

软件开发教研室

# 软件设计师培训

主讲：熊焕亮

主讲：熊焕亮

# 软件设计师考试说明

## ❖ 考试要求：

- (1) 掌握数据表示、算术和逻辑运算；
- (2) 掌握相关的应用数学、离散数学的基础知识；
- (3) 掌握计算机体系结构以及各主要部件的性能和基本工作原理；
- (4) 掌握操作系统、程序设计语言的基础知识，了解编译程序的基本知识；
- (5) 熟练掌握常用数据结构和常用算法；
- (6) 熟悉数据库、网络和多媒体的基础知识；

- (7) 掌握C程序设计语言，以及C++、Java、Visual Basic、Visual C++中的一种程序设计语言；
- (8) 熟悉软件工程、软件过程改进和软件开发项目管理的基础知识；
- (9) 熟练掌握软件设计的方法和技术；
- (10) 掌握常用信息技术标准、安全性，以及有关法律、法规的基本知识；
- (11) 了解信息化、计算机应用的基础知识；
- (12) 正确阅读和理解计算机领域的英文资料。

## ❖ 软件设计师的要求

通过本考试的合格人员能根据软件开发项目管理 and 软件工程的要求，按照系统总体设计规格说明书进行软件设计，编写程序设计规格说明书等相应的文档；组织和指导程序员编写、调试程序，并对软件进行优化和集成测试，开发出符合系统总体设计要求的高质量软件；具有工程师的实际工作能力和业务水平。

## ❖ 考试设置的科目包括:

(1) 上午考试主要考查计算机理论、软件设计理论的基础知识, 考试时间为150分钟, 笔试, 共75道选择题, 最后5道为专业英语题;

(2) 下午考试主要考查软件设计能力, 考试时间为150分钟, 笔试, 一般有7道题, 1~4为必做题, 通常分别为数据流图、数据库设计、UML图, 程序流程图(或C语言设计), 5~7道为选做题(三选一), 通常分别为C程序设计、C++程序设计、JAVA程序设计, 每次考试可能有改变。

# 内 容

- 系统开发与运行知识
- 数据库技术基础知识
- 数据结构与算法知识
- 操作系统知识
- 计算机网络基础知识
- 程序语言基础
- 面向对象技术
- 多媒体基础知识
- 软件知识产权和标准化
- 专业英语

# 1. 系统开发与运行知识

大纲要求：

## 1.1 软件工程基础知识

- 软件生命周期与软件生命周期模型
- 软件开发方法
- 软件开发项目管理
- 软件开发工具与软件开发环境
- 软件过程管理
- 软件质量管理和质量保证

## 1.2 系统分析基础知识

- 系统分析步骤
- 结构化分析方法——数据流图

### 1.3 系统设计基础知识

- 概要设计与详细设计的基本任务
- 系统设计的基本原理
- 系统模块结构设计
- 结构化设计方法
- 面向数据结构的设计方法
- 系统详细设计

### 1.4 系统实施基础知识

- 系统实施的基本内容



- 程序设计的基础模块
- 系统测试
- 系统转换

## 1.5 系统运行和维护基础知识

- 系统可维护性概念
- 系统评价的概念和类型

# 历年考试对本章内容的考查情况

考次	2004		2005		2006		2007		2008		2009
	5月	11月	5月	11月	5月	11月	5月	11月	5月	11月	5月
分值	21	22	36	11	24	41	53	39	26	27	26
比重	14%	14.7%	24%	7.3%	16%	27.3%	35.3%	26%	17.3%	18%	17.3%

本章的重点程度：★★★★★

## 1.1 软件工程基础知识

### ● 软件生命周期

软件的生存期划分为制定计划、需求分析、设计、编程实现、测试、运行维护等几个阶段，称为软件生命周期。

- **制定计划:** 确定待开发软件系统的总目标, 对其进行可行性分析, 并对资源分配、进度安排等做出合理的计划。  
参与者: 用户、项目负责人、系统分析师。  
产生的文档: 可行性分析报告、项目计划书。
- **需求分析:** 确定待开发软件系统的功能、性能、数据、界面等要求, 从而确定系统的逻辑模型。  
参与者: 用户、项目负责人、系统分析师。  
产生的文档: 需求规格说明书。

- **软件设计：**分为概要设计和详细设计。概要设计是对模块的分解，确定软件的结构，模块的功能和模块间的接口，以及全局数据结构的设计。详细设计是设计每个模块的实现细节和局部设局结构。  
参与者：概要设计阶段参加人员是系统分析师和软件设计师，详细设计阶段的参加人员是软件设计师和程序员。  
产生的文档：概要设计说明书、详细设计说明书。

- **编码：**用某种程序语言为每个模块编写程序。  
参与者：软件设计师和程序员。  
产生的文档：源程序清单。
- **测试：**保证软件质量的重要手段，主要方式是在设计测试用例的基础上检验软件的各个组成部分。  
参与者：另一个部门的软件设计师或系统分析师。  
产生的文档：软件测试计划、软件设计报告。
- **运行与维护：**对已交付的软件投入正式使用，并对软件运行中可能由于各方面的原因出现的问题进行后期修改。

- **【软件设计师考试2009年5月上午试题33】**

在开发信息系统时，用于系统开发人员与项目管理人员沟通的主要文档是 \_\_\_\_\_。

A. 系统开发合同

B. 系统设计说明

C. 系统开发计划

D. 系统测试报告

- **【软件设计师考试2008年11月上午试题33】**

系统测试人员与系统开发人员需要通过文档进行沟通，系统测试人员应根据一系列文档对系统进行测试，然后将工作结果撰写成 \_\_\_\_\_，交给系统开发人员。

A. 系统开发合同

B. 系统设计说明书

C. 测试计划

D. 系统测试报告



- 【软件设计师考试2007年5月上午试题18】

通常在软件的\_\_\_\_\_活动中无需用户参与。

A. 需求分析

B. 维护

C. 编码

D. 测试

- 【软件设计师考试2006年11月上午试题29】

\_\_\_C\_\_\_详细描述软件的功能、性能和用户界面，以使用户了解如何使用软件。

A. 概要设计说明书

B. 详细设计说明书计

C. 用户手册

D. 用户需求说明书



- 【软件设计师考试2004年11月上午试题10】

下述任务中，不属于软件工程需求分析阶段的是\_\_\_\_\_。

- A. 分析软件系统的数据要求.
- B. 确定软件系统的功能需求.
- C. 确定软件系统的性能要求.
- D. 确定软件系统的运行平台.

## ● 软件开发模型


为了指导软件的开发，用不同的方法将软件生存周期中的所有开发活动组织起来，形成不同的软件开发模型，它描述软件开发过程总各种活动如何执行的模型。常见的软件开发模型有瀑布模型、演化模型、螺旋模型、喷泉模型。

**瀑布模型（Waterfall Model）：**严格遵循软件生命周期各阶段的固定顺序——计划、分析、设计、编程、测试和维护，上一个阶段完成后才能进入下一个阶段，整个模型像一个飞流直下的瀑布。

瀑布模型缺乏灵活性，无法通过开发活动澄清本来不够明确的活动。因此，当用户需求比较明确时才使用此模型。

**演化模型（Evolutionary Model）：**也称为快速原型模型，由于开发软件在开始时对软件需求的认识是模糊的，因此，很难一次性开发成功。演化模型就是在获得一组基本的用户需求后，快速构造出该软件的一个初始可运行版本，这个初始的软件称为**原型**，实现客户或未来的用户与系统的交互，用户或客户对原型进行评价，进一步细化待开发软件的需求。通过逐步调整原型，最终可得到另用户满意的软件产品。

显然，快速原型方法可以克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险，具有显著的效果。



**螺旋模型（Spiral Model）：**将瀑布模型和演化模型相结合，提出了螺旋模型，综合了瀑布模型和演化模型的优点，并增加了风险分析。包含4个方面活动：

- 制定计划：确定软件的目标，选定实施方案，弄清项目开发的限制条件。
- 风险分析：分析所选的方案，识别风险，验证阶段性产品。
- 实施工程：实施软件开发，验证阶段性产品。
- 客户评价：评价开发工作，提出修改意见。

**喷泉模型（Water Fountain Model）：**主要用于描述面向对象的开发过程。喷泉一词体现了面向对象开发过程的迭代和无间隙特征。即允许开发活动交叉、迭代地进行。

**迭代：**模型中的开发活动常常需要重复多次，在迭代过程中不断完善软件系统。

**无间隙：**指在开发活动（如分析、设计、编码）之间不存在明显的边界。

**V模型（V Model）**：是瀑布模型的变形，与传统瀑布模型相比，该模型强调测试过程应如何与分析、设计等过程相关联。

**增量模型（Incremental Model）**：增量模型在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品。整个产品被分解成若干个构件，开发人员逐个构件地交付产品，这样做的好处是软件开发可以较好地适应变化，客户可以不断地看到所开发的软件，从而降低开发风险。

**构件**：是由多种相互作用的模块所形成的提供特定功能的代码片段构成。

- 【软件设计师考试2006年11月上旬试题15、16】

常见的软件开发模型有瀑布模型、演化模型、螺旋模型、喷泉模型等。其中 A 模型适用于需求明确或很少变更的项目，D 模型主要用来描述面向对象的软件开发过程。

A. 瀑布模型    B. 演化模型    C. 螺旋模型    D. 喷泉模型

- 【软件设计师考试2005年11月上旬试题6】

在开发一个系统时，如果用户对系统的目标是不很清楚，难以定义需求，这时最好使用 A。

A. 原型法    B. 瀑布模型    C. V-模型    D. 螺旋模型



采用瀑布模型进行系统开发的过程中，每个阶段都会产生不同的文档。以下关于产生这些文档的描述中，正确的是\_\_\_\_\_。

- A. 外部设计评审报告在概要设计阶段产生。
- B. 集成测评计划在程序设计阶段产生。
- C. 系统计划和需求说明在详细设计阶段产生。
- D. 在进行编码的同时，独立的设计单元测试计划。

## ● 软件开发方法

软件开发方法是一种使用早已定义好的技术集及符号表示习惯来组织软件生产的过程。

- 结构化的方法
- Jackson方法
- 面向对象开发方法

- 结构化方法

结构化的方法包括结构化的分析、结构化的设计、结构化的程序设计构成，是一种面向数据流的开发方法。

结构化方法总的指导思想是自顶向下、逐层分解，基本原则是功能的分解与抽象。

- **Jackson方法:**

是面向数据结构的开发方法，包括JSP  
( Jackson Structure programming ) 和JSD  
( Jackson System Development )

- 面向对象开发方法

面向对象方法是以对象为最基本的元素，对象也是分析问题和解决问题的核心。开发方法包括面向对象分析、面向对象设计、面向对象实现。

面向对象开发方法有Booch方法、Coad方法和OMT方法等。为了统一各种面向对象方法的术语、概念和模型，1997年推出了统一建模语言——UML（Unified Modeling Language），它有标准的建模语言，通过统一的语义和符号表示。

- 【软件设计师考试2008年5月上旬试题15】

\_\_\_\_\_是一种面向数据流的开发方法，其基本思想是软件功能的分解和抽象。

- A. 结构化开发方法
- B. Jackson系统开发方法
- C. Booch方法
- D. UML（统一建模语言）

软件开发中的瀑布模型典型地刻画了软件生存周期的阶段划分，与其最相适应的软件开发方法是\_\_\_\_\_。

- A. 构件化方法
- B. 结构化方法
- C. 面向对象方法
- D. 快速原型方法

## ● 软件项目管理

**软件项目管理**是软件在开发的过程中对软件开发项目的工作范围、可能遇到的风险、需要的资源、要实现的任务、经历的里程碑、花费的工作量（成本）、以及进度的安排进行管理。软件项目管理包括：

- 成本估计
- 风险分析
- 进度管理
- 人员管理

成本估算方法：有自顶向下估算法、专家估算法……。

成本估算模型：**IBM**模型、**Putnam**模型、**COCOMO**模型。





- 风险分析

风险分析包括4种风险评估活动：

**风险识别**—建立风险概念的尺度。试图系统化确定对项目计划（估算、进度、资源分配）的威胁。

**风险预测**—描述风险引起的后果。确定风险发生的可能性或概率以及如果风险发生了所产生的后果。

**风险评估**—估计风险影响的大小。

**风险控制**—确定风险估计的正确性。辅助项目组建立处理风险的策略。

- 【软件设计师考试2008年11月上午试题19】

在软件工程环境中进行风险识别时，常见的、已知的及可预测的风险类包括产品规模、商业影响等，与开发工具的可用性及质量相关的风险是\_\_\_\_\_。

A. 客户特性

B. 过程定义

C. 开发环境

D. 构建技术

风险识别的一个方法是建立风险条目检查表。该风险表可以用于识别风险，并使得人们集中来识别下列常见的、已知的及可预测的风险：

产品规模——与要建造或修改的软件的总体规模相关的风险；

商业影响——与管理或市场所诸的约束相关的风险；

客户特性——与客户的素质以及开发者和客户定期通信的能力相关的风险；

过程定义——与软件过程被定义的程度以及它们被开发组织所遵守的程序相关的风险；

开发环境——与用以构建产品的工具的可用性及质量相关的风险。

构建的技术——与待开发软件的复杂性及系统所包含技术的“新奇性”相关的风险；

人员数目及经验——与参与工作的软件工程师的总体技术水平及项目经验相关的风险。

- 【软件设计师考试2006年5月上旬试题18】

在软件项目开发过程中，评估软件项目风险时，\_\_\_\_\_与风险无关。

- A. 高级管理人员是否正式承诺支持该项目。
- B. 开发人员和用户是否充分理解系统的需求。
- C. 最终用户是否同意部署已开发的系统。
- D. 开发需要的资金是否能按时到位。

- 进度管理

进度管理就是对软件开发进度的合理安排，它是如期完成软件项目的重要保证，也是合理分配资源的重要保证。

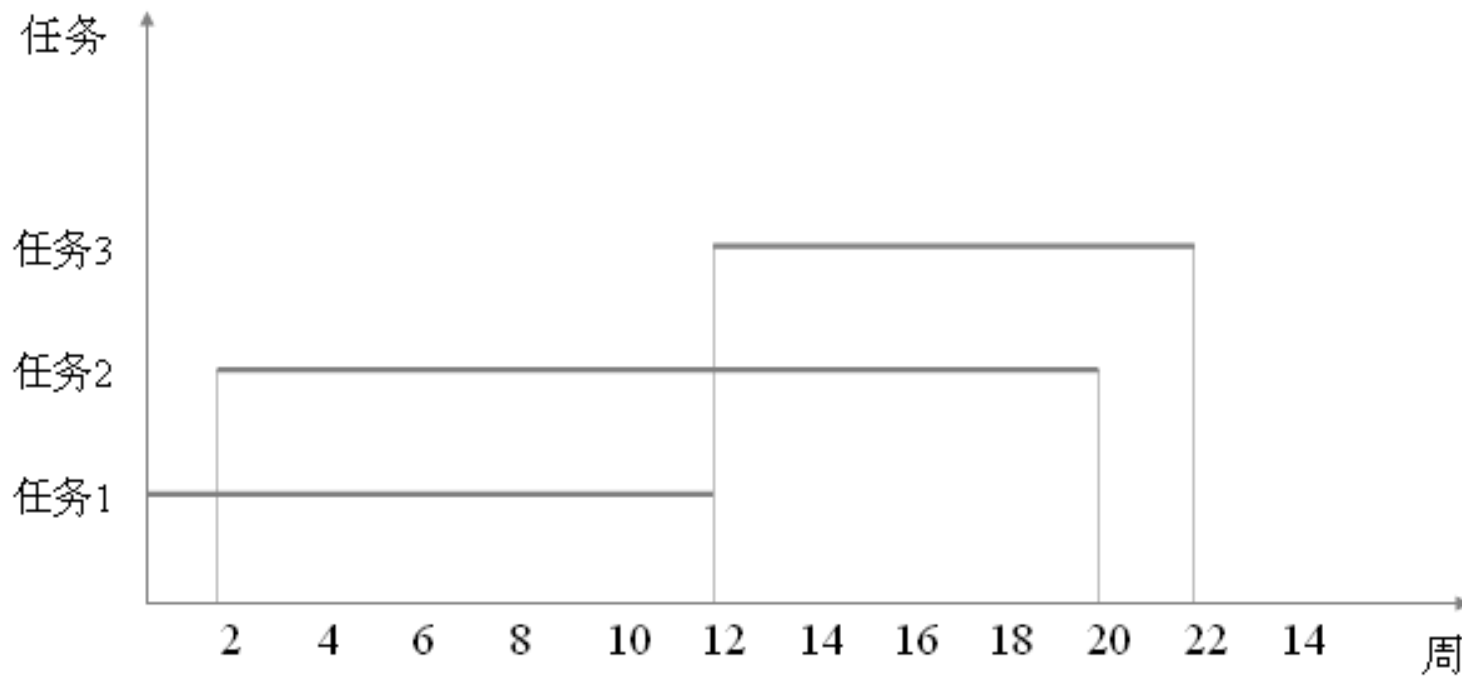
进度安排的常用描述方法有：

- 甘特图（Gantt图）
- 计划评审技术图（PERT 图）

## ➤甘特图（Gantt图）

Gantt图的横坐标表示时间，纵坐标表示任务，图中的水平线段表示对一个任务的进度安排，线段的起点和终点对应于横坐标上的时间，分别表示任务的开始和结束时间，线段的长度表示完成该任务所需的时间。

## Gantt图实例





Gantt图能清晰地描述每个任务从何开始，到何结束以及各个任务之间的并行性，甘特图优点是标明了各任务进度，能动态地反映项目开发进展；但是它不能清晰地反映出各任务之间的依赖关系，难以确定整个项目的关键所在，也不能反映计划中的潜力的部分。

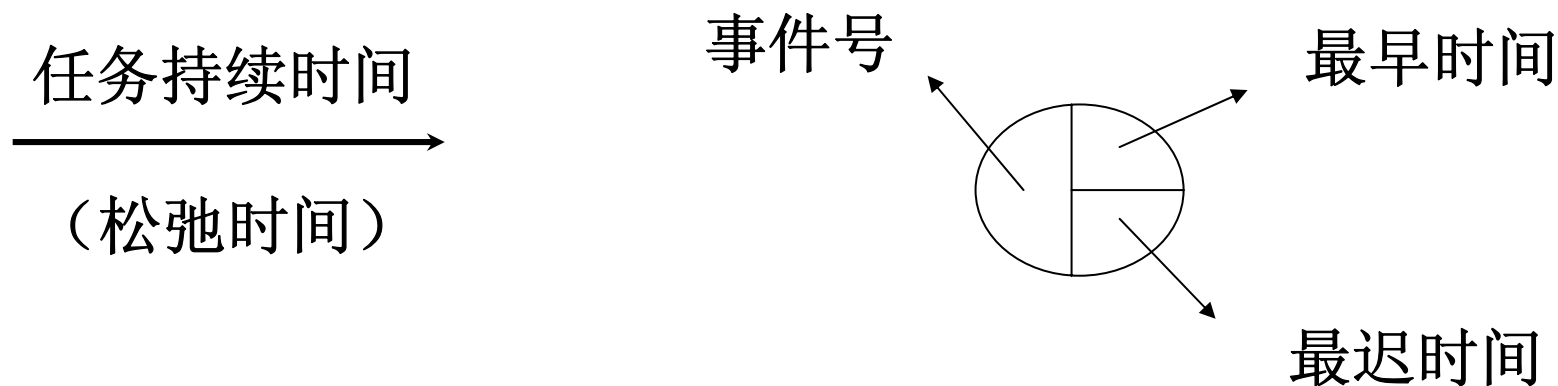
## ➤计划评审技术图（PERT 图）

即计划评审技术图，在实施一个工程计划时，若将整个工程分成若干工序，有些工序可以同时实施，有些工序必须在完成另一些工序之后才能实施，工序之间的次序关系可用有向图表示，这种图称为PERT图。

PERT图中有两个基本元素：

**箭头**：表示任务。

**节点**：表示流入节点的任务的结束，并开始流出节点的任务，称之为事件，即一个时间点。



**最早时刻:**表示此时刻之前从该事件出发的任务不可能开始。

**最迟时刻:**表示从该事件出发的任务必须在此时刻开始, 否则整个工程就不能如期完成。

**松弛时间:**表示在不影响整个工期的前提下, 完成该任务有多少机动余地。

**松弛时间为0的任务**就是完成整个工程的关键路径。

- 【软件设计师考试2009年5月上午试题17、18】

某项目主要由A~I任务构成，其计划图（如下图所示）展示了各任务之间的前后关系以及每个任务的工期（单位：天），该项目的关键路径（1）C。在不延误项目总工期的情况下，任务A最多可以推迟开始的时间是（2）B天

（1）A.  $A \rightarrow G \rightarrow I$

B.  $A \rightarrow D \rightarrow F \rightarrow H \rightarrow I$

C.  $B \rightarrow E \rightarrow G \rightarrow I$

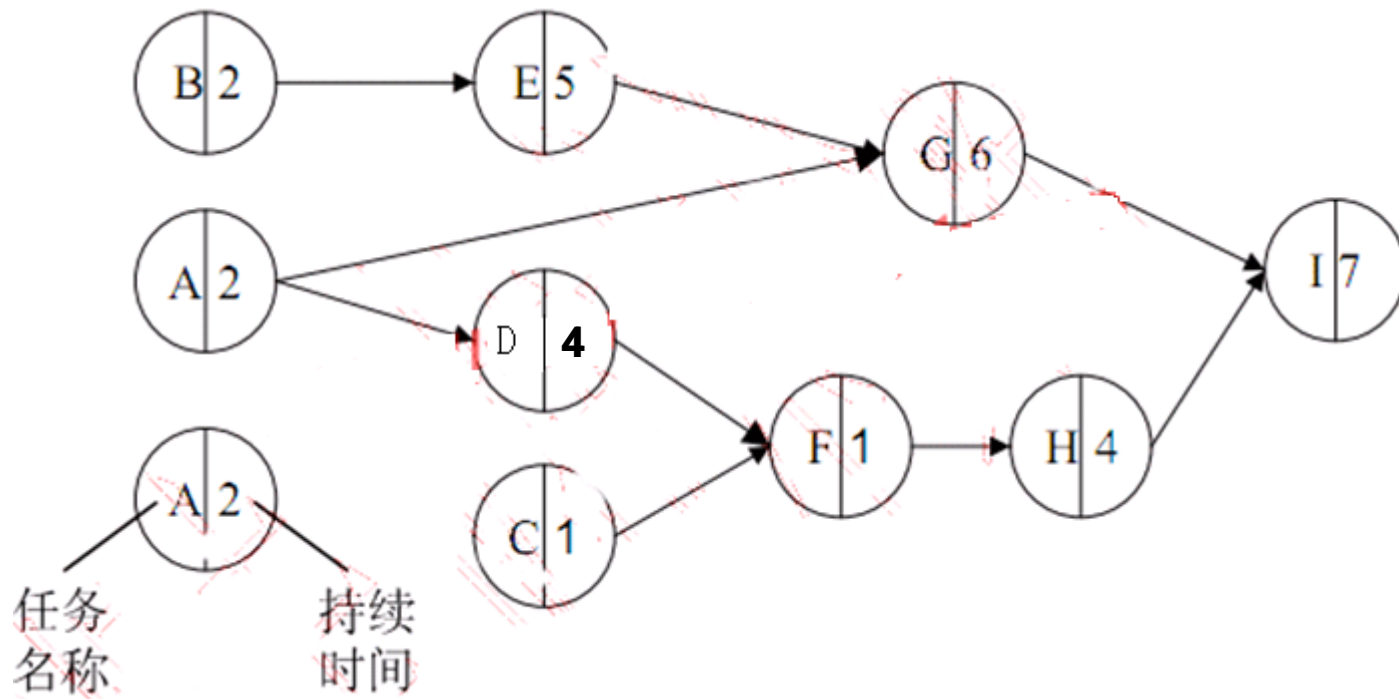
D.  $C \rightarrow F \rightarrow H \rightarrow I$

