南京大学

本 科 毕 业 设 计

院　系 软件学院

专　业 软件工程

题　目 基于自定义 SQL 的报表设计与实现

年　级 2011 级　学　号 111250109

学生姓名 陆蕾

指导教师 郑滔　职　称 教授

论文提交日期 2015/05/04

南京大学本科生毕业论文（设计）中文摘要

毕业论文题目： 基于自定义 SQL 的报表设计与实现

软件学院 院系 软件工程 专业 2011 级本科生姓名： 陆蕾

指导教师（姓名、职称）： 郑滔 教授

摘要：

人的社会活动会产生巨大的数据量，随着网络时代的到来，该数据量呈指数级爆炸性增长。但如此之多的杂乱无章的数据并不代表能完全发挥作用，只有将庞大的数据经过目的性的组织，规范化的展现，数据中蕴藏的含义才能被发掘。传统报表已无法适应当今数据的数据量之大，变更速度之快，及时性要求之高以及用户对报表的美观性、简便性、易操作性的高要求。

SQL 语言作为当今最为流行的关系数据库操作语言之一，具有庞大的用户群，它简单易上手，具有易学易用的特性。

本文以百度广告数据交易量的展示平台为例，重点阐述了基于自定义 SQL 的报表实现：如何通过用户对用于展示的 SQL 语言的自定义配置，将每日产生的巨大数据量，以图表的形式通过报表展现出来，并帮助用户分析数据分布，数据变化，数据走势等业务内容。同时，该报表实现中的设计并非只适用于百度公司的特定场景，对于其他企业级报表而言，该设计同样具有普适性及参考价值。

关键词： SQL、自定义报表、Echarts

南京大学本科生毕业论文（设计）英文摘要

THESIS: Implementation of Reporting System Based on User-defined SQL

DEPARTMENT: Software Institute

SPECIALIZATION: Software Engineering

UNDERGRADUATE: 2011

MENTOR: Tao Zheng

Abstract:

With the arrival of the Internet, the amount of data generated by human social activities is growing with an exponential and explosive speed. However, people cannot make full use of these data, only if the data is organized in purpose and displayed standardly. The traditional reporting system cannot satisfy the amount of the data, the speed of data change, and the demand of timelineless. They can either satisfy the high demand of aesthetics，simplicity and operability of system from current users.

As one of the most popular structured query language of relational database, SQL has massive user bases, since it is simple and easy to use.

In this document, I will introduce how to build a reporting system based on the SQL defined by user him/herself, with the system designed for Baidu's ad exchange as an example. It illustrates how to display the data by a report in the form of table and chart based on the user-defined SQL, and how to help them analyze the business content, such as the distribution of data, the change of data, and the trend of data and so on.

Keywords: SQL, reporting system, Echarts

目 录

图目录

表目录

第一章 引言

1.1 项目背景

作为百度公司盈利的重要来源之一，百度业务端展示广告每天会产生大量数据，这些数据反映了百度业务端广告的重要信息，它体现了广告策略的有效性，并可用于对广告反作弊的监控，还可用于分析不同广告产品线的收益情况等业务信息。

报表是管理业务的数据信息的重要呈现方式之一，它能够清楚明确地动态展现数据信息，并且格式美观多样，符合众多分析人员的需求。因此，产品经理需对展示广告周度数据进行业务监控，并制作出不同产品线的周报以作为汇报和总结材料。

在此项目开发之前，产品经理每次制作周报的流程非常复杂。首先需将数据从数据库导入 Excel 表格中，然后在表格中对所有数据进行分析汇总，并根据业务需要制作出相应的图表；再新建 PPT，将之前制作的图表进行复制粘贴，作为展示材料。

同时，由于展示广告业务是在不断发展变化、非常灵活的，产品线的划分可能产生变化，在发展过程中没有完全固定的模式，产品经理之前在 Excel 中制作图表时使用的公式在业务变更之后可能无法使用。产品经理寻求一个系统能够减轻制作图表及 PPT 的负担，而将精力更多地集中于对图表结果的业务分析。

再者，展示广告每天都会产生新的数据，数据处于一个不断变化的阶段，产品经理希望有一个平台能够动态地根据数据展现图表，使得得到的图表具有实时性。因此需要一个自定义报表系统支持业务的动态变化，通过该系统自动输出报表。

1.2 项目目标

本自定义报表系统在使用者熟悉基本 SQL 语法的前提下，旨在通过使用者自助配置数据导入 SQL、展现 SQL，实现数据存储、业务逻辑组合方式、展示内容、关联功能等方面的灵活配置，以适应业务新增、减少、重组等的不断变化,帮助使用者既灵活又高效地实现报表的输出。

1.3 现有报表系统分析与比较

1.3.1 Excel

Microsoft Excel 是 Microsoft 为使用 Windows 和 Apple Macintosh 操作系统的电脑编写的一款电子表格软件。直观的界面、出色的计算功能和图表工具，再加上成功的市场营销，使 Excel 成为最流行的个人计算机数据处理软件。在 1993 年，作为 Microsoft Office 的组件发布了 5.0 版之后，Excel 就开始成为所适用操作平台上的电子制表软件的霸主。

Excel 支持 Visual Basic for Applications（VBA）。VBA 是一款功能强大的工具，它使 Excel 形成了独立的编程环境。使用 VBA 和宏，可以把手工步骤自动化，VBA 也允许创建窗体来获得用户输入的信息。但是，VBA 的自动化功能也导致 Excel 成为宏病毒的攻击目标。[1]

使用 Excel 的好处为，用户对该软件的熟识度较高，学习成本低，用户能够轻松地将数据处理为报表。但是，如项目背景中所描述的，Excel 无法适应用户对变化性及灵活性的高要求。

1.3.2 其他商业报表

如今比较成熟的商业报表有很多，诸如 Smartbi，SAP Crystal Reports Server，FineReport（帆软）等。这些商业报表都具有很长的发展时间，逐渐变得成熟，且报表的格式多样美观。但是，从产品经理的需求出发，此类报表具有以下欠缺：

1. 成本代价高。 商业报表的价格一般都很高，购买一套成熟的商业报表需要很大的成本。
2. 适应性差。 由于商业报表是从产品的普适性出发，制作的报表系统以客户的统一需求为目标，而对于公司本身的特殊需求的适应性差。购买之后如果进行改造，人力成本也很大。
3. 安全隐患。 由于本系统所处理的数据为百度内部的商业数据，具有很高的商业参考价值。外有系统无法保证其自身的安全性，一旦数据泄露，后果不堪设想。

1.4 本文工作

本文介绍了基于自定义 SQL 实现的报表系统的相关理论知识，报表数据的获取流程，报表配置的过程与实现，报表控件的展示流程与实现。在该系统设计实现的过程中，我负责了报表的配置功能及展现功能，在本文中，主要对这两个方面进行阐述。由于该报表系统是在自定义 SQL 的基础上运行的，涉及的操作多为对数据库的操作，因此数据库中表结构的设计尤为重要，在本文中会着重对表结构设计进行描述。

1.5 本文结构

第一章为概述和前言部分，主要介绍了项目背景，描述了该论文的结构及各章所描述的内容。

第二章中介绍报表系统中使用的相关技术。

第三章中对报表系统进行需求分析，整体概述该报表系统，并从不同角度描述该系统的需求和概要设计。

第四章从设计角度描述系统，包括整体的设计框架，业务架构及技术架构等，项目的包设计、接口设计以及不同模块的详细设计。

第五章总结该项目已实现的功能，寻求已有功能的突破，探讨项目的缺点和不足，并指出该项目未来的发展方向。

第六章对在论文撰写过程中提供过帮助及指导的导师和同事致以谢意。

第二章 技术概述

2.1 Spring Framework

Spring Framework 是一个构建企业就绪应用的一站式的轻量级解决方案。它的模块化使得编程人员可以选择自己所需的部分。例如，你可以选择只使用 IoC 容器，加上其他任意 web 框架，同时你也可以只选择使用 Hibernate 集成代码或者是 JDBC 抽象数据层。Spring Framework 还支持声明式事务，通过 RMI 或 web 服务远程连接，提供存储数据的多种选择。它还提供了完整的 MVC 框架，支持在程序中实现透明地集成 AOP（面向切面编程）。

Spring 的设计是非侵入式的，这意味着你的领域逻辑代码大多是不依赖于框架本身的。在集成层（例如数据访问层），某些对数据访问技术以及 Spring 库的依赖是存在的。但是，将这些依赖于其他的基本代码剥离是非常简单的。

因此，Spring Framework 是一个能够为 Java 应用提供全面架构支持的 Java 平台，致力于帮助程序员专注于应用本身而非架构。

2.1.1 依赖注入和控制反转

无论是嵌入式应用还是服务端企业级应用，Java 程序总是由多个对象共同协作所实现的。因此，应用中的各个对象是相互依赖的。

尽管 Java 平台提供了大量的应用开发功能，它缺少了将基本的构建模块聚合成一致整体的方法，而是将这些问题抛给了开发人员。尽管存在诸如 Factory，Abstract Factory,Decorator,Builder,Service Locator 之类的生成多个类和对象实例的设计模式，但是这些设计模式只告知了开发人员模式的名字，模式做了什么，该在哪里运用，它所能解决的问题，在程序中的实现依然要靠开发人员自身。

Spring Framework 中的控制反转（IoC）组件解决了这个问题，它提供了一个将不同组件组合成协同工作供使用的整体的正式方法。它将已形成的设计模式整理成了一流的对象供开发人员集成至自己的项目中。多个组织和机构以这种方式来使用 Spring Framework，致力于构建健壮且持久的应用。

2.1.2 模块

Spring Framework 由 20 个各具特征的模块组成，这些模块被划分为 Core Container（核心容器），Data Access/Integeration（数据访问/集成），Web, AOP(Aspect Oriented Programming 面向切面编程), Instrumentation, Messaging（消息机制）以及 Test 这几个大模块（如图 2.1 所示）[2]

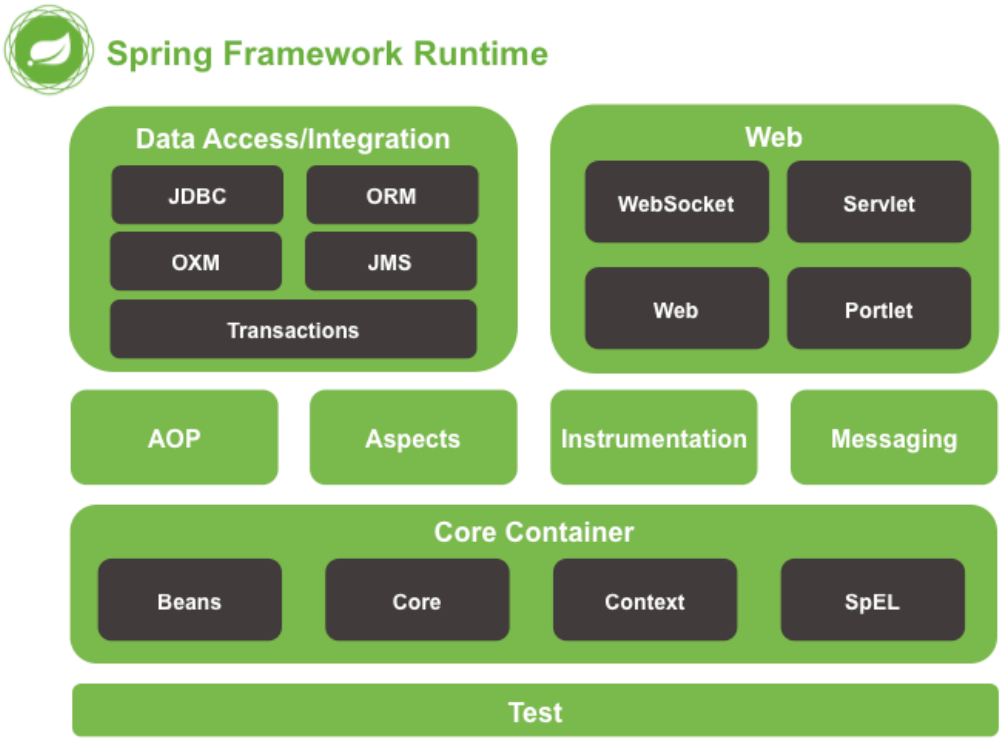

图 2.1 Spring Framework 模块图

2.2 Spring MVC

Spring MVC 是 Spring Framework 为展示层（尤其是基于 web 的展示层）提供的支持，同时还能够支持 web 应用中的 WebSocket 格式的消息机制。

