摘要

随着互联网的广泛应用，信息管理系统的应用也越来越普及。在信息管理系统中使用频率最高的是报表部分。为了提升信息管理系统中报表需求的响应速度，同时规范整合平台内部各种业务子系统中的报表功能，就急需一个报表子系统来统一提供报表相应的功能接口。

本文将柔性报表理论和思想应用于基于 WEB 的报表定制过程中，归纳总结了其中的难点和存在问题，针对其中的难点和以往基于 WEB 报表工具的不足，提出一种新型的报表定制模式和流程，并利用基于 PHP 的 WEB 体系结构、FLEX 表格渲染以及 XML 技术对其进行了实现。

首先，针对目前基于 WEB 的报表定制的静态性和对需求变化适应性差的问题，将柔性理论应用于基于 WEB 的报表定制过程中，使其可以由用户通过界面交互进行制作——抛弃了以往由开发人员编码制作的方式，提高了报表系统的灵活性和适应性。

其次，规划了一种基于 WEB 的柔性报表制作模式和流程。通过数据源获取报表核心数据，通过数据整理获得报表最终的应用数据，通过数据渲染对报表的展示样式进行规范。解决了报表数据定制和样式定制等关键问题，实现了报表子系统中各部件的分离和整合。

最后，在 WEB 环境下，实现一整套关于报表各部件的设置界面和展示界面，实现了关于柔性报表的预期设想，完整并统一了目标平台上业务子系统中的报表功能。

关键字： 柔性报表子系统，数据源，数据整理，数据渲染

Abstract

With the widespread application of Internet, Information Management System applied increasingly popular. In the system, the most frequently used part is reporting requirements. In order to improve the response speed of reporting requirements, integrate the reporting capabilities of various business subsystems within the system, it urgently need a report subsystem to provide statements corresponding function.

This paper used the flexible statements theories and ideas with the WEB-based report customization process, summarized the difficulty and problems, against the difficulty and the lack of reporting tools existing proposed a new report customization models and processes, and achieved with the PHP-based WEB Architecture, FLEX form and XML technology.

First of all, against currently WEB-based report customization's static state and poor adaptability to changes in demand, it used the flexible theory in the WEB-based report's customization process, to finish all configuration by user self through the interactive interface ---- abandoned the mode by developer through the coding production in the past, Improved the reporting system's flexibility and suitability.

Secondly, it planned a WEB-based flexible report's production model and processes. Get the core data by data-source, obtain the final application data by data processing, sort out the report's style by rendering of the data. It resolved the data customization and style customization in a report, achieved the component's separate and merger in report subsystem.

Finally, in the WEB environment, it achieved the interface about the component's setting and display, achieved the anticipation and tentative plan about the flexible report, unify the reporting capabilities perfect on the target platform subsystem.

Keywords: Flexible Reporting Subsystem，Data Source，Data Processing，Rendering of the data

目录

第一章　绪论

在互联网以及各种信息系统被广泛应用的今天，软件即是服务（Software as a Service，SaaS）在线信息系统的服务提供商、开发者和维护者们（以下均简称为“开发者”）面临着一个严峻的挑战，那就是用户在应用系统的同时，会根据自身不同的工作特点和流程特色，提出局部乃至全局的定制化需求，即使 SaaS 系统已经根据用户群的行业差异、总体流程差异进行版本区分，也很难避免定制化的需求的产生。

而这对于开发者们来说，某种程度上是具有颠覆性的。这种定制化需求，不仅会破坏即有系统的整体性，使开发者们不得不在有可能的任意地方增加分支或逻辑，用以调整程序的流程或算法；更重要的是上述的变更对于系统的稳定性和可维护性没有任何正面的意义，反而使代码变得复杂，结构变得臃肿，流程和算法的分支趋于混乱乃至整体逻辑产生矛盾，被定制部分功能的算法重用性和唯一性也不可避免的丧失殆尽，这些都为系统的最终崩溃提供了不可忽视的内因。

而系统的各种组件中，上述问题在报表部分的体现尤为明显。因为任何业务部分的定制变化，最终结果需要反映到报表统计结果上，这种业务变更引发的定制都不可能是常态的、经常性的。而且报表部分自身的定制更是多种多样，报表数据汇总的侧重点、报表数据渲染展示的重点等，这些都是会根据 SaaS 系统用户的理念或着眼点的变更而进行定制修改的。

所以，SaaS 系统的开发者们，需要一整套强壮的 SaaS 报表程序或子系统（以下简称“报表子系统”），用以应对日趋严峻的定制需求，尽可能的简化或者避免定制化需求的开发工作。

第一节　论文工作的背景

报表作为信息系统中不可或缺的组成部分，在上述这些被定制的毒害中是首当其冲的。不仅要满足业务部分定制功能在报表部分的数据体现，更有甚者，针对报表的数据范围、数据汇总、显示样式、显示范围等都可能存在精细化的

定制要求。这就使得系统中应用可重用的纯粹的过程化处理的报表程序不堪重负，开发者更是不堪其扰。

下面是几个 SaaS 进销存业务子系统（以下简称为“进销存子系统”）中的现役报表例子，通过定制化的前后对比，就不难看出报表定制工作的琐碎和繁复了。

例子 1：进销存子系统提供了产品库存情况表，用以展示产品的账面库存量以及可用库存量等库存信息，其结构如表 1.1 所示。

表 1.1 产品库存情况表结构

产品	仓库	库存量	可用库存量
可口可乐	南开库	1000	850
百事可乐	南开库	800	750

续例子 1：由于某用户的需求，作为“基础数据”的产品，需增加一个名为“规格”的描述属性，将产品进行进一步的细分，并且该属性需要在报表中进行体现，即以产品库存情况表为例，其结构需要变更为如表 1.2 所示。

表 1.2 产品库存情况表变更后结构

产品 ID	产品	规格	仓库	库存量	可用库存量
1501	可口可乐	1250ml	南开库	550	500
1502	可口可乐	2000ml	南开库	450	350
1601	百事可乐	1250ml	南开库	800	750

续例子 1：由此而引发的报表修正工作量将是恐怖的，因为新属性“规格”的出现，颠覆了所有涉及“基础数据”产品的报表的“列信息”设置和“数据获取”。又因为新属性“规格”并非大多数用户的“基础数据”产品的必要属性，所以无法将该属性增加到公共的报表之中，也就是出现了上面说到的“定制”情况，而且是为该用户定制几乎所有涉及“基础数据”产品的报表。

例子 2：进销存子系统提供了经销商信息表，用以展示“基础数据”经销商的相关数据，其结构如表 1.3 所示。

表 1.3 经销商信息表结构

经销商	联系人	电话	e-Mail	累计交易额
精工超市	张三	23132568	zhangsan@163.com	￥30,000

续例子 2：由于某用户的需求，所有经销商在“经销商信息表”中需分表显示，要求存在三张“经销商信息表”分别对应“大客户”、“一般客户”和“小

客户”，用以对应不同业务员的浏览权限。其中“大客户”的定义为累计交易额达到或超过 200 万元；“一般客户”的定义为累计交易额达到或超过 10 万元且未达到 200 万元；“小客户”的定义为累计交易额未达到 10 万元。

由此又带来了报表的定制需求。虽然所需的工作不算复杂，但是由于“累计交易额”这种非常规的数据范围设置，使得其中相对固定的边界值“10 万”以及“200 万”只能存在于过程化的处理中，而无法也不应该存在于实时交互的查询条件中。又因为这种固定边界值的多样性和可变性，造成进销存子系统中可能存在大量的同类报表，使得报表的可维护性降低。

例子 3：进销存子系统提供产品销售情况统计表，用以展示一定时期内产品的成本、销售以及利润情况，其结构如表 1.4 所示。报表中“毛利”列的计算逻辑为既定逻辑。此处未给出计算结果意为宏观上存在着不同的计算逻辑，无法统一计算。

表 1.4 产品销售情况统计表结构

产品	成本单价	销售单价	销售量	销售总成本	销售总价	实结总价	毛利
饼干	￥2.30	￥3.50	100	￥230.00	￥350.00	￥347.50	??

续例子 3：由于进销存子系统提供的默认成本计算方法为“移动加权平均法”，即产品的成本会根据进货和销售情况产生波动，所以造成表 1.4 中的“成本总价”并非是用户想象中的单纯的“成本总价＝成本单价×销售量”。

另外，表 1.4 中“毛利”的计算，在用户中其实也存在差异。有些用户应用“毛利＝销售总价－成本总价”；有些用户应用“毛利＝实结总价－成本总价”；甚至有些用户应用“毛利＝税后总价－成本总价”，需要再增加“税后总价”一列——以“销售总价”为基础，应用固定比率税率（通常为 17%）进行计算。这些对于报表数据内在逻辑的多样性理解和要求，对于重用的报表流程来说，影响很大。

上面三个例子所提到的报表，在进销存子系统中都是属于常用和基础的，这样都会存在如此之多的定制情况和可能，就更不必说个性化较强的全定制报表了。

在系统的日常运行维护中，类似的需求有很多，也就耗费了开发者大量的时间和精力，同时使系统本身变得日益臃肿、庞大——单就报表部分而言，情况更加明显。基于此，规划开发一个用以应对多变的需求，灵活强壮，易于使用便于操作的报表子系统，已经是当务之急。

第二节　论文工作的总述

由于上述所需的报表子系统要求在不同 SaaS 业务子系统中的可通用性强，以及进销存子系统的业务需求和报表需求更为典型和多样，所以报表子系统的规划和开发将以进销存子系统为目标环境进行。开发完成后，再逐步将 SaaS 系统中其他的业务子系统的报表功能转交至报表子系统进行。

论文的主要工作源于报表子系统的规划、实现和运维等各生命周期的过程。也忠实的记录了在各个生命周期里报表子系统的状态，期间有典型或非典型的经验和教训，以及由种种原因产生的惊喜和无奈。

具体工作的整体流程循规蹈矩，其中的细节部分不乏亮点。

1.2.1　报表子系统的规划和设计

对于报表子系统的规划，一直伴随着业务子系统的规划、开发和运行。最初的纯粹过程型的报表，无法很好的应对功能的扩展和需求的骤增，所以将报表数据与显示样式分离的方案被正式提出。

考虑到大量的数据定制和流程定制，为了报表数据部分的代码可重用，所以将报表数据部分进一步细分成为数据源和数据整理两大模块。将数据获取部分的功能独立成为数据源模块，再将其中对于进销存子系统可重用的数据获取部分独立成为基础数据源子模块。将所有的数据定制和流程定制规划到数据整理模块中，完成从原始数据到最终展示数据转变的功能。在明确约定模块间接口结构后，进一步在各模块内部进行更具针对性的局部设计。

同时考虑到报表的柔性需求，报表子系统还规划了与报表业务相配套的图形界面，提供给开发者甚至有权限的用户，对报表各部分进行设置和细部调整。

在报表子系统规划和设计期间同时参考了大量的同类系统。其中包括清华大学曹军威、范玉顺和吴澄的新一代 CIMS 应用集成平台系统体系结构[1]；南京航空航天大学方叙生和沈平的柔性化 MIS 系统的设计与开发[2]；王元珍和汪皓的达梦智能报表工具[3]；万琳和陈传波的智能报表系统[4]；此外还包括一些国内外的研究成果[5-13]。

详细的设计将在后面的章节进行说明。

1.2.2 报表子系统的编码和测试

具体到报表子系统的编码开发环节同样存在着细致的流程设计和算法设置，更存在与进销存子系统对接时出现的业务情况和业务需求。

报表中金额精度的控制问题；报表中计算公式的失控复杂度最优问题；报表子系统和进销存子系统之间的权责划分问题，以及后续出现的相互妥协。这些都是开发者需要谨慎考虑的。

至于报表子系统的测试，主要分为三个部分。一是在完成报表子系统各部分模块原型之后的原型测试；二是在报表子系统各部分模块逐步扩展开发完成时相应的功能测试；三是针对完整的报表子系统的综合测试，其中包括了报表子系统与进销存子系统之间的整合测试。

1.2.3 报表子系统的运行和扩展

计划总是跟不上变化，也就是说哪怕再完美的设计，依然会随着时间的推移而产生瑕疵，甚至颠覆性的需求变更。这是一个不争的事实，甚至是惯例。报表子系统同样面临着这样的命运，在经过了运行平稳期之后，设计时不可预期的情况或需求仍然会出现，对于程序的调整和扩展是不可避免的。

对于逻辑简单的问题可以通过对报表进行简单的维护或调整加以解决；对于一些性能上的问题可以对算法和数据结构进行进一步的重构和优化[14-15]；对于复杂逻辑的报表需求，在无法通过调整报表来满足时，就只能对报表子系统的功能进行扩充和完整。

上述的若干种情况在报表子系统上线运行维护的过程中都遇到过典型的需求。

1.2.4 报表子系统的总结和展望

报表子系统通过了平稳期的运行和各种需求的洗礼，笔者也不断地总结着该子系统的优点和不足，在不断的积累中等待着质变产生的契机。

报表显示样式的多样化是未来发展的必然；数据源和报表显示的彻底分割将是未来发展的方向；报表子系统之于业务子系统可以承担更多的工作，为业务子系统的开发和维护提供了强有力的支持。

第三节 论文工作的主要成果

报表子系统的最终建立、平稳过度和运行，圆满的完成了既定的任务目标。

期间，为解决报表生成器生成文件问题，开发了基于 PHP 中 file-system（文件系统）相关函数[16]的文件生成器文件基础操作函数类。其间针对服务器用户和其他用户对于文件读写权限进行了区别限制。

为解决报表页面计算公式反复调用问题，开发了批量计算公式插件，取代了原先的简单的计算公式运算模式。通过 PHP 中的 create_function 方法，首先对计算公式进行初始化解析优化重组，生成一个临时的可应用函数，在一张报表中反复使用同一个计算公式时，就可以应用统一的函数调用，完成运算。这个批量计算公式插件在大规模同类运算请求中将发挥显著的作用，为缩减开销提供帮助。

为解决报表子系统的相关设置的图形界面交互，开发了应用 EXTJS 的一整套设置界面，其中对于报表数据整理和渲染的功能进行了全面细致的支持，为报表柔性的可用性和可参与性做出了贡献。

为解决报表显示限制而扩展的 FLASH 表格控件，取代了原先的纯粹 HTML 的创建 DOM 元素进行展示的模式。FLASH 表格控件同样可以成为公共控件用于其它诸如单据列表的系统功能页面。

为解决报表导出限制而开发的基于 PHPEXCEL 插件的中间处理程序，取代了原先的根据 HTML 标签构建.CVS 格式文件的模式。对于 PHPEXCEL 插件的研究和中间处理程序的再封装，对于后续的数据导出类功能有着极大的借鉴意义，并提供良好的中间调用平台。

第四节 论文组成和各部分内容

本文的第二章是针对报表子系统的需求分析内容。从系统的总体概况、系统的业务需求、功能需求和性能需求等方面进行了阐述。确定了系统的功能范围和大体的角色分配，为系统的设计打下坚实的基础。

本文的第三章是报表子系统的总体设计。介绍了系统运行的硬件环境和网络环境以及系统的开发环境，从整体的角度阐述了报表子系统的功能需求并分模块进行了描述。

本文的第四章是报表子系统的数据库设计。其中考虑了子系统各个功能模块的特点和限制，对于各部分的数据库表结构进行了列举。

本文的第五章是报表子系统的详细设计和实现。依据第三章中的模块划分，对各个模块的功能进行了细致的阐述和限定以及具体实现。

本文的第六章是针对报表子系统中各个关键问题以及相应解决方案的介绍。其中涉及了数据源、数据整理和数据渲染等各个部分的核心算法和解决方案。

本文的第七章是关于报表子系统的总结和展望。对于目前存在的问题提出一个解决的思路，并对未来发展的趋势进行预估并提出应对的措施。

第二章　系统需求分析

系统的需求分析，是在确定系统需求之后，以实现系统目标为目的，通过对系统的功能分解，逐步明确各功能模块职能范围和交互接口，最终确定系统的开发路线和解决方案。

第一节　系统概述

报表子系统主要对进销存、客户关系管理、财务管理、物流管理以及协同办公等业务子系统的数据整合和显示提供支持。

原有报表模式，属于原始的功能性报表处理流程，是没有一个整体系统支撑的，是简单的基于后台数据库进行 SQL 查询，得到结果后进行相应的数据调整，最后输出显示。每张报表一个过程型文件，各个报表散落在平台的相应业务子系统中，没有一个统一高度的认识和处理，这样每一张不同的报表都需要进行大量重复代码编写，造成报表开发的效率偏低，周期过长，维护成本大的现状[17-18]。

第二节　系统业务总体描述

就整体而言，报表子系统所涉及的业务并不很复杂，至少相较于其它业务子系统来说，没有繁杂的工作流和个性化的业务规则。根据长时间的需求积累和客户需求调研，报表子系统的主要功能大项分为“系统管理”、“细部调整”和“报表展示”，如图 2.1 所示。

“系统管理”大项，用于统一管理报表子系统在客户业务子系统中的应用范围、权限和初始设置。“报表信息维护”主要用于维护客户业务子系统开启报表的范围，以及所开启报表的基础信息。“浏览权限分配”主要依据客户业务子系统的各个用户角色的不同，对其可应用的报表范围进行限定分配。“固定参数设置”主要用于报表子系统应用于客户业务子系统是所涉及的公共参数的维护。例如：客户业务子系统中报表的分组情况；报表数据导出的文件默认格式；客

户业务子系统中报表页面打开的默认方式等。

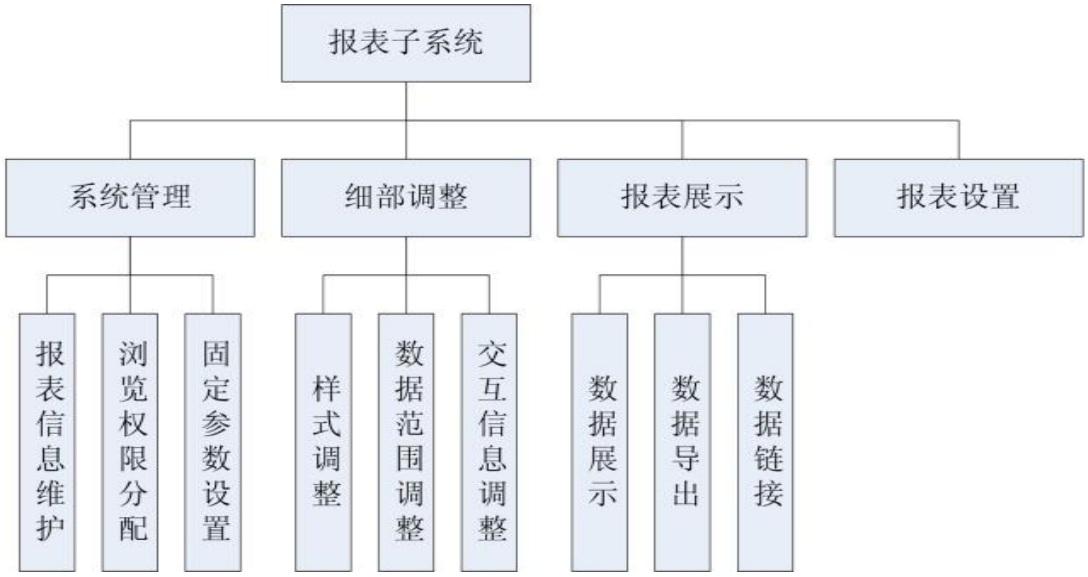

图 2.1 报表子系统功能结构图

“细部调整”大项，用于客户根据业务子系统的应用情况，报表浏览的习惯以及客户自身的人事关系、规章制度、业务流程，针对每张启用报表进行适应性的调整。“样式调整”可以变更报表中的数据显示颜色、字体字号、数字精度、数字千分位、金额符号等页面显示样式。“数据范围调整”可以变更客户业务子系统中的不同角色用户在报表中可见的信息范围，这对于客户的数据安全和保密性有着重要意义。“交互信息调整”可以维护报表浏览时报表子系统提供的实时交互的查询条件的范围和查询类型。

“报表展示”大项，用于客户业务子系统中可见报表的展示和应用。“数据展示”主要包括了报表列表展示页和报表数据展示页，是报表浏览的主要入口，是报表子系统在业务子系统中最主要的嵌入式应用。“数据导出”会提供.XLS 和.XLSX 格式文件的导出，并且兼容旧版中根据 HTML 标签构建.CVS 格式文件的导出模式。“数据链接”是指报表中应用的业务子系统中的基础数据向业务子系统中该基础数据展示的相关页面跳转的链接，数据链接的存在方便了客户在浏览报表时在相应数据间的切换，当然这个链接是客户通过报表子系统提供的接口注入到客户业务子系统应用的报表中的。

“报表设置”大项，就是一张报表从需求的逻辑到报表子系统中真实存在的整个演变发展的过程。其实报表设置是一个并不为客户所知的部分，因为无

论是从客户的专业背景还是报表需求的高度的逻辑抽象出发，报表子系统都是不能将报表的核心设置完全开放给客户应用的。但从报表子系统完整性的角度看，报表设置又是绝对不可或缺的一部分。鉴于此项对于客户的认知完全透明，所以具体的设计说明本文的将在第五章的第五章第四节中进行详细的阐述。

第三节　系统各主要业务和流程描述

按照本章第二节中关于系统总体业务的描述，下面将对报表子系统的主要业务流程进行细致的描述。

报表子系统的应用是属于其它业务子系统中的嵌入应用，子系统本身更像一个整理数据逻辑的工具或者提供数据的服务。从业务子系统的角度来看的话，如图 2.2 所示，一张报表从需求的提出到最终的废弃，期间存在若干状态，而这也可以视为单一报表的生命周期。

一张全新的报表，总是由一个报表需求开始的，这个需求可能是客户提出的，也可能是开发者根据业务子系统的功能流程的变化而破旧立新得来的，总之这是一个仅仅停留在逻辑层面的状态。

