Each (regular/strong) entity is transformed into a relation.

- Name of the entity ⇨ Name of the relation
- Identifier of the entity ⇨ Primary key of the relation
- Atomic attributes (simple, single-valued) ⇨ Attributes of the relation



(Coronel and Morris, 2015)

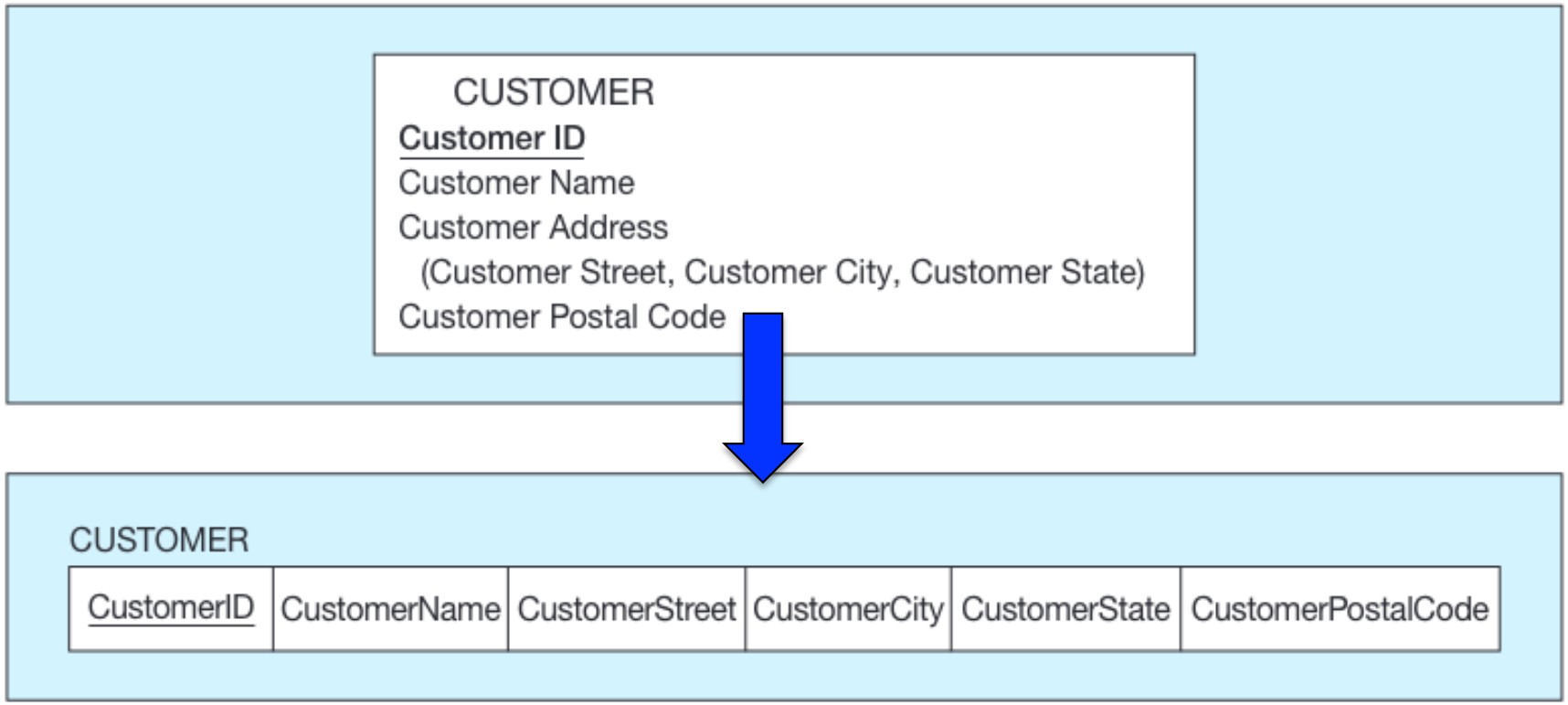
Mapping Rules

R2: Composite attribute mapping

Only the simple components of the composite attribute are included as the relation's attributes.

- Another possible solution is to use the composite attribute as a single attribute in the new relation rather than its components.

  ➔ The selection depends on transaction requirements.



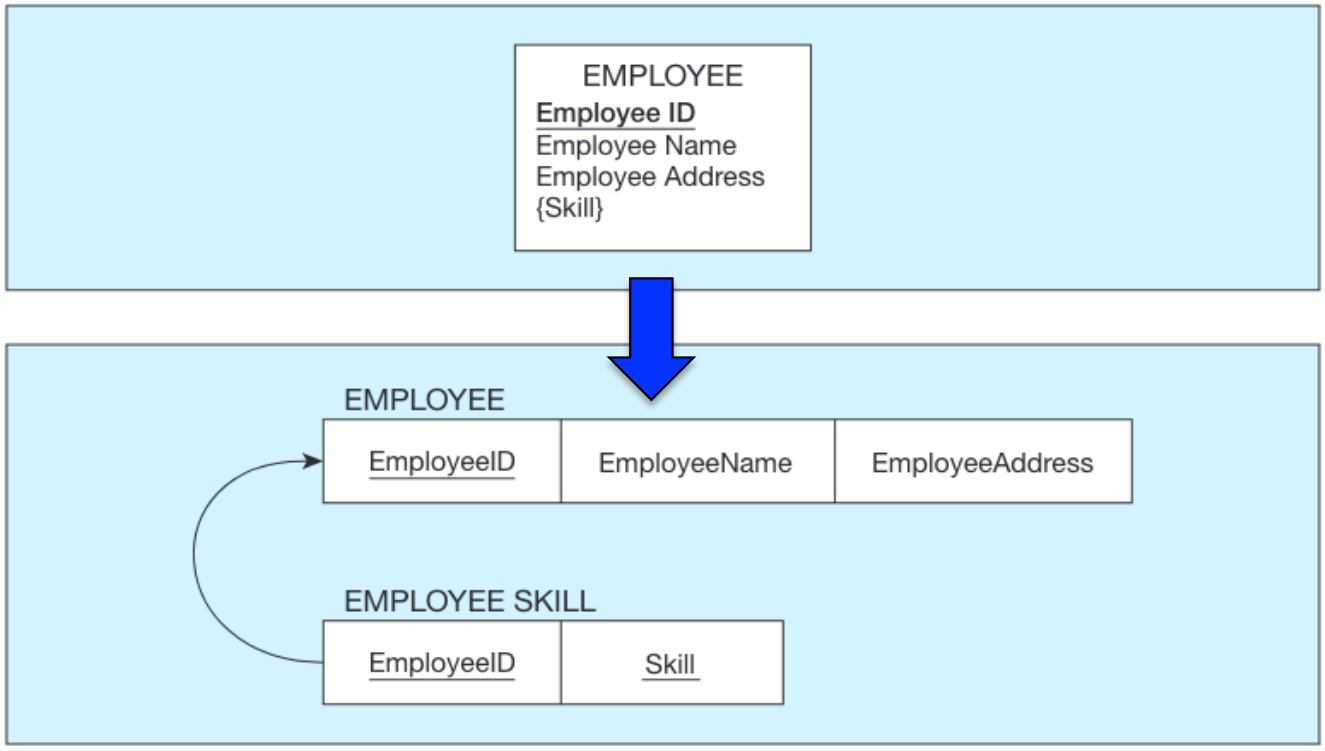
(Coronel and Morris, 2015)

Mapping Rules

R3: Multivalued attribute mapping

A multivalued attribute (an implicit entity type) will become a separate relation with a foreign key taken from the superior entity.

- The name of the new relation should capture the meaning of the multivalued attribute.
- Primary key of the relation: a composite key of the multivalued attribute and the primary key of the superior entity (frequently).



