软件体系结构原理、方法与实践

新的开始，新的起点，让我们一起为梦想而努力。

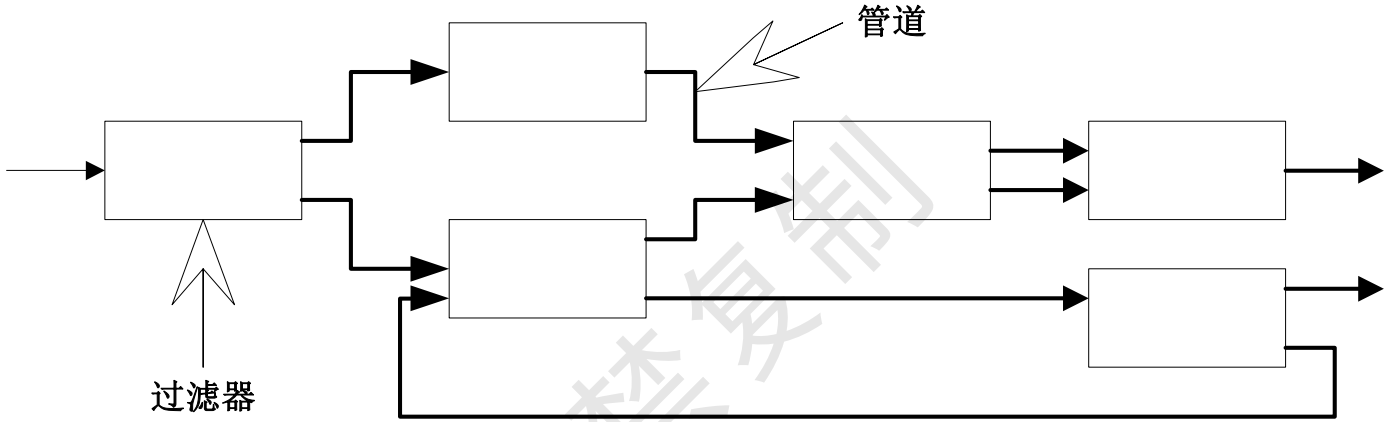
软件体系结构风格的定义

- 软件体系结构风格是描述某一特定领域中系统组织方式的惯用模式。
- 体系结构风格定义了一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。
- 体系结构风格反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

经典的软件体系结构风格

- 数据流风格：批处理序列；管道/过滤器。
- 调用/返回风格：主程序/子程序；面向对象风格；层次结构。
- 独立构件风格：进程通讯；事件系统。
- 虚拟机风格：解释器；基于规则的系统。
- 仓库风格：数据库系统；超文本系统；黑板系统。

经典软件体系结构风格 – 管道与过滤器 – 概述*



每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。

这里的构件被称为过滤器，这种风格的连接件就象是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。

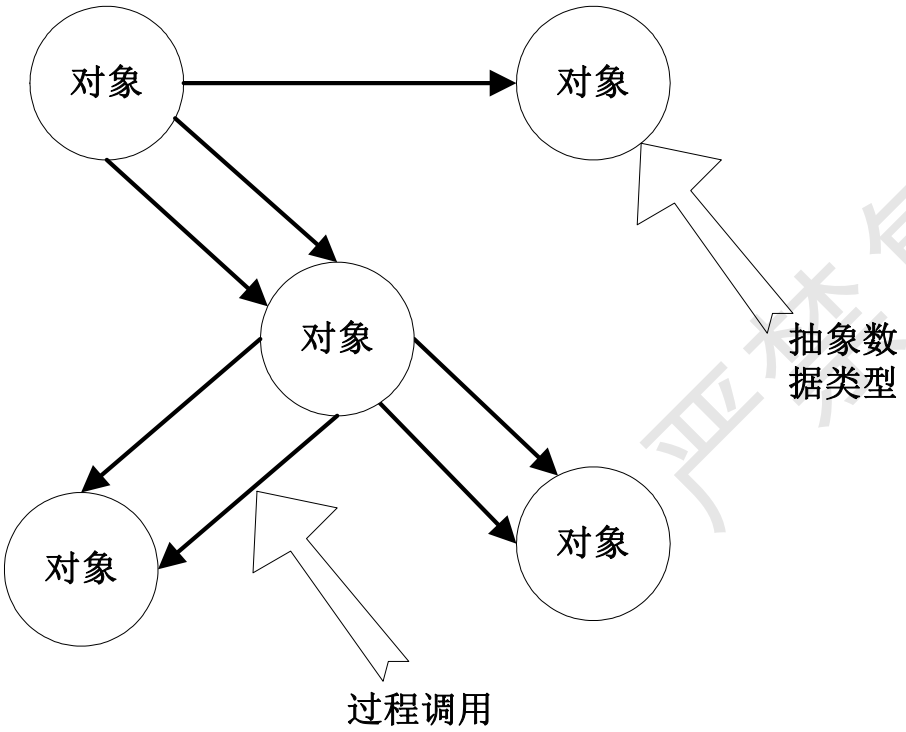
经典软件体系结构风格 – 管道与过滤器 – 优点

- 使得软构件具有良好的隐蔽性和高内聚、低耦合的特点；
- 允许设计者将整个系统的输入/输出行为看成是多个过滤器的行为的简单合成；
- 支持软件重用。只要提供适合在两个过滤器之间传送的数据，任何两个过滤器都可被连接起来；
- 系统维护和增强系统性能简单。新的过滤器可以添加到现有系统中来；旧的可以被改进的过滤器替换掉；
- 允许对一些如吞吐量、死锁等属性的分析；
- 支持并行执行。每个过滤器是作为一个单独的任务完成，因此可与其它任务并行执行。

经典软件体系结构风格 – 管道与过滤器 – 缺点

- 通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但它们是独立的，所以设计者必须将每个过滤器看成一个完整的从输入到输出的转换；
- 不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为严重；
- 因为在数据传输上没有通用的标准，每个过滤器都增加了解析和合成数据的工作，这样就导致了系统性能下降，并增加了编写过滤器的复杂性。

经典软件体系结构风格 – 数据抽象和面向对象组织 – 概述



这种风格建立在数据抽象和面向对象的基础上，数据的表示方法和它们的相应操作封装在一个抽象数据类型或对象中。这种风格的**构件是对象**，或者说是抽象数据类型的实例。对象是一种被称作管理者的构件，因为它负责保持资源的完整性。对象是通过函数和过程的调用来交互的。

封装 继承 多态

新的开始，新的起点，让我们一起为梦想而努力。

经典软件体系结构风格 – 数据抽象和面向对象组织 – 优点

- 因为对象对其它对象隐藏它的表示，所以可以改变一个对象的表示，而不影响其它的对象；
- 设计者可将一些数据存取操作的问题分解成一些交互的代理程序的集合。

经典软件体系结构风格 – 数据抽象和面向对象组织 – 缺点

- 为了使一个对象和另一个对象通过过程调用等进行交互，必须知道对象的标识。只要一个对象的标识改变了，就必须修改所有其他明确调用它的对象[**对对象的依赖性**]。
- 必须修改所有显式调用它的其它对象，并消除由此带来的一些副作用。例如，如果A使用了对象B，C也使用了对象B，那么，C对B的使用所造成的对A的影响可能是料想不到的。

经典软件体系结构风格 – 基于事件的隐式调用 – 概述

- 构件不直接调用一个过程，而是**触发或广播一个或多个事件**。系统中的其它构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。
- 这种风格的构件是一些模块，模块既可以是一些过程，又可以是一些事件的集合。过程可以用通用的方式调用，也可以在系统事件中注册一些过程，当发生这些事件时，过程被调用。
- 这种风格的主要特点是事件的触发者并不知道哪些构件会被这些事件影响。这样不能假定构件的处理顺序，甚至不知道哪些过程会被调用，因此，许多隐式调用的系统也包含显式调用作为构件交互的补充形式。

经典软件体系结构风格 – 基于事件的隐式调用 – 优点

- 为软件重用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中。
- 为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其它构件的接口。

