

Elasticsearch

一、简介

1.1 海量数据遇到瓶颈

如：当系统数据量上了10亿、100亿条的时候，我们在做系统架构的时候通常会从以下角度去考虑问题：

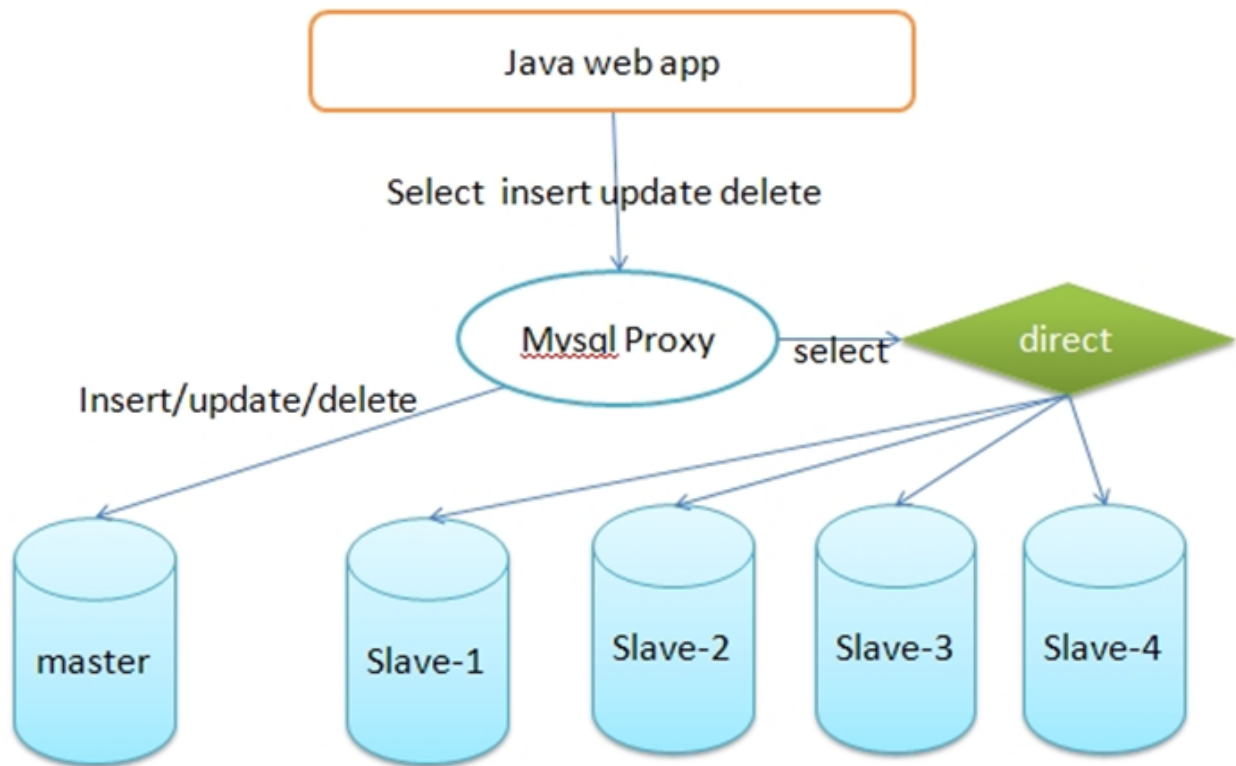
- 1) 用什么数据库好？(mysql、sybase、oracle、达梦、神通、mongodb、hbase, ProgreSQL...)
- 2) 如何解决单点故障；(lvs、F5、A10、Zookeeper、MQ)
- 3) 如何保证数据安全性；(热备、冷备、异地多活)
- 4) 如何解决检索难题；(数据库代理中间件：MyCat、mysql-proxy、S-jdbc、Saga等;)
- 5) 如何解决统计分析问题；(离线、近实时)

1.2 传统解决方案

对于关系型数据，我们通常采用以下或类似架构去解决查询瓶颈和写入瓶颈：

解决要点：

- 1) 通过主从备份解决数据安全性问题；
- 2) 通过数据库代理中间件心跳监测，解决单点故障问题；
- 3) 通过代理中间件将查询语句分发到各个slave节点进行查询，并汇总结果