对于客户业务子系统来说，经过一个“黑盒”的报表设置过程之后，在报表子系统模板库中生成了一张和所提需求对应的备选报表。需要强调的是，此时的报表是位于报表子系统的模板库中，无论该报表的需求是一般性通用需求还是客户个性化定制需求。

当报表存在于报表子系统的模板库中时，客户可以通过报表子系统提供的接口，将该报表复制到业务子系统中以备使用。需要说明的是，此时的报表存在于业务子系统中，是与报表子系统模板库中的报表区别存在的，所以此时所维护的报表基础信息是不会影响到报表子系统中的模板报表的，同样此时报表子系统中的模板报表如果发生改变，也不会对业务子系统报表现状造成影响，除非重新从模板库进行复制。当然，客户可以复制多张相同的报表，进行不同的设定以用于不同的用户角色进行浏览。

通过权限分配，业务子系统中的报表进入到“可用”状态，此时的报表已经可以在页面进行查询导出等操作，也可以针对使用中的一些问题对报表的细部进行调整。

当客户不再需要这张报表时，可以进行删除，该报表将会从业务子系统中

彻底清除，报表的生命周期也同时终止。此处被删除的是业务子系统中的报表，而非报表子系统模板库中的模板报表，也就是说，当客户重新需要的时候，可以从报表模板库中重新复制一张全新的报表重新开启报表的生命周期，但是新复制的报表已经不是原先删除的那张报表了。

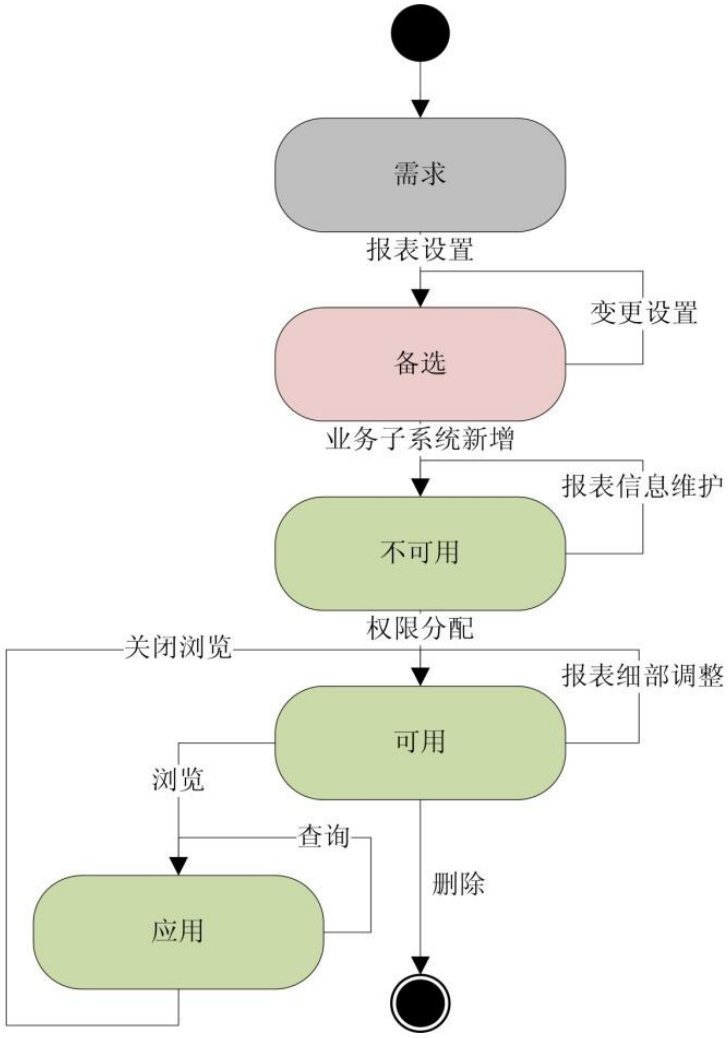

图 2.2　业务子系统报表状态图

第四节　系统功能要求描述

从客户应用的角度来说，希望业务子系统中的报表能够更灵活简便的应用，是最直接的要求，至于报表子系统与诸多业务子系统之间的通用性和协调性，是开发者需要规划和设计的。所以报表子系统中最先要讨论的报表子系统的参

与者和报表子系统在业务子系统中的功能范围。

2.4.1 系统参与者

按照报表子系统的初步规划和设计，报表子系统的参与者主要有系统管理员、高级用户和一般用户三种角色，如图 2.3 所示。由于报表子系统和业务子系统的特殊关系和功能划分，报表设置部分是独立于业务子系统的，而其他部分则属于报表子系统功能在业务子系统中的潜入应用，总的来说都是报表子系统的功能，但拥有不同的入口。

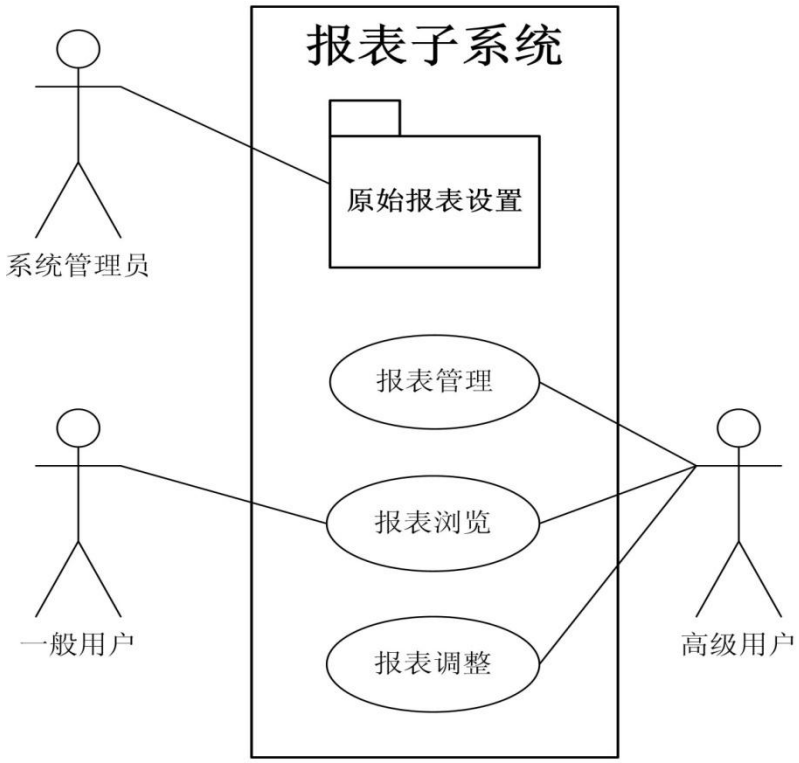

图 2.3 系统简单用例图

2.4.1.1 系统管理员

系统管理员的职责范围局限于报表子系统的独立核心部分，通过报表设置的一系列操作，将报表的需求转化成为真实存在的报表。主要针对报表子系统模板部分进行操作和维护，要严格遵守不干预业务子系统任何操作的原则。同样，这个角色对于客户来说是不可见的。

2.4.1.2 一般用户

一般用户仅能够针对业务子系统中有权限的报表进行浏览、查询以及导出等简单操作。因为这种角色的用户，在业务子系统中的业务职责本身大都存在局限性，不允许其对报表的数据范围等约束和参数进行控制。一旦可以自由设置报表的数据范围，那就意味着该用户可以看到所有范围的数据。

2.4.1.3 高级用户

相对于一般用户而言，高级用户的权限就大了很多，可以说在业务子系统中所涉及到的报表相关的操作，高级用户都可以进行调整。可以为当前业务子系统增加新报表，删除废弃的旧报表；为一般用户指定可浏览的报表范围，以及各个报表中的数据的范围；可以针对每张报表进行调整，增加自定义的数据汇总和简单逻辑运算。

2.4.2 系统功能需求

报表子系统的功能需求并不算复杂，在本章 2.1 节中也做过一定的阐述。这里要着重说的是报表的格式和样式需求。

针对目前的表格型报表而言，最主要的格式集中在行和列这两个元素上，而样式将会以单元格为载体进行体现。

做好以下三个方面，报表子系统在业务子系统开放设置的部分就基本完成了。

2.4.2.1 列元素的相应格式

相同的报表，因为不同的列格式，很可能形成不同的感官体验，并能够在一定程度上对于数据产生针对性的体现。下面依然就一张进销存子系统中的现役报表进行举例。

进销存子系统提供“产品规格库存统计”报表，顾名思义该报表的主要信息为“产品”、“规格（此处以尺码指代，应用于服装鞋帽等产品的数据汇总）”和相应的“库存量”，是一张很简单的报表。

一般格式的报表如表 2.1 所示，很明了的表格结构，美中不足的是，当数据量成长到一定规模时，要进行数据的搜索难度将会大幅度的增加，虽然可以使用实时交互进行数据筛选，但在需求大量数据是仍然会显得力不从心。所以

对于报表的格式会进行一定的修改。

表 2.1 产品规格库存统计（一般格式）

产品	尺码	库存量
T 恤	XXL	100
T 恤	XL	56
鞋子	42	32

一维横向化格式的报表如表 2.2 所示，将原先的“规格”一列进行一维展开，这样就得到了一张可以以“产品”为主索引的报表，这对于检索产品获取数据的效率将会大大提高。然而在实际情况中，仍然会存在问题，可以进行改进。因为不同种类的产品会应用不同种类的“尺码”，就好像服装要应用专用的尺码，包括 L、XL、XXL 等；鞋子要应用专用的尺码，包括 40、41、42 等，就造成了每一行数据中只有几个特定的“尺码列”存在有效数据，而其它的列全部为空。所以对于报表的格式会进行进一步的修改。

表 2.2 产品规格库存统计（一维展开格式）

产品	尺码			
	XXL	XL	41	42
T 恤	100	56		
鞋子				32

二维横向化格式的报表如表 2.3 所示，将原先的一维展开“规格”进一步进行二维的归类展开，这样就可以有效地压缩报表的列数而不用担心影响数据的统计。当然这里的“尺码”由于存在可抽象的“尺码类别”，才可以进行二维展开，如果换做是“颜色”的话，就另当别论了。

表 2.3 产品规格库存统计（二维展开格式）

产品	尺码		
	服装	XXL	XL
	鞋子	41	42
T 恤	服装	100	56
鞋子	鞋子		32

上面就同一张报表的不同格式结构进行了举例说明。而更复杂的报表还有很多，就上述“产品规格库存统计”而言，如果所涉及的规格增加一个“颜色”的话，如果在将“库存量”进行分库统计的话，报表的结构复杂程度将直线上

升。而由于受二维表格表现形式的约束，也很难再出现更为复杂的列元素结构了。

2.4.2.2 行元素的相应格式

上面提到了“产品规格库存统计”的复杂结构，而这用于描述行元素格式再合适不过了。

小计格式的报表如表 2.4 所示，报表中同时存在了“合计”、“产品小计”、“仓库小计”和“尺码小计”四重行汇总统计，而这中间必须遵守一个严格的逻辑级差约束。也就是说，如果“仓库小计”包含于“产品小计”，那么“仓库”的汇总就要受当前“产品”的约束，其它“产品”也还会存在其它产品的“仓库”汇总。

表 2.4 产品规格库存统计（小计格式）

产品	颜色	尺码	仓库	库存量
T 恤	红色	XXL	南开库	36
T 恤	蓝色	XXL	南开库	58
T 恤		XXL 小计		94
T 恤	红色	XL	南开库	125
T 恤	蓝色	XL	南开库	61
T 恤		XL 小计		186
T 恤			南开库小计	280
T 恤小计				280
合计				280

2.4.2.3 单元格的样式需求

单元格的样式多种多样，报表子系统中的应用到的，主要有“布局样式”、“文本样式”和“特种样式”三种。以下分情况进行简要的说明。

布局样式主要涉及单元格中的文本水平、垂直对齐方式，单元格背景色，单元格边框等。

文本样式主要涉及文本的字体、字号、粗体、斜体等。

特种演示主要针对数据类型的数据，涉及数据显示精度、千分位、货币符号等。另外，时间类型数据的现实精度也在此列。

第五节　系统性能要求描述

报表子系统所提供的报表是业务子系统唯一的汇总统计数据，因此更需要保证数据正确性之外的一些性能要求指标。

2.5.1　报表展示的响应速度

报表展示的响应速度包括页面的效应速度和报表数据的加载速度。其中页面的响应速度要求不高于 3 秒，支持不少于 100 名用户的并发访问。报表数据的加载速度是数据量情况而定。总的加载时间，普通报表要求不超过 10 秒，大数据量报表要求不超过 30 秒。

2.5.2　报表展示的浏览通用性

由于 SaaS 的特性，是以 Internet 为载体，以多种通讯方式（浏览器、e-Mail、手机短信、手机 WAP)为人机交互的主要媒介。所以要求报表子系统的报表展示页面对于各种终端设备都要有相应的展示方案。

2.5.3　报表子系统提供的数据精度

报表数据的最终来源仍然是业务子系统的数据库，所以报表子系统对于数据能够提供的精度极限，是受限于业务子系统的。显示精度可以自定设定，即数据中的数字所保留的精度可以在有效地数据库存储精度的范围内自定确定。

2.5.4　报表子系统细部调整的易用性

对于一般用户要求可以直接使用浏览功能；对于高级用户的报表维护功能要求界面简单，功能明确，易学易用；对于报表设定部分因为复杂度高，会涉及数据库相关知识，还要兼顾报表内部分的业务流程，所以暂时不会开放给客户进行操作。

2.5.5 报表子系统报表需求响应速度

开发者们比较关心的是对报表需求的响应复杂度，希望能够从机械的重复工作中解脱出来。报表子系统的存在，要求对可支持的报表逻辑能够高效快速的完成报表从需求到模板的过程，并且维护简单。

第三章　系统总体设计

报表子系统的总体设计还要从报表设置和报表生成的核心开始说起。在介绍总体设置之前，要先就报表子系统的开发环境和预期运行环境进行说明。只有明确了系统的开发环境和运行环境，才能在设计和实现的过程中选用最有针对性的数据结构和算法结构。

第一节　系统环境平台

由于报表子系统是 SaaS 系列产品的一部分，所以他的运行环境和相关的业务子系统没有差别，也不可能单独维护一套环境进行运行。SaaS 系列产品的网络平台系统架构如图 3.1 所示。

3.1.1　跨网性

SaaS 产品是以 Internet 为载体，以多种通讯方式（浏览器、E-mail、手机短信、手机 WAP）为人机交互的主要媒介，适合于企业分散经营、集中管理特点的一套完整的面向企业的网络化管理解决方案。

3.1.2　安全性

随着“云”的逐步推广和深入应用，网络数据的安全性问题也成为网络信息管理系统要考虑的重中之重。在系统架构中引入的实时备份数据库服务器，可以及时有效的进行数据备份防止数据丢失。之于防范外部病毒或恶意程序的攻击，除了构建强化防火墙，还没有更好的解决方案。

3.1.3　高性能

系统的构建是符合高性能系统的构造特点的。整体分为“WEB 前端系统”、“负载均衡系统”、“数据库集群系统”、“缓存系统”、“分布式服务器管理系统”以及“代码分发系统”。是整个系统实现高负载、高数据交互、高数据流动性。

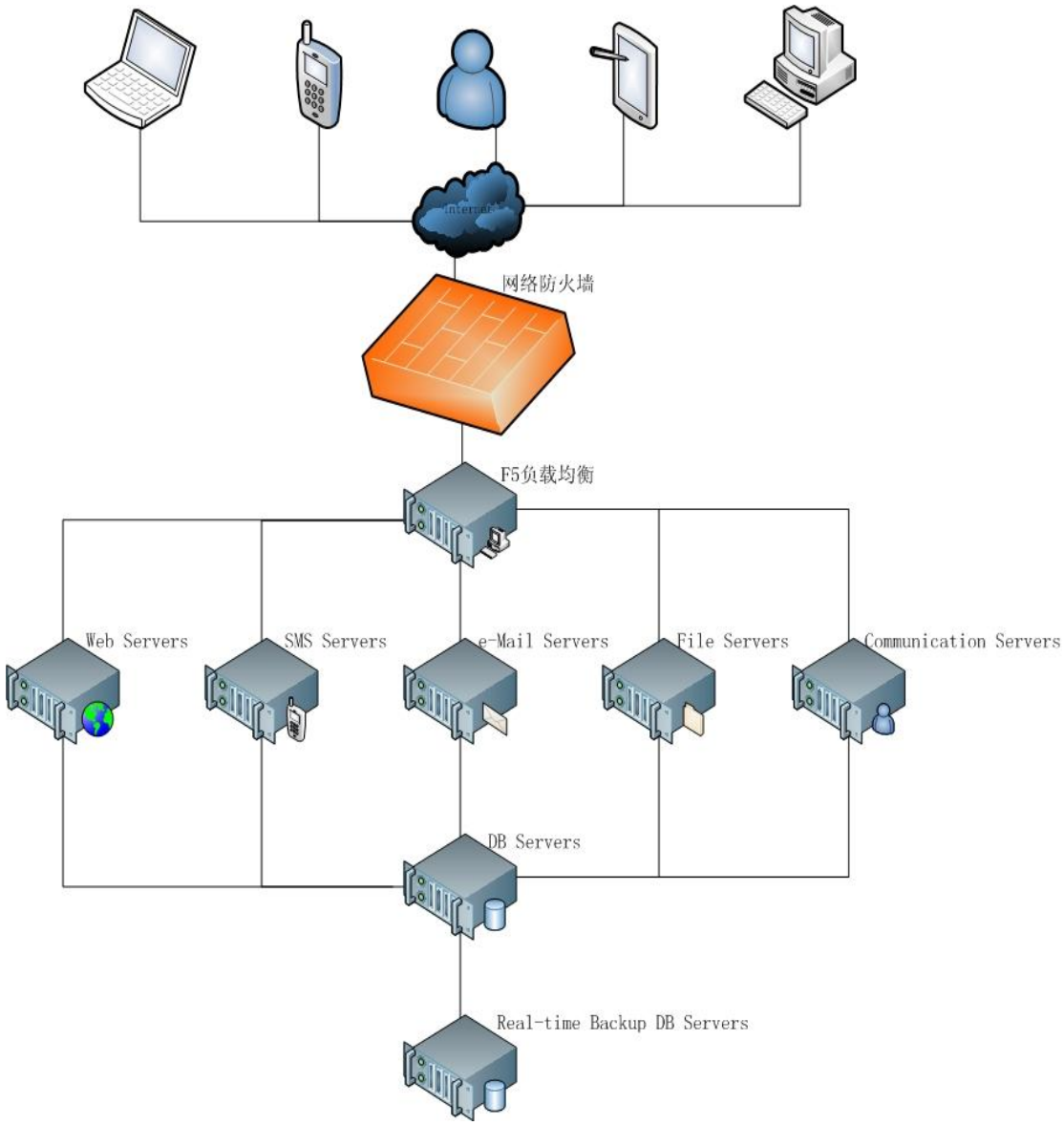

图 3.1 系统架构图

第二节 系统开发技术和工具

目前 Internet 上流行的网站构架方式是 LAMP（Linux + Apache + MySQL + PHP/Perl/Python）和 LNMP（Linux + Nginx + MySQL + php / perl / Python），即使用 Linux 作为操作系统，Apache 和 Nginx 作为 Web 服务器，MySQL 作为数据库，PHP/Perl/Python 作为服务器端脚本解释器。由于这四个软件都是免费或开放源码软件，因此使用这种方式不用花一分钱（除开人工成本）就可以建立

起一个稳定、免费的网站系统。

报表子系统的开发环境和技术主要是沿用了遗留系统，属于上述的 LAMP 模式，具体配置为“Linux + Apache + MySQL + PHP”，在这里就简单做一下介绍。

3.2.1 Linux 操作系统

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

这个系统是由全世界各地的成千上万的程序员设计和实现的。其目的是建立不受任何商品化软件的版权制约的、全世界都能自由使用的 Unix 兼容产品。

Linux 以它的高效性和灵活性著称，Linux 模块化的设计结构，使得它既能在价格昂贵的工作站上运行，也能够在廉价的 PC 机上实现全部的 Unix 特性，具有多任务、多用户的能力[19]。

3.2.2 Apache WEB 服务器

Apache HTTP Server（简称 Apache）是 Apache 软件基金会的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的 Web 服务器端软件之一。它快速、可靠并且可通过简单的 API 扩展，将 Perl/Python 等解释器编译到服务器中。

Apache HTTP Server 是世界使用排名第一的 Web 服务器软件。它可以运行在几乎所有广泛使用的计算机平台上。Apache 的特点是简单、速度快、性能稳定，并可做代理服务器来使用。本来它只用于小型或试验 Internet 网络，后来逐步扩充到各种 Unix 系统中，尤其对 Linux 的支持相当完美[20]。

3.2.3 PHP 开发语言

PHP，是英文超文本预处理语言 Hypertext Preprocessor 的缩写。PHP 是一种 HTML 内嵌式的语言，是一种在服务器端执行的嵌入 HTML 文档的脚本语

言，语言的风格有类似于 C 语言，被广泛地运用。

PHP5 的核心是第二代 Zend 引擎，并引入了对全新的 PECL 模块的支持。PHP5 的最大特点是引入了面向对象的全部机制，并且保留了向下的兼容性。程序员不必再编写缺乏功能性的类，并且能够以多种方法实现类的保护。另外，在对象的集成等方面也不再存在问题。使用 PHP5 引进了类型提示和异常处理机制，能更有效的处理和避免错误的发生。

随着 MySQL 数据库的发展，PHP5 还绑定了新的 MySQLi 扩展模块，它提供了一些更加有效的方法和实用工具用于处理数据库操作。这些方法大都以面向对象的方式实现，同时也极大地提高了基于数据库的 Web 项目的执行速度[21]。

3.2.4 MySQL 数据库

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司。MySQL 的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。由于 MySQL 的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

