7

**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

默认交易维度

表格

描述已自动生成

和pod维度

表格

描述已自动生成

点击row，会展开二维查询

表格

描述已自动生成

2.1.1.4 热加载 详见概要设计

### 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：原始数据

文本

描述已自动生成

文本

描述已自动生成

输出(ElasticSearch 文档)

文本

描述已自动生成

此处，analysis 组件，会将数据写到特定结构图，并透传到tolc 的kafka。kafka的数据会由logstash 写入tolc 的ElasticSearch

ElasticSearch的dynamic mapping 能力，会将servicecode 等交易信息，自动映射到索引之上。tolc 的mapping，是默认keyword类型，这里需要提需求给tolc，保证数据是keyword + text 类型。

### 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

### 2.3.0 交易设置页面



这里用了trans\_type 而不是urlpath。根据分布式体系架构，urlpath 是交易的具体表象，但是，在实际交易中，不可避免地碰到交易码这个概念，链路侧需要兼容二者。故，以交易类型trans\_type，泛指url path ，交易码，中文描述。其中，中文描述，会继承edsp的能力，在cpaas中表象是urlpath 和交易码的中文。而二级注册中心，是依照配置解析出来的交易中文/说明，因此，链路采取全部在链路下可配置的思路，将交易的描述信息呢，存储在链路侧。

另外，基于cpaas自身作为容器平台，以及以应用为维度or第一视角的设计来看，这里采用了id自增作为主键。trans\_type + project\_id 作为唯一标示。project\_id 作为外键。而不是严格按照分布式运维体系的设计，交易全局唯一。交易本身在分布式应用体系下，尚不能完全统一为交易url 还是 交易代码，结合容器云的特性，故未采取trans\_type 作为全局唯一的设计。

设置页面基础数据结构如下  
 

Interval 交易监控间隔，单位为秒；

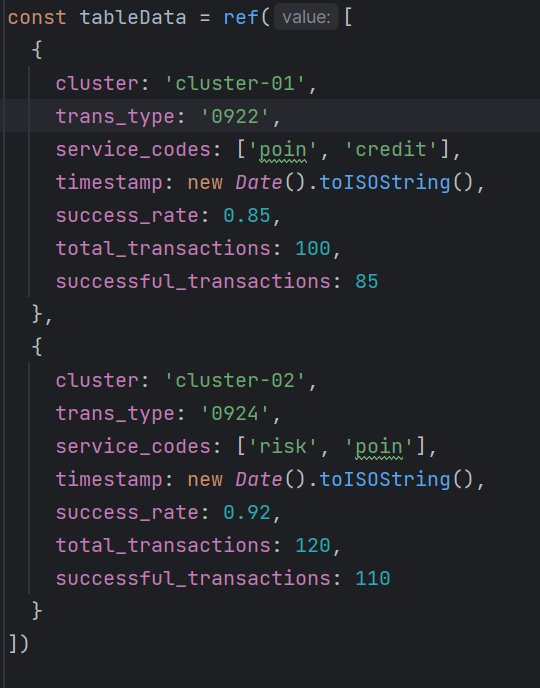
Podname：监控维度之一（可选项）

TransType：交易类型，大部分情况下是url path

RetCode（servicecode，取poin实际上收的名称）：交易返回码，对应的是poin的ServiceCode。当交易的ServiceCode in RetCode时候，视作成功交易。

