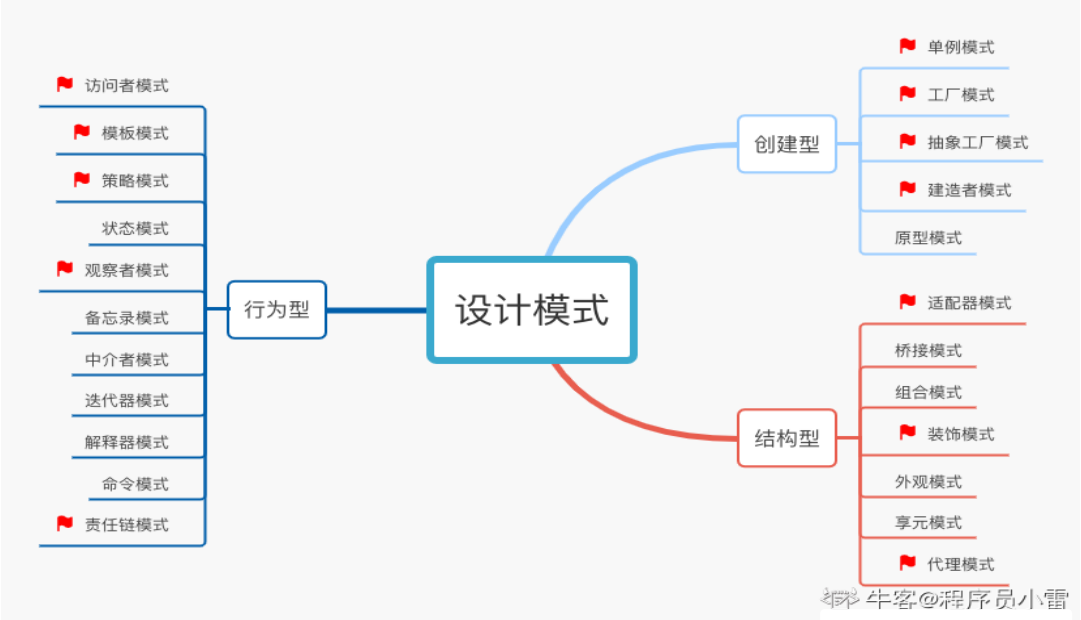
主要内容包括(干货满满)：
学 Java 有哪些就业方向？
数据结构和算法
设计模式
计算机基础
Java 入门
Java 高手进阶
基础框架（SSM）
微服务框架
常用中间件
数据库
分布式架构
必须掌握的工具软件
学习资源网站列表汇总
学习常见问题（FAQ）
学 Java 有哪些就业方向？
在介绍 Java 怎么学之前我给大家介绍一下学完了能干什么，因为有目标的学习才是最高效的。
很多 Java 入门学习者对岗位或者方向的概念非常模糊，今天学安卓、后天学大数据，三心二意的学习势必造成技术不精，这就是面试官通常说的：这位面试者基础比较差。
学习技术首先要认准一个方向专注下去，有了一定积累后再将自己的知识面扩宽，找到自己感兴趣的方向再沉下去学习，周而复始你就成为这个行业的专家了。
Java 这门语言，在公司里根据分工不同衍生出了众多的岗位或者技术方向。
我在 boss 直聘上搜索了 BAT 等大厂的岗位，目前有以下三类岗位非常热门：
（1）安卓开发
技能要求：
熟悉 Android UI 开发非常熟悉，对 UI 架构有理解，并了解基础的 UI 交互知识;
熟悉 Android 调试工具和方法，可以应付各种 Android 复杂问题;
熟悉 Android Framework 层，有通过 Android 源码阅读定位问题的经验;
（2）Java 后端开发
技能要求：
具备扎实的 Java 基础，对 JVM 原理有扎实的理解；对 Spring、MyBatis、Dubbo 等开源框架熟悉，并能了解它的原理和机制，具有大型分布式系统设计研发经验；
熟悉基于 Mysql 关系数据库设计和开发、对数据库性能优化有丰富的经验；
熟悉底层中间件、分布式技术（如 RPC 框架、缓存、消息系统等）；
（3）大数据/数据仓库
技能要求：
熟悉 Hadoop/Spark/sqoop/hive/impala/azkaban/kylin 等大数据相关组件;
精通 sql 及性能调优，熟练使用 java、python、scala 其中一种编程语言;
掌握数据仓库 (DW) / OLAP /商业智能 (BI) /数据统计理论，并灵活的应用，具备大型数据仓库设计经验;
敲黑板：认清自己，找准方向，越早确定方向越容易成功！
数据结构和算法
学什么？