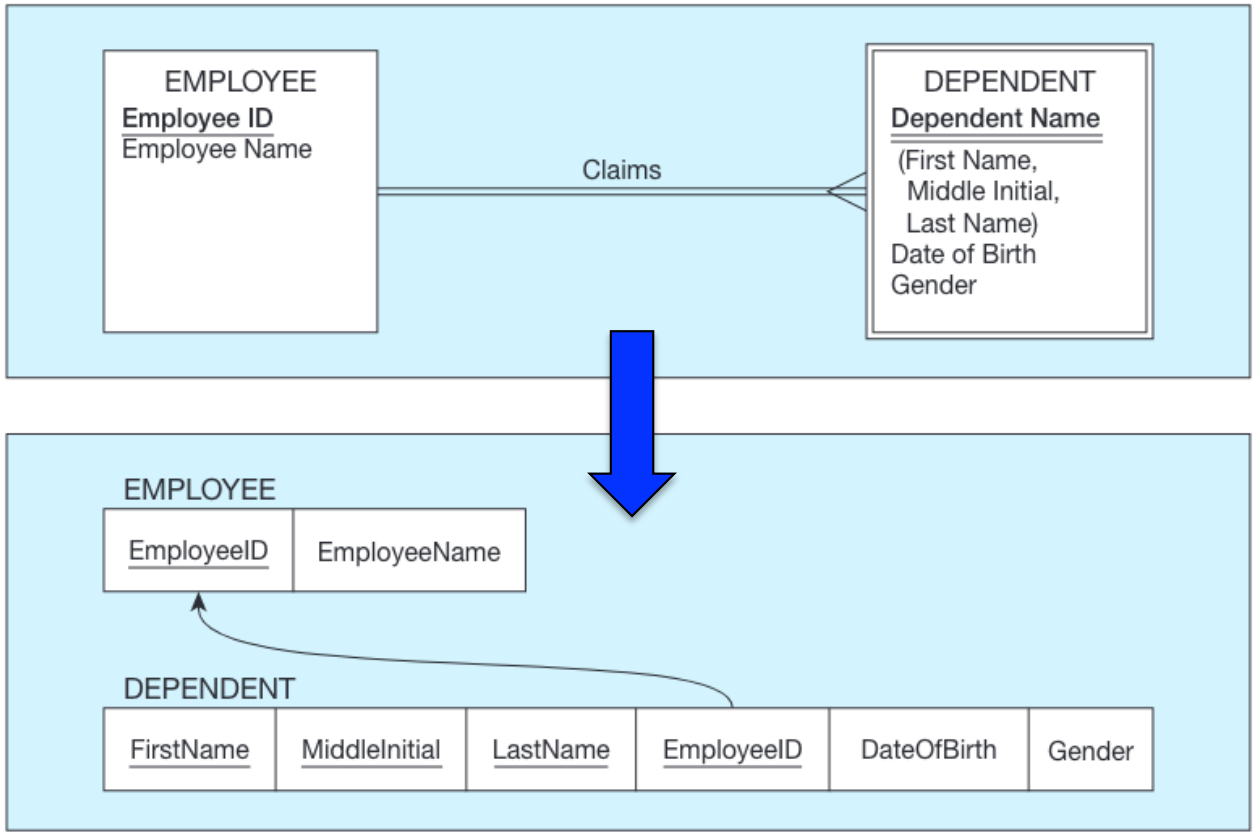
(Coronel and Morris, 2015)

Mapping Rules

R4: Weak entity mapping

Each weak entity type is transformed into a relation with the primary key is composed of the partial identifier of the weak entity and the identifier of its owner.

- Other non-key attributes ⇨ Follow R1, R2, and R3.

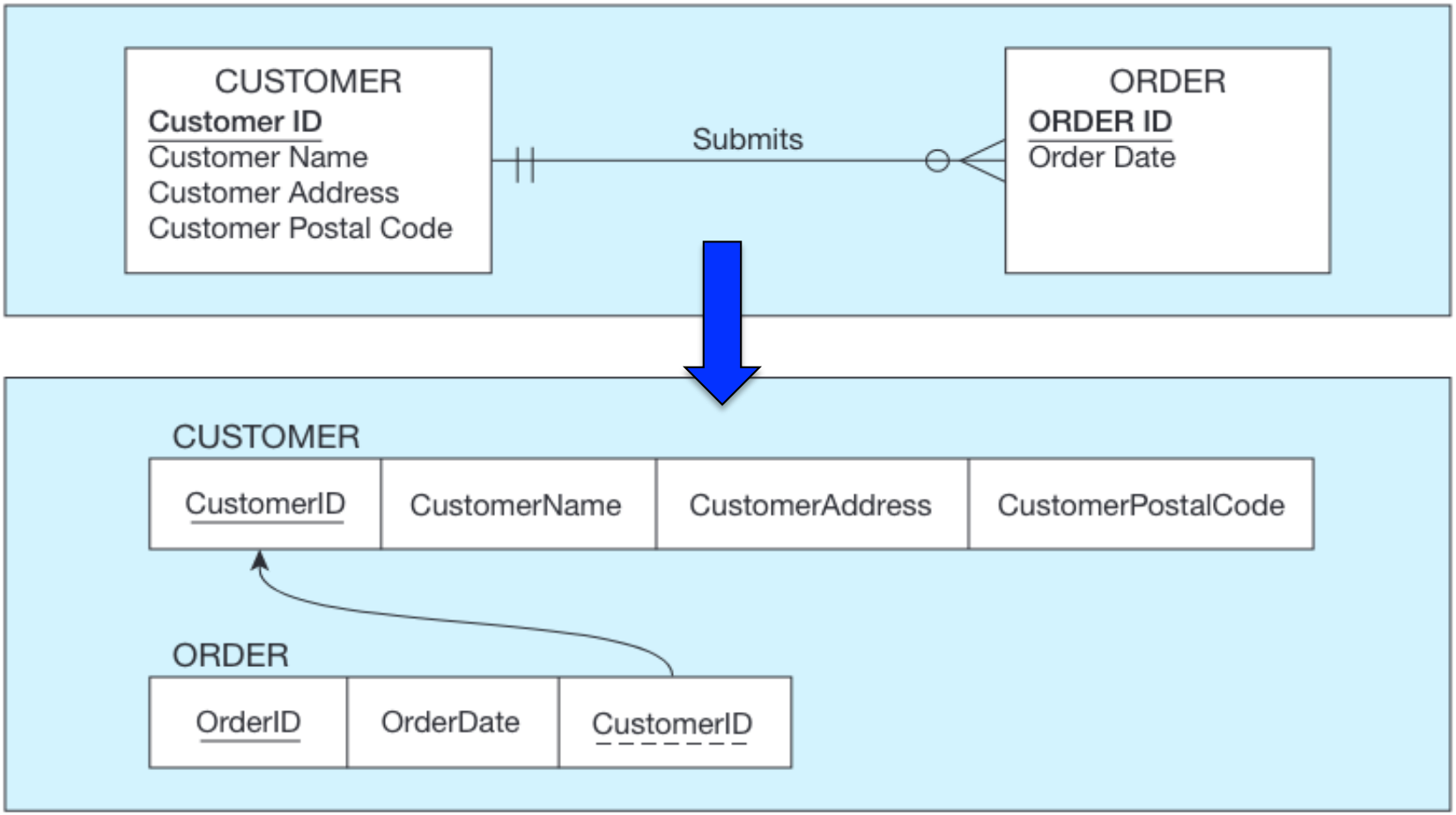


(Coronel and Morris, 2015)

Mapping Rules

R5: 1:N binary/unary relationship mapping

Include the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.

