
南京大學

# 研 究 生 毕 业 论 文
# （申请工程硕士学位）

论 文 题 目 基于反洗钱规则的结构化测试
数据生成系统设计与实现

作 者 姓 名 郭子琛

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 刘嘉 副教授

**2021** 年 **5** 月 **20** 日

学　　号：**MF1932061**

论文答辩日期：**2021** 年 **5** 月 **20** 日

指 导 教 师：  　　（签字）

# The Design and Implementation of AML Rule Based Structural Test Data Generation System

by

**Zichen Guo**

Supervised by

Associate Professor **Jia Liu**

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
Master of Engineering
in
Software Engineering


Software Institute
Nanjing University

May 24, 2021

# 学 位 论 文 原 创 性 声 明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不句含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：郭子琛

日期： 2021 年 5 月 20 日

# 南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于反洗钱规则的结构化测试数据生成系统设计与实现

工程硕士（软件工程领域）专业 2019 级硕士生姓名：郭子琛

指导教师（姓名、职称）：刘嘉 副教授

## 摘　要

在我国反洗钱领域，人民银行会定期对证券公司的交易数据进行稽查，要求公司上报洗钱交易数据。人民银行发掘真实洗钱数据并且与公司上报数据进行比对，如果公司存在漏报、误报的情况，则进行大额罚金处罚。其中，股票交易数据属于结构化数据，存储在关系型数据库中。由于表结构复杂，表与表之间关系繁多，可疑数据难以准确定位。因此，证券公司需要构造数据对自身交易合规监控软件进行测试。当前，测试数据通过反洗钱专员根据反洗钱规则进行人工构造，存在难以建模、难以泛化、覆盖率不足等问题。

本文依托于人民银行的反洗钱监管背景，根据相关证券公司真实需求，设计并实现了基于反洗钱规则的结构化测试数据生成系统，以期取代人工构造测试数据的模式。该系统的解决方案分为四个方面：第一，利用自然语言处理技术，对用户自定义的反洗钱规则文本进行关键参数抽取，进行自动化建模；第二，为用户提供关键参数选择功能，无需重构代码，提高数据生成的泛化性；第三，利用参数化组合规则技术，通过逐参数扩展算法生成规则集合，保证测试数据全覆盖；第四，对生成数据进行结果分析，保证测试数据质量。

系统划分为参数化组合规则模块、数据生成模块、结果分析模块和数据管理模块。系统的服务端基于 Django REST Framework 框架进行设计，支持用户自定义反洗钱规则文本和组合参数。数据生成方面，系统采用 Celery 和 Redis 消息队列对数据生成任务进行异步调度。最后，将生成数据存储在 MySQL 数据库中，以供结果分析模块和数据管理模块进行展示与操作。

实验案例证明，本系统能在合理时间内生成高质量、全覆盖的测试数据，可以对证券公司的交易合规监控软件进行全面测试，提高软件对洗钱数据检测的准确率。

**关键词：** 反洗钱，参数化组合规则，结构化数据，测试数据生成

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of AML Rule Based Structural Test Data Generation System

SPECIALIZATION: Software Engineering

POSTGRADUATE: Zichen Guo

MENTOR: Associate Professor **Jia Liu**

## Abstract

In the field of anti money laundering(AML) in China, the people's Bank of China will regularly inspect the transaction data of financial companies and require them to upload the money laundering transaction data. The people's Bank of China compares the real money laundering data with the data uploaded by the company. If the company fails to report data or misreports, it will be fined a large amount. Among these data, stock trading data belongs to structured data, which is stored in relational database. Due to the complexity of table structure and the various relationships between tables, it is difficult to locate suspicious data accurately. Therefore, financial companies need test data to validate their trading compliance monitoring software. At present, the test data is constructed manually by the anti money laundering commissioner according to the relevant rules, which is difficult to model, difficult to generalize, and insufficient coverage.

Based on the anti money laundering regulatory background of the people's Bank of China and the real needs of relevant financial companies, the thesis designs and implements AML rule based structural test data generation system, in order to replace the manual construction of test data mode. The solution of this technique is divided into four aspects. First, using natural language processing technique to extract the key parameters of user-defined rule text for automatic modeling. Second, providing users with the function of key parameter selection without refactoring code to improve the generalization of data generation. Third, using parametric combination rule technique to expand the algorithm parameter by parameter. And Fourth, analyzing the results of generated data to ensure the quality of test data.

This system is divided into parametric combination rule module, data generation module, result analysis module and data management module. The server of the system is designed based on Django REST framework, which supports user-defined rule text and combination parameters. In the aspect of data generation, this thesis uses the cell and redis message queue to schedule the data generation task asynchronously. Finally, the generated data is stored in MySQL database for display and operation of the result analysis module and data management module.

Experimental cases show that the technique can generate high-quality and comprehensive test data in a reasonable time, and can comprehensively test the transaction compliance monitoring software of financial companies, and optimize the accuracy of the software for money laundering data detection.

**keywords:** Anti Money Laundering, Parametric Combined Rules, Structured Data, Test Data Generation

# 目　　录

# 图目录

# 表目录

# 第一章　绪论

## 1.1　选题背景与意义

如今，随着金融科技（Financial Technology, FinTech）从应用系统建设逐步走向数据资产的集约利用和运营管理 [1]，金融数据已经成为政府、银行和公司打造核心竞争力，提高管理效能的重要手段 [2, 3]。金融数据作为 FinTech 智能软件运转的核心资源，其中最基础也是最庞大的部分是交易市场产生的结构化数据 [4]。结构化数据是指关系型数据库中的行数据，也就是存储在数据库中的数据，它可以通过二维表结构的方式逻辑表示和实现，例如大量流量数据 [5]、交易数据 [6] 和用户行为数据 [7] 等。

根据中国人民银行对于股票交易市场的反洗钱监管要求[①]，银行和相关证券公司投入了大量的人力物力资源来筛查交易数据中的洗钱数据。洗钱 [8]，是指通过多重途径和方式隐藏、伪装违法和犯罪情形以及相关获取金钱行为的非法交易活动。其中，贸易洗钱 [9]（Trade-based Money Laundering, TML）是借助贸易活动来实施的洗钱行为，其基本目标是将非法财产进行清洗与转移，以此获得合法收入。

由于结构化数据通常存储在关系型数据库中 [10]，在海量交易数据 [11] 环境下，存在表结构复杂、表与表关系繁多的问题，这导致通过人工检测方式难以识别异常数据。具体而言，在软件上线前和软件初启动的时间段，数据量稀疏，会面临数据缺失、数据覆盖率不足的难点；在投入市场运行后，随着相关软件的用户数量和行为呈指数级别增加，数据量愈来愈冗杂，软件会面临数据复杂、异常数据难以定位与分析的问题。因此，结构化测试数据对于智能软件测试起到了至关重要的作用。

经典结构化测试数据生成使用固定的规则 [12] 进行数据生成，可以生成满足指定规则的结构化测试数据 [13]。然而目前结构化测试数据生成技术基于固定规则，并不可以灵活改变规则的参数，也没有对规则解耦。基于固定规则的结构化测试数据生成技术有两个方面的问题：一方面，由于规则固定，会导致

[①]参照《金融机构反洗钱规定》相关条例。

生成数据的离散且有限；另一方面，当产生新的测试标准和规则时，需要进行相对复杂的代码重构来生成新的规则，泛化能力差。针对以上问题，本文将用户自定义的规则描述文本进行自然语言处理与分析，自动提取相关参数，并且组合成规则集合来进行结构化测试数据生成。

本文系统依托于在实验室期间和相关证券公司的反洗钱合作项目，针对金融领域股票交易市场的洗钱行为进行基于反洗钱规则的结构化测试数据生成。系统应用场景如图 1-1所示，从整体描述了文本系统落地的应用环境和实际需求。由于人民银行会对证券公司的交易数据进行监控，根据反洗钱法律规则发掘其中的洗钱数据，并对证券公司漏检漏报的洗钱交易行为进行大额罚金处罚，这对证券公司的交易合规监控软件识别洗钱数据的精准度提出了要求，证券公司希望能够做到精准上报洗钱数据，减少公司损失。本文设计并实现了基于反洗钱规则的测试数据生成系统，能够灵活按照用户输入的洗钱规则描述，快速模拟真实环境生成大量携带洗钱标签的交易测试数据来验证证券公司交易合规监控软件识别洗钱数据的准确性和可靠性，提高交易合规监控软件的识别精准度，减少证券公司误报洗钱数据造成的大额罚款和人力资源。

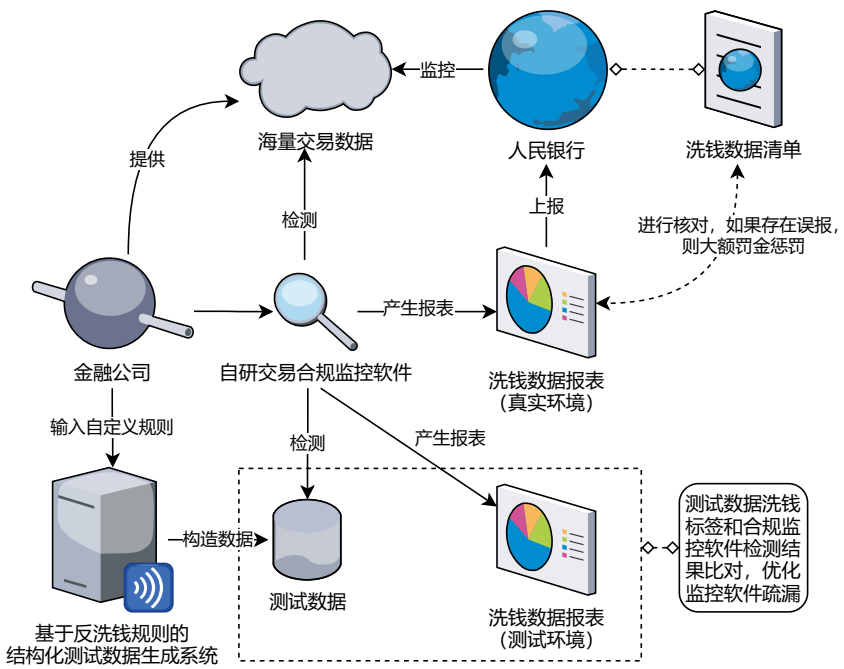


图 1-1: 应用场景视图

综上所述，本文以期能够一定程度解决反洗钱中领域结构化测试数据生成“难以建模、难以泛化、覆盖率不足”等问题，进行系统落地，帮助相关证券公司快速生成高覆盖率的测试数据，完善交易合规监控软件，减少漏检漏报的洗钱交易行为，节约公司的时间成本和人力成本。

## 1.2 国内外研究现状及分析

国内外研究现状主要分为四个部分，分别为反洗钱技术研究现状、文本关键参数抽取技术研究现状、组合测试技术研究现状和测试数据生成技术研究现状，目的是分析现有的结构化测试数据生成和参数化组合规则技术的在金融反洗钱领域的研究与应用。

### 1.2.1 反洗钱技术研究现状

反洗钱 [14]（Anti Money Laundering, AML），是指为防止犯罪以各种方式掩盖、隐瞒所得及其收益来源和性质的洗钱活动，应当依照有关法律规定采取相应措施的行为。

目前，相关发达国家在反洗钱方面，信息化技术水平很高，成果显著。美国建立了金融犯罪情报中心 [15]，该中心与美国所有金融机构和金融监管机构的数据库相连。在网络大数据帮助下，通过深度学习模型系统结合规则的模式筛选出可疑人员。调查人员有权限完全检查可疑人员的财务记录，工作条件和资产。通过合理的 RANK 算法 [16] 进行排序，从中抽取出用户的所有交易行为，为调查人员提供支撑线索和分析基础。英国政府研发了财务状况数据库 [17]，用来检索可疑人员并且收集信息、分析和报告可疑交易，整体检测模式仍然为规则筛选和人工审查为主。

中国一直在稳定控制外汇，因此国内洗钱情况相对而言不算猖獗。但是随着国家最高管理模式的逐步转变和股票交易市场的逐步开放，股票交易市场的洗钱行为受到越来越多的关注，国内银行的反洗钱制度 [18] 也在不断完善。目前，证券公司对于精细化、细粒度的股票交易数据的洗钱行为检测流程是采用交易合规监控软件 [19]，根据人民银行的洗钱规则，采取人工方式编写大量复杂繁琐的检测规则正则表达式对结构化数据进行识别，筛选出可疑人员，然后让反洗钱专员逐条审核可疑人员并作出判断。由于人民银行洗钱规则十分复杂，规则繁多，并且会定期对判定阈值作出改变。因此公司的检测规则正则表

达式需要耗费大量人力和时间去进行维护和更新，并且存在很多漏检误检的情况。为了保障自动化交易合规监控软件对规则检测的正确性和精确度，证券公司需要生成大量测试数据去进行测试与评估。目前，尚未有十分成熟的股票交易领域基于反洗钱规则的测试数据生成系统，只能依靠人工构造测试数据的方式进行小规模测试。

### 1.2.2 文本关键参数抽取技术研究现状

文本的关键参数，也叫参变量。对于规则描述文本而言，其中的关键参数的变化以及它们之间的相互关系指明了文本的具体意义。总结而言，关键参数是表示文本意义的不可再分单位，即最小单位。从总体角度划分，关键参数的获取存在两类可行方法：第一类为关键参数分配（Keyword Assignment, KA）[20]，关键参数是从受控词汇表或分类规则中进行选择；第二类是关键参数抽取（Keyword Extraction, KE） [21]，即关键参数是从原始文本中明确提到的单词中进行选择。正如 Beliga [22] 等人在 2014 年发表的文章中所述，关键参数抽取方法分为监督、半监督和无监督三种，监督和半监督方法都需要外界知识的支撑，无监督方法仅利用文本自身信息，不需要外接知识进行辅助。

自动抽取关键参数方法包括三个阶段：第一阶段是文本预处理 [23]，它涉及文本（集合）的组织、格式处理、分词、去除停用词等；第二阶段是候选词的选择，根据统计数据，其中的连词、助词、叹词几乎不作为关键参数，因此可以根据语言规则筛选出文本中的单词（短语），构成候选关键参数集合；第三阶段是评价和确定关键参数，对于每个候选词，通过特定方法或方法的组合，计算其特征，确定其是否为关键参数，并对其进行判断、评分或评分，将大于概率阈值的候选词和评级或得分高的候选词，按数量要求确定为关键参数。

经过调研发现，目前缺乏为反洗钱领域规则文本设计的关键参数语料库。本文将现有的文本关键参数抽取方式结合金融领域文本的语法特性，构建独有的外部知识库支撑，从而有效从候选参数中提取关键参数。

### 1.2.3 组合测试技术研究现状

组合测试（Combinatorial Testing） [24] 是一种可以降低测试成本，提高测试效率的方法。这种测试形式背后的深刻思想是，并非每个参数都会导致每个

故障，而且大多数故障都是由相对较少的参数之间的交互引起的。NIST 和其他相关人员 [25] 收集的经验数据表明，软件故障仅由几个相互作用的变量（6 个或更少）触发。这一发现对测试具有重要意义，因为它表明测试参数组合可以提供高效的故障检测。Jacek 等人 [26] 认为成对测试（2-way）有时用于以较低的成本获得相当好的结果，但成对测试可能会遗漏 10% 到 40% 或更多的测试场景，不能满足高精度、低误差的需求。因此对于本文所针对的反洗钱领域应用场景并不足够。超过双向的组合测试一直受到限制，主要是由于缺乏用于更高交互级别的良好算法，如 4-way 到 6-way 测试。多对测试的问题是参数组合复杂，耗时严重，使得组合测试超出了成对测试的在现实场景下的实用性。

本文中应用组合测试的思想，利用逐参数扩展算法对规则文本中抽取的关键参数进行组合，保证了组合生成规则集合能够覆盖所有情况，从而进行测试数据生成任务。此方法在反洗钱应用场景中，能够保证测试数据的高覆盖率。

### 1.2.4 测试数据生成技术研究现状

软件测试需要生成大规模测试数据来保证软件的质量，发现软件缺陷。经过调研，针对经典的大型智能软件，如果预先用结构化测试数据对其进行测试，能有效发现软件错误与漏洞，节省开发费用占比达到 10%。结构化测试数据生成技术一般有如下三种：随机法、符号执行法和 Korel 法。

1983 年，BIRD DL 等人提出采取随机方法自动生成结构化测试数据 [27]，即对输入进行全随机采样，生成大批量的模糊数据，然后观测软件的输出结果。随机法适合黑盒测试，能在短时间内对软件进行基础测试。因此，文本结合随机法的思想，根据参数化组合生成规则集合，在规则范围内对输入进行特定方式的采样，尽量模拟真实的股票交易数据。

Clarke 等人提出了符号执行法 [28]，与随机法一样，这也是一种静态的测试数据生成方法。符号执行法使用符号值取代实际的数据输入，并在测试过程中生成基于符号的代数表达式作为最终的输出结果。但是在复杂的智能软件中，用符号表达式对测试数据进行表征复杂度高，难以求解。因此，本文参考符号执行的思想，在测试数据生成的前置阶段对规则进行符号表示（参数化组合规则），用来作为数据生成的边界约束条件。

与前面两种静态方法相反，Korel 提出动态数据生成方法 [29]，Korel 法生成的测试数据是根据软件运行结果动态更新的，因此可以减少变量遍历的盲目性。本文参考 Korel 方法的思想，对于生成的测试数据进行结果分析，用评估

的数据质量在后续工作中对次轮任务需要生成的测试数据进行优化。

## 1.3 本文的主要研究工作

随着金融科技从信息基础设施和应用系统建设逐步走向数据资产的集约利用和运营管理，金融数据已经成为政府、银行和公司打造核心竞争力，提高管理效能的重要手段。作为智能软件运转的核心资源，Fintech 数据中最基础也是最庞大的部分是结构化数据。

针对结构化数据的测试数据生成，是在金融反洗钱领域一个需要重点关注的问题，能够实现落地的交易合规监控软件应当具备良好的的识别准确率和对输入数据的较强的鲁棒性。然而当前反洗钱领域中的交易合规监控系统的性能还不够完善。具体表现是误检、漏检概率高，缺乏对全量输入数据的多维度分析等。

本文研究总体流程如下：首先本文调研与分析了现在金融领域反洗钱监测技术，指出了反洗钱安全合规系统测试数据缺乏的问题，提出了需要对测试数据进行大规模生成。其次，结合实际股票交易数据的特性，根据证券公司的实际需求，提出基于反洗钱规则的结构化测试数据生成技术，具体分为参数化组合规则与结构化测试数据生成两种技术，填补了反洗钱测试数据自动生成技术的空白。最后，通过案例分析对该系统的可用性做出验证。

本文的主要工作分为两大部分，一部分为基于反洗钱规则的结构化测试数据生成算法的设计与实现。本文参考现有的结构化测试数据生成技术与方法，并根据金融反洗钱领域的特点，采取参数化组合规则的方式来进行测试数据生成，可以灵活、可控地识别用户输入的洗钱规则文本描述，解决证券公司之前测试数据生成“难以建模、难以泛化、覆盖率不足”的问题。

另一部分是基于反洗钱规则的结构化测试数据生成系统的设计与实现。本文设计了一套便于用户交互的结构化测试数据生成 Web 系统，主要由前端展示层、数据生成层、任务处理层和数据存储层组成。前端展示层主要负责为用户提供系统使用界面。通过 HTTPS 协议将用户请求及文件从客户端发送至服务器端，并在服务器处理请求后将结果发回至客户端展示。数据生成层为系统的核心业务层，负责提供不同形式的生成数据文件，从任务处理层获取用户上传的经过处理的格式化文本数据，根据用户自定义参数调整生成方式，最终为用户提供生成的结构化数据。数据生成层分为参数化组合规则模块和数据生成模

块两大基本模块。任务处理层是连接数据生成层和数据存储层的桥梁，负责处理数据生成层的生成计算任务。数据存储层主要由 Redis 数据库和 MySQL 数据库组成。其中，Redis 数据库负责为 Celery 任务调度提供消息中间件（Broker）和结果存储（Backend）。作为消息中间件，Redis 提供任务调度队列，便于任务处理层异步获取任务的信息。

## 1.4 本文的组织结构

本文将提供设计完整的基于反洗钱规则的结构化测试数据生成技术，并将该技术实现以 Web 应用系统为载体进行统一的设计与实现，以此用来直观显示与分析数据，便于用户交互与操作。本文的组织结构如下：

第一章 **绪论**。描述基于反洗钱规则的结构化测试数据生成相关技术的背景和意义、国内外在这方面的研究现状和文本的研究内容。

第二章 **技术综述**。描述基于反洗钱规则的结构化测试数据生成系统所涉及的相关技术，包括自然语言处理技术、参数化组合规则建模技术、结构化测试数据生成技术、Django REST Framework 框架、Celery 任务调度技术、缓存消息队列 Redis 以及进程管理工具 Supervisor。

第三章 **系统需求分析与架构设计**。首先对基于反洗钱规则的结构化测试数据生成系统进行详细的需求分析，描述其应用场景，然后将系统划分四个模块，分为四个模块：参数化组合规则模块、数据生成模块、结果分析模块和数据管理模块，并且对各个模块分别进行了详细的设计与分析。

第四章 **系统实现**。首先描述了基于反洗钱规则的结构化测试数据生成系统四个模块的具体实现，然后对系统功能进行案例分析，最后进行系统测试。

第五章 **总结与展望**。总结基于反洗钱规则的结构化测试数据生成系统的总体工作，指出存在的问题，对未来研究提出展望。

# 第二章　技术综述

## 2.1　自然语言处理技术

本文技术实现是根据用户输入规则，对金融反洗钱领域的结构化测试数据进行生成，因此在解析规则文本的语义时，应用到了对中文文本的处理操作。本文介绍了所用到的自然语言处理技术，包括中文分词技术和命名实体识别技术。

