

Data Science Joint Education Program

INFO 200 Information Systems Analysis

Chapter #7

Using Data Flow Diagrams

Keywords and Phrases

| | |
|---|----------------------------------|
| Ajax | level 0 diagram |
| base element | logical models |
| child diagram | online process |
| context-level data flow diagrams | parent process |
| data flow diagram (DFD) | partitioning |
| data store | physical data store |
| derived element | physical model |
| event modelling | primitive process |
| event response table | top-down approach |
| event trigger | transaction data store |
| exploding | transforming process |
| external entity (source (or) destination) | Unified Modelling Language (UML) |
| functionally primitive | vertical balancing |
| interface data flow | |

Chapter #7 Learning Objectives

- On completion of this chapter you will:
 - Understand the importance of using logical and physical data flow diagrams (DFDs) to graphically depict movement for humans and systems in an organization
 - How to create, use, and explode logical DFDs to capture and analyze the current system through parent and child levels
 - How to develop and explode logical DFDs that illustrate the proposed system
 - How to produce physical DFDs based on logical DFDs you have developed.
 - Understand and apply the concept of partitioning of physical DFDs.

Data Flow Diagrams

- Data-flow diagrams are a modelling approach to identify and show the how data flows within an organisation
- Data flow diagrams graphically characterize data processes and flows in a business system and depict:
 - System *inputs*
 - System *processes*
 - The *outputs* from a system

Data Flow Advantages

- The data flow approach has four main advantages over the narrative approach which explains how data moves through a system:
 - Freedom from committing to the technical implementation of the system too early
 - Further understanding of the interrelatedness of systems and subsystems
 - Communicating current system knowledge to users through DFDs
 - Analysis of a proposed system to determine whether the necessary data and processes have been defined

Major Topics Introduced

- The important topics addressed in Chapter #7 are:
 - Data flow diagrams (DFD)
 - Data flow diagram *symbols*
 - Data flow diagram *levels*
 - *Creating* data flow diagrams
 - *Physical* and *logical* data flow diagrams
 - *Partitioning*
 - *Communicating* using data flow diagrams

The Data Flow Approach to Human Requirements Determination

Requirements Determination

- Systems analysis attempts to understand system users information requirements
- an analysis must visualise and conceptualise how data flows through a system, the process (or transformation) that data undergo, and the nature of the outputs
- While interviews and investigations of hard data provide a system narrative
 - A visualisation and data depiction can usefully crystallise this information for users and the analyst

Requirements Determination

- Using a structured analysis technique termed DFD's:
 - A systems analyst can create a graphical representation of data processes in an organization
- The use of a standardised set of symbols:
 - The creation of a pictorial depiction processes can help to provide solid system documentation

Conventions Used in Data Flow Diagrams

Data Flow Diagram Basic Symbols (1)

- Four basic symbols are used in data movement in a DFD:
 - A double square
 - Used to depict an *external entity* (e.g., another department / a business / a person / a machine)
 - Also termed a *source* (or) *destination* of data
 - Named with a *noun*
 - An entity can be used *multiple times*
 - An arrow
 - Shows the *movement* of data (*data flows*)
 - It should be named as a *noun*

Data Flow Diagram Basic Symbols (2)

- Four basic symbols are used in data movement in a DFD:
 - An arrow:
 - Shows the movement of data (*data flows*) from one point to another
 - It should be named as a noun
 - Represents data about a person, place, or thing (an entity)
 - A rectangle with rounded corners
 - Shows the data transformation process
 - An open-ended rectangle
 - Closed on the left side and open on the right side
 - Represents a data store
- Figure 7.1 models the four basic symbols used in data flow diagrams with their meanings and examples

Notation – The Basic Symbols






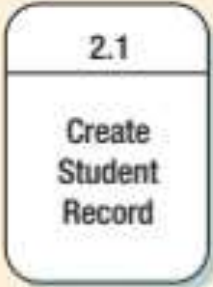


| Symbol | Meaning | Example |
|---|------------|---|
|  | Entity |  |
|  | Data Flow |  |
|  | Process |  |
|  | Data Store |  |

FIGURE 7.1

The four basic symbols used in data flow diagrams, their meanings, and examples.

The Process

- Process denotes:
 - A change in or transformation of data
 - Represents work being performed in the system
- Naming conventions:
 - 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

The Data Store

- A data store is:
 - A depository for data that allows examination, addition, and retrieval of data
 - Named with a noun and describes the data
 - Data stores are usually given a unique reference number (e.g., D1, D2, D3)
- Represents:
 - A database
 - A computerized file
 - A filing cabinet

Developing Data Flow Diagrams

Developing Data Flow Diagrams

- Data flow diagram development must be drawn systematically
- Figure 7.2 summarises the steps in drawing DFD's
- The process can be summarized as follows:
 1. The initial step is to conceptualise data flows from a top-down perspective
 2. Reduce the system narrative (or stories) into a list of four categories of external entities / data flow(s) / the data store
 3. The list provides a basis for determining the boundaries of a system
 4. The list provides the basis for the drawing of the context diagram

The Data Flow

- Figure 7.2 shows:
 - The steps in developing data flow diagrams

FIGURE 7.2

Steps in developing data flow diagrams.

Developing Data Flow Diagrams Using a Top-Down Approach

1. Make a list of business activities and use it to determine various
 - External entities
 - Data flows
 - Processes
 - 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.

Developing Data Flow Diagrams

- Data flow diagram development follows a number of basic rules:
 1. A DFD must have at least one process
 2. A DFD must not have any freestanding objects or objects connected to themselves
 3. A process must receive at least one data flow coming into the process and create at least one data flow leaving from the process
 4. A data store should be connected to at least one process
 5. External entities should not be connected to each other
 1. While external entities communicate independently
 2. The communication is not part of the system we design using DFD's

Creating the Context Diagram

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
- Processes that do not create a child diagram are called primitive

Context Diagrams

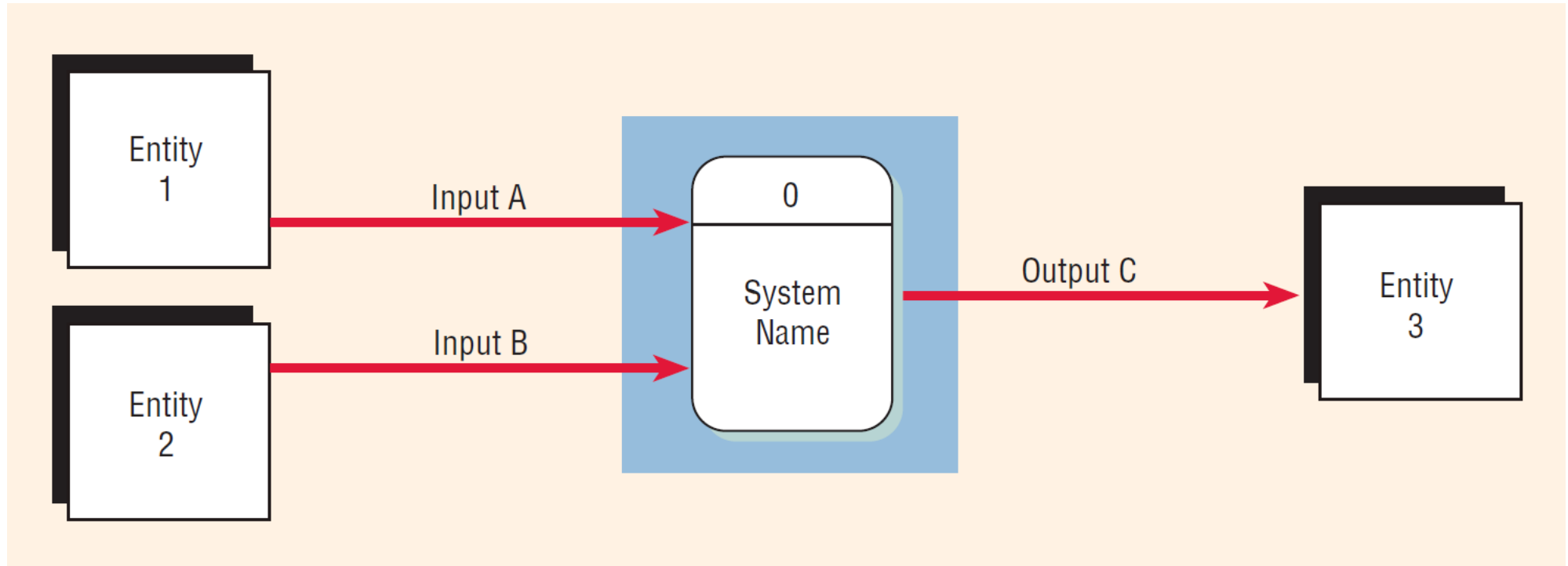
- In the top-down approach to modelling data flows DFDs move from the general to the specific
- The initial context diagram:
 - Should be an overview which includes basic inputs, the general system, and the outputs
 - The diagram will be the most general diagram which provides a 'birds-eye-view' of the data movement in the system with the broadest possible conceptualisation of the system

Context Diagrams

- The context diagram is the highest level in a DFD and contains only process representing the entire system
 - The process is given the number 0 (zero)
 - All external entities are shown on the context diagram with major data flow to and from the entities
 - The context diagram does not have any data stores
 - The context diagram is simple to create once the external entities and data flow to and from the entities are identified in the analysis
- Figure 7.3 shows how a context diagram can be “exploded” into diagram 0 (zero)
 - From Figure 7.3 we can see the increased detail in diagram 0 (zero)

Context Diagram (Figure 7.3)

- This slide shows part of Figure 7.3
- Shown below is the context diagram which is expanded into diagram 0 (zero)