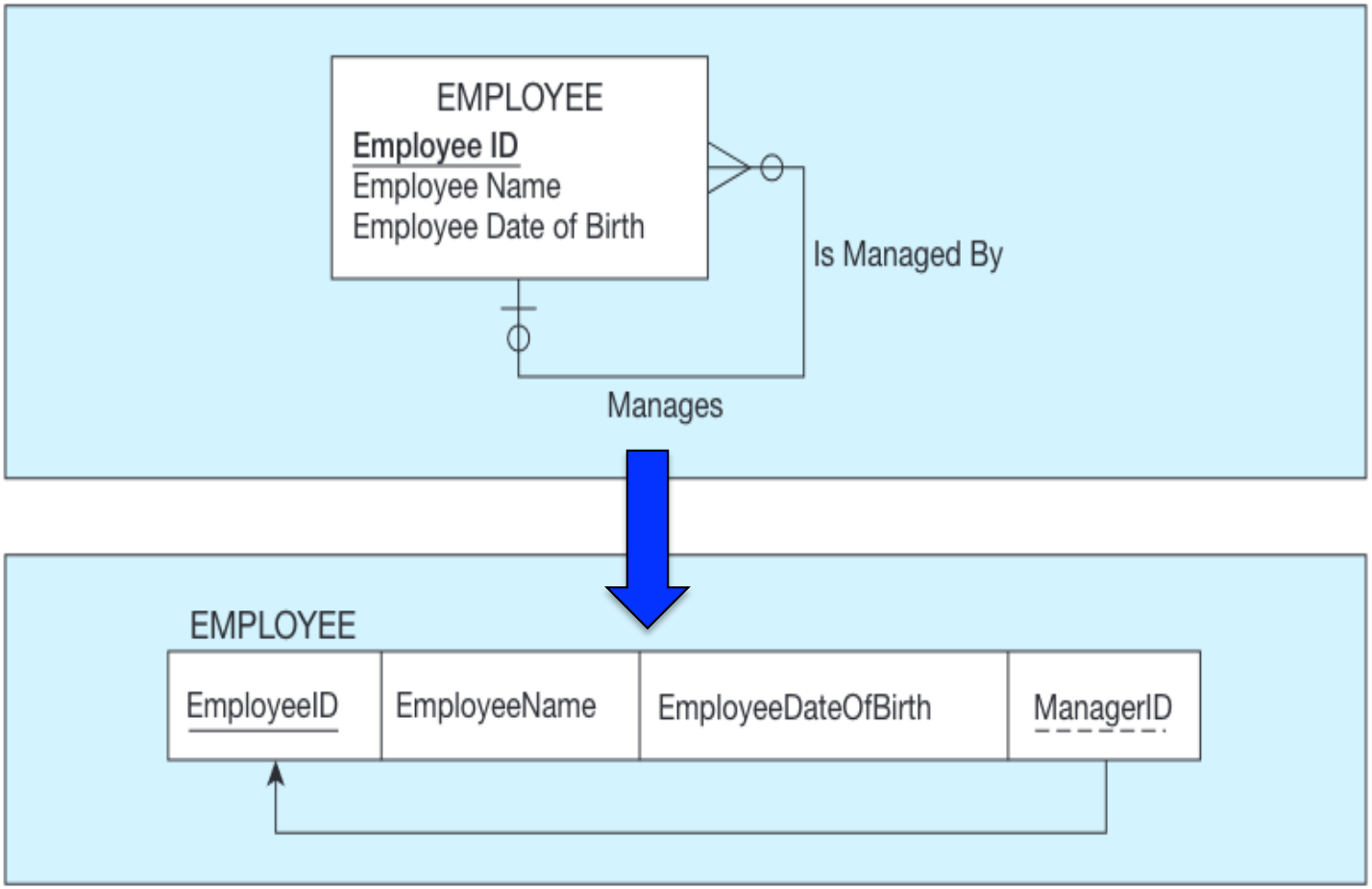


(Coronel and Morris, 2015)

Mapping Rules

R5: 1:N binary/unary relationship mapping

Include the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.



(Coronel and Morris, 2015)

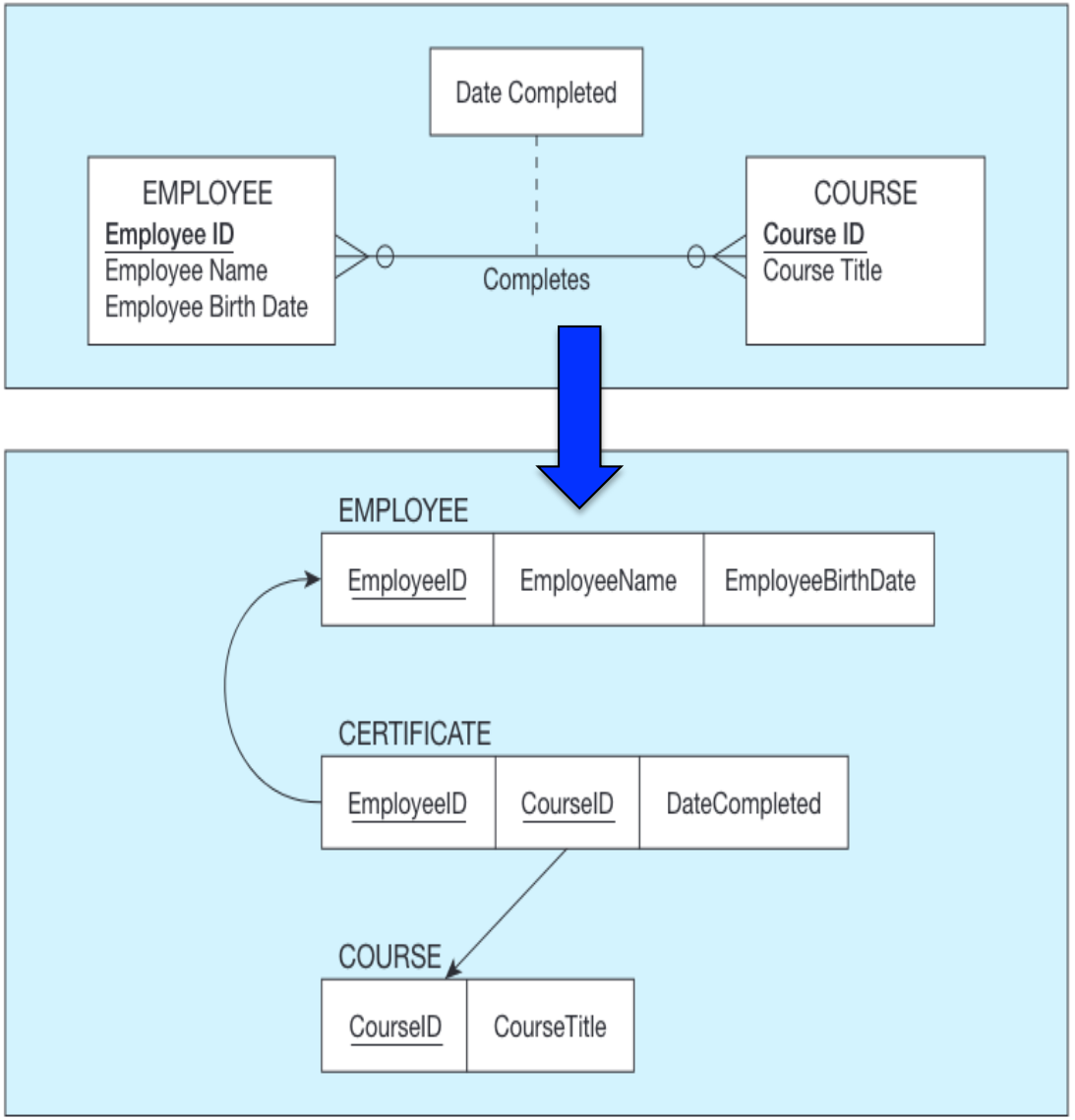
fit@hcmus

Mapping Rules

R6: N:N binary/unary relationship mapping

Create a relation with a composite primary key using the primary keys of the related entities.

- Other non-key attributes of the relationship can be integrated in the composite primary key if necessary or become atomic attributes.