### 2.1.1　中文分词技术

中文文本与英文的差异性在于中文中单词之间并没有明确的分割标记（比如英文中的空格作为分隔符），而是通过连续字的形式拼接成完整的语句。因此针对中文文本的自然语言处理任务必须对中文文本经过特殊的预处理，将连续字按照合理的语义划分为词，这个流程就是中文分词。中文分词技术主要可以划分为三个类型：机械分词技术、统计分词技术和深度学习技术 [30]。

机械分词技术是先建立中文字典，再通过串查找-匹配的模式进行分词操作，常见的字符串匹配算法有：正向最长词优先匹配法 [31]、逆向最大匹配法 [32]、双向最大匹配法 [33] 和全切分法 [34] 等。基于匹配的算法优势在于匹配切分速度快、领域针对性强等。缺点是分词存在很多歧义性问题 [35]，同时由于分词正确率和词典质量正相关，所以对词典的维护成本极高。

如今应用最多的中文分词技术是统计分词技术 [36]。统计分词技术中的常见算法包括最大熵模型、隐马尔可夫链模型和条件随机场等，条件随机场可以得到很好的分词准确率，但是相比于机械分词技术效率十分低下。Collobert 等人于 2011 年第一次用深度学习算法进行自然语言分词 [37]，中文分词也随后采用 CNN、LSTM 等神经网络进行分词计算。深度学习分词算法的缺点在于模型复杂，需要更多计算单元。

本文的技术实现采用了“jieba”分词，“jieba”分词是目前中文分词领域受众最广的工具，采用 Python 语言编写。对于每一条规则描述文本，其分词流程如下所示：

1) 在现有词条词典的基础上，采用字典树这种前缀树结构，实现了高效的词图扫描，由在句中产生的汉字的全部可能成词情况组成有向无环图（Directed Acyclic Graph, DAG）。

2) 在此基础上，通过动态规划方法寻找最大概率路径，得到基于词频的最大切分组合。

3) 对于未出现在词条字典中的新词，采用了基于汉字成词能力的 HMM 模型，并且用维特比算法进行求解，获得分词列表。

4) 对于分词列表，去除停用词字典中存在的停用词。停用词主要包含特殊字符，语气词，连词以及一些频繁出现但不影响内容的词汇。输出词序列。

5) 语料集经过分词处理，生成每条文本的单词集。

### 2.1.2 命名实体识别技术

在金融领域，公司、股票等机构和组织是金融活动的主体，它们包含着一系列与交易相关的行为和运作方法。目前学术界主要的命名实体识别方法是提出了基于规则的命名实体识别方法，基于统计的命名实体识别方法，以及两种识别策略的组合方法。

在命名实体识别的研究刚刚起步的时候，基于规则的识别方法较为普遍 [38]，该策略适用于金融、法律等领域的实体识别研究。在研究分析金融新闻文本的基础上，王宁等学者 [39] 专门研究了针对公司名称的命名实体识别，建立了用于识别公司命名实体的专家库，并充分考虑了公司名称在识别过程中的语境特征，在文章中，还提出了基于二次扫描的识别方法，并且在数据验证中获得了高于六成的准确率。Tang 等人 [40] 通过对中文金融领域词汇的上下文进行建模，通过词长、词序和在全程名称中的简写词形式提出了命名实体之间的相似度计算算法。

基于规则的命名实体识别方法的优点在于运算速度快，识别依赖于规则集合，十分精准。但是缺点也很明显，规则集的构造需要耗费大量的资源和时间，而且针对不同领域，其覆盖率并不理想。接下来，产生了基于统计的命名实体识别技术 [41]，包括隐性马尔科夫模型、支持向量机模型和条件随机场等算法。在中文领域，Yang 等人 [42] 用公开的报纸文本信息作为训练语料库，采用隐性马尔科夫模型对报纸文本中的金融组织名称进行排名，从而实现了自动识别组织机构命名实体的功能，并且在测试集上达到了九成的准确率。

综合而言，基于规则的命名实体识别和基于统计的命名实体识别相结合，

这样既能保证识别的效率和专业性，同时也可以利用机器学习得到规则覆盖不到的命名实体统计结果，提升了识别算法的准确性。

## 2.2 参数化组合规则建模技术

结构化数据在信息度量上具有离散化、不直观、场景多样化的多态特性。分离参数法 [43] 通过分析特定场景结构化数据的复杂多态特性，在已有的客观计算和主观认知的基础上，设置结构化数据所需要的参数，通过分离参数，从函数功能的角度讨论了主要变量的改变状态，从而确定了参数变化的主要范围。这样就可以避免分类讨论的繁琐步骤，使得问题得以顺利解决。在求解不等式恒成立、不等式有解、函数有零点、函数单调性等问题中，常用分离参数法求解参数的取值范围问题。此方法的重点是在分离出参数后，把原问题转换成求函数的最值问题或值域问题，从而利用参数的组合制定产生结构化数据的规则信息。

在分离出场景的参数集合之后，对于参数之间的约束关系进行分析，并根据数据库之中的表结构构造规则。将抽象出来的变量的取值进行组合并生成可能的测试规则，在一个受到多个参数约束和影响的系统中，其中每个因素的取值是离散且有限的。对于规则之间的组合进行穷举，分析并给出所有可能的组合规则。多规则（N-way，N>2）组合测试可以覆盖任意 N 个规则的所有取值组合，在理论上可以发现由 N 个规则共同作用引发的缺陷。

组合测试产生的测试用例集可以映射到矩阵 M 上，其中 M 的行表示测试用例，M 的列表示输入参数。矩阵 M 中至少出现一次任意元组，其中 f（f>=2）表示组合测试的覆盖强度。当 f=2 时，学界称其为成对组合测试。Mandl 等人首先在软件测试中引入了组合测试 [44]，即使用正交型拉丁方来测试 Ada 编译器，并且取得了显著的效果。随后，研究人员分析了组合测试的缺陷检测能力，例如 Kuhn 等人分析了组合覆盖强度和缺陷检测率之间的关系 [45]，并根据三个缺陷分析报告，发现成对组合测试能够发现 69% 的缺陷，当 f=3 时，three-way 组合测试能够发现 87% 的缺陷，要发现绝大部分缺陷，覆盖强度需达到 six-way。在当前的市场软件应用中，组合测试已经被广泛适用于 GUI 测试、兼容性测试、高度可配置的系统（Highly-configurablesystem）和 Web 应用测试之中。

## 2.3 结构化测试数据生成技术

结构化测试数据是测试用例最基础的数据类型之一，测试数据根据软件的运行结果自动生成，并通过多次迭代，按照指定的覆盖标准生成最终的数据集。结构化测试原则要求一定的程序组成部分作为覆盖参数，例如语句覆盖、分支覆盖、数据流覆盖、条件分支覆盖等。其中，数据流测试是一种结构性测试方法，其中，软件运行过程的数据流关系用于引导测试人员选取相关测试数据。其基本思想是：一个变量的定义，通过辗转的使用和定义，可以影响到另一个变量的值，或者影响到路径的选择等。因此，可以选择一定的测试数据，使程序按照一定变量的定义使用路径执行，并检查执行结果是否与预期的相符，从而发现程序的错误。动态测试数据生成应用在黑盒场景下，在初始测试之后，程序员面对的是额外的测试数据评价未覆盖的程序元素。目前有许多算法应用在测试数据自动生成中，如遗传算法、模拟退火算法、禁忌法。模拟退火算法及遗传算法结合关系类，样本空间，但是，关系和类没有被具体的估计，会做出错误的决定，导致搜索失败。

结构化测试的标准是必须对语句覆盖、分支覆盖、数据流覆盖、条件分支覆盖等程序元素进行评估。数据流测试方法是一种结构化的测试方法，它使用程序中的数据流关系来指导测试人员选择测试用例。它的基本思想是：一个变量的取值，经过反复使用和迭代，可能会影响到另一个变量的值，或影响到路径选择方案。所以，可以选择一定的测试数据，使程序按照定义的变量使用路径执行，并检查运行结果是否符合预期，从而发现程序的错误。在某种程度上，动态测试生成的程序被当作一个黑盒，大多数编程构造对算法是透明的。现有的测试数据自动生成算法 [46] 有禁忌法、模拟退火算法、遗传算法等。

本文针对关系型数据库中的结构化数据进行数据生成。软件产生数据存储早数据库中，作为实现程序功能的数据源。因此，往往需要对数据库的数据进行操作，例如插入、查找、删除、更新等操作。如果是大型项目，数据库还涉及到分库分表。如果进行性能测试，需要在数据库中导入或者导出大量数据。

Carsen 等人提出了两种生成数据库测试数据的方法 [47]：第一种，根据数据库表和应用系统的 SQL 语句生成适当的测试数据库，确保正确读取测试用例中所需的测试数据。第二种，从测试用例的预期输出结果开始，反向生成测试数据库，从而在很大程度上简化了测试数据集，提高了测试数据的有效性。

## 2.4 Django REST Framework 框架

Roy Fielding 在他 2000 年的博士论文中提出了 REST 即性状状态传递（Representational State Transfer, REST） [48]。这是一种软件的架构风格，REST 针对网络应用的设计与开发的痛点作出改进，可以大大减少开发的复杂性，并且提高系统的可扩展性。

目前三种使用频率最高的的 Web 服务实现方案中，由于 REST 模式下的 Web 服务相对于复杂的 SOAP（Formerly an Acronym for Simple Object Access Protocol）和 XML-RPC（XML Remote Procedure Call）来说更加简洁，越来越多的 Web 服务开始采用 REST 风格设计和实现。在 REST 模式下，所有的数据，无论是从网络协议获取的还是操作（增删改查）的数据，都是资源，把全部数据都看作资源是 REST 区别于其他架构风格的最重要特征。Django REST framework 是一个基于 REST 实现的 Web 框架，它通过 Python 编写的 Django 后台框架实现，用于构建 Web APIs。Django REST framework 通过 5 个 HTTP 动词，对服务器端资源进行操作，其定义的功能如表 2-1所示。

表 2-1: Django REST framework 5 种操作

| HTTP 动词 | 相关描述 |
|---|---|
| GET（SELECT） | 从服务器数据库获取资源（一项或者多项） |
| POST（CREATE） | 在服务器数据库新建一个资源 |
| PUT（UPDATE） | 在服务器数据库更新资源（客户端提供完整资源数据 |
| PATCH（UPDATE） | 在服务器数据库更新资源（客户端提供需要修改的资源数据） |
| DELETE（DELETE） | 从服务器数据库删除资源 |

在 DjangoREST Framework 中，当用户通过浏览器访问应用时，返回一个 Web 请求的工作全流程如图 2-1所示。

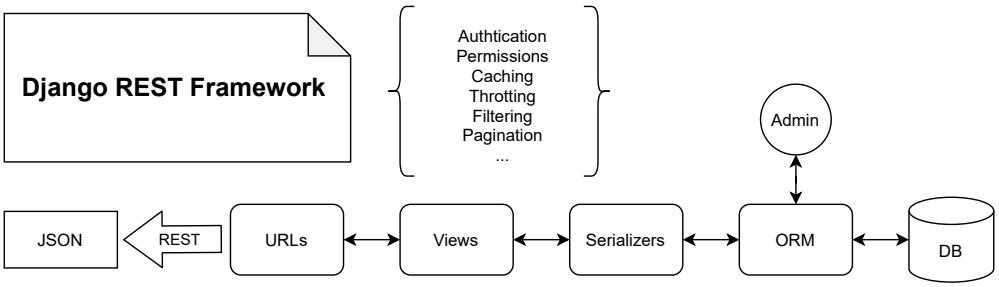


图 2-1: Django REST Framework 整体视图

首先，用户通过浏览器访问服务页面 URL（Uniform Resource Locator），

然后 Web 服务器向 WSGI 发送请求。WSGI（Web Service Gateway Interface），即 Web 服务器的网关接口。它定义了 Web 服务器与 Python Web 应用间的接口标准，以确保 Web 服务器能够与 Python Web 应用相互通信。接下来，服务器把用户请求的 HTTP/HTTPS 消息报文转换成 WSGI 规定的格式，然后传递给 Python 应用。遵循 WSGI 标准的 HTTP/HTTPS 处理函数接受两个参数，其中一个参数的格式是字典形式，包含请求的信息，例如请求的方式、请求的路径等等；另一个参数是在一个可调用对象中调用的函数，用来发起一个包含状态码和 headers 的响应请求。

URLConf 用来查找用户访问的 url 的视图（View），并将其定位到 Django 的 view.py 文件中所对应的业务逻辑函数。对于 Django REST Framework 框架，url 映射器通常存储在 urls.py 文件中，url 映射器定义了路由和相应的视图之间的映射列表，如果接收到的 HTTP 请求有一个与模板匹配的 url，urls.py 文件按照定义次序逐个匹配 urlconfig 的正则部分，一旦匹配成功，就将调用它关联的视图函数，并向该函数传递请求（Request）。

视图是 Web 应用程序的核心，它的功能是接受 HTTP/HTTPS 请求并且处理请求，然后返回相对应的响应值。视图通过数据层（Model）访问需要满足需求的数据，经过业务逻辑函数处理之后，会得到 HTTPResponse 对象，该对象采用 Python 字典对象的格式，并传递给前端。Models.py 文件主要处理 Web 应用程序的管理和数据库中数据的查询，它定义了一个结构来存储数据，包括字段类型，默认值等等。模型的定义是独立于基础数据库的，Django REST Framework 提供了查询 API 来用于进行数据库的搜索操作，因此不需要在视图函数中直接与数据库进行连接与交互。

## 2.5 Celery 任务调度技术

Celery 是一个分布式的任务队列，它主要关注方向是实时计算处理和任务调度，同时提供运行和维护分布式系统所需的工具。任务，又称为消息，执行一项任务所需的有效载荷都包含在消息中。因此，消息机制可使任务的执行与主程序完全分离，甚至可将其分配到其他主机上运行。在执行异步任务（Async Task）和定时任务（Crontab）时，开发者经常通过它来进行代码实现。

从图 2-2中可以看到 Celery 主要包含的 4 个模块和它们之间的相互关系，

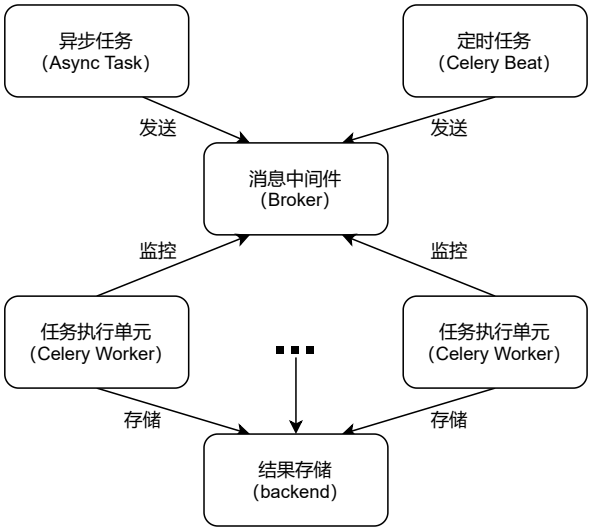


图 2-2: celery 任务调度基础架构

如下所示：

1. 任务模块（Task）：任务模块包含异步任务和定时任务。在这种状态下，即相关应用的业务逻辑中，异步任务通常会并发地向任务队列发送任务消息，而定时任务则由 CeleryBeat 进程读取配置文件的相关配置内容，定期地向任务队列发送配置中的定时任务。

2. 消息代理（Broker）：消息代理收到任务生产者发送的任务消息之后，将其存储到队列中，然后按顺序分发给任务使用者。Celery 本身并不提供队列服务，RabbitMQ 和 Redis 这两种消息队列方案是官方推荐的。

3. 任务执行单元（Worker）：任务执行单元作为执行任务的处理者，它可以实时监控消息队列，获取队列中调度的任务并执行任务。

4. 任务结果存储（Backend）：在任务处理完成后，任务结果存储保存查询的状态信息和执行结果。与消息中间件相同，其存储媒介可以采用 Redis、MongoDB 和 RabbitMQ 等方式。本文采用 Redis 作为任务结果存储的方式。

## 2.6 缓存消息队列 Redis

Redis 是一种使用 ANSCI 语言编写的基于内存的开源数据库，开发者通常使用 Redis 作为缓存数据库或者是消息队列。伴随着 Redis 性能的不断完善，它已经被越来越多的大型企业和公司所采用，包括亚马逊、谷歌、RedisLabs、阿里云和腾讯云在内的多个云服务提供商都提供了基于 Redis 或兼容 Redis 的

服务。Redis 支持五种不同的数据结构，它们分别是字符串（String）、列表（List）、字典（Hash）、集合（Set）和有序集合（Zset），并且可以为数据设置到期时间。此外，Redis 还具有拷贝、持久存储和分片等功能，编程人员可以利用这些功能不断扩展自己的 Redis 存储容量，使其能够达到几 TB 的数据规模，满足大型系统的日常使用。

作为消息队列，Redis 有以下 5 点优势：

1、消息队列的实现基于内存，大多数情况下请求都是纯内存操作，因此响应速度快。Redis 在内存中存储数据方式与 HashMap 相同，所以对数据的基本操作的时间复杂性都是 O(1)。

2、对 5 种数据结构字符串、列表、字典、集合和有序集合的操作十分简单，便于读写。

3、使用单线程模式，可以避免不必要的上下文切转和条件竞争情况的出现，而且不会因为多进程或多线程模式的转换浪费 CPU 资源，也不会考虑各种锁情况，因此能够很大程度上提升性能。

4、采用多路 I/O 复用模式，无阻塞，因此减少了系统调用的开销。

5、作为一个 Key-Value 类型的内存数据库，Redis 实现了自有的 VM 机制，因此可以节省移动和请求系统函数调用的时间。

缓存中的数据具有生命周期，为了确保合理地使用内存、保证缓存数据一致性，Redis 会定期对数据进行更新和删除操作。因此，缓存数据需要采取合理和合适的数据更新策略来进行数据更新。

Redis 的缓存更新策略如表 2-2所示：

表 2-2: Redis 缓存更新策略

| 更新策略 | 具体实现 |
| --- | --- |
| **LRU/LFU/FIFO 算法剔除** | 从服务器取出资源（一项或者多项） |
| **超时剔除** | 在服务器新建一个资源 |
| **主动更新** | 在服务器更新资源（客户端提供完整资源数据） |

除了缓存更新机制之外，Redis 还具有两种持久化机制：RDB（Redis DataBase）和 AOF（Append Only File）。RDB 机制的特点是，当运行 Redis 时，会将数据库中当前状态下的数据以快照得形式拷贝到磁盘上，如果磁盘上存在历史对应的 RDB 文件，则可以直接覆盖该文件，并通过在 Redis 重新启动时解析原始的 RDB 文件来恢复相关数据。但此机制保存的频次要结合实际项目的需求，当保存点设置太大时，尽管 IO 的次数会减少，但是当服务器出现

故障时，会损失较多的数据。因此可以通过减少保存点来减少系统停机时丢失的数据量，但是如果保存点设置太小，则会发生频繁的 IO 操作，造成 CPU 性能不足，导致数据存取速度下降。AOF 机制的特点是通过协议文本将所有对数据库进行过写入命令以及命令相关参数记录到 AOF 文件中，从而记录数据库的状态。

## 2.7 进程管理工具 Supervisor

Supervisor 是一套使用 Python 开发的通用进程管理程序，它可以将普通的命令行进程转换为后台的守护进程（Daemon），并实时监控进程状态。当进程发生异常崩溃时，它会自动重新启动进程进行保活。Supervisor 有两大优势：

1. 使用简单。Supervisor 提供了一种统一的方式来 start、stop、monitor 进程，进程可以单独控制，也可以成组的控制。可以在本地或者远程命令行或者 web 接口来配置 Supervisor。Supervisor 管理进程，就是通过 fork/exec 的方式把这些被管理的进程，当作 Supervisor 的子进程来启动。这样的话，开发者只要在 Supervisor 的配置文件中，把要管理的进程的可执行文件的路径写进去就 OK 了。第二，被管理进程作为 Supervisor 的子进程，当子进程挂掉的时候，父进程可以准确获取子进程挂掉的信息，也就可以对挂掉的子进程进行自动重启保活。同时，Supervisor 通过 INI 格式配置文件进行配置，很容易掌握，它为每个进程提供了很多配置选项，可以使用户很容易的重启进程或者自动的轮转日志。

Supervisor 为启动（Start）、停止（Stop）、监控（Monitor）进程提供了一种统一的方式，即可以对进程进行单独控制或分组控制。Supervisor 可通过本地或远程命令行或 web 界面进行配置。高级管理进程，即通过 fork/exec 将这些管理进程作为 Supervisor 的子进程来启动。在这种情况下，开发人员只需要在 Supervisor 的配置文件中写出进程的可执行路径就可以了。其次，Supervisor 作为 Supervisor 的子进程来管理，当子进程挂掉时，父进程可以精确地获得子进程挂掉的信息，或者自动重新启动挂掉的子进程。与此同时，Supervisor 通过 INI 格式的配置文件来配置，很容易掌握，它为每个进程提供了大量的配置选项，让用户可以轻松地重新启动进程或重新记录日志。

