搜索引擎之倒排索引： 产品经理需要了解的搜索算法

互联网时代， 信息纷繁海量， 人们通过搜索引擎直达 “心中所想 ”已是常态。那么搜索引擎到底是如何高效查找目标内容呢？本文主要介绍搜索引擎里一个比较重要的结构 —— 倒排索引。

1 倒排索引简介

倒排索引（英文： Inverted Index ），是一种索引方法，常被用于全文检索系统中的一种单词文档映射结构。

现代搜索引擎绝大多数的索引都是基于倒排索引来进行构建的，这源于在实际应用当中，用户在使用搜索引擎查找信息时往往只输入信息中的某个属性关键字，如一些用户不记得歌名，会输入歌词来查找歌名；输入某个节目内容片

段来查找该节目等等。

面对海量的信息数据，为满足用户需求，顺应信息时代快速获取信息的趋势，聪明的开发者们在进行搜索引擎开发时对这些信息数据进行逆向运算，研发了“关键词——文档”形式的一种映射结构，实现了通过了物品属性信息对物品进行映射，可以帮助用户快速定位到目标信息，极大地降低了信息获取难度。倒排索引又叫反向索引，它是一种逆向思维运算，是现代信息检索领域里面最有效的一种索引结构。

2 倒排索引 &FAQ

从用户请求到结果返回，许多朋友会对倒排索引在检索系统中的工作过程产生好奇，本小节就倒排索引的一些常规认识，有如下问题：

Q1：何为索引？倒排索引又是什么？

索引，是为了加快信息查找过程，基于目标信息内容预先创建的一种储存结构。例如：一本书，没有目录，理论上也是可读的，只是当你合上当前在读的内容时，下次再翻开书本去查找，就比较耗费时间了。如果增加几页目录，我们可以快速地了解书本的大体内容分布，以及每一个章节页面位置的分布情况，这样我们查询内容的效率自然就会提高。书的目录，就是书本内容一种简单索引。

