基于Elasticsearch的纪检数据搜索引擎的设计

摘要

# 第1章 绪论

## 1.1 研究背景

随着信息技术的迅速发展,各行各业的数据量呈爆发性的增长，人们对信息检索的需求越来越强烈，如何从海量数据中高效、准确的挖掘出人们需要的信息[1]成为热议的学术话题，搜索引擎由此应运而生。

传统搜索引擎的服务对象是是大众用户，随着信息的急剧扩张，其采集收录的信息内容不断扩大，搜索结果也逐渐覆盖到各行各业，无法为某一专业领域内的用户提供专一化的信息查询[2]。因此，近些年来，人们又对搜索引擎提出了新的需求，要求可以针对某一领域检索更为专业的信息，这就需要更加专业化和更具针对性的搜索引擎，由此发展出了面向特定领域的垂直搜索引擎[3]。

垂直搜索引擎是搜索引擎发展过程中的一个重要里程碑，其只收录特定专业领域的信息数据，向其提供深入、垂直的搜索功能，不属于该领域的信息均不收入。相比通用搜索引擎，垂直搜索引擎大大降低了搜索过程中无关信息的干扰现象的发生，提升了搜索的效率。同时，与通用搜索引擎使用网络爬虫采集数据的策略相比，垂直搜索引擎通常具备专门的信息获取方式，如直接从该领域数据库中采集数据，这不仅降低了数据更新的成本，也使得数据更新的周期大幅缩短[4]。

## 1.2 研究意义

社会信息化的不断发展，使我国的纪检监察信息化的建设取得了快速进步。纪检部门以“大数据”理念为指导，加大信息资源的整合力度，汇总其直属及各地方机构的信息纳入其监察体系，使得像过去“跑部门、做调研、要信息”的情况逐渐减少。这对实现数据资源的规范查询、问题线索的追踪溯源具有重要意义。

随着数据量级的不断增长，于纪检人员来说，如何高效地从海量数据中获取自己需要的信息，如何高效的针对一条问题线索顺藤摸瓜、举一反三，成为纪检人员的关注重点。

搜索引擎是数据检索的利器。因此，针对纪检数据检索的需求，我们以沈阳市纪委为实验对象，利用Elasticsearch，针对其纪检监察中的民生资金、招投标项目数据设计了搜索模块。目前，系统已稳定上线近一年，极大方便了纪检人员在以上领域的查案效率。

截至当前，沈阳市纪委拥有从市、区县（市）两级共计835家政府职能部门[5]采集的数据22亿余条，涉及数据库表总量达上千余个。数据的进一步扩充向我们提出了新的挑战：面对数千数据库数万数据表的大规模结构化数据，如何设计搜索引擎的索引使其可以高效的容纳这些数据，如何组织搜索使得搜索引擎可以兼顾性能与准确率，如何设计搜索结果的展示方式使其在大规模库表下仍保持保持良好的易读性，这都是需要考虑的问题。

面对这样的挑战，需要对原有系统的架构进行重新设计，使其能够兼容大规模的结构各异的数据。同时，针对数据扩充造成的系统运维管理不便的问题，还需设计一个便捷的搜索引擎管理工具，将复杂的索引操作向用户屏蔽，提供易用的交互界面供用户操作，使系统成为一个可以产品化的系统。

## 1.3 主要研究内容

本文旨在通过对搜索引擎相关技术的学习，结合纪检数据的特点与需求，利用Java、Elasticsearch等技术设计一个纪检监察大数据垂直搜索引擎系统。系统提供了数据字典的管理、搜索引擎的管理、数据同步、数据搜索与展示等功能，并提供友好的交互界面简化对上述模块的管理。同时，利用Spring Cloud技术，将系统以微服务的形式进行功能划分，使其成为一个高可用、高性能、高扩展性的分布式搜索引擎系统。

## 1.4 论文组织结构

本论文章节安排如下：

第一章 绪论：本章介绍了本文的研究背景、意义及主要研究内容。

第二章 搜索引擎概述：

第三章 纪检大数据搜索引擎系统需求分析

第四章 纪检大数据搜索引擎系统架构

第五章 纪检大数据搜索引擎系统详细设计

第六章 总结与展望

、 第七章 致谢

# 第2章 相关技术简介

## 2.1 搜索引擎概述

### 2.1.1 搜索引擎的概念与发展历程

搜索引擎(Search Engine)是一种可以从互联网或其他媒介上搜集信息，按照一定的策略将信息进行组织和存储，并根据用户的需要向用户提供检索服务，将相关的内容排序后展示给用户的系统。搜索引擎是用户获取数据的核心工具。

互联网的蓬勃发展，使得各种网络信息资源不断产生，其中最主要的有WWW(World Wide Web)信息资源、FTP(File Transfer Protocol)信息资源以及Blog信息资源等。这些资源具有信息量大、类型多样、分散无序的特点，为了满足对这些信息的检索请求，搜索引擎开始出现并发展起来。

* Archie——搜索引擎的鼻祖

Archie是由加拿大麦吉尔大学(McGill University)的三位学生Alan Emtage、Peter Deutsch、Bill Wheelan在1990年发明。在当时，WWW(万维网)还没有出现，人们主要通过FTP服务来共享文件，而大量的FTP服务器使得很多文件散布在多个不同的地方，文件的查询变得十分不方便。因此，Alan Emtage等三人便提出了一种可以直接查询出文件所在FTP地址的工具Archie。Archie可以自动对互联网上的FTP文件信息进行索引存储，用户通过相应文件的名称即可查询到该文件的FTP地址。Archie被称为所有搜索引擎的祖先。

* Gopher——菜单式搜索工具

1991年，在Archie的启发之上，美国内华达大学(University of Nevada)的“系统计算服务”小组发明了Gopher搜索工具。这是一种菜单式的信息浏览工具。用户选择某项菜单时，系统会将该项菜单的查询结果在用户终端上显示出来。如果要查询的信息在当前服务器上不存在，系统则会自动向另一台服务器发送相同的查询请求，直至检索到用户所需的信息。

* Yahoo Directory——目录式搜索引擎的代表

1994年，两位美国斯坦福大学（Stanford University）的博士生创办了Yahoo公司。Yahoo Directory是目录式搜索引擎的一种，其通过人工手动筛选，将各种类别的网站收集在一起。用户可以通过树形的分级目录逐级查找自己感兴趣的网页，也可以通过关键词的形式检索相关的网页。当以关键词形式检索时，系统会根据分类目录及网站与关键词的匹配度对网站进行排列，匹配度高的排在前面；当以目录形式检索时，系统会以字典序对网站进行排列。

* Metacrawer——元搜索引擎

1995年，美国华盛顿大学(University of Washington)的两名硕士生[Eric Selberg](http://web.archive.org/web/20010407110524/www.cs.washington.edu/homes/speed/home.html)和[Oren Etzioni发明了Metacrawer。这是一种新的搜索引擎形式——元搜索引擎(Meta Search Engine)。Metacrawer共支持12个独立的搜索引擎。当用户提交查询请求时，Metacrawer会将该次请求转换处理后提交给12个搜索引擎系统，并将这些系统的搜索结果经过汇总、筛选、排序后返回给用户。](http://www.cs.washington.edu/homes/etzioni/)

* Google——Web搜索引擎巨头

Google是Web搜索引擎的代表，其源自拉里·佩奇(Larry Page)的一个研究项目——BackRub。1998年，拉里·佩奇和另一位Google创始人[谢尔盖·布林](https://baike.baidu.com/item/%E8%B0%A2%E5%B0%94%E7%9B%96%C2%B7%E5%B8%83%E6%9E%97" \t "_blank)(Sergey Brin)提出了PageRank(网页排名)算法，这是一种分析网页的重要性的算法，其以网页之间超链接的个数和链接的网页的质量作为主要评估因素对网页进行排名。1998年9月，Google以私营企业的形式创立。2004年8月，Google在纳斯达克正式上市。现如今，Google已成为最流行的搜索引擎系统之一。

### 2.1.2 搜索引擎的分类

搜索引擎按照其工作原理的不同主要可以分为三种类型，分别为目录搜索引擎(Directory Search Engine)、全文搜索引擎(Full Text Search Engine)和元搜索引擎(Meta Search Engine)。

1. 目录搜索引擎

目录搜索引擎是一个按照目录分类的网站链接列表，其数据库大多是由专职编辑人员来建立和维护。编辑人员在查看某个Web网站后人工撰写一段该网站的概要，并将该网站的地址和概要信息置于一个事先确定的类别中。用户可以通过关键词检索到相关的网页，也可以通过分类目录的形式逐级的找到自己需要的信息。国内的新浪、搜狐以及国外的雅虎(Yahoo)等都属于目录搜索引擎。

