

Southwest University of Science and Technology

本科毕业设计（论文）



基于NoSQL的网络文学监测分析系统优化

|  |  |
| --- | --- |
| 学院名称 | 计算机科学与技术学院 |
| 专业名称 | 软件工程 |
| 学生姓名 | 曾丹 |
| 学号 | 20134484 |
| 指导教师 | 杨春明 副教授 |

二〇一七年六月

基于NoSQL的网络文学监测分析系统优化

摘要

网络文学监测分析系统是一款具有统计分析网络文学作者作品地域分布、性别分布、类型分布、排行分布以及评论词云等功能的系统，能够通过该系统清晰明了的洞察网络文学的发展趋势。目前每日可采集到相关作者作品信息上千条，随着时间的推移，数据的累积，现已采集作者数达30万，作品数近30万，评论数165万。由于该系统采用的传统关系型数据库Mysql，关系型数据库需提前设计表结构以及表之间关系，导致部分查询需要至少3个表的联表查询，导致部分统计分析接口响应时间超过5分钟，十分影响用户体验。针对上述问题，结合当下已成熟的应对海量数据的解决方案，决定更换数据库并修改逻辑，采用NoSQL数据库存储数据。NoSQL数据库即非关系型数据库的总称，分别有Key-Value键值对形式、面向列、面向文档的存储形式存储数据，其中主流NoSQL数据库有Hbase、Redis、MongoDB等数据库。MongoDB为最类似于关系型数据库的NoSQL数据库，十分适合注重海量数据访问效率提升的系统。本次采用反范式设计模式及索引技术对系统进行优化，经过使用与体验，总共有8个页面，20多个功能待优化，优化后与使用Mysql数据库时比对，性能显著提高，页面平均响应时间3秒。

关键词：网络文学监测分析系统，NoSQL数据库，MongoDB数据库

**Base on NOSQL database on the monitoring and analysis of network literature system optimization**

**ABSTRACT**

The monitoring and analysis of network literature system which has the functions of statistical analysis of the geographical distribution, gender distribution type distribution, ranking distribution and commentary clouds of the author's work, which can clarify the development trend of network literature through the system. At present, the daily collection of relevant works of thousands of works, with the passage of time, the accumulation of data, has been collected the number of 300,000, the number of works nearly 300,000, the number of comments 1.65 million. As the system uses the traditional relational database Mysql, the relational database needs to design the table structure in advance and the relationship between the table, resulting in some of the query requires at least three tables of the table query, resulting in part of the statistical analysis of the interface response time of more than 5 minutes, Affect the user experience. In response to the above problems, combined with the current solution to deal with massive data, decided to replace the database and modify the logic, using NoSQL database to store data. NoSQL database that is non-relational database of the general term, respectively, Key-Value key pairs of form, oriented, document-oriented storage form of storage data, which mainstream NoSQL database Hbase, Redis, MongoDB and other databases. MongoDB is the most similar to the relational database NoSQL database, is very suitable for focusing on massive data access efficiency of the system. The use of anti-paradigm design and indexing technology to optimize the system, through the use and experience, a total of eight pages, more than 20 functions to be optimized, optimized with the use of Mysql database compared to the performance significantly improved, the average page response Time 3 seconds.

**Key words:** The monitoring and analysis of network literature system, NoSQL, MongoDB

1. 绪论

1.1课题的背景

伴随信息时代的高速发展，数据以爆炸式方式增长。面对海量数据的存储与处理，传统关系型数据库已无法满足用户需求，甚至严重影响系统性能以致用户体验呈直线下降。随着Internet网的普及，人们的生活方式也发生了巨大变化，由现金支付转为网上支付，由纸质书转为电子书等等。

网络文学监测分析系统是一款统计分析网络文学作者作品相关信息的网站，以图表形式将统计分析数据清晰明了的展现给用户。现系统已采集作者数达30万，作品数近30万，评论数165万，由于该系统采用的传统关系型数据库Mysql，关系型数据库需提前设计表结构以及表之间关系，导致部分查询需要至少3个表的联表查询，导致部分统计分析接口响应时间超过5分钟，十分影响用户体验。

基于以上问题，所以本文主要研究网络文学监测分析系统在海量数据环境下NoSQL数据库的应用，通过更换数据库和更改系统逻辑实现，旨在优化网络文学监测分析系统性能，提升系统的响应速度，使得用户体验有质的飞跃。

1.2 课题研究现状

1.2.1 NoSQL数据库研究现状

NoSQL数据库最早可追溯到1991年Berkeley DB的第一版Berkeley DB是一个键值存储(Key-value Store)的Hash数据库，适用于数据类型相对简单但需要极高的插入和读取速度的嵌入式场合。NoSQL从2007年开始蓬勃发展。Google和Amazon的工程师们分别发表了有关BigTable和Dynamo数据库的论文，描述他们己经在使用的新型数据库的设计思想。BigTable提出了列存储模型，证明了数据持久存储可以扩展到数以千计的节点, Dynam。则提出了最终一致性的思想以达到更高的可用性和扩展性。分布式缓存系统Memcached证明了内存分布式数据存储可以达到极高的性能。事实上，Google的BigTable, Amazon的Dynamo和分布式缓存系统Memcached这三者可以说是所有NoSQL数据库的启蒙者。很多公司和组织受到他们思想观点的启发，开发了自己的开源 NoSQL数据库产品，例如Hypertable就是BigTable思想的开源实现。

现如今，国外的NoSQL数据库品种己经超过100个。除了Google的BigTable,Amazon的Dynamo，主流的数据库还有FaceBook的Cassandra, lOgen的MongoDB, Amazon的SimpleDB, Apache的HBase, Redis, CouchDB, Tokyo Cabinet/Tyrant, MemcacheDB等等。这些数据库在互联网中应用广泛，表现优异。Cassandra的用户包括Twitter, Cisco, Urban等多个公司，已知最大的Cassandra集群有超过400个节点，存储数据量超越300TB最受欢迎的MongoDB数据库在电子商务、政府、数据存储、实时统计、社交网络、游戏等众多领域都应用广泛，其中著名的有github, source forge, Disney等，国内的淘宝网、大众点评和视觉中国也在使用MongoDB国内的 N0SQL数据库领域在2009年也开始活跃起来。人人网下属UGC团队参考Dynamo设计文档，开发了自己的NoSQL数据库Nuclear。豆瓣推出了开源数据存储系统BeansDB。盛大创新院也基于Tokyo Cabinet开发了TCDatabase开源数据库。相比国外的NoSQL数据库，国内的这几款数据库并不成熟，也没有在生产中有更广泛的应用。国内大型的互联网公司多数还是选择国外支持度高的NoSQL数据库，除了淘宝、大众点评、视觉中国选用MongoDB外，新浪微博选用Redis存储数据，盛大也推出了基于MongoDB的“盛大云”MongoIC的数据库云服务。

在NoSQL蓬勃发展途中，有着成功案例的同时，NoSQL数据库也有一些失败的经历。2010年7月Twitter提出暂停使用Cassandra替代MySQL存储feed的计划，原因是Cassandra仍缺少大并发海量数据访问的案例及经验，新产品需要一定稳定期。2010年10月，使用MongoDB的foursquare网站由于数据碎片化和监控不力的原因，经历长达11小时的两次宕机。随着不断的完善，NoSQL也愈加成熟，得到广泛应用。

1.2.2 MongoDB数据库研究现状

MongoDB是一个性能卓越的开源的NoSQL数据库。它的存储模式自由，能够存储文档型数据等结构数据，灵活的存储方式在存储数据方面对传统的关系型数据库是一种强大的冲击，它逐渐的替代了Key/V alue存储方式。MongoDB的这些优点省略了开发过程中很多不必要的麻烦和繁琐的操作。同时，MongoDB能够很好的应对大多数互联网应用对大数据量、高并发的需求。MongoDB的水平扩展能力，使应用具备了处理上千万甚至上亿级别的数据处理能力。这些性能极大的满足了Web2.0和移动互联网系统对大数据存储的需求，同时，它的即插即用的特点也从某种程度上大大降低了相关行业网站的运行和维护的成本。两年来，跟随着Web2.0的发展势头，国内对NoSQL数据库出现了热忱的局面，MongoDB作为NoSQL中性能比较优越的一个，自然的成为了最有影响的一个。很多的互联网公司都将MongoDB作为数据库存储的最佳选择，在大数据的存储上也给专家、学者们在科研上提供了很好的存储服务，在行业内创造了很多具有不可估量的价值的互联网、移动应用。

1.3 课题的研究意义

由于网络科技的高速发展与移动设备的普及，基本人手一台手机，达到足不出户便能知晓天下事。小说阅读是当今人们生活中重要的娱乐方式之一，李克强曾指出：读书不仅事关个人修为，国民的整体阅读水准，也会持久影响到整个社会的道德水平。随着2015年“两会”的圆满结束，“全民阅读”的发展进程进一步加快。在移动互联网时代的大背景下，移动设备已逐渐取代了传统的实体书，成为了人们主要的阅读工具。据统计，截止2015年，网络文学用户规模已达3.5亿人，而纸质图书作为最“正统”的阅读工具，用户仅占15%，因此阅读移动化已经势不可挡，也正是由于网络文学作品的大量出产，所以掌握网络文学作品发展趋势以方便对其管控和了解用户需求是尤为重要也是极其必要的。而网络文学监测分析系统正是以简单有效的图表形式展现给用户，可更方便了解网络文学用户的需求以及网络文学作品发展态势。

1.3.1 必要性

网络文学监测分析系统的性能随着爬取的网络文学作者作品信息数量递增，出现明显的下降，经过几次维护与修改，仍然未能从根源彻底解决问题，导致系统使用者往往想要查看数据分析的图表需要花费大量的时间等待，甚至有时会超过用户等待时间的最大限度，导致系统用户渐渐减少，如果这样的情况不能得到及时有效的解决，此系统将会成为一款无人使用的系统，就如同一栋烂尾楼，无人问津。为避免网络文学监测分析系统成为一死系统，所以对系统的优化不能停滞不前，要从根源解决问题才是根本，经过问题的排查和之前的维护与修改经验，最终确定是使用传统关系型数据库Mysql存储数据的原因，Mysql数据库需要提前定义好表字段以及表字段间关系，由于要满足三范式，所以获取一条数据往往需要关联多张表数据，在大数据量的情况下，查询效率差，因此更换数据库，基于NoSQL数据库完成对系统的优化是极其必要的。

1.3.2 重要性

基于非关系型数据库的网络文学监测分析系统与基于传统关系型数据库的网络文学监测分析系统相比较，具有以下6个优点：

1. 将关系型数据库Mysql替换为非关系型数据库MongoDB能够从根源解决系统性能瓶颈问题；
2. MongoDB采用的是面向文档存储方式，不需要提前定义好表字段及表之间关系，内部使用的是BSON语法，能够更直观易理解；
3. MongoDB海量数据下，查询效率高效快速，进而解决系统性能低下问题；
4. MongoDB使用简单，可以支持JSON、mapreduce、正则表达式以及Javascript函数等查询方式；
5. MongoDB相较于Mysql，节省系统资源，对CPU等资源的耗费较小；
6. MongoDB具有高可扩展性，支持分布式集群，通过分片数据缩放处理理论可实现更高的吞吐量。

1.4课题目标和研究范围

1.4.1 目标

在当前Web2.0时代的互联网应用和大数据时代的海量数据管理的背景下，NoSQL数据库相较于传统的关系型数据库更适合处理高并发读写、海量数据处理、可用性和高扩展性的应用需求。本文目的在于针对NoSQL中表现优异的MongoDB在网络文学监测分析系统中展开研究与应用，使系统功能响应时间小于5秒。

