《十年沉思录》

题记：

弹指十年，终成此文，有名其曰：沉思录。

本系列仅是一家之言，只供参考，望读者能明鉴，取其用，弃之误。

因个人原因，武断误信，误入歧途，以致劳心伤神，事倍功半，概不负责。

最后告知：请不要过来找作者背锅。

联系作者

作者：MyBatis 中文官网，网址：http://www.mybatis.cn/，更新时间：2019 年 5 月

前言：作者的十年经历

一、科目的学习方法总结

2007 暑假留校复习高数，一直到 2008 年元旦，方才悟出数学考研之道，然后极短的时间大幅提升数学成绩，最终考研有惊无险。

2009 年开始准备考六级，适逢改革初期，听力占分比重很大，恰恰听力是弱项，第一次一败涂地考分很低，后来狠狠的背诵短文，上百篇小短文，第二次发挥失常，差一点就过了，第三次终于考过了。这两年（2009 年和 2010 年）是人生中最重要的两年，终于把英语搞明白了，掌握了通过六级的秘籍了。

虽然考过六级之后，再也没有花费大量的时间去学习英语。但是看到别人能流利的使用英文书写文章，内心甚是消沉，内心一直摸索如何将英语与编程统一起来。皇天不负有心人。终于在某天将两者统一了起来，心中无限感慨。

虽然数学成为了强项，但是除了考试以外，难以有其他的用武之处。甚至都没有看透，数据结构是数学的分支。数据结构，一直以来觉得以概念和理解为主，后来经过不断的思考， 发现：数据结构要刷题，刷到足够的量就是牛人。后悔当初没有采用题海战术的方式学习数据结构。目前来看，数据结构自认为学习的还可以，只是后悔走过的弯路。

二、编码的秘籍

读源码往往读起来头晕，恶心，这是因为大脑接受的消息量过大导致的。好比，秒杀把系统拖挂道理一样。后来，通过降维的绝招，终于解决了阅读大量源码的秘籍。

羡慕别人写的行云流水，后来才发现，这里面也有秘籍，目前也掌握了。

双重校验，编个故事，记一辈子。

一致性哈希，编个故事，永远不忘。

Listener 的用法，看似高大上，换个角度来看，非常简单。扔掉事件和监听的想法，就当做是一种调用而已。

Java 多线程方面，没有几个人能掌握和吃透，虽然市场上书很多，但是并无境界。幸亏看了看 VC++的内容，终于得道升仙。蓦然回首，花费了八年时间。

高并发的关键是 epoll，目前已经情有独钟。深刻的认识到：搞 Java 的人永远搞不定高并发，因为他们不接触 C/C++。

很久之前看新入职的同事，给大家分享 Paxos 和 ZooKeeper，内心感觉好厉害啊，比我厉害多了，深深不安。后来，把 ZooKeeper 变成了我的技术分水岭。技术上的飞跃就是从 ZK 开始的。

对 YARN 心有独爱，这块儿越复杂感觉越有意思。

Kafka 好久不看了，但是感觉悟出了里面的道理，算是占领了一块儿高地。

数组和哈希原理是有联系的，独创提出。

Shell 搞了很多年，感觉还是不行，只有某一天才发现，原来这样玩儿。终于从量变引发了质变。

不羡慕那些参与开源项目的 commit，没有完整做过项目的经验，只是在添砖添瓦而已。而且，很多东西，虽然是他们搞出来的，但是未必能深刻的说出道理来。

三、认知数学

明确了数学是什么？数学，是思维的工具。
推导出时间不是均衡流动
解决了读座右铭和马尔科夫链的关系
明白了田忌赛马的道理
认识到正态分布的深刻性

四、未来十年的规划

希望建立一个实验室，专注基础领域的研究，做到不与凡花争奇艳。

建立一个在线大学，希望做些与数学相关的内容传播，当然，人工智能属于数学的一个分支而已。在线大学，既能传播自己的一些学习心得和新的发现，也能挣些钱以安身立命。我所说的大学，可不是培训机构，可不是卖视频教程。

