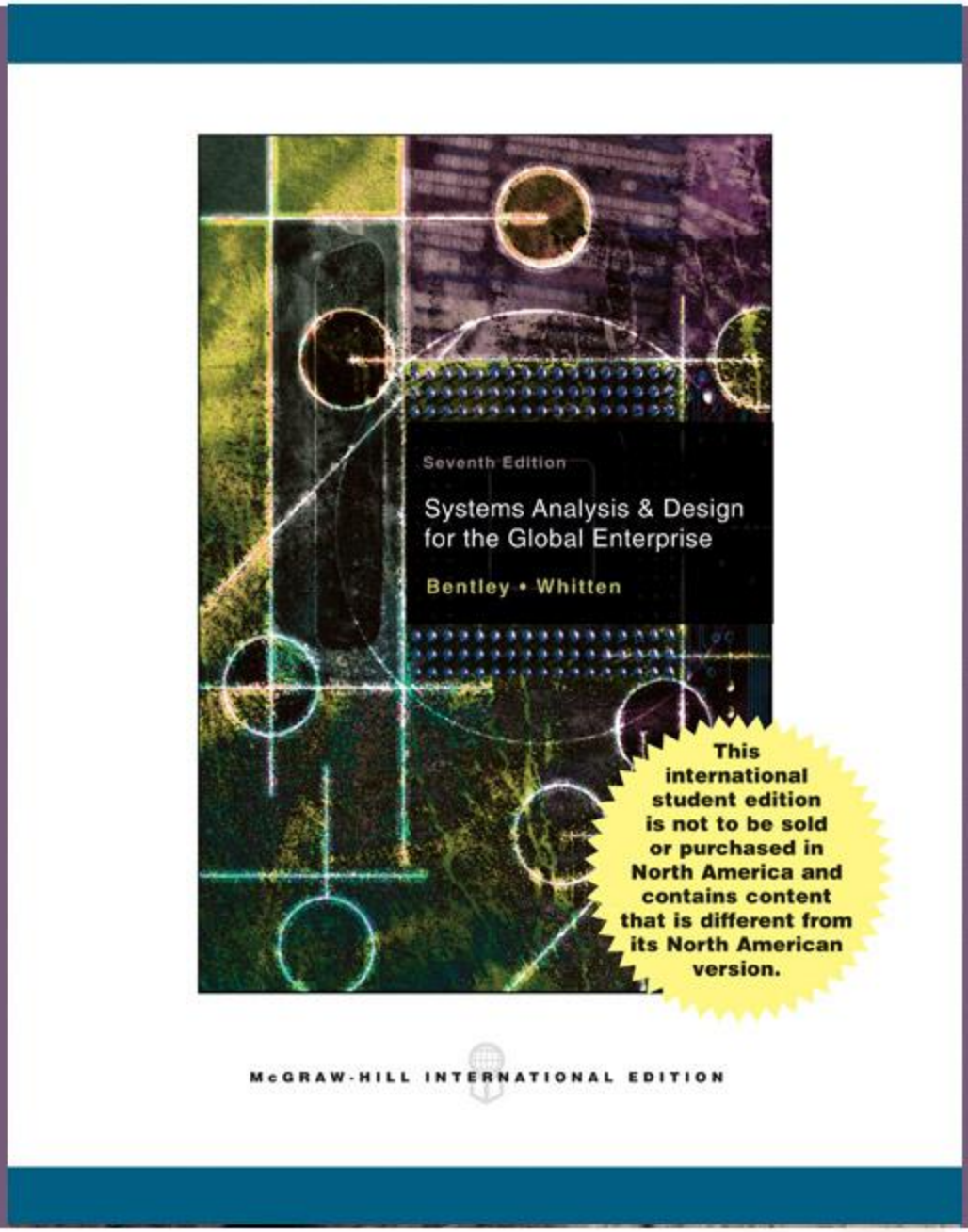

Seventh Edition
Systems Analysis & Design for the Global Enterprise
Bentley • Whitten
This international student edition is not to be sold or purchased in North America and contains content that is different from its North American version.
McGRAW-HILL INTERNATIONAL EDITION

Objectives

- **Describe the design phase tasks in terms of a computer-based solution for an in-house development project.**
- **Differentiate between logical and physical data flow diagrams, and explain how physical data flow diagrams are used to model an information system's architecture.**

Objectives

- Describe database and data distribution alternatives for system design.
- Draw physical data flow diagrams for an information system's architecture and processes.

12.1 System Design

Systems design – the specification of a detailed computer-based solution.

- Also called **physical design**.
- systems analysis emphasizes the business problem
- systems design emphasizes the technical or implementation concerns of the system.

Goal of the design phase: twofold

- The analyst seeks to design a system that both fulfills requirements and will be friendly to its end users.
- The analyst seeks to present clear and complete specifications to the computer programmers and technicians.


THE BUSINESS AND TECHNICAL COMMUNITY
(approval to continue project after decision analysis phase)
Approved System Proposal
SYSTEM OWNERS AND USERS (OR STEERING COMMITTEE)
Revised Project Plan
Facts, Recommendations, and Opinions
5.1
Design the Application Architecture
Application Architecture and Distribution Analysis
5.5
Update Project Plan
Application Schema
Revised Project Plan
Design Complete
Design Specifications
Repository
Database Schemas
5.2
Design the System Database
5.4
Package Design Specifications
User Interface Specifications
System Interface Specifications
5.3
Design the System Interface
Database, Input, and Output Specifications
Database Schemas

System Design Tasks For In-House Development (Build)

- **Design the Application Architecture**
  - Designing the application architecture involves considering network technologies and making decisions on how the systems' data, processes, and interfaces are to be distributed among the business locations.
  - Given the data models, process models and target solution, distribution decisions will need to be.
  - **PDFD**
- **Design the System Databases**
  - Database schema
  - Optimized for implementation DBMS
- **Design the System Interface**
  - Input, output, and dialogue specifications
  - Prototypes
- **Package Design Specifications**
  - Specifications to guide programmers
- **Update Project Plan**

模块独立

- 一、有效的模块化的软件比较容易开发出来。这是由于能够充分分割功能而且接口可以简化，当许多人分工开发同一软件时，这个优点尤其重要。

- 二、独立的模块比较容易测试和维护。

内聚和耦合

1. 耦合（coupling）：

耦合的强度依赖于以下几个因素：

（1）一个模块对另一个模块的调用；

（2）一个模块向另一个模块传递的数据量；

（3）一个模块施加到另一个模块的控制的多少；

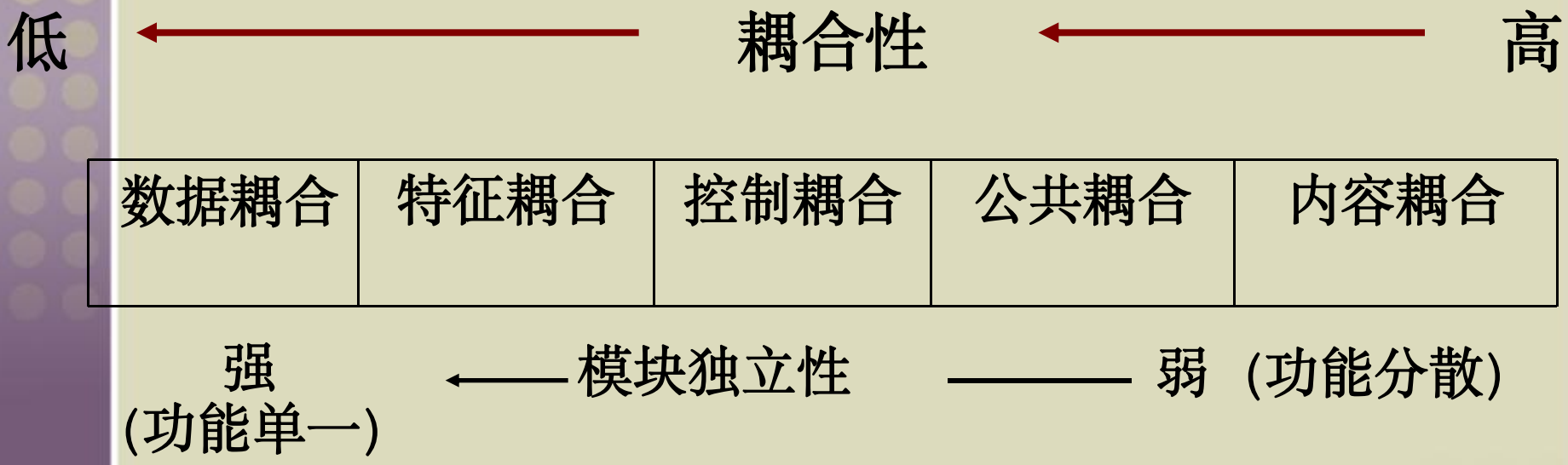
（4）模块之间接口的复杂程度。

耦合按从强到弱的顺序可分为以下几种类型：

（1）内容耦合　（2）公共耦合　（3）控制耦合

（4）标记耦合　（5）数据耦合

几种耦合方式的比较：



(1) 内容耦合

- 当一个模块访问另一个模块的内部数据
- 一个模块不通过正常入口而转入一个模块内部
- 两个模块有一部分程序代码重叠
- 一个模块有多个入口

直接修改或操作另一个模块的数据,或者直接转入另一模块时，就发生了内容耦合。此时，被修改的模块完全依赖于修改它的模块。

- 例如：模块A

  TRC： ……

  模块B

  GOTO TRC

模块A与模块B存在内容耦合，这是一种最坏的耦合。

```
class A {
    public int ma;   // ....}
class B {
    private A a = new A();
    public void methodA()
 {
        a.ma += 1; // 这里
    }
}
```

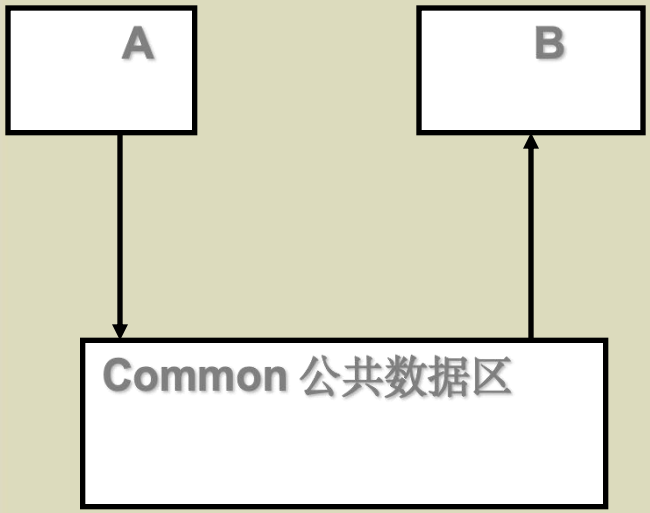
```
class A {
    private int ma; // public --
> private// ..
    //加 getter/setter
    public int getMa(){
        return this.ma;    }
    public void set(int ma) {
        this.ma = ma;    }}
class B {
    private A a = new A();
    public void methodA() {
     this.a.setMa(this.a.getMa()
                          + 1);
    }
}
```

(2) 公共耦合

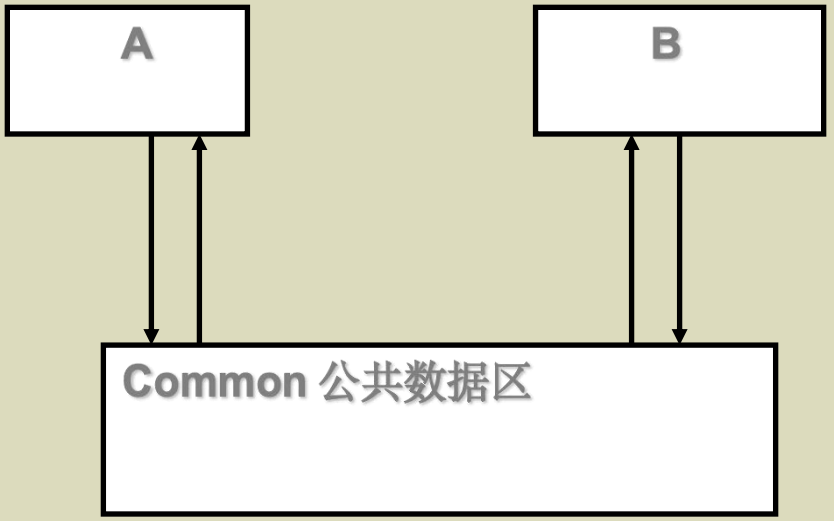
公共耦合：两个模块都要引用同一个公共数据域。

公共耦合是一种不良的耦合关系，它给模块的维护和修改带来困难。如公共数据要作修改，很难判定有多少模块应用了该公共数据，故在模块设计时，一般不允许有公共耦合关系的模块存在。

