目录

[摘要 2](#_Toc475285892)

[第一章 绪论 3](#_Toc475285893)

[1.1 课题背景 3](#_Toc475285894)

[1.2 研究现状及发展趋势 3](#_Toc475285895)

[1.3 课题意义与目的 3](#_Toc475285896)

[1.4 研究内容 4](#_Toc475285897)

[1.5 论文形式 4](#_Toc475285898)

[第二章 相关技术介绍 4](#_Toc475285899)

[2.1 elasticsearch 4](#_Toc475285900)

[第三章 需求分析 5](#_Toc475285901)

[3.1 技术可行性 5](#_Toc475285902)

[3.2 经济可行性 5](#_Toc475285903)

[3.3 功能需求分析 5](#_Toc475285904)

[第四章 总体设计 6](#_Toc475285905)

[4.1 结构设计 6](#_Toc475285906)

[4.2 功能模块设计 7](#_Toc475285907)

[4.3 信息封装类设 8](#_Toc475285908)

[第五章 详细设计与实现 9](#_Toc475285909)

[5.1 驱动模块 9](#_Toc475285910)

[5.2 控制模块 10](#_Toc475285911)

[5.3 查询模块 12](#_Toc475285912)

[5.4 添加模块 17](#_Toc475285913)

[5.5 删除模块 18](#_Toc475285914)

[5.6 更新模块 19](#_Toc475285915)

[5.7 创建模块 20](#_Toc475285916)

[5.8 修改模块 21](#_Toc475285917)

[5.9 ES客户端模块 21](#_Toc475285918)

[第六章 使用说明 21](#_Toc475285919)

[6.1 总体步骤 21](#_Toc475285920)

[6.2 功能操作 22](#_Toc475285921)

[6.2.1 查询 22](#_Toc475285922)

[6.2.2 增删改 23](#_Toc475285923)

[6.2.3 创建表 23](#_Toc475285924)

[6.2.4 修改表 24](#_Toc475285925)

[第七章 总结 24](#_Toc475285926)

**基于java的elasticsearch操作接口的封装**

**数学与计算机科学学院 计算机科学与技术专业**

**105032013072 黄学雯 指导教师：张大平**

**【摘要】无疑，我们已经进入互联网的时代，几乎每个人都在使用互联网产品，对于互联网产品来说，使用的人越多，产生的数据越多，所要承受的性能压力也越大。在海量数据的情况下，为了使产品仍然有高效的搜索性能，越来越多的互联网产品同elasticsearch结合。比起oracle等数据库，elasticsearch在查询性能上有明显的优势，而且，elasticsearch也可以用于数据的分析。所以，在大数据的驱动下，elasticsearch将得到广泛的应用。**

**Elasticsearch JDBC（以下简称ES JDBC）是针对在java开发中使用elasticsearch而开发的一套接口，它封装了elasticsearch基本操作，主要是数据的增删改查以及表的创建修改；并且它的接口是以类似于JDBC的形式呈现。**

**本文首先分析了ES JDBC的现状以及未来的发展趋势、ES JDBC开发的意义，同时也对相关技术进行简要介绍，然后是ES JDBC的需求分析，包括可行性分析以及功能分析，紧接着是ES JDBC的设计以及各模块的实现，最后对本次开发进行了总结。**

**ES JDBC的开发平台是eclipse，所依赖的jdk版本是jdk1.8.**

**ES JDBC简化了对elasticsearch的操作，减少了开发的工作量，缩短了开发周期。JDBC形式的接口也方便了开发人员的使用。**

**【关键词】**elasticsearch；java；JDBC

# 1.绪论

## 课题背景

随着计算机的普及以及网络的快速发展，各式各样的互联网产品应运而生，而这些互联网产品的诞生也极大地简便了人们的生活，因此，使用互联网产品的人越来越多，互联网产品逐渐成为人们生活中不可或缺的一部分。然而在使用人数人数不断增加以及全国大联网的形势下，这些互联网产品在性能方面也面临着严峻的挑战。它们必须能够在快速将请求信息返回给客户端，如果是使用常规的数据库，比如oracle等来存储数据，那么在海量数据的情况下，这样的业务要求是无法达到的，所以更多的互联网产品会选择使用elasticsearch来存储一些在产品使用过程中变化不大的静态数据，elasticsearch能够解决查询效率的问题，但是elasticsearch提供的原始的java接口使用起来较为复杂，而且产品的开发人员还要花一定的时间去学习使用elasticsearch，这无疑增加了开发的工作量，延长了开发周期。所以针对elasticsearch的使用情况，我们可以对elasticsearch的原始接口进行二次封装，提供一个更简单的接口——ES JDBC，从而达到简化开发的目的。

