北京信息科技大学

**毕业设计（论文）**

**题 目：** 基于Hadoop和Spark的统计分析系统的实现

**学 院：**

**专 业：**

**学生姓名：** **班级/学号**

**指导老师/督导老师：**

**起止时间：** 年 月 日 **至**  年 月 日

（以上宋体四号居中）

**摘 要**

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□。

**关键词：**Hadoop； Spark ；统计分析 ；消费投诉； 管理系统；

**Abstract**

（Times New Roman、16磅、加粗、居中）

（Times New Roman 、12磅）

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□。

**Keywords:** （ 3-8 words） □□□□□□;□□□□□□;□□□□□□;□□□□□□;□□□□□□;□□□□□□;□□□□□□

注：**Keywords:** 12磅加粗 ； □□□□□□ 12磅

**目 录**

（宋体、三号、加粗、居中）

**摘要** （中文） …………………………………………………………………………I

（英文） …………………………………………………………………………II

**第一章** □□□□□□□□□□□□□□□ ……………………………………

1.1 □□□□□□□□□□□□□□□□ ……………………………………

1.2 □□□□□□□□□□□□□□□□ ……………………………………

**第二章** □□□□□□□□□□□□□□□□ ……………………………………

2.1 □□□□□□□□□□□□□□□□ ……………………………………

2.2 □□□□□□□□□□□□□□□□ ……………………………………

∶

∶

**第X章** □□□□□□□□□□□□□□□□ …………………………………

**结束语** …………………………………

**参考文献** …………………………………

注：1 摘要、章、结束语、参考文献：（宋体、小四、加粗、居左）；

2 节：（宋体、五号）；

3 中文摘要、英文摘要、目录的页码依次编排用“Ⅰ、Ⅱ、Ⅲ …”罗马数字（宋体、小五号、居中、底部）。

**第一章 概述**

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□（宋体、五号）

**1.1 □□□□□□□□□□□□□□**（宋体、小四、加粗）

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□（宋体、五号）

**1.2**  **□□□□□□□□□□□□□□**（宋体、小四、加粗）

**1.2.1 □□□□□□□□□□□□**（宋体、五号、加粗）

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

**1.2.2 □□□□□□□□□□□□**

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

（行距：最小值，18磅）

**第二章 相关知识和技术**

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□（宋体、五号）

**2.1 Hadoop平台简介**

Apache Hadoop 是用于开发在分布式计算环境中执行的数据处理应用程序的框架。类似于在个人计算机系统的本地文件系统的数据，在 Hadoop 数据保存在被称为作为Hadoop分布式文件系统的分布式文件系统。

2.1.1 Apache Hadoop项目组成

1） Hadoop MapReduce : MapReduce 是一种计算模型及软件架构，编写在Hadoop上运行的应用程序。这些MapReduce程序能够对大型集群计算节点并行处理大量的数据。

2） HDFS (Hadoop Distributed File System): HDFS 处理 Hadoop 应用程序的存储部分。 MapReduce应用使用来自HDFS的数据。 HDFS创建数据块的多个副本，并集群分发它们到计算节点。这种分配使得应用可靠和极其迅速的计算。

虽然 Hadoop 是因为 MapReduce 和分布式文件系统 - HDFS 而最出名的， 该术语也是在分布式计算和大规模数据处理的框架下的相关项目。 Apache Hadoop 的其他相关的项目包括有：Hive, HBase, Mahout, Sqoop , Flume 和 ZooKeeper.

2.1.2 Hadoop 功能

• 适用于大数据分析

作为大数据在自然界中趋于分布和非结构化，Hadoop 集群最适合于大数据的分析。因为，它处理逻辑(未实际数据)流向计算节点，更少的网络带宽消耗。这个概念被称为数据区域性概念，它可以帮助提高基于 Hadoop 应用程序的效率。

• 可扩展性

HADOOP集群通过增加附加群集节点可以容易地扩展到任何程度，并允许大数据的增长。 另外，标度不要求修改到应用程序逻辑。

• 容错

HADOOP生态系统有一个规定，来复制输入数据到其他群集节点。这样一来，在集群某一节点有故障的情况下，数据处理仍然可以继续，通过使用存储另一个群集节点上的数据。

2.1.3 Hive介绍：

Hive是一个数据仓库基础工具在Hadoop中用来处理结构化数据。它架构在Hadoop之上，总归为大数据，并使得查询和分析方便。

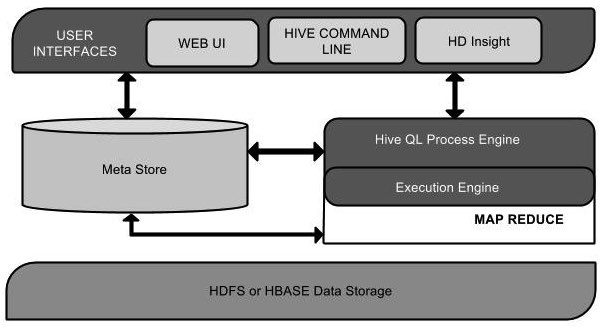
最初，Hive是由Facebook开发，后来由Apache软件基金会开发，并作为进一步将它作为名义下Apache Hive为一个开源项目。它用在好多不同的公司。例如，亚马逊使用它在 Amazon Elastic MapReduce。

