



NEW PLAN

软件高级架构师

• 一段新征程 •



时间段	考试科目	考试时长	注意事项
上午	综合知识、案例分析	总计240分钟 -综合知识：120-150分钟 -案例分析：剩余时间	综合知识科目作答时间最长150分钟，最短120分钟。 完成综合知识科目后，可选择继续作答案例分析科目。 考试结束前60分钟可交卷离场。
下午	论文	120分钟	论文科目考试时长为120分钟，不得提前交卷离场。

注意：人员只准携带黑色、蓝黑签字笔、钢笔或圆珠笔、2B铅笔、削笔刀、橡皮、直尺等进入考场，开考后不得相互借用文具。人员不得携带任何书籍、笔记、纸张、计算器、报刊、手机、电子设备及通讯工具等考试中禁止使用的物品进入考场。中途可以上厕所。每科及格分数为45分，一科不过，全部重考。从2024年开始，一年考2次

时间	所属范围	考察知识点
2026年5月 试题1	系统架构设计	质量属性+质量属性六要素(12分)+架构风格选择(6分)+架构风格的开放式问答题（7分）
2026年5月 试题2	数据库（秒杀场景）	悲观锁分类及含义和特点（12分）+乐观锁和悲观锁的优缺点及乐观锁的实现方式（6分）+高并发场景下redis的优化方式（7分）
2026年5月 试题3	前沿技术（智能辅导与学习路径推荐系统）	根据流程补充事件（12分）+冷启动（6分）+路径推理和训练数据的开放式问答题（7分）
2026年5月 试题4	物联网+边云协同	感知层、边缘层、AI决策层、应用层的主要职责（4分）+选词填空（11分）+结合云边协同特点,分析该四层架构的5个主要缺点（7分）
2026年5月 试题5	物联网+web架构	层次架构图的选词填空（12分）、时间序列数据+时序数据库（6分）+MQTT的服务质量(Quality of Service 缩写 QoS) 级别（7分）
2025年11月 试题1	系统架构设计	质量属性(11分)+质量属性六要素(8分)+质量属性描述的问答题（6分）
2025年11月 试题2	web应用	基于领域驱动设计（DDD）的限界上下文概念题（6分）+web结构选词填空（11分）+网络故障对端云同步的影响（8分）
2025年11月 试题3	数据库	redis做缓存时，面对缓存过期的两种解决方案，完成填图，一道是互斥锁方案，一道是逻辑过期方案，最后一道是分析这两种解决方案的优缺点
2025年11月 试题4	嵌入式	智能化操作系统的实质及功能介绍（9分）+OS for AI与AI for OS的差异（9分）+轻量化内核的特点（7分）
2025年11月 试题5	系统分析	购物平台流程的四个阶段活动（6分）、完善petri网的商品入库流程（10分）+填写petri网的关键矩阵+问答题（9分）
2025年5月 试题1	系统架构设计	质量属性(12分)+架构风格填空(6分)+解释器风格问答题（7分），架构风格填空是书本上的架构图原图
2025年5月 试题2	全新题材	知识图谱分层架构填空（9分）+Scrapy框架填空并简述异步io的特点和优势（6分）+知识图谱的构建和智能问答存储的实现（10分）
2025年5月 试题3	数据库	redis全量同步填空（10分）+增量复制填空（6分）+数据持久化的两种方式及优缺点（9分）
2025年5月 试题4	前沿技术	云端ai和侧端ai的问答+ 填空+填表和回答集中化资源池的三个缺陷
2025年5月 试题5	前沿技术	区块链的六个层次及介绍+信息填写人员、核对人员、审核人员的操作流程+智能合约是什么并说明智能合约包含哪三方面

时间	所属范围	考察知识点
2024年11月 试题1	系统架构设计	质量属性填空（14分）、ping/echo和心跳策略是怎么进行故障检测、从资源利用率的角度来看为啥心跳更好（11分）
2024年11月 试题2	数据库	cache-aside（旁路缓存）架构的两道填空题（核心思想就是旁路缓存的工作流程）（10+6分）、多线程进行读写操作的一致性风险，问你如何解决这个问题（9分）
2024年11月 试题3	嵌入式	机器人操作系统ROS的定义和特点，ROS2和ROS1的区别、四种通信服务选词填空
2024年11月 试题4	web应用	基于es的分词推荐系统中几个分词引擎的区别（Standard,Simple,Whitespace,Keyword）6分、web服务器架构图的填空（12分）、restful架构的特点及如何实现前后端分离（7分）
2024年11月 试题5	软件工程	危险驱动的安全分析的4个步骤，及简要说明（10分）、胰岛素系统填空（9分）、形式化开发和软件测试技术各自的特点（6分）
2024年5月 试题1	系统架构设计	微服务和单体架构对比、质量属性效用树、质量属性六要素
2024年5月 试题2	软件工程	UML序列图的消息、序列图填空、序列图和协作图对比、分支序列片段
2024年5月 试题3	数据库	基于mysql的分布式锁、redis的分布式锁的死锁场景、redis命令
2024年5月 试题4	嵌入式	some/ip协议、DDS协议、规控AP模块的流程框图
2024年5月 试题5	系统架构设计	架构图填空、MongoDB 矢量化存储、热数据、温数据和冷数据存储的原因
2023年11月 试题1	系统架构设计	大数据架构Lambda和Kappa
2023年11月 试题2	软件工程	SysML需求图和用例图、需求图七类关系等
2023年11月 试题3	数据库	读写分离架构、redis缓存、主从复制
2023年11月 试题4	Web应用	Hibernat架构、数据持久层、jwt
2023年11月 试题5	嵌入式	数字孪生概念、技术选择、架构图填空



根据**2023年-2026年**全批次真题考点复盘，结合软考高级命题组的出题偏好、科目定位调整，我们将架构案例分析真题规整为七大核心模块，同步梳理各模块考情、命题规律、未来预判和备考注意事项如下：

一、系统架构设计（固定必答题，核心拿分题）

- 该题型为案例分析中唯一必答题、每年必考1道，是全场最容易拿高分、必须牢牢把握住的题目，属于考生的保底分题型。除2023年单次考到教材下篇的大数据架构之外，近三年其余所有批次，本题均固定围绕核心架构基础考点出题，基本都是围绕着【质量属性 + 场景 + 效用树 + 架构评估】来出题。往往是给出一个行业背景（政务、工业互联网、金融、充电平台等）→ 识别场景中的质量属性→ 场景六要素填空→ 绘制 / 补全质量效用树→ 选择 ATAM/SAAM 评估并说明步骤→ 分析质量属性冲突与架构权衡策略或者是架构风格的提问
- 老师预估，第一题是所有案例分析题目里，唯一紧贴教材、极少超纲的题目，出题素材大概率源自课本核心章节；不过结合2024年11月、2026年5月的考试来看，本题也会零星出现教材之外、但属于架构常识类的拓展知识。因此大家务必警惕，要把架构核心板块的专业名词、定义、适用场景、设计策略吃透理解。

二、软件工程基础（早年高频，近年彻底淡出独立命题）

- 早年该模块几乎每年必考1道选做题，核心考点集中在面向对象设计、结构化设计等传统软件 Engineering 内容。但从近两年开始，这类题目不仅出题愈发灵活、超纲概率大幅升高，即便全力备考，也仅有一半左右的分值落在常规学习范围内。
- 比如2023年考到超纲的SysML图，2024年5月考了UML序列图的细节冷门考点，2024年11月考了风险驱动相关的偏门知识，这三次考试均出现明显超纲情况；更关键的是，2025年5月、11月两次考试，完全没有考查软件工程相关题目，彻底取消了该模块的独立命题，属于命题方向的重大调整，老师觉得单纯考软工的概率会越来越低。



三、数据库（高频选做题，稳分主流考点）

- 该模块是案例选做题中的高频主流题型，近三年考频极高，几乎年年出现，考点紧贴企业实战架构，不考纯理论，全是落地应用相关内容。核心考查范围涵盖：关系型数据库、内存数据库、NoSQL非关系型数据库的选型对比，反规范化设计、主从复制、读写分离、分库分表、分布式一致性、缓存容错、负载均衡等工程化考点。
- 老师明确预判：非关系型数据库是本题的必考核心，Redis相关考点会持续绑定考查，重点围绕缓存三剑客、分布式锁、持久化机制等高频点出题。后续考试还会逐步拓展多模数据库、湖仓一体、数据分片等新考点，大家一定要重点吃透关系型与非关系型数据库的组合应用、选型依据、场景适配，这是本题的核心得分突破口。

四、嵌入式系统（低频边缘化，冷门避坑题型）

- 该模块属于选做题，早年考频尚可，近年已经大幅降温，并非每年必考，属于边缘化冷门题型。核心仅考查嵌入式系统的实时性、可靠性、容错机制、嵌入式架构等基础概念，考题极易涉及嵌入式领域的小众专有技术，跨领域难度大，陌生考点多，备考性价比极低。
- 老师预判：后续考试该模块考频会持续走低，甚至逐步彻底淘汰，即便出题，也会偏向陌生冷门技术。建议考生备考不用深耕，考场遇到完全没接触过的技术，直接放弃本题，不浪费答题时间，优先选择其他熟悉的选做题。



五、Web应用开发（偶发选做题，融合考查为主）

- 本模块极少单独命题，主要结合系统架构、数据库等考点综合考查，属于偶发出题的选做题型，考频很低。核心涉及Web主流架构、前后端分离、网关、负载均衡、Web安全等相关技术，偶尔会涉及少量冷门新技术，出题灵活无固定套路。
- 老师预判：后续该模块依旧保持低频次、融合式的出题特点，不会成为主流考题。考场遇到完全陌生的技术考点可直接放弃；如果题目有基础填空题、或是有把握的小设问，再结合其他选做题的难度，按需选择作答即可。

六、新兴前沿技术（核心选做题，未来命题大势）

- 这类题材并非2025年新增，早年只是零星涉及，从2025年开始正式升级为案例核心选做题，单批次甚至考查两道，彻底取代软件工程、嵌入式等旧题型，成为未来的主流命题方向。老师预判：后续考试会固定设置1-2道前沿技术选做题，成为选做题中的主力，命题趋势不可逆。
- 出题方向主要分为两类：一是教材中未来信息技术章节的固有考点，二是当下行业热门的落地技术，出题难度适中，不考偏难怪，侧重考查核心原理、架构选型、应用场景、落地思路。
- 重点预判方向包括：云原生、Serverless、大模型应用架构、隐私计算、区块链、微服务治理、数据安全等，备考时既要吃透教材章节，也要紧跟行业热点，弄懂原理和架构应用逻辑，就能轻松拿分。



七、分布式与高并发架构（遗漏补全，后端核心高分题）

- 本模块之前总是作为web应用中的一个小题，但是近年来开始为独立选做题，聚焦分布式服务调度、流量管控、集群容错、高并发架构设计，是专门针对后端服务架构、分布式治理能力考查的题型，贴合后端开发实战。
- 核心考点完全聚焦服务层与流量层：CAP与BASE理论、分布式服务治理、负载均衡（服务层）、限流熔断降级、服务雪崩防护、微服务拆分与通信、分布式协调、秒杀流量削峰、异步解耦、集群弹性扩容、服务网关管控。
- 老师预判：后续考试会固定单独出题，成为选做题主力，考点稳定。



【后端考生建议选题】

一、第一题必须拿高分，低容错

系统架构设计：唯一必答题，考点固定、紧贴教材，纯理论记忆+套用答题模板即可拿高分，不用碰实战盲区，是全场最稳的送分题，必须重点攻克，保证正确率。

二、首选高分题

1. 分布式与高并发架构：完全贴合后端日常工作，考点都是日常用到的分布式、缓存、微服务、限流降级知识，出题直白、无冷门偏点，不用额外死记硬背，是后端同学的优势题，优先级最高。

2. 数据库系统：后端核心技能栈，关系库、NoSQL、Redis、主从复制、读写分离都是常用技术，考点实战、易理解，只要吃透常规知识点，很容易拿高分，重点深耕。

三、次选稳妥题

新兴前沿技术：优先选云原生、微服务治理、大模型后端部署这类贴合后端的方向，避开区块链、隐私计算等偏陌生的方向，考点浅显、侧重原理和应用，不用深挖底层，备考性价比高。



1

系统架构设计

- 1.简述微服务架构对比单体架构的优缺点。(7分)
- 2.质量属性及其场景（质量效用树）， 填空6个。(6分)
- 3.用质量属性6要素描述e)和h)两条可用性的场景描述。(12分)



【问题1】(6分)

答：微服务架构是一种分布式系统架构，将一个应用程序拆分为一组小型、独立的服务，每个服务都围绕特定的业务功能构建，并通过轻量级通信机制进行通信。相比之下，单体架构将整个应用程序作为一个单一的单元构建和部署。

微服务架构的优点：

- 灵活性和可扩展性：每个微服务都是独立的，可以独立部署和扩展，使系统更具弹性。
- 技术多样性：每个微服务可以使用不同的技术栈，使开发团队可以选择最适合其需求的技术。
- 易于理解和维护：微服务的小型化和聚焦性使得代码更易于理解、开发和维护。

微服务架构的缺点：

- 复杂性：微服务架构涉及到分布式系统，需要处理分布式事务、服务发现、服务治理等复杂问题。
- 部署和测试：由于微服务的数量增加，部署和测试变得更加复杂。
- 运维成本：微服务架构需要更多的运维工作，包括监控、日志收集、故障排查等。

【问题2】(7分)

就是一个质量效用树，忘了把哪几个空挖了，反正考察安全性，可用性，功能性，可修改性



试题答案

【问题3】(12分)

