**版本号：1.0.0**

**文档信息**

|  |  |
| --- | --- |
| **项目名称:** |  |
| **标题:** |  |
| **版本:** | 1.0.0 |
|  |  |
| **作者:** | liujiaming |
| **创建日期:** | 2025/01/01 |
| **上次更新日期:** |  |
|  |  |
| **分类:** |  |
| **关键词:** |  |
|  |  |
| **审批人:** |  |
| **审批日期:** |  |

**修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2013/08/19 | 1.0.0 | 创建文档 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 引言 1](#_Toc364934559)

[1.1. 目的 1](#_Toc364934560)

[1.2. 背景 1](#_Toc364934561)

[1.3. 术语与缩写解释 1](#_Toc364934562)

[1.4. 参考资料 1](#_Toc364934563)

[2. 总体设计 2](#_Toc364934564)

[2.1. 需求规定 2](#_Toc364934565)

[2.1.1. 系统功能 2](#_Toc364934566)

[2.1.2. 系统性能 2](#_Toc364934567)

[2.1.2.1. 精度 2](#_Toc364934568)

[2.1.2.2. 时间特性要求 2](#_Toc364934569)

[2.1.2.3. 可靠性 2](#_Toc364934570)

[2.1.2.4. 灵活性 2](#_Toc364934571)

[2.1.3. 输入输出要求 2](#_Toc364934572)

[2.1.4. 数据管理能力要求 2](#_Toc364934573)

[2.1.5. 故障处理要求 2](#_Toc364934574)

[2.1.6. 其他专门要求 3](#_Toc364934575)

[2.2. 运行环境 3](#_Toc364934576)

[2.2.1. 设备 3](#_Toc364934577)

[2.2.2. 支持软件 3](#_Toc364934578)

[2.2.3. 接口 3](#_Toc364934579)

[2.2.4. 控制 3](#_Toc364934580)

[2.3. 基本设计概念和处理流程 3](#_Toc364934581)

[2.4. 结构 3](#_Toc364934582)

[2.5. 功能需求与系统模块的关系 3](#_Toc364934583)

[2.6. 人工处理过程 4](#_Toc364934584)

[2.7. 尚未解决的问题 4](#_Toc364934585)

[3. 接口设计 5](#_Toc364934586)

[3.1. 用户接口 5](#_Toc364934587)

[3.2. 外部接口 5](#_Toc364934588)

[3.3. 内部接口 5](#_Toc364934589)

[4. 运行设计 6](#_Toc364934590)

[4.1. 运行模块组合 6](#_Toc364934591)

[4.2. 运行控制 6](#_Toc364934592)

[4.3. 运行时间 6](#_Toc364934593)

[5. 系统数据结构设计 7](#_Toc364934594)

[5.1. 逻辑结构设计要点 7](#_Toc364934595)

[5.2. 物理结构设计要点 7](#_Toc364934596)

[5.3. 数据结构与程序的关系 7](#_Toc364934597)

[6. 系统出错处理设计 8](#_Toc364934598)

[6.1. 出错信息 8](#_Toc364934599)

[6.2. 补救措施 8](#_Toc364934600)

[6.3. 系统维护设计 8](#_Toc364934601)

# 引言

## 目的

实现根据用户自定义的交易信息，实现对其交易质量的监控，统计，告警，分析。

## 背景

行内bpc系统会分析网络报文，解析出交易信息，云上链路，天生支持交易信息收集。故，对标bpc，实现云上，系统间调用的交易质量监控。

## 术语与缩写解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 参考资料

bpc-oda 的实现

# 总体设计

## 需求规定

用例图

图示

描述已自动生成

### 系统功能

2.1.1.1 配置交易评判标准

需要使用方配置好交易评判标准，即什么样的交易是异常交易如图：

我们认为，默认AAAAAA 和 s 开头的，是成功交易；如果在途中配置，例如bbbb，

那么AAAAAA ，bbbb和s 开头的都是成功的交易

2.1.1.2 交易概览（原型图）

第三步：如图，配置好交易维度，其中新零售中的交易码和交易渠道为固定的交易维度

应用名，为固定的内置维度，主要是给cpaas自己用

点击开启交易质量监控

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

点击保存，保存维度。点击发送，则是发送给oda。权限为项目管理员及以上

2.1.1.4 交易质量多维度查询

2.1.1.4 热加载

点击确认后，query组件

2.1.1.5 数据过滤

2.1.1.6 数据缓存

2.1.1.7 成功率流式计算

如图，我们在redis

### 系统性能

需要**benchmark**

#### 精度

需要**benchmark**

#### 时间特性要求

oda系统对交易质量的告警有着较高的实效性要求。即，需要在交易质量过低时，1min内，反映到oda项目的dashboard上。故不能用日志中心的logstash，和 ElasticSearch来实现数据聚合，因为logstash的延迟难以保证。也不能用prometheus，因为prometheus server的抓取周期和内存以及量的关系，现阶段无法评估。

