1. HashTable 和 HashMap 的相同和差异，是否都线程安全?

HashMap 和 HashTable 都是基于哈希表实现的 Map 接口的实现类，但是他们采用的哈希算法和数据结构不同：

HashMap 采用数据+链表+红黑树的数据结构，当哈希冲突发生时，会使用链表或者红黑树来解决冲突，并使用负载因子来进行扩容操作；可以存储 null 值和 null 键；

HashTable 采用数据+链表，当哈希冲突发生时，使用常量来决定扩容；不允许存储 null 值和 null 键；

HashaTable 是线程安全；HashMap 不是线程安全的类，其对应的线程安全类是 ConcurrentHashMap;

参考：
https://blog.csdn.net/weixin_58724261/article/details/131096333

2. AOP（面向切面编程 Aspect Oriented Programming）和 IoC（控制反转 Invension of Control）

IoC 控制反转是指将子对象的创建和管理交由外部来实现，由第三方容器来管理相关资源；严格来说 IoC 是一种设计模式，中心思想是“将原本在程序中手动创建对象的控制权，交由 Spring 框架来管理（IoC 在其他语言中也有应用，并非 Spring 特有）”，IoC 容器是 Spring 用来实现 IoC 的载体，IoC 容器实际上就是个 Map(key, value), Map 中存放的是各种对象。IoC 最常见以及最合理的实现方式叫做依赖注入（DI， Dependency Injection）

AOP 是面向对象编程（OOP）的一种延续；OOP 时通常用多态方式实现同一基类，相同的方法在子类中采用不同实现，可以达到不同的表现，但是父类中控制逻辑的代码就无法拆分，AOP 的作用就是将控制逻辑和业务逻辑分开，采用切点方式进行子类定制。AOP 主要用来解决，在不改变原有业务逻辑的情况下，增强横切逻辑代码，根本上解耦合，避免横切逻辑代码重复。

参考：
https://blog.csdn.net/m0_46657043/article/details/106407887

3. springboot 和 springcloud 的区别

SpringBoot 是一个款苏开发的轻量级框架，帮助快速整合第三方常用框架，完全采用注解化，简化 XML 配置，内置 HTTP 服务器。作用是简化 Spring 应用的初始搭建及开发，解决各种 JAR 包冲突问题。

SpringCloud 是一系列框架的有序集合，是一个分布式服务治理的框架，本身不会提供具体功能性的操作，是一个为开发者快速构建分布式系统的工具，简化了分布式系统基础设施的开发，包括服务发现，服务注册，配置中心，消息总线，负载均衡，断路器和数据监控等。SpringCloud 不是重复制造轮子，而是将目前各家公司开发的比较成熟的服务框架组合起来，通过 SpringBoot 风格进行再封装，屏蔽掉复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包，即默认大于配置。

SpringBoot+SpringCloud 实现微服务开发，具体就是，SpringCloud 具备微服务开发的核心技术：RPC 远程调用技术；SpringBoot 的 web 组件默认继承了 SpringMVC，可以实现 HTTP+JSON（RESTFUL）的轻量级传输，编写微服务接口，所以 SpringCloud 是依赖 SpringBoot 框架实现微服务开发。 SpringBoot 专注于快速开发单个微服务，

SpringCloud 是将 SpringBoot 开发的一个个单体微服务组合并管理起来，它是关注于全局的服务管理框架；

SpringBoot 可以离开 SpringCloud 独立使用开发项目，但是 SpringCloud 离不开 SpringBoot，属于依赖关系。

参考：

https://blog.csdn.net/weixin_44852524/article/details/108528093

4. MYSQL 中左连接（left join）右连接（right join）内连接(inner join)的区别

左连接的基础表为左侧数据表，先查左侧表，再加上右侧表与左侧表之间匹配的数据，如果右侧表没有则填空；

右连接的基础表为右侧数据表，与左连接相反，先查右侧表，再加上左侧表与右侧表之间匹配的数据，如果左侧表没有则天空；

内连接是将左右两边的表都存在的数据返回；

左连接和右连接为外连接；

参考：

https://blog.csdn.net/weixin_36098377/article/details/106482084

5. RESTFUL

RESTFUL 调用是一种基于 REST 架构风格的远程调用方式，是 HTTP 接口调用的一种特殊实现；RESTFUL 调用 URL 表达形式必须遵循 RESTFUL 架构风格的 URL 格式规范，调用方法只能使用 GET,POST,PUT 和 DELETE 四种请求方法。通常使用 JSON 或 XML 数据格式进行数据传输。要求每个资源都有一个唯一的标识符 URI，每个请丢都是独立的，服务器不会在请求 i 之间保留绘画状态以便将来使用。客户端可以在收到数据后将其存储在本地缓存中，在下一次请求相同资源时可以减少网络传输，提高性能。

参考：

https://blog.csdn.net/qq_44778023/article/details/130808597

6. YAML

YAML（/ˈjæməl/，尾音类似 camel 骆驼）是一个可读性高，用来表达数据序列化的格式。YAML 参考了其他多种语言，包括：C 语言、Python、Perl，并从 XML、电子邮件的数据格式（RFC 2822）中获得灵感。Clark Evans 在 2001 年首次发表了这种语言，另外 Ingy döt Net 与 Oren Ben-Kiki 也是这语言的共同设计者。当前已经有数种编程语言或脚本语言支持（或者说解析）这种语言。

