# 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。

# Elasticsearch是如何实现Master选举的？

Elasticsearch的选举是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。  
补充：master节点的职责主要包括集群、节点和索引的管理，不负责文档级别的管理；data节点可以关闭http功能。

# 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。

# 详细描述一下Elasticsearch索引文档的过程?

协调节点默认使用文档ID参与计算（也支持通过routing），以便为路由提供合适的分片。  
　　shard = hash(document\_id) % (num\_of\_primary\_shards)  
　　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;  
　　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;  
　　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中的集群、节点、索引、文档、类型是什么？

群集是一个或多个节点（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。  
　　节点是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。 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。

# ElasticSearch中的分片是什么?

在大多数环境中，每个节点都在单独的盒子或虚拟机上运行。

索引 - 在Elasticsearch中，索引是文档的集合。  
　　分片 - 因为Elasticsearch是一个分布式搜索引擎，所以索引通常被分割成分布在多个节点上的被称为分片的元素。

# 11.elasticsearch写入数据流程?

**elasticsearch写入数据时涉及到的核心概念讲解：**

**倒排索引:**即把文件ID对应到关键词的映射转换为关键词到文件ID的映射，每个关键词都对应着一系列的文件，这些文件中都出现这个关键词。

       得到倒排索引的结构如下：

       “关键词1”：“文档1”的ID，“文档2”的ID，…………。

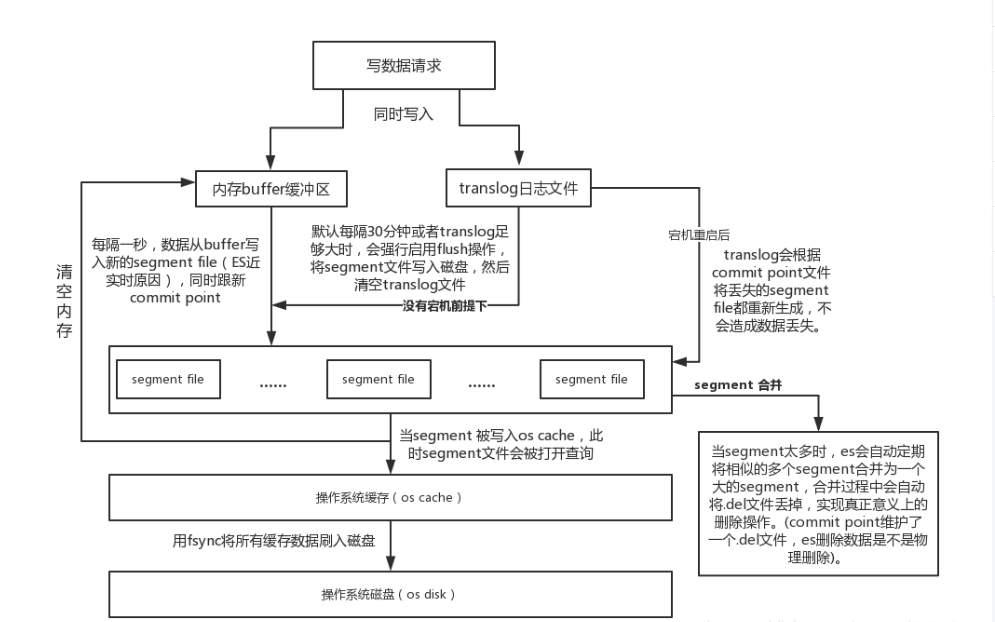
       “关键词2”：带有此关键词的文档ID列表。

**segment file:** 存储倒排索引的文件，每个segment本质上就是一个倒排索引，每秒都会生成一个segment文件，当文件过多时es会自动进行segment merge（合并文件），合并时会同时将已经标注删除的文档物理删除；

**commit point（重点理解）:** 记录当前所有可用的segment，每个commit point都会维护一个.del文件（es删除数据本质是不属于物理删除），当es做删改操作时首先会在.del文件中声明某个document已经被删除，文件内记录了在某个segment内某个文档已经被删除，当查询请求过来时在segment中被删除的文件是能够查出来的，但是当返回结果时会根据commit point维护的那个.del文件把已经删除的文档过滤掉；

**translog日志文件:** 为了防止elasticsearch宕机造成数据丢失保证可靠存储，es会将每次写入数据同时写到translog日志中(图中会有详解)。

**完整elasticsearch的写入数据流程如下：**



补充：将es中比较困惑的几个概念简单总结一下，这三种操作对理解es底层原理和优化很有帮助！

**refresh**

es接收数据请求时先存入内存中，默认每隔一秒会从内存buffer中将数据写入filesystem cache，这个过程叫做refresh；

**fsync**

translog会每隔5秒或者在一个变更请求完成之后执行一次fsync操作，将translog从缓存刷入磁盘，这个操作比较耗时，如果对数据一致性要求不是跟高时建议将索引改为异步，如果节点宕机时会有5秒数据丢失;

**flush**

es默认每隔30分钟会将filesystem cache中的数据刷入磁盘同时清空translog日志文件，这个过程叫做flush。