经典软件体系结构风格 – 基于事件的隐式调用 – 缺点

- 构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其它构件是否会响应它。
- 数据交换的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠数据库进行交互。**如果大量的数据与事件绑定进行广播传输，会带来性能问题。**
- 无法保证组件调用的顺序，对数据处理结果的正确性会造成负面的影响

经典软件体系结构风格 – 分层系统 – 概述

- 层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。
- 这种风格支持基于可增加抽象层的设计。允许将一个复杂问题分解成一个增量步骤序列的实现。由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。
- 下层无需知道上层的存在，每一层对其上层隐藏实现细节(虚拟机)

经典软件体系结构风格 – 分层系统 – 优点

- 支持基于抽象程度递增的系统设计，使设计者可以把一个复杂系统按递增的步骤进行分解成简单的层次；
- 支持功能增强，因为每一层至多和相邻的上下层交互，因此功能的改变最多影响相邻的上下层；
- 支持重用。只要提供的服务接口定义不变，同一层的不同实现可以交换使用。这样，就可以定义一组标准的接口，而允许各种不同的实现方法。

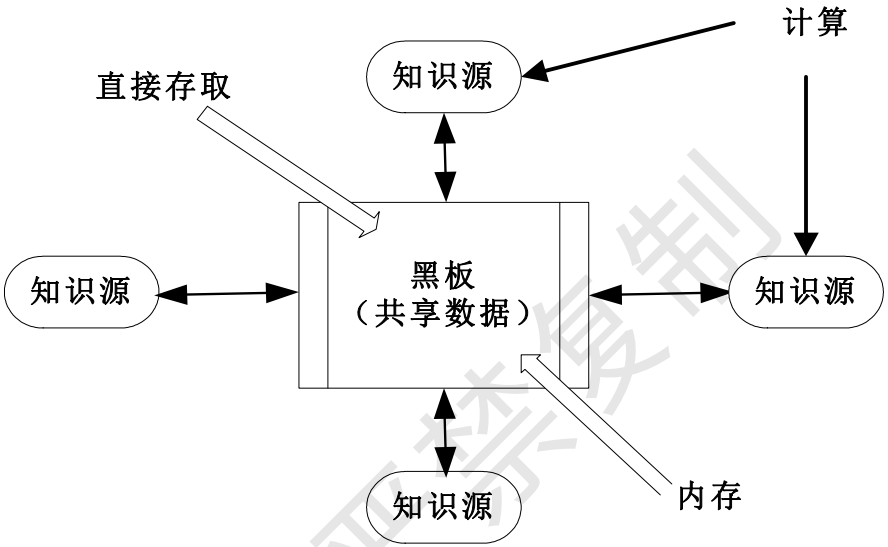
经典软件体系结构风格 – 分层系统 – 缺点

- 并不是每个系统都可以很容易地划分为分层的模式，甚至即使一个系统的逻辑结构是层次化的，出于对系统性能的考虑，系统设计师不得不把一些低级或高级的功能综合起来；
- 很难找到一个合适的、正确的层次抽象方法。

经典软件体系结构风格 – 仓库系统

- 在仓库风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据存贮上执行，仓库与外构件间的相互作用在系统中会有大的变化。
- 控制原则的选取产生两个主要的子类。若输入流中某类时间触发进程执行的选择，则仓库是一传统型数据库；另一方面，若中央数据结构的当前状态触发进程执行的选择，则仓库是一黑板系统。

经典软件体系结构风格 – 仓库系统

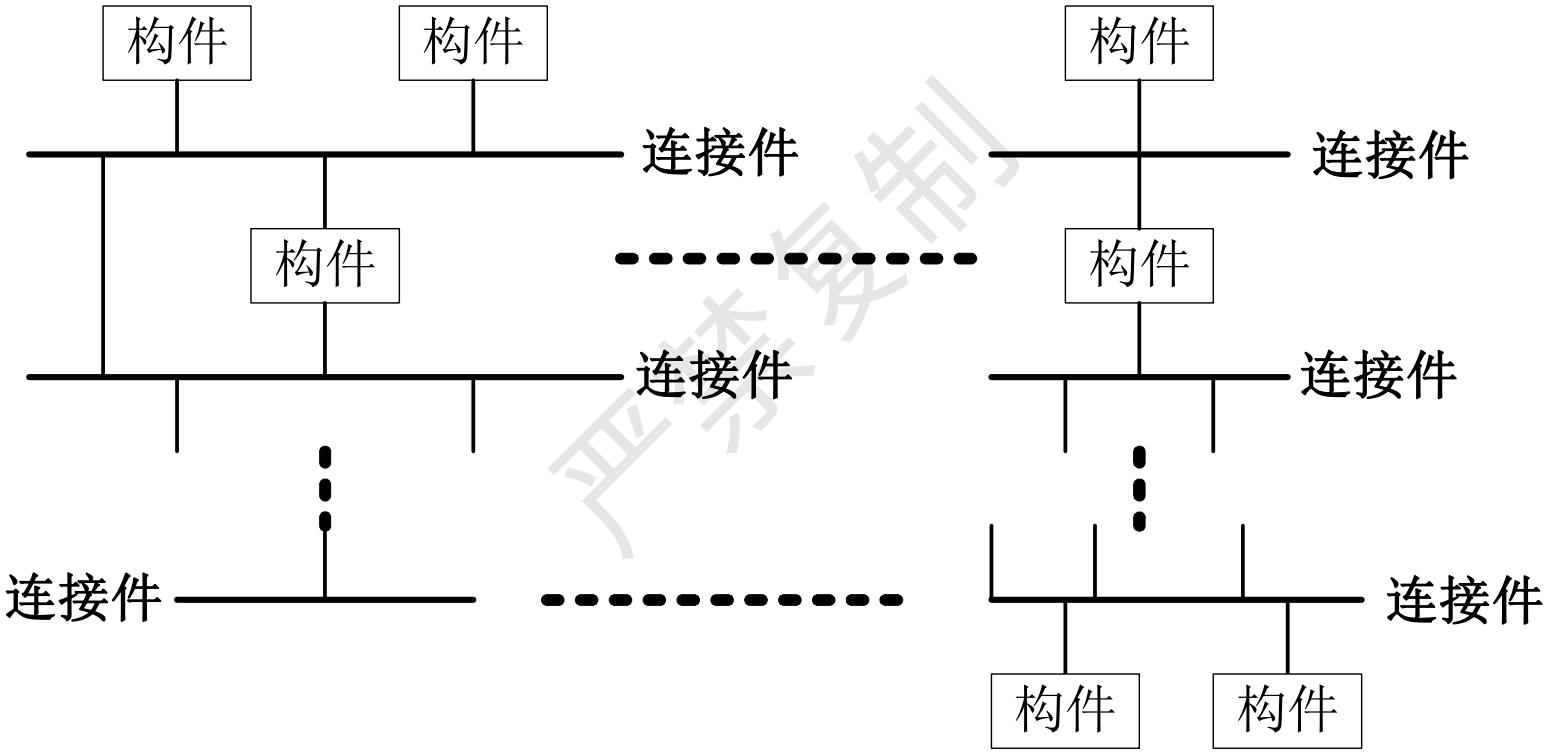


- 知识源：包含独立的、与应用程序相关的知识，知识源之间不进行直接通信，它们之间的交互只通过黑板来完成。
- 黑板数据结构：按照与应用程序相关的层次来组织的解决问题的数据，知识源通过不断的改变黑板数据来解决问题。
- 控制：完全由黑板的状态驱动，黑板状态的改变决定使用的特定知识。

经典软件体系结构风格 – C2风格 – 概述

- 系统中的构件和连接件都有一个顶部和一个底部；
- 构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直接连接是不允许的；
- 一个连接件可以和任意数目的其它构件和连接件连接；
- 当两个连接件进行直接连接时，必须由其中一个的底部到另一个的顶部。