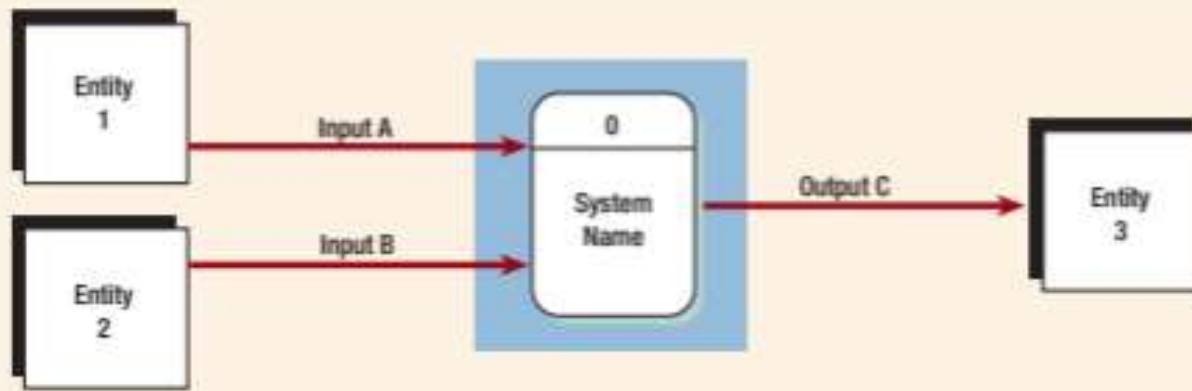
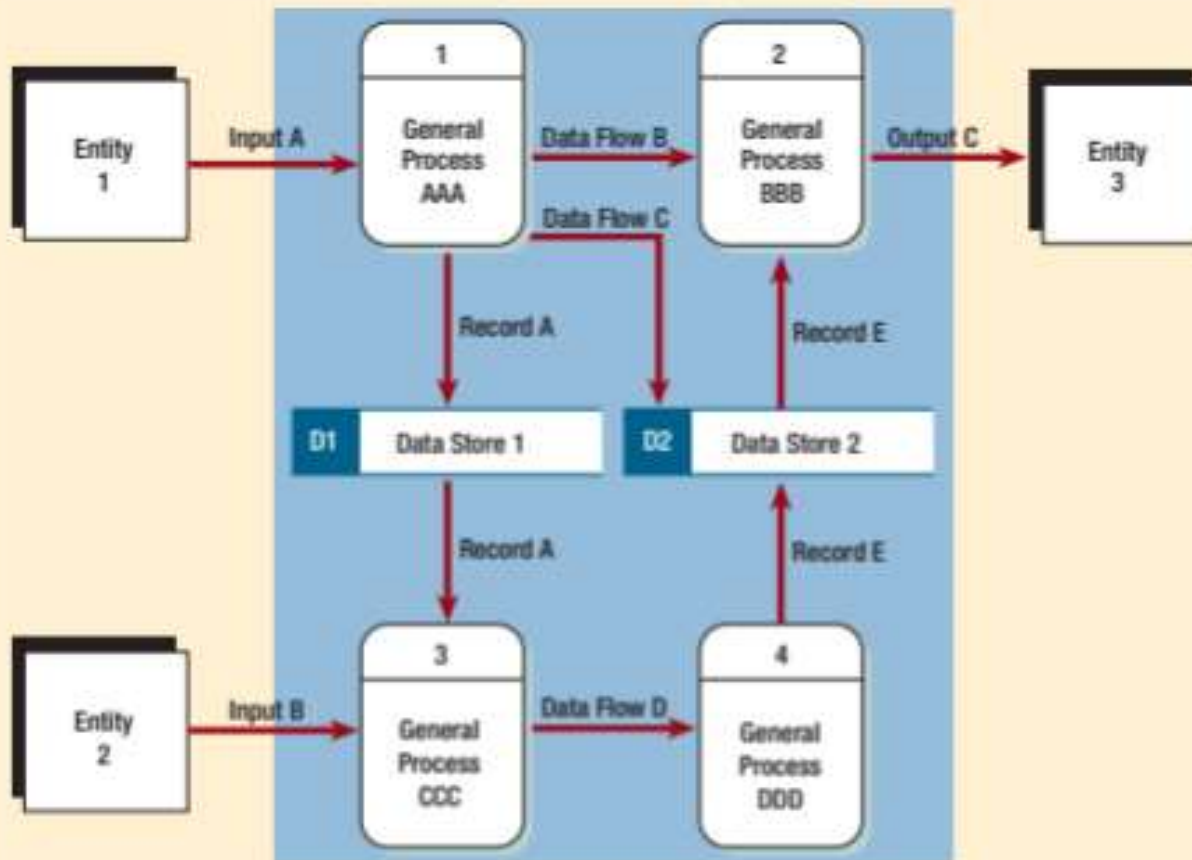


FIGURE 7.3

Context diagrams (top) can be “exploded” into Diagram 0 (bottom). Note the greater detail in Diagram 0.



Drawing Diagram 0 (zero) (the next level)

Diagram 0 (zero)

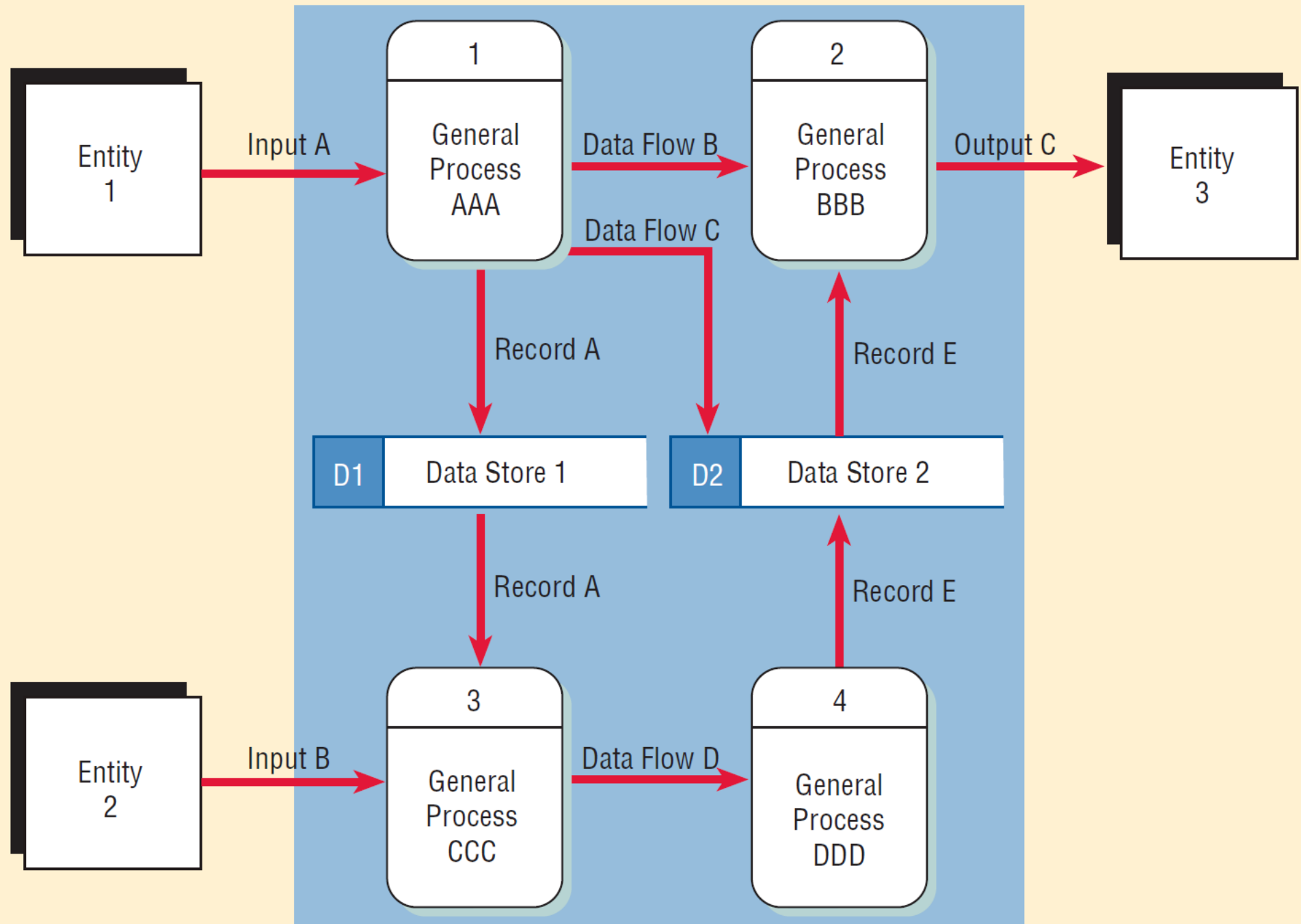
- Diagram 0 is the “explosion” of the context diagram (see Figure 7.3) and may use up to 9 processes
 - Each process is numbered with an integer and generally starts at the upper left-hand corner of the diagram working to the lower right-hand corner
 - The major data stores of the system (representing master files) and all external entities form part of diagram 0 (zero)
- As a DFD as two-dimensional (not linear) you may start at any point and work backwards and forwards through the diagram

Diagram 0 (zero)

- If it is not clear what to include at any point:
 - Take a different external entity, process, or data store, and start drawing the flow from it
- The process is as follows:
 - 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

Figure 7.3

- A development of the context diagram
- note the greater Detail in Diagram 0



Creating Child Diagrams (more detailed levels)

Child Diagrams

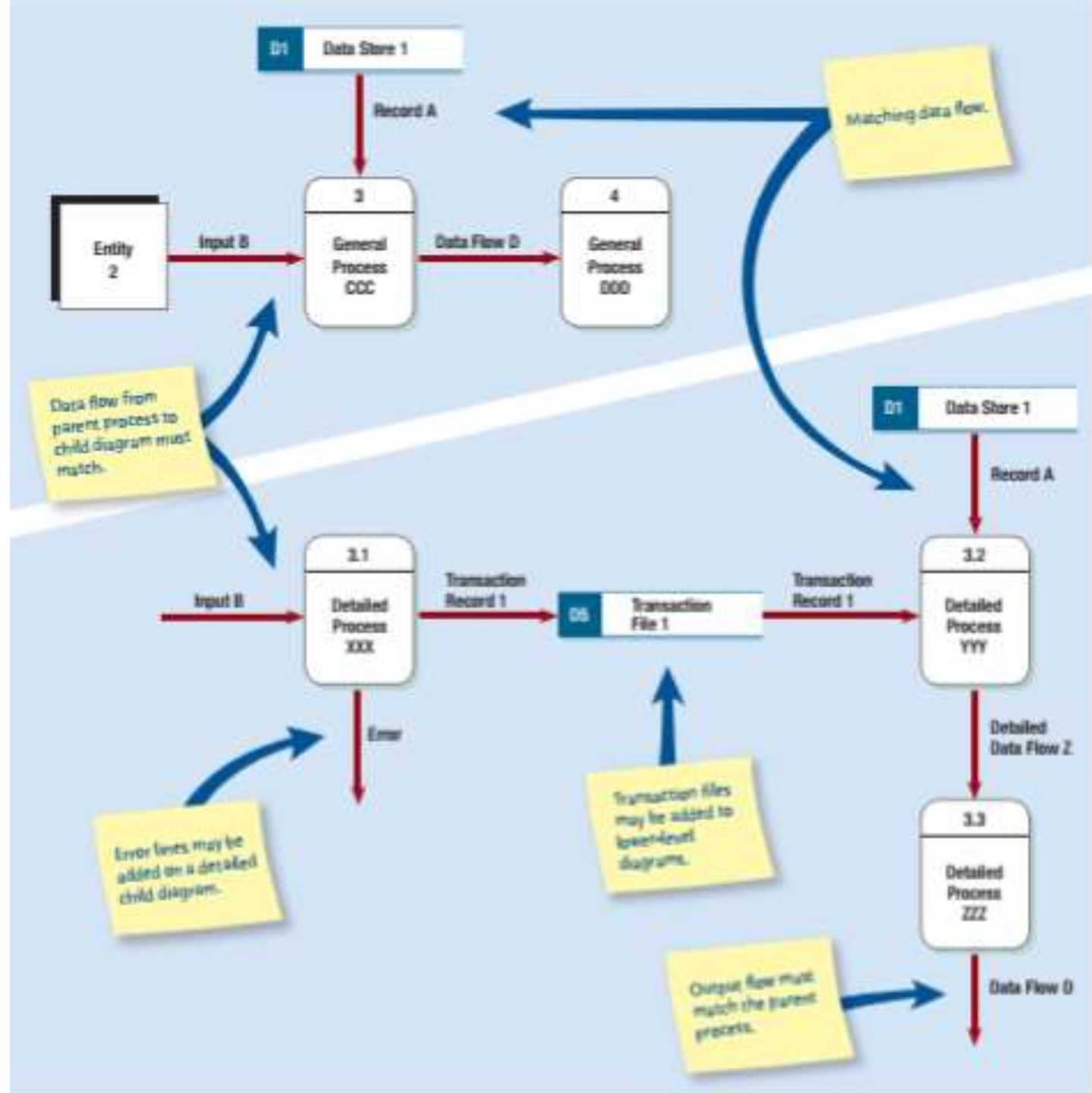
- Each process on a diagram 0 can be developed into a more detailed child diagram
- The primary process on diagram 0 (that is “exploded”) is termed a *parent process* – the resulting diagram is termed the *child diagram*
- The primary rule for creating child diagrams is *vertical balancing*
- All data flow into (or) out of the parent process must be shown flowing into (or) out of the child diagram
- The child diagram has the same number of processes as diagram 0
 - Process 3 would explode to Diagram 3

