4.2 软件体系结构设计

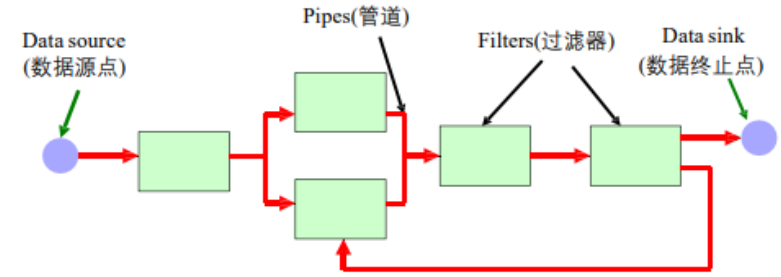
- 软件体系结构要素
  - 软件体系结构概念
    - 软件体系结构（Software Architecture）包括构成系统的设计元素的描述、设计元素之间的交互、设计元素的组合模式以及在这些模式中的约束
    - 软件体系结构=构件+连接件+约束
    - 构件
      - 构件是具有某种功能的可复用的软件结构单元，表示系统中主要的计算元素和数据存储
      - 构件是一个抽象的概念，任何在系统运行中承担一定功能、发挥一定作用的软件体都可看作是构件
      - 特点
        - 可分离：可独立部署执行码文件
        - 可替换
        - 可配置：外界通过规范化的配置机制修改构建配置数据
        - 可复用：可以不经源代码修改，无需重新编译，即可应用于多个软件项目或软件产品
      - 构件组成
        - 接口

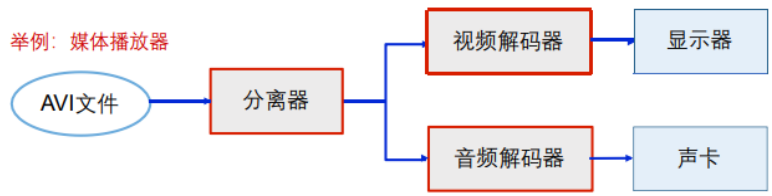
          构件接口是构件间的契约
          一个接口提供一种服务，完成某种逻辑行为
          构件作为一个封装的实体，只能通过其接口(Interface)与外部环境交互，表示了构件和外部环境的交互点，内部具体实现则被隐藏起来(Black-box)；
        - 实现功能

          构件接口服务的实现
          构件核心逻辑实现
          构件内部所实现的功能以方法、操作(functions、behaviors)的形式体现出来，并通过接口向外发布，进而产生与其它构件之间的关联。
    - 连接
      - 构件间建立和维护行为关联与信息传递的途径
      - 机制

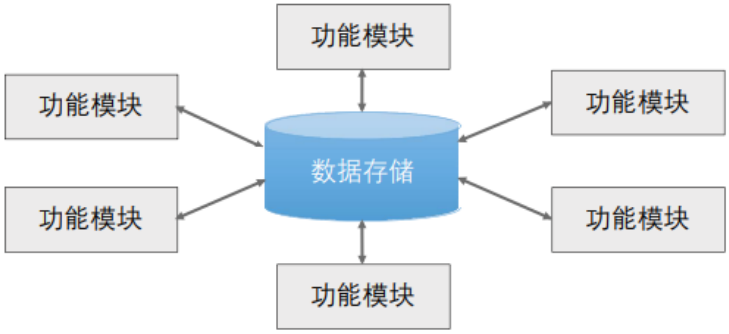
        过程调用、中断、I/O、事件、进程、线程、共享、同步、并发、消息、远程调用、动态连接、API 等等
        - 同步
        - 异步
      - 协议
        - 目的：使双方能够互相理解对方所发来的信息的语义
        - 对过程调用来说：参数的个数和类型、参数排列次序
        - 对消息传送来说：消息的格式