Hive特点：

* 它存储架构在一个数据库中并处理数据到HDFS。
* 它是专为OLAP设计。
* 它提供SQL类型语言查询叫HiveQL或HQL。
* 它是熟知，快速，可扩展和可扩展的。

Hive架构

下面的组件图描绘了Hive的结构：



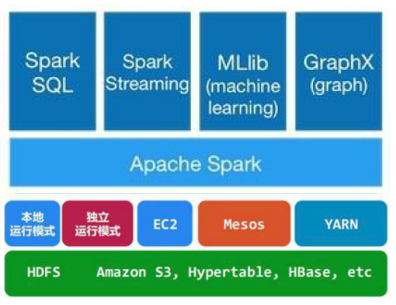
该组件图包含不同的单元。下表描述每个单元：

|  |  |
| --- | --- |
| 单元名称 | 操作 |
| 用户接口/界面 | Hive是一个数据仓库基础工具软件，可以创建用户和HDFS之间互动。用户界面，Hive支持是Hive的Web UI，Hive命令行，HiveHD洞察（在Windows服务器）。 |
| 元存储 | Hive选择各自的数据库服务器，用以储存表，数据库，列模式或元数据表，它们的数据类型和HDFS映射。 |
| HiveQL处理引擎 | HiveQL类似于SQL的查询上Metastore模式信息。这是传统的方式进行MapReduce程序的替代品之一。相反，使用Java编写的MapReduce程序，可以编写为MapReduce工作，并处理它的查询。 |
| 执行引擎 | HiveQL处理引擎和MapReduce的结合部分是由Hive执行引擎。执行引擎处理查询并产生结果和MapReduce的结果一样。它采用MapReduce方法。 |
| HDFS 或 HBASE | Hadoop的分布式文件系统或者HBASE数据存储技术是用于将数据存储到文件系统。 |

**1.2 Spark简介**

Apache Spark是一个围绕速度、易用性和复杂分析构建的大数据处理框架，最初在2009年由加州大学伯克利分校的AMPLab开发，并于2010年成为Apache的开源项目之一，与Hadoop和Storm等其他大数据和MapReduce技术相比，Spark有如下优势：

Spark提供了一个全面、统一的框架用于管理各种有着不同性质（文本数据、图表数据等）的数据集和数据源（批量数据或实时的流数据）的大数据处理的需求。官方资料介绍Spark可以将Hadoop集群中的应用在内存中的运行速度提升100倍，甚至能够将应用在磁盘上的运行速度提升10倍。通常当需要处理的数据量超过了单机尺度(比如我们的计算机有4GB的内存，而我们需要处理100GB以上的数据)这时我们可以选择spark集群进行计算，有时我们可能需要处理的数据量并不大，但是计算很复杂，需要大量的时间，这时我们也可以选择利用spark集群强大的计算资源，并行化地计算，其架构示意图如下：



* Spark Core：包含Spark的基本功能；尤其是定义RDD的API、操作以及这两者上的动作。其他Spark的库都是构建在RDD和Spark Core之上的
* Spark SQL：提供通过Apache Hive的SQL变体Hive查询语言（HiveQL）与Spark进行交互的API。每个数据库表被当做一个RDD，Spark SQL查询被转换为Spark操作。
* Spark Streaming：对实时数据流进行处理和控制。Spark Streaming允许程序能够像普通RDD一样处理实时数据
* MLlib：一个常用机器学习算法库，算法被实现为对RDD的Spark操作。这个库包含可扩展的学习算法，比如分类、回归等需要对大量数据集进行迭代的操作。
* GraphX：控制图、并行图操作和计算的一组算法和工具的集合。GraphX扩展了RDD API，包含控制图、创建子图、访问路径上所有顶点的操作

Spark架构的组成图如下：

* Cluster Manager：在standalone模式中即为Master主节点，控制整个集群，监控worker。在YARN模式中为资源管理器
* Worker节点：从节点，负责控制计算节点，启动Executor或者Driver。
* Driver： 运行Application 的main()函数
* Executor：执行器，是为某个Application运行在worker node上的一个进程

Spark本身并没有提供分布式文件系统，因此spark的分析大多依赖于Hadoop的分布式文件系统HDFS。Hadoop的Mapreduce与spark都可以进行数据计算，而相比于Mapreduce，spark的速度更快并且提供的功能更加丰富

**1.3 SSM简介**

**1.3.1 Spring**

Spring就像是整个项目中装配bean的大工厂，在配置文件中可以指定使用特定的参数去调用实体类的构造方法来实例化对象。

Spring的核心思想是IoC（控制反转），即不再需要程序员去显式地`new`一个对象，而是让Spring框架帮你来完成这一切。

**1.3.2 SpringMVC**

SpringMVC在项目中拦截用户请求，它的核心Servlet即DispatcherServlet承担中介或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller，Controller就是具体对应请求所执行的操作。SpringMVC相当于SSH框架中struts。

**1.3.3 mybatis**

mybatis是对jdbc的封装，它让数据库底层操作变的透明。mybatis的操作都是围绕一个sqlSessionFactory实例展开的。mybatis通过配置文件关联到各实体类的Mapper文件，Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