1.4.2 研究范围

1. NoSQL技术综述：NoSQL非关系型数据库技术使用松耦合的数据模式，支持水平伸缩，拥有在磁盘和(或)内存中的数据持久化能力，支持多种“Non-SQL”接口来进行数据访问。NoSQL的数据模型包括Key-Value键值对、Key一结构化数据、Key一文档、列簇式存储和图结构存储。NoSQL支持复杂的查询、弱事务机制，支持冗余备份保证单机可靠性、多种数据同步方式实现多机可靠性，支持哈希分区和范围分区来进行分布式扩展，提供强一致性和最终一致。
2. MongoDB服务机制分析和应用优化：MongoDB是当前应用广泛的NoSQL数据库。本文研究了MongoDB应用及优化以下方法与原则：
3. 针对MongoDB面向文档的数据模型，提出反范式模式设计方法；
4. 根据MongoDB查询实现机制，研究MongoDB索引的使用
5. MongoDB应用：网络文学监测分析系统抓取存储多个大型小说网站的作者和作品信息，存储量大且增长快。本文根据网络文学监测分析系统的存储、查询、扩展以及复制冗余等需求，给出了MongoDB的解决方案，并与之前应用的Mysql数据库时的查询效率以及系统功能响应时间进行了对比。

1.5课题实施方案

本课题将采用先对系统优化总体规划，然后系统功能优化分模块实施，需求调研与应用开发紧密结合的技术实施方案。

1. 在海量数据情景下，对NoSQL数据库的应用进行充分的调研，分析并抽取完成系统优化必要流程和步骤，对系统优化进行可行性分析。
2. 学习和掌握实现网络文学监测分析系统优化所需的必要技术支持，指定系统优化实现的技术方案；
3. 找出实现系统优化所要解决的技术关键点，给出解决该技术难点的可行解决方案；
4. 完成系统优化总体设计，包括实现数据库的更换，数据库的设计与数据的迁移，提取系统功能等；
5. 完成系统优化详细设计，包括功能模块优化具体实施方案的选择；
6. 根据前期的总体设计和详细设计，逐步分模块实现系统功能优化；
7. 完善系统所有功能优化，并加强系统在功能、安全和网络适应性等方面的性能。

1.6 本章小结

本章对课题所要研究的基于NoSQL的网络文学监测分析系统优化进行了简要介绍，与使用传统关系型数据库相比较，给出了本课题的研究意义，并且介绍了非关系型数据库的研究现状，明确了本课题所要实现的项目目标和研究范围，给出系统优化的实施方案，为后期系统的优化明确了方向和思路。

1. 系统优化需求分析

2.1 系统介绍

网络文学监测分析系统主要通过爬取各大型网络小说网站数据，然后分析其中例如地域，性别，书籍类别，点击量，评论数，推荐数，评论等等有效信息，以地图、饼图、折线图及柱状图或列表形式将统计分析后的数据以最简单最直观的形式展现给用户，使其能一目了然，能够快速获取其想要获取的数据信息。同时，通过系统抓取的数据可以有效的观察各大类型小说的发展趋势，以及当今小说流行与发展的状况和趋势，并且该系统还能提供其他好处，其一能够结合统计分析结果做书籍推荐，其二能够结合统计分析结果分析出网络文学作品现状，可方便做宏观调控，以方便管理。

2.2 系统优化可行性分析

系统优化可行性分析可以有效的避免对系统优化的错误投资，是保证系统优化成功的基本前提。本课题对基于NoSQL的网络文学监测分析系统优化进行了以下几方面的可行性分析：

经济可行性：本系统的开发人员没学生，由双导师指导，传统关系型数据库Mysql及非关系型数据库MongoDB均为开源软件，故此系统优化无需开发经费，开发成本低廉。而本系统的实现可给该系统带来无限好处，一是避免系统成为死系统，造成前期投入的人力物力造成浪费，二是从根源解决系统功能响应缓慢问题，大大提升系统性能，三是使系统有了更好的扩展性，随着以后数据的膨胀，只需加机器及简单配置，四是突破自我，研究除已熟悉的关系型数据库外的额外非关系型数据库，加强自己能力的同时也是为学弟学妹们打好跨出已知知识领域的第一步。因而本系统优化是有必要也是有意义的，在经济上是可行的。

技术可行性：技术可行性是指分析系统优化的技术条件是否满足需要。本课题将实现基于NoSQL的网络文学监测分析系统优化，系统将采用的开发技术包括Java、Spring、SpringMVC、MongoDB、JavaScript、JQuery、Echart等技术。目前，这些技术已得到广泛应用，可以借助大量的学习资料，因而系统优化在技术上是可行的。

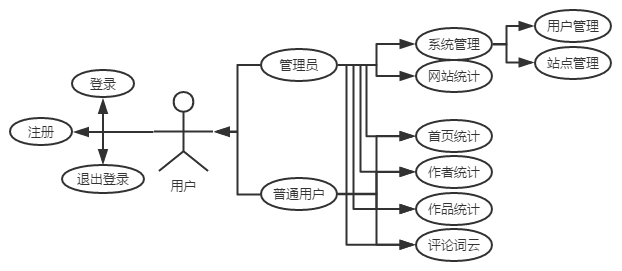
从以上两方面分析本系统优化是完全可行的，且系统操作简单且易于理解，但由于前期开发时数据量较小未过多注意系统性能问题导致随着数据的膨胀系统性能直线下降，严重影响用户体验，所以为了改善系统性能，优化是势在必为的。

2.3 系统中术语与缩写解释

**表2-1 术语/缩写及解释**

|  |  |
| --- | --- |
| 术语/缩写 | 解释 |
| SSM | Spring+SpringMVC+Mybatis开发框架 |
| MVC | Model-View-Controller将业务逻辑、数据、界面显示分离的思想 |
| JSON | 一种轻量级的数据交换格式 |
| Echart | 数据统计图表工具 |
| JDK1.8.0 | Java 编程语言构建应用和组件的开发环境，“1.8.0”为JDK版本号 |
| Tomcat8 | 免费开源的轻量级Web应用服务器 |

2.4 系统优化功能性需求分析



**图2-1 系统用例图**

通过该系统用例图，可以看出网络文学监测分析系统共有2个角色：管理员及一般用户，已实现的功能如下表所示：

**表2-2 功能及描述**

|  |  |
| --- | --- |
| 功能 | 功能描述 |
| 注册 | 用户需填写用户名密码、真实姓名、联系电话、邮箱及申请理由等信息注册账号 |
| 登录 | 已通过审核用户可以登录。用户数据存储于Mysql数据库，密码采取MD5加密算法，存储加密后密文 |
| 审核用户 | 仅管理员拥有的功能。管理员根据情况审核注册用户是否通过，并设定账号有效期天数 |
| 删除用户 | 仅管理员拥有的功能。管理员可以删除已失效或需要删除的用户数据 |
| 新增用户 | 仅管理员拥有的功能。管理员可以手动添加新用户 |
| 修改用户 | 仅管理员拥有的功能。管理员可以更改用户信息，如是否通过认证，用户名称，账号有效期等信息 |
| 系统参数修改 | 仅管理员拥有的功能。管理员可以根据需要修改页面初始大小值PageSize以及页面页脚显示信息 |
| 删除数据站点 | 仅管理员拥有的功能。管理员可以删除无用站点 |
| 新增数据站点 | 仅管理员拥有的功能。管理员可以新增爬取数据的数据站点 |
| 修改数据站点 | 仅管理员拥有的功能。若网站信息发生更改，管理员可以及时更改站点信息，以便不影响数据的抓取 |
| 网站统计 | 仅管理员拥有的功能。包括网站网址及其含有的文学作者、作品数量统计及最新更新时间，方便管理员管理网络文学监测分析网站，并且知道网站是否处于正常运行状态 |
| 首页统计 | 包括采集网站数、作者总人数、作者总数；作者区域分布百分比及地方人数地图、作者性别占比、Top10作者；作品Top类别占比、Top10作品；作者作品近期更新状况曲线图 |
| 作者统计 | 包括作者区域分别、性别占比、作者作品数统计、作者所有作品点击量评论数推荐数统计；作者详情列表，可按作品数、点击量、评论数、推荐数做升序降序排序功能 |
| 作者检索 | 包括按网站站点名称、按区域、按作者名称检索 |
| 作者所有作品分析 | 包括作者基本信息以及针对该作者的所有作品评论进行分词，提取出现最频繁有用的词语构建词云，可查看每部作品重要信息，每部作品的点击量占比及每部作品点击量、评论数、推荐数柱状图 |
| 作品统计 | 包括作品Top类别统计、作品点击量评论数推荐数统计、作品最新更新时间；作品详情列表，可按作品点击量、评论数、推荐数做升序降序排序功能 |
| 作品检索 | 包括按网站站点名称、按作品名称检索 |
| 作品分析 | 包括作品基本信息以及针对该作品的所有评论进行分词，以词频为准统计，分析出最重点且有用的词云信息，还有距最近更新时间往前推一周的该作品的点击量评论数推荐数曲线图 |
| 退出登录 | 需要切换账号或者其他情况下，可正常退出登录 |

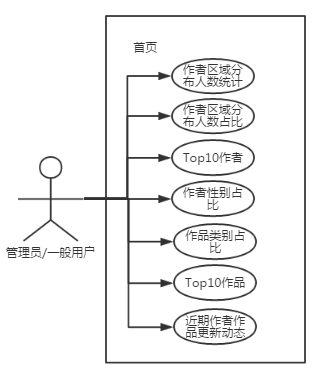
其中首页统计分析、作者首页统计分析及作者列表页面统计和作者详情页面统计分析、作品首页统计分析及作品列表页面统计和作品详情页面统计分析、网站统计页面是本次基于NoSQL数据库针对网络文学监测分析网站优化的重点，其他地方可以暂不做优化。

2.4.1 系统优化指标

**表2-3 描述及指标**

|  |  |
| --- | --- |
| 描述 | 指标 |
| 首页 | 共有8个功能，页面响应时间应小于5s |
| 作者统计页面 | 共有7个功能，页面响应时间应小于5s |
| 作者分页页面 | 共有5个功能，页面响应时间应小于3s |
| 作者详情页面 | 共有5个功能，页面响应时间应小于3s |
| 作品统计页面 | 共有5个功能，页面响应时间应小于5s |
| 作品分页页面 | 共有5个功能，页面响应时间应小于3s |
| 作品详情页面 | 共有3个功能，页面响应时间小于3s |
| 单条件查询 | 20ms以内 |
| 聚合条件查询 | 例如分组、排序等，500ms以内 |

2.4.2 首页

网络文学监测分析系统首页包括作者区域分布人数统计和人数占比，Top10作者，作者性别占比，作品类别占比，Top10作品及近期作者作品更新动态。用例图如下：  


**图2-2 首页用例图**

首页或作者统计页的作者区域分布人数统计用例规约描述如下：

**表2-4 作者区域分布人数统计用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者区域分布人数统计 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页或查看作者统计页面 |
| 前置条件 | 1a、管理员/一般用户在导航栏处点击“首页”或“作者统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作者处的“更多” |
| 基本事件流 | 1、作者区域分布地图显示统计的作者人数 |

首页的作者区域分布人数占比用例规约描述如下：

**表2-5 作者区域分布人数占比用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者区域分布人数占比 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页 |
| 前置条件 | 管理员/一般用户在导航栏处点击首页 |
| 基本事件流 | 1、作者区域分布占比列表显示数据 |