(2) 公共耦合



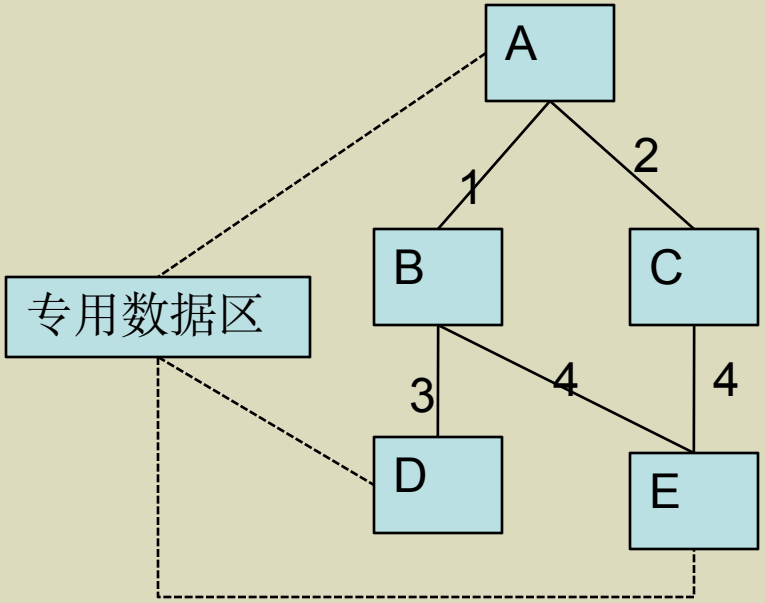
松散公共耦合



紧密公共耦合

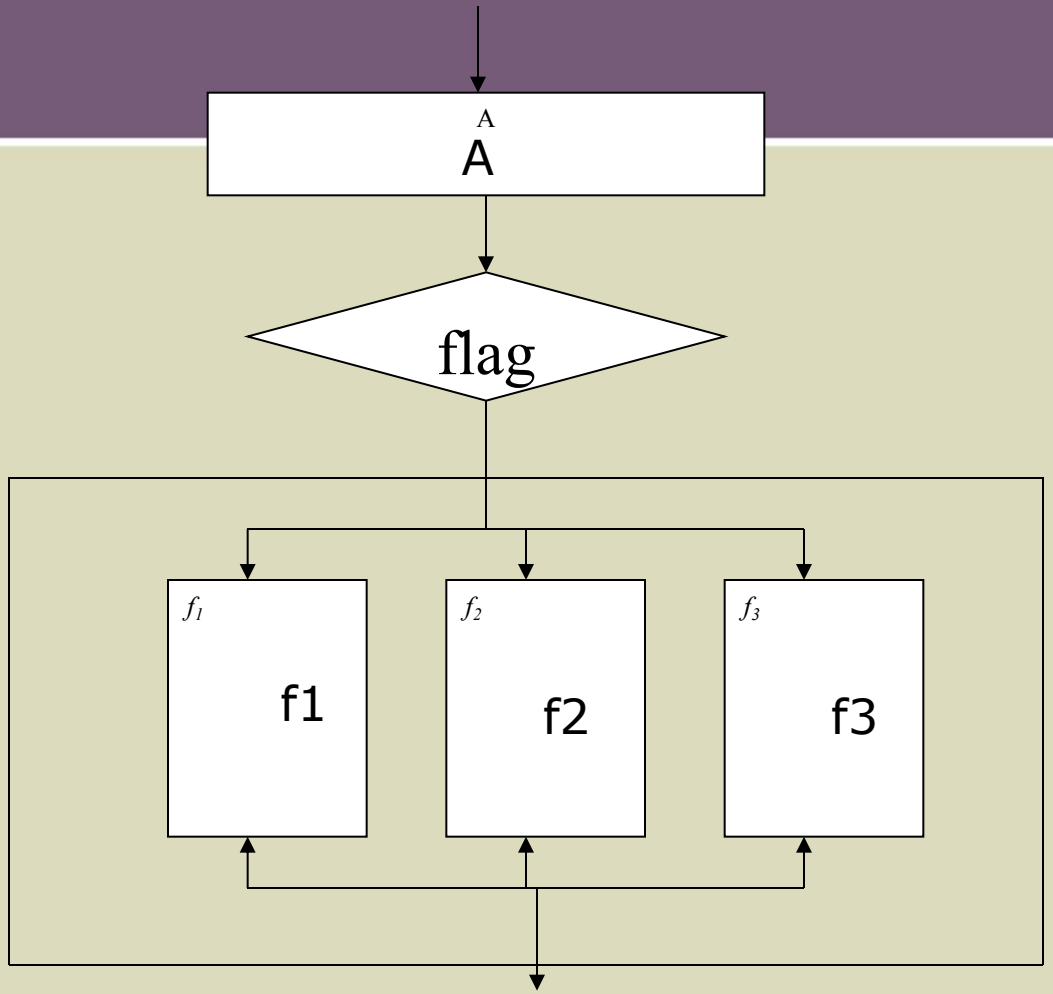
(2) 公共耦合

例. 下图中的程序有A, B、C、D、E五个模块组成，下表中描述了这些模块之间的接口，每个接口有一个编号。此外，模块A、D和E都要引用一个专用数据区。那么A和E之间的耦合关系是——。



(3) 控制耦合

- 一个模块调用另一个模块时，传递的是控制变量（如开关、标志等），被调模块通过该控制变量的值有选择地执行块内某一功能。
- 控制信息可以看作是一个开关量，它传递了一个控制信息或状态的标志。控制信息不同于数据信息，数据信息一般通过过程处理被处理的数据，而控制信息则是控制处理过程中的某些参数。
- 模块之间传递的不是数据信息



控制耦合

```
public int y;
Public A （string x）
{  if (x==”true”)
   {   y=1; }
   Else
     {   y=0; }
  }
```

```
Public void B ( )
    {
    if (y==1)
        {  F( );  // F( )是系统自定义的函数
        }
      Else
        { G( );  // G( )是系统自定义的函数
        }
    }
    。
```

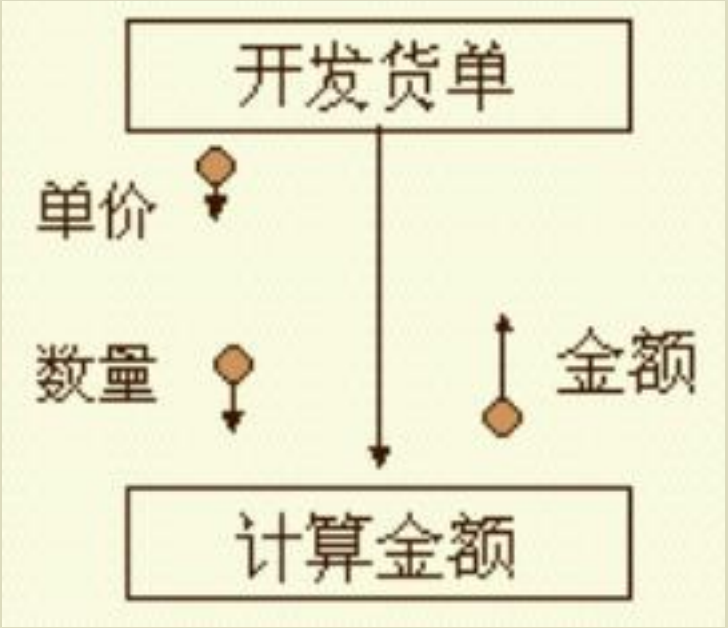
模块A&B之间为控制耦合因为两个模块间传递的y值是用作控制信号的开关量。改善方法就是把B模块调用的函数直接写入A模块中，然后删除B模块。

(4) 特征（标记）耦合

- 标记耦合指两个模块之间传递的是数据结构，如高级语言的数组名、记录名、文件名等这些名字即为标记，其实传递的是这个数据结构的地址。
- 一组模块通过参数表传递记录信息，就是标记耦合。

(5) 数据耦合

- 如果两个模块间彼此通过参数交换信息，而且交换的信息仅仅是数据，那么这种耦合称为数据耦合。这是模块之间影响最小的耦合关系,系统中必须存在这种耦合.



- sum(int a,int b)
  {int c;
  c=a+b;
  return(c);
  }

- main()
  {int x,y;
  ….
  printf("x+y=%d",su
  m(x,y));
  }

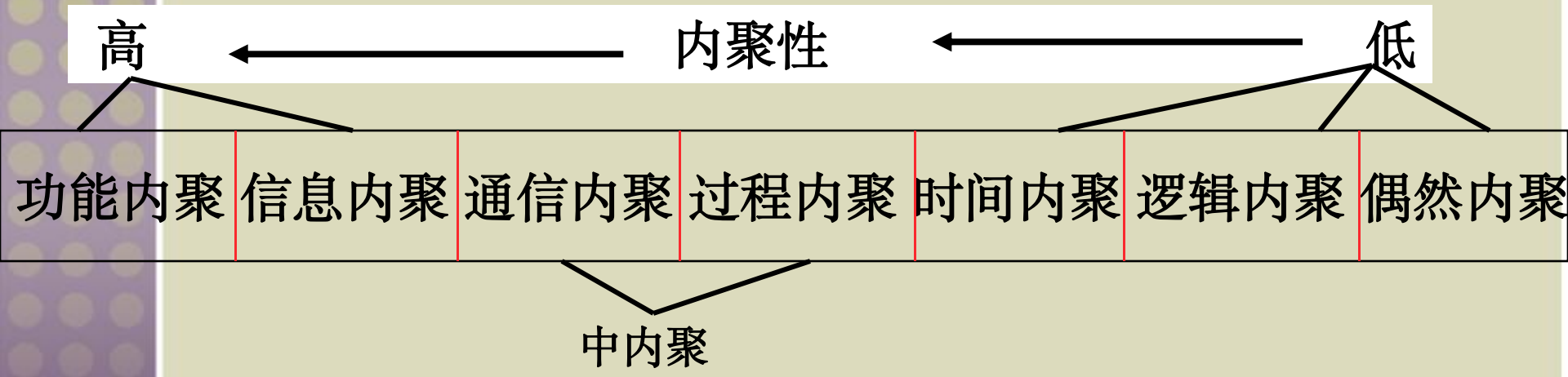
主函数与sum函数之间即为数据耦合关系

- 因此，如果模块间必须存在耦合，就尽量使用数据耦合，少用控制耦合，限制公共耦合的范围，坚决避免使用内容耦合。

2. 内聚（cohesion）

- 模块的内聚(cohesion)是指一个模块内部的各个组成部分的紧凑性，其处理动作的组合强度。
- 内聚有七种形式。

2. 内聚（cohesion）



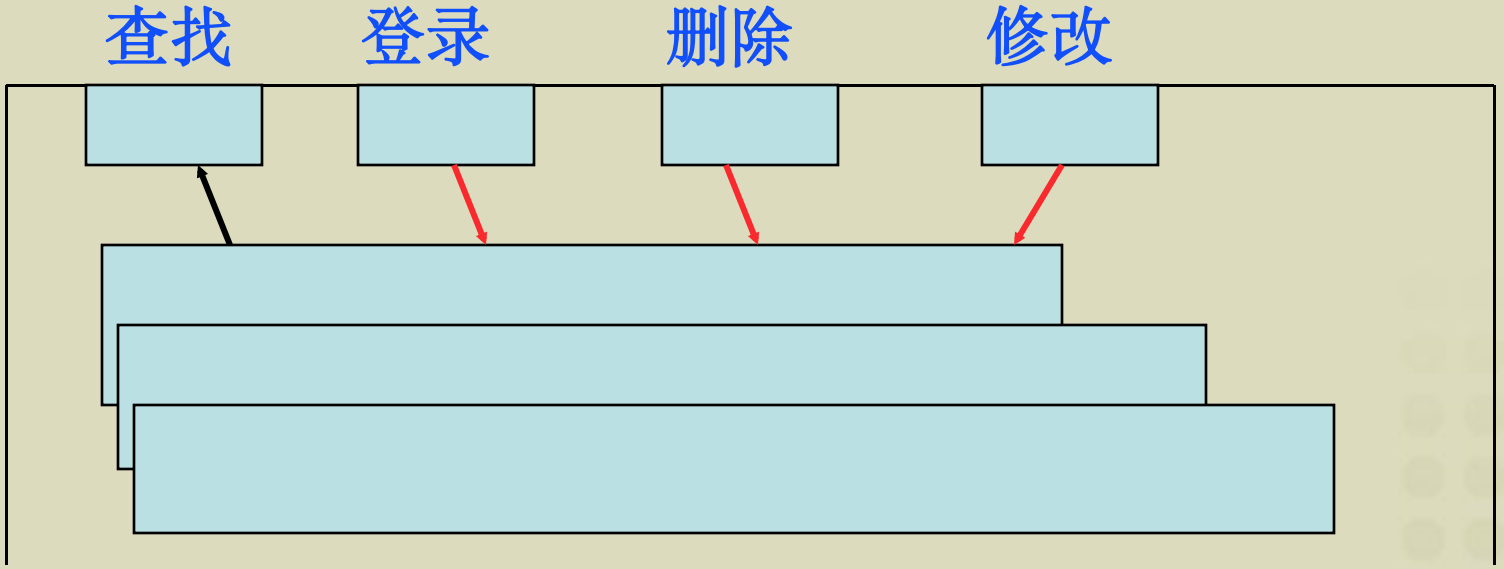
强 （功能单一） ← 模块独立性 → 弱 （功能分散）

A.功能内聚

- 如果一个模块内部的各组成部分的处理动作全都为执行同一个功能而存在，并且只执行一个功能，则称为功能内聚。
- 判断一个模块是不是功能内聚，只要看这个模块是“做什么”，是完成一个具体的任务，还是完成多任务。

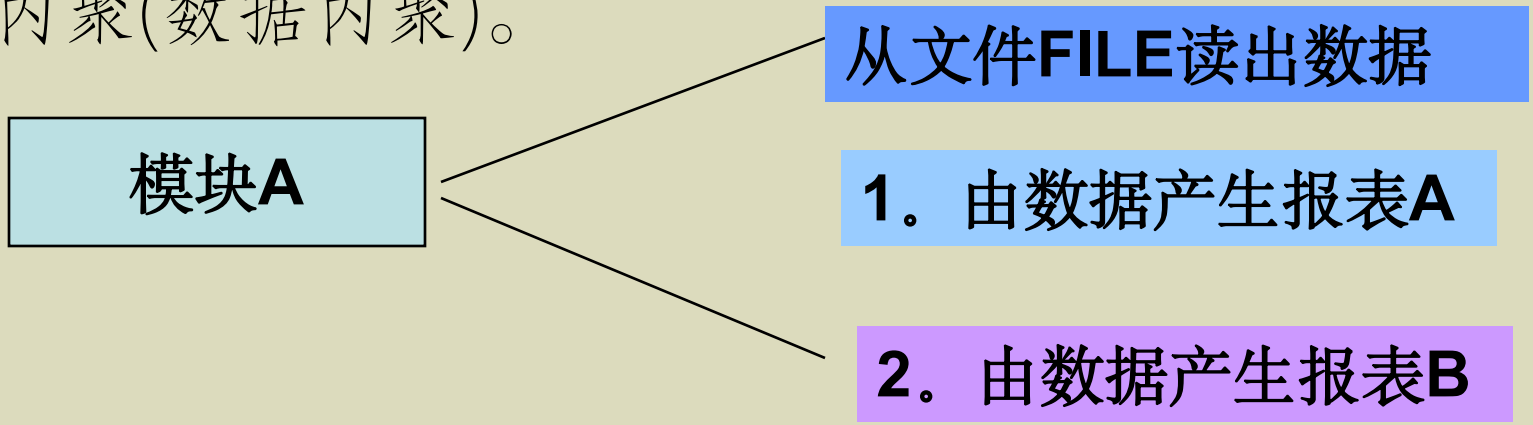
B.信息内聚

- 如果模块进行许多操作,每个都有各自的入口点,每个操作的代码相对独立,而且所有操作都在相同的数据结构上完成,则该模块具有“信息性内聚”。



C.通讯内聚

- 如果一个模块内各组成部分的处理动作都使用相同的输入数据或相同的输出数据，称为通讯内聚(数据内聚)。



例：模块A实现将传入的Date类型数据转换成String类型，以及将Date类型数据插入数据库，这两个操作都是对“Date类型数据”而言的。模块A中就是通信内聚。