倒排索引，是索引技术中的一种，它是基于信息主体的关键属性值进行构建的。如下图 1：

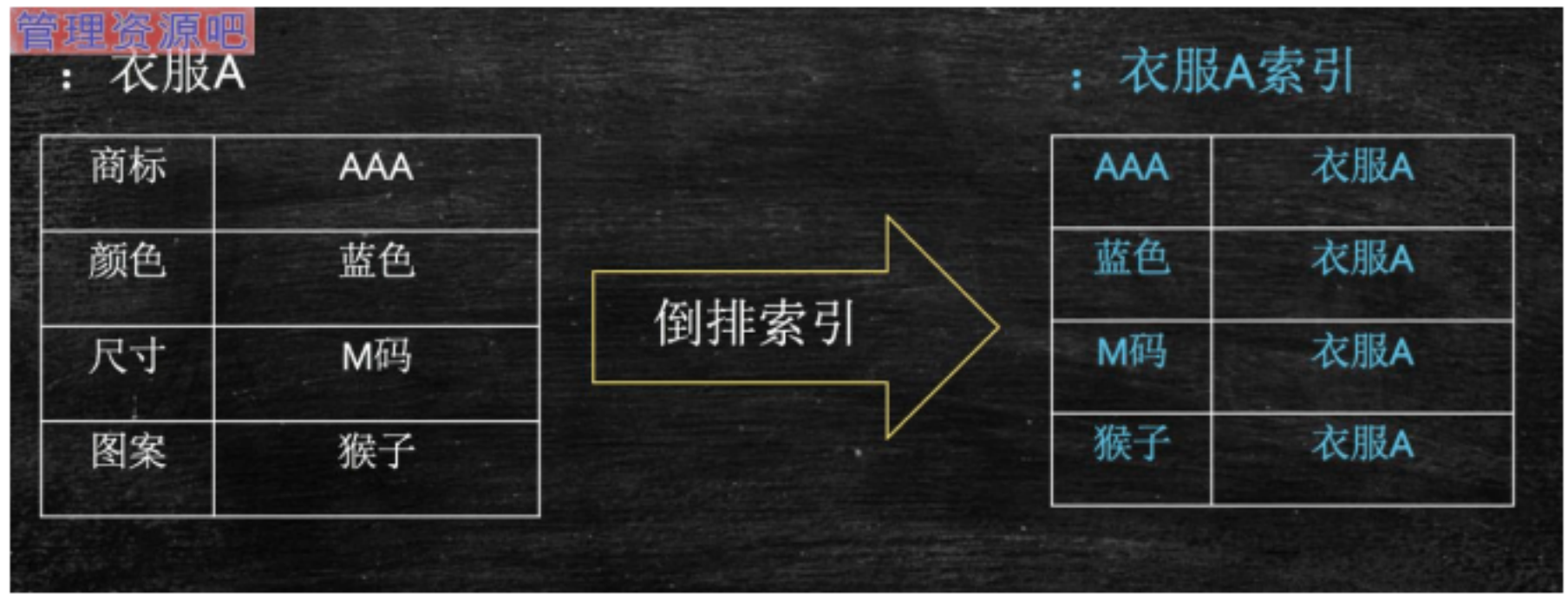

图 1 倒排索引概念示例图

假设检索系统中只有一个商品 —— 衣服 A，基于该商品构建其倒排索引结构之后，会产生上图右表中的索引结构，这样用户可以通过搜 “AAA”，“蓝色”，“M码”，“猴子”，均可找到该商品，加快了检索速度，扩大了检索范围。

Q2：当接受到用户查询请求时，倒排索引中发生了什么？

一般地，当接受到用户查询请求时，进入到倒排索引进行检索时，在返回结果的过程中，主要有以下几个步骤：

Step1：在分词系统对用户请求等原始 Query 进行分析，产生对应的 terms；

Step2：terms 在倒排索引中的词项列表中查找对应的 terms 的结果列表；

Step3：对结果列表数据进行微运算，如：计算文档静态分，文档相关性等；

Step4：基于上述运算得分对文档进行综合排序，最后返

回结果给用户。

上述该过程是较为简洁的一个检索过程。事实上，在生产环境中因为业务环境的繁杂，会使得索引的设计模式变得复杂且繁多。前文主要通过概念图来介绍倒排索引的架构体系，一个成熟的检索系统往往拥有一套较为稳定的算法体系，用于处理生产环境中的每一处细节技术需求。上述步骤中涉及了大量相关的数据储存技术、查找算法、排序算法、文本处理技术甚至 I/O 技术等等。

3 倒排索引技术剖析

构建倒排索引是搜索引擎里面至关重要的一个步骤，从技术层面去分析，对于构造一个倒排索引，主要分为两部分：

Doc2term 词项构造；

倒排记录表的构建。

3.1 term 词项构造

词项构造是在构建索引过程中必不可或缺的一个步骤，词项构造效果的好坏往往会直接影响到用户的搜索体验，以及搜索结果的召回。该过程主要是利用分词系统将文档中的各项属性的文本信息拆分成一些表意较强且重要的词汇，便于用户查找，如下图 2：

