专 业 学 位 硕 士 学 位 论 文

基于微服务的综合数据管理框架的设计与实现

Design and implementation of a comprehensive management framework for data based on microservices

作 者 姓 名： 方圆

学科、 专业： 软件工程

学 号： 31617003

指 导 教 师： 赖晓晨

完 成 日 期：

大连理工大学

Dalian University of Technology

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：

作 者 签 名 ： 日期： 年 月 日

摘 要

近年来随着企业业务的发展，企业数据膨胀迅速，管理、运营人员对企业数据的使用需求更加复杂。面对业务部门数十个数据库表、以T为单位的数据量，如何在秒级别的时间算出各种时间、空间聚合维度下的数十种指标的离线、实时的数据，是综合数据管理系统面对的难题。除了数据的查询、计算，管理、运营人员还希望对数据进行进一步的挖掘，以帮助业务发展的平稳推进。

本文设计了一套基于微服务的离线与实时数据数据的综合管理框架以解决上述问题。框架使用基于Spring boot和Protobuf的微服务架构，实现前后端、后端服务间的数据交互，管理模块架构和数据通信。框架还结合了CAS框架和数据系统的接口特性，实现了数据系统的服务间认证和日志管理架构，提高了安全性和可用性。框架中离线数据管理服务通过每日的定时任务控制Sqoop、Hive、Hbase、HDFS等工具对离线数据进行预处理，利用Kylin对预处理后的数据进行预计算和预聚合，以应对复杂的、高并发的离线数据数据需求。在线数据管理服务使用基于定时器的Mysql任务管理，通过每分钟的预计算，预处理数据，实现ms级别的数据查询响应。框架还使用了机器学习对数据进行分析，利用Holt-winters算法，结合数据实际，实现了基于本框架实时数据的时间序列预测算法。

本框架开发完成后，已经应用到了企业的数据管理系统中，解决了数据系统的使用瓶颈问题，为企业的运营和发展提供了可靠助力。

关键词：大数据；架构；预计算

Design and implementation of a comprehensive management framework for data based on microservices

Abstract

With the development of enterprise business in recent years, the data of enterprise has expanded rapidly. The demand of the enterprise data has become increasingly complex to the manager and the operational staff. Faced with dozens of database tables and T-unit-based data quantity in business department, how to calculate dozens of indices of offline and online database under the time and space polymerization dimensions in the second-level is the challenge of Integrated Data Management System. Except for the query and calculation of data, the manager and the operational staff also expect to make a further data mining, aiming to support the smooth running of the business.This thesis designed a set of integrated management framework for solving the above problems based on the offline and online data of micro services. The framework employs the micro service framework based on Spring boot and Protobuf to implement the data interaction from the front to the back-end and manage the module framework and data communication. Also, this framework integrates the interface characteristics of the CAS framework and data systems, which accomplishes the inter-service authentication and log management architecture of the data system and improves the security and availability.In the framework, the offline data management service preprocesses offline data by using daily timed task to control tools such as Sqoop, Hive, Hbase, HDFS, etc. Meanwhile, it uses Kylin to pre-calculate and pre-aggregate the preprocessed data to deal with the demand of complex and high concurrency offline data. The online data management service applies timer-based Mysql task management to preprocess the data and perform ms-level data query responses.

Since this framework was exploited, it has been applied to the enterprise's data management system and solved the bottleneck problem of the use of the data system, providing a reliable assistance for the operation and development of the enterprise.

Key Words：Big Data; Architecture; Precalculation

目 录

[摘 要 I](#_Toc508968295)

[Abstract II](#_Toc508968296)

[1 绪论 1](#_Toc508968297)

[1.1 研究背景与意义 1](#_Toc508968298)

[1.2 国内外研究现状与发展趋势 1](#_Toc508968299)

[1.3 论文主要工作 2](#_Toc508968300)

[1.4 论文组织结构 2](#_Toc508968301)

[2 相关算法和技术 4](#_Toc508968302)

[2.1 基于Sping boot框架和Protobuf协议的微服务架构 4](#_Toc508968303)

[2.2 基于Hadoop的数据处理和存储 5](#_Toc508968304)

[2.3 基于Holt-Winters的预测器和监控器 7](#_Toc508968305)

[2.4 基于Fast Cubing算法的数据预计算 8](#_Toc508968306)

[2.5 基于CAS的SSO过程 9](#_Toc508968307)

[3 系统需求分析 11](#_Toc508968308)

[3.1 总体需求分析 11](#_Toc508968309)

[3.2 功能性需求 11](#_Toc508968310)

[3.2.1 离线数据管理模块 11](#_Toc508968311)

[3.2.2 实时数据管理模块 13](#_Toc508968312)

[3.2.3 KPI管理模块 13](#_Toc508968313)

[3.2.4 认证和日志管理模块 15](#_Toc508968314)

[3.3 非功能性需求 15](#_Toc508968315)

[4 系统设计 17](#_Toc508968316)

[4.1 总体设计 17](#_Toc508968317)

[4.1.1 微服务架构设计 17](#_Toc508968318)

[4.1.2 离线数据处理流程设计 18](#_Toc508968319)

[4.1.3 实时数据处理流程设计 20](#_Toc508968320)

[4.2 数据管理设计 22](#_Toc508968321)

[4.2.1 Hive表设计 22](#_Toc508968322)

[4.2.2 Kylin的Cube设计 24](#_Toc508968323)

[4.2.3 MySQL表设计 25](#_Toc508968324)

[4.3 系统功能模块设计 26](#_Toc508968325)

[4.3.1 离线数据管理模块 26](#_Toc508968326)

[4.3.2 实时数据管理模块 28](#_Toc508968327)

[4.3.3 KPI管理模块 31](#_Toc508968328)

[4.3.4 认证和日志管理模块 34](#_Toc508968329)

[5 系统实现 36](#_Toc508968330)

[5.1 离线数据管理模块 36](#_Toc508968331)

[5.1.1 业务概览子模块 36](#_Toc508968332)

[5.1.2 经营主题子模块 39](#_Toc508968333)

[5.1.3 客户主题子模块 39](#_Toc508968334)

[5.2 实时数据管理模块 41](#_Toc508968335)

[5.2.1 监控概览子模块 41](#_Toc508968336)

[5.2.2 实时趋势图子模块 43](#_Toc508968337)

[5.2.3 预测和报警器子模块 44](#_Toc508968338)

[5.3 KPI管理模块 44](#_Toc508968339)

[5.2.1 KPI展示 44](#_Toc508968340)

[5.2.2 KPI批量上传 45](#_Toc508968341)

[5.2.3 KPI修改 46](#_Toc508968342)

[5.2.4 KPI下载 46](#_Toc508968343)

[5.4 认证和日志管理模块 47](#_Toc508968344)

[5.2.1 用户单点登录认证 47](#_Toc508968345)

[5.2.2 KPI日志 47](#_Toc508968346)

[5.2.2 数据日志 47](#_Toc508968347)

[6 系统测试 48](#_Toc508968348)

[6.1 测试理论 48](#_Toc508968349)

[6.2 系统典型模块功能测试 49](#_Toc508968350)

[6.2.1 离线数据管理模块 49](#_Toc508968351)

[6.2.2 实时数据管理模块 49](#_Toc508968352)

[6.2.3 KPI管理模块 49](#_Toc508968353)

[6.2.4 认证和日志管理模块 49](#_Toc508968354)

[6.3 性能测试 49](#_Toc508968355)

[结 论 51](#_Toc508968356)

[参 考 文 献 52](#_Toc508968357)

[致 谢 54](#_Toc508968358)

[大连理工大学学位论文版权使用授权书 55](#_Toc508968359)

1 绪论

## 1.1 研究背景与意义

随着企业的快速发展，企业业务日趋复杂，管理、运营人员对于企业数据的分析、使用日趋困难，如何帮助企业从繁杂、基础的业务数据中提取简单、有效、直观的数据成为企业运营的一大难题。由于业务项目日趋复杂，数据繁杂的问题，企业的数据量也在迅速膨胀，每一秒钟都有成百上千的数据进入数据库[1]。面对海量数据，企业运营及管理人员的日常运营工作对数据的依赖日趋明显，使用需求也日趋复杂，不论是对以分钟为单位，以空间区域为聚合维度的实时数据查询，还是以天、月甚至年为单位，以城市、区域、国家为聚合维度的离线数据查询，都给数据的的存储、分析和管理带来了极大的挑战。

面对如此复杂、高并发的数据查询和分析需求，数据系统需要进行大量计算和处理，使用传统数据系统的响应时间通常高达数十秒甚至上百秒。因此急需开发出一套快速响应、稳定性强、计算能力强的数据管理系统[2-3]。

于此同时，面对庞大的数据集，如何进一步分析、利用数据，进而促进业务的发展和运营，也是传统数据系统需要解决的难题。数据量越大，数据内部隐藏的规律性和实用性往往也越强，如果能够对数据进行合理的分析和处理，将给企业的业务发展带来强大的助力。

正是基于这样的背景，本论文开发出来一套能够能够实现快速查询、分析、处理大数据的实时与离线数据综合管理框架。框架通过任务调度结合大量的数据预抽取、预计算和预聚合工作，处理数据，极大的提升了响应速度和并发性能，解决数据管理的困难。同时框架通过机器学习算法，结合数据自身特点，进一步分析、挖掘数据，进一步提升数据的利用率，促进业务的发展。

## 1.2 国内外研究现状与发展趋势

大数据是高容量、高生成速率、种类繁多的信息价值，同时需要新的处理形式去确保判断的作出、洞察力的发现和处理的优化[4-8]。

## 1.3 论文主要工作

本论文的主要研究目标是基于微服务建立一个数据综合管理框架，该框架能够在大量数据的基础上，实现对离线数据和实时数据的处理、分析和快速查询。本文的主要工作在于：

（1） 通过对微服务框架和JSON、Thrift、protobuf等服务间数据传输方式进行研究，整合了Spring boot框架和Protobuf协议，构造出一套便于开发、部署、管理的微服务架构；

（2） 对Hive、Sqoop、HDFS等技术进行研究，实现从多个业务、数十个库表的数据到框架预备数据的ETL过程，再利用Kylin实现对数据的预聚合，通过查询Kylin，实现对100多项指标、多种聚合维度的海量数据的快速查询；

（3） 通过定时器实现对多个Mysql数据表的预处理和预计算，将处理过的数据导入新的Mysql，实现对实时数据的快速查询；

（4） 研究了移动平均分解法，Holt-Winters算法，局部加权回归法，自回归整合移动平均算法等时间序列预测算法，通过改进Holt-Winter算法，实现对数据的预测和监控功能；

（5） 研究CAS框架，实现单点登陆，统一认证和管理各服务。利用Cookie信息增加过滤器，实现对于数据接口的特殊过滤。

## 1.4 论文组织结构

本论文的组织结构如下：

第一章绪论，本章节主要提出了综合数据管理系统在当前企业发展背景下的意义，并对国内外研究发展趋势和本论文的主要工作进行了简单介绍。

第二章相关算法和技术介绍，本章节主要介绍了综合数据管理框架开发中所使用的各种算法和技术，主要包括：Spring-boot框架、Protobuf协议、Hadoop工具、Holt-Winters算法、Fast Cubing算法和CAS框架等内容。

第三章系统需求分析，本章节主要介绍了综合数据管理框架的总体需求概况，并对框架功能模块需求和非功能性需求进行了具体分析。

第四章系统设计，本章节主要介绍了综合数据管理框架的总体设计和数据处理流程中涉及的各类数据库表设计，并对框架各功能模块的设计情况进行了具体分析。

第五章系统实现，本章节介绍了综合数据管理框架各功能模块的具体实现方式及技术处理细节，包括离线数据管理模块、实时数据管理模块、KPI管理模块和用户日志管理模块。

第六章系统测试，本章节主要介绍了测试部分相关信息，主要包括测试理论，各模块典型功能测试和性能测试等内容。

2 相关算法和技术

## 2.1 基于Sping boot框架和Protobuf协议的微服务架构

Sping boot框架作为一种轻量级的Java开发框架，极大的简化了Java服务的架构和开发过程[9]。Spring boot作为服务端的开发框架，具有自动配置Spring的特点，极大的简化了配置过程，为微服务架构提供了可靠的支撑作用[10-11]。此外，Spring boot框架还具有以下特点：

（1） 集成Jackson框架

Jackson是一套JSON API，可以实现Java对象和JSON数据的快速转换。在当前主流的Java Web服务中，前后端通信采用的数据格式多是JSON数据，因此集成了Jackson的Spring boot框架，极大简化了前后端数据的交互过程。同时Jackson框架相较于Json-lib、Gson等JSON API，还具有性能高、稳定性强、序列化效果好的优点，可以进一步提高系统稳定性，提升响应服务速度。

（2） 集成Tomcat

由于Spring boot集成了Tomcat服务器，所以在开发时，方便快速调试，在部署时，也可以简化部署过程。因此使用Spring boot框架可以进一步简化开发工作，提升开发效率[12]。

（3） 提供配置检查功能

Spring boot提供了大量的默认配置，简化配置工作，而少量需要手动配置的信息，只需在applicat.properties或application.yml文件进行简单配置，即可通过注解方式读取配置信息。同时在启动过程中，Spring boot还会对数据库连接、Apache Kylin连接等状态进行检测，自动检测生产环境状况，提升服务可靠性。

微服务架构的多个后端服务间也经常有数据交互，如果按照前后端数据交互的方式使用JSON，则不能在服务间定义严格数据结构，不利于系统稳定。同时后端服务间的交互数据量通常较大，JSON格式的数据本身存在一定的冗余，序列化也相对较慢，因此使用会对服务性能有较大影响。基于以上问题，引入了Protobuf（Google Protocol Buffer）作为服务间的通信标准。Protobuf是一种数据交换格式，可以在使用不同语言的服务间进行通信，提升服务可扩展性[13]。除了支持多种语言，Protobuf还具有以下特点：

（1） 数据量小

Protobuf使用Base 128 Varints加ZigZag方式存储和编码数值。Base 128 Varints是一种变长存储方式，使用每个字节首位作为数字是否结束的标志位，这种存储方式看似增加了冗余，实际却相对节省空间，因为在实际应用中，较小的数字往往占有较大的比例，因此整体来看，使用Base 128 Varints减小了通信数据的长度。但这种方式没有办法表示负数，而Zigzag编码则可以通过正负数交错的方式解决这个问题。数据的类型和值使用key加value的形式，不需要分隔符和标签，使数据更加紧凑。因此，使用Protobuf提升了空间利用率，还能减少网络传输时间。

（2） 序列化速度快

不同于JSON、XML等文本数据，Protobuf传输的是编码后的二进制数据，序列化过程只需通过简单的位运算即可实现，因此不论是序列化还是反序列化速度都更快，极大的提升了服务性能。

使用基于Spring boot框架和Protobuf协议的微服务架构，可以优化通信过程，提升服务性能。

## 2.2 基于Hadoop的数据处理和存储

外部数据进入数据库时，为了最求高并发和高响应速度，会将数据存入Mysql表中，但是当对以天为单位的离线数据进行综合查询时，数据量极大，同时需要跨多表查询，这就需要对数据进行预处理。

