# Elasticsearch面试题



## 1、Elasticsearch是如何实现master选举的?

1、对所有可以成为master的节点根据nodeId排序，每次选举每个节点都把自己所知道节点排一次序，然后选出第一个（第0位）节点，暂且认为它是master节点。
2、如果对某个节点的投票数达到一定的值（可以成为master节点数n/2+1）并且该节点自己也选举自己，那这个节点就是master。否则重新选举。
3、对于brain split问题，需要把候选master节点最小值设置为可以成为master节点数n/2+1（quorum）

## 2、详细描述一下 Elasticsearch 索引文档的过程。

1、当分片所在的节点接收到来自协调节点的请求后，会将请求写入到 MemoryBuffer，然后定时（默认是每隔 1 秒）写入到 Filesystem Cache，这个从 MomeryBuffer 到 Filesystem Cache 的过程就叫做 refresh；
2、当然在某些情况下，存在 Momery Buffer 和 Filesystem Cache 的数据可能会丢失，ES 是通过 translog 的机制来保证数据的可靠性的。其实现机制是接收到请求后，同时也会写入到 translog 中，当 Filesystem cache 中的数据写入到磁盘中时，才会清除掉，这个过程叫做 flush；
3、在 flush 过程中，内存中的缓冲将被清除，内容被写入一个新段，段的 fsync将创建一个新的提交点，并将内容刷新到磁盘，旧的 translog 将被删除并开始一个新的 translog。
4、flush 触发的时机是定时触发（默认 30 分钟）或者 translog 变得太大（默认为 512M）时；

## 3、详细描述一下 Elasticsearch 更新和删除文档的过程。

1、删除和更新也都是写操作，但是 Elasticsearch 中的文档是不可变的，因此不能被删除或者改动以展示其变更。
2、磁盘上的每个段都有一个相应的.del 文件。当删除请求发送后，文档并没有真的被删除，而是在.del 文件中被标记为删除。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。
3、在新的文档被创建时，Elasticsearch 会为该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。

## 4、详细描述一下 Elasticsearch 搜索的过程?

1、搜索被执行成一个两阶段过程，我们称之为 Query Then Fetch；
2、在初始查询阶段时，查询会广播到索引中每一个分片拷贝（主分片或者副本分片）。 每个分片在本地执行搜索并构建一个匹配文档的大小为 from + size 的优先队列。
备注：在搜索的时候是会查询 Filesystem Cache 的，但是有部分数据还在 MemoryBuffer，所以搜索是近实时的。
3、每个分片返回各自优先队列中 所有文档的 ID 和排序值 给协调节点，它合并这些值到自己的优先队列中来产生一个全局排序后的结果列表。
4、接下来就是 取回阶段，协调节点辨别出哪些文档需要被取回并向相关的分片提交多个 GET 请求。每个分片加载并 丰富 文档，如果有需要的话，接着返回文档给协调节点。一旦所有的文档都被取回了，协调节点返回结果给客户端。
5、补充：Query Then Fetch 的搜索类型在文档相关性打分的时候参考的是本分片的数据，这样在文档数量较少的时候可能不够准确，DFS Query Then Fetch 增加了一个预查询的处理，询问 Term 和 Document frequency，这个评分更准确，但是性能会变差。

## 5、Elasticsearch 对于大数据量（上亿量级）的聚合如何实现?

Elasticsearch 提供的首个近似聚合是 cardinality 度量。它提供一个字段的基数，即该字段的 distinct 或者unique 值的数目。它是基于 HLL 算法的。HLL 会先对我们的输入作哈希运算，然后根据哈希运算的结果中的 bits 做概率估算从而得到基数。其特点是：可配置的精度，用来控制内存的使用（更精确 ＝ 更多内存）；小的数据集精度是非常高的；我们可以通过配置参数，来设置去重需要的固定内存使用量。无论数千还是数十亿的唯一值，内存使用量只与你配置的精确度相关。

## 6、在并发情况下，Elasticsearch 如果保证读写一致?

1、可以通过版本号使用乐观并发控制，以确保新版本不会被旧版本覆盖，由应用层来处理具体的冲突；
2、另外对于写操作，一致性级别支持 quorum/one/all，默认为 quorum，即只有当大多数分片可用时才允许写操作。但即使大多数可用，也可能存在因为网络等原因导致写入副本失败，这样该副本被认为故障，分片将会在一个不同的节点上重建。
3、对于读操作，可以设置 replication 为 sync(默认)，这使得操作在主分片和副本分片都完成后才会返回；如果设置 replication 为 async 时，也可以通过设置搜索请求参数_preference 为 primary 来查询主分片，确保文档是最新版本。

