

- 部署视图：部署视图把构件部署到一组物理节点上，表示软件到硬件的映射和分布结构。
- 用例视图：用例视图是最基本的需求分析模型。

另外，UML 还允许在一定的阶段隐藏模型的某些元素，遗漏某些元素，可不保证模型的完整性，但模型逐步地要达到完整和一致。

#### 4. 面向对象分析

OOA 的基本任务是运用 OO 方法，对问题域进行分析和理解，正确认识其中的事物及它们之间的关系，找出描述问题域和系统功能所需的类和对象，定义它们的属性和职责以及它们之间所形成的各种联系。最终产生一个符合用户需求，并能直接反映问题域和系统功能的 OOA 模型及其详细说明。OOA 模型独立于具体实现，即不考虑与系统具体实现有关的因素，这也是 OOA 和 OOD 的区别之所在。OOA 的任务是“做什么”，OOD 的任务是“怎么做”。

面向对象分析阶段的核心工作是建立系统的用例模型与分析模型。

##### 1) 用例模型

结构化分析（Structured Analysis, SA）方法采用功能分解的方式来描述系统功能，在这种表达方式中，系统功能被分解到各个功能模块中，通过描述细分的系统模块的功能来达到描述整个系统功能的目的。采用 SA 方法来描述系统需求，很容易混淆需求和设计的界限，这样的描述实际上已经包含了部分的设计在内。因此，系统分析师常常感到迷惑，不知道系统需求应该详细到何种程度。一个极端的做法就是将需求详细到概要设计，因为这样的需求描述既包含了外部需求也包含了内部设计。SA 方法的另一个缺点是分割了各项系统功能的应用环境，从各项功能项入手，很难了解到这些功能项如何相互关联来实现一个完整的系统服务。

从用户的角度来看，他们并不想了解系统的内部结构和设计，他们所关心的是系统所能提供的服务，这就是用例方法的基本思想。用例方法是一种需求合成技术，先获取需求并记录下来，然后从这些零散的要求和期望中进行整理与提炼，从而建立用例模型。在 OOA 方法中，构建用例模型一般需要经历四个阶段，分别是识别参与者、合并需求获得用例、细化用例描述和调整用例模型，如表 5-4 所示。其中前三个阶段是必须的。

表 5-4 构建用例模型的阶段

阶段	说明
识别参与者	参与者是与系统交互的所有事物，该角色不仅可以由人承担，还可以是其他系统和硬件设备，甚至是系统时钟。参与者一定在系统之外，不是系统的一部分。可以通过下列问题来帮助系统分析师发现系统的参与者：谁使用这个系统，谁安装这个系统，谁启动这个系统，谁维护这个系统，谁关闭这个系统，哪些（其他的）系统使用这个系统，谁从这个系统获取信息，谁为这个系统提供信息，是否有事情自动在预计的时间发生
合并需求获得用例	参与者都找到之后，接下来就是仔细地检查参与者，为每一个参与者确定用例。①要将获取到的需求分配给与其相关的参与者，以便可以针对每个参与者进行工作，而无遗漏；②在合并之前，要明确为什么合并，知道了合并的目的，才可能选择正确的合并操作；③将识别到的参与者和合并生成的用例，通过用例图的形式整理出来，以获得用例模型的框架

(续表)

阶段	说明
细化用例描述	用例建模的主要工作是书写用例规约(Use Case Specification),而不是画图。用例模板为一个给定项目的所有人员定义了用例规约的结果,其内容至少包括用例名、参与者、目标、前置条件、事件流(基本事件流和扩展事件流)和后置条件等,其他的还可以包括非功能需求和用例优先级等。
调整用例模型	在建立了初步的用例模型后,还可以利用用例之间的关系来调整用例模型。用例之间的关系主要有包含、扩展和泛化。利用这些关系,把一些公共的信息抽取出来,以便于复用,使得用例模型更易于维护。 <ul style="list-style-type: none"> <li>● 包含关系:当可以从两个或两个以上的用例中提取公共行为时,应该使用包含关系来表示它们。其中这个提取出来的公用用例称为抽象用例,而把原始用例称为基本用例或基础用例</li> <li>● 扩展关系:如果一个用例明显地混合了两种或两种以上不同场景,即根据情况可能发生多种分支,则可以将这个用例分为一个基本用例和一个或多个扩展用例,这样使描述可能更加清晰</li> <li>● 泛化关系:当多个用例共同拥有一种类似的结构和行为的时候,可以将它们的共性抽象成为父用例,其他的用例作为泛化关系中的子用例。在用例的泛化关系中,子用例是父用例的一种特殊形式,子用例继承了父用例所有的结构、行为和关系</li> </ul>

## 2) 分析模型

前文从用户的观点对系统进行了用例建模,但捕获了用例并不意味着分析的结束,还要对需求进行深入分析,获取关于问题域本质内容的分析模型。分析模型描述系统的基本逻辑结构,展示对象和类如何组成系统(静态模型),以及它们如何保持通信,实现系统行为(动态模型)。

为了使模型独立于具体的开发语言,系统分析师需要把注意力集中在概念性问题上而不是软件技术问题上,这些技术的起点就是领域模型。领域模型又称为概念模型或简称为域模型,也就是找到那些代表事物与概念的对象,即概念类。概念类可以从用例模型中获得灵感,经过完善将形成分析模型中的分析类。每一个用例对应一个类图,描述参与这个用例实现的所有概念类,而用例的实现主要通过交互图来表示。例如,用例的事件流会对应产生一个顺序图,描述相关对象如何通过合作来完成整个事件流,复杂的备选事件流也可以产生一个或多个顺序图。所有这些图的集合就构成了系统的分析模型。

建立分析模型的过程大致包括定义概念类,确定类之间的关系,为类添加职责,建立交互图等,其中有学者将前三个步骤统称为类-责任-协作者(Class-Responsibility-Collaborator,CRC)建模。类之间的主要关系有关联、依赖、泛化、聚合、组合和实现等,它们在UML中的表示方式,如表5-5所示。

表5-5 类之间的关系

关系	图例	描述
关联		关联提供了不同类的对象之间的结构关系,它在一段时间内将多个类的实例连接在一起。关联体现的是对象实例之间的关系,而不表示两个类之间的关系。其余的关系涉及类元自身的描述,而不是它们的实例。对于关联关系的描述,可以使用关联名称、角色、多重性和导向性来说明

(续表)

关系	图例	描述
依赖		两个类 A 和 B，如果 B 的变化可能会引起 A 的变化，则称类 A 依赖于类 B。依赖可以由各种原因引起，例如，一个类向另一个类发送消息，一个类是另一个类的数据成员，一个类是另一个类的某个操作参数等
泛化		泛化关系描述了一般事物与该事物中的特殊种类之间的关系，也就是父类与子类之间的关系。继承关系是泛化关系的反关系，也就是说，子类继承了父类，而父类则是子类的泛化
共享聚集		共享聚集关系通常简称为聚合关系，它表示类之间的整体与部分的关系，其含义是“部分”可能同时属于多个“整体”，“部分”与“整体”的生命周期可以不相同。例如，汽车和车轮就是聚合关系，车子坏了，车轮还可以用；车轮坏了，可以再换一个新的
组合聚集		组合聚集关系通常简称为组合关系，它也是表示类之间的整体与部分的关系。与聚合关系的区别在于，组合关系中的“部分”只能属于一个“整体”，“部分”与“整体”的生命周期相同，“部分”随着“整体”的创建而创建，也随着“整体”的消亡而消亡。例如，一个公司包含多个部门，它们之间的关系就是组合关系。公司一旦倒闭，也就没有部门了
实现		实现关系将说明和实现联系起来。接口是对行为而非实现的说明，而类中则包含了实现的结构。一个或多个类可以实现一个接口，而每个类分别实现接口中的操作

### 5.1.3 软件设计

软件设计是需求分析的延伸与拓展。需求分析阶段解决“做什么”的问题，而软件设计阶段解决“怎么做”的问题。同时，它也是系统实施的基础，为系统实施工作做好铺垫。合理的软件设计方案既可以保证系统的质量，也可以提高开发效率，确保系统实施工作的顺利进行。从方法上来说，软件设计分为结构化设计与面向对象设计。

#### 1. 结构化设计

结构化设计（Structured Design, SD）是一种面向数据流的方法，它以 SRS 和 SA 阶段所产生的 DFD 和数据字典等文档为基础，是一个自顶向下、逐步求精和模块化的过程。SD 方法的基本思想是将软件设计成由相对独立且具有单一功能的模块组成的结构，分为概要设计和详细设计两个阶段，其中概要设计又称为总体结构设计，它是开发过程中很关键的一步，其主要任务是将系统的功能需求分配给软件模块，确定每个模块的功能和调用关系，形成软件的模块结构图，即系统结构图。在概要设计中，将系统开发的总任务分解成许多个基本的、具体的任务，而为每个具体任务选择适当的技术手段和处理方法的过程称为详细设计。根据任务的不同，详细设计又可分为多种，例如，输入 / 输出设计、处理流程设计、数据存储设计、用户界面设计、安全性和可靠性设计等。

在 SD 中，需要遵循一个基本的原则：高内聚，低耦合。内聚表示模块内部各成分之间的联系程度，是从功能角度来度量模块内的联系，一个好的内聚模块应当恰好做目标单一的一件事情；耦合表示模块之间联系的程度。紧密耦合表示模块之间联系非常强，松散耦合表示模块之间联系比较弱，非耦合则表示模块之间无任何联系，是完全独立的。

## 2. 面向对象设计

面向对象设计（OOD）是 OOA 方法的延续，其基本思想包括抽象、封装和可扩展性，其中可扩展性主要通过继承和多态来实现。在 OOD 中，数据结构和在数据结构上定义的操作算法封装在一个对象之中。由于现实世界中的事物都可以抽象出对象的集合，所以 OOD 方法是一种更接近现实世界、更自然的软件设计方法。

OOD 的主要任务是对类和对象进行设计，包括类的属性、方法以及类与类之间的关系。OOD 的结果就是设计模型。对于 OOD 而言，在支持可维护性的同时，提高软件的可复用性是一个至关重要的问题，如何同时提高软件的可维护性和可复用性，是 OOD 需要解决的核心问题之一。在 OOD 中，可维护性的复用是以设计原则为基础的。常用的 OOD 原则包括：

- 单职原则：设计功能单一的类。本原则与结构化方法的高内聚原则是一致的。
- 开闭原则：对扩展开放，对修改封闭。
- 李氏替换原则：子类可以替换父类。
- 依赖倒置原则：要依赖于抽象，而不是具体实现；要针对接口编程，不要针对实现编程。
- 接口隔离原则：使用多个专门的接口比使用单一的总接口要好。
- 组合重用原则：要尽量使用组合，而不是继承关系达到重用目的。
- 迪米特原则（最少知识法则）：一个对象应当对其他对象有尽可能少的了解。本原则与结构化方法的低耦合原则是一致的。