2. 全文搜索引擎

全文搜索引擎是目前使用最广泛的搜索引擎，其代表有国外的Google搜索以及国内的百度、搜狗搜索等。全文搜索引擎通过爬虫等信息采集技术获取数据，使用倒排索引的数据结构将数据存储在磁盘，其能迅速地检索出与用户查询条件相符合的记录，并将搜索结果按照相关度排序后返回给用户。

3. 元搜索引擎

元搜索引擎本身并不存储数据，它相当于多个搜索引擎的代理。当用户向元搜索引擎发送查询请求时，元搜索引擎会将查询请求经过处理后同时发给其预先支持的多个搜索引擎，然后将这些搜索引擎的返回结果经过处理、筛选和排序后再返回给用户。图2.2.1展示了元搜索引擎的基本查询步骤。



图 2.2.1 元搜索引擎查询步骤

### 2.1.3 通用搜索引擎与垂直搜索引擎

目前我们接触的百度、谷歌等Web搜索引擎都属于通用搜索引擎，其主要特点数据资源丰富，搜索结果覆盖面广泛，但正式由于其包罗万象的特点，在使用过程中，通用搜索引擎也面临许多需要解决的问题：

1.无法满足某一专业领域的个性化查询需求[2]：由于通用搜索引擎的服务对象是大众用户，其收录的数据源十分广泛，因此搜索结果通常会涉及到各行各业，导致数据的查准率较低，用户很难在其中找到自己所属领域的信息。

2.实时性差： 由于互联网数据规模的日益庞大，数据类型的日新月异，通用搜索引擎在技术上很难做到与互联网信息的及时同步，这使得搜索引擎的实时性变得很低。

3. 成本高：数据的迅速扩张使得搜索引擎对存储设备的需求越来越大，同时，数据的索引、检索和排序也需要消耗大量的计算资源。这导致通用搜索引擎的成本开销愈来愈高。

针对通用搜索引擎针对性差、实时性低、成本较高的缺陷，近些年来，面向某一专业领域或学科的垂直搜索引擎逐渐出现并发展起来。垂直搜索引擎，又称主题搜索引擎。与通用搜索引擎同，垂直搜索引擎只针对某一专门的领域或某一特定的需求提供定制化的信息检索服务[5]，其只收录特定专业领域的信息数据，向其提供深入、纵向的搜索功能，该领域外的信息均不收录。常见的垂直搜索引擎有面向商品的小商品搜索引擎、面向新闻、歌曲的搜索引擎等等。

## 2.2 Elasticsearch简介

# 第3章 系统需求分析

## 3.1 纪检数据的特征

（1）数据来源广泛

纪检业务的数据来源十分广泛。为了满足纪检业务的需要，纪检部门通常会收集汇总及其直属及下属各地区各部门的电子政务数据，而下属部门的数据又是汇总自其下属乡镇、街道委员会相关单位，整个数据来源根据职能单位的级别近似形成了一个树形的结构。以沈阳市纪委为例，截至目前，该部门拥有从市、区县（市）两级共计835家政府职能部门[1]采集的数据22亿余条，涉及数据库总量达上千余个。这些数据通过人工拷贝、专网接口、专线网闸等方式流入纪委内部数据系统。

（2）数据类型多样

纪检数据汇集了多类异构的数据，其既有关系性数据库的结构化数据，也有类似XM形式的半结构化数据以及图片等非结构化数据，数据类型多种多样。以沈阳市纪委为例，其现有数据的类型涉及MySQL、Oracle、Access、Microsoft SQLServer等几乎市面上常用的各种数据库以及Excel、Word、GIS、报表等非结构化数据。

（3）数据结构散乱

大多数政务部门并不具备传统互联网公司那样大规模且专业的技术团队，其数据库表、字段的命名方式多种多样，例如使用拼音首字母、使用字母数字编码等等，这给阅读数据的人员带来很大的阻碍。而且，很多数据都是由Excel汇总导入至数据表，由于Excel在数据校验方面的不足，使得大量数据未经检查便被导入数据库，数据的格式不能保证统一。例如“2019-03-01”写法有“2019年3月1号”、“3-1-2019”、“二零一九年三月一日”等等。这些数据在存储时，便不能按照数字类型、日期类型等去存储，只能使用字符串类型，这也增加了数据错误、数据重复现象出现的概率。

（4）数据更新方式单一

## 3.2 搜索引擎在纪检业务中的应用场景

## 3.3 本系统需要解决的问题

### 3.3.1 数据索引——大规模库表下的Elasticsearch索引设计

Elasticsearch的索引为分布式的分片结构，每个分片相对独立，因此搜索请求会广播到索引的每个分片，每个分片并发的查询自己分片内的数据。各分片的查询结果有Elasticsearch汇总后返回给用户。

在传统的互联网应用中，索引的设计方式多为针对每个实体类（数据表）建立一个索引，索引各字段的名称与实体类（数据表）中的字段名称一一对应。这种查询方式适合单索引的查询，即每次查询请求都是在某一个具体的索引下进行。

而纪检数据面对的是成千上万的数据表，且表与表之间的结构、字段各不相同。查询方式为全库全文搜索，即每次查询针对的是所有库表所有字段进行搜索。在这种情况下，若使用传统的“一表一索引”的方式，假设库中有5000个数据表，每个表2个分片。当查询请求到来时，Elasticsearch会将请求广播至10000个分片中，这10000个分片会去一同竞争使用处理器的资源，使得处理器将大量时间花费在线程上下文的过程中，从而严重拖垮处理器及Elasticsearch的性能。

面对这样的情况，需要设计高效、通用的索引结构，使其不仅能够容纳不同类型的数据库表，还要确保高效的数据查询效率。

### 3.3.2 数据同步——非规范数据表与搜索引擎之间的数据同步

纪检数据具有数据库表多、数据量大及结构不规范的特点。针对库表多、数据量大的特征，需要设计便捷的自动化数据同步方案，并结合优秀的开源数据抽取工具，提高数据读写效率，减少数据的丢失率。针对结构不规范的特点，需要充分考虑到数据库表中的主键缺失、格式不一的现象，尽可能降低搜索引擎中数据重复、数据错误情况的发生率。

### 3.3.3 数据搜索——海量数据下的搜索方案

纪检数据搜索引擎针对的是大规模库表的结构化数据，在设计搜索方案时，要需要考虑到方案的通用性，确保其适用于系统下的绝大多数库表结构。同时，数据各表中也存在一定数量的无用字段，例如业务的预留空字段、标志位字段、测试字段等等，这些字段在一定程度上会加重搜索的负担，影响搜索结果的准确率，因此，在搜索时，如何设计灵活的规则，使得搜索引擎可以跳过无需检索的字段，这也是一项需要解决的问题。

### 3.3.4 数据展示——大规模库表下搜索结果的合理排布

搜索展示模块是与用户直接接触的模块，也是用户最为关心的一个模块。搜索展示主要需要考虑两部分：一为搜索结果的排布，二为搜索结果的易读性。

针对搜索结果的排布，需要充分利用库、表、字段信息的导向作用，将搜索结果进行合理的分类汇总。由于面对的是结构化的数据，若是像百度搜索那样以“标题—内容”的方式将搜索结果一条条的罗列，将会使页面十分散乱。下图3-3-4-1是本人所在部早期搜索引擎的查询结果截图，可以看到，在搜索关键词“项目”时，表中的“发放对象”、“资金种类”、“地点”、“发放时间”四个字段及其字段值被拼成一个文本作为一条搜索结果的标题，而这763页搜索结果没有经过合理的分类排布，只是单纯的罗列出来，这对后面的搜索定位造成了很大的苦难。



图 3-3-4-1 搜索结果示例图

二为搜索结果的易读性，纪检数据中，库、表、字段的命名几乎都是以拼音首字母命名，对于不熟悉数据业务的人来说，根本无从知晓这些属性所代表的意思。因此，对于这些晦涩的命名，还需要设计一个“数据字典”映射模块，存放这些命名计它们所对应的中文名称。在搜索引擎返回搜索结果前，先从字典模块做一次映射操作，将易读性好的名称展现给用户。

## 3.4 系统功能概述

本系统主要分为两大部分：搜索引擎部分，搜索引擎管理部分。其中，搜索引擎部分主要负责数据的搜索和展示。搜索引擎管理部分作为搜索引擎的后端管理工具，为搜索引擎提供服务，其负责数据库数据字典的管理、搜索引擎索引的管理以及数据库与搜索引擎之间的数据同步的管理等，这两部分可以划分出以下五个模块。

### 3.4.1 数据字典模块

数据字典模块主要承担数据库、数据表、表字段的命名对照信息的管理，其功能用例图如图3-4-1所示



图 3-4-1 数据字典模块用例图

业务人员主要负责对数据字典的管理。其中，数据字典的录入有Excel和界面两种形式：Excel录入时，系统会对录入的数据进行校验，当校验不通过时，会拒绝录入请求，以确保库、表、字段信息与数据库的一致性；界面录入时，系统会自动读取数据库的信息，保证数据的准确性，因此不需要再做校验操作。搜索引擎等外部模块会调用本模块的接口获取字典数据来完成相关信息的映射操作。

