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】本文以 elasticsearch 在 java 开发中的应用为研究方向，对 elasticsearch 现阶段的应用情况以及未来的发展趋势进行了研究，通过将 elasticsearch 和关系型数据库进行对比，分析 elasticsearch 在大数据时代下的优势，同时对 elasticsearch java API 的使用以及 JDBC 接口封装技术进行深入学习。以简化 elasticsearch java 开发、降低 elasticsearch 使用门槛为目的，对 elasticsearch java API 进行二次封装，开发了 Elasticsearch JDBC（以下简称 ES JDBC）。ES JDBC 提供了对数据的增删改查以及表的创建修改，以类似于 JDBC 的形式呈现调用接口。本文遵循软件工程的思想，对 ES JDBC 进行了需求分析，从而确定功能模块，在此基础上，给出 ES JDBC 的整体结构以及各个功能模块的详细设计，最后对本次课题的难点以及存在的问题进行分析总结。本次开发平台为 eclipse,采用 maven 框架，所依赖的 jdk 版本是 jdk1.8。

【关键词】elasticsearch; java; jdbc; 接口封装

目录

1. 引言 1
1.1 课题背景 1
1.2 研究现状及发展趋势 1
1.3 课题意义与目的 1
1.4 研究内容 1
1.5 论文形式 2
2. 相关技术介绍 2
2.1 ELASTICSEARCH 2
3. 需求分析 3
3.1 技术可行性 3
3.2 经济可行性 3
3.3 功能需求分析 3
4. 总体设计 4
4.1 总体流程 4
4.1.1 查询 sql 流程 4
4.1.2 更新 sql 流程 5
4.2 功能模块设计 6
4.3 信息封装类设计 6
5. 详细设计与实现 7
5.1 驱动模块 7
5.2 控制模块 8
5.3 查询模块 9
5.4 添加模块 11
5.5 删除模块 12
5.6 更新模块 13
5.7 创建模块 13
5.8 修改模块 14
5.9 ES 客户端模块 14
6. 使用说明 14
6.1 前期数据准备 14
6.2 总体步骤 14
6.3 功能操作 15
6.3.1 查询 15
6.3.2 增删改 16
6.3.3 创建表 16
6.3.4 修改表 17
7. 验证与对比 17
7.1 对比 17
7.2 验证 18
8. 总结 19

1. 引言

1.1 课题背景

在互联网产品的开发中，数据存储方式在一定程度上影响着产品的性能，现如今，大多数的互联网产品使用 oracl 等关系型数据库来存储数据。然而在使用人数不断增加以及全国大联网的形势下，常规的这些数据库已经无法为产品提供良好的性能，但是用户对产品的性能要求却越来越高，它们必须能够快速地将请求信息返回给客户端，很显然，在海量数据的情况下，使用常规的数据库根本无法达到这样的业务要求，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，比起 oracle 等数据库，elasticsearch 在查询性能上有明显的优势，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。

1.2 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]，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。

1.3 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。

1.4 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]，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]，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]。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]。

Elasticsearch 之所以有高效的搜索性能，很大一部分原因是它在存储数据的时候使用了倒排索引。但是，elasticsearch 无法直接为文档建立索引，所以，elasticsearch 使用分词器将文档分割成多个词,再为这些词创建倒排索引,倒排索引可以简化成一张表,这张表记录每个词分别出现在哪些文档中。比如有两个文档：

文档 1：{"name":"mike","title":"my school"}

文档 2：{"name":"jim","title":"my country"}

这两个文档经过分词器的分割后就产生了这些词：mike、jim、my、school、country，倒排索引记录这些词出现的次数以及在哪些文档中出现，如表 2-1：

表 2-1　倒排索引

词	次数	文档
name:mike	1	1
name:jim	1	2
title:my	2	1,2
title:school	1	1
title:country	1	2

elasticsearch 将词列保存成词文件，将次数列保存成频率文件，将文档列保存成位置文件。elasticsearch 中有一张哈希表，每个表项都是指向词文件的指针，而每个词文件也会有两个指针，分别指向对应的频率文件和位置文件，如下图：