首页的Top10作者用例规约描述如下：

**表2-6 Top10作者用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | Top10作者 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页 |
| 前置条件 | 管理员/一般用户在导航栏处点击“首页”，然后点击“最热作者” |
| 基本事件流 | 1、默认按作者点击量降序排序展示排名前10位作者信息，包括作者作品数、点击量、评论量、推荐量  2、点击“评论量/推荐量”，按作者评论量/推荐量降序排序展示排名前10位作者信息 |

首页或作者统计页的作者性别占比用例规约描述如下：

**表2-7 作者性别占比用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者性别占比 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页或查看作者统计页面 |
| 前置条件 | 1a、管理员/一般用户在导航栏处点击“首页”或“作者统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作者处的“更多” |
| 基本事件流 | 1、正常显示男女和未知性别比例的饼图 |

首页的作品类别占比用例规约描述如下：

**表2-8 作品类别占比用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品类别占比 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页 |
| 前置条件 | 1a、管理员/一般用户在导航栏处点击“首页”或“作品统计”  1b、管理员/一般用户在导航栏处点击“首页”，然后点击最热作品处的“更多” |
| 基本事件流 | 1、默认统计前9种类别占比，其余按其他类别归并统计 |

首页的Top10作品用例规约描述如下：

**表2-9 Top10作品用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | Top10作品 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页 |
| 前置条件 | 管理员/一般用户在导航栏处点击“首页”，然后点击“最热作品” |
| 基本事件流 | 1、默认按点击量降序排序展示排名前10个作品信息，包括作品点击量、评论量、推荐量  2、点击“评论量/推荐量”，按作品评论量/推荐量降序排序展示对应排名前10个作品信息 |

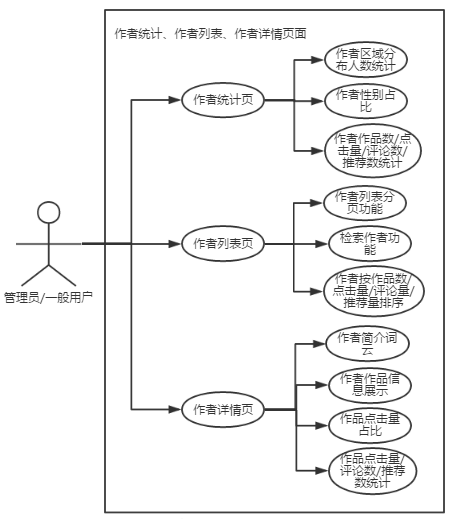
首页的近期作者作品更新动态用例规约描述如下：

**表2-10 近期作者作品更新动态用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 近期作者作品更新动态 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看首页 |
| 前置条件 | 管理员/一般用户在导航栏处点击“首页” |
| 基本事件流 | 1、按日期显示作者/作品更新数 |

2.4.3 作者统计、作者列表及作者详情页

网络文学监测分析系统作者统计、作者列表及作者详细页包括作者区域分布人数统计和人数占比，作者性别占比，作者列表，作品简介词云，分页检索功能等。用例图如下：



**图2-3 作者统计、作者列表及作者详情页用例图**

作者统计页的作者作品数/点击量/评论量/推荐量统计用例规约描述如下：

**表2-11 作者作品数/点击量/评论量/推荐量统计用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者作品数/点击量/评论量/推荐量统计 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者统计页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作者处的“更多” |
| 基本事件流 | 1、正常显示作者作品数/点击量/评论数/推荐数分布的柱状图 |

作者列表页的作者列表分页功能用例规约描述如下：

**表2-12 作者列表分页功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者列表分页功能 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者列表页面 |
| 前置条件 | 管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者” |
| 基本事件流 | 1、默认加载前三页数据，默认按作者总点击量降序排序  2、点击“上一页/下一页”，加载上一页/下一页数据  3、点击页面大小下拉框，选择每页展示数据的数量，页面展示数量改变  4、输入跳转页面，点击“跳转”，页面跳转 |

作者统计页的检索作者功能用例规约描述如下：

**表2-13 检索作者功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 检索作者功能 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者统计页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作者处的“更多”  2a、点击“作者”输入框，输入作者名称，点击“检索”  2b、点击“区域”输入框，输入区域名称，点击“检索”  2c、点击网站列表，选择任一网站，点击“检索” |
| 基本事件流 | 2a、显示模糊匹配作者名称的作者信息  2b、显示模糊匹配作者区域的作者信息  2c、显示选择网站下的作者信息 |

作者列表页的作者按作品数/点击量/评论量/推荐量排序用例规约描述如下：

**表2-14 作者按作品数/点击量/评论量/推荐量排序用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者按作品数/点击量/评论量/推荐量排序 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者列表页面 |
| 前置条件 | 管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者” |
| 基本事件流 | 1、点击“作品数量/总点击量/总评论量/总推荐量”，作者列表按作品数量/总点击量/总评论量/总推荐量升序排序，再次点击“作品数量/总点击量/总评论量/总推荐量”按其降序排序 |

作者详情页的作品简介词云用例规约描述如下：

**表2-15 作品简介词云用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品简介词云 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者”，任意选择一个作者进入作者详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作者”，任意选择一个作者进入作者详情页面 |
| 基本事件流 | 1、正常展示作品简介词云 |

作者详情页的作者作品信息展示用例规约描述如下：

**表2-16 作者作品信息展示用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者作品信息展示 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者”，任意选择一个作者进入作者详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作者”，任意选择一个作者进入作者详情页面 |
| 基本事件流 | 1、正常展示作者信息  2、正常展示每部作品信息详情 |

作者详情页的作者作品点击量占比用例规约描述如下：

**表2-17 作者作品点击量占比用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者作品点击量占比 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者”，任意选择一个作者进入作者详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作者”，任意选择一个作者进入作者详情页面 |
| 基本事件流 | 1、正常展示每部作品点击量占比的饼状图 |

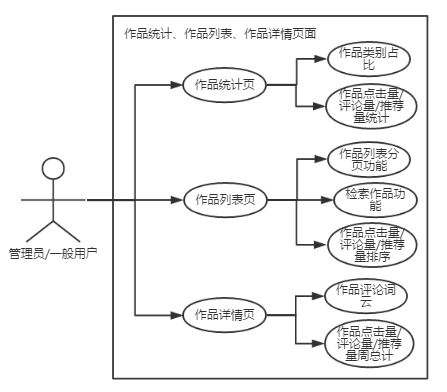
作者详情页的作者作品点击量/评论数/推荐数统计用例规约描述如下：

**表2-18 作者作品点击量/评论数/推荐数统计用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作者作品点击量/评论数/推荐数统计 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作者详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作者统计”，然后点击“所有作者”，任意选择一个作者进入作者详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作者”，任意选择一个作者进入作者详情页面 |
| 基本事件流 | 1、正常展示每部作品点击量/评论数/推荐数统计柱状图 |

2.4.4 作品统计、作品列表及作品详情页

网络文学监测分析系统作品统计、作品列表及作品详细页包括作品类别占比，作品列表，作品评论词云，分页检索功能等。用例图如下：



**图2-4 作品统计、作品列表及作品详情页用例图**

作品统计页的作品点击量/评论量/推荐量统计用例规约描述如下：

**表2-19 作品点击量/评论量/推荐量统计用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品点击量/评论量/推荐量统计 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品统计页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作品统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作品处的“更多” |
| 基本事件流 | 1、正常显示作品点击量/评论数/推荐数分布的柱状图 |

作品列表页的作品列表分页功能用例规约描述如下：

**表2-20 作品列表分页功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品列表分页功能 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品列表页面 |
| 前置条件 | 管理员/一般用户点击导航栏处“作品统计”，然后点击“所有作品” |
| 基本事件流 | 1、默认加载前三页数据，默认按作品总点击量降序排序  2、点击“上一页/下一页”，加载上一页/下一页数据  3、点击页面大小下拉框，选择每页展示数据的数量，页面展示数量改变  4、输入跳转页面，点击“跳转”，页面跳转 |

作品统计页的检索作品功能用例规约描述如下：

**表2-21 检索作品功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 检索作品功能 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品统计页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作品统计”  1b、管理员/一般用户点击导航栏处“首页”，然后点击最热作品处的“更多”  2a、点击“作品”输入框，输入作品名称，点击“检索”  2b、点击网站列表，选择任一网站，点击“检索” |
| 基本事件流 | 2a、显示模糊匹配作品名称的作品信息  2b、显示选择网站下的作品信息 |

作品列表页的作品按点击量/评论量/推荐量排序用例规约描述如下：

**表2-22 作品按点击量/评论量/推荐量排序用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品按点击量/评论量/推荐量排序 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品列表页面 |
| 前置条件 | 管理员/一般用户点击导航栏处“作品统计”，然后点击“所有作品” |
| 基本事件流 | 1、点击“总点击量/总评论量/总推荐量”，作品列表按总点击量/总评论量/总推荐量升序排序，再次点击“总点击量/总评论量/总推荐量”按其降序排序 |

作品详情页的作品评论词云用例规约描述如下：

**表2-23 作品评论词云用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品评论词云 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作品统计”，然后点击“所有作品”，任意选择一个作品进入作品详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作品”，任意选择一个作品进入作品详情页面 |
| 基本事件流 | 1、正常展示作品评论词云 |

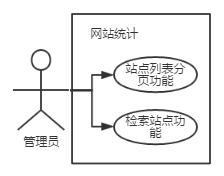
作品详情页的作品点击量/评论量/推荐量周统计用例规约描述如下：

**表2-24 作品点击量/评论量/推荐量周统计用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 作品点击量/评论量/推荐量周统计 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看作品详情页面 |
| 前置条件 | 1a、管理员/一般用户点击导航栏处“作品统计”，然后点击“所有作品”，任意选择一个作品进入作品详情页面  1b、管理员/一般用户点击导航栏处“首页”，然后点击“最热作品”，任意选择一个作品进入作品详情页面 |
| 基本事件流 | 1、正常展示作品点击量/评论量/推荐量周统计 |

2.4.5 网站统计

网络文学监测分析系统网站统计包括站点列表，分页检索功能。用例图如下：



**图2-5 网站统计用例图**

网站统计页的站点列表分页功能用例规约描述如下：

**表2-25 站点列表分页功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 站点列表分页功能 |
| 角色 | 管理员 |
| 用例说明 | 管理员查看网站统计页面 |
| 前置条件 | 管理员点击导航栏处“网站统计” |
| 基本事件流 | 1、默认加载前三页数据  2、点击“上一页/下一页”，加载上一页/下一页数据  3、点击页面大小下拉框，选择每页展示数据的数量，页面展示数量改变  4、输入跳转页面，点击“跳转”，页面跳转 |

网站统计页的检索站点功能用例规约描述如下：

**表2-26 检索站点功能用例规约描述**

|  |  |
| --- | --- |
| 用例名称 | 检索站点功能 |
| 角色 | 管理员/一般用户 |
| 用例说明 | 管理员/一般用户查看网站统计页面 |
| 前置条件 | 管理员/一般用户点击导航栏处“网站统计”，然后点击“作品”输入框，输入站点名称，点击“检索”，模糊匹配 |
| 基本事件流 | 显示检索匹配的站点信息 |

2.5 系统非功能性需求分析

非功能性需求是允许软件以满足用户的业务需求部分以外的要求。非功能性需求的是一个很容易被忽视的模块，但它对一个系统的开发和维护是非常重要的。系统非功能性需求应包括系统性能，可靠性，可维护性，健壮性等方面的问题。

