UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

TOPIC 7

PROCESS AND DATA MODELLING

ANALISIS DAN PERANCANGAN SISTEM INFORMASI

CSIM603183

Learning Objectives

1. Able to explain the concept of process and data modelling
2. Able to explain the rules for creating data flow diagram (DFD)
3. Able to create data flow diagram (DFD)
4. Able to explain the rules for creating entity relationship diagram (ERD)
5. Able to create entity relationship diagram (ERD)

Outline

1. Data flow diagrams concept
2. Creating data flow diagrams
3. ERD concept
4. Creating data flow diagrams

1.1
DATA FLOW DIAGRAMS

Introduction

- **Process model**
  - ❖A formal way of representing how a business system operates
  - ❖Illustrates the activities or processes that are performed and how data moves among them
  - ❖Both for as-is and to-be systems
  - ❖Part of structured systems analysis and design techniques
- **Data flow diagramming**
  - ❖a technique that diagrams the business processes and the data that pass among them.

Introduction

- **Logical** process models describe processes without suggesting how they are conducted → **Analysis phase**
- **Physical** process models provide information that is needed to build the system → **Design phase**

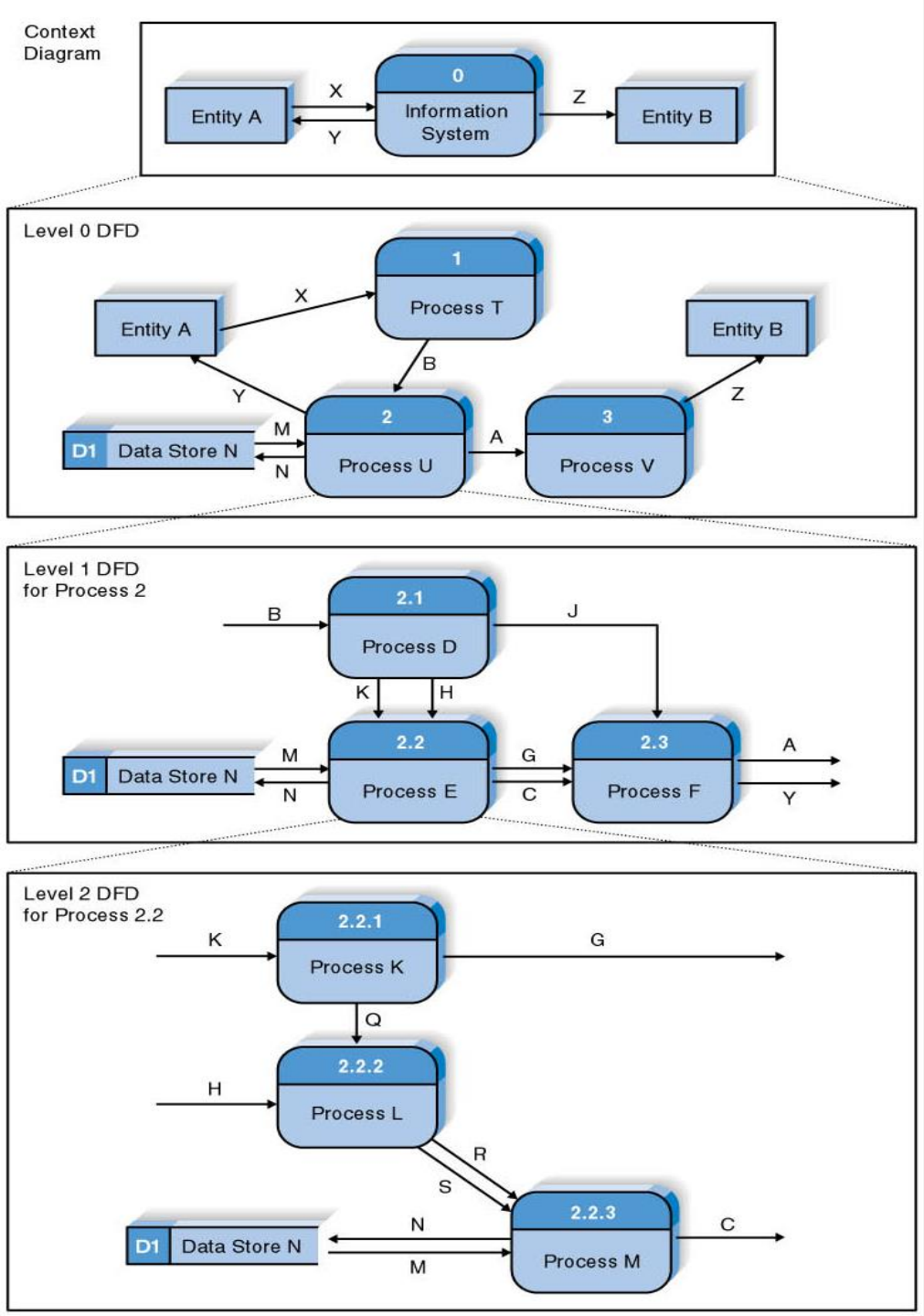
Creating DFD

- Use-cases (**requirement definition and use-case description**) is the main and fundamental reference in creating DFD.
- Because most business processes are too complex to be explained in one DFD, process models are therefore composed of a set of DFDs.

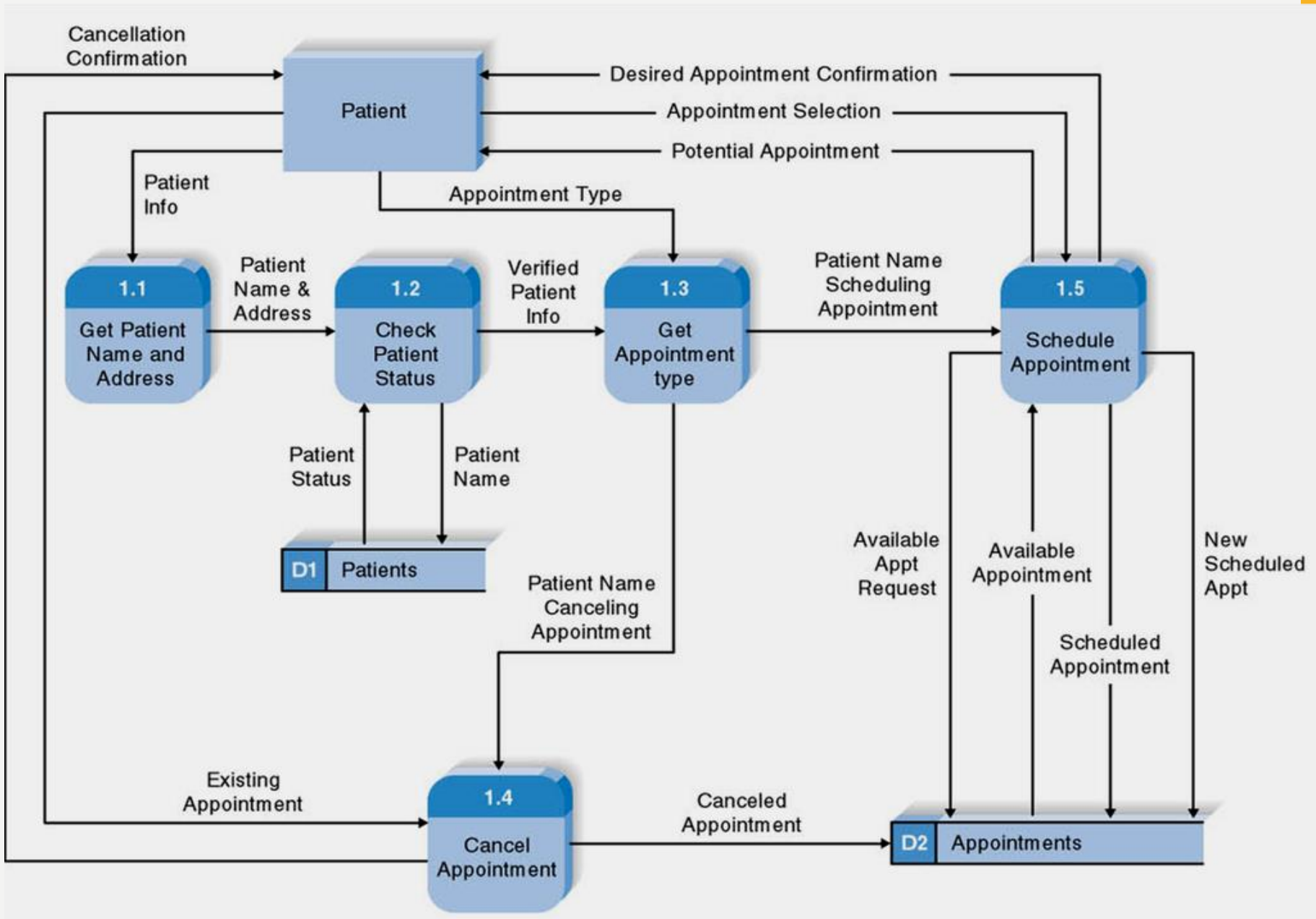
Creating DFD

- The **first DFD provides a summary** of the overall system, with additional DFDs providing more and more detail about each part of the overall business process.
- Thus, one important principle in process modeling with DFDs is **the decomposition of the business process** into a hierarchy of DFDs, with each level down the hierarchy representing less scope but more detail.

Relationship among Levels of DFDs



Reading DFD's



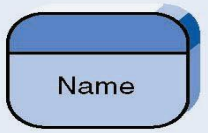
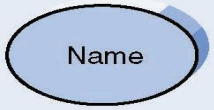
Elements of Data Flow Diagrams

Process

Data flow

Data store

External entity

Data Flow Diagram Element	Typical Computer-Aided Software Engineering Fields	Gane and Sarson Symbol	DeMarco and Yourdan Symbol
Every *process* has A number A name (verb phase) A description One or more output data flows Usually one or more input data flows	Label (name) Type (process) Description (what is it) Process number Process description (Structured English) Notes		
Every *data flow* has A name (a noun) A description One or more connections to a process	Label (name) Type (flow) Description Alias (another name) Composition (description of data elements) Notes		
Every *data store* has A number A name (a noun) A description One or more input data flows Usually one or more output data flows	Label (name) Type (store) Description Alias (another name) Composition (description of data elements) Notes		
Every *external entity* has A name (a noun) A description	Label (name) Type (entity) Description Alias (another name) Entity description Notes		

Elements of Data Flow Diagrams

- **Process**
  - An activity or function performed for a specific business reason
  - Manual or computerized
  - Every process should be named starting with a verb and ending with a noun (e.g., “Get Patient Information”).
  - Short, but clear (contains enough information).
  - Generally, performs only one activity. Avoid using the word “and” in process names.
  - Every process must have at least one input data flow and at least one output data flow.

Elements of Data Flow Diagrams

- **Data flow**
  - A single piece of data (e.g., patient name, available schedule) or a logical collection of data (e.g. new appointment)
  - Data flows are the glue that holds the processes together.
  - Every data flow should be named with a noun.
  - Always starts or ends at a process.
  - One end of every data flow will always come from or go to a process, with the arrow showing the direction into or out of the process.
  - Data flows show what inputs go into each process and what outputs each process produces.

Elements of Data Flow Diagrams

- **Data flow**
  - Every process must create at least one output data flow, because if there is no output, the process does not do anything.
  - Likewise, each process has at least one input data flow, because it is difficult, if not impossible, to produce an output with no input.

Elements of Data Flow Diagrams

- **Data Store**
  - A collection of data that is stored in some way
  - Every data store is named with a noun and is assigned an identification number and a description.
  - Data stores form the starting point for the data model (discussed in the next chapter) and are the principal link between the process model and the data model.
  - Data flows coming out of a data store indicate that information is retrieved from the data store.
  - Data flows going into a data store indicate that information is added or updated to the data store.

Elements of Data Flow Diagrams

- **Data Store**
  - All data stores must have at least one input data flow (or else they never contain any data), unless they are created and maintained by another information system or on another page of the DFD.
  - Likewise, they have at least one output data flow on some page of the DFD. (Why store data if you never use it?)
  - In cases in which the same process both stores data and retrieves data from a data store, there is a temptation to draw one data flow with an arrow on both ends.
  - This practice is incorrect, however. The data flow that stores data and the data flow that retrieves data should always be shown as two separate data flows.

Elements of Data Flow Diagrams

- **External entity**
  - A person, organization, or system that is **external** to the system but interacts with it.
  - The external entity typically corresponds to the primary actor identified in the use case.
  - External entities provide data to the system or receive data from the system, and serve to establish the system boundaries.
  - Every external entity has a name and a description. The key point to remember about an external entity is that it is external to the system, but may or may not be part of the organization.

Using a DFD to Define Business Processes

- *Decomposition* is the process of representing the system in a hierarchy of DFD diagrams
  - Child diagrams show a portion of the parent diagram in greater detail
- ***Balancing*** involves insuring that information presented at one level of a DFD is accurately represented in the next level DFD.

Context Diagram

- First DFD in every business process
- Shows the entire system in context with its environment.
- All process models have one context diagram.
- The context diagram shows the overall business process as just one process (i.e., the system itself) and shows the data flows to and from external entities.
- Data stores usually are not included on the context diagram, unless they are “owned” by systems or processes other than the one being documented.

Level 0 Diagram

- The level 0 diagram shows all the processes at the first level of numbering (i.e., processes numbered 1 through 3), the data stores, external entities, and data flows among them.
- The purpose of the level 0 DFD is to show all the major high-level processes of the system and how they are interrelated.
- All process models have one and only one level 0 DFD.
- Another key principle in creating sets of DFDs is balancing.
- Balancing means ensuring that all information presented in a DFD at one level is accurately represented in the next-level DFD. This doesn't mean that the information is identical, but that it is shown appropriately.

Level 0 Diagram

- Processes and some data flows in Level 0 Diagram might not shown on the context diagram because they are the internal components of process 0.
- The context diagram deliberately hides some of the system's complexity in order to make it easier for the reader to understand.

Level 1 Diagrams

- Each process on the level 0 DFD can be decomposed into a more explicit DFD, called a level 1 diagram, or level 1 DFD, which shows how it operates in greater detail.
- Generally, *level 1 diagram is created for every major process on the level 0 diagram.*
- Shows all the internal processes that comprise a single process on the level 0 diagram.
- Shows how information moves from and to each of these processes.
- If a parent process is decomposed into, for example, three child processes, these three child processes wholly and completely make up the parent process.

Level 2 Diagrams

- Shows all processes that comprise a single process on the level 1 diagram.
- Shows how information moves from and to each of these processes.
- Level 2 diagrams **may not be needed for all** level 1 processes.
- Correctly numbering each process helps the user understand where the process fits into the overall system

Alternative Data Flows

- Where a process can produce different data flows given different conditions.
- We show both data flows and use the process description to explain why they are alternatives
- **Tip --** alternative data flows often accompany processes with IF statements

Alternative Data Flows

- Nothing on the DFD itself shows that the data flows are mutually exclusive.
- For example, process 2.1 on the level 1 DFD produces three output data flows (H, J, K). Without reading the text description of process 2.1, we do not know whether these are produced simultaneously or whether they are mutually exclusive.