如图 2.2 所示，Spring MVC 的设计是围绕 DispatcherServlet 展开的，DispatcherServlet 通过可配置的 handler mapping，view resolution，locale，time zone ，theme resolutions 以及对 uploading file 的支持，将请求分发给特定的 handler。

以下为一个请求从浏览器发出到收到请求的生命周期。

当一个请求离开浏览器时①（对应图 2.2 中标识），它携带着用户所需要的信息：至少有一个请求的 URL，或者带有其他数据信息，例如用户通过表单所提交的信息。

请求到达的第一站为 DispatcherServlet,与其它的基于 Java 的 web 框架一样，Spring MVC 将请求汇聚到一个单独的前端控制器 servlet。前端控制器是一个常用的 web 应用设计模式，在这种模式中有一个单独的 servlet 承担着将请求分配给其它应用中的 component 来实现真正处理的职责。在 Spring MVC 中，DispatcherServlet 就是这样的前端控制器。

DispatcherServlet 的作用是将请求发送给某一个 Spring MVC 控制器。控制器是 Spring 中处理请求的 component。但一个典型的应用通常有多个控制器，DispatcherServlet 需要决定应把请求发给谁。因此 DispatcherServlet 询问一个或多个 handler mapping②来决定请求的下一站是哪里。Handler Mapping 通过请求中携带的 URL 来决定。

当选择了正确的控制器后，DispatcherServlet 将请求发送给选择的控制器③。在该控制器中，请求中携带的信息被处理。

控制器做的最后一件事就是将 model 数据打包并给需要生成输出的 view 命名，然后将携带有 modle 以及 view 名字的请求返回给 DispatcherServlet④。

因此，控制器不与特定的 view 相耦合，相反的，它只携带了一个可用于寻找真正渲染结果的 view 的逻辑名。DispatcherServlet 通过询问一个 view resolver⑤将逻辑名与真正的 view 实现相匹配。

在此之后，DispatcherServlet 知道真正渲染结果的 view，最后一步是 view 实现⑥。View 通过 model 数据渲染结果并将结果通过 response 返回至客户端⑦。[3]

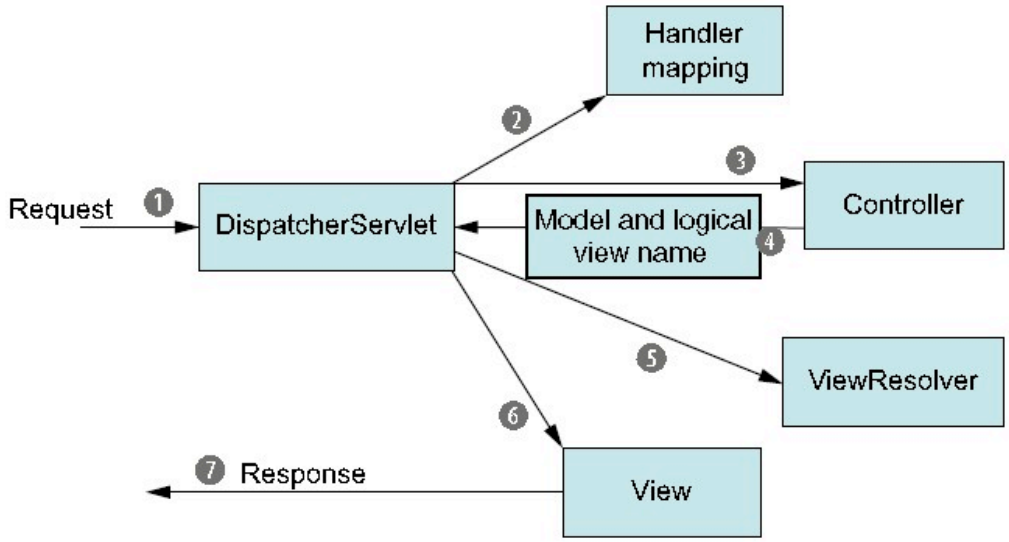

图 2.2 Spring MVC 请求过程图

2.3 MyBatis

MyBatis 的前身 IBatis, 一个 apache 组织的开源项目。在后期，因为项目被迁移至 google，更名为 MyBatis。MyBatis 是支持普通 SQL 查询、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手工设置参数以及抽取结果集。MyBatis 使用简单的 XML 或注解来配置和映射基本体，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java 对象)映射成数据库中的记录。[4]

2.4 Velocity

Velocity 是一个基于 Java 的模板引擎。它能够让网页设计者（前端工程师）引用 java 代码中的方法，并且让前端工程师与 java 后端开发工程师基于 MVC 模型并行工作，这意味着前端开发与后端开发很大程度地被剥离开，各司其职，便于 web 站点的长期维护。

通过模板，velocity 可以生成网页，SQL，PostScript 以及其他内容。它既可以作为一个独立的个体来生成源码和报告，同时也可以很方便地与其他系统集成。[5]

2.5 ECharts

ECharts，来自百度 EFE 数据可视化团队，缩写为 Enterprise Charts，商业级数据图表，一个纯 Javascript 的图表库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器（IE6/7/8/9/10/11，chrome，firefox，Safari 等），底层依赖轻量级的 Canvas 类库 Zrender（一个轻量级的 Canvas 类库，MVC 封装，数据驱动，提供类 Dom 事件模型[6]），提供直观，生动，可交互，可高度个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。

如图 2.3eCharts 结构图所示，eCharts 支持折线图（区域图）、柱状图（条状图）、散点图（气泡图）、K 线图、饼图（环形图）、雷达图（填充雷达图）、和弦图、力导向布局图、地图、仪表盘、漏斗图、事件河流图等 12 类图表，同时提供标题，详情气泡、图例、值域、数据区域、时间轴、工具箱等 7 个可交互组件，支持多图表、组件的联动和混搭展现。[7]

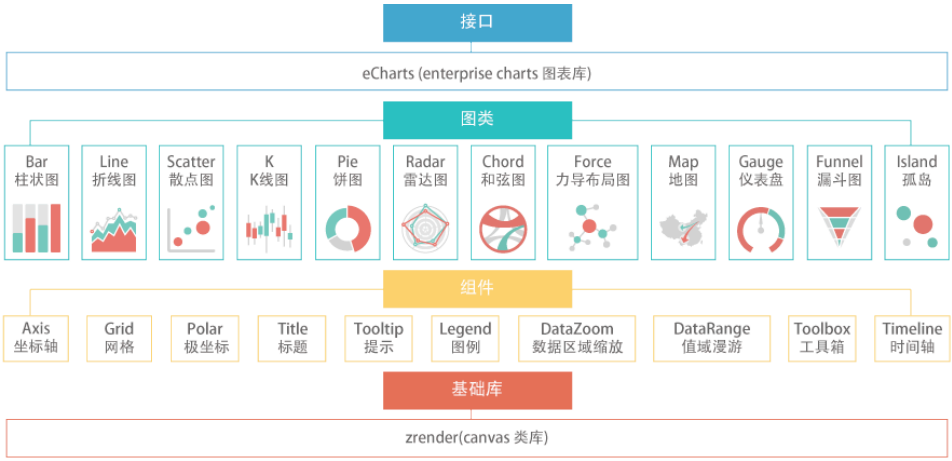

图 2.3 eCharts 结构图

第三章 自定义报表系统需求分析

3.1 系统整体概述

在运营过程中，企业会产生大量的数据，但是庞大且繁杂的数据本身用途甚微，只有将数据以合适的形式展现出来，数据才会产生它应有的价值。因此，制作美观简洁的报表成为企业中大量运行分析人员的共同需求。

传统的 Excel 以及现有各种报表系统易学性较好，但是在自定义方面有一定的局限性；同时，当数据变化时，传统 Excel 以及其他报表系统无法实时更新报表数据，缺乏报表本身所需的实时性；再者，当数据量极大时，传统报表也无法高效率地处理庞大的数据。

因此，自定义报表系统在力求易学易用的前提下，增强报表的可配置性、实时性以及高效性。

3.1.1 系统需求分析

根据对系统的分析，我们可以将系统拆分为两个方面的需求。一是系统配置，二是报表展现。产品经理需负责系统配置，以及能够查看报表，而业务线经理只可以查看报表。同时，我们可以将这两个需求拆分为具体子需求。具体需求划分如图 3.1 所示。

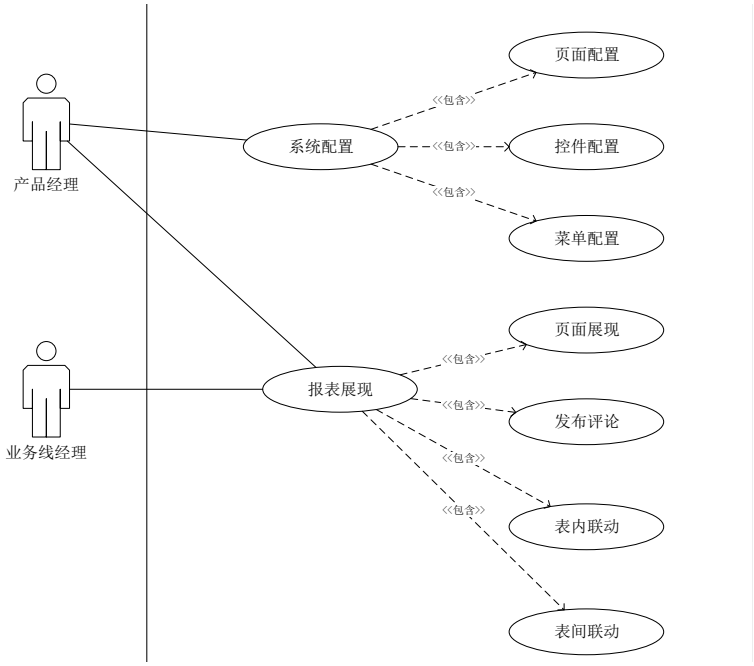

图 3.1 系统用例图

3.2 模块需求分析

3.2.1 页面配置模块需求分析

页面配置包括配置页面之间层次结构、控件布局、负责人、具体控件配置以及菜单配置五个方面的内容。下面对这五个方面依次进行描述。

(1) 配置页面层次结构

报表由多个展现产品线运营数据的页面（page）组成，由于存在多个产品线归属于同一个父产品线（即存在包含关系）的情况，这些产品线所对应的页面也存在包含关系，因此需要配置页面的树状层次结构。

(2) 配置页面内控件布局

报表由多个产品线所对应的页面组成，每个页面中含有包括图、表、评论、筛选条件区域在内的多个控件，用户可以自定义页面中控件的组成和布局。经过页配置之后，页面的基本框架已经形成。

(3) 负责人配置

每个页面由相应的产品经理负责，配置负责人后，该负责人负责对页面中的控件进行配置，并对配置完成后的报表数据进行监控。

(4) 控件配置

页面配置功能仅完成了页面基本框架的搭建，框架中控件的具体内容需要通过该功能进行配置。

表的配置包括对表标题、展现表格所需 SQL、行标题、列标题、数据格式、字体格式配置；图的配置包括对图标题、展现图所需 SQL、图类型、字体格式的配置；筛选条件的配置包括对筛选条件项、筛选条件组成方式的配置

(5) 菜单配置

在完成基本的报表页面之后，用户可以自定义配置页面中得菜单。通过配置菜单项，并将相关页面 URL 绑定到菜单项上，实现了页面链接的完全自定义配置。

同时，为了减少用户自定义配置的复杂度，系统根据用户选择提供相应的菜单模板，如根据页面结构生成菜单模板，根据负责人页面对应关系生成菜单模板等。通过模板生成菜单，节省了用户绑定 URL 至菜单项上的时间。

3.2.2 报表展现模块需求分析

报表展现模块包括页面展现、发布评论、表内联动及表间联动四个方面的内容。

(1) 页面展现

根据用户自定义的系统配置，按照页面配置中的页面框架，通过控件配置中对图和表配置的生成 SQL 语句，以用户定义的格式将图表等控件展现。

(2) 发布评论

用户阅览图表数据后，需对数据所反应的业务情况进行分析和评论。通过在之前配置的评论框中输入评论并保存，可以对报表发布评论。

(3) 表内联动

用户通过之前配置的筛选条件选项，选择筛选条件内容，表内的内容根据筛选条件进行联动变化，从而帮助用户筛选数据，分析业务情况。

(4) 表间联动

用户除了需对表内的数据进行筛选查询外，还涉及到单个筛选查询对多个表起作用、单击某表中某列字段触发其他表的变化、单击某表中某列字段触发弹出框，弹出框中可以放置相关图表控件等功能，这些涉及到多张表格之间互动的功能称为表间联动。

