系统分析师案例必备知识点汇总

一、系统规划（视频内容：系统分析师-专业知识模块中的系统规划视频）

1、可行性研究

经济可行性	技术可行性	法律可行性	用户使用可行性
也称为投资收益分析或成本效益分析，主要评估项目的建设成本、运行成本和项目建成后可能的经济收益。	技术可行性也称为技术风险分析，主要评估信息系统需要实现的功能和性能，以及技术能力约束。	法律可行性也称为社会可行性，具有比较广泛的内容，需要从政策、法律、道德、制度等社会因素来论证信息系统建设的现实性。	用户使用可行性也称为执行可行性，是从信息系统用户的角度来评估系统的可行性，包括企业的行政管理和工作制度、使用人员的素质和培训要求等。

2、盈亏临界分析有关公式

①盈亏临界点销售量＝总固定成本/（销售单价－单位变动成本）

②盈亏临界点销售额＝总固定成本/（1－总变动成本/销售收入）

③利润＝（销售单价－单位变动成本）×销售量－总固定成本

3、投资回收期与投资收益率

①静态投资回收期：

累计净现金流量开始出现正值的年份数-1＋｜上年累计净现金流量｜/当年净现金流量

②动态投资回收期：

累计折现值开始出现正值的年份数-1＋｜上年累计折现值｜/当年折现值

③投资收益率：投资收益/投资成本×100％

4、净现值

①现值

$$P=\frac{F}{(1+i)^n}$$

，其中 1/（1＋i）n 称为折现系数（折现因子）或贴现系数（贴现因子）。

②净现值

$$NPV=\sum_{t=0}^{n}\frac{(CI-CO)_t}{(1+i)^t}$$

其中（CI-CO）t 为第 t 年的净现金流量，CI 为现金流入，CO 为现金流出，i 为折现率。

二、系统分析（视频内容：系统分析师--案例分析模块中系统分析视频，此部分为新增视频）

1、fast 开发方法

2、系统约束条件

系统的改进目标可能受到约束条件的调节。约束条件可以分为：进度、成本、技术、政策。

3、结构化分析

通过功能分解方式把系统功能分解到各个模块中，分析结果以数据流图（DFD）和实体关系图（ERD）呈现。

1）数据流图的组成：

①数据流：由一组固定成分的数据组成，表示数据的流向。每一个数据流都有一个定义明确的名字。

②加工：描述了输入数据流到输出数据流之间的变换，即输入数据流经过什么处理后变成输出数据流。每个加工都有一个名字和编号。

③数据存储：用来存储数据。每个数据存储都有一个定义明确的名字标识。

④外部实体：是指存在于软件系统之外的人员或组织，它指出系统所需数据的发源地和系统所产生的数据的归宿地。每个外部实体都有一个定义明确的名字标识。

2）绘制数据流图的步骤

①画系统的输入和输出：在图的边缘标出系统的输入数据流和输出数据流。这一步其实是决定研究的内容和系统的范围。在画的时候，可以先将尽可能多的数据流画出来，然后再删除多余的，增加遗漏的。

②画 DFD 的内部：将系统的输入、输出用一系列的处理连接起来，可以从输入数据流画向输出数据流，也可以从中间画出去。

③为每一个数据流命名：命名的好坏与 DFD 的可理解性密切相关，应避免使用空洞的名字。

④为加工命名：使用动宾短语为每个加工命名。

4、面向对象分析

运用面向对象方法，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面向对象分析模型及其详细说明。

面向对象分析工作的两大成果：需求模型和分析模型。

①需求模型用用例图建立，属于需求工作成果，为分析工作提供依据。构建用例模型的 4 个阶段：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，其中前三个阶段是必需的。
②分析模型属于分析工作成果，用类图建立。建立分析模型的过程：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。