Process Descriptions

- Text-based process descriptions provide more information about the process than the DFD alone
- If the logic underlying the process is quite complex, more detail may be needed in the form of
  - Structured English
  - Decision trees
  - Decision tables

1.2 CREATING DATA FLOW DIAGRAMS

Integrating Scenario Descriptions

- DFDs start with the use cases and requirements definition
- Generally, the DFDs integrate the use cases
- Names of use cases become processes
- Inputs and outputs become data flows
- “Small” data inputs and outputs are combined into a single flow

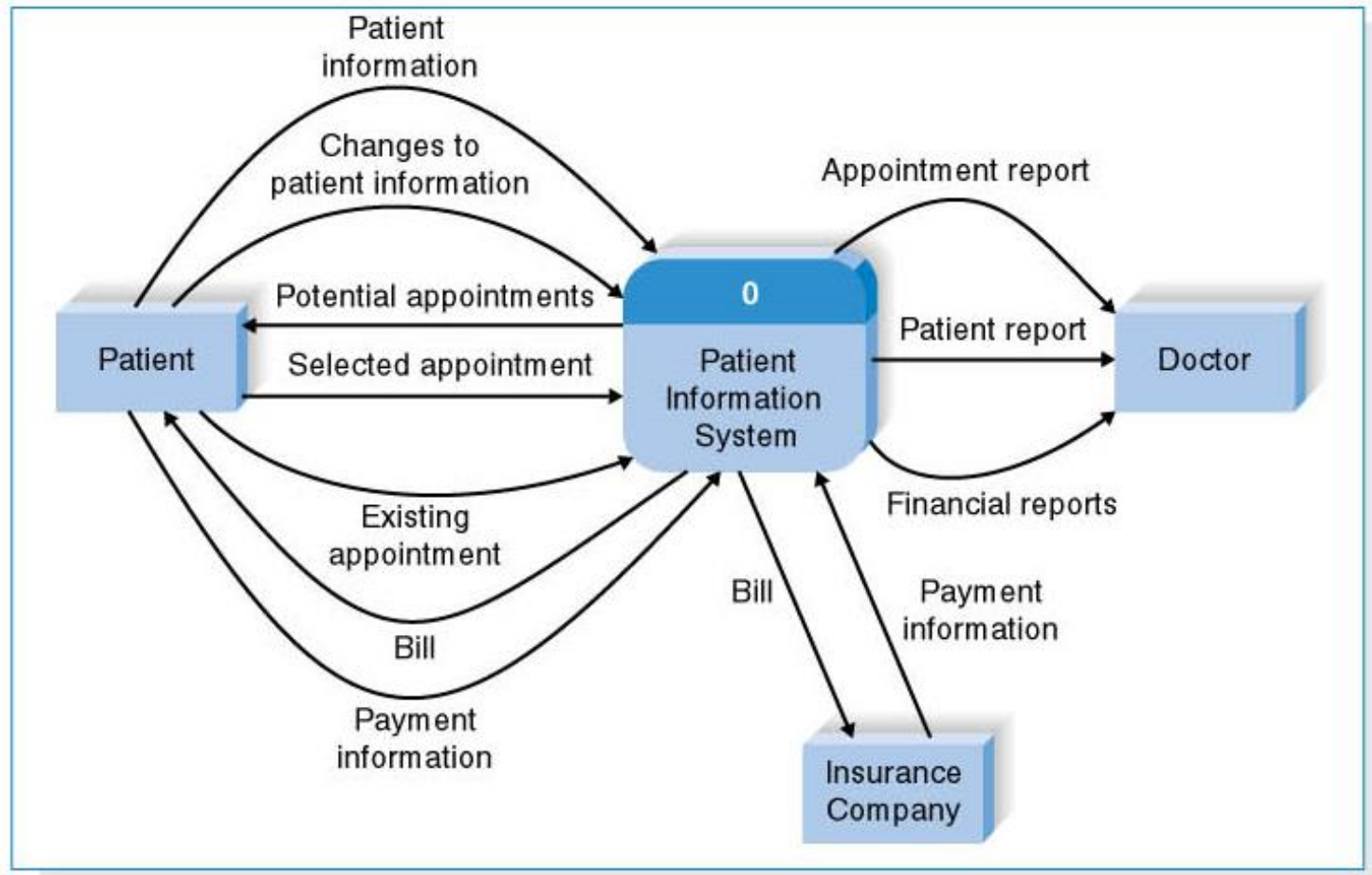
Steps in Building DFDs

1. Build the context diagram
2. Create DFD fragments for each use case
3. Organize DFD fragments into level 0 diagram
4. Decompose level 0 processes into level 1 diagrams as needed; decompose level 1 processes into level 2 diagrams as needed; etc.
5. Validate DFDs with user to ensure completeness and correctness

Creating the Context Diagram

- Draw one process representing the entire system (process 0)
- Find all inputs and outputs listed at the top of the use cases that come from or go to external entities; draw as data flows
- Draw in external entities as the source or destination of the data flows

A Context Diagram Example



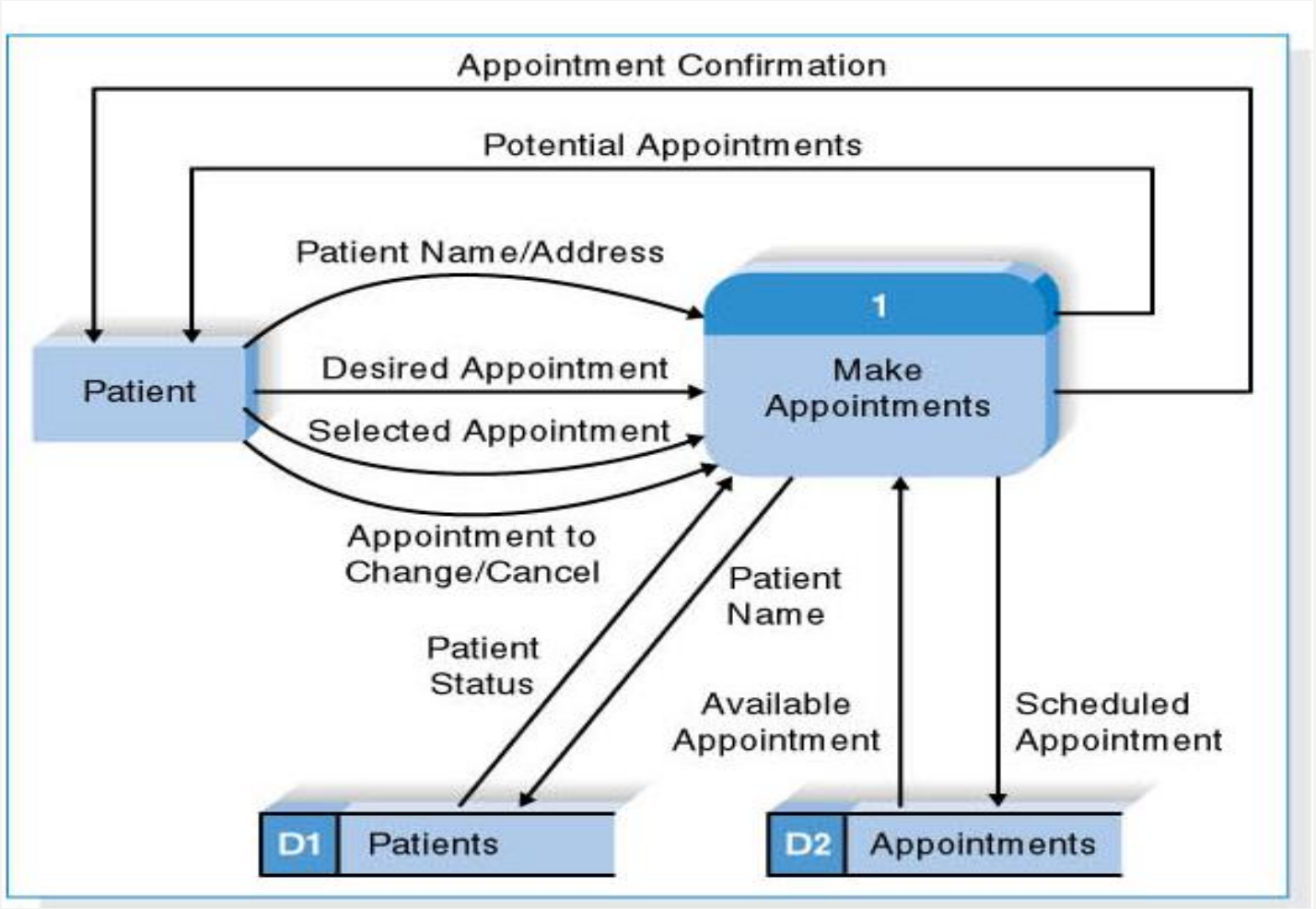
Creating DFD Fragments

- **Each use case is converted into one DFD fragment**
- Number the process the same as the use case number
- Change process name into verb phrase
- Design the processes from the viewpoint of the organization running the system

Creating DFD Fragments

- Add data flows to show use of data stores as sources and destinations of data
- Layouts typically place
  - processes in the center
  - inputs from the left
  - outputs to the right
  - stores beneath the processes

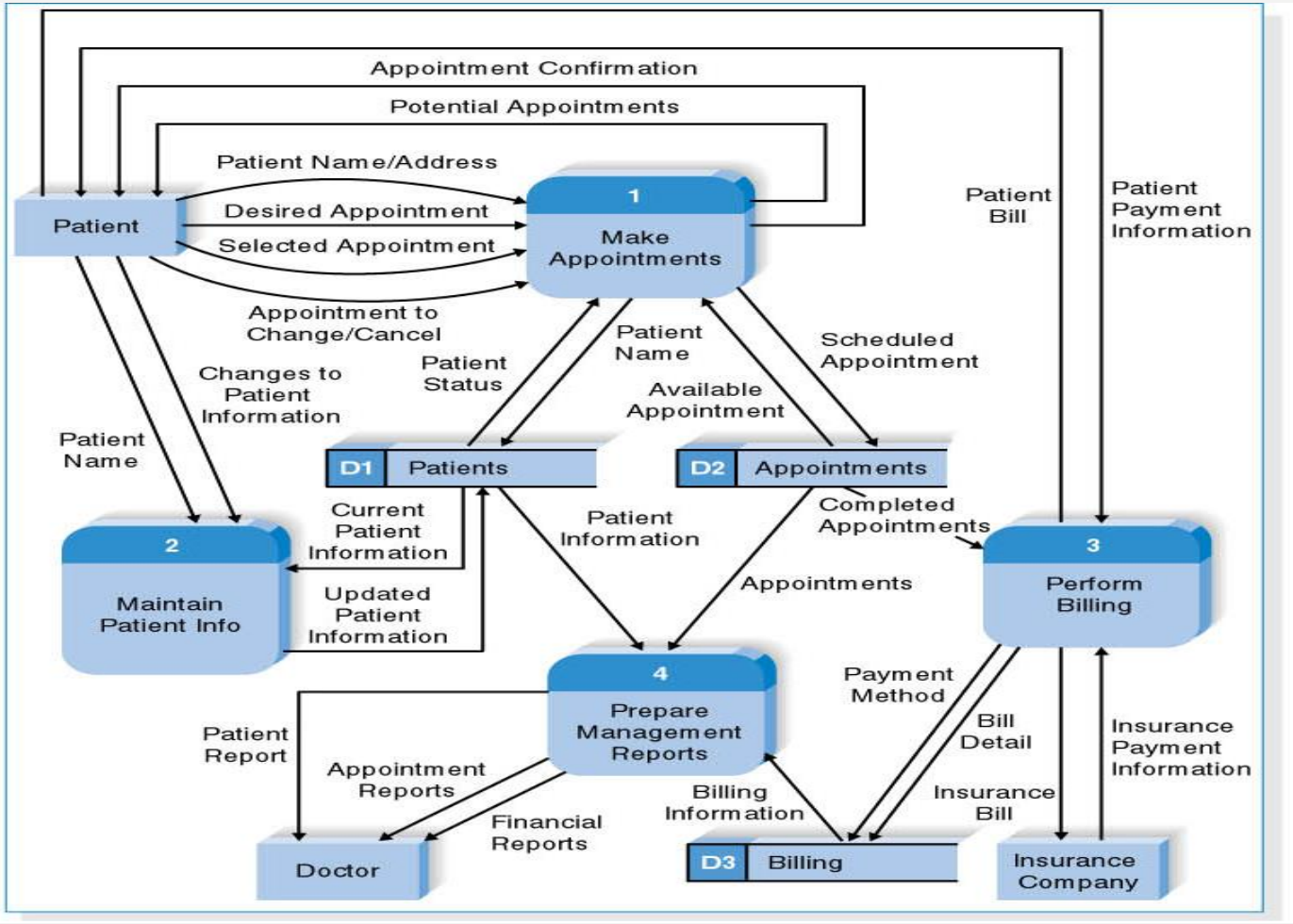
A DFD Fragment Example



Creating the Level 0 Diagram

- Combine the set of DFD fragments into one diagram
- Generally move from top to bottom, left to right
- Minimize crossed lines
- Iterate as needed
  - **DFDs are often drawn many times before being finished, even with very experienced systems analysts**

A Level 0 DFD Example



Creating Level 1 Diagrams (and Below)

- Each use case is turned into its own DFD
- Take the steps listed on the use case and depict each as a process on the level 1 DFD
- Inputs and outputs listed on use case become data flows on DFD
- Include sources and destinations of data flows to processes and stores within the DFD
- May also include external entities for clarity

Creating Level 1 Diagrams (and Below)

- When to stop decomposing DFDs?
  - Depends on the complexity of the system or business process being modeled.
  - In general, you decompose a process into a lower-level DFD whenever that process is sufficiently complex that additional decomposition can help explain the process.
  - Most experts believe that there should be at least three, and no more than seven to nine, processes on every DFD.