答：质量属性6要素描述：

e)可连续运行时间不少于240h，断电或故障后10s内应重启

刺激源：断电或故障

刺激：系统故障或断电

制品：系统

环境：运行环境

响应：重启

响应度量：10秒内

h)网络失效后，10s内应发起重新连接

刺激源：网络失效

刺激：网络失效

制品：系统

环境：网络环境

响应：重新连接

响应度量：10秒内

质量属性场景是一种用于描述系统如何满足特定质量属性需求的情境或情景。它由6部分组成：

- 刺激源(谁)：这是某个生成该刺激的实体(人、计算机系统或者任何其他刺激器)。
- 刺激(做什么)：该刺激是当刺激到达系统时需要考虑的条件。
- 环境(在什么样的环境下)：该刺激在某些条件内发生。当激励发生时，系统可能处于过载、运行或者其他情况。
- 制品(对哪个功能)：某个制品被激励。这可能是整个系统，也可能是系统的一部分。
- 响应(得到什么反馈)：该响应是在激励到达后所采取的行动。
- 响应度量(对反馈进行度量)：当响应发生时，应当能够以某种方式对其进行度量，以对需求进行测试。

可修改性质量属性场景描述实例：

场景要素	可能的情况
刺激源	最终用户、开发人员、系统管理员
刺激	希望增加、删除、修改、改变功能、质量属性、容量等
环境	系统设计时、编译时、构建时、运行时
制品	系统用户界面、平台、环境或与目标系统交互的系统
响应	查找架构中需要修改的位置，进行修改且不会影响其他功能，对所做的修改进行测试，部署所做的修改
响应度量	根据所影响元素的数量度量的成本、努力、资金；该修改对其他功能或质量属性所造成影响的程度

【问题1】(14分)

质量属性与功能性填空(给出场景填质量属性/功能性)

- a.系统发生业务故障时，3秒内启动XXX，属于可靠性
- b.系统可以数据进行导出，要求在3秒内完成数据导出，属于可用性
- c.质量属性描述，XXX，属于（ ）
- d.质量属性描述，XXX，属于可用性
- e.质量属性描述，XXX，属于（ ）
- f.系统需要增加新的模块，需要3天完成，属于（ ）
- g.系统分为三个不同的国家语言，完成xXXX功能，属于（ ）
- h.系统采用了云服务器，要求1分钟以内检测错故障，并且1个小时以内恢复
- i.系统使用XX检测云服务器故障，需要在2秒切换，

a	功能性
b	可靠性
c	(1)
d	(2)
e	(3)
f	
g 操作不同任务导入支持不同模型	
h	可用性

【问题2】(11分)

根据h(系统采用了云服务器，要求1分钟以内检测出错误故障，并且1个小时以内恢复)中描述，为了保证高可靠性，有ping/echo，心跳机制两种策略，从资源利用率问为啥心跳更好。王工觉得使用ping/echo好，但李工觉得从系统资源利用率来说，使用心跳机制好

- 1、针对质量属性可以使用ping/echo和心跳模式实现，分别简述ping/echo和心跳的实现原理。
- 2、张工认为心跳策略比较适合，简述为什么？

第一问:

Ping/Echo: 客户端定期向服务器发送请求 (如HTTP), 服务器需响应。若无响应则判定故障 (主动探测, 双向通信)。

心跳机制: 服务器定期向监控端发送状态信号 (如UDP包), 监控端超时未收到信号则判定故障 (服务端主动, 单向通信)。

第二问:

张工支持心跳机制的原因 (资源利用率)

网络带宽: 心跳仅发送小数据包 (如1个UDP包/次), Ping/Echo需双向交互 (请求+响应), 流量翻倍。

服务器负载: 心跳仅需简单发包, Ping/Echo需处理请求并返回数据, 消耗更多CPU和I/O资源。

客户端压力: 心跳无需客户端主动轮询, 仅需监听信号, 避免高频探测导致的线程阻塞。

结论: 心跳机制资源占用更低, 适合大规模云服务器场景。



2

软件工程

阅读以下关于软件系统设计与建模的叙述，在答题纸上回答问题1至问题3。

【说明】

煤炭生产是国民经济发展的主要领域之一，其煤矿的安全非常重要。某能源企业拟开发一套煤矿建设项目安全预警系统，以保护煤矿建设项目从业人员生命安全。本系统的主要功能包括如下(a)~(h)所述。

- | | | | |
|-------------|-----------|-----------|-----------|
| (a)项目信息维护 | (b)影响因素录入 | (c)关联事故录入 | (d)安全评价得分 |
| (e)项目指标预警分析 | (f)项目指标填报 | (g)项目指标审核 | (h)项目指标确认 |

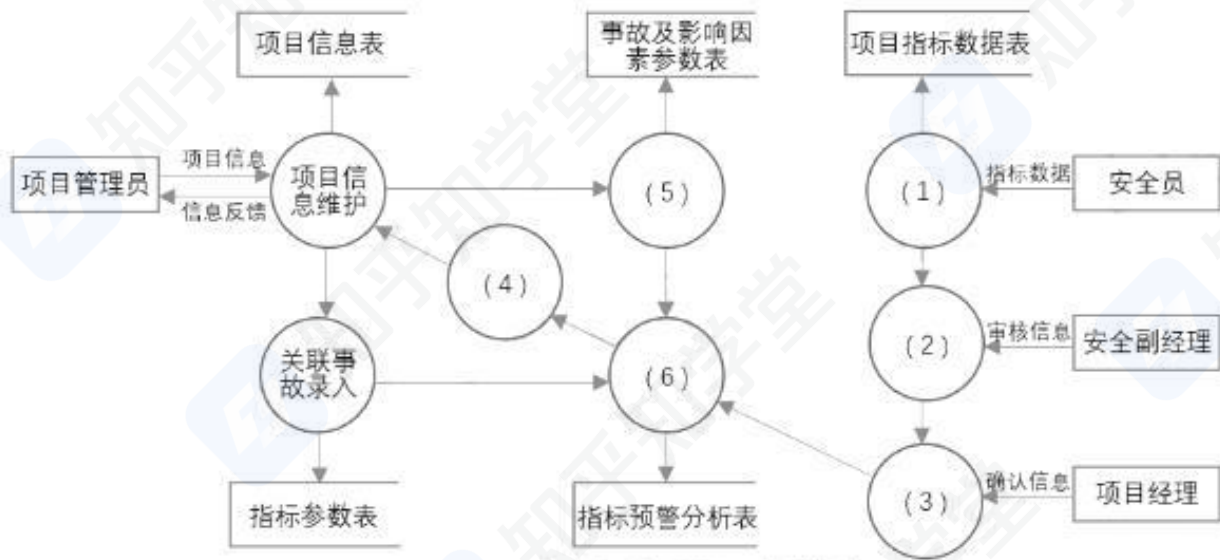


图 2-1 煤矿建设项目安全预警系统数据流图

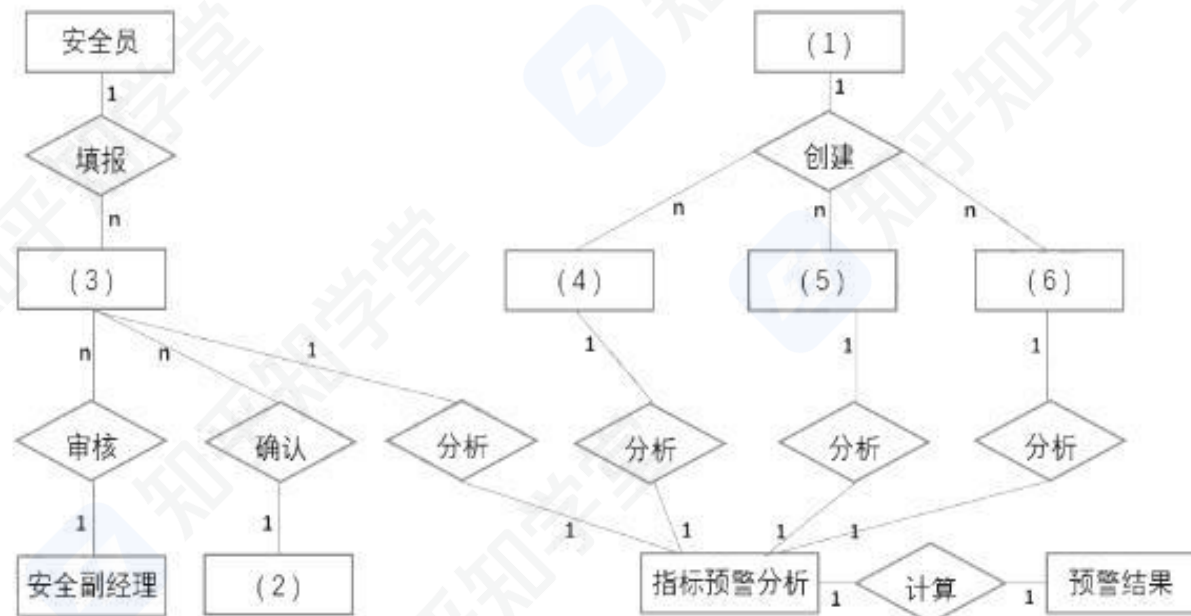


图 2-2 煤矿建设项目安全预警系统总体 E-R 图

【问题1】 (9分)

王工根据煤矿建设项目安全预警系统的功能要求，设计完成了系统的数据流图，如图2-1所示。请使用题干中描述的功能()~(h)，补充完善空(1)~(6)处的内容，并简要介绍数据流图在分层细化过程中遵循的数据平衡原则。

【问题2】 (9分)

请根据【问题1】中数据流图表示的相关信息，补充完善煤矿建设项目安全预警系统总体E-R图（见图2-2）中实体(1)~(6)的具体内容，将正确答案填在答题纸上。

【问题3】 (7分)