Child Diagrams

- Entities are not (usually) shown on the child diagram below diagram 0
- Data flow that matches the parent flow is termed an *interface data flow* and is shown as an arrow from (or) into a blank area of the child diagram
- Processes may (or may not) be “exploded” dependant on the complexity level
- Logic is written to describe these processes (see Chapter #9)
 - Figure 7.4 shows the detailed levels in a child DFD

Child Diagrams

- Figure 7.4
- The differences between the:
 - Parent diagram (top)
 - Child diagram (below)



Checking Diagrams for Errors

Checking DFD's for Errors

- A “pretty” model may look good but it may not be logical or correct
- Common errors in preparing DFD's include:
 - 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
 - Including more than nine processes on a data flow diagram
 - Omitting data flow
 - Creating unbalanced decomposition (or explosion) in child diagrams

DFD Errors

- Figure 7.5
- Typical errors that can occur in a DFD
- The figure shows the payroll example

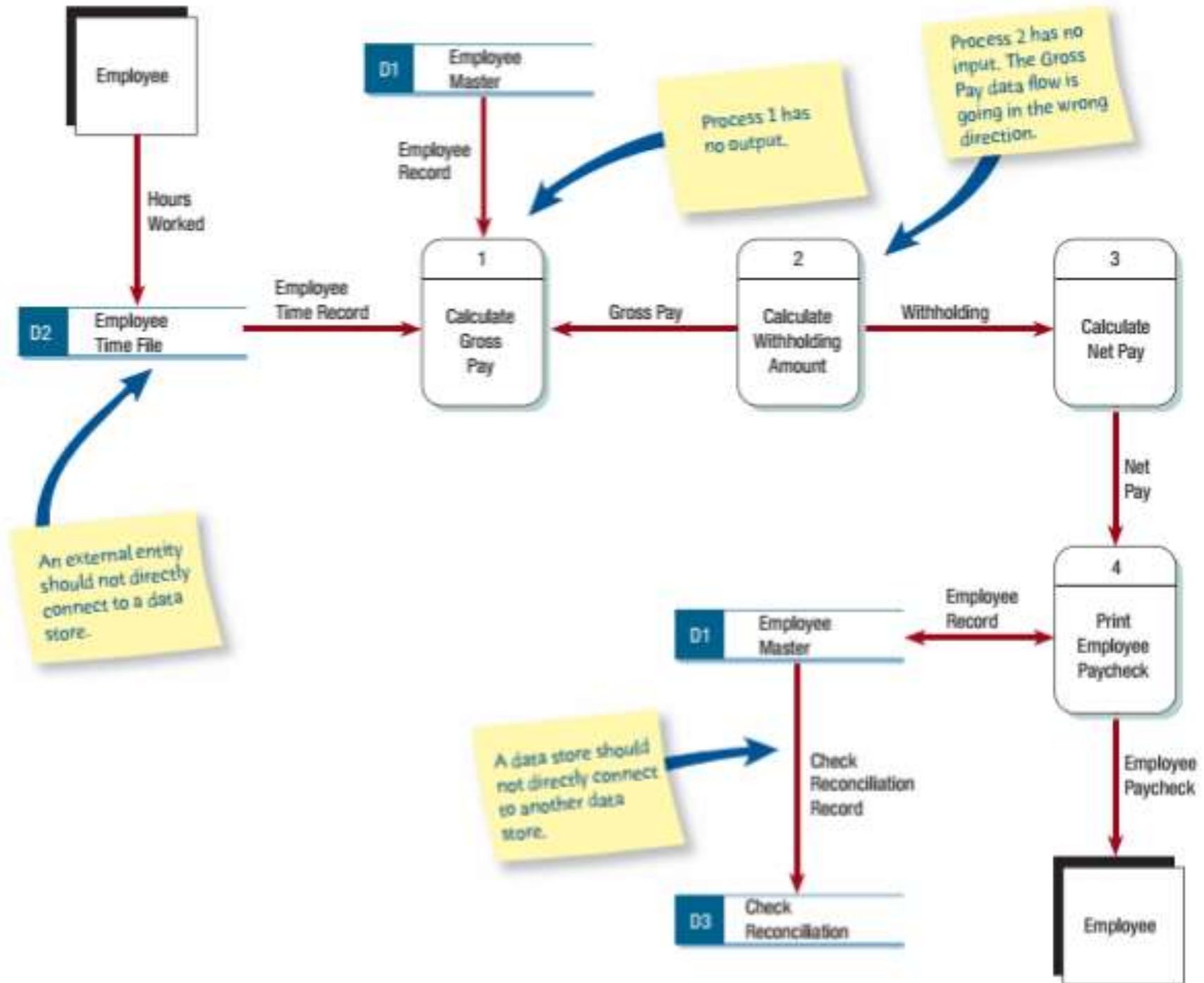
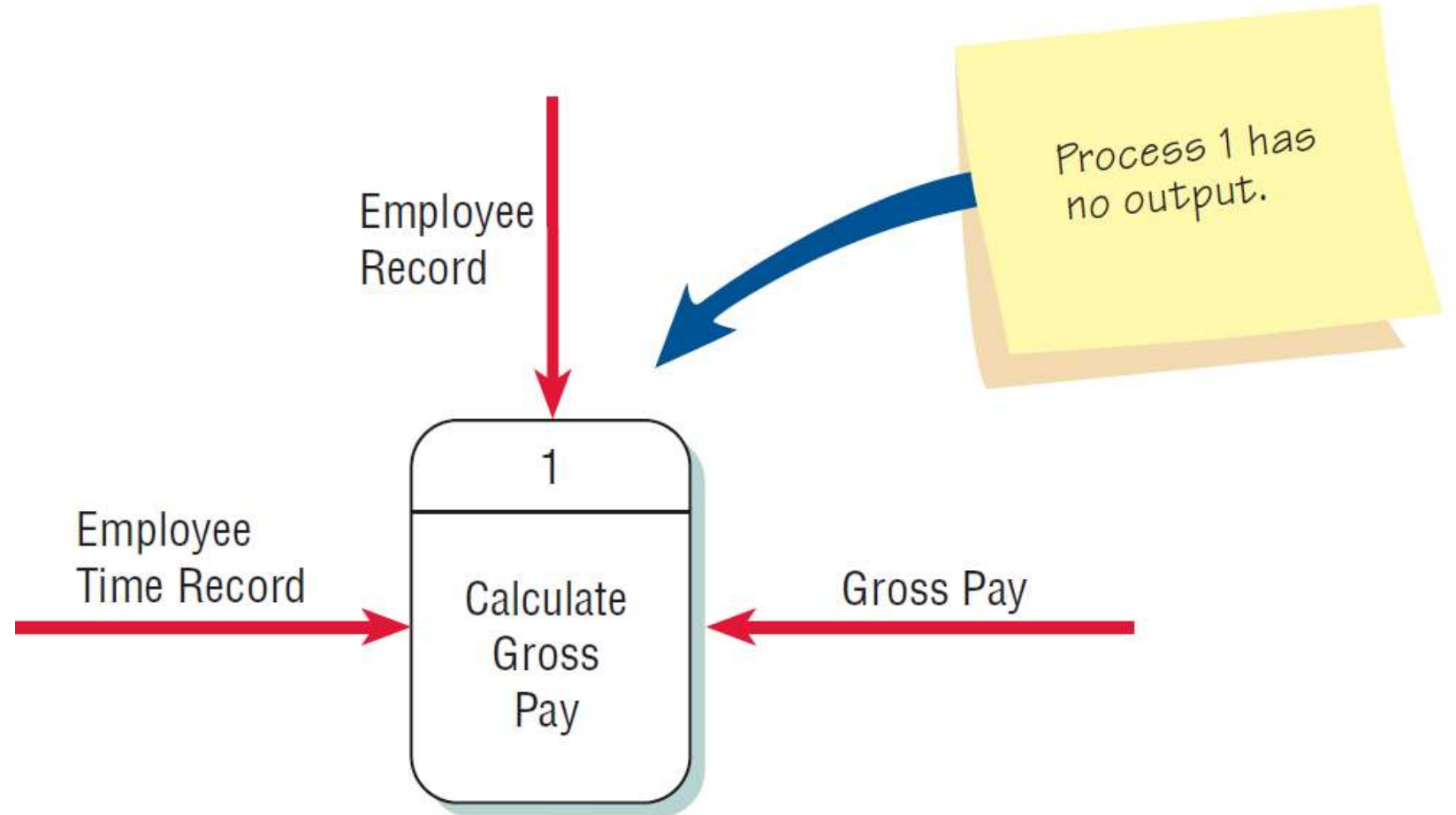


FIGURE 7.5

Typical errors that can occur in a DFD (payroll example).

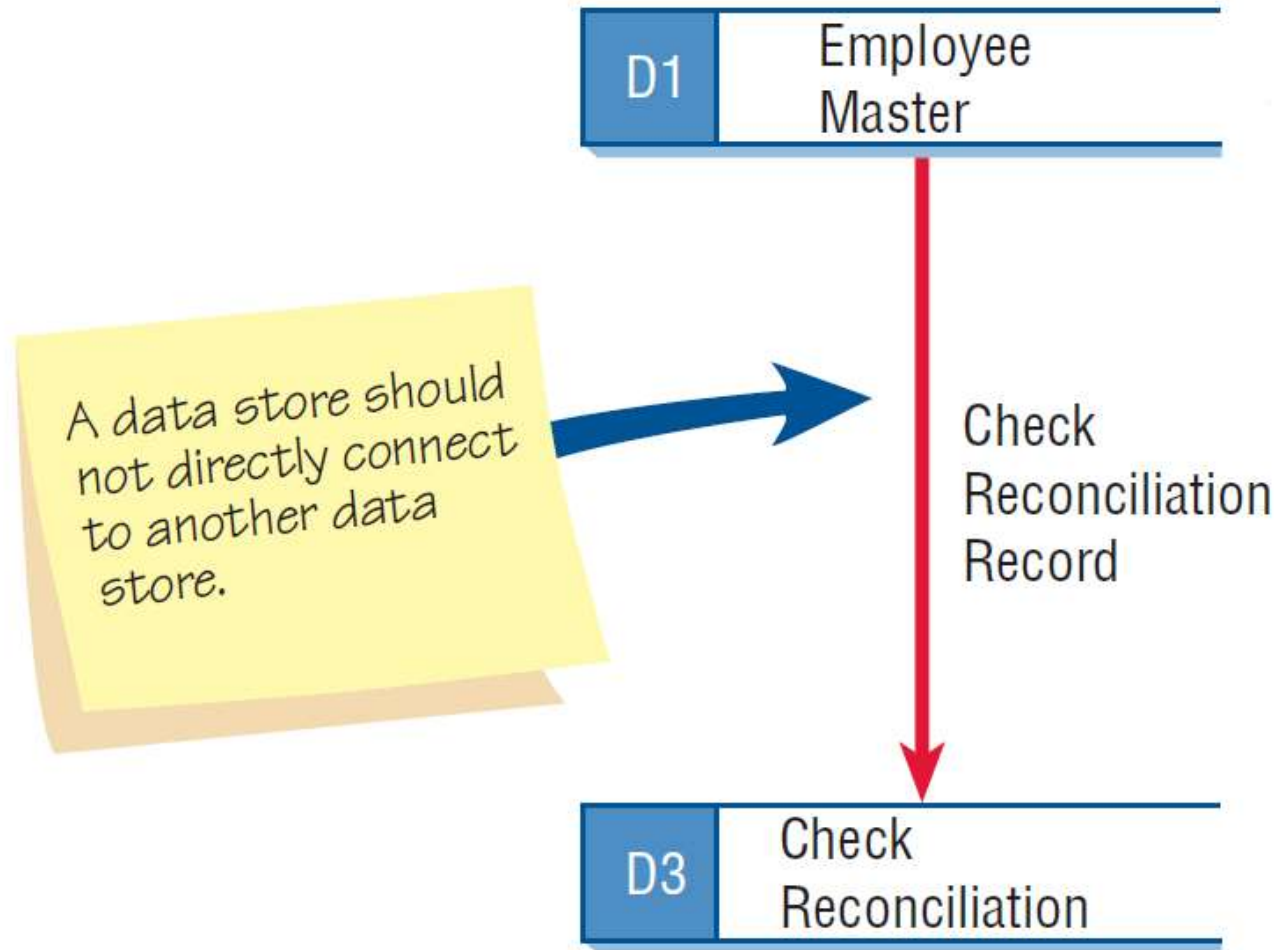
Checking DFD's for Errors

- Figure 7.5
- Forgetting to include a data flow or pointing an arrow in the wrong direction