本课题基于NoSQL的网络文学监测分析系统优化的非功能性需求定义如下：

实用性：系统满足网络文学作者作品统计分析要求；

易用性：界面美观，图表化易于理解，系统设计符合用户操作习惯；

可扩展性：系统优化结构设计合理，方便系统功能扩展；

可靠性：系统运行稳定，基本未发生故障；

健壮性：系统在发生故障时应有合理的处理和容错机制；

安全性：系统严格控制系统用户的使用权限及使用期限；

性能需求：系统数据量大，需要满足10万作者作品、300万评论时统计分析时间在5秒以内。

2.6 系统运行环境

软硬件需求描述如下表所示：

**表2-27 软硬件需求描述**

|  |  |
| --- | --- |
| 需求名称 | 详细要求 |
| 客户端硬件 | CPU：4.0GHz  内存：2048MB  硬盘：100GB  网卡：100/1000Mbps自适应  网络速度：2Mbps（独享） |
| 客户端软件 | 1、通用操作系统  2、建议使用IE9以上、Chrome、FireFox等浏览器 |
| 服务端硬件 | 物理1xCPU，逻辑2xCPU 2.0GHz，X86\_64，2核  内存：16GB  硬盘12G |
| 服务端软件 | 1、64位Linux操作系统  2、JDK1.8.0  3、Tomcat8 |
| 开发者环境 | 1. Window8.1专业版 2. JDK1.8.0 3. Tomcat8 4. Chrome浏览器 |

2.7 本章小结

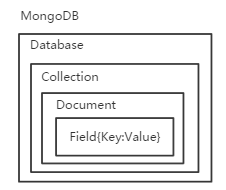
为能快速对网络文学监测分析系统进行优化，本章主要介绍了系统优化的可行性以及必要性，性能优化后需达到的指标，以页面为单位着重剖析了待优化功能点的需求，为之后的系统设计及开发做好充分的需求分析，有效避免因需求不清晰而进行的无用功。

1. 系统优化概要设计

3.1 MongoDB简介

3.1.1 MongoDB存储逻辑模型

MongoDB可以创建多个数据库，每个数据库由多个集合（Collection）组成，每个集合由任意个文档（Document）组成，而文档由任意个字段（Field）组成，每个字段是一个键值对（Key-Value），其中键（Key）是字段名称，值是对应的字段值，Mongo存储逻辑模型如下图：



**图3-1 MongoDB存储逻辑模型图**

1. 集合

MongoDB的集合本质上是一组文档，类似关系型数据库中的表。这些文档通常都有相同的字段结构，但MongoDB是模式自由(Schema-Free)的数据库，对字段结构没有强制的要求。用户不用预先定义一个集合的字段结构，可以存储不同结构的文档在同一个集合，在数据库运行时，可以随时动态地添加或删除文档的字段。MongoDB没有显式的创建集合的命令，集合在插入第一条文档时创建。

1. 文档

MongoDB是一个面向文档的数据库，这里的文档是指结构化的文档，以规则的key-value对的形式表示数据，而不是自由文本。这些文档可以理解为面向对象编程语言中的对象实例，但仅仅包含对象的数据，而不包含方法函数。

MongoDB的文档使用BSON语言表示，一个文档是一个BSON对象，包含多个键值对。BSON是一种二进制序列化的类JSON数据交换语言。它以二进制字节的形式存储键值对，键为字符串格式，值可以为任意数据类型，除了基本的整数，浮点数，字符串，日期等，还可以是数组或键值对。

例如一个学生的基本信息用BSON对象表示如下：

{

"name" : "zhangsan",

"address" : {"city" : "Beijing", "state" : "China"},

"favorites" : ["basketball", "football"],

"grade" : [{"lesson" : "Math","score" : 90}, {"lesson" : "English","score" : 100}]

}

可以看出该学生拥有姓名name、地址address、爱好favorites、成绩grade四个属性。姓名属性对应的值的类型为字符串，地址属性对应的是键值对，爱好属性对应的是数组，数组元素的类型是字符串，成绩属性对应的也是数组，但数组元素的类型是键值对。

BSON的二进制格式化是为节省存储空间设计的，即使在最坏的情况下，BSON格式也比JSON格式在最好情况下存储效率高。在某些情况下BSON会牺牲额外的空间让数据的传输更加方便。例如，字符串的传输的前缀会标识字符串的长度，而不是在字符串的末尾打上结束的标记。这样的传输形式有利于MongoDB修改传输的数据。

除了以上的优点外，BSON主要的优势在于它支持数组、键值对的复杂数据结构。这使得MongoDB可以嵌套子文档或数组，即文档的字段值仍是一个文档或数组类型。MongoDB可以将关系型数据库中本来要使用外键联接的多个集合归纳到一个集合中，减少了集合与集合之间的关联，提高了数据库性能。

例如上面的学生基本信息，MongoDB创建一个集合student就可以存储所有的信息。在关系型数据库中，因为其二维表元组的存储模型，address中的city城市和state国家需要创建除student表外的city表、state表，其中city表外键指向state表，student表外键指向city表，查询时需要三个表联接查询。学生的grade信息也需要另外创建lesson表、grade表，三个表之间进行外键联接查询。

3.2 系统优化策略

3.2.1 反范式模式

对一个应用来说，数据库的模式设计从根本上决定了数据存储、查询的效率和操作的复杂性。关系型数据库的规范化理论提出了1NF, 2NF, 3NF, BCNF, 4NF的范式概念，其基本思想是通过模式分解来逐步消除数据依赖中不合适的部分，使模式中的各关系模式达到某种程度的分离。与关系型数据库的二维表不同，MongoDB采用面向文档的数据逻辑模型，支持使用数组和嵌套子文档两种特殊的数据类型，规范化理论在MongoDB中不能完全适用。故针对非关系型MongoDB数据库提出了反范式模式设计方法。

由于网络文学监测分析系统着重在数据的统计分析与展示，其中涉及一个作品存在多次更新的记录，但只取最新一条记录的部分数据为有效数据，即此时为一对一。除此也存在一个文学作者创作了多部文学作品，而此时需要全部的作品信息数据，即此时为一对多，等等场景。

一个数据项Y对应多个数据项时，若对应数据项较少的情况下，MongDB允许将多个数据项以数组形式作为该数据项Y的一个子项，作为一个整体看待。

一个数据项X完全依赖于另一个数据项Y时，MongoDB嵌套子文档的数据类型允许将X作为Y的一个子项进行联合，作为一个整体看待。

而一个数据项部分依赖或传递依赖其他数据项时，因为数据项X完全函数依赖于(Y, Z)，即部分函数依赖于Y，那么X的值不能由Y唯一确定，所以数据项X不能作为Y的一个子项，作为一个整体看待，需要先根据2NF, 3NF和BCNF进行模式分解。

所以综合所有场景，针对MongoDB的反范式模式设计如下：

1. 若存在一对多场景时，多个数据项以数组形式作为子项，合并为一个数据项；
2. 若数据项X完全函数依赖于另一个数据项Y，且与其他数据项无函数依赖关系，则将X作为Y的一个子项进行联合，合并为一个数据项；
3. 若存在部分依赖和传递依赖关系时，需要先根据2NF, 3NF和BCNF进行模式分解。

3.2.2 索引技术

数据查询的速度很大程度上取决于索引的使用。MongoDB的索引机制类似于关系型数据库。MongoDB使用标准的B-Tree索引，提供全类型索引支持，包括普通数据类型、文档类型、数组类型，还支持多个字段的复合索引、嵌套子文档和数组的索引等。

1. 根据查询创建索引

若查询为单个字段，则创建单键索引即可；若查询为多个字段，则需创建复合索引。通过db.collectionName.ensureIndex({字段名:1/-1})创建索引。

1. 避免基数小的单键索引

通过db.collectionName.distinct(“字段”).length查看该字段存在多少个不相同的值，与集合大小比对，越接近越好。基数太小，索引没有明显的作用并且浪费索引空间。

1. 复合索引限制

若查询需要sort排序，则该字段必须为复合索引的最后一项；若查询中有范围查询，该字段必须是复合索引的最后一项；若两者皆有，只能选择其一作为复合索引的最后一项，可通过基数大小来决定使用哪个效果更好。

3.3 数据库设计

3.3.1 数据库环境

数据库：MongoDB

数据库可视化工具：RoboMongoDB

3.3.2 数据库表设计

根据需求分析，主要优化作者作品相关所有功能点及网站统计，总共涉及works、worksupdate、author、authorupdate、workscomment、crawlwebsite 6个数据库表，详情如下：

workTotalHits、workCommentsNum、workTotalRecoms三个字段在使用关系型数据库时是通过works表与worksupdate表关联查询出的数据，此处为了MongoDB避免联合查询，故增加这三个字段。

**表3-1 works表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 作品文档主键 |
| workAuthId | 作品作者id |
| workAuthor | 作品作者笔名 |
| workAuthorization | 作品作者签约情况 |
| workCommentsNum | 作品评论总数 |
| workDesc | 作品描述 |
| workId | 作品id |
| workMark | 作品内容标签 |
| workNature | 作品性质（公开或VIP） |
| workNum | 书号 |
| workOtherInfo | 作品其他固定信息 |
| workTitle | 作品名称 |
| workTotalHits | 作品点击总数 |
| workTotalRecoms | 作品推荐总数 |
| workType | 作品类型（言情、玄幻、悬疑等） |
| workUrl | 作品所在网站页 |
| workWebsiteId | 作品对应网站站点 |
| workWriteProcess | 作品写作进程（更新中、已完结等） |

worksupdate主要是记录每部作品更新进度，为方便日统计，加入了woupRoughTime字段。

**表3-2 worksupdate表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 作品更新文档主键 |
| woupId | 作品更新id |
| woupWorkId | 作品id |
| woupTime | 作品更新采集时间，date类型 |
| woupWordsNum | 作品更新字数 |
| woupTotalHits | 作品最新总点击数 |
| woupTotalRecoms | 作品最新总推荐数 |
| woupCollectNum | 作品最新总收藏数 |
| woupMTickets | 作品最新月票数 |
| woupFlowersNum | 作品最新鲜花数 |
| woupCommentsNum | 作品最新评论数 |
| woupGrade | 作品最新评分（总分10.0） |
| woupUpdateTime | 作品更新时间，date类型 |
| woupAttr1/woupAttr2... | 备用字段 |
| woupRoughTime | 作品更新采集时间，string类型，具体到日 |

作者统计页需要作者总作品数、点击量、评论量、推荐量等数据，利用反范式模式设计，故增加此4个字段，作者详情页需要展示每部作品信息，所以增加works字段，为数组类型。

**表3-3 author表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 作者文档主键 |
| authArea | 作者地域 |
| authDesc | 作者描述 |
| authGender | 作者性别 |
| authId | 作者id |
| authName | 作者笔名 |
| authUrl | 作者url |
| authWebsiteId | 采集作者网站站点id |
| authWorksNum | 作者作品总数 |
| authWorksHitsNum | 作者作品总点击数 |
| authWorksCommentsNum | 作者作品总评论数 |
| authWorksRecomsNum | 作者作品总推荐数 |
| works | 作者作品集合，数组类型 |

authorupdate表主要记录每个作者更新状态，为方便统计每日作者更新数，增加auupRoughTime，该字段具体到日。

**表3-4 authorupdate表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 作者更新文档主键 |
| auupAttr1 | "非vip用户", |
| auupAttr3 | "普通会员", |
| auupAuthId | 更新作者id |
| auupGrade | 更新作者等级 |
| auupId | 更新id |
| auupTime | 作者更新采集时间，date类型 |
| auupRoughTime | 作者更新采集时间，string类型，具体到日 |