Validating the DFD

1. Syntax errors – diagram follows the rules
   - Assure correct DFD structure

For each DFD:

check each process for:

A unique name: action verb phrase; number; description
At least one input data flow
At least one output data flow
Output data flow names usually different than input data flow names
Between 3 and 7 processes per DFD

For each DFD:

Check each **data flow** for:

- A unique name: noun; description
- Connects to at least one process
- Shown in only one direction (no two-headed arrows)
- A minimum number of crossed lines

Check each **data store** for:

- A unique name: noun; description
- At least one input data flow
- At least one output data flow

Check each **external entity** for:

- A unique name: noun; description
- At least one input or output data flow

Across DFDs:

Context Diagram:
Every set of DFDs must have one Context Diagram

Viewpoint:
There is a consistent viewpoint for the entire set of DFDs

Decomposition:
Every process is wholly and complete described by the processes on its children DFDs

Balance:

- Every data flow, data store, and external entity on a higher level
- DFD is shown on the lower level DFD that decomposes it
- No data stores or data flows appear on lower-lever DFDs that do not appear on their parent DFD

Validating the DFD

- Semantics errors – diagram conveys correct meaning
  - Assure accuracy of DFD relative to actual/desired business processes
- To verify correct representation, use
  - User validation/walkthroughs
  - Role-play processes
- Examine lowest level DFDs to ensure consistent decomposition
- Examine names carefully to ensure consistent use of terms