三、需求获取（视频内容：系统分析-专业知识模块中需求工程视频）
1、需求获取的技术：

用户访谈	优点：具有良好的灵活性，有较宽广的应用范围。 缺点是：用户忙，信息量大，记录困难，需要沟通技巧。
问卷调查	优点：短时间内收集数据。 缺点：缺乏灵活性，信息不全面，无法了解细节问题。
采样	优点：加快了数据收集的过程，提高了效率。利用数理统计原理，减少数据收集的偏差。 缺点：主观性强。
情节串联板	优点：用户友好、交互性强，对用户界面提供了早期的评审。 缺点：花费时间，速度慢。
联合需求计划	优点：用户参与，有利于消除矛盾信息。 缺点：会议的组织与相关人员的能力。
其他方法：实地观察、收集资料等 。	

四、系统架构设计（视频内容：系统分析师--案例分析模块中系统架构设计视频，此部分为新增视频）
1、面向服务的架构 SOA
SOA 是一种在计算环境中设计、开发、部署和管理离散逻辑单元（服务）模型的方法。关于服务，一些常见的设计原则有：明确定义的接口、自包含和模块化、粗粒度、松耦合、互操作性。
SOA 紧密相关的技术主要有 UDDI、WSDL、SOAP 和 REST 等，而这些技术都是以 XML 为基础而发展起来的。

UDDI	统一描述、发现和集成，提供了一种服务发布、查找和定位的方法，是服务的信息注册规范，以便被需要该服务的用户发现和使用它。
WSDL	Web 服务描述语言是对服务进行描述的语言。
SOAP	简单对象访问协议定义了服务请求者和服务提供者之间的消息传输规范。SOAP 用 XML 来格式化消息，用 HTTP 来承载消息。通过 SOAP，应用程序可以在网络中进行数据交换和远程过程调用 RPC。
REST	表述性状态转移是一种只使用 HTTP 和 XML 进行基于 Web 通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。 REST 提出了如下一些设计概念和准则： （1）网络上的所有事物都被抽象为资源。 （2）每个资源对应一个唯一的资源标识。 （3）通过通用的连接件接口对资源进行操作。 （4）对资源的各种操作不会改变资源标识。

	（5）所有的操作都是无状态的。

2、微服务

微服务是一种架构风格，将单体应用划分成一组小的服务，服务之间相互协作，实现业务功能每个服务运行在独立的进程中，服务间采用轻量级的通信机制协作（通常是 HTTP/JSON），每个服务围绕业务能力进行构建，并且能够通过自动化机制独立地部署。

- 微服务有以下优势：

（1） 通过分解巨大单体式应用为多个服务方法解决了复杂性问题。它把庞大的单一模块应用分解为一系列的服务，同时保持总体功能不变。

（2） 让每个服务能够独立开发，开发者能够自由选择可行的技术，提供 API 服务。

（3） 微服务架构模式是每个微服务独立的部署。开发者不再需要协调其他服务部署对本服务的影响。这种改变可以加快部署速度。

（4） 微服务使得每个服务独立扩展。你可以根据每个服务的规模来部署满足需求的规模。甚至可以使用更适合于服务资源需求的硬件。

- 微服务架构带来的挑战如下：

（1） 并非所有的系统都能转成微服务。

（2） 部署较以往架构更加复杂：系统由众多微服务搭建，每个微服务需要单独部署，从而增加部署的复杂度，容器技术能够解决这一问题。

（3） 性能问题：由于微服务注重独立性，互相通信时只能通过标准接口，可能产生延迟或调用出错。

（4） 数据一致性问题：作为分布式部署的微服务，在保持数据一致性方面需要比传统架构更加困难。

3、C/S 架构

二层 C/S 结构为单一服务器且以局域网为中心，所以难以扩展至大型企业广域网或 Internet；软、硬件的组合及集成能力有限；它的缺点主要有：

（1）服务器的负荷太重，难以管理大量的客户机，系统的性能容易变坏；

（2）数据安全性不好。因为客户端程序可以直接访问数据库服务器，那么，在客户端计算机上的其他程序也可想办法访问数据库服务器，从而使数据库的安全性受到威胁。

与二层 C/S 架构相比，在三层 C/S 架构中，增加了一个应用服务器。可以将整个应用逻辑驻留在应用服务器上，而只有表示层存在于客户机上。这种客户机称为瘦客户机。三层 C/S 架构将应用系统分成表示层、功能层和数据层三个部分。