D.过程内聚

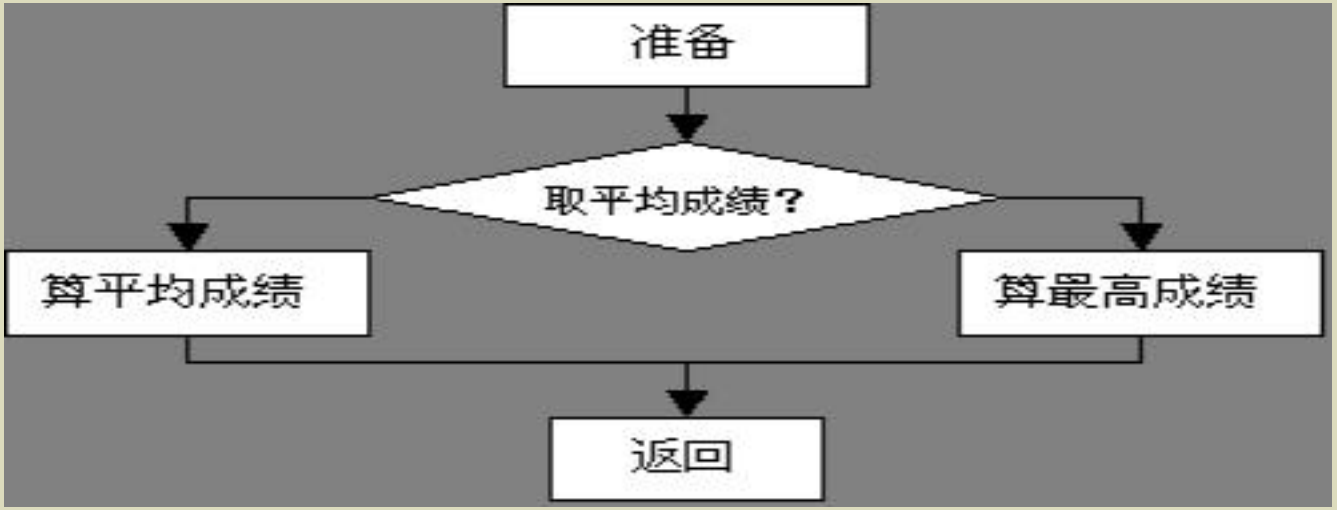
- 过程内聚：如果一个模块内的处理元素是相关的，而且必须以特定次序执行，则称为过程内聚；例如，把处理过程中的所有循环部分，判定部分和计算部分划分到一个模块，则它们都是过程内聚的，过程内聚的内部结构一般是由程序流程图直接演变出来的。

用户登陆了某某网站，A模块负责依次读取用户的用户名、邮箱和联系方式，这个次序是事先规定的，不能改变。模块A中就是过程内聚。

转换成功能内聚？

E. 逻辑内聚

- 逻辑内聚：这种模块把几种逻辑上相关的功能组合在一起，由上层调用模块向它发出控制信号，在多个关联性功能模块中选择执行某一个功能。会增加修改难度。



E. 逻辑内聚

- 一个子程序将打印季度开支报告、月份开支报告和日开支报告。打印哪一个，将由传入的控制标志决定，这个子程序具有逻辑内聚性，因为它的内部逻辑是由输进去的外部控制标志决定的。
- 显然，这个子程序不是按只完成一项工作并把它做好的原则。 应建立三个子程序：一个打印季度报告，一个打印月 报告、一个打印日报告，改进原来的子程序，让它根据传送去控制标志来调用这三个子程序之 一。调用子程序将只有调用代码而没有自己的计算代码，因而具有功能内聚性。
- void fun(int a) 当a==1输出 “OK”当a==其它值时输出 “ERROR”。

F.时间内聚

- 如果一个模块完成的功能必须在同一时间内执行(如系统初始化)，但这些功能只是因为时间因素关联在一起，则称为时间内聚。
- **例如：紧急故障处理模块中的关闭文件、报警等任务都必须无中断地同时处理。**
- 编程开始时，程序员把对所有全局变量的初始化操作放在模块A中。模块A中就是时间内聚。

G.偶然（机械）内聚

- 模块的各成分之间没有关联，只是把分散的功能合并在一起。偶然（机械）内聚。
- 例：A模块中有三条语句（一条赋值，一条求和，一条传参），表面上看不出任何联系，但是B、C模块中都用到了这三条语句，于是将这三条语句合并成了模块A。模块A中就是偶然内聚。

13.1 Application Architecture

Application architecture – a specification of the technologies to be used to implement information systems. The blueprint to communicate the following design decisions:

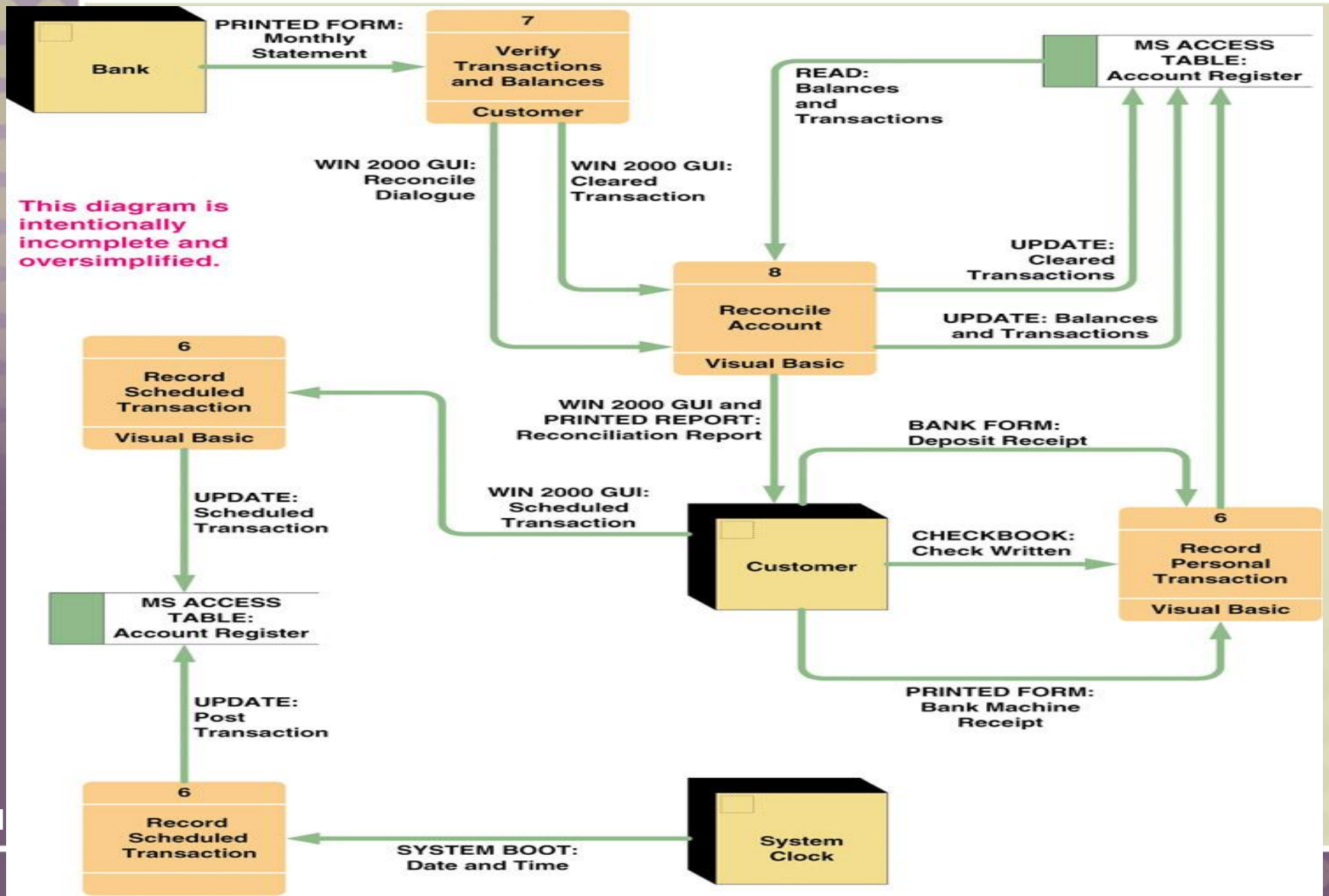
- The degree to which the information system will be centralized or distributed.
- The distribution of stored data.
- The implementation technology for software developed in-house.
- The integration of commercial off-the-shelf software.
- The technology to be used to implement the user interface.
- The technology to be used to interface with other systems

Physical Data Flow Diagram (PDFD)

Physical data flow diagram (PDFD) – a process model used to communicate the technical implementation characteristics of an information system.

- Communicate technical choices and other design decisions to those who will actually construct and implement the system.

Sample Physical Data Flow Diagram



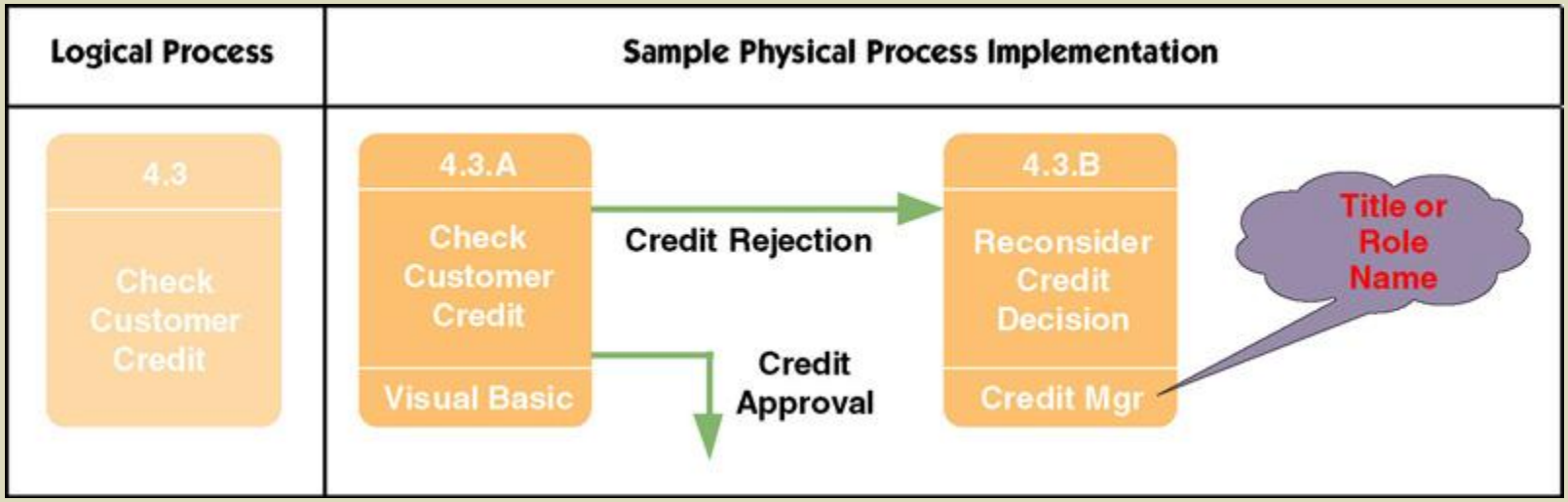
Physical Processes

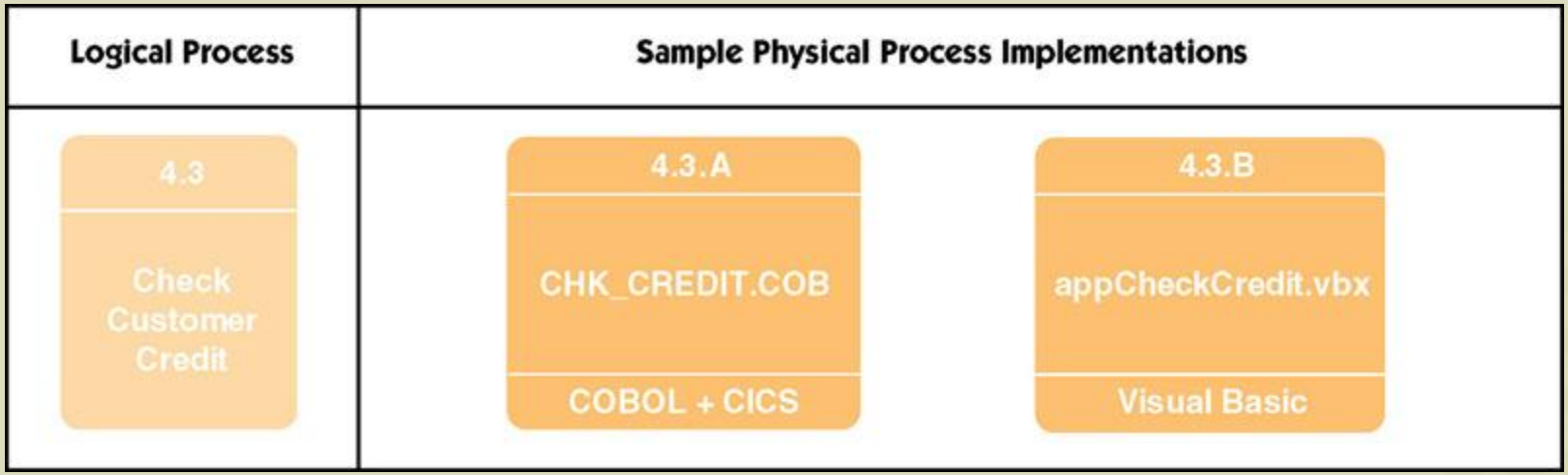
Physical process – either a *processor*, such as a computer or person, or a technical implementation of specific work to be performed, such as a computer program or manual process.

