

UNIVERSITY OF INFORMATION TECHNOLOGY

Faculty of Information Systems

Chapter 3

Data and Process Modeling – SDLC Method

Dr. Cao Thi Nhan

LEARNING OBJECTIVES

1. Understand basic concepts of Data Flow Diagram (DFD).
2. Have ability to create logical DFDs to analyse the current system through parent to child levels
3. Understand logical and physical DFDs
4. Understand basic concepts of Entity – Relationship Data model (ERD), Relational Data model (RD).
5. Have ability to create ERD, then convert to RD for real life situations

CONTENT

1. Process Analysis and Design

- a. Data Flow Diagram Symbols
- b. Data Flow Diagram levels
- c. Creating DFDs
- d. Physical and logical DFDs

2. Data Analysis and Design

- a. Entity – Relationship Data Model
- b. Steps to convert ERD to Relational model

Process Analysis and Design

Process Analysis and Design

1. Data Flow Diagram Symbols
2. Data Flow Diagram levels
3. Creating DFDs
4. Physical and logical DFDs

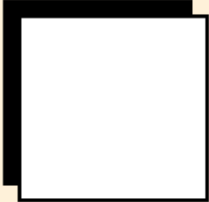
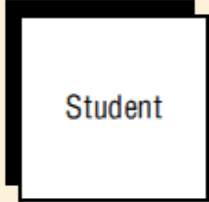
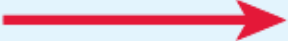
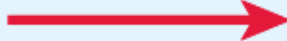
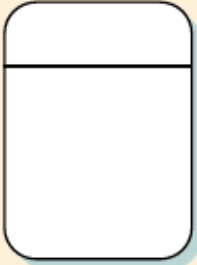
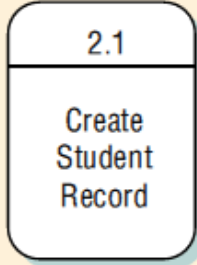

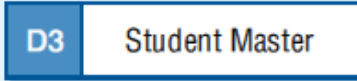
Data Flow Diagrams

- Graphically characterize data processes and flows in a business system
- Depict:
 - System inputs
 - Processes
 - Outputs

Basic Symbols

1. A (double) square: an external entity
2. An arrow: a movement of data from one point to another
3. A rectangle with rounded corners: a transforming process
4. An open-ended rectangle: a data store

The Four Basic Symbols Used in Data Flow Diagrams, Their Meanings, and Examples (Figure 7.1)

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

External Entities

- A **source** or **destination** of data, outside the boundaries of the system
- Represent another department, a business, a person, or a machine
- Should be named with a noun

Data Flow

- Shows movement of data from one point to another
- Described with a noun
- Arrowhead indicates the flow direction
- Represents data about a person, place, or thing

Process

- Denotes a change in or transformation of data
- Represents work being performed in the system
- Naming convention:
 - Assign the name of the whole system when naming a high-level process
 - To name a major subsystem attach the word subsystem to the name
 - Use the form verb-adjective-noun for detailed processes

Process

- Naming convention example:
 - To name of the whole system (a high-level process):
 - ◆ INVENTORY CONTROL SYSTEM
 - To name a major subsystem:
 - ◆ INVENTORY REPORTING SUBSYSTEM
 - ◆ INTERNET CUSTOMER FULFILLMENT SYSTEM
 - Use the form verb-adjective-noun for detailed processes
 - ◆ COMPUTE SALES TAX,
 - ◆ VERIFY CUSTOMER ACCOUNT STATUS,
 - ◆ PREPARE SHIPPING INVOICE,
 - ◆ SEND CUSTOMER EMAIL CONFIRMATION,
 - ◆ ADD INVENTORYRE RECORD,
 - ◆ COMPUTE CUSTOMER'S DISCOUNT

Data Store

- A depository for data that allows examination, addition, and retrieval of data
- Named with a noun, describing the data
- Data stores are usually given a unique reference number, such as D1, D2, D3
- Represents a:
 - Database
 - Computerized file
 - Filing cabinet

Data Store

- Examples:
 - Customer
 - Product
 - Order
 - OrderList
 - Employee
 - ...

Steps in Developing Data Flow Diagrams

1. Make a **list of business activities** and use it to determine various: (1) External entities, (2) Data flows, (3) Processes, (4) Data stores
2. Create a **context diagram** that shows external entities and data flows to and from the system. **Do not show any detailed processes or data stores.**
3. Draw **Diagram 0**, the next level. Show processes, but keep them general. Show data stores at this level.
4. Create a **child diagram** for each of the processes in Diagram 0.
5. Check for errors and make sure the labels you assign to each process and data flow are meaningful.
6. Develop a physical data flow diagram from the logical data flow diagram. **Distinguish between manual and automated processes**, describe actual files and reports by name, and add controls to indicate when processes are complete or errors occur.
7. Partition the physical data flow diagram by separating or grouping parts of the diagram in order to facilitate programming and implementation.

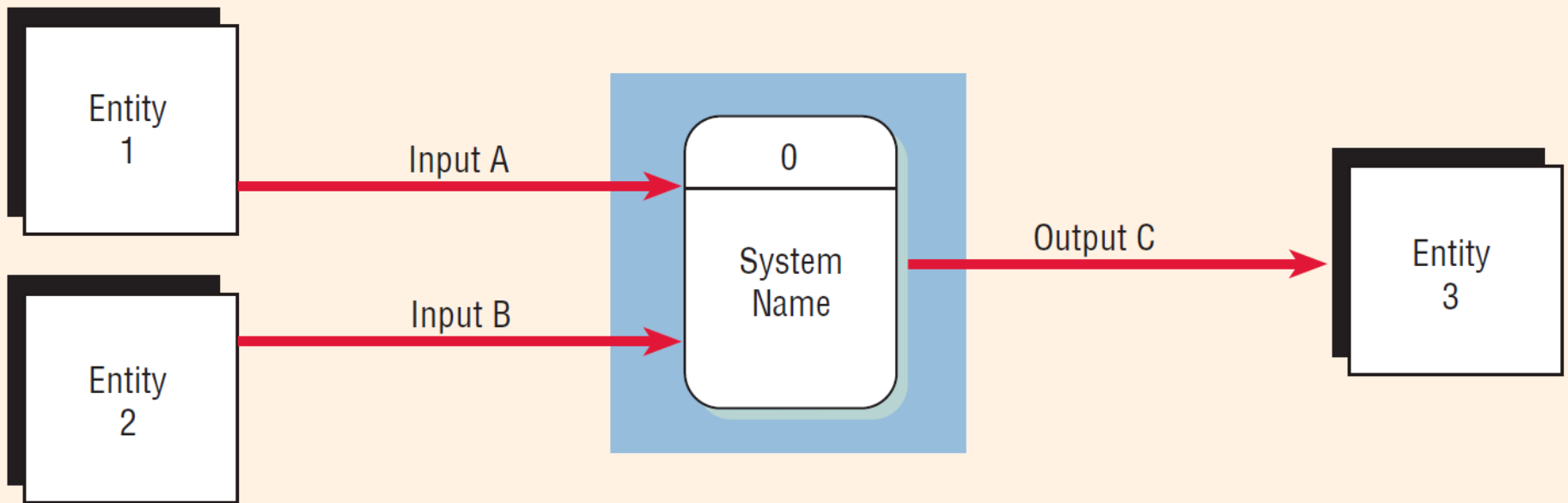
Creating the Context Diagram

- The highest level in a data flow diagram
- Contains only one process, representing the entire system
- The process is given the number 0
- All external entities, as well as major data flows are shown
- The diagram does not contain any data stores.

Basic Rules

- The data flow diagram must have one process
- Must not be any freestanding objects
- A process must have both an input and output data flow
- A data store must be connected to at least one process
- External entities should not be connected to one another

Context Diagram (Figure 7.3)



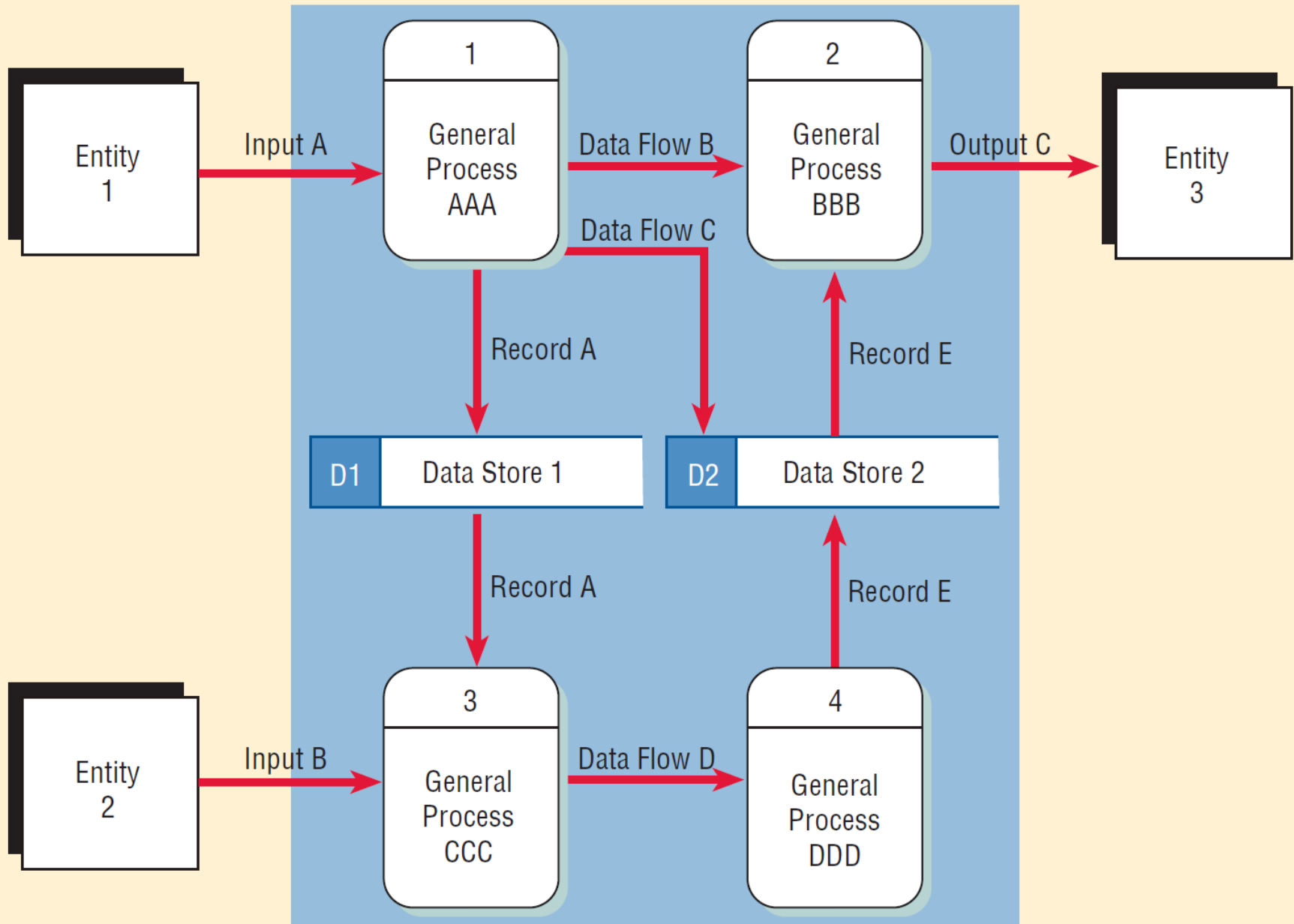
Drawing Diagram 0

- The explosion of the context diagram
- May include up to **nine** processes
- Each process is numbered
- Major data stores and all external entities are included

Drawing Diagram 0 (continued)

- Start with the data flow from an entity on the input side
- Work backward from an output data flow
- Examine the data flow to or from a data store
- Analyze a well-defined process
- Take note of any fuzzy areas

Note Greater Detail in Diagram 0 (Figure 7.3)



Data Flow Diagram Levels

- Data flow diagrams are built in layers
- The top level is the context level
- Each process may explode to a lower level
- The lower level diagram number is the same as the parent process number

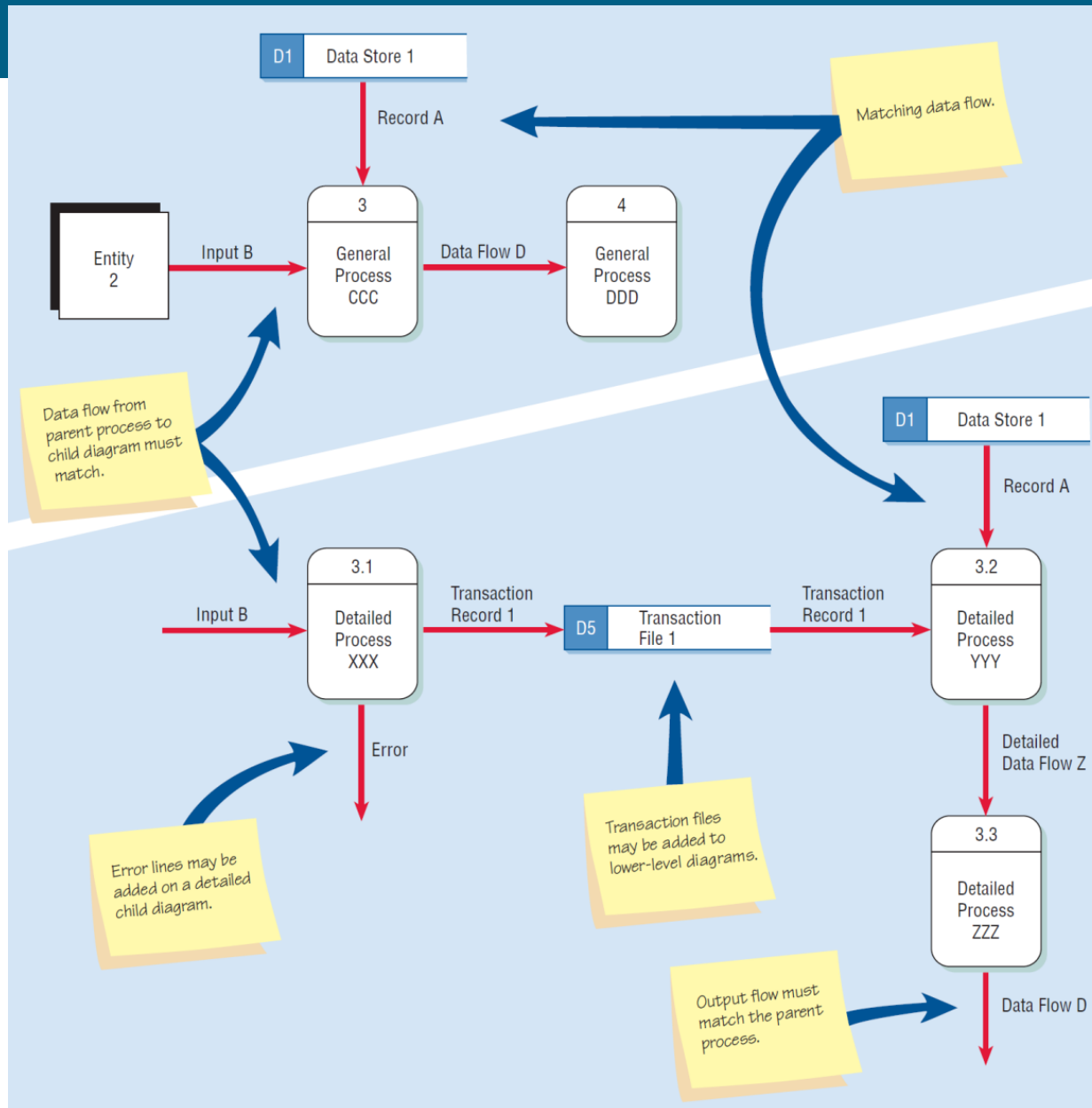
Creating Child Diagrams

- Each process on diagram 0 may be exploded to create a child diagram
- A child diagram cannot produce output or receive input that the parent process does not also produce or receive
- The child process is given the same number as the parent process
 - Process 3 would explode to Diagram 3