2. 便于集中管理。Supervisor 管理的进程，进程组信息，全部都写在一个 ini 格式的文件里。而且，用户管理 supervisor 的时候的可以在本地进行管理，

也可以远程管理，Supervisor 提供了一个 web 界面，可以在 web 界面上监控，管理进程。本地，远程和 web 管理的时候，需要调用 Supervisor 的 xml_rpc 接口。Supervisor 可以对进程组统一管理，也就是说咱们可以把需要管理的进程写到一个组里面，然后把这个组作为一个对象进行管理，如启动，停止，重启等等操作。linux 系统则没有这种功能，想要停止一个进程，只能逐个去停止。

Supervisor 的基本组件有四个。第一个是 Supervisord，其作为主进程，负责管理进程的 Server，它会根据配置文件 supervisord.conf 创建指定数量的应用程序的子进程，管理子进程的整个生命周期，对 crash 的进程重启，对进程变化发送事件通知等。同时内置 Web Server 和 XML-RPC Interface，轻松实现进程管理。本文使用的配置文件如图 2-3所示，可以看到，配置文件包含了进程启动、执行、重启和停止的命令参数，用于自动化运行服务器的 Celery Worker，并且保存日志记录。

```
[program: celery.worker]
;指定运行目录
directory = / home / xxx / webapps / drf_app
;运行目录下执行命令
command = celery - A
drf_app
worker - -loglevel
info - -logfile
celery_worker.log

;启动设置
numprocs = 1;
进程数
autostart = true;
当supervisor启动时, 程序将会自动启动
autorestart = true;
自动重启

;停止信号, 默认TERM
;中断: INT(类似于Ctrl + C)(kill - INT
pid)，退出后会将写文件或日志(推荐)
;终止: TERM(kill - TERM
pid)
;挂起: HUP(kill - HUP
pid), 注意与Ctrl + Z / kill - stop
pid不同
;从容停止: QUIT(kill - QUIT
pid)
stopsignal = INT
```

图 2-3: supervisord.conf

第二个是 Supervisorctl，它是客户端的命令行工具，提供一个类似 shell 的操作接口，通过它你可以连接到不同的 Supervisord 进程上来管理它们各自的子程序，命令通过 UNIX socket 或者 TCP 来和服务通讯。用户通过命令行发送消息给 Supervisord，可以查看进程状态，加载配置文件，启停进程，查看进程标准输出和错误输出，远程操作等。服务端也可以要求客户端提供身份验证之后才能进行操作。

另一种是 Supervisorctl，这是客户机的命令行工具，它提供了一个类似 shell 的操作界面，通过这个界面，用户可以连接到不同的 Supervisord 进程，以管理各自的子程序，并通过选择 UNIX socket 或是 TCP 的通信协议与服务通信。用户通过命令行向 Supervisord 发送消息，以查看进程状态、载入配置文件、启动进程，并且可以查看进程的标准输出和错误输出、远程操作等等状态。服务器也可以要求客户机在操作之前提供身份验证。

第三个是 Web Server，Superviosr 同样提供可视化的 Web 服务，可以给用户提供可以交互的 Web 界面，通过简单直接的方式控制进程。

最后一个组件是 XML-RPC Interface，它是 XML-RPC 接口，类比于 HTTP 提供 WEB API，XML-RPC Interface 用接口的方式控制和管理 Supervisor 的相关程序。

## 2.8 本章小结

本章首先介绍了与本系统相关的自然语言处理技术，包括中文分词技术和命名实体识别技术，为系统对反洗钱规则文本处理奠定基础；接着介绍了参数化组合规则建模技术，确定了参数化组合规则模块中反洗钱规则集构造的基本思路；其次，介绍了结构化测试数据生成技术基本方法与手段；最后介绍了 Django REST Framework 框架、Celery 任务调度技术和缓存消息队列 Redis 作为本文 Web 系统设计与实现的底层技术支撑。

# 第三章　系统需求分析与架构设计

本章基于证券公司反洗钱监控场景下对于测试数据的真实需求，研究基于反洗钱规则的结构化测试数据生成系统的软件需求。接着，从全局角度对本系统进行架构设计。基于反洗钱规则的结构化测试数据生成系统划分为四个子模块：参数化组合规则模块、数据生成模块、结果分析模块模块、数据管理模块。本章对每个模块的功能进行阐述，最后设计并且描述系统数据库。

## 3.1　系统需求分析

本部分主要对参数化组合规则模块、数据生成模块、结果分析模块和数据管理模块这四个模块进行需求分析。图 3-1是基于反洗钱规则的结构化测试数据生成系统用例图的具体描述。

参数化组合规则模块负责对用户输入的反洗钱规则文本做分词处理，并从中进行参数抽取，进而根据参数组合生成相应的规则集合，实现用户自定义规则。

数据生成模块是系统的核心功能，负责利用参数化组合规则模块提供的规则集合生成指定数量的结构化数据。包括规则解析、数据初始化和任务监控三个子功能。

结果分析模块承接数据生成模块产出的数据，对数据生成的时间、数据生成的规模、数据生成的质量进行分析与展示，从而给用户提供测试数据的基本表现和质量度量。

数据管理负责对系统用户生成的反洗钱测试数据进行管理，具体而言，提供数据显示、数据下载、数据删除的功能。

基于反洗钱规则的结构化测试数据生成系统将四个模块解耦，支持模块之间的相互调用，便于后期系统更新。具体的模块操作流程如下所示：首先，用户登录系统，进入数据生成任务页面，输入指定的规则文本描述，系统调用参数化组合规则模块对规则文本进行处理，分离出其中的参数，通过组合得到规则集。处理完成后，系统调用数据生成模块，根据规则集进行数据初始化，然

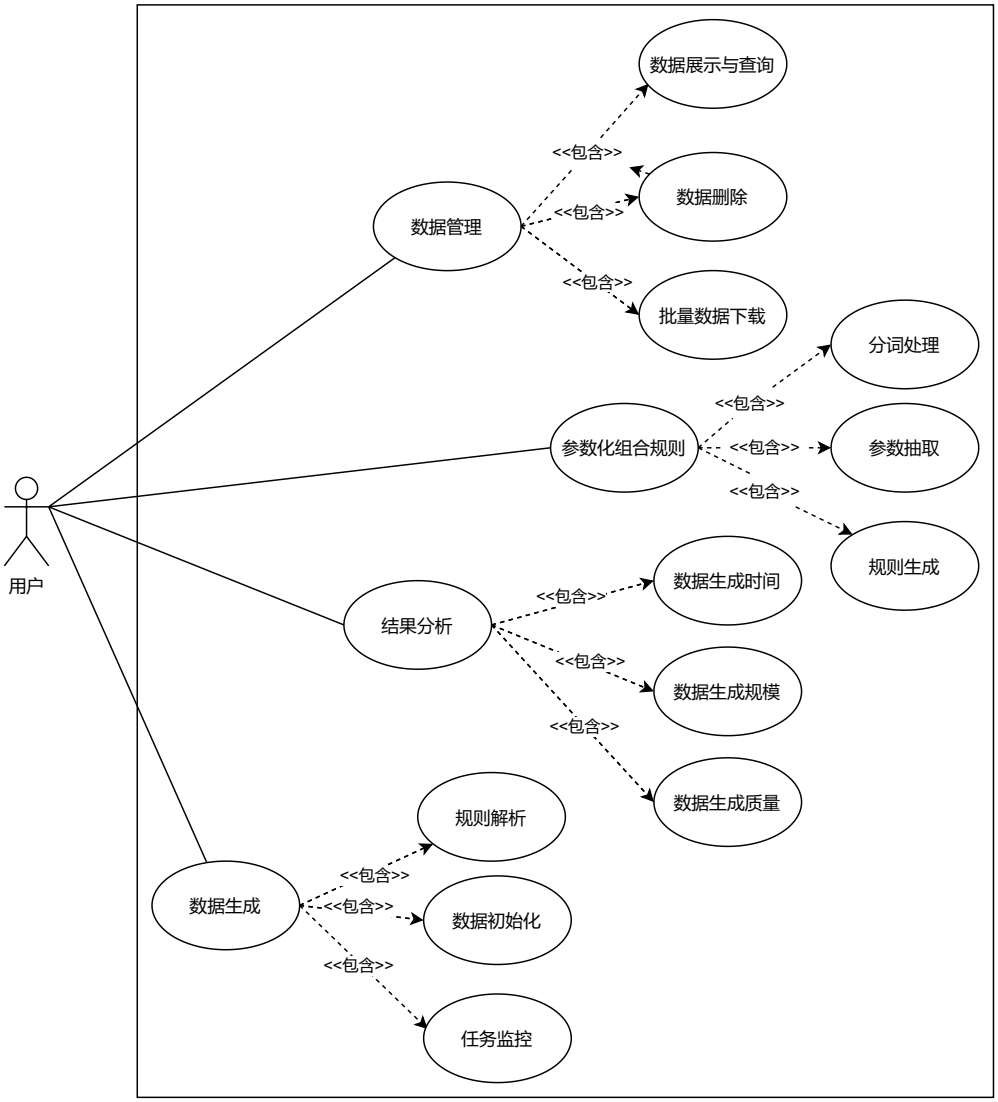


图 3-1: 基于反洗钱规则的结构化测试数据生成系统用例图

后开始数据生成任务进程。最后，数据生成任务结束，调用结果分析模块对整个任务流程和生成的数据进行分析。此外，数据管理模块可以对系统的历史数据进行快捷、直观的管理与操作。

### 3.1.1 参数化组合规则模块需求分析

参数化组合规则模块是结构化测试数据生成的前提，用户需要输入反洗钱规则文本描述，经过参数化组合规则模块处理文本数据之后，才会进行生成数据步骤。反洗钱规则由中国人民银行制定，其中的关键参数会定季度进行修订。参数化组合规则功能负责对用户制定的规则描述做出解析，理解其中的参

数化信息，并且给用户提供数据生成模块可以选择的规则集合。表 3-1提供了参数化组合规则模块的用例描述。

如表所示，参数化组合规则模块与 MySQL 数据库连接，需要将用户输入的反洗钱规则文本解析出来的参数和数据库中的字段进行匹配，以分析参数的可用性和值域。同时，将用户输入的规则文本和分析出的规则集合进行结构化存储，持久化存入 MySQL 数据库中，便于用户以后使用或者参考历史规则数据。参数化组合规则模块的正常流程首先需要调用分词 API（jieba 分词）对文本进行分词操作，然后对分词结果进行分析，提取其中的关键参数。紧接着，用户根据提取的参数自主进行选择，系统得到选择的参数，自动分析生成相关的规则集合并提供给数据生成模块进行处理，正常流程结束。在整个流程过程中，对于用户输入和选择参数部分，系统会进行校验并判断用户操作的合规性。

表 3-1: 参数化组合规则模块用例状态描述表

| ID | UCI |
|---|---|
| 名称 | 参数化组合规则模块 |
| 描述 | 将用户提交的反洗钱规则文本进行处理分析，提取出规则集合 |
| 参与者 | 普通用户 |
| 前置条件 | 用户登录系统，接着上传反洗钱规则文本 |
| 后置条件 | 返回生成的规则集合，用于进行数据生成 |
| 优先级 | 高 |
| 正常流程 | 1. 用户提交反洗钱规则文本，调用分词 API，对文本进行分词操作。<br>2. 对分词结果进行分析，接入系统级词典辅助判断，提取关键参数。<br>3. 用户选择关键参数的组合方式，调整关键参数的值，系统自动分析并生成相关规则集合。<br>4. 返回生成的规则集合至数据生成模块。 |
| 异常流程 | 2a. 对于用户输入的无关规则文本，则返回提示信息，重新提交。<br>3a. 如果用户不选择任何参数，返回提示信息，重新选择。<br>3b. 如果用户选择的参数化组合有冲突，返回提示信息，重新选择。 |

### 3.1.2 数据生成模块需求分析

数据生成模块是系统的重要组件，其作用是根据用户输入规则描述，生成基于反洗钱规则的结构化测试数据。表 3-2提供了数据生成模块的用例描述。

本模块从参数化组合规则模块返回的规则集合中解析规则，接下来在测试数据生成之前，对数据库与规则相关表字段的数据初始化，获取基础仿真数据

信息，比如股票行情信息、虚拟人物身份信息等。由于大规模数据生成任务耗时较长，因此下一步，系统调用 celery 进行异步任务，防止任务阻塞用户的操作流程，通过 redis 缓存消息队列，监控任务进行状态。当任务进程结束后，存储生成的结构化测试数据至 MySQL 数据库，便于数据管理模块对数据的查找等操作。最后，返回任务信息和生成数据至结果分析模块。由于参数化组合规则模块已经过滤了用户的不合规输入，所以数据生成模块的传入数据的合规性得到了保证，异常流程为空。

表 3-2: 数据生成模块用例状态描述表

| ID | UCI |
|---|---|
| 名称 | 数据生成模块 |
| 描述 | 按照参数化组合规则模块返回的规则集合进行数据生成 |
| 参与者 | 普通用户 |
| 前置条件 | 用户触发参数化组合规则模块 |
| 后置条件 | 生成测试数据返回结果分析模块 |
| 优先级 | 高 |
| 正常流程 | 1. 从参数化组合规则模块返回的规则集合中解析规则。<br>2. 对数据库与规则相关表字段的数据初始化，获取基础仿真数据信息。<br>3. 系统调用 celery 进行异步任务，并且通过 redis 缓存消息队列，监控任务进行状态。<br>4. 生成完成后存储测试数据至 MySQL 数据库。<br>5. 返回任务信息和生成数据至结果分析模块。 |
| 异常流程 | 无 |

### 3.1.3 结果分析模块需求分析

表 3-3提供了结果分析模块的用例描述。结果分析模块针对数据生成模块反馈的结果进一步分析，得出数据生成的时间、数据生成的规模和数据生成的质量指标。

数据生成时间统计从数据生成模块启动任务开始，到任务结束，数据全部持久化存入 MySQL 数据库结束的总体时间，与数据生成规模整合，反映了系统处理一定规模数据的效率。数据质量利用参数化组合规则模块产生的规则集合与测试数据生成的标签作对比，得出测试数据是否精确符合反洗钱规则的一致性检测结论。最后，将整体的结果分析报告通过前端展现给用户，正常流程结束。

表 3-3: 结果分析模块用例状态描述表

| ID | UCI |
|---|---|
| 名称 | 结果分析模块 |
| 描述 | 针对数据生成模块反馈的结果进行分析 |
| 参与者 | 普通用户 |
| 前置条件 | 数据生成任务完成 |
| 后置条件 | 无 |
| 优先级 | 中 |
| 正常流程 | 1. 针对数据生成任务的运行时间进行分析，与预期时间对比，得出性能评分。<br>2. 针对数据生成任务的规模进行分析，得出规模级别。<br>3. 针对数据生成任务中的数据标签分析，根据规则集合进行一致性检测。<br>4. 通过检测结果评估测试数据的数据质量。 |
| 异常流程 | 无 |

### 3.1.4 数据管理模块需求分析

数据管理带给用户的核心功能有：1. 直观的展示了洗钱和非洗钱标签用户的数量，并且提供了根据标签和根据客户 ID 的查询方式，可以进行快速的数据检索与查看。2. 结构化测试数据以列表的形式展现，每行数据包含了触发规则和相关任务的生成日期与 ID，同时提供了洗钱标签筛选和用户 ID 搜索的功能，便于快速搜索相关数据，并且可以选中相应的行数据查看具体的生成用户信息。3. 此外，数据管理模块提供删除数据和批量下载数据的功能，可以选择指定的数据，采用 SQL 或者 EXCEL 的文件格式进行下载。4. 支持用户在总览列表页进行搜索。表 3-4提供了用户数据管理模块的用例描述。用户可以在数据管理模块的页面查看历史生成过的所有数据信息。包括结构化测试数据的各个字段、数据量、产生时间、数据标签（是否洗钱）等。并且，数据管理模块支持用户选中指定的数据，导出结构化数据至 EXCEL 下载至本地。此外，用户可以选择需要删除的数据，系统会将 MySQL 持久化保存的数据行删除。异常流程考虑用户并没有历史生成数据的情况下，应当正常显示空白页面。

### 3.1.5 非功能性需求分析

系统的非功能性需求是整体需求中除去功能型需求以外的关键部分。它可以用于评估系统的安全性、可操作性和健壮性等，对于本文实现的应用软件而言至关重要。通过以上需求分析环节，得到系统需满足的功能性需求。接下

表 3-4: 数据管理模块用例状态描述表

| ID | UCI |
| --- | --- |
| 名称 | 数据管理模块 |
| 描述 | 查看生成的反洗钱测试数据，进行相关操作 |
| 参与者 | 普通用户 |
| 前置条件 | 用户登录系统 |
| 后置条件 | 无 |
| 优先级 | 中 |
| 正常流程 | 1. 用户通过前端界面，跳转数据管理页面。<br>2. 显示测试数据基本信息，包括任务 ID、触发规则等。<br>3. 支持数据下载，系统将导入结构化数据至 EXCEL 文件并下载到本地。<br>4. 用户点击“删除”按钮，删除数据库中所有相关联的记录数据。 |
| 异常流程 | 2a. 如果用户没有生成数据，则展示空页面。 |

来，根据反洗钱测试数据生成场景的要求，从系统可操作性、系统安全性、系统健壮性这三个方面对非功能性需求进行阐述。

（1）系统可操作性。系统应确保易于用户操作，提供简洁明了的前端展示和操作界面。用户可以通过交互行为进行反洗钱测试数据的相关操作（查看、删除、下载）。

（2）系统安全性。系统严格限制用户帐户数并且确保仅一些可信用户具有对服务器计算机的管理访问权限。同时，系统应确保具 Web 服务器访问权限的用户帐户不可访问 shell 功能。最后，系统应排查开放式 Web 应用程序安全项目（OWASP, Open Web Application Security Project），确保不存在相关漏洞。

（3）系统容错性。系统生成反洗钱测试数据以主从复制的方式存储在关系型数据库中，并且在主服务宕机的情况下能够快速切换至备用服务器。

## 3.2 系统架构设计

基于反洗钱规则的结构化测试数生成系统的总体结构如图 4-7所示，本系统采取前后端分离的模式，包括四个层级，分别是前端展示层、数据生成层、任务处理层和数据存储层。本章节对平台进行了总体设计，确定了系统框架。

前端展示层主要作用是为用户提供可视化、便于操作的 UI 界面。通过原生 jQuery 框架和 Bootstrap 框架相结合，搭建了 web 前端界面，通过浏览器和用户进行交互。在和 Django REST Framework 后端进行通信方面，系统的通信

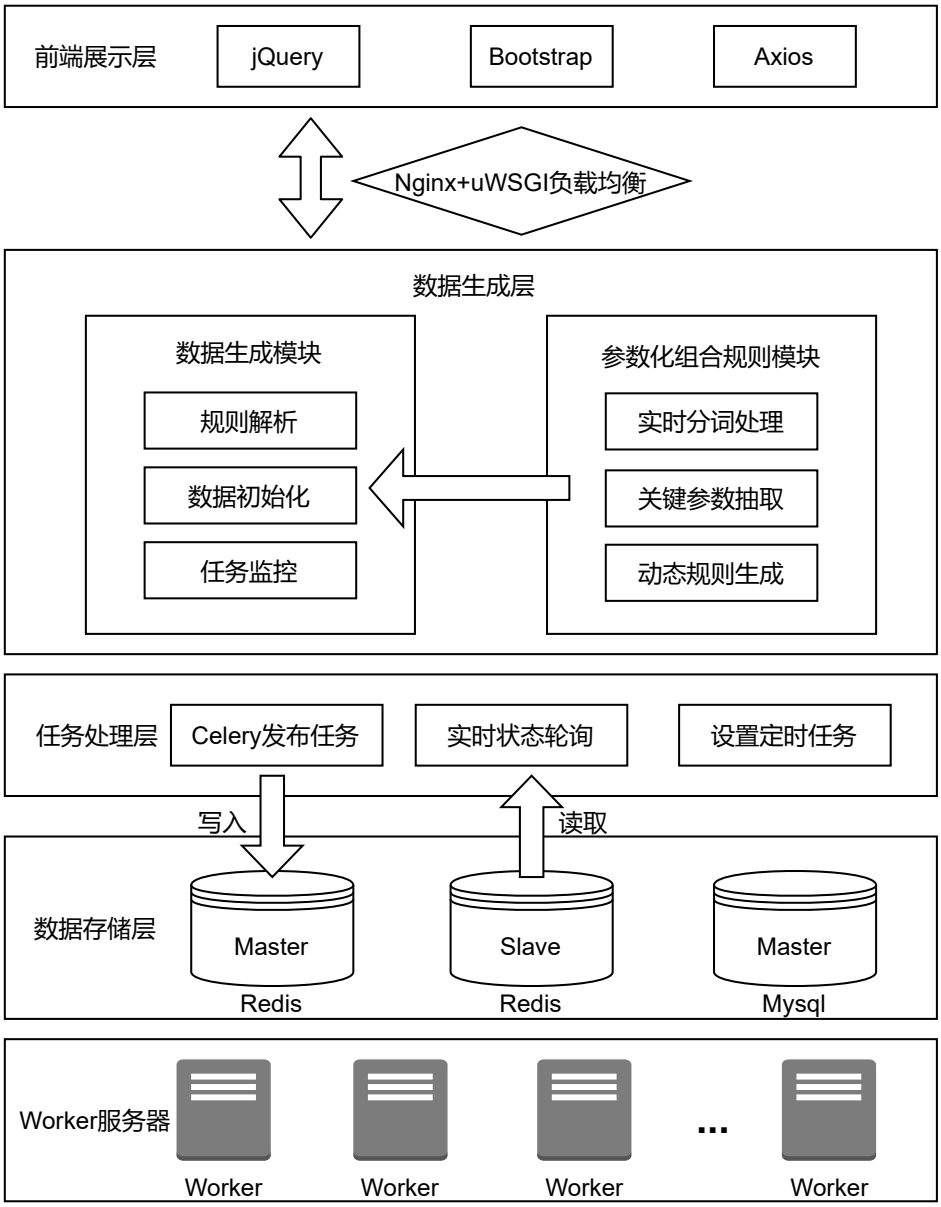


图 3-2: 基于反洗钱规则的结构化测试数据生成系统架构图

交给的 Axios 组件进行处理，Axios 是基于 promise 的 HTTP 库，它可以与后端进行异步高延时通信，这样保证了系统在进行长时间任务处理时不会阻塞前端，同时轮询后端任务处理层，进行任务进度的查询。

