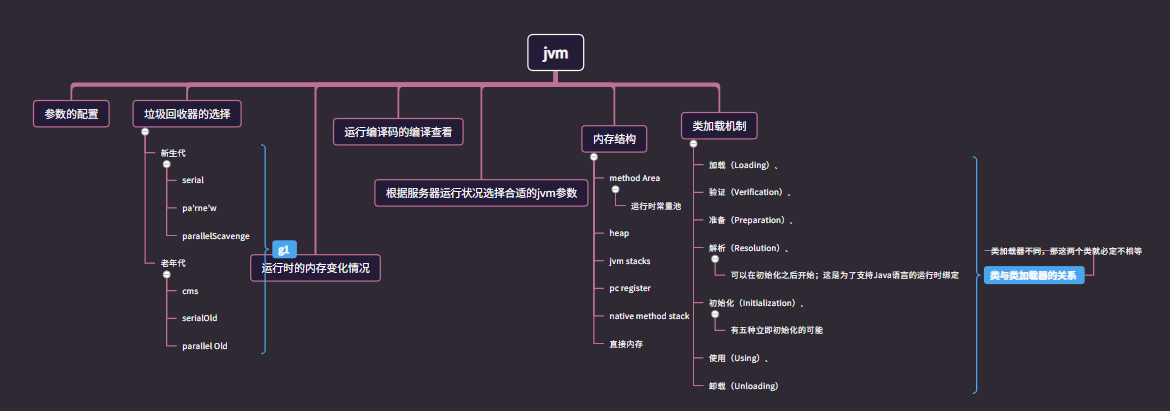
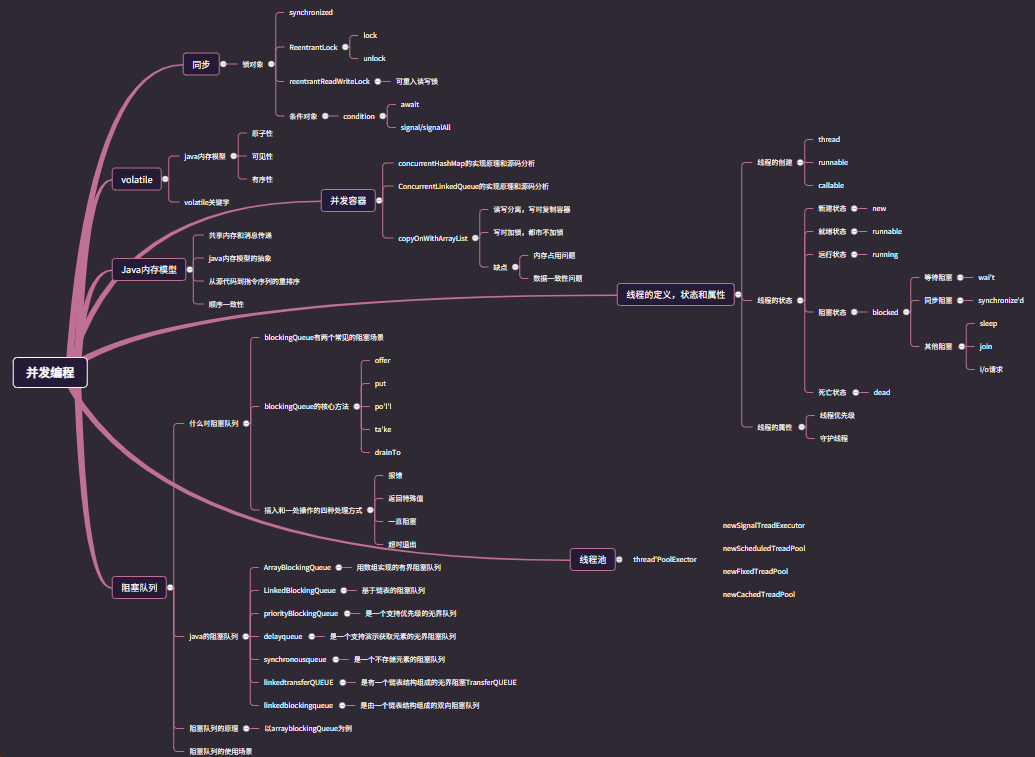
Jvm：需要记得清楚的是jvm内的内存分布情况，以及类加载时的具体步骤，调用了哪些类加载器；在运行java时参数的输入，还有gc的选择，运行过程中汇编码的观看，查看运行过程中那些类调用过多 还有具体的分析工具 用分析工具看那些类是调用过多，哪些类内存占用过多。记清楚gc的使用场景