与传统的二层架构相比，三层 C/S 架构具有以下优点：

（1）允许合理地划分三层的功能，使之在逻辑上保持相对独立性，从而使整个系统的逻辑结构更为清晰，能提高系统的可维护性和可扩展性。

（2）允许更灵活、有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层，并且这些平台和各个组成部分可以具有良好的可升级性和开放性。

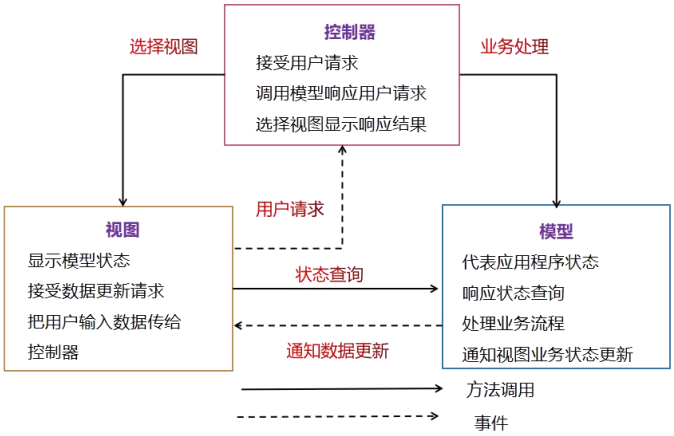
（3）系统的各层可以并行开发，各层也可以选择各自最适合的开发语言，使之能并行且高效地进行开发，达到较高的性能价格比。对每一层的处理逻辑的开发和维护也会更容易些。

（4）利用功能层可以有效地隔离表示层与数据层，未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，这就为严格的安全管理奠定了坚实的基础。

4、B/S 架构

B/浏览器/服务器（Browser/Server，B/S）架构是三层 C/S 架构的一种实现方式，其具体结构为“浏览器/Web 服务器/数据库服务器”。B/S 架构利用 WWW 浏览器技术，结合浏览器的多种脚本语言，用通用浏览器就实现了原来需要复杂的专用软件才能实现的强大功能，并节约了开发成本。

5、MVC



- 模型：执行业务流程（不包括输入输出），存储业务数据。模型不依赖于视图和控制器，提高了架构的灵活性。
- 视图：展示模型中的数据，用户的同一份数据可以通过不同的视图以不同的方式展示。视图必须了解模型中的数据结构，对模型有很强的依赖性，但是模型对于视图则没有依赖性。
- 控制器：把模型接收的事件和用户输入的数据转化为对模型方法的调用。控制器对用户的行为作出解释，并决定调用模型的哪个方法。

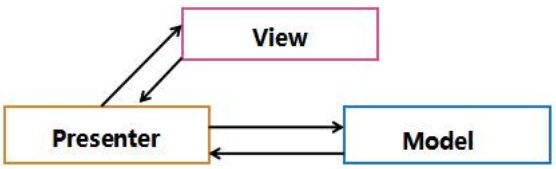
使用 MVC 模式来设计表现层，可以有以下的优点。

（1）允许多种用户界面的扩展。在 MVC 模式中，视图与模型没有必然的联系，都是通过控制器发生关系，这样如果要增加新类型的用户界面，只需要改动相应的视图和控制器即可，而模型则不需发生改动。