数据生成层主要负责本系统的核心业务，即基于反洗钱规则的结构化测试数据生成技术的实现，负责为用户提供指定规则的相关测试数据。数据生成层主要分为两个子模块：参数化组合规则模块和数据生成模块。用户操作前端 UI，输入规则描述文本返回给后端数据生成层，然后经过参数化组合规则模块

处理规则文本，提供给给用户参数选择，从而动态生成规则集合。数据生成模块获取规则集合后，负责提供数据生成所需要的计算服务，并将任务分发给任务处理层进行处理。

任务处理层是连接数据生成层和数据存储层的桥梁，负责处理数据生成层的生成计算任务。任务处理层具体负责三个操作：第一是系统调用 Celery 进行异步任务发布，将任务 ID 和完成状态写入 Redis 任务调度队列；第二是实时状态轮询，通过定时查询 Redis 任务调度队列，得到任务处理的实时状态；第三是设置定时任务，用户可设置任务的触发时间，由 Celery Beat 进程周期性地将任务发往 Redis 任务队列。

数据存储层主要由 Redis 数据库和 MySQL 数据库组成。其中，Redis 数据库负责为 Celery 任务调度提供消息中间件（Broker）和结果存储（Backend）。作为消息中间件，Redis 提供任务调度队列，便于任务处理层异步获取任务的信息。作为结果存储，当任务处理结束，将任务运行结果存入 Redis。同时，Redis 还进行了主从复制，实现了数据的热备份，方便故障时快速恢复系统功能。MySQL 数据库负责存储生成的结构化测试数据，包括用户基础信息（用户名称、用户密码、用户操作权限）和结构化数据相关信息（股票交易信息、交易账户信息等）、任务处理信息（任务 ID、任务描述、任务所属用户等）。

平台的数据生成计算服务分发到 Worker 服务器集群中，每台服务器作为任务执行的单元，即 Celery Worker，负责指定任务的测试数据生成运算，Worker 的计算完成后，会将结构化数据写入 MySQL 服务器中。

接下来采取“4+1”视图模式，详细描述进程视图、逻辑视图、场景视图、物理视图与开发视图。通过将整体系统进行解耦并且划分为方面，从而能够以直观、简易的方式理解系统构成，软件系统结构的整体内容由五种视图反映的不同方面共同决定。本文已经在 3.1 节的系统需求分析中以 UML 类图的形式，通过场景视图描述了基于反洗钱规则的结构化测试数生成系统总体使用场景。

本系统的逻辑视图如图 3-3所示，从业务逻辑入手，阐述系统软件功能拆解后的组件关系，组件约束和边界。可以看到，HTTPResponse 类是系统前后端分离下进行数据交互的基类，RuleTextUploadService 继承了 HTTPResponse 类，负责用户上传规则文本的功能。DataManageService 同样继承 HTTPResponse 类，负责数据管理模块与数据库接口的连接。TestDataGenerateService 类负责系统数据生成的服务，首先会调用 TextPreprocessService 类去对 RuleTex-

tUploadService 类输出的文本进行预处理，ParamExtractService 类对预处理的文本进行参数抽取，RuleAnalyseService 类紧接着对于参数进行组合，产生规则集合。CeleryTaskService 类负责根据规则集合，进行任务的发布（TaskReleaseService）、任务状态轮询（TaskPollService）和设置定时任务（TaskCronService）。

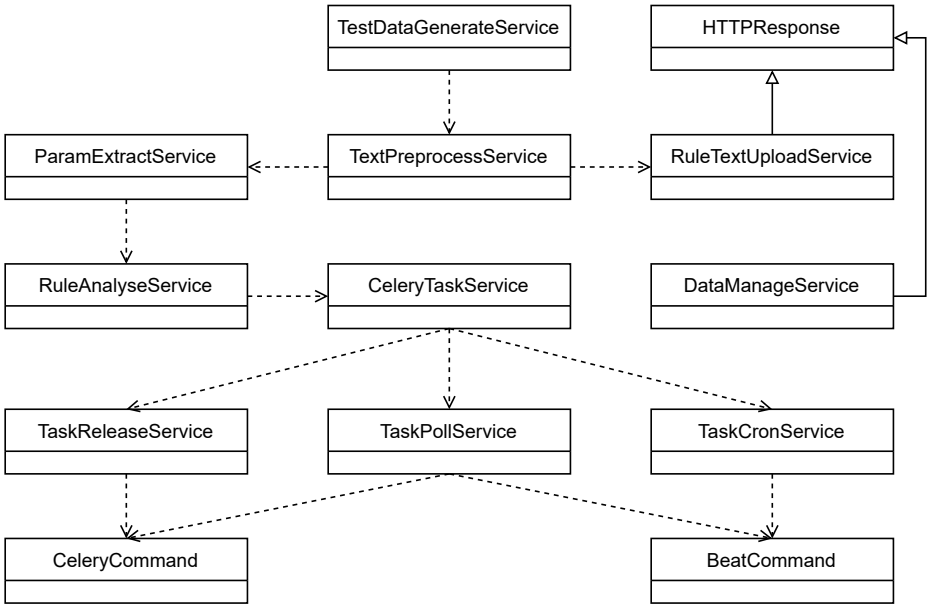


图 3-3: 基于反洗钱规则的结构化测试数据生成系统逻辑视图

图3-4是系统的开发视图。开发视图从开发者角度描述了系统的整体运转流程和结构。本系统主要由两个部分构成：前端和服务端。前端负责通过浏览器和用户交互，提供可视化、便于操作的设置规则服务、测试数据生成服务、生成数据管理服务和任务查看服务。用户通过 Web 界面操作来输入反洗钱规则，传递用户提交的规则文本和参数化组合规则集合至服务端。系统逻辑实现由 Web 服务端完成，服务端由参数化组合规则模块、数据生成模块、结果分析模块和数据管理模块四个模块组成。其中，参数化组合规则模块在系统中负责对前端用户设置的规则文本进行参数化处理并且组合成规则集合。数据生成模块承接参数化组合规则模块，是系统的核心开发模块，负责对规则集合进行解析，并进行数据初始化和数据生成，与 MySQL 数据库进行连接，进行生成的结构化测试数据的存储工作。结果分析模块对数据生成的时间、规模和质量进行全方位的分析与展示，用户可以在前端的任务结果查看模块对分析结果进行调取。数据管理模块也与 MySQL 数据库进行连接，在系统中负责数据交互，对接 MySQL 数据库，用户可以查看、下载、删除生成的数据。

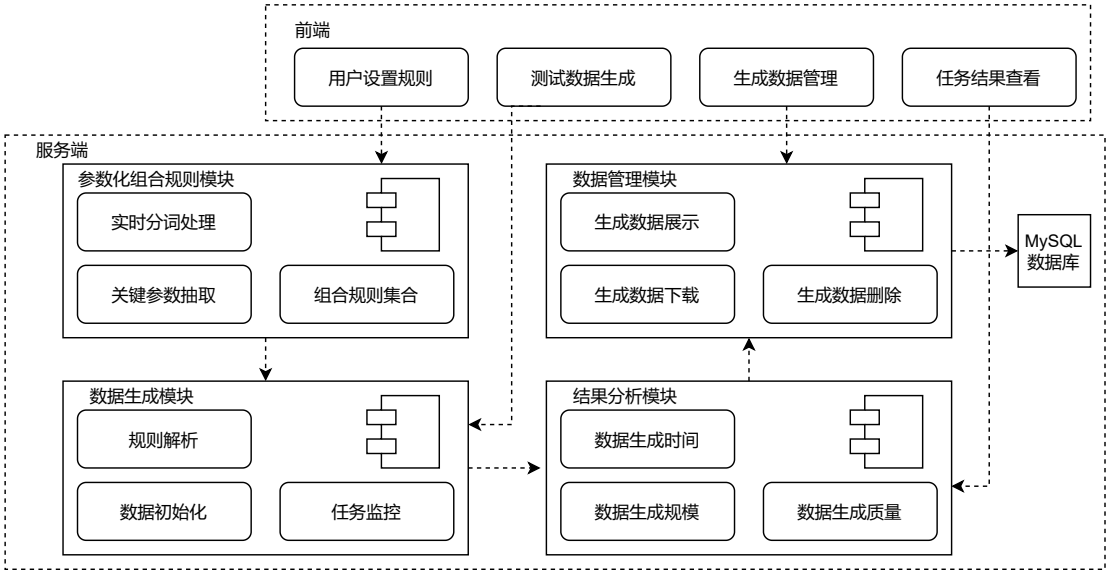


图 3-4: 基于反洗钱规则的结构化测试数据生成系统开发视图

图3-5是系统的进程视图，从系统集成的视角，进程视图描述了系统的运行特性，定义了系统间进程的通信流程和时序。首先，当一个数据生成任务被发起时，系统的主进程会调起一个异步线程，异步线程负责和 Celery Worker 服务器之间的通信。随后，异步线程会发起参数化组合规则的计算进程，此进程中会调用方法处理用户传入的规则文本并解析出参数化组合的规则集合，将用户输入的规则描述记录在 MySQL 数据库中，并将规则集合作为触发器传给测试数据生成进程，然后进行数据生成相关计算，并将结果存入 MySQL 数据库中。可以看到，MySQL 数据库服务器在系统运行期间一直拉起监听程序，能做到及时存取规则数据和测试数据。在此期间，主进程会轮询异步线程，调用 Redis 消息队列查询数据生成任务的完成情况。最后，生成任务完成后，数据生成进程将结果反馈给主进程，流程结束。

图 3-6是系统的物理视图，物理视图从运维的角度出发，描述了系统在物理意义上的部署结构和运行模式，为开发和运维人员提供了系统运行的实际流程。如图所示，用户用个人浏览器访问系统网站，通过 HTTP 请求对后台发起申请。首先会通过防火墙对恶意请求的拦截和过滤，然后请求流转进入 Nginx 服务器，Nginx 监听进程会实时监听请求并且进行代理转发来解决跨域问题，并将请求通过 Socket 快速转发给 uWSGI 服务器。uWSGI 内部提供 Django REST Framework 的 Web 应用服务，负责处理系统的具体业务逻辑，并返回 HTTP 响应。Django 与 MySQL 数据库服务器进行 TCP 通信，承担系统的数据

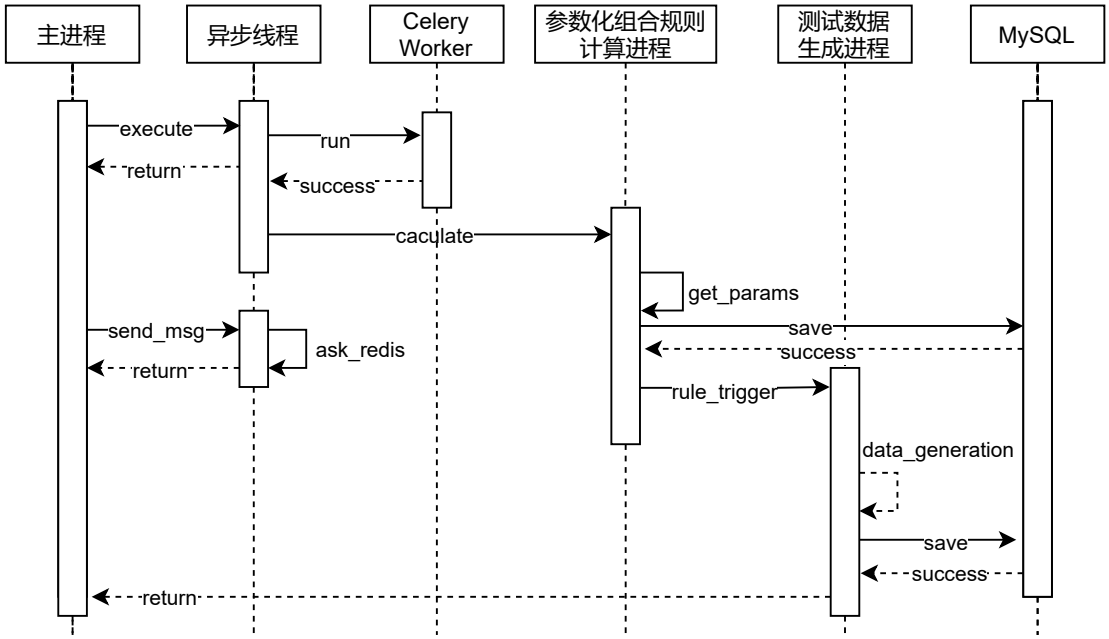


图 3-5: 基于反洗钱规则的结构化测试数据生成系统进程视图

存储、查询、增加、删除等相关数据操作服务。同时任务消息队列数据库服务器用于架设 Redis 数据库，作为任务的消息队列和结果存储服务，负责系统的数据生成计算服务器（Celery Worker）、uWSGI 服务器之间的异步通信（TCP 通信进行连接）。

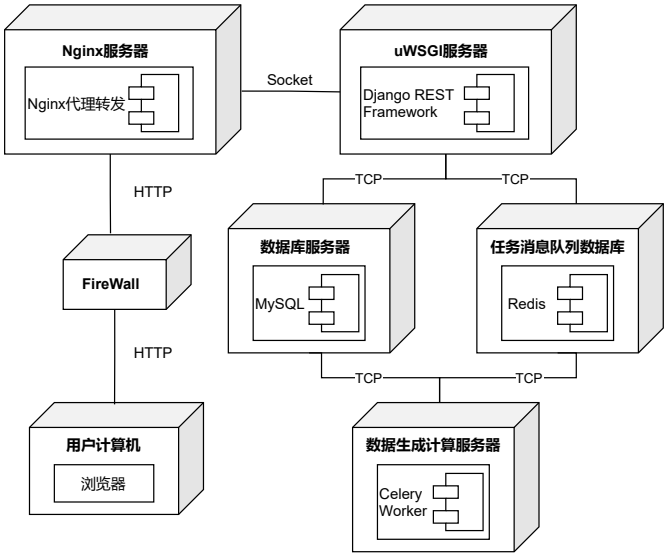


图 3-6: 基于反洗钱规则的结构化测试数据生成系统物理视图

综上所述，系统平台的功能模块图如图 3-7所示，包括参数化组合规则模块、数据生成模块、数据管理模块和结果分析模块。其中，参数化组合规则模

块和数据生成模块是系统的两大核心模块。参数化组合规则模块包括分词处理、参数抽取、规则生成三部分，数据生成模块包括规则解析、数据初始化和任务监控三部分，根据用户输入的规则描述生成相应的结构化测试数据。总体而言，四个模块互相依靠，实现了基于反洗钱规则的结构化测试数据生成技术系统级别的应用。

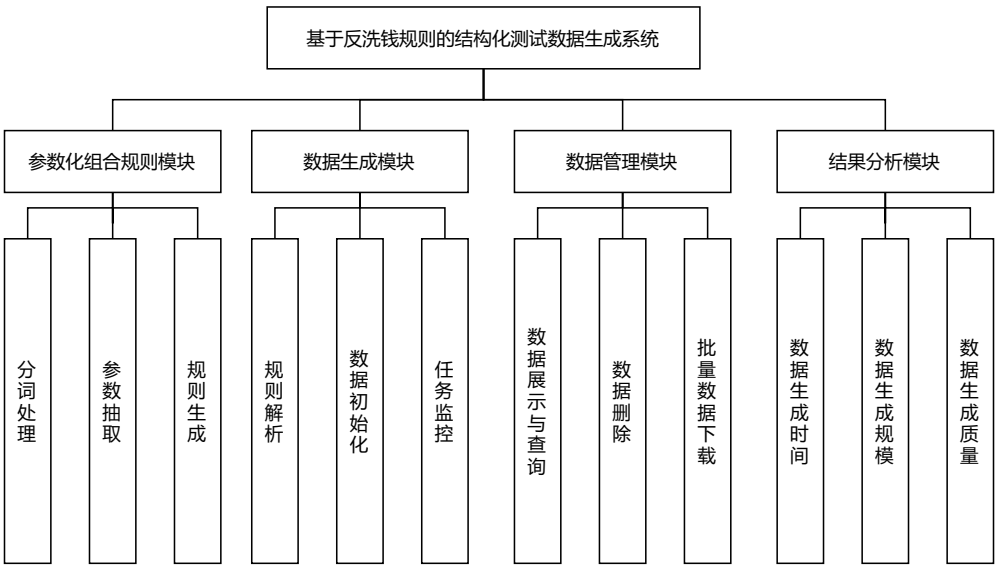


图 3-7: 基于反洗钱规则的结构化测试数据生成系统功能模块图

## 3.3 参数化组合规则模块设计

参数化组合规则模块负责对用户输入的规则文本做分词处理，并从中进行参数抽取，用户可以选择参数，进而组合生成相应的规则集合。参数化组合规则模块类图如图 3-8所示，TextPreprocessService、ParamExtractService 和 RuleGenerateService 共同实现了接口类 ParamCombRules。

首先，HTTPResponse 是系统前后端进行 Web 通信的基类，负责前后端通过 HTTP 协议传输信息。RuleTextUploadService 继承了 HTTPResponse，负责接收用户上传的规则文本功能。TextPreprocessService 对文本数据进行预处理，TextSegmentation 类继承了该类，并且实现了分词方法。首先调用 init_aml_dic() 方法加载反洗钱领域词典，然后对文本进行分词处理，输出结构化分词结果保存在 json 文件中。ParamExtractService 类调用 TextPreprocessService，负责对分词结果的 json 文件进行分析，judge_seg_to_param() 方法

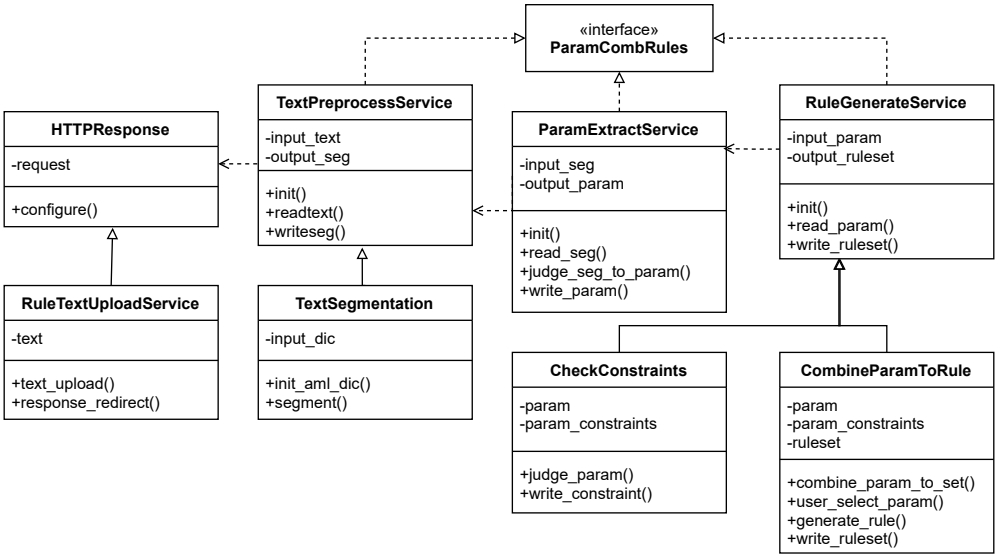


图 3-8: 参数化组合规则模块类图

遍历分词结果，判断并提取其中的关键参数，并输出参数结果。最后，RuleGenerateService 类调用 ParamExtractService，输入参数，CheckConstraints 类和 CombineParamToRule 继承该类。CheckConstraints 类实判断参数相关冲突，获取参数的约束关系，CombineParamToRule 类将参数打包成参数集，用户调用方法 user_select_param() 选择需要进行生成的参数，然后用选中的参数集组合成规则集，并保存结果到 MySQL 数据库中。当用户发起下一次生成任务的时候，可以从 MySQL 数据库中调取历史规则记录，提供参考和选择。

通过类图定义和方法实现可以得到参数化组合规则模块整体流程如图 3-9所示。

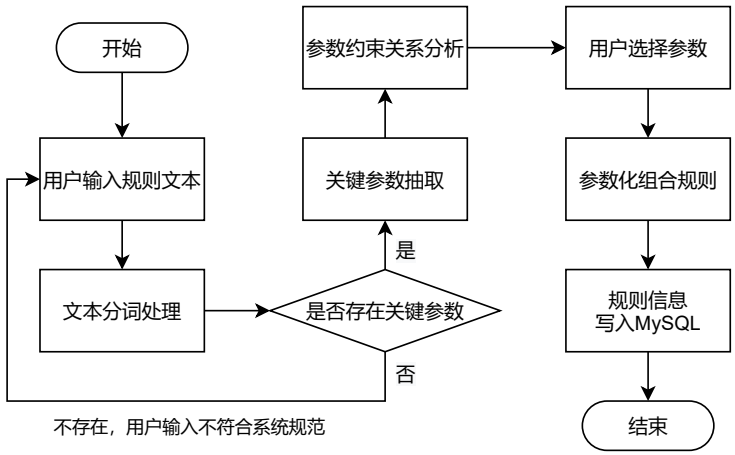


图 3-9: 参数化组合规则模块流程图

从用户输入规则文本开始，进入文本分词处理环节，此时会判断分词中是否存在关键参数，如果不存在，说明用户输入的规则文本不符合系统的规范，整体流程会返回到用户输入环境；如果存在关键参数，程序将进行后续关键参数抽取、规则集合生成步骤，对应 ParamExtractService 类和 RuleGenerateService 类。