Hadoop是一款能够对大量数据进行分布式处理的框架，Hadoop生态圈中包含了HDFS、MapReduce、Hive、Hbase、Sqoop、Pig、Yarn等众多项目和工具，本框架主要用到了HDFS、Hive、MapReduce、Sqoop和HBase，下面对三种工具进行详细介绍。

（1） HDFS（Hadoop Distributed Filesystem）是Hadoop中重要组成部分，它是一个可以用来管理和存储文件的文件系统，它是分布式的，服务器集群中每个节点都扮演着不同的角色完成相应的任务。HDFS的在物理结构上分块存储文件，而且块的大小也可以根据具体应用需求灵活配置。HDFS还会提供一个统一的抽象目录树，客户端可以利用这个抽象目录树通过路径直接访问文件。HDFS的设计模式，是针对数据一次性写入，多次读取的场景，并且不支持修改文件。具备了以上特性，使用HDFS，可以实现对集群的文件、数据的管理，可以保证高可靠性，数据访问的吞吐量和并发量以及数据一致性[14]。

（2） MapReduce是一个使用方便的软件框架。基于MapReduce可以写出可以在上千个机器组成的大型集群上运行的应用程序，不仅能够可以并行处理T级别的数据集，而且还能够保证操作的可靠容错性。通过map主节点输入数据，采用分治的思想把数据分成可以用相同方法处理的小数据集，接下来再把这些小数据集分配给不同的工作节点，每个节点采用同样的方法处理数据，再将数据处理结果返回给树节点。Reduce即是主节点在收集到所有子节点的处理结果之后，将所有结果进行组合封装然后返回输出。

（3） Hive是一种具备SQL风格的可以对任何规模数据进行分析的工具[15]。Hive具备和关系数据库的的SQL命令相似的特点。Hive本质上是通过SQL来处理Hadoop的大数据，处理的数据规模甚至可以伸缩扩展到100PB+，并且数据形式即可以是结构的还可以是非结构的。从Hive的应用场景出发，它更侧重于分析，而不善于实时在线计算。不具备事务机制，并且不具备数据库的灵活性，不能够随意的对数据进行插入和修改。它具备传统数据库没有的通过Hadoop的map/reduce进行分布式处理功能。

（4） Sqoop是传统数据库和Hadoop之间的数据传输和同步工具。Sqoop利用了MapReduce的分布式处理，从而使数据传输加快很多并且大大的增加了信息的容错性。由于sqoop1数据传输和数据结构是紧耦合的，所以connector不能够适应所有数据的格式，并且安全密钥是暴露的，这也难以保证他的安全性。Connector必须要符合JDBC模型并使用通用的JDBC词汇[16-17]。

（5） HBase是构建在HDFS之上的存储系统，它具备分布式，面向列的特点，当遇到实时读写、随机访问超大规模数据集的需求时，就可以借助HBase[18]。它采用线性的方式，从上到下增加节点从而实现扩展。虽然HBase不是关系型的，也不支持SQL，但是它具备自己的优点，它可以将大而散的表放在商用的服务器集群中。HBase的文件存储系统依然是Hadoop HDFS，它还用了Zookeeper来做协同服务。HBase的量级非常大，一个表可以有上亿行，上百万列。HBase具备稀疏的特点，因为空的列不会占用任何存储空间，所以表可以设计的非常稀疏。

## 2.3 基于Holt-Winters的预测器和监控器

Holt-Winters算法是一种基于时间序列进行预测的算法。数据系统中经常需要根据现有数据对未来数据进行预测，分析数据的值和趋势，为管理和运营提供数据支撑，因此本论文引入了时间序列预测算法。由于业务数据具有明显的趋势性和季节性特点，因此框架基于Holt-Winters算法建立了预测器和监控器，实时预测和监控数据。

研究表明，时间序列数据的当前数据与本周的过去数据和过去周期的当前数据具有很大的关联，所以经常会用到移动平均的算法，但是简单的移动平均会抛弃早期的数据，为了不抛弃早期数据，同时让越近的数据占有越大的权值，可以采用加权移动平均的算法。时间序列的趋势性指序列的整体走势，时间序列的季节性指数据呈周期变化的特点，Holt-Winters算法在实现加权移动平均的基础上，还能对含有季节性和趋势性的数据序列进行预测[19-20]。

Holt-Winters算法利用三次指数平滑历史数据，该算法主要使用到四个公式，公式如下。

平滑公式：

 （2.1）

趋势公式：

 （2.2）

季节公式：

 （2.3）

预测公式：

 （2.4）

其中，*α、β*和*γ*为平滑参数，取值均在0到1的范围内，Lt、Tt和St分别为t时刻的平滑因素值、趋势因素值和季节因素值。Yt为t时刻的实际值，Ft为预估值，h为预测值到时间t的时间差，m为周期长度。例如时间序列数据的周期是一天，每次预测的是下一分钟的数据，则m为1440，h为1。

由于实际业务系统的周期性特点不仅呈现日周期性，还在一定程度上表现了周周期性，所以本论文将holt-winters算法的季节公式进行了改进，具体公式如下：

 （2.4）

其中，*γ*和*λ*均为正数，且和小于1。这样，在引入了日周期的同时，也考虑了周周期的影响。

平滑参数的选取可根据经验结合现有变化曲线的平滑性、趋势性和周期性选取。例如：当时间序列水平变化趋势较为平滑时，*α*可选用较小的值。由于业务数据充足，可以根据实际历史数据，利用程序求最小方差得到最佳参数值。Lt、Tt和St的初值对算法的影响并不大，直接取0即可。

## 2.4 基于Fast Cubing算法的数据预计算

Fast Cubing算法是一种快速建立Cube的算法，是Apache Kylin的核心算法。随着硬件价格的降低，时间越来越成为制约系统性能的瓶颈。为了降低多维的查询的查询时间，可以通过预计算和预聚合，达到在常量级时间内实现快速查询[21]。数据通过一定的预处理过程，完成了从业务数据库原始数据到事实表、维度表的构建过程，Apache Kylin会基于这些事实表和纬度表，预计算出所有可能的查询结果（即cube），并存储在HBase中[22]。

Fast Cubing算法实现了快速构建Cube的过程，Cube是按照纬度和指标计算出的所有可能的组合结果，其中每一个组合结果是一个cuboid。Fast Cubing算法能够在离线时间将所有可能的Cuboid计算出来。Kylin在引入Fast Cubing前，使用的是逐层算法，使用该算法的构建一个三维Cube构建过程如图2.1所示。



图2.1 逐层算法Cube构建过程图

Fig. 2.1 Working principle diagram for Dokan

这种算法通过逐层的方式实现Cube构建，先构造一个N维的Cuboid，根据该Cuboid构建N个N-1维的Cuboid，逐层完成Mapper-Reduce过程，随着层次逐渐降低，最终构建出一个完成的Cube。由于该算法需要进行大量MapReduce过程，任务调度消耗较大，每一个的构建都依赖于上一层，需要大量HDFS读写，因此性能消耗大、耗时也更长。Fast Cubing算法则有效避免了上述问题，该算法构建过程如图2.2所示。



图2.2 Fast Cubing算法Cube构造过程图

Fig. 2.2 Working principle diagram for Dokan

Fast Cubing算法先将整个构造过程分成若干个子任务，每个子任务计算一部分完整的Cube块，最终将所有子Cube块组合，得到最终Cube。Cube预计算需要消耗大量时间，但在离线过程的计算并不影响当前数据系统的使用，使用Fast Cubing算法后，提升了计算能力，极大的缩短了Cube的构建时间，使得任务调度压力更小，时间更充足，对线上数据系统的影响几乎可以忽略不计。

## 2.5 基于CAS的SSO过程

微服务架构下，服务数量较多，用户登陆及权限管理困难，所以需要使用CAS（Central Authentication Service）实现SSO（Single Sign On）过程。只需在任何一个服务上进行登陆，就可以访问所有服务。使用单点登录，可以避免用户反复登陆和退出，简化认证和授权过程[23]。CAS认证过程依赖于CAS Server，CAS Server负责统一验证登陆信息，生成认证相关的票据信息，以及对票据信息的验证。每一个需要使用统一认证功能的服务都是一个CAS Client。CAS认证过程如图2.3所示。

图2.3 CAS认证过程图

Fig. 2.3 Process diagram for CAS authentication

用户向web服务即CAS Client发送HTTP请求，CAS Client判断HTTP请求中是否包含Service Ticket，如果不包含话，会将Service重定向到CAS Server端，CAS Server端要求用户登陆认证，认证完成后，CAS Server生成Server端的SessionID和Sevice Ticket并缓存，将带有SessionID和Service Ticke的请求重定向到CAS Client，CAS Client后续的过滤器会请求CAS Server端验证票据信息，验证成功后，会将用户信息写入web服务的Session，CAS Client可以从Session中得到当前用户的username。

当用户再次访问同一个web服务时，服务会检测当前本地Session并当前用户的信息。当用户访问其他统一认证的web服务时，当前服务所代表的CAS Client仍将请求重定向到CAS Server端，此使请求中会带有Server端的SessionID，CAS Server根据SessionID判断用户已经进行过认证，直接根据Session和Service生成Service Ticket。后续重定向、票据验证及web服务Session写入过程和首次认证时相同，因此不在赘述。

本论文在CAS框架的基础上加了两个Filter是其适配于数据系统的接口形式，具体内容在第四章进行详细介绍。

3 系统需求分析

## 3.1 总体需求分析

综合数据管理系统的主要功能是对业务系统的复杂数据的整合、分析和整理，从而实现根据需求对大数据的快速展示和管理。由于业务系统涉及服务数量多、数据量大，因此数据格式复杂，需要数据系统开发出一套能够查看以粗粒度时间、各种空间范围多种维度聚合的离线数据，以细粒度时间、各种空间范围多维度聚合的实时数据，以及根据实时数据进行在线预测和监控的综合数据管理框架。由于数据的敏感性和微服务架构的特点，还要求服务能够实现统一认证功能和日志监控模块。除了以上功能性需求，框架还需达到响应迅速、服务高吞吐量、稳定性强的要求。

根据以上需求，本框架涉及到离线数据管理、实时数据管理、KPI（Key Performance Indicator）管理，认证和日志管理四个模块。按照功能详细规划，离线数据管理包括数据概览、经营主体、客户主题、运营主体子模块；实时数据管理包括监控概览、实时趋势、预测及报警子模块；KPI管理包括业务目标KPI和质量监控子模块；认证和日志管理包括用户单点登录子模块、导出日志子模块、操作日志子模块。下面将对系统的功能性和非功能性需求进行具体介绍。

## 3.2 功能性需求

### 3.2.1 离线数据管理模块

离线数据是一天为单位的历史数据，需要展示企业运营中涉及的核心数据，根据使用数据系统的用户角色不同，需要展示的数据内容也不同。使用系统的主要角色有：管理层、经营规划团队、市场团队。由此将离线数据管理模块分为业务概览、经营主题、客户主题三个子模块，共需监控四十七项指标数据。

（1） 业务概览：业务概览面向管理层，展示各子模块的关键数据，为管理层提供核心数据支持。

① 关键指标看板：展示关键数据，KPI完成进度等数据，为管理层提供业务概览视图。通过日期选择，查看具体日期的离线数据。地图展示当前维度涉及区域范围，默认为全国维度，依次点击，可以进入大区、地区、城市维度，数据展示区跟随区域维度变化，展示具体数据。使用饼图展示核心指标单量和收入模块，表格展示当前维度各区域默认指标的昨日数据。默认展示单量、收入、单均收入、妥投率、时效达成率、人均效能指标数据，通过更多指标可以选取其他指标，最多同时展示八项指标内容。关键指标处有指示灯，指示KPI完成情况。表格上方可以选择区域类型，可以依次查看全部、分试点和重点区域的各项指标数据。

② 客户表现：展示客户经营状况概览，提供收入和单量占比情况，提高对合作客户的表现和合作趋势的评估效率。通过日期选择查看具体日期的离线数据，通过标签选择收入或单量指标，通过多级菜单选择具体区域维度。默认展示昨日全国维度收入。饼状图展示行业数据占比和客户数据占比，最多展示Top10。利用表格展示具体数据和日环比。

③ 运营效率：展示运营关键指标监控度以及对经营状况进行分析，支持营运团队实现数据化运营，全面透视业务运营情况和趋势。日期和地区控件选择需要展示的日期和空间维度，默认展示昨日维度数据。通过雷达图展示全职人均效能、单均计提、订单完成率和时效达成率四项指标的当日数据和KPI值。通过二维坐标图展示健康经营诊断图，展示思想指标的合理度和内部评分。

（2） 经营主题：经营主题面向经营规划团队，为经营规划团队提供KPI考核相关指标的完成情况。。

① 收入KPI：展示收入KPI相关指标和达成状况。通过日期选择展示当日离线数据，默认展示昨日数据。通过展示全国维度各大区订单量、收入、单均收入、月累计收入、月收入KPI达成进展、年累计单量、年累计收入，年收入KPI达成进展等指标。对月累计收入、年累计收入指标加KPI指示灯，展示达成进度和时间进度的比较结果。通过柱状图展示地区当前收入和目标收入。

② 质量监控KPI：展示质量监控相关KPI的指标和达成状况。通过日期和指标选择控件选择具体日期和指标，默认展示昨日的订单完成率。通过柱状图展示各地区指标达成率、全网平均值和指标KPI值。

（３） 客户主题：业务概览面向市场团队，从客户维度展示关键指标情况，提高市场部对客户合作价值和客户质量的评估效率。

① 各区表现：展示客户各项指标在各地区的表现情况。通过日期、品牌、指标筛选各地区当日指定商家的部分指标的数据，默认展示各地区昨日全部商家总完成单量、总收入、票均收入、订单完成率、时效达成率和店铺数信息。

② 客户关键指标趋势图：展示客户关键指标随时间的变化趋势。通过选择起始日期、商家、地区维度、指标类型筛选数据，默认展示上月全国维度所有商家的各指标数据。可选择多组折线图或者表格两种展示形式。

除了以上需求，离线数据管理模块的所有子模块数据除了网页展示外，均需提供数据导出功能，导出后的数据，可在任务列表查看导出状态，导出成功的数据可以以Excel形式下载，供各团队使用。

### 3.2.2 实时数据管理模块

管理运营团队希望对业务运营情况进行实时监控，及时发现异常情况，并进行干预，降低风险。监控维度展示当前日期信息及业务覆盖信息。日期不可修改，为实时数据，业务覆盖信息包括昨日业务覆盖到的城市、客户品牌和店铺数。监控维度分为全国维度和城市维度，监控内容分为监控概览、实时趋势图、预测和报警器三个模块。共需要监控二十七项指标实时状况。

（1） 监控概览：监控概览显示一分钟全国或某城市的核心指标数据。由于数据同步延时问题，默认显示当前分钟数减二时刻所在分钟数数据。监控概览还包括两个子模块，分别是业务概览和运力监控

① 业务概览：通过指标选择需要的数据，展示全国或某城市各项指标的实时数据、日同比和周同比。默认展示完成单量、资源在线率、时效达成率和超时接单率四项指标。

② 运力监控：通过地图展示人均为完成单量大于0的城市；通过表格展示各城市人均未完成单量、未完成单量、可派单骑士数三项指标的数据。按人均未完成单量排序，展示Top10数据。