（2）易于维护。控制器和视图可以随着模型的扩展而进行相应的扩展，只要保持一种公共的接口，控制器和视图的旧版本也可以继续使用。

（3）功能强大的用户界面。用户界面与模型方法调用组合起来，使程序的使用更清 晰，可将友好的界面发布给用户。

6、MVP



MVP 的优点包括：

（1）低耦合。模型与视图完全分离，可以修改视图而不影响模型。

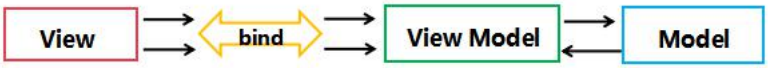
（2）可以更高效地使用模型，因为所有的交互都发生在一个地方—Presenter 内部。

（3）复用性好。可以将一个 Presenter 用于多个视图，而不需要改变 Presenter 的逻辑。这个特性非常的有用，因为视图的变化总是比模型的变化频繁。

（4）可测试性好。如果把逻辑放在 Presenter 中，就可以脱离用户接口来测试这些逻辑（单

元测试）。

7、MVVM



MVVM 是由 MVP 进化而来，MVVM 模式基本上与 MVP 相同，只是把 MVP 中的 P 变成了 VM，即 ViewModel，MVVM 中的数据可以实现双向绑定，当 Model 变化时，View-Model 会自动更新，View 也会自动变化。很好做到数据的一致性，不用担心，在模块的这一块数据是这个值，在另一块就是另一个值了。所以 MVVM 模式有些时候又被称作：model-view-binder 模式。因此 MVVM 框架比较适合逻辑复杂的前端项目，比如一些管理系统等。

8、轻量级架构

- SSH：指的是 Struts2(做前端控制器)，Spring(管理各层的组件)，Hibernate(负责持久化层)
- SSM：指的是 SpringMVC(做前端控制器)，Spring(管理各层的组件)，Mybatis(负责持久化层)

Hibernate 与 Mybatis 区别：

①开发方面：Hibernate 开发中，sql 语句已经被封装，直接可以使用；Mybatis 属于半自动化，sql 需要手工完成。

②sql 优化方面：对复杂查询的 sql 语句进行人工调优的时候，Mybatis 更方便。

③可移植性方面：Hibernate 使用时自动生成相应的 sql 语句，因此具备良好的数据库移植性，而 Mybatis 中手动编写的 sql 语句需要针对不同厂商的数据库进行修改。

五、系统设计（视频内容：系统分析师--案例分析模块中系统设计视频，此部分为新增视频）

1、面向对象设计

分析类图是从用户的角度出发得到的业务“系统”，而设计类图更多的是从系统、软件的角度来描述和表达系统。二者具体的区别：

- 分析类图：在需求分析阶段，类图是研究领域中的概念；分析类图主要用于描述应用领域中的概念，类图中的类从领域中得出，从需求中获取。
- 设计类图：在设计阶段，类图重点描述类与类之间的接口；设计类图用于描述软件的接口部分，而不是软件的实现部分，设计类图更易于开发者之间的相互理解和交流；设计类图通常是在分析类图的基础上进行细化和改进的。

设计类包括实体类、控制类和边界类三种类型。

实体类	边界类	控制类
● 名词。 ● 需要永久存储体 如学生、商品等。	● 用于系统接口与系统外部进行交互 如界面类，如对话框、窗口、菜单	● 动词 ● 控制用例工作的类 ● 体现应用程序的执行逻辑

类之间的关系有：

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。

聚合	整体与部分的关系，各自具有不同的生命周期。
组合	整体与部分的关系，具有相同的生命周期。
依赖	两个类 A 和 B，如果 B 的变化可能会引起 A 的变化。
泛化	父类与子类之间的关系。是继承的反关系。
实现	一个或多个类可以实现一个接口，每个类分别实现接口中的操作。

- 流程图与活动图的区别：

◆ 流程图着重描述处理过程，它的主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系。而活动图描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统的行为，而非系统的处理过程。

◆ 活动图能够表示并发活动的情形，而流程图不行。

◆ 活动图是面向对象的，而流程图是面向过程的。

- 状态图与活动图的区别

状态图	活动图
用来描述一个特定的对象所有可能的状态,以及由于各种事件的发生而引起的状态之间的转移和变化。	将进程或其他计算的结构展示为计算内部一步步的控制流和数据流，主要用来描述系统的动态视图。
状态图主要描述行为的结果。	活动图主要描述行为的动作。
用于对系统的动态方面建模。	用于对系统的动态方面建模。