接下来描述参数化组合规则模块实现生成规则集合的核心算法。首选提供算法所涉及到的相关符号及其描述，如表 3-5所示。

表 3-5: 相关符号及其描述

| 符号 | 描述 |
| --- | --- |
| $WS$ | 分词集合，字典形式 |
| $w_m$ | 分词集合中的第 m 个分词 |
| $c$ | 机构实体属性参数，例如“XX 证券交易所” |
| $p$ | 交易人员属性参数，例如“李 XX” |
| $t$ | 交易发生时间参数，例如“2021-04-15 14:10:08” |
| $l$ | 交易发生频率参数，例如“当日交易三次” |
| $a$ | 交易涉及交易金额参数，例如“300 万元人民币” |
| $d$ | 交易行为参数，例如“买入 5000 股” |
| $\mathcal{R}$ | 规则描述 |
| $\mathcal{A}$ | 洗钱触发关系运算符参数，例如“/” |

算法伪代码如下图所示，分为关键参数提取和规则集合生成两个大的阶段。算法将将分词集合 $WS$ 作为算法输入，最后得到生成的反洗钱规则集合。

**第一阶段**。关键参数提取。根据 WS 分词集合，实行实体命名识别算法并提取相关参数。本系统首先初始化元组 $K$，用于结构化保存提取的关键参数。接下来调用反洗钱领域字典 $AML_{dict}$。算法开始，遍历 $WS$，对于其中的每个分词 $w_m$，采用 findParamType() 方法对其进行命名实体识别，并且输出格式化表述，确定其参数属性。由于金融反洗钱领域的专业性，对其中的相关参数建立领域规则词典，通过正则表达式的形式进行判别，使得实体识别高效率而且准确。具体如表 3-6所示，findParamType() 通过调取相关规则的正则表达式，对分词进行属性归类和格式化。如果判断 $w_m$ 不符合命名实体识别规则词典的任何类别，findParamType() 返回值为 0。当返回值不为 0 时，调用 addParams() 方法，往元组 $K$ 中添加格式化参数和其属性组成的字典数据。遍历完成，流程结束，得到完备的关键参数元组 $K$。

**第二阶段**。规则集合生成。本部分算法对于参数进行组合，生成对应的规

**Algorithm 1:** 参数化组合规则整体算法

```
# 第一阶段：关键参数提取
initialize key params array K for saving params
fecth AML dictionary AML_dict for detecting params
foreach w_m in WS do
    if AML_dict.contains(w_m) then
        paramType ← findParamType(w_m)
        if paramType! = 0 then
            addParams(K, w_m, paramType)
# 第二阶段：规则集合生成
if K.getNum()< 2 then
    endAlgorithm()
else
    # 初始化规则用例集 U
    P_1 ← popFirst(K)
    P_2 ← popFirst(K)
    U ← {(v_1, v_2)|v_1v_2P_1P_2}
    # 对于在 K 中的每一条参数做以下循环
    foreach P_i in K, i >= 3 do
        # 对于在 U 中的每一条用例做以下循环
        foreach set_j in U do
            set_new ← addValue(set_j, P_i.getValue())
            U.addSet(set_new)
            set_mutation ← addValue(set_j, P_i.mutateValue())
            U.addSet(set_mutation)
        O ← doPairCombination(K)
        U ← U ∪ O
```

则。研究表明，对于组合测试用例构造技术，产生最小成对组合集合是一个 NP 完全问题。因此，本算法采用了逐参数扩展算法 (in-parameter-order, IPO) 对规则进行自动生成，对于一个含有两个以及两个以上关键参数的元组 $K$，IPO 算法首先为 $K$ 的前两个关键参数生成二元规则用例，紧接着将 $K$ 剩余的每个参数依次补充到现有的规则用例集合中。参数扩展首先在横向上进行扩展，对于每个参数，用 getValue() 取得其值，对于规则集 $U$ 中的每个 $set_j$ 规则，添加该参数值生成新的规则用例，更新到规则集 $U$ 中。接下来，mutateValue() 方法识

表 3-6: 反洗钱领域命名实体识别规则词典

| 类别 | 规则描述 |
| --- | --- |
| 机构 | 识别上海证券交易所、深圳证券交易所、香港证券交易所的所有上市公司名称，转换为股票代码 |
| 人员 | 人员相关属性，包括性别、年龄、资产、常用设备地址和访问地址，转换为字符串 |
| 时间 | 所有时间相关实体，转换为 ISO_8601 日期格式标准 |
| 交易频率 | 所有频率词相关实体，转换为元组 (次数, 时间段) |
| 交易金额 | 所有交易金额词相关实体，限定单位为人民币，转换为数字 |
| 交易行为 | 动词结构，如买入、卖出等，转换成对应字符串 |
| 关系运算 | 全部关系运算符和其中文表述，转换成格式化运算符字符串 |

别 $\mathcal{A}$ 类型参数，并对其进行同类运算符的参数值变异，添加该参数值生成新的规则用例，更新到规则集 $U$ 中，横向扩展结束。最后执行规则用例纵向扩展，即将所有的两两成对参数组合规则并入规则集合中，输出最终规则集合，纵向扩展结束。

整体而言，参数化组合规则模块将用户输入的规则文本进行解析，参数化，并且将解析后的参数化组合规则结果存储并展示在前端界面，使得用户可以选择数据生成所需的参数。

## 3.4 结构化测试数据生成模块设计

结构化测试数据生成模块是系统的核心功能，它负责为用户提供结构化测试数据生成服务，分为规则解析、数据初始化和任务监控三部分。规则解析包括对参数化组合规则模块生成的规则集合进行解析，并且映射规则集合到 MySQL 数据库。数据初始化服务包含环境初始化和边界条件初始化这两种方式，负责对测试数据生成提供基础数据支撑。最后，任务监控模块负责将测试数据生成的核心计算环节分发到 Worker 节点上进行运算。

结构化测试数据生成模块类图如图 3-10所示，其中，TextGeneration 为该模块的入口接口，它被 RuleAnalyseService、DataInit 和 TaskStart 这三个大类实现。三个类分别实现了规则解析、数据初始化和任务监控的基础功能和相关函数。

RuleAnalyseService 类调用 RuleLoader 来进行规则读取，其中，load_ruleset() 方法用来与 MySQL 数据库进行通信，读取参数化组合规则模块存入其中的规

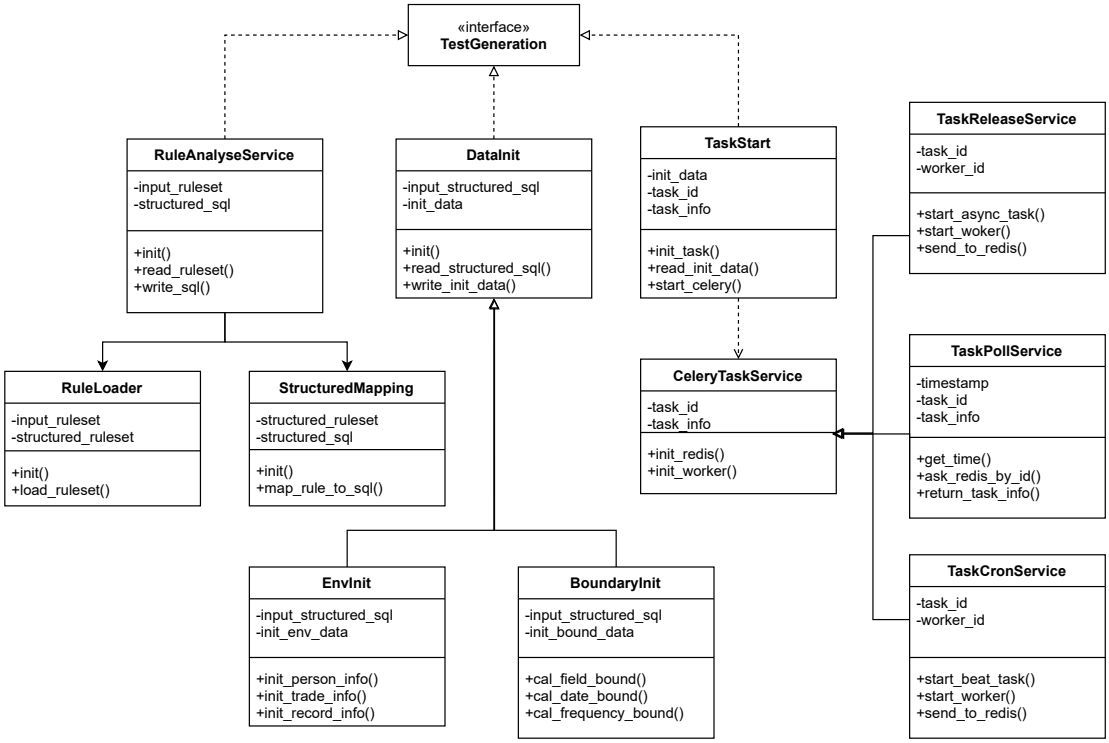


图 3-10: 结构化测试数据生成模块类图

则集合数据。StructuredMapping 类用来生成规则集合和 SQL 的映射关系，根据规则中的参数和数据库中的字段进行比对，自动生成对应的结构化 SQL 语法。

EnvInit 类和 BoundaryInit 类继承了 DataInit 父类，用于对生成任务的数据进行初始化操作。数据初始化的流程如图 3-11所示，详细描述了数据初始化的整体流程运转。EnvInit 类主要是初始化环境数据，init_person_info() 函数用于初始化股票交易的人员的基本信息，包括人员的姓名、身份证、出生日期、手机号等基础信息。init_trade_info() 用于初始化股票交易信息，包括真实的股票交易行情、交易人员的资金情况、交易日期等交易操作数据。init_record_info() 用于初始化用户的初始操作记录，一切初始化数据都模拟真实的软件运行环境和交易环境。另一个类 BoundaryInit 是针对数据生成规则中参数的边界条件进行初始化。cal_field_bound() 方法针对值域，计算参数字段的边界值，给出边界值条件。cal_date_bound() 方法针对股票交易日期，由于用户的交易行为可能不是在一个交易日期内完成，而是分散在规则定义的某个时间段内，所以需要计算出股票交易日期的边界条件。cal_frequency_bound() 方法针对用户股票交易的频率进行分析，计算出用户交易的频率边界条件。

在做好初始化工作后，TaskStart 类会发起测试数据生成计算任务。

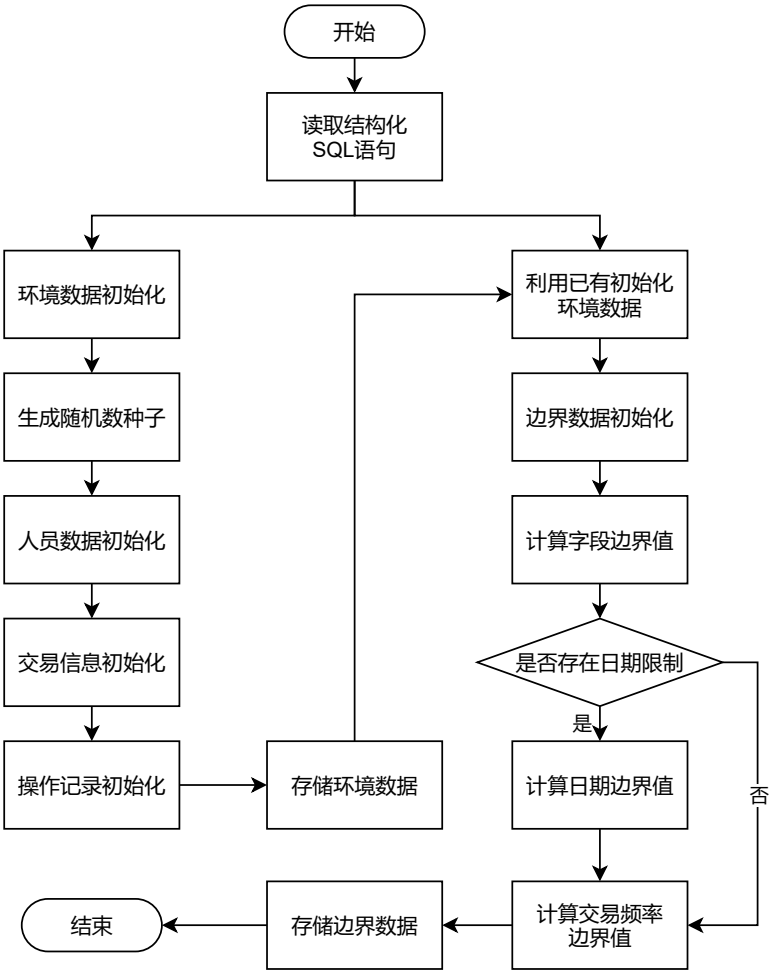


图 3-11: 结构化测试数据生成模块流程图

TaskStart 类会启动 Celery 服务来发起任务，start_celery() 会调用 CeleryTaskService 类，用于初始化 Redis 数据库和 Worker 服务器。TaskReleaseService、TaskPollService 和 TaskCronService 类继承 CeleryTaskService 父类，实现了异步任务分发、任务状态轮询和定时任务执行的功能。TaskReleaseService 调用 start_async_task() 方法发起异步任务，Redis 作为 Celery Broken 消息队列，生成任务队列 Task Queue。接下来，start_worker() 方法调用 worker 服务器，从 Broken 接受异步任务请求，作为任务消费者开始执行异步任务。任务完成后，调用 send_to_redis() 方法将任务结果信息存储到 Celery Backend(Redis)。TaskPollService 类会定时获取系统时间，通过 ask_redis_by_id() 方法查询 Redis 消息队列中的任务 ID，并且返回任务结果给前端，从而实现实时监控任务状态的功能。TaskCronService 类区别于 TaskReleaseService 的在于 start_beat_task() 方法，它是由 Beat Scheduler 定时发起的请求，取决于设置的任务执行时间。

## 3.5 结果分析模块设计

结果分析模块承接数据生成模块产出的数据，对数据生成的时间、数据生成的规模、数据生成的质量进行分析与展示，从而展现给用户提供测试数据的基本表现和质量度量。

结果分析模块类图如图 3-12所示，ResultAnalysis 类作为本模块的接口，会调用 task_state_listener() 方法监听 Celery 的 Backend，当任务执行完毕后，返回 Redis 中的任务状态，进而触发结果分析模块的进程。

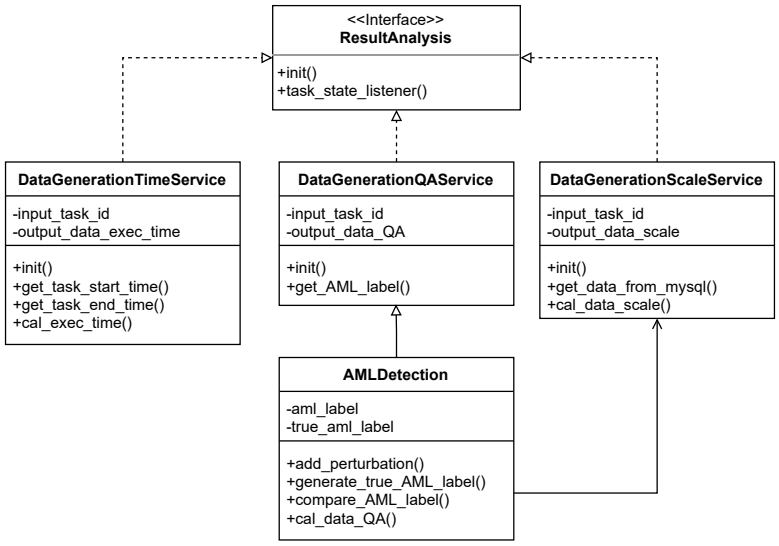


图 3-12: 结果分析模块类图

其中，DataGenerationTimeService 类负责分析计算结构化测试数据生成的耗费时间。get_task_start_time() 方法通过 task_id 参数，找到 worker 的任务开始时间并且返回；get_task_end_time() 方法通过 task_id 参数，直接查询 Backen Redis 数据库中的任务终止时间并且返回。最后，通过 cal_exec_time() 方法计算出 data_exec_time，也就是数据生成的时间。

DataGenerationScaleService 类是用来分析生成测试数据的规模大小，由于基于规则生成结构化测试数据，并不是一对一的简单对应关系，而是一对多的映射关系，所以需要从 MySQL 表中提取出相关联的生成数据并计算行数。get_data_from_mysql() 方法实现了与 MySQL 数据库的交互逻辑，并且从中提取相关规则的数据行并且返回，交给 cal_data_scale() 方法分析计算，得到数据规模，数据规模包括股票交易市场的用户数据规模、交易数据规模、标签数据规模和总体数据规模。

DataGenerationQAService 类是结果分析模块中最关键的部分，它用于评估和分析生成的测试数据的数据质量，呈现数据质量在实际使用过程中的准确性、丰富性和覆盖率这三个维度，数据质量分析流程图如图 3-13所示。add_perturbation() 方法在传入生成数据和对应生成标签之后，会对每一条生成数据进行洗钱规则判断，如果触发洗钱规则，则说明扰动值改变了生成标签的判定，则进行生成标签的变异操作，对洗钱或者非洗钱行为进行更改。如果没有触发规则检测，则说明扰动量并未更改标签。此时进入计算真实标签流程，调用 generate_true_AML_label() 方法，通过代入规则和数据的字段值进行计算，得到每条结构化数据的真实洗钱标签。接下来，调用 compare_AML_label() 方法，通过对比生成标签和计算得到的真实标签，计算数据的准确性。接下来，通过参数化组合规则构造的规则集合条数进行统计数据，计算根据相应规则生成的数据的数据覆盖率。最后，调用数据规模类 DataGenerationScaleService，对数据的丰富性指标进行评估，综合得出数据质量的三个维度指标，流程结束，

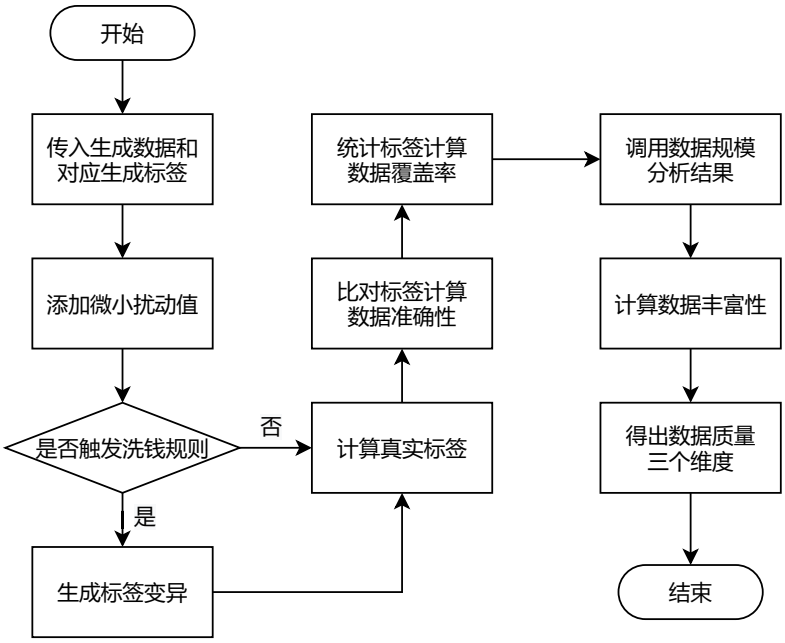


图 3-13: 数据质量分析实现流程

## 3.6 数据管理模块设计

数据管理模块负责对用户在系统中生成过的测试数据进行管理，本模块的类图如图 3-14所示。首先，用户进入数据管理界面，此时后台请求 MySQL 数据库进行校验，检测用户是否在数据库中存在暂时未删除的生成数据，如果不存在，则显示空白列表页面，如果存在，则通过 DataDisplay 类，调用 select_data() 方法显示用户生成的测试数据的关键字段的值列表。列表显示了数据的生成日期、生成标签、生成对应规则等基本信息，并且提供了删除数据和下载数据的相关选项。其中，DataDownload 类提供了两种结构化测试数据的下载方式，分别是 SQL 语句文件形式和 Excel 文件形式。DataDelete 类调用 delete_data() 方法，根据用户选中的数据的主键，从 MySQL 数据库中删除所有相关的用户操作记录，避免造成数据库的大量冗余数据。