- Logical processes may be assigned to physical processors such as PCs, servers, people, or devices in a network. A physical DFD would model that network structure.
- Each logical process requires an implementation as one or more physical processes.
- A logical process may be split into multiple physical processes:
  - To define aspects performed by people or computers.
  - To define aspects implemented by different technologies.
  - To show multiple implementations of the same process.
  - To add processes for exceptions and security.

Physical Process Notation

Sample Physical Process Implementations





Physical Data Flows

A physical data flow represents:

- Planned implementation of an input to, or output from a physical process.
- Database command or action such as create, read, update, or delete.
- Import of data from, or export of data to another information system.
- Flow of data between two modules or subroutines (represented as physical processes).

Implementation method:
Data flow name

OR

Data flow name
(Implementation method)

Sample Physical Data Flows

Logical Data Flow	Implementation	Sample Physical Data Flow
	Computer Input (Keyboard)	
	Computer Input (Internet)	
	Computer Input (Keyless)	
	Computer Input (Batch File)	
	Computer Output (Printed)	
	Computer Output (On-Line)	
	Create a record in a database	

Sample Physical Data Flows (continued)

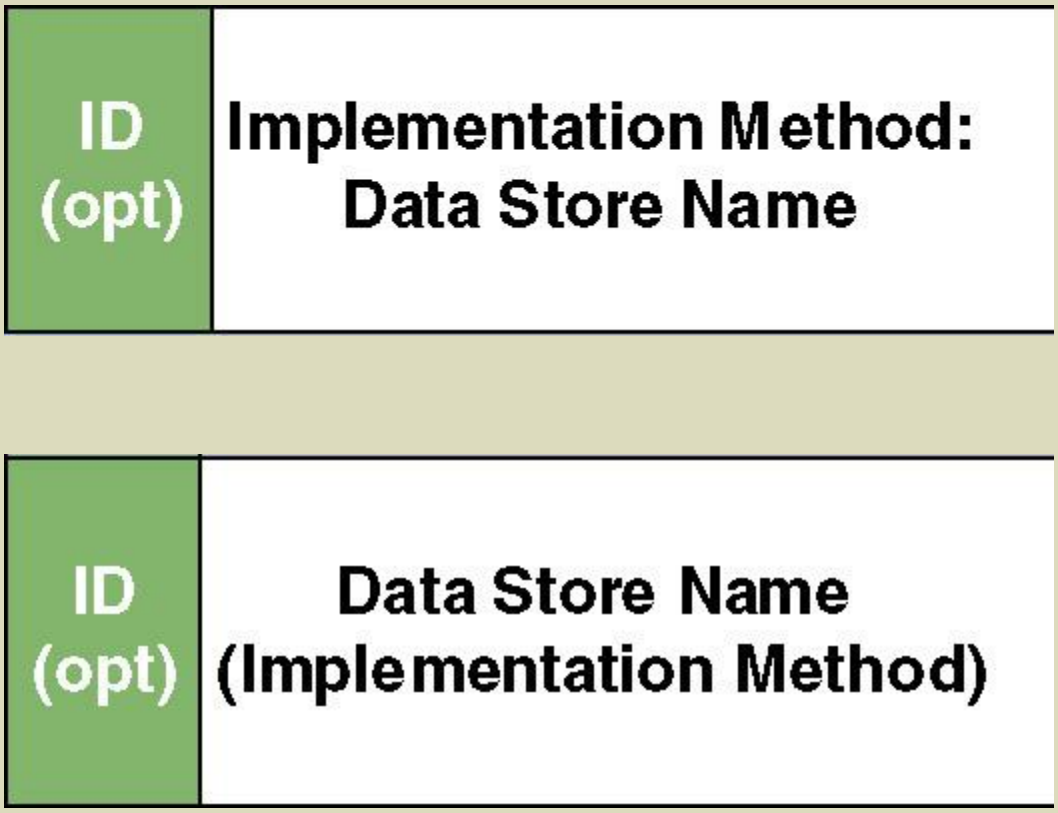
	Read records in a database	
	Update a record in a database	
	Delete a record in a database	
	Import a data file	
	Export a data file	
	Pass data between modules of a program	
	Pass a manual form	

Physical External Agents

Physical external agents are carried over from the logical DFD models.

- If scope changes, the logical models should be changed before the physical models are drawn.

Physical Data Store Notation



Physical Data Store Implementations

Logical Data Store	Implementation	Physical Data Store
Human Resources	A database (multiple tables)	Oracle : Human Resources DB
Marketing	A database view (subset of a database)	SQL Server: Northeast Marketing DB
Purchase Orders	A table in a database	MS Access: Purchase Orders
Accounts Receivable	A legacy file	VSAM File: Accounts Receivable
Tax Rates	Static data	ARRAY: Tax Table
Orders	An off-line archive	TAPE Backup: Closed Orders
Employees	A file of paper records	File Cabinet: Personnel Records
Faculty/Staff Contact Data	A directory	Handbook: Faculty/Staff Directory
Course Enrollments By Date	Archived reports (for reuse and recall)	REPORT MGR: Course Enrollment Reports

13.3 Distributed versus Centralized Systems

Distributed system – a system in which components are distributed across multiple locations and computer networks.

- Accordingly, the processing workload is distributed across multiple computers on the network.

Centralized systems – a system in which all components are hosted by a central, multi-user computer.

- Users interact with the system via terminals (or a PC emulating a terminal).
- Virtually all the actual processing and work is done on the host computer.

Computing Layers

- **Presentation layer**—the user interface
- **Presentation logic layer**—processing that must be done to generate the presentation, such as editing input data or formatting output data.
- **Application logic layer**—the logic and processing to support business rules, policies, and procedures
- **Data manipulation layer**—to store and retrieve data to and from the database
- **Data layer**—the actual business data

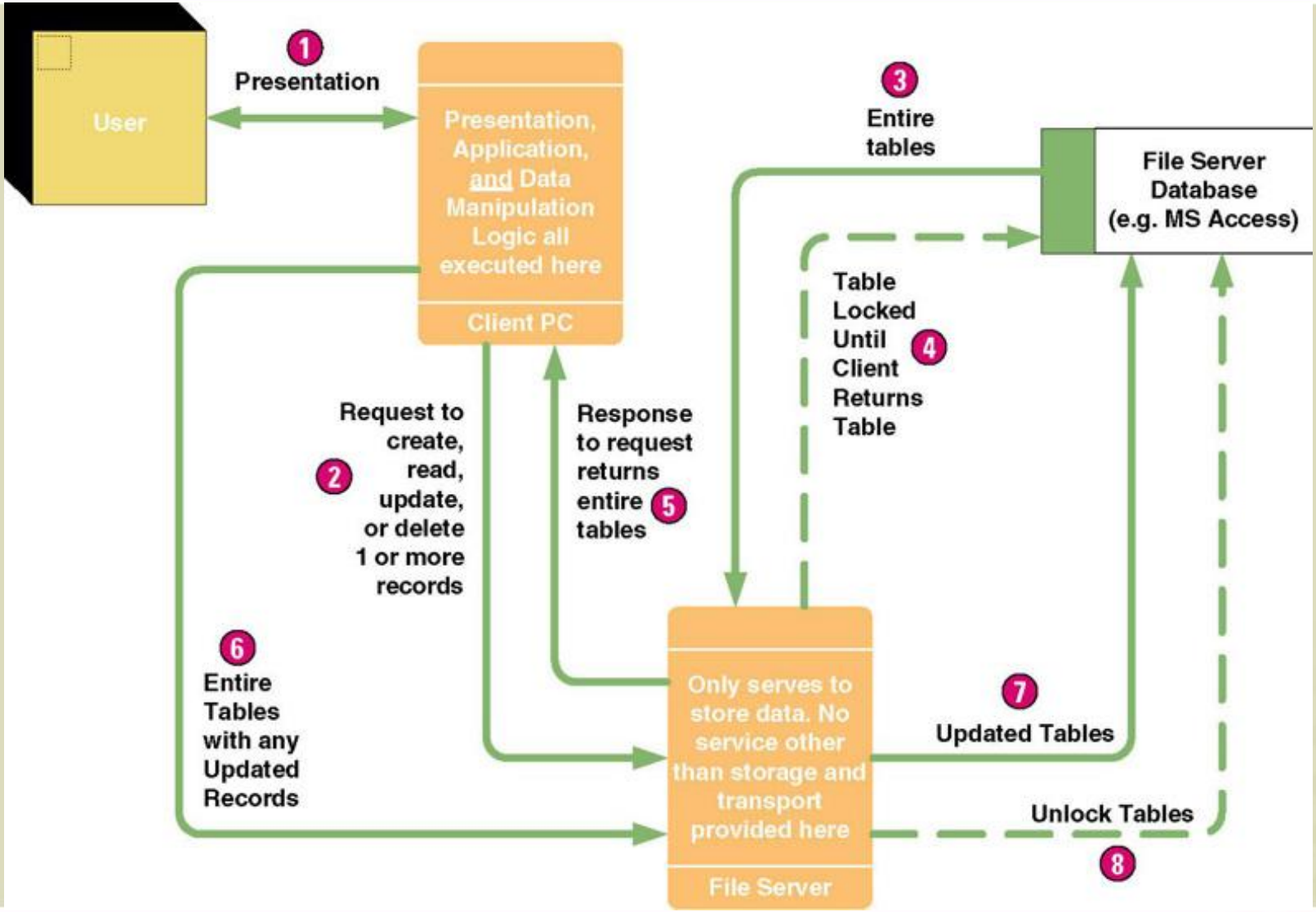
File Server Architecture

Local area network (LAN) – a set of client computers (PCs) connected over a relatively short distance to one or more servers.

File server system – a LAN in which a server hosts the data of an information system.

- All other layers are implemented on the client computers.
- Frequently excessive network traffic to transport data between servers and clients.
- Client must be fairly robust ("fat") because it does most of the work.
- Database integrity can be compromised.

File Server Architecture (自学)



Client/Server Architecture — Clients

Client/server system – a distributed computing solution in which the presentation, presentation logic, application logic, data manipulation, and data layers are distributed between client PCs and one or more servers.

Thin client – a personal computer that does not have to be very powerful because it only presents the user interface to the user.

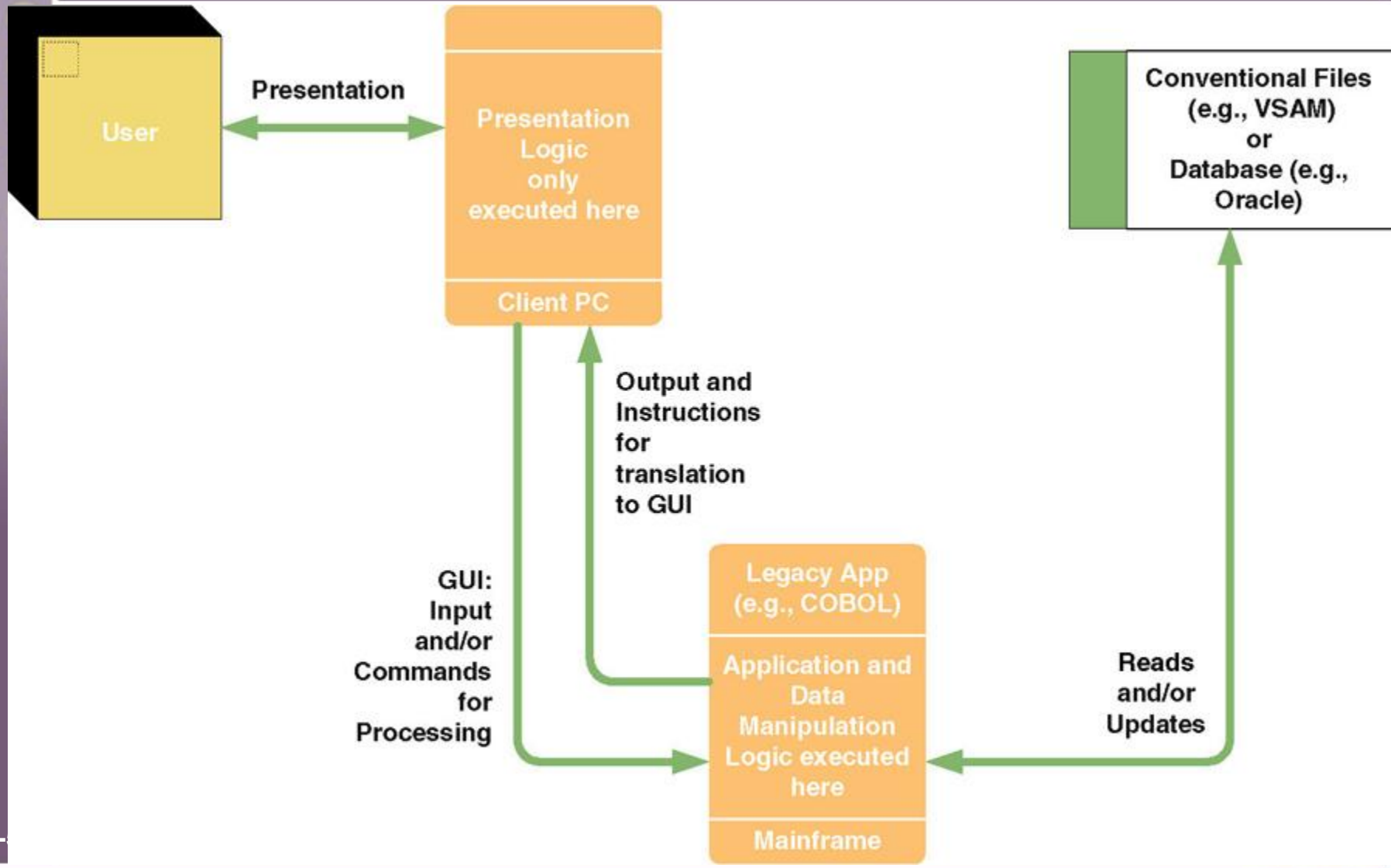
Fat client – a personal computer, notebook computer, or workstation that is typically powerful.