## 3. 设计模式

设计模式是前人经验的总结，它使人们可以方便地复用成功的软件设计。当人们在特定的环境下遇到特定类型的问题，采用他人已使用过的一些成功的解决方案，一方面可以降低分析、设计和实现的难度，另一方面可以使系统具有更好的可复用性和灵活性。设计模式包含模式名称、问题、目的、解决方案、效果、实例代码和相关设计模式等基本要素。

根据处理范围不同，设计模式可分为类模式和对象模式。类模式处理类和子类之间的关系，这些关系通过继承建立，在编译时刻就被确定下来，属于静态关系；对象模式处理对象之间的关系，这些关系在运行时刻变化，更具动态性。根据目的和用途不同，设计模式可分为创建型（Creational）模式、结构型（Structural）模式和行为型（Behavioral）模式三种：①创建型模式主要用于创建对象，包括工厂方法模式、抽象工厂模式、原型模式、单例模式和建造者模式等；②结构型模式主要用于处理类或对象的组合，包括适配器模式、桥接模式、组合模式、装饰模式、外观模式、享元模式和代理模式等；③行为型模式主要用于描述类或对象的交互以及职责的分配，包括职责链模式、命令模式、解释器模式、迭代器模式、中介者模式、备忘录模式、

观察者模式、状态模式、策略模式、模板方法模式、访问者模式等。

## 5.1.4 软件实现

### 1. 软件配置管理

软件配置管理通过标识产品的组成元素、管理和控制变更、验证、记录和报告配置信息，来控制产品的演进和完整性。软件配置管理与软件质量保证活动密切相关，可以帮助达成软件质量保证目标。

软件配置管理活动包括软件配置管理计划、软件配置标识、软件配置控制、软件配置状态记录、软件配置审计、软件发布管理与交付等活动。

软件配置管理计划的制订需要了解组织结构环境和组织单元之间的联系，明确软件配置控制任务。软件配置标识活动识别要控制的配置项，并为这些配置项及其版本建立基线。软件配置控制关注的是管理软件生命周期中的变更。软件配置状态记录标识、收集、维护并报告配置管理的配置状态信息。软件配置审计是独立评价软件产品和过程是否遵从已有的规则、标准、指南、计划和流程而进行的活动。软件发布管理和交付通常需要创建特定的交付版本，完成此任务的关键是软件库。

### 2. 软件编码

目前，人和计算机通信仍需使用人工设计的语言，也就是程序设计语言。所谓编码就是把软件设计的结果翻译成计算机可以“理解和识别”的形式——用某种程序设计语言书写的程序。作为软件工程的一个步骤，编码是设计的自然结果，因此，程序的质量主要取决于软件设计的质量。但是，程序设计语言的特性和编码途径也会对程序的可靠性、可读性、可测试性和可维护性产生深远的影响。

(1) 程序设计语言。编码的目的是实现人和计算机的通信，指挥计算机按人的意志正确工作。程序设计语言是人和计算机通信的最基本工具，程序设计语言的特性不可避免地会影响人的思维和解决问题的方式，会影响人和计算机通信的方式和质量，也会影响其他人阅读和理解程序的难易程度。因此，编码之前的一项重要工作就是选择一种恰当的程序设计语言。

(2) 程序设计风格。在软件生存周期中，开发者经常要阅读程序。特别是在软件测试阶段和维护阶段，编写程序的人员与参与测试、维护的人员要阅读程序。因此，阅读程序是软件开发和维护过程中的一个重要组成部分，而且读程序的时间比写程序的时间还要多。20世纪70年代初，有人提出在编写程序时，应使程序具有良好的风格。程序设计风格包括4个方面：源程序文档化、数据说明、语句结构和输入/输出方法。应尽量从编码原则的角度提高程序的可读性，改善程序的质量。

(3) 程序复杂性度量。经过详细设计后，每个模块的内容都已非常具体，因此可以使用软件设计的基本原理和概念仔细衡量它们的质量。但是，这种衡量毕竟只能是定性的，人们希望能进一步定量度量软件的性质。定量度量程序复杂程度的方法很有价值，把程序的复杂度乘以适当的常数即可估算出软件中故障的数量及软件开发时的工作量。定量度量的结构可以用于比

较两个不同设计或两种不同算法的优劣；程序的定量的复杂程度可以作为模块规模的精确限度。

(4) 编码效率。编码效率主要包括：①程序效率。程序的效率是指程序的执行速度及程序所需占用的内存空间。一般说来，任何对效率无重要改善，且对程序的简单性、可读性和正确性不利的程序设计方法都是不可取的。②算法效率。源程序的效率与详细设计阶段确定的算法的效率直接相关。在详细设计翻译转换成源程序代码后，算法效率反映为程序的执行速度和存储容量的要求。③存储效率。存储容量对软件设计和编码的制约很大。因此要选择可生成较短目标代码且存储压缩性能优良的编译程序，有时需要采用汇编程序，通过程序员富有创造性的努力，提高软件的时间与空间效率。提高存储效率的关键是程序的简单化。④ I/O 效率。输入 / 输出可分为两种类型：一种是面向人（操作员）的输入 / 输出；另一种是面向设备的输入 / 输出。如果操作员能够十分方便、简单地输入数据，或者能够十分直观、一目了然地了解输出信息，则可以说面向人的输入 / 输出是高效的。至于面向设备的输入 / 输出，主要考虑设备本身的性能特性。

### 3. 软件测试

软件测试是在将软件交付给客户之前所必须完成的重要步骤。目前，软件的正确性证明尚未得到根本的解决，软件测试仍是发现软件错误（缺陷）的主要手段。根据国家标准 GB/T 15532《计算机软件测试规范》，软件测试的目的是验证软件是否满足软件开发合同或项目开发计划、系统 / 子系统设计文档、SRS、软件设计说明和软件产品说明等规定的软件质量要求。通过测试发现软件缺陷，为软件产品的质量测量和评价提供依据。

软件测试方法可分为静态测试和动态测试。①静态测试是指被测试程序不在机器上运行，而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。静态测试包括对文档的静态测试和对代码的静态测试。对文档的静态测试主要以检查单的形式进行，而对代码的静态测试一般采用桌前检查（Desk Checking）、代码走查和代码审查。经验表明，使用这种方法能够有效地发现 30% ~ 70% 的逻辑设计和编码错误。②动态测试是指在计算机上实际运行程序进行软件测试，一般采用白盒测试和黑盒测试方法。白盒测试也称为结构测试，主要用于软件单元测试中。它的主要思想是，将程序看作是一个透明的白盒，测试人员完全清楚程序的结构和处理算法，按照程序内部逻辑结构设计测试用例，检测程序中的主要执行通路是否都能按预定要求正确工作。白盒测试方法主要有控制流测试、数据流测试和程序变异测试等。另外，使用静态测试的方法也可以实现白盒测试。例如，使用人工检查代码的方法来检查代码的逻辑问题，也属于白盒测试的范畴。白盒测试方法中，最常用的技术是逻辑覆盖，即使用测试数据运行被测程序，考查对程序逻辑的覆盖程度。主要的覆盖标准有语句覆盖、判定覆盖、条件覆盖、条件 / 判定覆盖、条件组合覆盖、修正的条件 / 判定覆盖和路径覆盖等。黑盒测试也称为功能测试，主要用于集成测试、确认测试和系统测试中。黑盒测试将程序看作是一个不透明的黑盒，完全不考虑（或不了解）程序的内部结构和处理算法，而只检查程序功能是否能按照 SRS 的要求正常使用，程序是否能适当地接收输入数据并产生正确的输出信息，程序运行过程中能否保持外部信息（例如，文件和数据库等）的完整性等。黑盒测试根据 SRS 所规定的功能来设计测试用例，一般包括等价类划分、边界值分析、判定表、因果图、状态图、随机测试、猜错法和正交试验法等。

## 5.1.5 部署交付

软件开发完成后，必须部署在最终用户的正式运行环境，交付给最终用户使用，才能为用户创造价值。传统的软件工程不包括软件部署与交付，但不断增长的软件复杂度和部署所面临的风险，迫使人们开始关注软件部署。软件部署是一个复杂过程，包括从开发商发放产品，到应用者在他们的计算机上实际安装并维护应用的所有活动。这些活动包括软件打包、安装、配置、测试、集成和更新等。同时，需求和市场的不断变化导致软件的部署和交付不再是一个一劳永逸的过程，而是一个持续不断的过程，伴随在整个软件的开发过程中。

### 1. 软件部署与交付

软件部署与交付是软件生命周期中的一个重要环节，属于软件开发的后期活动，即通过配置、安装和激活等活动来保障软件制品的后续运行。部署技术影响着整个软件过程的运行效率和成本投入，软件系统部署的管理代价占到整个软件管理开销的大部分。其中软件配置过程极大地影响着软件部署结果的正确性，应用系统的配置是整个部署过程中的主要错误来源。据 Standish Group 的统计，软件的缺陷所造成的损失，相当大的部分是由于部署与交付失败所引起的，可见软件部署与交付工作的重要意义。

目前，部署与交付常存在：分支冗余导致合并困难；缺陷过多导致阻塞测试；开发环境、测试环境、部署环境不统一导致的未知错误；代码提交版本混乱无法回溯；等待上线周期过长；项目部署操作复杂经常失败；上线之后出现问题需要紧急回滚；架构设计不合理导致发生错误之后无法准确定位等困境。

### 2. 持续交付

为解决部署与交付常存在的问题，持续交付应运而生，持续交付是一系列开发实践方法，用来确保让代码能够快速、安全地部署到生产环境中。持续交付是一个完全自动化的过程，当业务开发完成的时候，可以做到一键部署。持续交付提供了一套更为完善的解决传统软件开发流程的方案，主要体现在：

- 在需求阶段，抛弃了传统的需求文档的方式，使用便于开发人员理解的用户故事；
- 在开发测试阶段，做到持续集成，让测试人员尽早进入项目开始测试；
- 在运维阶段，打通开发和运维之间的通路，保持开发环境和运维环境的统一。

持续交付具备的优势主要包括：

- 持续交付能够有效缩短提交代码到正式部署上线的时间，降低部署风险；
- 持续交付能够自动、快速地提供反馈，及时发现和修复缺陷；
- 持续交付让软件在整个生命周期内都处于可部署的状态；
- 持续交付能够简化部署步骤，使软件版本更加清晰；
- 持续交付能够让交付过程成为一种可靠的、可预期的、可视化的过程。

在评价互联网公司的软件交付能力的时候，通常会使用两个指标：

- 仅涉及一行代码的改动需要花费多少时间才能部署上线，这也是核心指标；
- 开发团队是否在以一种可重复、可靠的方式执行软件交付。

目前，国内外的主流互联网组织的部署周期都以分钟为单位，互联网巨头组织单日的部署频率都在 8000 次以上，部分组织达 20000 次以上。高频率的部署代表着能够更快更好地响应客户需求。

### 3. 持续部署

对于持续交付整体来说，持续部署非常重要。

#### 1) 持续部署方案

容器技术目前是部署中最流行的技术，常用的持续部署方案有 Kubernetes+Docker 和 Matrix 系统两种。容器技术一经推出就被广泛地接受和应用，主要原因是对比传统的虚拟机技术优点主要有：