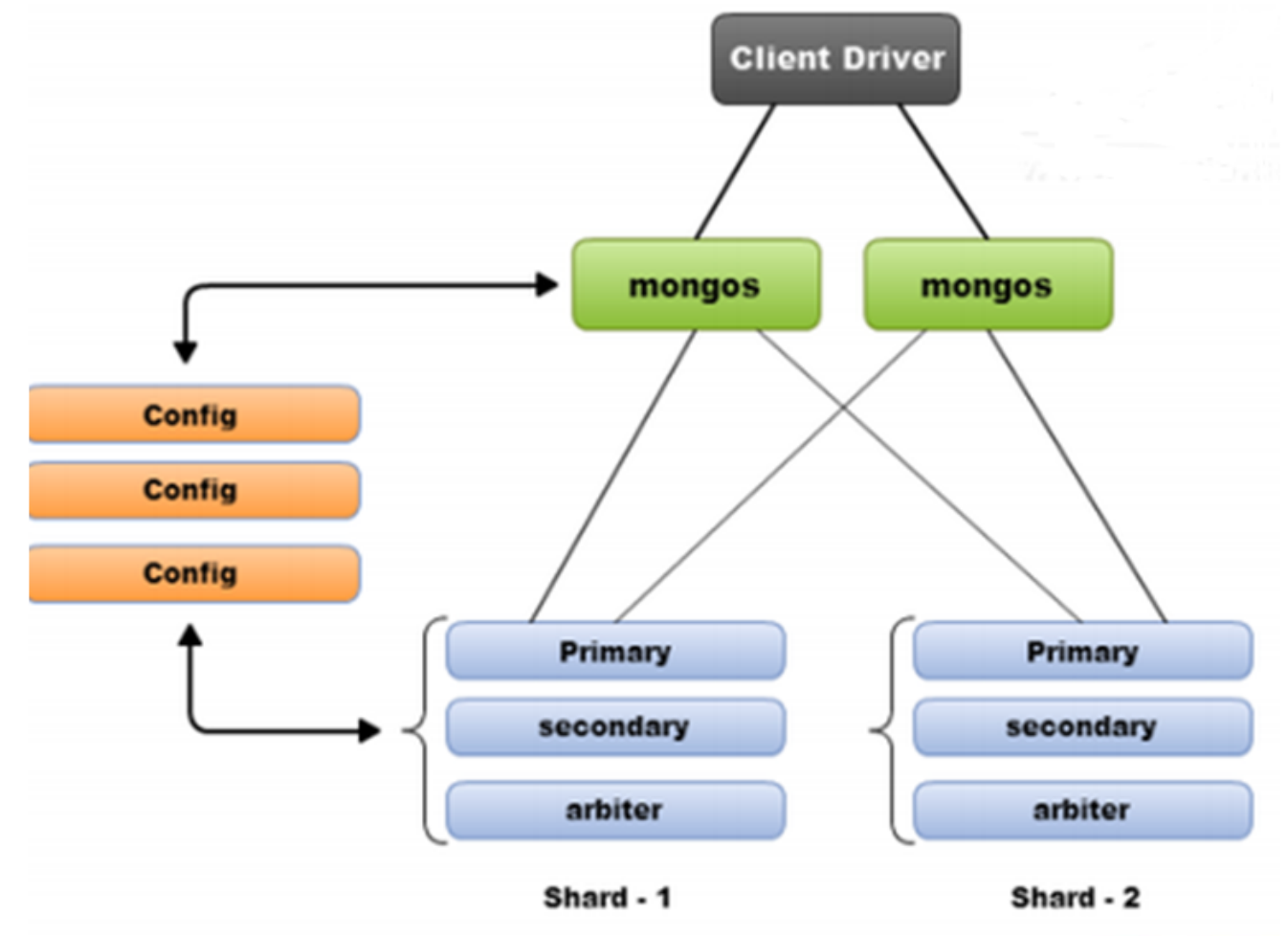


非关系型数据库的解决方案：

对于Nosql数据库，以mongodb为例，其它原理类似：

解决要点：

- 1) 通过副本备份保证数据安全性；
- 2) 通过节点竞选机制解决单点问题；
- 3) 先从配置库检索分片信息，然后将请求分发到各个节点，最后由路由节点合并汇总结果