经典软件体系结构风格 – C2风格 – 模型



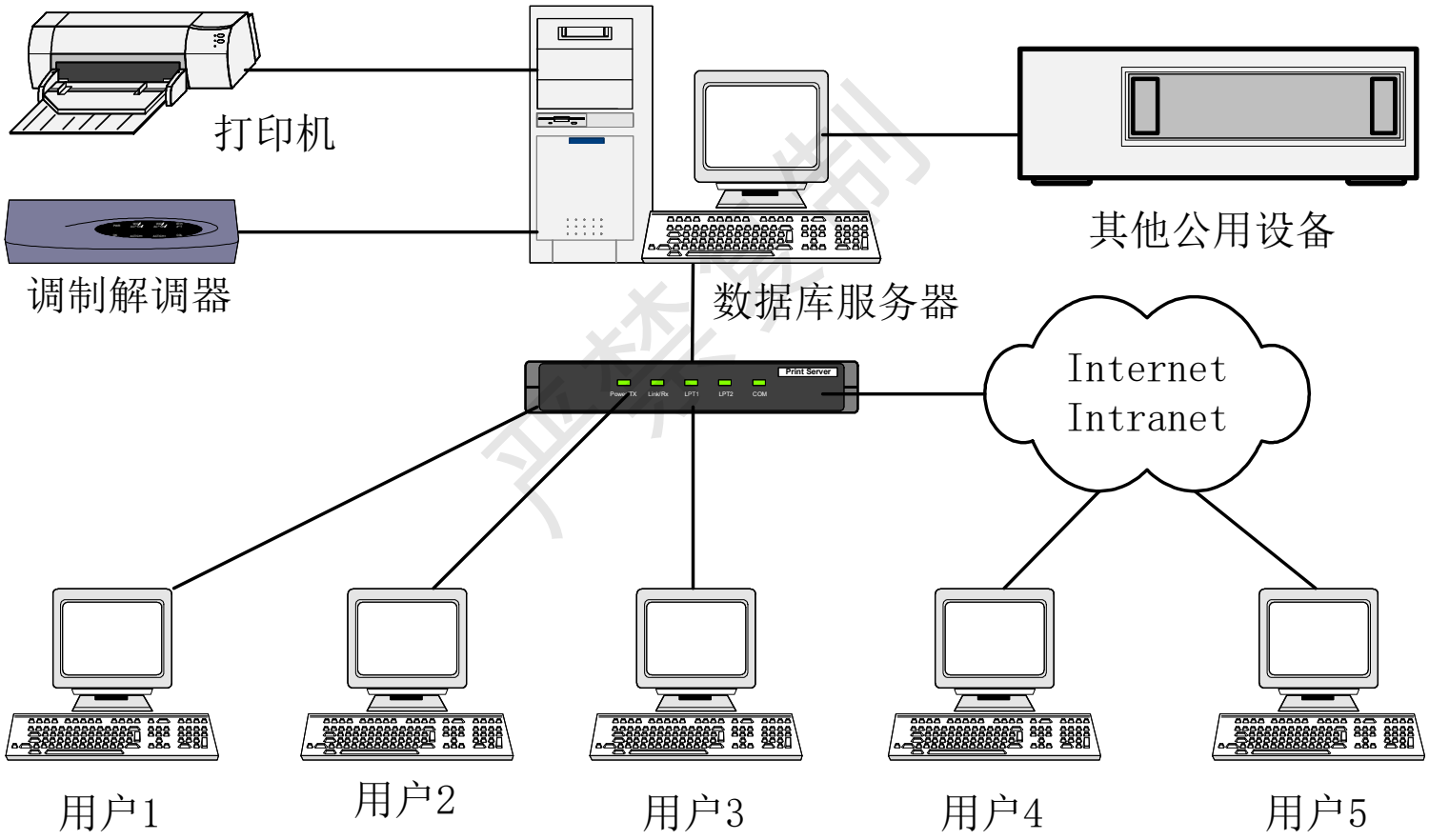
经典软件体系结构风格 – C2风格 – 特点

- 系统中的构件可实现应用需求，并能将任意复杂度的功能封装在一起；
- 所有构件之间的通讯是通过以连接件为中介的异步消息交换机制来实现的；
- 构件相对独立，构件之间依赖性较少。系统中不存在某些构件将在同一地址空间内执行，或某些构件共享特定控制线程之类的相关性假设。

C/S风格 – 概述

- C/S软件体系结构是基于资源不对等，且为实现共享而提出来的，是20世纪90年代成熟起来的技术，C/S体系结构定义了工作站如何与服务器相连，以实现数据和应用分布到多个处理机上。
- C/S体系结构有三个主要组成部分：数据库服务器、客户应用程序和网络。

C/S风格 – 模型


打印机
调制解调器
数据库服务器
其他公用设备
Internet
Intranet
用户1
用户2
用户3
用户4
用户5

C/S风格 – 任务分配

服务器负责任务：

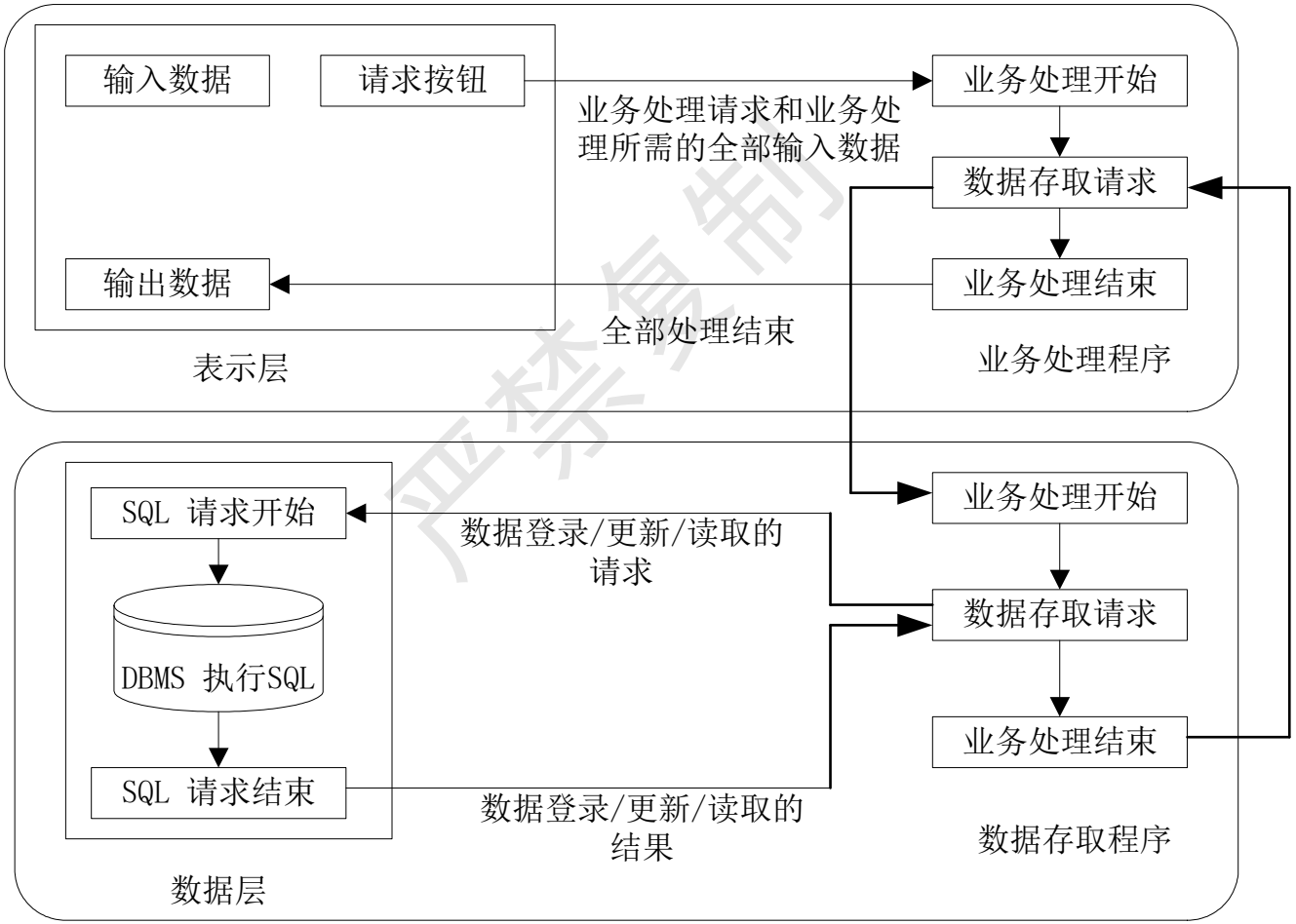
- 数据库安全性的要求；
- 数据库访问并发性的控制；
- 数据库前端的客户应用程序的全局数据完整性规则；
- 数据库的备份与恢复。