有些同学可能要问了：我学 Java 的有必要学习算法吗？答案是：别无选择！
国内互联网面试的流程逐渐在向国外靠拢，像字节跳动、BAT 等大厂，手撕算法题已经成为了必选动作。
确实， Java 相对于 C、C++有着丰富的类库和三方框架，进入工作后大部分人都是在写业务代码，俗称 API boy 或者 Crud boy，算法看起来并不是那么重要，但是考算法真的是公司面试筛选人的低成本办法，如果你写出了算法并且通过了，要么你聪明要么你勤奋（刷题了）。
所以不管你是学什么语言：C、C++、python、Java、GO，算法这一关你必须得过。数据结构和算法的面试核心知识点我已经列出来了，大家可以参考学习，逐个击破。
栈与队列：先进先出、后进先出
线性链表
查找：顺序查找、二分查找
排序：交换类、插入类、选择类
树、二叉树、图：深度优先（DFS）、广度优先（BFS）
递归
分治
滑窗
三大牛逼算法：回溯、贪心、动态规划（DP）
在刷题之前我建议你看一些书：
《漫画算法-小灰的算法之旅》
如果你之前没有任何算法基础，这边书很适合你，可以补充数据结构和算法的基础知识，像什么是时间复杂度空间复杂度、查找、排序等。
如果你有了一定基础了，建议你直接跳到最后面的算法实战部分。
《剑指 offer》
非常经典的一本书，学算法的人必刷。但是要注意了，这边书里面的题目是用 C++写的，如果你是 Java 开发人员可能会有点影响。但是要记住学习算法最关键的还是解题思路和方法，用什么语言实现是其次的，如果你时间比较多我是建议你用 Java 语言再实现一遍。
《labuladong 的算法小抄》
非常推荐！这是一本很新的书，写书前作者在 Github 开源了一个项目，主要讲解 LeetCode 解题套路，Start 总数排名前 40。在书的开头讲解了学习算法的基本思维和套路，建议看这边书的同时再配合 leetcode 刷题，疗效非常棒！
《算法导论》
要是不推荐这本书是不是显得我有点 low 了，这是一本科班出身的同学必看必学的经典大部头。国外大佬写的，国内翻译的经典之作，虽然是经典但是不建议刚入门算法的同学看，因为看了这本书你可能要放弃算法了，比较难看懂。建议有了一定基础再入手这边书。
如果你觉得看书比较枯燥，可以推荐你看一些极客时间的专栏，不过是收费，但是质量非常高。
《数据结构与算法之美》
这个专栏是文字+语音，作者是王争，前 Google 工程师。他采用最适合工程师的学习方式，不拘泥于某一特定编程语言，从实际开发场景出发，由浅入深教你学习数据结构与算法的方法，帮你搞懂基本概念和核心理论，深入理解算法精髓，帮你提升使用数据结构和算法思维解决问题的能力。
《算法面试通关 40 讲》
这个专栏是视频，作者是覃超，前 Facebook 工程师。作者会用白板带你一步一步解题，层