## 研究现状及发展趋势

现如今，大数据逐渐成为一种趋势，elasticsearch凭借其高效的搜索特性进入大众的视线，国内外已经有很多IT公司使用elasticsearch来搜索，分析数据，比如Github，Facebook以及百度等[[[1]](#endnote-1)]，所以随着大数据时代的到来，elasticsearch将会得到广泛的应用；另一方面，java作为世界上主流的编程语言之一，从中小型企业的应用到大型互联网系统的开发，java始终占据着重要的地位，而且，elasticsearch提供了java开发的接口，我们可以相信越来越多的基于java开发的产品会同elasticsearch相结合。但是elasticsearch提供的原始的java接口的使用过于复杂，开发人员还需要花一定的时间去学习，所以为了达到高效开发的要求，更多的IT公司会趋向于自己封装elasticsearch的java 操作接口，提供一个更简单的接口。

## 课题意义与目的

从ES JDBC本身的价值来讲，elasticsearch在java 开发中的应用已经越来越广泛，这说明java 开发人员要像熟悉数据库一样熟悉elasticsearch的操作，ES JDBC建立在elasticsearch原始的接口之上，封装了elasticsearch大部分的操作，包括对数据的增删改查等，更重要的是ES JDBC是采用类似于JDBC的形式，所有对elasticsearch的操作都用sql语句来表达。这样就简化了elasticsearch的操作代码，从而减轻了开发工作量，而且由于开发人员对JDBC接口的熟悉，所以开发人员不需要花费太多的时间去学习ES JDBC。总之，ES JDBC可以简化开发，方便开发人员操作elasticsearch，而且容易上手。

从自身的角度来讲，通过本次开发可以深入了解elasticsearch，掌握elasticsearch的应用以及操作；学习相关的设计模式，并将其应用于开发中；巩固并且加强基于java的程序设计开发，最重要的是在开发中培养发现问题，分析问题，解决问题的能力。

## 研究内容

本次课题的研究重点是ES JDBC的设计以及开发，具体内容概括如下：

1.需求分析，通过与开发人员的沟通交流，确定了ES JDBC需要提供对数据的增删改查，对表的创建修改功能。

2.分析目前部分技术提供的java API的形式，确定ES JDBC采用类似于JDBC的形式。

3.基于高内聚，低耦合的原则[[[2]](#endnote-2)]，设计出ES JDBC的结构，并且对结构中各个模块的具体实现做深入研究。

4.实现了ES JDBC 的开发，按照面向接口的开发原则，以生成器模式为主，将elasticsearch所能接收的数据模型的构建过程与表示分离[[[3]](#endnote-3)]，利用JDBC 提供的接口实现开发。

## 1.5 论文形式

本文由八个章节组成：

第一章：绪论。主要介绍课题背景、研究现状以及发展趋势、课题意义与目的、研究内容，最后大体介绍论文的形式。

第二章：相关技术介绍。该章节主要介绍elasticsearch的特点以及相关名词的解释

第三章：需求分析。该章节从技术可行性、经济可行性、功能需求三个方面对ES JDBC的开发进行分析

第四章：总体设计。主要介绍ES JDBC的结构、功能模块、信息分装类

第五章：详细设计与实现。详细地介绍了每个功能模块的具体实现

第六章：使用说明。举例说明如何使用ES JDBC对elasticsearch进行增删改查、创建表、修改表等功能

第七章：验证。对ES JDBC 的查询性能进行简单的验证，并且同mysql进行比较

第八章：总结。对本次课题的研究进行总结。分析了本次课题的难点、不足之处以及收获

# 2. 相关技术介绍

## 2.1 elasticsearch

Elasticsearch 是一个开源分布式搜索引擎，它在全文搜索、结构化查询以及数据分析等方面都表现出色，因此elasticsearch被广泛地应用于各种检索系统。它的特点是高可用性、高可扩展性和准实时性，并且elasticsearch通过REST API来操作数据，所有对elasticsearch的请求都能够以基于json的http请求来发送[[[4]](#endnote-4)]。关于elasticsearch有几个关键的概念需要了解：

1.索引：index，一个索引就是一个拥有相似特征的文档的集合，索引类似于关系型数据库中的数据库，一个elasticsearch集群可以有多个索引，每个索引可以分多个分片存储，而每个分片又可以有多个副本，这在一定程度上提高了搜索效率，保证了elasticsearch的高可用性，