C/S风格 – 任务分配

客户应用程序

- 提供用户与数据库交互的界面；
- 向数据库服务器提交用户请求并接收来自数据库服务器的信息；
- 利用客户应用程序对存在于客户端的数据执行应用逻辑要求。

C/S风格 – 处理流程



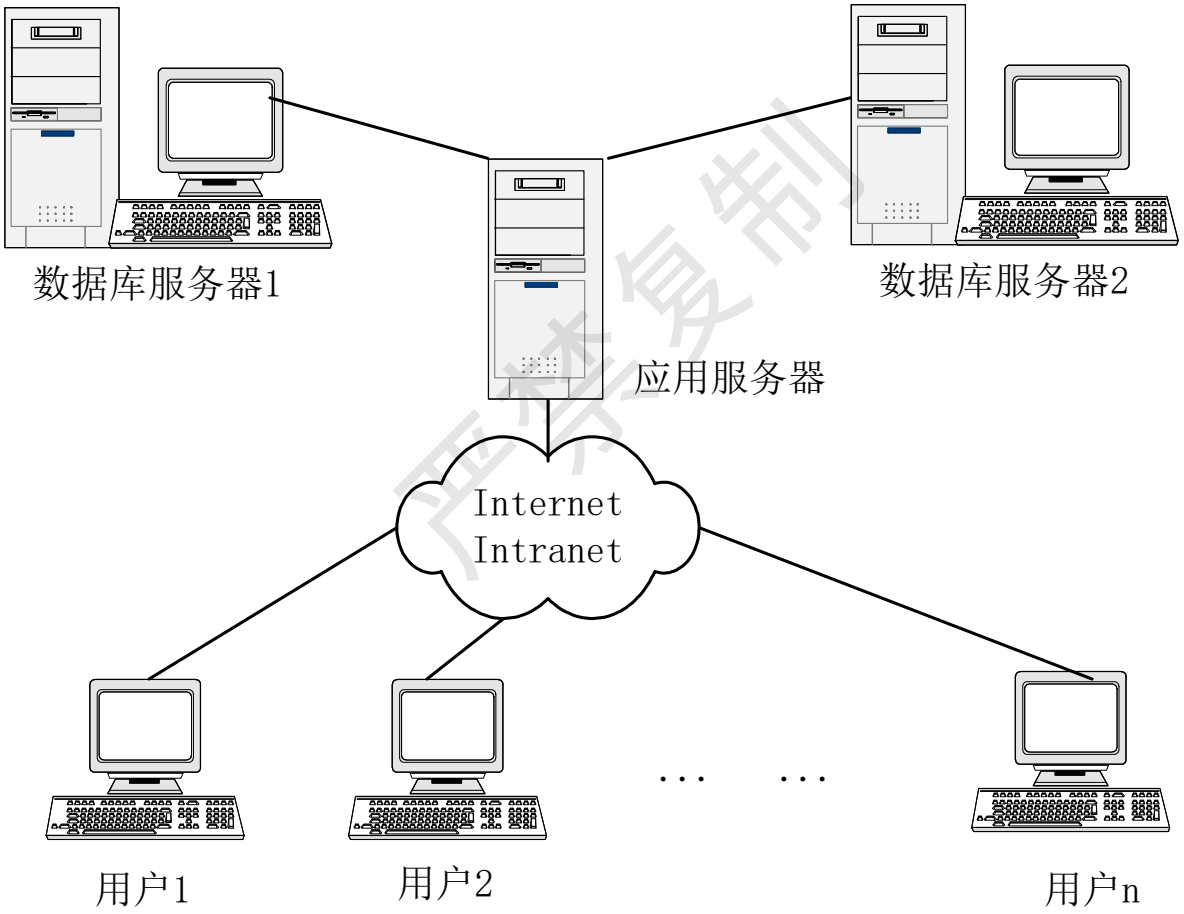
C/S风格 – 优点

- C/S 体系结构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受。
- 系统的客户应用程序和服务器构件分别运行在不同的计算机上，系统中每台服务器都可以适合各构件的要求，这对于硬件和软件的变化显示出极大的适应性和灵活性，而且易于对系统进行扩充和缩小。
- 在C/S体系结构中，系统中的功能构件充分隔离，客户应用程序的开发集中于数据的显示和分析，而数据库服务器的开发则集中于数据的管理，不必在每一个新的应用程序中都要对一个DBMS进行编码。将大的应用处理任务分布到许多通过网络连接的低成本计算机上，以节约大量费用。

C/S风格 – 缺点

- 开发成本较高
- 客户端程序设计复杂
- 信息内容和形式单一
- 用户界面风格不一，使用繁杂，不利于推广使用
- 软件移植困难
- 软件维护和升级困难
- 新技术不能轻易应用