#### 可靠性

在现有线路的数据流上，新加了一条专门处理交易信息的数据流。该数据流即便是crash了，也不会影响到主逻辑数据流，也就是现有功能不会因为交易信息功能crash导致不可用。

#### 灵活性

具有热加载功能。可以动态调节需要监控，告警的交易维度。

### 输入输出要求

输入

输出

### 数据管理能力要求

### 故障处理要求

### 其他专门要求

## 运行环境

### 设备

### 支持软件

### 接口

[说明该系统同其他系统之间的接口、数据通信协议等。]

### 控制

[说明控制该系统的运行的方法和控制信号，并说明这些控制信号的来源。]

## 基本设计概念和处理流程

### 热加载

点击发送至oda时候，会将选中的接口路径（后面简称交易），发布到redis的channel。db8-tradeInfo这个channel下。信息为

文本

中度可信度描述已自动生成

流程为

点击发送后，会发送事件到redis

图示

描述已自动生成

这个事件，会包含必要信息广播到所有的analysis组件

图示

描述已自动生成

此时会判断bsts的消费情况，并启动数据流，简称pipeline。

图片包含 图示

描述已自动生成

pipeline：一种抽象的数据处理模型（etl模型）。消费，处理，写入

新的消费者会起动，并根据事件查询table trans\_info(暂定)表结构。并初始化下面的处理节点：

**数据过滤节点**：我们只消费头节点以及被设置为需要监控的数据流，因此，除了符合要求的数据，也就是带有交易信息和span头节点的数据，均会被丢弃。

聚合计算节点：数据经过第一轮过滤后，成为如下结构

文本

描述已自动生成

当数据为设置或者默认值（AAAAA）时候，被计为成功，并写入redis key：

格式为<url:cluster:project_name>:成功与否=>统计数字

文本

描述已自动生成

当数据写够1min之后，会将上述数据加工，并发送至odakafka，发送数据结构如下

文本

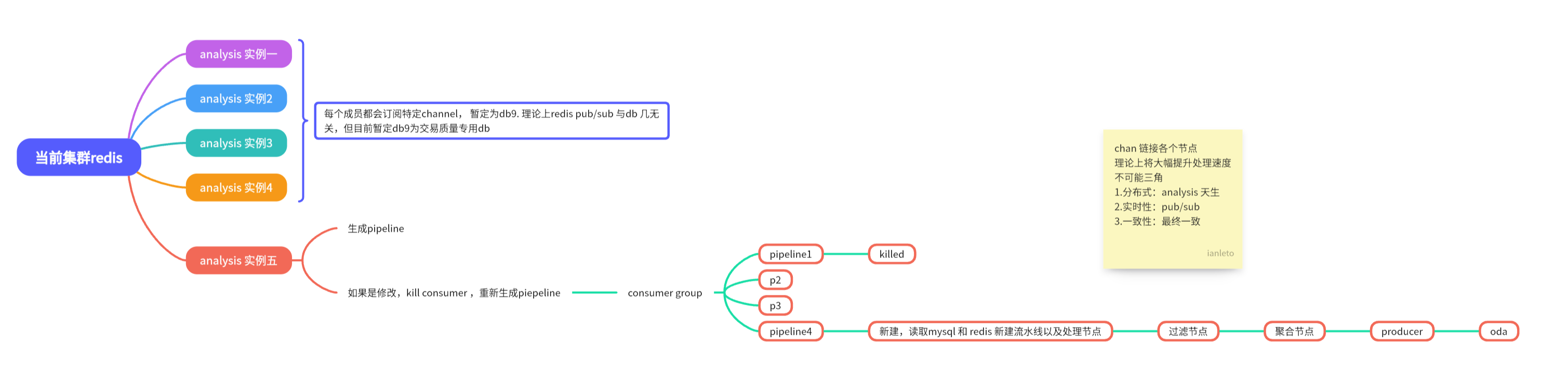
描述已自动生成

从上面可以看出，重点是时效性和数据流的处理。因此，将难点主要分成以下几块。

如图，引入了新的流水线机制和热加载机制。

**第一步：发布信息，当我们配置好维度后，点击发送**

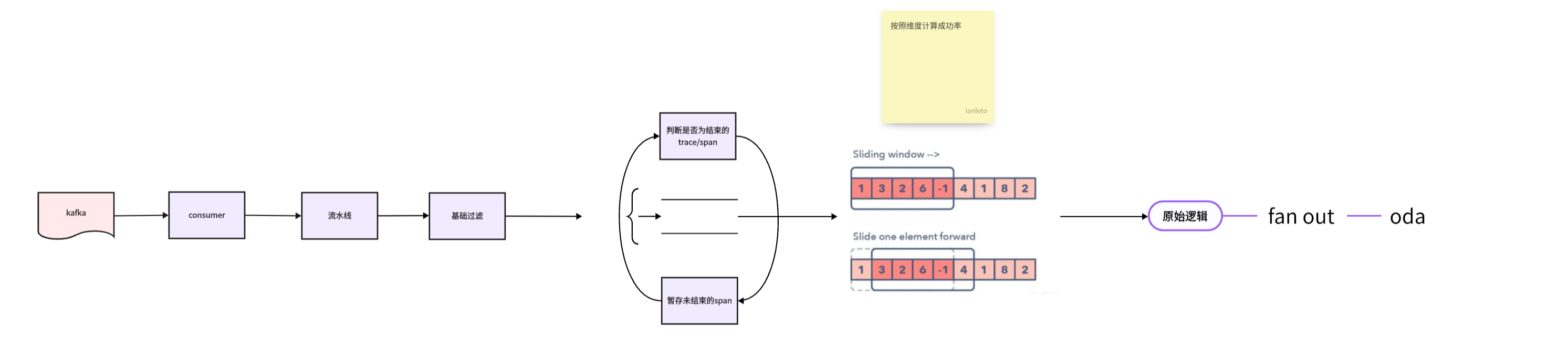
当所有订阅者拿到消息后，会开始分析，最终消费该项目数据的实例生成、或者kill掉自己的pipeline（数据流水线，一个抽象的，自己实现流式处理机制）如图实例五



拿到事件后，analysis会开始创建数据流水线，完成热加载功能

**第二步：数据流水线**

当analysis实例拿到事件后，会着手将数据发送至oda。此时会经历数据流水线，处理流式数据。因为如果将1min钟数据直接存入内存，会导致内存飙升，导致程序直接crash，并且对性能要求很高故此设计利用golang语言天生的流式线模型处理，如图逻辑架构