Dimension1/2 是该项目下的自定义维度。

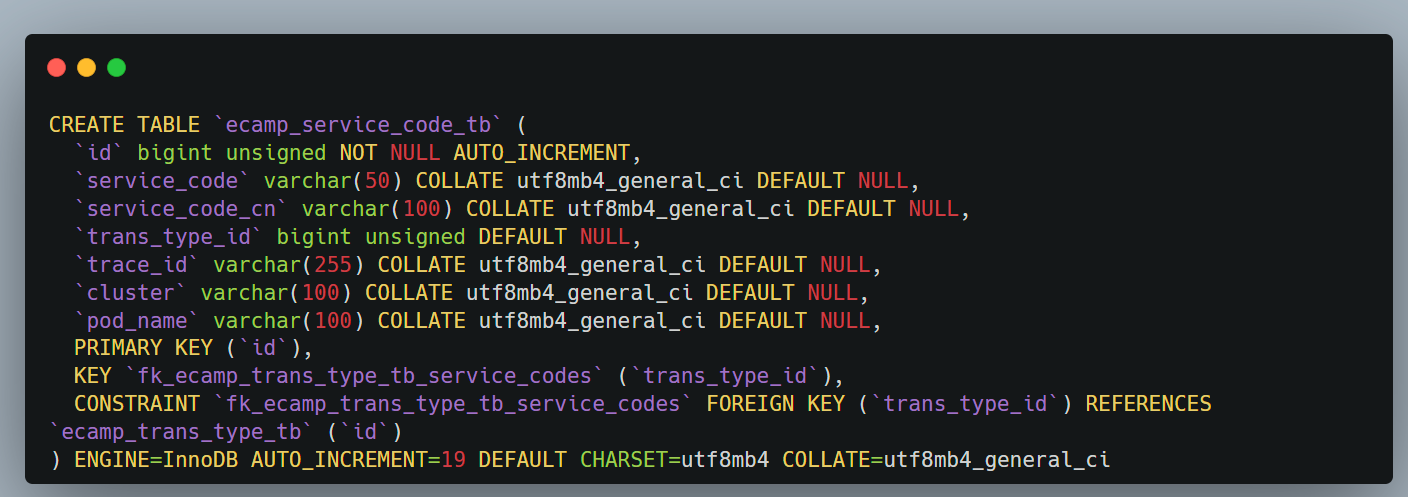
区别于响应质量监控，这里的信息不会体现出trace和span；即交易和响应在设计上是分开的。