1.3大型网站的搜索

百度、淘宝、京东 BAT TMD



在一些大型门户网站、电子商务网站等都需要站内搜索功能，使用传统的数据库查询方式实现搜索无法满足一些高级的搜索需求，比如：搜索速度要快、搜索结果按相关度排序、搜索内容格式不固定等，这里就需要使用全文检索技术实现搜索功能。

主流的全文检索技术：

1、Lucene

单独使用Lucene实现站内搜索需要开发的工作量较大，主要表现在：索引维护、索引性能优化、搜索性能优化等，因此不建议采用。

2、Solr

基于Solr实现站内搜索扩展性较好并且可以减少程序员的工作量，因为Solr提供了较为完备的搜索引擎解决方案，因此在门户、论坛等系统中常用此方案。

3、Elasticsearch

Elasticsearch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。

1.4 Solr

Solr 是Apache下的一个顶级开源项目，采用Java开发，它是基于Lucene的全文搜索服务器。Solr提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展，并对索引、搜索性能进行了优化。

Solr可以独立运行，运行在Tomcat、Jetty等这些Servlet容器中，Solr 索引的实现方法很简单，用 POST 方法向 Solr 服务器发送一个描述 Field 及其内容的 XML 文档，Solr根据xml文档添加、删除、更新索引。Solr 搜索只需要发送 HTTP GET 请求，然后对 Solr 返回Xml、json等格式的查询结果进行解析，组织页面布局。Solr不提供构建UI的功能，Solr提供了一个管理界面，通过管理界面可以查询Solr的配置和运行情况。

1.5 Elasticsearch

Elasticsearch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。

Elasticsearch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

Solr也是搜索引擎 基于Lucene

<https://www.elastic.co/cn/>

核心：

- 1、数据存储 海量 实时
- 2、数据检索 实时

1.6 Elasticsearch数据模型

关系数据库 ⇒ 数据库 ⇒ 表 ⇒ 行 ⇒ 列(Columns)

Elasticsearch ⇒ 索引(Index) ⇒ 类型(type) ⇒ 文档(Documents) ⇒ 字段(Fields)

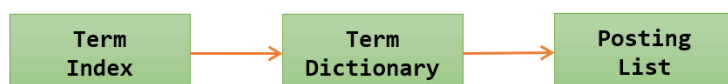
可以包含多个索引(数据库)，也就是说其中包含了很多类型(表)。这些类型中包含了很多的文档(行)，然后每个文档中又包含了很多的字段(列)。Elasticsearch的交互，可以使用Java API，也可以直接使用HTTP的Restful API方式