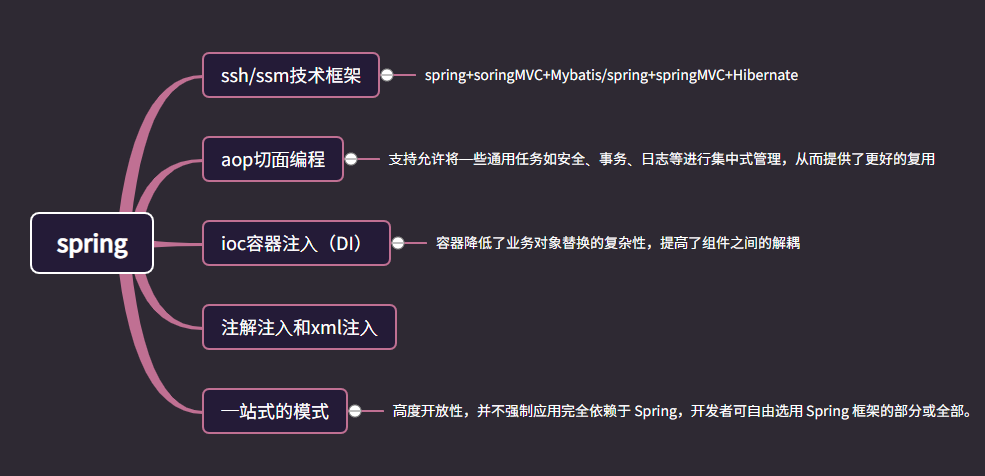
阻塞同步类似于将数据投送到一个管道之中，然后服务端和客户端可以自由的向管道之中投送数据和内容 并且自由的在管道内去除数据， 这样数据的传输过程就不会被堵塞住，而且有较强的并发能力，之后再将数据去除时，在经过selector选取出客户端所需要的数据内容



并发编程需要注意锁的粒度问题，锁越大，并发的时候效率越低；使用volatile修饰是因为在jvm中new Singleton()会出现指令重排，volatile避免happens before，避免空指针的问题。从一个线程安全的单例模式可以引申出很多，volatile和synchronized的实现原理，JMM模型，MESI协议，指令重排

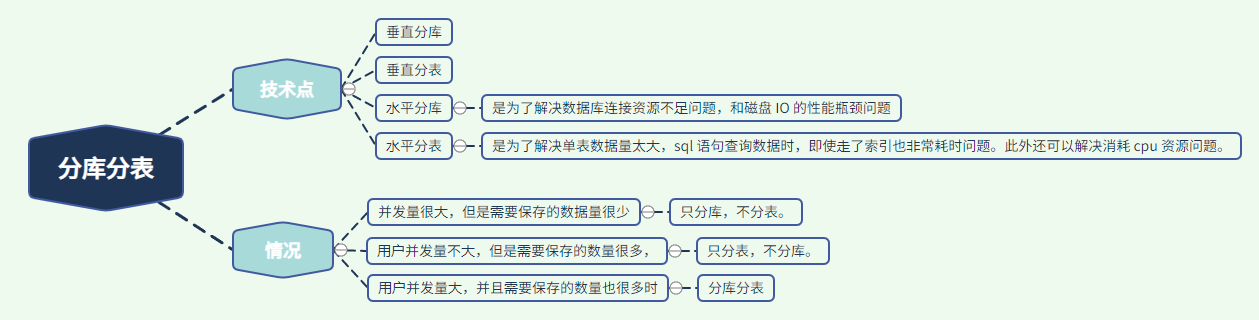


Spring有两种数据库注入，是ssh/ssm；aop的切面编程，能够让代码变得简洁，让关键的代码部分清晰起来；ioc注入的两种方式，能够给代码进行解耦，让类的调用变得简洁；高度的整合，spring可以整合很多个组合模块，可以方便的调用的其他组件进行加强代码