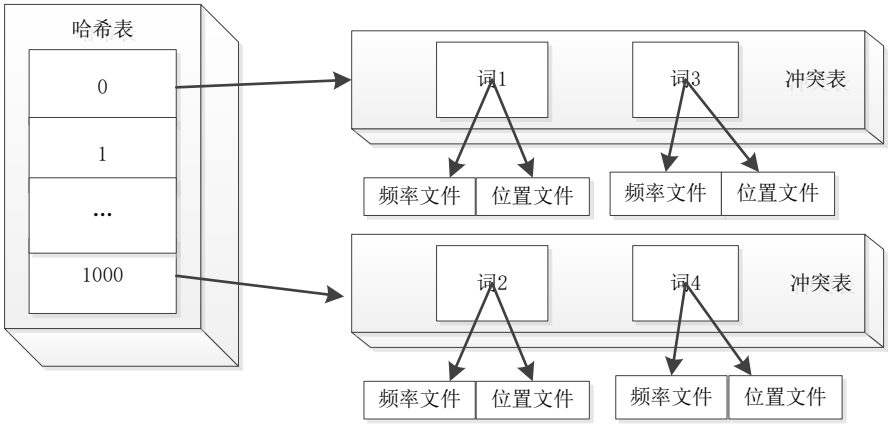

图 2-1　elasticsearch 索引文档

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

查询 sql 流程如图 4-1：

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。

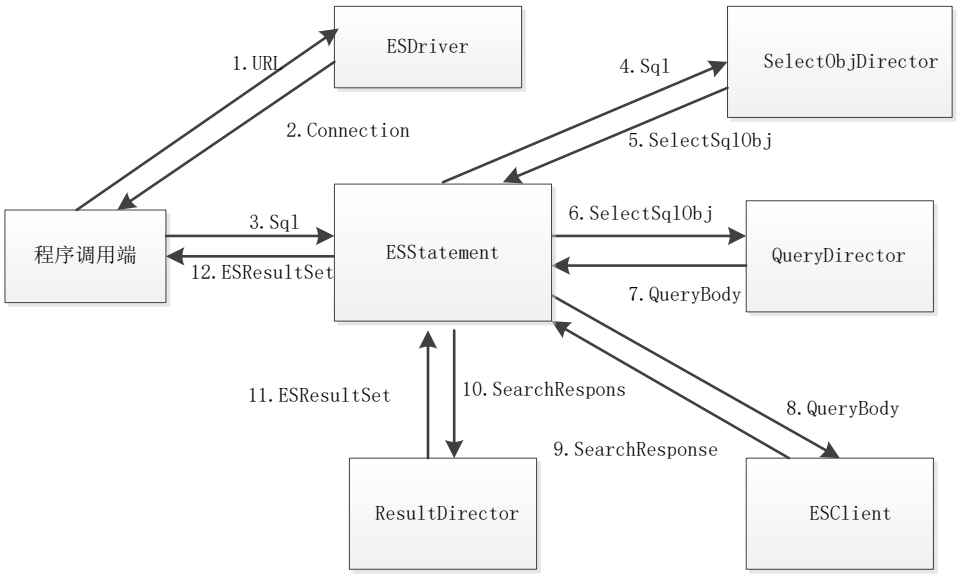

图 4-1　查询 sql 流程

4.1.2　更新 sql 流程

更新 sql 流程如图 4-2：

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。

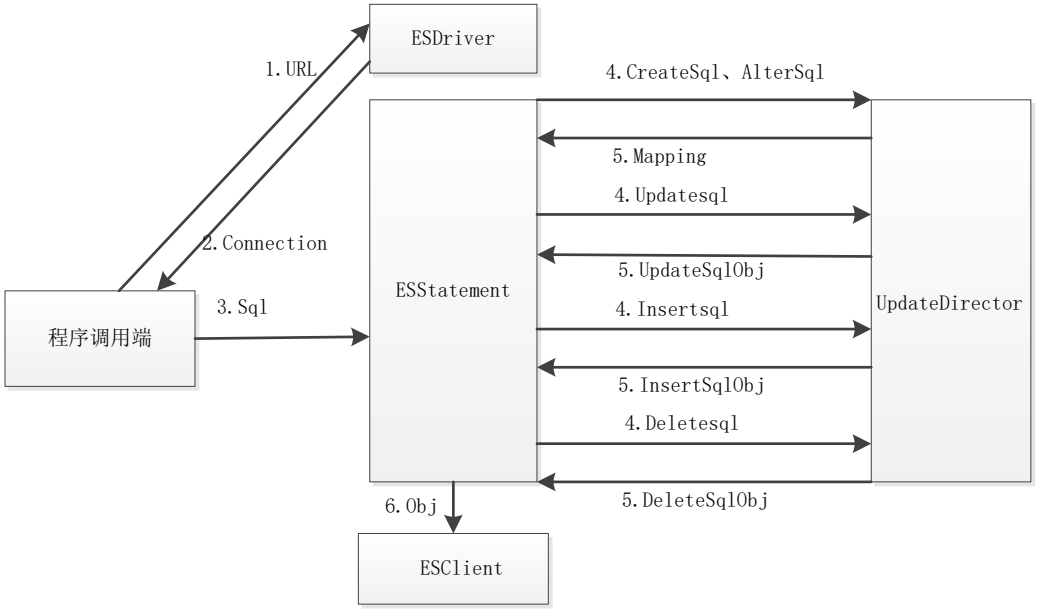

图 4-2　更新 sql 流程

4.2 功能模块设计

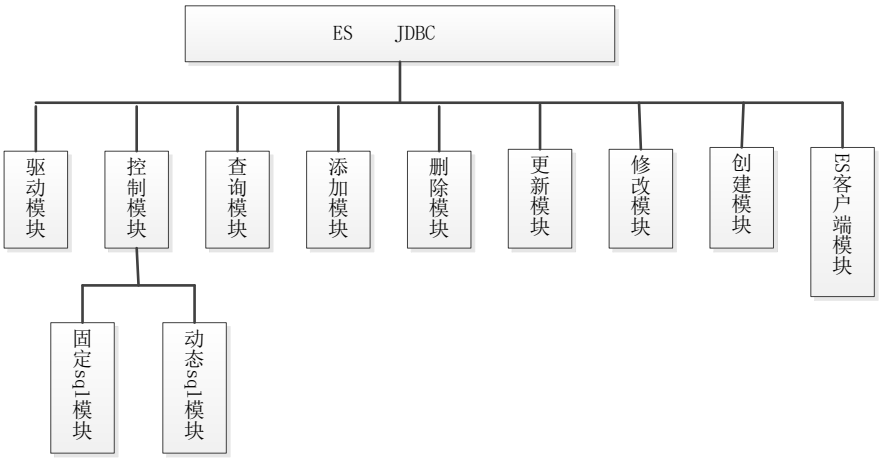

图 4-3 功能模块

ES JDBC 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。

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 为自定义类，包含列名，别名，聚合操作类型
from	String	表名