与其他的大型数据库例如 Oracle、DB2、SQL Server 等相比，MySQL 自有它的不足之处，但是这丝毫也没有减少它受欢迎的程度。对于一般的个人使用者和中小型企业来说，MySQL 提供的功能已经绰绰有余，而且由于 MySQL 是开放源码软件，因此可以大大降低总体拥有成本[22]。

其他语言和技术的应用还包括 JavaScript、Ajax、Flex 以及 XML、CSS 等，就不在这里一一介绍了。

第三节 系统业务功能结构

了解了报表子系统的系统架构以及开发环境，再来就需要明确报表子系统的业务流程和功能结构。本文第二章已经从客户需求的角度对系统的功能进行了论述，下面将从开发者的角度重新审视报表子系统的业务需求和功能结构。

报表子系统的所有业务需求，都是围绕“报表”这一主题进行的，所以系统的核心业务是针对报表的需求多样性和功能扩展性进行的规划和完善。在报

表“从无到有”的整个过程中，可以纵向的将报表的结构分解为“报表数据源”、“报表数据整理”和“报表数据展示”。下面就从这三个方面进行阐述。

3.3.1 报表数据源

报表子系统中的数据源模块是较早确定要独立的部分。作为报表的核心，数据永远是重中之重。将之与数据整理模块和数据渲染模块分割，是有利于理清报表子系统的功能结构和调用流程的。至于数据源自身的种类划分，则是要依据数据源应用的场景和真实数据来源的模式。

报表子系统中需要非常固定的用于获取时间信息的数据源，其中涉及到年份、季度、月份、日期甚至是自然周和月内旬等不尽相同的范围和粒度划分。而这些用函数来提供固定的结构和数据是最为快捷方便的。这就是所谓“函数数据源”。

业务报表中涉及的业务数据当然是从业务子系统数据库中检索而来的，而这个检索逻辑通常是通过 SQL 语句来实现的。用于承载这种 SQL 查询语句的数据源就是所谓“SQL 数据源”。当然还有的业务子系统对于数据库的检索具有更高级别的限制，同时业务子系统向外提供数据的获取接口。用于和这样的接口对接以获取数据的数据源就是所谓“接口数据源”。

由于报表核心通过独立文件存储，所以就有相同或近似逻辑的数据源因为归属于不同报表而反复出现。基于这些数据源的逻辑相同或近似，就可以将之抽象成为一个独立于报表存在的“基础数据源”。各个报表应用的数据源则可以直接“遗传”自基础数据源，就是所谓“遗传数据源”。这样就可以有效地将业务逻辑尽可能的进行统一管理。

由于报表逻辑的日益复杂，单一核心的模式，已经很难完成高复杂度的需求。即便能够完成，单一核心内部的逻辑复杂程度也可想而知，对于后期的维护来说，同样很困难。所以报表子系统的核心将被分解为多个次级核心，通过各自相对简单的数据源获取部分数据，然后再根据逻辑组装合并，这也就是所谓“集合数据源”。

报表的数据源，顾名思义就是报表数据的来源。可以说对于一张报表，数据源是基础是最开始的阶段。可是对于报表子系统来说，数据源部分中的实际数据并不在报表子系统中进行维护。这也就造成报表子系统对于实际数据的可控

性由于子系统之间的分割而降低。

而由此所造成的报表数据的不稳定是要最先被考虑和克服的。所以，在数据源层面对数据获取功能的分类是很必要的。

更贴近业务子系统部分的数据源需要考虑的更多在于业务子系统提供数据的方式和限制。如果业务子系统开放数据库，那么报表数据源可以将用于查询数据的制式 SQL 语句分解到报表子系统相关数据结构中，经过进一步的数据整理和筛选，完成提供数据的功能，这也就是比较灵活的“SQL 数据源”。如果业务子系统不开放数据库，那么该业务子系统就需要为报表子系统提供配套接口用来获取固定结构的数据，再进一步进行数据整理和筛选，完成提供数据的功能，这也就是对业务子系统比较依赖的“接口数据源”。

更贴近报表子系统部分的数据源也存在获取原始数据的部分。报表子系统自身会提供相关的函数用来获取报表有关的固定数据。数据源可以从这些函数获取固定数据，然后进一步进行数据整理和筛选，完成提供数据的功能，这也就是和业务子系统完全无关的“函数数据源”。

上面所说的三种数据源是用于获取原始数据的“基础数据源”。而一般情况原始数据源的数据被直接应用于报表的情况并不多见，通常原始数据还需要经过“项的削减”和“逻辑的拼接”，才能适应报表基础的展示范围和业务逻辑。

为了保证基础数据源的通用性，基础数据源中会存在冗余的数据项，其获取的数据往往相较于特定报表要求的为多。为了在基础数据源的基础上实现“项的削减”，完善数据源之间的内部关系，实现报表对于基础数据源的约束，设计一种逻辑上贴近报表，形式上遗传自基础数据源的“遗传数据源”。这种数据源的基础范围遗传自基础数据源，作用到报表上的时候会依据报表的数据要求对数据项范围进行削减。这种数据源在基础数据源和报表数据要求之间起到了承接的作用，是一种“中间数据源”。

为了保证基础数据源的简单性，基础数据源中是不会存在过于复杂的数据逻辑，其获取的数据往往相较于特定报表要求的简单。为了在基础数据源的基础上实现“逻辑的拼接”，完善数据源之间的逻辑关系，打破报表对于基础数据源的完全函数依赖，设计一种合并于多个数据源的“集合数据源”。这种数据源的主要作用是依据报表逻辑将各个基础数据源进行合并，最终的结果数据源能够合适的应用于报表。

综上，数据源部分总共分为了两类：“基础数据源”和“中间数据源”。共

计五种：“SQL 数据源”、“接口数据源”、“函数数据源”、“遗传数据源”和“集合数据源”，其中“SQL 数据源”、“接口数据源”和“函数数据源”属于“基础数据源”，“遗传数据源”和“合并数据源”属于“中间数据源”。

3.3.2 报表数据渲染

报表的数据展示，子系统中主要体现在表格展示中。可以说对于一张报表，数据展示是最终的阶段，也是最贴近客户应用的部分，也就是被客户要求最多的部分。

基础的表格结构，主要分为“行”、“列”和“格”三个部分，报表子系统也不例外。在这个部分，开发者更关注数据的展示样式而非数据本身。样式部分主要分为表格样式、数据样式和业务样式，这些样式的设置是肯定要开放给客户自主进行的。

表格样式主要包括表格的列宽、行高和背景色等。数据样式主要包括数据文本的字体、字号和前景色等。业务样式主要包括日期的格式、数字的精度、金额的千分位和货币符号前缀等。上述这些样式是建立在报表页面表格控件的渲染上的。

由于业务数据的需求，表格行的种类也需要按逻辑区分，主要是合计行以及各种逻辑小计行。这也就意味着逻辑行和数据行需要完整显示才能在整体上体现各行数据之间的逻辑关系和数据完整。所以报表的页面表格控件采取无分页的整体显示。

由于“逻辑行”的出现，原先简单的实时查询条件交互和数据排序都变得复杂起来。

区别于纯粹的表格数据展示，为了使业务子系统中的报表和相应子系统之间的联系更紧密，同时使存在业务关联或逻辑关联的报表便于相互检验，报表子系统的数据展示部分还规划了“单元格跳转”功能。该功能将在报表子系统中的报表单元格中建立链接，目标可以锁定在业务子系统的数据页面或报表页面。这样就可以在报表和业务子系统之间构建快捷并且系统的关联。

综上，数据展示部分主要解决样式问题，其中包括“表格样式”、“数据样式”和“业务样式”，而且为数据整理实时交互提供干预平台，为良好的用户体验提供基础。

3.3.3 报表数据整理

上面说到了报表数据的来源和最终展示形式，虽然这两部分也会涉及一些业务子系统的需求，但是更多的业务需求和报表特定需求还是需要特定的部分加以整理，而这个部分就是所谓的“报表数据整理”。

这部分主要的功能，是报表数据结构整理的过程，也是报表逻辑数据生成的过程，这两项就是报表业务需求的主要体现。具体的整理项目又可以细化成为若干原子的整理过程，这些原子过程将在下面进行具体的规划。

报表子系统中的数据整理模块独立存在是数据源模块和数据渲染模块确定独立的结果。而从整体上看，数据整理模块的独立存在也是合乎系统设计的原则和规范的。

报表数据整理的结果是要用于数据渲染模块的，对于表格报表来说，渲染模块所需的数据结构无疑是二维表格式的。报表整理的数据来源是数据源提供的，而数据源的基础数据又是来源于关系型数据库，因此来源的数据结构也是二维表格式的。综上，报表子系统的数据整理部分是一套对于二维表格数据的逻辑整理。

二维表格中的两重维度分别是行和列，所有数据所属的单元格，都会存在行和列的相关属性。可以说数据整理是在数据源数据的基础上依据各自的业务逻辑和显示逻辑进行的格式化整理。

数据整理的方式也是分为两种，分别是行数据整理和列数据整理。

3.3.3.1 逻辑数据计算过程

此种计算的目标是通过相应逻辑运算完善当前行数据，主要分为公式逻辑计算、函数逻辑计算和接口逻辑计算。其中公式逻辑计算实现常规的四则运算；函数逻辑计算实现常规的非四则运算；接口逻辑计算实现业务子系统中特定的规则逻辑运算。

3.3.3.2 逻辑数据汇总过程

此种汇总的目标是通过相应逻辑运算完善逻辑行数据，汇总逻辑主要分为“合计”、“条数累计”以及“平均”等，根据业务子系统的需求，后续可以引入“期望”、“方差”甚至“加权”等统计学概念。

3.3.3.3 逻辑数据补全过程

此种逻辑的目标是通过既定逻辑完善空数据，主要包括基础数据源提供的“NULL 数据”和逻辑行中的“提示数据”。数据的补全不仅为了报表最终的数据展示，同时也可以避免由于空数据而引起的数据逻辑计算可能出现的错误。

3.3.3.4 逻辑数据筛选过程

此种逻辑的目标是通过报表实时交互的数据约束对基础数据和逻辑整理数据进行筛选，当然主要针对逻辑整理数据，因为基础数据源是存在基础的数据筛选功能的。要注意的是数据判定时的组合逻辑和筛选时的行间逻辑。另外为了简化报表数据量，还会删除一部分在业务上无意义的数据。

3.3.3.5 数据结构整理过程

此种调整的目标是将基础数据源获得的数据调整成为符合报表展示要求的结构。尤其是存在逻辑数据行的时候，逻辑数据行和基础数据行之间以及逻辑数据行之间的关系描述的特定数据结构。

3.3.3.6 数据展示整理过程

此种整理的目标是在数据结构整理完毕之后，根据一些显示要求对数据进一步的调整，主要包括对于数据的逻辑排序、对重复数据的省略显示等。

综上，报表数据的所有逻辑计算和整理过程都大略的介绍完了。

第四节　系统主体业务流程

报表子系统的主体业务流程的特点是“动态生成、独立存储、广泛应用”。

所谓“动态生成”主要是指模板报表的核心算法文件和客户报表的应用参数文件是有后台程序动态生成的。PHP 语言对于文件系统相关的功能简便强大，而且属于脚本语言的 PHP 文件是无需编译可直接运行的，这些使动态生成文件的设想更容易实现，而且简单易用。

所谓“独立存储”是指模板报表的核心算法文件独立存储；客户报表的应用参数文件独立存储；报表数据整理流程文件独立存储。独立存储的文件体积更小，更便于系统加载和人工维护。虽然文件的数量会有所增加，但是合理的目

录结构能够很好的解决这一点。

所谓“广泛应用”主要是指模板报表的核心算法和报表数据整理的流程需要被广泛应用。这对于业务子系统中各个报表的业务保持逻辑一致性是有积极意义的，而且尽可能的代码重用也提高了代码的可维护性。

报表子系统的主体业务流程如图 3.2 所示，可以看出，报表数据的获取主要源于核心算法和整理流程，但是更重要的是自定义参数，他会参与到报表数据获取的每一个步骤中，对任何一个细节加以控制，以保证数据的正确性和完整性，也是报表个体区别于其他报表个体的最主要依据。

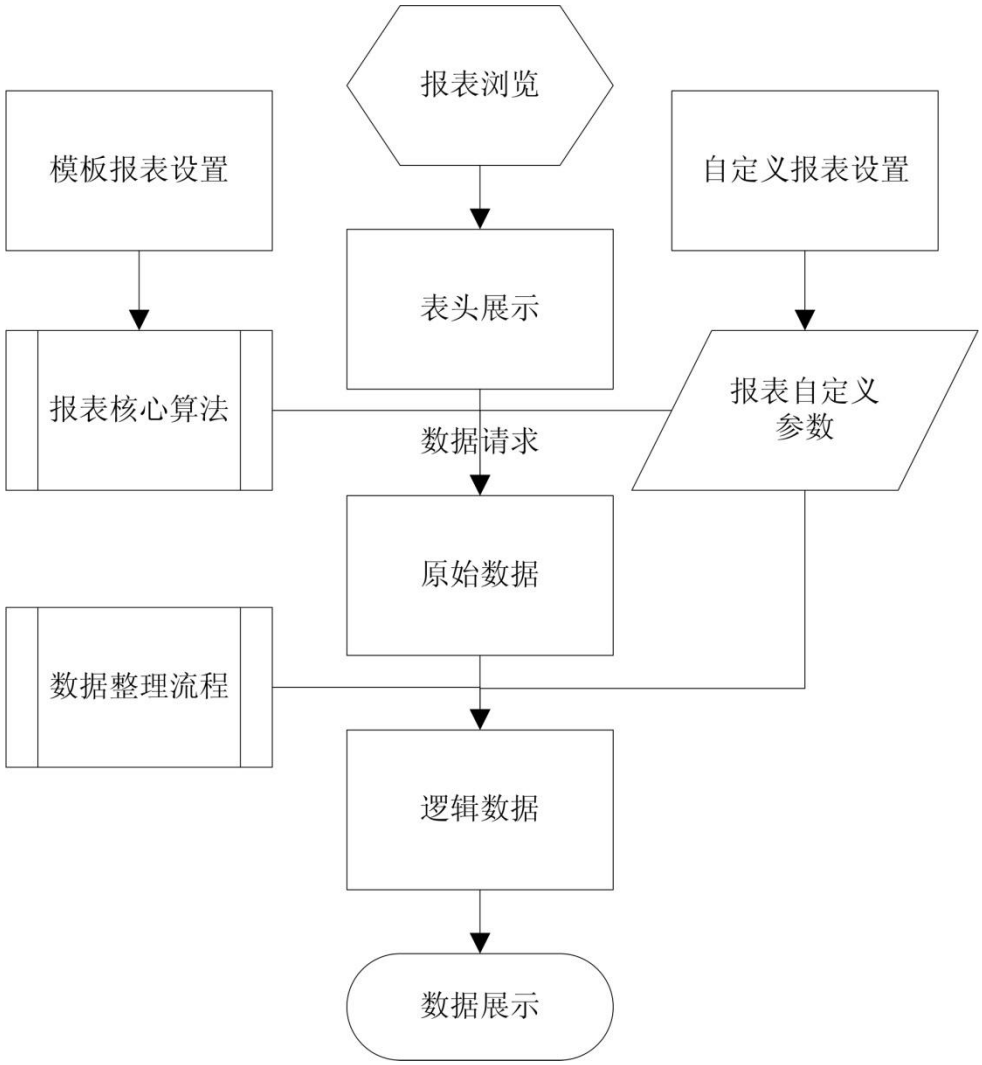

图 3.2　报表子系统主体流程图

第四章　系统数据库设计

信息系统数据库的设计向来是系统开发中的重中之重，因为数据库的结构在一定程度上会影响系统逻辑和算法的设计，而且作为系统数据的承载者也需要一个规范合理并且便于应用的结构。

第一节　数据库整体设计

从报表子系统的功能和服务来看，子系统需要为多个业务子系统，大量的客户提供报表服务。在这中间，会存在客户对报表的定制需求，但提供更多的是固定制式的报表。为了保证固定制式报表的独立完整性，就需要将这些固定制式的报表与客户实际应用的报表完全分开。

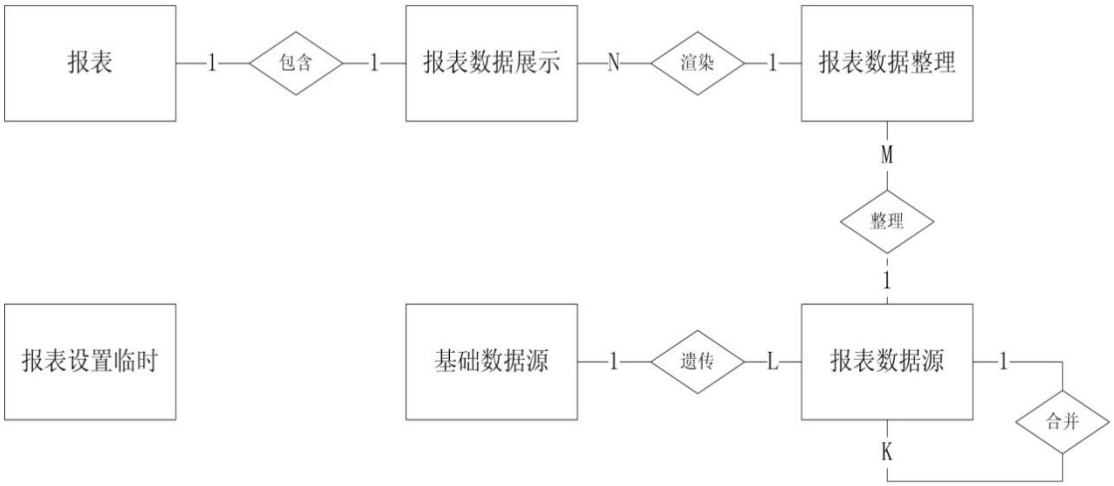

图 4.1　报表子系统数据库结构概况图

报表子系统的设计是，将数据库分为“标准报表模板库”和“业务报表库”，其中的结构大致相同，这样就可以对报表子系统提供的标准报表进行管理。当有新的客户需要报表子系统提供服务时，也可以从“标准报表模板库”中拷贝一份相应的报表到“业务报表库”中。

具体到一个数据库中，从报表子系统的整体来看，数据库表大致分为五部

分，分别是“报表基础信息数据表类”、“报表数据源信息数据表类”、“报表数据整理信息表类”、“报表数据展示信息数据表类”和“报表设置信息数据表类”。其间的大致关系如图 4.1 所示。

其中的设置部分由于功能的特点，在报表上线之前可以直接应用其他部分的数据库结构，上后的维护过程也只需要一个复制信息的临时表即可，所以报表设置部分的数据库结构相对来说是最简单的。

下面将就每一个部分进行详细的说明。报表基础信息数据表

第二节　报表基础信息数据表

报表的基础信息其实很有限，报表子系统提供报表的分类，数据库表结构如表 4.1 所示。报表类别是一个相对独立的字典，里面只涉及了报表类别自身的信息和控制标识。

表 4.1　报表类别表 - tbl_reportclass

名称	类型	解释	备注
strNo	VARCHAR(10)	类别编号	
strName	VARCHAR (20)	类别名称	
isLeafClass	TINYINT(1) UNSIGNED	叶子类别标识	
strRelation	VARCHAR (255)	类别关系序列	

报表基础信息表，是系统对一张报表存在与否的唯一判断依据，数据库表结构如表 4.2 所示。其中涉及报表的类别，报表子系统的规则是“一张报表只能属于一个报表类别”，即报表和报表类别的关系是 1:N 的。

表 4.2　报表基础表 - tbl_reporttable

名称	类型	解释	备注
strNo	VARCHAR(10)	报表编号	
strName	VARCHAR (20)	报表名称	
strClassNo	VARCHAR (10)	归属类别编号	
eStatus	ENUM(‘config’, ’apply’, ’upgrade’, ’discard’)	状态	config: 设置; apply: 使用; upgrade: 升级; discard: 废弃;
strVersion	VARCHAR(20)	版本	

另外对于报表的版本控制是很重要的，因为标准报表模板的变更不会同步变更业务报表库中的标准报表，这也就造成“业务报表库”中的标准报表与“标准报表模板库”中的有可能存在不同，而这种功能的差异就需要通过版本进行标记，对于版本的维护有单独的数据库表进行维护，数据库表结构如表 4.3 所示。

表 4.3 报表版本表 - tbl_reportversion

名称	类型	解释	备注
strTableNo	VARCHAR(10)	报表编号	
strVersion	VARCHAR(20)	版本	I 段: 系统; II 段: 基础业务; III 段: 自定义业务;
strAuthor	VARCHAR(10)	责任人	
strDescription	TEXT	功能变更说明	

报表基础信息的最后一部分是查询条件部分，由于报表的查询条件前台展示应用了遗留系统中的查询控件，所以数据库表的结构和内容就需要项已有空间的结构和内容有所倾斜。数据库表结构如表 4.4 所示。

表 4.4 报表查询条件表 - tbl_reportquery

名称	类型	解释	备注
strNo	VARCHAR(10)	查询条件编号	
strName	VARCHAR (20)	查询条件名称	
strTableNo	VARCHAR(20)	归属报表编号	
eType	ENUM(‘text’, ’number’, ’date’, ’enum’)	查询类型	text: 文本; number: 数字; date: 日期; enum: 枚举;
isDefault	TINYINT(1) UNSIGNED	默认查询标识	
strDefaultValue	VARCHAR(100)	默认查询值	
isUsedCheck	TINYINT(1) UNSIGNED	需检验标识	
strCheckValue	VARCHAR(100)	检验约束值	

第三节 报表数据源信息数据表

在图 4.1 中数据源被分为了“基础数据源”和“报表数据源”两部分，是考

虑到数据源单独存在和数据源归属于报表的两种情况，但这两种情况并不妨碍数据源部分的数据库结构的通用性。

数据源是报表子系统中举足轻重的一部分，几乎所有的原始数据都来自于此，所以对于这部分数据库结构的设计更需要注意。

报表子系统的数据源表记录了数据源整体性的信息，数据库表结构如表 4.5 所示。主要用 eUsedType 字段区分数据源类型，为其他数据源部分的数据库表的应用进行约束和规范。

