

案例分析答题模板册

—— 依据 16 套真题 75 大题 / 214 小问提取整理 ——

目 录

模块一	先看这页：得分逻辑 + 自评功能	5 条
模块二	Web 应用设计（19 大题 / 25%）	9 条
模块三	系统架构设计与评估（15 大题 / 20%）	10 条
模块四	系统设计与建模（12 大题 / 16%）	8 条
模块五	嵌入式系统架构设计（12 大题 / 16%）	8 条
模块六	数据库系统与缓存设计（10 大题 / 13%）	9 条
模块七	通用答题句式与时间分配	4 条

模块一 先看这页：案例题的得分逻辑

1. 案例题怎么给分

案例5 选 3，每题 25 分，共 75 分，合格线 45 分。阅卷按要点给分，不按字数给分。

⚠ 考试形态已改：综合知识与案例分析上午连考

- 08:30 - 12:30 综合知识 + 案例分析连考，总时长 240 分钟
- 综合知识：最长 150 分钟、最短 120 分钟
- 第一个科目结余的时长可以让给案例分析——做得快就是赚到
- 综合知识交卷后：不考案例的可直接离场；考案例的，考试结束前 60 分钟可交卷

- 按点给分：每个小问有固定采分点，答到关键词就给分，写得多不加分
- 写了错的不扣分：但写错的点会挤占时间，且暴露概念不清
- 先答结论再补理由：结论对了先拿分，理由只影响该点是否满分
- 字数是硬约束：写“200 字以内”就控制在 200 字内，超了可能不计分

★ “关键词命中 = 得分”。别写大段铺垫，直接上术语和结论。综合知识做得越快，案例分析时间越充裕——这是连考最大的红利。

2. 214 个小问的能力分布

搞清楚题目在考什么能力，才知道该准备什么。按题干问法统计：

能力类型	占比	典型问法
补充填空（把表/图填完）	39.7%	“将合适的质量属性填入图 1-1 的 (1) (2) 空白处”
设计方案（怎么做）	29.9%	“逻辑数据模型设计包含哪些任务？应包含哪些实体？”
概念说明（是什么）	26.2%	“说明时序数据库的含义”
方案选型 + 理由	17.8%	“管道-过滤器 vs 仓库，更适合哪个？为什么？”
风险/缺点列举	16.8%	“列举该四层架构可能存在的五类不足”

★ 最大头是补充填空（近 40%）——这类题的答案往往就在题干里，考的是“认得出”，不是“背得出”。

3. 案例 vs 选择题：两种考法

维度	综合知识（同场前半段）	案例分析（同场后半段）
问法	“XX 是什么”	“给你一段场景，判断/选择/设计”
答案来源	记忆	题干 + 知识，一半在材料里
答题形态	选项	填空 / 简答 / 表格补全
准备方式	背小册	练答题套路

★ 案例题不要死记，要记“这类问法该怎么起手”。

4. 三道题怎么选

5 道大题选 3 道作答，选题本身就是策略。

- 先快速扫每题的最后一问——问“设计/改进”的比问“计算”的好拿分
- 优先选题干短、图少的题：图越多，读图成本越高
- 避开没接触过的新技术（如某年的向量数据库、知识图谱），除非别无选择
- 选定后不要换：换题的时间成本远高于硬做

★ “三长一短选最短”——题干短的题，说明信息密度低、容易读。

模块二 Web 应用设计（19 大题 · 最大板块）

5. 配套练习：应用里已有「对答案自评」

应用（`http://127.0.0.1:8321`）的案例卷里，每个小问下方都有「 对答案自评」按钮——点开就是这道题的采分点清单，你可以边写边勾，实时看命中率。

- 全覆盖：16 套案例卷 214 个小问，每个都有采分点（共 1318 条，平均每题 6.2 条）
- 人工判定：采分点是我逐条读官方解析后判定的，不是关键词机械抽取
- 带雷区：每题附一句「本题最易丢分处」
- 勾选记忆：勾选状态存在浏览器本地，刷新不丢