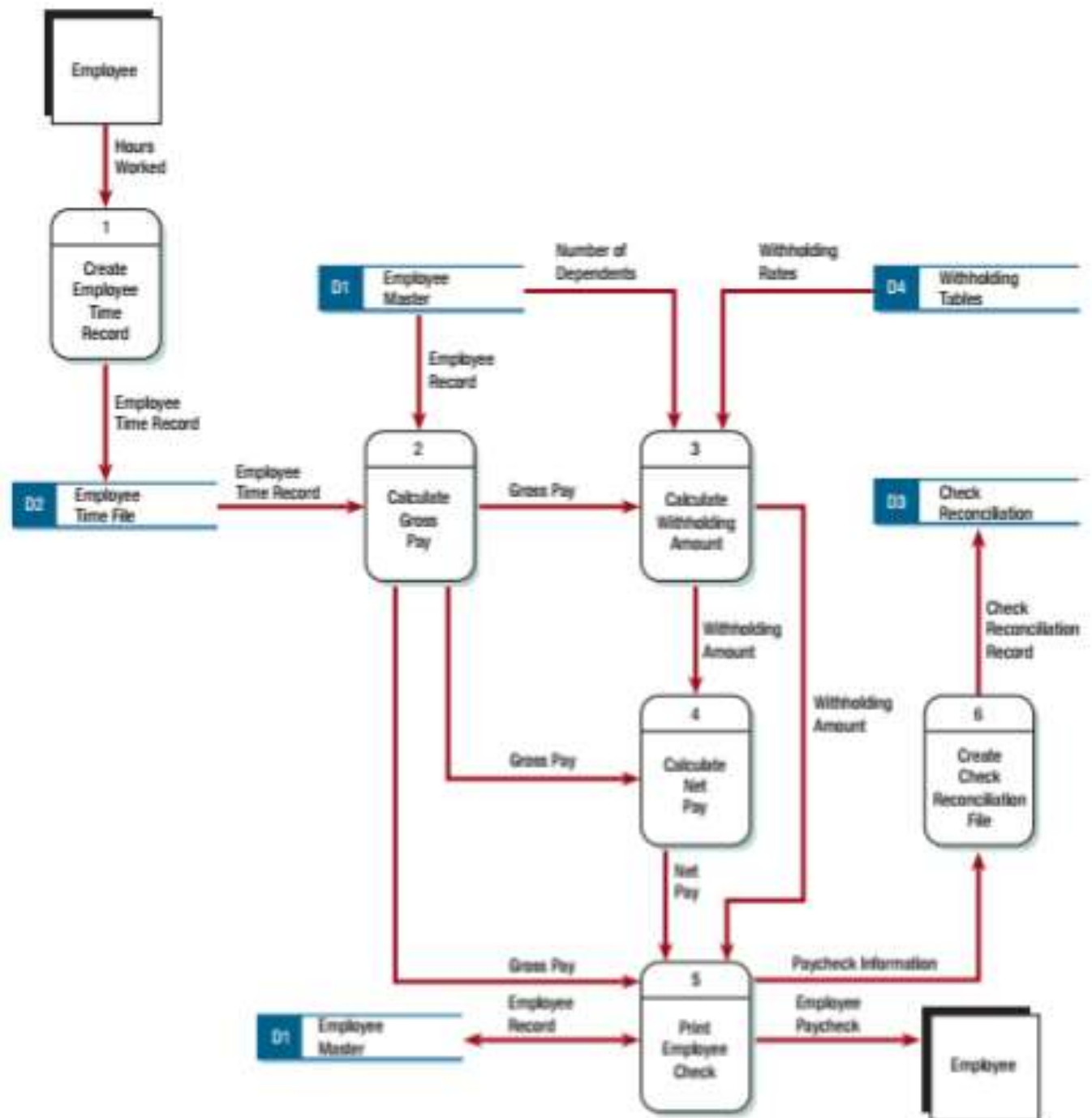
Checking DFD's for Errors

- Figure 7.5
- Connecting data stores and external entities directly to each other



Corrected DFD

- Figure 7.6
- Figure 7.6 shows a corrected DFD for the payroll example



Logical and Physical Data Flow Diagrams

Logical and Physical Data Flow Diagrams

- Data flow diagrams are categorized as *logical* or *physical* DFD's
- Such diagrams an organization and how the business operates
- Logical DFD's
 - Focuses on the business and how the business operates
 - Are 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 DFD's
 - Shows how the system will be implemented
 - Depicts the system

Diagram 0 (zero)

- Systems are developed:
 - By analysing the *current logical* DFD and then adding features that the new system should include (the *proposed logical* DFD)
 - Then the best methods for implementing the new system to be developed (the *physical* DFD)

Logical and Physical Data Flow Diagrams

- Figure 7.7 shows:
 - The features of *logical* and *physical* models
 - The *logical* model shows the business
 - The *physical* model shows the systems
- Figure 7.8 shows:
 - The progression of models from a logical DFD to a physical DFD
- Figure 7.9 shows the development of a *logical* DFD (top) into a *physical* DFD
 - The *physical* DFD (bottom) provides details not included on the logical DFD (top)

System Analysis and Development

- Systems are developed:
 - By analysing the *current* logical DFD and then adding features that the new system should include (the *proposed* logical DFD)
 - Then the best methods for implementing the new system to be developed (the physical DFD)
 - This progression is shown in Figure 7.8 shows
- Development of *logical* DFD's provide:
 - 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

FIGURE 7.7

Features common to both logical and physical data flow diagrams.

| 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). |

Figure 7.8

- Figure 7.8 shows the progression of models from a logical DFD to a physical DFD

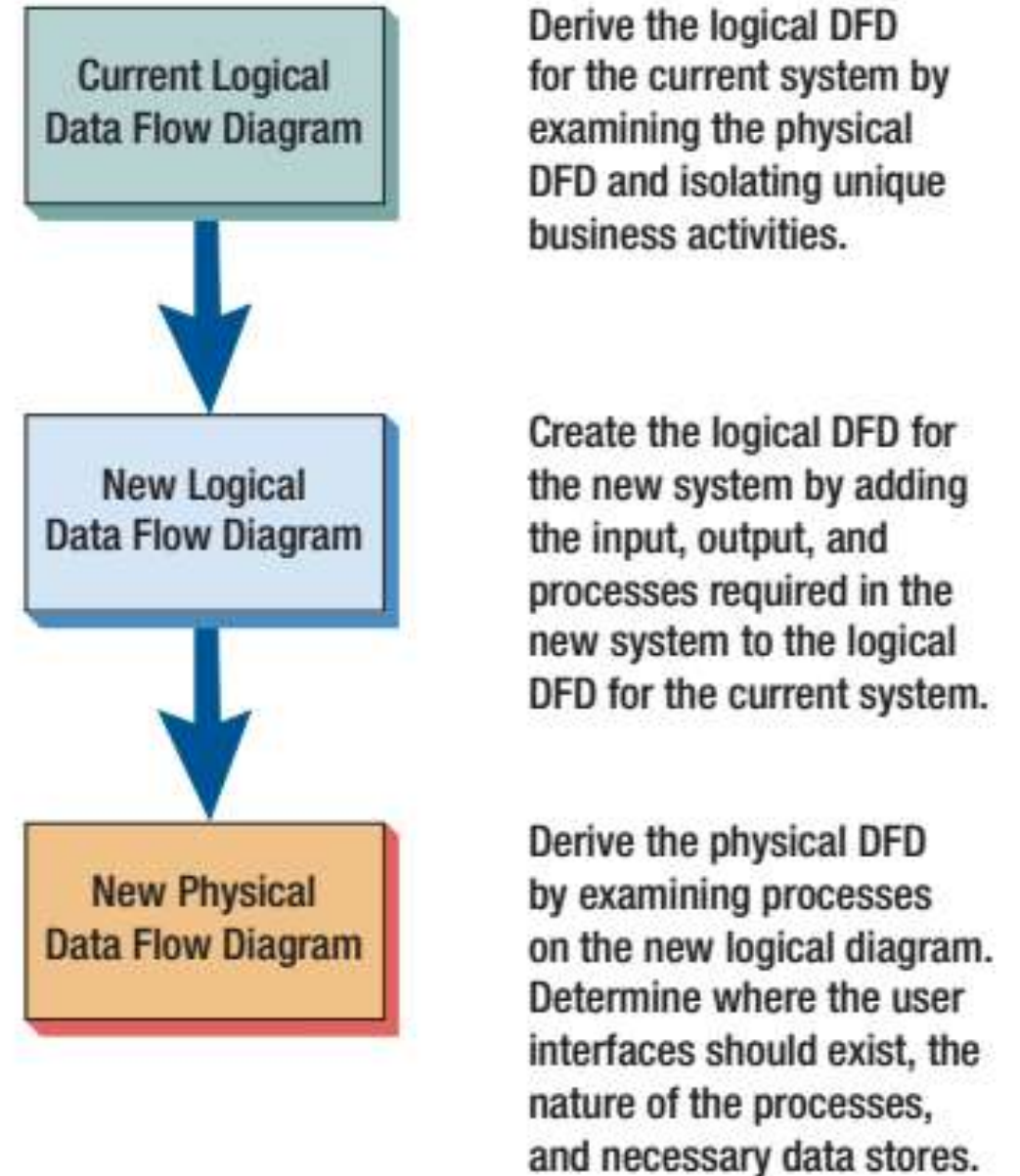
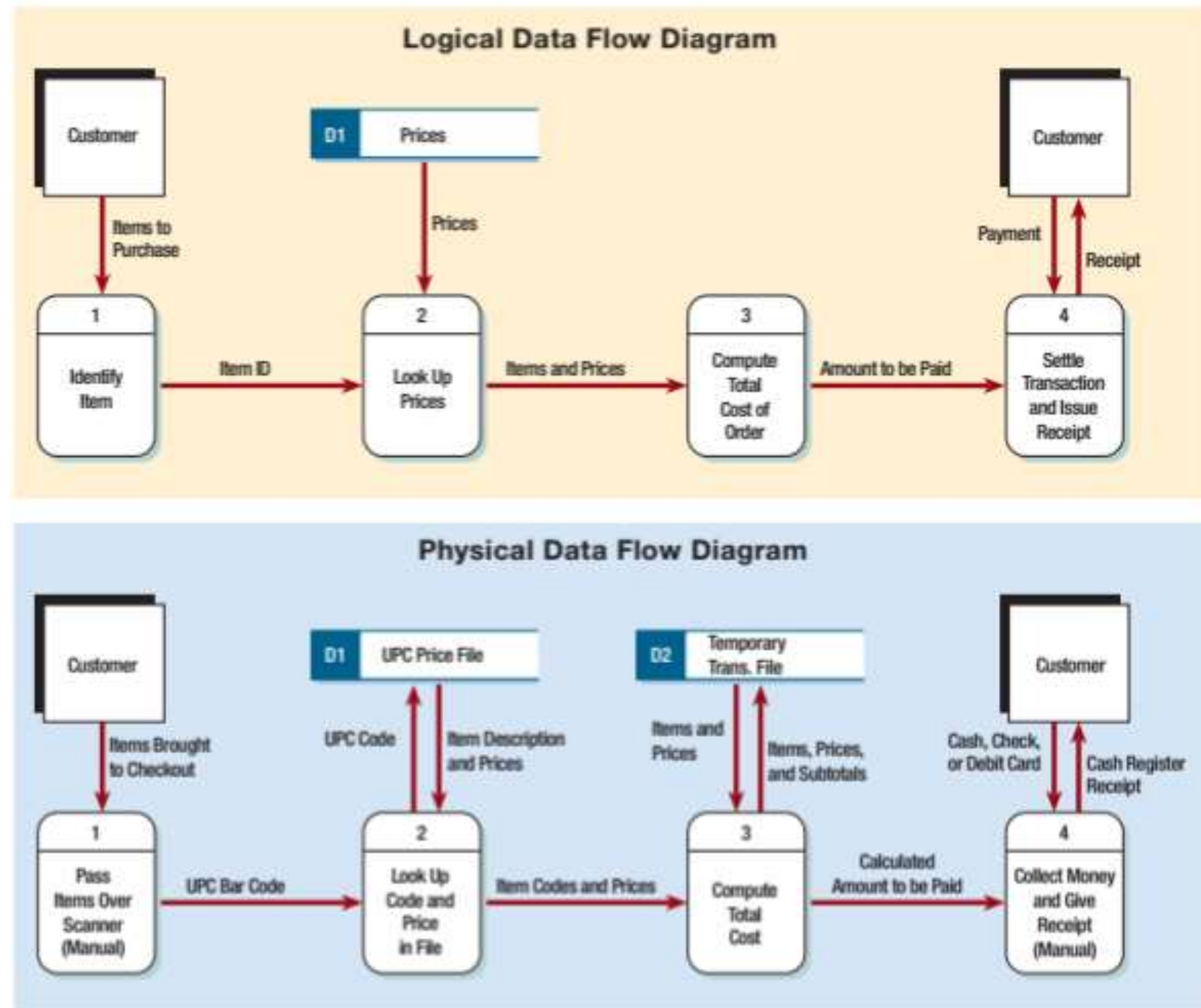