Illegal Process

1. Spontaneous generation (miracles)

   Process with no inputs

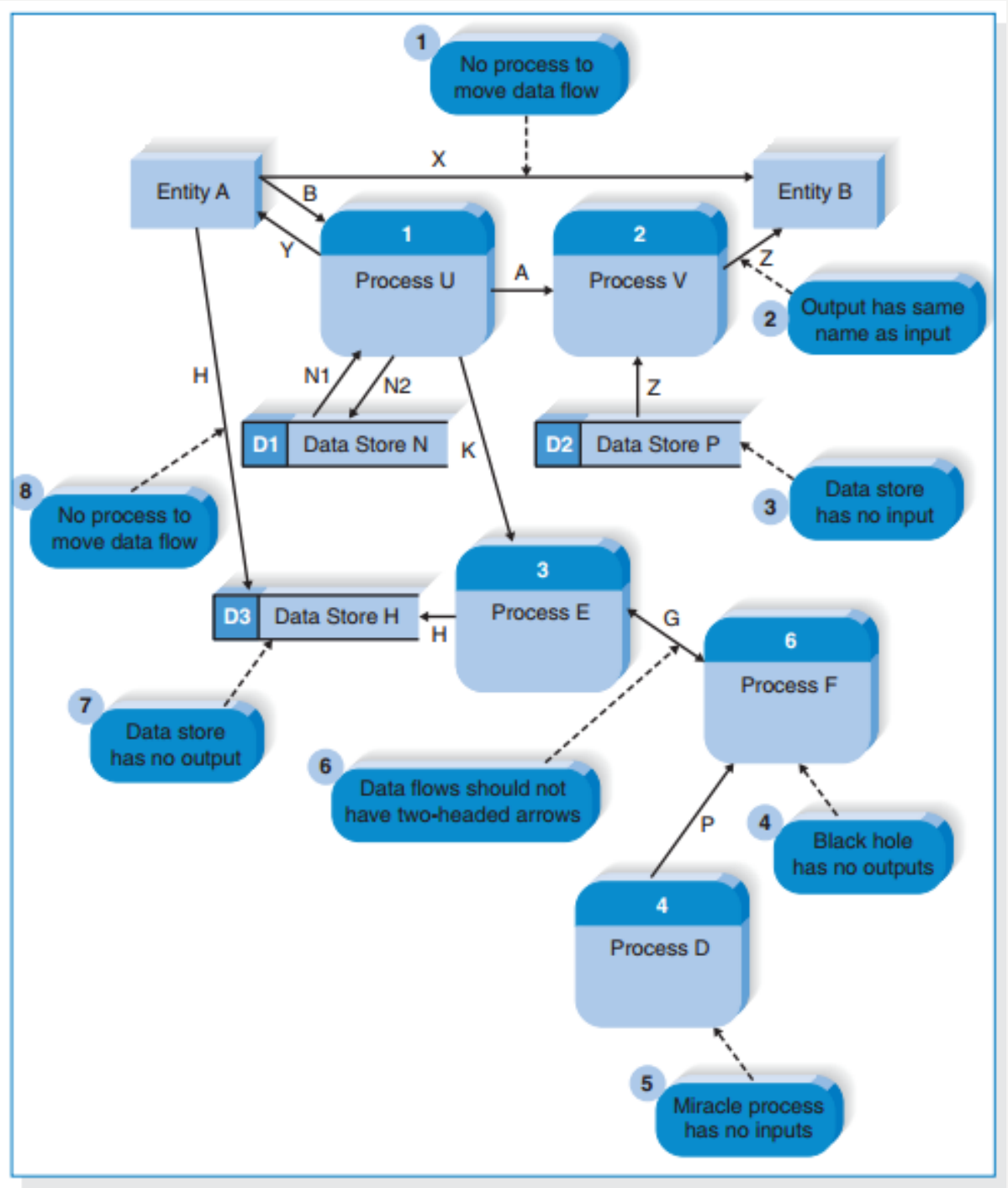
2. Black Hole

   Process with no outputs

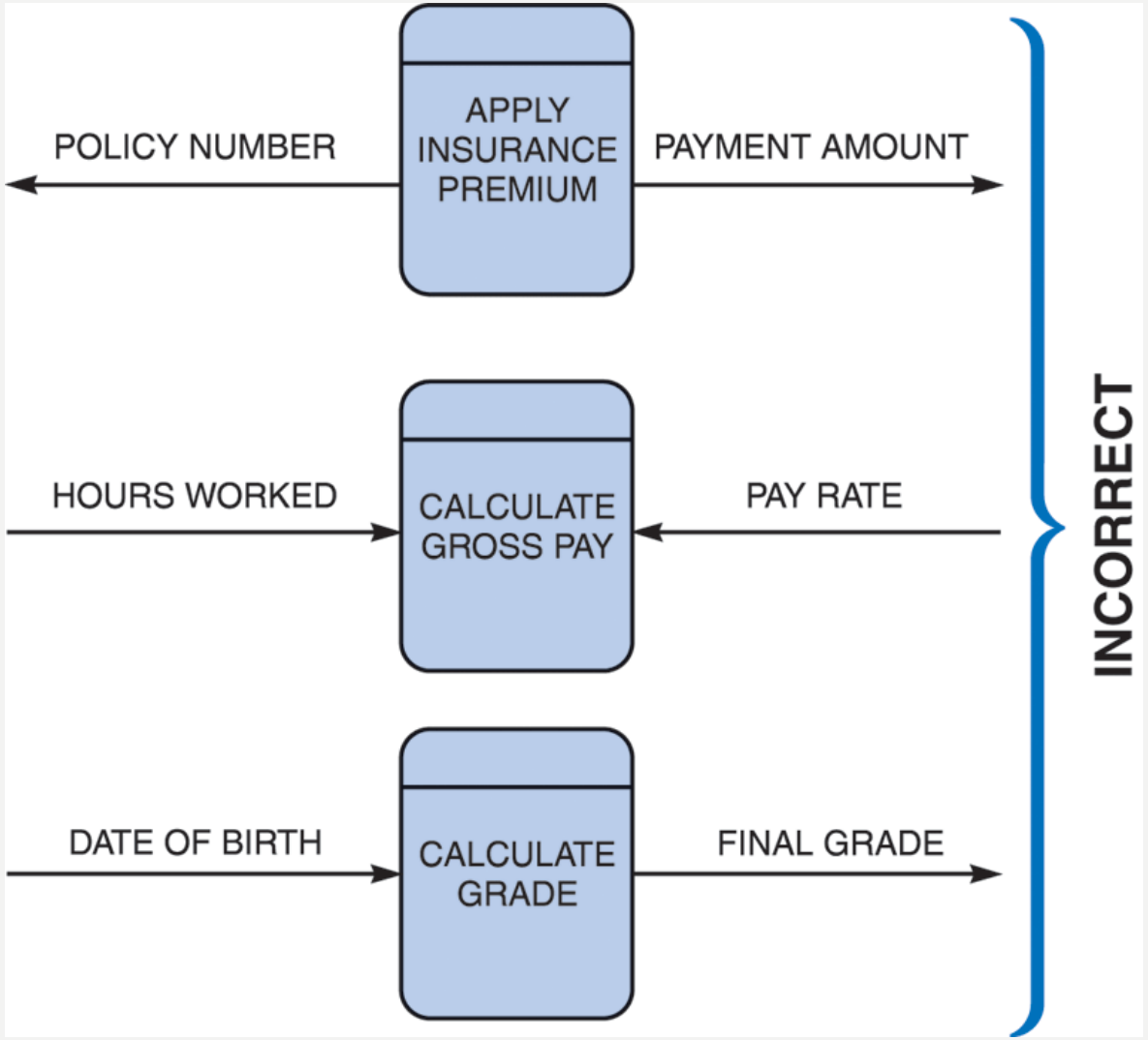
3. Gray Hole

   The inputs is insufficient to generate the output

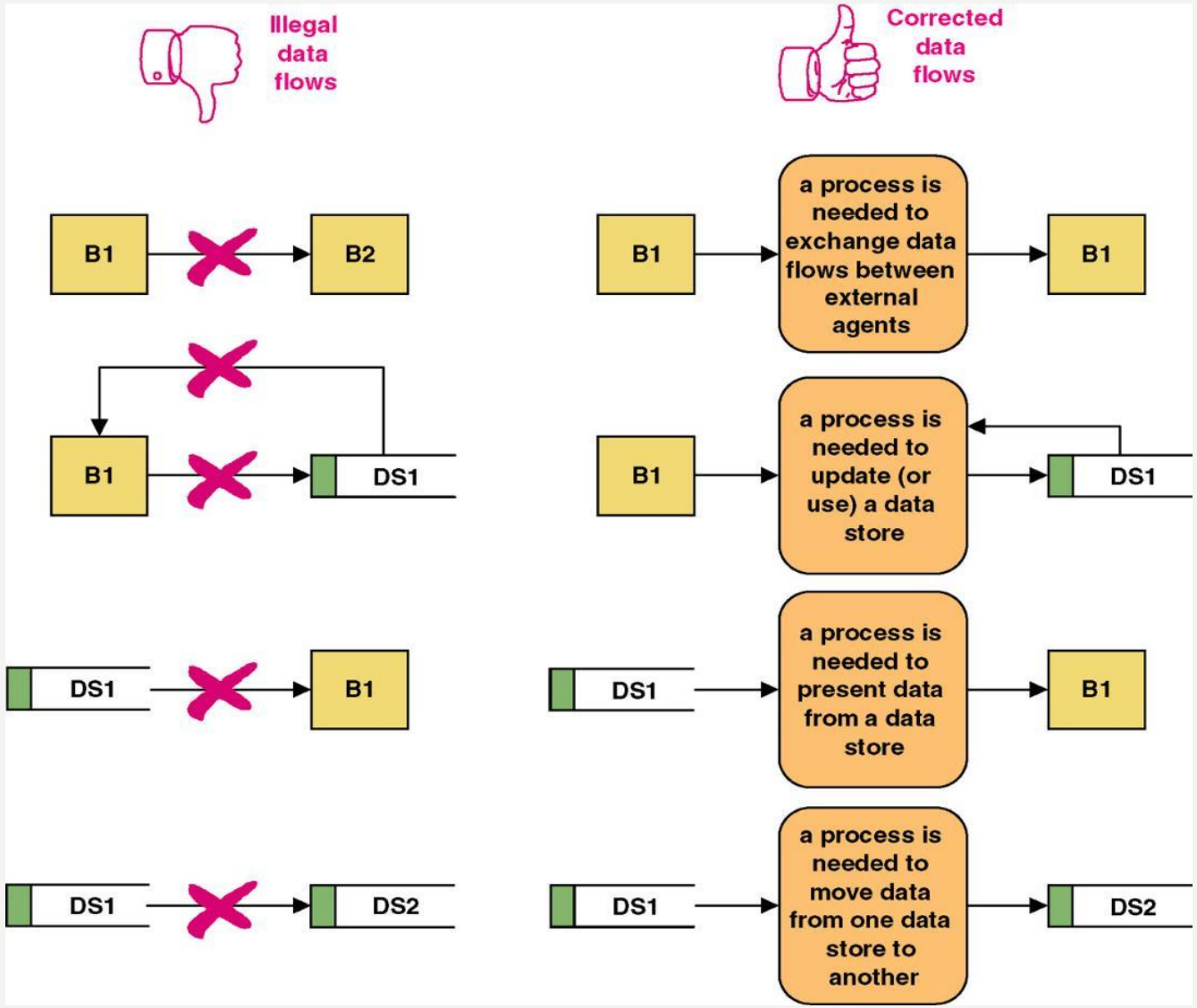
Some Common Error



Some Common Error

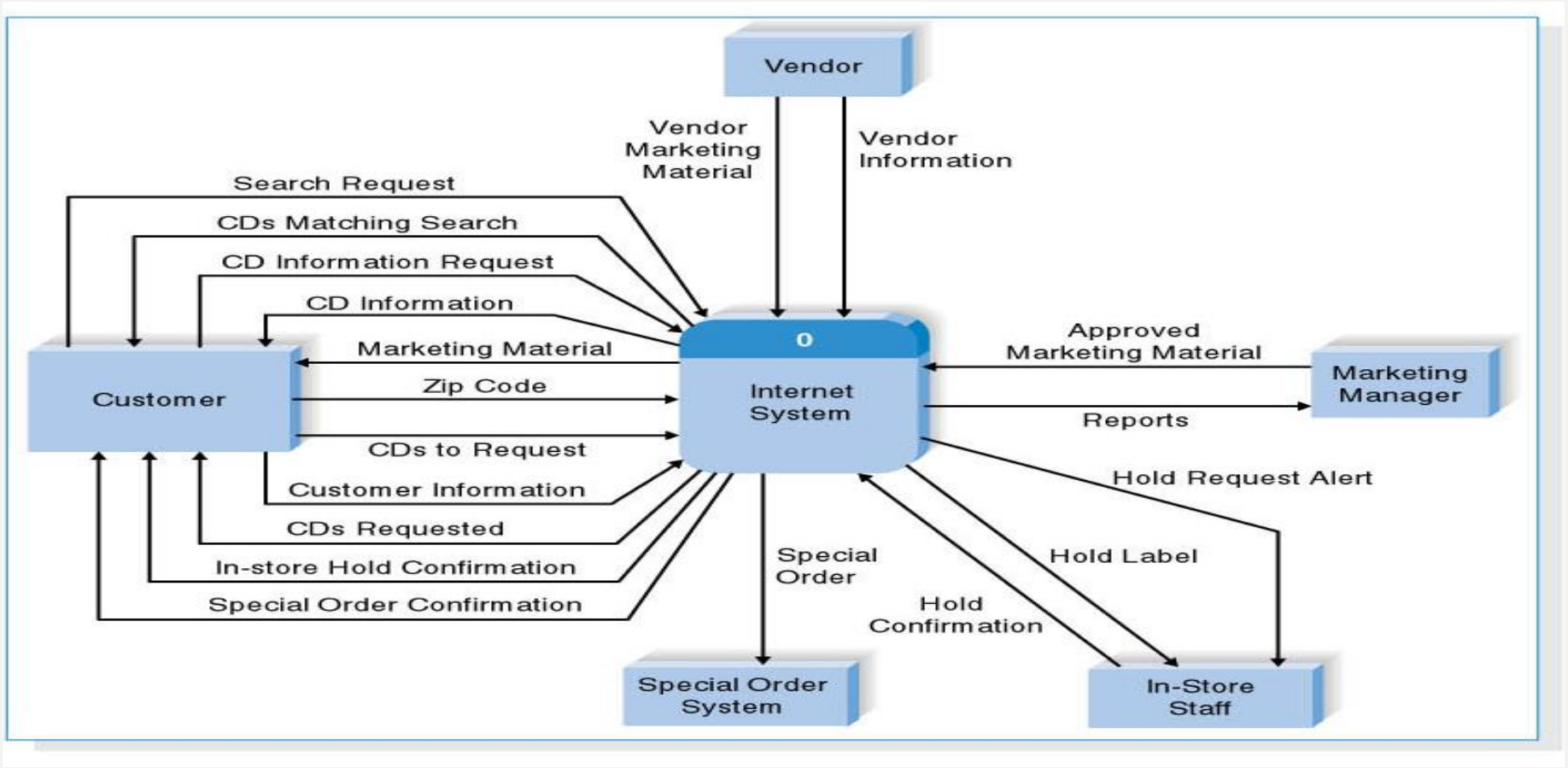


Illegal Data Flows

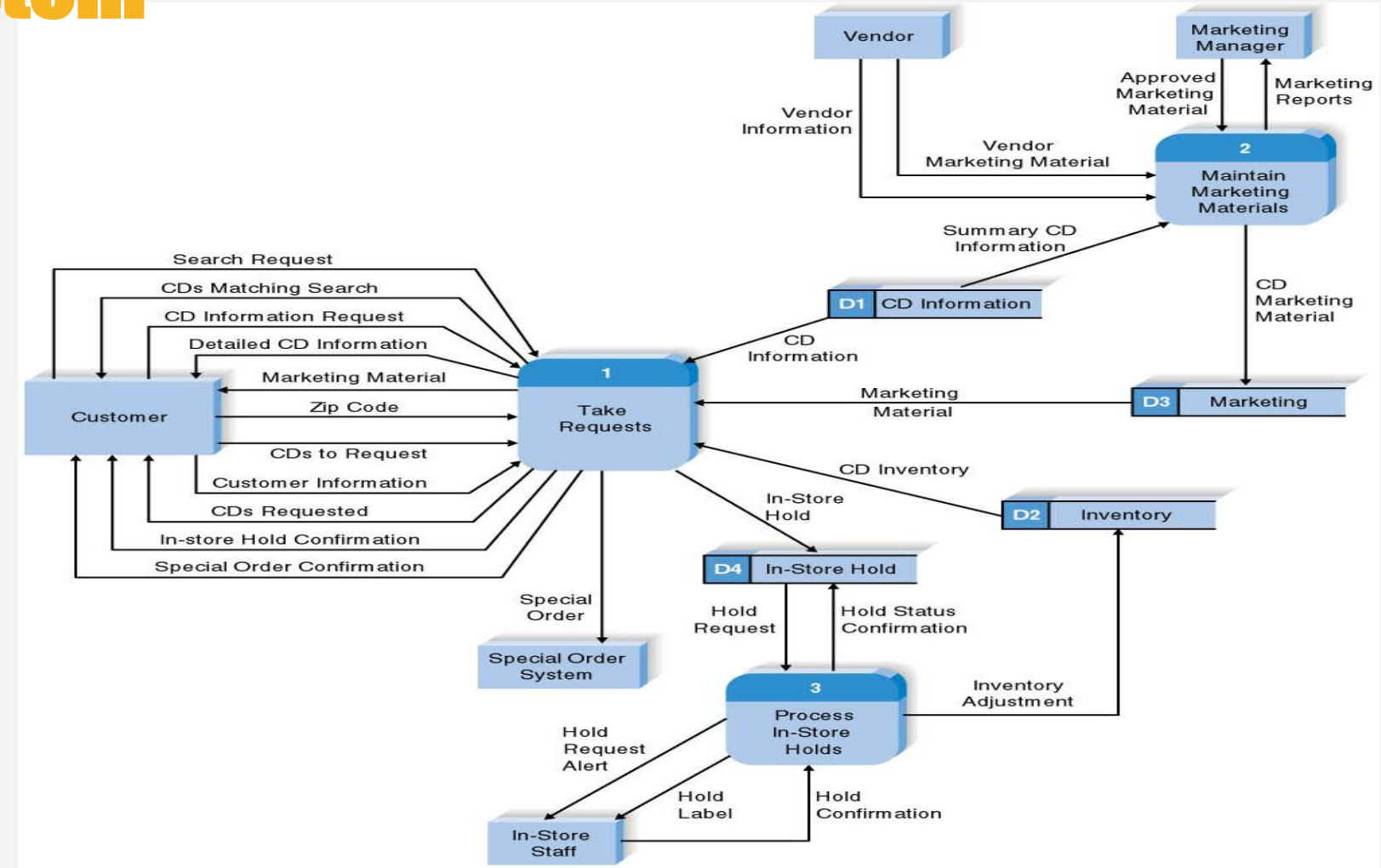


A Quick Review of Decomposition for CD Selections

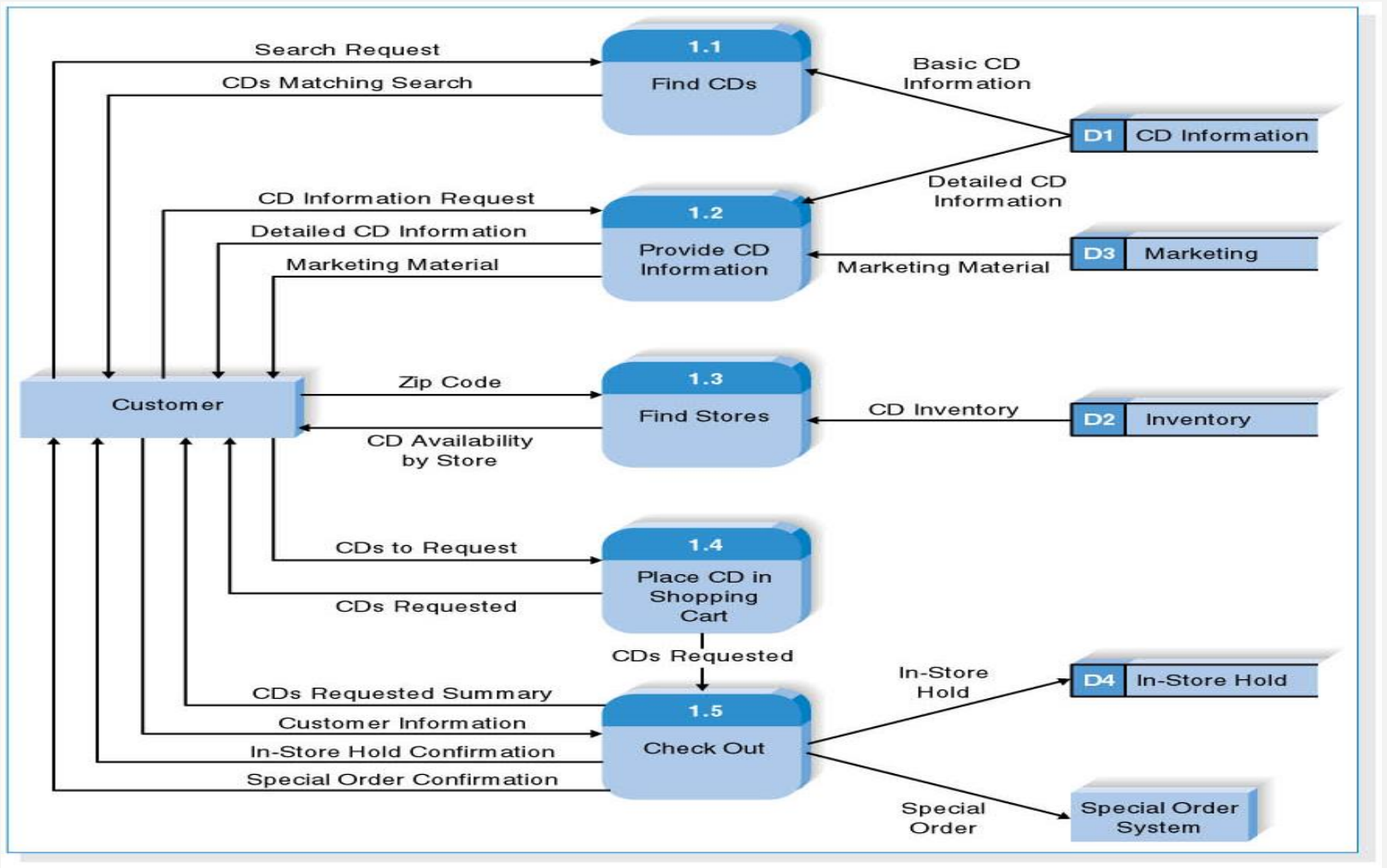
Context Diagram for CD Selections Internet Sales System



Level 0 DFD for CD Selections Internet System



Level 1 DFD for CD Selections Process 1: Take Requests



Summary

- The Data Flow Diagram (DFD) is an essential tool for creating formal descriptions of business processes.
- Use cases record the input, transformation, and output of business processes and are the basis for process models.
- Eliciting use cases and modeling business processes are critically important skills for the systems analyst to master.

Test Yourself

1. Data flow diagrams show what a system does, not how it does it (T/F).

Test Yourself

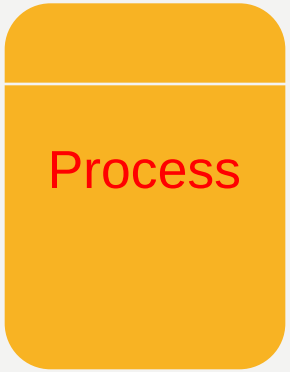
1. Data flow diagrams show what a system does, not how it does it (T/F). **True**

Test Yourself

2. The following symbols are from the _____ set. Name them:

Test Yourself

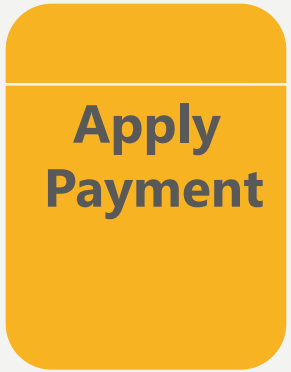
2. The following symbols are from the Gane and Sarson set. Name them:



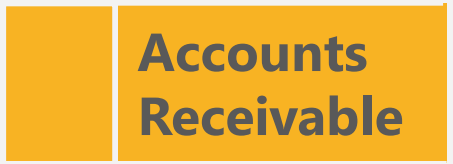
Test Yourself

2. Select the correct example below.

A)



B)



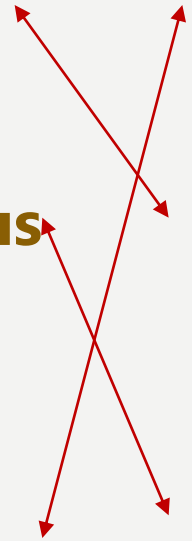
Test Yourself

4. Match the terms in the left column to the proper definitions in the right column.

1. Black Hole	**a. A process with at least 1 input and output, but the input is insufficient to generate the shown output.**
2. Spontaneous Generation Process	**b. A process that has no output**
3. Gray Hole	**c. Used to describe an unexplained generation of data or information.**

Test Yourself

4. Match the terms in the left column to the proper definitions in the right column.



1. Black Hole	**a. A process with at least 1 input and output, but the input is insufficient to generate the shown output.**
2. Spontaneous Generation Process	**b. A process that has no output**
3. Gray Hole	**c. Used to describe an unexplained generation of data or information.**

Your Turn!

Create a **context diagram** for an **online university registration system**.

The system should enable the **staff of each academic department** to **examine the courses offered** by their department, **add and remove courses**, and **change the information** about them (e.g., the maximum number of students permitted). It should permit **students** to **examine currently available courses**, **add and drop courses to and from their schedules, and examine the courses for which they are enrolled**. Academic department staff should be able to **print a variety of reports** about the courses and the students enrolled in them. The system should ensure that **no student takes too many courses** and that **students who have any unpaid fees are not permitted to register**. (Assume that a fees data store is maintained by the university's financial office, which the registration system accesses but does not change).

Your Turn!

Use Case Name: Search and browse tunes	ID: UC -1	Priority: High
Actor: Tune Shopper		
Description: This use case describes a tune shopper who searches and browses through tunes		
Trigger: Tune shopper arrives at Web site to search and browse through tunes Type: ☑ External ☐ Temporal		
Preconditions: Web site is available Tune database is on-line		

Normal Course:	Information for Steps:
1.0 Search and browse tunes and select tune to purchase	
1. System displays default home page or customized page	
2. Tune Shopper browses links on page or enters account username and password	← Username/password
3. Tune Shopper want to create an account: perform Create Account use case	
4. Tune Shopper enters search request	← Search criteria
5. System displays tune(s) matching search request	← Tunes matching search
6. Tune Shopper selects a tune and wants to hear a sample	← Tune samples → New Interest
7. Tune Shopper selects a tune to add to Favorites	→ New Favorites
8. Tune Shopper selects a tune to remove from Favorites	→ Modified Favorites
9. Tune Shopper selects a tune to buy by placing it in shopping cart	→ New Shopping Cart Entry
10. Tune Shopper selects a tune to remove from shopping cart	→ Modified Shopping Cart
Alternative courses:	
1.1 Tune Shopper is a return visitor (branch at step 1)	
1. System displays page customized for the return visitor using Interests from prior visits	← Interests database
1.2 Tune Shopper has created an account (branch at step 2)	← Favorites database
1. System displays welcome message to account holder	← Targeted Promotions database
2. Page is customized for the account holder using Favorites List and Targeted Promotions	

Postconditions:
1. One or more tunes are added to shopper Interests
2. Account holder favorites list may be modified
3. Shopping cart contents may be modified

Exceptions:
E1: Account is not valid (occurs at step 2)
1. System displays message that username/password is not valid.
2. System asks Tune Shopper to re-enter username/password or contact customer service for help.

E2: Search request returns no results (occurs at step 3)
1. System displays message that no results were found for that search
2. System asks Tune Shopper to try another search

Summary

Inputs	Source	Outputs	Destination
Username/password	Tune Shopper	New Interest	Interests database
Search criteria	Tune Shopper	New Favorites	Favorites database
Tunes matching search	Tunes database	Modified Favorites	Favorites database
Tune samples	Tune Samples database	New Shopping Cart Entry	Shopping Cart database

Your Turn!