**1.2.2 现有相关平台应用案例**

我国国家工商总局在2013年10月就对外正式公布了企业发展工商指数，作为我国政府探索大数据技术的首创成果，企业发展工商指数运用大数据技术，进行数据挖掘，实时全量的分析工商登记信息，为政府部门运用大数据技术提供了可以借鉴的道路。并且现在公安部门、法律部门，也在原有的系统中加入大数据的技术。例如警察部门通过引入大数据技术。系统可以通过对交通出行数据、个体行为数据、社交数据等，找出与某个人关联较紧密的人，这一功能极大的加大了公安部门执法办案的效率。

而消费者投诉作为工商部门面向群众最重要的一环，诸多地方部门政府也积极电子信息化，现在各个省都有在线的投诉平台，但是工商部门没有对应分析投诉信息的相关系统。

**第三章 系统需求分析**

需求分析的结果是软件开发阶段的基础，因此它关系到软件开发的成败和软件的质量。软件的需求包括四个不同的层次：业务需求、用户需求、功能性需求和非功能性需求。本章依据软件需求的四个层次，对学生事务数据分析系统进行需求分析，其中也对数据仓库的需求作了相应的概述。

**3.1 业务需求**

业务需求从总体上描述开发这样一个系统的原因，即组织或者客户高层次的目标。业务需求是系统整体目标，是某个组织或者客户对将要被开发系统的期望。大数据统计分析系统旨在帮助消费者和工商管理者快速了解投诉发生的倾向和状况，这就需要对数据进行复杂的多维度的分析。随着信息化的普及和大数据技术的发展，开发一个基于大数据的统计分析系统变成了最佳的选择。

经过深入研究和分析，可以归纳出12315统计分析平台的业务需求如下：

1. 投诉分析。消费者和工商部门工作人员需要知道一段时间内投诉的数量变化。如果发生一定时间内某公司或者某类产品的投诉量较以往突然增多，可以让消费者和工商部门工作人员迅速反应解决问题，在造成更大的社会影响前解决相应问题。同时需要对投诉总量分析。消费者和工商部门工作人员需要知道在一段时间内的不同维度下的投诉量的数量。比如近一个月投诉数量最多的公司，这样消费者和部门工作人员才能针对性的处理。
2. 投诉查看及管理。消费者需要在该平台发起投诉，而工商部门管理者则需要在平台查看以及处理投诉。

**3.2**  **用户需求**

通过对系统的业务需求进行分析，确定了系统的业务范围。用户需求分析是在系统的业务范围内对业务进行进一步的具体化分析，从而得出用户所要求系统必须完成的功能。

**3.2.1 统计需求分析**

作为系统的主要功能，统计分析需要在多个维度下进行。

1. 公司分析

消费者和工商部门工作人员对大量的投诉数据，首先一点就是要知道在近期或者特定时间段内有哪些公司频频遭到投诉，统计分析系统就需要分析出这段时间内被投诉最多的公司，以方便工商管理人员监管。同时也需要针对某个公司做分析，不仅可以让管理者看到公司的长期质量监管的力度，还能很好的规范市场。

1. 类别分析

消费者和工商部门经常也需要对某一种类的投诉进行分析。比如消费者想知道近一段时间内奶制品哪些公司投诉比较多，这样消费者就可以避开有问题产品的公司。而工商部门也可以根据这一分析来监管企业，并及时通报群众。

1. 性别分析
2. 金额分析
3. 问题类型分析
4. 诉求类型分析

3.2.2 用户登录模块需求分析

3.3 功能性需求描述

功能需求描述了一个系统必须提供的活动和服务，即开发人员必须在产品中实现的软件功能，用户利用这些功能来完成各自的任务，满足业务需求。本文采用用例方法来描述系统的功能性需求。用例模型是遵循面向对象的思想，运用统一建模语言建立的一种无歧义信息模型，并被广泛用于需求描述和获取的方法。要想建立用例模型，其关键是找出参与者、用例以及定义用例的过程。用例模型主要包括用例图和用例规约。

系统主要用例图

由业务需求和用户需求可知，本系统主要需要进行统计分析、用户管理等功能，系统主要用例图如图所示，包含了各个角色的主要功能，通过对主要用例的分析，得出系统的功能需求

用例规约是描述每一个用例的详细信息。每个用例都应该有一个与之相对应的用例规约文档，文档中应准确描述用例的细节内容。用例规约一般包括简要说明、事件流（基本流和备用流）、用例场景、特殊需求、前置条件、后置条件几个主要部分。

本文选取公司数量分析功能来作详细分析，如表3-2 所示。

|  |  |
| --- | --- |
| 用例名称 | 公司投诉数量分析 |
| 简要说明 | 对一段时间内公司的投诉数量统计，输出较高的几位 |
| 执行者 | 消费者或工商管理者 |
| 前置条件 | 1. 消费者或工商部门管理人员已经登录系统 2. 系统完成统计分析操作 |
| 后置条件 | 输出报表 |
| 基本流 | 1. 消费者或工商部门管理人员登录系统网站 2. 登陆成功选择统计分析功能 3. 选择公司数量分析 4. 选择开始计算日期和结束计算日期 5. 计算机执行输出后置条件 |
| 备用流 | 1. 无账号选择注册 2. 输入信息完成注册进入2 |
| 异常 | 1. 登录系统失败重回1，并提示登录失败 2. 开始日期不能晚于结束日期，且结束日期需要在当前日期前一天，否则返回4 |
| 业务实体 | 用户 |