层深入一环扣一环，每一题还会用多种解题方法。我基本看完了，收获颇多。
leetcode、书和极客专栏可以并行，学练结合，不要光看不练哦。
设计模式
学什么？
金庸小说中牛叉的武功太多了，综合性最强的还是九阳真经，九阴真经分为上、下两卷，上卷为内功基础，下卷为武功招式，这些都是极负盛名的武学秘籍。
那大家思考一下什么是武学秘籍？其实打开来开就是一些固定的招式，牢记这些招式并运用好就是绝顶高手了。
回到编程上来，除了要写干净的代码（clean code），还要运用各种设计模式使代码可读性强、扩展性好、松耦合，这便是大家经常说的编码大牛。
所以不管是学武功还是学编码，都是有一些固定的招式，也就是设计模式。
说到设计模式很多同学可能会跳出来：这个我知道，就是单例模式、工厂模式……
巴拉巴拉说了一堆，但是真正在写代码的时候又是一脸蒙：为什么我写的代码用不到设计模式？究其原因是你的代码经验不够。
想一下设计模式是怎么来的？上个世纪四个大男人搞了一个组合叫 GoF，并出版了一本书，这本书共收录了 23 种设计模式，后面逐渐被人熟知。这四个人从大量的代码实践中总结了一套方法论（写代码的套路），而我们作为一个在学校的学生或者刚工作的新人，可能连代码都写的少，怎么可能轻松快速地掌握这么多设计模式。
所以说你学完了设计模式，但是还不会运用到日常的代码实践中，这个是很正常的，因为代码经验还不够。
那还学不学？当然要学，因为面试的时候有可能会问到。设计模式的理论知识我们还是要打好基础，需要掌握这些知识点：
设计模式的六大原则：单一职责、里氏替换、依赖倒置、接口隔离、迪米特法则、开闭原则
UML 基础知识
设计模式三大分类：创建型、结构型、行为型
常用设计模式基本原理
经典设计模式总共有 23 种（现在远不止 23 种了，还有一些变种），全部掌握难度太大了，我们只需要掌握一些常用的就好了，必须要掌握的我用小红旗已经标出来了。



怎么学？
网上关于设计模式的学习资料非常多，质量也是参差不齐，大家找的时候可要擦亮眼睛。
在看书之前我还是推荐你熟悉一下 UML 的理论知识，因为你如果不懂 UML 那任何一本设计模式的书你都可能读不下去， UML 是设计模式的前提。
不要花太多时间学习 UML，简单理解入门即可。
假设你已经入门 UML 了，那下面的这些书你可以考虑学习一下了：
《Head First 设计模式》
Head First 是一个比较经典的系列丛书，有些人非常喜欢这种风格。这本书讲枯燥的设计概念讲解的生动有趣，作为一本入手书非常值得推荐。
《大话设计模式》
大话系列是国内非常经典的系列丛书，有众多粉丝。这本大话设计模式以对话的形式讲解知识，在当时可开创了先河。虽然书中有些例子比较牵强，但任然不失为一本入门的好书。
《图解设计模式》
图解系列是日本的一位作者写的，有一本图解 HTTP 非常经典，这本图解设计模式也是类似的风格。由于是翻译过来的，书中有些例子可能听起来比较奇怪，貌似翻译过来的技术书都有这个问题。
《设计模式-可复用面向对象软件的基础》
又是一本黑色大部头书，书的作者就是 GoF，大家都说经典。但是呢，经典归经典，读起来真的是晦涩难懂，对新人非常不优化，如果你想入门学习设计模式，这本书就不推荐了。
不推荐为什么要说出来？经典的书如果不提，你们又要说我菜。（害）
这几本书都要看吗？当然不是，如果你是在准备面试，我个人建议是读其中一本就够了。至于说看哪一本，你可以找对应的电子书，挑一个章节试读一下，符合你的胃口就选择这一本继续读下去。
如果你已经有几年的编码经验，又想把代码写好，建议你多挑基本读读，吸收每本书的精华。
计算机基础
科班出身的同学对《计算机网络》和《操作系统》这两门课应该不会陌生，至于掌握了多少，你懂得，都是在考前一两周突击学习的，哈哈。
现在大公司对于应届生的要求越来越高，计网和操作系统这两门课是必考的。那些拿了 SSSP Offer 的大牛计算机基础都非常扎实。
（1）计算机网络
学什么？
计算网络的协议非常非常多，很多同学学完都一头雾水，或者仅仅懂一点 HTTP，但是真正要掌握的东西可不少：
OSI 七层模型、TCP/IP 五层模型
常见网络协议：HTTP、TCP/IP、UDP
网络安全：非对称加密、数字签名、数字证书
网络攻击：DDOS、XSS、CSRF 跨域攻击
怎么学？
计算机网络面试有一道非常经典的面试题：说说你从 URL 输入到最终页面展现的过程。这一题可以覆盖大部分计网的知识点，可以从 DNS 解析到 HTTP、TCP/IP 协议、物理层协议，一直到浏览器渲染页面，你技术功底有多深你就可以聊多深。希望大家学完了也能试着回答一下这个问题。
推荐几本倍受好评的书：
《网络是怎么连接的》

