

# 数据结构与算法（史上总结最全的笔记）

有馬 2020-08-13 (6,546) 阅读34分钟

关注

## 算法

### 算法基本特征

- 算法：指解题方案的准确而完整的描述（≠程序 ≠计算方法）
- 算法四个特点：**可行性 确定性 有穷性 足够的情报**

程序的设计不可能优于算法的设计

有穷性：算法程序的运行时间是有限的，需在有限步骤后终止

### 算法的基本要素

- 对数据对象的运算和操作：**算术运算、逻辑运算、关系运算、数据传输**
- 算法的控制结构
  - 算法中各操作之间的执行顺序
  - 描述算法的工具：**传统流程图、N-S结构化流程图、自然描述语言、伪代码描述、程序**
  - 一个算法可以用**顺序、选择（分支）、循环（重复）**三种基本结构组合而成
- 算法的复杂度
  - **时间复杂度：**执行算法所需要的**计算工作量**（基本运算次数），与所用计算工具无关，

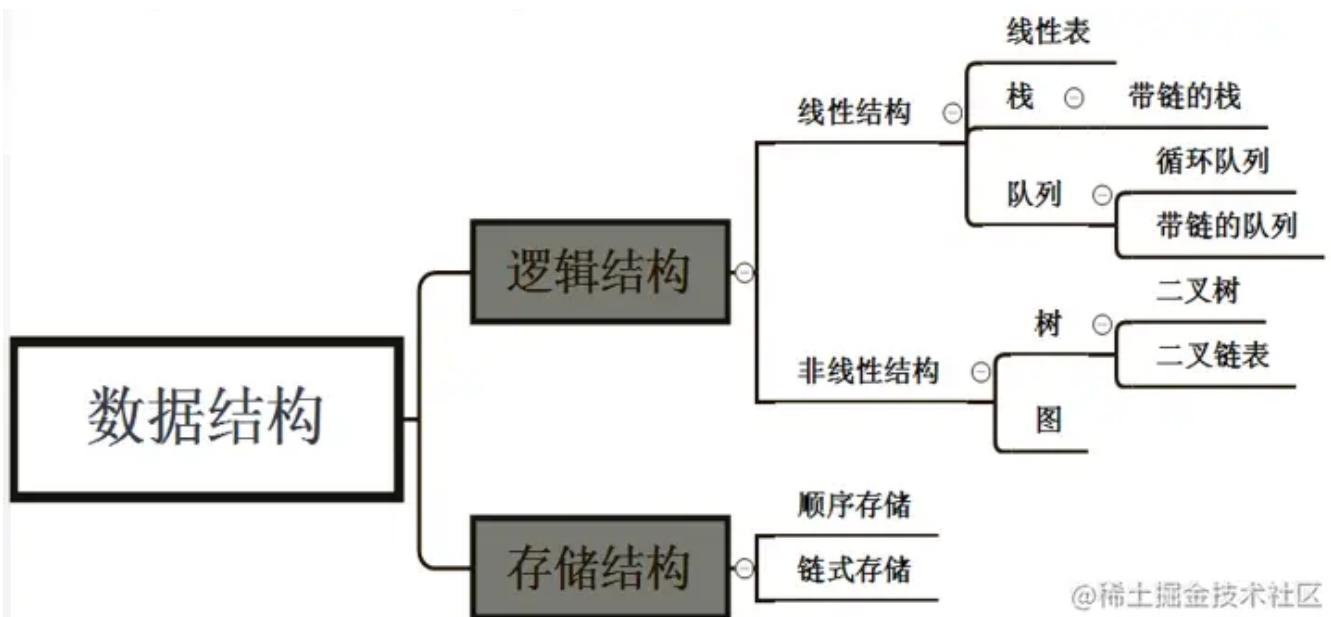
设计算法时需要考虑算法的时间和空间复杂度，但两者相互独立，毫无关联

- 算法分析方法：平均性态、最坏情况复杂性
- 算法分析目的：降低算法复杂度，提高算法的执行效率，以求改进
- 算法设计方法：列举法、归纳法、递推、递归、减半递推、回溯

算法的执行效率与数据的存储结构 **有关**

算法强调动态的执行过程，不同于静态的计算公式

## 数据结构



## 数据

- 数据：需要处理的数据元素的集合
- 数据元素：数据的基本单位，即数据集合中的个体，也是结点

有时一个数据元素可由若干 **数据项(Data Item)** 组成，数据项是数据的 **最小单位**

- 结构：集合中各数据元素之间存在的某种前后件关系

【真题】设数据集合为  $D=\{1, 2, 3, 4, 5\}$ , 则结构  $B=(D, R)$  中为非线性结构的是

$R=\{(1,2), (2,3), (4,3), (3,5)\}$  非线性 【 $1 \rightarrow 2 \rightarrow 3 \rightarrow 5, 4 \rightarrow 3$ 】 (✓)

$R=\{(1,2), (2,3), (3,4), (4,5)\}$  线性 【 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ 】 (✗)

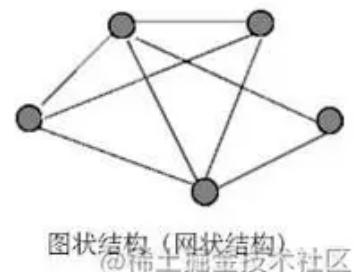
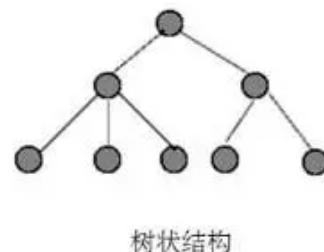
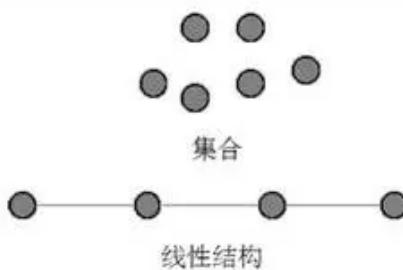
## 数据结构的分类

数据结构作为计算机的一门学科，主要研究三个方面：**数据间固有逻辑结构关系**、**数据的存储结构关系**、**数据结构的运算**

- 逻辑结构：反映数据元素之间的前后件逻辑关系的数据结构（与所使用的计算机无关）
  - 线性结构（线性表、栈、队列）
  - 非线性结构（树、图、二叉链表）
- 存储结构：即物理结构，数据的逻辑结构在计算机空间中的存放方式（不同的存储结构，数据处理的效率不同，**与效率有关**）
  - 顺序结构：主要用于线性的数据结构
  - 链式结构：每一个结点至少包含一个指针域，用指针的指向来体现数据元素之间在逻辑上的联系，优点是**便于插入和删除操作**
  - 索引结构：带有目录与内容

没有根结点或没有叶子结点的数据结构一定是非线性结构。

所有数据结构不用必须有根结点或终端结点。



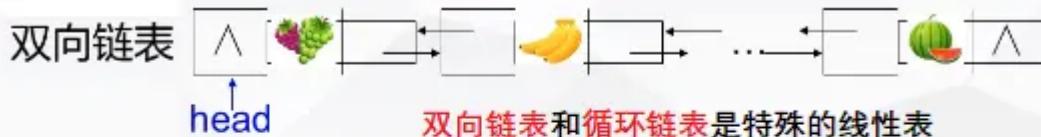
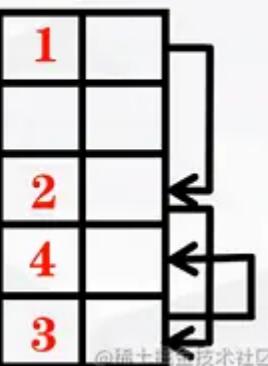
## 顺序存储结构