（2） 实时趋势图：实时趋势图以曲线图的形式展示全国或某城市今日、昨日、上周各项指标一天的完整数据。每一天的数据的一个指标是一条曲线，可以通过指标选择具体需要展示的数据，最多选择八个指标。默认展示实时进单量、未完成单量、人均未完成单量和资源在线率四项指标。可派单骑士数接口除了展示三天的实时可派单骑士数据外，还应展示当前时刻具体配送员相关数据，包括供应商骑士、同城兼职骑士数据、同城非全日制专职骑士和同城全日制专职骑士数据。

（3） 预测和报警器子模块：由于实时进单量对于业务运营具有重要意义，当实时进单量过大时，可能造成运力压力过大，因此对于实时进单量信息，管理运营团队希望根据历史数据预测下一分钟数据，同时将每一分钟的数据和预测数据进行对比，如果和预测数据比对，偏差较大，则产生报警信息。

### 3.2.3 KPI管理模块

KPI管理由业务团队使用，KPI根据指标类型分为业务KPI和质量监控KPI，根据时间范围分为月度KPI和年度KPI。KPI管理模块需要实现KPI展示、KPI批量导入、单个地区KPI修改、KPI下载功能，KPI管理模块用例如图3.1所示。



图3.1 KPI管理用例图

Fig. 3.1 Use Case diagram of KPI management

（1） KPI展示：KPI展示模块提供分类展示功能，根据KPI类型和时间，需要展示所有区域KPI值。默认展示当前月份或年份数据，可以选择其他时间。业务KPI需要展示收入和完成单量KPI数据，质量监控KPI需要展示取货违规率、配送违规率、超时接单率、订单完成率和时效内达成率KPI值。

（2） KPI批量导入：由于KPI涉及的指标和业务地区划分会随着时间变动，因此KPI模板需要动态生成，即需要用户先下载KPI模板。KPI模板下载需要根据指标类型、时间维度类型、地区表数据、KPI指标实时生成所需模板。模板包括地区信息、KPI名称和填写说明等信息，以Excel形式保存。用户填写下载的KPI模板并上传后，可以批量导入当前、未来月份的KPI值。系统需要对表格数据进行校验，校验时间是否符合导入要求，校验城市无重复、缺失，校验指标和导入的模块符合，同时还要根据指标具体情况校验数据类型和数值范围，校验成功后上传数据库。如果数据库没有相关数据则插入新数据，否则更新旧的数据。上传后提供反馈信息，若失败，需要提供失败原因和导致失败的数据。

（3） 修改单个地区KPI：选择需要修改的地区，并在各指标处输入需要更新的指标值，不需要更新的指标不用填写。系统判断系统是否由历史数据，如果没有，需要先导入。对修改的数据进行校验，校验时间是否符合要求，校验值类型和数值范围。校验成功后，更新数据库。修改完成后，提供反馈信息，若修改失败，需提示失败原因。

（4） KPI下载：对于已经上传过的数据KPI，有时候需要批量修改重新上传，因此需要实现KPI下载功能。需要将当前月份或年份的所有地区的各项KPI值导入Excel中，用户下载后直接编辑。

### 3.2.4 认证和日志管理模块

系统需要统一认证，登陆后展示用户工号，由于数据的敏感性，所有数据操作均需记录操作人、操作详细信息和操作时间。根据以上需求，将认证和日志管理模块分为用户单点登录子模块、KPI日志和数据日志三个子模块，下面对三个子模块进行具体介绍。

（1） 用户单点登录：用户打开系统下任一子服务的任一界面，系统都会判断用户是否登陆，并请求系统工号。所有数据服务均需登录后才能使用，同时登陆任何一个子服务，即可访问其他服务。登录后系统保存登录时间、登陆员工工号等信息。

（2） KPI日志：用户对KPI的任何操作都可以通过KPI日志查看。KPI日志需记录用户的操作时间，用户的工号，修改的区域指标、类型和值变化情况。如果是批量首次导入，需要记录导入的所有值信息。每次操作是一条日志。日志按操作日期从近期到远期排序，分页展示。

（3） 数据日志：所有数据的下载、导出都需要在数据日志部分记录。导出日志需要记录导出模块和任务名称，记录导出时间、保存时间和员工编号。同时由于数据量较大，还可以通过业务线、任务名称、导出时间和员工编号进行筛选，查看所需数据。日志按照操作时间从近到远排序，分页展示。

## 3.3 非功能性需求

非功能需求指除了功能之外的需求，这些需求不影响具体功能的实现，但这些需求直接关系着用户的使用体验和系统的整体可用性，对于系统的运行和发展具有重要意义。下面将对性能需求、可扩展性需求和可靠性需求等非功能性需求进行详细介绍。

（1） 性能需求：性能是影响用户体验的最直观因素，只有接口的响应速度在用户接受范围内时，才会给用户流畅、舒适的体验感。综合数据管理系统的离线数据涉及的数据量大，且指标计算过程复杂，所以要求接口响应时间应在1.5s以内，其他接口响应时间均应在1s以内。

（2） 可扩展性需求：可扩展性需求是一个系统能够得到长期运行和维护的基本要求，只有满足可扩展性需求，才能在企业发展过程中，不断满足新的需求。因此系统需要满足可扩展性需求，保证系统和框架的良性发展。

（3） 可靠性需求：系统的SLA标准需要达到99.9%，不允许单点存在，支持横向扩展和多机房部署，同时能够实现监控。此外，离线数据系统数据在工作时间内能够使用，其他数据系统在所有时间内都需达到可用。

（4） 安全性需求：要求测试环境和生产环境系统分离、数据分离。限制线上数据系统的读写账号的远程访问IP只能是线上服务器，测试服务器只能通过只读账号访问生产环境数据。要求技术人员访问生产环境需使用密码加token码的形式，非技术人员不得访问生产环境数据库。

4 系统设计

## 4.1 总体设计

### 4.1.1 微服务架构设计

根据功能模块将系统分为四个服务，利用基于Spring boot和Protobuf的微服务架构连接各服务。同时根据模块内部提携结构将系统分为表现层、业务逻辑层和数据层四层，体系结构图如图4.1所示。



图4.1 系统体系结构图

Fig 4.1 System architecture diagram

表现层主要时用户的操作界面，用户可以通过表现层与系统交互，使用离线数据管理服务、在线数据管理服务、KPI管理服务以及认证和日志服务。这部分主要使用HTML网页编程技术，CSS样式调整服务、Echarts动态数据展示、JQuery网页控件调用，Ajax动态交互，Javascript的JSON数据交换等Web开发技术。

业务逻辑层能够通过控制器接受HTTP请求，主要是GET、POST请求，然后调用响应业务处理逻辑，通过一系列后台运算和对数据层的调用操作，最终实现用户需求或返回表现层需要的结果。业务逻辑层与表现层通过JSON返回数据。Spring boot框架内部整合的Jackson框架可以将后台返回的实体对象自动转换为JSON对象，也可以将前端的JSON格式请求自动注入到Java实体对象中。离线数据管理、在线数据管理、KPI管理以及认证和日志管理四个服务的业务数据请求也在业务逻辑层实现。利用Protobuf协议，各服务间可以通过强Schema方式转换调用对象。相较于JSON，传输同样的内容，Protobuf交换的数据量更小，速度也更快。

数据层主要是和数据库直接交互的模块，业务逻辑层对数据库的增、删、改、查都在数据层完成。利用Spring boot整合的JPA接口，可以简化对数据库的操作编程。只需要继承指定接口，然后按照规定的格式定义方法名，就可以达到数据库的操作目的。

基于Spring boot和Protobuf的微服务架构，为系统的可扩展性提供了有力支持，同时也在一定程度上提升了系统的性能和可靠性。

### 4.1.2 离线数据处理流程设计

离线数据系统设计到四十多个指标，指标的计算方式主要有：直接同步、统计、加和、一次计算、两次计算等方式。由于数据以天为单位，所以数据量很大、计算过程复杂，下面将离线数据流程进行介绍。

（1） 同步线上数据到数据库。梳理需要用到的指标，从业务系统数据库表中找到对应的字段。梳理线上所用的指标后，从线上数据库梳理、统计数据来源，将需要使用到线上的表格同步到Mysql的DTS表中。各表表名及说明见表4.1，表名中的loc、tc、da为业务表的来源，DTS表每天零点从业务系统同步数据到数据系统的Mysql中，业务涉及到的表共计16个，本文只展示其中几项。

表4.1 DTS表说明

Tab. 4.1 Description of DTS table

|  |  |
| --- | --- |
| DTS表 | 说明 |
| loc\_order\_order\_list | 订单基本 |
| tc\_open\_api\_order\_basic | 订单api |
| loc\_rider | 骑士子账号 |
| tc\_merchant\_shop\_basic | 店铺基本 |
| loc\_schedule\_info | 排班信息 |
| da\_city | 城市表 |

（2） 将数据从Mysql导入Hive。通过Python脚本编写Sqoop命令将DTS表数据导入到Hive中。脚本中指定数据每一个字段的数据来源和具体信息，Sqoop通过MapReduce实现数据的同步过程。由于离线数据最小聚合维度是天，所以利用定时任务控制脚本零点执行进行同步。由于系统需要的指标中，有很多数据不能直接通过同步得到，需要多次计算得到，因此将数据同步到temp\_dts、temp\_distance、temp\_judge表，temp\_dts表的主要内容参考表4.2，由于涉及字段有98个，只写出其中一小部分，temp\_distance和temp\_judge不再展示。部分可以直接生成的表直接同步fact表中，如fact\_shop表，表格内容参考章节4.2.1。

表4.2 temp\_dts表说明

Tab. 4.2 Description of table temp\_dts

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | dts表来源 | 类型 | 注释 |
| order\_id | loc\_order\_order\_list | BIGINT | 订单ID |
| create\_date | loc\_order\_order\_list | DATE | 订单创建时间 |
| out\_order\_id | loc\_order\_order\_list | BIGINT | 外单ID |
| order\_index | loc\_order\_order\_list | BIGINT | 订单小号 |
| delivery\_type | loc\_order\_order\_list | TINYINT | 订单配送类型 |
| is\_test | tc\_open\_api\_order\_basic | BOOLEAN | 是否是测试单 |
| is\_dubious | tc\_open\_api\_order\_basic | BOOLEAN | 是否是可疑单 |
| region\_name | region | STRING | 大区名字 |

（3） 利用Hive做MapReduce。用Hive SQL生成需要的字段和表信息后，Hive根据SQL转换成MapReduce，进行进一步的处理，生成需要的fact表。最终生成fact\_order、fact\_shop、fact\_ridertime表。最终生成的Hive表参看章节4.2.1。

（4） 梳理Hive表内容，根据数据系统的查询需求将需要在同一个接口中查询的数据建立在同一个Cube中。梳理各项指标的使用到的字段的计算方式，解析计算时用到的维度，及需要预聚合的维度的各种组合方式整理好，最终建立Cube。通过这种方式建立Cube，可以避免数据过度膨胀。Kylin根据预聚合、预计算的要求建立Cube，每天增量备份数据，并将数据保存HBase中，最终实现对于离线数据的预处理过程。根据各模块的组合查询需求，建立了6个Cube。业务概览-关键指标看板、经营主题-质量监控KPI子模块使用一个Cube，该Cube可能使用到的维度组合见表4.3。其余子模块各使用一个Cube。Cube具体内容参考章节4.2.2。

表4.3 Cube维度组合方式

Tab. 4.3 Dimension combination of Cube

|  |  |
| --- | --- |
| 维度组合 | 组合方式 |
| 维度组合1 | order\_date,source\_name,order\_status,in\_shixiao,is\_early,is\_overtime,is\_illegal,is\_pickup\_illegal,is\_distribution\_illegal,is\_take\_delivery\_illegal,is\_redistributed,in\_30min,in\_40min,in\_60min,in\_4h,distribution\_duration,region\_id,region\_area\_id,region\_area\_city\_id,city\_id,city\_supplier\_id,city\_supplier\_shop\_id,city\_supplier\_shop\_rider\_id |
| 维度组合2 | order\_date,source\_name,order\_status,in\_shixiao,is\_early,is\_overtime,is\_illegal,is\_pickup\_illegal,is\_distribution\_illegal,is\_take\_delivery\_illegal,is\_redistributed,in\_30min,in\_40min,in\_60min,in\_4h,distribution\_duration,region\_id,region\_area\_id,region\_area\_city\_id,city\_id,city\_aoi\_id,city\_aoi\_rider\_id |
| 维度组合3 | order\_date,source\_name,order\_status,in\_shixiao,is\_early,is\_overtime,is\_illegalis\_pickup\_illegal,is\_distribution\_illegal,is\_take\_delivery\_illegal,is\_redistributed,in\_30min,in\_40min,in\_60min,in\_4h,distribution\_duration,area\_id |

（5） 部分表信息，如city、area、region等表信息，由于需要和前端交互，为了提升数据查询速度，仍需使用Sqoop同步数据到MySql中。

### 4.1.3 实时数据处理流程设计

实时数据的涉及到17个指标，指标的计算方式主要有：直接同步、统计、加和、一次计算、二次计算等方式。实时数据指标以分钟为单位，如果每次请求数据都要进行计算，用时较长，且如果一次请求多个指标，计算量过大会导致服务性能极差。因此可以使用Quartz定时器每分钟进行将所有指标的数据进行预计算，不同维度的数据也提前进行计算，这个计算过程通过SQL优化可以控制在一分钟以内。各服务的数据层使用数据时，从处理过的表中直接读取数据或者进行简单的一次计算即可得到需要的数据。具体处理流程如下。

（1） 同步线上数据到数据库。梳理需要用到的指标，从业务系统数据库表中找到对应的字段。梳理线上所用的指标后，从线上数据库梳理、统计数据来源，将需要使用到线上的表格同步到Mysql的DTS表中。根据梳理，用到表有：loc\_order\_order\_list、loc\_order\_order\_info、tc\_open\_api\_order\_basic、等9个表，各表的说明可以参考表4.1， DTS表每分钟从业务系统同步数据到数据系统的Mysql中。

（2） 梳理各指标具体计算方式，使用Quartz定时器实时计算数据，每分钟计算一次。实时数据管理系统涉及指标17项，本文只介绍部分指标的计算方式，具体内容参考表4.5。还有店铺信息，每日变动不大，所以定时器规定每天计算一次。所有计算后的数据保存在Mysql中，便于实时查询，具体内容参考4.2.3小节。

表4.5 实时指标计算方式

Tab. 4.5 Real-time calculation method for index

|  |  |  |  |
| --- | --- | --- | --- |
| 数据需求 | 表 | 计算方式 | 注释 |
| count\_city\_has\_order | order\_list | select count(distinct city\_id) from order\_list where list\_time between 0 and current. | 城市数 |
| count\_brand\_has\_order | order\_list | select count(distinct brand) from order\_list left join shop\_basic on order\_list.shop\_id=shop\_basic.shop\_id where list\_time between 0 and current. | 客户数 |
| avg\_order\_not\_finish | Order\_list Rider | count\_order\_not\_finish / count\_rider\_distribution | 人均未完成单量 |
| count\_order\_in\_minute | order\_listorder\_info | order\_status=9100 finish\_time between current-60 and current. | 实时完成单量 |