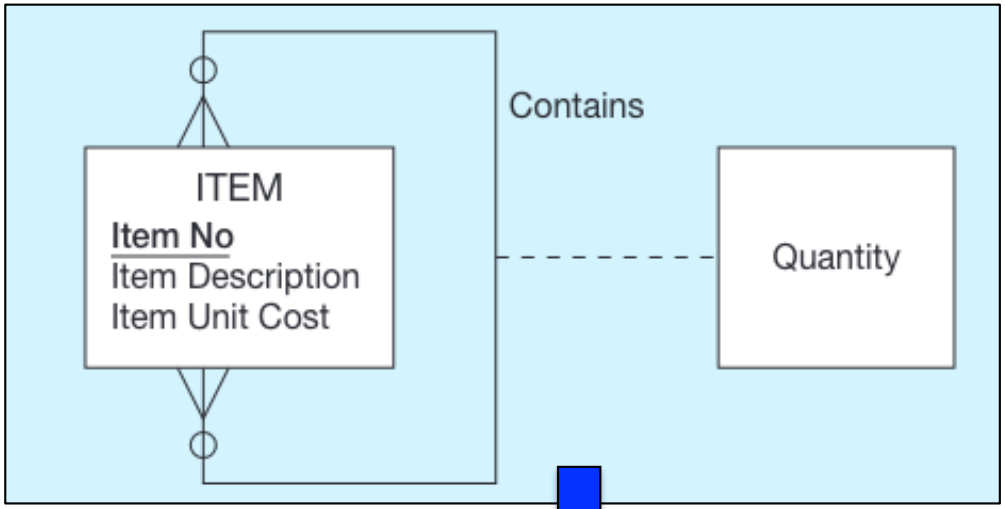
(Coronel and Morris, 2015)

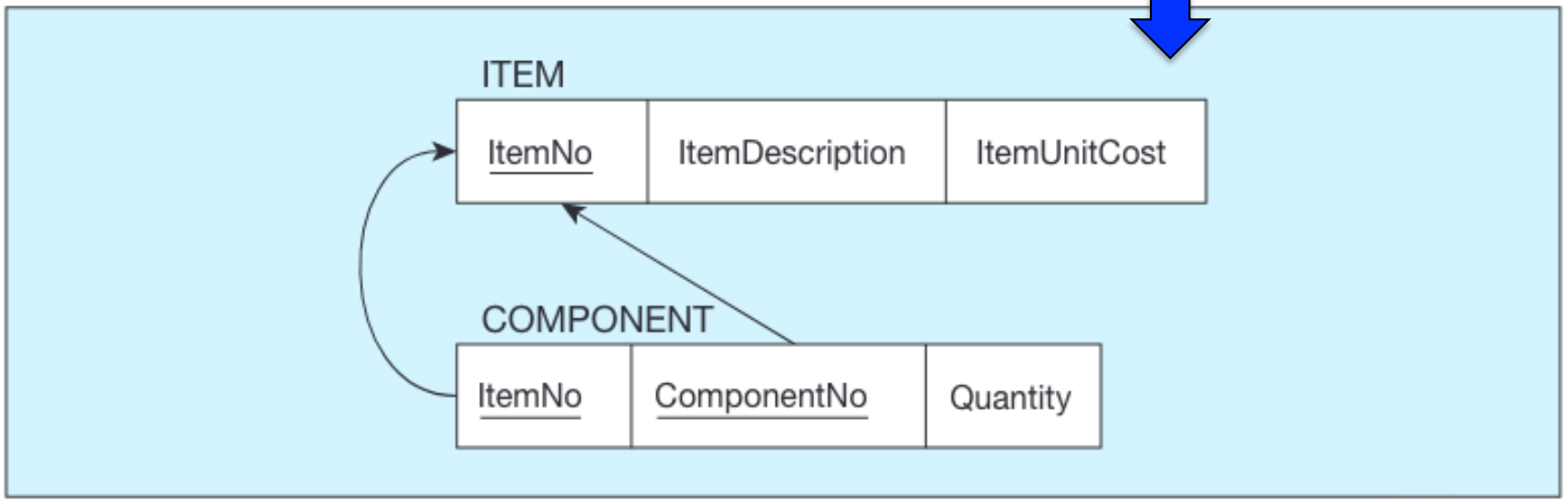
Mapping Rules

R6: N:N binary/unary relationship mapping

Create a relation with a composite primary key using the primary keys of the related entities.

- Other non-key attributes of the relationship can be integrated in the composite primary key if necessary or become atomic attributes.





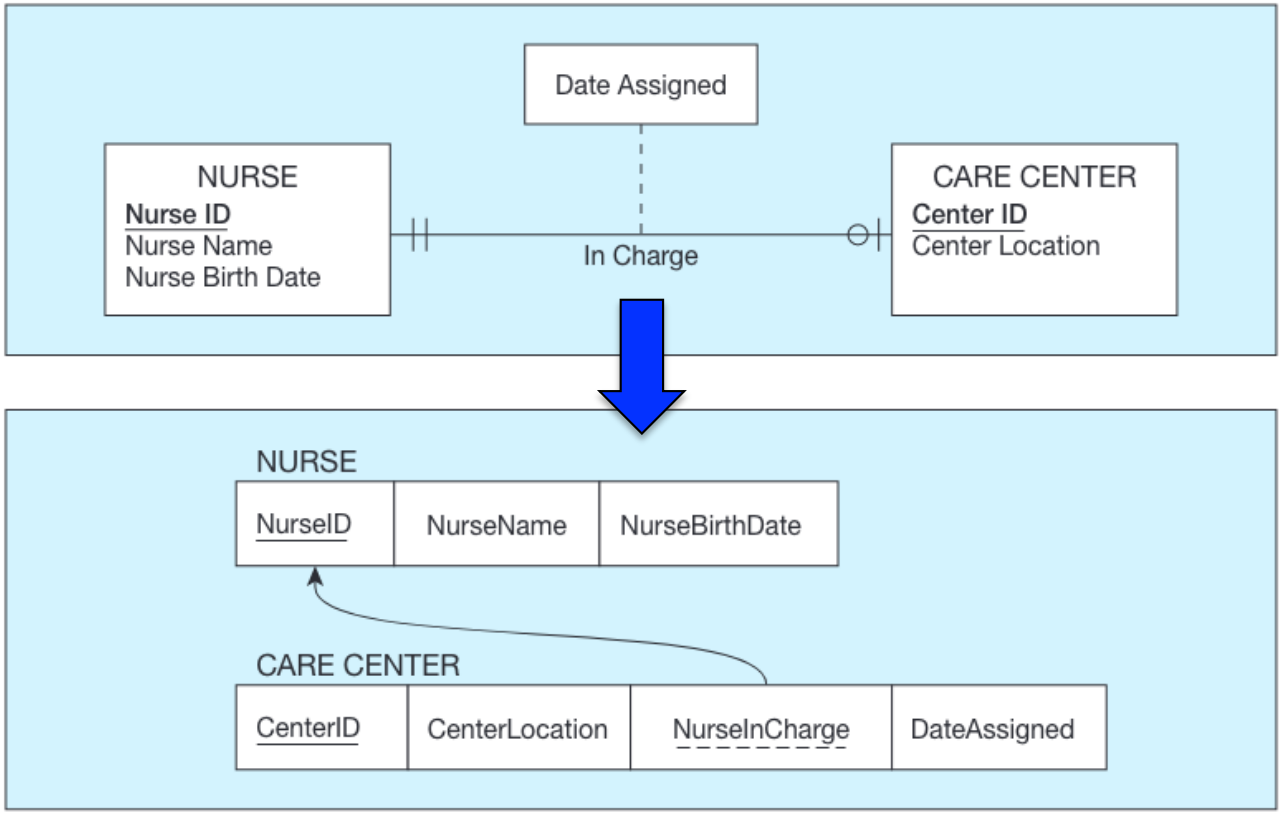
(Coronel and Morris, 2015)

Mapping Rules

R7: 1:1 binary/unary relationship mapping

The primary key of the entity on the mandatory side is included as a foreign key in the other relation if the relationship is optional-mandatory (in most cases). Additional specific consideration:

- Optional-Optional ⇨ Consider restructuring the schema.
- Mandatory-Mandatory ⇨ Consider combining the two related entities.