1 2 3 4

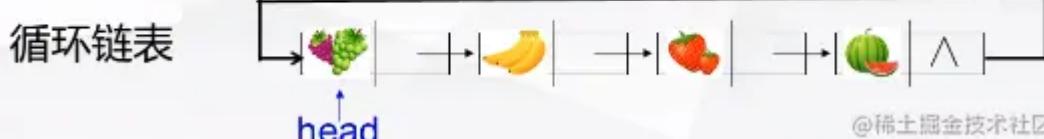


1. 一种逻辑结构可以有多种存储结构。
2. 不同的存储结构其数据处理的效率不同。

## 链式存储结构



双向链表和循环链表是特殊的线性表



@稀土掘金技术社区

- 运算：插入、删除、查找、排序

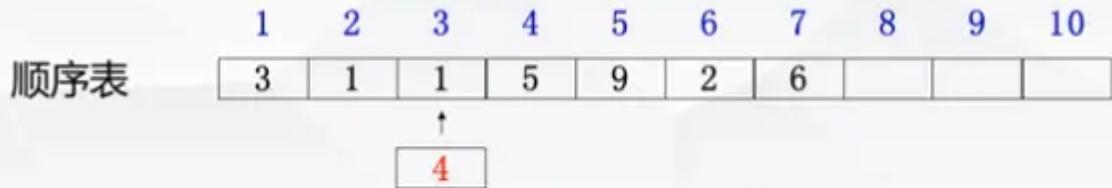
## 线性表

- 线性表：由  $n(n \geq 0)$  个数据元素构成的有限序列，表中由且只有一个根结点和一个终端结点，除根元素外的其它元素有且只有一个前件，除终端元素外的其它元素有且只有一个后件（如：春 → 夏 → 秋 → 冬）
- 线性表的顺序存储结构叫做 **顺序表**（随机存取），线性表的链式存储结构叫做 **线性链表**（顺序存取）

## 顺序表

2. 线性表中数据元素在存储空间中是 按逻辑顺序依次存放 的
3. 可以 随机访问 数据元素
4. 做插入、删除时需移动大量元素，因此线性表不便于插入和删除元素
5. 其存储空间连续，各个元素所占字节数相同，元素的存储顺序与逻辑顺序一致

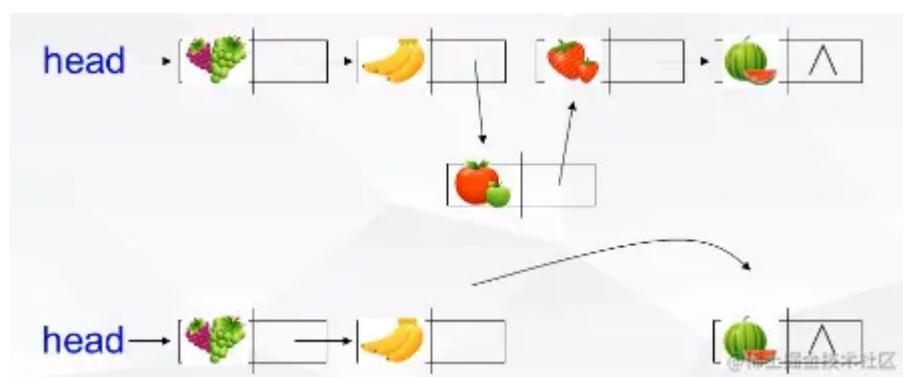
## 顺序线性表的插入



@稀土掘金技术社区

## 线性链表

1. 各数据结点的存储空间可以 不连续
  2. 各数据元素的存储顺序与逻辑顺序可以不一致， 可任意
  3. 所占存储空间 大于 顺序存储结构（每节点多出至少一个指针域）
  4. 查找结点时要比顺序存储 慢
  5. 插入删除元素比顺序存储 灵活
- 线性链表的操作：在线性链表中进行插入与删除， 不需要 移动链表中的元素

**相关**
[稀土掘金](#) [首页](#) ▾

探索稀土掘金

编程  
轨迹

1. 栈的入口和出口是 同一个口， 只能在 栈顶 进行 插入和删除
2. 栈的修改原则是 “先进后出” 或 “后进先出”
3. 栈的 栈底指针bottom 和 栈顶指针top， 从 入栈 到 栈满 再到 退栈， 栈底指针bottom不变， 栈中元素随栈顶指针的变化而动态变化（指针存放的是地址而非数据）
4. 栈能临时保存数据， 具有 记忆功能
5. 栈支持子程序调用



一个栈的初始状态为空。将元素abcde依次入栈，不可能的出栈顺序为 (E)

A.edcba B.dcbae C.badce D.cbaed E.eabcd

用一个长度为50的数组(数组元素的下标从0到49)作为栈的存储空间，如果  
bottom=49, top=30(数组下标)，则栈中具有的元素个数为 【20】 ( $49-30+1=20$ )

设栈的存储空间为S (1: 60)，初始状态为top=61。现经过一系列正常的入栈与退栈操作后，top=25，则栈中的元素个数为 【36】 ( $60-25+1=36$ )

栈内元素个数计算：  $|Top - Bottom| + 1$  (其中Bottom $\geq$ 1)， 若T=B=0说明栈空

链式栈较特殊，当中存放元素与地址，但其栈底指针可以改变但不能省略

与顺序栈相比的优点：入栈操作时不会受栈存储空间的限制而发生溢出，不受空间大小的限制，不需要考虑栈满的问题

## 队列

- 队列中 **队头指针front** 指向对头元素的 **前一位置**， **队尾指针rear** 指向最末元素，从 **入队** 到 **出队**
- 队列的入口和出口 **非同一个口**，只允许在 **队尾插入**，而在 **队头删除**
- 队列的修改原则是 **“先进先出”** 或 **“后进后出”**（**先到先服务的作业调度**）
- 队列中元素随**front**和**rear**的变化而动态变化，并非固定

## 循环队列

- 将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间，供队列循环使用

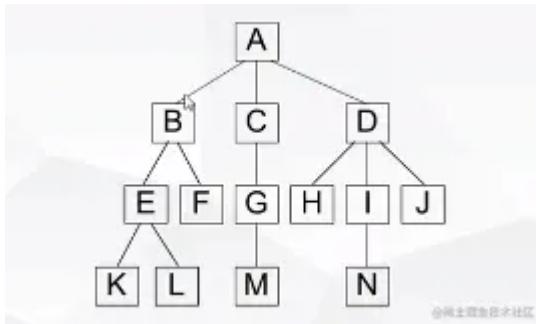


求队列中的 **元素个数s** （由**rear**与**front**共同决定）

- $\text{rear} > \text{front}$   $s = \text{rear} - \text{front}$  .  $\text{rear} < \text{front}$   $s = \text{rear} - \text{front} + \text{总容量}$  .  $\text{rear} = \text{front}$   $s = 0$  或者  $s = \text{容量(满)}$

设循环队列的存储空间为Q (1:40)，初始状态为**front=rear=40**。经过一系列正常的入队与退队操作后，**front=rear=15**，此后又正常地退出了一个元素，则循环队列中的元素个数为 **【39】**（当头尾指针都是15之后还能退出元素，说明初始状态为满，故退出一个元素，即 $40-1=3$ ）

- 树：指  $n(n > 0)$  个元素的有限集合，它有且仅有一个称为根的元素，其余元素是互不相交的子树



- 概念术语：

- 父结点（A是BCD的父）、子结点（BCD是A的子）
- 根结点（A）、叶子结点（KLFMHNJ）
- 结点的度：一个结点所拥有的后件的个数（A的度为3、B的度为2）
- 树的度：具有结点中最大的度（上图为3）
- 树的深度：整棵树的层数（上图为4）
- 子树：在一棵树中以某个结点的一个子结点为根所构成的树（如B→EF→KL这一左半部分）