- 容器技术上手简单，轻量级架构，体积很小；
- 容器技术的集合性更好，能更容易对环境和软件进行打包复制和发布；
- 容器技术的引入为软件的部署带来了前所未有的改进，不但解决了复制和部署麻烦的问题，还能更精准地将环境中的各种依赖进行完整的打包。

#### 2) 部署原则

在持续部署管理的时候，需要遵循一定的原则，主要包括：

- 部署包全部来自统一的存储库；
- 所有的环境使用相同的部署方式；
- 所有的环境使用相同的部署脚本；
- 部署流程编排阶梯式晋级，即在部署过程中需要设置多个检查点，一旦发生问题可以有序地进行回滚操作；
- 整体部署由运维人员执行；
- 仅通过流水线改变生产环境，防止配置漂移；
- 不可变服务器；
- 部署方式采用蓝绿部署或金丝雀部署。

#### 3) 部署层次

部署层次的设置对于部署管理来说也是非常重要的。首先要明确部署的目的并不是部署一个可工作的软件，而是部署一套可正常运行的环境。完整的镜像部署包括三个环节：Build—Ship—Run。

- Build：跟传统的编译类似，将软件编译形成 RPM 包或者 Jar 包；
- Ship：则是将所需的第三方依赖和第三方插件安装到环境中；
- Run：就是在不同的地方启动整套环境。

制作完成部署包之后，每次需要变更软件或者第三方依赖以及插件升级的时候，不需要重新打包，直接更新部署包即可。

#### 4) 不可变服务器

不可变服务器是一种部署模式，是指除了更新和安装补丁程序以外，不对服务器进行任何更改。不可变服务器是技术逐步演化的结果。在早期阶段，软件的部署是在物理机上进行的，

每一台服务器的网络、存储、软件环境都是不同的，物理机的不稳定让环境重构变得异常困难。后来逐渐发展为虚拟机部署，在虚拟机上借助流程化的部署能较好地构建软件环境，但是第三方依赖库的重构不稳定为整体部署带来了困难。现阶段使用容器部署不但继承和优化了虚拟机部署的优点，而且很好地解决了第三方依赖库的重构问题，容器部署就像一个集装箱，直接把所有需要的内容全部打包并进行复制和部署。

### 5) 蓝绿部署和金丝雀部署

在部署原则中提到两大部署方式为蓝绿部署和金丝雀部署。蓝绿部署是指在部署的时候准备新旧两个部署版本，通过域名解析切换的方式将用户使用环境切换到新版本中，当出现问题的时候，可以快速地将用户环境切回旧版本，并对新版本进行修复和调整。金丝雀部署是指当有新版本发布的时候，先让少量用户使用新版本，并且观察新版本是否存在。如果出现问题，就及时处理并重新发布；如果一切正常，就稳步地将新版本适配给所有的用户。

## 4. 部署与交付的新趋势

持续集成、持续交付和持续部署的出现及流行反映了新的软件开发模式与发展趋势，主要表现为：

- 工作职责和人员分工的转变：软件开发人员运用自动化开发工具进行持续集成，进一步将交付和部署扩展，而原来的手工运维工作也逐渐被分派到了开发人员的手里。运维人员的工作也从重复枯燥的手工作业转化为开发自动化的部署脚本，并逐步并入开发人员的行列之中。
- 大数据和云计算基础设施的普及进一步给部署带来新的飞跃：云计算的出现使得计算机本身也可以进行自动化创建和回收，这种环境管理的范畴将得到进一步扩充。部署和运维工作也会脱离具体的机器和机房，可以在远端进行，部署能力和灵活性出现了质的飞跃。
- 研发运维的融合：减轻运维的压力，把运维和研发融合在一起。

### 5.1.6 过程管理

软件过程能力是组织基于软件过程、技术、资源和人员能力达成业务目标的综合能力。包括治理能力、开发与交付能力、管理与支持能力、组织管理能力等方面。软件过程能力成熟度是指组织在提升软件产品开发能力或软件服务能力过程中，各个发展阶段的软件能力成熟度。常见的软件过程管理方法和实践包括国际常用的能力成熟度模型集成（Capability Maturity Model Integration, CMMI，详见本书 20.5.1 节）和中国电子工业标准化技术协会发布的 T/CESA 1159《软件过程能力成熟度模型》（Software Process Capability Maturity Model）团体标准，简称 CSMM。

#### 1. 成熟度模型

CSMM 定义的软件过程能力成熟度模型旨在通过提升组织的软件开发能力帮助顾客提升软件的业务价值。该模型借鉴吸收了软件工程、项目管理、产品管理、组织治理、质量管理、卓越绩效管理、精益软件开发等领域的优秀实践，为组织提供改进和评估软件过程能力的一个成

熟度模型，其层次结构如图 5-1 所示。

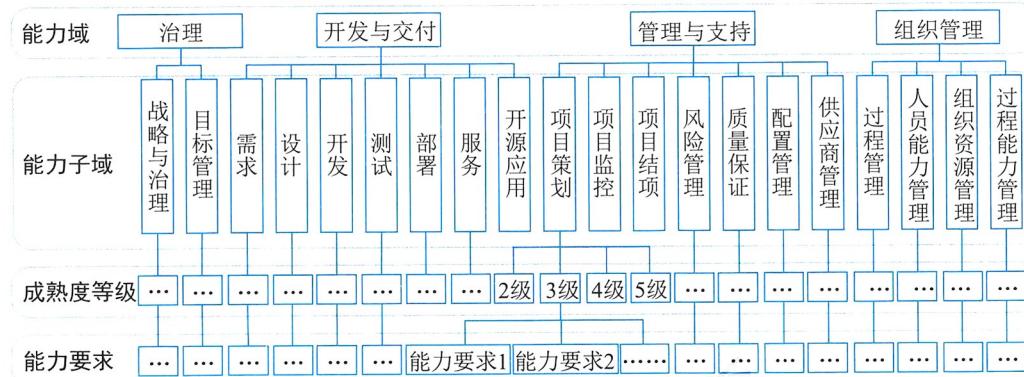


图 5-1 软件过程能力成熟度层次结构

CSMM 模型由 4 个能力域、20 个能力子域、161 个能力要求组成：

- 治理：包括战略与治理、目标管理能力子域，用于确定组织的战略、产品的方向、组织的业务目标，并确保目标的实现。
- 开发与交付：包括需求、设计、开发、测试、部署、服务、开源应用能力子域，这些能力子域确保通过软件工程过程交付满足需求的软件，为顾客与利益干系人增加价值。
- 管理与支持：包括项目策划、项目监控、项目结项、质量保证、风险管理、配置管理、供应商管理能力子域，这些能力子域覆盖了软件开发项目的全过程，以确保软件项目能够按照既定的成本、进度和质量交付，能够满足顾客与利益干系人的要求。
- 组织管理：包括过程管理、人员能力管理、组织资源管理、过程能力管理能力子域，对软件组织能力进行综合管理。

## 2. 成熟度等级

按照软件过程能力的成熟度水平由低到高演进发展的形势，CSMM 定义了 5 个等级，高等级是在低等级充分实施的基础之上进行。成熟度等级的总体特征如表 5-6 所示。

表 5-6 成熟度等级的总体特征

等级	结果特征	行为特征
1 级：初 始级	软件过程和结果具 有不确定性	<ul style="list-style-type: none"> <li>● 能实现初步的软件交付和项目管理活动</li> <li>● 项目没有完整的管理规范，依赖于个人的主动性和能力</li> </ul>
2 级：项 目规范级	项目基本可按计划 实现预期的结果	<ul style="list-style-type: none"> <li>● 项目依据选择和定义管理规范，执行软件开发和管理的基础过程</li> <li>● 组织按照一定的规范，为项目活动提供了支持保障工作</li> </ul>
3 级：组 织改进级	在组织范围内能够 稳定地实现预期的 项目目标	<ul style="list-style-type: none"> <li>● 在 2 级充分实施的基础之上进行持续改进</li> <li>● 依据组织的业务目标、管理要求以及外部监管需求，建立并持续改进组织标准过程和过程资产</li> <li>● 项目根据自身特征，依据组织标准过程和过程资产，实现项目目标，并贡献过程资产</li> </ul>

(续表)

等级	结果特征	行为特征
4 级：量化提升级	在组织范围内能够量化地管理和实现预期的组织和项目目标	<ul style="list-style-type: none"> <li>● 在3级充分实施的基础上使用统计分析技术进行管理</li> <li>● 组织层面认识到能力改进的重要性，了解软件能力在业务目标实现、绩效提升等方面的重要作用，在制定业务战略时可获得项目数据的支持</li> <li>● 组织和项目使用统计分析技术建立了量化的质量与过程绩效目标，支持组织业务目标的实现</li> <li>● 建立了过程绩效基线与过程绩效模型</li> <li>● 采用有效的数据分析技术，分析关键软件过程的能力，预测结果，识别和解决目标实现的问题以达成目标</li> <li>● 应用先进实践，提升软件过程效率或质量</li> </ul>
5 级：创新引领级	通过技术和管理的创新，实现组织业务目标的持续提升，引领行业发展	<ul style="list-style-type: none"> <li>● 在4级充分实施的基础上进行优化革新</li> <li>● 通过软件过程的创新提升组织竞争力</li> <li>● 能够使用创新的手段实现软件过程能力的持续提升，支持组织业务目标的达成</li> <li>● 能将组织自身软件能力建设的经验作为行业最佳案例进行推广</li> </ul>

能力域的等级要求如表 5-7 所示。

表 5-7 能力域与成熟度等级的对应关系

		战略与治理	目标管理	需求	设计	开发	测试	部署	服务	开源应用	项目策划	项目监控	项目结项	风险管理	质量保证	配置管理	供应商管理	过程管理	人员能力管理	组织资源管理	过程能力管理
成熟度等级	5	5																5		5	
	4	4	4							4	4	4	4					4	4	4	
	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
分类		治理						开发与交付						管理与支持						组织管理	

## 5.2 数据工程

数据工程是信息系统的基础工程。围绕数据的生命周期，规范数据从产生到应用的全过程，目标是为信息系统的运行提供可靠的数据保障和服务，为信息系统之间的数据共享提供安全、高效的支撑环境，为信息系统实现互连、互通、互操作提供有力的数据支撑。它是实现这些目标的一系列技术、方法和工程建设活动的总称。数据工程的主要研究内容包括数据建模、数据

标准化、数据运维、数据开发利用和数据安全等理论和技术。

### 5.2.1 数据建模

数据建模是对现实世界中具体的人、物、活动和概念进行抽象、表示和处理，变成计算机可处理的数据，也就是把现实世界中的数据从现实世界抽象到信息世界和计算机世界。数据建模主要研究如何运用关系数据库设计理论，利用数据建模工具，建立既能正确反映客观世界，又便于计算机处理的数据模型。

#### 1. 数据模型

根据模型应用目的不同，可以将数据模型划分为三类：概念模型、逻辑模型和物理模型。

##### 1) 概念模型

概念模型也称信息模型，它是按用户的观点来对数据和信息建模，也就是说，把现实世界中的客观对象抽象为某一种信息结构，这种信息结构不依赖于具体的计算机系统，也不对应某个具体的DBMS，它是概念级别的模型。概念模型的基本元素如表5-8所示。