Client/Server—Distributed Presentation

Distributed presentation – a client/server system in which the presentation and presentation logic layers are shifted from the server to reside on the client.

- The application logic, data manipulation, and data layers remain on the server (frequently a mainframe).

Client/Server—Distributed Presentation

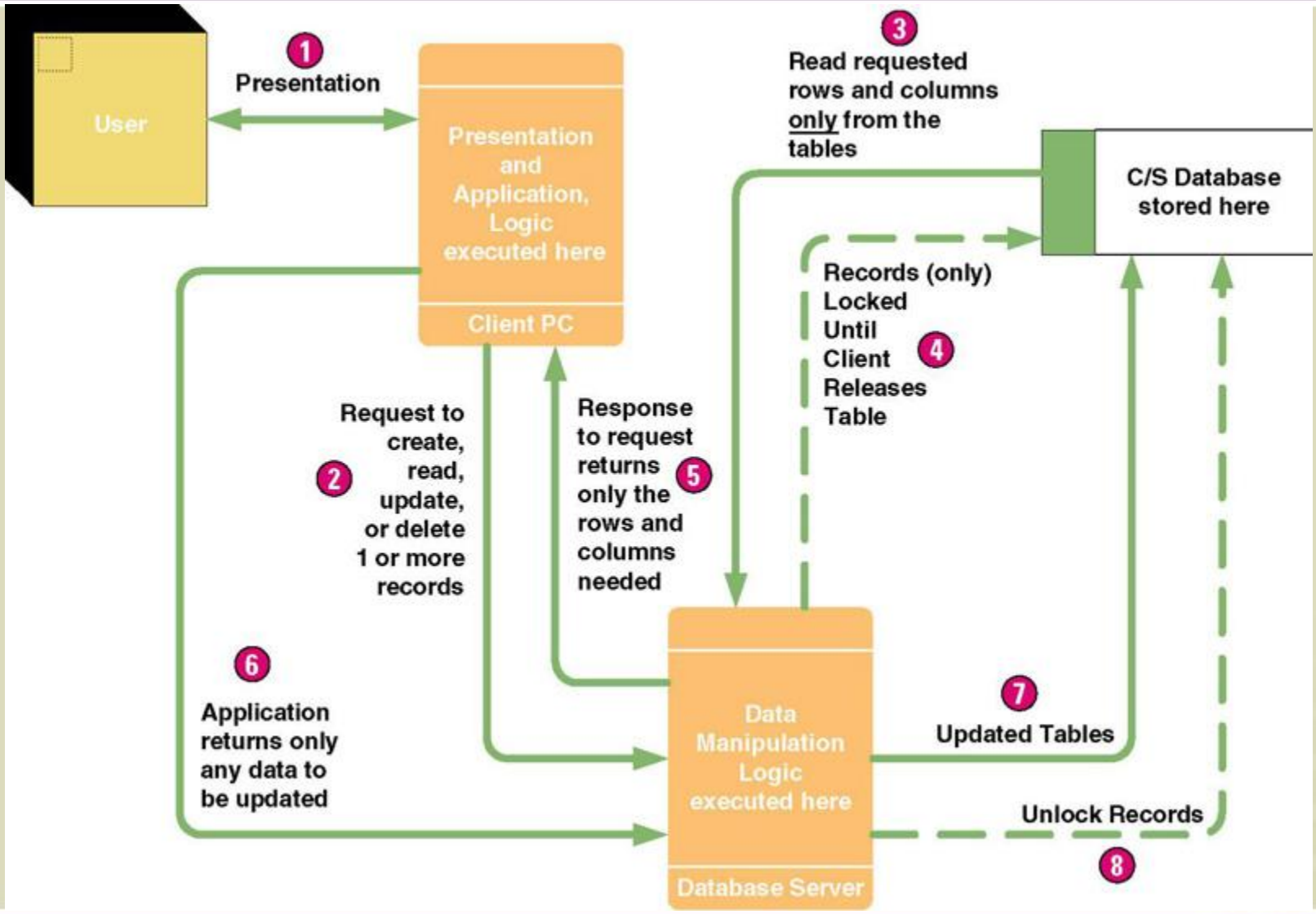


Client/Server—Distributed Data

Distributed data – a client/server system in which the data and data manipulation layers are placed on the server(s), and other layers are placed on the clients.

- Sometimes called two-tiered client/server computing.
- Difference to file server systems is where the data manipulation commands are executed.
- Much less network traffic than file server systems because only the database requests and the results of those requests are transported across the network.
- Database integrity is easier to maintain.

Client/Server—Distributed Data



Client/Server—Distributed Data and Application

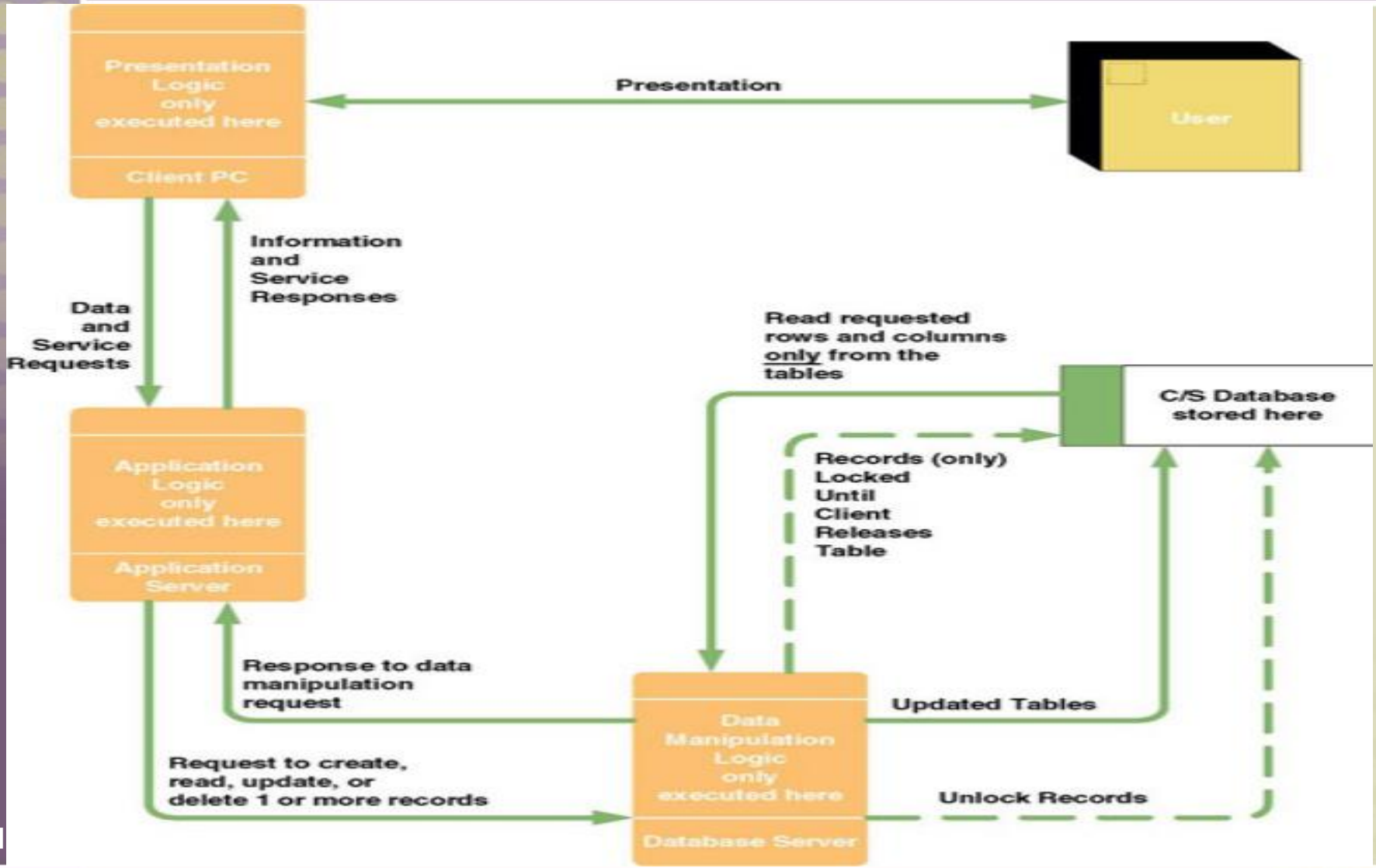
Distributed data and application – client/server system:

1. The data and data manipulation layers are placed on their own server(s),
2. The application logic is placed on its own server,
3. The presentation logic and presentation layers are placed on the clients.

- Also called **three-tiered** or **n-tiered client/server computing**.
- Requires design partitioning.

Partitioning – the art of determining how to best distribute (duplicate) application components across the network.

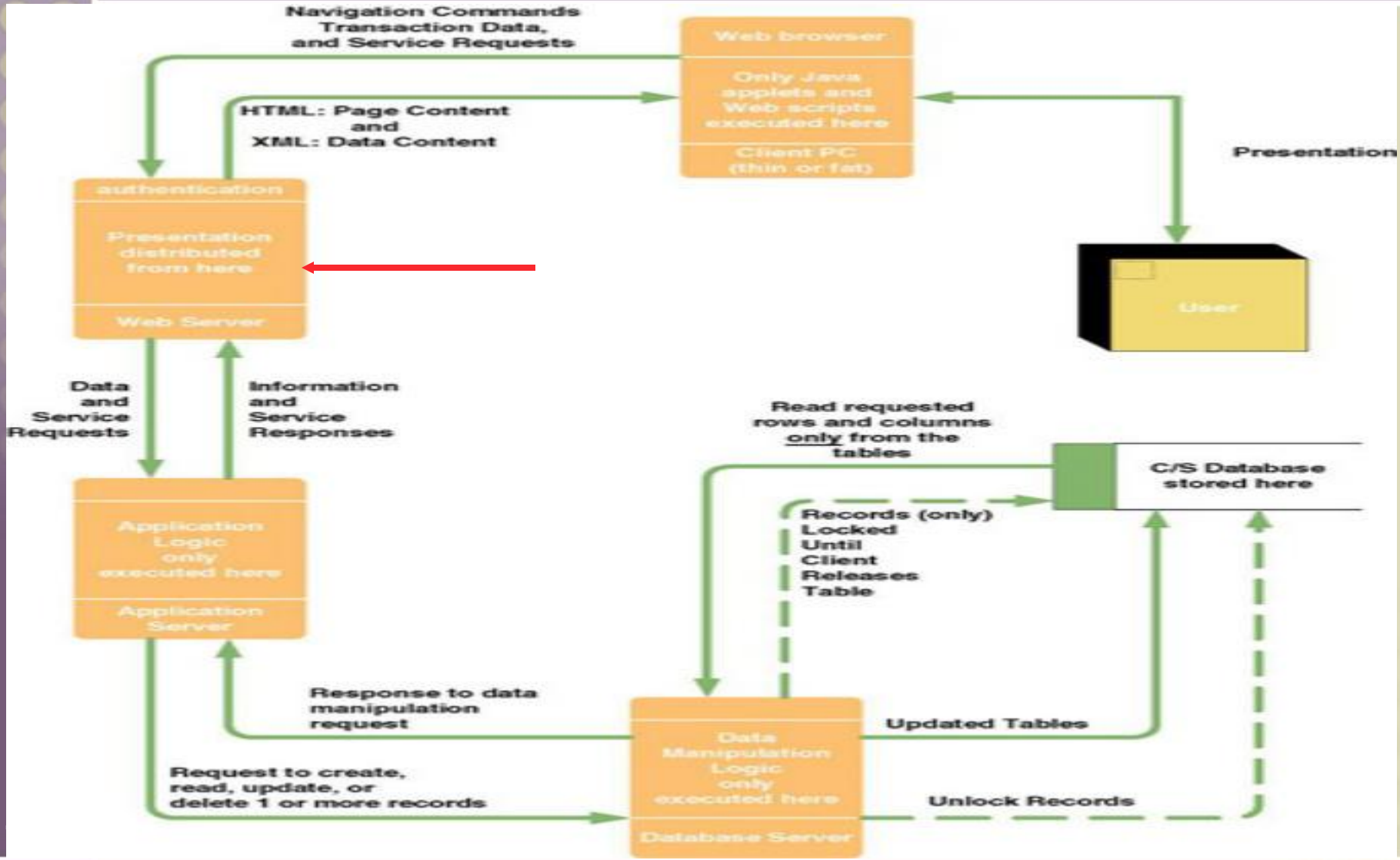
Client/Server — Distributed Data and Application



Client/Server Architecture — Servers

- **Database server** – a server that hosts one or more databases and executes all data manipulation commands at the server.
- **Transaction server** – a server that hosts services which ensure that all database updates for a transaction succeed or fail as a whole.
- **Application server** – a server that hosts application logic and services for an information system.
- **Messaging or groupware server** – a server that hosts services for e-mail, calendaring, and other work group functionality.
- **Web server** – a server that hosts Internet or intranet websites.

Network Computing System: Internet/Intranet



13.3.2 Data Architectures

Relational database stores data in tabular form. Each file is implemented as a table. Each field is a column in the table. Related records between two tables are implemented by intentionally duplicated columns in the two tables.

Distributed relational database – A database system that duplicates tables to multiple database servers located in geographically important locations.

Distributed relational database management system – a software program that controls access to and maintenance of stored data in the relational format.

Types of Data(base) Distribution

Data partitioning truly distributes rows and columns of tables to specific database servers with little or no duplication between servers.

- Vertical partitioning assigns different columns to different servers.
- Horizontal partitioning assigns different rows to different servers.

Data replication duplicates some or all tables on more than one database server.

- Propagates updates on one database server to any other database server where the data is duplicated.