Creating Child Diagrams (continued)

- Entities are usually not shown on the child diagrams below Diagram 0
- If the parent process has data flow connecting to a data store, the child diagram may include the data store as well
- When a process is not exploded, it is called a primitive process

Differences between the Parent Diagram (above) and the Child Diagram (below) (Figure 7.4)



Data Flow Diagrams Error Summary

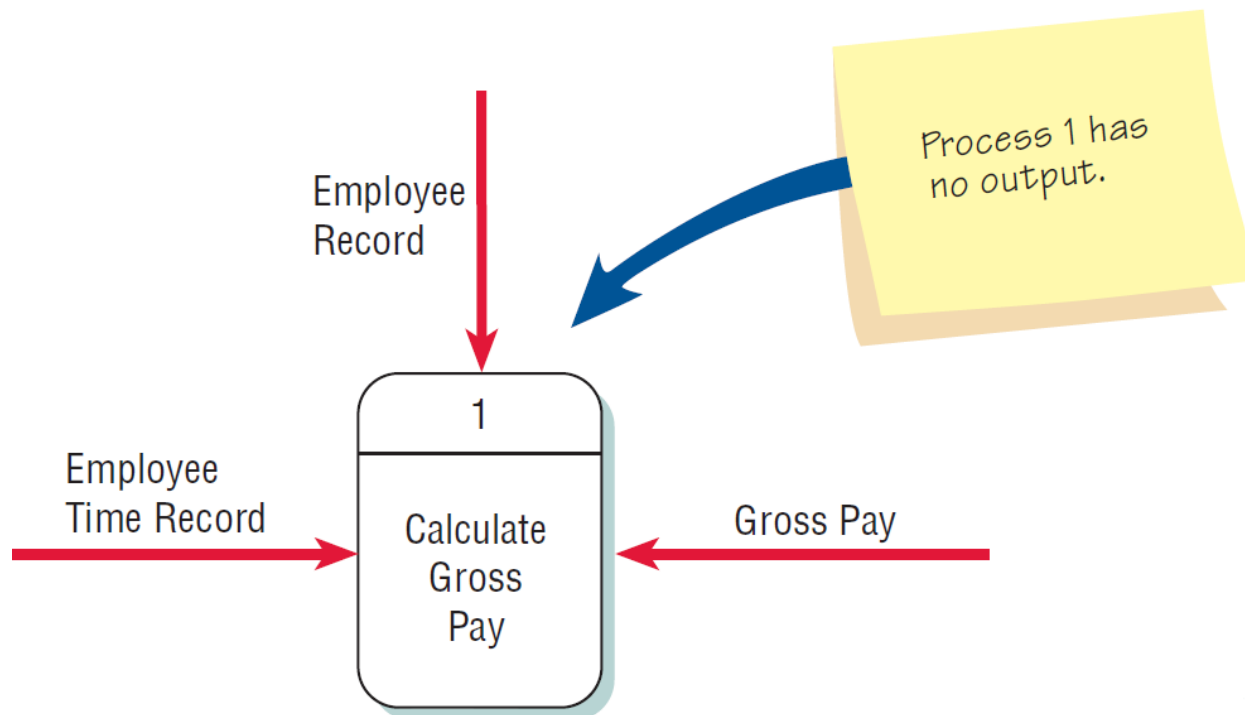
- Forgetting to include a data flow or pointing an arrow in the wrong direction
- Connecting data stores and external entities directly to each other
- Incorrectly labeling processes or data flow

Data Flow Diagrams Error Summary (continued)

- Including more than nine processes on a data flow diagram
- Omitting data flow
- Creating unbalanced decomposition (or explosion) in child diagrams

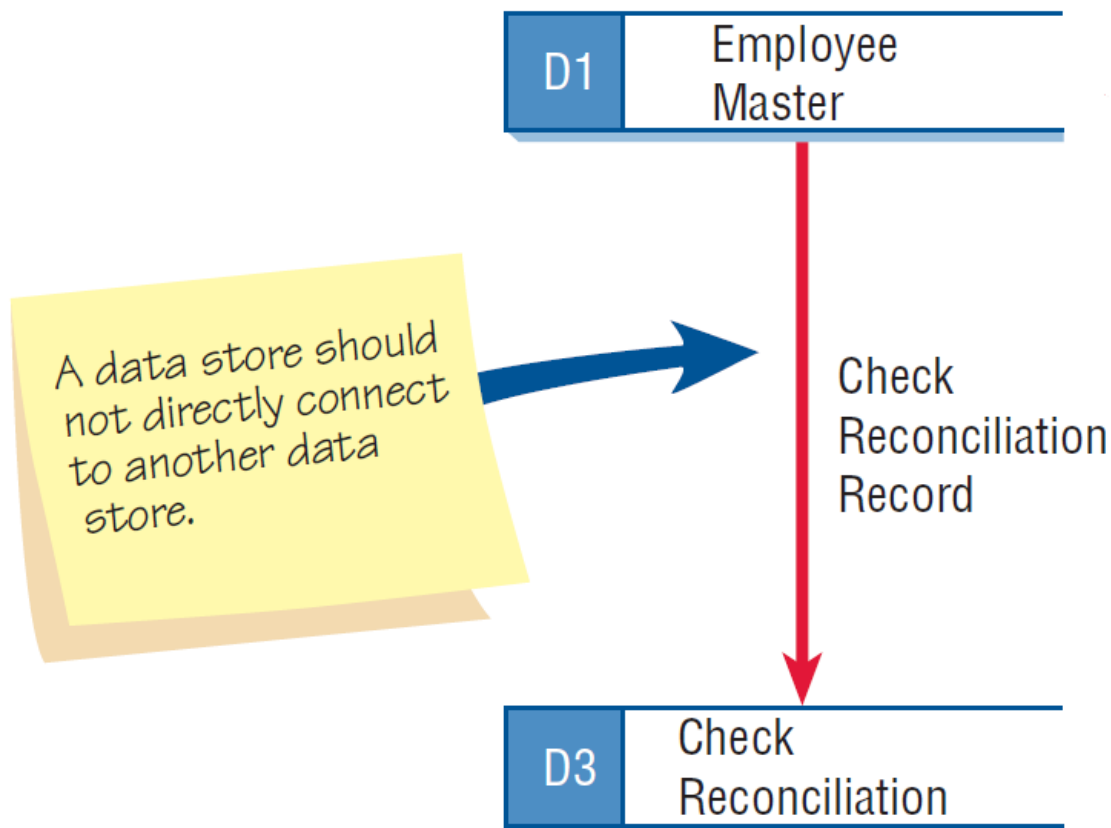
Checking the Diagrams for Errors (Figure 7.5)

- Forgetting to include a data flow or pointing an arrow in the wrong direction

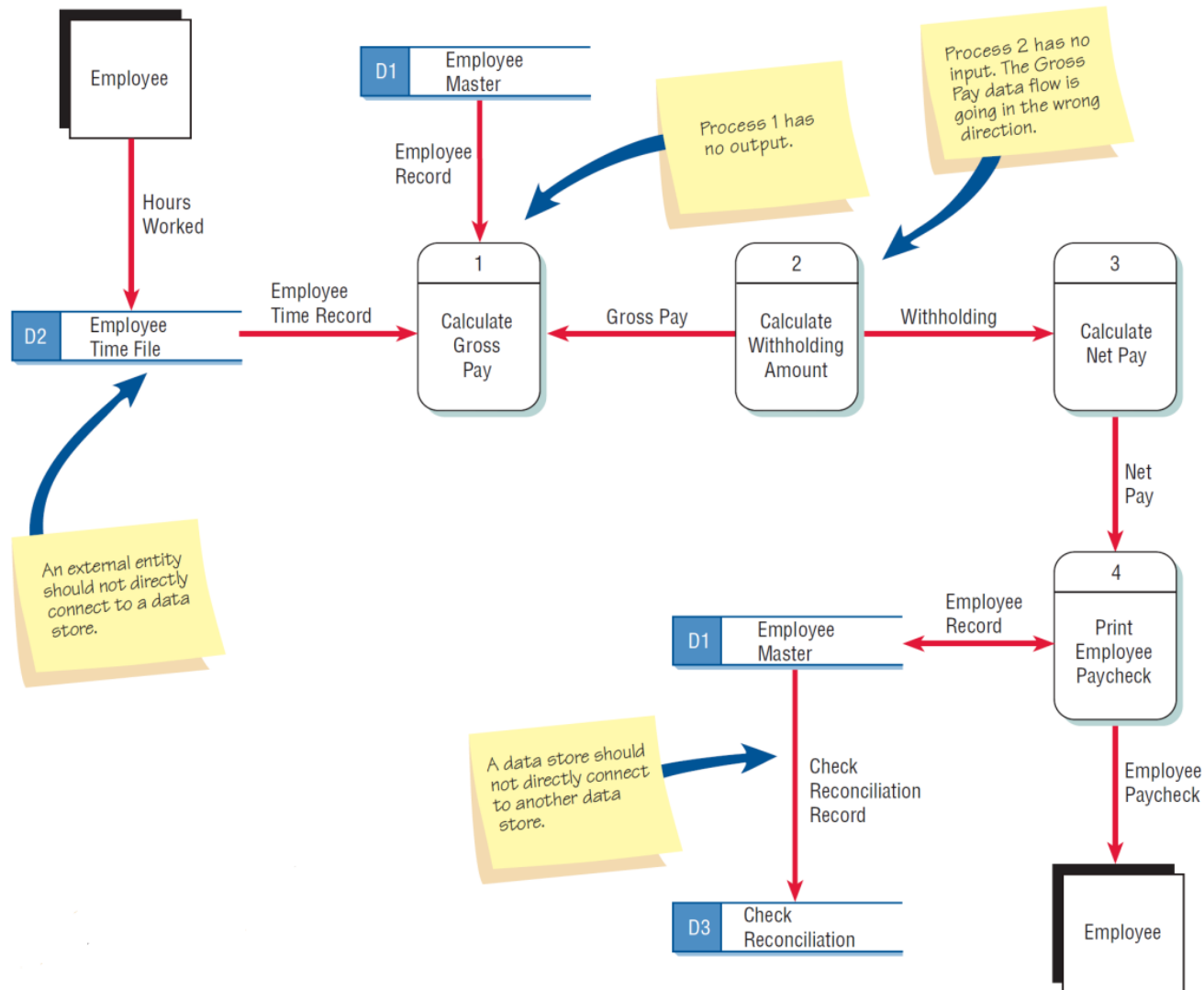


Checking the Diagrams for Errors (continued Figure 7.5)

- Connecting data stores and external entities directly to each other



Typical Errors that Can Occur in a Data Flow Diagram (Payroll Example) (continued Figure 7.5)