Based on use-case description given in the previous page:

1. Create **a DFD fragment** of “Search and browse tune” use-case, as part of level 0 DFD.
2. Create **a level 1 DFD** of “Search and browse tune” use-case

1.3 DATA MODELLING

Data Model

- **Analyst need to understand the information that is used and created by the business system.**
- **Data model**
  - A formal way of representing the data that are used and created by a business system
  - Shows the people, places and things about which data is captured and the relationships among them.
- **Logical data model**
  - shows the organization of data without indicating how it is stored, created, or manipulated
  - Focus on matching the diagram to the real business requirements of the system

Data Model

- **Physical data model**
  - shows how the data will actually be stored in databases or files.
- **Normalization** is the process analysts use to validate data models.
- Data models should **balance** with process models

What Is an ERD?

- A picture showing the information created, stored, and used by a business system.
- Entities generally represent similar kinds of information
- Lines drawn between entities show relationships among the data
- High level business rules are also shown

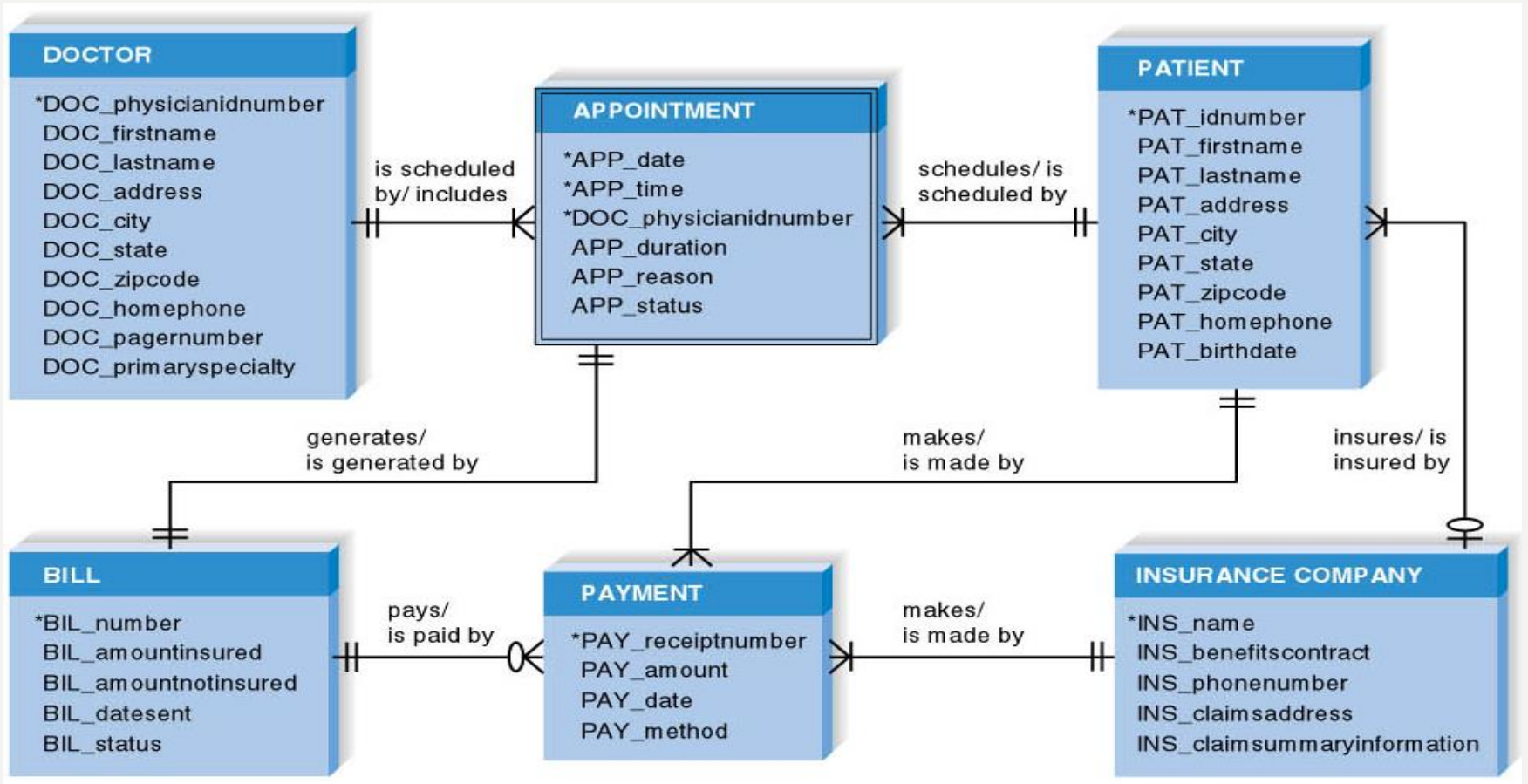
Using the ERD to Show Business Rules

- Business rules are constraints that are followed when the system is in operation.
- ERD symbols can show when one instance of an entity must exist for an instance of another to exist (no order)
  - **A doctor must exist before appointments for the doctor can be made**

Using the ERD to Show Business Rules

- ERD symbols can show when one instance of an entity can be related to only one or many instances of another entity
  - **One doctor can have many patients; each patient may have only one primary doctor**
- ERD symbols show when the existence of an entity instance is optional for a related entity instance
  - **A patient may or may not have insurance coverage**

An ERD Example



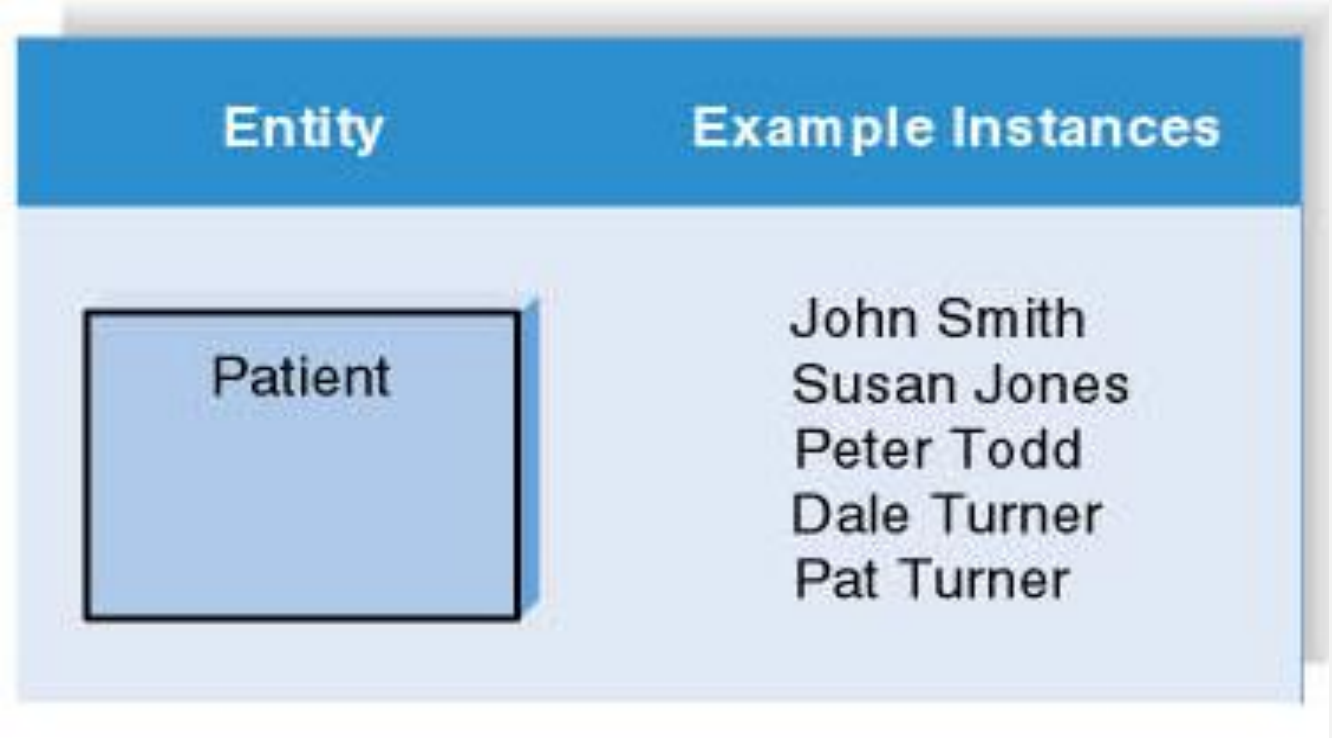
ERD Elements

	IDEF1X	Chen	Crow's Foot
An ENTITY: ✓ Is a person, place, or thing ✓ Has a singular name spelled in all capital letters ✓ Has an identifier ✓ Should contain more than one instance of data	ENTITY-NAME Identifier	ENTITY-NAME	ENTITY-NAME *Identifier
An ATTRIBUTE: ✓ Is a property of an entity ✓ Should be used by at least one business process ✓ Is broken down to its most useful level of detail	ENTITY-NAME Attribute-name Attribute-name Attribute-name	Attribute-name	ENTITY-NAME Attribute-name Attribute-name Attribute-name
A RELATIONSHIP: ✓ Shows the association between two entities ✓ Has a parent entity and a child entity ✓ Is described with a verb phrase ✓ Has cardinality (1 : 1, 1 : N, or M : N) ✓ Has modality (null, not null) ✓ Is dependent or independent	Relationship-name	Relationship-name	Relationship-name

Entity

- A person, place, event, or thing about which data is collected
- Must be multiple occurrences to be an entity
  - Example: If a firm has only one warehouse, the warehouse is not an entity. However, if the firm has several warehouses, the warehouse could be an entity if the firm wants to store data about each warehouse instance.
  - There is no need to capture data in the system about something having just a single instance

Entities and Instances



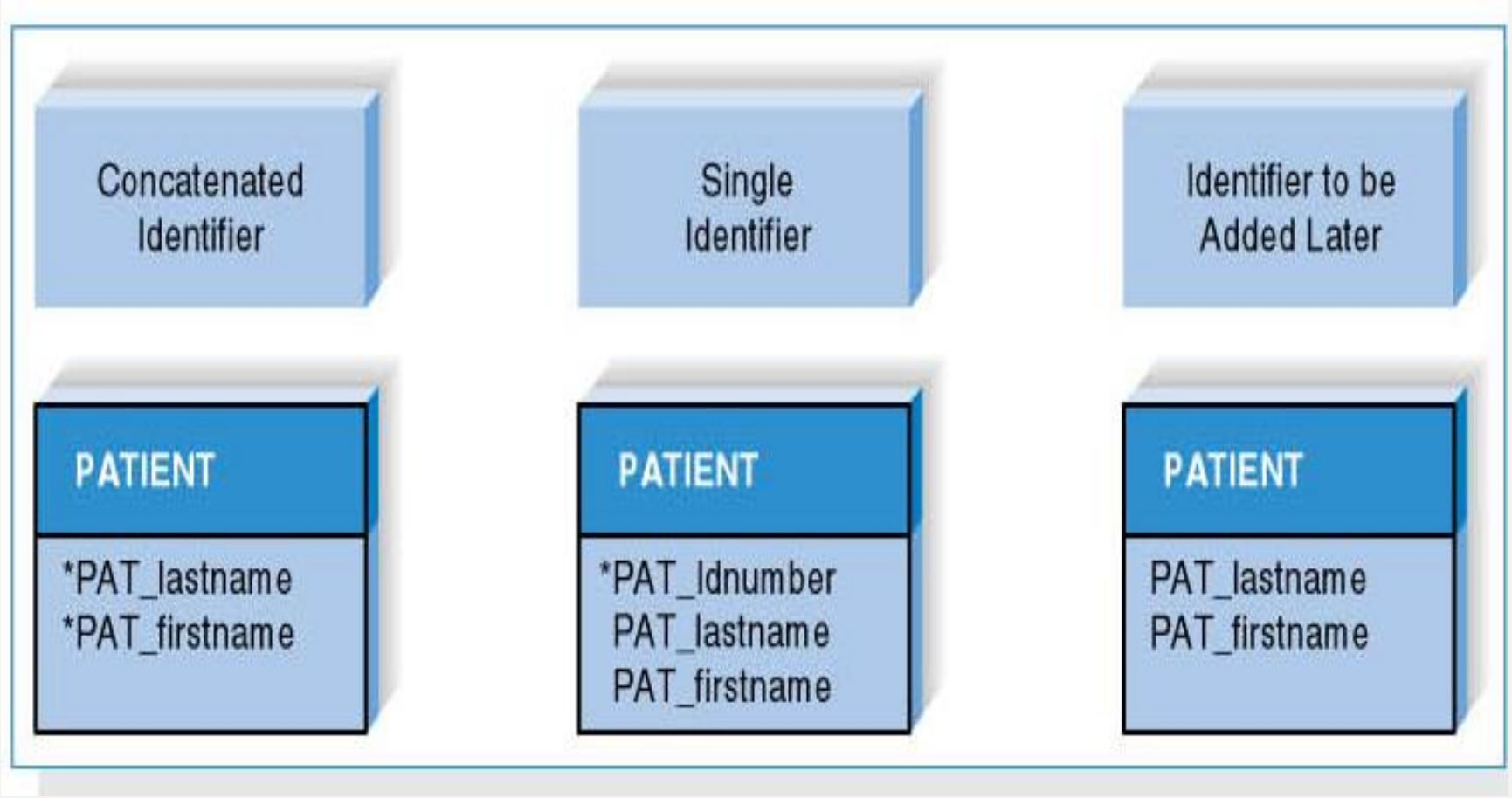
Attributes

- Information captured about an entity
- Only those used by the organization should be included in the model
- Attribute names are nouns
- Sometimes entity name is added at the beginning of the attribute name for clarity

Identifiers

- One or more attributes can serve as the entity **identifier**, uniquely identifying each entity instance
- **Concatenated identifier** consists of several attributes
- An identifier may be ‘artificial,’ such as creating an ID number
- Identifiers may not be developed until the Design Phase

Choices for Identifiers



Relationships

- Associations between entities
- The first entity in the relationship is the ***parent*** entity; the second entity in the relationship is the ***child*** entity
- Relationships should have active verb names
- Relationships go in both directions
  - Verb “schedule” between Patient and Appointment means:
    - A patient schedules an appointment
    - An appointment is scheduled by a patient

Cardinality

- **Cardinality**
  - refers to the number of times instances in one entity can be related to instances in another entity
    - One instance in an entity refers to one and only one instance in the related entity (1:1)
    - One instance in an entity refers to one or more instances in the related entity (1:N)
    - One or more instances in an entity refer to one or more instances in the related entity (M:N)