表5-8 概念模型基本元素说明

基本元素	说明
实体	客观存在的并可以相互区分的事物称为实例，而同一类型实例的抽象称为实体，如学生实体（学号、系名、住处、课程、成绩）、教师实体（工作证号、姓名、系名、教研室、职称）。实体是同一类型实例的共同抽象，不再与某个具体的实例对应。相比较而言，实例是具体的，而实体则是抽象的
属性	实体的特性称之为属性。学生实体的属性包括学号、系名、住处、课程、成绩等，教师实体的属性包括工作证号、姓名、系名、教研室、职称等
域	属性的取值范围称为该属性的域。例如，性别的域是集合{"男", "女"}。域的元素必须是相同的数据类型
键	能唯一标识每个实例的一个属性或几个属性的组合称为键。一个实例集中有很多个实例，需要有一个标识能够唯一地识别每一个实例，这个标识就是键
关联	在现实世界中，客观事物之间是相互关系的，这种相互关系在数据模型中表现为关联。实体之间的关联包括一对一、一对多和多对多三种

通常对概念模型要求有：

- 概念模型是对现实世界的抽象和概括，它应该真实、充分地反映现实世界中事物和事物之间的联系，有丰富的语义表达能力，能表达用户的各种需求；
- 概念模型应简洁、明晰、独立于机器、容易理解，方便数据库设计人员与用户交换意见，使用户能够积极参与数据库的设计工作；
- 概念模型应易于变动。当应用环境和应用要求改变时，容易修改和补充概念模型；
- 概念模型应容易向关系、层次或网状等各种数据模型转换，易于从概念模型导出与DBMS相关的逻辑模型。

## 2) 逻辑模型

逻辑模型是在概念模型的基础上确定模型的数据结构，目前主要的数据结构有层次模型、网状模型、关系模型、面向对象模型和对象关系模型。其中，关系模型成为目前最重要的一种逻辑数据模型。

关系模型的基本元素包括关系、关系的属性、视图等。关系模型是在概念模型的基础上构建的，因此关系模型的基本元素与概念模型中的基本元素存在一定的对应关系，见表 5-9。

表 5-9 关系模型与概念模型的对应关系

概念模型	关系模型	说明
实体	关系	概念模型中的实体转换为关系模型的关系
属性	属性	概念模型中的属性转换为关系模型的属性
联系	关系 . 外键	概念模型中的联系有可能转换为关系模型的新关系，被参照关系的主键转化为参照关系的外键
	视图	关系模型中的视图在概念模型中没有元素与之对应，它是按照查询条件从现有关系或视图中抽取若干属性组合而成

关系数据模型的数据操作主要包括查询、插入、删除和更新数据，这些操作必须满足关系的完整性约束条件。关系的完整性约束包括三大类型：实体完整性、参照完整性和用户定义的完整性。其中，实体完整性、参照完整性是关系模型必须满足的完整性约束条件，用户定义的完整性是应用领域需要遵照的约束条件，体现了具体领域中的语义约束。

## 3) 物理模型

物理数据模型是在逻辑数据模型的基础上，考虑各种具体的技术实现因素，进行数据库体系结构设计，真正实现数据在数据库中的存放。物理数据模型的内容包括确定所有的表和列，定义外键用于确定表之间的关系，基于性能的需求可能进行反规范化处理等内容。在物理实现上的考虑，可能会导致物理数据模型和逻辑数据模型有较大的不同。物理数据模型的目标是如何用数据库模式来实现逻辑数据模型，以及真正地保存数据。物理模型的基本元素包括表、字段、视图、索引、存储过程、触发器等，其中表、字段和视图等元素与逻辑模型中基本元素有一定的对应关系。

## 2. 数据建模过程

通常来说，数据建模过程包括数据需求分析、概念模型设计、逻辑模型设计和物理模型设计等过程。

(1) 数据需求分析。简单地说，数据需求分析就是分析用户对数据的需要和要求。数据需求分析是数据建模的起点，数据需求掌握的准确程度将直接影响后续阶段数据模型的质量。数据需求分析通常不是单独进行的，而是融合在整个系统需求分析的过程之中。开展需求分析时，首先要调查清楚用户的实际要求，与用户充分沟通，形成共识，然后再分析和表达这些要求与

共识，最后将需求表达的结果反馈给用户，并得到用户的确认。数据需求分析采用数据流图作为工具，描述系统中数据的流动和变化，强调数据流和处理过程。

(2) 概念模型设计。经过需求分析阶段的充分调查，得到了用户数据开发利用需求，但是这些应用需求还是现实世界的具体需求，应该首先把它们抽象为信息世界的结构，下一步才能更好地、更准确地用某个DBMS来实现用户的这些需求。将需求分析得到结果抽象为概念模型的过程就是概念模型设计，其任务是确定实体和数据及其关联。

(3) 逻辑模型设计。概念模型独立于机器，更抽象，从而更加稳定，但是为了能够在具体的DBMS上实现用户的需求，还必须在概念模型的基础上进行逻辑模型的设计。由于现在的DBMS普遍都采用关系模型结构，因此逻辑模型设计主要指关系模型结构的设计。关系模型由一组关系模式组成，一个关系模式就是一张二维表，逻辑模型设计的任务就是将概念模型中实体、属性和关联转换为关系模型结构中的关系模式。

(4) 物理模型设计。经过概念模型设计和逻辑模型设计，数据模型设计的核心工作基本完成，如果要将数据模型转换为真正的数据库结构，还需要针对具体的DBMS进行物理模型设计，使数据模型走向数据存储应用环节。物理模型考虑的主要问题包括命名、确定字段类型和编写必要的存储过程与触发器等。

## 5.2.2 数据标准化

数据标准化是实现数据共享的基础。数据标准化主要为复杂的信息表达、分类和定位建立相应的原则和规范，使其简单化、结构化和标准化，从而实现信息的可理解、可比较和可共享，为信息在异构系统之间实现语义互操作提供基础支撑。数据标准化的主要内容包括元数据标准化、数据元标准化、数据模式标准化、数据分类与编码标准化和数据标准化管理。

### 1. 元数据标准化

元数据最简单的定义是：元数据是关于数据的数据（Data About Data）。在信息界，元数据被定义为提供关于信息资源或数据的一种结构化数据，是对信息资源的结构化描述。其实质是用于描述信息资源或数据的内容、覆盖范围、质量、管理方式、数据的所有者、数据的提供方式等有关的信息。

#### 1) 信息对象

元数据描述的对象可以是单一的全文、目录、图像、数值型数据以及多媒体（声音、动态图像）等，也可以是多个单一资源组成的资源集合，或是这些资源的生产、加工、使用、管理、技术处理、保存等过程及其过程中产生的参数的描述。

#### 2) 元数据体系

根据信息对象从产生到服务的生命周期、元数据描述和管理内容的不同以及元数据作用的不同，元数据可以分为多种类型。从最基本的资源内容描述元数据开始，指导描述元数据的元元数据，形成了一个层次分明、结构开放的元数据体系，如图5-2所示。

信息内容	内容元数据 标记数字对象内容及结构的元数据
内容对象	专门元数据 描述单一数字对象的内容、属性及外在特征的元数据
内容对象集合	资源集合元数据 按照科学、主题、资源类型、用户范围、生成过程，使用管理范围形成的信息资源集合的描述
对象的管理与保存	管理元数据 数字对象加工、存档、结构、技术处理、存取、控制、版权管理以及相关系统等方面的信息描述
对象的服务 服务过程 服务系统	服务元数据 数字资源服务的揭示与表现、服务过程、服务系统等方面的相关信息的描述
元数据的管理	元元数据 对元数据的标记语言、格式语言、标识符、扩展机制、转换机制等的描述

图 5-2 元数据体系与元数据类型

## 2. 数据元标准化

随着国际电子商务和贸易的快速发展，需要一个互连、互通、互操作的开放系统互连环境（Open Systems Interconnection Environment, OSIE）。OSIE 四个基本要素（硬件、软件、通信和数据）中的三个要素（硬件、软件和通信），已经或正在制定相应的标准。为了使数据在各种不同的应用环境中易于交换和共享，国际标准化组织（International Organization for Standardization, ISO）提出了数据元标准的概念，要求按共同约定的规则进行统一组织、分类和标识数据，规范统一数据的含义、表示方法和取值范围等，保证数据从产生的源头就具备一致性。

### 1) 数据元

数据元是数据库、文件和数据交换的基本数据单元。数据库或文件由记录或元组等组成，而记录或元组则由数据元组成。数据元是在数据库或文件之间进行数据交换时的基本组成。数据元通过一组属性描述其定义、标识、表示和允许值的数据单元。在特定的语义环境中被认为是不可再分的最小数据单元。数据元一般来说由三部分组成：①对象。对象类是可以对其界限和含义进行明确的标识，且特性和行为遵循相同规则的观念、抽象概念或现实世界中事物的集合。它是人们希望采集和存储数据的事物。对象类在面向对象的模型中与类相对应，在实体-关系模型中与实体对应，如学员、教员、军事院校等。②特性。特性是指一个对象类的所有成员所共有的特征。它用来区别和描述对象，构成对象类的内涵。特性对应于面向对象模型或实体-关系模型中的属性，如身高、体重、血压、脉搏、血型等。③表示。表示可包括值域、数

据类型、表示类（可选的）和计量单位四部分，其中任何一部分发生变化都成为不同的表示。值域是数据元允许值的集合，例如“学生总数”这一数据元的值域是用非负实数集作为它的允许值集合。数据类型是表达数据元不同可选值的集合。以这些值的特性和运算为特征，例如学生姓名的数据类型是“字符”。表示类是表示类型的分类，它是可选的，例如“性别代码”这一值域的表示类是“类别”。计量单位是用于计量相关值的实际单位，例如学生身高的计量单位是“厘米”。

### 2) 数据元提取

数据元提取是数据元标准化的一项重要内容，为了确保数据元具有科学性和互操作性，需要采用合理的数据元提取方法。目前常用的数据元提取方法有两种：自上而下（Top-Down）提取法和自下而上（Down-Top）提取法。对于新建系统的数据元提取，一般适用“自上而下”的提取法。基本步骤是在流程和功能分析的基础上，通过建模分析，确立关心的“对象”。在概念数据模型和逻辑数据模型的基础上，分析提取数据元及其属性。自下而上提取法也称逆向工程，对于已建系统的数据元提取，一般适用这种自下而上提取法。在这种情况下，数据元直接来自各个信息系统。数据元创建者依据数据元标准化方法，对信息系统及相关资源的数据，在分析、梳理的基础上，归纳整理出数据元；根据数据元的实际应用，阐明并写出相关数据元在采集、存储和交换过程中各个属性以及属性的约束要求；描述和定义各个属性所需要的属性描述符及其约束要求；根据给定的命名表示规范形成数据元。