分库分表其实理解起来很简单，主要是实操的难度；总结：单表数据多且查询多 要水平操作 如果是磁盘空间不够 那就分库

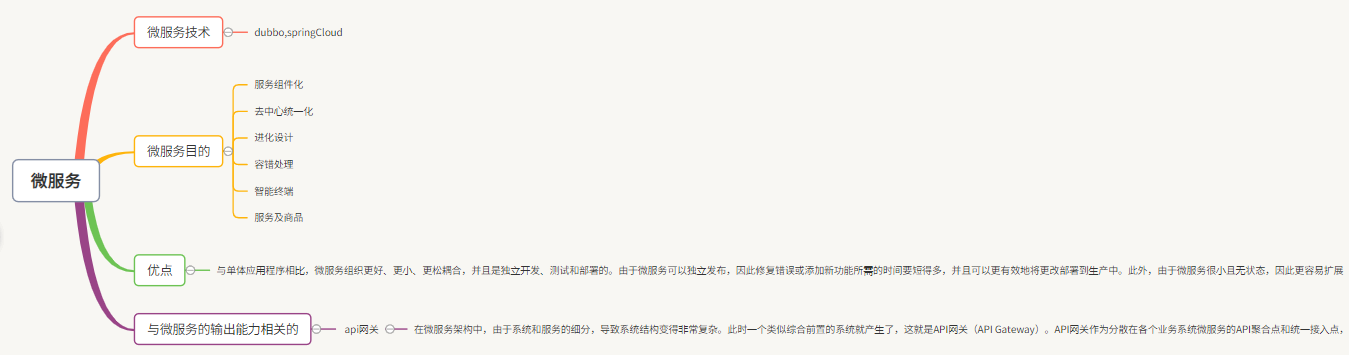
分库分表可以利用sharedingspere-proxy进行分库分表操作 但是记得调校sharedingspere。Xml文件的，在其中设置表

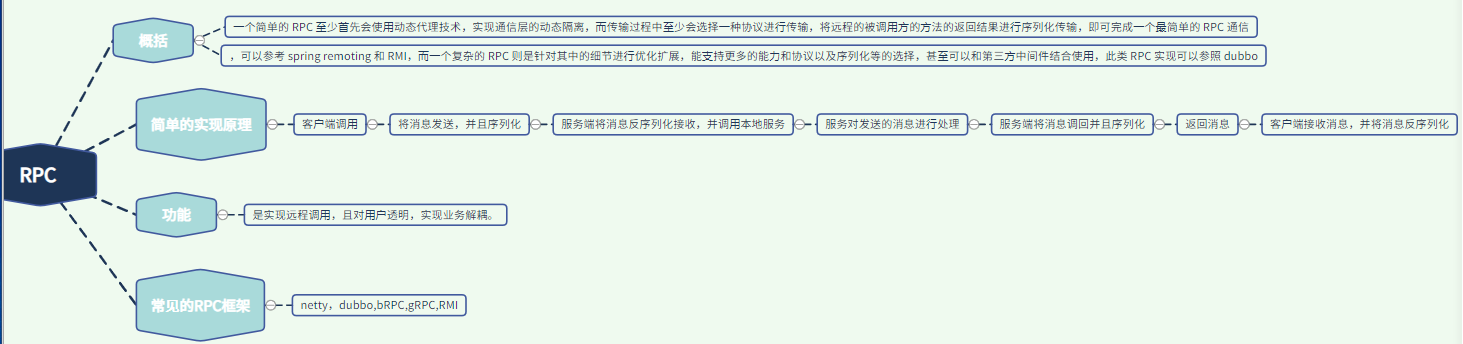


Mysql细节操作可以查看《mysql实战》 操作过程中尽量少用索引操作，避免一次性过多的调用数据造成sql查询锁死 如果查询死机了可以使用 kill-9命令 杀死进程 一定要在保证查询的目的正确才可以调用表索引 适度的表笼余是可取，因为考虑到使用性的问题，七大范式一般只要求前四大范式