在结构化分析和设计过程中，数据流图和数据字典是常用的技术手段，请用200字以内的文字简要说明它们在软件需求分析和设计阶段的作用。

【问题1】

(1)f (2)g (3)h (4)d (5)b (6)e

分层细化的数据平衡原则：

1、子图与父图之间的平衡：

(1)父图与子图之间平衡是指任何一张DFD子图边界上的输入/输出数据流必须与其父图对应加工的输入/输出数据流保持一致。

(2)如果父图中某个加工的一条数据流对应于子图中的几条数据流，而子图中组成这些数据流的数据项全体正好等于父图中的这条数据流，那么它们仍然是平衡的。

2、子图内部：加工的输入和输出需要平衡。

【问题2】

(1)项目管理员 (2)项目经理 (3)项目指标数据
(4)~ (6)指标参数、项目信息、事故及影响因素参数

【问题3】

数据流图分析阶段：建立系统的功能模型，从而完成需求分析。

数据流图设计阶段：为模块划分与模块之间接口设计提供依据。

数据字典在分析与设计阶段的作用为：是所有人员工作的依据，统一的标准。它可以确保数据在系统中的完整性和一致性。

具体作用包括：按各种要求列表、相互参照、由描述内容检索名称、一致性检验和完整性检验。



2024年真题



知乎知学堂

- 1.序列图的哪三种消息和概念。
- 2.序列图补全填空。
- 3.系统分析设计过程中两种交互图的选取原则。
- 4.顺序图表示条件分支序列片段有哪些。

【问题1】

同步消息：同步消息是指发送方在发送消息后，必须等待接收方处理完成，才能继续执行。换句话说，发送方会暂停，直到收到接收方的响应

异步消息：异步消息是指发送方在发送消息后，不必等待接收方处理完成，可以继续执行其他操作。这意味着发送方与接收方可以并行执行

返回消息：返回消息表示接收方处理完同步消息后，返回结果给发送方。通常用于表达方法或函数的执行结果。

【问题2】略

【问题3】

在UML中，交互图主要用于描述在特定语境中对象之间的交互，它们可以在分析和设计阶段使用。交互图主要包括两种类型：序列图和通信图

序列图：强调消息的时间顺序，展示对象之间的动态合作关系，常用于分析阶段。

通信图：强调参与交互的对象以及它们如何相互关联，常用于设计阶段。

在分析阶段，你可能想要创建序列图来捕捉对象之间的动态合作，并且能够清晰地展示时序和并发。

在设计阶段，你可能想要创建通信图来定义交互模式，并且能够清晰地展示对象之间的静态关系和它们之间的关联。

【问题4】

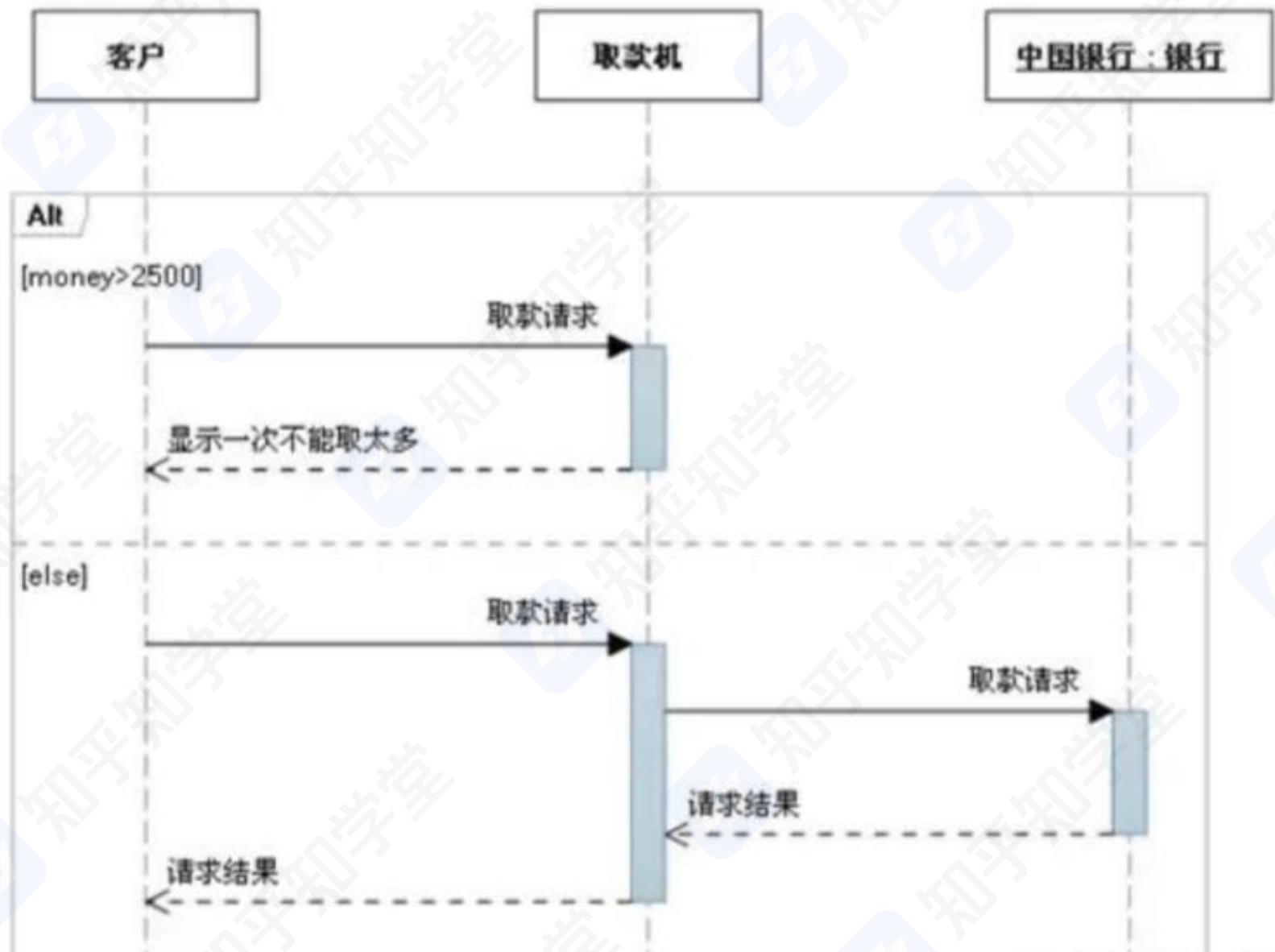
答：顺序图表示条件分支序列片段包括：

Alt 片段：表示条件分支，可以理解为“if-else”结构。

Opt 片段：只有在特定条件满足时才会执行对应的消息序列，类似于“if”结构。

Loop片段：用于表示循环逻辑，类似于“for”或“while”循环

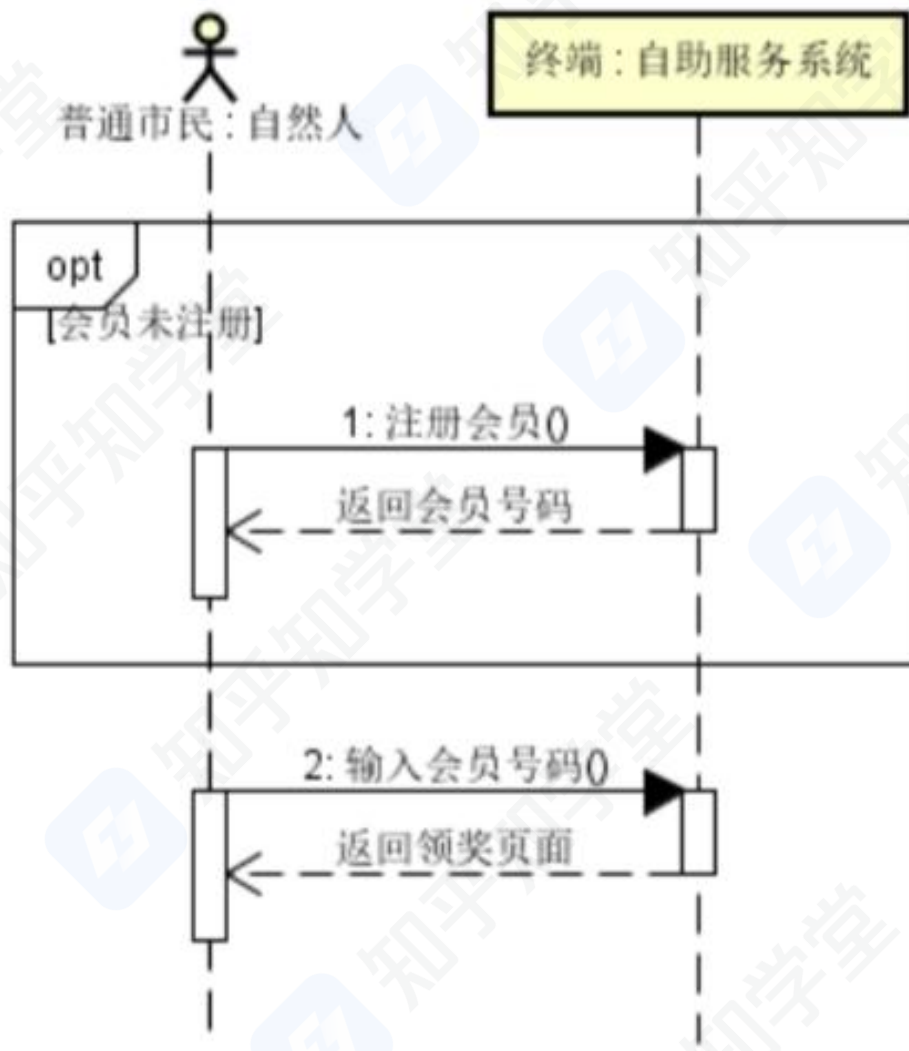
Break(中断)：表示在某个条件满足时，中断当前流程，并执行中断后的操作。这一片段用于处理异常或特殊情况。

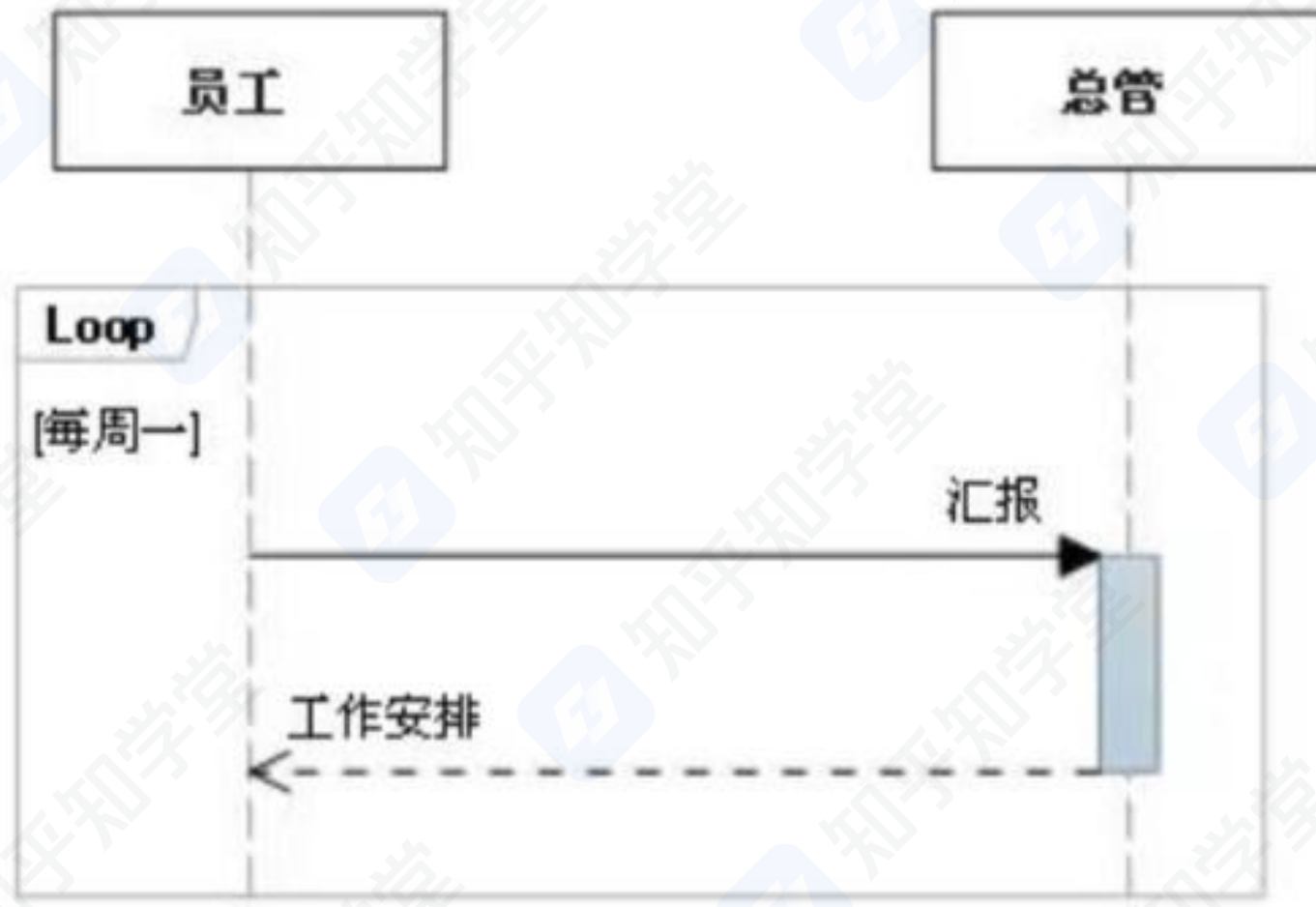


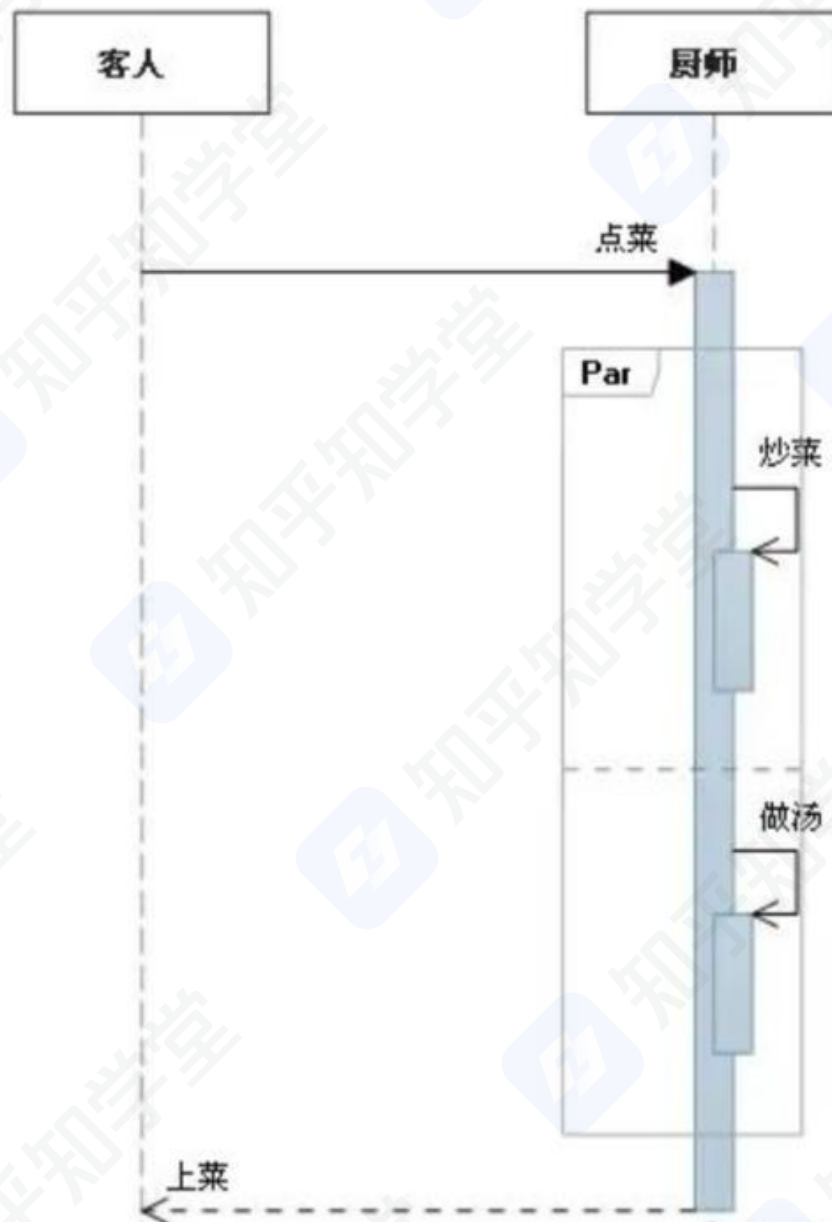
.....