中国还有很多人处于贫困和无望之中，做一个有担当的人，去帮助这些人。

第一节：如何提高四六级的英语听力

首先要明白一个重要道理：听力的考察，实际上考察的还是说的能力。如果考察口语水平，其实教育部和考试院的投入成本很大，而且不容易设立公平和公正的考核标准。

明白了上述道理，下一步就好说了，务必要多读多背。比起多读我更乐意去多背，我感觉熟练地背诵一篇文章的的效果，顶上读一百篇文章的效果。

有学生给我说，他考了一次托福，一个感受就是自己的听力能力很差，一个是辨音能力比较弱，一个是长时间听时容易疲劳分心。

听力的提高在于背书，听力考察的不是听，而是说的能力。之前给他说过的道理，好像他并没有听进去。虽然他在听听力的时候都会去复述相关内容，但是我仍然建议他要背一定数量的文章。我之前背了 200 多篇文章，感觉对听力很有帮助。

英语听力理解要求的时间非常短，必须在极短的时间反映出对说的是什么意思，要想形成这么快的反应能力，必须走背诵这条路了。之前我也曾使用过复述和多读的方式，但是感觉效果不大，直到我改变了学习方法，以背诵为主之后才出现了明显的提升。多背是根本，多读是辅助，不要搞的本末倒置了。任何想提高英语四六级的同学都可以试一试多背这个学习方

法。

第二节：如何学习数学?

群成员提问：群主有木有学习数学的经验?

群主回答：

目前来看，我这辈子能走到今天靠的就是数学。你这个问题问到我身上，那是真问对人了。在数学方面，怎么学，我算是研究的很透彻。但是我要讲的话，估计得讲几天几夜都难以讲完。

有的人能学好数学，但是他说不出来，怎么能学好；有的人学不好数学，他说的其实都是错误的。我属于那种能学好，也知道怎么学的。

把考试大纲拿过来，把需要了解和理解的知识点全部扔掉，把需要掌握的知识点，往死里学习，往死里练习，把这些知识点倒背如流都不为过。拿到考卷，自然就知道怎么做了，就是碰到不会做的题，想一想那些知识点还没有用到，套进去就可以了。

考卷有个要求：必须覆盖 90%以上的知识点（需要掌握层次），这个规定其实就是一个最大的漏洞。

这个是考试数学的方法。另外，要想在数学上有创新，那还得靠悟性，可不是刷题就刷出来的。

数学，一般都是三十岁之后才能学好的科目。

任何一种经验都是一种绝活，掌握到火候才能发挥出巨大的威力，没有掌握到火候，顶多算是看看热闹而已。

第三节：读源码？NO，别被人忽悠了。

培训机构怎么吸引眼球呢？分析源码啊。公众号怎么涨粉呢？分析源码啊。当真读源码的方式就是最好的学习途径吗？非也。

我不认可过分强调读源码的学习方式。我觉得：上来就读源码，这种学习方式是大错特错的。

一定先熟悉使用文档，这才是关键。大多数人读完源码，不能说没有提升，但是相比投入时间，产出比太低了。培训机构，都靠挖掘源码当噱头、当卖点，我们不要上他们的当。

出去面试，一个人数据结构和算法厉害，一个人读源码多，肯定面试官选择第一人。学习技术，我们要明白那些是重点，那些是非重点。

学习技术，一定要把使用文档搞清楚，文档本身也是源码，是源码的更高层封装，我们读这些使用文档，其实也是在读源码的。而且，别人写好的文章，我们读一遍，相当于借力，这样对我们的成长速度很有帮助。

如果框架学习的很好了，而且有业余时间，也有兴趣，那么可以看看源码。

我读 MyBatis 源码花费了很多时间，忽略了使用文档，感觉效果不好。后来我学习 zk 的时候，不去读 zk 的源码，但是我搜遍了 zk 的介绍和使用，我发现 zk 的使用远比 mybatis 更有感觉。

