Elasticsearch中的倒排索引是什么?

- 正排索引-文档 Id 到文档内容和单词的关联
- 倒排索引-单词到文档 Id 的关系
- 底层的数据结构是FST，位图

Elasticsearch是如何实现Master选举的

- Elasticsearch的选主是ZenDiscovery模块负责的，主要包含Ping（节点之间通过这个RPC来发现彼此）和Unicast（单播模块包含一个主机列表以控制哪些节点需要ping通）这两部分；
- 对所有可以成为master的节点（node.master: true）根据nodeId字典排序，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。
- 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。否则重新选举一直到满足上述条件。

Elasticsearch中的节点（比如共20个），其中的10个选了一个master，另外10个选了另一个master，怎么办?

- 当集群master候选数量不小于3个时，可以通过设置最少投票通过数量（discovery.zen.minimum_master_nodes）超过所有候选节点一半以上来解决脑裂问题；
- 当候选数量为两个时，只能修改为唯一的一个master候选，其他作为data节点，避免脑裂问题。

Elasticsearch集群脑裂问题

- 减少误判：discovery.zen.ping_timeout节点状态的响应时间，默认为3s，可以适当调大
- 选举触发：discovery_zen.minimum_master_nodes:该参数用于控制选举行为发生的最小集群主节点数量。 设置（n/2）+1
- 角色分离：即master节点与data节点分离
    - 主节点配置为：node.master:true node.data:false
    - 从节点配置为：node.master:false node.data:true
- 在最新版7.x中， minimum_master_node 这个参数已经被移除了，这一块内容完全由es自身去管理，这样就避免了脑裂的问题，选举也会非常快

详细描述一下Elasticsearch索引文档的过程。

shard = hash(document_id) % (num_of_primary_shards) 进入协调节点
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Memory Buffer，然后定时（默认是每隔1秒）写入到Filesystem Cache，这个从Momery Buffer到Filesystem Cache的过程就叫做refresh；

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；

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。

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；

详细描述一下Elasticsearch更新和删除文档的过程

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；

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。

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。

详细描述一下Elasticsearch搜索的过程

搜索被执行成一个两阶段过程，我们称之为 Query Then Fetch；

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。PS：在搜索的时候是会查询Filesystem Cache的，但是有部分数据还在Memory Buffer，所以搜索是近实时的。

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。

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。

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。

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 .

在并发情况下，Elasticsearch如果保证读写一致?

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；

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。

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。

ElasticSearch中的分析器是什么?

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。

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。

什么是ElasticSearch中的编译器?

编译器用于将字符串分解为术语或标记流。一个简单的编译器可能会将字符串拆分为任何遇到空格或标点的地方。Elasticsearch有许多内置标记器，可用于构建自定义分析器。

启用属性，索引和存储的用途是什么?

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。
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。
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。

深度分页解决方案

- 限制分页数量 100页
- 调整分片最大文档数1万为更高

滚动查询

- 滚动id
- 有效时间