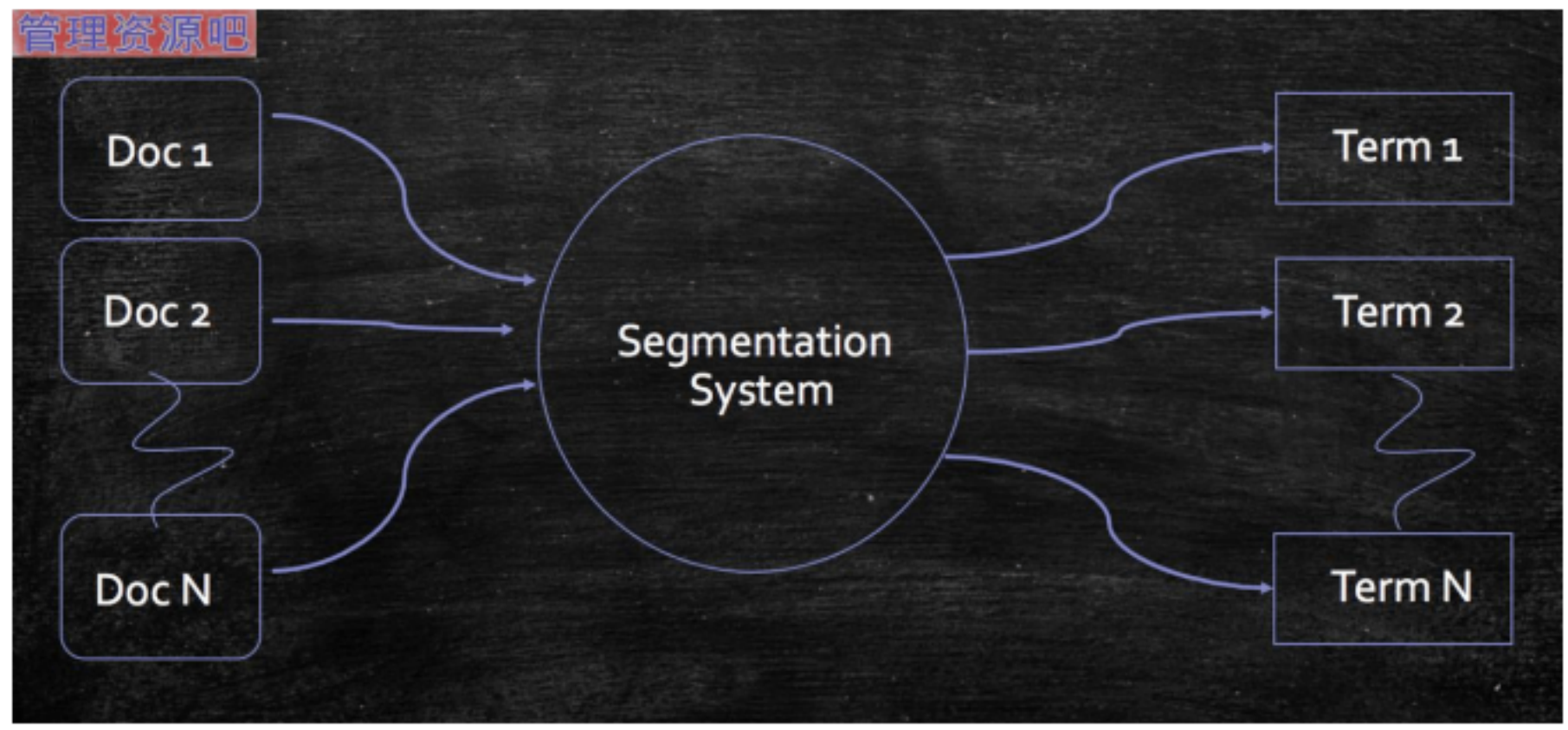

图 2 词项构造概念图

在词项构造的过程中，利用分词系统对文本进行处理时往往涉及到很多方面的问题，而且对于不同语种，会有不同的处理机制。下面主要介绍在处理文本时涉及到的几个问题：

（1）文本词条化

一段文本信息，它本身是一个由语言组成的字符串系列，本项技术点的主要任务是将一段连续的文本序列信息拆分成多个子序列。它与语言本身相关，面对不同的语言，处理文本的方式往往会不一样。对于中文，由于其语言多歧义且表意紧凑的特性，在实际应用中，一般需要借助 NLP 的相关技术对内容进行特征抽取，甚至人工标注等，生成对应的词典，随后再基于词典利用分词器进行分词，才能看到较好的文本词条效果。

而对于英文，普遍的英文句子，段落内容，它会以空格符作为单词之间的分隔符，所以一般情况下，以空格符对英

文内容进行拆分，已经可以取得比较好的效果，不过英文中也会存在一些特殊模式，如带上撇号的格式——“Teacher's office”，连字符格式——“English-speaking”也需要进行对应的处理，把单词提取出来。

（2）停用词过滤

停用词是指在文档列表中出现的频数较高且价值不大的词。以英文为例，在英文文档中出现次数较多的停用词如：“is”、“the”、“I”、“and”、“me”等等；这一类词语在往往出现在所有文档中，若以此类词语为 term 进行索引构建，则会产生多个全量文档索引列表。停用词过滤的使用往往依赖于实际使用场景，关键字查询使用得较为频繁的场景如某一个电商品牌的垂直型搜索引擎，一个合适的停用词表显得尤为重要；而对于 Web 搜索引擎如百度、Google 等，该类型的搜索引擎面向的查询场景较多，通用性较强，往往不需要停用词过滤。

（3）词条归一化

基于上述两点，将文档内容转换成一个或多个 term 后，在查询时，最理想的情况是用户输入的关键字刚好与 term 完全匹配，实际上，很多时候用户输入的 query 与词条之间往往不会完全匹配，而用户们还是希望 query 能与词条进行匹配，比如用户在查询“color”时，用户肯定也希望能看到关于“colour”的返回结果。词条归一化的任务就是将一些看起来不

完全一致的词条划分为一个等价类，比如英式单词 colour 和美式单词 color 归为一类、Air-conditioner 和 airconditioner 归为一类等等；这样，用户在查询时，只要对等价类中的任意单词进行搜索，都会返回包含等价类中的任意一个单词的文档。