（2）A. 0

B. 2

C. 5

D. 7



- $TE_A=2 \quad TE_B=2 \quad TE_C=1 \quad TE_D=TE_A+4=2+4=6$

$$TE_E=TE_B+5=2+5=7$$

$$TE_F=\max\{TE_C+1, TE_D+1\}=\max\{1+1, 6+1\}=7$$

$$TE_G=\max\{TE_A+6, TE_E+6\}=\max\{2+6, 7+6\}=13$$

$$TE_H=TE_F+4=7+4=11$$

$$TE_I=\max\{TE_G+7, TE_H+7\}=\max\{13+7, 11+7\}=20$$

- $TL_I=TE_I=20$

$$TL_H=TL_I-7=20-7=13 \quad TL_G=TL_I-7=20-7=13$$

$$TL_F=TL_H-4=13-4=9 \quad TL_E=TL_G-6=13-6=7$$

$$TL_D=TL_F-1=9-1=8 \quad TL_C=TL_F-1=9-1=8$$

$$TL_B=TL_E-5=7-5=2 \quad TL_A=\min\{TL_D-4, TL_G-6\}=\min\{8-4, 13-6\}=4$$



$TE_i$ 表示第*i*个任务的最早完成时间。

$TL_i$ 表示第*i*个任务的最晚完成时间。

$TE_i = \max \{ \text{第 } i \text{ 个任务的前驱任务最早完成时间} + \text{第 } i \text{ 个任务的持续时间} \}$

$TL_i = \min \{ \text{第 } i \text{ 个任务的后驱任务最晚完成时间} - \text{第 } i \text{ 个任务的后驱任务持续时间} \}$

$TE_{\text{最终任务}} = TL_{\text{最终任务}}$

松弛时间 =  $TL_i - TE_i$

松弛时间为0的任务路径为关键路径



- 【软件设计师考试2008年11月上午试题17、18】

若一个项目由9个主要任务构成，其计划图(如下图所示)展示了任务之间的前后关系以及每个任务所需天数，该项目的关键路径是(1) A，完成项目所需的最短时间是(2) D 天。

(1) A.  $A \rightarrow B \rightarrow C \rightarrow D \rightarrow I$

B.  $A \rightarrow B \rightarrow C \rightarrow E \rightarrow I$

C.  $A \rightarrow B \rightarrow C \rightarrow F \rightarrow G \rightarrow I$

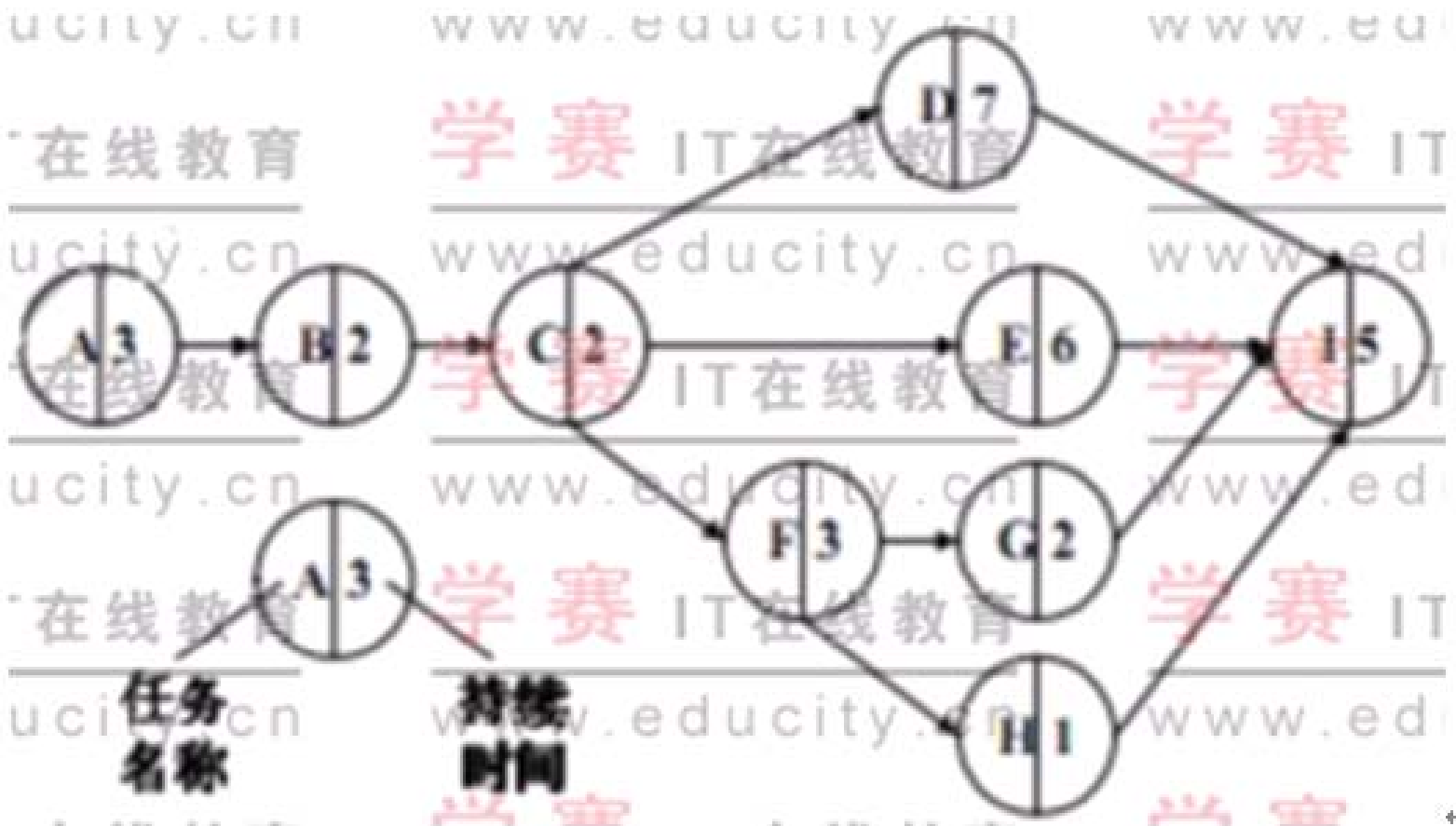
D.  $A \rightarrow B \rightarrow C \rightarrow F \rightarrow H \rightarrow I$

(2) A. 16

B. 17

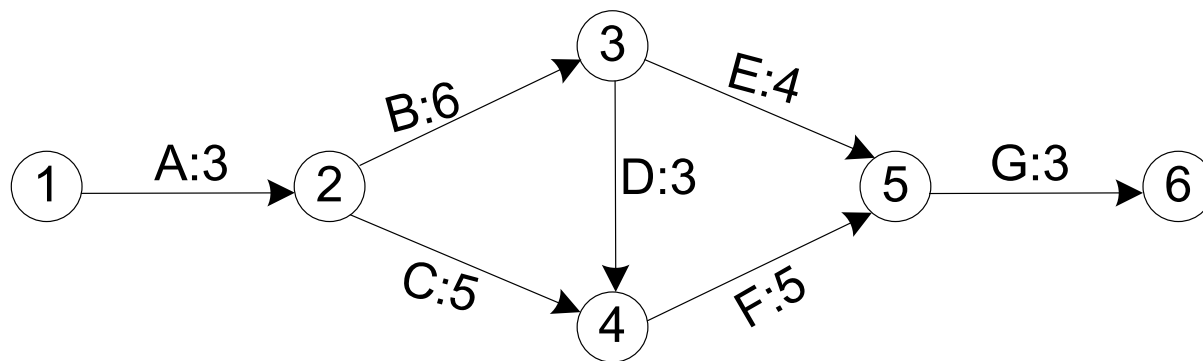
C. 18

D. 19



- 【软件设计师考试2006年5月上午试题27】

某工程计划图如下图所示，弧上的标记为作业编码及其需要的完成时间（天），作业E最迟应在第 D 天开始。



A. 7

B. 9

C. 12

D. 13

## ➤ 关键路径法（CPM 图）

CPM和PERT的区别是： CPM是以经验数据为基础来确定各项工作的时间，而PERT则把各项工作的时间作为随机变量来处理。所以，前者往往被称为肯定型网络计划技术，而后者往往被称为非肯定型网络计划技术。前者是以缩短时间、提高投资效益为目的，而后者则能指出缩短时间、节约费用的关键所在。

- 【软件设计师考试2007年11月上午试题29、30】

在软件开发中，  D  不能用来描述项目开发的进度安排。在其他三种图中，可用  A  动态地反映项目开发进展情况。

A. 甘特图

B. PERT图

C. PERT/CPM图

D. 鱼骨图

鱼骨图用于质量管理

- 【软件设计师考试2006年5月上午试题15】

在软件项目管理中可以使用各种图形工具来辅助决策，下面对Gantt图的描述中，不正确的是\_\_\_\_\_。

- A. Gantt图表现各个活动的持续时间。
- B. Gantt图表现了各个活动的起始时间。
- C. Gantt图反映了各个活动之间的依赖关系。
- D. Gantt图表现了完成各个活动的进度。

- 【软件设计师考试2005年11月上午试题15】

在软件项目管理中可以使用各种图形工具来辅助决策，下面对Gantt图的描述不正确的是\_\_\_\_\_。

A. Gantt 图表现各个活动的顺序和它们之间的因果关系。

B. Gantt 图表现哪些活动可以并行进行。

C. Gantt 图表现了各个活动的起始时间。

D. Gantt 图表现了各个活动完成的进度。

## ● 软件过程管理

**软件过程**—人们用于开发和维护软件及其相关产品（项目计划、设计文档、代码、测试用例、用户手册等）的一系列活动、包括软件工程活动和软件管理活动，其中必然涉及相关的方法和技术。

- 软件能力成熟度模型 (CMM)
- 统一软件开发过程 (RUP)
- 极限编程 (XP)



- 软件能力成熟度模型 (Capability Maturity Model, 简称CMM)

**软件过程能力**—描述（开发组织和项目组）通过遵循其软件过程能够实现预期结果的程度。

**软件能力成熟度**— 一个特定软件过程被明确和有效地定义、管理、测量及控制的程度。成熟度可指明一个软件开发组织软件过程的能力的增长潜力。



CMM模型将软件过程能力成熟度划分为5个级别：

- (1) **初始级**：软件过程是无序的，有时甚至是混乱的，对过程几乎没有定义，成功取决于个人努力。
- (2) **可重复级**：建立了基本的项目管理过程来跟踪费用、进度和功能特性。制定了必要的过程纪律，能重复早先类似应用项目取得的成功。
- (3) **已定义级**：已将软件管理和工程两方面的过程文档化、标准化，并综合成该组织的标准软件过程。所有项目均使用经批准化、剪裁的标准软件过程来开发和维护软件。

(4) **已管理级**：收集对软件过程和产品质量的详细度量，对软件过程和产品都有定量的理解和控制。

(5) **优化级**：过程的量化反馈和先进的思想，新技术促使过程不断改进。

每一个成熟度等级为过程改进达到下一个等级提供一个基础，当前一个等级没有达到时，不能进入下一个等级。

- 【软件设计师考试2009年11月上旬试题29】

软件能力成熟度模型（CMM）将软件能力成熟度自低到高依次划分为 5 级。目前，达到CMM第3级（已定义级）是许多组织努力的目标，该级的核心是\_\_\_\_\_。

- A. 建立基本的项目管理和实践来跟踪项目费用、进度和功能特性.
- B. 使用标准开发过程（或方法论）构建（或集成）系统.
- C. 管理层寻求更主动地应对系统的开发问题.
- D. 连续地监督和改进标准化的系统开发过程.

- 【软件设计师考试2008年11月上午试题29】

软件能力成熟度模型 (CMM) 将软件能力成熟度自低到高依次划分为初始级、可重复级、定义级、管理级和优化级。其中\_\_\_\_\_对软件过程和产品都有定量的理解与控制。

- A. 可重复级和定义级      B. 定义级和管理级  
C. 管理级和优化级      D. 定义级、管理级和优化级

- 【软件设计师考试2006年11月上午试题29】

软件能力成熟模型（CMM）是目前国际上最流行、最实用的软件生产过程标准和软件企业成熟度的等级认证标准。该模型将软件能力成熟度自低到高依次划分为初始级、可重复级、已定义级、已管理级、优化级。从\_\_\_开始，要求企业建立基本的项目管理过程的政策和管理规程，使项目管理工作有章可循。

- A. 初始级      B. 已定义级    C. 可重复级      D. 已管理级

RUP (Rational Unified Process, 统一软件开发过程, 统一软件过程): 是一个面向对象且基于网络的程序开发方法论。 RUP好像一个在线的指导者, 它可以为所有方面和层次的程序开发提供指导方针, 模版以及事例支持。

软件工程过程定义谁在做什么、怎么做以及什么时候做，RUP用四个主要的建模元素表达：

- 角色 (Workers) —— “谁”
- 活动 (Activities) —— “怎么做”
- 产品(工件) (Artifacts) —— “做什么”
- 工作流 (Workflows) —— “什么时候做”



**角色：**它定义的是所执行的一组活动和所拥有的一组文档与模型。是抽象的职责定义，描述某个人或者一个小组的行为与职责。角色并不代表个人，而是说明个人在业务中应该如何表现以及他们应该承担的责任。 RUP预先定义了很多角色：

**分析员角色集：**分析员角色集用于组织主要从事需求获取和研究的各种角色

**开发角色集：**开发人员角色集用于组织主要从事软件设计与开发的各種角色。

**测试员角色集：**测试员角色集用于组织主要从事软件测试的各种角色。

**经理角色集：**经理角色集用于组织主要从事软件工  
程流程的管理与配置的各种角色。

**活动：**是一个有明确目的的独立工作单元。即承担这一角色的人必须完成的一组工作。

例如：找出用例和角色是系统分析员的活动，执行性能测试是测试员的活动，等等。

**产品（工件）：**产品是一个过程所生产、修改或使用的一段信息。产品是项目切实的成果，是项目为生产出最终的产品而制造或使用的东西。产品可以具有不同的形式，如

- 模型，如用例模型或设计模型。
- 模型元素，如类，用例或子系统。
- 文档，如商业用例或软件体系结构文档。
- 源代码。
- 可执行程序

**workflows:** 仅仅把所有的角色、活动和产品都列举出来还不能够组成过程，另外还需要一种有效的方式，把产生有价值结果的活动序列描述出来，并显示角色之间的交互。工作流是一个产生具有可观察的结果活动序列。UML中，可以用一个序列图、协作图或活动图来表示工作流。

RUP被划分为六个核心“工程” workflow:

- 商业建模 workflow
- 需求 workflow
- 分析和设计 workflow
- 实现 workflow
- 测试 workflow
- 展开 workflow

- 【软件设计师考试2009年5月上午试题16】

一个软件开发过程描述了“谁做”、“做什么”、“怎么做”和“什么时候做”，RUP用  A  来表述“谁做”。

A. 角色      B. 活动      C. 制品      D. workflow

过程随着时间动态组织，把软件的生存期划分为一些周期，每个周期都影响新一代产品。RUP把一个开发周期划分为四个连续的阶段：

- 初始阶段 (Inception phase)
- 精化阶段 (Elaboration phase)
- 构造阶段 (Construction phase)
- 移交阶段 (Transition phase)



每个阶段的结果都是一个里程碑。里程碑是一个时间点，在这个时间点上必须做出重要的决策，达到一些关键的目标。

**初始阶段:** 为系统建立商业用例，确定项目的边界。——生命周期目标里程碑。

**精化阶段:** 分析问题领域，建立一个健全的体系结构基础，编制项目规划，淘汰项目中风险最高的元素。——生命周期体系结构里程碑。

**构造阶段:** 将开发所有剩余的构件和应用部件，对它们进行测试，并集成到产品中。——初始运行能力里程碑。

**移交阶段:** 把软件产品交付给用户群。——产品发布里程碑。

- 【软件设计师考试2009年5月上午试题30】

RUP 在每个阶段都有主要目标，并在结束时产生一些制品。  
在 C 结束时产生“在适当的平台上集成的软件产品”。

A. 初期阶段

B. 精化阶段

C. 构建阶段

D. 移交阶段

- 【软件设计师考试2008年5月上午试题18】

RUP分为4个阶段，每个阶段结束时都有重要的里程碑，其中  
生命周期架构是在 A 结束时的里程碑。

A. 初期阶段

B. 精化阶段

C. 构建阶段

D. 移交阶段

## ● 软件质量管理

**软件质量**是指反映软件系统或软件产品满足规定或隐含需求的能力的特征和特性全体。**软件质量保证**是指为软件系统或软件产品充分满足用户要求的质量而进行的有计划、有组织的活动，其目的是产生质量的软件。

### • 软件质量模型

ISO/IEC 9126软件质量模型

Mc Call软件质量模型

## ISO/IEC 9126软件质量模型

由3个层次组成：质量特性，质量子特性，度量指标。

功能性：适合性、准确性、互用性、依从性、安全性。

可靠性：成熟性、容错性、易恢复性。

易使用性：易理解性、易学性、易操作性。

效率：时间特性、资源特性。

可维护性：易分析性、易改变性、稳定性、易测试性。

可移植性：适应性，易安装性、一致性、易替换性。

- 【软件设计师考试2008年11月上午试题31】

ISO/IEC 9126 软件质量模型中第一层定义了六个质量特性，并为各质量特性定义了相应的质量子特性。子特性C属于可靠性质量特性。

A. 准确性    B. 易理解性    C. 成熟性    D. 易学性

## 1.2 系统分析基础知识

系统分析侧重于从业务全过程的角度进行分析，  
主要任务。主要内容有：

- 业务和数据的流程是否通畅，是否合理
- 数据、业务过程和组织管理之间的关系
- 原系统管理模式改革和新系统管理方法的实现是
- 否具有可行性等。

## ● 系统分析的步骤

- 对当前系统进行详细调查，收集数据；
- 建立当前系统的逻辑模型；
- 对现状进行分析，提出改进意见和新系统应达到的目标；
- 建立新系统的模型；
- 编写系统方案说明书

## ● 结构化分析方法

数据流图 (Data Flow Diagram, DFD): 数据流图就是组织中信息运动的抽象, 是信息系统逻辑模型的主要形式。它是一种便于用户理解、分析系统数据流程的图形工具。



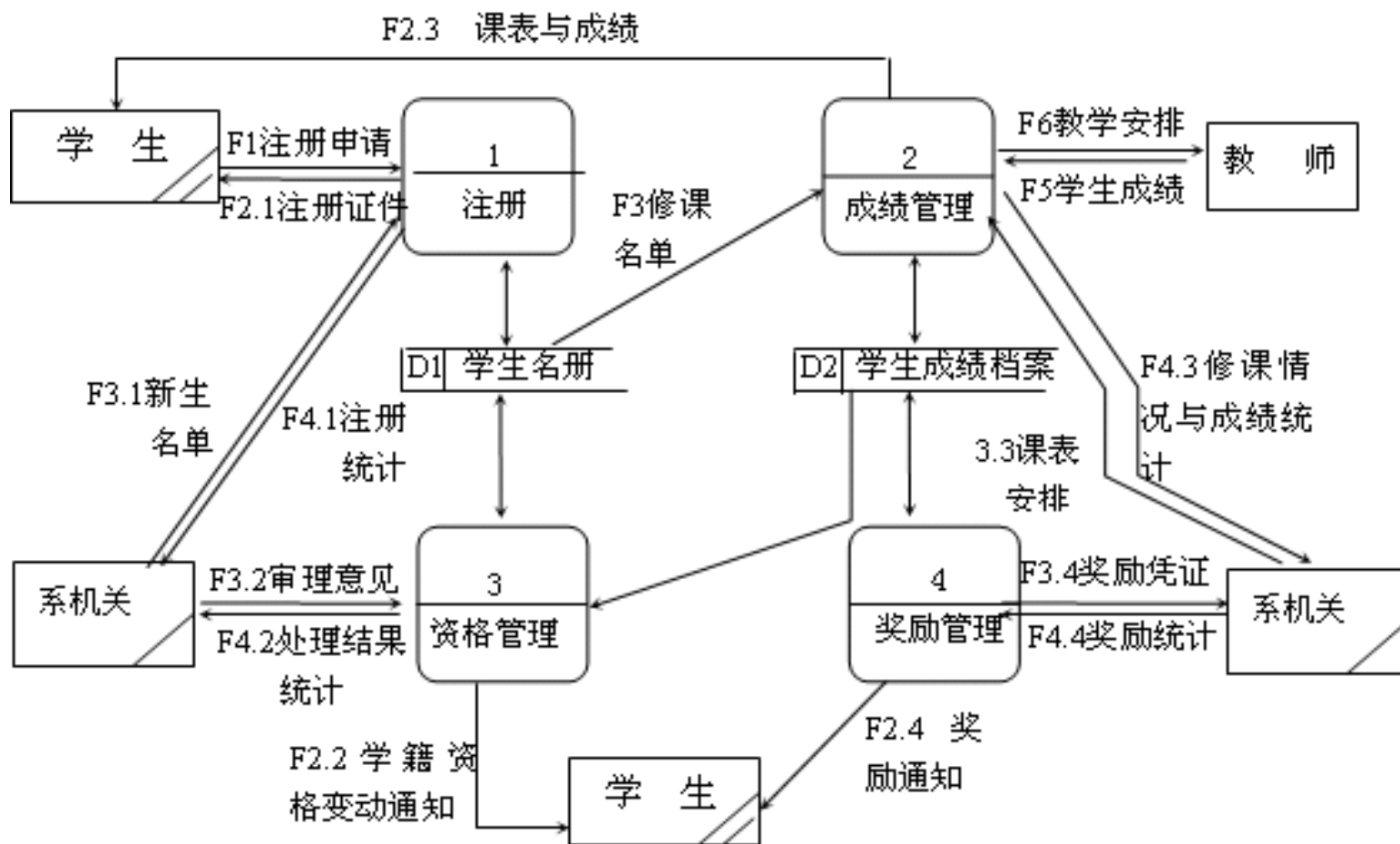
- 【软件设计师考试2007年5月上午试题15】

结构化开发方法中，数据流图是\_\_\_\_\_阶段产生的成果。

A. 需求分析  
C. 详细设计

B. 总体设计  
D. 程序编程



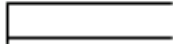



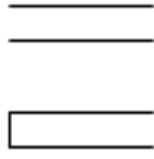





## 例如：学籍管理系统的数据流图



## 数据流图的基本组成及符号

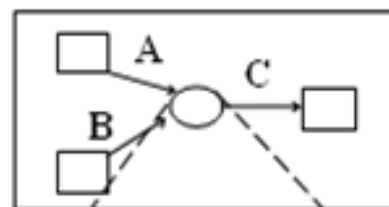
- 外部项(外部实体): 外部项在数据流图中表示所描述系统的数据来源和去处的各种实体或工作环节。系统开发不能改变这些外部项本身的结构和固有属性。
- 加工(数据加工): 又称数据处理逻辑, 描述系统对信息进行处理的逻辑功能。
- 数据存储: 逻辑意义上的数据存储环节, 即系统信息处理功能需要的、不考虑存储物理介质和技术手段的数据存储环节。
- 数据流: 与所描述系统信息处理功能有关的各类信息的载体, 是各加工环节进行处理和输出的数据集合。

## 给出了常用的三类数据流图基本成分的符号

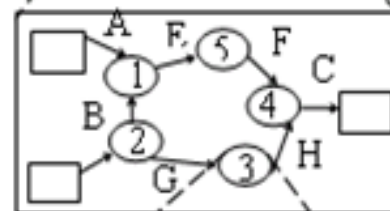
成分 类型	外部项 (外部实体)	加工 (处理逻辑)	数据存储	数据流
I				
II				
III				