这本书是一本日本作者写的。文章围绕着在浏览器中输入网址开始，一路追踪了到显示出网页内容为止的整个过程，图文并茂生动有趣，非常推荐！
《图解 HTTP》
也是一名日本作者写的。这本书对 HTTP 协议进行了全面系统的介绍，列举了很多常见通信场景及实战案例，相信读完会有恍然大悟的感觉。书很薄，几天就可以读完，强烈推荐！
《TCP/IP 详解卷 1：协议》
计算机网络的经典教材， 大部头书籍，很难啃。建议挑重点看。
（2）操作系统
学什么？
作为一名 Javaer 在平时的工作中可能不会直接跟操作系统打交道，因为 JVM 帮我们屏蔽了众多差异。但是要想学好 JVM，懂一点操作系统更有助于你深刻理解 JVM 工作原理。
Java 学习者这部分的要求可以稍微放低，但是你如果是搞 C++的，那这部分可是你的重点。
进程和线程的区别
进程间的通信方式：共享内存、管道、消息
内存管理、虚拟内存
死锁检测和避免
怎么学？
想要精通操作系统难度非常大，但是在面试中你要能讲出一些具体的操作系统知识，面试官会对你刮目相看。
推荐书籍资料：
《深入理解计算机系统 CSAPP》
赫赫有名的 CSAPP，全称：Computer Systems：A Programmer‘s Perspective。科班同学的圣经，哈哈，黑色大部头书籍，难啃。
《现代操作系统 （第 3 版）》
操作系统领域的经典之作，因为是翻译过来的，遇到比较晦涩的先跳过，多读几遍才能消化。
Java 入门
学什么？
Java 语言从诞生到现在已经有 20 多年了，从 Tiobe 排行榜上来看，Java 语言常年霸榜经久不衰，所以不要怕学完 Java 后突然不流行了，至少这几年 Java 就业机会非常多。
如果你有其他语言的基础，比如之前学过 C、C++等，那学起 Java 应该是非常容易的，也容易上手。如果你没有语言基础，又不想了解太底层的东西，那学 Java 还是不错的。至于说 python，光从语言层面上看，python 确实非常简单，估计你一周内就可以学会并且代码写的还不错，但是 Java 不一样，一周你只能简单了解一下语法，想写好代码几乎不可能。另外 Go 语言势头很猛，大家也可以关注一下。
一般来说 Java 入门你需要掌握下面这些知识点：
面向过程 VS 面向对象
面向对象基本特征：封装、继承、多态
访问控制符：private、default、protected、public
数据类型：基本类型、引用类型
控制流程：for、while、switch 等
序列化
异常处理（有点难度）
泛型（有点难度）
怎么学？

如果你是零基础，建议你可以找一些 Java 入门的视频看一下，网上视频鱼龙混杂，大家注意甄别。推荐一个比较好的平台：

敲黑板啦：视频不要贪多，因为没有一个大牛是看视频看出来的。 看视频是别人将知识点往你脑袋里灌，最大的好处是能让你快速入门，如果你想学到更多，你需要的是自我学习，带有思考的自我学习。

看书是一种高效的自我学习方式，推荐基本比较好的书：

《Java 核心技术卷 I》

这本书建议作为 Java 之旅的第一本书，涵盖的内容非常全，比起那些 30 天学会 Java 之类的书，这边书更加务实。书中有些章节其实不用看，比如 Swing GUI 的直接略过，因为用 Java 写桌面端应用已经过时了。

