第 1 章　基础知识

考纲要求

1．计算机系统的组成和应用领域。

2．计算机软件基础知识

3．计算机网络的基础知识和应用知识

4．信息安全的基本概念

1.1　知识点

1.1.1　计算机发展阶段

以计算机物理器件的变革作为标志，计算机的发展经历了四代：

第一代（1946 年～1958 年）是电子管计算机。代表机型有：ENIAC、IBM650（小型机）、IBM709（大型机）等。

第二代（1959 年～1964 年）是晶体管计算机。代表机型有：IBM7090、IBM7094、CDC7600 等。

第三代（1965 年～1970 年）是集成电路计算机。代表机型有：IBM360 系列、富士通 F230 系列等。

第四代（1971 年至今）是大规模和超大规模集成电路计算机。这个时期，计算机的类型除了小型、中型、大型机之外，开始向巨型机和微型机两个方面发展。

1.1.2　计算机系统的组成

一个完整的计算机系统包括硬件系统和软件系统两个部分。

硬件系统

计算机硬件是组成计算机物理设备的总称，它们由各种器件和电子线路组成，是计算机完成工作的物质基础。

计算机硬件由 5 个部分组成：运算器、控制器、存储器、输入设备和输出设备。

（1）运算器。运算器又称算术逻辑单元（ALU），它接收由存储器送来的二进制代码，并对代码进行算术和逻辑运算。

（2）控制器。控制器是用于控制计算机的各个部件，并按照从存储器取出的指令，向各部件发出操作指令，同时，它接收由各部件传来的反馈信息，并对这些信息进行分析，决定下一步操作。

（3）存储器。存储器是存放源数据、中间数据、程序以及最终结果的部件。它在计算机运行过程中，一方面不停的向运算器提供数据，另一方面又保存从运算器送回的计算结果，存储器还保存程序，且不断的取出指令传送给控制器。

（4）输入设备。输入设备接收用户提交给计算机的源程序、数据及各种信息，并把它们转换成为二进制代码，传送给存储器。

（5）输出设备。输出设备的功能是将计算机内部的二进制信息转换为人和设备能识别的信息。

通常将运算器和控制器合称为中央处理器（CPU）;中央处理器和内存储器合成为主机；输入设备、输出设备和外存储器合称为外部设备，外部设备通过接口线路与主机相连。

软件系统

没有配置任何软件的计算机称为裸机，只有配置了相关系统软件的系统才是完整的计算机系统。

软件分为系统软件和应用软件，系统软件是在计算机上的第一层应用功能扩展。

1.1.3 计算机应用领域

当前计算机的应用已经遍布人类社会各个领域，按照其所涉及的技术内容，计算机应用可以分为几种类型。

科学和工程计算

科学计算也称数值计算。在科学试验和工程设计过程中，经常会遇到各种数学问题需要求解，利用计算机并应用数值方法进行求解是解决这类问题的主要途径，这种应用被称为科学和工程计算，其特点是计算量大，而逻辑关系相对简单。

数据和信息处理

数据处理是指对数据的收集、存储、加工、分析和传送的全过程。

过程控制

过程控制是生产自动化的重要技术内容和手段，它是由计算机对所采集到的数据按一定方法经过计算，然后输出到指定执行机构去控制生产的过程。

辅助设计

计算机辅助设计不仅应用于产品和工程辅助设计，而且还还包括辅助制造、辅助测试、辅助教学以及其他多方面的内容。

人工智能

计算机模拟人脑的过程称为人工智能，人工智能是利用计算机来模拟人的思维的过程，并利用计算机程序来实现这些过程。

1.1.4 计算机分类

根据计算机在信息处理系统中的地位和作用，并且考虑到计算机分类的演变过程和可能的发展趋势，IEEE 提出一种分类方法将计算机分成六类：

巨型计算机（Supercomputer）。巨型计算机也称为超级计算机，它采用大规模并行处理的体系结构，具有极强的运算能力。巨型计算机通常应用在尖端科技研究、重大工程项目研究等领域。世界上仅有少数几个国家研究开发巨型计算机。

小巨型机（Mini Supercomputer）。小巨型计算机也称为小型超级计算机，它的性能接近巨型计算机，但使用了更加先进的大规模集成电路与制造技术，体积小、成本低。价格比巨型计算机便宜许多。

大型机（Mainframe）。大型机或称主干机、主机。它的运算速度快、处理能力强、存储容量大、可扩充性好、通信联网功能完善，并且有丰富的系统软件和应用软件。大型计算机一般落户于大中型企事业单位，由专人管理维护。

超级小型计算机（Super Minicomputer）。超级小型计算机为中小企业所拥有。

工作站（Workstation）。工作站主要应用于有特殊要求的专业领域，如图形工作站等。它具有高速运算能力和很强的图形处理功能。

个人计算机（Personal Computer）。个人计算机也称为个人电脑（PC 机）或微机。个人计算机因为其性能价格比高而得以快速普及和广泛应用。个人计算机可分为台式机和便携机两大类。

1.1.5 计算机语言

计算机语言

计算机语言是一类面向计算机的人工语言，它是进行程序运行的工具，又称为程序设计语言。现有的程序设计语言可分为 3 类：机器语言、汇编语言、高级语言。

机器语言

机器语言是最初级的依赖于硬件的计算机语言。机器语言直接在计算机硬件级上执行，所以效率比较高，能充分发挥计算机高速计算的能力。

汇编语言

用有助于记忆的符号和地址来表示指令的程序设计语言叫做汇编语言。也称为符号语言。用汇编语言编写的程序与机器语言相比，除较直观和易记忆外，仍然存在工作量大，面向机器、无通用性等缺点，所以，汇编语言又称作“ 低级语言 ”。

高级语言

高级语言是一类人工设计的语言，它对具体的算法进行描述，所以又称作为算法语言。高级语言是一类面向问题的程序设计语言，且独立于计算机的硬件，其表达方式接近于被描述的问题，易于人们的理解和掌握。

1.1.6 计算机软件

计算机软件可以分为系统软件和应用软件两种。

系统软件

系统软件一般包括：操作系统、语言处理程序和数据库管理系统以及服务程序等。

操作系统是系统软件的核心，它管理计算机软件、硬件资源，调度用户作业程序和处理各种中断，从而保证计算机各个部分协调有效的工作。

语言处理程序的任务，就是将各种高级语言编写的源程序翻译成机器语言表示的目标程序。语言处理程序按照处理的方式不同，可以分为解释型程序与编译型程序两大类。解释型程序的处理采用边解释边执行的方法，不产生目标程序，称为对源程序的解释执行。编译型程序先将源程序翻译成为目标程序才能够执行，称为对源程序的编译执行。

数据库管理系统是对计算机中所存放的大量数据进行组织、管理、查询并提供一定处理功能的大型系统软件。

服务型程序是一类辅助性的程序，它提供各种运行所需的服务。

应用程序

应用软件是为解决实际应用问题所编写的软件的总称，它涉及到计算机应用的所有领域，各种科学和工程计算的软件和软件包、各种管理软件、各种辅助设计软件和过程控制软件都属于应用软件的范围。

1.1.7 计算机网络

计算机网络的功能

计算机网络具有下列基本功能：

（1）资源共享。其目的是让网络上的用户都能使用网络中的程序、设备，尤其是数据，而不管资源和用户在什么地方。换言之，用户即使是在本地也能使用千里之外的数据。

（2）高可靠性。依靠可替代的资源来提供高可靠性。例如，所有文件可以在两台或三台计算机上进行备份，如果其中之一由于硬件故障不能使用，可使用其他备份。

（3）可用性。当工作负荷增大时，只要增加更多的处理器，就能逐步改善系统的性能。对集中式主机而言，一旦系统能力达到极限，就必须用更强大的主机替代它，而这样做代价大，对用户的影响也大。

（4）实现分布式的信息处理。对于综合的大型问题，可以采用合适的算法，将任务分散到网络中不同的计算机上进行分布处理。多台微型机通过网络可连成具有高性能的计算机系统，使它具有解决复杂问题的能力，而费用大为降低。

（5）提供强大的通信手段。通过网络，两个或多个生活在不同地方的人可以共同起草报告。当某人对联机文档的某处作了修改时，其他人员可以立即看到这个变更，而不必花几天的时间等待信件。这种速度上的提高使得广泛分布的群体之间的合作变得很容易。

网络的分类

计算机网络的分类方法很多，可以从不同的角度进行分类。

（1）从网络的交换功能进行分类：电路交换网、报文交换网、分组交换网和 ATM 网。

（2）从网络的拓扑结构进行分类：星型、环型、总线型和网状型。

（3）从网络的作用范围进行分类：广域网、局域网和城域网或市域网。

1.1.8 数据通信基本原理

所谓的数据通信是指传统的通信技术通过使用计算机来实现信息的传输、交换、存储和处理。现代通信系统由数据传输系统和数据处理系统两部分组成。数据传输系统又称为通信子系统或通信子网，其主要任务是实现不同数据终端设备之间的数据传输；数据处理系统又称为资源子系统或资源子网，它是由许多数据终端设备组成，负责提供信息、接受信息和处理信息。

物理信道按照传输介质的类型可以把信道分为有线信道和无线信道。传输介质是数据传输系统中收方和发方之间的物理路径。有多种物理介质可用于实际传输，每一种物理介质在带宽、延迟、成本和安装维护难度上都不相同。介质可以大致分为有线介质和无线介质。

其中有线介质包括双绞线、同轴电缆、光纤等；无线信道包括微波信道和卫星信道。

所谓基带，就是指信号所固有的基本频带，基带信号通常是由数据直接转换成的、未经频率调制的波形。与基带信号频谱相适应的信道称为基带信道。将数字设备（如计算机）发出的数字信号（即基带信号）直接在信道中进行传输，称为基带传输。

所谓频带传输就是把数字信号调制成音频信号后在电话线路上传输，到达接收端时再把音频信号解调还原成原来的数字信号。在频带传输中，要求在发送端安装调制器，在接收端安装解调器。

用一对传输线同时传送几路信息，称为多路复用。多路复用的典型方式有两类，即频分多路复用和时分多路复用。通过多路复用，可以提高线路的利用率。

所谓频分多路复用，就是将传输线路的总频带划分成若干个子频带，每一个子频带作为一条逻辑信道提供给一对终端使用。频分多路适用于传输模拟信号，多用于电话系统。

波分多路复用是在光纤信道上使用频分多路复用的一个变种。在这种方法中，两条光纤连到一个棱柱，每条光纤的能量处于不同的波段。两束光通过棱柱，合成到一条共享的光纤上，传送到远方的目的地，随后再将它们分解开。

任何两个终端间的通信业务量分布总是非均匀的，建立固定的点到点连接从线路利用效率来说很不经济，特别是当终端数目增加时，要在每对终端间建立起固定的点到点线路就更显得既无必要也不切合实际。解决这个问题的方法就是将各地的终端连到一个具有某种交换能力的交换网络。这个交换网络包括若干条通信线路和交换机，由交换机根据每次通信的要求和网络运行状态动态地选择通信路径。目前在计算机网络中使用的交换技术有如下几种：电路交换、报文交换、分组交换、帧中继和异步转移模式（ATM）。

1.1.9 网络体系结构与 TCP/IP 协议

计算机网络的分层及其所使用的协议的集合，就是所谓的网络体系结构。具体地说，网络体系结构即是层次与协议的集合。体系结构的描述必须包含足够的信息，使实现者可以用来为每一层编写软件和设计硬件，并使之符合有关协议。

网络协议是关于双方通信过程中的一组约定规则，用来建立通信关系，进行数据交换。完整的通信协议相当复杂。为了简化协议的设计，便于协议的实现及维护，大多数网络都将协议按层（Layer）或级（Level）的方式组织。每层都向它的上层提供一定的服务，而将如何实现服务的细节对上层屏蔽，即低层协议对高层而言是透明的。相邻两层之间为层间接口。

国际标准化组织制定了一个开放系统互连参考模型（OSI）。该标准规定，整个网络的通信功能划分为 7 个层次。每一层完成系统信息交换所需的部分功能，通过层间的接口与其相邻层连接，从而实现不同系统之间、不同结点之间的信息交换。

TCP/IP（Transmission Control Protocol/Internet Protocol）传输控制协议／网际协议最初由美国国防部高级研究计划局（Department of Defence Advanced Research Project Agency，DARPA）在 1969 年提出，并把它用在 ARPANET 中。随着 ARPANET 在规模和作用范围的日益扩大，TCP/IP 协议也逐渐完善，最终成为 Internet 的基础，并且应用范围也愈来愈广，几乎已成为广域网和局域网内的标准网络协议。

TCP/IP 是个协议集，根据 OSI 的七层理论，TCP/IP 可以分为四层。分别是接口层、网络层、传输层和应用层。

1.1.10　信息安全基础

信息安全

信息安全的目的，就是要防止非法的攻击和病毒的传播，以保证计算机系统和通信系统的正常运行；就是要保证信息的保密性、完整性、可用性和可控性；就是保证电子信息的有效性。

信息保密

信息保密是信息安全的重要方面，为保密而进行加密是防止破译信息系统中机密信息的技术手段。加密的办法就是使用数学方法来重新组织信息，使除合法接收者外的其他任何人要看懂变化后的数据或信息是非常困难或不可能的。加密前的信息称为明文，加密后的信息称为密文。

信息认证

信息认证是信息安全的另一个重要方面，信息认证，首先是验证信息的发送者的真实性，即不是假冒的；其次是验证信息的完整性，即验证信息在传送或存储过程中未被篡改、重放或延迟等。认证是防止对系统进行主动进攻的重要技术手段。主要的信息认证技术有：数字签名技术、身份识别技术和信息完整性校验技术等。

密钥管理

密钥管理影响到密码系统的安全，而且还会涉及到系统的可靠性、有效性和经济性。

密钥管理包括密钥的产生、存储、装入、分配、保护、丢失、销毁以及保密等内容。其中解决密钥的分配和存储是最关键的技术。

1.1.11 操作系统安全

操作系统应提供的安全服务服务包括：内存保护、文件保护、存取控制和存取鉴别等，以防止由于用户程序的缺陷而损害系统。

操作系统安全方法

一般操作系统的安全措施可从隔离、分层、和内控 3 个方面考虑。隔离是操作系统安全保障的措施之一，它又可以分为：物理隔离、时间隔离、逻辑隔离和密码隔离。

物理隔离是指使不同安全要求的进程使用不同的物理实体。

时间隔离是指使不同的进程在不同的时间运行。

逻辑隔离是指限制程序的存取，是操作系统不能存取允许范围以外的实体。

密码隔离是指进程以其他进程不了解的方式隐蔽数据和计算。

操作系统的安全措施

访问控制是保障信息安全的有效措施，访问控制的目的是：

（1）保护存储在计算机上的需要保护的信息秘密性，通过对访问进行控制，使机密信息保密。

（2）保护存储在计算机内的个人信息保密性。

（3）维护计算机内信息的完整性。拒绝非授权用户访问，减少非法用户对重要文件进行修改的机会。

（4）减少病毒感染的机会，从而减少和延缓病毒的传播。

存储保护是对安全操作系统的基本要求。存储保护保证系统内的任务互不干扰。在多道程序系统中，内存中既有操作系统，也有用户程序，为避免内存中程序相互干扰，必须对内存中的程序和数据进行保护，采用的措施一般有：

防止地址越界，规定每个进程都具有相对对立的进程空间，当进程运行时地址越界，可能侵犯其他进程的空间，从而影响其他进程的正常工作。也可能侵犯操作系统，导致系统混乱，因此必须对进程所产生的地址进行检查，发现地址越界时产生中断，再由操作系统进行处理。

防止操作越权，对于多个进程共享的公共区域，每个进程都享有访问权，如有些进程可执行写操作，而其他进程只能进行读操作，因此需要对公共区域的访问加以限制和检查。

文件保护措施是为了防止由于误操作而对文件造成破坏，文件保密措施是为了防止未经授权的用户对文件的进行访问。

1.2 重点难点

1.2.1 计算机网络的组成及其拓扑结构

计算机网络拓扑的定义

计算机网络设计首先需要解决计算机网络在满足响应时间、吞吐量和可靠性的条件下通过选择合适的线路、线路通信容量、接入方式，实现整个网络的结构合理，成本低廉，为了解决复杂的网络结构设计，人们提出了网络拓扑的概念。

拓扑学是几何学的一个分支，它从图论演化而来。拓扑首先需要将实体抽象成与现实中大小、形状无关的点，将他们之间的线路抽象成线，进而研究点、线、面之间的关系。计算机网络拓扑是通过网中结点与通信线路时间的几何关系表示网络结构，反映出各个试题之间的结构关系。拓扑结构设计是计算机网络设计的第一步，也是实现各种网络协议的基础，他对网络性能、系统可靠性和通信费用都有重大的影响。计算机网络拓扑结构主要是指通信子网的拓扑结构。

网络拓扑的分类方法

网络拓扑可以根据通信子网中通信信道类型分为两类：

（1）点—点线路通信子网的拓扑。

（2）广播信道通信子网的拓扑。

在点－点线路的通信子网中，每条物理线路连接一对结点，它有 4 中基本的拓扑构型：星型、环型，树型，网状型。

采用广播信道的通信子网中，一个公共的通信信道被多个网络结点共享，它也有 4 中基本拓扑构型：总线型，树型，环型，无线通讯和卫星通信型。

在星型结构中，结点通过点—点通信线路与中心结点相连，中心结点控制整个网络的通信，任何两个结点之间的通信都要通过中心结点。这种拓扑结构具有结构简单，容易实现，方便管理的特点，但是由于网络中心负责整个网络的可靠性通信，所以中心结点的故障将导致整个网络的瘫痪。

在环型结构中，结点通过点—点通信线路连接成闭合环路，环中的数据将沿着一个方向往逐个往下一站发送。这种结构比较简单，传输延时稳定，但是环中的每个结点与通信结点之间的线路都是整个环路中的瓶颈，所以环中的任何一个节点出现线路故障，都可能导致整个网络瘫痪。为了保证环路正常工作，需要复杂的控制和维护技术。

树型结构是星型结构的一个扩展。在树型结构中节点按层次进行连接，信息交换主要发生在上、下节点之间。

网状拓扑结构又称为无规则型。在这种结构中，结点之间的连接是任意的，可以没有任何规律。网状结构的特点是可靠性高，网内的任何一个结点的故障都不会影响到整个网络的正常运行。但是它的结构复杂，必须要采用合适的路由选择算法和介质访问控制方式及流量控制。