2.类型：type，一个类型就是索引上的一个分类，类似于关系型数据库中的表，一个索引中可以有多个类型。

3.文档：document，文档是是存储在elasticsearch中的主要实体 ，类似于关系数据库中表的每一条记录[[[5]](#endnote-5)]。

# 3.需求分析

## 3.1 技术可行性

ES JDBC的开发是基于elasticsearch 为java提供的接口的二次封装，通过实现JDBC 的相关接口来实现类似于数据库查询的形式；ES JDBC有两个技术关键点：

1.sql语句的解析，即从字符串类型的sql语句中获取操作类型、表名、条件。通过使用facebook的SqlParser类可以降低解析的复杂度。

2.elasticsearch 响应体的解析，即将elasticsearch的查询结果转换成resultSet的形式。设计好resultSet类的结构，以递归的形式获取响应体中每一层的结构，存放到resultSet中。

## 3.2 经济可行性

ES JDBC开发周期一般为两个月，开发所需要的硬件设备只需要普通的pc机以及一台运行elasticsearch的服务器，这样的配置要求，一般的公司都可以达到，不需要额外的花费；另外，ES JDBC的使用是作为jar依赖导入到其他项目中的，所以不需要另外的安装部署；总之ES JDBC的开发、运行以及维护都不需要太高的费用，所以，ES JDBC在经济上是可行的。

## 3.3 功能需求分析

ES JDBC应该支持如下操作

1.查询数据

（1）可以获取所有列的值，也可以获取特定列的值

（2）and、or组成的复杂条件查询

（3）聚合函数查询

（4）分组查询

（5）分页查询

2.新增数据

（1）插入单条数据

（2）批量插入数据

3.修改数据

（1）修改单条数据

（2）批量修改数据

4.删除数据

（1）删除单条数据

（2）批量删除数据

5.创建表

（1） 创建指定的表

6.修改表字段

（1）对已创建的表可以增加表的字段

# 4.总体设计

## 4.1 总体流程

为了使ES JDBC的使用更简单，开发人员能够容易上手，同时又能够达到封装elasticsearch操作的目的，该API采用类似于jdbc的形式，通过发送sql语句向elasticsearch发送操作命令。ES JDBC的处理流程按照输入的sql语句的类型分为查询sql流程和更新sql流程两种。查询sql指select 形式的sql；更新sql包括create table、alter table、insert、delete、update这些形式的sql。

### 4.1.1 查询sql流程

12.ESResultSet

10.SearchResponse

11.ESResultSet

9.SearchResponse

8.QueryBody

7.QueryBody

6.SelectSqlObj

5.SelectSqlObj

4.Sql

3.Sql

2.Connection

1.URL

ESDriver

ESStatement

SelectObjDirector

QueryDirector

ESClient

ResultDirector

ES JDBC

图4-1 查询sql流程

查询sql流程：

1.用户将要连接的elasticsearch服务的信息以url的形式发送给es驱动ESDriver

2.ESDriver解析url，建立到elasticsearch的连接，并且将自定义的连接对象Connection返回给用户

3.用户将查询sql发送给ESStatement对象

4.ESStatement调用SelectObjDirector对象，并且将sql语句传递给它。

5.SelectObjDirector解析sql语句，从中获取查询信息，比如表名、查询列、查询条件等，并将这些信息分装成自定义的SelectSqlObj对象返回给ESStatement。

6.ESStatement调用QueryDirector，并将SelectSqlObj传递给它。

7.QueryDirector利用封装好的查询信息，构造elasticsearch的QueryBuilder和AggregationBuilder对象，将这两个对象封装在自定义对象QueryBody中返回给ESStatement

8.ESStatement调用ESClient，将QueryBody传递给它。

9.ESClient对elasticsearch执行查询操作，将elasticsearch的查询响应对象SearchResponse返回给ESStatement

10.ESStatement调用ResultDirector，将SearchResponse传递给它。

11.ResultDirector解析SearchResponse，将查询结果集封装成自定对象ESResultSet，将ESResultSet返回给ESStatement。