Where	ConditionExp	查询条件，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 为自定义类，包含排序字段，排序类型
limit	PageMate	分页信息；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；elasticsearc 类
aggregationBuilder	AggregationBuilder	聚合信息；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。如图 5-1，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。

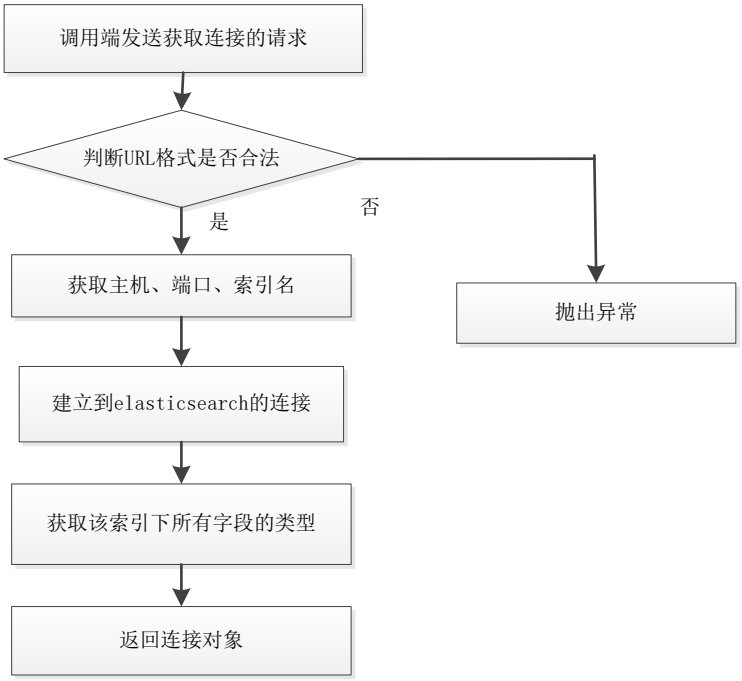

图 5-1　驱动模块流程

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；如图 5-2，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。

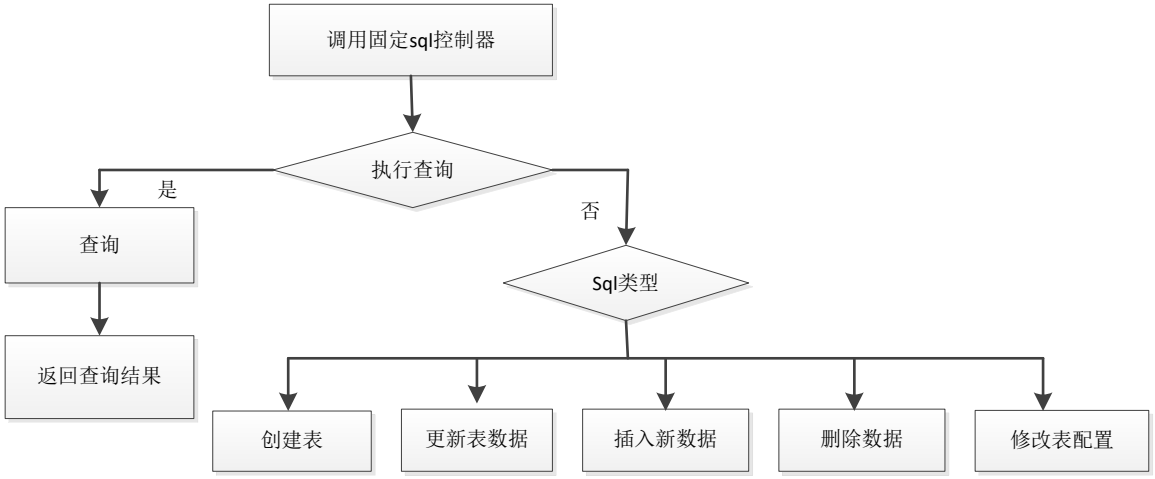

图 5-2　固定 sql 控制器流程

2.动态 sql 控制模块：该模块实现 JDBC 的 PreparedStatement 接口；如图 5-3，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。