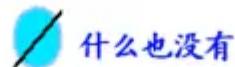
## 二叉树

二叉树是一个有限的结点集合，该集合或为空，或由一个根结点及其两棵互不相交的左右二叉子树所组成，**二叉链表** 是树的二叉链表实现方式。

二叉树有五种基本形态，如下图所示：

空二叉树：

只有一个结点的二叉树



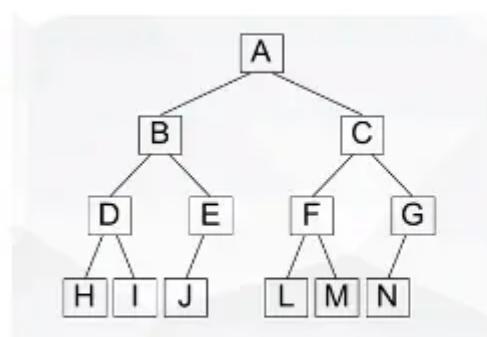
只有左子树的二叉树



只有右子树的二叉树



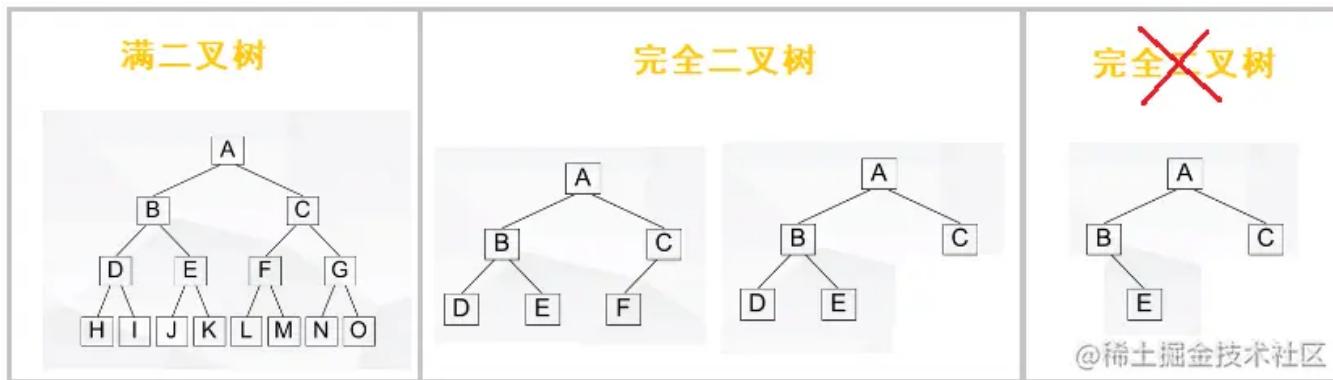
左右子树双全的二叉树



- 二叉树的特点

1. 非空二叉树只有一个根结点
2. 每一个结点最多有两棵子树，且分别称为该结点的左子树与右子树

- 特殊二叉树



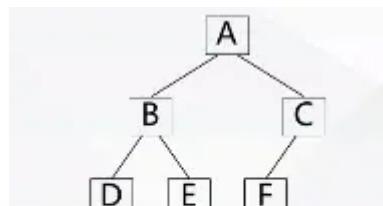
满二叉树：除最后一层外，每一层上的结点数均达到最大值。

完全二叉树：除最后一层外，每一层上的结点数均达到最大值，在最后一层上只缺少右边的若干结点

满二叉树 是 完全二叉树，完全二叉树 不是 满二叉树

- 二叉树的性质：

1. 非空二叉树只有一个根结点，每个结点最多有两棵子树，分别称为 **左子树** 和 **右子树**
  2. 在二叉树的第k层上，最多有  $2^{(k-1)}$  个结点（指定第几层求结点）
  3. 深度为m的二叉树最多有  $2^{(m)-1}$  个结点（指定层数求结点）
  4. 度为0的结点称为 **叶子结点**，度=0的结点总比度=2的结点 多1个
  5. 有n个结点的二叉树深度至少为  $\lceil \log_2 n \rceil + 1$  （指定结点求深度）
- 二叉树的遍历：按照一定的顺序不重复不遗漏地访问二叉树中的结点



前序遍历：访问根结点→前序遍历左子树→前序遍历右子树（ABDGECF）（根左右） 中序遍历：中序遍历左子树→访问根结点→中序遍历右子树（DGBEACF）（左根右） 后序遍历：后序遍历左子树→后序遍历右子树→访问根结点（GDEBFCA）（左右根）

## 查找技术

- 顺序查找：从第一个或最后一个记录开始，一直到查找出给定值，结束查找
- 二分查找：在有序表中取中间记录作为比较对象，若给定值与中间记录的关键字相等，查找成功；若小于则找左半区，若大于则找右半区，不断重复直到成功

查找技术	时间复杂度	适用数据结构
顺序查找	n	无序表、有序与有序链式线性表
二分查找	$\log_2 n$	顺序存储的有序表

@稀土掘金技术社区

注意：即使是有序线性表，如果采用链式存储结构，也只能用顺序查找

## 排序

- 冒泡排序：从头到尾重复比较相邻的两元素，如果前比后大，就交换彼此，直到没有相邻元素需要交换则完成排序，越大的元素会经交换慢慢“浮”到数列的后端
- 快速排序：选第一个数作为基准数，将之后所有的数与其做比较，比基大放右区，比基小放左区，此时线性表被分割为两个子表，再对两个子表重复上述步骤，直到各子表的长度为1则完成排序
- 简单插入排序：将无序序列中的各元素依次插入到已经有序的线性表中，像军训时按身高逐个纠正排序
- 希尔排序：将整个无序序列分割成若干小的子序列分别进行简单插入排序
- 简单选择排序：扫描整个线性表，从中选出最小的元素，将它交换到表的最前面，以此类推直到所有元素均排序完毕
- 堆排序：选建堆，然后将堆顶元素与堆中最后一个元素交换，再调整为堆

类别	排序方法	最坏情况(时间复杂度)
交换类	冒泡排序	$n(n-1)/2$
	快速排序	$n(n-1)/2$
插入类	简单插入排序	$n(n-1)/2$
	希尔排序	$n^{1.5}$
选择类	简单选择排序	$n(n-1)/2$
	堆排序	$n \log_2 n$

@稀土掘金技术社区

最坏情况下，\*\*比较次数最少（时间复杂度最低）\*\*的是 **堆排序**

要求内存量最大的是 **归并排序**

当数据表中每个元素距其最终位置不远，最节省时间的是 **简单插入排序**

选择排序与冒泡排序的区别：冒泡排序通过依次交换相邻两个顺序不合法的元素位置，从而将当前最小（大）元素放到合适的位置；而选择排序每遍历一次都记住了当前最小（大）元素的位置，最后仅需一次交换操作即可将其放到合适的位置。

## 程序设计基础

### 程序设计方法与风格

- 相关术语

- 程序：将计算机语言代码依据一定的语法规则，描述为完成特定任务的算法的指令序列，程序执行的效率与数据的存储结构密切相关
- 程序设计：程序设计为完成一项程序工作的过程
- 计算机语言：计算机语言是人与计算机交流的工具
- Wirth公式：算法+数据结构=程序

- 良好的程序设计风格：**清晰第一，效率第二**

- 选择标识符的名字
- 程序的视觉组织
- 程序的注释
  - 功能性注释：位于模块内部，用于描述其后语句或程序主要功能
  - 序言性注释：位于模块首部，用于说明模块的相关信息（程序的标题 功能的说明 主要的算法 模块接口 开发历史 程序设计者 复审者 复审日期 修改日期）