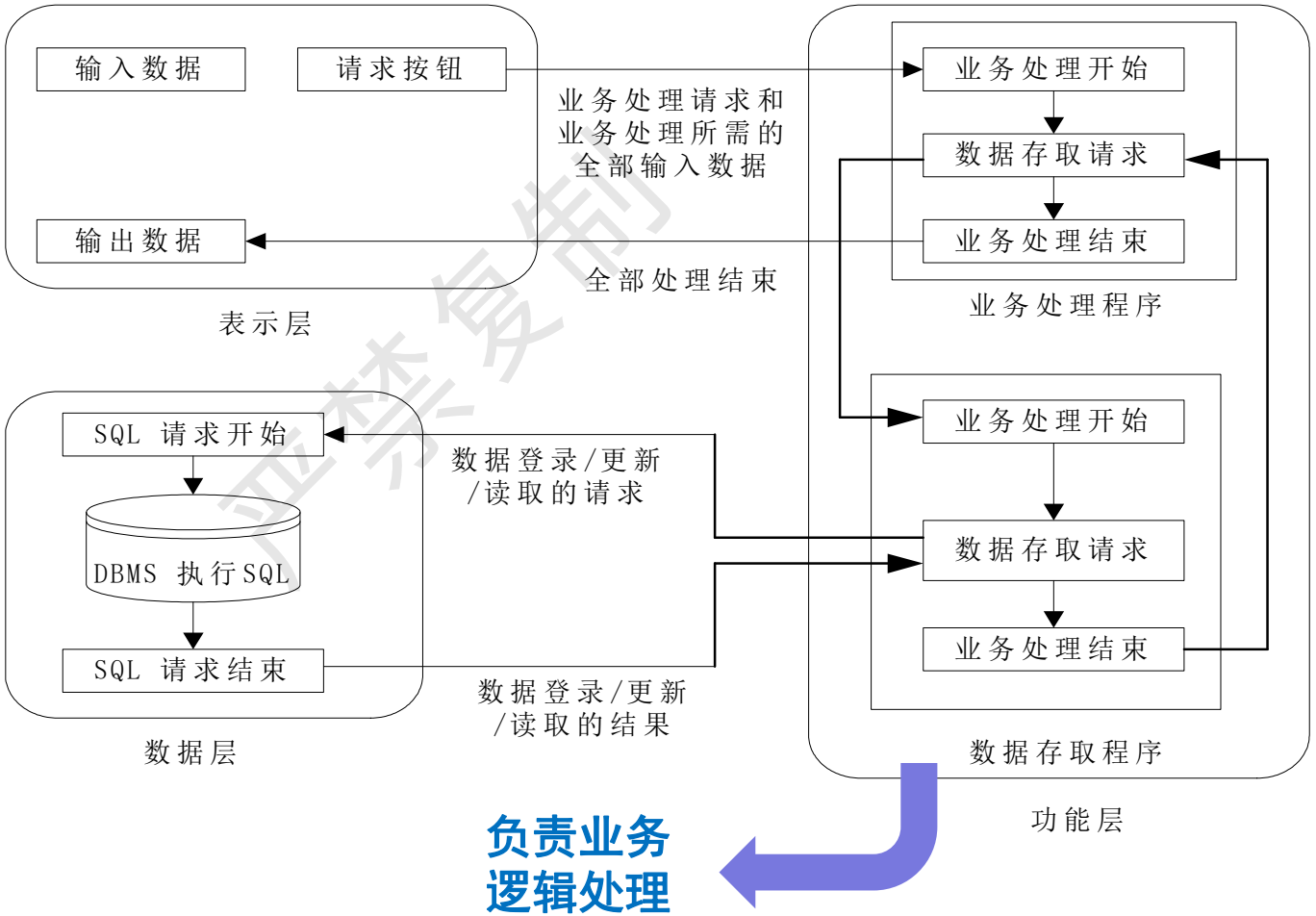
三层C/S风格 – 模型



三层C/S风格 – 处理流程

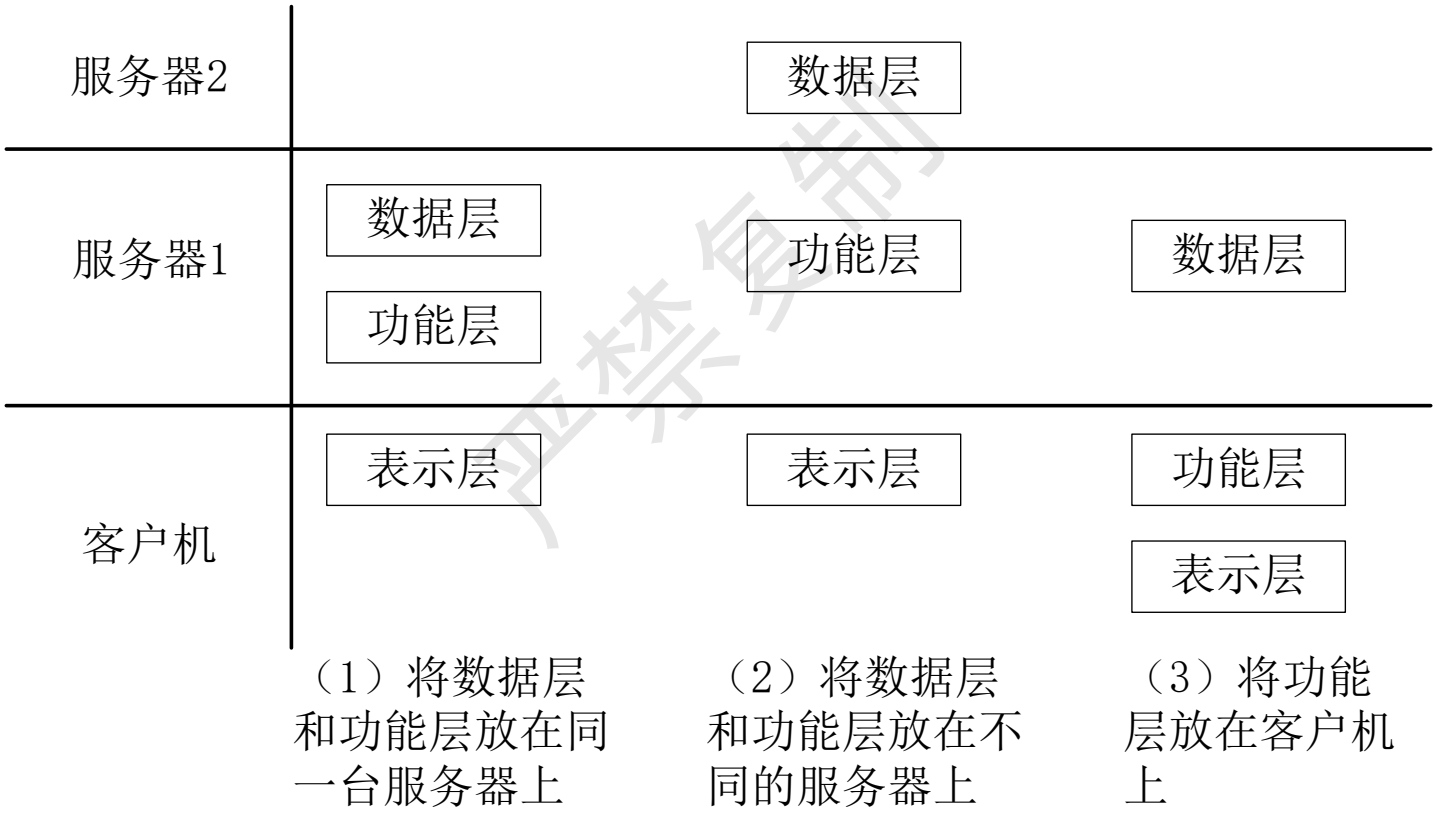
负责用户与应用之间的交互，检查输入，显示输出。

负责对数据的读写



负责业务逻辑处理

三层C/S风格 – 物理结构



三层C/S风格 – 优点

- 允许合理地划分三层结构的功能，使之在逻辑上保持相对独立性，能提高系统和软件的可维护性和可扩展性。
- 允许更灵活有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层；并且这些平台和各个组成部分可以具有良好的可升级性和开放性。
- 应用的各层可以并行开发，可以选择各自最适合的开发语言。
- 利用功能层有效地隔离开表示层与数据层，未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，为严格的安全管理奠定了坚实的基础。