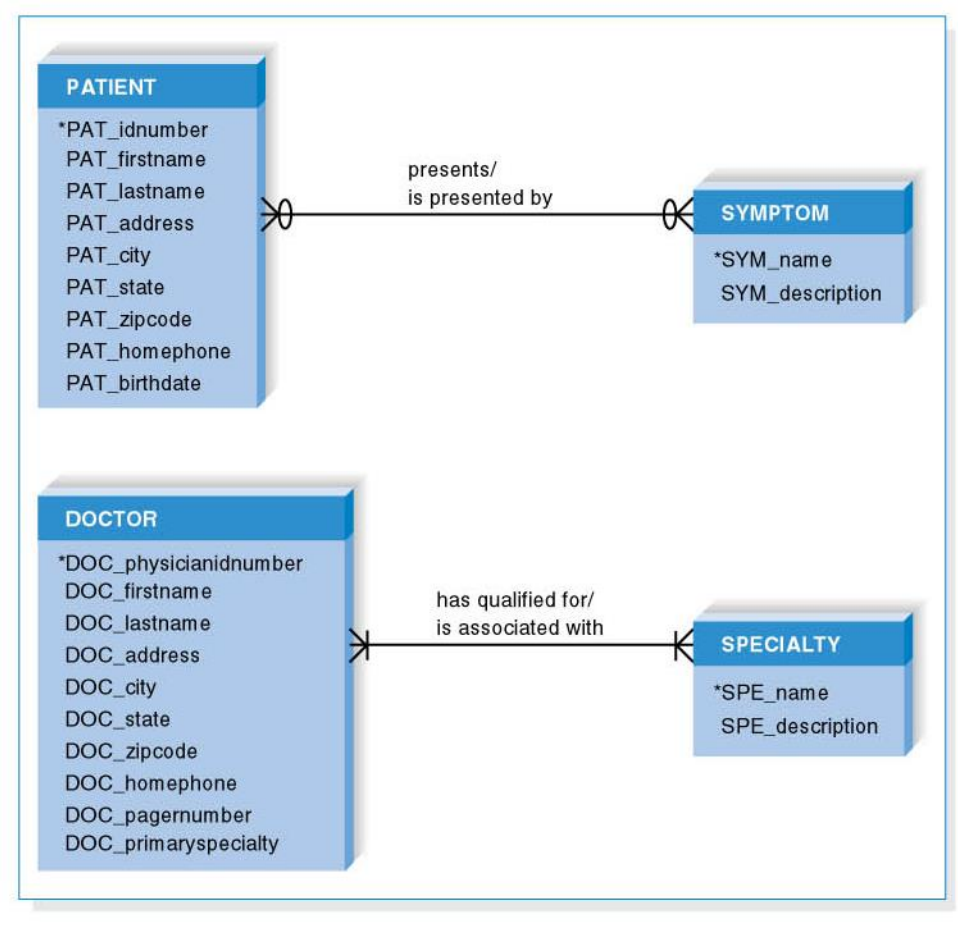
Modality

- ***Modality***
  - Refers to whether or not an instance of a child entity can exist without a related instance in the parent entity
    - **Not Null** means that an instance in the related entity must exist for an instance in another entity to be valid
    - **Null** means that no instance in the related entity is necessary for an instance in another entity to be valid

Modality

- Can you have an appointment without a doctor ?
- Can you have a bill without an appointment ?
  - The modality is not null
- We can have a patient in our system who does not have an insurance company → the modality is null
- A patient can exist in the system without presenting symptoms
- A doctor must be qualified at least one specialty

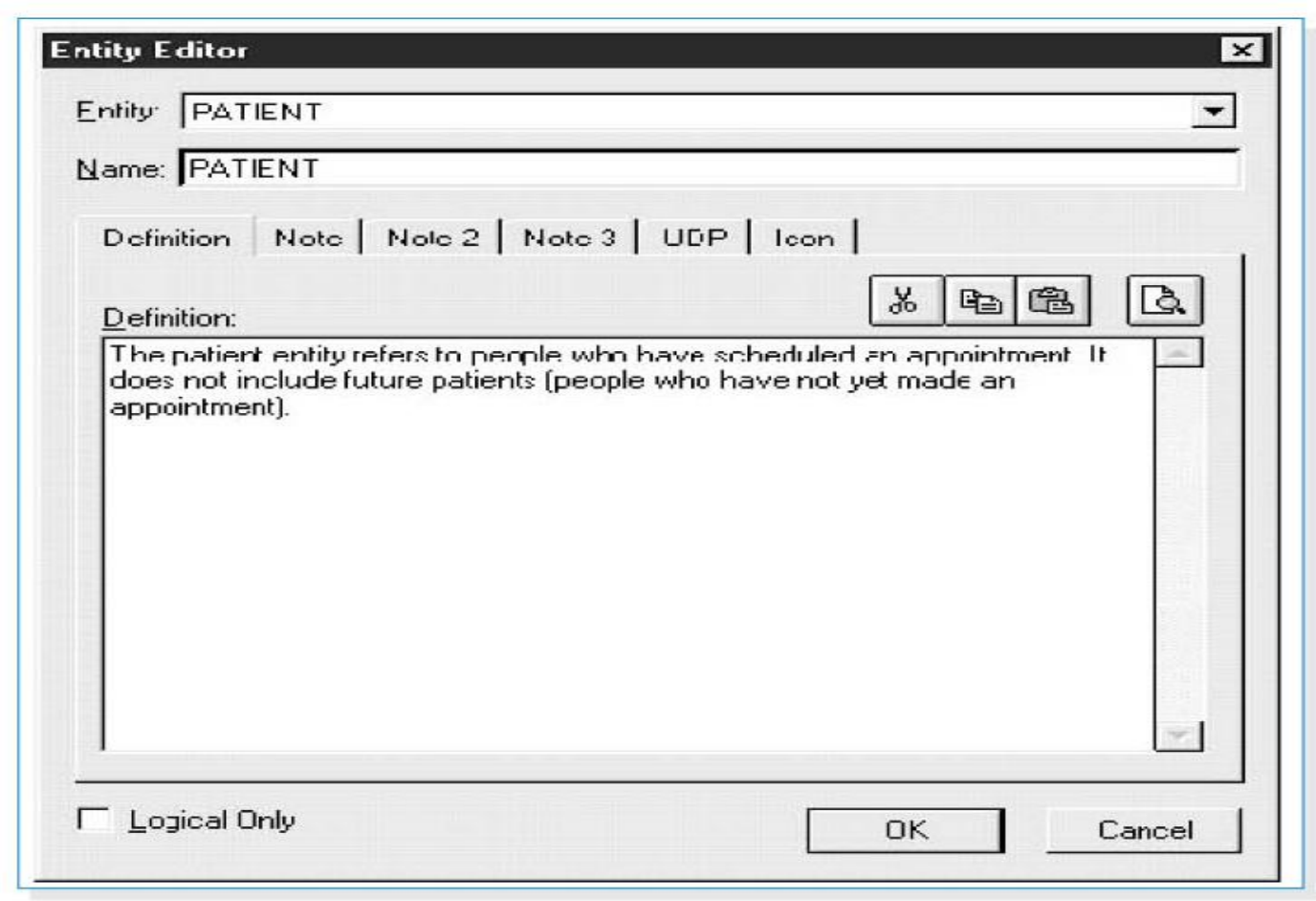
M : N Relationships



The Data Dictionary and Metadata

- **Metadata** is information stored about components of the data model
- Metadata is stored in the data dictionary so it can be shared by developers and users throughout the SDLC
- A complete, shareable data dictionary helps improve the quality of the system under development

Data Dictionary Entry for the Patient Entity (Shown Using Erwin)



1.4 BUILDING AN ENTITY RELATIONSHIP DIAGRAM

ERD Basics

- Drawing the ERD is an iterative process of trial and revision
- ERDs can become quite complex
  - There are systems that have ERDs containing hundreds or thousands of entities

Steps in Building ERDs

- Identify the entities
- Add attributes and assign identifiers
- Identify relationships

Identify the Entities

- Identify major categories of information
  - If available, check the process models for data stores, external entities, and data flows
  - Check the major inputs and outputs from the use cases
- Verify that there is more than one instance of the entity that occurs in the system

Add Attributes and Assign Identifiers

- Identify attributes of the entity that are relevant to the system under development
  - Check the process model repository entries for details on data flows and data stores
  - Check the data requirements of the requirements definition
  - Interview knowledgeable users
  - Perform document analysis on existing forms and reports
- Select the entity's identifier

Identify Relationships

- Start with an entity and identify all entities with which it shares relationships
- Describe the relationship with the appropriate verb phrase
- Determine the cardinality and modality by discussing the business rules with knowledgeable users

ERD Building Tips

- Data stores of the DFD should correspond to entities
- Only include entities with more than one instance of information
- Don’t include entities associated with implementation of the system (they will be added later)

Advanced Syntax (1)

- **Independent Entity (Strong Entity)**
  - Can exist without the help of another entity
  - Identifiers created from the entity's own attributes
  - Attributes from other entities are not needed to uniquely identify instances of these entities
  - Include an independent child entity → non-identifying relationship

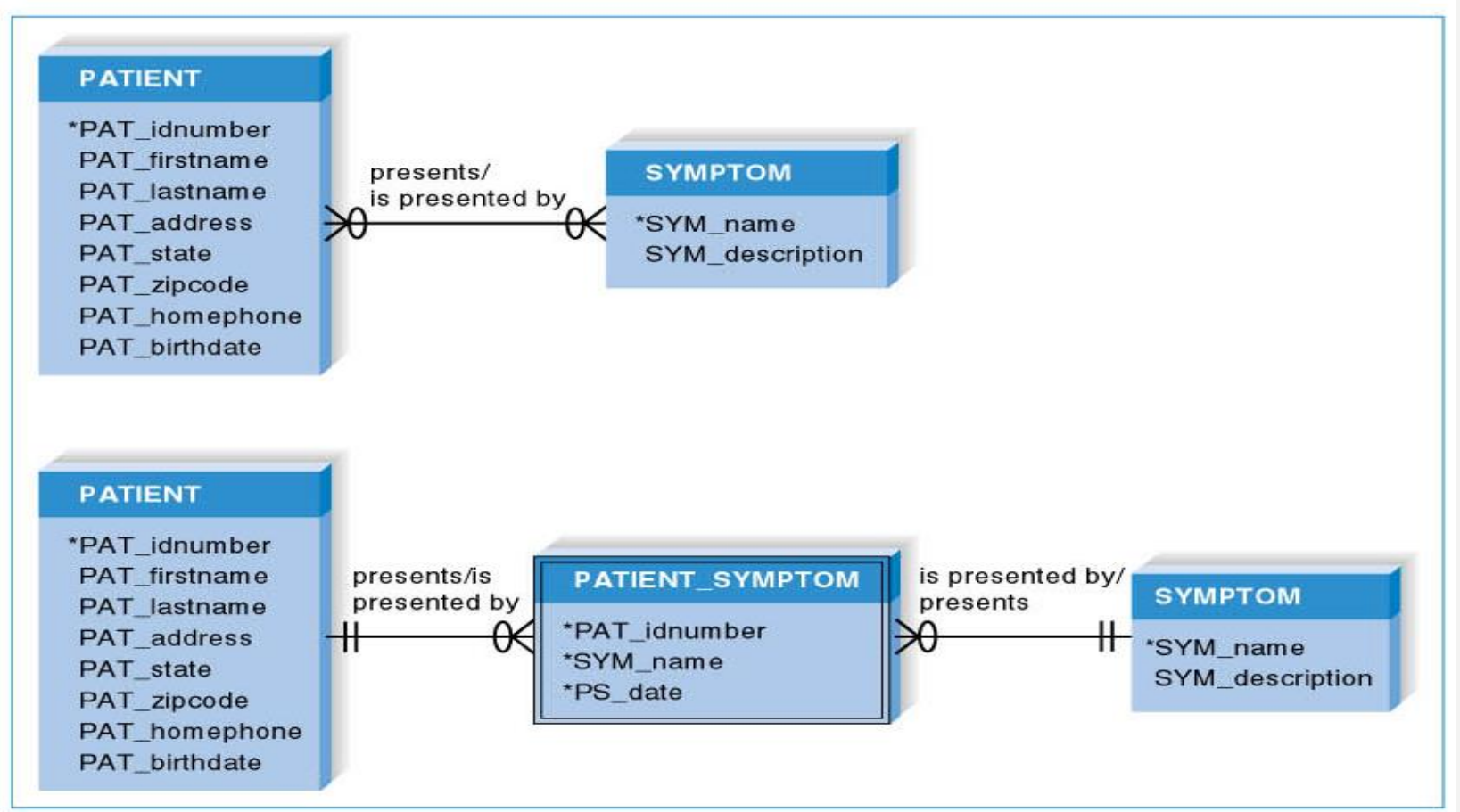
Advanced Syntax (2)

- **Dependent Entity (Weak Entity)**
  - Relationships when a child entity does require attributes from the parent entity to uniquely identify an instance ・ at least one attribute
  - Have a dependent child entity ・ identifying relationships
  - Appointment

Advanced Syntax (3)

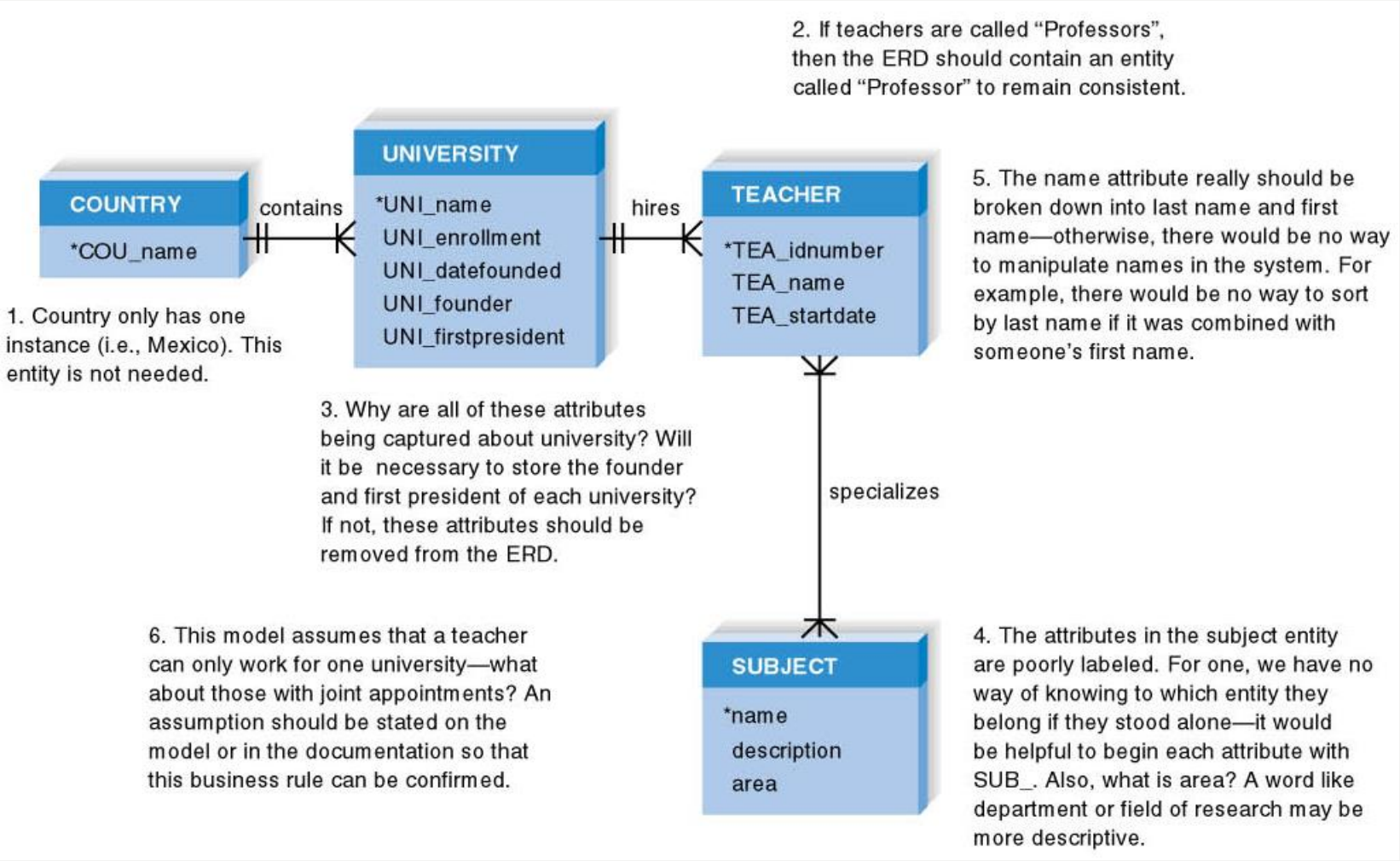
- **Intersection Entity**
  - Exists in order to capture some information about the relationship that exists between two other entities. Typically, intersection entities are added to a data model to store information about two entities sharing an M : N relationship.