Context-level DFD

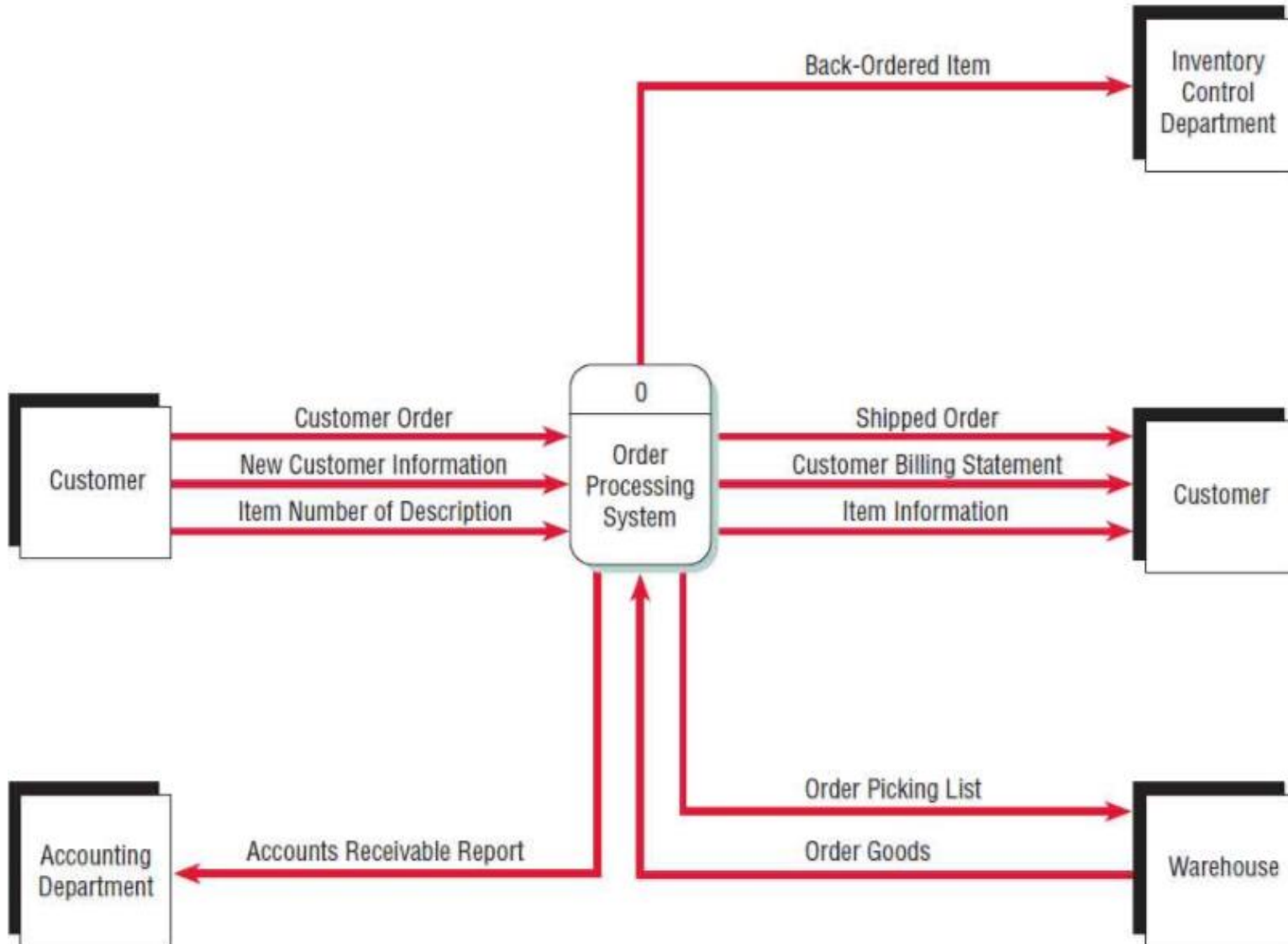


Diagram 0 of the order processing system

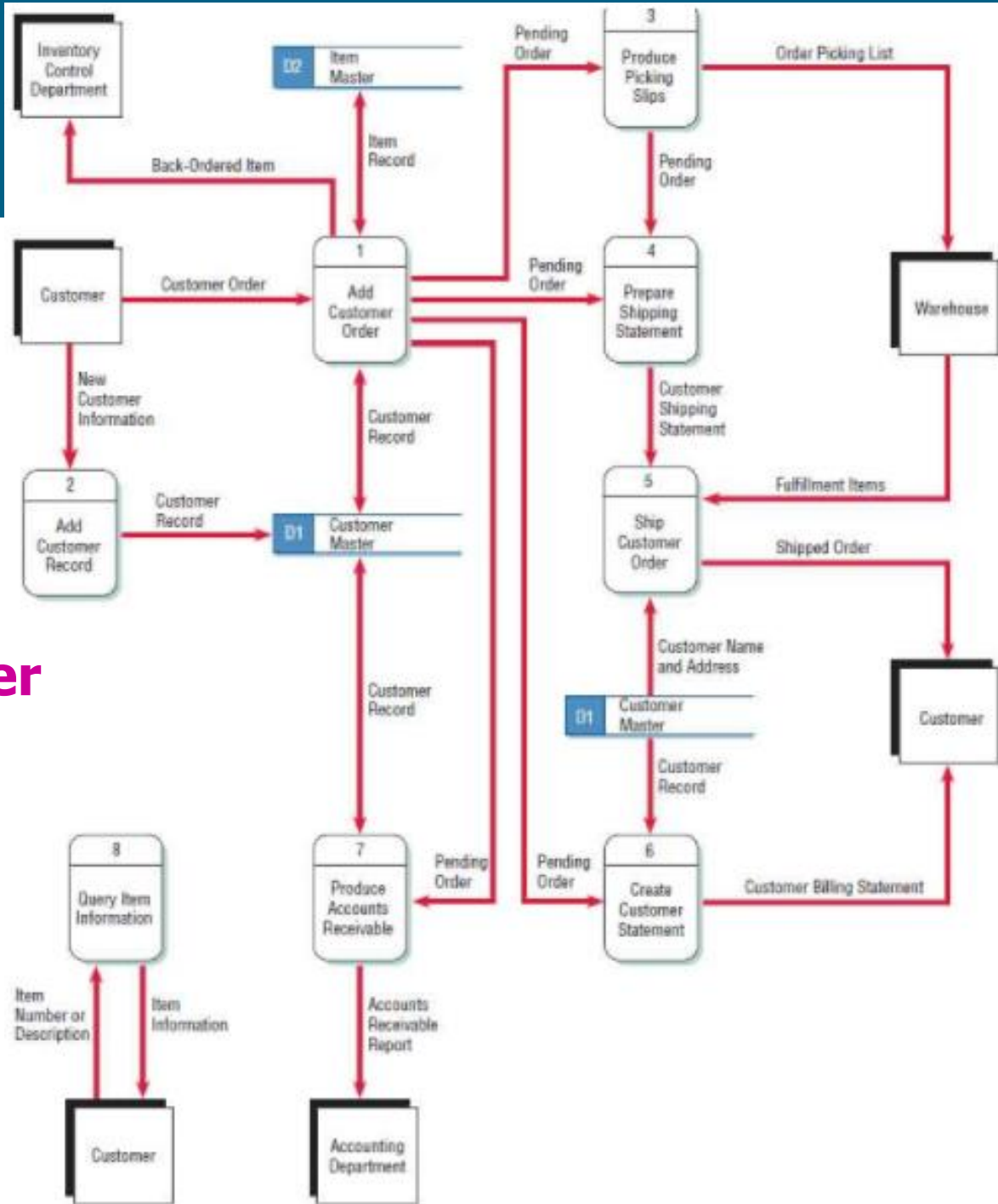
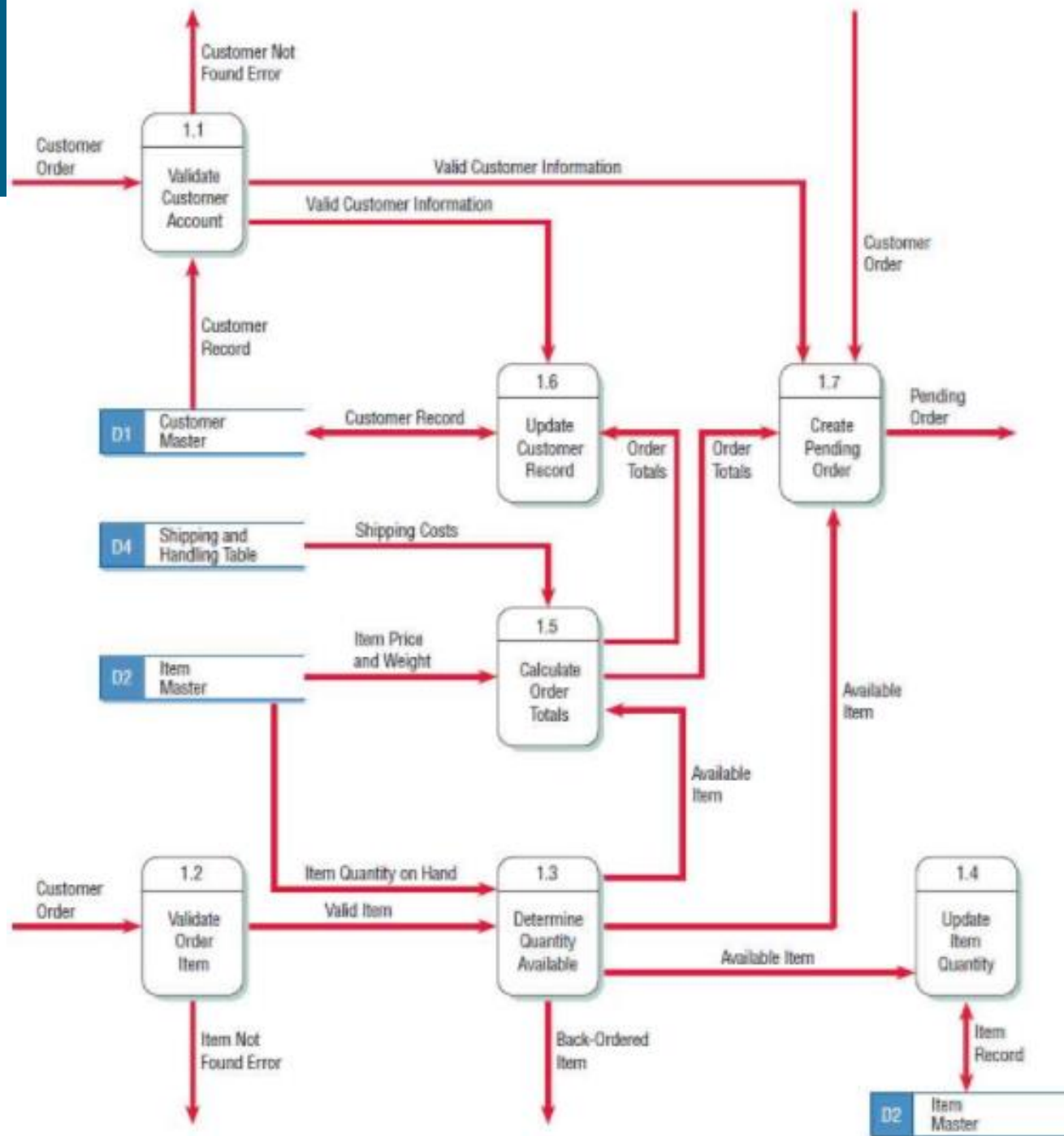


Diagram 1 of the order processing system



Logical and Physical Data Flow Diagrams

● Logical

- Focuses on the business and how the business operates
- Not concerned with how the system will be constructed
- Describes the business events that take place and the data required and produced by each event

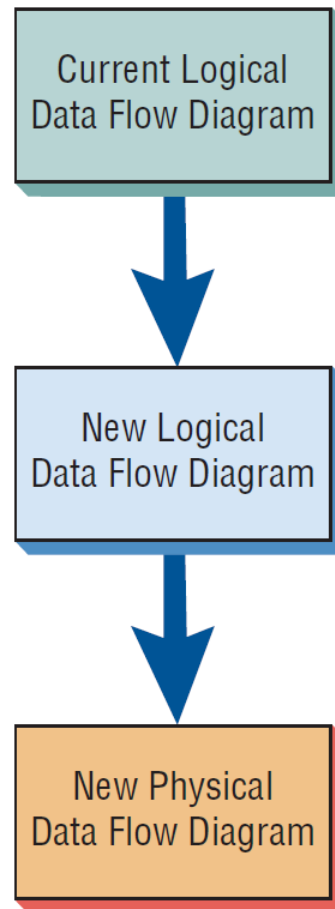
● Physical

- Shows how the system will be implemented
- Depicts the system

Features Common of Logical and Physical Data Flow Diagrams (Figure 7.7)

Design Feature	Logical	Physical
What the model depicts	How the business operates.	How the system will be implemented (or how the current system operates).
What the processes represent	Business activities.	Programs, program modules, and manual procedures.
What the data stores represent	Collections of data regardless of how the data are stored.	Physical files and databases, manual files.
Type of data stores	Show data stores representing permanent data collections.	Master files, transition files. Any processes that operate at two different times must be connected by a data store.
System controls	Show business controls.	Show controls for validating input data, for obtaining a record (record found status), for ensuring successful completion of a process, and for system security (example: journal records).

The Progression of Models from Logical to Physical (Figure 7.8)



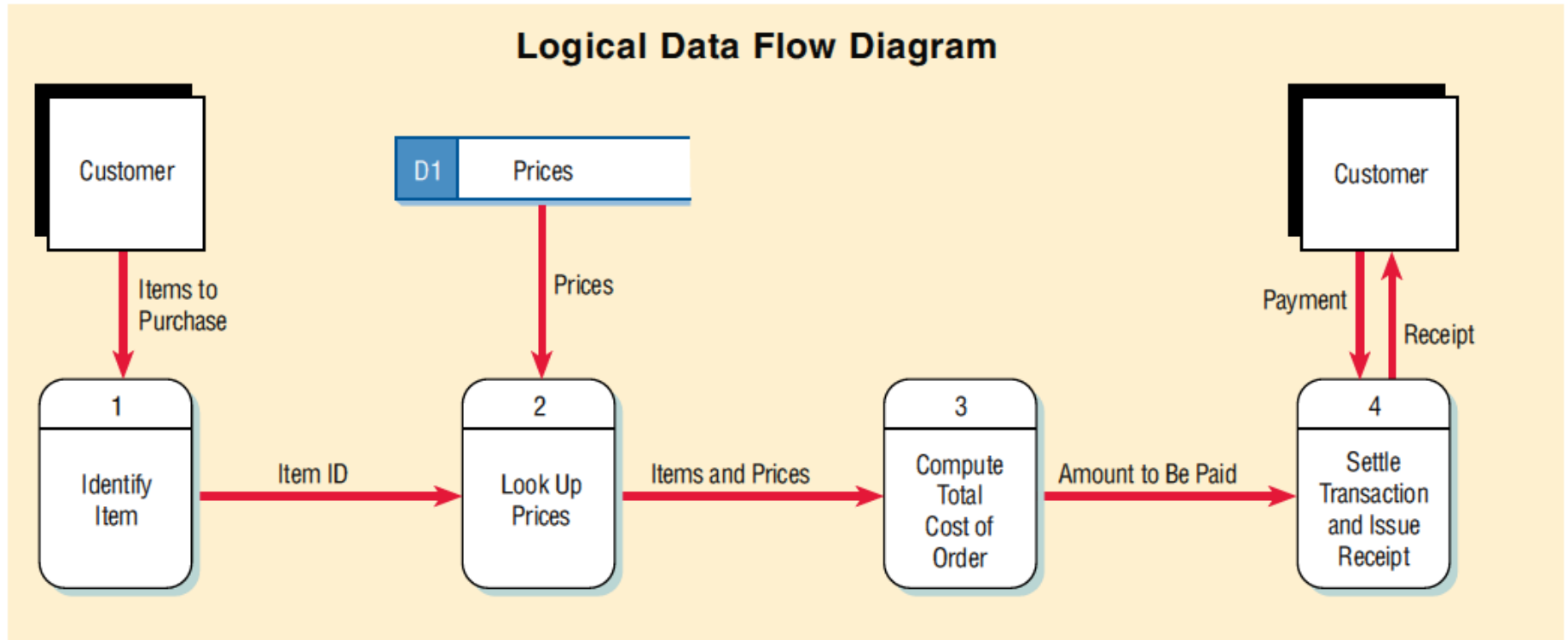
Derive the logical data flow diagram for the current system by examining the physical data flow diagram and isolating unique business activities.

Create the logical data flow diagram for the new system by adding the input, output, and processes required in the new system to the logical data flow diagram for the current system.