Data Partitioning versus Data Replication

Logical Data Store	Physical Data Stores Using Partitioning	Physical Data Stores Using Replication
1 CUSTOMERS	1P.# Oracle 7: REGION 1 CUSTOMERS 1P.# Oracle 7: REGION 2 CUSTOMERS	Not applicable. Branch offices do not need access to data about customers outside of their own sales region.
2 PRODUCTS	Not applicable. All branch offices need access to data for all products, regardless of sales region.	2M Oracle 8i: PRODUCTS (Master) 2R Oracle 8i: PRODUCTS (Replicated Copy)

为什么要分布数据？ (P356)

1. 有些数据实例仅仅在局部有意义
2. 性能经常可以通过划分数据子集到多个地点而得到提高
3. 有些数据被本地化以确定数据的管理关系

Why distribute data store?

- Some data instances are of local interest only.
- Performance can often be improved by subsetting data to multiple locations.
- Some data needs to be localized to assign custodianship (管理关系） of that data .

Interface Architectures – Inputs, Outputs, & Middleware(自学)

- Batch inputs and outputs
- Online inputs and outputs
- Remote batch
- Keyless data entry (and automatic identification)
- Pen input
- Electronic messaging and work group technology
- Electronic Data Interchange (EDI)
- Imaging and document interchange
- Middleware

Middleware

Middleware – utility software that enables communication between different processors in a system.

- It may be built into the respective operating systems or added through purchased middleware products.
- Presentation middleware： browser and GUI, HTTP
- Application middleware：
- Database middleware： ODBC

13.5 Drawing Physical DFDs for Network Architecture

- Develop a physical data flow diagram (DFD) for the network architecture.
  - Each process symbol represents a server or class of clients.
- For each processor, develop a physical DFD to show the event processes (from Chapter 9) that are assigned to that processor.
- All but simple processes should be factored into (分解)design units and modeled as a more detailed physical DFDs.

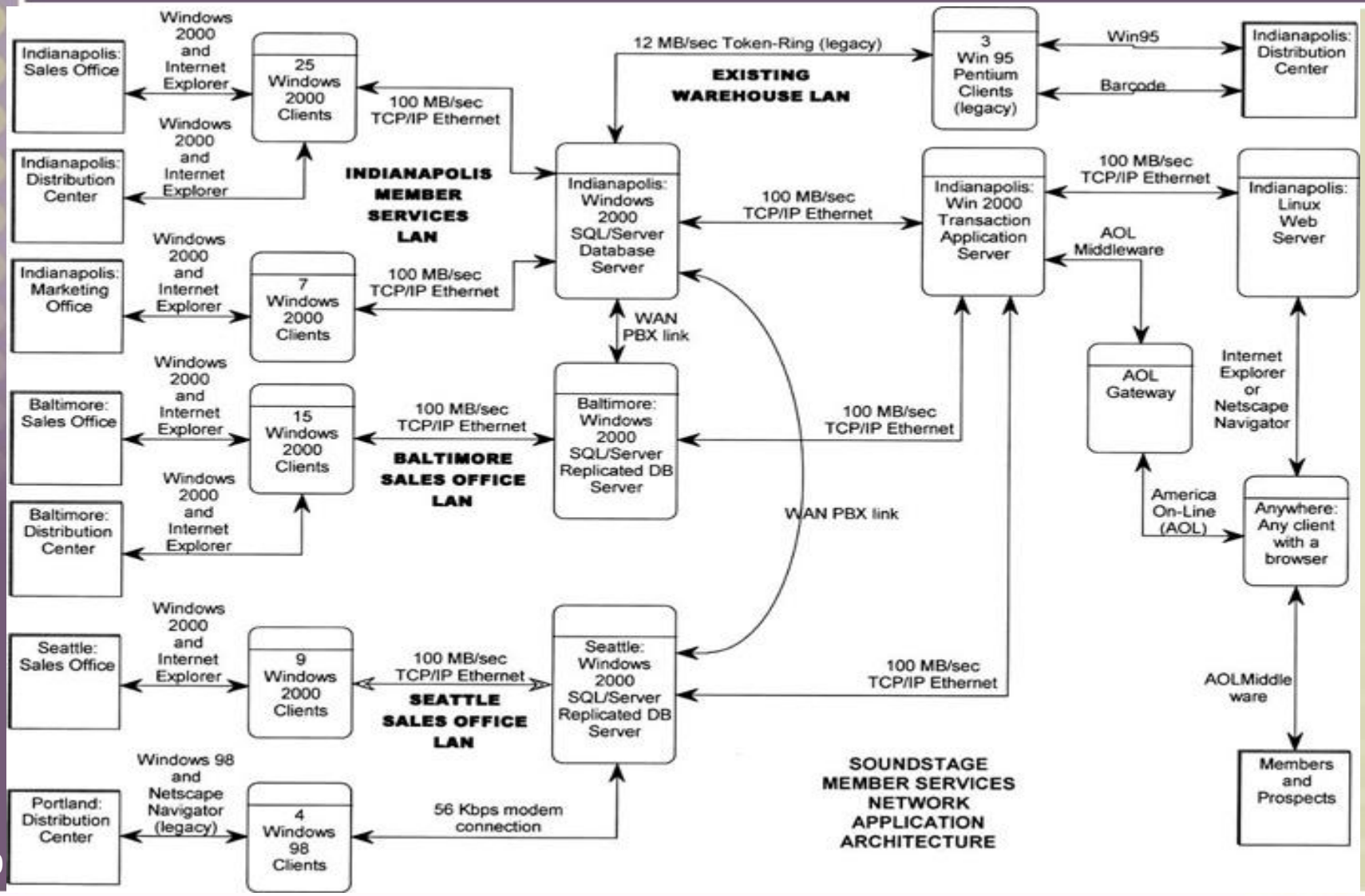
The Network Architecture DFD

Network architecture – a physical DFD that allocates processors (clients and servers) and devices (machines and robots) to a network and establishes:

- the connectivity between clients and servers
- where users will interface with the processors

- Server and its geographic location
- Client and its geographic location
- Processor description
- Protocol

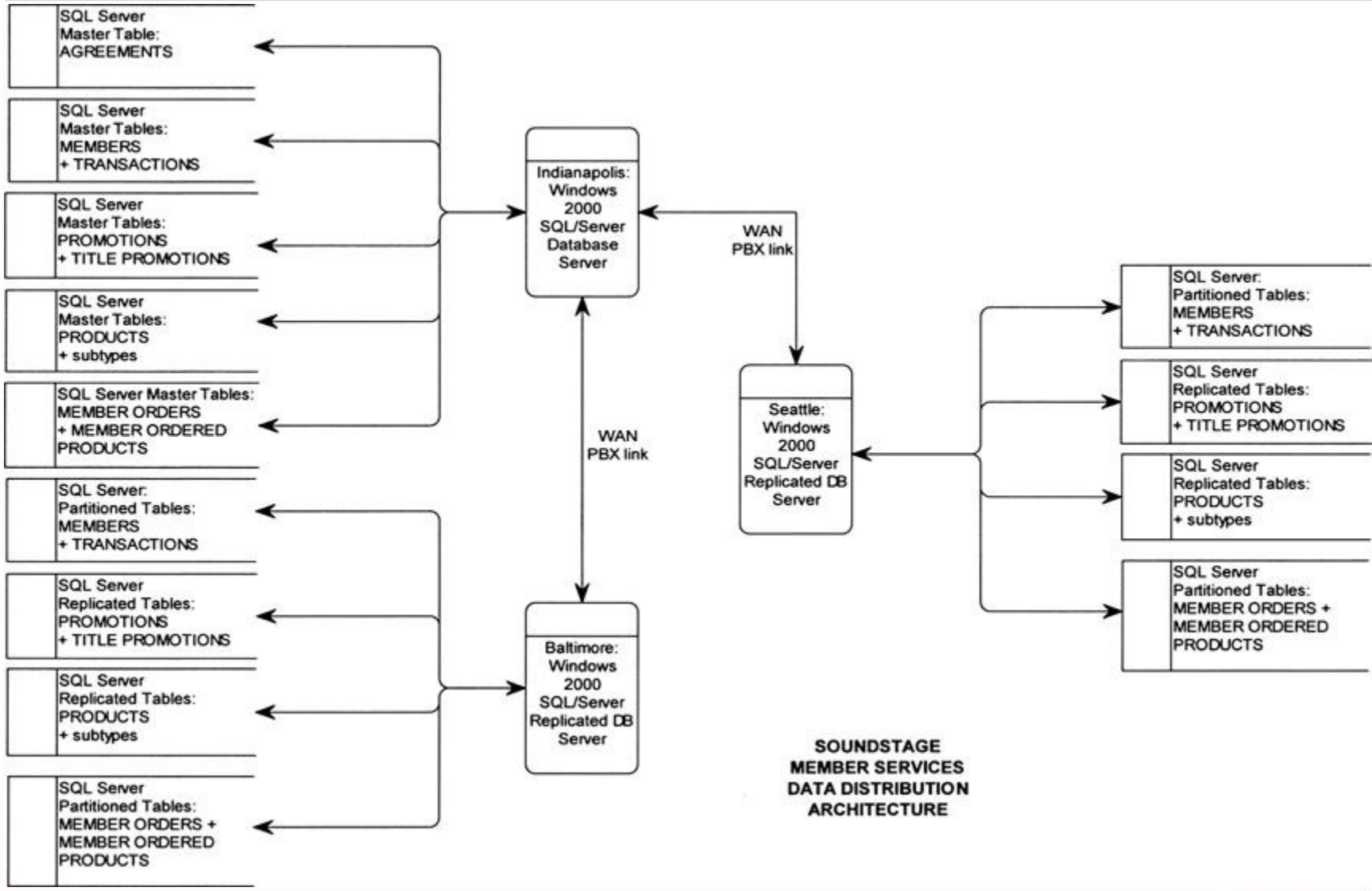
Network Architecture DFD （P355）



Data Distribution Options

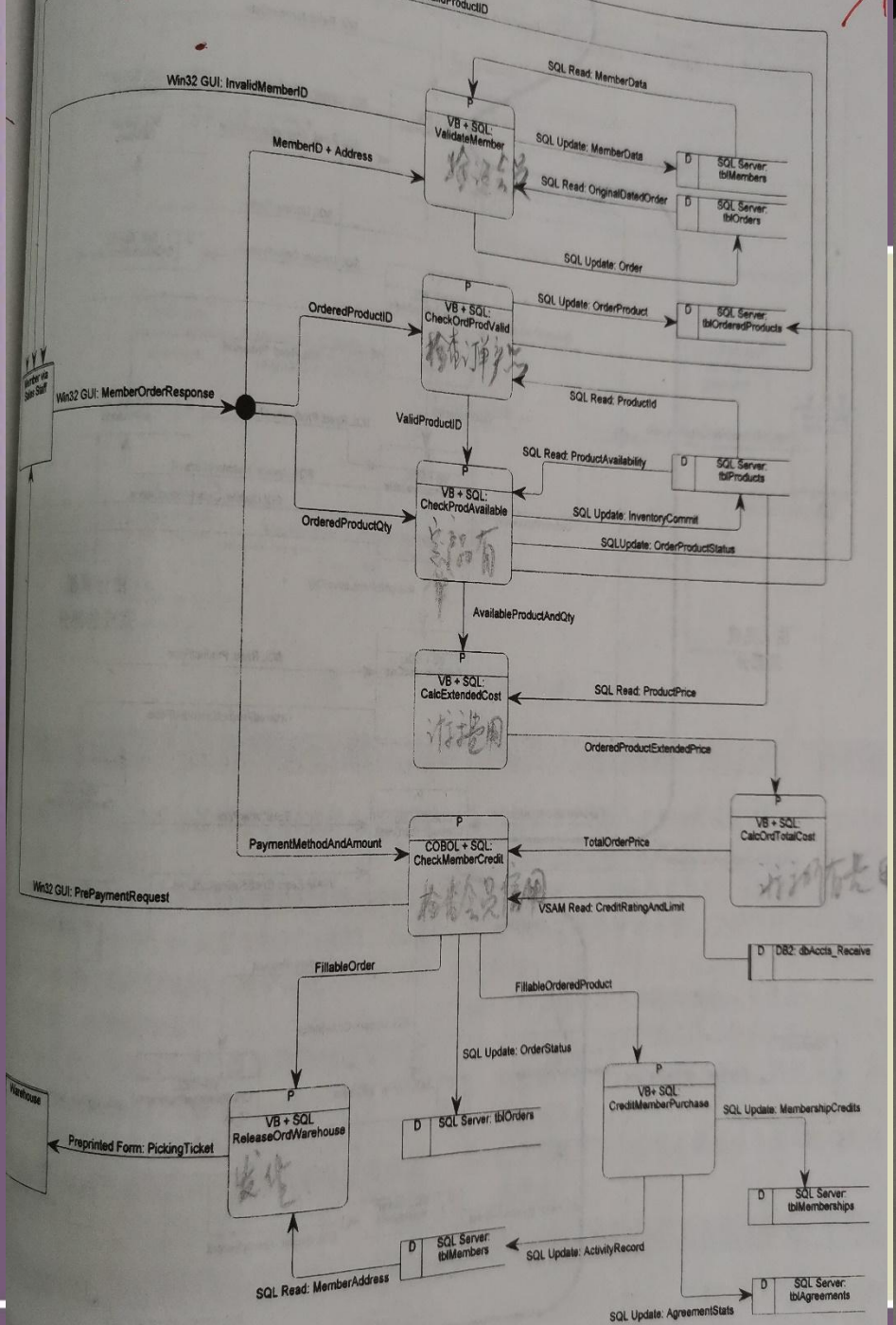
- Store all data on a single server.
- Store specific tables on different servers.
- Store subsets of specific tables on different servers.
- Replicate (duplicate) specific tables or subsets on different servers.

Data Distribution and Technology Assignments DFD (P356)



Process Distribution and Technology Assignments

- For two-tiered client/server systems, all logical even diagrams are assigned to the client.
- For three-tiered client/server and network computing systems, must closely examine each event's primitive (detailed) DFD.
  - Determine which primitive processes should be assigned to the client and which should be assigned to an application server.
  - Generally data capture and editing are assigned to servers
  - If different aspects of a single DFD are partitioned to different clients and servers, draw separate physical DFD for each.