- 序列图与协作图的区别

序列图	协作图
序列图主要用来更直观的表现各个对象交互的时间顺序，将体现的重点放在以时间为参照，各个对象发送、接收消息，处理消息，返回消息的时间流程顺序，也称为时序图。	协作图是一种类图,强调参与交互的 各个对象的结构信息 和组织。
强调时间次序	强调空间结构
二者均显示了对象间的交互	

2、Web 设计常见技术（无视频，文字知识点）

负载均衡技术	LVS、Haproxy
缓存服务器	Varnish、Ngnix、squid、Memcache、Redis、Ehcache
分布式文件系统	Hadoop、FastDFS
Web 应用服务器	Jetty、Jboss、Apache
分布式数据库	Mysql、MongoDB、Oracle

3、软件产品线

软件产品线是一个产品集合，这些产品共享一个公共的、可管理的特征集，这个特征集能满足特定领域的特定需求。软件产品线是一个十分适合专业开发组织的软件开发方法，能有效地提高软件生产率和质量，缩短开发时间，降低总开发成本。

六、数据库

1、数据库基础知识（视频内容：系统分析师--案例分析模块中数据库视频（此部分为新增视频）、计算机基础知识模块中数据库基础视频）

- 数据库设计

规划	进行建立数据库的必要性及可行性分析，确定数据库系统在企业和信息系统中的地位，以及各个数据库之间的联系。
需求分析	通过调查研究，了解用户的数据和处理要求，并按一定格式整理形成需求说明书。
概念设计	在需求分析阶段产生的需求说明书的基础上，按照特定的方法将它们抽象为一个不依赖于任何数据库管理系统的数据模型，即概念模型。
逻辑设计	将概念模型转化为某个特定的数据库管理系统上的逻辑模型。设计逻辑结构时，首先为概念模型选定一个合适的逻辑模型（通常是关系模型），然后将其转化为由特定 DBMS 支持的逻辑模型，最后对逻辑模型进行优化。
物理设计	对给定的逻辑模型选取一个最适合应用环境的物理结构，所谓数据库的物理结构，主要是指数据库在物理设备上的存储结构和存取方法。

- 规范化

类型	特征
1NF	若关系模式R的每一个分量是不可再分的数据项，则关系模式R属于第一范式。
2NF	若关系模式R∈1NF，且每一个非主属性完全依赖主键时，则关系式R是2NF（第二范式）。
3NF	即当2NF消除了非主属性对码的传递函数依赖，则称为3NF。

- 数据库安全性技术

措 施	说明
用户标识和鉴别	最外层的安全保护措施，可以使用用户账户、口令和随机数检验等方式
存取控制（数据授权）	对用户进行授权，包括操作类型（例如，查找、更新或删除等）和数据对象的权限
密码存储和传输	对远程终端信息用密码传输
视图的保护	通过视图的方式进行授权
审计	用一个专用文件或数据库，自动将用户对数据库的所有操作记录下来

视图是保存在数据库中的 SELECT 查询，其内容由查询定义，因此，视图不是真实存在的基础表，而是从一个或者多个表中导出的虚拟的表。同真实的表一样，视图包含一系列带有名称的列和行数据，但视图中的行和列数据来自由定义视图的查询所引用的表，并且在引

用视图时动态生成。

视图优点有：视点集中、简化操作、定制数据、合并分割数据、保证安全性

➢ 数据库完整性技术

（1）存储过程

存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。

存储过程是数据库所提供的一种数据库对象，通过存储过程定 义一段代码，提供给应用程序调用来执行。 从安全性的角度考虑，更新数据时，通过提供存储过程让第三方调用，将需要更新的数据传入存储过程，而在存储过程内部用代码分别对需要的多个表进行更新，从而避免了向第三方提供系统的表结构，保证了系统的数据安全。