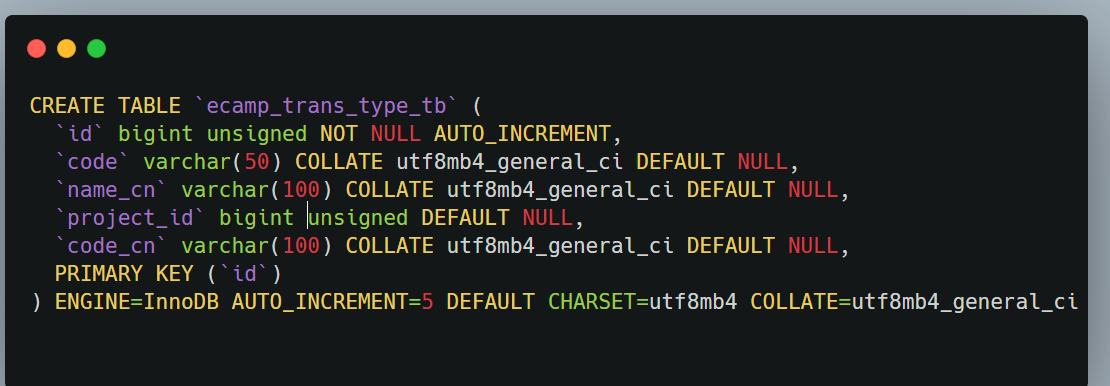


一个项目有多个交易，每个交易又有多个交易码。其中三者表结构如图 ecamp\_project\_info\_tb（项目表）

 ecamp\_trans\_type\_tb（交易类型表） 新增

 ecamp\_service\_code\_tb（交易码表） 新增

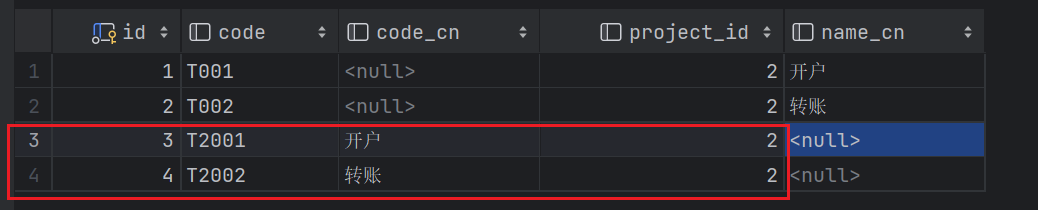




交易码不允许单独存在

其他：查询使用使用

如图：下面设置好RetCode 等内容，并持久化到db 中。



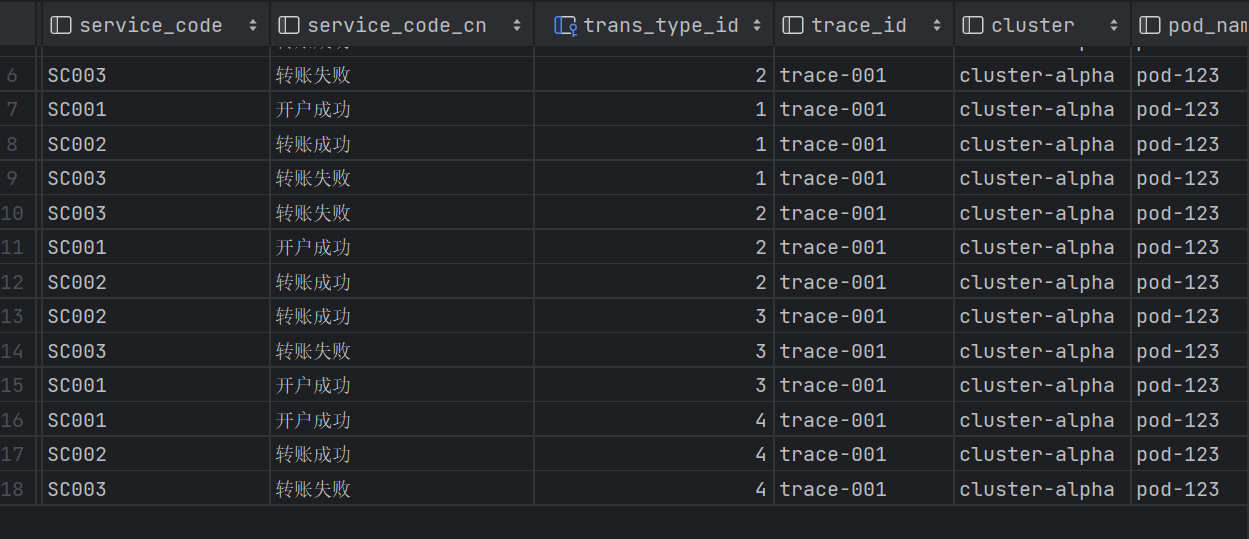
数据结构如图

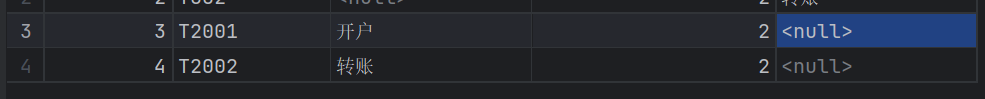
配置好后，视作开启交易质量展示；；

但请求输入时候，例如



会分别在交易类型表和交易码表新建数据如图





开户交易存在两种交易码分别是SC003,SC001 作为成功失败的码

该数据为poin新加入字段，如下



当文档中有trans\_info时候，无论业务如何，均视为开启交易质量展示。

**非功能期望**

期望记录数量为svcname \* 交易类型 \* cluster；目前评估无需联合索引。如果有需要，后面专门提出非功能优化，增加索引。