### 3) 数据元标准

一般来说，制定一个数据元标准，应遵循若干个基本过程，如表 5-10 所示。

表 5-10 数据元制定的基本过程

步骤	说明
描述	用于描述数据的内容、覆盖范围、质量、管理方式、数据的所有者、数据的提供方式等信息，是数据与用户之间的桥梁
界定业务范围	通过对业务范围的明确界定，确定所要研究的数据元的范围
开展业务流程分析与信息建模	数据虽然是任何业务的核心所在，但并不能脱离业务流程而单独存在，它总是服务于业务流程，因此通过对业务流程的透彻分析，并建立清晰的数据模型，可以理清整个业务流程中涉及的所有数据元
借助于信息模型，提取数据元，并按照一定的规则规范其属性	GB/T 18391《信息技术数据元的规范与标准化》清晰地给出了如何对数据元进行描述的方法，以及如何赋予数据元属性的值。比如如何描写数据元的定义，如何对数据元进行命名，如何区分数据元的数据类型等
对于代码型的数据元，编制其值域，即代码表	代码表的编写可以按照 GB/T 7026《标准化工作导则信息分类编码标准的编写规定》进行
与现有的国家标准或行业标准进行协调	这一步是非常重要的工作，编制数据元标准首先要与相应的国家标准保持一致。首先，如果能直接使用现有的国家标准，则可直接使用，或在国家标准的基础上进行扩展；其次要与相关的行业标准保持一致；然后还必须考虑与本行业或领域内已有标准保持最大兼容性，因此要全面考虑协调性和配套性

(续表)

步骤	说明
发布实施数据元标准并建立相应的动态维护管理机制	数据元的标准化工作是一项长期持续的工作，任何行业或领域的数据元标准化工作都不可能在短时间内全部完成，它不仅需要各级业务部门的长期工作和共同努力，还需要根据业务需求的不断变化对其进行修改、补充和完善。因此，需要一种动态维护管理的机制来保障数据元标准化的持续进行。通过建立数据元注册系统，对数据元进行动态维护管理，一方面可以方便用户定位、查找和交换数据元规范，另一方面可以有效保证数据元标准的时效性

### 3. 数据模式标准化

数据模式是数据的概念、组成、结构和相互关系的总称。本质上，数据模式反映的是人类对客观世界的主观认知，而不同的人群对相同的客观世界的主观认知会有所不同，这就造成了在相同领域有不同的数据模式存在。在数据共享过程中，这种差异对人们进行信息的共享与交换形成了障碍。为了保证能够顺畅进行信息的共享，对特定领域而言，需要一个统一的数据模式作为数据共享与交换的基础。同时也保证该领域的相关人员对统一的数据模型有准确的、无歧义的理解。

但在物理和技术层面，各类数据资源的数据格式、存储方式等各不相同，因此需要采用跨越物理和技术层面的方法来进行描述，也就是从数据的逻辑层面对数据集的内容、组成及其结构信息，进行合理的、规范的、本质上的说明和描述。通过数据集模式的标准化，一方面对数据的内容、组成、结构以及各部分的相互关系进行统一规范，相关领域、部门或者数据集制作者都可以根据数据模式制作出标准化的数据；另一方面，数据集按照数据库理论对数据进行了规范化处理，有利于减少数据冗余。

在建立各数据集的数据模式的过程中，需要对客观世界的实体进行分析和抽象，利用图形、文字等方法定义各种实体和相互关系。为对数据模式形成一致的理解，必须有规范的方法来客观、无歧义地描述数据集的内容、组成及其结构。数据模式的描述方式主要有图描述方法和数据字典方法。图描述方法常用的有IDEFIX方法和UML图，主要用来描述数据集中的实体和实体之间的相互关系；数据字典形式用来描述模型中的数据集、单个实体、属性的摘要信息。

### 4. 数据分类与编码标准化

数据分类是根据内容的属性或特征，将数据按一定的原则和方法进行区分和归类，并建立起一定的分类体系和排列顺序。数据分类有分类对象和分类依据两个要素。分类对象由若干个被分类的实体组成；分类依据取决于分类对象的属性或特征。任何一种信息都有多种多样的属性特征，这些属性特征有本质和非本质属性特征之别。分类应以相对最稳定的本质属性为依据，但是对具有交叉、双重或多重本质属性特征的信息进行分类，除了需要符合科学性、系统性等原则外，还应符合交叉性、双重或多重性的原则。

数据编码是将事物或概念（编码对象）赋予具有一定规律和易于计算机、人识别处理的符号，形成代码元素集合。代码元素集合中的代码元素就是赋予编码对象的符号，即编码对象的代码值。所有类型的数据都能够进行编码，如关于产品、人、国家、货币、程序、文件、部件

等各种各样的信息。

所谓数据分类与编码标准化就是把数据分类与编码工作纳入标准化工作的领域，按标准化的要求和工作程序，将各种数据按照科学的原则进行分类以编码，经有关方面协商一致，由主管机构批准、注册，以标准的形式发布，作为共同遵守的准则和依据，并在其相应的级别范围内宣贯和推行。

数据分类与编码标准化是简化信息交换、实现信息处理和信息资源共享的重要前提，是建立各种信息管理系统的重要技术基础和信息保障依据。通过分类与编码标准化，可以最大限度地消除对信息命名、描述、分类和编码的不一致造成的混乱、误解等现象，可以减少信息的重复采集、加工、存储等操作，使事物的名称和代码的含义统一化、规范化，确立代码与事物或概念之间的一一对应关系，以保证数据的准确性和相容性，为信息集成与资源共享提供良好的基础。数据分类与编码的作用主要包括用于信息系统的共享和互操作，统一数据的表示法和提高信息处理效率。

### 5. 数据标准化管理

在数据标准化活动中，首先要依据信息需求，并参照现行数据标准、信息系统的运行环境以及法规、政策和指导原则，在数据管理机构、专家组和开发者共同参与下，运用数据管理工具，得到注册的数据元素、物理模式和扩充的数据模型。数据标准化阶段的具体过程包括确定数据需求、制定数据标准、批准数据标准和实施数据标准四个阶段。

(1) 确定数据需求。本阶段将产生数据需求及相关的元数据、域值等文件。在确定数据需求时应考虑现行的法规、政策，以及现行的数据标准。

(2) 制定数据标准。本阶段要处理“确定数据需求”阶段提出的数据需求。如果现有的数据标准不能满足该数据需求，可以建议制定新的数据标准，也可建议修改或者封存已有数据标准。推荐的、新的或修改的数据标准记录于数据字典中。这个阶段将产生供审查和批准的成套建议。

(3) 批准数据标准。本阶段的数据管理机构对提交的数据标准建议、现行数据标准的修改或封存建议进行审查。一经批准，该数据标准将扩充或修改数据模型。

(4) 实施数据标准。本阶段涉及在各信息系统中实施和改进已批准的数据标准。

### 5.2.3 数据运维

数据开发利用的前提是通过合适的方式将数据保存到存储介质上，并能保证有效的访问，还要通过数据备份和容灾手段，保证数据的高可用性。数据质量管理是在数据产品的生产过程中，确定质量方针、目标和职责，并通过质量策划、质量控制、质量保证和质量改进，来实现所有管理职能的全部活动。

#### 1. 数据存储

所谓数据存储就是根据不同的应用环境，通过采取合理、安全、有效的方式将数据保存到物理介质上，并能保证对数据实施有效的访问。这里面包含两个方面：①数据临时或长期驻留的物理媒介；②保证数据完整安全存放和访问而采取的方式或行为。数据存储就是把这两个方

面结合起来，提供完整的解决方案。

(1) 数据存储介质。数据存储首先要解决的是存储介质的问题。存储介质是数据存储的载体，是数据存储的基础。存储介质并不是越贵越好、越先进越好，要根据不同的应用环境，合理选择存储介质。存储介质的类型主要有磁带、光盘和磁盘三种。

(2) 存储管理。存储管理在存储系统中的地位越来越重要，例如如何提高存储系统的访问性能，如何满足数据量不断增长的需要，如何有效的保护数据、提高数据的可用性，如何满足存储空间的共享等。存储管理的具体内容如表 5-11 所示。

表 5-11 存储管理的主要内容

管理	主要内容
资源调度管理	资源调度管理的功能主要是添加或删除存储节点，编辑存储节点的信息，设定某类型存储资源属于某个节点，或者设定这些资源比较均衡地存储到节点上。它包含存储控制、拓扑配置以及各种网络设备如集线器、交换机、路由器和网桥等的故障隔离
存储资源管理	存储资源管理是一类应用程序，它们管理和监控物理和逻辑层次上的存储资源，从而简化资源管理，提高数据的可用性。被管理的资源包括存储硬件如 RAID、磁带以及光盘库。存储资源管理不仅包括监控存储系统的状况、可用性、性能以及配置情况，还包括容量和配置管理以及事件报警等，从而提供优化策略
负载均衡管理	负载均衡是为了避免存储资源由于资源类型、服务器访问频率和时间不均衡造成浪费或形成系统瓶颈而平衡负载的技术
安全管理	存储系统的安全主要是防止恶意用户攻击系统或窃取数据。系统攻击大致分为两类：一类以扰乱服务器正常工作为目的，如拒绝服务攻击 DoS 等；另一类以入侵或破坏服务器为目的，如窃取数据、修改网页等

## 2. 数据备份

数据备份是为了防止由于用户操作失误、系统故障等意外原因导致的数据丢失，而将整个应用系统的数据或一部分关键数据复制到其他存储介质上的过程。这样做的目的是保证当应用系统的数据不可用时，可以利用备份的数据进行恢复，尽量减少损失。

当前最常见的数据备份结构可以分为四种：DAS 备份结构、基于 LAN 的备份结构、LAN-FREE 备份结构和 SERVER-FREE 备份结构。常见的备份策略主要有三种：完全备份、差分备份和增量备份。

在数据备份系统中，备份服务器、RAID 和磁带机等设备提供了硬件基础，具体备份策略的制定、备份介质的管理以及一些扩展功能的实现都需要软件来完成。备份软件主要分为两大类：一是操作系统自带的软件，如麒麟操作系统的“备份”工具，这类软件实现的功能都很简单；二是专业备份软件，其能够实现比较全面的功能。

## 3. 数据容灾

一切引起系统非正常停机的事件都可以称为灾难，包括不可预料、不可抗拒的自然灾害，系统软硬件故障、人为误操作和恶意攻击等。根据容灾系统保护对象的不同，容灾系统分为应用容灾和数据容灾两类。应用容灾用于克服灾难对系统的影响，保证应用服务的完整、可靠和

安全等一系列要求，使得用户在任何情况下都能得到正常的服务；数据容灾则关注于保证用户数据的高可用性，在灾难发生时能够保证应用系统中的数据尽量少丢失或不丢失，使得应用系统能不间断地运行或尽快地恢复正常运行。

在一般情况下，数据容灾是应用容灾的一个子集，也是应用容灾最根本的基础，因为“得数据者得天下”，数据是应用系统的基础。容灾是一个工程，而不仅仅是技术，有其完整的流程、规范及其具体措施。