1.2.2 计算机网络的构成

计算机网络主要由多个计算机及通信设备构成，具体如下：

（1）各种类型的计算机。

（2）网络适配器。网络适配器提供通信网络与计算机相连的接口。

（3）网络传输介质，包括双绞线、光纤以及无线通信等。
（4）共享的外部设备。
（5）局部网络通信设备，如集线器、中继器。
（6）网络互联设备，如调制解调器、网桥、路由器、交换机。
（7）网络软件。

1.2.3 计算机网络与分布式系统

计算机网络与分布式系统（Distributed System）是两个不同的概念。

用户透明性观点定义计算机网络“存在一个能为用户自动管理计算机资源的网络操作系统，由它调用完成拥护任务所需要的资源，而整个网络对用户是透明的。”它所描述的就是一个分布式系统。

分布式系统有 5 个特征：

（1）系统拥有多种通用的物理和逻辑资源，可以动态的给他们分配任务。
（2）系统中分散的物理和逻辑资源通过计算机网络实现信息交换。
（3）系统中存在一个以全局方式管理系统资源的分布式操作系统。
（4）系统中联网的各计算机既合作又自治。
（5）系统内部结构对用户是完全透明的。

从以上的讨论可以看出分布式系统和计算机网络系统的共同点是：多数分布式系统是建立在计算机网络之上的，所以分布式系统与计算机网络在物理结构上是基本相同的。

他们的区别在于：分布式操作系统的设计思想和网络操作系统是不同的，这决定了它们在结构、工作方式和功能上也不同。网络操作系统要求网络用户在使用网络资源时首先必须了解网络资源，网络用户必须知道网络中各个计算机的功能与配置、软件资源、网络文件结构等情况，在网络中如果用户要读一个共享文件时，用户必须知道这个文件放在那一台计算机的那一个目录下；分布式操作系统是以全局方式管理系统资源的，它可以为用户任务调度网络资源，并且调度过程是“透明”的。当用户提交一个作业时，分布式操作系统能够根据需要在系统中选择最合适的处理器，将用户的作业提交到该处理程序，在处理器完成作业后，将结果传给用户。在这个过程中，用户并不会意识到有多个处理器的存在，这个系统是就像是一个处理器一样。

所以，分布式系统与计算机网络的最主要的区别不在于它们的物理结构，而是在系统软件上。分布式系统是一个建立在网络之上的软件系统，这种软件系统保证了系统高度的一致性和透明性，分布式系统不需要用户关心网络环境中的资源分布情况和各个联网计算机的性能差异，系统的作业管理和文件管理对用户是完全透明的。

1.2.4 计算机局域网

局域网（LAN）是在小范围内通过传输介质及一定的接口电路将许多数据设备互相连接起来进行数据通信、资源共享的计算机网络系统。数据设备可以是个人计算机、小型计算机、中型计算机或大型计算机，也可以是终端或外围设备（如打印机），但主要是个人计

算机。

将数据设备互连成局域网，可以实现软、硬件资源共享，如共享贵重的激光打印机、绘图仪、共享数据库及复杂的软件包等，也便于各种设备间及时交换数据，实现信息的实时采集、处理及报表输出，对办公自动化及管理决策提供必要的支持。

局域网的特点

局域网具有以下几个方面的特点：

（1）覆盖的地理范围不大，一般属于一个单位所有。

（2）通常采用专用的传输线路（也有用宽带、无线等传输方式），数据传输速率高。

（3）拓扑结构简单，容易实现。一般常采用总线型、星型或环型结构。

（4）与广域网相比，LAN 软件规范设计多限制在物理层、数据链路层和传输层。

（5）通信延迟时间较低，可靠性较好。

（6）能按广播方式或组播方式进行通信。

从原理上讲，决定局域网性能和费用的主要技术问题是网络拓扑结构、传输介质及通道访问控制策略。

与 OSI 参考模型相比，局域网只相当于 OSI 的最低两层物理层和数据链路层。物理层用于计算机的物理连接以及数据在媒体上的传输。局域网的数据链路层分为两个子层：媒体接入控制或媒体访问控制子层和逻辑链路控制子层。由于局域网不存在路由选择问题，因此局域网可以没有网络层。

局域网的分类

局域网的分类可以从拓扑结构、传输介质和通道访问控制方式三个方面进行。

按网络拓扑结构分类，局域网可分为星型、树型、环型、总线型和混合型几种。星型网属于集中控制方式，协议简单，易于监测和管理，扩充比较方便，一个工作站出现故障不会波及全网，但交换中心是全网的核心，其可靠性极为重要。

在环型拓扑结构中，通过点到点链路将接口处理机联接成一个闭合环路，每个接口处理机都与两条链路相联，而工作站则直接与接口处理机连接。环型网络是一种分布式控制方式，适用于实时传输要求较高的场合，性能受网络负载影响较小，由于采用遍绕环的传播方式，避免了复杂的中央控制和路由选择功能，协议相对简单。但对环上结点和链路的可靠性要求较高，如果一个结点出现故障，则会导致全网无法工作。一般须在网络结点接头处采用旁路措施将故障结点从网上隔离；而对链路故障，只有换新线或排除故障后才能使环型网恢复工作。

总线型拓扑结构是一种非常流行且广为采用的一种结构。在该结构中，每个工作站均通过一个接口电路挂在一条总线传输介质上，共享一条公共的总线进行数据传输。总线型局域网适于实时要求不高、通信距离短、负载较轻的场合。它扩充方便，可靠性较高，属分散控制。但其网络性能受负载变化影响很大，重负载时总线上报文碰撞的机会增大，网络性能降低。

局域网中的传输介质种类很多，一般有双绞线、同轴电缆、光纤等有线介质，还有诸如电磁波、微波、卫星通信、无线通信、激光等空间介质。如果从使用频带看，又可分为基带和宽带两种传输介质。基带传输时，通道中为数字信号，整个通道的频谱均为一路数

字信号所占用。宽带传输通道中的信号为模拟信号，通道可以采用频率划分方式进行通道的多路复用，多采用 75W 的同轴电缆作为传输介质，容易实现多种用途的通道复用，例如图像或语音传输，也可用于数字传输。

通道访问技术是局域网的关键技术之一。所谓通道访问技术，就是研究如何有效地利用通信信道问题，目的是使网络如何适应数据传输的突发性特点。采用恰当的策略，可以提高通信信道的利用率，减少数据传输中的时延，提高网络的吞吐率。网络的性能在很大程度上都与网络所采用的通道访问技术有关。通道访问控制也叫通道访问协议。

以太网

以太网（Ethernet）是常用的局域网之一，它是一种公共总线型局域网，适用于小型机和微型机连网。在以太网中，所有的节点通过专门的网卡（网络适配器）连接到一条总线上，并采用广播方式进行通信而无需路由功能。

以太网在 1972 年由 Xerox（施乐）公司首先开始研制，1975 年 Xerox 公司和 Stanford（斯坦福）大学合作推出了 Ethernet 产品。IEEE 802.3 国际标准是在 Ethernet 标准的基础上制定的。现在，根据不同的物理介质又发展出多种子标准，形成了一个 IEEE 802.3 标准系列。

Ethernet 中的计算机相互通信时采用的是载波侦听多路访问/冲突检测的方法，即 CSMA/CD（Carrier Sense Multiple Access/Collision Detection），它在轻负载情况下具有较高的网络传输效率。Ethernet 的组网非常灵活和简便，既可以使用细缆和粗缆组成总线型局域网，也可以使用 3 类无屏蔽双绞线组成星型网络（即 10BASE-T 技术），还可以将同轴电缆的总线型网络与 UTP 双绞线的星型网络混合起来。Ethernet 是目前国内外应用最为广泛的 10Mbps 局域网。

以太网大多采用集线器和双绞线进行组网。集线器（Hub）有共享式和交换式之分。共享式是一种无源耦合器，具有多个端口，支持节点通过双绞线接入 Hub，虽然物理上看上去是多级星形结构，但实际的媒体访问控制方式仍然是总线方式，联结在总线上的所有设备共享该总线的带宽。交换式 Hub 则不同，联结在总线上的每一个设备，各自独享一定的带宽。

FDDI 网络

FDDI 网络又称为光纤分布式数字接口网。FDDI 网主要用于主机与其外围设备之间、主机之间、各宽带工作站间的互连，又可作为主干网与 IEEE 802 局域网互连。此外 FDDI-II 作为 FDDI 的改进型，除了处理常规数据外，还具有电路交换能力，可以传输声音、图像等综合数据。由于采用光纤作为传输介质，FDDI 具有高数据传输率、抗电磁干扰和信号传输衰减的特点。另一方面，FDDI 也具有环形网重负载时效率高、覆盖面大、具有实时和优先访问等优点。FDDI 对同步和异步数据传输都适用，能处理数字电话和图像的实时业务，主要采用分组交换，也允许用电路交换。此外 FDDI 还具有重构特性，可靠性较高。

1.2.5 计算机病毒及其特征

计算机病毒的定义

计算机病毒是一种特殊的具有破坏性的计算机程序，它具有自我复制能力，可以通过非授权入侵而隐藏在可执行程序或数据文件中，当计算机运行时病毒能够把自身精确拷贝或者有修改的拷贝到其他程序体中，影响和破坏正常程序的正确性。

计算机病毒的特征

计算机病毒是一类特殊的程序，它与其他程序一样可以存储和执行，但它具有其他程序没有的特性。

（1）传染性。计算机病毒的传染性是指病毒具有把自身复制到其他程序中的特性。

（2）破坏性。破坏性包括占用计算机 CPU 时间；占用内存空间；破坏数据和文件；干扰系统的正常运行。

（3）潜伏性。计算机病毒的潜伏性是指计算机病毒具有依附其他程序而存在的能力。

（4）隐蔽性。计算机病毒的隐藏性表现在两个方面。一是传染的隐蔽性，大多数病毒在进行传染的时候速度很快，一般没有外部表现，不容易被发现。二是病毒存在的隐蔽性，病毒大多潜伏在正常的程序之中，在其发作或产生破坏作用之前，一般不容易被发现。

（5）可激发性。在一定条件下，通过外界刺激可使病毒活跃起来，激发是一种条件控制，由病毒的设定。

1.2.6 计算机病毒的危害、分类和防治

计算机病毒的破坏作用

病毒造成的危害是严重和多方面的，主要表现在以下几个方面：

（1）破坏磁盘文件分配表，使用户在磁盘上的文件无法使用。

（2）删除磁盘上的可执行文件或数据文件。

（3）修改或破坏文件中的数据。

（4）将非法的数据写入内存参数区，造成死机甚至引起系统崩溃。

（5）改变磁盘分配表，造成数据写入错误。

（6）在磁盘上产生坏的扇区，使磁盘空间减少。

（7）更改或重写磁盘卷标。

（8）因病毒程序自身在系统中的多次复制而使内存可用空间减少。

（9）对整个磁盘或磁盘的特定磁道或扇区进行格式化。

（10）在系统中产生新的信息。

（11）改变系统的正常运行过程。

计算机病毒的结构

计算机病毒程序主要包括三个部分：一是传染部分（传染模块）；二是表现和破坏部分；三是触发部分（触发模块）。

计算机病毒的分类

（1）按传染的方式可分为：引导型病毒、一般应用程序型、系统程序型。

（2）按寄生的方式可分为：操作系统型病毒、外壳型病毒、入侵型病毒、源码型病毒。

计算机病毒的预防

计算机病毒以预防为主，防止病毒的入侵要比入侵后再去发现和排除要好的多。主要的预防措施有：

（1）采用抗病毒的硬件。

（2）计算机硬件安全措施。

（3）社会措施。

第 2 章　数据结构与算法

考纲要求

1．数据结构、算法的基本概念
2．线性表的定义、存储和运算
3．树形结构的定义、存储和运算
4．排序的基本概念和排序算法
5．检索的基本概念和检索算法

2.1　知识点

2.1.1　数据结构的基本概念

数据是描述客观事物的数字、字符以及所有能直接输入到计算机中并被计算机程序处理的符号的集合。

数据对象是具有相同性质的数据元素的集合。通常，一个数据对象中的数据元素不是孤立的，而是彼此之间存在着一定的联系，这种联系就是数据结构。数据对象中数据元素之间的联系需要在对数据进行存储和加工中反映出来，因此，数据结构概念一般包括三方面的内容：数据之间的逻辑关系、数据在计算机中的存储方式、以及在这些数据上定义的运算的集合。

数据的逻辑结构

数据的逻辑结构只抽象地反映数据元素之间的逻辑关系，它与数据的存储无关，是独立于计算机的。

数据的存储结构

数据的存储结构是数据的逻辑结构在计算机存储器里的实现（亦称为映象）。它是依赖于计算机的，并有四种基本的存储映象方法，分别为顺序存储方法、链接存储方法、索引存储方法、散列存储方法。这四种基本的存储方法也可以组合起来对数据结构进行存储映象。

数据的运算

数据的运算定义在数据的逻辑结构之上，每种逻辑结构都有一个运算的集合。常用的运算有：查找、插入、删除、更新、排序等。显然，对数据运算的具体实现方法只有在确定了存储结构之后才能加以考虑。

2.1.2 算法的基本概念

算法及其特征

简单地说，一个算法就是一种解题方法，更严格地说，算法是由若干条指令组成的有穷序列，它必须具有以下特征：

（1）有穷性：一个算法必须在执行有穷步后结束。

（2）确定性：算法的每一步必须是确切地定义的，无二义性。

（3）可行性：算法中的所有待实现的运算必须在原则上能够由人使用笔和纸在做有穷次运算后完成。

（4）输入：一个算法具有 0 个或多个输入的外界量，它们是算法开始前对算法最初给出的量。

（5）输出：一个算法至少产生一个输出，它们是与输入有某种关系的量。

对一个算法的描述可以采用自然语言、数学语言、约定的符号语言、以及图解等方式。

算法与程序的区别

算法的含义与程序十分相似，但二者又有区别。一个程序不一定满足有穷性，操作系统就是如此，只要整个系统不被破坏、操作系统就永远不会停止，所以操作系统程序不是一个算法。另外，程序中的指令必须是机器可以执行的，而算法中的指令则无此限制。但是，一个算法如果用机器可执行的语言书写，则它就是一个程序。

算法的分析

求解同一个问题可以有多种不同的算法，评价一个算法的优劣除了正确性和简明性外，主要考虑两点：一是执行算法所耗费的时间，二是执行算法所耗费的存储空间，特别是辅助存储空间的耗费。就这两者而言，前者显得比后者更为重要，在数据结构中往往更注重对算法执行时间的分析。一个算法所耗费的时间是该算法中每条语句的执行时间之和，而每条语句的执行时间是该语句执行次数（频度）与该语句一次执行所需时间的乘积。

2.1.3 线性表

线性表是 $n \geq 0$ 个元素的一个有限序列：（a_1，a_2，a_3，....a_{n-1}，a_n）

线性表的一个重要特性是可以按照诸元素在表中的位置确定它们在表中的先后次序。若 $n \geq 0$，则 a_1 为第一个元素，a_n 为最后一个元素。元素 a_{i-1} 先于 a_i，我们称 a_{i-1} 为 a_i 的前驱；a_i 在 a_{i-1} 之后，a_i 为 a_{i-1} 的后继。除第一个元素外，每个元素都有一个且仅有一个直接前驱；除最后一个元素外，每个元素都有一个且仅有一个直接后继。

用顺序存储结构存储的线性表称做顺序表；用链式存储结构存储的线性表称做链表；用散列方法存储的线性表称做散列表。如果对线性表插入和删除运算发生的位置加以限制，则成为两种特殊的线性表——栈和队列。若线性表中的元素都是单个字符，则称做串。

2.1.4 线性表的存储

有多种存储方式能将线性表存储在计算机内，其中最常用的是顺序存储和链接存储。

线性表的顺序存储

线性表的顺序存储是最简单的存储方式。程序通常用一个足够大的数组，从数组的第一个元素开始，将线性表的结点依次存储在数组中。即线性表的第 i 个结点存储在数组的第 i（0≤i≤n-1）个元素中，用数组元素的顺序存储来体现线性表中结点的先后次序关系。用数组存储线性表的最大优点是能直接访问线性表中的任一结点。

用数组存储线性表的缺点主要有两个：一是程序中的数组通常大小是固定的，可能会与线性表的结点可以任意增加和减少的要求相矛盾；二是执行线性表的结点插入、删除操作时要移动存于数组中的其他元素，使插入和删除操作不够简便。

线性表的链接存储

线性表链接存储是用链表存储线性表，最简单的用单链表。如从链表的第一个表元开始，将线性表的结点依次存储在链表的各表元中。即线性表的第 i 个结点存储在链表的第 i（0＜i＜n-1）个表元中。链表的每个表元除要存储线性结点的信息外，还要有一个成分用来存储其后继结点的指针。单链表就是通过链接指针来体现线性表中结点的先后次序关系。每个链表还要有一个指向链表的第一个表元，链表的最末一个表元的后继指针值为空。用链表存储线性表的优点是线性表的每个表元的后继指针就能完成插入或删除的操作，不需移动任何表元。

其缺点也主要有两条：一是每个表元增加了一个后继指针成分，要花费更多的存储空间；二是不便随机地直接访问线性表的任一结点。

2.1.5 线性表的运算

查找运算

查找线性表的第 i（0≤i≤n-l）个表元；

在线性表中查找具有给定键值的表元；

插入运算

把新表元插在线性表的第 i（0≤i≤n）个位置上；

把新表元插在具有给定键值的表元的前面或后面；

删除运算

删除线性表的第 i（0≤i≤n-1）个表元；

删除线性表中具有给定键值的表元；

其他运算

统计线性表元的个数；

输出线性表各表元的值；

复制线性表；
线性表分析；
线性表合并；
线性表排序；
按某种规则整理线性表。

2.1.6 数组

数组是有限个同类型数据元素组成的序列。一个二维数组（或称矩阵）可以看成是由 m 个行向量所组成的向量，也可以看成是由 n 个列向量所组成的向量。数组的运算主要有查找或存取数组中某个元素。

由于计算机存储单元是一维的结构，而数组是多维的结构，因此就必须把多维结构映射为一维的结构，即把多维结构按一定次序排列成一维的，常用的排列次序有行优先顺序和列优先顺序两种。

给出任一数组元素的下标，可以直接计算数组元素的存放地址。二维数组元素地址的计算公式是：

（l）行优先顺序下：

$$\mathbf{LOC}(a_{ij}) = \mathbf{LOC}(a_{11}) + ((i-1)\times n + (j-1)) \times \lambda$$

（2）列优先顺序下：

$$\mathbf{LOC}(a_{ij}) = \mathbf{LOC}(a_{11}) + ((j-1)\times m + (i-1)) \times \lambda$$

