1. 用例图有哪些基本元素

（1）参与者

参与者是指存在于系统外部，并于系统交互的任何实体，可以是使用系统的用户，也可以是硬件和系统实体

（2）用例

用例表示系统提供的服务，它定义参与者如何使用系统，它描述的是参与者为了使用系统的功能与系统之间的一组对话

（3）通信关联

通信关联表示的是用例与用例之间，或者用例与用户之间的关系。

箭头指向的一方表示用例的接收者，箭头尾部表示用例的发起者。

如果系统不想表示主动和被动的关系，可以不使用箭头

构建用例模型一般要经历几个阶段:

构建用例模型一般要经历四个阶段:识别用例参与者，合并需求获得用例，细化用例描述，调整用例模型，其中前三个阶段是必须的。

1. 数据库索引优化

（1）以物化视图代替多表关联，或者尽可能减少多表关联

（2）以不相关子查询代替相关子查询

（3）去掉查询不相关的字段

（4）把in查询转化为or查询

（5）经常commit的查询，尽早释放锁

（6）避免嵌套游标和多重循环

（7）经常查询的几个字段建立联合索引

1. web应用

微服务的优缺点：

优点：

（1）每个微服务都很小，都能聚焦一个业务功能和业务需求

（2）微服务能够被小团队开发，通常2到5个人

（3）微服务是松耦合的，无论是开发阶段，还是部署阶段，都是独立的

（4）微服务可以使用不同的语言开发

（5）每个微服务都有自己单独的存储 可以有独立的数据库

缺点：

（1）微服务很难实现在不采用分布式事务的情况下实现跨服务的功能

（2）微服务测试困难

（3）跨服务实现要求功能要求团队之间紧密协作

（4）微服务部署复杂

微服务网关的主要作用：7个

（1）提供统一的入口

（2）可以进行权限认证等安全管理

（3）可以根据流量进行限流

（4）数据缓存

（5）性能监控

（6）异常监控

（7）服务降级

1. 应用服务器的概念

应用服务器概念

应用服务器是指通过各种协议把商业逻辑暴露给客户端的程序。

它提供了访问商业逻辑的途径以供客户端使用。

应用服务器为实现web服务器的功能和资源访问机制，提供了简单、可管理的方式。

它是一个开发、部署、运行、管理、维护的平台 ，它可以提供集群的功能，让多个异构的服务器协调工作、相互备份，满足企业级应用需要的可用性，高性能，可靠性和伸缩性

（1）当系统处理能力不够时，可以通过增加硬件的方式，提高系统的可扩展性

（2）动态调整不同主机之间的负载，最大限度地使用系统资源，提供单机稳定性

（3）动态调整工作主机，没有单点故障

1. 各种需求定义

操作性需求：

指系统完成任务所需的操作环境要求及如何满足系统将来可能的需求变更的要求

性能需求:

指响应时间、吞吐量、准确性、有效性、资源利用率等与系统效率有关的指标。

安全性需求：

指为了防止系统崩溃和保证数据安全所需采取的保护措施的要求，为系统提供合理的预防措施

文化需求：

指使用本系统的不同用户群体对系统提出的特有要求

胖瘦客户端对比：

瘦客户端可以在服务器端通过集群实现。

瘦客户端将业务迁移到应用服务器，所以有故障只需要修改服务器上的内容，而胖客户端要修改所有的客户端，工作量大，相比之下，瘦客户端更有优势

胖客户端后端是数据库，要做加密传输没有基础，而瘦客户端可以

1. 实体与类的区别

区别:实体用于数据建模，而类用于面向对象建模，实体有属性，而类有属性和行为

抽象用例和基本用例的区别：

essential use case：抽象用例，是从基本用例抽取的公共用例部分，为了避免重复工作，优化结构提出的用例

real use case：基本用例，是实实在在和用户需求有对应关系的用例，是从用户获取需求的渠道获得的

1. Redis总结

数据类型：

存储方案：

主从模式，哨兵模式，集群模式

分片方案：

客户端分片，即在客户端根据key的hash值对应到不同的服务器

中间件实现分片，在应用软件中间，采用codis等，由中间件实现服务到后台redis节点的路由分派

客户端服务端协作分片。redis cluster模式，客户端可采用一致性hash，服务端提供错误节点的重定向服务

Redis的RDB和AOF对比



1. 架构总结

风险点:

架构设计中潜在的、存在问题的架构决策所带来的问题

敏感点：

为了实现某种特定的质量属性，一个或者多个组件所具有的特性

权衡点:

影响多个质量属性的特性，对多个质量属性的敏感点

1. 状态图和活动图的区别

状态图：动态行为，状态序列，事件，动作

主要描述一个对象在其生存期间的动态行为，表现对象所经历的状态序列，引起状态转移的事件，以及伴随状态改变产生的动作。

活动图：工作流程和并发行为，特殊形式，下一个活动

是描述系统的工作流程和并发行为。它可以看做状态图的特殊形式，一个活动结束，立即进入下一个活动。

区别:结果，活动；并发，一个不可以并发

状态图侧重描述行为的结果，活动图侧重于描述行为的动作；活动图可以描述活动的并发行为，而状态图不可以

1. 什么是持久层？好处？

持久层：根据分层思想，通过建立逻辑数据操作接口，采取一定的对象/关系映射策略，隐藏数据库访问代码策略，向业务开发人员提供透明的对象持久化机制。

好处：

分离业务逻辑和数据层，降低两者之间的耦合；

通过对象关系映射向业务逻辑提供面向对象的数据访问；

简化数据访问层，隐藏数据库连接、数据的读写操作和事务管理细节

1. 什么是mvc架构风格？

MVC 架构风格最初是 Smaltalk-80中用来构建用户界面时采用的架构设计风格。其中 M 代表模型( Model ), V 代表视图( View ), C 代表控制器( Contr oler )。在该风格中,模型表示待展示的对象,视图表示模型的展示,控制器负责把用户的动作转成针对模型的操作。模型通过更新视图的数据来反映自身的变化。

风格对比：



1. 逻辑结构设计阶段的主要任务？

逻辑结构设计阶段的主要任务是确定数据模型、将ER图转化为指定的数据模型、确定完整性约束、确定用户视图

什么是超类实体？

超类实体是由多个实体中所共有的属性组成。

什么是派生属性？

派生属性是可以由其他属性计算所得，派生属性由于存储计算结果值。派生属性：资费、总计。

1. 统一数据访问机制原因？

标准的数据访问机制可以屏蔽不同通信协议的差异，为应用程序提供一个统一的接口，从而实现不同协议之间的数据交换