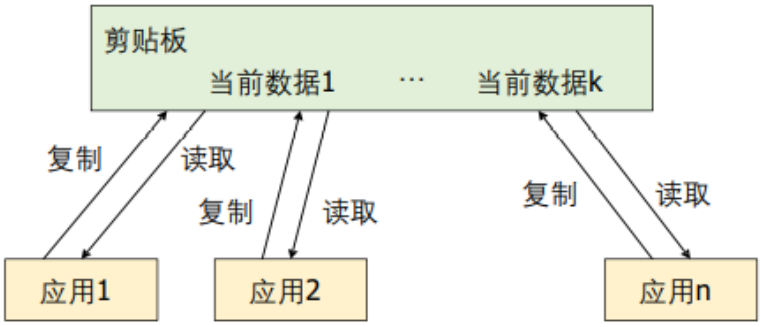
    - 约束
        - 高层次的软件元素可以向低层次软件元素发出请求，低层次软件元素完成计算后向高层次发送服务应答，反之不行
        - 每个软件元素根据其职责位于适当的层次，不可错置，如核心层不能包含界面输入接收职责
        - 每个层次都是可替换的，一个层次可以被实现了同样的对外服务接口的层次所替代
- 软件体系结构的目标
    - 软件体系结构关注的是如何将复杂的软件系统划分为模块、如何规范模块的构成和性能、以及如何将这些模块组织为完整的系统
    - 主要目标：建立一个一致的系统及其视图集，并表达为最终用户和软件设计者需要的结构形式，支持用户和设计者之间的交流与理解。
- 四种设计观
    - 分解与综合
    - 搜索
    - 讨论
    - 情景设计
- 软件体系结构风格
    - **描述特定邻域中软件系统家族的组织方式的惯用模式**，反映了邻域中众多系统所**共有的结构和语义特性**，并知道如何将各个模块和子系统有效地组织成一个完整的系统
    - 数据流风格





        - 管道-过滤器风格
        - 把系统任务分成若干连续的处理步骤，这些步骤由通过系统的数据流连接，一个步骤的输出就是下一个步骤的输入
        - 每个过滤器独立于其上游和下游的构件而工作，**过滤器的设计要针对某种形式的数据输入，并且产生某种特定形式的数据输出（到下一个过滤器）**
    - 以数据为中心的风格（仓库）



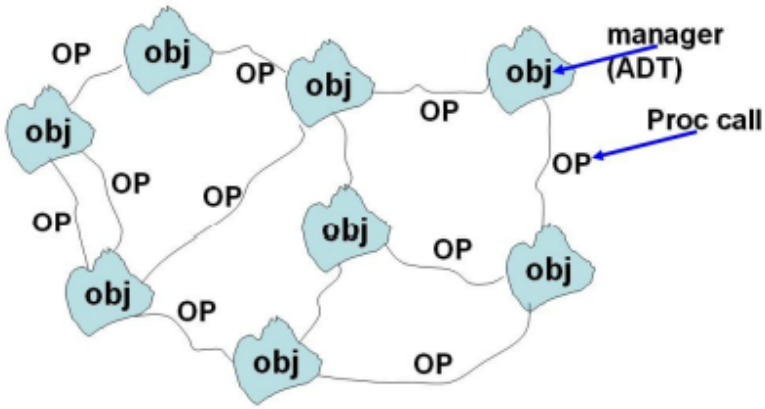


- 数据存储位于这种体系结构的中心，其他构件会经常访问该数据存储，并对存储中的数据进行更新、增加、删除或者修改
- 例：剪贴板、注册表

- 调用和返回体系结构风格

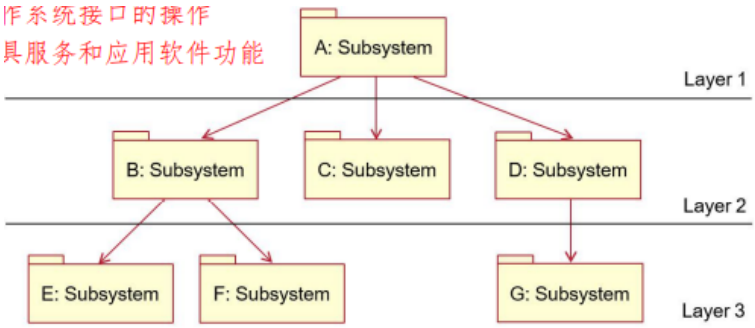
- 本质：**将大系统分解为若干模块，主程序调用这些模块实现完整的系统功能**
- 主程序-子过程
- 构件：主程序、子程序
- 连接器：调用-返回机制
- 拓扑结构：层次化结构