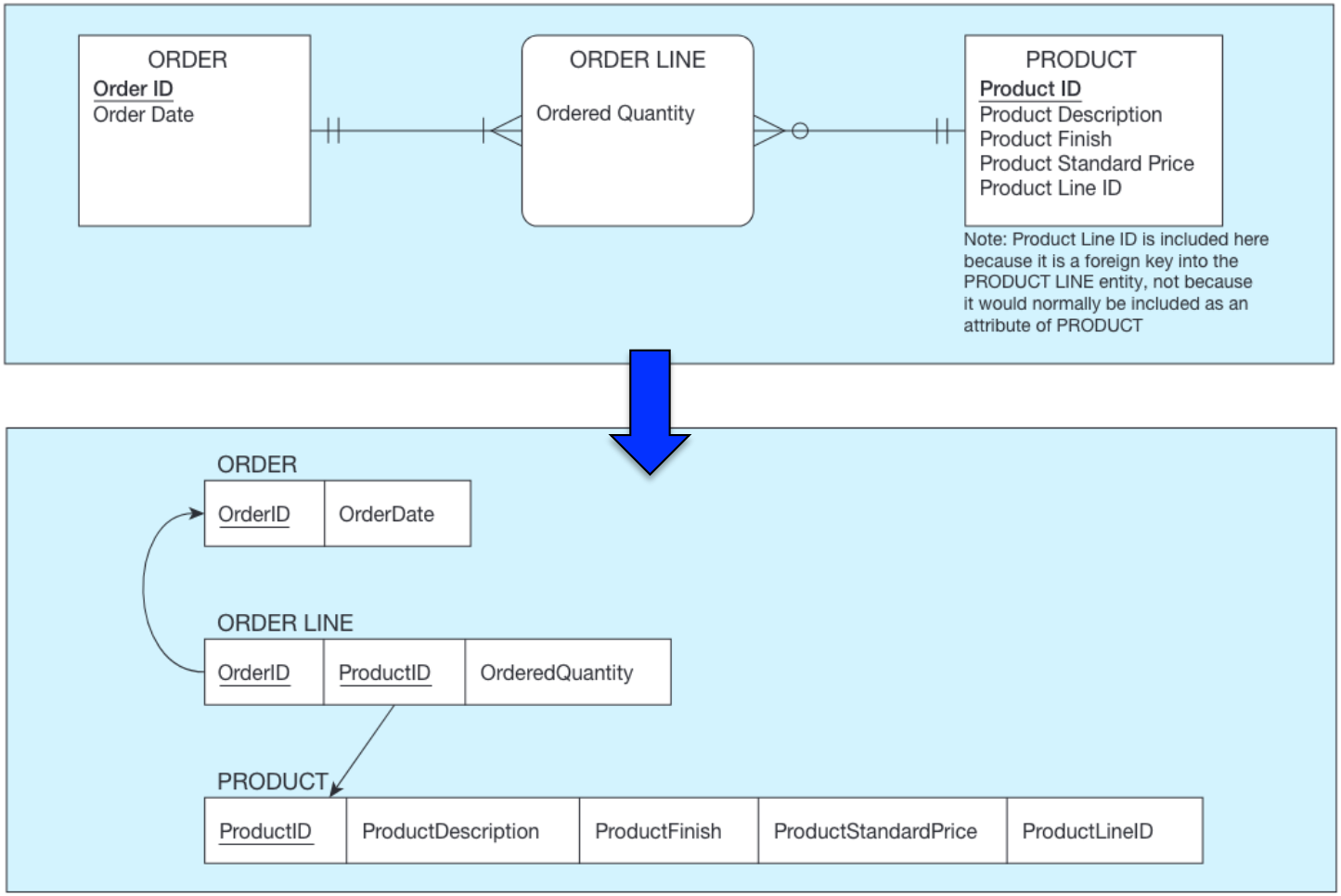


(Coronel and Morris, 2015)

Mapping Rules

R8: Associative entity (or ternary) mapping

For an associative entity without identifiers or a ternary relationship: Create a new relation using R6 as N-N binary relationships.



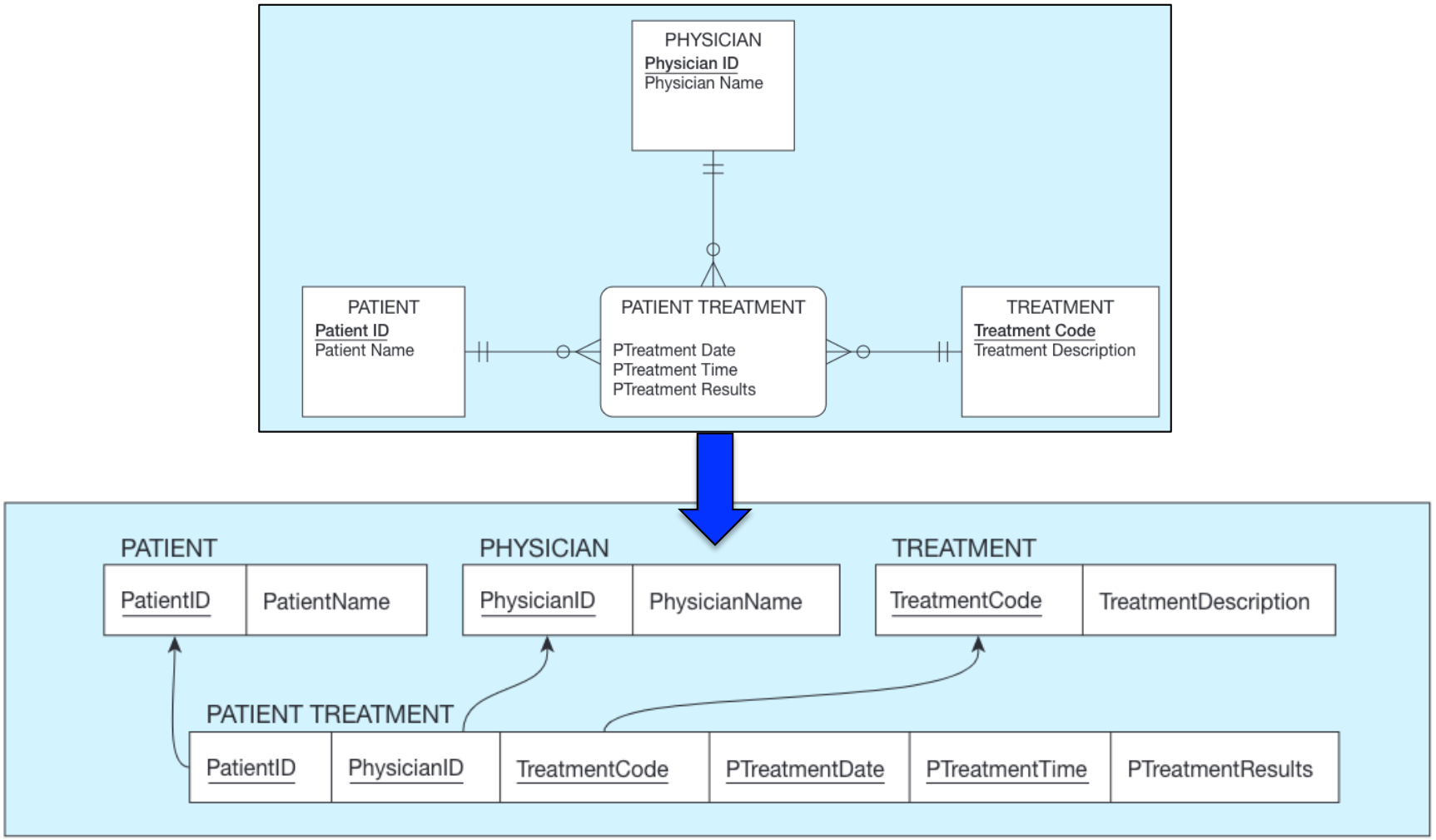
(Coronel and Morris, 2015)

fit@hcmus

Mapping Rules

R8: Associative entity (or ternary) mapping

For an associative entity without identifiers or a ternary relationship: Create a new relation using R6 as N-N binary relationships.