- 并配「🤖 AI 批改思路」：写完思路可让 AI 逐点评，两者配合用

建议用法（每个小问 3 分钟）

1. 先自己写思路，不看任何提示
2. 点「对答案自评」，逐条勾你答到的点
3. 没勾上的，回题干找对应信息——这才是提分的地方
4. 看一眼雷区，确认自己没踩

★ 自评命中率低于 60% 的题，就是你这一类的薄弱点，回去补对应模块。

6. 这一块考什么

Web 应用设计是案例第一大板块（19/75 大题），高频三件套：缓存 / 高并发 / 微服务与消息队列。

子题型	出现次数	典型问题
Redis 缓存与一致性	最高	缓存穿透/击穿/雪崩、双写一致性、过期策略
Redis 主从与持久化	高	主从同步流程、RDB vs AOF
高并发与分布式锁	高	悲观锁 vs 乐观锁、秒杀、库存扣减
微服务与消息队列	中	服务拆分、异步解耦、幂等
负载均衡	中	四层 vs 七层、算法选择

★ 这一块几乎年年考缓存，Redis 两件事（一致性 + 持久化）必须能默写。

7. 缓存三大问题（必背）

缓存穿透、击穿、雪崩，名字像但成因完全不同，考题最爱让你区分。

问题	成因	解决方案
穿透	查不存在的数据，请求全打数据库	布隆过滤器、缓存空值
击穿	单个热点 key 过期瞬间大量请求涌入	互斥锁重建、逻辑过期
雪崩	大量 key 同时过期或 Redis 宕机	过期时间加随机、多级缓存、高可用

答题模板：解释某个缓存问题的解决方案

1. 先给问题下定义（一句话说清成因） 1分
2. 再给方案名称（布隆过滤器 / 互斥锁 / 逻辑过期） 2分
3. 最后说方案怎么起作用（拦截 / 串行化 / 兜底旧值） 2分

★ “穿不存在、击单个、雪一片”——三个字区分开，别答串。

8. 缓存与数据库一致性

缓存和数据库双写时，怎么保证不脏读。核心是**更新顺序**和**删除策略**。

- 先更新数据库、再删除缓存（Cache Aside）——最常用
- 为什么是删除不是更新：并发下更新缓存容易写入旧值
- 延迟双删：删缓存 → 更新库 → 再延迟删一次，应对主从延迟
- 强一致场景：加分布式锁 或 走 binlog 订阅（Canal）

雷区

- 写“先删缓存、再更新数据库”——并发下必然脏读，是最常见错误答案
- 写“更新缓存”而不是“删除缓存”——并发写会互相覆盖

★ “先库后删”四个字。问“能否强一致”，答“不能，只能最终一致，需要锁或 binlog”。

9. Redis 持久化：RDB vs AOF

维度	RDB（快照）	AOF（日志）
原理	定时把内存 全量 写成二进制快照	记录每条 写命令
恢复速度	快	慢（要重放）
数据完整性	差（两次快照间会丢）	好（可配置 fsync）
文件体积	小（压缩）	大
对性能影响	fork 时短暂卡顿	持续写盘开销

答题模板：持久化方案怎么选

1. 先说**两种各是什么**（一句话） **2分**
2. 按**四个维度对比**：恢复速度 / 完整性 / 体积 / 性能 **3分**
3. 给**生产建议**：两者混用（RDB 定期 + AOF 实时） **1分**

★ “RDB 快而全、AOF 细而慢”。生产上两个一起开，不是二选一。

10. Redis 主从同步

第一次全量同步、之后增量同步，两阶段。

- **第一阶段（全量）**：从库发 PSYNC → 主库 fork 生成 RDB 发送 → 从库载入
- **第二阶段（增量）**：主库把期间的写命令放入**复制积压缓冲区**，从库重放
- 断线重连：从库上报 offset，主库判断能否**部分重同步**（在缓冲区内）
- 哨兵（Sentinel）负责**故障转移**，不负责复制本身

★ “先 RDB 全量、再命令增量、断线看 offset”。三个动作对应三个阶段。

11. 悲观锁 vs 乐观锁

维度	悲观锁	乐观锁
思想	假设一定冲突，先加锁再操作	假设很少冲突，提交时校验版本
实现	数据库行锁、分布式锁（Redis/ZK）	版本号、CAS、时间戳
并发性能	差（串行化）	好（无阻塞）
失败处理	等待或超时	重试
适用	写多读少、冲突频繁	读多写少、冲突少