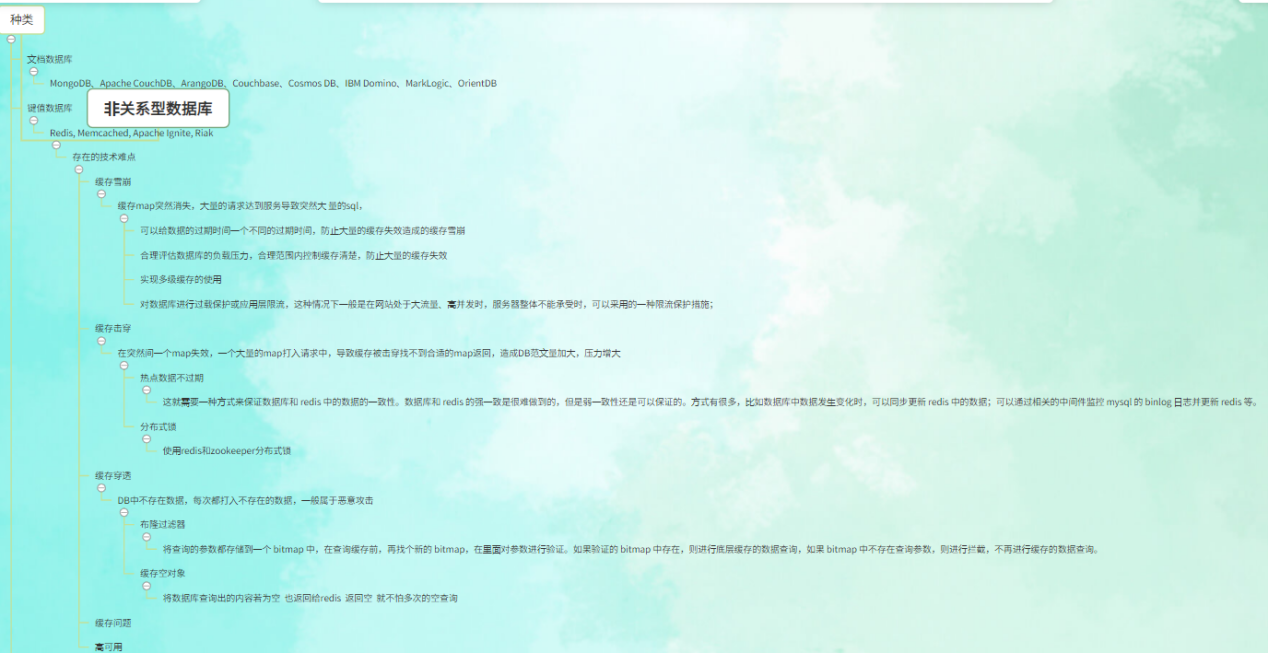


微服务的大部分都是基于rpc原理进行操作的 rpc中具体的消息回调方式有多种 http tcp pnc netty等多种数据调用方式 微服务我认为最核心的作用 莫非是 将模块分离，团队合作，方便后期维护，防止模块樊笼；Rpc作为大多数微服务的核心功能模块需要吃透 可学习书籍有 rpc原理性解剖





非关系型数据库国内现在大多都使用redis redis自身也有分库分表也能 在一些高并发场景中应用非常广范 难点和mysql差不太多 只是自身还经常作为服务全局变量使用 redis作为缓存中间件的使用课上有简单实例，我自身不太懂得地方其实有redis的锁使用，因为没有过具体的实操 所以还只是很浅面的使用



消息队列 现在大多用于秒杀系统 高并发系统 削风平谷 模块解耦的作用，我对于他的理解也很浅薄，目前还在看资料中

