在 Elasticsearch 中，搜索一般包括两个阶段，query 和 fetch 阶段，可以简单的理解，query 阶段确定要取哪些doc，fetch 阶段取出具体的 doc. 简单来说就是query阶段只返回了doc id, 结果量小, 而fetch阶段返回详细的结果数据量大. query 阶段知道了要取哪些数据，但是并没有取具体的数据，这就是 fetch 阶段要做的

ES的搜索是分2个阶段进行的，即Query阶段和Fetch阶段。  Query阶段比较轻量级，通过查询倒排索引，获取满足查询结果的文档ID列表。  而Fetch阶段比较重，需要将每个shard的结果取回，在协调结点进行**全局**排序。  通过From+size这种方式分批获取数据的时候，随着from加大，需要全局排序并丢弃的结果数量随之上升，性能越来越差。  
   
而Scroll查询，先做轻量级的Query阶段以后，免去了繁重的全局排序过程。 它只是将查询结果集，也就是doc id列表保留在一个上下文里， 之后每次分批取回的时候，只需根据设置的size，在每个shard内部按照一定顺序（默认doc\_id续)， 取回这个size数量的文档即可。   
   
由此也可以看出scroll不适合支持那种实时的和用户交互的前端分页工作，其主要用途用于从ES集群分批拉取大量结果集的情况，一般都是offline的应用场景。  比如需要将非常大的结果集拉取出来，存放到其他系统处理，或者需要做大索引的reindex等等。

Query阶段并不需要fetch真实的数据，只需要拿到满足条件的doc id列表。 即使有100万个doc id的列表存放并不需要耗费多少内存，因为ES使用了非常紧凑的数据结构和压缩算法来存放这些ID。 所以Query阶段是轻量级的， 真正耗费内存是在fetch阶段的全局排序。 仔细阅读以上上面的文档，有讲到全局排序的原理以及为什么深度分页时，开销越来越高。

scroll 并不适合用来做实时搜索，而更适用于后台批处理任务，比如群发

深度分页的代价根源是结果集全局排序