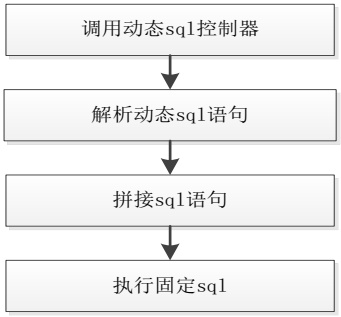

图 5-3　动态 sql 控制器流程

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：如图 5-4，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。

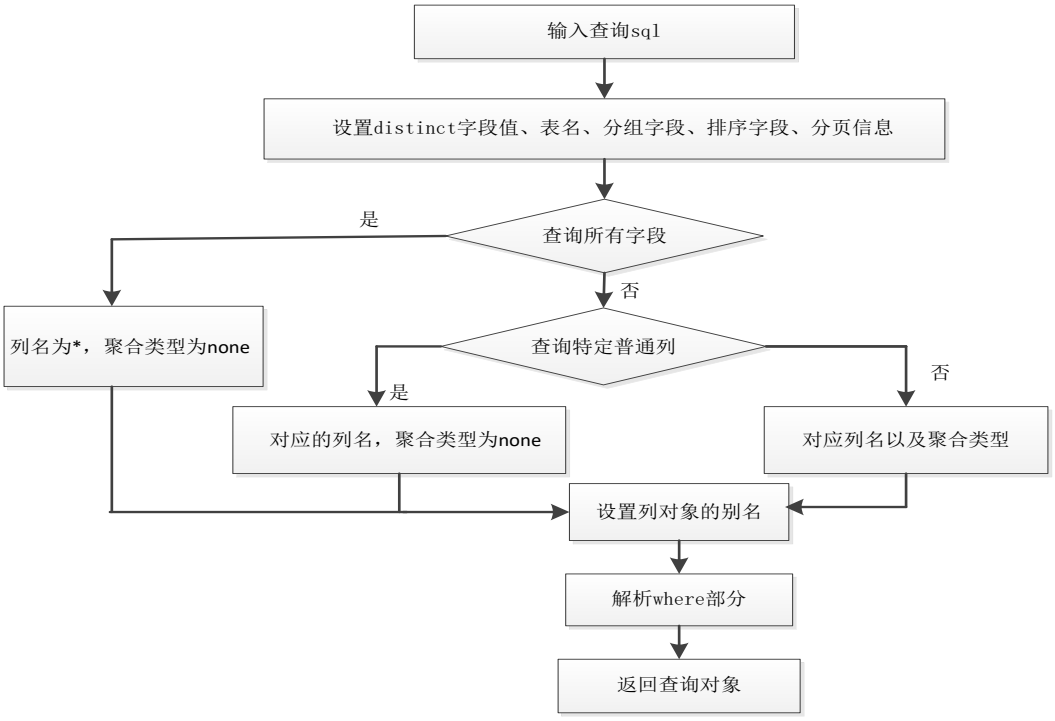

图 5-4　sql 解析流程

2.查询构建阶段:如图 5-5,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，否则，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。

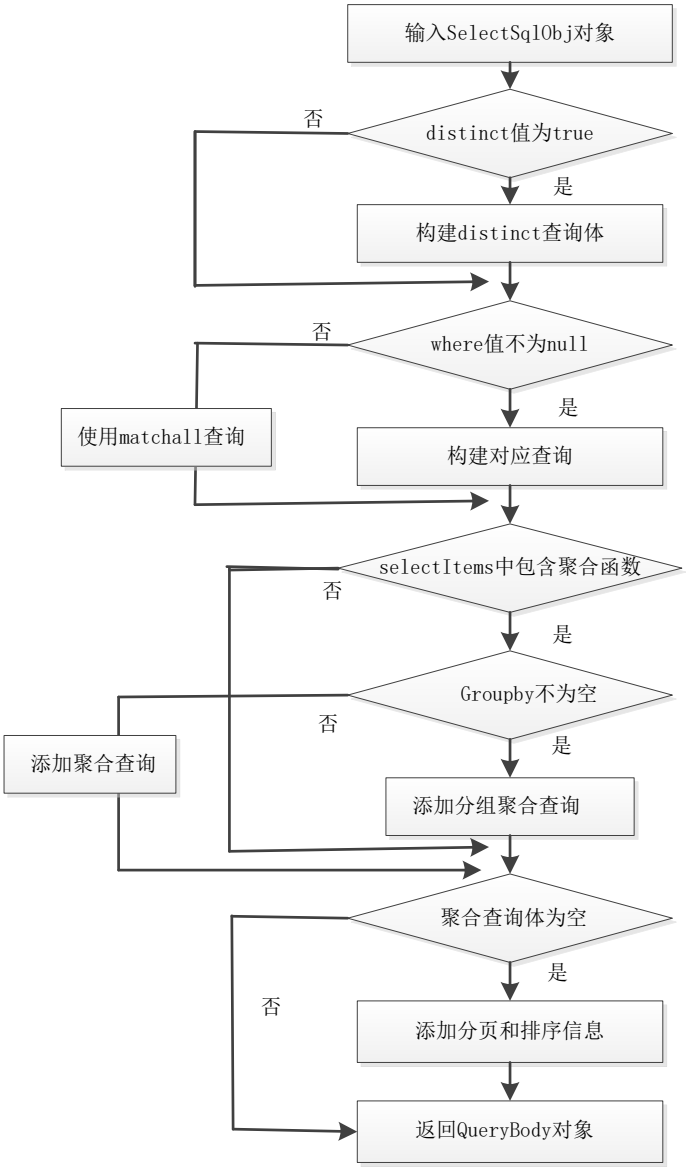

图 5-5　查询体构建流程

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:如图5-6,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：如图 5-7，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。