## 绘制数据流图按照自顶向下的原则

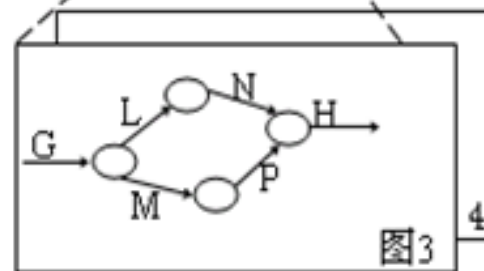
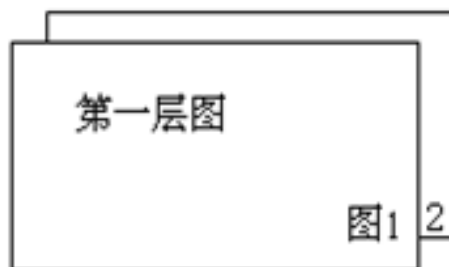
顶层图



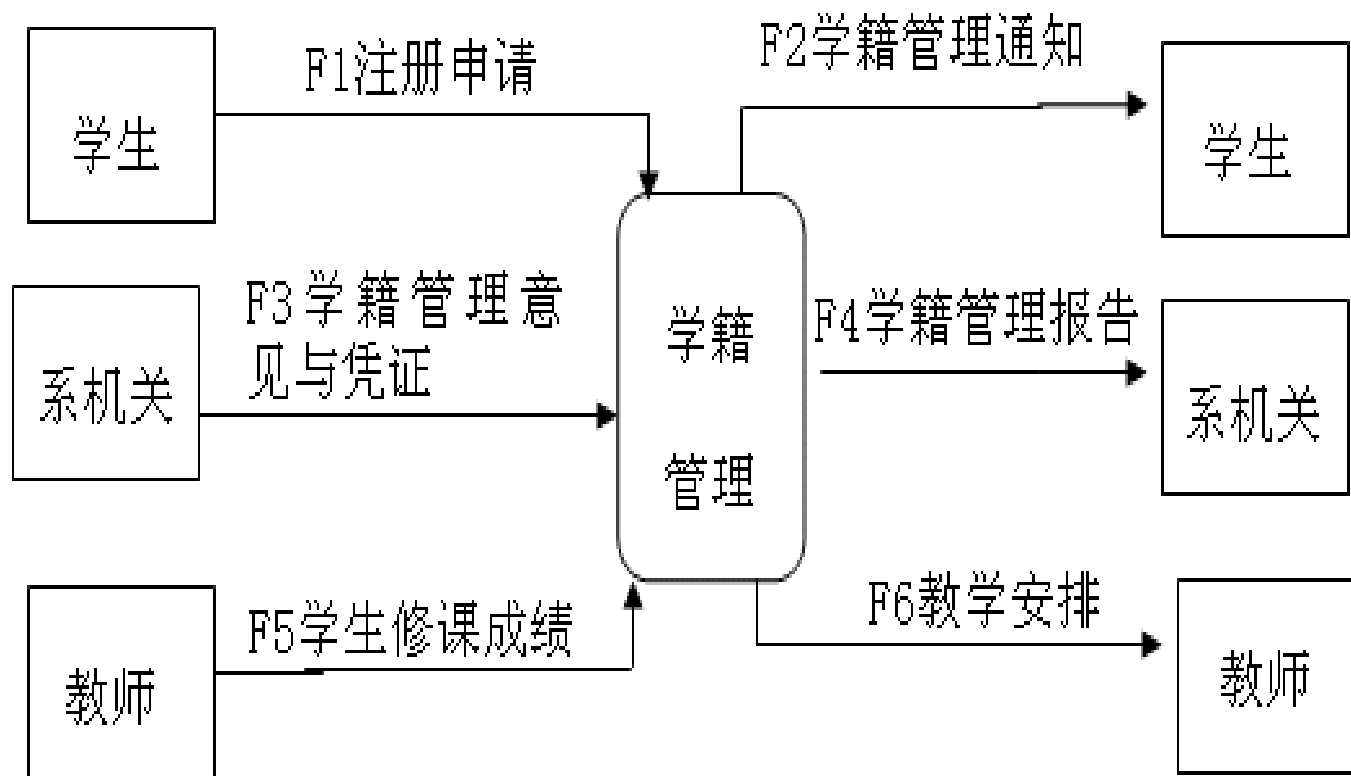
第0层图



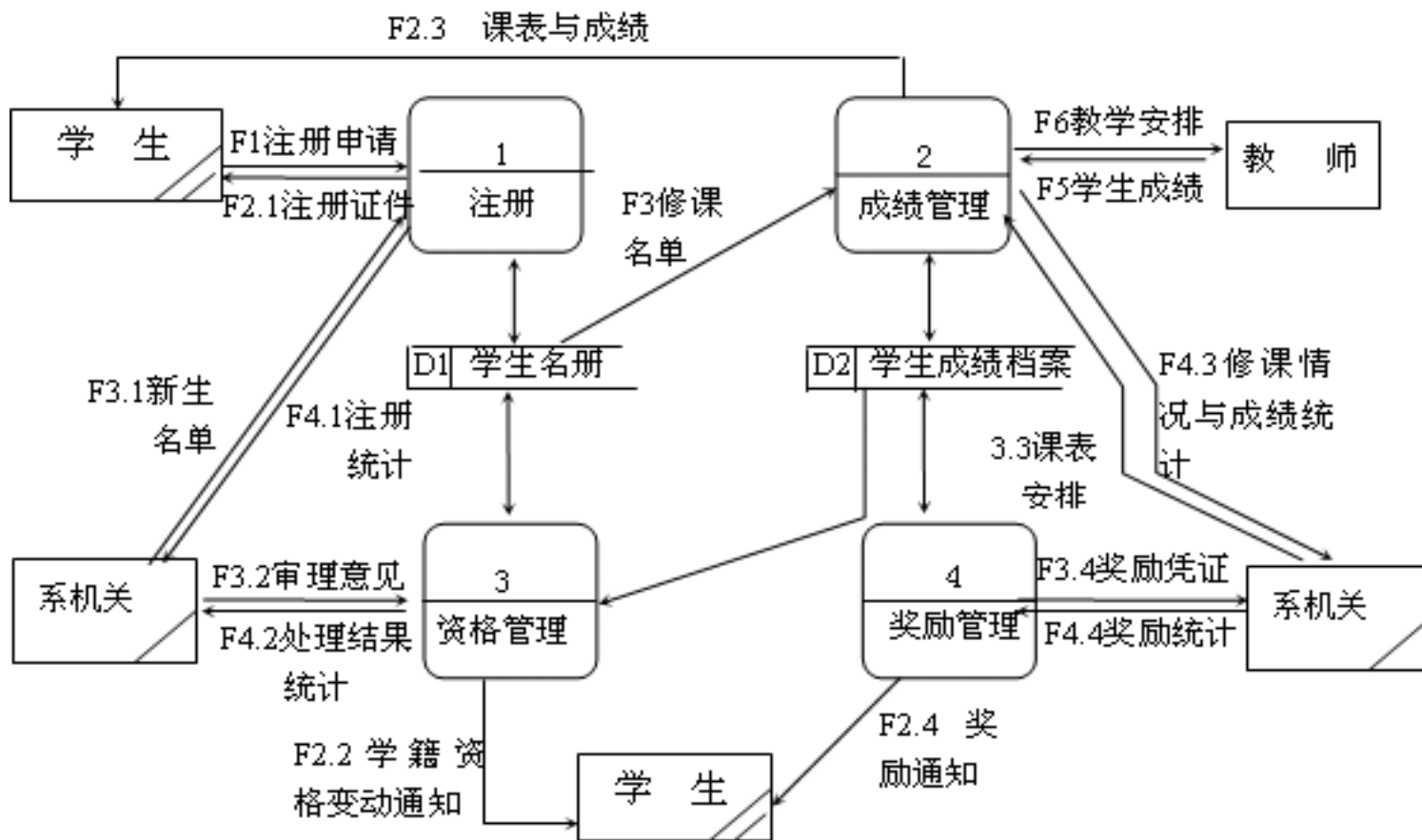
第一层图



# 例如：学籍管理系统的顶层图



## 例如：学籍管理系统的第0层图



## 绘制数据流图的几点注记:

1. 关于自顶向下、逐层分解
2. 数据流必须通过加工，即送去加工或从加工环节发出。
3. 数据存储环节一般作为两个加工环节的界面。
4. 命名
  - 名称要反映被命名的成分的真实和全部的意义；
  - 名称要意义明确，易理解，无歧义；
  - 进出数据存储环节的数据流如内容和存贮者的数据相同，可采用同一名称。



## 5. 编号

每个数据加工环节和每张数据流图都要编号。按逐层分解的原则，父图与子图的编号要有一致性，一般子图的图号是父图上对应的加工的编号。类似地，在分层数据流图中，如下层图上的数据流或数据存储是由上层图某个成分的分解而得，则父项与子项的编号要体现数据流图分解的完整性与一致性的原则，如父项编号为 F1或D1，则其子项分别为 F1.1，F1.2，…，或D1.1，D1.2，…等。

下列要素中，不属于DFD的是(1) D。当使用DFD对一个工资系统进行建模时，(2) A可以被认定为外部实体。

(1) A. 加工    B. 数据流    C. 数据存储    D. 联系

(2) A. 接收工资单的银行    B. 工资系统源代码程序  
C. 工资单    D. 工资数据库的维护

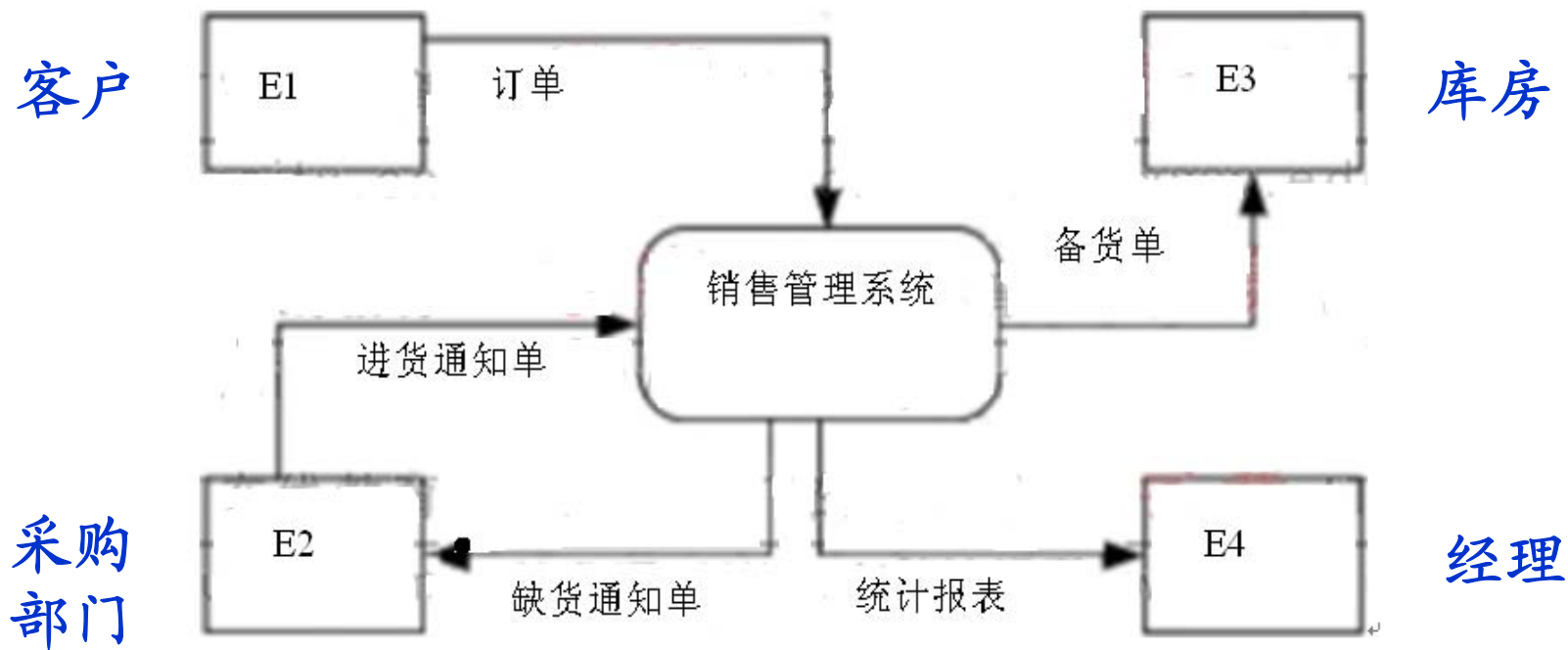
# 【软件设计师考试2008年11月下午试题31】

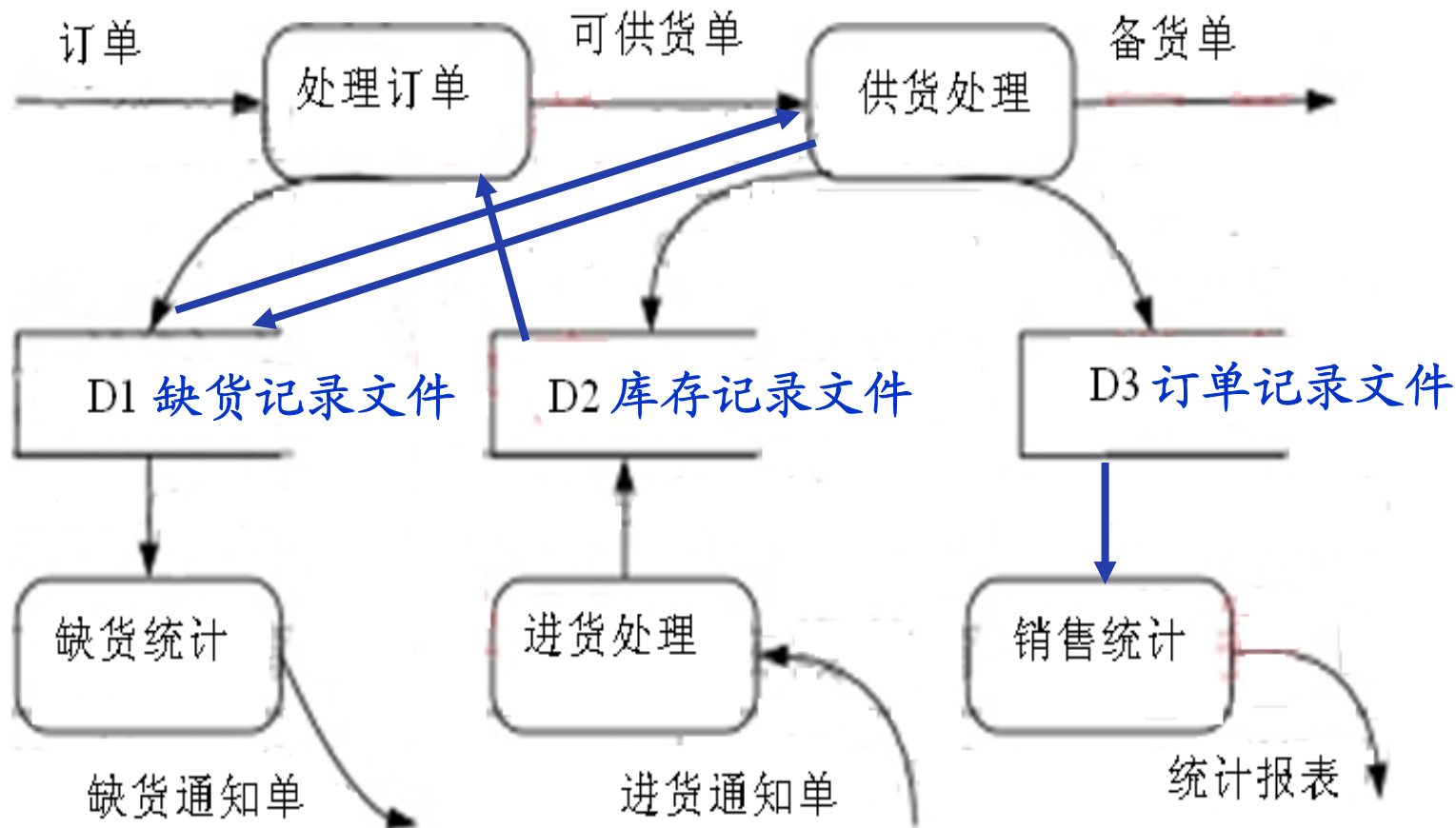
## 【说明】

某公司拟开发一个销售管理系统，其主要功能描述如下：

1. 接受客户订单，检查库存货物是否满足订单要求。如果满足，进行供货处理：即修改库存记录文件，给库房开具备货单并且保留客户订单至订单记录文件；否则进行缺货处理：将缺货订单录入缺货记录文件。
2. 根据缺货记录文件进行缺货统计，将缺货通知单发给采购部门。
3. 根据采购部门提供的进货通知单进行进货处理：即修改库存记录文件，并从缺货记录文件中取出缺货订单进行供货处理。
4. 根据保留的客户订单进行销售统计，打印统计报表给经理。

现采用结构化方法对销售管理系统进行分析与设计，获得如图所示的顶层图和第0层数据流图。





**【问题1】（4分）**

使用说明中的词语，给出顶层图的外部实体E1~E4的名称。

**【问题2】（3分）**

使用说明中的词语，给出第0层图的数据存储D1~D3的名称。

**【问题3】（8分）**

第0层数据流图缺少了四条数据流，根据说明及顶层数据流图提供的信息，分别指出这四条数据流的起点和终点。

起 点	终 点
D3 订货记录文件	销售统计
D2 库存记录文件	处理订单
D1 缺货记录文件	供货处理
进货处理	D1 缺货记录文件



## 【说明】

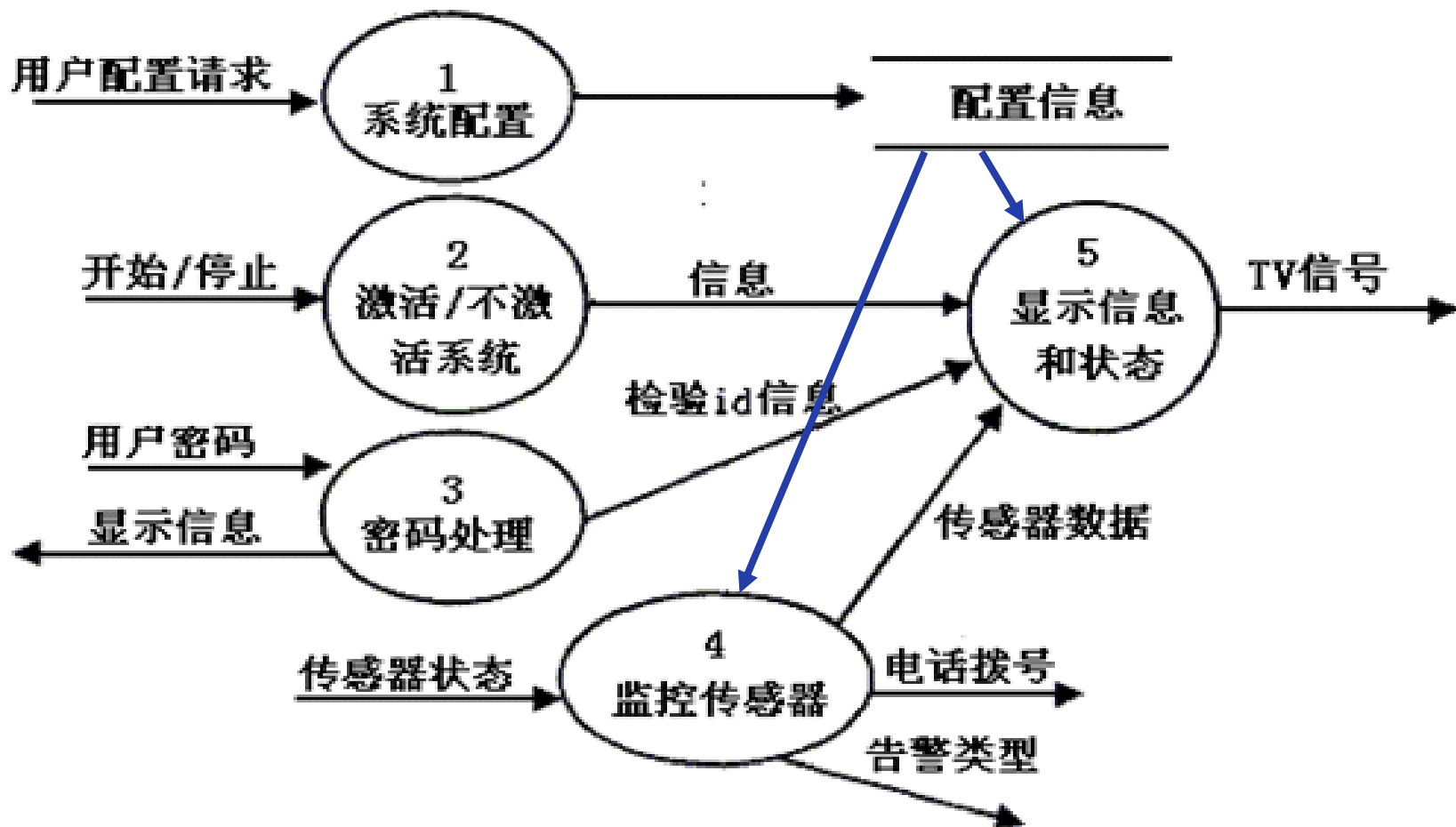
某基于微处理器的住宅安全系统，使用传感器(如红外探头、摄像头等)来检测各种意外情况，如非法进入、火警、水灾等。

房主可以在安装该系统时配置安全监控设备(如传感器、显示器、报警器等)，也可以在系统运行时修改配置，通过录像机和电视机监控与系统连接的所有传感器，并通过控制面板上的键盘与系统进行信息交互。在安装过程中，系统给每个传感器赋予一个编号(即id)和类型，并设置房主密码以启动和关闭系统，设置传感器事件发生时应自动拨出的电话号码。当系统检测到一个传感器事件时，就激活警报，拨出预置的电话号码，并报告关于位置和检测到的事件的性质等信息。