式中，n 和 m 分别为数组每行和每列的元素个数，λ为每个数组元素占用的存储单元个数。

2.1.7 稀疏矩阵

具有大量 0 元素的矩阵称做稀疏矩阵。对于稀疏矩阵可以进行压缩存储，只存储非 0 元素。若非 0 元素的分布有规律，则可以用顺序方法存储非 0 元素，仍可以用公式计算数组元素的地址。一般的稀疏矩阵主要有两种存储方法。三元组法和十字链表法。

2.1.8 广义表

广义表（又称列表）是线性表的推广，是由零个或多个单元素或子表所组成的有限序列。广义表和线性表的区别在于：线性表的成分都是结构上不可分的单元素，而广义表的成分既可以是单元素，又可以是有结构的表。

广义表一般记做：

$$\mathbf{LS} = (d_1, d_2, \cdots, d_n)$$

其中 LS 是广义表的名字，n 是长度，d 是广义表成分。

广义表有如下特征：

（1）广义表的元素可以是子表，而子表的元素还可以是子表。

（2）广义表可被其他广义表所共享（引用）。

（3）广义表可以是递归的表，即广义表也可以是本身的一个子表。

和线性表类似，可对广义表进行的运算有查找、插入和删除等。但广义表的两个非常重要的基本运算是取广义表表头 HEAD（LS）和取广义表表尾 TAIL（LS）。

2.1.9 树

树的基本概念

树是 n（n＞0）个结点的有穷集合，有且仅有一个称为根的结点；其余结点分为 m（m≥0）个互不相交的非空集合，T_1，T_2，…，T_m，这些集合中的每一个都是一棵树，称为根的子树。

在树上，根结点没有直接前趋。对树上任一结点 X 来说，X 是它的任一子树的根结点唯一的直接前趋。树上任一结点所拥有的子树的数目称为该结点的度。度为 0 的结点称为叶子或终端结点。度大于 0 的结点称为非终端结点或分支点。一棵树中所有结点的度的最大值称为该树的度。若树中结点 A 是结点 B 的直接前趋，则称 A 为 B 的双亲或父结点，称 B 为 A 的孩子（即“子女”）或子结点。任何结点 A 的孩子 B 也就是 A 的一棵子树的根结点，父结点相同的结点互称为兄弟。一棵树上的任何结点（不包括根本身）称为根的子孙。反之若 B 是 A 的子孙，则称 A 是 B 的祖先，结点的层数（或深度）从根开始算起：根的层数为 1，其余结点的层数为其双亲的层数加 1。一棵树中所有结点层数的最大值称为该树的高度或深度。

树的基本运算

（1）求根 ROOT(T)，引用型运算，其返回值是结点 X 在树 T 的根结点。

（2）求双亲 PARENT(T,X)，引用型运算，其结果是结点 X 在树 T 上的；若 X 是树 T 的根或 X 不在 T 上，则返回值为一特殊标志。

（2）求孩子 CHILD(T,X,i) ，引用型运算，其结果是树 T 上的结点 X 的第 i 个孩子；若 X 不在 T 上或 X 没有第 i 个孩子，则返回值为一特殊标志。

（3）建树 CREATE(X,T_1,…, T_k) k≥1，加工型运算，其作用是建立一棵以 X 为根，以 T_1，…，T_k 为第 1，…k 棵子树的树。

（4）剪枝 DELETE(T,X,i)，加工型运算，其作用是删除树 T 上结点 X 的第 i 棵子树；若 T 无第 i 棵子树，则为空操作。

2.1.10 二叉树

二叉树的基本概念

二叉树是 n≥0 个结点的有穷集合，它或者是空集，或者同时满足下述两个条件：

有且仅有一个称为根的结点；其余结点分为两个互不相交的集合 T_1、T_2，T_1 与 T_2 都是二叉树，并且 T_1 与 T_2 有顺序关系（T_1 在 T_2 之前），它们分别称为根的左子树和右子树。

二叉树是一类与树不同的树形结构。它们的区别是：第一，二叉树可以是空集，这种

二叉树称为空二叉树。第二，二叉树的任一结点都有两棵子树（当然，它们中的任何一个可以是空子树），并且这两棵子树之间有次序关系，也就是说，它们的位置不能交换。相应地，二叉树上任一结点左、右子树的根分别称为该结点的左孩子和右孩子。另外，二叉树上任一结点的度定义为该结点的孩子数（即非空子树数）。除这几个术语之外，树的其它术语也适用于二叉树。

特别值得注意的是，由于二叉树上任一结点的子树有左、右之分，因此即使一结点只有一棵非空子树，仍须区别它是该结点的左子树还是右子树，这是与树不同的。

二叉树的基本运算

（1）初始化 INITIATE(BT)，加工型运算，其作用是设置一棵空二叉树 BT=Φ

（2）求根 ROOT(BT)，引用型运算，其结果是二叉树 BT 的根结点；若 BT 为空二叉树，运算结果为一特殊标志。

（3）求双亲 PARENT(BT,X)，引用型运算，其结果是结点 X 在二叉树 BT 上的双亲；若 X 是 BT 的根或 X 根本不是 BT 上的结点，运算结果为一特殊标志。

（4）求左孩子 LCHILD(BT,X)和求右孩子 RCHILD(BT,X)，引用型运算，其结果分别为结点 X 在二叉树 BT 上的左、右孩子；若 X 为 BT 的叶子或 X 不在 BT 上，运算结果为一特殊标志。

（5）建树 CREAT(X,LBT,RBT)，加工型运算，其作用是建立一棵以结点 X 为根，以二叉树 LBT、RBT 分别为 X 的左、右子树的二叉树。

（6）剪枝 DELLEFT(BT,X)和 DELRIGHT(BT,X)，加工型运算，其作用分别为删除二叉树 BT 上结点 X 的左、右子树；若 X 无左或右子树，运算为空操作。

与其它数据结构相同，在实际应用中根据需要对基本运算适当增减。

2.1.11　二叉树的存储结构

二叉树通常有两类存储结构，顺序存储结构和链式存储结构。

二叉树的顺序存储结构

二叉树的顺序存储结构由一个一维数组构成，二叉树上的结点按某种次序分别存入该数组的各个单元。显然，这里的关键在于结点的存储次序，这种次序应能反映结点之间的逻辑关系（父子关系），否则二叉树的基本运算就难以实现。

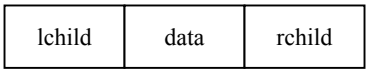
由二叉树的性质可知，若对任一完全二叉树上的所有结点按层编号，则结点编号之间的数值关系可以准确地反映结点之间的逻辑关系。因此，对于任何完全二叉树来说，可以采用“以编号为地址”的策略将结点存入作为顺序存储结构的一维数组。具体地说就是：将编号为 i 的结点存入一维数组的第 i 个单元。

在这一存储结构中，由于一结点的存储位置（即下标）也就是它的编号，故结点间的逻辑关系可通过它们下标间的数值关系确定。

二叉树的链式存储结构

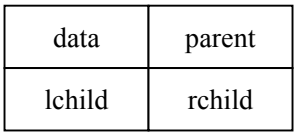
二叉树有不同的链式存储结构，其中最常用的是二叉树链表与三叉链表。

二叉树链表的结点形式如下：



其中，data 域称为数据域，用于存储二叉树结点中的数据元素；lchild 域称为左孩子指针域，用于存放指向本结点左孩子的指针（这个指针及指针域有时简称为左指针）。类似地，rchild 域称为右孩子指针域，用于存放指向本结点右孩子的指针（简称右指针）。若某结点的某个孩子不存在，则相应的指针为空。具有 n 个结点的二叉树中，一共有 2n 个指针域，其中只有 n—1 个用来指向结点的左右孩子，其余的 n+1 个指针域为 NULL。

三叉链表的每个节点由四个域组成，其结点形式如下：



其中，data、lchild 以及 rchild 三个域的意义同二叉链结构；parent 域为指向该结点双亲结点的指针。

2.1.12 二叉树和树的遍历

遍历是树形结构的一种重要运算。遍历一个树形结构就是按一定的次序系统地访问该结构中的所有结点，使每个结点恰被访问一次。可以按多种不同的次序遍历树形结构。

2.1.13 排序

排序是数据处理中经常使用的一种重要运算。

设有{R_1，R_2，…，R_n}是由 n 个记录组成的文件，其相应的关键码集合为{K_1，K_2，…，K_n}。所谓排序就是将记录按关键码值递增（或递减）的次序排列起来。

由于文件大小不同使排序过程中涉及的存储器不同，可将排序分为内部排序和外部排序两类。整个排序过程都在内存进行的排序，称为内排序，这是排序的基础。

内排序的方法很多，最常用的有插入排序、选择排序、交换排序和归并排序等。

2.1.14 查找

查找就是在数据结构中找出满足某种条件的结点。若从数据结构中找到满足条件的结点，则称查找成功，否则查找失败。衡量一个查找算法的主要标准，是查找过程中对关键码进行的平均比较次数，或称平均检索长度，检索长度以 n 的函数的形式表示，n 是数据结构中的结点个数。常用的查找算法有顺序查找、二分法查找、分块查找、散列表查找等。

2.2 重点难点

2.2.1 数据的逻辑结构

数据的逻辑结构分为线性结构和非线性结构两大类，线性结构的逻辑特征是：有且仅有一个开始结点和一个终端结点，并且所有的结点都最多有一个直接前驱和一个直接后继。线性表就是一个典型的线性结构，常见的线性表有顺序表、链表、栈、队列、串。非线性结构的逻辑特征是：一个结点可能有多个直接前驱和直接后继。树、二叉树、图等都是非线性结构。

栈

栈是一种特殊的线性表，这种线性表只能在固定的一端进行插入和删除操作。允许插入和删除的一端称为栈顶，另一端称为栈底。一个新元素只能从栈顶一端进入，删除时，只能删除栈顶的元素，即刚刚被插入的元素。由于元素是按后进先出的次序入栈和出栈的，所以栈又称后进先出表（Last In First Out），简称 LIFO 表。

栈的基本操作有：

（1）create(s)建立一个空栈 s。

（2）empty(s)测试栈是否为空栈。

（3）full(s)测试栈是否满。

（4）push(x,s)将元素 x 插入栈 s 的栈顶。

（5）top(s)取栈顶元素。

（6）pop(s)删除栈顶元素。

由于栈是一种特殊的线性表，栈的各种操作实际上是线性表的操作的特殊情形，所以表示线性表的方法同样可以用来表示栈。

队列

队列可看作是插入在一端进行，删除在另一端进行的线性表，允许插入的一端称为队尾，允许删除的一端称为队头。在队列中，只能删除队头元素。队列的最后一个元素一定是最新入队的元素。因此队列又称先进先出表（First－In－First－Out）。

日常生活中排队购物就是队列应用的例子：新来的顾客排在队尾等待，排在队头的顾客购物后离开队伍。

队列的基本操作有：

create（Q）建立一个空队列。

empty（Q）测试队列是否为空队列。

full（Q）测试队列是否为满。

front(Q)取队头元素。

enq（X，Q）向队列中插入一个元素 X。

deq（Q）删除队头元素。

串

串（或字符串）是由零个或多个字符组成的有限序列，一般记为 S＝’a_1 a_2…a_n’。其中，S 是串的名字，用单引号括起来的若干字符是串的值。零个字符的串是空串。串中字符的数目就是串的长度。n 是串中的字符，可以是字母、数字或其他字符。空串与空格构成的串如：“ ”是不同的。

串的存储同样有顺序存储和链式存储两种。顺序存储时，既可以采用非紧缩方式，也可以采用紧缩方式。

串的基本运算有连接、赋值、求长度、全等比较、求子串、找子串位置以及替换等。有些程序设计语言以标准子程序（函数或过程）方式提供了这些运算。

2.2.2 数据存储结构

顺序存储结构

这种存储方式主要用于线性的数据结构，它把逻辑上相邻的数据元素存储在物理上相邻的存储单元里，结点之间的关系由存储单元的邻接关系来体现。

顺序结构的主要特点是：

（1）结点中只有自身信息域，没有链接信息域。因此，存储密度大，存储空间利用率高。

（2）可以通过计算直接确定数据结构中第 i 个结点的存储地址 L_i，计算公式是：

$$L_i＝L_0＋（i－1）×m$$

其中 L_0 为第一个结点的存储地址，m 为每个结点所占用的存储单元个数。

（3）插入、删除运算会引起大量结点的移动。

链式存储结构

链式存储结构就是在每个结点中至少包括一个指针域，用指针来体现数据元素之间逻辑的联系。这种存储结构，可把人们从计算机存储单元的相继性限制中解放出来，可以把逻辑相邻的两个元素存放在物理上不相邻的存储单元中，还可以在线性编址的计算机存储器中表示结点之间的非线性联系。

链式存储结构的主要特点是：

（1）结点中除自身信息外，还有表示链接信息的指针域，因此比顺序存储结构的存储密小，存储空间利用率低。

（2）逻辑上相邻的结点物理上不必邻接，可用于线性表、树、图等多种逻辑结构的存储表示。

（3）插入、删除操作灵活方便，不必移动结点，只要改变结点中的指针值即可。

散列法

散列表（又称哈希表）是线性表的一种重要存储方式和检索方法。在散列表里可对结点进行快速检索。

散列法的基本思想是：由结点的关键码值决定结点的存储地址，即以关键码值 k 为自

变量，通过一定的函数关系 h（称为散列函数），计算出对应的函数值 h（k）来，把这个值解释为结点的存储地址，将结点存入该地址中去。检索时再根据要检索的关键码值用同样的散列函数计算地址，然后到相应的地址中去取要找的结点。

常用的散列函数有：

（1）除余法

选择一个适当的正整数 p（通常选 p 为不大于散列表存储区域大小的最大素数），用 p 去除关键码值，取其余数作为地址。即 h（key）＝key mod p。例如，如果散列表存储区域大小为 1024，则可以选 p＝1019，它将关键码值 5876 对应到地址 781，将关键码值 1234 对应到地址 215，…除余法地址计算公式非常简单，实践证明恰恰是这种简单的方法在许多情况下效果较好。

（2）数字分析法

当关键码的位数比存储区域地址码位数多时，可以对关键码的各位进行分析，去掉分布不均匀的位，留下分布均匀的位作为地址。

（3）折叠法将关键码值从某些地方断开，分为几段，折叠相加，作为地址。

（4）中平方法将关键码值平方，取中间的几位数作为地址。

在散列法中，不同的关键码值可能对应到同一存储地址，即 $k_1 \neq k_2$，但 $h(k_1)=h(k_2)$ 这种现象称做碰撞（冲突）。发生碰撞的两个或多个关键码值称做同义词。需要有办法处理碰撞，即为对应到同一地址的多个同义词安排存储地址。

常用的处理碰撞的方法有两类：拉链法和开地址法。

用拉链法处理碰撞就是给散列表的每个结点增加一个 link 字段，当碰撞发生时利用 link 字段拉链，建立链接方式的同义词子表。每个同义词子表的第一个元素都在散列表基本区域中。同义词子表的其他元素存储在何处，通常采用建立溢出区的方法，即另开辟一片存储空间作为溢出区，用于存放各同义词子表的其他元素。

用开地址法处理碰撞就是当碰撞发生时形成一个探查序列，沿着这个序列逐个地址探查，直至找到一个开放的地址（即未被占用的单元），将发生碰撞的关键码值存人该地址中。最简单的探查序列是线性探查，即若发生碰撞的地址为 d，则探查的地址序列是：

d＋1，d＋2，…，m-1，0，1，…，d-1

其中，m 是散列表存储区域的大小。

散列表的一个重要参数是负载因子 α，它的定义是：

$$\alpha = \frac{\text{散列表中结点的数目}}{\text{基本区域能容纳的结点数}}$$

负载因子的大小体现散列表的装满程度，α 越大，则散列表装得越满，发生碰撞的可能性越大，一般取 a＜1。

2.2.3 二叉树的性质

二叉树的性质主要有五个：

性质 1

二叉树第 i(i≥1)层上至多有 2^{i-1} 个结点。

二叉树是结点的有限集合，这个有限集合或者为空集，或者由一个根结点及两棵不相交的、分别称做这个根的左子树和右子树的二叉树组成。这是一个递归的定义，二叉树的根的左子树和右子树又都是二叉树，它们又都是结点的集合，或者为空集，或者由根结点和左子树、右子树构成，左子树和右子树又都是二叉树，如此递归下去。

可以用归纳法证明这个性质：

当 i＝1 时，二叉树只有一个结点，即二叉树的根结点。显然 $2^{i-1}=2^0=1$。假设对所有 j（$1\leq j\leq i-1$）性质一成立，即第 j 层上最多有 2^{i-1} 个结点。证明当 j＝i 时性质 1 也成立即可。

根据归纳假设，第 i-1 层上最多有 2^{i-2} 个结点；又由于每个结点最多有 2 棵子树。因此，第 i 层的结点数目最多是第 i-1 层上最大结点数目的 2 倍，即有 $2\times 2^{i-2}=2^{i-1}$。命题得证。

性质 2

深度为 k(k≥1)的二叉树至多有 2^k-1 个结点。

显然，深度为 k 的二叉树只有是满二叉树时才能达到最大结点数，即每层的结点数目都达到该层的最大结点数 2^{i-1}，这时二叉树的结点数目是：

$$\sum_{i=1}^{k}(\text{第}i\text{层上结点的最大数目})=\sum_{i=1}^{k}2^{i-1}=2^k-1$$

性质 3

对任何二叉树，若度为 2 的结点数为 n_2，则叶子数 $n_0=n_2+1$。

设度为 1 的结点数目为 n_1，则二叉树的结点总数 n 是：

$$n=n_0+n_1+n_2$$

由于二叉树中除根结点以外，每个结点都有一个分支指向它，因此二叉树中总的分支数 B 为：

$$B=n-1$$

而所有这些分支不是由度为 1 的结点发出的就是由度为 2 的结点发出的，即每个度为 1 的结点发出一个分支，每个度为 2 的结点发出两个分支。于是应有：

$$B=n_1+n_2$$

三式联立可得到：

$$n_0=n_2+1$$

命题得证。

性质 4

具有 n 个结点的完全二叉树的深度为［$\log_2 n$］＋1。

深度为 k（k≥1）且有 2^k-1 个结点的二叉树称为满二叉树。由性质 2 知，满二叉树上各层的结点数已达到了二叉树可以容纳的最大值。如果在一棵深度为 k（k≥1）的满二叉树上删去第 k 层上最右边的连续 j($0\leq j<2k-1$＝个结点，就得到一棵深度为 k 的完全二叉树，即除了最下面一层外，各层都被结点充满了。显然，满二叉树也是完全二叉树，但反之不