### 3.4.2 搜索引擎配置模块

搜索引擎配置模块是数据库与搜索引擎之间相接的“桥梁”，其负责搜索引擎索引的管理、数据表与搜索引擎之间的索引配置等。其功能用例图如图3-4-2所示：



图 3-4-2 搜索引擎配置模块用例图

索引配置操作主要配置数据表与搜索引擎之间的字段参数以及搜索相关参数。通过数据表筛选和字段筛选两项配置需要加入到搜索引擎中的数据；通过字段的搜索、分词、展示三项分别配置某个表的需要被搜索引擎搜索到的数据、需要被分词的数据、需要在前端展示页面中显示的字段；通过设置字段的权重来控制各个字段的重要程度，从而动态调整搜索结果的排序。

### 3.4.3 数据同步模块

数据同步模块负责将数据库中相关表的数据按照3.4.3中配置的规则导入到搜索引擎中。系统支持数据的全量更新与增量更新：全量更新即是将某表的所有数据一次性导入搜索引擎中；增量更新是在全量更新完成后，只将后期增加或变化的数据定时或实时的导入搜索引擎中。模块用例图如图3-4-3所示



图 3-4-3 数据同步模块用例图

业务人员通过前端页面选择相应的数据表，选择该表数据所要流向的索引（目的索引）以及更新周期。配置完成后，系统会在后台自动完成数据的全量更新并根据更新周期所设置的时间定时进行数据的增量更新。

### 3.4.4 数据统计模块

数据统计模块主要为数据同步模块提供服务，其会统计数据字典模块中各库、表、字段信息

### 3.4.5 数据搜索与结果展示模块

搜索与展示模块主要包括数据的搜索与搜索结果的展示。当用户输入搜索关键词后，系统首先在全库范围内进行查询，并将查询结果按照库、表两级分类聚合后返回给用户，用户可以点击相应表的链接进入到具体表的查询结果页面。

## 3.5 本章小结

# 第4章 系统设计

## 4.1 系统架构

## 4.2 系统核心功能设计

### 4.2.1 Elasticsearch索引结构设计

(1) Elasticsearch索引概念

在Elasticsearch中，索引是具有相似特性的文档的集合。每个索引由若干个主分片(shards)和副本分片(replicas)组成。主分片负责存储数据，当索引有新增数据时，Elasticsearch会将该条数据按照一定的路由规则插入到某个主分片中。因此，在一个索引中，各个主分片是相互独立的，每个主分片都拥有不同的数据。副本分片是主分片的一个拷贝，每个主分片都拥有0个或多个副本分片，副本分片主要作为硬件故障时保护数据不丢失的冗余备份，同时可为数据的搜索和文档的返回提供服务。

分片的设计使得Elasticsearch天生就具备分布式的优势。图5-1所示为一个具有三个节点的Elasticsearch集群示例图。

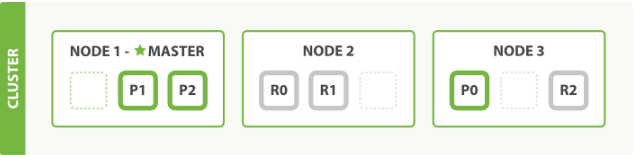


图 5-1 Elasticsearch集群示例图

其中，Pn(n=1,2,3)为索引的主分片，Rn(n=1,2,3)为索引的副本分片。从图5-1可以看出，该索引是一个拥有3个主分片和3个副本分片的索引，Elasticsearch将6个分片均衡的分配到这三个节点上。每个主分片和其对应的副本分片都不在一个节点上，这使得在某个节点宕机时数据仍保持完整不丢失，保证了服务的高可用性。

(2) Elasticsearch索引结构组成

Elasticsearch的索引结构为JSON结构，其由索引名、索引设置项、类型映射项和索引别名项组成，如图5-2所示。

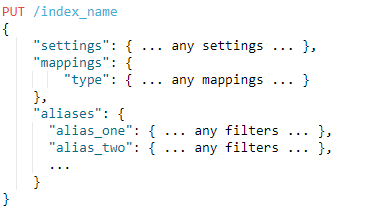


图 5-2 Elasticsearch索引结构图

索引设置项(settings)包含了索引的一些参数配置，如主分片数、副本分片数等。

类型映射项(mappings)主要配置索引的各个字段。对于每个字段，可以配置其字段类型、分词器、字段权重、是否索引等参数。字段类型中，最重要的为字符串类型，其可分为以下两种类型：

* text类型：用于存储长文本，例如家庭住址、项目内容等，系统会根据字段所配置的分词器对该字段进行分词。例如，字段值为“沈阳建筑公司”时，分词器可能会将其分为“沈阳”、“建筑”、“公司”三个词，当任意搜索三个词中的一个或多个时，都可以检索到该字段。
* Keyword类型：用于存储需要精确搜索的字段，如姓名、身份证号、手机号等，系统不会对该字段进行分词。仍以“沈阳建筑公司”为例，当且仅当搜索词为“沈阳建筑公司”时该字段才可以被检索到。因此，keyword类型在字段值与搜索词完全匹配的情况下才可以被搜索。

分词器用于对字段的分词，一条字段值会被分为多个单词分别存储倒排索引中。由于本系统所面对的数据都为中文数据，这里的分词器选用第2章介绍的hanlp中文分词器。“字段权重”为字段的重要程度，这是一个浮点数类型的正数值，某字段权重越高，该字段在最后搜索结果评分中的得分便越高，其搜索结果也会越靠前。可见，字段权重主要影响搜索结果的排序。“是否索引”参数用于设置某字段是否加入倒排索引中，若参数值为“false”，即不加入索引，则该字段的值不会被搜索到。

索引别名项(aliases)用于配置索引的别名，索引的别名可以类比Windows系统中文件的快捷方式。一个索引可以创建多个索引别名，每个索引别名可以指向一个或多个索引，对索引别名的操作等同于对其指向的索引的操作。同时，索引别名还提供了类似关系型数据库中“视图”的功能。每个索引别名都可以配置一个过滤条件(filter)来过滤出不同的数据，如过滤“年龄字段值小于18的所有数据”。因此，索引别名可以像“视图”一样将一个物理索引从逻辑上划分为多个相互独立的“虚拟索引”。

(3) 单表单索引的索引结构设计

系统最早采用的索引设计方案为单表单索引的方案，即对每个表都建立一个Elasticsearch索引，数据表中的字段与索引中的字段一一对应。这种方案的优点为实现简单，当进行数据表至索引的数据同步时基本不需要作任何转换操作。

其大致结构如图5-3所示

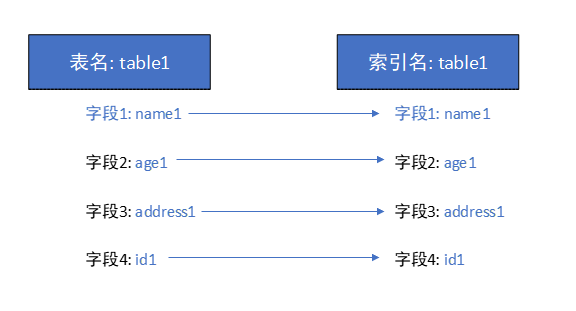


图 5-3 单表单索引的索引结构图

由于索引分片式的组成方式，随着数据库、表的日益增多，Elasticsearch中的分片数目也会日益庞大。在这种情况下，进行单表查询时，Elasticsearch只是将查询请求广播到该表下的分片，由于单个表的分片数恒定不变，因此查询效率较之前差距不大。而针对纪检业务的多库多表查询需求时，查询请求会被广播至这些索引的所有分片，每个分片作为一个独立单位各自执行查询操作，由于系统资源的有限性，大量分片竞争使用系统的计算资源，使得系统在资源调度和上下文切换中耗费大量时间，从而大幅降低查询的效率。

因此，单表单索引的结构适合于针对单索引或少量索引的查询，并不满足系统的多库查询需求。

（4） 单库单索引的索引结构设计

单库单索引是系统曾采用的另一种索引设计方案，即针对一个库下的表只建立一个索引，该索引就类似一个库，囊括了该库下若干表的数据。索引的字段为加入到该索引的表的字段的并集。以库为单位的索引设计大大降低了Elasticsearch中的索引数目，使得查询效率有了大幅度提升。