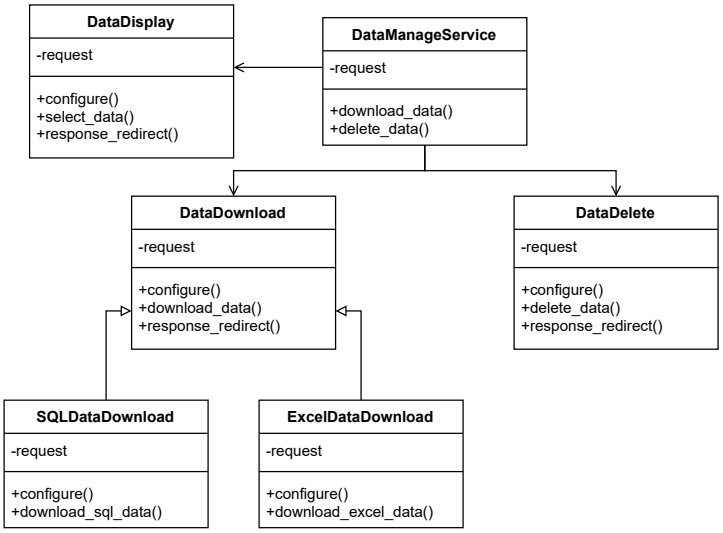


图 3-14: 数据管理模块类图

## 3.7 数据库设计

本章节介绍基于反洗钱规则的结构化测试数据生成系统中涉及到的数据库结构。在整个系统流程中，结构化测试数据生成之后需要存储在数据表之中，模拟真实的股票交易操作，以便对反洗钱监控系统进行测试，直接连接数据库对测试数据进行批量扫描与检测。同时，用户也可以从数据管理模块迅速地寻找和操作到存在数据表中的结构化数据。本节根据数据模型中的数据对象，

生成实体关系 ER 图来描述实体信息、实体中的属性信息和实体之间的关联信息，并给出数据库表的详细说明作为依据。本系统数据库分为系统数据管理数据库和股票交易数据库，股票交易数据库参照中国人民银行下发的《证券机构反洗钱执法检查数据提取接口规范（试行）》[①]制定表结构。

如图 3-15所示的实体关系图，表示了系统数据管理数据库的存储关系。系统的数据管理的核心围绕用户的数据生成任务展开，任务相关信息存储在 Task 表中，包括任务 ID 字段 task_id，任务名称字段 task_name，任务创建者字段 owner，任务执行机器 ID 字段 worker_id，任务开始时间 start_time，任务完成时间 end_time 和任务的生成规则 rule。用户表 User 用于存储系统用户的基本信息，和 Task 表是 1 对 N 的关系，也就是一个用户可以对应多个任务。User 表包括用户 ID user_id，用户名称 user_name，用户密码 password 等用户信息，用于登录平台的权限认证。DataLink 表负责存储数据生成的股票账户信息，与 Task 表是 N 对 1 的关系，它包含了股票账户 ID securities_acc_no，洗钱标签 aml_label 字段和 task_id 字段，用于使得任务和生成数据之间构成对应关系。

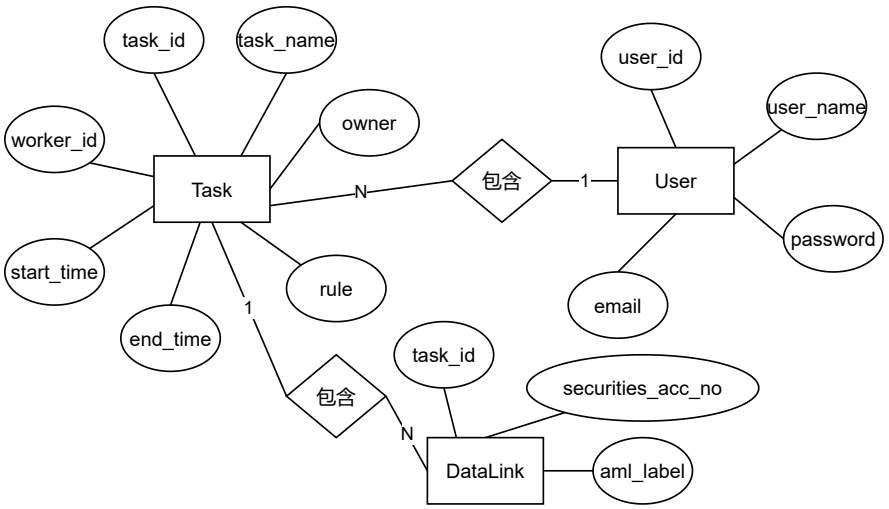


图 3-15: 系统数据管理数据库实体关系图

如果 3-16所示，该 ER 图描述了股票交易数据库实体间的相互关系。整体的股票交易操作主要围绕 TbDeliever 表展开，该表定义了股票交易委托操作的全部字段。TbDeliever 表作为测试数据生成的主要表结构，每条表记录对应一个用户信息，存储在 TbCstPers 表中，包括资金账号 securities_acc_no，客户号 cst_no，客户姓名 cst_name，创建日期 open_time，销户日期 close_time，客

[①]参照中国人民银行关于印发《证券期货保险机构反洗钱执法检查数据提取接口规范》的通知及相关文件。

户性别 cst_sex，国籍和地区 nation，身份证件号码 id_no，年收入 income 和联系方式 contact。存量客户当前风险等级表 TbRisk 表和 TbCstPers 表是一一对应的，它负责存储每个用户的操作风险等级，包含风险等级 risk_code，更新日期 time，评定分值 score 和账户风险划分 norm。

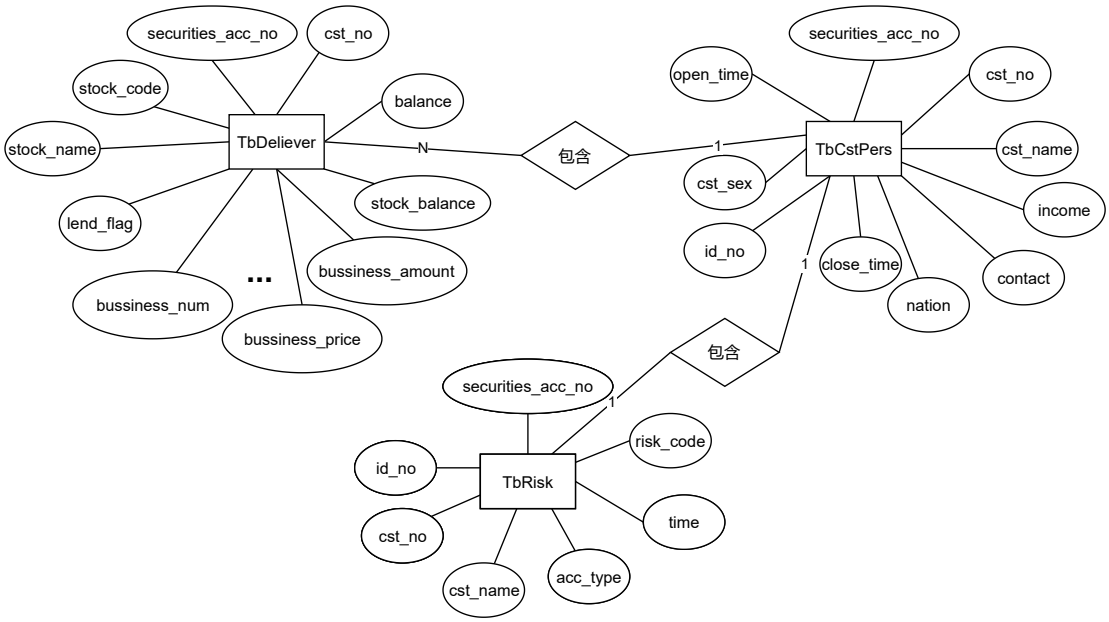


图 3-16: 股票交易数据库实体关系图

总体而言，数据库表结构设计如下所述：

数据生成任务表 Task：如表 3-7所示，包括任务 ID 字段 task_id，任务名称字段 task_name，任务创建者字段 owner，任务执行机器 ID 字段 worker_id，任务开始时间 start_time，任务完成时间 end_time 和任务的生成规则 rule。Task 表主要用于记录数据生成任务的基础信息，在数据管理模块可以通过任务 ID 查询相关任务生成数据。

用户表 User：如表 3-8所示，包括用户 ID user_id，用户名称 user_name，用户密码 password，用户注册邮箱账号 email。User 表主要用于记录系统操作用户的基础信息。为了保证安全，用户密码采用 MD5 算法进行加密，防止非法入侵数据库行为导致的密码泄露问题。

生成数据信息连接表 DataLink：如表 3-9所示，包含了股票账户 ID securities_acc_no，洗钱标签 aml_label 字段和 task_id 字段。

证券交割表 TbDeliever：如表 3-10所示，本表数据范围为检查对象的所有客户（不包括产品户）的资金账户在检查期限内已成交的所有证券交易，仅涉

表 3-7: 数据生成任务 Task 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| task_id | varchar(40) | 是 | 是 | 任务 ID |
| task_name | varchar(60) | 否 | 是 | 任务名 |
| owner | varchar(40) | 否 | 是 | 任务创建者，对应 user_id |
| worker_id | varchar(40) | 否 | 是 | Celery worker 机器 ID |
| start_time | datetime(8) | 否 | 是 | 任务开始时间 |
| end_time | datetime(8) | 否 | 是 | 任务结束时间 |
| rule | varchar(60) | 否 | 是 | 数据生成规则 |

表 3-8: 用户 User 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| user_id | varchar(40) | 是 | 是 | 用户 ID |
| user_name | varchar(50) | 否 | 是 | 用户姓名 |
| password | varchar(32) | 否 | 是 | 用户密码，MD5 算法加密 32 位 |
| email | varchar(50) | 否 | 是 | 注册邮箱账号 |

及经纪业务，不包括分红、结息等非主动发起的交易。此表是本系统进行股票交易数据生成的核心表。

存量个人客户身份信息表表 TbCstpers：如表 3-11所示，相关说明如下：1. 本表数据范围为截止检查期限结束日，检查对象提供过服务（产品）的所有个人客户最新的身份信息记录。2. 对于检查期限起始日至提取数据日期间销户的客户，其身份信息也应导入本表；对于检查期限起始日前已销户的客户，其身份信息可不导入本表。3. 如客户身份信息要素涉及多个系统，应分别从各系统取数，确保提供要素的完整性。每个客户号均生成一条完整的记录。

证券交割表 TbRisk：如表 3-12所示，本表数据范围为截止取数日，券商提供过服务的客户（不包括产品户）的当前的风险等级划分记录。

表 3-9: 数据连接表 DataLink 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| task_id | varchar(40) | 是 | 是 | 任务 ID |
| securities_acc_no | varchar(40) | 是 | 是 | 资金账号，与任务 ID 共同组成主键 |
| aml_label | varchar(2) | 否 | 是 | 洗钱标签 (1-洗钱，2-非洗钱) |

表 3-10: 证券交割表 TbDeliever 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| securities_acc_no | varchar(40) | 是 | 是 | 资金账号 |
| cst_no | varchar(50) | 否 | 是 | 客户号 |
| cst_name | varchar(120) | 否 | 是 | 客户姓名 |
| ticd | varchar(40) | 否 | 是 | 流水序号 |
| report_no | varchar(40) | 否 | 是 | 交易委托编号 |
| date | datetime(8) | 否 | 是 | 交易委托日期 (格式 YYYYMMDD) |
| time | varchar(6) | 否 | 是 | 交易委托时间 (格式 HHMMSS) |
| date1 | datetime(8) | 否 | 是 | 成交日期 (格式 YYYYMMDD) |
| time1 | varchar(6) | 否 | 是 | 成交时间 (格式 HHMMSS) |
| stock_code | varchar(6) | 否 | 是 | 股票代码 |
| stock_name | varchar(32) | 否 | 是 | 股票名称 |
| lend_flag | varchar(2) | 否 | 是 | 资金流向 (11-买，12-卖，13-无变化) |
| bussiness_num | decimal(19,2) | 否 | 是 | 成交数量 |
| business_price | decimal(18,3) | 否 | 是 | 成交价格 |
| bussiness_amount | decimal(19,2) | 否 | 是 | 成交金额 |
| balance | decimal(19,2) | 否 | 是 | 账户余额 |
| stock_balance | decimal(19,2) | 否 | 是 | 证券余额 |
| IP_address | varchar(39) | 否 | 是 | IPV4 地址，以点分十进制记录 |
| mac_address | varchar(12) | 否 | 否 | 电脑端 MAC 地址 |
| imei | varchar(15) | 否 | 否 | 安卓端操作时填写 |
| uuid | varchar(32) | 否 | 否 | iOS 端操作时填写 |

表 3-11: 存量个人客户身份信息表 TbCstPers 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| securities_acc_no | varchar(40) | 是 | 是 | 资金账号 |
| cst_no | varchar(50) | 否 | 是 | 客户号 |
| cst_name | varchar(120) | 否 | 是 | 客户姓名 |
| open_time | datetime(8) | 否 | 是 | 创建日期 (格式 YYYYMMDD) |
| close_time | datetime(8) | 否 | 是 | 销户日期 (格式 YYYYMMDD，如果客户未销户填写 99991231) |
| cst_sex | varchar(2) | 否 | 是 | 客户性别 (11-男，12-女) |
| nation | varchar(20) | 否 | 是 | 国籍（地区），按照 GB/T 2659-2000 世界各国和地区名称代码标准填写 |
| id_no | varchar(50) | 否 | 是 | 身份证件号码 |
| income | decimal(16,2) | 否 | 是 | 年收入 |
| contact | varchar(40) | 否 | 是 | 联系方式 |

表 3-12: 存量客户当前风险等级表 TbRisk 表关键字段结构

| 字段名 | 类型 | 主键 | 非空 | 说明 |
|---|---|---|---|---|
| securities_acc_no | varchar(40) | 是 | 是 | 资金账号 |
| cst_no | varchar(50) | 否 | 是 | 客户号 |
| cst_name | varchar(120) | 否 | 是 | 客户名称 |
| id_no | varchar(50) | 否 | 是 | 身份证件号码 |
| acc_type | varchar(2) | 否 | 是 | 公私标识（11-个人，12-单位) |
| risk_code | varchar(2) | 否 | 是 | 风险等级（10-高，11-中，12-低） |
| time | datetime(8) | 否 | 是 | 最新一次更新日期（格式 YYYYM-MDD） |
| score | decimal(3,2) | 否 | 是 | 评定分值 |
| norm | varchar(1000) | 否 | 是 | 划分依据 |

## 3.8 本章小结

本章将基于反洗钱规则的结构化测试数据生成系统划分为参数化组合规则模块、数据生成模块、数据管理模块和结果分析模块，第一步，针对这四个模块进行具体的需求分析。第二步，描述系统总体架构设计，以“4+1”视图为切入点对系统的四个模块的架构进行全面的阐述和设计。最后进行数据库设计，通过实体关系图来描述系统的数据库关系，并且详细描述数据库各表的基本字段和功能。

# 第四章　系统实现

本章围绕金融领域的反洗钱数据生成任务案例，介绍了基于反洗钱规则的结构化测试数据生成系统在参数化组合规则模块、数据生成模块、结果分析模块和数据管理模块的核心实现。具体而言，介绍了各模块在流程以及算法方面的实现，通过结果分析模块分析并展示了结构化测试数据的生成结果。最后，对系统的核心功能进行全面的系统测试。

## 4.1　参数化组合规则模块

参数化组合规则模块的系统界面如图 4-1所示，用户输入反洗钱规则文本，系统对规则文本进行解析，分析出其中的参数，并且提供让用户选择的参数值和生成个数，注意生成个数是基于股票交易的用户数量，与数据库中的操作记录存在一对多的关系。


图 4-1: 参数化组合测试模块系统界面展示

文本预处理是参数化组合规则模块的头部步骤，其目的是在用户提交规则文本后，调用分词 API，对文本进行分词操作，并且返回分词后的结构化数据存储在 Json 格式文件中。除了图 4-1中展示案例，本文给出另一则反洗钱规则文本描述案例：“大量或频繁进行股票的高买低卖，且大幅亏损。指标描述：1. 大量：单笔委托金额 >= 人民币 400 万；2. 频繁：10 个交易日内 >=3 次；3. 高买低卖的价格偏离标准：（加权平均买入价-任一笔卖出价）/加权平均买入价 *100%>=2%；4. 亏损金额标准：当日亏损 20 万元以上；5. 本指标不设预警间隔，触发即预警。”可以看到，标准的反洗钱规则本文会对洗钱行为进行详尽的描述和相关触发规则，其中的关键参数可以灵活改变。

```
class TextSegmentation(TextPreprocessService):
    def __init__(self, input_text):
       self.input_text = input_text
    ...
    # 初始化金融反洗钱领域专业词典
    def init_aml_dict(self, input_dic):
        jieba.load_userdict(input_dic)
    ...
    # 分词处理
    def segment(self):
        result = jieba.posseg.cut(self.input_text)
        json_data = json.dumps(result)
        return json_data
```

图 4-2: 反洗钱规则文本预处理核心代码

图 4-2是参数化组合规则模块中文本预处理的核心代码。可以看到，预处理类首先用 init_aml_dict() 加载了金融反洗钱领域的专业词典，本系统采用“CNEconDict”中文经济词典[①]，包含经济学、金融学专业词汇主要有如下几类：上市公司名称、知名公司的名称与简称、行业名称、金融产品名称、货物产品名称等，可以更精确的辅助分词。接下来，segment() 函数调用 jieba 分词 API 对文本进行分词，返回分词集合 WS（WordSet）。

接下来进行参数约束关系分析，提取参数可能取值范围，并且提供用户可选择的取值选项。参数化组合规则如上述所示实例，采用逐参数扩展算法生成规则集合，本案例所示，可以根据四个参数生成 54 条规则组成的集合。

[①]参照https://github.com/sijichun/CNEconDict。

## 4.2 数据生成模块

结构化测试数据生成模块是系统的核心功能，它负责为用户提供结构化测试数据生成服务，主要分为规则解析、数据初始化和任务监控三部分。

规则解析部分的主要任务是针对规则集 $U$ 中的规则用例，映射规则中参数到 MySQL 数据库的相应字段，相关规则参数与数据库字段映射关系如表 4-1所示。

表 4-1: 规则参数与字段映射关系表

| 规则参数 | 字段名称 |
|---|---|
| $c$ | TbDeliever.stock_code |
| | TbDeliever.stock_code |
| $p$ | TbCstpers.cst_sex |
| | TbCstpers.nation |
| | TbCstpers.id_no |
| | TbCstpers.income |
| | TbCstpers.contact |
| | TbRisk.acc_type |
| | TbRisk.risk_code |
| $t$ | TbDeliever.date |
| | TbDeliever.time |
| | TbDeliever.date1 |
| | TbDeliever.time1 |
| $a$ | TbDeliever.bussiness_num |
| | TbDeliever.bussiness_price |
| | TbDeliever.bussiness_amount |
| | TbDeliever.balance |
| | TbDeliever.stock_balance |
| $d$ | TbDeliever.lend_flag |

数据初始化主要目的是模拟真实的软件环境，初始化交易人员的基本信息和股票交易账户行为，模拟用户生成的核心代码如图 4-3所示。可以看到，create_person() 方法通过 random.choice() 方法，对所有的相关数据（MAC 地址、IP 地址、用户 ID 等）的随机数实现了均匀分布，均匀分布的特点是测量值在某一范围中各处出现的机会一样，即均匀一致。此外，对于用户的年龄数据，通过 random.normal() 方法，对随机数采用正态分布的方式。这样对于数据初始化而言，能尽可能的贴近真实的数据信息。所有用户的股票信息和价格都

是通过 tushare 接口调取真实股票交易市场的实时数据，保证了模拟环境的真实性。

```
def create_person(start, end, path):
    print(path)
    person_dict = {}
    for name in name_chosen[start: end]:
        uln = conf.get_uln()
        acc_no = random.sample(uln, 12)
        # 生成Capital_acc_no
        Capital_acc_no = ''.join(acc_no)
        head = conf.get_head()
        sec = conf.get_sec()
        # 生成Securities_acc_no
        Securities_acc_no = random.choice(head) + ''.join(random.sample(sec, 9))
        # 生成ip
        ip = random.choice(ip_list).replace('0/', '')
        # 初始化股票基本信息
        stock_num = random.randrange(100, conf.get_max_stock_num(), 100)
        stock = random.choice(stock_name)
        datetime = random.choice(cal_date)
        data = pro.daily(ts_code=stock[0], trade_date=datetime)
        try:
            price = data.loc[0].high
        except:
            continue
        # 基本信息
        info = {
            'name': name,
            'Stock': {
                'stock_id': stock,
                'buy_info': [stock_num, price]
            },
            'Capital_acc_no': Capital_acc_no,
            'Cst_no': Capital_acc_no,
            'IP_address': ip,
            'MAC_address': "".join(["%02x" % x for x in
                                    map(lambda x: random.randint(0, 255), range(6))]),
            'IMEI': '@N',
            'UUID': '@N',
        }
        if person_dict == {}:
            person_dict = {Securities_acc_no: info}
        else:
            if Capital_acc_no in person_dict.keys():
                print('already')
                continue
            else:
                person_dict.update({Securities_acc_no: info})
                print('[+]',Securities_acc_no)
```

图 4-3: 数据初始化核心代码

当数据初始化完毕后，Django Rest Framework 系统会对唤起 Celery 发起数据生成的异步任务，异步任务核心代码如图 4-4所示。可以看到，用户在前端

点击数据生成按钮后，home() 函数接收了来自前端的请求 request，并用 Celery 模块的 delay() 方法发起异步任务，然后及时返回给前端 HTTPResponse，耗时的异步任务则交给 Celery Worker 进行计算处理。

```
from celery.decorators import task
# 异步数据生成任务方法
@celery_app.task
def start_task(task_id):
    generate_test_data(task_id)
    return True
...
def home(request):
    # 耗时任务，用delay执行方法
    start_task.delay(task_id)
    return HttpResponse(json.dumps(data), content_type='application/json')
```

