隐私数据脱敏管理平台

系统设计文档

2024-8-30

脱敏组

目录

[一、 引言 3](#_Toc176108946)

[1.1 编写目的 3](#_Toc176108947)

[1.2 文档范围 3](#_Toc176108948)

[1.3 文档读者 3](#_Toc176108949)

[1.4 参考文献 3](#_Toc176108950)

[二、 总体描述 4](#_Toc176108951)

[2.1 系统功能 4](#_Toc176108952)

[2.2 系统边界 4](#_Toc176108953)

[2.3 用户特性 4](#_Toc176108954)

[2.4 假设和依赖 4](#_Toc176108955)

[三、 系统结构设计 5](#_Toc176108956)

[3.1 系统设计思想 5](#_Toc176108957)

[3.1.1 系统构思 5](#_Toc176108958)

[3.1.2 关键技术与算法 5](#_Toc176108959)

[3.2 基本处理流程 5](#_Toc176108960)

[3.3 系统结构设计 6](#_Toc176108961)

[3.4 系统部件 6](#_Toc176108962)

[3.4.1 用户身份验证组件 6](#_Toc176108963)

[3.4.2 数据处理组件 6](#_Toc176108964)

[3.4.3 任务组件 7](#_Toc176108965)

[3.4.4 安全组件 7](#_Toc176108966)

[3.5 接口设计 7](#_Toc176108967)

[3.5.1 用户接口 7](#_Toc176108968)

[3.5.2 外部接口 7](#_Toc176108969)

[3.5.3 内部接口 8](#_Toc176108970)

[四、 运行与维护设计 8](#_Toc176108971)

[4.1 运行设计 8](#_Toc176108972)

[4.2 错误处理设计 9](#_Toc176108973)

[4.3 维护设计 9](#_Toc176108974)

# 引言

## 编写目的

本系统设计文档的编写旨在全面、详细地描述隐私数据脱敏管理平台的整体架构、功能模块、数据库设计、界面设计以及安全性设计。其主要目标是确保项目团队成员及相关利益方能够清晰理解该平台的设计思路和实现方式，以便指导后续的开发、测试和部署工作。

## 文档范围

本文档覆盖的内容范围包括隐私数据脱敏管理平台的所有功能模块，具体涵盖用户管理、数据上传、读取与展示、数据隐私感知和脱敏任务的管理、任务列表管理以及任务状态显示等核心功能。此外，本文档还将涉及数据库设计、界面设计和安全性设计等方面的详细规划。

## 文档读者

本文档的目标读者包括但不限于：

* 开发人员：为其提供系统编码实现的参考。
* 测试人员：参考以设计测试方案，制定测试计划。
* 项目经理：据此监控项目进度，协调资源分配。
* 项目顾问：据此为系统设计提供建议和指导。
* 客户代表：作为最终用户或利益相关者，据此了解系统功能以确认是否满足业务需求。
* 其他相关人员。

## 参考文献

在编写本文件时，参考了以下文档：

* 本项目的《项目需求规格说明》
* 本项目的《项目概要设计说明》
* 《计算机软件文档编制规范》
* 公司内部文档编写规范
* 内部技术会议记录及讨论纪要

# 总体描述

## 2.1 系统功能

隐私数据管理平台旨在实现以下主要功能：

* **用户管理：**支持用户登录，注册以及权限管理等功能。
* **数据处理：**支持用户上传数据库文件，获取处理后的数据库文件。
* **任务管理：**支持任务状态的查看和管理。
* **结果生成：**支持生成任务处理结果，供用户查看。

## 2.2 系统边界

系统的边界定义如下：

* **内部系统：**指本系统内部各个模块之间相互作用的部分。
* **外部系统：**指本系统与其他系统（如第三方服务、外部数据库等）之间的接口。
* **用户界面：**指用户与系统交互的界面，包括Web端。
* **数据存储：**指系统中涉及的所有数据存储组件。

## 2.3 用户特性

系统的目标用户群体包括：

* **管理员：**负责系统配置、用户管理、权限分配等高级管理任务。
* **普通用户：**使用系统的基本功能，完成日常工作任务。
* **合作伙伴：**通过系统进行业务合作和数据交换。