其大致结构如图5-4所示

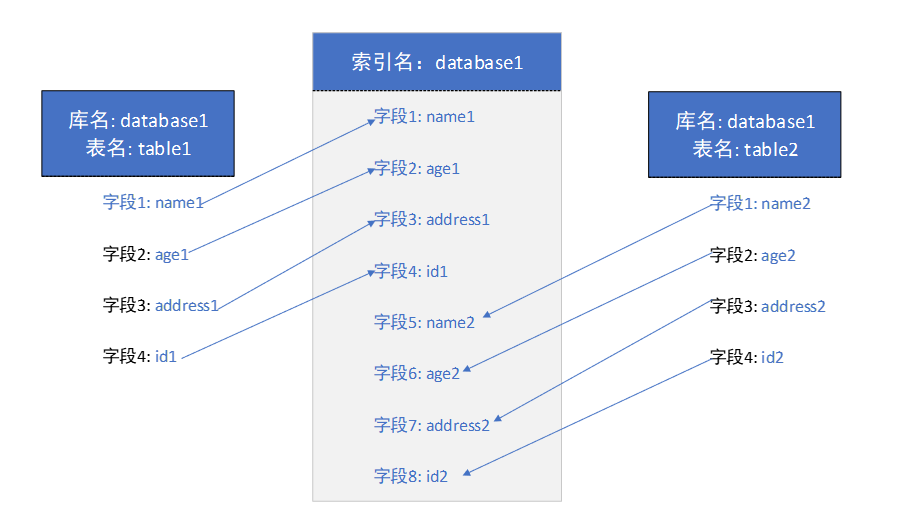


图 5-4 单库单索引的索引结构图

然而，由于库下不同表之间的字段命名可能相同，这种设计有时会导致索引中字段类型不一致的现象的出现。例如，A表中有一项名为“update\_time”的字段,字段格式为“date”格式，B表中也有名为“update\_time”的字段，字段格式为“timestamp”格式，两个字段的格式不一致，索引中的“update\_time”字段无法同时兼容这两种格式，从而影响数据的导入。

（5） 基于表、字段二级映射的索引结构设计

（3）所述方式与（4）所述方式存在一个共同的特点：索引与某个具体的库/表一一对应，索引字段与具体的表字段一一对应。当索引与数据表一一对应时，每个索引的字段数较少，然而所有索引的总分片数与表的数目成正比，分片数过多成为影响系统性能的重要因素。当索引与数据库一一对应时，索引的总分片数较前者大大减少，然而这会造成每个索引的字段数成倍增加。同时，不同表之间同名字段的类型可能不同，容易造成数据的不一致性。

纪检数据的查询需求为全库数据的搜索，单表单索引的结构与单库单索引的结构都不能很好的适应查询需求。在吸取前两者的优势之上，本系统设计了一种基于表、字段二级映射的索引结构，其大致结构如图5-3所示

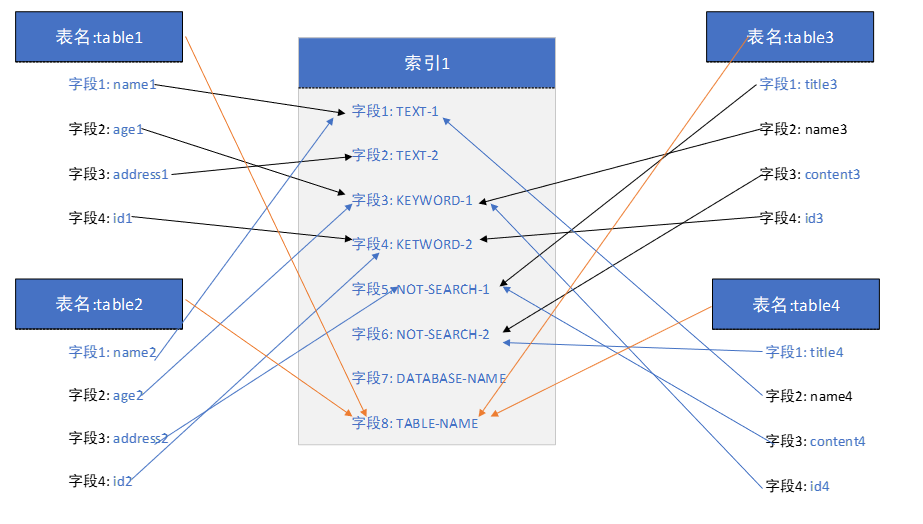


图 5-3 基于表、字段二级映射的索引结构图

在这种结构下，索引不再与具体的库、表相对应，索引字段也不与具体的表字段一一对应，实现了索引与库、表、字段的解耦合,使得索引的分片数及字段数不再受数据库、表、字段数目的影响。

索引字段分为三种类型：TEXT类型(其编号为TEXT-\*)、KEYWORD类型(其编号为KEYWORD-\*)、NOT-SEARCH类型(其编号为NOT-SEARCH-\*)。其中，TEXT类型为被搜索且被分词的字段，对应elasticsearch中的text类型，用于地址等长文本字段；KEYWORD类型为被搜索但不分词的字段，对应elastic search中的keyword类型，用于姓名、身份证等需要精确搜索的字段；NOT-SEARCH类型为不被搜索的字段，用于与业务无关的字段；库名、表名分别作为两个字段DATABASE\_NAME、TABLE\_NAME存储在索引中。索引与库、表均为一对多的关系。每个索引初建立时只有DATABASE\_NAME、TABLE\_NAME两个字段。每当有新表加入到索引时，系统会根据表中配置的搜索、分词属性将其所对应的索引字段分类命名为TEXT-1、TEXT-2…, KEYWORD-1、KEYWORD-2…, NOT-SEARCH-1、NOT-SEARCH-2…，若索引中不存在某个字段，则动态地添加相应的字段。

假设待写入的一条表数据为图5-4所示数据：



图 5-4 待写入的表数据图

该数据中的各字段与索引的映射关系如图5-5所示：

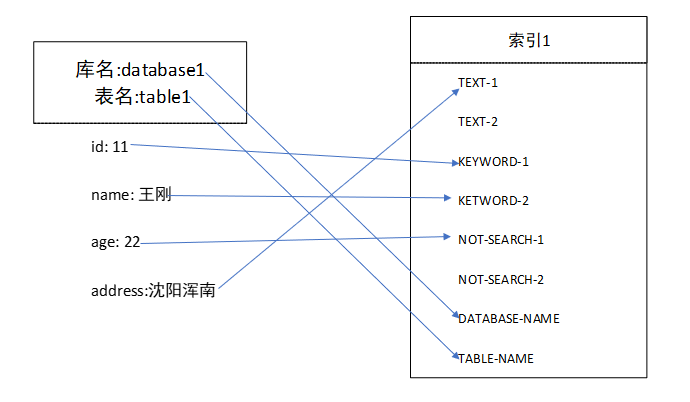


图 5-5 表数据与字段映射关系图

经过系统字段映射后，最终进入到索引中的数据如下图5-6所示



图 5-6 表数据与字段映射关系图

(6) 索引查询性能测试

为了测试上述三种索引结构对查询速度的影响，我们从沈阳市纪委数据库池中随机挑选了10个库(共计173个表，2.68亿数据)，针对3种索引结构，在相同的服务器配置下分别部署了三套服务，从数据库中随机挑出10个关键词进行测试。测试结果如图5-3所示

图 5-3 三种索引结构查询速度测试图

从折线图中可以看出，基于表、字段二级映射的索引结构在查询速度上要明显的优于其它两种索引结构。

### 4.2.2 数据同步方案设计

（1）数据同步概述

数据同步即保持数据表与索引之间数据的一致性。本系统主要通过数据的全量更新和增量更新两种方式来实现数据的同步。

数据的全量更新是在数据初始化时，将数据表中的数据一次性导入到Elasticsearch中的过程。本系统使用数据库流式读取的方式获取数据库数据，借助kafka、logstash两个开源工具完成数据的导入。

数据的增量更新是在数据全量更新之后，将后期数据表中数据的新增、变更或删除同步到elasticsearch中。本系统采用两种方式进行数据的增量同步：采用canal获取数据库变化的方式；采用定时任务对数据进行定期全量更新的方式。

（2） 基于数据库流式读取、kafka、logstash的数据全量更新

数据全量更新的示例图如图5-3所示：

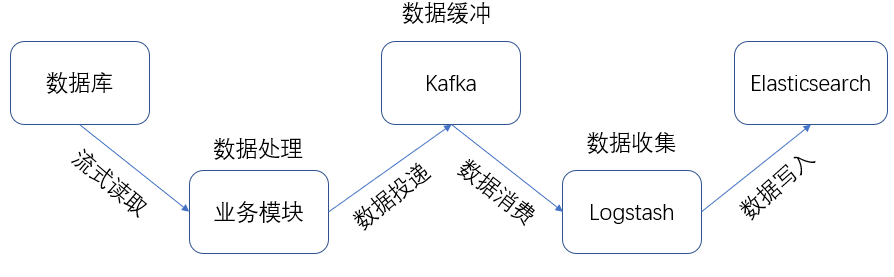


图 5-3 数据全量更新示例图

当执行查询操作时，数据库默认会将查询到的数据全部加载到内存中，而纪检数据库中通常存在数据规模在千万级甚至亿级的表，若这些数据一次性加载到内存中，会造成操作系统内存溢出的后果。因此，本模块使用数据库流式读取的方式来获取数据，使得查询结果以流的方式逐条返回，系统可以及时处理返回数据以避免数据的堆积。