12.ESStatement将ESResultSet返回给用户

### 4.1.2 更新sql流程

6.Obj

5. DeleteSqlObj

4.Deletesql

5. InsertSqlObj

4.Insertsql

5. UpdateSqlObj

4.Updatesql

5.Mapping

4.CreateSql、AlterSql

3.Sql

2.Connection

1.URL

ESDriver

ESStatement

UpdateDirector

ESClient

ES JDBC

图4-2 更新sql流程

更新sql流程：

1.用户将要连接的elasticsearch服务的信息以url的形式发送给es驱动ESDriver

2.ESDriver解析url，建立到elasticsearch的连接，并且将自定义的连接对象Connection返回给用户

3.用户将更新sql发送给ESStatement对象

4.ESStatement根据sql不同的形式调用UpdateDirector对应的处理方法。

5.UpdateDirector解析sql，从中获取更新信息，比如表、更新字段等，将这些信息封装成自定义的类，不同形式的sql返回不同的信息封装对象。

6.ESStatement将信息分装对象传递给ESClient，由ESClient对elasticsearch执行对应的操作。

## 4.2 功能模块设计

ES JDBC 各模块功能如下：

ES JDBC

添加模块

控制模块

查询模块

删除模块

更新模块

修改模块

创建模块

图4-3 ES JDBC 功能模块

ES客户端模块

驱动模块

固定sql模块

动态sql模块

1.驱动模块：解析连接信息，建立对elasticsearch的连接

2.控制模块：ES JDBC的核心，通过判断输入的sql语句的类型（查询或者更新等）将sql请求转发给对应的模块来处理，最后接收相应的结果返回给用户；支持固定sql和动态sql

3.查询模块： 解析sql语句，根据解析的结果对elasticsearch执行查询，最后封装查询结果；支持的查询类型包括：聚合查询，去重查询，条件查询，排序查询，分页查询。

4.添加模块：解析sql语句，对已存在的表插入新数据；支持批量插入

5.删除模块：解析sql语句，删除符合某个条件的数据；支持批量删除

6.更新模块：解析sql语句，更新某些记录的值；支持批量更新

7.修改模块：解析sql语句，为表添加新字段或者修改已有字段的属性

8.创建模块:读取源文件下的配置文件，根据配置文件创建对应的表

9.ES客户端模块:持有elasticsearch连接对象，封装对elasticsearch的操作，供其他功能模块调用

## 4.3 信息封装类设计

1.SelectSqlObj 类:该类封装查询sql的信息，是查询sql解析阶段的产物

表4-1 SelectSqlObj 类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| distinct | Boolean | 是否需要去重；需要为true，否则为false |
| selectItems | List<ColumnMate> | 查询列；ColumnMate：自定义类，包含列名，别名，聚合操作类型 |
| from | String | 表名 |
| where | ConditionExp | 查询条件；ConditionExp：自定义类，包含当前元条件、下一个元条件、与下一个元条件的关系；  包含and/or的逻辑表达式根据and/or分成多个元条件，元条件之间的关系即为and/or |
| groupby | List<ColumnMate> | 分组信息 |
| having | ConditionExp | 分组聚合后需要满足的条件 |
| orderby | List<OrderbyMate> | 排序信息；OrderbyMate：自定义类，包含排序字段，排序类型 |
| limit | PageMate | 分页信息；PageMate自定义类，包含起始记录下标，获取记录总数 |

2.QueryBody类:该类封装elasticsearch所能接收的数据模型，是查询体构建阶段的产物

表4-2 QueryBody类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| queryBuilder | QueryBuilder | 查询条件信息；QueryBuilder：elasticsearch类 |
| aggregationBuilder | AggregationBuilder | 聚合信息；AggregationBuilder：elasticsearch类 |
| pageMate | PageMate | 分页信息 |
| orderby | List<OrderbyMate> | 排序信息 |

3.ESResultSet 类:该类封装查询结果，是查询结果构建阶段的产物

表4-3 ESResultSet 类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| resultList | List<Map<String, Object>> | 符合查询条件的记录，一个map对象代表一条记录的一个字段值 |
| orderby | List<OrderbyMate> | 排序信息 |
| typeAllColumns | List<String> | 该表的所有字段名 |
| metaData | ESResultSetMetaData | 有关 ESResultSet中列的名称和类型的信息 |
| total | Int | 查询结果记录总数 |
| index | Int | 遍历结果集时，记录当前记录的下标 |

4.UpdateSqlObj类:该类封装更新sql的信息，是更新sql解析阶段的产物

表4-4 UpdateSqlObj类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| index | String | 索引名 |
| type | String | 表名 |
| updateList | List<ColumnValue> | 需要更新的列 |
| ids | List<String> | 满足更新条件的记录Id |

5.DeleteSqlObj类：该类封装删除sql的信息，是删除sql解析阶段的产物

表4-5 DeleteSqlObj类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| type | String | 表名 |
| ids | List<String> | 满足删除条件的记录Id |

6.InsertSqlObj类：该类封装插入sql的信息，是插入sql解析阶段的产物

表4-6 InsertSqlObj类字段说明

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| type | String | 表名 |
| valueList | List<ColumnValue> | 需要插入的数据 |