《阿里巴巴 Java 开发手册》

大厂阿里巴巴出品的，这其实是一本 Java 编码规范，编码习惯从一开始就要养好。

《Java 编程思想（Thinking In Java）》

这是一本非常非常经典的书，你要问搞 Java 的人如果没听过这本书那算是白学了，哈哈。其实说实话这本书我试图看过几次，最终都没有看完，一个原因是它太厚了，另外我觉得讲得太啰嗦了，所以我现在拿来垫桌子，高度合适挺好的。所以呢，建议新人不要一开始看这边书，不然你会怀疑人生还没入门就放弃了，就把它当做编程圣经，等你后面有经验了拿起来再翻翻吧。

敲黑板了：学习编程要有耐心，不要急于求成，要打好基础。也许你一个月两个月还在运行一些简单示例，这是正常的，多学习多思考。

Java 高手进阶

学什么？

恭喜你终于 Java 入门了，大牛和菜鸟的区别在于菜鸟永远止步于入门水平，而大牛已经找到新大陆了，翻过这几座山你离高手就不远了。

Java 高手进阶需要掌握的东西非常非常多，这里列举一些核心知识点，必须全部掌握的。这是 Java 面试高频考点，也是传说中 Java 八股文的一部分，面好了进入下一面，面不好回家等消息。

Java 集合类源码

线程池

Java 代理

IO 模型

JVM

Java 并发编程（JUC）

怎么学？

Java 已经入门了，你都想进阶了，建议你不要再找视频看了，一边看书一边思考吧。

《Effective Java》

书中列举了很多编程建议，其实就是告诉怎样去写好代码，你需要从能写代码（入门）过渡到会写代码，这本书值得一看。如果你的编码经验比较少，那这边书你可以稍微往后延，因为看完了你可能没有感同身受。

《Java8 实战》

Java15 都出来了为什么还要学 Java8 ？因为现在很多公司都还停留在 Java8， Java8 是继 Java5 之后改动很大的一个版本，得好好学。Java8 之后的版本非常不给力，换一个 JDK 版本费时费力，收益也不明显，公司肯定不愿意动了。这边书将 Java8 所有的新特性都详细讲解了，非常推荐。

《深入理解 Java 虚拟机 第 3 版》
周志明大神写的，非常非常经典，已经更新到第三版了。Java 虚拟机也就是 JVM，JVM 是 Java 面试必考的知识，不懂这个直接回家等消息吧。这边书我看了很多遍，每次看完都有新的收获，墙裂建议大家看完。
《Java 并发编程的艺术》
这是一本专门讲解 Java 并发的书，涉及到各种锁、常见安全的集合类，基本就是将 JUC（java.util.concurrent 包的简称）里所有的内容覆盖了一遍，看完你一定有收获。强烈推荐！
上面推荐的几本书可能不太容易读懂，建议多读几遍。书中看不懂的地方可以在网上搜，多找一些优质的博客或者公众号看。
至此 Java 语言特性基本学习完了，就算达不到高手的水平，你也在正轨上了。
基础框架（SSM）
学什么？
学习 Java 语言特性可能比较枯燥，接下来可以学习基础框架动手做一些项目，比如 Java 领域非常流行的 Spring 框架，这就是为 Java 后端量身定做的，非常好用。
在 spring 流行之前，还出现 Struts 这样流行的框架，后面由于种种原因还是被 Spring 打败了。
大家在网上应该可以经常看到 SSM 的缩写，其实就是 Spring+SpringMVC+MyBatis 的缩写了。
你需要掌握以下这些：
Spring 全家桶（Spring、Spring MVC、Spring Boot）使用
ORM 框架（MyBatis、Hibernate）使用
Spring 原理
ORM 框架原理
怎么学？
学习 SSM 框架最好是动手完成一个简单的项目，建议跟着视频并且把代码敲出来，一来熟悉项目的开发流程，也可以给自己带来成就感。
敲黑板：阶段性成就感非常重要，没有这个很容易放弃学习，所以要不定时给自己定个小目标，加加鸡腿啥的。
有很多新手在做项目的时候非常纠结界面，作为一个 Java 后端程序员，你又不是全栈开发，纠结这个干什么，我的建议：要么不要界面只写接口，要么自己动手写点 html，不需要美观，实现功能即可。
跟着视频做完项目之后需要干什么？答案是：深入理解框架原理。会用框架并不代表你懂框架，作为一个有追求的程序员，懂原理是永远的必修课，谁让这一行太卷了呢，人无你有你最棒。
推荐几本书：
《Spring 基础内幕》
首先声明一下这是一本讲解 Spring 源码的书，不是教你做项目的书。如果需要深入理解 Spring 的技术原理，这是一本非常推荐的书。有点难啃，多读几遍。
《MyBatis 技术内幕》
MyBatis 是 ORM 框架的一种，在国内使用比较多，据说在国外喜欢用 Hibernate。这本书对 MyBatis 的使用和基本原理都介绍比较清楚了。
敲黑板：技术更新迭代很快，抓住技术的本质才能与时俱进。
关于基础框架这部分，大神们的学习方法是：使用框架 -> 懂框架 -> 造轮子。
微服务框架
学什么？