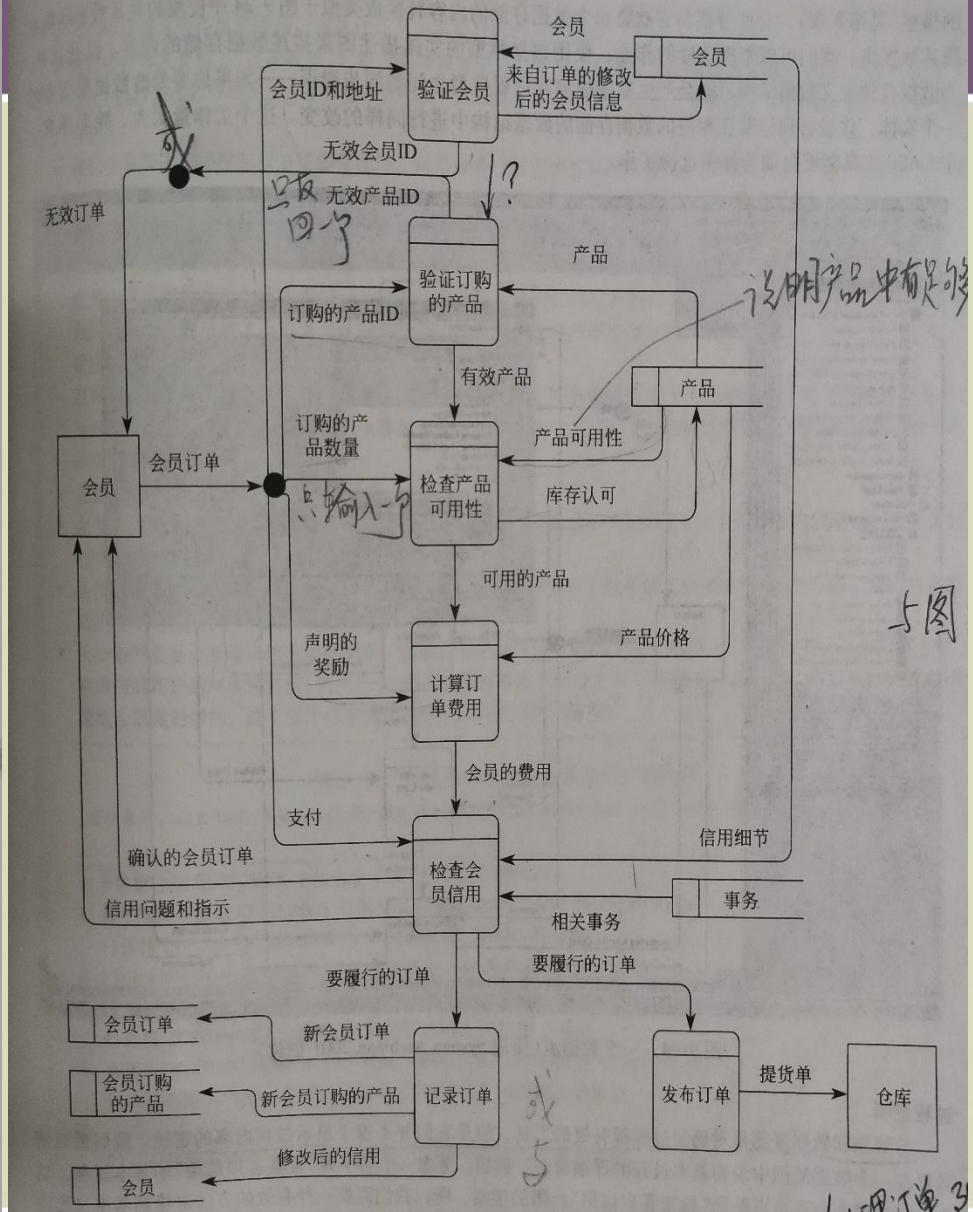
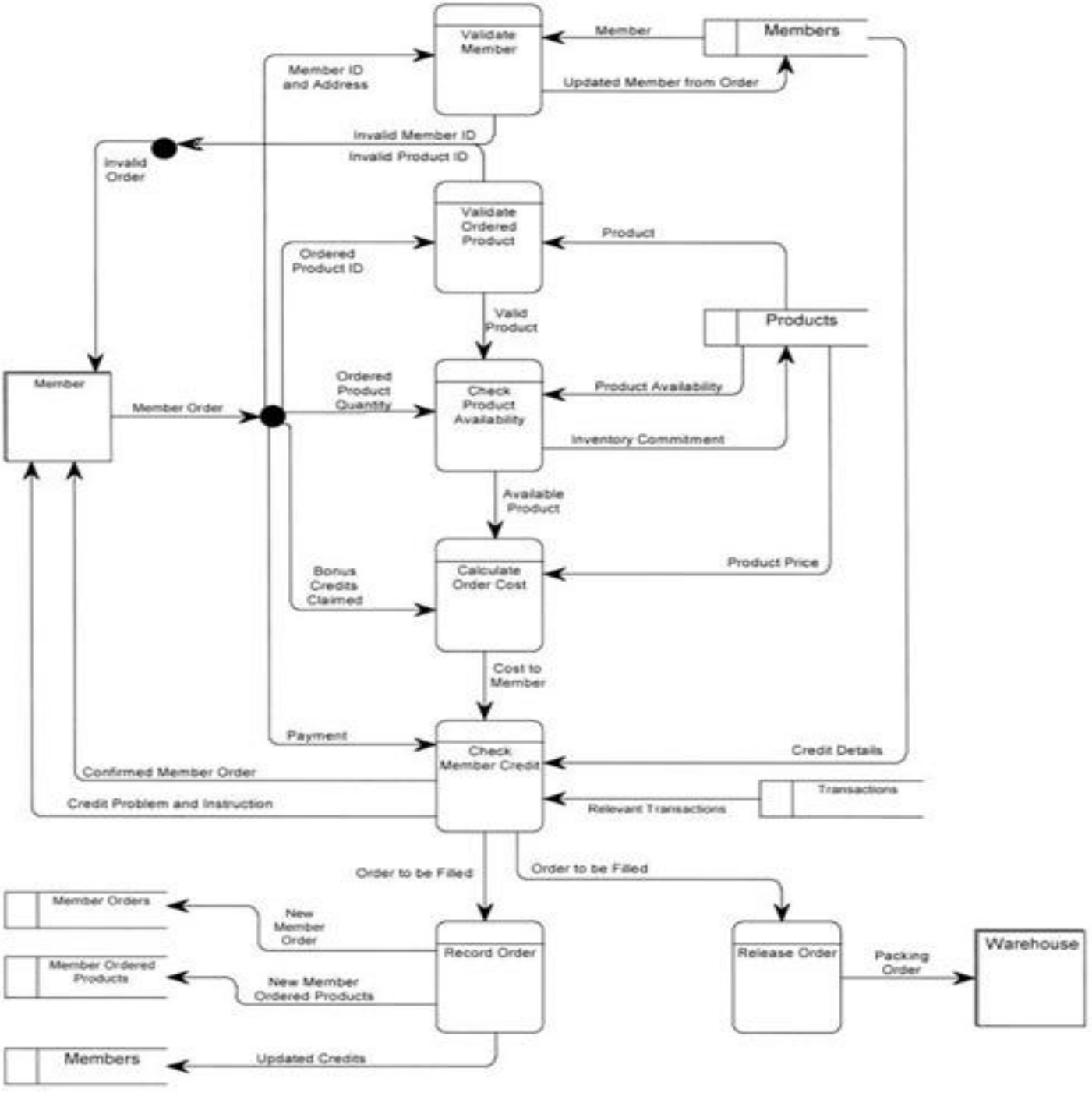
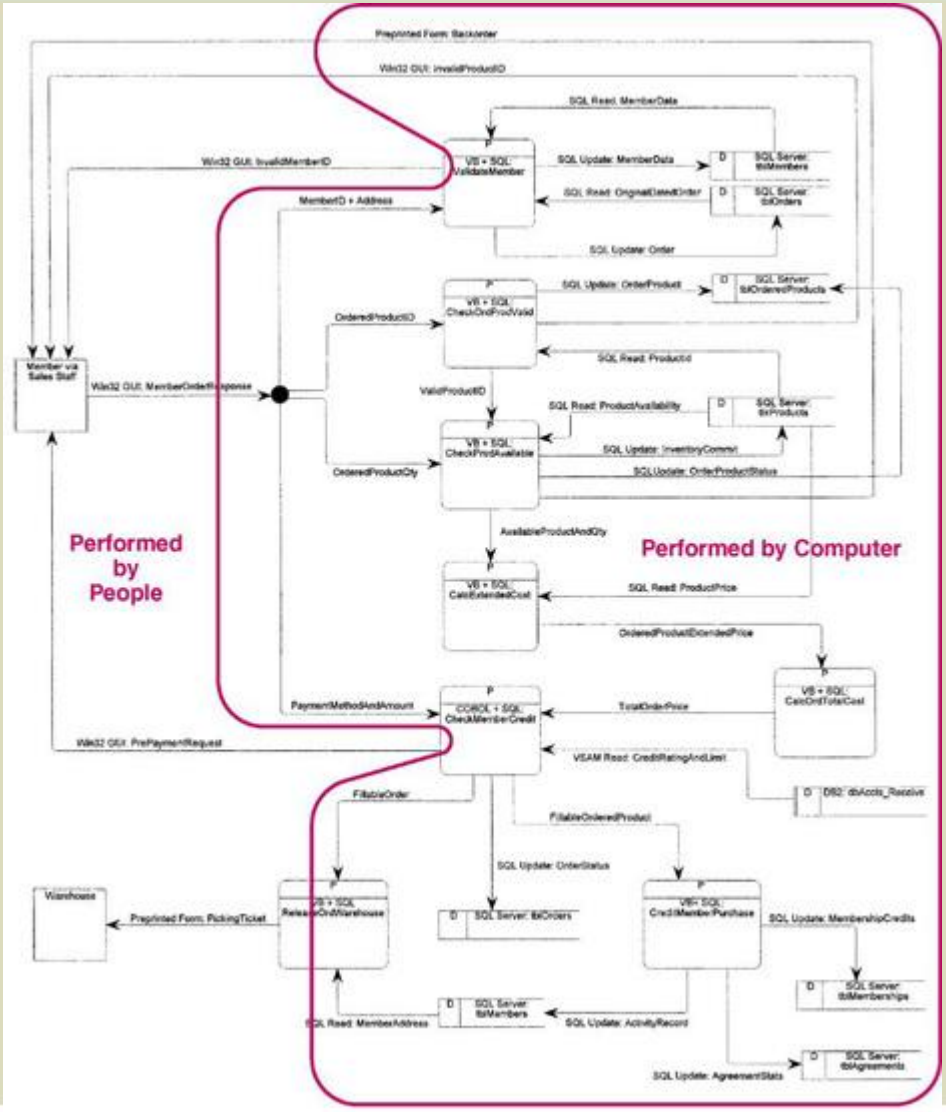


图 9-23 一个基本图(使用 System Architect 2001 创建)

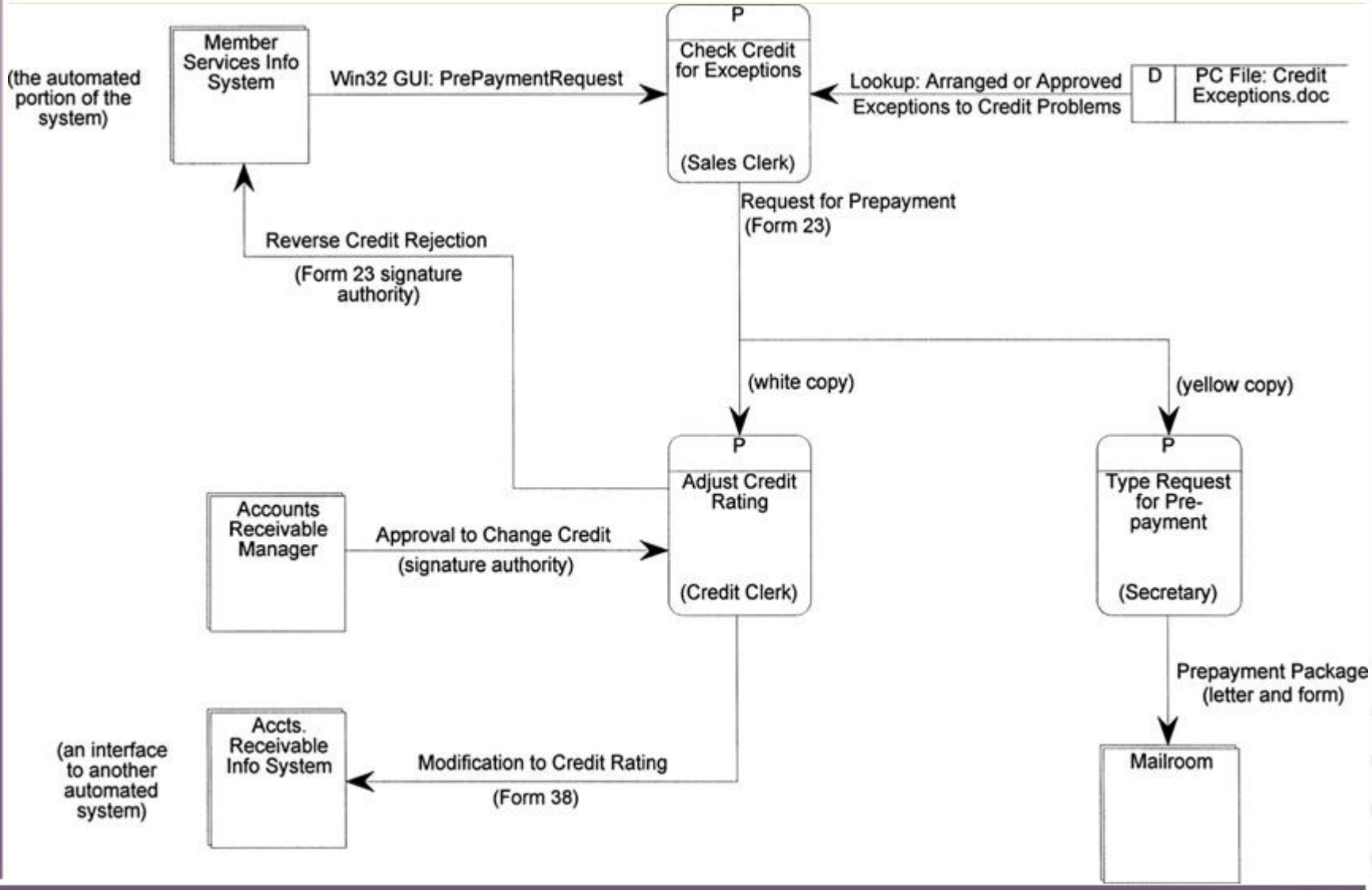

Validate Member
Member
Members
Member ID and Address
Updated Member from Order
Invalid Member ID
Invalid Product ID
Invalid Order
Validate Ordered Product
Product
Ordered Product ID
Valid Product
Products
Member
Member Order
Ordered Product Quantity
Check Product Availability
Product Availability
Inventory Commitment
Available Product
Calculate Order Cost
Product Price
Bonus Credits Claimed
Cost to Member
Payment
Check Member Credit
Credit Details
Confirmed Member Order
Credit Problem and Instruction
Transactions
Relevant Transactions
Order to be Filled
Order to be Filled
Member Orders
New Member Order
Record Order
Release Order
Packing Order
Warehouse
Member Ordered Products
New Member Ordered Products
Members
Updated Credits