## 2. 数据说明

- 数据说明的次序规范化
- 说明语句中变量安排有序化
- 使用注释来说明复杂数据的结构

## 3. 语句的结构

- 在一行内只写一条语句
- **程序编写应优先考虑清晰性**
- **程序编写要做到清晰第一，效率第二**
- 在保证程序正确的基础上再要求提高效率
- 避免使用临时变量而使程序的可读性下降
- **避免不必要的转移**
- **尽量使用库函数**
- 避免采用复杂的条件语句
- 尽量减少使用“否定”条件语句
- 数据结构要有利于程序的简化
- **要模块化，使模块功能尽可能单一化**
- **利用信息隐蔽，确保每一个模块的独立性**
- 从数据出发去构造程序
- 不要修补不好的程序，要重新编写

## 4. 输入和输出

- 对输入数据检验数据的合法性
- 检查输入项的各种重要组合的合法性
- 输入格式要简单，使得输入的步骤和操作尽可能简单
- 输入数据时，应允许使用自由格式
- 应允许缺省值
- 输入一批数据时，最好使用输入结束标志
- 在以交互式输入/输出方式进行输入时，要在屏幕上使用提示符明确提示输入的请求，同时在数据输入过程中和输入结束时，应在屏幕上给出状态信息

良好程序设计风格不仅有助于提高程序的可靠性、可理解性、可测试性、可维护性和可重复性，而且也能够促进技术的交流，改善软件的质量

## 结构化程序设计

- 软件设计的基本原则：抽象、信息隐蔽、模块化、局部化、确定性、一致性、完备性、可靠性
- 结构化程序设计风格：**程序结构良好、易读性好、易测试、易维护**（最强调的是易读性）
- 结构化程序设计原则
  - **自顶向下**：先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标
  - **逐步求精**：对复杂问题，先设计一个目标作为过渡，然后逐步细化
  - **模块化**：把程序要解决的总目标分解为一个一个的模块
  - **限用goto语句**：程序的质量与goto语句数量成反比
- 结构化程序的基本结构：**顺序、选择（分支）、循环（重复）**
- 程序设计语言的基本成分：**数据成分、运算成分、控制成分、传输成分**

设计程序时，应采纳的原则之一是程序结构应有助于读者理解（强调**程序易读性**）

## 面向对象的程序设计

- 面向对象方法的主要优点
  1. 与人类习惯的思维方法致
  2. 稳定性好
  3. 可重用性好
  4. 易于开发大型软件产品
  5. 可维护性好

与传统的的面向过程的方法不同之处：

## 描述客观事物

- 对象有关概念术语

- 对象：在现实世界中，每个实体都是对象（大学生、汽车、电视机、空调）
- 属性：用于描述对象的状态
- 方法：用于描述对象的行为
- 类：一组具有相同属性和相同操作的对象的集合
- 消息：是一个实例与另一个实例之间传递的信息，**对象间的通信靠消息传递**
  1. 接收消息的对象的名称
  2. 消息标识符，也称消息名
  3. 零个或多个参数

- 对象的基本特点

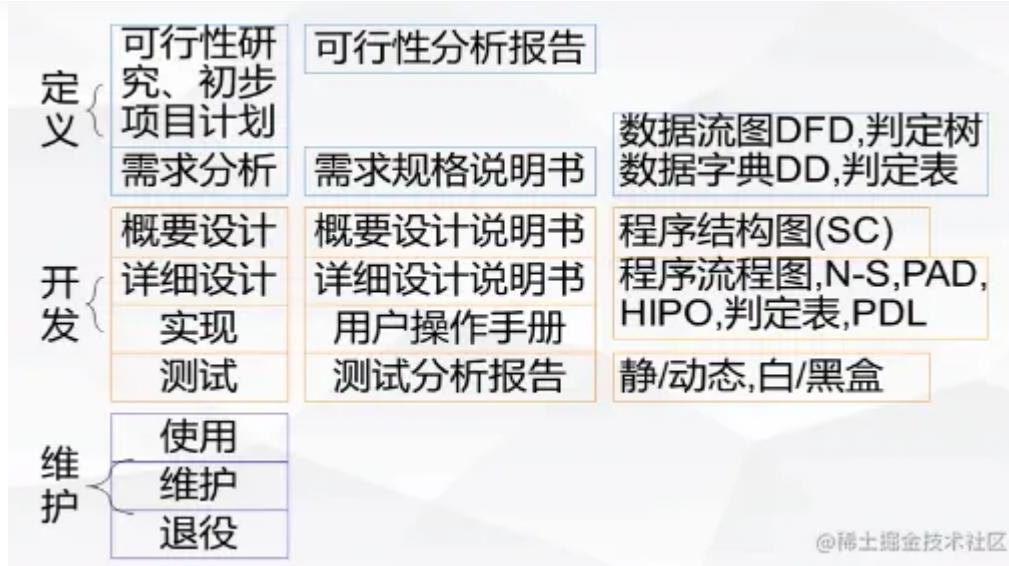
1. **标识唯一性**：对象可由内在本质来区分，而不是通过描述来区分
2. **分类性**：可以将具有相同属性和操作的对象 **抽象** 成类
3. **多态性**：同一操作可以是不同对象的行为，同样的消息被不同对象接受时可导致完全不同的行为的现象
4. **封装性**：从外面看不到对象的内部，只能看到对象外部特征，**实现信息隐蔽**，是**数据与操作的结合**，是**属性与方法的封装**
5. **模块独立性好**：对象是面向对象的软件的基本模块，**高内聚低耦合**
6. **继承性**：使用已有的类建立新类的定义技术，能直接获得已有的性质，而不必重复定义他们，是**类之间共享属性和操作的机制**
7. **依赖性**：某个对象的功能依赖于另外的某个对象，而被依赖的对象只是作为一种工具在使用，而并不持有对它的引用

“**对象是属性和方法的封装体**”、“**任何对象不一定有继承性**”

“**对象是类的具体实例，类是对象的抽象**”、“**操作是对象的动态属性**”

基于同一类产生的对象可以分别设置各自的属性，对象中的属性只能通过该对象所提供的操作来存取或修改

# 软件工程基础



## 软件工程基本概念

- 软件：指计算机系统中与硬件相互依存的另一部分，包括程序、数据和相关文档的完整集合，特点是
  1. 软件是一种逻辑实体，具有抽象性；
  2. 软件的生产与硬件不同，它没有明显的制作过程；
  3. 软件在运行、使用期间不存在磨损、老化问题；
  4. 软件的开发、运行对计算机系统具有依赖性，受计算机系统的限制，这导致了软件移植的问题；
  5. 软件复杂性高，成本昂贵；
  6. 软件开发涉及诸多的社会因素。
- 软件分类：
  - 系统软件：操作系统、编译程序、汇编程序、网络软件、数据库管理系统
  - 应用软件：事务处理软件、工程与科学计算软件、实时处理软件、人工智能软件、财务管理软件
  - 支撑软件（工具软件）：需求分析工具软件、编译工具软件、测试工具软件、维护工具软件、Word编辑软件

学生成绩管理系统、教务管理系统属于应用软件

- 软件危机：需求增长、开发难控、质量难保、难以维护、成本提高、生产率低

- **软件生命周期**：将软件产品从提出、实现、使用维护到停止使用退役的过程

分为3个时期8个阶段，维护是持续时间最长，花费代价最大的一个时期

1. 软件定义阶段：①问题定义可行性研究、②需求分析
2. 软件开发阶段：③概要设计、④详细设计、⑤实现、⑥测试
3. 软件运行维护阶段：⑦使用、⑧维护

- 软件工程：应用于计算机软件的定义、开发和维护的一整套方法、工具、文档、实践标准和工序

- 目的：提高软件生产率/质量/可维护性，降低软件成本；
- 核心思想：把软件当作一个工程产品来处理
- 软件工程的三要素
  - 方法：完成软件工程项目的技术手段
  - 工具：支持软件的开发、管理和文档生成
  - 过程：支持软件开发的各环节的控制和管理
- 原则：抽象、信息隐蔽、模块化、局部化、确定性、一致性、完备性、可验证性