##### 查询流程

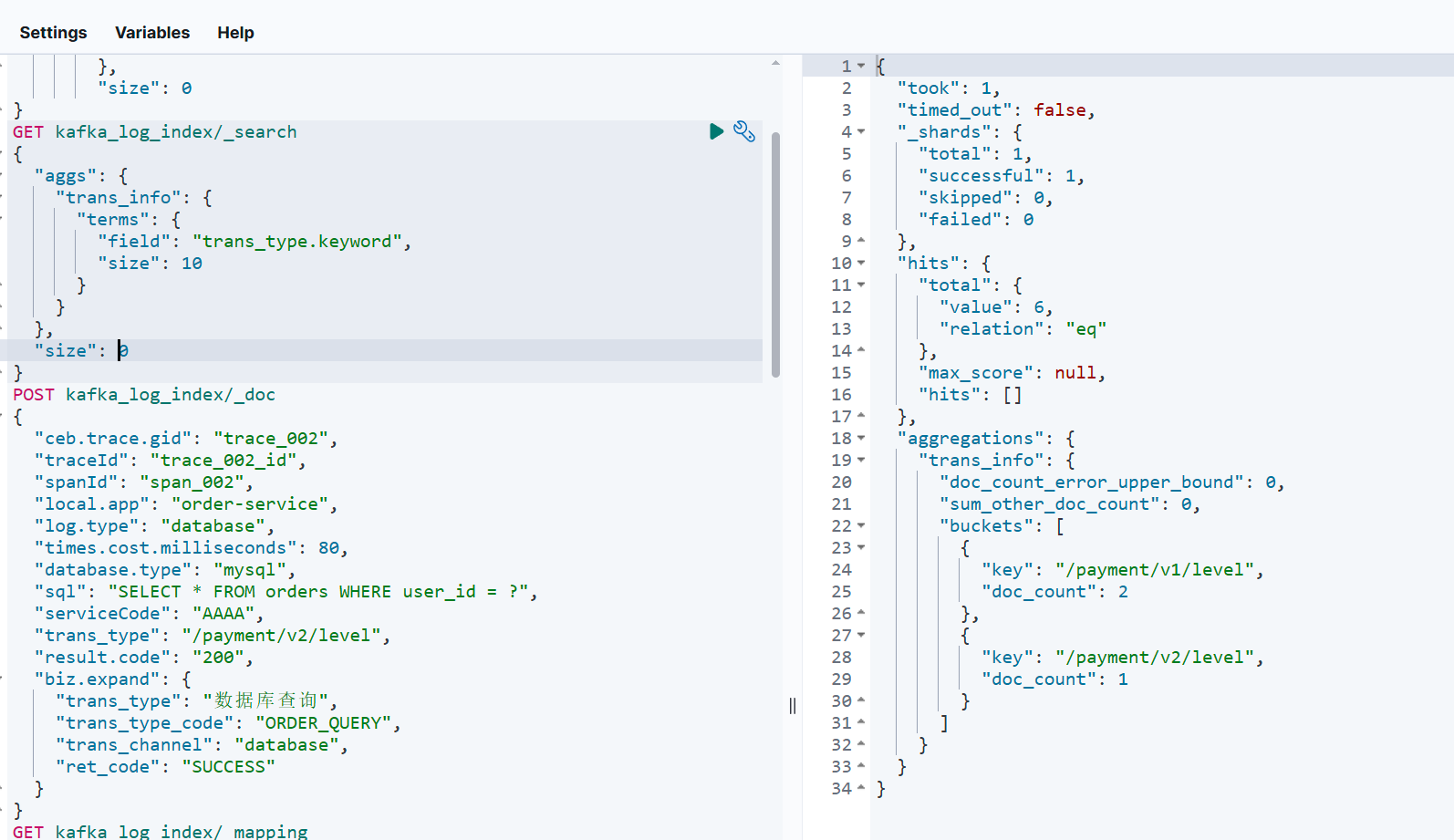
数据流概要如下

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

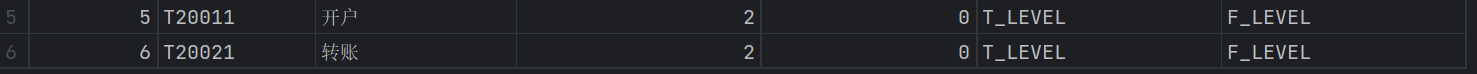
原始数据会被analysis 处理成含有交易信息的ElasticSearch doc。并存放至科技运营平台，简称tolc。

第一次进入该页面会先一步查询ElasticSearch 文档，聚合出所有的交易信息如图



其中拿到了交易中的url和交易总数。

之后，去db中查询对应的交易中文描述



当配置好交易质量展示后，ElasticSearch的索引mapping中，会增加额外信息，如图  
trans\_info:

地图

描述已自动生成部分mapping

图形用户界面, 应用程序

描述已自动生成

当首次进入页面时候，默认会先去es中去和出三天内所有的交易信息，如图

其中，trans\_type， cn 等，分别为清洗过后的数据，具体名称可能会按照实际情况更改名称，但字段总数和含义不变。

清洗规则如下

利用es dynamic mapping的能力，交易数据，会在trans这个properties之下。首次进入，会按照三天内的交易聚合出来，做为项目下的交易设置初始页面。使用者会在上面编辑。