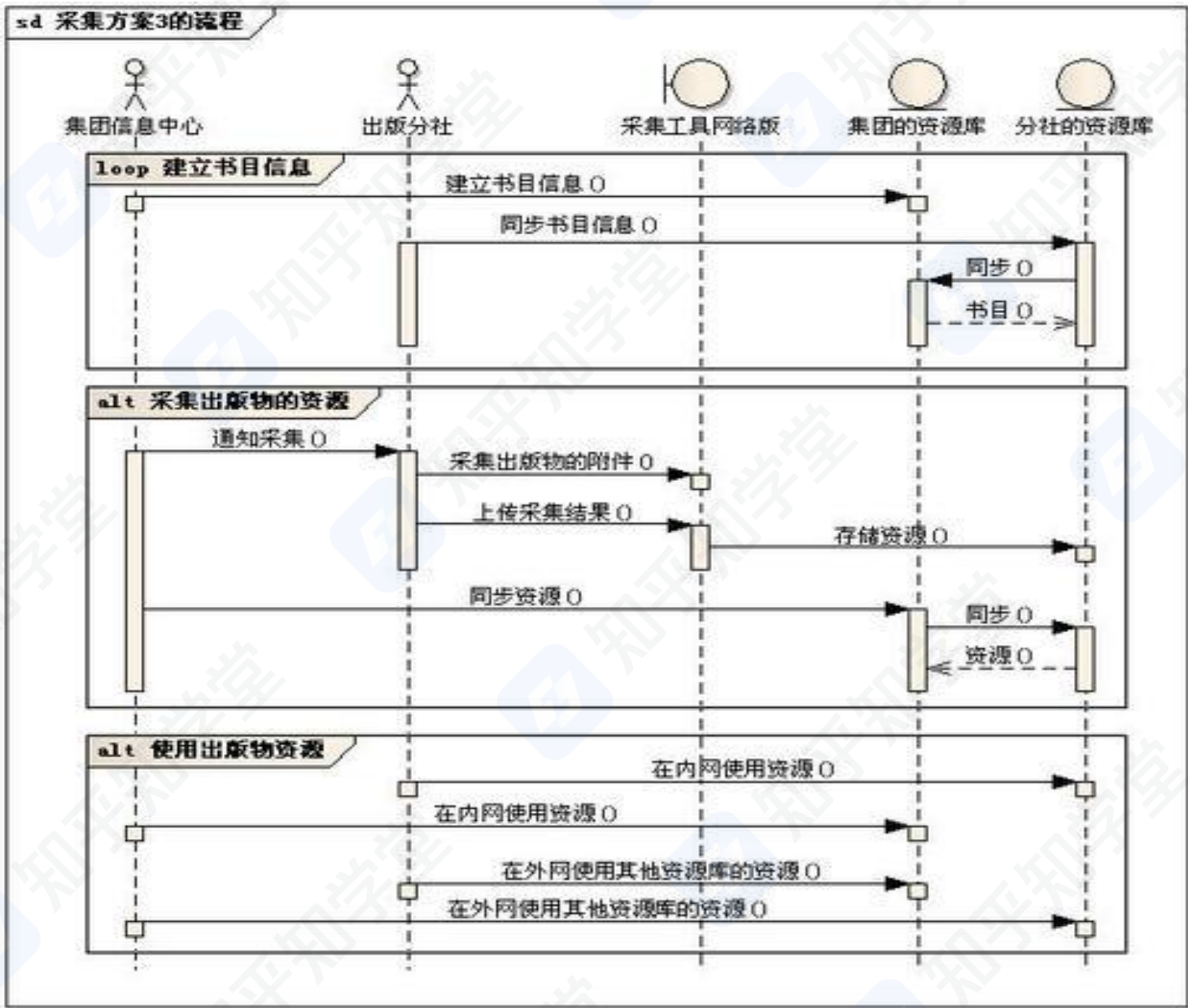
opt (Option)
表示当警戒值为真（符合
条件）的情况下进行执行
处理的意思。
opt需要填写参数。

图解：
市民来到终端领取免费奖
品。这时，如果市民是属
于未注册会员的时候，就
先要在终端进行会员注册
才能领取。
[会员未注册]是条件选
项。











3

数据库设计

根据近几年数据库设计的考题进行分析，我们可以得到如下几个特点：

- 1.一定会拿出一个数据库进行技术选型或者是某个数据库应用场景
- 2.必考nosql数据库的内容，其中被考到概率最高的就是redis
- 3.这几年的考试趋势是会有至少一用来问你一些操作的选择，若你没有使用过的话，基本是答不上来的，所以数据库的题目你想拿7-10分的分数是比较容易的，但是想拿高分，就很考验你的工作经验了
- 4.如果运气好，碰到了数据库的题目是带有比较表格的，那就可以选它了，这类题目基本可以拿上10分以上了

阅读以下关于数据库缓存的叙述，在答题纸上回答问题1至问题3。

【说明】

某大型电商平台建立了一个在线**B2B**商店系统，并在全国多地建设了货物仓储中心，通过提前备货的方式来提高货物的运送效率。但是在运营过程中，发现会出现很多跨仓储中心调货从而延误货物运送的情况。为此，该企业计划新建立一个全国仓储货物管理系统，在实现仓储中心常规管理功能之外，通过对在线**B2B**商店系统中订单信息进行及时的分析和挖掘，并通过大数据分析预测各地仓储中心中各类货物的配置数量，从而提高运送效率，降低成本。当用户通过在线**B2B**商店系统选购货物时，全国仓储货物管理系统会通过该用户所在地址、商品类别以及仓储中心的货物信息和地址，实时为用户订单反馈货物起运地（某仓储中心）并预测送达时间。反馈送达时间的响应时间应小于1秒。

为满足反馈送达时间功能的性能要求，设计团队建议在全国仓储货物管理系统中采用数据缓存集群的方式，将仓储中心基本信息、商品类别以及库存数量放置在内存的缓存中，而仓储中心的其它商品信息则存储在数据库系统。

【问题1】(9分)

设计团队在讨论缓存和数据库的数据一致性问题时，李工建议采取数据实时同步更新方案，而张工则建议采用数据异步准实时更新方案。

请用200字以内的文字，简要介绍两种方案的基本思路，说明全国仓储货物管理系统应该采用哪种方案，并说明采取该方案的原因。

【问题2】(9分)

随着业务的发展，仓储中心以及商品的数量日益增加，需要对集群部署多个缓存节点，提高缓存的处理能力。李工建议采用缓存分片方法，把缓存的数据拆分到多个节点分别存储，减轻单个缓存节点的访问压力，达到分流效果。

缓存分片方法常用的有哈希算法和一致性哈希算法，李工建议采用一致性哈希算法来进行分片。请用200字以内的文字简要说明两种算法的基本原理，并说明李工采用一致性哈希算法的原因。

【问题3】(7分)

全国仓储货物管理系统开发完成，在运营一段时间后，系统维护人员发现大量黑客故意发起非法的商品送达时间查询请求，造成了缓存击穿。张工建议尽快采用布隆过滤器方法解决。请用200字以内的文字解释布隆过滤器的工作原理和优缺点。

【问题1】

实时同步更新：在每次数据更新时，立即同步更新缓存和数据库，确保两者数据一致。该方案保证了缓存中的数据始终是最新的，适合对数据一致性要求较高的场景。

异步准实时更新：是在数据库数据更新后，先更新数据库，然后在稍后通过异步机制更新缓存。它不会立即更新缓存，但能通过批量或定时同步，保证数据在一定时间内趋于一致，适合对实时性要求不高的场景。

方案选择：

全国仓储货物管理系统应采用数据异步准实时更新方案。原因是对于订单反馈货物起运地和送达时间的功能，读操作远多于写操作，且对系统响应速度要求极高。异步准实时更新能减少实时同步的性能开销，保证系统的高效性和用户体验。

【问题2】

哈希算法：通过将数据的键值输入到哈希函数中，生成一个固定长度的哈希值，然后根据哈希值将数据映射到不同的缓存节点上。传统哈希算法存在缓存节点扩展或减少时需要重新分配大量数据的问题。

一致性哈希算法：一致性哈希算法将缓存节点和数据映射到一个虚拟的环形空间中，通过哈希值找到最近的缓存节点存储数据。当新增或减少缓存节点时，只有一小部分数据需要重新分配给新的节点，从而提高了系统的扩展性和稳定性。

选择原因：

李工建议采用一致性哈希算法，因为随着系统扩展，仓储中心和商品数量会不断增加或变化。一致性哈希算法在节点变化时只需重分配少量数据，能有效避免传统哈希算法的“缓存抖动”问题，提高系统的稳定性和扩展性。

【问题3】

布隆过滤器工作原理：布隆过滤器是一种空间效率高的概率型数据结构，它通过多个哈希函数将数据映射到一个位数组中。当需要判断一个数据是否存在时，只需检查该数据的多个哈希值对应的位是否都为1。如果有任意一个位为0，说明数据一定不存在；如果全部为1，数据可能存在，但可能有误判。

优点和缺点：

优点是布隆过滤器占用内存小、查询速度快，适合处理大规模数据。缺点是可能存在假阳性，即误判某个不存在的数据为存在，无法删除已添加的数据。

解决缓存击穿：

在全国仓储货物管理系统中，可以通过布隆过滤器提前过滤掉那些无效或非法的查询请求，从而减少缓存击穿对系统的影响。

科普缓存击穿：缓存击穿是指在缓存中存储的某些特定数据由于高并发访问，缓存中的数据失效（过期或未命中），导致大量请求直接涌向数据库，造成数据库瞬间承受巨大的压力，甚至引发系统崩溃。这种现象通常发生在热点数据上，因为热点数据会被频繁访问，一旦该数据在缓存中失效，就会引发大量请求同时直接查询数据库。



2023年和2024年真题简介



知乎知学堂

2023年和2024年因为没有出来具体的题目内容，所以我们只知道对应数据库的题型：

2023年：多数据源

2024年上：分布式锁

- 1.基于 MySQL 实现分布式锁的缺点。(9分)
- 2.举一个产生 Redis 分布式锁死锁的场景。(10分)
- 3.填写 Redis 命令，基于 ZSet。(6分)

2024年参考答案:

【问题1】

- 性能瓶颈: MySQL数据库本身可能成为性能瓶颈, 特别是在高并发情况下, 大量的锁请求和释放可能导致数据库性能下降。
- 单点故障: MySQL单点的特性使得其成为系统的单点故障, 如果数据库出现故障, 将导致整个系统的分布式锁失效。
- 锁粒度问题: MySQL的锁粒度可能过大或者过小, 过大的锁粒度会导致并发性能降低而过小的锁粒度可能会增加锁冲突的概率, 影响系统的并发性能。
- 数据一致性问题: 分布式系统中, 不同的数据库节点之间的数据同步可能存在延迟或者不一致的情况, 这可能导致分布式锁的有效性受到影响。
- 扩展性差: 随着系统规模的扩大, 单个MySQL数据库可能无法满足系统的性能和容量需求, 需要进行垂直或者水平扩展, 这会增加系统的复杂性和成本。
- 容错性差: MySQL数据库本身的容错性可能不如专门设计的分布式锁方案, 例如基于ZooKeeper 或者Redis的分布式锁方案, 因此在面对网络分区或者其他故障时可能无法提供可靠的锁服务。

2024年参考答案:

【问题2】

假设一个客户端A在获取Redis分布式锁后，正在执行任务，而执行任务的时间超过了锁的有效期，锁自动过期并释放，客户端B获取了该锁。此时客户端A尚未完成任务，继续执行，但认为自己仍然持有锁，这就产生了死锁问题，导致多个客户端对资源的并发操作混乱。

【问题3】

关于 nosql 平时都是关注的理论层面的，集群，主从，同步，分片等等答案：

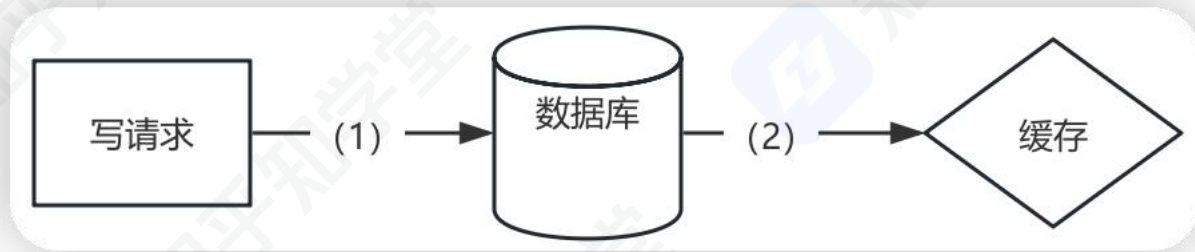
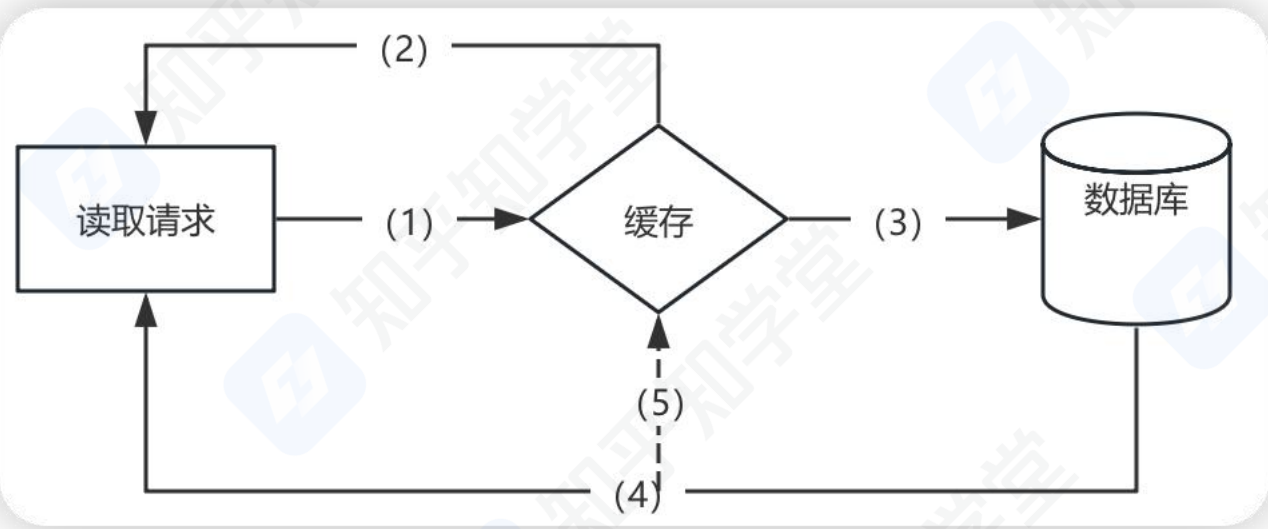
- 存入秒杀的分数命令：ZADD
- 获取分数范围的命令：ZRANGE
- 获取分数：ZSCORE