## 7、ElasticSearch中的集群、节点、索引、文档、类型是什么?

群集：一个或多个节点（服务器）的集合，它们共同保存您的整个数据，并提供跨所有节点的联合索引和搜索功能。群集由唯一名称标识，默认情况下为“elasticsearch”。此名称很重要，因为如果节点设置为按名称加入群集，则该节点只能是群集的一部分。

节点：属于集群一部分的单个服务器。它存储数据并参与群集索引和搜索功能。

索引：就像关系数据库中的“数据库”。它有一个定义多种类型的映射。索引是逻辑名称空间，映射到一个或多个主分片，并且可以有零个或多个副本分片。
eg: MySQL =>数据库　　ElasticSearch =>索引

文档：类似于关系数据库中的一行。不同之处在于索引中的每个文档可以具有不同的结构（字段），但是对于通用字段应该具有相同的数据类型。
MySQL => Databases =>　Tables => Columns / Rows ElasticSearch => Indices => Types =>具有属性的文档

类型：是索引的逻辑类别/分区，其语义完全取决于用户。

# 8、Elasticsearch的倒排索引是什么?

1、倒排索引是搜索引擎的核心。搜索引擎的主要目标是在查找发生搜索条件的文档时提供快速搜索。倒排索引是一种像数据结构一样的散列图，可将用户从单词导向文档或网页。它是搜索引擎的核心。其主要目标是快速搜索从数百万文件中查找数据。
2、传统的我们的检索是通过文章，逐个遍历找到对应关键词的位置。而倒排索引，是通过分词策略，形成了词和文章的映射关系表，这种词典+映射表即为倒排索引。有了倒排索引，就能实现o（1）时间复杂度的效率检索文章了，极大的提高了检索效率。

学术的解答方式：
倒排索引，相反于一篇文章包含了哪些词，它从词出发，记载了这个词在哪些文档中出现过，由两部分组成——词典和倒排表。
加分项：倒排索引的底层实现是基于：FST（Finite State Transducer）数据结构。
lucene从4+版本后开始大量使用的数据结构是FST。FST有两个优点：
1）空间占用小。通过对词典中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))的查询时间复杂度。

# 9、ElasticSearch中的分析器是什么?

1、在ElasticSearch中索引数据时，数据由为索引定义的Analyzer在内部进行转换。 分析器由一个Tokenizer和零个或多个TokenFilter组成。编译器可以在一个或多个CharFilter之前。分析模块允许您在逻辑名称下注册分析器，然后可以在映射定义或某些API中引用它们。
2、Elasticsearch附带了许多可以随时使用的预建分析器。或者，您可以组合内置的字符过滤器，编译器和过滤器器来创建自定义分析器。

# 10、启用属性，索引和存储的用途是什么?

1、Enabled属性适用于各类ElasticSearch特定/创建领域，如index和size。用户提供的字段没有“已启用”属性。 存储意味着数据由Lucene存储，如果询问，将返回这些数据。
2、存储字段不一定是可搜索的。默认情况下，字段不存储，但源文件是完整的。因为您希望使用默认值(这是有意义的)，所以不要设置store属性 该指数属性用于搜索。
3、索引属性只能用于搜索。只有索引域可以进行搜索。差异的原因是在分析期间对索引字段进行了转换，因此如果需要的话，您不能检索原始数据。

# 11、Elasticsearch了解多少，说说你们公司es的集群架构，索引数据大小，分片有多少，以及一些调优手段 。

比如：ES集群架构13个节点，索引根据通道不同共20+索引，根据日期，每日递增20+，索引：10分片，每日递增1亿+数据，每个通道每天索引大小控制：150GB之内。
仅索引层面调优手段：

1.1、设计阶段调优
1）根据业务增量需求，采取基于日期模板创建索引，通过roll over API滚动索引；
2）使用别名进行索引管理；
3）每天凌晨定时对索引做force_merge操作，以释放空间；
4）采取冷热分离机制，热数据存储到SSD，提高检索效率；冷数据定期进行shrink操作，以缩减存储；
5）采取curator进行索引的生命周期管理；
6）仅针对需要分词的字段，合理的设置分词器；
7）Mapping阶段充分结合各个字段的属性，是否需要检索、是否需要存储等。 …