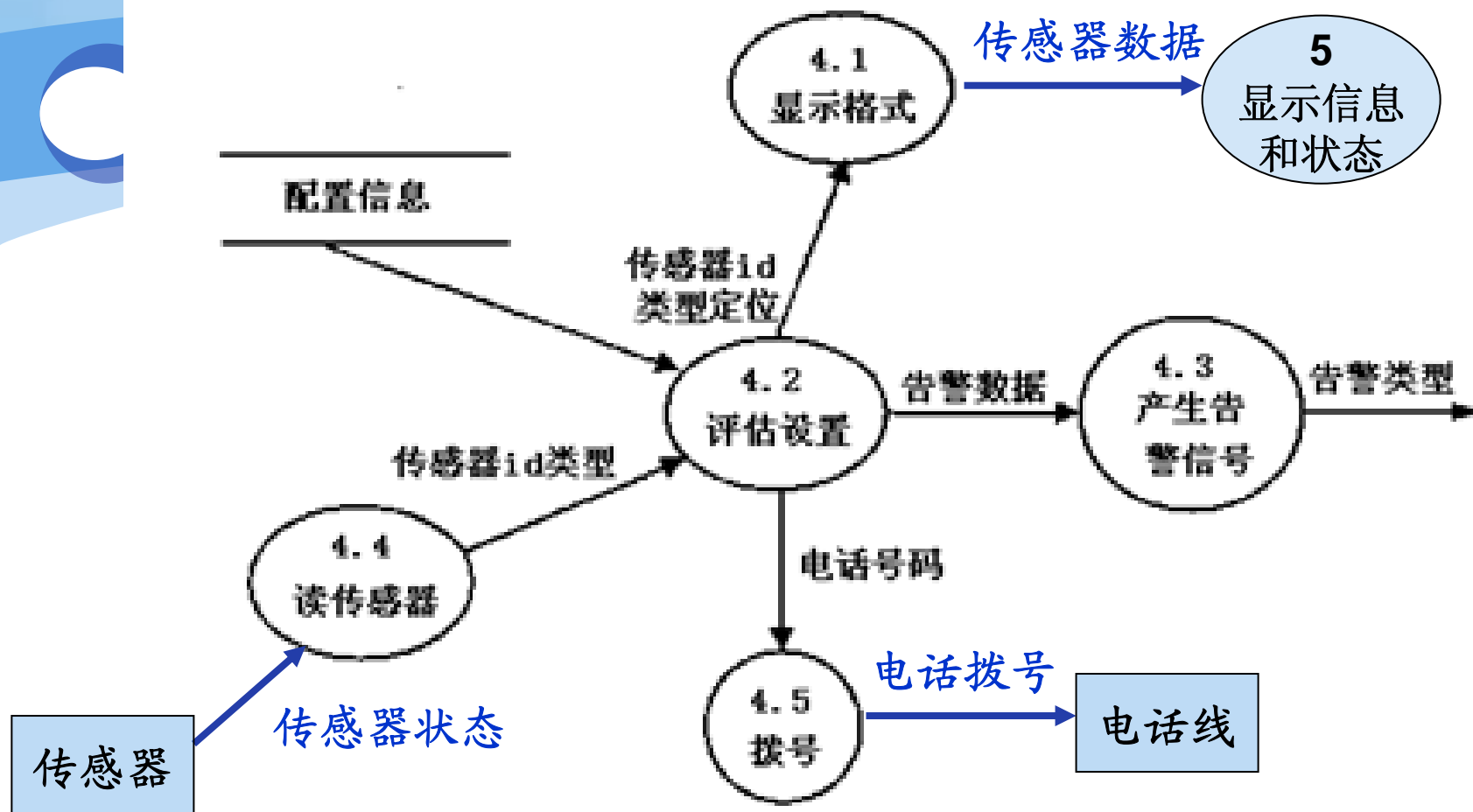


# 传感器





【问题2】住宅安全系统第0层DFD图中的数据存储“配置信息”会影响图中哪些加工？



**【问题3】** 将加工4的细化图中的数据流补充完整，并指明加工名称、数据流的方向(输入/输出)和数据流名称。



加工名称	数据流方向	数据流名称
4.1 显示格式	输出	传感器数据
4.4 读传感器	输入	传感器状态
4.5 拨号	输出	电话拨号

- **【软件设计师考试2006年5月上午试题31】**

在绘制数据流图时，应遵循父图与子图平衡的原则，所谓平衡是指\_\_\_\_\_。

- A. 父图和子图都不得改变数据流的性质。
- B. 子图不改变父图数据流的一致性。
- C. 父图的输入/输出数据流与子图的输入/输出数据流一致。
- D. 子图的输出数据流完全由父图的输入数据流确定。

## 1.3 系统设计基础知识

- 系统模块结构设计

- 【软件设计师考试2006年5月上旬试题16、17】

耦合度描述了\_\_\_\_\_。

- A. 模块内各种元素结合的程度。
- B. 模块内多个功能之间的接口。
- C. 模块之间公共数据的数量。
- D. 模块之间相互关联的程度。

内聚是一种指标，表示一个模块\_\_\_\_\_。

- A. 代码优化的程度
- B. 代码功能的集中程度
- C. 完成任务的及时程度
- D. 为了与其他模块连接所要完成的工作量

- 【软件设计师考试2007年11月上午试题32】

内聚性和耦合性是度量软件模块独立性的重要准则，软件设计时应力求  B  。

- |            |            |
|------------|------------|
| A. 高内聚，高耦合 | B. 高内聚，低耦合 |
| C. 低内聚，高耦合 | D. 低内聚，低耦合 |

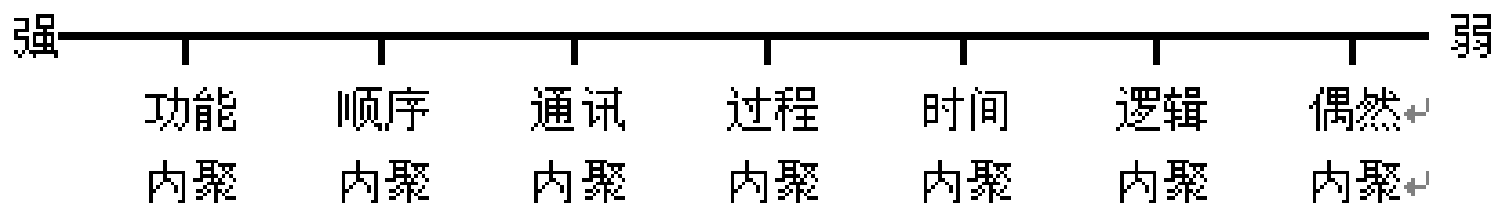
**耦合**: 系统内不同模块之间互连程度的度量。块间耦合强弱取决于模块间联系形式及接口的复杂程度。模块间接口的复杂性越高, 说明耦合的程度也越高。

- **数据耦合**: 如果两个模块彼此间通过数据交换信息, 而且每一个参数仅仅为数据, 那么这种块间耦合称之为数据耦合。
- **控制耦合**: 如果两个模块彼此间传递的信息中有控制信息, 那么这种块间耦合称为控制耦合。
- **公共耦合**: 如果两个模块彼此之间通过一个公共的数据区域传递信息时, 则称之为公共耦合或公共数据域耦合。
- **内容耦合**: 如果一个模块需要涉及另一个模块的内部信息时, 则这种联系称为内容耦合。

块间耦合形式	可读性	错误扩散能力	可修改性	通用性
数据耦合	好	弱	好	好
控制耦合	中	中	中	中
公共耦合	不好	强	不好	较差
内容耦合	最差	最强	最差	差



**内聚**: 模块内部元素的联系方式, 块内联系标志一个模块内部各个元素间彼此结合的紧密程度, 主要表现在模块内部各个元素为了执行某一功能而结合在一起的程度。



- **偶然内聚**: 如果一个模块所要完成的动作之间没有任何关系, 或者即使有某种关系, 也是非常松散的, 就称之为偶然组合。
- **逻辑内聚**: 如果一个模块内部的各个组成部分在逻辑上具有相似的处理动作, 但功能上、用途上却彼此无关, 则称之为逻辑组合。
- **时间内聚**: 如果一个模块内部的各个组成部分所包含的处理动作必须在同一时间内执行, 则称之为时间组合。
- **过程内聚**: 如果一个模块内部的各个组成部分所要完成的动作彼此间没什么关系, 但必须以特定的次序(控制流)执行, 则称之为过程组合。
- **通信内聚**: 如果一个模块内部的各个组成部分所完成的动作都使用了同一个输入数据或产生同一个输出数据, 则称之为通信组合。
- **顺序内聚**: 对于一个模块内部的各个组成部分, 如果前一部分处理动作的输出是后一部分处理动作的输入, 则称之为顺序组合。
- **功能内聚**: 如果一个模块内部的各个组成部分全部为执行同一功能而结合成为一个整体, 则称之为功能组合方式的模块。

## 1.4 系统实施基础知识

### ● 系统测试

**系统测试：**是为了发现错误而执行程序的过程

**系统测试的目的：**系统测试是以找错误为目的，我们不是要证明程序无错，而是要精心选取那些易于发生错误的测试数据，以十分挑剔的态度，去寻找程序的错误。

## 系统测试的基本原则：

- 测试工作应避免由原开发软件的个人或小组来承担
- 设计测试用例不仅要包括合理、有效的输入数据，还要包括无效的或不合理的输入数据。
- 设计测试案例时，不仅要确定输入数据，而且要根据系统功能确定预期输出的结果。
- 不仅要检验程序做了该做的事，还要检查程序是否同时做了不该做的事。
- 严格按照测试计划进行，避免测试的随意性
- 保留测试用例，将会给重新测试和追加测试带来方便。

## 测试过程:

1. 制定测试计划
2. 编制测试大纲
3. 根据测试大纲设计和生成测试用例
4. 实施测试
5. 生成测试报告

## 系统测试的方法:

### 1. 静态测试

被测程序不在机器上运行，而是采用人工检测和计算机辅助静态分析的手段对程序进行检测。

### 2. 动态测试

通过运行程序发现错误。有黑盒测试法和白盒测试法。

**黑盒测试：**将软件看作黑盒子，在完全不考虑程序的内部结构和特性的情况下，研究软件的外部特性。根据软件的需求规格说明书设计测试用例，只在软件的界面上进行测试，从程序的输入和输出特性上测试是否满足设定的功能。

黑盒测试方法（黑盒测试法用例）主要有等价类划分、边界值分析、错误推测等，主要用于软件系统测试阶段。

**白盒测试：**是在已知程序内部结构和处理过程的前提下，通过测试来检测程序中的每条路径是否按预定要求正常运行。

该方法将被测试的对象看成一个透明的白盒子，测试人员完全知道程序的内部结构和处理算法，并按照程序内部的逻辑测试程序，对程序中尽可能多的逻辑路径进行测试。

白盒测试方法主要有逻辑覆盖、基本路径测试等。



## 系统测试的步骤：

- 单元测试**：程序中的一个模块或一个子程序，是程序设计的最小单元，是程序最小的独立编译单位。
- 集成测试（组装测试）**：在每个模块完成了单元测试以后，需要按照设计时作出的层次模块图把它们连接起来，进行组装测试。
- 确认测试**：经过组装测试，软件已装配完毕，接下来进行的确认测试和系统测试将是以整个软件做为测试对象，且采用黑盒测试方法。
- 系统测试**：将信息系统的所有组成部分包括软件、硬件、用户以及环境等综合在一起进行测试，以保证系统的各组成部分协调运行。

下面有关测试的说法正确的是\_\_\_\_\_。

- A. 测试人员应该在软件开发结束后开始介入
- B. 测试主要是软件开发人员的工作
- C. 要根据软件详细设计中设计的各种合理数据设计测试用例
- D. 严格按照测试计划进行，避免测试的随意性。

为验证程序模块A是否正确实现了规定的功能，需要进行（1）；为验证模块A能否与其他模块按照规定方式正确工作，需要进行（2）。

- |     |         |         |
|-----|---------|---------|
| （1） | A. 单元测试 | B. 集成测试 |
|     | C. 确认测试 | D. 系统测试 |
| （2） | A. 单元测试 | B. 集成测试 |
|     | C. 确认测试 | D. 系统测试 |

- 【软件设计师考试2008年11月上午试题36】

在模拟环境下，常采用黑盒测试检验所开发的软件是否与需求规格说明书一致。其中有效性测试属于\_\_\_\_\_中的一个步骤。

- A. 单元测试
- B. 集成测试
- C. 确认测试
- D. 系统测试

## 1.5 系统运行和维护基础知识

系统可维护性概念：

维护人员理解、改正、改动和改进这个软件的难易程度。

系统的可维护性的评价指标：

可理解性、可测试性、可修改性。

系统维护的内容和类型：

硬件维护、软件维护、数据维护。

软件维护：

根据需求变化或硬件环境的变化对应用程序进行部分或全部修改。

## 软件维护包括:

**正确性维护:** 改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。占整个维护工作量的17%-20%。

**适应性维护:** 使应用软件适应信息技术变化和管理需求变化而进行的修改。占整个维护工作量的18%-25%。

**完善性维护:** 为扩充功能和改善性能而进行的修改，主要是对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。占整个维护工作量的50%-60%。

**预防性维护:** 为改进应用软件的可靠性和可维护性，为了使使用硬件环境的变化，主动增加预防性的新的功能，以使应用系统适应各类变化而不被淘汰。占整个维护工作量的4%左右。

- 【软件设计师考试2008年11月上午试题34】

系统的可维护性可以用系统的可维护性评价指标来衡量。系统的可维护性评价指标不包括\_\_\_\_\_。

- A. 可理解性
- B. 可修改性
- C. 准确性
- D. 可测试性

各类软件维护活动中，\_\_\_\_\_维护占整个维护工作的比重最大

- A. 完善性
- B. 改正性
- C. 适应性
- D. 预防性



❖ 谢谢大家！



# 软件设计师培训

# 历年考试对本章内容的考查情况

考次	2004		2005		2006		2007		2008		2009
	5月	11月	5月	11月	5月	11月	5月	11月	5月	11月	5月
分值	6	20	20	20	20	20	22	20	21	21	21
比重	4%	13.3%	13.3%	13.3%	13.3%	13.3%	14.7%	13.3%	14%	14%	14%

本章的重点程度：★★★★

## 2. 数据库技术基础知识

### 大纲要求：

- 数据库管理系统的功能和特征
- 数据库体系结构（概念模式、外模式、内模式）
- 数据模型，ER图，第一范式、第二范式、第三范式
- 数据操作（集合运算和关系运算）
- 数据库语言（SQL）
- 数据库的控制功能（并发控制、恢复、安全性、完整性）
- 数据仓库和分布式数据库基础知识

## 2.1 数据库系统的基本概念

### ● DB、DBMS和DBS的定义

**DB (数据库)** 是长期存储在计算机内、有组织的、统一管理的相关数据的集合。

**DBMS(数据库管理系统)**是数据库系统中管理数据的软件系统。位于用户与操作系统之间的一层管理软件。

**DBS (数据库管理系统)** 在计算机系统中引入数据库后的系统。是数据库、硬件、软件、数据库管理员及用户的集合。

- 数据库管理系统的功能

1. 数据库的定义功能

DBMS提供数据定义语言 (DDL) 定义数据库的三级结构两级映象，定义数据的完整性、安全控制约束。

2. 数据库的操作

DBMS提供数据操作语言 (DML) 实现对数据库中数据的操作。基本数据操作有：检索(查询)、和更新(插入、删除、修改)。

### 3. 数据库的保护功能

DBMS对数据的保护主要通过四个方面实现，因而DBMS中包括四个子系统。

- 数据库恢复
- 数据库的并发控制
- 数据库的完整性控制
- 数据库的安全性控制

## 4. 数据库存储管理

DBMS的存储管理子系统提供了数据库中数据和应用程序的一个界面，DBMS存储管理子系统的职责是把各种DML语句转换成底层的与磁盘中数据打交道的操作系统的文件系统命令，起到数据的存储、检索和更新的作用。



## 5. 数据库的维护功能

- 数据装载程序
- 备份程序
- 文件重组组织程序
- 性能监控程序

## 6. 数据字典（DD）

数据库系统中存放三级结构定义的数据库称为数据字典。对数据库的操作都要通过访问DD才能实现。DD中还存放数据库运行的统计信息，例如记录个数、访问次数等。

## 【软件设计师考试2007年11月上旬试题51】

在数据库系统中，数据的完整性约束的建立需要通过数据库管理系统提供的 (51) A 语言来实现。

(51) A. 数据定义

B. 数据操作

C. 数据查询

D. 数据控制

## 2.2 数据库系统的体系结构—三级结构两级映象

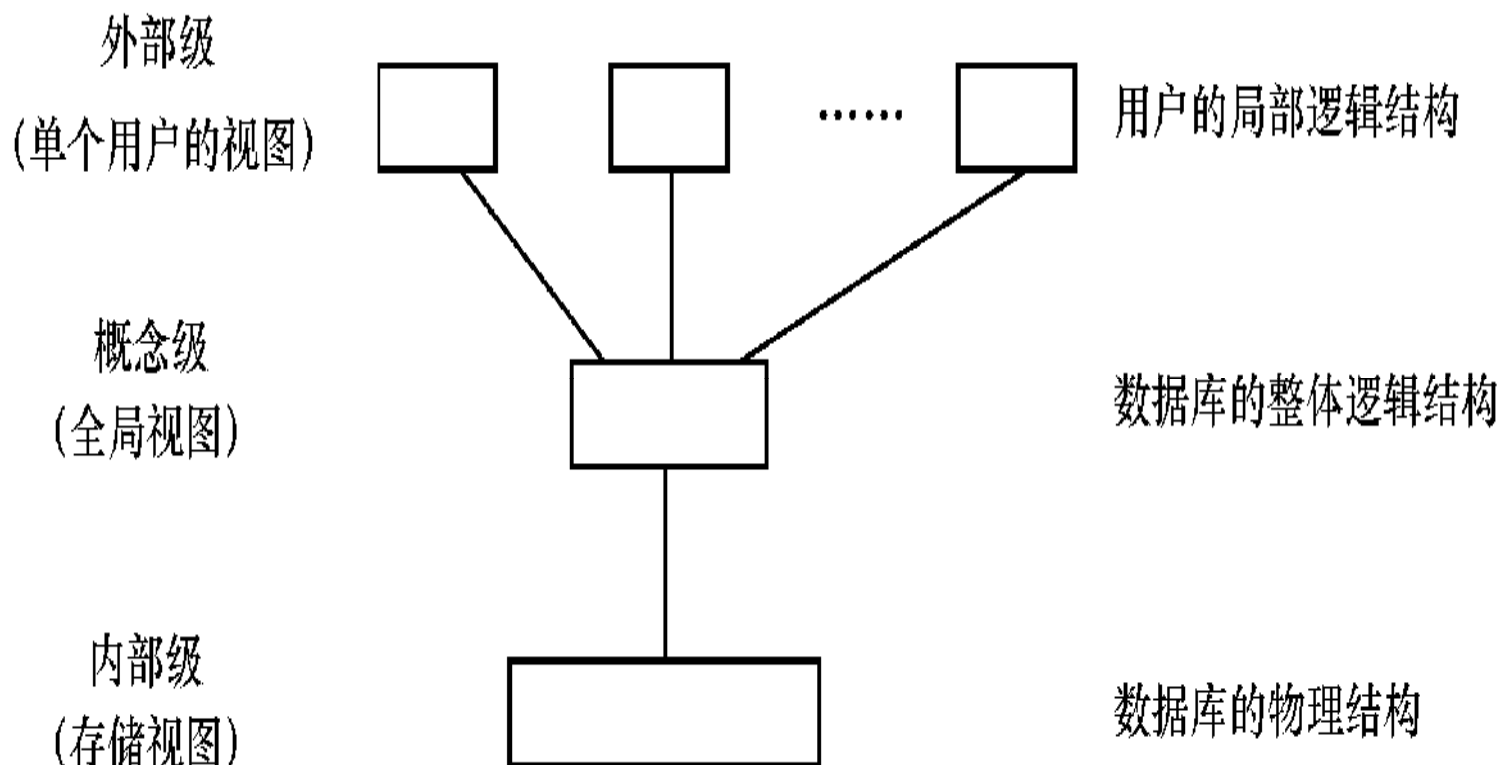
- 三级结构：外模式、概念模式、内模式
- 两级映象：外模式/模式映象、模式/内模式映象

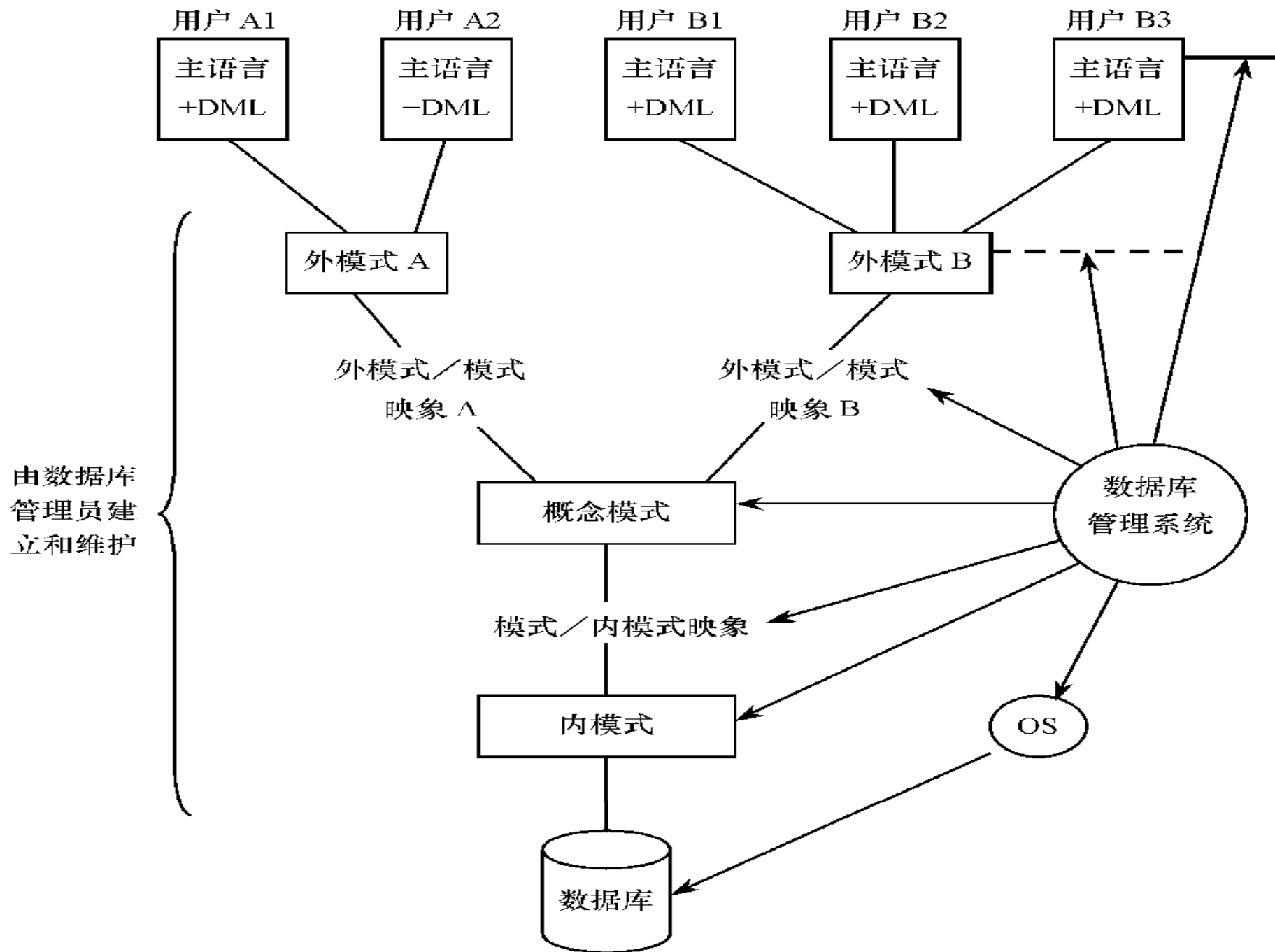
- 三级结构:

**外模式:** 单个用户使用到的那部分数据的描述。

**概念模式:** 是用户定义的数据库中全部数据逻辑结构的描述。

**内模式:** 是数据库在物理存储方面的描述, 接近于物理存储设备, 涉及到实际数据存储的结构。





## ● 两级映射

**模式 / 内模式映射：**存在于概念级和内部级之间，用于定义概念模式和内模式之间的对应性。

**外模式 / 模式映射：**存在于外部级和概念级之间，用于定义外模式和概念模式之间的对应性。

## ● 两级数据独立性

数据独立性是指应用程序和数据库的数据结构之间相互独立，不受影响。

数据独立性分成物理数据独立性和逻辑数据独立性两个级别。

**物理数据独立性**：数据的内模式修改，模式 / 内模式也要进行相应的修改，但概念模式尽可能保持不变。

**逻辑数据独立性**：数据的概念模式修改，外模式 / 模式也要进行相应的修改，但外模式尽可能保持不变。



按用户的观点对  
数据和信息建模的

## 2.3 数据模型

**数据模型**: 表示实体类型及实体间联系的模型.

根据模型应用的不同目的可以将模型化分为**概念数据模型**和**结构数据模型**

- 概念模型 — ER模型（实体联系模型）
- 结构数据模型

层次模型、网状模型、关系模型

按计算机的观点对  
数据建模的，直接  
面向数据库的结构

概念模型

结构数据模型

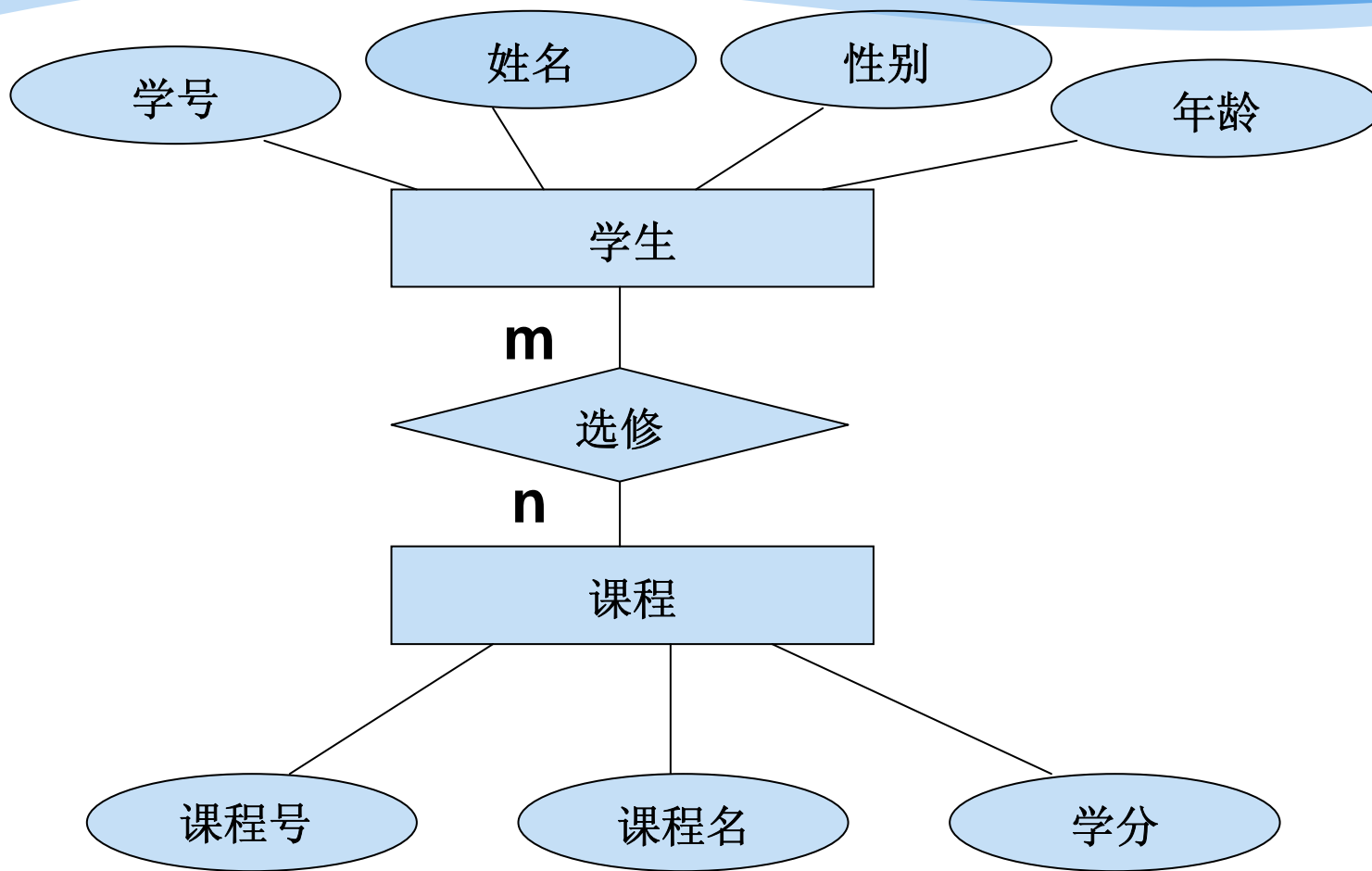
现实世界  信息世界  机器世界

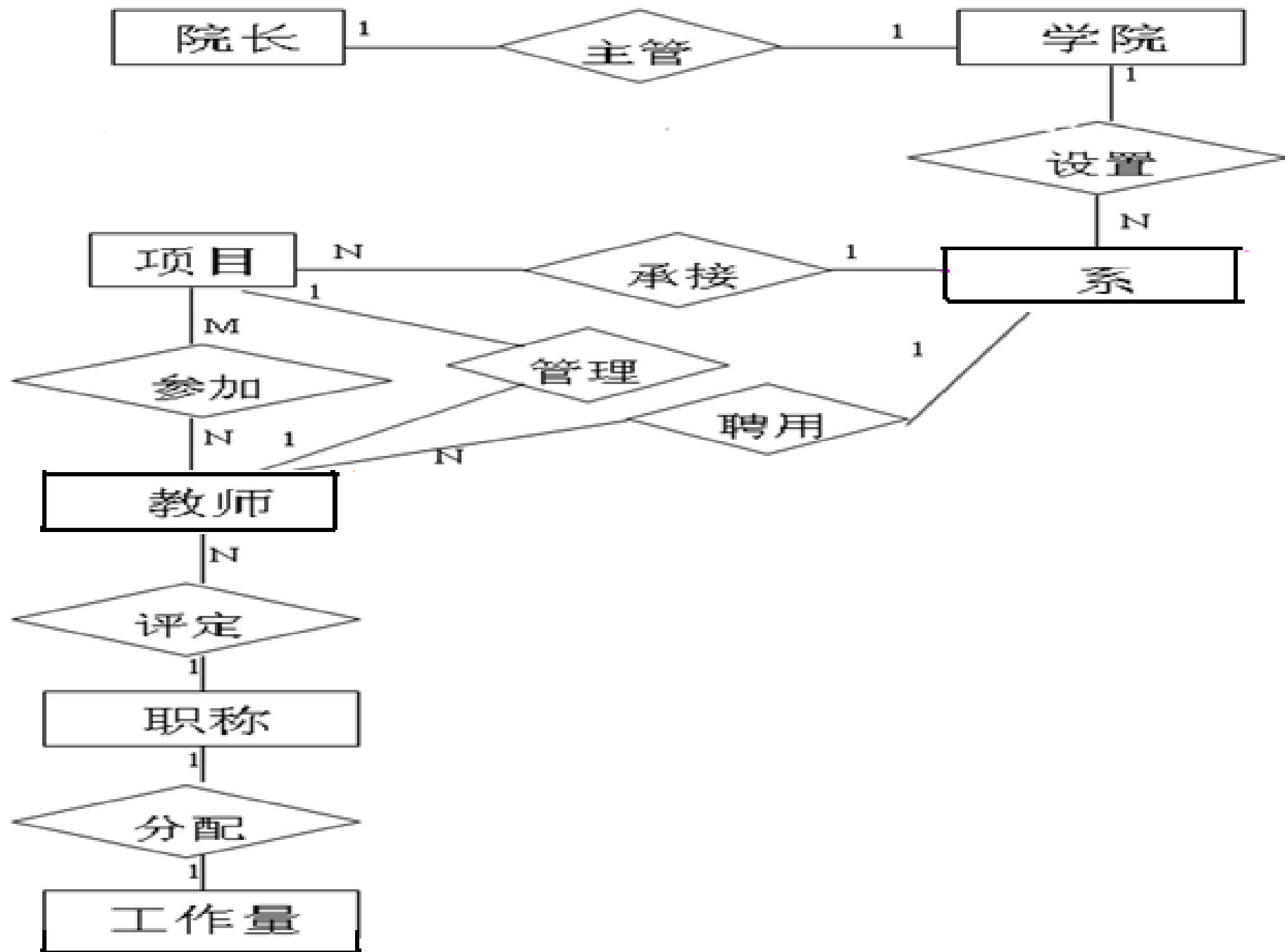


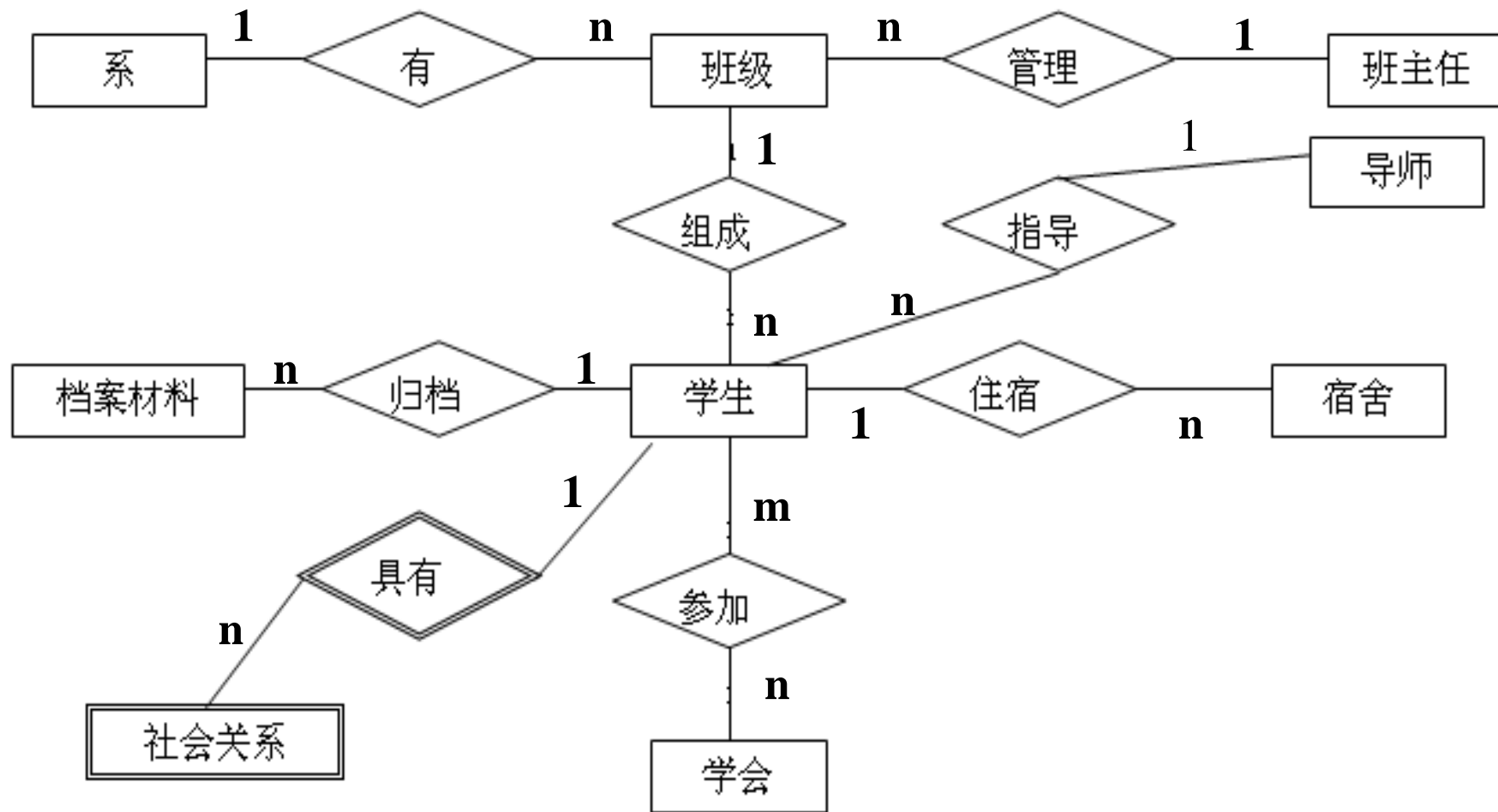
- 概念模型—— ER模型（实体联系模型）

**实体间的联系：** 实体集内部以及实体集的联系。包括一对一的联系、一对多的联系、多对多的联系。

描述实体间联系的模型称为**实体联系模型**  
简称ER模型。







## ● 数据结构模型

数据库领域中常用的数据结构模型：

- 层次模型
- 网状模型
- 关系模型

## 【软件设计师考试2009年5月上午试题51】

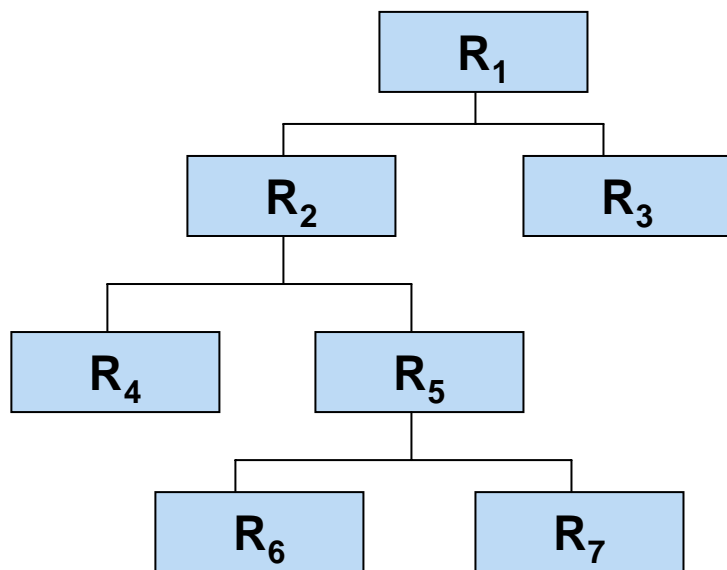
采用二维表格结构表达实体类型及实体间联系的数据模型是 （51） 。

- （51） A. 层次模型  
B. 网状模型  
C. 关系模型  
D. 面向对象模型



层次模型：

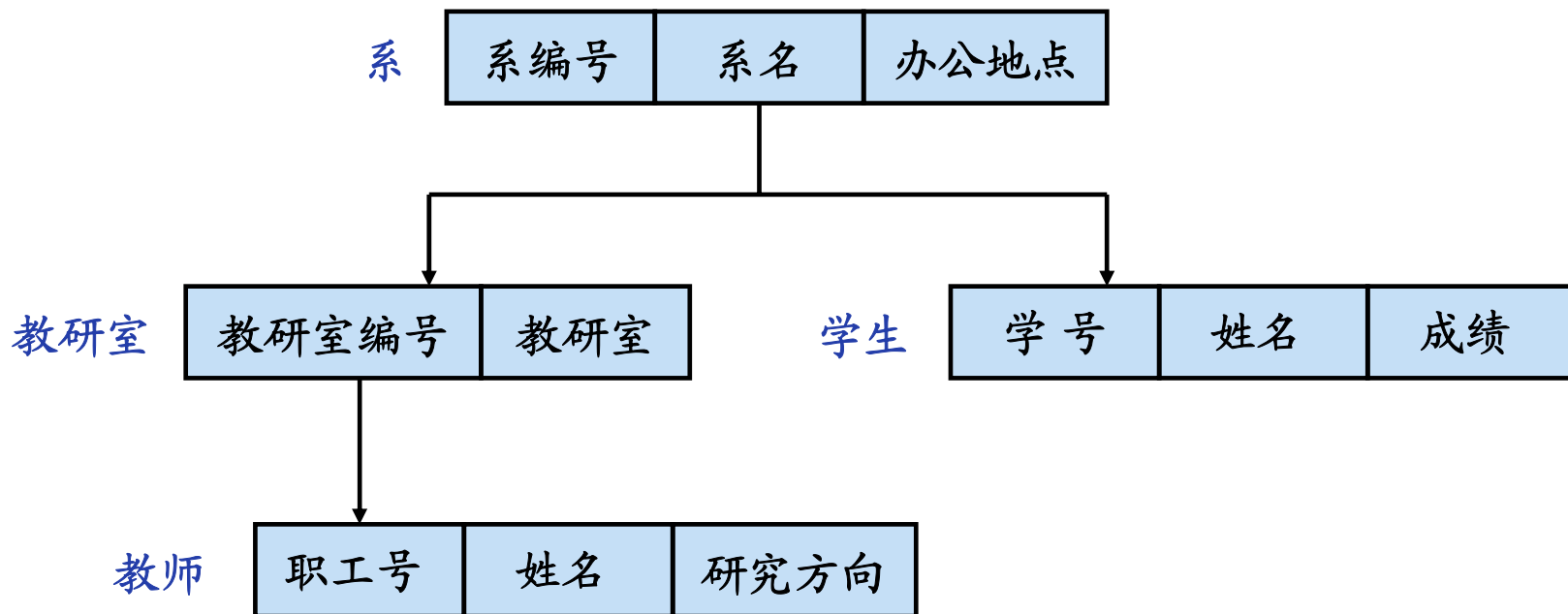
用树型结构表示实体类型及实体间联系的数据模型。



特点：

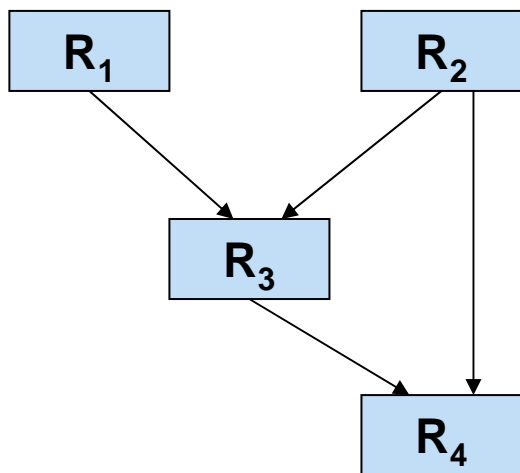
1. 有且只有一个结点无父结点；
2. 其它结点有且只有一个父结点；
3. 适用于一对多的实体联系。

## 层次模型的实例



网状模型：

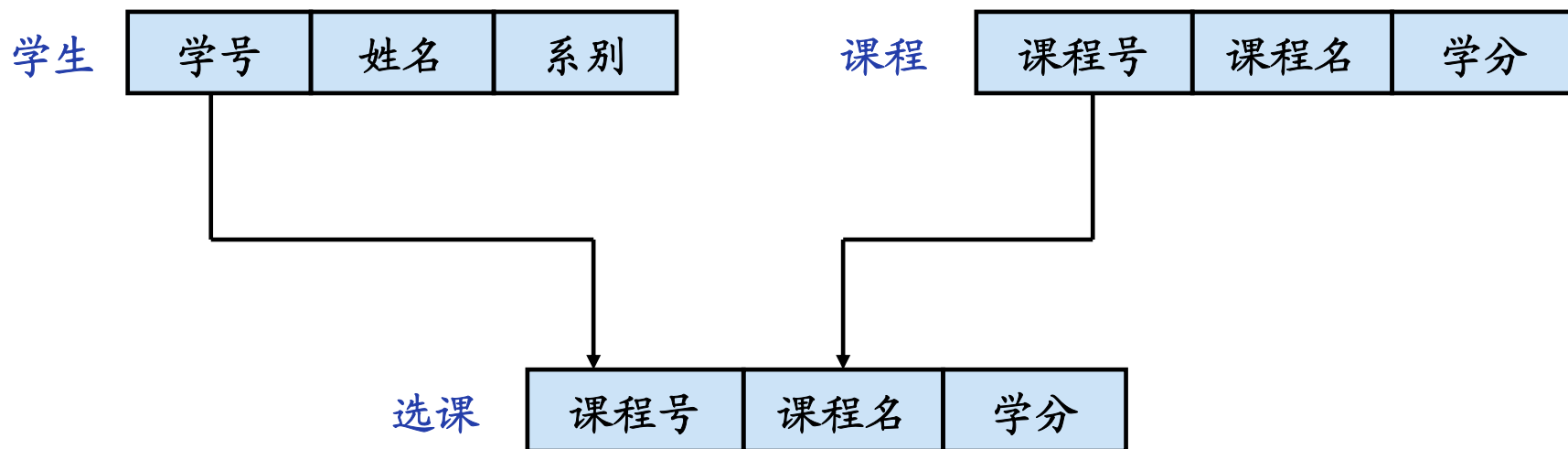
用有向图表示实体类型及实体间联系的数据模型。



特点：

1. 任何一个结点可以有一个或一个以上的父结点；
2. 任何一个结点可以没有父结点；
3. 适用于多对多的实体联系。

## 网状模型的实例



关系模型：

用二维表格结构表达实体间的联系的数据模型

学号	姓名	性别	出生日期	籍贯	所在系	相片
2007842501	张三	男	1985. 9. 4	山西	信息工程	1. jpg
2007842502	李四	女	1984. 11. 26	江苏	财务管理	2. jpg
2007842503	王五	男	1985. 4. 20	河北	电子商务	3. jpg
.....	.....	.....	.....	.....	.....	.....