然。

根据完全二叉树以及二叉树的性质二，有

$$2^{k-1}-1<n\leq 2^{k}-1$$

或 $2^{k-1}\leq n<2^{k}$

于是有

$$k-1\leq <k$$

由于 k 为正整数，因此

$$k=\log_2 n+1$$

性质 4 成立。

性质 5

如果将一棵有 n 个结点的完全二叉树按层编号，则对任一编号为 i（1≤i≤n）的结点 X 有：

（1）若 i=1，则结点 X 是根；若 i＞1，则 X 的双亲 PARENT(X)的编号为 ［i/2］。

（2）若 2i＞n，则结点 X 无左孩子（且无右孩子）；否则，X 的孩子 LCHILD(X)的编号为 2i。

（3）若 2i＋1＞n，则结点 X 无右孩子，否则，X 的右孩子 RCHILD(X)的编号为 2i+l。

2.2.4 二叉树和树的遍历

二叉树的遍历

遍历一棵二叉树就是按某种次序系统地“ 访问 ”二叉树上的所有结点，使每个结点恰好被“ 访问 ”一次。所谓“ 访问 ”一个结点，是指对该结点的数据域进行某种处理，处理的内容依具体问题而定，通常比较简单。遍历运算的关键在于访问结点的“ 次序 ”，这种次序应保证二叉树上的每个结点均被访问一次且仅一次。

由定义可知，一棵二叉树由三部分组成：根、左子树和右子树。因此对二叉树的遍历也可相应地分解成三项“ 子任务 ”：

（1）访问根；

（2）遍历左子树（即依次访问左子树上的全部结点）；

（3）遍历右子树（即依次访问右子树上的全部结点）。

因为左、右子树都是二叉树（可以是空二叉树），对它们的遍历可以按上述方法继续分解，直到每棵子树均为空二叉树为止。由此可见，上述三项子任务之间的次序决定了遍历的次序。若以 D、L、R 分别表示这三项子任务，则有六种可能的次序：DLR、LDR、LRD、DRL、RDL 和 RLD。通常限定“ 先左后右 ”，即子任务 2 在子任务 3 之前完成，这样就只剩下前三种次序，按这三种次序进行的遍历分别称为先根遍历（或前序遍历）。中根（或中序）遍历、后根（或后序）遍历。三种遍历方法的定义如下

先根遍历

若需遍历的二叉树为空，执行空操作；否则，依次执行下列操作：

（1）访问根结点；

（2）先根遍历左子树；
（3）先根遍历右子树。

中根遍历

若需遍历的二叉树为空，执行空操作，否则，依次执行下列操作：
（1）中根遍历左子树；
（2）访问根结点；
（3）中根遍历右子树。

后根遍历

若需遍历的二叉树为空，执行空操作，否则，依次执行下列操作；
（1）后根遍历左子树。
（2）后根遍历右子树。
（3）访问根结点。

上述三种遍历方法的区别在于执行子任务“访问根结点”的“时机”不同；最先（最后、在中间）执行此子任务，则为先根（后根、中根）遍历。

按某种遍历方法遍历一棵二叉树，将得到该二叉树上所有结点的不同访问系列。

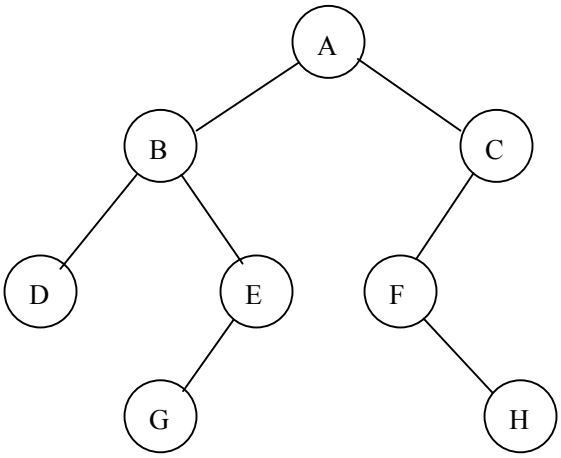


图 2-1　二叉树

对图 3－1 所示的二叉树，按先根遍历方式得到的结点序列是 ABDEGCFH，按中根遍历方式得到的结点序列是 DBGEARFHC，按后根遍历得到的结点序列是 DGEBHFCA。

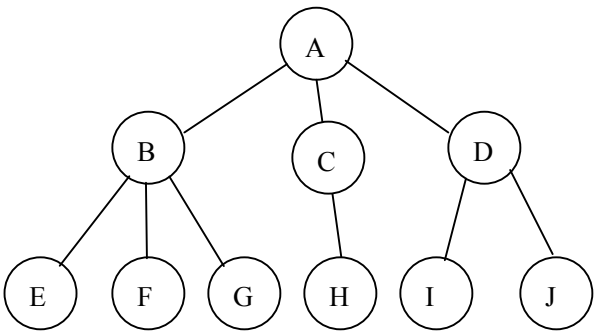


图 2-2　树

遍历树

与二叉树类似，树的主要遍历方法有以下三种。

（1）先根遍历

若树非空，则先访问根结点，然后依次先根遍历根的各个子树 T_1，…，T_m。

（2）后根遍历

若树非空，则依次先根遍历根的各个子树 T_1，…，T_m，再访问根结点；

（3）层次遍历

若树非空，则先访问根结点，若第 1，…，i（i＞1）层结点已被访问，且第 i+1 层结点未访问，则从左到右依次访问第 i＋1 层结点。按层次遍历所得的结点访问序列中，各结点的序号与按层编号所得的编号一致。

对图 3－2 所示的树，按先根遍历方式得到的结点序列是 ABEFGCHDIJ，按后根遍历方式得到的结点序列是 EFGBHCIJDA，按层次遍历得到的结点序列是 ABCDEFGHIJ。

2.2.5　二叉树的转换

树与二叉树之间的转换

一般树转换为二叉树的基本思想是：将树中每个结点用两个链接表示就可以了，一个指向它最左边的孩子，另一个指向它右边的一个兄弟，从图形上看，具体步骤是：

（1）加线：在兄弟结点直接加一虚线。

（2）抹线：对每个结点，除了其最左的一个孩子外，抹去该结点原来与其余孩子之间的边线。

（3）旋转：将新加上的虚线改为实线，并将虚线以及有关的实线顺时钟旋转 45 度。

二叉树还原为一般树的步骤是：

（1）加线：若某结点是一父结点的左孩子，则将该结点的右孩子以及沿着右链搜索到的所有右孩子结点都用线与那个父结点连接起来；

（2）抹线：抹去原二叉树中所有结点与其右孩子的连线；

（3）旋转：将虚线及有关实线道时钟旋转约 45 度，并将几个结点按层次排列。

2.2.6　二叉排序树

二叉排序树定义

二叉排序树实际上是将线性表中的结点信息（或结点中的关键码值和结点地址）组织成二叉树形式，以达到与二分法查找相同的查找效率，而又具有链表那样的插入、删除运算的灵活性。

二叉排序树的特点是：每个结点的左子树中所有的结点的关键码值都小于该结点的关键码值，而右子树中所有结点的关键码值都大于该结点的关键码值。

二叉排序树的查找方法非常简单，将待查关键码值与树根的关键码值比较，若相等则

查到，否则根据比较结果确定进入左子树或右子树，继续查找，如此进行下去，直至找到待查的关键码值，或确定二叉排序树中没有这样的关键码值。

二叉排序树的平均查找长度与二分法查找同量级，为 O（$\log_2 n$）。

按对称序遍历一棵二叉排序树，可以得到排好序的关键码序列，二叉排序树的插入要保证插入后仍具有二叉排序树的特点。插入的过程是：首先在二叉排序树中按查找的方法查找待插入的关键码值，查找会停止在某结点的空的左指针或右指针处，说明二叉排序树中无此关键码值。这时将关键码值插入到该位置作为一个新的树叶。

最佳二叉排序树

二叉排序树的平均查找长度是 O（$\log_2 n$），但并不是每一棵二叉排序树都有如此高的查找效率。在极端的情况下，查找长度甚至能达到 n。同一个关键码集合，其关键码插入二叉排序树的次序不同，就构成不同的二叉排序树。这些二叉排序树的平均查找长度也不相同，把平均查找长度最小的二叉排序树称做最佳二叉排序树。

最佳二叉排序树在结构上具有这样的特点：除了最下面的一层可以不满外，其他各层都是充满的。显然最佳二叉排序树的查找效率较高。

平衡的二叉排序树

平衡的二叉排序树是对二叉排序树的一种“平衡化”处理。结点的平衡因子定义为其右子树高度减去左子树高度。若任一结点的平衡因子均取值-l，或 0，或+l，则此二叉排序树为平衡的二叉排序树（AVL 树）。

平衡的二叉排序树的查找方法与一般的二叉排序树完全一样，其优点是总能保持查找长度为 O（$\log_2 n$）量级。

往平衡的二叉排序树插入新结点时，需要对树的结构进行必要调整，以动态地保持平衡二叉排序树的特点。

2.2.7 B 树和 B+树

B 树和 B+树用于组织外存储器中文件的动态索引结构。所谓动态索引结构是指文件创建，初始装入记录时生成索引结构，在系统运行过程中插入或删除记录时，索引结构本身可能发生改变，改变索引结构的目的是为了保持较好的性能，如较高的查找效率。

B 树

B 树是一种平衡的多路查找树。一棵 m 阶 B 树或者为空，或者满足以下条件：

（1）每个结点至多有 m 棵子树；

（2）根结点至少有两棵子树（除非根结点为叶结点，此时 B 树只有一个结点）；

（3）除根结点外，其它每个分支结点至少有「$\frac{m}{2}$」棵子树；

（4）非叶结点包含如下信息：{n，P_0，K_1，P_1，K_2，…，K_n，P_n}，其中，$K_i(1\leq i\leq n)$为关键码，且满足 $K_i<K_{i+1}(i=1，2，\cdots，n-1)$；$P_i(1\leq i\leq n)$为指向子树根结点的指针。

（5）叶结点均出现在同一层次，且不含任何关键码信息（可以把叶结点看成实际上不

存在的外部结点，指向“叶结点”的指针为空；实际为标志查找失败使用)。

B 树的运算

（1）查找

在 B 树里查找给定的关键码的方法是，首先把根结点取来，在根结点所包含的关键码 K_1，…，K_j 中查找给定的关键码值（当结点包含的关键码不多时，就用顺序查找；当结点包含的关键码数目较多时，可以用二分法查找)，若找到则查找成功；否则，一定可以确定要查的关键码值是在某个 K_i 和 K_{i+1}；之间（因为在结点内部的关键码是排序的)，于是取 P_i 所指向的结点继续查找，直到找到，或 Pi=∧查找失败。在含有 n 个有关键码的 m 阶 B 树上进行查找时，从根结点到关键码所在结点的路径上涉及的结点数不超过 $\log_{[\frac{m}{2}]}(\frac{n+1}{2})+1$。

（2）插入

在 B 树里插入一个关键码的方法是：对于叶结点处于第 i 层的 B 树，插入的关键码总是进入第 i－l 层的结点。插入可能导致 B 树朝着根的方向生长。如果要插入的那个结点已满时，不能再往里插了。在这种情况下，要把这个结点分裂为两个，并把中间的一个关键码拿出来插到结点的双亲结点里去。双亲结点也可能是满的，就需要再分裂，再往上插。最坏的情况，这个过程可能一直传到根，如果需要分裂根，由于根是没有双亲的，这时就建立一个新的根结点。整个 B 树增加了一层。

（3）删除

删除的过程是类似的，但要稍微复杂。如果删除的关键码不在第 i－1 层，则先把此关键码与它在 B 树里的后继对换位置，然后再删除该关键码。

如果删除的关键码在第 i－1 层，则把它从它所在的结点里去掉，这可能导致此结点所包含的关键码的个数小于 $[\frac{m}{2}]-1$。这种情况下，考察该结点的左或右兄弟，从兄弟结点移若干个关键码到该结点中来（这也涉及到它们的双亲结点中的一个关键码要做相应变化)，使两个结点所含关键码个数基本相同。只有在兄弟结点的关键码个数也很少，刚好等于 $[\frac{m}{2}]-1$ 时，这个移动不能进行。这种情况下，要把删除了关键码的结点、它的兄弟结点及它们的双亲结点中的一个关键码合并为一个结点。

从双亲结点中这样拿出一个关键码有时也可能导致进一步的合并，甚至这种合并一直传到根结点。在根结点只包含一个关键码的情况下，将发生直到根结点的合并，使根结点和它的两个子女进行合并，形成新的根结点，从而使整个树减少了一层。

B+树

在实际应用中产生了一些 B 树的变形。B+树是其中应用得最广泛的一种。B+树的所有关键码都出现在叶结点上，上面各层结点中的关键码均是下一层相应结点中最大关键码的复写。m 阶 B+树的结构定义如下：

1．每个结点至多有 m 棵子树；

2．根结点至少有两棵子树（除非根结点为叶结点，此时 B 树只有一个结点）；

3．除根结点外，其它每个分支结点至少有「$\frac{m}{2}$」棵子树；

（4）有λ个子树的结点必有λ个关键码，它包含如下信息：(P_0，K_1，P_1，K_2，…，$P_{\lambda-1}$，K_λ)

B+树的查找、插入、删除过程基本上与 B 树类似，只是有如下差别：当查找时，若在上层已找到待查的关键码，并不停止，而是继续沿指针向下一直查到叶结点层的这个关键码。当插入时，一个全满的 m 个关键码的结点再加一个新关键码，就要进行分裂，分裂为两个结点分别有「$\frac{m+1}{2}$」和「$\frac{m+1}{2}$」个关键码，并要保证上层结点中有这两个结点的最大关键码或最小关键码。

B 树只适于随机查找，不适于顺序查找，而 B+树把所有的关键码都存在叶结点上，这就为顺序查找也提供了方便，在实际中用的大多是 B 树的这种变形——B+树。

2.2.8 排序

插入排序

（1）直接插入排序

直接插入排序的基本思想是把表中元素依次插入一个已排好序的表中，就像人们打扑克摸牌时把牌插入手中的若干张牌里一样。

比如下面的序列：

49 38 65 97 76 13 27

如果用直接插入排序将序列按从小到大排列，排序过程是：

第一趟 [38 49] 65 97 76 13 27

第二趟 [38 49 65] 97 76 13 27

第三趟 [38 49 65 97] 76 13 27

第四趟 [38 49 65 76 97] 13 27

第五趟 [13 38 49 65 76 97] 27

第六趟 13 27 38 49 65 76 97

表中 n 个元素依次插入的比较次数为 1＋2＋3＋…＋（n-1）=n（n-1）/2。在插入时，元素的移动次数最多为 1＋2＋3＋…＋（n-l）=n（n-1）/2。如果表中元素已排好序，则只需比较 n-l 次，而移动次数为 0。

（2）二分法插入排序

在已排好的序列中找插入位置时，用二分法查找，找到插入位置后和直接插入排序法同样处理。

二分法插入排序关键码比较次数降为 O（$n\log_2 n$），记录移动个数仍为 O（n^2）。

（3）shell 排序法（缩小增量法）

按增量将文件分组。首先取增量 d_1＜n，把全部记录分成 d_1 个组，所有距离为 d_1 倍数

的记录放在一组中，各组内用插入法排序，然后取 $d_2<d_1$，重复上述分组和排序工作，直至取 d=1，即所有记录放在一个组中排序为止。

比如下面的序列：

49 38 65 97 76 13 27 49’

如果用 shell 排序，排序过程是：

第一趟 49 13 27 49’ 76 38 65 97 （d＝4，[49、49’、97]为一组，[13、76]为一组，[27、38]为一组）

第二趟 27 13 49 38 65 49’ 76 97 （d＝2，[27、49、65、76]为一组，[13、38、49’、97]为一组

第三趟 13 27 38 49 65 76 97 （d＝1）

选择排序

1．直接选择排序

直接选择排序的基本思想是在表的 n 个元素中，经过 n-l 次比较得到其最大值（或最小值，下同），这就排好了第一个元素；再经过 n-2 次比较得到余下元素中的最大值，这就排好了第二个元素…直到比较 1 次后排好第 n-1 个元素，第 n 个元素的位置也就自然确定了。

比如下面的序列：

49 38 65 97 76 13 27

如果用直接选择排序将序列按从小到大排列，排序过程是：

第一趟 13 38 65 97 76 49 27 （13 最小，与 49 交换位置）

第二趟 13 27 65 97 76 49 38 （27 最小，与 38 交换位置）

第三趟 13 27 38 97 76 49 65 （38 最小，与 65 交换位置）

第四趟 13 27 38 49 76 97 65 （49 最小，与 97 交换位置）

第五趟 13 27 38 49 65 97 76 （65 最小，与 76 交换位置）

第六趟 13 27 38 49 65 76 97 （76 最小，与 97 交换位置）

所需的比较次数为（n-1）＋（n-2）＋…＋1＝n（n-1）/2。所需移动次数最多也为 n（n-l）/2。如果表中元素排好序，也需要比较 n（n-l）/2 次，而移动次数为 0。

2．堆排序

堆排序是完全二叉树结构的一个重要应用，是对直接选择排序的改进。堆是一个关键码序列（K_1，K_2，…，K_n），它具有如下特性：

$$K_i \le K_{2i},\ K_i \le K_{2i+1} \qquad i=1,2,\cdots,[n/2]$$

堆实质上是一棵完全二叉树结点的层次序列，此完全二叉树的每个结点对应于一个关键码，根结点对应于关键码 K_1。堆的特性在此完全二叉树里解释为：完全二叉树中任一结点的关键码值都小于或等于它的两个子女结点的关键码值。由此可知，根结点 K 是完全二叉树中关键码值最小的结点，也就是堆中的最小元素，并且具有此特性的完全二叉树的任何一棵子树都对应于一个堆。例如，关键码序列{6，24，17，69，95，73，72，74}就是一个堆，表示成完全二叉树的形式如图 3－3 所示。

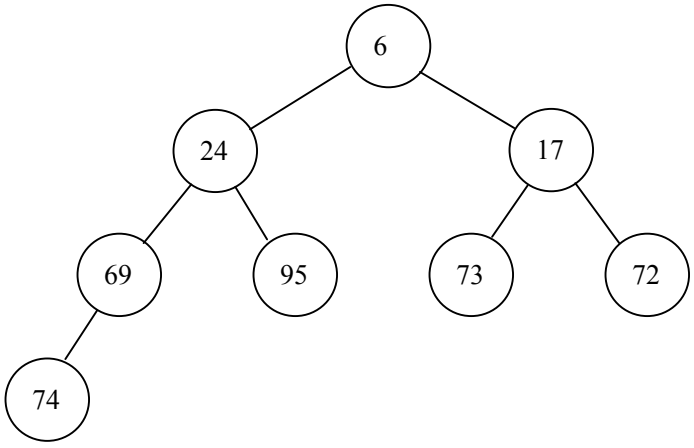


图 3－3　堆

堆排序的基本思想是：对一组待排序的关键码，首先把它们按堆的定义排成一个序列（称为建堆），这就找到了最小的关键码，然后将最小的关键码取出，用剩下的关键码再建堆，便得到次最小的关键码，如此反复进行，直到将全部关键码排好序为止。