YAML 是"YAML Ain't a Markup Language"（YAML 不是一种标记语言）的递归缩写。在开发的这种语言时，YAML 的意思其实是："Yet Another Markup Language"（仍是一种标记语言），但为了强调这种语言以数据做为中心，而不是以标记语言为重点，而用反向缩略语重命名。

YAML 的语法和其他高级语言类似，并且可以简单表达清单、散列表，标量等数据形态。它使用空白符号缩进和大量依赖外观的特色，特别适合用来表达或编辑数据结构、各种配置文件、倾印调试内容、文件大纲（例如：许多电子邮件标题格式和 YAML 非常接近）。尽管它比较适合用来表达层次结构式（hierarchical model）的数据结构，不过也有精致的语法

可以表示关系性（relational model）的数据。由于 YAML 使用空白字符和分行来分隔数据，使得它特别适合用 grep/Python/Perl/Ruby 操作。其让人最容易上手的特色是巧妙避开各种封闭符号，如：引号、各种括号等，这些符号在嵌套结构时会变得复杂而难以辨认
YAML 语法：
大小写敏感，使用缩进表示层级关系，缩进不允许使用 TAB，只允许使用空格，缩进的空格数不重要，只要相同层级的元素对齐即可，#表示注释。

参考：
https://blog.csdn.net/Hh20161314/article/details/104351224

7. NACOS

Nacos 是 Dynamic Naming and Configuration Service 的首字母简称，一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。

Nacos 致力于帮助您发现、配置和管理微服务。Nacos 提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。

Nacos 帮助您更敏捷和容易地构建、交付和管理微服务平台。 Nacos 是构建以“服务”为中心的现代应用架构（例如微服务范式、云原生范式）的服务基础设施。
服务（Service）是 Nacos 世界的一等公民。Nacos 支持几乎所有主流类型的“服务”的发现、配置和管理：Kubernetes Service、gRPC & Dubbo RPC Service、Spring Cloud RESTful Service
Nacos 的关键特性包括: 服务发现和服务健康监测，动态配置服务，动态配置服务，动态配置服务

参考：
https://nacos.io/zh-cn/docs/what-is-nacos.html

8. GRPC

更多人可能接触的更多的是基于 REST 的通信。我们已经看到，REST 是一种灵活的体系结构样式，它定义了对实体资源的基于 CRUD 的操作。 客户端使用请求/响应通信模型跨 HTTP 与资源进行交互。尽管 REST 是广泛实现的，但一种较新的通信技术 gRPC 已在各个生态中获得巨大的动力。

gRPC，其实就是 RPC 框架的一种，前面带了一个 g，代表是 RPC 中的大哥，龙头老大的意思，另外 g 也有 global 的意思，意思是全球化比较 fashion，是一个高性能、开源和通用的 RPC 框架，基于 ProtoBuf(Protocol Buffers) 序列化协议开发，且支持众多开发语言。面向服务端和移动端，基于 HTTP/2 设计，带来诸如双向流、流控、头部压缩、单 TCP 连接上的多复用请求等特。这些特性使得其在移动设备上表现更好，更省电和节省空间占用。

在 gRPC 里客户端应用可以像调用本地对象一样直接调用另一台不同的机器上服务端应用的方法，使得您能够更容易地创建分布式应用和服务。与许多 RPC 系统类似，gRPC 也是基于以下理念：定义一个服务，指定其能够被远程调用的方法（包含参数和返回类型）。

在服务端实现这个接口，并运行一个 gRPC 服务器来处理客户端调用。在客户端拥有一个存根能够像服务端一样的方法。

参考：

https://zhuanlan.zhihu.com/p/411315625

9. 时序数据库

时序数据库全称为时间序列数据库。时间序列数据库指主要用于处理带时间标签（按照时间的顺序变化，即时间序列化）的数据，带时间标签的数据也称为时间序列数据。
时间序列数据主要由电力行业、化工行业、气象行业、地理信息等各类型实时监测、检查与分析设备所采集、产生的数据，这些工业数据的典型特点是：产生频率快（每一个监测点一秒钟内可产生多条数据）、严重依赖于采集时间（每一条数据均要求对应唯一的时间）、测点多信息量大（常规的实时监测系统均有成千上万的监测点，监测点每秒钟都产生数据，每天产生几十 GB 的数据量）。

基于时间序列数据的特点，关系型数据库无法满足对时间序列数据的有效存储与处理，因此迫切需要一种专门针对时间序列数据来做优化的数据库系统，即时间序列数据库。
对于时序大数据的存储和处理往往采用关系型数据库的方式进行处理，但由于关系型数据库天生的劣势导致其无法进行高效的存储和数据的查询。时序大数据解决方案通过使用特殊的存储方式，使得时序大数据可以高效存储和快速处理海量时序大数据，是解决海量数据处理的一项重要技术。该技术采用特殊数据存储方式，极大提高了时间相关数据的处理能力，相对于关系型数据库它的存储空间减半，查询速度极大的提高。时间序列函数优越的查询性能远超过关系型数据库，Informix TimeSeries 非常适合在物联网分析应用。

IoTDB 是针对时间序列数据收集、存储与分析一体化的数据管理引擎。它具有体量轻、性能高、易使用的特点，完美对接 Hadoop 与 Spark 生态，适用于工业物联网应用中海量时间序列数据高速写入和复杂分析查询的需求。

参考：

https://iotdb.apache.org/zh/UserGuide/V1.0.x/IoTDB-Introduction/What-is-IoTDB.html