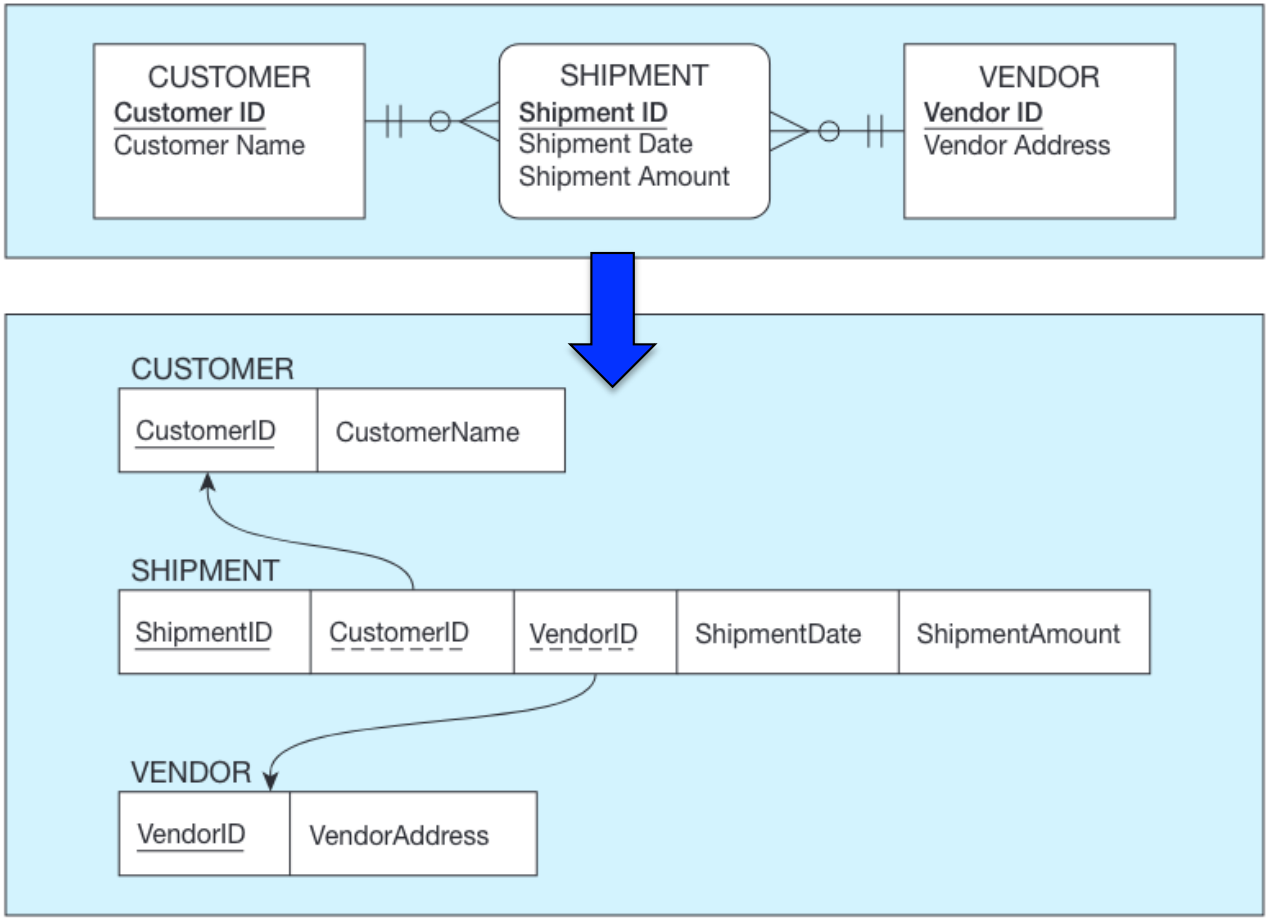


(Coronel and Morris, 2015)

Mapping Rules

R8: Associative entity (or ternary) mapping

For an associative entity owning its identifier: Create a new relation where the primary key is the identifier, and the identifiers of related entities become the foreign keys of this new relation.



(Coronel and Morris, 2015)

Mapping Rules

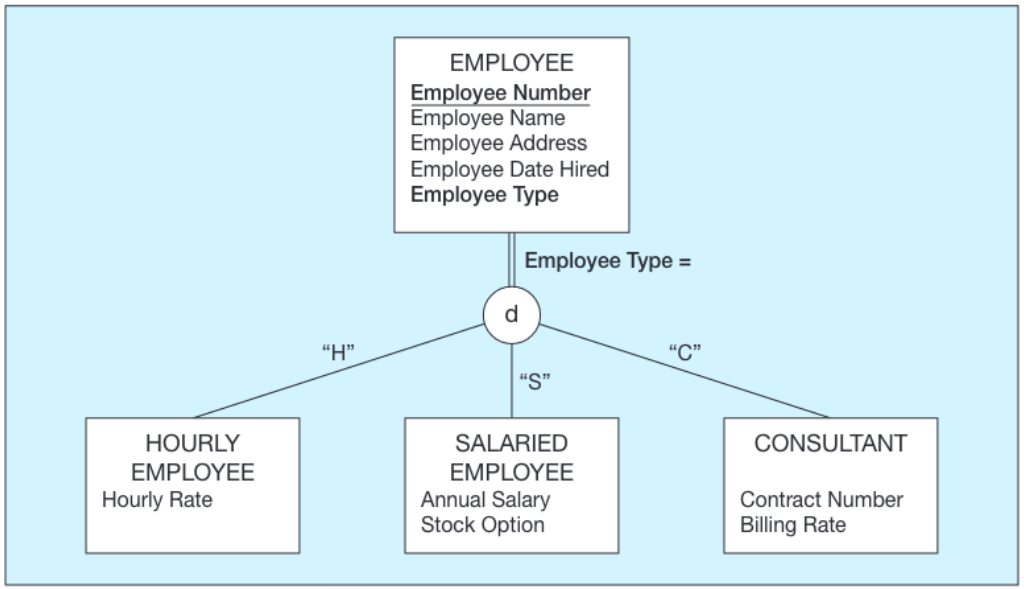
R9: Supertype/Subtype relationship mapping.

- **Option A: Create relations for subtypes and supertypes** (**suitable** for **any** specialization).
- **Option B: Create relations for subtypes only** (**only** for **total** specialization).
- **Option C: Create a single relation with a single "type" discriminator** (**only** for **disjoint** specialization).
- **Option C: Create a single relation with multiple boolean discriminators** (**suitable** for **overlap** specialization).

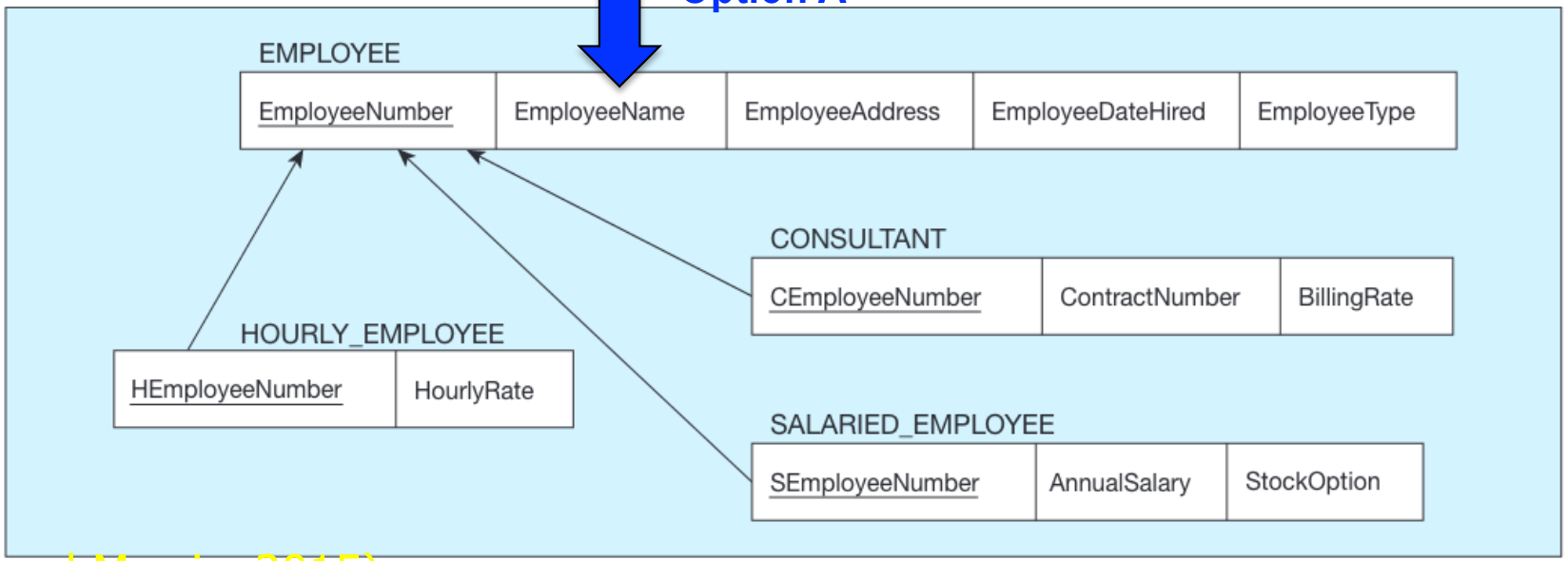
(Coronel and Morris, 2015)

Mapping Rules

R9: Supertype/Subtype relationship mapping.