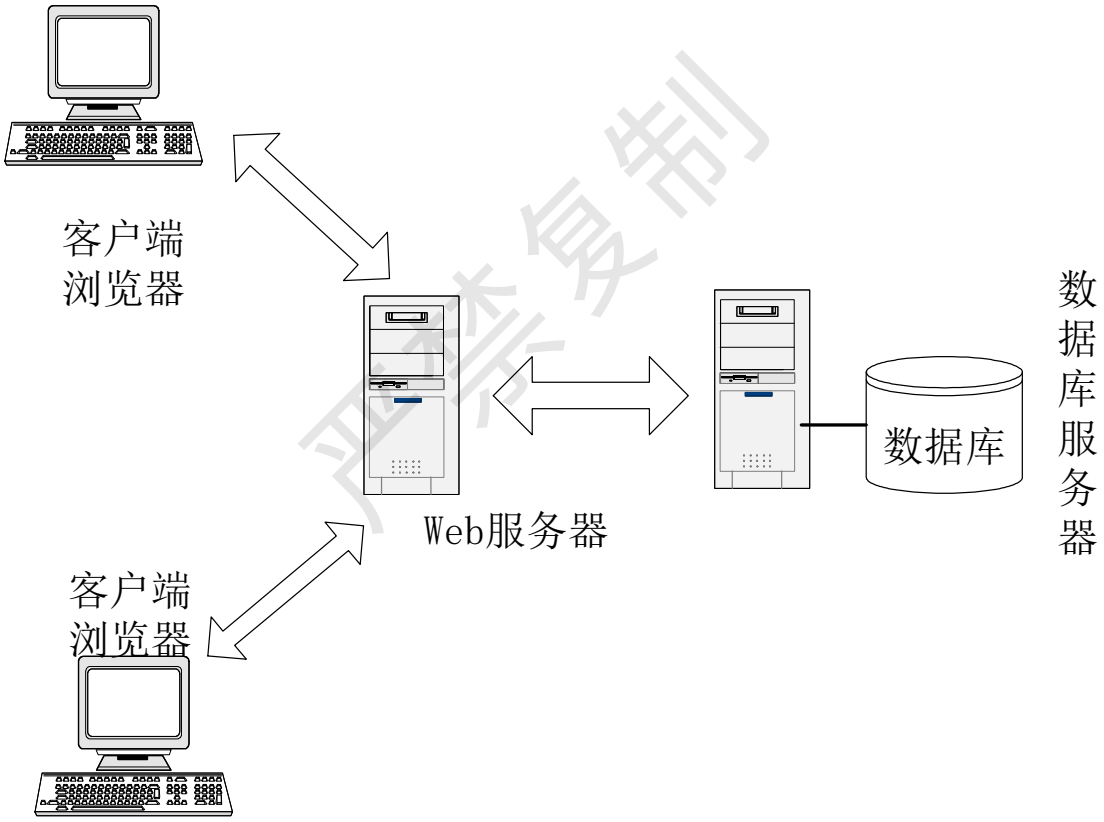
三层C/S风格 – 要注意的问题

- 三层C/S结构各层间的通信效率若不高，即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。
- 设计时必须慎重考虑三层间的通信方法、通信频度及数据量。这和提高各层的独立性一样是三层C/S结构的关键问题。

三层B/S风格 – 概述

- 浏览器/服务器（B/S）风格就是上述三层应用结构的一种实现方式，其具体结构为：浏览器/Web服务器/数据库服务器。
- B/S体系结构主要是利用不断成熟的WWW浏览器技术，结合浏览器的多种脚本语言，用通用浏览器就实现了原来需要复杂的专用软件才能实现的强大功能，并节约了开发成本。从某种程度上来说，B/S结构是一种全新的软件体系结构。

三层B/S风格 – 模型



三层B/S风格 – 优点

- 基于B/S体系结构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。
- B/S体系结构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

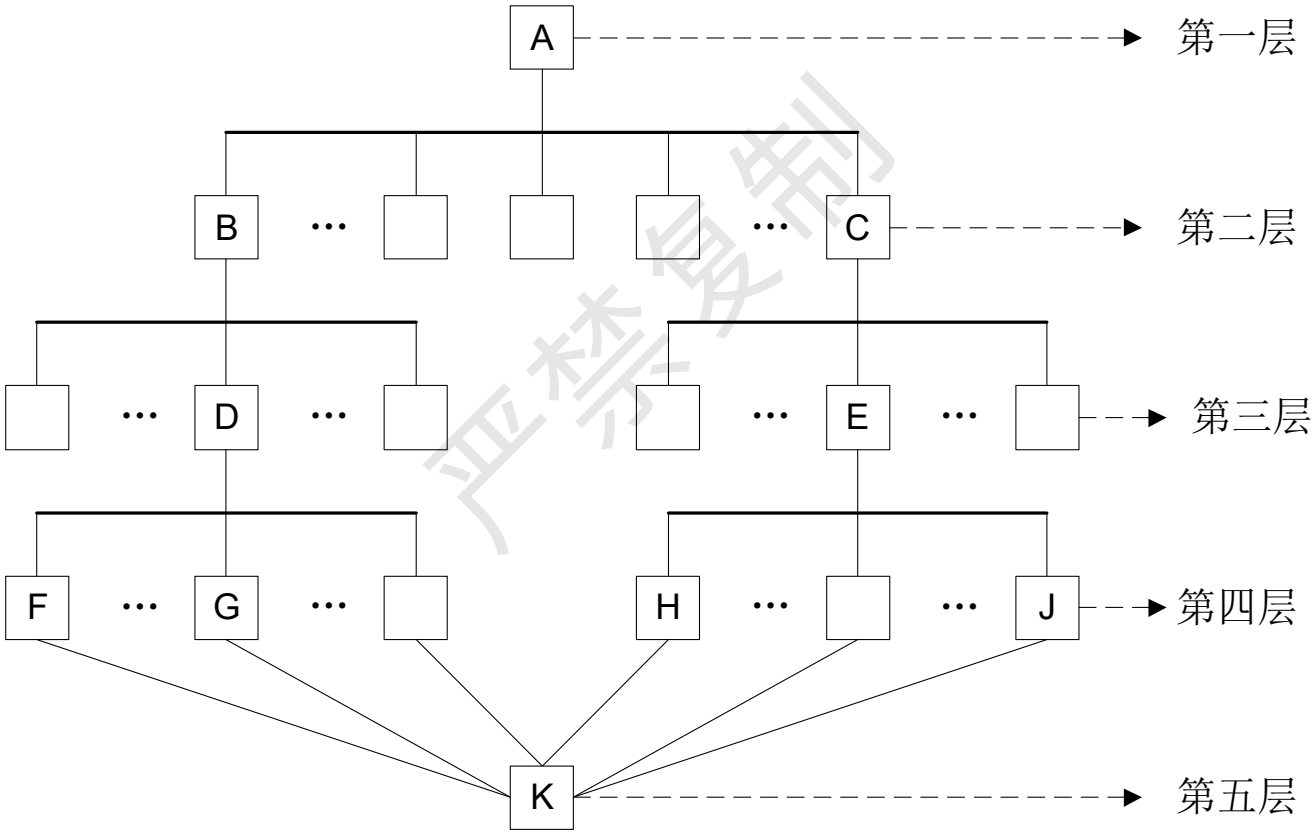
三层B/S风格 – 缺点

- B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。
- B/S体系结构的系统扩展能力差，安全性难以控制。
- 采用B/S体系结构的应用系统，在数据查询等响应速度上，要远远地低于C/S体系结构。
- B/S体系结构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理(OLTP)应用。