经分析得到本系统的基础功能列表，如表3-3 所示。

**1.3** **非功能性需求**

非功能需求定义了软件产品为满足用户业务需求必须具备的功能需求之外的特性。软件产品的非功能性需求主要涵盖系统的整体性能（数据管理、软件发布管理等）、性能、可靠性、可维护性、可扩充性以及对技术和业务的适应性等。 下面介绍本系统的非功能性需求。

1. 可靠性需求

该系统旨向服务广大消费者，这需要系统有一定的可靠性，如果长时间无法运作，那么消费者的诉求就不能快捷及时的反馈。并且如果系统出现问题，也应能在短时间内恢复正常。

1. 用户界面友好型需求

该系统由于面向各种各样的消费者，消费者文化水平不平均，系统就需要有明确的指示、友好的界面来引导消费者快速的找到投诉的入口，掌握系统的运用。

1. 安全性需求

该系统首先需要对不同级别的工商局做出限制，低级别的工商端不能访问限制之外的数据，另外用户也不能随意查看、修改其他人的投诉信息。用户的账号密码等信息也需要保密存储，不能泄露。系统也要能应对例如SQL注入等攻击，保证数据库及网站的完整性和安全性。

1. 性能需求

该系统需要响应时间在几秒之内，让用户有更加优质的体验，快捷的看到自己所需要的统计分析结果。系统需要优化架构、优化数据存储等解决相应性能问题达到要求。

1.4小结

本章根据软件需求层次，首先对系统的业务需求进行分析，通过分析明确了系统需要到达的目标；并对用户需求进行分析，明确了系统的主要业务流程和参与用户，规范了不同用户需要完成的任务；再次是对功能性需求进行分析，明确了用例模型和用例规约，总结了系统所具备的功能需求；最后是对非功能性需求进行分析，明确了系统在可靠性、用户界面友好性、安全性与性能方面的需求。

**第四章 系统总体设计**

在需求分析之后，系统明确了所要实现的功能和对应的用户角色，接下来就要进行总体设计目标是将系统合理的分层，划分具体的模块，实现软件工程所要求的高内聚低耦合的开发目标，保证开发系统的可维护性和高扩展性。

**1.1 系统体系结构设计**

系统总体分为几层，分别是原始数据层，数据处理层，和应用层。不同的层负责不同的功能

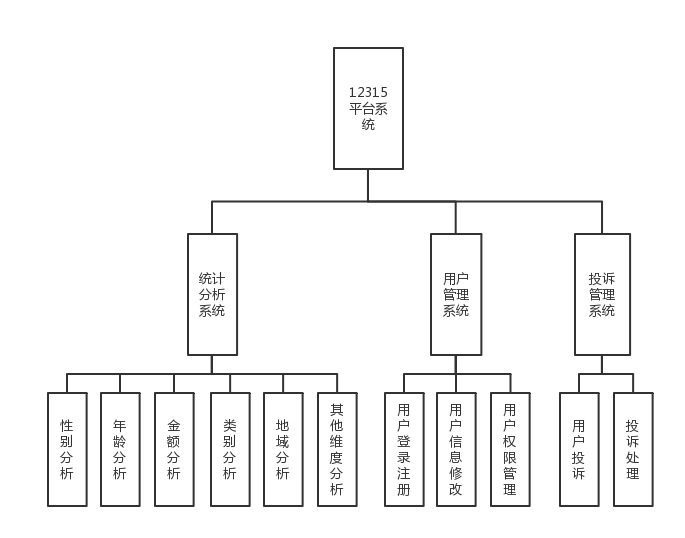
1. 原始数据层。在前台应用用户提交的投诉数据都存储在mysql中，在后台通过Sqoop传入Hive中，将投诉数据、用户数据、公司数据存放在HDFS上，方便后期进行处理和计算。
2. 数据处理层。这一层主要是利用Spark和SparkSQL将Hive中的数据进行统计计算，并将计算完的结果放入Mysql中方便用户读取
3. 应用层。这一层则是使用SSM这类JAVAEE的技术做展现，根据用户所选时间区间或者其他参数从Mysql中读取对应的数据并进行展现

在应用层中，我主要使用SSM作为系统框架，并将系统分为以下几个部分：DAO层，Service层，Conroller层，Web层。不同的层同样负责不同的功能：

1. DAO层：利用Mybatis从数据库读取所需要的图表数据、投诉数据和进行数据库操作
2. Service层：对某种对象使用DAO层进行业务功能的实现，实现具体的插入删除统计前的逻辑。
3. Controller层：对网络请求进行匹配并交由对应的Service进行处理
4. Web层：负责网络页面的展现。

**1.2 系统功能结构设计**

系统功能根据需求划分为以下几个主要功能



**1.2.1 统计分析模块**

统计分析模块是本系统的主要功能，针对投诉信息在多方面多维度进行分析。用户可以选择分析的时间区间，对这段时间内的总量进行分析也可以根据单一项目按时间进行分析。所有分析都会以图表的形式进行展现。

性别分析是统计一段时间内（用户选择）投诉用户的性别和对应的投诉数量。

年龄分析是统计一段时间内用户不同年龄段的投诉数量，让用户能够针对性分析在不同年龄区间下投诉的状况

金额分析是统计一段时间内用户不同金额段的投诉数量

类别分析是统计一段时间内用户不同类别的投诉数量，后台会统计这段时间内该类别下（用户可选择一级大类、二级大类、三级大类）的不同类别商品的投诉总数量，并返回Top10供用户查看