2024年下：数据库cache-aside架构

1.把该结构的读取数据的过程来填空 (10 分)

2.把该结构的写入数据的过程来填空 (6分)

3.王工使用了多线程技术进行缓存处理，线程1负责写入，线程2负责读取，可能存在数据不一致性问题，请解释其原因，并给出3个以上的解决办法 (9分)



【问题1】

- 1、查询缓存是否有目标数据
- 2、有缓存目标数据则命中并返回商品信息
- 3、若未命中则访问数据库查询该商品信息
- 4、将查询到的数据库数据更新到缓存
- 5、将查询到的数据库目标数据返回

【问题2】

- 1、更新数据库中的目标商品信息
- 2、将数据库中更新的商品信息写入到缓存中（确保数据一致性）

【问题3】

原因：读写线程并发导致缓存与数据库不一致（如写线程未更新完缓存时，读线程获取旧数据）。

解决办法：

- 加锁同步：对缓存操作加互斥锁，确保读写互斥。
- 设置缓存过期时间：强制定期刷新缓存，缩小不一致时间窗口。
- 延迟双删：写数据库后延迟再次删除缓存，防止旧数据残留。
- 异步队列：将缓存更新操作串行化到队列处理，避免并发冲突。

更详细的Redis相关知识请参考下面文档:



Redis的基础理论知识.pdf



Redis常见场景详解.pdf



4

案例答题技巧

出现频率第一的问题：请说明理由 / 阐述原因

1、请说明理由 / 阐述原因

使用场景：技术方案选型、架构风格选择、策略决策等所有需要论证的场景，几乎每题必出典型示例：

- 经过详细分析，最终选择了李工提出的架构方案。请说明选择该方案的理由。
- 为什么推荐采用微服务架构而非单体架构？请阐述原因。

答题要点：

- 结合案例背景的业务需求，如高并发、可扩展性要求
- 对比其他方案的优缺点，突出所选方案的优势
- 强调技术可行性与成本效益
- 引用架构设计原则，如高内聚低耦合、康威定律

答题模板：

- 满足业务需求：符合题干中业务场景的高并发、高可用、可扩展、易维护等核心需求。
- 技术优势显著：XX 方案在性能、可靠性、安全性、可扩展性等方面优于其他方案。
- 架构原则匹配：遵循高内聚低耦合、分层架构、关注点分离等软件架构设计原则。
- 落地可行性高：技术成熟、团队掌握度高、实施成本合理，便于后期运维与迭代。
- 规避潜在风险：可有效避免单点故障、性能瓶颈、数据不一致等系统风险。



出现频率第一的问题：请说明理由 / 阐述原因



1、请说明理由 / 阐述原因

题目1：某企业系统最终选用微服务架构，请说明选择该架构的理由。

标准答案

- 1) 满足业务快速迭代需求，各微服务可独立开发、部署、升级 → ①满足业务需求
- 2) 具备良好可扩展性，可针对高并发服务单独扩容 → ②技术优势显著（扩展性）
- 3) 故障隔离性好，单个服务故障不会导致整个系统瘫痪 → ⑤规避潜在风险（故障扩散）
- 4) 技术栈灵活，可根据业务场景选择合适技术实现 → ④落地可行性高（技术适配）

题目2：在缓存方案中选择 Redis 集群而非单机 Redis，请说明理由。

标准答案

- 1) Redis 集群可实现数据分片存储，支持海量数据存储 → ②技术优势显著（存储能力）
- 2) 具备主从复制与故障自动转移，提高系统高可用性 → ⑤规避潜在风险（单点故障）
- 3) 可横向扩展节点，提升系统并发读写能力 → ②技术优势显著（性能 / 并发）
- 4) 避免单机 Redis 存在的单点故障风险 → ⑤规避潜在风险（可用性保障）
- 5) 能够支撑秒杀、大促、节假日流量峰值等核心业务场景，满足业务高并发访问需求 → ①满足业务需求
- 6) 保障系统 7×24 小时不间断服务，支撑企业业务持续运营与规模扩张 → ①满足业务需求



出现频率第二的问题：请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异



2、请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异

使用场景：架构风格对比（微服务 vs 单体、Lambda vs Kappa）、技术方案评估、缓存策略选择等

典型示例：

- 对比单体架构和微服务架构的优缺点。
- 分析互斥锁方案与逻辑过期方案在缓存同步中的差异。

答题要点：

- 分点清晰列出**优缺点**
- 结合**业务场景**说明适用条件
- 重点**突出**性能、可用性、可扩展性、维护成本等关键维度
- 根据案例需求给出选型建议



出现频率第二的问题：请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异



2、请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异

答题模板：

➤ 优点

- **性能层面**：提升系统响应速度、吞吐量，降低资源消耗。
- **架构层面**：降低耦合度，提高内聚性，便于扩展与维护。
- **运维层面**：部署灵活、故障隔离、便于升级迭代。
- **业务层面**：适配业务快速变化，支撑业务规模扩张。

➤ 缺点

- **复杂度提升**：架构 / 实现逻辑复杂，开发与调试难度增加。
- **成本增加**：硬件资源、研发成本、运维成本有所上升。
- **一致性问题**：可能带来数据一致性、分布式事务等问题。
- **技术门槛高**：对团队技术能力要求较高，学习成本大。

➤ 对比差异万能句

- XX 方案侧重强一致性 / 高可用，适用于 XX 场景；
- YY 方案侧重高性能 / 低延迟，适用于 YY 场景；



出现频率第二的问题：请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异

2、请分析 XXX 的优缺点 / 对比 XXX 与 XXX 的差异

真题示例 1：请分析单体架构与微服务架构的主要差异。

标准答案：

- 1) 耦合性：单体架构耦合度高，微服务架构低耦合高内聚 → ②架构层面（耦合度）
- 2) 扩展性：单体只能整体扩容，微服务可按服务独立扩容 → ①性能层面 + ②架构层面（扩展性）
- 3) 维护性：单体维护成本高，微服务便于分工维护与迭代 → ③运维层面
- 4) 复杂度：微服务架构复杂度更高，存在分布式事务等问题 → 缺点①复杂度提升 + ③一致性问题

真题示例 2：对比缓存互斥锁方案与逻辑过期方案的优缺点。

标准答案：

- 1) 互斥锁：数据一致性强，实现简单，但会造成线程阻塞，并发性能差 → 互斥锁优点：一致性强（②架构层面）；缺点：性能差（①性能层面）
- 2) 逻辑过期：并发性能高、响应快，但会短暂返回旧数据，一致性较弱，需额外维护过期时间戳，实现复杂 → 逻辑过期优点：性能高（①性能层面）；缺点：一致性弱（③一致性问题）
- 3) 互斥锁方案侧重强一致性，适用于金融支付、订单交易等场景；逻辑过期方案侧重高性能、低延迟，适用于商品展示、首页访问等场景 → 对比差异万能句



出现频率第三的问题：请提出 XXX 的解决方案 / 优化措施



3、请提出 XXX 的解决方案 / 优化措施

使用场景：系统性能瓶颈、数据同步问题、网络故障影响、缓存一致性等问题解决类题目

典型示例：

- 网络故障对数据同步有什么影响？如何解决？
- 针对系统响应慢的问题，提出三种优化措施。

答题要点：

- 先分析问题**根源**，如网络分区、缓存穿透、数据库锁竞争
- 提出具体可落地的**技术方案**，如消息队列、冗余设计、异步处理
- 说明方案的实施步骤与**预期效果**
- 考虑**成本与风险**，如技术复杂度、运维成本

答题模板：

- **冗余与高可用**：采用集群、主备、多副本、多链路冗余设计，提高了系统高可用性，避免单点故障。
- **异步解耦**：引入消息队列实现异步通信，削峰填谷，保证消息可靠投递。
- **缓存优化**：构建多级缓存，减少数据库访问，提升读取性能。
- **数据可靠机制**：采用本地存储、断点续传、增量同步、幂等设计保证数据不丢失不重复。
- **容错与重试**：设计故障重试、降级、熔断机制，提升系统容错能力。
- **监警告警**：建立全链路监控与告警机制，快速定位与处理故障。



出现频率第三的问题：请提出 **XXX** 的解决方案 / 优化措施



3、请提出 **XXX** 的解决方案 / 优化措施

真题示例 1：网络故障对端云数据同步有何影响？给出解决措施。

标准答案：

- 1) 终端采用本地持久化存储，支持离线业务操作 → ④数据可靠机制（离线存储）
- 2) 采用断点续传与增量同步，网络恢复后仅同步增量数据 → ④数据可靠机制（高效同步）
- 3) 引入消息队列实现异步解耦，保证数据可靠投递 → ②异步解耦（削峰填谷）
- 4) 设计幂等性接口，避免数据重复提交与冲突 → ④数据可靠机制（冲突避免）

真题示例 2：针对数据库查询性能瓶颈，提出优化措施。

标准答案：

- 1) 建立合理索引，优化慢查询 SQL 语句 → ①性能层面（查询效率优化）
- 2) 引入分布式缓存，减轻数据库查询压力 → ③缓存优化（分担负载）
- 3) 采用读写分离架构，主库写、从库读，分担查询负载 → ①冗余与高可用（读写分流）
- 4) 对大表进行分库分表，降低单表数据量 → ①性能层面 + ②架构层面（水平扩展）



出现频率第四的问题：请简要回答 XXX 的定义 / 概念

4、请简要回答 XXX 的定义 / 概念

使用场景：领域驱动设计（限界上下文）、架构评估方法（ATAM）、质量属性概念等基础理论题
典型示例：

- 请简要回答限界上下文的定义与划分意义。
- 什么是质量属性场景？请说明其 6 个要素。

答题要点：

- 准确引用**官方定义**，避免个人理解偏差
- 用**简洁**语言解释，控制在 300 字以内
- **结合案例**说明实际应用价值
- 突出**核心要点**，如限界上下文的 "边界划分" 作用

答题模板：

- **核心定义**：用一句话概括 XX 概念的本质内涵。
- **核心目的**：阐述该概念用于解决什么问题、实现什么目标。
- **核心特征 / 要素**：列出 2-3 个关键特征或组成要素。
- **典型应用**：简要说明在架构设计中的应用场景。



出现频率第四的问题：请简要回答 **XXX** 的定义 / 概念

4、请简要回答 **XXX** 的定义 / 概念

真题示例 1：请简要回答限界上下文的定义。

标准答案：

- 1)、限界上下文是 **DDD** 核心概念，用于划分领域模型的边界 → ①核心定义
- 2)、每个上下文内维护统一的业务术语与领域模型 → ③核心特征
- 3)、上下文之间通过上下文映射进行交互 → ④典型应用（交互方式）
- 4)、避免模型混乱、消除业务歧义 → ②核心目的

真题示例 2：请简述 **CAP** 理论的含义。

标准答案：

- 1)、**CAP** 理论指分布式系统中三大特性：一致性、可用性、分区容错性 → ①核心定义
- 2)、三者无法同时满足，最多同时满足其中两个 → ③核心特征 / 结论
- 3)、用于指导分布式系统架构设计时的取舍决策 → ④典型应用
- 4)、核心目的是在分布式场景下权衡一致性与可用性 → ②核心目的



5

前沿技术



大模型定义：大模型是指具有大规模参数和复杂计算结构的机器学习模型。这些模型通常由深度神经网络构建而成，拥有数十亿甚至数千亿个参数。大模型的设计目的是为了提高模型的表达能力和预测性能，能够处理更加复杂的任务和数据。大模型在各种领域都有广泛的应用，包括自然语言处理、计算机视觉、语音识别和推荐系统等。大模型通过训练海量数据来学习复杂的模式和特征，具有更强大的泛化能力，可以对未见过的数据做出准确的预测。

