摘要

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2秒内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，这个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API进行二次封装，提供一个更简单的接口，从而简化开发。

## 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，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API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，该API建立在elasticsearch原有的接口之上，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API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本套API。总之，本套API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

本次课题的研究内容如下：

1. 需求分析，通过与开发人员的沟通交流，确定了本套API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，确定本套API采用类似于JDBC的形式
3. 基于高内聚，低耦合的原则，设计出API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elasticsearch API 的开发，该API按照面向接口的开发原则，以建造者模式为主，利用JDBC 提供的接口实现开发。

## 1.5 论文形式

本文由四个章节所组成：

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的应用，相关名词

1. 相关技术介绍

## 2.1 elasticsearch

## 2.2 java

1. 需求分析

## 3.1 技术可行性

本套API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；关键的技术在于如何解析sql语句以及elasticsearch查询响应体。

## 3.2 经济可行性

## 3.3 功能需求分析

本套API应该具有如下功能：

（1）查询

1. 总体设计

## 4.1 结构设计

为了使该API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。结构设计图如下：

更新体

查询体

Sql

Sql

连接对象

连接信息

驱动程序获取对elasticsearch的连接

控 制 器

更新指挥器

查询指挥器

Elasticsearch 服务器

调 用 端

结果

结果

Sql

结果

图 4-1 API 结构图

## 4.2 功能模块设计

Elasticsearch API

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 4-2 API 功能模块

ES客户端模块

驱动模块

固定sql模块

动态sql模块

Elasticsearch API 各模块功能如下：

1. 驱动模块：

解析连接信息，建立对elasticsearch的连接

1. 控制模块：

整个API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

1. 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。

1. 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

1. 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

1. 更新模块

解析sql语句，更新某些记录的值；支持批量更新

1. 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

1. 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

1. 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

|  |  |  |
| --- | --- | --- |
| 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查询体所需要的信息，是查询体构建阶段的产物

|  |  |  |
| --- | --- | --- |
| 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

|  |  |  |
| --- | --- | --- |
| 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

|  |  |  |
| --- | --- | --- |
| 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

|  |  |  |
| --- | --- | --- |
| 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

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

1. 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。

判断URL格式是否合法

调用端发送获取连接的请求

抛出异常

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

## 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

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

是

是

是

是

是

否

否

否

否

否

（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

## 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：

|  |  |
| --- | --- |
| 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

否

否

否

否

是

是

是

是

是

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

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

返回查询对象

1. 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

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

否

返回QueryBody对象

1. 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

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

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

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

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

## 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

## 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

## 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

1. 使用说明
2. 总结

## 7.1 难点分析

## 7.2 心得