Figure 7.9

- Figure 7.9 shows the development of a logical DFD (top) into a physical DFD
- The physical DFD (bottom) provides details not included on the logical DFD



Developing Logical Data Flow Diagrams

Logical Data Flow Diagrams

- Always create a *logical* DFD for the current system before the creation of a *physical* DFD
- Development of *logical* DFD's provide a number of benefits including:
 - 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
- A *logical* model is easiest to use when communicating with users of the system because it's focus is on business activities

Logical Data Flow Diagrams

- Users will be familiar with the essential activities and many of the human information requirements of each activity
- Systems formed using a logical DFD are often relatively stable because they are based on business events and not on a particular technology or method of implementation
- *Logical* DFDs represent features of a system that would exist regardless of the *physical* means of doing business.
- For example:
 - Activities such as applying for a fitness club membership, accumulating rewards points, and paying for annual dues would all occur whether the club had an automated, manual, or hybrid system

Developing Physical Data Flow Diagrams

Developing Physical Data Flow Diagrams

- A *physical* DFD provides a number of advantages including:
 - 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
- Figure 7.10 shows:
 - The possible contents to be included in a *physical* DFD
 - Many of the items identified are not found in the *logical* DFD

Physical Data Flow Diagrams

- Figure 7.10 shows:
- The possible contents to be included in a *physical* DFD
- Many of the items in a *physical* DFD are not found in the *logical* DFD

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

The CRUD Matrix

- Physical DFDs are often more complex than logical DFDs simply because of the many data stores present in a system
- 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

The CRUD Matrix

- A CRUD matrix is a tool to represent where each of these processes occurs in a system
- Figure 7.11 shows:
 - A CRUD matrix for an online store
 - The tool can be used to represent the four processes within a system
 - There are processes with more than one activity
 - Data processes such as keying and verifying are part of physical DFD's

The CRUD Matrix (Figure 7.11)

- Figure 7.11 shows a CRUD matrix
- The tool can be used to represent the four processes within a system
- C: *create*
- R: *read*
- U: *update*
- D: *delete*

| 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 |

Physical Data Flow Diagrams

- Physical DFDs have intermediate data stores, often a transaction file or a temporary database table
- An very simple example of this concept is found in the everyday experiences of grocery shopping, meal preparation, and eating
- The activities are:
 1. Selecting items from shelves
 2. Checking out and paying the bill
 3. Transporting the groceries home
 4. Preparing a meal
 5. Eating the meal
- This simple example demonstrated the concept of “timing” which is a feature of physical DFD's

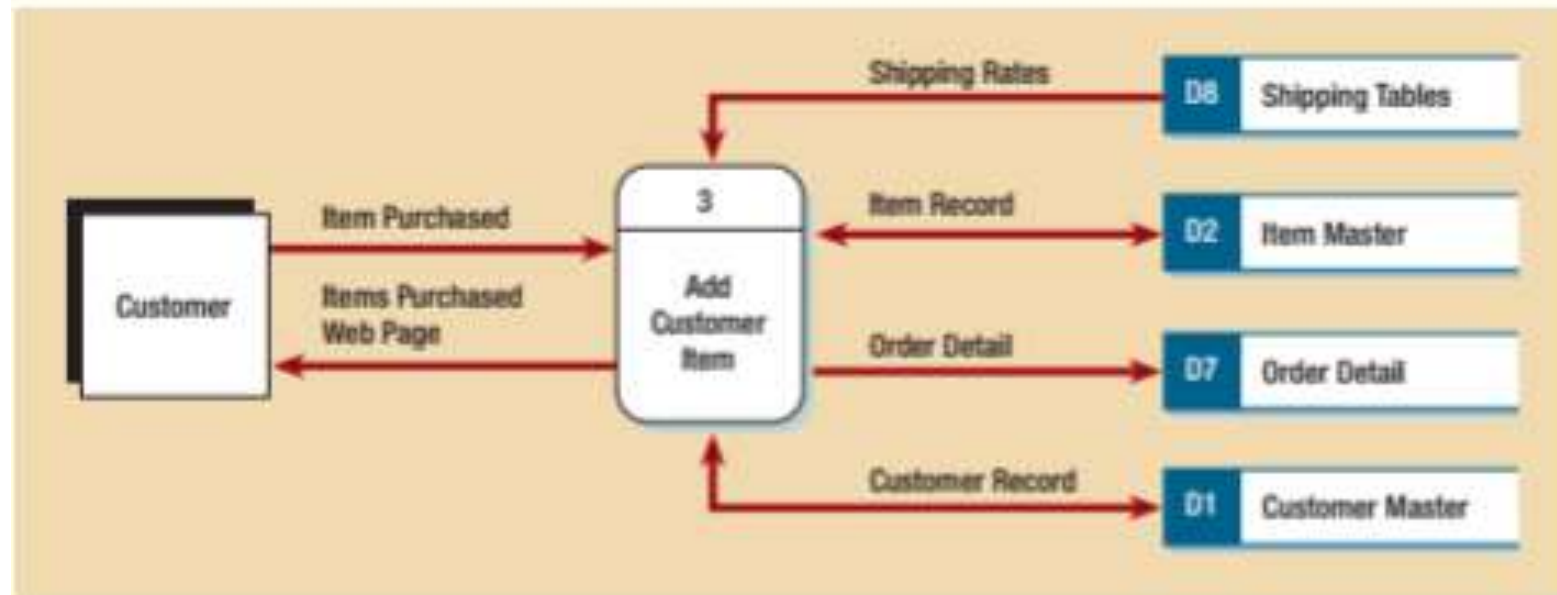
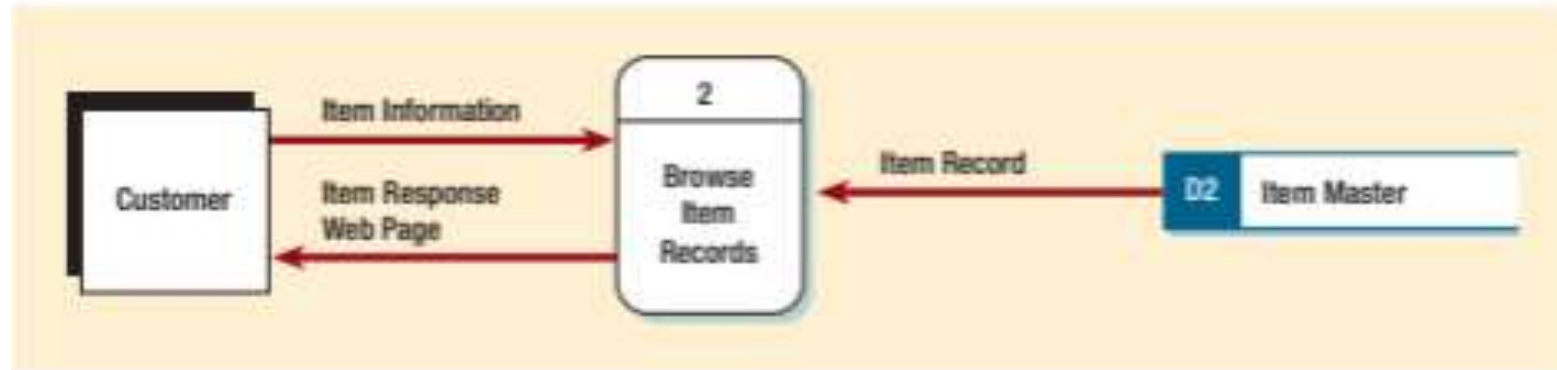
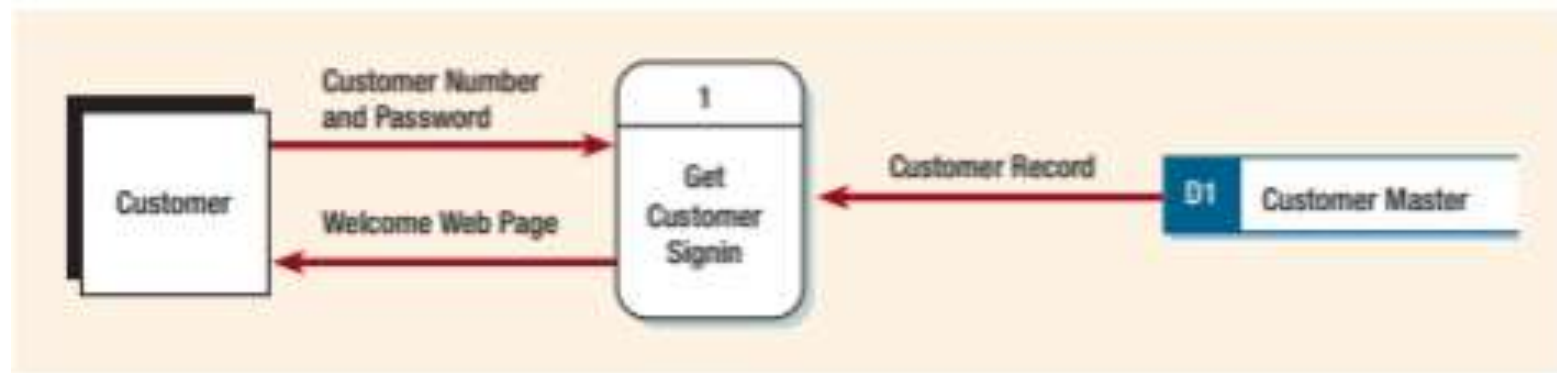
Event Modelling and Data Flow Diagrams