★ “悲观先锁、乐观后验”。高并发抢购用乐观锁 + 重试，或用 Redis 扣减。

12. 消息队列的作用与坑

消息队列解决异步、解耦、削峰三件事，代价是一致性、重复、顺序三件事。

收益	代价 / 风险
异步：不阻塞主流程	最终一致，不是强一致
解耦：生产消费互不感知	链路变长，排查困难
削峰：缓冲瞬时流量	消息堆积、积压延迟

幂等三件套（问“如何保证不重复消费”必答）

- 唯一业务 ID + 去重表
- 数据库唯一索引
- 状态机（只允许从某状态迁移一次）

★ “异步解耦削峰，换来重复和延迟”。问风险必答“重复消费、消息丢失、顺序错乱”。

13. 负载均衡四层与七层

层次	工作在哪	能做什么	代表
四层	传输层（TCP/UDP）	按 IP+端口转发，快	LVS、F5
七层	应用层（HTTP）	按 URL/Header 路由，灵活	Nginx、HAProxy

★ “四层快、七层灵活”。要按内容路由就七层，要极致性能就四层。

14. Web 类答题通用骨架

遇到"如何设计/如何优化"类小问，照这个顺序写

1. 先定位问题：这是读多写少？还是写多读少？（一句话）

1分

2. 给方案：缓存 / 异步 / 分片 / 读写分离，选 1-2 个说

3分

3. 说怎么实现：具体到组件（Redis、MQ、Nginx）和配置策略

2分

4. 补一句代价：一致性变弱 / 运维复杂 / 成本上升

1分

★ 四段式：定位 → 方案 → 实现 → 代价。有“代价”这一句，答案立刻显得专业。

模块三 系统架构设计与评估（15 大题）

15. 这一块考什么

这一块是案例里最“送分”的板块——考的都是必背小册里背过的概念，只是换成了填空和简答。

高频点	出现	问法
质量属性识别	14 次	“依次写出每条需求对应的质量属性”
效用树填空	9 次	“把 (a)~(m) 填入效用树 (1)~(6)”
架构风格选型	6 次	“更适合哪种风格？说明理由”
敏感点/权衡点	6 次	“指出哪些是敏感点、哪些是权衡点”

★ 这一块的答案基本在必背小册第 1-10 条里，背熟就能拿分。

16. 质量属性识别（最高频，14 次）

题干给一串需求，让你判断每条属于哪个质量属性。这是案例第一大送分题。

题干关键词	答案
“1 分钟内完成” / “响应时间不超过” / “TPS 达到”	性能
“99.9%” / “故障切换不超过 N 分钟” / “7×24 小时”	可用性
“加密” / “权限控制” / “防止未授权访问”	安全性
“不停机升级” / “更换数据库不影响业务” / “模块可替换”	可修改性
“新手上手时间” / “操作步骤减少”	易用性
“与第三方系统对接” / “标准协议”	互操作性
“单点故障不影响” / “数据不丢失”	可靠性

答题格式（按这个写，别写成一段话）

1. 需求（1）：**性能**
2. 需求（2）：**可用性**
3. ……逐条对应，一条一行

★ 看**动词**：能算时间的是性能、能算占比的是可用性、涉及权限的是安全性、涉及“改/换”的是可修改性。

17. 效用树填空（9 次）

给一张效用树图，挖几个空让你填。填空位置通常在**第 2 层（质量属性）**或**第 4 层（场景）**。

答题模板

1. 看清空在哪一层：第 2 层填**质量属性名**（性能/可用性/安全性/可修改性） **2分**
2. 第 4 层填**题干里的具体场景描述**（照抄题干原句更保险） **2分**
3. 若问“优先级”，格式是（**重要性，实现难度**），如（H，M） **1分**

雷区

- 根节点填“质量属性”——根是“效用”
- 把“战术”填到叶节点——叶是“场景”
- 第 2 层填了具体技术（如“Redis”）——那属于场景或战术，不是质量属性