第四章 项目设计与实现

4.1 总体设计

如图 4.1 所示，可以将系统拆分为数据提取、系统配置、报表展现三个主要部分。

1．数据提取：即报表数据的来源，从源数据中提取展现数据至中间库

2．系统配置：自定义配置生成中间表数据的 SQL，展现报表的 SQL，页面图、表、菜单等控件

3．报表展现：根据自定义配置，通过展现 SQL，读取中间库中数据，在后端对数据加工，处理为易于前端处理的格式，经过前端报表库渲染展现报表。

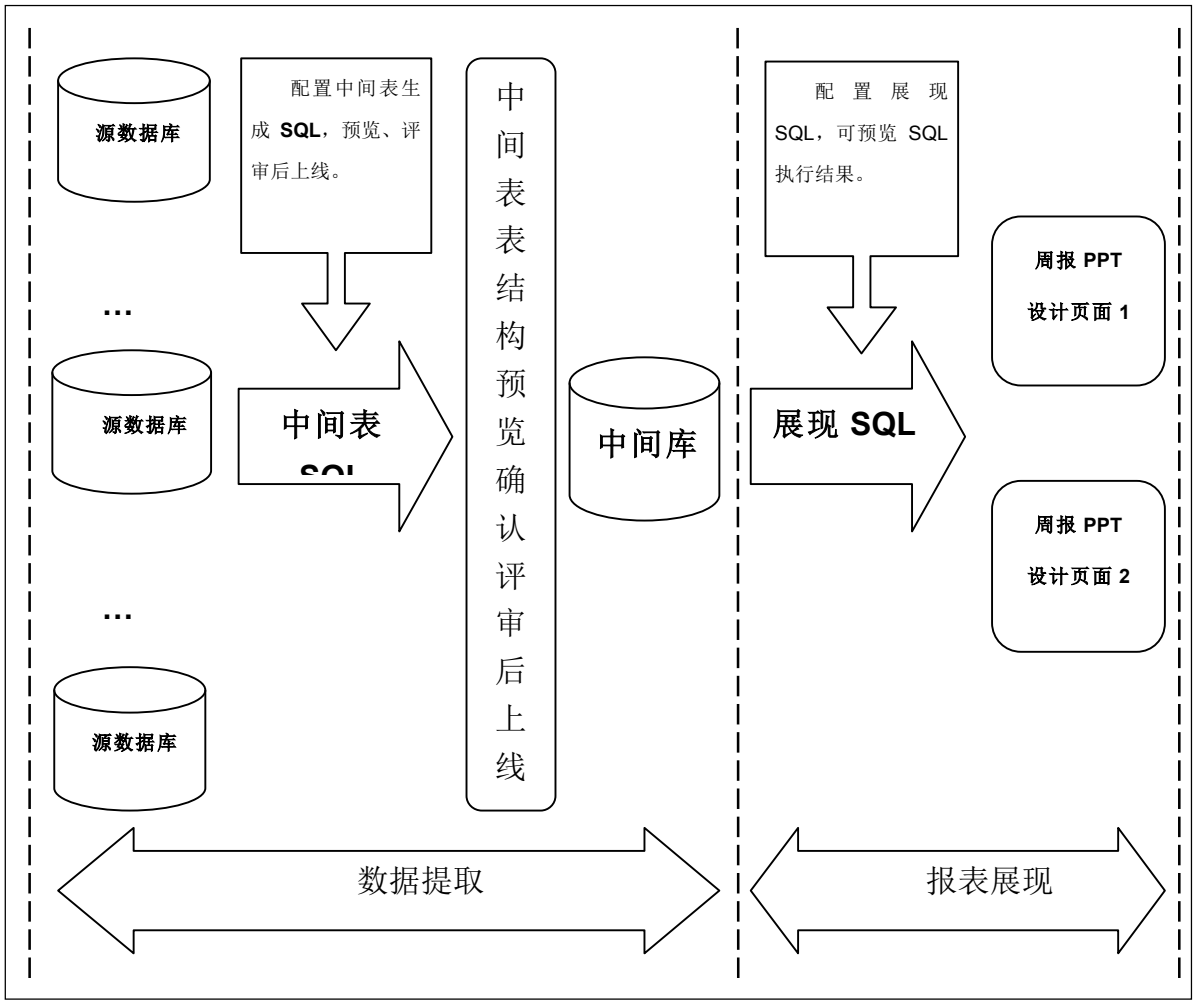

图 4.1 总体设计流程图

4.2 架构设计

4.2.1 应用架构图

根据系统总体设计，应用架构图设计如图 4.2 所示。

报表数据从数据源通过源数据接入层导入本系统数据库，包括元数据管理、任务调度、数据验证的过程，从 source 库中经过 temp 库导入 Mid 库。经过一系列配置，可以将数据依赖关系、数据完备性以及数据本身统一展现，同时还提供报表的生成与下载。

因篇幅有限，本文所论述内容未包含该应用架构中的所有内容。数据依赖关系、数据完备性因涉及数据保密内容未进行描述。元数据管理、任务调度、数据验证为底层数据生成部分，与报表展现无关，因此也不作描述。

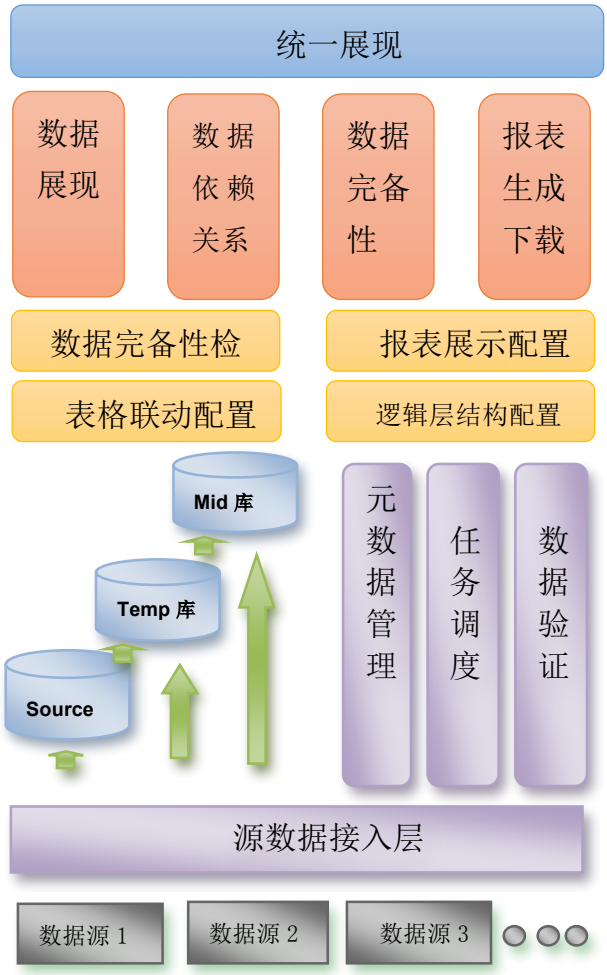

图 4.2 应用架构图

4.2.2 技术实现

运用 velocity + Spring MVC + mybatis 框架，并使用 Spring 中的控制反转和面向切面编程。单元测试采用 junit 测试框架，并利用 jmockit 作为 mock 工具。

4.3 模块具体设计与实现

4.3.1 数据提取模块的设计与实现

源数据库中的数据更新快，来源广，读取和分析的数据量大，导致了 SQL 语句执行速度的降低，影响报表的输出效率。为了减少报表的生成时间，减少多数据源导致的逻辑和维护复杂度，设计了中间库，用来存储中间数据，减少展现 SQL 读取的数据量，从而提高输出效率。

由于报表系统中产品线的变更会比较频繁，并且 SQL 语句由用户自主输入，在考虑系统灵活性的同时，还需考虑系统的易用性。MySQL 作为一种常用的数据库系统，对于用户而言，简单熟悉，易学易用，因此，选择 MySQL 作为数据载体。

通过定时任务脚本，读取用户配置的自定义 SQL 语句，将源数据库中的数据通过生成汇总表的 SQL 语句汇总至汇总表，再通过生成临时表的 SQL 语句提取至临时表，最终通过生成中间表的 SQL 语句提取至中间库的中间表中，实现数据的提取功能。

由于该部分涉及的内容多为数据库操作及定时任务 Shell 脚本内容，在此不做详细描述。

4.3.2 系统配置模块的设计与实现

本节对系统配置模块进行整体概述，并具体描述该模块的设计与实现。

(一) 模块概述

由于灵活性是本系统设计的重点，因此能够给用户提供灵活的系统配置十分重要。然而，灵活性也相应地带给了用户配置时的复杂度。因此，如何在满

足用户灵活配置的需求的前提下，给用户配置提供足够的便利成为此模块设计需考虑的一个方面。

(二) 具体阐述

对于此模块的阐述分配置流程设计、数据库设计、包划分三部分进行。配置流程阐述实现一系列系统配置的流程，以及各配置之间如何协同工作；数据库设计阐述实现该配置的 ER 模型，包划分阐述实现该配置的具体程序设计

(1) 配置流程设计

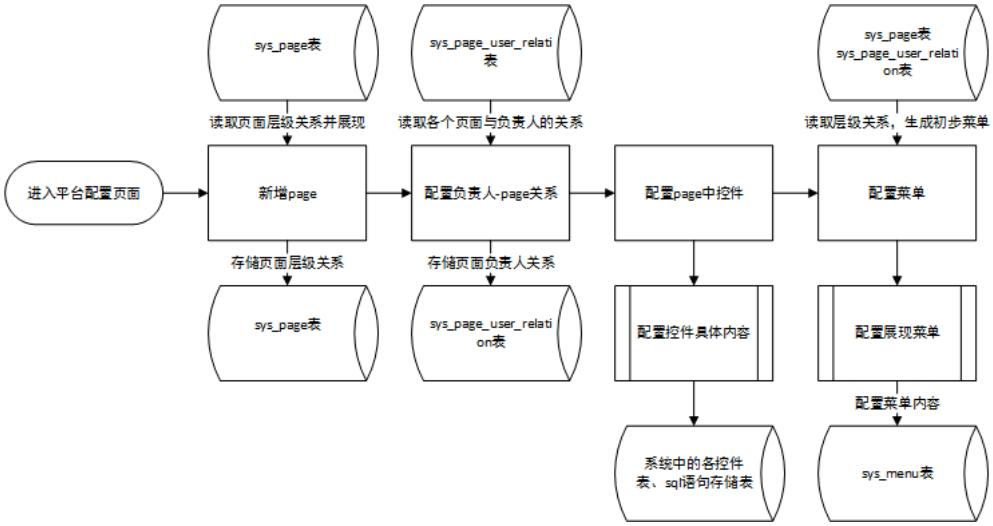

图 4.3 配置流程图

如图 4.4 所示，page（即页面）是系统配置中的核心，所有的配置都以 page 配置为中心。

a. 用户进入系统配置页面，选择新增页面。

b. 进行页面-负责人配置，将该 page 分配给相关负责人

c. 对该页面中插入图表等不同控件

d. 对页面中插入的图表等控件根据控件的不同类型进行配置，例如图表需要配置展现所需的 SQL 语句、文字格式、数字格式等，评论控件只需配置标题，条件筛选控件需要配置筛选项。