（2）触发器

一种特殊的存储过程，当数据发生变化时，触发器会产生某种动作。使用触发器有助于保持数据库的数据完整性。

2、数据库性能优化

➢ 集中式数据库性能优化

（1）硬件升级：处理器、内存、磁盘子系统和网络。

（2）数据库设计：反规范化技术（逻辑）、数据库分区（物理）。

（3）索引优化策略：选择经常查询不常更新的属性、数据量小的不设置索引等。

（4）查询优化：建立物化视图或尽可能减少多表查询等。

✧ 反规范化技术

增加冗余列	增加冗余列是指在多个表中具有相同的列，它常用来在查询时避免连接操作。
增加派生列	增加派生列指增加的列可以通过表中其他数据计算生成。它的作用是在查询时减少计算量，从而加快查询速度。
重新组表	重新组表指如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。
水平分割表	按记录进行分割，把数据放到多个独立的表中，主要用于表数据规模很大、表中数据相对独立或数据需要存放到多个介质上时使用。
垂直分割表	对表进行分割，将主键与部分列放到一个表中，主键与其它列放到另一个表中，在查询时减少 I/O 次数。

✧ 数据库分区技术

	范围分区	哈希分区	列表分区
数据值	连续	连续离散均可	离散
数据管理能力	强	弱	强
实施难度与可维护性	差	好	差
数据分布	不均匀	均匀	不均匀

➢ 分布式数据库性能优化

（1）主从复制、读写分离
（2）数据库分片（分表）、分库
（3）分布式缓存技术

✧ 主从复制，读写分离

读写分离设置物理上不同的主/从服务器，让主服务器负责数据的写操作，从服务器负责数据的读操作，从而有效减少数据并发操作的延迟。 引入主从复制机制所带来的好处有：

（1）避免数据库单点故障：主服务器实时、异步复制数据到从服务器，当主数据库宕机时，可在从数据库中选择一个升级为主服务器，从而防止数据库单点故障。

（2）提高查询效率：根据系统数据库访问特点，可以使用主数据库进行数据的插入、删除及更新等写操作，而从数据库则专门用来进行数据查询操作，从而将查询操作分担到不同的从服务器以提高数据库访问效率。

Mysql 的主从复制方式：

	一致性	可用性	特征	典型框架/系统
同步复制技术	强	弱	主数据库需要等待所有备数据库均操作成功才可以响应用户，影响用户体验。	MySQL
异步复制技术	弱	强	主数据库处理完请求后可直接给用户响应，而不必等待备数据库完成同步，备数据库会异步进行数据的同步。	MySQL、Redis、Oracle
半同步复制技术	较强	较弱	用户发出写请求后，主数据库会执行写操作，并给备数据库发送同步请求，主数据库可以等待一部分备数据库同步完成后响应用户写操作执行成功。	MySQL、Zookeeper、CloudSQLServer、Redis、Oracle、Etcd等

✧ 分布式缓存技术

为了减轻数据库服务器的压力，可以采用分布式缓存系统，将应用系统经常使用的数据放置在内存，降低对数据库服务器的查询请求，提高系统性能。

使用缓存后读写数据的基本步骤：

（1）读数据

- 根据 key 读缓存；
- 读取成功则直接返回；
- 若 key 不在缓存中时，根据 key 读数据库 ；
- 读取成功后，写缓存 ；
- 成功返回。

（2）写数据

- 根据 key 值写数据库；
- 成功后更新缓存 key 值 ；
- 成功返回。

3、分布式数据库

分布式数据库是由一组数据组成的，这组数据分布在计算机网络的不同计算机上，网络中的每个节点具有独立处理的能力（称为场地自治），它可以执行局部应用，同时，每个节点也

能通过网络通信子系统执行全局应用。
为了提高分布式数据库的性能采用的常见技术有：数据分片、读写分离技术、负载均衡技术、分布式缓存技术等。

4、nosql 数据库

	关系型数据库	Nosql数据库
并发支持	支持并发、效率低	并发性能高
存储与查询	关系表方式存储、SQL查询	海量数据存储、查询效率高
扩展方式	向上扩展	向外扩展
索引方式	B树、哈希等	键值索引
应用领域	面向通用领域	特定应用领域