表 4.5 报表数据源表 - tbl_reportsource

名称	类型	解释	备注
strNo	VARCHAR(10)	数据源编号	
strName	VARCHAR (20)	数据源名称	
strTableNo	VARCHAR(10)	归属报表编号	可以不归属报表
eUsedType	ENUM(‘sql’, ’if’, ’func’, ’inherit’, ’set’)	应用类型	sql: SQL; if: 接口; func: 函数; inherit: 遗传; set: 集合;
strSubSource	VARCHAR(255)	子数据源序列	eUsedType 为 set 时有效
eMergeType	ENUM(‘continue’, ‘half’, ‘match’, ‘full’, ‘hit’)	合并类型	continue: 接续; half: 偏集; match: 交集; full: 并集; hit: 命中;
intLimitCount	TINYINT(4) UNSIGNED	数据裁剪量	0 为不裁剪

报表子系统数据源应用数据库表的情况在这里记录，数据库表结构如表 4.6 所示。这是一张应用 SQL 类型数据源的专用表，其中记录了数据源应用的每个数据库表，以及应用的方式和顺序。这张表可以说为数据源的数据范围作了一个大范围的限定。

表 4.6 报表数据源数据库表应用表 - tbl_reportsource_table

名称	类型	解释	备注
strNo	VARCHAR(10)	数据库表应用编号	
strName	VARCHAR (20)	数据库表应用名称	
strSourceNo	VARCHAR(10)	归属数据源编号	

续表 4.6 报表数据源数据库表应用表 - tbl_reportsource_table

名称	类型	解释	备注
strTableName	VARCHAR(10)	数据库表名称	
eUsedType	ENUM(‘index’, ’left’, ’inner’)	应用类型	index: 主表; left: 左关联表; inner: 内关联表;
intUsedOrder	TINYINT(2) UNSIGNED	加载顺序	

应用其他类型数据源的数据范围是其本身就已经加以限定的。在大范围限定之后，进一步精确字段信息的数据由数据源项表存储，数据库表结构如表 4.7 所示。这里将就每一个数据源项进行记录和设置，会标记该项在数据源整体中是否参与了数据排序和分组以及应用聚合函数。

表 4.7 报表数据源项表 - tbl_reportsource_field

名称	类型	解释	备注
strNo	VARCHAR(10)	数据源项编号	
strName	VARCHAR (20)	数据源项名称	
strSourceNo	VARCHAR(10)	归属数据源编号	
strField	VARCHAR(100)	项值	
eUsedType	ENUM(‘default’, ’table’, ’formula’, ’const’)	应用类型	default: 默认; table: 表; formula: 公式; const: 常量;
strTableNo	VARCHAR(10)	归属数据库表应用编号	eUsedType 为 table 时有效
intGroupOrder	TINYINT(4) UNSIGNED	参与分组顺序	0 为不参与分组
eAggregateType	ENUM(‘none’, ’sum’, ’avg’, ’count’, ’concat’)	应用聚合函数	none: 无; sum: 求和; avg: 求平均; count: 求总数; concat: 连接;
eSortType	ENUM(‘none’, ‘asc’, ’desc’)	参与排序类型	none: 不参与; asc: 升序; desc: 降序;
intSortOrder	TINYINT(4) UNSIGNED	参与排序顺序	

其中对于 strField 字段值的认定需要借助 eUsedType 字段进行：eUsedType

为 table 时，strField 中的字符串当解析为数据库字段；eUsedType 为 formula 时，strField 中的字符串当解析为公式，其形如：“{#=strFieldNo#}+{#=strFieldNo#}”；eUsedType 为 const 时，strField 中的内容当直接使用。

和数据源数据库表应用表配套存在的是关联条件表，这也是一张应用 SQL 类型数据源的专用表。他记录了关联表之间的关联条件，数据库表结构如表 4.8 所示。报表子系统中要求关联表之间的关联关系需要是等于。其中 strLogicArea 存储的逻辑区域序列是用于在第六章第三节中说明的逻辑长句的拼接。

表 4.8 报表数据源表关联条件表 - tbl_reportsource_join

名称	类型	解释	备注
strNo	VARCHAR(10)	关联条件编号	
strName	VARCHAR (20)	关联条件名称	
strSourceNo	VARCHAR(10)	归属数据源编号	
strTableNo	VARCHAR(10)	作用关联表应用编号	
strAnchorFieldNo	VARCHAR(10)	锚项编号	
strMarkFieldNo	VARCHAR(10)	比较项编号	
strLogicArea	VARCHAR(100)	逻辑区域序列	

数据源数据约束表，记录了数据源中的数据约束信息，数据库表结构如表 4.9 所示。

表 4.9 报表数据源数据约束表 - tbl_reportsource_scope

名称	类型	解释	备注
strNo	VARCHAR(10)	数据约束编号	
strName	VARCHAR (20)	数据约束名称	
strSourceNo	VARCHAR(10)	归属数据源编号	
strFieldNo	VARCHAR(10)	应用数据源项编号	
strTableNo	VARCHAR(10)	作用表应用编号	_HAVING_ 为作用于 having 子句
eMatchType	ENUM(‘=’, ’<>’, ’<’, ’>’, ’like’, ’in’)	匹配方式	
eValueType	ENUM(‘const’, ’func’, ’source’)	约束值类型	const: 常量; func: 函数; source: 数据源;
strValue	VARCHAR(100)	简单约束值	
strValueSourceNo	VARCHAR(10)	值约束数据源编号	
strValueFieldNo	VARCHAR(10)	值约束数据源项编号	

续表 4.9 报表数据源数据约束表 - tbl_reportsource_scope

名称	类型	解释	备注
strValueFuncFile	VARCHAR(100)	值约束函数文件	
strValueFuncName	VARCHAR(100)	值约束函数名称	
strValueFuncParam	VARCHAR(100)	值约束函数参数	JSON 存储
strLogicArea	VARCHAR(100)	逻辑区域序列	

其中数据约束的值是存在多种来源的。常量约束时最常被用到的。还可以通过报表子系统提供的函数获得数据，函数中是可以想业务子系统接口请求数据的。最后还可以通过一个数据源获得约束用的数据集合，这也解决了 SQL 类型数据源不支持子查询的问题。

另外 strTableNo 字段在数据源应用 SQL 类型时有效，其记录数据约束的范围，备选项是数据源应用表中记录的表编号和特殊的应用于 having 子句的“_HAVING_”标识。

和数据源数据约束表功能相似的是数据源查询条件表，数据库表结构如表 4.10 所示。都是起到了数据筛选的作用，只是查询条件的筛选更注重的是客户交互。

表 4.10 报表数据源查询条件表 - tbl_reportsource_query

名称	类型	解释	备注
strNo	VARCHAR(10)	查询条件应用编号	
strName	VARCHAR (20)	查询条件应用名称	
strSourceNo	VARCHAR(10)	归属数据源编号	
strQueryNo	VARCHAR(10)	查询条件编号	
strTableNo	VARCHAR(10)	作用表应用编号	_HAVING_为作用于 having 子句
strIndexFieldNo	VARCHAR(10)	键匹配数据源项编号	
strValueFieldNo	VARCHAR(10)	值匹配数据源项编号	
strLogicArea	VARCHAR(100)	逻辑区域序列	

数据源查询条件表的设计出了必要的外键之外，将作用查询条件的数据源项拆分成为“strIndexFieldNo”和“strValueFieldNo”两项进行记录。是因为数据源无发预知前台查询条件的类型：查询条件中的文本类型、日期类型需要作用到“strValueFieldNo”上，而枚举类型需要作用到“strIndexFieldNo”上。这样的设计，可以使数据源的查询条件设置完全不必考虑前台查询条件的交互类型，是一种有利于维护的设置方法。

和关联条件表一样，数据源数据约束表和查询条件表存在 strLogicArea 字段，用以标记条件子句的逻辑区域。以 SQL 类型数据源为例，在一句 SQL 中 on 子句需要关联条件，关联表数据约束，关联表查询条件三者共同应用 strLogicArea，来生成最终的逻辑长句子句，where 子句和 having 子句也是同样的道理。所以上面所说的三张表是不能割裂来看的，这三张表一起构建起了 SQL 语句中的语句主干。

数据源关系表记录的是两个数据源之间数据源项的对应关系，数据库表结构如表 4.11 所示。

表 4.11 报表数据源遗传合并关系表 - tbl_reportsource_relation

名称	类型	解释	备注
strNo	VARCHAR(10)	数据源项关系编号	
strName	VARCHAR (20)	数据源项关系名称	
strSourceNo	VARCHAR(10)	数据源编号	
strFieldNo	VARCHAR(10)	数据源项编号	
strRootSourceNo	VARCHAR(10)	来源数据源编号	
strRootFieldNo	VARCHAR(10)	来源数据源项编号	

数据源关系表中记录的是两个数据源的数据源项之间的对应关系，这个关系只是一个单纯的映射，遗传数据源可以应用这个关系实现自身对于数据项的筛选，及和数据源可以根据这个关系确定数据源合并过程中的关联字段。所以这个看似简单的映射关系对于整个数据源体系的影响是深远而巨大的。

第四节 报表数据整理信息数据表

报表数据的整理，在这里就不再赘述了。不过要说明的是从整个报表数据获取整理显示的流程来看，从数据整理开始，报表数据才有了明显的表格结构的痕迹。数据整理部分的数据库表就是根据“表”、“行”、“列”和“格”进行设置的。

数据整理的主体信息，数据库表结构如表 4.12 所示。其中要说明的是 strIndexField 字段记录的值只有在横向化展开列是会被应用到，作为展开过程中的锚字段应用。eZeroType 是记录零记录过滤类型的标记，所谓零记录过滤就是将整行没有意义的数据丢弃的过程。intLimitCount 字段的剪裁标记不同于数据源

阶段的剪裁，此处的剪裁将体现在最终的数据显示量上，例如一系列的“TOP - 10”报表。

表 4.12 报表整理基础表 - tbl_reportcommon

名称	类型	解释	备注
strNo	VARCHAR(10)	数据整理编号	
strName	VARCHAR (20)	数据整理名称	
strTableNo	VARCHAR(10)	报表编号	
strIndexField	VARCHAR(255)	关键字段序列	
eZeroType	ENUM(‘none’, ‘basic’, ‘all’)	过滤零标识	none: 不过滤; basic: 基础行过滤; all: 全过滤;
intLimitCount	TINYINT(4) UNSIGNED	数据裁剪量	0 为不裁剪

数据整理行信息表记录了表格结构中行的种类和数量，数据库表结构如表 4.13 所示。这里的行记录是侧重于数据行的，类似标题行的信息在这里是不予记录的。逻辑行的逻辑约束列是行本身逻辑执行的信息主体。

表 4.13 报表整理行信息表 - tbl_reportcommon_row

名称	类型	解释	备注
strNo	VARCHAR(10)	报表行编号	
strName	VARCHAR (20)	报表行名称	
strCommonNo	VARCHAR(10)	归属数据整理编号	
eUsedType	ENUM(‘basic’, ‘logic’)	应用类型	basic: 基础行; logic: 逻辑行;
strAnchorColumn	VARCHAR(255)	逻辑约束列	eUsedType 为 logic 时有效
eLogicType	ENUM(‘sum’, ‘avg’, ‘count’)	逻辑算法类型	sum: 求合计; avg: 求平均; count: 求数量;
strBindColumn	VARCHAR(255)	运算约束列	eUsedType 为 logic 时有效
intComputeOrder	TINYINT(4) UNSIGNED	加载顺序	

数据整理列信息表格记录了每一列的信息，数据库表结构如表 4.14 所示。并记录了每一种列类型所需要的扩展字段，公式、函数、引用以及横向化等各种情况。

前面所说的过滤零记录的机制，并不是刻板的认为只要存在一个不为零（或空）的数量金额信息就是有效数据，因为在销售类报表中，需要关注的只有销

售量，产品的成本价、零售价等一些价格一般是不会为零的，所以，参与零记录过滤的字段就需要在这里通过 isZeroField 进行设置。

在 0 中提到的数据混合整理的设置是通过 intComputeOrder 实现的，和整理行的 intComputeOrder 相似的，这个字段是需要通过严谨的逻辑才能产生的，所以在应用是必须被格的遵守，不同在于，整理行是不允许同时加载的，而整理列是可以的。

表 4.14 报表整理列信息表 - tbl_reportcommon_column

名称	类型	解释	备注
strNo	VARCHAR(10)	报表列编号	
strName	VARCHAR (20)	报表列名称	
strCommonNo	VARCHAR(10)	归属数据整理编号	
strRootFieldNo	VARCHAR(10)	来源项编号	
eUsedType	ENUM(‘basic’, ’formula’, ’func’, ’exp’, ’quote’)	应用类型	basic: 基础; formula: 公式; func: 函数; exp: 横向化扩展; quote: 数据引用;
strFormula	VARCHAR(255)	公式	
strFuncName	VARCHAR(50)	函数名	
strFuncParam	VARCHAR(255)	函数参数	JSON 存储
strExpandSourceNo	VARCHAR(10)	扩展数据源编号	
strExpandIndexFieldNo	VARCHAR(10)	扩展键数据源项编号	
strExpandValueFieldNo	VARCHAR(10)	扩展项数据源项编号	
strQuoteRowNo	VARCHAR(10)	引用行编号	
strQuoteColumnNo	VARCHAR(10)	引用列编号	
strQuoteAnchorColumn	VARCHAR(10)	应用逻辑约束列	
intQuoteDepth	TINYINT(4)	引用深度	
isZeroField	TINYINT(1) UNSIGNED	参与过滤零记录标识	
intComputeOrder	TINYINT(4) UNSIGNED	加载顺序	

表格的原子对象就是单元格，同时报表子系统对于设置的记录页是以单元格为单位的,所以对于单元格的描述也最是复杂,数据库表结构如表 4.15 所示。

对于一个单元格来讲，他同时属于一个行和一个列，所以对于它本身的算法就需要在他所归属的行或列中进行选择，也就有了 eUsedType 中“row”和“column”的选项，当单元格不继承行或列的算法时，应设置为“basic”项，

另外，单元格可以应用“const”项，这样可以脱离表格数据范畴，应用该表“strValue”字段进行独立的数据赋值。

“strDefaultNull”字段描述的是数据整理过程中，该单元格的数据出现了“NULL”的情况时的应对数据，这对于参与运算或公式的数据有一个纠错的功能。

数据整理中的数据排序功能对应的设置存储也存在于单元格表，应用“eSortType”和“intSortOrder”两个字段描述所有参与排序的单元格，再通过各个单元格所归属的行和列进行第六章第七节中所描述的组合排序整理过程。

表 4.15 报表整理格信息表 - tbl_reportcommon_cell

名称	类型	解释	备注
strNo	VARCHAR(10)	报表格编号	
strName	VARCHAR (20)	报表格名称	
strCommonNo	VARCHAR(10)	归属数据整理编号	
strRowNo	VARCHAR(10)	行编号	
strColumnNo	VARCHAR(10)	列编号	
eUsedType	ENUM(‘basic’, ’row’, ’column’, ’const’)	应用类型	basic: 基础格; row: 继承行; column: 继承列; const: 常量信息;
isUsedBind	TINYINT(1) UNSIGNED	启用运算约束	
strValue	VARCHAR(100)	常量信息值	eUsedType 为 const 时有效
strDefaultNull	VARCHAR(100)	空值默认值	
eSortType	ENUM(‘none’, ’asc’, ’desc’)	排序方式	none: 不排序; asc: 升序; desc: 降序;
intSortOrder	TINYINT(4) UNSIGNED	排序顺序	

第五节 报表表格展示信息数据表

报表的表格显示与报表整理的结构很相似，都是基于二维表格的基本格式，只是在显示部分更侧重于页面样式的渲染设置。

报表整体显示，数据库表结构如表 4.16 所示。在这里只需要确定数据整理的来源，就可以将整张报表的数据全部取出，根据行列格填充到整张表格中。

表 4.16 报表表格基础信息表 - tbl_reportgrid

名称	类型	解释	备注
strNo	VARCHAR(10)	报表编号	
strName	VARCHAR (20)	报表名称	
strCommonNo	VARCHAR(10)	应用数据调整编号	
intBlockCount	TINYINT(4) UNSIGNED	左固定列数	

报表展示用的行沿用了数据整理的行编号，并根据不同的行设置不同的行显示样式，数据库表结构如表 4.17 所示。主要涉及了对齐方式和配色方案。

表 4.17 报表表格行信息表 - tbl_reportgrid_row

名称	类型	解释	备注
strNo	VARCHAR(10)	报表行编号	
strName	VARCHAR (20)	报表行名称	
strTableNo	VARCHAR(10)	归属报表编号	
eAlign	ENUM(‘left’, ’center’, ’right’)	水平对齐方式	left: 左对齐; center: 居中; right: 右对齐;
eValign	ENUM(‘top’, ’middle’, ’bottom’)	垂直对齐方式	top: 上对齐; middle: 居中; bottom: 下对齐;
strForegroundColor	VARCHAR(6)	前景色	6 位 RGB 码
strBackgroundColor	VARCHAR(6)	背景色	6 位 RGB 码
strBorderColor	VARCHAR(6)	边框色	6 位 RGB 码

报表展示用的列同样沿用了数据整理的列编号，并根据不同的列设置不同的数据样式和显示样式，数据库表结构如表 4.18 所示。

表 4.18 报表表格列信息表 - tbl_reportgrid_column

名称	类型	解释	备注
strNo	VARCHAR(10)	报表列编号	
strName	VARCHAR (20)	报表列名称	
strTableNo	VARCHAR(10)	归属报表编号	
eDisplayType	ENUM(‘text’, ’money’, ’number’, ’date’)	显示类型	text: 文本; money: 金额; number: 数量; date: 日期;
intOrder	TINYINT(4) UNSIGNED	显示顺序	

续表 4.18 报表表格列信息表 - tbl_reportgrid_column

名称	类型	解释	备注
intWidth	SMALLINT(6)	列宽度	
strPrefix	VARCHAR(10)	数据前缀	
strSuffix	VARCHAR(10)	数据后缀	
intDecimalCount	TINYINT(4) UNSIGNED	小数位数	
isShowThousandth	TINYINT(1) UNSIGNED	启用千分位标识	
strMoneyOperator	VARCHAR(10)	货币符号	
eAlign	ENUM('left', 'center', 'right')	水平对齐方式	left: 左对齐; center: 居中; right: 右对齐;
eValign	ENUM('top', 'middle', 'bottom')	垂直对齐方式	top: 上对齐; middle: 居中; bottom: 下对齐;
strForegroundColor	VARCHAR(6)	前景色	6 位 RGB 码
strBackgroundColor	VARCHAR(6)	背景色	6 位 RGB 码
strBorderColor	VARCHAR(6)	边框色	6 位 RGB 码

相较于数据整理中的单元格，数据展示中的单元格更加简单，数据库表结构如表 4.19 所示。只需要确定单元格继承行或者列即可，不存在第三种可能，这里所说的继承，也仅限于对齐方式和配色方案，类似数据类型，小数位数等列特有的属性不在此列。

表 4.19 报表表格格信息表 - tbl_reportgrid_cell

名称	类型	解释	备注
strNo	VARCHAR(10)	报表格编号	
strName	VARCHAR (20)	报表格名称	
strTableNo	VARCHAR(10)	归属报表编号	
strRowNo	VARCHAR(10)	行编号	
strColumnNo	VARCHAR(10)	列编号	
eUsedType	ENUM('row', 'column')	应用类型	row: 继承行; column: 继承列;

此处的报表数据展示表格相关的数据结构，应用简单，而且便于设置，同时基本涵盖了表格显示的样式需求。

第六节　报表设置信息数据表

报表的设置部分由于不允许“暂存”状态的存在，数据库结构属于一张临时表，数据库表结构如表 4.20 所示。表中存在汇总的各个数据库表的联合建，用以标识记录，同时将设置信息散列在表结构中，这样既节省数据列，同时也可以提出绝大部分的设置冗余存储。

这样的数据库表结构，可以有效地将报表业务部分和设置部分分开，而且在临时数据持有量方面也适中，经过评估是一种不错的报表设置临时数据表结构。

表 4.20　报表子系统设置临时表 - tbl_reportconfig

名称	类型	解释	备注
strAuthor	VARCHAR (20)	操作者	
strTableName	VARCHAR(100)	数据库表名称	
strFieldName	VARCHAR(100)	数据库表字段名称	
strValue	TEXT	原始值	
strModifyValue	TEXT	修正值	
strTableNo	VARCHAR(10)	报表编号	
strSourceNo	VARCHAR(10)	数据源编号	
strFieldNo	VARCHAR(10)	数据源项编号	
strJoinNo	VARCHAR(10)	关联表编号	
strScopeNo	VARCHAR(10)	数据范围编号	
strQueryNo	VARCHAR(10)	数据查询编号	
strCommonNo	VARCHAR(10)	数据整理编号	
strRowNo	VARCHAR(10)	行编号	
strColumnNo	VARCHAR(10)	列编号	

第五章　系统详细设计与实现

报表子系统的详细设计依然要从报表的数据源、数据整理和数据展示三个部分开始说起。根据第三章中总体的设计，进行各个部分的详细设计，相应的数据库设计已经在第四章进行统一说明。

第一节　报表数据源部分

经过 3.3.1 中阐述已经可以明确，报表的数据源可以细分为“SQL 数据源”、“接口数据源”、“函数数据源”、“遗传数据源”和“集合数据源”五种。由于需要向数据整理部分功能提供统一数据结构的数据，故此无论数据源内部的处理机制如何，都要保证对外输出的数据结构一致。