此外，对于用户自定义配置的 SQL 语句，系统会运用百度内部的 sqlParsered 的 SQL 语法解析器进行分析，以保证 SQL 语句的正确性。

e. 配置菜单，系统根据页面的层次结构关系以及页面负责人对应关系生成相应模板，用户可以选择是否根据模板生成菜单，自动绑定相关页面 URL 至菜单项。

至此，报表系统的配置流程结束，配置完成。

(2) 实体关系模型

实体关系模型如图 4.5 所示，每个负责人可以负责多个页面，同时一个页面也可以由多个负责人负责，因此负责人与页面之间的关系为 m:n。

每个页面可以包含多个评论控件，因此页面与评论控件之间的关系为 1:n。

每个页面包含多个表控件、图控件，因此页面与表控件、图控件之间的关系为 1:n。

每个图控件由多个展现 SQL 语句实现，因此图控件与展现 SQL 之间的关系为 1:n。

每个表控件由一个展现 SQL 语句实现，因此表控件与展现 SQL 之间的关系为 1:1。

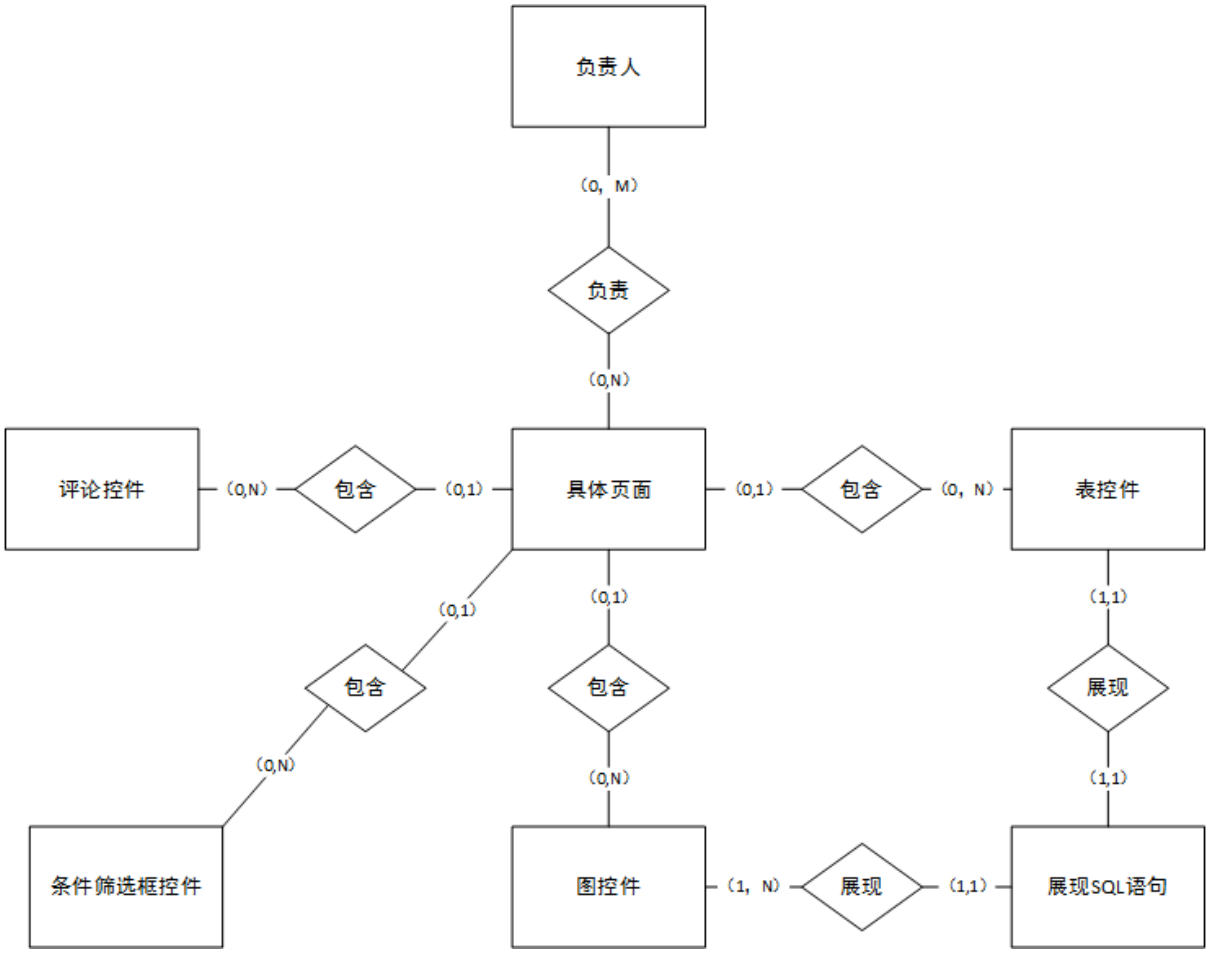

图 4.4 页面配置 ER 图

(3) 表结构描述：

a.**sys_user** 表 （负责人表）

该表存储负责人的相关信息

表 4.1 负责人表

字段名称	字段解释

id	负责人 id（自增）
name	负责人姓名
other_info	负责人相关信息(可扩展字段)

b. **sys_page_info 表**

该表存储报表中各页面的相关信息

表 4.2 页面信息表

字段名称	字段解释
id	页面 Id（自增）
parent_id	父页面 id（存储页面之间的包含关系）
page_title	页面标题

c. **sys_user_page_relation 表**

该表存储负责人与页面的对应关系，一个负责人可以负责多个页面，一个页面也可以有多个负责人负责（多对多）。

表 4.3 负责人-页面关联表

字段名称	字段解释
id	负责人-页面关系 id（自增）
user_id	负责人 id
page_id	页面 id

d. **sys_page_widget_table 表**

该表定义表控件的基本信息

表 4.4 表控件信息表

字段名称	字段解释
id	表格模型 id(自增)
table_title	表格主标题
table_background_color	标题背景颜色
content_background_color	数据内容背景颜色（配置形式：red 或#AABB99）
content_font	数据内容文字字体
content_bold	0-正常 1-粗体
show_row_title	数据内容字体颜色（配置形式：red 或#AABB99）
show_column_title	是否显示多级行标题

e.**sys_page_widget_chart** 表

该表定义图控件的基本信息

表 4.5 图控件信息表

字段名称	字段解释
id	图表模型 id(自增)
chart_title	图表主标题
left_ytype	图形左边 y 轴类型 0 数字 1 百分比
right_ytype	图形右边 y 轴类型 0 数字 1 百分比

f. **sys_page_widget_comment** 表

该表定义评论控件的基本信息

表 4.6 控件-评论关联表

字段名称	字段解释
id	评论控件 id（自增）
comment_title	评论标题
comment_content	评论内容

g.**sys_page_widget_condition_area** 表

该表定义条件筛选区域控件的基本信息

表 4.7 控件-筛选条件信息表

字段名称	字段解释
id	条件筛选区域控件 id（自增）
area_title	区域标题（可为空）
remarks	备注

h.**sys_page_widget_relation** 表

该表存储页面与控件的关联关系，并描述控件在页面上的布局

表 4.8 页面-控件关联表

字段名称	字段解释
id	页面-控件关联关系 id（自增）
page_id	页面 id
widget_id	控件 id
widget_type	控件类型，1-图，2-表，3-评论框，4-条件筛选区域
display_order	展现顺序，用于描述控件在页面上布局

i. **sys_condition_info** 表

该表存储用于筛选的各种条件信息

表 4.9 筛选条件信息表

字段名称	解释
id	筛选条件 id（自增）
condition_area_id	条件筛选区域 id
display_order	展现顺序
condition_title	筛选条件标题，显示在筛选条件控件前面的文字
title_column_name	筛选条件 value 对应数据表标题字段名称。筛选条件为单行文本框时该字段有效。其他情况为空字符串即可。
show_type	页面显示筛选条件控件类型： 1-下拉菜单 2-单行文本框 3-单选按钮组 4-复选框组 5-多行文本框 (目前系统仅支持下拉菜单、单行下拉菜单)
remarks	备注

j. **sys_condition_item_info** 表

该表存储条件筛选项，当筛选条件不是输入框（即为选择框）时，需要提供选择的筛选项。通过该表，存储条件筛选项，并存储与条件筛选的关联信息。

表 4.10 筛选条件项信息表

字段名称	解释
id	筛选条件项 ID（自增）
condition_id	筛选条件 ID，填写 sys_condition_info 表 id 字段值
item_value	筛选条件项含义值，填写条件表达式，用于改写查询 SQL 语句为带筛选条件的查询 SQL 语句
show_text	筛选条件项页面显示文本，显示在条件是筛选控件上条件项的文字。

k. **sys_query_data_SQL_info** 表

该表存储用于展示数据的 SQL 查询语句

表 4.11 查询 SQL 信息表

字段名称	字段解释

id	查询数据 SQL id (自增)
widget_id	SQL 从属控件 Id
type	SQL 从属控件类型 1-图类型 2-表类型
SQL_title	SQL 名称（图控件专用字段，此字段表示该 SQL 所对应数据线标题）
SQL_content	SQL 内容
SQL_order	相同 widget 的 SQL 执行顺序
chart_type	图控件专用字段，图类型： 0-线性，1-饼图，2-面积图，3-柱状图
bind_yaxis	图控件专用字段，绑定的 y 轴是左边还是右边： 0-左边， 1-右边
feature_type	图控件专用字段，指标类型：0-数字，1-百分比
feature_title	图控件专用字段，指标名称
feature_color	图控件专用字段，指标颜色（可用 red 颜色英文名或#990030RGB 数值的形式）

l. **sys_page_widget_table_title_row** 表

该表存储行标题基本信息，以及标题之间的父子关系

表 4.12 行标题信息表

字段名称	字段解释
id	行标题 id（自增）
parent_title_id	父标题 id
query_data_SQL_id	查询 SQL id
title	标题名称
show_order	显示顺序

m. **sys_page_widget_table_title_column** 表

该表存储列标题基本信息，以及标题之间的父子关系

表 4.13 列标题信息表

字段名称	字段解释
id	列标题 Id
parent_title_id	父标题 Id
query_data_SQL_id	查询 SQL id

title	标题名称
query_SQL_column_name	查询 SQL 字段名称
query_SQL_column_real_name	查询 SQL 字段原本名称
font	字体
size	字号
bold	是否粗体： 0-否 1-是
italic	是否斜体： 0-否 1-是
dash	是否下划线： 0-否 1-是
tend_color	是否显示趋势颜色：0-否 1-是
font_color	字体颜色：颜色英文名/RGB
background_color	背景颜色：同上
show_text_type	文本显示类型
decimal_type	小数保留位数
order_type	排序类型 0-英文排序 1-中文排序
show_order	显示顺序

n.**sys_menu** 表

该表存储菜单中菜单项的基本信息

表 4.14 菜单信息表

字段名称	字段解释
Id	菜单项 id（自增）
parent_id	父菜单项 id
type	菜单类型
menu_id	菜单项所属菜单
name	菜单项名字
url	菜单项所对应的 url
Sequence	菜单项展现顺序

(4) 包划分

如图 4.5 所示，采用典型的分层结构设计包和类的划分。将项目划分为 bo 层、dao、service 层和 controller 层。bo 层为简单实体对象，与数据库中实体表数据结构保持一致；dao 层利用 MyBatis 这一持久层框架与数据库进行交互，进行增删改查等数据库操作；service 层为业务逻辑层，报表中表、图等控

件数据的组合拼接等业务逻辑操作在该层实现；controller 层与前端引擎模板交互，接收前端请求，以 ModelAndView 格式或 json 格式返回请求结果。