作品评论表主要记录每部作品的具体评论，通过分词分析可获得作品评论词云。

**表3-5 workscomment表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 作品评论文档主键 |
| wocoId | 作品评论id |
| wocoWorkId | 作品id |
| wocoCritic | 作品评论人 |
| wocoTitle | 评论标题 |
| wocoContent | 评论内容 |
| wocoTime | 评论时间 |

crawlwebsite表主要记录爬取的网站站点信息。

**表3-6 crawlwebsite表字段及描述**

|  |  |
| --- | --- |
| 字段名 | 描述 |
| \_id | 爬取网站站点文档主键 |
| crwsId | 网站站点id |
| crwsSiteName | 网站站点名称 |
| crwsEncode | 网站站点编码 |
| crwsDomainLimit | 限制 |
| crwsSeedUrl | 爬取的网站站点url |
| crwsCrawlStyle | 爬取方式:  1表示作品页与作者页分开爬取;  2表示作品与作者信息均在作品页爬取 |
| crwsUpdateStyle | 更新频率(与网站活跃度与用户关注度有关):  1表示每轮都爬取;  2表示每两轮爬取一次,依次类推(数字最小,更新频率越大,最大取5) |
| crwsMaxThread | 网站最大允许的线程数 |
| crwsCurrencyUnit | 网站的货币单位(一般用于打赏),例如起点币,潇湘币 |
| crwsAuthorUpdate | 需要抓取的作者信息(动态更新)  格式为："推荐数:recommendedNum:1 作品类型:type"，即多个属性用空格隔开，每个属性包含中文名（页面显示）、英文名（抽取文件使用）和数据类型（可选，1表示int型，2表示float型,3表示时间字符型，不指明默认为字符串），之间用冒号隔开。最多设置5个属性 |
| title", |  |
| crwsWorksUpdate | 需要抓取的作品信息(动态更新),  格式同crws\_authorUpdate字段，最多设置10个属性 |
| crwsExtraWorksInfo | 作品除了标题、作者等常规信息之外，其他需要抓取的信息（存入Works表的work\_otherInfo字段），例如“编辑评价:editAssess 所属系列:series），格式同crws\_authorUpdate字段（最终所有类型都会转为str保存） |
| crwsInnerFilter | 网站过滤规则 |
| crwsAuthorFilter | 作者页过滤规则 |
| crwsWorksFilter | 作品页过滤规则 |
| crwsCommentFilter | 评论抽取规则，为空表示该规则合并在了作品抽取规则中 |
| crwsAuthorExtractRule | 作者抽取规则，为xml格式，多套规则用三个#号(###)隔开。 |
| crwsWorksExtractRule | 图书抽取规则，格式同作者抽取规则。 |
| crwsCommentExtractRule | 评论抽取规则，格式同作者抽取规则。为空表示评论和作品信息在同一个html且未在单独的iframe中。 |
| crwsAuthorJsHandler | 作者页是否需要js动态处理，0表示不需要（默认），1表示需要 |
| crwsWorksJsHandler | 内容页是否需要js动态解析，同上 |
| crwsCommentJsHandler | 评论页是否需要动态js解析，同上 |
| crwsEnable | 该网站是否激活（未激活不会被爬取）,1表示激活，2表示未激活 |
| crwsParamFilter | 爬取过滤字段 |
| crwsTotalAuthorNum | 网站站点作者总数 |
| crwsTotalWorkNum | 网站站点作品总数 |
| crwsAuthorOrWorkUpdateTime | 网站站点最新更新时间 |

3.4 本章小结

本章主要介绍了MongoDB的存储策略，基于存储模型系统优化的具体策略，系统优化涉及的表从Mysql替换为MongoDB后采取优化策略后文档存储方式。通过对系统的总体设计介绍，为后期系统优化详细设计及开发做充分准备。

1. 系统优化详细设计与实现

4.1 技术难点解决

4.1.1 Spring与MongoDB的整合

本次对网络文学监测分析系统的优化，首要是将Mysql数据库替换为MongoDB，已完成大数据量存储、查询速度的优化。该系统底层使用的Spring与Mybatis的整合，Mybatis映射Mysql数据库，需要配置数据源以及映射的mapper文件和model类。而替换为MongoDB数据库，结合大部分资料发现，仅需导入mongodb驱动包及spring与mongodb的整合包，而不需要额外的sql文件，而是通过Java语言直接操作MongoDB数据库。

Spring与MongoDB整合步骤如下：

1. 导入相关jar包：

<**dependency**>  
 <**groupId**>org.mongodb</**groupId**>  
 <**artifactId**>mongo-java-driver</**artifactId**>  
 <**version**>3.4.1</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.springframework.data</**groupId**>  
 <**artifactId**>spring-data-mongodb</**artifactId**>  
 <**version**>1.9.6.RELEASE</**version**>  
</**dependency**>

1. MongoDB配置文件mongodb-config.xml如下：
2. 配置web.xml，如下：

<**context-param**>  
 <**param-name**>contextConfigLocation</**param-name**>  
 <**param-value**>  
 classpath:spring/applicationContext.xml,  
 **classpath:mongodb/mongodb-config.xml**,  
 classpath:spring/dao.xml,  
 classpath:spring/service.xml  
 </**param-value**>  
</**context-param**>

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:mongo="http://www.springframework.org/schema/data/mongo"  
 xsi:schemaLocation="http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context-4.2.xsd  
 http://www.springframework.org/schema/data/mongo  
 http://www.springframework.org/schema/data/mongo/spring-mongo-1.8.xsd  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans-4.2.xsd">  
*<!-- 加载mongodb的属性配置文件 -->*

<context:property-placeholder location="classpath:mongodb.properties" />  
*<!-- mongo对象 -->*<mongo:mongo id="mongo" host="${mongo.hosts}" port="${mongo.port}" />  
<mongo:db-factory id="mongoDbFactory" dbname="${mongo.databaseName}" mongo-ref="mongo"/>  
 <bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate">  
 <constructor-arg ref="mongoDbFactory"/>  
 </bean>  
</beans>

4.1.2数据的迁移

由于需要将Mysql数据库迁移至MongoDB数据库，由于MongoDB文档字段设计不同于Mysql表结构的设计，所以只能通过代码方式跑数据，由于数据量大，本地电脑配置不高，寻找了好几种方式均不妥当，比如可能导致堆溢出，可能耗时过长等，几经周折，寻找到较为快速的方法如下：

**public void** mysql2Mongo(String databaseName, String colName, String space, Integer pageSize){  
 **try**{

*//获取mongodb连接* MongoDatabase database = getMongoConn(databaseName);  
 MongoCollection collection = database.getCollection(colName);  
 *//获取记录条数* GenericQuery query = **new** GenericQuery();  
 **int** rows = **mysqlDao**.selectAllCount(space, query);  
 **int** count = rows/pageSize;  
 **if**(count>0){  
 **int** startIndex=0, i=0;  
 **while**(i<=count){  
 startIndex = (rows-startIndex)>pageSize?i\*pageSize:rows;  
 *//分段查询* query = **new** GenericQuery();  
 query.put(**"startIndex"**, startIndex);  
 query.put(**"pageSize"**, pageSize);  
 List list = **mysqlDao**.selectAllData(space, query);  
 *//存入数据到mongo* collection.insertMany(saveDocuments(list));  
 i++;  
 }  
 }**else**{  
 List list = **mysqlDao**.selectAllData(space, query);  
 collection.insertMany(saveTemp(list));  
 }  
}**catch** (Exception e){  
 e.printStackTrace();  
}

}

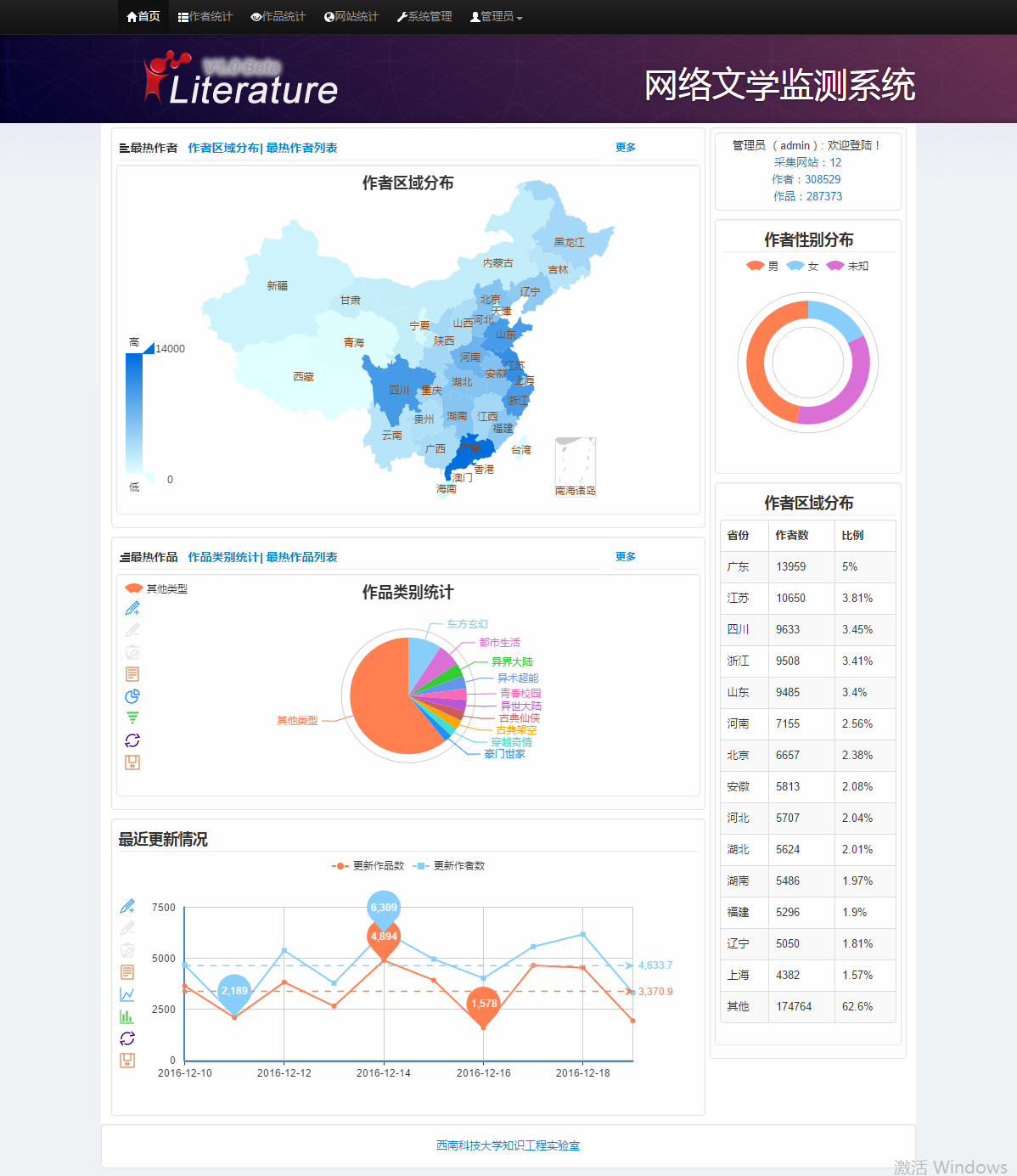
**private** List<Document> saveDocuments (List<Object> list) {  
 List<Document> documents = Lists.newArrayList();  
 **for**(Object obj:list){  
 Gson gson = **new** Gson();  
 String auupStr = gson.toJson(obj);   
 Document document = Document.parse(auupStr);  
 documents.add(document);  
 }  
 **return** documents;