图 4-4: 发起异步任务核心代码

Celery Worker 分布在多个计算服务器上，因此需要使用进程管理工具 Supervisor 来自动化运行计算服务器的 Celery Worker，并且保留日志记录。计算服务器只需要运行命令 supervisord -c supervisord.conf，就可以启动 Celery Woker 进行数据生成。

在数据生成部分，getBuyInPrice() 通过初始化股票交易日期，代入规则中的频率参数，模拟触发洗钱规则和未触发洗钱规则的交易操作。通过 3.4 小节可以看到，类 BoundaryInit 是针对数据生成规则中参数的边界条件进行初始化。cal_field_bound() 方法针对值域，计算参数字段的边界值，给出边界值条件。cal_date_bound() 方法针对股票交易日期，由于用户的交易行为可能不是在一个交易日期内完成，而是分散在规则定义的某个时间段内，所以需要计算出股票交易日期的边界条件。cal_frequency_bound() 方法针对用户股票交易的频率进行分析，计算出用户交易的频率边界条件。在一定的交易日期内，买卖频次和买卖股票数量均是随机生成，buyOrSell 参数用于控制随机买入或者卖出。

## 4.3 结果分析模块

结果分析模块承接数据生成模块产出的数据，对数据生成的时间、数据生成的规模、数据生成的质量进行分析与展示，从而展现给用户提供测试数据

的基本表现和质量度量。如图4-5所示是结果分析模块系统实现功能图，可以看出，结果分析模块的三个子功能。其中，数据生成规模分析包含人员属性统计、交易规模统计和洗钱行为统计；数据生成质量分析包括准确性指标、覆盖率指标和丰富性指标。

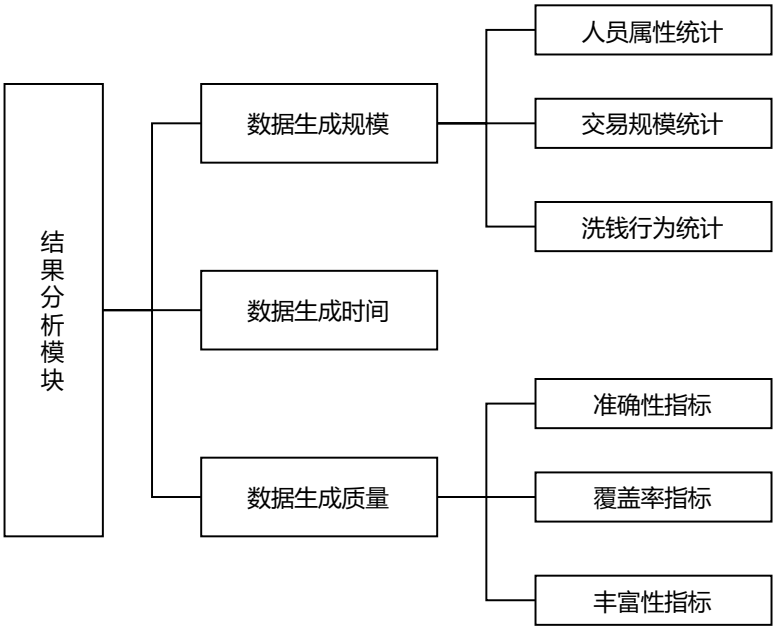


图 4-5: 结果分析模块实现功能图

第一部分是测试数据生成时间结果分析，其系统实现的代码如图 4-6所示，其中，cal_run_time() 方法作为装饰器，在测试数据生成的计算函数 generate_test_data() 前后分别调用系统时间，两者做差得到数据生成时间并且打印保存到日志中。生成时间能合理和有效的反映出系统处理数据的能力，是用户重点关心的需求点，用来评价系统是否能够高效生成大规模测试数据。

```
def cal_run_time(func):
    def wrapper(*args, **kw):
        local_time = time.time()
        func(*args, **kw)
        logging('celery task [%s] runtime is %.2f' % (func.__name__ ,time.time() - local_time) )
    return wrapper

@cal_run_time
def generate_test_data():
    ...
```

图 4-6: 数据生成时间结果

第二部分是测试数据生成规模结果分析，通过 SQL 语句查询结构化测试数

据的人员属性、交易规模和洗钱行为。具体统计的规模描述和对应数据表中的查询 SQL 如表 4-2所示，对于三种规模类型的相关属性实现了数据抽取。具体而言，人员属性包括性别、年龄、年收入和所在区域，使用 SQL 根据对应的字段进行提取和分类。交易规模包括交易次数和交易金额两种相关属性，交易次数统计 TbDeliever 证券交割表中的所有 lend_flag 不为 13，也就是买入或者卖出的交易行为，交易金额可以直接用 bussiness_amount 字段得到。最后洗钱行为主要由洗钱标签来进行判断，DataLink 表中对每个生成用户的洗钱标签做了标记。

表 4-2: 数据生成规模计算 SQL 语句

| 规模类型 | 相关属性 | 对应 SQL 语句 |
|---|---|---|
| 人员属性 | 性别 | select cst_sex,count(*) from TbCstPers group by cst_sex; |
| | 年龄 | select sum(case when datediff(year,birth,getdate()) <= 20 then 1 else 0 end) '20 岁年龄段'; 以此类推 |
| | 年收入 | 同上分类查询，相关字段 TbCstPers.income |
| | 所在区域 | 同上分类查询，相关字段 TbDeliever.IP_address |
| 交易规模 | 交易次数 | select count(*) from TbDeliever where lend_flag<>13; |
| | 交易金额 | select sum(bussiness_amount) from TbCstPers; |
| 洗钱行为 | 洗钱标签 | select * from TbDeliever where DataLink.aml_label=1; |

第三部分是测试数据生成质量结果分析，包括准确性指标、覆盖率指标和丰富性指标。在获取到生成数据和对应的洗钱标签后，会对于每条数据添加微小扰动值，如果触发洗钱规则，则说明扰动值改变了生成标签的判定，则进行生成标签的变异操作，对洗钱或者非洗钱行为进行更改。如果没有触发规则检测，则说明扰动量并未更改标签。接下来计算真实标签，模拟银行的合规监控系统，通过规则计算每条数据的真实标签，然后比对测试数据的洗钱标签计算数据的准确性。接下来，通过参数化组合规则构造的规则集合条数进行统计数据，计算根据相应规则生成的数据的数据覆盖率。最后，调用数据规模类 DataGenerationScaleService，对数据的丰富性指标进行评估，综合得出数据质量的三个维度指标，流程结束。具体而言，丰富性指标依赖于数据生成规模分析的人员属性、交易规模、洗钱行为这三种类型。

## 4.4 数据管理模块

数据管理模块为用户提供了生成测试数据的管理功能。数据管理模块的系统运行截图如图 4-7所示，可以看到，数据管理界面直观的展示了洗钱和非洗钱标签用户的数量，并且提供了根据标签和根据客户 ID 的查询方式，可以进行快速的数据检索与查看。结构化测试数据以列表的形式展现，每行数据包含了触发规则和相关任务的生成日期与 ID，同时提供了洗钱标签筛选和用户 ID 搜索的功能，便于快速搜索相关数据，并且可以选中相应的行数据查看具体的生成用户信息。此外，数据管理模块提供删除数据和批量下载数据的功能，可以选择指定的数据，采用 SQL 或者 EXCEL 的文件格式进行下载。

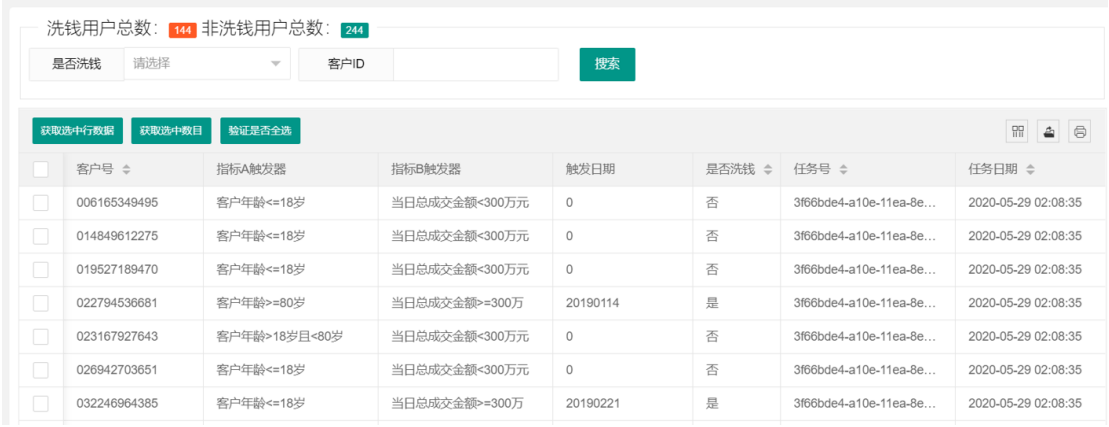


图 4-7: 数据管理模块系统实现

## 4.5 案例分析

本节给出了基于反洗钱规则的结构化测试数据生成的具体应用案例，并且调用结果分析模块，对生成数据的可用性做了分析。本节选取中国人民银行反洗钱条例的真实规则作为用户输入，用户输入的具体反洗钱规则如下：交易人员当日成交总额大于等于 300 万元且交易人员年龄大于等于 80 岁或者小于等于 18 岁，则视为洗钱行为。接下来，参数化组合规则模块会将规则文本进行解析，实验中，用户全部选中参数，则生成规则集合如表 4-3所示。

可以看到，在此规则文本中，系统一共可以解析出两种参数类型，一种是年龄参数，以 18 岁和 80 岁为临界值；另一种是交易额参数，以 3,000,000 人民币为临界值。在此基础上，一共有六种参数化组合规则，其中 2 种定义为洗钱

表 4-3: 生成规则集合表

| 年龄参数 | 交易额参数 | 规则描述 | 是否洗钱 | 标签 |
|---|---|---|---|---|
| age<=18 | daily_amount>=3,000,000 | 小于等于 18 岁，当日交易额大于等于 300 万 | 是 | r1 |
| age>18&&age<80 | daily_amount>=3,000,000 | 18 岁到 80 岁，当日交易额大于等于 300 万 | 否 | r2 |
| age>=80 | daily_amount>=3,000,000 | 大于等于 80 岁，当日交易额大于等于 300 万 | 是 | r3 |
| age<=18 | daily_amount<3,000,000 | 小于等于 18 岁，当日交易额小于 300 万 | 否 | r4 |
| age>18&&age<80 | daily_amount<3,000,000 | 18 岁到 80 岁，当日交易额小于 300 万 | 否 | r5 |
| age>=80 | daily_amount<3,000,000 | 大于等于 80 岁，当日交易额小于 300 万 | 否 | r6 |

规则，4 种为非洗钱规则。系统基于反洗钱规则的结果进行结构化测试数据生成，在此实验中，规定了生成数据的用户规模为 10,000 个。最后对于生成测试数据的结果调用结果分析模块进行分析。

接下来，对于单一轮次的结果进行数据生成规模分析。首先是人员属性规模，其中性别和年龄的分布如图 4-8所示。

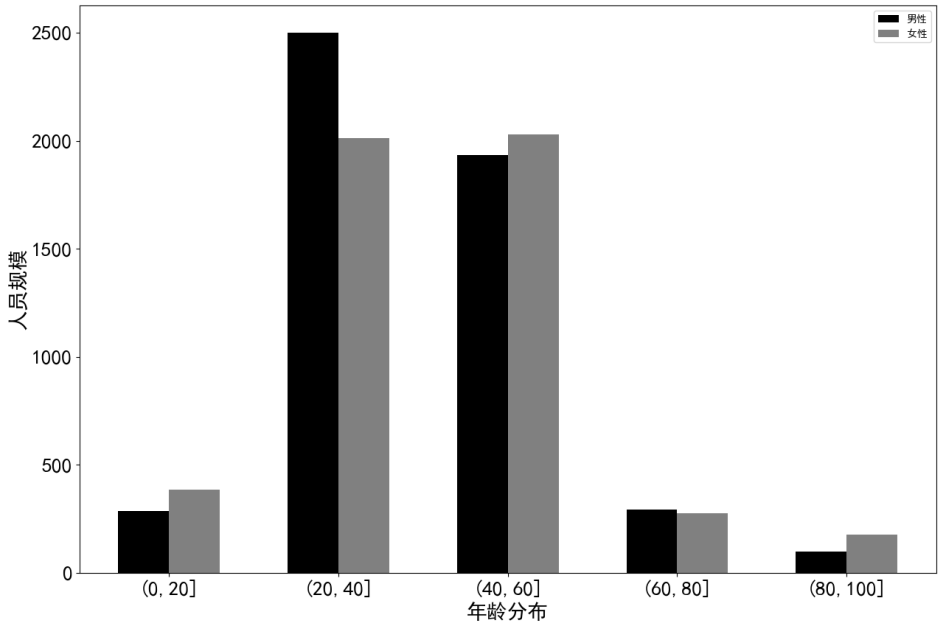


图 4-8: 性别年龄分布规模

可以看到，用户的性别呈现均匀分布，男性和女性比例相近，年龄呈现正态分布，20 岁至 60 岁的交易用户基数最大，符合实际情况。经过 SQL 语句统计查询，数据库中的 TbDeliever 表中一共生成了 123,345 条股票交易数据，时间跨度从 2019 年 1 月 2 日至 2020 年 1 月 23 日。洗钱用户数量为 3325 个，占比 33.25%，洗钱交易操作共有 6140 条，即用户存在多次交易触发洗钱判断的行为。在时间跨度内，123,345 次交易的全部的交易额达到 24,272,828,510 元人民币。

本文选择其中一位洗钱用户的关键数据进行直观展示，如表 4-4所示，可以看到该用户的整体操作行为，所有数据均来自 TbDeliever 股票操作表中（具体字段说明可见 3.7 数据库设计）。其中在 2019 年 2 月 12 日对于某股票进行了大额卖出操作，总操作金额达到 5,282,966 元人民币，且该用户的年龄为 83 岁，触发了生成规则集合中的 r3 规则，视为洗钱用户。

表 4-4: 洗钱案例展示

| Cst_no | Cst_name | Stock_code | Date | Lend_flag | Bussiness_amount | IP_address |
|---|---|---|---|---|---|---|
| D840753162 | 梁 XX | 002089 | 20190102 | 12 | 2,731,968 | 203.23.73.24 |
| D840753162 | 梁 XX | 603318 | 20190123 | 11 | 1,664,933 | 203.23.73.24 |
| D840753162 | 梁 XX | 603318 | 20190212 | 12 | 5,282,966 | 203.23.73.24 |
| D840753162 | 梁 XX | 603318 | 20190226 | 12 | 2,151,310 | 203.23.73.24 |
| D840753162 | 梁 XX | 002089 | 20200121 | 12 | 1,974,438 | 203.23.73.24 |

数据生成时间结果分析调用 cal_run_time() 方法进行计算，为了验证数据生成时间的稳定性，本文进行了 5 轮次的规模为 10,000 个用户的数据生成任务，由于数据生成时间涉及数据生成和数据库插入性能，结果合并到 4.6.3 节中进行展示。

数据生成质量结果分析由准确性指标、覆盖率指标和丰富性指标三部分组成。经过自校验计算，准确性为 100%，在一般情况下，本系统的数据生成考虑精度丢失问题在内，因此测试数据的标签准确率不会出现偏差。覆盖率如表 4-5所示，可以看出，数据生成采取了平均分配策略，将 10,000 个用户平均对应至 6 种规则进行数据生成，每种规则均有大量匹配数据，因此数据的覆盖率为 100%。最后是数据丰富性指标，数据丰富性指标以前端图表的方式呈现给用户，展现数据生成规模中的人员时空分布。

表 4-5: 规则覆盖统计表

| 规则编号 | 触发规则数据量 | 洗钱标签数量 | 非洗钱标签数量 |
|---|---|---|---|
| r1 | 1,661 | 1,661 | 0 |
| r2 | 1,665 | 0 | 1,665 |
| r3 | 1,664 | 1,664 | 0 |
| r4 | 1,667 | 0 | 1,667 |
| r5 | 1,667 | 0 | 1,667 |
| r6 | 1,666 | 0 | 1,666 |

## 4.6 系统测试

本章的上述小节分别给出了基于反洗钱规则的结构化测试数据生成系统各个模块的具体设计和实现细节。本节对于上述的系统功能进行了功能测试和性能测试，保证了系统的可靠性和质量。

### 4.6.1 测试环境

系统测试环境如表格 4-6所示。在测试环境中，测试用户通过 Chrome 浏览器对系统进行访问，并且利用专业抓包工具 BurpSuite 对系统进行性能测试。系统应用服务器和存储数据库 MySQL 都部署在同一台 Linux 服务器上，计算服务器负责承担 Celery Worker 的职责，对数据生成任务进行计算。

### 4.6.2 功能测试

本节对系统的具体模块进行功能测试，其中包括四个模块：参数化组合规则模块、数据生成模块、结果分析模块以及数据管理模块，通过模拟用户行为的方式对需求拟定场景用例进行测试。

在参数化组合规则模块，主要包含反洗钱规则文本上传、用户进行参数选择这两种测试场景。表 4-7展示了规则文本上传测试用例，对于系统的文本上传功能进行测试。用户可以在数据生成页面上传数据生成所需的规则文本，系统处理分析传入的文本，如果文本符合规则，则进行参数化组合规则生成。结果表明，测试结果按照正常流程，如果不存在关键参数，提示用户进行重新上传；如果存在关键参数，则将关键参数返回给用户进行选择。流程逻辑符合预测期望，测试完成。

表 4-6: 系统测试环境

| 设备名称 | 参数 | 参数详情 |
| --- | --- | --- |
| 测试用户计算机 | 操作系统 | Windows 10 |
| | Chrome | 75.0.2345.21（64 位） |
| | BurpSuite | v1.7.37 |
| 应用服务器 | 操作系统 | Ubuntu 16.04.6 |
| | 内存 | 8GB |
| | Python3 | 3.6.5 |
| | MySQL | 5.6 |
| | Celery | 4.3.0 |
| 计算服务器 | 操作系统 | Ubuntu 16.04.6 |
| | 内存 | 8GB |
| | Python3 | 3.6.5 |
| | Celery | 4.3.0 |
| | Redis | 6.0.6 |

表 4-7: 规则文本上传测试用例

| | |
| --- | --- |
| **用例编号** | DQ_RS_01 |
| **测试目标** | 用户在数据生成主页面上传数据生成所需反洗钱规则文本，系统处理分析，如果文本符合规则，则进行参数化组合规则生成 |
| **参与者** | 测试用户 |
| **前置条件** | 用户登录系统 |
| **正常流程** | 1. 用户按照系统要求的格式提交反洗钱规则文本<br>2. 使用系统的接口上传文本<br>3. 系统接收上传的文本，对文本进行分词处理<br>4. 系统对分词结果进行关键参数抽取，如果存在不存在关键参数，说明规则文本不符，提示用户重新上传<br>5. 如果存在关键参数，则将关键参数返回给用户进行选择 |
| **预期结果** | 用户上传规则文本后，成功进行规则检测并返回结果，如果文本符合格式，则进行参数化组合规则步骤 |
| **实际结果** | 与预期结果相符 |

用户参数选择测试用例表如表 4-8所示，针对用户选择参数阶段进行测试。主要功能是根据用户上传的规则文本，提供给用户参数选择，然后系统根据选择参数进行规则生成，转入后续的测试数据生成操作流程。结果表明，测试结果符合预测期望。

在数据生成模块，主要包含结构化测试数据生成和任务监控两个测试用例。结构化测试数据生成测试用例如表 4-9所示，针对用户上传的规则文本和

表 4-8: 用户参数选择测试用例

| | |
|---|---|
| 用例编号 | DQ_RS_02 |
| 测试目标 | 根据用户上传的规则文本，提供给用户参数选择，以便于后续的规则生成 |
| 参与者 | 测试用户 |
| 前置条件 | 用户传入规则文本并通过规则文本上传判断 |
| 正常流程 | 1. 读取关键参数字典<br>2. 返回给用户所有关键参数<br>3. 用户进行自主参数选择，并返回结果给系统<br>4. 系统对参数选择结果进行判断，如果用户选择为空，则提示用户重新选择<br>5. 如果选择参数，则进行参数组合，生成规则集合 |
| 预期结果 | 用户选择相关参数后，系统进行参数选择检测，如果符合规定，则进行规则生成步骤 |
| 实际结果 | 与预期结果相符 |

自选择参数，生成基于规则的结构化测试数据。结果表明，系统根据传入的规则文本及用户选择参数，顺利生成基于规则的结构化测试数据，符合预测期望。

表 4-9: 结构化测试数据生成测试用例

| | |
|---|---|
| 用例编号 | DQ_RS_03 |
| 测试目标 | 根据用户上传的规则文本和自选择参数，生成基于规则的结构化测试数据 |
| 参与者 | 测试用户 |
| 前置条件 | 用户传入规则文本并通过规则文本上传判断，用户选择相关参数传入系统并计算出规则集合 |
| 正常流程 | 1. 读取规则集合并且解析<br>2. 将规则集合保存到数据库中，便于用户下次数据生成参考<br>2. 启动 Celery Worker，发起数据生成任务<br>3. 根据规则集合，进行基于规则的结构化测试数据生成<br>4. 系统对任务结果进行查询，如果完成，将生成数据存储在 MySQL 数据库中 |
| 预期结果 | 系统根据传入的规则文本及用户选择参数，生成基于规则的结构化测试数据 |
| 实际结果 | 与预期结果相符 |