近些年微服务架构非常火，究其原因是因为传统的单体架构和面向服务的架构逐渐不能满足互联网快速迭代的需求。微服务可以更容易提供持续继承和持续部署的能力，让产品更快速交付推向市场。

面向服务的架构其实在五六年前就已经提出，期间经过了一段低潮期，泡沫散去后逐渐浮现了一些好用的框架，国外以 SpringCloud 为代表，国内以 Dubbo 为代表。

springCloud 和 Dubbo 有区别但是很多基本原理也是类似，大家学习的时候需要掌握技术的本质。下面列举一些核心知识点：

Dubbo 框架

SpringCloud 框架

服务注册与发现

分布式服务链路追踪

服务隔离、熔断、降级

服务网关

怎么学？

springCloud 和 Dubbo 在官网都有很详细的介绍文档：

看官网技术文档大家可能会很懵，但这些确实是最权威的资料，也是一手的。

SpringCloud 和 Dubbo 是这几年刚刚流行的技术，从目前看来相关书籍还是比较少，也缺少一些经典的书，我还是列几本，大家按需获取。

《深入理解 Apache Dubbo 与实战》

Dubbo 最开始是阿里巴巴开源的，后面捐赠给 Apache 了。建议大家读这本配合源码一起看。

《Spring Cloud 微服务实战》

读这本书之前你最好先学习 spring 和 spring boot，不然会很懵。另外这本书是 2017 年出版的，稍微显旧，大家注意分辨新旧特性。

如果技术网站和书籍还不能满足你，建议你去搜一些视频学习，这里不做推荐以免认为是广告。推荐搜索平台：B 站、慕课网、网易云课堂。

敲黑板：微服务框架涵盖的内容非常多，也是有难点的技术，大家戒躁保持耐心。

常用中间件

学什么？

最终用户并不直接使用中间件，换言之中间件不是大众消费类软件产品。但是在大公司里中间件是不可或缺的，它是支撑大型网站架构的一些基础的组件和服务，所以非常非常有必要学。