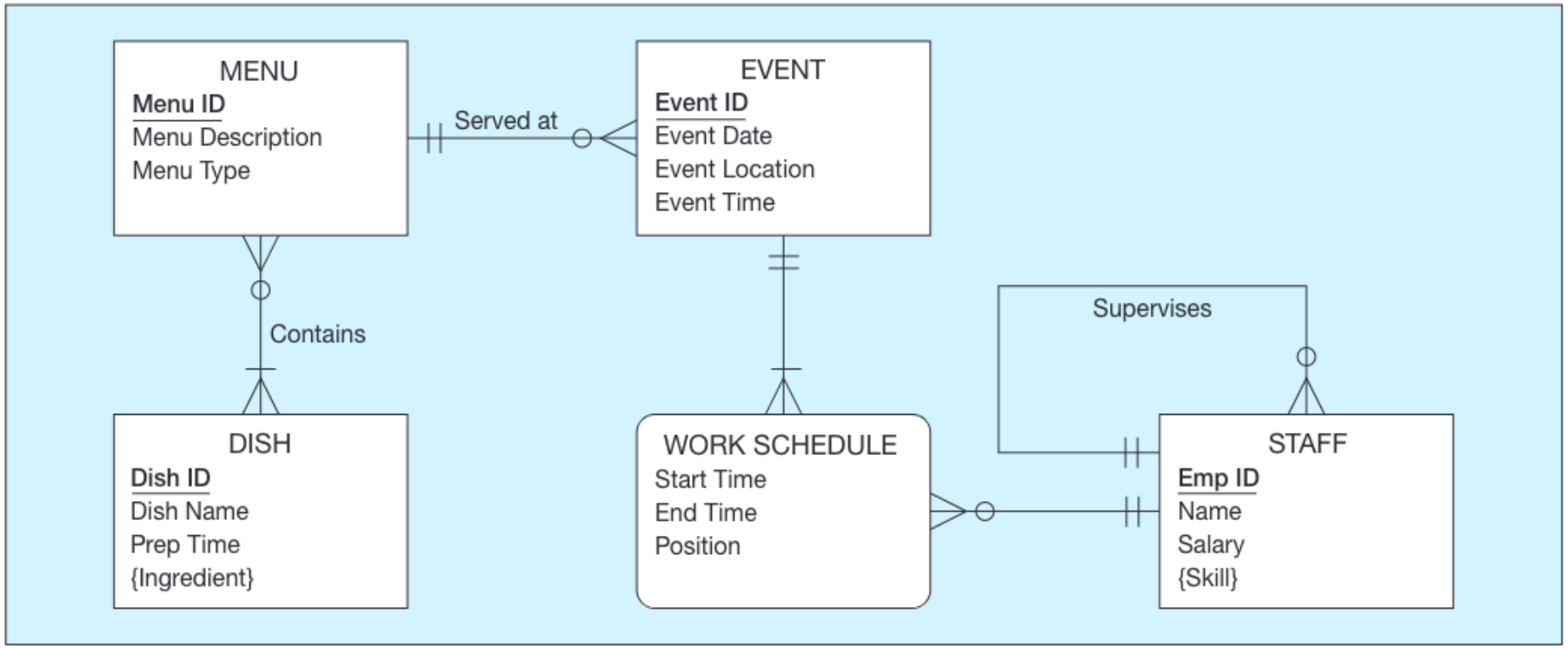


Option A



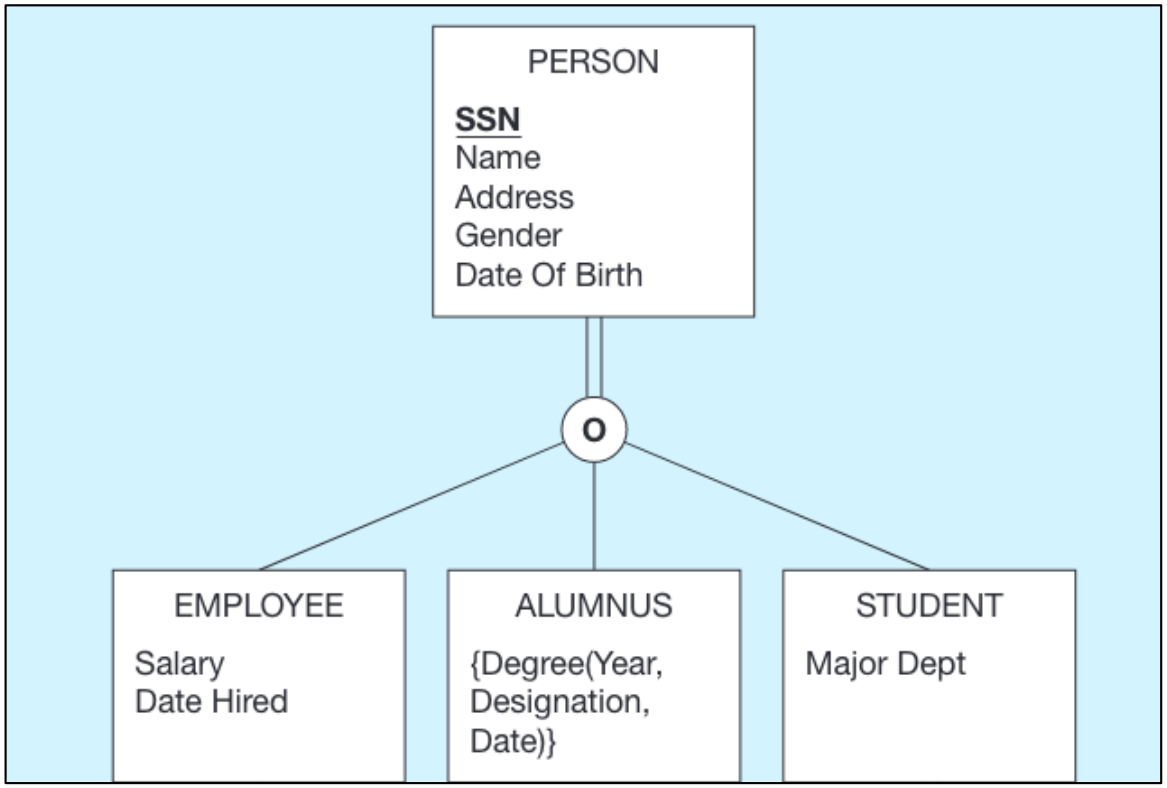
(Coronel and Morris, 2015)

Practice 1.1: **ERD → Relations**



(Coronel and Morris, 2015)