5、数据库备份

备份方式	优点	缺点
冷备份	非常快速的备份方法(只需复制文件)；容易归档（简单复制即可)；容易恢复到某个时间点上(只需将文件再复制回去）：能与归档方法相结合，做数据库“最佳状态” 的恢复；低度维护，高度安全	单独使用时，只能提供到某一时间点上的恢复;在实施备份的全过程中， 数据库必须要作备份而不能做其他工作;若磁盘空间有限，只能复制到磁带等其他外部存储设备上，速度会很慢;不能按表或按用户恢复
热备份	可在表空间或数据库文件级备份，备份的时间短；备份时数据库仍可使用;可达到秒级恢复（恢复到某一时间点上）；可对几乎所有数据库实体做恢复；恢复是快速的	不能出错，否则后果严重；若热备份不成功,所得结果不可用于时间点的恢复;因难于维护,所以要特别小心不能以失败告终

七、可靠性分析与设计 （视频内容：系统分析-专业知识模块中系统可靠性分析与设计视频）

1、负载均衡技术分类：

①基于 http 的负载均衡。

根据用户的 http 请求计算出一个真实的 web 服务器地址，并将该 web 服务器地址写入 http 重定向响应中返回给浏览器，由浏览器重新进行访问。

- 优点：比较简单
- 缺点：浏览器需要两次请求服务器才能完成一次访问，性能较差。

②基于 DNS 的负载均衡。

- 优点：将负载均衡的工作交给了 DNS，省却了网站管理维护负载均衡服务器的麻烦，同时许多 DNS 还支持基于地理位置的域名解析,将域名解析成距离用户地理最近的一个服务器地址，加快访问速度，改善性能。
- 缺点：目前的 DNS 解析是多级解析，每一级 DNS 都可能化缓存记录，当某一服务器下线后，该服务器对应的 DNS 记录可能仍然存在，导致分配到该服务器的用户访问失败。负载均衡效果并不是太好。

③基于 NAT（Network Address Translation，网络地址转换）的负载均衡。

将一个外部 IP 地址映射为多个内部 IP 地址，对每次连接请求动态地转换为一个内部节点的地址，将外部连接请求引到转换得到地址的那个节点上，从而达到负载均衡的目的。

- 优点：在响应请求时速度较反向服务器负载均衡要快。
- 缺点：当请求数据较大（大型视频或文件）时，速度较慢。

④反向代理负载均衡。

- 优点：部署简单，处于 http 协议层面。
- 缺点：用了反向代理服务器后，web 服务器地址不能直接暴露在外，因此 web 服务器不需要使用外部 IP 地址，而反向代理服务作为沟通桥梁就需要配置双网卡、外部内部两套 IP 地址。

八、系统维护（视频内容：系统分析-专业知识模块中系统运行与维护视频）

1、遗留系统

遗留系统（Legacy System）是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统，它通常具有以下特点：