Derive the physical data flow diagram by examining processes on the new logical diagram. Determine where the user interfaces should exist, the nature of the processes, and necessary data stores.

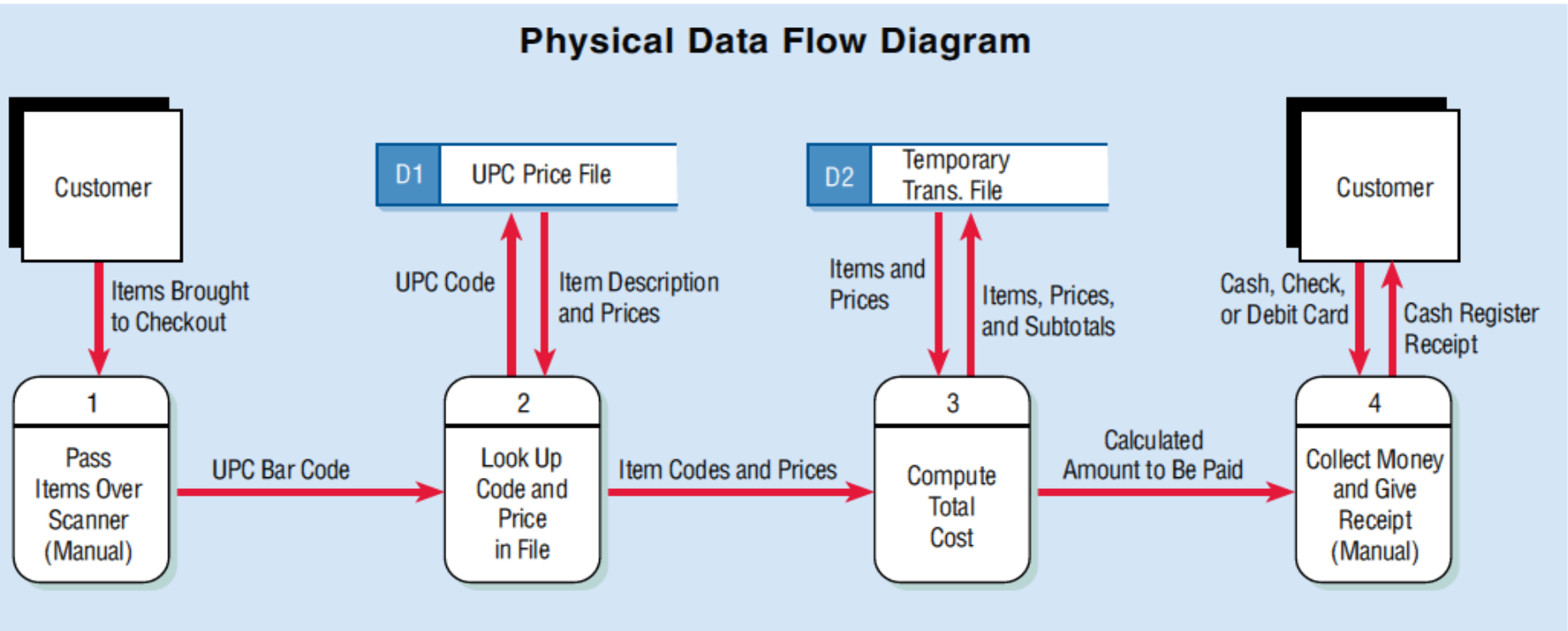
Logical Data Flow Diagram Example

(Figure 7.9)



Physical Data Flow Diagram

Example (Figure 7.9)



Developing Logical Data Flow Diagrams

- Better communication with users
- More stable systems
- Better understanding of the business by analysts
- Flexibility and maintenance
- Elimination of redundancy and easier creation of the physical model

Developing Physical Data Flow Diagrams

- Clarifying which processes are performed by humans and which are automated
- Describing processes in more detail
- Sequencing processes that have to be done in a particular order
- Identifying temporary data stores
- Specifying actual names of files and printouts
- Adding controls to ensure the processes are done properly

Physical Data Flow Diagrams Contain Many Items Not Found in Logical Data Flow Diagrams (Figure 7.10)

Contents of Physical Data Flow Diagrams

- Manual processes
- Processes for adding, deleting, changing, and updating records
- Data entry and verifying processes
- Validation processes for ensuring accurate data input
- Sequencing processes to rearrange the order of records
- Processes to produce every unique system output
- Intermediate data stores
- Actual file names used to store data
- Controls to signify completion of tasks or error conditions

CRUD Matrix

- The acronym CRUD is often used for
 - Create
 - Read
 - Update
 - Delete
- These are the activities that must be present in a system for each master file
- A CRUD matrix is a tool to represent where each of these processes occurs in a system

CRUD Matrix (Figure 7.11)

Activity	Customer	Item	Order	Order Detail
Customer Logon	R			
Item Inquiry		R		
Item Selection		R	C	C
Order Checkout	U	U	U	R
Add Account	C			
Add Item		C		
Close Customer Account	D			
Remove Obsolete Item		D		
Change Customer Demographics	RU			
Change Customer Order	RU	RU	RU	CRUD
Order Inquiry	R	R	R	R

Use Cases and Data Flow Diagrams

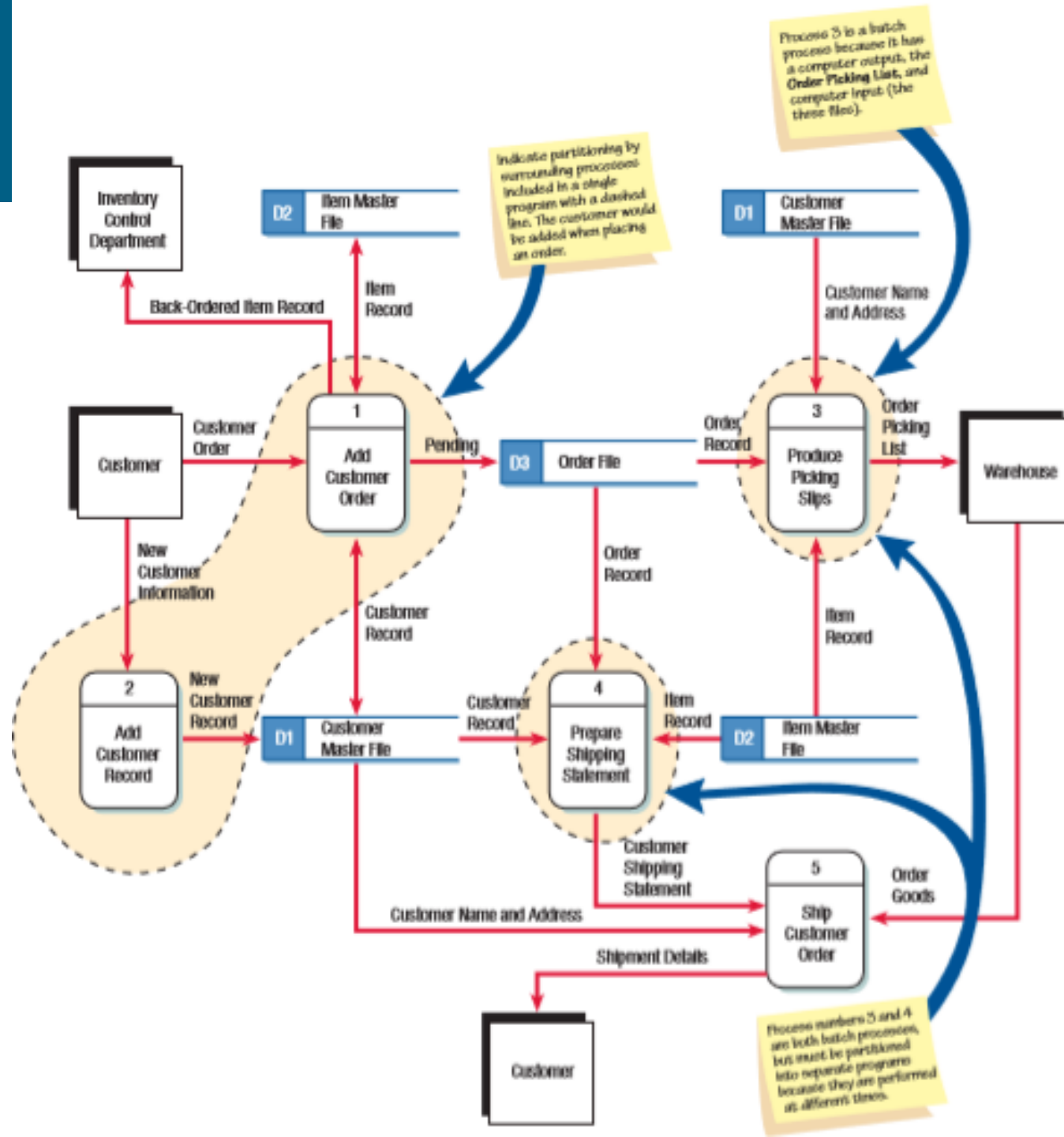
- Each use case defines one activity and its trigger, input, and output
- Allows the analyst to work with users to understand the nature of the processes and activities and then create a single data flow diagram fragment

Partitioning Data Flow Diagrams

- Partitioning is the process of examining a data flow diagram and determining how it should be divided into collections of manual procedures and computer programs
- A dashed line is drawn around a process or group of processes that should be placed in a single computer program

Reasons for Partitioning

- Different user groups
- Timing
- Similar tasks
- Efficiency
- Consistency of data
- Security



Process Analysis and Design

1. Data Flow Diagram Symbols
2. Data Flow Diagram levels
3. Creating DFDs
4. Physical and logical DFDs

Data Analysis and Design

Data Analysis and Design

1. Entity – Relationship Data Model
2. Steps to convert ERD to Relational model

CONTENT

1. Entity-Relationship data model
 - a. Entities, Attributes, entity sets
 - b. Relationship and relationship sets
 - c. Key
 - d. Relationship cardinality
 - e. Extended Entity-relationship model
2. Entity- Relationship model to Relational model
 - a. Converting Class Hierarchies
 - b. Converting Entity set to tables
 - c. Converting Relationships
 - d. Normalization

Review

Entity-Relationship data model

Entity Relationship Data Model (ERD)

1. Entities, Attributes, entity sets
2. Relationship and relationship sets
3. Key
4. Participation constraints
5. Extended Entity-relationship model

Entity Relationship Data Model (ERD)

- Entity Relationship Data Model (ERD)
invented by Peter Pin-Shan CHEN in 1976.
- ERD describes the data in a real-world enterprise in terms of objects and their relationships.
- ERD is used for database design in the logical design

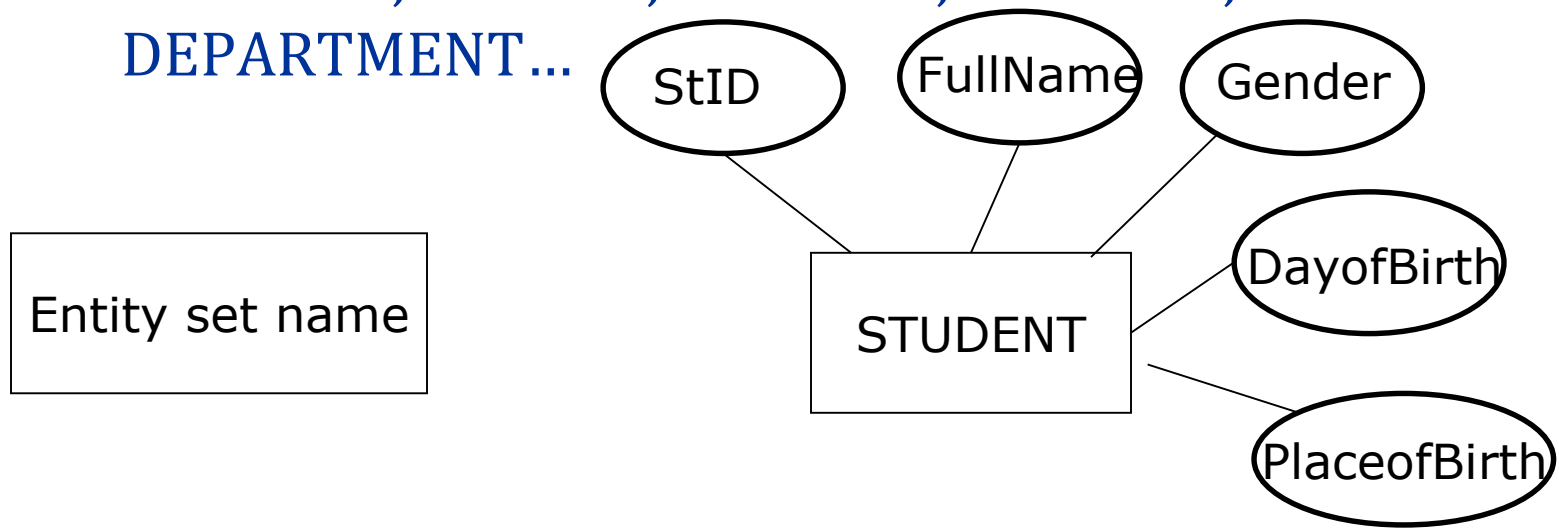
Entities, Attributes, entity sets

- Entity:

- an object in the real world.
- A Student with some **attributes**: (16520098, Nguyen Van Manh, 1.1.1990)

- Entity set:

- a collection of entity.
- STUDENT, COURSE, LECTURE, FACULTY, DEPARTMENT...



Types of Attribute

- Simple

- Unique value.
- StudentID, NameofCourse, NumberofCredit,...

- Composite

- The value of attribute can be divided into some smaller parts
- Address: (house number, Street, District, City, Country), Name (FirstName, MidName, LastName).