小百科

中间件（Middleware）通常是指在一个大型分布式的系统中，负责各个不同组件（Component）/服务（Service）之间管理以及交互数据的。

业界开源的优秀中间件非常多，通常会根据业务的需要在系统中引入若干，下面列举了一些常见的，都是必学的，非可选哈。

缓存：Redis、Memcached（ 推荐 Redis）

消息队列：Kafka、RocketMQ、RabbitMQ、ActiveMQ、ZeroMQ（推荐 Kafka）

数据库中间件：ShardingSpere、Mycat

怎么学？

推荐几本相关的书：

《Redis 设计与实现》

这时 Redis 口碑比较好的一本书，书中详细讲解了 Redis 实现原理，如果你只是想学会怎么用，可以跳过一些章节。

《深入理解 Kafka：核心设计与实践原理》
这本书既适合新手入门扫盲也适合高手进阶，想知道怎么用看前四章即可，想深入学习可以从第五章开始看，写的非常好，推荐学习！
《分布式数据库架构及企业实践——基于 Mycat 中间件》
Mycat 相关的书非常少，这本书是 16 年写的，有些陈旧了，如果对 Mycat 非常感兴趣可以简单翻一翻，但是不是特别推荐。
书看完了你还想深入学习，建议大家关注一下极客时间的两门课：
胡夕：《Kafka 核心技术与实战》，老师是 Apache Kafka Committer，很专业。
蒋德钧：《Redis 核心技术与实战》
不过课程是付费的，手头紧的建议慎重哈。免费资源网上也有，靠大家搜索了~
中间件的学习是一个漫长的过程，不仅需要很多理论知识还需要实践经验。
比如你学 Redis 的时候，要思考五种基本数据类型各自使用场景、布隆过滤器是什么原理、用 Redis 怎么实现分布式锁，带着问题去学习效率非常高。
比如你学 Kafka 消息队列，要对比常见消息队列的优缺点、Kafka 为什么吞吐量高、Kafka 会不会丢消息以及怎么解决。
比如你学数据库中间件，要想数据库为什么要分库分表、分库分表 ID 如果处理等等。
数据库
学什么？
数据库非常重要，面试也是必考的，可以考的点非常多，可以考得很浅：问一下 SQL 使用，也可以考的很深：问索引和锁的实现原理。下面列了一些常见的知识点。
数据库基本理论：范式、索引原理、数据库引擎
SQL 基本语法
SQL 调优，explain 执行计划
数据库事务（ACID）
数据库锁：乐观锁、悲观锁、表锁、行锁等
怎么学？
建议数据库零基础的同学还是要先学习一下数据库的基本理论，因为我看到很多人都是一上来就学 SQL ，最终也只是会用而已，到后面 SQL 调优的时候就很迷茫了。如果你只是想用一用数据库，这部分也可以跳过。
关于原理部分有一本非常经典的教材《数据库系统概念》以供学习，经典书籍一般都比较难啃坑也比较厚，建议大家先看目录，挑重点看。大学学过这本书的可以直接跳过了。
有了一些理论后就可以开始学习 SQL 语法了，这里推荐一本《MySQL 必知必会》，一边看书一边对着电脑敲。
当然面试大厂肯定会问一下比较难的东西，你需要搞懂索引的原理、事务 ACID、锁，问数据库这些东西必考哦！
MySQL 学习书籍清单：
《数据库系统概念》
经典数据库教材，理解一些基本原理，可略看。
《MySQL 必知必会》
SQL 语法入门好书，推荐！
《MySQL 技术内幕 : InnoDB 存储引擎》
数据库进阶必看，理解存储引擎以及事务、锁、索引等原理。
分布式架构
学什么？

