

**要想搜索数据，先在数据库中存一份**

**索引**：对应数据库中的一个**库**

**类型**：对应一张**表（6.0之后废弃）**

**文档**：对应数据库表中的**一行**（通常**json**格式）（6.0规定为固定的名字\_doc）

**字段**：表中的**一列**（json格式中的每一个属性）

**6．0以后废弃了类型的概念，用索引来代替**

**节点**：**集群中的一台服务器**

**分片**：将**一个索引拆分成多个分片存储**，提高其**并发**能力

**副本**：**分片的备份**，提高可用性

在用搜索引擎搜索时，会**先**将搜索内容**分词**，将一句话拆成多个独立单词，把**搜索独立单词后的数据，合并后返回给客户端**，搜索分词插件通过**git**下载，对于新出的词要写到配置文件里，而且可以加一些扩展停止词，比如并,了, 的什么的，到停止词就结束

**向索引中添加数据**：提交数据用put请求，put后边的**url格式为服务器地址/索引名称/文档名称/数据id**，在下边的框中写请求体，将要添加的数据用json格式写入，如果用命令行来添加数据会很麻烦

**为什么不用数据库搜索**

数据库也可以提供搜索，但是他是**通过特定的列来查找，但用户在搜索时输入的不是这某个列所包含的**，输入的东西**可能包含多个列**，**要通过分词全文匹配**，这是数据库不能实现的，而搜索引擎可以

**面试题**

1. **为什么用ES**

因为是分布式的实时文件存储，**可以每个字段都存到索引让他之后能被检索到**

搜索引擎可以扩展到上百台服务器，处理PB(内存单位)级数据高度集成化的服务，

应用可以通过简单的 Restful API、各种语言的客户端、命令行交互

1. **ES读写数据、搜索数据、更新和删除数据**

写数据

1.客户端选择一个 node 发送请求过去，这个 node 就是 coordinating node（协调节点）。

2.coordinating node 对 document 进行路由，将请求转发给对应的 node（有 primary shard）。

3.实际的 node 上的 primary shard 处理请求，然后将数据同步到 replica node。

4.coordinating node 如果发现 primary node 和所有 replica node 都搞定之后，就返回响应结果给客户端。

底层：数据先写入内存 buffer，然后每隔 1s，将数据 refresh 到 os cache，到了 os cache 数据就能被搜索到（所以我们才说 es 从写入到能被搜索到，中间有 1s 的延迟）。每隔 5s，将数据写入 translog 文件（这样如果机器宕机，内存数据全没，最多会有 5s 的数据丢失），translog 大到一定程度，或者默认每隔 30mins，会触发 commit 操作，将缓冲区的数据都 flush 到 segment file 磁盘文件中。数据写到segment file后，同时建好了倒排索引

读数据

可以通过 doc id 来查询，会根据 doc id 进行 hash，判断出来当时把 doc id 分配到了哪个 shard 上面去，从那个 shard 去查询。

1.客户端发送请求到任意一个 node，成为 coordinate node。

2.coordinate node 对 doc id 进行哈希路由，将请求转发到对应的 node，此时会使用 round-robin 随机轮询算法，在 primary shard 以及其所有 replica 中随机选择一个，请求负载均衡。

3.接收请求的 node 返回 document 给 coordinate node。

4.coordinate node 返回 document 给客户端。

搜数据

1.客户端发送请求到一个 coordinate node。

2.协调节点将搜索请求转发到所有的 shard 对应的 primary shard 或 replica shard都可以。

3.query phase：每个 shard 将自己的搜索结果（其实就是一些 doc id）返回给协调节点，由协调节点进行数据的合并、排序、分页等操作，产出最终结果。

4.fetch phase：接着由协调节点根据 doc id 去各个节点上拉取实际的 document 数据，最终返回给客户端。

删数据--也是写操作

因为ES里文档是不可变的，所以删除操作commit 的时候会生成一个 .del 文件，里面将某个 doc 标识为 deleted 状态，那么搜索的时候根据 .del 文件就知道这个 doc 是否被删除了。

该文档依然能匹配查询，但是会在结果中被过滤掉。当段合并时，在.del 文件中被标记为删除的文档将不会被写入新段。

底层：细节：buffer 每 refresh 一次，就会产生一个 segment file，所以默认情况下是 1 秒钟一个 segment file，这样下来 segment file 会越来越多，此时会定期执行 merge。每次 merge 的时候，会将多个 segment file 合并成一个，同时这里会将标识为 deleted 的 doc 给物理删除掉，然后将新的 segment file 写入磁盘，这里会写一个 commit point，标识所有新的 segment file，然后打开 segment file 供搜索使用，同时删除旧的 segment file。