- 需求分析：确定系统的逻辑模型。参加人员有用户、项目负责人和系统分析员

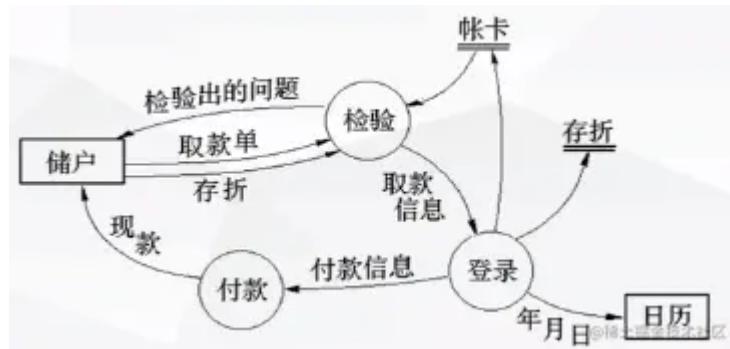
- 需求分析的工作：需求获取、需求分析、编写需求规格说明书、需求评审
- 需求分析阶段产生的主要文档为 **需求规格说明书** (SRS)，其作用为
  1. 便于用户、开发人员进行理解交流
  2. 反映用户问题的结构，可以作为软件开发工作的基础和依据
  3. 作为确认测试和验收的依据
- 说明书的特点：有正确性、无歧义性、完整性、可验证性、一致性、可理解性、可修改性和可追踪性。其中最重要的是无歧义性。

## 结构化分析方法

- 需求分析方法 包括 **结构化需求分析方法**、**面向对象的分析方法**

- 数据流图 (DFD) 是结构化方法的需求分析工具，其图形元素：

- ○ 加工：输入数据经加工变换产生输出
- → 数据流：沿箭头方向传递数据的通道
- = 存储文件（数据源）：存放各种数据的文件
- □ 源（潭）：系统和环境的接口



- 数据字典 (DD)：对数据流图中所有元素定义的集合，它是结构化分析的核心。

## 软件设计

- 软件设计是确定系统的物理模型，软件设计的基本目标是用比较抽象、概括的方式确定目标系统如何完成预定的任务。
- 软件设计的划分（阶段）
  - 按工程管理角度划分：概要设计、详细设计
  - 按技术观点划分：结构设计、数据设计、接口设计、过程设计
- 软件设计基本原理
  - 抽象化：在软件设计中，可以定出多个抽象级别，抽象层次从概要设计到详细设计逐步降低。
  - 模块化：把一个待开发的软件分解成若干小的简单的部分，自顶向下逐层把软件划分成若干模块。
  - 信息隐蔽和局部化：一个模块内的信息，对于不需要这些信息的其他模块来说不能访问。
  - 模块独立性：每个模块只完成独立的子功能，并且与其他模块的联系少且接口简单。模块的独立程度是评价设计好坏的重要度量标准

- 软件模块独立性

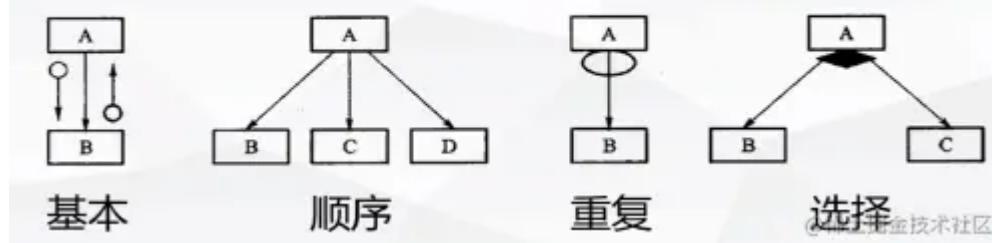
- 高内聚性：指一个模块内部各个元素间彼此结合的紧密程度
- 低耦合性：指模块间互相连接的紧密程度

耦合包括非直接耦合、数据耦合、标记耦合、控制耦合、外部耦合、公共耦合、内容耦合（耦合度逐渐增强）

## 结构化设计方法

- 概要设计的任务：划分出组成系统的物理元素、设计软件的结构
  - 常用工具：[程序结构图 \(SC\)](#)
  - 基本图形：□ 一般模块、○→ 数据信息、●→ 控制信息

### 程序结构图的基本形式：



- 结构图的四种模块类型：传入模块、传出模块、变换模块、协调模块

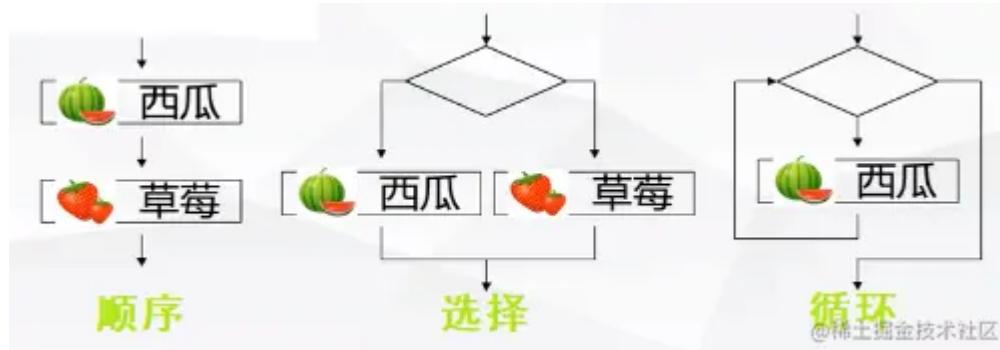
## 系统结构图

- 最大扇入数：系统图中进入某一节点的最大节点数
- 宽度：指整体控制跨度，即横向最大模块数
- 深度：从最顶层出发延伸最长的层数

上图的最大扇入数为n，宽度为5，深度为3

- 详细设计的任务：确立每个模块的实现算法和局部数据结构，用适当方法表示算法和数据结构的细节。

- 图形工具：程序流程图、N-S图、PAD、HIPO
- 表格工具：判定表
- 语言工具：PDL（伪码）
- 程序流程图的基本图形：→ 控制流、□ 加工步骤、◇ 逻辑条件



## 软件测试

- 软件测试的 **目的**：发现程序中的错误
- 软件测试的 **准则**
  1. 所有测试都应追溯到用户需求
  2. 在测试之前制定测试计划，并严格执行
  3. 充分注意测试中的群集现象
  4. 避免由程序的编写者测试自己的程序
  5. 不可能进行穷举测试
  6. 妥善保存测试分析报告，为维护提供方便
- 软件测试的 **方法**
  - 按是否需要执行
    - 静态测试：不实际运行软件，通过人发挥思维优势发现程序的错误
    - 动态测试：基于计算机的测试，是为了发现错误而执行程序的过程
  - 按功能
    - 白盒测试：把测试对象看作一个打开的盒子，利用程序内部的逻辑结构，对程序所有逻辑路径进行测试。白盒测试针对程序的内部逻辑结构

### 方法：逻辑覆盖测试（条件/分支/语句覆盖）、基本路径测试

- 黑盒测试：完全不考虑程序内部的逻辑结构，只检查程序是否能接收输入数据而产生正确的输出信息，黑盒针对程序的外部功能

### 方法：等价类划分法、边界值分析法、错误推测法、因果图

- 软件测试的 **步骤实施**

1. 单元测试：对软件设计的最小单位——模块进行测试，目的是发现各模块内部的错误。
2. 集成测试：把模块按照设计要求组装起来的同时进行测试，目的是发现与接口有关的错误。
3. 确认测试：验证软件的功能和性能是否满足各种需求，以及软件配置是否完全、正确。
4. 系统测试：将软件作为一个元素，与计算机系统其他元素组合在一起，进行集成测试。