}

4.2 系统模块设计与优化

4.2.1 网站首页统计分析模块

根据用户的需求，将作者作品重要的统计信息展示于首页页面。首页效果图如下：



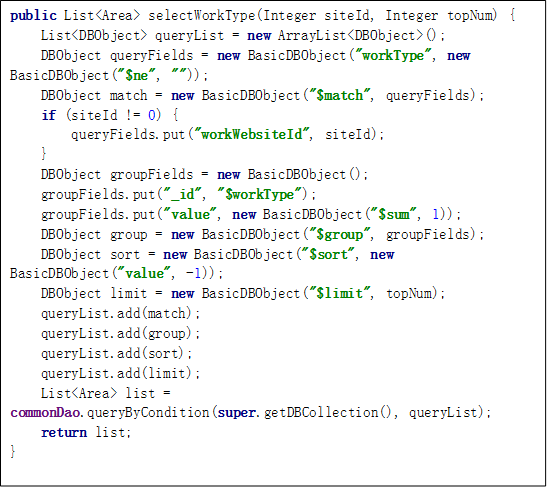
**图4-1 首页效果图**

由于功能较为多，只选几个典型的列举。

首页展示作者区域人数统计及比例统计关键代码如下：

**public** List<Area> countInfoArea(Integer wesiId) {  
 DBObject queryFields = **new** BasicDBObject();  
 queryFields.put(**"\_id"**, **"$authArea"**);  
 queryFields.put(**"value"**, **new** BasicDBObject(**"$sum"**, 1));  
 DBObject group = **new** BasicDBObject(**"$group"**, queryFields);  
 List<DBObject> list = Lists.*newArrayList*();  
 **if** (wesiId != 0) {  
 DBObject match = **new** BasicDBObject(**"$match"**, **new** BasicDBObject(**"authWebsiteId"**, wesiId));  
 list.add(match);  
 }  
 list.add(group);  
 List<Area> areaList = **commonDao**.queryByCondition(**super**.getDBCollection(), list);  
 **return** areaList;  
}

首页作品类别占比统计关键代码如下：



首页近期作者作品更新数量统计shell命令及关键代码如下：

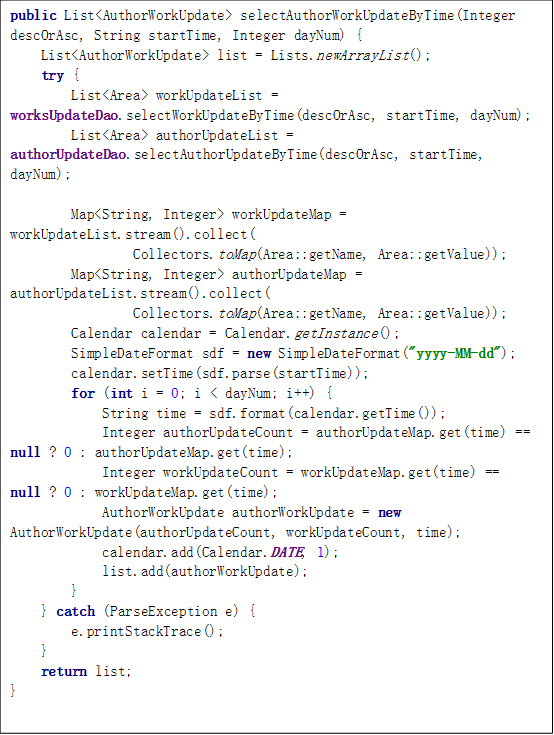
1. 作者、作品更新集合的聚合操作shell命令

*db.authorupdate.aggregate([*

*{"$match":{"woupRoughTime":{"$gte":startTime, "lt":endTime}}},  
{$group:{"\_id":"$auupRoughTime", "value":{$sum:1}}},  
{"$sort":{"auupRoughTime":-1}},  
{"$limit":dayNum} ]);*

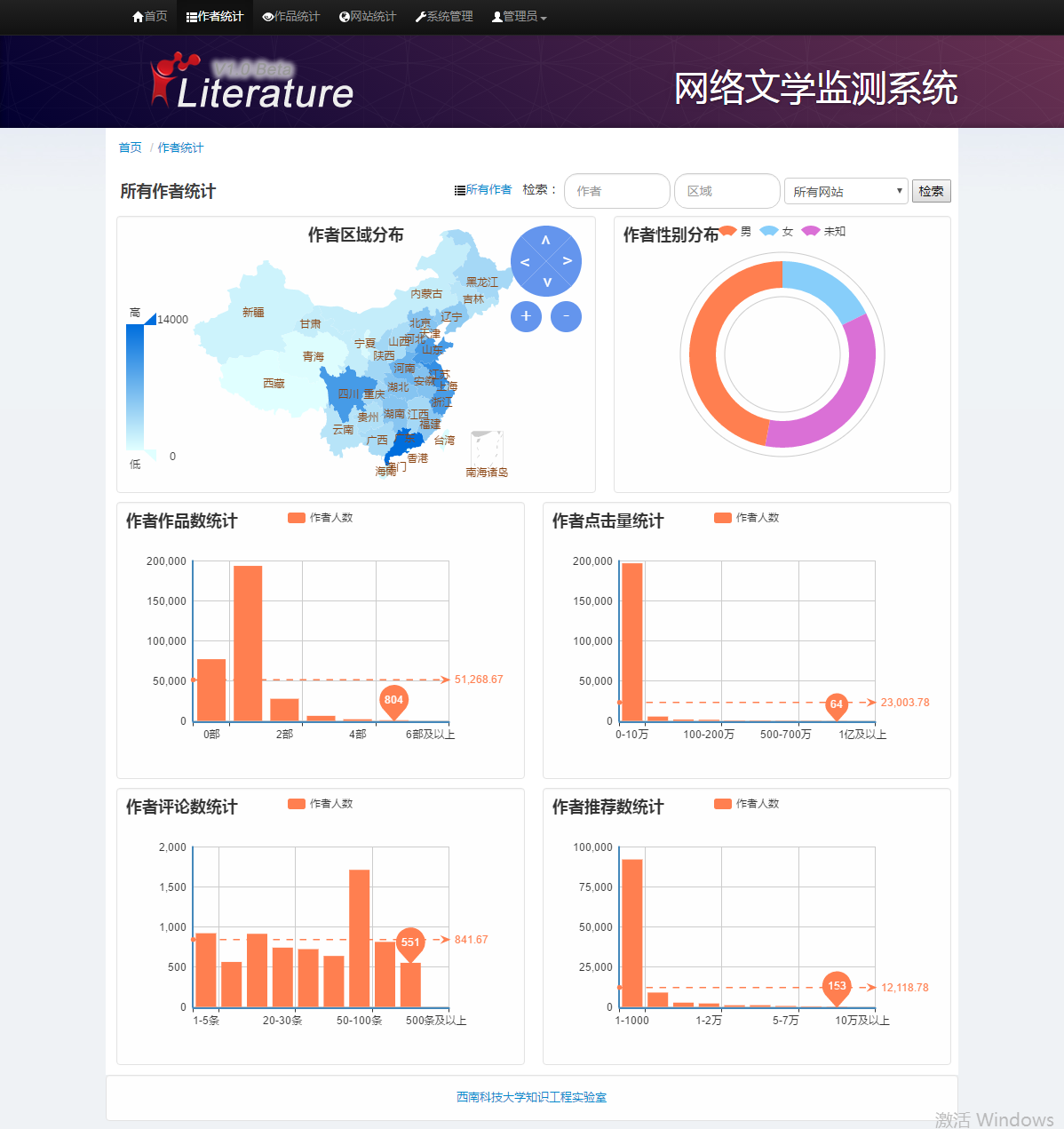
*db.worksupdate.aggregate([  
{"$match":{"woupRoughTime":{"$gte":startTime, "lt":endTime}}},  
{"$group":{"\_id":"$woupRoughTime", "value":{$sum:1}}},  
{"$sort":{"woupRoughTime":-1}},  
{"$limit":dayNum} ]);*

1. 作者、作品近期日更新数量统计关键代码



4.2.2 网站作者相关统计分析模块

作者统计分析包括3个部分，即作者统计页面，作者列表页面及作者详情页面。分别如下图：



**图4-2 作者统计页效果图**

作者统计页主要统计作者的作品数/点击数/评论数/推荐数量，以柱状图形式展示，可以清晰的了解大多数作者的相关数据的分布范围。如上图，60%以上的作者都是只发布了一部作品等等信息。其关键代码如下：

**public** Area selectRangeWithAuthorCount(Integer siteId, Integer sortField, Integer min, Integer max) {  
 *// shell代码  
/\*db.author.find({{"authWebsiteId":siteId},{"authWorksHitsNum":{"$gte":min,"$lt":max}}});\*/* String fieldStr = **""**;  
 **if** (sortField == 1) {  
 fieldStr = **"authWorksHitsNum"**;  
 } **else if** (sortField == 2) {  
 fieldStr = **"authWorksCommentsNum"**;  
 } **else if** (sortField == 3) {  
 fieldStr = **"authWorksRecomsNum"**;  
 } **else if** (sortField == 4) {  
 fieldStr = **"authWorksNum"**;  
 }  
 DBObject query = **new** BasicDBObject();  
 **if** (siteId != 0) {  
 query.put(**"authWebsiteId"**, siteId);  
 }  
 query.put(fieldStr, **new** BasicDBObject().append(**"$gte"**, min).append(**"$lt"**, max));  
 **int** authorCount = **super**.getDBCollection().find(query).count();  
 Area area = **new** Area(min.toString(), authorCount);  
 **return** area;  
}