推排序的执行时间为 $O(n\log_2 n)$，且仅需要一个用于交换的附加存储结点。因此堆排序是一种适合于较大文件的排序方法。

起泡排序

起泡排序的基本思想是将表中元素两个相邻元素依次比较，若不符合排序要求，则交换位置，这样经过 n-1 次比较后，将确定出最大（或最小）元素的位置，这称为一趟扫描。经过 n-1 次扫描后，就完成了整个表的排序。

起泡的方法：先将序列中的第 1 个元素与第 2 个元素进行比较，若前者大于后者，则两个元素交换位置；否则不交换。然后第 2 个元素与第 3 个元素比较，若前者大于后者，则两个元素交换位置；否则不交换。依次类推，直到第 n-1 个元素与第 n 个元素比较（或交换）为止。经过如此一趟排序，使得 n 个元素中最大者被安置在的 n 个位置上。此后，再对前 n-1 个元素进行同样的过程，使得 n-1 个元素中最大者被安置在的 n-1 个位置上。然后，再对前 n-2 个元素重复上述过程。直到某一趟排序过程中不出现元素交换位置的动作，排序结束。通过相邻元素之间的比较和交换，使值较小的元素逐步从后面移到前面，就像水底下的气泡一样向上冒，所以又称为冒泡排序。

对下面的序列：

49　38　65　97　76　13　27

如果用起泡排序将序列按从小到大排列，排序过程是：

第一趟　38　49　65　76　13　27　97　　（97 起泡到尾部）

第二趟　38　49　65　13　27　76　97　　（76 起泡到尾部）

第三趟　38　49　13　27　65　76　97　　（65 起泡到尾部）

第四趟　38　13　27　49　65　76　97　　（49 起泡到尾部）

第五趟　13　27　38　49　65　76　97　　（38 起泡到尾部）

第一趟扫描的比较次数是 n－1，第二趟扫描的比较次数是 n—2……，总的比较次数是

（n-1）＋（n－2）＋……＋1＝n（n-1）/2。

如果恰好表中元素按反序排列，则需要移动的次数为 3×n（n—l）/2。如果表中元素已排好序，并采用交换标志来表示并未发生过交换，则只需一趟扫描，只比较 n－1 次，就够了；当然，移动次数也是 0。

快速排序

快速排序是对起泡排序的一种改进，其基本思想是把表中某元素作为基准，将表划分为大于该值和小于该值的两部分，然后用递归的方法处理这两个子表，直到完成整个表的排序。

快速排序的比较次数为（n—1）＋（n—2）＋…＋1＝n（n—1）/2，移动次数最多也是 n（n 一 1）/2。如果每次的基准元素刚好是表的中值，使表分为大小相等的两个子表，则比较次数为 $nlOg_2n$；如果表已排好序，则移动次数为 0。

快速排序又称为分区交换排序，为了节省空间，数据交换位置时可采取两端往中间夹入的方式，即先取出 K_1，这样空出前端第一个关键码的位置，用 K_1 与 K_n 相比较，若 K_n＞K_1，则将 K_n 留在原处不动，继续用 K_1 与 K_{n-1} 相比；若 K_n＜K_1，则将 K_n 移到原来 K_1 的位置，从而空出 K_n 的位置，这时用 K_1 的值再与 K_2，K_3，…相比，找出一个大于 K_1 的关键码，将它移动到后面刚刚空出的位置，如此往复比较，一步一步地往中间夹入，便将大于 K_1 的关键码都移动到后部，而把小于或等于 K_1 的关键码都移动到前部，最后在空出的位置上填入 K_1，便完成了一趟排序的过程。对分开的两部分继续分别执行上述过程，最终可以实现全部排序，如下面的序列：

49　38　65　97　76　13　27

先选第一个元素 49 作为基准，交换过程是：

第一次交换　27　38　65　97　76　13　49　　（49 与 27 交换）

第二次交换　27　38　49　97　76　13　65　　（49 与 65 交换）

第三次交换　27　38　13　97　76　49　65　　（49 与 13 交换）

第四次交换　27　38　13　49　76　97　65　　（49 与 97 交换）

则第一趟排序后序列分为两部分：

[27　38　13]　49　[76　97　65　]

后续各趟排序状态为：

第二趟　[13] 38　[27] 49　[76　97　65　]

第三趟　13　38　27　49　[65] 76　[97]

归并排序

归并排序的基本思想是表中元素两两比较排序，使表中的 n 个元素变成 n_0 个已排序的组，再两两组比较，而变成 n／4 个已排序的组……，直到表中只含有一个已排序的组，即完成排序。

所需要的比较次数为 $nlog_2n$，移动次数为 n。若表已排好序，则比较次数仍为 $nlog_2n$，但移动次数为 0。

常用排序方法的性能比较

如表 3－1 所示：

表 3－1 常用排序方法的性能比较

排序方法	平均时间	最坏情况的时间	辅助储存
起泡法、直接选择法、直接插入法	$O(n^2)$	$O(n^2)$	O(1)
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	O(1)
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	O(n)

如果在待排序的表中含有多个码值相同的记录，经过排序后，这些记录的相对次序不变，则称这种排序方法是稳定的，否则是不稳定的。

根据上述说法，可以看出直接插入法、归并法是稳定的；而直接选择法、起泡法、快速排序法、堆排序法是不稳定的。

2.2.9 查找

顺序查找

顺序查找是将待查找的关键码值与线性表中个结点的关键码值逐一比较，直到找到所需的记录，查找成功；或者在表中找不到所需记录而查找失败。

顺序查找的优点是对线性表的结点的逻辑次序无要求（不必按关键码值排序），对线性表的存储结构无要求（顺序存储、链接存储皆可）。但顺序查找的平均检索长度长，设线性表有 n 个元素，则最多查找次数为 n，最少查找次数为 1，检索长度与表中结点个数 n 成正比。若检索的关键码与线性表里第 i 个结点的关键码值相等，则需要进行 i 次比较才能查到，设检索每个关键码值的概率相等，均为 1/n，则成功检索的平均比较次数是：

$$\frac{1}{n}\sum_{i=1}^{n} i = \frac{n+1}{2}$$

二分法查找

二分法查找要求线性表结点按关键码值排序且以顺序方式存储。在查找时，首先与表的中间位置上结点的关键值比较，若相等则查找成功；否则根据比较结果确定下一步在表的前半部或后半部中继续进行。

例如下面的关键码序列：

2　5　8　11　15　16　22　24　27　35　50

用二分法查找关键码值 22，查找过程如下：

第一次比较　[2　5　8　11　15　16　22　24　27　35　50]　（中点值为 16）

第二次比较　2　5　8　11　15　16　[22　24　27　35　50]　（中点值为 27）

第三次比较　2　5　8　11　15　16　[22　24]　27　35　50　（中点值为 22）

二分法查找的效率较高，设线性表有 n 个元素，则最多的查找次数为大于 $\log_2 n$ 的最小整数，最少的查找次数为 1。

分块查找

分块查找把线性表分成若干块，块内结点不必有序，但块与块之间必须有序，即每一块中各结点的关键值必须大于（或小于，与此类推）前一块最大关键值。为加快查找，还要建立一个索引表，表中给出每一块的最大关键值和指向块内第一个结点位置的指针。

分块查找分两步进行，先查索引表，确定要找的记录在哪一块；然后再在相应的块中查找。分块查找适合于线性表很大，数据又是动态变化的情况。在查索引表时，可采用顺序法或二分法；在块内查找所求记录时，采用顺序法。由于分块而缩小了查找范围，从而加快查找速度。

散列表查找

根据关键值，就可以迅速找到该记录所对应的存储位置，这就是建立在散列函数基础上的散列查找。设记录的关键值为 k，则该记录的存储位置可用散列函数 H 来计算 H＝H（k）。

散列表的一个重要特征是平均检索长度不直接依赖于元素个数。平均检索长度不随表中元素个数的增加而增加，而是随负载因子 α 的增大而增加。如果安排得好，平均检索长度可以小于 1.5。

第 3 章　操作系统

考纲要求
1．操作系统的基本概念、主要功能和主要分类。
2．进程、线程、进程间通信的基本概念
3．存储管理、文件管理、设备管理的主要技术
4．典型操作系统的使用

3.1　知识点

3.1.1　操作系统的概念

计算机软件包括系统软件和应用软件，操作系统是配置在计算机上必不可少的系统软件，是对硬件系统的功能扩充。无论是 PC 机还是巨型计算机，都配置了至少一种操作系统，其他的软件都是建立在操作系统基础之上的。

操作系统是一组程序模块的组合，具有如下基本功能：

（1）它们有效的控制和管理计算机系统中的硬件和软件资源。

（2）它们合理的组织计算机工作流程，以改善系统性能。

（3）提供一个易于使用、功能强大的工作环境，从而在计算机和用户之间起到接口的作用。

3.1.2　操作系统的功能

从计算机的资源和软件资源管理的角度来看，操作系统的功能主要包括以下 5 个方面：

（1）处理器管理。它负责为每一个进程分配处理器。处理器管理也称作进程管理。

（2）存储管理。它负责有效的管理系统的存储资源，特别是内存资源。

（3）文件管理。文件管理是对计算机系统中软件资源的管理，其任务是采用统一、标准的方法对文件进行相应的管理。

（4）设备管理。设备管理负责组织和管理系统中的各种输入/输出设备，完成实际的

输入/输出操作。

（5）作业管理。作业管理负责向用户提供如何使用操作系统完成各种信息处理任务的手段和方法。

3.1.3 操作系统的类型

按照操作系统的功能分类，操作系统可以分为多种类型。

（1）单用户单任务操作系统。该系统每次只能支持一个用户并且只能运行一个程序或任务。

（2）批处理操作系统。该系统允许将若干个用户的作业按一定顺序排列、成批的提交给计算机完成。

（3）分时操作系统。分时系统是指一台主机带有若干台终端，多个用户共享一台计算机，而计算机的中央处理器在时间上划分成许多片段（即时间片）并轮流分配给每个终端用户。

（4）实时操作系统。实时操作系统包括实时控制系统和实时处理系统。其主要特点是对外界的作用和信息响应及时、迅速，并要求可靠性高，以满足实时处理的需求。

（5）网络操作系统。网络操作系统是为计算机组网而配置的操作系统。

（6）分布式操作系统。分布式操作系统是为分布式计算机系统配置的操作系统。

3.1.4 批处理操作系统

批处理就是将若干个用户的作业按照一定顺序排列，成批的提交给计算机完成。批处理操作系统具有如下主要特征：

（1）用户脱机使用计算机。用户提交作业以后就脱离了计算机，用户与计算机之间没有交互，不能直接控制作业的运行，作业一旦开始执行，就不能中止。

（2）高效率。操作系统自动的选择合适的作业运行，在运行过程中用户不干预自己的作业，从而大大的提高了系统资源的利用率和作业的吞吐能力。

（3）批处理操作系统具有的一些主要缺点：

（4）缺乏交互性。用户一旦提交了作业，就失去了对作业的控制能力。

（5）平均周转时间长。因为是批处理，一个作业一旦提交运行就会运行到完成，作业的周转时间增长。

3.1.5 分时操作系统

分时操作系统是指一台主机带有若干个终端，多个用户共享一台计算机，而计算机的中央处理器在时间上划分成长短相同或基本相同的时间段，每个时间段称为一个时间片。分时系统采用时间片轮转算法的方式，将 CPU 分配给多个用户。虽然每运行一个时间片就产生一次中断，但由于时间片足够的小，每个终端的用户感觉不到其他用户的存在，就好像独占一台计算机一样。分时操作系统的主要特点是：多用户、交互性、及时性、独立性。

3.1.6 实时操作系统

实时操作系统也是一类联机操作系统，又称实时系统。实时操作系统包括实时控制系统和实时处理系统。它的主要特点是对外界的作用和信息响应及时、迅速，并要求系统具有较高的可靠性，以满足实时处理的需求。

3.1.7 网络操作系统

网络操作系统就是通过通信设备和通信线路将地理位置分散、功能独立的计算机系统连接起来，以实现资源共享、信息交换、交互操作和协作处理的计算机系统。它是计算机技术和通信技术结合的产物。

3.1.8 分布式操作系统

计算机网络使得网络上的用户实现资源共享。但当本地需要远程的另一台计算机协作处理一个任务时，必须先告知对方，请求对方准备好作业运行环境，然后才能完成其指派的任务，这种使用环境让用户感觉到其他网络资源与自己存在不同，不能象使用本地资源那样使用远程资源。

3.1.9 进程的基本概念

在计算机科学中，程序是指令的有序集合，它反映了用户要求计算机系统完成的各种操作和这些操作的执行顺序。在早期的单道程序系统中，所编写的程序在执行时是顺序执行的。进程是具有独立功能的程序对某个数据集独立运行和分配资源的基本单位。进程具有如下特征：

(1) 动态性。
(2) 并发性
(3) 独立性
(4) 制约性

3.1.10 线程的基本概念

线程是进程中可以独立执行的子任务，一个进程中可以有一个或多个线程，每个线程都有一个唯一的标识符。线程具有以下属性：

(1) 每个线程有一个唯一的标识符和一张线程描述表，线程描述表记录了线程执行时的寄存器和堆栈等现场状态。

(2) 不同的线程可以执行相同的程序，即同一个服务程序被不同用户调用时操作系统为它们创建成不同的线程。

（3）同一进程中的各个线程共享分配给进程的主存空间。

（4）线程是处理器的调度单位，多个线程可以并发执行。在单处理器的计算机系统中，各线程可以交替的占用处理器。在多个处理器的计算机系统中，各线程可同时占用不同的处理器，若各个处理器同时为一个进程内各线程服务则可缩短进程的处理时间。

（5）一个线程被创建后便开始了它的生命周期，直到终止，线程在生命周期内会经历等待状态、就绪状态和运行状态等各种状态的变化。

3.1.11 存储管理

存储管理具有以下主要功能：

（1）内存分配和回收。在多道程序中，要存放操作系统中若干个用户作业，存储管理要适当的对它们分配内存空间，使它们不互相干扰，顺利运行，并提供合理的分配算法，以解决资源的回收问题和提高存储器的利用率。

（2）地址转换或重定位。当用户的程序调入主存时，必须将程序的逻辑地址转换为物理地址，包括对程序中有关地址的指令做相应的调整，这个过程称为重定位。重定位方法有两种：静态地址重定位和动态地址重定位。

（3）存储保护。在多道程序并发运行的环境下，内存中的系统程序和数据段可供不同的用户进程共享。为了保护存储区内各类程序和信息不受某些错误程序的干扰和破坏，保证系统的安全和进程对内存信息的正确访问，需要对内存中的程序和数据段采取保护措施。基本的内存保护方式有两种：越界保护和存取控制保护。

（4）存储扩展。单一连续的存储容量是有限的，为了满足用户的作业对内存空间的需要，操作系统采取将内、外存联合起来的措施，向用户提供一个容量比实际内存空间大得多的虚拟存储器。实现虚拟存储器的关键是提供快速有效地进行自动地址变换地硬件机构和相应地软件算法。

3.1.12 存储器分类

主存（内存）

主存是存放系统和用户指令以及数据的设备，是 CPU 可以直接访问的存储器。主存包括 RAM，ROM 两个部分，由于 ROM 是系统固定拥有的，所以通常讨论的主存指的是 RAM。

高速缓存（Cache）

高速缓存是一类速度很高、容量较小的存储器，用来存放主存中当前频繁使用的信息，以方便高速的 CPU 读取数据。

外存（辅存）

外存是指常用的硬盘、磁盘、光盘等，其存取速度比内存慢，但容量比内存大很多。外存的信息只有交换到内存之后才能访问。

3.1.13 虚拟内存

虚拟存储技术也称为虚拟内存技术。在程序装入时，不必将其全部读入到内存，而只需将当前需要执行的一部分页读入内存，就可让程序开始执行。在程序执行过程中，如果需要执行的指令或访问的数据尚未在内存，则由处理器通知操作系统将相应的页调入到内存，然后继续执行程序。

3.1.14 设备管理

设备管理主要完成以下任务：

（1）为用户提供一个方便而且独立于设备的接口。它将用户和硬件特性分开，用户在编制应用程序时不必关心外设的硬件细节，系统按照用户的要求控制设备工作。

（2）尽量提高外设的利用率。在计算机系统中，CPU 的速度比设备的速度高出很多，设备管理的一个最重要的任务就是极大限度的提高 CPU 与 I/O 设备之间、设备与设备之间并行工作程度，从而提高外部设备资源的使用效率。

设备管理的主要功能如下：

（1）设备分配。按照设备的类型和合理的分配算法，把设备和其他相关的硬件分配给请求该设备的进程，并在使用以后将其收回，以便其他进程使用。

（2）缓冲区管理。由于计算机 CPU 的执行速度和访问内存的速度比外围设备的数据流通速度快出很多，系统中通常设有缓冲区来暂存数据，以缓解 CPU 与外设速度不匹配的问题，设备管理程序负责对缓冲区进行分配、管理和释放工作。

（3）控制设备的 I/O 操作。当进程要求设备资源时，由设备控制程序控制具体的设备进行实际的 I/O 操作，并实现设备的中断处理。

3.2 重点难点

3.2.1 操作系统的概念

计算机软件包括系统软件和应用软件，操作系统是配置在计算机上的必不可少的最基本的系统软件，它是对硬件系统功能的首次扩充。操作系统用来管理硬件和软件资源、控制程序运行、改善人机界面和为应用软件提供支持。

配置操作系统的目的有两个方面：

（1）从系统资源管理的角度看，操作系统能合理地组织和管理计算机工作流程，最大限度地发挥计算机资源的利用率。计算机的很多资源是可以为多个用户共享的，如 C P U、内存和输出/输入设备等等，操作系统必须准确、高效率、公平地分配调用资源，尽可能使多个操作和任务并发地进行。