【真题】软件测试用例包括——输入数据和预期输出结果

## 程序调试

- 程序调试是对程序进行了成功的测试之后将进入程序调试，通常称为Debug（排错），主要在开发阶段进行。
- 程序调试的 **任务**：诊断和改正程序的错误
- 程序调试的 **基本步骤**
  1. 错误定位
  2. 修改设计和代码，以排除错误
  3. 进行回归测试，防止引进新的错误
- 程序调试的 **方法**：强行排除法、回溯法、原因排除法

## 数据库设计基础

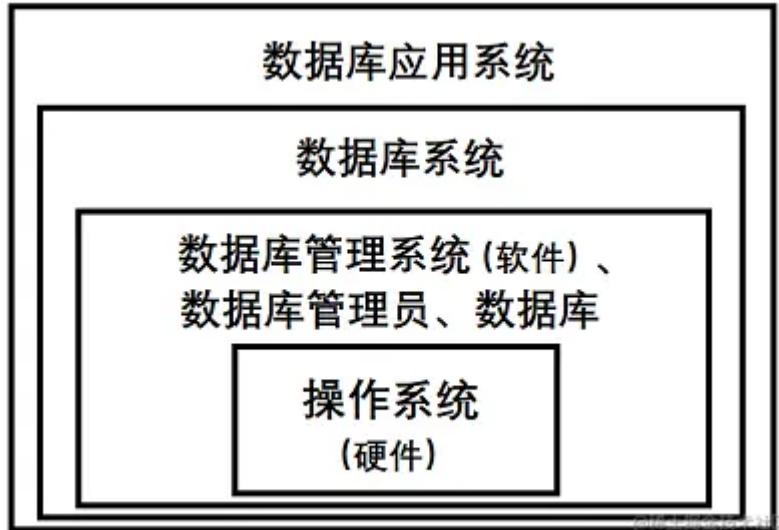
## 数据库基本概念

- 数据 (data) : 描述事物的符号记录
- 数据库 (Database) : 数据的集合, 具有 **统一的结构形式** 并存放于 **统一的存储介质** 内, 是多种应用数据的集成, 并可被各个应用程序所共享

数据库中的数据具有两大特点: “**集成**” 与 “**共享**”

数据库技术的根本目标: **解决数据共享问题**

- 数据库管理系统 (DBMS) : 数据库的机构, 一种 **系统软件**, 负责数据库中的数据组织、数据操纵、数据维护、控制及保护和数据服务, 它是数据库系统的 **核心**
  - 数据定义语言 (DDL, Definition) : 数据模式定义、数据存取的物理构建
  - 数据操纵语言 (DML, Manipulation) : 数据操纵 (包括 **查询** 与增、删、改等操作)
  - 数据控制语言 (DCL, Control) : 数据的完整性、安全性的定义与检查、并发控制、故障恢复
  - 数据查询语言 (DQL, Query) : 集数据定义、操纵和控制功能于一体的数据库语言, 是数据库的核心语言, 具有操作所有关系型数据库管理系统的功能
- 数据库管理员 (DBA) : 主要工作包括 **数据库规划、设计、维护、监视**, 改善系统性能、提高系统效率
- 数据库应用系统 (DBAS) : 利用数据库系统进行应用开发可构成一个数据库应用系统, 它是专门为一类用户设计的系统, 不具有通用性, 由 **数据库系统**、**应用软件** 以及 **应用界面** 组成



- 数据库系统（DBS）的组成
  - 数据库（数据）
  - 数据库管理系统DBMS（软件）
  - 数据库管理员DBA（人员）
  - 软件平台
  - 硬件平台
- 数据库系统的特点：高集成性、 高共享低冗余 、 物理独立性与逻辑独立性 、统一管理与控制

数据独立性是数据与程序间的互不依赖性，即数据的逻辑结构、存储结构与存取方式的改变不会影响应用程序， ==在数据系统中，数据的物理结构并不一定与逻辑结构一致==

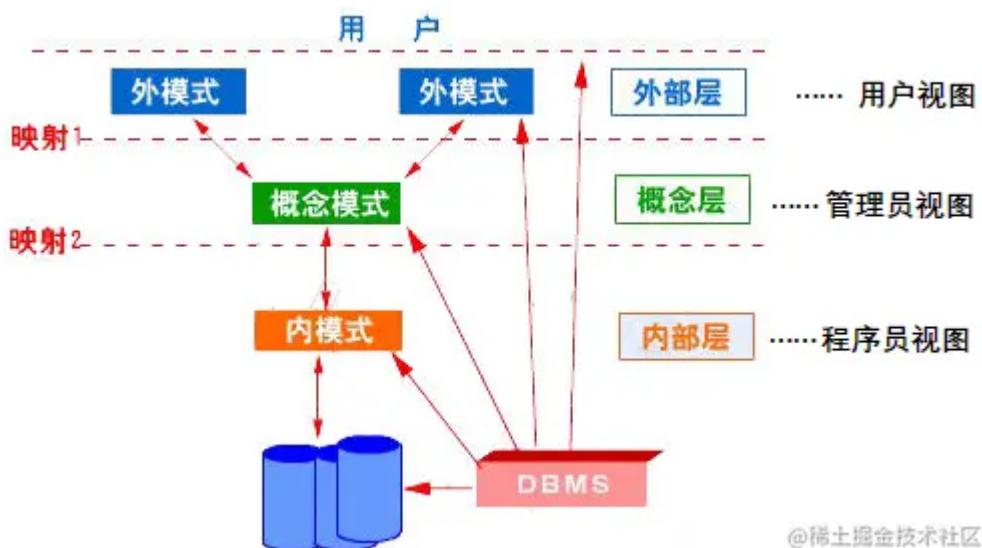
逻辑独立性——指用户的应用程序与数据库的逻辑结构是相互独立的，即当数据的逻辑结构改变时，用户程序也可以不变，当数据库中数据总体逻辑结构发生变化，而应用程序不受影响

- 数据库系统的内部结构体系
  - 三级模式
    - 概念模式（概念数据库）：全局数据逻辑结构的描述，即全体用户的公共数据视图
    - 外模式（用户数据库）：又称子模式或用户模式，即用户的数据视图，反映用户对数据的要求

## 存取路径

- 二级映射

- 外模式/概念模式的映射：实现了外模式到概念模式之间的相互转换（当逻辑模式发生变化时，通过修改相应的外模式/逻辑模式映射，使得用户所使用的那部分外模式不变，从而应用程序不必修改，保证数据具有较高的逻辑独立性）
- 概念模式/内模式的映射：实现了概念模式到内模式之间的相互转换（当数据库的存储结构发生变化时，通过修改相应的概念模式/内模式的映射，使得数据库的逻辑模式不变，其外模式不变，应用程序不用修改，从而保证数据具有很高的物理独立性）



三级模式反映了模式的三个不同环境及其他们的不同要求

两级映射保证了数据库中数据具有较高的逻辑独立性和物理独立性

- 数据管理发展的三个阶段：人工管理 → 文件系统 → 数据库系统

在数据管理技术发展过程中，文件系统与数据库系统的主要区别是数据库系统具有  
特定的数据模型

## 数据模型

▪ 精选文章 · 精选广告 · 精选评论 · 精选问答

- 数据模型的三要素

数据结构：描述数据的类型、内容、性质以及数据间的联系

数据操作：描述在相应数据结构上的操作类型与操作方式

数据约束：描述数据结构内数据间的语法、语义联系，它们之间的制约与依存关系

用计算机表示每一个实体时，由其所有信息项组成一条 **记录**，相应于属性的数据称为 **数据项**

实体内部的联系反映在数据上是数据项之间的联系，实体间的联系反映在数据上是记录间的联系。实体与属性有“型”与“值”之分，型是指结构（对应行），值是指结构约束下的具体取值（对应列）