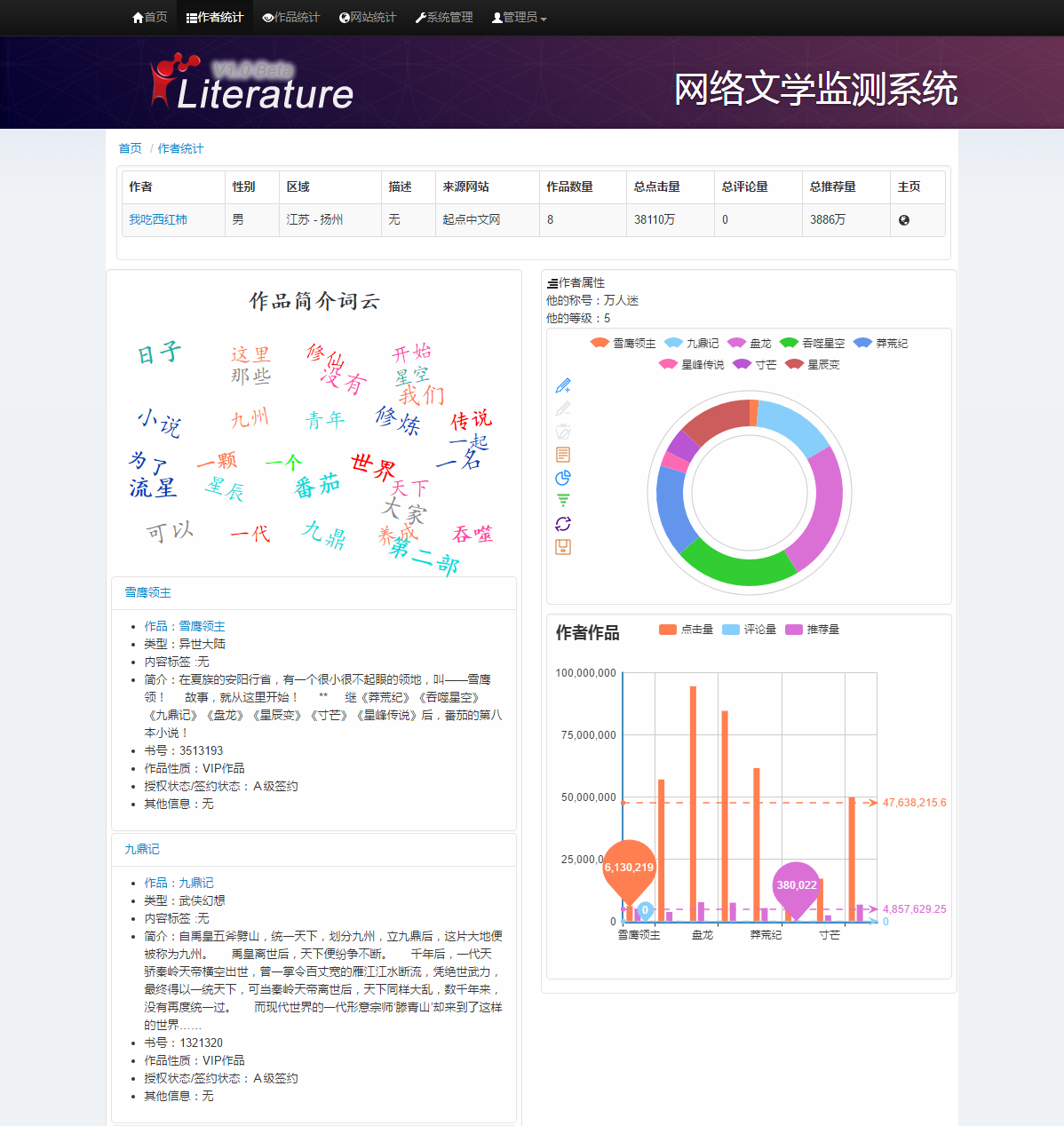
以上为获取作者点击量统计数据关键代码，其余雷同，就不一一展示了。



**图4-3 作者列表页效果图**

由于改代码中含有分页功能实现，所以就只展示shell命令，如下：

*db.works.aggregate([  
{"$match":{"workWebsiteId": siteId}},  
{"$sort":{"workTotalHits":-1}},  
{"$skip":i\*perPageCount},  
{"$limit":perPageCount},  
{"$lookup":{  
 from: "crawlwebsite",  
 localField: "workWebsiteId",  
 foreignField: "crwsId",  
 as: "crawlWebsiteList"  
}}]);*



**图4-4 作者详情页效果图**

该页面作品简介关键代码如下：

**public** List<WorkDescription> selectWorksDescByAuthId(Integer authId) {  
 List<TempWorks> worksList = **worksInfoDao**.selectWorksByAuthId(authId);  
 List<WorkDescription> workDescList = Lists.*newArrayList*();  
 worksList.forEach(work -> {  
 WorkDescription workDesc = **new** WorkDescription();  
 workDesc.setWorkId(work.getWorkId());  
 workDesc.setDescription(work.getWorkDesc());  
 workDescList.add(workDesc);  
 });  
 String[] allCommentWords = **null**;  
 **for** (**int** i = 0; i < workDescList.size(); i++) {  
 allCommentWords = (String[]) ArrayUtils.*addAll*(allCommentWords, workDescList.get(i).**Words**);  
 }  
 DescriptionFreqs freqs = **new** DescriptionFreqs();  
 List<WorkDescription> wordcloud = freqs.getHighFreqWords(allCommentWords);  
 **return** wordcloud;  
}

4.2.3 网站作品统计分析模块

作品统计分析包括3个部分，即作品统计页面，作品列表页面及作品详情页面。除作品详情页面，其余页面功能与作者统计、作者列表页面处理方式类似，故只贴出作品详情页面的关键代码。分别如下图：



**图4-5 作品统计页效果图**



**图4-6 作品列表页效果图**



**图4-7 作品详情页效果图**

周统计实现关键代码如下：

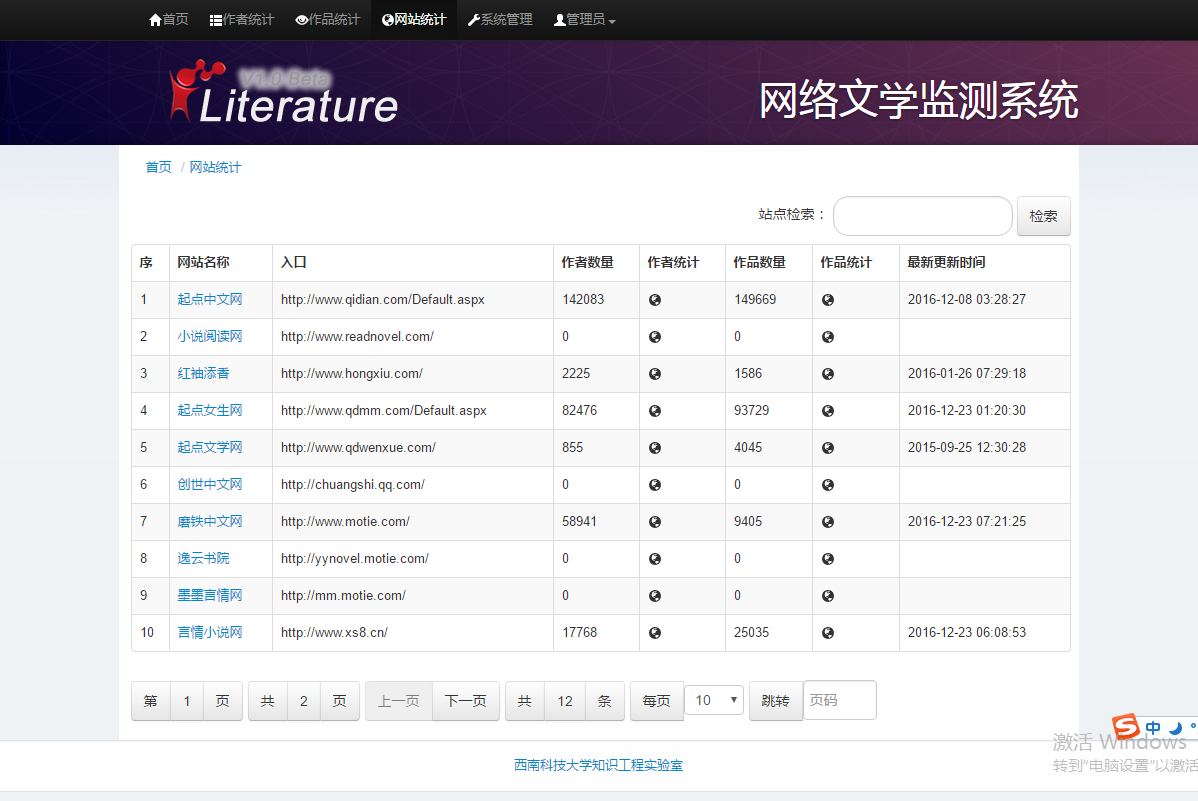
**public** List<TempWorksUpdate> selectWeekOfWorkInfo(Integer workId, String startTime, String endTime) {  
 DBObject queryFields = **new** BasicDBObject(**"woupWorkId"**, workId);  
 queryFields.put(**"woupRoughTime"**, **new** BasicDBObject(**"$gt"**, startTime).append(**"$lte"**, endTime));  
 DBObject sort = **new** BasicDBObject(**"woupRoughTime"**, 1);  
 DBCursor cursor = **super**.getDBCollection().find(queryFields).sort(sort);  
 List<TempWorksUpdate> worksUpdateList = Lists.*newArrayList*();  
 **while**(cursor.hasNext()){  
 TempWorksUpdate worksUpdate = decode(cursor.next(), TempWorksUpdate.**class**);  
 worksUpdateList.add(worksUpdate);  
 }  
 **return** worksUpdateList;  
}

4.2.4 网站站点统计模块

网站统计只有管理员才有权限查看。

网站站点信息统计原本是通过site，author，works，authorupdate，worksupdate五表嵌套加关联查询查询出的结果，再以分页形式展示。将其利用反范式模式设计原理，数据以文档存入MongoDB时，文档中添加了作者作品总数统计以及最近更新时间字段，以便查询。

网站站点效果图如下：



**图4-8 网站统计页效果图**

4.3 本章小结

本章对网络文学监测分析系统优化中遇到的技术难点及系统优化的模块进行了介绍。对每个模块涉及的页面效果图进行了展示，由于涉及的功能点较多，所以仅选取了部分页面较为复杂功能的关键代码实现。最后对比了接口优化前后的性能，从数据层面能够明显的看出优化后性能的提升。通过本章对系统优化详细设计与实现的介绍，可更深入了解系统功能。

1. 系统测试

5.1 测试原则

当开发人员已经将全部模块开发完成，各单元模块已经过开发人员代码检查和程序互测（单元测试），单元测试后，各单元模块集成在一起，独立测试正式开始。

系统测试准则为实现了本项目的软件需求规格说明书中所有功能，在操作使用中没有出现错误，系统运行及业务处理正常。确保测试的功能正常，其中包括导航，数据输入，处理和检索等功能。测试中所发现的错误已经全部得到修复。非功能需求（包括性能、安全性等）满足需求规格说明书中规定要求。

当所测内容均都达到软件需求规格说明书要求，所发现的问题已全部解决。遗留问题中无致命性和严重性错误，一般性错误不能超过总数的2％，轻微性错误不能超过8％，遗留问题总数不能超过问题总数的10％时，测试可结束，当系统在部署期间或使用过程中发现严重错误或客户需求已发生重大变化时，测试可以再启动。

5.2 测试环境

硬件环境：Core i5-3210M双核处理器 2.5GHz 4GB DDR3内存

软件环境：Windows7旗舰版

IntelliJ IDEA 2016.2.2

JDK 1.8.0

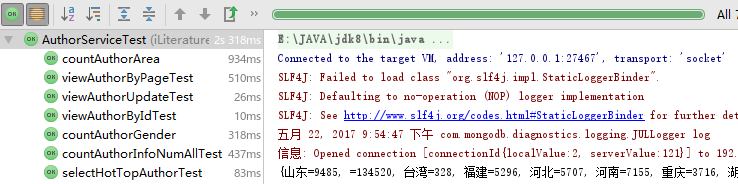
Google Chrome 56.0.2924.87 m

Microsoft IE 8.0

5.3 测试结果

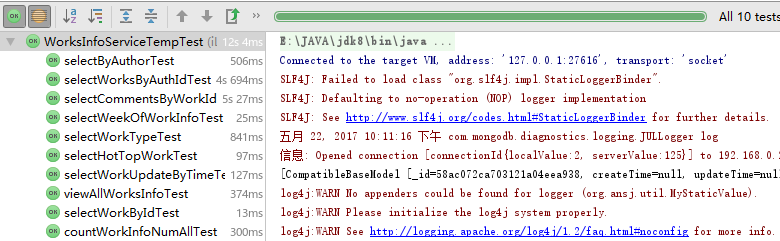
5.3.1 单元测试

作者相关功能，如作者区域分布统计，作者性别统计，作者作品数/点击量/评论数/推荐数，作者列表等功能接口的单元测试结果如图：



**图5-1 作者相关功能单元测试图**

作品相关功能，如作品类别占比统计，Top10作品，作品点击量/评论数/推荐数统计，作品列表，作品更新状态周统计等功能接口的单元测试结果如图：



**图5-2 作品相关功能单元测试图**

5.3.2 接口测试

数据是最能说话的武器，由于接口太多，故只选取部分接口列举，如下表：

**表5-1 部分接口优化前后性能对比表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 优化前接口名称 | 优化后接口名称 | 描述 | 优化前响应时间(ms) | 优化后响应时间(ms) |
| countInfoArea | countInfoArea | 作者区域分布人数统计 | 10250 | 325 |
| selectWorkType | selectWorkType | 作品类型占比统计 | 1960 | 812 |
| selectNovelUpdateByTime | selectWorkUpdateByTime | 作者作品更新状态 | 38430 | 173 |
| countInfoNumAll | countAuthorInfoNumAll | 作者作品数/点击量/评论数/推荐数统计 | 6970 | 237 |
| viewAllAuthor | viewAllAuthor | 作者列表 | 38080 | 216 |
| selectByAuthor\_description | selectWorksByAuthId | 作者详情的作品简介词云 | 2720 | 142 |
| selectWorkComment;  selectWorkRecom;  selectWorkHit | countWorkInfoNumAll | 作品评论数统计  作品推荐数统计  作品点击量统计 | 21850;  23530;  23640 | 186 |
| viewAllWorksInfo | viewAllWorksInfo | 作品列表 | 456000 | 832 |
| viewAuthorAndWorkNum | viewAuthorAndWorkNum | 网站统计列表 | 1340 | 28 |
| commentsByWork | selectCommentsByWorkId | 作品详情的评论词云 | 335 | 223 |