★ “根效用、二属性、叶场景”。填空前先定位层号，再决定填什么。

18. 架构风格选型（6 次）

给两个方案让你选，并说明理由。这是**最好拿分的开放题**——只要理由扣住质量属性就行。

四步答题模板（照抄结构）

1. **表态**：“应采用 XXX 风格。” **1分**
2. **扣质量属性说理由**：“该系统强调……（性能/可修改性/可扩展性），而 XXX 风格的构件……（特点），能满足该要求。” **3分**
3. **说另一方案为何不合适**：“YYY 风格虽然……，但……（缺点），不满足……” **2分**
4. **补一句代价或前提**（可选） **1分**

题干信号	倾向选
数据要经过多道加工、每步可独立替换	管道-过滤器
多来源数据汇总、围绕中心数据操作	仓库 / 黑板
松耦合、异步、发布订阅	事件驱动
分层职责、上层依赖下层	分层
规则频繁变化、要动态执行	解释器

★ “先表态、再扣属性、后踩对方”。理由里必须出现质量属性词，否则不给分。

19. 敏感点 / 权衡点 / 风险点（6 次）

题干给一堆架构决策，让你分类。判据是**影响几个属性、有没有坏结果**。

类型	判据	题干信号
敏感点	影响一个质量属性	“对 XX 有显著影响”
权衡点	影响多个，且有好有坏	“提高 A 但降低 B”
风险点	可能导致不良后果	“存在隐患”“可能不可接受”
非风险点	经分析确认没问题	“已验证”“满足要求”

★ “一个=敏感、多个=权衡、可能坏=风险、已确认=非风险”。四句话对应四类。

20. ATAM 与评估方法（案例常考）

问“用什么方法评估 / 为什么选 ATAM”	
1. 点名方法：ATAM（架构权衡分析法）	1分
2. 说它管什么：多质量属性权衡，不只是单一属性	2分
3. 说产出：效用树 + 四类点（敏感/权衡/风险/非风险）	2分
4. 说前提：架构尚未落地时评估，需要干系人参与	1分

★ ATAM 三件套：多属性、效用树、四类点。缺一个理由就不完整。

21. DDD 与限界上下文（近年新增）

2025-11 起考 DDD。限界上下文是高频填空点。

- DDD 基本思想：以领域模型为核心，业务与技术分离，用通用语言统一沟通
- 限界上下文：一个领域模型的边界，边界内术语含义唯一，边界外可同名不同义
- 作用：划分微服务边界、避免模型歧义、支持团队并行
- 与微服务关系：限界上下文是拆分的依据，但一个上下文不一定等于一个服务

★ “同一个词，在不同上下文里可以有不同含义”——这句话就是限界上下文的定义核心。

22. 六边形架构（2026-05 真题）

又称**端口与适配器**架构。用端口定义内外交互契约，用适配器隔离技术实现，业务核心不依赖框架。

组成	职责	举例
领域核心	业务规则（聚合、实体、值对象、领域服务）	订单、库存规则
进站端口	定义外部调用核心的用例接口	创建订单、确认库存
出站端口	定义核心访问外部的抽象	订单仓储、支付网关、消息发布
适配器	把 HTTP/RPC/MQ/DB 转成端口调用	Controller、Repository 实现

问"依赖方向"（最常考）

1. 外部依赖核心，核心不依赖外部 2分

2. 核心只依赖端口抽象，不依赖 Spring MVC / MyBatis / Redis SDK 2分

3. 靠依赖倒置实现：适配器实现端口接口，运行时注入 1分

★ “核心在内、适配器在外、依赖由外向内”。三层九个字。

23. 架构评估类答题骨架

遇到"评估某系统架构"类小问

1. 列质量属性：从题干需求里抽 3 - 5 个 2分

2. 建场景：每个属性给一个“刺激源+刺激+响应+度量” 2分

3. 找点：指出敏感点/权衡点/风险点 2分

4. 给建议：针对风险点提改进 1分

★ 四步：属性 → 场景 → 找点 → 建议。这就是 ATAM 的简化版。

24. 这一块的送分清单

以下概念能默写，这一板块就能拿 15+ 分：

- 质量属性场景六要素：刺激源、刺激、环境、制品、响应、响应度量
- 效用树四层：效用 → 质量属性 → 属性分类 → 场景
- 四类点定义与判据
- ATAM 四阶段九步（知道第 5 步生成效用树）
- SAAM / ATAM / CBAM 三者的差别（单属性 / 多属性 / 加成本）
- 六边形架构四个组成 + 依赖方向
- 限界上下文定义