1.7Elasticsearch索引

一切设计都是为了提高搜索的性能

1、倒排索引 (Inverted Index) 也叫反向索引

适用于快速的全文搜索。一个倒排索引由文档中所有不重复词的列表构成，对于其中每个词，有一个包含它的文档列表。

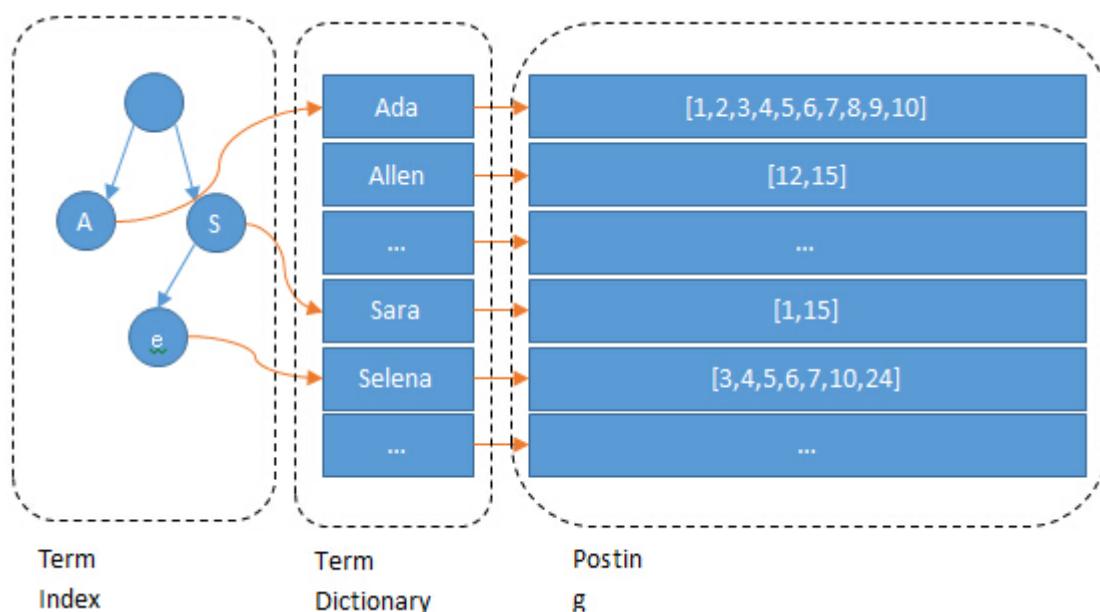


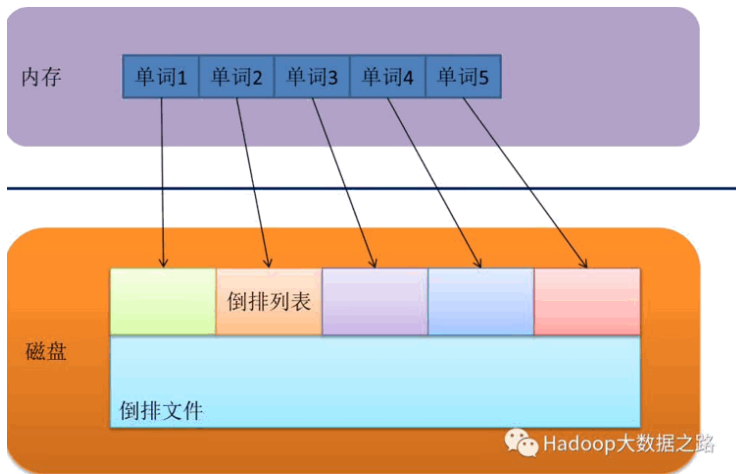
Term (单词)：一段文本经过分析器分析以后就会输出一串单词，这一个个的就叫做Term (直译为：单词)

Term Dictionary (单词字典)：顾名思义，它里面维护的是Term，可以理解为Term的集合

Term Index (单词索引)：为了更快的找到某个单词，我们为单词建立索引

Posting List (倒排列表)：倒排列表记录了出现过某个单词的所有文档的文档列表及单词在该文档中出现的位置信息，每条记录称为一个倒排项(Posting)。根据倒排列表，即可获知哪些文档包含某个单词