（4）词干提取、词形还原

这是词条规范化的两种重要方式，用于扩展检索范围。词干提取的主要思想是“缩减”，将词条转化为词干，如：将“beaches”处理成“beach”，将“bananas”处理成“banana”等；词形还原的主要思想是“转换”，如：将“doing、”“done、”“did”转化成原型“do，”将“given、”“gave”转化成原型“give”等；词干提取的实现方法一般是基于规则对词条后缀进行缩减，至于词形还原，其实现方法需要词典来进行词形变化的映射；基于在此结合词条归一化技术，对扩展检索范围会产生一定的正向作用。

3.2 倒排记录表的构建

倒排记录表的构建过程面向的是海量的文档数据集合，在大小规模上它比词项集合要大得多，无法完全存放在内存当中，需要写入磁盘。因此，在构建倒排记录表时我们有必要为内存的使用作考虑。

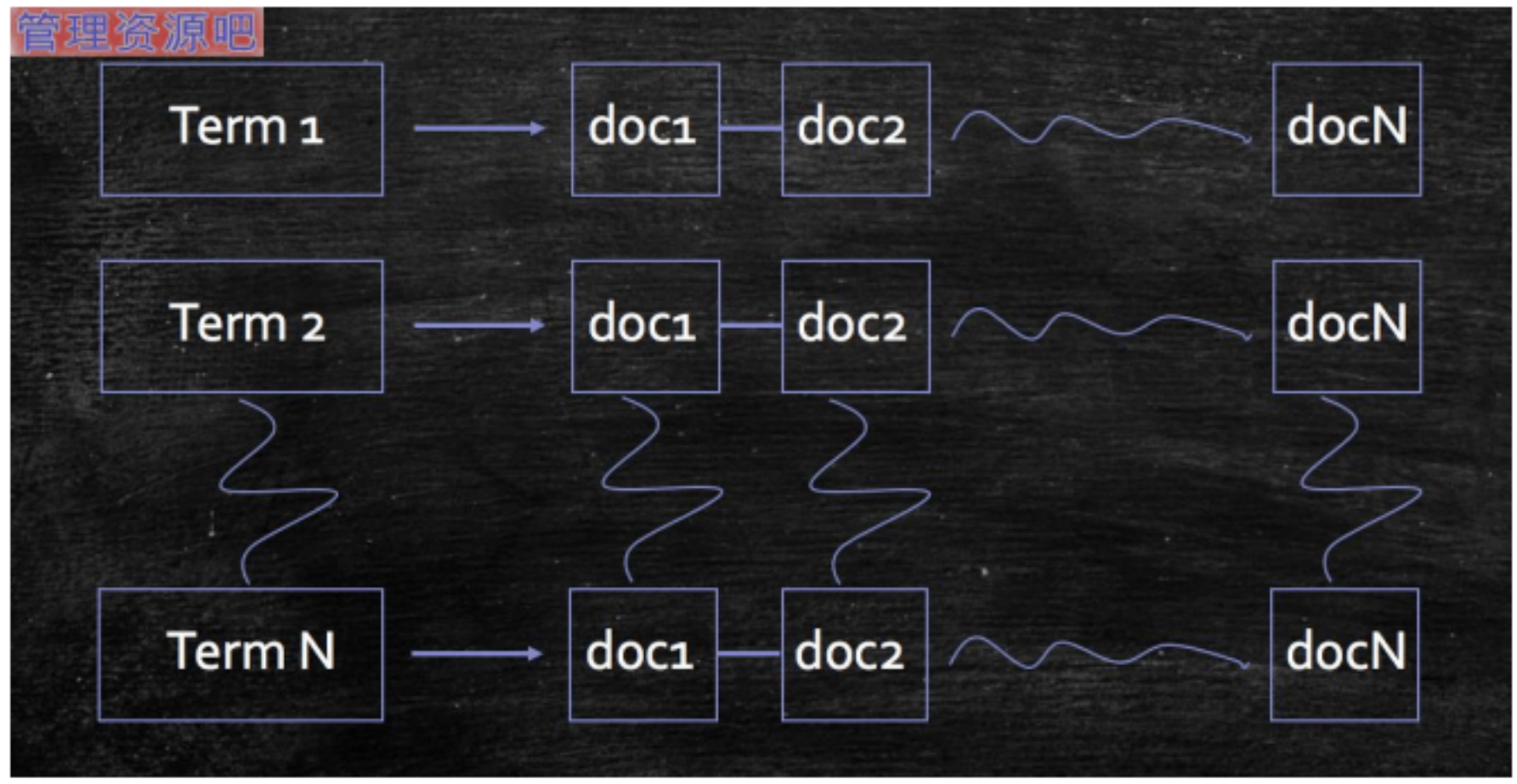

图 3 倒排索引概念图

在无法全内存的情况下，倒排记录表的主要构建思想是“分割”，亦即基于一定的处理逻辑对全量文档集合进行等份的批量处理。对于不同的业务需求，构建倒排记录表的方法往往会不一样。基本的构建方法如下：