## 4.2 数据管理设计

### 4.2.1 Hive表设计

通过预处理，最终得到的fact表为：fact\_order、fact\_shop、fact\_ridertime，其中fact\_order表的最终字段有143个，fact\_shop表字段有39个，fact\_ridertime表字段有47个。最终得到的维度表为：area表、city表、region表、rider表、industry、brand表。由于表格量大，每个表格内的字段多，本文只将其中重要表格的重要字段列出，其余部分不做展示。

fact\_order表的主要是和订单相关的表，展示的是核心数据，包括订单号、订单状态、配送员信息、店铺信息、配送位置等内容，具体内容参考表4.5，由于表格属性过多，下面只展示其中小部分内容。

表4.5 fact\_order表说明

Tab. 4.5 Description of table fact\_order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 注释 | 计算方式 | 默认值 |
| out\_order\_id | BIGINT | 外单ID | loc\_order\_order\_list.out\_order\_id | 0 |
| group\_id | BIGINT | 订单组编号 | loc\_order\_order\_info.group\_id | 0 |
| order\_index | BIGINT | 订单小号 | loc\_order\_order\_list.order\_index | 0 |
| order\_flag | BIGINT | 订单标记 | loc\_order\_order\_list.order\_flag | 0 |
| categry\_name | STRING | 订单品类 | loc\_order\_order\_list.category\_name | 空字符串 |
| rollback\_type | BIGINT | 订单撤回原因 | loc\_order\_order\_info.rollback\_type | 0 |

fact\_shop表是店铺的信息，包括店铺所属的商圈、城市等信息，表格具体有39个字段，本文只展示部分字段。表格具体内容参考表4.6。

表4.6 fact\_shop表说明

Tab. 4.6 Description of table fact\_shop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 注释 | 计算方式 | 默认值 |
| shop\_id | BIGINT | 店铺id | tc\_merchant\_shop\_basic.shop\_id | 0 |
| shop\_name | STRING | 店铺name | tc\_merchant\_shop\_basic.shop\_name | 空字符串 |
| supplier\_id | BIGINT | 商家id | tc\_merchant\_shop\_basic.supplier\_id | 0 |
| [longitude](javascript:void(0)) | STRING | 坐标经度 | tc\_merchant\_shop\_basic.longitude | 空字符串 |
| [latitude](javascript:void(0)) | STRING | 坐标纬度 | tc\_merchant\_shop\_basic.latitude | 空字符串 |
| [contacts\_name](javascript:void(0)) | STRING | 联系人姓名 | tc\_merchant\_shop\_basic.contacts\_name | 空字符串 |
| [contacts\_phone](javascript:void(0)) | STRING | 联系人电话 | tc\_merchant\_shop\_basic.contacts\_phone | 空字符串 |

地理信息主要保存在city、area、region表中。其中city表保存了城市信息，包括城市的id、名称、所属地区、城市坐标等内容，表格的具体信息参考表4.7。area表展示了地区信息，包括地区id、名称，所属大区、地区坐标等内容，表格的具体设计不做详细展示。region表保存了大区的id、名称、大区坐标等信息，具体内容也不做展示。

表4.7 城市信息表

Tab. 4.7 Table of city

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 注释 |
| id | int(10) | 0 | city的ID值 |
| region\_id | int(10) | 0 | 所属大区ID |
| area\_id | int(10) | 0 | 所属地区ID |
| city\_id | varchar(50) | 空字符串 | 城市编码 |
| sf\_city\_id | varchar(50) | 空字符串 | sf城市编码 |
| city\_name | varchar(50) | 空字符串 | 城市名称 |
| city\_cp | varchar(50) | 空字符串 | 城市坐标 |

店铺的相关信息信息保存在aoi、brand、industry等表格中，主要保存了商圈信息、品牌信息、行业信息。商圈表aoi表设计如表4.8所示，展示了商圈的城市、位置、名称、配送范围等信息。brand和industry表的具体信息不做详细展示。

表4.8 商圈信息表

Tab. 4.8 Table of aoi

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 注释 | 默认值 |
| id | BIGINT(20) | 商圈 | 0 |
| city\_id | INT(11) | 城市ID | 0 |
| lc\_id | INT(11) | 配送中心ID | 0 |
| name | VARCHAR(50) | 商圈名称 | 空字符串 |
| map\_coords | MEDIUMTEXT | 商圈范围 | 空字符串 |
| simplified\_map\_coords | MEDIUMTEXT | 简化后的商圈范围 | 空字符串 |
| creator | VARCHAR(255) | 创建人账号 | 空字符串 |

### 4.2.2 Kylin的Cube设计

根据查询的聚合需求，将整个离线部分的大数据查询整合到了六个Cube中，下面对业务概览-关键指标看板、经营主题-质量监控KPI子模块使用到的Cube进行详细说明，该Cube使用到字段有26个，本论文介绍其中部分字段，具体内容参考表4.9。

表4.9 Cube说明

Tab. 4.9 Description of Cube

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | fact表来源 | 计算方式 | 使用维度 | 注释 |
| count\_order | order | count(1) | 无 | 总单量 |
| count\_order\_is\_finished | order | count(1) | order\_status='9100' | 总完成单量 |
| count\_order\_is\_cancelled | order | count(1) | order\_status='9100' | 总取消单量 |
| count\_order\_in\_shixiao | order | count(1) | order\_status='9100' | 时效达成单量 |
| count\_order\_is\_pickup\_illegal | order | count(1) | order\_status='9100'&& is\_pickup\_illegal=1 | 取货违规单量 |
| count\_order\_is\_distribution\_illegal | order | count(1) | order\_status='9100'&& is\_distribution\_illegal=1 | 配送违规单量 |

### 4.2.3 MySQL表设计

离线数据中有一部分数据字典信息，其他模块也需要使用，如city、area、region、task表等信息。即使这些表信息在Hive中存储了一份，依然需要在Mysql中再次创建，以提升和前端交互速度。由4.2.2小节已经做了介绍，本小节不在赘述。除了上述表，实时数据管理服务中需要使用的Mysql表有guard\_index\_minute和guard\_other表。KPI管理服务需要使用的Mysql表有KPI、KPIDD表，用户和日志部分需要使用的表有user、data\_log、op\_log表。

定时器每分钟向guard\_index\_minute表中写入当前分钟的数据，表中保存了当前城市、日期、时间、各项指标数据等内容，表中共有24个字段，本文只介绍其中部分字段，guard\_index\_minute表的内容参考表4.10。定时器每天将店铺相关信息写入guard\_other表中，表中保存了城市信息、日期、店铺数等信息，具体内容不做展示。

表4.10 guard\_index\_minute表说明

Tab. 4.10 Description of table guard\_index\_minute

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 注释 | 默认值 |
| id | BIGINT(20) | 自增ID | 0 |
| city\_id | INT(11) | 城市id | 0 |
| index\_day | INT(11) | 日期 | 0 |
| index\_minute | INT(11) | 时间 | 0 |
| count\_order\_is\_finished | INT(11) | 完成单量 | 0 |
| count\_in\_order\_list | INT(11) | 实时进单量 | 0 |

KPI管理服务涉及到KPIDD表和KPI表，KPIDD主要保存了KPI类型和数据类型等信息，KPI表则保存了每个KPI值和值对应的地区、时间、类型等信息。KPI表内容参考表4.11，KPIDD表内容不做具体展示。

表4.11 kpi表说明

Tab. 4.11 Description of table kpi

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 注释 | 默认值 |
| id | BIGINT(11) | 自增ID | 0 |
| kpiid | VARCHAR(10) | kpiid | 空字符串 |
| kpitime | VARCHAR(20) | kpi时间 | 空字符串 |
| areaid | VARCHAR(10) | 城市编码 | 空字符串 |
| kpivalue | VARCHAR(20) | kpi值 | 空字符串 |
| status | VARCHAR(40) | 审批状态 | 空字符串 |

认证和日志管理服务涉及user、data\_log和op\_log表，user存储了用户工号、登陆时间等信息，op\_log表存储了操作时间、操作详情、操作人工号等信息，data\_log表保存了导出和下载操作的具体信息，存储了数据源、任务名称、导出时间、保存时间、操作人工号等信息。

## 4.3 系统功能模块设计

### 4.3.1 离线数据管理模块

离线数据管理模块分为三个子模块，但是每个模块的核心任务都是数据查询。数据查询的核心是数据预处理和预计算，这个模块的具体设计在4.1.2小节进行了详细介绍，使用数据时，可以直接从Kylin读取提前处理好的数据。因此这三个模块的核心任务均被抽象成了访问Kylin，并将访问到的数据直接拼凑或简单计算得到最终的返回数据集。除了数据查询部分，其余接口均为数据查询做辅助，主要是为了得到各数据字典信息。根据这种抽象方式，将离线数据管理模块分为数据查询模块和数据字典模块两个部分。

（1） 离线数据的查询模块，通过Kylin得到具体数据，而对Kylin的查询，需要知道具体查询的Cube，此时需要告诉Kylin查询的groupby维度和setting的维度。根据这些条件，为数据访问的查询接口设计了如下参数：

① groupby参数：该参数主要设置了groupby维度，可以取值region\_id、area\_id、city\_id，表明对Kylin查询聚合的空间维度。

② measures参数：表明需要查询的指标列，可以取值：count\_order、count\_order\_is\_finished等。离线数据管理服务涉及的四十余项指标，均可出现在参数中。但根据需求，已经将此部分分为了若干个小模块，并根据小模块建立了Cube，因此同时出现的指标都是在同一个小模块用到的指标。

③ setting\_ds参数：表明设定的维度，此参数中包含了start\_time、end\_time、brand\_id、region\_id、area\_id等信息，其中start\_time和end\_time为必选项，其他项则根据实际情况进行设置。

举个例子：当需要查询A大区下B区域中所有城市从2018年1月1日的C、D指标数据，那么group\_by取值“city\_id”，measures取值“C，D”，settings\_ds的start\_time和end\_time均取值“2018年1月1日”，region\_id取值“A”，area\_id取值“D”。通过这种动态拼凑参数的方式，将Kylin查询使用到的维度拼凑出来，最大限度的抽象出了一个通用的数据查询接口。

数据查询接口通过Controller接口接受响应的请求，然后调用Handler类进行具体的业务处理，Handler利用KylinJDBC和KylinClient实现对Kylin的访问和查询，得到查询结果，将返回对象组装成需要的Response对象，并由Controller返回。Spring boot将对象转化为JSON返回给前端。此接口涉及的类图如下图4.2所示，由于部分类成员和方法较多，本文只展示主要内容。

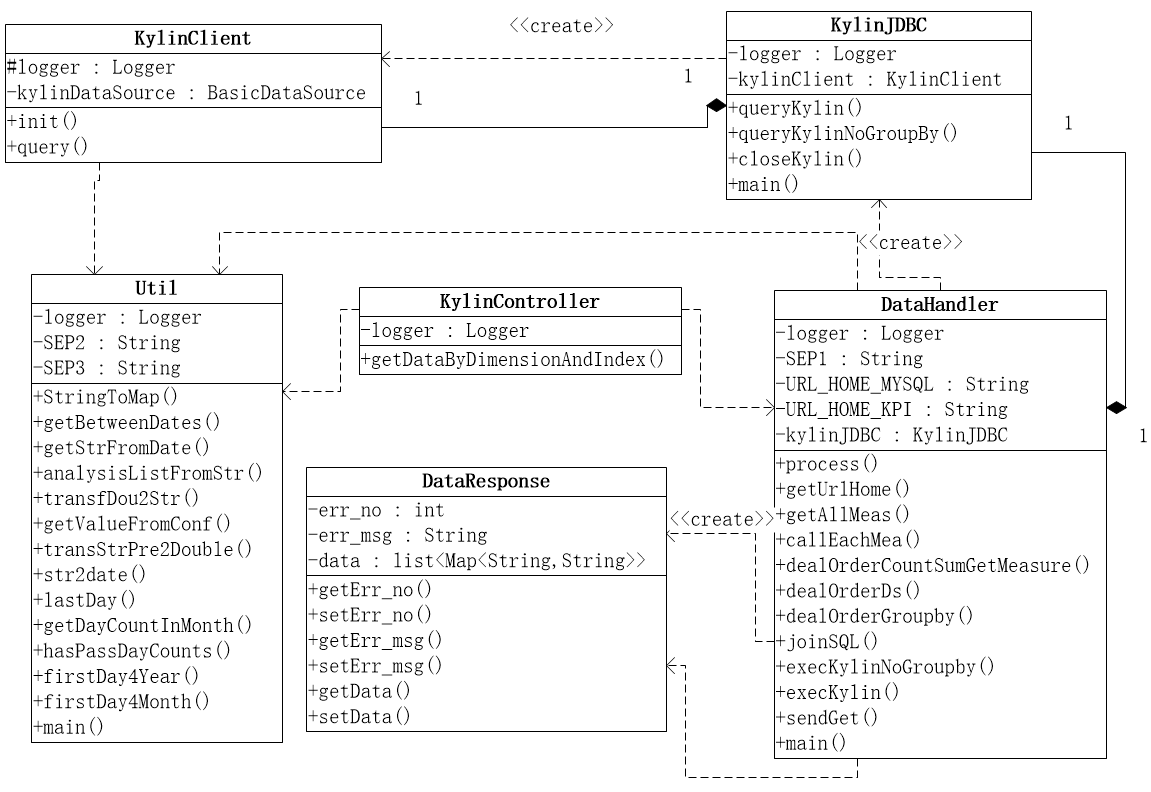


图4.2 离线数据查询模块类图

Fig 4.2 Class diagram of offline-data query sub-module

（2） 数据字典模块设计，除了得到数据信息，各个模块还需得到默认指标、可用指标、大区、地区、城市等信息。这些数据的获取均通过表现层、业务层、数据层三层模型得到，Contrller类中接受并响应前端请求，然后调用对应的service接口，对应接口的实现类serviceImpl进行进一步的逻辑处理，然后调用数据层从Mysql中获取响应的数据。得到数据后，业务层会将数据组装成需要返回的对象，最终Controller将结果对象返回。Spring boot将对象转化为JSON对象返回给前端。前端通过请求数据字典信息，可以得到全国、大区、地区的数据信息，然后利用Echarts工具实现地图拼接。通过请求数据接口可以得到各项数据，然后利用Echarts工具实现数据的各种图类展示形式。

（3） 数据导出和下载模块：由于离线数据数据量大，前端展示数据时，可以分页展示，分次请求，导出数据则不能，同时导出过程需要进行Excel写入和文件流传输，对导出性能有较大影响，导致一次导出耗时较长，影响用户体验。针对导出的耗时，增加了任务列表子模块。当Controller接受请求后，向task表写入一条任务，将任务的状态设置为导出中，然后关闭HttpServletResponse的输出流，接口会提前返回，此时用户无需等待，可以进行其他查询、导出操作，或查看任务列表中任务的导出状态。服务器继续进行导出工作，利用HDFS实现文件的存储。导出完成后，将任务状态修改为导出完成，并提供下载URL，点击URL可以直接请求下载接口，下载接口从HDFS获取文件，并通过输出流返回给前端。