**1.2.2 用户管理模块**

用户管理模块主要是负责用户登录注册，并且由于用户角色的不同会对应到不同的网页。

**1.3**  **数据库设计**

**1.2.1 12315系统ER关系图**

E-R图也称实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。ER图明确了数据库的结构

**1.2.1 数据库详细设计**

普通用户表（user）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 备注 |
| UID | Varchar(30) | 主键 | 用户 |
| CARD\_TYPE | Int |  | 证件类型 |
| CARD\_NUM | Varchar(30) |  | 证件号码 |
| SEX | Int |  | 性别 |
| BIRTHDAY | Date |  | 生日 |
| NAME | Varchar(20) |  | 姓名 |
| USERNAME | Varchar(20) |  | 用户名 |
| PASSWORD | Varchar(20) |  | 密码 |
| PHONE | Varchar(30) |  | 电话 |
| ADDR | Varchar(100) |  | 地址 |

投诉表（complaint）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 备注 |
| CID | Varchar(50) | 主键 | 投诉ID |
| UID | Varchar(30) | 外键 | 用户ID |
| EID | Varchar(50) | 外键 | 企业ID |
| MERCHANDISE\_NAME | Varchar(20) |  | 商品服务名称 |
| SUBMIT\_TIME | Timestamp |  | 提交时间 |
| HAPPEN\_TIME | Timestamp |  | 问题发生时间 |
| MERCHANDISE\_TYPE | Varchar(20) |  | 商品服务类型 |
| PURCHASE\_WAY | Int |  | 购买方式 |
| PROBLEM\_TYPE | Int |  | 问题类别 |
| COMPLAINT\_CONTENT | Varchar(500) |  | 投诉内容 |
| APPEAL\_TYPE | Varchar(30) |  | 诉求类型 |
| COUNT | Varchar(50) |  | 数量 |
| AMOUNT | Bigint |  | 金额 |
| STATUS | Int |  | 状态 |
| IID | Varchar(50) | 外键 | 工商ID |

工商局表（IAC）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 备注 |
| IID | Varchar(50) | 主键 | 工商局ID |
| NAME | Varchar(50) |  | 工商局名称 |
| ADDR | Varchar(50) |  | 地址 |
| CONTACT | Varchar(20) |  | 联系方式 |
| LEVEL | Int |  | 级别（1:总局，2:省级，3：市级） |
| USERNAME | Varchar(20) |  | 登录用户名 |
| PASSWORD | Varchar(30) |  | 密码 |

企业表（enterprise）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 备注 |
| EID | Varchar(50) | 主键 | 企业代码 |
| IID | Varchar(50) | 外键 | 工商ID |
| NAME | Varchar(50) |  | 公司名 |
| INDUSTRY\_TYPE | Varchar(10) |  | 行业类别 |
| ADDR | Varchar(100) |  | 地址 |
| SCOPE | Varchar(500) |  | 范围 |

各类维度表

以性别统计为例

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 备注 |
| SEX | Int |  | 性别 |
| COUNT | Bigint |  | 数量 |
| TIME | Date |  | 日期区间 |

**1.2.2 小结**

本章在对12315系统进行需求分析的基础上，对系统的总体设计进行了介绍，根据设计原则选择了合适的体系结构和系统框架，并对框架的工作流程做了简单的介绍并且明确了模块的功能和划分，最后通过ER图确定了实体关系，并以此设计了数据库表结构

**第五章 系统详细设计与实现**

通过总体设计，本系统基本明确了所需要实现的功能和相关的结构，选择了以Spark+Hive的数据处理方式和SSM的展现层技术。本章主要介绍系统环境的配置和框架的搭建过程，以及模块的详细流程和具体实现代码。

**5.1 系统框架的实现**

本系统采用Spark+Hadoop+Hive的分布式框架，首先介绍相关环境信息，主机（虚拟机）信息如下：

|  |  |  |  |
| --- | --- | --- | --- |
| IP | 主机名 | 内存大小 | 角色 |
| 192.168. | Master | 4G | Master Namenode ScedonaryNamenode |
|  | Slave1 | 2G | Slaves Datanode |
|  | Slave2 | 2G | Slaves Datanoed |

Hadoop采用2.8.3版本 ，Hive为2.3.2，Spark为2.2.1，JDK8，Scala 11.3和MySQL5.7 ，系统则是CentOS7并已经完成系统的网络设置。

开发平台为Eclipse origin作为JavaEE的开发和IDEA则是作为Scala的开发环境，操作系统为windows7

**5.1.1 Hadoop/Spark框架的配置**

5.1.1.1 Hadoop配置：

Hadoop配置需要首先配置免密登录和相关环境变量，三台机子的SSH免密登录需要在每台机器下运行ssh-keygen –t rsa命令创建秘钥，然后将每台机器下的~/.ssh下的id\_rsa.pub，也就是公钥部分放在一个authorized\_keys，这样三台机器就可以直接通过ssh连接，使得hadoop的启动脚本等可以顺畅的运行。在配置完免密登陆后，解压相应软件的安装包后，需要更改对应的环境变量文件/etc/profile，以下为环境变量的设置

export JAVA\_HOME=/usr/java/

export PATH=$JAVA\_HOME/bin:$PATH

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export SCALA\_HOME=/usr/scala