- Multi-valued

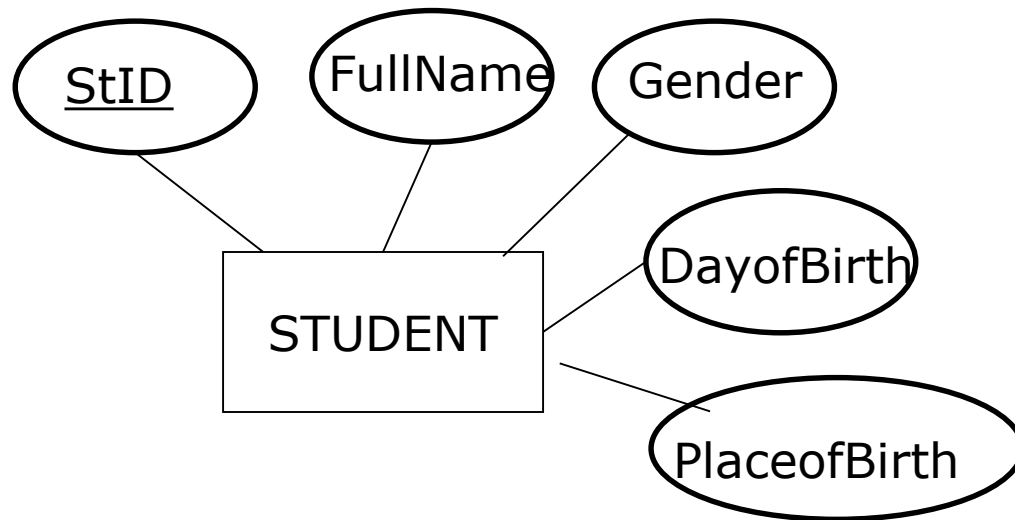
- The value of attribute can have various values for one entity

- Derived

- The value of attribute can be calculated from other attributes
- The value of Result based on the values of Score.

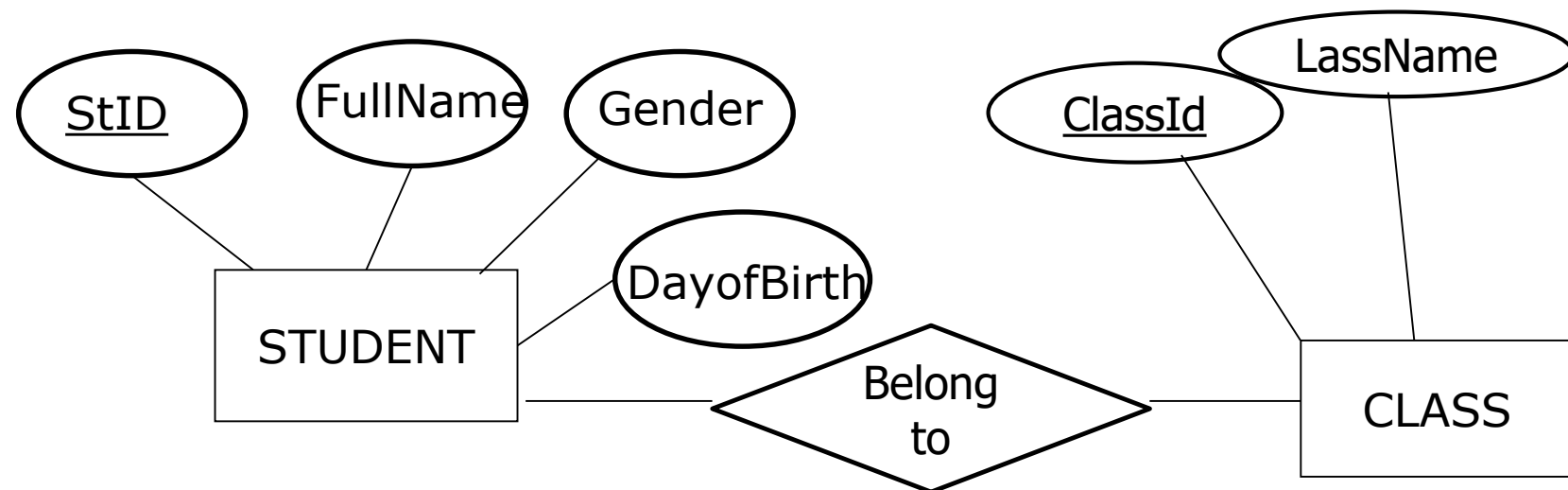
Key of an entity set

- Key is a minimal set of attributes whose values uniquely identify an entity in the set
- Each student has unique StusentID



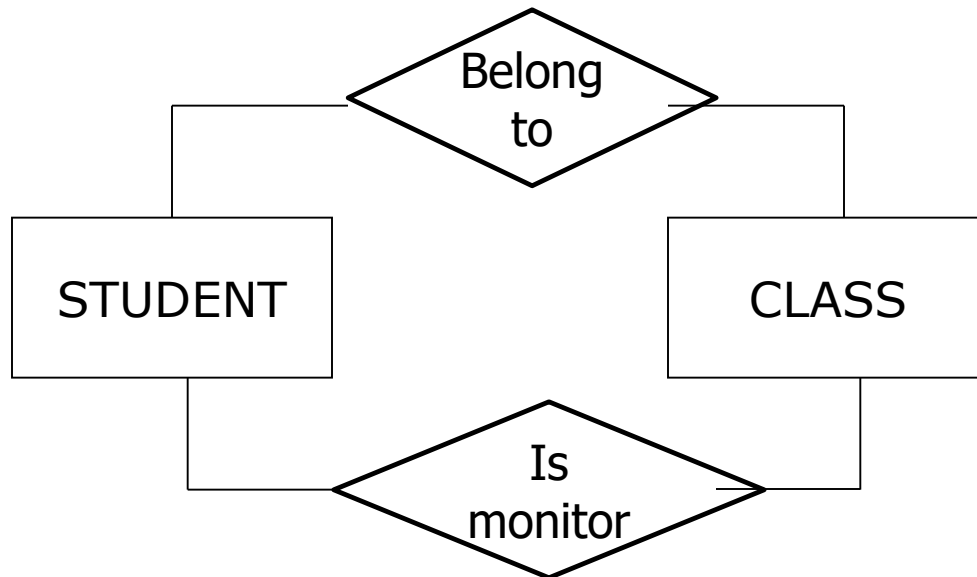
Relationship Sets

- A relationship is an association among 2 or more entities. Ex: Student A belongs to the class named IS2016
- A set of similar relationship is a relationship set