1.2、写入调优
1）写入前副本数设置为0；
2）写入前关闭refresh_interval设置为-1，禁用刷新机制；
3）写入过程中：采取bulk批量写入；
4）写入后恢复副本数和刷新间隔；
5）尽量使用自动生成的id。

1.3、查询调优
1）禁用wildcard；
2）禁用批量terms（成百上千的场景）；
3）充分利用倒排索引机制，能keyword类型尽量keyword；
4）数据量大时候，可以先基于时间敲定索引再检索；
5）设置合理的路由机制。

1.4、其他调优
部署调优，业务调优等。

# 12、Elasticsearch 索引数据多了怎么办，如何调优，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
一旦之前没有规划，这里就属于应急策略。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。

# 13、在使用 Elasticsearch 时要注意什么?

> 由于ES使用的Java写的，所有注意的是GC方面的问题

1、倒排词典的索引需要常驻内存，无法 GC，需要监控 data node 上 segmentmemory 增长趋势。
2、各类缓存，field cache，filter cache，indexing cache，bulk queue 等等，要设置合理的大小，并且要应该根据最坏的情况来看 heap 是否够用，也就是各类缓存全部占满的时候，还有 heap 空间可以分配给其他任务吗？避免采用 clear cache等“自欺欺人”的方式来释放内存。
3、避免返回大量结果集的搜索与聚合。确实需要大量拉取数据的场景，可以采用scan & scroll api 来实现。
4、cluster stats 驻留内存并无法水平扩展，超大规模集群可以考虑分拆成多个集群通过 tribe node 连接
5、想知道 heap 够不够，必须结合实际应用场景，并对集群的 heap 使用情况做持续的监控。

## 14、Elasticsearch 支持哪些类型的查询?

查询主要分为两种类型：精确匹配、全文检索匹配。
精确匹配，例如 term、exists、term set、 range、prefix、 ids、 wildcard、regexp、fuzzy等。
全文检索，例如match、match_phrase、multi_match、match_phrase_prefix、query_string 等

## 15、你能否列出与 Elasticsearch 有关的主要可用字段数据类型?

1、字符串数据类型，包括支持全文检索的 text 类型 和 精准匹配的 keyword 类型。
2、数值数据类型，例如字节，短整数，长整数，浮点数，双精度数，half_float，scaled_float。
3、日期类型，日期纳秒Date nanoseconds，布尔值，二进制（Base64编码的字符串）等。
4、范围（整数范围 integer_range，长范围 long_range，双精度范围 double_range，浮动范围 float_range，日期范围 date_range）。
5、包含对象的复杂数据类型，nested 、Object。
6、GEO 地理位置相关类型。
7、特定类型如：数组（数组中的值应具有相同的数据类型）

## 16、如何监控 Elasticsearch 集群状态?

Marvel 让你可以很简单的通过 Kibana 监控 Elasticsearch。你可以实时查看你的集群健康状态和性能，也可以分析过去的集群、索引和节点指标。

## 17、有了解过Elasticsearch的性化搜索方案吗?

基于word2vec和Elasticsearch实现个性化搜索
（1）基于word2vec、Elasticsearch和自定义的脚本插件，我们就实现了一个个性化的搜索服务，相对于原有的实现，新版的点击率和转化率都有大幅的提升；
（2）基于word2vec的商品向量还有一个可用之处，就是可以用来实现相似商品的推荐；
（3）使用word2vec来实现个性化搜索或个性化推荐是有一定局限性的，因为它只能处理用户点击历史这样的时序数据，而无法全面的去考虑用户偏好，这个还是有很大的改进和提升的空间；

## 18、是否了解字典树?

| 数据结构 | 优缺点 |
| --- | --- |
| Array/List | 使用二分法查找，不平衡 |
| HashMap/TreeMap | 性能高，内存消耗大，几乎是原始数据的三倍 |
| Skip List | 跳跃表，可快速查找词语，在lucene,redis,HBase中有实现 |
| Trie | 适合英文词典，如果系统中存在大量字符串且这些字符串基本没有公共前缀 |
| Double Array Trie | 适合做中文词典，内存占用小，很多分词工具军采用此种算法 |
| Ternary Search Tree | 一种有状态的转移机，Lucene 4有开源实现，并大量使用 |