Event Modelling

- A practical approach to creating physical DFDs is to create a simple DFD fragment for each unique system event
 - Events cause the system to do something and act as a trigger to the system
 - An input flow from an external entity is sometimes called a trigger because it starts the activities of a process
- Figure 7.13 shows a DFD for three unique events
 - The figure shows a DFD for the first three rows of the Internet storefront event response table (see Figure 7.12)

Unique Event DFD

- Figure 7.13
- The figure shows a DFD for the first three rows of the Internet storefront event response table
- The event response table is shown in Figure 7.12



Event Response Tables

- Events are usually summarized in an event response table
 - An example of an event response table for an Internet storefront business is shown in Figure 7.12
- An event table is used to create a data flow diagram by analyzing each event and the data used and produced by the event
- Every row in an event table represents a data flow diagram fragment and is used to create a single process on a data flow diagram

Figure 7.12

| Event | Source | Trigger | Activity | Response | Destination |
|---|----------|--|---|---|-------------------------------------|
| Customer logs on | Customer | Customer number and password | Find customer record and verify password. Send Welcome web page. | Welcome web page | Customer |
| Customer browses items at Web storefront | Customer | Item information | Find item price and quantity available. Send Item Response web page. | Item Response web page | Customer |
| Customer places item into shopping basket at Web storefront | Customer | Item purchase (item number and quantity) | Store data on Order Detail Record. Calculate shipping cost using shipping tables. Update customer total. Update item quantity on hand. | Items Purchased web page | Customer |
| Customer checks out | Customer | Clicks "Check Out" button on web page | Display Customer Order web page. | Verification web page | |
| Obtain customer payment | Customer | Credit card information | Verify credit card amount with credit card company. Send. | Credit card data Customer feedback | Credit card company Customer |
| Send customer email | | Temporal, hourly | Send customer an email confirming shipment. | | Customer |

Use-Cases and Data Flow Diagrams

Use Cases and Data Flow Diagrams

- The use-case concept (see Chapter #2) is used in creating DFD's
- Each use case defines one activity and its trigger, input, and output
- Use-cases allow the analyst to:
 - Work with users to understand the nature of the processes and activities
 - Create a single data flow diagram fragment
- Figure 7.14 shows a use-case form for the Internet storefront
 - The use-case form describes the Add Customer Item activity and it's:
 - Triggers
 - Input
 - output

Use Cases and DFD

- Figure 7.14 shows a use-case form for the Internet storefront
- The use-case form describes the Add Customer Item activity and it's:
 - Triggers
 - Input
 - output

FIGURE 7.14

A use case form for the Internet storefront describes the Add Customer Item activity and its triggers, input, and output.

| Use case name: Add Customer Item | | Process ID: 3 | |
|---|----------|---|-------------|
| Description: Adds an item for a customer Internet order. | | | |
| Trigger: Customer places an order item in the shopping basket. | | | |
| Trigger type: External <input checked="" type="checkbox"/> Temporal <input type="checkbox"/> | | | |
| Input Name | Source | Output Name | Destination |
| Item Purchased (Item Number and Quantity) | Customer | Items Purchased Confirmation Web Page | Customer |
| | | | |
| | | | |
| Steps Performed | | Information for Steps | |
| 1. Find Item Record using the Item Number. If the item is not found, place a message on the Items Purchased web page. | | Item Number, Item Record | |
| 2. Store item data on Order Detail Record. | | Order Detail Record | |
| 3. Use the Customer Number to find the Customer Record. | | Customer Number, Customer Record | |
| 4. Calculate Shipping Cost using shipping tables. Using the Item Weight from the Item Record and the Zip Code from the Customer Record, look up the Shipping Cost in the Shipping Tables. | | Zip Code, Item Weight, Shipping Table | |
| 5. Modify the Customer Total using the Quantity Purchased and the Item Price. Add the Shipping Cost. Update the Customer Record. | | Item Record, Quantity Purchased, Shipping Cost, Customer Record | |
| 6. Modify the Item Quantity on Hand and update the Item Record. | | Quantity Ordered, Item Record | |

Partitioning and Data Flow Diagrams

Partitioning and 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
- The reasons for partitioning:
 - Different user groups
 - Timing
 - Similar tasks
 - Efficiency
 - Consistency of data
 - Security

Partitioning Websites

- Partitioning Websites provides a basis for:
 - Improving how humans use a website
 - Improving speed of processing
 - Improving the ease of maintaining the website
 - Keeping a transaction secure

A Data Flow Diagram Example

The Data Flow Diagram Example

- The following example is intended to illustrate the development of a DFD by selectively looking at each of the components explored earlier in this chapter
- This example is:
 - Called “World’s Trend Catalog Division”
 - Used to illustrate concepts covered Chapters #7
 - Aspects of the data flow example is used in Chapters 8 and 9
- The example and the figures demonstrate the data flow concepts and DFD’s covered in this Chapter

Developing the List of Business Activities

List of Business Activities

- Developing a list of business activities for “World’s Trend” is shown in Figure 7.15
- The list can be developed using:
 - Information gathered using *interactive* and *unobtrusive* methods
- The list can be used to identify:
 - External entities
 - When used to develop the level 0 and child diagrams the list can be used to define *processes*, *data flows*, and *data stores*,



World's Trend is a mail order supplier of high-quality, fashionable clothing. Customers place orders by telephone, by mailing an order form included with each catalog, or via the website.

Summary of Business Activities

1. When customer orders come in, the item master and the customer master files are both updated. If an item is out of stock, the inventory control department is notified.
2. If the order is from a new customer, a new record is created in the customer master file.
3. Picking slips are produced for the customer order and sent to the warehouse.
4. A shipping statement is prepared.
5. The process of shipping a customer order involves getting the goods from the warehouse and matching up the customer shipping statement, getting the correct customer address, and shipping it all to the customer.
6. The customer statement is generated and a billing statement is sent to a customer once a month.
7. An accounts receivable report is sent to the accounting department.

FIGURE 7.15

A summary of business activities for World's Trend Catalog Division.

Creating a Context Level Data Flow Diagram

The Context-Level DFD

- The creation of the context-level diagram follows the development of the list of business activities
- A context-level DFD is shown in Figure 7.16 which shows:
 - The *order processing system*
 - Five external entities (two separate entities called customer are the same)
 - The dataflows to and from the external entities

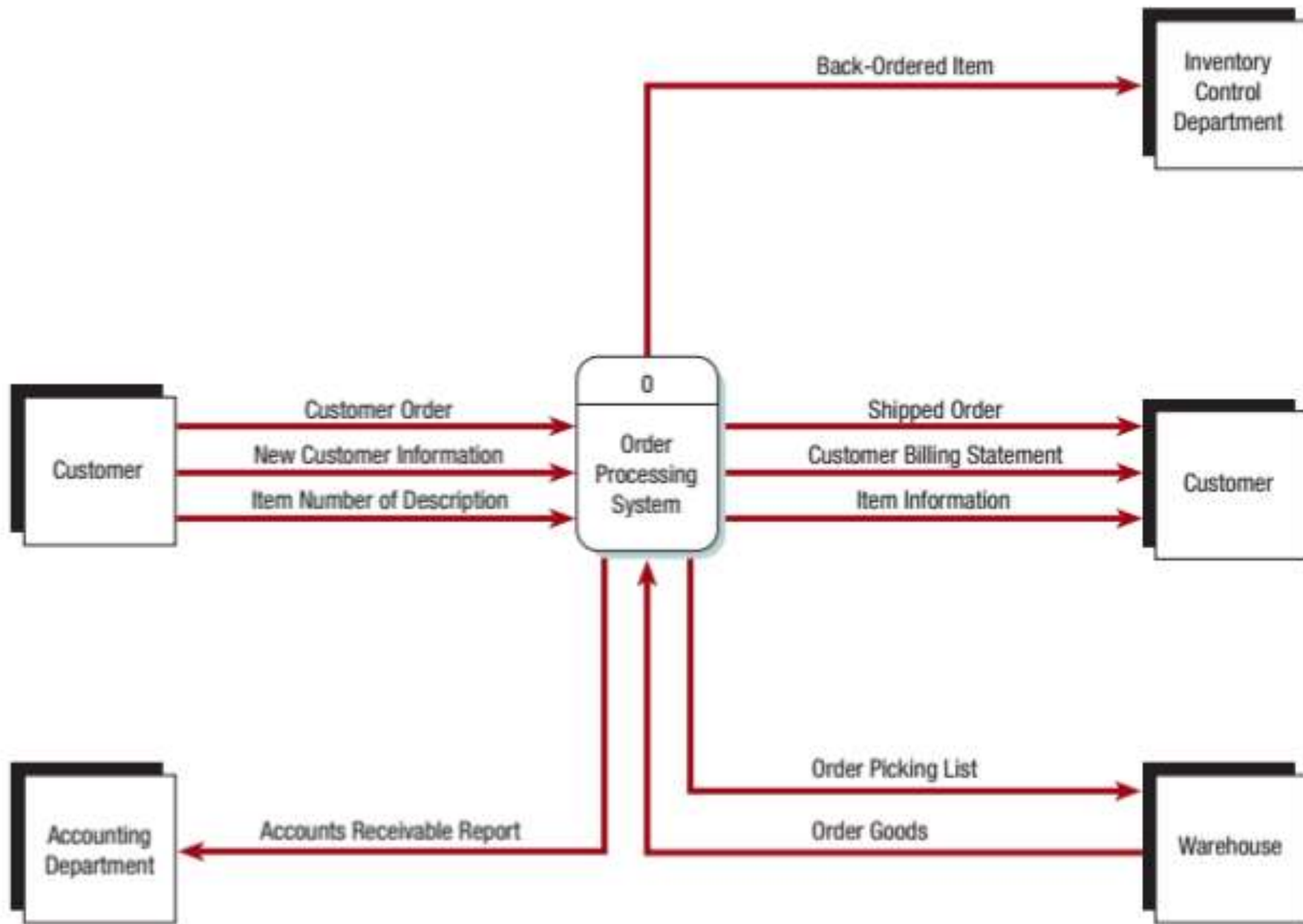


FIGURE 7.16

A context-level data flow diagram for the order processing system at World's Trend.