★ 这七条全在必背小册第 1 - 10、21 条里。案例和选择题在这里是同一套知识，背一遍两处都能用。

模块四 系统设计与建模（12 大题）

25. 这一块考什么

这一块的主线是**数据流图（DFD）**（36 小问里 8 次）和**E-R 建模**，属于“画图 + 补图 + 找错”型题目。

子题型	典型问法
DFD 补全	“在图中补充缺失的数据流 / 加工 / 数据存储”
DFD 找错	“指出图中语法错误并说明错误类型”
E-R 建模	“设计超类实体”“指出派生属性”“给出主键”
逻辑模型	“逻辑数据模型设计包含哪些任务”

★ 这一类题答案高度依赖图，但“错误类型”和“设计任务”是固定知识，可以提前背。

26. DFD 四要素与常见错误

数据流图只有四种元素：**外部实体、加工、数据存储、数据流**。

元素	画法	作用
外部实体	矩形	系统外的数据来源或去向
加工（处理）	圆角矩形 / 圆	对数据做变换
数据存储	开口矩形 / 双横线	静态存储的数据
数据流	箭头	数据流动方向

DFD 语法错误（考“找错”必答这几类）

- **数据流方向错**：数据流不能从存储直接到外部实体而不经加工
- **黑洞加工**：有输入无输出
- **奇迹加工**：有输出无输入
- **灰洞加工**：输入不足以产生输出
- **父子图不平衡**：父图的输入输出与子图不一致

★ “黑洞无出、奇迹无入、灰洞不够、父子不平”——四类错误一句话记完。

27. 分层数据流图

DFD 按层次细化：**顶层图（0 层）→ 0 层图 → 1 层图……**

- **顶层图**：整个系统就是一个加工，只画外部实体和进出数据流
- **0 层图**：把系统拆成若干主要加工

- **平衡原则**：子图的输入输出必须与父图对应加工一致
- **数据字典**：定义所有数据流和存储的组成，**贯穿各层**

★ “顶层一个圈，层层往下钻，父子要平衡”。

28. E-R 建模：超类与派生属性

这两个概念是案例爱考的“名词解释 + 应用”组合。

概念	定义	应用要点
超类实体	把多个实体的 公共属性 抽出来形成的父实体	列出公共属性 + 说明子实体继承
派生属性	可由 其他属性计算 得出，不必单独存储	如“单价×数量=金额”中的金额
弱实体	不能独立存在，依赖强实体	如“订单明细”依赖“订单”

★ “超类抽公共、派生靠计算、弱实体靠别人”。三个词一句话。

29. 逻辑数据模型设计四任务

经典简答题：“逻辑数据模型设计过程包含哪些任务？”

- ① **实体识别**：从需求中找出实体
- ② **属性定义**：确定每个实体的属性
- ③ **关系确定**：确定实体间联系与基数
- ④ **规范化**：消除冗余，达到 3NF/BCNF
- （常加一句）**确定主键与外键**

★ “找实体、定属性、连关系、再规范”四步，加上主外键更完整。

30. 需求到架构的映射

案例考过：“如何把软件需求映射到软件架构？”从三个角度答。

角度	答什么
描述语言	用 ADL / UML 把需求转成架构描述
非功能需求	把质量属性转成 质量属性场景 ，再落到架构决策
需求与架构的对应	建立 需求-架构 追踪矩阵，逐条验证

★ 三角度：**语言、场景、追踪**。这是标准的三段式答案。

31. 建模类答题骨架

遇到"补图 / 找错 / 建模"类小问

1. 先抄题干里的名词：题干给你的实体/数据流名称，答案基本就在里面

2分

2. 按标准元素命名：不要自创叫法，用"XX 信息""XX 表"