读取数据后，业务模块按照5.1.5章节中的索引结构将数据进行字段映射并投递至kafka。Kafka是一个开源的流处理平台，其可以高效处理海量的流式数据。在本模块中，kafka主要用于数据缓冲，将业务模块处理后的数据进行短暂落盘，使得业务模块可以高效处理数据库的返回数据，避免其长时间持有数据库连接；同时，kafka的数据落盘功能使得elasticsearch在出现故障或写入失败时可以重新消费数据，降低了数据的丢失概率。

logstash是一款开源的ETL工具，在此模块中作为数据写入工具，其作为kafka的一个消费者，负责收集消费业务模块投递的数据，并将其批量写入到elasticsearch中。

（3） 基于canal、kafka、logstash的数据增量更新

增量更新的难点为获取数据的删除、更新信息，系统使用canal来监控数据库的二进制日志(binlog)以获取数据库的变更。

canal是阿里巴巴的开源的一款基于binlog日志增量订阅的工具，其工作原理如图5-4所示。canal会模拟数据库的主从交互(master/slave)协议，将自己伪装成数据库的一个slave，向数据库发送dump请求。数据库的master端收到dump请求后，就会将数据库的变更以binlog日志的形式推送给canal，从而实现了数据实时的获取功能。

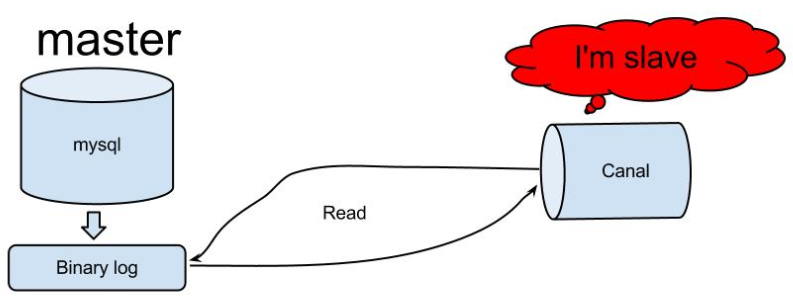


图 5-4 canal工作原理图

kafka在此充当消息队列的职责，其既作为生产者，又作为消费者。作为消费者来说，canal作为其信息来源，由业务模块将canal解析的binlog数据库变更信息处理后发送给kafka，kafka将消息落盘起来；作为生产者来说，其为logstash提供数据，logstash会实时的消费kafka中的变更信息，并写入到elasticsearch。整个增量更新的示例图如图5-5所示。

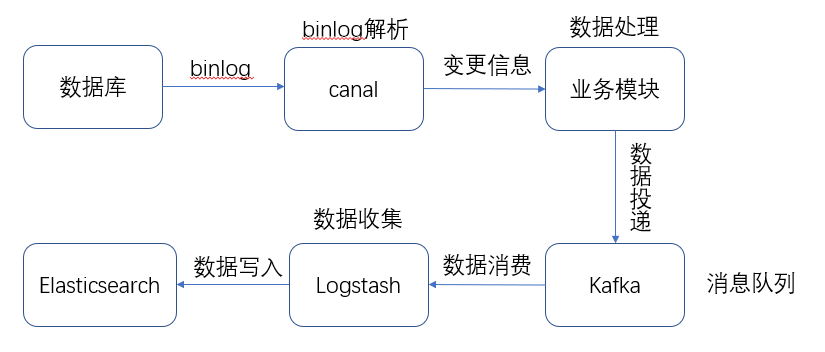


图 5-5 数据增量更新示例图

由于网络等因素，增量更新可能会出现数据丢失的情况，影响搜索的质量。为了保证数据的完整性，系统也采用了5.2.2所述的方式对数据进行周期性的全量更新。

（4） 数据去重

数据重复是纪检数据表中普遍存在的问题，数据重复会降低搜索引擎的查询效果，影响用户的体验。系统使用MD5信息摘要算法(MD5 Message-Digest Algorithm)，为每条数据生成一个唯一的MD5码的方式来降低重复概率。MD5码作为elasticsearch中每条数据的ID，当数据插入时，elasticsearch会检测数据的ID是否在索引中存在，只有在不存在的情况下才会执行数据插入，从而确保了具有相同MD5码的数据在每个索引中只会出现一次，降低了数据重复的概率。

## 4.3 系统模块设计

在本系统中，每个数据表需要经过三个步骤后其数据才能被搜索引擎检索到，表的状态变化如图4-1所示。



图 4-1 表的状态变化图

用户需要从数据库的原始表中选择相应的表并配置该表及该表字段的数据字典信息，该功能由“数据字典模块”实现。配置完成后，该表变为“已配数据字典”状态。

第二步需要配置表的字段属性，例如，哪些字段的内容需要存储到搜索引擎、哪些字段需要被搜索、分词、哪些字段不需要再展示页面中显示、每个字段的搜索权重是多少，该功能由“搜索引擎配置模块”实现，配置完成后，该表变为“已完成搜索配置”状态。

第三步需要配置表的数据所要存储的索引的名称、表的全量更新周期，该功能由“数据同步模块”实现。配置完成后，该表变为“以进行数据同步“状态。

这三步完成后，该表的数据会进入到搜索引擎中，通过“搜索与展示模块”检索数据。同时，数据同步需要获取表和索引的记录条数以及字段的变化等，搜索结果展示时也需要列出每个库的更新日期，这些数据由“数据统计模块”提供。

下面将分别介绍各模块的主要业务流程

### 4.3.1 数据字典模块

（1）数据字典录入

数据字典的录入包含库信息录入、表信息录入、字段信息录入，录入方式有界面手动录入和excel录入两种形式。由于库信息、表信息、字段信息的录入流程大致相同，本节主要列出库信息录入的业务流程。

手动录入库信息时，通过点击库信息管理页中的手动录入选项进入到录入页面。系统会调用后台接口获取到所有未录入到字典中的数据库，用户通过下拉框的形式选择相应的数据库，填写库的中文名和库的详情，页面中可以同时添加多个库的字典信息。添加完成后，点击提交按钮即可完成数据提交。其业务流程图如图4-1所示



图 4-1 库字典手动录入业务流程图

excel录入库字典信息时，通过excel录入选项进入录入页面。系统会提供excel模板，用户下载excel模板后根据excel模板提供的表头信息添加字典信息，对于库字典来说，表头信息包含库的中英文名以及库的详情。填写好后点击上传按钮提交数据。由于excel录入不能保证数据的准确性，系统会检测用户提交的信息是否正确，如表头是否正确，数据库是否存在、库名是否为空，校验通过后即提交成功。其业务流程图如图4-2所示。



图 4-2 库字典Excel录入业务流程图

表信息录入、字段信息录入方式与库信息录入方式大致相同，这里不再详述。

（2）数据字典修改

数据字典修改的交互流程较为简单，只需点击相应的库、表、字段项的修改按钮进行编辑即可。目前，库对象可以修改库中文名和库详情，表对象、字段对象仅可以修改其中文命名。另外，系统为了提升查询效率，在设计数据库表结构时，某些关联字典信息表的数据表时除存储相应的外键id外，还冗余存储了字典的中英文信息。因此，当字典信息修改时，也需要对这些表的信息进行更新。为此，系统采用了kafka的订阅—发布功能，将数据的更新广播给各订阅模块，由相应模块自行更新。字典修改的业务流程图如图4-3所示。



图 4-3 字典修改业务流程图

（3）数据字典删除

由于搜索引擎模块的大多数操作都依赖于数据字典模块。因此，系统对数据字典的删除作了限制，在不满足条件的情况下会拒绝用户的删除操作。以表字典删除为例，当该表已经被加入到Elasticsearch索引中时，系统会禁止对该表及字段的数据字典及删除操作，否则搜索引擎模块在数据展示时将无法完成对该表的中文命名转换。字段字典的删除同样受其所在表是否加入到搜索引擎的限制，另外，由于表字典与字段字典一对多的关联关系，表字典的删除会同时删除该表的字段字典，其删除操作的业务流程图如图4-4所示。库字典的删除有所不同，只有当该库下的表字典均被删除时，才可删除相应库的信息。



图 4-4 表字典删除的业务流程图

### 4.3.2 搜索引擎配置模块

搜索引擎配置模块主要负责索引的管理以及数据表搜索配置，本小节主要概述索引的创建以及数据表的搜索字段配置。

（1）索引的创建

Elasticsearch中的索引(index)类似关系型数据库中的“库”的概念，是数据的仓库，所有写入Elasticsearch中的数据都会存储在索引中。本系统的索引结构采用4.2.1小节提出的“表—字段二级映射”的结构，在创建索引时指定索引名和主分片数，索引初始只有库名、表名、id等字段，与各数据表相关的字段将在配置表映射时动态的添加。索引创建的流程如下：