大模型的特点：

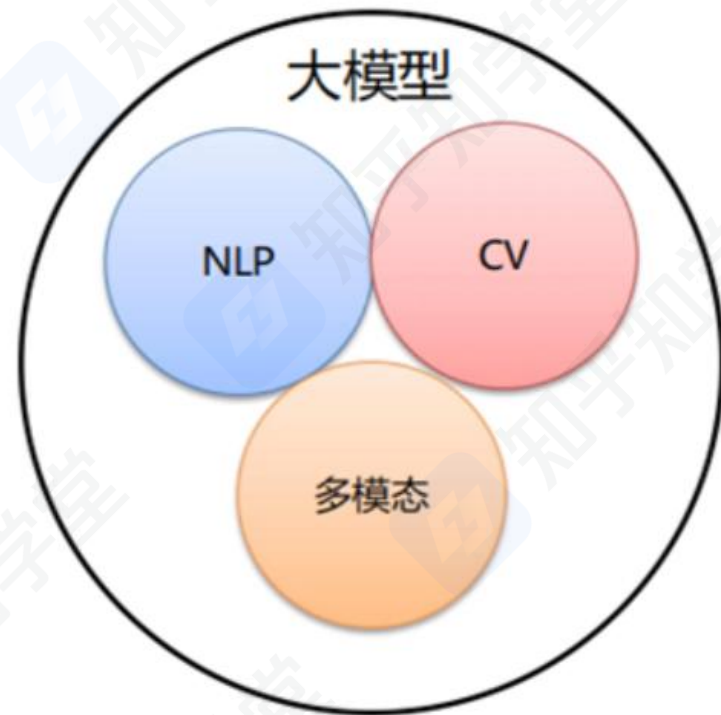
- 巨大的规模：大模型包含数十亿个参数，模型大小可以达到数百 GB 甚至更大。巨大的模型规模使大模型具有强大的表达能力和学习能力。
- 更好的性能和泛化能力：大模型通常具有更强大的学习能力和泛化能力，能够在各种任务上表现出色，包括自然语言处理、图像识别、语音识别等。
- 大数据训练：大模型需要海量的数据来训练，通常在 TB 以上甚至 PB 级别的数据集。只有大量的数据才能发挥大模型的参数规模优势。
- 强大的计算资源：训练大模型通常需要数百甚至上千个 GPU，以及大量的时间，通常在几周到几个月。
- 迁移学习和预训练：大模型可以通过在大规模数据上进行预训练，然后在特定任务上进行微调，从而提高模型在新任务上的性能。
- 自监督学习：大模型可以通过自监督学习在大规模未标记数据上进行训练，从而减少对标记数据的依赖，提高模型的效能。
- 领域知识融合：大模型可以从多个领域的数据中学习知识，并在不同领域中进行应用，促进跨领域的创新。
- 自动化和效率：大模型可以自动化许多复杂的任务，提高工作效率，如自动编程、自动翻译、自动摘要等。

按照输入数据类型的不同，大模型主要可以分为以下三大类：

语言大模型（NLP）：是指在自然语言处理（Natural Language Processing, NLP）领域中的一类大模型，通常用于处理文本数据和理解自然语言。这类大模型的主要特点是它们在大规模语料库上进行了训练，以学习自然语言的各种语法、语义和语境规则。

视觉大模型（CV）：是指在计算机视觉（Computer Vision, CV）领域中使用的大模型，通常用于图像处理和分析。这类模型通过在大规模图像数据上进行训练，可以实现各种视觉任务，如图像分类、目标检测、图像分割、姿态估计、人脸识别等。例如：VIT 系列（Google）、文心UFO、华为盘古 CV、INTERN（商汤）。

多模态大模型：是指能够处理多种不同类型数据的大模型，例如文本、图像、音频等多模态数据。这类模型结合了 NLP 和 CV 的能力，以实现对多模态信息的综合理解和分析，从而能够更全面地理解和处理复杂的数据。例如：DingoDB 多模向量数据库（九章云极 DataCanvas）、DALL-E(OpenAI)、悟空画画（华为）、midjourney。





1. 数据安全隐患：一方面大模型训练需要大量的数据支持，但很多数据涉及到机密以及个人隐私问题，如客户信息、交易数据等。需要保证在训练大模型的同时保障数据安全，防止数据泄露和滥用。OpenAI在发布ChatGPT模型的时候用了数月来保证数据安全以及符合人类正常价值观标准。
2. 成本高昂：大模型的训练和部署需要大量的计算资源和人力资源，成本非常高昂。对于一些中小型企业而言，难以承担这些成本，也难以获得足够的技术支持和资源。
3. 无法保障内容可信：大模型会编造词句，无法保障内容真实可信、有据可查。当前使用者只能根据自己需求去验证生成的内容是否真实可信，很难具有权威说服力。
4. 无法实现成本可控：直接训练和部署千亿级参数大模型成本过高，企业级应用应使用百亿级基础模型，根据不同需求训练不同的垂直模型，企业则只需要负担垂直训练成本。但是，如何实现高效的垂直训练，如何控制成本，仍是大模型面临的问题之一。

以上挑战依然有很大空间值得改进，需要进一步研究和探索新的技术和方法。比如可以采用数据加密、隐私保护等技术来保障数据安全；可以通过改进模型架构、优化训练算法、利用分布式计算等方式来提高大模型的效率和性能；此外，还可以通过开源和共享模型资源来降低成本、促进大模型的普及和应用等方式。

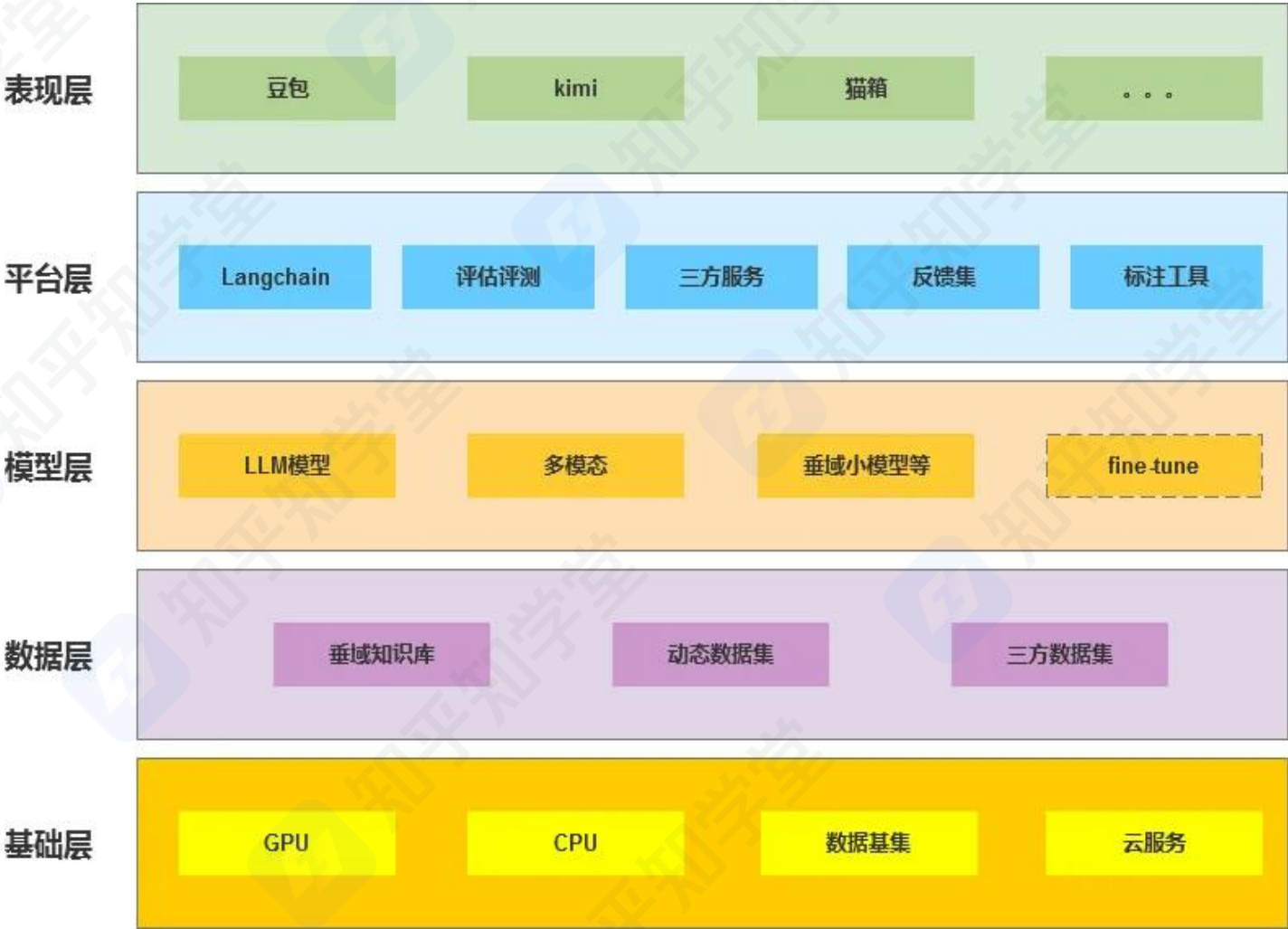
1. 基础层：为大模型提供硬件支撑，数据支持等
例如A100、数据服务器等等。

2. 数据层
这里的数据层指的不是用于基层模型训练的数据基集，而是企业根据自己的特性，维护的垂域数据。分为静态的知识库，和动态的三方数据集

3. 模型层：LLM或多模态模型
LLM(large-language-model)，也就是大语言模型，例如GPT，一般使用transformer算法来实现。
多模态模型即市面上的文生图、图生图等的模型，训练所用的数据与llm不同，用的是图文或声音等多模态的数据集

4. 平台层：模型与应用间的平台部分
比如大模型的评测体系，或者langchain平台等，提供模型与应用间的组成部分

5. 表现层：也就是应用层，用户实际看到的地方
这个就很好理解了，就不用我多作解释了吧



AI大模型-完整架构图

一、基础设施层

基础设施层是 AI 大模型平台运行的硬件根基。GPU（图形处理器）和 NPU（微处理器）凭借强大的并行计算能力，大幅加速深度学习模型的训练与推理过程，尤其适用于处理大规模数据和复杂算法。CPU 则负责协调和管理整个系统的运行。

二、云原生层

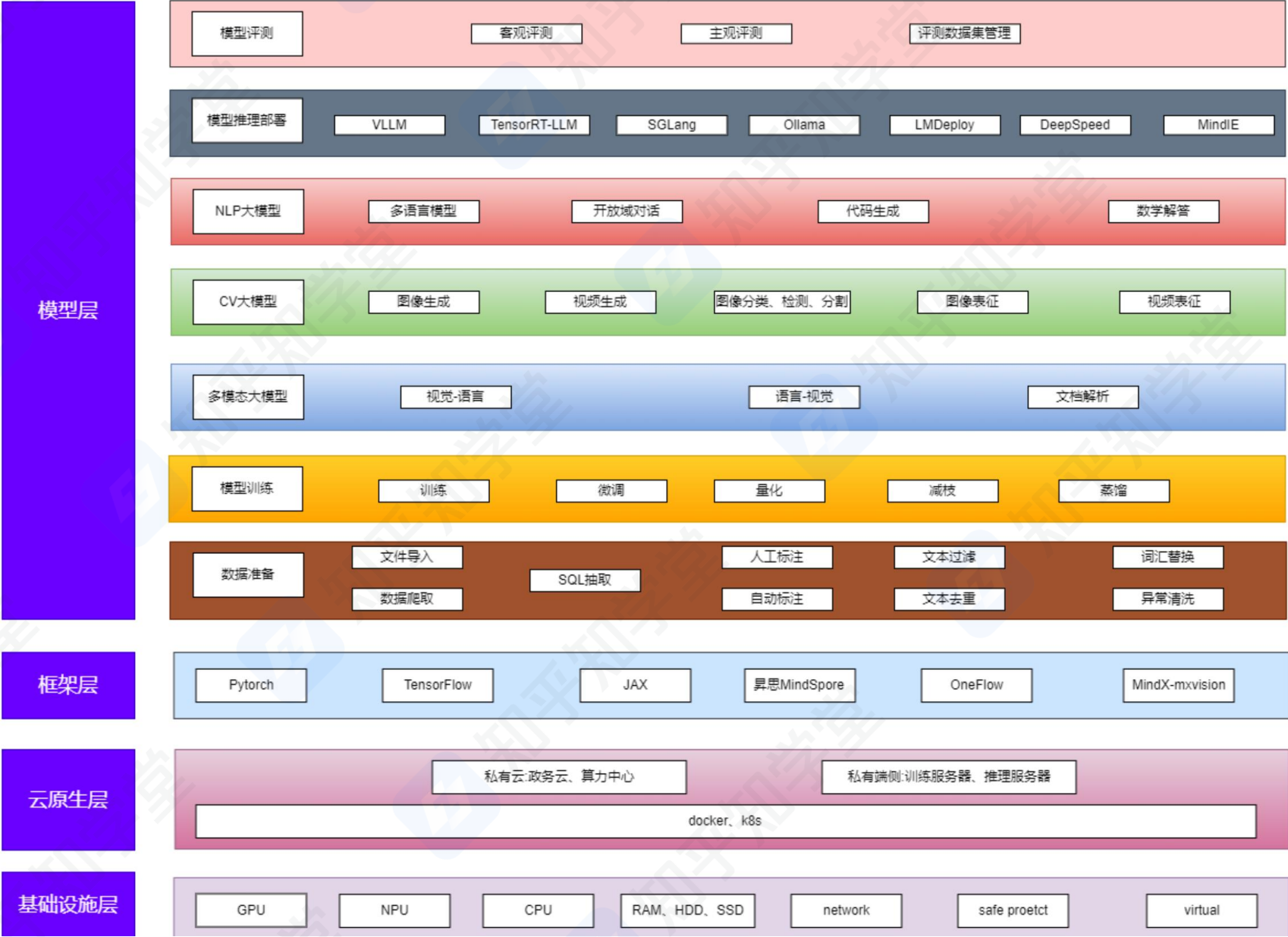
云原生层基于云计算技术构建，提供灵活且可扩展的资源管理与服务环境。私有云、政务云、算力中心等为平台提供了多样化的部署选择，满足不同用户对数据隐私和安全的需求。

三、框架层

框架层为 AI 模型的开发提供了基础工具和平台。PyTorch 以其动态计算图的特性，受到众多科研人员和开发者的青睐，便于快速实验和迭代模型。TensorFlow 则以其高度的灵活性和广泛的生态系统，适用于从研究到生产环境的各种场景。