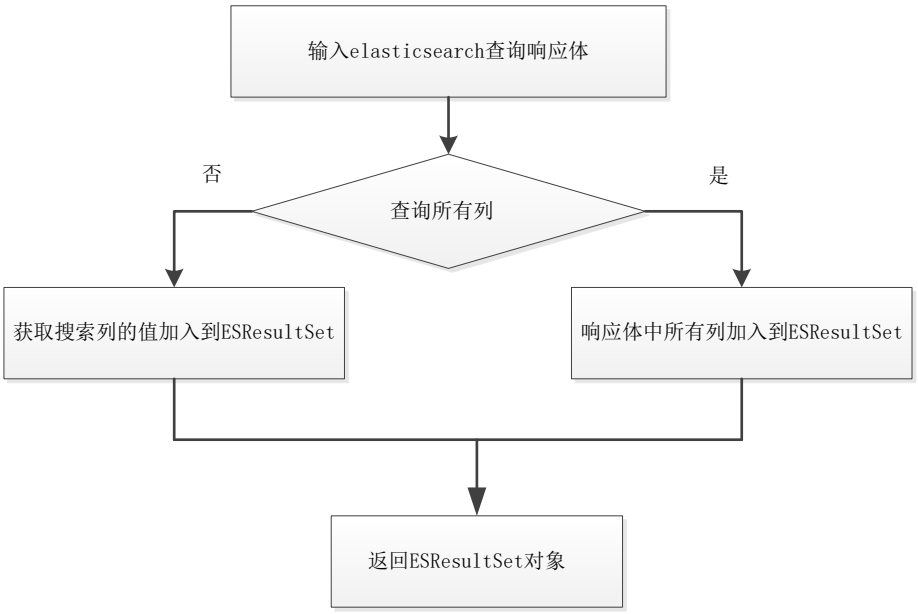

图 5-6　条件查询结果集构建流程

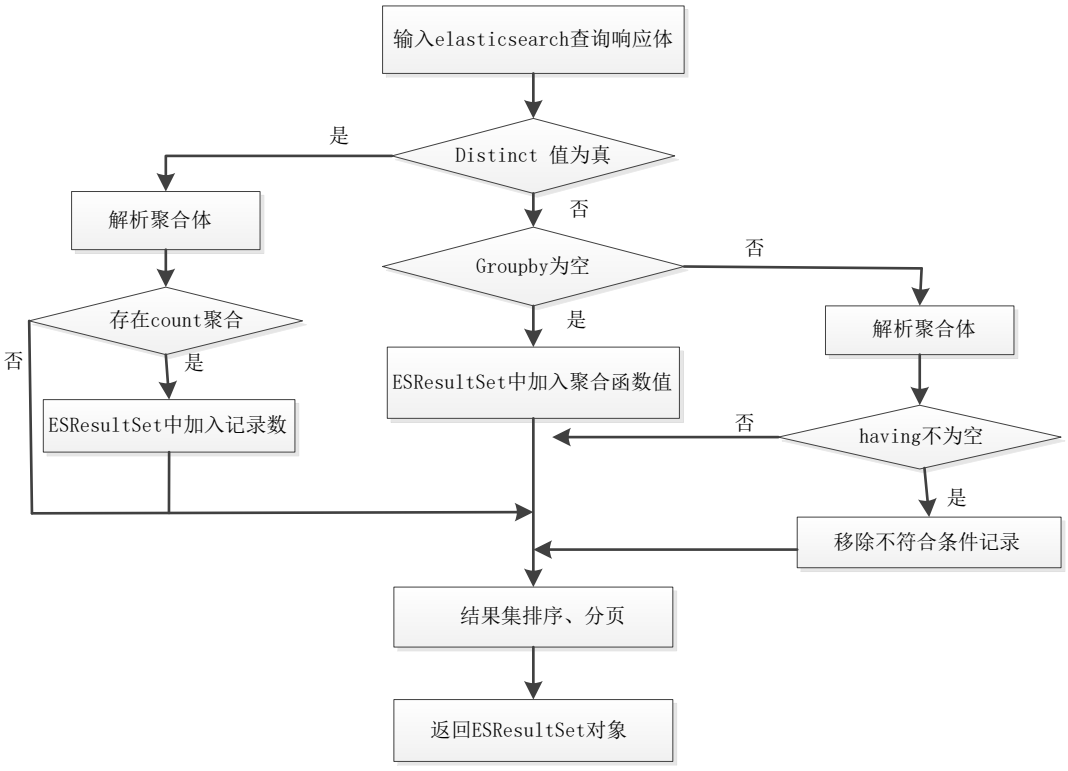

图 5-7　聚合查询结果集构建流程

5.4　添加模块

如图 5-8，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。

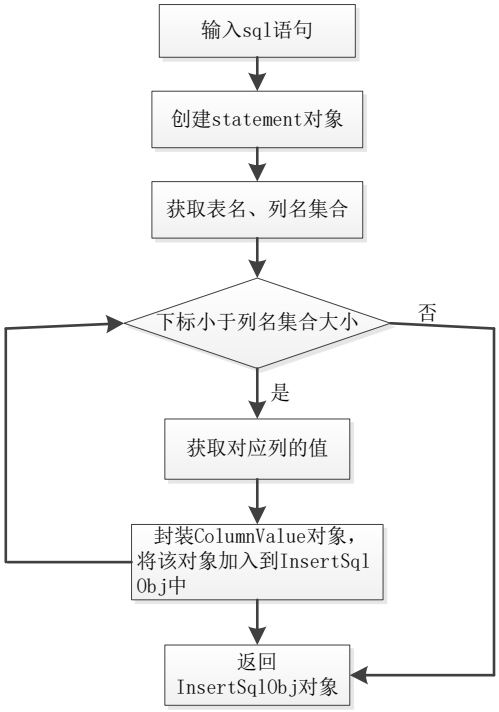

图 5-8　新增数据流程

5.5　删除模块

如图 5-9，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。

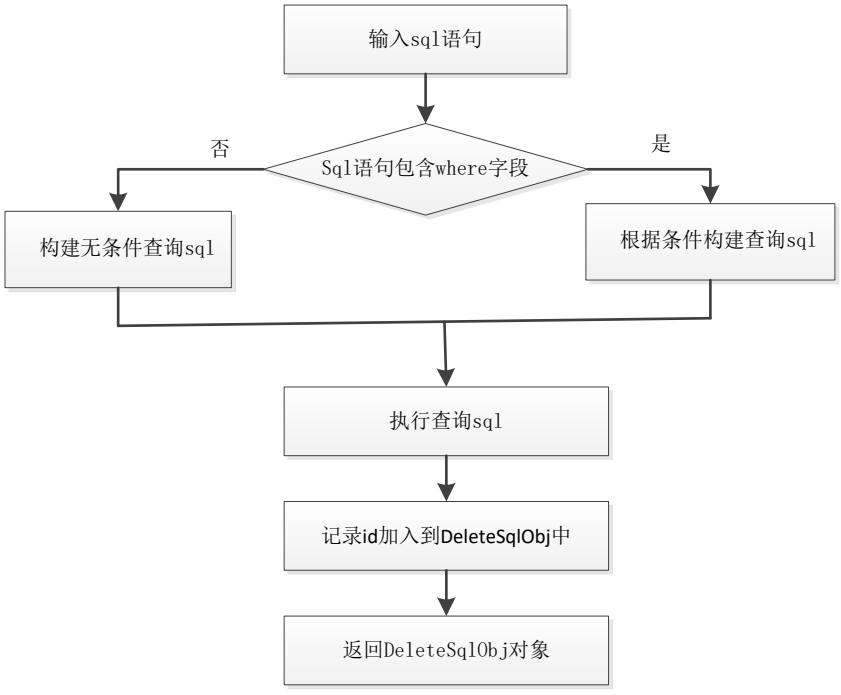

图 5-9　删除数据流程

5.6 更新模块

如图 5-10，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。

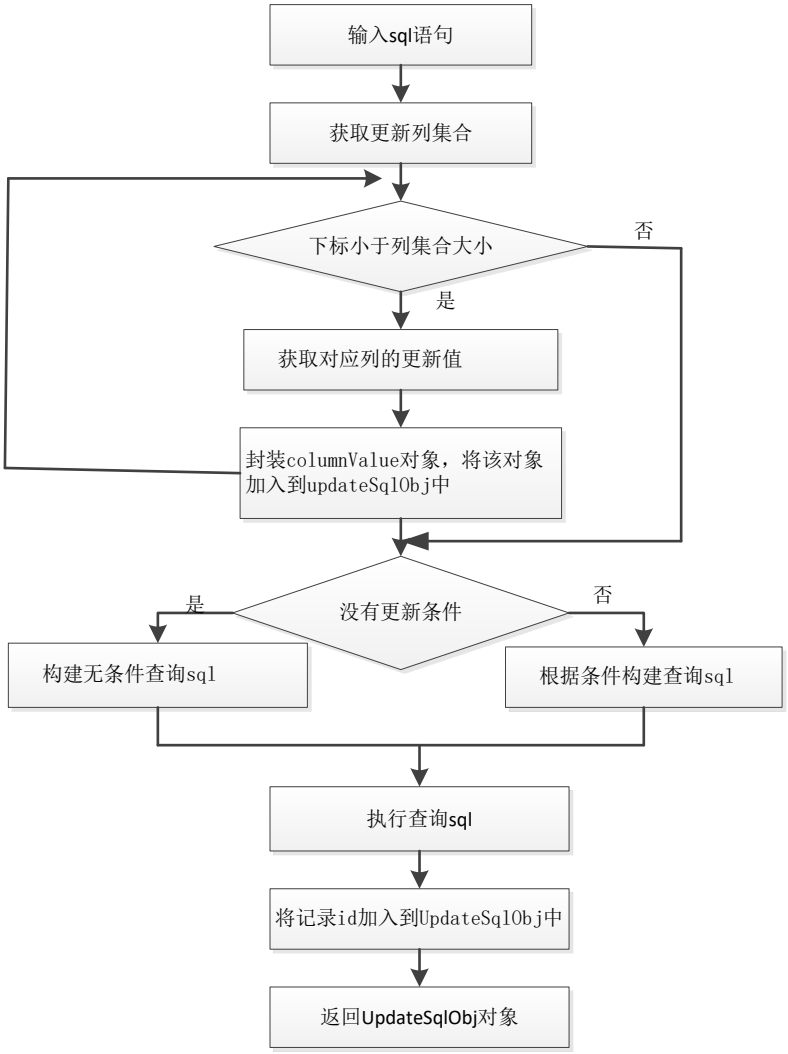

图 5-10 更新数据流程

5.7 创建模块

如图 5-11，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，所以用配置文件的方式为表设置 mapping。

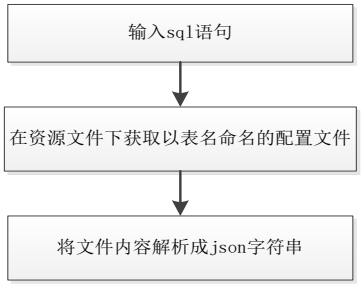

图 5-11 创建表流程

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

6.1 前期数据准备

要将关系型数据库中的数据导入 elasticsearch，需要借助一个 elasticsearch 的插件—elasticsearch importer，这个插件以脚本文件的形式运行，通过执行 sql 语句从数据库中获取数据，然后将获取的数据导入 elasticsearch，脚本文件的格式如下：