export HIVE\_HOME=/home/hadoop/hive-2.3.2

export SPARK\_HOME=/home/hadoop/spark-2.2.1

export HADOOP\_HOME=/home/hadoop/hadoop-2.8.3

export PATH=$SPARK\_HOME/bin:$SCALA\_HOME/bin:$HADOOP\_HOME/bin:$HIVE\_HOME/bin:$PATH

export PYTHON\_HOME=/usr/python/

export PATH=$PYTHON\_HOME/bin:$PATH

export MAVEN\_HOME=/usr/maven

export PATH=$MAVEN\_HOME/bin:$PATH

export ACCUMULO\_HOME=/var/lib/accumulo

完成环境变量设置后便是对hadoop配置环境文件的设置，在新版本的Hadoop2.7+，配置文件在$HADOOP\_HOME/etc/hadoop下。

1. core-site.xml负责配置Hadoop的核心配置包括hdfs路径等

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hadoop/hadoop-2.8.3/tmp</value>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://hdf1:9000</value>

</property>

</configuration>

1. hdfs-site.xml配置hdfs的详细信息，包括hdfs的副本数量、data文件和name文件夹、网页管理UI地址等

<configuration>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>/home/hadoop/hadoop-2.8.3/hdfs/name</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/home/hadoop/hadoop-2.8.3/hdfs/data</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>hdf1:9001</value>

</property>

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

</configuration>

1. mapred-site.xml

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

1. yarn-site.xml

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.resourcemanager.address</name>

<value>hdf1:18040</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>hdf1:18030</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>hdf1:18088</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>hdf1:18025</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>hdf1:18141</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>hdf1:8088</value>

</property>

</configuration>

1. 最后在slaves中加入从机的主机名，在hadop-env.sh加上Java的环境变量就可以进入$HADOOP\_HOME/sbin用start-all.sh启动hadoop，成功启动后可以在50070端口看到hdfs的信息

5.1.1.2 Spark配置：

进入conf文件夹编写Spark-env.sh，加上java和scala的环境变量，做以下配置：

export SPARK\_WORK\_MEMORY=1g

export SPARK\_MASTER\_IP=hdf1

export MASTER=spark://hdf1:7077

export HADOOP\_CONF\_DIR=/home/hadoop/hadoop-2.8.3/etc/hadoop

指定工作内存大小、master地址、hadoop的配置文件地址，同样你可以在该文件中编辑spark中worker、executor所用的内存大小和cpu核心数。此后在slaves中加入从机主机名就可以在bin目录下以standalone模式运行，或是在hadoop启动后以yarn模式让spark在集群上运行

5.1.1.2 Hive配置：

进入conf文件夹，编辑hive-site.xml,修改对应的链接地址、用户名、密码，并且根据所使用的metastore选择对应的驱动包

configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://192.168.63.151:3306/hive?createDatabaseIfNotExist=true&amp;characterEncoding=UTF-8&amp;useSSL=false</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>009649</value>

</property>

<property>

<name>hive.exec.scratchdir</name>

<value>/home/hadoop/hive-2.3.2/tmp</value>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/home/hadoop/hive-2.3.2/warehouse</value>

</property>

<property>

<name>hive.querylog.location</name>

<value>/home/hadoop/hive-2.3.2/log</value>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

</configuration>

并在hive-env.sh配置好java、hadoop环境变量和hive的jar包地址，并在lib文件夹下放入所连接的metastore的驱动包，运行 schematool –initSchema –dbType mysql初始化metastore数据。

**5.1.2 SSM框架实现**

利用Maven构建工程，设置好pom.xml后配置相应的Spring配置文件、Mybatis配置文件、SpringMVC配置文件和Web.xml，SSM架构中具体分为DAO层，通过Mybatis用mapper.xm来和数据库交互，Service层则利用DAO层完成具体业务，Controller则对不同路径地址产生交互。系统到这一步完成了环境配置和基本展现层框架的搭建。

**5.2**  **统计分析功能实现**

**5.2.1 数据库准备和数据装载**

本演示系统使用数据主要是由Scala脚本生成的模拟数据和现有数据组成。系统利用SparkSQL对数据做处理，由于这里SparkSQL是基于Hive的，所以要现在Hive中创建数据库或者利用Sqoop 运行如下命令

sqoop import --connect jdbc:mysql://localhost:3306/consume --table complaint -m 1 --hive-import --username root –P

当手动加载数据时，需要先在Hive创建表，进入hive后

CREATE TABLE IF NOT EXISTS default.user(

uid long ,

card\_type int ,

car\_number string ,

name string ,

username string ,

password string ,

birthday timestamp ,

sex string,

phone long,

addr string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' ;

准备好数据文件后利用load data加载文件

load data local inpath '/user.txt' into table default.user ;