# 5.详细设计与实现

## 5.1 驱动模块

驱动模块通过实现JDBC的Driver接口来实现驱动注册和获取连接的功能。当调用端发送包含连接信息的url时，程序首先通过正则表达式来判断该URL是否满足一定的格式，若是满足，则解析该URL，从中获取elasticsearch的主机地址，端口号以及索引名，然后调用ES客户端模块建立到elasticsearch的连接，紧接着获取elasticsearch该索引的mapping，从而获取该索引下的所有字段以及字段类型，最后封装连接对象，返回给调用端；若是不满足则抛出url不合法的异常。

图5-1 驱动模块流程图

判断URL格式是否合法

调用端发送获取连接的请求

获取主机、端口、索引名

建立到elasticsearch的连接

获取该索引下所有字段的类型

返回连接对象

抛出异常

## 5.2 控制模块

1.固定sql控制模块：该模块实现JDBC的Statement接口，该接口的executeQuery方法和executeUpdate分别处理查询sql和更新sql；若是执行查询sql，则直接调用查询模块，获取查询结果，返回给调用端；若是执行更新sql，则进一步判断是哪种类型的更新操作，再调用对应的模块执行相应操作，若所有类型的更新操作都不匹配，则抛出sql不合法的异常。

否

否

否

否

是

是

是

是

是

否

否

是

调用固定sql控制器

执行查询

查询

返回查询结果

Sql以create开头

Sql以update开头

Sql以insert开头

Sql以delete开头

Sql以alter开头

创建表

更新表数据

插入新数据

删除数据

修改表配置

抛出异常

图5-2 固定sql控制器流程图

2.动态sql控制模块：该模块实现JDBC的PreparedStatement接口；首先解析sql语句，将sql语句中的占位符分解出来，然后根据调用端设置的每个占位符代表的参数拼接sql语句，这样就将动态sql转化为固定sql，最后调用固定sql控制模块来执行该sql语句。

调用动态sql控制器

解析动态sql语句

拼接sql语句

执行固定sql

图5-3 动态sql控制器流程图

## 5.3 查询模块

查询模块的处理过程包括：sql解析阶段、查询体构建阶段、结果集构建阶段

1.sql解析阶段：sql的解析主要借助facebook的SqlParser类，SqlParser类可以对sql语句进行简单的解析，拆分出查询列，表名，限制条件，封装成一个Statement的对象；所以该阶段主要是在Statement对象的基础上，对各部分进行进一步解析，最终封装成SelectSqlObj对象。

distinct字段值、表名、分组字段、排序字段、分页信息可以从Statement对象的对应字段获取；而查询列需要根据三种形式select \* 、select column、select function（column）分别创建对应的列对象，最后设置列的别名；最后是解析where部分，where 条件分为以下几种类型：

表5-1 where 条件类型

|  |  |
| --- | --- |
| 类型 | 形式 |
| 关系运算表达 | 字段名 关系运算符 值（eg：age>=20） |
| null表达式 | 字段名 is null/is not null |
| In表达式 | 字段名 in （..） |
| not表达式 | 字段名 not 条件 |
| 区间表达式 | 字段名 between .. and .. |
| 逻辑表达式 | 条件1 and/or 条件2 and/or 条件3…. |

除了逻辑表达式以外的类型，直接构造对应的表达式类，而逻辑表达式则是以and/or 将表达式分为左右两个部分，每个部分递归构造。

否

是

否

是

输入查询sql

设置distinct字段值、表名、分组字段、排序字段、分页信息

查询所有字段

查询特定普通列

列名为\*，聚合类型为none

对应的列名，聚合类型为none

对应列名以及聚合类型

设置列对象的别名

解析where部分

返回查询对象

图5-4 Sql解析流图程

否

否

否

否

否

是

是

是

是

是

传入查询sql

Where为关系运算表达式

Where 为null表达式

Where为 In表达式

Where为not表达式

Where为区间表达式

构建Inequality对象

构建NullExpression对象

构建InExpression对象

构建ConNotExpression对象

构建BetweenExpression对象

获取where条件左部

获取where条件右部

图5-5 Where解析流程图

2.查询构建阶段：首先是distinct构建，若distinct的值为true，则使用elasticsearch的subAggregation构建查询体；接下来判断where 字段是否为空，若为空表示全匹配，使用elasticsearch的matchall 查询，若是不为空，则根据where 条件的类型使用elasticsearch的不同的查询；接着遍历selectItems集合，若查询列中包含聚合函数，则需要进一步判断是单纯在结果集中使用聚合函数，还是分组后使用聚合函数，前者直接使用elasticsearch对应的聚合函数接口，后者需要先构建聚合体，在最后一个聚合体中加入聚合函数。最后，通过判断聚合体是否为空来判断是否有聚合查询，若无，则添加分页和排序信息；若有，则不做任何处理，将分页和排序放到构建结果集阶段处理。