1分

3. 找错题先分类再举例：先说错误类型，再说图中哪一处

3分

4. 检查平衡性：DFD 必查父子平衡，ER 必查基数

1分

★ 补图题的答案90% 能从题干原文里找到，先通读题干划名词，再往图上套。

32. 这一块的送分清单

- DFD 四要素与四类错误
- 分层 DFD 的平衡原则
- 超类实体 / 派生属性 / 弱实体三个定义
- 逻辑数据模型设计四任务
- E-R 基数 (min, max) 的读法

★ 这五条是这一板块的“通用弹药”，任何一年都能用上。

模块五 嵌入式系统架构设计（12 大题）

33. 这一块考什么

嵌入式是案例里最“套路化”的板块——考来考去就是实时性、可靠性、调度、冗余这几件事。

高频点	出现	问法
实时性保障	5 次	“如何保证系统的实时性”
可靠性设计	5 次	“采取哪些可靠性措施”
模块划分	5 次	“系统应划分为哪些模块”
冗余与容错	—	“如何避免单点故障”
看门狗 / 中断	—	“看门狗的作用”

★ 这一块背模板就能拿分，因为问法高度重复。

34. 实时性保障六招（必背）

问“如何保证实时性”，从这六个角度答，答到四个就够。

- 实时操作系统（RTOS）：用抢占式调度，保证高优先级任务及时响应
- 优先级调度：关键任务设高优先级，配合优先级继承防反转

- **中断优化**：中断服务程序尽量短，耗时工作交给任务
- **资源预留**：关键任务独享 CPU / 内存，避免争抢
- **减少阻塞**：避免长临界区、避免动态内存分配
- **时间确定性**：用静态分析或 WCET 估算保证最坏响应时间

★ “RTOS + 优先级 + 短中断 + 预留 + 少阻塞 + 算 WCET”六招。答四条即可。

35. 可靠性设计五招（必背）

- **冗余**：硬件双机热备、软件 N 版本编程、数据多副本
- **容错**：故障检测 + 隔离 + 恢复（看门狗、心跳）
- **降级**：故障时关闭非核心功能，保核心业务
- **故障注入与测试**：提前暴露问题
- **日志与追溯**：故障可定位、可复盘

看门狗的作用（高频单点）

- 程序“跑飞”或死锁时，看门狗定时器超时，强制复位系统
- 本质是故障检测 + 自动恢复，防的是“程序挂死不响应”

★ “冗余、容错、降级、注入、日志”五招。看门狗 = 定时喂狗，不喂就复位。

36. 嵌入式分层与模块划分

问“系统划分为哪些模块”，按**硬件抽象 → 系统 → 应用**三层答。

层	模块
硬件层	传感器、执行器、通信接口、电源管理
驱动 / 抽象层	设备驱动、BSP、HAL
系统层	RTOS、文件系统、通信协议栈、存储管理
应用层	业务逻辑、数据处理、人机界面

★ “硬件 → 驱动 → 系统 → 应用”四层。分层答题永远比罗列模块得分高。

37. 硬实时 vs 软实时

类型	超时后果	例子
硬实时	灾难性，绝对不允许超时	刹车、飞控、心脏起搏器
软实时	体验下降，可接受偶尔超时	视频播放、界面刷新
固实时	偶尔超时可容忍，但超太多就失效	工业控制

★ “硬实时超时=事故，软实时超时=卡顿”。判断题看到“必须在 X ms 内完成”就是硬实时。

38. DO-178B 安全等级

航空机载软件的安全等级标准，分 A~E 五级。

等级	失效影响
A（灾难级）	可能导致坠机，要求最严
B（危险级）	严重危害，机组难以处理
C（较大级）	增加机组负担
D（较小级）	轻微不便
E（无影响）	无安全影响

★ “A 最严、E 无影响”。A 级对应“灾难”，别记反。

39. 中断与调度

嵌入式的两个高频机制。

- 中断处理五步：请求 → 判优 → 响应（保存现场）→ 服务 → 返回（恢复现场）
- 中断服务程序要短：只做紧急处理，耗时工作交给任务
- 抢占式调度：高优先级任务可打断低优先级，保证实时性
- 优先级反转：低优先级持锁挡住高优先级，用**优先级继承**解决