### 4.3.2 实时数据管理模块

实时数据管理模块分为监控概览、实时趋势图、预测和报警子模块。实时数据的处理预处理方式需要使用Quartz定时器，定时器任务主要是：每天定时读取前一天的有单的店铺、城市信息，每分钟根据表4.10的计算方式定时读取各城市、全国范围下的各项指标。定时器实时完成数据准备工作，当聚合维度是城市时，通过业务数据的city\_id进行计算，city\_id保持不变，当聚合维度是全国时，将所有city\_id数据进行取和，并将city\_id设置为0，将所有定时器的设计类图如图4.3所示，图中只展示主要内容。

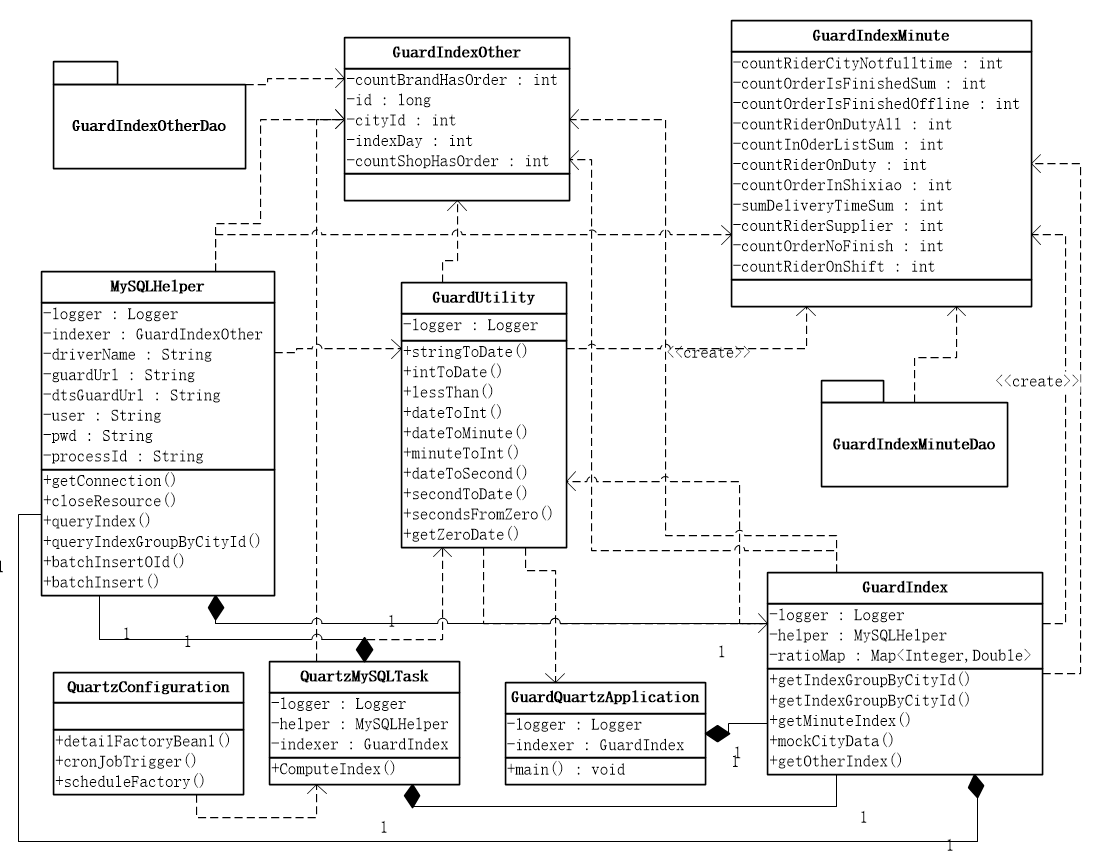


图4.2 定时器模块类图

Fig 4.2 Class diagram of realtime-data query sub-module

与前端交互的数据主要是监控概览、实时趋势图、监控和报警子模块三个模块，三个模块的具体设计如下：

（1） 监控概览子模块：监控概览子模块展示的是实时业务的核心数据，这个模块显示的是每分钟的实时数据，分为业务概览和运力监控两个模块，业务概览子模块的指标信息通过XML文档保存，通过读取指标接口可以从XML中获取信息。

① 业务概览子模块：前端发送请求的多个指标的信息，通过Spring boot自动注入GetMinuteRequest对象中，业务概览通过调用GuardController接口接受前端请求，GuardController调用处理的业务逻辑GetMinuteHandler类处理具体的业务逻辑。调用过程中将对应的数据处理对象GetMinuteDao通过参数传入处理完成后，GetMinuteHandler根据服务器时间请求数据库，并将返回的数据拼接成返回对象，业务逻辑处理完成后，返回GetMinuteResponse对象，并通过Spring boot的Jackson框架自动将返回对象转化为JSON对象。业务概览子模块返回的是每分钟数据，定时器处理过各项指标的数据存入数据库后，业务概览模块通过Dao请求当前、昨日、上周此分钟数据，并将数据注入MinuteData对象中，每个指标的具体值可以从对象中获取的直接获取数据或一步计算得到指标数据，同时可以计算得到日同比和周同比返回前端。

② 运力监控子模块：运力监控通过请求当前全国人均未完成单量排名前十的城市数据，此模块包括人均未完成当量、未完成单量、可派单骑士三项与运力相关的数据，并将数据展示在表格中。同时请求城市信息接口获取当前所有城市的city\_id和对应位置信息。数据库中访问City表的数据层位于其他服务中，因此需要使用ProtoBuf访问，定义好Protobuf数据结构，利用Protobuf根据数据结构生成需要的City类。handler利用Protobuf生成的City类请求到地区信息。匹配数据接口和城市接口的city\_id，得到人均未完成单量排名前十的城市的位置信息，并根据人均未完成单量的数值大小定义颜色深浅，将信息展示在地图上。点击地图的颜色可以进入对应的城市界面。

（2） 实时趋势图子模块：实时趋势图模块展示了各项指标当日、昨日、上周同日的数据。每项指标的三日值构成一个画有三个直线的折线图，每条线上有1440个点，对应一天里的1440个分钟。由于凌晨时，业务量较低，会导致点数不一致，此时需要程序将缺少的点自动补齐。前端发送请求的多个指标的信息，通过Spring boot自动注入GetCurveRequest对象中，业务概览通过调用GuardController接口接受前端请求，GuardController调用处理的业务逻辑GetCurveHandler类处理具体的业务逻辑。调用过程中将对应的数据处理对象GetMinuteDao通过参数传入处理完成后，GetMinuteHandler根据服务器时间请求数据库，请求分为三次，分别请求三日对应city\_id的所有数据，存放在List<MinuteData>中，得到数据层返回的数据后，GetCurveHandler会对数据进行补全操作，最终形成List<MinuteData>数据，然后根据指标内容计算所需的的值，拼装GetCurveResponse对象。实时趋势图子模块的指标信息通过XML文档保存，通过读取指标接口可以从XML中获取信息。

由于返回对象数据较大，使用Jackson框架只能在一定程度上提升了性能，耗时无法控制在100ms以内，所以针对此接口，使用StringBuilder进行String拼接，然后HttpServletResponse将返回对象类型设置为JSON即可。

（3） 预测和报警子模块：预测和报警子模块采用基于Holt-Winters算法的时间序列预测数据，此模块主要针对实时进单量指标进行预测和监控。将Holt-Winters算法结合订单量的实际情况进行改进，加入*λ*参数，体现上周同日的季节因素。取一个月的数据进行训练，然后利用Holt-Winters算法进行参数计算。Holt-Winters算法的各参数通过计算可以通过四层循环求解，将日方差均值最小的时的*α、β*、*γ*和*λ*参数的组合作为最后选用的参数。根据此参数进行预测，与实际值进行对比，对比结果如下图4.3所示。

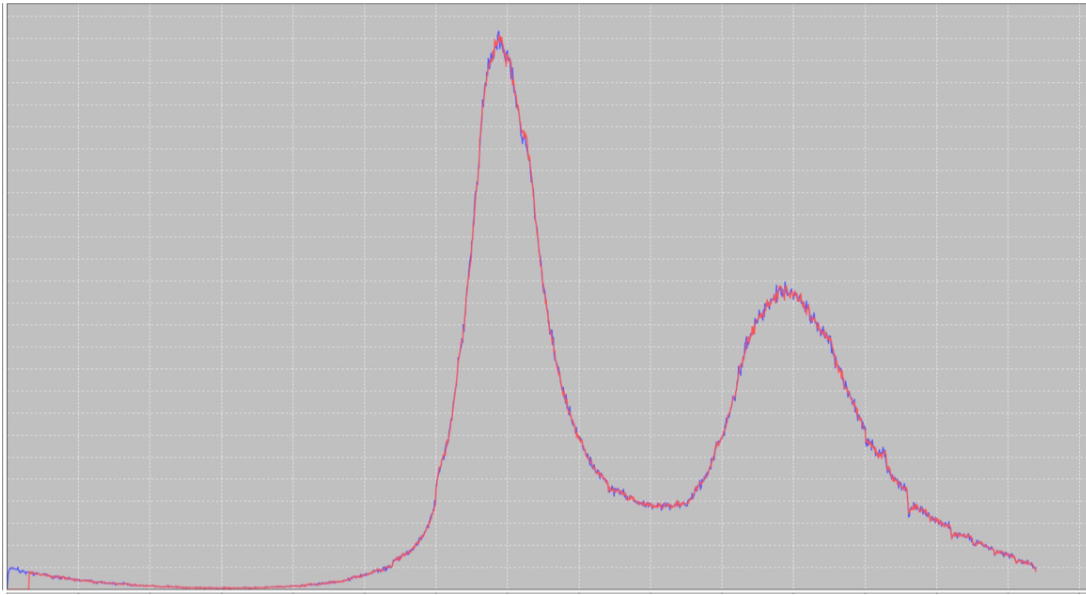


图4.3 预测和实际数据对比图

Fig 4.2 Contrast diagram of prediction and actual data

实际预测过程中，由于季节因素的计算不仅不是实时平移计算的过程，计算量极大，且涉及昨日、上周值，因此需要采用定时器进行计算，并将数据存入数据库。结合定时器后，可以避免重复计算，提升接口性能。

每分钟的实际值和预测值偏差较大时，需要进行监控报警，提示可能的异常。当数据量偏差超过一定值系统会自动报警，而偏差值的阈值根据在每周的不同日期和时间有不同的值。具体的阈值则是在Holt-Winter计算过程中，根据不同时刻实际值和预测值的偏差乘以一定的系数得到。数据异常时，错误日志会进行提示，系统监控会监控业务日志进而对运营团队进行提示。当数据偏差更大时，数据点为异常点，此数据对于后续数据的预测意义不大，因此后续的预测过程会忽略此值，避免更大的误差引入。

### 4.3.3 KPI管理模块

KPI管理模块是对关键关键指标的月、年需要达到的目标值的管理，主要分为KPI展示、KPI批量上传、KPI修改和KPI导出等模块，下面对各模块设计进行详细介绍。

（1） KPI展示：前端将KPI类型、KPI时间维度、KPI时间信息参数传给后端，后端通过Spirng boot注解自动注入到KPIController的参数中，KPIController利用注解得到KPIDao、KPIDDDao，用来管理KPI和KPIDD表的数据。KPIController调用ShowKPIHandler，并将前端传入的参数、KPIDao、KPIDDDao传给handler。定义好Protobuf数据结构，利用Protobuf根据数据结构生成需要的Area类。handler利用Protobuf生成的Area类请求到地区信息，并通过KPIDDDao得到前端指定条件下的KPI指标信息。根据地区、指标信息和KPI时间，最终得到各地区指定时间的KPI值，将结果写入ShowKPIResponse对象。Spring boot将对象自动转化为JSON对象返回给前端。此接口除了返回值，还会返回KPI类型名称和地区信息。KPI展示子模块类图，如图4.4所示，由于部分类成员和方法较多，本文只展示主要内容。

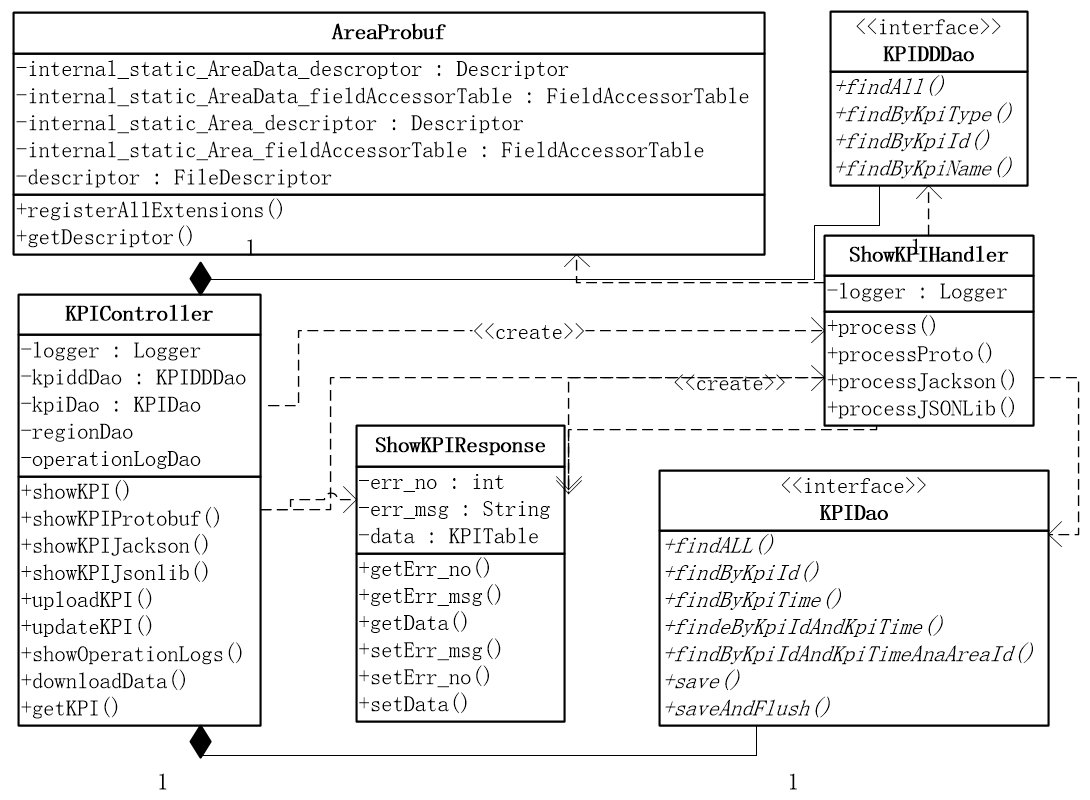


图4.4 KPI展示子模块类图

Fig 4.4 Class diagram of KPI display sub-module

（2） KPI批量导入：首先需要实现KPI模板下载功能，前端将KPI类型和KPI时间维度参数传给后端，后端通过KPIController接受请求，并调用DownloadModelHandler处理请求。DownloadModel通过Protobuf的AreaProtobuf得到地区信息，通过KPIDDDao到指定类型的KPI指标信息，然后通过poi包下的相关类进行Excel处理，将信息写入Excel中，利用输出流传给前端。KPI批量导入子模块的前端将KPI类型和KPI时间维度参数传给后端，并通过表单将文件上传，后端通过MultipartFile接受文件。KPIController接受请求后，调用UploadKPIHandler实现处理逻辑。Upload利用ExcelUtil实现Excel校验，根据KPIDDDao得到KPI的类型信息，判断值类型是否正确，判断取值范围是否合理。判断完成后，通过KPIDao查询数据库是否存在已经上传的数据，如果没有，就插入新的数据，如果有，就用KPIDao更新数据。KPI上传模块类图如图4.5所示，由于部分类成员和方法较多，本文只展示主要内容。