虽然我也有时候读源码，但是我不鼓励大家去读源码，而且不建议上来就读源码，另外一定还要控制住量，否则的话，变成了码海战术了，肯定是个大弯路。

别说初学者了，中高级开发者也很容易犯过分强调读源码的毛病。因为我们的环境无时不刻有人在误导你，例如培训机构，公众号等等。做开发真不容易啊，到处都是套路！一不小心，你就被误导，被焦虑，被灌鸡汤了。

最后，强调一点：笔者不是反对读源码，而是要明白做事要讲究先后顺序。

补充一点吧，比读源码提升更快的学习方式应该是写工具，写一个属于自己的东西。这东西很实在，拿得出手，贴简历上是亮点，而且能锻炼人。比起读源码更难，有难度自然提升才更快。

第四节：外包要不要去?

某群成员的观点如下：

我总结下，像我这种普通本科的：外包的好处就是能接触到一些优秀的人，甲方有些人可能能力不咋地，但是他们绝对是认真负责的。这一点很值得学习。 坏处：工作毫无体验可言，没有归属感。

我的观点是：任何岗位不要待的超过 2 年，最好 1 到 2 年左右即可，可以去跳槽或者内部岗位轮换。除非非常好的岗位，不再此列。同一岗位待的过长，就变成重复工作了。重复工作是技术人的最大杀手。

立足于在这个大的原则下，去不去外包不是一件纠结的事情，反正到哪里都会走人的，顺心就多待，不顺心就少待。

跳出舒适圈，说白了，就是要多挑战自己，多跳槽也是挑战自己。

重复工作一两年，没有成长和突破，当时可能感觉无所谓，等年龄大了就会明白了，一两年真的很关键的，在生命中其实很重要的。一两年有的人可能提升很快，一两年有的人几乎就没有什么发展，等后面的新人涌入，那还有那么的机会等着给你啊，那个时候就憋屈了。

第五节：编程的艺术

背景介绍

本部分内容出自个人写的项目的开发经验，因为涉及商业利益，不再公开了。人总是要吃饭的。这些项目能挣点钱，让我做自己喜欢的事情，例如收集域名，购买图书等等。

5.1 编程不是大盒子装小盒子

在实际生活中，大盒子装小盒子，大盒子能感知小盒子的存在，但是小盒子无法感知大盒子的存在。

而在编程过程中，大盒子装小盒子（大盒子持有小盒子的引用），而且小盒子也要感知大盒子的存在（小盒子也要持有大盒子的引用）。

5.2 论函数转内部类实现异步执行的设计思想

我一直觉得：只做 Java 的人，永远成不了 Java 高手。下面内容是顿悟出来的，是基于 Python 语言的特性：函数，也是一种对象。立足这个思想，将 Java 里面函数，升华成内部类，而且让原功能带上线程池+异步执行的特性。具体如何实现呢？请看下文：