★ “中断短、调度抢占、反转继承”三个短句。

40. 嵌入式答题骨架

遇到“如何设计这个嵌入式系统”

1. 先分层：硬件 → 驱动 → 系统 → 应用 2分

2. 说实时：RTOS + 优先级 + 短中断 2分

3. 说可靠：冗余 + 看门狗 + 降级 2分

4. 说接口：通信协议（CAN/串口/以太网）与外部设备 1分

★ 四段：分层 → 实时 → 可靠 → 接口。嵌入式案例基本都能套。

模块六 数据库系统与缓存设计（10 大题）

41. 这一块考什么

这一块两条主线：**缓存设计**（12 次）和**数据一致性**（7 次），与 Web 板块有重叠但更偏数据库侧。

子题型	典型问法
缓存策略	“如何设计缓存？用什么淘汰策略？”
读写分离 / 分库分表	“数据量增长后如何扩展？”
分布式事务	“如何保证跨库一致性？”
范式与索引	“判断范式”“为什么建这个索引”

★ 这一块和必背小册模块九（数据库）是同一套知识，只是换成案例问法。

42. 读写分离与主从延迟

读写分离解决读多写少，代价是主从延迟导致读到旧数据。

- 架构：主库写，从库读，主从异步复制
- 延迟问题：刚写完立刻读从库，可能读到旧值
- 解决：写后强制读主库 / 延迟读 / 半同步复制
- 适用：读远多于写（如 10:1 以上）

★ “主写从读，写后读主”。被问“延迟怎么办”，答这八个字。

43. 分库分表

方式	拆什么	解决
垂直分库	按业务拆库	业务耦合、单库压力
垂直分表	按字段拆表（冷热分离）	单表字段过多
水平分表	按行拆（取模/范围/哈希）	单表数据量过大

分库分表带来的问题（必答）

- 跨库 JOIN 困难
- 分布式事务一致性
- 全局唯一 ID（雪花算法）
- 分页、排序、聚合变复杂

★ “垂直拆业务、水平拆数据”。问代价必答“JOIN、事务、ID、分页”四样。

44. 分布式事务三种方案

方案	一致性	特点
2PC / XA	强一致	两阶段提交，阻塞、性能差
TCC	最终一致	Try-Confirm-Cancel，业务侵入大
Saga / 消息最终一致	最终一致	长事务、补偿，适合微服务

★ “要强一致用 2PC，要高可用用最终一致”。答题先问“能不能接受中间态”。

45. 索引与范式

案例里常以“为什么慢 / 该怎么建索引”出现。

- **B+ 树索引**：范围查询友好，是 MySQL 默认
- **最左前缀**：联合索引 (a, b, c)，查 a、a+b、a+b+c 都能用，跳 a 不行
- **索引失效**：函数运算、隐式转换、前导模糊 (%xx)、OR 连接非索引列
- **范式**：1NF 原子、2NF 消部分依赖、3NF 消传递依赖、BCNF 全候选键

★ “最左前缀、前导模糊失效”。问“为什么索引没生效”，先答这两条。

46. 缓存淘汰策略

策略	淘汰谁
LRU	最久未使用
LFU	使用次数最少
FIFO	最早进入
TTL / 过期删除	到期即删（惰性 + 定期）
maxmemory-policy	Redis 可配 allkeys-lru / volatile-lru 等

★ “LRU 看时间、LFU 看次数”。Redis 默认 noeviction，要显式配淘汰策略。

47. 数据一致性等级

案例常问“如何保证一致性”，先分清**强一致** / **最终一致**。

- **强一致**：任何时刻读到的都是最新值（2PC、锁）
- **最终一致**：短暂不一致，最终会同步（消息、补偿、对账）
- 大多数互联网场景接受**最终一致**，换取可用性
- BASE 理论：基本可用、软状态、最终一致

★ “强一致牺牲可用，最终一致牺牲实时”。答题先亮这个取舍。

48. 数据库设计四步

问“数据库设计过程”，标准四步。

- ① **需求分析**：产出数据字典、数据流图
- ② **概念结构设计**：产出 E-R 图
- ③ **逻辑结构设计**：E-R 转关系模式，规范化
- ④ **物理结构设计**：索引、分区、存储参数