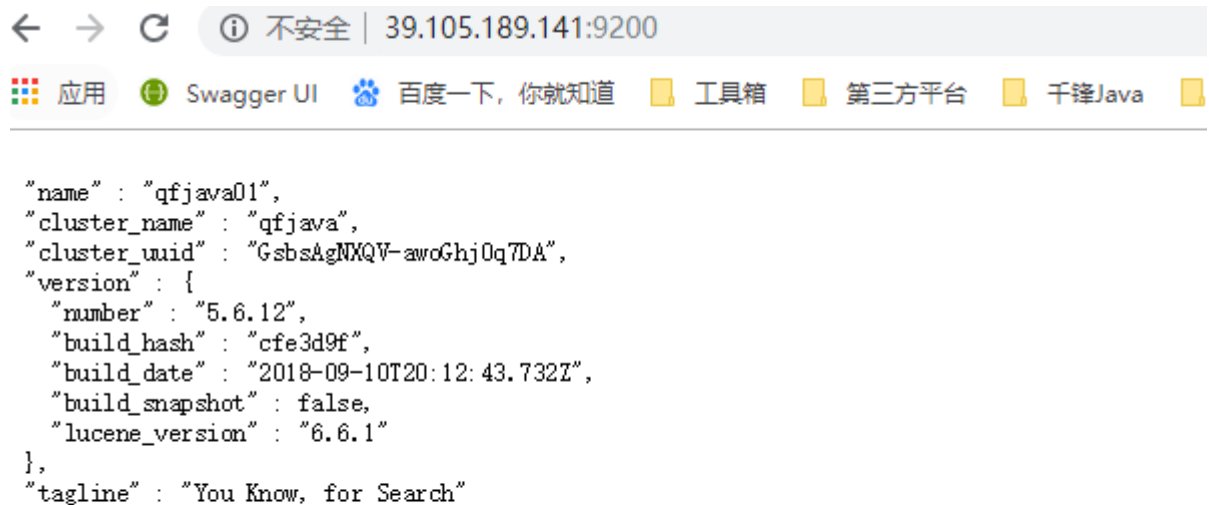




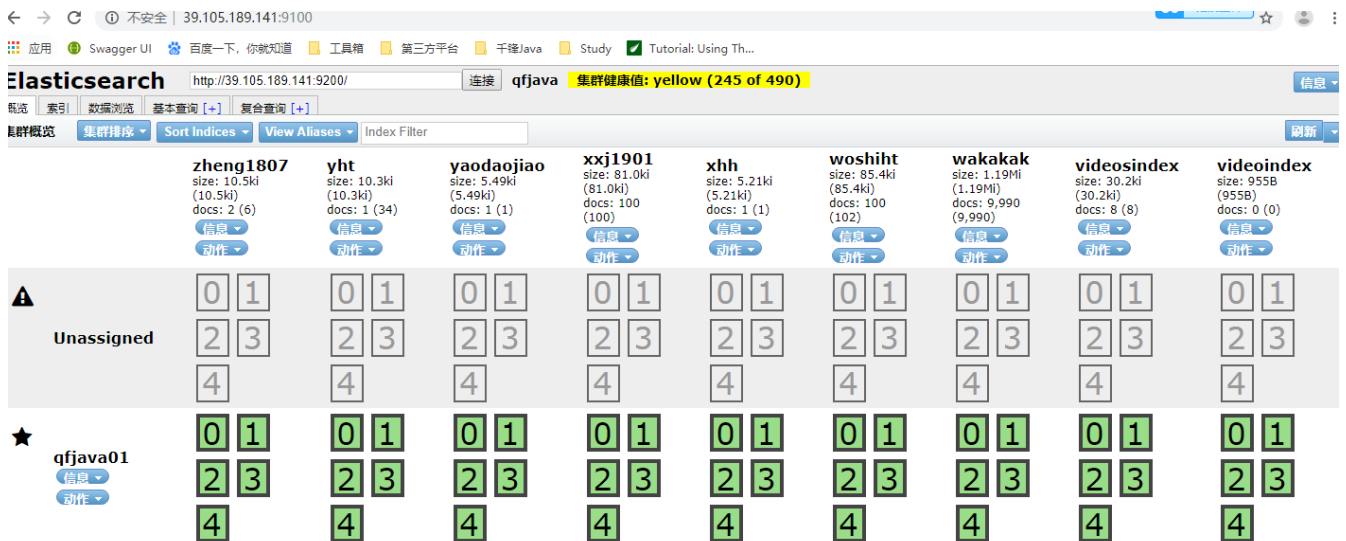
二、初体验

1. 搭建ES服务器

基于Docker搭建ES服务器



基于Docker搭建ES-Head 可视化网页



<http://39.105.189.141:9100/> 可视化地址

<http://39.105.189.141:9200/> ES服务器地址

客户端连接使用9300

公网的ES:

主机: 39.105.189.141

端口号: 9300

集群名称: qfjava

2、代码实现

1、依赖jar

..

```
<!-- https://mvnrepository.com/artifact/org.elasticsearch/elasticsearch-x-content -->
<dependency>
    <groupId>org.elasticsearch</groupId>
    <artifactId>elasticsearch-x-content</artifactId>
    <version>6.5.4</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.elasticsearch.client/transport -->
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>transport</artifactId>
    <version>6.5.4</version>
</dependency>
```