- 数据模型按不同的应用层次

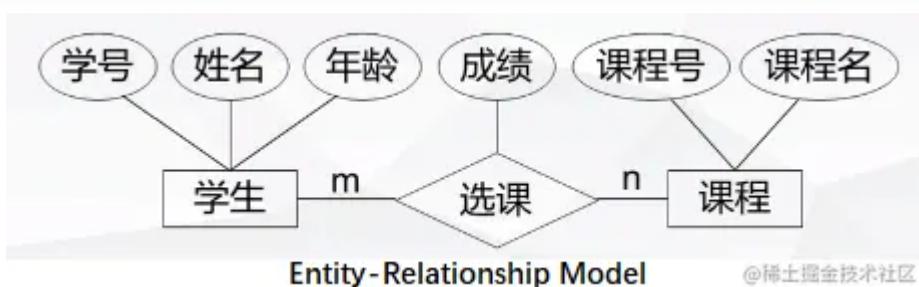
- 概念数据模型（概念模型）：E-R模型、谓词模型
- 逻辑数据模型（数据模型）：层次模型、网状模型、关系模型
- 物理数据模型（物理模型）：面向计算机物理结构的模型

概念数据模型——用于对客观世界中复杂事物的结构及它们之间的联系进行描述，与具体的数据库管理系统无关，与具体的计算机平台无关

逻辑数据模型——面向数据库系统、**考虑数据库实现** 的模型，着重于在数据库系统一级的实现

## 数据模型实例

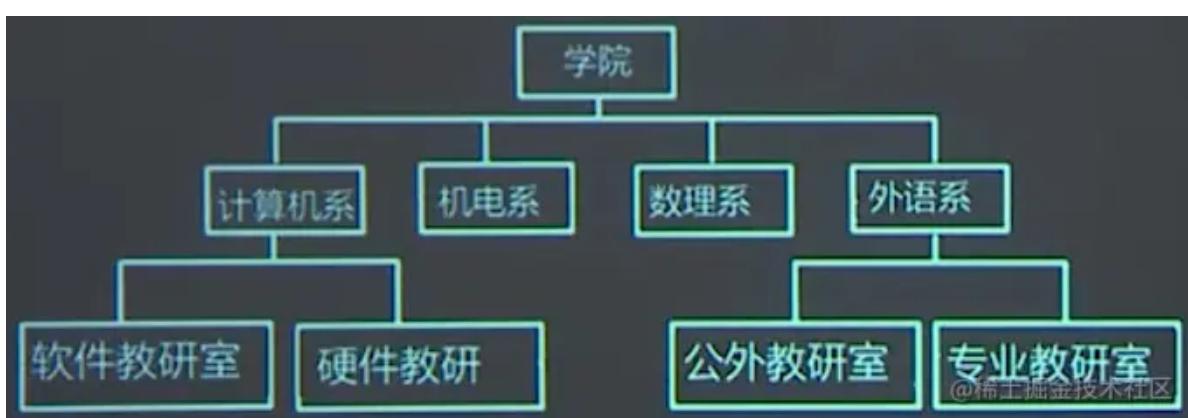
- E-R模型：实体联系模型，由实体、联系、属性组成



E-R模型图形符号： □ 实体 、 ○ 属性 、 ◇ 联系 、 - 联接关系

实体集间的相互关系： 一对—、一对多、多对一、多对多

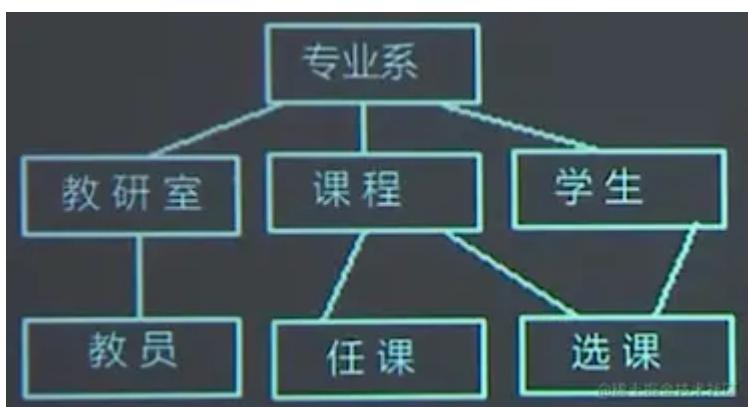
- 层次模型：最早发展起来的数据库模型，如 **树**



层次模型的特点：

- 每棵树有且仅有一个无双亲结点，称为根
- 树中除根外所有结点有且仅有一个双亲

- 网状模型：一个不加任何条件限制的 **无向图**



- 网状模型：一个不加任何条件限制的 **无向图** 其本结构所建立的模型 中本框加乃本元组织构成 本框加由n个命之

层次型、网状型、关系型数据库的划分原则是 **数据之间的联系方式**

## 二维表

<b>主表</b>	<b>主键</b>	<b>属性(字段)</b>			
	<b>学号</b>		<b>姓名</b>	<b>性别</b>	<b>班级</b>
<b>元组(记录)</b>	2007102	张浩然	男	07动画1班	沈阳
	2007103	李一鸣	男	07播音3班	南宁
	2007104	王丽	女	07通信2班	成都

<b>从表</b>	<b>外键</b>			
	<b>学号</b>	<b>语文</b>	<b>数学</b>	<b>英语</b>
	2007102	91	112	79
	2007103	97	109	120

@稀土掘金技术社区

- 二维表的概念：用来表示实体间联系，每一个二维表称为一个 **关系**，为了建立一个关系，**首先要指定关系的属性**
  - 属性：二维表中的一 **列**，每个属性的取值范围称为值域（一个关系的属性名表称为该关系的 **关系模式**，其记法为 **<关系名>(<属性名1>, <属性名2>, ..., <属性名n>)**）
  - 元组：二维表中的一 **行**（各元组的每一个 **分量** 是表示 **最小单位** 的基本数据项，分量不可再分）
  - 主键：也称 **关键字**、**主码**，其值能 **唯一** 标识表中一个元组，主码属性不能取空
  - 外键：也称 **外部关键字**，在一个关系中含有与另一个关系的关键字相对应的属性组，表的外键是另一表的主键，外键可重复、可空值

例：在关系A(S, SN, D)和B(D, CN, NM)中，A的主键是S，B的主键是D，则D是A的外键

- 关系中的数据约束

- **实体完整性约束**：要求关系的主键中属性值不能为空值，为主键是唯一决定元组的，若属性为空值则其唯一性就无意义了，即一个关系中 应有一个或多个 候选关键字
- **参照完整性约束**：这是关系之间相互关联的基本约束， 不允许关系引用不存在的元组，即在关系中的外键要么是所关联关系中实际存在的元组，要么为空值
- **用户定义的完整性约束**：反映某一具体应用所涉及的数据必须满足的语义要求，例如某个属性的取值范围在0-100之间

- 关系的特点

1. 关系必须规范化
2. 在同一个关系中不能出现相同的属性名
3. 关系中不能有相同的元组
4. 在一个关系中 元组的次序 无关紧要，任意交换两行位置并不影响数据的实际含义
5. 在一个关系中 属性的次序 无关紧要，任意交换两列位置也不影响数据的实际含义

## 关系代数

- 关系模型的基本操纵：增加、删除、修改、查询

- **并  $R \cup S$** ：将两个以上表的行并到一起，去除相同的行
- **差  $R - S$** ：在关系R中减去关系S共有的行，保留的是S中没有的行，运算时依次将两个表的一行一行相减
- **笛卡儿积  $R \times S$** ：行与列重新组合
- **投影  $\pi$** ：从所有字段中选取一部分字段及其值进行纵向操作（选列）
- **选择  $\sigma$** ：从二维表的全部记录中把那些符合指定条件的记录挑出来（选行）
- **连接  $R \bowtie S (A \theta B)$** ：将两个关系模式拼接成一个更宽的关系模式，在新的关系上做选择操作，生成的新关系中包含满足联接条件的元组
- **交  $R \cap S$** ：求两个以上表的公共行，前提是参与运算的表的结构相同
- **除  $T = R \div S$  (笛卡儿反运算)**：行与列重新拆分，剩下的合并相同行