否

是

是

否

否

否

是

否

是

是

输入SelectSqlObj对象

distinct值为true

构建distinct查询体

where值不为null

构建对应查询

使用matchall查询

selectItems中的列包含聚合函数

Groupby不为空

添加分页和排序信息

聚合查询体为空

添加分组聚合查询

添加聚合查询

返回QueryBody对象

图5-6 查询体构建流程图

3.结果集构建阶段：根据elasticsearch响应体中是否包含聚合体可以将构建分为两种情况：聚合查询的结果集构建和条件查询的结果构建：

（1）条件查询的结果集构建：这种类型的搜索列只有可能是selelct \*或者 selelct column。前者直接遍历响应体中的所有列，将对应值加入到ESResultSet中；后者需要遍历搜索列，到响应体中查找对应列的值再将值加入到ESResultSet中。

（2）聚合查询的结果集构建：首先判断Distinct 值是否为真，若是为真，则解析聚合体，获取聚合体中所有字段的值，由于elasticsearch中没有count()这一聚合函数，所以需要在此阶段获取响应体中记录数作为count()函数的值；然后判断Groupby是否为空，若为空，表示该查询为单纯的聚合函数查询，从响应体中获取聚合函数的值加入到ESResultSet中，若不为空，表示该查询是先分组后聚合，解析聚合体，获取对应组的值。对于有分组操作的结果集，需要判断是否有having的限制条件，若有则从结果集中移除不符合条件的记录。最后根据orderby 和limit 对结果集排序和分页。

是

否

输入elasticsearch查询响应体

查询所有列

获取搜索列的值加入到ESResultSet

响应体中所有列加入到ESResultSet

返回ESResultSet对象

图5-7 结果集构建流程图

图5-7 获取搜索列结果集构建流程图

否

是

否

否

是

否

是

输入elasticsearch查询响应体

ESResultSet中加入记录数

Distinct 值为真

存在count聚合

Groupby为空

解析聚合体

ESResultSet中加入聚合函数值

解析聚合体

ESResultSet中移除不符合条件记录

结果集排序、分页

返回ESResultSet对象

having不为空

## 5.4 添加模块

首先使用SqlParser类创建statement对象，再从statement对象中获取表名和列名集合，然后遍历列名集合，获取列的值，将列名和值封装成ColumnValue对象，将ColumnValue对象添加到InsertSqlObj对象中的valueList，最后返回InsertSqlObj对象

图5-9 新增数据流程图

否

是

创建statement对象

获取表名、列名集合

输入sql语句

下标小于列名集合大小

获取对应列的值

封装ColumnValue对象，将该对象加入到InsertSqlObj中

返回InsertSqlObj对象

## 5.5 删除模块

首先通过判断sql语句是否包含where字段来区分是删除所有记录还是删除满足特定条件的记录，若是前者则构建查询语句：selelct \_id from table;后者构建的查询语句为：selelct \_id from table where ….然后调用查询模块获取要删除的记录的id，最后将id加入到DeleteSqlObj对象的ids中，将DeleteSqlObj对象返回

是

否

输入sql语句

Sql语句包含where字段

根据条件构建查询sql

构建无条件查询sql

执行查询sql

获取记录的id，加入到DeleteSqlObj中

返回DeleteSqlObj对象

图5-9 删除数据流程图

## 5.6 更新模块

首先使用正则表达式截取sql语句，获取更新列集合，接着遍历集合，将列名和新值封装成ColumnValue，添加到UpdateSqlObj对象中的updateList；然后根据是否有更新条件分别构建查询sql：select \_id from table或者 select \_id from table where ….,再调用查询模块获取要更新的记录的id，最后将id加入到UpdateSqlObj对象中的ids，返回UpdateSqlObj对象。