- 面向对象体系结构风格



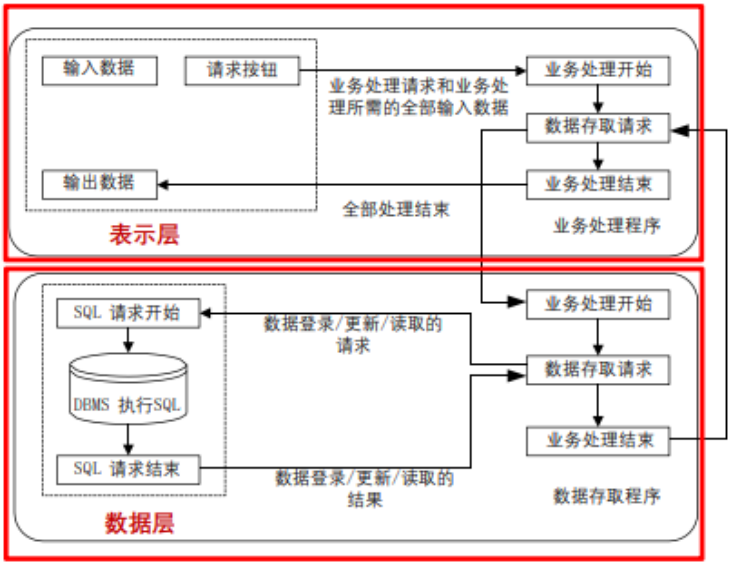
- 系统为对象集合，对象有功能集合
- 数据及作用在数据上的操作被封装成抽象数据类型
- 只通过接口与外界交互，内部的 设计决策则被封装起来
- 构件：类
- 连接件：类之间函数调用、消息传递

- 层次体系结构风格

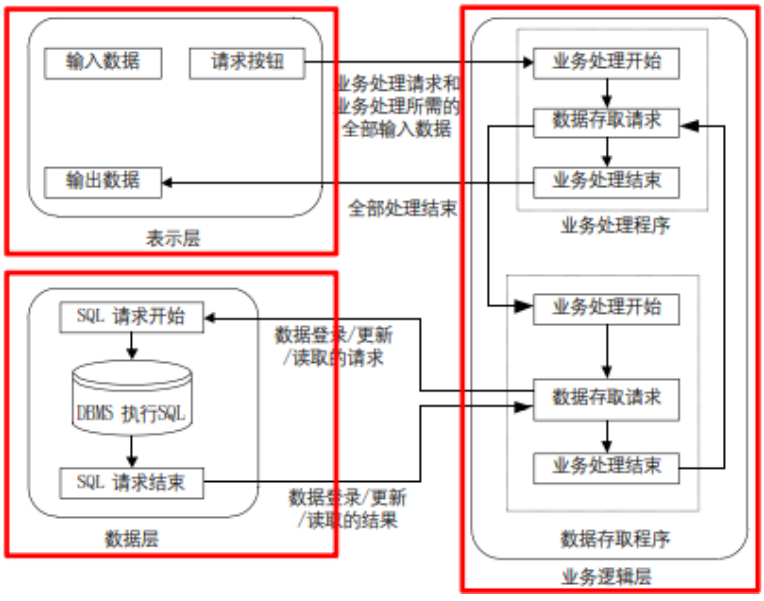


  - 层次系统中，系统被组织成若干个层次，每个层次由一系列构件组成
    - 外层：构件建立用户界面
    - 中层：提供各种使用工具服务和应用软件功能
    - 内层：构件完成建立操作系统接口的操作
  - 层次系统的优点
    - 允许将一个复杂问题分解成一个增量步骤序列
    - 允许每层使用不同方法实现，支持软件复用
    - 严格分层和松散分层
      - 严格分层
        - 要求严格遵循分层原则， 限制一层中的构件只能与对等实体以及 与它紧邻的下面一层进行交互
        - 优点：修改简单
        - 缺点：效率低下
      - 松散分层
        - 松散的分层应用程序放宽了此限制，它 允许构件与位于它下面的任意层中的组 件进行交互