类里面包括两种物质：属性和函数。在某些编程语言里面，将函数也当做一个对象。从这个角度来看，函数就像一个内部类。

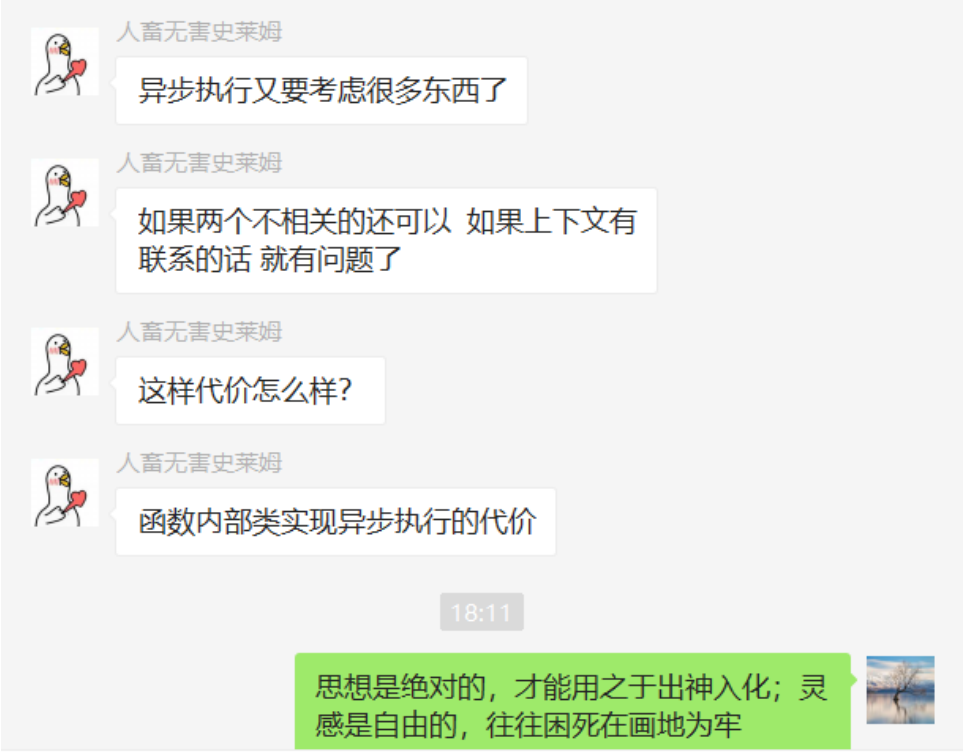
把函数当做内部类，这种想法别出心裁，挺新颖的，但是并没有特别使用之处。可以将其彻底升华一下，从而将这种想法发挥出巨大威力，以提升代码的设计能力和思想境界：

（1）函数变成内部类，而这个内部类 implements Runnable，给函数的执行增加了异步的功能。这个内部类，可以是普通的类，也可以看做一个事件类
（2）函数变成内部类之后，之前函数的参数变成内部类的属性
（3）函数变成内部类之后，之前函数的函数体不变，而此时内部类提供 run 方法，去调用旧的函数体。
（4）函数变成内部类之后，外部类增加线程池，外部类执行函数的过程变成了：将参数封装成内部类，然后扔到线程池，进行异步执行的过程。

后记：

有的人提了一堆的问题，对此类异议不再一一回复，统一回答为：

思想是绝对的，才能用之于出神入化；灵感是自由的，往往困死在画地为牢。



5.3 死类和活类

有的类是死类，例如 Person，只有赋值与未赋值两种情况，还有一种类是活类，生下来就活蹦乱跳的。这种类，必须采用 Thread 的形式，否则堵塞注了，连初始化都完成不了的。

5.4 监听器跟事件没有关系

所谓的监听器，无处不在啊。它代表着一种处理功能，按照这个意图来说，任何函数其实就是监听器的。监听器这个术语往往与事件监听器连在一起被人学习，被人传播，造成了一种不好的印象：一提到监听器，人人都觉得它跟事件有关系，其实它没有关系啊。