（2）从方便用户的角度看，直接使用一台“ 裸机 ”将是很不现实的，操作系统是在“ 裸

机”上扩充的最基本的系统软件，是系统软件的核心。它使得计算机变得容易理解和便于使用，为计算机和用户之间提供了一个良好的界面和接口。

3.2.2 进程间的通信

用户的一个作业通常被分成若干个进程并发地执行，它们之间并不是彼此独立的，既要相互竞争有限的系统资源，又要保持联系才能协调一致地完成指定地任务。这种联系是指进程之间需要交换一定数量地信息，称为进程通信。

进程通信是为了解决进程的同步和互斥的问题，同时可在进程间传送信息，以使系统内的若干个进程有序的运行。进程通信根据通信内容可分为控制信息的传送和大批量数据的传送。进程间控制信息的交换称为低级通信，目的是为了控制进程的执行速度。大批量的数据交换称为高级通信，目的是为了交换信息。

3.2.3 进程间的同步与互斥

进程同步

进程之间的是异步运行的，但当他们要协调一致地完成一个任务时，常常需要协调一下它们的工作，直到完成任务。进程之间的这种协调配合的关系称之为进程的同步。

进程的互斥

为了充分利用系统资源，多个进程要共享一些软件资源或硬件资源，而部分资源只能在一段时间内由一个进程使用，其他进程不能使用，这就引起进程之间为使用这类资源而相互竞争。

临界资源和临界区

一次只允许一个进程使用的资源称为临界资源，一个进程中访问临界资源的那段程序代码称为临界区。在计算机系统中许多物理设备都是临界资源，软件中的某些变量、数据等同样也属于临界资源，它们不允许两个及两个以上的进程同时对其进行访问或修改。

3.2.4 信号量和 P-V 操作

信号量

在系统中信号量（S）是一个整数。谓 $S \geq 0$ 时，表示可供并发进程使用的资源实体数；当 $S<0$ 时，S 代表等待使用临界区的进程数目。

用 P-V 操作实现进程间的互斥

令信号量 S 的初值为 1，假设有 P_1 和 P_2 两个进程相互竞争进入临界区，用 P-V 操作实现进程互斥，可将程序设计为：

设 S 的初始值为 1

P_1 进程　　P_2 进程

……	……
P（S）	P（S）
临界区	临界区
V（S）	V（S）
……	……

假定 P_1 进程先运行，执行 P（S）操作后将信号量 S 的值减 1，即 S 为 0，然后进入临界区。这时，如果信号量 P_2 执行 P（S）操作，则将信号量 S 的值减 1，即 S 为-1，P_2 进程被封锁并插入到与信号量 S 相对应的等待队列中。当 P_1 退出临界区后，将信号量 S 的值加 1，即 S 为 0，这时系统将处于封锁状态的 P_2 恢复为就绪状态，P_2 一旦拥有 CPU 以后，马上进入临界区。当 P_2 退出临界区后，将 S 恢复为 1

用 P-V 操作实现进程间的同步

令信号量 S_1 的初始值为 0，假设进程 P_1 必须在 X_1 点与进程 P_2 同步，程序设计如下：

P_1 进程	P_2 进程
……	……
X_1P（S）	X_2V（S）
临界区	临界区
V（S）	V（S）
……	……

假定进程 P_1 先运行，到 X_1 点执行 P（S）操作，信号量 S 的值减 1，即 S 为-1，P_1 被封锁并转入到信号量 S 的等待队列中，这时系统转而执行进程 P_2。必须等待进程 P_2 执行到 X_2 点到执行 V（S）操作后，信号量 S 的值加 1，即 S 为 0，才能将 P_1 从封锁状态转入就绪状态，P_1 等待拥有了 CPU 后就可以从 X_1 点运行下去。

3.2.5 消息缓冲通信

消息通信的基本思路是：让系统管理一组消息缓冲区，每一个缓冲区可以存放一个信息。发送进程在发送消息之前，先在自己的内存空间设置一个发送区，把欲发送的消息填入其中，然后在向系统申请到一个消息缓冲区以后，将发送区里的消息发送到缓冲区中。然后系统将缓冲区连接到接收进程的消息队列中，接收进程在适当的时候，从消息队列中将消息读出并送到自己的消息接收区去使用，同时释放消息缓冲区。

3.2.6 进程与线程的比较

进程存在的缺点

（1）每个进程要占用一个进程控制块和一个私有的主存空间，开销较大

（2）进程之间的通信必须要由通信机制来完成，速度较慢

（3）进程增多会给调度和控制带来复杂性，增加了死锁的机会。

多线程技术的特性

（1）创建线程不需另行分配资源，因而创建线程的速度比创建进程的速度快，且系统的开销也比较小。

（2）线程之间的通信在同一地址空间中进行，故不需要额外的通信机制，使通信更简便，信息传送速度也快。

（3）线程能独立运行，能充分利用和发挥处理器与外围设备并行工作的能力。

当多个线程协作完成一项任务时，线程间必须要同步以协调工作。线程的同步是指一个线程主动暂停执行，等待其他的线程执行某些操作。支持线程的操作系统都必须提供一种同步方式，不同的操作系统提供的同步方式可能不同，但必须使一个线程在需要时能等待其他线程完成某项工作，也应允许一个线程通知其他线程，它已完成了某项工作。

3.2.7 单一连续的存储管理

单一连续存储区管理是一种最简单的存储管理方式，主要用于单道环境。因为只有一个用户或作业，进程或作业执行时除了系统占用部分外，可用内存区全部被一个进程或作业占用。在可以将内存划分为 3 个区域：系统区、用户区和空闲区。当 CPU 要运行一个作业时内存中必须没有其他作业占用。除非用户释放或作业运行完毕，系统才会回收整个用户区。单一连续区存储管理主要采用静态重定位方式。进程或作业进入内存后，要等到进程或作业执行完成后才能释放内存。

3.2.8 分区存储管理

分区存储管理是在单一连续区的基础上发展起来的一种存储管理方法，是满足多道程序设计的最简单的存储管理模式。基本原理是将内存划分为若干个分区，按照连续分配的方式将每个分区分配给作业，使若干个进程可以并发执行。硬件需要提供必要的保护手段，保证各个作业互不干扰，按分区的划分方式可分为固定分区和可变式分区。

1．固定式分区。存储器在处理作业前已经划分为若干个任意大小的区域，然后，将这些区域分配给每个用户作业。区域的划分可以由操作系统决定，在系统的整个执行过程中，区域的大小和边界是不能改变的。用户必须为每个作业规定最大的存储量，然后由存储管理程序找出一个足够大的未分配分区根配给它。

2．可变式分区。 由于可变式分区是动态地分配内存，而系统在开始运行以后，对存储空间会出现一系列的分配与释放，所以应该考虑对内存占用情况记录和存储空间的分配策略。对内存占用情况可以用一些表格加以记录。分配策略常常采用不同的分配算法，主要有：最佳使用算法、最坏适应算法和首次适应算法。

3.2.9 页式存储管理

内存分配

页式存储管理把内存空间分成相同大小的若干个存储区域，每个区域称为一个“ 块 ”，

把用户的作业地址空间也分成与“块”同样大小的若干个“片断”，称为“页”。页和块均从 0 开始按顺序编号，分别称为“页号”和“块号”。

页表

在页式存储管理中，每一个作业在内存中的页面分散在不连续的块中，有时作业的全部页并不是都调入了内存，那么当进程运行时，就需要进行虚拟地址和是在地址之间的转换。因此系统在内存中的固定区域为每个作业建立了一个页面映象表，简称“页表”。页表中包含有页号、块号和状态。其中状态是表示该页是否已经调入内存，1 为已经调入，0 为尚未调入。

快表

用页表进行地址变换，要先访问内存中的页表，然后根据形成的物理地址再次访问内存。执行一条指令要进行两次访问操作。从而使系统的处理速度大大降低。为了提高地址变换的速度，许多页式系统在地址变换机构中增设了一个且有并行查询能力的特殊高速缓存，用来存放当前访问最频繁的少数活动页号和对应的块号，称为“快表”。

缺页中断处理

当进程访问的页在页表中的状态为“0”时，系统发出缺页中断，并暂停用户作业的运行，转入缺页中断处理程序。缺页中断处理程序负责将所需的页从外存调入内存，若内存中剩余有空白区，则将页调入内存，并且修改页表的内容，返回用户程序，从被中断指令处继续执行。若内存 中已无空白区，则要选择某一页淘汰到外存，以装入缺页。

3.2.10 页面淘汰算法

若页面淘汰处理不当，则会出现被淘汰的页面需要访问而再次调入内存的现象，这种反反复复的频繁调度，称为“抖动”。抖动式计算机的大部分时间都用在了页面的调度上，降低了系统的运行速度。为了避免抖动现象，常用以下调度算法：

（1）先进先出（FIFO）

（2）最近最久未使用算法（LRU）

（3）最不常用算法（LFU）

3.2.11 段页式存储管理

段页式存储管理方法结合了分段式存储管理和页式存储管理的优点，具有分段在逻辑上的优点和分页在存储空间上的优点。段页式管理方法是将一个用户程序分为若干个段，各段具有各自的段名，在将每段划分为大小相同的页，最后不足一页的部分仍然占一页。段页式管理的逻辑地址由 3 个部分组成，即段号、页号和页内地址。

为了实现从逻辑地址到物理地址的转换，系统为每个作业建立建立一个段表，为每一段建立一个页表。段表中的状态指出该段是否在内存，地址字段给出页表的起始位置；页表的状态表示该页是否在内存，地址字段给出该页在内存的块号。

在段页式存储管理方式中，需要为段表设置一个段表寄存器，用于存放段表的长度和起始位置。当作业进行访问时，首先由段表地址寄存器找到段表的起始位置，根据段表起始位置和逻辑地址中的段号找到该段的段表项，从中得到该段表项所对应的页表在内存中的地址，然后根据逻辑地址中的段内号找到相应页表的位置。取出该页的物理块号，由块号和逻辑地址中的页内地址构成物理地址。

3.2.12 缓冲技术

单缓冲

单缓冲是在系统内只设置一个缓冲区。在输入时，输入设备先将数据送入缓冲区，CPU 从缓冲区中把数据取出进行处理，外设在将另外的数据装入缓冲区。输出时，CPU 将数据先装入缓冲区，输出设备在从缓冲区将数据取走。

双缓冲

双缓冲是在系统内设置两个缓冲区。CPU 和外设交替使用缓冲区，当 CPU 访问一个缓冲区时，外设可以使用另外一个缓冲区，从而解决临界区资源互斥的问题，提高了效率。

多缓冲

多缓冲技术是在系统内设置多个缓冲区，并且把多个缓冲区连接起来，一部分用于输入，另一部分用于输出。这些缓冲区扩大了缓冲区的容量，可以为不同的外设分别访问，能很好的与 CPU 和外设协调工作。

3.2.13 文件和文件目录

文件

文件是具有文件名的一组相关元素的有序集合。文件系统是指操作系统提供的与文件管理有关的软件被管理的文件以及实施管理所涉及的一些数据结构的总体。一个文件系统必须完成以下工作：

（1）对磁盘等辅助存储器空间进行统一的管理，以便合理地存放文件。

（2）为了实施“ 按名存取 ”，应该有一个用户可见的文件逻辑结构，让用户能够根据文件逻辑结构所给予的方式存取和加工信息。

（3）具有对存储在设备上的文件信息进行查找。

（4）文件在存储设备上应该是合理地顺序存放，以便于信息地存放和处理。

（5）实现文件信息共享，提供可靠地保护和保密措施。

文件目录

为了实现“ 按名存取 ”，系统为所有存入系统的文件建立一个从文件名到文件存储地址的映射。映射信息和其他管理信息组成了文件的说明，这个说明称为文件控制块。文件控制块的有序集合称为文件目录。文件目录提供了用户与文件系统之间的结构。文件目录的结构形式按系统的大小分为一级目录、二级目录和多级目录。

3.2.14 网络环境下 Windows 安全特性

Windows 98 环境下实现安全性的方法有以下几种：

（1）控制用户对资源的访问和改变软硬件设置的能力，这可以通过使用用户配置文件和系统策略来实现。

（2）在用户访问 Windows 工作站前，网络服务器对用户进行身份验证。

（3）实行用户级安全性。

（4）依靠服务器的安全性来保护重要的数据文件。

3.2.15 计算机病毒的特性

（1）灵活性

计算机病毒都是一些可以直接运行或间接运行的程序，它们小巧灵活，一般占有很少的字节，可以隐藏在可执行程序或数据文件中，不易被人发现。

（2）传染性

计算机病毒传染性是指病毒程序在计算机系统中的传播和扩散。病毒程序进入计算机系统后等待时机修改别的程序并把自身复制进去。在计算机运行过程中不断自我复制，不断感染别的程序。被感染的程序在运行时又会继续传染其他程序，于是很快就传染到整个计算机系统。在计算机网络中，病毒程序的传染速度更快，受害面也更大。计算机病毒有很强大的传染性，通过自我复制，计算机病毒可以迅速地在程序之间、计算机之间、计算机网络之间传播。

（3）隐蔽性

病毒程序一般都短小精悍，技巧性相当高，极具隐蔽性，很难被发现。它通常依附于一定的媒介，不单独存在。因此，在病毒发作之前不容易被发现，而一旦发现，计算机系统往往已经被感染或受到破坏。

（4）潜伏性

计算机病毒可以长时间潜伏在文件中，在触发机制被满足之前，计算机病毒可能不会表现出任何症状，只有触发了特定的条件才会进行传染或对计算机系统进行破坏。触发条件可以是一个或多个，例如某个日期、某个时间、某个事件的出现，某个文件的使用次数以及某种特定的软硬件环境等。例如 CIH 病毒一般在 4 月 26 日会发作。

（5）破坏性

计算机病毒可能对计算机系统产生不同程度的损害，主要表现为占用系统资源、破坏文件和数据、干扰程序运行、打乱屏幕显示甚至摧毁系统等，其破坏方式取决于病毒程序的设计者。

计算机病毒造成的后果，有良性和恶性之分。良性病毒只占用系统资源或干扰系统工作，并不破坏系统数据；恶性病毒一旦发作则会破坏系统数据、覆盖或删除文件甚至造成系统瘫痪，如黑色星期五病毒、磁盘杀手病毒等。

PC 机的病毒防治

计算机病毒危害很大。使用计算机系统，尤其是微型计算机系统，必须采取有效措施，防止计算机病毒的感染和发作。

（1）人工预防

人工预防也称标志免疫法。因为任何一种病毒均有一定标志，将此标志固定在某一位置，然后修复程序，达到免疫的目的。

（2）软件预防

目前主要是使用计算机病毒的疫苗程序，这种程序能够监督系统运行，并防止某些病毒入侵。国际上推出的疫苗产品如英国的 Vaccin 软件，它发现磁盘及内存有变化时，就立即通知用户，由用户采取措施处理。

（3）硬件预防

硬件预防主要采取两种方法：一是改变计算机系统结构；二是插入附加固件。目前主要是采用后者，即将防病毒卡的固件（简称防病毒卡）插到主机板上，当系统启动后先自动执行，从而取得微处理器的控制权。

（4）管理预防

这是目前最有效的一种预防病毒的措施。目前世界各国大都采用这种方法。一般通过以下三种途径。

法律制度：规定制造计算机病毒是违法行为，对病毒制造者进行法律制裁。

计算机系统管理制度：有系统使用权限的规定、系统支持资料的建立和健全的规定、文件使用的规定、定期清除病毒和更新磁盘的规定等。

教育：这是一种防止计算机病毒的重要策略。通过宣传、教育，使用户了解计算机病毒的常识和危害，尊重知识产权，不随意复制软件，养成定期检查和清除病毒的习惯，杜绝制造病毒的犯罪行为。

第 4 章　数据库技术基础

考纲要求

1.数据库的基本概念，数据库系统的组成。

2.数据模型概念和主要的数据模型。

4.1　知识点

4.1.1　信息、数据与数据处理

信息

信息是现实世界事物的存在方式或运动状态的反映。

数据

数据是描述现实世界事物的符号记录，是指用物理符号记录下来的可以鉴别的信息。

信息和数据的关联

数据是信息的符号表示，或称载体；信息是数据的内涵，是数据的语义解释。信息与数据是密切相关联的。

信息处理的基本环节

源信息往往需要经过信息处理，使之成为有价值的数据。信息处理的基本环节有信息收集、信息表示、信息加工、信息传输、信息存储、信息检索、信息解释。

4.1.2　数据管理技术的发展

数据管理技术的发展经过了三个阶段：人工阶段、文件系统阶段和数据库阶段。

人工管理阶段

20 世纪 50 年代中期以前为人工管理阶段，其数据管理特点为：数据不保存在机器中；没有专用的软件对数据进行管理；数据不共享；数据不具有独立性。

文件系统阶段

20 世纪 50 年代后期至 60 年代中期为文件系统阶段，其数据管理的特点是：数据可长期保存在外存的磁盘上；由文件系统管理数据；数据共享性差，数据冗余度大；数据独

立性差。

数据库阶段

20 世纪 60 年代末开始至今为数据库阶段，其数据管理的特点是：数据结构化；数据共享性高、冗余度小、易扩充；数据独立性高；统一的数据管理和控制。

4.1.3 数据库（DB，DataBase）

数据库是长期储存在计算机内、有组织的、可共享的数据集合。数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度，较高的数据独立性和易扩展性，并可为一定范围内的各种用户共享。

4.1.4 数据库管理系统（DBMS，DataBase Management System）

数据库管理系统是用来科学地组织和存储数据,高效地获取和维护数据一个系统软件，它是位于用户与操作系统之间的一个数据管理软件，它的基本功能有数据定义、数据操纵、数据库的运行管理、数据库的建立和维护等几个方面。

4.1.5 数据库系统（DBS，DataBase System）

数据库系统是指在计算机系统中引入数据库后的系统构成，一般由数据库、操作系统、数据库管理系统（及其工具）、应用系统、数据库管理员和用户构成。数据库管理员是负责数据库的建立、使用和维护等工作的专门人员，数据库管理员又称为 DBA（DataBase Administrator）。

4.1.6 数据模型

数据模型是现实世界数据特征的抽象，它是数据库系统的数学形式框架，是用来描述数据的一组概念和定义，包括描述数据、数据联系、数据操作、数据语义以及数据一致性概念的工具，即：

（1）数据的静态特征，它包括对数据结构和数据间联系的描述。

（2）数据的动态特征，一组定义在数据上的操作，包括操作的含义、操作符、运算规则及其语言等。

（3）数据的完整性约束，这是一组规则，数据库中的数据必须满足这组规则。

数据模型应满足三方面要求：一是能比较真实地模拟现实世界；二是容易为人们所理解；三是便于在计算机上实现。一种数据模型要很好地满足这三方面的要求，在目前尚很困难。在数据库系统中针对不同的使用对象和应用目的，采用不同的数据模型。