四、模型层

4.1.数据准备：负责训练数据的采集、存储、清洗、标注等全流程管理，保障数据质量。



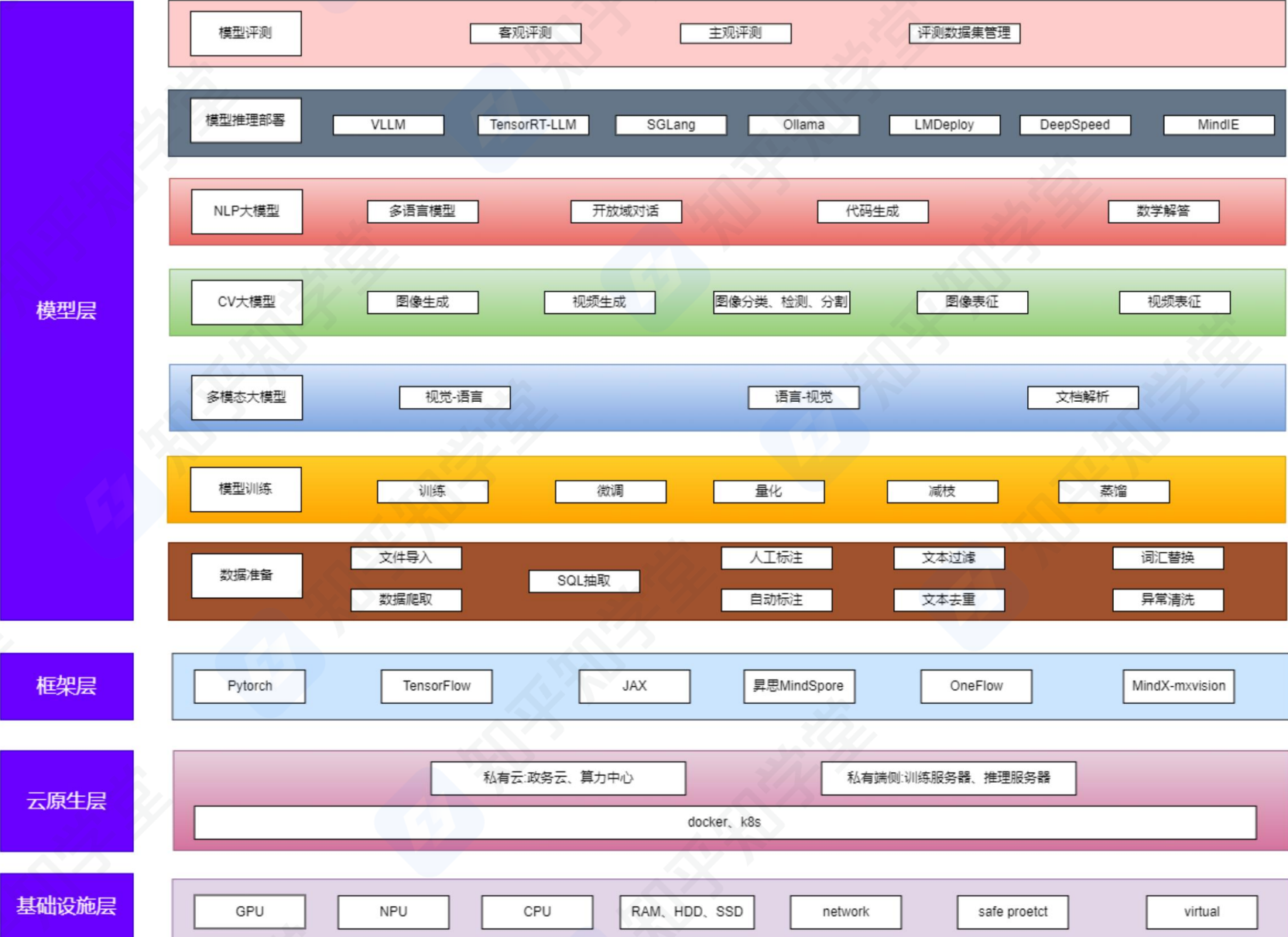
AI大模型-完整架构图

4.2. 模型训练：承担 AI 大模型从初始训练到优化迭代的关键环节。涵盖基础训练流程，支持对已有模型微调以适配特定任务；通过量化技术压缩模型规模，在保障性能的同时降低资源消耗；借助蒸馏方法提取复杂模型核心能力到轻量模型中；还可通过相关优化操作，为模型高效部署与应用提供支撑，是连接数据与可用模型的核心步骤。

4.3. 多模态大模型：打破单一模态限制，实现视觉与语言、语言与视觉等多模态信息的融合交互。支持视觉-语言任务，能基于视觉内容生成文本描述，或根据文本指令生成视觉内容；完成语言-视觉任务，实现文本到视觉的跨模态理解与生成；具备文档解析能力，对各类格式文档进行结构分析、内容提取与理解，为跨模态信息处理与应用提供强大支持。

4.4. CV 大模型（计算机视觉大模型）：聚焦图像、视频等视觉领域的任务能力构建。可实现图像生成，创造逼真或创意性图像内容；支持视频生成，完成动态视频创作与合成；在图像分类、检测、分割任务上表现出色，精准识别物体类别、定位位置并划分区域；具备图像表征能力，将图像转化为便于计算分析的特征向量；还能进行视频表征，提取视频关键特征，为视频理解、分析等上层应用提供基础。

4.5. NLP 大模型（自然语言处理大模型）：致力于处理人类语言相关任务。支持多语言模型，实现不同语种文本的理解与生成；能进行开放域对话，与用户开展无固定主题的自然语言交互；具备代码生成能力，自动生成符合语法规则的代码片段；还可作为数字解答助手，针对各类问题提供精准、结构化答案，广泛应用于智能客服、知识问答等场景。

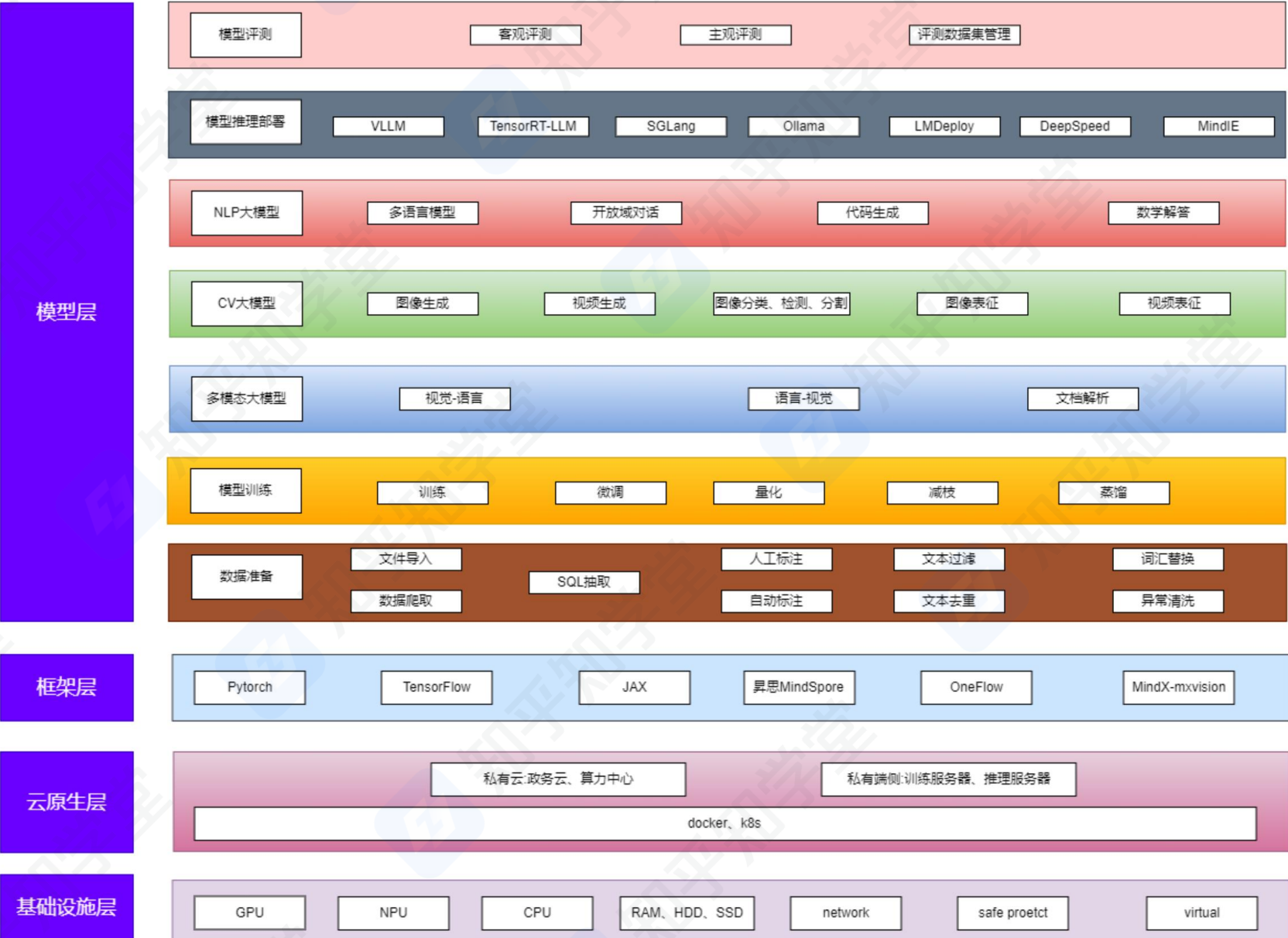


AI大模型-完整架构图

4.6.模型推理部署层：模型推理部署是将训练好的 AI 大模型转化为实际服务的关键环节。VLLM、TensorRT-LLM、SGLang、Ollama、LMDeploy、DeepSpeed、MindIE 等工具各有优势：或优化大语言模型（LLM）的推理效率，或适配特定硬件提升计算性能，或简化部署流程降低应用门槛，最终实现推理过程的低延迟、高吞吐，支持大模型在各类硬件环境下快速、稳定地对外提供推理服务（如文本生成、图像理解等任务的实时响应）。

4.6.评测与数据集管理：该部分是检验 AI 大模型性能、保障模型迭代优化的重要支撑。

- 模型评测：是检验 AI 大模型性能与效果的核心手段，通过构建全面评估体系，判断模型是否达到预期任务目标（如图像生成质量、对话交互自然度等）。
- 客观评测：基于量化指标与可重复实验流程，对模型在特定任务（如图像分类准确率、文本生成困惑度、代码生成语法正确率等）上的性能精准测量，提供客观、量化的评估结果，为模型优化方向提供数据支撑。
- 主观评测：从人类主观感受角度出发，邀请专业人员或普通用户对模型生成结果（如对话流畅性、图像艺术感、解答内容实用性等）进行体验与评价，弥补客观评测在“人性化感知”“创意性”等维度的不足，让模型评估更全面。
- 评测数据集管理：负责评测所需数据集的采集、整理、标注与维护，确保数据集的质量、多样性与代表性，为客观评测和主观评测提供可靠数据源，保障评测结果的准确性与说服力，是模型评测体系中不可或缺的基础支撑环节。