2、代码实现

1、创建设置对象 指定集群名称

2、设置服务器地址 创建客户端对象

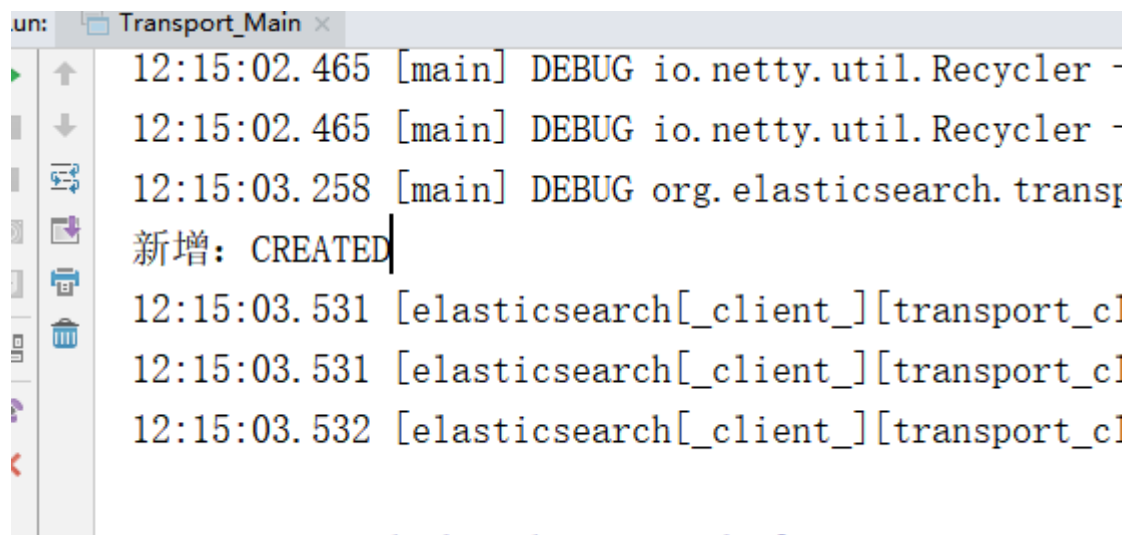
3、实现操作 增删改查

4、关闭

代码示例：

..

```
public static void main(String[] args) {
    //1、创建设置对象
    Settings settings=Settings.builder().put("cluster.name","qfjava").build();
    //2、创建连接对象
    Client client=new PreBuiltTransportClient(settings).
        addTransportAddress(new TransportAddress(new
InetSocketAddress("39.105.189.141",9300)));
    //3、操作ES服务器 CRUD
    //新增
    /**
     * 新增或修改 id就是新增 id就是修改
     * 参数说明:
     * 1、索引名称
     * 2、类型
     * 3、id 唯一值*/
    IndexResponse indexResponse=client.prepareIndex("es1902","food","1").
        setSource("{\"id\":\"1\",\"name\":\"鸡腿\"}",XContentType.JSON).get();
    System.out.println("新增: "+indexResponse.status().toString());
    //4、销毁
    client.close();
}
```



```
un: Transport_Main x
12:15:02.465 [main] DEBUG io.netty.util.Recycler -
12:15:02.465 [main] DEBUG io.netty.util.Recycler -
12:15:03.258 [main] DEBUG org.elasticsearch.transp
新增: CREATED
12:15:03.531 [elasticsearch[_client_][transport_c]
12:15:03.531 [elasticsearch[_client_][transport_c]
12:15:03.532 [elasticsearch[_client_][transport_c]
```




三、Java操作ES

3.1Java操作ES的方式

Java程序操作ES服务器有2种方式：

- 1、ES 的客户端 Transport
- 2、Spring Data Elasticsearch

Spring Data 系列：Spring给出的操作一切数据的框架

- 1、Spring Data JPA
- 2、Spring Data Redis
- 3、Spring Data ElasticSearch

存储数据：

- 1、数据库 关系型数据库

Mysql Oracle

- 2、NO-SQL数据库

Redis

- 3、全文检索技术（搜索引擎技术）

ElasticSearch

3.2 Transport操作Java

3.2.1 基本操作

新增或修改:

```
prepareIndex    id就是新增 id就是修改
```

``

```
/**
 * 新增或修改 id就是新增 id就是修改
 * 参数说明:
 * 1、索引名称
 * 2、类型
 * 3、id 唯一值*/
IndexResponse indexResponse=client.prepareIndex("es1902","food","1").
    setSource("{\"id\":1,\"name\":\"鸡腿\"}",XContentType.JSON).get();
```

修改:

``

```
UpdateResponse updateResponse=client.prepareUpdate("es1902","offer",offer.getId()+"").
    setDoc(JSON.toJSONString(offer),XContentType.JSON).get();
System.out.println("修改: "+updateResponse.status().name());
```