根据模型应用的不同目的，可以将这些模型划分为两类，它们分属于两个不同的层次。第一类模型是概念模型，也称信息模型，它是按用户的观点对数据和信息建模。另一类模

型是结构模型，主要包括网状模型、层次模型、关系模型和面向对象模型等，它是按计算机系统的观点对数据建模。

4.1.7 数据模型的要素

任何一种数据模型都是严格定义的概念的集合。这些概念必须能够精确地描述系统的静态特性、动态特性和完整性约束条件。因此，数据模型通常都是由数据结构、数据操作和完整性约束 3 个要素组成。

4.1.8 信息世界中的基本概念

实体（entity）

客观存在并可相互区别的事物称为实体。

属性（attribute）

实体所具有的某一特性称为属性。一个实体可以由若干个属性来刻画。

主码（primary key）

惟一标识实体的属性集称为主码。

域（domain）

属性的取值范围称为该属性的域。

实体型（entity type）

具有相同属性的实体必然具有共同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体，称为实体型。

实体集（entity set）

同型实体的集合称为实体集。

联系（relationship）

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系。两个实体型之间的联系可以分为 3 类：一对一联系（1：1）、一对多联系（1：n）、多对多联系（m：n）。

4.1.9 概念模型——E-R 模型

概念数据模型是指独立于计算机系统的模型，完全不涉及信息在系统中的表示，只是用来描述某个特定组织所关心的信息结构。其中最常用的模型是“ 实体联系模型 ”，简称 E—R 模型。E—R 模型是直接从现实世界中抽象出实体类型及实体间联系，然后用 E—R 图表示的数据模型。E—R 图包括四个基本成分：矩形框、菱形框、椭圆形框和直线。

4.1.10 常用数据结构模型

不同的数据模型具有不同的数据结构形式。目前最常用的数据结构模型有层次模型（hierarchical model）、网状模型（network model）、关系模型（relational model）和面向对象数据模型（object oriented model），其中层次模型和网状模型统称为非关系模型。

4.1.11 数据库系统中模式的概念

在数据模型中有“型”（type）和“值”（value）的概念。型是指对某一类数据的结构和属性的说明，值是型的一个具体赋值。模式（schema）是数据库中全体数据的逻辑结构和特征的描述，它仅仅涉及到型的描述，不涉及到具体的值。模式的一个具体值称为模式的一个实例（instance）。同一个模式可以有很多实例。模式是相对稳定的，而实例是相对变动的，因为数据库中的数据是在不断更新的。模式反映的是数据的结构及其联系，而实例反映的是数据库某一时刻的状态。

4.1.12 数据库系统的三级模式结构

从数据库管理系统的角度看，数据库通常采用模式、外模式、内模式三级结构。模式是对数据库中全部数据的整体逻辑结构的描述，它由若干个概念记录类型组成；外模式是用户与数据库的接口，是用户用到的那部分数据的描述，它由若干个外部记录类型组成；内模式是数据库在物理存储方面的描述，定义所有的内部记录类型、索引和文件的组织方式，以及数据控制方面的细节。

4.1.13 数据库的二层映像

为了能够在内部实现数据库三级模式之间的联系和转换，数据库管理系统在这三级模式之间提供了两层映像：外模式／模式映像和模式／内模式映像。

外模式／模式映像

模式描述的是数据库数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。对应于同一个模式可以有任意多个外模式。对于每一个外模式，数据库系统都有一个外模式／模式映像，它定义该外模式与模式之间的对应关系。这些映像定义通常包含在各自外模式的描述中。

模式／内模式映像

数据库中只有一个模式，也只有一个内模式，所以模式／内模式映像是惟一的，它定义数据库全局逻辑结构与存储结构之间的对应关系。

4.2 重点难点

4.2.1 数据库的基本概念

信息和数据是数据库管理的基本内容和对象。信息是现实世界事物状况的反映，通过加工它可以用一系列数据来表示。如“某商品的进货价为100元，卖出价为120元。”这是一条能够说明商品利润的信息。这条信息可以加工为商品进价(100元)和商品市场价(120元）两条数据，这两条数据同样表达了商品利润为20元的信息。

数据是记录现实世界中各种信息并可以识别的符号，是信息的载体，是信息的具体表现形式。数据与信息是密切关联的。信息是向人们提供关于现实有关事物的知识；数据则是载荷信息的物理符号。二者是不可分离而又有一定区别的两个相关的概念。信息可以用不同的数据来表示，也不随它的数据形式不同而改变。但在一些不是很严格的场合下，对它们又没有做严格的区分，甚至当作同义词来使用，如信息处理与数据处理、信息采集与数据采集等。

4.2.2 数据管理技术的发展

数据管理技术是指对数据的分类、组织、编码、存储、检索和维护的技术。数据管理技术的发展是和计算机技术及其应用的发展联系在一起的，经历了三个阶段：人工管理阶段、文件系统阶段、数据库系统阶段。这三个阶段的特点比较如表 5－1 所示：

表 5－1　数据管理技术发展的三个阶段的比较

比较项目 \ 阶段		人工管理阶段	文件系统阶段	数据库系统阶段
背景	应用背景	科学计算	科学计算、管理	大规模管理
	硬件背景	无直接存取设备	磁盘、磁鼓	大容量磁盘
	软件背景	没有操作系统	有文件系统	有数据库管理系统
	处理方式	批处理	联机实时处理、批处理	联机实时处理、分布处理、批处理
特点	数据的管理者	用户（程序员）	文件系统	数据库管理系统
	数据面向的对象	某一应用程序	某一应用	现实世界中的一定范围
	数据的共享程度	无共享、冗余度极大	共享性差、冗余度大	共享性高、冗余度小
	数据的独立性	不独立，完全依赖于程序	独立性差	具有高度的物理独立性和一定的逻辑独立性
	数据的结构化	无结构	记录内有结构、整体无结构	整体结构化，用数据模型描述

	数据控制能力	应用程序自己控制	应用程序自己控制	由数据库系统提供数据安全性、完整性、并发控制和恢复能力

数据库系统阶段与人工管理和文件系统阶段相比，主要有以下特点：

数据结构化

在文件系统中尽管其记录内部已有了某些结构，但记录之间没有联系，在数据库系统中，数据不再针对某一应用，而是面向全组织，具有整体的结构化。不仅数据是结构化的，而且存取数据的方式也很灵活，可以存取数据库中的某一个数据项、一组数据项、一个记录或一组记录、而在文件系统中，数据的最小存取单位是记录．粒度不能细到数据项。数据库系统实现整体数据的结构化，是数据库的主要特征之一，也是数据库系统与文件系统的本质区别。

数据共享性高、冗余度小、易扩充

系统从整体角度看待和描述数据，数据不再面向某个应用而是面向整个系统，因此数据可以被多个用户、多个应用共享使用。数据共享以大大减少数据冗余，节的存储空间．数据共享还能够避免数据之间的不相容性与不一致性。

所谓数据的不一致性是指同一数据不同拷贝的值不一样。采用人工管理或文件系统管理时，由干数据被重复存储，当不同的应用使用和修改不同的拷贝时就很容易造成数据的不一致。在数据库中数据共享，减少了由于数据冗余造成的不一致现象。

由于数据面向整个系统。是有结构的数据。不仅可以被多个应用共享使用，而且容易增加新的应用，这就使得数据库系统弹性大，易于扩充．可以适应各种用户的要求。可以取整体数据的多种子集用于不同的应用系统，当应用需求改变或增加时．只要重新选取不同的子集或加上一部分数据便可以满足新的需求。

数据独立性高

数据独立性是数据库领域的一个常用术语，包括数据的物理独立性和数据的逻辑独立性。数据的物理独立性是指用户的应用程序与存储在磁盘上的数据库中数据是相互独立的。也就是说，数据在磁盘上的数据库中是怎样存储的是由 DBMS 管理的，用户程序不需要了解，应用程序要处理的只是数据的逻辑结构，这样当数据的物理存储改变时，应用程序不用改变。

数据的逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的，也就是说，数据的逻辑结构改变了，用户程序也可以不变。

数据和程序的独立性，把数据的定义和描述从应用程序中分离出去。此外，数据的存取又有 DBMS 管理，用户不必考虑存取路径等细节，从而简化了应用程序的编制，大大减少了应用程序的维护和修改工作量。

数据由 DBMS 统一管理和控制

数据库是长期存储在计算机内有组织的大量的共享的数据集合，它可以供在一定范围内的各种用户共享，且具有最小的冗余度和较高的数据与程序的独立性，由于多种程序并发地使用数据库，为了能有效、及时地处理数据，并提供安全性和完整性，必须有一个软

件系统—数据库管理系统在数据库建立、运用和维护时对数据库进行统一控制，以保证数据的完整性、安全性，同时在多用户使用数据库时进行并发控制，在发生故障后对系统进行恢复。

4.2.3 数据库系统的用户

数据库系统的用户有数据库管理员、系统分析员和数据库设计人员、应用程序员和最终用户。其中，数据库管理员是负责数据库的建立、使用和维护等工作的专门人员，数据库管理员又称为 DBA（DataBase Administrator）。

数据库管理员具体的职责

（1）决定数据库中的信息内容和结构。数据库中要存放哪些信息，DBA 要参与决策。因此 DBA 必须参加数据库设计的全过程，并与用户、应用程序员、系统分析员密切合作共同协商，搞好数据库设计。

（2）决定数据库的存储结构和存取策略。DBA 要综合各用户的应用要求，与数据库设计人员共同决定数据的存储结构和存取策略，以求获得较高的存取效率和存储空间利用率。

（3）定义数据的安全性要求和完整性约束条件。DBA 的重要职责是保证数据库的安全性和完整性。因此 DBA 负责确定各个用户对数据库的存取权限，数据的保密级别和完整性约束条件。

（4）监控数据库的使用和运行。监视数据库系统的运行情况，及时处理运行过程中出现的问题。比如系统发生各种故障时，数据库会因此遭到不同程度的破坏，DBA 必须在最短时间内将数据库恢复到正确状态，并尽可能不影响或少影响计算机系统其他部分的正常运行。为此，DBA 要定义和实施适当的数据库的备份和恢复策略，如周期性的转储数据，维护日志文件，等等。

（5）数据库的性能改进。在系统运行期间监视系统的空间利用率、处理效率等性能指标，对运行情况进行记录、统计分析、依靠工作实践并根据实际应用环境，不断调整和改进数据库参数的设置。不少数据库产品都提供了对数据库运行情况进行监视和分析的实用程序，DBA 可以使用这些实用程序完成这项工作。

（6）定期对数据库进行重组和重构，以提高系统的性能。因为在数据库系统运行过程中，大量数据不断插人、删除、修改，时间长，会影响系统的性能。当用户的需求增加和改变时，DBA 还要对数据库进行较大的改造，包括修改部分设计，即数据库的重组或重构。

系统分析员和数据库设计人员

系统分析员负责应用系统的需求分析和规范说明，他们要和用户及 DBA 相结合，确定系统的硬软件配置并参与数据库系统的概要设计。

数据库设计人员负责数据库中数据的确定、数据库各级模式的设计。数据库设计人员必须参加用户需求调查和系统分析，然后进行数据库设计。在很多情况下，数据库设计人员就由数据库管理员担任。

应用程序员

应用程序员负责设计和编写应用系统的程序模块，并进行调试和安装。

用户

这里用户是指最终用户（End User）他们通过应用系统的用户接口使用数据库。常用的接口方式有浏览器、菜单驱动、表格操作、图形显示、报表书写，等等。

4.2.4 数据模型

根据模型应用的不同目的，可以将这些模型划分为两类，第一类模型是概念模型，也称信息模型；另一类模型是结构模型。

E-R 模型

E-R 图提供了表示实体型、属性和联系的方法：.

实体型：用矩形表示．矩形框内写明实体名。

属性：用椭圆形表示，并用无向边将其与相应的实体连接起来。

例如：学生实体具有学号、姓名、性别、出生年份、系、入学时间等属性，用 E-R 图表示如图 5－1 所示：

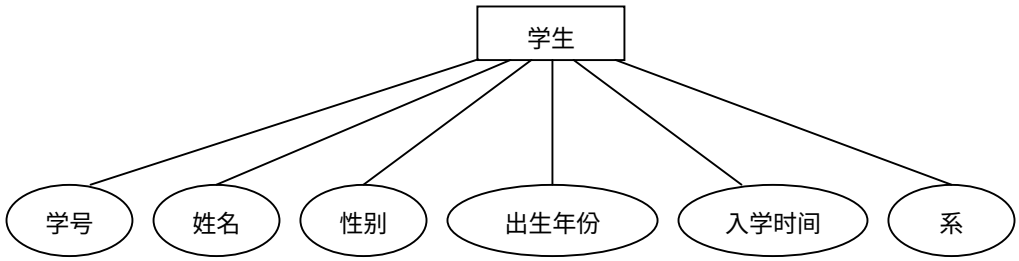


图 5－1　学生实体及属性

联系；用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型或．需要注意的是，如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来。

例如用“ 供应量 ”来描述联系“ 供应 ”的属性。表示某供应商供应了多少数量的零件给某个项目。那么这三个实体及其之间联系的 E－R 图表示可如图 5－2 所示

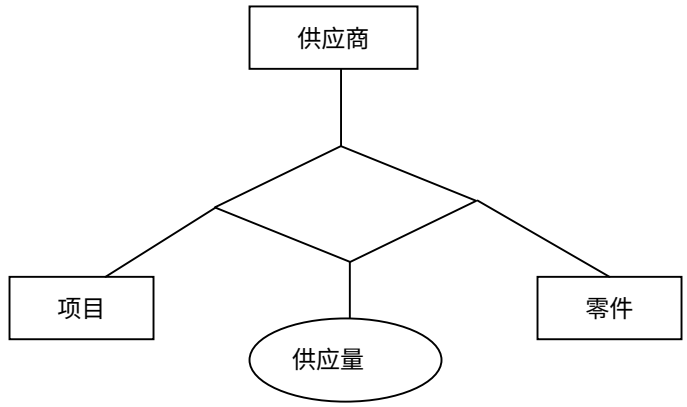


图 5－2　联系的属性

4.1.5　常用的数据结构模型

层次模型(hierarchical mode）

用树型（层次）结构表示实体类型以及实体间的联系是层次模型的主要特征。层次结构是一棵有向村，树的结点是记录类型，根结点只有一个，根结点以外的结点有且只有一个父结点。上一层记录类型和下一层记录类型间的联系是 1：m 联系（包括 1：1 联系）。层次模型的另一个最基本的特点是，任何一个给定的记录值只有按其路径查看时，才能显出它的全部意义，没有一个子记录值能够脱离双亲记录值而独立存在。

网状模型（network model）

用网状结构表示实体类型及实体之间联系的数据模型称为网状模型。在网状模型中，一个子结点可以有多个父结点，在两个结点之间可以有一种或多种联系。网状模型实现实体间 m：n 联系比较容易。记录之间联系是通过指针实现的，因此，数据的联系十分密切。网状模型的数据结构在物理上也易于实现，效率较高，但是编写应用程序较复杂，程序员必须熟悉数据库的逻辑结构。

关系模型（relational model）

用表格形式结构表示实体类型以及实体间联系的模型称为关系模型。关系模型比较简单，容易为初学者接受。关系在用户看来是一个表格，记录是表中的行，属性是表中的列。关系模型与网状、层次模型的最大区别是关系模型用表格的数据而不是通过指针链来表示和实现实体间联系。关系模型的数据结构简单，用户易懂，只需用简单的查询语句就可对数据库进行操作。

面向对象模型（Object-Oriented Model）

现实世界存在着许多含有更复杂数据结构的实际应用领域，例如 CAD 数据、图形数据等，需要有一种数据模型来表达这类信息，随后在人工智能研究中也出现了类似的需要，这种数据模型就是面向对象的数据模型。

对象是现实世界中实体的模型化，并具有唯一的对象标识。对象具有一系列属性，并封装有一系列状态和行为。所有具有相同属性和方法集的对象构成了一个对象类。任何一个对象都是某个对象类的一个实例。

面向对象数据模型比网络、层次、关系数据模型具有更加丰富的表达能力。但正因为面向对象模型的丰富表达能力，模型相对复杂，实现起来较困难。

4.1.6　数据库系统的模式结构

实际应用的数据库在体系结构上通常都具有相同的特征，即采用模式、外模式、内模式三级结构和外模式／模式映像、模式／内模式映像两级映像。其结构如图 5－3 所示：

图 5－3　数据库系统的三级模式结构

模式

模式也称逻辑模式或概念模式，是数据库中全体数据的逻辑结构和特征的描述，它由若干个概念记录类型组成。它是数据库系统模式结构的中间层，既不涉及数据的物理存储细节和硬件环境,也与具体的应用程序以及所使用的应用开发工具及高级程序设计语言(如 FORTRAN、C、COBOL 等）无关。

模式实际上是数据库数据在逻辑层上的视图。一个数据库只有一个模式。数据库模式以某一种数据模型为基础，比如在关系结构模型中，模式就是关系模型的表格。

外模式

外模式是用户与数据库的接口，是用户用到的那部分数据的描述，它由若干个外部记录类型组成。一个数据库可以有多个外模式，外模式通常是模式的子集，外模式处理的数据并不实际存储在数据库中，而仅可以从模式中构造出来，因此，外模式比模式的抽象级别更高。比如在关系模型中，由基本表构造出来的视图就属于一个外模式。

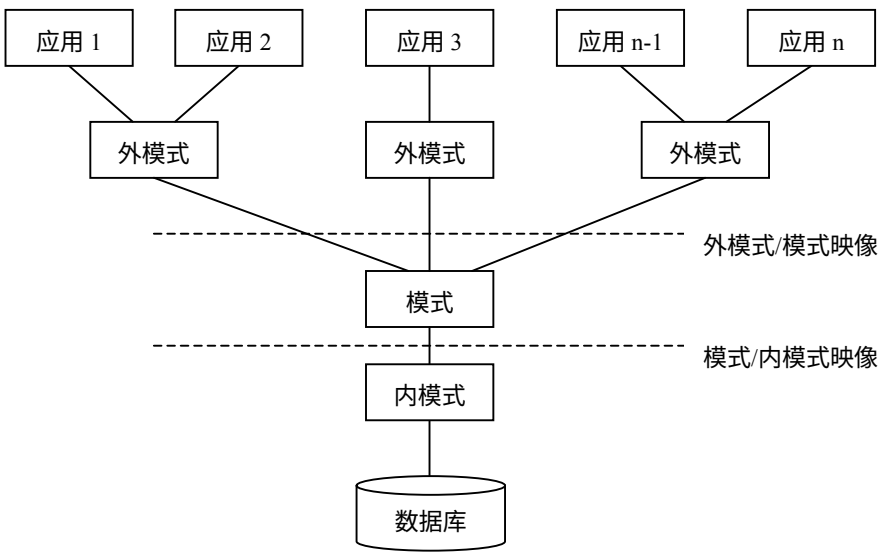
内模式

内模式是数据库在物理存储方面的描述，定义所有的内部记录类型、索引和文件的组织方式，以及数据控制方面的细节。一个数据库只有一个内模式。它是数据物理结构和存储方式的描述，是数据库内部的表示方法。例如，记录的存储方式是顺序存储、按照 B 树结构存储还是按 hash 方法存储；索引按照什么方式组织；数据是否压缩存储，是否加密；数据的存储记录结构有何规定等。