5.1.1　报表数据源的输出结构

作为表格数据应用最为方便和基础的结构是二维的结构，也就是关系型数据库查询输出地结构。故此“SQL 数据源”的数据是可以直接输出的。同为“基础数据源”的“接口数据源”和“函数数据源”就需要在既得数据的基础上进一步的调整数据结构。

其中的“接口数据源”依赖于业务子系统的数据提供接口，而要保证数据结构的统一，在报表子系统中就需要用一个数据结构整理的函数进行处理，于是“接口数据源”就在形式上趋同于“函数数据源”了。

“遗传数据源”就是遗传自父数据源的子数据源，所以在数据结构上是同富数据源严格保持一致的。“集合数据源”是将若干数据源进行逻辑合并而得到的数据源，它的数据结构是可以通过逻辑控制而达到设计要求的。

5.1.2　SQL 数据源的内部机制函数

报表子系统对于开放数据库的业务子系统来说，应用更多更方便的是 SQL 数据源。SQL 数据源支持常规的单句查询 SQL 语法：

一、支持各种字段查询

应用 SQL 进行字段查询的方式有很多种，常量字符串查询、简单的单字段查询、字段联合的公式查询、以 date_format 和 concat 为代表的应用整理函数的字段查询、配合数据分组的应用聚合函数的字段查询、字段枚举和逻辑枚举等等。为了使得 SQL 查询的结果能够更多的体现业务逻辑，报表子系统的 SQL 数据源规划中，上述的字段查询方式都是需要支持的。

二、支持多表关联

根据数据库设计范式的要求，通常 SQL 查询业务数据的时候都需要进行多表关联，这其中又分为到内关联（inner join）和左关联（left join）。要说明的是，报表子系统的 SQL 数据源是不支持构建右关联（right join）的，因为从形式上，右关联和左关联是很相似的，而且在同一句 SQL 语句中右关联和左关联一般是不会同时出现的。报表子系统的 SQL 数据源规划中，需要支持数据库表之间的关联关系和关联条件。

三、支持逻辑条件

从逻辑上讲，SQL 语句的约束条件可以分为连接于 join-on 之后的关联表条件，连接于 where 之后的主表条件，连接于 having 之后的整理条件。数据获取过程中的业务逻辑约束和用户实时交互筛选大部分是需要作用到 SQL 语句中的，这样不仅可以在基础数据源阶段获取更为精简的数据，也为后续数据整理部分节省了不必要的开销。

四、支持数据分组、排序和裁剪

对于经过既定逻辑约束筛选的数据，进行分组（group by）、排序（order by）和裁剪（limit），在形式上完整了 SQL 语句的功能。

上述四大部分以外，要着重说明的是，目前的 SQL 数据源不支持构建子查询语句，不支持构建应用 union 的语句，不支持将枚举应用于查询项之外的任何位置。SQL 数据源可以实现大部分 SQL 语法，但是为了降低由于 SQL 使用的随意性带来的数据源代码复杂度，还是不得不将 SQL 数据源所支持的 SQL 语法进行进一步的限制和规范。

5.1.3 数据源的内部机制

上面已经说过了，函数数据源中分为了报表子系统内部的特定函数和业务子

系统中的借口函数整理。而需要统一到函数数据源的是原始数据获取之后的数据整理和筛选部分。

当然，有的业务子系统提供的借口是支持特定数据筛选的，但报表子系统既然无法保证所有的业务子系统借口存在足够的数据筛选范围，就只能将数据筛选功能在报表子系统内部进行实现，这也是区别于 SQL 数据源最主要的一点。

报表子系统中的函数数据源内部处理机制的主要流程如图 5.1 所示，入口是统一的，然后调用相应的报表子系统的内部函数或者业务子系统中的接口函数。当然如果没有相应的函数就只能返回默认的数据了，一般为空数组（array()）。然后以获取的数据为基础，通过数据筛选和调整的配置，在对结果数据进行进一步的整理，最终将结果数据输出。

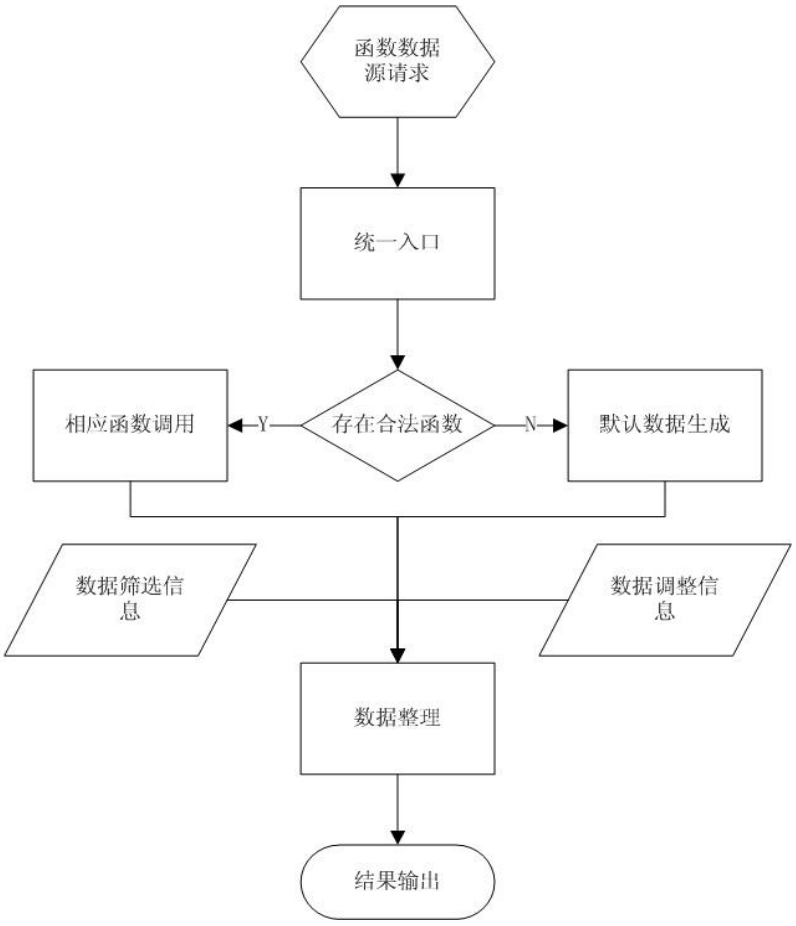

图 5.1 函数数据源内部处理流程

5.1.4 集合数据源的内部机制

集合数据源是需要着重说明的一种数据源，因为大多数报表都不是简单逻辑能够实现的，另外受限于基础数据源所能够承载的逻辑复杂度以及业务子系统的数据分离，所以对于集合数据源的功能要求就更加高了。

集合数据源是通过既定的算法将若干个数据源的数据进行合并，达到数据集合的目的。数据源之间的集合，从本质上说是一种对于多组数据进行的甄选和重新组合，其中对于数据甄选起关键作用的数据源，在报表子系统中被称为“关键数据源”，而对于数据的重新组合，通过总结，这其中的算法总共分为以下五种，下面就各种合并机制进行详细的说明，应用相同的数据源通过不同的算法获取的不同的预期结果。

5.1.4.1 接续（continue）

数据源之间的接续算法，是五种集合算法中唯一的不考虑数据源之间数值关联关系的算法，算法示例如图 5.2 所示。

***source i**

index	value i
a	a_i
b	b_i

&

***source ii**

index	value ii
b	b_ii
b	b_ii_i
c	c_ii

→

source final (after continue)

index	value i	value ii
a	a_i	null
b	b_i	null
b	null	b_ii
b	null	b_ii_i
c	null	c_ii

图 5.2 集合数据源接续算法机制示意图

接续算法是一个纯粹的拼接算法，是一个可以不用考虑任何数据源关系而进行直接堆砌的算法。这种算法简单、直接，但是缺乏业务数据关联的承载能力。

5.1.4.2 命中（hit）

数据源之间的命中算法，算法示例如图 5.3 所示。

命中算法采用的是“单循环关联-命中即止”的算法逻辑，这种逻辑主要是为“关键数据源”进行数据补充而准备。将“非关键数据源”中的数据，通过关联关系合并到“关键数据源”中，冗余的“非关键数据源”数据将被丢弃。

*source i	
index	value i
a	a_i
b	b_i

&

source ii	
index	value ii
b	b_ii
b	b_ii_i
c	c_ii

→

source final (after hit)		
index	value i	value ii
a	a_i	null
b	b_i	b_ii

图 5.3 集合数据源命中算法机制示意图

5.1.4.3 交集（match-join）

数据源之间的交集算法，算法示例如图 5.4 所示。

*source i	
index	value i
a	a_i
b	b_i

&

*source ii	
index	value ii
b	b_ii
b	b_ii_i
c	c_ii

→

source final (after match-join)		
index	value i	value ii
b	b_i	b_ii
b	b_i	b_ii_i

图 5.4 集合数据源交集算法机制示意图

交集算法采用的是“多循环关联-命中有效”的算法逻辑，这种逻辑主要是为多个“关键数据源”之间进行约束筛选而准备。这个算法中的所有涉及到的数据源都是“关键数据源”，而多个“关键数据源”中都涉及的数据才是有效数据，其余数据将被丢弃。

5.1.4.4 偏集（half-join）

数据源之间的偏集算法，算法示例如图 5.5 所示。

*source i	
index	value i
a	a_i
b	b_i

&

source ii	
index	value ii
b	b_ii
b	b_ii_i
c	c_ii

→

source final (after half-join)		
index	value i	value ii
a	a_i	null
b	b_i	b_ii
b	b_i	b_ii_i

图 5.5 集合数据源偏集算法机制示意图

偏集算法采用的是“单循环关联-命中有效”的算法逻辑，这种逻辑和命中算法中的逻辑很相似，区别在于偏集算法中不会因为一条记录命中而终止循环过程，这样就可能将“关键数据源”中的数据进行扩充。

5.1.4.5 并集（full-join）

数据源之间的并集算法，算法示例如图 5.6 所示。

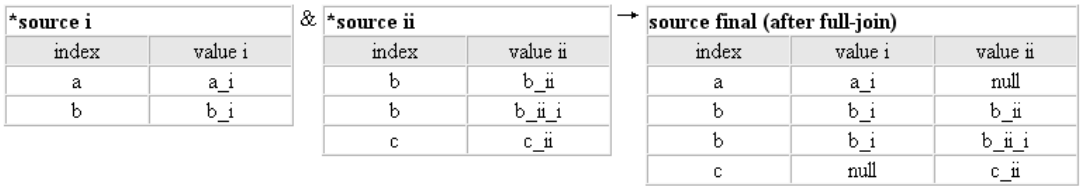

图 5.6 集合数据源并集算法机制示意图

并计算法采用的是“多循环关联-全命中”的算法逻辑，这种逻辑其本质就是获取多个数据集合的笛卡尔积。但从实际业务出发，这种逻辑是很少被应用到的，不过为了数据源集合的整体逻辑完整性还是需要将这种逻辑加以规划设计和实现。

5.1.5 遗传数据源的内部机制

遗传数据源是为了提高数据源通用性而存在的，使得基础数据源可以脱离特定报表环境而独立存在。报表可以通过遗传数据源来获取既定逻辑的数据，只需要传递数据范围、字段范围等配置信息。

可以说，数据源的遗传是一种过滤既定逻辑数据的机制，因为遗传的数据源是不支持在既定逻辑的基础上再进行逻辑扩展的，可以说遗传数据源的业务逻辑和数据是被遗传数据源的子集。

遗传数据源机制的设置，更好的维护了业务子系统中各种逻辑的通用性和独立性，使得报表和基础数据源的耦合度大大降低。

第二节 报表数据整理部分

对于报表原始数据的整理，是报表子系统数据获取部分中不可或缺的环节，尤其是在报表数据源可以分散存在的情况下。报表的数据整理是对报表业务逻辑的主要体现和补充，数据结构的最终确定。

5.2.1 报表逻辑行的数据整理

报表的逻辑行是报表业务逻辑体现的主要部分，它体现了报表对于数据的汇总侧重，进而体现了报表业务对于逻辑侧重。同样的基础数据，通过不同的数据汇总就可以体现不同的业务逻辑。

进销存子系统中提供的产品销售出库情况明细表如表 5.1 所示，展示的是进销存业务子系统中日常的销售出库活动的单据的记录，但是单纯的业务数据的堆砌就无法体现出对于这一业务活动的内在逻辑体现。于是就有了以下两种汇总方法。

表 5.1 产品销售出库情况明细表

出库单号	仓库	产品	出库量	出库总价
XC20130601001	南开仓	百事可乐	100	￥680.00
XC20130601002	南开仓	可口可乐	120	￥780.00
XC20130601003	和平仓	可口可乐	150	￥975.00
XC20130601004	南开仓	可口可乐	200	￥1,300.00
XC20130601005	和平仓	可口可乐	200	￥1,300.00
XC20130601006	和平仓	百事可乐	80	￥544.00

第一种是先按照仓库进行汇总，再按照产品进行汇总，如表 5.2 所示。这样的基础汇总仍然是按照仓库进行，然后在仓库汇总的基础上再进一步进行产品汇总。通过这样的数据汇总，可以清晰地看到各个仓库的出库情况，各个仓库中各种产品的出库情况，这样就可以获得当前期的产品出库情况分布，进而可以预期未来一段时间的产品出库情况分布。用这些数据可以及时的调整和控制各个仓库中产品的库存。

表 5.2 产品销售出库情况明细表（仓库产品汇总）

出库单号	仓库	产品	出库量	出库总价
XC20130601001	南开仓	百事可乐	100	￥680.00
产品小计：		百事可乐	100	￥680.00
XC20130601002	南开仓	可口可乐	120	￥780.00
XC20130601004	南开仓	可口可乐	200	￥1,300.00
产品小计：		可口可乐	320	￥2,080.00
仓库小计：	南开仓		420	￥2,760.00
XC20130601006	和平仓	百事可乐	80	￥544.00
产品小计：		百事可乐	80	￥544.00

续表 5.2 产品销售出库情况明细表（仓库产品汇总）

出库单号	仓库	产品	出库量	出库总价
XC20130601003	和平仓	可口可乐	150	￥975.00
XC20130601005	和平仓	可口可乐	200	￥1,300.00
产品小计：		可口可乐	350	￥2,275.00
仓库小计：	和平仓			￥2,819.00

另一种是先按照产品进行汇总，在按照仓库进行汇总，如表 5.3 所示。这样的基础汇总仍然是按照产品进行，然后在产品汇总的基础上再进一步进行仓库汇总。通过这样的数据汇总，可以清晰地看到各种产品的出库情况，各种产品中从各个仓库的出库情况，这样就可以获得当前期的产品总的出库情况和各个仓库的出库分布，可以有效地获取当期产品的出库总量和分库出库量，为当期的产品销售出库情况提供消息的数据。

表 5.3 产品销售出库情况明细表（产品仓库汇总）

出库单号	仓库	产品	出库量	出库总价
XC20130601002	南开仓	可口可乐	120	￥780.00
XC20130601004	南开仓	可口可乐	200	￥1,300.00
仓库小计：	南开仓		320	￥2,080.00
XC20130601003	和平仓	可口可乐	150	￥975.00
XC20130601005	和平仓	可口可乐	200	￥1,300.00
仓库小计：	和平仓		350	￥2,275.00
产品小计：		可口可乐	670	￥4,355.00
XC20130601006	和平仓	百事可乐	80	￥544.00
仓库小计：	和平仓		80	￥544.00
XC20130601001	南开仓	百事可乐	100	￥680.00
仓库小计：	南开仓			￥680.00
产品小计：		百事可乐	180	￥1,224.00

上面所显示的数据嵌套汇总，能够最大程度的体现数据逻辑和数据挖掘，为客户的决策提供丰富的数据支持。当然，较简单的单一情况汇总，也能在一定程度上反应业务数据中的逻辑。

5.2.2 报表逻辑列的数据整理

报表的逻辑列主要分为公式列和函数列两种。这里所涉及的逻辑列是数据源数据的有效补充，除去可以作为单一数据源中数据计算调整的备用方案，更多

的是为多数据源之间的数据计算的解决方案。

公式列的设置主要针对单一数据行范围内的数据计算。进销存子系统中提供的产品库存情况表，结构如表 5.4 所示，其中的“库存成本”就是一个公式列，其计算公式为：

库存成本＝成本×库存量

表 5.4 产品库存情况表

产品	成本	仓库	库存量	库存成本
可口可乐	￥6.50	南开仓	2,600	￥16,900
百事可乐	￥6.80	南开仓	1,800	￥12,240

函数列的设置同样主要针对单一数据行范围内的数据计算。进销存子系统提供的单据审核情况表，结构如表 5.5 所示，其中的“审核人”就是一个函数列，数据的内在逻辑是完全依赖于进销存子系统的相关权限部分功能。

表 5.5 单据审核情况表

单据	业务员	状态	审核人
XC20130601001	张三	一级待审核	1 组长
XC20130601002	李四	二级待审核	1 经理，1 主管
XC20130601003	李四	审核通过	--

当然，函数列的算法逻辑并不局限于业务子系统的接口，报表子系统本身同样也存在一些数据列算法逻辑。报表子系统规划有多列数据连接算法——类似于 SQL 语法中的 concat；多列数据枚举匹配算法——类似于 SQL 语法中的 case-when；多列数据汇总算法——类似于 SQL 语法中的聚合函数 sun、count、avg 等；同样根据数据汇总的需求，还可以将统计学上的期望、方差等概念也扩展到逻辑列的算法中来。

5.2.3 报表的数据引用

报表数据引用的功能规划同样是源于报表功能的优化。

进销存子系统提供的产品销售情况表，结构如表 5.6 所示，其中的“占比”是一个公式列，计算公式为“占比（%）＝（出库总价÷合计总价）×100”。占比的存在能更好的展现各个产品的销售情况和比较情况，是一个重要的统计指标，其中所蕴含的逻辑意义相较于出库总价要丰富的多。

表 5.6 产品销售情况表

产品	出库量	出库总价	占比（%）	合计总价
可口可乐	120	￥780.00	53.42	￥1,460.00
百事可乐	100	￥680.00	46.58	￥1,460.00
合计		￥1,460.00		

在这个计算公式中“合计总价”的数据可以从数据源中直接获取，但是直接获取，就造成数据源逻辑的复杂度提高，而且合计的逻辑就固定了，一旦报表需要区分仓库，结构如表 5.7 所示，就必须从数据源阶段修改合计总价的获取逻辑，虽然可以满足需求，但是可维护性偏低。

表 5.7 产品分库销售情况表

产品	仓库	出库量	出库总价	占比（%）	合计总价
可口可乐	和平仓	90	￥585.00	68.26	￥857.00
百事可乐	和平仓	40	￥272.00	31.74	￥857.00
仓库小计：	和平仓		￥857.00		
可口可乐	南开仓	30	￥195.00	32.34	￥603.00
百事可乐	南开仓	60	￥408.00	67.66	￥603.00
仓库小计：	南开仓		￥603.00		
合计			￥1,460.00		

鉴于此，报表子系统规划数据引用功能，可以通过设置动态引用合计行中的出库总价或者相应仓库小计行中的出库总价。这样就为跨行的逻辑运算提供了较好的解决基础，也间接的简化了报表逻辑列运算的复杂度。

5.2.4 报表整体的数据整理

除了上述的几种报表数据的逻辑整理部分之外，报表的数据整理还有其他几种整理过程，只是其中承载的业务逻辑相对较少。

一、报表中数据的横向化整理：

在本文第 2 章中的 2.4.2 中，已经有了明确的需求描述，所谓的报表数据横向化，就是将原本纵向顺序排列的数据（如表 2.2 和表 2.3 所示的“库存量”数据），通过某些固定键（如表 2.2 和表 2.3 所示的“产品”列）确定行域，填补到某个固定键（如表 2.2 和表 2.3 所示的各个“尺码”列）扩展的列域中相应的位置的过程。

二、报表中空数据的补全整理：

报表的原始数据都是源于数据源的获取，而数据库中存储的 NULL 数据，是无法应用于上面所说的公式或函数等逻辑运算的，这就需要在参与逻辑运算之前，将 NULL 转换成为相应的有意义数据，才能保证逻辑运算的正确性和可操作性。

三、报表中数据的排序：

报表数据的排序，在特定的报表中是存在其逻辑意义的，例如各类的 TOP-10 排名表，而且报表数据的排序，也能够有助于用户对于数据的有目的的浏览。

四、报表中数据的省略整理：

进销存子系统提供的采购单据（含明细）统计表，结构如表 5.5 所示，进销存子系统的业务允许一张单据可以存在若干明细数据，即单据表头部分和明细部分的数据条数比为 1：N，则会有一部分明细数据行存在冗余的表头信息，这部分信息的清除无论在视觉上还是在页面的布局上都是有必要的。

综上，对于报表数据的整理是一个复杂的过程，其逻辑性和时序性的要求也同样很高，将各种类型的调整有机的组合起来，形成一个结构严密，逻辑严谨的整体。当然不同的报表会存在不同的整理流程，而这就需要通过报表设置部分进行统一的设置。

第三节　报表数据渲染部分

报表数据的展示部分，是报表子系统对外提供的数据展示的页面和相应的机制。

除了基础的数据表格之外，展示页面还需要提供信息实时交互以及数据导出等功能。其中表格数据的请求是整个页面的核心，它肩负着向后台发起请求，传输所有相应交互数据的作用。

应用 flex 表格控件，完整的报表浏览请求顺序如图 5.7 所示。

第四节　报表设置部分

报表子系统的设置部分说起来很简单，就是要应用图形界面的交互方式提供报表子系统中数据源和报表的设置功能。但真正的规划和开发实现就并非如此

简单了。

报表子系统的设置又可以分为两个大部分，分别对应数据源设置和报表设置，其中数据源设置可以依托于报表，也可以独立进行设置。不过通常报表会间接应用一部分基础数据源以简化报表内部简单数据源的重复性，再应用一部分报表数据源以完成报表中特定的业务逻辑。