AI大模型-完整架构图

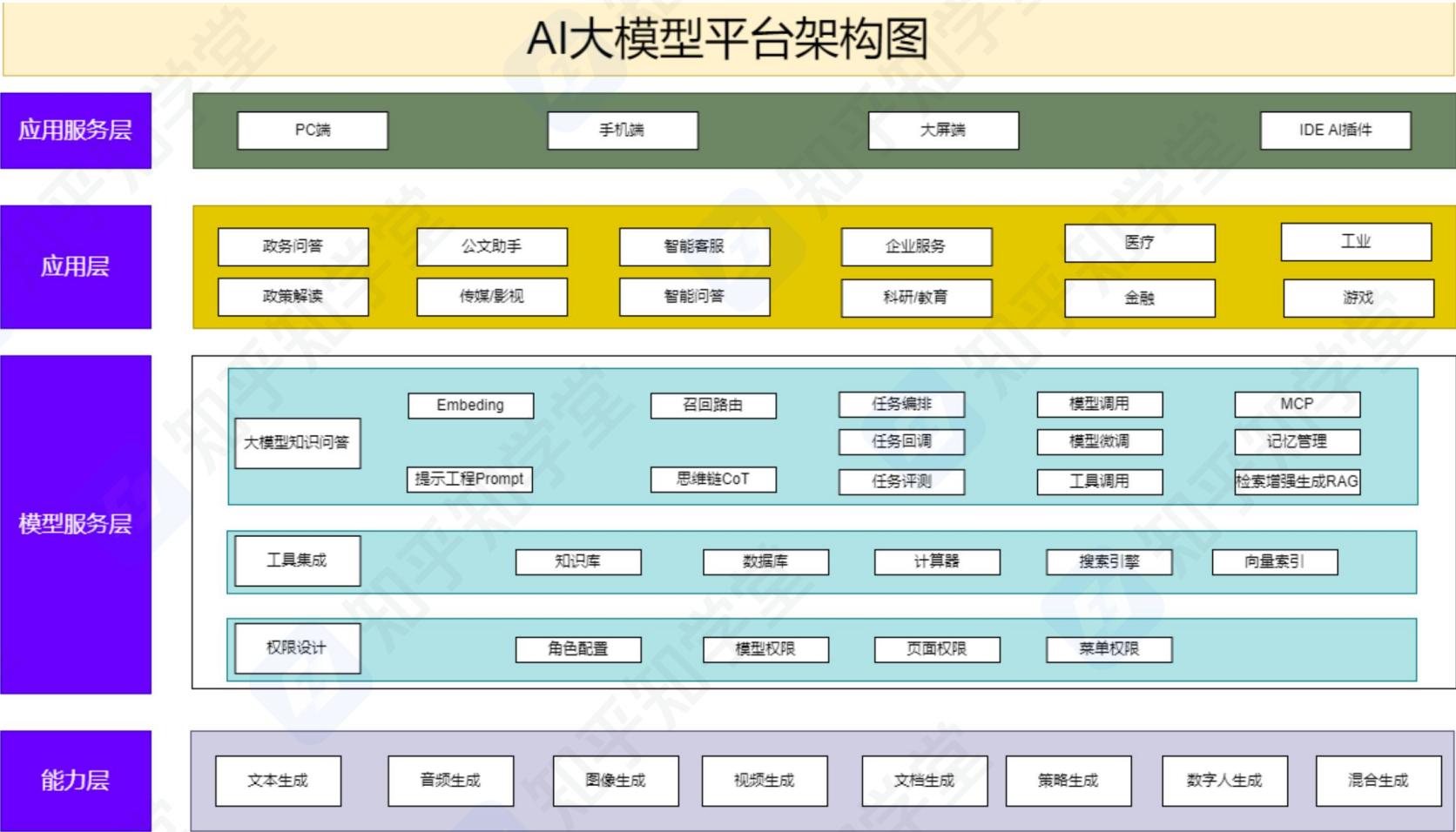
五、能力层

作为 AI 大模型平台的核心能力底座，提供多模态的生成类基础能力，为上层应用场景提供底层支撑。具备文本生成（自动创作文章、对话等文本）、音频生成（合成语音、音乐等音频素材）、图像生成（生成创意或写实图像）、视频生成（创作动态视频内容）、文档生成（自动化生成报告、公文等格式文档）、策略生成（为决策场景提供智能方案，如运营、风控策略）、数字人生成（打造可交互的数字人形象与行为）、混合生成（融合文本、音视频等多模态内容生成）等能力，覆盖多维度创作与生成需求。

六、模型服务层

是连接“底层能力”与“上层应用”的核心枢纽，负责大模型能力的封装、调度、增强，同时保障服务安全与高效。

- 上层（大模型交互与任务处理）：围绕“大模型知识问答”，用 Embedding、Prompt、CoT 等技术，通过召回路由、任务编排等流程，结合模型调用/微调、工具调用及 MCP、记忆管理、RAG，提升问答智能化。
- 中层（工具与数据支撑）：工具集成外部工具（计算器、搜索引擎等），知识库、数据库等为大模型提供精准信息来源。
- 下层（权限与安全控制）：通过权限设计，结合角色、模型、页面、菜单权限，保障平台安全规范。



Retrieval-Augmented Generation (RAG) 是一种结合了检索和生成技术的自然语言处理 (NLP) 模型。RAG模型通过引入外部知识库，利用检索模块 (Retriever) 从大量文档中提取相关信息，并将这些信息传递给生成模块 (Generator)，从而生成更加准确和有用的回答或文本。

RAG模型的核心思想是通过检索和生成的有机结合，弥补生成模型（如GPT-4、DeepSeek等）在处理知识密集型任务时的不足。传统的生成模型在面对复杂问题时，常常因缺乏足够的知识而生成出错误或无关的回答。而RAG通过检索模块获取相关的背景信息，能够在生成过程中参考这些信息，生成更具可信度和准确性的答案。

RAG的工作原理：RAG模型的工作流程可以分为两个主要阶段：**检索阶段和生成阶段**。

- 检索阶段：在该阶段，RAG模型首先接收输入的查询（如用户提问），然后通过检索模块从预定义的知识库中查找与查询相关的文档或段落。这一步骤通常涉及到高效的向量搜索技术，如基于BERT的向量化方法，将文档和查询映射到相同的向量空间中，从而能够快速计算出文档与查询的相似度。检索模块会返回若干个与查询最相关的文档，这些文档将作为生成阶段的输入。
- 生成阶段：在该阶段，RAG模型将检索到的文档和原始查询一起输入到生成模块中。生成模块通常是一个强大的生成模型，它将利用检索到的文档信息生成最终的回答或文本。在这一过程中，生成模块能够参考检索到的背景信息，从而生成内容更加丰富、信息更加准确的回答。

RAG架构图

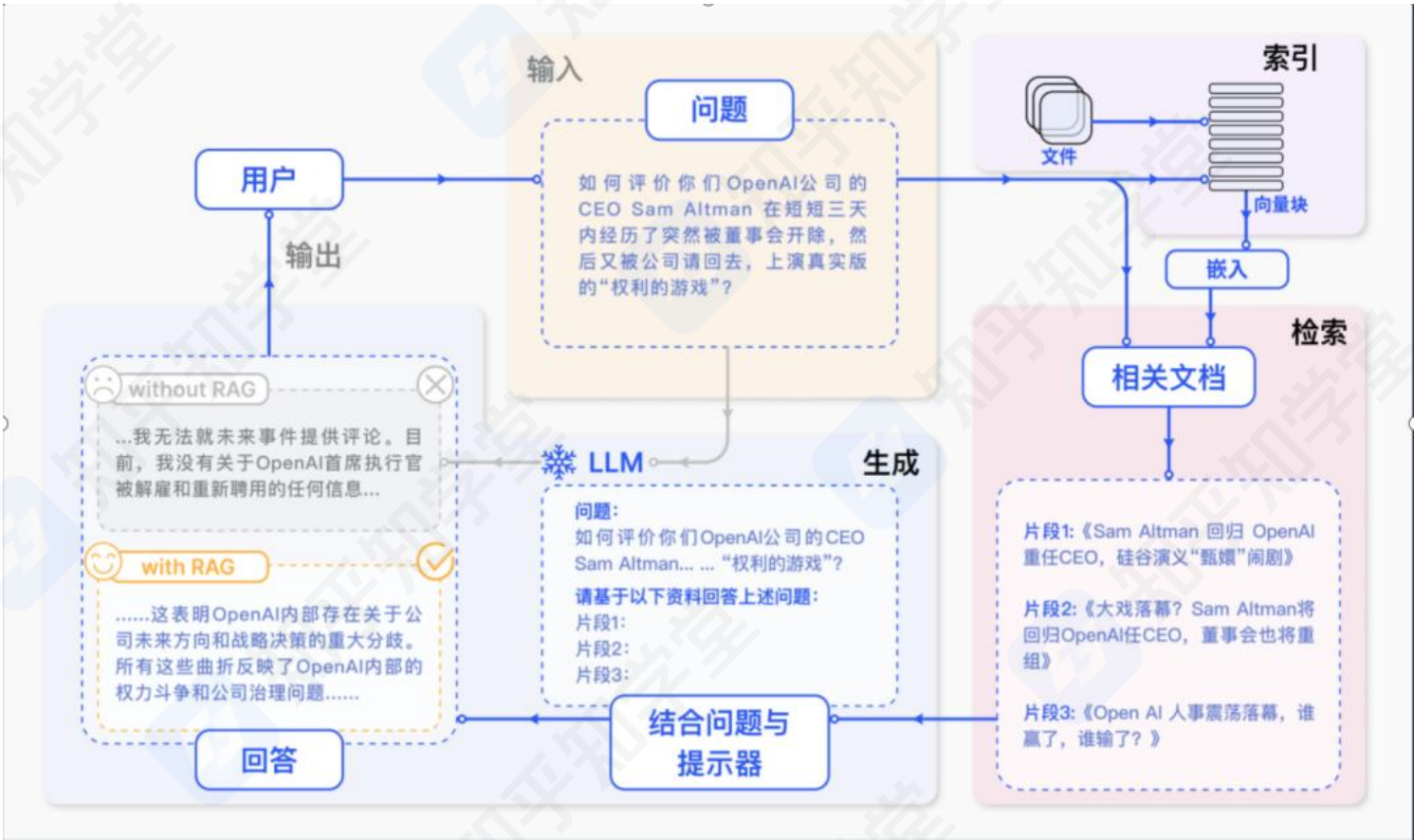
RAG (Retrieval-Augmented Generation) 模型的技术架构包括两个主要部分：检索模块 (Retriever) 和生成模块 (Generator)。这两个模块通过一个统一的框架无缝集成，以实现高效的检索和生成任务。

检索模块 (Retriever) :

- 负责从大规模的知识库或文档集合中检索与输入查询相关的信息。
- 使用预训练的双塔模型 (dual-encoder) 进行高效的向量化检索。
- 输出若干个与查询相关的文档或段落，作为生成模块的输入。

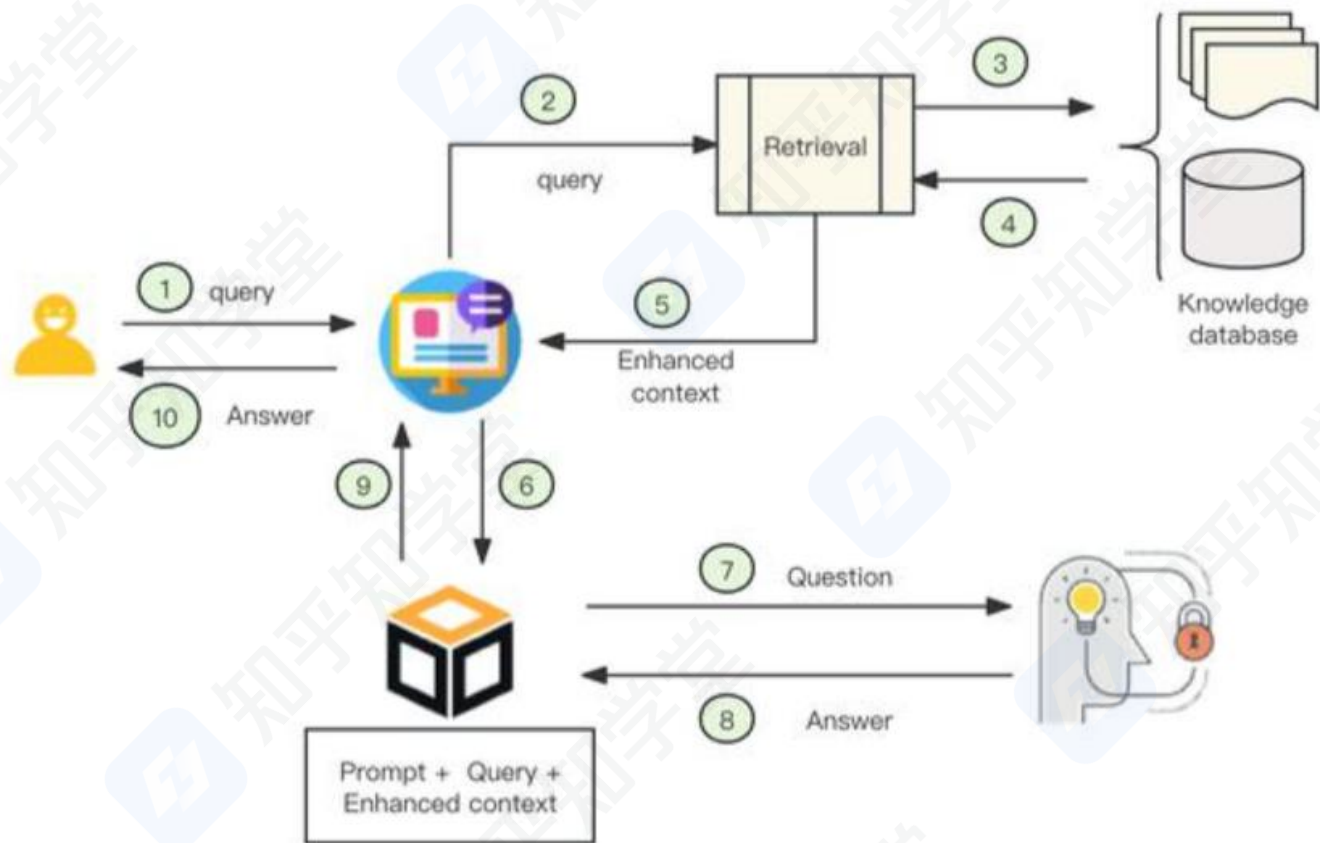
生成模块 (Generator) :

- 负责根据检索到的文档和输入查询生成最终的回答或文本。
- 使用强大的生成模型 (如T5或BART) 对输入进行处理。
- 输出连贯、准确且信息丰富的生成内容。



RAG架构图

1. 用户发起查询：用户向系统提交问题（query）。
2. 转发查询到检索模块：系统将query传递给“Retrieval（检索）”模块。
3. 从知识库检索相关内容：检索模块针对query，从“Knowledge database（知识数据库）”中查找相关文档 / 片段。
4. 知识库返回检索结果：知识数据库把相关内容回传给检索模块。
5. 生成增强上下文：检索模块整合结果，生成与query高度相关的“Enhanced context（增强上下文）”。
6. 构建 Prompt 组合：系统将「Prompt（提示词） + 用户 Query + 增强上下文」组合，形成更丰富的输入。
7. 传递给大语言模型（LLM）：把组合后的“Question”发送给 LLM（右侧图标，代表具备推理能力的大模型）。
8. LLM 生成回答：LLM 基于输入的 Prompt、Query 和增强上下文，生成针对性回答（Answer）。
9. 回答回传交互层：LLM 的 Answer 回传给系统交互模块（中间蓝色图标）。
10. 返回最终回答给用户：交互模块将 Answer 反馈给用户。





THE END

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