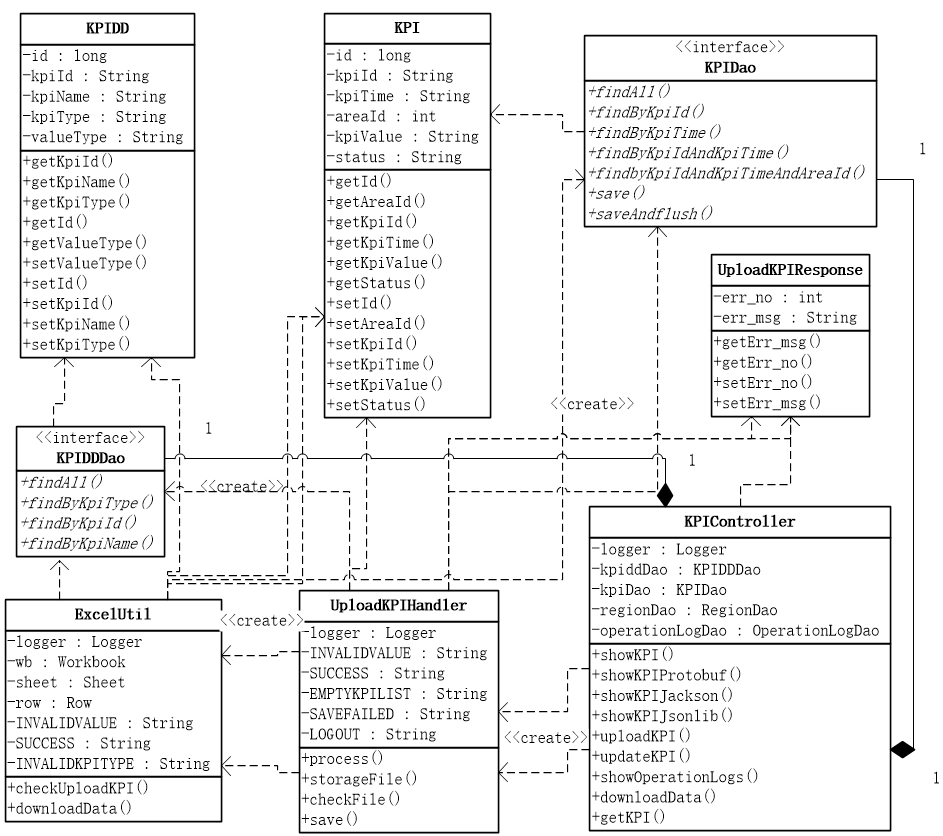


图4.4 KPI批量导入子模块类图

Fig 4.4 Class diagram of KPI upload sub-module

（3） KPI修改：在KPI展示的业务，可以进行KPI修改，选择需要修改KPI的地区。用户对需要更新的指标进行修改。UpdateKPIHandler负责此模块的业务逻辑。通过对前端传回的值校验，判断数据是否合理，然后利用KPIDao得到需要修改的数据，更新数据，保存即可。

（4） KPI下载：KPI下载和KPI模板下载模块类似，在KPI模板生成的基础上，加上KPI的值。批量获取所有指定时间的所有KPI的值，根据KPI指标信息和地区信息写入Excel即可。然后利用输出流传给前端即可。

### 4.3.4 认证和日志管理模块

认证和日志管理子模块是对系统统一认证及各种操作、导出、下载日志的统一管理。分为用户单点登录子模块、KPI日志和数据日志三个子模块，下面对三个子模块的设计进行具体介绍。

（1） 用户单点登录子模块：使用CAS实现单点登录对于普通的系统即可满足需求。但由于数据系统的大多数接口返回的都是JSON对象，当CAS重定向到CAS Serve端进行认证时会返回状态码302，这类接口无法识别302状态吗，此时需要对CAS框架进行一定的修改。单点登录模块的流程图参考图4.6。



图4.5 单点登陆流程图

Fig 4.5 Flow diagram of Single Sign On sub-module

CAS Client端利用SSOAuthenticationFilter判断用户是否认证，如果没有认证需要进行登陆，利用cas20ProxyReceivingTicketValidationFilter验证票据，利用httpServletRequestWrapperFilter获得首次登陆的用户信息。单点登录子模块在CAS框架基础上加入了LoginFilter和AddCookieFilter两个Filter。LoginFilter是否有本系统种下的用户认证相关Cookie，如果没有，会返回错误码-1，然后前端根据此错误码跳转请求login的Servlet，此Servlet不属于数据接口，可以直接进行重定向。当后续认证和票据验证通过后，使用AddCookieFilter验证是否含有用户认证相关Cookie，如果没有，则将用户信息利用RC4加密，种在Cookie中，同时将用户的工号和登陆时间等信息写入user表。

（2） KPI日志：用户对KPI数据进行修改时，日志模块记录操作的类型和数值改变，同时利用保存有用户信息的Cookie得到用户的工号，利用KPILogDao访问数据库，将修改信息保存在op\_log表中。查看Log信息时，Controller接受请求，调用OpLogHandler实现业务逻辑，将OpLogDao传给业务逻辑实现类，业务逻辑利用数据层得到操作日志信息，并将信息写入OpLogResponse中，返回给前端。

（3） 数据日志：用户导出数据后，会在data\_log中记录一条数据，数据中保存有导出时间、业务线、任务名称等信息，用户工号信息从Cookie中解密得到，此时下载时间为空字符串，当用户下载导出过的数据时，利用DataLogDao访问数据库，设置下载时间。如果不是第一次下载，则插入一条新数据，除了导出时间外，其余信息不变。由于导出日志数据量较多，因此可以通过业务线、任务名称进行筛选。

5 系统实现

基于本框架的数据管理系统已经开发完成，离线数据管理模块、实时数据管理、KPI管理和认证日志管理四个模块分别部署在四个服务上，由Protobuf实现数据交互管理，本章节将对各模块的核心实现进行介绍。

## 5.1 离线数据管理模块

离线数据管理模块包括多种数据组合和展示维度，不同的接口和展示维度需要展示的内容和数据信息不尽相同。本节将对离线数据管理部分的主要功能点实现进行介绍。

### 5.1.1 业务概览子模块

进入离线数据管理模块，系统首先展示业务概览子模块。业务概览页面分三个部分，上部展示关键指标看板，中部展示客户表现、运营效率三个板块。

（1） 关键指标看板

默认展示昨日全国各大区的数据，可以选择修改日期。地图部分展示大区位置信息，右侧饼状图展示各大区的收入和单量信息，如图5.1所示。下方表格展示个大区多个关键指标的各大区数据。

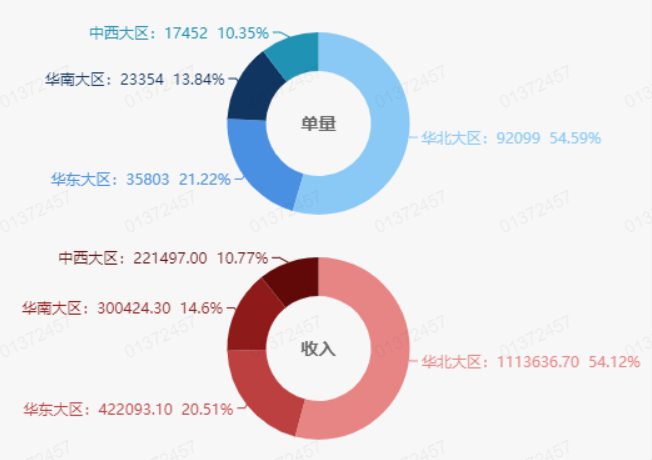


图5.1 关键指标看板界面

Fig. 5.1 Page of key index display module

表格上方可以点击更多指标按钮，然后在复选框中选择需要展示的其他指标，关键指标看板部分共涉及34个指标，指标分为订单类和配送类，表格最多同时展示8个指标。点击地图进入子维度时，右侧饼状图和下方表格联动展示子维度数据，指标和表格展示可以参考图5.2。



图5.2 关键指标看板数据和指标展示图

Fig. 5.2 Data and indicator display diagram for Key indicators display module

（2） 客户变现

客户表现默认展示昨天全国主要各行业、各品牌收入和单量占比。通过日期控件可以选择展示日期，通过标签选择收入、单量，通过地区级联菜单选择展示的地区维度。地区维度可以依次从全国到大区、地区、城市维度。行业占比数据，右侧展示品牌占比数据。两个模块上方饼状图展示了占比信息，下方表格展示占比和日环比信息。图5.3展示了某地区维度的品牌收入测试数据的占比信息图，其余图表不再展示。

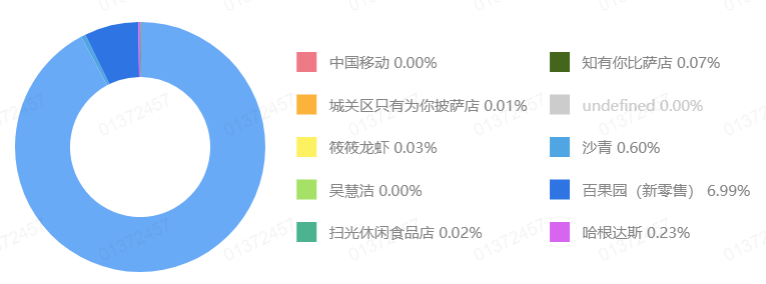


图5.3 各品牌收入占比界面

Fig. 5.3 Page of the proportion for brand income

（3） 运营效率

运营效率展示了各个维度下的运营效率和资源利用率评分。可以通过选择日期和维度选择决定展示的具体数据来源，默认展示昨日全国维度的数据。图5.4展示了全国维度的运营效率数据，图中的目标值将目标值也作为指标写入measures参数，后端处理带有KPI的measure时，利用Protobuf请求KPI管理模块得到具体数据。健康诊断根据目标值和今日值的比例进行打分，当质量指标和资源指标的达成率均在80%以上时，诊断情况为合理，质量指标和资源指标的达成率中有一项在80%以上时，诊断情况为较为合理，当两项达成率均不足80%时，为不合理。健康诊断图的的横纵坐标值由两项指标的达成率决定。

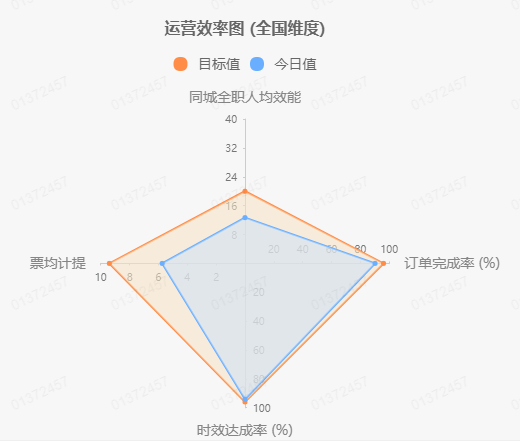


图5.4 运营效率图

Fig. 5.4 Diagram of Operation efficiency

### 5.1.2 经营主题子模块

经验主题的核心是收入和几项质量监控数据的实际值和KPI值的数据展示。

（1） 收入KPI

收入KPI模块展示和月累计收入、年累计收入相关的指标值和KPI值，可以通过日期选择查看不同日期时的达成进度。表格部分时大区的KPI，利用指示灯展示当前时间段达到的实际值和根据KPI计算当前应达到的目标进度的比较，当KPI达成100%时，指示灯为绿色，当KPI达成快于时间进度时，指示灯为橙色，否则为红色。同样将指标对应的KPI通过measure参数传给后端，后端利用Protobuf远程请求得到KPI管理模块得到目标值。下方的柱状图以地区为单位展示了累计目标值和当前时间达到的实际值，如图5.5所示。

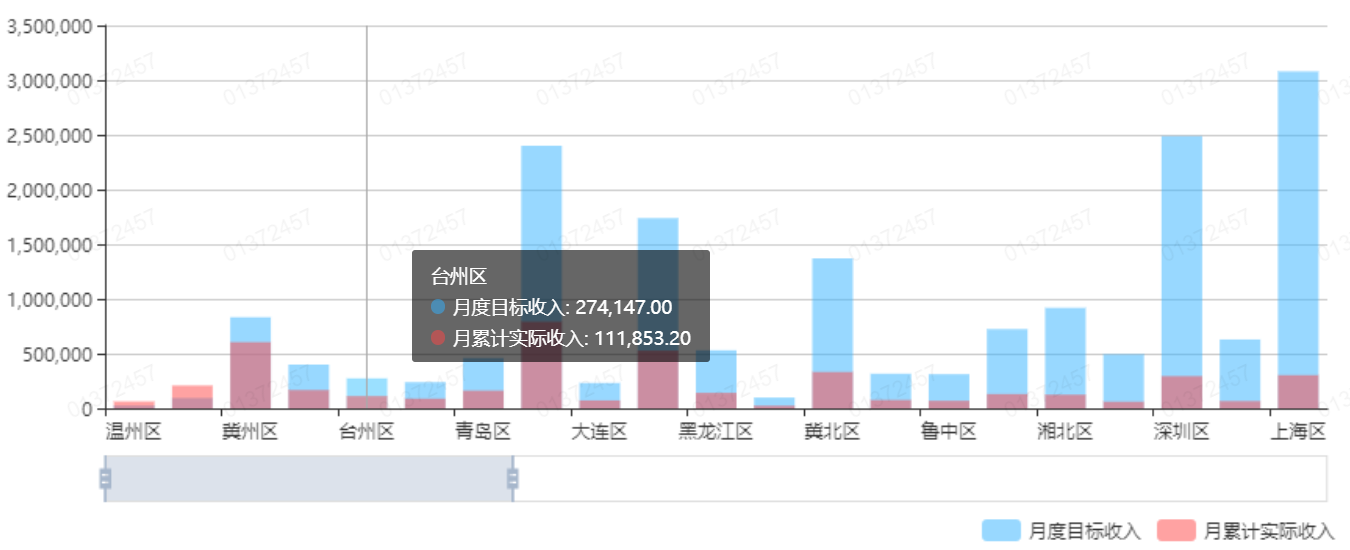


图5.5 收入KPI柱状图

Fig. 5.5 Histogram of Income KPI

（2） 质量监控KPI

质量监控KPI可以通过日期选择不同日期的质量监控相关指标的达成进度，通过指标选择具体的展示指标，然后利用柱状图展示地区实际达成率、全网平均值和目标值。目标值的取得方式和收入KPI相同，不在具体介绍。