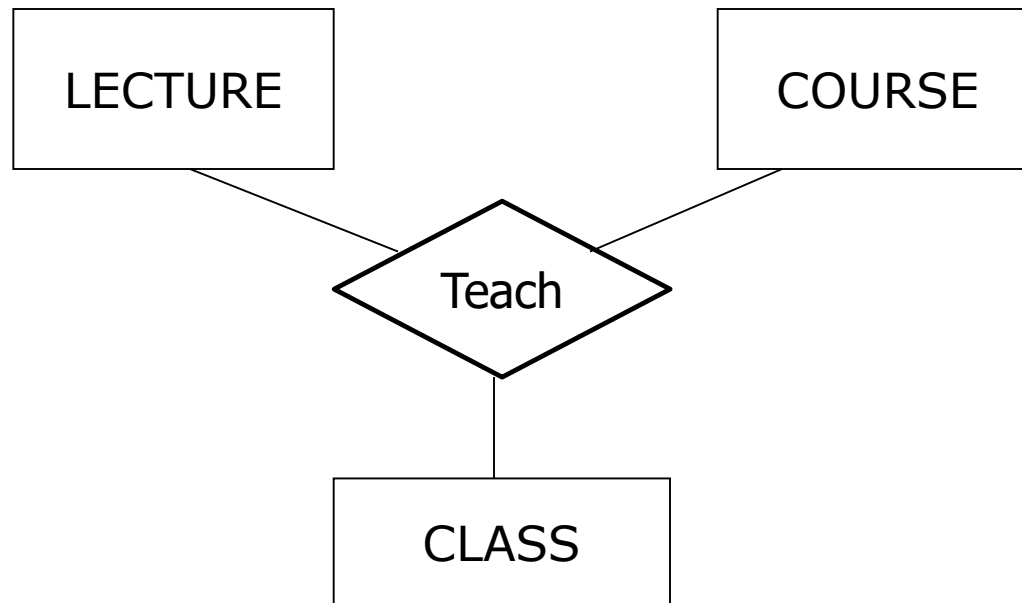


Relationship Sets

- Among entities may have more than one relationship set

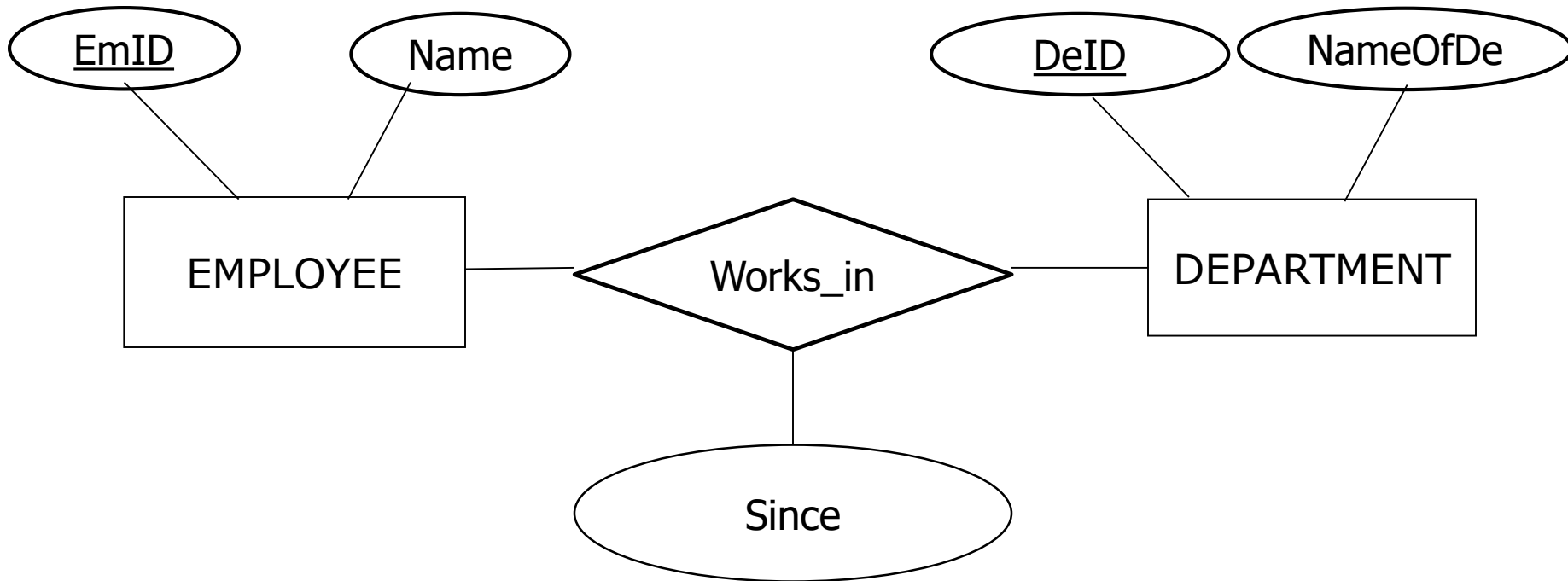


Relationship Sets

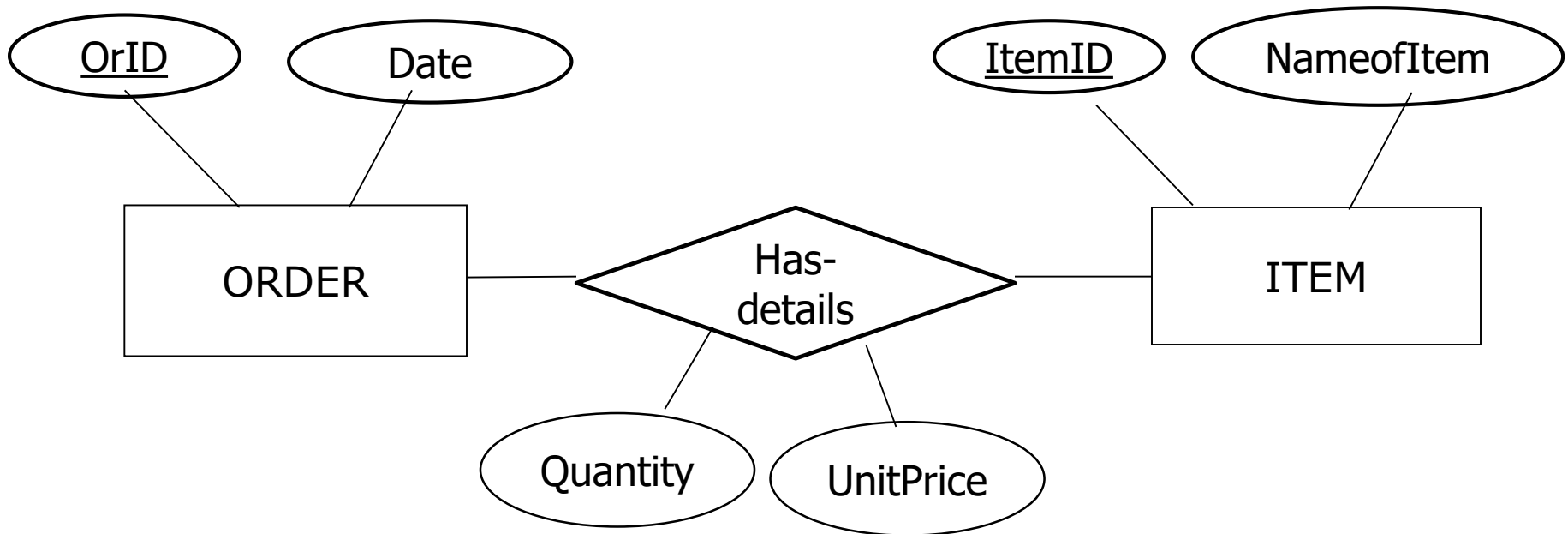


Attribute of relationship set

- To record the information about the relationship

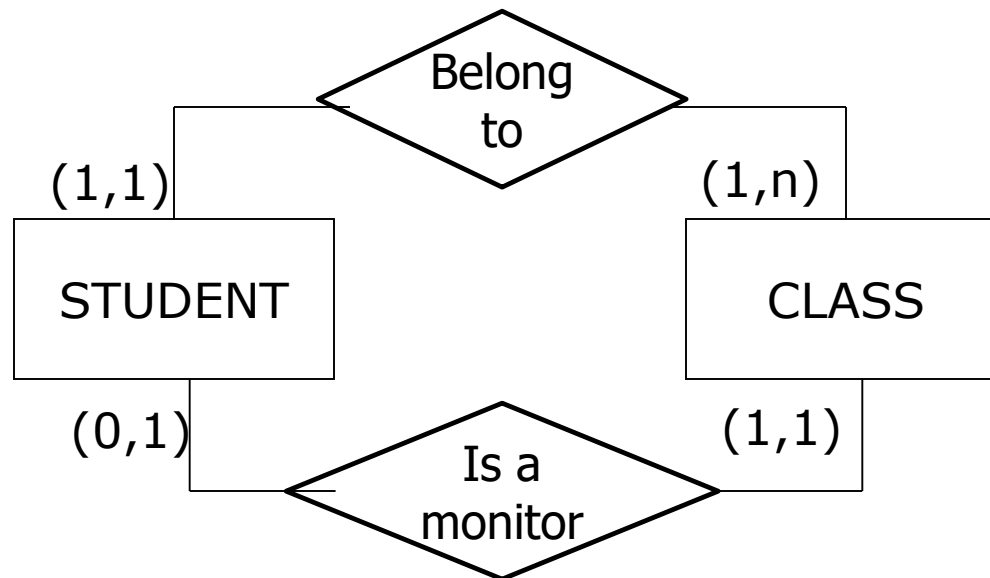


Attribute of relationship set

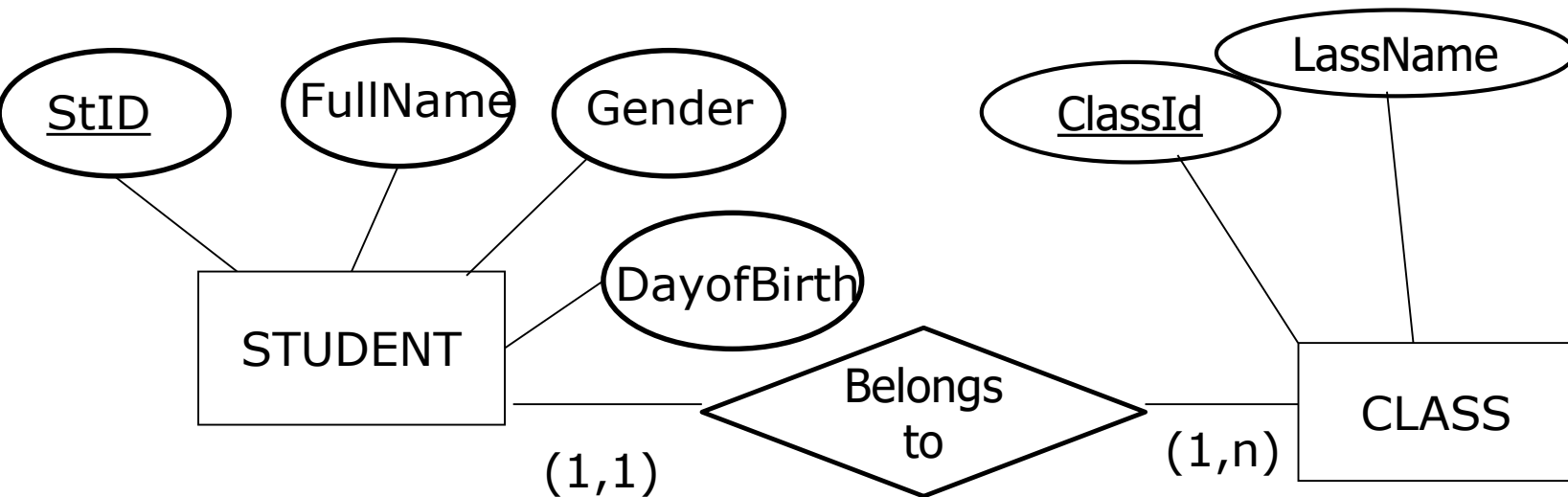


Relationship cardinality

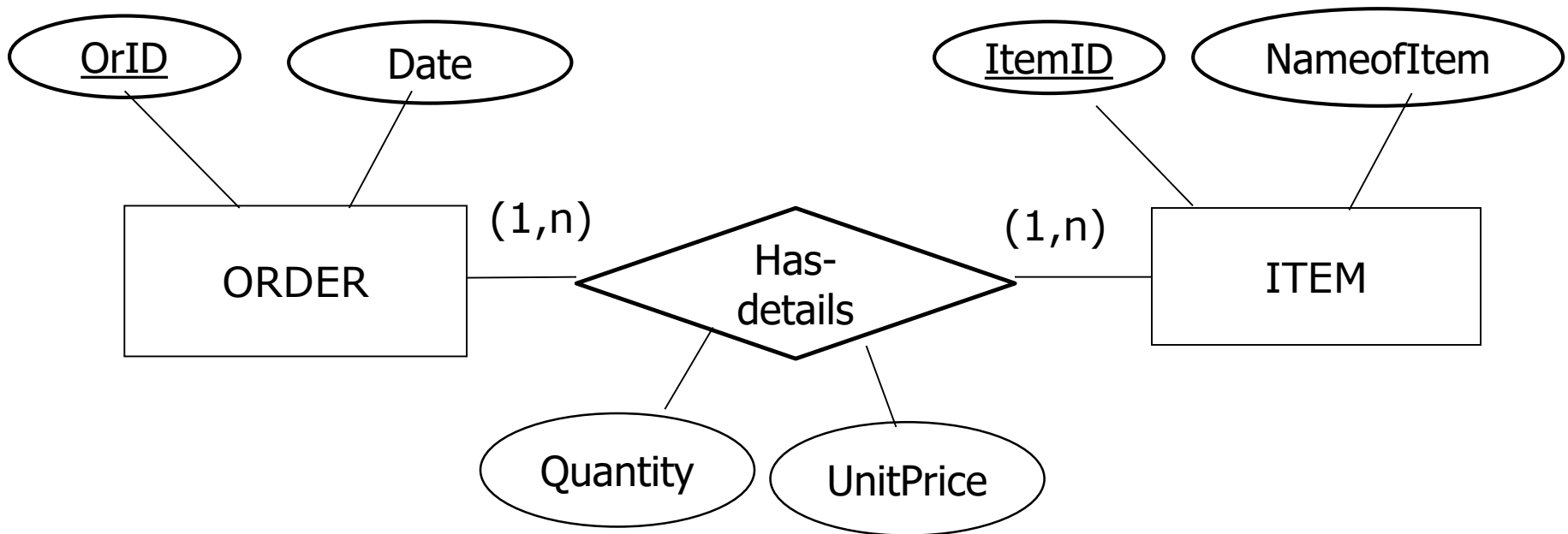
- Shows the minimum and maximum entities associate with relationship.
- (min, max)



Relationship cardinality



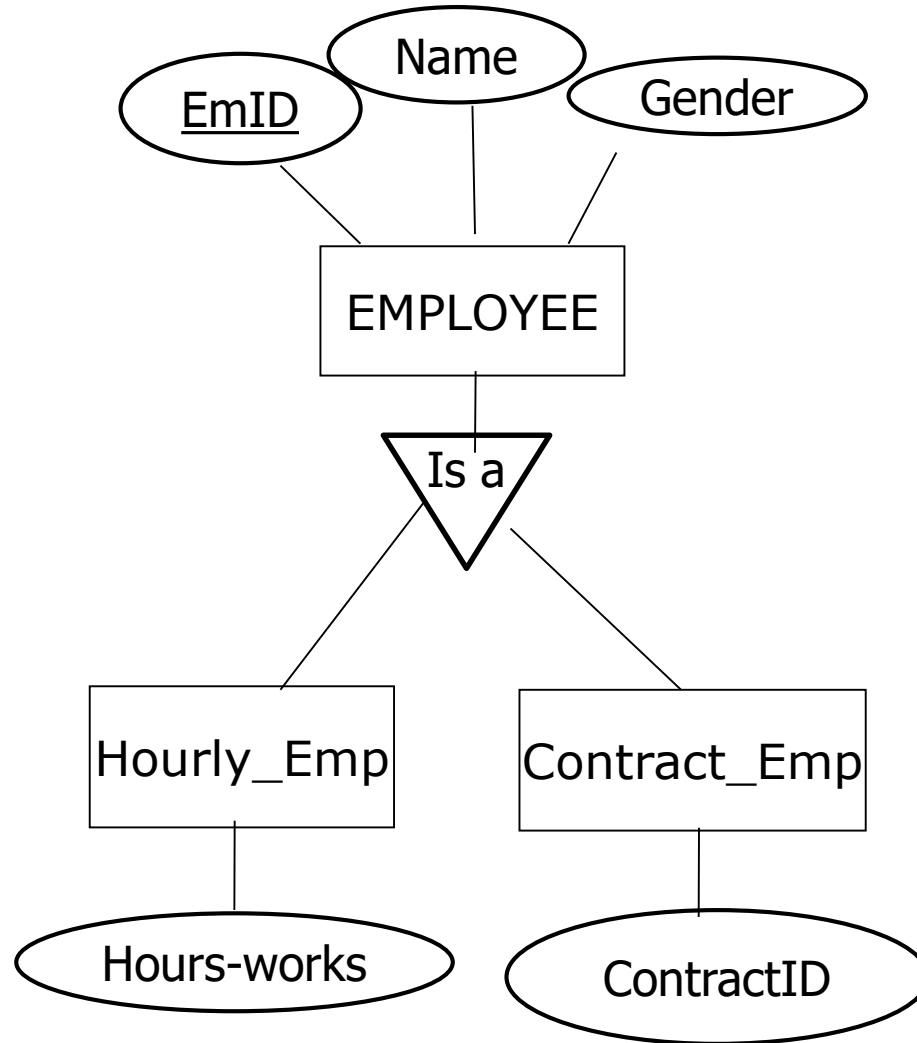
Relationship cardinality



Extended Entity-relationship model

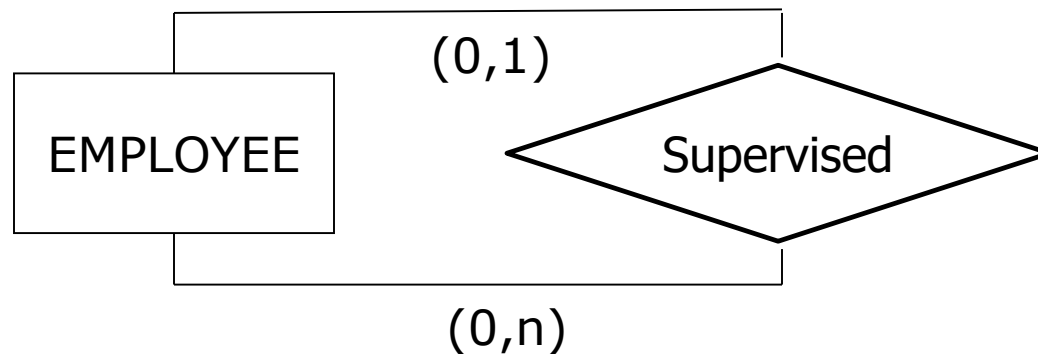
- Generalization and Specialization
- Recursive Relationship
- Weak entity
- Extended relationship

Generalization and Specialization



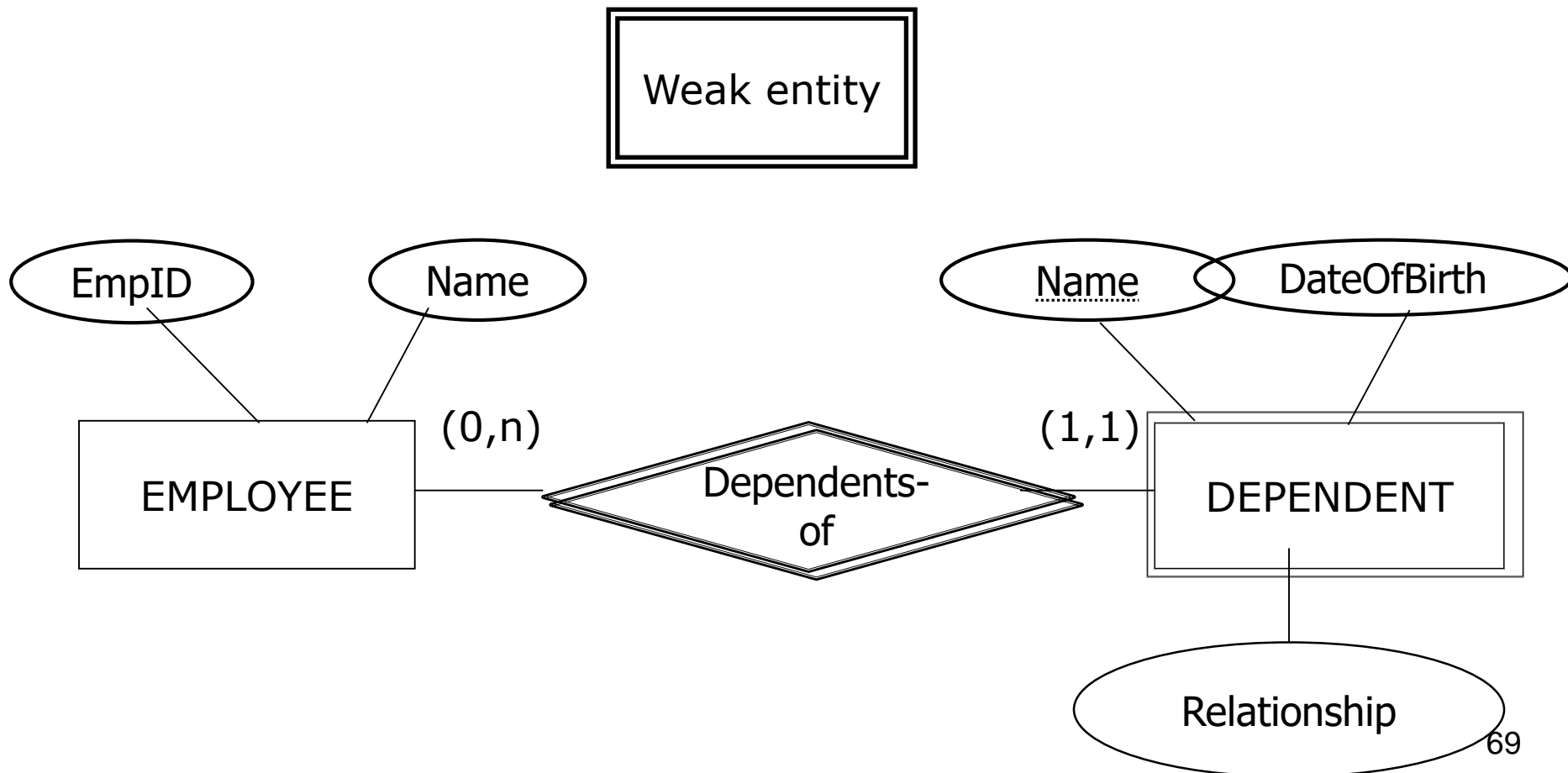
Recursive Relationship

- Relationship with the same entity.