数据备份是数据容灾的基础。数据备份是数据高可用的最后一道防线，其目的是为了在系统数据崩溃时能够快速恢复数据。虽然它也算一种容灾方案，但这种容灾能力非常有限，因为传统的数据备份主要是采用磁带进行冷备份，备份磁带一般存放在机房中进行统一管理，一旦整个机房出现了灾难，如火灾、盗窃和地震等灾难时，这些备份磁带也随之毁灭，起不到任何容灾作用。

容灾不是简单备份。真正的数据容灾就是要避免传统冷备份所具有先天不足，它在灾难发生时能全面、及时地恢复整个系统。容灾按其灾难恢复能力的高低可分为多个等级，例如国际标准 SHARE 78 定义的容灾系统有七个等级：从最简单的仅在本地进行磁带备份，到将备份的磁带存储在异地，再到建立应用系统实时切换的异地备份系统。恢复时间也可以从几天到小时级到分钟级、秒级或零数据丢失等。从技术上看，衡量容灾系统有两个主要指标：RPO（Recovery Point Object）和 RTO（Recovery Time Object），其中 RPO 代表了当灾难发生时允许丢失的数据量；而 RTO 则代表了系统恢复的时间。

#### 4. 数据质量评价与控制

在不同时期，数据质量有不同的概念和标准。目前普遍认为，数据质量高低必须从用户使用的角度来看，即使准确性相当高的数据，如果时效性差或者不为用户所关心，仍达不到质量管理标准。数据质量是一个广义的概念，是数据产品满足指标、状态和要求能力的特征总和。

##### 1) 数据质量描述

数据质量可以通过数据质量元素来描述，数据质量元素分为数据质量定量元素和数据质量非定量元素。

##### 2) 数据质量评价过程

数据质量评价过程是产生和报告数据质量结果的一系列步骤，图 5-3 描述了数据质量评价过程。

##### 3) 数据质量评价方法

数据质量评价程序是通过应用一个或多个数据质量评价方法来完成的。数据质量评价方法分为直接评价法和间接评价法：

- 直接评价法：通过将数据与内部或外部的参照信息，如理论值等进行对比。确定数据质量。
- 间接评价法：利用数据相关信息，如数据只对数据源、采集方法等的描述推断或评估数据质量。

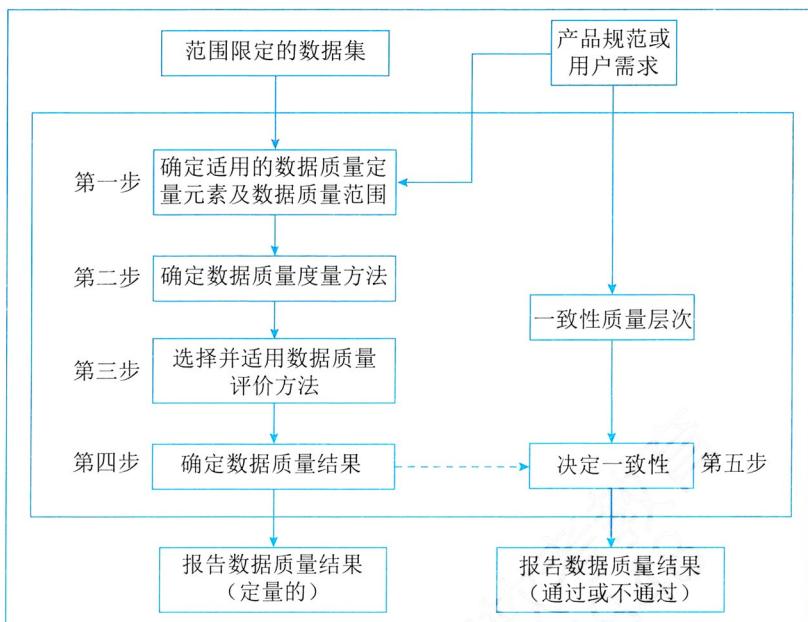


图 5-3 数据质量评价过程

#### 4) 数据质量控制

数据产品的质量控制分成前期控制和后期控制两个大部分。前期控制包括数据录入前的质量控制、数据录入过程中的实时质量控制；后期控制为数据录入完成后的后处理质量控制与评价。依据建库流程可分为：

- 前期控制：是在提交成果（即数据入库）之前对所获得的原始数据与完成的工作进行检查，进一步发现和改正错误。
- 过程控制：实施减少和消除误差和错误的实用技术和步骤，主要应用在建库过程中，用来对获得的数据在录入过程中进行属性的数据质量控制。
- 系统检测：在数据入库后进行系统检测，设计检测模板，利用检测程序进行系统自检。
- 精度评价：对入库属性数据用各种精度评价方法进行精度分析，为用户提供可靠的属性数据。

#### 5) 数据清理

数据清理也称数据清洗。从广义上讲，是将数据库精简以除去重复记录，并使剩余部分转换成符合标准的过程。而狭义上的数据清理是特指在构建数据仓库和实现数据挖掘前对数据源进行处理，使数据实现准确性、完整性、一致性、唯一性、适时性、有效性以适应后续操作的过程。从提高数据质量的角度出发，凡是有助于提高数据质量的处理过程，都可以认为是数据清理。一般说来，数据清理主要包括数据分析、数据检测和数据修正三个步骤，如图 5-4 所示。

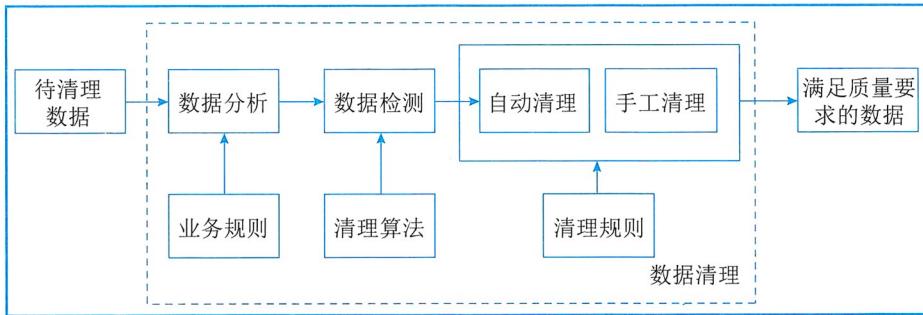


图 5-4 数据清理的流程

数据清理的三个步骤：

- 数据分析：是指从数据中发现控制数据的一般规则，比如字段域、业务规则等，通过对数据的分析，定义出数据清理的规则，并选择合适的清理算法。
- 数据检测：是指根据预定义的清理规则及相关数据清理算法，检测数据是否正确，比如是否满足字段域、业务规则等，或检测记录是否重复。
- 数据修正：是指手工或自动地修正检测到的错误数据或重复的记录。

#### 5.2.4 数据开发利用

数据只有得到充分的开发利用才能发挥出它的作用。通过数据集成、数据挖掘和数据服务（目录服务、查询服务、浏览和下载服务、数据分发服务）、数据可视化、信息检索等技术手段，帮助数据用户从数据资源中找到所需要的数据，并将数据以一定的方式展现出来，实现对数据的开发利用。

##### 1. 数据集成

数据集成就是将驻留在不同数据源中的数据进行整合，向用户提供统一的数据视图（一般称为全局模式），使得用户能以透明的方式访问数据。其中“数据源”主要是指 DBMS，广义上也包括各类 XML 文档、HTML 文档、电子邮件、普通文件等结构化、半结构化和非结构化数据。这些数据源存储位置分散，数据类型异构，数据库产品多样。

数据集成的目标就是充分利用已有数据，在尽量保持其自治性的前提下，维护数据源整体上的一致性，提高数据共享利用效率。实现数据集成的系统称为数据集成系统，它为用户提供了统一的数据源访问接口，用于执行用户对数据源的访问请求。典型的数据集成系统模型如图 5-5 所示。

##### 2. 数据挖掘

数据挖掘是指从大量数据中提取或“挖掘”知识，即从大量的、不完全的、有噪声的、模糊的、

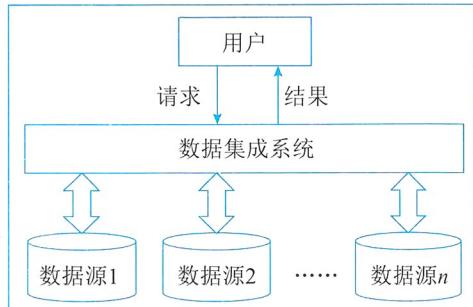


图 5-5 数据集成系统模型

随机的实际数据中，提取隐含在其中的、人们不知道的、却是潜在有用的知识。它把人们对数据的从低层次的简单查询，提升到从数据库挖掘知识，提供决策支持层面。数据挖掘是一门交叉学科，其过程涉及数据库、人工智能、数理统计、可视化、并行计算等多种技术。

数据挖掘与传统数据分析不同：①两者分析对象的数据量有差异，数据挖掘所需的数据量比传统数据分析所需的数据量大，数据量越大，数据挖掘的效果越好；②两者运用的分析方法有差异，传统数据分析主要运用统计学的方法、手段对数据进行分析，而数据挖掘综合运用数据统计、人工智能、可视化等技术对数据进行分析；③两者分析侧重有差异，传统数据分析通常是回顾型和验证型的，通常分析已经发生了什么，而数据挖掘通常是预测型和发现型的，预测未来的情况，解释发生的原因；④两者成熟度不同，传统数据分析由于研究较早，其分析方法相当成熟，而数据挖掘除基于统计学等方法外，部分方法仍处于发展阶段。

数据挖掘的目标是发现隐藏于数据之后的规律或数据间的关系，从而服务于决策。数据挖掘常见的主要任务包括数据总结、关联分析、分类和预测、聚类分析和孤立点分析。数据挖掘流程一般包括确定分析对象、数据准备、数据挖掘、结果评估与结果应用五个阶段，如图 5-6 所示，这些阶段在具体实施中可能需要重复多次。为完成这些阶段的任务，需要不同专业人员参与其中，专业人员主要包括业务分析人员、数据挖掘人员和数据管理人员。

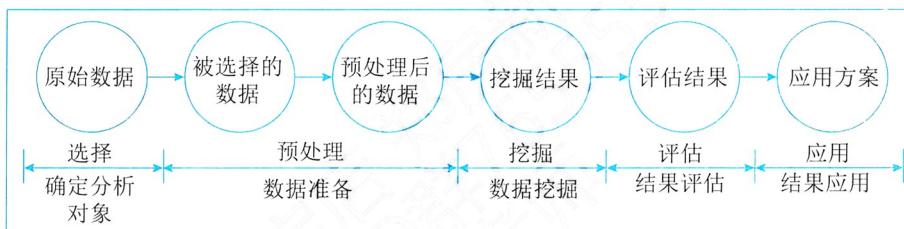


图 5-6 数据挖掘流程图

数据挖掘的结果经过决策人员的许可才能实际运用，以指导实践。将通过数据挖掘得出的预测模式和各个领域的专家知识结合在一起，构成一个可供不同类型的人使用的应用程序。也只有通过对分析知识的应用，才能对数据挖掘的成果做出正确的评价。