    - 优点：效率高
    - 缺点：修改困难
- 客户机/服务器（C/S）

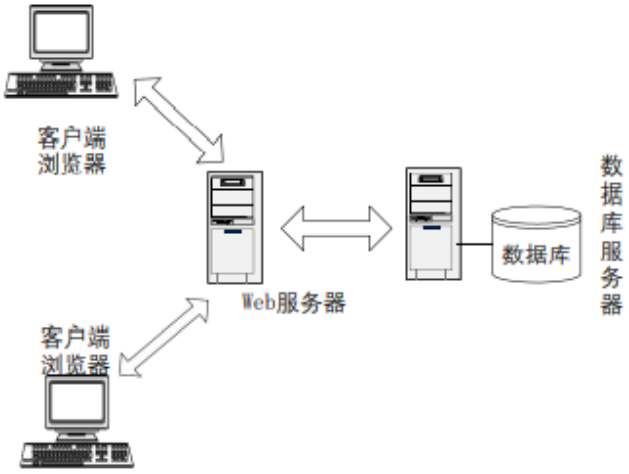


    - 一个应用系统被分为两个逻辑上 分离的部分，每一部分充当不同的角色、完成不同的功能，多台计算机共同完成统一的任务
    - 客户机（前端）：用户交互、业务逻辑、与服务器通讯的接口
    - 服务器（后端）：与客户机通讯的接口、业务逻辑、数据管理
- 三层客户机/服务器



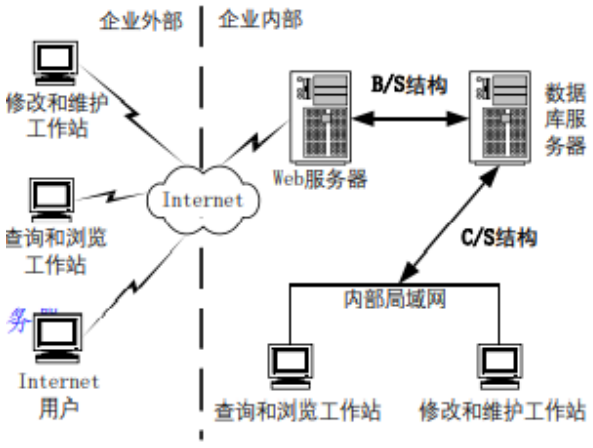
在客户端与数据库服务器之间增加了一个中间层

- 胖客户端与瘦客户端
    - 胖客户端：客户端执行大部分的数据处理操作
    - 瘦客户端：客户端具有很少或没有业务逻辑
- 浏览器/服务器（Browser/Server）是四层C/S分割的一种实现



- 表现层：浏览器
- 逻辑层：Web服务器
- 逻辑层：应用服务器
- 数据层：数据库服务器
- 特点
  - 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，系统维护成本低
  - **客户端无任何业务逻辑**
  - **良好的灵活性和可扩展性**
  - 是**瘦客户端**，具备较高稳定性、延展性和执行效率
  - B/S将服务集中在一起管理，统一服务于客户端，具备良好的**容错能力和负载平衡能力**

- C/S + B/S 混合模式



  - 企业内部用户通过局域网直接访问数据库服务器
    - C/S 结构
    - 交互性增强
    - 数据查询与修改的相应速度高
  - 企业外部用户通过 Internet 访问 Web 服务器/应用服务器
    - B/S 结构

- 用户不直接访问数据，数据安全