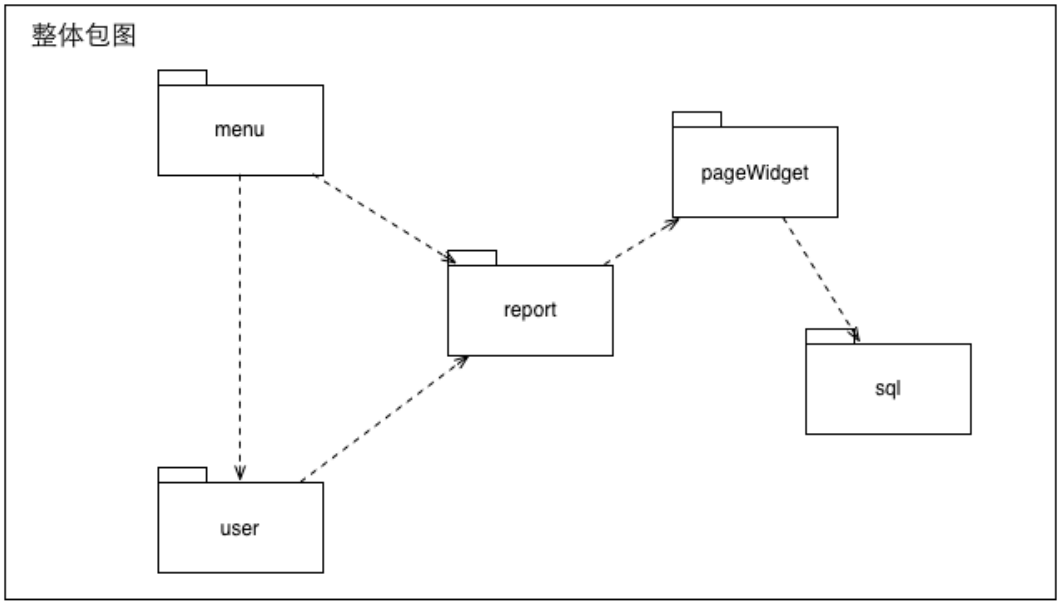

图 4.5 整体包图

4.3.3 报表展现模块的详细设计

该节从页面展现、发布评论、表内联动、表间联动四个方面介绍报表展现模块，描述如何将配置内容展现为报表形式。

（一）页面展现

（1）图控件展现

图控件是本系统中较为简单的一个控件，可以将其划分为图标题、横坐标、纵坐标、图例等几个部分，如图 4.6 所示（饼图除外）：

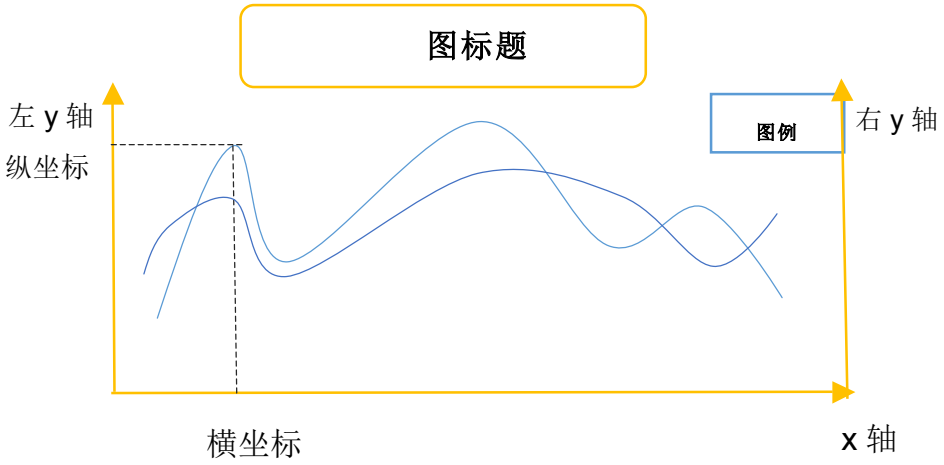

图 4.6 图控件布局示意图

1) 图控件展现原则

由于图控件在前端展现的多样性，图的展现遵循几个原则：

a. 线图中一根线或柱状图中一种柱状对应一条查询 SQL 语句

一个图控件当中，可能会包含多种数据展示，每个数据展示对应一条查询 SQL 语句。如图 4.7 所示，它包含了两种数据展示，分别用橙色线和蓝色线显示，这两根线各使用了一条 SQL 查询出的结果进行展示。

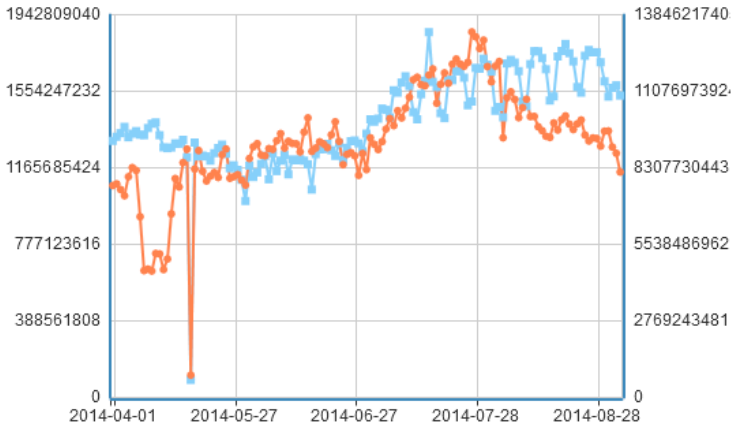

图 4.7 折线图控件示意图

b. 每个饼图对应一个 SQL

c. 每条查询 SQL 两个字段

每条 SQL 查询结果只能包含两个字段，分别对应图中横轴和纵轴，对应横轴的字段别名为"name"，纵轴对应的字段别名为"value"。例如：select date as 'name', XXX as 'value' from .....

2) 图控件渲染流程图

如图 4.8 所示，页面开始渲染控件，判断是否为图控件，若是则读取图控件所关联的展现 SQL 语句内容，执行该 SQL 语句，得到所需数据之后，将纯数据与该图控件相关的图控件配置组装，结合成前端易于读取并渲染的 json 数据，前端根据 json 数据，利用 eCharts 图表渲染库渲染，将图控件显示在页面上。

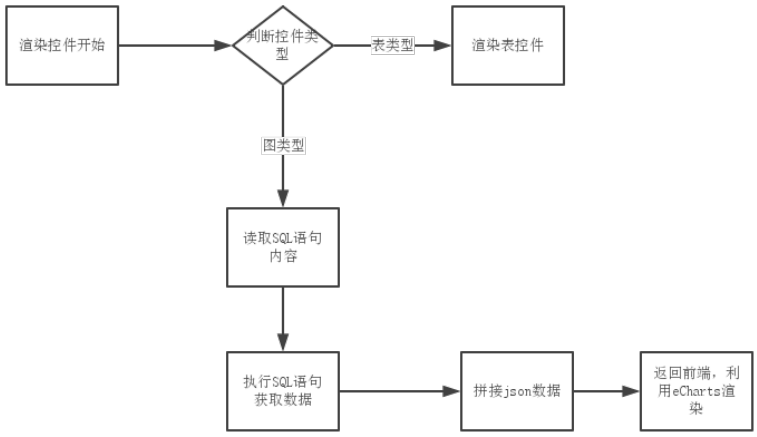

图 4.8 图控件渲染流程图

3) 图控件渲染类图

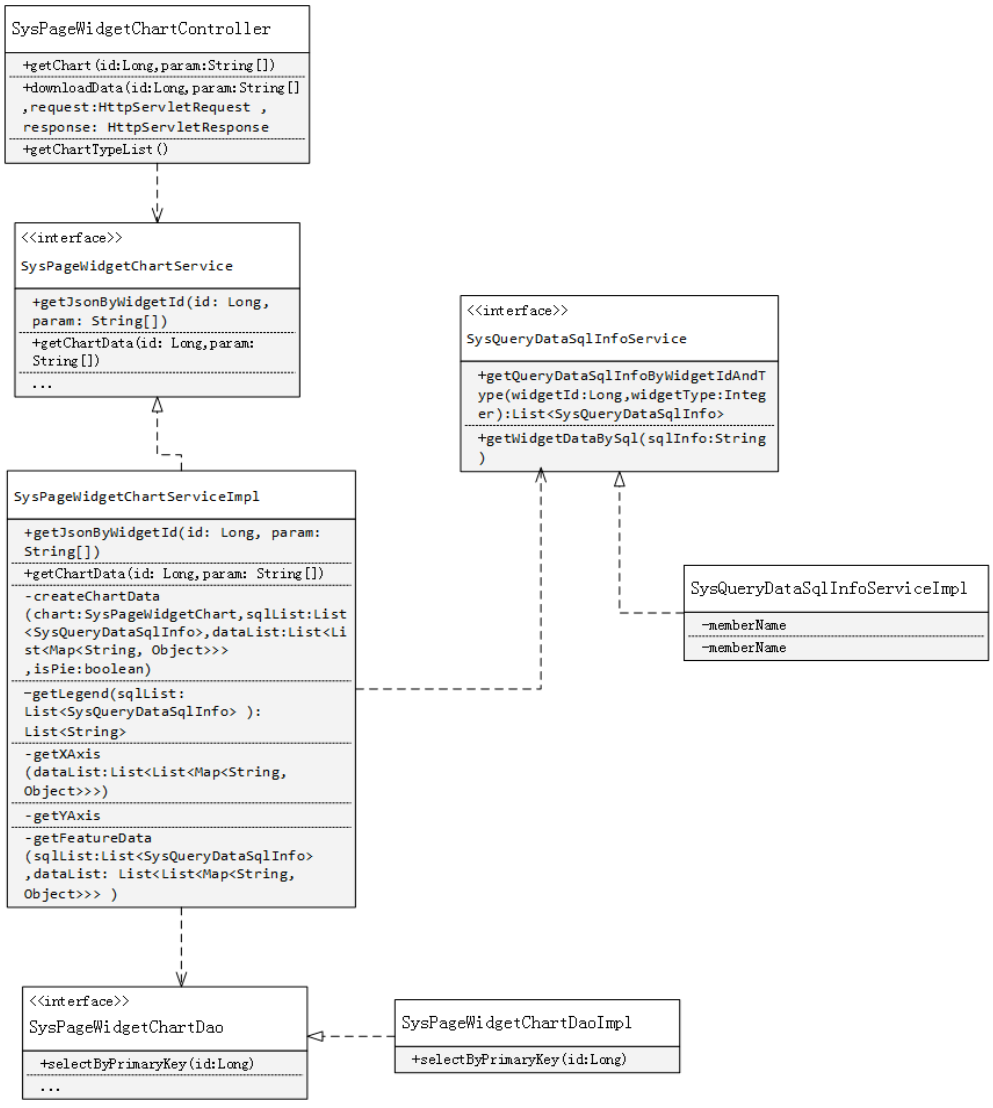

图 4.9 图控件渲染类图

如图 4.9 所示，controller 负责响应前端请求，并将其转发给相应的 service 接口，Dao 层负责持久层的操作。

4) 图控件 Json 格式

■ 线、面积、柱状图或其复合图 json 格式

```
{
"type": "chart", //  表明为图控件
"data":
        {
"title": "title1",
"subTitle": "subTitle1",
"legend": ["lg1", "lg2", "lg3"],//分类解释  由与 widget 关联的 SQL 决定，每个 SQL 一个分类，每个 SQL 代表一条曲线或一个柱状图
"XAxis": ["x1", "x2", "x3"],    //x 轴底线的值  由与 widget 关联的 SQL 决定，约定 SQL 查询出的第一列数据集合为 x 轴底线的值
"yAxis": [{type:1,unit:1},{type:1,unit:1}], //y 轴的值及 title  可定义多个 y 轴
"Data": [{
    "name":  "lg1",//与上面的分类解释对应，约定  SQL  只有两列  title  和 value，约定 SQL 结果集的第一行的 title 值必须为此 SQL 的指标名称
      "type":  "bar  or  line",//曲线对应的图表类型  柱状或曲线或几何图  配置在 SQL 中  (面积图也用 type=line)
      "smooth": true,//面积图专用
      "itemStyle": {normal:{areaStyle: {type:'default'}}},//面积图专用
      "data":  ["val1",  "val2",  "val3"],//SQL 查出的数据集  约定从第二行开始的 value 字段值集合（第一行是 title 指标名称）
      "yAxisIndex" :"0||1",//该曲线用哪个 y 轴    左边还是右边
      "color": "red"//指标线条颜色可选
                  },
                  {
                          "name": "lg1",
                          "type": "bar or line or lineArea",
                          "data": ["val1", "val2", "val3"]
                    }
              ]
          }
        }
```

图 4.10 图控件渲染 json 格式 1

■ 饼图 json 格式

```
{
          "type": "chart",//饼图
          "data":
          {
                "title": "title1",
```

```
            "subTitle": "subTitle1",
            "legend": ["lg1", "lg2", "lg3", "lg4", "lg5"],
            "Data": [
                {
                    "name": "tooltip",
                    "type": "pie",
                    "data": [
                        {"value":335, "name":"直接访问"},
                        {"value":310, "name":"邮件营销"},
                        {"value":234, "name":"联盟广告"},
                        {"value":135, "name":"视频广告"},
                        {"value":1548, "name":"搜索引擎"}
                    ]
                }
            ]
        }
    }
```

图 4.11 图控件渲染 json 格式 2

5) 案例：

a. sys_query_data_SQL_info 表

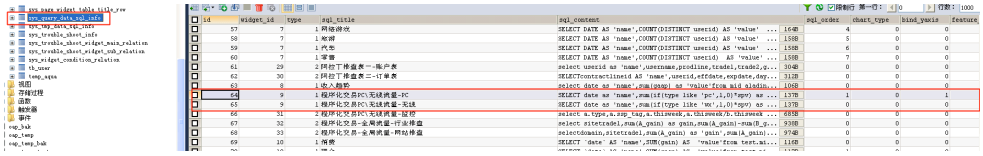

图 4.12 图控件实现案例表数据 1

b. sys_page_widget_chart 表

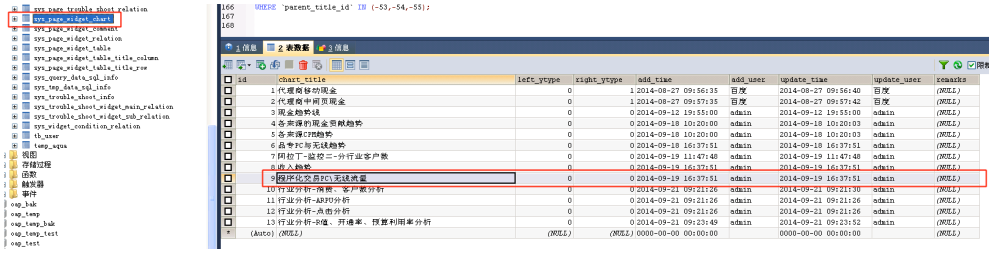

图 4.13 图控件实现案例表数据 2

(2) 表控件展现

表控件是该报表系统中比较复杂的控件之一，主要涉及数据查询、数据下载、自定义多级列标题、自定义多级行标题、分页、排序、条件查询、事件触发、弹框等内容。为简单明确的描述表控件的设计与实现，把表控件分成五大区域，五大区域的描述详见图 4.14 表控件区域示意图。