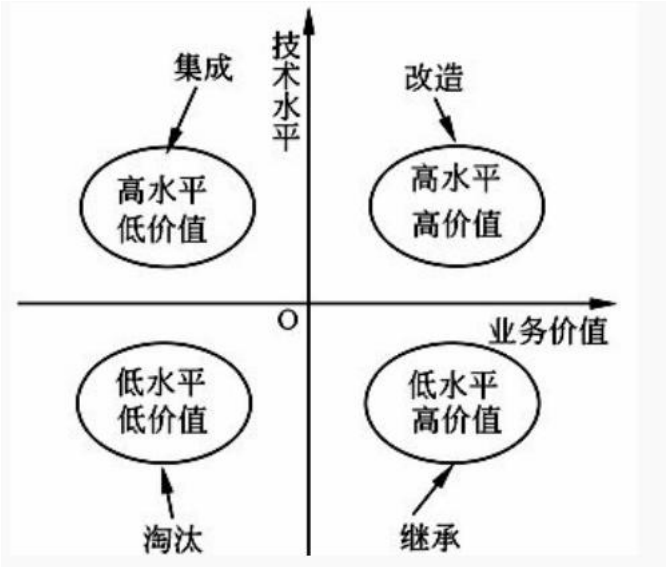
（1）系统虽然完成企业中许多重要的业务管理工作，但仍然不能完全满足要求。一般实现业务处理电子化及部分企业管理功能，很少涉及经营决策。

（2）系统在性能上已经落后，采用的技术已经过时。例如，多采用主机/终端形式或小型机系统，软件使用汇编语言或第三代程序设计语言的早期版本开发，使用文件系统而不是数据库。

（3）通常是大型的软件系统，已经融入企业的业务运作和决策管理机制之中，维护工作十分困难。

（4）没有使用现代信息系统建设方法进行管理和开发，现在基本上已经没有文档，很难理解。

2、遗留系统的演化策略



改造策略	继承策略	淘汰策略	集成策略
改造包括系统功能的	在开发新系统时，需	全面重新开发新的系	采用企业应用集成的

增强和数据模型的改造两个方面。系统功能的增强是指在原有系统的基础上增加新的应用要求，对遗留系统本身不做改变；数据模型的改造是指将遗留系统的旧的数据模型向新的数据模型的转化。	要完全兼容遗留系统的功能模型和数据模型。为了保证业务的连续性，新老系统必须并行运行一段时间，再逐渐切换到新系统上运行。	统以代替遗留系统。	方式进行系统集成。

3、新旧系统转换策略

（1）直接转换策略

直接转换就是在原有系统停止运行的某一时刻，新系统立即投入运行，中间没有过渡阶段。采用这种方式时，人力和费用最省，适用于新系统不太复杂或现有系统完全不能使用的场合，但是，新系统在转换之前必须经过详细而严格的测试，转换时应做好准备，万一新系统不能达到预期目的时，必须采取相应措施。

（2）并行转换策略

并行转换就是新系统和现有系统并行工作一段时间，经过这段时间的试运行后，再用新系统正式替换下现有系统。在并行工作期间，手工处理和计算机处理系统并存，一旦新系统有问题就可以暂时停止而不会影响现有系统的正常工作。

（3）分段转换策略

分段转换策略也称为逐步转换策略，这种转换方式是直接转换方式和并行转换方式的结合，采取分期分批逐步转换。一般比较大的系统采用这种方式较为适宜，它能保证平稳运行，费用也不太高；或者现有系统比较稳定，能够适应自身业务发展需要，或新旧系统转换风险很大（例如，在线订票系统、银行的中间业务系统等），也可以采用分段转换策略。

4、软件维护

在系统运行过程中，软件需要维护的原因是多样的，根据维护的原因不同，可以将软件维护分为以下 4 种：

①改正性维护。为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程就称为改正性维护。

②适应性维护。在使用过程中，外部环境（新的硬、软件配置）、数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）可能发生变化。为使软件适应这种变化，而去修改软件的过程就称为适应性维护。

③完善性维护。在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动称为完善性维护。

④预防性维护。这是指预先提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。通常，预防性维护可定义为“把今天的方法学用于昨天的系统以满足明天的需要”。也就是说，采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编码和测试。

九、项目管理（视频内容：系统分析-进阶知识模块中项目管理视频）

①关键路径

项目的工期，持续时间最长的路径，可以有一条或者多条。
②浮动时间
每个活动的松弛时间=最晚开始时间-最早开始时间或最晚结束时间-最早结束时间
其中最早时间可以采用“正推法”，最晚时间采用“逆推法”。