这些用户有如下典型需求：

* **高效便捷：**系统应具备直观的操作界面和流畅的用户体验。
* **数据准确性：**用户期望系统能提供准确无误的数据支持。
* **响应迅速：**系统需要具备快速响应能力。
* **安全保障：**用户关心其数据的安全性，期望系统具有可靠的安全机制。

## 2.4 假设和依赖

为了确保系统的正常运行，以下假设和依赖条件必须成立：

* **硬件要求：**在运行该系统的平台上，内存要求为2GB以上，存储要求为128GB以上，需要支持虚拟化。
* **操作系统要求：**操作系统需要为Linux或Windows Server。
* **网络连接：**平台需要稳定的互联网连接。
* **数据源：**系统需要接入的数据源必须提供准确、及时的数据更新。
* **用户行为假设：**用户将按照预期的方式使用系统，不进行恶意操作。

# 系统结构设计

## 3.1 系统设计思想

### 3.1.1 系统构思

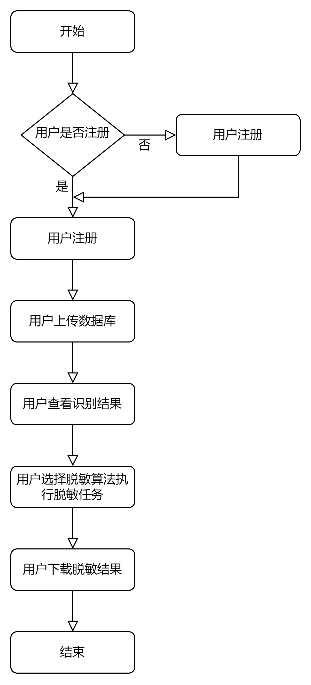
本系统的设计思路为：向用户提供在线的工具服务，系统仅提供简要的数据库服务（查看等），在后台仅存储数据库文件，不运行或运行功能特化的数据库系统用于用户上传的数据处理，与维护系统运行的数据库系统分离。据此系统具有较高的定制性，既可以作为独立的系统运行，也可以与已有数据库结合，或作为云服务的一个部分提供给用户。

### 3.1.2 关键技术与算法

本系统采用的关键技术为识别与脱敏功能与数据库结合的中间件，通过该中间件，可以将不同的脱敏算法和不同的数据库进行结合，从而提高了本系统的通用性。

## 3.2 基本处理流程

系统的处理流程如下图所示：



## 3.3 系统结构设计

系统的整体结构如下所示：



系统的技术选型为：

前端：选用Vue.js作为前端框架，该框架的生态较好，可以在较短的迭代周期内开发出现代的浏览器页面。

后端：使用Java开发，以springboot作为后端框架，该框架可以通过少量的配置快速搭建起一个MVC架构的软件系统，且与其他Java工具的兼容性较好。

用户信息数据库：使用OpenGauss数据库，OpenGauss的安全方案比较成熟，可以较好地保护用户信息。

数据库解析系统：使用SQLite，SQLite能满足本项目对用户上传的数据库文件的解析需求，且轻量化，对系统要求低。

## 3.4 系统部件

本部分将介绍各个组件的功能描述、组件间的关系以及可能的技术实现细节。

### 3.4.1 用户身份验证组件

该组件负责用户的认证与授权，包括但不限于：

* 用户注册、登录验证
  + - 用户角色与权限管理
    - 会话管理与令牌生成

### 3.4.2 数据处理组件

该组件是系统的核心组件，负责处理用户上传的数据库文件，包括但不限于

* + - 数据解析
    - 识别和脱敏算法实现
    - 文件处理

在数据解析部分，本项目使用SQLite来解析数据库，在需要的情况下，也可以换用其他工具。识别和脱敏算法在本项目中由内部实现，也可接入外部数据。

### 3.4.3 任务组件

该组件负责处理用户的任务，包括但不限于：

* + - 管理用户制定的任务
    - 记录用户任务历史
    - 自动执行任务

### 3.4.4 安全组件

安全管理组件关注整个系统的安全防护：

* + - 网络安全防护
    - 数据加密
    - 加密算法管理

组件之间的交互遵循微服务架构原则，各组件通过定义良好的API接口进行通信。

## 3.5 接口设计

### 3.5.1 用户接口

1. 注册窗口：用于新用户注册的界面，包括用户名输入框、密码输入框、确认密码输入框和注册按钮等。
2. 登录窗口：用于用户登录系统的界面，包括用户名输入框、密码输入框和登录按钮等。
3. 数据上传界面：用于用户上传数据的界面，包括输入窗口、上传进度显示和取消上传等。
4. 数据查看界面：用于用户查看脱敏后数据的界面，包括数据列表、数据详情展示等。
5. 任务列表界面：用于显示和管理用户的所有脱敏任务，包括任务进度条、任务执行列表和任务操作按钮等。
6. 日志查看界面：用于维护人员查看系统操作日志，包括日志列表、日志详情等。