关系模型中的基本术语:

**关系**: 二维的数据表, 它描述了实体之间的联系。

**元组 (实体)**: 数据表中的每一行表示一个实体。

**属性 (字段)**: 数据表中的每一列。

**域**: 属性的取值范围。

**关系模式**: 对关系的描述称为关系模式。

关系名 (属性名1, 属性名2, ……, 属性名n)

例: 学生 (学号, 姓名, 性别, 出生日期, 籍贯, 所在系)

**超键(超码)**：在关系模式中，能唯一标识元组的属性集。这个属性集可能含有多余的属性。

**候选键(候选码)**：能唯一标识元组，且又不含有多余的属性一个属性集，即超键中删除多余属性剩下的属性集。

**主键(主码)**：从候选键中选择一个作为关系模式中用户使用的候选键称为主键。

例如：在关系模式（工号，姓名，年龄，性别，工资）

（工号，姓名）是关系的一个超键；

（工号）是候选键；

（工号）是主键。

**主属性：**包含在任何候选键中的属性称为主属性。不包含在任何候选键中的属性称为非主属性。

**外键（外码）：**当关系R中的某个属性（或属性组）虽然不是该关系的码，但却是另一个关系S的码，称该属性（或属性组）为R关系的外键。

**全键（全码）：**关系模型中所有属性都是这个关系的关键字

例：R（教师，课程，学生）



例1: 学生 ( 学号, 姓名, 性别, 年龄, 系号 )

系 ( 系号, 系名, 系主任 )

外键

例2: 选课 ( 学号, 课程号, 成绩 )

外键

学生 ( 学号, 姓名, 性别 )

课程 ( 课程号, 课程名, 学分 )

## 【软件设计师考试2004年5月上午试题17、18】

已知关系R如下表所示，关系R的主属性为(17) D，候选关键字分别为(18) D。

(17) A. ABC                      B. ABD                      C. ACD                      D. ABCD

(18) A. ABC

B. AB 、 AD

C. AC、AD和CD

D. AB、AD、BD的CD

R

A	B	C	D
a	b	c	d
a	c	d	e
b	d	e	f
a	d	c	g
b	c	d	g
c	b	e	g



## 【软件设计师考试2004年11月上午试题42-44】

假定每一车次具有唯一的始发站和终点站。如果实体“列车时刻表”属性为车次、始发站、发车时间、终点站、到达时间，该实体的主键是 (42) **A**；如果实体“列车运行表”属性为车次、日期、发车时间、到达时间，该实体的主键是 (43) **D**。通常情况下，上述“列车时刻表”和“列车运行表”两实体间 (44) **C** 联系。

(42) A. 车次      B. 始发站      C. 发车时间      D. 车次，始发站

(43) A. 车次      B. 始发站      C. 发车时间      D. 车次，日期

(44) A. 不存在      B. 存在一对一

C. 存在一对多      D. 存在多对多

## 关系模型的完整性约束（数据完整性）

**数据完整性**是用来确保数据库中数据的正确性和可靠性。

数据完整性包括：

- 实体完整性：主键的取值**必须唯一**，并且**不能为空**。
- 域完整性：保证**数据的取值**在**有效**的范围内。
- 参照完整性：参照完整性是通过**主键**和**外键**来保证相关联的表间**数据保持一致**，避免因一个表的数据修改，而导致关联生效。



## 【软件设计师考试2006年5月上午试题48】

某数据库中有：

员工关系E（员工编号，姓名，部门）

产品关系P（产品编号，产品名称，型号，尺寸，颜色）；

仓库关系W（仓库编号，仓库名称，地址，负责人编号）；

库存关系I（仓库编号，产品编号和产品数量）

若要求仓库关系的负责人引用员工关系的员工编号，员工关系E的员工编号、仓库关系W的仓库编号和产品关系P的产品编号不能为空且惟一标识一个记录，并且仓库的地址不能为空，则依次要满足的完整性约束是（48）。

- (48) A. 实体完整性、参照完整性、用户定义完整性  
B. 参照完整性、实体完整性、用户定义完整性  
C. 用户定义完整性、实体完整性、参照完整性  
D. 实体完整性、用户定义完整性、参照完整性

## 【软件设计师考试2006年5月上午试题48】

在数据库系统中，数据的完整性约束的建立需要通过数据库管理系统提供的(51)语言来实现。

- (51) A. 数据定义  
B. 数据操作  
C. 数据查询  
D. 数据控制



## 2.4 数据操作

关系数据库的数据操作语言（DML）的语句分成查询语句和更新语句两大类。查询语句用于描述用户的各种检索要求；更新语句用于描述用户进行插入、删除、修改等操作。关于查询的理论称为“关系运算理论”。

关系查询语言根据其理论基础的不同分成两大类：

关系代数语言：查询操作是以集合操作为基础的DML语言。

关系演算语言：查询操作是以谓词演算为基础的DML语言。



- 关系代数
  - 关系代数的五个基本操作
    - 并、差、笛卡尔积、投影、选择
  - 关系代数的四个组合操作
    - 交、联接、自然连接、除法
- 关系演算
  - 元组关系演算
  - 域关系演算

- 关系代数
- 关系代数的五个基本操作

### (1) 并 (Union)

设关系R和S具有相同的模式，R和S的并是由属于R或属于S的元组构成的集合，记为 $R \cup S$ 。

形式定义如下：

$$R \cup S \equiv \{t \mid t \in R \vee t \in S\},$$

t是元组变量，R和S的元数相同。

R

A	B	C
a	b	c
d	a	f
c	b	d

S

A	B	C
b	g	a
d	a	f

RUS

A	B	C
a	b	c
d	a	f
c	b	d
b	g	a

## (2) 差 (Difference)

设关系R和S具有相同的模式，R和S的差是由属于R但不属于S的元组构成的集合，记为 $R - S$ 。

形式定义如下：

$R - S \equiv \{ t \mid t \in R \wedge t \notin S \}$ ，R和S的元数相同。

R

A	B	C
a	b	c
d	a	f
c	b	d

S

A	B	C
b	g	a
d	a	f

R - S

A	B	C
a	b	c
c	d	b

### (3) 笛卡尔积

设关系R和S的元数分别为r和s。定义R和S的笛卡尔积是一个 $(r+s)$ 元的元组集合，每个元组的前r个分量（属性值）来自R的一个元组，后s个分量来自R的一个元组，记为 $R \times S$ 。

形式定义如下：

$$R \times S \equiv \{ t \mid t = \langle t^r, t^s \rangle \wedge t^r \in R \wedge t^s \in S \}$$

若R有m个元组，S有n个元组，则 $R \times S$ 有 $m \times n$ 个元组。

R

A	B	C
a	b	c
d	a	f
c	b	d

S

A	B	C
b	g	a
d	a	f

 $R \times S$ 

R.A	R.B	R.C	S.A	S.B	S.C
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

#### (4) 投影 (Projection)

这个操作是对一个关系进行垂直分割，消去某些列，并重新安排列的顺序。

设关系  $R$  是  $k$  元关系， $R$  在其分量  $A_{i_1}, \dots, A_{i_m}$  ( $m \leq k$ ,  $i_1, \dots, i_m$  为 1 到  $k$  间的整数) 上的投影用  $\pi_{i_1, \dots, i_m}(R)$  表示，它是一个  $m$  元元组集合，

形式定义如下：

$$\pi_{i_1, \dots, i_m}(R) \equiv \{ t \mid t = \langle t_{i_1}, \dots, t_{i_m} \rangle \wedge \langle t_1, \dots, t_k \rangle \in R \}$$



例如， $\pi_{3,1}(R)$  表示关系R中取第1、3列，组成新的关系，新关系中第1列为R的第3列，新关系的第2列为R的第1列。

如果R的每列标上属性名，那么操作符 $\pi$ 的下标处也可以用属性名表示。例如，关系R(A, B, C)，那么 $\pi_{C,A}(R)$ 与 $\pi_{3,1}(R)$ 是等价的。

R		
A	B	C
a	b	c
d	a	f
c	b	d

S		
A	B	C
b	g	a
d	a	f

 $\pi_{C, A}(R)$ 

C	A
c	a
f	d
d	c



## (5) 选择 (Selection)

选择操作是根据某些条件对关系做水平分割，即选取符合条件的元组。条件可用命题公式（即计算机语言中的条件表达式） $F$ 表示。

$F$ 中有两种成分：运算对象和运算符

形式定义如下：

$$\sigma_F(R) = \{ t \mid t \in R \wedge F(t) = \text{true} \}$$

$\sigma$  为选择运算符， $\sigma_F(R)$  表示从  $R$  中挑选满足公式  $F$  为真的元组所构成的关系。

例如,  $\sigma_{2>'3'}(R)$  表示从R中挑选第2个分量值大于3的元组所构成的关系。

常量用引号括起来, 而属性序号或属性名不要用引号括起来。

R		
A	B	C
a	b	c
d	a	f
c	b	d

S		
A	B	C
b	g	a
d	a	f

$$\sigma_{B=v}(R)$$

A	B	C
a	b	c
c	b	d

- 关系代数的四个组合操作

- (1) 交 (intersection)

关系R和S的交是由属于R又属于S的元组构成的集合，记为 $R \cap S$ ，这里要求R和S定义在相同的模式上。

形式定义如下：

$$R \cap S \equiv \{t \mid t \in R \wedge t \in S\}, \text{ R和S的元数相同。}$$

## (2) 连接 (join)

连接有两种:  $\theta$  连接和F连接

### ① $\theta$ 连接 ( $\theta$ 是算术比较符)

$\theta$  连接是从关系R和S的笛卡儿积中选取属性间满足某一 $\theta$ 操作的元组,

$$R \bowtie_{i \theta j} S \equiv \{t \mid t = \langle t^r, t^s \rangle \wedge t^r \in R \wedge t^s \in S \wedge t^r_i \theta t^s_j \}$$

因此,  $\theta$  连接由笛卡尔积和选择操作组合而成。

$$R \bowtie_{i \theta j} S \equiv \sigma_{i \theta (r+j)} (R \times S)$$

## ② F连接 (F是公式)

F连接是从关系R和S的笛卡儿积中选取属性间满足某一公式F的元组，这里F是形为 $F_1 \wedge F_2 \wedge \dots \wedge F_n$ 的公式，每个 $F_p$ 是形为 $i \theta j$ 的式子，而i和j分别为关系R和S的第i、第j个分量的序号。



### (3) 自然连接 (natural join)

两个关系R和S的自然连接操作具体计算过程如下:

- ① 计算  $R \times S$  ;
- ② 设R和S的公共属性是  $A_1, \dots, A_K$ , 挑选  $R \times S$  中满足  $R.A_1=S.A_1, \dots, R.A_K=S.A_K$  的那些元组;
- ③ 去掉  $S.A_1, \dots, S.A_K$  这些列。

形式定义如下:

$$R \bowtie S \equiv \pi_{i_1, \dots, i_m} (\sigma_{R.A_1=S.A_1 \wedge \dots \wedge R.A_K=S.A_K} (R \times S)),$$

其中  $i_1, \dots, i_m$  为R和S的全部属性, 但公共属性只出现一次。

自然连接就是等值连接去掉重复列。

## 【软件设计师考试2007年11月上午试题543、55】

关系R、S如下图所示， $R \bowtie S$ 可由 (54)<sup>A</sup> 基本的关系运算组成， $R \bowtie S =$  (55)<sup>D</sup>。

(54) A.  $\pi$ ,  $\sigma$  和  $\times$

B.  $-$ ,  $\sigma$  和  $\times$

C.  $\cap$ ,  $\sigma$  和  $\times$

D.  $\pi$ ,  $\sigma$  和  $\cap$

A	B	C		A	C	D
a	b	c		a	c	d
b	a	d		d	f	g
c	d	e		b	d	g
d	f	g				
R				S		

(55) A.	A	B	C	B.	A	B	C	D
	a	b	c		a	b	c	d
	b	a	d		b	a	d	g
	c	d	e		d	f	g	g
C.	A	B	C	D.	A	B	C	D
	a	b	c		a	b	c	d
	b	a	d		b	a	d	g

#### (4) 除法 (division)

设关系R和S的元数分别为r和s (设 $r > s > 0$ )，那么 $R \div S$ 是一个 $(r-s)$ 元的元组的集合。 $(R \div S)$ 是满足下列条件的最大关系：其中每个元组t与S中每个元组u组成的新元组 $\langle t, u \rangle$ 必在关系R中。

$$R \div S \equiv \pi_{1, 2, \dots, r-s}(R) - \pi_{1, 2, \dots, r-s}((\pi_{1, 2, \dots, r-s}(R) \times S) - R)$$

**R**

A	B	C	D
a	b	c	d
a	b	e	f
a	b	d	e
b	c	e	f
e	d	c	d
e	d	e	f

**S**

C	D
c	d
e	f

 **$R \div S$** 

A	B
a	b
e	d

① 先列出R-S的属性

② 属性值A、B同时满足S中 (c, d) 和 (e, f) 的元组。

例：选修了所有课程的学生们的学号和姓名。

选修表

S#	C#
s1	c1
s1	c2
s2	c2
s3	c1
s3	c2
s4	c1

课程

C#
c1
c2

选修表 ÷ 课程

S#
s1
s3



例：检索所学课程包含学生s1所学课程的学生学号。

选修表

S#	C#	Grade
s1	c1	90
s1	c2	85
s2	c4	76
s3	c1	68
s3	c2	65
s4	c1	70
s4	c3	86

课程

C#	Cname
c1	数据库
c2	英语
c3	高数
c4	体育

S#
s1
<u>s3</u>

$$\pi_{S\#, C\#}(\text{选修表}) \div \pi_{C\#}(\sigma_{S\# = 's1'}(\text{选修表}))$$

## 【软件设计师考试2005年11月上午试题33、34】

在关系代数运算中，关系 S、SP和R如下表所示。若先 --- (33) ---，则可以从 S 和 SP 获得 R。其对应的关系表达式为 --- (34) ---。

S

部门号	部门名
010	家电部
021	百货部
030	食品部
035	五金部

SP

部门号	商品号	销售量
010	01020210	500
010	01020211	780
010	01020212	990
021	02110200	580
025	02520100	1290
035	03530311	4680

R

部门名	商品号	销售量
家电部	01020210	500
家电部	01020211	780
家电部	01020212	990



- (33) A. 对 S 进行选择运算，再与 S 进行自然连接运算  
 B. 对 S 进行选择运算，再与 SP 进行自然连接运算，最后进行投影运算  
 C. 对 S 和 SP 进行笛卡尔积运算，再对运算结果进行投影运算  
 D. 分别对 S 和 SP 进行投影运算，再对运算结果进行笛卡尔积运算

- (34) A.  $\sigma_{1='010'}(S) \bowtie SP$       B.  $\pi_{1,3,4}(\sigma_{1='010'}(S) \bowtie SP)$   
 C.  $\pi_{2,3,4}(\sigma_{1='010'}(S) \bowtie SP)$       D.  $\pi_{2,3,4}(S \times SP)$

如下的 SQL 语句可以查询销售总量大于 1000 的部门号。

Select 部门名

From S

Where 部门号 in

(Select 部门号

From SP

Group by --- (35) ---)

- (35) A. 部门号 where sum(销售量)>1000  
B. 部门号 having sum(销售量)>1000  
C. 商品号 where sum(销售量)>1000  
D. 商品号 having sum(销售量)>1000

## 【软件设计师考试2006年5月上午试题47】

设有关系R、S如下所示，则关系代数表达式 $R \div S$ 的结果集为 (47) C。

**R**

A	B	C
$a_1$	$b_1$	$c_1$
$a_1$	$b_2$	$c_1$
$a_2$	$b_2$	$c_2$

**S**

B	D
$b_1$	$d_1$
$b_2$	$d_1$

(47)

A.		B.		C.			D.	
A		A		A	C		A	C
$a_1$		$a_1$		$a_1$	$c_1$		$a_1$	$c_1$
$a_1$							$a_2$	$c_2$

- 关系演算

- 元组关系演算

在元组关系演算中，元组关系演算表达式简称为元组表达式，其一般形式为：

$$\{ t | P(t) \}$$

其中， $t$ 是元组变量，表示一个元数固定的元组； $P$ 是公式，在数理逻辑中也称为谓词，也就是计算机语言中的条件表达式。 $\{ t | P(t) \}$ 表示满足公式 $P$ 的所有元组 $t$ 的集合。

在元组表达式中，公式由原子公式组成。

原子公式（Atoms）有下列三种形式：

- ①  $R(s)$ ：s是R的一个元组。
- ②  $s[i] \theta u[j]$ ：元组s的第i个分量与元组u的第j个分量之间满足 $\theta$ 关系。
- ③  $s[i] \theta a$ 或 $a \theta u[j]$ ：a是常量。

在定义关系演算操作时，要用到“自由”和“约束”变量概念。在一个公式中，如果元组变量未用存在量词 $\exists$ 或全称量词 $\forall$ 符号定义，那么称为自由元组变量，否则称为约束元组变量。

例：已知关系R，S，给出R1-R5五个关系的元组关系演算表达式。

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

关系R

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

关系S

A	B	C
3	4	6
5	6	9

R1

$$R1 = \{ t \mid S(t) \wedge t[1] > 2 \}$$



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

关系 R

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

关系 S

A	B	C
4	5	6
7	8	9

R2

$$\mathbf{R2 = \{ t \mid R ( t ) \wedge \neg S ( t ) \}}$$

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

关系 R

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

关系 S

A	B	C
1	2	3
3	4	6

R3

$$R3 = \{ t \mid (\exists u) (S(t) \wedge R(u) \wedge t[3] < u[2]) \}$$





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

关系R

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

关系S

A	B	C
4	5	6
7	8	9

R4

$$R4 = \{ t \mid ( \forall u ) ( R(t) \wedge S(u) \wedge t[3] > u[1] ) \}$$

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

关系 R

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

关系 S

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

R5

$$R5 = \{ t \mid (\exists u) (\exists v) (R(u) \wedge S(v) \wedge u[1] > v[2] \wedge t[1] = u[2] \wedge t[2] = v[3] \wedge t[3] = u[1]) \}$$

关系代数表达式到元组表达式的转换:

$R \cup S$  可用  $\{ t \mid R(t) \vee S(t) \}$  表示;

$R - S$  可用  $\{ t \mid R(t) \wedge \neg S(t) \}$  表示;

$R \times S$  可用  $\{ t \mid (\exists u) (\exists v) (R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=u[3] \wedge t[4]=v[1] \wedge t[5]=v[2] \wedge t[6]=v[3]) \}$  表示。

投影操作是  $\pi_{2,3}(R)$ , 那么元组表达式可写成:

$\{ t \mid (\exists u) (R(u) \wedge t[1]=u[2] \wedge t[2]=u[3]) \}$

$\sigma_F(R)$  可用  $\{ t \mid R(t) \wedge F' \}$  表示,  $F'$  是  $F$  的等价表示形式。

例如  $\sigma_{2='d'}(R)$  可写成  $\{ t \mid (R(t) \wedge t[2]='d') \}$ 。

## 【软件设计师考试2005年5月上午试题36、37】

某数据库中有供应商关系S和零件关系P，其中；供应商关系模式S (Sno, Sname, Szip, City)，零件关系模式P (Pno, Pname, Color, Weight, City)，供应模式SP (Sno, Pno, Status, Qty)，要求一个供应商可以供应多种零件，而一种零件可由多个供应商供应。

查询供应了“红”色零件的供应商名、零件号和数量的元组演算表达式为：

$\{t \mid (\exists u) (\exists v) (\exists w) ( \underline{\quad (36) \quad} \wedge u[1]=v[1] \wedge v[2]=w[1] \wedge w[3]='$   
 $\text{红}' \wedge \underline{\quad (37) \quad} ) \}$

(36) A.  $S(u) \wedge SP(v) \wedge P(w)$

B.  $SP(u) \wedge S(v) \wedge P(w)$

C.  $P(u) \wedge SP(v) \wedge S(w)$

D.  $S(u) \wedge P(v) \wedge SP(w)$

(37) A.  $t[1]=u[1] \wedge t[2]=w[2] \wedge t[3]=v[4]$

B.  $t[1]=v[1] \wedge t[2]=u[2] \wedge t[3]=u[4]$

C.  $t[1]=w[1] \wedge t[2]=u[2] \wedge t[3]=v[4]$

D.  $t[1]=u[2] \wedge t[2]=v[2] \wedge t[3]=v[4]$

- 关系数据库SQL语言
  - SQL的数据定义
  - SQL的数据查询
  - SQL的数据更新

**【软件设计师考试2005年11月上午试题33、34】**

关系R，S如下图所示，元组演算表达式

$\{t \mid (\forall u) (R(t) \wedge S(u) \wedge t[3] > u[1])\}$  的结果为 (47)

A	B	C
1	2	3
4	5	6
7	8	9
10	11	12

R

A	B	C
3	7	11
4	5	6
5	9	13
6	10	14

S



(47) A.	A	B	C	B.	A	B	C
	1	2	3		3	7	11
	4	5	6		4	5	6
C.	A	B	C	D.	A	B	C
	7	8	9		5	9	13
	10	11	12		6	10	14



- SQL的数据定义——创建表

```
CREATE TABLE <表名>
( <列名1>    <数据类型>    <完整性约束条件1>,
  <列名2>    <数据类型>    <完整性约束条件2>,
  ..... ,
  <列名n>    <数据类型>    <完整性约束条件n>
)
```

## 涉及相应属性列的完整性约束条件:

- 主键约束: PRIMARY KEY
- 检查约束: CHECK
- 外键约束: FOREIGN KEY
- 唯一性约束: UNIQUE
- 是否为空值约束: NULL / NOT NULL
- 默认值: DEFAULT

例：使用T-SQL语言创建Student 表

Student ( Sno , Sname, Ssex )

**CREATE TABLE Student**

**(sno CHAR(10) PRIMARY KEY ,**

**sname CHAR(8) NOT NULL UNIQUE,**

**ssex CHAR(2) DEFAULT ('男')**

**)**

例：使用T-SQL语言创建Crouse表

Crouse ( cno, cname, credit )

**CREATE TABLE Course**

**( cno char(10) PRIMARY KEY,**

**cname char (8) NOT NULL UNIQUE,**

**credit int**

**)**

例：使用T-SQL语言创建SC表

SC ( sno, cno, score )

主键表名

```
CREATE TABLE SC
(  sno  CHAR(10)  FOREIGN KEY REFERENCES Student (sno),
   cno  CHAR(10)  FOREIGN KEY REFERENCES Course (cno) ,
   score INT      CHECK (score BETWEEN 0 AND 100) ,
   PRIMARY KEY (sno, cno)
)
```

主键



CREATE TABLE SC

```
( sno CHAR(10),  
  cno CHAR(10),  
  score INT,  
  PRIMARY KEY (sno, cno),  
  FOREIGN KEY (sno) REFERENCES Student (sno),  
  FOREIGN KEY (cno) REFERENCES Course (cno) ,  
  CHECK (score BETWEEN 0 AND 100)  
)
```

某数据库中有供应商关系S和零件关系P，其中；供应商关系模式S (Sno, Sname, Szip, City) 零件关系模式P (Pno, Pname, Color, Weight, City) 要求一个供应商可以供应多种零件，而一种零件可由多个供应商供应。请将下面的SQL语句空缺部分补充完整。

```
CREATE TABLE SP (Sno CHAR(5),  
                  Pno CHAR(6),  
                  Status CHAR(8),  
                  Qty NUMERIC(9),  
                  -- (33) -- (Sno, Pno),  
                  -- (34) -- (Sno),  
                  -- (35) -- (Pno));
```

供选择的答案:

(33) – (35)

A. FOREIGN KEY

B. PRIMARY KEY

C. FOREIGN KEY ( S<sub>no</sub> ) REFERENCES S

D. FOREIGN KEY ( P<sub>no</sub> ) REFERENCES P

(33) A (33) C (35) D



- SQL的数据查询

数据查询语句基本格式如下:

SELECT <查询项的列表>

FROM <表名>

WHERE <条件表达式>

## 6个聚合函数:

- SUM(列名): 求某一列的总和 (此列的值必须是数值型)
- AVG(列名): 求某一列的平均值 (此列的值必须是数值型)
- MIN(列名): 求某一列中的最小值
- MAX(列名): 求某一列中的最大值
- COUNT(列名): 传回一列中的非NULL值的个数
- COUNT(\*): 传回符合查询条件的元组的个数

例：查询男生的总人数，以及他们的平均年龄。

```
SELECT  COUNT(*), AVG(年龄)
FROM    学生
WHERE   性别= '男'
```

例：查询财管1班的所有学生的英语总成绩和均分。

```
SELECT  SUM(c1), AVG(c1)
```

```
FROM    成绩
```

```
WHERE   c# = 'c1'
```

## ORDER BY子句——对查询的结果进行排序

SELECT语句中可以使用ORDER BY子句对查询的结果进行排序，带ASC参数时为升序，带DESC参数时为降序，不待任何参数时为默认方式——升序。

SELECT <查询项的列表>

FROM <表名>

WHERE <条件表达式>

ORDER BY <排序表达式> [ASC | DESC]

例：查询全体学生的姓名和年龄，年龄按降序排列显示。

```
SELECT  姓名, 年龄  
FROM    学生  
ORDER BY 年龄  DESC
```

## GROUP BY 子句、HAVING 子句

——按条件分类统计

在SELECT语句中可以使用GROUP BY子句进行分类统计。GROUP BY子句可以将表达式的值相同的记录归为同一组，从而进行统计。

语法格式如下：

**GROUP BY** <分组表达式>

**HAVING** 子句指定组或聚合的搜索条件，只能和**SELECT**语句一起使用，通常和**GROUP BY**连用。

语法格式如下：

**HAVING** <组条件表达式>



例：查询女生每一年龄组有多少人。

```
SELECT  年龄, count ( * )
```

```
FROM    学生
```

```
WHERE   性别='女'
```

```
GROUP BY 年龄
```

例：查询女生每一年龄组超过10人的年龄。

```
SELECT  年龄,  
FROM    学生  
WHERE   性别='女'  
GROUP BY 年龄  
HAVING COUNT(*)>10
```

例：查询平均成绩低于60分的课程。

```
SELECT  课程号, AVG (成绩)
FROM    成绩
GROUP BY 课程号
HAVING  AVG (成绩) <60
```

例: 下面给一个关系模式

R ( TNAME, ADDRESS, C#, CNAME )

关系模式中的属性一次表示教师姓名, 教师地址, 课程编号, 课程名。分析该模式存在的问题。

TNAME	ADDRESS	C#	CNAME
t1	a1	c1	n1
t1	a1	c2	n2
t1	a1	c3	n3
t2	a2	c4	n4
t2	a2	c5	n2
t3	a3	c6	n4