### 3. 数据服务

数据服务主要包括数据目录服务、数据查询与浏览及下载服务、数据分发服务。

(1) 数据目录服务。由于专业、领域、主管部门、分布地域和采用技术的不同，数据资源呈现的是海量、多源、异构和分布的特点。对于需要共享数据的用户来说，往往存在不知道有哪些数据、不知道想要的数据在哪里、不知道如何获取想要的数据等困难。数据目录服务就是要解决这些问题，是用来快捷地发现和定位所需数据资源的一种检索服务，是实现数据共享的重要基础功能服务之一。

(2) 数据查询与浏览及下载服务。数据查询、浏览和下载是网上数据共享服务的重要方式，用户使用数据的方式有查询数据和下载数据两种。

(3) 数据分发服务。数据分发是指数据的生产者通过各种方式将数据传送到用户的过程。

通过分发，能够形成数据从采集、存储、加工、传播向使用流动，实现数据的价值。分发服务的核心内容包括数据发布、数据发现、数据评价和数据获取。

#### 4. 数据可视化

可视化技术是指将抽象的事物或过程变成图形图像的表示方法。科学计算可视化（Visualization in Scientific Computing）的基本含义是运用图形学的原理和方法，将科学与工程计算等产生的大规模数据转换为图形、图像，以直观的形式表示出来。

数据可视化（Data Visualization）概念来自科学计算可视化。数据可视化（见图 5-7）主要运用计算机图形学和图像处理技术，将数据转换成为图形或图像在屏幕上显示出来，并能进行交互处理，它涉及计算机图形学、图像处理、计算机辅助设计、计算机视觉及人机交互技术等多个领域，是一门综合性的学科。

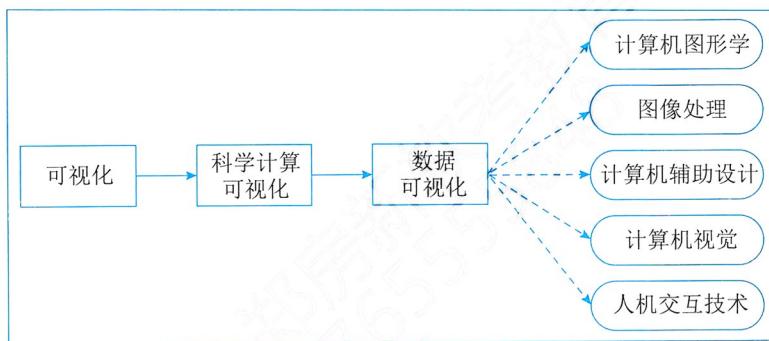


图 5-7 数据可视化

由于所要展现数据的内容和角度不同，可视化的表现方式也多种多样，主要可分为七类：一维数据可视化、二维数据可视化、三维数据可视化、多维数据可视化、时态数据可视化、层次数据可视化和网络数据可视化，如表 5-12 所示。

表 5-12 常见数据可视化表现方式

表现方式	说明
一维数据可视化	一维数据就是简单的线性数据，如文本或数字表格、程序源代码都基于一维数据。一维数据可视化取决于数据大小和用户想用数据来处理什么任务
二维数据可视化	在数据可视化中，二维数据是指由两种主要描述属性构成的数据，如一个物体的宽度和高度、一个城市的平面地图、建筑物的楼层平面图等都是二维数据可视化的实例。最常见的二维数据可视化就是地理信息系统（Geographic Information System, GIS）
三维数据可视化	三维数据比二维数据更进了一层，它可以描述立体信息。三维数据可以表示实际的三维物体，因此可视化的许多应用是三维可视化。物体通过三维可视化构成计算机模型，供操作及试验，以此预测真实物体的实际行为
多维数据可视化	在可视化环境中，多维数据所描述事物的属性超过三维，为了实现可视化，往往需要降维

(续表)

表现方式	说明
时态数据可视化	时态数据实际上是二维数据的一种特例，即二维中有一维是时间轴。它以图形方式显示随着时间变化的数据，是可视化信息最常见、最有用的方式之一
层次数据可视化	层次数据即树形数据，其数据内在结构特征为：每个节点都有一个父节点（根节点除外）。节点分兄弟节点（拥有同一个父节点的节点）和子节点（从属该节点的节点）。拥有这种结构的数据很常见，如商业组织、计算机文件系统和家谱图都是按树形结构排列的层次数据
网络数据可视化	网络数据指与任意数量的其他节点有关系的节点的数据。网络数据中的节点不受与它有关系的其他节点数量的约束（不同于层次节点有且只有一个父节点），网络数据没有固有的层次结构，两个节点之间可以有多条连接路径，也就是说节点间关系的属性和数量是可变的

## 5. 信息检索

信息检索（Information Retrieval）有广义和狭义之分。广义的信息检索是指将信息按一定的方式组织和存储起来，然后根据用户需求查找出特定信息的技术，所以全称是信息存储与检索（Information Storage and Retrieval）。狭义的信息检索仅指用户查找特定信息这部分，即按照用户的检索需求，利用已有的检索工具或数据库，从中找出特定信息的过程。

信息检索的主要方法如下：

- (1) 全文检索。以文本数据为主要处理对象，根据数据资料的内容而不是外在特征来实现的信息检索手段。
- (2) 字段检索。把检索对象按一定标准在不同字段中进行著录，并把不同字段作为检索依据。
- (3) 基于内容的多媒体检索。按检索内容可分为图像检索、视频检索和声音检索等。
- (4) 数据挖掘。从大量的、不完全的、模糊的、随机的数据中，提取隐含在其中且人们事先不知道的潜在、有用的信息和知识的过程。

信息检索的常用技术包括布尔逻辑检索技术、截词检索技术、临近检索技术、限定字段检索技术、限制检索技术等。

- (1) 布尔逻辑检索技术。严格意义上的布尔检索法是指利用布尔逻辑运算符连接各个检索词，然后由计算机进行相应的逻辑运算，以找出所需信息的方法。
- (2) 截词检索技术。截词检索技术是指用截断的词的一个局部进行检索，并认为凡是满足这个词局部的所有字符的信息，都为命中的信息。截词符用“?”或“\*”表示（不同系统、不同数据库，其代表的含义有所不同）。
- (3) 临近检索技术。临近检索又称位置检索，主要是通过检索式中的专门符号来规定检索词在结果中的相对位置。在某些情况下，若不限制检索词之间的位置关系则会造成误检，影响查准率。
- (4) 限定字段检索技术。限定字段检索即指定检索词在记录中出现的字段。检索时，计算机只对限定字段进行匹配运算，以提高检索效率和查准率。
- (5) 限制检索技术。限制检索是通过限制检索范围，达到优化检索的方法。限制检索的方

式有很多种，例如进行字段检索，使用限制符，采用限制检索命令等。

### 5.2.5 数据库安全

数据是脆弱的，它可能被无意识或有意识地破坏、修改，需要采用一定的数据安全措施，确保合法的用户、采用正确的方式、在正确的时间、对相应的数据进行正确的操作，确保数据的机密性、完整性、可用性和合法使用。数据库安全是指保护数据库，防止不合法的使用所造成的数据泄露、更改或破坏。

#### 1. 数据库安全威胁

在数据库环境中，不同的用户通过数据库管理系统访问同一组数据集合，这样减少了数据的冗余、消除了不一致的问题，同时也免去程序对数据结构的依赖。然而，这也导致数据库面临更严重的安全威胁。数据库安全威胁的主要类型如表 5-13 所示。

表 5-13 数据库安全分类及说明

维度	表现方式		说明
安全后果	非授权的信息泄露	未获授权的用户有意或无意得到信息。通过对授权访问的数据进行推导分析获取非授权的信息也属于这一类	
		包括所有通过数据处理和修改而违反信息完整性的行为。非授权修改不一定会涉及非授权信息泄露，因为即使不读取数据也可以进行破坏	
	拒绝服务	包括会影响用户访问数据或使用资源的行为	
威胁方式	无意	自然或意外灾害	如地震、水灾、火灾等。这些事故可能会破坏系统的软硬件，导致完整性破坏和拒绝服务
		系统软硬件中的错误	这会导致应用实施错误的策略，从而导致非授权的信息泄露、数据修改或拒绝服务
		人为错误	导致无意地违反安全策略，导致的后果与软硬件错误类似
	有意	授权用户	他们滥用自己的特权造成威胁
		恶意代理	病毒、特洛伊木马和后门是这类威胁中的典型代表

#### 2. 数据库安全对策

根据数据库安全威胁的特点，数据库安全对策如表 5-14 所示。

表 5-14 常用的数据库安全对策

安全对策	说明
防止非法的数据访问	这是数据库安全关键的需求之一。数据库管理系统必须根据用户或应用的授权来检查访问请求，以保证仅允许授权的用户访问数据库。数据库的访问控制要比操作系统中的文件访问控制复杂得多。首先，控制的对象有更细的粒度，如表、记录、属性等；其次，数据库中的数据是语义相关的，所以用户可以不直接访问数据项而间接获取数据
防止推导	推导指的是用户通过授权访问的数据，经过推导得出机密信息，而按照安全策略，该用户是无权访问此机密信息的。在统计数据库中需要防止用户从统计聚合信息中推导得到原始个体信息，特别是统计数据库容易受到推导问题的影响

(续表)

安全对策	说明
保证数据库的完整性	该需求指的是保护数据库不受非授权的修改，以及不会因为病毒、系统中的错误等导致的存储数据破坏。这种保护通过访问控制、备份 / 恢复以及一些专用的安全机制共同实现。备份 / 恢复在数据库管理系统领域得到了深入的研究，它们的主要目标是在系统发生错误时保证数据库中数据的一致性
保证数据的操作完整性	定位于在并发事务中保证数据库中数据的逻辑一致性。一般而言，数据库管理系统中的并发管理器子系统负责实现这部分需求
保证数据的语言完整性	在修改数据时，保证新值在一定范围内符合逻辑上的完整性。对数据值的约束通过完整性约束来描述。可以针对数据库定义完整性约束（定义数据库处于正确状态的条件），也可以针对变换定义完整性约束（修改数据库时需要验证的条件）
审计和日志	为了保证数据库中数据的安全，一般要求数据库管理系统能够将所有的数据操作记录下来。这一功能要求系统保留日志文件，安全相关事件可以根据系统设置记录在日志文件中，以便事后调查和分析，追查入侵者或发现系统的安全弱点。审计和日志是有效的威慑和事后追查、分析工具。与数据库中多种粒度的数据对应，审计和日志需要面对粒度问题。因为记录对一个细粒度对象（如一个记录的属性）的访问可能有用，但是考虑到时间和代价，这样做可能并不实用
标识和认证	同计算机系统的用户管理类似，使用的方法也非常类似。与其他系统一样，标识和认证也是数据库的第一道安全防线。标识和认证是授权、审计等的前提条件
机密数据管理	数据库中的数据可能有部分是机密数据，也有可能全部是机密数据（如军队的数据库），而有些数据库中的数据全部是公开的数据。同时保存机密数据和公开数据的情况比较复杂。对于同时保存机密和公开数据的数据库而言，访问控制主要保证机密数据的保密性，仅允许授权用户的访问。这些用户被赋予对机密数据进行一系列操作的权限，并且禁止传播这些权限。此外，这些被授权访问机密数据的用户应该与普通用户一样可以访问公开数据，但是不能相互干扰。另一种情况是用户可以访问一组特定的机密数据，但是不能交叉访问。此外，还有一种情况是用户可以单独访问特定的机密数据集合，但是不能同时访问全部机密数据
多级保护	多级保护表示一个安全需求的集合。现实世界中很多应用要求将数据划分不同保密级别。在多级保护体系中，进一步的要求是研究如何赋予多数据项组成的集合一个恰当的密级。数据的完整性和保密性是通过给予用户权限来实现的，用户只能访问拥有的权限所对应级别的数据
限界	限界的的意义在于防止程序之间出现非授权的信息传递。信息传递出现在“授权通道”“存储通道”和“隐通道”中。授权通道通过授权的操作提供输出信息，例如编辑或编译一个文件。存储通道是存储区，一个程序向其中存储数据，而其他程序可以读取。隐通道指的是使用系统中并非设计用来进行通信的资源在主体间通信的信道