图 4-5 索引创建业务流程图

（2）数据表的搜索字段配置

数据表的搜索字段配置主要配置表的字段属性，包括要存储到搜索引擎中的字段，需要搜索、分词、前端展示的字段以及每个字段的权重，配置完成后，系统会将该表的配置信息保存在数据库中，供其他模块调用。业务流程如图4-6所示



图 4-6 搜索引擎表字段配置业务流程图

字段信息配置后支持后期修改，根据表的状态的不同可以修改的内容也不同。当表在“已完成搜索配置”状态时，可以任意增删字段、修改字段的配置；当表在“已配置数据同步”状态时，由于数据已经同步到搜索引擎且搜索引擎不支持字段的修改，因此该状态下不允许增删字段，不允许修改字段的搜索、分词属性，只可以修改字段权重以及非搜索字段的“是否展示”属性。业务流程如图4-7所示



图 4-7 搜索引擎表字段编辑业务流程图

字段配置信息不可随意删除，当某表处于“以进行数据同步”状态时，系统会禁止对该表的字段配置信息的删除操作，否则搜索引擎将无法正常搜索该表的数据。字段删除的业务流程如图4-8所示



图 4-7 搜索引擎表字段编辑业务流程图

### 4.3.3 数据同步模块

数据同步模块主要负责维护数据表与Elasticsearch之间数据的一致性。在本模块中，数据的同步可分为三个步骤：

* 映射关系配置：设置表与数据写入索引的对应关系以及表字段与索引字段的对应关系
* 数据全量更新：将表的现有数据一次性全部写入索引中
* 数据增量更新：在数据全量更新后，定时或实时地对新增、变更数据进行更新

整个数据同步的业务流程图如图4-8所示。首先，用户选择相应表进入数据同步配置页，在该页中配置表数据所要写入的索引的名称以及该表的更新周期，配置完成后将数据提交至后端系统。而后，系统会获取搜索引擎配置模块中该表的搜索字段配置并检测这些字段在索引中是否存在，如果不存在则在索引中新增缺失的字段。字段检测完成后，系统会生成该表与索引的映射关系以及该表字段与索引字段的映射关系，并将映射关系存储至数据库。存储完成后，系统进行全量更新操作将数据写入索引中，并定时/实时地对数据进行更新。



图 4-8 数据同步业务流程图

### 4.3.4 数据统计模块

数据统计模块主要统计存储纪检数据库中字段、表、库的数据情况，为其他模块提供服务。该模块不由用户触发，由系统设置定时任务自动完成。

对于字段，系统会计算每个字段的缺陷度，计算方法如下：随机抽取表中该字段的若干条数据，设数据条数为n，n条数据中空数据的条数为m，若m/n>=c，则该字段为缺陷字段，其中c为一个自定义的常数，当前系统设置的值为0.9。缺陷字段的数据将不会写入到搜索引擎中。

对于表，计算每个表的数据条数、字段数、缺陷字段数以及更新日期，若前三个属性中的任意一个发生变化，则更新“更新日期”属性。

对于库，计算每个库的表的数目、数据条数以及更新日期。

### 4.3.5 搜索模块

一般来讲，搜索是指根据用户输入的一串查询文本，在索引中查找与该文本相关数据，并将搜索结果按找排序算法排序后分页的展示给用户的过程。纪检数据搜索系统的搜索流程与之略有不同，其流程图如图4-9所示。



图 4-9 纪检数据搜索流程图

当用户输入查询文本时，系统首先对查询文本进行分词，分词完成后构造查询串并在纪检数据全库索引范围内进行搜索。搜索完成后将搜索结果按照库、表作两级分类并汇总，将统计结果分页展示给用户。

在搜索结果统计页上，用户可以看到与该查询串相关的所有的库、表信息以及各表下查询结果的数量。通过点击表的链接可以查看具体表的查询结果。

在查看具体表的搜索结果时，系统会获取该表的搜索字段以及各字段权重，字段的权重会改变搜索结果的排序。系统根据这些信息再次执行搜索过程，并将搜索结果进行字段映射后分页展示给用户。

## 4.4 系统数据库设计

### 4.4.1 数据库概念设计

纪检数据搜索系统涉及到的实体对象主要有库对象、表对象、字段对象、表映射、字段映射、ES索引和ES字段。

库对象、表对象与字段对象分别存储纪检数据库、表、字段的中英对照信息，库与表、表与字段均为一对多的关系。

表映射、字段映射两个映射表分别存储纪检数据表与Elasticsearch索引的对应关系，纪检数据表字段与Elasticsearch索引字段的对应关系，每个表、字段均对应一条映射，因此，表与表映射、字段与字段映射均为一对一的关系。

ES索引、ES字段两个表存储elasticsearch的索引信息以及每个索引的字段信息，每个索引有多个字段，ES索引与ES字段为一对多关系。同时，每个ES索引会容纳多个表的数据，表中的字段也会根据一定的映射规则映射到相应的索引字段。因此，ES索引与表映射、字段映射均为一对多的关系。

系统主要实体的关系图如图4-1所示



图 4-1 系统实体关系图

### 4.4.2 数据库逻辑设计

（1）库字典表(dict\_database)

库字典表主要包含中英库名、库详情等，具体属性见表4-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 库id | id | bigint(20) | 主键 |
| 2 | 英文库名 | en\_database | varchar(255) | not null |
| 3 | 中文库名 | ch\_database | varchar(255) | not null |
| 4 | 库详情 | detail | text | not null |
| 5 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 6 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-1 库对象表

（2）表字典表(dict\_table)

表字典表主要包含中英表名、所在库id以及“是否配置到搜索引擎”项，具体属性见表4-2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 表id | id | bigint(20) | 主键 |
| 2 | 所在库id | database\_id | bigint(20) | 外键,关联库对象表 |
| 3 | 英文库名 | en\_database | varchar(255) | 冗余字段 |
| 4 | 英文表名 | en\_table | varchar(255) | not null |
| 5 | 中文表名 | ch\_table | varchar(255) | not null |
| 6 | 是否加入到搜索引擎 | is\_add\_to\_se | bit(1) | 0未加入，1已加入 |
| 7 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 8 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-2 表对象表

（3）字段字典表(dict\_column)

字段字典表主要包含中英字段名、所在库表的id等，具体属性见表4-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 字段id | id | bigint(20) | 主键 |
| 2 | 所在库id | database\_id | bigint(20) | 冗余外键,关联库对象表 |
| 3 | 所在表id | table\_id | bigint(20) | 外键，关联表对象表 |
| 3 | 英文库名 | en\_database | varchar(255) | 冗余字段 |
| 4 | 英文表名 | en\_table | varchar(255) | 冗余字段 |
| 5 | 英文字段名 | en\_column | varchar(255) | not null |
| 6 | 中文字段名 | ch\_column | varchar(255) | not null |
| 7 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 8 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-3 字段对象表

（4）ES索引表(es\_index)

ES索引表主要存储Elasticsearch中创建的索引的名称、分片数，具体属性见表4-4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 索引id | id | bigint(20) | 主键 |
| 2 | 索引名称 | index\_name | varchar(255) | 字母必须小写 |
| 3 | 主分片数 | num\_shards | tinyint(4) | 小于10 |
| 4 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 5 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-4 ES索引表

（5）ES索引字段表(es\_column)

ES索引字段表主要包含索引字段名称、所在索引id项，具体属性见表4-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 索引字段id | id | bigint(20) | 主键 |
| 2 | 所在索引的id | index\_id | bigint(20) | 外键，关联ES索引表 |
| 3 | 字段名 | name | varchar(255) | not null |
| 4 | 字段类型 | type | varchar(255) | 取值为（TEXT、KEYWORD、NOT-SEARCH） |
| 5 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 6 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-5 表对象表

（6）搜索引擎表(se\_table)

搜索引擎表负责维护所有完成搜索字段配置的表，这些表在配置表映射后其数据就可以进入到搜索引擎中。具体属性见表4-6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 库对象id | database\_id | bigint(20) | 冗余外键，关联库对象id |
| 3 | 表对象id | table\_id | bigint(20) | 外键，关联表对象id |
| 4 | 是否配置表映射 | is\_sync | bit(1) | 0未配置，1已配置。取值为1时，该表的数据才会进入索引中 |
| 5 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 6 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-6 搜索引擎配置记录表

（7）表映射信息表(mapping\_table)