否

是

否

是

输入sql语句

获取更新列集合

下标小于列集合大小

获取对应列的更新值

封装ColumnValue对象，将该对象加入到UpdateSqlObj中

没有更新条件

根据条件构建查询sql

构建无条件查询sql

执行查询sql

获取记录的id，加入到UpdateSqlObj中

返回UpdateSqlObj对象

图5-11 更新数据流程图

## 5.7 创建模块

Elasticsearch创建表的时候需要为该表配置mapping，由于mapping需要组织成json格式，所以以配置文件的方式为表设置mapping。

输入sql语句

获取表名

在资源文件下获取以表名命名的配置文件

读取文件

返回json字符串

图5-12 创建表流程图

## 5.8 修改模块

修改模块指的是修改表的mapping，该模块的流程同创建模块相同。

## 5.9 ES客户端模块

客户端模块封装了以下操作：构造函数（建立连接）、创建表、修改表、查询记录、更新单条记录、插入单条记录、删除单条记录、批量更新记录、批量插入记录、批量删除记录、获取mapping、设置mapping、修改mapping。

# 6.使用说明

## 总体步骤

1.将elasticsearch-jdbc作为jar添加到项目中

2.通过“com.bosssoft.platform.es.jdbc.driver.ESDriver”加载该驱动，如下：

Class.forName("com.bosssoft.platform.es.jdbc.driver.ESDriver");

3.通过DriverManager.getConnection(url)获取连接对象，url格式：jdbc: es://host:port/index

．host：要连接的elasticsearch的主机名或ip

．port：用于传输客户端的可选端口号，elasticsearch默认使用9300

．index：数据库名

如下：要连接的elasticsearch运行在本机，端口是9300，连接的数据库名为demo

Connection con = DriverManager.getConnection("jdbc:es://localhost:9300/ demo");

4.发送sql：可以通过Statement或PreparedStatement来发送sql语句。PreparedStatement可以通过占位符设置可变的sql，如下：

（1）通过Statement

Statement st = con.createStatement();

st.executeUpdate("update mytest set user\_name='Mike' ");

（2）通过PreparedStatement

String sql=" update mytest set user\_name=? ";

PreparedStatement ps = con.prepareStatement(sql);

ps.setString(1, " Mike ");

ps.executeUpdate();

5.释放资源

rs.close();

con.close();

## 6.2 功能操作

### 6.2.1 查询

1. 支持如下的查询sql：

Select … from tableName …...

．COUNT, MIN, MAX, SUM, AVG

．DISTINCT

．WHERE (=, >, >=, <, <=, <>, IN, LIKE, AND, OR, IS NULL, IS NOT NULL, NOT [condition])

．GROUP BY

．HAVING

．ORDER BY

．LIMIT offset,num (offset:开始位置，从0开始；num：获取的记录数) ，没有设置

默认返回前100条

2.获取查询结果：通过遍历ResultSet来获取查询结果

（1）输出所有查询列的结果：从ResultSetMetaData中获取所有列名：如下，输出mytest表中所有的字段值

ResultSet rs = st.executeQuery("select \* from mytest");

ResultSetMetaData metaData=rs.getMetaData();

int ncols=metaData.getColumnCount();

while(rs.next()){

for (int i=1;i<=ncols;i++) {

System.out.println(metaData.getColumnName(i)+": "+rs.getObject(i));

}

}

（2）输出特定列的结果：如下，获取所有的用户名

ResultSet rs = st.executeQuery("select \* from mytest ");

while(rs.next()){

System.out.println("dept\_no:"+rs.getString("name"));

}

### 6.2.2 增删改

1. 支持如下的操作sql：

新增记录：insert into tableName(columns..) values(..,..)

修改记录：update tableName set column1=value1,column2=value2… where …..

删除记录：delete from table where….

2.执行

（1）单条记录：使用Statement的executeUpdate方法，如下：将mytest表中所有记录name字段值改为‘Mike’。

Statement st = con.createStatement();

st.executeUpdate("update mytest set name='Mike' ");

（2）批量：使用PreparedStatement的addBatch()和executeBatch()方法，如下：批量删除mytest表中no为n01和n02的记录

String sql="delete from mytest where no=?";

PreparedStatement ps = con.prepareStatement(sql);

ps.setString(1, "n01");

ps.addBatch();

ps.setString(1, "n02");

ps.addBatch();

ps.executeBatch();

### 6.2.3 创建表

1.sql语句：create table tableName

2.配置mapping：在执行sql语句之前，需要在类路径下的资源文件resource/mapping下创建一个以表名.json命名的配置文件，在配置文件中说明该表的字段、字段属性，配置文件的格式是json格式，格式如下：

{

表名: {

"properties": {

字段名: {

"type": 字段类型

},

}

}

}

3.执行

String sql="create table user";

st.executeUpdate(sql);