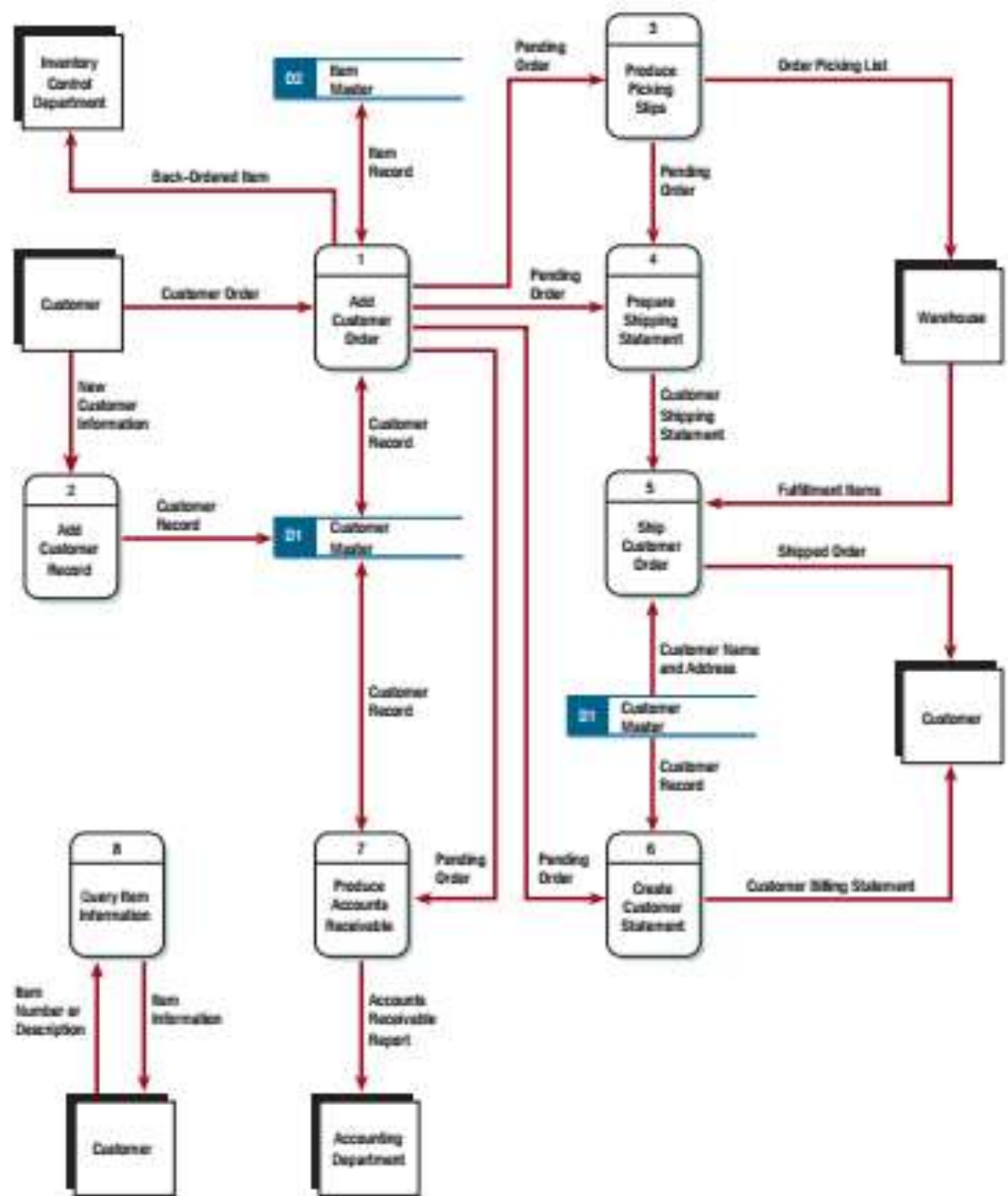
Drawing Diagram 0 (zero)

Diagram 0 (zero)

- The next step is the development of diagram 0
- The steps are:
 - Return to the business activity list and make a new list of all the processes and data stores
 - Draw a level 0 diagram (e.g., see Figure 7.17)
 - Call this diagram 0 (keep processes general to avoid excessive complication and complexity)
 - Further details can be added later
 - Take time to number the processes and data stores
 - Pay attention to make labels meaningful
 - Check for errors and make corrections (before the next step)

Diagram 0 (zero)

- Figure 7.17
- Diagram 0 (zero) of the order processing system for the “World’s Trend” catalogue division



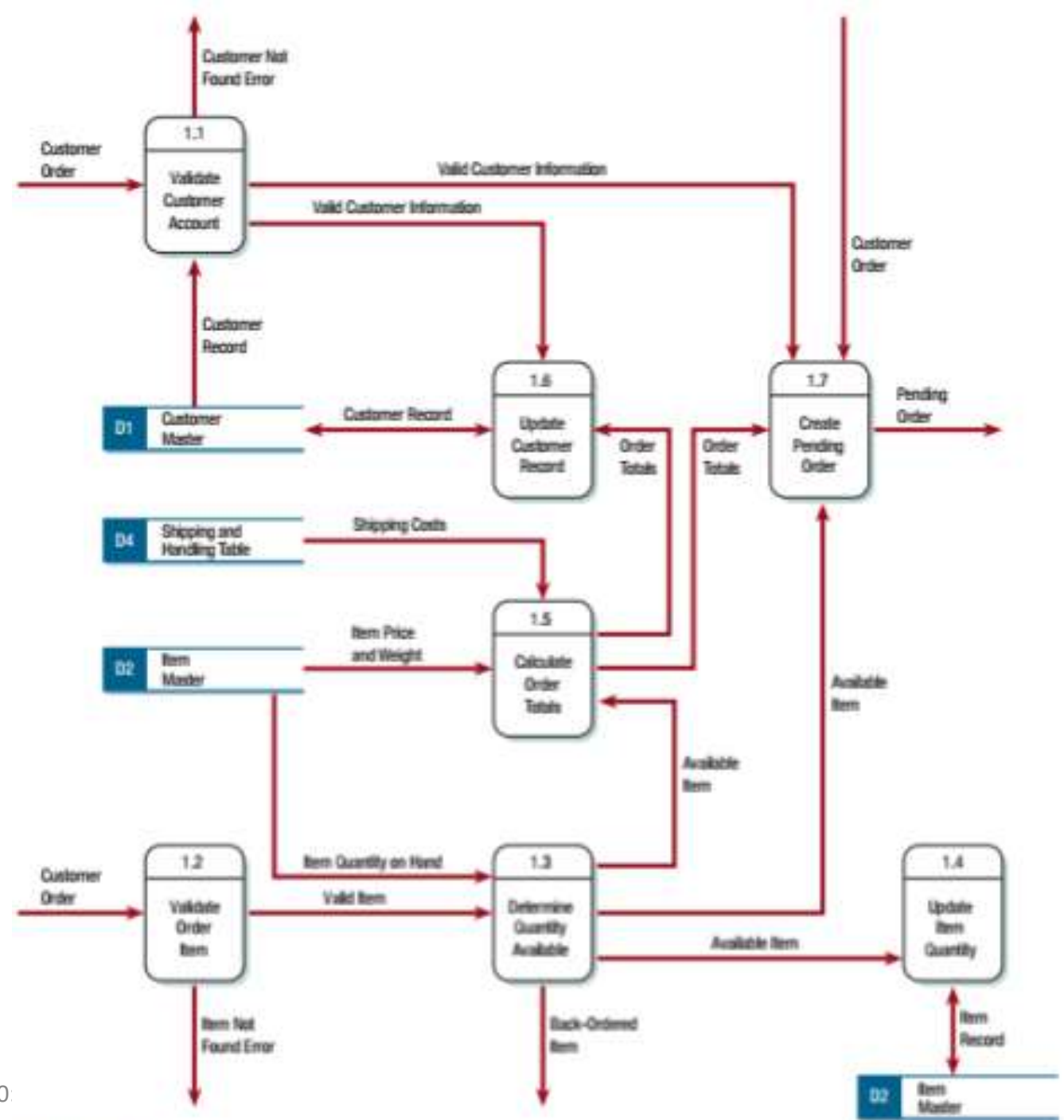
Creating a Child Diagram

The Child Diagram

- At this point in the development draw a child diagram (level 1 diagram)
 - Child processes are more detailed and show the logic required to produce the output
 - Child diagrams are numbered *Diagram 1*, *Diagram 2* etc in accordance with numbers assigned to each process in the level 0 diagram
 - When drawing a child diagram make a list of sub-processes
 - Connect these subprocesses to one another (and) to data stores when appropriate
 - Sub-processes may not be connected to external entities as the parent (level 0) DFD can be referred to
 - Label sub-processes 1.1, 1.2, 1.3, etc
 - Check for errors and make sure labels make sense (are meaningful)

The Child Diagram

- Figure 7.18
- Diagram 1 (child diagram) for the order processing system for “World’s Trend” catalog division



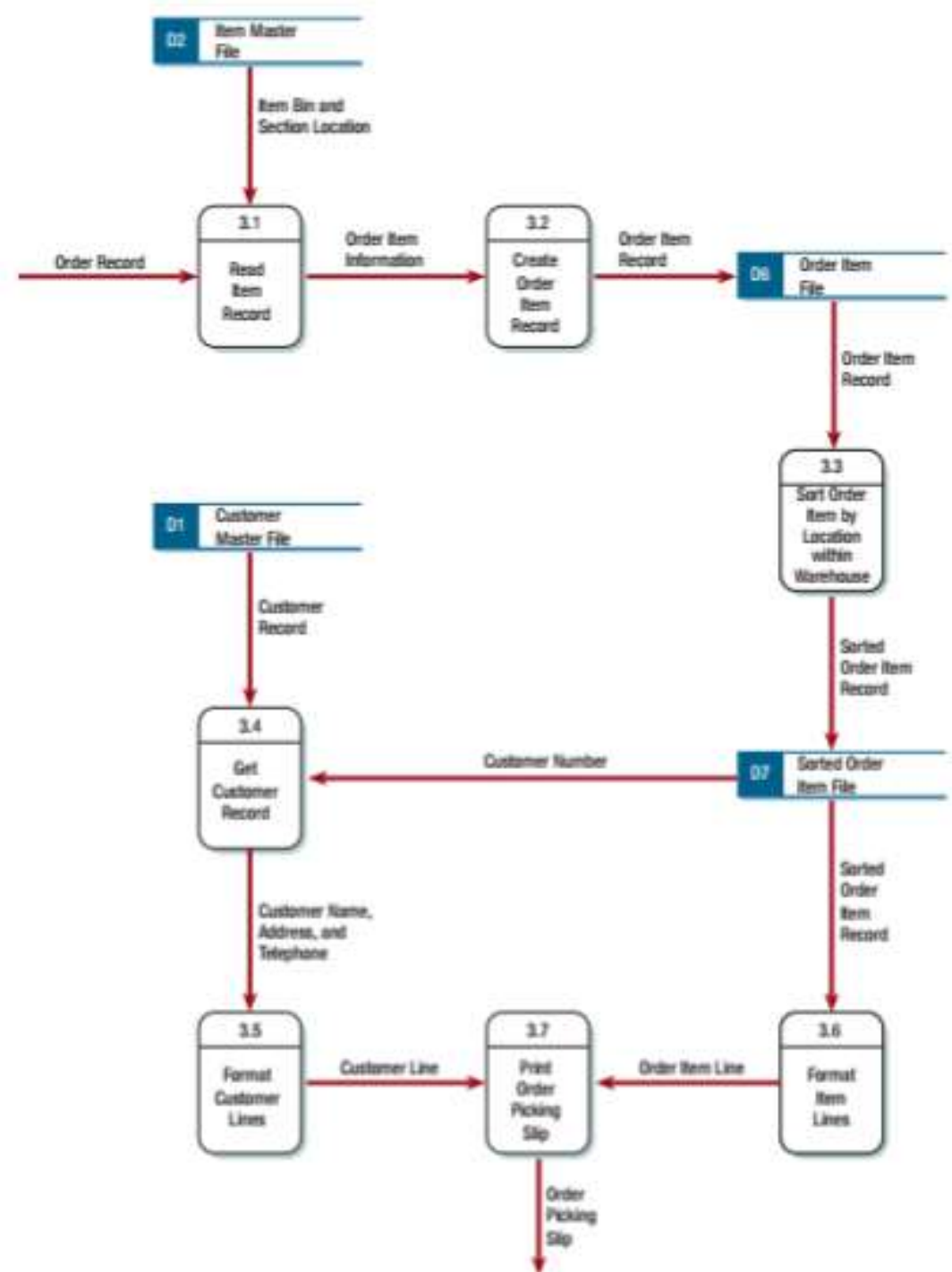
Creating a Physical Data Flow Diagram from The Logical Data Flow Diagram

The Physical DFD

- Extend the logical model and draw a physical model (see Figure 7.19:
 - An example of a physical data flow child diagram of process 3
- Physical DFD's provide a basis for describing processes in greater detail: for example in sub-process 3.3 you can identify processes for:
 - Scanning barcodes / displaying screens / locating records / creating and updating files
- The sequence of operations is important because:
 - The focus is on how the system functions and in what order events happen
- When data flows are described detail the actual form, report, or screen

The Physical DFD

- Figure 7.19
- A physical data flow child diagram for the “World’s Trend” catalogue division
- In Figure 7.19:
 - The ‘**SORT ORDER**’ is labelled
 - ‘**SORT ORDER ITEM BY LOCATION WITHIN WAREHOUSE**’