```
{
    "type" : "jdbc",
     "jdbc" : {
          "url" : "jdbc:mysql://localhost:3306/db",     //数据库的连接 URL
          "user" : "root",       //数据库的用户名
          "password" : "123",//数据库密码
          "sql" :"",              //要执行的 sql 语句
           "index" : "demo"    //elasticsearch 索引名
          }
}
```

比如要导入 user 表的数据，则 sql 语句为：selelct * from user。ES JDBC 不支持关联查询，所以有关联关系的多张表要先通过这个插件的 sql 语句将数据进行拼接。

6.2 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。

（1）通过 Statement

```
Statement st = con.createStatement();
st.executeUpdate("update mytest set user_name='Mike' where user_no=100");
```

（2）通过 PreparedStatement

```
String sql=" update mytest set user_name=? where user_no=?";
PreparedStatement ps = con.prepareStatement(sql);
ps.setString(1, " Mike ");
ps.setInt(2,100);
ps.executeUpdate();
```

5.释放资源

```
rs.close();
con.close();
```

6.3 功能操作

6.3.1 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)，如果没有设置，默认返回 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：

```
Statement st = con.createStatement();
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

使用 ResultSet 的 get 方法获取指定的列，如下获取 name 字段的值：

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from mytest ");
 while(rs.next()){
        System.out.println("name:"+rs.getString("name"));
  }
```

6.3.2 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();
```

6.3.3 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);
```

6.3.4 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. 验证与对比

7.1 对比

为了说明使用 ES JDBC 的简便性，以获取连接和查询为例，将程序调用端使用 ES JDBC 的流程和使用 elasticsearch API 的流程进行对比。

1. 获取连接：如图 7-1，使用 elasticsearch API，程序调用端需要调用多个复杂的类，分别将连接信息传递给这些类才能获取客户端连接对象；而对于 ES JDBC，如图 7-2，DriverManager 封装了创建复杂类的过程，只需要调用 DriverManager 即可。