Advanced Syntax – Resolving an M : N Relationship



1.5 VALIDATING AN ERD

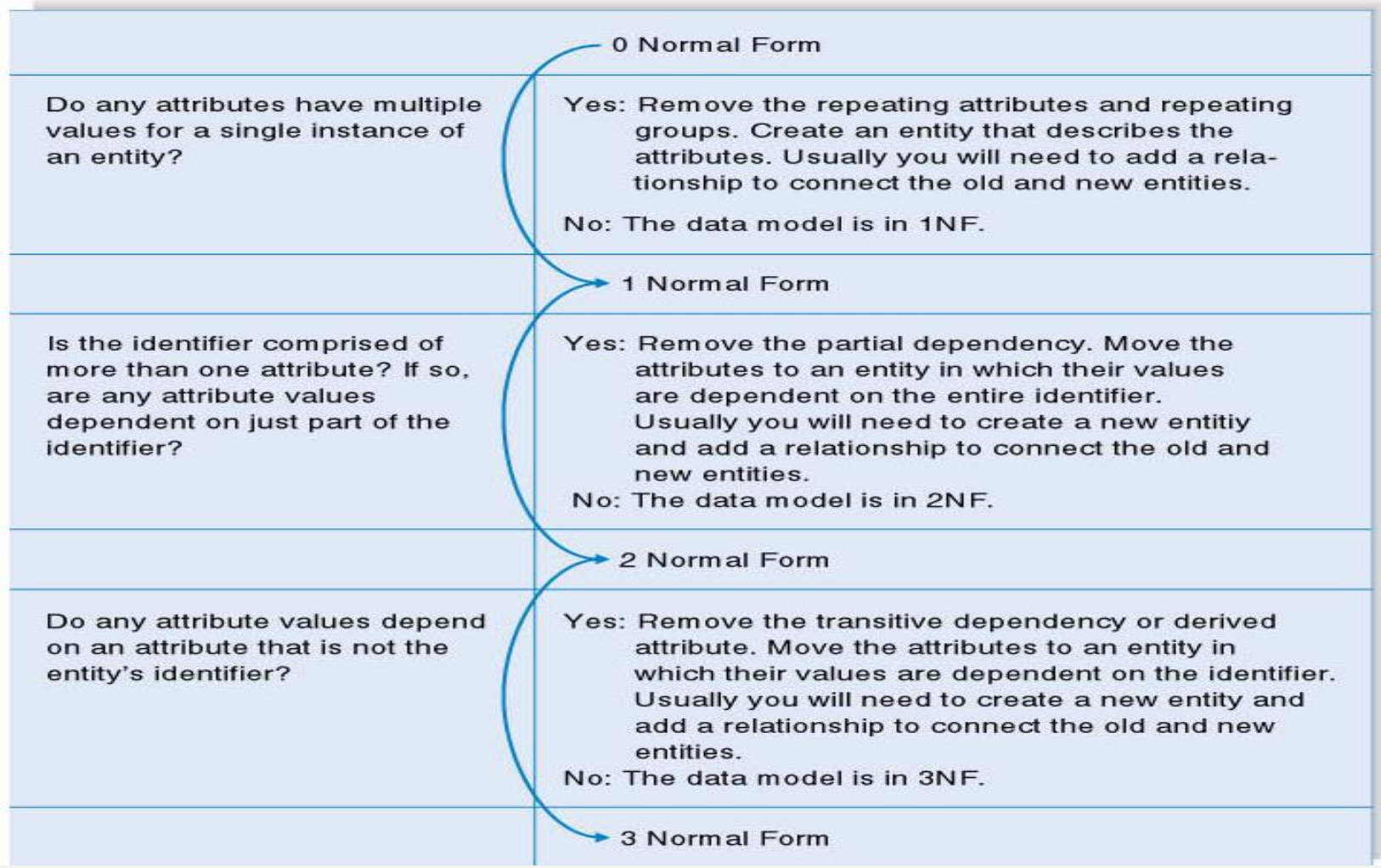
Design Modeling Guidelines Summary



Normalization

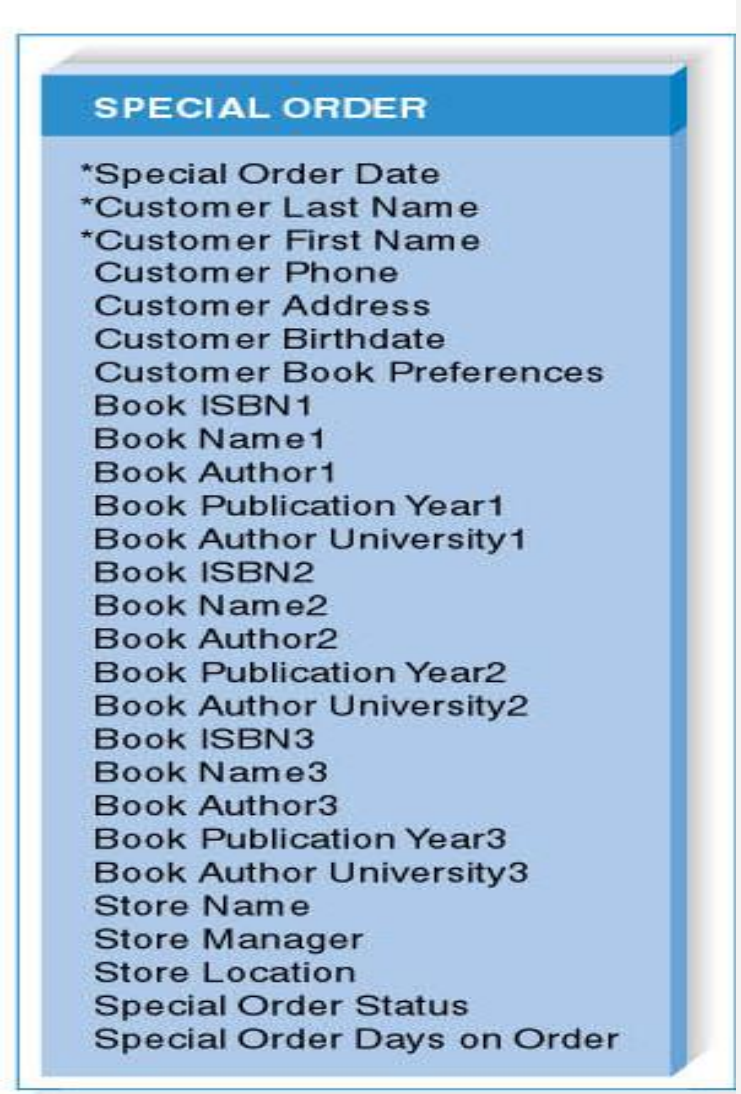
- Technique used to validate data models
- Series of rules applied to logical data model to improve its organization
- Three normalization rules are common

Normalization Steps



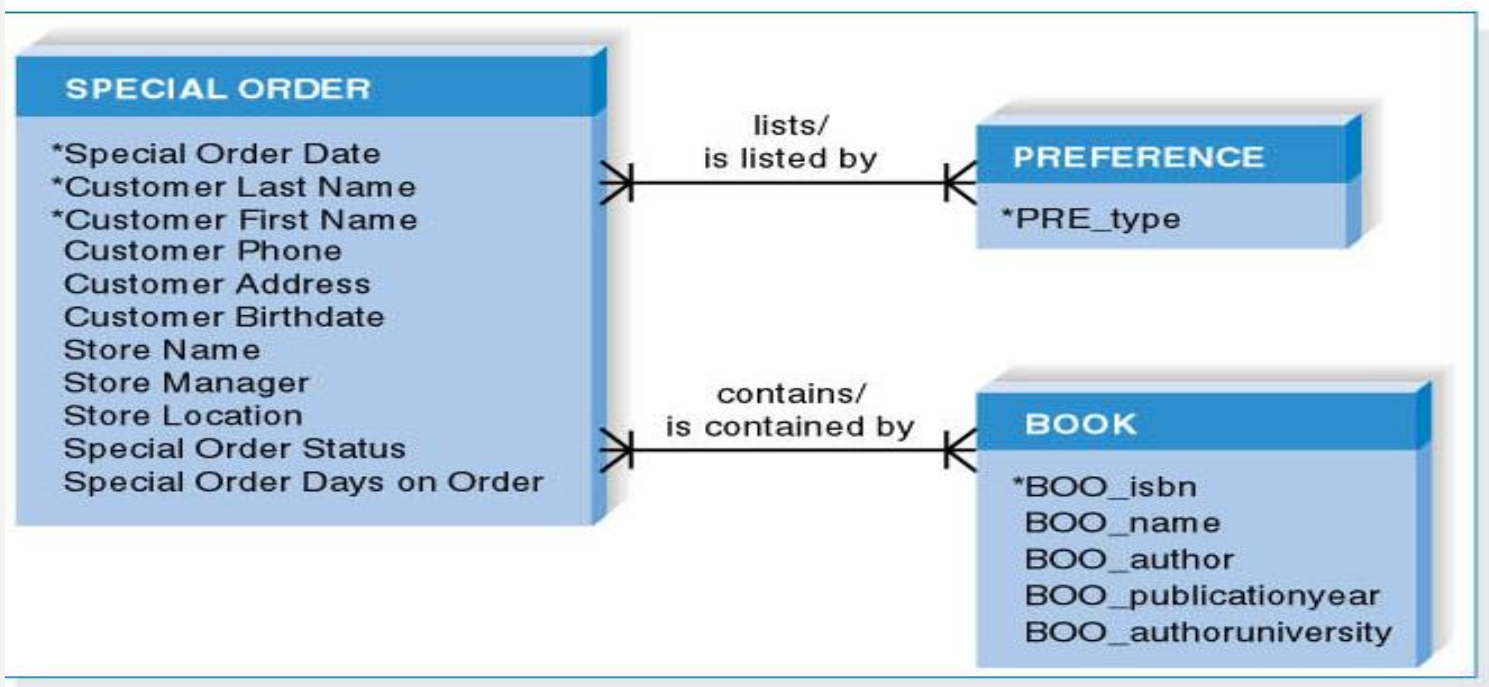
Un-normalized Entity

Begin with an entity from the logical data model



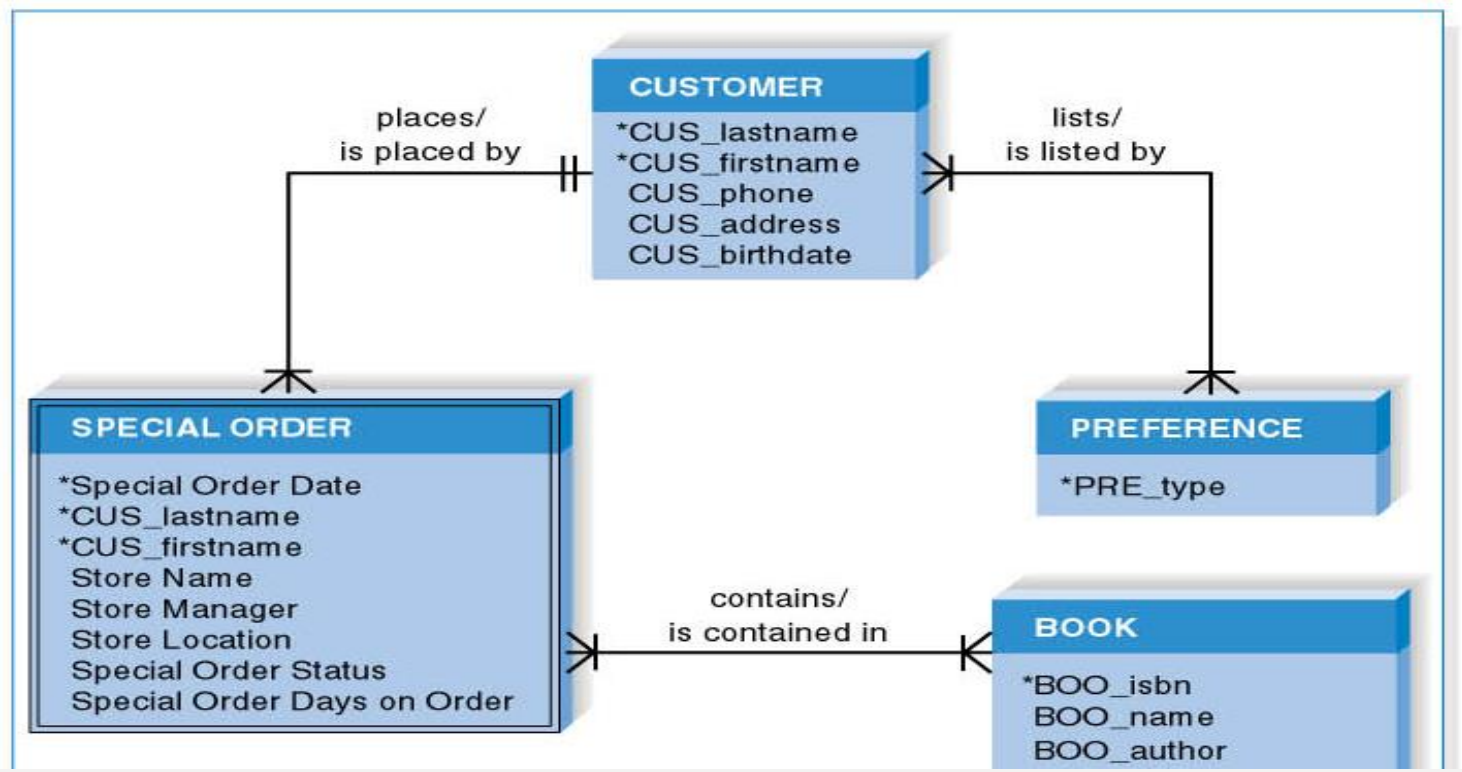
First Normal Form (1NF)