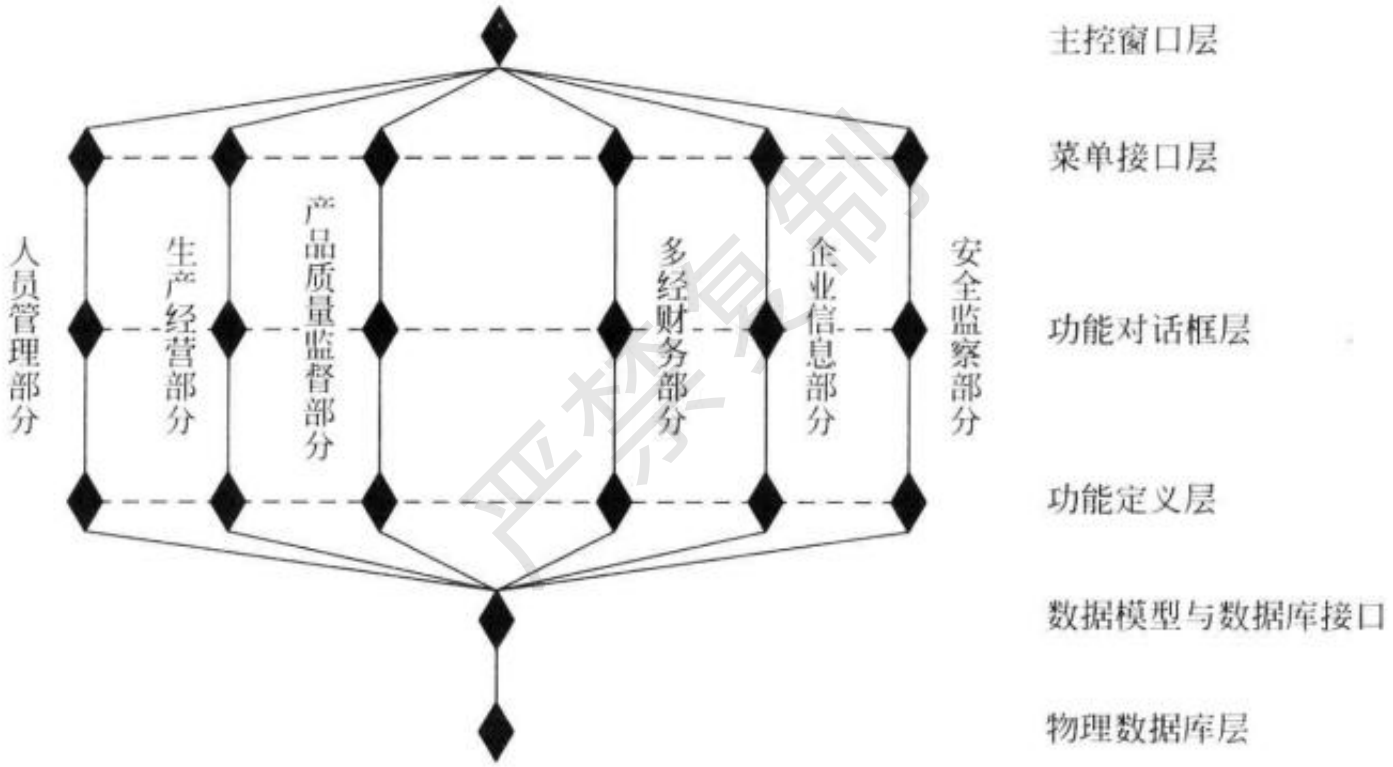
正交软件体系结构 – 概述

- 正交软件体系结构由组织层和线索（子系统）的构件构成。层是由一组具有相同抽象级别的构件构成。线索是子系统的特例，它是由完成不同层次功能的构件组成（通过相互调用来关联），每一条线索完成整个系统中相对独立的一部分功能。每一条线索的实现与其他线索的实现无关或关联很少，在同一层中的构件之间是不存在相互调用的。
- 如果线索（子系统）是相互独立的，即不同线索中的构件之间没有相互调用，那么这个结构就是完全正交的。

正交软件体系结构 – 模型



正交软件体系结构 – 模型实例



正交软件体系结构 – 特征

- 正交软件体系结构由完成不同功能的n（n > 1）个线索（子系统）组成；
- 系统具有m（m > 1）个不同抽象级别的层；
- 线索之间是相互独立的（正交的）；
- 系统有一个公共驱动层（一般为最高层）和公共数据结构（一般为最低层）。

正交软件体系结构 – 优点

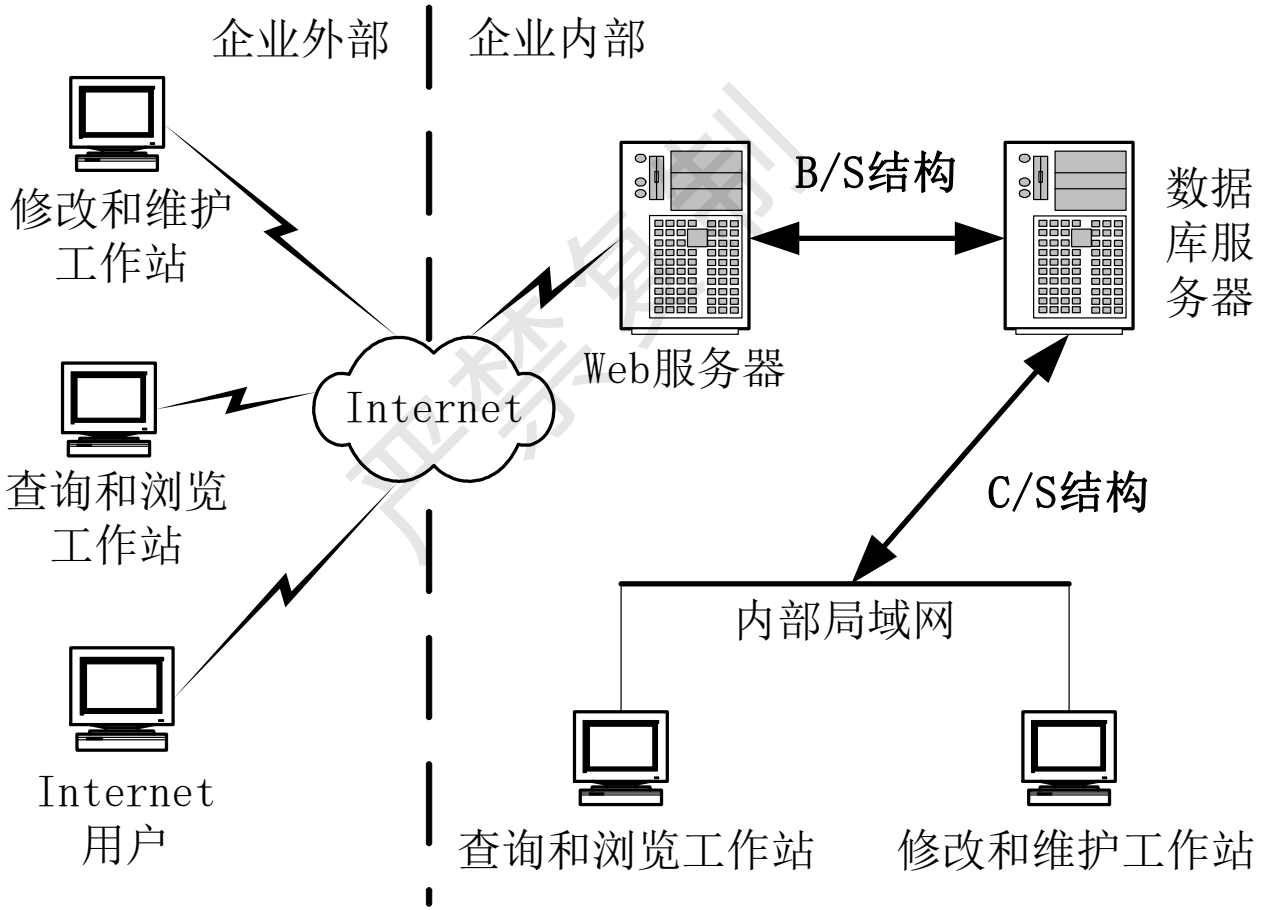
- 结构清晰，易于理解
- 易修改，可维护性强
- 可移植性强，重用粒度大

异构风格 – 概述

随着软件规模的扩大，系统功能也越来越复杂，所有系统不能在单一的标准的体系结构上进行设计:

- 不同的结构有不同的处理能力的强项和弱点，一个系统的体系结构应该根据实际需要进行选择，以解决实际问题。
- 关于软件包、框架、通信以及其他一些体系结构上的问题，目前存在多种标准。即使在某段时间内某一种标准占统治地位，但变动最终是绝对的。
- 实际工作中，我们总会遇到一些遗留下来的代码，它们仍有效用，但是却与新系统有某种程度上的不协调。然而在许多场合，将技术与经济综合进行考虑时，总是决定不再重写它们。
- 即使在某一单位中，规定了共享共同的软件包或相互关系的一些标准，仍会存在解释或表示习惯上的不同。