再select后就能看到插入的数据。本系统第一次的数据为手动加载，后续Mysql更新数据会通过Sqoop的append模式添加到Hive中。

**5.2.2 数据处理**

数据处理部分利用Spark下的SparkSQL实现，SparkSQL在统计Hive中的数据，对于基本的计数、求和等非常适合，基本可以让开发人员无缝从传统的SQL开发过渡到大数据的分析开发。SparkSQL主要基于的对象类型是DataFrame，这种类型可以理解为SparkCore的主要类型RDD加上了一个Schema，让一个单纯地集合变成了和表类似的东西。DataFrame可以注册临时表，并在此基础上运行SQL语句。同时SparkSQL也可以直接在Hive上以类似HQL的方式运行。本系统主要的数据分析便是直接使用SparkSQL直接在Hive上统计，并直接写入Mysql。

val spark = SparkSession  
 .builder()  
 .enableHiveSupport()  
 .appName("Spark SQL")  
 .getOrCreate()  
  
val prop = new Properties()  
prop.put("user", "root") //表示用户名是root  
prop.put("password", "password") //表示密码  
prop.put("driver","com.mysql.jdbc.Driver") //表示驱动程序是com.mysql.jdbc.Driver  
  
  
spark.sql("set spark.sql.shuffle.partitions=20")  
//企业每日投诉  
val enterprise\_daily\_DF = spark.sql("SELECT eid, submit\_time, count(1) as count FROM " +  
 "default.complaint group by eid, submit\_time")  
enterprise\_daily\_DF.show()  
enterprise\_daily\_DF.write.mode("append").jdbc("jdbc:mysql://192.168.63.151:3306/consume",  
 "consume.enterprise\_daily\_result", prop)

在SparkSession加上enableHiveSupport()使Spark支持Hive，然后初始化需要存入数据库的相关信息，最后使用Spark.sql()就可以使用SQL语句进行查询，最后用DataFrame写出JDBC的方法直接导出为表，方便展现层快速调用。

**5.2.3数据展示**

SSM体系下，Mybatis利用映射器来获取数据库数据，随后结合SpringMVC和Spring提供的功能，定义Service层和Controller层，由Controller层处理请求，Service层处理业务。这里以统计一段时间内的各公司投诉数量为例，介绍流程。

在映射器中需要配置SQL语句和配置参数。Mybatis的映射器由接口和一个XML文件组成。这一部分也就对应的是DAO层，具体代码如下：

<resultMap id="CountEnterpriseResultMap" type="com.latoris.consume.beans.VOBean">

<result column="name" property="name" jdbcType="VARCHAR" />

<result column="count" property="count" jdbcType="INTEGER" />

</resultMap>

<select id="CountByEnterprise" resultType="com.latoris.consume.beans.VOBean">

select enterprise.name, a.count from (select eid, sum(count) count from enterprise\_daily\_result

where submit\_time between #{para.startTime} and #{para.endTime}

GROUP BY eid

order by count desc)a left join enterprise on a.eid=enterprise.eid

</select>

这里定义了一个resultMap，属性id是标识，而type则是需要映射的对象类（这里就是com.latoris.consume.beans.VOBEAN,也就是我定义的视图Pojo类），然后对应需要映射的变量，都有property属性代表映射类的对应变量名，column属性代表数据库中列名。这样在select中说明对应的resultMap属性，Mybatis就可以将数据自动包装成对应的类。映射器的接口文件代码如下：

public interface CountMapper {

public List<VOBean> CountByEnterprise(@Param(value="para") VOBean para);

}

并且在xml文件中需要对应namespace保证接口和xml文件能顺利使用。

然后我们在Service中定义相关业务的代码，为了保证代码可控和避免过多信息暴露我们使用接口和实现类的方法，接口定义代码如下：

public interface VOService{

public List<VOBean> CountByEnterprise(VOBean parameter);

}

实现类代码如下：

@Service

public class VOServiceImpl implements VOService{

@Autowired

private CountMapper mapper;

@Override

public List<VOBean> CountByEnterprise(VOBean para) {

// TODO Auto-generated method stub

return mapper.CountByEnterprise(para);

}

}

这里利用SpringMVC的@Service注解标识这是一个Service层类，让SpringMVC能更方便的进行调用，而变量mapper，也就是DAO层的类成员使用了@Autowired注释，这个注释意味着SpringIOC在完成定位Bean之后，会对这个变量按照类型注入，完成变量的自动装配。在Service中由于这个实例不需要多余的业务流程，所以直接调用DAO层的方法。再往上一级便是Controller层，也就是展现层的部分。代码部分如下：

@RequestMapping("/count\_enterprise")

public void countEnterprise(HttpServletRequest request, HttpServletResponse response, @RequestBody VOBean para) {

System.out.println("count");

System.out.println(para.getStartTime()+" "+para.getEndTime());

List<VOBean> list = VOService.CountByEnterprise(para);

response.setContentType("text/html; charset=utf-8");

JSONArray json = new JSONArray();

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

JSONObject jo =new JSONObject();

jo.put("eid", list.get(i).getName());

jo.put("count", list.get(i).getCount());

json.put(jo);

}

System.out.println(json.toString());

// 返回给前端页面

PrintWriter out;

try {

out = response.getWriter();

out.println(json);

out.flush();

out.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

这里根据网页前端采用ECharts需要的利用Json字符串来初始化图表，所以在Controller中要完成获取参数然后根据参数从数据库取值，并将值包装成Json然后发送给前台的功能。这里的@RequestMapping 表示所需要匹配的路径，@RequestBody注释，标识SpringMVC会将接收到Json字符串自动转换为对应的实体类，这里@RequestBody VOBean para，就会将传递过来的Json参数包装成VOBean对象。在调用Service完成业务得到List后，Controller将对应的List转化成Json字符串然后利用PrintWriter让前台接收到。最后，网页上的代码如下，在Head部分引入layui，jQuery，ECharts相关的文件