有现实世界中的事实可知：

一个教师只有一个地址（户籍所在地）；

一个教师可教若干门课程；

每门课程只有一个教师任教。

即：TNAME  $\rightarrow$  ADDRESS

C#  $\rightarrow$  CNAME

C#  $\rightarrow$  TNAME

于是关系模式R的主键是 C#

虽然这个模式只有四个属性，但在使用过程中明显存在下列问题。

(1) 数据冗余。在上述关系中，如一名教师教多门课程，那么这位教师的地址就要重复多次。

(2) 更新异常。由于数据的冗余，在数据更新时会出现问题，例如：一个教师教三门课程，在关系中就会有三个元组，如果他的地址改变了，这三个元组中的地址都要改变，若有一个元组中的地址没有更改，就造成这个教师的地址不惟一，产生错误的信息。

(3) 插入异常。在关系方法中，每个关系必须用键值区分关系中的不同元组。如果新增加一名教师，尚未分配教学任务，那么要存储该教师的姓名和地址到关系中去时，在属性C#和CNAME上就没有值，在本例中C#是主键的一部分，键值为空的元组违反了实体完整性约束。

(4) 删除异常。与插入异常相反，删除操作会引起一些信息的丢失。如一个教师原来有教学任务，目前没有安排那么要把这个教师的所有元组都要删去，这样就把这个教师的姓名和地址信息也从数据库中删去了，这也是一种不合理的现象。



对于上述模式中存在的问题可采用分解的方法，将其分解成两个模式：

R1 ( TNAME, ADDRESS )

R2 ( TNAME, C#, CNAME )

那么分解后的关系模式的好坏，用什么标准来衡量呢？

—— 范式：满足特定要求得关系模式。



设有关系模式 $R(U)$ ,  $X$ 和 $Y$ 是属性集 $U$ 的子集,  $r$ 是 $R$ 任一具体关系, 如果对 $r$ 的任意两个元组 $t_1$ 和 $t_2$ , 都有 $t_1[X] = t_2[X]$ 导致 $t_1[Y] = t_2[Y]$ , 那么称 $X$ 函数决定 $Y$ 或 $Y$ 函数依赖 $X$ , 记为 $X \rightarrow Y$ ,  $X \rightarrow Y$ 为模式 $R$ 的一个函数依赖。

## 第一范式

如果关系模式R的每个关系r的属性值都是不可分的原子值，那么称R是第一范式（first normal form，简记为1NF）的模式。

满足1NF的关系称为规范化的关系，否则称为非规范化的关系。关系数据库研究的关系都是规范化的关系。例如关系模式R（NAME，ADDRESS，PHONE），如果一个人有两个电话号码（PHONE），那么在关系中至少要出现两个元组，以便存储这两个号码。

1NF是关系模式应具备的最起码的条件。在建立关系数据模型时，必须将非规范化形式规范化，

厂名↵	生产情况（每月）↵	
	产品名称↵	数量↵
F1↵	P1↵	300↵
	P2↵	200↵
	P3↵	400↵
	P4↵	200↵
	P5↵	100↵
	P6↵	100↵
F2↵	P1↵	300↵
	P2↵	400↵
F3↵	P2↵	200↵
F4↵	P2↵	200↵
	P3↵	300↵
	P4↵	400↵

厂名↵	产品名称↵	每月产量↵
F1↵	P1↵	300↵
F1↵	P2↵	200↵
F1↵	P3↵	400↵
F1↵	P4↵	200↵
F1↵	P5↵	100↵
F1↵	P6↵	100↵
F2↵	P1↵	300↵
F2↵	P2↵	400↵
F3↵	P2↵	200↵
F4 ↵	P2↵	200↵
F4↵	P3↵	300↵
F4↵	P4↵	400↵

## 第一范式



## 第二范式

如果关系模式R是1NF，且每个非主属性完全函数依赖于候选键，那么称R是第二范式（2NF）模式。如果数据库模式中每个关系模式都是2NF，则称数据库模式为2NF的数据库模式。

对于函数依赖 $W \rightarrow A$ ，如果存在 $X \sqsubset W$ 有 $X \rightarrow A$ 成立，那么称 $W \rightarrow A$ 是局部依赖（A局部依赖于W）；否则称 $W \rightarrow A$ 是完全依赖。

例：现在分析以下关系，它符合第一范式。假定职工号是每个职工的唯一标识，而工资完全由级别所确定。毕业时间由职工号和学历确定。

这表明：姓名、级别、工资等的取值依赖于职工号，而毕业时间要由职工号和学历两个属性才能唯一确定。

职工号	姓 名	级 别	工 资 (元)	学 历	毕业时间 (年)
001	张三	技 6	1500	中专	1983
001	张三	技 6	1500	大学	1988
001	张三	技 6	1500	研究生	1995
002	李四	技 8	1200	大学	1990



那么，在这个表中各个属性之间表示的相互依赖关系如图。



存在（姓名，级别，工资）对（职工号，学历）的局部依赖

因此，可以拆开原来关系为以下两个关系，使之成为第二范式。

**职工号为主关键字**

职工号	姓名	级别	工资（元）
001	张三	技 6	1500
002	李四	技 8	1200

**（职工号、学历）为主关键字**

职工号	学历	毕业时间（年）
001	中专	1983
001	大学	1988
001	研究生	1995
002	大学	1990

## 第三范式

如果关系模式R是1NF，且每个非主属性都不传递依赖于R的候选键，那么称R是第三范式

（3NF）的模式。如果数据库模式中每个关系模式都是3NF，则称其为3NF的数据库模式。

如果 $X \rightarrow Y$ ， $Y \rightarrow A$ ，且 $Y \not\rightarrow X$ 和  $A \notin Y$ ，那么称 $X \rightarrow A$ 是传递依赖（A传递依赖于X）。



**职工号为主关键字**

职工号	姓名	级别	工资（元）
001	张三	技6	1500
002	李四	技8	1200

由以上关系可以看到实际上可能很多人的级别都是技6级的，那么他们的工资都是1500元，如果把技6级的工资改为1800元，则所有技6级的工资都要改，只要一个人改错了，就会造成同一级工资不一样的错误。

原因就在于这些属性间存在着传递依赖关系：

职工号→级别，级别→工资 从而使：职工号→级别→工资

进一步的规范化就是要消去非主属性对主关键字的传递依赖性，变为第三范式。

现在只要把第二范式的关系恰当拆开为几个关系即可达目的。

职工号为主关键字

职工号	姓名	级别
001	张三	技 6
002	李四	技 8

级别为主关键字

级别	工资（元）
技 6	1500
技 8	1200

## 【软件设计师考试2004年11月上旬试题45、46】

建立一个供应商、零件数据库。其中“供应商”表S (Sno, Sname, Zip, City) 分别表示：供应商代码、供应商名、供应商邮编、供应商所在城市，其函数依赖为： $Sno \rightarrow (Sname, Zip, City)$ ， $Zip \rightarrow City$ 。“供应商”表S属于 (53) B。

(53) A. 1NF

B. 2NF

C. 3NF

D. BCNF

数据库模式设计原则：

关系模式R相对于函数依赖集分解成数据库模式  $\rho = \{R_1, R_2, \dots, R_k\}$ ，一般应具有下面四项特性。

- (1) P 中每个关系模式 上应有某种分时性质（3NF或BCNF）
- (2) 无损联接。
- (3) 保持函数的依赖集。

## 无损联接

设R是一个关系模式，F是R上的一个函数依赖集。R分解成数据库模式  $\rho = \{ R_1, \dots, R_k \}$ 。

如果对R中满足F的每一个关系r，都有

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \dots \bowtie \pi_{R_k}(r)$$

那么称分解  $\rho$  相对于F是“无损联接分解” 简称为“无损分解”，否则称为“损失分解”。

无损联接测试:

设  $\rho = \{ R_1, R_2 \}$  是关系模式  $R$  的一个分解,  $F$  是  $R$  上成立的 FD 集, 那么分解  $\rho$  相对于  $F$  是无损分解的充分必要条件是:

$$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$$

或

$$(R_1 \cap R_2) \rightarrow (R_2 - R_1)。$$

## 保持函数的依赖集

保持关系模式一个分解是等价的另一个重要条件是关系模式的函数依赖集在分解后仍在数据库模式中保持不变。

设  $\rho = \{ R_1, \dots, R_k \}$  是  $R$  的一个分解,  $F$  是  $R$  上的函数依赖, 如果有  $\bigcup \pi_{R_i}(F) = F$ , 那么称分解  $\rho$  保持函数依赖集  $F$ 。

## 【软件设计师考试2004年11月上旬试题45、46】

关系模式 $R(U, F)$ ，其中 $U = \{W, X, Y, Z\}$ ， $F = \{WX \rightarrow Y, W \rightarrow X, X \rightarrow Z, Y \rightarrow W\}$ 。关系模式 $R$ 的候选建是(45) A、(46) C是无损连接并保持函数依赖的分解。

(45) A.  $W$ 和 $Y$       B.  $WY$       C.  $WX$       D.  $WZ$

(46) A.  $p = \{R1(WY), R2(XZ)\}$

B.  $p = \{R1(WZ), R2(XY)\}$

C.  $p = \{R1(WXY), R2(XZ)\}$

D.  $p = \{R1(WX), R2(YZ)\}$



$U = \{W, X, Y, Z\}$ ,  $F = \{WX \rightarrow Y, W \rightarrow X, X \rightarrow Z, Y \rightarrow W\}$

求属性集U关于函数依赖F的属性闭包 $U^+$ 。

设  $W^+ = W$

找到左边被W包含的函数依赖关系： $W \rightarrow X$ ,

将  $W^+ = W \cup X = WX$

同理：找到左边被WX包含的函数依赖关系： $WX \rightarrow Y$ ,  $X \rightarrow Z$

将  $W^+ = WX \cup Y \cup Z = WXYZ$

所以 $W^+ = WXYZ$ ，即W能决定多有的属性，W为候选键

$U = \{W, X, Y, Z\}$  ,  $F = \{WX \rightarrow Y, W \rightarrow X, X \rightarrow Z, Y \rightarrow W\}$

# 软件设计师培训

—— 操作系统

### 3. 操作系统知识

大纲要求:

- 操作系统的内核（中断控制）、进程、线程概念
- 处理机管理（状态转换、共享与互斥、分时轮转、抢占、死锁）
- 存储管理（主存保护、动态连接分配、分段、分页、虚存）
- 设备管理（I/O控制、假脱机）
- 文件管理（文件目录、文件组织、存取方法、存取控制、恢复处理）
- 作业管理（作业调度、作业控制语言（JCL）、多道程序设计）
- 汉字处理，多媒体处理，人机界面
- 网络操作系统和嵌入式操作系统基础知识
- 操作系统的配置

## 3.1 操作系统的基本概念

### ● 操作系统的定义

能有效地组织和管理系统中的各种软、硬件资源，合理地组织计算机系统工作流程，控制程序的执行，并且向用户提供一个良好的工作环境和友好的接口。

硬件资源：包括CPU，存储器，输入/输出资源等物理设备。

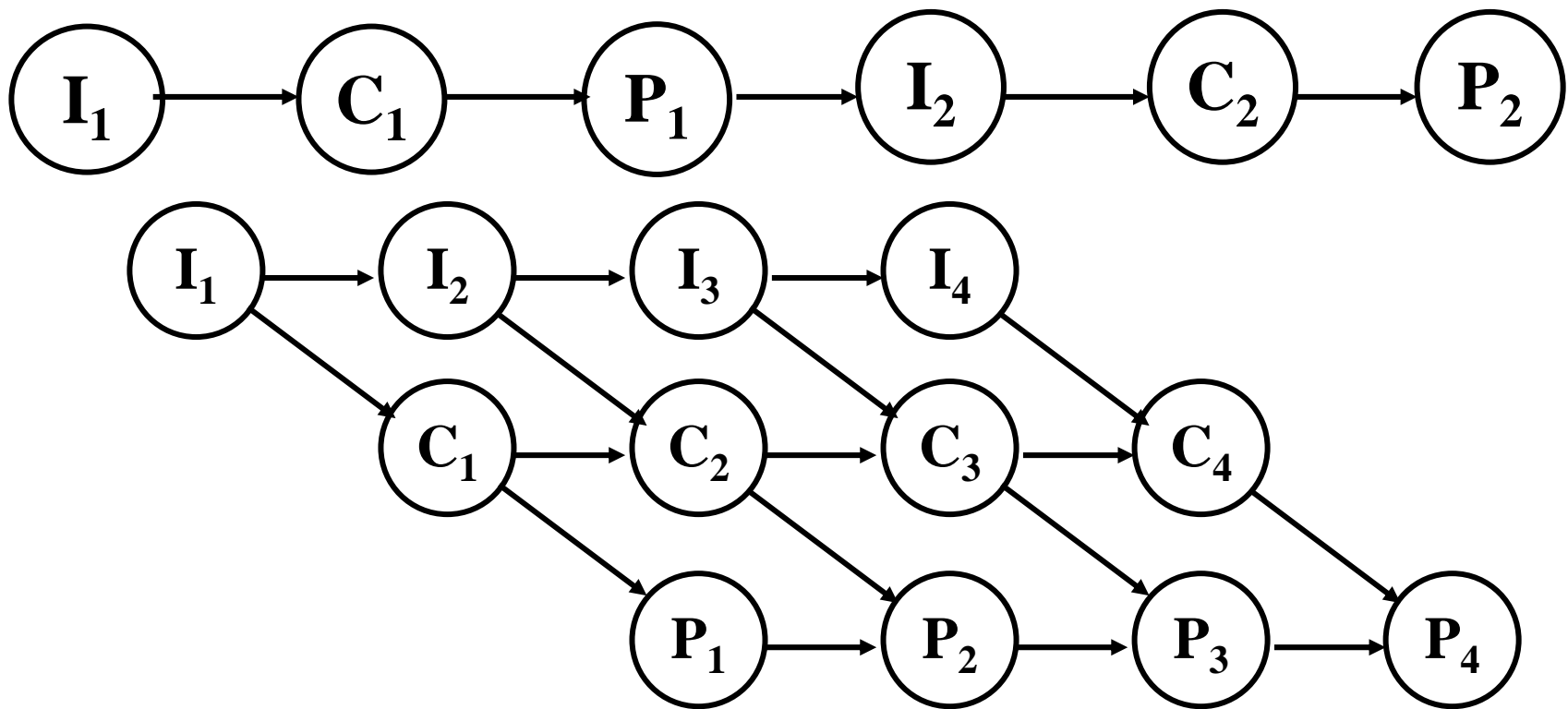
软件资源：以文件形式保存在存储器上的程序和数据等信息。

- 操作系统的2个重要作用:

- (1) 通过资源管理提高计算机系统的效率
- (2) 改善人机界面，向用户提供友好的工作环境

## ● 操作系统的4个特征

(1) 并发性：计算机系统存在着许多并发执行的活动



- (2) 共享性：系统中各个并发活动要共享计算机系统各种软、硬件资源。
- (3) 虚拟性：虚拟是操作系统中的重要特征，所谓虚拟就是把物理上的一台设备变成逻辑上的多台设备。
- (4) 不确定性(异步性)：指进程的执行顺序和执行时间及执行结果的不确定性。



## ● 操作系统的5大管理功能

- (1) 进程管理
- (2) 存储管理
- (3) 设备管理
- (4) 文件管理
- (5) 作业管理

## 3.2 进程管理

### ● 基本概念

**多道程序设计原理**：在计算机内存中同时存放几道相互独立的程序，它们和管理程序的控制下相互穿插地运行，共享CPU和外设等资源。

**程序**：具有特定功能的一组指令集合，它指出了处理器执行操作的步骤。

**进程**：进程是一个程序在一个数据集合上的一次执行。

## 程序和进程区别：

- (1) 程序是静态的，进程是动态的。
- (2) 进程与程序的对应关系：通过多次执行，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。
- (3) 进程是暂时的，程序的永久的：进程是一个状态变化的过程，程序可长久保存。
- (4) 进程与程序的组成不同：进程的组成包括程序、数据进程控制块（即进程状态信息）。

## 进程通常由三部分组成：

- (1) 程序：描述了进程所要完成的功能，是进程执行时不可修改的部分。
- (2) 数据集合：程序执行时所需要的数据和工作区，为一个进程专用，可修改。
- (3) 进程控制块PCB (Process Control Block)：包含了 进程的描述信息和控制信息，是进程的动态特性的集中反映。PCB包含以下几类信息：进程描述信息、进程控制信息、资源占用信息、CPU现场保护结构：

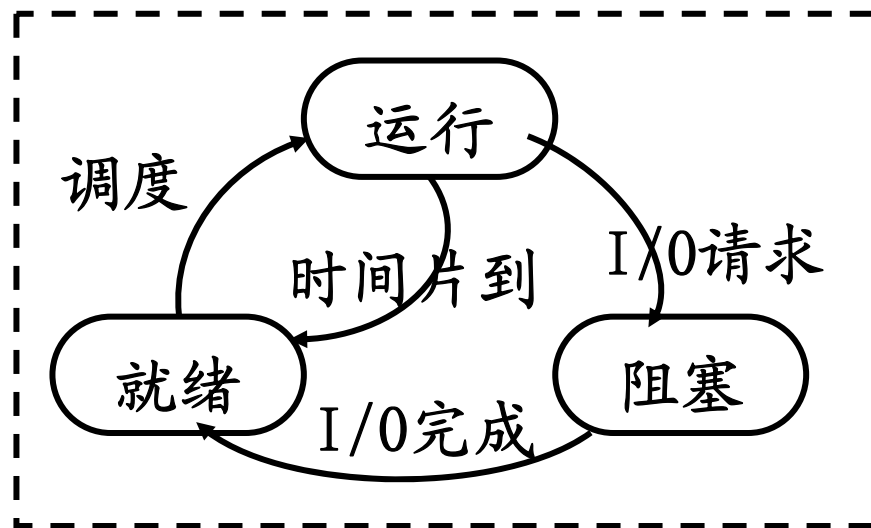
## ● 进程的基本状态及转换:

进程在生命期内处于且仅处于三种基本状态之一:

**运行态:** 当一个进程在处理机上运行时, 则称该进程处于运行状态。

**就绪态:** 一个进程获得了除处理机外的一切所需资源, 一旦得到处理机即可运行, 则称此进程处于就绪状态。

**阻塞态:** 当一个进程正在等待某一事件发生 (例如请求I/O而等待I/O完成等) 而暂时停止运行, 这时即使把处理机分配给进程也无法运行, 故称该进程处于阻塞状态。注意与就绪状态的不同在于即使处理机处于空闲状态也无法运行。



- ① 就绪→运行: 调度程序选择一个新的进程运行。
- ② 运行→就绪: 运行进程用完时间片被中断或在抢占调度方式中, 因为一高优先级进程进入就绪状态
- ③ 运行→阻塞: 进程发生I/O请求或等待某事件时
- ④ 阻塞→就绪: 当I/O完成或所等待的事件发生时

## ● 进程调度

在多道程序环境下，系统中有多多个进程同时运行。多个的进程竞争处理机，就要求系统提供进程调度功能，将处理机动态地分配给系统中的各个进程，使之能够协调一致的运行。

**进程调度程序：**主要任务是按照一定的调度算法从就绪队列中选取一个进程，把处理机分配给此进程使用。

## • 进程调度方式

- (1) 非抢占方式：在非抢占方式下，调度程序一旦把 CPU 分配给某一进程后便让它一直运行下去，直到进程完成或发生某事件而不能运行时，才将CPU分给其它进程。

这种调度方式通常用在批处理系统中。它的主要优点是简单、系统开销小。

- (2) 抢占方式：当一个进程正在执行时，系统可以基于某种策略剥夺CPU给其它进程。剥夺的原则有：优先权原则、短进程优先原则和时间片原则。

这种调度方式多用在分时系统和实时系统中，以便及时响应各进程的请求。



## • 进程调度算法

### (1) 先来先服务FCFS (先进先出调度算法, FIFO)

#### 【算法思想】：最简单的算法

- 按照进程进入就绪队列的先后次序，分派CPU；
- 当前进程占用CPU，直到执行完或阻塞，才出让CPU（非抢占方式）。
- 在进程唤醒后（如I/O完成），并不立即恢复执行，通常等到当前进程出让CPU。

#### 【特点】：

- 比较有利于长作业，而不利于短作业。
- 有利于CPU繁忙的作业，而不利于I/O繁忙的作业。

## (2) 短进程优先调度算法 (SJF, SPF)

【算法思想】：选择就绪队列中估计运行时间最短的进程投入运行。通常后来的短作业**不抢先**正在执行的作业。

【优点】：

- 比FCFS改善**平均周转时间**和**平均带权周转时间**，缩短作业的等待时间；
- 提高系统的**吞吐量**；

【缺点】：

- **对长作业非常不利**，可能长时间得不到执行；
- 未能依据作业的**紧迫程度**来划分执行的优先级；
- **难以准确估计作业（进程）的执行时间**，从而影响调度性能。

### (3) 优先权调度算法 (HPF—Highest Priority First)

【算法思想】：优先选择就绪队列中优先级最高的进程投入运行。分为：

- 非抢占式优先级算法：仅发生在进程放弃CPU。
- 抢占式优先级算法：可剥夺当前运行进程CPU。

#### 【优先权的类型】

- 静态优先级：在进程创建时指定优先级，在进程运行时优先数不变。
- 动态优先级：在进程创建时创立一个优先级，但在其生命周期内优先数可以动态变化。如等待时间长优先数可改变。

#### 【确定优先级的依据】

进程类型、对资源的需求、根据用户要求。

#### (4) 高响应比优先 (HRRN, Highest Response Ratio Next):

HRRN是FCFS和SJF的折衷算法, 响应比R用下式动态计算:

$$\text{响应比} R = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$

#### 【特点】:

- 等待时间相同要求服务的时间越短优先权越高, 有利于短作业。
- 要求服务时间相同, 等待时间越长优先权越高, 近似于先来先服务。
- 长作业的优先权会随等待时间加长而升高, 长作业也会得到执行。

## (5) 时间片轮转调度算法

【算法思想】：通过时间片轮转，提高进程并发性和响应时间特性，从而提高资源利用率。

- 将系统中所有的就绪进程按照FCFS原则，排成一个队列。
- 每次调度时将CPU分派给队首进程，让其执行一个时间片。时间片的长度从几个ms到几百ms。
- 在一个时间片结束时，发生时钟中断。
- 调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过CPU现场切换执行当前的队首进程。
- 进程可以未使用完一个时间片，就出让CPU（如阻塞）。

## (6) 多级反馈队列算法(多队列轮转法)

### 【算法思想】：

- 设置多个就绪队列，分别赋予不同的优先级，队列1的优先级最高，其他逐级降低。每队列分配不同的时间片，规定优先级越低则时间片越长。
- 新进程就绪后，先投入队列1的末尾，按FCFS算法调度。若一个时间片未能执行完，则降低投入到队列2的末尾；依此类推，降低到最后的队列，则按“时间片轮转”算法调度直到完成。

- 进程由于等待事件而放弃CPU后，进入等待队列，一旦等待的事件发生，则回到原来的就绪队列。
- 仅当较高优先级的队列为空，才调度较低优先级的队列中的进程执行。如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾。

## ● 进程同步和互斥

在计算机系统中，多个进程可以并发执行，因此进程间必然存在资源共享和相互合作的问题。



## 例：共享数据变量资源造成的错误

终端售票处理进程：

**PROCESS  $P_i$  ( $i=1,2,3,\dots,n$ )**    */\*  $P_i$ 为顾客服务的处理进程\*/*

**Begin**

    {按旅客订票要求找到 $A_j$ };    */\*  $A_j$ 为公共数据区\*/*

$R_i := A_j$ ;    */\*  $R_i$ 为各进程执行时所用的工作单元\*/*

    if  $R_i \geq 1$  then begin  $R_i := R_i - 1$ ;

$A_j := R_i$ ;

        {输出一张票};

    end

    else {输出“票已售完”};