分布式这一部分就是面试的加分项了，答好了面试官会觉得你技术功底深厚，答不好，只要你前面的基础还不错也能过。所以呢，作为一个有追求的技术人，千万不要放过加分的机会。
分布式相关的内容非常多，下面列举几个在项目中或者面试中经常会遇到的知识点：
分布式事务：两阶段提交（2PC）、补偿事务（TCC）
分布式锁：基于关系型数据库（MySQL）、基于 Redis、基于 Zookeeper
分布式 ID：雪花算法（Snowflake）、美团 Leaf
怎么学？
这部分内容学好非常难，在很多书中都是轻轻带过，没有深入讲解原理，所以就不推荐书了。那怎么学呢？大家可以针对每个知识点到网上搜索优质的博客，后面我也会逐步更文讲解这些知识点，敬请期待，欢迎催更哟。
必须掌握的工具软件
工欲善其事，必先利其器。作为一个 Java 开发人员，你需要学习业界常用的软件，软件工具用得越熟你的编码效率越高，下班的时间可能越早（打工人太难了）。
Java 最聪明的 IDE：IntelliJ IDEA （请放弃使用 Eclipse，我有一堆理由睡服你）
地球上最好用的版本管理工具：Git
经久不衰的依赖管理工具：Maven
Docker
这些软件你要是用不好，那只能说明…… 你再多学学吧。
学习资源网站列表汇总
（1）Github
Java 知识地图（推荐）：https://github.com/smileArchitect/JavaMap
（2）阿里师兄 JAVA 核心知识点整理（283 页，超级详细）
链接:https://pan.baidu.com/s/1Q_7451jkUgAlaXsNbBNvww 提取码:8oj5
（3）视频网站
B 站（推荐）：https://www.bilibili.com/
网易云课堂：https://study.163.com/
极客学院：https://www.jikexueyuan.com/
慕课网：https://www.imooc.com/
（4）专栏
极客时间（推荐）：https://time.geekbang.org/
Gitchat https://gitbook.cn/
（5）技术博客：
CSDN 博客：https://blog.csdn.net/
博客园：https://www.cnblogs.com/
掘金社区（推荐）：https://juejin.cn/
InfoQ：https://xie.infoq.cn/
思否：https://segmentfault.com/
开源中国：https://www.oschina.net/blog
（6）搜索引擎：
百度：https://www.baidu.com/
谷歌：https://www.google.com/
（7）知识问答：
知乎（推荐）：https://www.zhihu.com/
stackoverflow（推荐）： https://stackoverflow.com/

（8）刷题：
力扣（推荐）：https://leetcode-cn.com/
牛客：https://www.nowcoder.com/
（9）云笔记：
石墨：https://shimo.im/
语雀：https://www.yuque.com/
有道云笔记：http://note.youdao.com/
印象笔记：https://www.yinxiang.com/
看个人习惯去选择，不推荐了。
（10）在线画图：
processOn：https://www.processon.com/
drawio：https://app.diagrams.net/
各有特色，都推荐。
学习常见问题（FAQ）
（1）学了容易忘怎么办？
这是大家学习会遇到的头号大问题，怎么解决？重复学习。
打个比方，假如你正在学习 spring 注解，突然发现了一个注解@Aspect，不知道干什么用的，你可能会去查看源码或者通过博客学习，花了半小时终于弄懂了，下次又看到@Aspect了，你有点郁闷了，上次好像在哪哪哪学习，你快速打开网页花了五分钟又学会了。
从半小时和五分钟的对比中可以发现多学一次就离真正掌握知识又近了一步。
人的本性就是容易遗忘，只有不断加深印象、重复学习才能真正掌握，所以很多书我都是推荐大家多看几遍。哪有那么多天才，他只是比你多看了几遍书。
（2）推荐这么多书都要看完吗？
当然不是！有一些书都是同类型的，作者写书的侧重点不一样，大家要学会挑重点看。
拿到一本书，首先要把目录多看一遍，一般而言书的前几章都是介绍型的内容，如果你已经有了基础，可以直接跳到后面原理解析或者实战部分。
（3）需要学多久才能成为技术大牛？
学习无止境！
业界说法，通过不断努力学习，一到两年可以达到初级水平，三到四年达到中级水平，五年可以达到高级水平。
实际上每个人的学习能力和精力不一样，时间参考意义不大。
只要你在一个方向或领域有自己的建树，就可以叫你大牛；如果你在公司是技术骨干、技术专家、架构师，也可以称之为大牛。
敲黑板：技术学习千万不要浮躁，谦卑一点多学一点，天外有天。
（4）现在 python、Go 语言很火，要不要直接学它们？
不要纠结语言，语言只是工具。今天 Go 很火，明天会有其他语言。
我有一个同学毕业去阿里写 Java，后面跳槽到深圳腾讯写 C++，现在又跳到字节跳动写 Go，在大佬面前这些语言只是语法不一样而已。所以建议大家打好基础，答应我一定打好基础。
（5）学习很难坚持下去怎么办？
建议大家学习的时候找一些兴趣相同的人一起，一个人学习很孤独，一群人学习可以相互鼓励互相促进，不容易放弃。