S1: 通过一系列的处理将文档集合转化为“词项 ID — 文档 ID”对；

S2: 对词项 ID、文档 ID 进行排序，将具有相同词项对文档 ID 归并到该词项所对应的倒排记录表中，效果如图 3 所示；

S3: 将上述步骤产生的倒排索引写入磁盘，生成中间文件；

S4: 将上述所有的中间文件合并成最终的倒排索引。

从业务应用场景的角度出发，倒排记录表的构建方法主要有：单遍扫描和多遍扫描；从工程角度出发，倒排记录表

的构建方法主要有：分布式构建和动态构建。

3.2.1 单遍扫描构建

顾名思义，单遍扫描指的是仅对文档集合进行一次遍历，即可完成倒排索引的构建。由于内存开销问题，会将全量文档集进行分割，转换成几个内存大小相同的文档集合，然后依次执行前文中提及到的构建方法。该方法能快速构建一个简单可行的倒排索引，帮助用户通过关键字匹配快速找到目标文档。

3.2.2 多遍扫描构建

多遍扫描主要用于构建索引时获取关于文档的更多相关信息，如一些词项 TF-IDF 指标、词频、文档内容关系等，以丰富倒排记录表的内容，为搜索引擎进行功能扩充；在工业流水线上，单遍扫描构建索引由于其查询类型的丰富度不够，显然已经不能满足广大用户的需求了。搜索用户的需求并不止于关键字查询，像短语查询、模糊查询、精确筛选、模糊筛选、排序、聚合统计等等需求。这意味着我们在构建倒排列表时要尽可能获取文档的更多信息，便于查询时的微运算、重排序、相关性分析等技术需求。

3.2.3 分布式构建

对于一些大型搜索引擎如 Web 搜索引擎，单台机器已无法支撑其索引构建，需要多台机器组成集群对其进行分布式处理，将构建成的倒排索引进行分割，分布在多台机器上，

每台机器各自形成独立的索引结构，当用户发出请求时，会有多台机器响应，并且根据用户的搜索需求在各自的索引结构进行查询，返回相关结果，再将所有结果在内存中进行集中处理，最后把处理过的最优结果返回给用户。在具体的实现过程中，工程师们往往更钟情于一些通用的面向大规模机器计算的分布式架构如 Hadoop 中的 MapReduce、Java 中的 Fork/join 架构等，极大地提高了软件开发效率。

3.2.4 动态构建

该方法中的文档集合是变化的，这要求在对文档集进行索引构建时也要对文档的更新进行自适应。此问题常见于电商领域里，如商品的上下架、商品内容的更新等，都会引发索引的动态更新问题。于此，我们常采取一些策略型方法来解决该类型的问题，提高索引的实时性。常见的策略如下两种：

周期性对文档进行全量重建索引；

基于主索引的前提下，构建辅助索引，用于储存新文档，维护于内存中，当辅助索引达到一定的内存占用时，写入磁盘与主索引进行合并。

策略 1 是最简单直接、且有效的索引更新策略，对于数量级较大的搜索引擎来说处理简单便捷，由于动态索引计算的复杂性，使用其它策略会使得索引难维护，甚至引发严重的性能问题。所以大型搜索引擎往往更倾向于周期性重建索

引，不过这会涉及到索引热切换的问题，大量的文档经常会产生持续性的文档更新情况，这对于索引热切换时会造成一定的困难，处理不好会导致数据丢失，用户查不到新文档等问题。

策略 2 中在进行主辅索引合并时会遇到比较大的储存开销，由于文档量较大，这意味着在进行合并操作时会涉及到大量倒排文件的读写操作，要想将该过程高效化，目前能处理该问题的文件系统极其稀少，所以该策略在生产环境中往往可用性并不高。

4 总结

在实际生产环境中，由于业务的繁杂，倒排索引的技术体系会比本文所阐述的技术点要复杂得多。本文主要讲解了倒排索引的作用、索引构建方法、用户行为分析以及索引的应用场景，从整体出发，向大家介绍现代倒排索引大致的技术体系，帮助大家了解倒排索引的概念，了解搜索引擎。可能本文阐述的技术点、架构体系会因为笔者个人的理解偏差而存在一些不足或欠缺丰富。