文本, 信件

描述已自动生成

一笔交易有多个状态码。状态码必须要绑定交易和项目信息。

api 创建和修改如下

删除流程。考虑到交易码和交易相关且仅仅在当前项目下生效。故将交易码和交易绑定，而不是项目。因此，删除或者是修改某个交易设置，要同步删除其交易码信息。

注意，mysql中存储的是交易的设置信息。原始数据存储在ElasticSearch 中，无法修改。

其中交易的三种状态：

图示

描述已自动生成

当用户点击保存时候，将会将交易明细存储在db中。

当设置为开启交易质量监控时，则视为将当前交易数据发送至oda。

#### 交易概览页

ElasticSearch 索引mapping 已经文档如下

如图地图

描述已自动生成，数据写入后，会将servicecode，等数据，作为keyword 添加至ElasticSearch 的mapping中

页面加载后，会将存在的字段聚合出来，biz.expend 和 上述固定字段。其中固定字段为交易信息，poin提供。pod，容器维度，为链路侧提供。来源为analysis组件在处理时候加上。biz.expend 为自定义维度。里面的内容，会被认为是交易信息。如果该内容，超过ElasticSearch最大限额，导致查询失败或者不符合预期，需要配置人员自行解决。todo

当选定维度后，下面内容，为某个维度的聚合信息。例如，下图的语义为，对账信息二级中，不同pod的交易成功率为50%。

除了交易成功率之外，还会提供如下指标，计算公式为

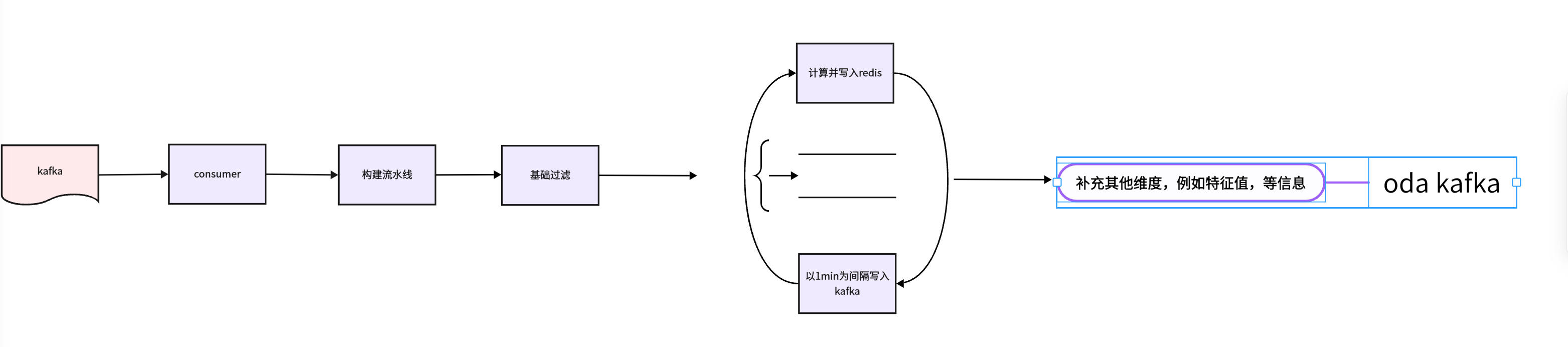
###### 页面指标计算

|  |  |  |
| --- | --- | --- |
| 指标 | 计算公式 | 是否涉及告警 |
| 成功率 | 交易码为指定的数据总数/所有交易数据总数 | 是 |
| 成功数 | 交易码存在且为AAAA等指定的数量 | 是 |
| 失败率 | 交易码不为指定的数据总数/所有交易数据总数 | 是 |
| 失败数 | 交易码存在且不为AAAA等指定的数量 | 是 |
| 响应数 |  | 否 |
| 响应时间 | 某个交易的头节点的响应时间的平均值 | 否 |
| 健康度 |  | 否 |
| 成功率5min | 略 | 否下同 |
| 成功数5min |  |  |
| 失败率5min |  |  |
| 失败数5min |  |  |
| 响应数5min |  |  |
| 响应时间5min |  |  |
| 健康度5min |  |  |
| 交易抵达（todo） |  |  |