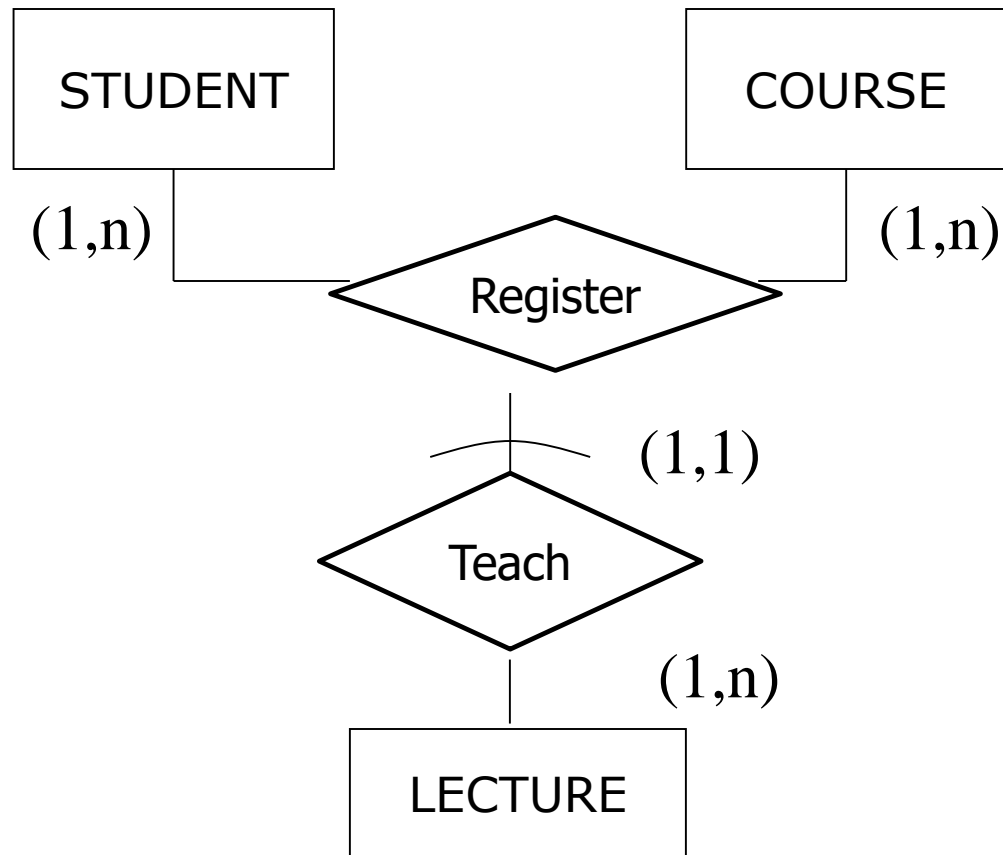


Weak entity

- Entity has no key.
- It depends on other entity



Extended relationship





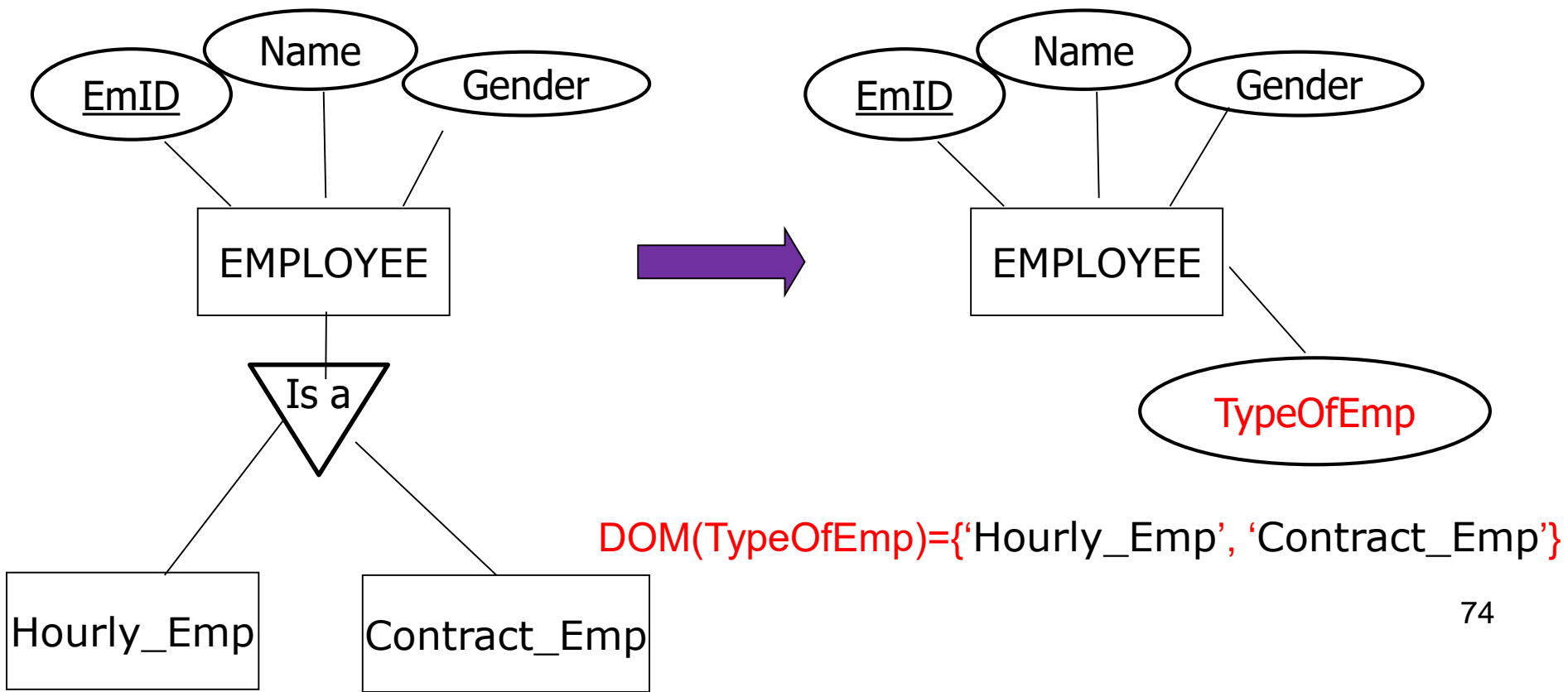
Converting Entity-Relationship data model to Relational data model

Converting Entity-Relationship data model to Relational data model

1. Converting Class Hierarchies
2. Converting Entity set
3. Converting Relationships
4. Normalization

Converting Class Hierarchies

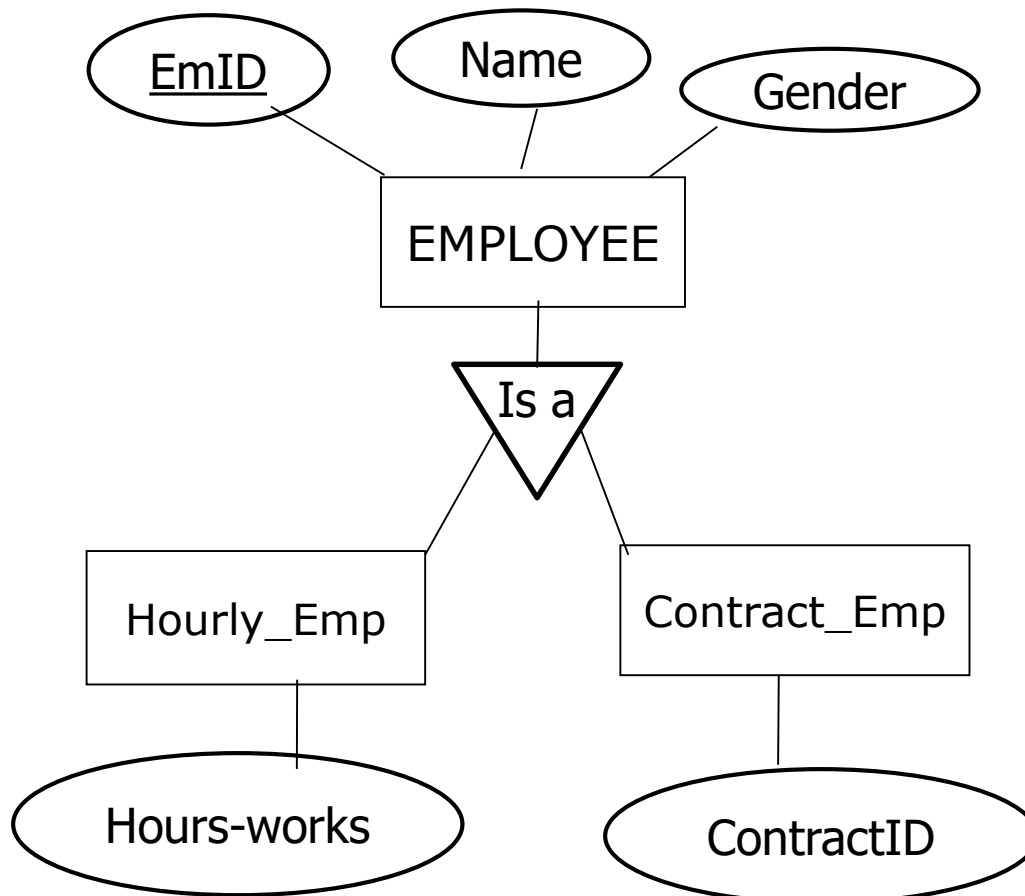
1. Specialization entities do not have their own attributes
 - Adding Type attribute to the Generalization entity
 - Adding constrains for the value of Type attribute



Converting Class Hierarchies

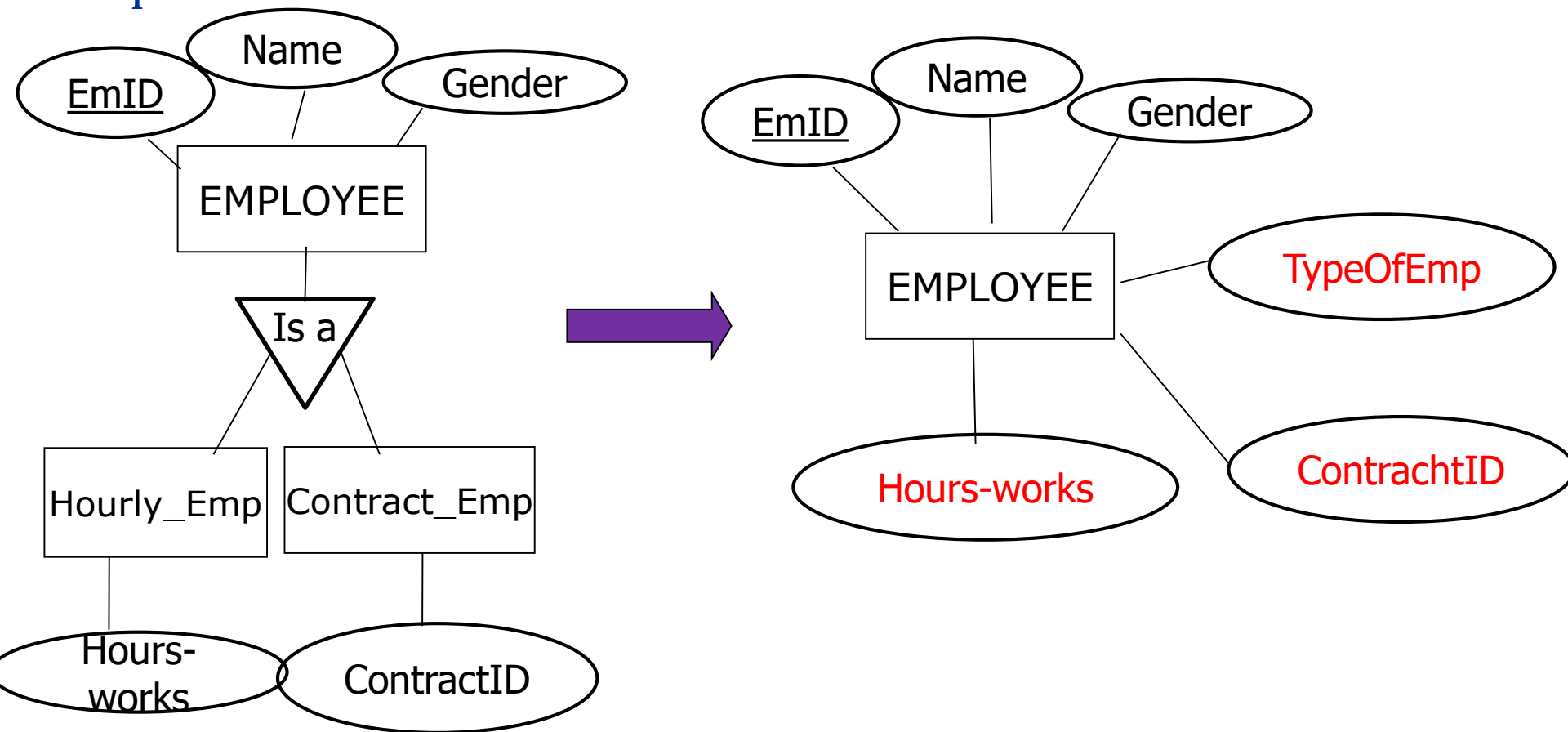
1. Specialization entities have a few attributes

- Adding to the Generalization entity: Type attribute, specialization attributes
- Adding constraints for: the value of Type attribute, specialization attributes