监听器就是函数，但是将函数改造成监听器需要以下几个步骤：
（1）监听接口的定义，监听具体执行类的定义
（2）参数封装成事件

普通的函数，以监听器的思想来重构一下，里面显得代码能力十分出众。

5.5 出口和入口绑定的端口都是监听端口

无论是出口还是入口，socket 绑定的端口，都是监听端口。这个有奇怪，反常识思维。

发送数据的时候，操作系统随机找个端口发送出去，并不是通过绑定的端口。

接受数据的时候，是通过绑定的端口，实现监听获取连接。

发送数据的时候，是随机挑选的端口，用不上绑定的端口，那为什么还需要绑定端口呢？

答案是：肯定是有用的，只可意会不可言传。

第六节：经常写技术文章的人未必代码境界高

在笔者看来，大多数喜欢写技术文章的人，水平都一般。这个结论的推导如下：

（1）人的精力是有限的，在这方面投入多了，在另外一方面必然少。

（2）写技术文章和写代码需要是思维方式是不一样的，甚至是矛盾的。一个人整天写些技术文章，必然导致思维方式已经不适应写代码，自然代码境界不会很高。

所以，笔者觉得：大多数喜欢写技术文章的人，水平都一般

笔者虽然也写一些技术文章，其实也适用上述道理。刚开始技术水平有限，经常写些技术文章。后来技术提升之后，再也无往年之热情了。

第七节：Java 的学习路线

写给工作年限为 1-3 年，立志在技术领域获得突破，立志想成为架构师的人。

世界是残酷的，IT 职场更是如此。如果在职场的头三年，还未明确发展方向，还没有获得小的成绩（当上小厂小主管，进入大厂拧螺丝等），以后就非常被动。不是说不可能再成功，而是成功的概率就会变的很低。我们都是平凡的人，只要概率很低事情，对我们来说已经不可能了，所以我们没有必要再自欺欺人。不要幻想着大器晚成，那个时候也许至亲的父母都未必还在，奋斗的价值已经大打折扣了。趁着当下大好时光，要不顾一切的往前冲，才无悔于年华，无悔于父母。

见：http://www.mybatis.cn/503.html

第八节：SQL 内功

SQL 是内功，但是大多数人还停留在招式层次。所谓招式，就是停留在简单的 select，create 等基本操作上，并没有深入的研究 SQL 背后的应用场景和技术思想。我感觉内功包括下面几个重点内容：

（1）表锁和行锁的理解和运用
（2）事务，事务隔离的深刻理解
（3）悲观锁，乐观锁的使用
（4）B/B+树的深刻认识
（5）索引的掌握
（7）分库分表的使用

如何掌握 SQL 内功呢，我个人走过的路分下面几个阶段：

第一阶段是大学阶段：学习数据库基本原理
第二阶段是校招的自学阶段：这个阶段大概有 2 个月，每天定时学一点，比上个阶段理解的更深一点，但是远远没有达到内功阶段。
第三阶段是工作中随用随百度阶段+收藏：这个阶段没有什么好说的，大家应该都是如此吧。

第四阶段是：系统化学习阶段。我自己写了一个小册子，大概有四五十小节的内容吧。这个阶段我才体会到了 sql 内功。上个阶段，提到了收藏，但是收藏属于看的层次，而这个层次在于写，其实差别非常大的，自然收获的差异也是非常大的。

第九节：同与不同

人与人是不同的，但是这种不同往往占很少的一部分而已，绝大部分都是相同的。所以，我们不应该过分的夸大不同。我们应该改变自己去适应正确的成才之路，而不是让正确的方法调整成错误的方法去迁就你。

第十节：老手与新人

绝大多数做技术的人都是实在人，但是也存在一少部分浮躁自负，甚至心理阴暗，充满戾气之徒。有的人工作了十多年了，跟他解释，他会说：我懒得听。跟他摆出证据，他会说：你这些都是书上的理论，不足信。跟他贴出某著名开源框架的源码，他就不说话了，连回复都不懒得回复的。总之，我觉得：无论是新人，还是老手，哪怕你工作了二十年，在技术方面心态要平和，骄傲和狂妄，不仅讨人嫌，最终坑的还是自己的技术生涯。
我碰见过很多刚刚入职的实习生，也碰见过很多工作了五六年的老手。新人因为技术水平欠缺、经验有限，对待新事物充满认可和好学的心态，这是新人最大的优势。那些工作了几年的职场老手，增加了辨别力，开始反思：是否适合自己。能想到适合与不适合，本身没有错误。但是，很多人却忘记了：人是活到老，学到老，需要不断的改变和纠正自己。不是主动改变，而是寻找合适，这也许是职场老手最大的劣势。新人有优势，老手有劣势，世界就是这么的残酷！
对于老手而言，最重要的一点是：需要不断地调整和改变，去适应客观规律，去靠近正确的发展方式。很多时候，答案也并不是丰富多彩的，很多时候，条条大路也不是全都能走向罗马的，否则的话，人人的学习方法都是正确的，人人都能考上名校，人人都能拿上大厂的 offer。

第十一节：对于术业有专攻的理解

术业有专攻弄，这个道理没错，但是看怎么个专法？以 socket 编程为例，把 Java 的 socket 学一下，把 netty 搞一遍，把 Linux 和 c++的 select 和 epoll 搞一下，把 python 的 tornado 搞一下，这才叫专攻。如果仅仅学习 Java 那么不叫专攻。举例来说，你使用汉语，你算精通法律吗？不能啊，你还需要研究英美的律法，这才叫专攻。生活中的道理，我们都知道，但是我们在计算机上却依然犯错栽跟头。

第十二节：什么是架构师?