更新数据--也是写操作

将原来的 doc 标识为 deleted 状态，然后新写入一条数据。就是在新的文档被创建时，Elasticsearch 会给文档指定一个版本号，当执行更新时，旧版本的文档在.del 文件中被标记为删除，新版本的文档被索引到一个新段。旧版本的文档依然能匹配查询，但是会在结果中被过滤掉。

**3.底层Lucene**

lucene 是一个 jar 包，里面包含了封装好的各种建立倒排索引的算法代码。用 Java 开发的时候，引入 lucene jar然后基于 lucene的api就可以开发. 里面包括索引创建、索引、搜索

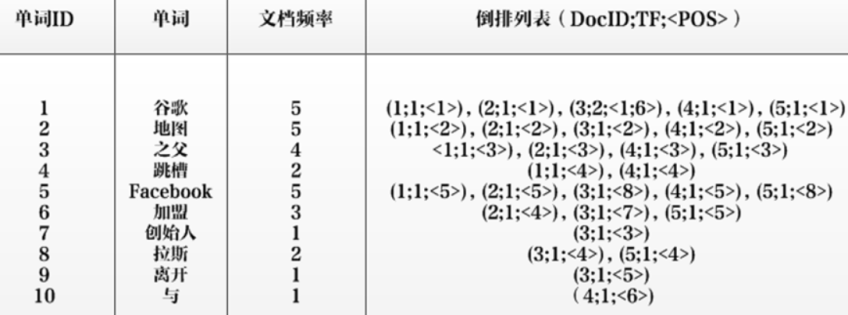
通过 lucene，我们可以将已有的数据建立索引，lucene 会在本地磁盘上面，给我们组织索引的数据结构。

**4.倒排索引**

搜索引擎中，每个文档都有一个对应的文档 ID，文档内容被表示为一系列关键词的集合。比如文档 1 经过分词，提取了 20 个关键词，每个关键词都会记录它在文档中出现的次数和出现位置。

倒排索引就是关键词到文档 ID 的映射，每个关键词都对应一系列的文件，这些文件中都出现了关键词

底层是FST(Finite State Transducer)数据结构,优点是1.空间占用小(对词典里单词前缀和后缀的重复利用，压缩了存储空间) 2.查询速度快 O(len(str))



单词“加盟”为例，其单词编号为6，文档频率为3，代表整个文档集合中有三个文档包含这个单词，对应的倒排列表为{(2;1;<4>),(3;1;<7>),(5;1;<5>)}，含义是在文档2，3，5出现过这个单词，在每个文档的出现过1次，单词“加盟”在第一个文档的POS是4，即文档的第四个单词是“加盟”

* 所有词项对应一个或多个文档；
* 倒排索引中的词项****根据字典顺序升序排列****
* 查docid用跳表

**5.索引多了怎么调优和部署**

1. 动态索引层面

基于模板+时间+rollover api 滚动创建索引，举例：设计阶段定义：blog 索引的模板格式为：blog\_index\_时间戳的形式，每天递增数据。

这样做的好处：不至于数据量激增导致单个索引数据量非常大，接近于上线 2 的32 次幂-1，索引存储达到了 TB+甚至更大。一旦单个索引很大，存储等各种风险也随之而来，所以要提前考虑+及早避免。

2. 存储层面

冷热数据分离存储，热数据（比如最近 3 天或者一周的数据），其余为冷数据。对于冷数据不会再写入新数据，可以考虑定期 force\_merge 加 shrink 压缩操作，节省存储空间和检索效率。

3. 部署层面

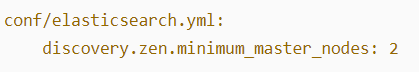
结合 ES 自身的支持动态扩展的特点，动态新增机器的方式可以缓解集群压力，注意：如果之前主节点等规划合理，不需要重启集群也能完成动态新增的。

1. **ES怎么实现master选举**

集群中可能会有多个master-eligible node，此时就要进行master选举，保证只有一个当选master。如果有多个node当选为master，则集群会出现脑裂，脑裂会破坏数据的一致性，导致集群行为不可控，产生各种非预期的影响。

为了避免产生脑裂，ES采用了常见的分布式系统思路，保证选举出的master被多数派(quorum)的master-eligible node认可，以此来保证只有一个master。这个

1.quorum通过以下配置进行配置：



2.先判定是否具备 master 资格，具备候选主节点资格的优先返回；

若两节点都为候选主节点，则 id 小的值会主节点。注意这里的 id 为 string 类型。

获取节点 id 的方法：

6ff6d45935007654dd1107625124c69

1. **ES集群**



1. **ZooKeeper介绍和用ZooKeeper实现**