外模式／模式映像

外模式／模式映像存在于外部级和概念级之间，用于定义外模式和概念模式间的对应性，一般在外模式中描述。

当模式改变时（例如增加新的关系、新的属性、改变属性的数据类型等），数据库管理员对各个外模式／模式的映像做相应改变，可以使外模式保诗不变。应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。



模式／内模式映像

模式／内模式映像存在于概念级和内部级之间，用于定义概念模式和内模式间的对应性，一般在内模式中描述。

数据库中的模式/内模式映像是唯一的，它定义了数据库全局逻辑结构与存储结构之间的对应关系，例如，说明逻辑记录和字段在内部是如何表示的。当数据库的存储结构改变了（例如选用了另一种存储结构），由数据库管理员对模式／内模式映像做相应改变，可以使模式保持不变，从而应用程序也不必改变。保证了数据与应用程序的物理独立性，简称数据的物理独立性。

第 5 章　关系数据库系统

考纲要求

1．关系数据模型的基本概念

2．关系操作和关系代数

5.1　知识点

5.1.l　关系数据库系统

关系数据库系统是支持关系数据模型的数据库系统，关系数据库应用数学方法来处理数据库中的数据。著名的关系数据库管理系统有：IBM DB2，Oracle，Ingres，SYBASE，Informix 等。

5.1.2　关系数据模型

关系数据模型由关系数据结构、关系操作集合和关系完整性约束 3 大要素组成。

关系数据结构

关系模型的数据结构单一，在关系模型中，现实世界的实体以及实体间的各种联系均用关系来表示。在用户看来，关系模型中数据的逻辑结构是一张二维表。

关系操作集合

关系模型中常用的关系操作包括：选择（select）、投影（project）、连接（join）、除（divide）、并（union）、交（intersection）、差（difference）等，以及查询（query）操作和增（insert）、删（delete）、改（update）操作两大部分，查询的表达能力是其中最主要的部分。

关系的完整性约束

数据库的数据完整性是指数据库中数据的正确性和相容性，包括两个方面：

（1）与现实世界中应用需求的数据的相容性和正确性。

（2）数据库内数据之间的相容性和正确性。

数据完整性由完整性规则来定义，关系模型的完整性规则是对关系的某种约束条件。

关系模型中可以有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。

5.1.3 关系模型的基本术语

关系（relation）

一个关系对应一个二维表，二维表名就是关系名。

属性（attribute）和值域（domain）

在二维表中的列（字段），称为属性。属性的个数称为关系的元数，列的值称为属性值；属性值的取值范围称为值域。

关系模式（relation schema）

在二维表中的行定义（记录的型），即对关系的描述称为关系模式，一般表示为：

关系名（属性 1，属性 2，…，属性 n）

元组（tuple）

在二维表中的一行（记录的值），称为一个元组。关系模式和元组的集合通称为关系。

分量（component）

元组中的一个属性值。

候选码（candidate key）或候选键

如果在一个关系中，存在多个属性（或属性组合）都能用来唯一标识该关系的元组，这些属性（或属性组合）称为该关系的候选码或候选键。

主码（Primary key）或主键

在一个关系的若干个候选码中指定一个用来唯一标识该关系的元组，这个被指定的候选码称为该关系的主码或主键。

主属性（primary attribute）和非主属性（nonprimary attribute）

关系中包含在任何一个候选码中的属性称为主属性或码属性，不包含在任何一个候选码中的属性称为非主属性或非码属性。

外码（foreign key）或外键

当关系中的某个属性（或属性组）虽然不是该关系的主码或只是主码的一部分，但却是另一个关系的主码时，称该属性（或属性组）为这个关系的外码。

参照关系（referencing relation）与被参照关系（referenced relation）

参照关系也称从关系，被参照关系也称主关系，它们是指以外码相关联的两个关系。以外码作为主码的关系称为参照关系；外码所在的关系称为被参照关系或目标关系。

5.1.4 关系的形式定义

从数学的观点定义关系称为关系的形式定义。有如下两种定义方法：

（1）用集合论的观点定义关系：关系是一个元数为 K 的元组集合，即这个关系有若干个组，每个元组有 K 个属性值（把关系看成一个集合，集合中的元素是元组）。

（2）用值域的概念来定义关系：关系是属性值域笛卡尔积的一个子集。设一个关系的同是 A_1，…，A_n，其对应的值域为 D_1，…，D_n（也可以有相同的），定义 D_1，…，D_n 的笛卡尔积 $D = D_1 X \cdots X D_n = \{ (d_1, \cdots, d_n) | d_i \in D_i, 1 \leq i \leq n \}$。D 中的每一个子集 D_i 称为关系。这里 D 的元素（d_1，…，d_n）就是一个 n 元元组（n-tuple），元素中的每一个值 d_i 称为元组的一个分量。

若 D_i（i=1，2，…，n）为有限集，其基数（Cardinal number）为 m_i（i＝1，2，…，n），则 $D_1 \times D_2 \times \cdots \times D_n$ 的基数 M 为：

$$M = \prod_{i=1}^{n} m_i$$

笛卡尔积可表示为一个二维表，表中的每行对应一个元组，表中的每列对应一个域。

5.1.5 关系数据库对关系的限定

关系模型的数据结构表示为二维表，但不是任意的一个二维表都能表示一个关系。从上面关系的形式定义可见关系数据库对关系是有限定的。这些限定有：每一个属性是不可分解的；每一个关系模式中属性的数据类型以及属性的个数是固定的，并且每个属性必须命名，在同一个关系模式中，属性名必须是不同的；每一个关系仅仅有一种记录类型，即一种关系模式；在关系中元组的顺序（即行序）是无关紧要的；在关系中属性的顺序可任意交换，交换时应连同属性名一起交换才行，否则顺序是重要的；同一个关系中不允许出现完全相同的元组。

5.1.6 关系模型的完整性约束

数据完整性由完整性规则来定义，关系模型的完整性规则是对关系的某种约束条件。关系模型中可以有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。

为了护维数据库中数据的完整性，在对关系数据库执行插入、删除和修改操作时，必须遵循三类完整性规则：实体完整性规则（entity integrity rule）、参照完整性规则（reference integrity rule）和用户定义的完整性规则。

5.1.7 实体完整性规则

实体完整性规则是对关系中的主属性值的约束，它的内容是：

若属性 A 是关系 R 的主属性，则属性 A 不能取空值。

实体完整性规则规定关系的所有主属性都不能取空值，而不仅是主码整体不能取空值。

5.1.8 参照完整性规则

现实世界中的实体之间往往存在某种联系，在关系模型中实体及实体间的联系都是用关系来描述的，这样就自然存在着关系与关系之间的参照（引用）。参照完整性规则内容是：

若属性（或属性组）**F** 是关系 **R** 的外码，它与关系 **S** 的主码 **Ks** 相对应（关系 **R** 和 **S** 不一定是不同的关系），则对于 **R** 中每个元组在 **F** 上的值必须为：或者取空值（**F** 的每个属性值均为空值）；或者等于 **S** 中某个元组的主码值。

5.1.9 用户定义的完整性

关系数据库系统根据现实世界中其应用环境的不同，往往还需要一些另外的约束条件，用户定义的完整性就是针对某一具体应用要求来定义的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。用户定义的完整性通常是定义对关系中除主码与外码属性之外的其他属性取值的约束，即对其他属性的值域的约束。

对属性的值域的约束也称为域完整性规则（domain integrity rule），是指对关系中属性取值的正确性限制，包括数据类型、精度、取值范围、是否允许空值等。

5.1.10 关系代数

关系代数是关系操纵语言的一种传统表示方式，它以集合代数为基础发展起来的，但它的运算对象和运算结果均为关系。关系代数也是一种抽象的查询语言，它通过对关系的运算来表达查询。

任何一种运算都是将一定的运算符作用于一定的运算对象上，得到预期的运算结果。所以运算对象、运算符、运算结果是运算的三大要素。关系代数的运算对象是关系，它将一定的关系代数运算符作用于一定的关系上，得到预期的运算结果亦为关系。

关系代数的运算可分为两类：传统的集合运算和专门的关系运算。

在关系代数运算中，把基本的关系代数运算经过有限次复合的式子称为关系代数运算表达式（简称为代数表达式）。这种表达式的运算结果仍是一个关系，可以用关系代数表达式表示所需要进行的各种数据库查询和更新处理的需求。

5.1.11 传统的集合运算

传统的集合运算是二目运算，包括并、交、差、广义笛卡尔积 4 种运算。

并（union）

设关系 **R** 和关系 **S** 具有相同的目 n（即两个关系都有 n 个属性），且相应的属性取自同一个域，则关系 **R** 与关系 **S** 的并由属于 **R** 或属于 **S** 的元组组成。其结果关系仍为 n 目

关系，记作:

$$R \cup S=\{t/t \in R \vee t \in S\}$$

差（difference）

设关系 **R** 和关系 **S** 具有相同的目 n，且相应的属性取自同一个域，则关系 **R** 与关系 **S** 的差由属于 **R** 而不属于 **S** 的所有元组组成。其结果关系仍为 n 目关系，记作:

$$R-S=\{t/t \in R \wedge t \notin S\}$$

交（intersection）

设关系 **R** 和关系 **S** 具有相同的目 n，且相应的属性取自同一个域，则关系 **R** 与关系 **S** 的交由既属于 **R** 又属于 **S** 的元组组成，其结果关系仍为 n 目关系，记作：

$$R \cap S=\{t/t \in R \cap t \in S\}$$

显然

$$R \cap S=R-(R-S)$$

广义笛卡尔积（extended cartesian product）

设关系 **R** 和 **S** 的元数分别是 r 和 s，定义 **R** 和 **S** 的笛卡尔积是一个(r+s)元元组的集合，每一个元组的前 r 个分量来自 **R** 的一个元组，后 s 个分量来自 **S** 的一个元组。若 **R** 有 k_1 个元组，**S** 有 k_2 个元组，则关系 R 和关系 S 的广义笛卡尔积有 $k_1 \times k_2$ 个元组，记作：

$$R \times S=\{t/t=<t_r,t_s> \wedge t_r \in R \wedge t_s \in S\}$$

5.1.12 专门的关系运算

专门的关系运算主要包括：对单个关系进行垂直分解（投影操作）或水平分解（选择操作和对多个关系的结合（连接操作）等。

选择（ selection）

选择又称为限制（restriction），它是在关系 R 中选择满足给定条件的诸元组，记作：

$$\sigma_F(\mathbf{R})=\{t/t \in \mathbf{R} \wedge \mathbf{F}(t)='\text{真}'\}$$

其中 F 表示选择条件，它是一个逻辑表达式，取逻辑值‘ 真 ’或‘ 假 ’。

逻辑表达式 F 的基本形式为：

$$X_1 \theta Y_1[\phi X_2 \theta Y_2]\ldots$$

θ 表示比较运算符，它可以是＞、≥、＜、≤或≠。X_1 ，Y_1 等是属性名或常量或简单函数。属性名也可以用它的序号来代替。φ表示逻辑运算符，它可是¬、∧或∨。[ ]表示任选项，即[ ]部分可以要也可以不要，...表示上述格式可以重复下去。

投影（projection）

对关系 R 的投影操作，实际上是从 R 中选择出若干属性列组成新的关系，记作：

$$\prod_A(\mathbf{R})=\{t[A]/t \in \mathbf{R}\}$$

其中 **A** 为 **R** 的属性列。

连接（join）

连接也称为 θ 连接，它是从两个关系的笛卡尔积中选取它们的属性间满足一定条件的元组，记作:

$$R \underset{A\theta B}{\triangleright\!\triangleleft} S = \{t_r t_s / t_r \in R \vee t_s \wedge t_s \in S \wedge t_r[A]\theta t_s[B]\}$$

其中，**A** 和 **B** 分别为 **R** 和 **S** 上度数相等且可比的属性。θ 是比较运算符。连接运算从 **R** 和 **S** 的笛卡尔积 **R×S** 中选取在 **A** 属性组（**R** 关系）上的值与在 **B** 属性组（**S** 关系）上值满足比较关系 θ 的元组。

除（division）

给定关系 R（X，Y）和 S（Y，Z），其中 X，Y，Z 为属性组。R 中的 Y 与 S 中的 Y 可以有不同的属性名，但必须出自相同的域集。R 与 S 的除运算得到一个新的关系 P（X），P 与 R 中满足下列条件的元组在 X 属性列上的投影：元组在 X 上分量值 X 的象集 Y，包含 S 在 Y 上投影的集合，记作:

$$R \div S = \left\{ t_r[X] \middle| t_r \in R \wedge \prod_Y (S) \subseteq Y_X \right\}$$

其中 Y_X 为 X 在 R 中的象集，$x=t_r[X]$。

5.2 重点难点

5.2.1 关系模型的数据结构

在关系数据模型（relation model）中，数据结构用单一的二维表结构来表示实体及实体间的联系。

编号	姓 名	性别	民族	籍 贯	出生年月	教研室	学 历	职 称
0101	李新蓓	女	汉	河 北	1963.02	地理	大 学	讲 师
0202	田宏业	男	汉	江 苏	1958.04	物理	大 学	讲 师
0303	王俊华	男	汉	江 苏	1944.08	数学	大 学	讲 师
0404	张国平	男	汉	河 北	1972.03	语文	大 学	讲 师
0505	程进华	男	汉	湖 南	1972.09	化学	大 专	助 讲
0606	肖婷婷	女	汉	河 南	1967.07	英语	大 学	讲 师

编号	教研室	办公室	电话
01	地理	209	337

02	物理	301	362
03	数学	202	316
04	语文	303	323
05	化学	205	359
06	英语	206	380

图 5－1　关系数据模型结构示例

一个关系对应一个二维表，二维表名就是关系名。图 6－1 包含两个表，也即两个关系：教师登记表关系和教研室信息表关系。下面简单分析一下这两个关系的数据结构。

属性和值域

在二维表中的列（字段），称为属性。属性的个数称为关系的元数，列的值称为属性值；属性值的取值范围称为值域。

图 5－1 中教师登记表关系的属性有编号、姓名、性别、民族、籍贯、出生年月、教研室、学历、职称，所以元数是 9。出生年月属性的值域是 1944 年 8 月至 1972 年 9 月。系信息表关系的属性有编号、教研室、办公室、电话，所以元数是 4。

关系模式

在二维表中的行定义（记录的型），即对关系的描述称为关系模式，一般表示为：

关系名（属性 1，属性 2，…，属性 n）

元组

在二维表中的一行（记录的值），称为一个元组。关系模式和元组的集合通称为关系。

在学生登记表关系中的一个元组是：

（0101，李新蓓，女，汉，河北，1963.02，地理，大学，讲师）

教研室信息表关系中一个元组是：

（01，地理，209，337）

分量

元组中的一个属性值。例如，在学生登记表关系中元组（0101，李新蓓，女，汉，河北，1963.02，地理，大学，讲师）的每一个属性值：0101，李新蓓，女，汉，河北，1963.02，地理，大学，讲师，都是它的分量。

候选码与主码

如果在一个关系中，存在多个属性（或属性组合）都能用来唯一标识该关系的元组，这些属性或属性组合）都称为该关系的候选码或候选键。例如，在教师登记表关系中，如果姓名不允许重名时，编号和姓名都是候选码。

在一个关系的若干个候选码中指定一个用来唯一标识该关系的元组，这个被指定的候选码称为该关系的主码或主键。在教师登记表关系中，编号和姓名都是候选码，若选中编号作为唯一标识，那么，编号就是学生登记表关系的主码。候选码与主码是进行关系检索

和操作的重要参数。

主属性和非主属性

关系中包含在任何一个候选码中的属性称为主属性或码属性，不包含在任何一个候选码中的属性称为非主属性或非码属性。在教师登记表关系中，编号和姓名是主属性，其他属性是非主属性。

外码或外键

当关系中的某个属性（或属性组）虽然不是该关系的主码或只是主码的一部分，但却是另一个关系的主码时，称该属性（或属性组）为这个关系的外码。

例如，如果图 5－1 中的教研室信息表关系的主码是“ 教研室 ”，那么，在教师登记表关系中的“ 教研室 ”就是外码，因为它是另一个关系的主码。

参照关系与被参照关系

参照关系也称从关系，被参照关系也称主关系，它们是指以外码相关联的两个关系。以外码作为主码的关系称为参照关系；外码所在的关系称为被参照关系或目标关系。被参照关系与参照关系是通过外码相联系的，这种联系通常是 1:n 的联系。

例如，图 5-1 中的教研室信息表关系是被参照关系，而教师登记表关系是参照关系，它们通过外码“ 教研室 ”相联系。

5.2.2　关系数据库对关系的限定

关系模型的数据结构表示为二维表，但不是任意的一个二维表都能表示一个关系。从上面关系的形式定义可见关系数据库对关系是有限定的。

首先每一个属性必须是不可分解的。

例如表 5－1 中的关系就不符合要求。该表反映的是某校全国研究生统考通过初试的学生的情况，属性成绩被分为外语、数学等多项，这相当于大表中还有一张小表（关于成绩的表），所以是不符合限定标准的。

表 5－1　不满足关系数据库限定的关系

考生姓名	学科、专业名称	毕业院校	考生所在单位	成绩				
				政治	外语	数学	专业	总分
刘义	计算机应用技术	上海交通大学	浙江大学计算机学院	75	59	98	117	349.0
陆飞	计算机应用技术	浙江大学	浙江大学计算机学院	65	73	96	126	360.0
张峰	计算机应用技术	浙江大学	浙江大学计算机学院	69	65	117	110	361.0
陈彦铮	计算机应用技术	浙江大学	浙江大学计算机学院	57	66	93	120	336.0
李丹娜	计算机应用技术	浙江大学	浙江大学计算机学院	61	74	121	136	392.0
王雪盈	计算机应用技术	四川大学	四川大学计算机学院	64	45	107	140	356.0
张加星	计算机应用技术	东南大学	东南大学计算机学院	63	65	105	142	375.0