从设置角度来看，一张报表的设置流程如图 5.8 所示，大致可以分为九个部分。

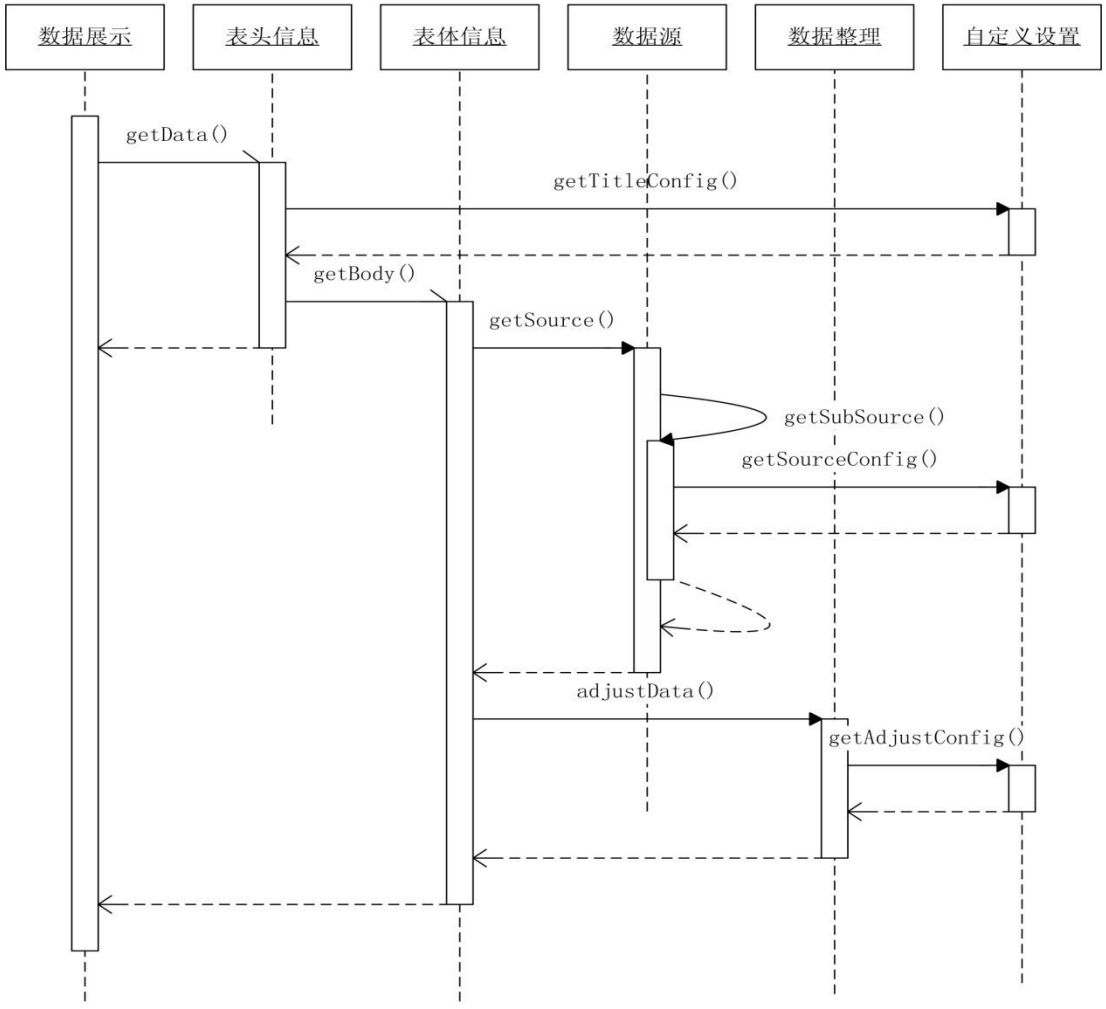

图 5.7　报表浏览请求时序图

其中第一、第三和第四部分都是涉及到数据源设置的，之所以被分割是因为第一部分“基础数据源设置”是可以独立于报表设置而存在的，第三部分“报

表数据源设置”是一个数据搜集的过程，第四部分“报表数据源集合设置”是一个数据整合的过程。通过第三部分的报表数据是一堆零散的“数据零件”，只有通过了第四部分之后，这些数据才会被邮寄的拼装集合起来，形成初步的标准二维结构。

第二部分“报表编号生成”是一张报表从无到有的一个状态变化，在报表的内部业务逻辑方面并没有提升。

第五部分“报表数据整理设置”承接第四部分的二维结构，对其进一步进行结构调整设置，可以增加复杂的报表业务逻辑，使报表数据更加丰满。

第六部分“报表数据展示设置”主要设置的是报表可是部分的内容、样式和规则。至此一张全新的报表可以说基本完工了。

第七部分“报表测试”会根据一定量的测试数据，试运行完工的报表得出测试报告，给出每个数据源以及各部数据整理的内存消耗和时间消耗，最终的消耗汇总以及报表整体的健康程度。这些数据可以为设置人员提供相应的参考信息，判断报表各部分存在问题的可能，为问题的追溯提供方向。

第八部分“报表上线”是报表在通过最后一轮报表测试之后的一个状态变更，也是报表交付的操作。

第九部分“报表维护设置”是报表在服役期间的微调设置，其功能范围是“报表数据展示设置”的子集，其中并不会涉及到报表核心业务逻辑的调整。

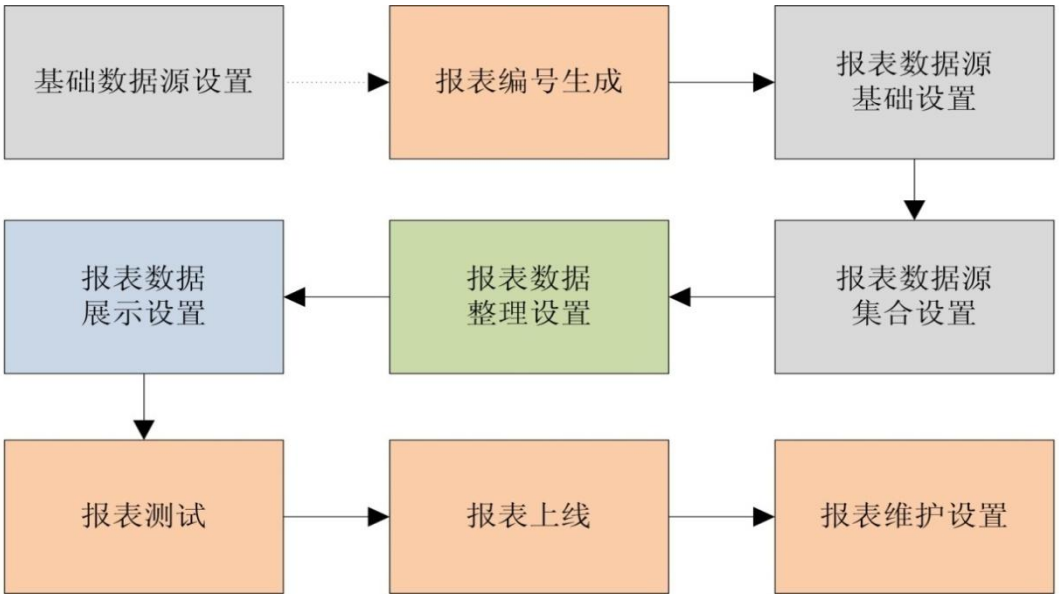

图 5.8　报表设置部分基本流程示意图

报表设置的功能规划是紧贴报表本身的功能规划的，一个强壮的易用的图形设置界面会给用户一个良好的体验，其中的困难在于要尽可能用简单的交互完成复杂的逻辑设置，而这也是在报表设置部分开发的时候遇到的最大的问题。

第五节 系统实现

报表子系统的开发过程是由功能模块开始，之后功能集成完成功能链，最后回到功能模块进行模块内部的拓展和挖掘。

5.5.1 报表展示页面

报表子系统提供的报表展示页面如图 5.9 所示。其中上半部分是遗留系统中的查询条件控件，下半部分是报表子系统开发的基于 FLEX 表格控件的数据展示控件。这部分展示功能，集中地体现了报表所有的数据获取、数据整理等后台功能，也是测试人员主要需要关注的数据结果页面。

图 5.9 报表子系统展示页

5.5.2 报表设置页面

报表子系统开发了两版报表设置界面，第一版是支持高级用户应用的，界面如图 5.10 所示；第二版是支持管理员应用的，界面如图 5.13 所示。

其中支持高级用户应用的设置页，会更多的关心报表子系统在业务子系统中的应用方式和相应配置。会有针对报表组的信息维护，对于报表的调整功能

也相对简单，对于报表的新增存在限制，这些特性都是根据“高级用户”的操作需求的权限范围所决定的。

5.5.2.1 高级用户设置页面

高级用户拥有完整的报表类别维护功能（如图 5.10 左侧部分所示），是因为高级客户更关心对于报表的浏览，相应的也就更关心对于报表的分类。

图 5.10 高级用户应用设置界面

高级客户对于业务子系统的报表新增存在限制（如图 5.11 所示），新增的备选报表源于“标准报表模板库”。而这所谓的限制，是相对于管理员来说的，高级用户可以根据业务子系统的需要，从“标准报表模板库”中复制相应功能的报表到业务子系统中，但是无权构建一张全新的报表。

图 5.11 高级用户新增报表界面

5.5.2.2 报表展示设置页面

新报表复制之后，高级用户可以根据业务需要对报表进行细部调整（如图5.12 所示）。

可以对报表整体部分信息进行维护：包括报表的分类、名称、备注、左侧固定列数以及零数据过滤方式，这些是报表的基础信息和整理过程中整体控制设置部分。

可以对报表二维结构中的行、列、格进行调整：可以对已有的报表结构进行调整，可以隐藏一部分数据列或数据行。可以增加自定义的列来扩充报表的业务计算，这种扩展目前只能支持数据行内公式运算，是在一个可控范围内的自定义扩展。可以设置单元格中的固定文本、相应的显示样式和数据格式。

可以对报表的数据范围进行有限的追加：可以根据已有的数据追加单张报表的特定数据范围，这就为业务子系统中权限分配结果进行配套报表提供了可能。

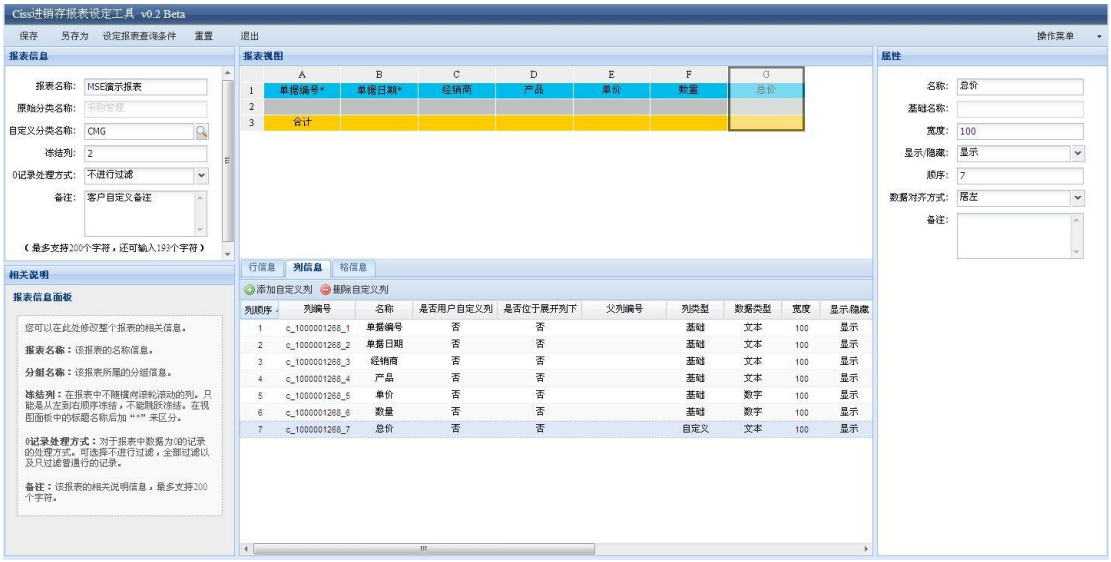

图 5.12 报表细部调整界面

可以对报表的查询条件进行调整：可以在已有查询条件的基础上对诸如查询条件名称，查询类型等进行调整。

可以对报表进行复制，用以获得一张相同的报表。

5.5.2.3 管理员设置页面

支持管理员应用的报表设置页，更侧重报表本身的结构构建和功能实现。

在此处的报表新增，将是不受“标准报表模板库”约束的新增。新报表会经过以下几个步骤的完善，最终正式上线使用。

首先是对报表数据源的设置，由于数据源的多种来源并存，所以设置界面也会因为各种不同种类数据源的特点进行设计。

图 5.13 管理员应用设置界面

5.5.2.4 数据源设置页面

报表数据源中的 SQL 数据源设置（如图 5.14 所示），需要通过数据库表构建 SQL 主体，然后对包括主表、关联表、需用字段、关联条件、数据范围、分组排序等功能设置。其中的关联条件设置以及数据范围设置都是非常复杂的。

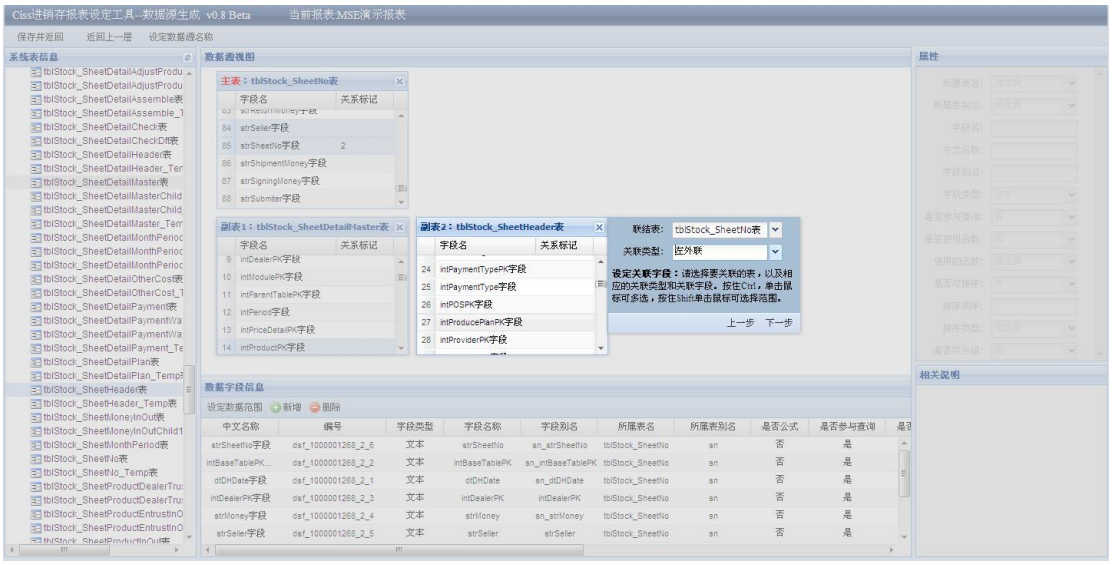

图 5.14 报表 SQL 数据源设置界面

报表数据源中的函数数据源设置（如图 5.15 所示），需要通过备选的应用

函数获取函数主体，然后对函数返回数据的应用范围进行进一步的设置。

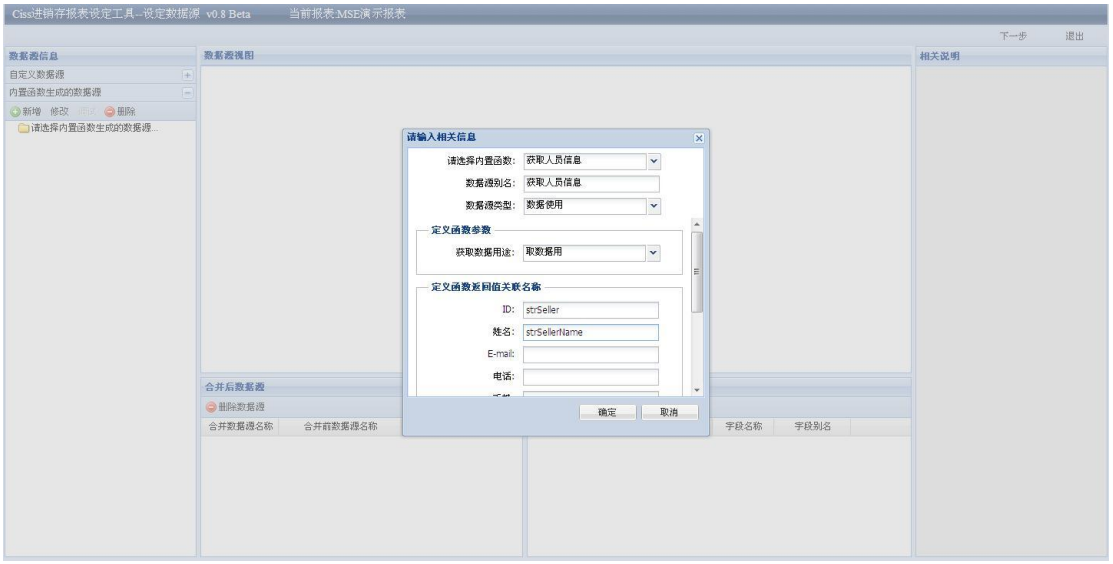

图 5.15 报表函数数据源设置界面

报表数据源中的集合数据源设置（如图 5.16 所示），主要考虑的是数据源之间的合并逻辑，以及关联合并逻辑中的关联字段配对设置。

图 5.16 报表集合数据源设置界面

5.5.2.5 数据整理设置页面

在通过数据源设置获得最终的数据源之后，就需要进行数据整理的设置过程（如图 5.17 所示）。在这个部分首先将散列化的数据源形式转换成为二维表

格结构，将数据源项直接对应到整理表格的列结构中，之后针对这个整理表格进行业务逻辑的附加整理。

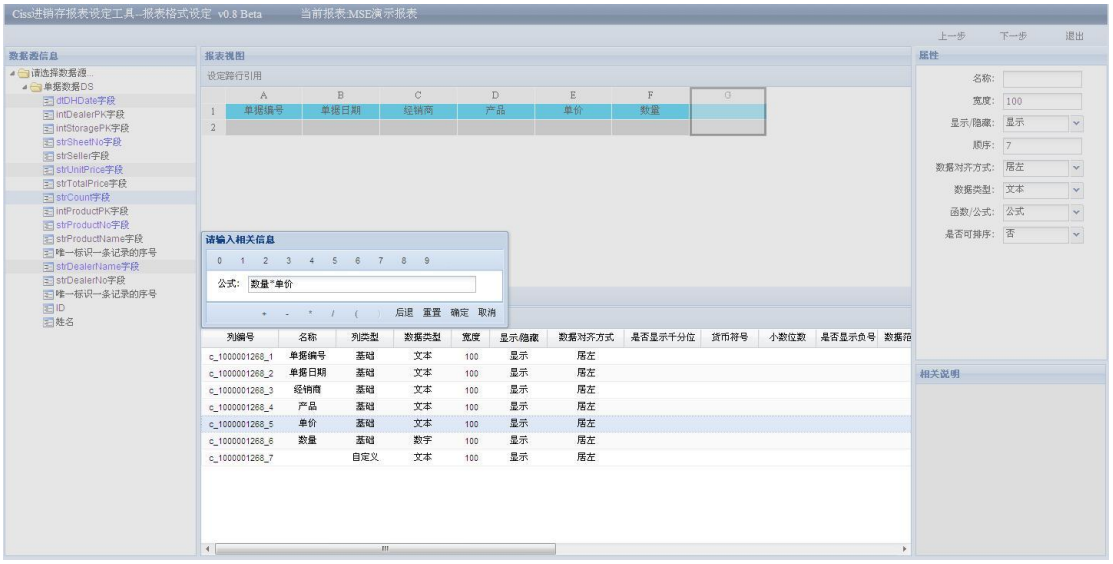

图 5.17 报表数据整理设置界面

在业务逻辑中比较重要的是逻辑行以及逻辑列的设置，图 5.17 中高亮的部分所展示的是逻辑列中的公式设置，其余还有函数逻辑列、引用逻辑列以及逻辑行等功能扩充。

图 5.18 报表数据源查询条件设置界面

之后是对于报表查询条件的设置过程（如图 5.18 所示），这里对于查询条

件的设置是直接从数据源项引入查询条件的，这样就可以直接的将查询条件绑定在数据源中，从而省却了由整理列结构转接分配的繁琐和可能出现的问题。

5.5.2.6 报表检验评估页面

报表的后台部分已经设置结束，在这个时间点需要对已有的设置进行试运行评估（如图 5.19 所示），针对各个数据源以及数据整理部分进行结果情况汇总，提供给开发者用以判断报表上线应用的可行性。这是一个必须的过程，只有经过评估合格的报表才能正式上线。