我个人觉得：
Java + C + Shell + SQL + DataStructure + Coding >= 架构师
架构师需要具备数据库知识，所以 SQL 这块要精通。架构师不同于 DBA，DBA 侧重于数据库的运维，而架构师更侧重于 SQL 编程（锁，事务，索引）。架构师需要具备 Linux 知识，所以 Shell 这块要熟练。SQL 里面的各种树，Java 里面的各种高级类，背后都是 DataStructure 的知识。想成为一名架构师，Coding 一定得能力过人。我所说的 Coding 可不是工作中的增删改查，你要很多的小工具，小应用，这非常能锻炼人的。为什么呢？原因很简单，因为没有产品和老板催你上线，你有时间和精力能做出高质量的代码，所以能锻炼人。

第十三节：了解，理解，掌握

了解：进入知识空间，你摸了几块知识。

理解：进入知识空间，你能把每块知识都能看懂，知识之间的联系也能理清楚。

掌握：知识空间与人结合。

以一致性哈希举例来说，了解就是大概浏览了一下介绍页面，理解就是逐字读完并能读懂实现过程，掌握就是结合，类似这样的：建三峡大坝了，某县要淹没，人口要迁移到邻县。具体的细节自己可以想象出来。一定要学会把看似简单和平淡的生活场景改造搓揉成所学知识的原理。这是非常关键的学习能力。

第十四节：论认知偏差

当年校招的时候，我为了找个满意的工作，每天晚上都没有关过灯，都是和衣而眠，困了就睡一会儿，醒了就看书，狠狠的拼了一把，最终获得一个差不多满意的 offer。其实，人都是这样的：要想人前显贵，必定人后受罪。

在 IT 之路上，除了努力，我还认为，要有一个正确的认知，这样才能少走很多弯路。

优秀（挣钱多，升职快）=勤奋+好的认知。

说到认知，我经常发现很多人存在偏差。

1、关于技术交流和解决问题方面

我想提醒大家一点：仅仅解决工作中的问题，这种成长不叫成长，只有系统化的学习知识，由量变成质变，才是真正的成长。作为群主和站长，我接触最多的就是群里成员的问题，我希望大家能够认识到这点。

系统化学习，虽然是我强调的，但是我也不能总是做到。系统化学习很难做到，包括的方面很多。下面随便列举几条：
（1）比方说，学习基本语法，需要看多本书，这样各得其精华，而不是看一本书或者上网查找一下解疑即可。
（2）再比方说，看一本书，不是泛读，而是切入一点，搜索网上的资料，让这点成面，变得立体，形成一个小的科目。
（3）再比方说，各种笔记，敢于删除无关的，让有关的内容更内聚。
（4）通过解决工作中的疑问，这个不算系统化学习。除非你已经建立了知识体系，能纳入其中，内化成系统的一部分。

2、抓好面试题

抓好面试题并不是取巧，不踏实的行为。工程知识和面试知识还是有很大的区别的。我们往往在工程上花费了太多的时间，而在面试知识的准备上仍然投入不足。

抓好面试题可不是为了找到好的工作，可不是在面试之前就临时抱佛脚的事情，而是为了更好的学习知识。很多知识被包装了好多层，而面试题目才能抓住核心，这是非常重要的。

第十五节：时间换收入

以下内容出自阮一峰公众号。说的不错，站长深受启发，分享给大家看看。正文如下：

前几天，我听一个广播节目。主持人问，现在很多人开网约车，这样能赚多少钱，能够赚到大钱吗？
这个问题很容易回答，答案就是不能。出租车司机的收入，主要由营业时间的长短决定。基本上，一天开 12 个小时，就是比开 6 个小时，收入高出一倍。每天只有 24 个小时，因此收入存在上限，不可能偏离平均水平很远。
出租车是“时间换收入”的典型行业，投入的时间越多，收入越高，在家休息就没收入。很多行业都属于“时间换收入”，所有此类行业都赚不到大钱。因为你能用来交换的时间

