**Elasticsearch的个人理解和总结**

ElaticSearch，简称为ES， ES是一个开源的高扩展的分布式全文检索引擎，它可以近乎实时的存储、检索数据；本身扩展性很好，可以扩展到上百台服务器，处理PB级别的数据。ES也使用Java开发并使用Lucene作为其核心来实现所有索引和搜索的功能，但是它的目的是通过简单的Restful Api来隐藏Lucene的复杂性，从而让全文搜索变得简单。

## **ES核心概念：**

(1) cluster（集群）：每个集群至少包含两个节点.

(2) node：集群中的每个节点，一个节点不代表一台服务器

(3) field：一个数据字段，与index和type一起，可以定位一个doc

(4) document：ES最小的数据单元 Json

(5)Field: 定义每个document应该有的字段。

### (6) Index：索引 包含一堆有相似结构的文档数据

### (7) shard：分片

index数据过大时，将index里面的数据，分为多个shard，分布式的存储在各个服务器上面。可以支持海量数据和高并发，提升性能和吞吐量，充分利用多台机器的cpu。

### (8) replica：副本

在分布式环境下，任何一台机器都会随时宕机，如果宕机， index的一个分片没有，导致此index不能搜索。所以，为了保证数据的安全，我们会将每个index的分片经行备份，存储在另外的机器上。保证少数机器宕机es集群仍可以搜索。  
能正常提供查询和插入的分片我们叫做主分片（primary shard），其余的我们就管他们叫做备份的分片（replica shard）。  
es6默认新建索引时， 5分片， 2副本，也就是一主一备，共10个分片。所以， es集群最小规模为两台。

**ElasticSearch与SpringBoot的整合**

导入依赖：

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-elasticsearch</artifactId>

</dependency>

进行配置：

data:

elasticsearch:

repositories:

enabled: true

cluster-nodes: 120.56.195.135:9300 # es的连接地址及端口号

cluster-name: elastic # es集群的名称

创建一个查询实体：

@Data

@Builder

@Document(indexName = "book", shards = 3, replicas = 0)

public class Book {

@Id

private Long id;

@Field(name = "author\_name", type = FieldType.Keyword, store = true)

private String authorName;

//存储分词为ik\_max\_word, 查询分词为ik\_smart

@Field(name = "book\_name", store = true, analyzer = "ik\_max\_word", searchAnalyzer = "ik\_smart")

private String bookName;

@Field(name = "desc", store = true, analyzer = "ik\_max\_word", searchAnalyzer = "ik\_smart")

private String desc;

@Field(name = "pub\_date", type = FieldType.Text, format = DateFormat.custom, pattern = "yyyy-MM-dd HH:mm:ss", store = true)

private Date pubDate;

@Override

public String toString() {

return "Book{" +

"id=" + id +

", authorName='" + authorName + '\'' +

", bookName='" + bookName + '\'' +

", desc='" + desc + '\'' +

", pubDate=" + pubDate +

'}';

}

}

**索引操作：**

public class Index {

@Autowired

private ElasticsearchRestTemplate restTemplate;

/\*\*

\* 创建索引

\*/

@Test

public void testCreateIndex() {

boolean isCreated = restTemplate.indexOps(Book.class).create();

log.info("test create index : {}", isCreated);

}

/\*\*

\* 查询索引是否存在

\*/

@Test

public void testExistsIndex() {

boolean isExists = restTemplate.indexOps(Book.class).exists();

log.info("test exists index : {}", isExists);

}

/\*\*

\* 删除索引

\*/

@Test

public void testDeleteIndex() {

boolean isDeleted = restTemplate.indexOps(Book.class).delete();

log.info("test delete index : {}", isDeleted);

}

}

**新增文档：**

public class Document {

@Autowired

private ElasticsearchRestTemplate restTemplate;

/\*\*

\* 新增 or 全量更新

\*/

@Test

public void testAdd() {

Book book = Book.builder()

.id(1L)

.authorName("lin")

.bookName("ElasticSearch")

.desc("学编程使我快乐-1")

.pubDate(new Date())

.build();

IndexQuery indexQuery = new IndexQueryBuilder()

.withId(book.getId().toString())

.withObject(book)

.build();

IndexCoordinates indexCoordinates = restTemplate.getIndexCoordinatesFor(Book.class);

String documentId = restTemplate.index(indexQuery, indexCoordinates);

log.info("test index add content : {}", documentId);

}

**查询数据：**

public class SearchQuery {

@Autowired

private ElasticsearchRestTemplate restTemplate;

/\*\*

\* 简单查询

\*/

@Test

public void search1() {

Criteria criteria = new Criteria()

.and("author\_name").is("lin")

.and("book\_name").contains("java");

Query query = new CriteriaQuery(criteria);

IndexCoordinates indexCoordinates = restTemplate.getIndexCoordinatesFor(Book.class);

SearchHits<Book> search = restTemplate.search(query, Book.class, indexCoordinates);

log.info("search1----{}",search.getTotalHits());

}

/\*\*

\* 标准查询

\*/

@Test

public void search2() {

Query query = new NativeSearchQueryBuilder()

//查询条件

.withQuery(QueryBuilders.matchQuery("book\_name","java"))

//分页

.withPageable(PageRequest.of(0, 5))

//排序

.withSort(SortBuilders.fieldSort("id").order(SortOrder.DESC))

//高亮字段显示 及自定义tag

.withHighlightFields(new HighlightBuilder.Field("book\_name").preTags("<span>").postTags("</span>"))

.build();

IndexCoordinates indexCoordinates = restTemplate.getIndexCoordinatesFor(Book.class);

SearchHits<Book> search = restTemplate.search(query, Book.class, indexCoordinates);

log.info("search1----{}", search.getTotalHits());

}