The Person/Machine Boundary

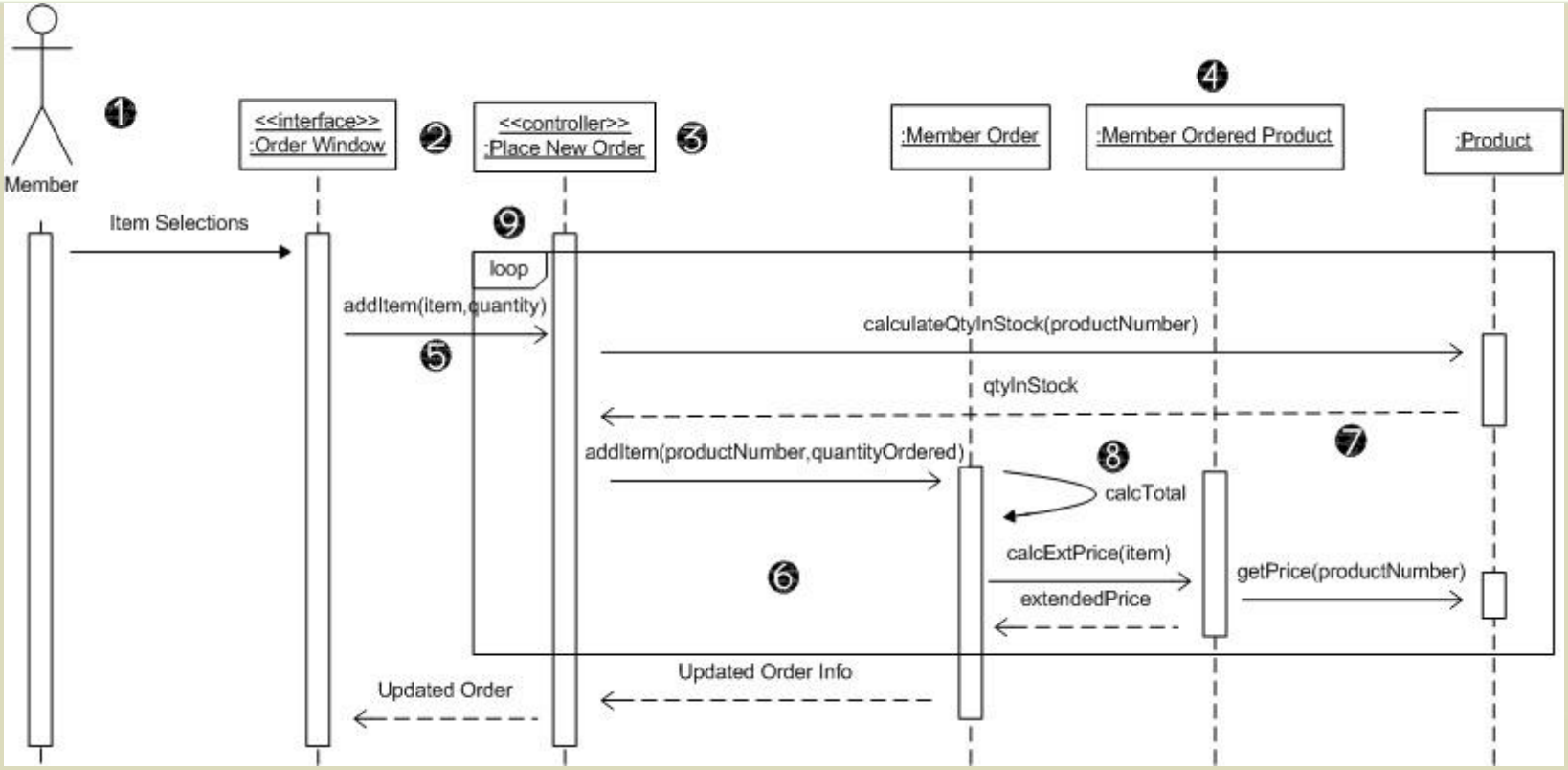


See Figure 13-30 in text for a more readable version

A Manual Design Unit



Sequence Diagram(P463)



1. Actor
2. Interface class
3. Controller class
4. Entity classes
5. Messages
6. Activation bars
7. Return messages
8. Self-call
9. Frame

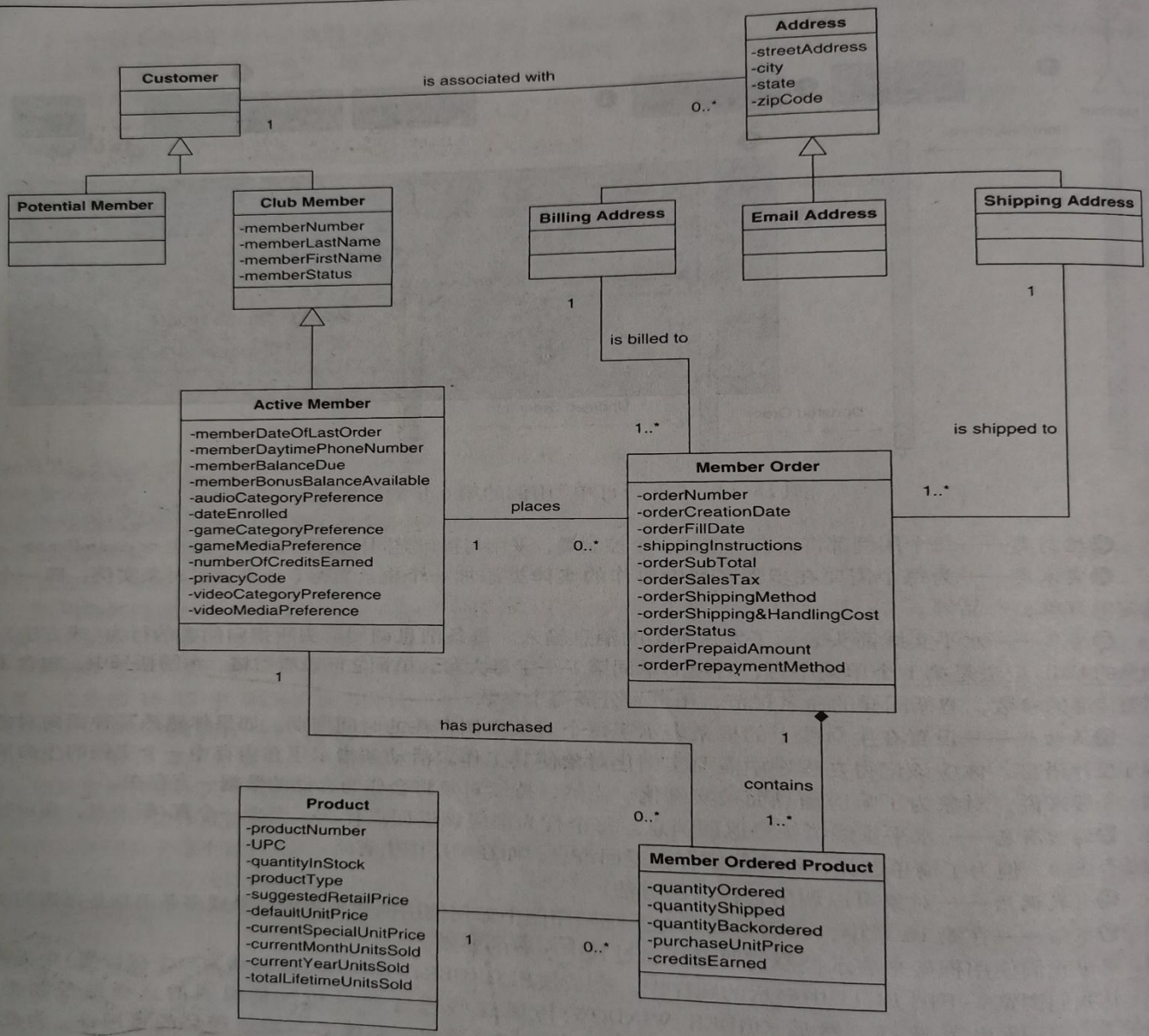
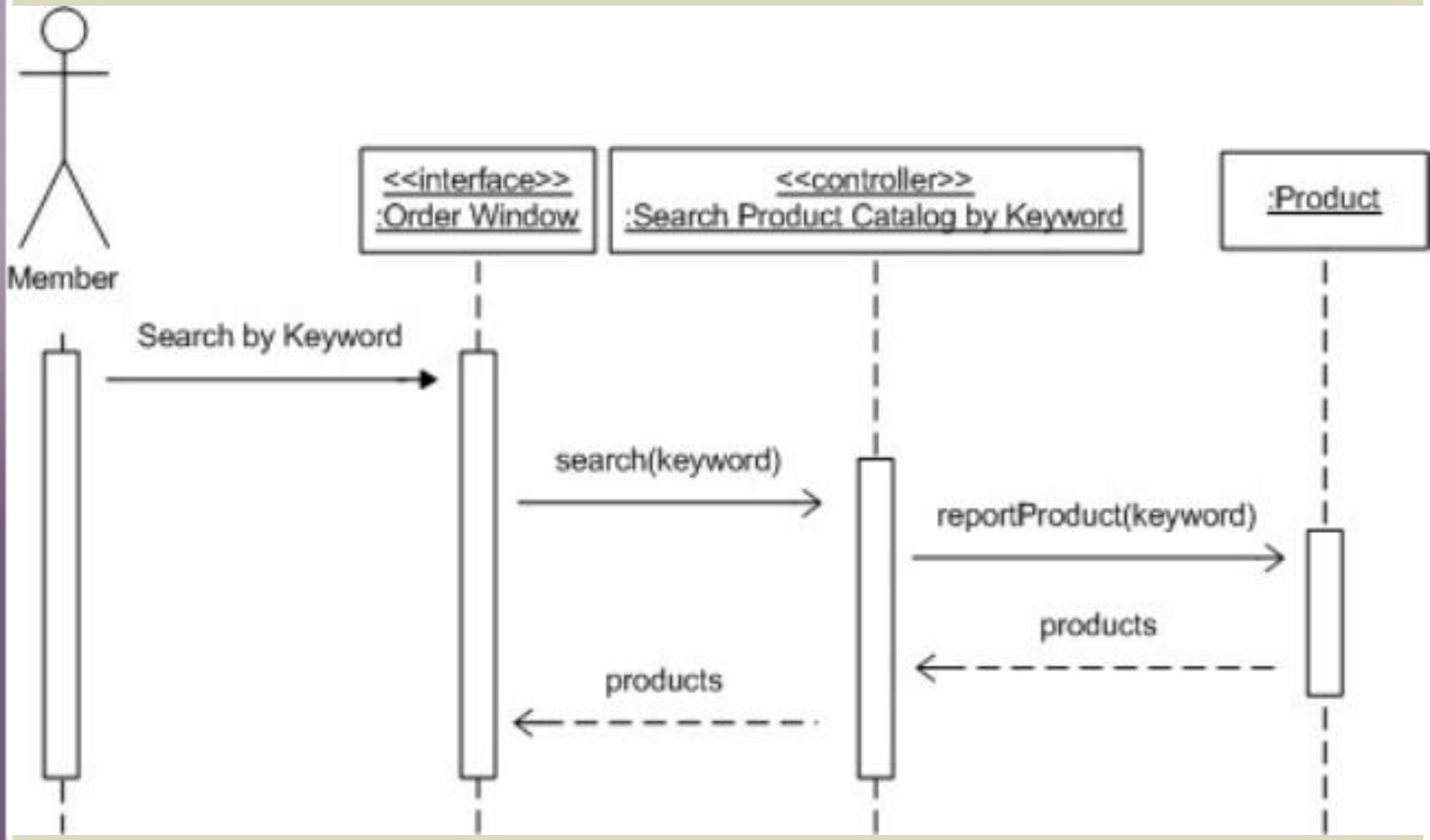


图 18-12 “下新订单”用例的部分类图

calcExtPrice，计算需要 PRODUCT 的

Another Sequence Diagram



Guidelines for Constructing Sequence Diagrams

- Identify the scope of the sequence diagram, whether entire use-case scenario or one step.
- Draw actor and interface class if scope includes that.
- List use-case steps down the left-hand side.
- Draw boxes for controller class and each entity class that must collaborate in the sequence (based on attributes or behaviors previously assigned).
- Add persistence and system classes if scope includes that.
- Draw messages and point each to class that will fulfill the responsibility.
- Add activation bars to indicate object instance lifetimes.
- Add return messages as needed for clarity.
- Add frames for loops, optional steps, alternate steps, etc.

Partial Design Class Diagram