关系R		
A	B	C
1	2	3
4	5	6
7	8	9

关系S		
A	B	C
2	4	6
4	5	6

@稀土掘金技术社区

并 RUS

A	B	C
1	2	3
4	5	6
7	8	9
2	4	6

差 R-S

A	B	C
1	2	3
7	8	9

投影  $\pi_C, A(R)$ 

C	A
3	1
6	4
9	7

笛卡尔积 R×S

选择  $\sigma_{B>'4'}(R)$ 

A	B	C
4	5	6
7	8	9

R.A	R.B	R.C	S.A	S.B	S.C
1	2	3	2	4	6
1	2	3	4	5	6
4	5	6	2	4	6
4	5	6	4	5	6
7	8	9	2	4	6
7	8	9	4	5	6

@稀土掘金技术社区

说明：

笛卡尔积——若R有m个元组，S有n个元组，则R×S有m×n个元组

投影——C和A为属性名，说明要选择的列

选择——B&gt;'4'是选择语句的条件，对关系做水平分割，选择符合条件的元组

关系R		
A	B	C
1	2	3
4	5	6
7	2	9

关系S	
D	E
2	4
5	6
7	8

R  $\bowtie_{(2=1)}^{\infty}$  S

A	B	C	D	E
1	2	3	2	4
4	5	6	5	6
7	2	9	2	4

@稀土掘金技术社区

**R**

A	B
a	1
b	2

**S**

H	C
1	x
1	y
3	z

**R x S**

A	B	H	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	1	x
b	2	1	y
b	2	3	z

**R x S**  
B ⊂ H

A	B	H	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	3	z

连接  
@稀土掘金技术社区

**R**

A	B
a	1
b	2

**S**

B	C
1	x
1	y
3	z

**R x S**

A	B	B	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	1	x
b	2	1	y
b	2	3	z

**R x S**

A	B	C
a	1	x
a	1	y

自然连接  
@稀土掘金技术社区

### 除

**R**

A1	A2	A3
a	b	c
d	b	c
a	e	c
a	e	f
d	b	f
a	e	g
a	e	h
a	b	l

**S**

A3
c

**R ÷ S**

A1	A2
a	b
d	b
a	e

(1)

**S**

A3
c
f
g
h

**R ÷ S**

A1	A2
d	b
a	e

(2)

**S**

A3
c
f

**R ÷ S**

A1	A2
d	b
a	e

(3)

**S**

A2	A3
b	c

**R ÷ S**

A1
a
d

(4)  
@稀土掘金技术社区

**R**

A	B	C
a	0	k1
b	1	n1
n	2	x1

**S**

A	B	C
f	3	h2
a	0	k1

**T**

A	B	C
a	0	k1

交  $R \cap S = T$

交运算 能不改变关系表中的属性个数但能减少元组个数

## 数据库设计与管理

- 数据库设计：设计一个能满足用户要求，性能良好的数据库，它是 **数据库应用系统中的核心问题**
  - 基本任务：根据用户对象的信息需求、处理需求和数据库的支持环境设计出数据模式
  - 两种方法
    - 面向数据的方法：以信息需求为主，兼顾处理需求（主流）
    - 面向过程的方法：以处理需求为主，兼顾信息需求
- 数据库设计的步骤：（一般采用生命周期法）
  1. 需求分析阶段——包括信息/处理/安全性/完整性需求，建立数据字典
  2. 概念设计阶段——分析数据间内在语义的关联建立抽象模型，用**E-R图来描述信息结构**但不涉及信息在计算机中的表示
  3. 逻辑设计阶段——把**E-R图转换为关系模式**（实体与联系表示成关系，属性转换成关系中的属性）
  4. 物理设计阶段——对数据库内部物理结构作调整并选择合理的存取路径，以提高数据库访问速度及有效利用存储空间，包括优化数据库系统查询性能的**索引设计**、**集簇设计**和**分区设计**
  5. 编码阶段
  6. 测试阶段
  7. 运行阶段
  8. 进一步修改阶段

- 数据库管理

- 数据库的建立
- 数据库的调整
- 数据库的重组
- 数据库安全性控制与完整性控制
- 数据库的故障恢复
- 数据库监控

标签： 数据结构

## 评论 1



平等表达，友善交流



0 / 1000 [?](#) 发送

最热 最新



用户8679026032741

中序遍历：中序遍历左子树→访问根结点→中序遍历右子树 (DGBEACF) (左根右) ? ?

11月前 点赞 评论

...

## 目录

收起 ^

### 算法

算法基本特征

算法的基本要素

### 数据结构

数据



稀土掘金 首页 ▾

探索稀土掘金



顺序表

线性链表

栈

队列

循环队列

树

[二叉树](#)

查找技术

排序

程序设计基础

程序设计方法与风格

结构化程序设计

面向对象的程序设计

软件工程基础

软件工程基本概念

结构化分析方法

软件设计

结构化设计方法

系统结构图

软件测试

程序调试

数据库设计基础

数据库基本概念

数据模型

数据模型实例

二维表

关系代数

数据库设计与管理

## 相关推荐

 稀土掘金 首页 ▾

探索稀土掘金



编程  
轨迹

144k阅读 · 4.1k点赞

### JavaScript 算法与数据结构

28k阅读 · 1.5k点赞

「算法与数据结构」你可能需要的一份前端算法总结

57k阅读 · 1.3k点赞

### 数据结构与算法 (java)

28k阅读 · 205点赞

面试了十几个高级前端，竟然连（扁平数据结构转Tree）都写不出来

307k阅读 · 4.1k点赞

## 为你推荐

### 前端该如何准备数据结构和算法？

ConardLi 4年前  144k  4.1k  132

前端 算法

### JavaScript 算法与数据结构

IT程序狮 5年前  28k  1.5k  16

前端框架 JavaSc... 算法

### 「算法与数据结构」你可能需要的一份前端算法总结

TianTianUp 3年前  57k  1.3k  60

算法 前端

### 数据结构与算法 (java)

ohcomeyes 5年前  28k  205  7

后端 Java 算法

### 面试了十几个高级前端，竟然连（扁平数据结构转Tree）都写不出来

杰出D 2年前  307k  4.1k  2.1k

前端 算法

### 前端你应该了解的数据结构与算法

幸福拾荒者 5年前  20k  653  35

前端 程序员 算法

### 「算法与数据结构」梳理6大排序算法

TianTianUp 3年前  17k  371  58

算法 JavaSc...

### 恋上数据结构与算法

JackLee666 1年前  9.2k  46  25

算法 数据结构

### 万字总结！CS数据结构与算法的备注

 稀土掘金 首页 ▾

探索稀土掘金



编程  
轨迹

## 通俗易懂的Redis数据结构基础教程

老錢 5年前 75k 942 53

Redis 后端 Java

## 「算法与数据结构」分治算法之美

TianTianUp 3年前 13k 228 17

前端 算法

## 看图轻松理解数据结构与算法系列(双向链表)

超人汪小建 5年前 28k 140 4

算法 后端 深度学习

## 前端数据结构与算法入门篇

幻灵尔依 3年前 11k 158 27

数据结构

## (1.8w字)负重前行，前端工程师如何系统练习数据结构和算法？【上】

神三元 3年前 63k 1.4k 82

JavaScript ECMAScr...

## 基础面试题 — 数据结构与算法

Joelixy 4年前 15k 59 3

算法