Look for repeating groups of attributes and remove them into separate entities



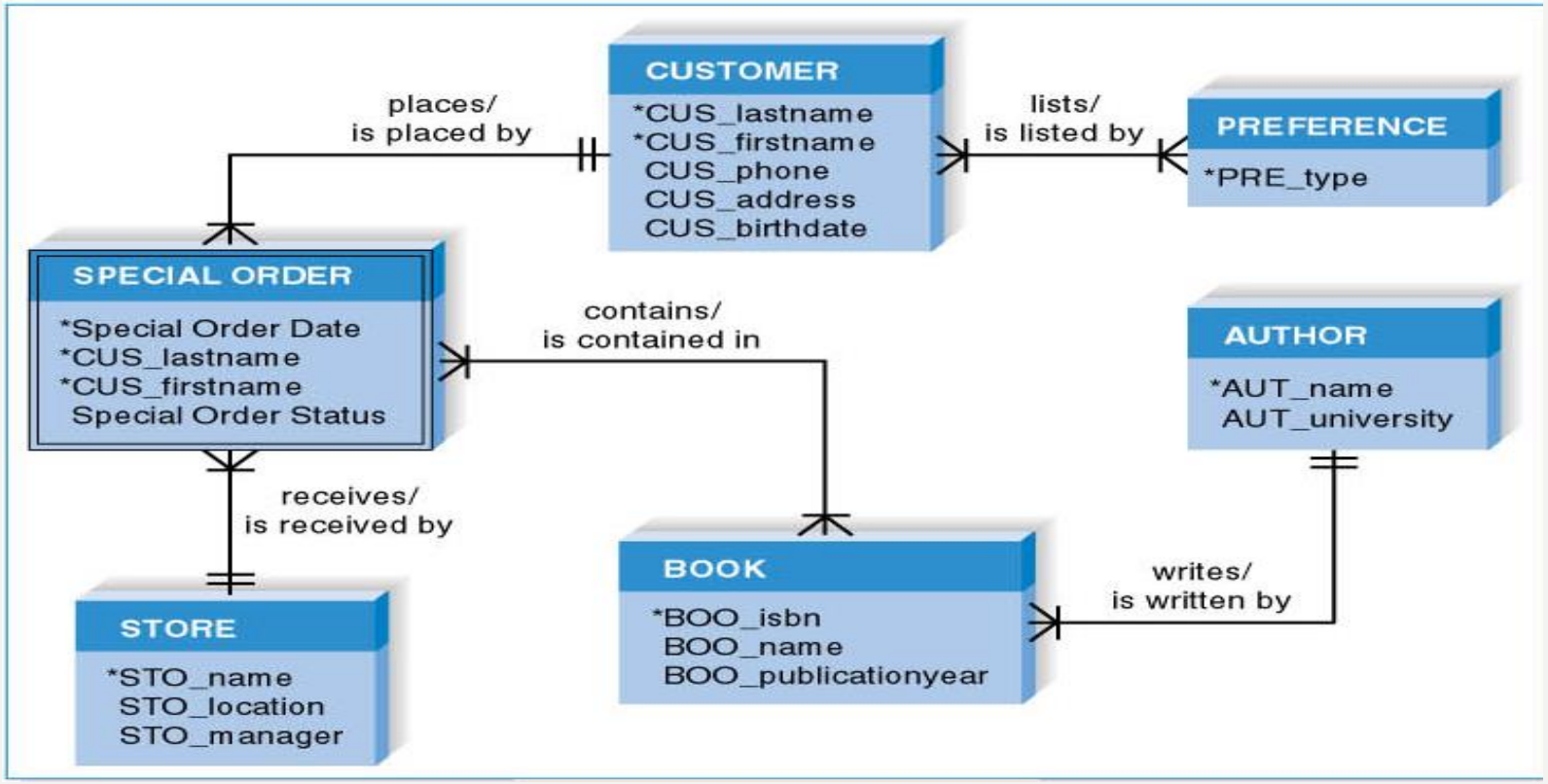
Second Normal Form (2NF)

If an entity has a concatenated identifier, look for attributes that depend only on part of the identifier. If found, remove to new entity.



Third Normal Form (3NF)

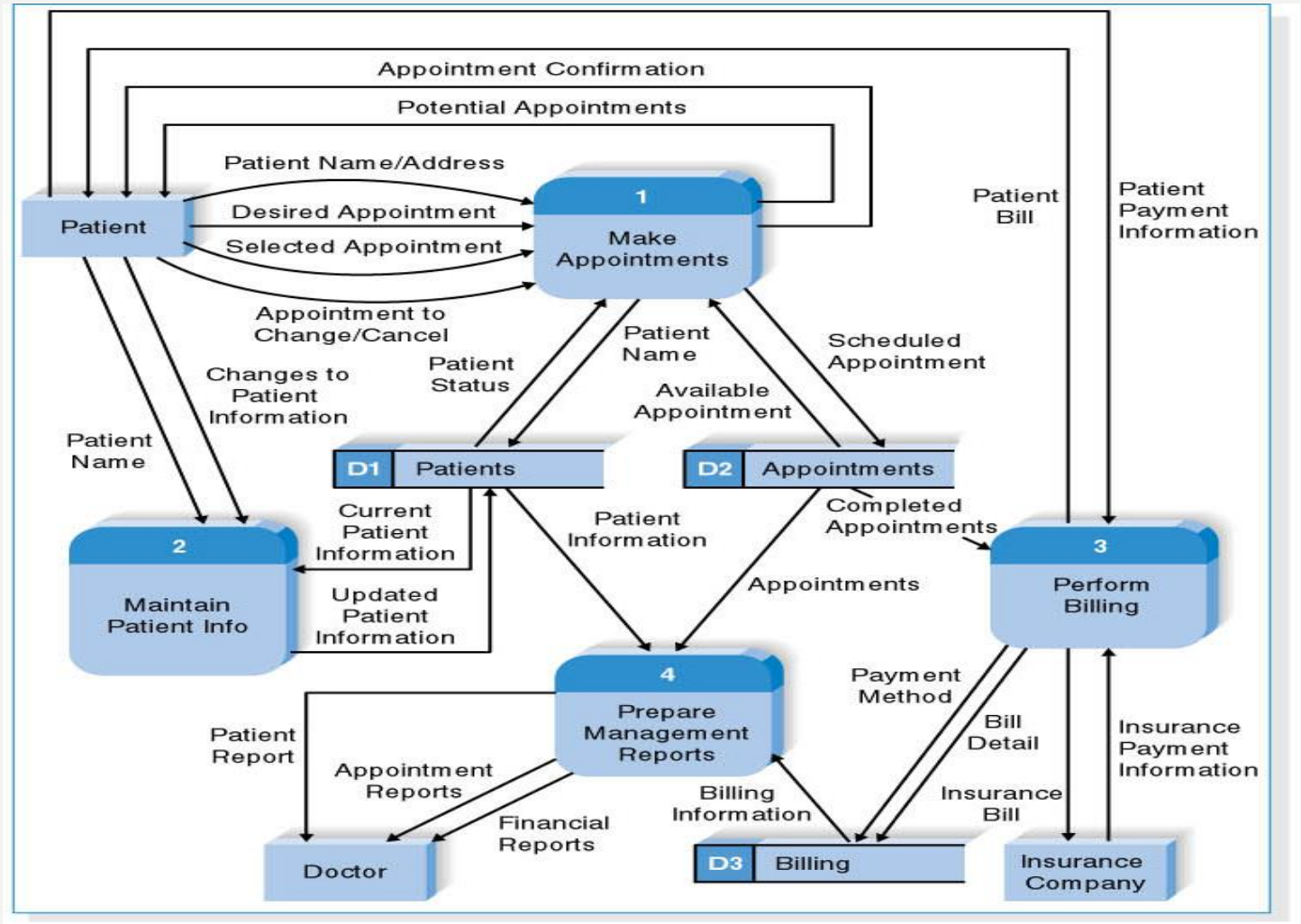
Look for attributes that depend only on another non-identifying attribute. If found, remove to new entity. Also remove any calculated attributes.



Balancing ERDs with DFDs (1)

- All analysis activities are interrelated
- Process models contain two data components
  - Data flows and data stores
- The DFD data components need to balance the ERD's data stores (entities) and data elements (attributes)
- Many CASE tools provide features to check for imbalance
- Check that all data stores and elements correspond between models
  - Data that is not used is unnecessary
  - Data that has been omitted results in an incomplete system
- Do not follow thoughtlessly -- check that the models make sense!

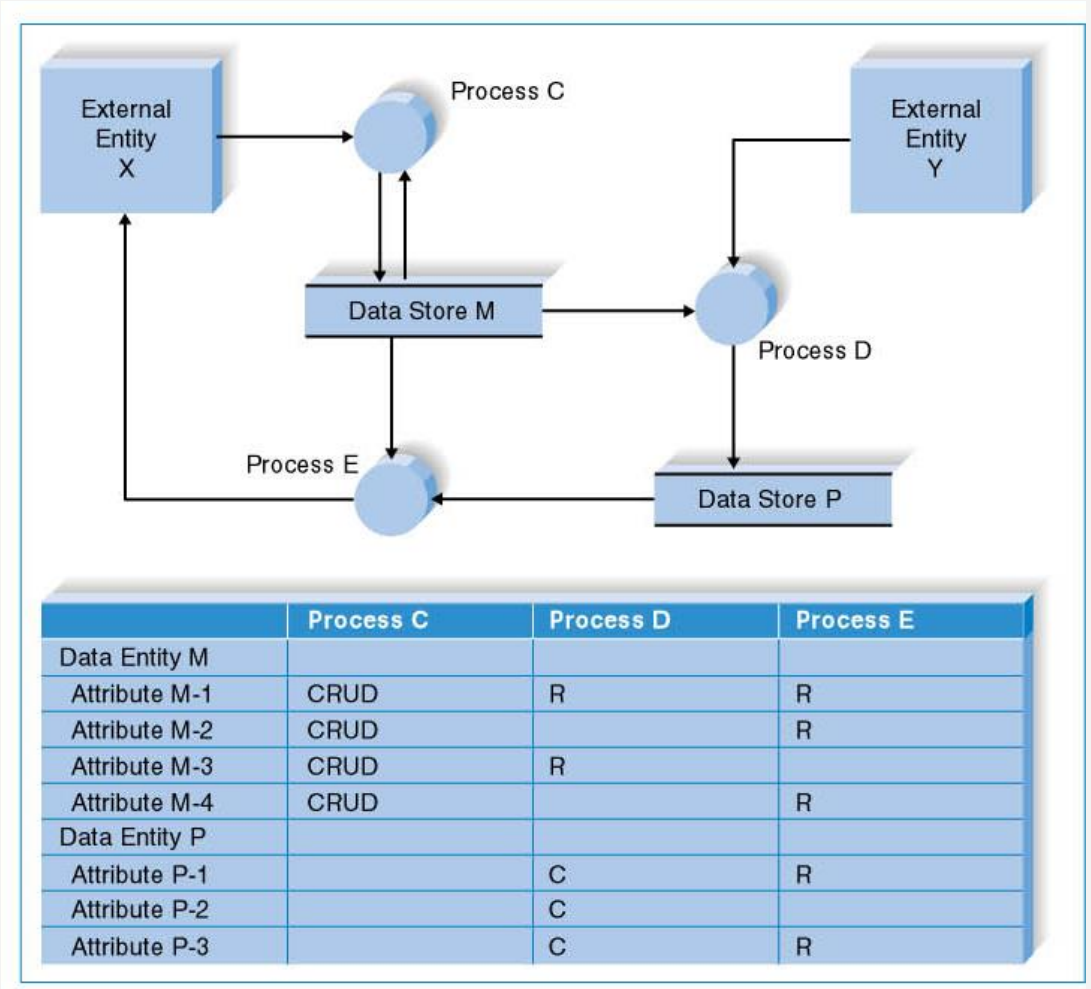
Balancing ERDs with DFDs (2)



Balancing ERDs with DFDs (3)

- Take a look at the DFD in Figure 6-8 and ERD in Figure 7-1.
- The doctor, payment and insurance company do not appear on the DFD as data store.

Partial Process Model and CRUD Matrix



	Process C	Process D	Process E
Data Entity M			
Attribute M-1	CRUD	R	R
Attribute M-2	CRUD		R
Attribute M-3	CRUD	R	
Attribute M-4	CRUD		R
Data Entity P			
Attribute P-1		C	R
Attribute P-2		C	
Attribute P-3		C	R

Summary

- The ERD is the most common technique for drawing data models. The building blocks of the ERD are:
  - **Entities** describe people, places, or things
  - **Attributes** capture information about the entity
  - **Relationships** associate data across entities
- Intersection, dependent, and independent entities must be recognized.
- The ERD must be balanced with the DFD.

Your Turn ☺

Draw an entity relationship diagram (ERD) for the following situations:

Whenever new patients are seen for the first time, they complete a patient information form that asks their name, address, phone number, and insurance carrier, all of which are stored in the patient information file. Patients can be signed up with only one carrier, but they must be signed up to be seen by the doctor. Each time a patient visits the doctor, an insurance claim is sent to the carrier for payment. The claim must contain information about the visit, such as the date, purpose, and cost. It would be possible for a patient to submit two claims on the same day.

Your Turn ☺

Draw an entity relationship diagram (ERD) for the following situations:

Jim Smith's dealership sells Fords, Hondas, and Toyotas. The dealership keeps information about each car manufacturer with whom it deals so that employees can get in touch with manufacturers easily. The dealership also keeps information about the models of cars that it carries from each manufacturer. It keeps such information as list price, the price the dealership paid to obtain the model, and the model name and series (e.g., Honda Civic LX). The dealership also keeps information about all sales that it has made. (For instance, employees will record the buyer's name, the car the buyer bought, and the amount the buyer paid for the car.) To allow employees to contact the buyers in the future, contact information is also kept (e.g., address, phone number, e-mail)

References

- Systems Analysis and Design: An Object Oriented Approach 5th ed. Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth © 2015