异构风格 – 内外有别模型

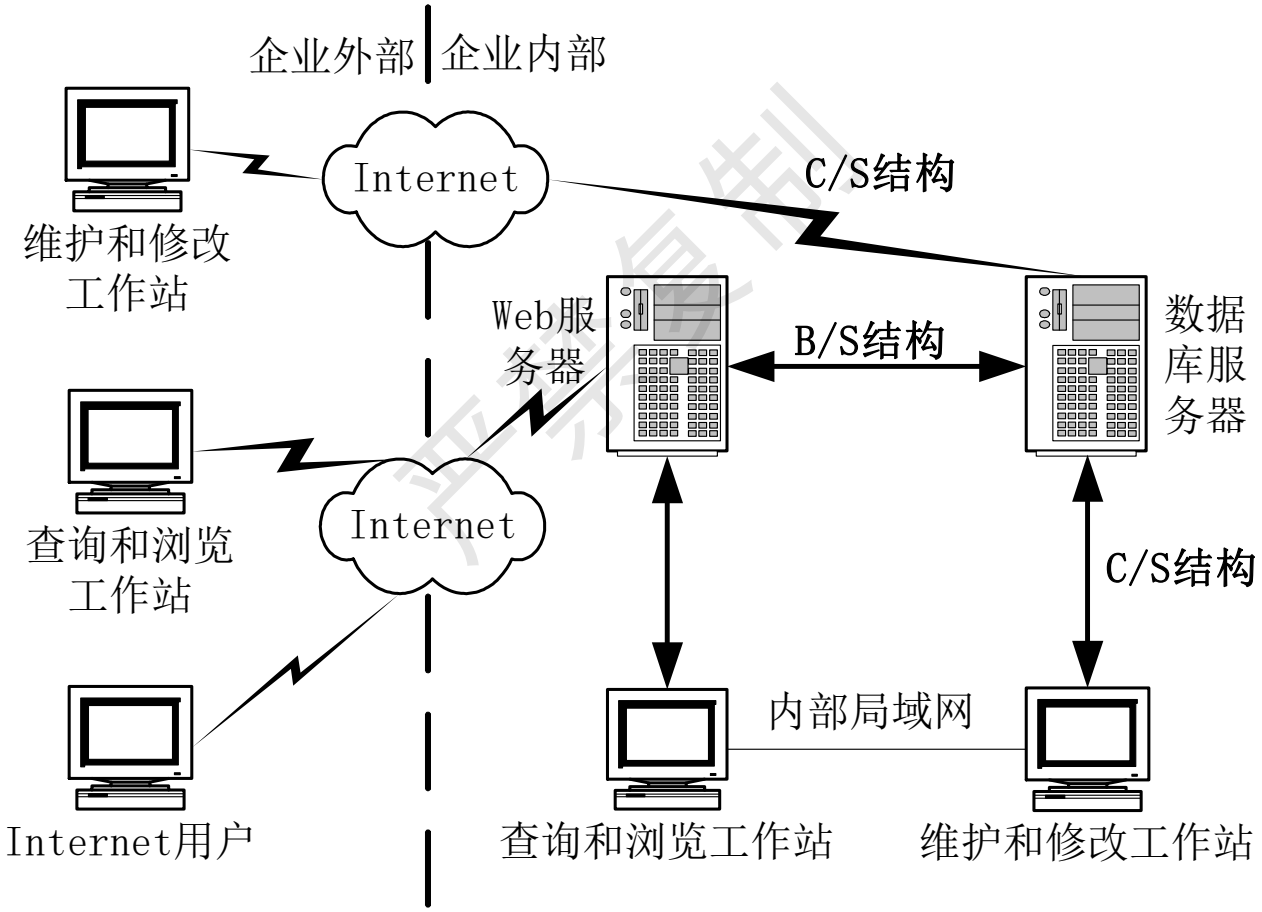

企业外部
企业内部
修改和维护
工作站
查询和浏览
工作站
Internet
用户
Internet
Web服务器
B/S结构
数据
库服
务器
C/S结构
内部局域网
查询和浏览工作站
修改和维护工作站

异构风格 – 内外有别模型

优点:
外部用户不直接访问数据库，服务器，能保证企业数据库的相对安全，企业内部用户的交互性强，数据查询和修改响应速度快。

缺点:
企业外部用户修改和维护数据时，速度较慢，较繁琐，数据的动态交互性不强。

异构风格 – 查改有别模型

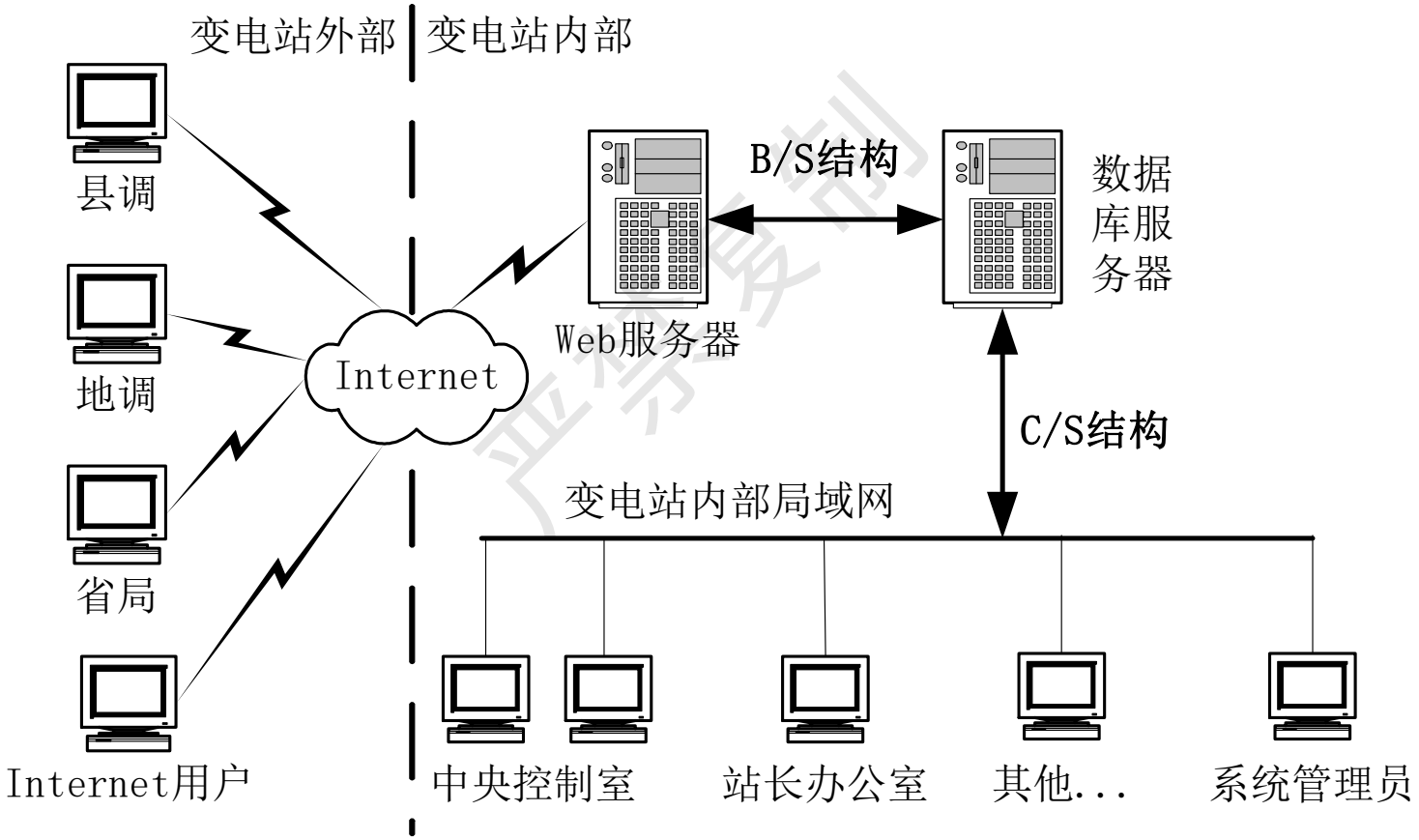

企业外部
企业内部
Internet
C/S结构
维护和修改
工作站
Web服
务器
B/S结构
数据
库服
务器
查询和浏览
工作站
Internet
C/S结构
内部局域网
Internet用户
查询和浏览工作站
维护和修改工作站

异构风格 – 查改有别模型

优点：
企业内部用户的交互性强，数据查询和修改响应速度快，
企业外部用户修改和维护数据效率高。

缺点：
外部用户可直接访问数据库服务器，企业数据容易暴露给外部用户，给数据安全造成了一定的威胁

异构风格 – 实例


变电站外部
变电站内部
县调
地调
省局
Internet用户
Internet
Web服务器
B/S结构
数据库服务器
C/S结构
变电站内部局域网
中央控制室
站长办公室
其他...
系统管理员