### 3. 数据库安全机制

数据库安全机制是用于实现数据库的各种安全策略的功能集合，正是由这些安全机制来实现安全模型，进而实现保护数据库系统安全的目标。数据库安全机制包括用户的身份认证、存取控制、数据库加密、数据审计、推理控制等内容。

## 5.3 系统集成

随着信息技术的发展，系统集成逐步成为信息系统实施中一项重要的工作。此处的系统集成概念专指计算机系统的集成，包括计算机硬件平台、网络系统、系统软件、工具软件、应用软件的集成，围绕这些系统的相应咨询、服务和技术支持。它是以计算机有关技术储备为基础，以可靠的产品为工具，用以实现某一特定的计算机系统功能组合的工程行为。

### 5.3.1 集成基础

系统集成的内容包括技术环境的集成、数据环境的集成和应用程序的集成。对于大型信息系统的设计师来说，如何理解它的体系结构，如何实现它的系统集成，应该是值得深思熟虑的头等大事。网络信息系统的系统集成就是运用先进的计算机与通信技术，将支持各个信息孤岛的小运行环境集成统一在一个大运行环境之中。

以系统集成的观点，一个典型的网络信息系统由不同的系统组成。这些系统通常来自多个供应商，包括多种不兼容的硬件和软件平台，运行各种商业、科学计算及工程应用程序。现在，用户希望把所有不同的系统连接起来，构成一个完整的组织级系统。为了实现把这些异构的系统连接起来，并把应用程序从一种系统移植到另一种系统上，现存的专有系统必须适应标准的接口，进而向开放系统过渡。用户希望得到的是多供应商平台间的可互操作性。可以说，系统集成是开放系统驱动的，顺应了计算机工业发展的潮流。

系统集成的工作在信息系统项目建设中非常重要，它通过硬件平台、网络通信平台、数据库平台、工具平台、应用软件平台将各类资源有机、高效地集成到一起，形成一个完整的工作台面。系统集成工作的好坏对系统开发、维护有极大的影响。因此，在技术上需要遵循的基本原则包括：开放性、结构化、先进性和主流化。

(1) 开放性。系统硬软件平台、通信接口、软件开发工具、网络结构的选择要遵循工业开放标准，这是关系到系统生命周期长短的重要问题。对于稍具规模的信息系统，其系统的硬、软件平台很难由单一厂商提供。即使由单一厂商提供也存在着扩充和保护原有投资的问题，不是一个厂商就能解决得了的。由不同厂商提供的系统平台要集成在一个系统中，就存在着接口的标准化和开放问题，它们的连接都依赖于开放标准。所以，开放标准已成为建设信息系统应该考虑的问题。一个集成的信息系统必然是一个开放的信息系统。只有开放的系统才能满足可互操作性、可移植性以及可伸缩性的要求，才可能与另一个标准兼容的系统实现“无缝”的互操作，应用程序才可能由一种系统移植到另一种系统，不断地为系统的扩展、升级创造条件。

(2) 结构化。复杂系统设计的最基本方法依然是结构化系统分析设计方法。把一个复杂系统分解成相对独立和简单的子系统，每一个子系统又分解成更简单的模块，这样自顶向下逐层模块化分解，到底层每一个模块都是可具体说明和可执行的为止。这一思想至今仍是复杂系统设计的精髓。

(3) 先进性。先进性有两层意义：目前先进性和未来先进性。系统的先进性是建立在技术先进性之上的，只有先进的技术才有较强的发展生命力，系统采用先进的技术才能确保系统的优点和较长的生存周期。系统的先进性还表现在系统设计的先进性：先进技术的有机集成、问

题的合理划分，以及应用软件符合人们认知特点等。系统设计的先进性贯穿在系统开发的整个生命周期，乃至整个系统生存周期的各个环节，一定要认真对待。

（4）主流化。系统构成的每一个产品应属于该产品发展的主流，有可靠的技术支持，有成熟的使用环境，并具有良好的升级发展势头。

### 5.3.2 网络集成

计算机网络系统集成不仅涉及技术问题，而且涉及组织的管理问题，因而比较复杂，特别是大型网络系统更是如此。从技术角度讲，网络集成不仅涉及不同厂家的网络设备和管理软件，也会涉及异构和异质网络系统的互联问题。从管理角度讲，每个组织的管理方式和管理思想千差万别，实现向网络化管理的转变会面临许多人为的因素。因此，组织需要结合实际情况，建立网络系统集成的体系框架，指导网络系统建设，实现真正的网络化管理。计算机网络集成的一般体系框架，如图 5-8 所示。

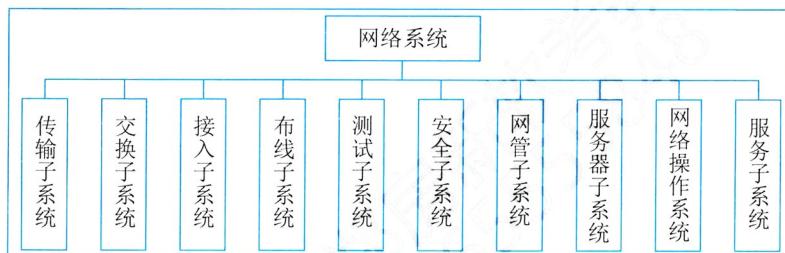


图 5-8 网络集成的体系框架

下面针对其中比较重要的几个方面进行说明。

（1）传输子系统。传输是网络的核心，是网络信息的“公路”和“血管”。传输线路带宽的高低不仅体现了网络的通信能力，也体现了网络的现代化水平。并且，传输介质在很大程度上也决定了通信的质量，从而直接影响到网络协议。目前主要的传输介质分为无线传输介质和有线传输介质两大类。常用的无线传输介质主要包括无线电波、微波、红外线等，常用的有线传输介质主要包括双绞线、同轴电缆、光纤等。

（2）交换子系统。网络按所覆盖的区域可分为局域网、城域网和广域网，由此网络交换也可以分为局域网交换技术、城域网交换技术和广域网交换技术。

（3）安全子系统。由于网络的发展，安全问题一直是网络研究和应用的热点。网络安全主要关注的内容包括：使用防火墙技术，防止外部的侵犯；使用数据加密技术，防止任何人从通信信道窃取信息；访问控制，主要是通过设置口令、密码和访问权限保护网络资源。

（4）网管子系统。网络是一种动态结构。随着组织规模的扩大和改变，网络也会跟着扩大和改变。配置好网络以后，必须对其进行有效的管理，确保网络能连续不断地满足组织的需要。对于任何网管子系统来说，关键的任务便是保证网络的良好运行。由于网络规模的扩大，通常会出现网络“瓶颈”问题，使系统的速度变慢。网管的职责便是找出瓶颈并解决它。

（5）服务器子系统。服务器是网络中的关键设备。服务器的作用就是向工作站提供处理器

内存、磁盘、打印机、软件数据等资源和服务，并负责协调管理这些资源。由于网络服务器要同时为网络上所有的用户服务，因此要求网络服务器具有较高的性能，包括快的处理速度、较大的内存、较大的磁盘容量和高可靠性。根据网络的应用情况和规模，网络服务器可选用高配置微机、工作站、小型机、超级小型机和大型机等。选择网络服务器时要考虑以下因素：①CPU的速度和数量；②内存容量和性能；③总线结构和类型；④磁盘容量和性能；⑤容错性能；⑥网络接口性能；⑦服务器软件等。

(6) 网络操作系统。网络操作系统的主要任务是调度和管理网络资源，并为网络用户提供统一、透明使用网络资源的手段。网络资源主要包括网络服务器、工作站、打印机、网桥、路由器、交换机、网关、共享软件和应用软件等。

(7) 服务子系统。网络服务是网络应用最核心的问题。带宽再高的网络，如果没有好的网络服务，就不能发挥网络的效益。网络服务主要包括互联网服务、多媒体信息检索、信息点播、信息广播、远程计算和事务处理以及其他信息服务等。

### 5.3.3 数据集成

数据集成的目的是运用一定的技术手段将系统中的数据按一定的规则组织成为一个整体，使得用户能有效地对数据进行操作。数据集成处理的主要对象是系统中各种异构数据库中的数据。数据仓库技术是数据集成的关键。

#### 1. 数据集成层次

数据集成是将参与数据库的有关信息在逻辑上集成为一个属于异构分布式数据库的全局概念模式，以达到信息共享的目的。数据集成可以分为基本数据集成、多级视图集成、模式集成和多粒度数据集成四个层次。

(1) 基本数据集成。基本数据集成面临的问题很多。通用标识符问题是数据集成时遇到的最难的问题之一。由于同一业务实体存在于多个系统源中，并且没有明确的办法确认这些实体是同一实体时，就会产生这类问题。处理该问题的办法包括：

- 隔离：保证实体的每次出现都指派一个唯一标识符。
- 调和：确认哪些实体是相同的，并且将该实体的各次出现合并起来。

当目标元素有多个来源时，指定某一系统在冲突时占主导地位。数据丢失问题是常见的问题之一，通常的解决办法是为丢失的数据产生一个非常接近实际的估计值来进行处理。

(2) 多级视图集成。多级视图机制有助于对数据源之间的关系进行集成：底层数据表示方式为局部模型的局部格式，如关系和文件；中间数据表示为公共模式格式，如扩展关系模型或对象模型；高级数据表示为综合模型格式。视图的集成化过程为两级映射：①数据从局部数据库中，经过数据翻译、转换并集成为符合公共模型格式的中间视图；②进行语义冲突消除、数据集成和数据导出处理，将中间视图集成为综合视图。

(3) 模式集成。模型合并属于数据库设计问题，其设计的好坏常视设计者的经验而定，在实际应用中可参考的成熟理论较少。实际应用中，数据源的模式集成和数据库设计仍有相当的差距，如模式集成时出现的命名、单位、结构和抽象层次等冲突问题，就无法照搬模式设计的经验。在