删除:

``

```
DeleteResponse
deleteResponse=client.prepareDelete("es1902","offer",offer.getId()+"").get();
System.out.println("删除: "+deleteResponse.status().name());
```

查询:

``

```
GetResponse getResponse=client.prepareGet("es1902","offer",offer.getId()+"").get();
String json=getResponse.getSourceAsString();
System.out.println("查询: "+json);
```

3.2.2 各种查询

Transport 提供了常用的查询

RangeQueryBuilder 范围查找 gt:大于 lt:小于 gte:大于等于 lte:小于等于

wildcardQueryBuilder 模糊查询 使用*表示任意字符

TermQueryBuilder 精确查询

BoolQueryBuilder 布尔类型查询 用于多条件的拼接
must 必须的 文档必须满足匹配条件
must_not 文档必须不匹配条件
should 或者 文档只要满足一个或多个条件

代码示例:

``

```
public static void main(String[] args) {
    //1、创建设置对象
    Settings settings=Settings.builder().put("cluster.name","qfjava").build();
    //2、创建连接对象
    Client client=new PreBuiltTransportClient(settings).
        addTransportAddress(new TransportAddress(new
InetSocketAddress("39.105.189.141",9300)));
    //3、操作ES服务器
    //条件查询
    //1、创建查询内容
    //范围查询
    // RangeQueryBuilder rangeQueryBuilder=QueryBuilders.rangeQuery("id").gt("1").lt("10");
    //模糊查询
    wildcardQueryBuilder wildcardQueryBuilder=QueryBuilders.wildcardQuery("name","*小*");
    //单词查询 精确查询
    TermQueryBuilder termQueryBuilder=QueryBuilders.termQuery("id","101");
    //2、条件拼接
    BoolQueryBuilder boolQueryBuilder=QueryBuilders.boolQuery();
    boolQueryBuilder.should(wildcardQueryBuilder);
    boolQueryBuilder.should(termQueryBuilder);
    //3、创建查询对象
    SearchSourceBuilder searchSourceBuilder=new SearchSourceBuilder();
    searchSourceBuilder.query(boolQueryBuilder);
    //4、执行条件查询 链式编程
    SearchResponse searchResponse=client.prepareSearch("es1902").//指定查询的索引名称
        setTypes("offer"). //设置查询的类型
        addSort("id",SortOrder.DESC). //设置排序
        setQuery(searchSourceBuilder.query()). //设置查询条件
}
```

```

        get(); //获取查询结果

//5、解析查询结果
SearchHits searchHits=searchResponse.getHits();
SearchHit[] arrhits=searchHits.getHits();
//遍历数组
for(SearchHit sh:arrhits){
    System.out.println(sh.getSourceAsString());
}

//4、销毁
client.close();
}

```

3.2.3 批处理

BulkRequestBuilder 批处理

代码示例：

、、

```

public static void main(String[] args) {
    //1、创建设置对象
    Settings settings=Settings.builder().put("cluster.name","qfjava").build();
    //2、创建连接对象
    Client client=new PreBuiltTransportClient(settings).
        addTransportAddress(new TransportAddress(new
    InetSocketAddress("39.105.189.141",9300)));
    //3、操作ES服务器
    //实现批处理
    BulkRequestBuilder bulkRequestBuilder=client.prepareBulk();
    for(int i=1;i<10001;i++){
        Offer offer=new Offer();
        offer.setAddress("叙利亚");
        offer.setMoney(200000);
        offer.setName("轩轩");
        offer.setId(1000000+i);
        bulkRequestBuilder.add(client.prepareIndex("es1902","offer",offer.getId()+"").
            setSource(JSON.toJSONString(offer),XContentType.JSON));
    }
    //执行操作
    BulkResponse bulkItemResponses=bulkRequestBuilder.get();
    System.out.println("批处理: "+bulkItemResponses.status().name());
    //4、销毁
    client.close();
}

```

完整代码示例：

3.3 Spring Data Elasticsearch

四、Elasticsearch核心

1、数据存储的核心：

1、索引

存储数据的对象 类似之前数据库中的数据库，可以存储多种类型的数据

2、类型

数据的类型 类似之前数据库中表

3、id

数据的唯一值

4、source

内容 一般存储都是JSON格式字符串