<!-- JQUERY -->

<script src="${pageContext.request.contextPath}/resources/js/jquery/jquery-2.0.3.min.js"></script>

<script type="text/javascript" src="${pageContext.request.contextPath}/resources/js/echarts.min.js">

<script type="text/javascript" src="${pageContext.request.contextPath}/resources/js/require.js">

<script src="${pageContext.request.contextPath}/resources/layui/layui.js"></script>

<link rel="stylesheet" href="${pageContext.request.contextPath}/resources/layui/css/layui.css">

</script>

然后在body部分主要为以下代码，利用layui绘制两个日期控件并注册，如下所示：

<div class="layui-inline">

<label class="layui-form-label">开始日期 </label>

<div class="layui-input-inline">

<input type="text" class="layui-input" id="test1"/>

</div>

</div>

利用以下代码来注册日期控件

layui.use('laydate', function(){

var laydate = layui.laydate;

//执行一个laydate实例

laydate.render({

elem: '#test1' //指定元素

});

});

然后为Echarts图表流出预留空间，并设置option值，设定X轴，y轴，标题等信息，在Ajax中异步加载

// 基于准备好的dom，初始化echarts图表

myChart = echarts.init(document.getElementById('main'));

var option = {

title:{

text:'投诉数量文件分析'

},

tooltip:{

//show:true

},

legend:{

data:['数量']

},

xAxis:[

{

type:'category',

nameTextStyle:{

width:"70%"

}

}

],

yAxis:{},

series:[

{

"name":"投诉量",

"type":"bar",

}

]

};

var name = "a"

var count = "0"

var startTime = $("#test1").val();

var endTime=$("#test2").val();;

$.ajax({

type: "post",

async: false,//同步执行

url:"count\_enterprise",

data:JSON.stringify({

name:name,

startTime:startTime,

endTime:endTime,

count:count}),

contentType:"application/json;charset=utf-8",

dataType:"json",//返回数据形式为json

success:function(result){

if(result){

//初始化option.xAxis[0]中的data

option.xAxis[0].data=[];

for(vari=0;i<result.length;i++){

option.xAxis[0].data.push(result[i].eid);

}

//初始化option.series[0]中的data

option.series[0].data=[];

for(vari=0;i<result.length;i++){

option.series[0].data.push(result[i].count);

}

}

}

});

//为echarts对象加载数据

myChart.setOption(option);

这样就是本系统的统计分析功能的主要流程，针对不同分析，会有不同的具体的参数列表等，但主题流程基本都是由页面创建对象Json字符串，和其他变量值，传至Controller处理，经由Service调用DAO完成对数据库的查询。

5.3用户登录注册模块设计与实现

**第六章 结语**

**6.1论文总结**

本项目是源自企业和工商局的项目，通过参与项目，设计和实现了基于Hadoop和Spark的数据分析系统，该系统是增对在项目遇到的需求自己独立开发的，由于原项目对数据量有相关的需求，所以采用了大数据的技术而不是传统的Oracle数据库。这次本人开发的系统主要用到的大数据技术是基于Hadoop和Spark环境的SparkSQL和Hive，在大数据技术上门槛相对传统MR等更低。而展现层则选择了在企业中比较流行的SSM架构，SSM架构用较为复杂的结构的代价提高了系统的解耦度。在本次开发中，一方面在实际项目中深入了解一个软件项目的开发过程，另一方面通过自学多种技术，加大了自己的技术栈，并对SSM系统较平时教学中有了更多的理解。

本文主要研究的内容有以下：

第一部分是项目的相关背景和技术，12315系统的背景，为何引入大数据技术。还有大数据技术有哪些，Hadoop、Spark等技术背后的实现原理和基本功能。

第二部分是项目的需求分析。主要是12315系统的业务需求、用户需求、非功能性需求，描述了不同角色的相应用例

第三部分是项目的总体设计，根据需求分析，做出功能模块的划分，明确所选用的技术栈，明确数据库、数据字典的设计。为详细设计奠定基础

第四部分是项目的详细设计，也就是具体的开发规程，将实际系统根据设计编写出来。

此系统同时尚有许多缺陷和问题，首先是系统的美观程度不够，页面分离。然后本系统由于主要是一个演示系统，所以公司数据和工商等数据都是通过模拟生成或先前准备，没有足够的现实意义。其次当前的系统数据是等于存储了两份，一份是Mysql中用来查询，一份在HDFS中，让Spark基于Hive做统计分析，在数据大小更大的情况下，双重副本的模式并不理想。最后一个问题就是当前所实现统计分析主要是以计数为主，并没有引入机器学习、数据挖掘相关内容，数据价值没有完全挖掘出来。

**6.2**  **工作展望**

该系统能实现一个数据分析系统的初步建立，帮助消费者和工商局管理人员从投诉信息中获取更多的价值，随着投诉系统的完善和相关数据的引入，系统可以更加深入的挖掘和预测数据。

如果要增大系统的能力，让投诉数据不再只是数据而能帮助工商管理人员、企业家、消费者做出决策，体现出作为信息时代下的社会的优点，就需要改进该系统，主要需要在以下方面改进：

1. 增加数据源，单一的投诉数据所能发现的信息过于少，当前信息爆炸的时代，引入社交数据、行为数据能很大程度上提升数据的价值。
2. 为系统引入机器学习和数据挖掘的模块，运用机器学习相关理论可以将分析深入化
3. 优化展现层架构