任务监控测试用例如表 4-10，针对用户创建测试数据生成任务后，使用任务监控功能监控任务完成状态功能进行测试。结果表明，用户可以顺利进入任务监控界面，成功查询任务进行状态，符合预测期望。

在结果分析模块，结果分析测试用例如表 4-11所示，针对用户查看数据生成结果模块进行测试。结果表明，用户可以顺利进入结果分析界面，成功查询数据生成结果，符合预测期望。

表 4-10: 任务监控测试用例

| | |
|---|---|
| **用例编号** | DQ_RS_04 |
| **测试目标** | 用户创建测试数据生成任务后，使用任务监控功能监控任务完成状态 |
| **参与者** | 测试用户 |
| **前置条件** | 用户创建测试数据生成任务成功 |
| **正常流程** | 1. 用户打开任务监控页面<br>2. 系统请求 Redis 数据库，查询任务状态，如果用户没有发起过数据生成任务，则跳转到数据生成模块<br>3. 如果用户发起过数据生成任务，返回前端任务列表<br>4. 用户选择具体任务，查看任务进度信息<br>5. 如果任务完成，界面显示任务已完成；如果任务正在进行，界面显示任务正在进行中；如果任务执行失败，界面显示任务执行失败 |
| **预期结果** | 进入任务监控界面，用户成功查询任务进行状态 |
| **实际结果** | 与预期结果相符 |

表 4-11: 结果分析测试用例

| | |
|---|---|
| **用例编号** | DQ_RS_05 |
| **测试目标** | 数据生成任务完成后，使用结果分析模块查看生成结果 |
| **参与者** | 测试用户 |
| **前置条件** | 用户创建的测试数据生成任务执行完成 |
| **正常流程** | 1. 用户打开结果分析界面，如果任务未执行完成，系统提示任务未完成<br>2. 用户查看数据生成时间，系统计算并返回，如果生成任务执行失败，返回时间参数为空<br>3. 用户查看数据生成规模，系统计算并返回，如果生成任务执行失败，返回规模参数为空<br>4. 用户查看数据生成质量，系统计算并返回，如果生成任务执行失败，返回质量参数为空 |
| **预期结果** | 进入结果分析界面，用户成功查询数据生成结果 |
| **实际结果** | 与预期结果相符 |

在数据管理模块中，数据管理测试用例表如 4-12所示，针对系统的数据管理模块功能进行测试。测试用户登录系统，进入数据管理页面，数据管理模块显示用户历史生成的数据列表，提供下载数据和删除数据功能。结果表明，系统显示数据列表，可以成功查看、下载、删除相，符合预测期望。

### 4.6.3 性能测试

为了保证基于反洗钱规则的结构化测试数据生成系统的实用性和可靠性，本小节对系统进行性能测试。本节性能测试采用的工具是 BurpSuite，受制于篇

表 4-12: 数据管理测试用例

| | |
|---|---|
| **用例编号** | DQ_RS_06 |
| **测试目标** | 显示用户历史生成的数据列表，提供下载数据和删除数据功能 |
| **参与者** | 测试用户 |
| **前置条件** | 用户创建的测试数据生成任务执行成功 |
| **正常流程** | 1. 系统从 MySQL 数据库搜索用户生成的数据，返回生成的资金账户基础信息和洗钱标签，展示在列表中<br>2. 用户选择下载数据方式为 SQL 语句，从数据库中选择相应数据并导出成 SQL 文件，下载文件到用户本地计算机<br>3. 用户选择下载数据方式为 Excel 文件，从数据库中选择相应数据并导出成 Excel 文件，下载文件到用户本地计算机<br>4. 用户选择删除数据，从 MySQL 数据库中删除资金账户和所有相关操作记录 |
| **预期结果** | 系统显示数据列表，可以成功查看、下载、删除相关数据 |
| **实际结果** | 与预期结果相符 |

幅限制，文本只探究系统首页压力测试和数据库压力测试。

对系统首页的压力测试通过 BurpSuite 工具的 Intruder 模块完成。该工具可以提供可视化界面对用户指定的接口进行多线程大量并发的模拟发包请求。使用 Intruder 模块进行并发发包时，需要指定的主要参数有：访问总数量（Request_count）、访问并发数（Threads）和待测 URL（Target）。当访问总数量和并发量增大时，被测服务器可能会造成过大压力。所以参考真实环境下的参数设置 $Request\_count \in \{1000, 2000, 3000\}, Threads \in \{10, 20\}$，对系统首页的压测数据如表 4-13所示。可以看到，当访问总数量和访问并发数量上升时，系统的处理速度和传输速率都会有不同程度的下降，但是平均响应时间可以稳定在 2.5ms 以内，达到了快速响应的基本要求。

表 4-13: 系统首页压力测试

| 访问总数量（#） | 并发数量（#） | 每秒处理（#/sec） | 平均响应（ms） | 传输速率（kB/sec） |
|---|---|---|---|---|
| 1000 | 10 | 728.32 | 1.5 | 472.68 |
| 2000 | 10 | 709.05 | 1.7 | 387.44 |
| 3000 | 10 | 692.54 | 1.7 | 350.17 |
| 1000 | 20 | 620.86 | 1.9 | 379.51 |
| 2000 | 20 | 604.10 | 2.1 | 343.72 |
| 3000 | 20 | 578.93 | 2.1 | 320.04 |

在一般系统的压力测试过程中，主要关注的是对接口以及服务器硬件性能进行压力测试，评估请求接口和硬件性能对服务的影响。但是对于基于反洗钱

规则的结构化测试数据生成系统应用来说，整个系统的瓶颈在于对数据的操作，所以对数据库的压测尤为重要。根据 4.5 节中的实验，进行 5 轮次数据生成任务，每轮次构造 10,000 用户数，数据生成任务的耗时如表 4-14所示，可以看到，数据生成任务的耗费时间和数据的交易规模成正相关的关系，稳定在约 10 秒至 80 秒这个合理的秒级区间。

表 4-14: 数据生成耗时表

| 生成用户数量 | 交易规模 | 生成时间（sec） |
|---|---|---|
| 10,000 | 35,621 | 9.81 |
| 10,000 | 52,800 | 11.97 |
| 10,000 | 123,345 | 32.14 |
| 10,000 | 144,485 | 54.35 |
| 10,000 | 194,017 | 77.02 |

表 4-15展示了测试用户在不断发起数据生成任务时，数据库的吞吐量状态十分稳定。当数据生成完成，数据开始写入 MySQL 数据库时，MySQL 每秒接收到的字节数会有较大波动，当读写处于峰值时，为每秒 1-2kBps。在没有生成任务的时候，每秒接收到的 MySQL 字节数的平均值处于合理范围，为 317.41Bps。

表 4-15: MySQL 压测结果

| 关键时间段 | 平均每秒接收字节数（Bps） | 平均每秒发送字节数（Bps） |
|---|---|---|
| 2021-03-07 17:47-17:58 | 250 | 1,704 |
| 2021-03-07 17:58-18:04 | 1,200 | 1,970 |
| 2021-03-07 18:04-18:06 | 374 | 1,832 |
| 2021-03-07 18:06-18:09 | 582 | 1,723 |
| 2021-03-07 18:09-18:20 | 235 | 1,691 |
| 2021-03-07 18:20-18:23 | 877 | 1,850 |

## 4.7 本章小结

本章对系统的各个功能模块进行了具体实现细节的介绍与描述。首先介绍了参数化组合规则模块的具体实现，主要包含分词处理的代码实现、参数化组合规则的核心算法和实现，并给出了系统展示。然后介绍了数据生成模块中规

则解析、数据初始化、数据生成的核心实现，并且详细描述了数据生成任务是如何通过异步任务框架 Celery 搭建在分布式机器上。第三步，介绍了结果分析模块对数据生成任务结果的评估和展示。接下来，描述了数据管理模块对数据的展示、查询、删除和下载操作。在案例与实验分析部分，对系统进行了案例分析，并对生成数据的可用性进行了实验。最后，对系统的核心功能进行了系统测试。

# 第五章　总结与展望

## 5.1　总结

本文依托于中国人民银行的反洗钱监控的真实需求，人民银行会对证券公司的交易数据进行监控，基于规则发掘其中的洗钱数据，并对公司漏检漏报的洗钱交易行为进行大额处罚，这对证券公司的合规监控平台识别洗钱数据的精准度提出了要求。本文根据人民银行提出的反洗钱规则描述，研究基于反洗钱规则的测试数据生成技术，快速生成大量交易数据来验证证券公司合规监控平台对反洗钱数据识别的精准度。

经典结构化测试数据生成使用固定的规则进行数据生成，可以生成满足指定规则的结构化测试数据。然而目前结构化数据生成技术基于固定规则，并不可以灵活改变规则的参数，也没有对规则解耦。一方面，由于规则固定，会导致生成数据的离散且有限，另一方面，当产生新的标准和规则时，需要进行复杂的代码重构来生成新的规则。本文构造的基于反洗钱规则的结构化测试数据生成系统能在股票交易反洗钱领域中处理“难以建模、难以泛化、覆盖率不足”的问题，自动提取相关参数，模拟真实场景进行大规模测试数据生成，拓展结构化测试数据生成在反洗钱领域的应用实践。

本文为用户提供了基于反洗钱规则的结构化数据生成方法，将系统分为参数化组合规则模块、数据生成模块、数据管理模块和结果分析模块，对上述模块进行需求分析和详细设计，保障系统的安全性、可操作性和鲁棒性。

本文为用户提供直观简易的前端操作界面，用于用户自定义规则和生成测试数据，并支持 Excel 和 SQL 形式下载生成的测试数据。此外，实现统一的数据管理，支持生成数据文件的重复下载。整体系统使用 MySQL 作为数据管理系统。

## 5.2 进一步工作

本文初步实现了基于反洗钱规则的结构化测试数据生成系统，满足了相关证券公司对洗钱和非洗钱数据生成的基本要求。但是，目前系统功能较为单一，与用户的交互方式较少。在本文基础上，还有以下方面可以做进一步修改与完善：

第一点，生成技术的可移植性有待提高。本文针对的是股票交易市场的结构化测试数据，根据反洗钱领域规则构造相应规则识别算法。如果移植到其余领域，则需要根据特定场景预先定义好数据库的相关表结构，如何智能化、通用化、自动化识别数据库并且构造相应的参数化组合规则是接下来需要研究的重点。

第二点，系统对数据生成的速度还有待提升，在系统测试中可以看到，数据生成的耗时随着规模扩大可预见性的会突破百秒级别。在数据生成任务执行时，系统仅使用单一 Celery Worker 作为计算资源，数据生成速度受到限制。如何根据数据规模大小做到分布式与弹性计算，提高数据生成效率，是接下来的研究方向。

第三点，测试数据生成技术单一化。在测试数据的生成中，除了本文实现的基于传统算法相关技术，还有一些通用技术，包括深度学习技术可以经过改进应用在金融数据中。此外，对于测试数据的生成，可以定义更广泛维度的测试指标，并根据指标需要建立多种类的测试数据，优化完成的测试数据喂入相关的被测系统，专家可以根据数据的特性、参数的权重逐步构建逻辑清晰、可配置、可扩展的信息度量指标。从而反馈过程调整优化，建立场景结构化数据闭环，持续迭代优化结构化数据的智能化处理过程。多层次信息建模体系应用于不同场景领域，对不同场景实现大数据的智能提升，实现数据治理体系和治理能力现代化。

# 参考文献


[1] CHEN J, CHEN Y, DU X, et al. Big data challenge: a data management perspective[J]. Frontiers of Computer Science, 2013, 7(2): 157 – 164.

[2] KIM G H, TRIMI S, CHUNG J H. Big-data applications in the government sector[J]. Communications of the ACM, 2014, 57(3): 78 – 85.

[3] RABL T, SADOGHI M, JACOBSEN H-A, et al. Solving big data challenges for enterprise application performance management[J]. arXiv preprint arXiv:1208.4167, 2012.

[4] EDIGER D, MCCOLL R, RIEDY J, et al. Stinger: High performance data structure for streaming graphs[C] // 2012 IEEE Conference on High Performance Extreme Computing. 2012: 1 – 5.

[5] 张新琳, 张锐. 多目标跟踪中基于结构化学习的目标身份感知网络流量技术研究 [J]. 中国电子科学研究院学报, 2018, 013(003): 284 – 290.

[6] LIU G, WEI L, ZHANG X, et al. Detecting financial data dependence structure by averaging mixture copulas[J]. Econometric Theory, 2018: 1 – 39.

[7] SYAN M, CHEN, JIAWEI, et al. Data mining: an overview from a database perspective[J]. Knowledge and Data Engineering, IEEE Transactions on, 1996.

[8] ANON. 中华人民共和国反洗钱法 [J]. 中华人民共和国全国人民代表大会常务委员会公报, 2006(37): 5 – 12.

[9] BRATZADEH M, HARATI J, LASHKARI M. A Study of Factors Affecting Iran's Trade Base Money Laundering (TBML): (Ferwerda Gravity Model Application)[J]. Journal of Research in Economic Modeling, 2018, 9(33): 151 – 188.

[10] JALALI N, SEDLAR E, AGARWAL N, et al. Mechanism to efficiently index structured data that provides hierarchical access in a relational database system[J]. US, 2008.

[11] THALHEIM B. Entity-relationship modeling: foundations of database technology[M]. [S.l.]: Springer Science & Business Media, 2013.

[12] KAKSONEN R. Method and arrangement for test case creation[J], 2009.

[13] MALLICK S, MAJUMDAR D. Genetic Algorithm Based Test Data Generation for Structured Software Testing[J]. Journal of the Association of Engineers India, 2015, 85(3-4): 52.

[14] MULLER W H, KALIN C H, GOLDSWORTH J G. Anti-Money Laundering: international law and practice[M]. [S.l.]: John Wiley & Sons, 2007.

[15] MOSTERT D. Utilisation of the financial intelligence centre as a crime intelligence source[J]. university of south africa, 2015.

[16] DREŻEWSKI R, SEPIELAK J, FILIPKOWSKI W. The application of social network analysis algorithms in a system supporting money laundering detection[J]. Information Sciences, 2015, 295: 18–32.

[17] BELLO A U, HARVEY J. From a risk-based to an uncertainty-based approach to anti-money laundering compliance[J]. Security Journal, 2017, 30(1): 1–15.

[18] 朱宝明. 我国银行业反洗钱的成本与收益分析——从博奕论的视角 [J]. 金融研究, 2004, 4: 57–65.

[19] DIANXIANG W, QIANG W, YONGPO X. The Options on Risk and Compliance Management of Securities Companies under the New Normal Conditions[J]. Securities Market Herald, 2017: 01.

[20] SINGHAL A, KASTURI R, SHARMA A, et al. Leveraging web resources for keyword assignment to short text documents[J]. arXiv preprint arXiv:1706.05985, 2017.

[21] ZHANG K, XU H, TANG J, et al. Keyword extraction using support vector machine[C] // international conference on web-age information management. 2006: 85–96.

[22] BELIGA S. Keyword extraction: a review of methods and approaches[J]. University of Rijeka, Department of Informatics, Rijeka, 2014: 1–9.

[23] PALMER D D. Text Preprocessing.[J]. Handbook of natural language processing, 2010, 2 : 9 – 30.

[24] KUHN R, KACKER R, LEI Y, et al. Combinatorial software testing[J]. Computer, 2009, 42(8) : 94 – 96.

[25] KUHN D R, KACKER R N, LEI Y. Practical combinatorial testing[J]. NIST special Publication, 2010, 800(142) : 142.

[26] CZERWONKA J. Pairwise testing in real world[C] // 24th Pacific Northwest Software Quality Conference : Vol 200. 2006.

[27] BIRD D L, MUNOZ C U. Automatic generation of random self-checking test cases[J]. IBM systems journal, 1983, 22(3) : 229 – 245.

[28] HEINZELMAN W R, CHANDRAKASAN A, BALAKRISHNAN H. Energy-efficient communication protocol for wireless microsensor networks[C] // Proceedings of the 33rd annual Hawaii international conference on system sciences. 2000 : 10 – pp.

[29] KOREL B. Automated software test data generation[J]. IEEE Transactions on software engineering, 1990, 16(8) : 870 – 879.

[30] 冯俐. 中文分词技术综述 [J]. 现代计算机 (专业版), 2018.

[31] ZHANG L, LI Y, JIAN M. Design of Chinese Word Segmentation System Based on Improved Chinese Converse Dictionary and Reverse Maximum Matching Algorithm[J]. Web Information Systems-wise Workshops, 2006.

[32] 刘挺, 吴岩, 王开铸. 串频统计和词形匹配相结合的汉语自动分词系统 [J]. 中文信息学报, 1998, 12(1) : 18 – 26.

[33] GAI R L, GAO F, DUAN L M, et al. Bidirectional maximal matching word segmentation algorithm with rules[C] // Advanced Materials Research : Vol 926. 2014 : 3368 – 3372.

[34] HUANG X, LUO Z, JIAN T. A quick method for Chinese word segmentation[C] // Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE International Conference on. 1997.

[35] YUN X. Solving Combinatorial Ambiguity in Chinese Word Segmentation Using Contextual Information[J]. Computer Engineering and Applications, 2001.

[36] ZHANG Q, YU C. Research on Chinese Word Segmentation Algorithm Based on Special Identifiers[M]. [S.l.] : Computing and Intelligent Systems, 2011.

[37] COLLOBERT R, WESTON J, BOTTOU L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(ARTICLE) : 2493 – 2537.

[38] FARMAKIOTOU D, KARKALETSIS V, KOUTSIAS J, et al. Rule-based named entity recognition for Greek financial texts[C] // Proceedings of the Workshop on Computational lexicography and Multimedia Dictionaries (COMLEX 2000). 2000 : 75 – 78.

[39] WU Y, ZHAO J, XU B. Chinese named entity recognition combining statistical model wih human knowledge[C] // Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition. 2003 : 65 – 72.

[40] XIANCHAO T, ZHENXING L, XIA F, et al. A Method of Abbreviated Chinese Organization Names Recognition Based on Similarity[C] // 2013 10th Web Information System and Application Conference. 2013 : 138 – 142.

[41] WU Y, ZHAO J, XU B. Chinese named entity recognition combining statistical model wih human knowledge[C] // Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition. 2003 : 65 – 72.

[42] YANG X, YAN L, YOU H. Chinese organization names recognition combined with CCRF and rules[J]. Computer engineering, 2011, 37(8) : 169 – 171.

[43] SALAZAR A, RODRÍGUEZ J. The use of the load separation parameter Spb method to determine the J–R curves of polypropylenes[J]. Polymer testing, 2008, 27(8) : 977 – 984.

[44] MANDL R. Orthogonal Latin squares: an application of experiment design to compiler testing[J]. Communications of the ACM, 1985.

[45] COHEN D M, DALAL S R, FREDMAN M L, et al. The AETG system: An approach to testing based on combinatorial design[J]. IEEE Transactions on Software Engineering, 1997, 23(7) : 437 – 444.

[46] PARGAS R P, HARROLD M J, PECK R R. Test-data generation using genetic algorithms[J]. Software testing, verification and reliability, 1999, 9(4) : 263 – 282.

[47] LO E, BINNIG C, KOSSMANN D, et al. A framework for testing DBMS features[J]. Vldb Journal, 2010, 19(2) : 203 – 230.

[48] FIELDING R T. REST: architectural styles and the design of network-based software architectures[J]. Doctoral dissertation, University of California, 2000.

# 致　　谢

本论文完成之际，我要向所有关心我帮助过我的人致以最诚挚的感谢。

感谢陈振宇老师。陈老师治学严谨，热爱生活，在我的研究生阶段给了我很大的帮助，从踏入实验室的那一刻，教会了我如何去科研和学习，如何坚持自己喜欢的事物并且做好它。

感谢何铁科老师和刘嘉老师。何老师和刘老师待人友善，有问必答，给予我学习与工作的经验。

感谢老师、同学和我的家人，在学习和生活中给予了我莫大的帮助，在精神上给予我可以依靠的港湾。疫情居家期间给予了我舒适的学习环境，让我可以专心学业。

最后感谢为本文答辩评审的各位老师们，谢谢诸位的辛勤工作。

# 简历与科研成果

## 基本信息

郭子琛，男，汉族，1996 年 12 月出生，江苏省南京人。

## 教育背景

**2019 年 9 月 — 2021 年 6 月** 南京大学软件工程 硕士

**2015 年 9 月 — 2019 年 6 月** 郑州大学软件工程 本科

## 攻读工程硕士学位期间完成的学术成果

1. **Guo Z**, Liu J, He T, et al. TauJud: test augmentation of machine learning in judicial documents[C]//Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2020: 549-552.
2. **Guo Z**, He T, Qin Z, et al. A Content-Based Recommendation Framework for Judicial Cases[C]//International Conference of Pioneering Computer Scientists, Engineers and Educators. Springer, Singapore, 2019: 76-88.

# 《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：郭子琛

2021 年 5 月 20 日

| | | | | | |
|---|---|---|---|---|---|
| 论文题名 | 基于反洗钱规则的结构化测试数据生成系统设计与实现 | | | | |
| 研究生学号 | MF1932061 | 所在院系 | 软件学院 | 学位年度 | 2021 |
| 论文级别 | ☐ 硕士　　☑ 硕士专业学位<br>☐ 博士　　☐ 博士专业学位<br>（请在方框内画勾） | | | | |
| 作者 Email | mf1932061@smail.nju.edu.cn | | | | |
| 导师姓名 | 刘嘉 副教授 | | | | |

论文涉密情况：

☑ 不保密

☐ 保密，保密期 (_____年_____月_____日 至 _____年_____月_____日 )

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。