其中展示内容，均为从ElasticSearch 查询出来，有一定的延迟。

其中交易成功率查询语句如下

### 热加载

总数据流图如下  


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

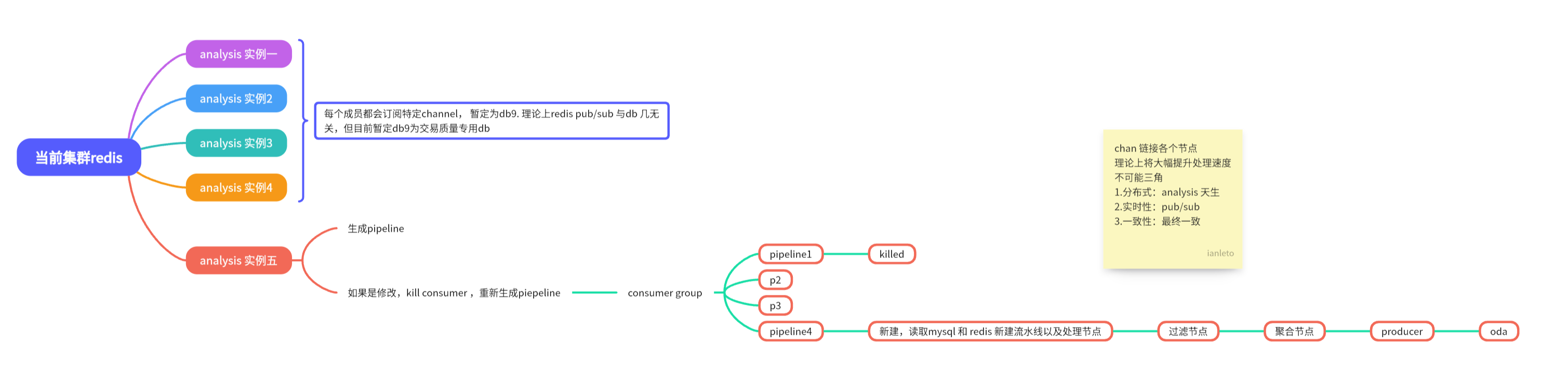
描述已自动生成

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**

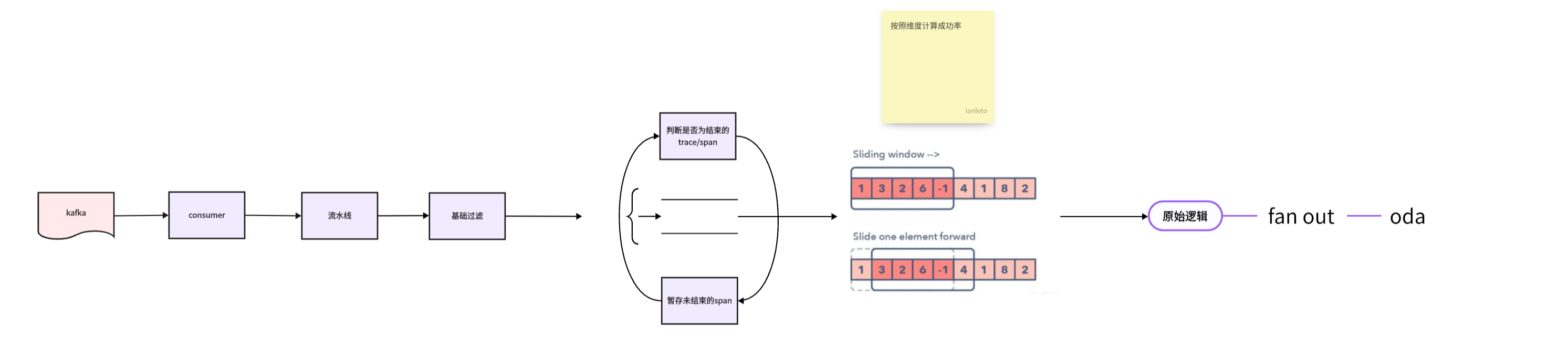
当所有订阅者拿到消息后，会开始分析，最终消费该项目数据的实例生成、或者kill掉自己的pipeline（数据流水线，一个抽象的，自己实现流式处理机制）如图实例五