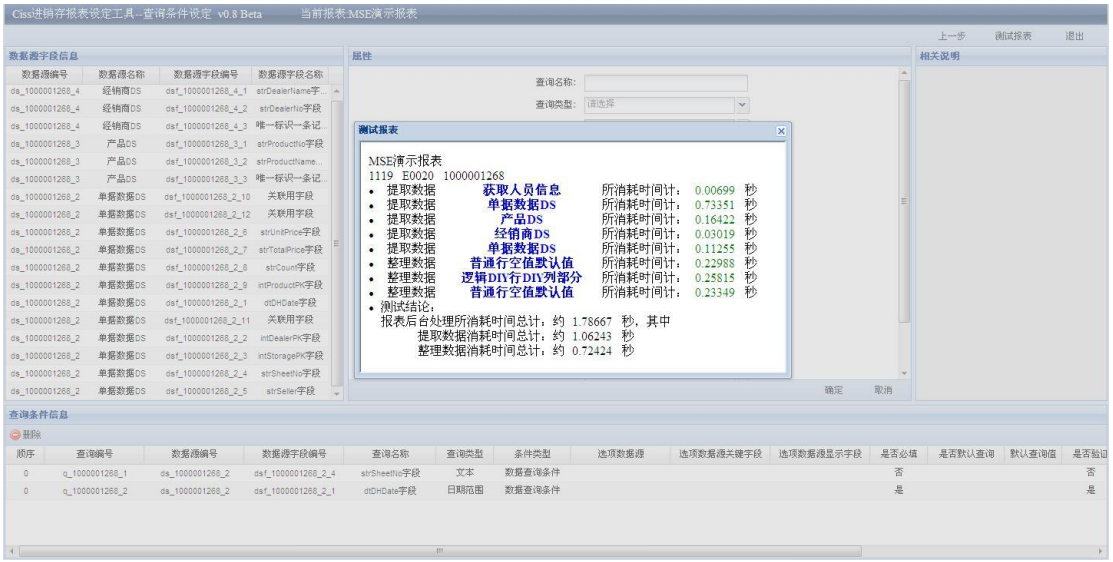

图 5.19 报表整体测试评估报告界面

对于报表的展示样式的调整在前面支持高级用户引用的设置中已经提到，如图 5.12 所示。

上述是一个完整的报表设置流程，也是支撑用户界面交互的主体。

第六章　系统中关键问题和技术

一个系统的开发会受到很多因素的干扰和制约，这里主要讨论的是系统开发过程中涉及到的技术性问题，期间虽然会由于工期或者应用既有控件等原因使得对一些问题的解决方案进行调整，但最总是要完成一个足够强壮和完美的报表子系统。

为了更快捷的开发，为了更好地提升报表的性能，尽可能的降低内存的开销和时间的消耗，在时间和空间的权衡中获得足够好的解决方案，同时使得报表子系统中各部分功能的逻辑更加清晰严谨，在系统的开发期间需要解决很多问题。

第一节　报表子系统开发流程

论文工作的过程也如同系统的开发过程，总会面临着这样那样的问题，而我们要做的就是尽最大的努力，在有限的时间内，以解决问题为目标，权衡时间和空间的开销，最终确定一个相对最优的解决方案，并将之实现。

之于一个工程或一篇论文，期间涉及到的问题，是障碍，也是阶梯，是从无到有的必然经历，更是核心价值的体现。

报表子系统的分部规划和分步实现的特点决定了报表子系统在开发过程中所采用的开发流程。

从前面的章节中不难看出，报表子系统的功能块是相互独立且存在明显先后顺序的，这就要求在开发的过程中，构建一条完整的报表运行原型功能链是非常必要的，之后才是这条功能链中各个环节的功能扩展和相应的设置，如图6.1 所示。

报表子系统的整个开发流程基本是遵循瀑布模型的，只是再系统开发和测试的环节中借鉴了一部分快速原型模型和增量模型的思想，从而形成了这种混合型的开发流程。

这种开发模式可以兼顾系统的整体性和各模块的功能性，可以缩短开发时间也可以在一定程度上增加测试的有效时间，为系统的高质量提供保证。

6.1.1 报表子系统的整体流程和思路

确定前进的方向往往比前进本身更重要，系统的开发自然也不例外。

具体到本文所讨论的报表子系统来说：最初的纯粹过程型的报表文件，“一种报表一个核心一个文件”的模式使得报表所涉及的各个功能块，在单张报表层面高度内聚的同时，在报表体系层面显得过于零散和随意；这种模式可以更好的控制数据结构对于内存的占用，针对零碎的变更需求也能够更灵活的调整。

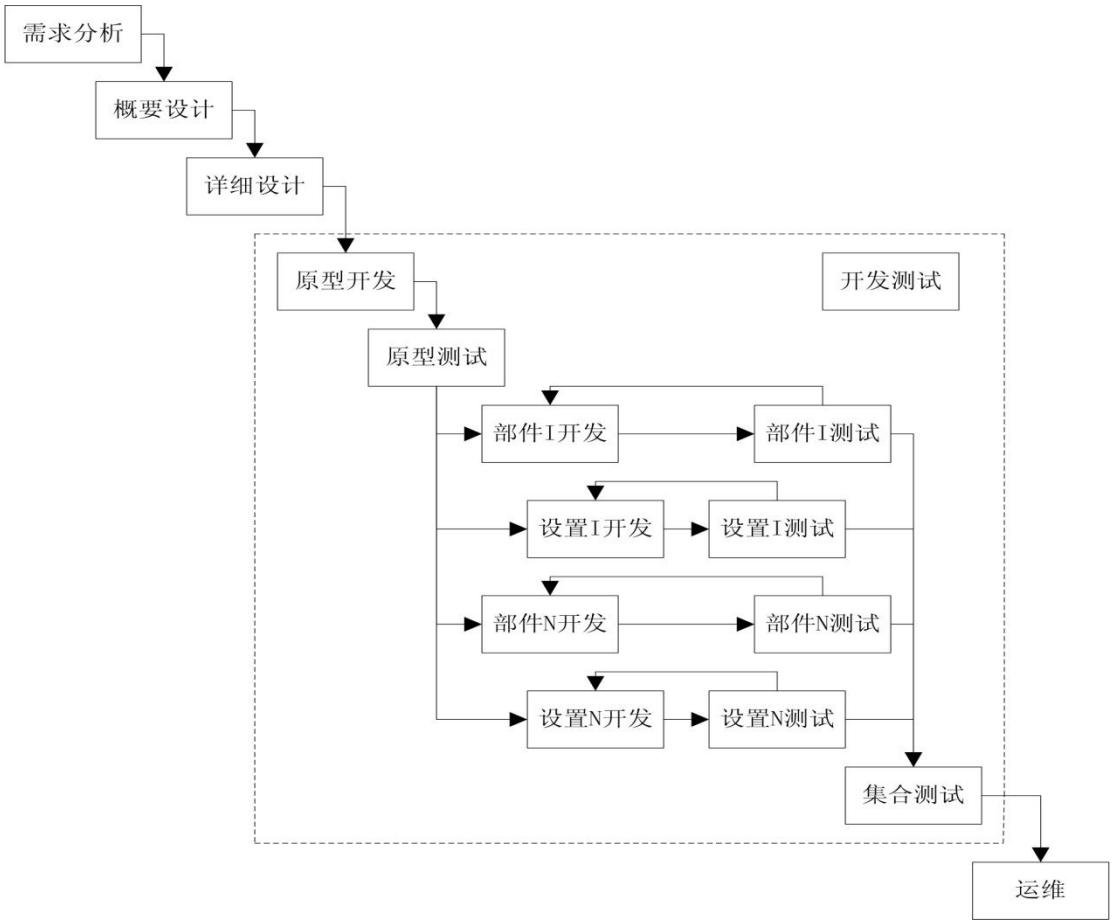

图 6.1 系统开发流程图

所以本文所涉及的报表子系统不会从根本上抛弃这种利弊参半的文件模式。可以根据报表各个功能块的通用性进行提取，抽象成为层级较高的功能模块核心；同时将报表核心进行独立文件存储，以保证数据结构和足够的灵活性与独立性。在所有独立文件之上存在统一的文件生成器，便于报表文件的批量修改

和重复报表的绝对独立。

相对独立的数据源，可以最大程度的统一业务子系统中报表使用的业务数据。实体报表应用的数据，源自若干独立数据源的整合，这就保证了业务子系统中数据的一致性和规则的一致性。

专用的整理流程，可以根据每张特定报表的不同情况进行不同的逻辑整理和调整，即使是不同的用户需求，也可以在互不影响的情况下，简单完成。

统一的展示平台可以保证报表数据的结构统一，展示样式统一，实时交互方式统一，使得报表子系统的界面通用性高，同时也促使展示界面变得简练和规范。

文件生成器可以保证报表文件的规格统一，逻辑运用方式统一，代码风格统一，使得文件的可读性和可维护性不因为报表需求的追加和累计而变差，这一点在子系统的维护时期是至关重要的。

6.1.2 用户与报表的应用模式

由于报表子系统要求同时为众多用户提供服务，而且其中大部分的报表是多数用户都需要的。那么在这个“多对多”的应用模式下，如何才能将通用性和个性统一起来，就成为决定报表子系统组织结构和运行模式的关键因素。

报表子系统为了凸显用户应用报表的独立性，而采用的各用户各报表单独文件配置的模式，在用户相同报表的通用性上，明显存在欠缺。为了弥补这方面的缺失，同时也使得报表子系统中的报表不至于太过于随意和零散，在用户应用报表的上层，存在一整套不限用户的基础报表的基准模板。这样既可以保存基础报表最初的形态，同时不妨碍各个用户对报表进行定制，又为基础报表保留了一个标准的版本，为新用户的报表数据初始化提供了一个快捷准确的数据来源。

当然，完全由用户自定义的报表不在此列。但是通过业务子系统运行的需求分析和用户反馈，也同样可以将用户的完全自定义报表略作改动以适用于更多用户，加以推广，扩充到报表子系统的“标准报表模板库”中。

6.1.3 开放用户参与报表设计

报表的最终目的是为用户提供一个数据汇总、分析的展示平台。而其中开

发人员能做的是通过对报表深层次的逻辑理解和分析，构建一个核心的报表原型，再通过进一步的数据约束、数据筛选、渲染方式固化，并提供实时数据交互的机制，最终完成一张成品报表。

这也是目前报表子系统，开发人员需要完成的。但是最终的结果是这样的：频繁的渲染方式变更，频繁的增减实时数据交互，频繁的数据约束扩展和变更，这些使得开发人员不胜其扰。究其原因，这些被频繁变更的部分，往往是和用户自身人事关系、规章制度、业务流程有着密切关系的。显然，这些属于由用户自身情况变更而造成的报表变更。如果这些变更都由开发人员进行维护的话，不论是对用户还是开发人员，两者之间沟通所花费的时间和成本都是不容忽视的。

因此才有了，报表子系统中开放用户参与设计报表的部分。由于用户专业背景的限制，对于数据源的设计并不适合全部开放。最先开放的部分主要集中在非核心的数据约束部分、渲染方式变更部分和实时数据交互部分等更贴近用户业务的部分。

这里借鉴了软件柔性的概念和精神，将报表子系统从刚性和通用性的道路上彻底扭转了过来。可以说柔性的思想贯穿了报表子系统的各个模块，进而通过报表设置部分提供的处理和交互接口进行统一的实现[23-25]。

第二节　报表数据源之间数据约束传递机制

通常情况下的报表是由若干个基础数据源集合而成的，集合的逻辑已经在本文 6.1.4 中进行了详细的阐述。在集合的过程中，基础数据源的数据，往往会因为集合的逻辑而被丢弃，这些对于最终结果毫无意义的数据，是数据源在获取数据是就应当剔除的，而不应当长时间的驻留在内存中。但是不同的基础数据源存在不同的数据约束，要想在最初获得精简的数据，就需要将其他数据源的数据约束移植到当前的数据源上，这个移植的过程就是数据源之间的数据约束传递。

以进销存子系统提供的产品库存情况表为例，报表的大致结构如表 6.1 所示。这是一个简单的数据展示报表，其中并没有复杂的业务逻辑。报表涉及三个数据实体分别是“产品”、“仓库”和“库存”，依据数据库第二范式的要求，其数据库结构如图 6.2 所示。其中 storage 表中的 strName 为报表中的仓库名称

信息；product 表中的 strName 为报表中的产品信息；stock 表中的 iCount 为报表中的库存量信息。

表 6.1 产品库存情况表

产品	仓库	库存量
可口可乐	南开仓	850
百事可乐	和平仓	630

对应进销存子系统中的数据库结构，报表子系统中涉及了“产品基础数据源”和“仓库基础数据源”，提供产品信息和仓库信息，在报表中可以直接应用于遗传数据源。报表中的库存量需要单据构建数据源获取，之后将三个数据源通过 idProduct 和 idStorage 应用命中逻辑进行集合，获得最终的数据结果。

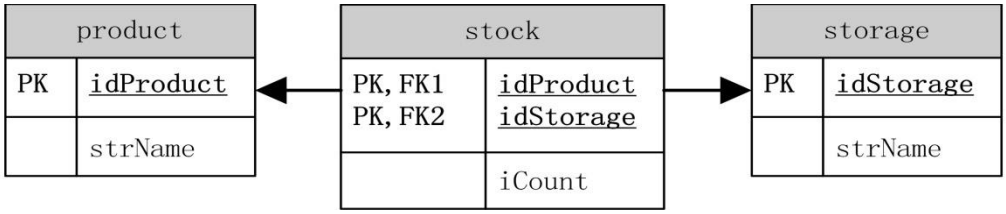

图 6.2 产品-仓库-库存实体关系图

现在的问题是，当客户需要查看“南开仓”相关产品库存的时候，会通过实时交互提交“仓库＝南开仓”，转换到数据源层面的 SQL 查询为“storage.strName=’南开仓’”。此时的产品基础数据源无条件约束，库存数据源无条件约束，就会使这两个数据源的临时结果存在大量的冗余。这就是数据约束传递机制要解决的问题。

应用数据约束传递机制后，在查询库存量时，会根据仓库基础数据源“storage.strName=’南开仓’”转化为“storage.idStorage=1”，进一步移植到库存量数据源“stock.idStorage=1”。同理在查询产品是会进一步根据“stock.idStorage=1”转化为“stock.idProduct in (1,2,3)”，而后移植到产品基础数据源“product.idProduct in (1,2,3)”。这样就可以在基础数据源获取数据时过滤掉绝大多数的冗余数据。

而且为了控制在产品和仓库同时存在实时交互的情况下可能出现的死循环，在数据约束传的中间还必须增加中断机制。否则仓库基础数据源的查询需要移植产品基础数据源，而产品基础数据源的查询有需要移植仓库基础数据源，那不仅不能提升报表整体的效率，反而弄巧成拙。

最后在数据源层面扩展两个属性分别用来标记该数据源是否向外发送数据约束和是否向内接受数据约束，默认是按照类似上述的规则进行数据约束的传递。这样就可以更精确有效地控制数据约束的传递，使之在系统设置的区域进行传播，能够更好的起到过滤数据的作用。

第三节　报表 SQL 数据源中的逻辑长句维护

对于 SQL 语句来说，所能承载的复杂逻辑主要来源于表的关联以及逻辑条件的复杂程度。对于关联表的多寡，在这里并不作为重点，因为关联表存在一个或者多个，对于整体的设计并没有带大的影响。

这里主要涉及的是关于 SQL 语句中复杂逻辑语句拼接的设计思路和解决方案。逻辑长句的拼接，关键是需要记录每个逻辑子句同逻辑长句之间的关系。

首先分别定义 A、B、C、D、E、F 为逻辑子句，模拟逻辑长句之间的关系为：(A or (B and C)) and D and (E or F)。然后按照规则将逻辑长句分解，在报表子系统中，这个分解的过程借鉴了数据结构中“树”的概念，将逻辑长句按照逻辑级别分解到一棵树中，如图 6.3 所示。

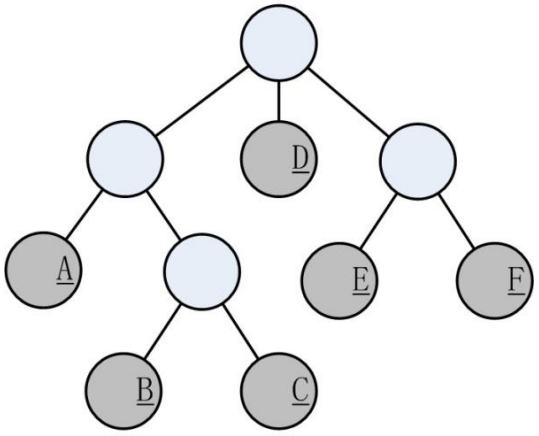

图 6.3　逻辑长句原始分解状态

在整体的树结构中进一步标记每个叶子（逻辑子句）的位置标识，通常的方法是将每一个分支进行编号，然后就可以得到每个叶子一个的唯一路径，从而确定各个叶子的绝对位置，如图 6.4 所示。得到每个叶子的位置标记为：

D=1; ——第一与逻辑层

A=0.0; E=2.0; F=2.1; ——第二或逻辑层

B=0.1.0; C=0.1.1; ——第三与逻辑层

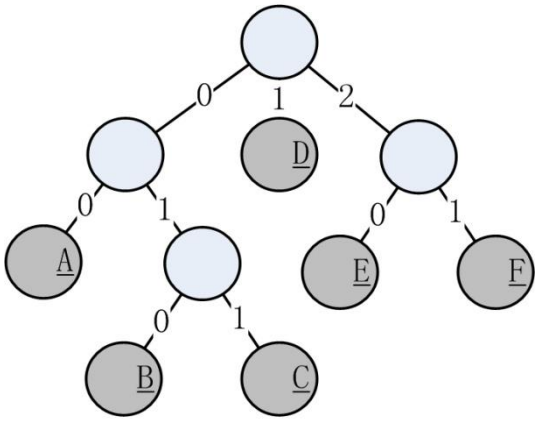

图 6.4 逻辑子句绝对位置模式标记

但是考虑到逻辑子句之间同级逻辑之间先后顺序并不重要，所以报表子系统中采用的是逻辑子句归属区域来确定每个叶子的相对位置，如图 6.5 所示。得到每个叶子的位置标记为：

D=0; ——第一与逻辑层

A=0.0; E=0.1; F=0.1; ——第二或逻辑层

B=0.0.0; C=0.0.0; ——第三与逻辑层

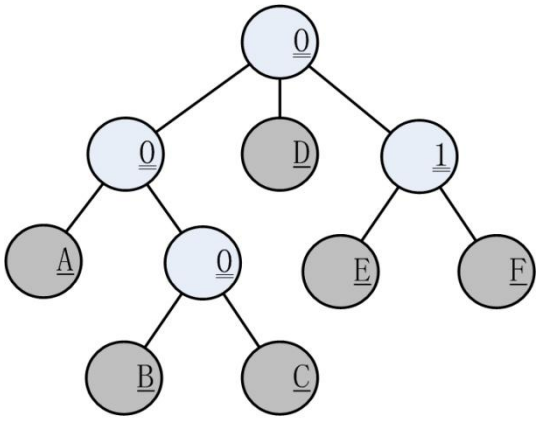

图 6.5 逻辑子句相对区域模式标记

按照逻辑级别来分的话，A 属于二级或逻辑，B 属于三级与逻辑，C 属于三

级与逻辑，D 属于一级与逻辑，E 属于二级或逻辑，F 属于二级或逻辑。之所以需要区分逻辑级别，是因为奇数级别为与逻辑，偶数级别为或逻辑。

当然应用逻辑子句的归属区域来标记逻辑子句位置，还有很好的可扩展性。本章第一节中提到的数据约束传递机制就会产生在报表设置之外的逻辑子句，这种计划外的逻辑子句在归属区域标记的算法中更容易被增加到树中。

第四节　报表数据整理的混合模式

报表数据的整理的各个整理部分在本文 4.2 中已经进行了详细的阐述。在这里要着重说明的是，报表数据的整理是一个有机的整体，不存在绝对的先后顺序，以进销存子系统提供的产品销售仓库分布表为例，其结构如表 6.2 所示。

表 6.2　产品销售仓库分布表

产品	南开仓		金额小计	占比（%）
	销售额	退货额		
可口可乐	￥1,950.00	￥325.00	￥1,625.00	54.44%
百事可乐	￥1,360.00	￥0.00	￥1,360.00	45.56%
合计	￥3,310.00	￥325.00	￥2,985.00	

报表中涉及的整理过程主要包括“金额小计”列和“占比”列的计算以及“合计”行的计算。由于存在了一维展开列“仓库”和“占比”，是的整个数据整理过程变得复杂和曲折。报表的原始数据结构如表 6.3 所示。

表 6.3　产品销售仓库分布表 - 原始结构

产品	仓库	销售额	退货额
可口可乐	南开仓	￥1,950.00	￥325.00
百事可乐	南开仓	￥1,360.00	￥0.00

首先，报表子系统提供计算一维展开列下数据的函数方法仅针对一个子列，所以需要先将每行的发生额计算出来，即应用“发生额＝销售额－退货额”公式进行计算。

第二，将所有数据按照“产品”列为锚点字段确定行域，“仓库”列为固定键确定列域，进行横向化，将“销售额”、“退货额”和“发生额”作为“仓库”列的子列进行展示。

第三，应用函数计算“仓库”列下“发生额”子列的合计值，生成“金额

小计”列。并对数据进行合计处理。

第四，引用“合计”行中的“金额小计”，生成“金额合计”列，在应用公式“占比＝（金额小计÷金额合计）×100%”计算“占比”列。

至此，报表的数据整理过程才告一段落，报表数据的结构如表 6.4 所示，其中“发生额”子列和“金额合计”列在展示的过程中将被隐藏。

表 6.4 产品销售仓库分布表 - 整理结构

产品	南开仓			金额小计	占比（%）	金额合计
	销售额	退货额	发生额			
可口可乐	￥1,950.00	￥325.00	￥1,625.00	￥1,625.00	54.44%	￥2,985.00
百事可乐	￥1,360.00	￥0.00	￥1,360.00	￥1,360.00	45.56%	￥2,985.00
合计	￥3,310.00	￥325.00		￥2,985.00		

由此可以看出，数据整理不会是一个一成不变的过程，它对应每一张报表都会是一个独特的整理过程，应用不同的整理顺序和机制。而报表子系统需要提供的就是足够应用的核心整理机制和一套可配置顺序的数据整理设置程序。

第五节 报表数据公式整理批量应用

具体到数据整理的公式应用部分，通常的做法是将原子数据替换到公式中然后运行之，这样的方法简单，但是面对着报表中大数据量的应用，就必然会造成一些不必要的时间开销，比如替换数据的过程和公式解析的过程等。

报表子系统采用的是公式批量应用的机制。应用统一的公式解析程序，生成一个承载该公式的临时函数，将原始数据数组（报表一行的数据）作为参数传入，函数内部进行公式的计算，输出最终的计算结果。