**end**



**进程互斥**——是指当有若干进程都要使用某一资源时，任何时刻最多只允许一个进程去使用，其他要使用该资源的进程必须等待，直到占用资源者释放了该资源。

这样的资源称为称为互斥资源——打印机，共享变量等。

共享资源的互斥的使用就是限定并发进程互斥的进入相关临界区。

**临界区**——并发进程中与共享变量有关的程序段。

如例题中的临界区：

$R_i := A_j$ ; if  $R_i \geq 1$  then begin  $R_i := R_i - 1$ ;  $A_j := R_i$ ;

**PV操作**——就是一种实现相关临界区的管理，实现进程互斥的方法。

- PV操作——进程的互斥

PV操作由P操作和V操作组成，P操作和V操是两个在信号量S上进行的操作。定义如下：

Procedure P (S)

begin S:=S-1;

if  $S < 0$  then 则该进程进入等待队列;

end; {P}

Procedure V (S)

begin S:=S+1;

if  $S \leq 0$  then 唤醒一个等待队列中的进程进入就绪;

end; {V}

**begin S: semaphore**

**S: =1**

**cobegin**

**PROCESS  $P_i(i=1,2,3,\dots,n)$**

**{按旅客订票要求找到 $A_j$ };**

**$P(S)$**

**$R_i:=A_j$ ;**

**if  $R_i \geq 1$  then**

**begin  $R_i:=R_i-1$ ;**

**$A_j:=R_i$ ;**

**$V(S)$**

**{输出一张票};**

**end**

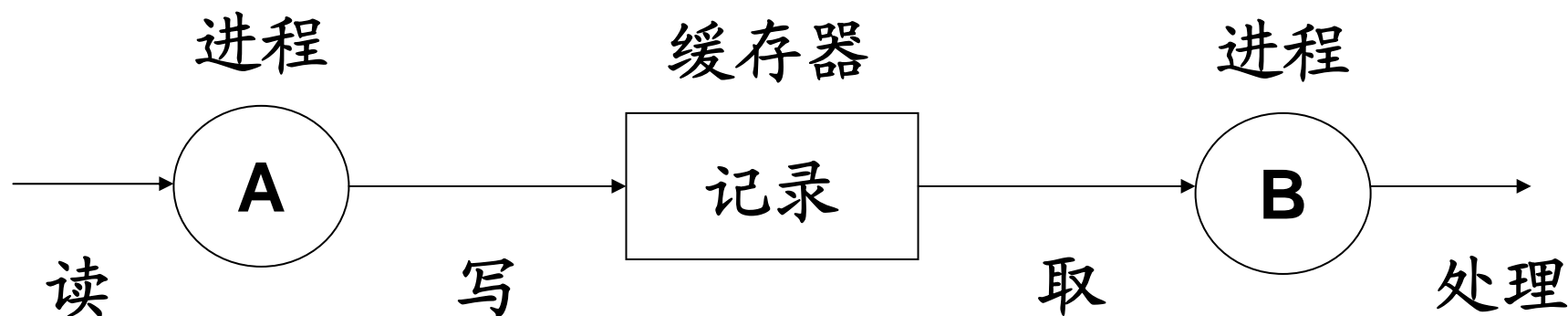
**else begin  $V(S)$**

**{输出“票已售完”};**

**coend**

**end**

## 例：共享缓存器资源造成的错误



(1) A的执行速度操作B的执行速度，造成缓存器中的数据还没拿走，A又读入新数据覆盖了原有数据。

(2) B的执行速度操作A的执行速度，B从缓存器取出一个记录并加工后，A还没有读入新数据，造成B在缓存器中重复取同一个记录加工。

**进程同步**—是指并发进程之间存在一种制约关系，一个进程的执行依赖另一个进程的消息，当一个进程没有得到另一个进程的消息时应等待，直到消息到达才被唤醒。

- PV操作——进程的互斥

1. 调用P操作测试消息是否到达。若消息尚未到达则 $S=0$ ，调用P(S)后，让调用者称为等待信号量S的状态；若消息已经存在则 $S \neq 0$ ，调用P(S)后进程不会成为等待状态而可继续执行。
2. 调用V操作发送消息。任何进程要向进程发送消息时可调用V操作。若调用V操作之前 $S=0$ ，表示消息产生且无等待消息进程，这是调用V(S)，执行 $S:=S+1$ 使 $S \neq 0$ ，意味着消息已存在。若调用V操作之前 $S < 0$ ，表示消息未产生前已有进程在等待消息，这是调用V(S)后释放一个等待消息者，即表示该进程等待的消息已经到达可以继续执行。



```
begin
Buffer: integer;
SP, SG: semaphore;
SP:=1; SG:=0;
Cobegin
    PROCESS Producer
begin
    L1: produce a product
    P ( SP ) ;
    Buffer:=product;
    V ( SG ) ;
    goto L1
End;
```

```
PROCESS Consumer
begin
    L2:  P ( SG ) ;
    take a product from buffer;
    V ( SP ) ;
    consume
    goto L2
End;
Coend;
End;
```

## 【软件设计师考试2004年5月上午试题23-24】

- 若有一个仓库，可以存放P1、P2两种产品，但是每次只能存放一种产品。要求：

①  $w = P1 \text{ 的数量} - P2 \text{ 的数量}$

②  $-i < w < k$  ( $i$ 、 $k$ 为正整数)

若用PV操作实现P1和P2产品的入库过程，至少需(23)个同步信号量及(24)个互斥信号量，其中，同步信号量的初值分别为(25)，互斥信号量的初值分别为(26)。

(23) A. 0      B. 1      **C. 2**      D. 3

(24) A. 0      **B. 1**      C. 2      D. 3

(25) A. 0      B.  $i, k, 0$       C.  $i, k$       **D.  $i-1, k-1$**

(26) **A. 1**      B.  $1, 1$       C.  $1, 1, 1$       D.  $i, k$

首先找题目中的互斥与同步，每次只能存放一种产品是互斥；①条件是同步题目中有说明：  
①  $w = p1 \text{ 的数量} - p2 \text{ 的数量}$  ②  $-i < w < k$  将公式①代入②里可得到另外两个新公式： $p1 \text{ 的数量} - p2 \text{ 的数量} < k$ ； $p2 \text{ 的数量} - p1 \text{ 的数量} < i$ 。取两个临界条件，当  $p2 = 0$  的时候， $p1$  最多取  $k - 1$ ；当  $p1 = 0$  的时候， $p2$  最多取  $i - 1$ ；

完整过程如下:

`mute=1; //互斥信号量`

`p1=k-1, p2=i-1; //同步信号量`

`p1`产品生产:

`p(p1)` //如果`p1=0`, 则说明仓库中不能存放该类产品了, 需要进行阻塞. 如果可以使用的, 就阻塞入库的进程

`p(mute)`

`p1`入库

`v(mute)`

`v(p2)`

=====p2产品生产

$p(p2)$

$p(mute)$

p2入库

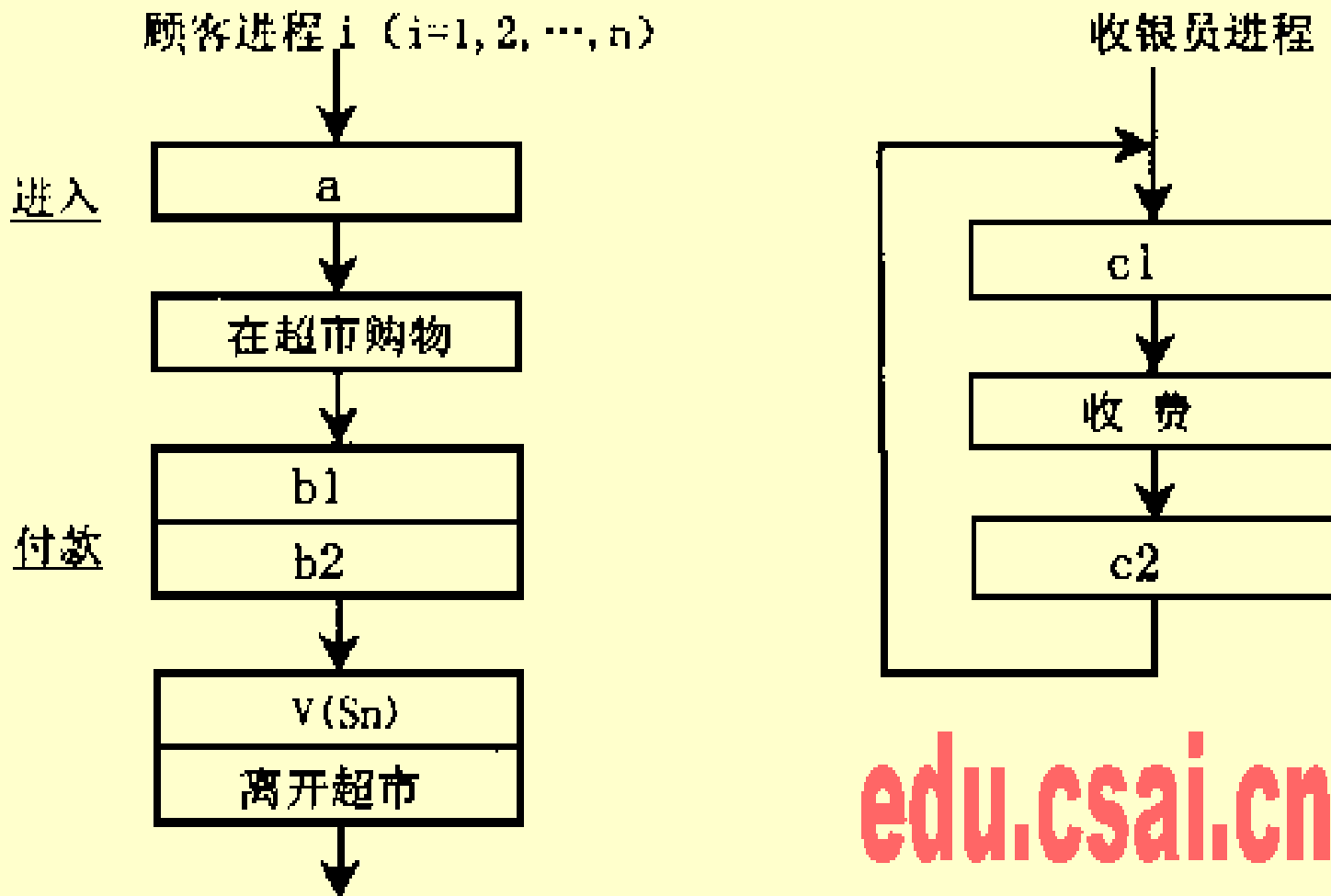
$v(mute)$

$v(p1)$

=====而且 $p(p2)$ 和 $p(mute)$ 的顺序不能互换, 假设 $p2=0$ , 则进入仓库临界资源之后, 由于 $p(p2)$ 的值 $<0$ , 则会造成该进程的死锁, 释放不出来仓库资源。

## 【软件设计师考试2005年5月上午试题23-24】

在某超市里有一个收银员，且同时最多允许有 $n$ 个顾客购物，我们可以将顾客和收银员看成是两类不同的进程，且工作流程如下图所示。为了利用PV操作正确地协调这两类进程之间的工作，设置了三个信号量 $S_1$ 、 $S_2$ 和 $S_n$ ，且初值分别为0、0和 $n$ 。这样图中的 $a$ 应填写-- (24) --，图中的 $b_1$ 、 $b_2$ 应分别填写-- (25) --，图中的 $c_1$ 、 $c_2$ 应分别填写-- (26) --。



- (24) A.  $P(S1)$  B.  $P(S2)$   
C.  $P(Sn)$  D.  $P(Sn)$ 、 $P(S1)$
- (25) A.  $P(Sn)$ 、 $V(S2)$  B.  $P(Sn)$ 、 $V(S1)$   
C.  $P(S2)$ 、 $V(S1)$  D.  $V(S1)$ 、 $P(S2)$
- (26) A.  $P(S1)$ 、 $V(S2)$  B.  $P(Sn)$ 、 $V(S1)$   
C.  $P(S2)$ 、 $V(S1)$  D.  $V(S1)$ 、 $P(S2)$



解析：这是一道考查PV操作的题，所以首先得弄清楚那些地方需要互斥、那些地方需要同步。题目中给出了两类进程：顾客进程与收银元进程，由于超市是顾客进程之间的公有资源，而且超市里限制最多允许有 $n$ 个顾客购物，所以要设置一个公有信号量 $S_n$ ，初值是 $n$ ，顾客进程在进入超市时要执行 $P(S_n)$ ，离开超市时要执行 $V(S_n)$ 操作。

顾客购物后要到收银员处付款，因此顾客进程与收银员进程之间是同步的关系，一次只允许一个顾客进程付款，整个超市只有一个收银员进程收费，所以需要为顾客进程设置一个私有信号量S2，为收银员进程设置一个私有信号量S1，由于开始时没有顾客去付款，收银员也没有收费，所以S1和S2的初值为0。当有顾客买完东西去付款时执行V(S1)，通知收银员进程有顾客付款，此时收银员进程执行P(S1)操作后就可进入收费，收费完成后收银元进程执行V(S2)，以通知顾客收费完毕，此时顾客执行P(S2)就可离开收银台，在离开超市时需执行V(Sn)，释放资源。

## ● 死锁

- 死锁的概念
- 死锁产生的原因
- 死锁的必要条件
- 处理死锁的基本方法

时刻

进程 $P_1$

进程 $P_2$

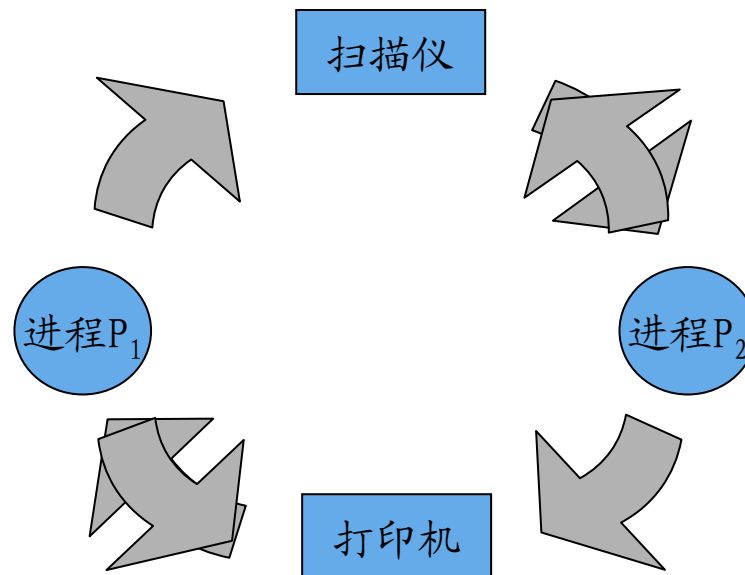
...

...

T4

...

request (打印机);



- 死锁的概念:

指多个进程因**竞争资源**而造成的一种僵局，若无外力作用，这些进程都将永远不能再向前推进。

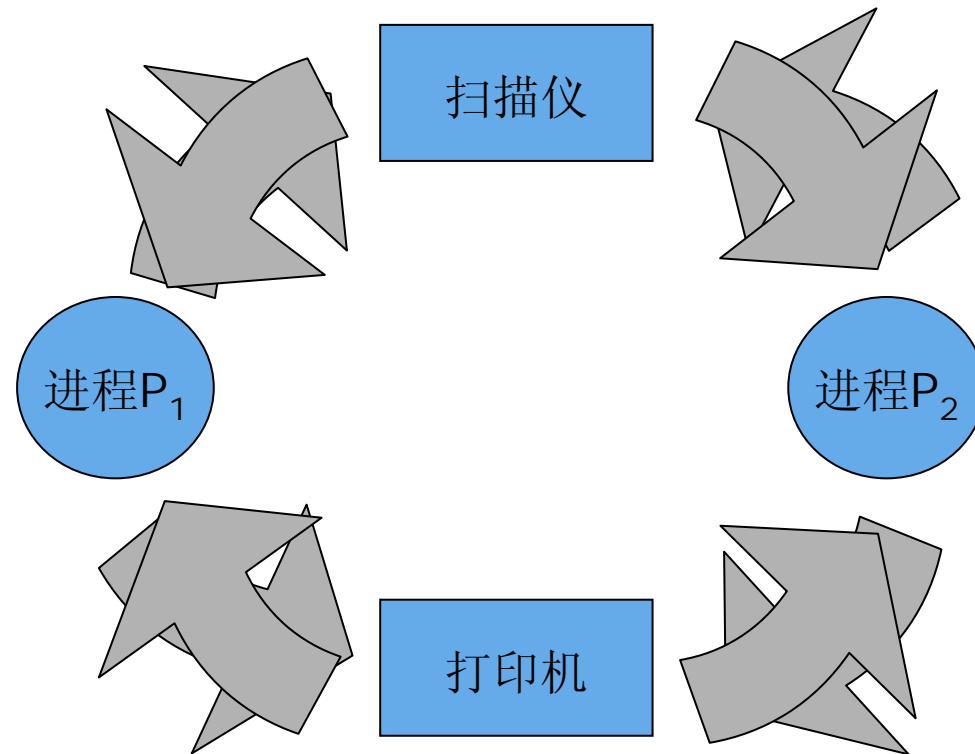
- 死锁产生的原因

- (1) 竞争资源

当系统中供多个进程所共享的资源,不足以同时满足它们的需要时,引起它们对资源的竞争而产生死锁。

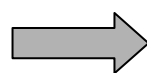
- (2) 进程推进顺序不当

进程在运行过程中,请求和释放资源的顺序不当,导致了进程的死锁。

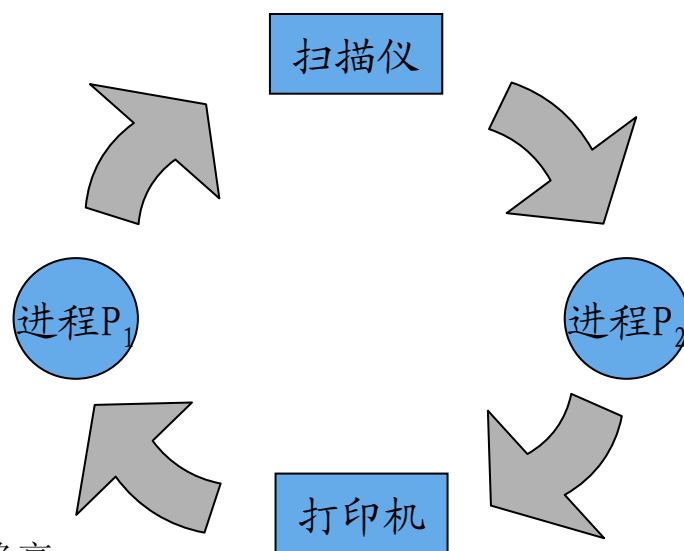


- 死锁产生的必要条件

- (1) 互斥使用资源
- (2) 占有并等待资源
- (3) 不可剥夺资源
- (4) 循环等待资源



死 锁





- 处理死锁的基本方法

- 预防死锁
- 避免死锁 —— 银行家算法
- 检测死锁
- 解除死锁

## • 避免死锁 —— 银行家算法

### 【基本步骤】

- (1) 进程首次申请时，测试该进程对资源的最大需求量，如果系统现存的资源可以满足它的最大需求量则按当前的申请量分配，否则推迟。
- (2) 当进程在执行中继续申请资源时，先测试该进程已占用的资源数与本次申请的资源数之和是否超过了该进程对资源的最大需求量，若超过了则拒绝。



(3) 若没有超过则再测试系统现存的资源能否满足该进程尚需的最大资源量，若能满足按当前的申请量分配资源，否则也要推迟分配。

(4) 这样，能保证在任何时刻至少有一个进程可以得到所需要的全部资源而执行到结束，执行结束后归还的资源加入到系统的剩余资源中，这些资源又至少可以满足一个进程的最大需求，最终保证了所有进程都能在有限的时间内得到需要的全部资源。

## 【基本思想】

银行家算法是通过动态地检测系统中资源分配情况和进程对资源的需求情况来决定如何分配资源的，在能确保系统处于安全状态时才能把资源分配给申请者，从而避免系统发生死锁。

## 【软件设计师考试2007年11月上午试题】

某系统中有四种互斥资源R1、R2、R3和R4，可用资源数分别为3、5、6和8。假设在T0时刻有P1、P2、P3和P4 四个进程，并且这些进程对资源的最大需求量和已分配资源数如下表所示，那么在T0时刻系统中R1、R2、R3和R4的剩余资源数分别为 （25）。如果从T0时刻开始进程按 （26） 顺序逐个调度执行，那么系统状态是安全的。



资源 请求	最大需求量				已分配资源数			
	R1	R2	R4	R3	R1	R2	R3	R4
P1	1	2	3	6	1	1	2	4
P2	1	1	2	2	0	1	2	2
P3	1	2	1	1	1	1	1	0
P4	1	1	2	3	1	1	1	1

系统可用资源数为: **3    5    6    8**

已分配资源数为: **3    4    6    7**

系统剩余资源数为: **0    1    0    1**



资源 请求	尚需资源				已分配资源数			
	R1	R2	R4	R3	R1	R2	R3	R4
P1	0	1	1	2	1	1	2	4
P2	1	0	0	0	0	1	2	2
P3	0	1	0	1	1	1	1	0
P4	0	0	1	2	1	1	1	1

系统剩余资源数为：0 1 0 1

P3, P3满足后归还资源, 系统剩余资源数为 1 2 1 1

P2, P2满足后归还资源, 系统剩余资源数为 1 3 3 3

P1, P1满足后归还资源, 系统剩余资源数为 2 4 5 7

P1, P1满足后归还资源, 系统剩余资源数为 3 5 6 8

(25) A. 3、5、6和8

B. 3、4、2和2

C. 0、1、2和1

D. 0、1、0和1

(26) A.  $P1 \rightarrow P2 \rightarrow P4 \rightarrow P3$

B.  $P2 \rightarrow P1 \rightarrow P4 \rightarrow P3$

C.  $P3 \rightarrow P2 \rightarrow P1 \rightarrow P4$

D.  $P4 \rightarrow P2 \rightarrow P3 \rightarrow P1$



## 【软件设计师考试2008年11月上午试题】

假设系统中有四类互斥资源R1、R2、R3和R4，可用资源数分别为9、6、3和3。在T0时刻系统中有 P1、P2、P3和P4四个进程，这些进程对资源的最大需求量和已分配资源数如下表所示。在 T0时刻系统剩余的可用资源数分别为(23)。如果 P1、P2、P3和P4进程按(24)序列执行，那么系统状态是安全的。



资源 请求	最大需求量				已分配资源数			
	R1	R2	R4	R3	R1	R2	R3	R4
P1	6	4	2	1	1	1	1	1
P2	2	2	2	1	2	1	1	1
P3	8	1	1	1	2	1	0	0
P4	2	2	1	1	1	2	1	1

系统可用资源数为： **9      6      3      3**

已分配资源数为： **6      5      3      3**

系统剩余资源数为： **3      1      0      0**



资源 请求	尚需资源				已分配资源数			
	R1	R2	R4	R3	R1	R2	R3	R4
P1	5	3	1	0	1	1	1	1
P2	0	1	1	0	2	1	1	1
P3	6	0	1	1	2	1	0	0
P4	1	0	0	0	1	2	1	1

系统剩余资源数为： **3    1    0    0**

- (23) A. 2、1、0和1      B. 3、1、0和0  
C. 3、1、1和1      D. 3、0、1和1
- (24) A.  $P1 \rightarrow P2 \rightarrow P4 \rightarrow P3$       B.  $P2 \rightarrow P1 \rightarrow P4 \rightarrow P3$   
C.  $P3 \rightarrow P4 \rightarrow P1 \rightarrow P2$       D.  $P4 \rightarrow P2 \rightarrow P1 \rightarrow P3$