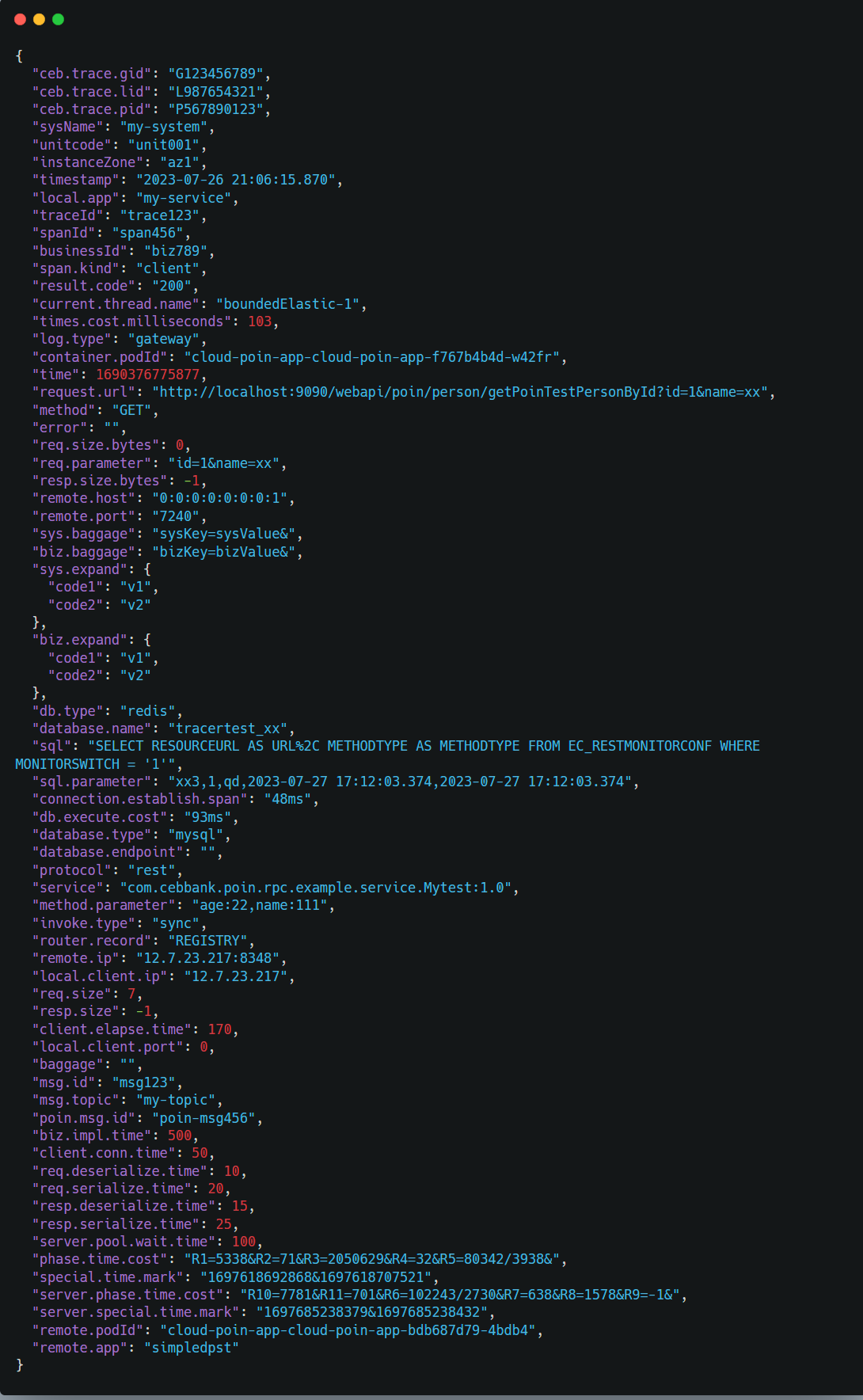


数据进入流水线之前，我们会根据db中的内容，为流水线配置

1. 过滤节点
2. 判断逻辑
3. 结束节点
4. 聚合计算节点

过滤节点：

我们会过滤掉脏数据，并将数据缩小如图



清洗成

判断节点：

流水线建立好后，会初始化判断逻辑，例如，交易码为AAAA的为正常数据，否则视为异常交易

结束节点：

数据流经这个节点，当未检测到头span时候，则视为该交易未结束，进程，会停滞在缓存中，指导超时或者检测到头span，则认为交易完成

**聚合计算节点（重要）：**

**聚合计算节点，指的是一些列用来处理流式数据的节点，流程如下**

**1** 按traceid先聚合一次数据

## 结构

[给出系统结构总体框图（包括软件、硬件结构框图），说明本系统的各模块的划分，扼要说明每个系统模块的标识符和功能，分层次地给出各模块之间的控制与被控制关系。]

## 功能需求与系统模块的关系

[本条用一张矩阵图说明各项功能需求的实现同各模块的分配关系。]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **[系统模块1]** | **[系统模块2]** | **[……]** | **[系统模块m]** |
| [功能需求1] | √ |  |  |  |
| [功能需求2] |  | √ |  |  |
| [┇] |  |  |  |  |
| [功能需求n] |  |  |  | √ |

## 人工处理过程

[说明在本系统的工作过程中不得不包含的人工处理过程。]

## 尚未解决的问题

[说明在概要设计过程中尚未解决而设计者认为在系统完成之前必须解决的各个问题。]

# 接口设计

## 用户接口

[说明将向用户提供的命令和它们的语法结构，以及相应的回答信息。

说明提供给用户操作的硬件控制面板的定义。]

## 外部接口

[说明本系统同外界的所有接口的安排包括软件与硬件之间的接口、本系统与各支持系统之间的接口关系。]

## 内部接口

[说明本系统之内的各个系统元素之间的接口的安排。]

# 运行设计

## 运行模块组合

[说明对系统施加不同的外界运行控制时所引起的各种不同的运行模块组合，说明每种运行所历经的内部模块的支持软件。]

## 运行控制

[说明每一种外界的运行控制的方式方法和操作步骤。]

## 运行时间

[说明每种运行模块组合将占用各种资源的时间。]

# 系统数据结构设计

[不涉及软件设计可不包含。]

## 逻辑结构设计要点

[给出本系统内软件所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。]

## 物理结构设计要点

[给出本系统内软件所使用的每个数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系、设计考虑和保密条件。]

## 数据结构与程序的关系

[说明各个数据结构与访问这些数据结构的各个程序之间的对应关系。]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **[程序1]** | **[程序2]** | **[……]** | **[程序m]** |
| [数据结构1] | √ |  |  |  |
| [数据结构2] |  | √ |  |  |
| [┇] |  |  |  |  |
| [数据结构n] |  |  |  | √ |

# 系统出错处理设计

## 出错信息

[用一览表的方式说明每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。]

## 补救措施

[说明故障出现后可能采取的变通措施。包括：

* 后备技术：说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术。
* 降效技术：说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录。
* 恢复及再启动技术：说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。]

## 系统维护设计