表映射信息表主要存储表对象及其所在的索引，同时存储该表的记录数与索引中该表的记录数、该表的更新周期项用于数据同步。具体属性见表4-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 库对象id | database\_id | bigint(20) | 冗余外键，关联库对象id |
| 3 | 表对象id | table\_id | bigint(20) | 外键，关联表对象id |
| 4 | 所在索引的id | index\_id | bigint(20) | 外键，关联ES索引id |
| 5 | 英文库名 | en\_database | varchar(255) | 冗余字段 |
| 6 | 英文表名 | en\_table | varchar(255) | 冗余字段 |
| 7 | 表记录数 | table\_records | bigint(20) | not null |
| 8 | 索引记录数 | index\_records | bigint(20) | not null |
| 9 | 更新日期 | update\_date | date | not null |
| 10 | 更新周期 | update\_period | tinyint(4) | not null |
| 5 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 6 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-7 表映射信息表

（8）字段映射信息表(mapping\_column)

字段映射信息表主要包含字段对象及其索引字段名称、所在索引id，每个字段的搜索、分词、展示项以及字段权重等，具体属性见表4-8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 表对象id | table\_id | bigint(20) | 外键，关联表对象id |
| 3 | 字段对象id | column\_id | bigint(20) | 外键，关联字段对象id |
| 4 | 索引字段名 | es\_column | varchar(255) | not null |
| 5 | 索引字段类型 | type | varchar(255) | 取值为（TEXT、KEYWORD、NOT-SEARCH） |
| 6 | 是否搜索 | is\_searched | bit(1) | 0不搜索，1搜索 |
| 7 | 是否分词 | is\_analyzed | bit(1) | 0不分词，1分词 |
| 8 | 是否展示 | is\_displayed | bit(1) | 0不展示，1展示 |
| 9 | 字段权重 | boost | double | not null |
| 10 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 11 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-8 字段映射信息表

（9）库信息统计表(stats\_database)

库信息统计表主要包含数据库的表数、记录数信息，具体属性见表4-9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 库对象id | database\_id | bigint(20) | 外键，关联库对象的id |
| 3 | 表数目 | total\_tables | int(11) | not null |
| 4 | 记录数 | total\_records | bigint(20) | not null |
| 5 | 库更新日期 | update\_date | date | not null |
| 6 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 7 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-8 库信息统计表

（10）表信息统计表(stats\_table)

表信息统计表主要包含数据表的记录数以及字段信息，具体属性见表4-10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 库对象id | database\_id | bigint(20) | 冗余外键，关联库对象的id |
| 3 | 表对象id | table\_id | bigint(20) | 外键，关联表对象的id |
| 3 | 字段数 | total\_columns | tinyint(4) | not null |
| 4 | 缺陷字段数 | defect\_columns | tinyint(4) | not null |
| 5 | 记录数 | total\_records | bigint(20) | not null |
| 6 | 表更新日期 | update\_date | date | not null |
| 7 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 8 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-10 表信息统计表

（11）字段信息统计表(stats\_column)

字段信息统计表主要统计各字段是否为缺陷字段（90%的记录中该字段都为空），具体属性见表4-11。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 字段 | 类型 | 备注 |
| 1 | 主键 | id | bigint(20) | 主键 |
| 2 | 库对象id | database\_id | bigint(20) | 冗余外键，关联库对象的id |
| 3 | 表对象id | table\_id | bigint(20) | 冗余外键，关联表对象的id |
| 3 | 字段对象id | column\_id | bigint(20) | 外键，关联字段对象的id |
| 4 | 是否缺陷 | is\_defected | bit(1) | 0不缺陷,1缺陷 |
| 7 | 创建时间 | gmt\_create | datetime | 默认当前时间 |
| 8 | 修改时间 | gmt\_modified | datetime | 默认当前时间 |

表 4-11 字段信息统计表

## 4.5 本章小结

# 第5章 系统实现

## 5.1 系统物理结构

## 5.2 系统物理部署

## 5.3 系统主要功能实现

本节将从一个具体的表实例出发，分析其从字典录入开始到数据可被搜索到的整个实现流程，从而串联起系统各模块的主要功能。

假设选取的表为房产库(zfsj\_fc)下的房地产开发商表(dsj\_fcj\_ggfc\_fdckfs\_old)。该表的字段列表如表5-3-1所示。

### 5.3.1 数据字典录入

在本例中，需要录入房产合同表的库字典信息(所在库的中英文名以及库的详情介绍)、表字典信息(表的中英文名信息)以及字段字典信息(该表下各字段的中英文名信息)。

从图5-3-1可知，该库的库字典信息已经录入，因此，只需录入表、字段的字典信息。

数据字典的录入有界面录入和excel录入两种，这里针对表使用界面录入的方式，针对字段使用excel录入的方式。

（1）表字典的界面录入

点击进入表字典的录入界面，如图5-3-2所示。

点击表英文名的下拉框，系统会获取并返回该库的所有未添加字典信息的表的英文名。其接口定义如下,其中，database参数为数据库英文名

List<String> listTablesNotAdd(long database);

该方法首先获取到database库下的所有表名集合，再从表字典表中获取该库下已添加字典信息的表的集合，并将两个集合求差集以list列表的形式返回给前端。可以点击“增加”按钮添加多个表的信息，信息填写好后提交即可。

（2）字段字典的excel录入

字段excel录入首先要下载excel模板，按照模板的表头填写字段信息，本例中，填写的字段excel如图5-3-3所示，其中，enDatabase、enTable、enColumn和chColumn分别为库英文名、表英文名、字段英文名、字段中文名。填写好后提交即可。

字段excel录入的接口定义如下，其中file为上传的excel文件

void saveAll(MultipartFile file)

字段字典的实体类的属性如表5-3-1所示

由于不能保证excel填写的正确性，系统会对用户提交的excel文件做多重校验，当出现错误时，系统会拒绝录入请求并将错误信息返回给用户，错误分为下列五种：

* Excel表头错误：检测用户所提交的Excel的表头与系统提供的Excel模板表头是否一致，以防止用户提交错误文件。
* 数据库不存在：检测用户填写的数据库(enDatabase)是否真实存在
* 数据表不存在：在数据库存在的基础上，检测数据表(enTable)填写是否正确,即enDatabase库下是否存在enTable表。
* 数据表的字段有误：在数据表存在的基础上，检测字段(enColumn)是否填写正确，即enDatabase库下的enTable表中是否有enColumn字段。
* 字段中文名存在空值：字段中文名是否有漏填的现象。

只有校验全部通过后系统才会将数据存储起来。

### 5.3.2 索引创建

数据表的数据最终要写入具体的索引中，若系统无索引或现有索引负载过重(索引的单个分片大小超过50GB)时，需要创建新的索引。索引创建只需在前台指定索引名称以及索引的主分片数，如图5-3-4所示。

系统创建索引的接口如下，其中，indexDTO为前台传入的索引表单，包含索引名称和索引的主分片数。

void createIndex(EsIndexDTO indexDTO)

索引的创建包含两个步骤，首先在数据库中保存索引的数据，即索引名和主分片数，其次在elasticsearch中创建对应的索引，两个步骤共同组成一个事务操作，代码结构如下

@Transactional(rollbackFor = Exception.class)

public void createIndex(EsIndexDTO indexDTO) throws IOException {

EsIndex esIndex = BeanUtil.toBean(indexDTO, EsIndex.class);

esIndex.setIndexName(indexPrefix + esIndex.getIndexName());

// 数据库中存储索引数据

esIndexRepo.save(esIndex);

// elasticsearch中创建索引

elasticRestService.createIndex(indexDTO);

}

下面重点介绍elasticsearch中创建索引的过程。

Elasticsearch提供了强大的JAVA api接口，系统使用Elasticsearch的Java High Level REST Client，通过调用封装后的RestFul api来执行相关操作。本系统的索引创建可以分为以下五个步骤。

首先调用索引创建请求并设置索引名称。

CreateIndexRequest request = new CreateIndexRequest(

indexDTO.getIndexName());

其次，设置索引的settings参数。系统设置了索引的主、副分片数以及索引刷新间隔，主分片数由用户设置，副分片数设为1，索引刷新时间调大为60s(数据写入索引后60s才可以被搜索到，调大该参数有助于提升写入性能)。

第三，创建索引的初始化字段。系统为每个新索引都新建了4个字段，如图5-3-5所示。

* elastic\_database\_id：库字典的id，存放每条数据所对应的数据库在库字典表中的id，根据这个id可以从库字典表中查询到该库的中英文名
* elastic\_table\_id：表字典的id，存放每条数据所对应的数据表在表字典表中的id，根据这个id可以从表字典表中查询到该表的中英文名
* all\_fields\_text和all\_fields\_keyword：这是两个复合字段，分别存储一条记录中所有被搜索且分词的字段值的集合与所有被搜索但不分词的字段值的集合。这两个字段覆盖了一条记录中所有被搜索的内容。在搜索聚合过程中，为了提高搜索效率，将使用这两个字段作为每个索引的搜索字段。