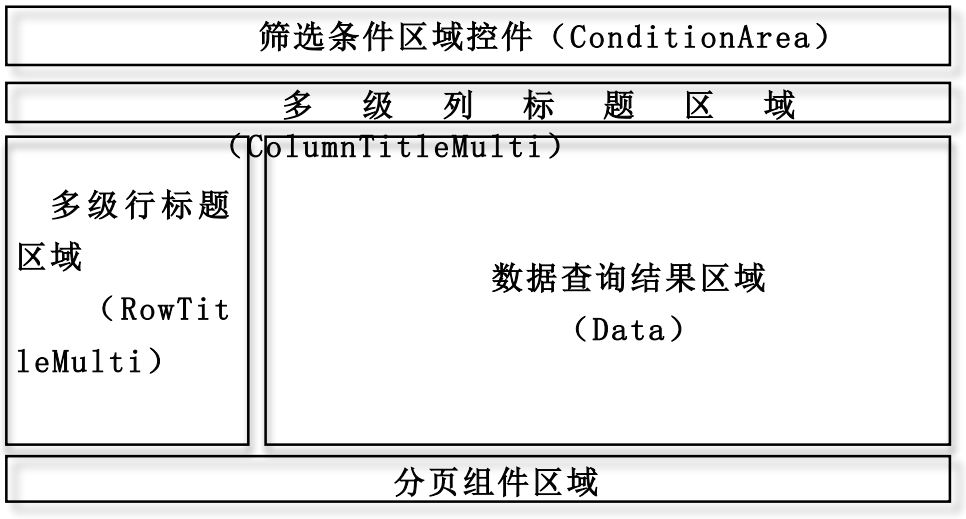

图 4.14 表控件区域示意图

1) 表控件渲染类图

如图 4.15 所示，在自定义报表系统报表渲染模块中，sysPageWidgetTableController 负责响应前端请求，将请求转发给 SysPageWidgetTableService 对应的接口处理，SysPageWidgetTableServiceImpl 实现了 Service 接口，同时调用 SysPageWidgetTablePaginationService 接口获取分页信息、SysPageWidgetTableTitleRowService 接口获取行标题信息、SysPageWidgetTableTitleColumnService 接口获取列标题信息、SysQueryDataSqlInfoService 接口获取展现 SQL 信息、SysConditionInfoService 获取条件筛选信息，通过这些 service 的协作，实现了表控件的渲染以及表控件之间的联动。

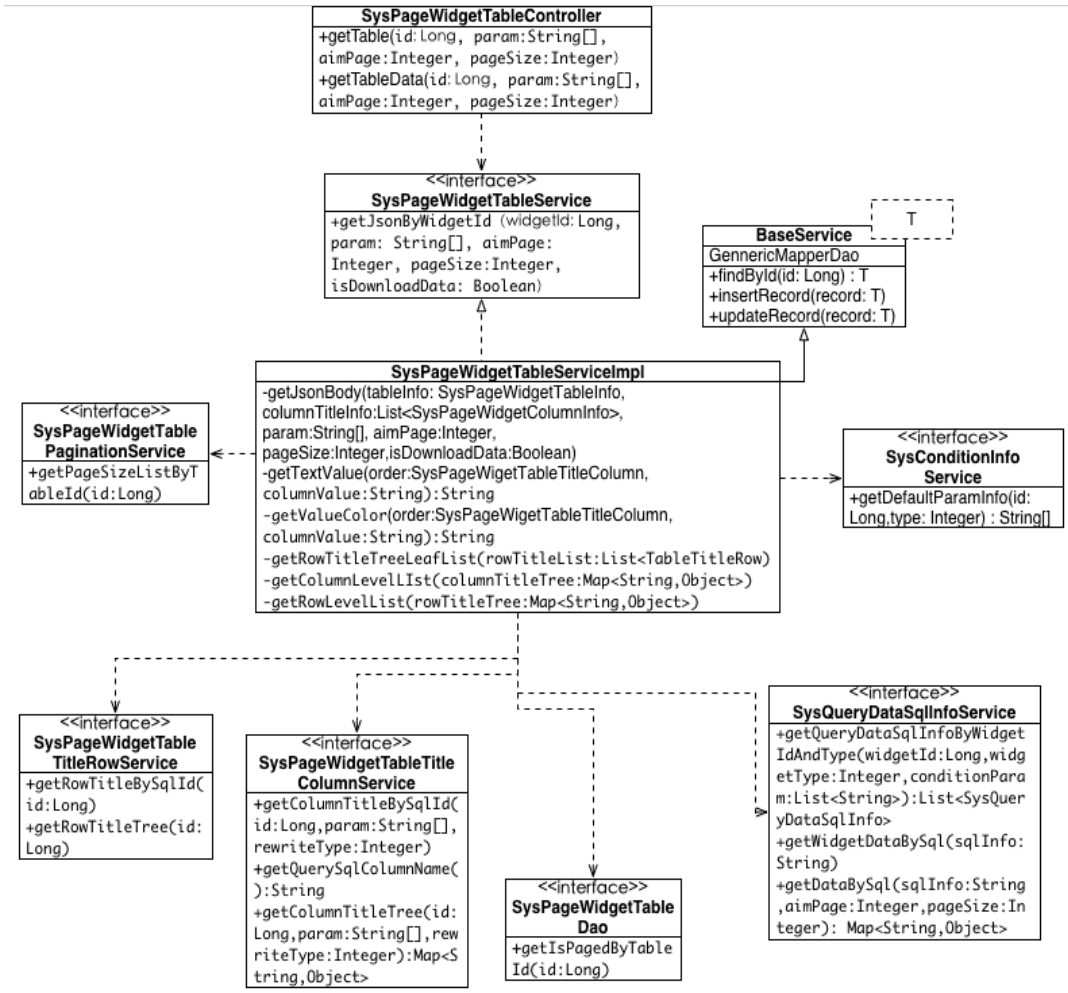

图 4.15 表控件渲染类图

2) 表控件的区域描述

a. 筛选条件区域控件（ConditionArea）：该组件是为了满足需求分析中所提及的“表内联动”功能而设计。在项目前期与特定控件绑定，主要用于放置条件筛选（condition）组件，用于为表控件传递筛选条件。项目后期，为了使得页面配置更加灵活，将组件与页面绑定，作为独立的区域参与页面布局，因此该组件升级为页面中的控件。page、conditionArea、condition、conditionItem之间的包含关系可参考图 4.16 筛选条件区域包含关系图：

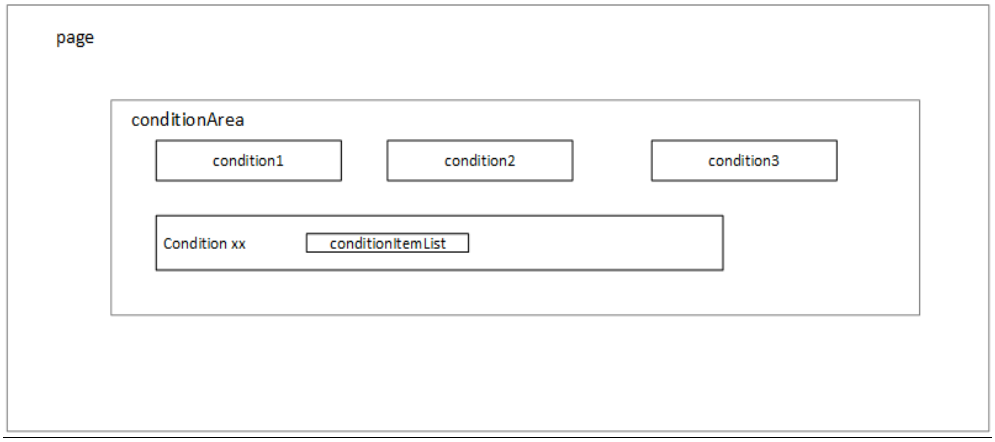

图 4.16 筛选条件区域包含关系图

涉及数据表:

表 4.15 筛选条件涉及数据表

字段	解释
sys_page_widget_condition_area	筛选条件区域控件基本信息
sys_condition_info	条件组件基本信息
sys_condition_item_info	条件组件内部项目信息（下拉菜单类型）
sys_page_widget_relation	页面与控件绑定信息

b. 多级列标题区域（ColumnTitleMulti）:

表控件存在列标题为一级或多级的情况，如果列标题为一级，则按顺序形成列表在前端渲染即可。如果存在多级标题，则需从数据库中读取并形成父子结构，再转化为前端易渲染的格式传输。

如上文所述，数据库中列标题的存储格式如下表所示（重点为 id 与 parent_title_id）：

表 4.16 列标题信息表

字段名称	字段解释
id	列标题 Id
parent_title_id	父标题 Id
query_data_SQL_id	查询 SQL id
title	标题名称
query_SQL_column_name	查询 SQL 字段名称
query_SQL_column_real_name	查询 SQL 字段原本名称

font	字体
size	字号
bold	是否粗体： 0-否 1-是
italic	是否斜体： 0-否 1-是
dash	是否下划线： 0-否 1-是
tend_color	是否显示趋势颜色： 0-否 1-是
font_color	字体颜色：颜色英文名/RGB
background_color	背景颜色：同上
show_text_type	文本显示类型
decimal_type	小数保留位数
order_type	排序类型 0-英文排序 1-中文排序
show_order	显示顺序

同时，需要注意两点：

1、为了减少不必要数据的传输，多级列标题（可以看成是一颗树）只有叶子节点标题传递全属性字段，其他上级标题只传递“标题”和“横纵向合并单元格数”。

2、后端向前端传递标题信息时，按照 html 的行进行传递，详见示意图（图 1-2）红色为第一行数据，蓝色为第二行数据。

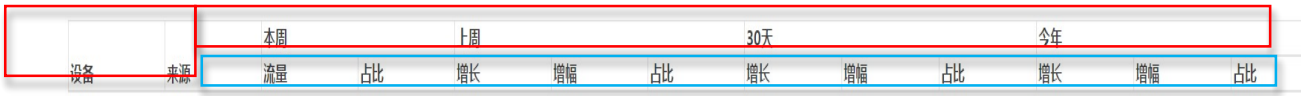

图 4.17 多级列标题示意图

a) 涉及数据表:

表 4.17 列标题涉及数据表

表名	注释
sys_query_data_SQL_info	用于查询数据的 SQL 语句信息
sys_page_widget_table_title_column_multi	列标题配置及标题树状结构

b) 涉及算法：标题树状结构转换成前端渲染格式算法

从数据库中读取标题信息，并根据父子关系生成树状结构（getColumnTitleTree），将树状结构需转换为易于前端渲染的 Json 格式(getColumnLevelList)。

算法思想： 使用队列对标题树进行非叶子节点层次遍历，叶子节点作为特殊处理的树节点，作为最后一行追加至渲染数据的最后。

算法伪代码描述：

```
Function getColumnLevelList(Map<String, Object> tree)
Begin
    新建 List 为 result 存储返回结果
childList <- tree.getChild()
If <childList 不为空> Then
{
    queue <- childList
    int pCount <- childList.size()
    int cCount <- 0
result.add(childList)
赋值一个新的 ArrayList 给 childList
新建一个 ArrayList leafLevel 用于存储叶节点
     While （queue.size>0） do
      {
        从 queue 取出一项赋值给 item，然后从队列中删除该项
        child = item.getChild
        if <child 不为空> Then
        {
          For co : child.first to child.last {
              If <co 是叶节点> Then
              {将 co 添加至 LeafLevel 中}
              Else
              {
                  将 co 添加至 childList
                  将 co 添加至 queue
                   cCount ++
              }
          }
        }
        移除 item 中的所有 child
        pCount –
        if <pCount 等于 0> Then
        {
            pCount <- cCount  //之前的子节点变为现在的父节点
            if <childList 不为空> Then
            { result.add(childList) }
            新建一个 ArrayList 赋值给 childList，用于存储当前节点的子
点
        }
      }
```

```
        result.add(leafLevel)
}
return result
End
```