### 5.1.3 客户主题子模块

客户主题展示客户的各项表现，核心是各区表现和客户各指标数据的变化趋势图。

（1） 各区表现

各区表现部分可以通过日期、商家品牌、指标选择展示各地区下的数据。以表格的形式展示各项信息。各区变现的实现方式和业务概览下的关键指标看板实现方式类似，具体不做详细展示。

（2） 客户关键指标趋势图

关键指标看板默认展示最近30天的数据趋势。通过起始日期、商家、地区维度和指标选择可以展示不同的数据。日期最多选择30天范围，商家可以模糊匹配进行检索，地区维度可以从全国到大区、地区、城市维度。请求到数据后以曲线和表格两种形式展现。指标趋势图的曲线展示如图5.6所示。

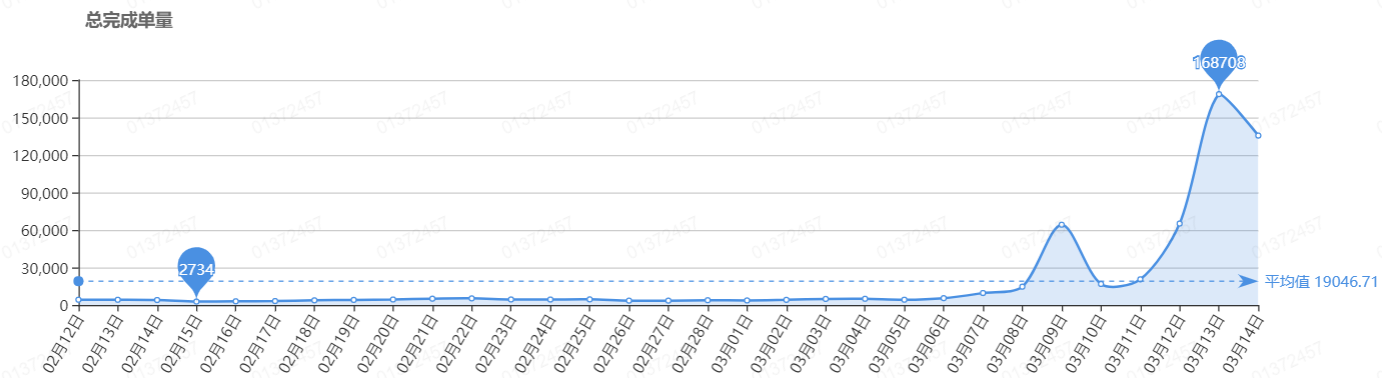


图5.6 指标趋势图

Fig. 5.6 Trend diagram of the index sign

点击数据导出后，服务会提前返回，然后后台进行数据的读取和Excel写入等工作，此时可以在任务列表查看任务列表，人物列表显示任务的完成状态和任务的详细信息，包括任务任务名称和导出时间等信息。任务列表的页面如图5.7所示。

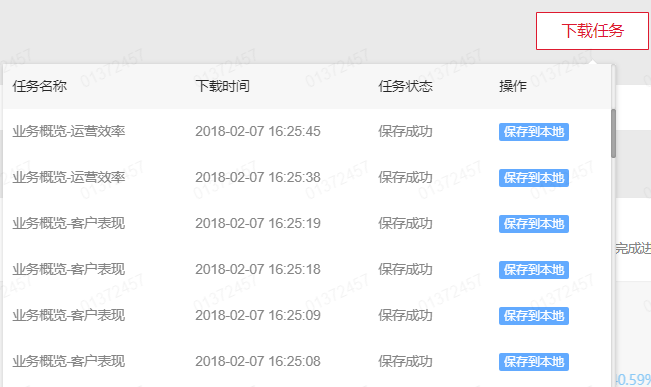


图5.7 任务列表子模块

Fig. 5.7 Page of t ask list sub-module

点击任务列表的下载连接，可以下载数据。由于任务导出后，会将表格数据存储在HDFS上，因此数据下载业务依赖HDFS，下载操作的核心代码如下：

Configuration conf = new Configuration();

conf.addResource(Thread.getResource("/").getPath() + "core-site.xml");

conf.addResource(Thread.getResource("/").getPath() + "hdfs-site.xml");

conf.set("fs.hdfs.impl", org.apache.hadoop.hdfs.DistributedFileSystem.class.getName());

FileSystem fs = FileSystem.get(URI.create(hdfspath + "/" + filename), conf, user);

FSDataInputStream in = fs.open(new Path(hdfspath + "/" + filename));

response.setContentType("application/vnd.ms-excel;charset=UTF-8");

response.setHeader("Content-Disposition", "attachment;filename=" + outputName);

OutputStream os = response.getOutputStream();

IOUtils.copyBytes(in, os, 4096, true);

## 5.2 实时数据管理模块

实时数据管理模块通过监控概览和实时趋势图展示系统数据，通过业务日志监控实现预测和报警。实时数据管理的地区维度分为全国和城市，在城市界面，可以利用城市名称模糊匹配，得到城市各模块数据。本节将对实时数据管理的主要功能实现进行介绍。

### 5.2.1 监控概览子模块

监控概览子模块可以从全国或城市维度查看指标信息。可以在业务概览模块查看数据指标信息，还可以在运力监控模块查看运力指标信息。

（1） 业务概览

点击更多指标，可以在复选框中选择需要查看的指标，最多查看8个指标信息。鼠标在停留在指标名称上时，会有浮层显示指标定义。指标定义、展示模块和默认位置信息通过指标接口可以获得。业务概览展示指标的值和日同比、周同比数据。业务概览模块实现如图5.8所示。



图5.8 业务概览子模块界面

Fig. 5.8 Page of business overview sub-module

（2） 运力监控

运力监控模块在可以查看全国范围内运力最紧急的城市数据。左侧地图的不同着色可以看出紧急程度，右侧表格可以查看各城市具体数据。图5.9展示了各城市人均未完成单量的实时信息。

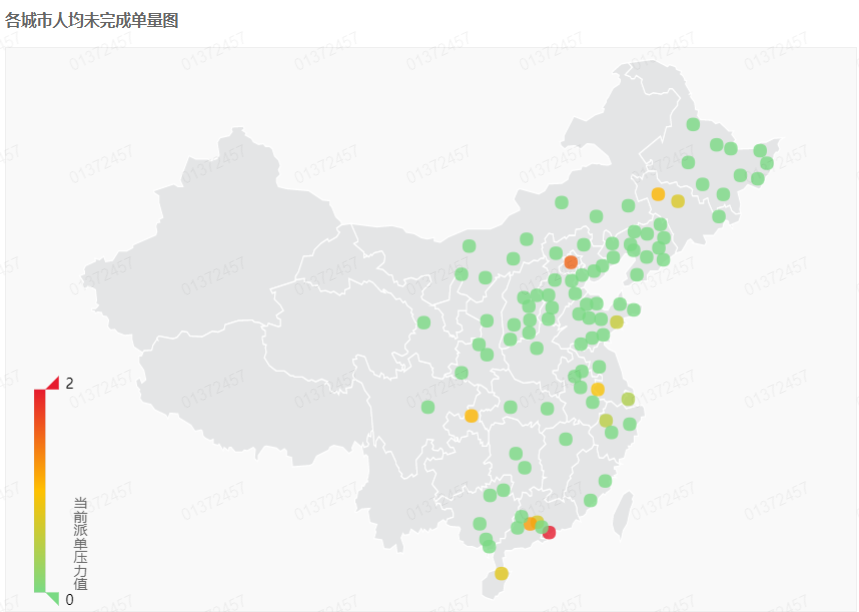


图5.9 城市运力监控界面

Fig. 5.9 Page of capacity monitoring sub-module

### 5.2.2 实时趋势图子模块

实时趋势图的指标信息可以通过指标获取接口得到，除了指标的名称，还有指标的定义、类型、默认展示位置等信息。实时趋势图中，通过更多指标可以选择展示最多8个指标的数据。批量请求所需指标今日、昨日、上周同日3天的数据量，每个指标有4320个点。当日未来的时间点的数据，由于还未得到，服务会自动进行补0操作。实时趋势图的信息参考图5.10。

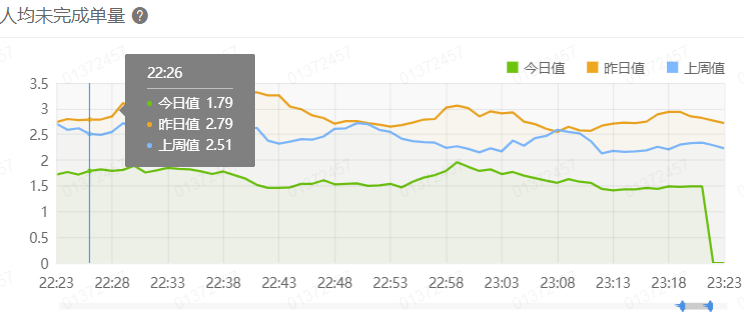


图5.10 实时趋势图界面

Fig. 5.10 Page of real-time tendency chart sub-module

### 5.2.3 预测和报警器子模块

预测器通过Holt-Winters算法实现，基于业务数据的特点，对季节因素的计算方式做了修改，将季节引入的输入参数改为了两个。报警器的计算根据日期和时间特点有不同的阈值比例，发现异常后，在服务日志输出报警信息，运维监控会将报警信息发送给相关人员，

## 5.3 KPI管理模块

进入KPI管理界面，默认进入展示界面。用户可以通过标签选择需要展示或操作的KPI的指标类型和时间类型。

### 5.2.1 KPI展示

KPI展示默认展示业务目标各地区的当月KPI值，通过时间、类型选择可以查看其他月、年、类型的KPI值。KPI数据通过分页表格的形式展示，每页显示10条数据，具体实现参考图5.11。



图5.11 KPI展示界面

Fig. 5.11 Page of KPI display sub-module

### 5.2.2 KPI批量上传

在KPI展示界面点击批量上传可以弹出KPI批量上传窗口，点击模板下载，可以下载此类型下的KPI模板，上传文件后，后台对文件进行检查，校验文件完整性和数据完整性，KPI批量上传界面如图5.12所示。



图5.12 KPI批量上传界面

Fig. 5.12 Page of KPI upload sub-module

### 5.2.3 KPI修改

在KPI展示界面点击修改单个地区KPI可以弹出KPI修改窗口，选择需要修改的地区，在需要修改的指标后的输入框输入新的值即可，点击确认即可。修改请求出发后，服务端对上传数据的值进行校验，校验数值类型和合理性。修改过的数据KPI修改界面如图5.13所示。

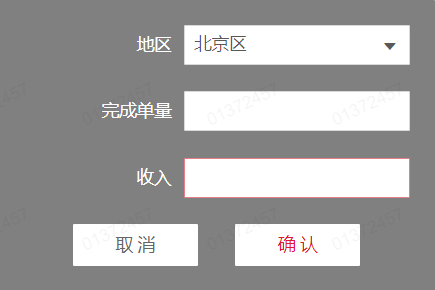


图5.13 KPI修改界面

Fig. 5.13 Page of KPI update sub-module

### 5.2.4 KPI下载

在KPI展示界面点击导出Excel可以下载KPI数据，请求出发后，后台根据选择的类型和时间生成Excel文件，然后以数据流的形式传给前端。下载后的Excel如图5.14所示。

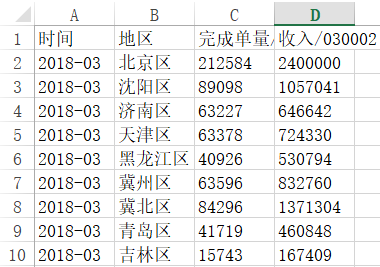


图5.14 KPI下载界面

Fig. 5.14 Page of KPI download sub-module

## 5.4 认证和日志管理模块

用户从系统首页进入任一服务，都会调用认证和日志管理模块，对数据的任何导出、修改、下载操作，也都会涉及认证和日志管理模块。

### 5.2.1 用户单点登录认证

用户单点登录认证模块由五个Filter实现，所以每次服务登陆，系统都能得到用户的工号，进入系统时，前端会请求服务得到登陆状态，从而得到用户的工号信息。利用AddCookieFilter可以保证任何请求到达Controller或Servlet处理逻辑时，系统都已经种过了Cookie，因此只需要将加密过的Cookie解密即可得到用户信息。点击系统右上方的头像，会显示用户的工号信息。如图5.15所示。



图5.15 用户信息展示界面

Fig. 5.15 Page of user information display sub-module

### 5.2.2 KPI日志

用户批量上传KPI或修改KPI模块时，一旦操作成功，都会利用Protobuf请求认证和日志管理模块在数据库插入一条KPI操作数据，写入操作类型、KPI时间类型、操作的具体内容、操作时间和操作人工号等信息。点击KPI操作日志，可以查看KPI操作日志，具体页面如图5.16所示。



图5.16 KPI日志展示界面

Fig. 5.16 Page of KPI log display sub-module

### 5.2.2 数据日志

进入数据日志模块，分页展示所有数据日志。可以利用业务线、任务名称、导出日期、员工编号进行筛选。默认展示所有业务线、全部任务、所有日期、所有员工的导出和下载日期。点击查询即可显示筛选后的数据，点击返回可以退出此页面。数据日志的界面如图5.17所示。



图5.17 数据日志展示界面

Fig. 5.17 Page of data log display sub-module

6 系统测试

软件测试是一个检验计算机软件的整体质量、正确性以及完成性的过程。软件测试能够保证系统满足预定的需求，保证软件运行能够得到预期的结果，是软件开发中一个必不可少的步骤。通常来讲，所有模块的基本功能都需要包含在测试用例中，并且需要详细的测试每一个分支，这是保证系统能够在任何情况下都能够完美运行的必要条件。

## 6.1 测试理论

近年来，计算机技术在飞速的发展，使得保证软件的可靠性具有越来越重要的意义，因为大多数时候只要软件系统出现一点小错误，就会引起难以估量的损失。所以为了保证系统能够在线上正常运行，需要在上线前针对软件进行系统测试，发现软件系统中可能存在的隐患，进而加以修复，以最大幅度降低由于软件系统漏洞造成损失的风险。

随着计算机技术的不断发展，测试理论也在不断进步，当前在实际的软件开发过程中主要有三种测试方法，包括黑盒测试、灰盒测试和白盒测试。黑盒测试针对软件的功能进行测试，这种方法不需要关心代码的内部逻辑，也不需要了解软件API接口如何定义，只需要针对软件的整体功能进行测试即可，所以这种方法主要在软件确认的过程中使用。白盒测试就是将软件系统当成一个透明的盒子，无论是代码的内部逻辑还是软件的系统结构对测试人员来说都是可见的。白盒测试依然要从系统功能的角度出发，保证系统功能是软件测试的基本目的。白盒测试需要测试人员对系统有足够多的了解，需要将所有分支涵盖到测试用例中，所以开发人员在开发完软件系统之后的自测过程中大部分会采取白盒测试方法。灰盒测试是处于白盒和黑盒测试之间的一种测试方法。灰盒测试主要关注的是软件系统在一定条件下是否能够产出准确的输出，更关注极端用例，因此灰盒测试对测试人员的测试专业度有很高的要求。