5.3.3 查询优化测试

针对网络文学监测分析系统常见的查询，查询某站点的作品信息及其最近更新时间，按点击量排序。查询命令如下：

db.works.find({workWebsiteId:1}).sort({workTotalHits:-1}).limit(10000);

该查询包括查询条件字段workWebsiteId和排序字段workTotalHits。分别有三种查询计划：workWebsiteId单键索引、workTotalHits单间索引及{workWebsiteId, workTotalHits}复合索引。在30万数据量的情况下测试该查询的三种查询计划优化情况。

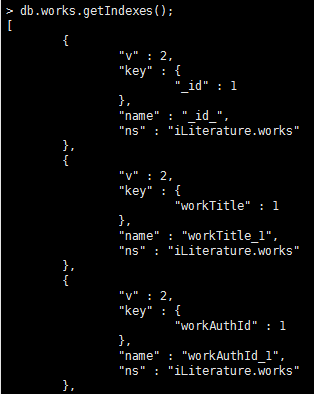
首先在iLiterature数据库创建单键索引workWebsiteId、workTotalHits和复合索引{workWebsiteId, workTotalHits}，命令如下：

db.works.ensureIndex({workWebsiteId:1});

db.works.ensureIndex({workTotalHits:-1});

db.works.ensureIndex({workWebsiteId:1, workTotalHits:-1});

利用MongoDB提供的db.works.getIndexes()可以查看works集合的索引，如下图：



**图5-3 works集合索引图**

索引建立完成后，使用hint命令指定查询使用的索引，测试对应索引的查询耗时。由于数据量过大时可能超出MongoDB配置的最大内存，导致执行的shell命令报错，故仅选取3万作品作为本次索引测试的基础数据，针对每种索引测试20次，计算平均值。测试索引命令如下：

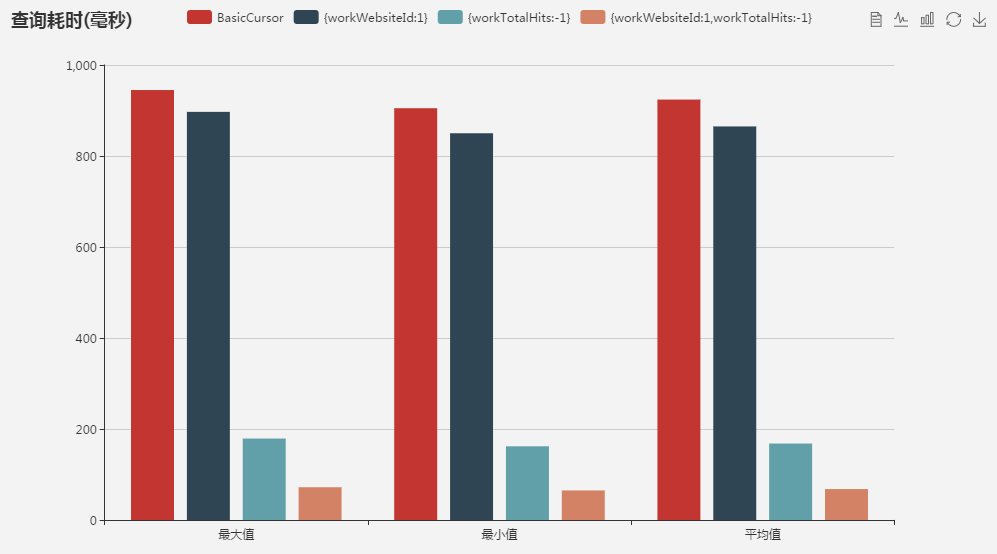
db.works.find({workWebsiteId:1}).limit(30000).sort({workTotalHits:-1}).hint({workWebsiteId:1}).explain(true);

workWebsiteId单键索引、workTotalHits单间索引及{workWebsiteId, workTotalHits}复合索引此三种索引执行的最大值、最小值及平均值耗时如下表：

**表5-2 索引查询性能对比表**

|  |  |  |  |
| --- | --- | --- | --- |
| 查询计划 | 最大值(ms) | 最小值(ms) | 平均值(ms) |
| BasicCursor | 945 | 905 | 924 |
| {workWebsiteId:1} | 897 | 850 | 865 |
| {workTotalHits:-1} | 179 | 162 | 168 |
| {workWebsiteId:1,workTotalHits:-1} | 72 | 65 | 68 |

无索引及三种索引耗时对比如下图：



**图5-4 索引耗时对比图**

5.3.4 数据模型性能测试

根据前面介绍的优化策略，可得知本次针对网络文学监测分析系统的优化采用的MongoDB的文档数据模型采用的是反范式模式设计，而原本使用的Mysql中表结构采用的是三范式模式设计。

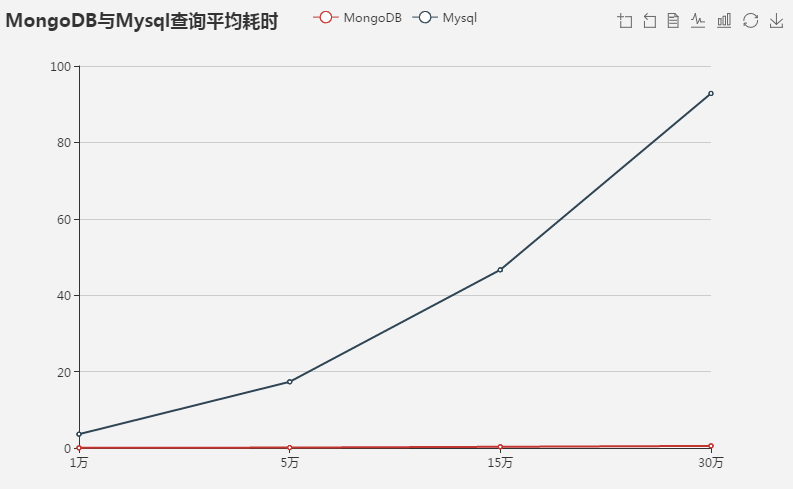
针对数据模型的性能测试，采用最常见的查询：按点击量/评论数/推荐数排序的作品信息及其最近更新时间。基于最优索引的前提，在数据量1万，5万，15万，30万的情景下分别执行20次查询，取其耗时平均值，比较MongoDB与Mysql的性能。

如下表，给出了Mysql与MongoDB的耗时最大值、耗时最小值及耗时平均值。

**表5-3 MognoDB与Mysql查询性能对比表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据量（万） | 耗时最大值（s） | | 耗时最小值（s） | | 耗时平均值（s） | |
| MongoDB | Mysql | MongoDB | Mysql | MongoDB | Mysql |
| 1 | 0.071 | 3.689 | 0.020 | 3.613 | 0.032 | 3.621 |
| 5 | 0.146 | 17.367 | 0.104 | 17.300 | 0.113 | 17.326 |
| 15 | 0.324 | 46.679 | 0.306 | 46.603 | 0.316 | 46.644 |
| 30 | 0.562 | 92.943 | 0.532 | 92.755 | 0.547 | 92.811 |

MongoDB与Mysql查询平均耗时对比图如下：



**图5-5 MongoDB与Mysql查询平均耗时图**

综合测试结果的图表可知，非关系型数据库MongoDB的查询性能明显高于关系型数据库Mysql的查询性能。随着数据量的增长，Mysql查询性能以几何倍数增长，耗时严重，而MongoDB性能增长不明显，所以采用MongoDB反范式模式设计数据存储模型策略，避免了多表联查，对本系统优化是成功且有成效的。

5.4 本章小结

本章首先介绍了测试原则、环境，着重网络文学监测分析系统优化后的性能测试，如索引探索与创建的测试和反范式模式设计的数据存储模型的测试，通过图表数据对比形式，强而有力的说明了针对本系统的优化效果是明显且有效的。

结论

本文首先介绍了本课题的研究背景与研究意义，以及研究需要用到的技术的研究现状，并介绍了课题的目标和研究范围。在需求分析阶段对系统进行了简单的介绍，然后从经济和技术上分析了系统优化的可行性。对系统待优化的功能进行了详细的介绍并表明优化需达到的指标，以保证此次优化不是做无用功而是性能能够得到明显提升。除此，还交代了系统非功能性需求及运行环境。通过对基于NoSQL的网络文学监测分析系统优化进行的需求分析，为后期的系统优化设计及实现做充分准备。在系统优化概要设计章节，对接下来即将使用的非关系型数据库MongoDB进行了简单的介绍，详细介绍了其数据存储模型。针对MongoDB不同于关系型数据库的数据存储模型，详细介绍了本次系统优化采用的策略，包括反范式模式设计技术、索引技术。与此同时，介绍了从Mysql替换为MongoDB数据库涉及的6张数据表的表结构设计，该表结构不同于关系型数据库，表字段为非固定字段。紧接着介绍了系统优化的详细设计与实现，首先介绍了本次优化的技术难点及解决方案，然后分模块详细介绍了系统的优化。最后，在系统测试阶段，简单介绍了测试的原则和环境，着重测试了索引优化的优化前后性能对比、利用反范式模式设计的数据存储模型的优化前后性能对比，从测试结果的图表来看，本次基于NoSQL的网络文学监测分析系统优化是达到预期指标的，且效果显著。

致谢

参考文献

[1] 张华强. 关系型数据库与NoSQL数据库[J]. 电脑知识与技术,2011,(20):4802-4804.

[2] 沈姝. NoSQL数据库技术及其应用研究[D].南京信息工程大学,2012.

[3] 陈明. NoSQL数据库系统[J]. 计算机教育,2013,(11):107-111.

[4] 刘卓. 基于NoSQL的空间数据云存储的研究[D].河南大学,2014.

[5] 饶翔. 面向海量数据的多数据库并行查询关键技术研究与实现[D].国防科学技术大学,2006.

[6] 田爱雪. 基于海量数据存储的性能测试与优化研究[D].长春理工大学,2014.

[7] 覃涛. 海量数据查询的优化处理及其在用电稽查系统中的应用[D].中南大学,2012.

还未完整