图 4.18 列标题格式转换算法伪代码

c) 列标题 Json 格式

```
"columnTitle": [
                    [
                        {"id": 602,"vertical": 1,"distance": 1, "title": "主标题
","horizon": 3},
                        {"id": 587, "vertical": 1, "distance": 1, "title": "本周",
"horizon": 2},
                        {"id": 588, "vertical": 1, "distance": 1, "title": "上周",
"horizon": 3},
                        {"id": 589,"vertical": 1,"distance": 1,"title": "30 天
","horizon": 3},
                        {"id": 590,"vertical": 1,"distance": 1,"title": "今年
","horizon": 3}
                    ],
                    [
                    {//该部分为完整的子标题信息，以下省略子标题字体配置
具体信息
                    "vertical": 1,"fontColor": "#000000","orderType": 0,
                    "querySqlColumnRealName": null,"backgroundColor":

                    "font": "宋体","horizon": 1,"italic": false,"size": 10,
                    "id": 603,"tendColor": false,"title": "总体标题",
                    "distance": 0,"bold": false,"dash": false,
                    "decimalDigit": 0,"querySqlColumnName":
Type": 0
                    },
                    {
                    "vertical": 1,"querySqlColumnRealName": null,"id": 585,
                    "title": "设备", "querySqlColumnName":
extType": 0
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "来源",
                        "querySqlColumnName": "name","showTextType":
                    0
                    },
```

```
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "流量,
                        "querySqlColumnName":
                    "spv_thisweek","showTextType": 0
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "占比,
                        "querySqlColumnName":"spv_thisweek_share_r",
                        "showTextType": 0
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "增长,"querySqlColumnName":
lastweek_grow",
                        "showTextType": 2
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "增幅","querySqlColumnName":
lastweek_rate_r",
                        "showTextType": 1
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "占比,
                        "querySqlColumnName": "spv_lastweek_share_r",
                        "showTextType": 0
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":

                        "title": "增长","querySqlColumnName":
lastmonth_grow",
                        "showTextType": 3
                    },
                    {
                        "vertical": 1,"querySqlColumnRealName": null,"id":
```

```
                    "title": "增幅",
                    "querySqlColumnName": "spv_lastmonth_rate_r",
                    "showTextType": 0
                },
                {
                    "vertical": 1,"querySqlColumnRealName": null,"id":

                    "title": "占比",
                    "querySqlColumnName": "spv_lastmonth_share_r",
                    "showTextType": 4
                },
                {
                    "vertical": 1,"querySqlColumnRealName": null,"id":

                    "title": "增长","querySqlColumnName":
thisyear_grow",
                    "showTextType": 3
                },
                {
                    "vertical": 1,"querySqlColumnRealName": null,"id":

                    "title": "增幅","querySqlColumnName":
thisyear_rate_r",
                    "showTextType": 5
                },
                {
                    "vertical": 1,"querySqlColumnRealName": null,"id":

                    "title": "占比","querySqlColumnName":
thisyear_share_r",
                    "showTextType": 6
                }
            ]
        ]
```


图 4.19 列标题 json 格式图

c. 多级行标题区域（RowTitleMulti）：

主要用于渲染多级行标题，渲染原理、传递数据与列标题类似。

涉及数据表:

表 4.18 行标题涉及数据表

表名	注释

sys_page_widget_table_title_row_multi	行标题配置及标题树结构

d. 数据查询结果区域（Data）：根据配置的数据查询 SQL，将 SQL 查询的结果显示至此区域。值得注意的是：

a) 数据列显示的顺序是与列标题的顺序一致。

我们约定，sys_query_data_SQL_info 表中的 SQL 语句通常遵循以下格式：SELECT xx as name, xx as columnName1, xx as columnName2, … … FROM ……

“name” 所对应的内容作为行标题，通过将“columnName1”、“columnName2”与 sys_page_widget_table_title_column_multi 表中的 query_SQL_column_name 字段对应，可以找到所查询每列数据所对应的列标题。因此，数据列显示的顺序与列标题的顺序是一致的。

b) 在配置了行标题的情况下，对表格数据进行排序操作，只能在同类一个（行标题）大类的行标题内进行排序。

c) 在配置了行标题的情况下，当数据第一列没有与配置的行标题匹配的时候，自动添加到“其他”行标题下。

涉及数据表：

表 4.19 数据查询结果涉及数据表

表名	注释
sys_query_data_SQL_info	控件数据查询 SQL 配置表

e. 分页组件区域（Pagine）：分页组件显示区域，在渲染表控件时，携带分页组件参数，参数名称及含义如下：

表 4.20 分页组件参数含义表

参数名	注释
totalPage	总页数
totalRow	总行数
currentPage	当前页数

涉及数据表:

表 4.21 分页组件涉及数据表

表名	注释
sys_page_widget_table	表控件基本信息（包括分页类型配置）
sys_page_widget_table_pagination	分页列表信息

分页功能实现流程：

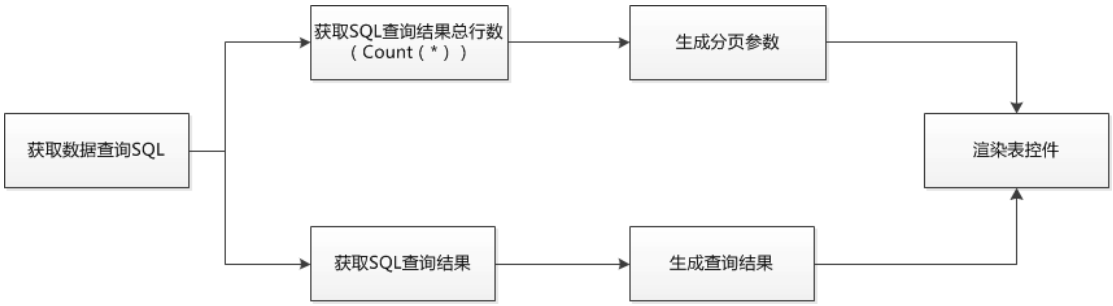

图 4.20 分页功能实现流程图

f. 前端展现拼接

表控件展现按照前述的几个区域，生成之后拼接而来。表控件渲染数据处理流程如图 4.3.3-15

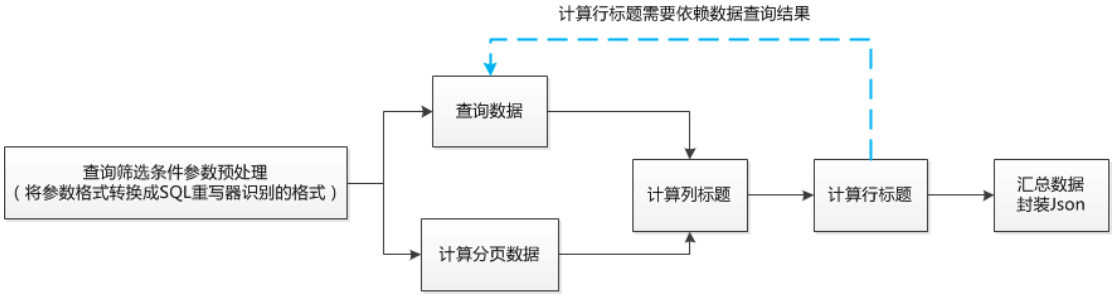

图 4.21 表控件渲染数据处理流程图

根据如上表控件渲染数据处理流程，最终生成提交给前端渲染的“表控件展现 Json”。表控件展现 json 接口定义如下：

```
{
    "success": "true",
    "hasSuccess": true,
    "resultInfo": "查找成功",
    "data": {
        "type": "table",
        "data": {
            "columnTitle": [
                ....//列标题
            ],
            "rowTitle": [
                ....//行标题
            ],
            "body": {//数据体
                "bodyBackgroundColor": null,
                "bodyFont": null,
                "bodyBold": false,
                "bodyFontColor": null,
                "data": [
```

```
                        ......//数据内容
                    ],
                    "otherList": [
                        ......//未配置行标题的数据，前端渲染时暂未使用，可忽略此项数据
                    ],
                    "dataTitle": [
                        ......//首列数据，前端渲染时暂未使用，可忽略此项数据
                    ]
                }
            },
            "paginationInfo": {//分页信息
                "curPage": 1,
                "totalPage": 1,
                "totalRow": 14,
                "pageSize": 20,
                "pageSizeList": []
            },
            "orderInfo": {//排序信息
                "column_value": "contractlineid",
                "sequence": "DESC"
            }
        }
}
```

图 4.22 表控件整体 json 示意图

3) 表控件总结

表控件中组件的复杂度和多样性决定了其实现的复杂度，通过对表控件的区域划分，使得各个区域既相互独立，又能协同工作。在实现过程中，我们经历了从简单的展现，到条件筛选，到分页、排序功能的不断增加，形成了“参数传递”+“SQL 改写”的思想。

(一) 发布评论

用户阅读展现的报表，对业务数据进行总结分析，将分析评论撰写在系统页面上并保存。该模块基于上一个部分所提及的系统配置中的控件配置，用户在空间配置中配置的评论框输入评论并保存，该项功能即可实现，因此较为简单。类图实现如下：

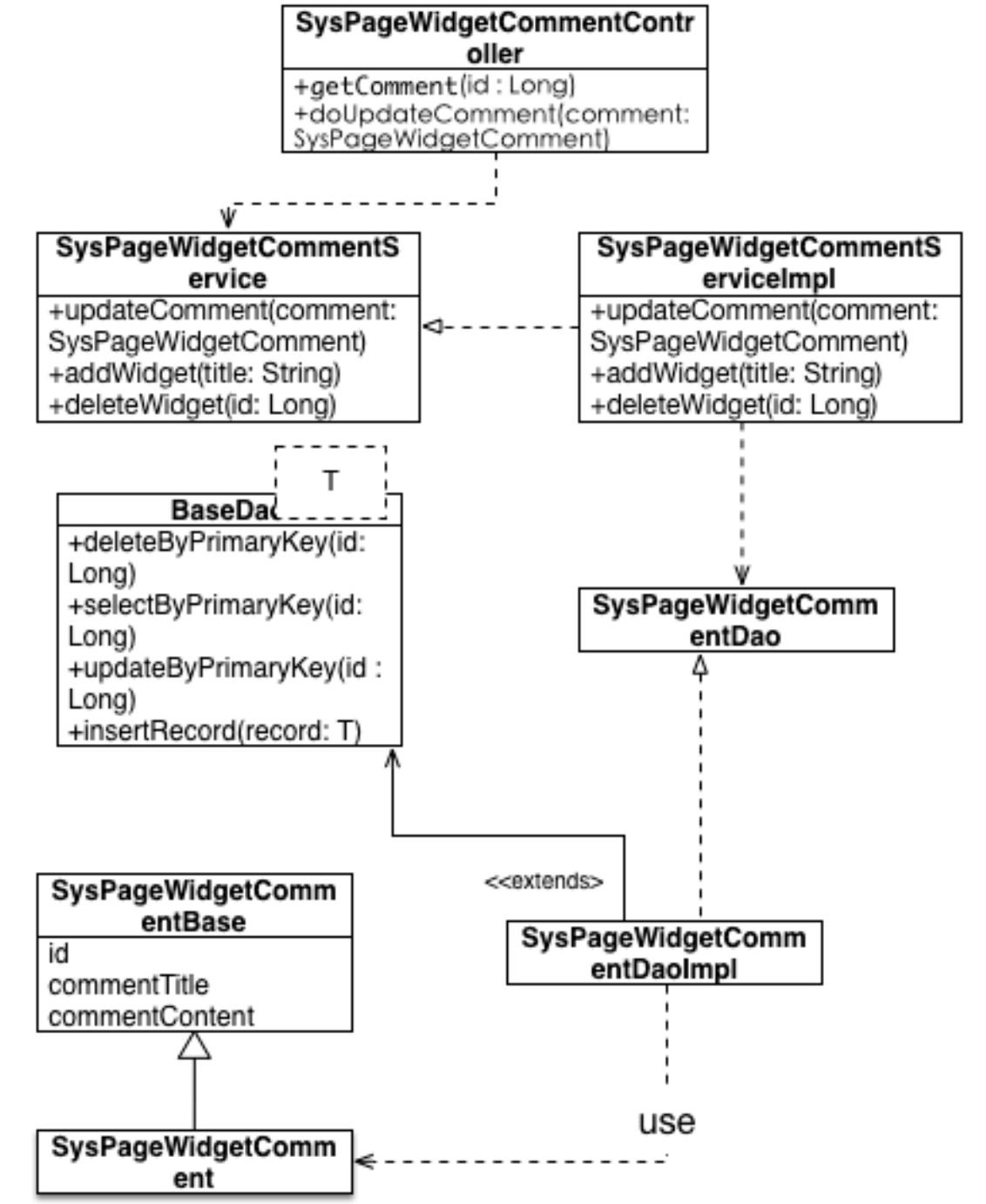

图 4.23 评论控件实现类图

(二) 表间联动

所谓表间联动，即为参数在不同控件之间的传递。某控件中的某一元素的动作触发了某一事件，该事件会影响页面中另外一个或多个控件的展现，这种影响即是通过参数在控件中间的参数传递实现的。