### 6.2.4 修改表

1.sql语句：alter table user

2.修改mapping：修改该表对应的配置文件，在，配置文件中新增字段

3.执行

Statement st = con.createStatement();

st.executeUpdate("alter table mytest");

# 7.验证

为了验证ES JDBC的查询效率，对它进行简单的查询性能的测试，并且同mysql进行比较。

1.验证环境：

（1）elasticsearch 服务器运行环境：

cpu: 4核 Intel xeon E321xx 主频：1.8GHZ 内存：15G

（2）mysql 服务器运行环境：

cup：双核 Intel(R) Core(TM) i5-3230M 主频：2.60GHZ 内存：4G

（3）程序运行环境：

cup：双核 Intel(R) Core(TM) i5-3230M 主频：2.60GHZ 内存：4G

2.验证步骤：

（1）在elasticsearch和mysql中创建如下表：user\_info表

表7-1 user\_info表

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| user\_no | varchar | 主键 建主键索引 |
| User\_name | varchar |  |
| User\_salary | double |  |
| Dept\_no | varchar | 建普通索引 |

（2）在数据量为10万、100万、1000万的情况下，分别用ES JDBC和mysql执行如下两种sql：

in 查询：

SELECT user\_no,user\_name from user where user\_salary in (6000.0,2200.0) order by user\_no

简单条件查询：

SELECT \* FROM user where dept\_no='d0'

3.验证结果

（1）in查询结果

表7-2 in查询结果

|  |  |  |  |
| --- | --- | --- | --- |
| 数据量  类型 | 10万 | 100万 | 1000万 |
| ES JDBC | 0.658s | 1.565s | 2.399s |
| Mysql | 0.760 | 2.078s | 198.088s |

（2）简单条件查询结果

表7-3 简单条件查询结果

|  |  |  |  |
| --- | --- | --- | --- |
| 数据量  类型 | 10万 | 100万 | 1000万 |
| ES JDBC | 0.588s | 0.670s | 0.954s |
| Mysql | 0.838s | 2.079s | 501.76s |

4.验证结论：在数据量较小的情况下，mysql的查询速度和ES JDBC相差不多，但是当数据量达到千万级别时，mysql的查询速度远慢于ES JDBC，而ES JDBC基本稳定在3s内。

# 8.总结

经过三个月的努力，终于完成了ES JDBC的开发，在从elasticsearch的学习到ES JDBC的设计最后实现的这段时间里，遇到了不少的困难。首先是自学elasticsearch的难度，对elasticsearch的学习完全是从零开始，而且网上关于elasticsearch java接口的资料比较少，所以经常需要根据HTTP RESTFUL的接口去猜测和验证对应的java接口，elasticsearch不同版本之间接口的差异也为学习增加了难度；其次是信息封装类的设计，特别是where条件的封装，既要方便获取又要准确表达子条件的判断先后顺序，经过反复推敲，修改，最后才确定类的各个变量。

虽然困难重重，但是在老师和公司前辈的指导下，还是顺利完成了开发，而且基本符合功能需求，但是ES JDBC还存在着一些不足，比如由于elasticsearch自身对关联查询的支持较弱，所以本套API的所有操作都只适用于单表；另外，ES JDBC可能还存在一些bug，还需要经过大量的测试才能应用于开发中。

通过ES JDBC的开发，基本掌握了elasticsearch的相关应用，我相信这对我以后的工作会有一定的帮助，但是，仅仅只是掌握它的应用还是不够的，所以在接下来的时间里应该对elasticsearch进行更深入的学习，不能因为课题研究的结束而止步。除了学习了新知识，也巩固了已有的专业知识；最重要的是各方面能力的提高，在一次次解决问题的过程中，不断锻炼自己分析问题、解决问题的能力。

1. 参考文献

   [] XiafeiLei，ZheWang，yuzhenHe3. The Data Management and Real-time Search Based on Elasticsearch[A]. ICCMCEE 2015,2015,823 [↑](#endnote-ref-1)
2. [] 齐治昌，谭庆平，宁洪.软件工程[M].北京：高等教育出版社，2012，161-168 [↑](#endnote-ref-2)
3. [] Erich Gamma，Richard Helm，Ralph Johnson等.设计模式[M].北京：机械工业出版社，2000,63-70 [↑](#endnote-ref-3)
4. [] 陈俊杰，黄国凡. 应用 Elasticsearch 重构图书馆站内搜索引擎[J].情报探索，2014，2014（11），114-115 [↑](#endnote-ref-4)
5. [] PafalKuc，MarekRogozinski.elasticsearch服务器开发[M].北京：人民邮电出版社，2015,4-5 [↑](#endnote-ref-5)