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

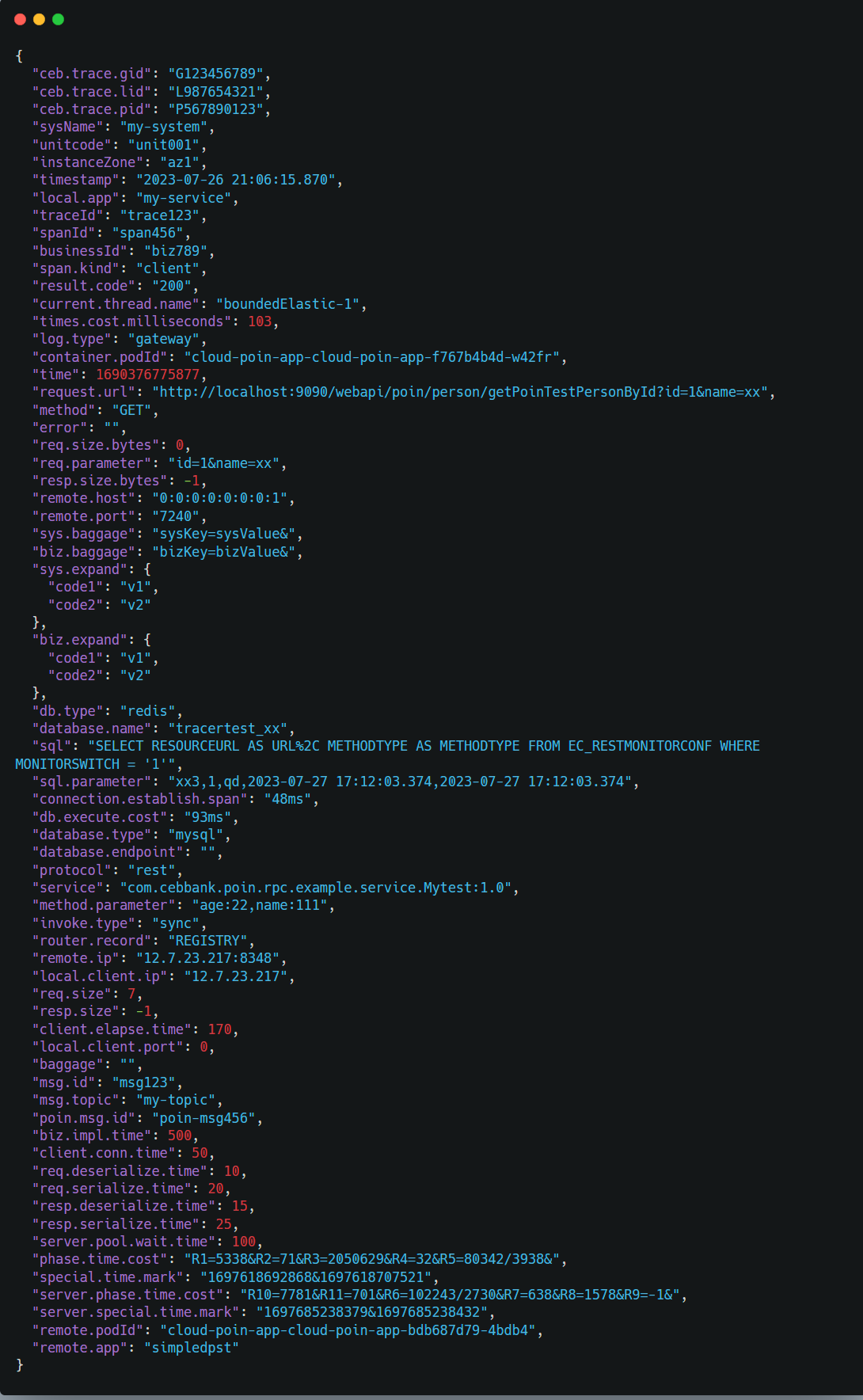


数据进入流水线之前，我们会根据db中的内容，为流水线配置

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

### 页面指标计算

|  |  |  |
| --- | --- | --- |
| 指标 | 计算公式 | 是否涉及告警 |
| 成功率 | 交易码为指定的数据总数/所有交易数据总数 | 是 |
| 成功数 | 交易码存在且为AAAA等指定的数量 | 是 |
| 失败率 | 交易码不为指定的数据总数/所有交易数据总数 | 是 |
| 失败数 | 交易码存在且不为AAAA等指定的数量 | 是 |
| 响应数 |  | 否 |
| 响应时间 | 某个交易的头节点的响应时间的平均值 | 否 |
| 健康度 |  | 否 |
| 成功率5min | 略 | 否下同 |
| 成功数5min |  |  |
| 失败率5min |  |  |
| 失败数5min |  |  |
| 响应数5min |  |  |
| 响应时间5min |  |  |
| 健康度5min |  |  |

## 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