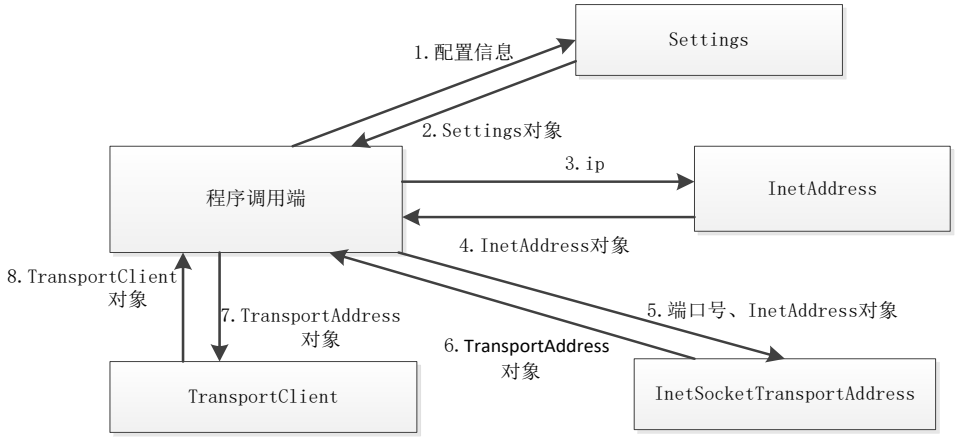

图 7-1 elasticsearch API 获取连接使用流程

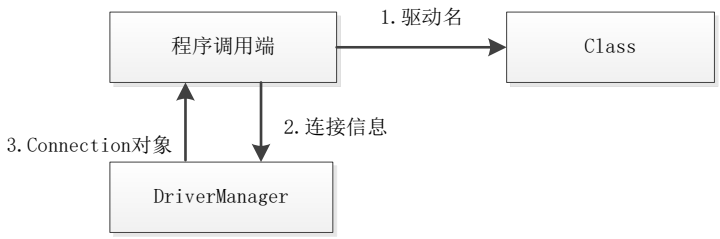

图 7-2 ES JDBC 获取连接使用流程

2.查询：如图 7-3，若是使用 elasticsearch API，查询的每个匹配条件和聚合条件都需要分别创建一个 AggregationBuilders 和 QueryBuilder 对象；而对于 ES JDBC，如图 7-4，只要将条件以 sql 形

式传递给 Statement 即可。

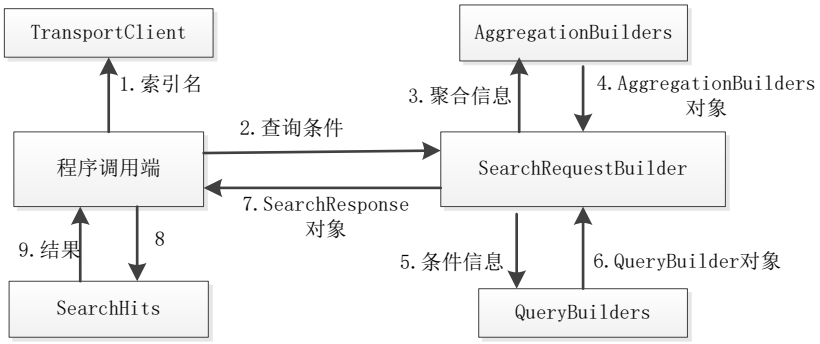

图 7-3　elasticsearch API 查询使用流程

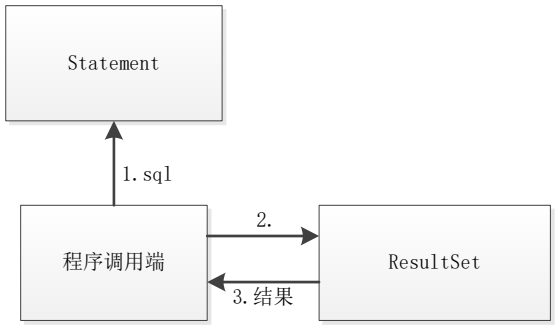

图 7-4　ES JDBC 查询使用流程

7.2　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：

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

（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	创建普通索引

（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 count（*）from user where user_salary in (6000.0,2200.0)
```

简单条件查询：

```
SELECT count（*）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	0.698s	0.745s
Mysql	0.421s	6.040s	196.088s

（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	0.637s	0.659s	0.805s
Mysql	0.408s	0.439s	1.082s

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，如果 mysql 没有建索引或者索引无法发挥作用则查询速度远慢于 ES JDBC，而 ES JDBC 基本稳定在 1s 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。

参考文献

[1] XiafeiLei，ZheWang，yuzhenHe3. The Data Management and Real-time Search Based on Elasticsearch[A]. ICCMCEE 2015,2015,823

[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

[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

[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

[5] PafalKuc，MarekRogozinski.elasticsearch 服务器开发[M].北京：人民邮电出版社，2015,4-5

Encapsulation of elasticsearch operation interface based on Java

HUANG Xue-wen 105032013072 Advisor：ZHANG Da-ping
Major in Computer Science and Technology School of Mathematics and Computer Science

【**Abstract**】This article studies the application of elasticsearch and its development trend in the future based on the application of elasticsearch in java development,analysing the advantages of elasticsearch in the era of big data through making comparison with relational database and studying about the using of elasticsearch java API and the technology of packaging JDBC interface.It developed Elasticsearch JDBC (hereinafter referred to as ES JDBC) by packaging elasticsearch java API in order to simplify the using of elasticsearch.ES JDBC encapsulates the basic operation of elasticsearch including adding,deleteing,modifing,searching data and the creating,altering table.Its interface is presented in a form similar to JDBC.This article making demand analysis for ES JDBC,thus determing the function module and the overall structure of ES JDBC,including detailed design of each functional module.Last it analyzes the difficulties and existing problems about this work. The development platform is eclipse,using the Maven framework, the JDK version is dependent on JDK1.8.

【**Keywords**】elasticsearch;java;jdbc;the encapsulation of interface