## 6.2 系统典型模块功能测试

测试用例是针对一些特定目标设定的，需要根据具体目标指定相关的输入条件和执行方法，以及对应的预想结果，然后从测试用例的运行结果进行分析，来验证系统是否已经满足预期的要求。由于论文的篇幅限制，本文只包含了核心模块的功能测试。

### 6.2.1 离线数据管理模块

离线数据管理模块的功能包括业务概览、经营主题、客户主题三个模块，对应的测试用例包括数值准确性验证，默认展示指标信息、默认展示指标位置、图表显示信息完整性、各地区维度切换、数据表格排序等内容。离线数据管理模块的主要测试用例如表6.1所示。

表6.1 离线数据管理模块的测试用例

Tab. 6.1 Test cases of offline data management module

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 操作 | 预期结果 | 测试结果 |
| 1 | 查看业务概览-关键指标看板模块图表 | 默认显示昨日数据、默认显示完成单量等6项指标信息 | 通过 |
| 2 | 续1，选择更多指标 | 正常显示指标和数据，最多可选择8项指标信息 | 通过 |
| 3 | 续2，选择其他日期 | 正常显示所选日期指标和数据 | 通过 |
| 4 | 查看客户主题-客户表现模块图表 | 正常显示全部商家默认的6项指标和数据 | 通过 |
| 5 | 续4，切换地区维度到大区、地区、城市 | 正常显示具体地区维度下的指标和数据 | 通过 |
| 6 | 查看客户主题-客户关键指标趋势图模块图表 | 正常显示过去30天，所有商家全国维度的默认5个指标的数据 | 通过 |
| 7 | 续6，依次切换商家、地区、指标 | 正常显示所商家、地区维度、指定指标的数据信息 | 通过 |
| 8 | 续7，选择起始、终止日期 | 正常显示所选日期数据，最多查看30天信息 | 通过 |
| 9 | 导出客户主题-客户关键指标趋势图模块数据 | 任务列表可以看到导出成功，并提供下载链接，下载正常 | 通过 |

表6.1展示的是测试用例中的主要部分，更多细节测试用例不再详细介绍。从表中可以看出，针对离线管理模块的测试基本覆盖了离线数据管理模块的核心功能，经营主题下的各个模块的数据展示和维度筛选和业务概览、客户主题下的相关测试类似，不在详细介绍。除了页面展示信息的完备性，还根据业务系统数据进行了数据准确性验证。通过各项测试，证明了模块功能满足需求。

### 6.2.2 实时数据管理模块

实时数据管理模块包括监控概览、实时趋势图、预测和报警器三个子模块。对应的测试用例包括数据准确性验证、默认展示指标信息，默认展示指标位置、全国城市数据切换、更多指标数据展示、报警器响应性等功能测试。实时数据管理模块的主要测试用例如表6.2所示。

表6.2 实时数据管理模块的测试用例

Tab. 6.2 Test cases of real-time data management module

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 操作 | 预期结果 | 测试结果 |
| 1 | 查看监控概览-业务概览模块 | 默认显示当前分钟数减2时刻分钟的资源在线率等4项指标的数据信息 | 通过 |
| 2 | 续1，选择更多指标 | 正常显示指标和数据，最多可选择8项指标信息 | 通过 |
| 3 | 查看监控概览-运力监控模块 | 正常显示地图和表格数据，表格数据、排序正常 | 通过 |
| 4 | 续3，点击地图上的城市 | 进入城市的数据展示界面 | 通过 |
| 5 | 查看实时趋势图模块 | 正常显示实时进单量等4项指标和数据 | 通过 |
| 6 | 续5，依次切换商家、地区、指标 | 正常显示所商家、地区维度、指定指标的数据信息 | 通过 |
| 7 | 向数据库插入一条试试进单量为2000的数据 | 收到报警信息，提示数据异常 | 通过 |

表6.2展示的是实时数据管理模块测试用例中的主要部分，更多细节测试用例不再详细介绍。从表中可以看出，针对实时数据管理模块的测试基本覆盖了此模块的核心功能。通过各项测试，证明了模块功能满足需求。

### 6.2.3 KPI管理模块

KPI管理模块包括KPI展示、KPI批量导入、KPI修改、KPI下载等功能。对应的测试用例包括数据校验验证、指标类型校验、下载数据格式检测等。KPI管理模块的主要测试用例如报表6.3所示。

表6.3 KPI管理模块的测试用例

Tab. 6.2 Test cases of KPI management module

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 操作 | 预期结果 | 测试结果 |
| 1 | 查看本月业务目标、月度KPI数据 | 正常显示各地区完成单量和收入两项指标数据 | 通过 |
| 2 | 从批量上传模块下载KPI模板 | 模板格式和表头、地区信息正常 | 通过 |
| 3 | 续2，上传业务目标、年度KPI数据 | 经校验，KPI类型错误，上传失败 | 通过 |
| 4 | 续3，在整数类数据中上传小数 | 经校验，值类型错误，上传失败 | 通过 |
| 6 | 修改北京区质量监控指标KPI值 | 修改成功，正确显示导入的数据 | 通过 |
| 7 | 导出历史上传数据 | 导出正常，Excel格式数值正确 | 通过 |

表6.3展示的是KPI管理模块测试用例中的主要部分，更多细节测试用例不再详细介绍。可以看出，KPI和用户的各种交互功能全部通过测试，模块功能满足用户需求。

### 6.2.4 认证和日志管理模块

认证和日志管理模块包括用户单点登录认证、KPI日志、数据日志等功能。对应的测试用例包括多服务登陆认证、KPI日志查看、数据日志查看、筛选等。认证和日志管理模块的主要测试用例如报表6.4所示。

表6.4 认证和日志管理模块的测试用例

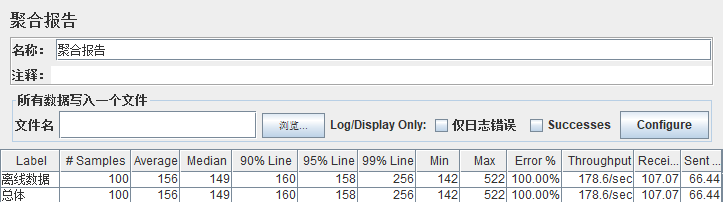
Tab. 6.4 Test cases of authentication and log management module

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 操作 | 预期结果 | 测试结果 |
| 1 | 分别从多个窗口点击多个服务 | 只有首次打开系统需要登陆 | 通过 |
| 2 | 查看KPI历史日志 | 展示操作时间等相关信息 | 通过 |
| 3 | 查看数据日志 | 正确显示所有数据导出、下载日志 | 通过 |
| 4 | 续3，通过工号筛选 | 只显示筛选工号对应的数据日志 | 通过 |

认证和日志管理模块的测试结果显示，此模块的功能正常，不论是单点登录还是各种日志显示、筛选，都满足用户日常需求。

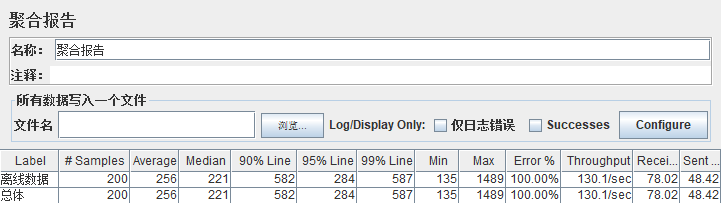
## 6.3 性能测试

通过观察tomcat的access日志可以发现，系统各服务接口响应时间基本都在200ms以内，各接口中，离线数据的接口由于和Kylin交互相对较慢，因此对此接口进行了压测。使用压测工具Jmeter模拟了100、150和200线程情况下，离线数据查询接口的响应情况。100线程同时并发的响应结果的聚合报告如图6.1所示，200线程同时请求的结果的聚合报告如图6.2所示。



6.1 模拟 200 用户并发下性能测试聚合报告

Fig. 6.1 Aggregation report of performance testing by simulating 200 users



6.2 模拟 200 用户并发下性能测试聚合报告

Fig. 6.2 Aggregation report of performance testing by simulating 200 users

从聚合报告来看，当模拟100个用户并发访问时，平均响应时间仅达到156ms，90%的用户能够在160ms内收到返回，当模拟用户换成200的时候，平均响应时间变成256ms，速度有所下降，90%的用户能够在582ms内收到返回。当模拟100个用户进行并发访问时，接口相应的中位数为156ms，这足以证明接口能够非常快的响应请求。相对来说接口出错率较低，在0.05%以内，这证明系统编码可靠性达到了要求。当150个用户模拟连续并发请求系统时，最大吞吐量TPS能够达到426/s。根据实际情况，数据系统的使用者为管理和运营人员，因此对并发量的需求并不高，当前压测的并发量完全满足需求。对系统的性能需求方面，结合access日志和压测的聚合报告，可以看出，框架独特的架构方式，使得系统的响应速度非常快，性能完全满足用户需求。

结 论

本文设计并实现了一个基于微服务的数据管理框架，该框架根据企业的业务数据来源的多样和综合查询的复杂性基础上，设计除了一套大数据的综合管理框架。该框架在各类型企业的数据管理都能起到重要作用。

本框架利用Spring boot和Protobuf设计了一款对离线、实时数据进行综合管理的微服务架构框架模型。通过Hadoop下的HDFS、HBase、Sqoop和Hive工具，搭建了一套数据处理的环境，结合企业业务数据和数据处理环境，设计了一套数据预处理和存储的可用流程。框架利用Apache Kylin的Fast Cubing算法对离线数据进行预计算，实现了对离线数据的多维度、多数据管理，以满足用户多层次、多维度、多指标、多计算的查询需求。本框架还利用了Quartz定时器和Jparepository对实时数据进行了预处理，结合Holt-Winter和业务数据的基本信息，对特殊指标进行预测和监控，最终实现了对实时数据的综合查询和管理。此外，框架还基于MySql和POI框架实现了对于KPI的管理和使用，基于CAS框架实现了认证和日志管理服务。本框架的架构和数据处理流程简化了数据系统的开发流程，为高性能数据系统的开发和实现提供了可靠支持。

但由于数据的敏感性，数据系统需要增加对于权限的管理控制，因此在未来的开发过程中，需要为框架增加用户组和用户权限控制服务，进一步与增加系统的安全性和可靠性。

参 考 文 献

[1] 李学龙,龚海刚.大数据系统综述[J].中国科学:信息科学,2015, 45(1):1-44.

[2] 陶雪娇,胡晓峰,刘洋.大数据研究综述[J].系统仿真学报,2013(s1):142-146.

[3] 程学旗,靳小龙,王元卓,等.大数据系统和分析技术综述[J].软件学报,2014(9):1889- 1908.

[4] 许静.多级存储技术及光盘库在实际工程中的应用探讨[J].智能建筑电气技术，2011,5(3):

97-100.

[5] 姜晓青,王钦若.大数据环境下冷数据存储技术概述[J].工业控制计算机,2016,29(6):58- 60.

[6] 马晓铭,马维华.光盘库嵌入式主控系统[J].计算机系统应用,2011,20(6):21-24.

[7] 赵伟东.电子档案蓝光存储应用探究[J].档案学研究,2015,6(3):88-95.

[8] 林子明.蓝光光盘库缓存管理机制的研究与实现[D].广州:中山大学,2013.

[9] 王永和,张劲松,邓安明,等.Spring Boot研究和应用[J].信息通信,2016(10):91-94.

[10] 杨家炜.基于Spring Boot的web设计与实现[J].轻工科技,2016(7):86-89.

[11] 张峰.应用SpringBoot改变web应用开发模式[J].科技创新与应用,2017(23):193-194.

[12] Gutierrez F. Spring Boot, Simplifying Everything [J]. 2014.

[13] 聂晓旭,于凤芹,钦道理.基于Protobuf的数据传输协议[J].计算机系统应用,2015,24(8): 112-116.

[14] Zhao W T, Ding Y, Zhang X H. Available Storage Space Sensitive Replica Placement Strategy of HDFS [J]. Applied Mechanics & Materials, 2014, 644-650:3224-3229.

[15] Yu D, Dou W, Zhu Z, et al. Materialized View Selection Based on Adaptive Genetic Algorithm and Its Implementation with Apache Hive [J]. International Journal of Computational Intelligence Systems, 2015, 8(6):1091-1102.

[16] 杨彬.Sqoop数据收集与入库系统的应用[J].电子制作,2017(21).

[17] 于金良,朱志祥,梁小江.一种基于Sqoop的数据交换系统[J].物联网技术,2016,6(3):35- 37.69.

[18] Xu J W, Liang J L. Research on a Distributed Storage Application with HBase [J]. Advanced Materials Research, 2013, 631-632:1265-1269.

[19] 林海,康宝中.基于Holt-Winters时间序列的图书选题预测模型[J].数字技术与应用, 2017(3):51-53.

[20] Dantas T M, Oliveira F L C, Repolho H M V. Air transportation demand forecast through Bagging Holt Winters methods [J]. Journal of Air Transport Management, 2017, 59:116-123.

[21] 陈慧,龚婷雨.大数据分析与Apache-Kylin应用[J].江西通信科技,2016(4):26-29.

[22] Li Y. Apache Kylin from eBay: Extreme OLAP engine for Hadoop - O'Reilly Media Free, Live Events [J]. 2017.

[23] 李建佳,王晶.基于JA-SIG CAS统一认证平台(SSO)的设计与实现[J].广东海洋大学学报,2013,33(3):78-83.[23]姜晓青.蓝光光盘库文件管理系统设计与实现[D].广州:广东工业大学,2016.

致 谢

时间过的很快，仿佛就在昨日，才收到保研通知，现在两年的研究生生涯就将要结束了。我也要和一起生活、学习、成长了六年的软件学院道别了，即使有万般不舍，我们也依然要继续前行，去谱写我们人生中新的篇章。

很荣幸这几年在软件学院，能够结识如此多优秀的老师和同学。在遇到困难时，能得到老师无微不至的关怀和悉心的指导。在生活和学习中，能和同学们一起互帮互助，相互鼓励着共同成长。在完成毕业设计的过程中，我也得到了老师和同学们热心的帮助。在这里要诚挚的感谢关心我、帮助我的师长和各位亲人以及朋友们，保证了我可以顺利的完成系统设计、需求开发和论文撰写。我要重点感谢我的指导老师，是赖晓晨老师对待学术无比严谨的态度，以及他对待工作废寝忘食、完美主义的精神时刻影响着我，他的自律、严格、耐心、细致的态度也一直是我学习的典范。每当我懈怠的时候，赖老师总是能够以身作则提醒我保持端正严谨的态度，每当我失落的时候，赖老师都能够耐心的鼓励我。再次感谢赖老师的支持和帮助，引领我少走弯路，快速成长。另外，还要感谢实验室的学长和同学们。很高兴能够和大家一起度过这段美好的时光，也十分感激大家一直以来的支持和帮助，这也将是我在大连理工大学软件学院最美好的一段回忆。最后，还要真心感谢答辩委员会的各位老师，你们专业的指导和严谨的态度，让我深受感染，很高兴能够顺利的完成毕业设计，我也在这次毕业设计的完成过程的完成过程中受益匪浅。

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：

作 者 签 名 ： 日期： 年 月 日

导 师 签 名 ： 日期： 年 月 日