Practice 1.2: **ERD → Relations**

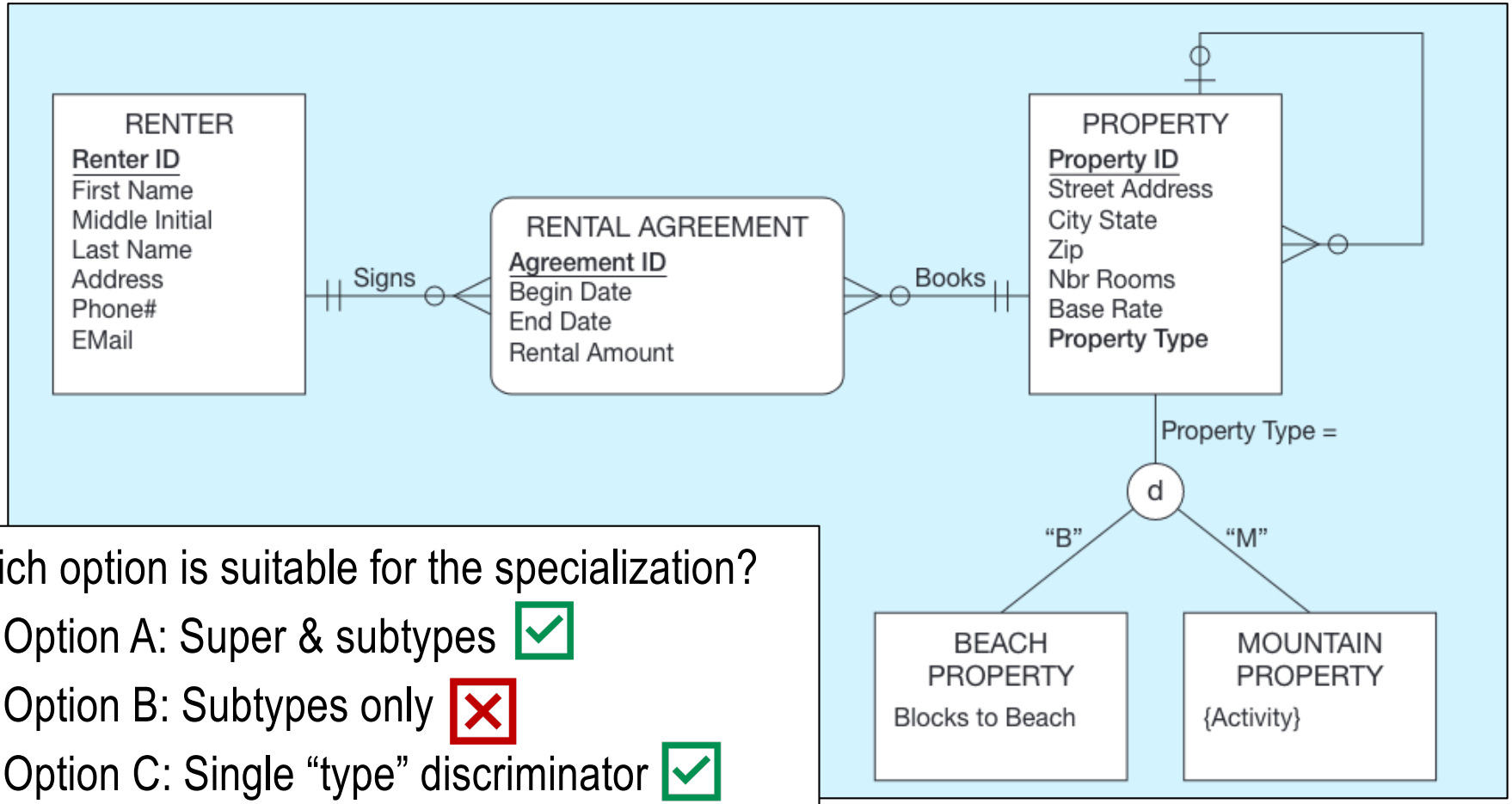


Which option is suitable?

- Option A: Create relations for subtypes & supertypes ✓
- Option B: Create relations for subtypes only ✓
- Option C: Create a single relation with a single "type" discriminator ✗
- Option C: Create a single relation with multiple boolean discriminators ✓

(Coronel and Morris, 2015)

Practice 1.3: **ERD → Relations**



Which option is suitable for the specialization?

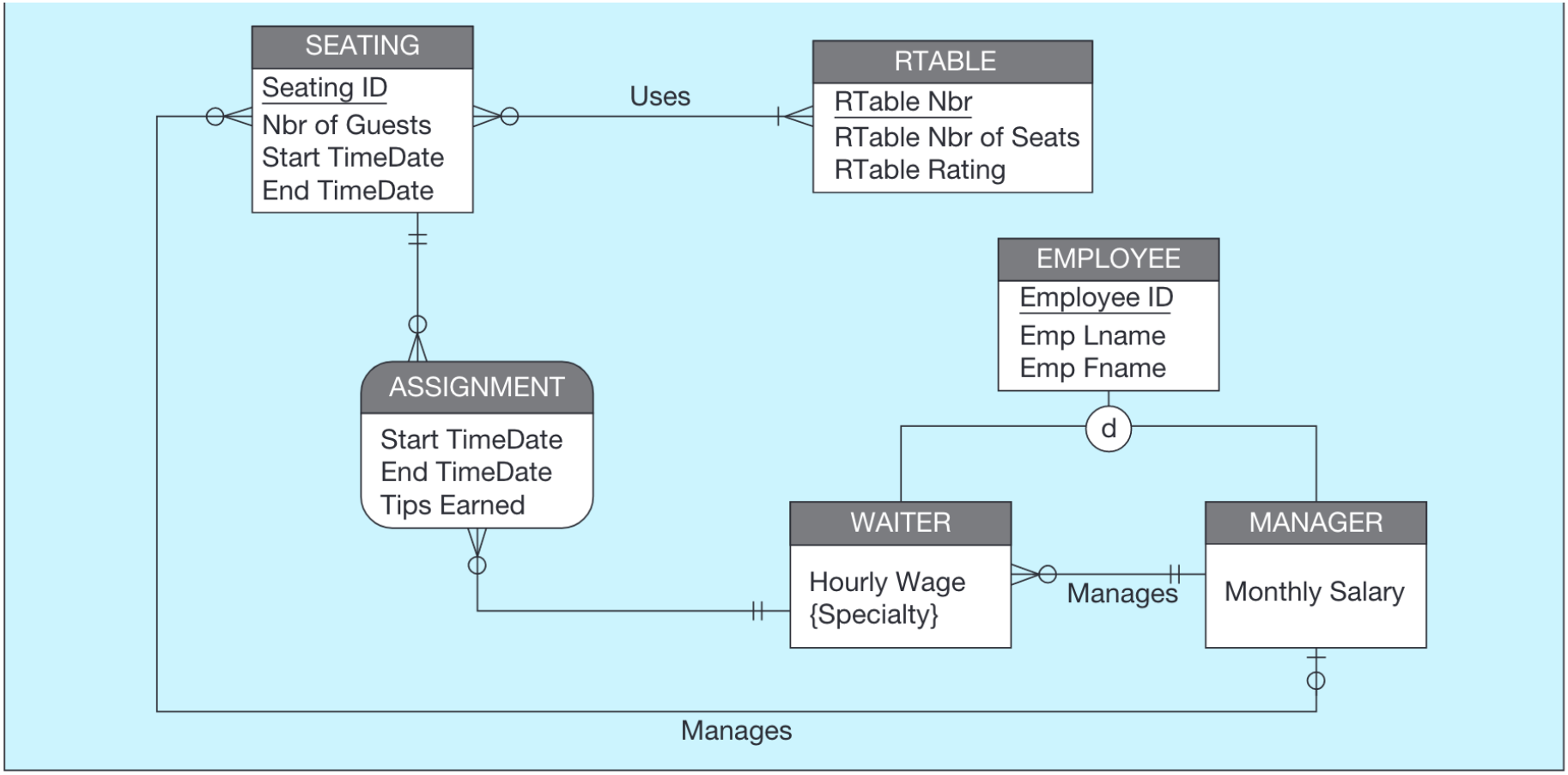
- Option A: Super & subtypes ✓
- Option B: Subtypes only ✗
- Option C: Single "type" discriminator ✓
- Option C: Multiple boolean discriminators ?

(Coronel and Morris, 2015)

Practice 2:

Given a ERD for a restaurant as follows.

a. Identify the functional dependencies from the ERD.
b. Draw a relational schema in 1NF from the proposed ERD.
c. Draw a relational schema in 2NF from the proposed ERD.
d. Draw a relational schema in 3NF from the proposed ERD.



(Coronel and Morris, 2015)

Practice 3:

Given a shipping manifest data shown in the following form.

a. Draw a relational schema and diagram the functional dependencies in the relation in 1NF.

b. In what normal form is this relation?

c. Decompose MANIFEST into a set of 3NF relations.

d. Draw a relational schema for your 3NF relations and show the referential integrity constraints.

Shipping manifest form

Shipment ID:	00-0001	Shipment Date:	01/10/2018
Origin:	Boston	Expected Arrival:	01/14/2018
Destination:	Brazil		
Ship Number:	39	Captain:	002-15 Henry Moore

Item Number	Type	Description	Weight	Quantity	TOTALWEIGHT
3223	BM	Concrete Form	500	100	50,000
3297	BM	Steel Beam	87	2,000	174,000
				Shipment Total:	224,000

(Coronel and Morris, 2015)

References

(Coronel and Morris, 2015)

Database System: Design, Implementation, and Management, 12th Edition, Carlos Coronel & Stevem, Morris, 2015.

(Hoffer et al., 2019)

Modern Database Management, 13th Edition, Jeffrey A. Hoffer, V. Ramesh, Heikki Topi, 2019.

(Silberschatz et al., 2020)

Database System Concepts, 17th Edition, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2020.

THE END