★ “需求 → 概念 → 逻辑 → 物理”四步，每步的产物要能对上（E-R 图在概念阶段）。

49. 数据库类答题骨架

遇到"数据量大了怎么办 / 如何优化"

1. 先定位瓶颈：读多还是写多？单表多大？

1分

2. 纵向先做：索引优化、SQL 优化、字段精简

2分

3. 横向再扩：读写分离 → 分库分表 → 引入缓存

3分

4. 补一致性代价：主从延迟、分布式事务

2分

★ “先索引、再分离、后分片、补缓存”——这个顺序不能反，先上分库分表是过度设计。

模块七 通用答题句式与时间分配

50. 万能答题句式

案例题有些句式，写上去就能拿“表述分”。

- 定义式：“X 是指……，其核心是……”
- 方案式：“可采用……方案，具体做法是……，优点是……，代价是……”
- 对比式：“A 侧重……，B 侧重……；场景 X 应选 A，因为……”
- 风险式：“该方案存在……风险，可通过……缓解”
- 兜底句（实在不会）：“该设计需结合具体业务规模与一致性要求权衡”

★ 五种句式对应五种问法。不会的题也别空着，用兜底句至少拿 1 分。

51. 时间分配：连考 240 分钟怎么切

上午 08:30 - 12:30 两科连考共 240 分钟，综合知识最长 150 分钟、最短 120 分钟。核心策略是综合知识尽快交卷，把时间让给案例分析。

方案	综合知识	案例分析	适用
推荐	120 分钟（最短）	120 分钟	综合知识有把握、案例需要时间
平衡	135 分钟	105 分钟	两科都不稳
保守	150 分钟（最长）	90 分钟	综合知识必须反复检查

案例 120 分钟的切法

阶段	时间	做什么
选题	8 分钟	扫 5 道题的最后一问，定 3 道
答题	每题 34 分钟 × 3	按小问顺序，先易后难
检查	10 分钟	补漏、检查字数、确认没有空题

连考的三个坑

- **综合知识恋战**：卡在一两道题上耗到 150 分钟，案例只剩 90 分钟——这是最常见的翻车方式
- **忘记时间会结转**：综合知识提前交卷省下的时间是给案例用的，不是白省
- **中途不可逆**：综合知识一交卷就回不去了，交之前把该检查的检查完

★ “综合知识 120 分钟必须交卷”。多花 10 分钟检查综合知识，可能换来案例少答半道题——20 分和 2 分，孰轻孰重很清楚。

52. 五个高频踩分动作

- **分点作答**：一条一行，不写大段文字
- **先结论后理由**：结论单独成句
- **用题干原词**：题干叫什么你就叫什么，别换说法
- **控制字数**：写了“200 字以内”就数一下
- **每题都填**：宁可写错也别空，空题 0 分

★ 这五条与知识无关，但**能实打实多拿 5 - 8 分**。

53. 最后：案例与选择题的分工

要背的	在哪本
质量属性、架构风格、评估方法	必背小册 1 - 10 条
数据库范式、E-R、连接运算	必背小册 55 - 61 条
OS、网络、嵌入式基础	必背小册 71 - 94 条
答题套路与踩分点	本册

★ 两册分工：**必背小册管“知道”，本册管“答得出”**。知识在必背册，套路在这里。

编制说明：本册依据 `quiz/quiz.db` 中 16 套案例真题（75 大题 / 214 小问）与官方解析编制，重点覆盖占 91% 题量的五大板块（Web 应用设计 19、系统架构设计与评估 15、系统设计与建模 12、嵌入式系统架构设计 12、数据库系统与缓存设计 10）。

与必背小册的分工：必背小册解决“概念是什么”（选择题 + 案例的概念题），本册解决“案例怎么答”（答题骨架 + 踩分点 + 雷区）。两册配合使用，不重复。

使用建议：先通读模块一（得分逻辑）与模块七（通用句式）；再按你最弱的一类精读对应模块；考前只看每条的 ★ 记忆技巧 与 答题模板。