是有限的，而且进入中年以后，你就拿不出更多的时间来交换。开出租车赚零花钱，或者作为短期过渡，这是没问题的，但作为终身职业是很糟糕的。

我觉得，越来越多的程序员正在落入这个陷井，用编码的时间换取收入。只有不停地做项目，才能拿到钱。项目做得越多，收入越高。这个项目开发完了，公司又让他去干下一个项目。 忙了好几年，项目完成了一大堆，但是自己什么也没留下，以后的收入还要取决于从零开始的新项目。这样的话，你跟出租车司机有何两样，哪一天你不写代码了，不是照样没收入。

那些赚到大钱的人，没有一个是靠时间换取收入的。他们要么通过积累资产致富，要么购买他人的时间，为自己创造财富。你应该警惕，不要落入“时间换取收入”的陷井，不要只顾着为别人生产代码，而要注意积累自己的资产，以及适时开展属于自己的业务。

积累财产，我目前写的小册子算一种方式；开展自己的业务，我目前做的 MyBatis 中文官网，也算是自己的业务。

第十六节：SQL 是最有价值的技能之一

1、推荐理由

这是一篇流传甚广的技术博文，出处是：
http://www.craigkerstiens.com/2019/02/12/sql-most-valuable-skill/，
推荐大家看看这篇文章，顺便提升英语能力。

站长对此文表示共鸣，并结合自身的学习情况，增加了一些感悟和总结。先看一下原文：

2、原文欣赏

我的职业生涯学到了很多技能，但没有比 SQL 更有用的技术技能。由于以下几个原因，SQL 对我来说是最有价值的技能。

（1）SQL 适用很多方面。

作为产品经理，你需要从数据库看数据。知道如何使用 SQL 查看原始数据，可以为你节省大量的精力，无需向其他人询问数字。

作为一名工程师，SQL 通常可以让我更快地获取我想要的信息，而不需要用 Ruby 或 Python 编写脚本。Web 应用变得缓慢时，了解所执行的 SQL 以及优化它的方法是不可或缺的。

（2）SQL 只需学习它一次，且不需要重新学习。

过去 20 年，SQL 并没有真正改变。当然，有一些新的改进，但是比起其他语言，它算是毫无变化。是的，每隔几年 SQL 会有一个新的标准，偶尔会出现一些新东西，但 SQL 的基础知识是非常永久的。学习 SQL 一次将允许您在职业生涯中重复使用它，而无需重新学习。

不要误会我的意思，我喜欢学习新的东西，但我宁愿学习一些真正新的东西，而不仅仅是另一种完成同样任务的方法。

（3）SQL 很酷。

熟练掌握 SQL 人并不多，大多数开发者跳过它，很少有人真正了解 SQL，所以掌握 SQL 的人可能看起来比实际更像精英。过去在一家拥有数百名工程师的公司中，我每周会收到多个同样的请求，来自从初级工程师到主要工程师各种人：“嘿，你能帮忙写一个查询吗？”因为你很擅长这样的事情，可以帮助其他人。

3、站长共鸣

本文引发站长共鸣，其实，学好 SQL 带来的受益远比上午描述的更多，以下是站长的现身感觉：

（1）学好 SQL，有助于理解排序，二分查找，B+树等数据结构和算法内容。

（2）学好 SQL，有助于理解和运用锁，例如行锁，表锁，悲观锁，乐观锁。最终实现内功的修炼，进而提升系统设计能力，可以说这是成为一名架构师的捷径。

（3）学好 SQL，掌握事务部分，有助于学习 Spring 框架的内容，这才是事务内容的根源所在。

（4）学好 SQL，掌握事务部分，进而延伸出分布式事务领域的内容，这是成为高级开发者的必由之路。