图5-3-5 索引初始字段列表

第四步，为索引设置一个公共的别名，使用别名可以向使用索引名那样直接操作索引，因此，当所有索引共用一个别名时，该别名可以同时操作所有索引。这在全库查询调用全索引时十分方便。

最后，向elasticsearch发起创建索引的请求，即可完成索引的创建，代码如下：

client.indices().create(request, RequestOptions.DEFAULT)

### 5.3.3 数据表搜索配置

在进行数据写入前，需先对数据表进行配置，包括数据表的搜索字段配置与数据表到索引的映射配置。

（1）数据表的搜索字段配置

数据表的搜索字段配置主要配置表中需要展示、搜索、分词的字段以及各字段的权重。这使得用户可以灵活的定义每个表中需要被搜索的字段，忽略无关字段，以此提高搜索引擎的内容质量；同时可以根据各字段的重要程度为字段设置不同的权重，以改变搜索引擎的排序，提高内容的相关度。通过点击图5-3-6上的“添加到搜索引擎”选项可以进入该表的字段配置页，如图5-3-7所示。

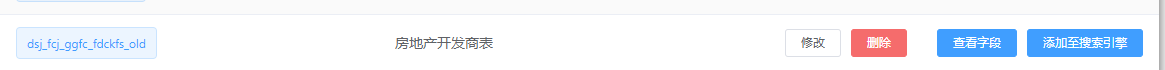


图5-3-6字段配置页入口

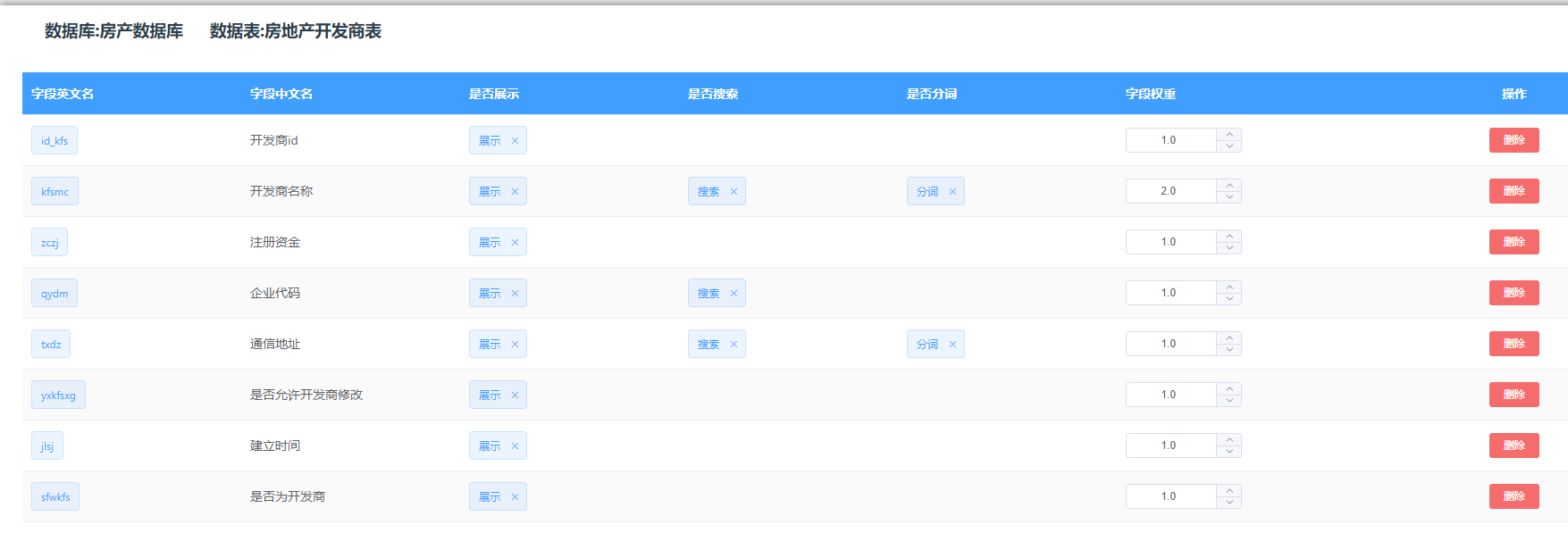


图5-3-7字段配置列表



图5-3-7字段数据内容

在该页中，系统首先获取字段字典表中房地产开发商表的所有字段，同时向数据同步模块获取该表的缺陷字段，两者求差集后，将剩余的字段返回到字段配置页。

在字段配置页中，可以通过点击每个字段右边的“是否展示”、“是否搜索”、“是否分词”三列来设置其相关属性，页面中，字段右边有搜索标签的为需要搜索的字段，分词标签、展示标签同理；通过调整“字段权重”列的数值可以改变字段的权重；点击“删除”选项可以删除不必要的字段。鼠标指向字段的英文名可以展示该字段的20条数据，用于字段设置的参考，如图5-3-8所示。配置完成并提交后，由系统转换并存储配置信息。

在配置信息中，最重要的为字段的配置信息，其实体类的主要属性如图5-3-10所示：

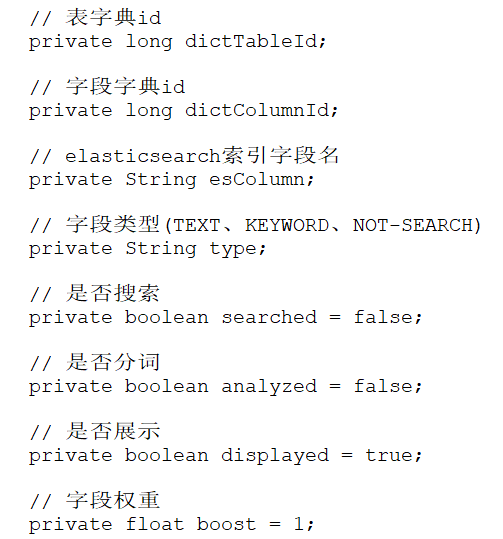


图 5-3-10 字段配置实体类属性

其中，type字段值可以有searched、analyzed两个属性得出，计算流程如图5-3-11所示。



图 5-3-11 type属性计算流程

esColumn根据type及各type类型字段的数量来计算。若字段列表中有三个TEXT类型的字段，则这三个字段的esColumn分别为TEXT-1、TEXT-2、TEXT-3；同理，若有两个KEYWORD类型的字段，则其esColumn分别为KEYWORD-1、KEYWORD-2。因此，计算esColumn，首先将字段按type分类，而后对每个type下的字段按顺序赋值即可，代码如下：

// 字段按type分类

Map<String, List<MappingColumnDTO>> columnsGroupByType =

seTableAddDTO.getColumns()

.stream()

.collect(groupingBy(this::getColumnType));

// 计算各个字段的esColumn属性

columnsGroupByType.forEach((type, list) -> {

int typeCount = 0;

for (int i = 0; i < list.size(); ++i) {

MappingColumnDTO mappingColumnDTO = list.get(i);

MappingColumn mappingColumn = BeanUtil.toBean(mappingColumnDTO, MappingColumn.class);

mappingColumn.setEsColumn(type + "-" + (++typeCount));

mappingColumn.setType(type);

mappingColumns.add(mappingColumn);

}

});

（2）数据表到索引的映射配置

这一步需要配置房地产开发商表的数据所要写入的目的索引的名称。在图5-3-12所示页中，点击“配置数据同步”选项，配置索引名和更新周期后提交即可。



图 5-3-12数据同步配置页

### 5.3.4 数据更新

### 5.3.5 数据搜索

# 第6章 总结和展望

本文针对沈阳市纪委的大规模结构化数据的检索需求，基于Elasticsearch这款优秀的开源搜索引擎服务，借助SpringBoot、SpringCloud等技术设计并实现了纪检数据搜索系统

# 第7章 致谢

# 第8章 参考文献

[1]Kwang-I Yu,Shi-Ping Hsu,Peggy Otsubo.The fast data finder—an architecture for very high speed data search and dissemination[C].Intemational Conference on Data Engineering,1984(4):50-55.

[2] 张莉. 垂直搜索引擎中分词和排序技术的研究与应用 [D]; 西南交通大学, 2014.

[4] 陈翠婷. 无分类小商品搜索引擎关键技术研究 [D]; 中国科学院大学, 2016.

[4] 姜华. 基于Lucene面向主题搜索引擎的研究与设计 [D]; 华东师范大学, 2007.

[5]中国青年网.沈阳：用大数据深挖“不正之风” 管住身边腐败[EB/OL].https: //baijiahao.baidu.com/s?id=1618821229579984841&wfr=spider&for=pc,2018-12-03.

Hhhhhhhhhhhhhhhhhhh’

[4] 徐海. 基于Lucene垂直搜索引擎的研究与实现 [D]; 西安科技大学, 2009.

[5] 肖亮. 垂直搜索引擎的研究与实现[D]; 北京交通大学, 2007.