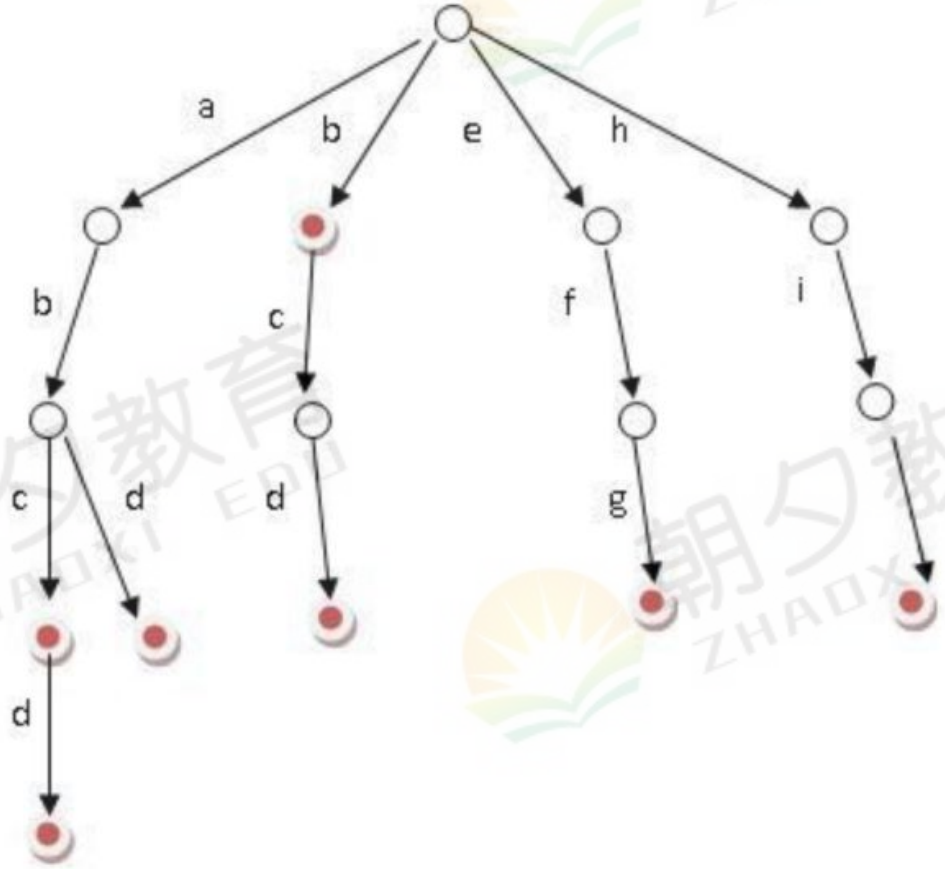
Trie 的核心思想是空间换时间，利用字符串的公共前缀来降低查询时间的开销以

**达到提高效率的目的。它有 3 个基本性质：**

1、根节点不包含字符，除根节点外每一个节点都只包含一个字符。

2、从根节点到某一节点，路径上经过的字符连接起来，为该节点对应的字符串。

3、每个节点的所有子节点包含的字符都不相同。




1、可以看到，trie 树每一层的节点数是 26^i 级别的。所以为了节省空间，我们
还可以用动态链表，或者用数组来模拟动态。而空间的花费，不会超过单词数×单
词长度。
2、实现：对每个结点开一个字母集大小的数组，每个结点挂一个链表，使用左儿子右兄弟表示法记录这棵树；
3、对于中文的字典树，每个节点的子节点用一个哈希表存储，这样就不用浪费太大的空间，而且查询速度上可以保留哈希的复杂度 O(1)。

## 19、ElasticSearch是否有架构?

1、ElasticSearch可以有一个架构。架构是描述文档类型以及如何处理文档的不同字段的一个或多个字段的描述。Elasticsearch中的架构是一种映射，它描述了JSON文档中的字段及其数据类型，以及它们应该如何在Lucene索引中进行索引。因此，在Elasticsearch术语中，我们通常将此模式称为“映射”。
2、Elasticsearch具有架构灵活的能力，这意味着可以在不明确提供架构的情况下索引文档。如果未指定映射，则默认情况下，Elasticsearch会在索引期间检测文档中的新字段时动态生成一个映射。

## 20、为什么要使用Elasticsearch?

因为在我们商城中的数据，将来会非常多，所以采用以往的模糊查询，模糊查询前置配置，会放弃索引，导致商品查询是全表扫面，在百万级别的数据库中，效率非常低下，而我们使用ES做一个全文索引，我们将经常查询的商品的某些字段，比如说商品名，描述、价格还有id这些字段我们放入我们索引库里，可以提高查询速度。