Converting Class Hierarchies

2. Specialization entities have a few attributes

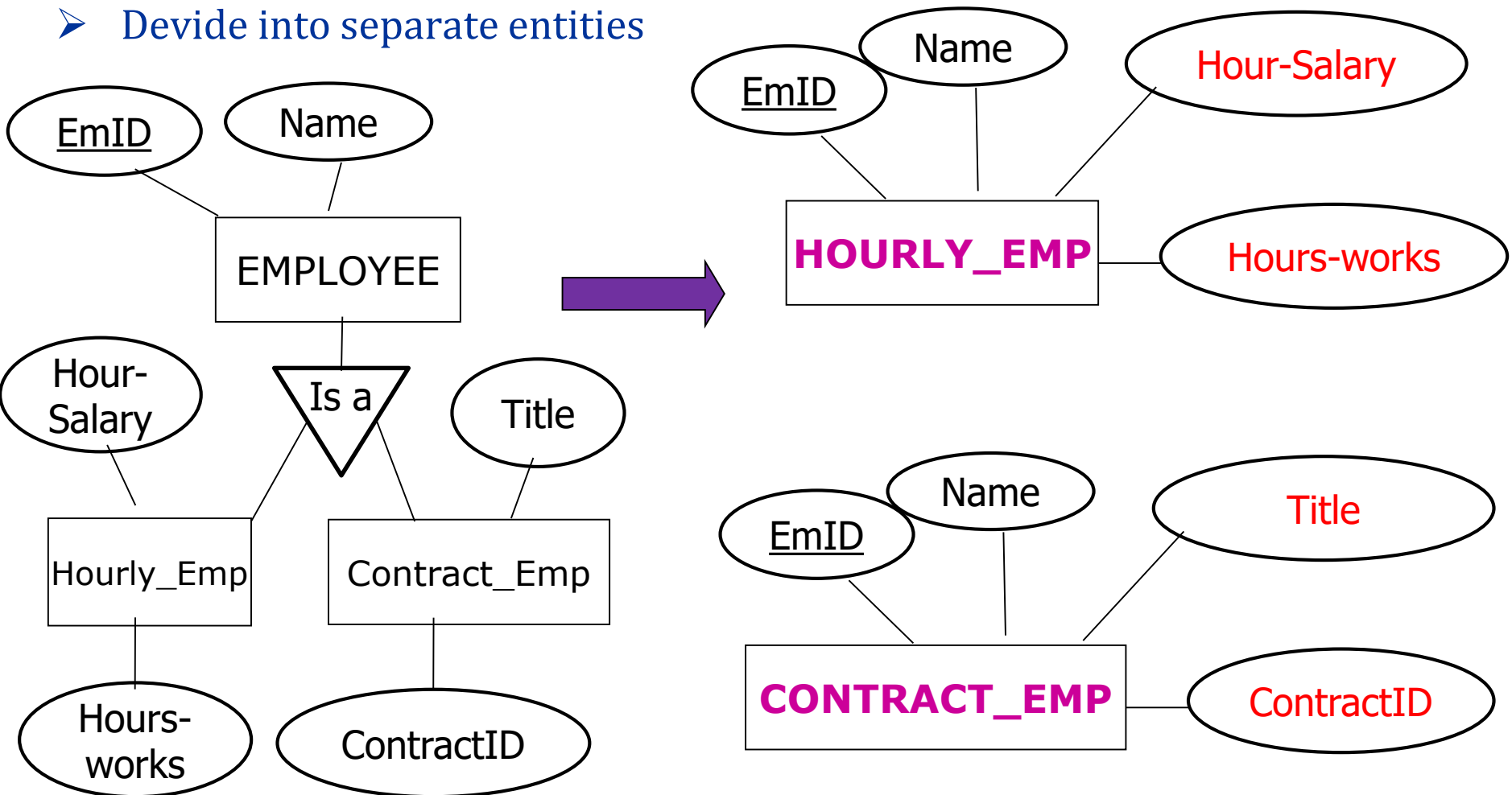


1. $DOM(\text{TypeOfEmp}) = \{\text{'Hourly_Emp'}, \text{'Contract_Emp'}\}$
2. If $\text{TypeOfEmp} = \text{Hourly_Emp}$ then Hours-works is enabled and ContractID is disabled
3. If $\text{TypeOfEmp} = \text{Contract_Emp}$ then Hours-works is disabled and ContractID is enabled

Converting Class Hierarchies

3. Specialization entities have a lot of attributes

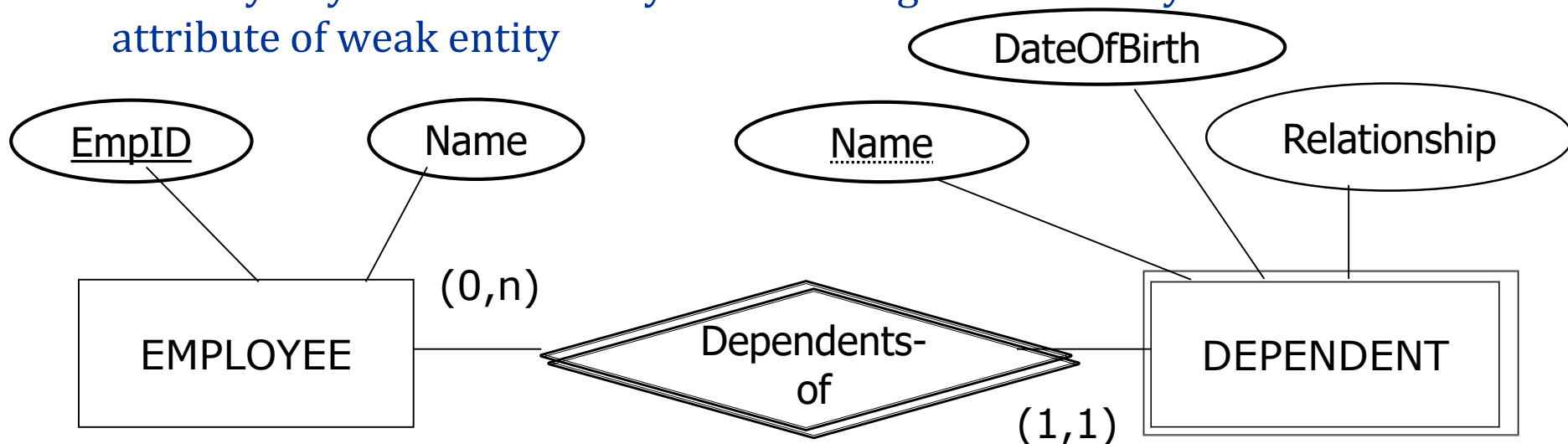
- Devide into separate entities



Converting Entity set

1. Weak Entity

- Name of the table: Name of the entity
- Attributes of the table: Key of the strong related entity, and attributes of the weak entity
- Primary Key of the table: Key of the strong related entity and the difference attribute of weak entity

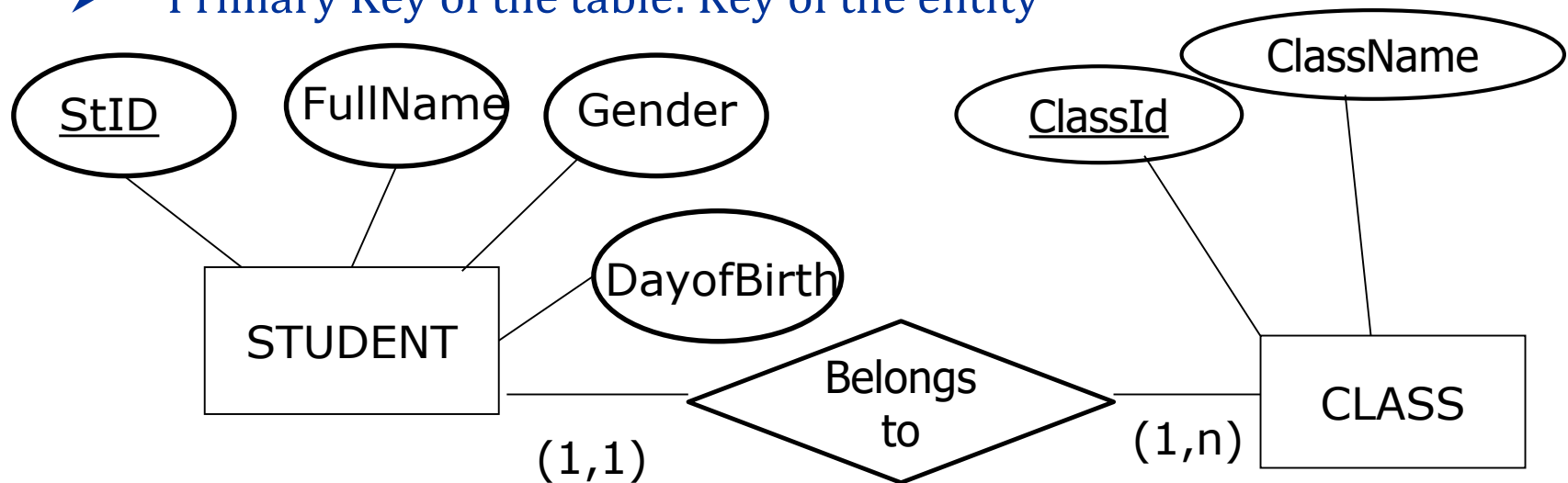


DEPENDENT(EmpID, Name, DateOfBirth, Relationship)

Converting Entity set

2. Entity set

- Name of the table: Name of the entity
- Attributes of the table: Attributes of the entity
- Primary Key of the table: Key of the entity

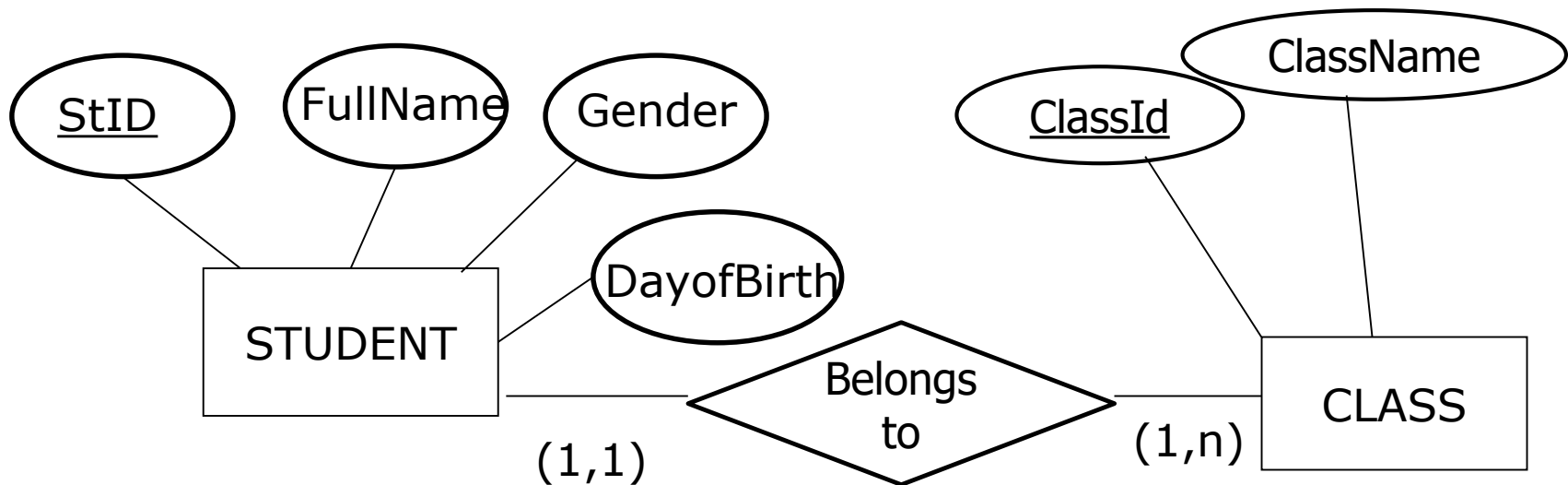


1. STUDENT(StID, FullName, Gender, DayOfBirth)
2. CLASS (ClassID, ClassName)

Converting Relationships

1. (1, 1) and (1,n) relationship

- Do not become a new table
- Add key of the (1,n) entity to the table represented the (1,1) entity.

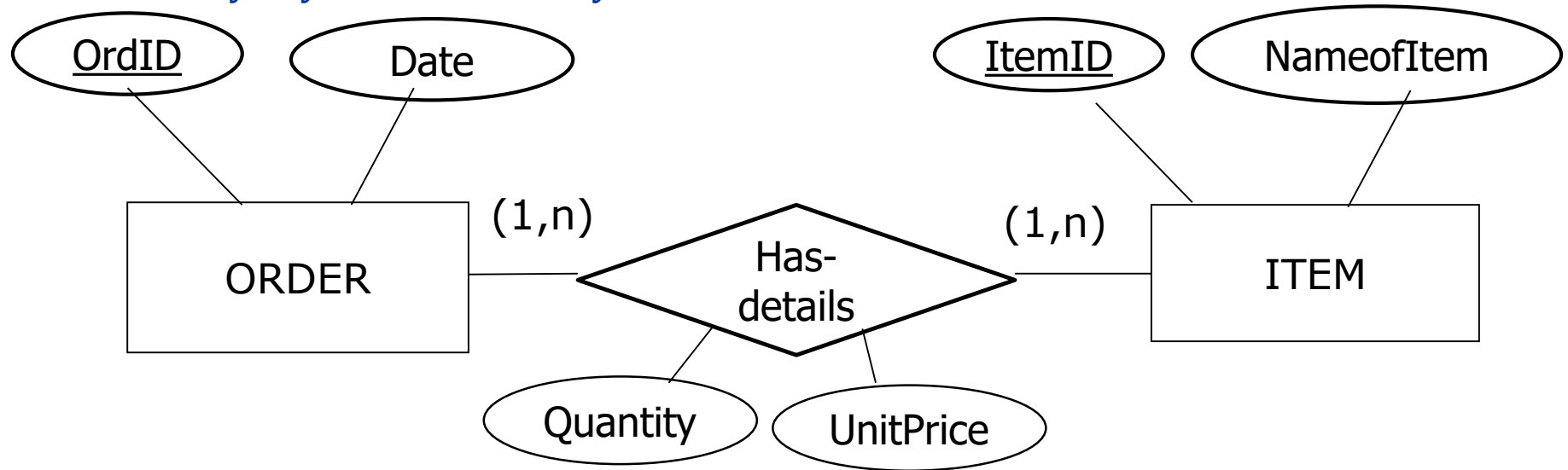


1. STUDENT(StID, FullName, Gender, DayOfBirth, **ClassID**)
2. CLASS (ClassID, ClassName)

Converting Relationships

2. (1, n) and (1,n) relationship

- Do become a new table
- Attributes of the new table: keys of related entities and attributes of the relationship
- Primary key of the table: Key of related entities.



ORDER_ITEM (OrdID, ItemID, Quantity, UnitPrice)



CONTENT

1. Entity-Relationship data model
 - a. Entities, Attributes, entity sets
 - b. Relationship and relationship sets
 - c. Key
 - d. Relationship cardinality
 - e. Extended Entity-relationship model
2. Entity- Relationship model to Relational model
 - a. Converting Class Hierarchies
 - b. Converting Entity set to tables
 - c. Converting Relationships
 - d. Normalization

CONTENT

1. Process Analysis and Design

- a. Data Flow Diagram Symbols
- b. Data Flow Diagram levels
- c. Creating DFDs
- d. Physical and logical DFDs

2. Data Analysis and Design

- a. Entity – Relationship Data Model
- b. Steps to convert ERD to Relational model

Reference

1. Kendall & Kendall, *Systems Analysis and Design*, 9th edition, Prentice Hall, 2014.
- 2.