PHP 中提供的 create_function($_param, $_code);函数能够很简单快捷的生成所需的临时函数，为整个公用公式机制提供了强有力的保障。在这个过程中，对于公式的解析是最为重要的也是最为复杂的，这其中不仅要对公式自身的合法性进行检验，要对公式计算过程中可能出现的精度问题、除零问题等做出相应的判断和解决方案，还需要尽可能的兼容一部分不规范的书写方式。

这样就通过统一解析，降低了公式每次都需要解析的时间开销，通过函数的参数传递，降低了公式每次应用替换的时间开销，在大量重复公式计算的时候是有明显的优势的。

第六节　报表数据整理过程中的逻辑约束

报表子系统逻辑行运算的时候并不是简单的重复计算。考虑到数据列之间可能的函数依赖抽象记做（6.1），报表子系统提供解决数据间约束的函数 f 记做（6.2）。

其中 markvalue 为数据集合中 mark 项的值；linevalue 为整体数据集合；markfield 为 mark 项的名称；anchorfield 为 anchor 项的名称。

anchor→mark　　（6.1）

markvalue＝f(linevalue, markfield, anchorfield)　　（6.2）

以进销存子系统提供的采购单据（含明细）统计表为例，其结构如表 6.5 所示。

表 6.5　采购单据（含明细）统计表

表头部分			明细部分			
单据编号	供货商	整单金额	产品	数量	单价	总价
JR201306010001	精工超市	￥1,194.00	可口可乐	100	￥6.50	￥650.00
			百事可乐	80	￥6.80	￥544.00
JR201306010002	自采	￥1,440.00	芬达	120	￥6.00	￥720.00
			雪碧	120	￥6.00	￥720.00
合计：		￥2,634.00		420		￥2,634.00

其中为了不影响数据整理的其他过程，表头部分数据的省略是在最后才进行的，也就是说报表的整理结构如表 6.6 所示。对于“合计”行“整单金额”列的计算就不是简单将每行数据中“整单金额”进行相加。

表 6.6　采购单据（含明细）统计表 － 整理结构

表头部分			明细部分			
单据编号	供货商	整单金额	产品	数量	单价	总价
JR201306010001	精工超市	￥1,194.00	可口可乐	100	￥6.50	￥650.00
JR201306010001	精工超市	￥1,194.00	百事可乐	80	￥6.80	￥544.00
JR201306010002	自采	￥1,440.00	芬达	120	￥6.00	￥720.00
JR201306010002	自采	￥1,440.00	雪碧	120	￥6.00	￥720.00
合计：		￥2,634.00		420		￥2,634.00

由于“整单金额”依赖于“单据编号”，所以在合计“整单金额”时就需要考虑“单据编号”对于数据的约束，也就是在计算“整单金额合计”时应用的

是考虑单据编号约束后的结果。对应所提供的函数 f 具体到该报表，其中 linevalue 是报表行数据集合，markfield 是“整单金额”列标识，anchorfield 是“单据编号”列标识，markvalue 是最终得到的参与到合计运算中的整单金额。

第七节　报表数据组合排序机制

报表数据的排序，是数据整理中重要的一环，对于数据的排序可以有效地展现数据之间内在的逻辑关系。很多页面控件，比如 flash 表格控件、ExtJS 表格控件等等，也同样支持在页面的排序功能。

报表子系统提供的报表结构比显示用的二维表格要复杂得多，因为存在了不同级别的逻辑行——主要是各种粒度的小计行，横向化展开的枚举数据列，使得对于数据的逻辑排序变得非常的复杂。

首先逻辑行会把整体的数据分割成为一个个的小块，如果存在多级逻辑行那么这个分割的粒度将会更加细，不过由于规则的限制，不同级别间的数据范围是递进式的收窄，这也就避免了不同逻辑行所代表的数据块的部分重叠情况的出现，如图 6.6 所示。

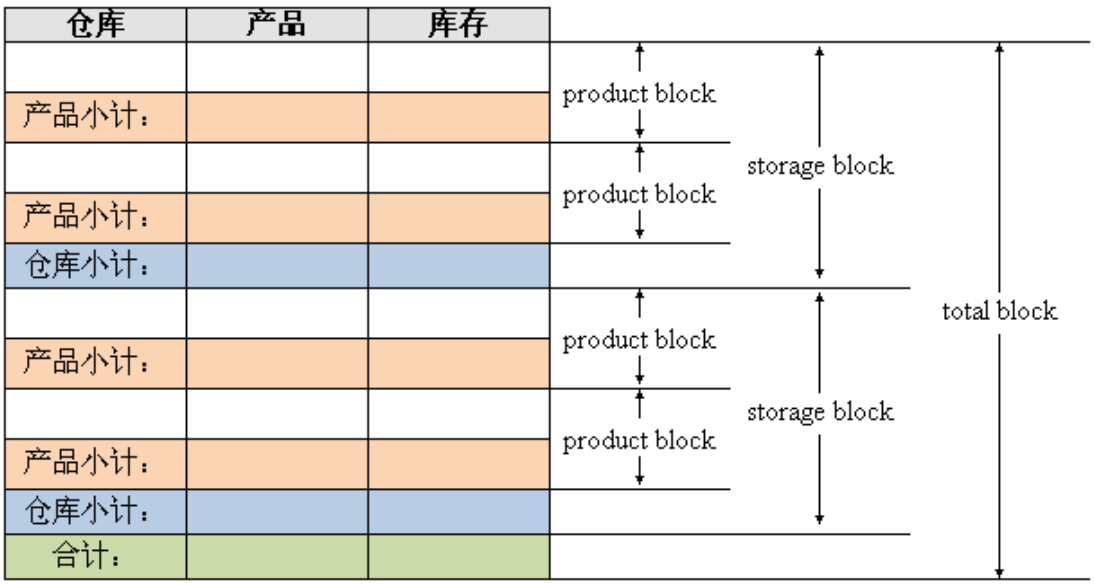

图 6.6　报表组合排序分块示意图

第二，报表子系统的排序功能是局限在顶层的数据列的，像横向化的枚举

数据列是不需要排序的，而这些列下的子列由于会存在于每个枚举数据列之下，所以对于这些子列的排序设置是无法锁定一个明确的数据列进行操作的。

最终确定的数据排序机制分为了两个部分，一个是对应逻辑数据行的“数据块排序”，另一个是对应基础数据行的“（块内）数据明细排序”。

以进销存子系统提供的产品分库销售情况表为例，其结构如表 6.7 所示。报表要求的数据排序是按照“出库总价”降序排列。具体的解决方法是先针对“仓库小计行”所在的数据块按照“出库总价”进行降序排列，之后在每个数据块内部再按照“出库总价”进行降序排列。

表 6.7　产品分库销售情况表

产品	仓库	出库量	出库总价	占比（%）	合计总价
可口可乐	和平仓	90	￥585.00	68.26	￥857.00
百事可乐	和平仓	40	￥272.00	31.74	￥857.00
	和平仓小计		￥857.00		
可口可乐	南开仓	30	￥195.00	32.34	￥603.00
百事可乐	南开仓	60	￥408.00	67.66	￥603.00
	南开仓小计		￥603.00		
合计			￥1,460.00		

鉴于此，报表子系统规划数据引用功能，可以通过设置动态引用合计行中的出库总价或者相应仓库小计行中的出库总价。这样就为跨行的逻辑运算提供了较好的解决基础，也间接的简化了报表逻辑列运算的复杂度。

第八节　报表文件生成器的原理和应用

报表子系统已经被分为基础的数据源、数据整理和数据展示三个相互独立的模块。具体到一张报表的时候，对应每个模块都需要相应的配置参数来参与驱动。

针对数据源模块，需要生成关于基础数据源应用、自定义数据源配置和数据源集合方式的文件；针对数据整理模块，需要生成关于各项既定整理过程驱动参数的文件；针对数据渲染模块，需要生成关于展示样式标准和应用数据的文件。

报表需要的所有的参数信息都将通过文件生成器生成格式化的文件，这样

可以通过图形交互界面对各个参数的细节信息进行设置，使得开发人员可以尽可能的脱离编写代码而实现报表的功能需求和改进。这是实现报表柔性的最为关键的一步。

报表文件的独立存储是有利于简化报表开发和维护的。由报表文件生成器统一生成的标准制式文件更能把其中的优势发挥出来。

报表文件生成器的主体主要有两个部分组成，文件系统操作部分和报表逻辑解析部分。其中的文件系统操作部分，通过 PHP 中 file-system（文件系统）相关函数进行功能封装，完成对于文件的新增、删除、修改以及读写权限控制功能。

报表逻辑的解析部分是将报表设置中的配置信息转化成为功能代码或者程序变量的集合。其中涉及到的函数化功能设置近 50%，并且涵盖了几乎所有参数化功能设置，对于报表设置的整体数据流如图 6.7 所示。因此，可以说生成器中的逻辑解析部分是整个报表子系统的核心。

对于报表文件生成器的应用总是需要配合报表设置和报表测试一同进行的。其中报表设置部分为报表文件提供所有必须的参数设置，报表测试部分为生成的文件验证逻辑正确性和业务正确性。

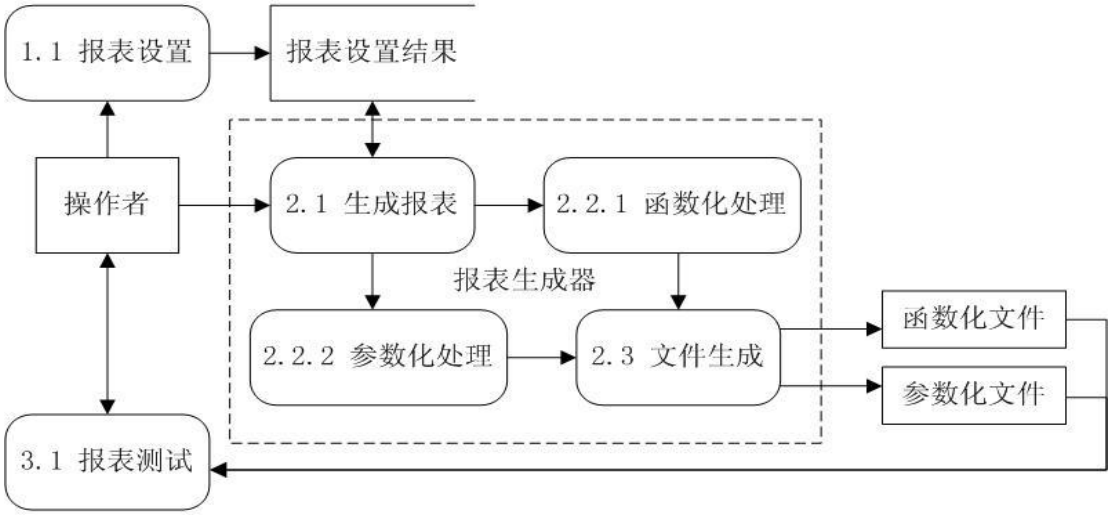

图 6.7　报表文件生成器数据流图

由此可见，报表生成器在报表子系统中有着举足轻重的地位，所有的报表都需要通过生成器，把相应设置转化成为文件，才能继续下面的调整步骤，进而上线使用。报表生成器也是把程序员从报表开发的编码方式转换成为交互设

置模式的关键。

第九节　报表数据渲染多样化

报表浏览的主要方式是二维表格，但是二维表格的生成又存在了多种的方式。报表子系统中仅涉及了 html 表格对象和 flex 表格控件量种。

对于html表格对象来说，报表的数据展示是通过请求将数据完整的通过table标签进行堆砌，其中需要考虑很多样式的渲染，在编写的过程中是比较复杂的。但是考虑到目前类似 iPad 的设备对于 flex 的支持情况，html 版本的表格展示还是有其必要性的。下面重点针对 flex 表格控件的使用进行阐述。

报表浏览过程的主要由表头请求和数据请求两个请求依次完成的。首先在浏览页面上需要增加 flex 表格控件对象——文件是已经编译好的 flashgrid.swf。

```
<object type="application/x-shockwave-flash" data="flashgrid.swf">
    <param name="movie" value="flashgrid.swf" />
    <param name="wmode" value="window" />
    <param name="quality" value="high" />
    <param name="bgcolor" value="#ffffff" />
    <param name="allowFullScreen" value="true" />
    <param name="allowScriptAccess" value="always" />
    <param name="FlashVars"
value='ColInfo=trans.php?param={"no":"***","action":"head"}'/>
</object>
```

上述对象代码加载时会发起一个获取报表的表头数据请求。后台的响应将通过一个 XML 流进行反馈传输，结构如下：

```
<menus label="parent"
dataProviderUrl="trans.php?param={"no":"***","action":"data"}">
    <col label="数据列 1" width="*" dataField="_column1" dataType="text"
strAlign="left"/>
    ……
    <col label="数据列 N" width="*" dataField="_columnN" dataType="date"
strAlign="right"/>
```

```
    <col label="" width="100"/>
</menus>
```

为了对于表格整体列宽的控制，在所有有效的数据列之后，会增加一个占位用的空数据列。这时 FLASH 表格控件会初始化表格的列，之后将会发起一个获取报表数据的数据请求。后台的响应将透过一个 XML 流进行反馈传输，结构如下：

```
<data>
    <row>
        <_column1>***</_column1>
        ……
        <_columnN>***</_columnN>
    </row>
    ……
    <row>
        <_column1>***</_column1>
        ……
        <_columnN>***</_columnN>
    </row>
</data>
```

根据各行的标签与表头数据列的 dataField 相匹配，最终命中数据位于表格中的单元格位置，完成报表的表格展示工作。

其实对于二维表格的生成方式是多种多样的，jQuery 框架和 ExtJS 框架等都提供二维表格的控件，而且其功能都很强大。但是考虑到这些框架对于复杂结构表格的支持情况以及既有功能与报表子系统中部分功能需求存在冲突，所以在初版规划时并没有考虑这两种。不过为了以后程序的可扩展性和展示多样性，报表子系统的展示页面将数据展示部分以独立文件方式处理，这样展示页面随时可以填充任何形式的展示部件。

第七章　总结和展望

报表子系统的开发过程也是对于系统设计和功能规划的总结过程，是对数据结构、功能结构和系统架构的检验，也是为后续子系统可能的升级积累经验和触发灵感。

第一节　总结

总的来说，报表子系统的规划和开发是成功的。很好的解决了各个业务子系统之间报表功能不统一的问题，解决了报表部分维护复杂以及需求相应时间长的问题，解决了报表针对业务子系统之间的数据获取方式随意混乱的问题。

报表子系统的开发完成，意味着 SaaS 系统上有了统一独立的程序对业务子系统的报表需求进行支持，并可以在全局的高度考虑业务子系统间的数据逻辑关系和业务逻辑关系，为业务子系统之间的数据关联提供了平台。

在报表子系统的开发过程中，积累了很多对于系统后续开发升级有益的经验和教训。其中对于 FLEX 表格控件的扩展开发，以及对于 EXTJS 框架的应用，都为后续系统调整打下基础。数据源的生成和应用模式也为后续系统调整提供了思路。报表子系统中的模块分布对于信息管理系统来说是有一定的借鉴意义的。

第二节　存在问题和解决思路

当然，报表子系统中还是存在一些问题的。

首先，报表的展示部分，最直观的只有表格形式的展示，这就显得有些单调，而且在一些特殊的业务场景，数据的汇总是需要图标来进行展示的。

进销存子系统中的例如销售趋势、库存分布、数据挖掘等高逻辑复杂度数据的简单展示情况，应用图表相较于表格，不管是从直观的视觉角度，还是对数据间逻辑比较的展示部分，都要更加适合。

客户关系管理（Customer Relationship Management，CRM）子系统中就有一

张很重要的数据汇总“客户漏斗”，这个图表中的数据自上而下，表示的是“意向客户”、“接洽客户”以及“签约客户”等客户关系的各个阶段。每个阶段的客户数逐渐递减用以表示客户关系的状况。由于目前的报表子系统没有图表的展示形式，这个部分就只能在 CRM 子系统中自行编制，虽然他的数据内核完全可以用报表子系统中的数据源来获取。

由此可见，对于数据的多样式展示是一个很实用而且必要的功能。在报表子系统的整体结构上，增加图表展示的部分是比较简单的，可以在数据整理之后，将最终的数据应用于图表展示模块，也就是数据展示模块中表格和图表处于同等的并列地位。

对于图表展示的功能规划和数据库结构规划，可以借鉴 EXCEL 等对图表功能有成熟设计的商务软件进行规划。

另外，报表的打印也是一个问题。目前的报表子系统是不提供数据打印的，如果需要打印，就只能先将数据导出，在应用 EXCEL 的打印功能进行打印。这对于用户来说是不够方便的，之所以做这样的处理，主要是对于不定宽度和长度的数据进行打印，其在打印过程中的布局是很难确定的，直到目前也没有很好的独立解决方案。如果需要解决打印的问题，最好应用独立的第三方控件来完成，这样可以抛弃复杂的布局计算，而且可以较好的完成打印的任务。

第三节　未来发展和进一步措施

报表子系统的开发过程，因为是对于信息管理系统整体的思考过程。

信息管理系统，简单的说就是对于数据的逻辑性处理和存储的系统，其中对于数据库的操作是没有太大差异的，这和报表子系统中的数据源部分的功能很类似，只是信息管理系统所需要的是对于数据库中数据增、删、改、查等方法，而目前的数据源部分只提供了数据提取过程。

所以对于未来信息管理系统以及报表子系统的规划，是需要把数据源部分独立成为数据源子系统的，用以对所有的系统提供针对数据库的所有数据操作。数据源可以通过设置程序进行设置，之后通过生成程序直接生成相应的类文件，为各个业务子系统提供数据支持。可以理解为在业务子系统和数据库之间构筑一层中间层来提供服务。

这也是为整个 SaaS 系统提供统一数据库操作接口的一个解决方案。

参考文献

[1] 曹军威，范玉顺，吴澄. 新一代 CIMS 应用集成平台系统体系结构. 清华大学学报（自然科学版），1999， 39（7）：68-71

[2] 方叙生，沈平. 柔性化 MIS 系统的设计与开发. 南京航空航天大学学报，1993，25（5）：597-599

[3] 王元珍，汪皓. 达梦智能报表工具的设计与实现. 计算机工程与应用， 2001，37（4）：65-67

[4] 万琳，陈传波. 智能报表生成系统模型的研究与实现. 计算机应用研究，2000，17（5）：25-26

[5] 郑志琴，钟叔玉. 柔性 MIS 及其支持技术. 昆明理工大学学报，2001，26（2）：8-11

[6] 张巨俭，甘仞初. MIS 系统中报表系统设计方法探讨. 计算机工程与应用，2003（4）：207

[7] X. De. Groote. The Flexibility of Production Process: A General Framework. Management Science，1994，40（7）：933-945

[8] Matsuura S，Kuruma H，Honiden S. EVA: A Flexible Programming Method for Evolving System. IEEE Transactions on Software Engineering，1997， 23（5）：196-312

[9] 何丽，申利民. 基于柔性的企业信息发布平台的开发研究. 2007 通信理论与技术新发展——第十二届全国青年通信学术会议论文集（上册），2007

[10] Dasgupta A. Flexible report generation and literate programming using R and Python’s docutils module，2010

[11] Shilin H, Sanqiang Z, Dengsong S，et al. Development Research of Flexible Report Subsystem in MES. Manufacturing Technology & Machine Tool，2010，8：37

[12] Rybkin AV，Wilson M. A web-based flexible communication system in radiology. Journal of digital imaging，2011，24（5）：890-896

[13] Hays C A，Carlson J D. Systems and methods for flexible report designs including table. Matrix and hybrid designs. U.S. Patent 7,707,490，2010-4-27

[14] Martin Fowler. Refactoring: Improving the Design of Existing Code. 北京：中国电力出版社，2003，8

[15] 文华. 提高 ERP 系统中报表运行效率的研究与实现. 铝加工，2009，2（187）

[16] http://www.w3school.com.cn/php/php_ref_filesystem.asp

[17] 陈媛，徐传运，孙曙光. 基于 Ajax 的 Web 通用报表生成工具的设计. 重庆工学院学报（自然科学版)，2009，4

[18] 申利民，张鹏，李峰. 基于 web 服务的报表模型研究. 燕山大学学报，2012，2

[19] http://www.redhat.com/

[20] http://www.apache.org/

[21] http://www.php.net/

[22] http://dev.mysql.com/

[23] 刘曙光，陈荣秋，鞠静. 柔性的比较性定义和性质. 华中理工大学学报，1997，5（210）：41-44

[24] 申利民，穆运峰. 软件柔性的概念和度量. 计算机集成制造系统，2004，10（10）：1314-1320

[25] 申利民. 柔性软件开发技术. 北京：国防工业出版社，2003：7-83

致谢

本论文在****导师的悉心指导下完成的。导师渊博的专业知识、严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，严于律己、宽以待人的崇高风范，朴实无华、平易近人的人格魅力对本人影响深远。不仅使本人树立了远大的学习目标、掌握了基本的研究方法，还使本人明白了许多为人处事的道理。本次论文从选题到完成，每一步都是在导师的悉心指导下完成的，倾注了导师大量的心血。完成论文的过程中，遇到的很多问题，在老师耐心指导下，都得以解决。在此，谨向导师表示崇高的敬意和衷心的感谢！

还要感谢学校所有辛苦工作、为我们精心安排每次学习与活动的老师们，感谢他们给予我们学业上无私的教诲和生活上亲切的关怀！

此外，本文研究和实践工作还得到了千乡万才科技（中国）有限公司各位同仁的热情帮助，他们在业务领域的经验给了本人极大的帮助，在此向他们表示真诚的谢意！

同时，尤其感谢多年来一直给予本人鼎力支持和无私奉献的父母，没有他们的付出与牺牲，本人的课题研究就谈不上顺利完成，再次真心地感谢和祝福他们！

最后，谨向所有在攻读工程硕士学位期间曾经关心和帮助过本人的老师和同学表示最诚挚的谢意！