### 3.5.2 外部接口

1. 数据加密接口：实现对数据的加密和解密操作。
2. 数据传输接口：实现数据库数据的上传和下载功能。
3. 识别脱敏算法接口：实现利用外部脱敏引擎处理脱敏。

### 3.5.3 内部接口

1. 用户界面(GUI)模块与用户管理模块之间的接口：用于在用户界面和用户管理模块之间传递用户注册、登录、权限验证等数据和命令。
2. 用户界面(GUI)模块与数据库管理模块之间的接口：用于在用户界面和数据库管理模块之间传递数据库文件的上传、下载和展示等数据和命令。
3. 用户界面(GUI)模块与数据脱敏模块之间的接口：用于在用户界面和数据脱敏模块之间传递数据脱敏的请求和结果。
4. 数据脱敏模块与数据库管理模块之间的接口：用于在数据脱敏模块和数据库管理模块之间传递加密、解密和脱敏处理的数据。
5. 用户界面(GUI)模块与任务列表管理模块之间的接口：用于在用户界面和任务列表管理模块之间传递任务管理的请求和进度信息。
6. 用户界面(GUI)模块与日志记录模块之间的接口：用于在用户界面和日志记录模块之间传递操作日志的记录请求和展示命令。
7. 用户管理模块与日志记录模块之间的接口：用于在用户管理模块和日志记录模块之间传递用户操作的相关信息，以便记录日志。

# 运行与维护设计

## 4.1 运行设计

为了保证系统高效运行，应做到以下运行设计：

1. 高可用架构:
   1. 数据库应采用主从复制或集群模式部署，确保数据的一致性和持久性。
   2. 对关键组件采用冗余设计，例如消息队列、缓存服务等。
2. 异步处理:
   1. 对于耗时较长的任务（如大规模数据脱敏），应使用异步消息队列的方式处理，以避免阻塞前端请求。
3. 性能优化:
   1. 使用缓存技术减少对数据库的频繁访问。
   2. 对常用查询进行索引优化，提高查询效率。
   3. 在适当的地方使用批处理逻辑减少I/O操作次数。
4. 监控与报警:
   1. 实施全面的监控系统来跟踪应用程序、数据库和其他基础设施的健康状况。
   2. 设置阈值触发报警机制，当资源利用率超过预设值时通知管理员。
5. 备份与恢复:
   1. 定期自动备份数据库和重要文件。

## 4.2 错误处理设计

为了避免系统错误影响运行，应做到以下错误处理设计：

1. 用户友好的错误提示:
   1. 当发生错误时，向用户显示友好的错误信息而非技术细节。
   2. 提供错误码或链接指向帮助文档，指导用户如何解决问题。
2. 重试机制:
   1. 对于网络请求失败等情况，实现自动重试逻辑。
3. 降级策略:
   1. 当某个服务不可用时，可以暂时关闭该服务的功能或提供简化版本的服务。并记录降级事件，以便后续分析问题原因。
4. 回滚机制:
   1. 对于重要的数据变更操作，实现事务处理，确保要么全部成功要么全部失败。
   2. 在必要时提供手动或自动回滚功能。

## 4.3 维护设计

为使得系统可维护，应做到以下维护设计：

1. 文档化:
   1. 编写详细的系统文档，包括架构设计、接口定义、环境搭建步骤等。
   2. 为开发者和运维人员提供清晰的操作手册。
2. 定期审计:
   1. 定期检查代码质量，寻找潜在的性能瓶颈和安全漏洞。
   2. 执行合规性审计，确保系统符合相关的法律法规要求。
3. 支持与更新:
   1. 使用第三方组件，应及时获取更新和支持。
   2. 对系统进行定期的技术栈更新，以利用最新的技术进步。
4. 培训与知识传递:
   1. 定期为团队成员提供培训，确保他们了解最新的开发实践和技术趋势。
   2. 建立内部知识共享机制，促进最佳实践的传播。