我们所关心的有两个方面：

- 参数传递的方向

● 参数形成的方式

(1) 参数传递流程图

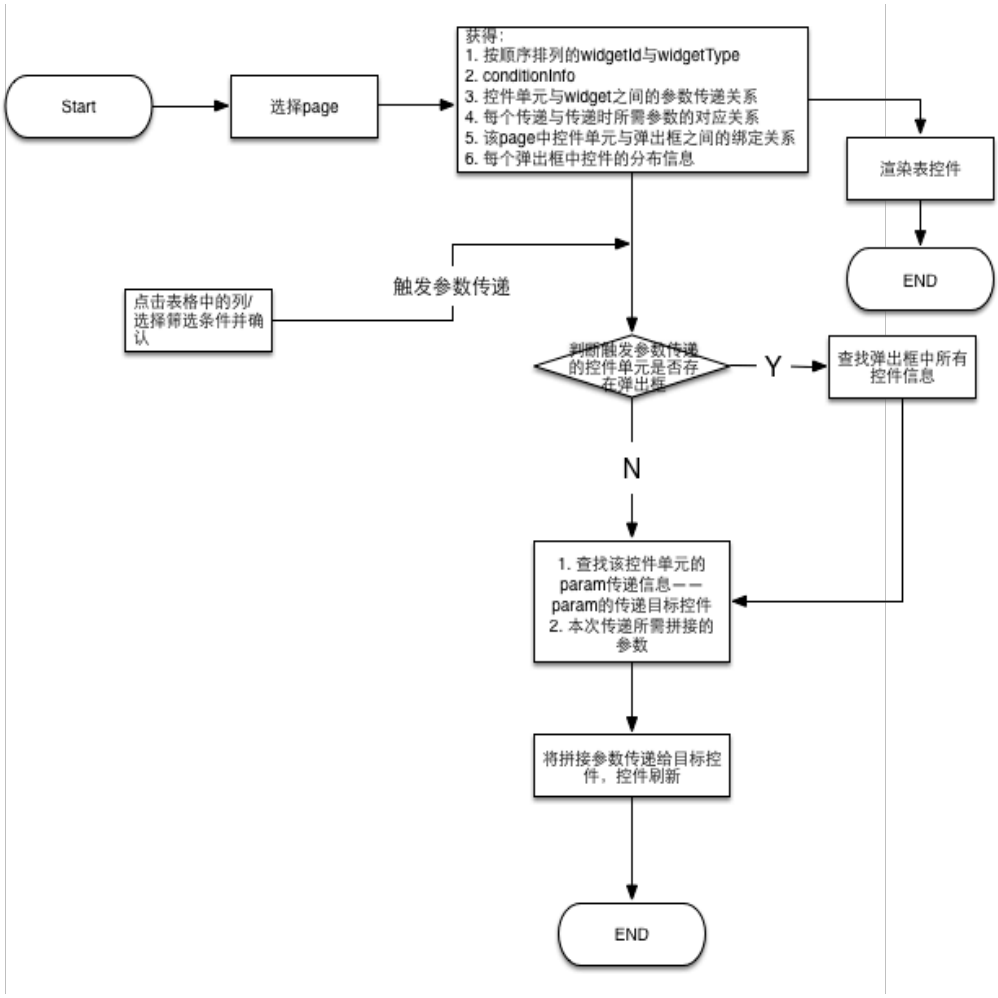

图 4.24 参数传递流程图

(2) 概念解释

控件单元： 如上文描述，将页面设定为表控件、图控件、条件区域控件等控件的组合。同时，我们又可以将控件本身划分为多个元素，其中能够触发参数传递的元素部分，称为控件单元。现有两种控件单元，一是表格中的列，二是条件区域中所有的筛选条件所组成的集合。

弹出框： 点击如上所说的“列”控件单元时，如果该列配置了弹出框，则会弹出响应的弹出框，弹出框中同样包含图表控件。该次点击会触发参数传递，该次传递同样也会传递给弹出框中的控件中。如图 4.3.3-18 所示。

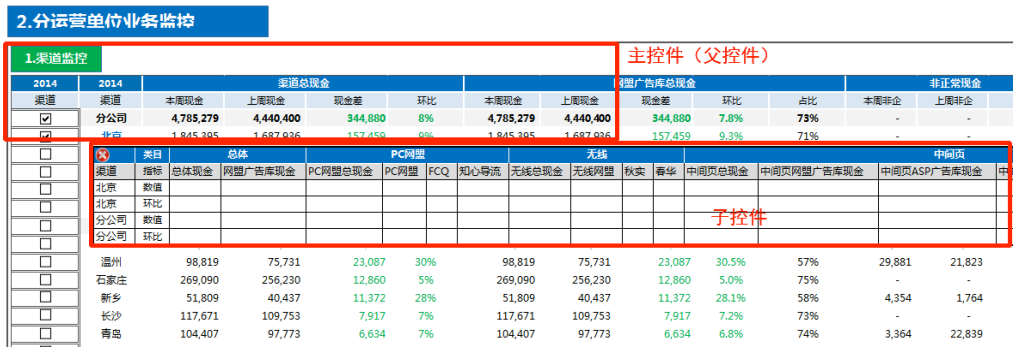

图 4.25 弹出框示意图

(3) 关键表结构描述

a. sys_page_widget_table_title_column 表

该表存储的为表控件的列标题信息，由于点击列标题会引发表间联动，因此可以将列标题看作为表控件的一个控件单元，该控件单元触发参数的传递。

b. sys_condition_info_area 表

该表存储筛选条件信息，由于筛选条件的选取确定后会引发表间联动，因此一个条件信息区域可以看作为表控件的一个控件单元，该控件单元触发参数的传递。

c. sys_param_transform 表 参数传递信息表

表 4.22 参数传递信息表

字段名	作用
id	主键标识
widget_item_id	控件单元 id
widget_item_type	控件单元的类型，1 表示列标题，2 表示条件筛选
passive_widget_id	接收参数的控件 Id
passive_widget_type	接收参数的控件类型，1 表示图控件，2 表示表控件，3 表示 comment 控件
page_id	表示该参数传递是在哪个 page 中发生

widget_item_id 和 widget_item_type 决定了触发联动的控件单元；

passive_widget_id 和 passive_widget_type 决定了接收联动参数的控件；这四个字段决定了参数传递的方向。参数传递可以是一对一，也可以是一对多。

d. sys_param_transform_item_relation 表

表 4.23 参数传递-控件单元关联表

字段名	作用
id	主键自增
param_transform_id	参数传递关系 Id
widget_item_id	控件单元 Id
widget_item_type	控件单元类型

sys_param_transform 表决定了参数传递的方向，只有方向是不够的，还需要将各种条件拼成参数。 该表决定了参数形成的方式。

一次参数传递可以对应多个控件单元当前的条件。通过参数传递关系 id，查找该传递中的参数所需要的控件单元条件，进行拼接，形成参数。

e. sys_window 表 弹出框信息表

需将弹出框作为一个区域，在其中设置它所包含的 widget

f. sys_window_widget_relation 弹出框控件关系表

表 4.24 弹出框-控件关联表

字段名	作用
id	主键标识
widget_id	控件 id
window_id	弹出框 id
widget_type	控件类型
display_order	控件展示顺序
is_hidden	是否隐藏

g. sys_widget_item_window_relation 控件单元与弹出框关系表

表 4.25 控件单元-弹出框关联表

字段名	作用
id	主键标识
widget_item_id	控件单元 id
widget_item_type	控件单元类型
window_id	弹出框 id

注：此处的 window 与 widget 会有两种关系：sys_window_widget_relation 和 sys_widget_item_window_relation

sys_window_widget_relation 存储弹出框与弹出框内控件的关系

sys_windget_item_window_realtion 存储控件单元与它所触发的弹出框的关系

第五章 总结与展望

5.1 总结

本文首先介绍了基于自定义 SQL 的报表的开发背景以及意义，阐述了在数据量大，表数据变化频繁，用户要求灵活配置的情况下，基于自定义 SQL 的报表相对于传统报表的优势。

然后对报表系统中所使用的前后端技术进行了简要的介绍，包括 Spring 框架，Spring MVC 以及前端请求在 Spring MVC 框架中的处理过程，前端渲染模板引擎 velocity，前端渲染图表库 ECharts。

随后文章对报表系统进行了需求分析，将系统拆分为系统配置和报表展现两大部分，并根据 PM 的需求将这两部分进行了更为细致的模块划分。

最后对系统进行了总体设计的阐述，并按需求分析中的模块进行了模块设计描述，包括各模块的包设计、类设计及表设计，模块中的部分重要算法，并对图控件和表控件传递给前端用于渲染的 json 数据进行了简要描述。

5.2 展望

该报表系统最初设定的最终目标为：按照设计输出相应格式周报，提供排查各环节所需核心关注的指标和数据，支持分析结论输入和前期分析结论查询、展现。当前的系统已经能够支持 web 端的所有配置、展现、查询、分析操作，但是将 web 端的图表转化为用于展示的 PPT 格式并未实现。

经过前期调查，目前能够操作 Microsoft Office 系列文件的工具较少，其中比较知名的工具是 Apache POI。Apache POI 是 Apache 软件基金会的开放源码函数库，POI 提供 API 给 Java 程序对 Microsoft Office 格式文档的读和写功能。因此，该报表系统在生成 PPT 格式的周报的功能上仍可基于之前的调查结果继续完善。

同时，由于灵活性与简便性是两个互相矛盾的要求，配置的灵活性必然带来了配置的复杂度，因此，在这两者的平衡中，本系统做的仍然较为欠缺，用户配置时的复杂度仍然较高。

除此之外，由于需求方对报表展现的需求较为紧急，项目实现时并未完整考虑对用户权限的限制。不同用户具有的权限是不一致的，各产品线的负责人应该只能看到自身负责的产品线报表数据，但部门经理具有查看所有报表的权限。因此，权限控制也是该系统需要考虑的部分。

第六章 致谢

经过在百度实习的若干个月，我的毕业设计终于圆满完成了。在整个报表系统的开发过程中，由于需求的不断变更，灵活性要求的不断提高，遇到了不少的问题和挫折，但是在学校导师、企业导师和同事的耐心指导和帮助下，我解决了这些问题。这次毕业设计使我认识到了软件开发过程中团队协作的魅力所在，团队成员共同努力，应对设计开发过程中不断出现的新问题，满足了用户对于灵活性的高要求。

感谢郑滔老师在百忙之中对论文的关心、指导和敦促，让我找到了论文写作的方向。

感谢百度（中国）有限公司提供了这次弥足珍贵的实习机会。

感谢企业导师和同事在项目开发过程中对我的包容、耐心和帮助，是你们的严格要求让我养成了独立思考的习惯，也是你们的耐心和包容让我不断成长与进步。

第七章 参考文献

[1] 维基百科. Microsoft Excel [G/OL] . :http://zh.wikipedia.org/wiki/Microsoft_Excel，2015/05/08

[2] Pivotal Software, Inc. Spring Framework Reference Documentation [EB/OL]. : http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/, 2015/4/25

[3] Walls C, HC/Datenkommunikation/Netze/Mailboxen. Spring in Action Fourth Edition[J]. 2014

[4] The Apache Software Foundation.Mybatis – User Guide [EB/OL]. :http://mybatis.github.io/mybatis-3/zh/index.html, 2015/5/4

[5] The Apache Software Foundation. Velocity Engine - User Guide [EB/OL].http://velocity.apache.org/engine/releases/velocity-1.7/us-er-guide.html, 2010.

[6] Baidu, Inc. Introduction[EB/OL].: http://ecomfe.github.io/zrender/doc/api/index.html, 2015/5/4

[7] Baidu Inc . Introduction[EB/OL]. : http://echarts.baidu.com/doc/doc.html，2015/4/25