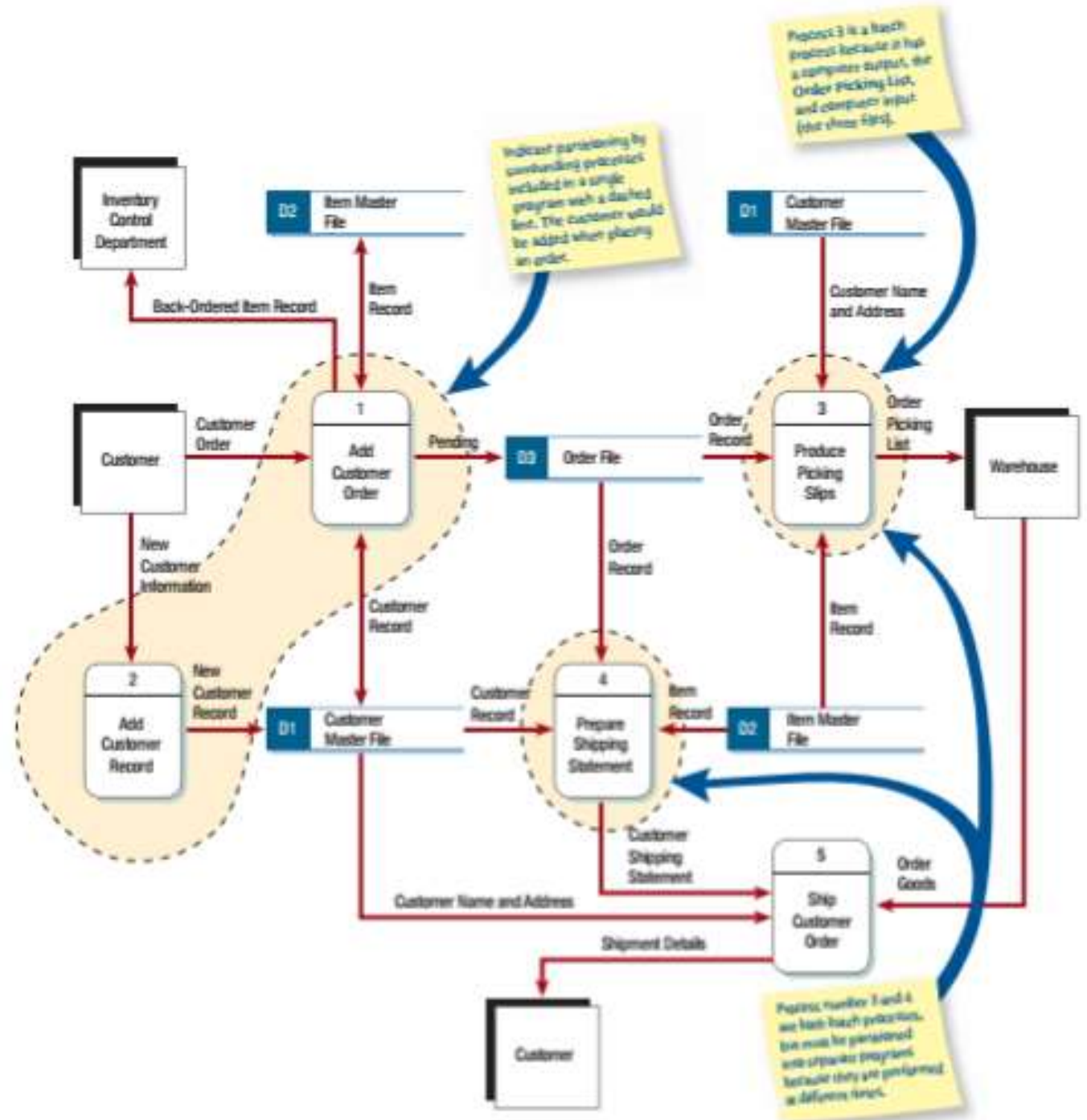
Partitioning the Physical Data Flow Diagram

Partitioning the Physical DFD

- The final step is partitioning the physical DFD
- This process involves combining (or) separating processes
- Figure 7.20 shows the usefulness of partitioning for the “World’s Trend” and shows:
 - The grouping of processes 1 and 2 would make sense to enable the addition of new customers when the first order is placed
 - Processes 3 and 4 can then be placed in two separate partitions
 - These processes must be carried out at different times (these processes cannot be grouped)
- The “Worlds Trend” example is used in Chapters #8 and #9

Partitioning

- Figure 7.20
- This figure shows the partitioning of a physical DFD
- We can see the partitioning the data flow diagram (showing part of Diagram 0).



Partitioning Websites

Partitioning Websites (1)

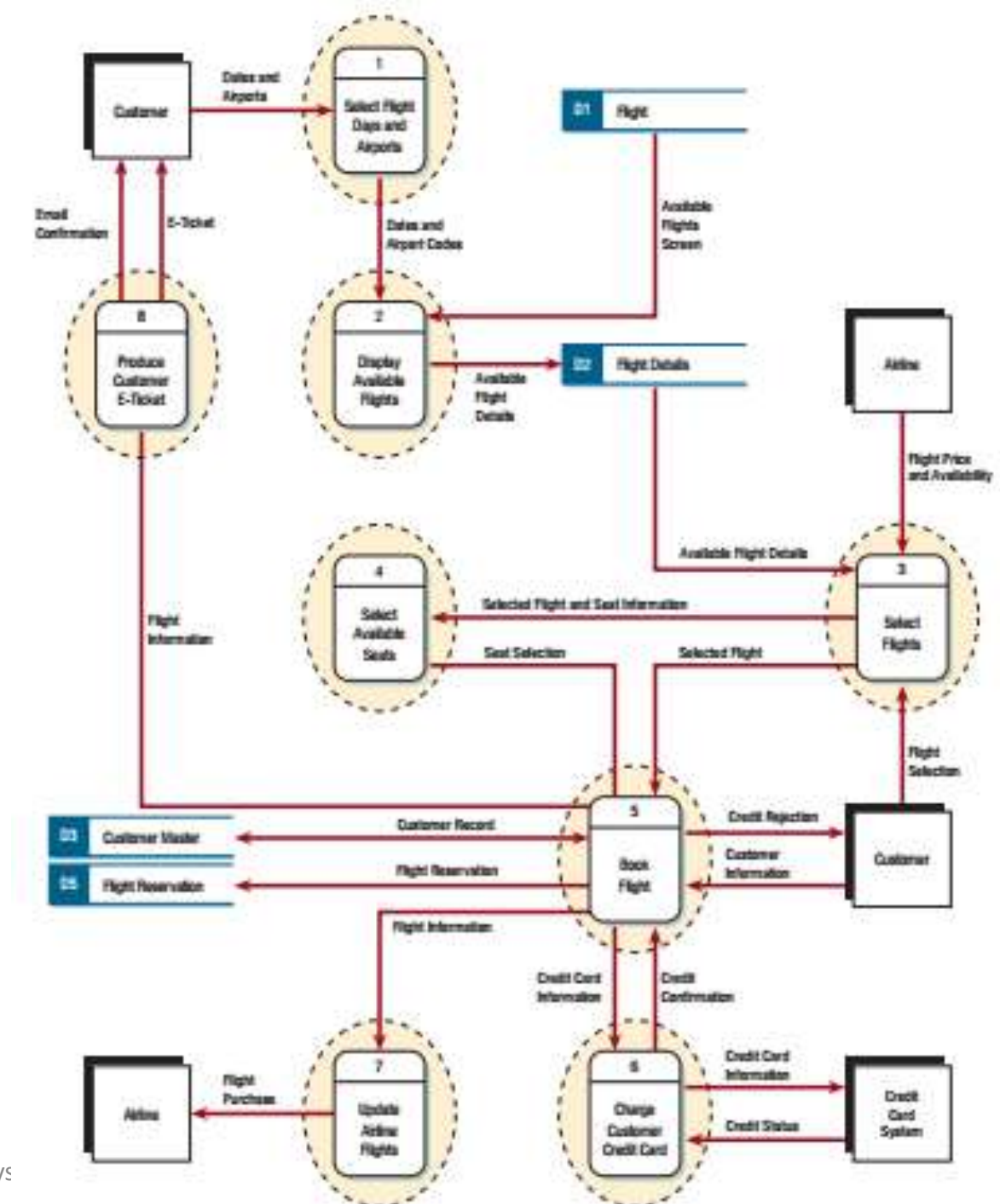
- Partitioning Websites is a useful principle in website design
- Designers using forms to collect data may opt to divide a website into multiple web pages as this can:
 - Improve the way humans use the site
 - Improve the speed of processing
 - Improve the efficacy in maintaining the site
 - Keep the transaction secure (use a unique web form and DFD process to validate and process data)
- Designers may also use Ajax:
 - Send a request to the server and obtain a small amount of data (or) an XML document returned to the same page
 - However, several pages should be created as required

Partitioning Websites (2)

- An example of partitioning is a web-based travel booking site
 - The focus is on the airline booking part of the system
 - Figure 7.21 shows this process
- Note: the designer has created several processes and unique partitions in making a flight reservation
 - For example process 1 receives and validates the dates and airports entered by the customer (or the booking agent)
 - The selection data is used to obtain flight details and create a transaction data store of flight details that match the flight request

Partitioning Websites

- Figure 7.21
- Procedures must be partitioned into a series of interacting processes:
 - Each process will have a corresponding web page or interaction with an external system



Communicating Using Data Flow Diagrams

Communicating Using Data Flow Diagrams

- DFD's are useful throughout the analysis and design process
- Use original (unexploded) DFD's early in identifying information requirements:
 - At this stage DFD's provide an overview of data flows with a visual perspective not available in narrative data
- While the logic of the data flows may be clear (to analysts) DFD's correctly drawn and labelled (use meaningful labels) provide clarity to users
- Finally, DFD's document a system and remember:
 - Data flows will persist longer than designers
 - DFD's document 'high' or 'low' levels of analysis and help to state clearly the logic underlying the data flows in an organization

Summary

Chapter #7 Summary (1)

- In this chapter we have considered:
 - Developing data flow diagrams (DFD)
 - Creating *context* diagrams
 - Drawing diagram 0 (zero)
 - Creating *child* diagrams
 - Checking diagrams for errors
 - Logical and physical DFD
 - Developing *logical* DFD
 - Developing *physical* DFD
 - Event modelling and DFD
 - Use-cases and DFD
 - Partitioning DFD

Chapter #7 Summary (1)

- In this chapter we have introduced a data flow example:
 - A data flow example
 - Developing a list of business activities
 - Creating a context-level DFD
 - Drawing diagram 0
 - Creating a child diagram
 - Creating a physical DFD from a logical DFD
 - Partitioning the physical DFD
 - Partitioning websites
 - Communicating using DFD's

Chapter #7 Summary (1)

- In this chapter we have introduced:
 - Data flow diagrams
 - Structured analysis and design tools that allow the analyst to comprehend the system and subsystems visually as a set of interrelated data flows
 - DFD symbols
 - Rounded rectangle
 - Double square
 - An arrow
 - Open-ended rectangle
 - Creating the *logical* DFD
 - Context-level data flow diagram
 - Level 0 logical data flow diagram
 - Child diagrams

Chapter #7 Summary (2)

- In this chapter we have considered:
 - Creating the *physical* DFD
 - Create from the *logical* data flow diagram
 - Partitioned to facilitate programming
- Partitioning data flow diagrams
 - Whether processes are performed by different user groups
 - Processes execute at the same time
 - Processes perform similar tasks
 - Batch processes can be combined for efficiency of data
 - Processes may be partitioned into different programs for security reasons

Case Studies